

# Performance Characterization

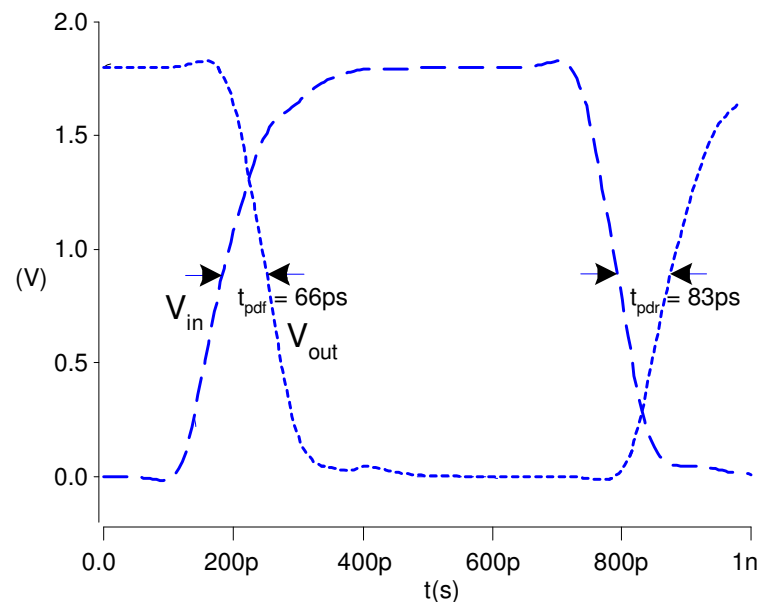
- Delay analysis
- Transistor sizing
- Logical effort
- Power analysis

# Delay Definitions

- $t_{\text{pdr}}$ : *rising propagation delay*
  - From input to rising output crossing  $V_{\text{DD}}/2$
- $t_{\text{pdf}}$ : *falling propagation delay*
  - From input to falling output crossing  $V_{\text{DD}}/2$
- $t_{\text{pd}}$ : *average propagation delay*
  - $t_{\text{pd}} = (t_{\text{pdr}} + t_{\text{pdf}})/2$
- $t_{\text{r}}$ : *rise time*
  - From output crossing  $0.2 V_{\text{DD}}$  to  $0.8 V_{\text{DD}}$
- $t_{\text{f}}$ : *fall time*
  - From output crossing  $0.8 V_{\text{DD}}$  to  $0.2 V_{\text{DD}}$

# Simulated Inverter Delay

- Solving differential equations by hand is too hard
- SPICE simulator solves the equations numerically
  - Uses more accurate I-V models too!
- But simulations take time to write

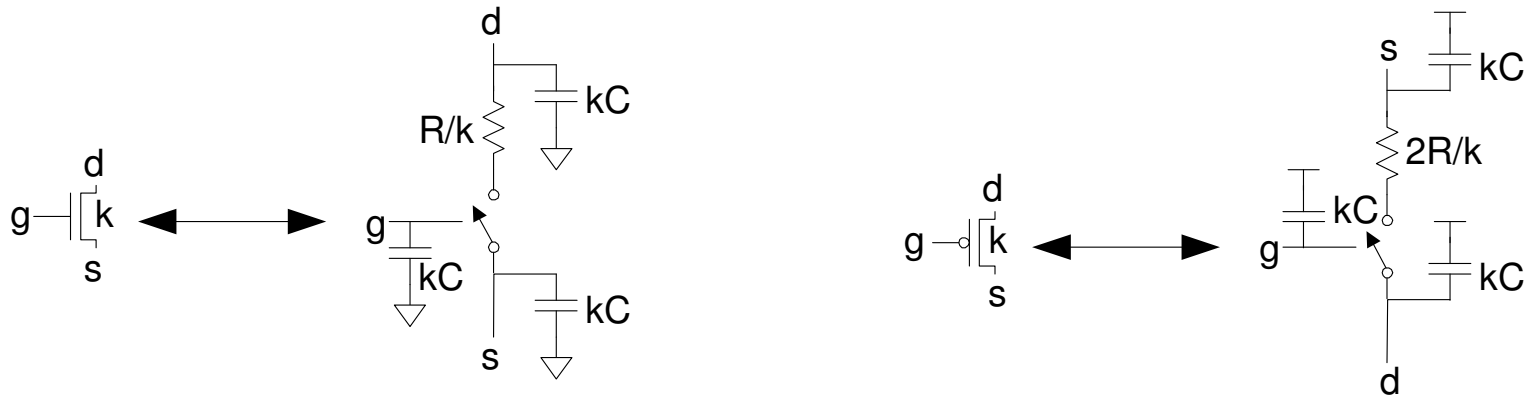


# Delay Estimation

- We would like to be able to easily estimate delay
  - Not as accurate as simulation
  - But easier to ask “What if?”
- The step response usually looks like a 1<sup>st</sup> order RC response with a decaying exponential.
- Use RC delay models to estimate delay
  - $C$  = total capacitance on output node
  - Use *effective resistance*  $R$
  - So that  $t_{pd} = RC$
- Characterize transistors by finding their effective  $R$ 
  - Depends on average current as gate switches

# RC Delay Models

- Use equivalent circuits for MOS transistors
  - Ideal switch + capacitance and ON resistance
  - Unit nMOS has resistance  $R$ , capacitance  $C$
  - Unit pMOS has resistance  $2R$ , capacitance  $C$
- Capacitance proportional to width
- Resistance inversely proportional to width

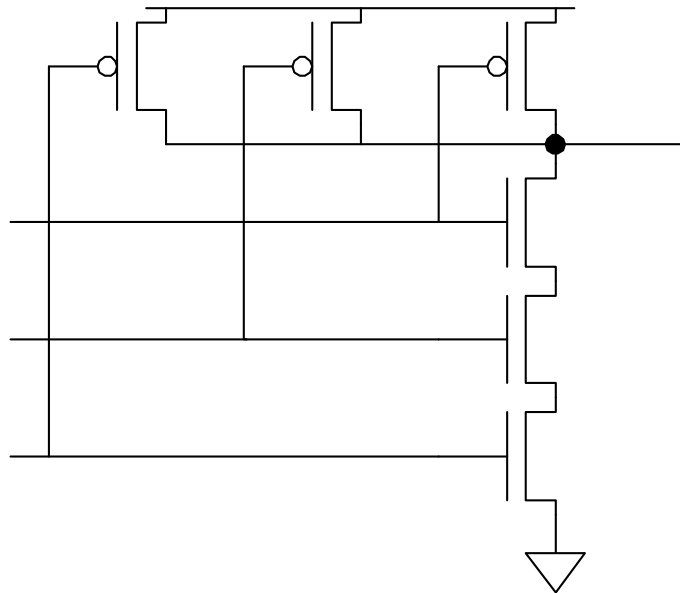


# Example: 3-input NAND

- Sketch a 3-input NAND with transistor widths chosen to achieve effective rise and fall resistances equal to a unit inverter (R).

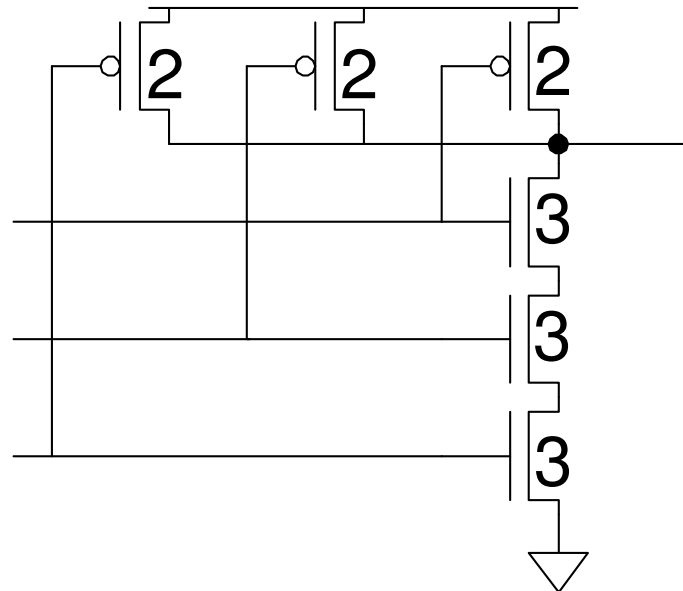
# Example: 3-input NAND

- Sketch a 3-input NAND with transistor widths chosen to achieve effective rise and fall resistances equal to a unit inverter (R).



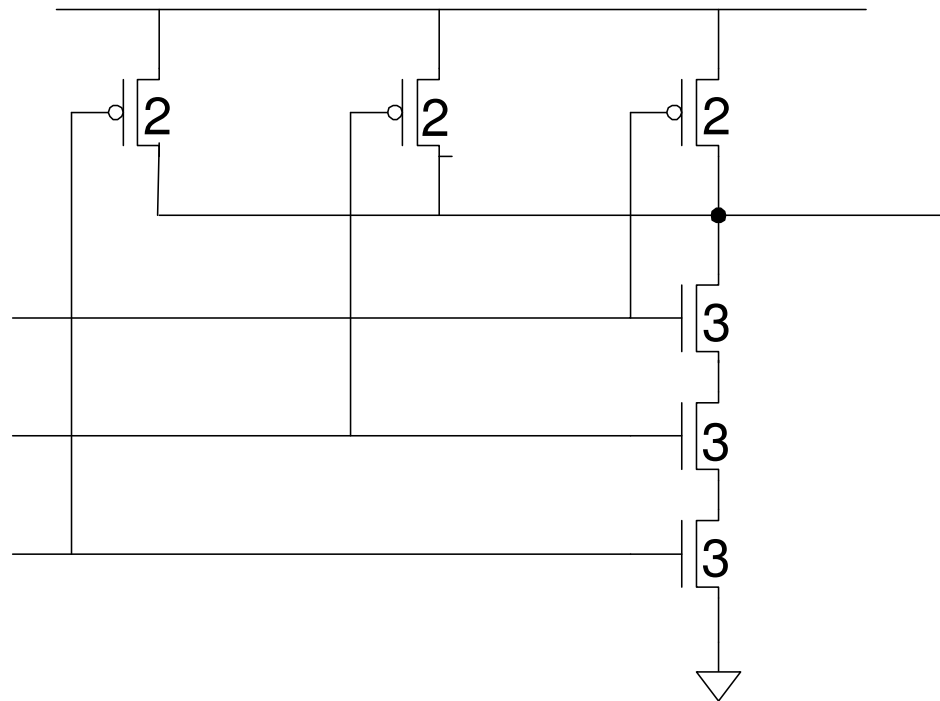
# Example: 3-input NAND

- Sketch a 3-input NAND with transistor widths chosen to achieve effective rise and fall resistances equal to a unit inverter (R).



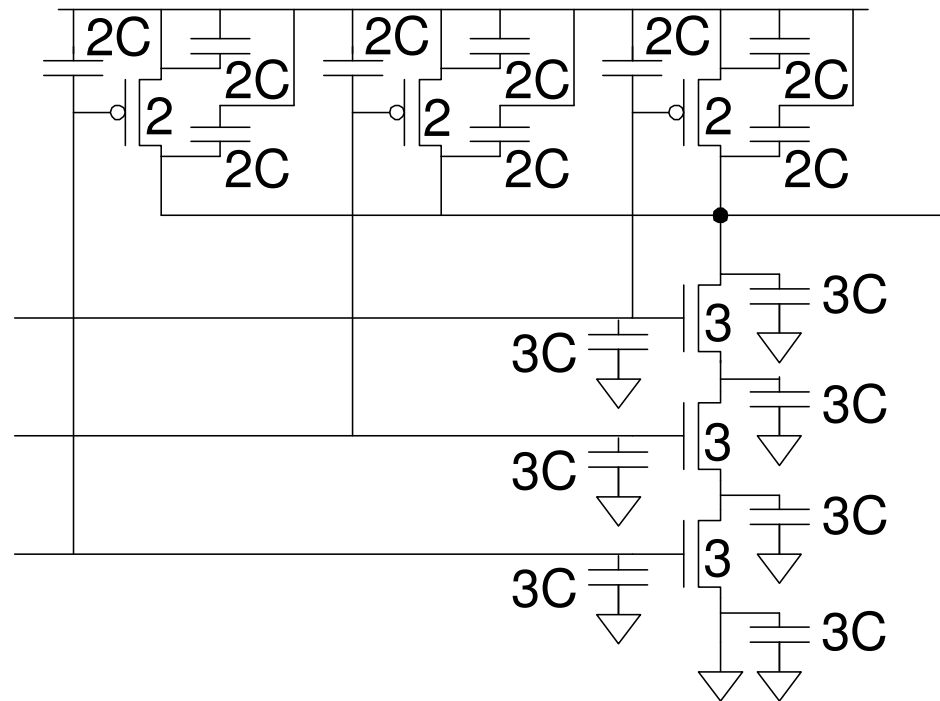
# 3-input NAND Caps

- Annotate the 3-input NAND gate with gate and diffusion capacitance.



