COMBINATIONAL LOGIC

Exam

- April 3rd, 9am – 12am
- Written exam, open book, also printed slides with your notes on it.

Material	Topic	Remarks
Ch 1	Intro	Completely, pay attention to §1.3
Ch 2	Manufacturing	Only §2.1 - §2.3
Ch 3	Devices	Completely
Ch 4	Interconnect	Except §4.4.5, §4.5
Ch 5	Inverter	Completely
Ch 6	Combinational	Only §6.1, §6.2
Ch 7	Sequential	Only §7.1, §7.2

- **Final Lecture: March 25, 10:45-12:30**
- **Room D *(probably)***
- **§7.1 – 7.2**

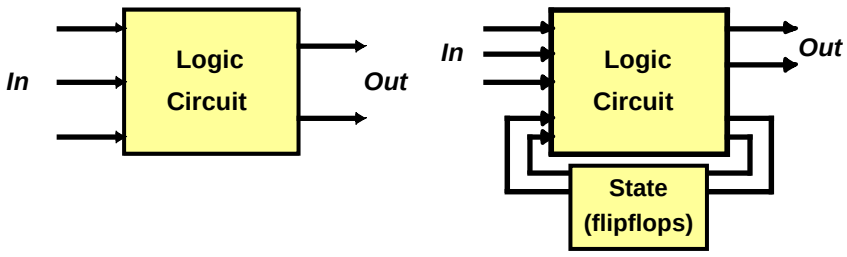
March 2008			
Thu 20	Tue 25		
15:45-17:30	8:45-10:30	10:45-12:30	13:45-15:30
OK		OK	OK
OK	OK	OK	OK
	OK	OK	OK
		OK	
	OK	OK	OK
		OK	OK
	OK	OK	
	OK	OK	OK
	OK	OK	OK
	OK	OK	OK
		OK	
		OK	
	OK	OK	OK
OK		OK	
	OK	OK	OK
		OK	OK
		OK	
	OK	OK	OK
☐	☐	☐	☐
15:45-17:30	8:45-10:30	10:45-12:30	13:45-15:30

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

§ 6.2

Output = *f* (In)

Output = *f* (In, History)

Reminder

DeMorgan Transformations

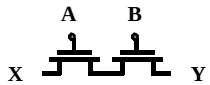
$$\overline{A+B} = \overline{A}.\overline{B}$$

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

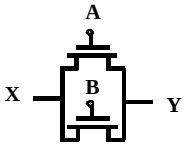
NMOS Transistors in Series/Parallel Connection

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



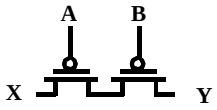
Y = X if A or B

§ 6.2.1

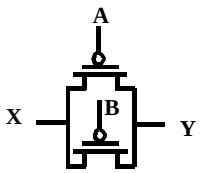
PMOS Transistors in Series/Parallel Connection

