CG2027 Transistor-Level Digital Circuits

Handout #3: CMOS Logic

National University of Singapore

Kelvin Fong

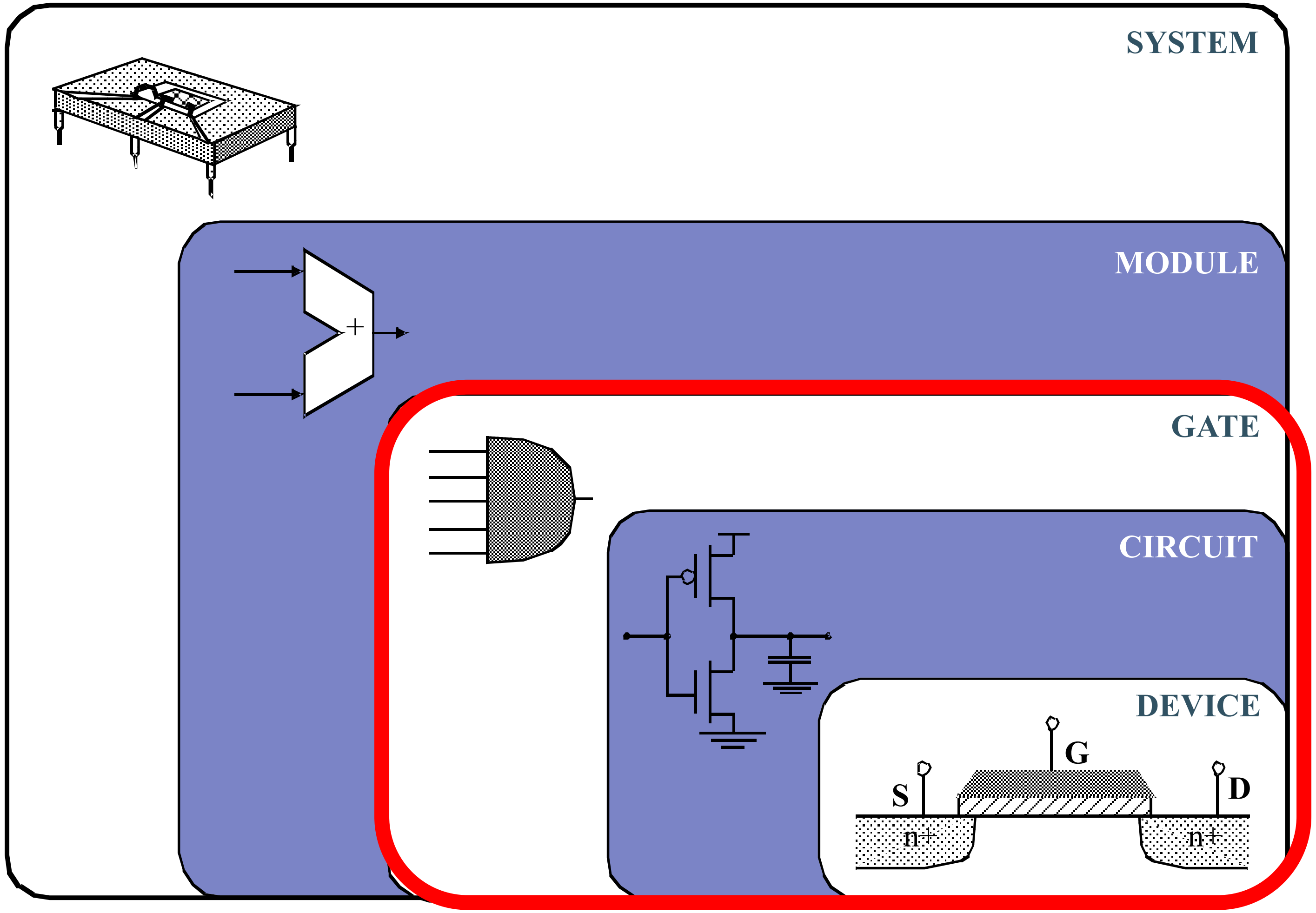
Lecture note adapted from J. Rabaey, A. Chandrakasan, B. Nikolic, “Digital Integrated Circuits” 2nd, Copyright prentice Hall/Pearson.

Lecture Overview

In this lecture, you will learn about

- CMOS logic design
- CMOS performance analysis (delay, power and sizing)
- Basic transmission gates

Design Abstraction Levels

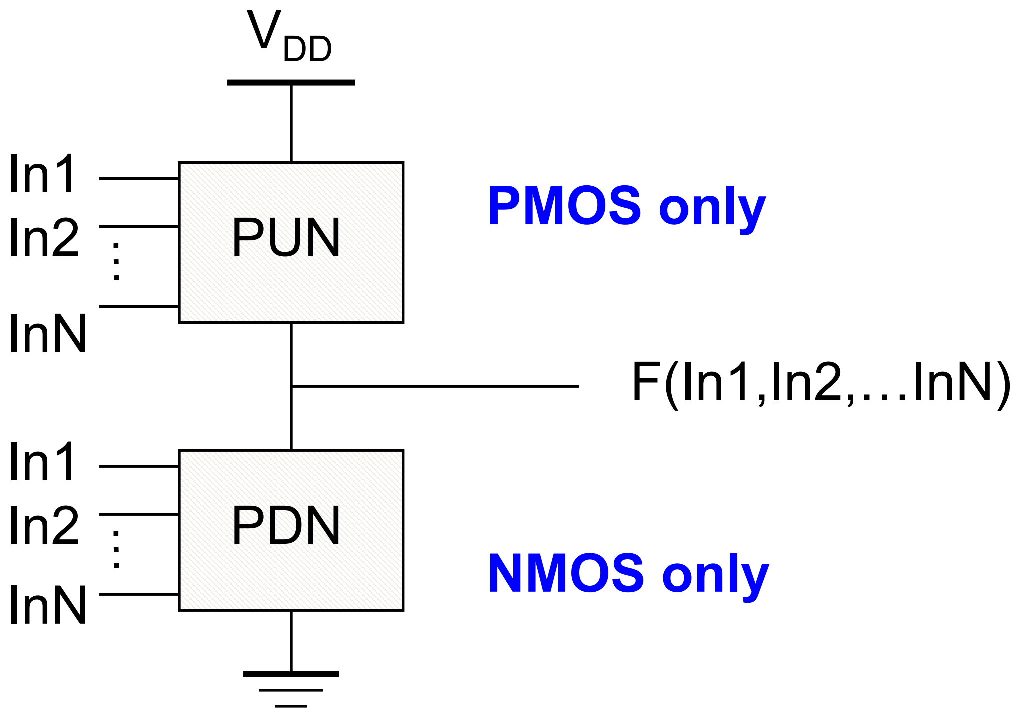


Static CMOS Circuit

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-resistance path.

The outputs of the gates **assume at all times the value of the Boolean function**, implemented by the circuit (ignoring the transient effects during switching periods).

Static Complementary CMOS

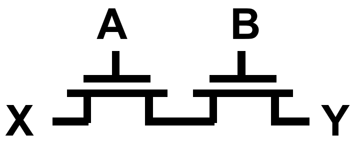


Pull-Up Network (PUN) and Pull-Down Network (PDN)
are **dual** logic networks

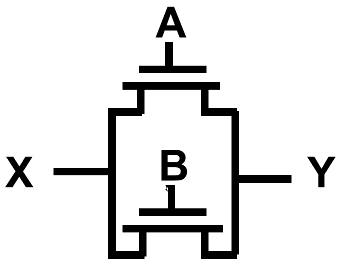
NMOS in Series/Parallel Connection

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

NMOS switch closes when switch control input is high



Y = X if A AND B = AB

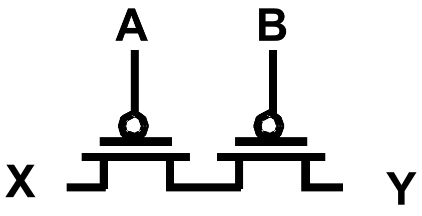


Y = X if A OR B = A+B

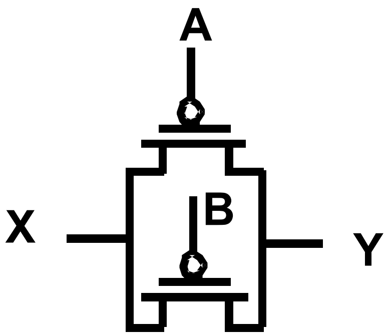
NMOS Transistors pass a “strong” 0, but a “weak” 1

PMOS in Series/Parallel Connection

PMOS switch closes when switch control input is low



Y = X if $\overline{A}$ AND $\overline{B}$ = $\overline{A + B}$



Y = X if $\overline{A}$ OR $\overline{B}$ = $\overline{AB}$

PMOS Transistors pass a "strong" 1 but a "weak" 0

Complementary CMOS Logic Style

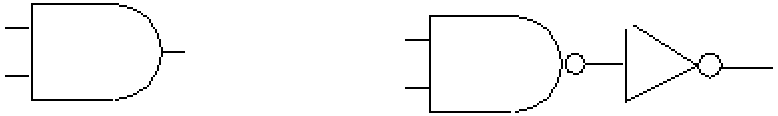
- PUN is the DUAL of PDN

  (can be shown using DeMorgan's Theorem's)

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

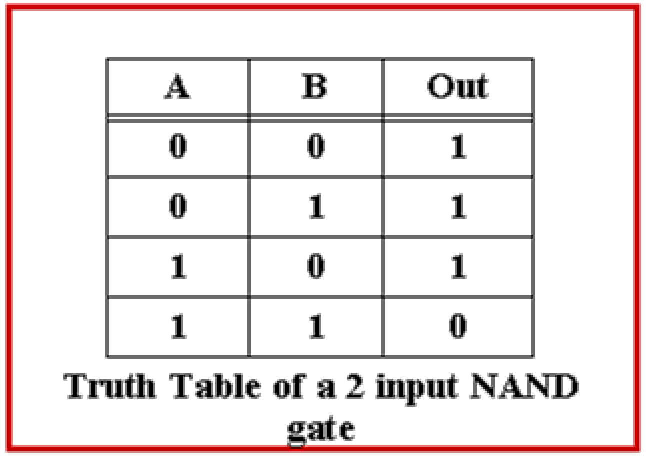
$$\overline{AB} = \bar{A} + \bar{B}$$

- The complementary gate is inverting



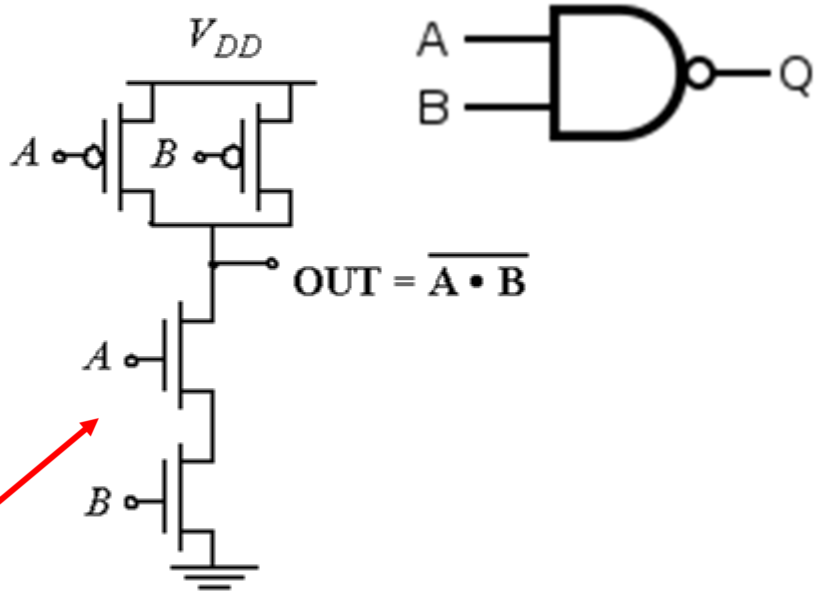
AND = NAND + INV

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



Pull Down Network (PDN): When both A and B are logic high, output is pulled down to zero as needed for NAND gate. Otherwise it remains high.

