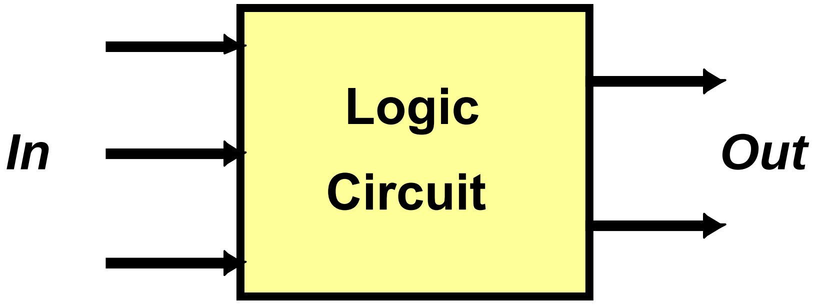
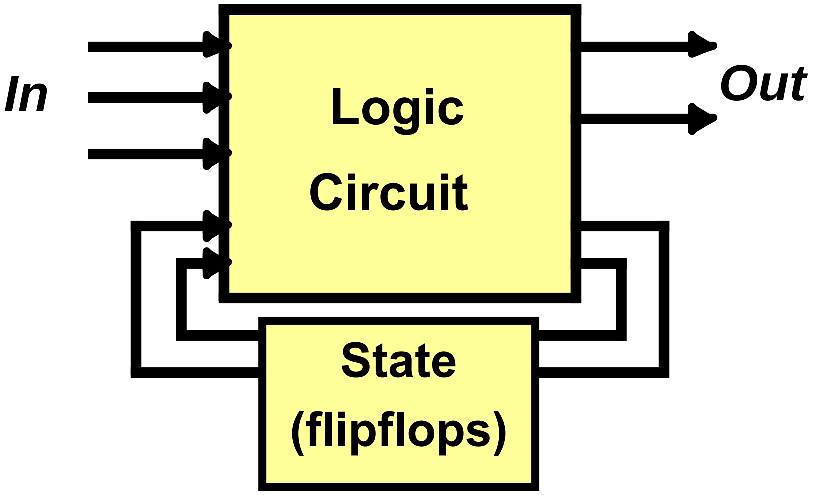
COMBINATIONAL LOGIC

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

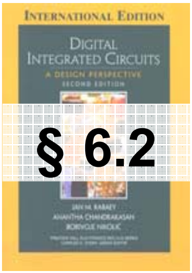
Combinational vs. Sequential Logic





(a) Combinational

(b) Sequential



§ 6.2

Output = *f* (In)

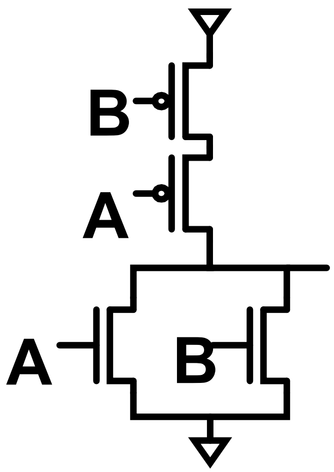
Output = *f* (In, History)

Complementary static CMOS

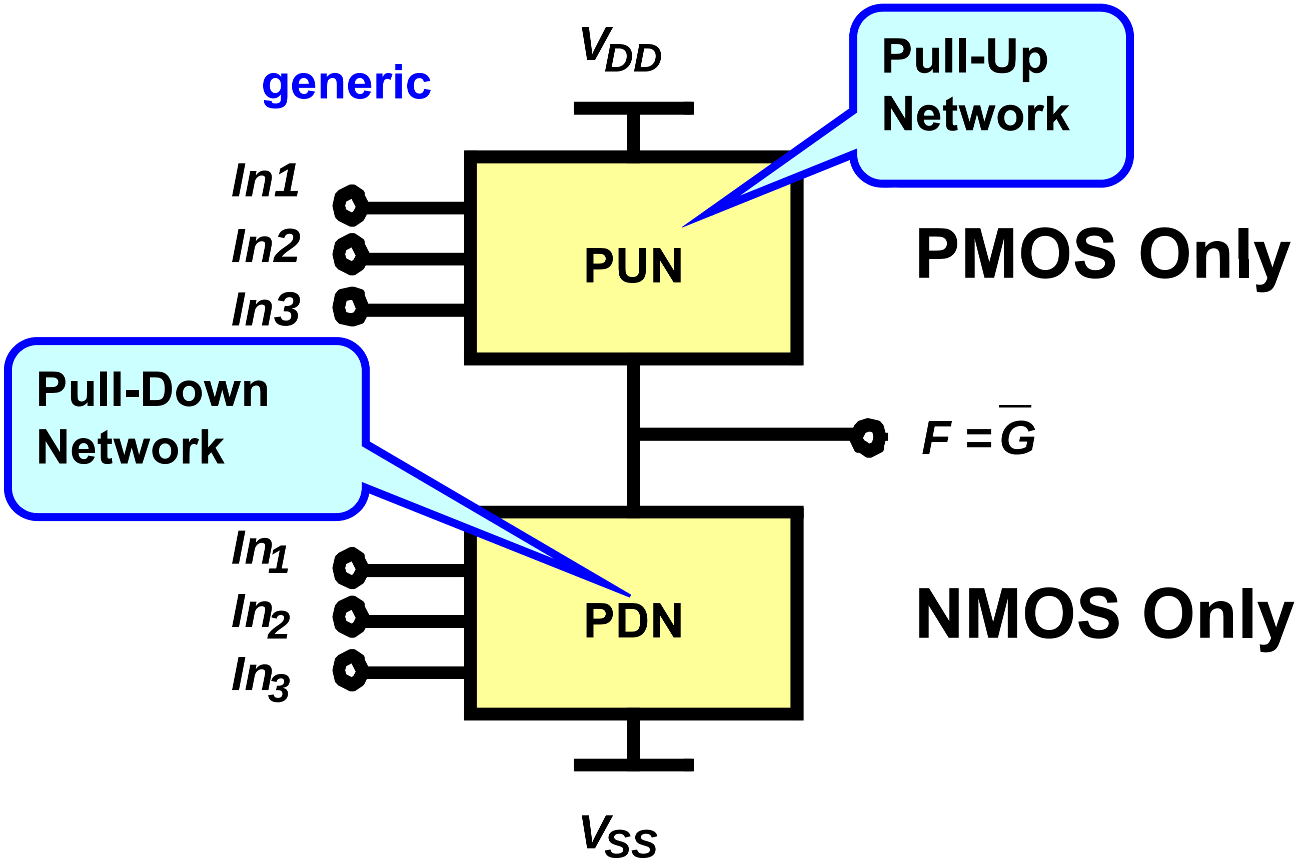
- Complex Logic Gates
- VTC, Delay and Sizing

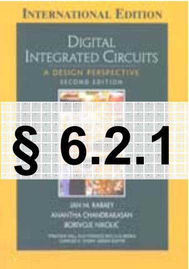
Complementary Static CMOS

example



generic

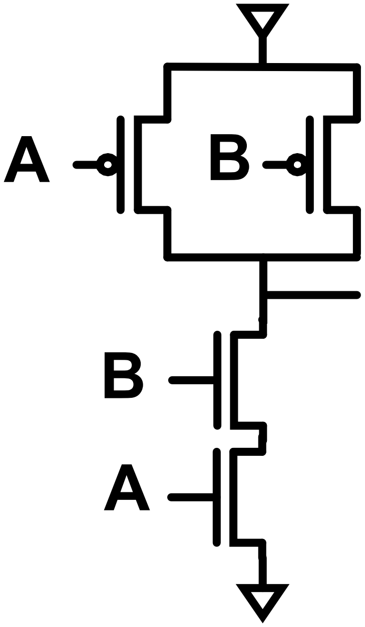




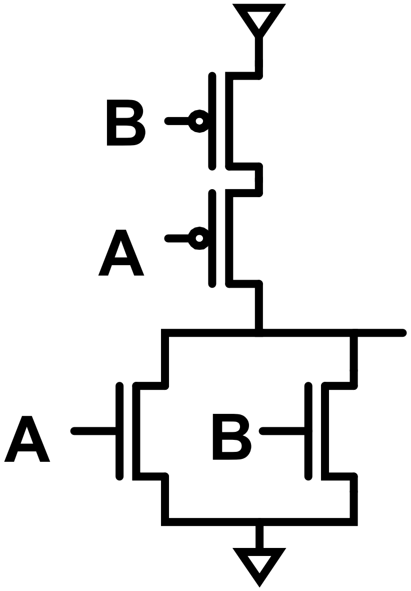
§ 6.2.1

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

2-input Nand/Nor



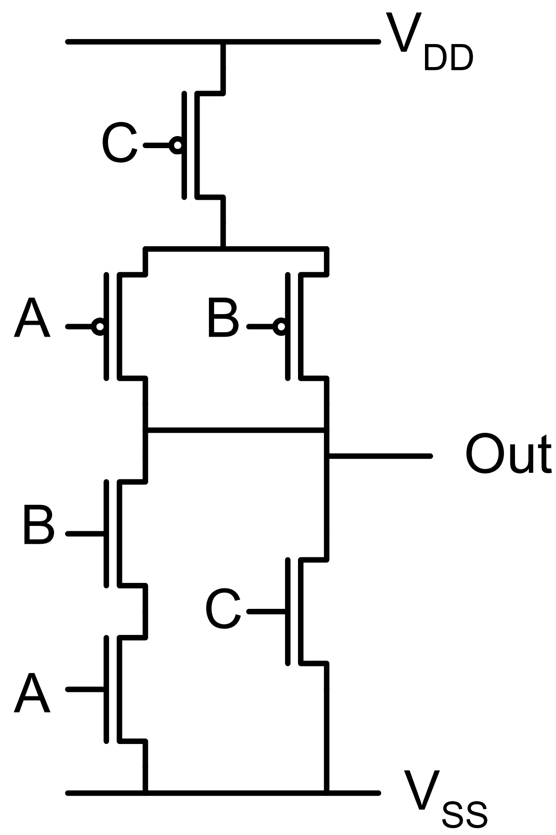
$Y = \overline{\mathbf{A} \text{ AND } \mathbf{B}}$



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

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.

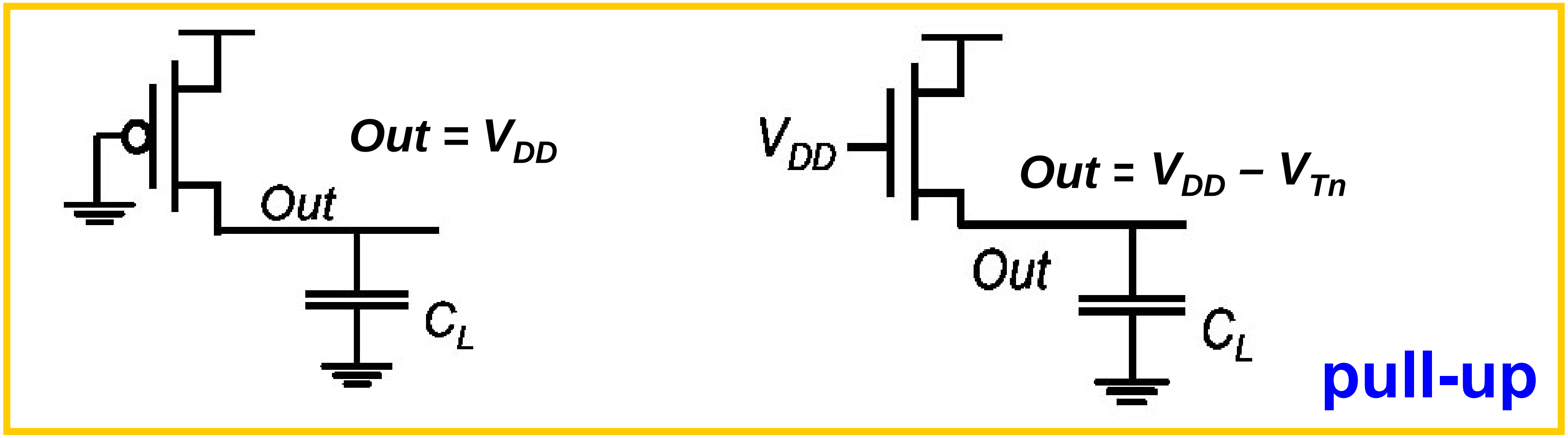
Complementary Static CMOS (2)

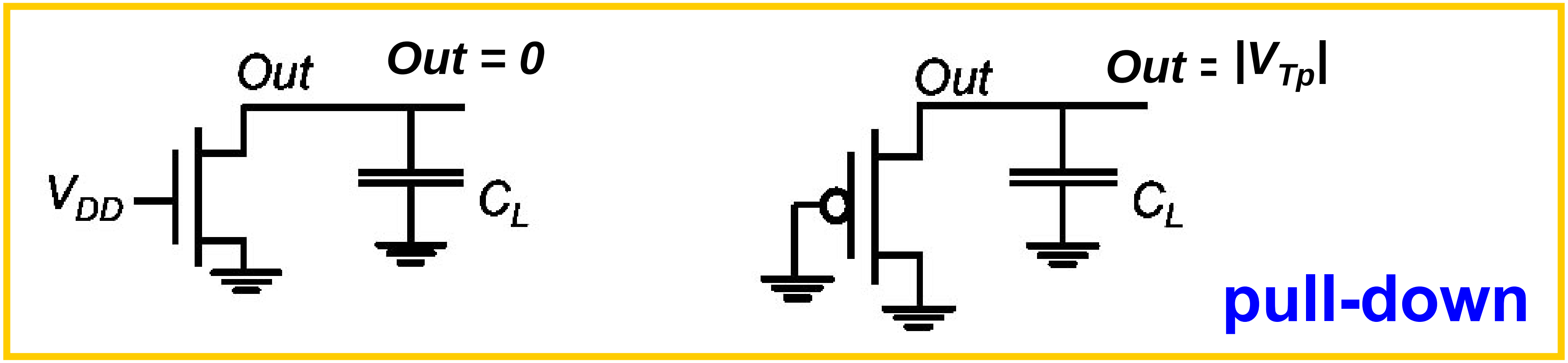
- **Conduction of PUN and PDN must be mutually exclusive**
- **PUN is dual (complement) network of PDN**

  series ⇔ parallel

  nmos ⇔ pmos
- **Complementary gate is inverting**
- **No static power dissipation**
- **Very robust**
- **Wide noise margin**
- **Need 2N transistors for N-input gate**

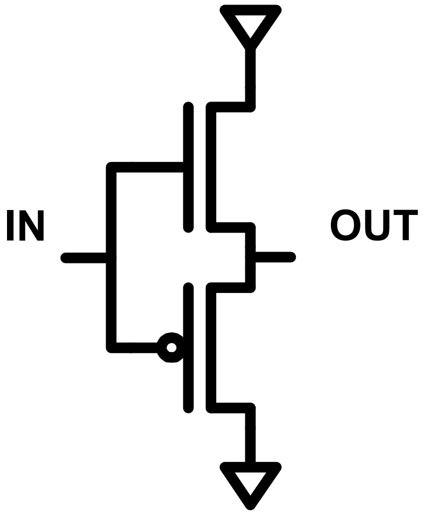
NMOS vs. PMOS, pull-down vs. pull-up





- **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

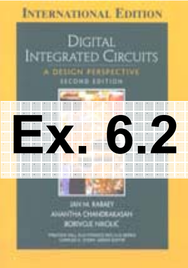
For a non-inverting Complementary CMOS Gate, you can only use 2 inverting gates

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

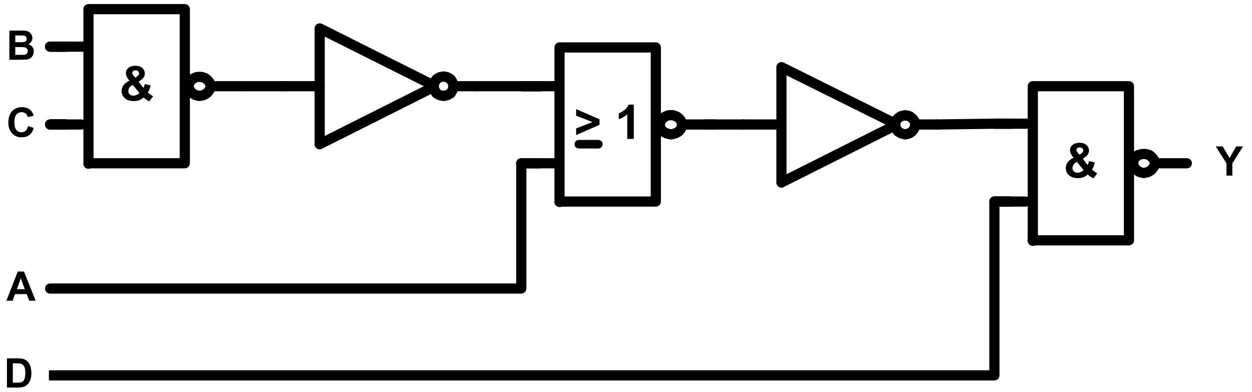
- **Remember: only inverting gates available**



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

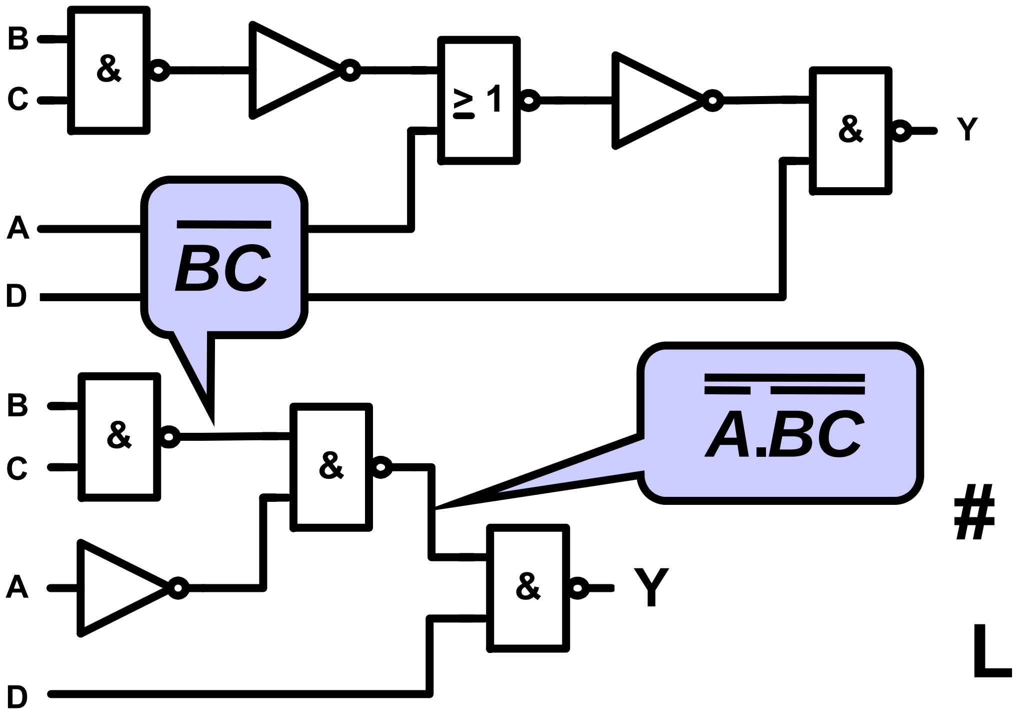


transistors □ logic depth □

- {TPS}: Can this be improved? If so, how?

