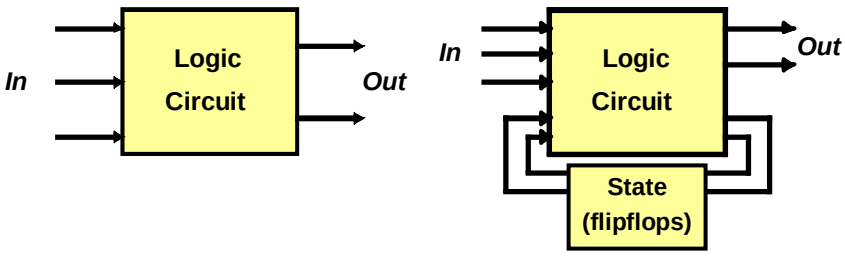
COMBINATIONAL LOGIC

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

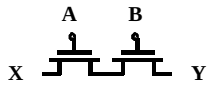
$$\overline{A+B} = \overline{A}.\overline{B}$$

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

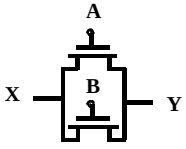
NMOS Transistors in Series/Parallel Connection

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



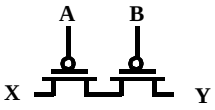
Y = X if A or B

§ 6.2.1

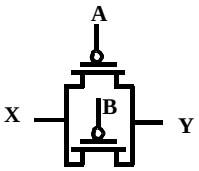
PMOS Transistors in Series/Parallel Connection

