



# *Digital Integrated Circuits*

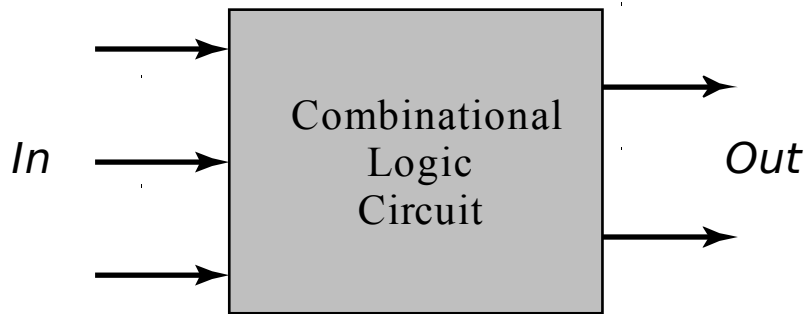
## *A Design Perspective*

Jan M. Rabaey  
Anantha Chandrakasan  
Borivoje Nikolić

## Designing Combinational Logic Circuits

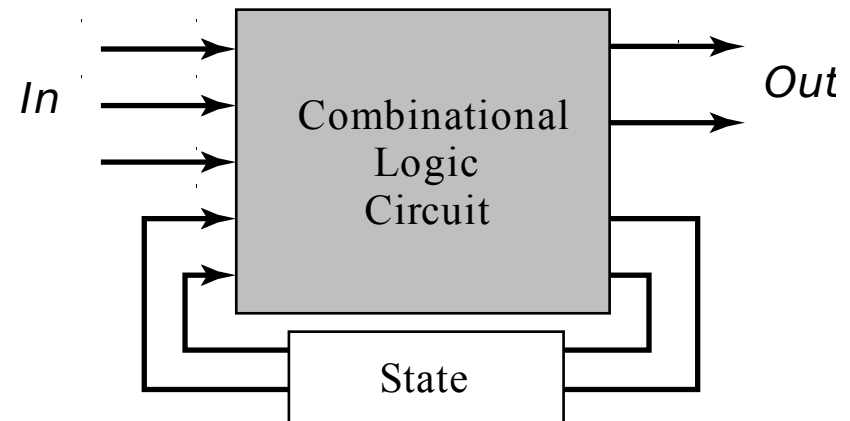
*November 2002.*

# Combinational vs. Sequential Logic



Combinational

$$\text{Output} = f(\text{In})$$



Sequential

$$\text{Output} = f(\text{In}, \text{Previous In})$$

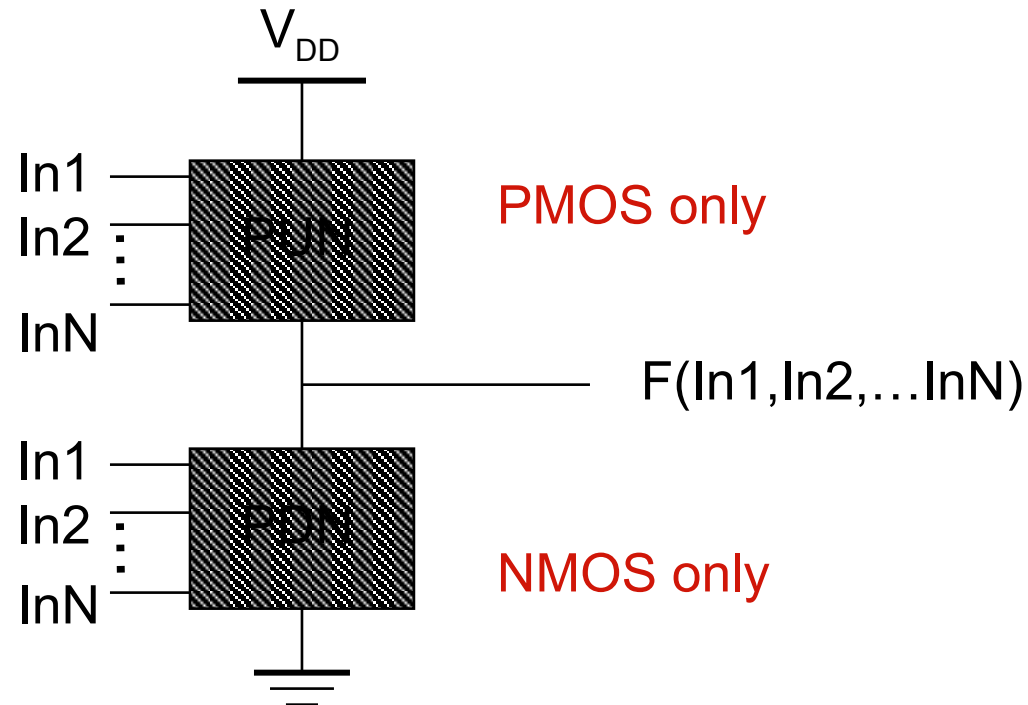
# Static CMOS Circuit

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 (ignoring, once again, the transient effects during switching periods).

This is in contrast to the *dynamic* circuit class, which relies on temporary storage of signal values on the capacitance of high impedance circuit nodes.

# Static Complementary CMOS

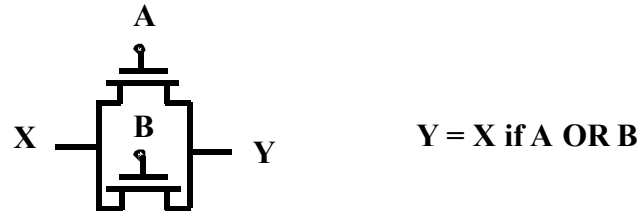


PUN and PDN are **dual** logic networks

# NMOS Transistors in Series/Parallel Connection

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

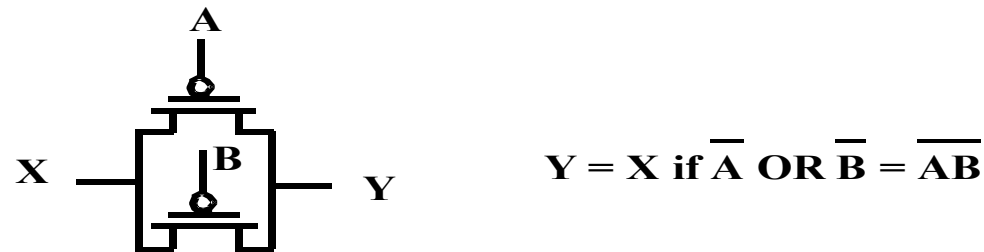
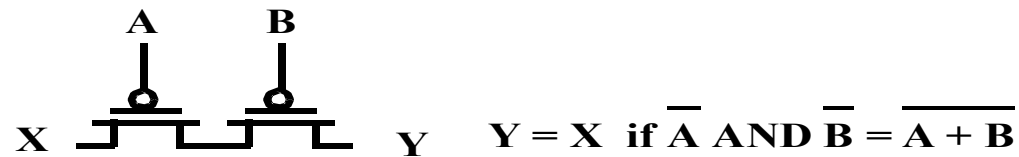
**NMOS switch closes when switch control input is high**



**NMOS Transistors pass a “strong” 0 but a “weak” 1**

# PMOS Transistors in Series/Parallel Connection

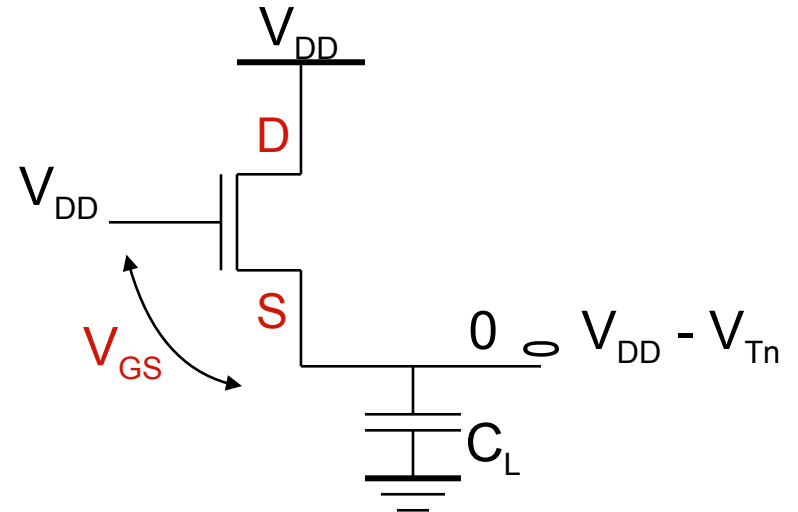
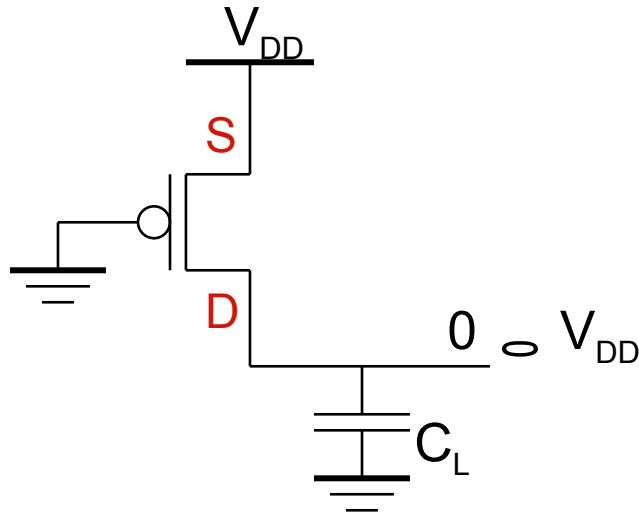
**PMOS switch closes when switch control input is low**



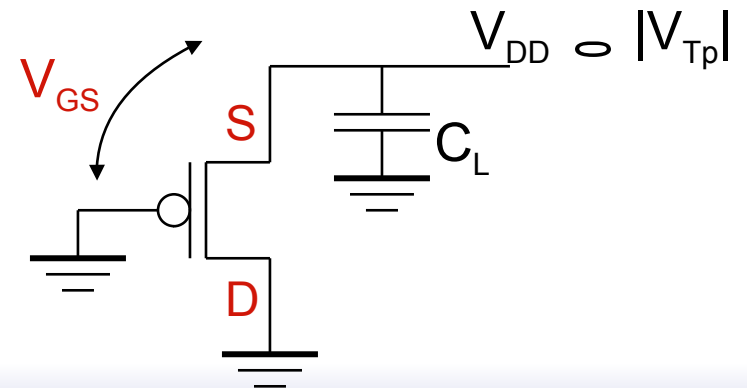
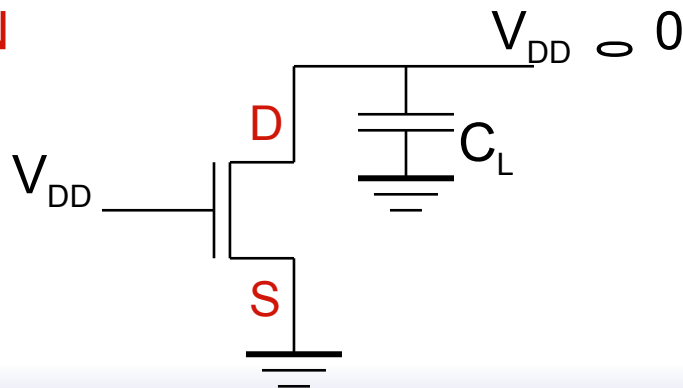
**PMOS Transistors pass a “strong” 1 but a “weak” 0**

# Threshold Drops

PUN



PDN



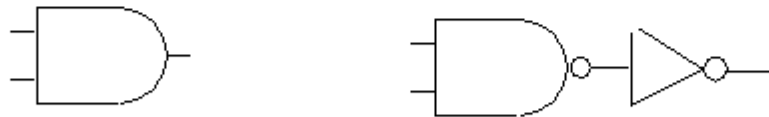
# Complementary CMOS Logic Style

- PUP is the DUAL of PDN  
(can be shown using DeMorgan's Theorem's)

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

$$\overline{AB} = \bar{A} + \bar{B}$$

- The complementary gate is inverting



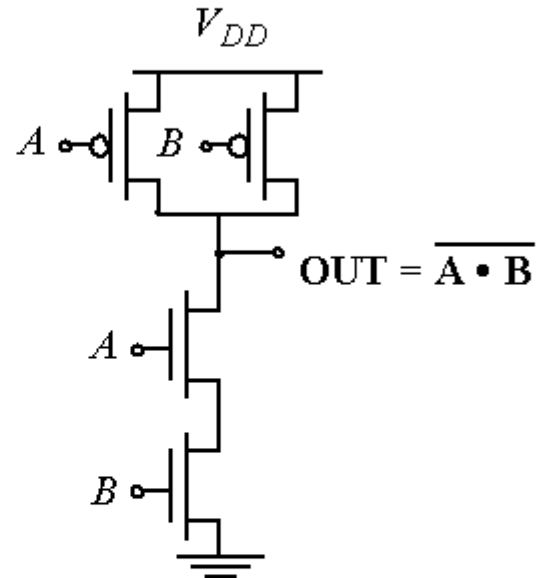
$$\text{AND} = \text{NAND} + \text{INV}$$



# Example Gate: NAND

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

Truth Table of a 2 input NAND gate



PDN:  $G = A B \Rightarrow$  Conduction to GND

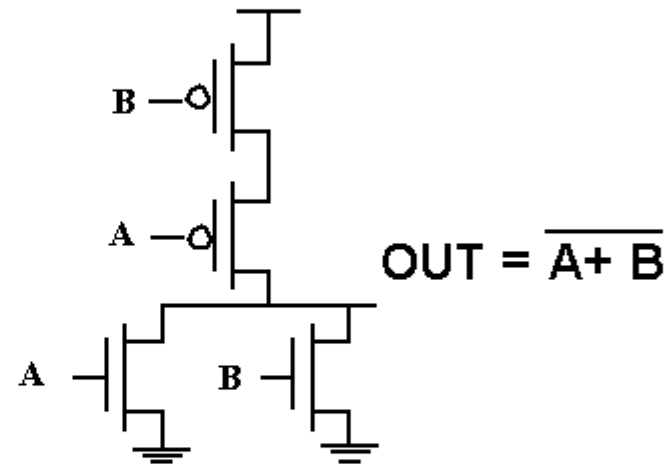
PUN:  $F = \overline{A} + \overline{B} = \overline{AB} \Rightarrow$  Conduction to  $V_{DD}$

$$\overline{G(In_1, In_2, In_3, \dots)} \equiv F(\overline{In_1}, \overline{In_2}, \overline{In_3}, \dots)$$

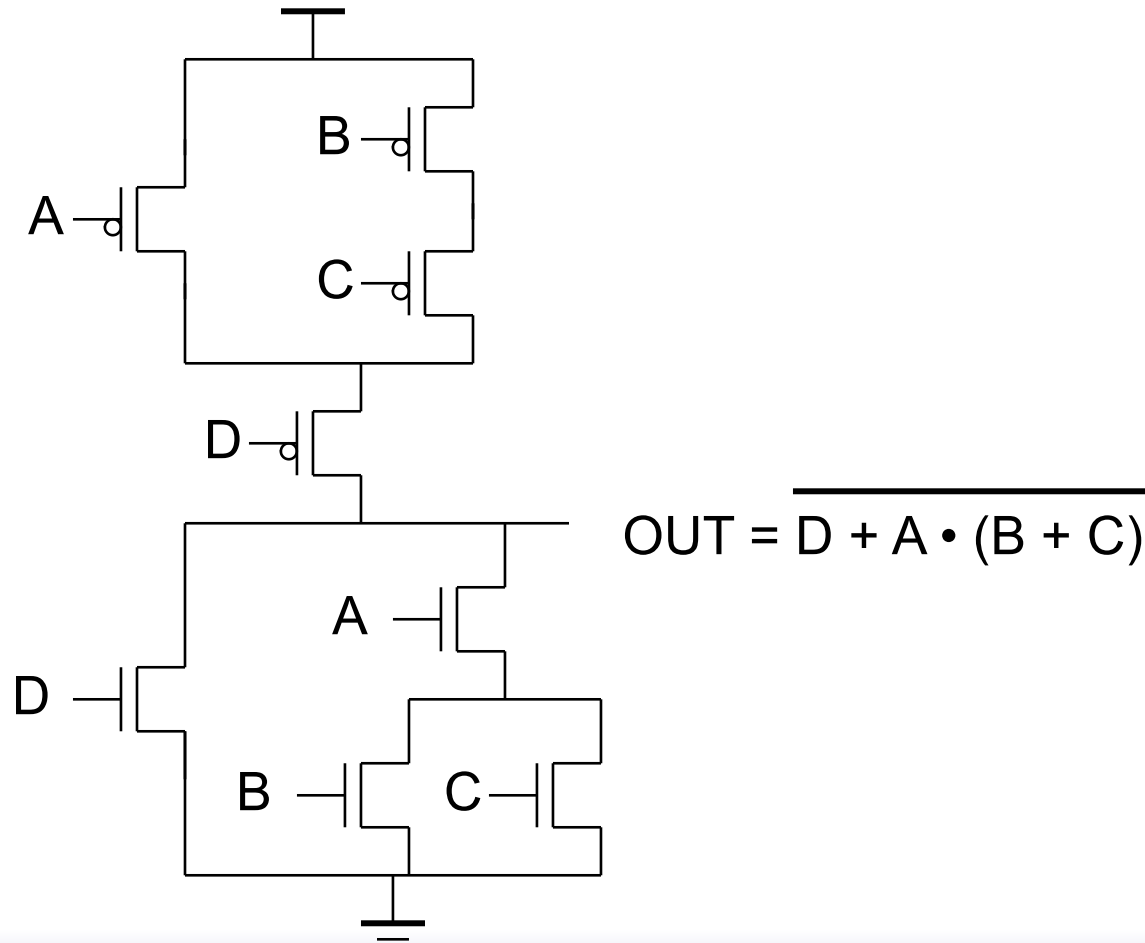
# Example Gate: NOR

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

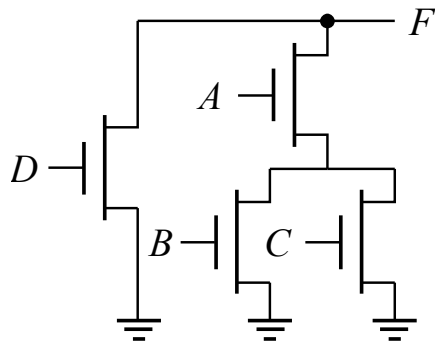
Truth Table of a 2 input NOR gate



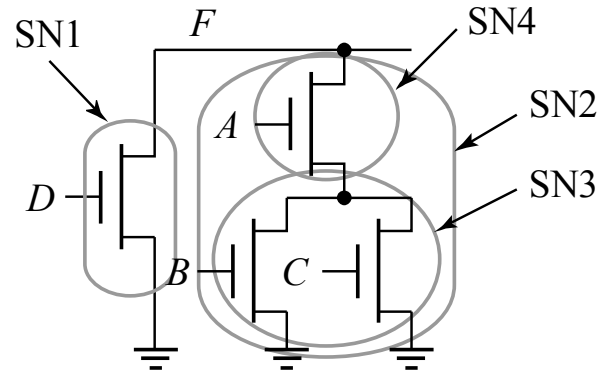
# Complex CMOS Gate



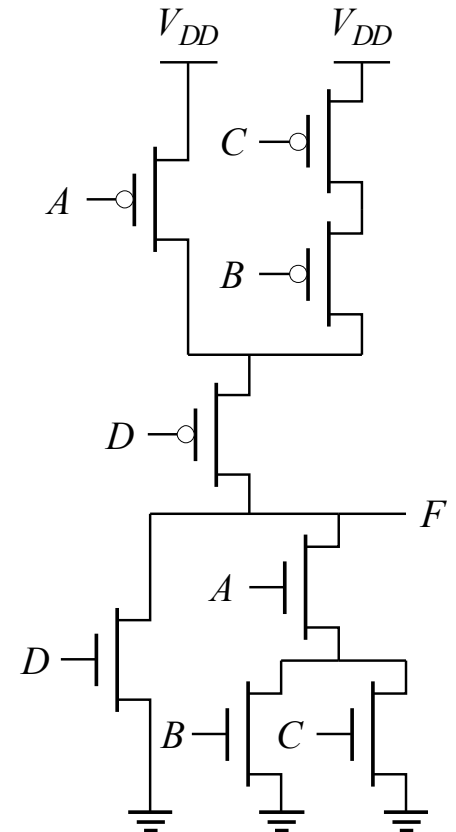
# Constructing a Complex Gate



(a) pull-down network



(b) Deriving the pull-up network hierarchically by identifying sub-nets



(c) complete gate

# Cell Design

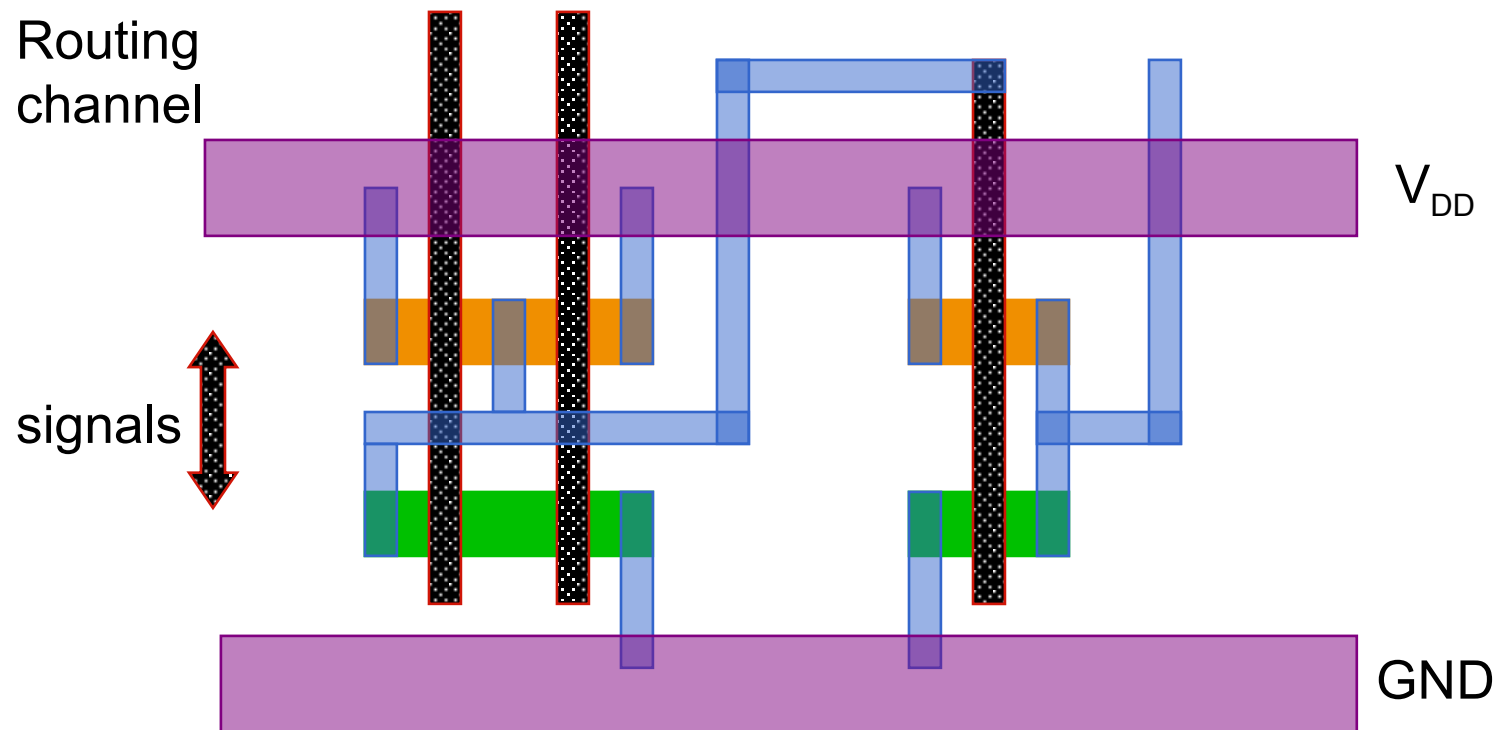
## □ Standard Cells

- General purpose logic
- Can be synthesized
- Same height, varying width

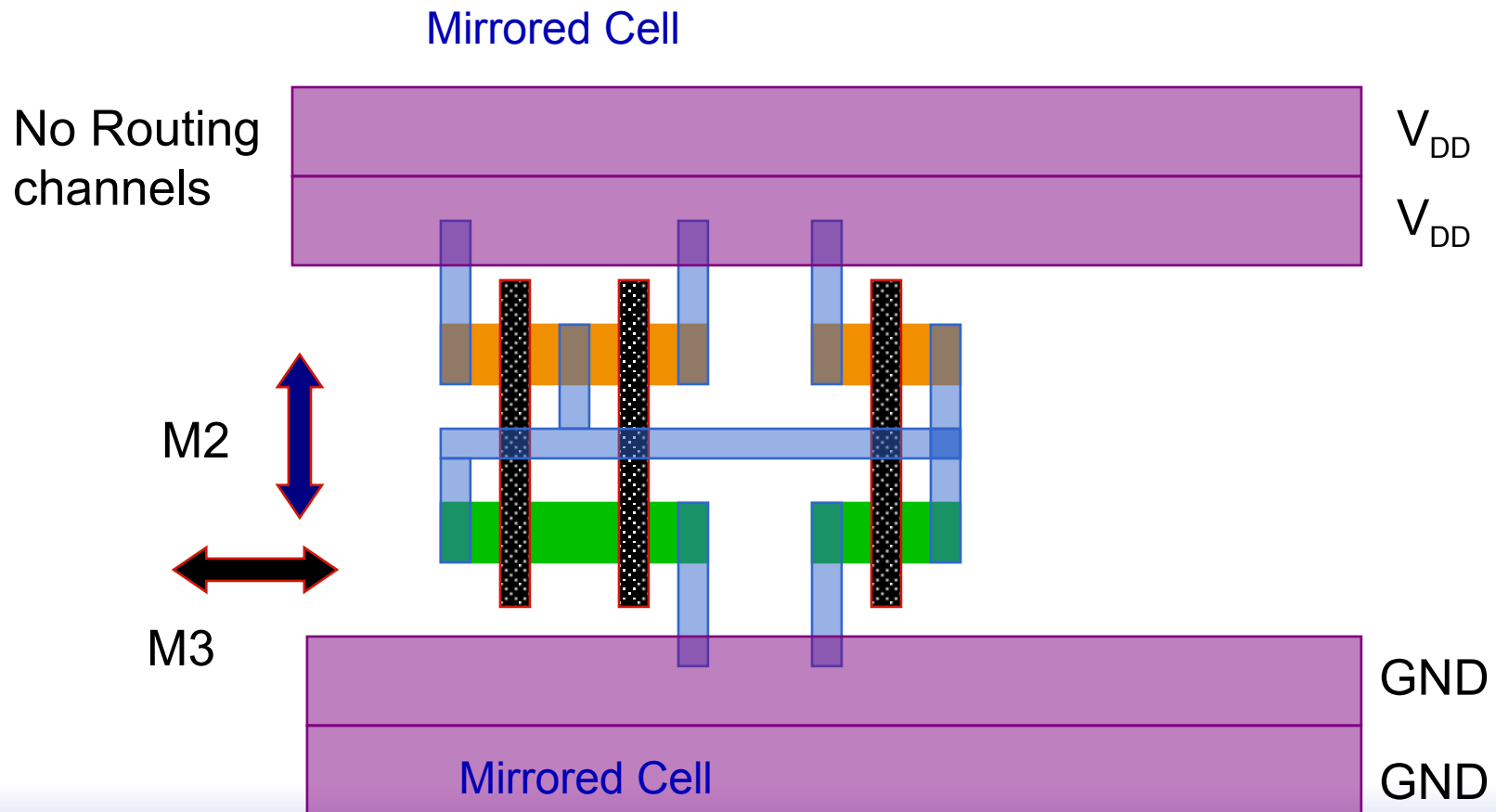
## □ Datapath Cells

- For regular, structured designs (arithmetic)
- Includes some wiring in the cell
- Fixed height and width

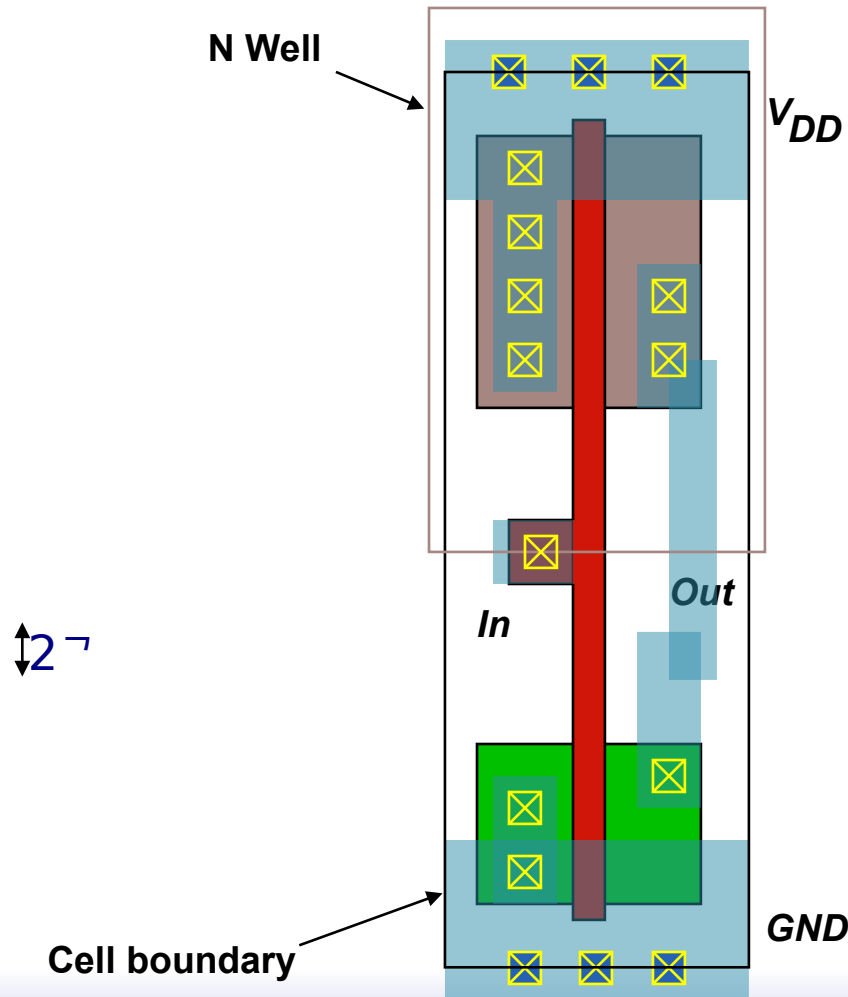
# Standard Cell Layout Methodology – 1980s