PMOS switch closes when switch control input is low

Y = X if …

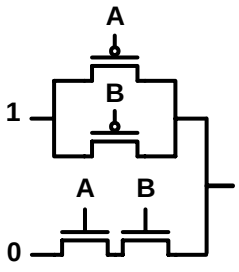


Y = X if $\bar{A}$ and $\bar{B}$



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

2-Input Nand



$Y = 1 \text{ if } \bar{A} \text{ OR } \bar{B}$

$Y = 1 \text{ if } \overline{A \text{ AND } B}$ } DeMorgan

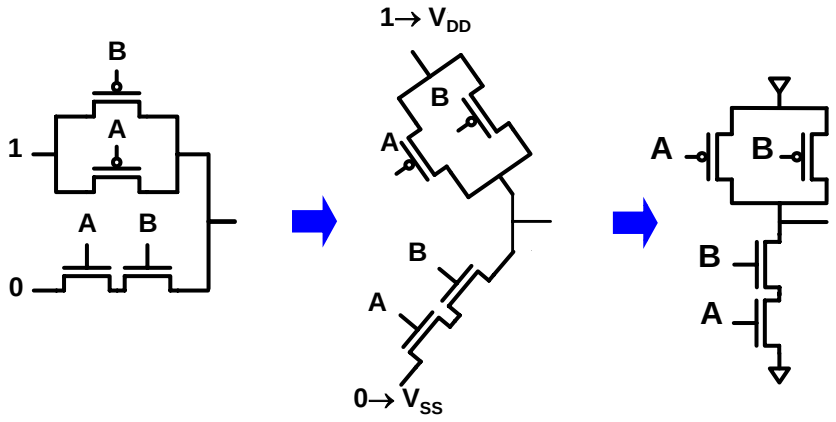
$Y = \overline{A \text{ AND } B}$

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

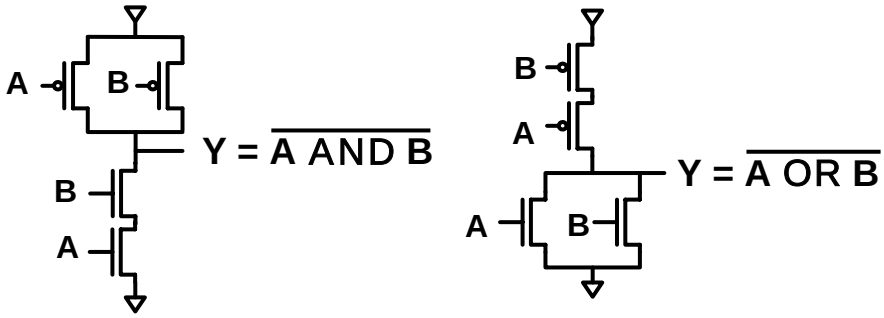
B	A	Y
0	0	1
0	1	1
1	0	1
1	1	0

2-Input Nand

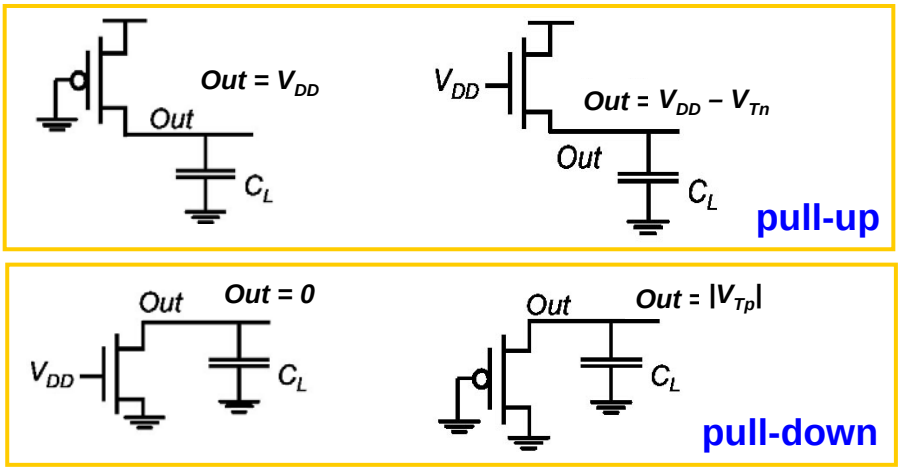
$Y = \overline{A \text{ AND } B}$



2-input Nand/Nor

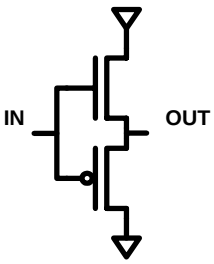


NMOS vs. PMOS, pull-down vs. pull-up



- 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

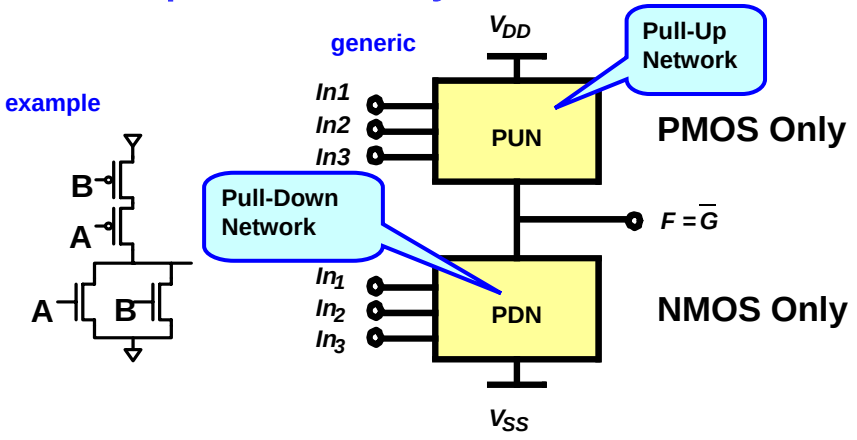
Inverting behavior

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

Complementary static CMOS

- **Complex Logic Gates**
- **VTC, Delay and Sizing**

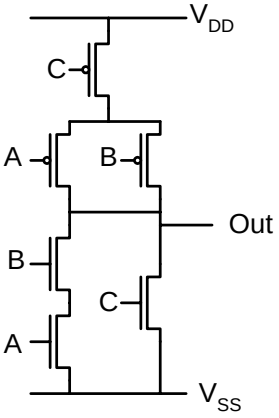
Complementary Static CMOS



- 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	PDN Off, PUN On
0	0	1	?	1	1	
0	1	0	?	1	1	
