EEC 118 Lecture #7: CMOS Logic

Rajeevan Amirtharajah
University of California, Davis
Jeff Parkhurst
Intel Corporation

Announcements

- **Hspice documentation (html and pdf): /pkg/avanti/current/docs**
- **Lab 3, Part 1 reports due this week at lab section**
- **HW 4 due this Friday at 4 PM in box, Kemper 2131**

Outline

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

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**

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

Inverter Power Consumption

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

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

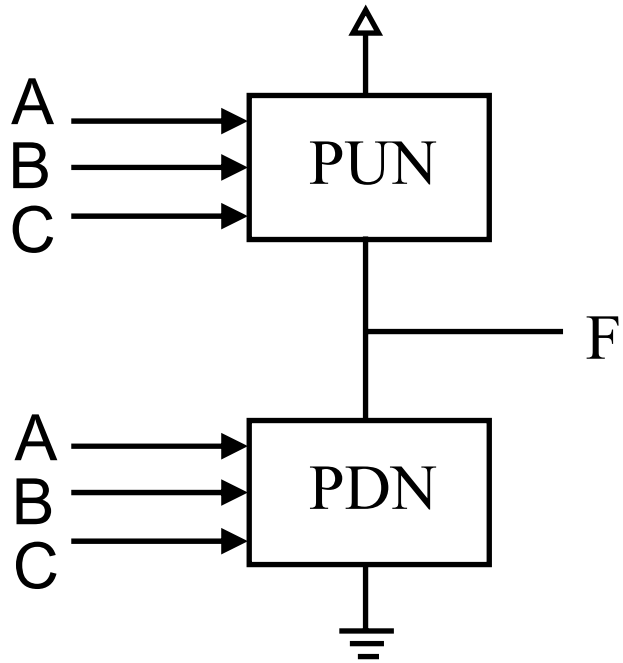
$$P_{avg} = \frac{1}{T}\left[\int_0^{T/2} V_{out}\left(-C_{load}\frac{dV_{out}}{dt}\right)dt + \int_{T/2}^{T}\left(V_{DD}-V_{out}\right)\left(C_{load}\frac{dV_{out}}{dt}\right)dt\right]$$

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

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

Static CMOS

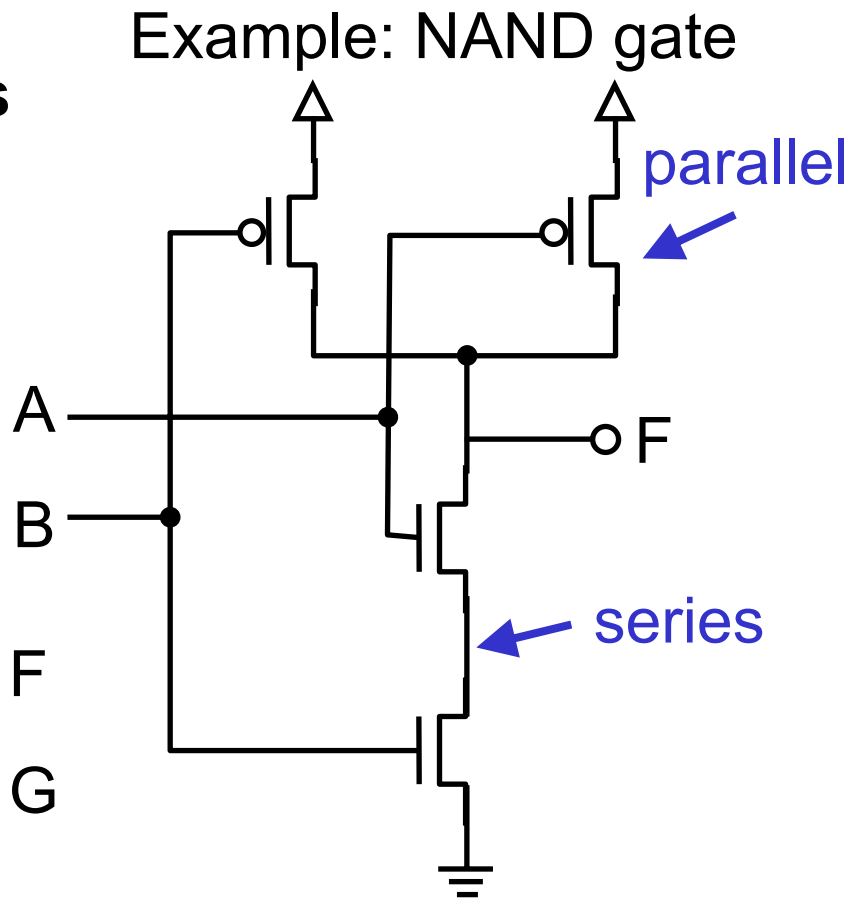
- **Complementary pullup network (PUN) and pulldown network (PDN)**
- **Only <u>one</u> network is on at a time**
- **PUN: PMOS devices**
  - Why?
- **PDN: NMOS devices**
  - Why?
- **PUN and PDN are *dual* networks**