# Standard Cell Layout Methodology – 1990s



# Standard Cells



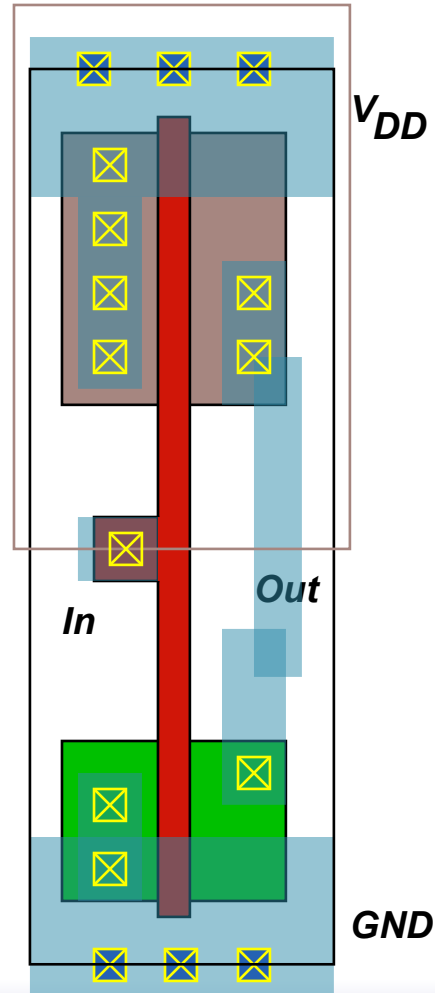
Cell height 12 metal tracks  
 Metal track is approx.  $3^{-7} + 3^{-7}$   
 Pitch =  
 repetitive distance between objects  
 Cell height is "12 pitch"

Rails  $\sim 10^{-7}$

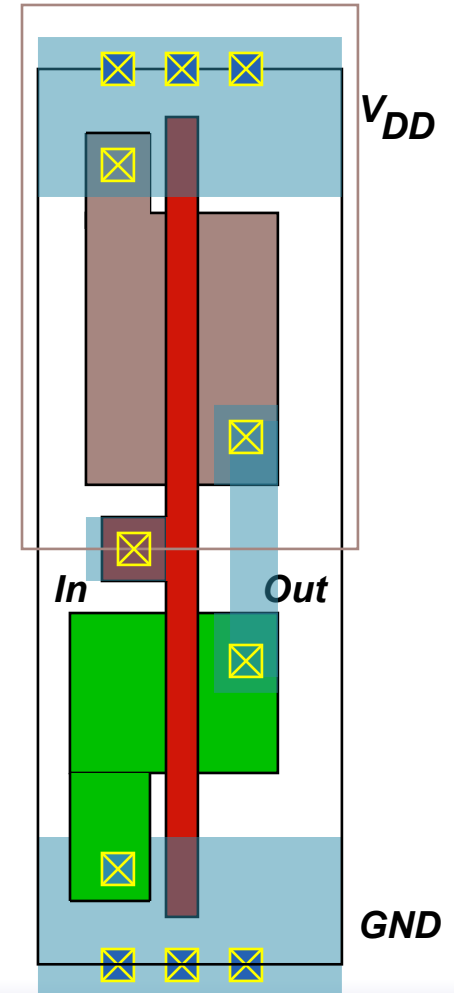


# Standard Cells

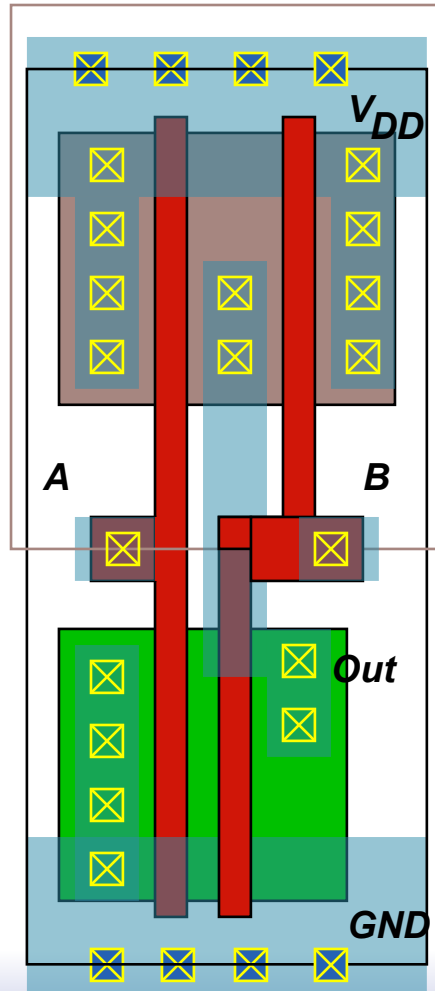
With minimal  
diffusion  
routing



With silicided  
diffusion



# Standard Cells



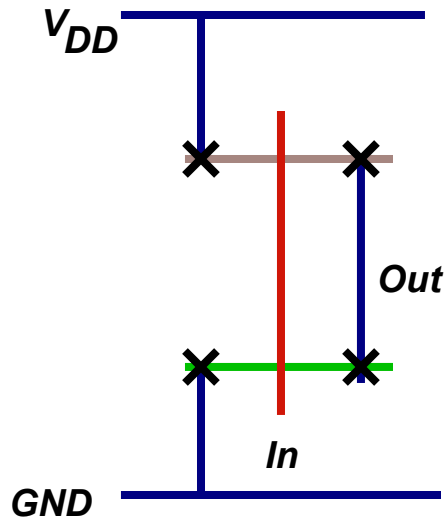
2-input NAND gate

# Stick Diagrams

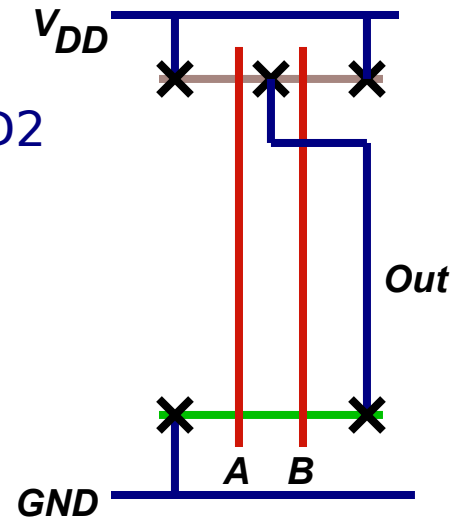
Contains no dimensions

Represents relative positions of transistors

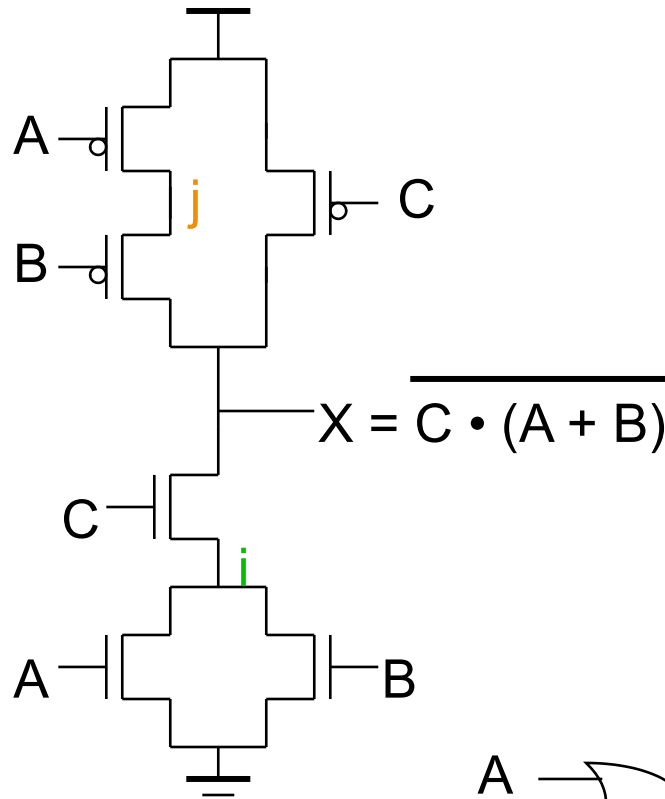
Inverter



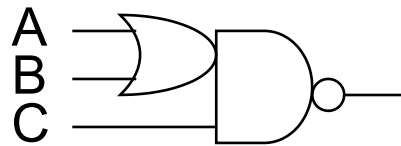
NAND2



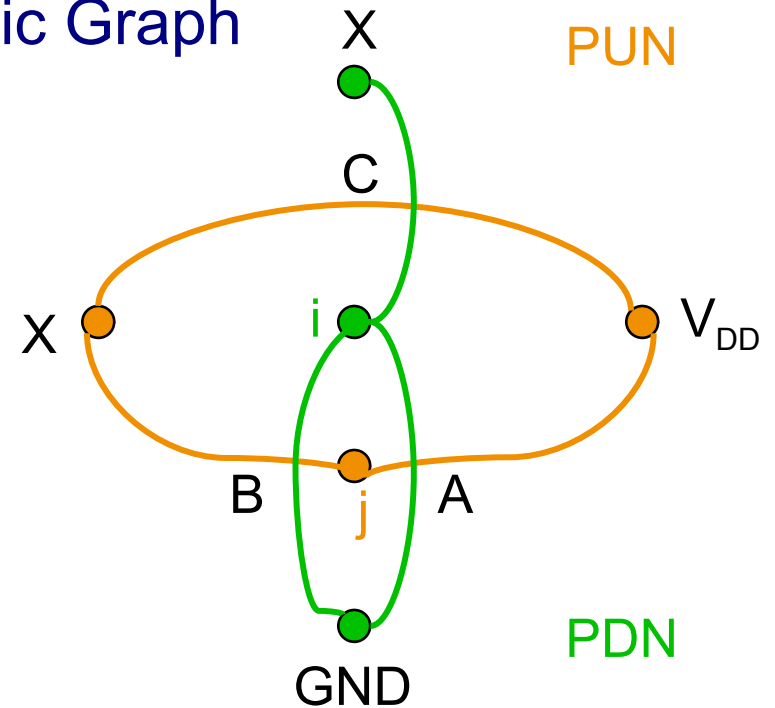
# Stick Diagrams



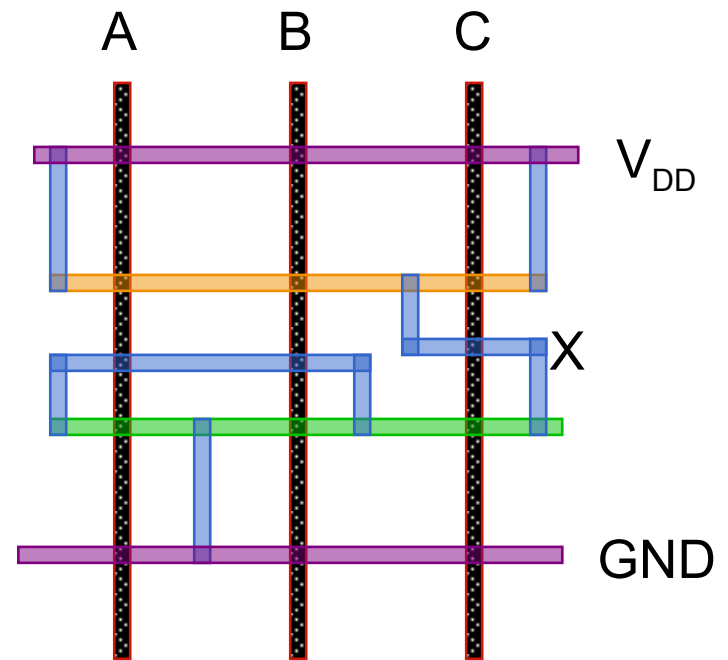
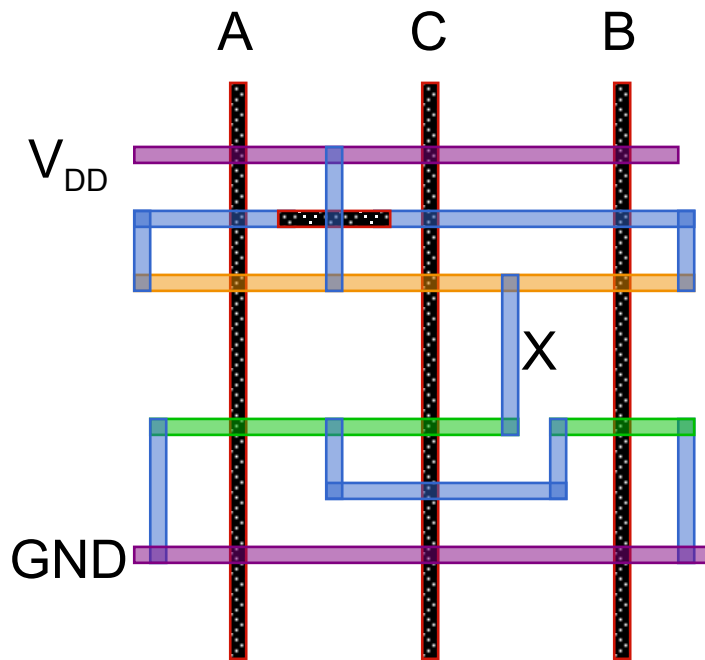
$$X = C \cdot (A + B)$$



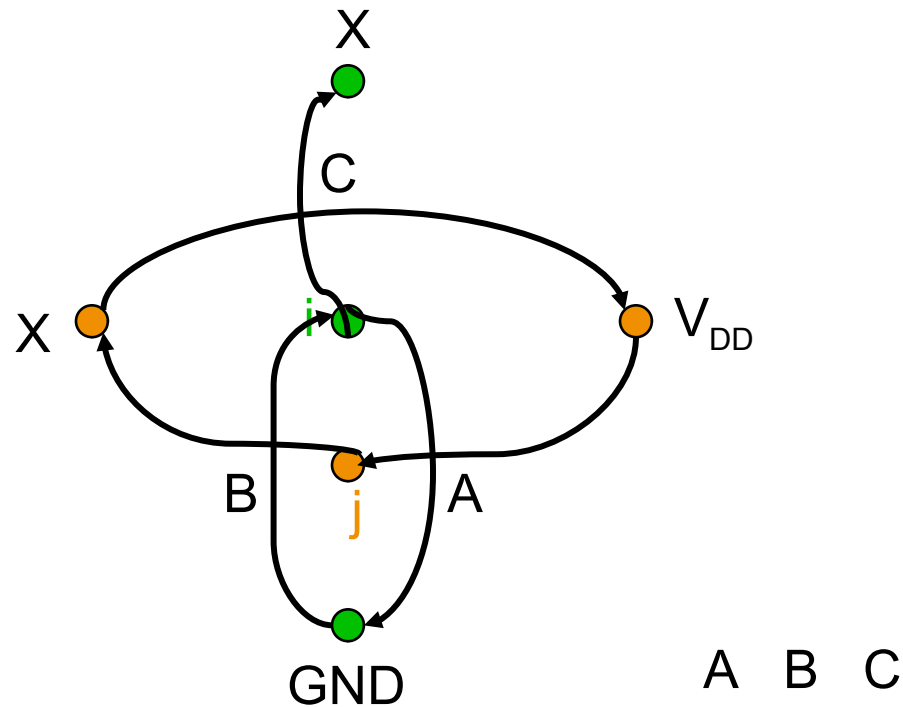
Logic Graph



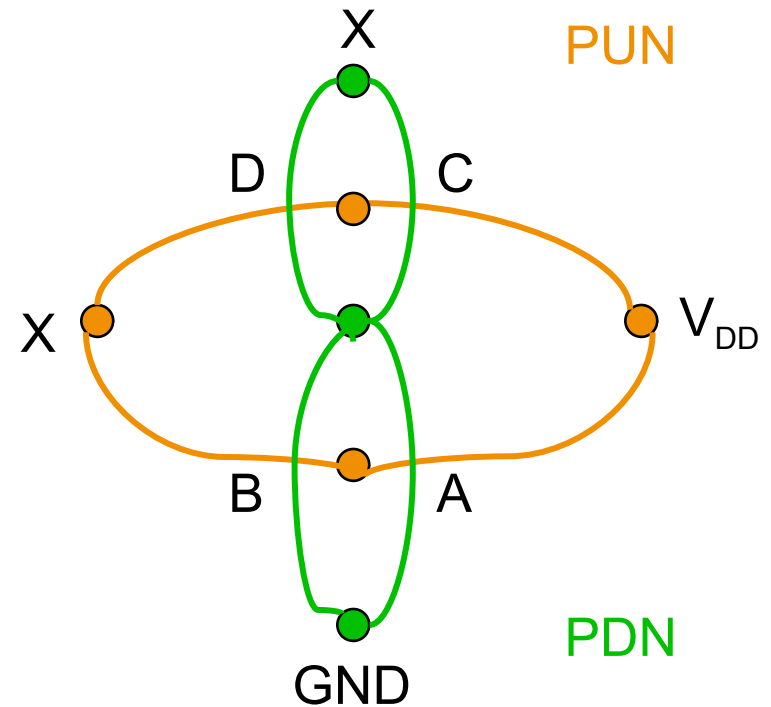
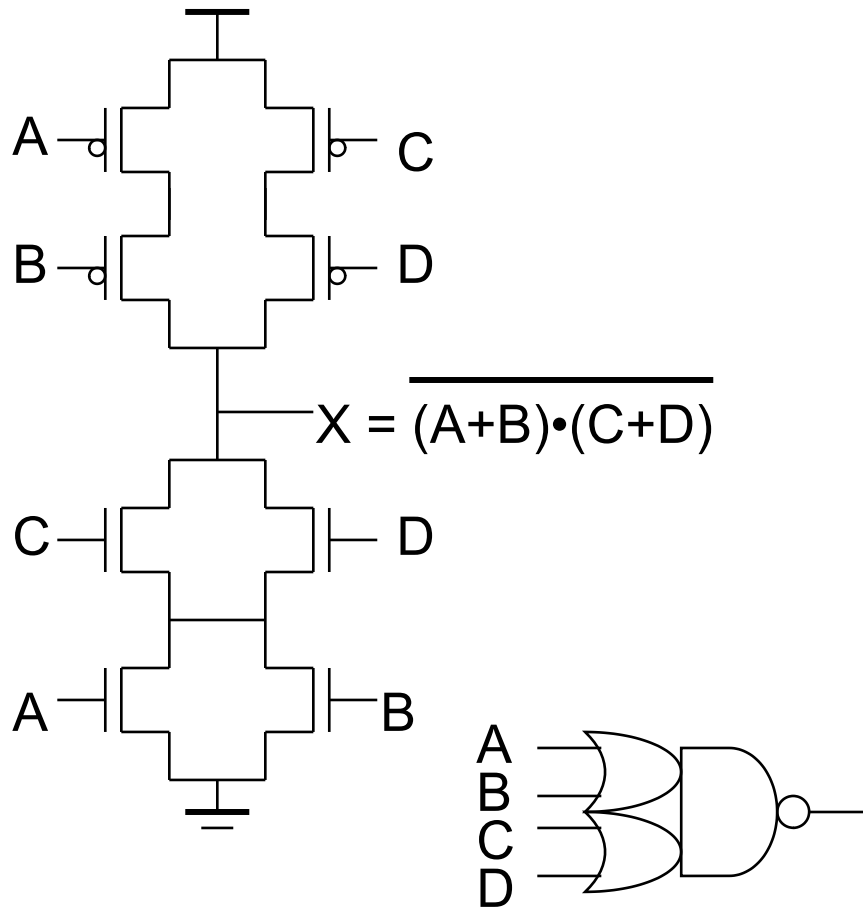
# *Two Versions of $C \cdot (A + B)$*



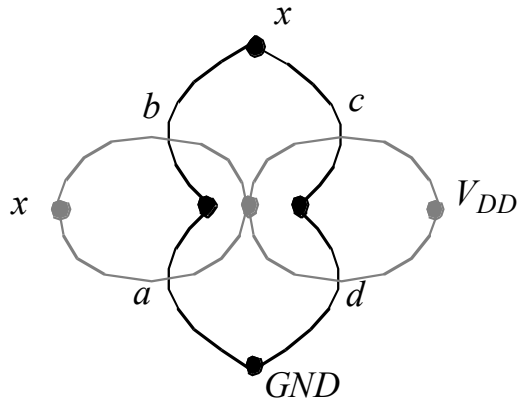
# Consistent Euler Path



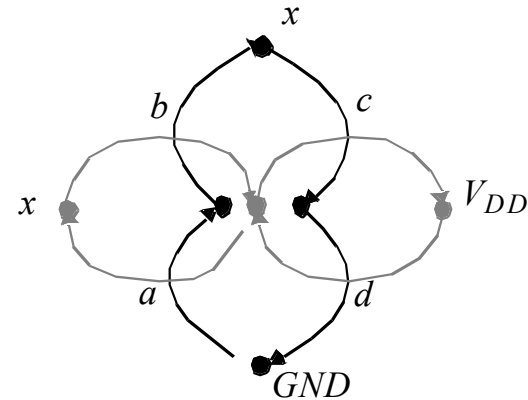
# OAI22 Logic Graph



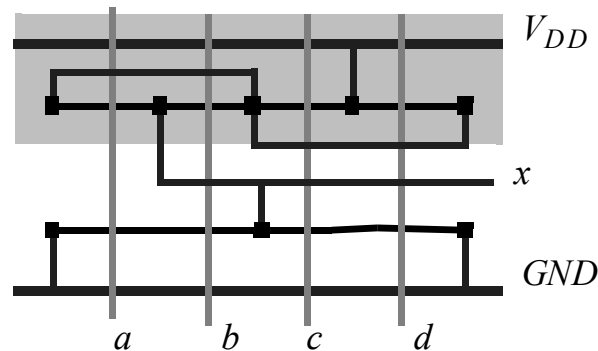
# Example: $x = ab + cd$



(a) Logic graphs for  $\overline{ab+cd}$



(b) Euler Paths  $\{a b c d\}$

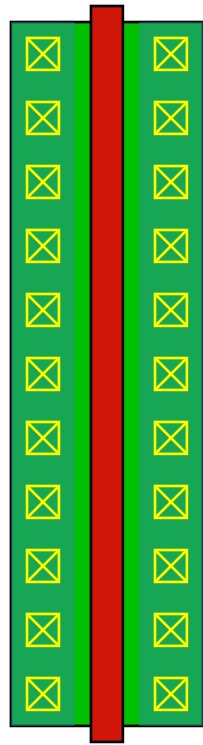


(c) stick diagram for ordering  $\{a b c d\}$

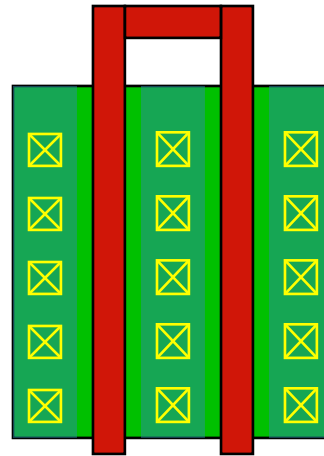


# Multi-Fingered Transistors

One finger



Two fingers (folded)



Less diffusion capacitance

# Properties of Complementary CMOS Gates

## Snapshot

***High noise margins:***

**$V_{OH}$  and  $V_{OL}$  are at  $V_{DD}$  and  $GND$ , respectively.**

***No static power consumption:***

**There never exists a direct path between  $V_{DD}$  and  $V_{SS}$  ( $GND$ ) in steady-state mode.**

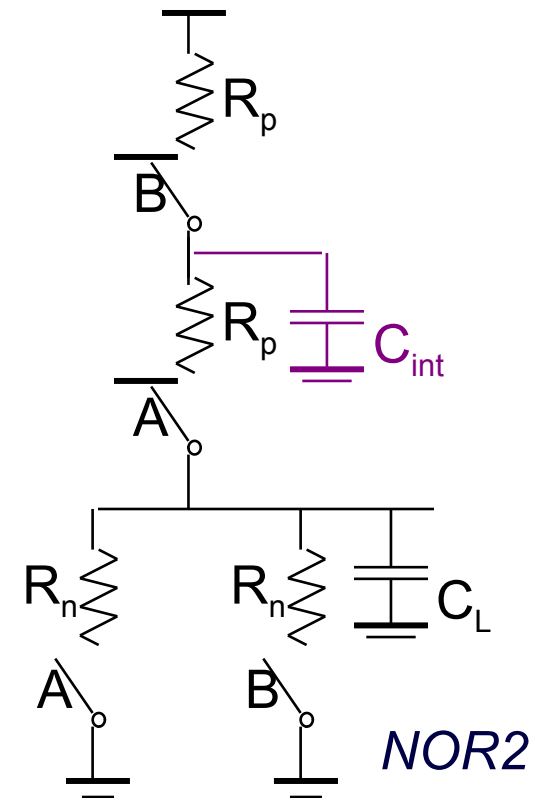
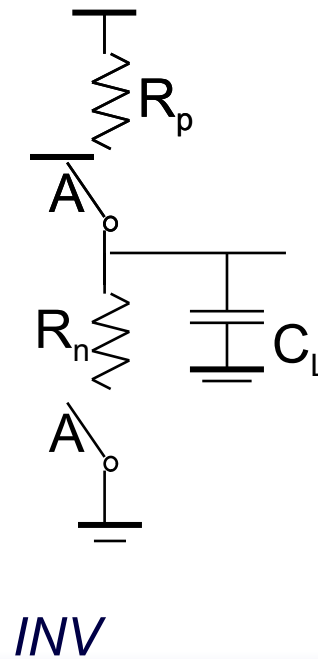
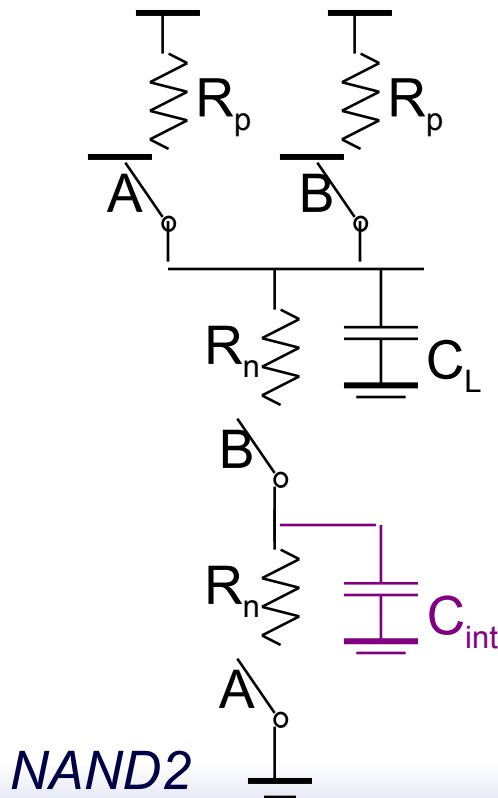
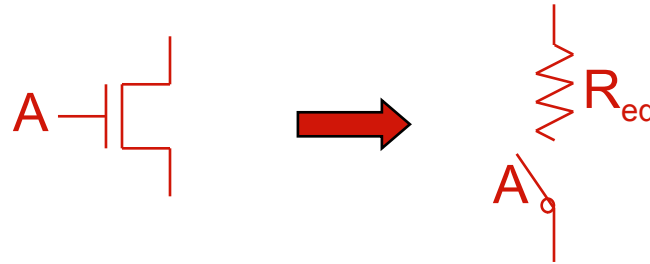
***Comparable rise and fall times:***

**(under appropriate sizing conditions)**

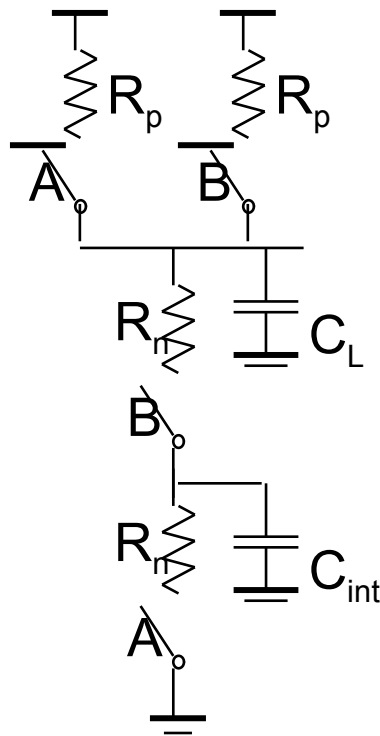
# CMOS Properties

- ❑ 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**
- ❑ Propagation delay function of load capacitance and resistance of transistors

# Switch Delay Model



# Input Pattern Effects on Delay



- Delay is dependent on the **pattern** of inputs
- Low to high transition
  - both inputs go low
    - delay is  $0.69 R_p/2 C_L$
  - one input goes low
    - delay is  $0.69 R_p C_L$
- High to low transition
  - both inputs go high
    - delay is  $0.69 2R_n C_L$

