EEC 116 Lecture #5: CMOS Logic

Rajeevan Amirtharajah Bevan Baas
University of California, Davis
Jeff Parkhurst
Intel Corporation

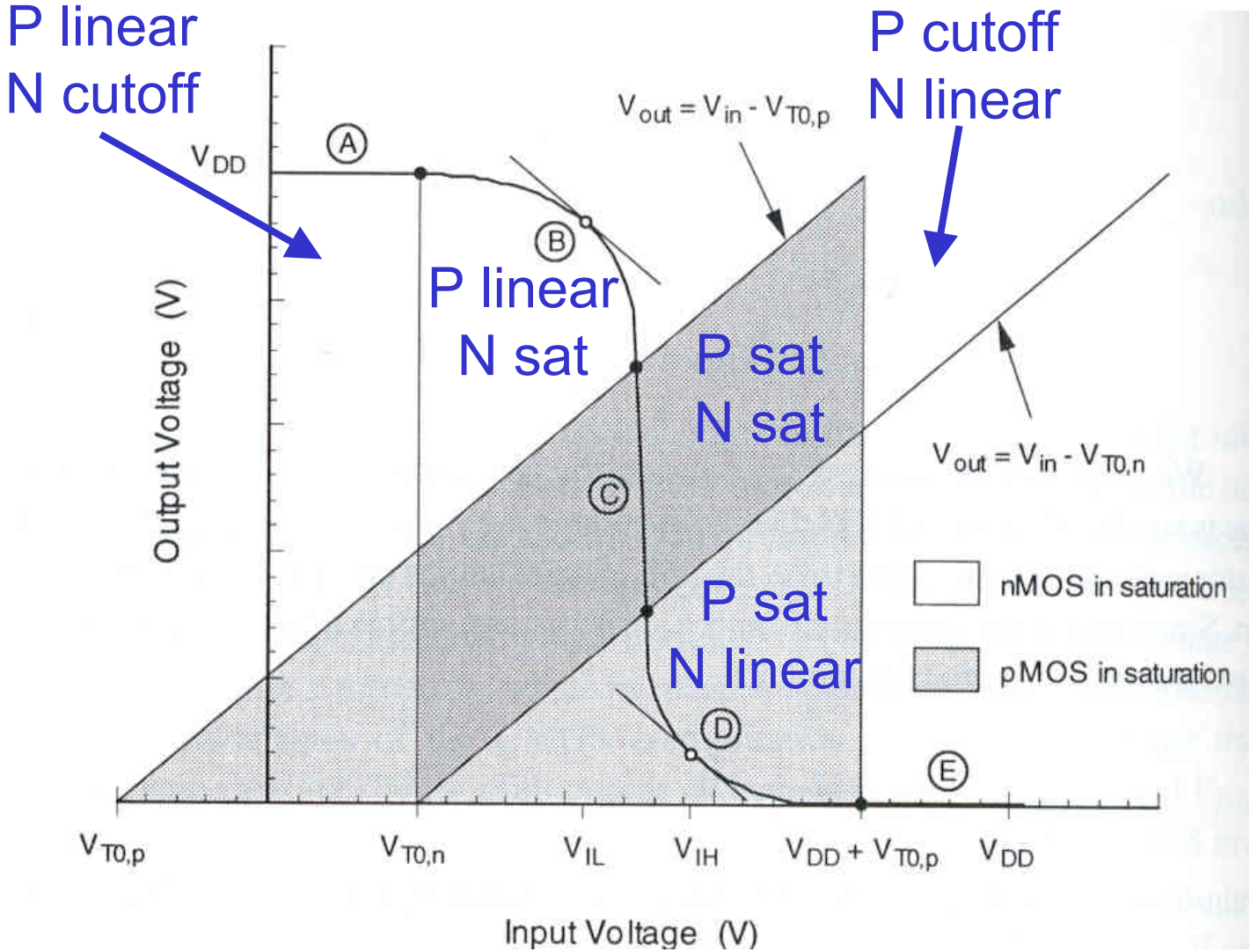
Announcements

- **Quiz 1 today!**
- **Lab 2 reports due this week**
- **Lab 3 this week**
- **HW 2 due this Wednesday at 4 PM in box, Kemper 2131**

Outline

- **Review: CMOS Inverter Transient Characteristics**
- **Review: Inverter Power Consumption**
- **Combinational MOS Logic Circuits: Rabaey 6.1-6.2 (Kang & Leblebici, 7.1-7.4)**
- **Combinational MOS Logic Transient Response**
  - AC Characteristics, Switch Model

Review: CMOS Inverter VTC



Review: Logic Circuit Delay

- **For CMOS (or almost all logic circuit families), only one fundamental equation necessary to determine delay:**

$$I = C\frac{dV}{dt}$$

- **Consider the discretized version:** $I = C\frac{\Delta V}{\Delta t}$
- **Rewrite to solve for delay:** $\Delta t = C\frac{\Delta V}{I}$
- **Only three ways to make faster logic: ↓C, ↓ΔV, ↑I**

Review: Inverter Delays

- **High-to-low and low-to-high transitions (exact):**

$$t_{PHL} = \frac{C_L}{k_n(V_{OH} - V_{T0,n})}\left[\frac{2V_{T0,n}}{V_{OH} - V_{T0,n}} + \ln\left(\frac{4(V_{OH} - V_{T0,n})}{V_{OH} + V_{OL}} - 1\right)\right]$$

$$t_{PLH} = \frac{C_L}{k_p(V_{OH} - V_{OL} - |V_{T0,p}|)}\left[\frac{2|V_{T0,p}|}{V_{OH} - V_{OL} - |V_{T0,p}|} + \ln\left(\frac{4(V_{OH} - V_{OL} - |V_{T0,p}|)}{V_{OH} + V_{OL}} - 1\right)\right]$$

- **Similar exact method to find rise and fall times**
- **Note: to balance rise and fall delays (assuming $V_{OH} = V_{DD}$, $V_{OL} = 0V$, and $V_{T0,n}=V_{T0,p}$) requires**

$$\frac{k_p}{k_n} = 1 \qquad \left(\frac{W}{L}\right)_p \Big/ \left(\frac{W}{L}\right)_n = \frac{\mu_n}{\mu_p} \approx 2.5$$

Review: Inverter Power Consumption

- **Static power consumption (ideal) = 0**
  - Actually DIBL (Drain-Induced Barrier Lowering), gate leakage, junction leakage are still present
- **Dynamic power consumption**

$$P_{avg} = \frac{1}{T}\int_0^T v(t)i(t)dt$$

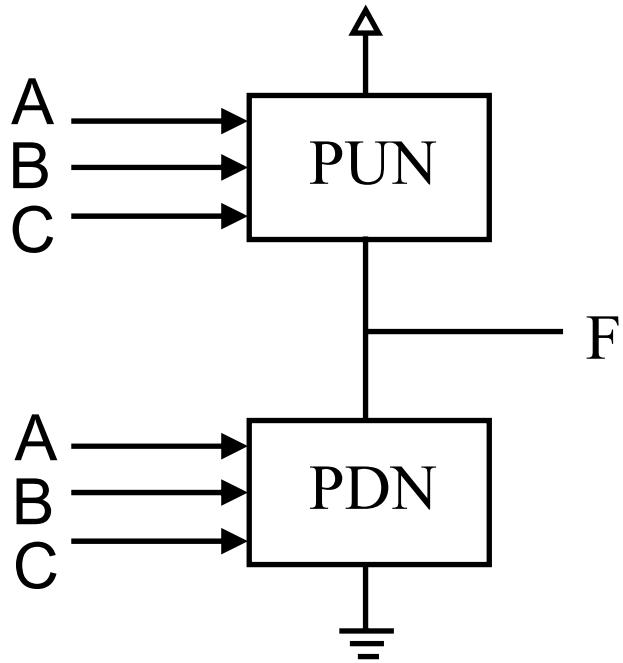
$$P_{avg} = \frac{1}{T}\left[\int_0^{T/2} V_{out}\left(-C_{load}\frac{dV_{out}}{dt}\right)dt + \int_{T/2}^{T}(V_{DD}-V_{out})\left(C_{load}\frac{dV_{out}}{dt}\right)dt\right]$$

$$P_{avg} = \frac{1}{T}\left[\left(-C_{load}\frac{{V_{out}}^2}{2}\right)\Bigg|_0^{T/2} + \left(V_{DD}V_{out}C_{load} - \frac{1}{2}C_{load}{V_{out}}^2\right)\Bigg|_{T/2}^{T}\right]$$

$$P_{avg} = \frac{1}{T}C_{load}V_{DD}^2 = C_{load}V_{DD}^2 f$$

Static CMOS

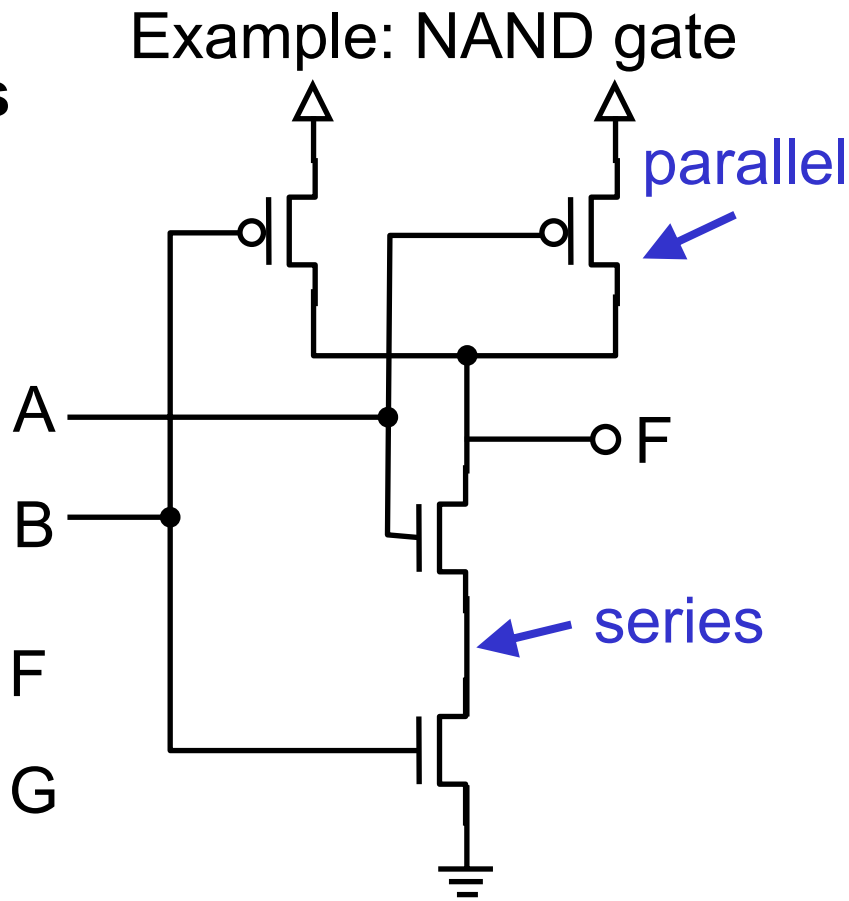
- **Complementary pullup network (PUN) and pulldown network (PDN)**
- **Only <u>one</u> network is on at a time**
- **PUN: PMOS devices**
  - Why?
- **PDN: NMOS devices**
  - Why?
- **PUN and PDN are *dual* networks**