Improved Gate Level Implementation

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



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

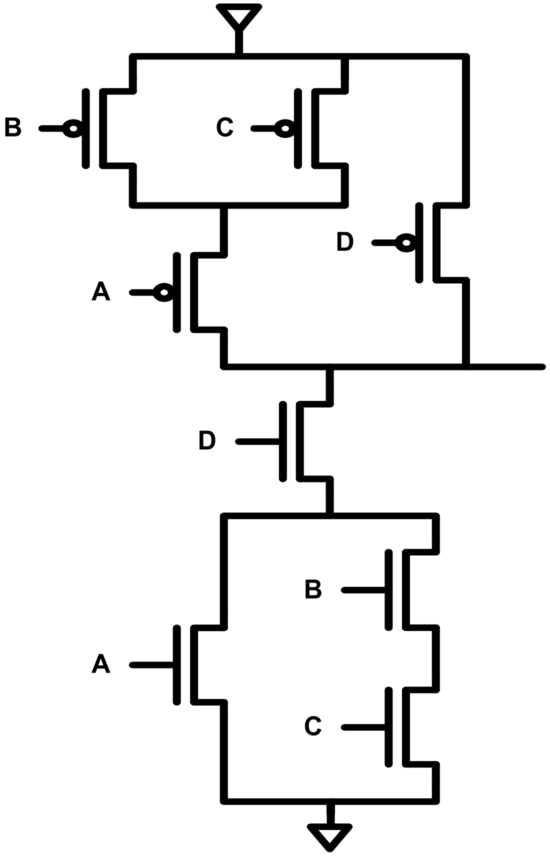
# transistors □

Logic depth □

- **{TPS}: Can this be further improved?**

Complex CMOS Logic Gates

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



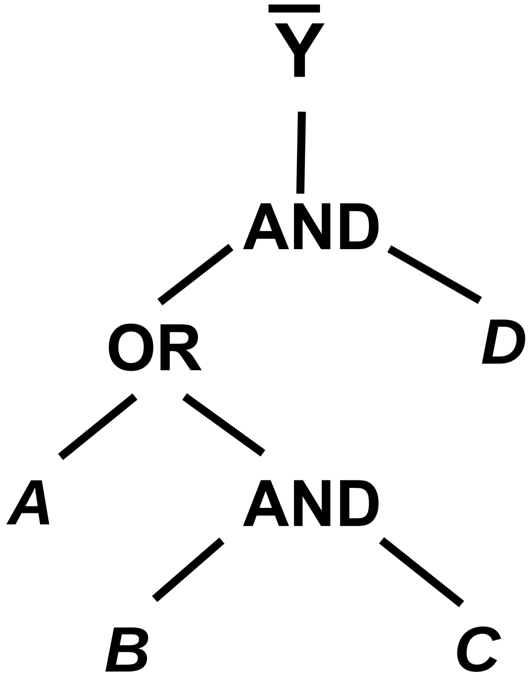
$$Y = \overline{(A + BC)D}$$

# transistors: 8

Logic depth: 1

How to Synthesize Complex Gates

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

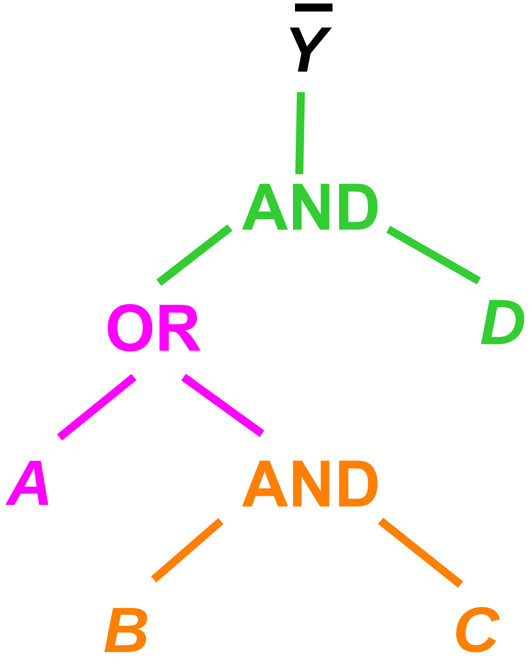


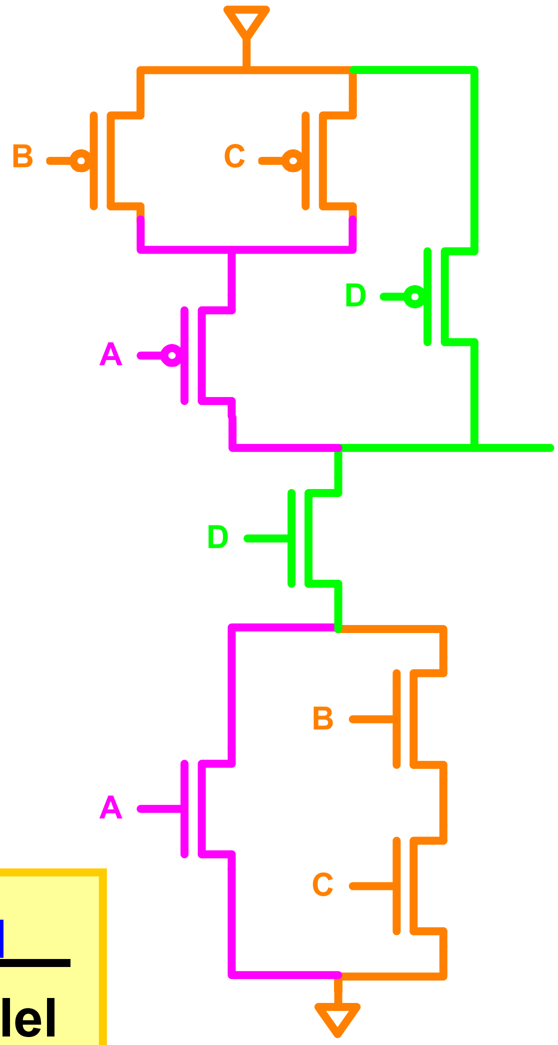
- Using tree representation of Boolean function
- ***Operator*** with branches for ***operands***
- As a **series-parallel** network

	PDN	PUN
AND	Series	Parallel
OR	Parallel	Series

Complex Gate Synthesis Example

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



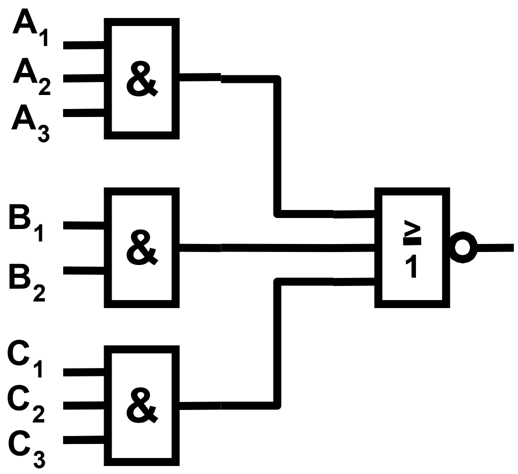


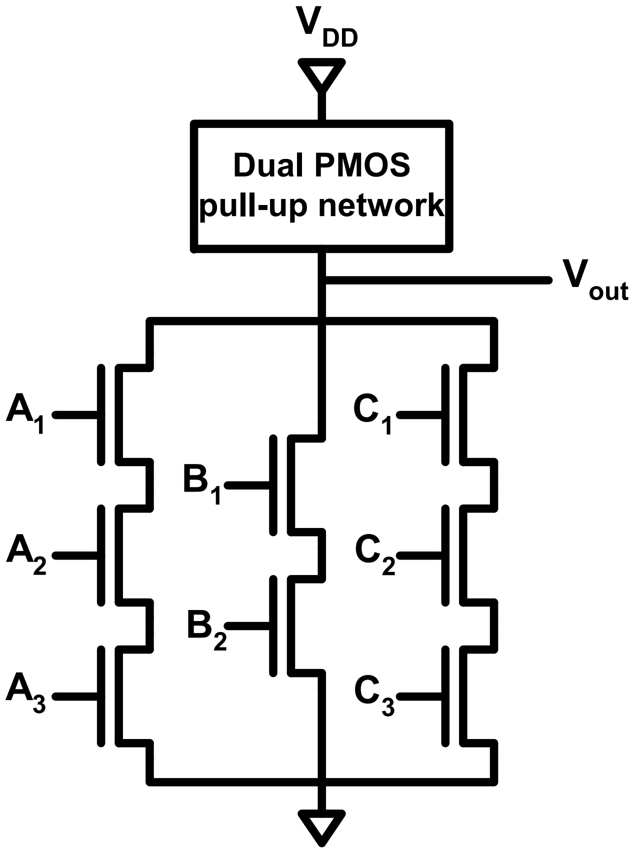
	PDN	PUN
AND	Series	Parallel
OR	Parallel	Series

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

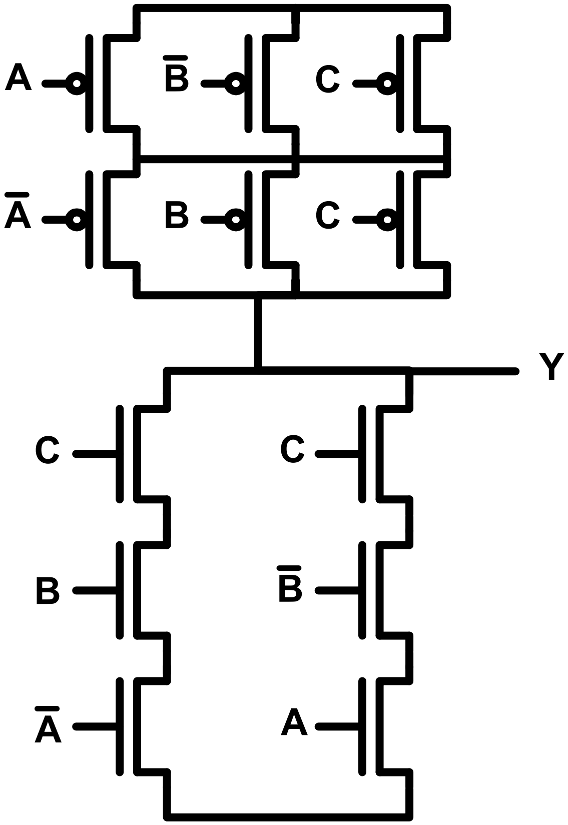




And-Or-Invert Example

- From a Truth-Table: take 0-outputs

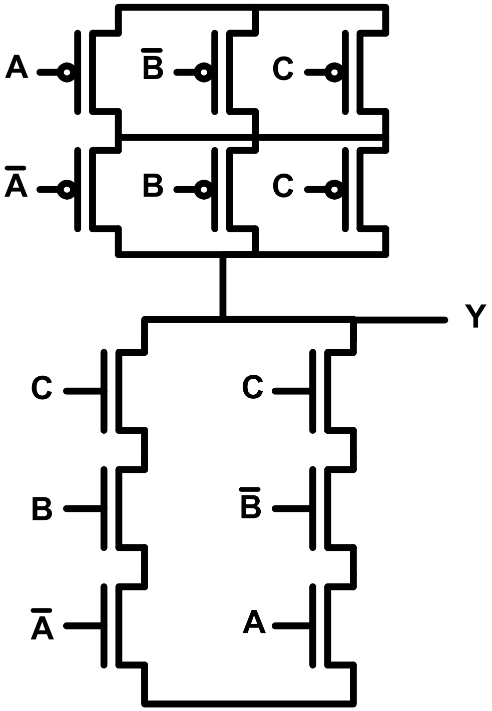
A	B	C	Y	
0	0	0	1	
0	0	1	1	
0	1	0	1	
0	1	1	0	→ $\bar{A}BC$
1	0	0	1	
1	0	1	0	→ $A\bar{B}C$
1	1	0	1	
1	1	1	1	

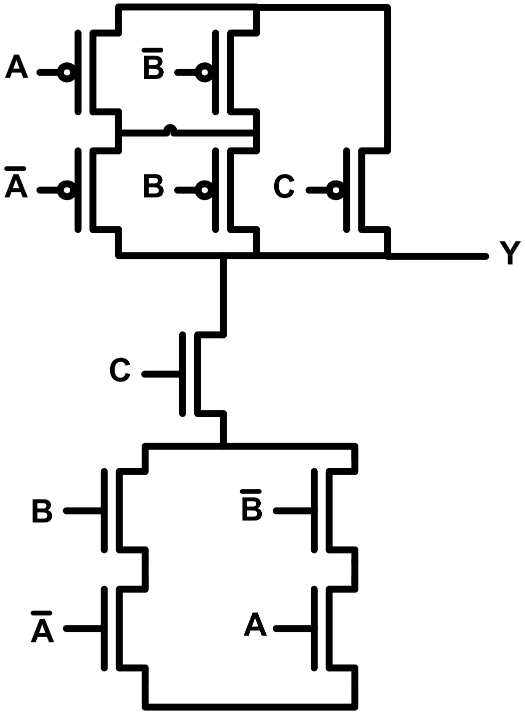


$$\bar{Y} = \bar{A}BC + A\bar{B}C$$

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

And-Or-Invert Improvement





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

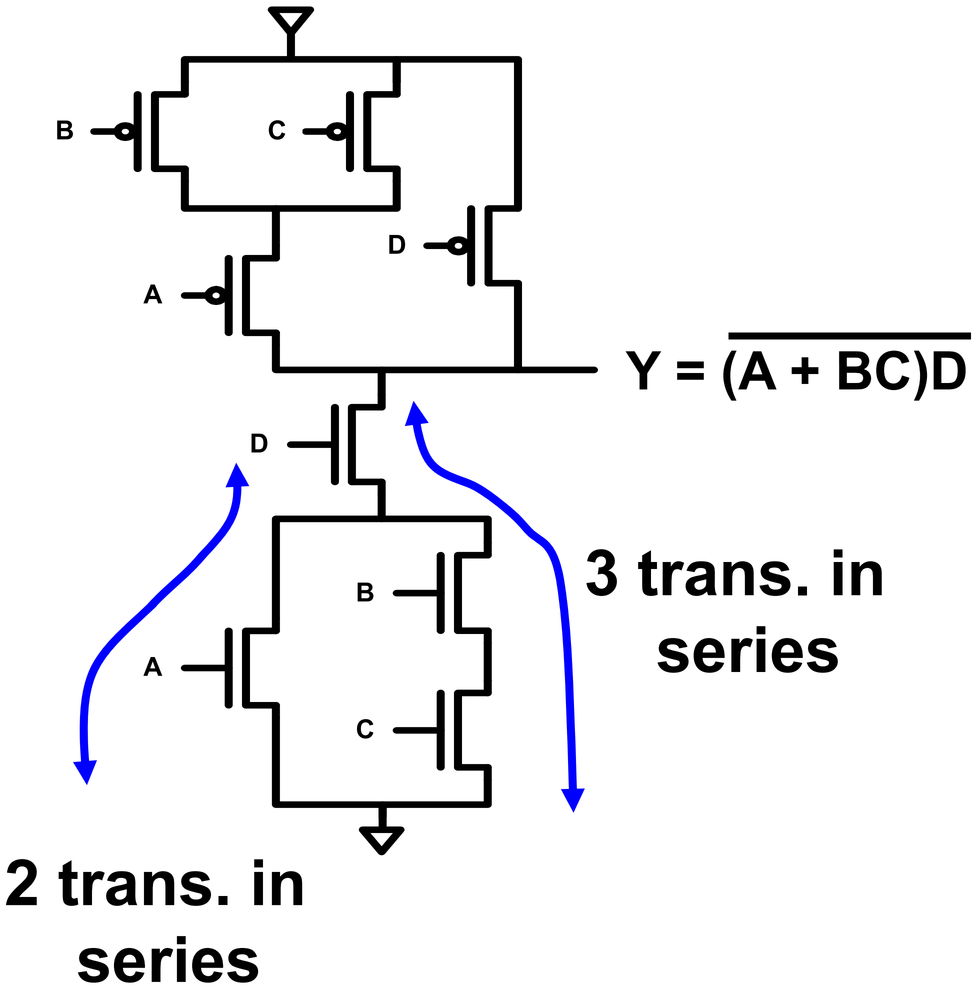
12 transistors

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

10 transistors

2-level logic minimization: boolean algebra technique

CMOS Complex Gate Sizing

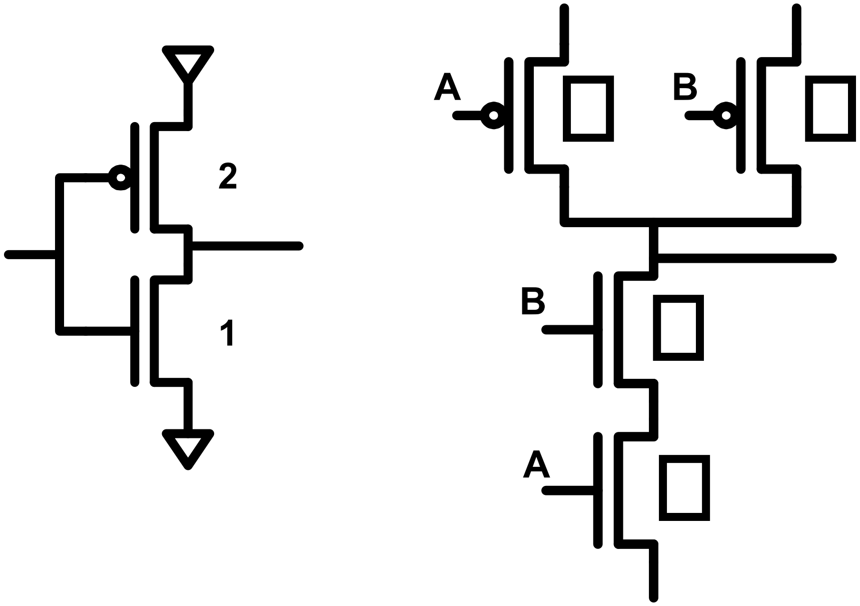


- Function of gate independent of transistor sizes: ratio-less
- But current-drive capability (timing) depends on transistor sizes
- Worst-case current-drive depends on number of transistors in series

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.**

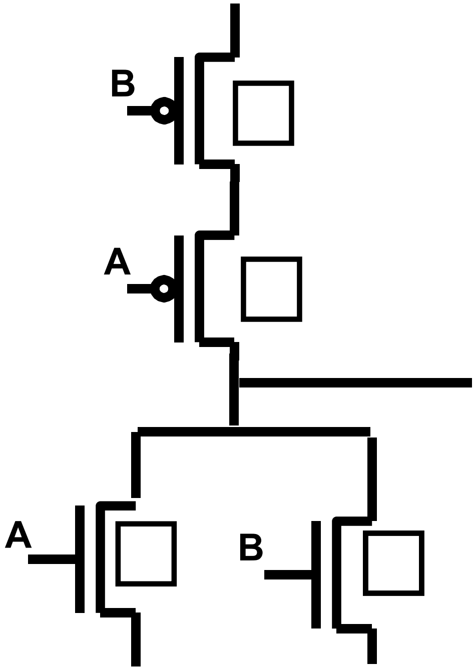
Gate Sizing



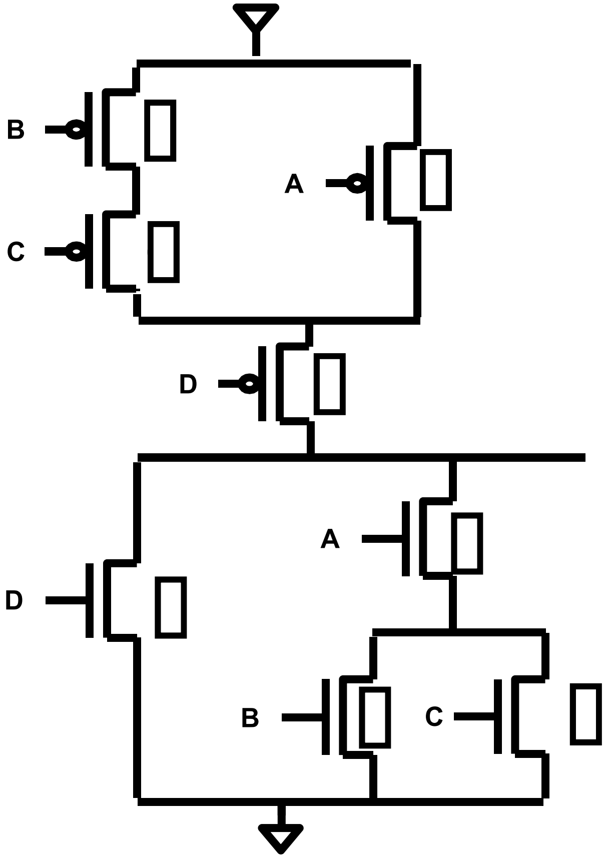
- **W/L ratios**
- **what are the W/L of 2-input NAND for the same drive strength?**

0-th order calculation

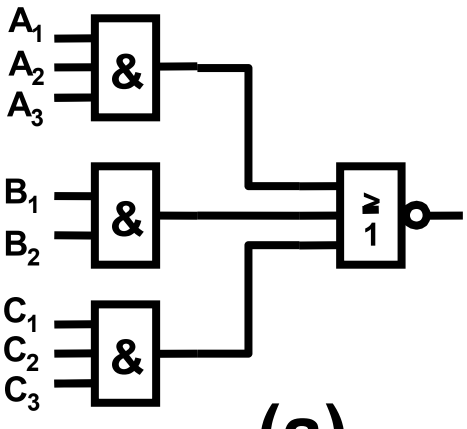
Gate Sizing



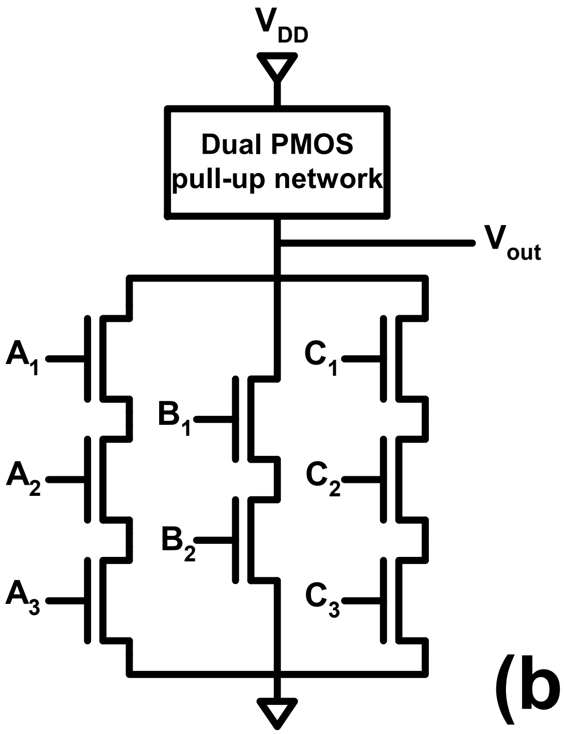
$$Y = \overline{A\,(B+C)+D}$$



Exercise



(a)

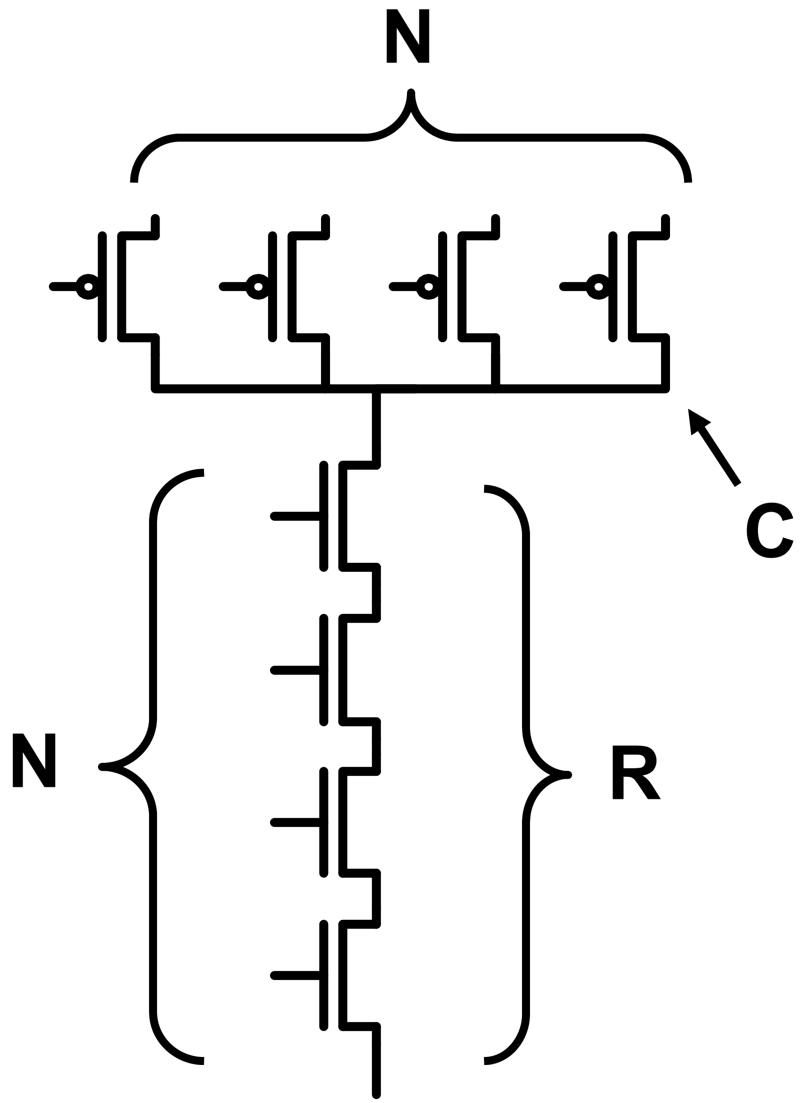


(b)