# Delay Dependence on Input Patterns



Input Data Pattern	Delay (psec)
A=B=0 to 1	67
A=1, B=0 to 1	64
A= 0 to 1, B=1	61
A=B=1 to 0	45
A=1, B=1 to 0	80
A= 1 to 0, B=1	81

$$\text{NMOS} = 0.5 \frac{\mu\text{m}}{\text{m}} / 0.25$$

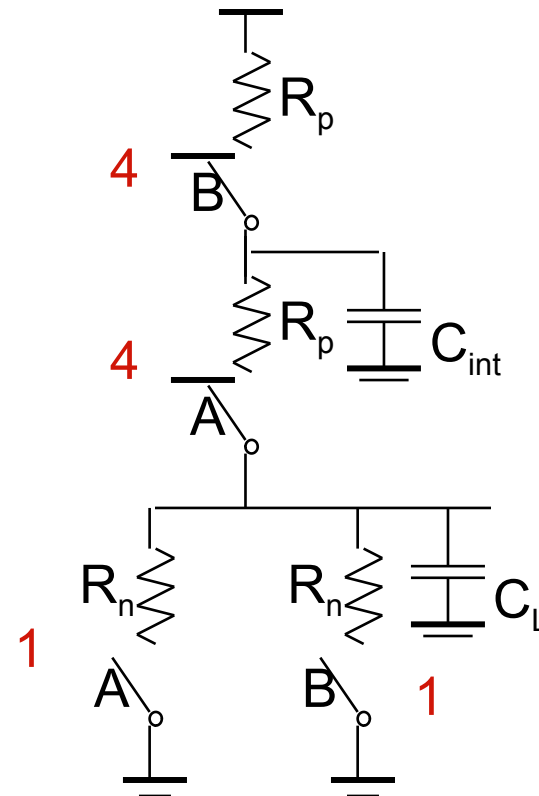
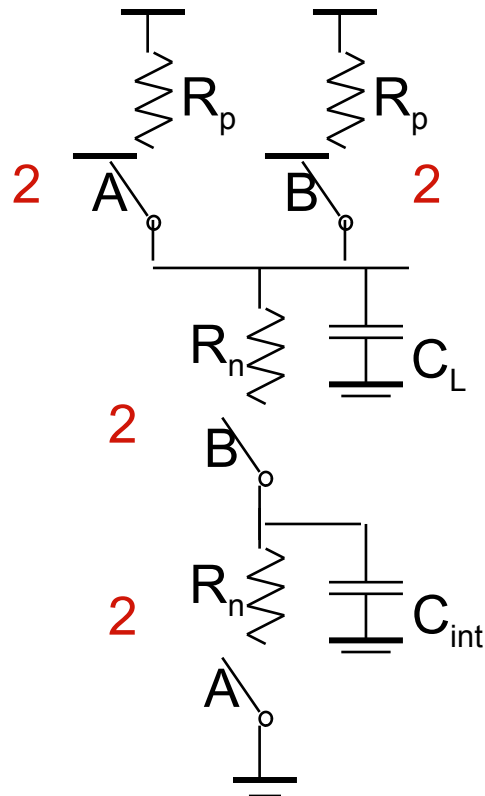
$$\frac{\mu\text{m}}{\text{m}}$$

$$\text{PMOS} = 0.75 \frac{\mu\text{m}}{\text{m}} / 0.25$$

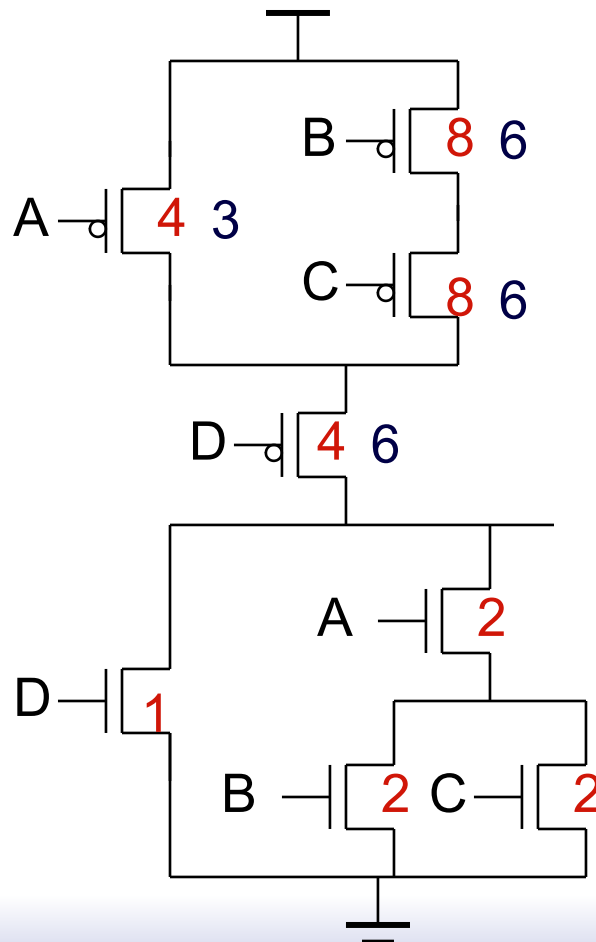
$$\frac{\mu\text{m}}{\text{m}}$$

Combinational Circuits

# Transistor Sizing



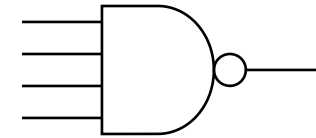
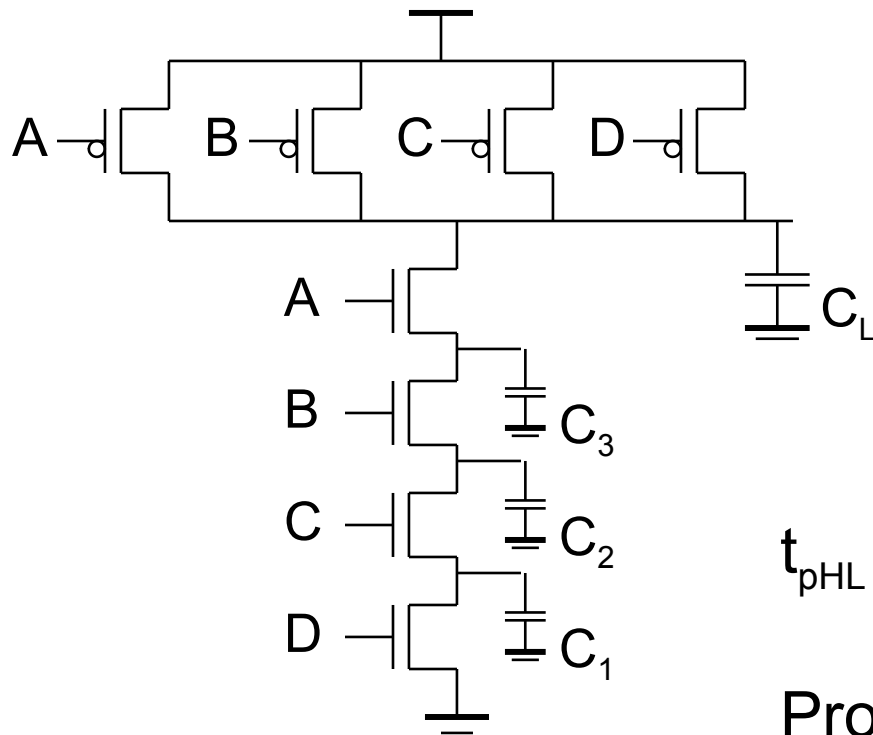
# Transistor Sizing a Complex CMOS Gate



$$\text{OUT} = \overline{D + A \cdot (B + C)}$$



# Fan-In Considerations

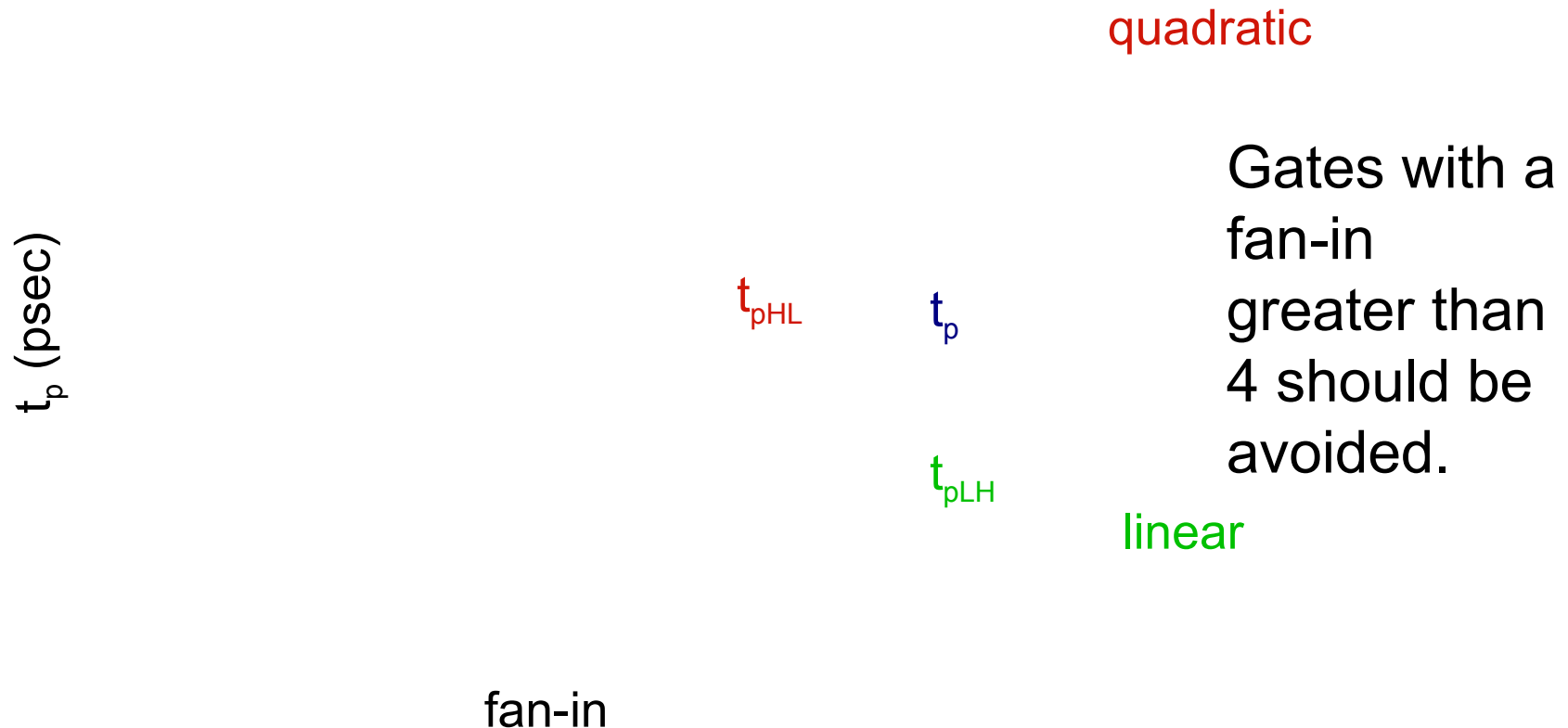


Distributed RC model  
(Elmore delay)

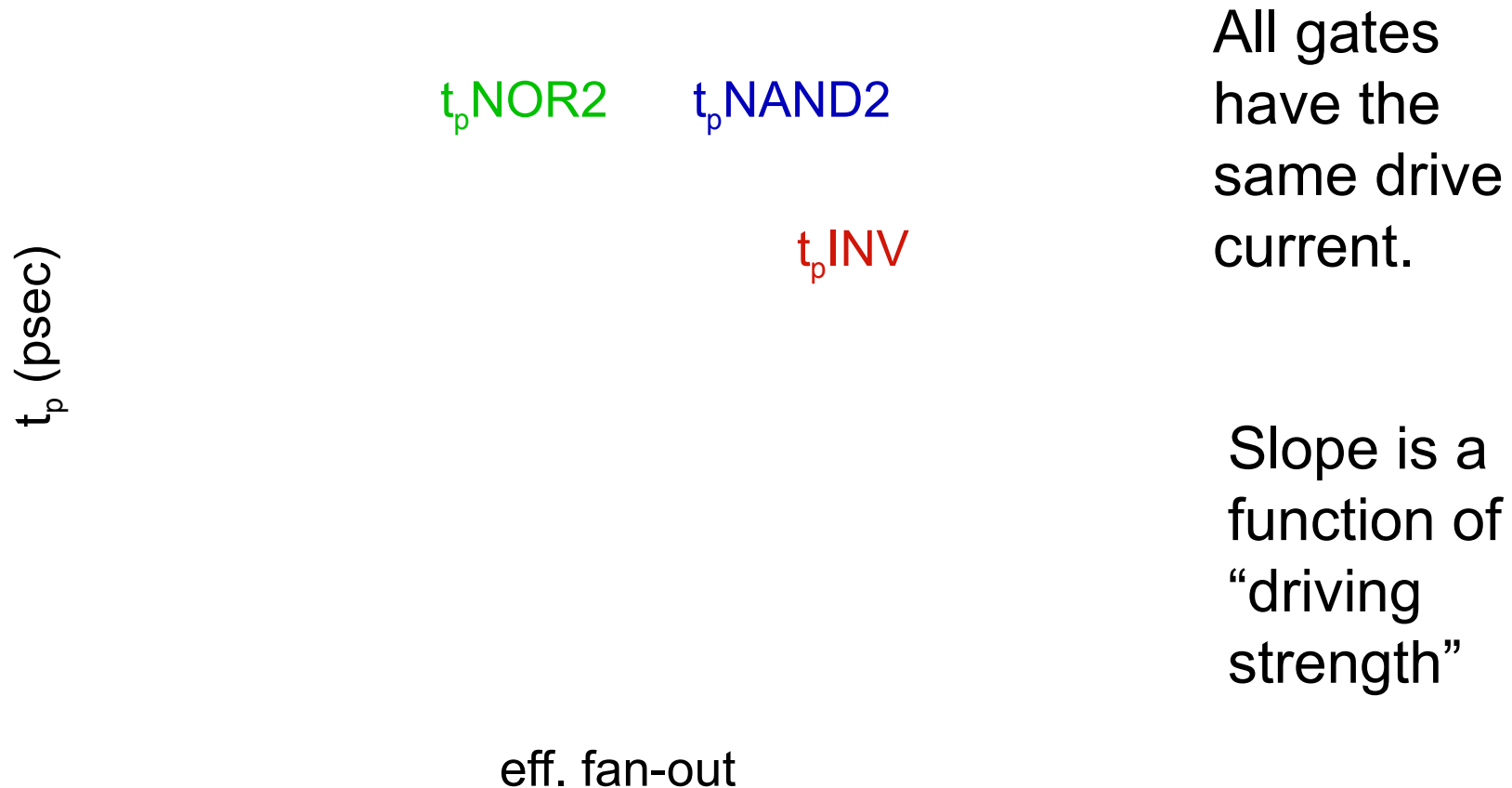
$$t_{pHL} = 0.69 R_{eqn} (C_1 + 2C_2 + 3C_3 + 4C_L)$$

Propagation delay deteriorates rapidly as a function of fan-in – **quadratically** in the worst case.

# $t_p$ as a Function of Fan-In



# $t_p$ as a Function of Fan-Out



# $t_p$ as a Function of Fan-In and Fan-Out

- Fan-in: **quadratic** due to increasing resistance and capacitance
- Fan-out: each additional fan-out gate adds **two** gate capacitances to  $C_L$

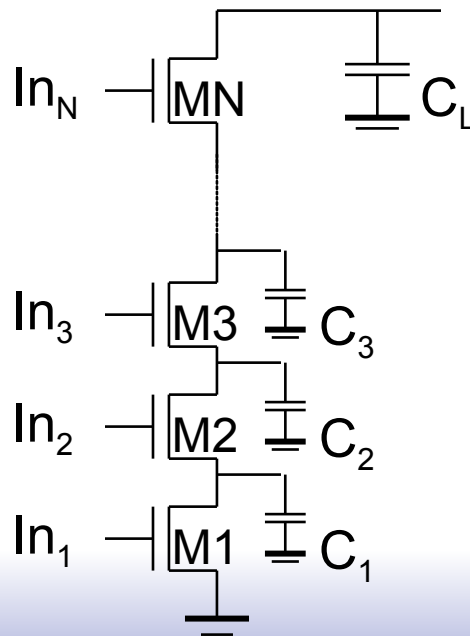
$$t_p = a_1 FI + a_2 FI^2 + a_3 FO$$

# Fast Complex Gates: Design Technique 1

## □ Transistor sizing

- as long as fan-out capacitance dominates

## □ Progressive sizing



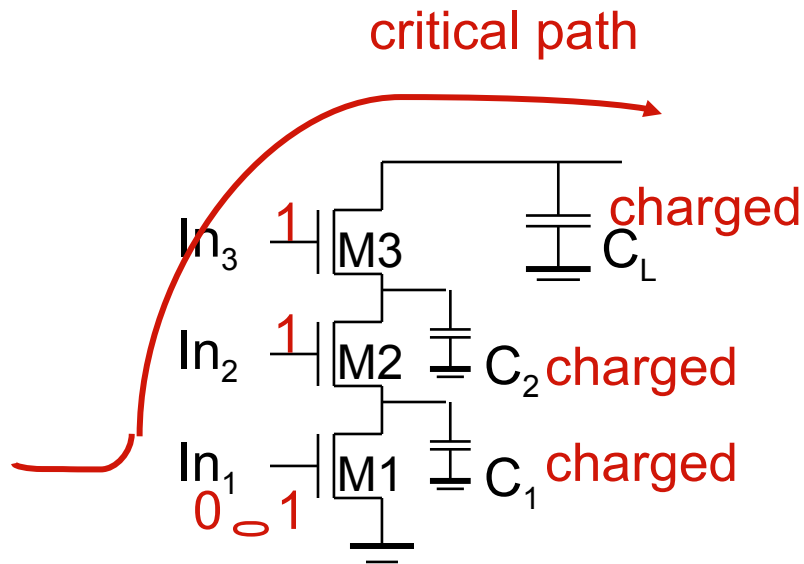
Distributed RC line

$M1 > M2 > M3 > \dots > MN$   
(the fet closest to the  
output is the smallest)

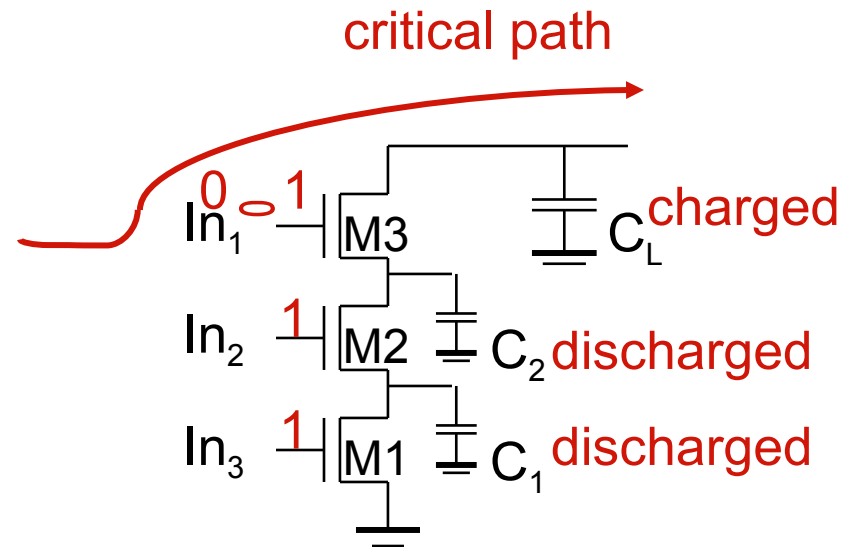
Can reduce delay by more than  
20%; decreasing gains as  
technology shrinks

# Fast Complex Gates: Design Technique 2

## □ Transistor ordering



delay determined by time to discharge  $C_L$ ,  $C_1$  and  $C_2$

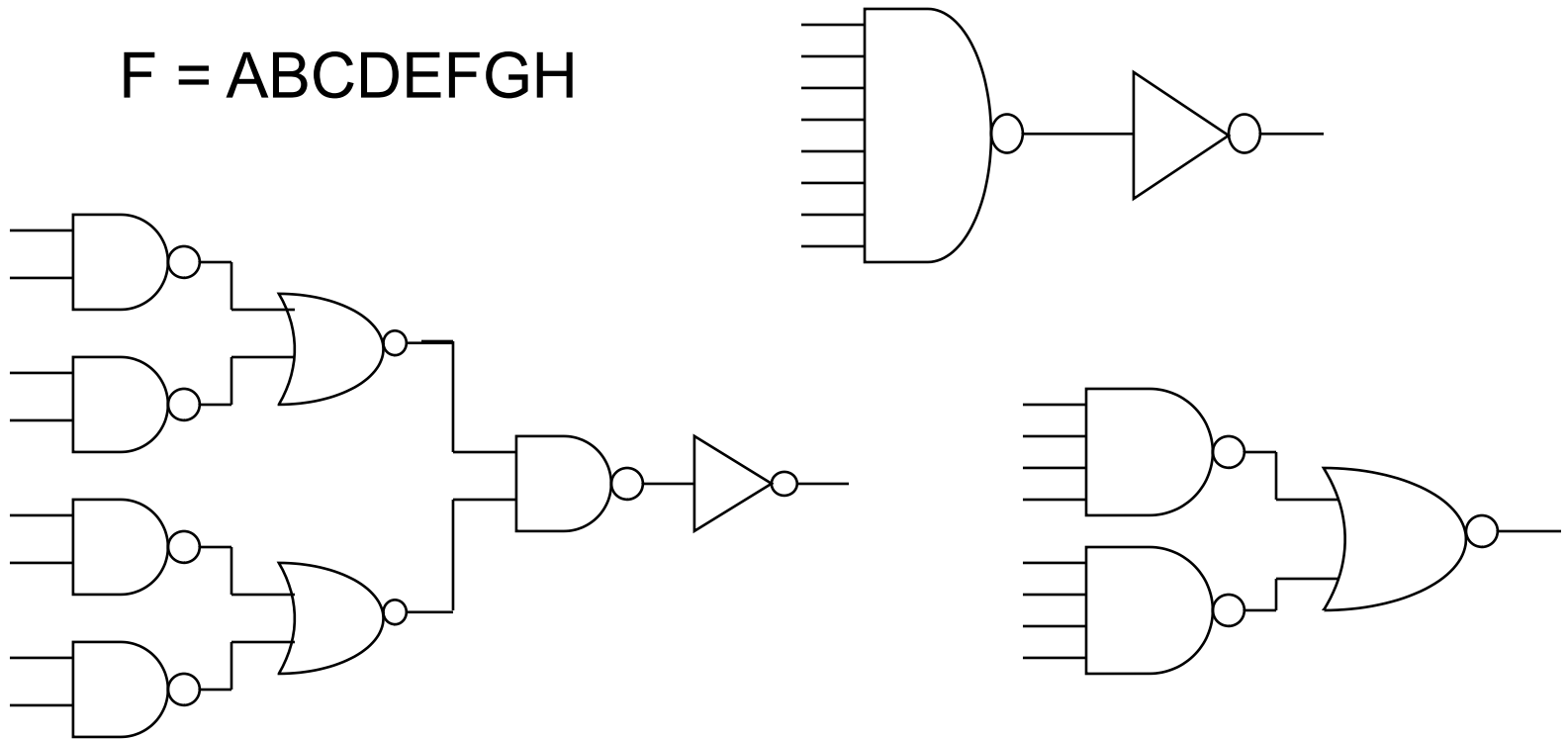


delay determined by time to discharge  $C_L$

# Fast Complex Gates: Design Technique 3

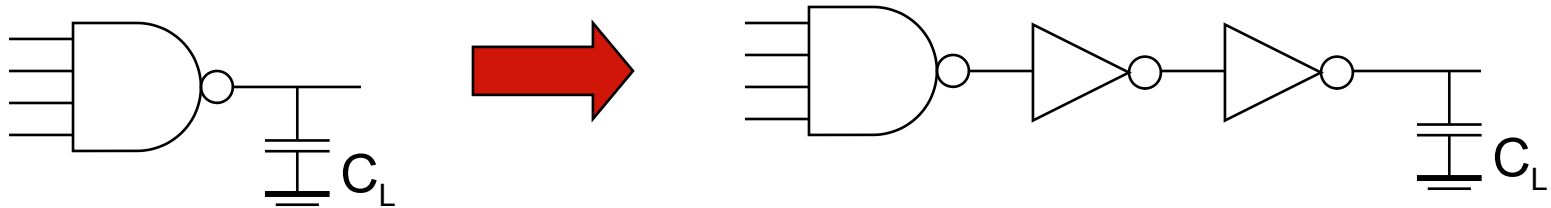
## □ Alternative logic structures

$$F = ABCDEFGH$$



# Fast Complex Gates: Design Technique 4

- Isolating fan-in from fan-out using buffer insertion





# Fast Complex Gates:

## Design Technique 5

- ❑ Reducing the voltage swing

$$t_{pHL} = 0.69 (3/4 (C_L V_{DD}) / I_{DSATn})$$

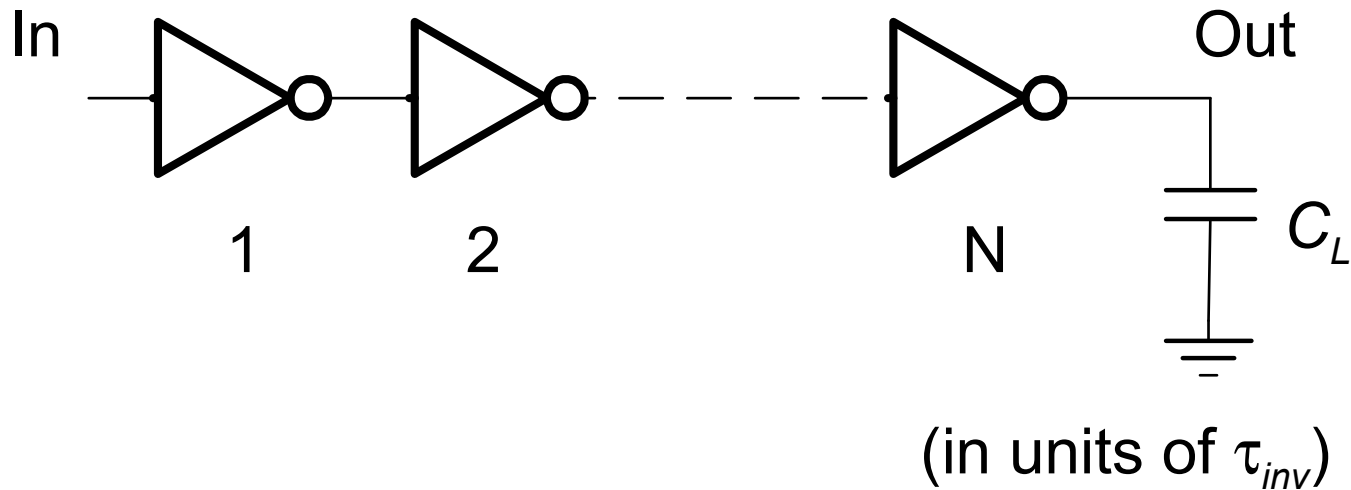
$$= 0.69 (3/4 (C_L V_{swing}) / I_{DSATn})$$

- linear reduction in delay
  - also reduces power consumption
- ❑ But the following gate is much slower!
  - ❑ Or requires use of “sense amplifiers” on the receiving end to restore the signal level (memory design)

# *Sizing Logic Paths for Speed*

- ❑ Frequently, input capacitance of a logic path is constrained
- ❑ Logic also has to drive some capacitance
- ❑ Example: ALU load in an Intel's microprocessor is 0.5pF
- ❑ How do we size the ALU datapath to achieve maximum speed?
- ❑ We have already solved this for the inverter chain – can we generalize it for any type of logic?

# Buffer Example



For given  $N$ :  $C_{i+1}/C_i = C_i/C_{i-1}$

To find  $N$ :  $C_{i+1}/C_i \sim 4$

How to generalize this to any logic path?

# Logical Effort

$p$  – intrinsic delay ( $3kR_{\text{unit}}C_{\text{unit}}\gamma$ ) - gate parameter  $\propto f(W)$

$g$  – logical effort ( $kR_{\text{unit}}C_{\text{unit}}$ ) – gate parameter  $\propto f(W)$

$f$  – effective fanout

Normalize everything to an inverter:

$$g_{\text{inv}} = 1, p_{\text{inv}} = 1$$

Divide everything by  $\tau_{\text{inv}}$

(everything is measured in unit delays  $\tau_{\text{inv}}$ )

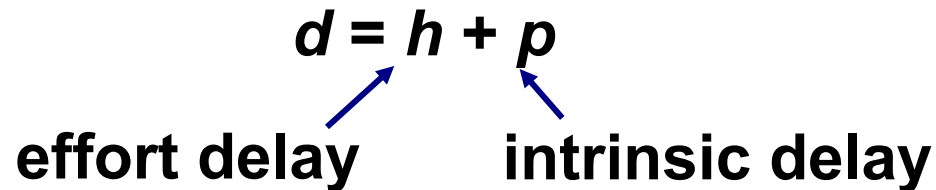
Assume  $\gamma = 1$ .