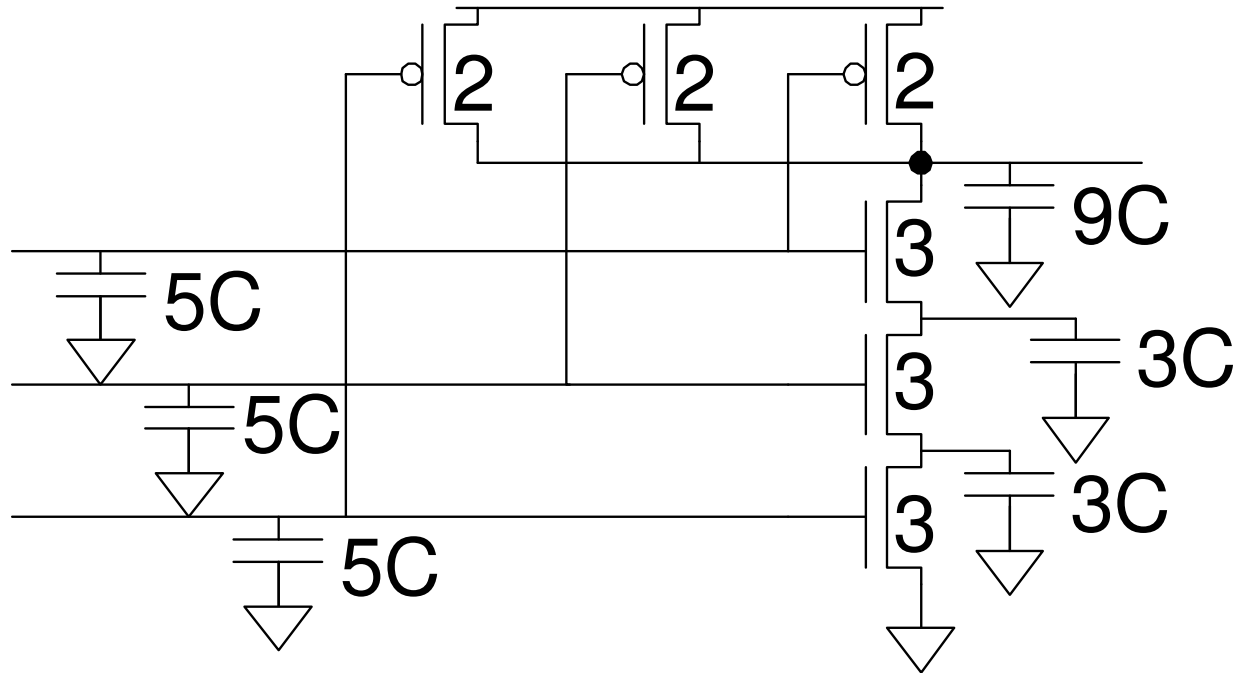
# 3-input NAND Caps

- Annotate the 3-input NAND gate with gate and diffusion capacitance.



# 3-input NAND Caps

- Annotate the 3-input NAND gate with gate and diffusion capacitance.

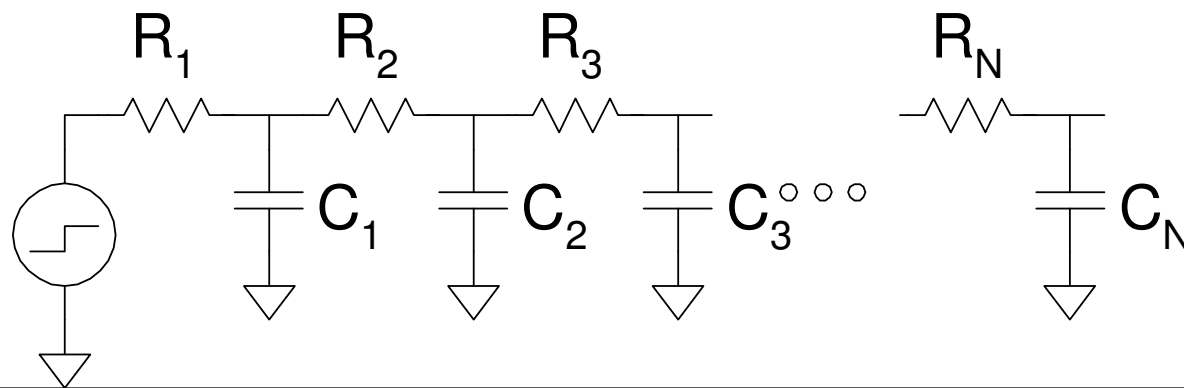


# Elmore Delay

- ON transistors look like resistors
- Pullup or pulldown network modeled as *RC ladder*
- Elmore delay of RC ladder

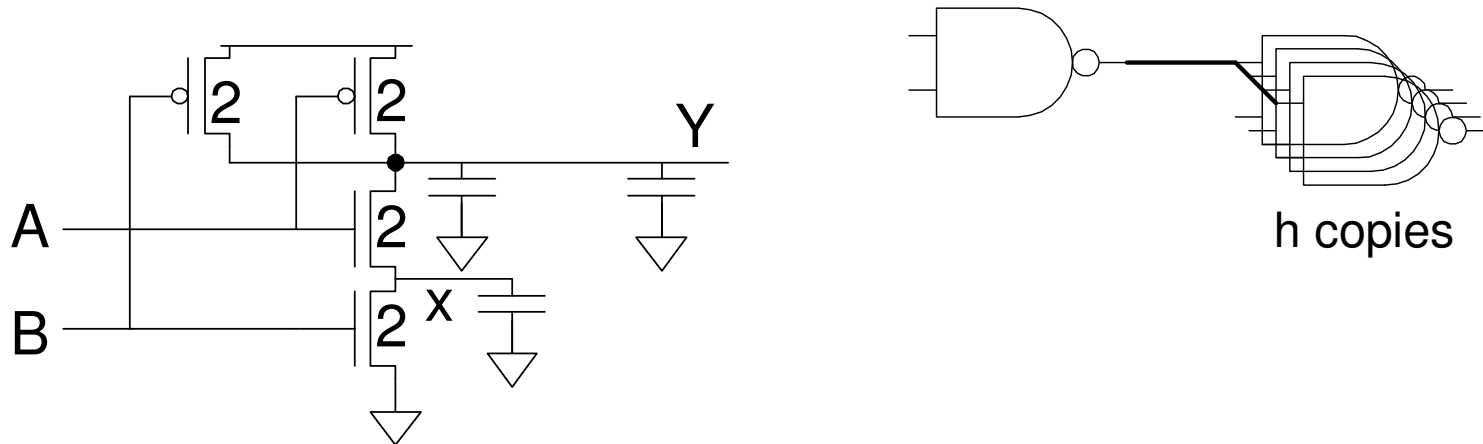
$$t_{pd} \approx \sum_{\text{nodes } i} R_{i\text{-to-source}} C_i$$

$$= R_1 C_1 + (R_1 + R_2) C_2 + \dots + (R_1 + R_2 + \dots + R_N) C_N$$



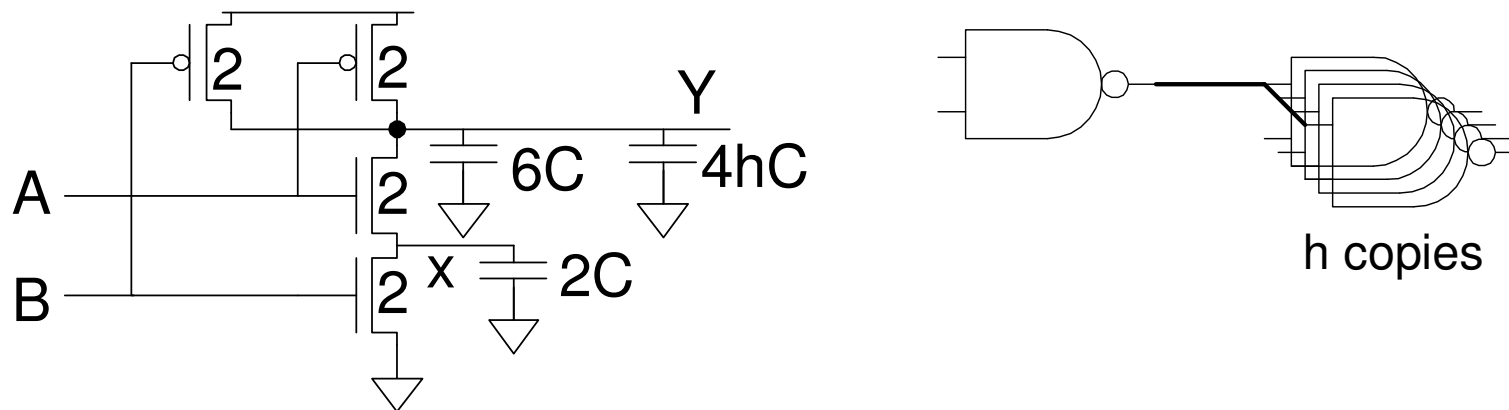
# Example: 2-input NAND

- Estimate worst-case rising and falling delay of 2-input NAND driving  $h$  identical gates.



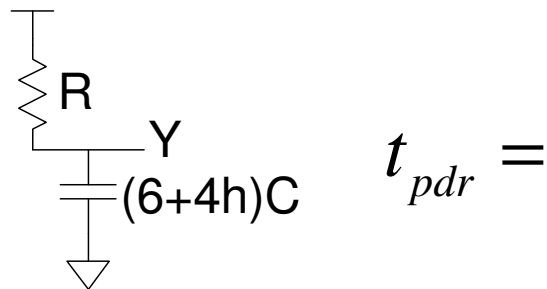
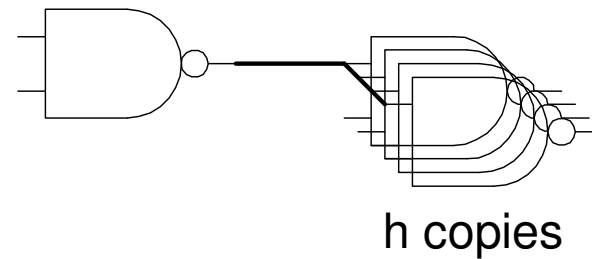
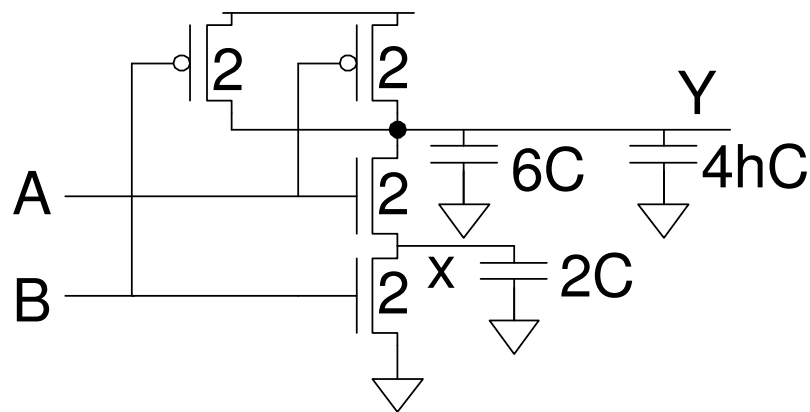
# Example: 2-input NAND

- Estimate rising and falling propagation delays of a 2-input NAND driving  $h$  identical gates.



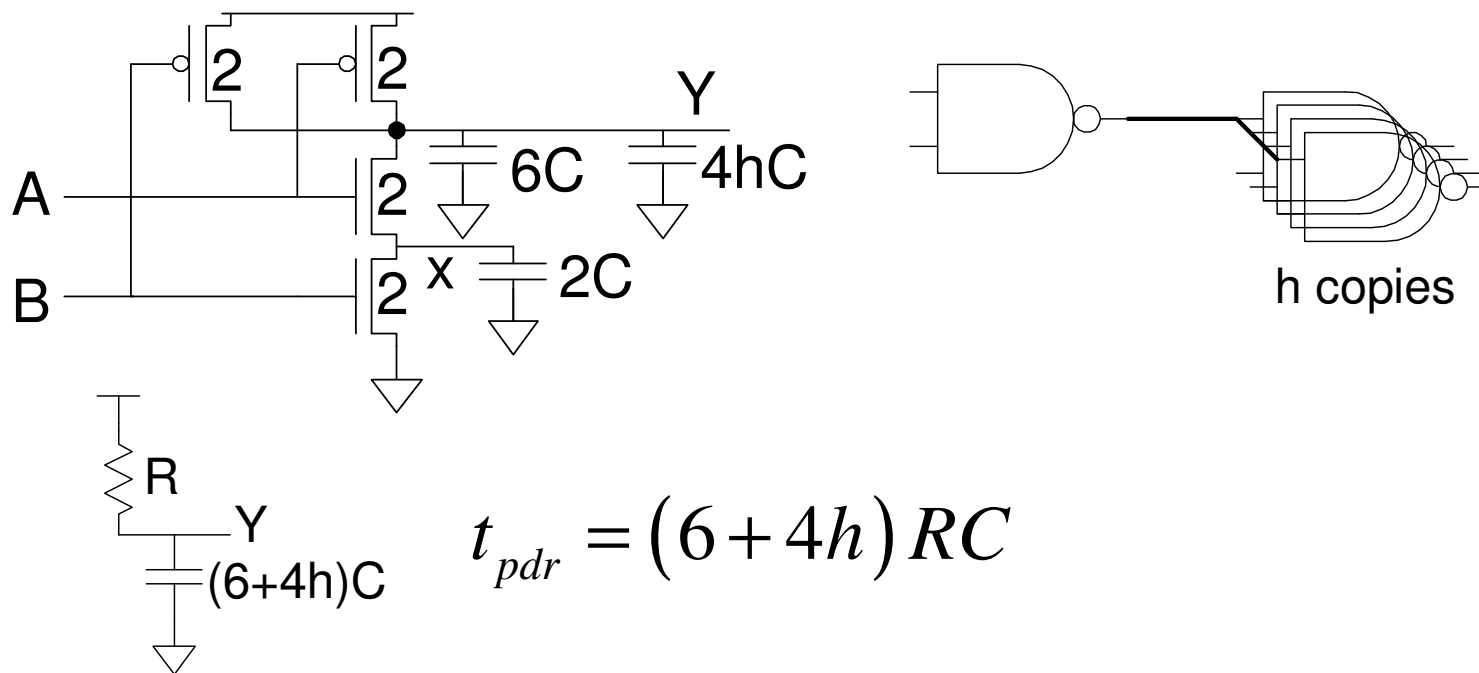
# Example: 2-input NAND

- Estimate **rising** and falling propagation delays of a 2-input NAND driving  $h$  identical gates.