0	1	1	0	?	0	PUN Off, PDN On
1	0	0	0	?	0	
1	0	1	0	?	0	
1	1	0	0	?	0	
1	1	1	0	?	0	

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

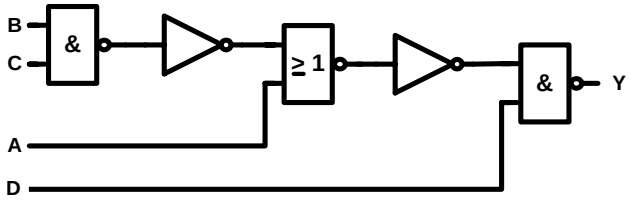
Implementation of Combinational Logic

- 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

Implementation of Combinational Logic

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

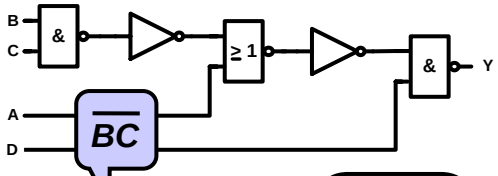


transistors ☐ logic depth ☐

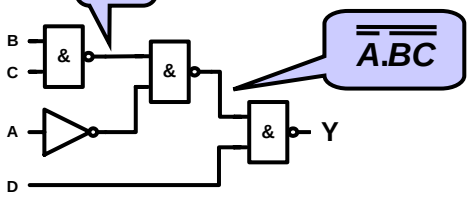
- Q: Can this be improved?

Improved Gate Level Implementation

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



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



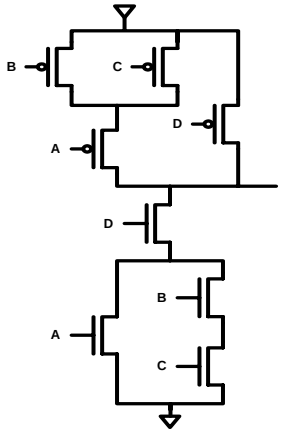
transistors ☐

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

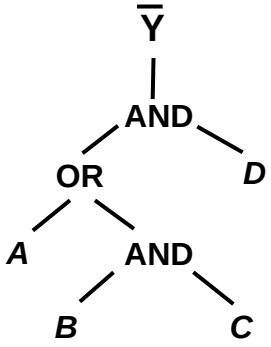


$Y = \overline{(A + BC)D}$

transistors: 8
Logic depth: 1

How to Synthesize Complex Gates

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

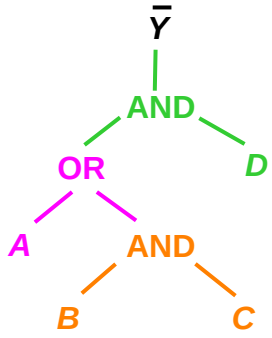


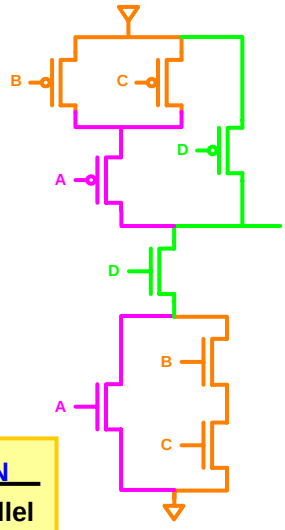
- Using tree representation of Boolean function
- *Operator* with branches for *operands*
- As a series-parallel network