# Delay in a Logic Gate

**Gate delay:**

$$d = h + p$$

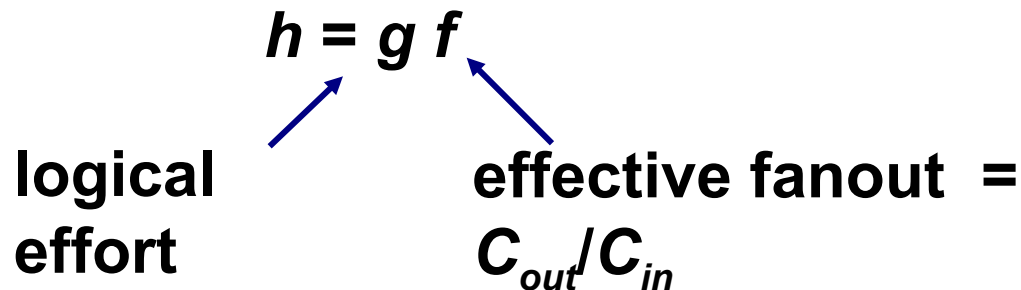
effort delay      intrinsic delay



**Effort delay:**

$$h = g f$$

logical effort      effective fanout =  $C_{out}/C_{in}$



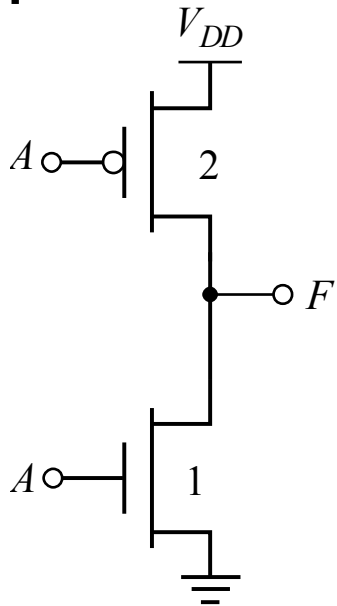
**Logical effort is a function of topology, independent of sizing**  
**Effective fanout (electrical effort) is a function of load/gate size**

# *Logical Effort*

- ❑ Inverter has the smallest logical effort and intrinsic delay of all static CMOS gates
- ❑ Logical effort of a gate presents the ratio of its input capacitance to the inverter capacitance when sized to deliver the same current
- ❑ Logical effort increases with the gate complexity

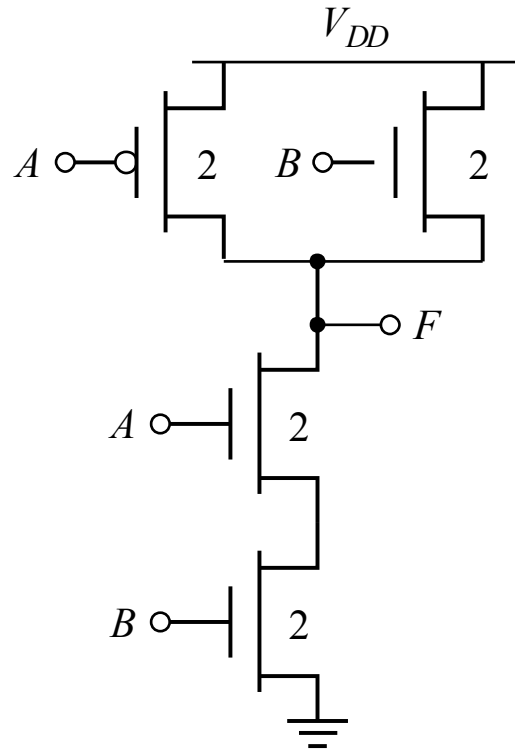
# Logical Effort

Logical effort is the ratio of input capacitance of a gate to the input capacitance of an inverter with the same output current



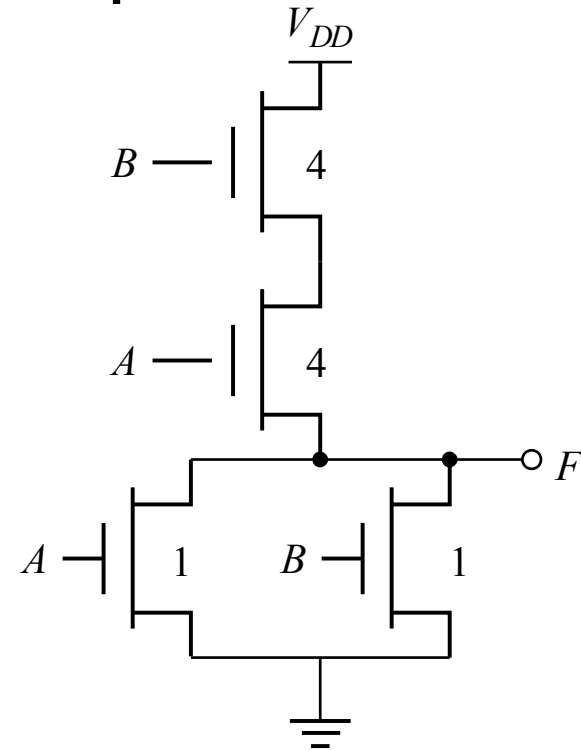
Inverter

$$g = 1$$



2-input NAND

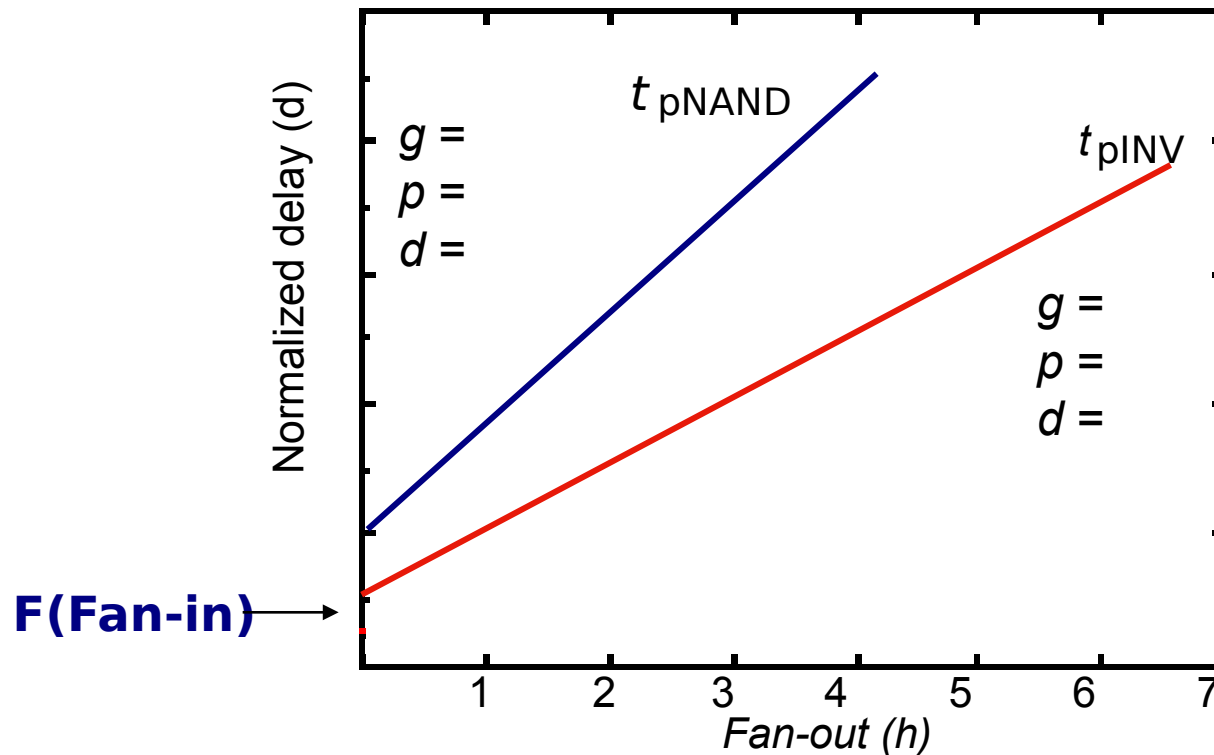
$$g = 4/3$$



2-input NOR

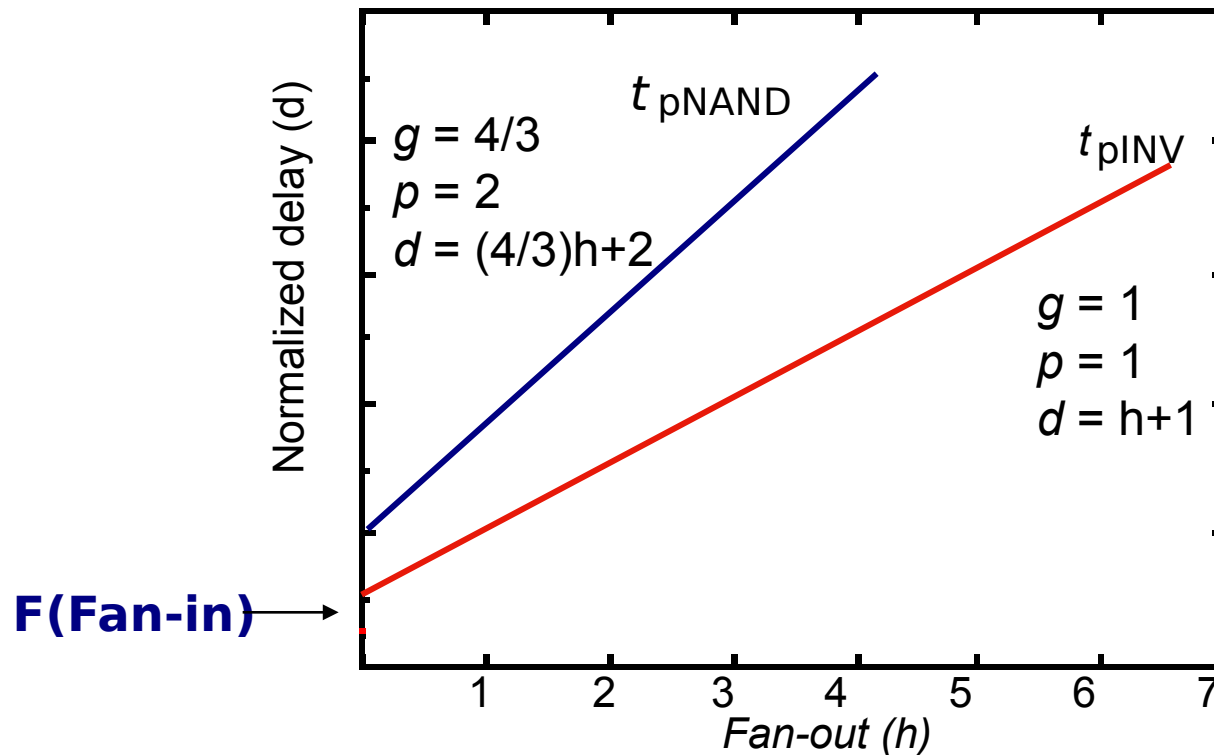
$$g = 5/3$$

# Logical Effort of Gates

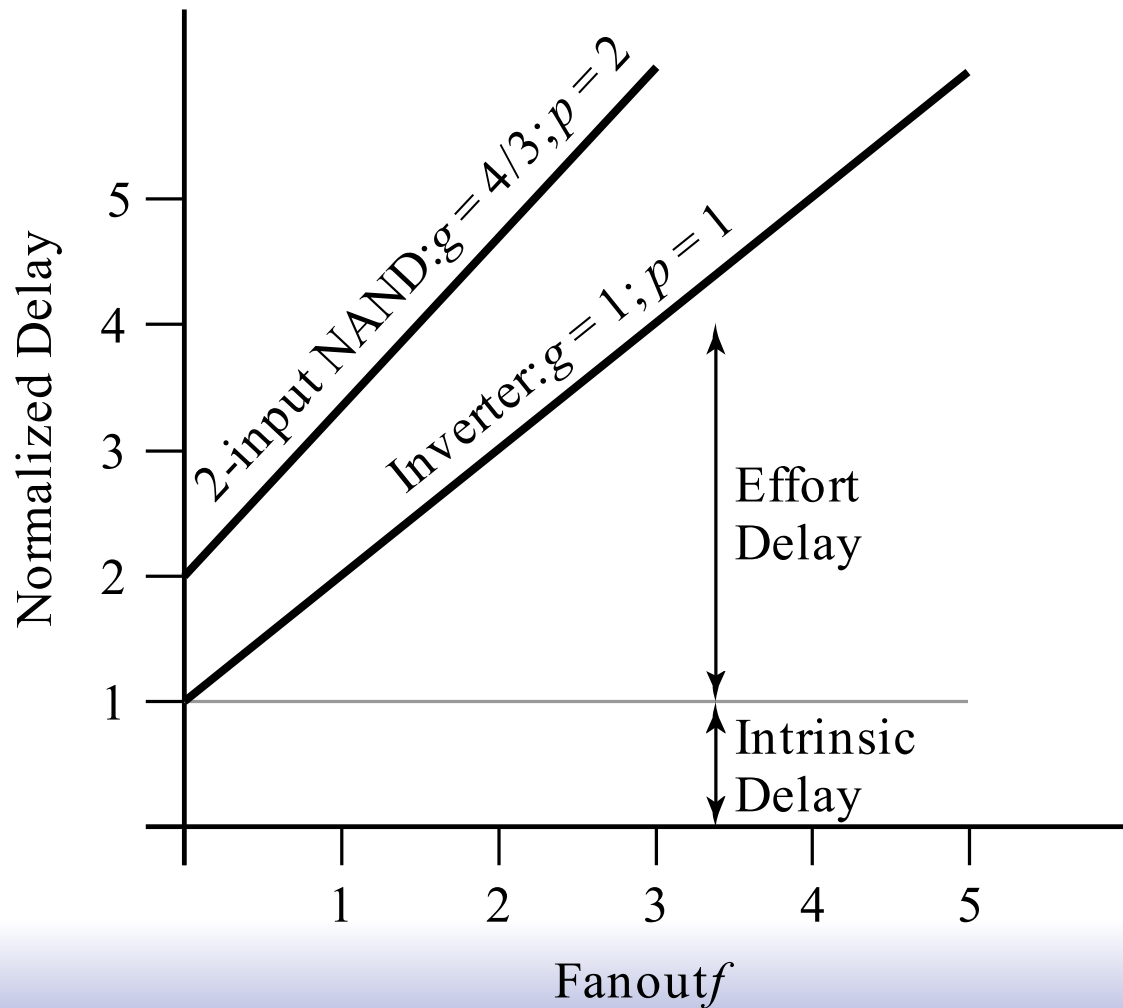




# Logical Effort of Gates

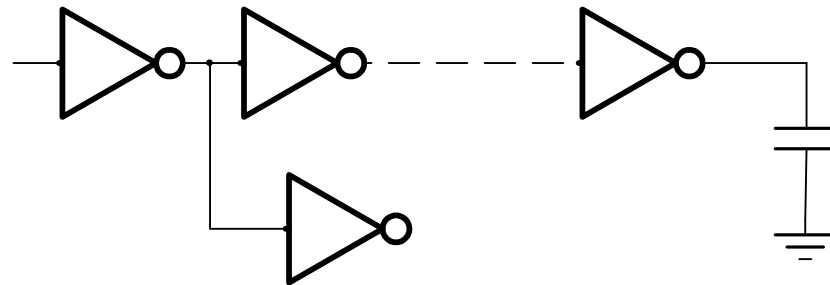


# Logical Effort of Gates



# *Add Branching Effort*

Branching effort:



# Multistage Networks

Stage effort:  $h_i = g_i f_i$

Path electrical effort:  $F = C_{out}/C_{in}$

Path logical effort:  $G = g_1 g_2 \dots g_N$

Branching effort:  $B = b_1 b_2 \dots b_N$

Path effort:  $H = GFB$

Path delay  $D = \sum d_i = \sum p_i + \sum h_i$

# *Optimum Effort per Stage*

When each stage bears the same effort:

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

Effective fanout of each stage:

Minimum path delay

# *Optimal Number of Stages*

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

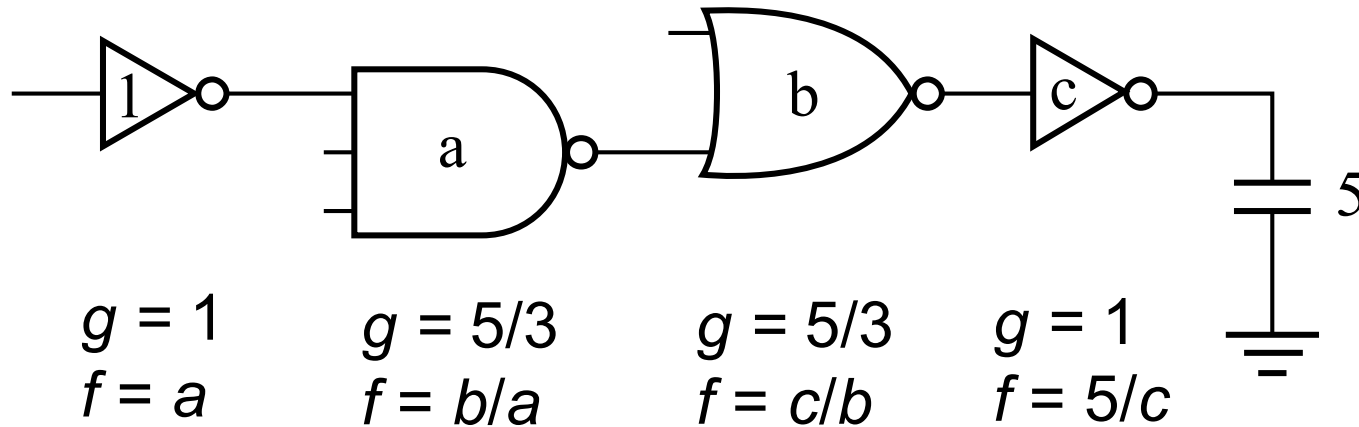
Substitute 'best stage effort'

# Logical Effort

Gate Type	Number of Inputs			
	1	2	3	n
Inverter	1			
NAND		$4/3$	$5/3$	$(n + 2)/3$
NOR		$5/3$	$7/3$	$(2n + 1)/3$
Multiplexer		2	2	2
XOR		4	12	

From Sutherland, Sproull

# Example: Optimize Path



Effective fanout,  $F =$

$G =$

$H =$

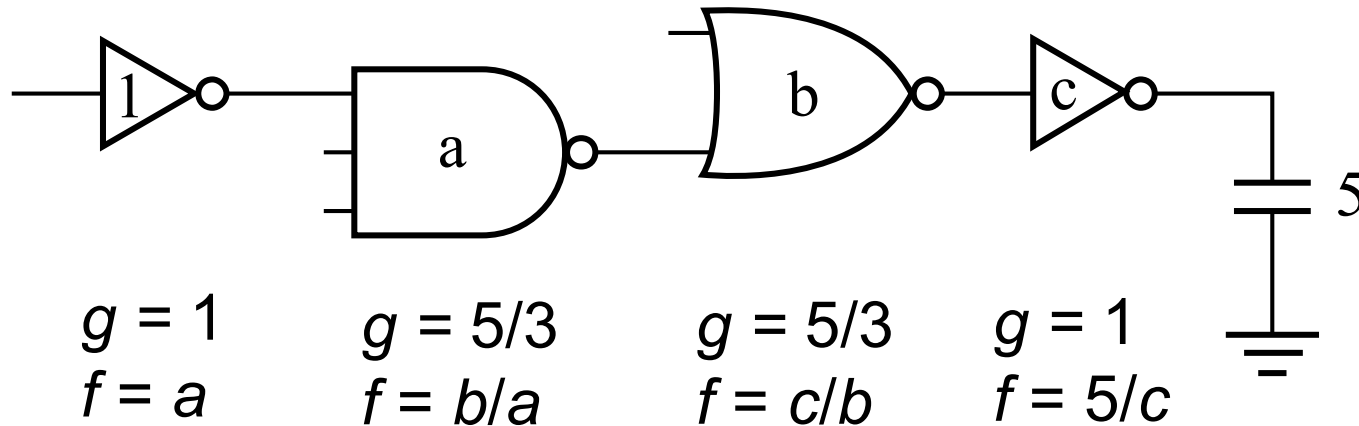
$h =$

$a =$

$b =$



# Example: Optimize Path



Effective fanout,  $F = 5$

$G = 25/9$

$H = 125/9 = 13.9$

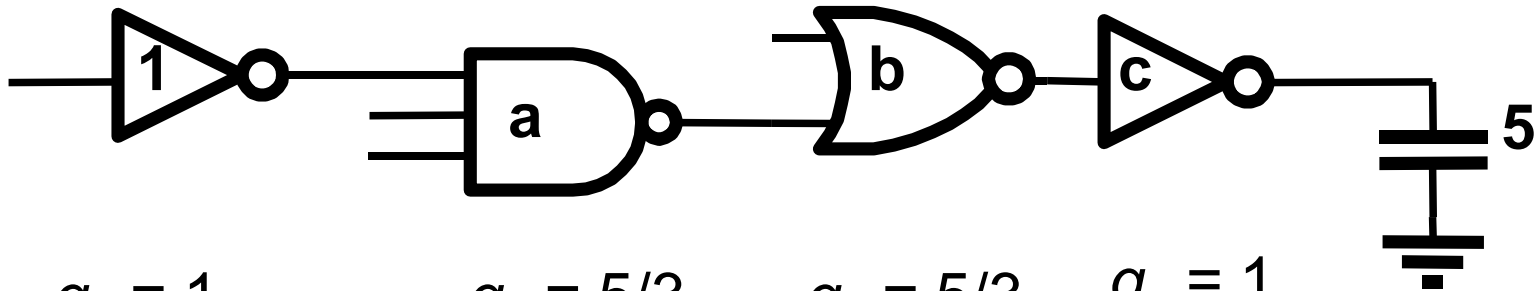
$h = 1.93$

$a = 1.93$

$b = ha/g_2 = 2.23$

$c = hb/g_3 = 5g_4/f = 2.59$

# Example: Optimize Path



$$g_1 = 1$$

$$g_2 = 5/3$$

$$g_3 = 5/3$$

$$g_4 = 1$$

*Effective fanout,  $H = 5$*

$$G = 25/9$$

$$F = 125/9 = 13.9$$

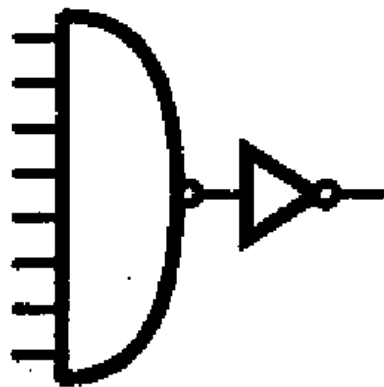
$$f = 1.93$$

$$a = 1.93$$

$$b = fa/g_2 = 2.23$$

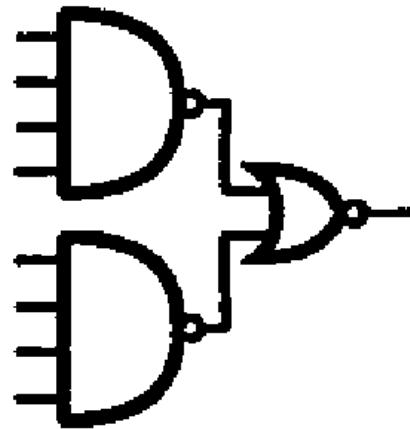
$$c = fb/g_3 = 5g_4/f = 2.59$$

## Example – 8-input AND



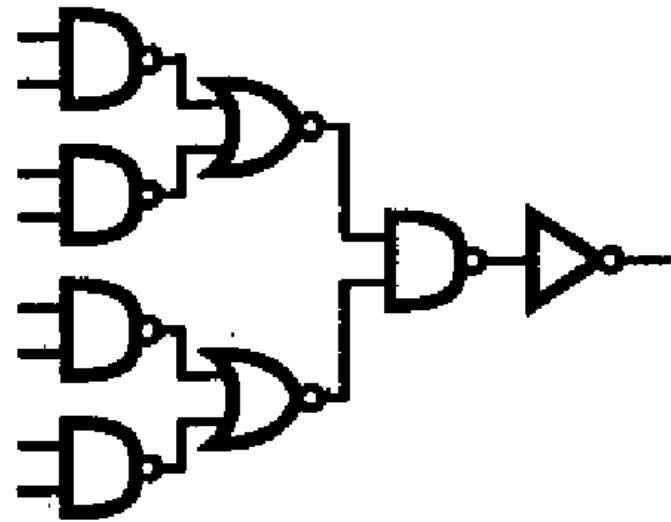
$$g=10/3 \quad g=1$$

(a)



$$g=2 \quad g=5/3$$

(b)



$$g=4/3 \quad g=5/3 \quad g=4/3 \quad g=1$$

(c)

# Method of Logical Effort

- ❑ Compute the path effort:  $F = GBH$
- ❑ Find the best number of stages  $N \sim \log_4 F$
- ❑ Compute the stage effort  $f = F^{1/N}$
- ❑ Sketch the path with this number of stages
- ❑ Work either from either end, find sizes:  
$$C_{in} = C_{out} * g/f$$

Reference: Sutherland, Sproull, Harris, “Logical Effort, Morgan-Kaufmann 1999.

# Summary

**Table 4: Key Definitions of Logical Effort**

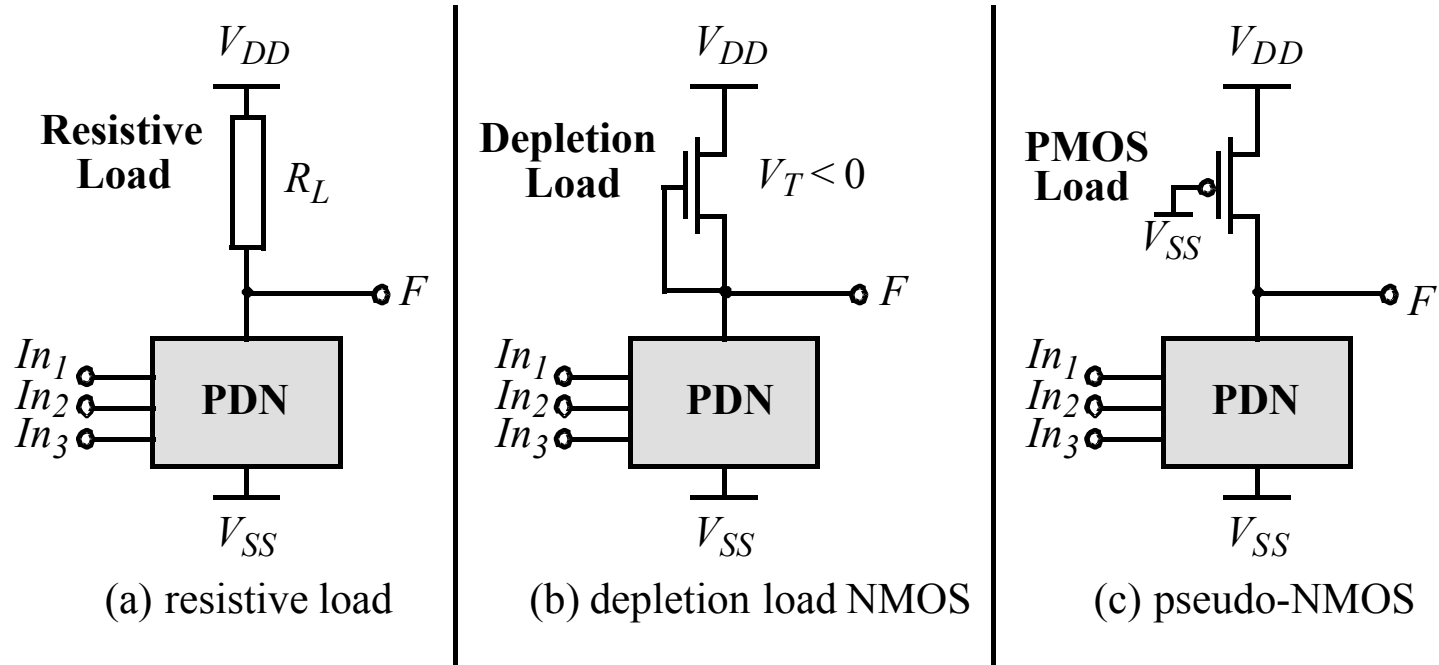
Term	Stage expression	Path expression
Logical effort	$g$ (see Table 1)	$G = \prod g_i$
Electrical effort	$h = \frac{C_{out}}{C_{in}}$	$H = \frac{C_{out (path)}}{C_{in (path)}}$
Branching effort	n/a	$B = \prod b_i$
Effort	$f = gh$	$F = GBH$
Effort delay	$f$	$D_F = \sum f_i$
Number of stages	1	$N$
Parasitic delay	$p$ (see Table 2)	$P = \sum p_i$
Delay	$d = f + p$	$D = D_F + P$

