MODULE 5

COMBINATIONAL LOGIC

Course Material for Combinational

Extra: Slides about how to implement a static combinational gate with NMOS/PMOS transistors, given the Boolean function

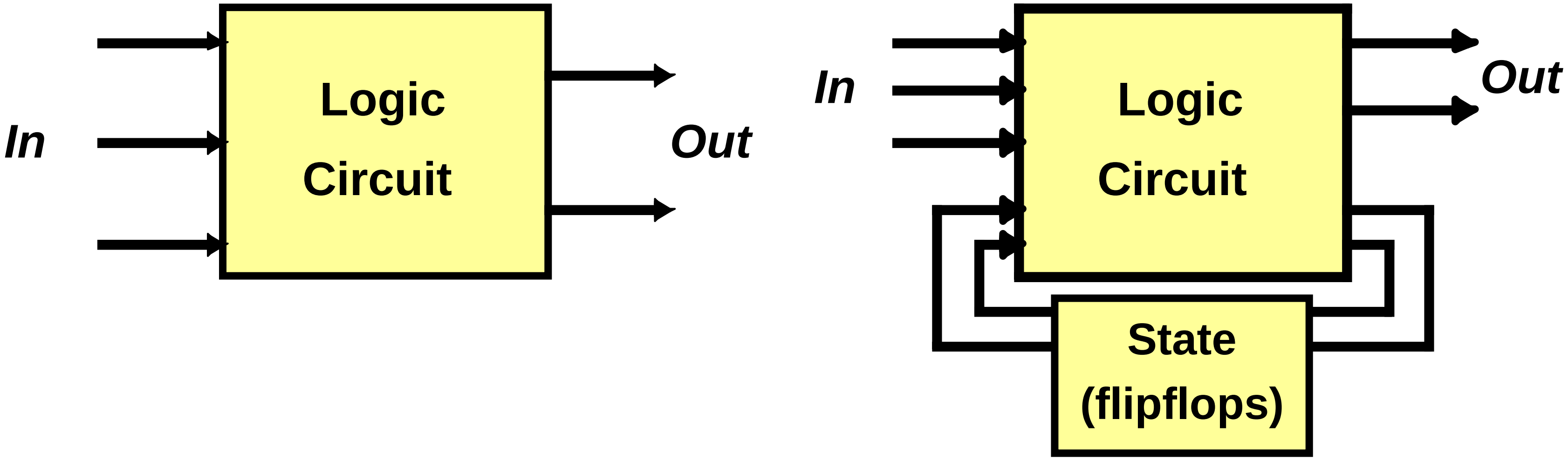
P	6.1	Introduction	236
P	6.2	Static CMOS Design	236 – 237
P	6.2.1	Complementary CMOS	237 – 242
I		Propagation Delay of Complementary CMOS gates	242 – 249
I		Design Techniques for large fan-in	249 – 251
O		Optimizing performance in combinational networks	251 – 257
O		Power consumption in CMOS logic gates	257 – 263
P	6.2.2	Ratioed Logic	263 – 267
I		How to build even better loads	267 – 268
P	6.2.3	Pass-transistor basics	269 – 270
I		Example 6.10	271 – 272
O		Diversen	272 – 277
P		Solution 3: Transmission gate logic	277 – 280
I		Rest of § 6.2.3	280 – 284
I	6.3	Dynamic CMOS Design	
I	6.3.1	Dynamic Logic: Basic Principles	284 – 286
I	6.3.2	Speed and Power Dissipation of Dynamic Logic	287 – 290
I	6.3.3	Signal Integrity Issues in Dynamic Design	290 – 295
O	6.3.4	Cascading Dynamic Gates	295 – 303
O	6.4	Perspectives	303 – 306
P	6.5	Summary	306 – 307

Combinational Logic - Outline

- **Conventional Static CMOS basic principles**
- **Complementary static CMOS**
  - **Complex Logic Gates**
  - **VTC, Delay and Sizing**
- **Ratioed logic**
- **Pass transistor logic**
- **Dynamic CMOS gates → only illustration**

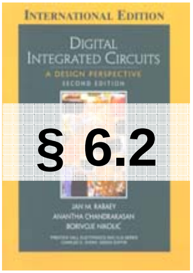
Complementary Static CMOS
Basic Principles

Combinational vs. Sequential Logic



(a) Combinational

(b) Sequential



Output = *f* (In)

Output = *f* (In, History)

Reminder

DeMorgan Transformations

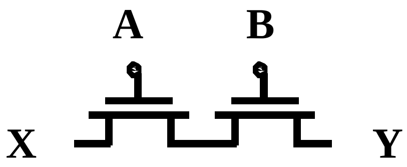
$$\overline{A+B}=\overline{A}.\overline{B}$$

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

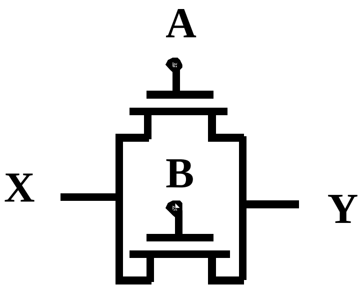
NMOS Transistors in Series/Parallel Connection

Transistors can be thought as a switch controlled by its gate signal

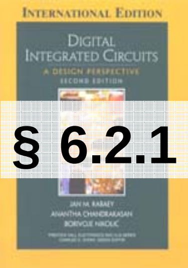
NMOS switch closes when switch control input is high



Y = X if A and B



Y = X if A or B

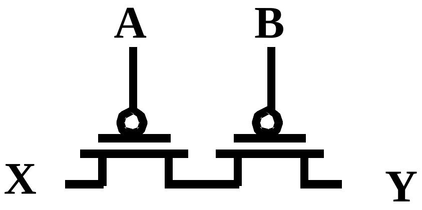

INTERNATIONAL EDITION
DIGITAL INTEGRATED CIRCUITS
A DESIGN PERSPECTIVE
SECOND EDITION
JAN M. RABAEY
ANANTHA CHANDRAKASAN
BORIVOJE NIKOLIĆ

§ 6.2.1

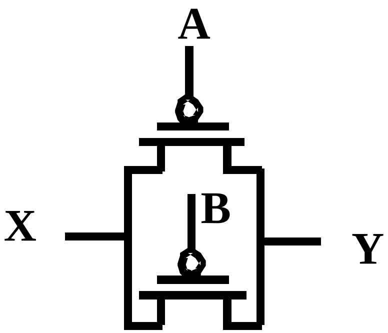
PMOS Transistors in Series/Parallel Connection

PMOS switch closes when switch control input is low

Y = X if …

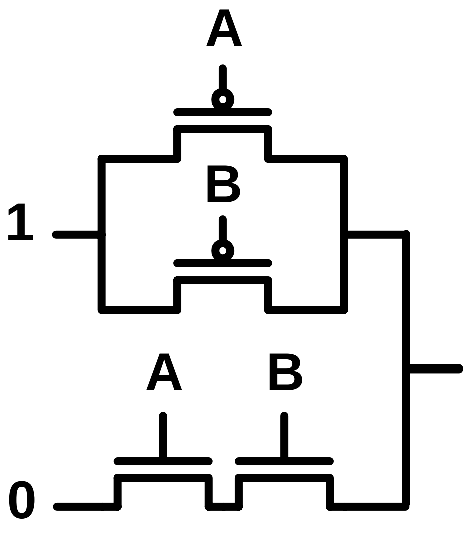


Y = X if $\overline{A}$ and $\overline{B}$



Y = X if $\overline{A}$ or $\overline{B}$

2-Input Nand



$$\left.\begin{array}{l} Y = 1\ \mathit{if}\ \overline{A}\ \mathit{OR}\ \overline{B} \\ Y = 1\ \mathit{if}\ \overline{A\ \mathit{AND}\ B} \end{array}\right\}\ \text{DeMorgan}$$