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

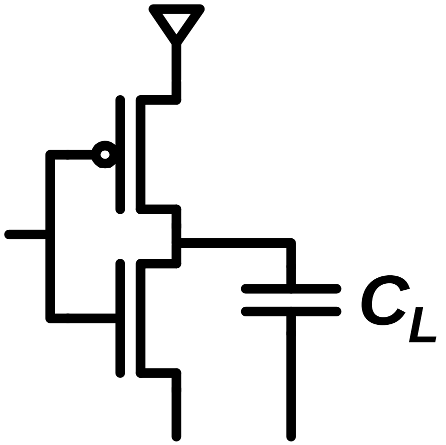
Delay $\propto$ RC **quadratic** in N

- -Transistor Sizing
- -Progressive Transistor Sizing
- -Input Re-Ordering
- -Logic Restructuring

Empirical

$$\text{Delay} = a_1\text{FI} + a_2\text{FI}^2 + a_3\text{FO}$$

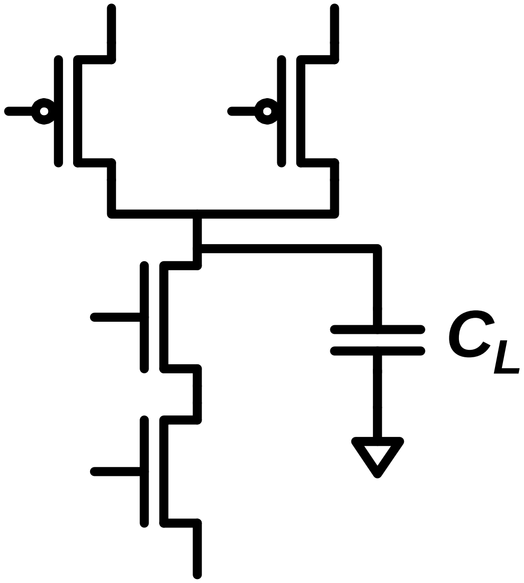
Data-Dependent Timing



$$t_{PHL} = 0.69R_NC_L$$

$$t_{PLH} = 0.69R_PC_L$$

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

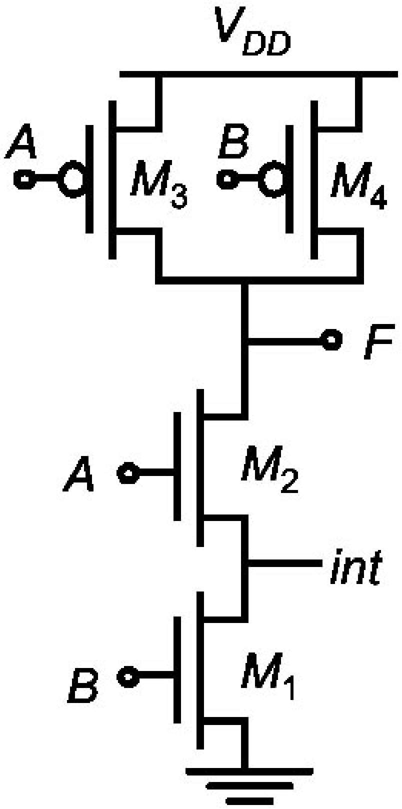
$t_{PLH} = 0.69(R_p/2)C_L$ Two inputs go low, parallel connection

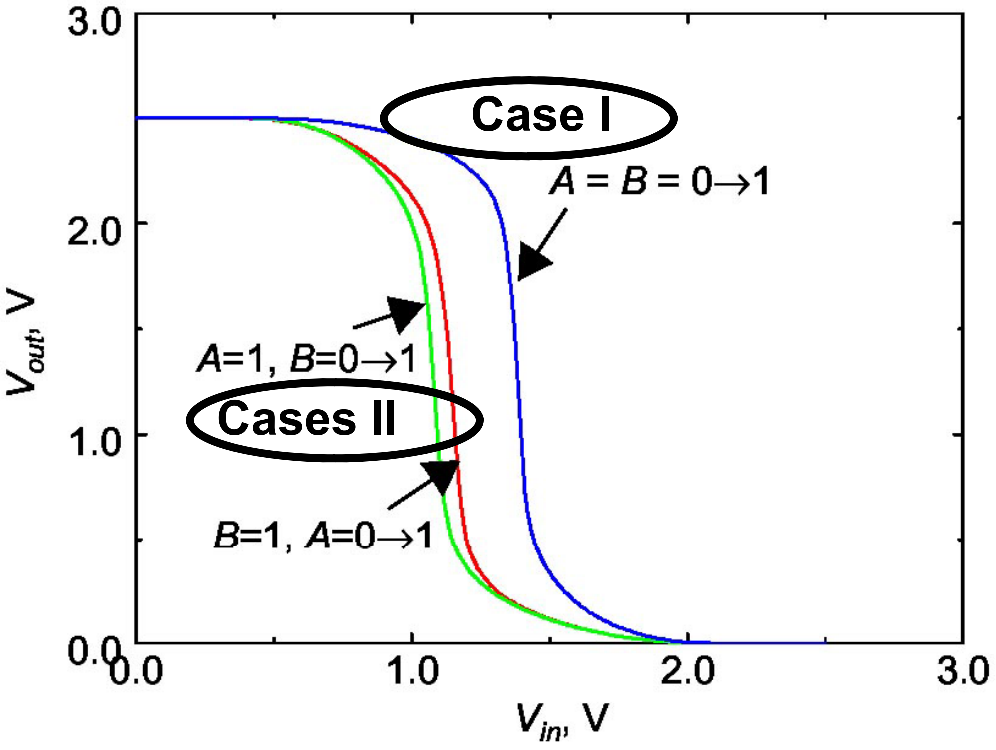
2^{nd} Order Effects

- **Much more to say about performance of static gates**
- **Simulator can give accurate answer**
- **Understanding needed to make design decisions**

- **Data-dependent VTC**
- **Data-dependent Timing**

Data-dependent VTC: 2nd order effects

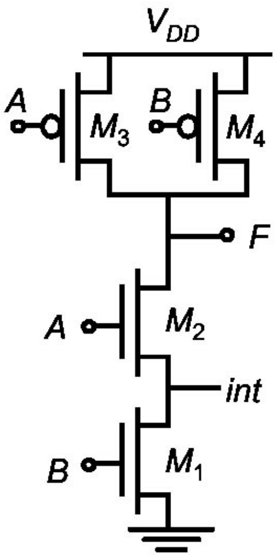


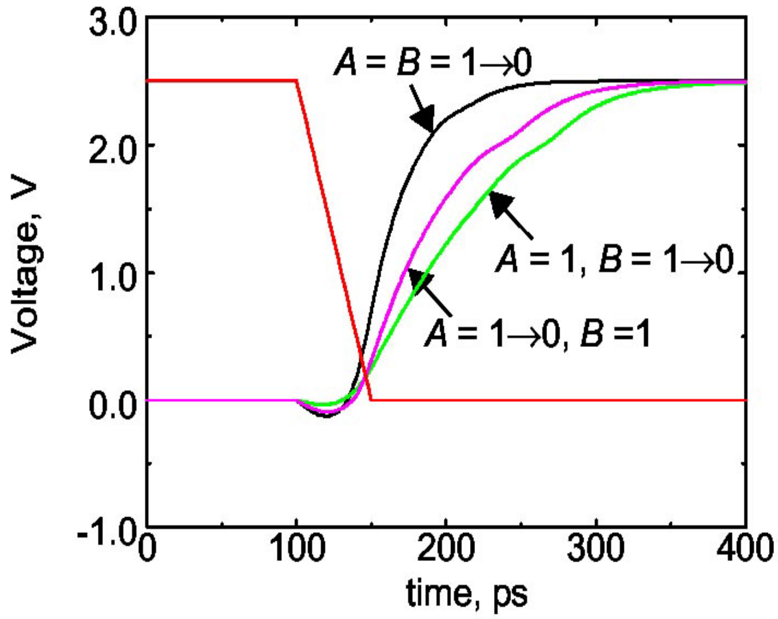


- Charge at 'int'
- Body effect in M_2
- Short-circuit currents

{TPS}:
Explain VTC difference between I and II

Data-dependent Timing





Input Data Pattern	Delay (pS)
$A = B = 0 \to 1$	69
$A = 1, B = 0 \to 1$	62
$A = 0 \to 1, B = 1$	50
$A = B = 1 \to 0$	35
$A = 1, B = 1 \to 0$	76
$A = 1 \to 0, B = 1$	57

A=1, B=↓: need to charge *int*

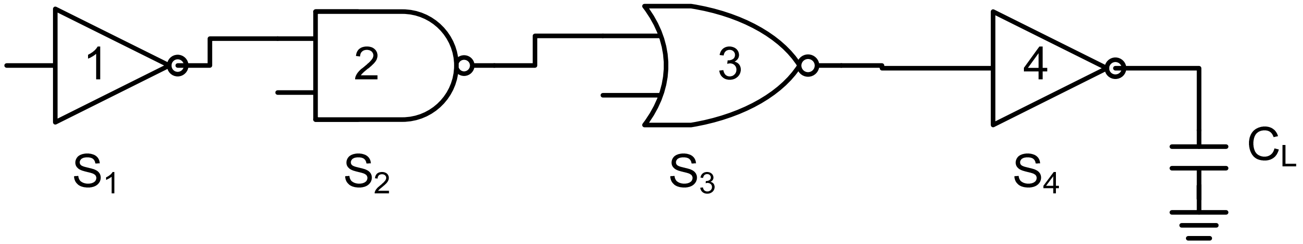
A=↓, B=1: *int* does not need to be charged

A=↓, B=↓: twice the pull-up strength

{TPS}:
Explain differences in t_{pLH}

LOGICAL EFFORT

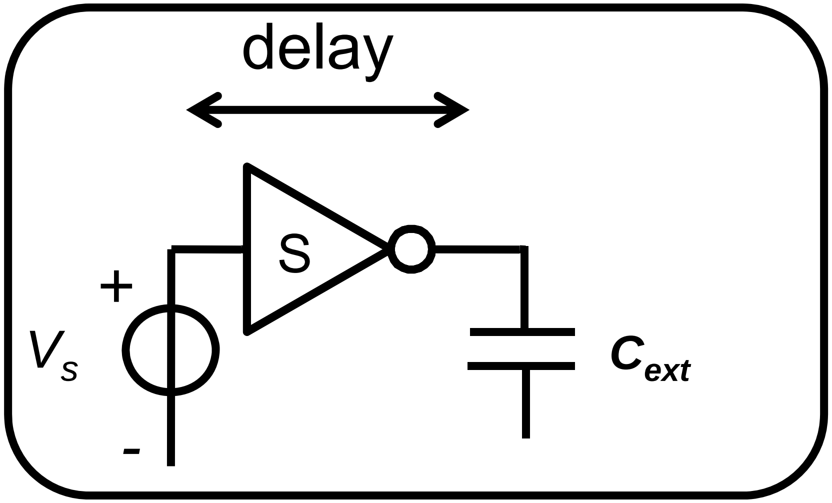
Combinational Path Sizing for Timing



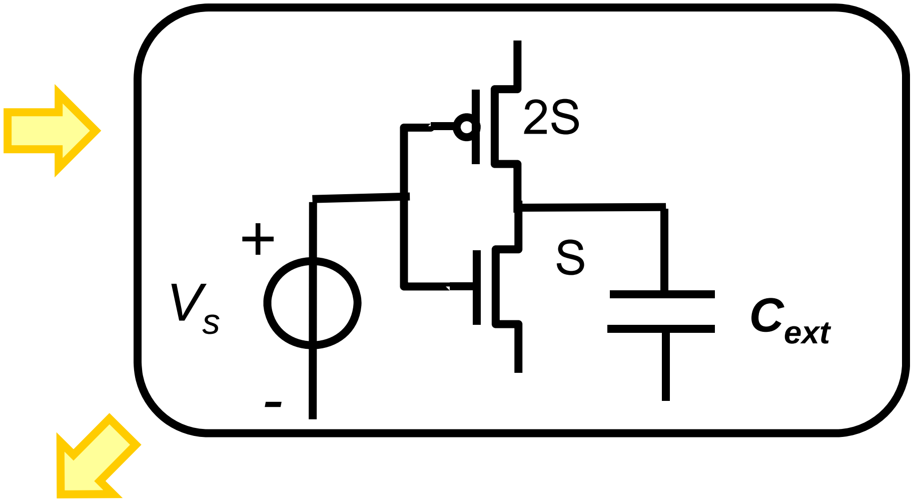
Given: C_L, S_1
Determine S_2, S_3, S_4 to minimize delay

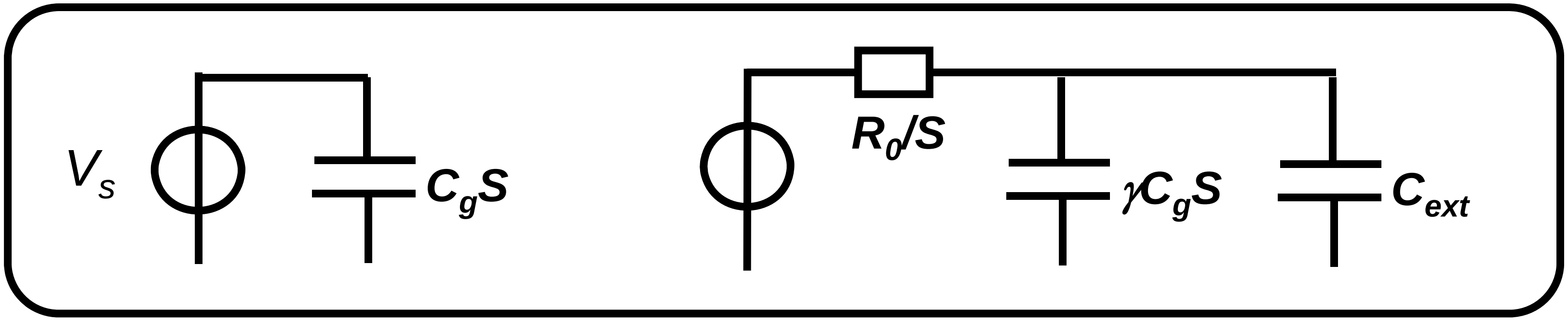
We know how to optimally size string of inverters:
make equal stage delays
{TPS}: What is different in comparison to string of inverters?

Recap: Inverter Delay



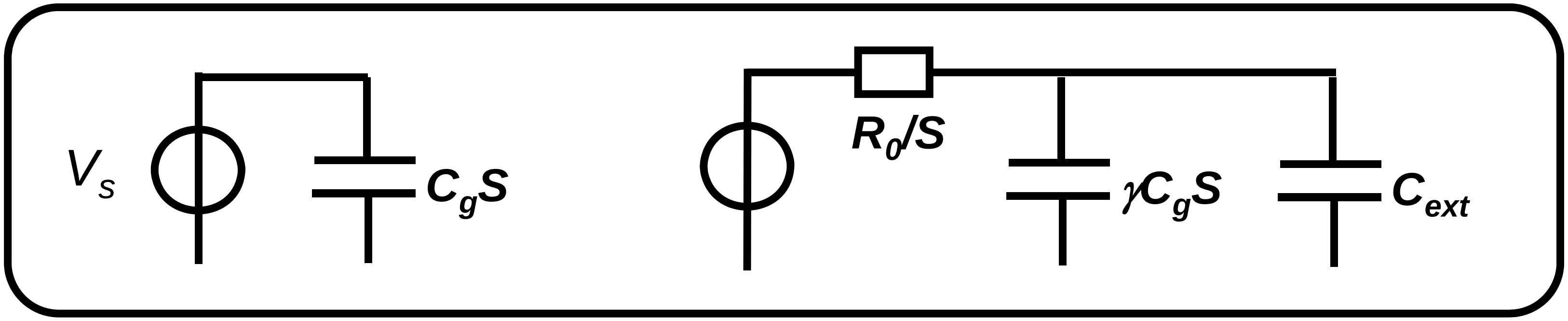
S: relative size of inverter





R_0, C_g, γC_g : output res, input cap and output cap of min size inverter

Recap: Inverter Delay



R_0, C_g, γC_g : output res, input cap and output cap of min size inverter

$$t_p = \frac{R_0}{S}\left(\gamma C_g S + C_{ext}\right)$$

$$= \gamma R_0 C_g\left(1 + \frac{C_{ext}}{\gamma S C_g}\right) = t_{po}\left(1 + \frac{C_{ext}}{\gamma C_{in}}\right) \quad \text{with } C_{in} = SC_g$$

$$= t_{po}\left(1 + \frac{f}{\gamma}\right) \qquad \text{with } t_{po} = \gamma R_0 C_g \qquad f = \frac{C_{ext}}{C_{in}}$$

t_{p0}: Delay of unloaded inverter, independent of sizing

Inverter Delay Summary

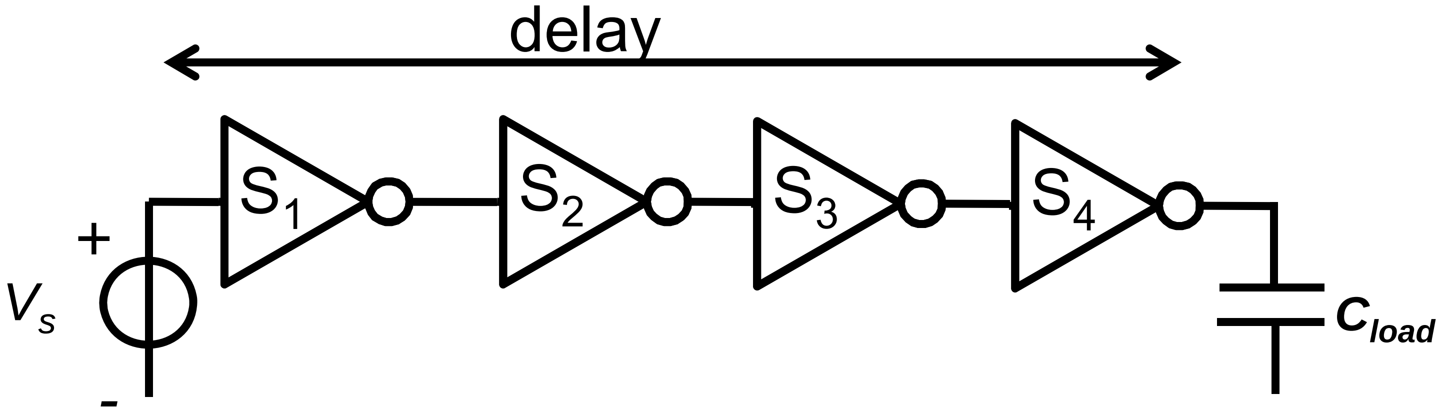
$$\boxed{t_p = \frac{R_0}{S}\left(\gamma C_g S + C_{ext}\right) = t_{po}\left(1 + \frac{f}{\gamma}\right) \qquad \text{with } t_{po} = \gamma R_0 C_g, f = \frac{C_{ext}}{C_{in}}}$$

$$\boxed{d = 1 + f/\gamma}$$

In units of t_{p0}

R_0	Equivalent output resistance of min size inverter
C_g	Input cap of min size inverter
$C_0 = \gamma C_g$	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}
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}

S_i: relative sizes of inverters



Path delay is minimized if all stage delays are equal

For string of inverters:
when ratio of load cap over input cap is identical for each stage

If C_g = input cap of inverter with size 1 (minimum size):

$$\frac{S_2 C_g}{C_{in}} = \frac{S_2 C_g}{S_1 C_g} = \frac{S_2}{S_1} = \frac{S_3}{S_2} = \frac{S_4}{S_3} = \frac{C_{load}}{S_4 C_{inv}}$$

Logical Effort Methodology

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

In units of t_{p0}

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

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

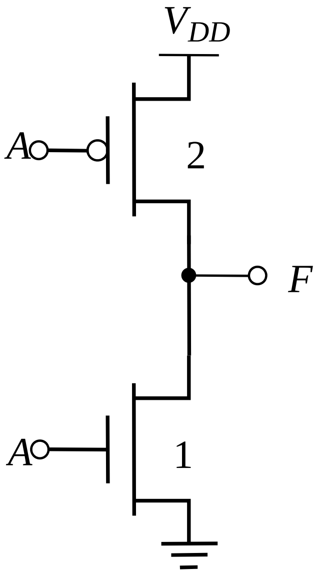
- how much more load the gate creates
- ratio of input cap for same drive strength

h gate effort, *h* = *gf*

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

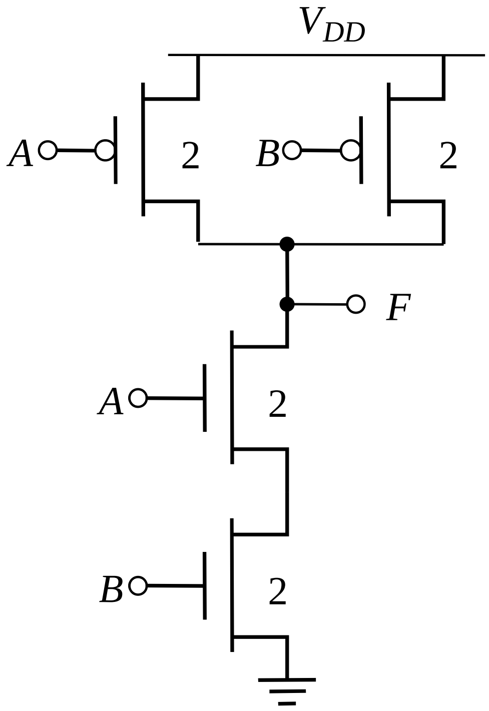
Beware: compared to most texts, incl. Sutherland, Rabaey swaps definition of *f* and *h*