Sutherland,  
Sproull  
Harris



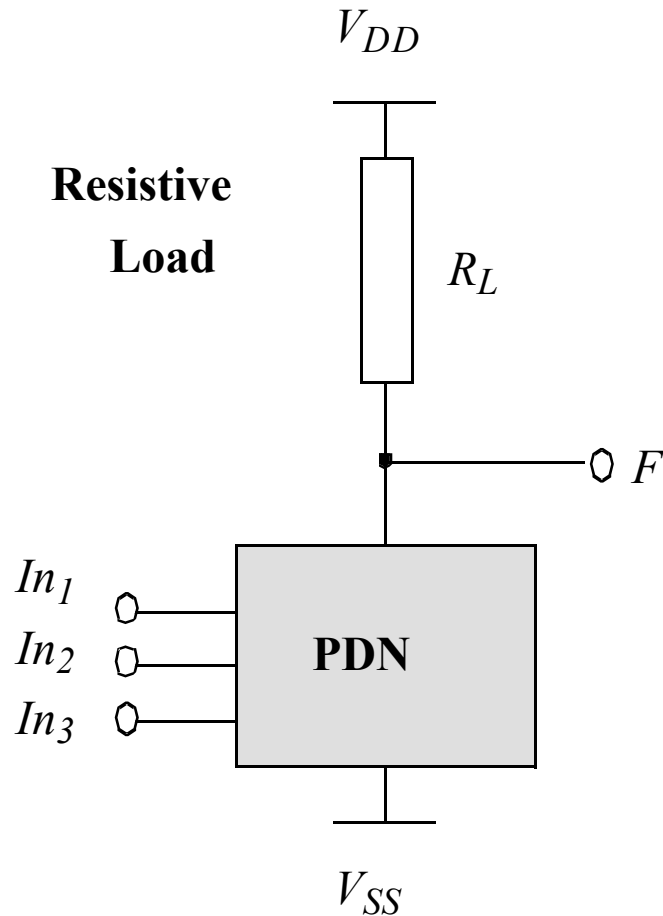
# *Ratioed Logic*

# Ratioed Logic



**Goal: to reduce the number of devices over complementary CMOS**

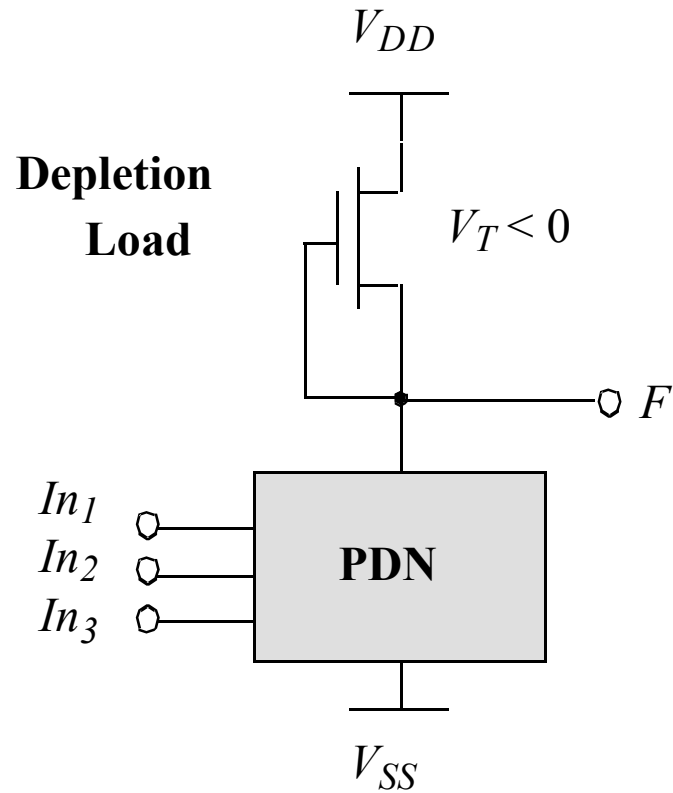
# Ratioed Logic



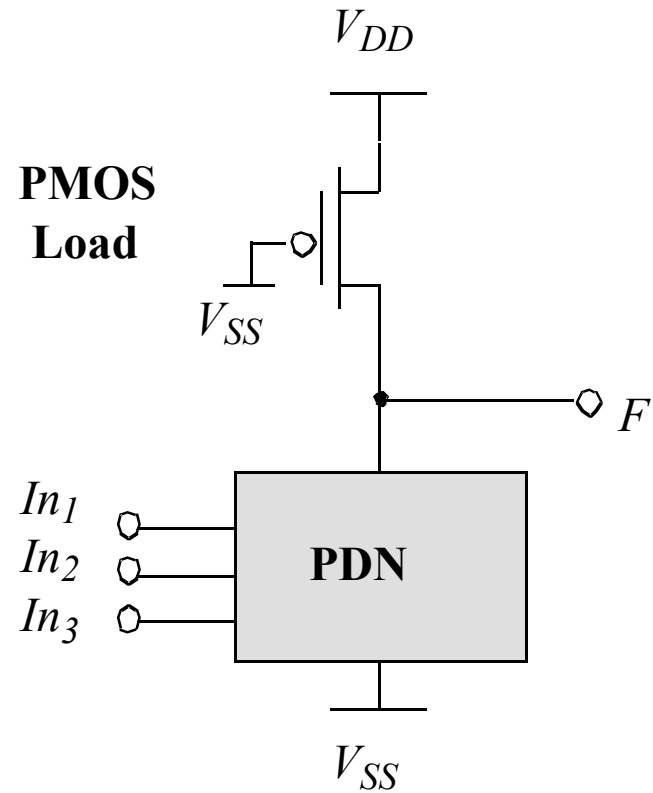
- **N transistors + Load**
- $V_{OH} = V_{DD}$
- $V_{OL} = \frac{R_{PN}}{R_{PN} + R_L}$
- **Assymetrical response**
- **Static power consumption**
- $t_{pL} = 0.69 R_L C_L$



# Active Loads

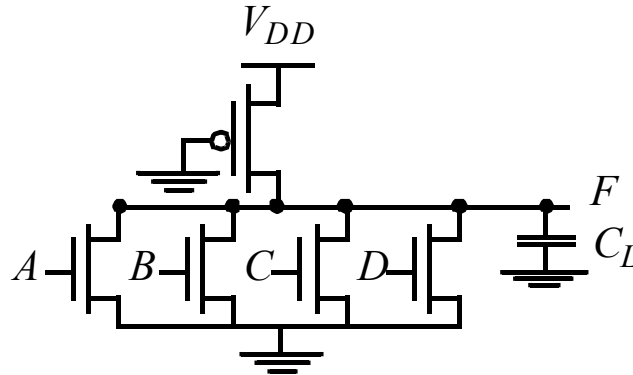


depletion load NMOS



pseudo-NMOS

# Pseudo-NMOS



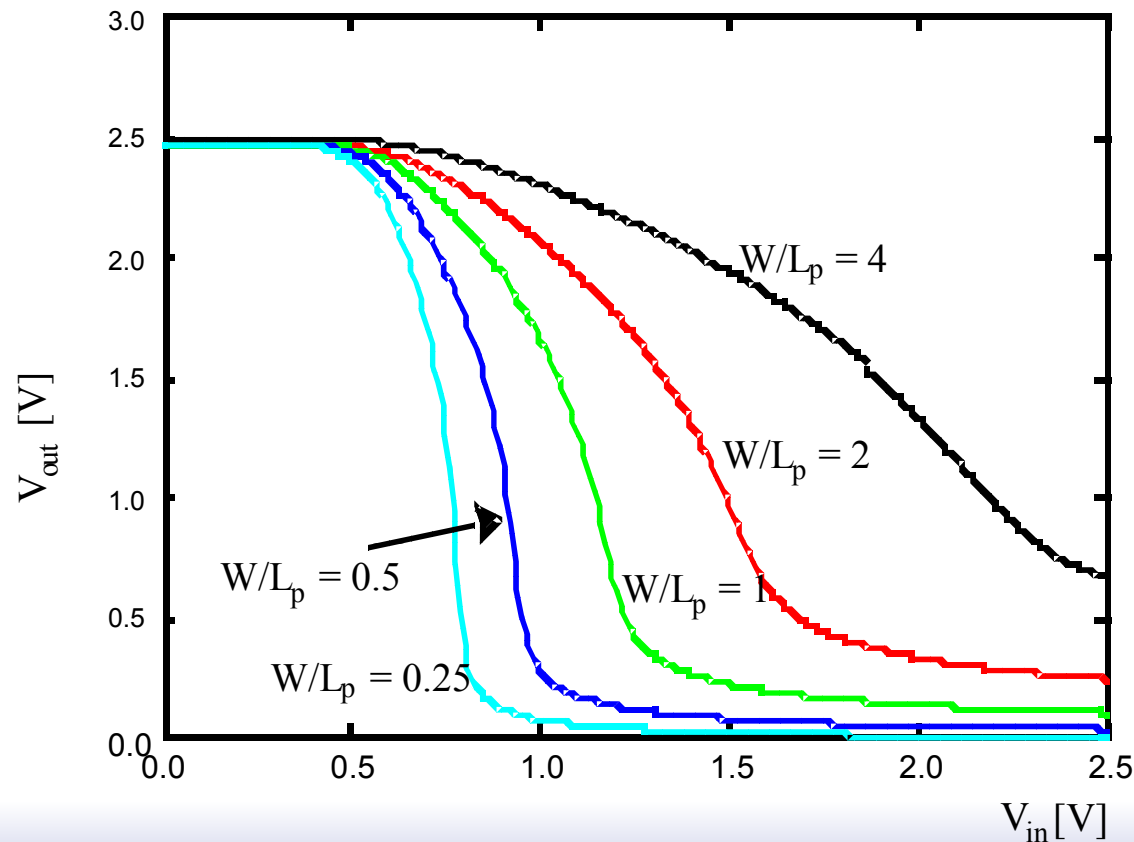
$V_{OH} = V_{DD}$  (similar to complementary CMOS)

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

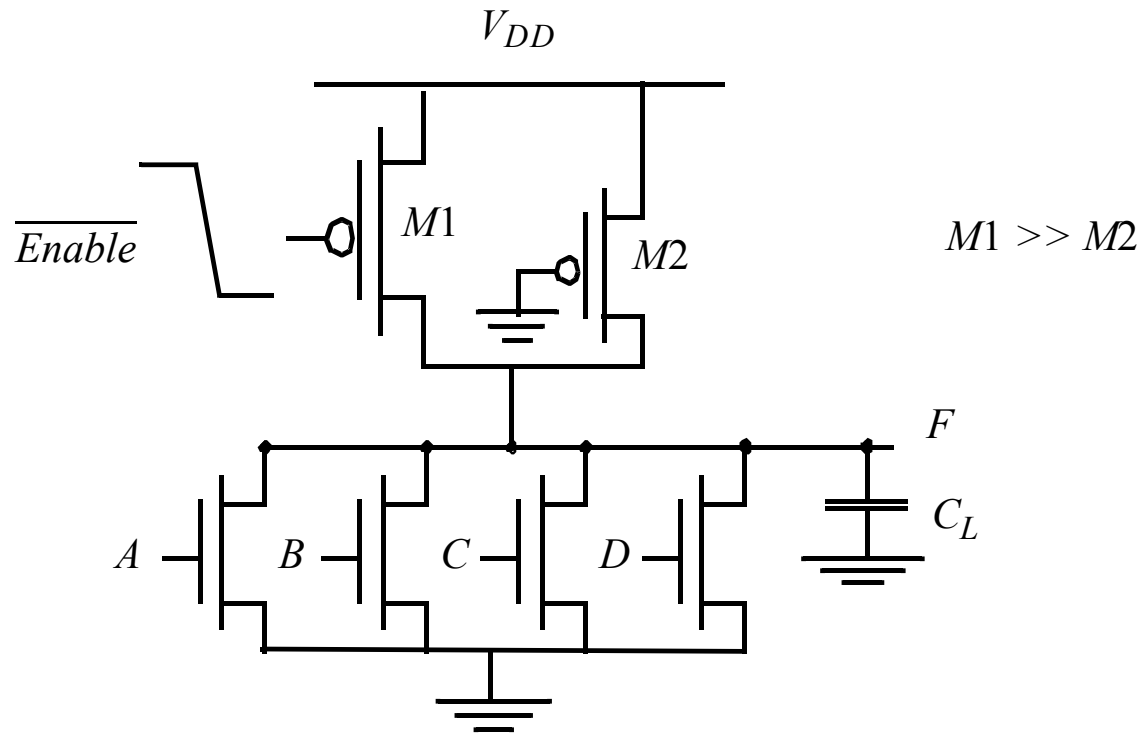
$$V_{OL} = (V_{DD} - V_T) \left[ 1 - \sqrt{1 - \frac{k_p}{k_n}} \right] \text{ (assuming that } V_T = V_{Tn} = |V_{Tp}| \text{)}$$

***SMALLER AREA & LOAD BUT STATIC POWER DISSIPATION!!!***

# Pseudo-NMOS VTC

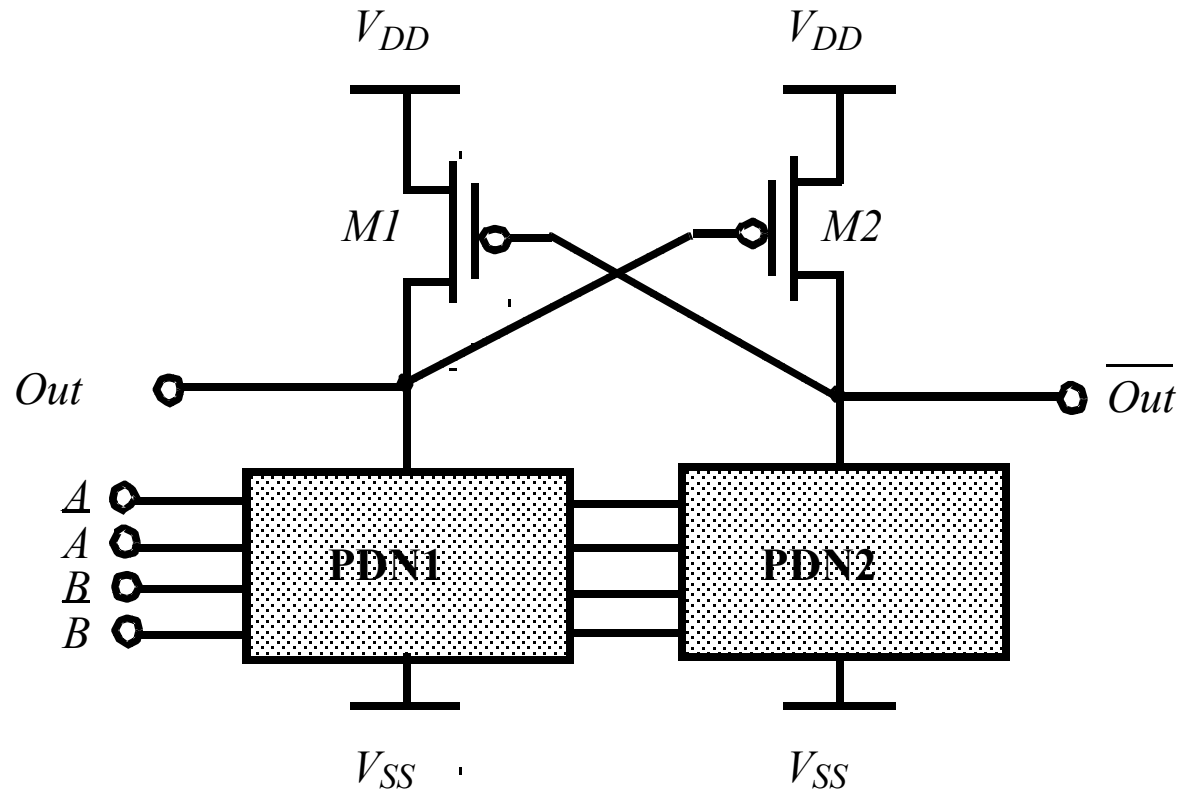


# Improved Loads



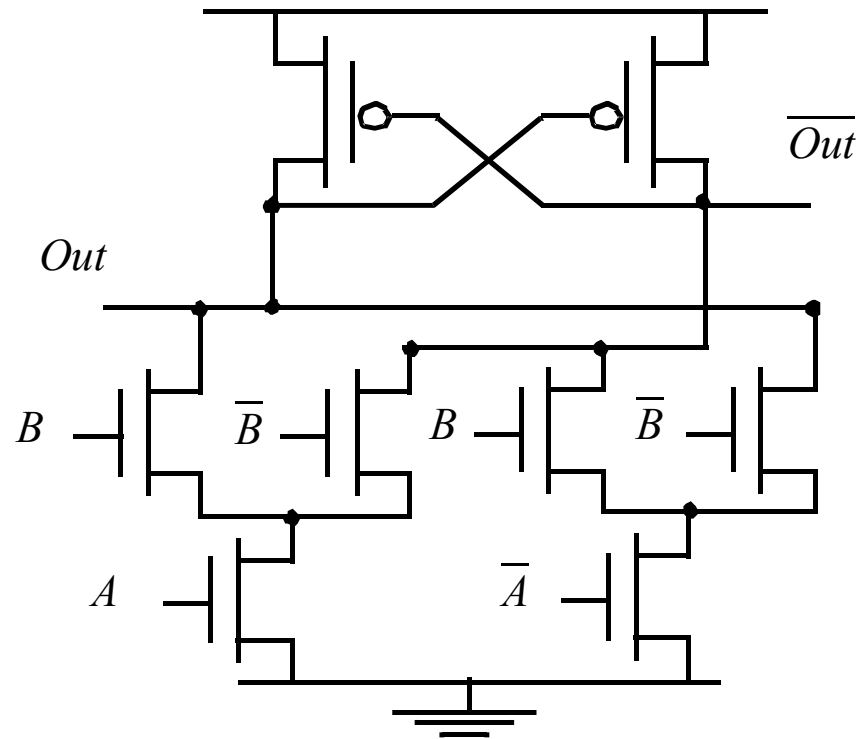
**Adaptive Load**

## Improved Loads (2)



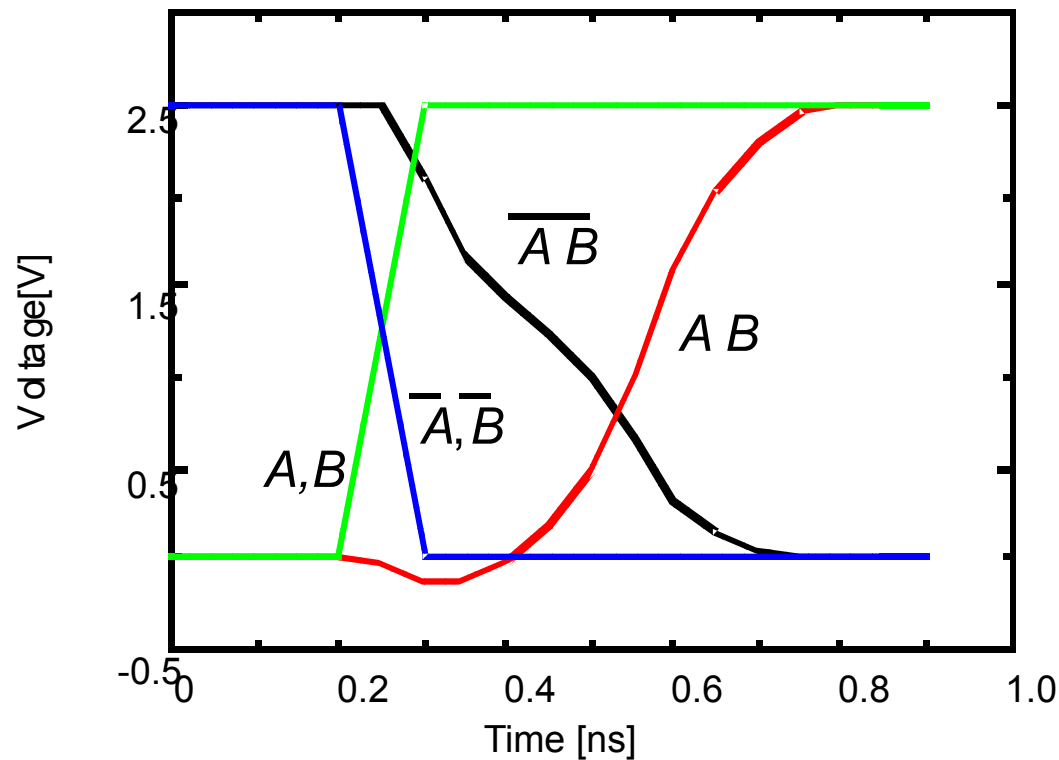
**Differential Cascode Voltage Switch Logic (DCVSL)**

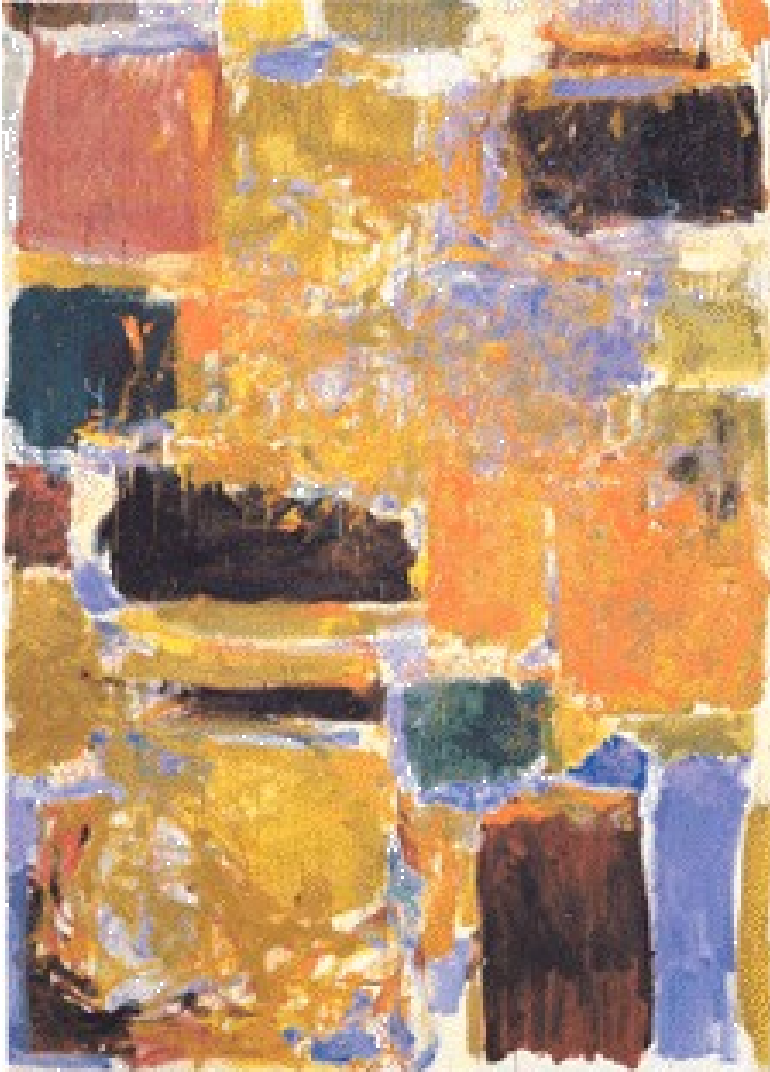
# DCVSL Example



XOR-NXOR gate

# DCVSL Transient Response

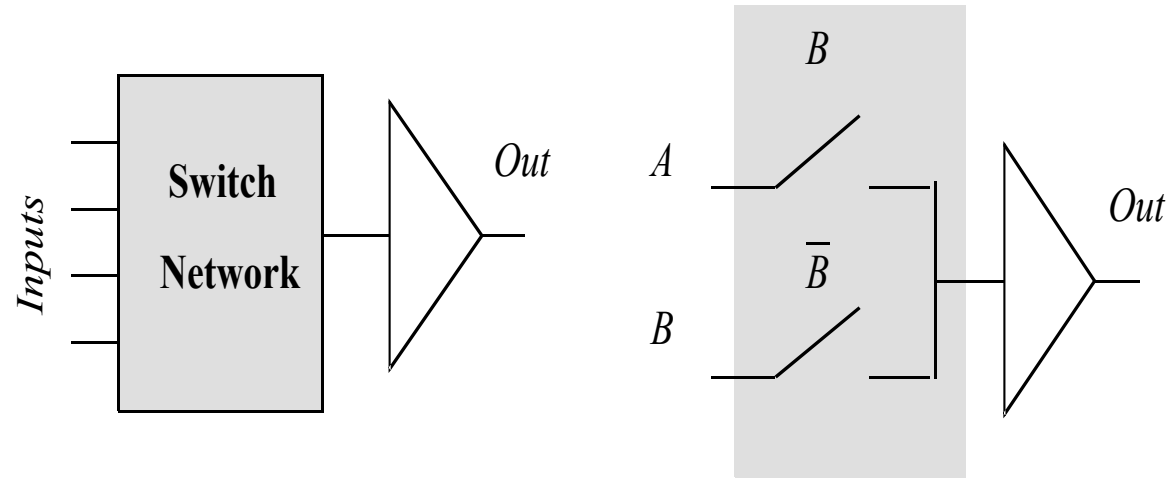




# *Pass-Transistor Logic*

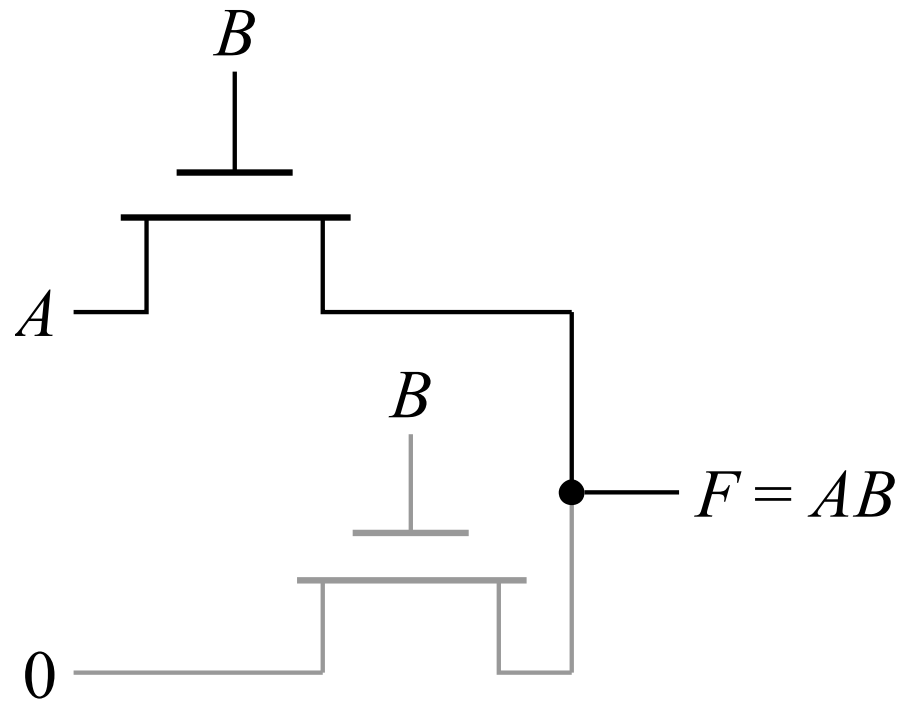


# Pass-Transistor Logic

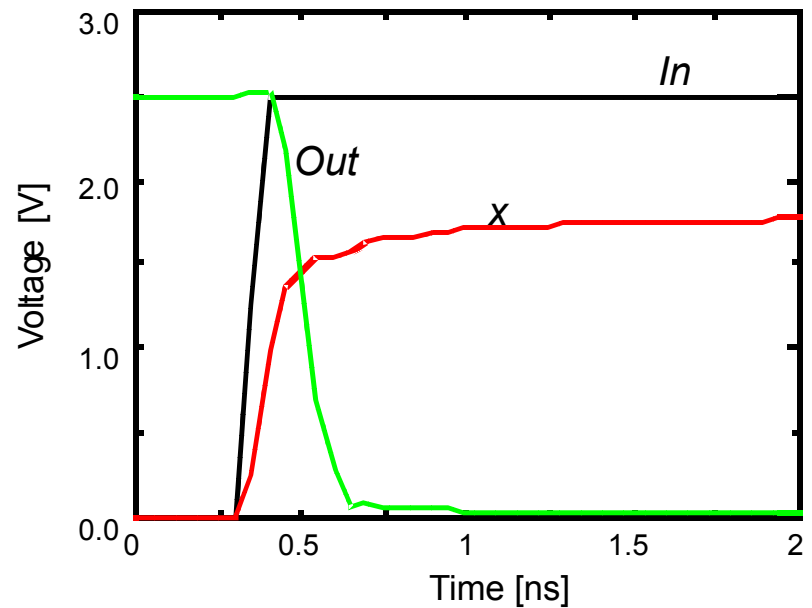
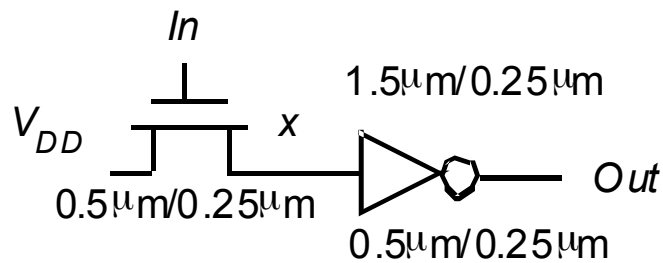