Dual Networks

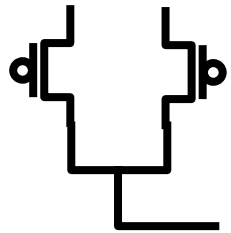
- **Dual networks: parallel connection in PDN = series connection in PUN, vice-versa**

- **If CMOS gate implements logic function F:**
  - PUN implements function F
  - PDN implements function G = $\overline{F}$

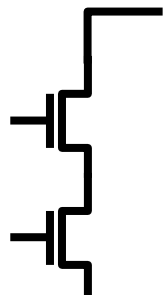


NAND Gate

- **NAND function: $F = \overline{A \bullet B}$**



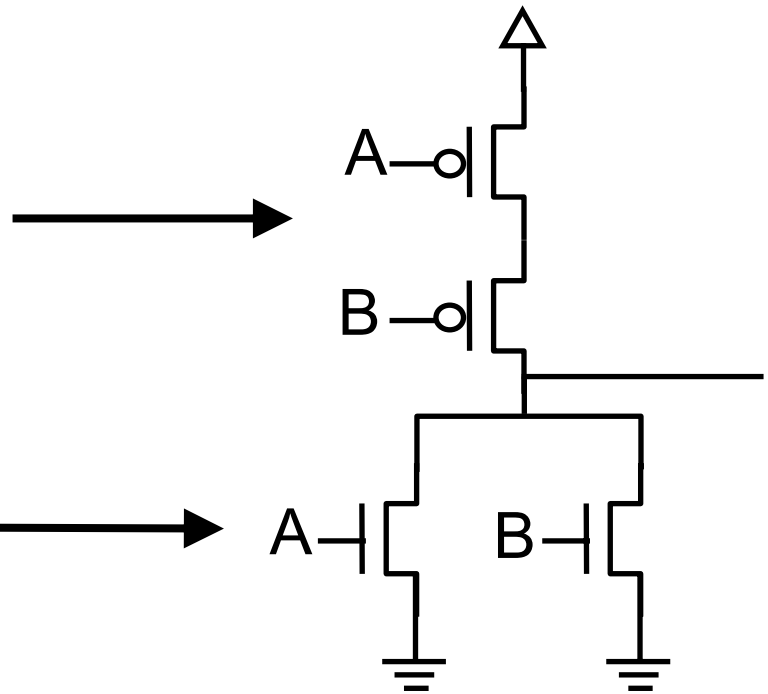
- **PUN function: $F = \overline{A \bullet B} = \overline{A} + \overline{B}$**
  - "Or" function (+) → parallel connection
  - Inverted inputs $\overline{A}$, $\overline{B}$ → PMOS transistors



- **PDN function: $G = \overline{F} = A \bullet B$**
  - "And" function (•) → series connection
  - Non-inverted inputs → NMOS transistors

NOR Gate

- **NOR gate operation: $F = \overline{A+B}$**
- PUN: $F = \overline{A+B} = \overline{A} \bullet \overline{B}$
- PDN: $G = \overline{F} = A+B$



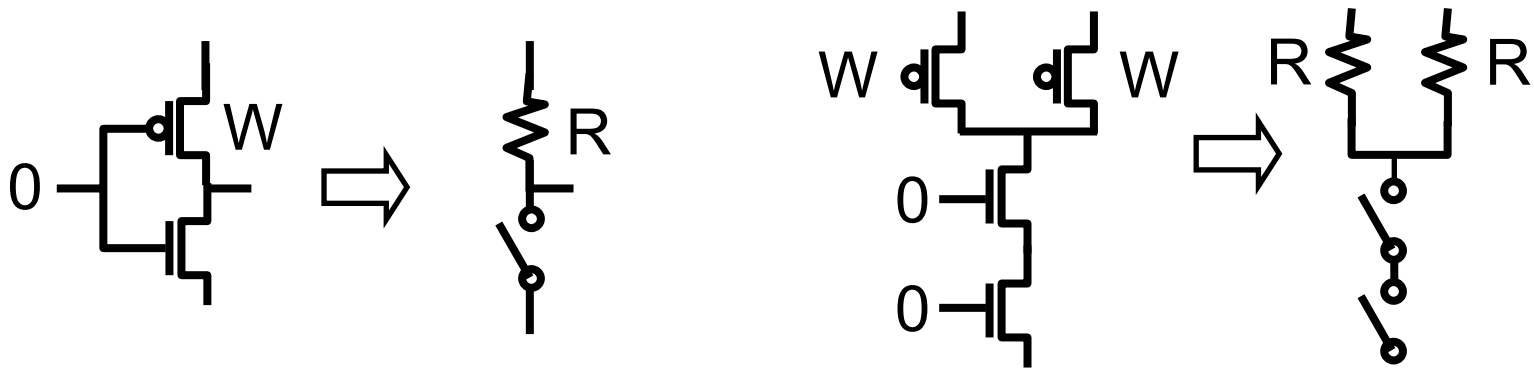
Analysis of CMOS Gates

- **Represent "on" transistors as resistors**

- Transistors in series → resistances in series
  - Effective resistance = 2R
  - Effective length = 2L

Analysis of CMOS Gates (cont.)

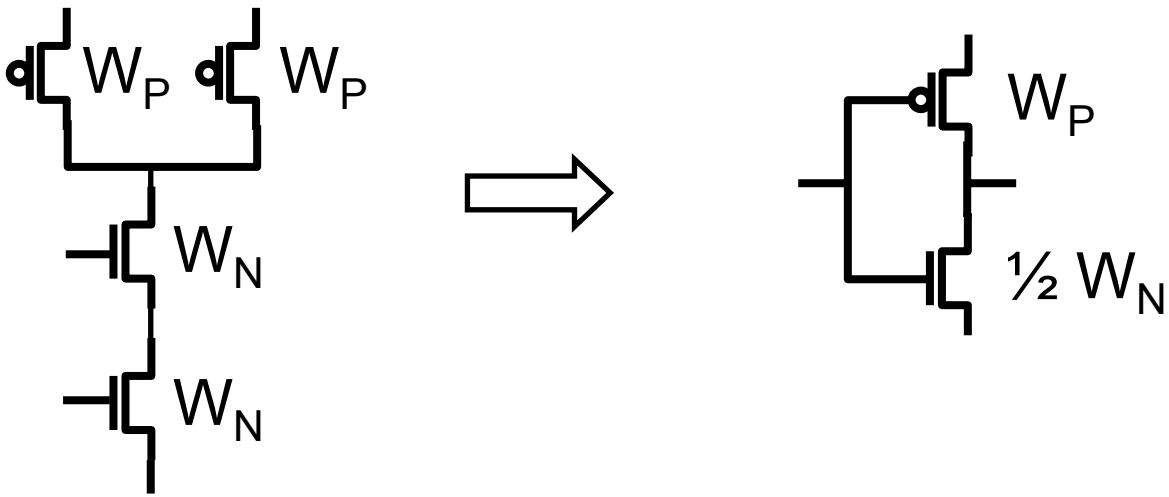
- Represent “on” transistors as resistors



- Transistors in parallel → resistances in parallel
  - Effective resistance = ½ R
  - Effective width = 2W

CMOS Gates: Equivalent Inverter

- **Represent complex gate as inverter for delay estimation**
- **Typically use worst-case delays**
- **Example: NAND gate**
  - Worst-case (slowest) pull-up: only 1 PMOS “on”
  - Pull-down: both NMOS “on”



Example: Complex Gate

Design CMOS gate for this truth table:

A	B	C	F
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	0

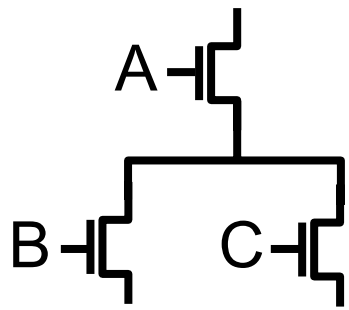
$$F = \overline{A \bullet (B+C)}$$

Example: Complex Gate

Design CMOS gate for this logic function:

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

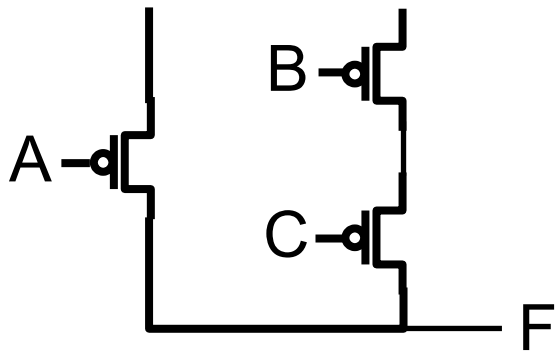
1. Find NMOS pulldown network diagram:
   $G = \overline{F} = A \bullet (B+C)$



