

*Lecture*

# 4

ECE M216A

# Logical Effort

**Prof. Dejan Marković**

ee216a@gmail.com

# Concept of Logical Effort

---

Instead of running lots of simulations

- **Simplified calculation of delay**

$$\text{Delay} \propto R_{\text{gate}}(C_{\text{load}} + C_{\text{self}}) = R_{\text{gate}}C_{\text{load}} + R_{\text{gate}}C_{\text{self}}$$

- **Normalize to a technology time constant,  $\tau$**

Logical effort delay equation ( $D = d \cdot \tau$ ):

The diagram shows the equation  $d = f + p$  enclosed in a red rectangular box. Below the box, three red labels are positioned: "Normalized delay" on the left, "Effort delay" in the center, and "Parasitic delay" on the right. Three gray arrows point upwards from these labels to the equation: one from "Normalized delay" to the variable  $d$ , one from "Effort delay" to the variable  $f$ , and one from "Parasitic delay" to the variable  $p$ .

$$d = f + p$$

Normalized delay      Effort delay      Parasitic delay

# The Logical Effort Way of Thinking

---

- Gate delay we used until now:

$$Delay = 0.69 R_{gate} (C_{par} + C_{out})$$

- Another way to write this formula:

$$Delay = 0.69 R_{gate} C_{in,gate} \left( \frac{C_{par}}{C_{in,gate}} + \frac{C_{out}}{C_{in,gate}} \right)$$

$$Delay = 0.69 \tau_{gate} \left( \frac{C_{par}}{C_{in,gate}} + \frac{C_{out}}{C_{in,gate}} \right)$$

# Now Normalize the Delay

---

Strategy: normalize to the time constant of an inverter

- 1** • Method 1: normalize to  $\tau_{INV} = R_{INV}C_{in,INV}$

$$\frac{Delay}{\tau_{INV}} = \frac{\tau_{gate}}{\tau_{INV}} \left( \gamma_{gate} + \frac{C_{out}}{C_{in,gate}} \right)$$

- 2** • Method 2: normalize to  $\tau_{p0,INV} = R_{INV}C_{par,INV}$

$$\frac{Delay}{t_{p0,INV}} = \frac{\tau_{gate}}{\tau_{p0,INV}} \left( \gamma_{gate} + \frac{C_{out}}{C_{in,gate}} \right)$$

# We Will Use **Method 1**

---

Normalize to the time constant of an inverter

$$\tau_{INV} = R_{INV}C_{in,INV}$$

$$\frac{Delay}{\tau_{INV}} = \frac{\tau_{gate}}{\tau_{INV}} \left( \gamma_{gate} + \frac{C_{out}}{C_{in,gate}} \right)$$

- Used in the original logical effort theory
- Doesn't really matter: just a constant

# Normalized Delay

---

$$\frac{\text{Delay}}{\tau_{INV}} = d = \frac{\tau_{gate}}{\tau_{INV}} \left( \gamma_{gate} + \frac{C_{out}}{C_{in,gate}} \right)$$



Normalized delay:  $d = g (\gamma_{gate} + h)$



Even simpler:  $d = g \cdot h + p \cdot \gamma$

**Logical  
effort**

**Fanout**

**Parasitic  
effort**

# Logical Effort Terms: Mathematical View

$$\frac{\text{Delay}}{\tau_{INV}} = d = \frac{\tau_{gate}}{\tau_{INV}} \left( \gamma_{gate} + \frac{C_{out}}{C_{in,gate}} \right)$$



$$d = g \cdot h + p \cdot \gamma$$

Logical effort

$$g = \frac{R_{gate} \cdot C_{in,gate}}{R_{INV} \cdot C_{in,INV}}$$

Fanout

$$h = \frac{C_{out}}{C_{in,gate}}$$

Parasitic effort

$$p = \frac{C_{par,gate}}{C_{par,INV}}$$

# Logical Effort (g): Intuitive View

---

$$g = \frac{R_{gate} \cdot C_{in, gate}}{R_{INV} \cdot C_{in, INV}}$$

- 1** •  $R_{on}$  ratio for equal  $C_{in}$
- 2** •  $C_{in}$  ratio for equal  $R_{on}$



# Logical Effort (g): Intuitive View

---

$$g = \frac{R_{gate} \cdot C_{in, gate}}{R_{INV} \cdot C_{in, INV}}$$

**1** •  $R_{on}$  ratio for equal  $C_{in}$

**2** •  $C_{in}$  ratio for equal  $R_{on}$

## Cost of doing logic

# **g is NOT a Function of Gate Size**

---

- **Unitless inherent characteristic of the gate**
- **A function of the construction of the gate**  
(topology and relative size of transistors)
- **The cost of implementing the function**

# Fanout (h)

---

$$h = \frac{C_{out}}{C_{in,gate}}$$

**Ratio of gate caps (only)**

Diffusion counts in the p term

# Parasitic Effort (p): Intuitive View

Note: parasitic delay =  $p \cdot \gamma$

$$p = \frac{C_{par,gate}}{C_{par,INV}}$$

Ratio of parasitic caps

assuming equal  $R_{on}$

$$p = g \frac{\gamma_{gate}}{\gamma_{INV}} = \frac{RC_{in,gate}}{RC_{in,INV}} \cdot \frac{C_{par,gate}}{C_{in,gate}} \cdot \frac{C_{in,INV}}{C_{par,INV}} = \frac{C_{par,gate}}{C_{par,INV}}$$

# Computing Logical Effort: $g$

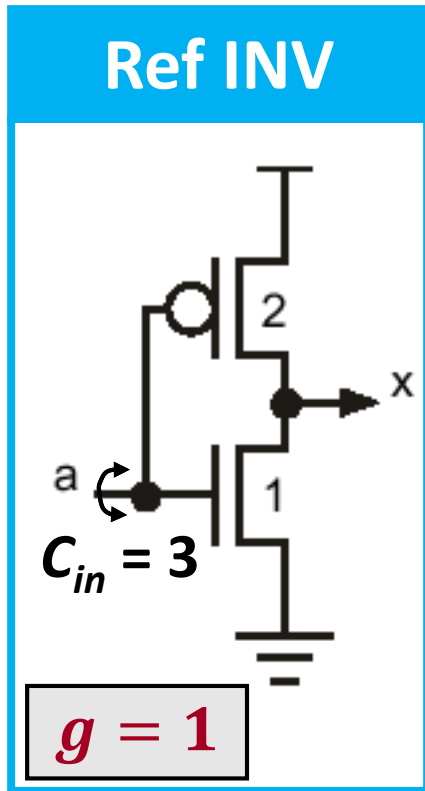
---

1. Choose an input, find total device with of that input
2. Find  $W_p$  of a single device that has equivalent drive strength as the gate's pull-up of that input
3. For a reference INV (given  $\beta$ ) and  $W_p$  from Step 2, find the total gate widths of the inverter devices
4. Divide Step 2 by Step 3 to determine  $g_{up}$
5. Repeat Steps 2-4 for pull-down for  $g_{down}$

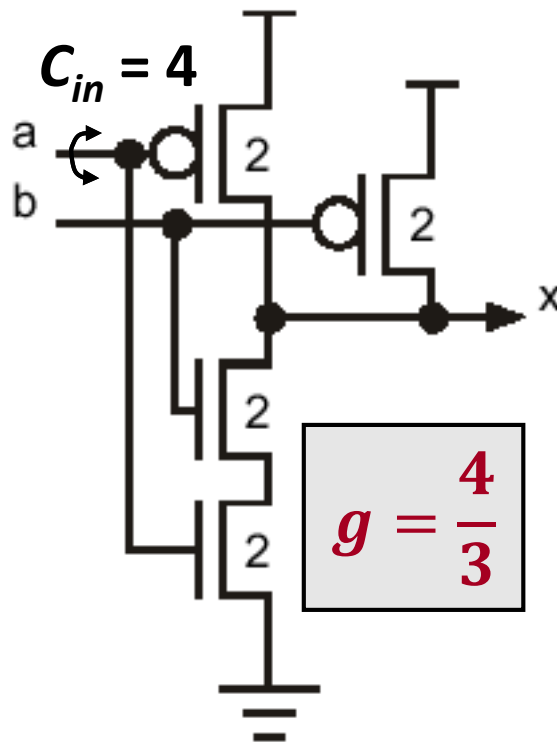
[Under what  $\beta$  is  $g_{up} = g_{down}$ ?]