- N transistors
- No static consumption

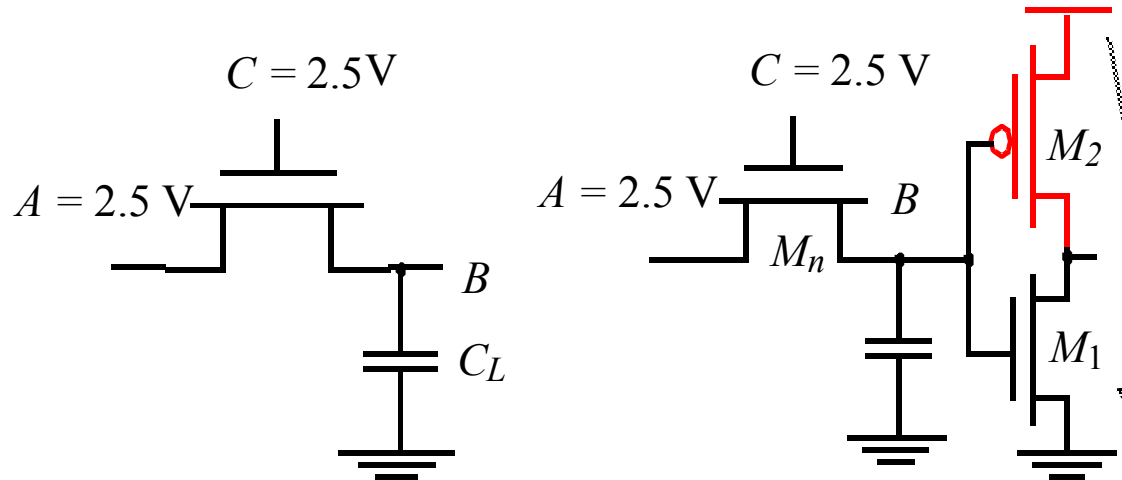
# Example: AND Gate



# NMOS-Only Logic



# NMOS-only Switch

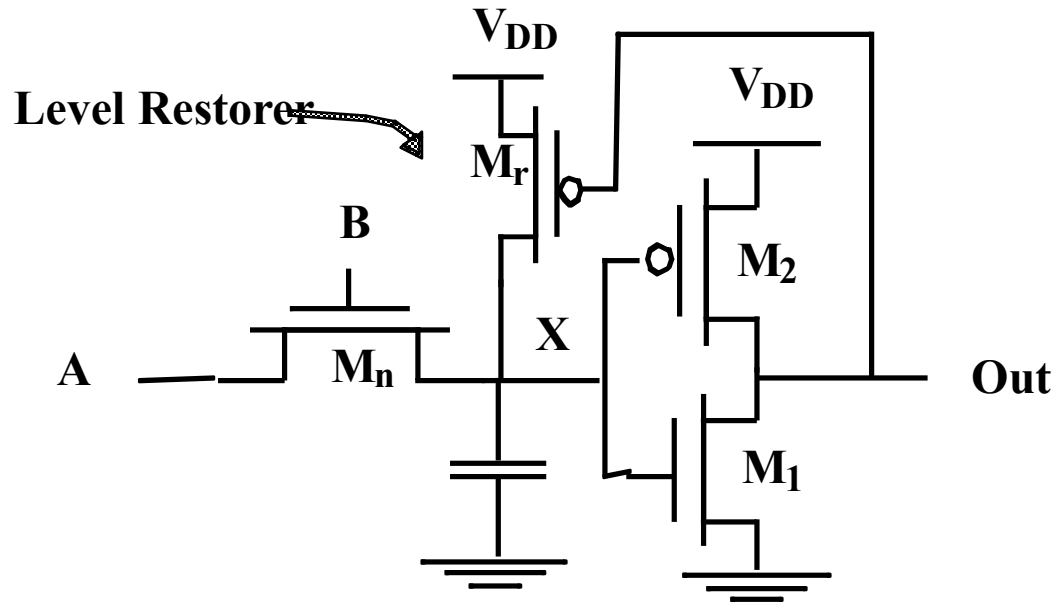


**$V_B$  does not pull up to 2.5V, but  $2.5V - V_{TN}$**

**Threshold voltage loss causes  
static power consumption**

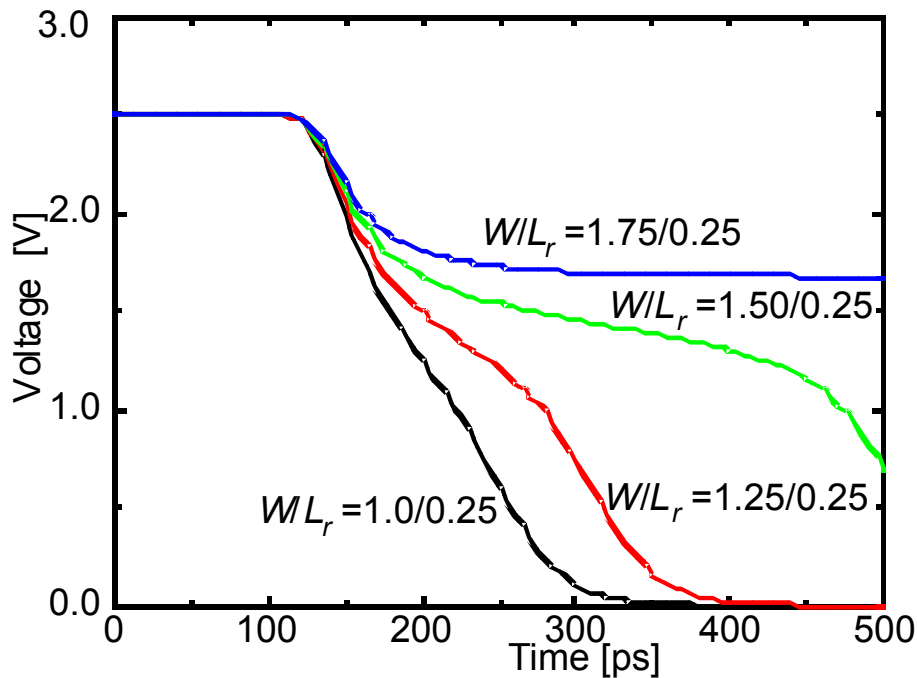
**NMOS has higher threshold than PMOS (body effect)**

# NMOS Only Logic: Level Restoring Transistor



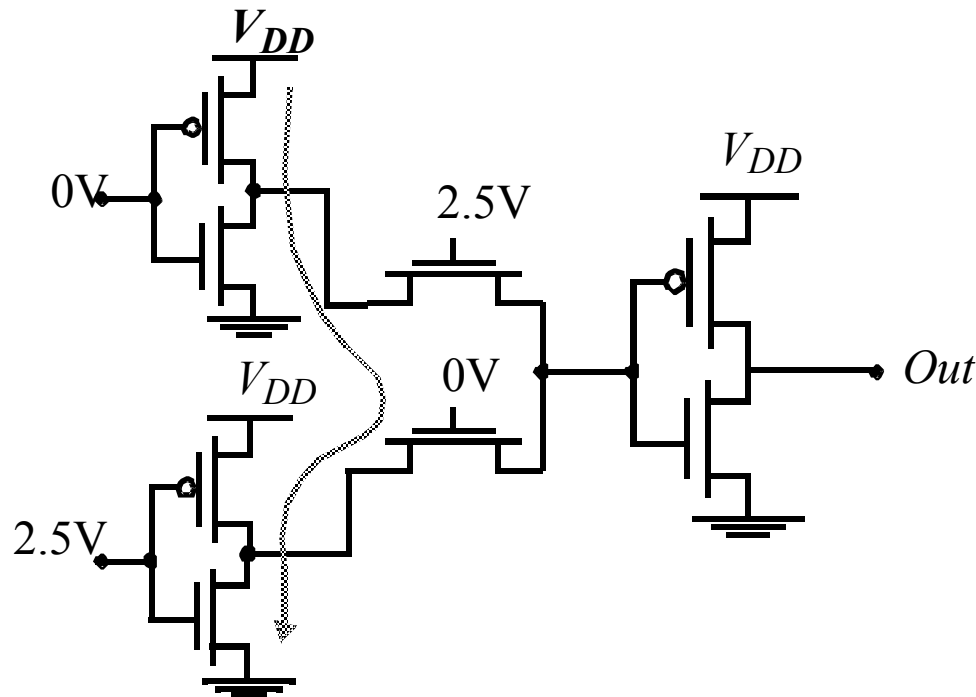
- Advantage: Full Swing
- Restorer adds capacitance, takes away pull down current at X
- Ratio problem

# Restorer Sizing



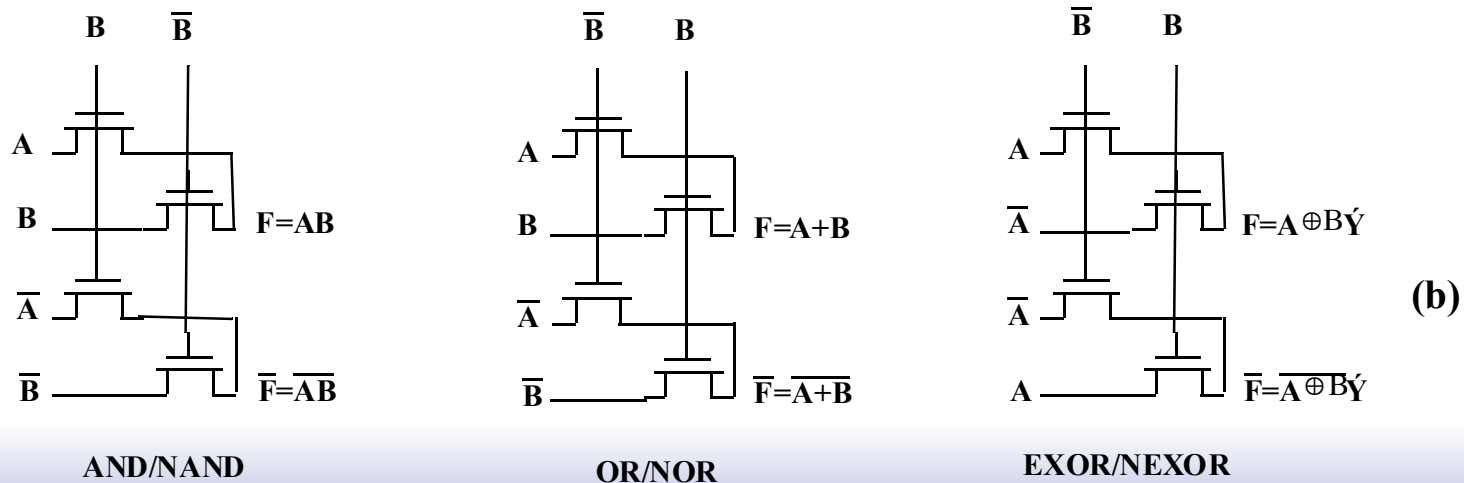
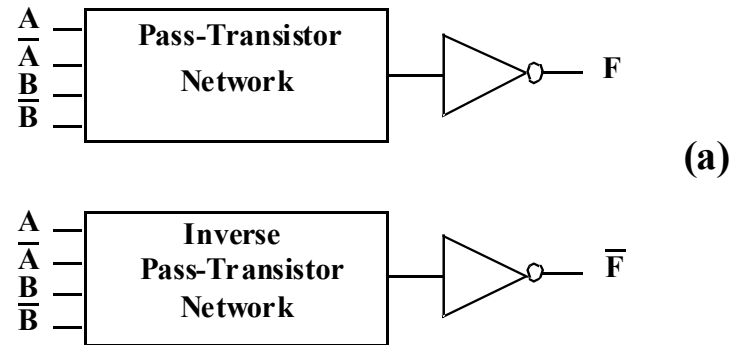
- Upper limit on restorer size
- Pass-transistor pull-down can have several transistors in stack

# Solution 2: Single Transistor Pass Gate with $V_T=0$



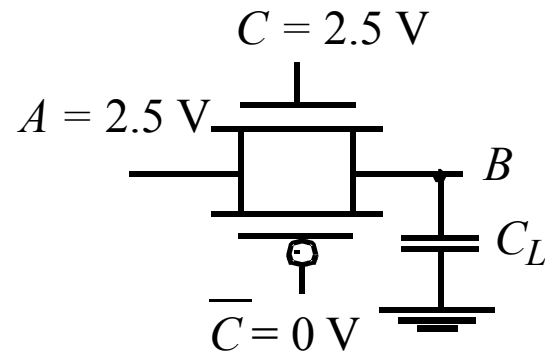
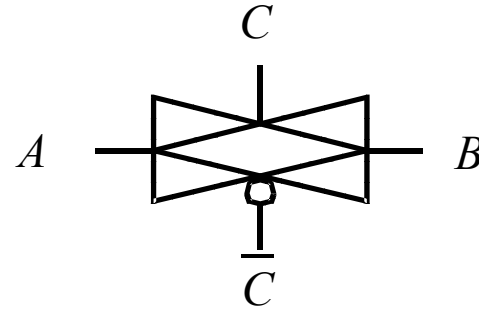
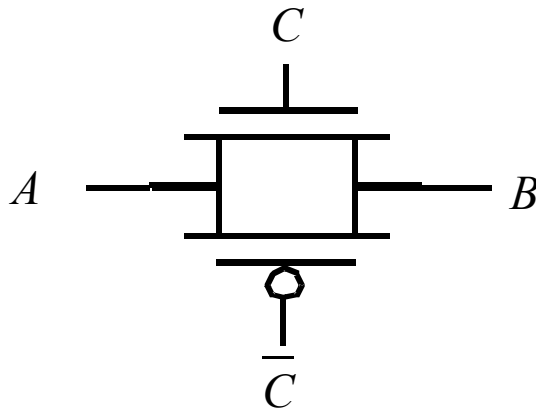
**WATCH OUT FOR LEAKAGE CURRENTS**

# Complementary Pass Transistor Logic

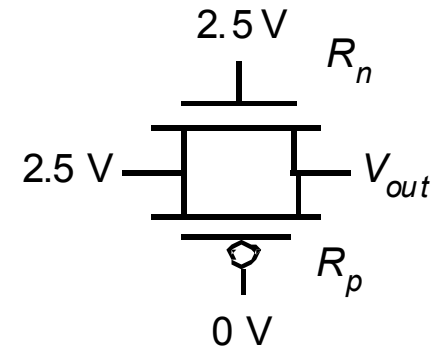
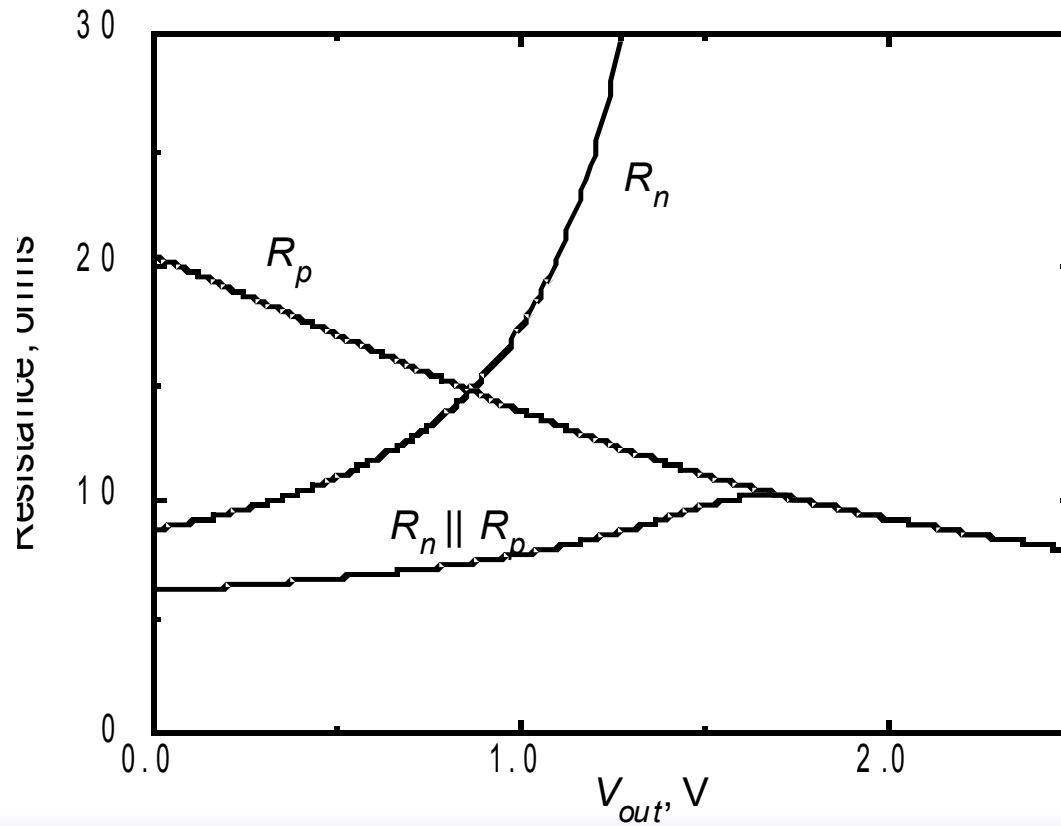




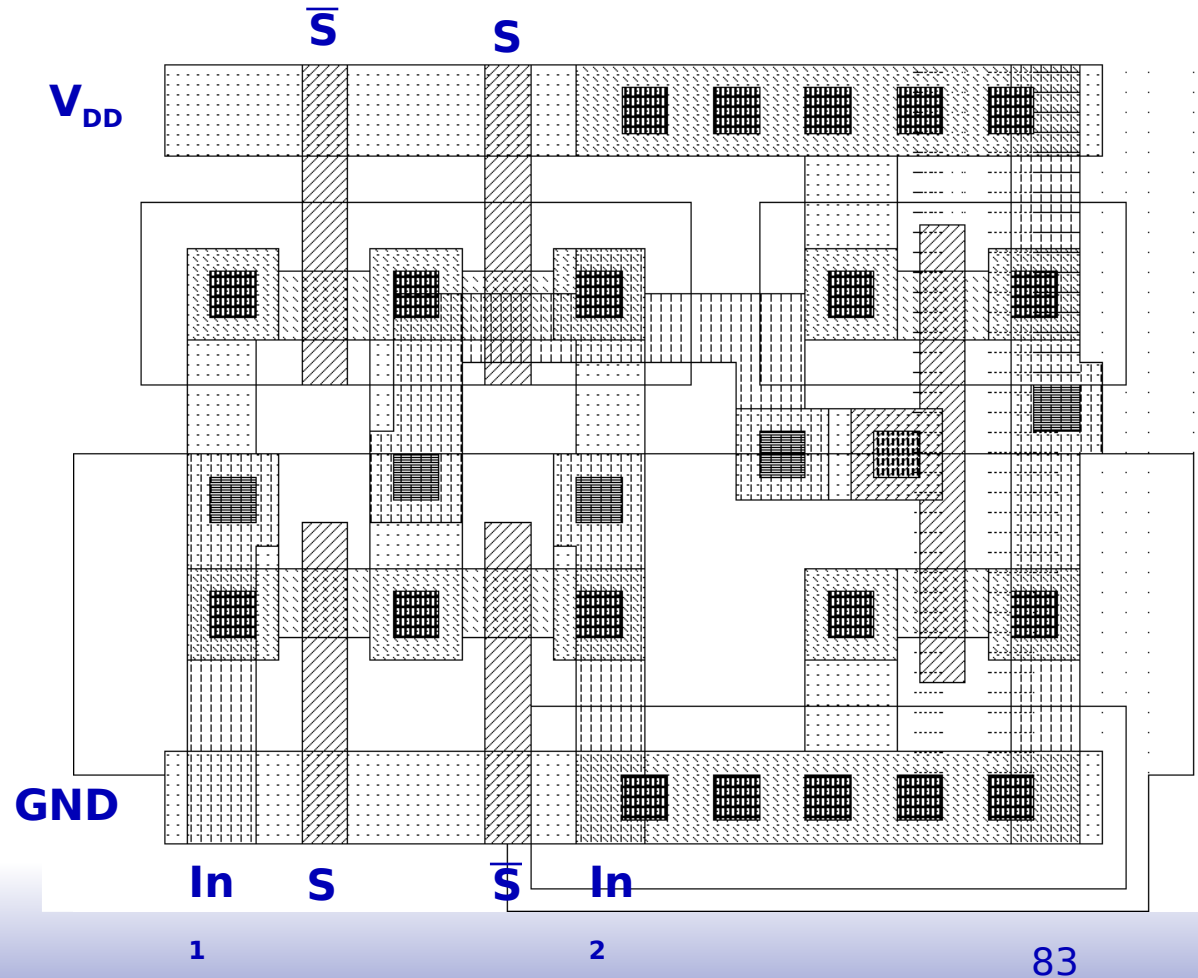
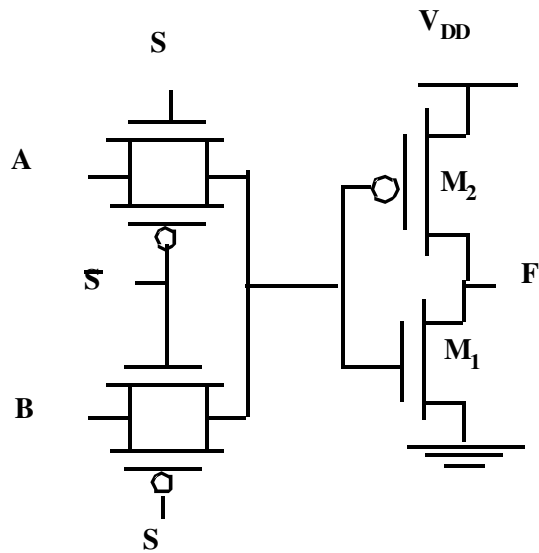
# Solution 3: Transmission Gate



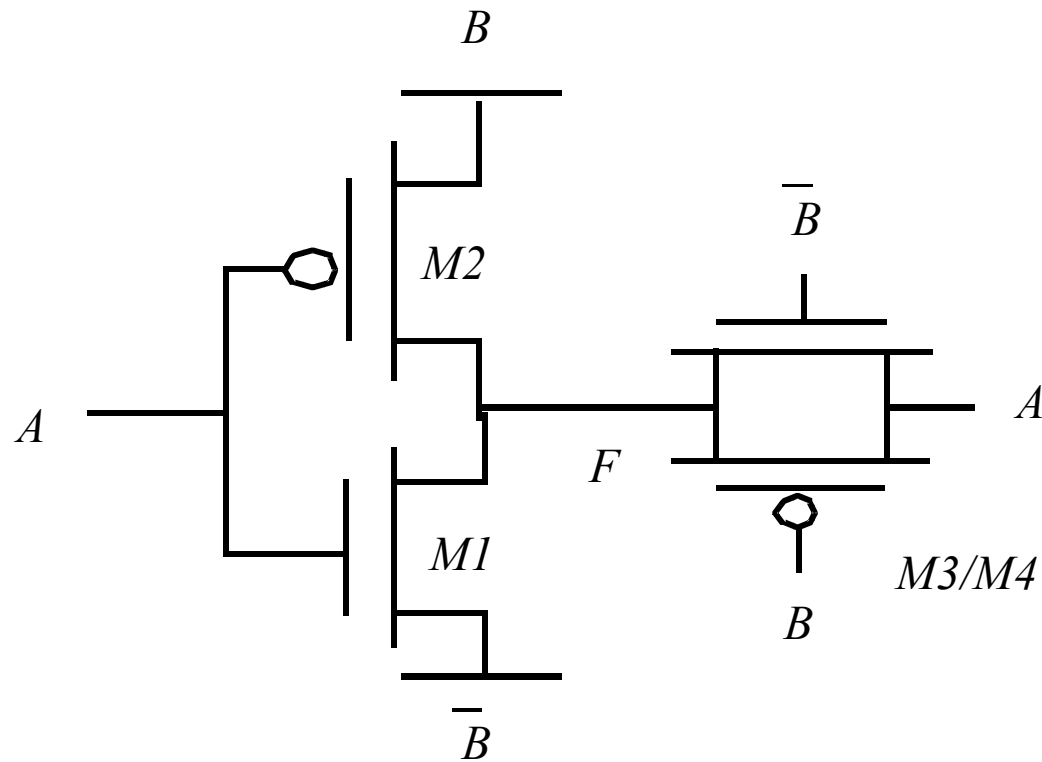
# Resistance of Transmission Gate



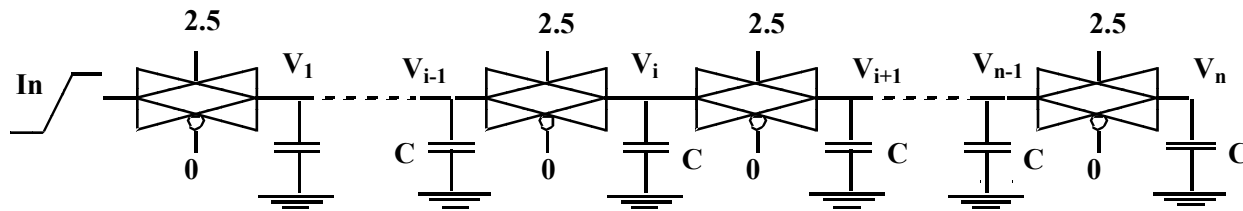
# Pass-Transistor Based Multiplexer



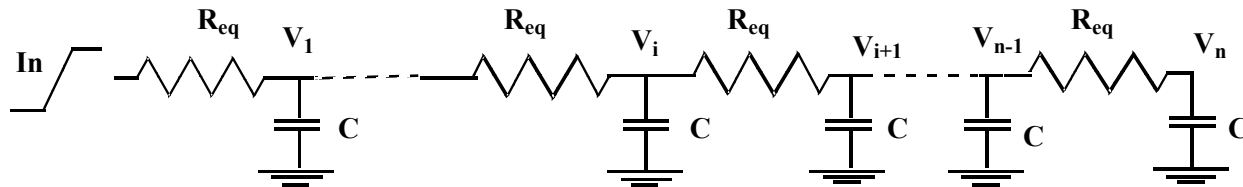
# Transmission Gate XOR



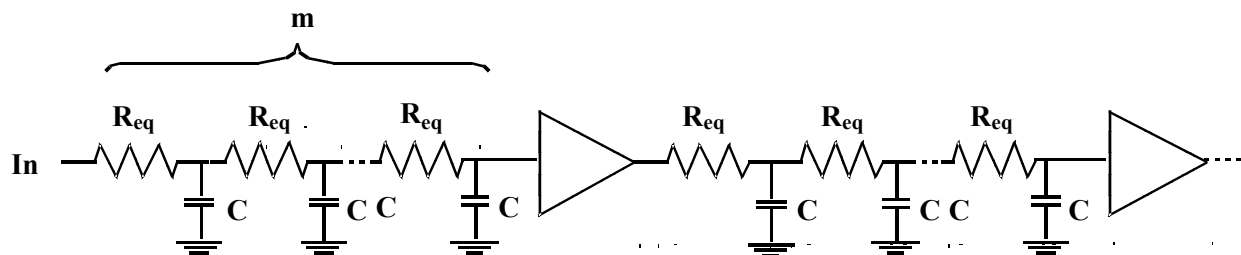
# Delay in Transmission Gate Networks



(a)



(b)



(c)

# Delay Optimization

- Delay of RC chain

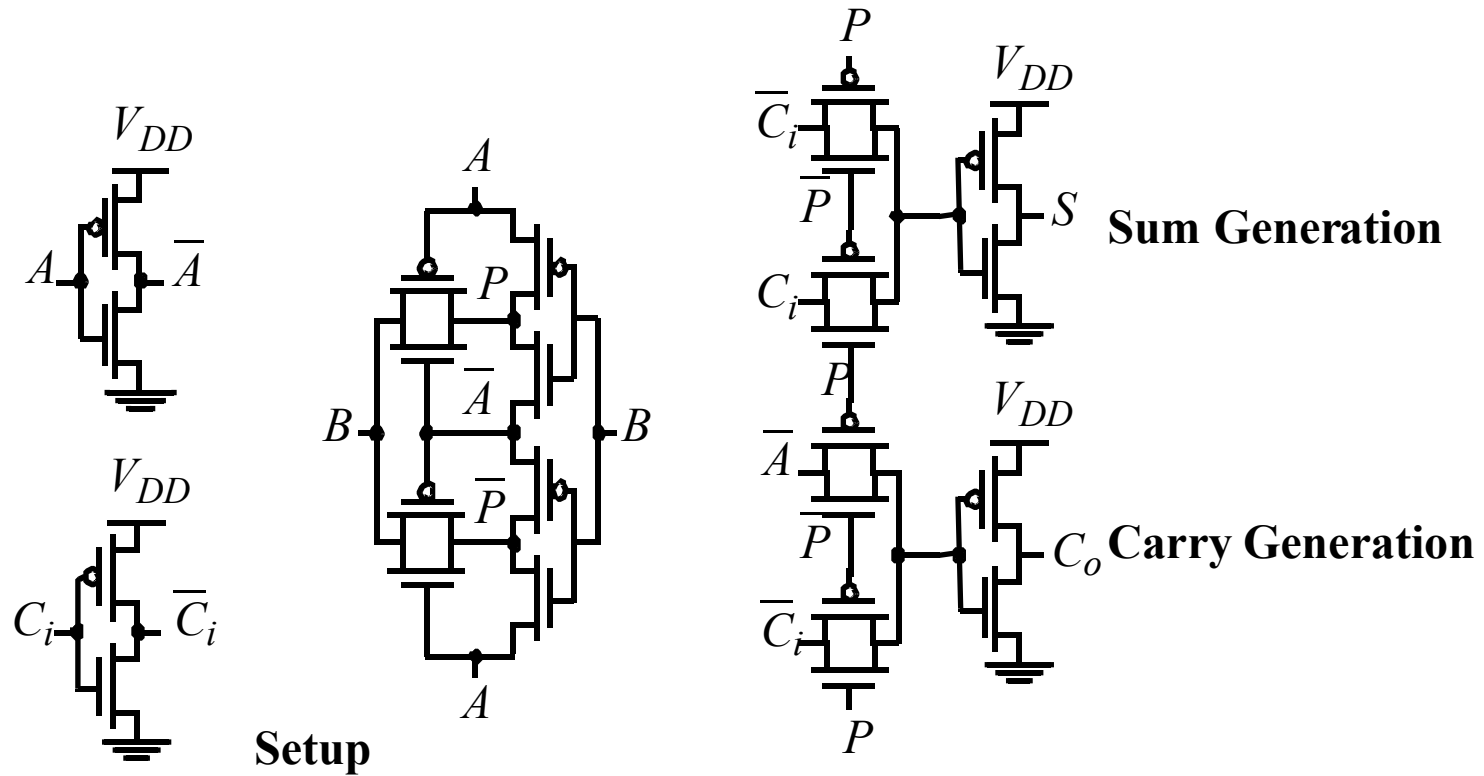
$$t_p = 0.69 \sum_{k=0}^n CR_{eq}^k = 0.69 CR_{eq} \frac{n(n+1)}{2}$$