Intrinsic, Parasitic Delay



Inverter

p=1



2-input NAND

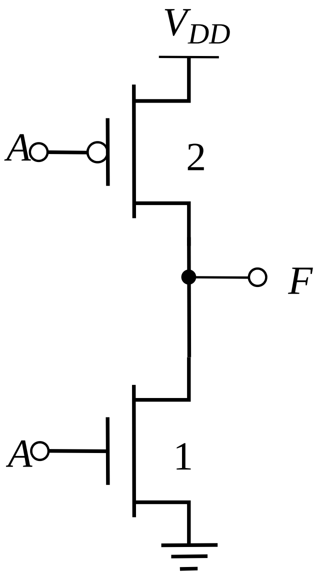
p=2

- ***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**

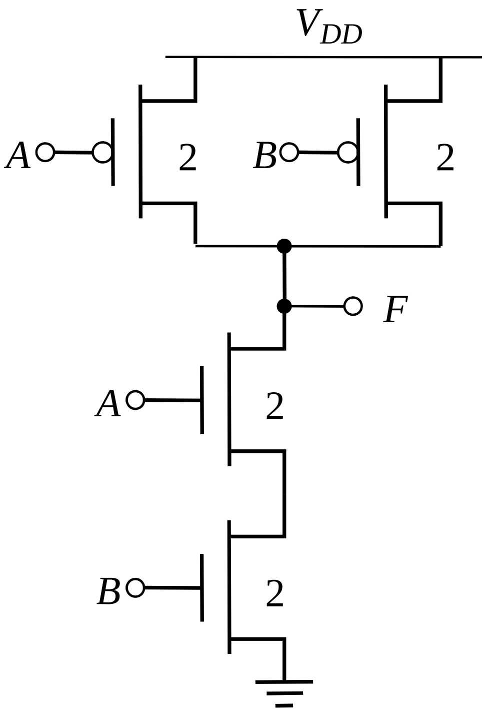
$$d_{gate} = p + gf/\gamma$$

Logical Effort



Inverter

g=1



2-input NAND

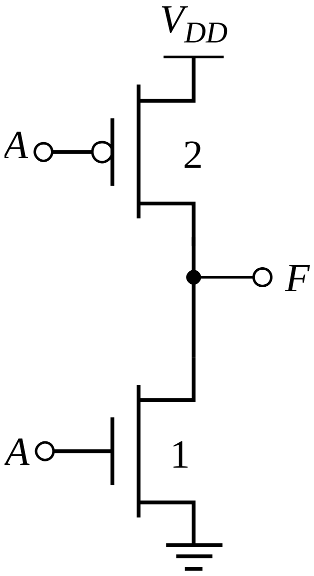
g=4/3

- ***g*** **logical effort: how much load a gate provides relative to inverter for same drive strength**
- **g ratio of input cap (per pin) if gate is sized for identical drive strength**

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

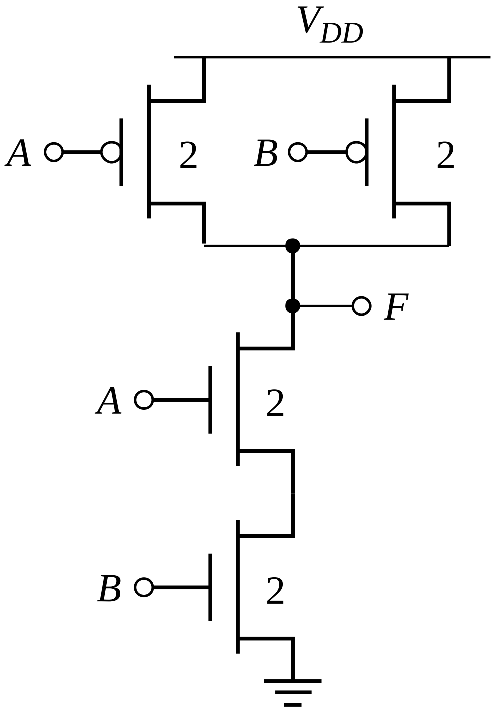
$$d_{gate} = p + gf/\gamma$$

Logical Effort



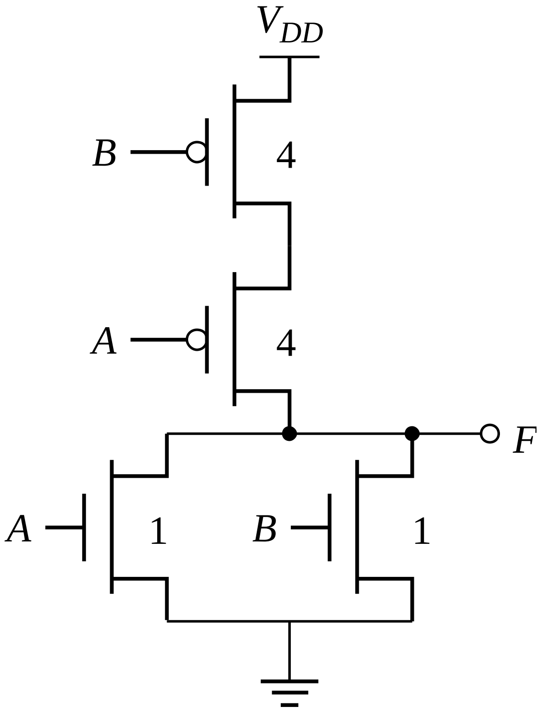
Inverter

p=1, g=1



2-input NAND

p=2, g=4/3



2-input NOR

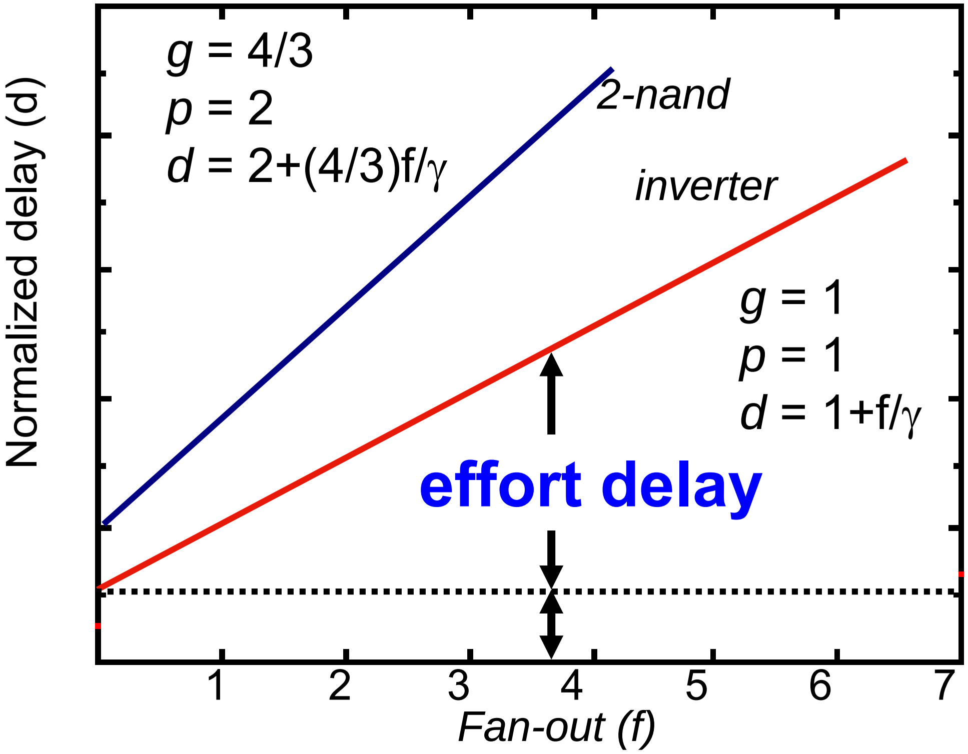
p=2, g=5/3

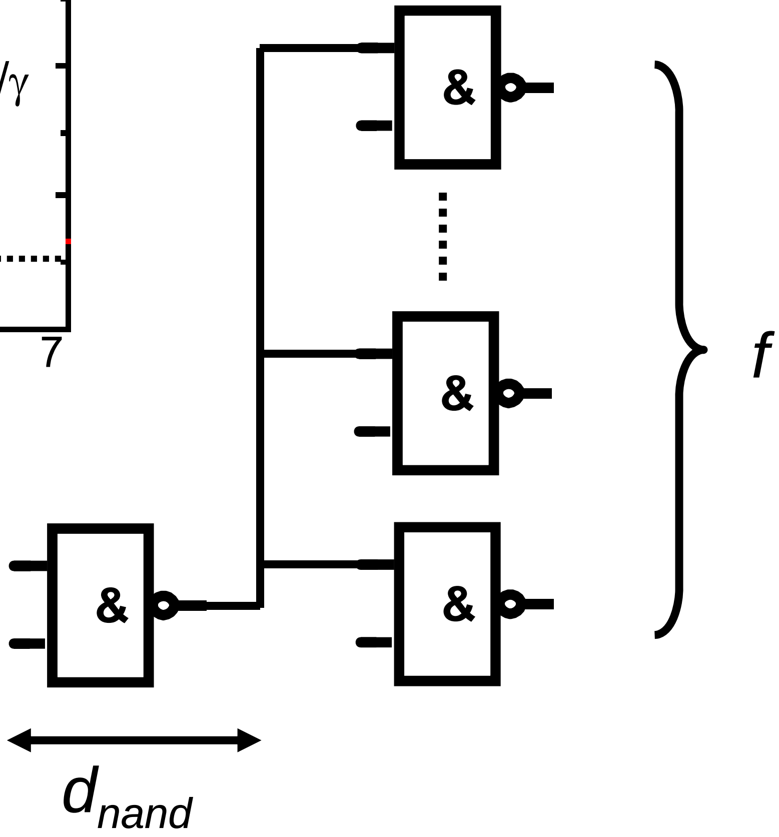
p **ratio of intrinsic delay compared to inverter**

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

p, g independent of sizing, only topology of gate

Delay vs Fan-Out





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

Multistage Networks

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

Normalized w.r.t. unit delay, assume γ = 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, optimal effort h*:

$$h_*^N = H$$

$$h_* = \sqrt[N]{H}$$

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

Effective fanout of each stage: $f_i = h_* / g_i$

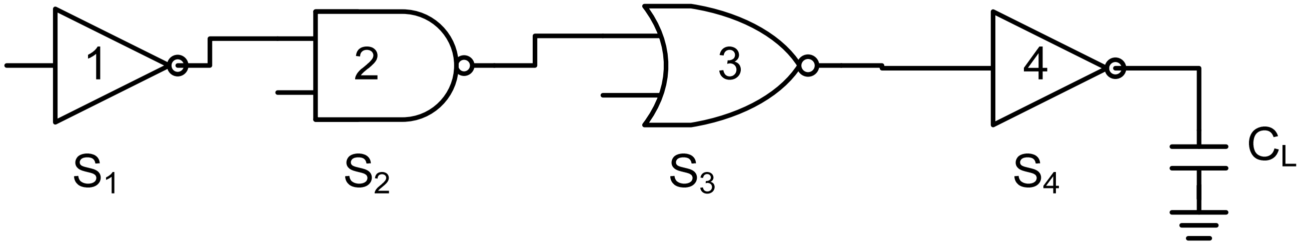
larger fanout for simpler stages

Minimum path delay

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

Combinational Path Sizing for Timing

S_1 = 1, C_L = 36.45



g_1=1	g_2=4/3	g_3=5/3	g_4=1	$G = \Pi g_i = 20/9$
$f_1 = S_2$	$f_2 = S_3/S_2$	$f_3 = S_4/S_3$	$f_4 = 36.45/S_4$	$F = \Pi f_i = 36.45$

H = FG = 81 $$h_* = \sqrt[N]{H} = \sqrt[4]{81} = 3$$

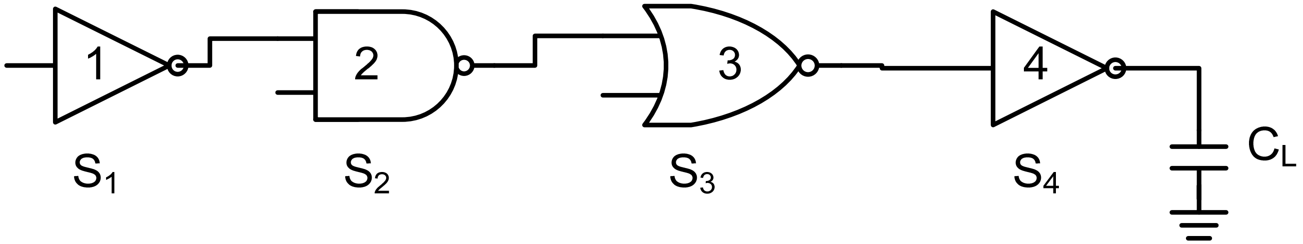
$f_1 g_1 = 3 \Rightarrow S_2 = 3$

$f_2 g_2 = 3 \Rightarrow S_3 = 27/4 = 6.75$

$f_3 g_3 = 3 \Rightarrow S_4 = 12.15$

Combinational Path Sizing for Timing

S_1 = 1, C_L = 36.45



$f_1g_1=3 \Rightarrow S_2=3$ $f_2g_2 = 3 \Rightarrow S_3 = 27/4 = 6.75$ $f_3g_3 = 3 \Rightarrow S_4 = 12.15$

	INV	**NAND**	**NOR**	**INV**
Width of N:	1	3x3/2 = 4.5	3x6.75/5 = 4.05	3x12.15/3=12.15
Width of P:	2	3x3/2 = 4.5	3x4x6.75/5 = 16.2	3x2x12.15/3 = 24.3
Nrmlzd C_{in}	1	9 / 3 = 3	20.25 / 3 = 6.75	36.45 / 3 = 12.15

C_L= 36.45 C_{in}

$f_1 = 3$ $f_2 = 3/g_2$ $f_3 = 3/g_3$ $f_4 = 3$

Logical Effort - Summary

- **Numerical logical effort characterizes gates**
- **Inverters and NAND2 best for driving large caps**
- **NANDs are faster than NORs in CMOS**
- **Extension needed (see book) for branching**
- **Simplistic delay model**
  - **Neglects input rise time effects**
- **Interconnect**
  - **Iteration required in designs with wire**
- **Maximum speed only**
  - **Not minimum area/power for constrained delay**

DYNAMIC POWER DISSIPATION

Dynamic Power Dissipation

Power = Energy/transition • Transition rate

$$= C_L V_{DD}^2 \bullet f_{0\rightarrow 1}$$

$$= C_L V_{DD}^2 \bullet f \bullet p_{0\rightarrow 1}$$

$$= C_{switched} V_{DD}^2 \bullet f$$

- -Transistor Sizing
  - Physical capacitance
- -Input and output rise/fall times
  - Short-circuit power
- -Threshold and temperature
  - Leakage power
- -Switching activity

- **Power dissipation is data dependent – depends on the switching probability $p_{0\rightarrow 1}$**
- **Switched capacitance $C_{switched} = p_{0\rightarrow 1} C_L = \alpha\, C_L$ (α is called the switching activity)**

Transition Probability

$p_{A=1} = p_A$: given probability of value of signal A being 1 in any clock cycle

$p_{A=0} = p_{A'}$: given probability of value of signal A being 0 in any clock cycle

Note the ’ prime (for inversion of signal)

Transition probability:

Probability of 1 to 0 or 0 to 1 transition at clock edge: $\alpha = p_A(1-p_A) = p_A p_{A'}$

Impact of Logic Function

Example: Static 2-input NAND gate

A	*B*	*Out*
0	0	1
0	1	1
1	0	1
1	1	0

Assume **signal probabilities**

$$p_{A=1} = 1/2$$

$$p_{B=1} = 1/2$$

Then **transition probability**

$$\alpha = p_{0\rightarrow 1} = p_{Out=0} \times p_{Out=1}$$

$$= 1/4 \times 3/4 = 3/16$$

If inputs switch every cycle

$$\boxed{\alpha_{NAND} = 3/16}$$

NOR gate yields similar result

Impact of Logic Function

Example: Static 2-input XOR Gate

A	*B*	*Out*
0	0	0
0	1	1
1	0	1
1	1	0

Assume **signal probabilities**

$$p_{A=1} = 1/2$$

$$p_{B=1} = 1/2$$

Then **transition probability**

$$p_{0\rightarrow 1} = p_{Out=0} \times p_{Out=1}$$

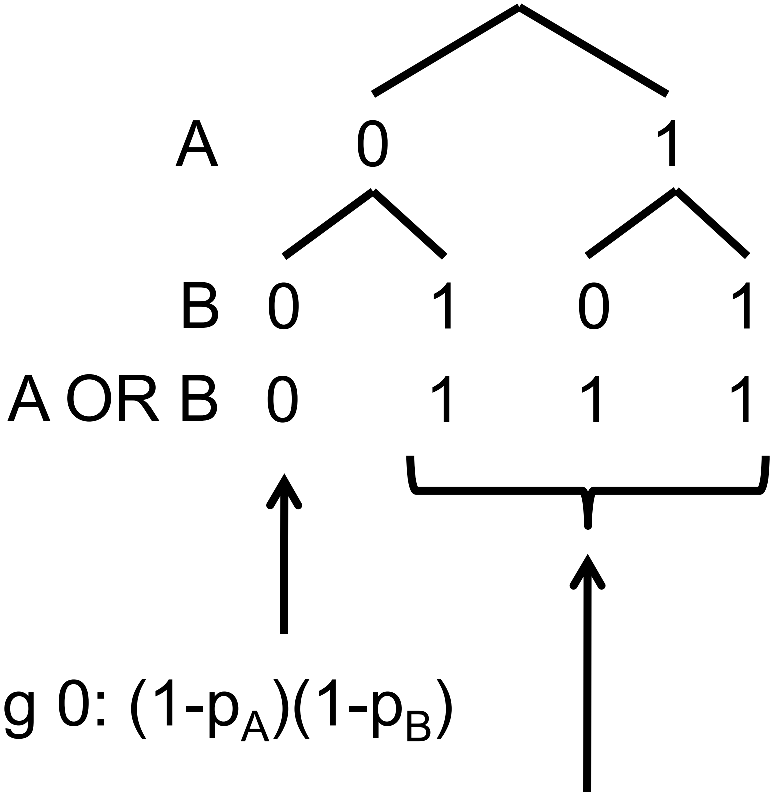
$$= 1/2 \times 1/2 = 1/4$$

If inputs switch in every cycle

$$\boxed{P_{0\rightarrow 1} = 1/4}$$

Signal and Transition Probabilities
OR Gate

p_A: Probability of A being 1
p_B: Probability of B being 1



A	0		1	
B	0	1	0	1
A OR B	0	1	1	1

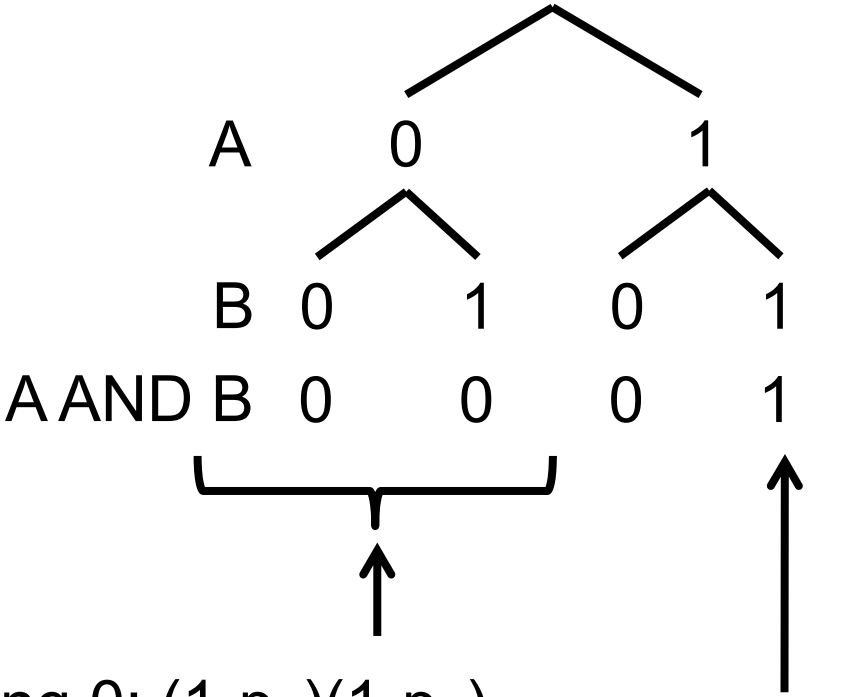
Probability of Output being 0: $(1-p_A)(1-p_B)$