# Example: Calculating Logical Effort

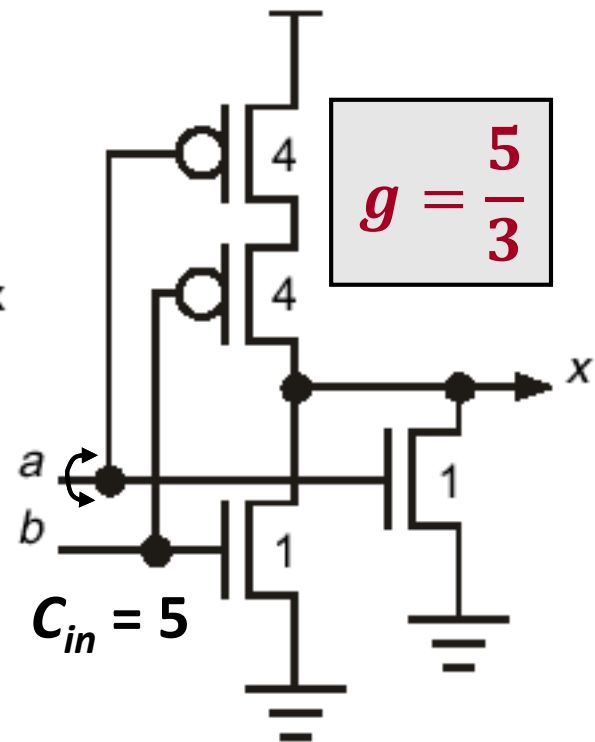
- Ratio of the gate input cap to the input cap of an equal-strength (same output current) inverter



(by definition)



NAND2



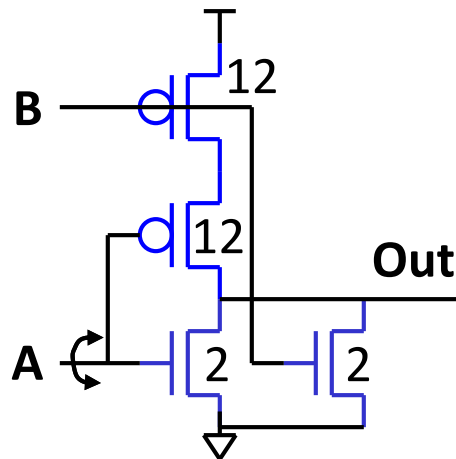
NOR2

## Example 4.1a: Logical Effort, NOR ( $\beta = 3$ )

**Assumption:**

**2-input NOR**

- $\beta = k = 3$



⇒ **Equivalent INV**

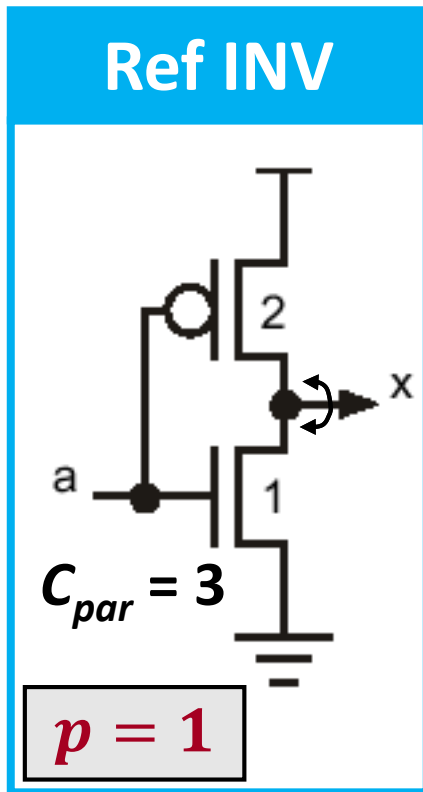
- $W_p : W_n = 6 : 2$
- $C_{\text{gate,INV}} = 8$

⇒ •  $C_{\text{gate,NOR}} = 14$

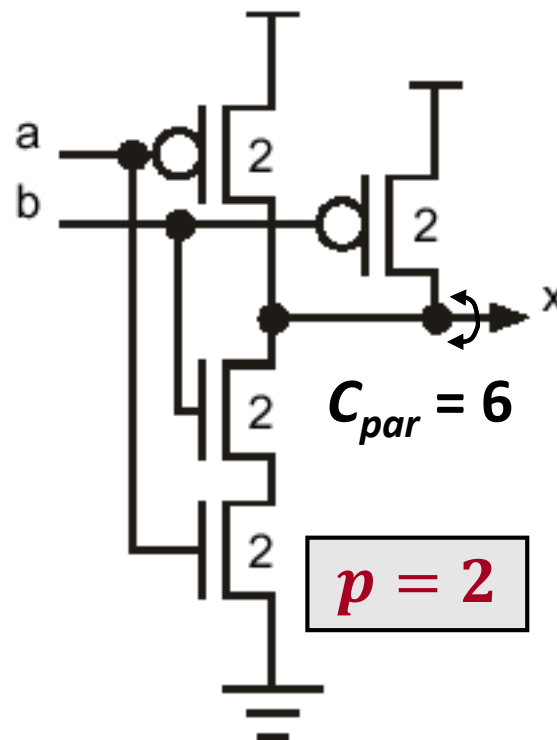
$$g_{\text{NOR}} = \frac{7}{4}$$

# Example: Calculating Parasitic Delay

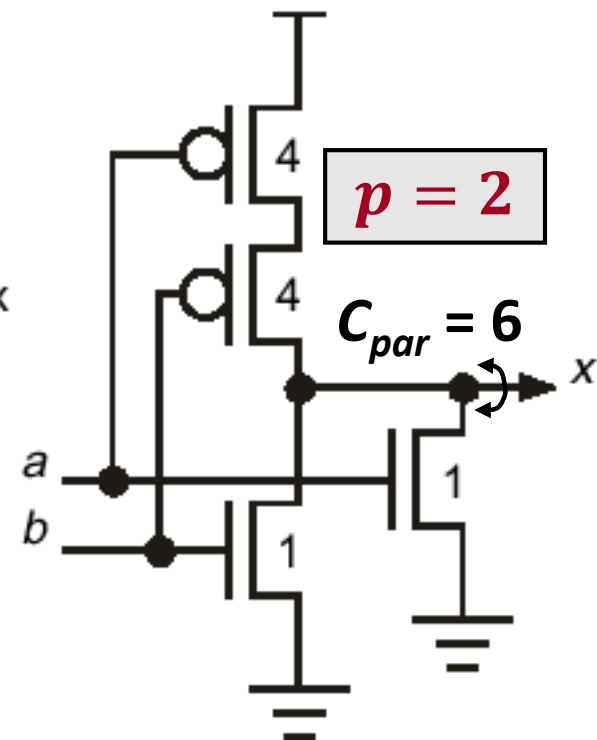
- Ratio of intrinsic cap at the gate output and intrinsic cap at the output of an equivalent inverter



(by definition)



NAND2



NOR2

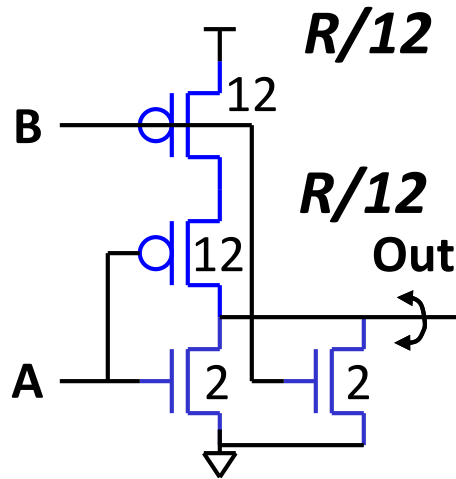


## Example 4.1b: Parasitic Delay, NOR ( $\beta = 3$ )

Assumption:

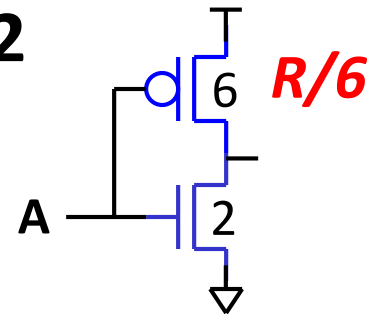
$$\gamma = 0.5$$

$$2R/12 = R/6$$



⇒ Equivalent INV

- $W_p : W_n = 6 : 2$
- $C_{\text{par,INV}} = 8$



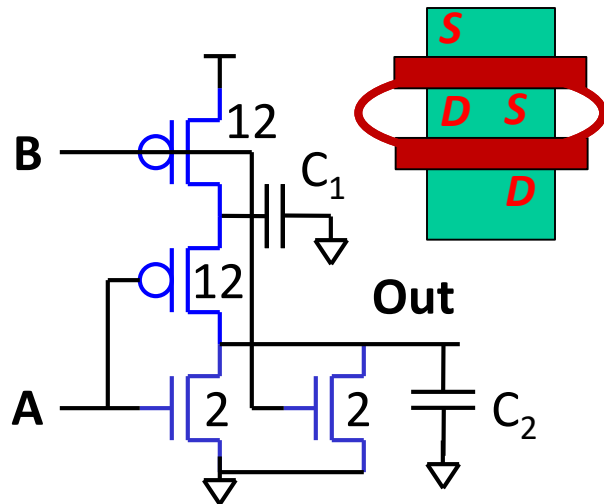
⇒ •  $C_{\text{par,NOR}} = 16$

$$p_{\text{NOR}} = 2$$

## Example 4.1c: Intermediate Nodes? (p)

- One way to account for them is to use an “effective”  $p$
- Input and transition dependent (lots of  $p$ 's to track)

Example: pull up, B input



- $C_1 = 6C_0$  (shared)
- $C_2 = 8C_0$

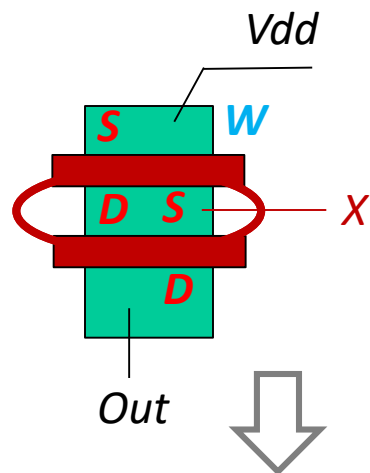
$$t_{par,NOR} \propto \frac{R}{2} C_1 + RC_2$$

$$\Rightarrow p_{UP}(B) = \frac{\frac{C_1}{2} + C_2}{C_{par,INV}}$$

$$p_{UP}(B) = \frac{11}{4} = 2.75$$

# Diffusion Sharing: Stack vs. Finger Layout

## • Stack (W)

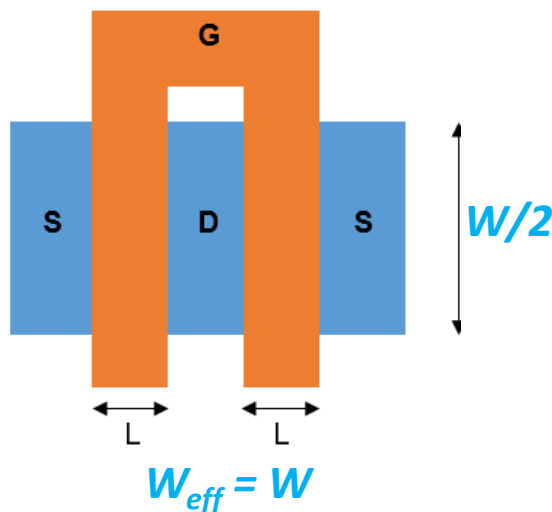


Diffusion caps:

$$C_{out} \sim W$$

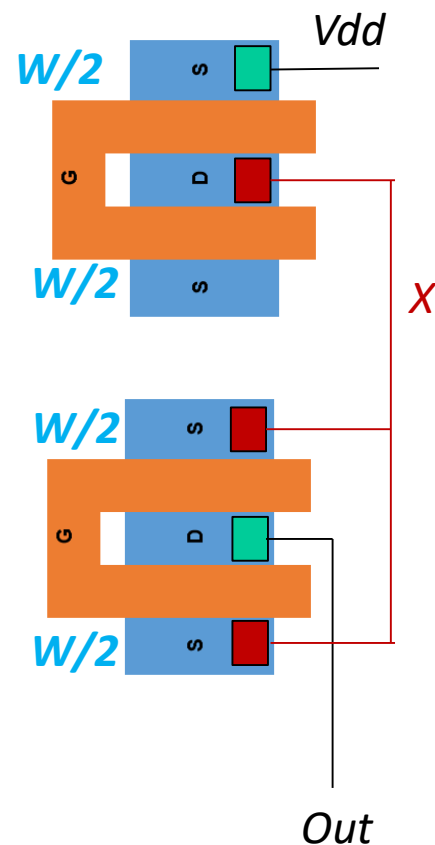
$$C_x \sim W$$

## • Two finger layout



$$W_{eff} = W$$

## • Stack + finger layout



Diffusion caps:

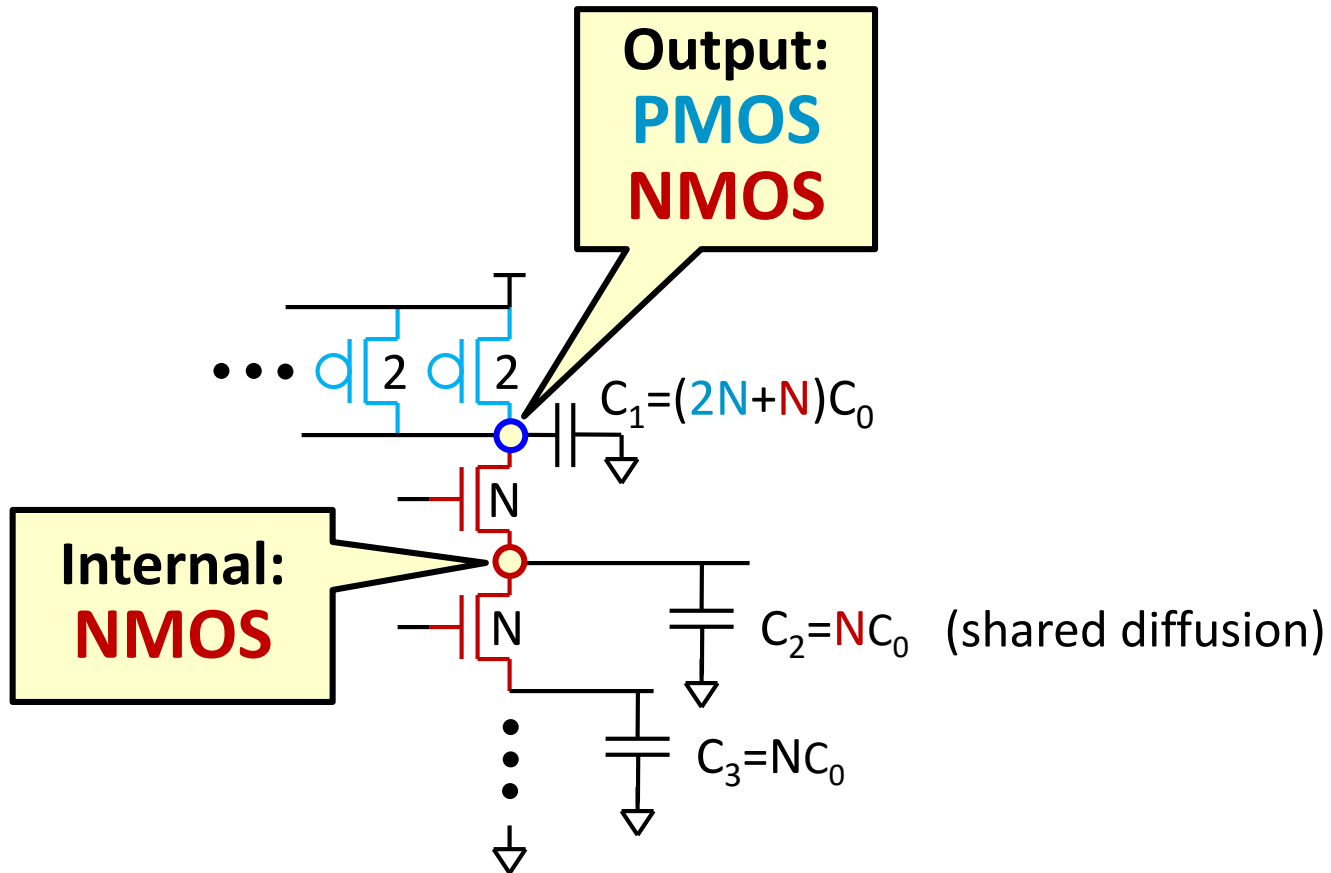
$$C_{out} \sim W/2$$

$$C_x \sim 1.5 * W$$

### Takeaways:

- 1) Finger layout reduces  $C_{par}$  at output
- 2) Finger layout increases  $C_{par}$  at intermediate
- 3) Stacking allows sharing of non-finger layout transistors and results in lower  $C_{par}$
- 4)  $R_{on} \sim 1/W$  and is not affected by D/S sharing