Not a unique solution: can exchange order of series connection

Example: Complex Gate

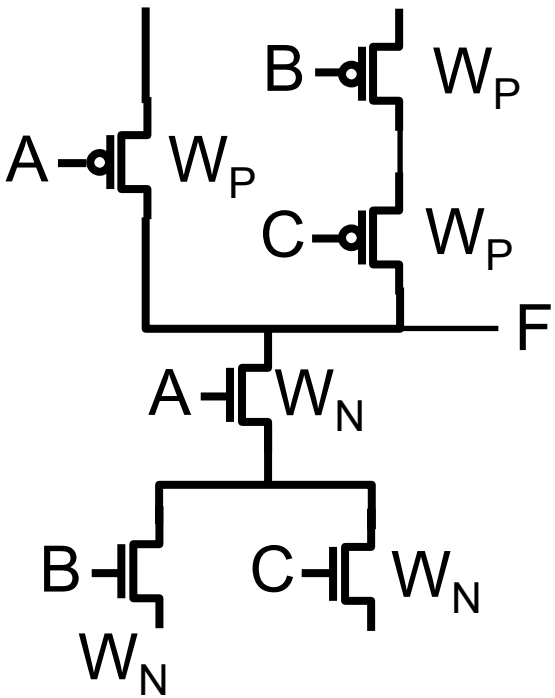
2. Find PMOS pullup network diagram: F = $\overline{A}+(\overline{B}\bullet\overline{C})$



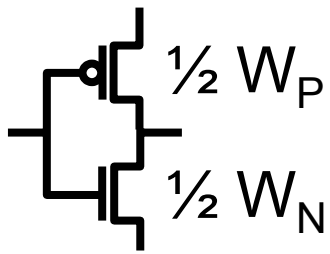
Not a unique solution: can exchange order of series connection (B and C inputs)

Example: Complex Gate

Completed gate:



- What is worse-case pullup delay?
- What is worse-case pulldown delay?
- Effective inverter for delay calculation:



CMOS Gate Design

- **Designing a CMOS gate:**
  - Find pulldown NMOS network from logic function or by inspection
  - Find pullup PMOS network
    - By inspection
    - Using logic function
    - Using dual network approach
  - Size transistors using equivalent inverter
    - Find worst-case pullup and pulldown paths
    - Size to meet rise/fall or threshold requirements

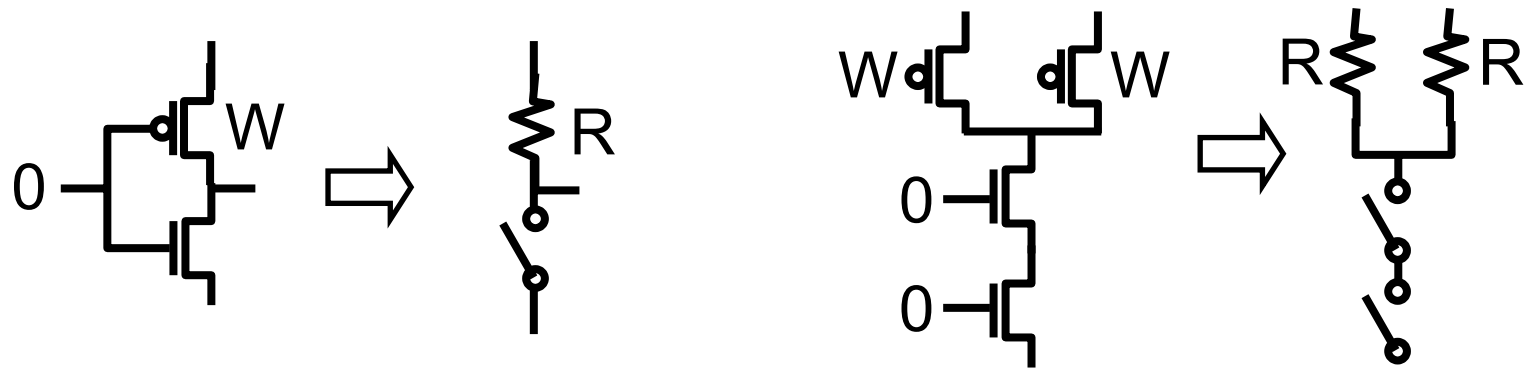
Analysis of CMOS gates

- **Represent "on" transistors as resistors**

- Transistors in series → resistances in series
  - Effective resistance = 2R
  - Effective width = ½ W (equivalent to 2L)
  - Typically use minimum length devices ($L = L_{min}$)

Analysis of CMOS Gates (cont.)

- Represent “on” transistors as resistors



- Transistors in parallel → resistances in parallel
  - Effective resistance = ½ R
  - Effective width = 2W
  - Typically use minimum length devices (L = Lmin)

Equivalent Inverter

- **CMOS gates: many paths to Vdd and Gnd**
  - Multiple values for V_M, V_{IL}, V_{IH}, etc
  - Different delays for each input combination
- **Equivalent inverter**
  - Represent each gate as an inverter with appropriate device width
  - Include only transistors which are on or switching
  - Calculate V_M, delays, etc using inverter equations

Static CMOS Logic Characteristics

- **For V_M, the V_M of the equivalent inverter is used (assumes all inputs are tied together)**
  - For specific input patterns, V_M will be different
- **For V_{IL} and V_{IH}, only the worst case is interesting since circuits must be designed for worst-case noise margin**
- **For delays, both the maximum and minimum must be accounted for in race analysis**

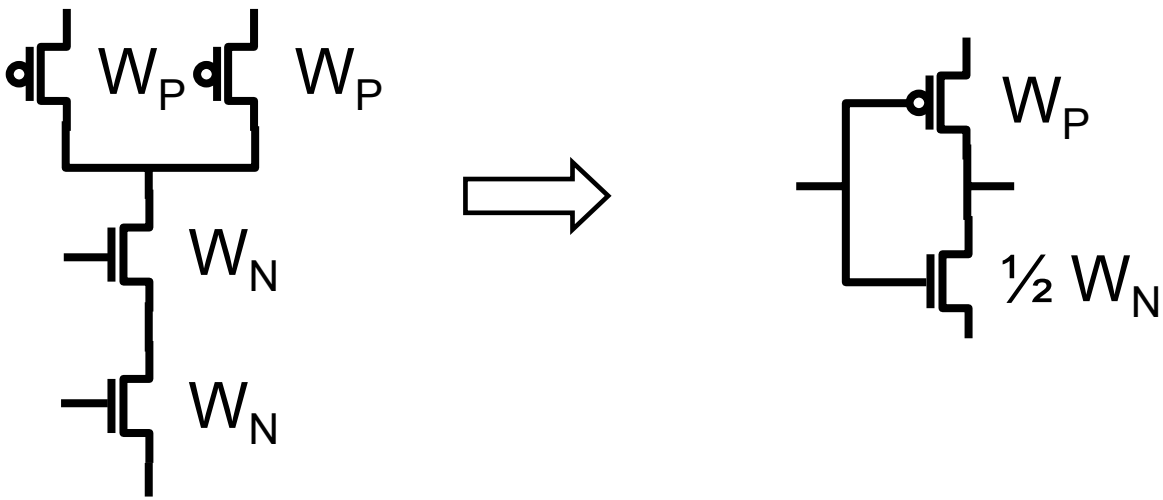
Equivalent Inverter: V_M

- **Example: NAND gate threshold V_M**
  Three possibilities:
  - A & B switch together
  - A switches alone
  - B switches alone

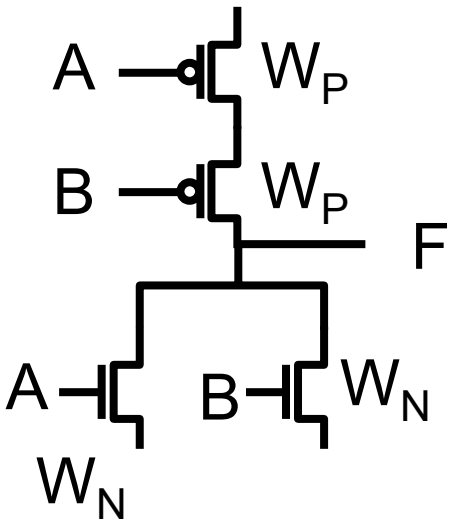
- **What is equivalent inverter for each case?**

Equivalent Inverter: Delay

- **Represent complex gate as inverter for delay estimation**
- **Use worse-case delays**
- **Example: NAND gate**
  - Worse-case (slowest) pull-up: only 1 PMOS “on”
  - Pull-down: both NMOS “on”



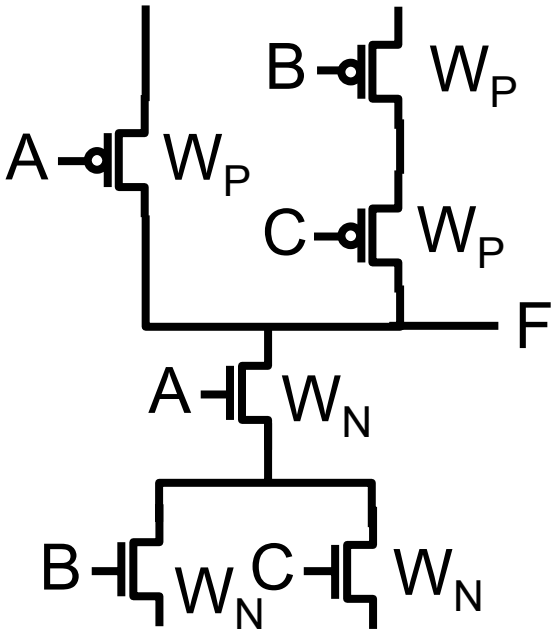
Example: NOR gate



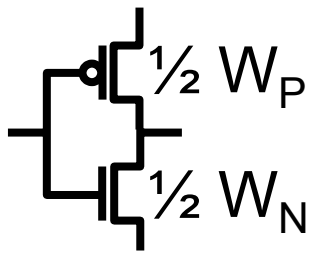
- **Find threshold voltage V_M when both inputs switch simultaneously**
- **Two methods:**
  - Transistor equations (complex)
  - Equivalent inverter
  - Should get same answer