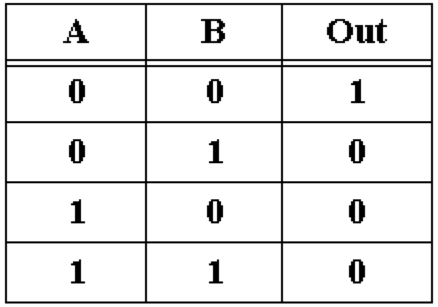
PDN: $G = A\,B \Rightarrow$ Conduction to GND

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

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

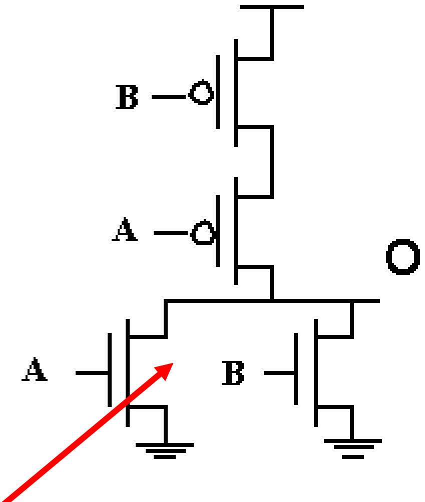
PUN is complementary to PDN, i.e. p-MOS devices are in parallel as n-MOS devices are in series.

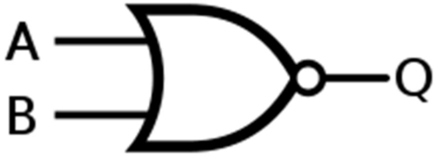
Example Gate: NOR



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

Truth Table of a 2 input NOR gate



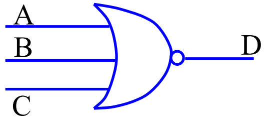


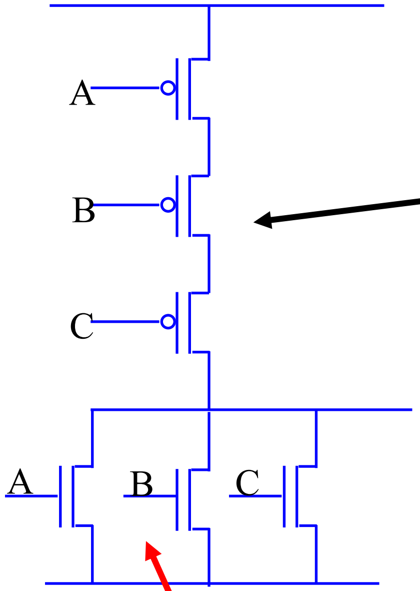
$OUT = \overline{A + B}$

Pull Down Network (PDN): When either A and B, or both are logic high, output is pulled down to zero as needed for NOR gate. Otherwise it remains high.

PUN is complementary to PDN, i.e. p-MOS devices are in series as n-MOS devices are in parallel.

Example Gate: 3-Input NOR

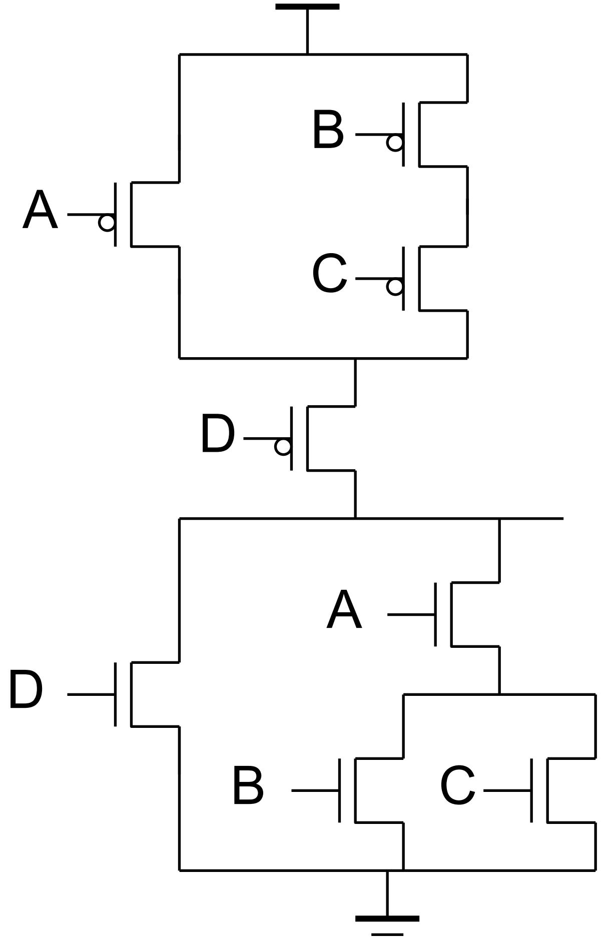




PUN: Three p-MOS devices are omplementary i.e. in series. Hence all A, B and C need to be low to make output high.

Pull Down Network (PDN): When any of A, B, or C are logic high, output is pulled down to zero as needed for a 3-input NOR gate. Otherwise it remains high.

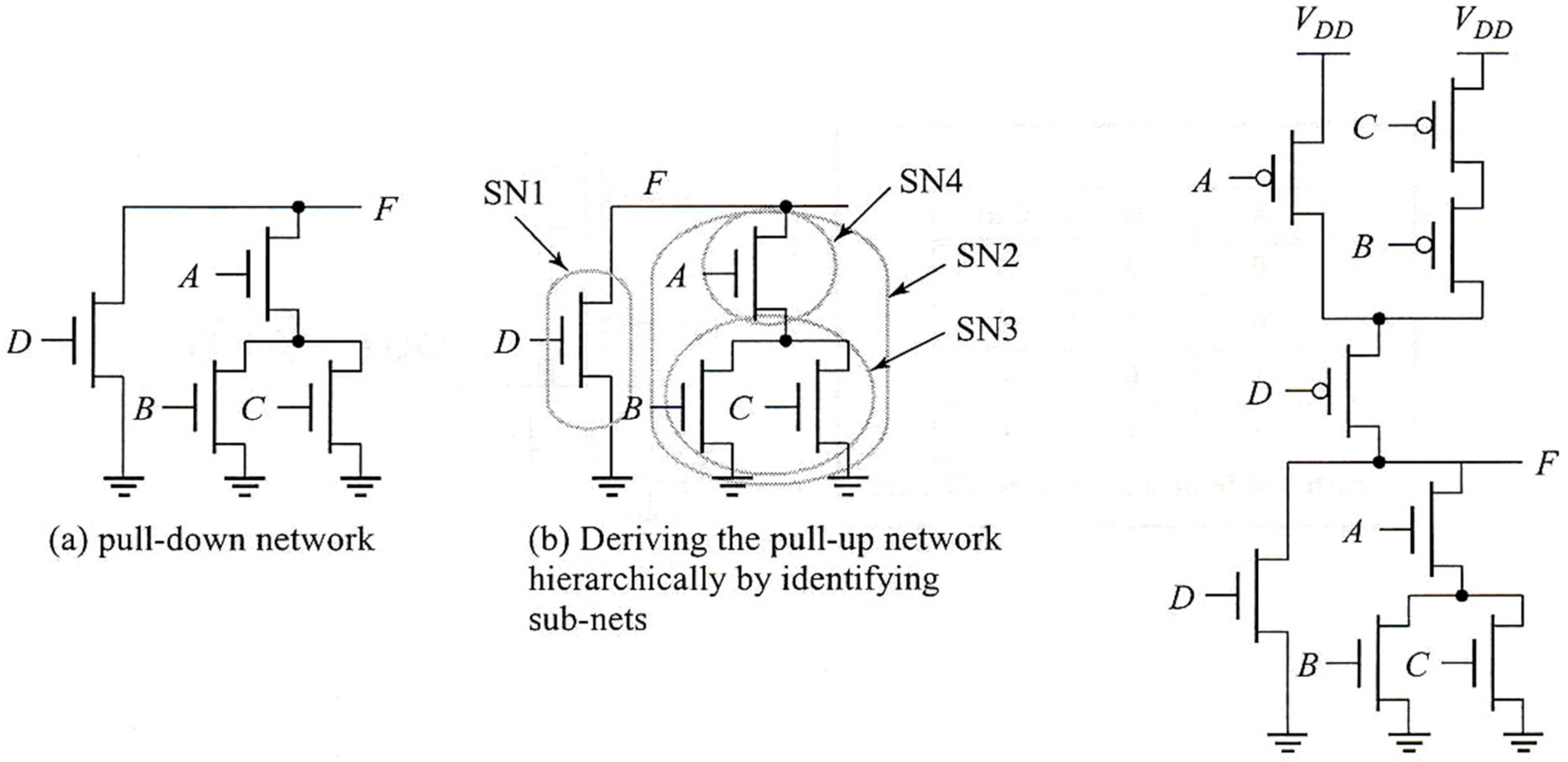
Complex CMOS Gate



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

How do we construct a complex CMOS gate with a given Boolean Function?

Constructing a Complex Gate



(a) pull-down network

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

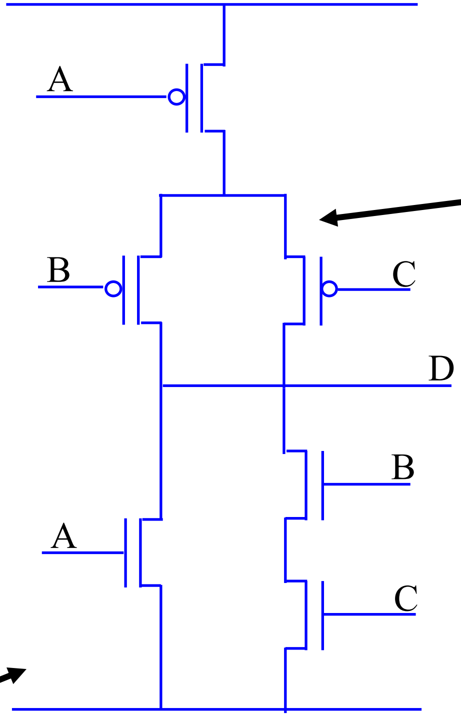
(c) complete gate

CMOS Logic Implementations

- Here is implementation of arbitrary combinational logic in CMOS.