# Generalize N-input NAND




# Generalize N-input NAND

Total pull down parasitic delay

$$t_{par,NAND} \propto R(3NC_0) + \sum_{i=1}^{N-1} \frac{iR}{N} NC_0$$

$$d_{par,NAND} = 3N + \frac{N(N-1)}{2}$$

Large N:  
**BAD NEWS!**

$$p_{NAND} = N + \frac{N(N-1)}{6} \propto N^2$$


- **Even worse for PMOS (NOR)**
  - Reality is even worse since  $C_{GS}$  makes each intermediate node capacitance  $> NC_0$

# A Catalog of Gates

| Gate type   | g for different number of inputs |     |     |     |      |            |
|-------------|----------------------------------|-----|-----|-----|------|------------|
|             | 1                                | 2   | 3   | 4   | 5    | n          |
| Inverter    | 1                                |     |     |     |      |            |
| NAND        |                                  | 4/3 | 5/3 | 6/3 | 7/3  | $(n+2)/3$  |
| NOR         |                                  | 5/3 | 7/3 | 9/3 | 11/3 | $(2n+1)/3$ |
| Multiplexer |                                  | 2   | 2   | 2   | 2    | 2          |
| XOR, XNOR   |                                  | 4   | 12  | 32  |      |            |

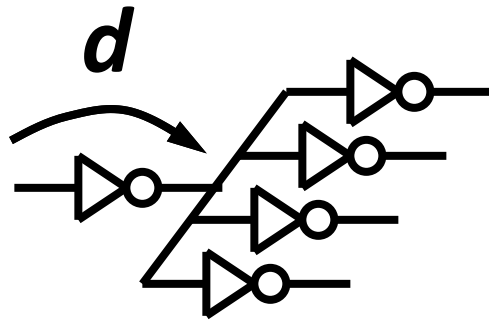
## Assumptions:

- $\beta = k = 2$
- Mux: tri-state INVs shorted together
- XOR: bundled in (a, a')

| Gate type         | Parasitic delay         |
|-------------------|-------------------------|
| Inverter          | $1 \cdot \gamma$        |
| n-input NAND      | $n \cdot \gamma$        |
| n-input NOR       | $n \cdot \gamma$        |
| n-way Multiplexer | $2n \cdot \gamma$       |
| 2-input XOR, XNOR | $n2^{n-1} \cdot \gamma$ |

does not include intermediate nodes

## Example 4.3: Fanout-of-4 Inverter Delay?



$$\gamma = 1$$

Logical Effort :  $g = 1$  (by definition)

Electrical Effort :  $h = C_{\text{out}}/C_{\text{in}} = 4$

Parasitic Delay :  $p = 1$  (by definition)

Stage Delay :  $d = g \cdot h + p \cdot \gamma = 5$

# Logical Effort Recap

---

**Normalized delay:  $d = g \cdot h + p \cdot \gamma$**

- **$g (= C_{\text{in,gate}}/C_{\text{in,INV}})$  is the logical effort**
  - INV sized such that  $R_{\text{INV}} = R_{\text{gate}}$
- **$h (= C_{\text{out}}/C_{\text{in}})$  is the electrical effort (fanout)**
  - $g \cdot h$  is the effort delay (“effective fanout”)
- **$p (= C_{\text{par,gate}}/C_{\text{par,inv}})$  is the parasitic effort**
  - $p \cdot \gamma$  is the parasitic delay
- **May have different  $g, p$** 
  - Per input, and for pull-up/down

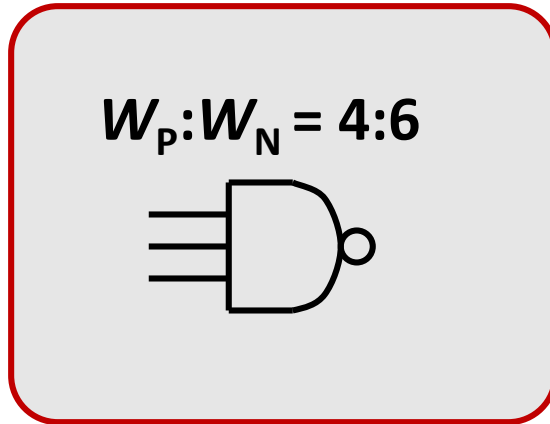


# What About:

- Different inputs
- Reference inverter
- Asymmetry

# A Side Note: Total Gate Effort ( $g_{\text{tot}}$ )

Take total input capacitance:  $g_{\text{tot}} = C_{\text{in,tot}}/C_{\text{INV}}$



## Example: 3-in NAND

- Equivalent INV is 4:2
- **One in:**  $g_{\text{in}} = 10/6 = 5/3$
- **Total:**  $g_{\text{tot}} = 30/6 = 5$

- **$g_{\text{tot}}$  not very useful to calculate delay**
  - But it is an indication of the “cost” of the gate
  - Can be useful in gate mapping (logic synthesis)
    - Which gate is best to use to map a given Boolean expression

# A Note on Asymmetry

---

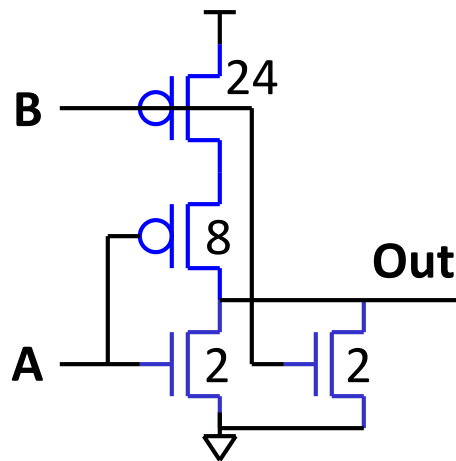
- **The gate examples so far have been symmetric**
  - All inputs are essentially identical
  - P:N ratio is approximately equal to the mobility ratio
    - Same as the reference inverter
- **Inputs of most gates are not symmetric**
  - Different inputs may see different capacitances
  - Even series stacked gates may not be the same size
- **Pull-up and pull-down is rarely equal resistance**
  - Call this “skewed” gates

# Asymmetric Example: Different Stack Sizing

Assumption:

2-input NOR

•  $\beta = k = 3$



$$\frac{1}{8} + \frac{1}{24} = \frac{1}{6}$$

⇒ Equivalent INV

•  $W_p : W_n = 6 : 2$

•  $C_{\text{gate,INV}} = 8$

⇒ •  $C_{\text{gate,A}} = 10$

•  $C_{\text{gate,B}} = 26$

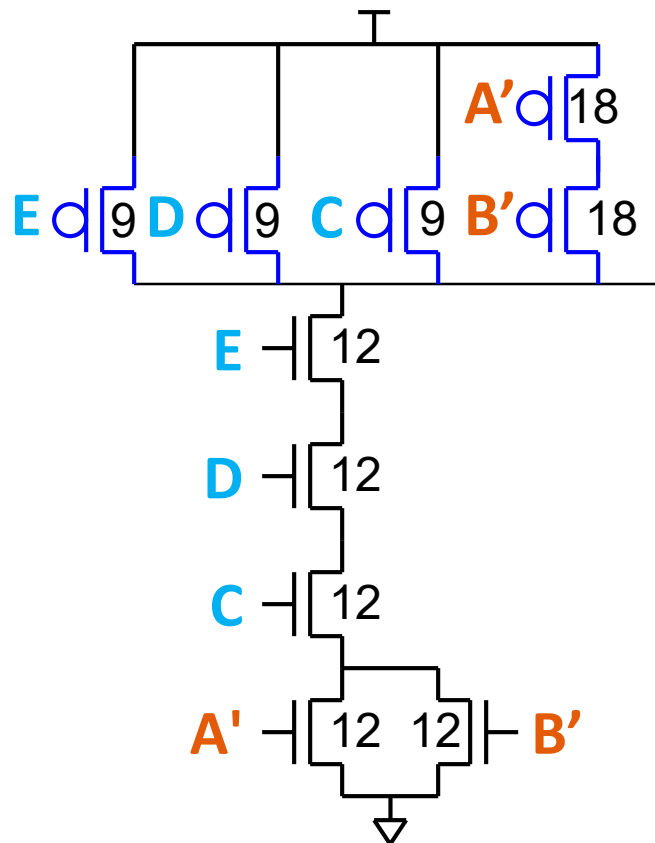
**The larger g is BAD!**  
(More effort to do logic)