Example: Complex Gate

Completed gate:



- What is worse-case pullup delay?
- What is worse-case pulldown delay?
- Effective inverter for delay calculation:



Transistor Sizing

- **Sizing for switching threshold**
  - All inputs switch together
- **Sizing for delay**
  - Find worst-case input combination
- **Find equivalent inverter, use inverter analysis to set device sizes**

Common CMOS Gate Topologies

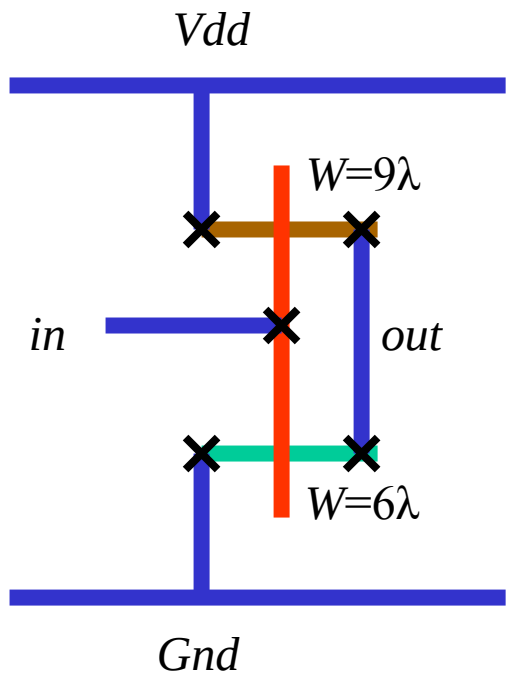
- **And-Or-Invert (AOI)**
  - Sum of products boolean function
  - Parallel branches of series connected NMOS
- **Or-And-Invert (OAI)**
  - Product of sums boolean function
  - Series connection of sets of parallel NMOS

Stick Diagrams

- **Dimensionless layout sketches**
- **Only topology is important**
- **Two primary uses**
  - Useful intermediate step
    - Transistor schematic is the first step
    - Layout is the last step
  - Final layout generated automatically by “compaction” program
    - Not widely used; a topic of research
- **Use colored pencils or pens whose colors match Cadence layer colors**

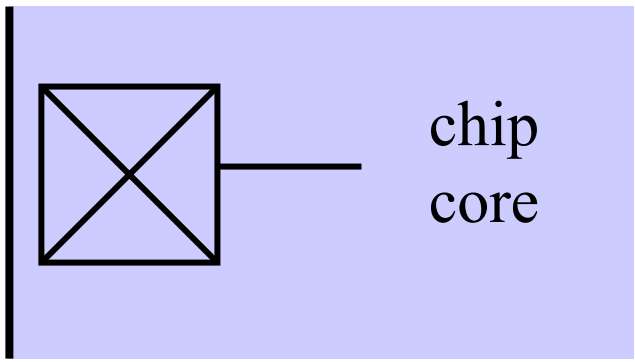
Inverter Stick Diagram

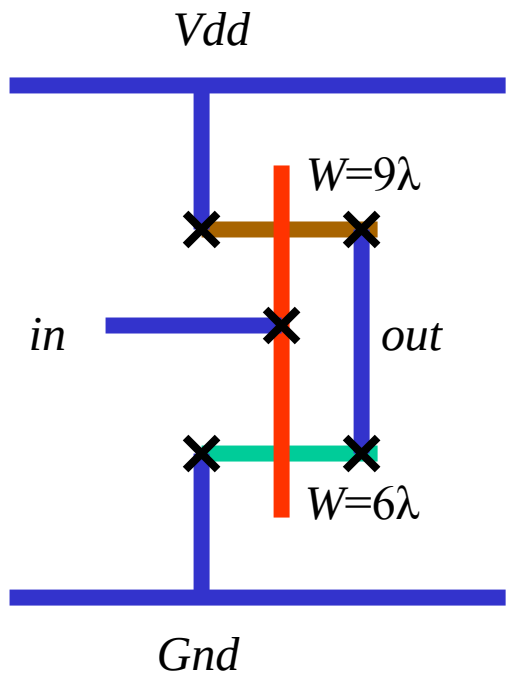
- **Diagram here uses magic standard color scheme**
- **Label all nodes**
- **Transistor widths (*W*) often shown—with varying units**
  - Often in λ in this class
  - Also nm or μm
  - Sometimes as a unit-less ratio—this stick diagram could also say the PMOS is 1.5x wider than the NMOS (saying "1" and "1.5" instead of "6λ" and "9λ"



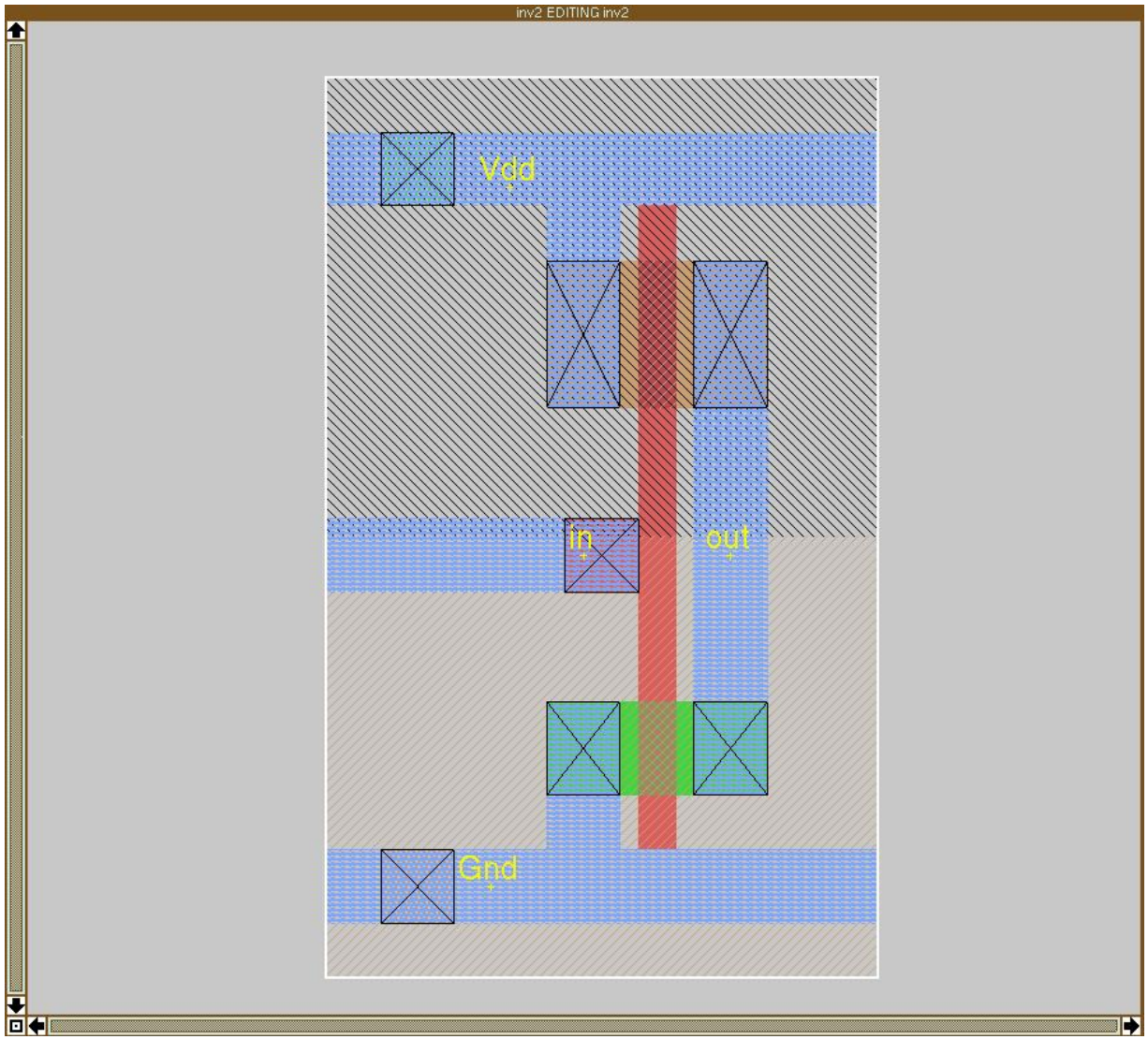
Stick Diagrams

- **Can also draw contacts with an "X"**
- **Do not confuse this "X" with the chip I/O and power pads on the edge of chip (shown with a box with an "X") or any other markers**





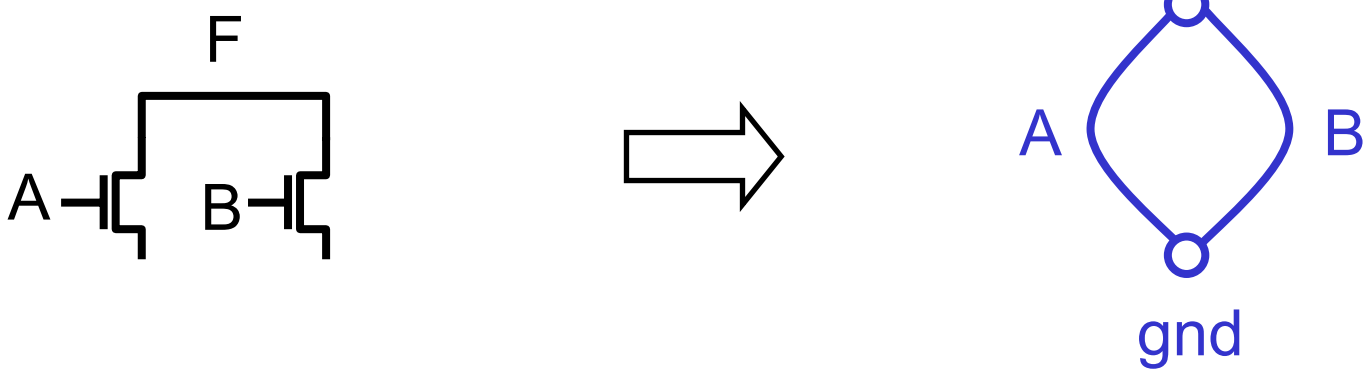
Layout for the Inverter in the Stick Diagram