	PDN	PUN
AND	Series	Parallel
OR	Parallel	Series

Complex Gate Synthesis Example

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



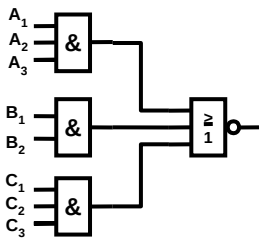


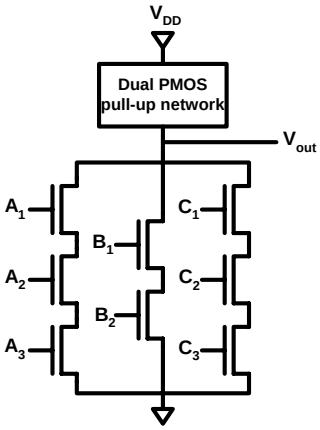
	PDN	PUN
AND	Series	Parallel
OR	Parallel	Series

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

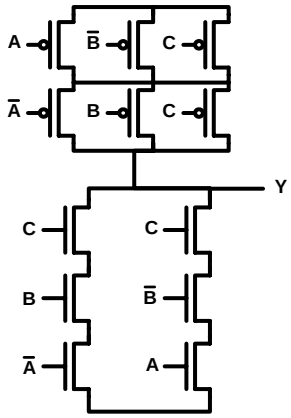




And-Or-Invert Example

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

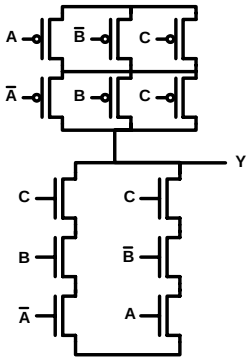
A	B	C	Y	
0	0	0	1	
0	0	1	1	
0	1	0	1	
0	1	1	0	→ $\bar{A}BC$
1	0	0	1	
1	0	1	0	→ $A\bar{B}C$
1	1	0	1	
1	1	1	1	

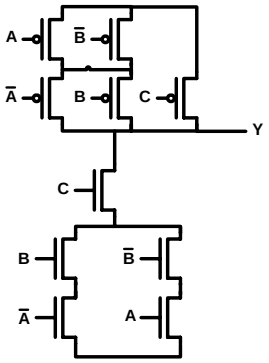


$\bar{Y} = \bar{A}BC + A\bar{B}C$

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

And-Or-Invert Improvement





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

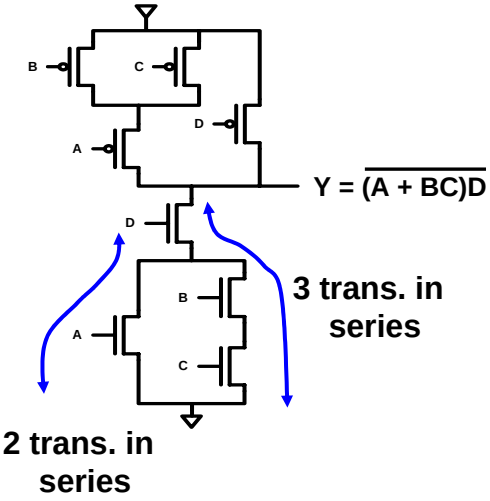
12 transistors

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

10 transistors

2-level logic minimization: boolean algebra technique

CMOS Complex Gate Sizing

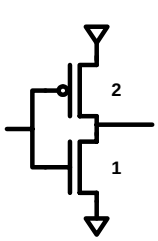


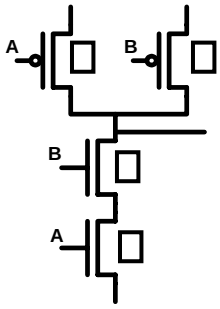
- Function of gate independent of transistor sizes: ratio-less
- But current-drive capability (timing) depends on transistor sizes
- Worst-case current-drive depends on number of transistors in series