$$g_B = \frac{13}{4} \neq \frac{5}{4} = g_A$$

# Asymmetric Example: More Complex Gates

**Assumption:**

- $\beta = k = 3$



**Equivalent INV**

- $W_p : W_n = 9 : 3$

- $C_{\text{gate,INV}} = 12$

**NOT 6 : 2**

$$g_{EDC} = \frac{21}{12} = \frac{7}{4}$$

$$g_{A'B'} = \frac{30}{12} = \frac{5}{2}$$

# What is the Reference Inverter?

---

- The assumption of logical effort is that the reference inverter has equal rise and fall delays
- Practical case when pull-up and pull-down resistances are different (rising and falling delays not equal)
  - Similar to the parasitic delay calculation earlier
    - $d_{UP} = g_{UP} \cdot h + p_{UP}$
    - $d_{DN} = g_{DN} \cdot h + p_{DN}$
- Since static CMOS gates are inverting, the transitions through subsequent gates must be alternating
  - Use an average logical effort (and parasitic effort)
    - $g_{AVG} = (g_{UP} + g_{DN})/2$

# Gates with Skewed P:N Ratio

---

- Assume:  $\beta = 1.7$ ,  $k = 3$

## Example: Inverter

- Pull-up: **reference inverter** is sized P:N of  $W_p:W_p/k$ 
  - $g_{UP} = W_p(1+1/1.7)/W_p(1+1/k) = 1.19$
- Pull-down: **reference inverter** is sized P:N of  $kW_n:W_n$ 
  - $g_{DN} = W_n(1+1.7)/W_n(1+k) = 0.675$
- $g_{AVG} = 0.93$  (instead of 1)
  - Confirms that  $\beta$  should be  $< k$  for speed  
Reference INV not optimized for speed

# NAND Gate with Skewed P:N Ratio

---

- Assume:  $\beta = 1.7$ ,  $k = 3$

**Example:** 2-input NAND Gate

- $g_{UP} = W_P(1+2/1.7)/W_P(1+1/k) = 1.63$
- $g_{DN} = W_N(2+1.7)/W_N(1+k) = 0.925$
- $g_{AVG} = 1.28$  (instead of  $5/4$ )

  
for  $\beta = 3$



# Know Your Reference INV

**R1** • Equal rise/fall time:  $\beta = 3, k = 3$        $g = 1$

• Minimum delay:  $\beta = 1.7, k = 3$        $g_{AVG} = 0.93$

## Example: 2-input NAND

•  $\beta = 1.7, k = 3$

▪  $g_{AVG} = 1.28$

ref **R1** ← different PU, PD

•  $\beta = 3, k = 3$

▪  $g = 5/4$

ref **R1**

$1.25 < 1.28$

but  $\beta = 1.7$  should  
be min delay?

# Know Your Reference INV

---

**R1** • Equal rise/fall time:  $\beta = 3$ ,  $k = 3$        $g = 1$

**R2** • Minimum delay:  $\beta = 1.7$ ,  $k = 3$        $g_{\text{AVG}} = 0.93$

**Example:** 2-input NAND

- $\beta = 1.7$ ,  $k = 3$


- $g = 1.37$       ref **R2**     $\times$      $0.93 = 1.28$       ref **R1**

**Translate b/w two reference INVs**

# Choosing a Reference Inverter

- The reference inverter can be any  $\beta$
- Choose the reference  $\times$  inverter  $\beta$   
Use a normalization to adjust the  $g$

## Example: 2-input NAND

- $g_{\text{REF}} = 1$  with  $\beta = 1.7$  ( $k = 3$ )
- $g_{\text{AVG\_INV}(\beta=k)} = 1.16$  for  $\beta = 3$   Captures speed penalty for  $\beta = 3$
- NAND2  $\beta = 1.7$  has  $g = 1.37$
- NAND2  $\beta = 3$  has  $g_{\text{AVG}} = 1.45 = 1.25 \cdot 1.16$

$g_{\text{NAND}}$  for  $\beta = 3$  w.r.t.  $\beta = 3$  INV 

# Velocity Saturation

---

- Need to account for the change in resistance
- Assume reference inverter is  $W_p=2W_n$

**Example:** 2-input NAND,  $W_p=2$ ,  $W_n=2$

- Assuming  $R_{n\_nostack}=(4/3)R_{n\_stack}$ ,  $R_{p\_nostack}=(6/5)R_{p\_stack}$
- The equivalent INV with same drive resistance
  - Pull-up: equiv inv = 2:1,  $g_{UP} = 4/3$  (same as before)
  - Pull-down: equiv inv = 8/3:4/3,  $g_{DN} = 4/4 = 1$
- Makes sense because v-sat allows the transition through the series stacking be faster (more current)

# **Multi-Stage Networks**

# Same Concept Applies at the Path Level

---

$$\text{Stage effort : } f_i = g_i \cdot h_i$$

$$\text{Path electrical effort : } H = C_{\text{out}} / C_{\text{in}}$$

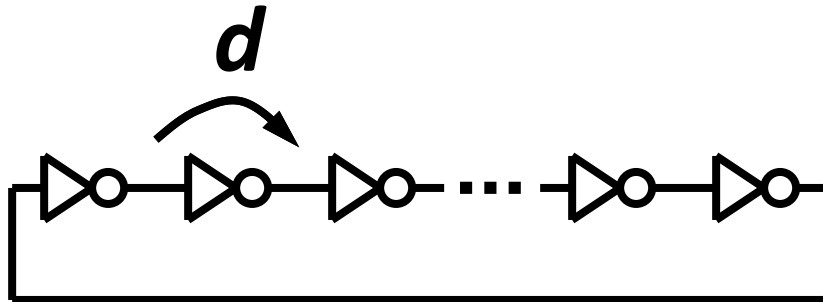
$$\text{Path logical effort : } G = g_1 g_2 \dots g_N$$

$$\text{Path effort : } F = G \cdot H$$

$$\text{Path delay : } D = \gamma_{INV} \sum_{i=1}^N p_i + \sum_{i=1}^N g_i h_i$$

## Example 4.2: Ring Oscillator

- Estimate the frequency of an  $N$ -stage ring oscillator:



$$\gamma = 1$$

Logical Effort :  $g = 1$

Electrical Effort :  $h = C_{\text{out}}/C_{\text{in}} = 1$

Parasitic Delay :  $p = 1$

Stage Delay :  $d = g \cdot h + p \cdot \gamma_{\text{INV}} = 2$

$$\text{OSC frequency : } f_{\text{osc}} = \frac{1}{2Nd\tau} = \frac{1}{4N\tau}$$

# Total Effort for a Path

---

Logical effort

$$G = \prod_{i=1}^N g_i$$

Fanout

$$H = \frac{C_{out}}{C_{in}}$$

Path Effort

$$F = \prod_{i=1}^N f_i$$

**Treat the path as a single “gate”**



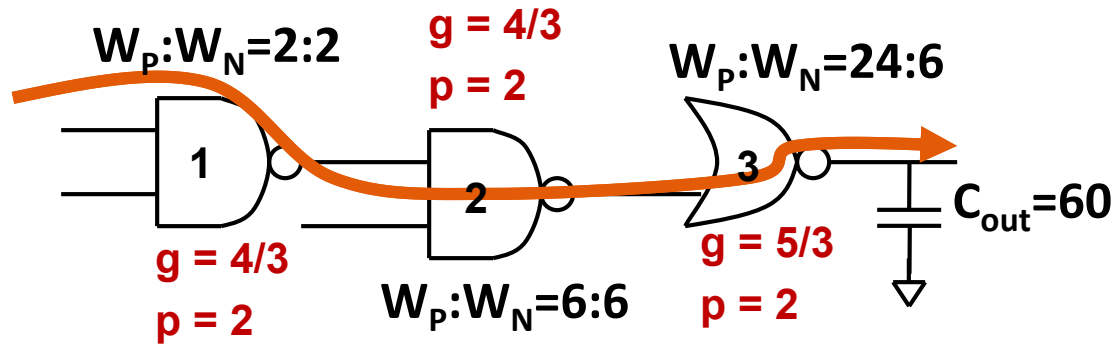
## Example 4.3a: Total Effort (G, H, F)?