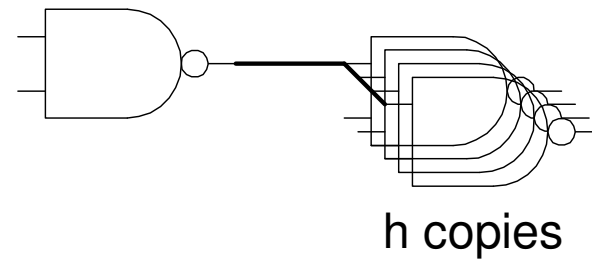
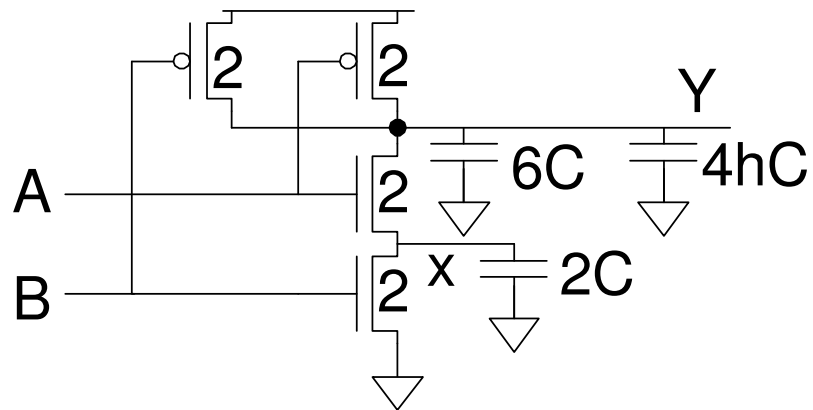
# Example: 2-input NAND

- Estimate **rising** and falling propagation delays of a 2-input NAND driving  $h$  identical gates.



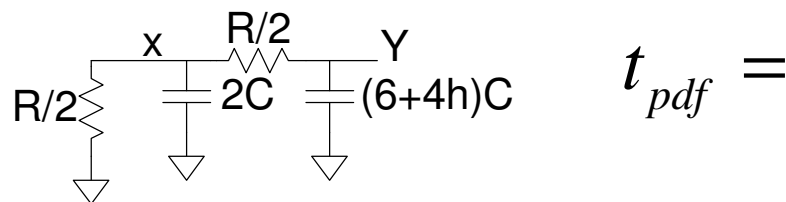
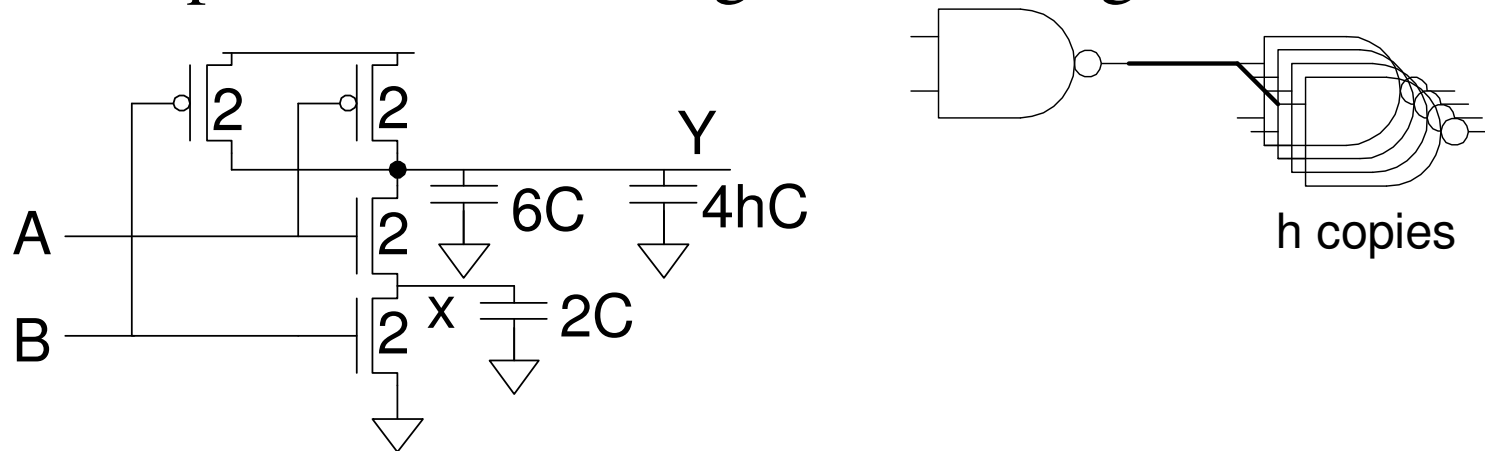
# Example: 2-input NAND

- Estimate rising and **falling** propagation delays of a 2-input NAND driving  $h$  identical gates.



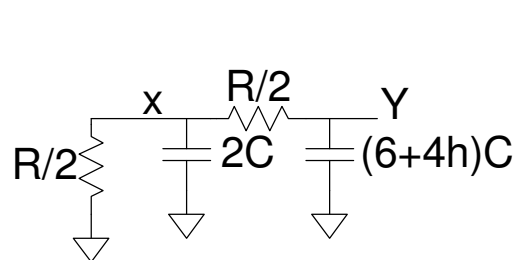
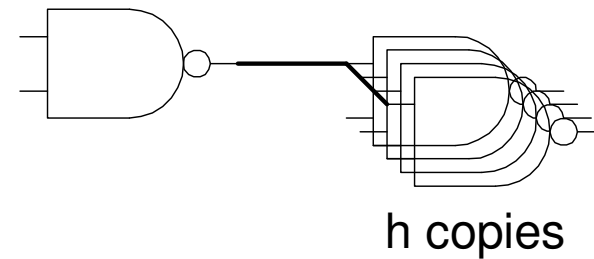
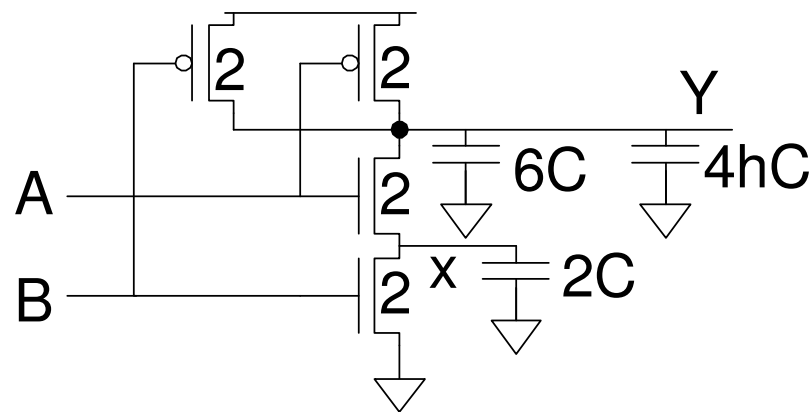
# Example: 2-input NAND

- Estimate rising and **falling** propagation delays of a 2-input NAND driving  $h$  identical gates.



# Example: 2-input NAND

- Estimate rising and **falling** propagation delays of a 2-input NAND driving  $h$  identical gates.



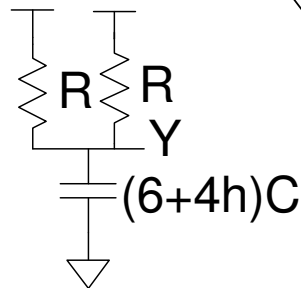
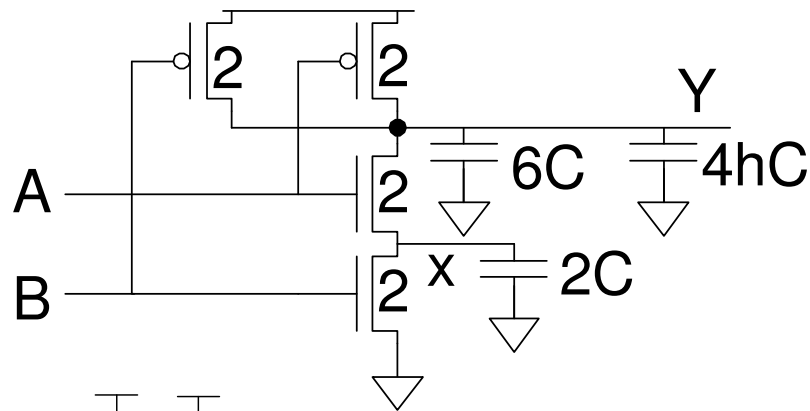
$$\begin{aligned}
 t_{pdf} &= (2C) \left( \frac{R}{2} \right) + \left[ (6 + 4h) C \right] \left( \frac{R}{2} + \frac{R}{2} \right) \\
 &= (7 + 4h) RC
 \end{aligned}$$

# Delay Components

- Delay has two parts
  - *Parasitic delay*
    - 6 or 7 RC
    - Independent of load
  - *Effort delay*
    - 4h RC
    - Proportional to load capacitance

# Contamination Delay

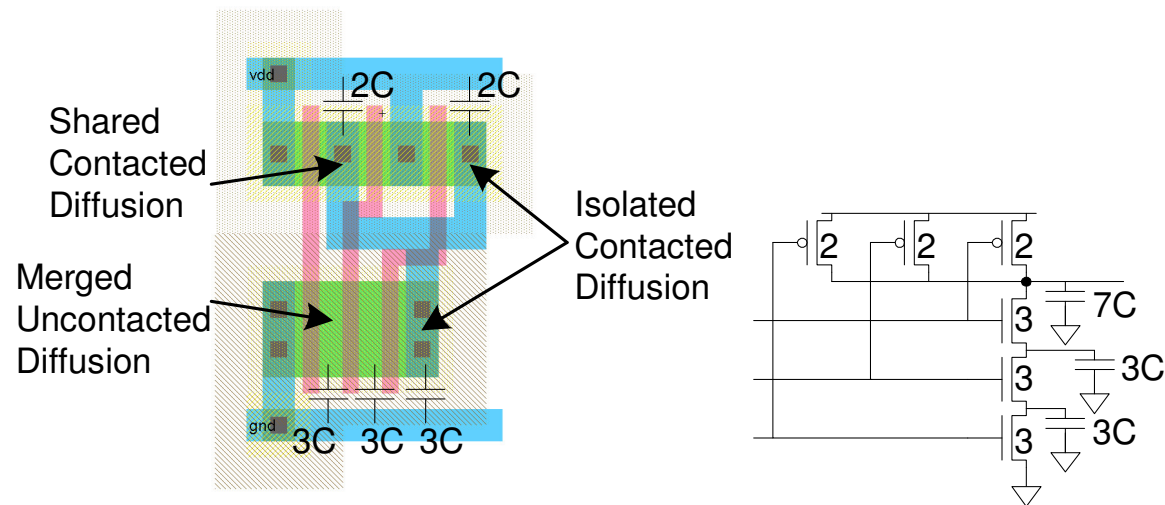
- Best-case (contamination) delay can be substantially less than propagation delay.
- Ex: If both inputs fall simultaneously



$$t_{cdr} = (3 + 2h) RC$$

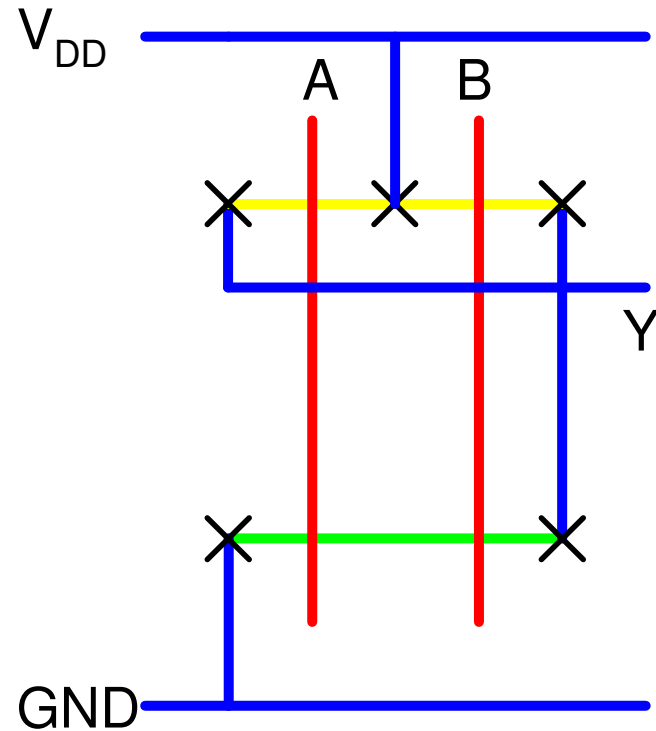
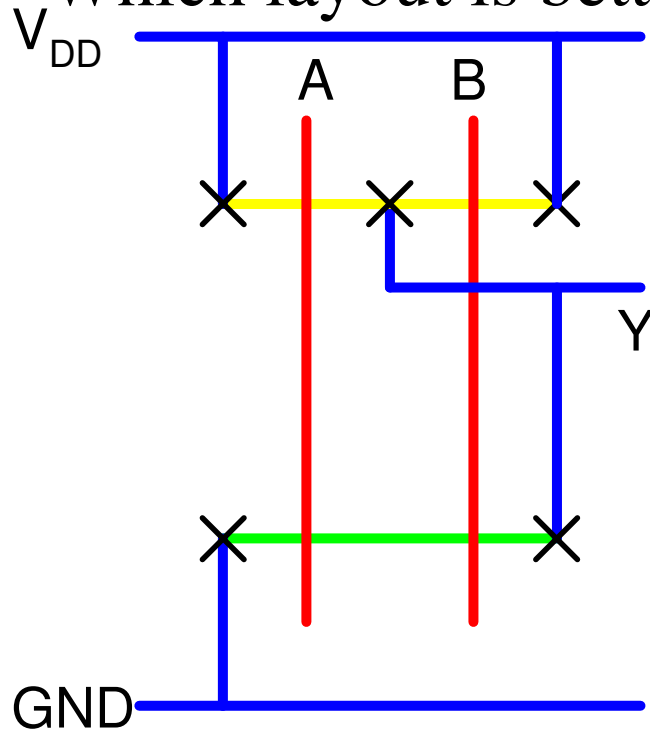
# Diffusion Capacitance

- We assumed contacted diffusion on every s / d.
- Good layout minimizes diffusion area
- Ex: NAND3 layout shares one diffusion contact
  - Reduces output capacitance by  $2C$
  - Merged uncontacted diffusion might help too



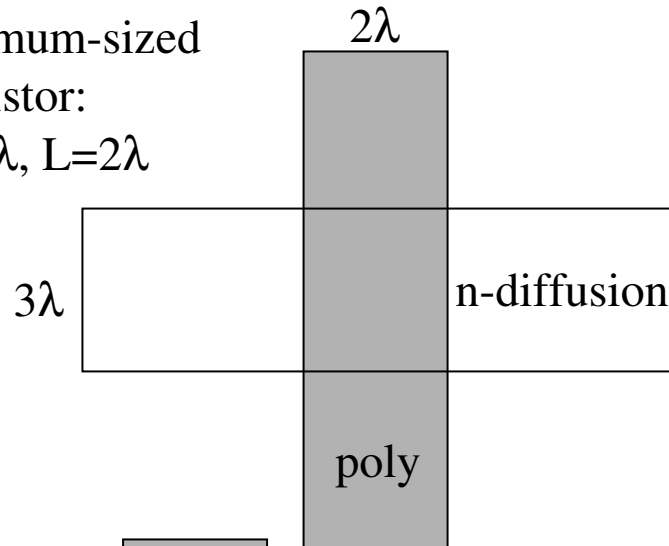
# Layout Comparison

- Which layout is better?