PMOS switch closes when switch control input is low

Y = X if …

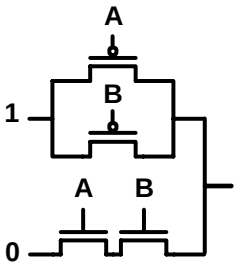


Y = X if $\overline{A}$ and $\overline{B}$



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

2-Input Nand



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

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

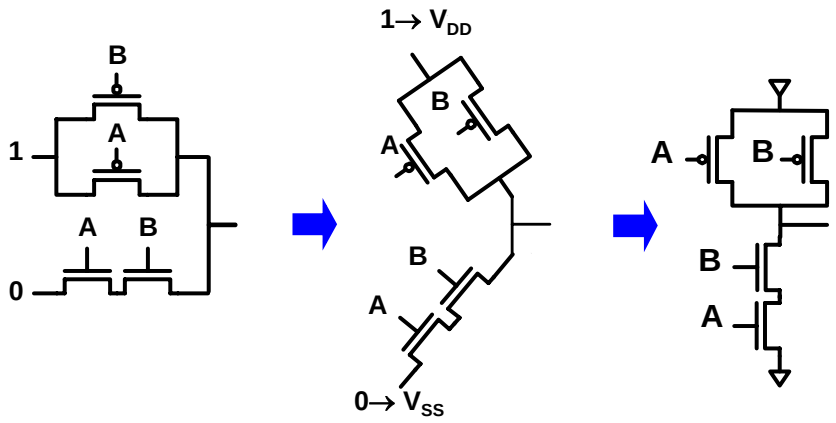
$Y = \overline{A \text{ AND } B}$

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

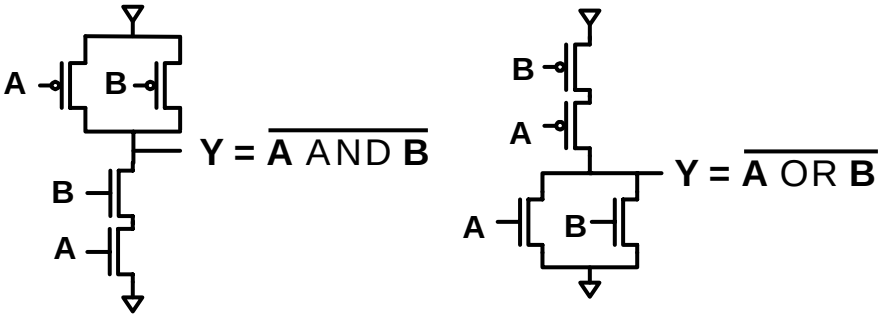
B	A	Y
0	0	1
0	1	1
1	0	1
1	1	0

2-Input Nand

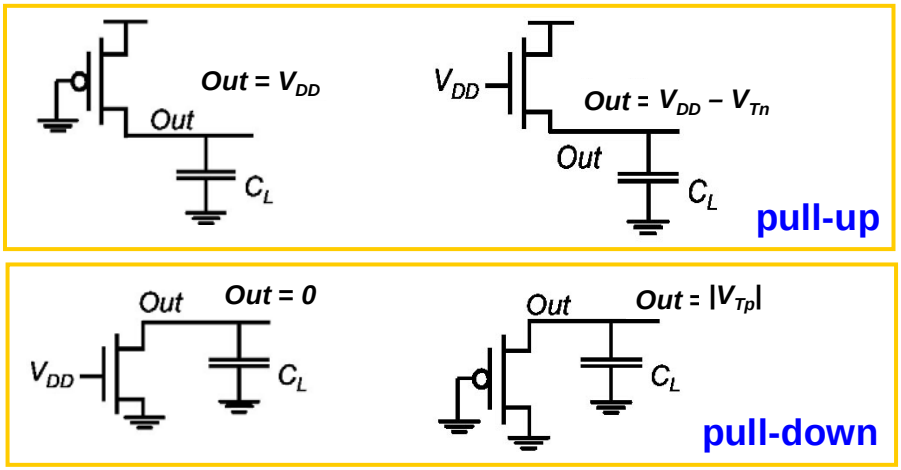
$Y = \overline{A \text{ AND } B}$



2-input Nand/Nor

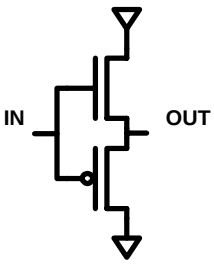


NMOS vs. PMOS, pull-down vs. pull-up



- 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

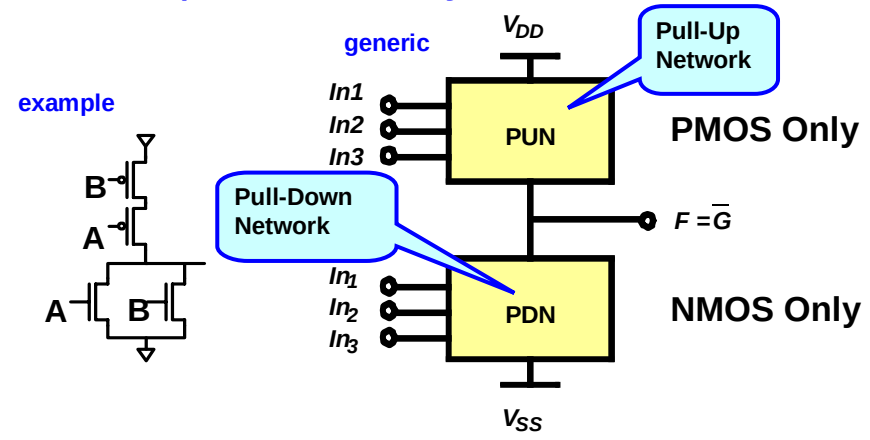
Inverting behavior

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

Complementary static CMOS

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

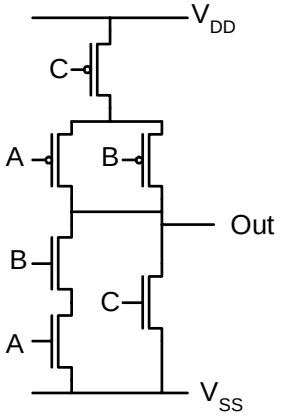
Complementary Static CMOS



- 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

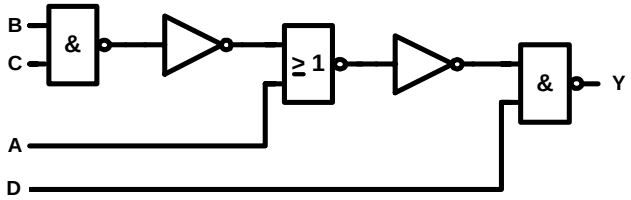
Implementation of Combinational Logic

- 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

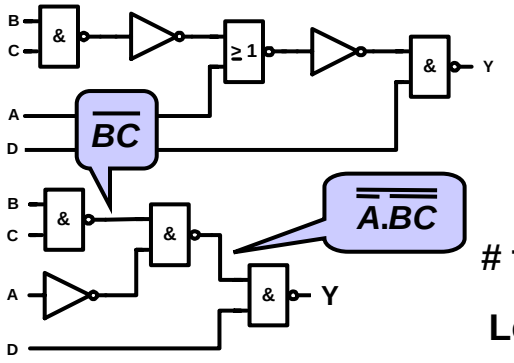