### Calculations:

- $G = (4/3)^2(5/3) = 3$
- $H = 60/4 = 15$
- $F = 45$ ;  **$F \neq GH$**   
(yes for this case)

### Assumption:

- $\beta = 2$  (REF)



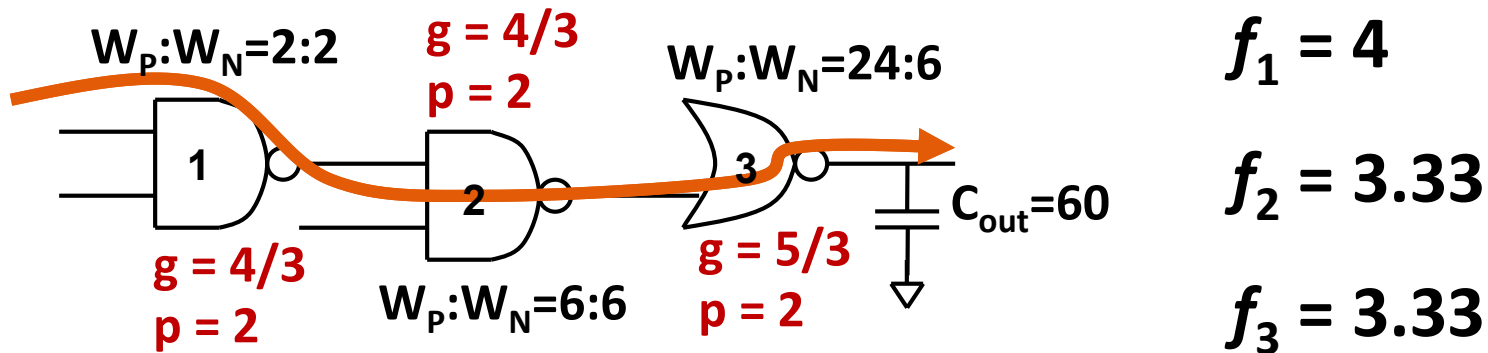
$$f_1 = 4$$

$$f_2 = 3.33$$

$$f_3 = 3.33$$

## Example 4.3a: Total Path Delay?

Assume:  $\gamma = 1$



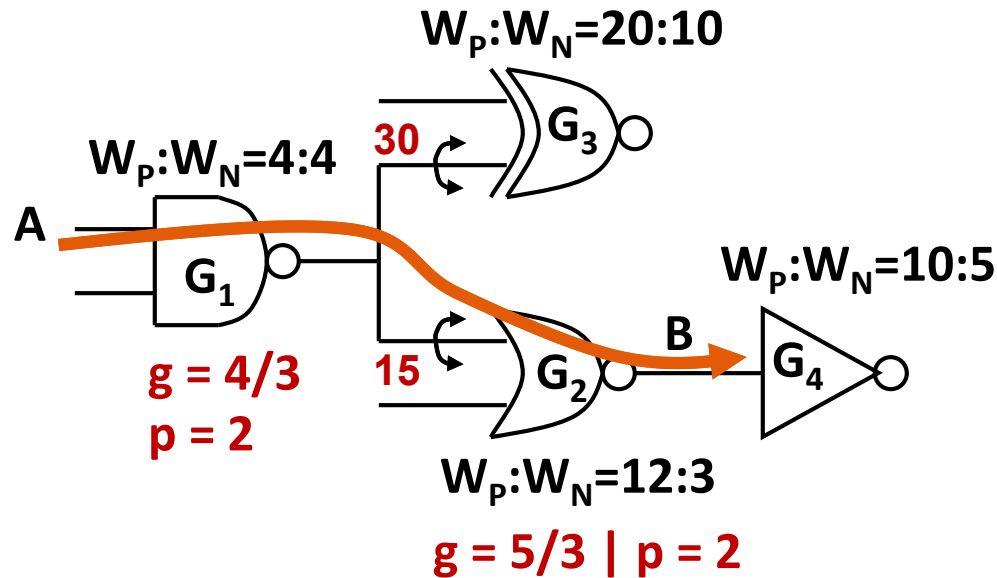
Delay of a multi-stage network = sum of stage delays

$$D_F = 10.66, P = 6$$

$$D = 16.66 (\tau \text{ delays})$$

# Branching Effort

# Example 4.4: Delay from A to B?



Assume:

- $\beta = k = 2$
- $\gamma = 1$

$$h_1 = \frac{45}{8} = 5.625$$

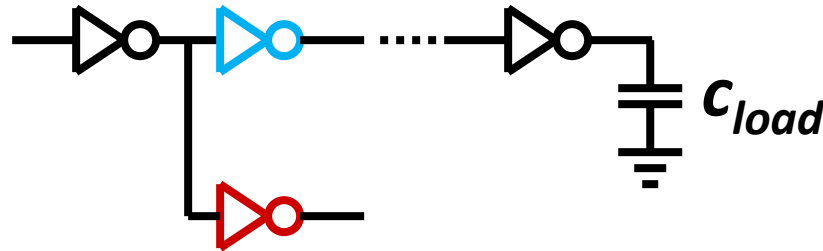
$$h_2 = \frac{15}{15} = 1$$

- $d_1 = g_1 h_1 + p_1 = 9.5$
- $d_2 = g_2 h_2 + p_2 = 3.66$

Delay = 13.16

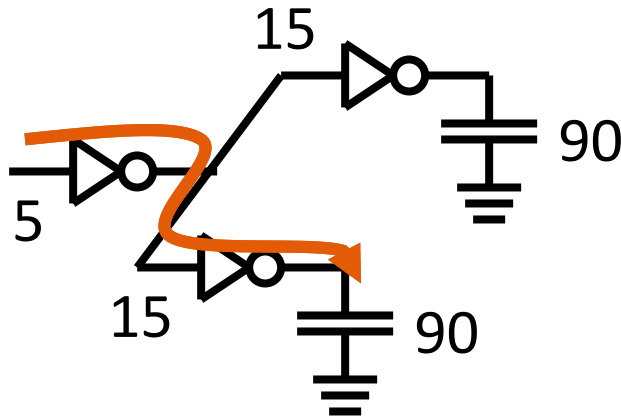
# Add Branching Effort

- Ratio of total to on-path capacitance
  - How much more current is needed to supply on-path (given that some current “flows” off-path)



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

## Example 4.5: Path Effort $F = ?$



$$G = 1$$

$$H = 90/5 = 18$$

$$F = 18 \text{ (wrong!)}$$

$$f_1 = (15+15)/5 = 6$$

$$f_2 = 90/15 = 6$$

$$F = 36, \text{ not } 18!$$

- Introduce new kind of effort to account for branching:

**Branching Effort (BE)**

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



**Path BE**

$$B = \prod_{i=1}^N b_i$$

# Multistage Networks with Branching

---

$$\text{Stage effort : } f_i = g_i \cdot h_i$$

$$\text{Path electrical effort : } H = C_{\text{out}}/C_{\text{in}}$$

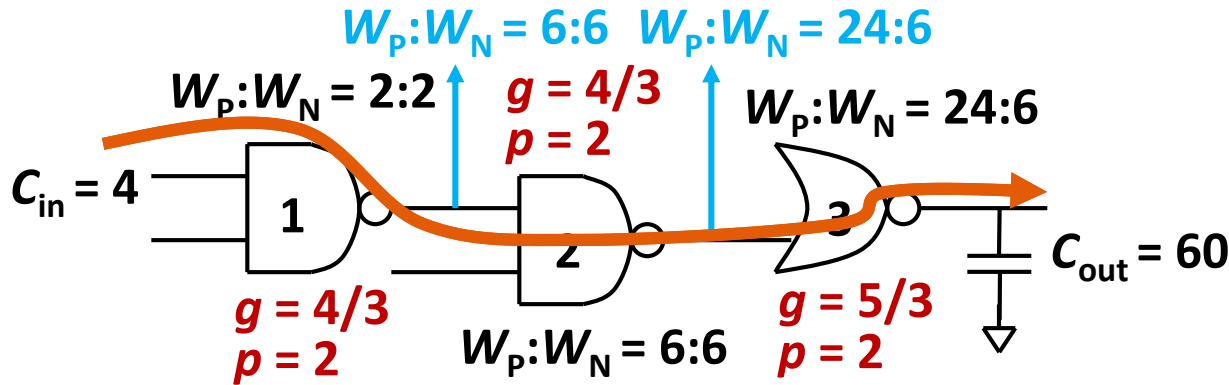
$$\text{Path logical effort : } G = g_1 g_2 \dots g_N$$

$$\text{Path branching effort : } B = b_1 b_2 \dots b_N$$

$$\text{Path effort : } F = G \cdot H \cdot B$$

$$\text{Path delay : } D = \gamma_{INV} \sum_{i=1}^N p_i + \sum_{i=1}^N g_i h_i$$