$= \overline{A + B \cdot C}$

D

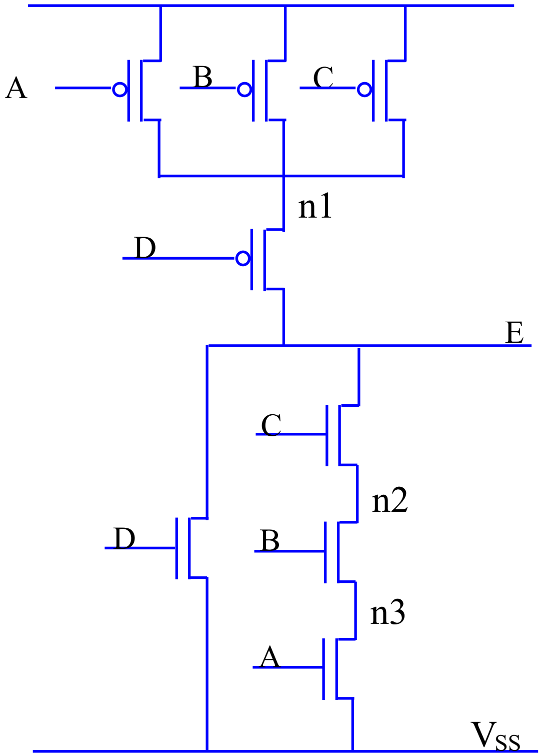


- Three p-MOS devices are complementary to n-MOS i.e. B and C p-MOS are in parallel and this combination is in series with p-MOS for A.

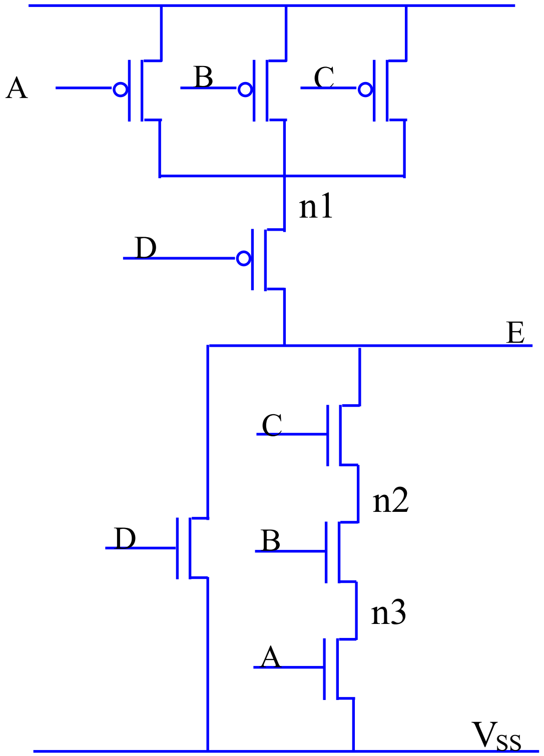
- Three n-MOS devices are arranged so that high A alone can pull the output D to low. Both B and C need to be high to pull D low if A is low.

CMOS Logic Implementations

- Here is another arbitrary logic implemented in CMOS. What is the logic function implemented?



CMOS Logic Implementations

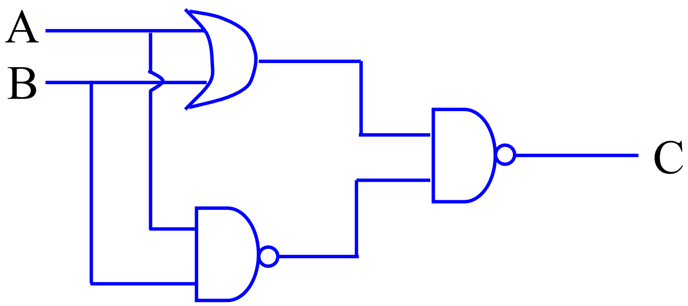


- Design MOSFET sizes so that the maximum rise and fall delay are the same and are equal to that of minimum size inverter. For inverter, nMOS W=L=2 λ, pMOS W=4 λ,L=2 λ where 2 λ is the minimum feature..
- The delay from each of independent series connected paths must be equal to that of inverter as this gives the maximum delay. If more than one path are turned on, the delay will be lower as these provide additional pull up or pull down paths in parallel.
- There are 3 series connected devices A,B,C in the pull down path. Hence their widths have to be W=6 λ. All L are maintained at 2 λ. W for D will be same as that of inverter W=2 λ.
- All pull up paths have 2 series connected devices and hence all pMOS devices will have L=2 λ and W=8 λ to achieve the same maximum delay as inverter. The minimum delay will be 2/3 of the inverter when all 3 parallel paths are turned on.

CMOS Logic Implementations

- Here is one of the most complex XNOR logic which will be implemented in CMOS.

$$\text{XNOR}: \ \overline{(A+B)\cdot\overline{AB}}$$



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.

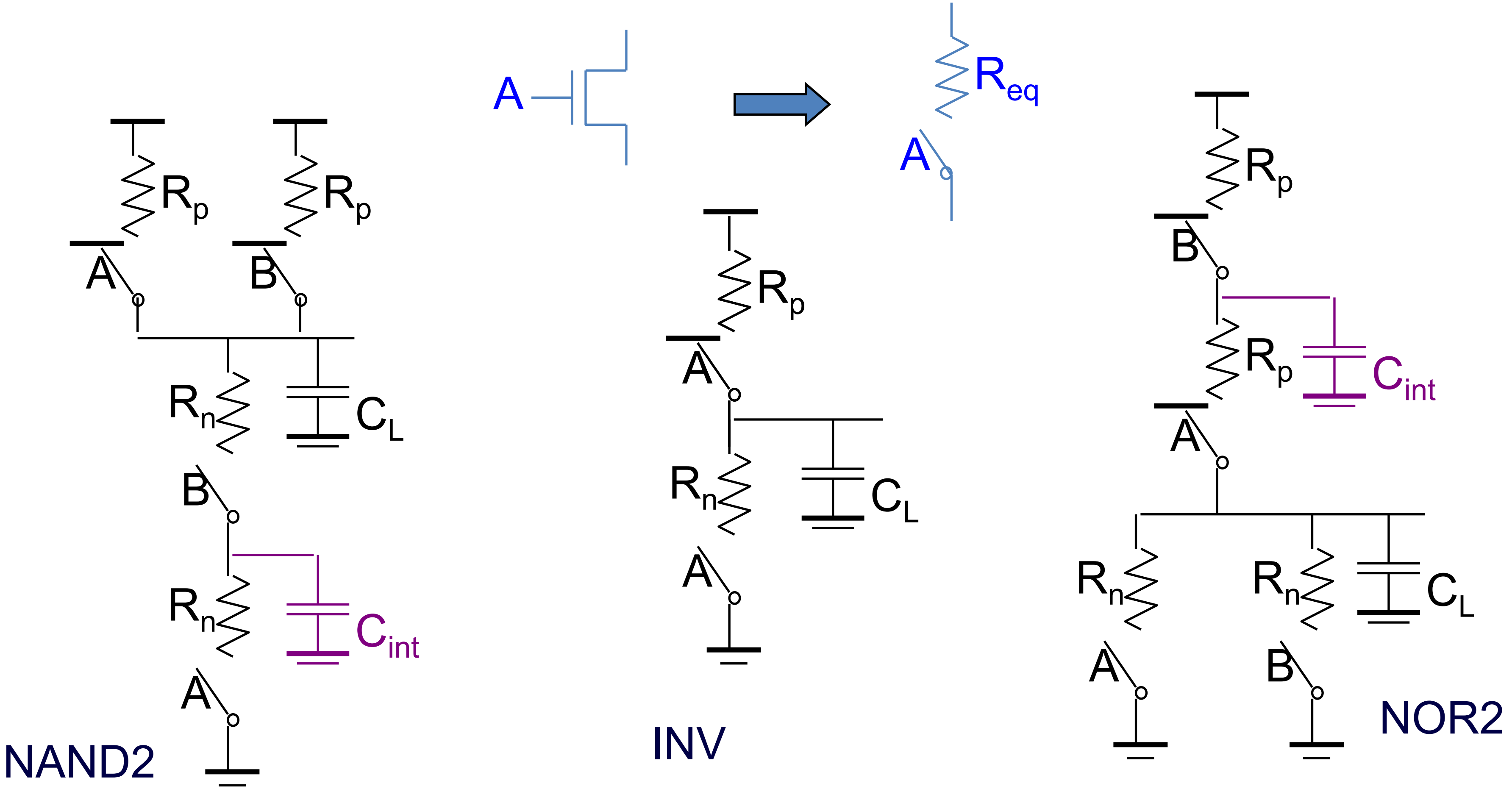
Comparable rise and fall times:

(under appropriate sizing conditions)

CMOS Properties

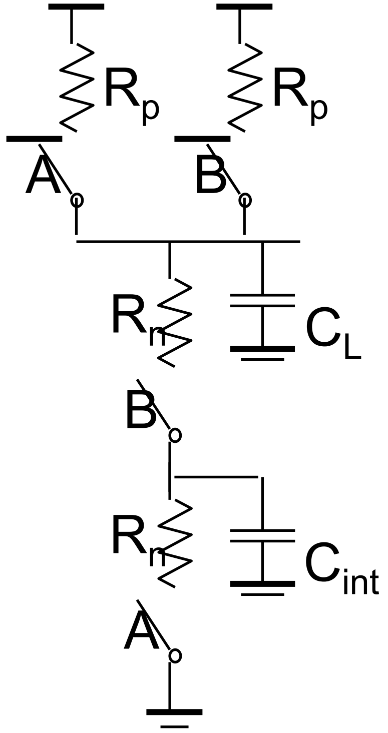
- Full rail-to-rail swing; **high noise margins**
- Logic levels not dependent upon the relative device sizes; **ratioless**
- Always a path to Vdd or Gnd in steady state from the output; **low** output impedance
- Extremely **high** input resistance; nearly zero steady-state input current
- No direct path in steady state between power and ground; no **static** power dissipation
- Propagation delay function of load capacitance and resistance of transistors