transistors ☐ logic depth ☐

- Q: Can this be improved?

Improved Gate Level Implementation

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



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

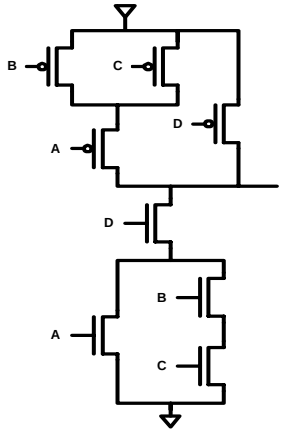
transistors ☐

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



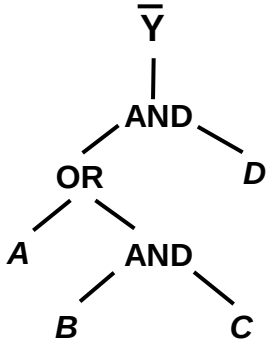
$Y = \overline{(A + BC)D}$

transistors: 8

Logic depth: 1

How to Synthesize Complex Gates

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

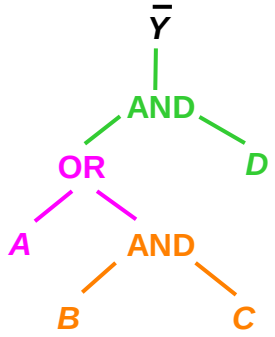


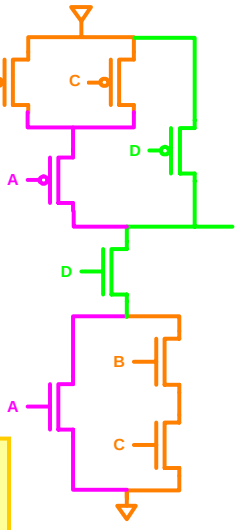
- Using tree representation of Boolean function
- *Operator* with branches for *operands*
- As a series-parallel network

	PDN	PUN
AND	Series	Parallel
OR	Parallel	Series

Complex Gate Synthesis Example

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



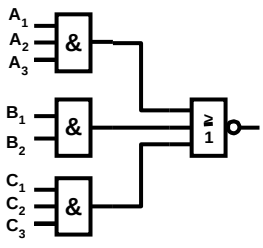


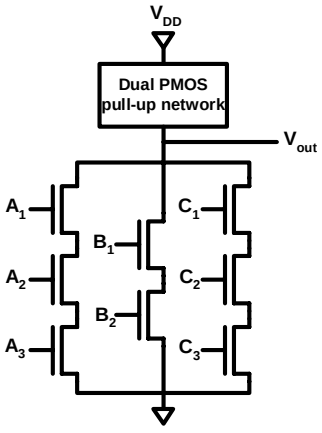
	PDN	PUN
AND	Series	Parallel
OR	Parallel	Series

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

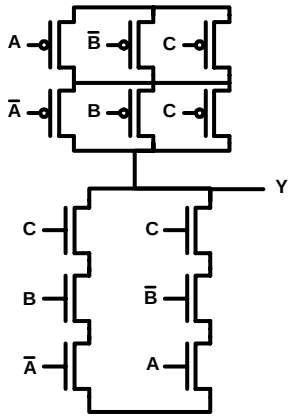




And-Or-Invert Example

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

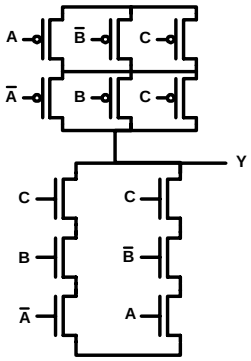
A	B	C	Y	
0	0	0	1	
0	0	1	1	
0	1	0	1	
0	1	1	0	→ $\bar{A}BC$
1	0	0	1	
1	0	1	0	→ $A\bar{B}C$
1	1	0	1	
1	1	1	1	