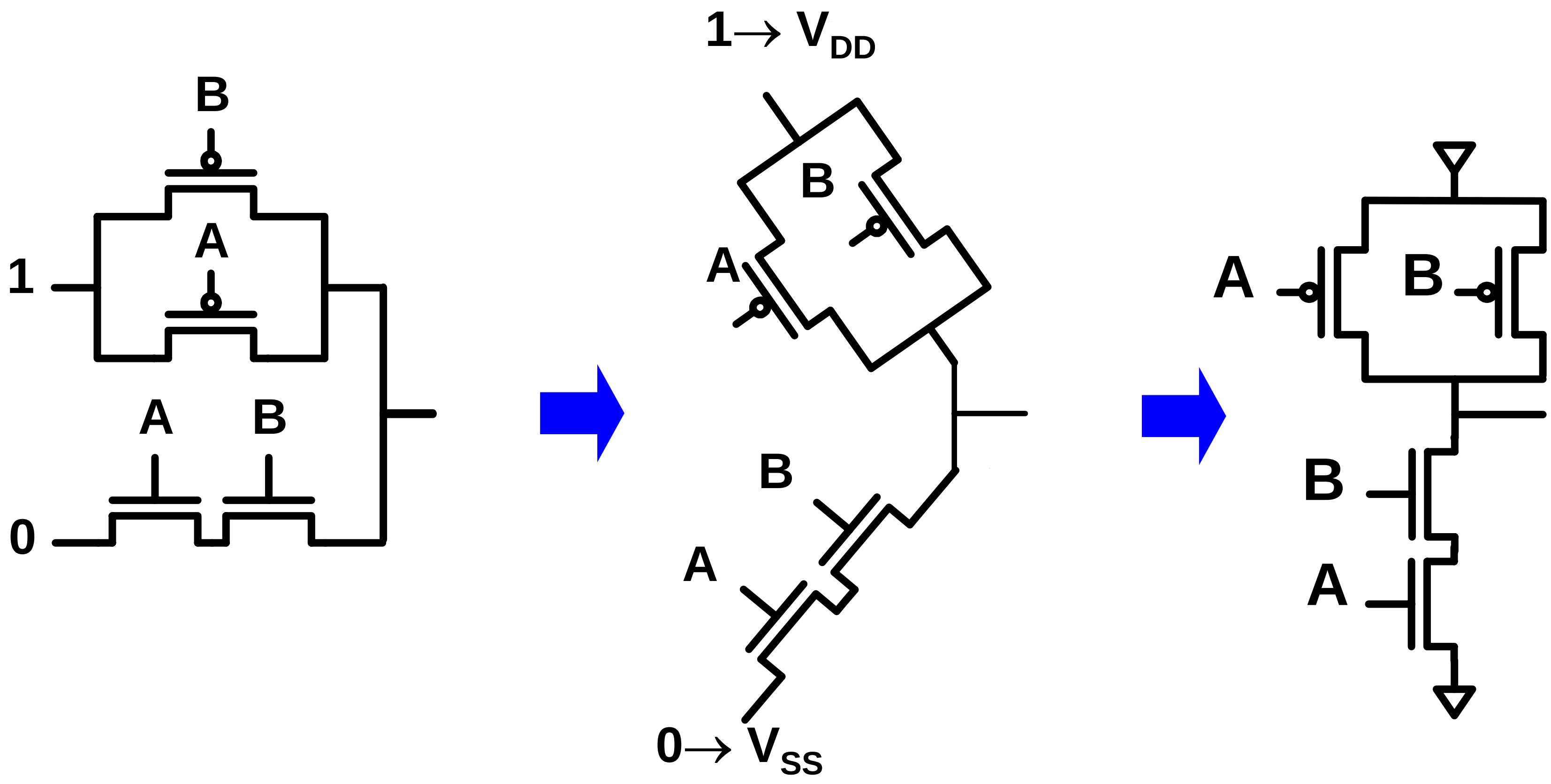
$$Y = \overline{A\ \mathit{AND}\ B}$$

$$Y = 0\ \mathit{if}\ A\ \mathit{AND}\ B$$

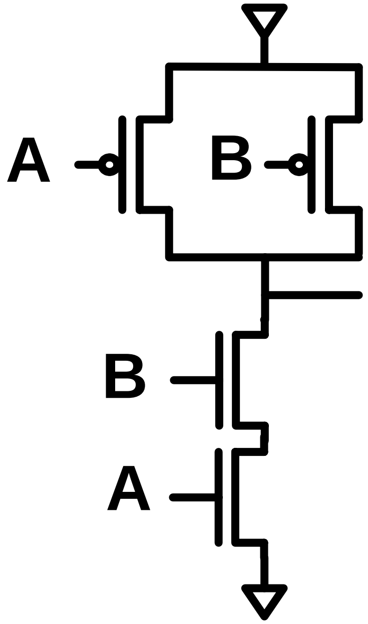
B	A	Y
0	0	1
0	1	1
1	0	1
1	1	0

2-Input Nand

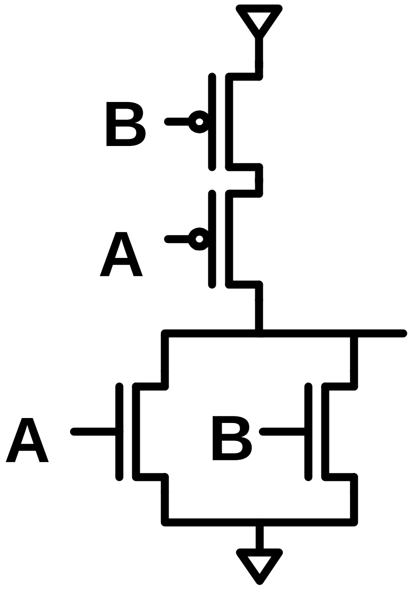
$$Y = \overline{A\ AND\ B}$$



2-input Nand/Nor

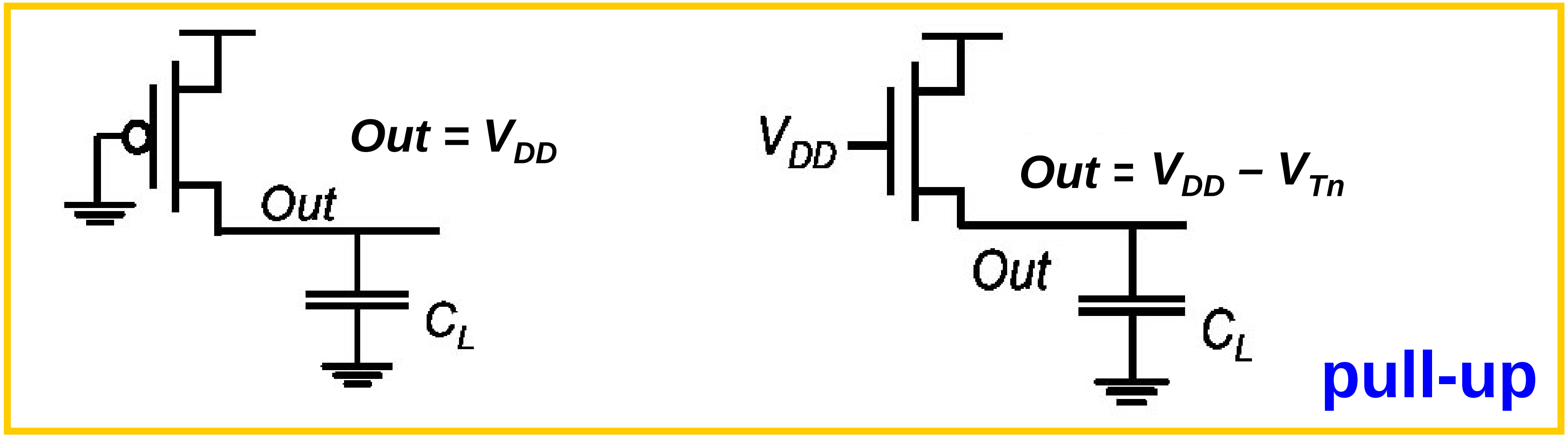


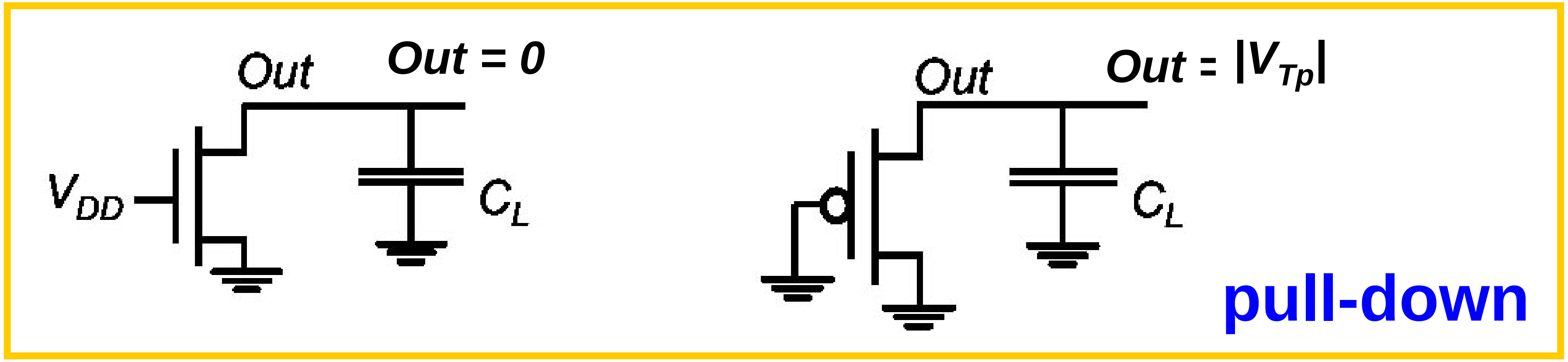
$Y = \overline{\mathbf{A}\ \text{AND}\ \mathbf{B}}$



$Y = \overline{\mathbf{A}\ \text{OR}\ \mathbf{B}}$

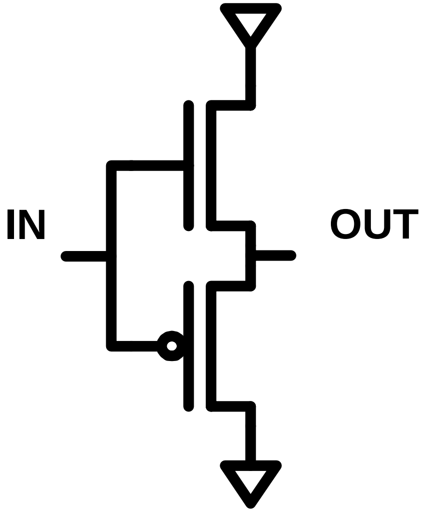
NMOS vs. PMOS, pull-down vs. pull-up





- **PMOS is better pull-up**
- **NMOS is better pull-down**

Bad Idea



Exercise: **Determine logic function**

Determine V_{out}

for $V_{in} = V_{DD}$ and $V_{in} = V_{SS}$

Why is this a bad circuit?