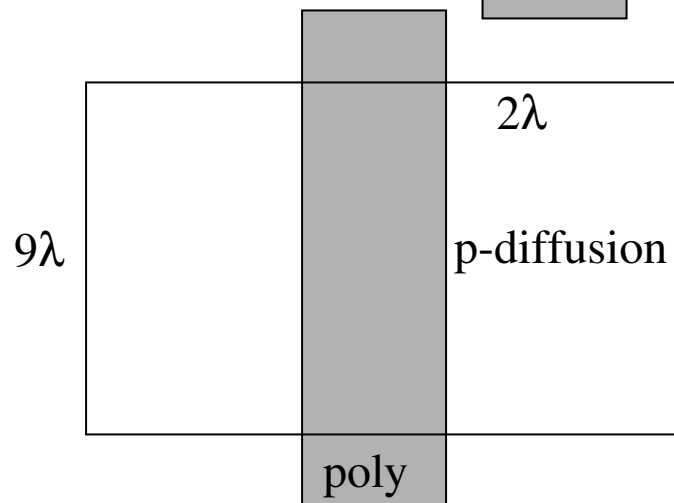
# Resizing the Inverter

Minimum-sized  
transistor:  
 $W=3\lambda$ ,  $L=2\lambda$



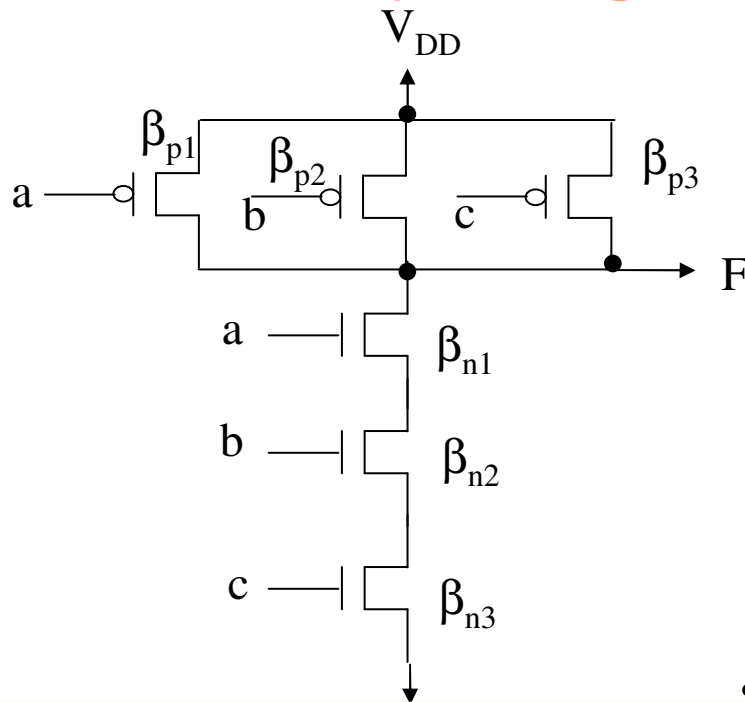
To get equal rise and fall times,  
 $\beta_n = \beta_p \Rightarrow W_p = 3W_n$ , assuming  
that electron mobility is three times that  
of holes

$$W_p = 9\lambda$$



*Sometimes the function  
being implemented  
makes resizing  
unnecessary!*

# Analyzing the NAND Gate



$$\beta_{n, \text{eff}} = \frac{1}{\frac{1}{\beta_{n1}} + \frac{1}{\beta_{n2}} + \frac{1}{\beta_{n3}}}$$

Resistances are in series (conductances are in parallel)

If  $\beta_{n1} = \beta_{n2} = \beta_{n3} = \beta_n$  then  $\beta_{n, \text{eff}} = \beta_n/3$

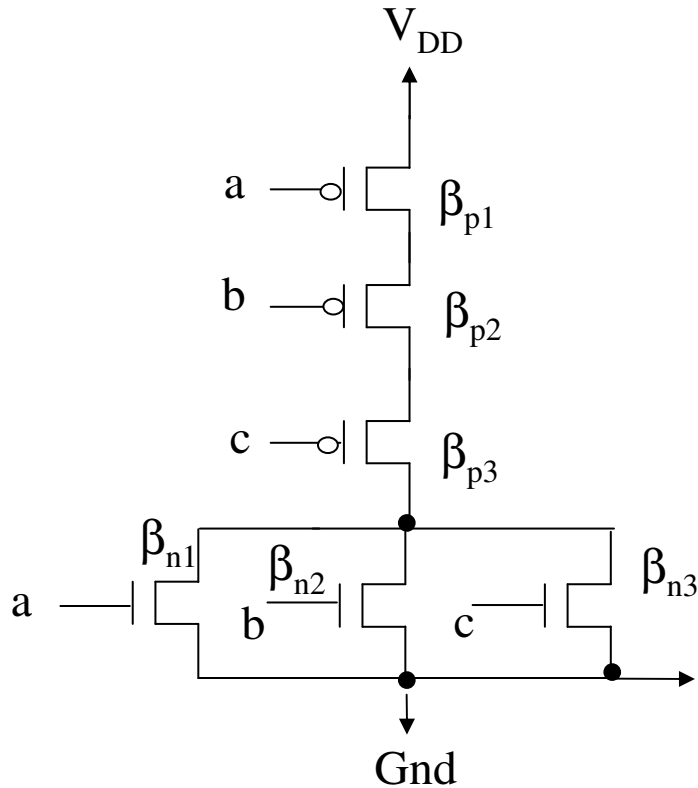
- Pull-down circuit has three times resistance, one-third times the conductance

*Why not consider resistances in parallel?*

For pull-up, only one transistor has to be on,  $\beta_{p, \text{eff}} = \min\{\beta_{p1}, \beta_{p2}, \beta_{p3}\}$

If  $\beta_{p1} = \beta_{p2} = \beta_{p3} = \beta_p = \beta_n/3$  then  $\beta_{n, \text{eff}} = \beta_p \Rightarrow$  no resizing is necessary

# Analyzing the NOR Gate



$$\beta_{p, \text{eff}} = \frac{1}{\frac{1}{\beta_{p1}} + \frac{1}{\beta_{p2}} + \frac{1}{\beta_{p3}}}$$

Resistances are in series (conductances are in parallel)

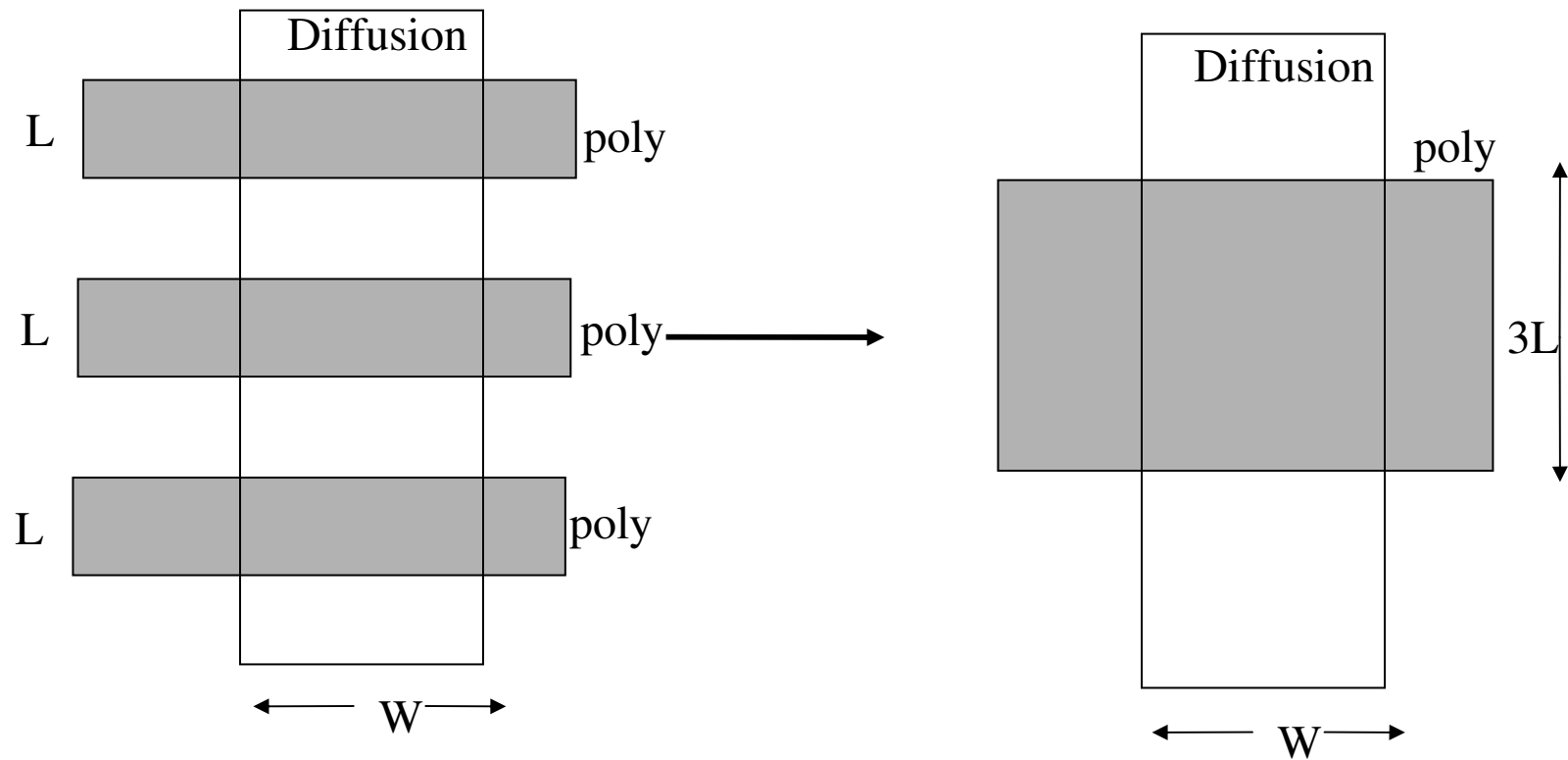
If  $\beta_{p1} = \beta_{p2} = \beta_{p3} = \beta_p$  then  $\beta_{p, \text{eff}} = \beta_p/3$

- Pull-up circuit has three times resistance, one-third times the conductance

For pull-down, only one transistor has to be on,  $\beta_{n, \text{eff}} = \min\{\beta_{n1}, \beta_{n2}, \beta_{n3}\}$

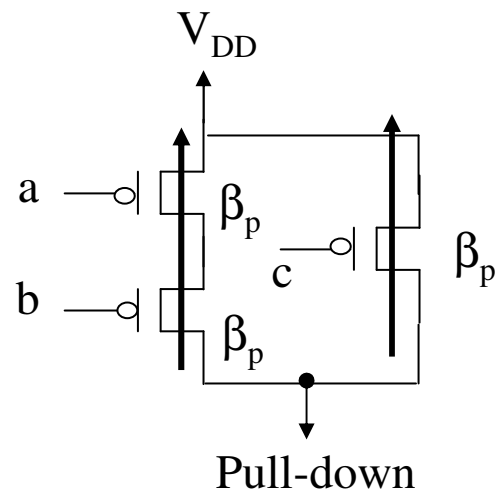
If  $\beta_{n1} = \beta_{n2} = \beta_{n3} = \beta_n = 3\beta_p$  then  $\beta_{n, \text{eff}} = 9\beta_{p, \text{eff}} \Rightarrow$  considerable resizing is necessary  
 $W_p = 9W_n!$

# Effect of Series Transistors



# Effect of Series Transistors

Transistor  
resizing  
example



Resize the pull-up transistors to  
make pull-up times equal

After resizing:

a:  $2\beta_p$ , b:  $2\beta_p$ , c:  $\beta_p$

# Transistor Placement (Series Stack)

How to order transistors in a series stack?