Switch Delay Model

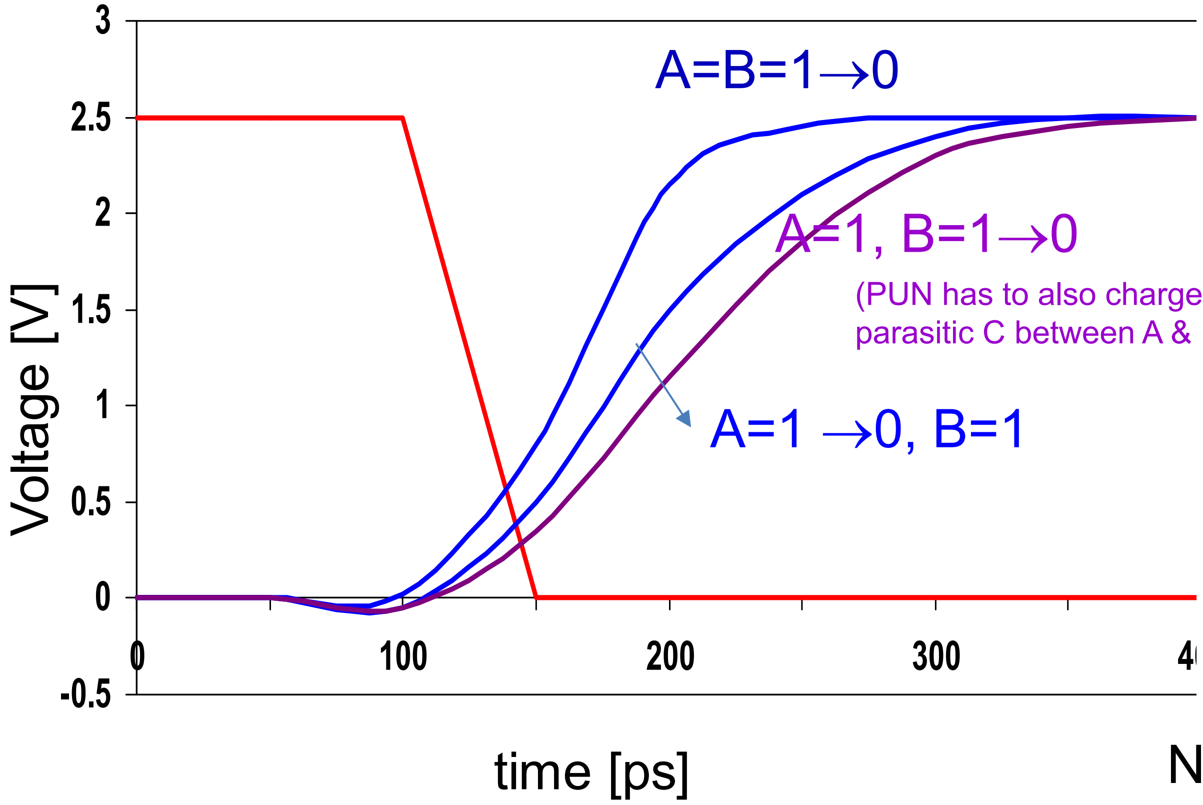


Input Pattern Effects on Delay

- 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

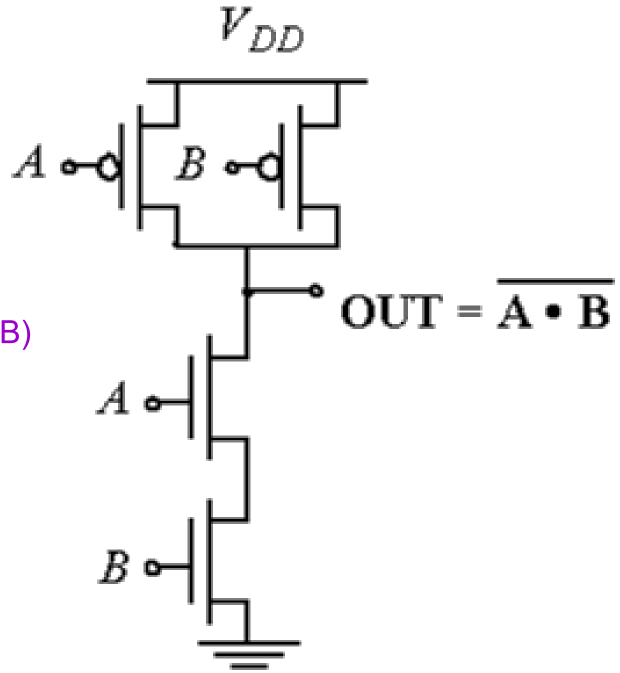


A=B=1→0

A=1, B=1→0

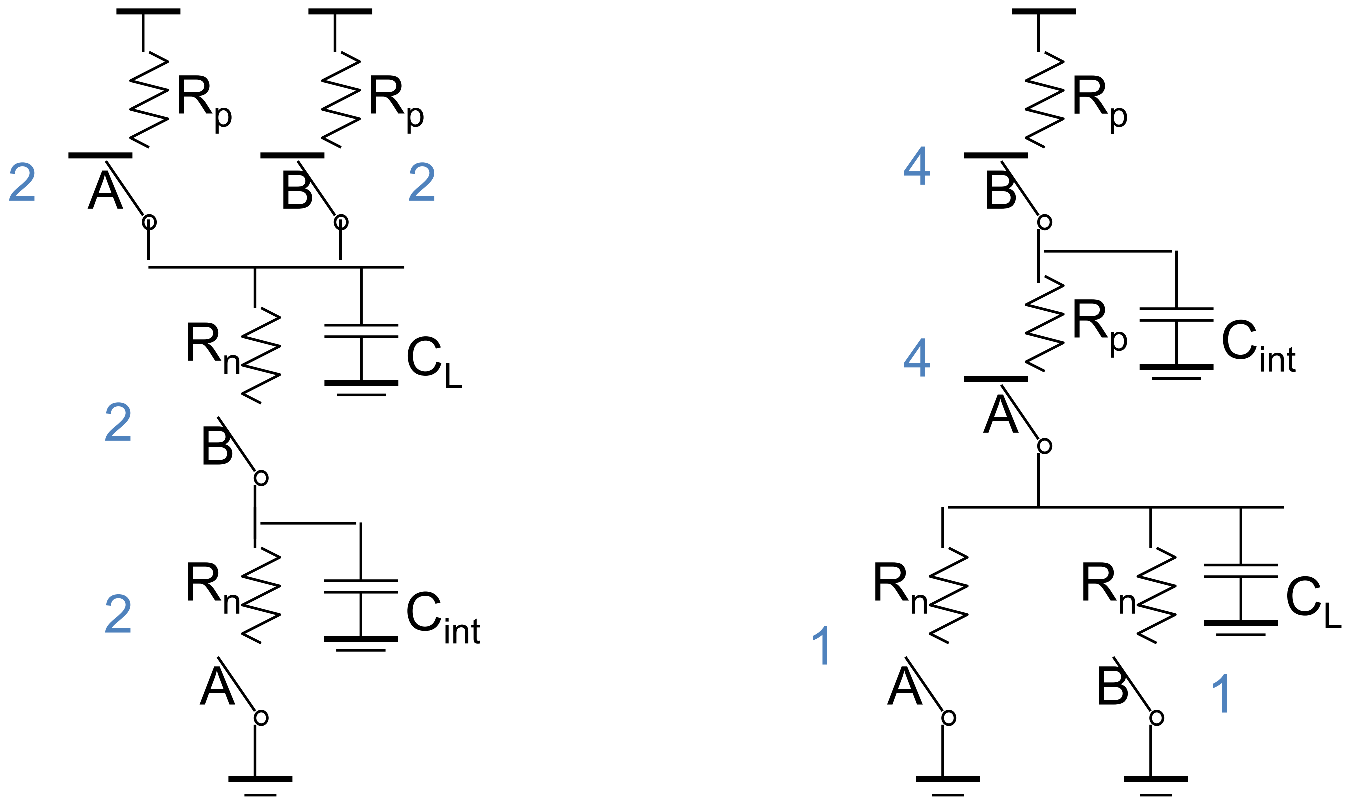
(PUN has to also charge parasitic C between A & B)

A=1 →0, B=1



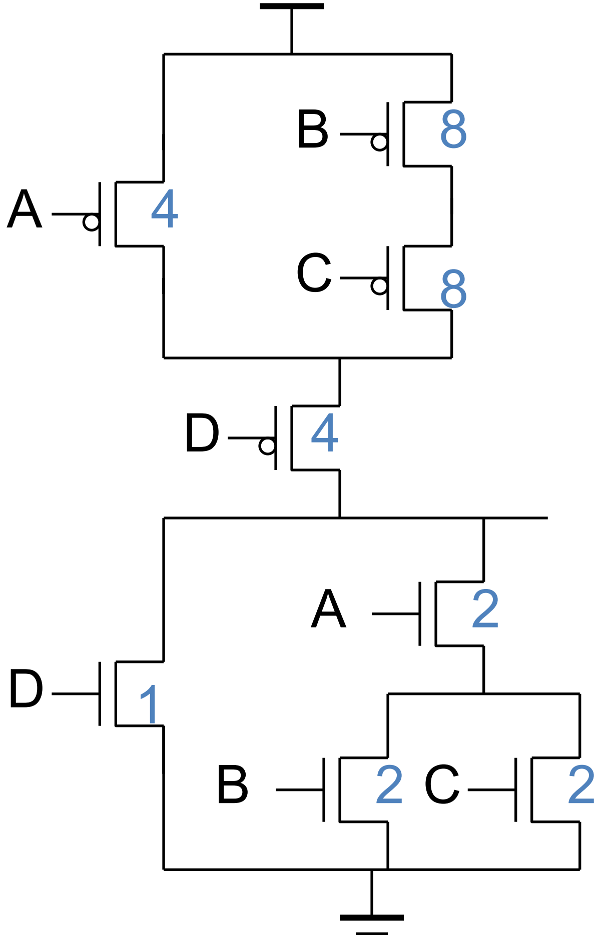
NMOS = 0.5μm/0.25 μm
PMOS = 0.75μm/0.25 μm
C_L = 100 fF