CMOS Gate is Inverting

Assume full-swing inputs (high = V_{DD}, low = V_{SS})

- Highest output voltage of NMOS is

  $V_{GS} - V_{Tn} = V_{DD} - V_{Tn}$
- An 1 on NMOS gate can produce a strong 0 at the drain, but not a strong 1
- Lowest output voltage of PMOS is

  $V_{DD} + V_{GS} - V_{Tp} = |V_{Tp}|$
  (with V_{GS}, $V_{Tp} < 0$ for PMOS)
- An 0 on PMOS gate can produce a strong 1 at the drain, but not a strong 0
- Need NMOS for pull-down, PMOS for pull-up

  A 1 at input can pull-down, 0 at input can pull-up
  A 1 can produce a 0, a 0 can produce a 1

Inverting behavior

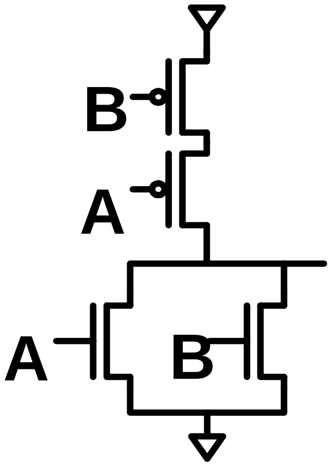
For a non-inverting Complementary CMOS Gate, you can only use 2 inverting gates

Complementary static CMOS

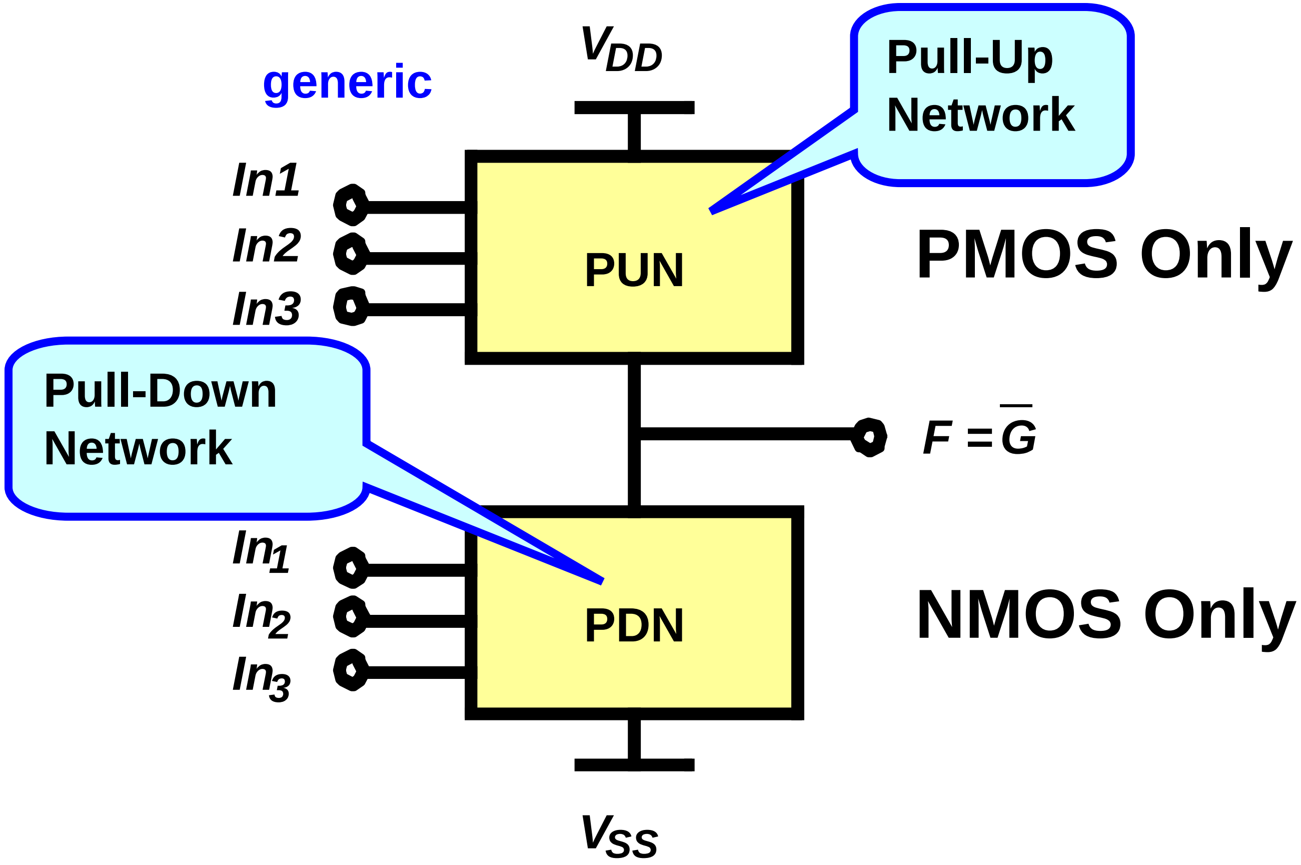
- Complex Logic Gates
- VTC, Delay and Sizing

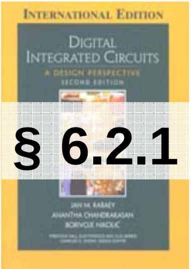
Complementary Static CMOS

example



generic



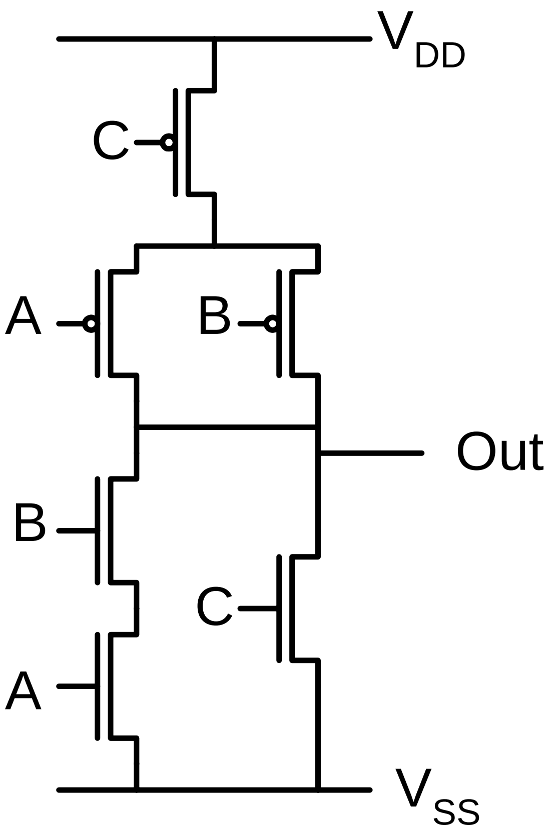


§ 6.2.1

- **Conduction of PDN and PUN must be mutually exclusive (Why?)**
- **Pull-up network (PUN) and pull-down network (PDN) are dual**

Mutual Exclusive PDN and PUN

Out = (AB + C)'



C	B	A	PDN	PUN	Out
0	0	0	?	1	1
0	0	1	?	1	1
0	1	0	?	1	1
0	1	1	0	?	0
1	0	0	0	?	0
1	0	1	0	?	0
1	1	0	0	?	0
1	1	1	0	?	0

PDN Off
PUN On

PUN Off
PDN On

For all Complementary Static CMOS Gates, either the PUN or the PDN is conducting, but never both.

Complementary Static CMOS (2)

- **Conduction of PUN and PDN must be mutually exclusive**
- **PUN is dual (complement) network of PDN**

  series ⇔ parallel

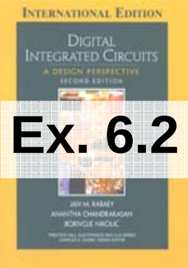
  nmos ⇔ pmos
- **Complementary gate is inverting**
- **No static power dissipation**
- **Need 2N transistors for N-input gate**

Implementation of Combinational Logic

- **How van we construct an arbitrary combinational logic network in general, using NMOS and PMOS transistors (using Complementary static CMOS)?**

- **Example:** $Y = \overline{(A + BC)D}$

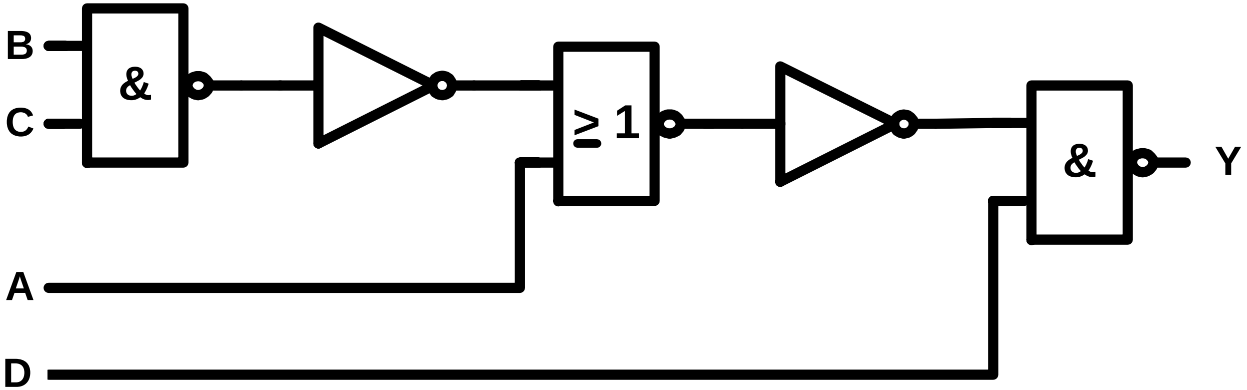
- **Remember: only inverting gates available**



Ex. 6.2

Implementation of Combinational Logic

- Example: $Y = \overline{(A + BC)D}$
- Remember: only inverting gates available
- Logic depth: number of gates in longest path ⇨ DELAY

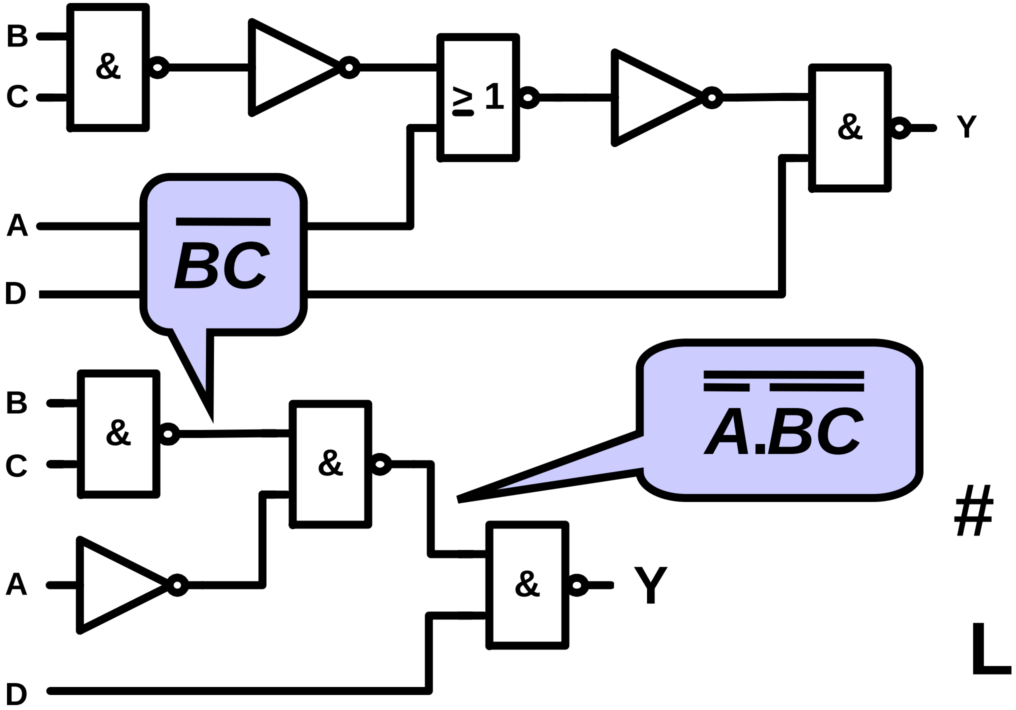


transistors □ logic depth □

- Q: Can this be improved?