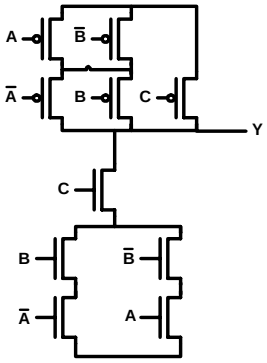


$\bar{Y} = \bar{A}BC + A\bar{B}C$

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

And-Or-Invert Improvement





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

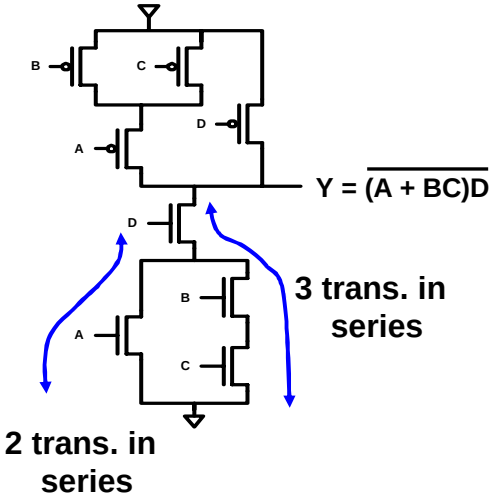
12 transistors

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

10 transistors

2-level logic minimization: boolean algebra technique

CMOS Complex Gate Sizing

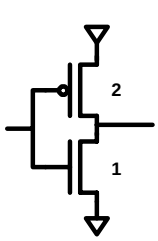


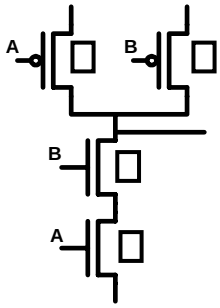
- Function of gate independent of transistor sizes: ratio-less
- But current-drive capability (timing) depends on transistor sizes
- Worst-case current-drive depends on number of transistors in series

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

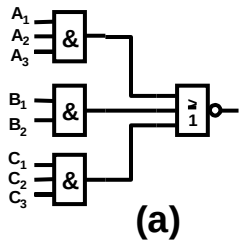




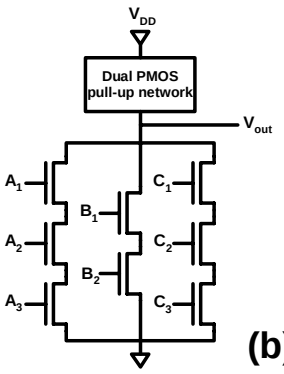
- W/L ratios
- what are the W/L of 2-input NAND for same drive strength?

  0-th order calculation

Exercise



(a)

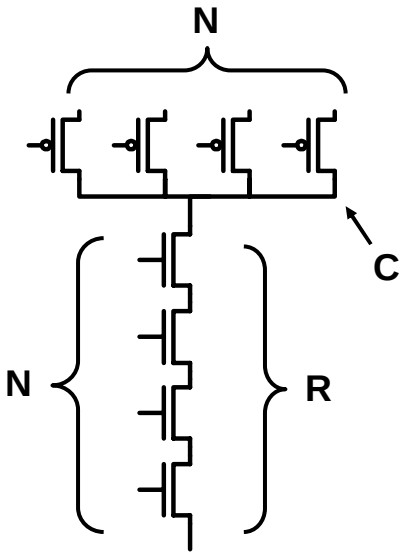


(b)