# Example 4.6: Path Effort F = ?



Branching

No branch.

$$f_1 = 8$$

$$f_1 = 4$$

$$f_2 = 6.66$$

$$f_2 = 3.33$$

$$f_3 = 3.33$$

$$f_3 = 3.33$$

Calculate F:

- $h_1 = (2 \cdot 12)/4 = 6$
- $h_2 = (2 \cdot 30)/12 = 5$
- $h_3 = 2$
- $F = (4/3 \cdot 6) \cdot (4/3 \cdot 5) \cdot (5/3 \cdot 2) = 177.8$

For circuits with branching:

- $G$  is the same = 3
- $H$  is the same = 15
- $F$  differs

$$F = G \cdot B \cdot H$$

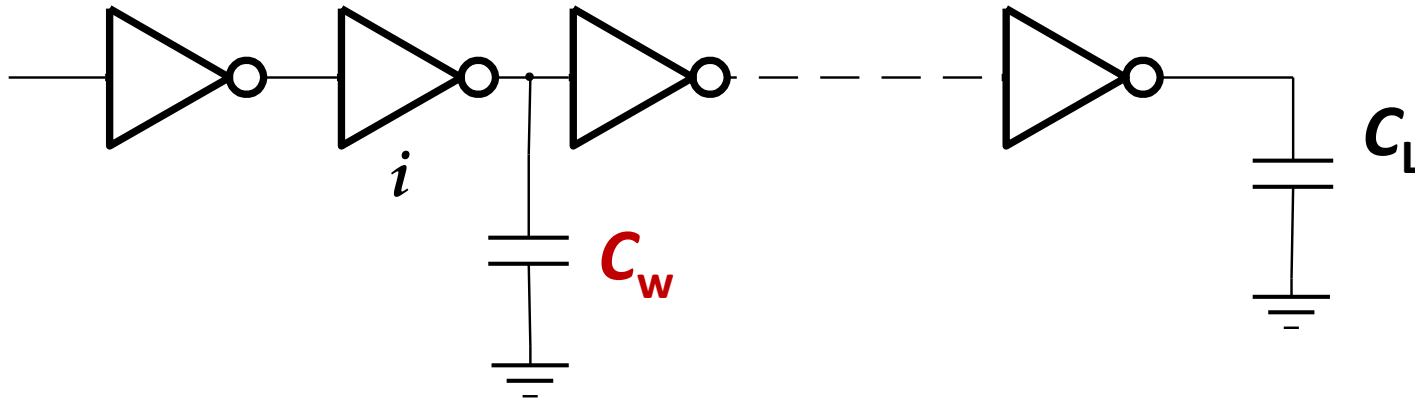


# What About Interconnects?

---

- C from wires are difficult to deal with
- **Fixed load** so intuitively **increase fanout**
- **Short wires** – small parasitic capacitance
  - Treat them as increasing  $p$  for each gate
    - Not exact but accounts for the effect
- **Long wires** – large load capacitance
  - Size of driving gate is as if driving a large C
- **Medium wires** – most difficult ( $C \approx$  gate load)
  - Delay as function of gate and wire cap
    - $N^{\text{th}}$ -order polynomial and differentiate
  - More realistic method is to iterate

# Handling Wires & Fixed Loads



Fixed cap

$$D = \gamma_{INV} \sum_{i=1}^N p_i + \sum_{i=1}^N g_i \left( h_i + \frac{C_{w,i+1}}{C_i} \right)$$

# Logical Effort Delay Calculation: Summary

---

- Delay normalized by inverter delay,  $d = g \cdot h + p \cdot \gamma_{\text{INV}}$
- $g$  and  $p$  are characteristics of a logic gate that depends on its structure and does not depend on gate size
  - $g$ 's and  $p$ 's depend on input and PU / PD
  - **Simplify:** use  $g_{\text{AVG}}$ , ignore  $C$ 's of intermediate nodes
- Once a table of  $g$ 's and  $p$ 's are created for the catalog of gates, delay can be calculated quickly and easily
- Next we will look at how to **size** a network (instead of just analyzing it)

# **Gate Sizing**

## **Using Logical Effort**

# Optimum Effort per Stage

---

When each stage bears the same effort:

$$f^N = G \cdot B \cdot H = F$$

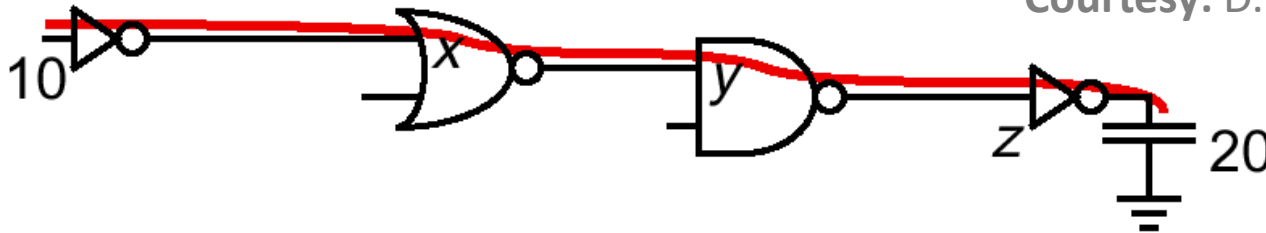
$$f = (F)^{\frac{1}{N}}$$

| Stage FO              | <b>Complex gates should<br/>drive smaller load</b> |
|-----------------------|--|
| $h_i = \frac{f}{g_i}$ |  |

Minimum path delay :  $D_{min} = N \cdot f + P$

## Example 4.7: (x, y, z) for Min Delay?

Courtesy: D. Harris (HMC)



$$g_1 = 1$$

$$h_1 = x/10$$

$$g_2 = 5/3$$

$$h_2 = y/x$$

$$g_3 = 4/3$$

$$h_3 = z/y$$

$$g_4 = 1$$

$$h_4 = 20/z$$

- First, compute path effort:

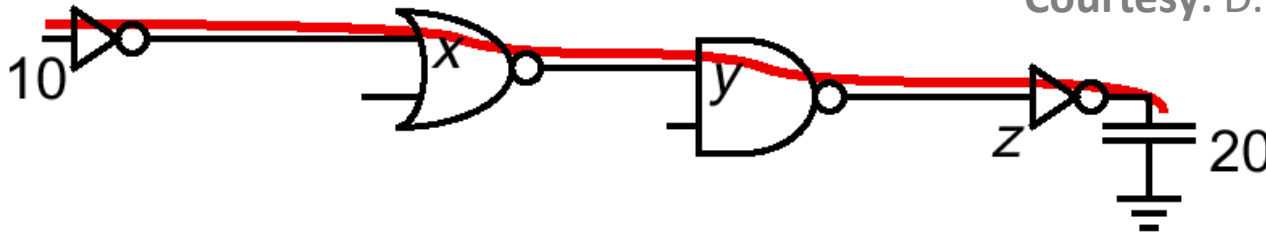
$$F = G \times H = \left( 1 \cdot \frac{5}{3} \cdot \frac{4}{3} \cdot 1 \right) \times \left( \frac{20}{10} \right) = \frac{40}{9}$$