Probability of Output being 1: $(1-(1-p_A)(1-p_B))$

Transition probability: $\alpha=(1-p_A)(1-p_B)(1-(1-p_A)(1-p_B))$

Signal and Transition Probabilities
AND Gate

p_A: Probability of A being 1
p_B: Probability of B being 1



A 0 1

B 0 1 0 1

A AND B 0 0 0 1

Probability of Output being 0: $(1-p_A)(1-p_B)$

Probability of Output being 1: $p_A p_B$

Transition probability: $\alpha = (1-p_A)(1-p_B)p_A p_B)$

Transition Probabilities for Basic Gates

As a function of the input probabilities

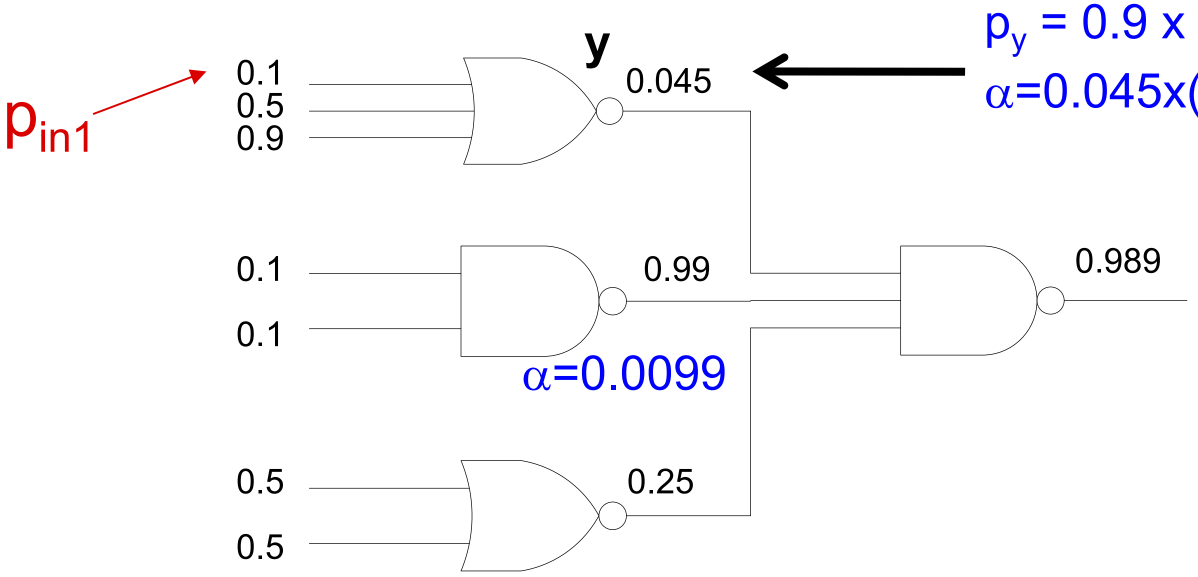
	$A = p_{\mathbf{0 \rightarrow 1}}$
AND	$(1 - p_A p_B) p_A p_B$
OR	$(1 - p_A)(1 - p_B)(1 - (1 - p_A)(1 - p_B))$
XOR	$(1 - (p_A + p_B - 2p_A p_B))(p_A + p_B - 2p_A p_B)$

Activity for static CMOS gates: $\alpha = p_0 p_1$

Because of symmetry: AND ⇔ NAND, OR ⇔NOR

Evaluating Power Dissipation of Complex Logic

Simple idea: start from inputs and propagate signal probabilities to outputs



p_y = 0.9 x 0.5 x 0.1= 0.045

α=0.045x(1-0.045) = 0.043

α=0.0099

But:

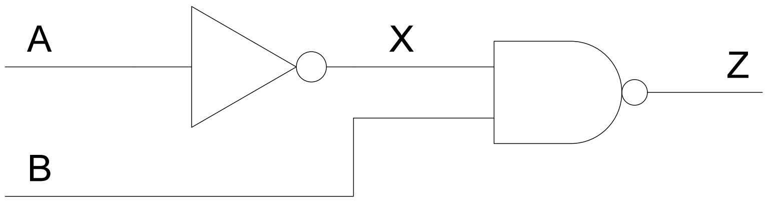
Reconvergent fanout

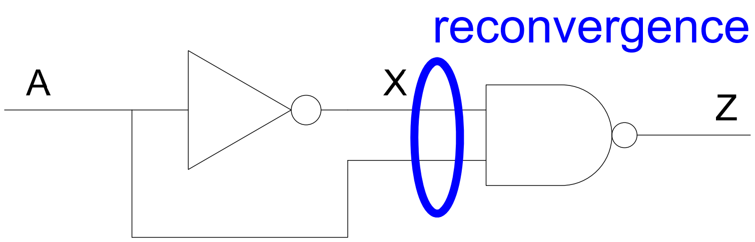
Feedback and temporal/spatial correlations

13/03/14

Reconvergent Fanout (Spatial Correlation)

Inputs to gate can be interdependent (correlated)





no reconvergence

$p_Z = 1-p_xp_B =1-(1-p_A)p_B$

reconvergent

$p_Z = 1-(1-p_A)p_A$?

NO!

$p_Z = 1$

Must use conditional probabilities

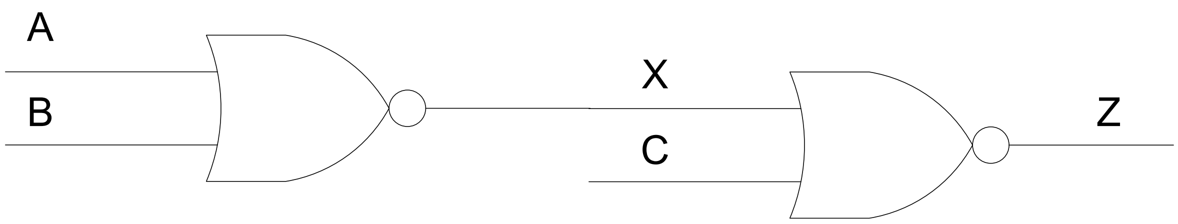
$$\boldsymbol{p_Z = 1- p_A \,.\, p(X|A) = 1}$$

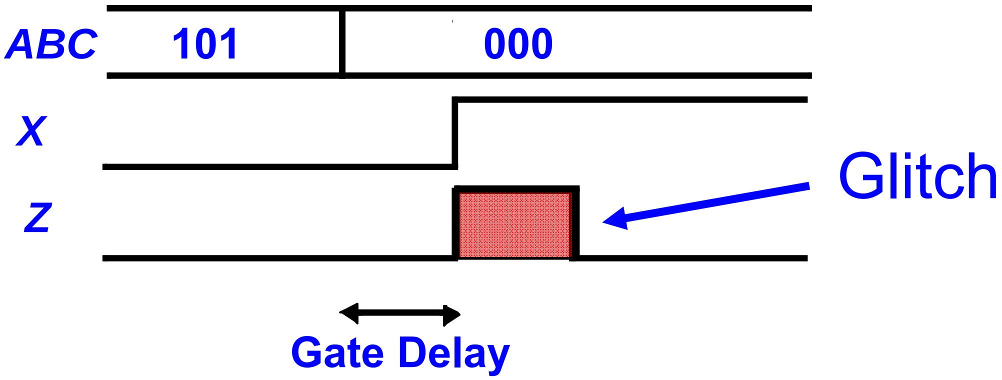
probability that X=1 given that A=1

Becomes complex and intractable real fast

Glitching in Static CMOS

Analysis so far did not include timing effects



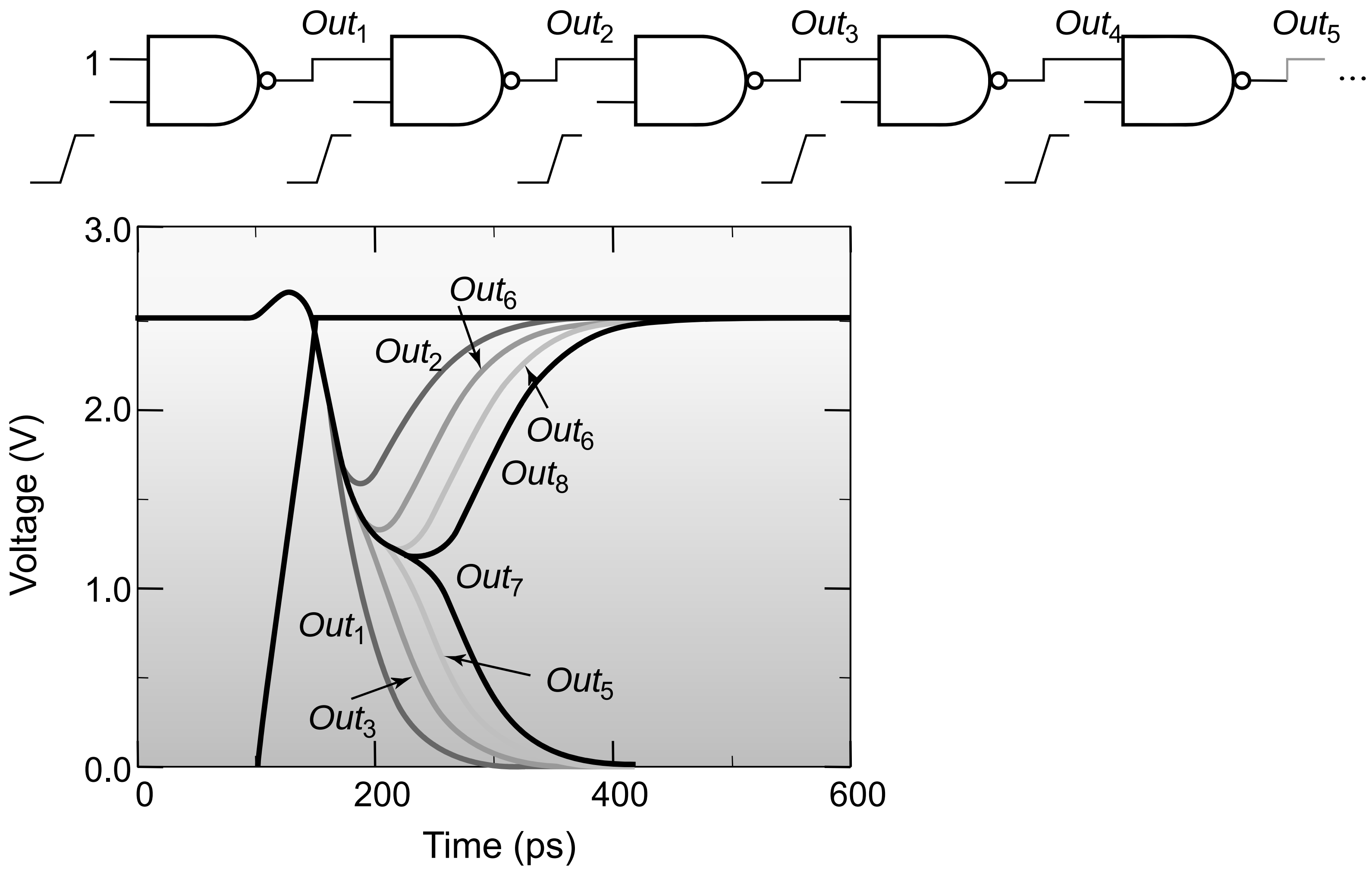


The result is correct,
but extra power is dissipated

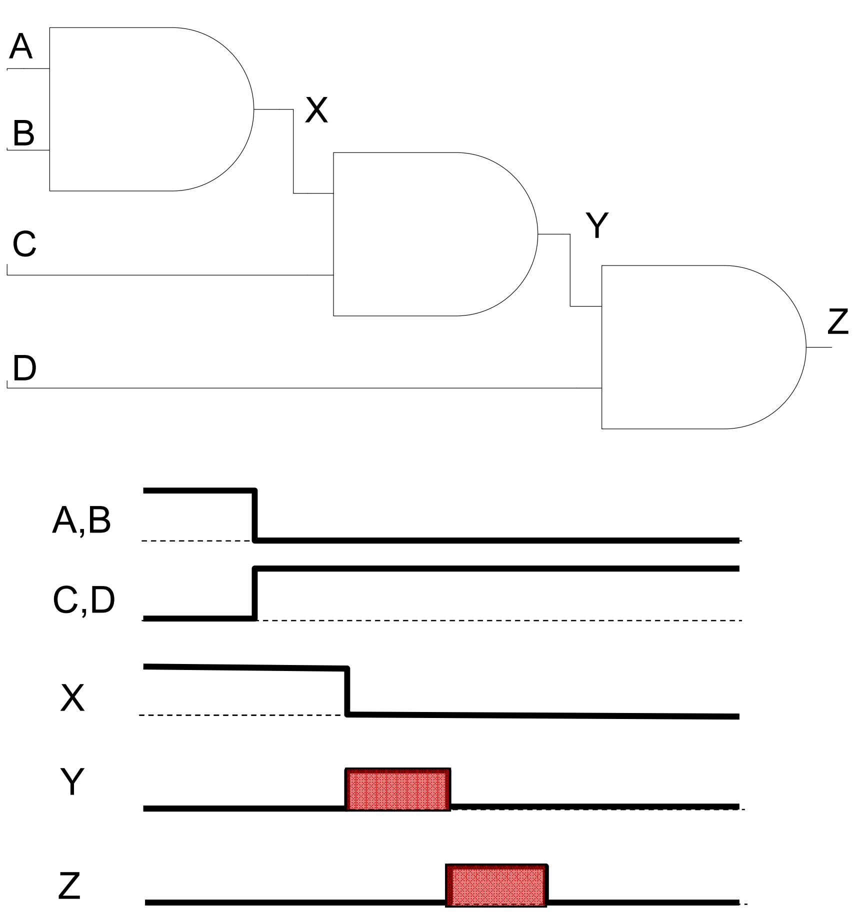
Also known as dynamic hazards:

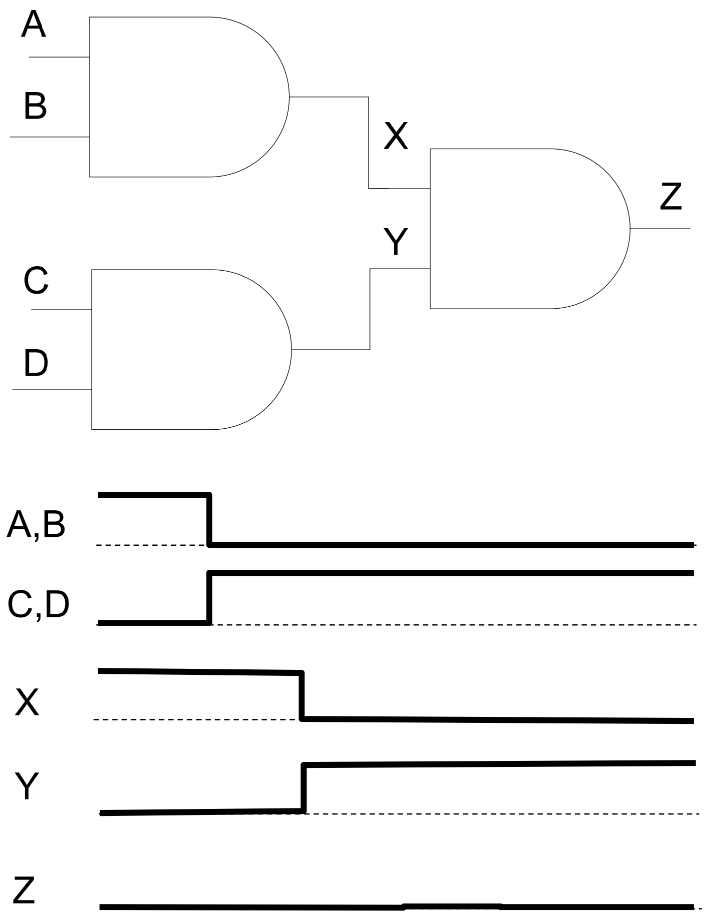
“A single input change causing multiple changes in the output”

Example: Chain of NAND Gates



What Causes Glitches?





Uneven arrival times of input signals of gate due to unbalanced delay paths

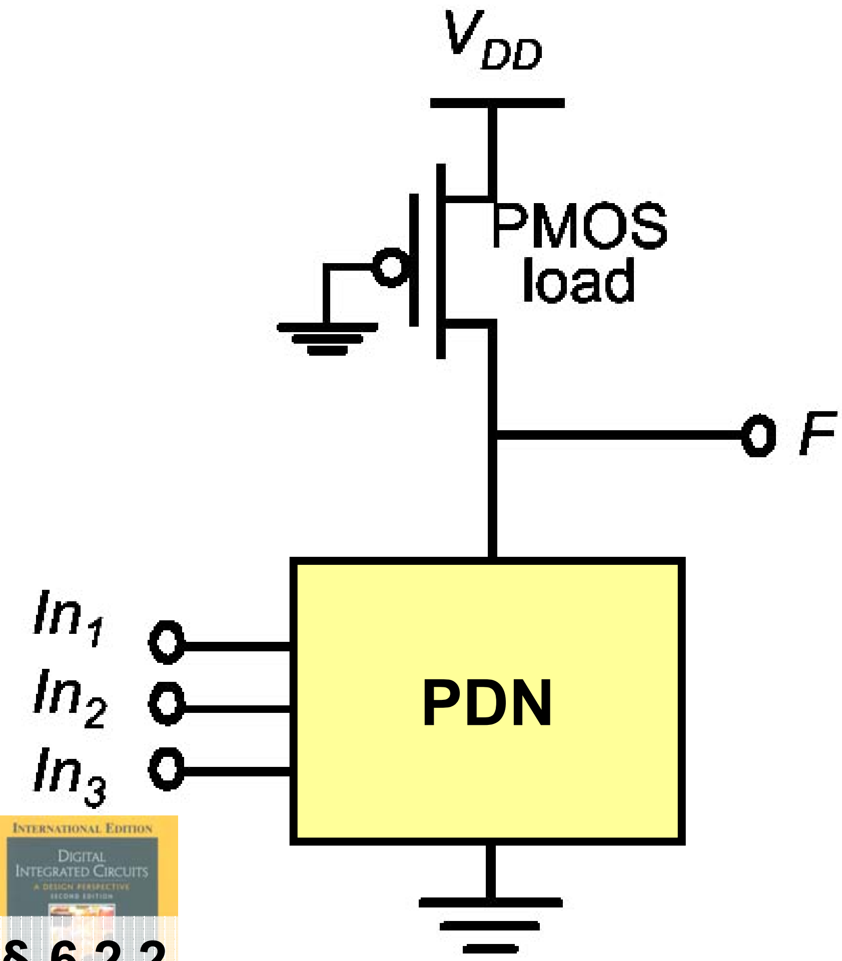
Solution: balancing delay paths!

Complimentary CMOS Gates - Summary

- Full rail-to-rail swing; high noise margins
- Logic levels not dependent upon the relative device sizes; ratioless
- Always a path to Vdd or Gnd in steady state; low output impedance
- Extremely high input resistance; nearly zero steady-state input current
- No direct path steady state between power and ground; no static power dissipation
- Need 2N transistors for N-input gate

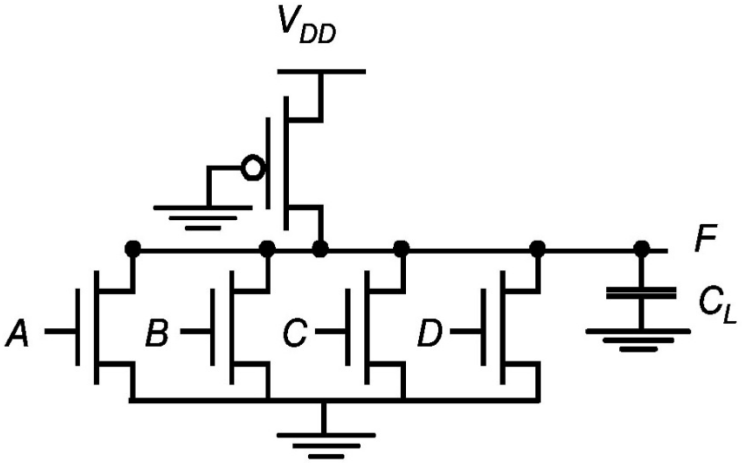
RATIOED LOGIC

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)

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

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

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

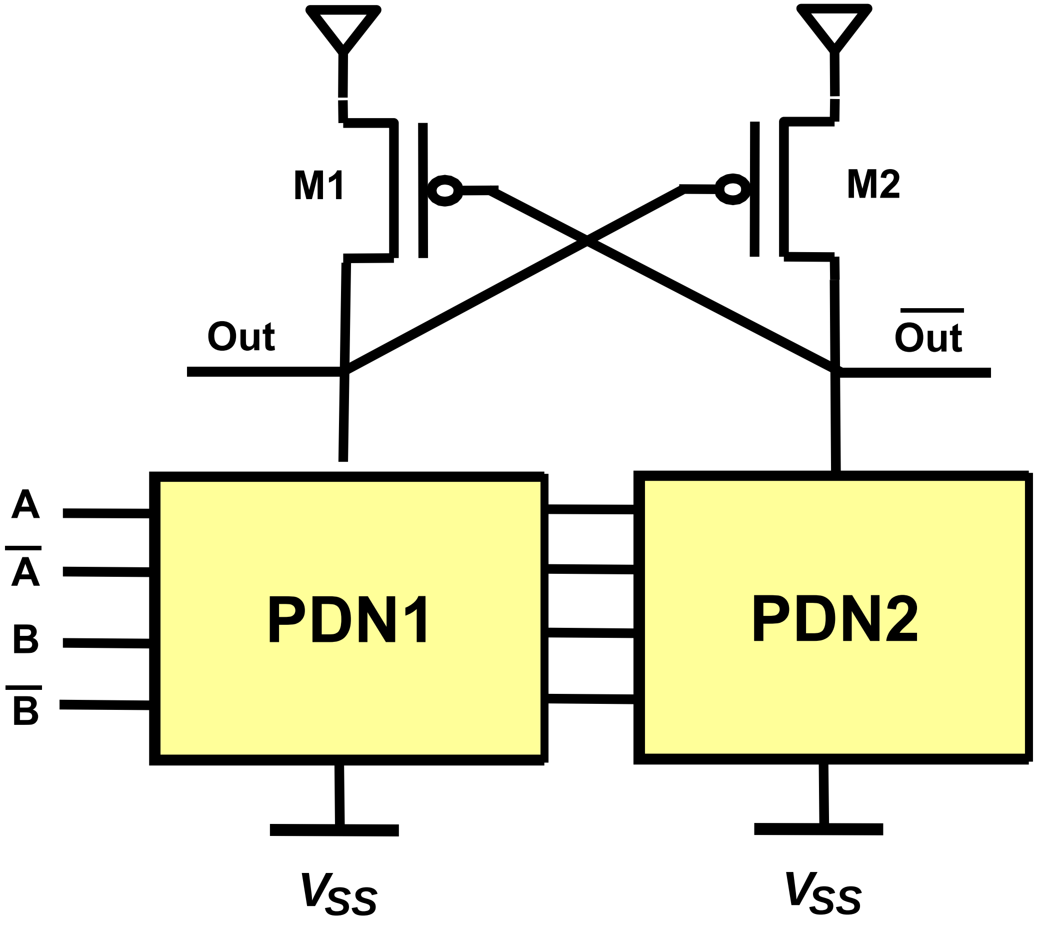
Pseudo NMOS Ratioed Logic

Performance of a pseudo-NMOS inverter

Size	V_{OL} [V]	Power [µW]	t_{pLH} [ps]
4	0.693	564	14
2	0.273	298	56
1	0.133	160	123
0.5	0.064	80	268
0.25	0.031	41	569