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

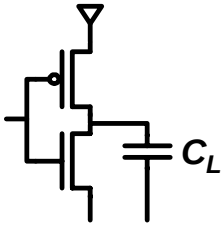
R **linear** in N

Delay $\propto$ RC **quadratic** in N

Empirical

Delay = $a_1FI + a_2FI^2 + a_3FO$

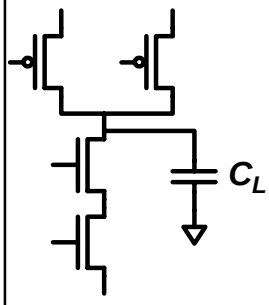
Data-Dependent Timing



$t_{PHL} = 0.69R_NC_L$

$t_{PLH} = 0.69R_PC_L$

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

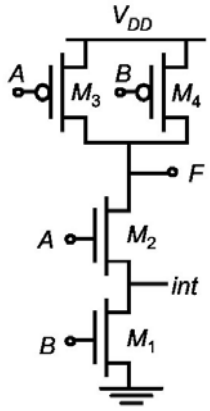


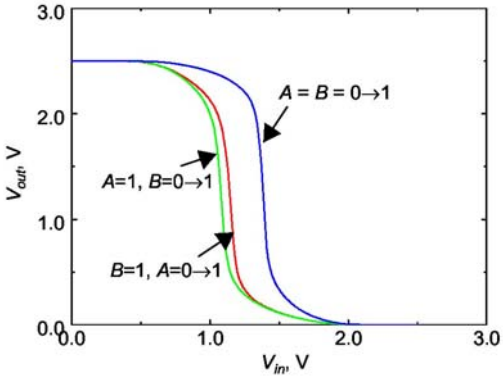
$t_{PHL} = 0.69(R_N \times 2)C_L$ **Series** connection