Dual Networks

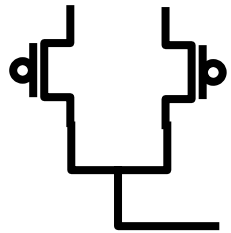
- **Dual networks: parallel connection in PDN = series connection in PUN, vice-versa**

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

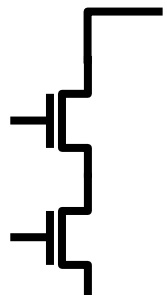


NAND Gate

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



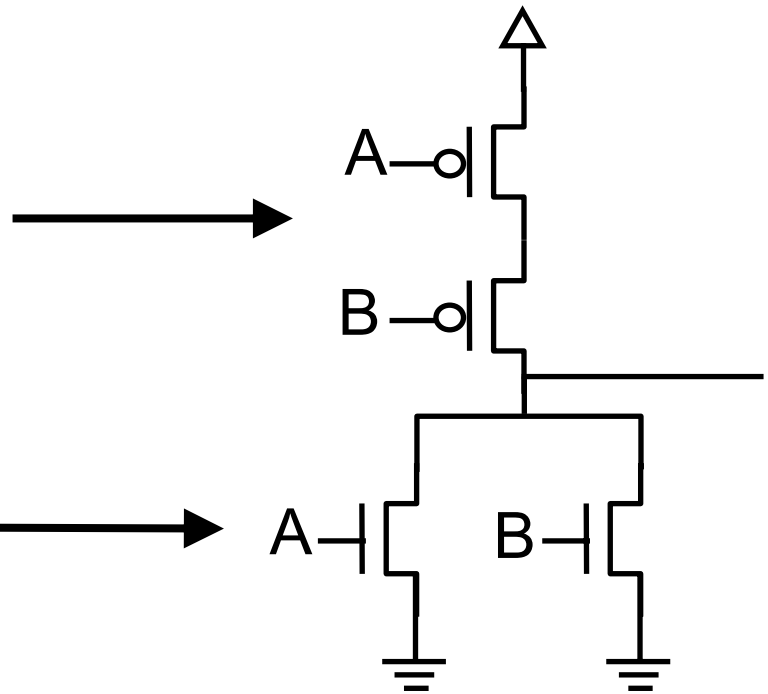
- **PUN function: $F = \overline{A \bullet B} = \overline{A} + \overline{B}$**
  - "Or" function (+) → parallel connection
  - Inverted inputs $\overline{A}$, $\overline{B}$ → PMOS transistors



- **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: $G = \overline{F} = A+B$