Body effect:  $\delta V_t \propto$

$$\sqrt{V_{sb}}$$

- At time  $t = 0$ ,  $a=b=c=0$ ,  $f=1$ , capacitances are charged

- Ideally  $V_{ta} = V_{tb} = V_{tc} \approx 0.8V$

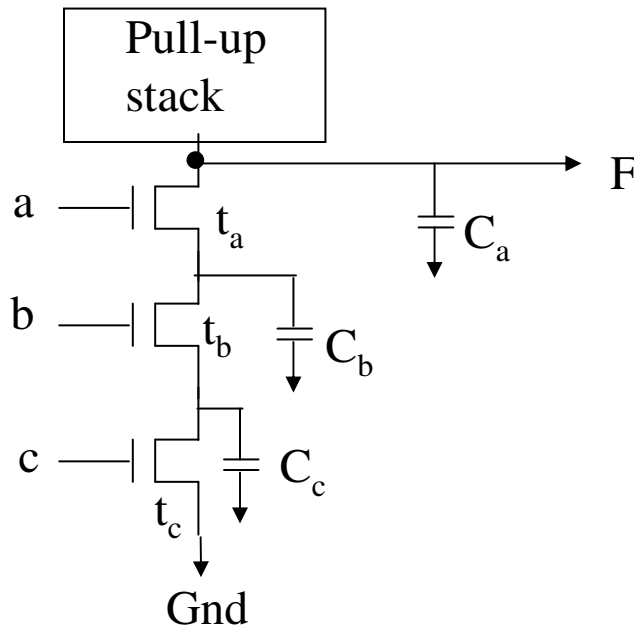
- However,  $V_{ta} > V_{tb} > V_{tc}$  because of body effect

- If  $a, b, c$  become 1 at the same time, which transistor will switch on first?

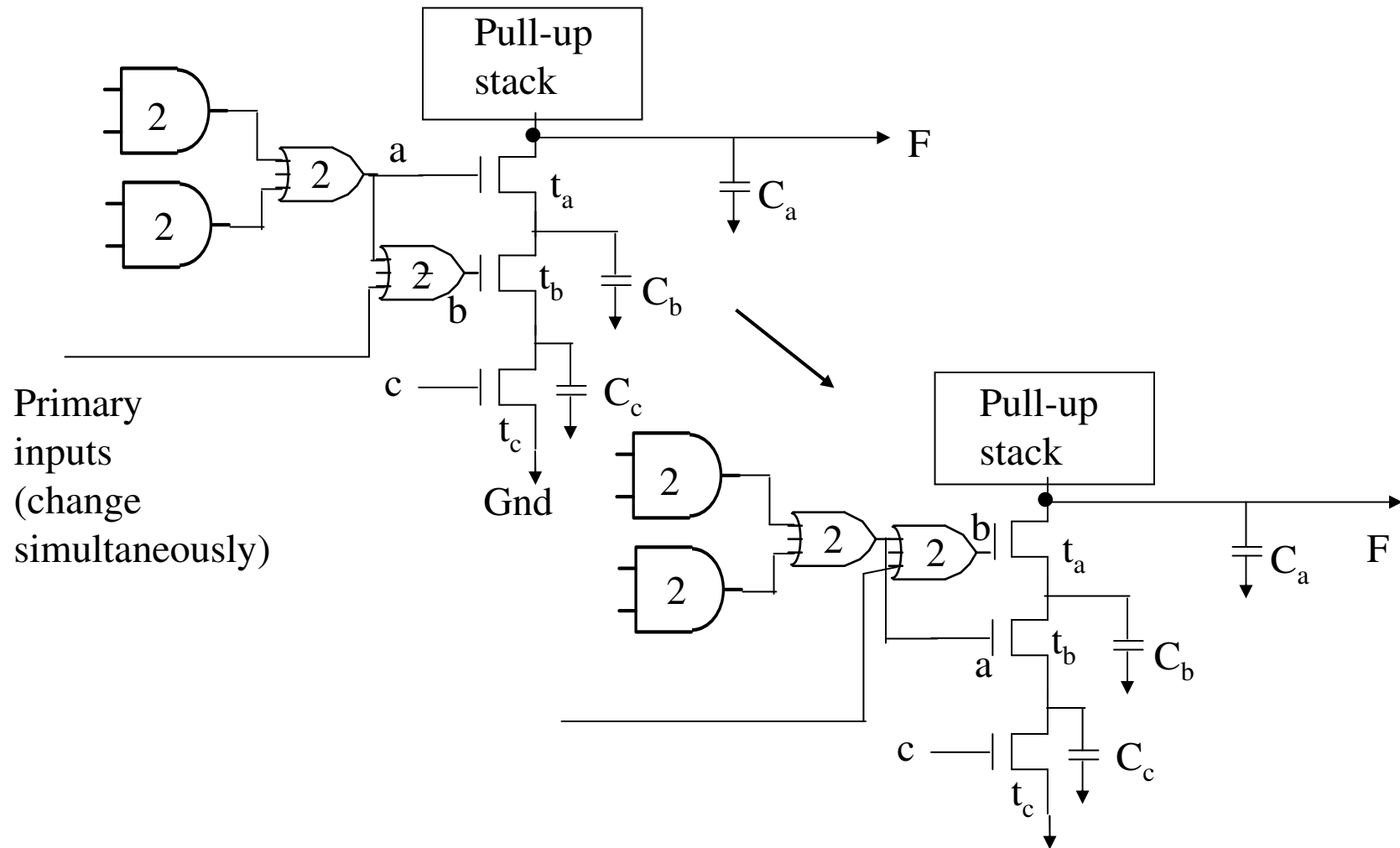
- $t_c$  will switch on first ( $V_{sb}$  for  $t_c$  is zero),  $C_c$  will discharge, pulling  $V_{sb}$  for  $t_b$  to zero

- If signals arrive at different times, how should the transistors be ordered?

- Design strategy: place latest arriving signal nearest to output-early signals will discharge internal nodes



# Transistor Placement

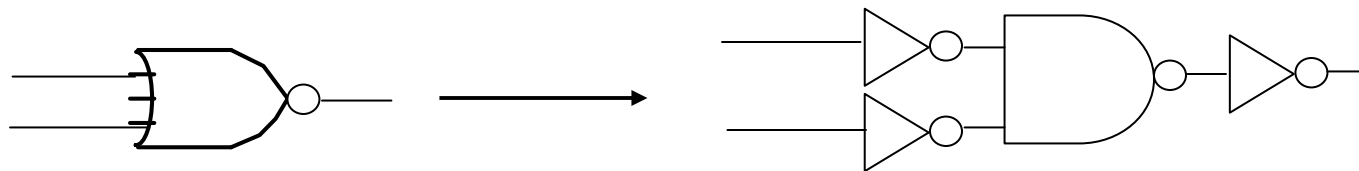


# Some Design Guidelines

- Use NAND gates (instead of NOR) wherever possible
- Place inverters (buffers) at high fanout nodes to improve drive capability
- Avoid use of NOR completely in high-speed circuits:  $A_1 + A_2 + \dots + A_n = A_1 \cdot A_2 \cdot \dots \cdot A_n$

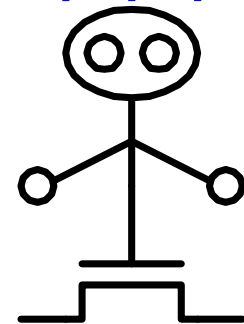
# Some Design Guidelines

- Use limited fan-in (<10): high fan-in  $\Rightarrow$  long series stacks
- Use minimum-sized gates on high fan-out nodes: minimize load presented to driving gate



# Logical Effort

- Chip designers face a bewildering array of choices ? ? ?
  - What is the best circuit topology for a function?
  - How many stages of logic give least delay?
  - How wide should the transistors be?
- Logical effort is a method to make these decisions
  - Uses a simple model of delay
  - Allows back-of-the-envelope calculations
  - Helps make rapid comparisons between alternatives
  - Emphasizes remarkable symmetries



# Example

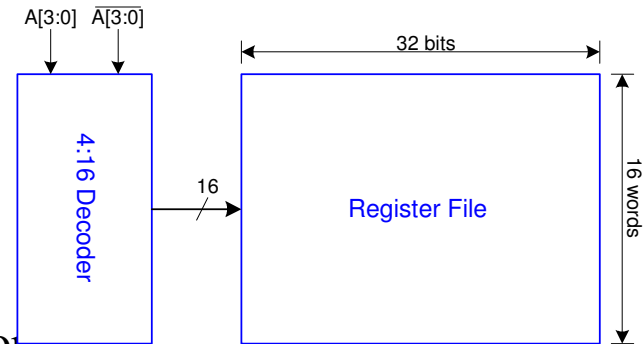
- Ben Bitdiddle is the memory designer for the Motorola 68W86, an embedded automotive processor. Help Ben design the decoder for a register file.

- Decoder specifications:

- 16 word register file
- Each word is 32 bits wide
- Each bit presents load of 3 unit-sized transistors
- True and complementary address inputs  $A[3:0]$
- Each input may drive 10 unit-sized transistors

- Ben needs to decide:

- How many stages to use?
- How large should each gate be?
- How fast can decoder operate?



# Delay in a Logic Gate

- Express delays in process-independent unit

$$d = \frac{d_{abs}}{\tau}$$

$$\tau = 3RC$$

≈ 12 ps in 180 nm process

40 ps in 0.6 μm process

# Delay in a Logic Gate

- Express delays in process-independent unit

$$d = \frac{d_{abs}}{\tau}$$

- Delay has two components

$$d = f + p$$

# Delay in a Logic Gate

- Express delays in process-independent unit

$$d = \frac{d_{abs}}{\tau}$$

- Delay has two components

$$d = f + p$$

- *Effort delay*  $f = gh$  (a.k.a. *stage effort*)
  - Again has two components

# Delay in a Logic Gate

- Express delays in process-independent unit

$$d = \frac{d_{abs}}{\tau}$$

- Delay has two components

$$d = f + p$$

- Effort delay  $f = gh$  (a.k.a. stage effort)
  - Again has two components
- $g$ : *logical effort*
  - Measures relative ability of gate to deliver current
  - $g \equiv 1$  for inverter

# Delay in a Logic Gate

- Express delays in process-independent unit

$$d = \frac{d_{abs}}{\tau}$$

- Delay has two components

$$d = f + p$$

- Effort delay  $f = gh$  (a.k.a. stage effort)
  - Again has two components
- $h$ : *electrical effort* =  $C_{out} / C_{in}$ 
  - Ratio of output to input capacitance
  - Sometimes called fanout

# Delay in a Logic Gate

- Express delays in process-independent unit

$$d = \frac{d_{abs}}{\tau}$$

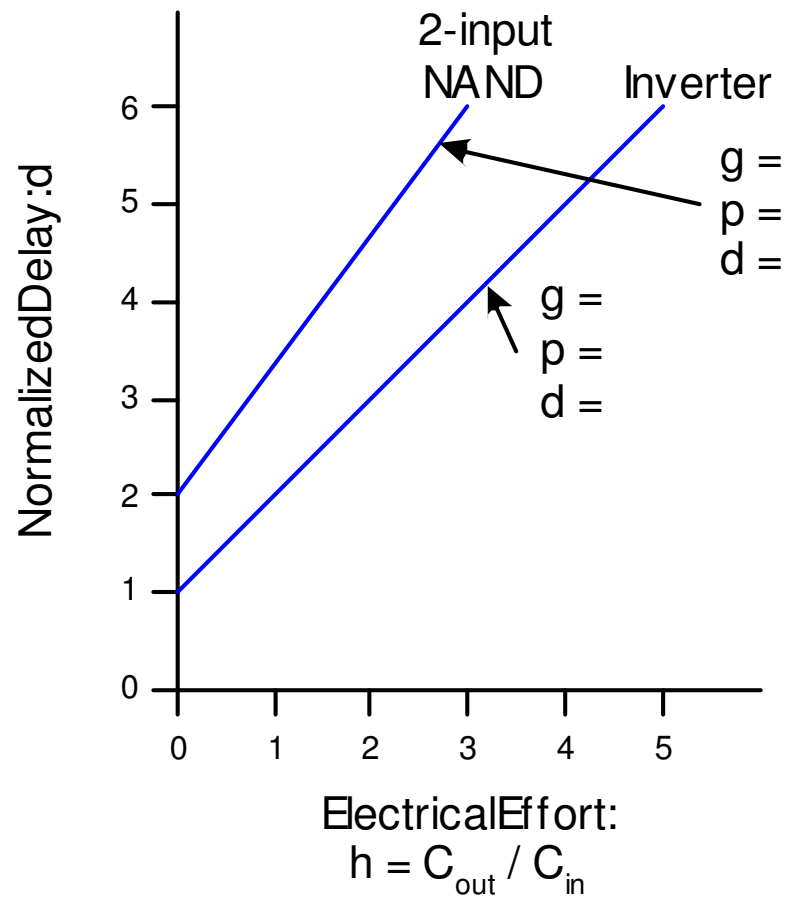
- Delay has two components

$$d = f + p$$

- Parasitic delay  $p$ 
  - Represents delay of gate driving no load
  - Set by internal parasitic capacitance

# Delay Plots

$$d = f + p$$
$$= gh + p$$

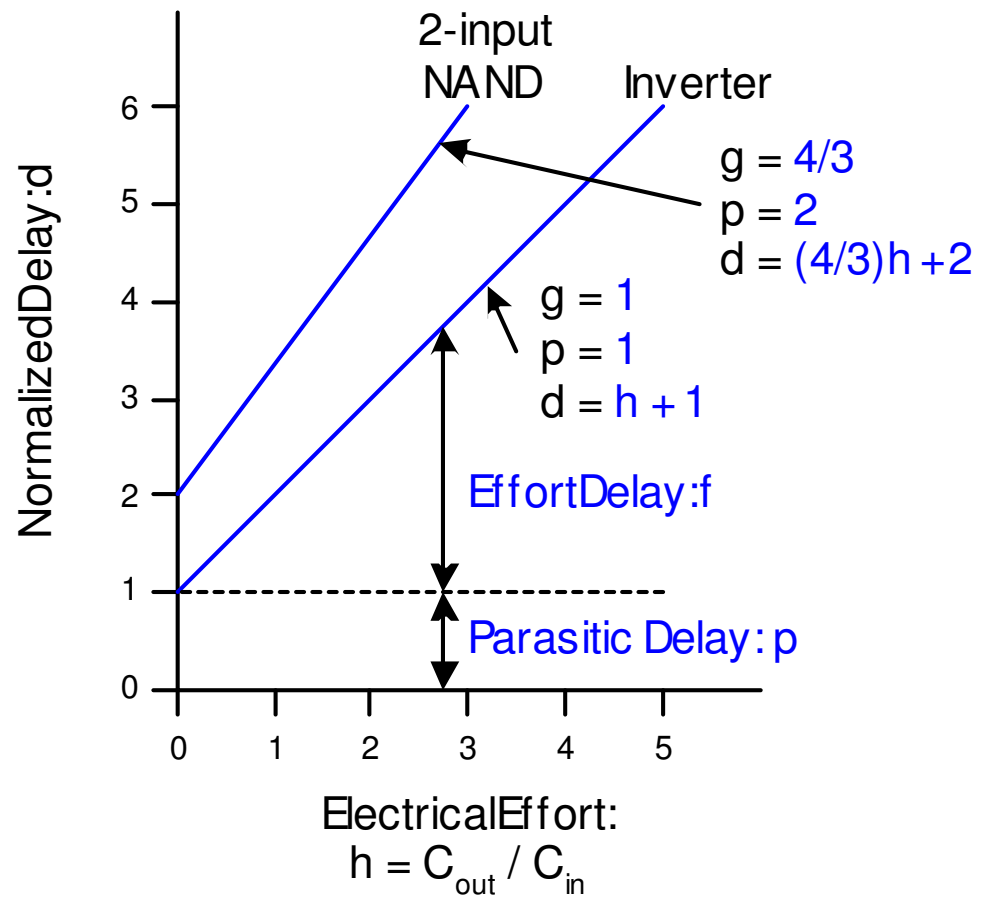


# Delay Plots

$$d = f + p$$

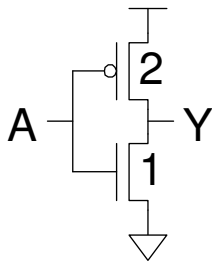
$$= gh + p$$

- What about NOR2?