Pseudo NMOS Ratioed Logic

Differential Cascode Voltage Switch Logic (DCVSL)



☺ Rail-to-rail swing

☺ No static dissipation

☹ Rationed

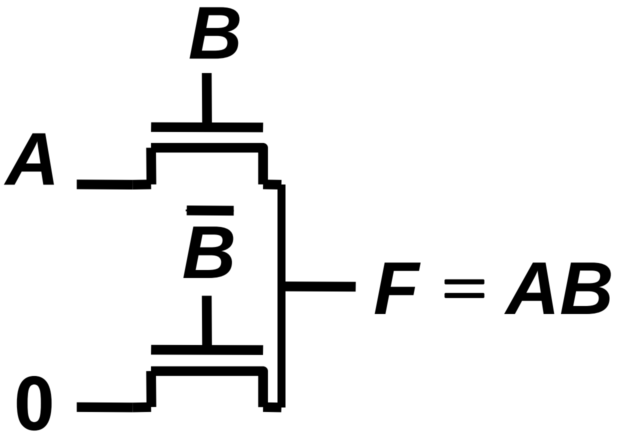
☹ Cross-over currents

☹ Wiring

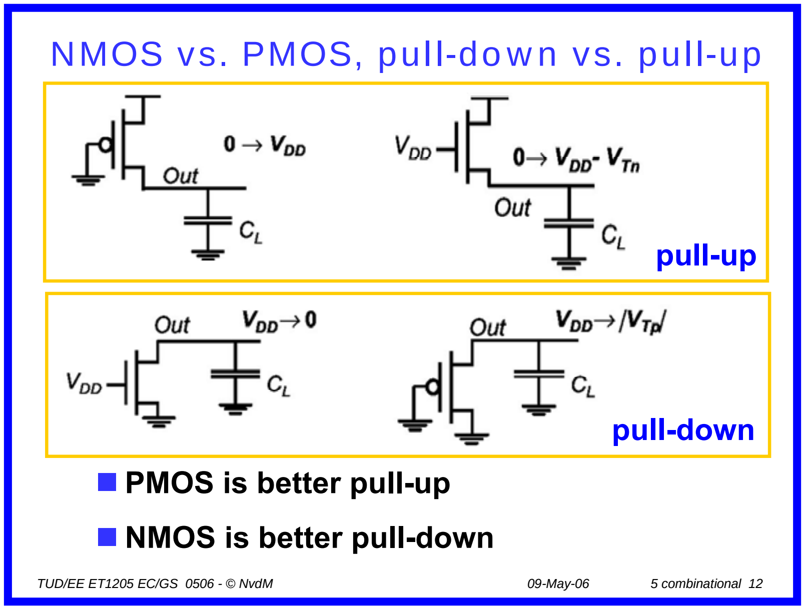
PASS TRANSISTOR LOGIC PASS GATE LOGIC

Pass Transistor Logic

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



But remember:

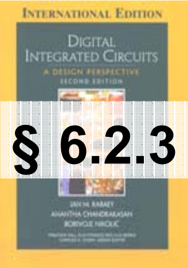


NMOS vs. PMOS, pull-down vs. pull-up

$0 \rightarrow V_{DD}$
Out
C_L
V_{DD}
$0 \rightarrow V_{DD} - V_{Tn}$
Out
C_L
pull-up
Out
$V_{DD} \rightarrow 0$
V_{DD}
C_L
Out
$V_{DD} \rightarrow |V_{Tp}|$
C_L
pull-down

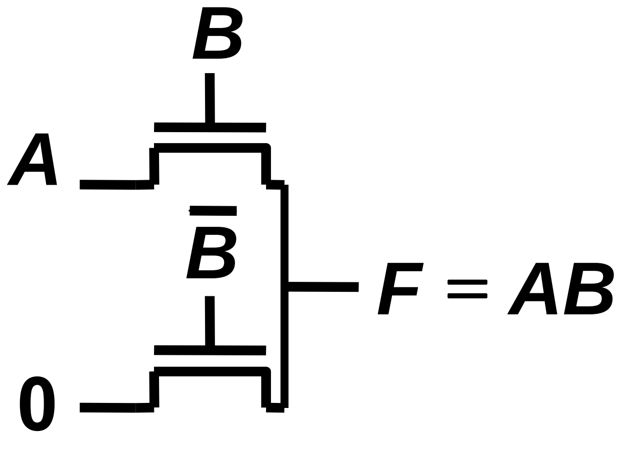
- PMOS is better pull-up
- NMOS is better pull-down

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



§ 6.2.3

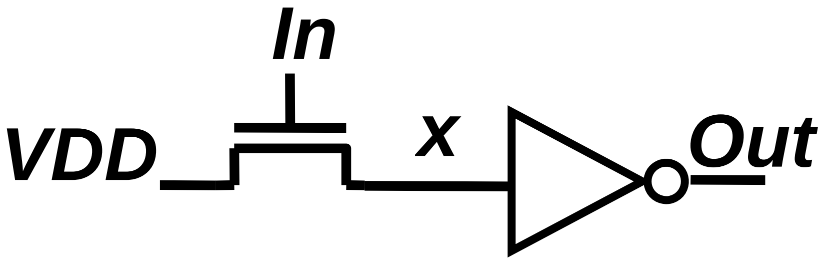
Pass Transistor Logic

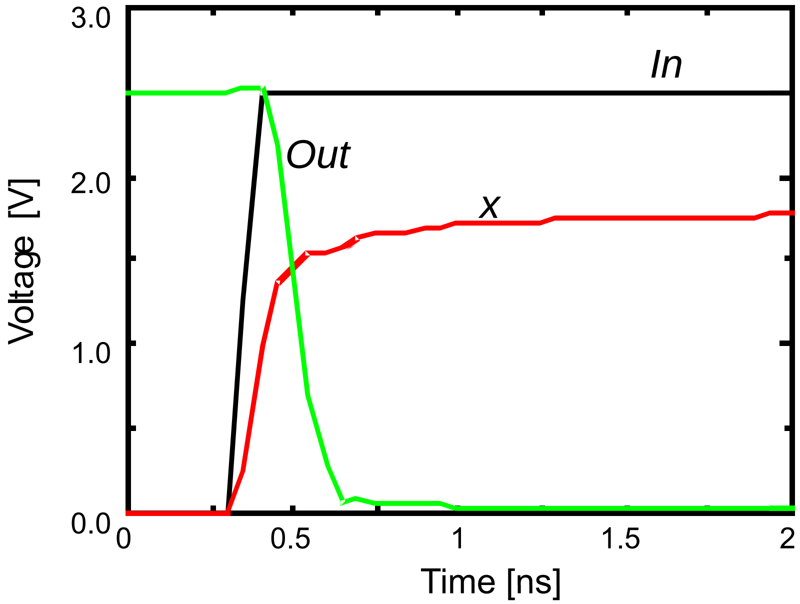


- Save **area, capacitances**
- Need complementary inputs (might mean extra inverters)
- Reduced V_{OH}, **noise margins**
- $V_{OH} = V_{DD} - \left(V_{Tno} + \gamma\left(\left(\sqrt{|2\phi_f| + V_{OH}}\right) - \sqrt{|2\phi_f|}\right)\right)$
- **Static dissipation** in subsequent static inverter/buffer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

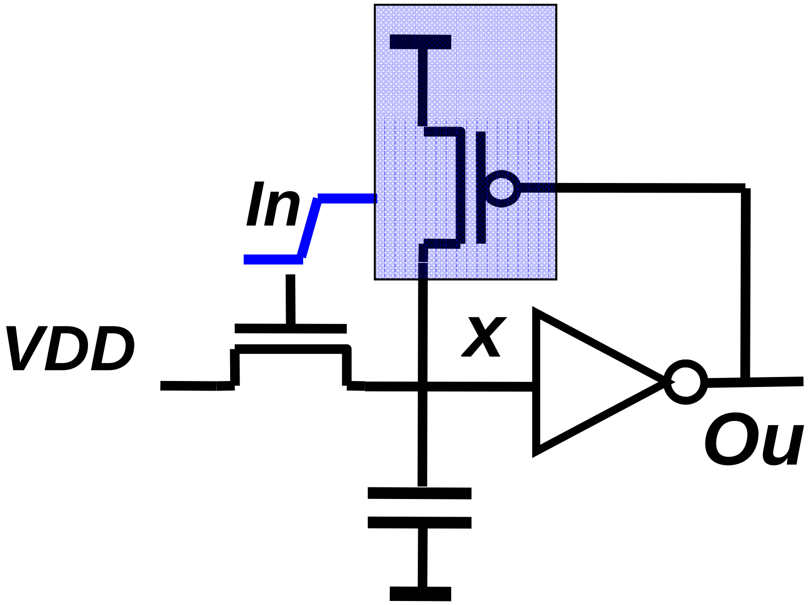
{TPS}: Why is there static dissipation in next conventional gate?

Pass Transistor Logic





- Level restoring circuit

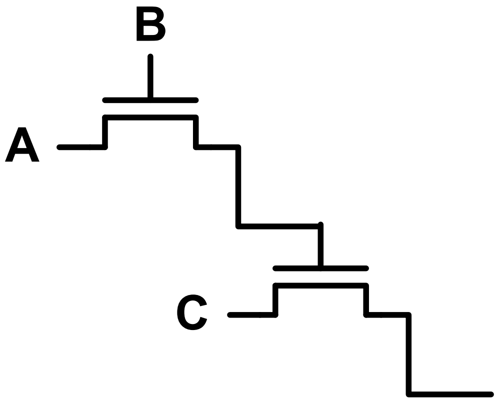


Level restoring circuit

- ☺ **Rail-to-rail swing**
- ☺ **No static dissipation**
- ☹ **Rationed – use with care**
- ☹ **Increased capacitance**

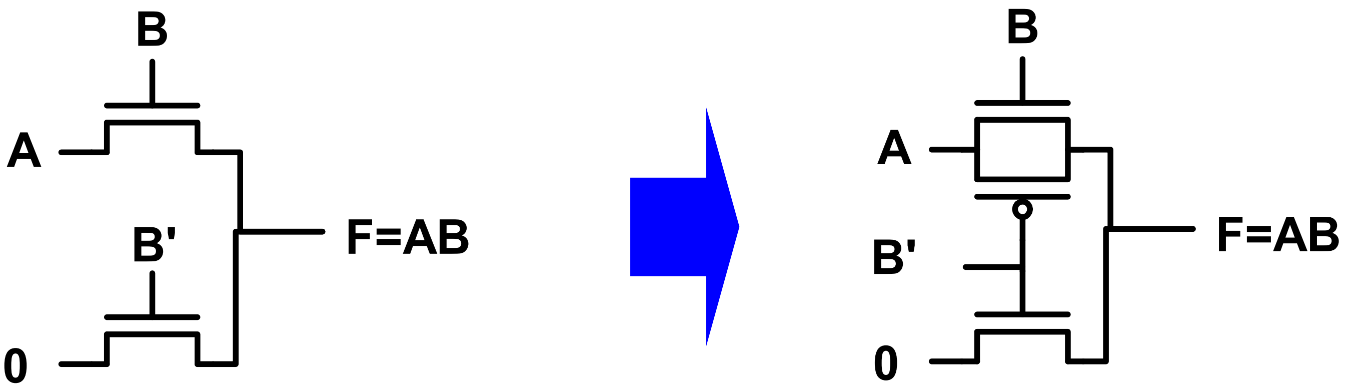
Exercise

- **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?**



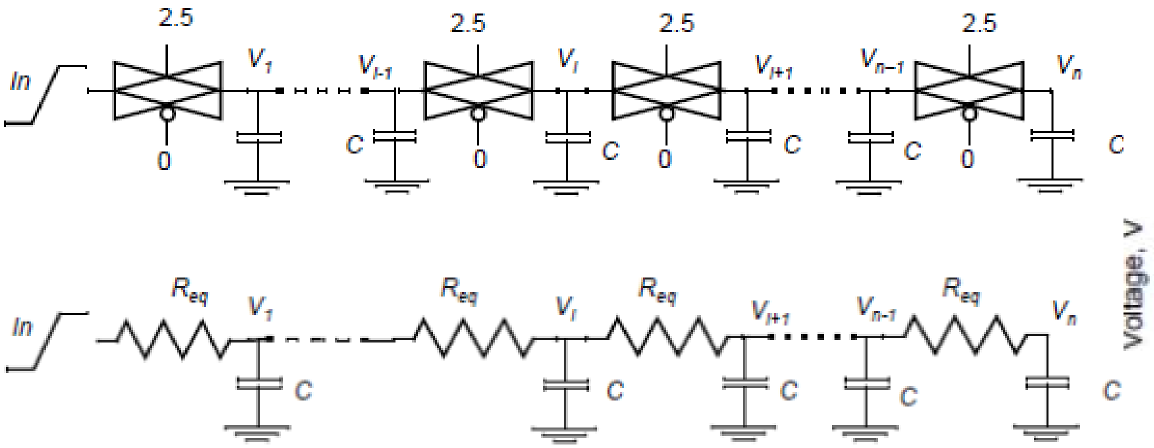
Pass Gates (Transmission Gates)

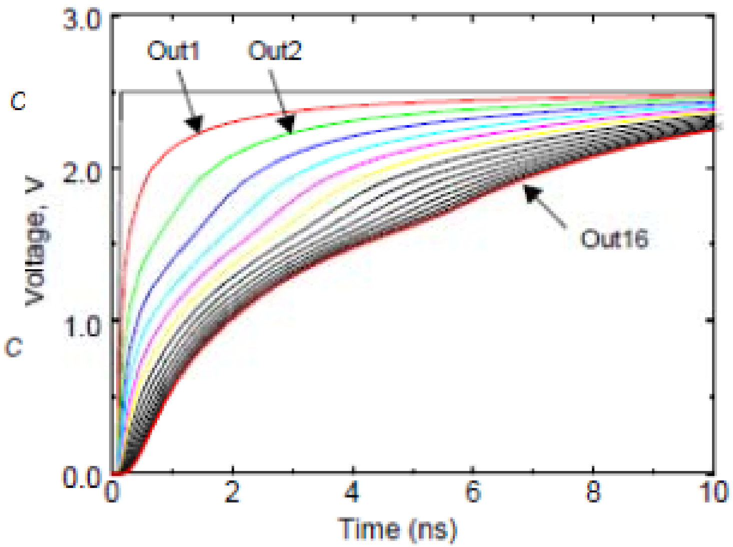
- use an N-MOS and a P-MOS in parallel



- Pass gates eliminate some of the disadvantages of simple pass-transistors
  - Eliminates reduced noise margins & static power consumption
- Disadvantages of pass gate:
  - Requires both NMOS and PMOS in different wells, both true and complemented polarities of the control signal needed, increases node capacitance
- Design remains a trade-off!

Pass Transistor Logic





$$t_p(V_n) = 0.69\sum_{k=0}^{n} kCR_{eq} = 0.69CR_{eq}\frac{n(n+1)}{2}$$

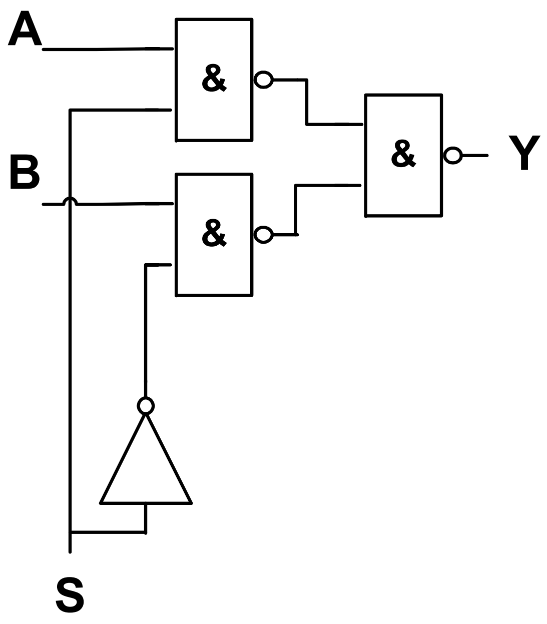
- **Propagation delay is proportional to n²!**
- **Insert buffers**

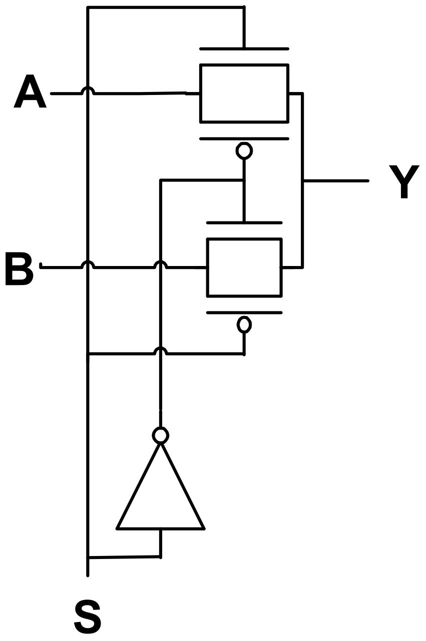
$$m_{opt} = 1.7\sqrt{\frac{t_{pbuf}}{CR_{eq}}}$$

- **In current technologies, m_{opt} is typically 3 or 4**

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





Dynamic CMOS gates

Static vs. Dynamic CMOS Circuits

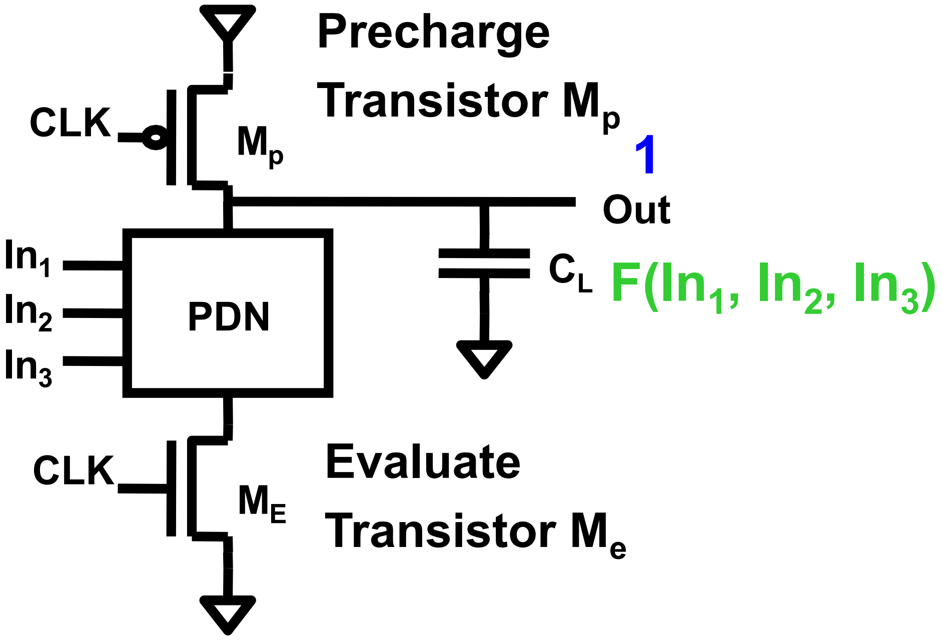
Static

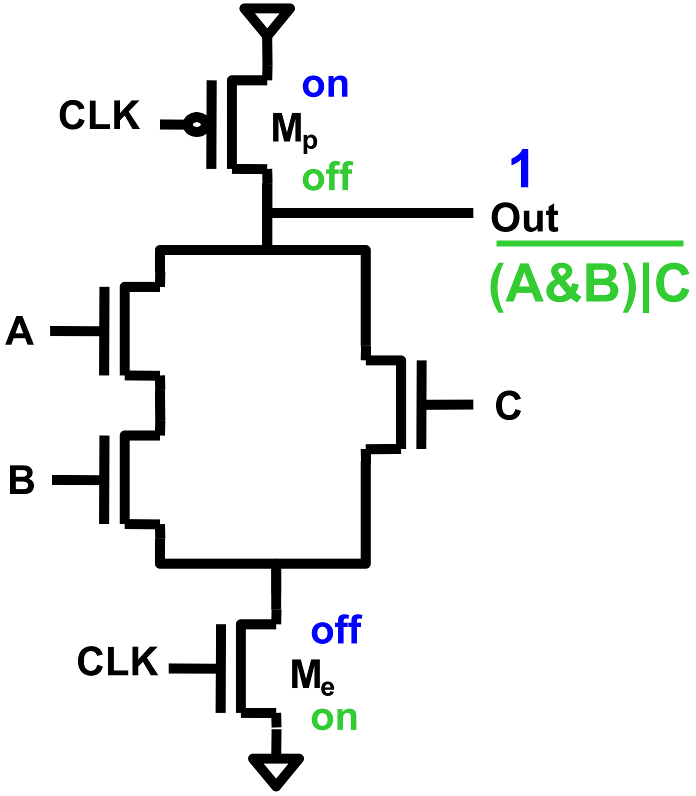
- At every point in time (except during the switching transients) each **gate output is connected to either V_{DD} or V_{SS}** via a low-resistive path.
- The outputs of the gates assume at all times the **value of the Boolean function**, implemented by the circuit (except during switching periods)
- Require 2N transistors for N inputs (fan-in of N)