Improved Gate Level Implementation

- **Using DeMorgan** $A + BC = \overline{\overline{A}.\overline{BC}}$



$Y = \overline{(A + BC)D}$

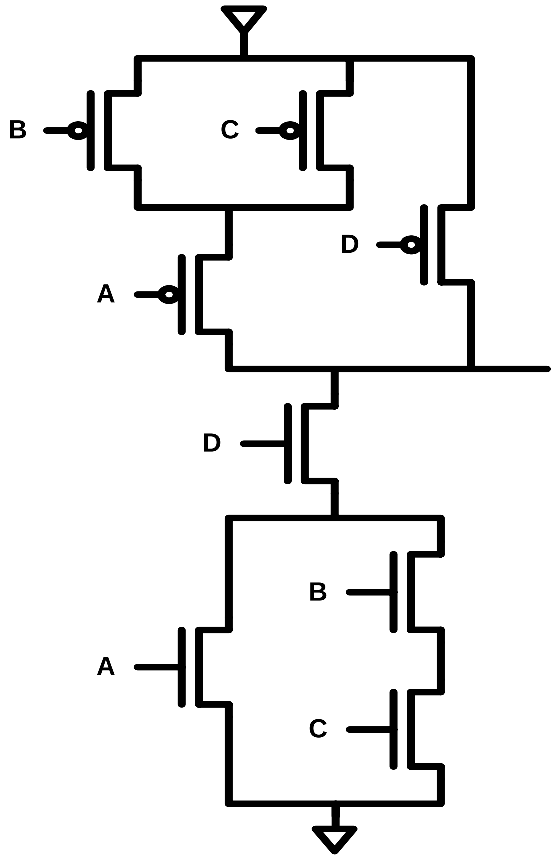
# transistors □

Logic depth □

- **Q:** **Can this be further improved?**

Complex CMOS Logic Gates

- Restriction to basic NAND, NOR etc. not necessary
- Easy to synthesize complex gates



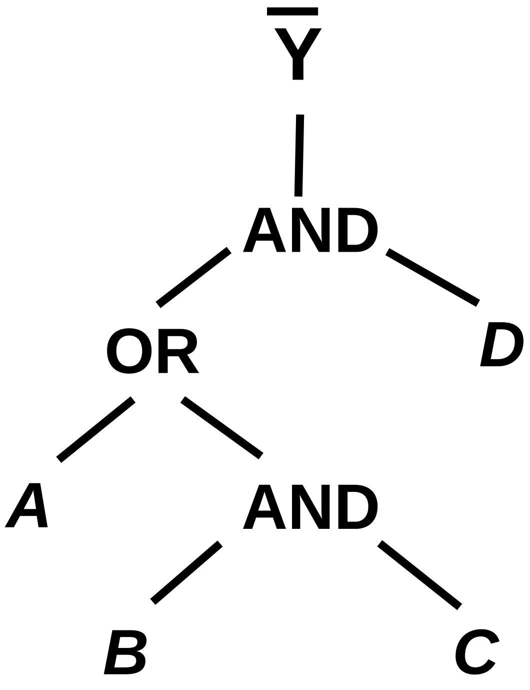
$Y = \overline{(A + BC)D}$

# transistors: 8

Logic depth: 1

How to Synthesize Complex Gates

$Y = \overline{(A + BC)D}$

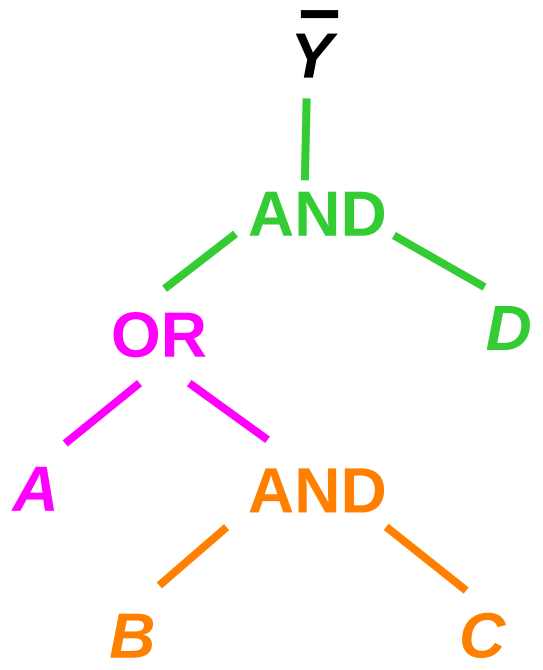


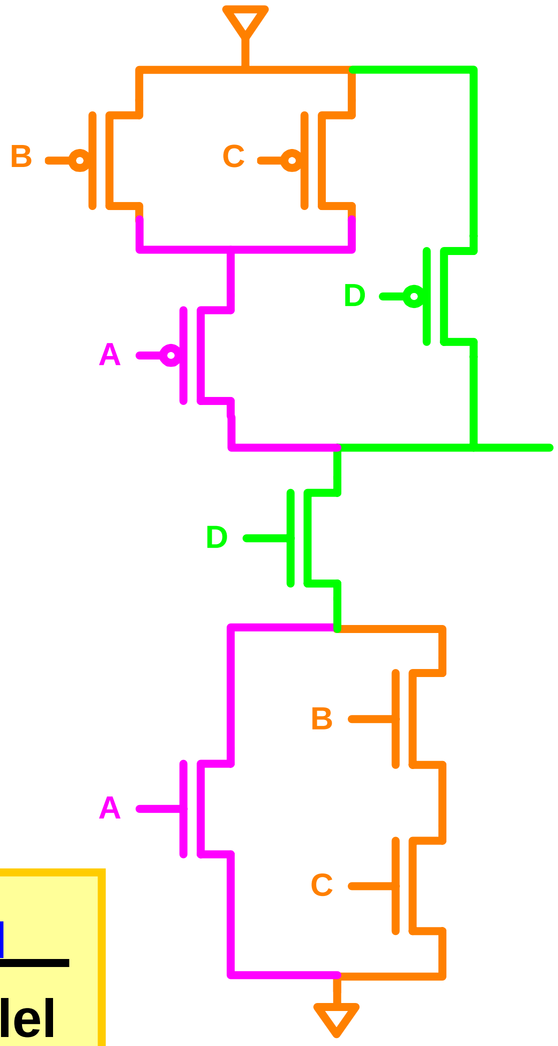
- Using tree representation of Boolean function
- ***Operator*** with branches for ***operands***
- As a **series-parallel** network

	PDN	PUN
AND	Series	Parallel
OR	Parallel	Series

Complex Gate Synthesis Example

$$\overline{Y} = (A + (BC))D$$



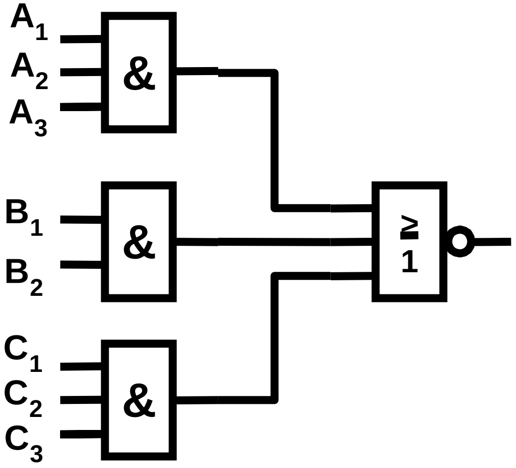


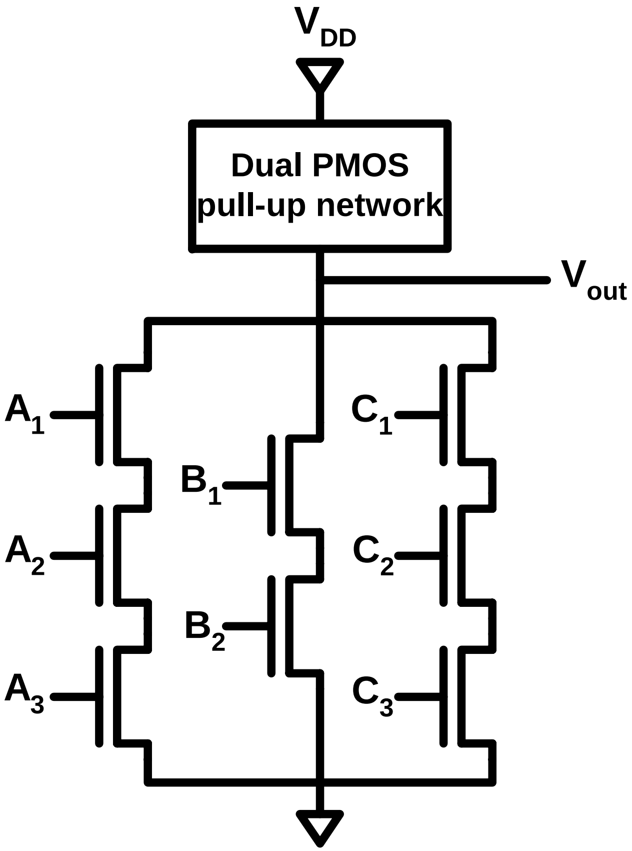
	PDN	PUN
AND	Series	Parallel
OR	Parallel	Series