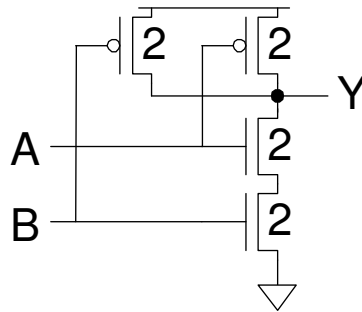


# Computing Logical Effort

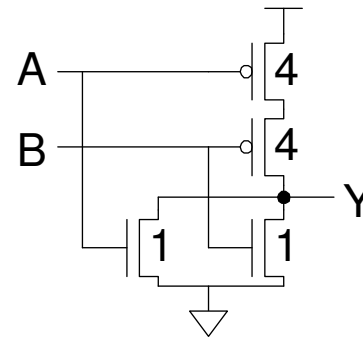
- DEF: *Logical effort is the ratio of the input capacitance of a gate to the input capacitance of an inverter delivering the same output current.*
- Measure from delay vs. fanout plots
- Or estimate by counting transistor widths



$$C_{in} = 3$$
$$g = 3/3$$



$$C_{in} = 4$$
$$g = 4/3$$



$$C_{in} = 5$$
$$g = 5/3$$

# Catalog of Gates

- Logical effort of common gates

Gate type	Number of inputs				
	1	2	3	4	n
Inverter	1				
NAND		$4/3$	$5/3$	$6/3$	$(n+2)/3$
NOR		$5/3$	$7/3$	$9/3$	$(2n+1)/3$
Tristate / mux	2	2	2	2	2

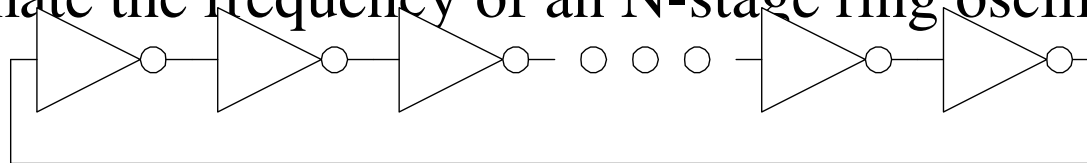
# Catalog of Gates

- Parasitic delay of common gates
  - In multiples of  $p_{inv}$  ( $\approx 1$ )

Gate type	Number of inputs				
	1	2	3	4	n
Inverter	1				
NAND		2	3	4	n
NOR		2	3	4	n
Tristate / mux	2	4	6	8	2n
XOR, XNOR		4	6	8	

# Example: Ring Oscillator

- Estimate the frequency of an N-stage ring oscillator



Logical Effort:  $g =$

Electrical Effort:  $h =$

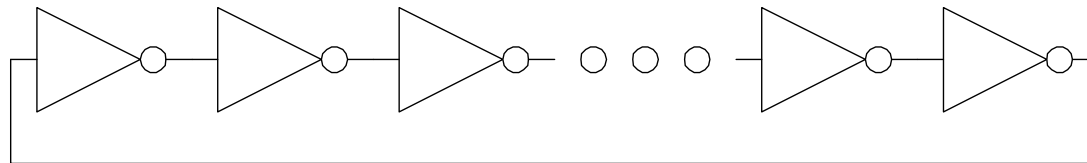
Parasitic Delay:  $p =$

Stage Delay:  $d =$

Frequency:  $f_{\text{osc}} =$

# Example: Ring Oscillator

- Estimate the frequency of an N-stage ring oscillator



Logical Effort:  $g = 1$

Electrical Effort:  $h = 1$

Parasitic Delay:  $p = 1$

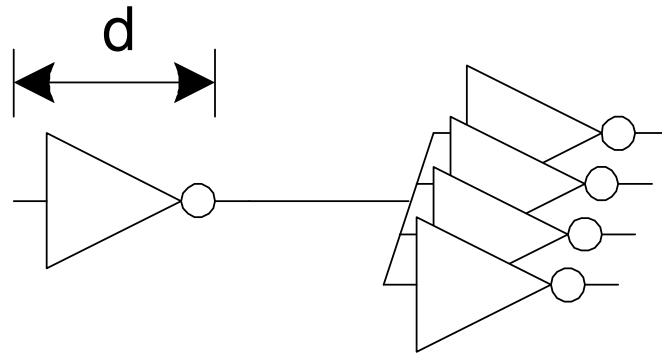
Stage Delay:  $d = 2$

Frequency:  $f_{\text{osc}} = 1/(2*N*d) = 1/4N$

31 stage ring oscillator in  
0.6  $\mu\text{m}$  process has  
frequency of  $\sim 200$  MHz

# Example: FO4 Inverter

- Estimate the delay of a fanout-of-4 (FO4) inverter



Logical Effort:  $g =$

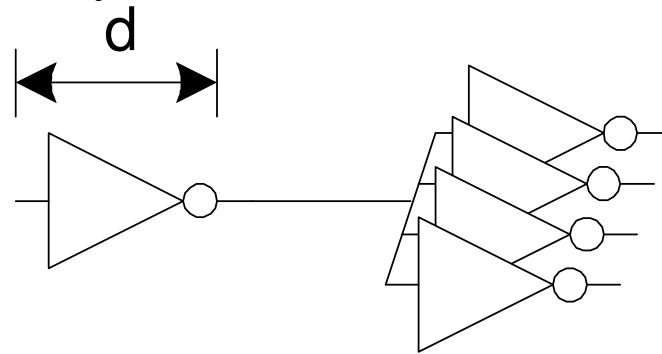
Electrical Effort:  $h =$

Parasitic Delay:  $p =$

Stage Delay:  $d =$

# Example: FO4 Inverter

- Estimate the delay of a fanout-of-4 (FO4) inverter



Logical Effort:  $g = 1$

Electrical Effort:  $h = 4$

Parasitic Delay:  $p = 1$

Stage Delay:  $d = 5$

The FO4 delay is about

200 ps in 0.6  $\mu\text{m}$  process

60 ps in a 180 nm process

$f/3$  ns in an  $f$   $\mu\text{m}$  process

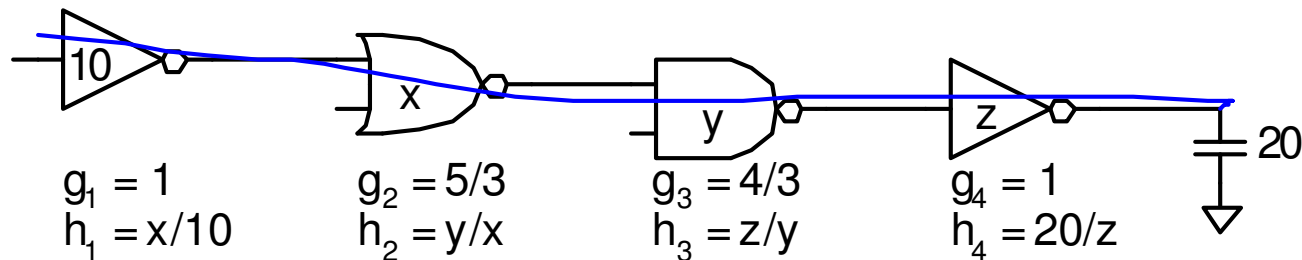
# Multistage Logic Networks

- Logical effort generalizes to multistage networks

- *Path Logical Effort*  $G = \prod g_i$

- *Path Electrical Effort*  $H = \frac{C_{\text{out-path}}}{C_{\text{in-path}}}$

- *Path Effort*  $F = \prod f_i = \prod g_i h_i$

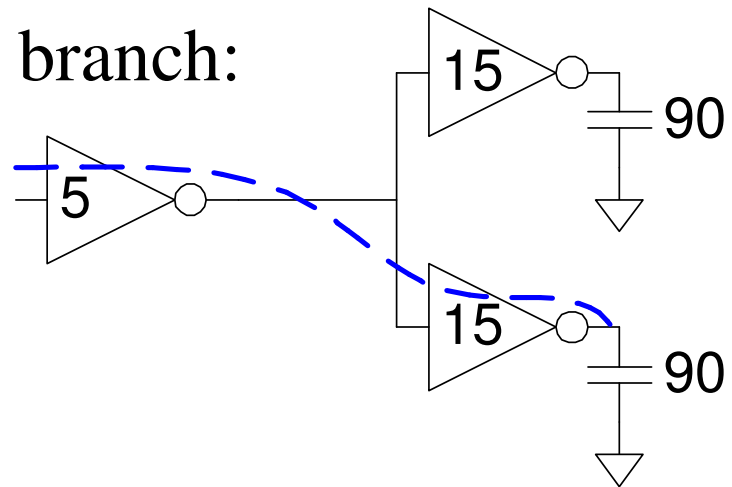


# Multistage Logic Networks

- Logical effort generalizes to multistage networks
- *Path Logical Effort*  $G = \prod g_i$
- *Path Electrical Effort*  $H = \frac{C_{out-path}}{C_{in-path}}$
- *Path Effort*  $F = \prod f_i = \prod g_i h_i$
- Can we write  $F = GH$ ?

# Paths that Branch

- No! Consider paths that branch:



G =

H =

GH =

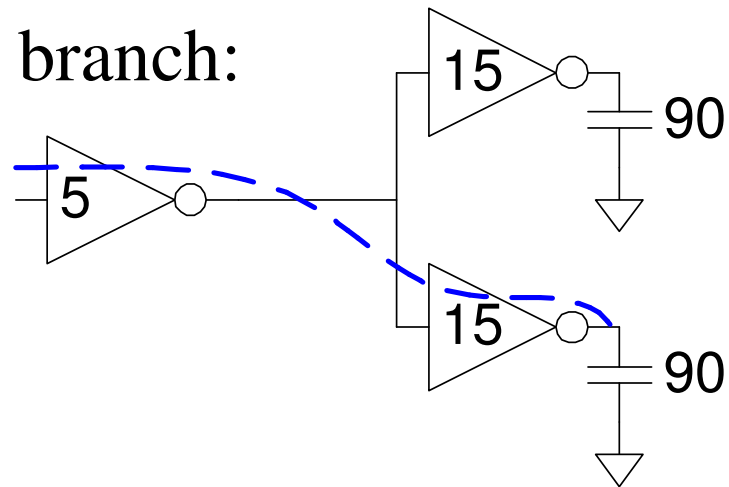
$h_1$  =

$h_2$  =

F = GH?

# Paths that Branch

- No! Consider paths that branch:



$$G = 1$$

$$H = 90 / 5 = 18$$

$$GH = 18$$

$$h_1 = (15 + 15) / 5 = 6$$

$$h_2 = 90 / 15 = 6$$

$$F = g_1 g_2 h_1 h_2 = 36 = 2GH$$

# Branching Effort

- Introduce *branching effort*
  - Accounts for branching between stages in path

$$b = \frac{C_{\text{on path}} + C_{\text{off path}}}{C_{\text{on path}}}$$

$$B = \prod b_i$$

Note:

$$\prod h_i = BH$$

- Now we compute the path effort
  - $F = GBH$

# Multistage Delays

- Path Effort Delay  $D_F = \sum f_i$
- Path Parasitic Delay  $P = \sum p_i$
- Path Delay  $D = \sum d_i = D_F + P$

# Designing Fast Circuits

$$D = \sum d_i = D_F + P$$

- Delay is smallest when each stage bears same effort

$$\hat{f} = g_i h_i = F^{\frac{1}{N}}$$

- Thus minimum delay of N stage path is

$$D = NF^{\frac{1}{N}} + P$$

- This is a **key** result of logical effort
  - Find fastest possible delay
  - Doesn't require calculating gate sizes

# Gate Sizes

- How wide should the gates be for least delay?