Dynamic

- **Output** not permanently connected to Vdd or Vss
- Output value partly relies on storage of signal values on the **capacitance** of high impedance circuit nodes.
- Input only active when **clock** is active
- Requires N+2 transistors for N inputs

Dynamic Gate





Two phase operation

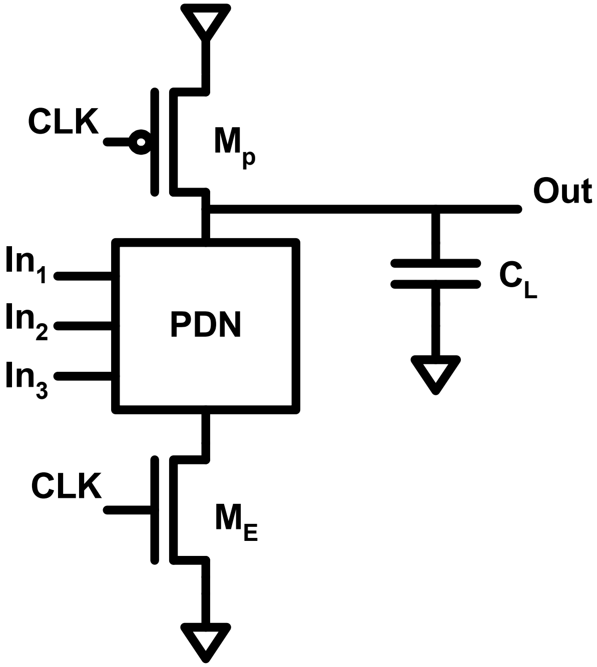
§ 6.3

Precharge (CLK = 0)

Evaluate (CLK = 1)

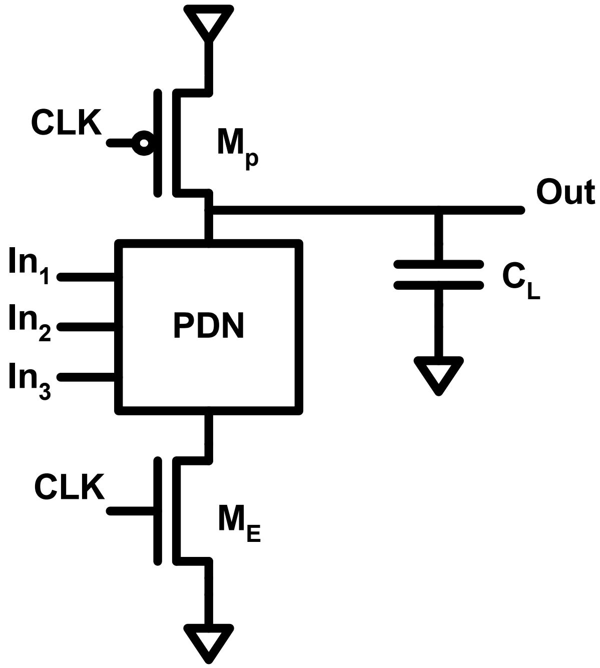
- Only look at output after evaluation phase
- On rhythm of global clock
- Example: use edge-triggered FF with trigger on 1→0 transition of clock to sample logical value

Conditions on Output



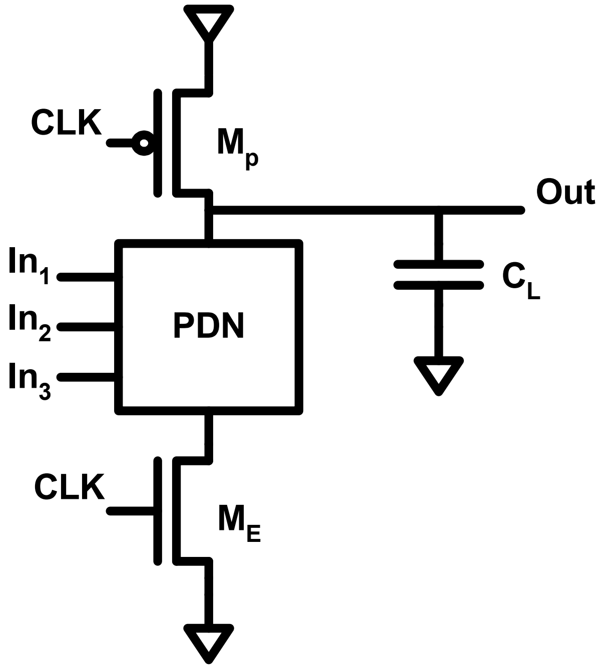
- Only **one output transition per clock cycle**, after CLK 0→1. It cannot be charged again until the next precharge operation
- **Inputs** to the gate can make **at most one transition** during evaluation
- Output can be in the **high impedance** state during and after evaluation (PDN off), state is stored on C_L

Properties of Dynamic Gates (1)



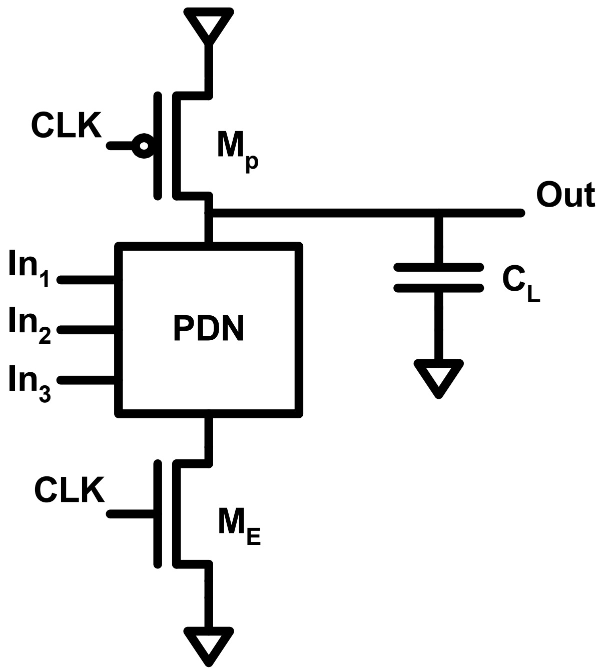
- **Logic function is implemented by the PDN only**
- **Number of transistors is N + 2 (versus 2N for static complementary CMOS)**
- **Full swing outputs ($V_{OL} = V_{SS}$ and $V_{OH} = V_{DD}$)**
- **Nonratioed – sizing of the devices is not important for proper functioning**

Properties of Dynamic Gates (2)



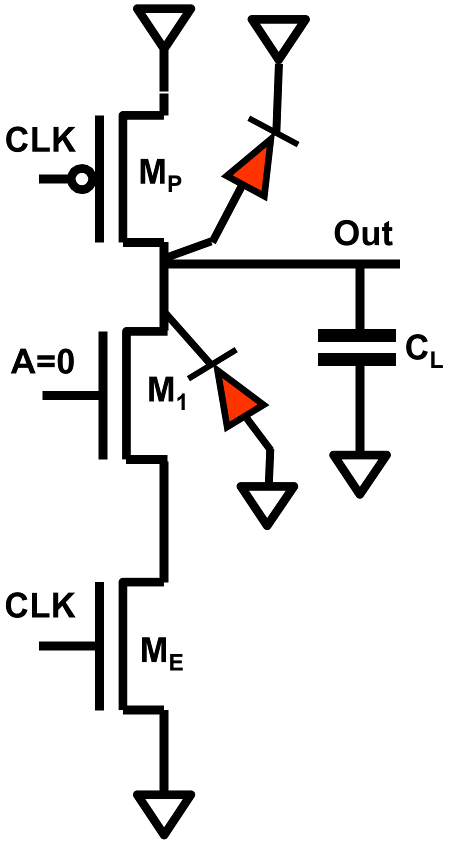
- **Faster switching speeds**
  - **reduced capacitive load to predecessor (only PDN)**
  - **reduced internal capacitance (drain cap. of only one pull-up)**
- **Low noise margin (NM_L)**
  - **PDN starts to work as soon as the input signals exceed v_{TN} ⇨ V_M and V_{IL} equal to V_{TN}**

Properties of Dynamic Gates (3)

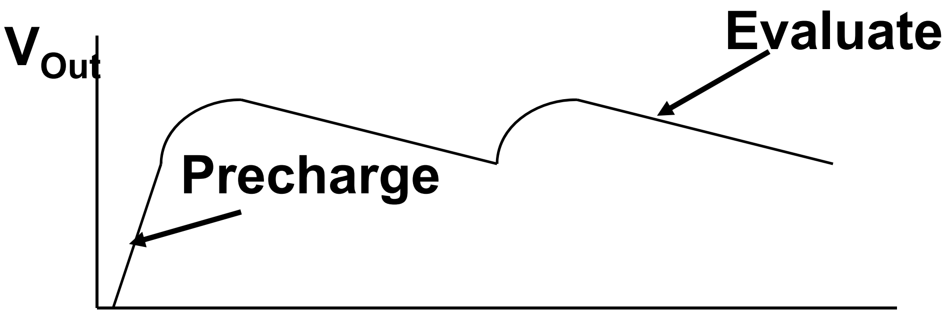


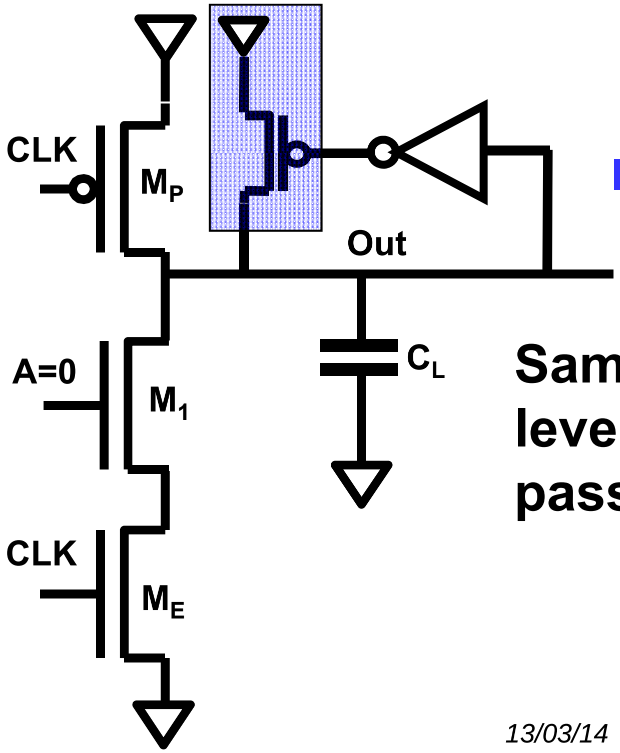
- Needs a **precharge clock**
- {TPS} compare power of dynamic vs static CMOS: higher or lower
- Overall power **dissipation** usually **significantly higher** than static CMOS
  - ☺ Reduced capacitance
  - ☺ no static current path ever exists between V_{DD} and GND (including P_{sc})
  - ☺ no glitching
  - ☹ higher transition probabilities
  - ☹ extra load on CLK

Issues in Dynamic Design (1)



- **Charge leakage – via reversed-biased diffusion diodes and subthreshold leakage**

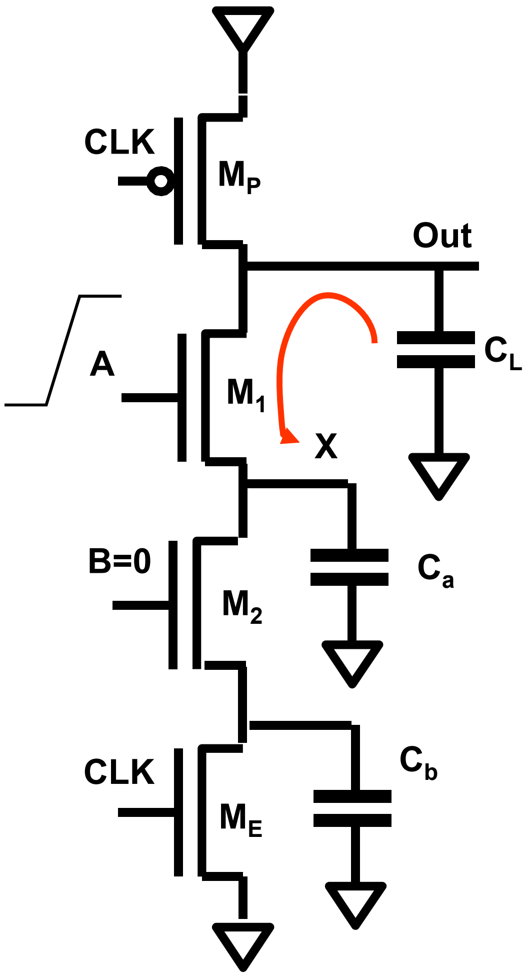




- **Static bleeders**

Same approach as level restorer for pass-transistor logic

Issues in Dynamic Design (2)



- **Charge redistribution**

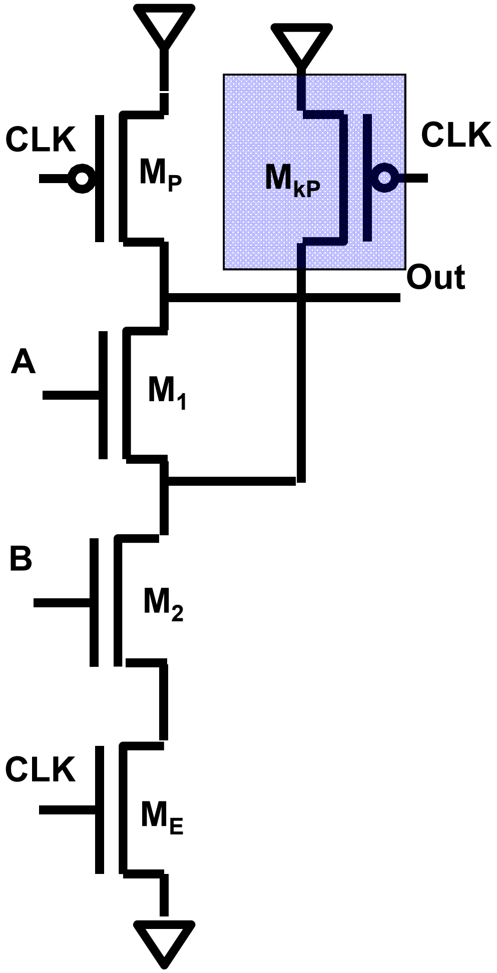
Charge stored originally on C_L is redistributed (shared) over C_L and C_A leading to reduced robustness

If $\Delta V_{out} > V_{Tn}$ then V_{out} and V_x reach the same value

$$\Delta V_{out} = -V_{DD}\frac{C_a}{C_a + C_L}$$

Target is to keep $\Delta V_{out} < |V_{Tp}|$ since output may drive a static gate

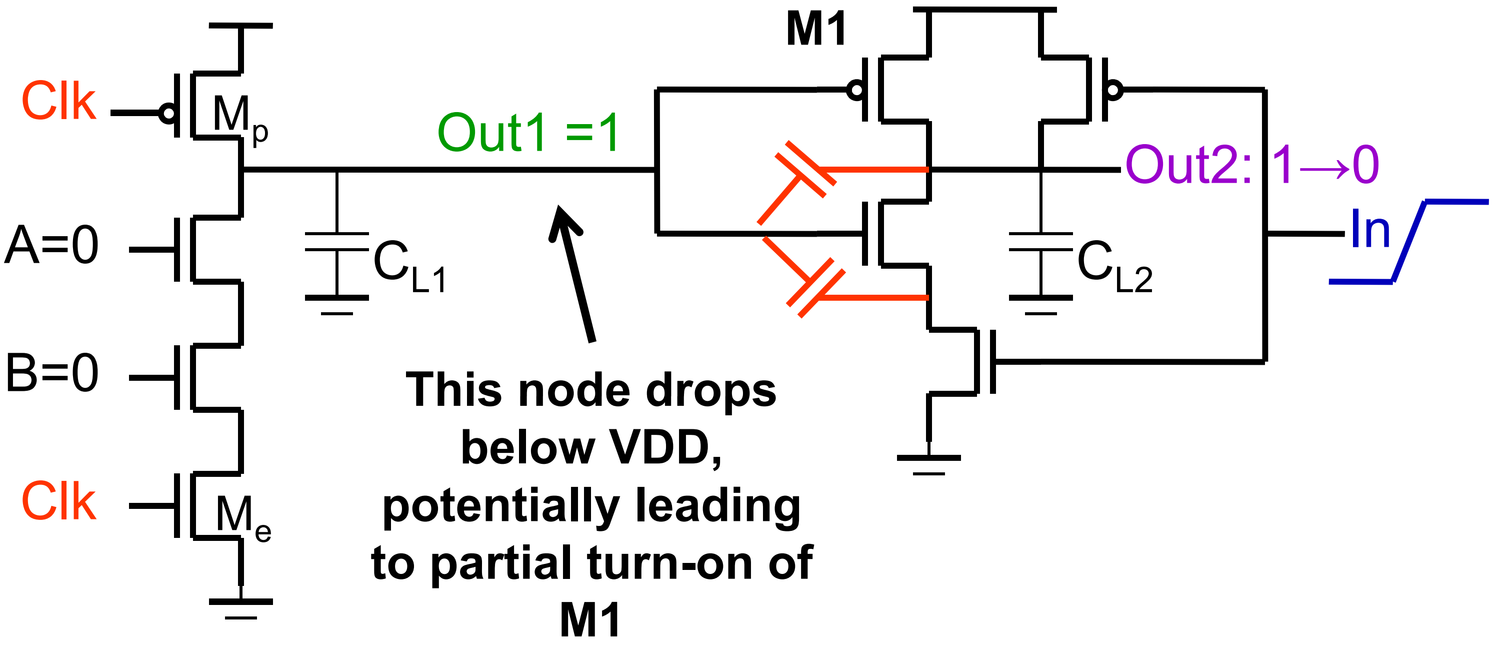
Issues in Dynamic Design (2)



- **Solution to charge redistribution**
  - **Pre-charge internal nodes using a clock-driven PMOS transistor (at the cost of increased area and power)**

Issues in Dynamic Design (3)

- **Backgate Coupling**



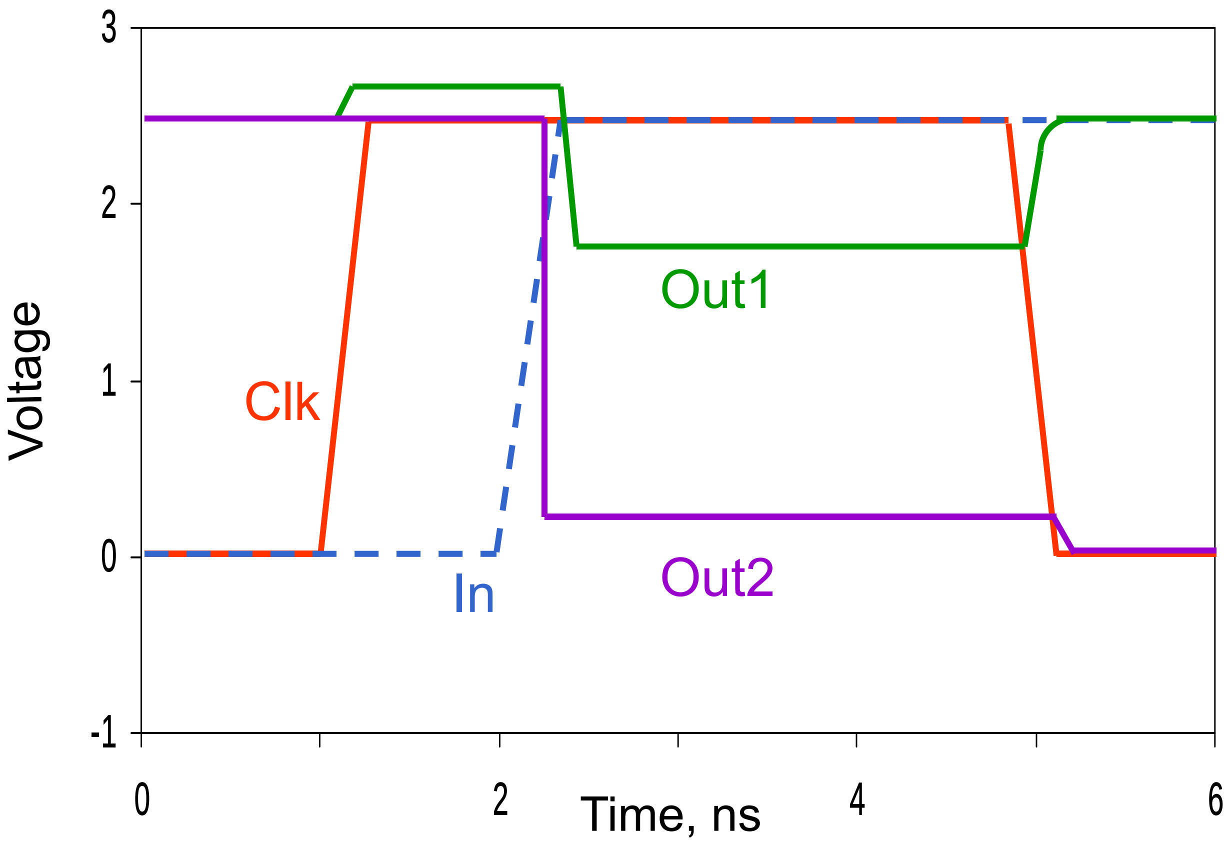
This node drops below VDD, potentially leading to partial turn-on of M1