Source: Omar Sattari

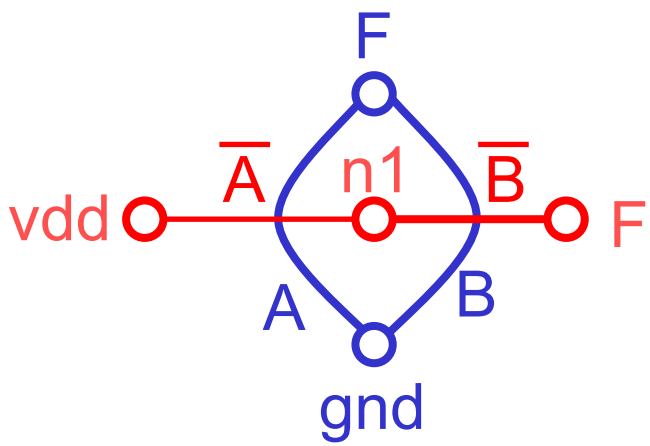
Graph-Based Dual Network

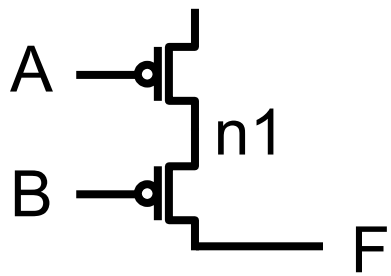
- **Use graph theory to help design gates**
  - Mostly implemented in CAD tools
- **Draw network for PUN or PDN**
  - Circuit nodes are vertices
  - Transistors are edges



Graph-Based Dual Network (2)

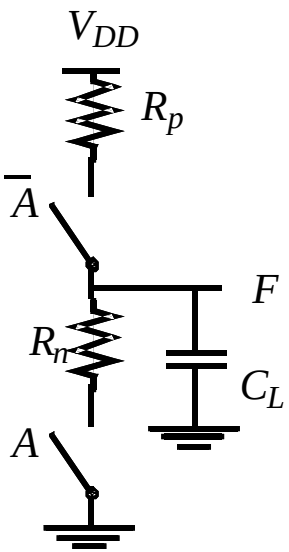
- **To derive dual network:**
  - Create new node in each enclosed region of graph
  - Draw new edge intersecting each original edge
  - Edge is controlled by inverted input



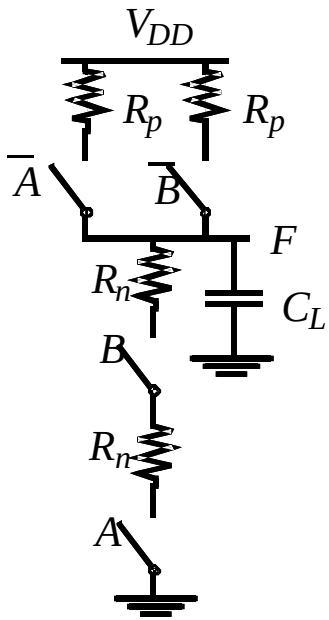


  - Convert to layout using consistent Euler paths

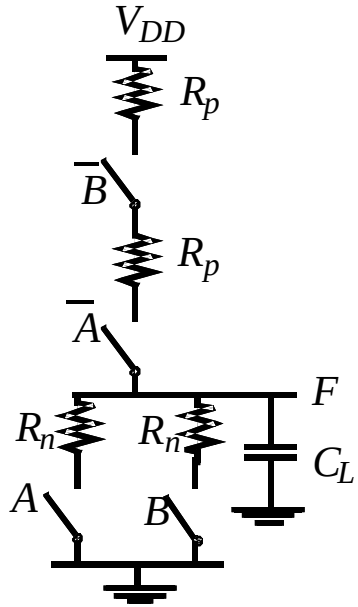
Propagation Delay Analysis - The Switch Model



(a) Inverter



(b) 2-input NAND



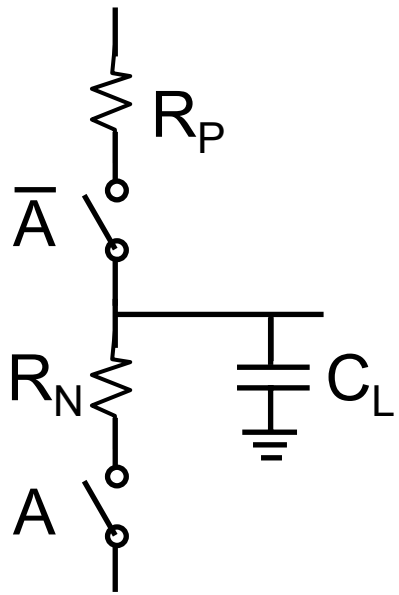
(c) 2-input NOR

$$t_p = 0.69\ R_{on}\ C_L$$

(assuming that C_L dominates!)

Switch Level Model

- **Model transistors as switches with series resistance**
- **Resistance R_{on} = average resistance for a transition**
- **Capacitance C_L = average load capacitance for a transition (same as we analyzed for transient inverter delays)**



What is the Value of R_{on}?

- *Depends strongly on the operating region*
- *For hand analysis use a fixed value of R which it the average of the two end points of the transition*
- *Similar to the previous approach of averaging currents*

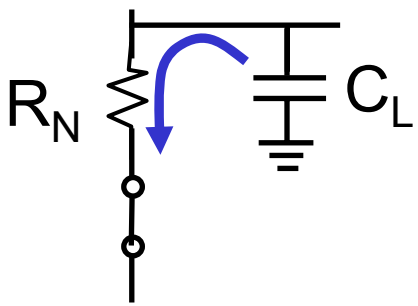
EXAMPLE: For t_{pHL} for an inverter, the R_{on} is:

$$R_{on} = \frac{1}{2}(R_{NMOS}(V_{out} = V_{DD}) + R_{NMOS}(V_{out} = V_{DD}/2))$$

$$= \frac{1}{2}\left(\left(\frac{V_{DS}}{I_D}\right)_{V_{out} = V_{DD}} + \left(\frac{V_{DS}}{I_D}\right)_{V_{out} = V_{DD}/2}\right)$$

Switch Level Model Delays

Delay estimation using switch-level model (for general RC circuit):



$$I = C\frac{dV}{dt} \quad \rightarrow \quad dt = \frac{C}{I}dV$$

$$I = \frac{V}{R} \quad \rightarrow \quad dt = \frac{RC}{V}dV$$

$$t_1 - t_0 = t_p = \int_{V_0}^{V_1} \frac{RC}{V}dV$$

$$t_p = RC\left[\ln(V_1) - \ln(V_0)\right] = RC\ln\left(\frac{V_1}{V_0}\right)$$

Switch Level Model RC Delays

- **For fall delay t_{phl}, $V_0=V_{DD}$, $V_1=V_{DD}/2$**

$$t_p = RC\ln\left(\frac{V_1}{V_0}\right) = RC\ln\left(\frac{\frac{1}{2}V_{DD}}{V_{DD}}\right)$$

$$t_p = RC\ln(0.5)$$

$$t_{phl} = 0.69R_nC_L$$

$$t_{plh} = 0.69R_pC_L$$

← Standard RC-delay equations from literature

Numerical Examples

- **Example resistances for 1.2 μm CMOS**

$V_{DD} = 5V$, $W/Leff=1$ ($W/L_{eff}=2$ *is a minimum sized device 1.8μm/0.9μm*)
$L_{eff} = 1.2\mu m - 2(.15\mu m) = 0.9\mu m$

$R_n(W/L_{eff}= 2) = (5\ V / 0.46\ mA + 2.5\ V / 0.29\ mA) / 2 = 9.7\ k\Omega$ (for t_{pHL})

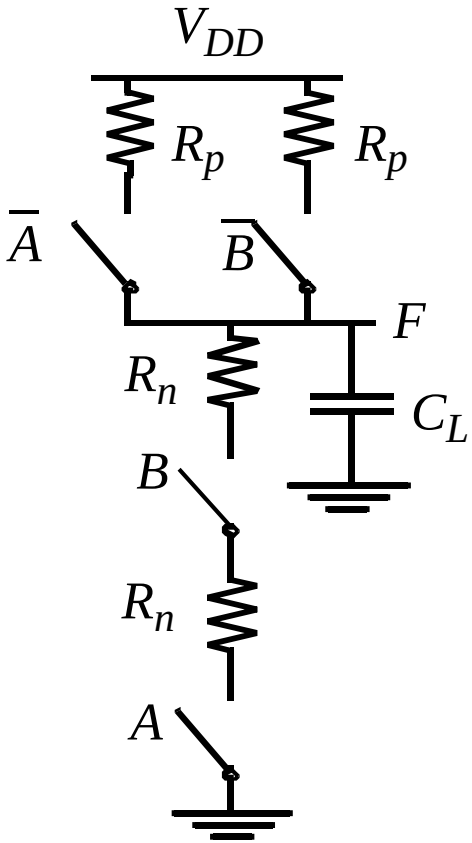
$R_n(W/L_{eff}= 1) = 9.7 * 2 = 19.4\ k\Omega$ (for t_{pHL})