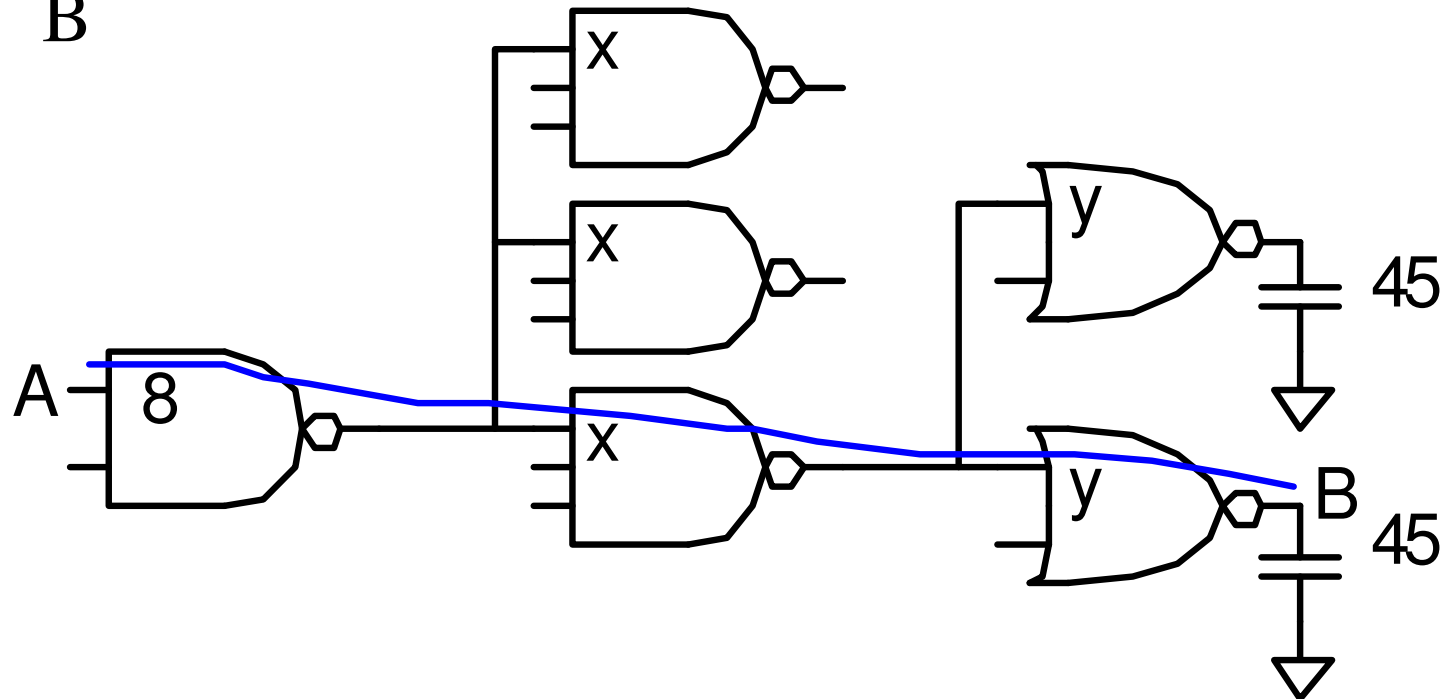
$$\hat{f} = gh = g \frac{C_{out}}{C_{in}}$$

$$\Rightarrow C_{in_i} = \frac{g_i C_{out_i}}{\hat{f}}$$

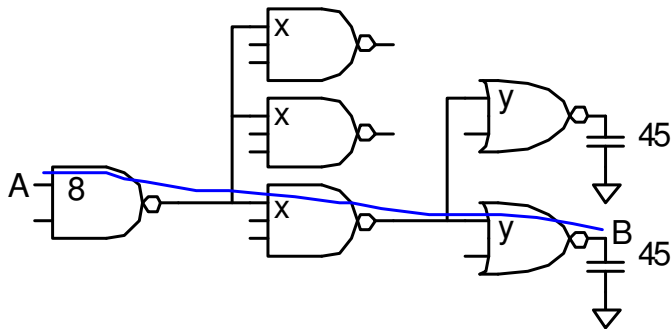
- Working backward, apply capacitance transformation to find input capacitance of each gate given load it drives.
- Check work by verifying input cap spec is met.

# Example: 3-stage path

- Select gate sizes  $x$  and  $y$  for least delay from  $A$  to  $B$



# Example: 3-stage path



Logical Effort  $G =$

Electrical Effort  $H =$

Branching Effort  $B =$

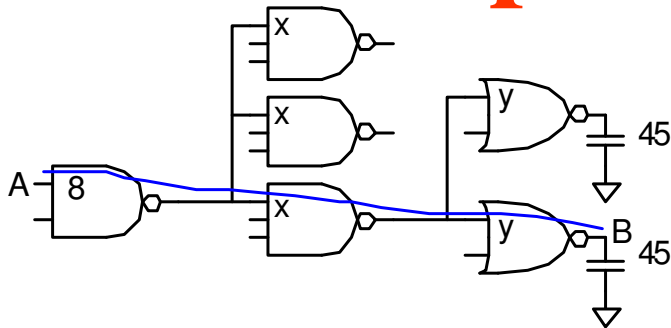
Path Effort  $\hat{F} =$

Best Stage Effort  $\hat{f} =$

Parasitic Delay  $P =$

Delay  $D =$

# Example: 3-stage path



Logical Effort

$$G = (4/3) * (5/3) * (5/3) = 100/27$$

Electrical Effort

$$H = 45/8$$

Branching Effort

$$B = 3 * 2 = 6$$

Path Effort

$$F = GBH = 125$$

Best Stage Effort

$$\hat{f} = \sqrt[3]{F} = 5$$

Parasitic Delay

$$P = 2 + 3 + 2 = 7$$

Delay

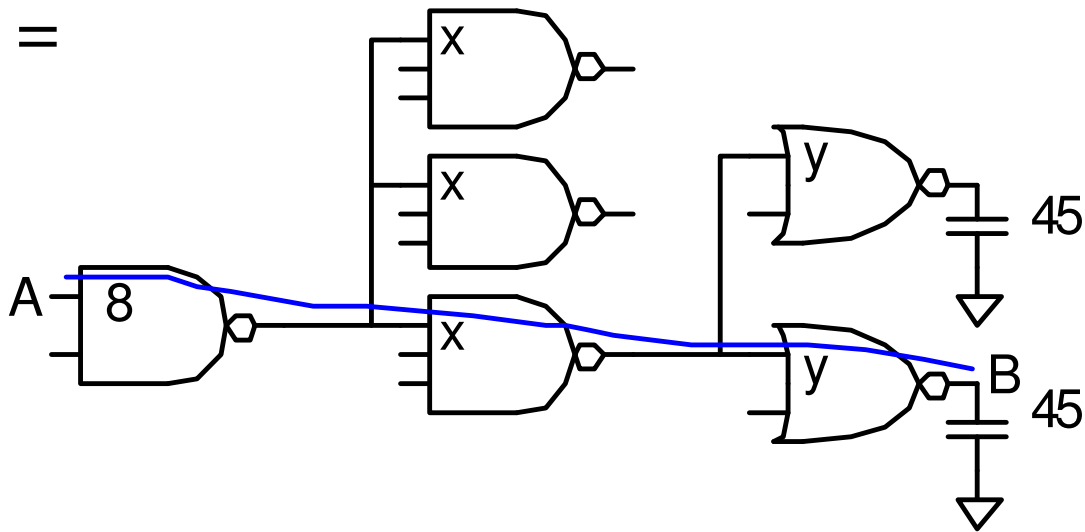
$$D = 3 * 5 + 7 = 22 = 4.4 \text{ FO4}$$

# Example: 3-stage path

- Work backward for sizes

$y =$

$x =$

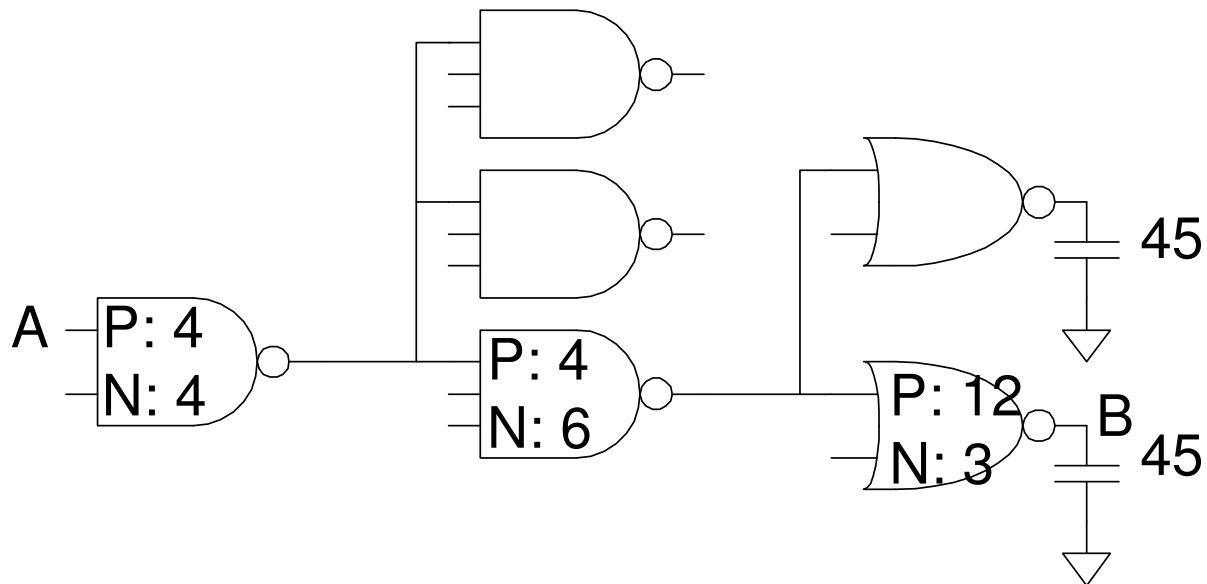


# Example: 3-stage path

- Work backward for sizes

$$y = 45 * (5/3) / 5 = 15$$

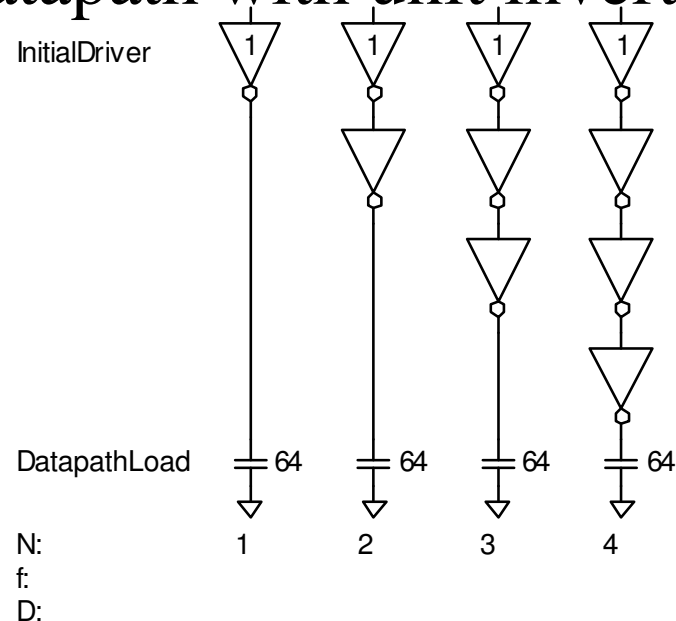
$$x = (15*2) * (5/3) / 5 = 10$$



# Best Number of Stages

- How many stages should a path use?
  - Minimizing number of stages is not always fastest
- Example: drive 64-bit datapath with unit inverter

D =

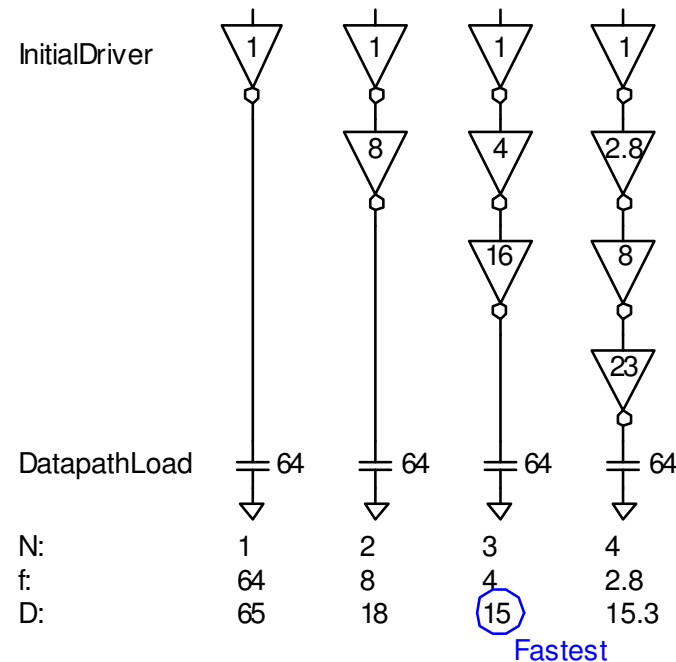


# Best Number of Stages

- How many stages should a path use?
  - Minimizing number of stages is not always fastest
- Example: drive 64-bit datapath with unit inverter

$$D = NF^{1/N} + P$$

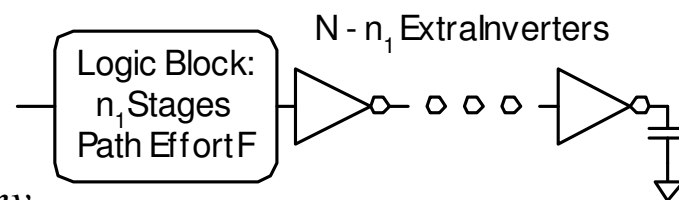
$$= N(64)^{1/N} + N$$



# Derivation

- Consider adding inverters to end of path
  - How many give least delay?

$$D = NF^{\frac{1}{N}} + \sum_{i=1}^{n_1} p_i + (N - n_1) p_{inv}$$



$$\frac{\partial D}{\partial N} = -F^{\frac{1}{N}} \ln F^{\frac{1}{N}} + F^{\frac{1}{N}} + p_{inv} = 0$$