Recipe

- Write $\overline{Y}$ = f(inputs)
- Decompose f in tree form
- Realize tree branches according to table at bottom-left
- Use inverted inputs if necessary

And-Or-Invert Gate

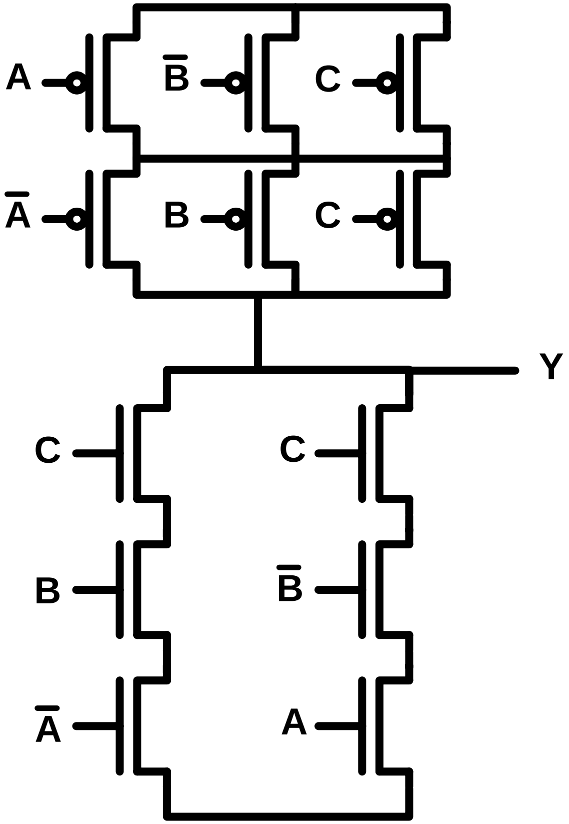




And-Or-Invert Example

- **From a Truth-Table: take 0-outputs**

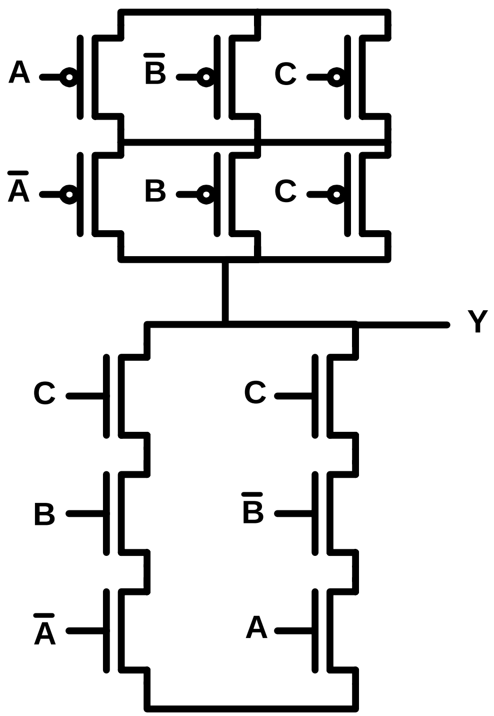
A	B	C	Y	
0	0	0	1	
0	0	1	1	
0	1	0	1	
0	1	1	0	→ $\bar{A}BC$
1	0	0	1	
1	0	1	0	→ $A\bar{B}C$
1	1	0	1	
1	1	1	1	



$$\bar{Y} = \bar{A}BC + A\bar{B}C$$

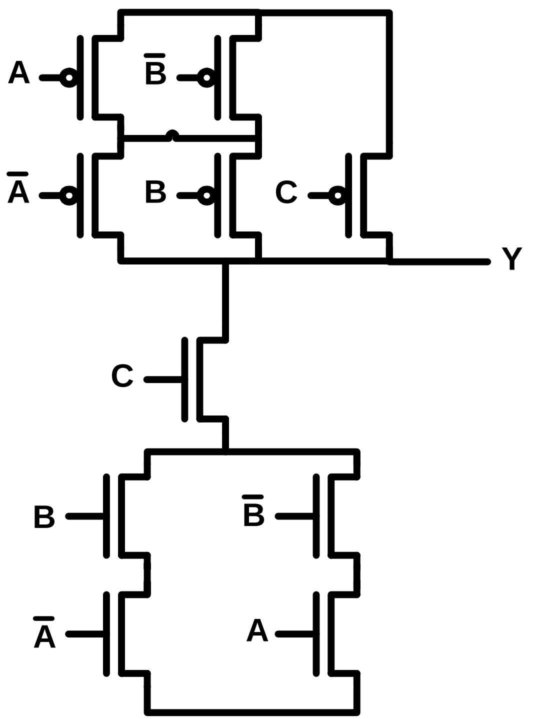
$\bar{A}$, $\bar{B}$ to be created with extra inverters (or by restructuring previous circuits)

And-Or-Invert Improvement



$$Y = \overline{\overline{A}BC + A\overline{B}C}$$

12 transistors

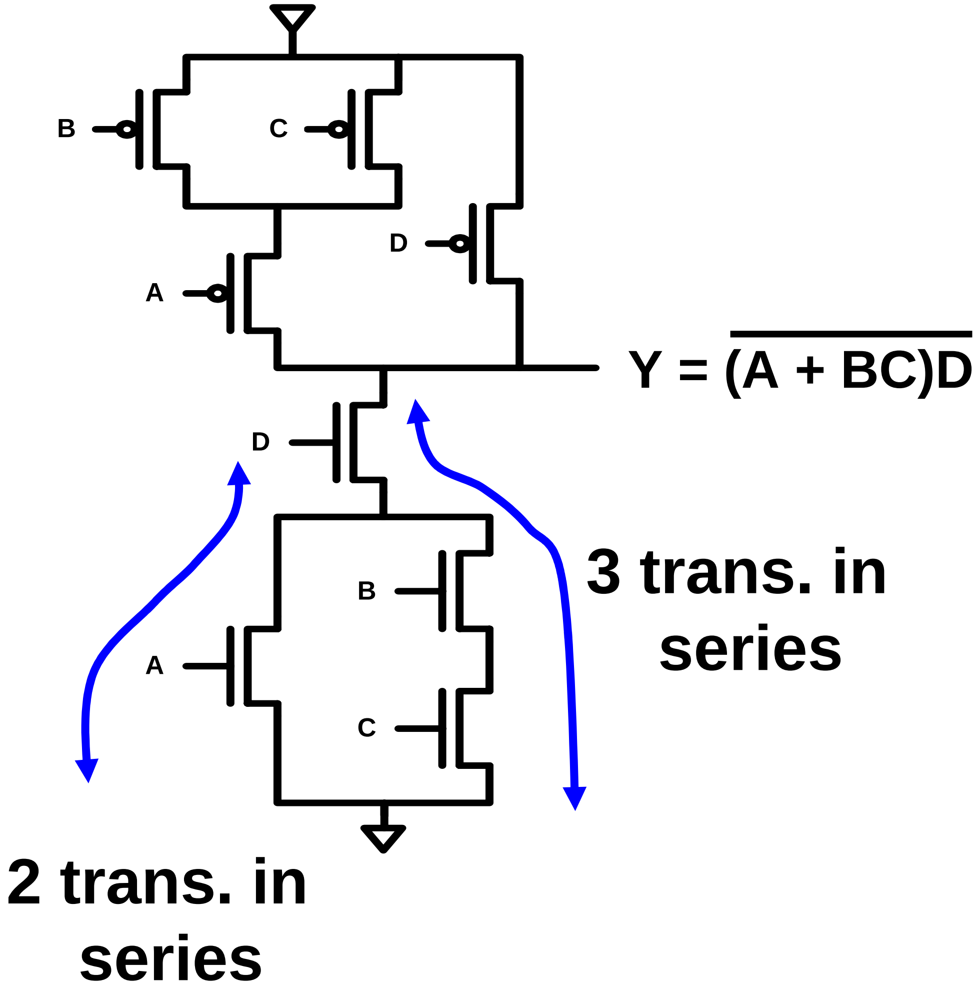


$$Y = \overline{(\overline{A}B + A\overline{B})C}$$

10 transistors

2-level logic minimization: see Katz (CS1), § 2.3

CMOS Complex Gate Sizing



$Y = \overline{(A + BC)D}$

3 trans. in series

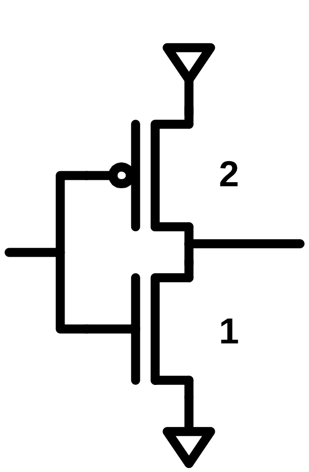
2 trans. in series

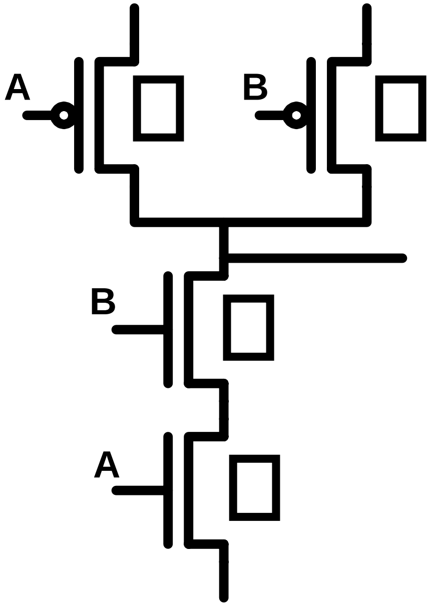
- **Function of gate independent of transistor sizes: ratio-less**
- **But current-drive capability depends on transistor sizes**
- **Worst-case current-drive depends on number of transistors in series**