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

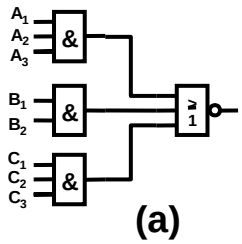




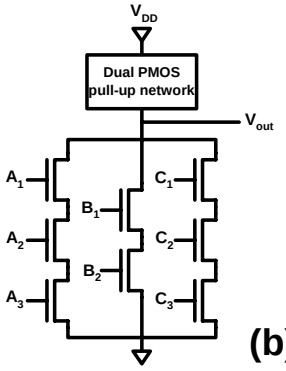
- W/L ratios
- what are the W/L of 2-input NAND for same drive strength?

  0-th order calculation

Exercise



(a)

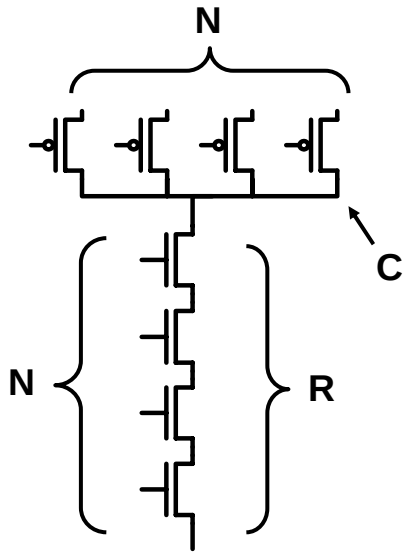


(b)

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

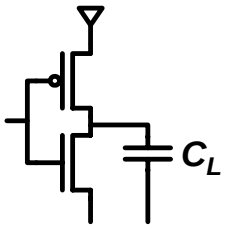
R **linear** in N

Delay ∝ RC **quadratic** in N

Empirical

Delay = $a_1FI + a_2FI^2 + a_3FO$

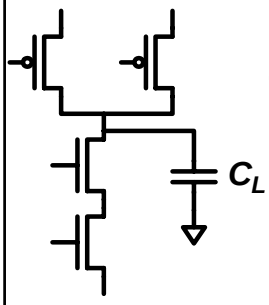
Data-Dependent Timing



$t_{PHL} = 0.69R_NC_L$

$t_{PLH} = 0.69R_PC_L$

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

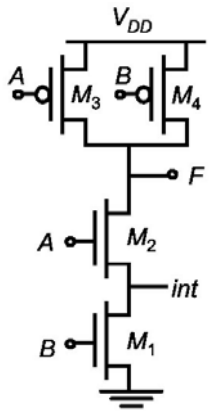


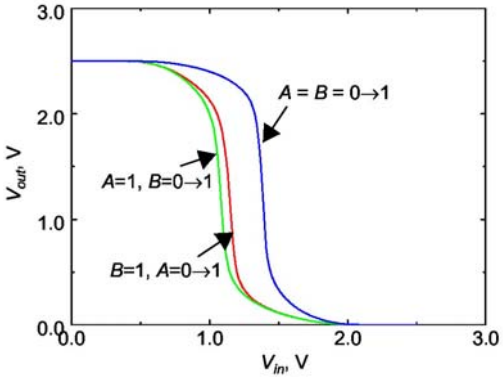
$t_{PHL} = 0.69(R_N \times 2)C_L$ **Series** connection

$t_{PLH} = 0.69R_pC_L$ **One input** goes low

$t_{PLH} = 0.69(R_p/2)C_L$ **Two inputs** go low, **parallel** connection

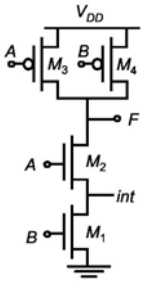
Data-dependent VTC: 2nd order effects

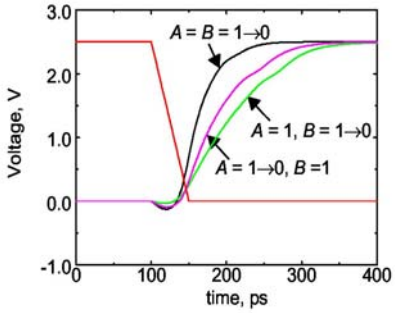




- **Charge at 'int'**
- **Body effect in M_2**
- **Short-circuit currents**

Data-dependent Timing





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

A=1, B=↓: need to charge *int*
A=↓, B=1: *int* does not need to be charged
A=↓, B=↓: twice the pull-up strength