Dynamic NAND

Static NAND

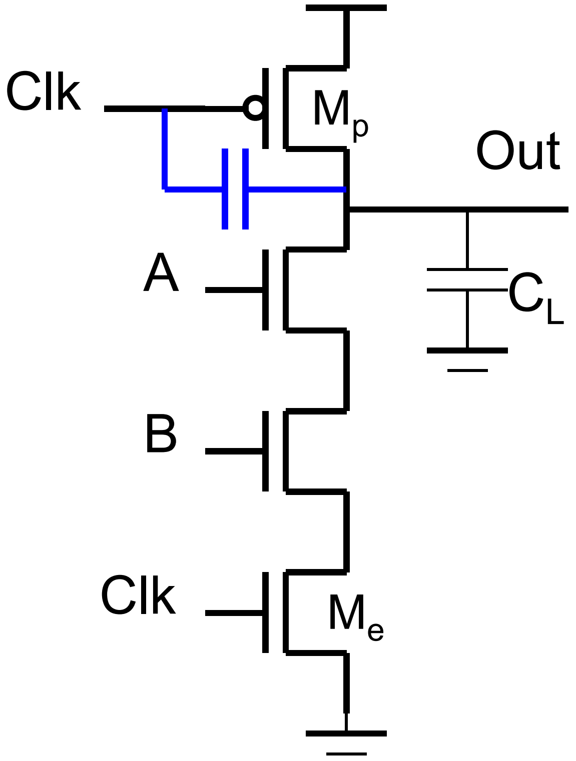
Issues in Dynamic Design (3)

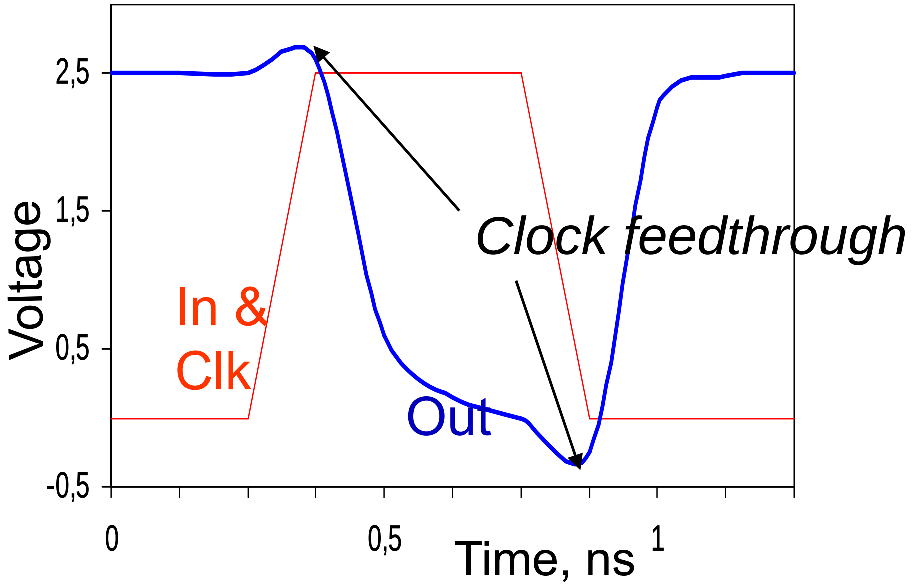
- **Backgate Coupling**



Issues in Dynamic Design (4)

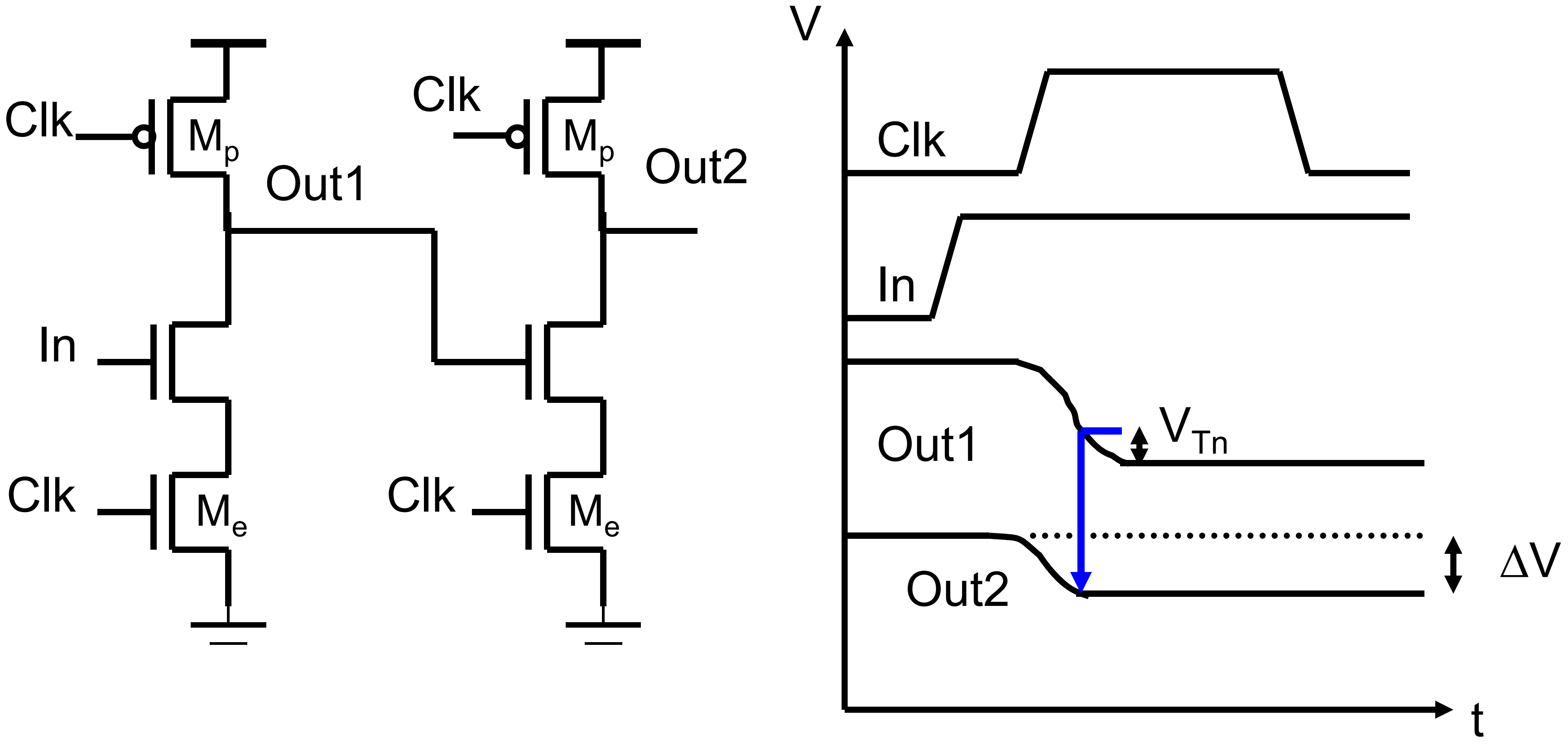
- **Clock Feedthrough**
  - **Coupling between V_{Out} and Clk_{in} of the pre-charge device due to the C_{GD}**
  - **May forward bias the junction and inject electrons into substrate**
  - **V_{Out} can rise above V_{DD}**
  - **The fast rising (and falling edges) of the clock couple to Out**





Issues in Dynamic Design (5)

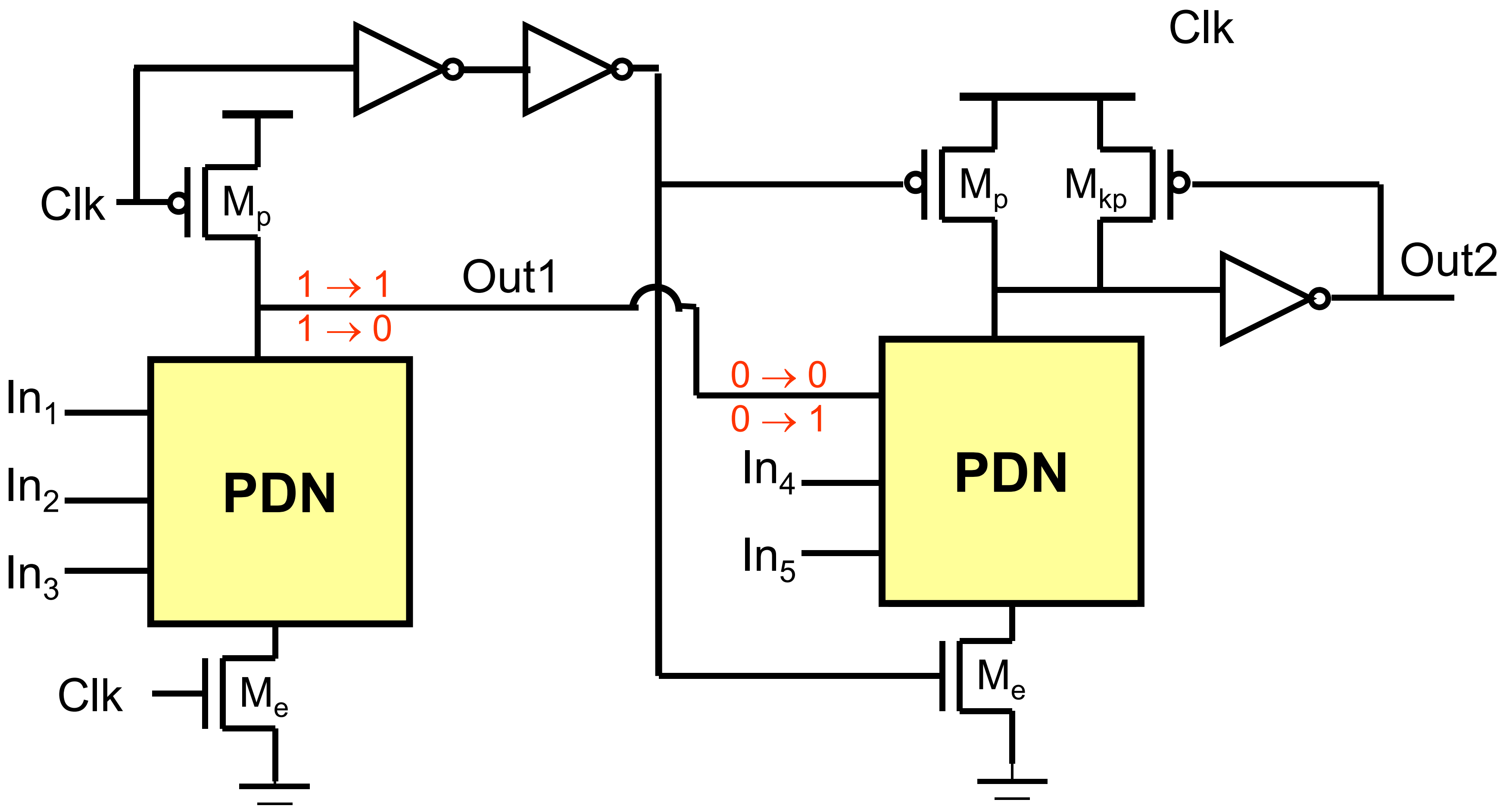
- **Cascading Dynamic Gates**



Only 0 → 1 transitions allowed at inputs!

Problem when input of 2nd gate not being 0 during precharge

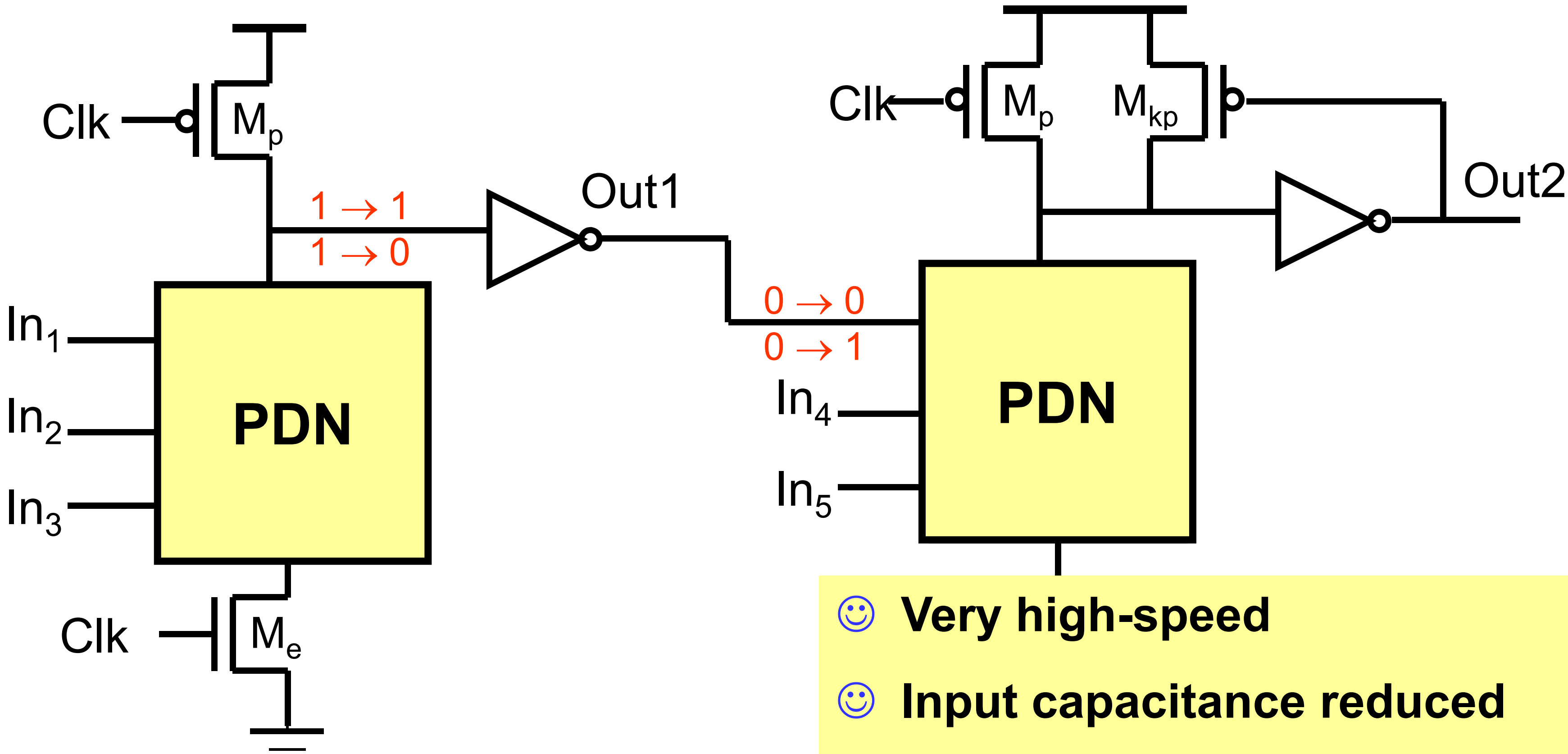
Delayed Clocks



Evaluation starts when Out1 is stable

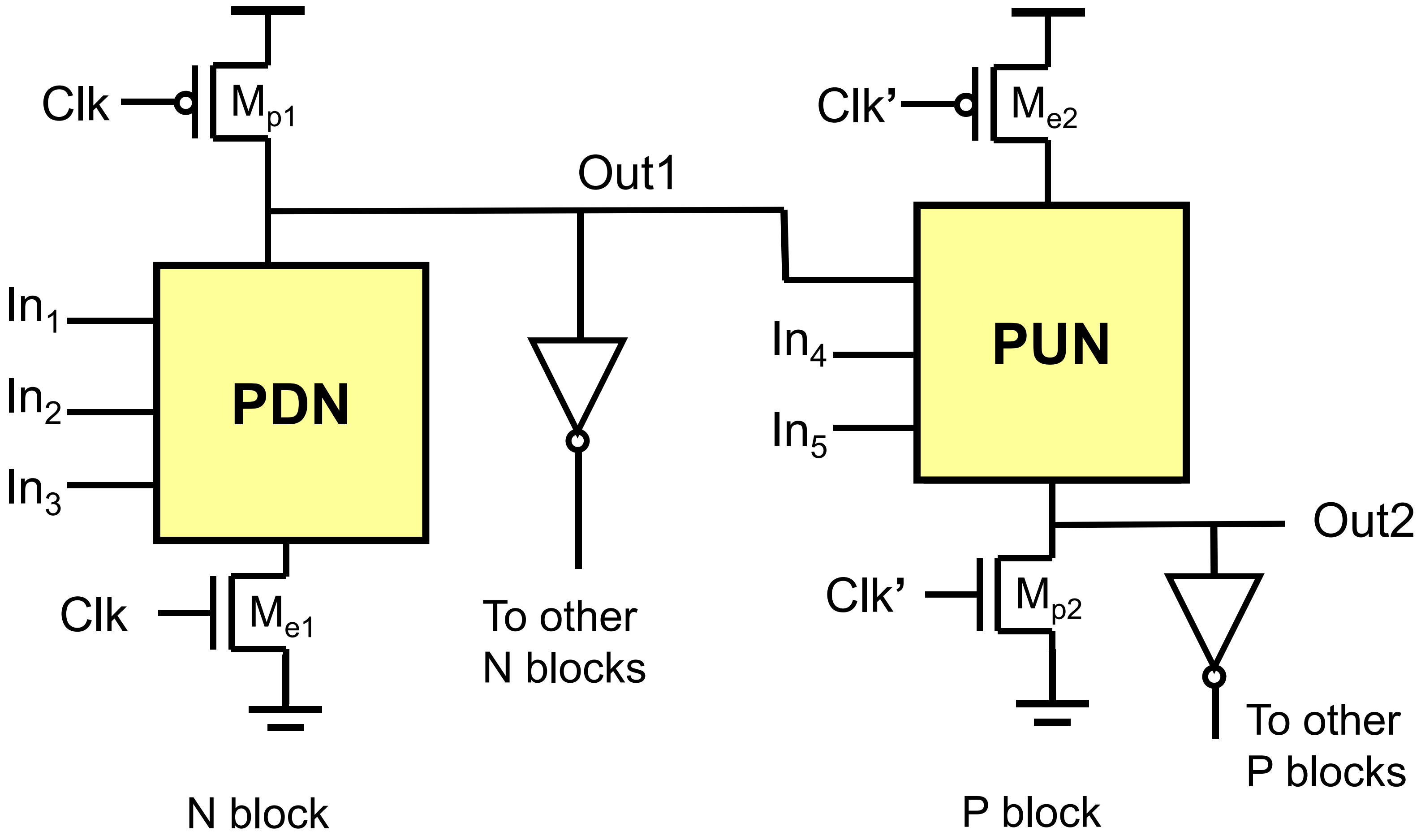
Domino Logic

Ensures all inputs to the Domino gate is set to 0 during precharge period

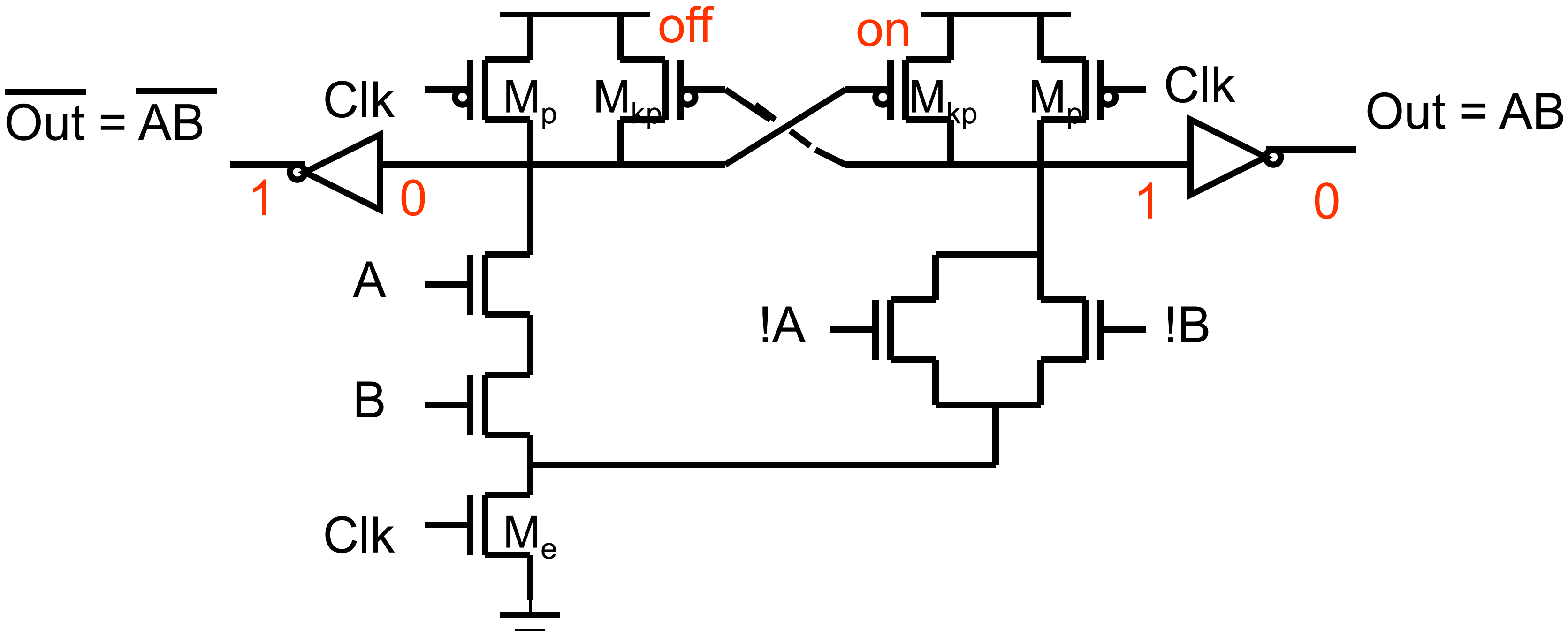


- ☺ **Very high-speed**
- ☺ **Input capacitance reduced**
- ☺ **No static dissipation**
- ☹ **High dynamic dissipation**
- ☹ **Only non-inverting logic**

NP Logic, aka NORA Logic



Differential (Dual Rail) Domino Logic



Solves the problem of non-inverting logic

Summary

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