CMOS Complex Gate Sizing

- **Assume all transistors will have mininum length L**
- **Determine W_n for PDN transistor of inverter that would give the desired 'drive strength'**
- **For each transistor in PDN of complex gate do the following:**
  - **Determine the length l of the longest PDN chain in which it participates**
  - **Set $W = l\ W_n$**
- **Repeat this procedure for PUN, using W_p for PUN transistor of inverter.**

Gate Sizing

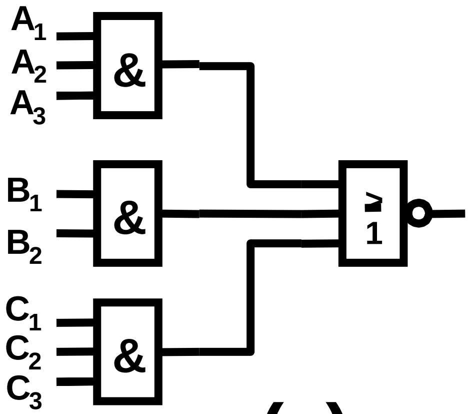




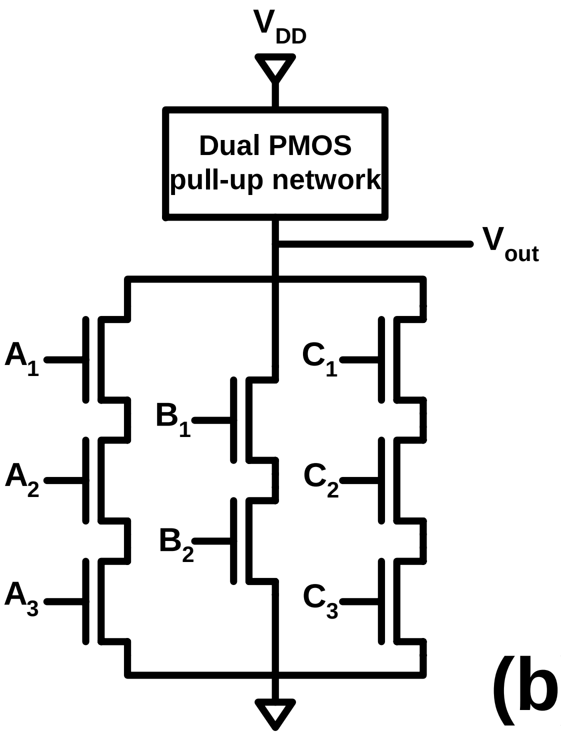
- **W/L ratios**
- **what are the W/L of 2-input NAND for same drive strength?**

0-th order calculation

Exercise



(a)

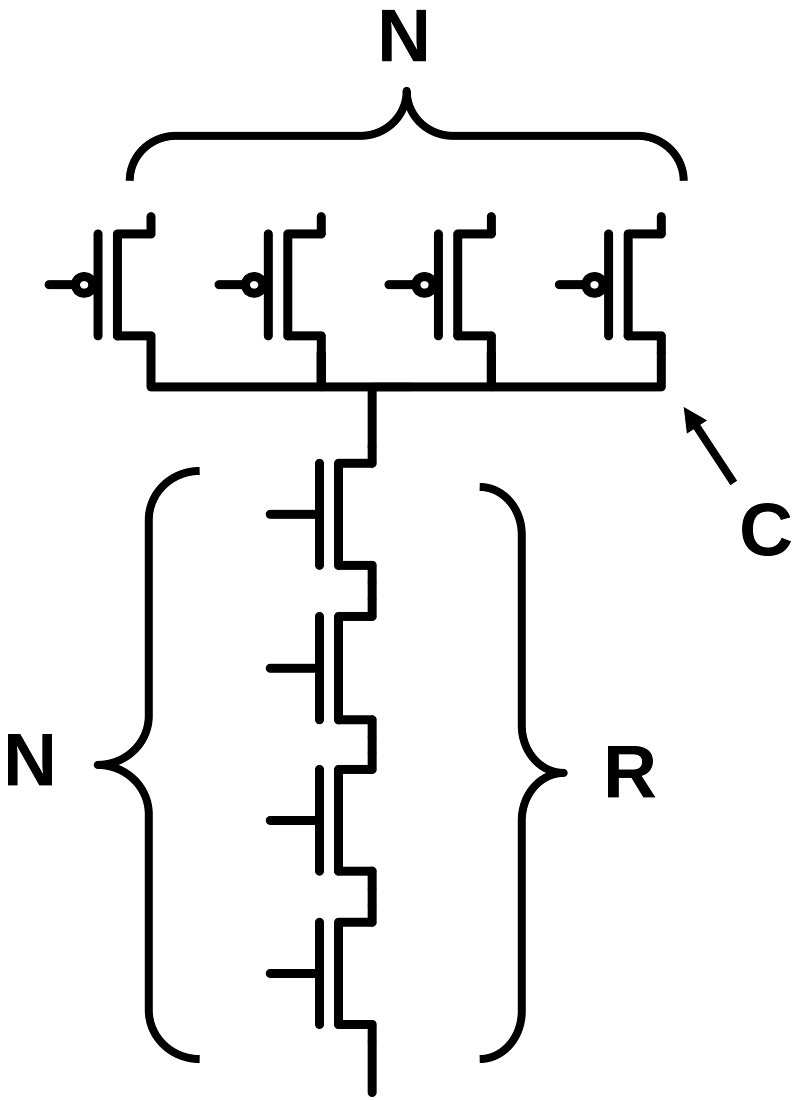


(b)

Exercise:

- **Perform gate sizing of (a) for nominal drive strength equal to that of min size inverter, assume PU/PD = 3**
- **Determine PUN of (b)**
- **Perform gate sizing of (b) for same drive strength (same PU/PD)**
- **Compare sum of gate areas in (a) and (b). Note: area ~ width**

Avoid Large Fan-In



C linear in N

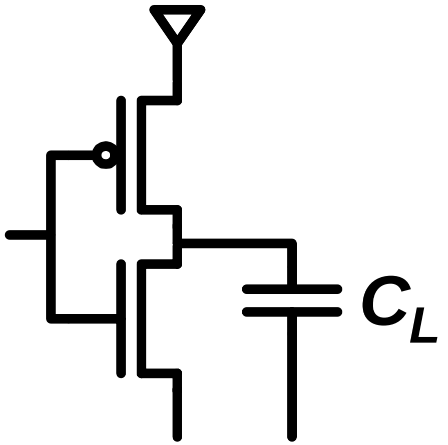
R linear in N

Delay $\propto$ RC quadratic in N

Empirical

Delay = a_1FI + a_2FI^2 + a_3FO

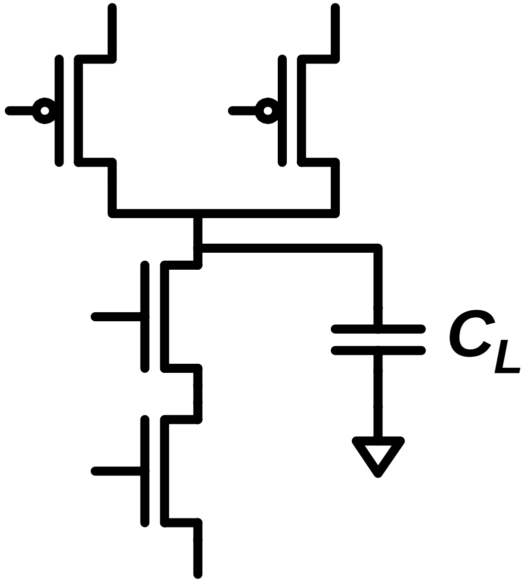
Data-Dependent Timing



$$t_{PHL} = 0.69 R_N C_L$$

$$t_{PLH} = 0.69 R_P C_L$$

You should be able to identify the transistor paths that charge or discharge C_L, and calculate resulting RC delay model, including effects of wires and fan-out