Logical Effort Methodology

Inverter delay: $t_p = t_{p0}\left(1+\frac{f}{\gamma}\right)$

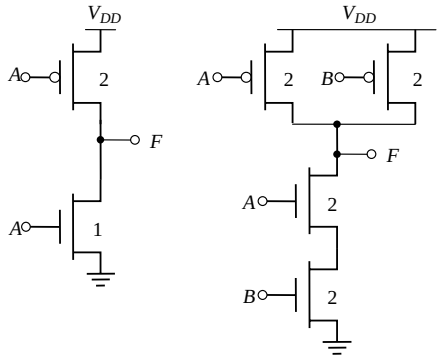
Gate delay: $t_p = t_{p0}\left(p+\frac{gf}{\gamma}\right) = t_{p0}\left(p+\frac{h}{\gamma}\right)$

Logical Effort Methodology Definitions:

- ***p*** ratio of **intrinsic delay** compared to inverter
- ***f*** **electrical effort** – ratio between C_{load} and C_{in}
- ***g*** **logical effort** – ratio of drive strength for same size
- ***h*** **gate effort** – $h = f \cdot g$

[Logical Effort – Designing Fast CMOS Circuits, Sutherland, Sproul, Harris]

Intrinsic Delay



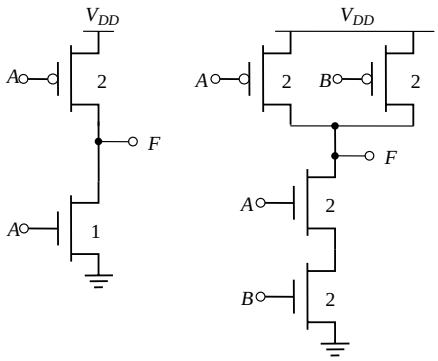
Inverter p=1

2-input NAND p=2

- ***p*** ratio of intrinsic delay compared to inverter
- ***p*** is ratio of output capacitances if gate is sized for identical drive strength
- p_{nand} is (2+2+2)/(2+1) = 2

$t_p = t_{p0}\left(p+\frac{gf}{\gamma}\right)$

Logical Effort



Inverter

g=1

2-input NAND

p=2

***g* logical effort – ratio of drive strength for same size**

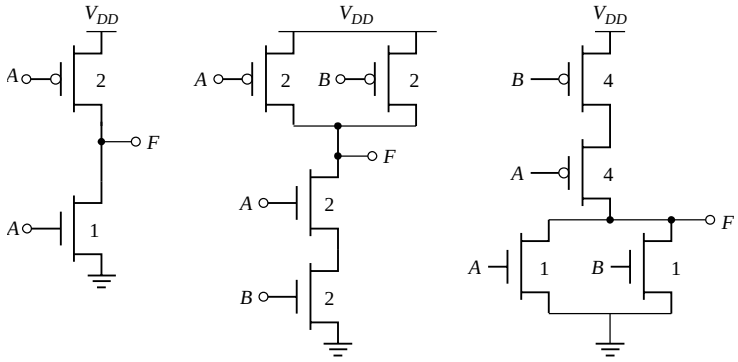
***g* same as ratio of input cap for same strength**

***g*$_{nand}$ is (2+2)/(2+1) = 4/3**

$$t_p = t_{p0}\left(p + \frac{gf}{\gamma}\right)$$

- **Logical effort: a multi-input gate has more input cap compared to inverter for same drive strength**
- **Needs to be corrected for in delay calculations**
- **Can be done by inclusion in electrical effort: replace *f* by *g*·*f***