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

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

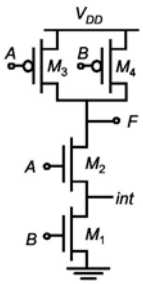
Data-dependent VTC: 2nd order effects

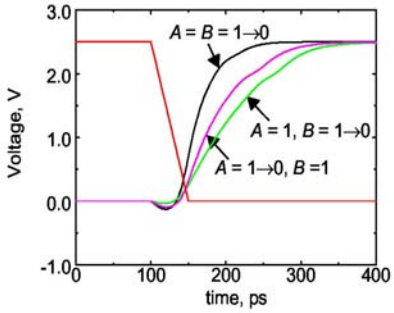




- **Charge at 'int'**
- **Body effect in M_2**
- **Short-circuit currents**

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

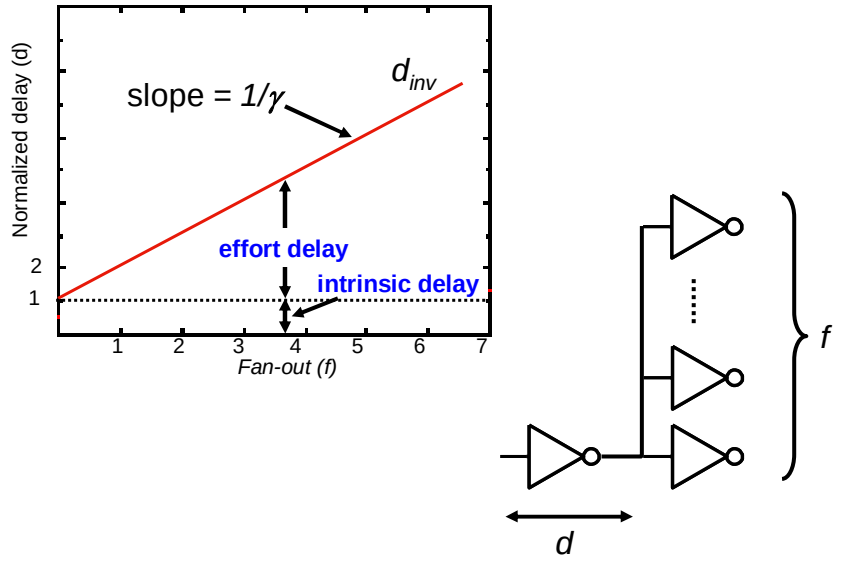
Remember: Inverter Delay

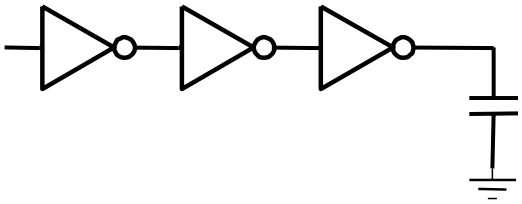
$$t_p = \frac{R_{ref}}{S}\left(C_{ref}S + f\frac{C_{ref}S}{\gamma}\right) = t_{po}\left(1+\frac{f}{\gamma}\right) \qquad \text{with } t_{po} = R_{ref}C_{ref}$$

$$d = 1 + f/\gamma$$

R_{ref}	Equivalent output resistance of min size inverter
C_{ref}	Drain (and Miller) cap of min size inverter
S	Size of inverter (relative to min inverter)
f	*electrical effort* – ratio between C_{load} and C_{in} (number of identical copies as load)
γ	ratio of drain cap to gate cap
t_{p0}	*intrinsic delay* - delay of unloaded inverter
	$t_{p0} \approx$ 20 ps for a 250 nm process, $t_{p0} \approx$ 5 ps for a 45 nm process
d	*normalized delay* = t_p/t_{po}

$$d_{inv} = 1 + f/\gamma$$





Logical Effort Methodology

Inverter delay: $d_{inv} = 1 + f/\gamma$

Gate delay: $d_{gate} = p + gf/\gamma$

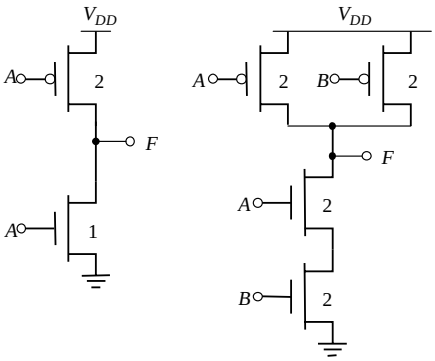
Logical Effort Methodology Definitions:

- ***p*** **parasitic delay**
  - **ratio of intrinsic delay compared to inverter**
  - **ratio of output cap for same drive strength**
- ***g*** **logical effort**
  - **how much more load the gate creates**
  - **ratio of input cap for same drive strength**
- ***h*** **gate effort, *h* = *gf***

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

Beware: most texts, incl. Sutherland, interchange *f* and *h*

Intrinsic, Parasitic Delay



Inverter — **p=1**

2-input NAND — **p=2**

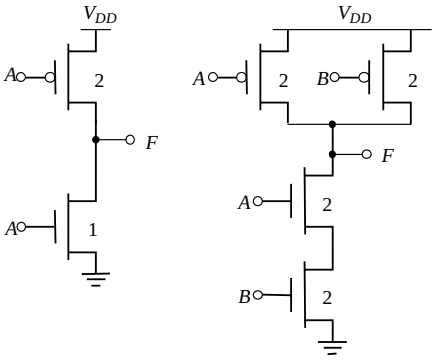
p **parasitic delay - ratio of intrinsic delay compared to inverter**

p **is ratio of output capacitances if gate is sized for identical drive strength**

p_{nand} **is (2+2+2)/(2+1) = 2**

$$d_{gate} = p + gf/\gamma$$

Logical Effort



Inverter — **g=1**

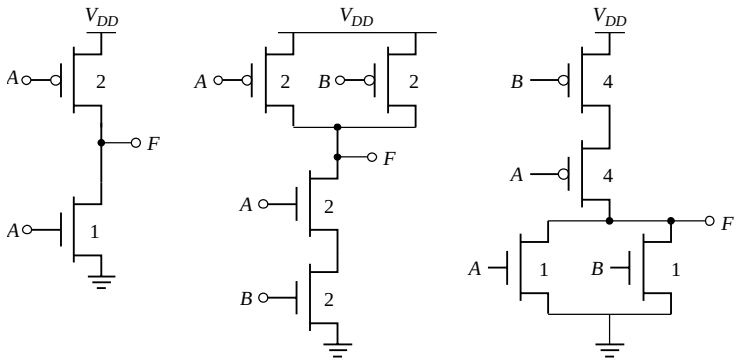
2-input NAND — **p=2**

g **logical effort – ratio of input cap (per pin) if gate is sized for identical drive strength**