$R_p(W/L_{eff}= 6) = (5\ V / 0.57\ mA + 2.5\ V / 0.24\ mA) / 2 = 9.6\ k\Omega$ (for t_{pLH})

$R_p(W/L_{eff}= 1) = 9.6 * 6 = 57.6\ k\Omega$ (for t_{pLH})

SOLVE RC NETWORK TO DETERMINE DELAYS

Analysis of Propagation Delay



2-input NAND

1. Assume R_n=R_p= resistance of minimum sized NMOS inverter

2. Determine "Worst Case Input" transition (Delay depends on input values)

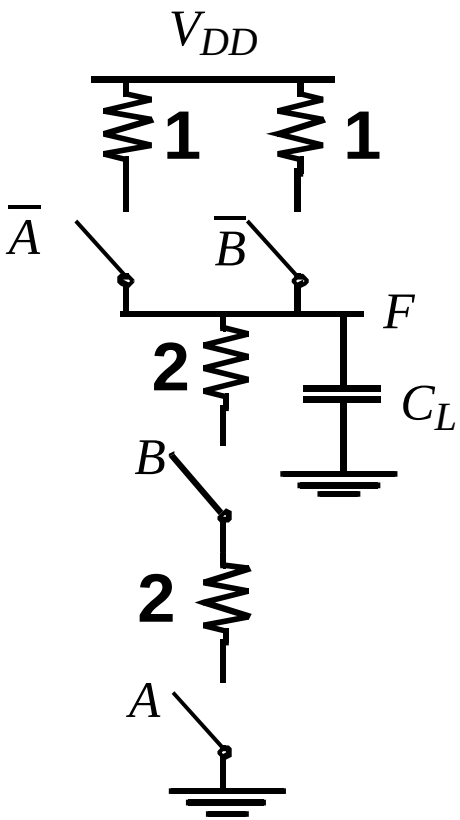
3. Example: t_{pLH} for 2input NAND
   - Worst case when only ONE PMOS Pulls up the output node
   - For 2 PMOS devices in parallel, the resistance is lower

$$t_{pLH} = 0.69 R_p C_L$$

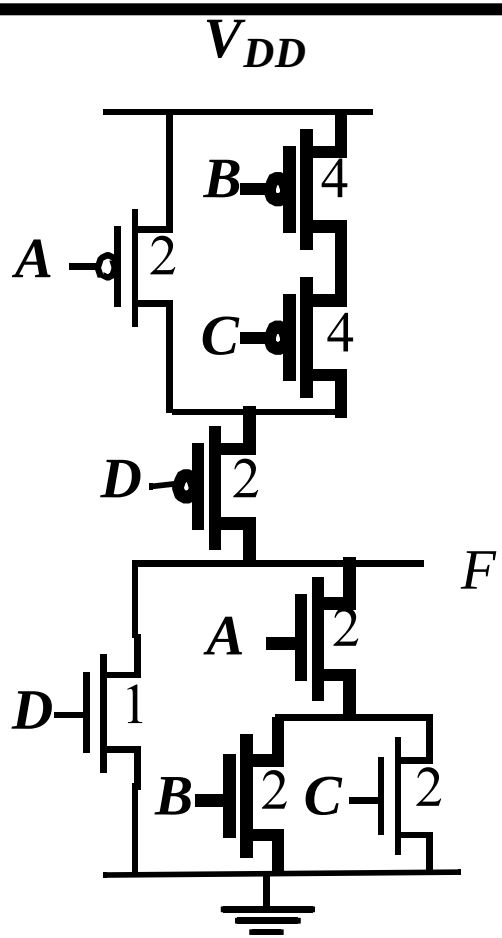
4. Example: t_{pHL} for 2input NAND
   - Worst case : TWO NMOS in series

$$t_{pHL} = 0.69(2R_n) C_L$$

Design for Worst Case



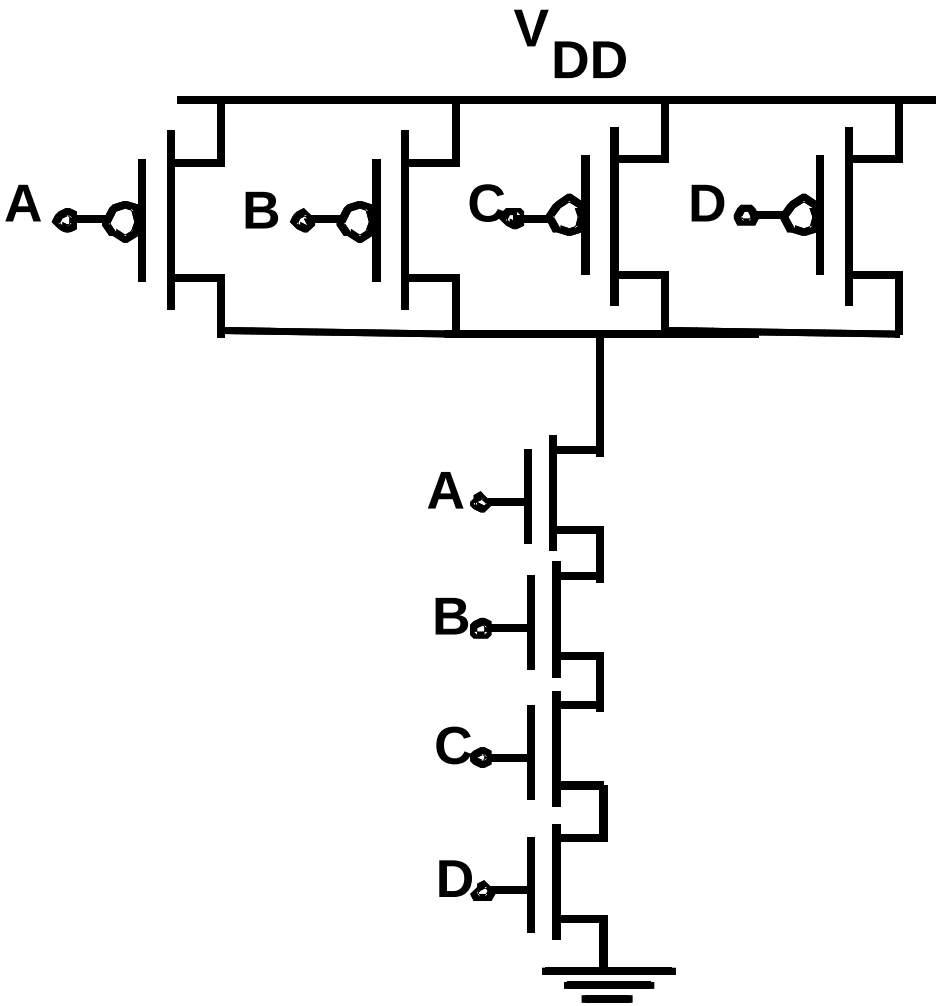
NAND Gate



Complex Gate

Here it is assumed that $R_p = R_n$

Fan-In and Fan-Out



Fan-Out

Number of logic gates connected to output

(2 FET gate capacitances per fan-out)

Fan-In

Number of logical inputs

Quadratic delay term due to:

1. Resistance increasing
2. Capacitance increasing

for t_{pHL} (series NMOS)

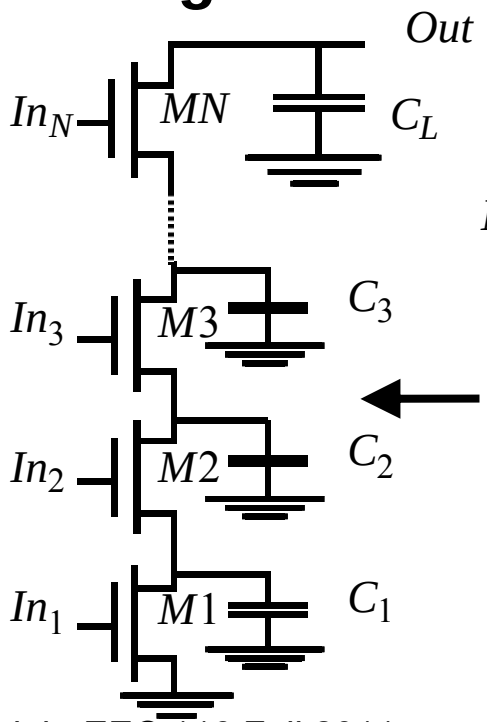
t_p proportional to $a_1FI + a_2FI^2 + a_3FO$

Fast Complex Gates - Design Techniques

- **Increase Transistor Sizing:**

  Works as long as Fan-out capacitance dominates self capacitance (S/D cap increases with increased width)