Transistor Sizing



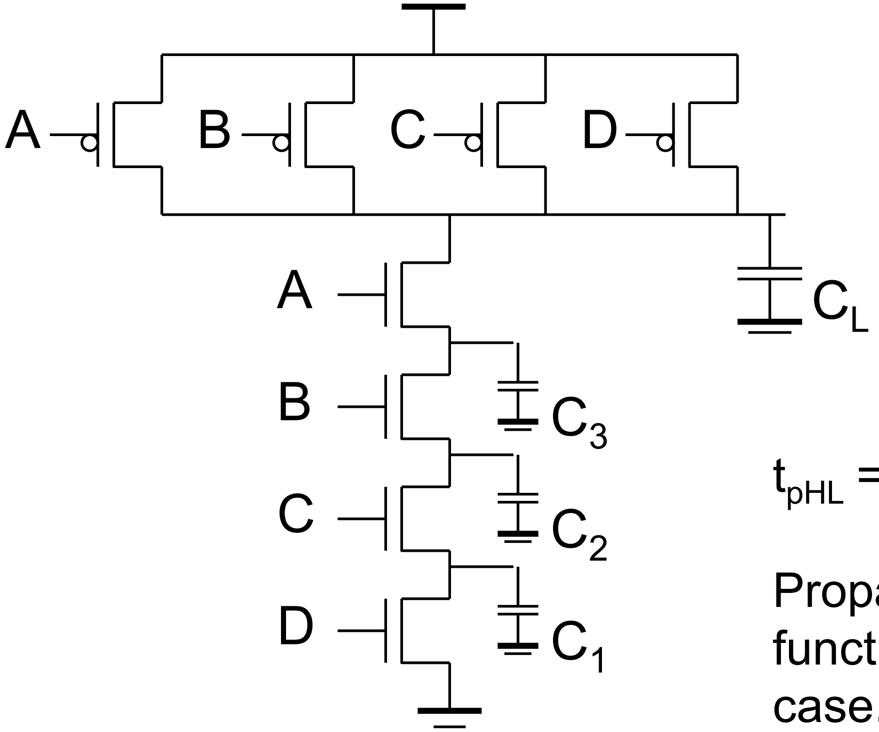
NAND vs. NOR

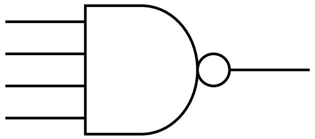
Transistor Sizing a Complex CMOS Gate



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

Fan-In Considerations



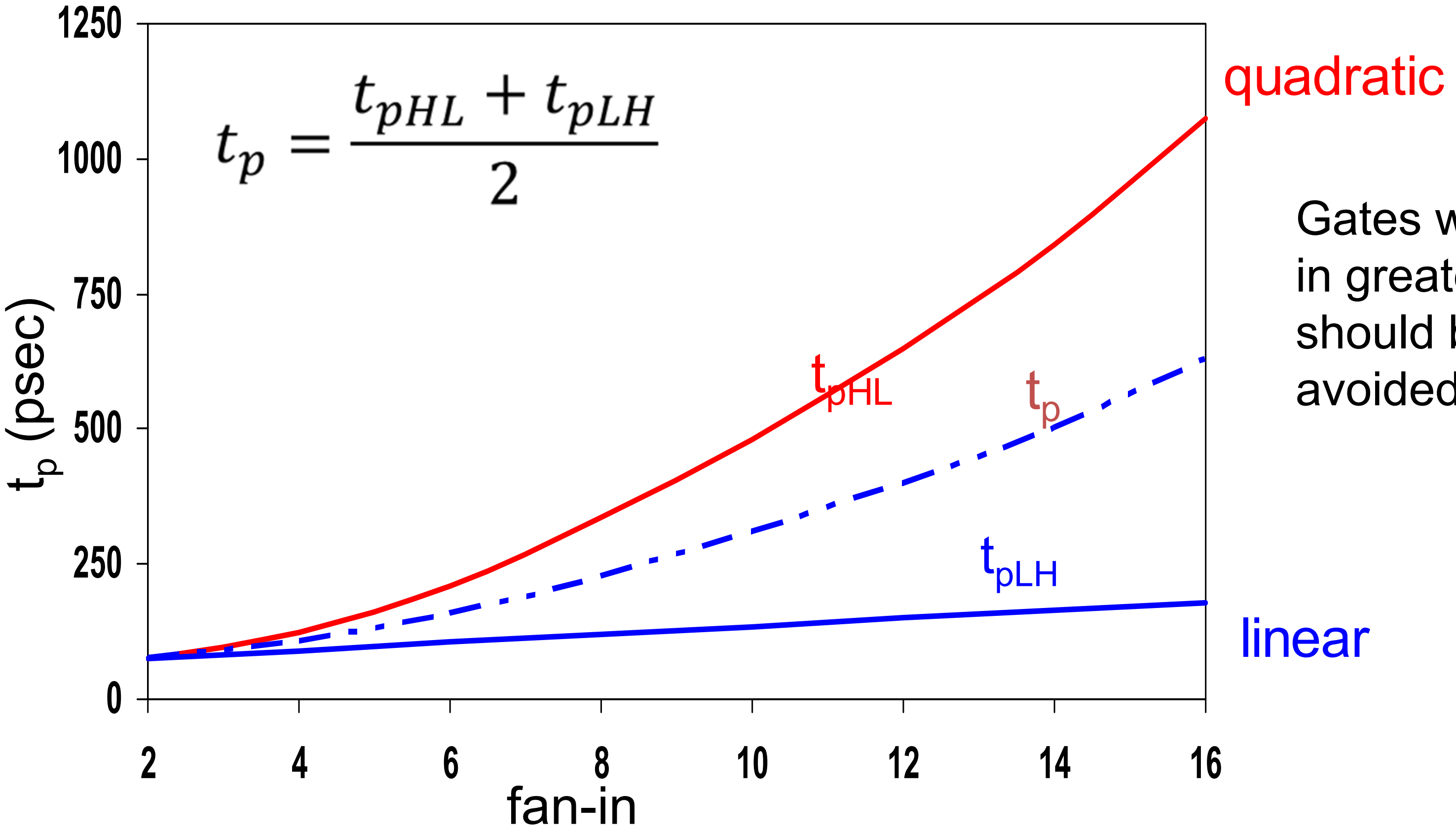


Distributed RC model
(Elmore delay)

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

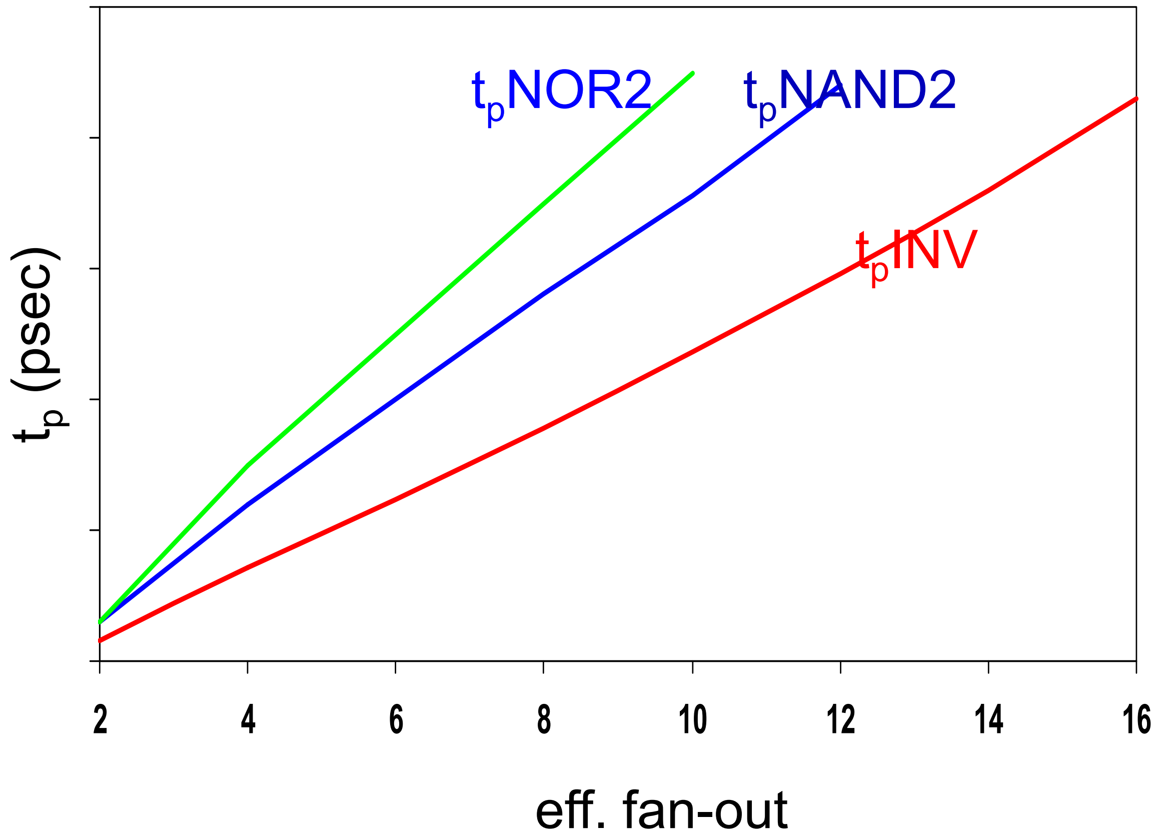
Propagation delay deteriorates rapidly as a function of fan-in – **quadratically** in the worst case.

t_p as a Function of Fan-In



Gates with a fan-in greater than 4 should be avoided.

t_p as a Function of Fan-Out

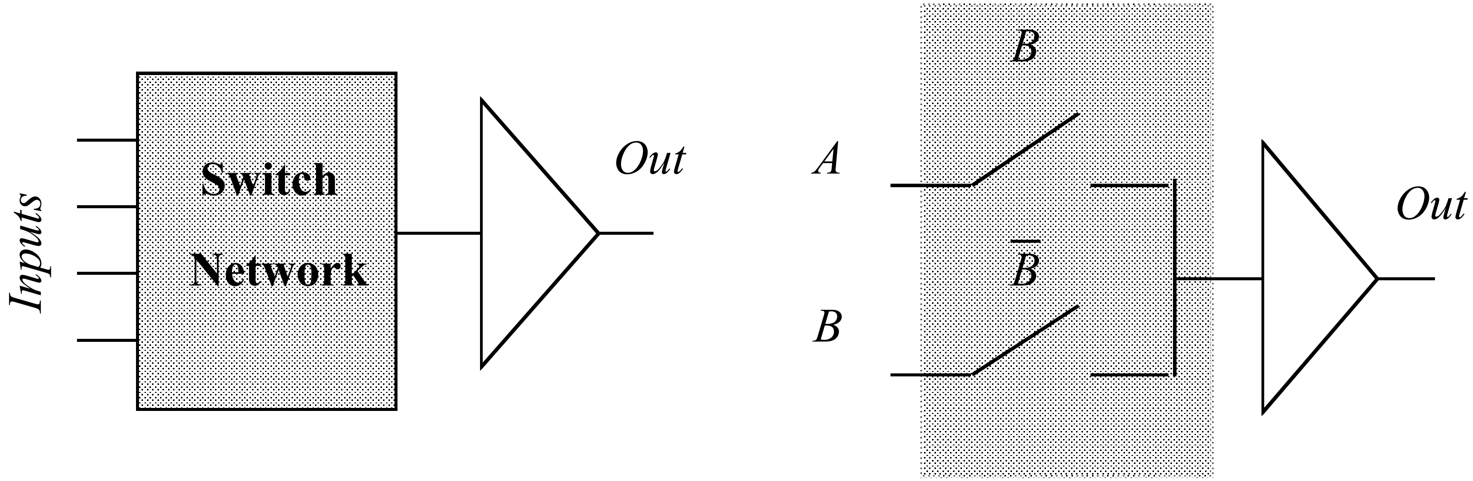


All gates have the same drive current.