g_{nand} **is (2+2)/(2+1) = 4/3**

$$d_{gate} = p + gf/\gamma$$

Logical Effort



Inverter — **p=1, g=1**

2-input NAND — **p=2, g=4/3**

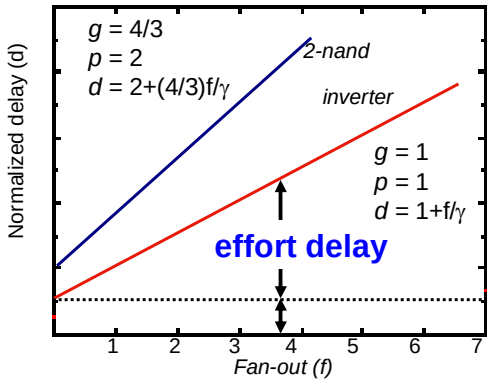
2-input NOR — **p=2, g=5/3**

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

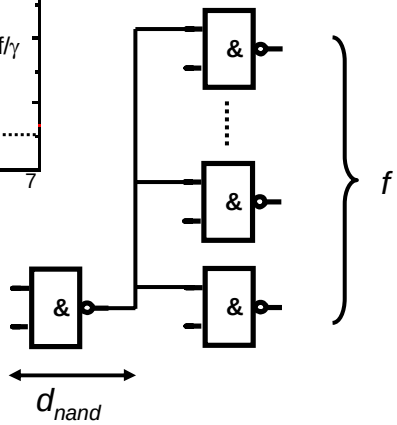
g **logical effort – ratio of inp. cap for same strength**

p, g independent of sizing, only topology of gate

Delay vs Fan-Out



$$d_{gate} = p + gf/\gamma$$



Multistage Networks

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

Normalized w.r.t. unit delay, assume $\gamma = 1$

Stage effort:	$h_i = g_i f_i$
Path electrical effort:	$F = f_1 f_2 \ldots f_N = C_{out}/C_{in}$
Path logical effort:	$G = g_1 g_2 \ldots g_N$
Path effort:	$H = GF$
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: $f_i = h/g_i$

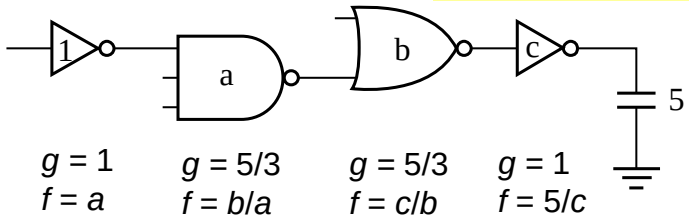
larger fanout for simpler stages

Minimum path delay

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

Example: Optimize Path

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



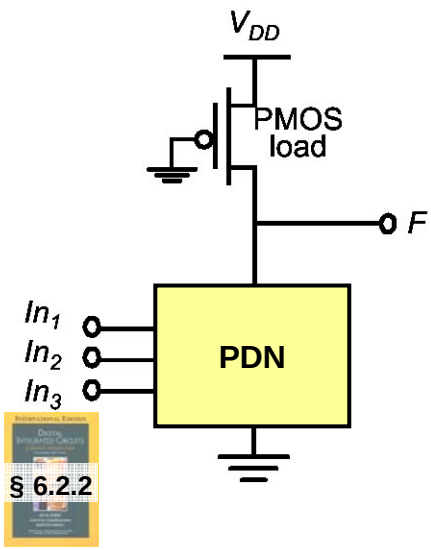
$g = 1$ $f = a$ | $g = 5/3$ $f = b/a$ | $g = 5/3$ $f = c/b$ | $g = 1$ $f = 5/c$