- **Progressive Sizing:**



$$M1 > M2 > M3 > MN$$

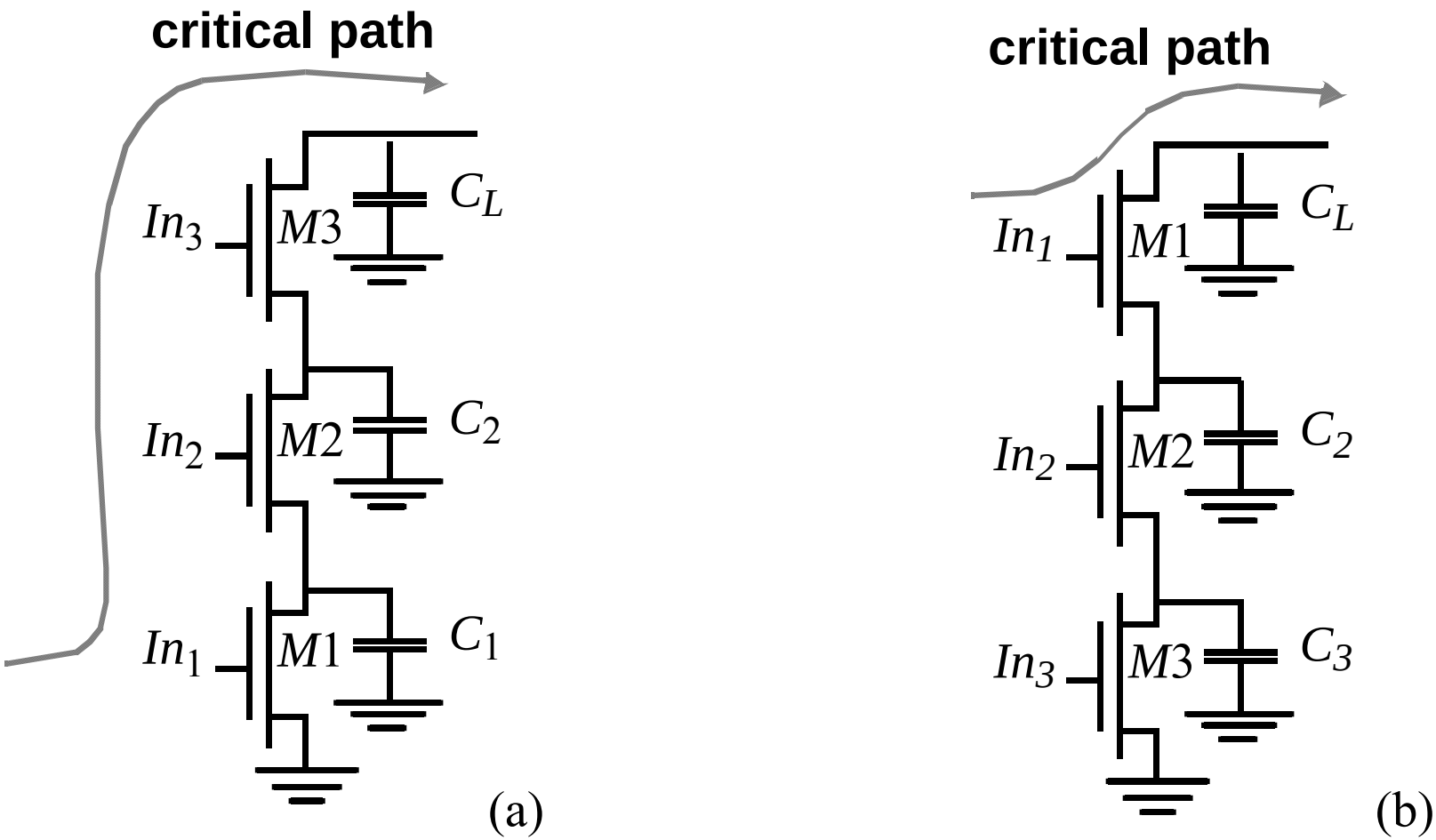
Distributed RC-line

Can Reduce Delay by more than 30%!

Fast Complex Gates - Design Techniques (2)

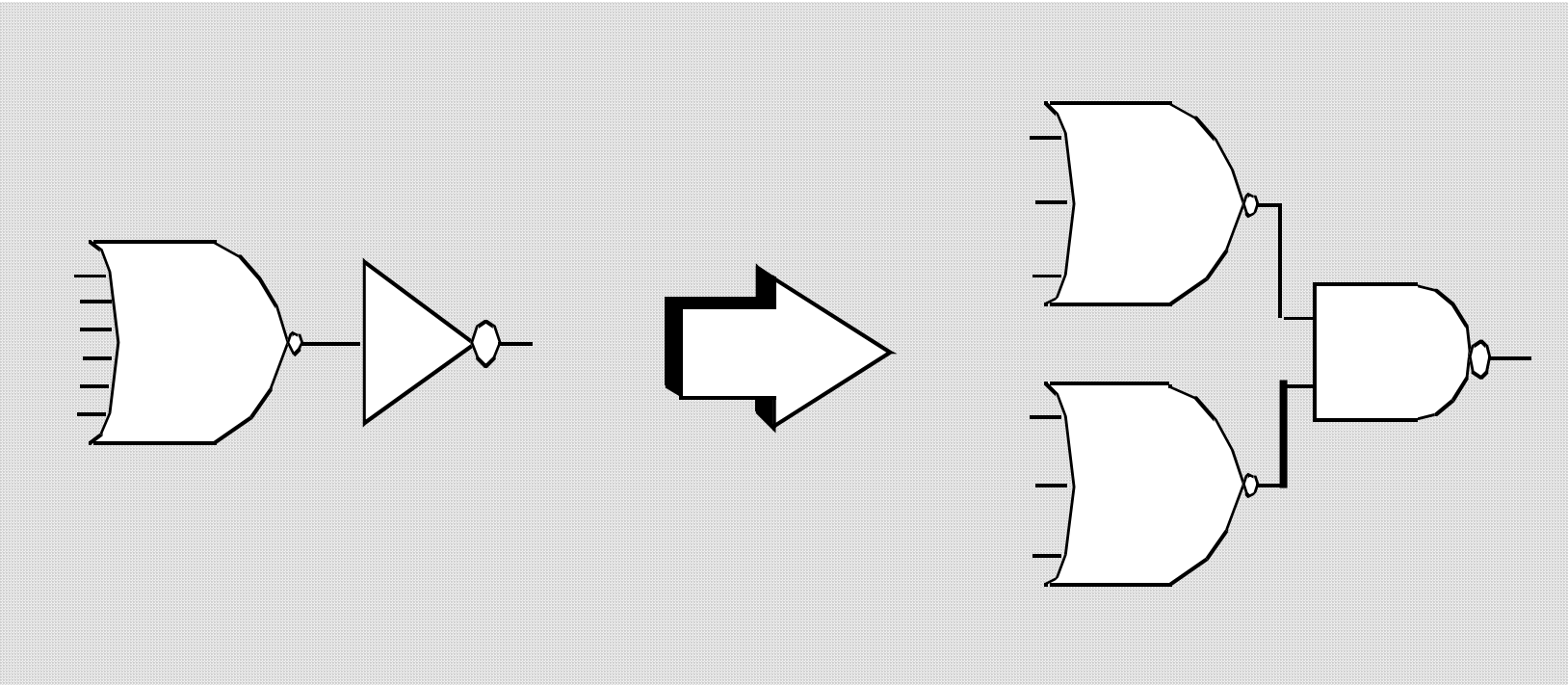
- **Transistor Ordering**

  Place last arriving input closest to output node



Fast Complex Gates - Design Techniques (3)

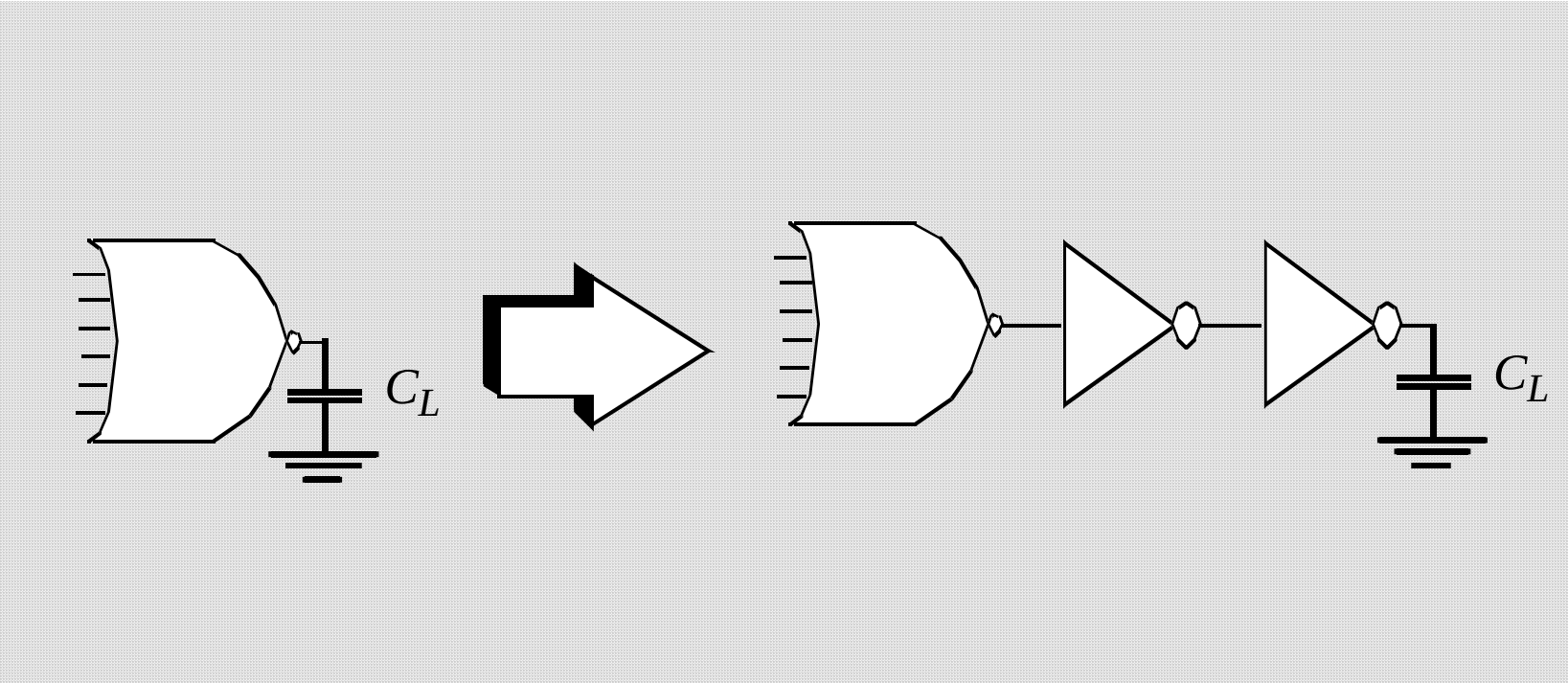
- **Improved Logic Design**



Note Fan-Out capacitance is the same, but Fan-In resistance lower for input gates (fewer series FETs)