Slope is a function of "driving strength"

PASS-TRANSISTOR LOGIC

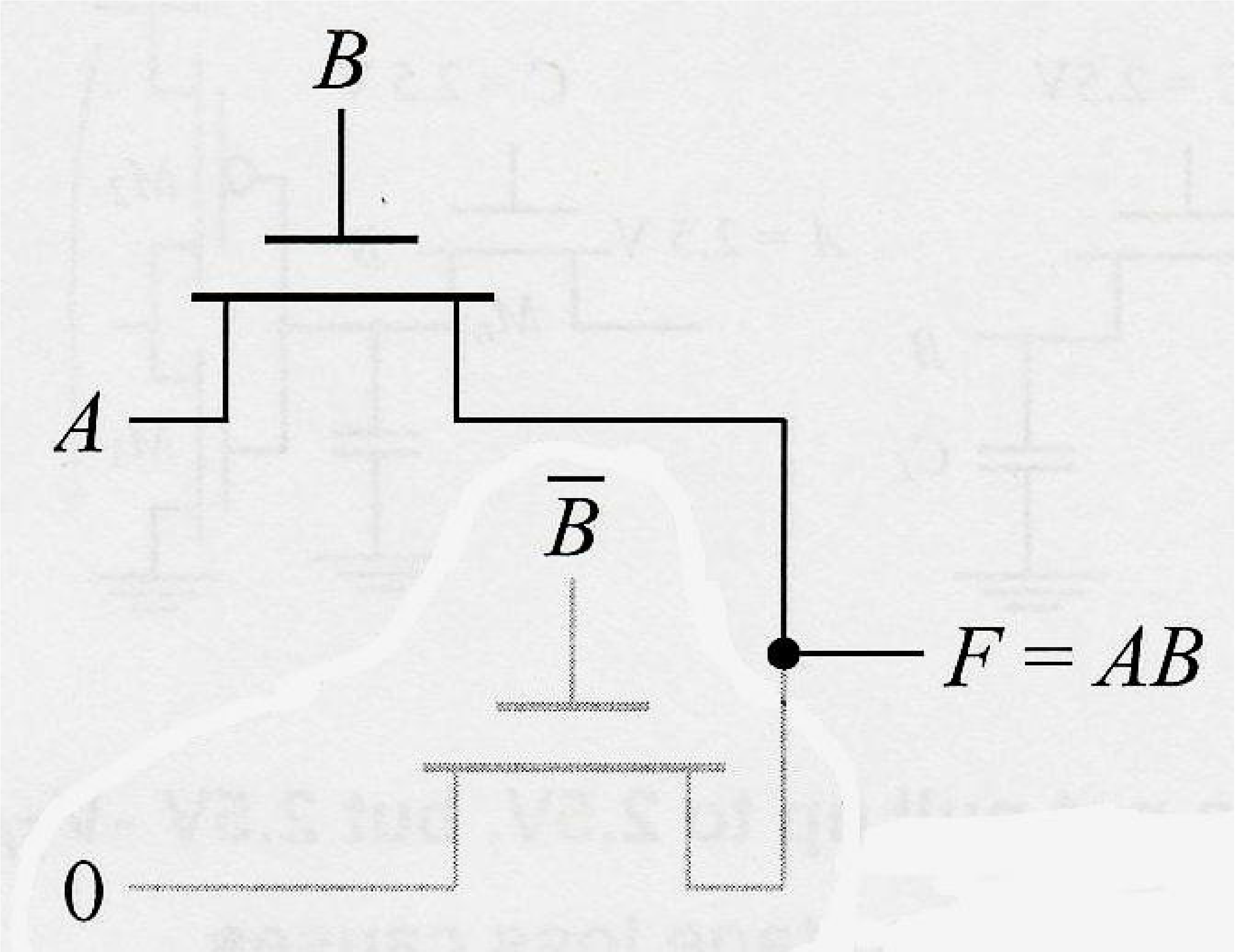
Pass-Transistor Logic



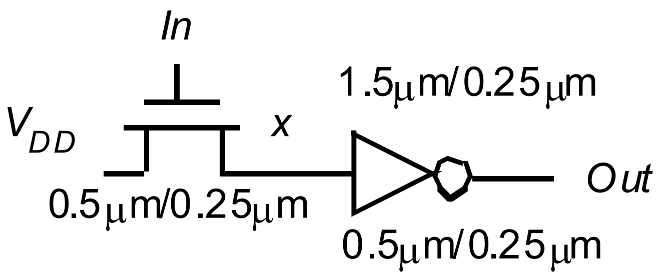
- N transistors
- No static consumption

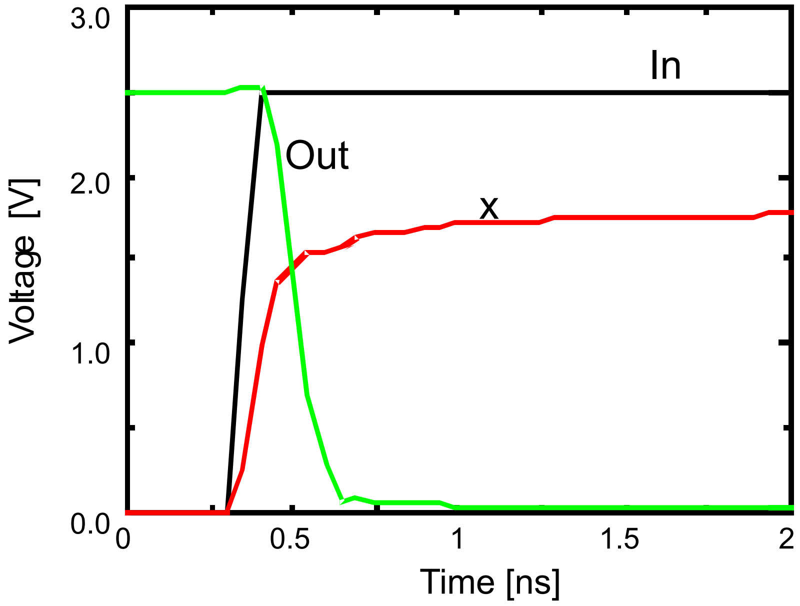
Note: Pass transistor logic trades off reliability vs. area