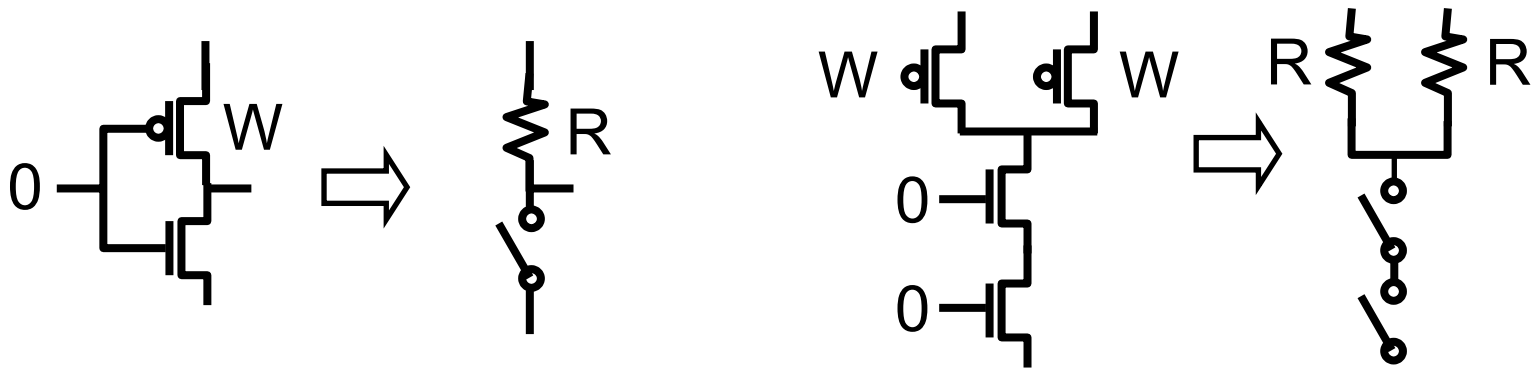
Analysis of CMOS Gates

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

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

Analysis of CMOS Gates (cont.)

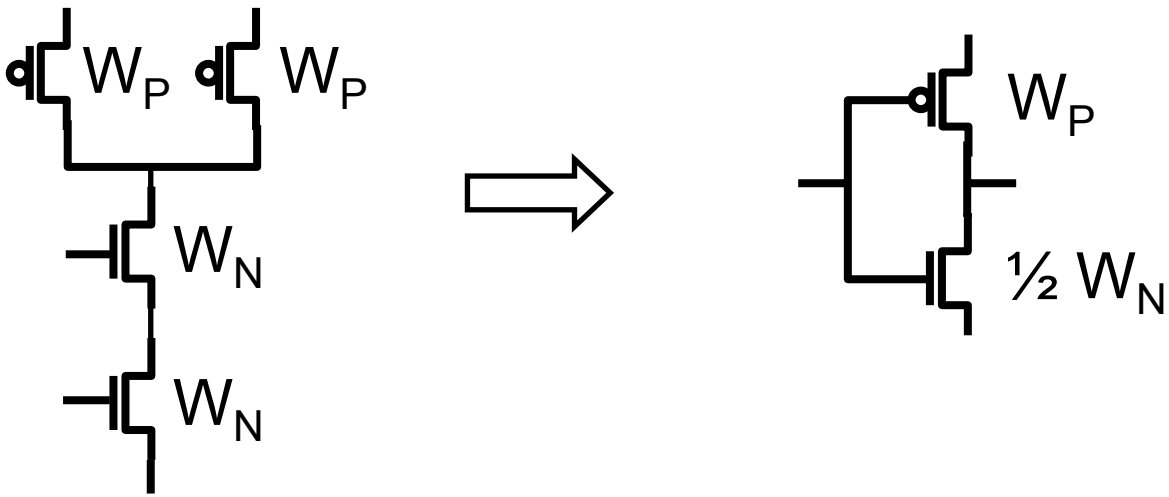
- Represent “on” transistors as resistors



- 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

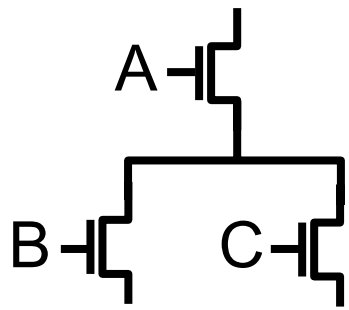
$$F = \overline{A \bullet (B+C)}$$

Example: Complex Gate

Design CMOS gate for this logic function:

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

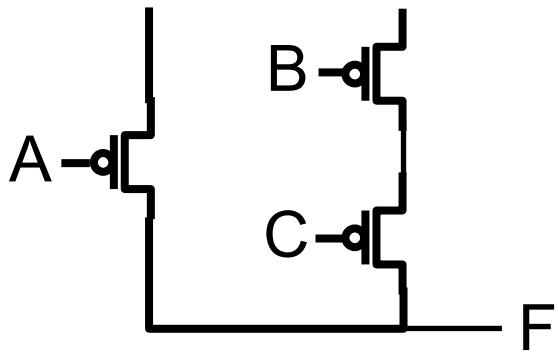
1. Find NMOS pulldown network diagram:
   $G = \overline{F} = A \bullet (B+C)$