$$t_{PHL} = 0.69(R_N \times 2)C_L$$

Series connection

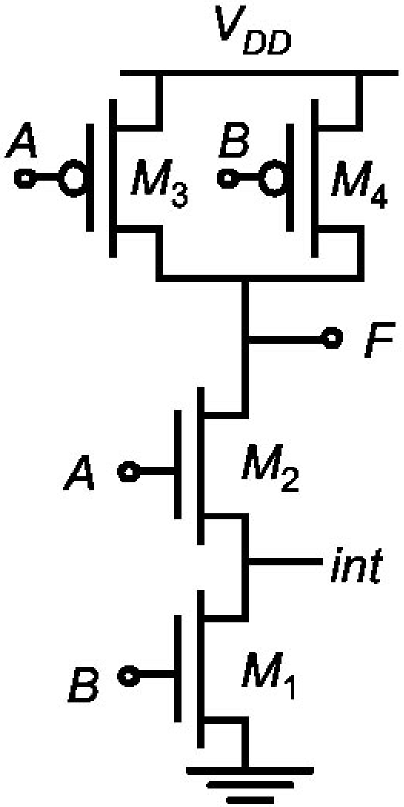
$$t_{PLH} = 0.69 R_p C_L$$

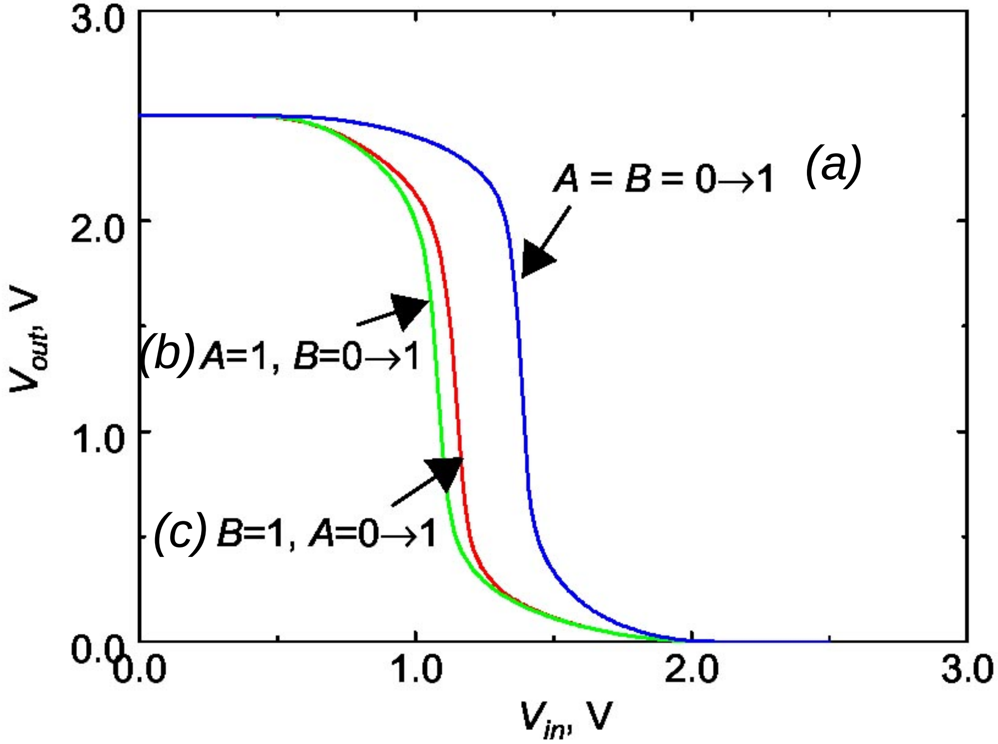
One input goes low

$$t_{PLH} = 0.69(R_p/2)C_L$$

Two inputs go low, parallel connection

Data-dependent VTC: 2nd order effects

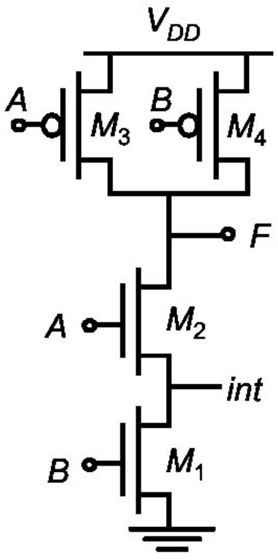


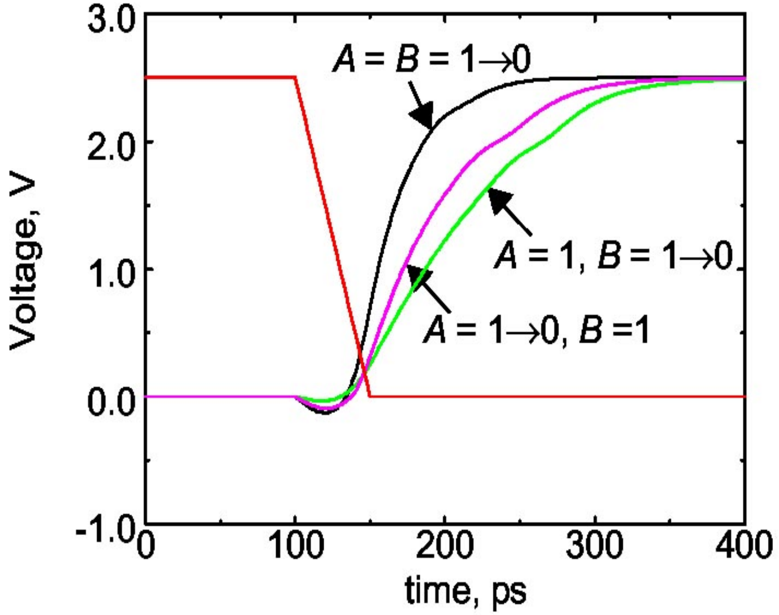


- **Charge at 'int'**
- **Body effect in M_2**
- **Short-circuit currents**

- **Don't need to be able to work with these effects**
- **But remember: there is more going on than shown by our simple, 1^{st} order model**

Data-dependent Timing (2)



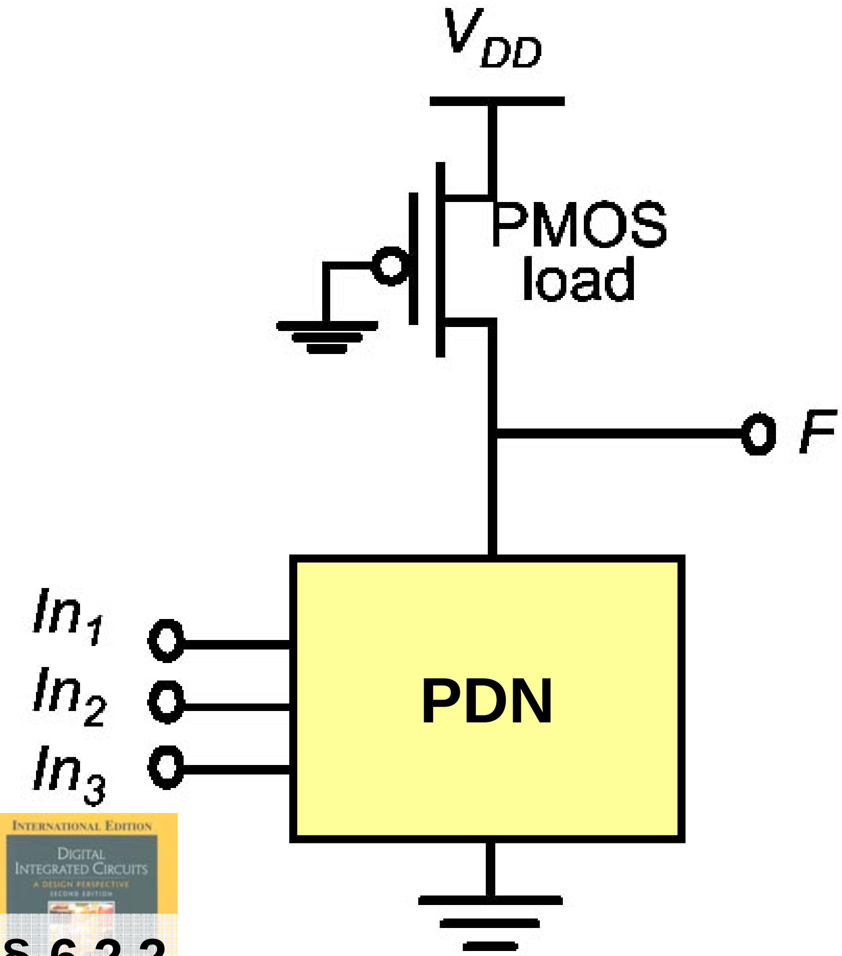


Input Data Pattern	Delay (pS)
A = *B*= 0→1	69
A = 1, *B*= 0→1	62
A= 0→1, *B* = 1	50
A=*B*=1→0	35
A=1, *B* = 1→0	76
A= 1→0, *B* = 1	57

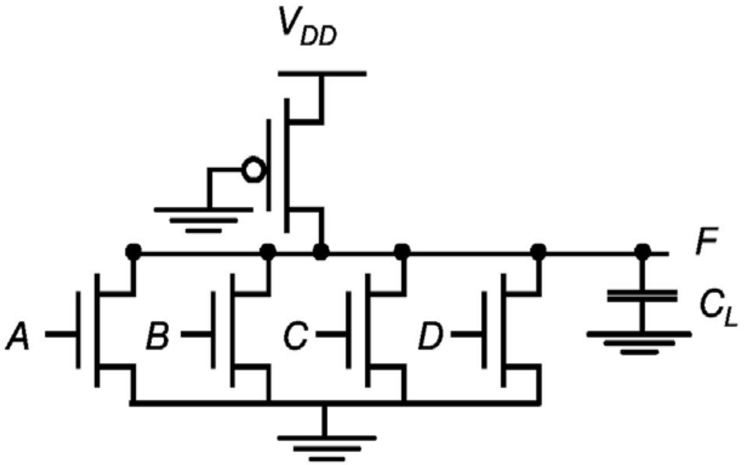
Ratioed logic

Pass transistor logic

Pseudo NMOS Ratioed Logic



- ☺ **Reduced area**
- ☺ **Reduced capacitances**
- ☹ **Increased V_{OL}**
- ☹ **Reduced noise margins**
- ☹ **Static dissipation**



§ 6.2.2

Ratioed Logic V_{OL} Computation

Exercise: verify these assumptions/steps

I_{Dn} (linear) = I_{Dp} (saturation)

$$k_n\left((V_{DD}-V_{Tn})V_{OL}-\cancel{\frac{V_{OL}^2}{2}}\right)=k_p\left((-V_{DD}-V_{Tp})V_{DSAT}-\cancel{\frac{V_{DSAT}^2}{2}}\right)$$

Ignore quadratic terms (they are relatively small)

$$k_n(\cancel{V_{DD}-V_{Tn}})V_{OL}\approx k_p(-\cancel{V_{DD}-V_{Tp}})V_{DSAT}$$