- Delay of Buffered Chain

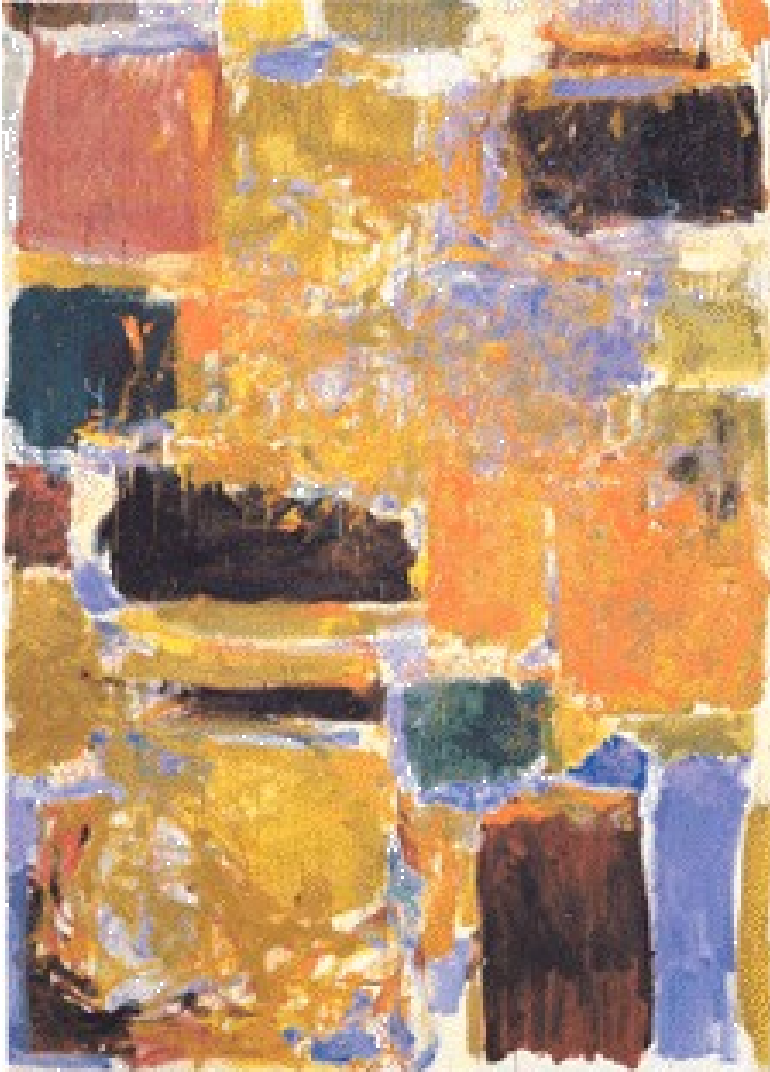
$$\begin{aligned} t_p &= 0.69 \left[ \frac{n}{m} CR_{eq} \frac{m(m+1)}{2} \right] + \left( \frac{n}{m} - 1 \right) t_{buf} \\ &= 0.69 \left[ CR_{eq} \frac{n(m+1)}{2} \right] + \left( \frac{n}{m} - 1 \right) t_{buf} \end{aligned}$$

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

# Transmission Gate Full Adder



*Similar delays for sum and carry*



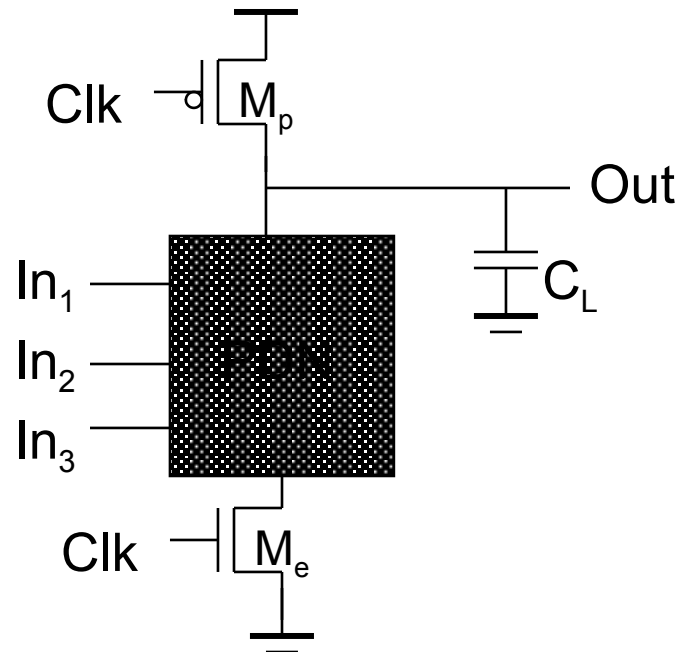
# *Dynamic Logic*



# Dynamic CMOS

- ❑ In **static** circuits at every point in time (except when switching) the output is connected to either GND or  $V_{DD}$  via a low resistance path.
  - fan-in of  $n$  requires  $2n$  ( $n$  N-type +  $n$  P-type) devices
  
- ❑ **Dynamic** circuits rely on the temporary storage of signal values on the capacitance of high impedance nodes.
  - requires on  $n + 2$  ( $n+1$  N-type + 1 P-type) transistors

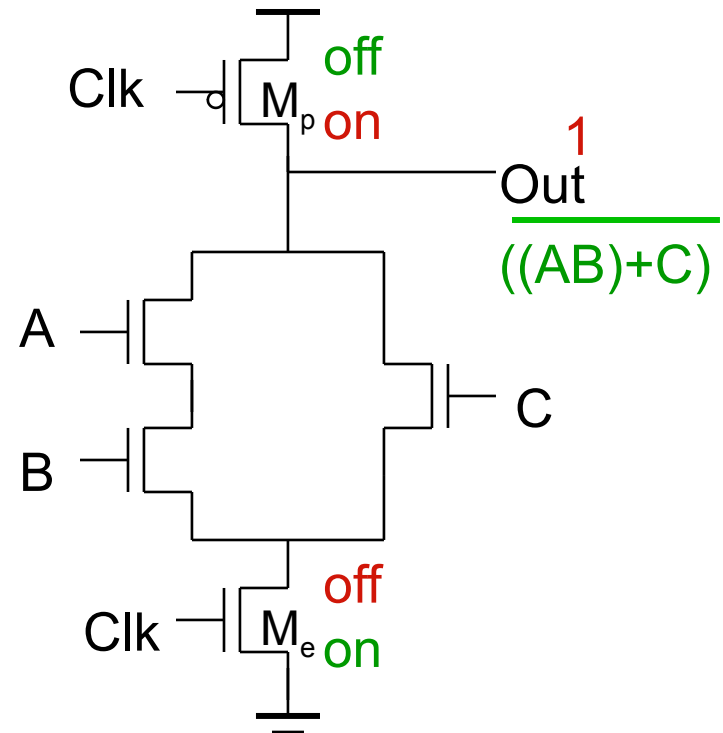
# Dynamic Gate



Two phase operation

**Precharge** (Clk = 0)

**Evaluate** (Clk = 1)



# Conditions on Output

- ❑ Once the output of a dynamic gate is discharged, 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  $C_L$

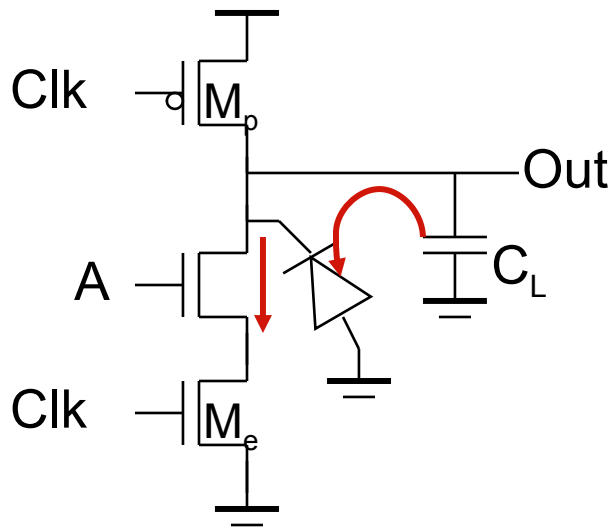
# Properties of Dynamic Gates

- ❑ Logic function is implemented by the PDN only
  - number of transistors is  $N + 2$  (versus  $2N$  for static complementary CMOS)
- ❑ Full swing outputs ( $V_{OL} = \text{GND}$  and  $V_{OH} = V_{DD}$ )
- ❑ Non-ratioed - sizing of the devices does not affect the logic levels
- ❑ Faster switching speeds
  - reduced load capacitance due to **lower input** capacitance ( $C_{in}$ )
  - reduced load capacitance due to smaller output loading ( $C_{out}$ )
  - no  $I_{sc}$ , so all the current provided by PDN goes into discharging  $C_L$

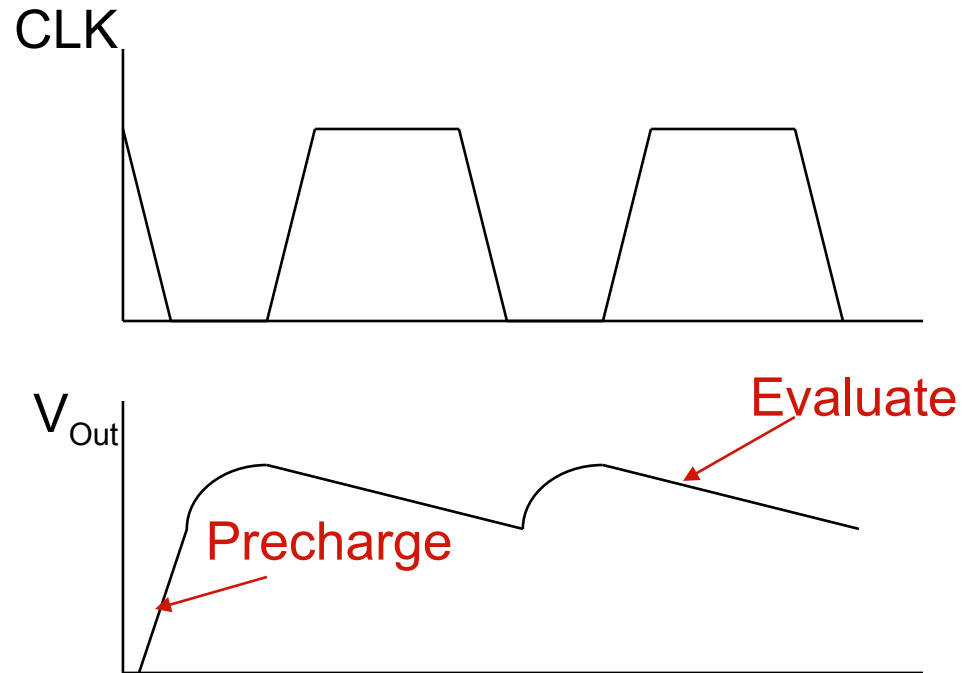
# Properties of Dynamic Gates

- ❑ Overall power dissipation usually **higher** than static CMOS
  - no static current path ever exists between  $V_{DD}$  and GND (including  $P_{sc}$ )
  - no glitching
  - **higher transition probabilities**
  - **extra load on Clk**
- ❑ PDN starts to work as soon as the input signals exceed  $V_{Tn}$ , so  $V_M$ ,  $V_{IH}$  and  $V_{IL}$  equal to  $V_{Tn}$ 
  - low noise margin ( $NM_L$ )
- ❑ Needs a precharge/evaluate clock

# Issues in Dynamic Design 1: Charge Leakage

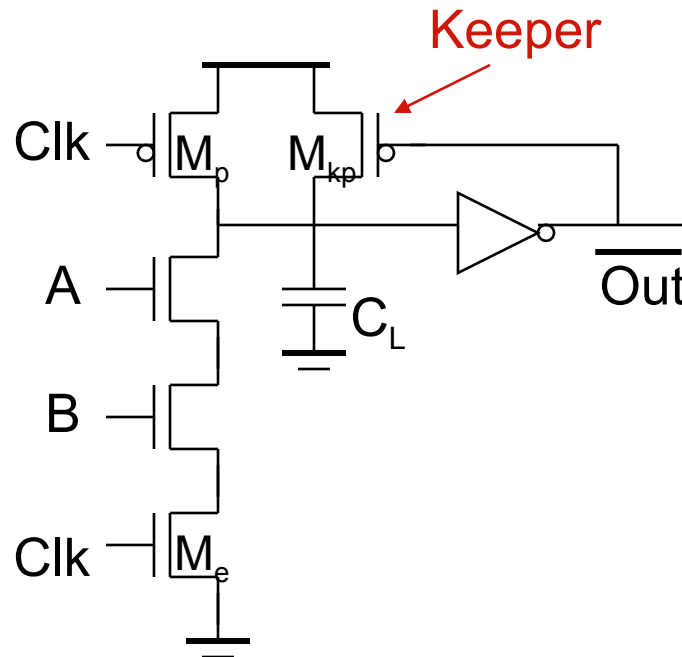


Leakage sources



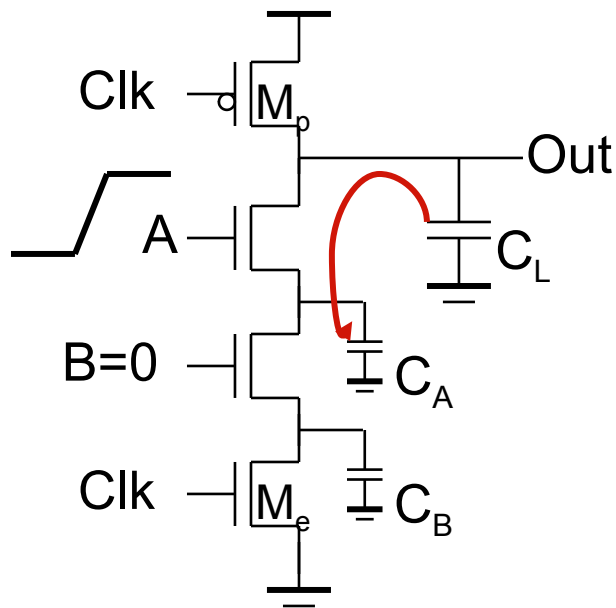
*Dominant component is subthreshold current*

# *Solution to Charge Leakage*



Same approach as level restorer for pass-transistor logic

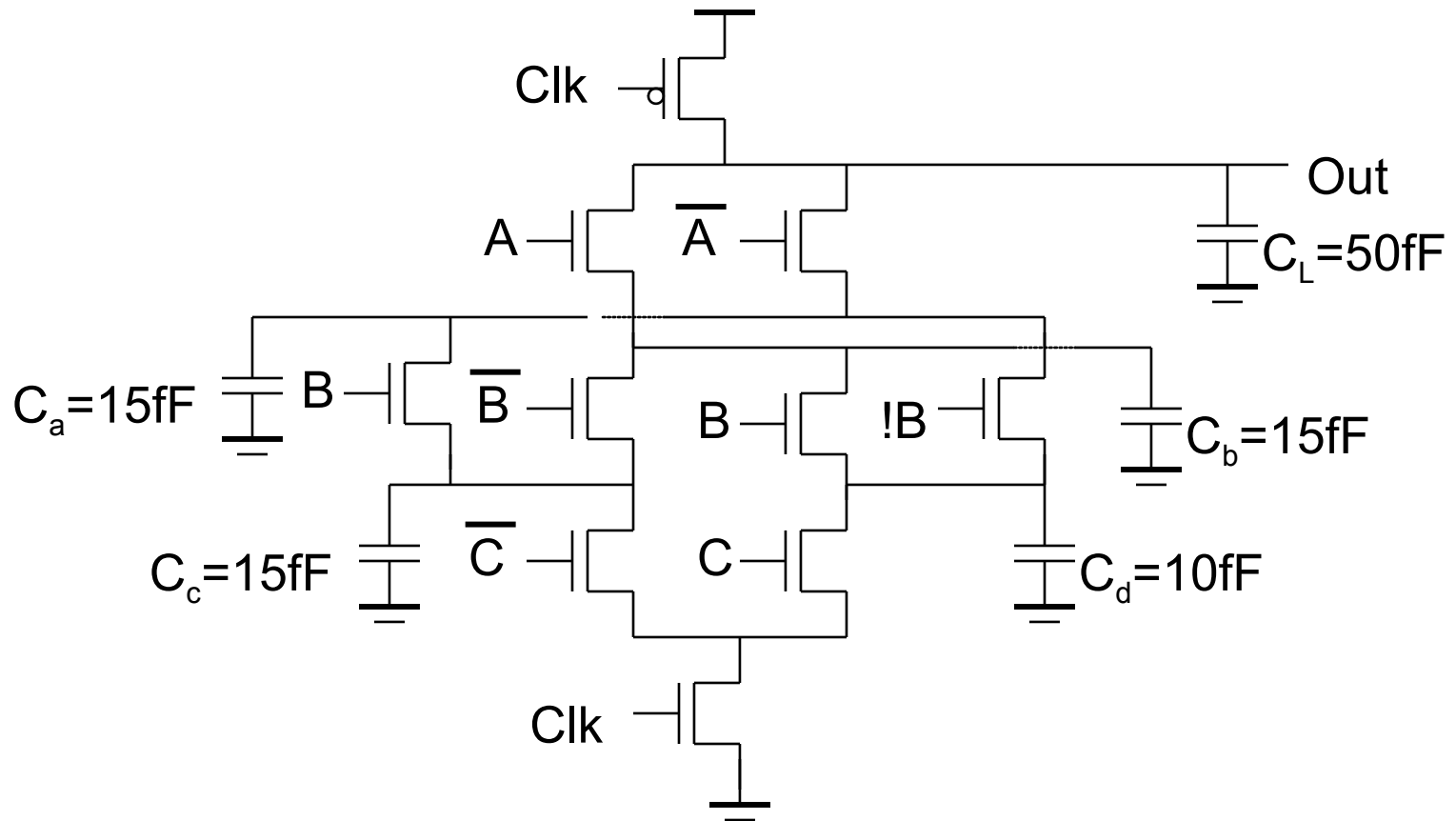
# Issues in Dynamic Design 2: Charge Sharing



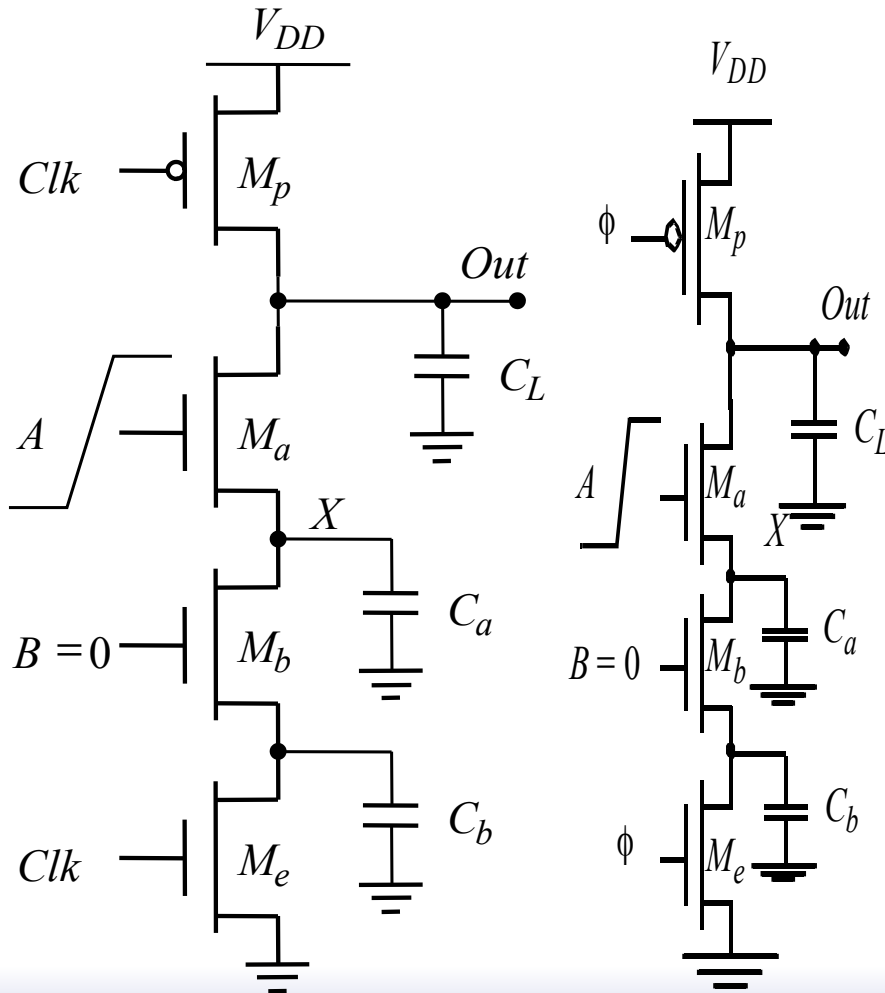
Charge stored originally on  $C_L$  is redistributed (shared) over  $C_L$  and  $C_A$  leading to reduced robustness



# Charge Sharing Example



# Charge Sharing



**case 1) if  $\Delta V_{out} < V_{Tn}$**

$$C_L V_{DD} = C_L V_{out}(t) + C_a (V_{DD} - V_{Tn}(V_X))$$

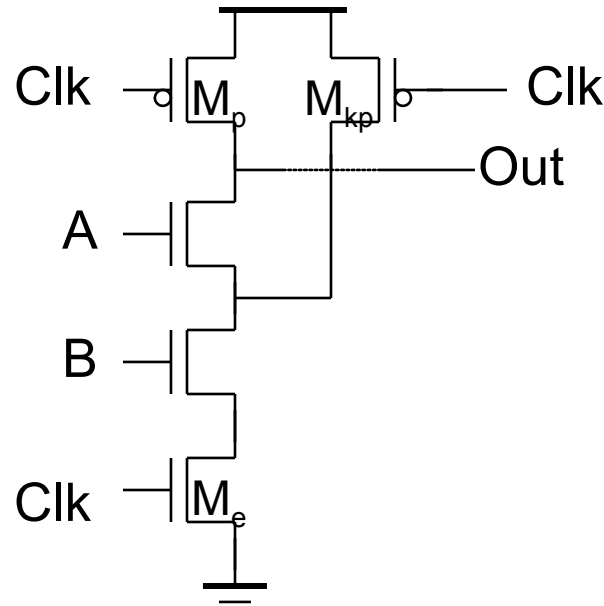
or

$$\Delta V_{\text{out}} = V_{\text{out}}(t) - V_{\text{DD}} = -\frac{C_a}{C_L}(V_{\text{DD}} - V_{\text{Tn}}(V_X))$$

**case 2) if  $\Delta V_{out} > V_{Tn}$**

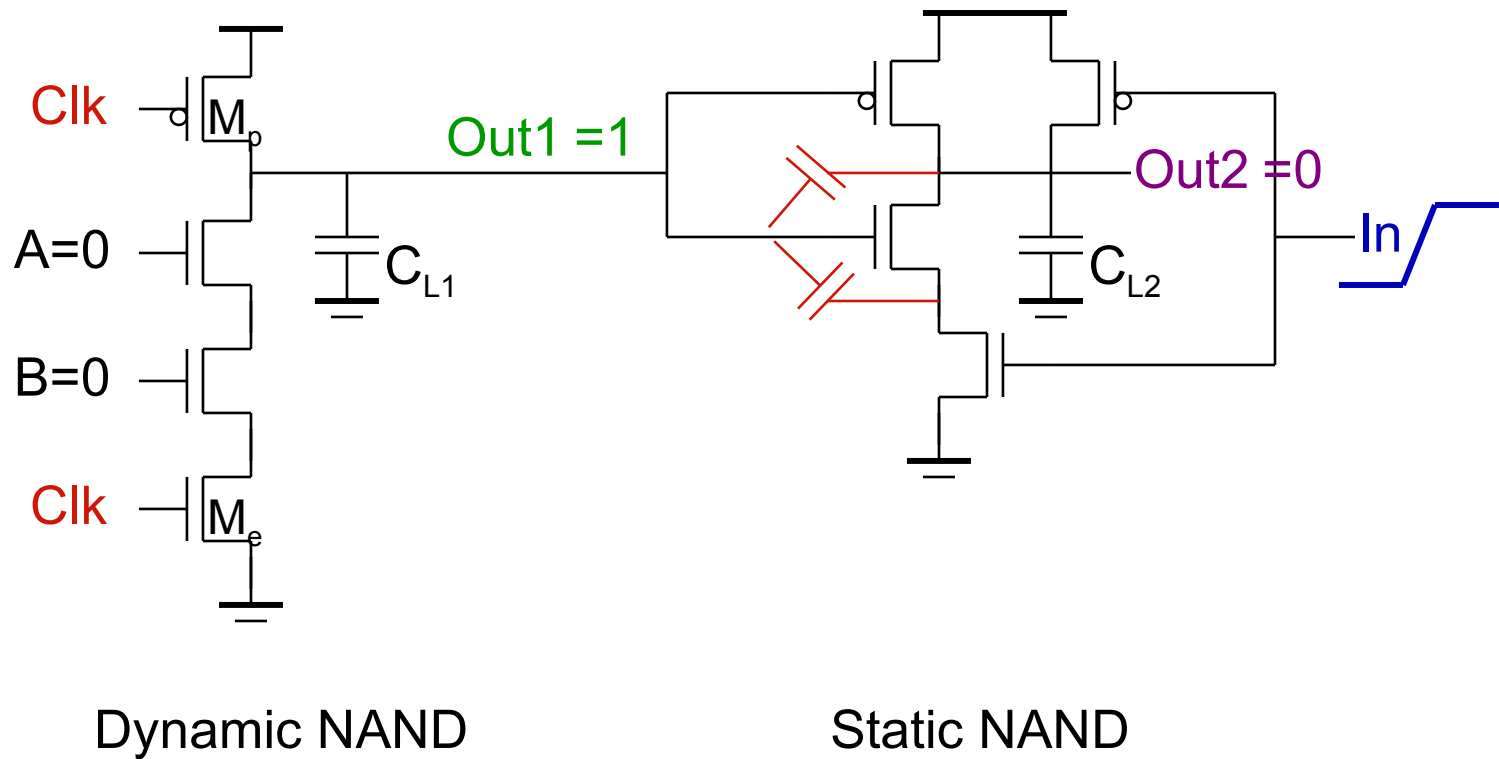
$$\Delta V_{\text{out}} = -V_{\text{DD}} \left( \frac{C_a}{C_a + C_L} \right)$$

# *Solution to Charge Redistribution*

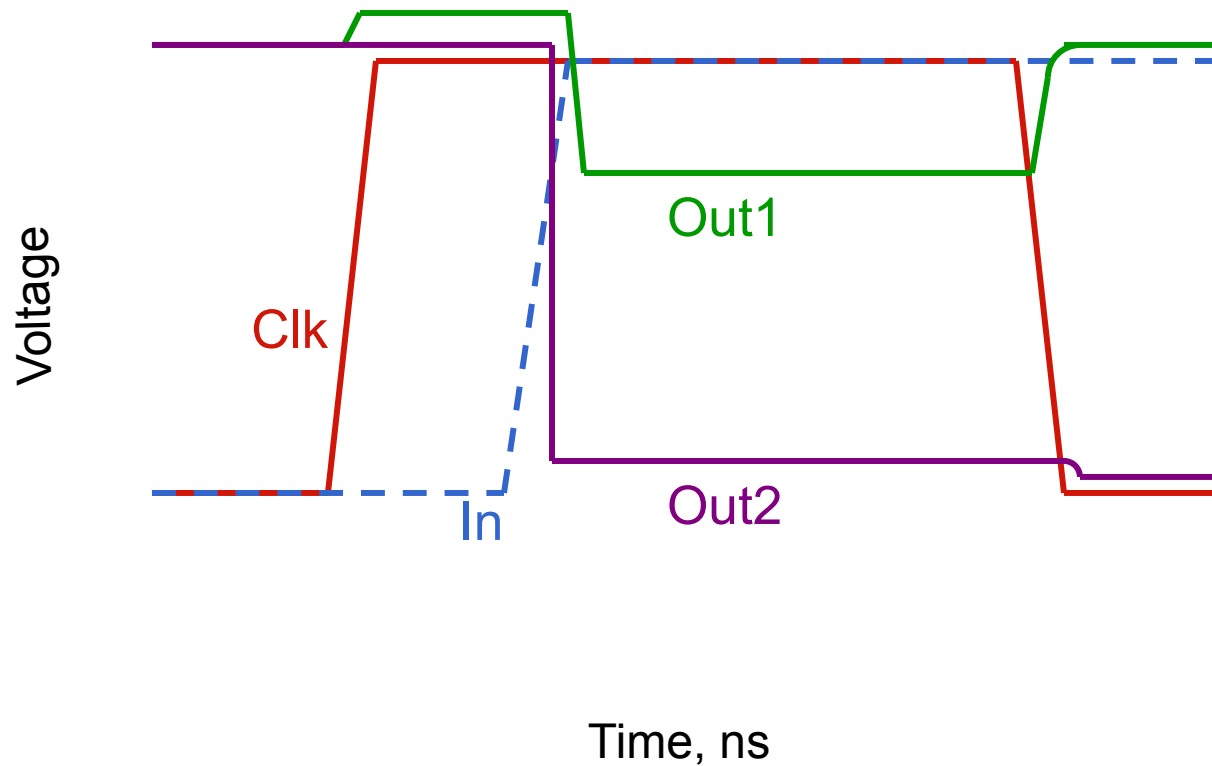


Precharge internal nodes using a clock-driven transistor  
(at the cost of increased area and power)

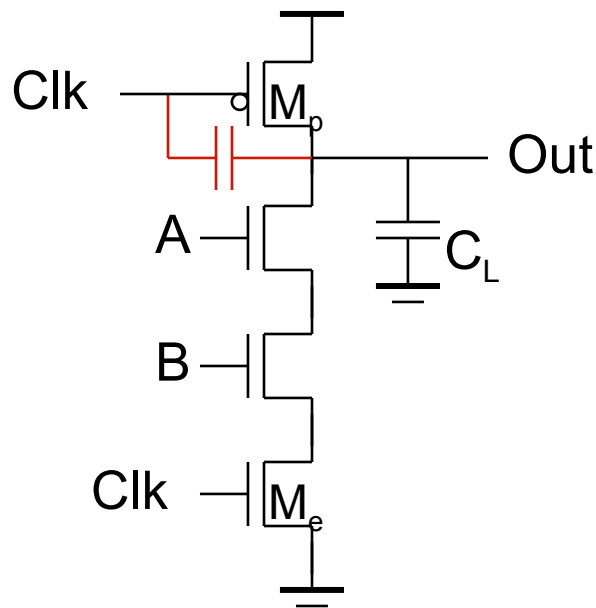
# Issues in Dynamic Design 3: Backgate Coupling



# Backgate Coupling Effect

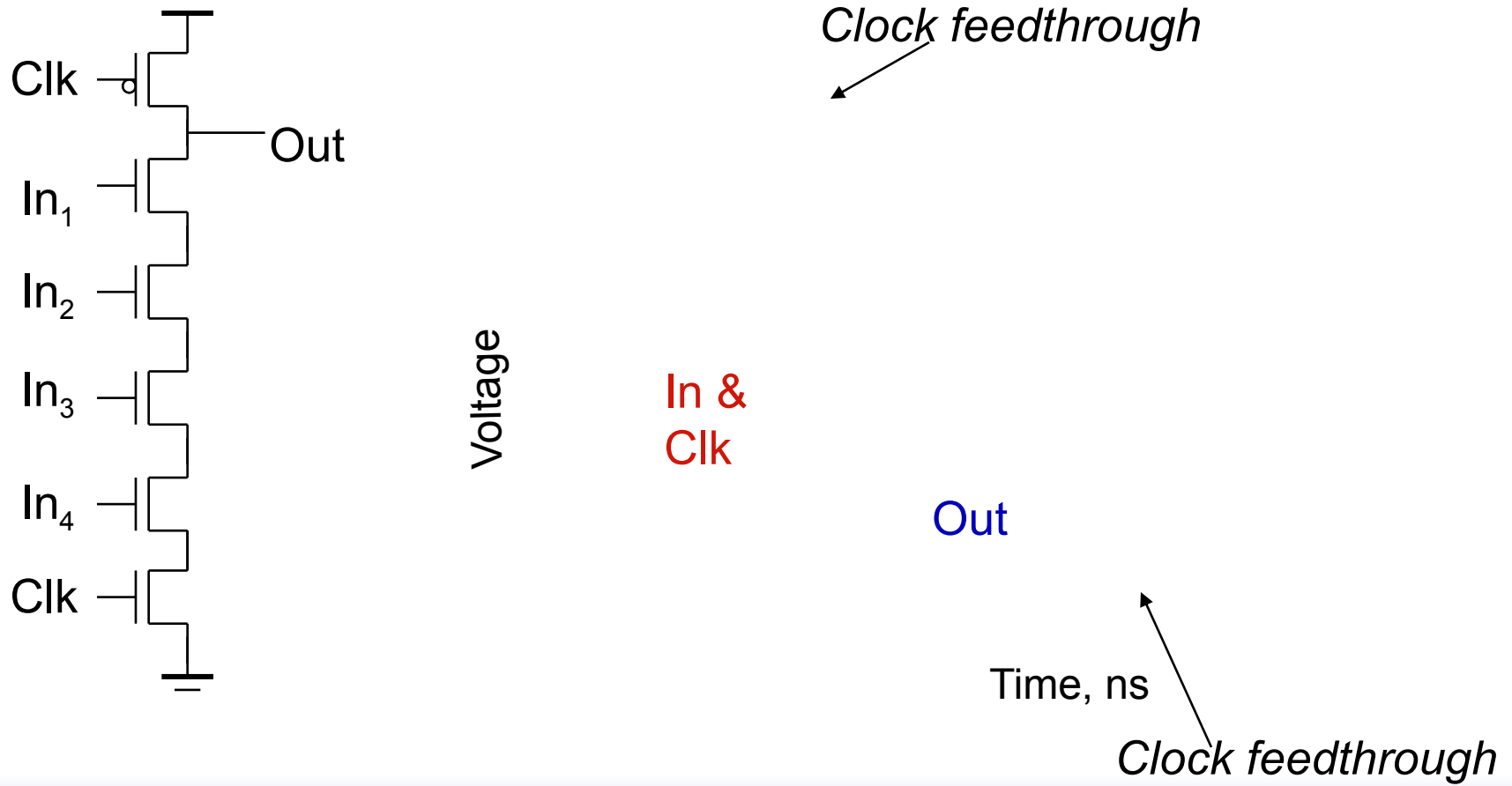


# Issues in Dynamic Design 4: Clock Feedthrough



Coupling between Out and Clk input of the precharge device due to the gate to drain capacitance. So voltage of Out can rise above  $V_{DD}$ . The fast rising (and falling edges) of the clock **couple** to Out.