Not a unique solution: can exchange order of series connection

Example: Complex Gate

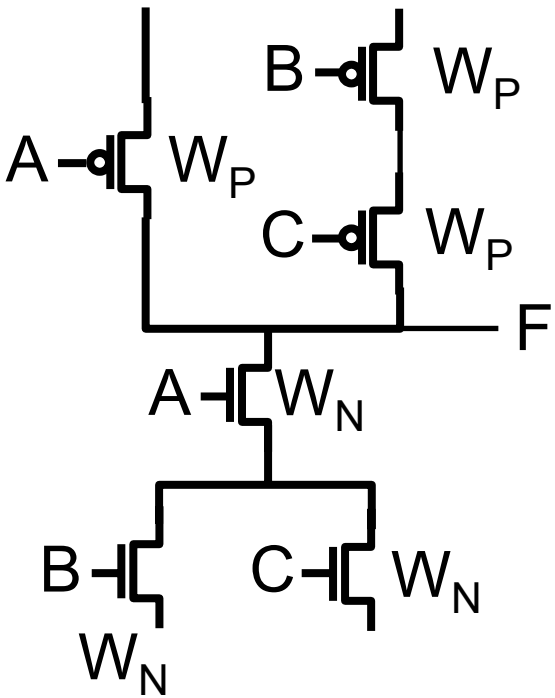
2. Find PMOS pullup network diagram: $F = \overline{A}+(\overline{B}\bullet\overline{C})$



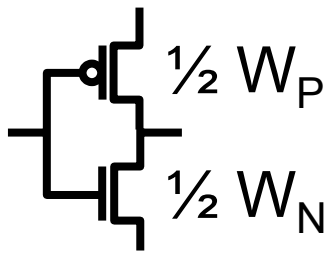
Not a unique solution: can exchange order of series connection (B and C inputs)

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

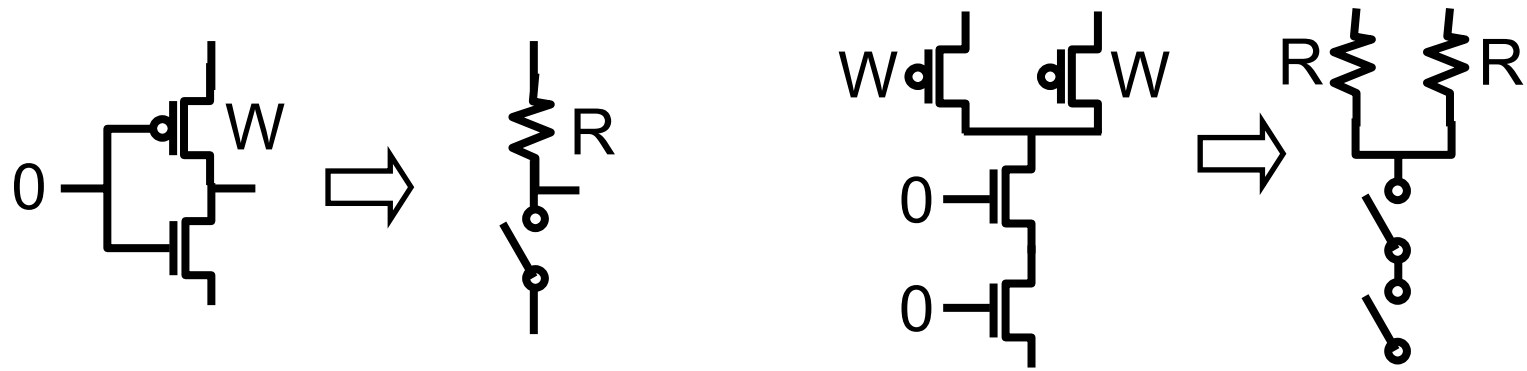
Analysis of CMOS gates

- **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_{TH}, 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_{TH}, delays, etc using inverter equations