Logical Effort



Inverter

p=1, g=1

2-input NAND

p=2, g=4/3

2-input NOR

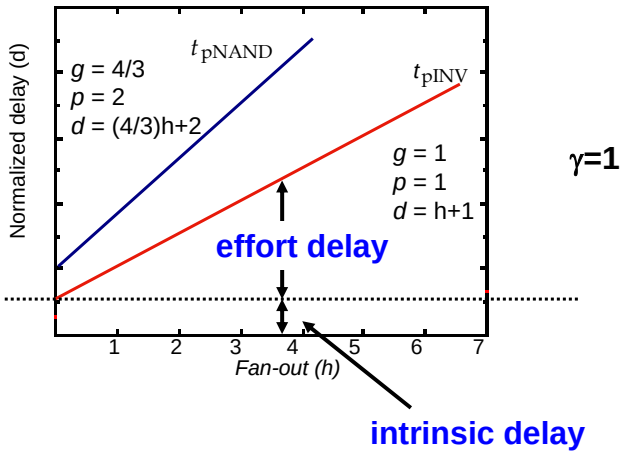
p=2, g=5/3

***p* ratio of intrinsic delay compared to inverter**

***g* logical effort – ratio of drive strength for same size**

p, g independent of sizing, only topology of circuit

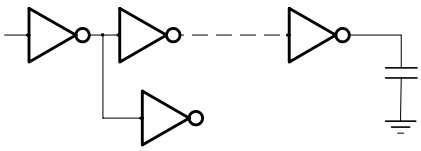
Delay vs Fan-Out



Add Branching Effort

Branching effort:

$$b = \frac{C_{on-path} + C_{off-path}}{C_{on-path}}$$



Multistage Networks

$$Delay = \sum_{i=1}^{N} (p_i + g_i \cdot f_i)$$

Normalized w.r.t. unit delay

Stage effort:	$h_i = g_i f_i$
Path electrical effort:	$F = C_{out}/C_{in}$
Path logical effort:	$G = g_1 g_2 \ldots g_N$
Branching effort:	$B = b_1 b_2 \ldots b_N$
Path effort:	$H = GFB$
Path delay	$D = \Sigma d_i = \Sigma p_i + \Sigma h_i$

Optimum Effort per Stage

When each stage bears the same effort:

$$h^N = H$$

$$h = \sqrt[N]{H}$$

Stage efforts: $g_1 f_1 = g_2 f_2 = \ldots = g_N f_N = h$

Effective fanout of each stage: $f_i = h/g_i$

larger fanout for simpler stages

Minimum path delay

$$\hat{D} = \sum (g_i f_i + p_i) = NH^{1/N} + P$$

Optimal Number of Stages

For a given load,
and given input capacitance of the first gate
Find optimal number of stages and optimal sizing

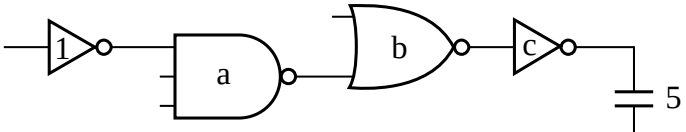
$$D = NH^{1/N} + Np_{inv}$$

$$\frac{\partial D}{\partial N} = -H^{1/N}\ln\left(H^{1/N}\right) + H^{1/N} + p_{inv} = 0$$

Substitute 'best stage effort' $h = H^{1/\hat{N}}$

Example: Optimize Path

a, b, c: gate sizes t.b.d.



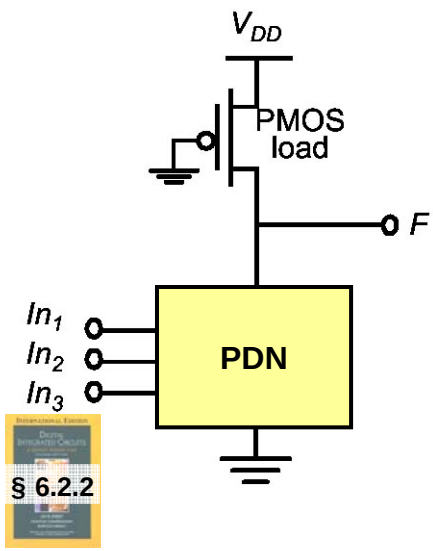
$g = 1$	$g = 5/3$	$g = 5/3$	$g = 1$
$f = a$	$f = b/a$	$f = c/b$	$f = 5/c$