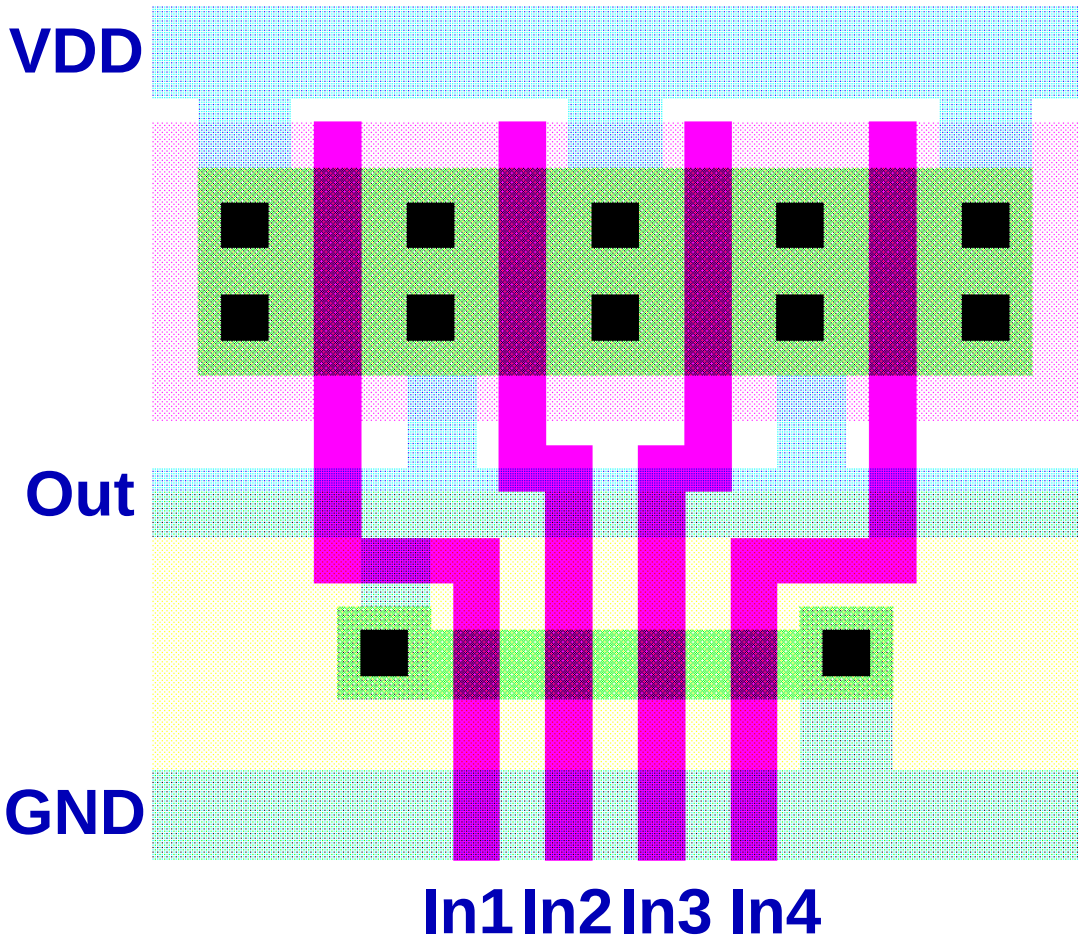
Fast Complex Gates - Design Techniques (4)

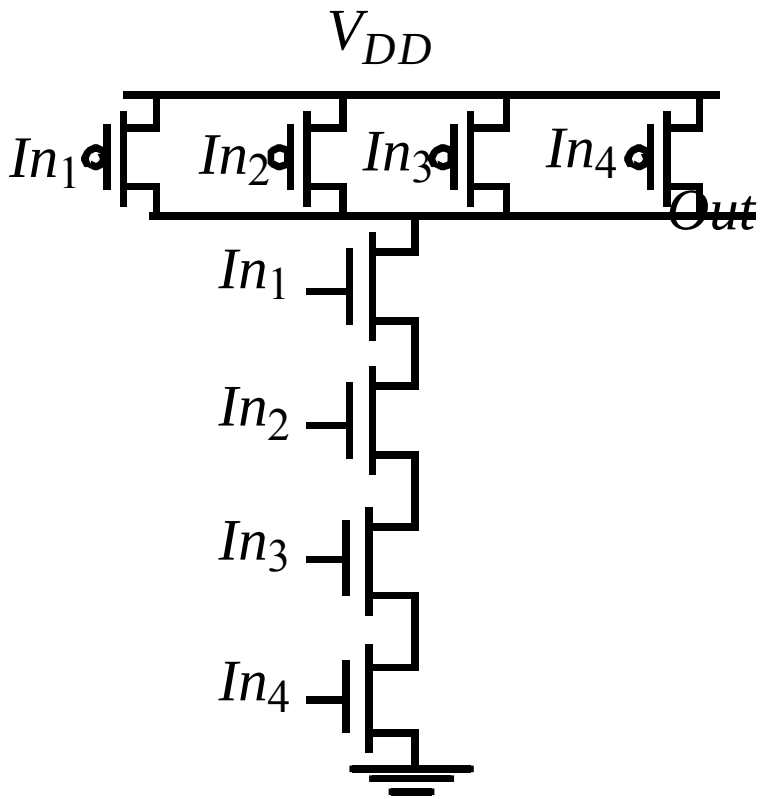
- **Buffering: Isolate Fan-in from Fan-out**



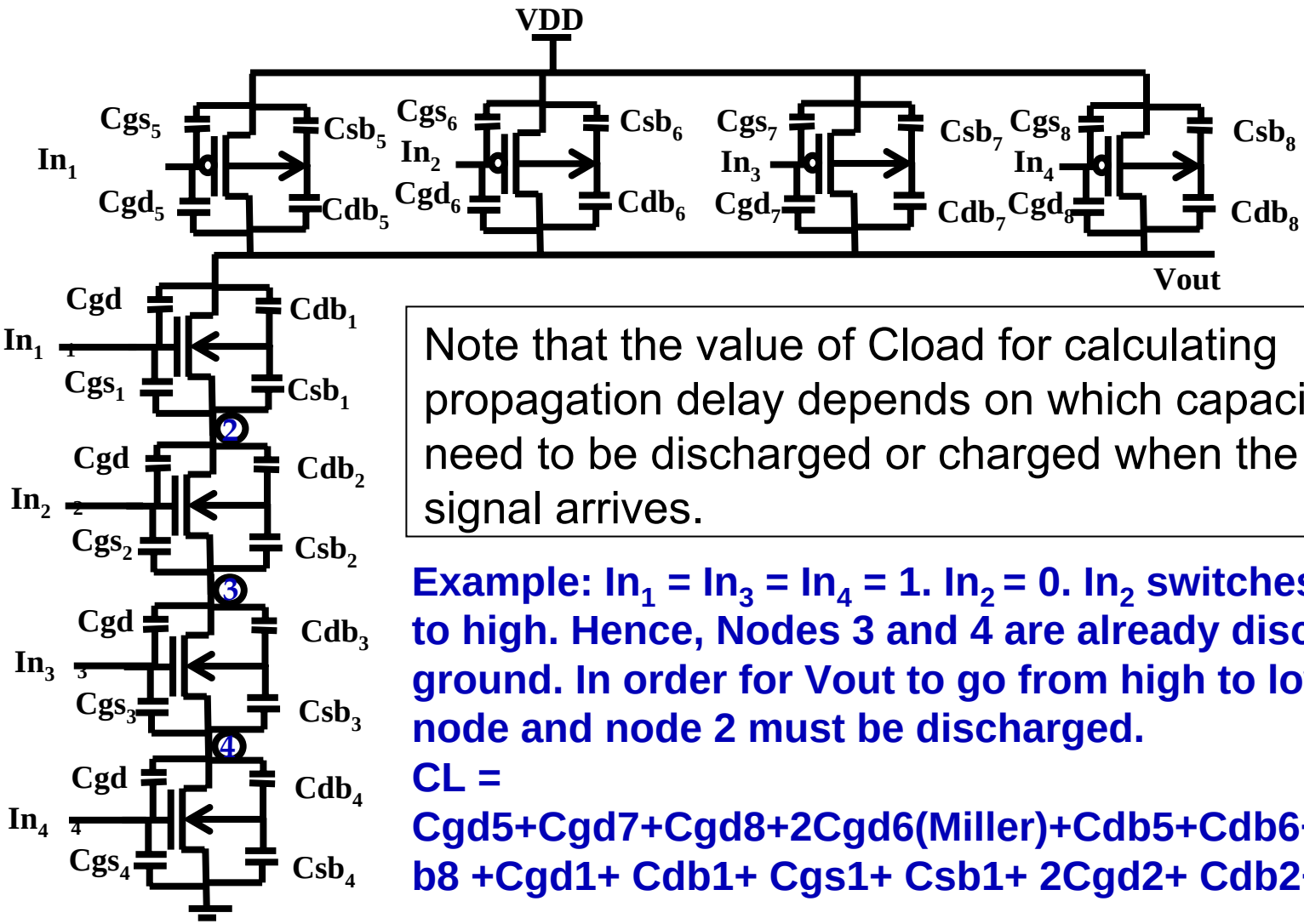
Keeps high fan-in resistance isolated from large capacitive load C_L

4 Input NAND Gate





Capacitances in a 4 input NAND Gate



Note that the value of Cload for calculating propagation delay depends on which capacitances need to be discharged or charged when the critical signal arrives.

Example: $In_1 = In_3 = In_4 = 1$. $In_2 = 0$. In_2 switches from low to high. Hence, Nodes 3 and 4 are already discharged to ground. In order for Vout to go from high to low… Vout node and node 2 must be discharged.

CL = Cgd5+Cgd7+Cgd8+2Cgd6(Miller)+Cdb5+Cdb6+Cdb7+Cdb8 +Cgd1+ Cdb1+ Cgs1+ Csb1+ 2Cgd2+ Cdb2+ Cw

Next Topic: Arithmetic

- **Computing arithmetic functions with CMOS logic**
  - Half adder and full adder circuits
  - Circuit architectures for addition
  - Array multipliers