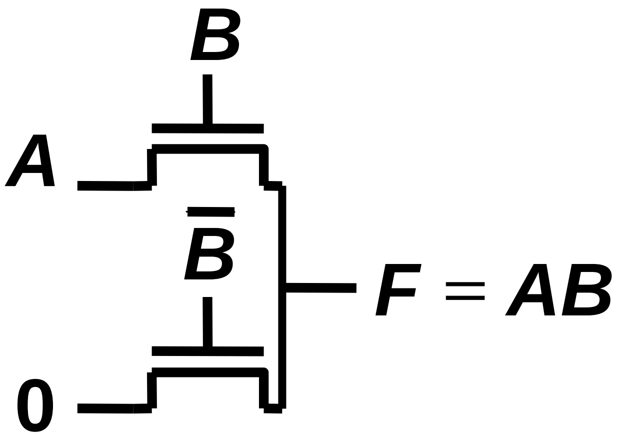
Ignore, because approximately equal

$$V_{OL}\approx\frac{k_p}{k_n}\left|V_{DSAT}\right|\approx\frac{\mu_p W_p}{\mu_n W_n}\left|V_{DSAT}\right|$$

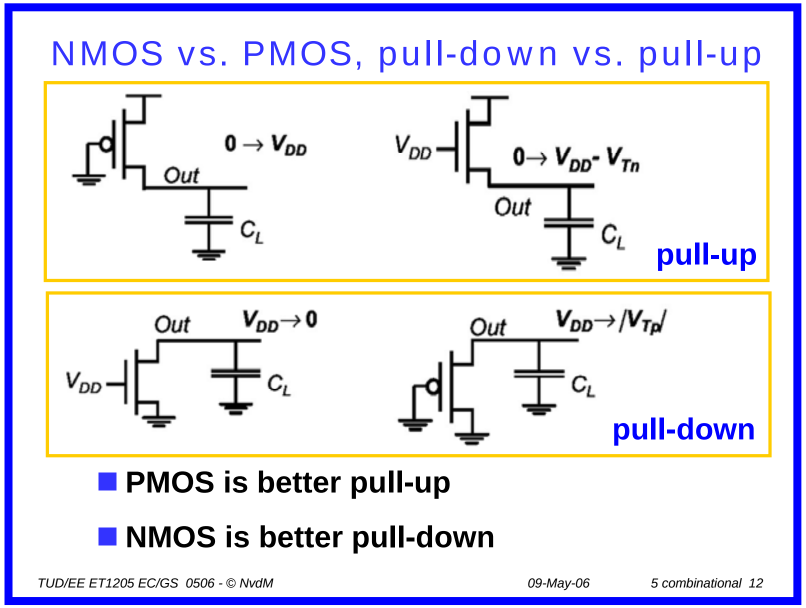
Pass-transistor and Pass-gate circuits

Pass Transistor Logic

- Save area, capacitances
- Need complementary inputs (extra inverters)



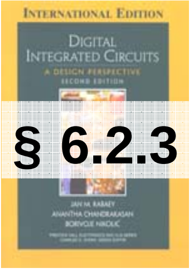
But remember:



NMOS vs. PMOS, pull-down vs. pull-up

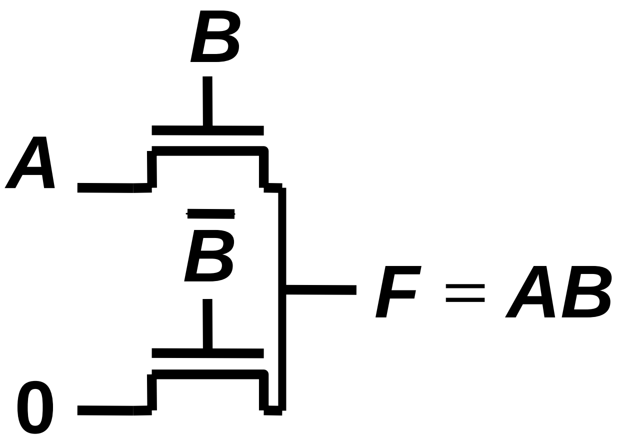
- PMOS is better pull-up
- NMOS is better pull-down

TUD/EE ET1205 EC/GS 0506 - © NvdM 09-May-06 5 combinational 12



§ 6.2.3

Pass Transistor Logic

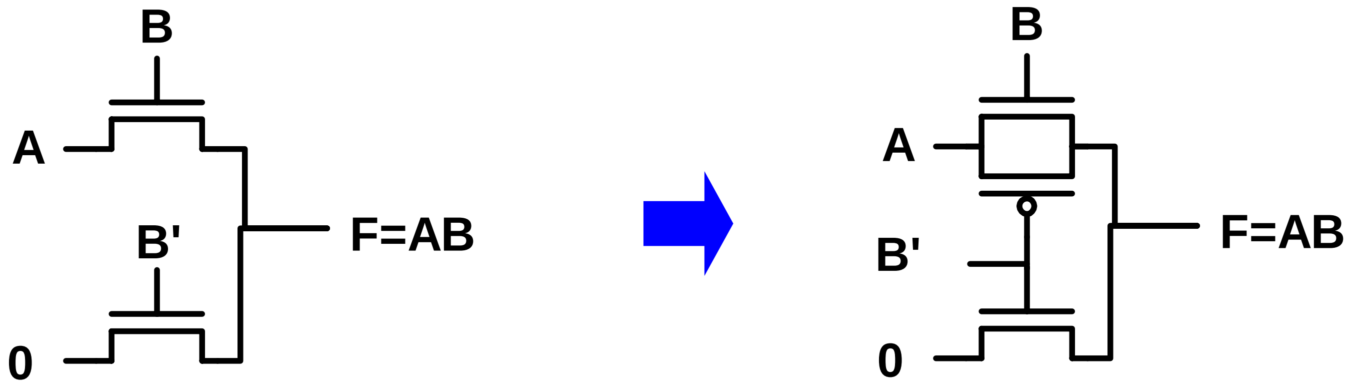


- Save **area, capacitances**
- Need complementary inputs (might mean extra inverters)
- Reduced V_{OH}, noise margins
- $V_{OH} = V_{DD} - \left(V_{Tno} + \gamma\left(\left(\sqrt{|2\phi_f| + V_{OH}}\right) - \sqrt{|2\phi_f|}\right)\right)$
- **Static dissipation** in subsequent static inverter/buffer
- Disadvantages (and advantages) may be reduced by **complementary pass gates** (NMOS + PMOS parallel)

Exercise: Why is there static dissipation in next conventional gate?

Pass Gates

- Remedy: use an N-MOS and a P-MOS in parallel

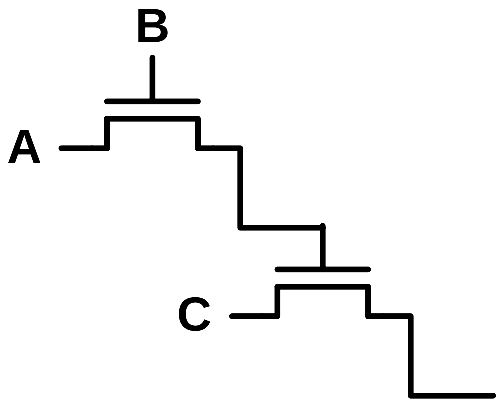


- Pass gates eliminate some of the disadvantages of simple pass-transistors
- But also some of the advantages
- Design remains a trade-off!

Pass-gate a.k.a. Transmission-gate

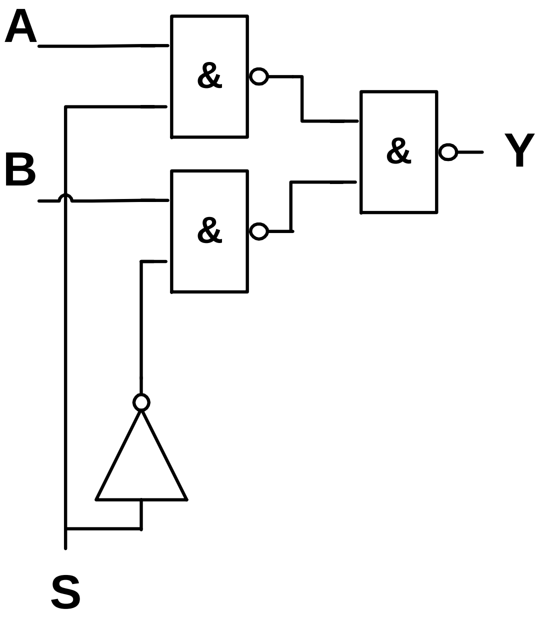
Exercise

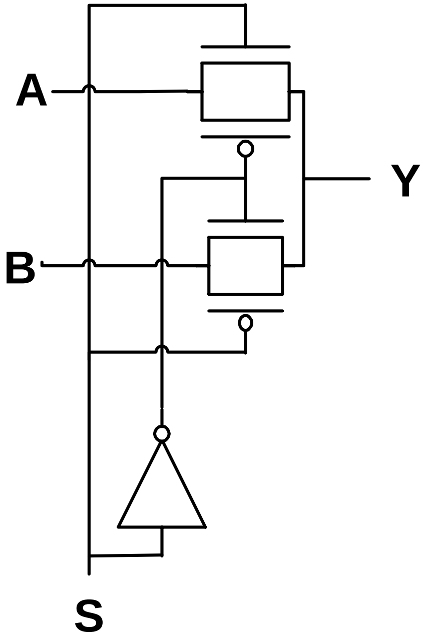
- Discuss what happens when you connect the output of a single pass-transistor (not a pass-gate) to the input of another pass-transistor stage (i.e. the gate of another pass-transistor). Why should you never use such a circuit?



Pass Transistor Logic

- Most typical use: for multiplexing, or path selecting
- Assume in circuit below it is required to either connect A or B to Y, under control by S
- Y = AS + BS' (S' is easier notation for S-bar = S-inverse = $\overline{S}$)
- Y = ((AS)' (BS)')' allows realization with 3 NAND-2 and 1 INV: 14 transistors
- Pass gate needs only 6 (or 8) transistors (see also Katz, section 4.2)





Summary

- **Conventional Static CMOS basic principles**
- **Complementary static CMOS**
  - **Complex Logic Gates**
  - **VTC, Delay and Sizing**
- **Ratioed logic**
- **Pass transistor logic**