Example: AND Gate



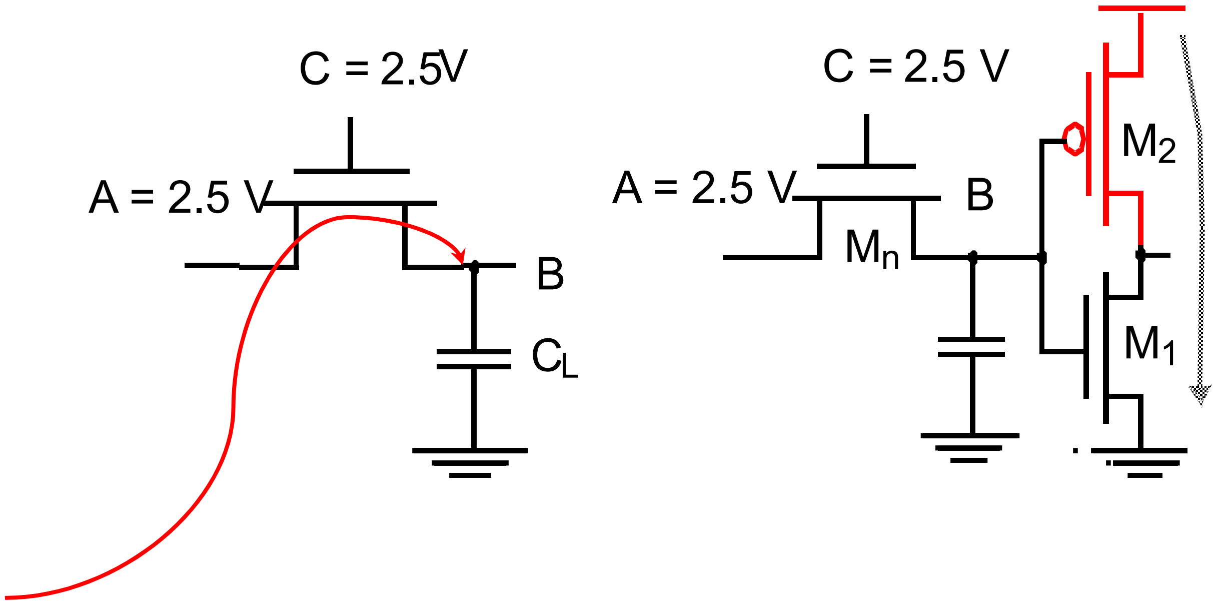
NMOS-Only Logic





Voltage drop is a serious problem in NMOS-only logic

NMOS-Only Switch

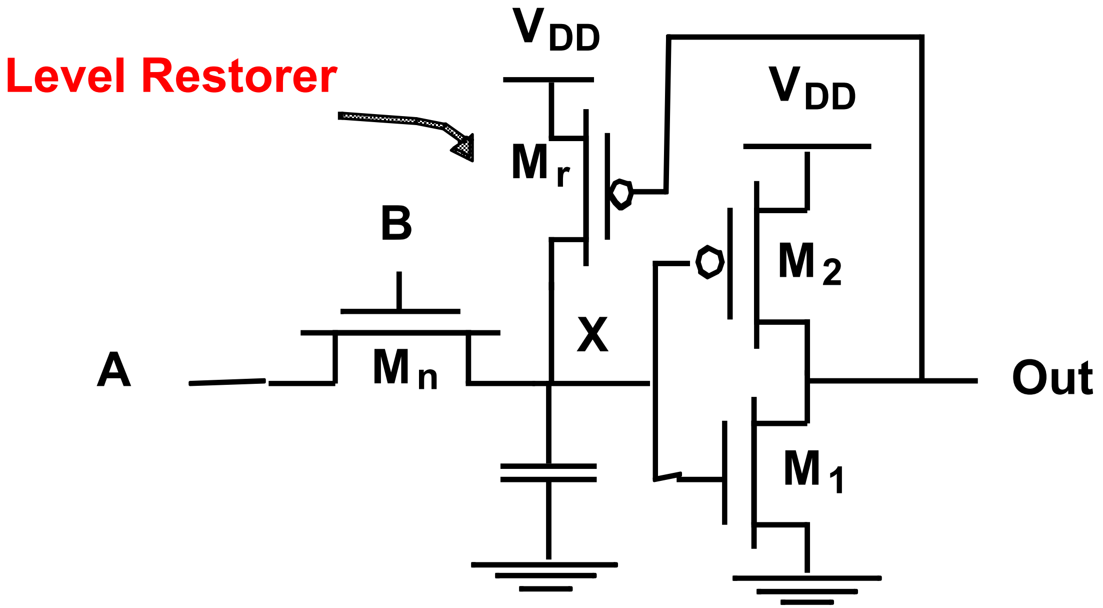


V_B does not pull up to 2.5V, but only up to 2.5V - V_{TN}

Threshold voltage loss causes
static power consumption

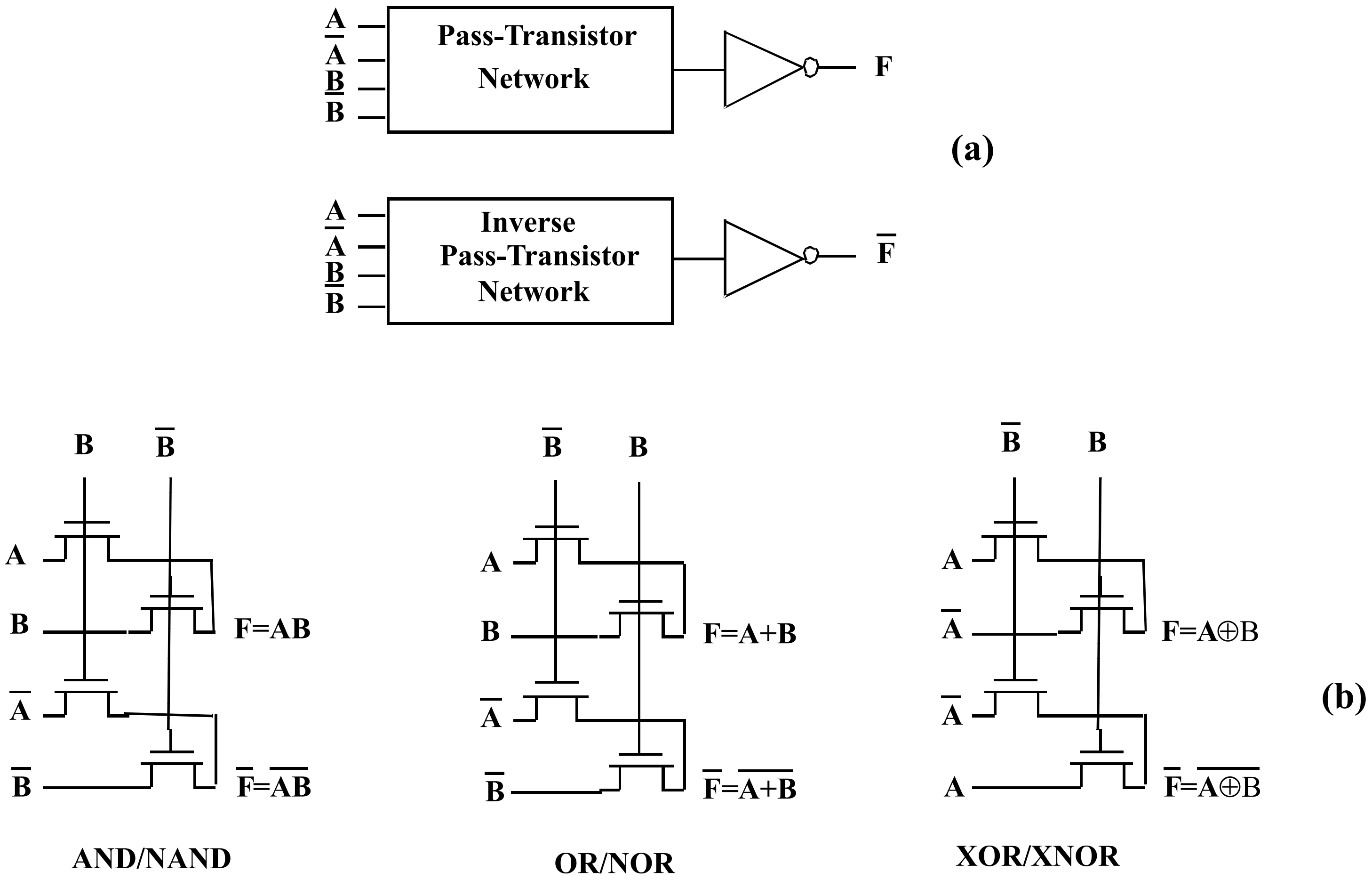
NMOS has higher threshold than PMOS (body effect)

Solution 1: Level Restoring Transistor

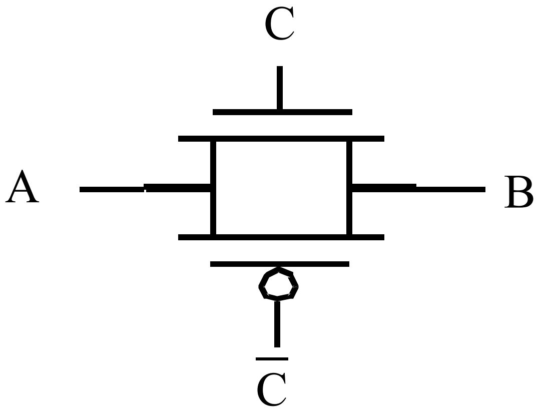


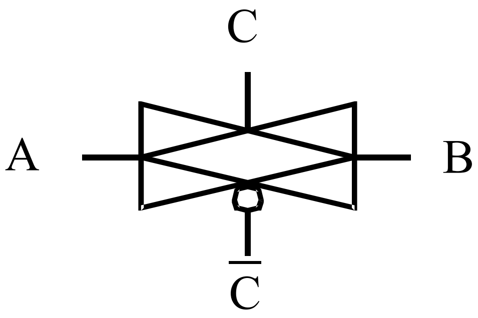
- Advantage: Full Swing
- Restorer adds capacitance, takes away pull down current at X
- Ratio problem (if too strong, Out won’t switch to “0”)