Effective fanout, F = 5
$G = \Pi g_i = 25/9$
$H = GFB = 125/9 = 13.9$
$h = H^{1/4} = 1.93$
$a = 1.93$
$b = ha/g_2 = 2.23$
$c = hb/g_3 = 5g_4/f = 2.59$

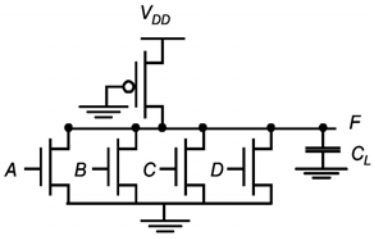
Ratioed logic

Pass transistor logic

Pseudo NMOS Ratioed Logic



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



Ratioed Logic V_{OL} Computation

Exercise: verify these assumptions/steps

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

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

Ignore quadratic terms (they are relatively small)

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

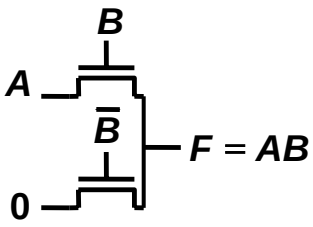
Ignore, because approximately equal

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

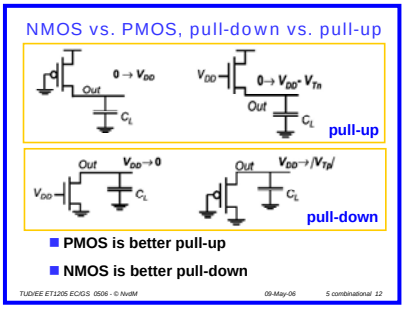
Pass-transistor and Pass-gate circuits

Pass Transistor Logic

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

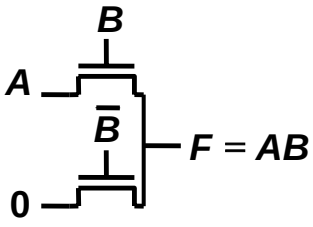


But remember:



Pass Transistor Logic

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

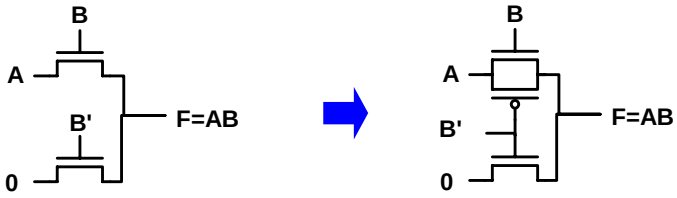


- Reduced V_{OH}, noise margins
- $V_{OH} = V_{DD} - \left(V_{Tno} + \gamma\left(\left(\sqrt{|2\phi_f| + V_{OH}}\right) - \sqrt{|2\phi_f|}\right)\right)$
- Static dissipation in subsequent static inverter/buffer
- Disadvantages (and advantages) may be reduced by complementary pass gates (NMOS + PMOS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

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

Pass Gates

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

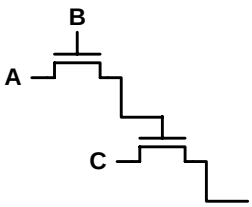


- Pass gates eliminate some of the disadvantages of simple pass-transistors
- But also some of the advantages
- Design remains a trade-off!

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

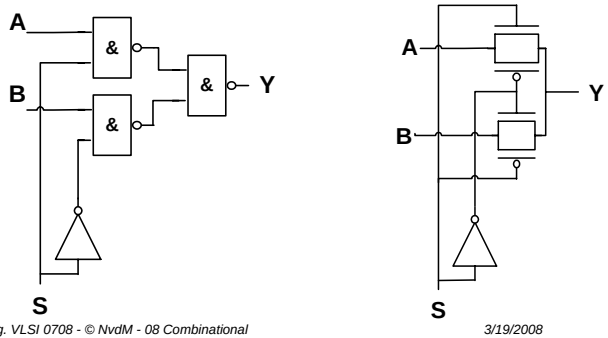
Exercise

- 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