Static CMOS Logic Characteristics

- **For V_{TH}, the V_{TH} of the equivalent inverter is used (assumes all inputs are tied together)**
  - For specific input patterns, V_{TH} 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**

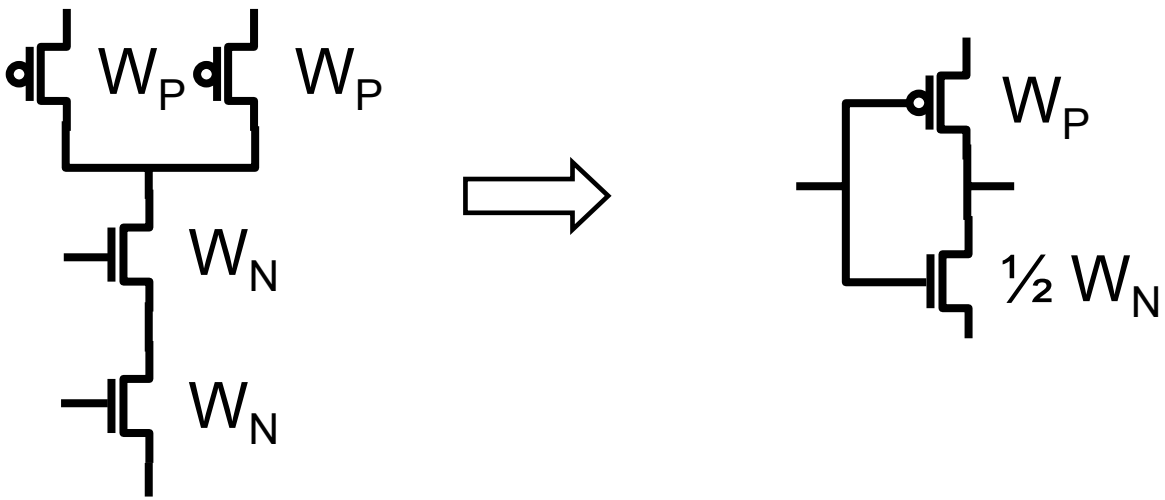
Equivalent Inverter: V_{TH}

- **Example: NAND gate threshold V_{TH}**
  Three possibilities:
  - A & B switch together
  - A switches alone
  - B switches alone

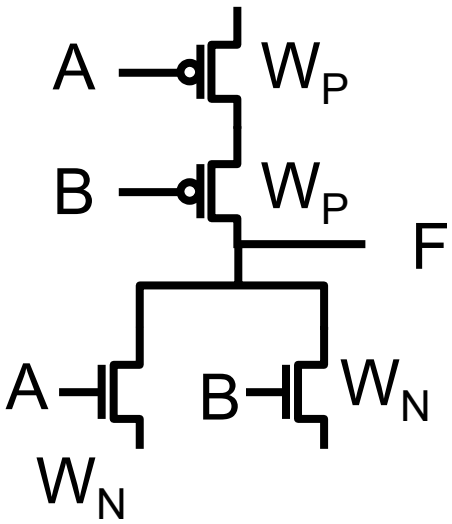
- **What is equivalent inverter for each case?**

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”



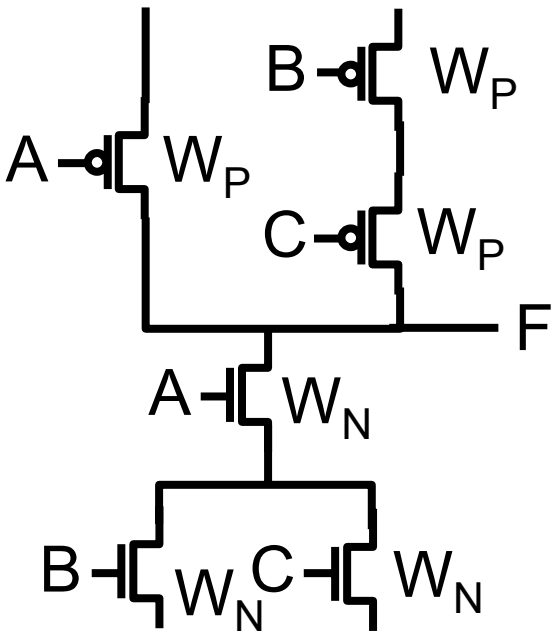
Example: NOR gate



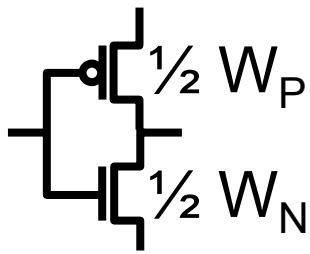
- **Find threshold voltage V_{TH} when both inputs switch simultaneously**
- **Two methods:**
  - Transistor equations (complex)
  - Equivalent inverter
  - Should get same answer