# Clock Feedthrough

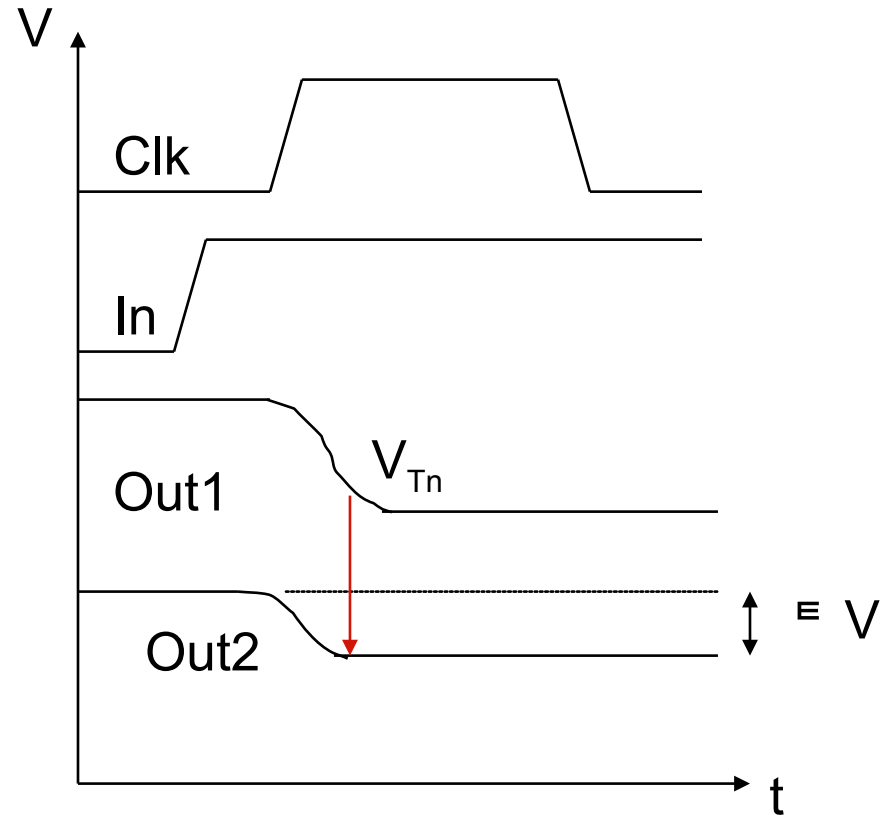
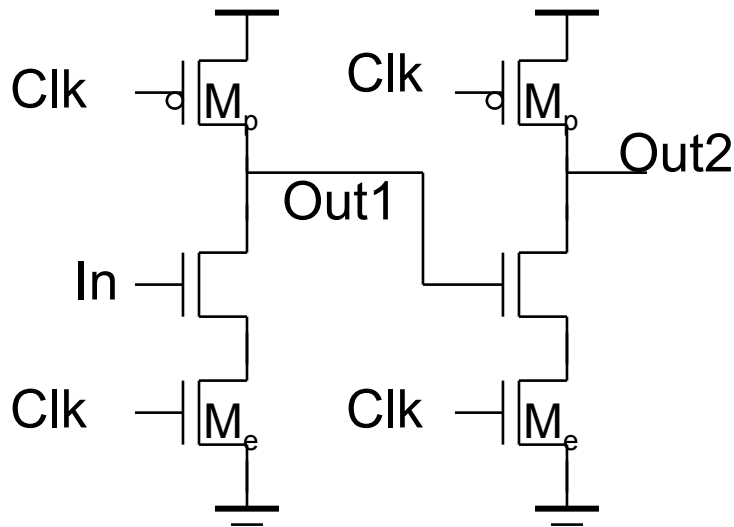


# *Other Effects*

- ❑ Capacitive coupling
- ❑ Substrate coupling
- ❑ Minority charge injection
- ❑ Supply noise (ground bounce)

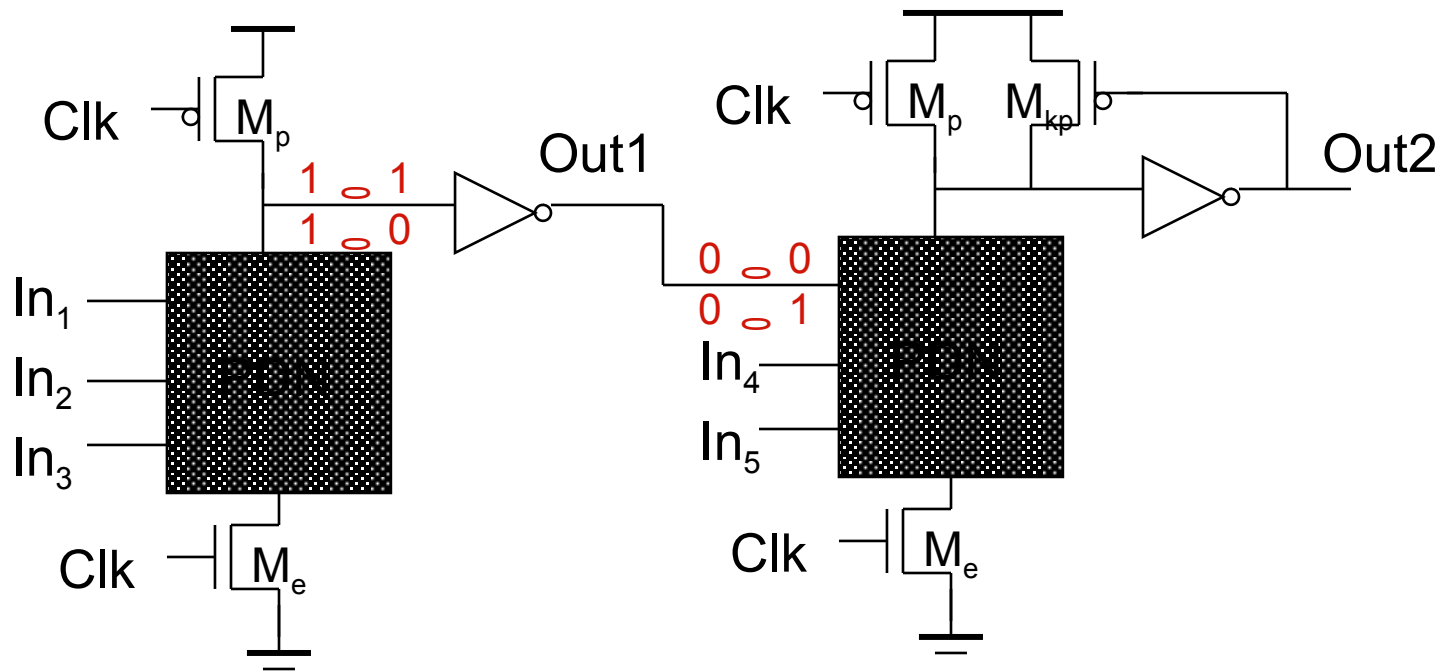


# Cascading Dynamic Gates

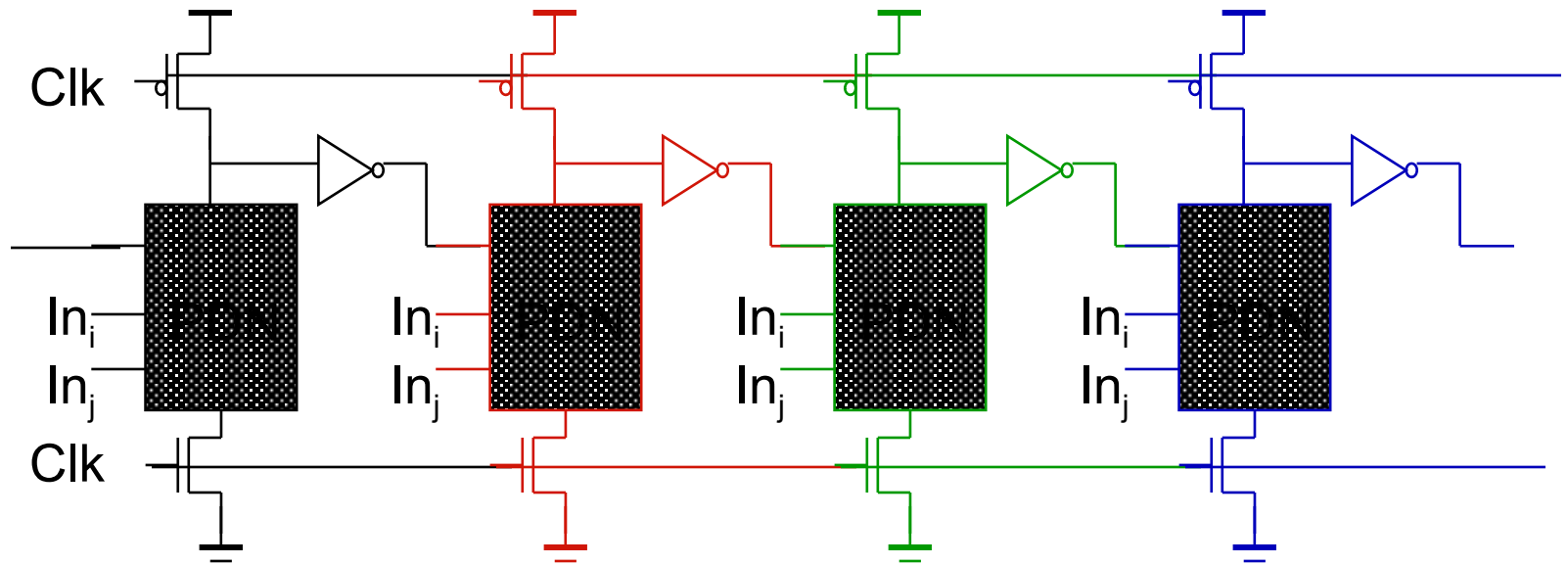


Only 0  $\rightarrow$  1 transitions allowed at inputs!

# Domino Logic



# Why Domino?

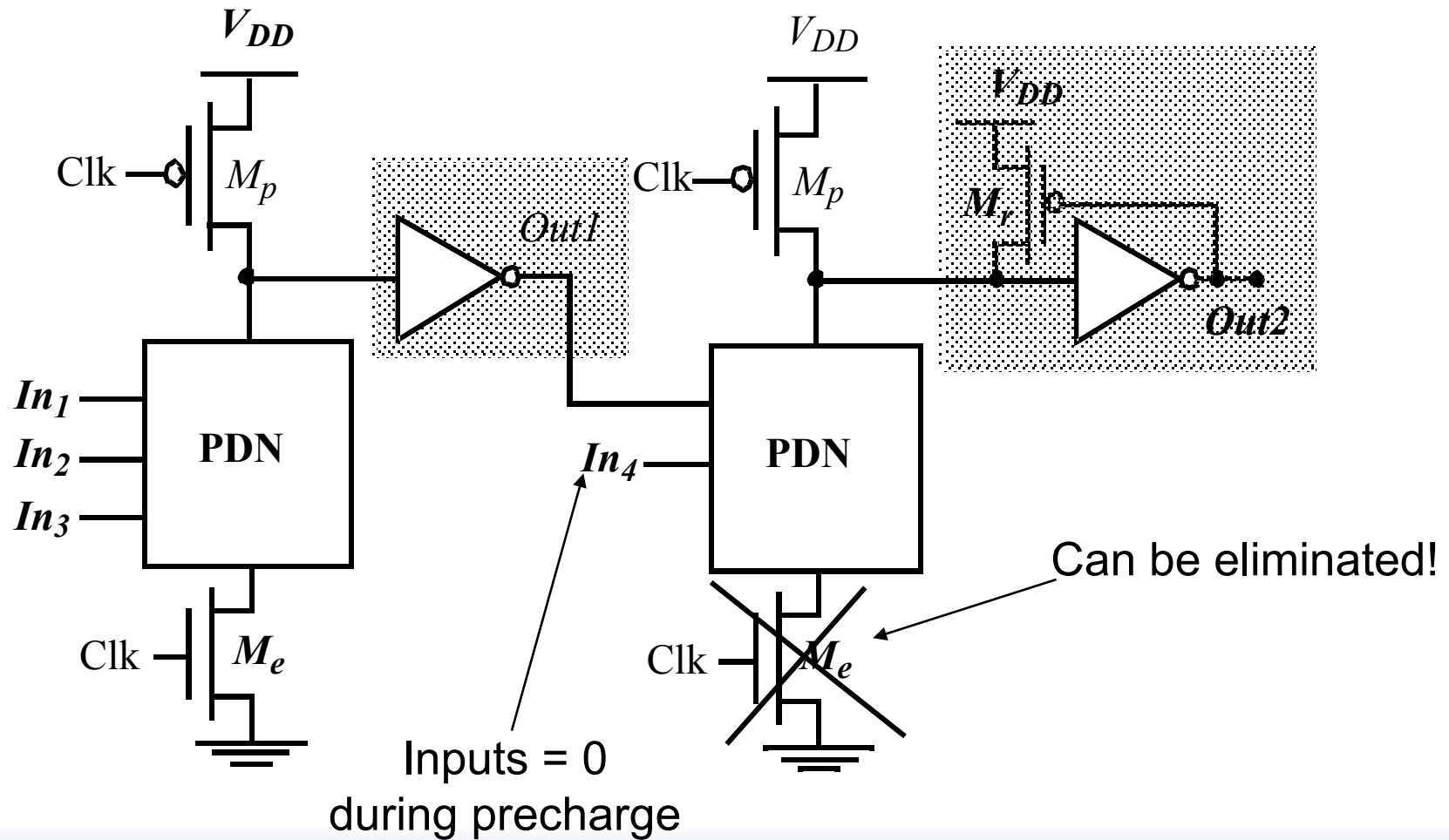


# Like falling dominos!

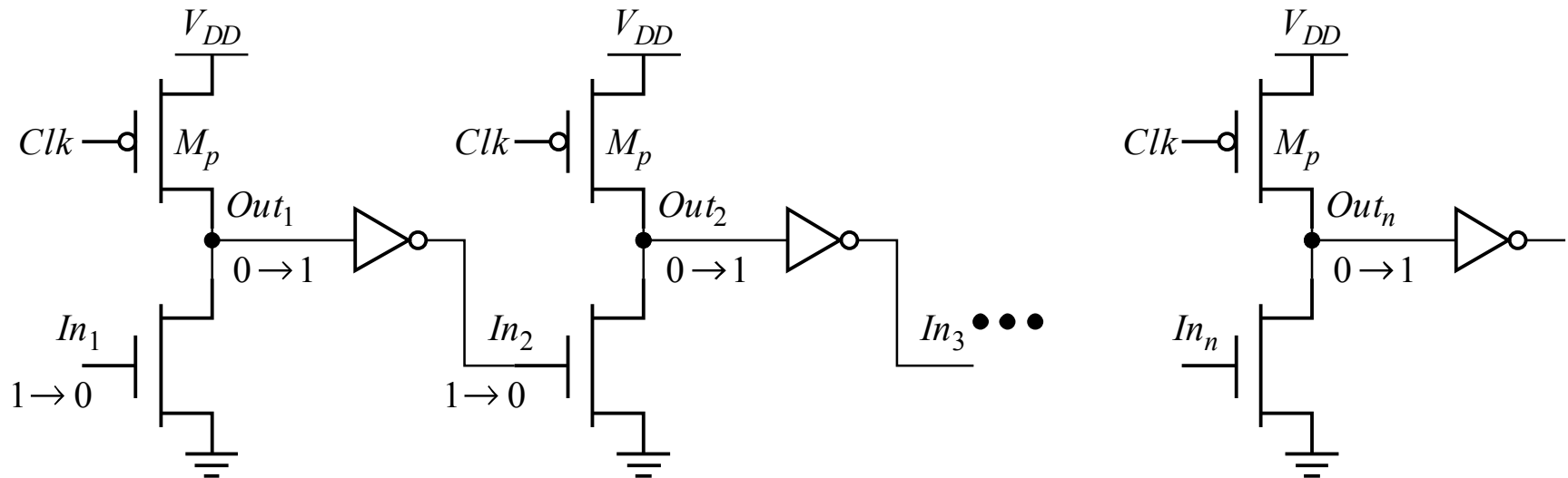
# *Properties of Domino Logic*

- ❑ Only non-inverting logic can be implemented
- ❑ Very high speed
  - static inverter can be skewed, only L-H transition
  - Input capacitance reduced – smaller logical effort

# Designing with Domino Logic

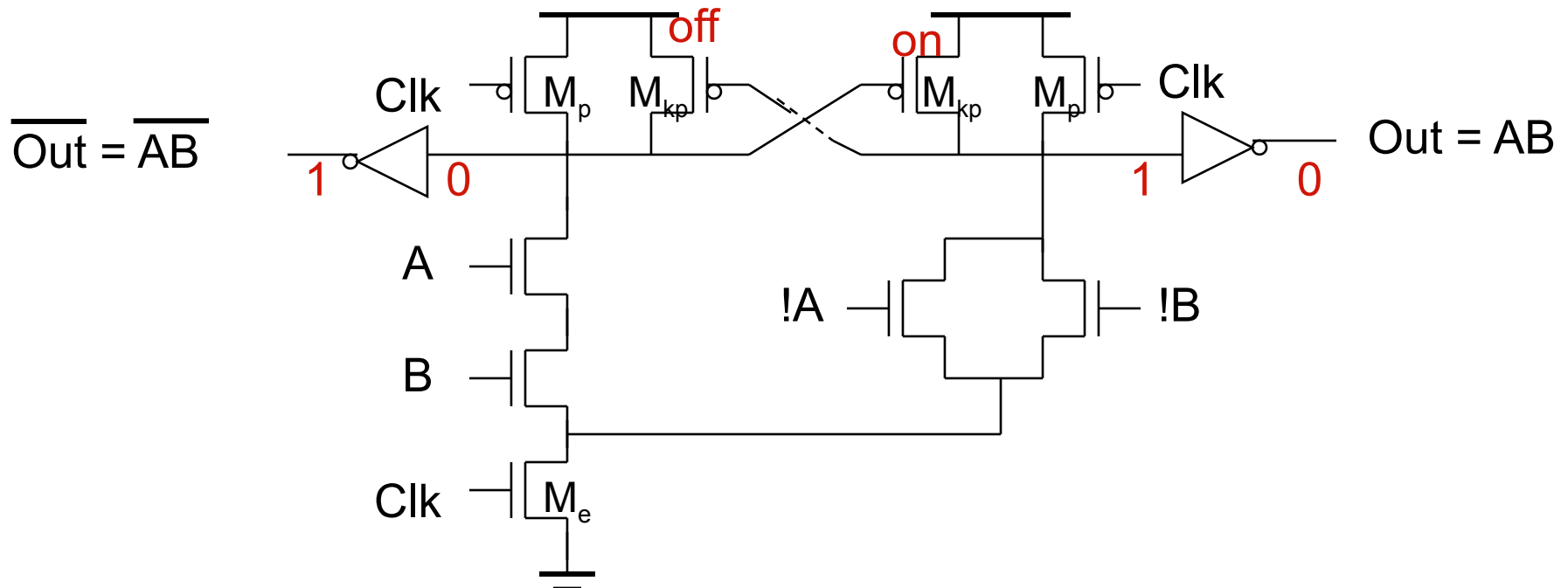


# Footless Domino



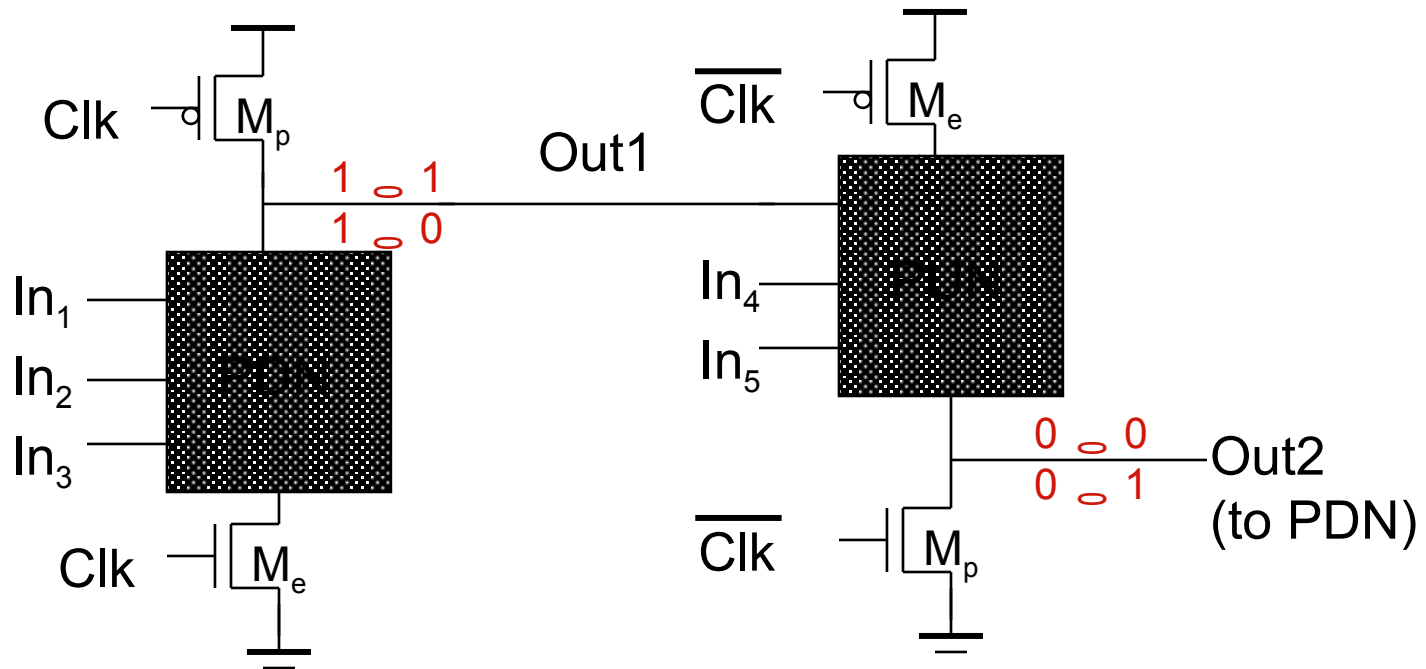
The first gate in the chain needs a foot switch  
Precharge is rippling – short-circuit current  
A solution is to delay the clock for each stage

# Differential (Dual Rail) Domino



Solves the problem of non-inverting logic

# np-CMOS

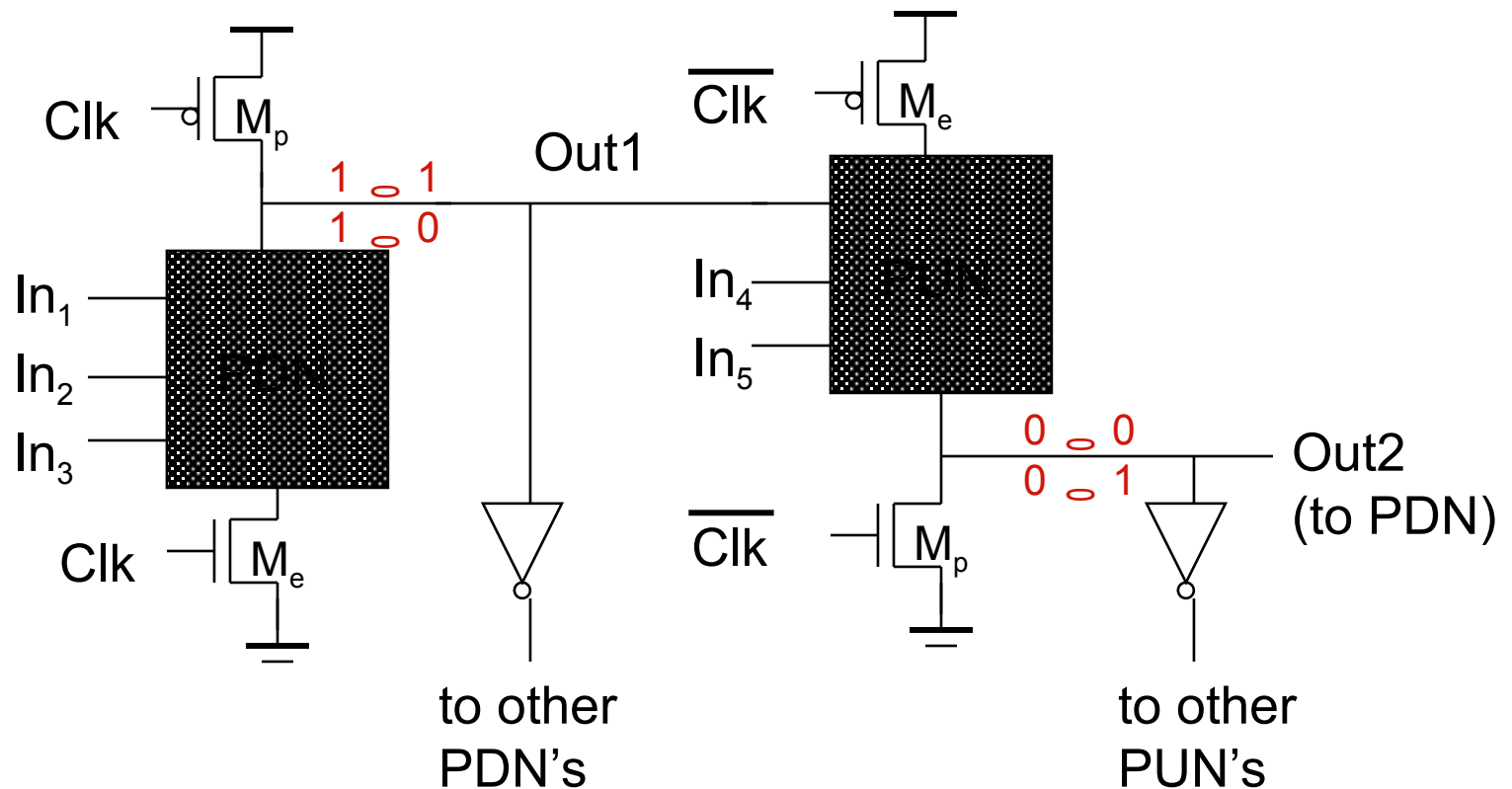


Only  $0 \rightarrow 1$  transitions allowed at inputs of PDN

Only  $1 \rightarrow 0$  transitions allowed at inputs of PUN



# NORA Logic



**WARNING:** Very sensitive to noise!