- Define best stage effort  $\rho = F^{\frac{1}{N}}$

$$p_{inv} + \rho(1 - \ln \rho) = 0$$

# Best Stage Effort

$$p_{inv} + \rho(1 - \ln \rho) = 0$$

- has no closed-form solution
- Neglecting parasitics ( $p_{inv} = 0$ ), we find  $\rho = 2.718$  (e)
- For  $p_{inv} = 1$ , solve numerically for  $\rho = 3.59$

# Review of Definitions

Term	Stage	Path
number of stages	1	$N$
logical effort	$g$	$G = \prod g_i$
electrical effort	$h = \frac{C_{\text{out}}}{C_{\text{in}}}$	$H = \frac{C_{\text{out-path}}}{C_{\text{in-path}}}$
branching effort	$b = \frac{C_{\text{on-path}} + C_{\text{off-path}}}{C_{\text{on-path}}}$	$B = \prod b_i$
effort	$f = gh$	$F = GBH$
effort delay	$f$	$D_F = \sum f_i$
parasitic delay	$p$	$P = \sum p_i$
delay	$d = f + p$	$D = \sum d_i = D_F + P$

# Method of Logical Effort

1) Compute path effort

$$F = GBH$$

2) Estimate best number of stages

$$N = \log_4 F$$

3) Sketch path with N stages

4) Estimate least delay

$$D = NF^{\frac{1}{N}} + P$$

5) Determine best stage effort

$$\hat{f} = F^{\frac{1}{N}}$$

6) Find gate sizes

$$C_{in_i} = \frac{g_i C_{out_i}}{\hat{f}}$$

# Limits of Logical Effort

- Chicken and egg problem
  - Need path to compute  $G$
  - But don't know number of stages without  $G$
- Simplistic delay model
  - Neglects input rise time effects
- Interconnect
  - Iteration required in designs with wire
- Maximum speed only
  - Not minimum area/power for constrained delay

# Summary

- Logical effort is useful for thinking of delay in circuits
  - Numeric logical effort characterizes gates
  - NANDs are faster than NORs in CMOS
  - Paths are fastest when effort delays are  $\sim 4$
  - Path delay is weakly sensitive to stages, sizes
  - But using fewer stages doesn't mean faster paths
  - Delay of path is about  $\log_4 F$  FO4 inverter delays
  - Inverters and NAND2 best for driving large caps
- Provides language for discussing fast circuits
  - But requires practice to master

# Power and Energy

- Power is drawn from a voltage source attached to the  $V_{DD}$  pin(s) of a chip.

$$P(t) = i_{DD}(t)V_{DD}$$

- Instantaneous Power:

$$E = \int_0^T P(t)dt = \int_0^T i_{DD}(t)V_{DD}dt$$

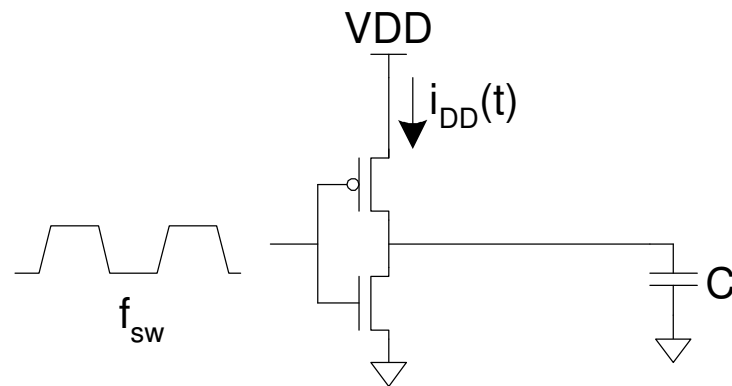
- Energy:

$$P_{\text{avg}} = \frac{E}{T} = \frac{1}{T} \int_0^T i_{DD}(t)V_{DD}dt$$

- Average Power:

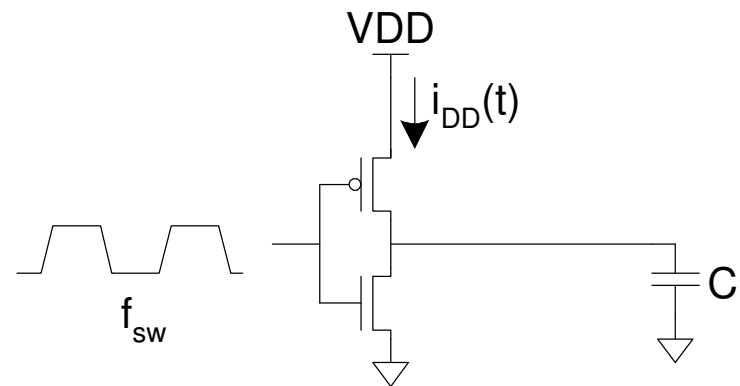
# Dynamic Power

- Dynamic power is required to charge and discharge load capacitances when transistors switch.
- One cycle involves a rising and falling output.
- On rising output, charge  $Q = CV_{DD}$  is required
- On falling output, charge is dumped to GND
- This repeats  $T_{f_{sw}}$  times over an interval of  $T$



# Dynamic Power Cont.

$$P_{\text{dynamic}} =$$



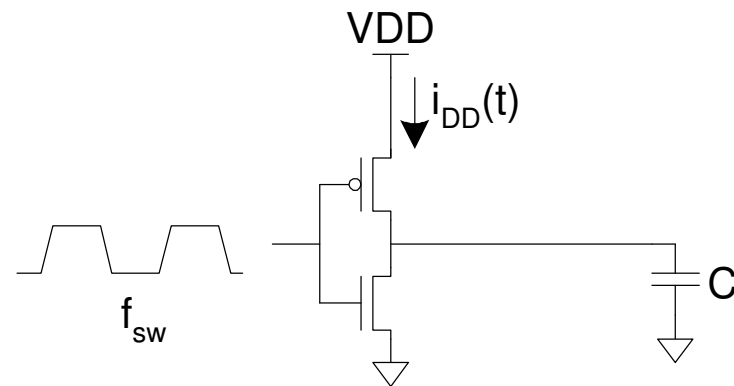
# Dynamic Power Cont.

$$P_{\text{dynamic}} = \frac{1}{T} \int_0^T i_{DD}(t) V_{DD} dt$$

$$= \frac{V_{DD}}{T} \int_0^T i_{DD}(t) dt$$

$$= \frac{V_{DD}}{T} [T f_{\text{sw}} C V_{DD}]$$

$$= C V_{DD}^2 f_{\text{sw}}$$



# Activity Factor

- Suppose the system clock frequency =  $f$
- Let  $f_{sw} = \alpha f$ , where  $\alpha$  = activity factor
  - If the signal is a clock,  $\alpha = 1$
  - If the signal switches once per cycle,  $\alpha = 1/2$
  - Dynamic gates:
    - Switch either 0 or 2 times per cycle,  $\alpha = 1/2$
  - Static gates:
    - Depends on design, but typically  $\alpha = 0.1$

- Dynamic power:  $P_{dynamic} = \alpha C V_{DD}^2 f$

# Short Circuit Current

- When transistors switch, both nMOS and pMOS networks may be momentarily ON at once
- Leads to a blip of “short circuit” current.
- $< 10\%$  of dynamic power if rise/fall times are comparable for input and output

# Example

- 200 Mtransistor chip
  - 20M logic transistors
    - Average width:  $12 \lambda$
  - 180M memory transistors
    - Average width:  $4 \lambda$
  - 1.2 V 100 nm process
  - $C_g = 2 \text{ fF}/\mu\text{m}$

# Dynamic Example

- Static CMOS logic gates: activity factor = 0.1
- Memory arrays: activity factor = 0.05 (many banks!)
- Estimate dynamic power consumption per MHz. Neglect wire capacitance and short-circuit current.

# Dynamic Example

- Static CMOS logic gates: activity factor = 0.1
- Memory arrays: activity factor = 0.05 (many banks!)
- Estimate dynamic power consumption per MHz. Neglect wire capacitance.

$$C_{\text{logic}} = (20 \times 10^6)(12\lambda)(0.05\mu\text{m} / \lambda)(2\text{fF} / \mu\text{m}) = 24\text{nF}$$

$$C_{\text{mem}} = (180 \times 10^6)(4\lambda)(0.05\mu\text{m} / \lambda)(2\text{fF} / \mu\text{m}) = 72\text{nF}$$

$$P_{\text{dynamic}} = [0.1C_{\text{logic}} + 0.05C_{\text{mem}}](1.2)^2 f = 8.6\text{ mW/MHz}$$

# Static Power

- Static power is consumed even when chip is quiescent.
  - Ratioed circuits burn power in fight between ON transistors
  - Leakage draws power from nominally OFF devices

$$I_{ds} = I_{ds0} e^{\frac{V_{gs} - V_t}{nV_T}} \left[ 1 - e^{\frac{-V_{ds}}{V_T}} \right]$$

$$V_t = V_{t0} - \eta V_{ds} + \gamma \left( \sqrt{\phi_s + V_{sb}} - \sqrt{\phi_s} \right)$$

# Ratio Example

- The chip contains a 32 word x 48 bit ROM
  - Uses pseudo-nMOS decoder and bitline pullups
  - On average, one wordline and 24 bitlines are high
- Find static power drawn by the ROM
  - $\beta = 75 \mu\text{A}/\text{V}^2$
  - $V_{tp} = -0.4\text{V}$

# Ratio Example

- The chip contains a 32 word x 48 bit ROM
  - Uses pseudo-nMOS decoder and bitline pullups
  - On average, one wordline and 24 bitlines are high
- Find static power drawn by the ROM
  - $\beta = 75 \mu\text{A}/\text{V}^2$
  - $V_{tp} = -0.4\text{V}$

- Solution:
$$I_{\text{pull-up}} = \beta \frac{(V_{DD} - |V_{tp}|)^2}{2} = 24\mu\text{A}$$
$$P_{\text{pull-up}} = V_{DD} I_{\text{pull-up}} = 29\mu\text{W}$$
$$P_{\text{static}} = (31 + 24)P_{\text{pull-up}} = 1.6 \text{ mW}$$

# Leakage Example

- The process has two threshold voltages and two oxide thicknesses.
- Subthreshold leakage:
  - 20 nA/ $\mu\text{m}$  for low  $V_t$
  - 0.02 nA/ $\mu\text{m}$  for high  $V_t$
- Gate leakage:
  - 3 nA/ $\mu\text{m}$  for thin oxide
  - 0.002 nA/ $\mu\text{m}$  for thick oxide
- Memories use low-leakage transistors everywhere
- Gates use low-leakage transistors on 80% of logic

# Leakage Example Cont.

- Estimate static power:

# Leakage Example Cont.

- Estimate static power:

- High leakage:  $(20 \times 10^6)(0.2)(12\lambda)(0.05 \mu m / \lambda) = 2.4 \times 10^6 \mu m$

- Low leakage:  $(20 \times 10^6)(0.8)(12\lambda)(0.05 \mu m / \lambda) +$   
 $(180 \times 10^6)(4\lambda)(0.05 \mu m / \lambda) = 45.6 \times 10^6 \mu m$

$$I_{static} = (2.4 \times 10^6 \mu m) \left[ (20 nA / \mu m) / 2 + (3 nA / \mu m) \right] +$$
$$(45.6 \times 10^6 \mu m) \left[ (0.02 nA / \mu m) / 2 + (0.002 nA / \mu m) \right]$$
$$= 32 mA$$

$$P_{static} = I_{static} V_{DD} = 38 mW$$

# Leakage Example Cont.

- Estimate static power:  $(20 \times 10^6)(0.2)(12\lambda)(0.05 \mu m / \lambda) = 2.4 \times 10^6 \mu m$ 
  - High leakage:  $(20 \times 10^6)(0.8)(12\lambda)(0.05 \mu m / \lambda) +$
  - Low leakage:  $(180 \times 10^6)(4\lambda)(0.05 \mu m / \lambda) = 45.6 \times 10^6 \mu m$

$$I_{static} = (2.4 \times 10^6 \mu m) \left[ (20 nA / \mu m) / 2 + (3 nA / \mu m) \right] + \\ (45.6 \times 10^6 \mu m) \left[ (0.02 nA / \mu m) / 2 + (0.002 nA / \mu m) \right] \\ = 32 mA$$

$$P_{static} = I_{static} V_{DD} = 38 mW$$

- If no low leakage devices,  $P_{static} = 749 mW (!)$

# Low Power Design

- Reduce dynamic power
  - $\alpha$ :
  - C:
  - $V_{DD}$ :
  - f:
- Reduce static power

# Low Power Design

- Reduce dynamic power
  - $\alpha$ : clock gating, sleep mode
  - C:
  - $V_{DD}$ :
  - f:
- Reduce static power

# Low Power Design

- Reduce dynamic power
  - $\alpha$ : clock gating, sleep mode
  - $C$ : small transistors (esp. on clock), short wires
  - $V_{DD}$ :
  - $f$ :
- Reduce static power

# Low Power Design

- Reduce dynamic power
  - $\alpha$ : clock gating, sleep mode
  - $C$ : small transistors (esp. on clock), short wires
  - $V_{DD}$ : lowest suitable voltage
  - $f$ :
- Reduce static power

# Low Power Design

- Reduce dynamic power
  - $\alpha$ : clock gating, sleep mode
  - C: small transistors (esp. on clock), short wires
  - $V_{DD}$ : lowest suitable voltage
  - f: lowest suitable frequency
- Reduce static power

# Low Power Design

- Reduce dynamic power
  - $\alpha$ : clock gating, sleep mode
  - C: small transistors (esp. on clock), short wires
  - $V_{DD}$ : lowest suitable voltage
  - f: lowest suitable frequency
- Reduce static power
  - Selectively use ratioed circuits
  - Selectively use low  $V_t$  devices
  - Leakage reduction:  
stacked devices, body bias, low temperature