Example: Complex Gate

Completed gate:



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



Transistor Sizing

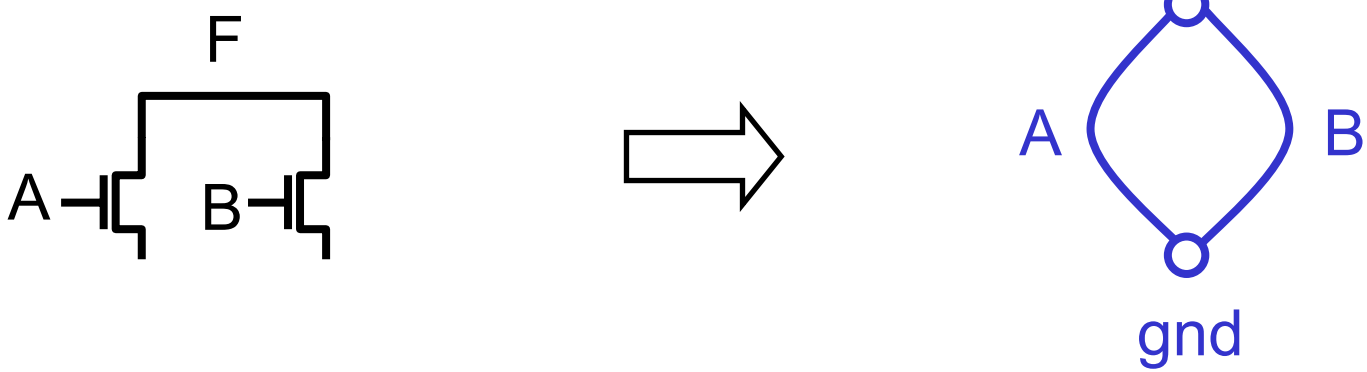
- **Sizing for switching threshold**
  - All inputs switch together
- **Sizing for delay**
  - Find worst-case input combination
- **Find equivalent inverter, use inverter analysis to set device sizes**

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

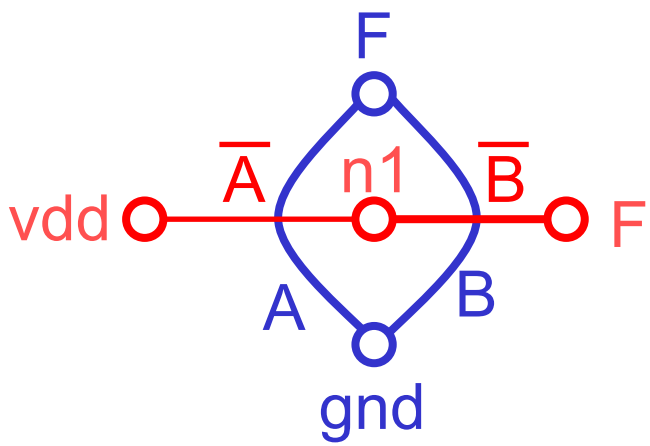
Graph-Based Dual Network

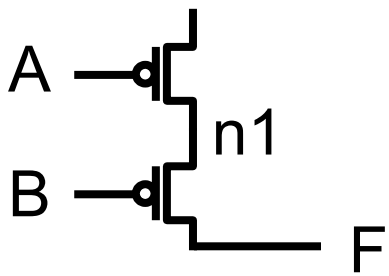
- **Use graph theory to help design gates**
  - Mostly implemented in CAD tools
- **Draw network for PUN or PDN**
  - Circuit nodes are vertices
  - Transistors are edges



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

Next Time: More Combinational Logic

- **Combinational MOS Logic Transient Response**
  - AC Characteristics, Switch Model