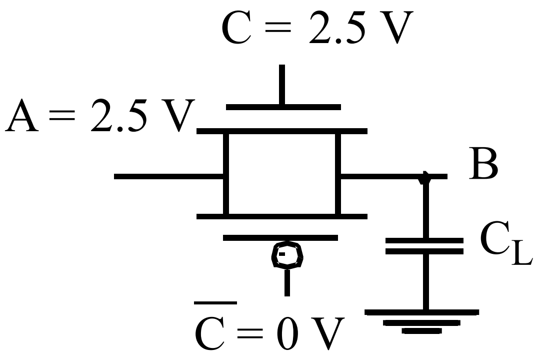
Complementary Pass Transistor Logic



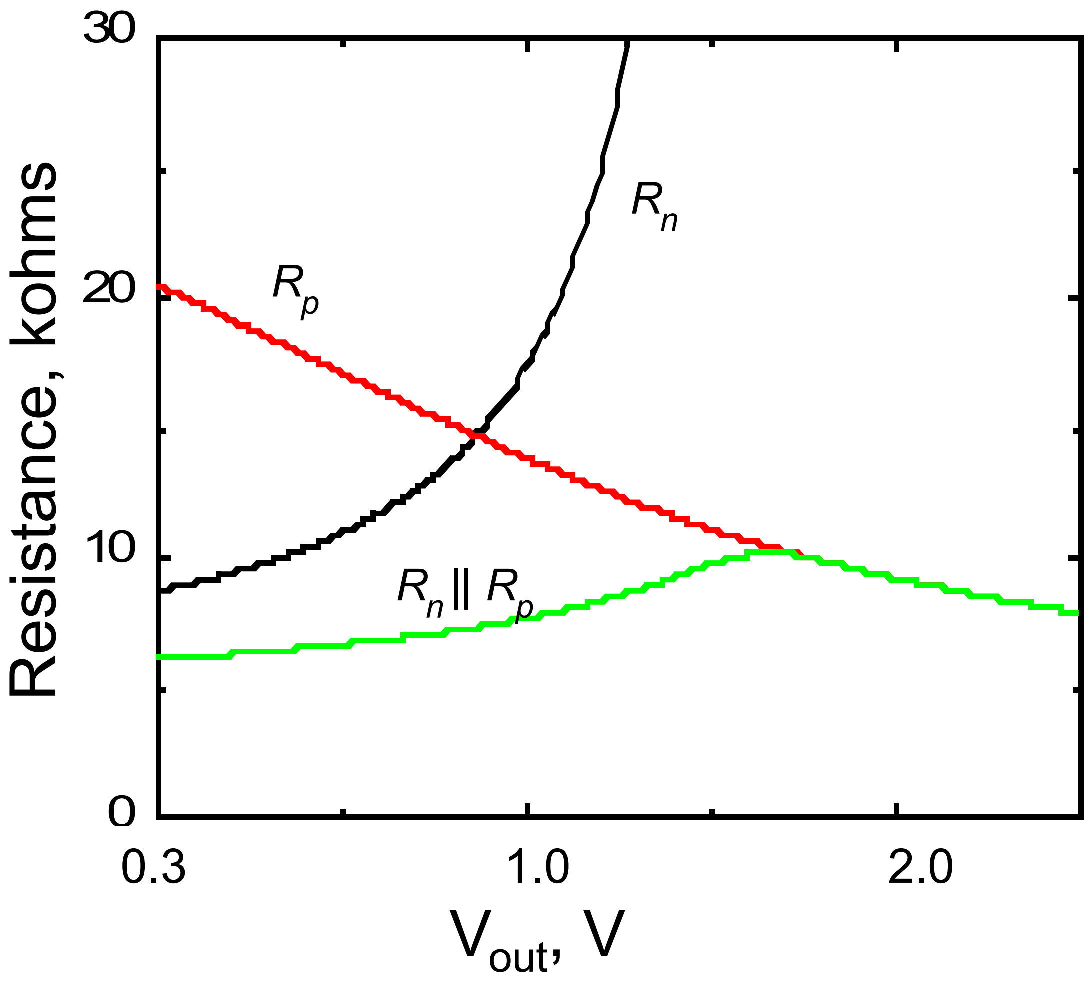
Solution 2: Transmission Gate

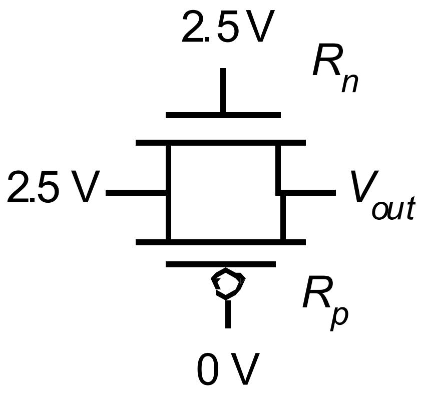




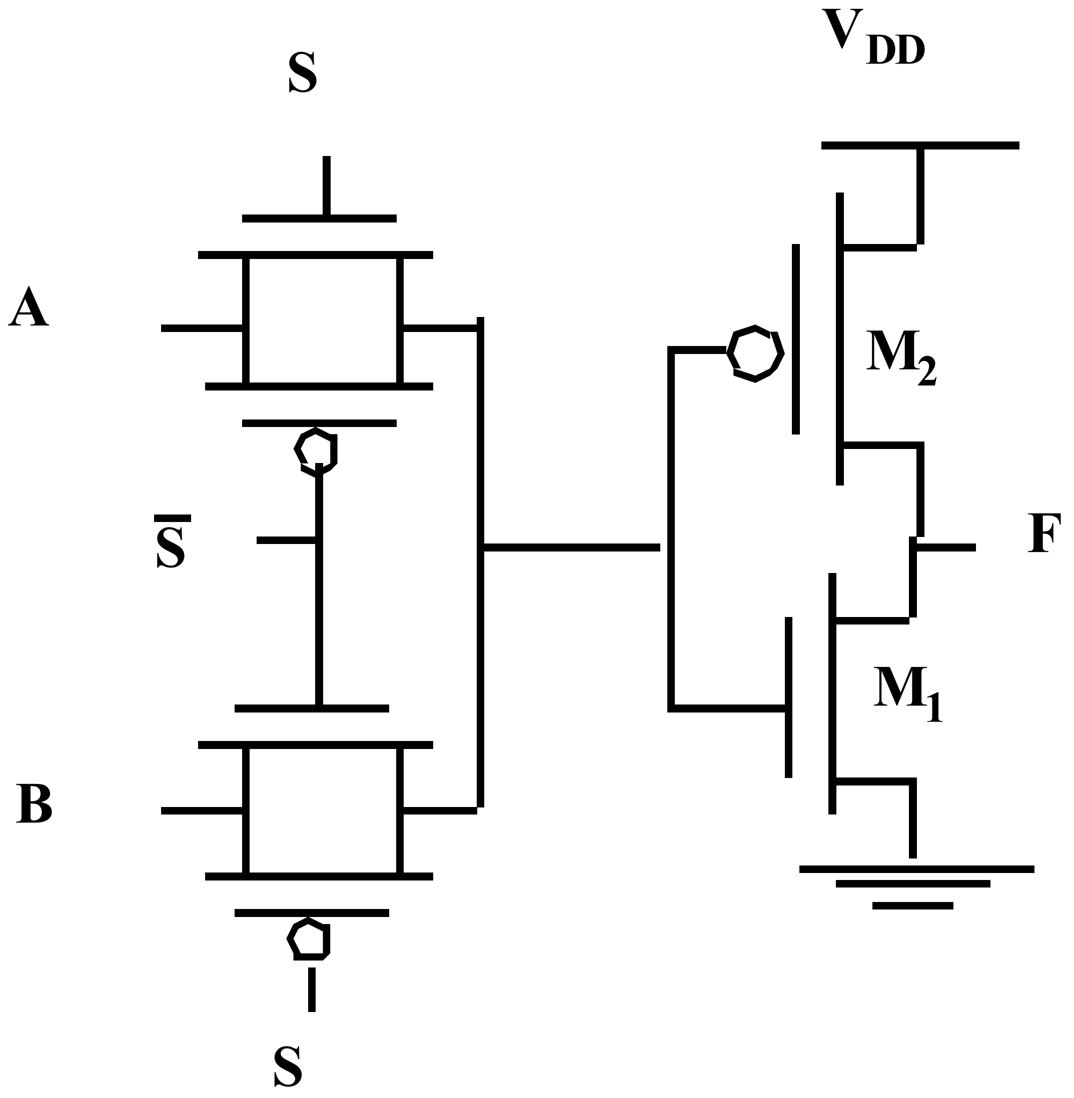


Resistance of Transmission Gate

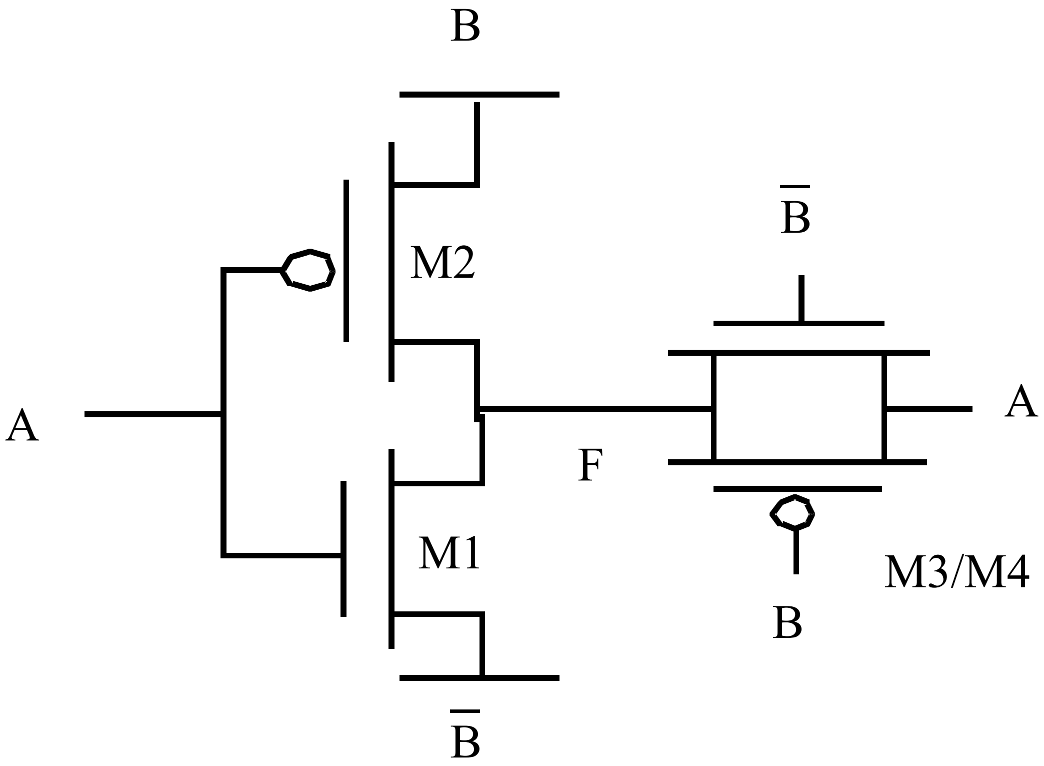




Pass-Transistor Based Multiplexer

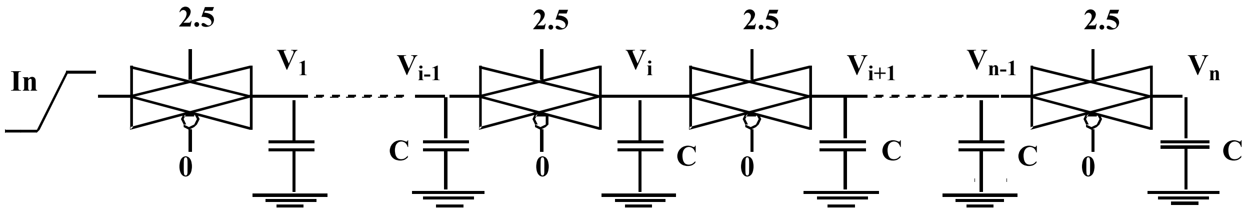


Transmission Gate XOR

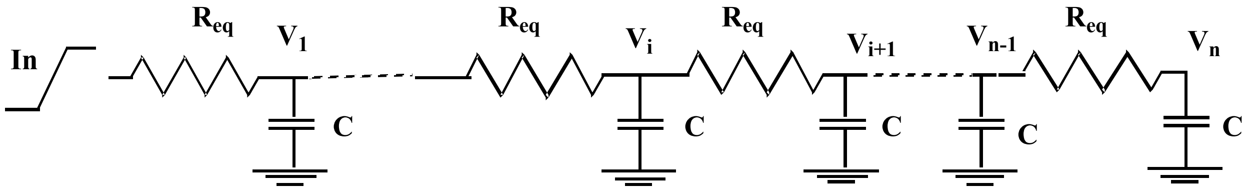


SUPPLEMENTARY

Delay in Transmission Gate Networks



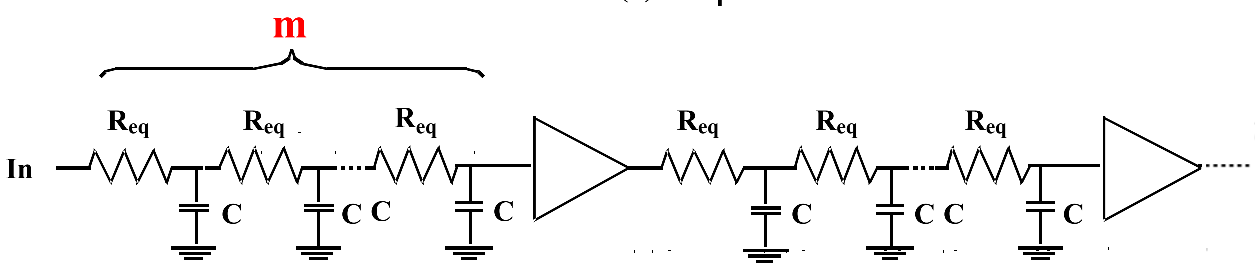
(a) A chain of transmission gates



$$t_p(V_n) = 0.69 \sum_{k=0}^{n} C R_{eq} k$$

$$= 0.69 C R_{eq} \frac{n(n+1)}{2}$$

(b) Equivalent RC network



$$t_p(V_n) = 0.69 C R_{eq} \frac{n(m+1)}{2} + t_{buf}\left(\frac{n}{m} - 1\right)$$

(c) Inserting buffers to reduce delay

NUS National University of Singapore

Delay Optimization

- **Delay of RC chain**

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

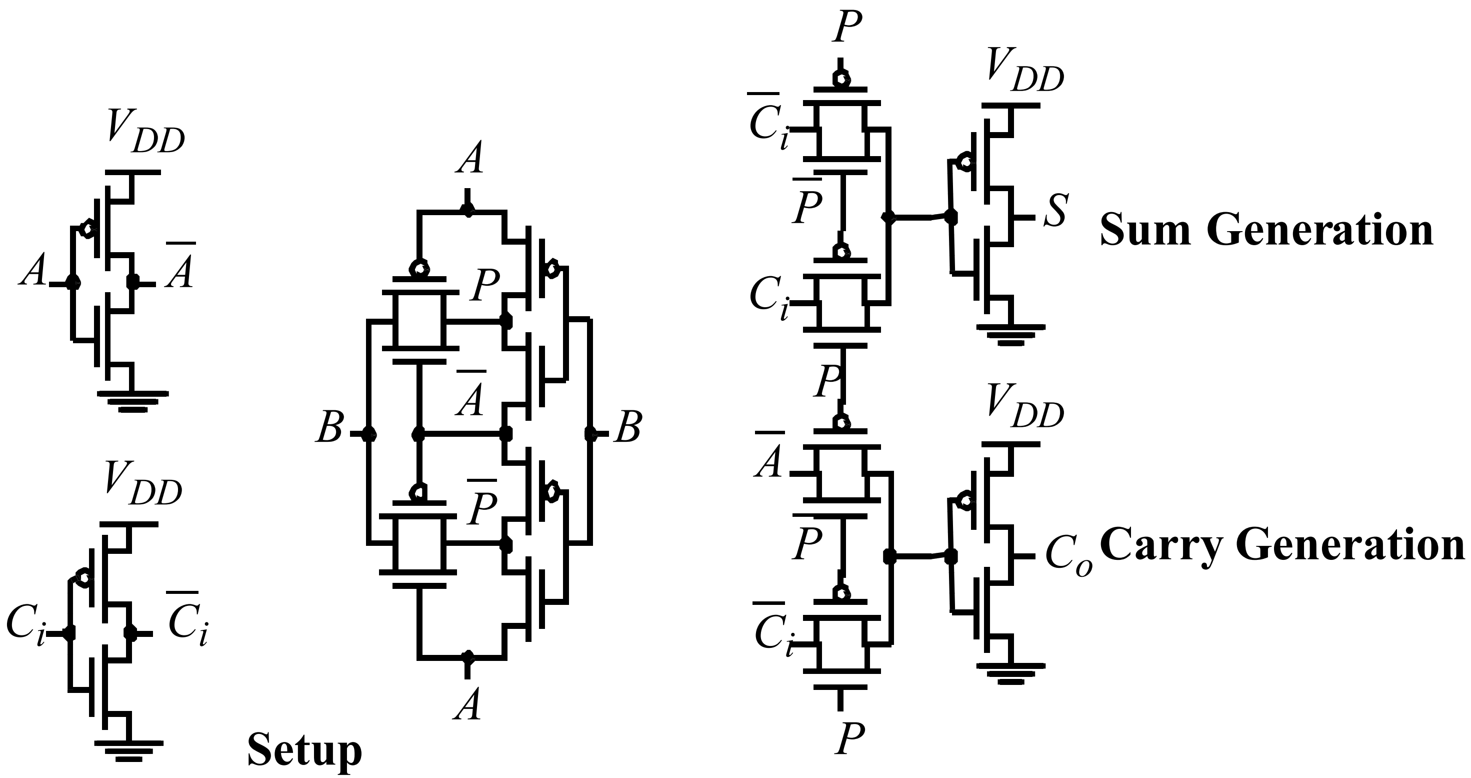
- **Delay of Buffered Chain**

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

$$= 0.69\left[CR_{eq} \frac{n(m+1)}{2}\right] + \left(\frac{n}{m} - 1\right) t_{buf} \qquad \sim \boldsymbol{n}$$

$\frac{\partial t_p}{\partial m} = ($ ➡ $m_{opt} = 1.7 \sqrt{\frac{t_{pbuf}}{CR_{eq}}}$

Transmission Gate Full Adder



Similar delays for sum and carry