Effective fanout, F = 5
$G = \Pi g_i = 25/9$
$H = GF = 125/9 = 13.9$
$h = H^{1/4} = 1.93$
$g_1 f_1 = h \Rightarrow 1 \cdot a = h \Rightarrow a = 1.93$
$g_2 f_2 = h \Rightarrow (5/3\ b)/1.93 = h \Rightarrow b = 2.23$
$g_3 f_3 = h \Rightarrow (5/3\ c)/2.23 = h \Rightarrow c = 2.59$
$(g_3 f_3 = h \Rightarrow 5/c = h \Rightarrow c = 2.59)$

Ratioed logic

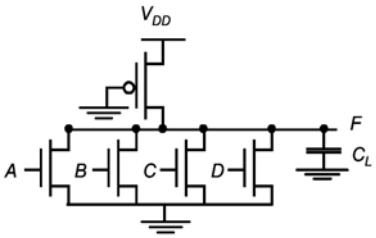
Pass transistor logic

Pseudo NMOS Ratioed Logic



§ 6.2.2

- ☺ 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)

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

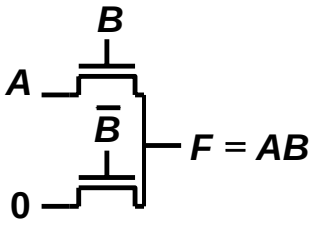
Ignore, because approximately equal

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

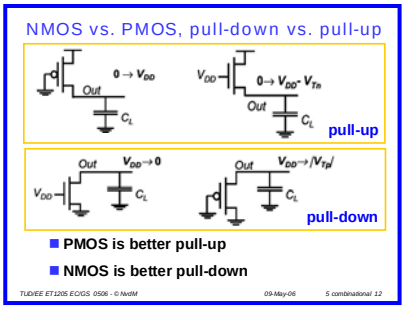
Pass-transistor and Pass-gate circuits

Pass Transistor Logic

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

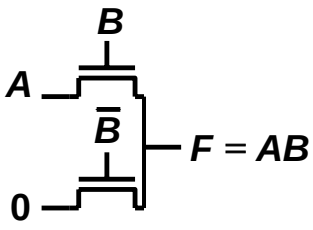


But remember:



Pass Transistor Logic

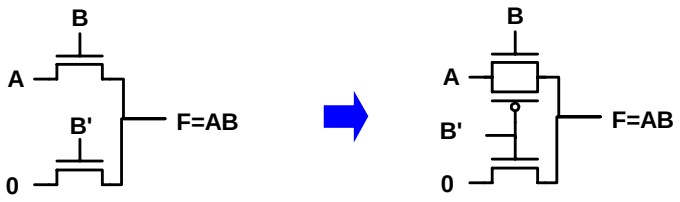
- Save area, capacitances
- Need complementary inputs (might mean extra inverters)
- Reduced V_{OH}, noise margins
- $V_{OH} = V_{DD} - \left(V_{Tno} + \gamma\left(\left(\sqrt{|2\phi_f| + V_{OH}}\right) - \sqrt{|2\phi_f|}\right)\right)$
- Static dissipation in subsequent static inverter/buffer
- Disadvantages (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

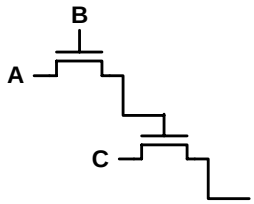


- Pass gates eliminate some of the disadvantages of simple pass-transistors
- But also some of the advantages
- Design remains a trade-off!

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

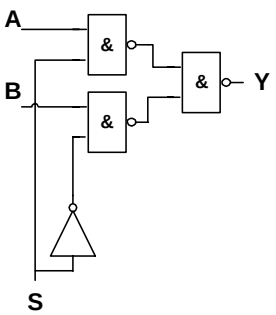
Exercise

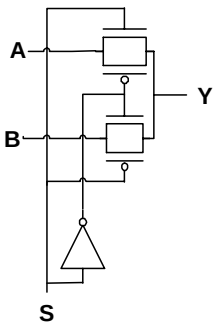
- **Discuss what happens when you connect the output of a single pass-transistor (not a pass-gate) to the input of another pass-transistor stage (i.e. the gate of another pass-transistor). Why should you never use such a circuit?**



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