- The optimal stage effort is:

$$f = g \cdot h = \left( \frac{40}{9} \right)^{\frac{1}{4}} = 1.45$$

## Example 4.7: (x, y, z) for Min Delay?

Courtesy: D. Harris (HMC)



$$g_1 = 1$$
$$h_1 = x/10$$

$$g_2 = 5/3$$
$$h_2 = y/x$$

$$g_3 = 4/3$$
$$h_3 = z/y$$

$$g_4 = 1$$
$$h_4 = 20/z$$

- We can now size the gates: →

$$z = 1 \cdot \frac{20}{1.45} = 13.8$$

$$x = \frac{5}{3} \cdot \frac{y}{1.45} = 14.5$$

$$C_{in} = g \cdot \frac{C_{out}}{f}$$

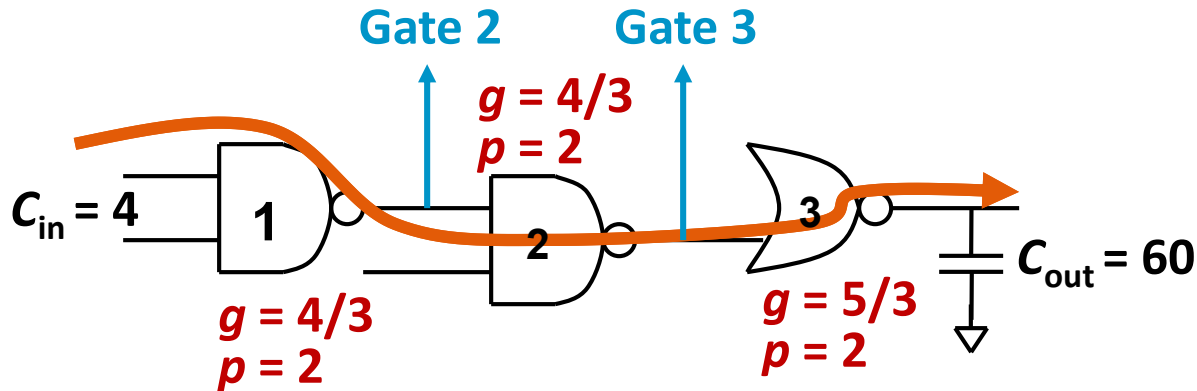
$$y = \frac{4}{3} \cdot \frac{z}{1.45} = 12.7$$

$$C_{in} = 1 \cdot \frac{x}{1.45} = 10 \text{ (sanity check)}$$

- Finally, calculate  $D_{min}$  ( $\gamma = 1$ ):

$$D = 4 \cdot f + P = 4 \cdot 1.45 + 6 = 11.8$$

# Example 4.8: Min Delay? (Branching)



- $\beta = 2$
- $C_{in2}, C_{in3} = ?$
- $W_{p3}, W_{p2} = ?$

$$B = b_1 b_2 = 4$$

$$F = 177.8, f_{opt} = 5.62$$

$$C_{in3} = 5/3 * (1 * 60) / 5.6 = 17.9 \quad \rightarrow * 4/5 = 14.3 = W_{p3}$$

$$C_{in2} = 4/3 * (b_2 * 17.9) / 5.6 = 8.5 \quad \rightarrow * 1/2 = 4.25 = W_{p2}$$

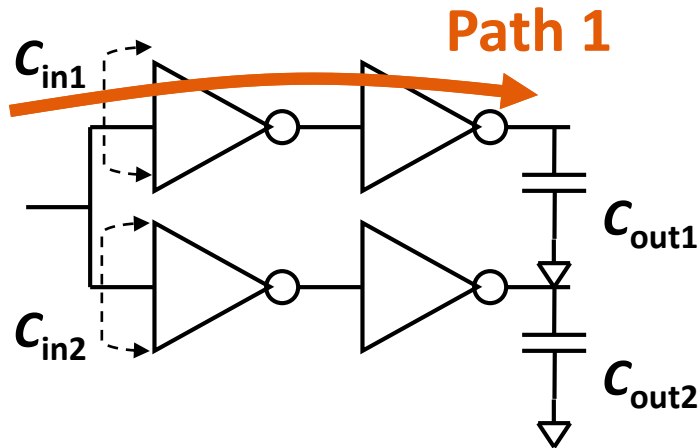
$$C_{in1} = 4/3 * (b_1 * 8.5) / 5.6 = 4$$

$$C_i = g_i \cdot \frac{b_i C_{i+1}}{f}$$



# Branches: Same # Stages, Different Loads

- Optimal system has all paths with equal delay
- Branching for path 1,  $b_1 = 1 + x$ 
  - Assumption is that  $P_1 \approx P_2$



$$C_{in2}/C_{in1} = ?$$

$$D_1 = D_2$$

$$N \cdot f_1 + P_1 = N \cdot f_2 + P_2$$

$$F_1 = \frac{G_1 C_{out1}}{C_{in1}} = F_2 = \frac{G_2 C_{out2}}{C_{in2}}$$

$$\Rightarrow \frac{C_{in2}}{C_{in1}} = x = \frac{C_{out2}}{C_{out1}}$$

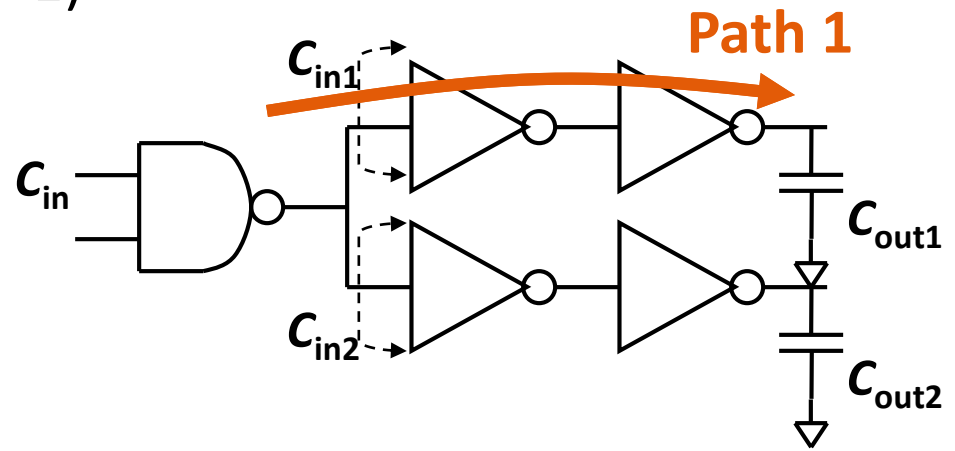
# Path Effort Estimate

- With equal-stage branches,  $H$  can be estimated without knowing each stage's  $b$

- $H = (C_{out1} + C_{out2}) / C_{in}$  (with  $B = 1$ )

- Example for Path 1:

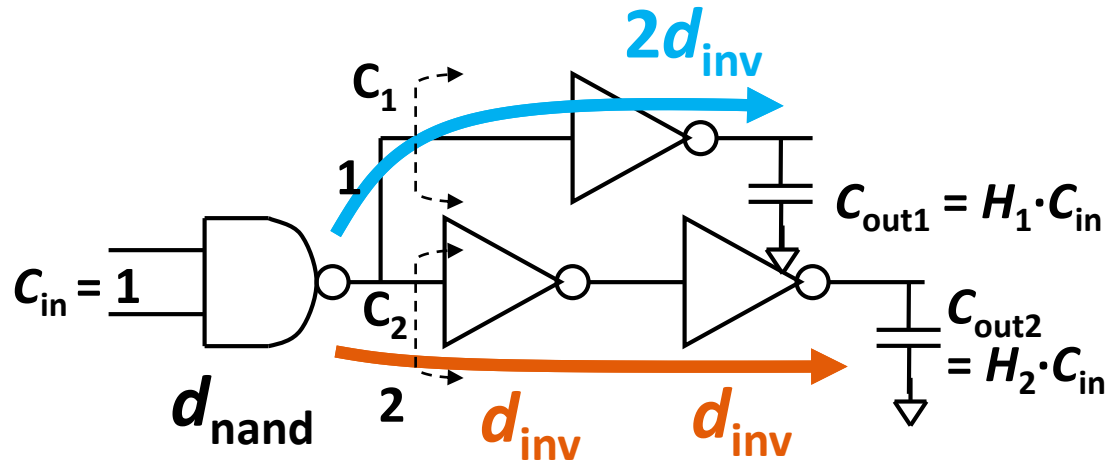
- $G = g_{\text{NAND}}$
  - $H_1 = C_{out1} / C_{in}$
  - $B_1 = (1 + C_{out2} / C_{out1})$
  - $F = G \cdot H_1 \cdot B = g_{\text{NAND}} (C_{out1} + C_{out2}) / C_{in}$
  - Same as  $F = G \cdot H$



- Error if different  $G$  and  $P$  in the two paths

# Unequal-Length Branches (Not Easy)

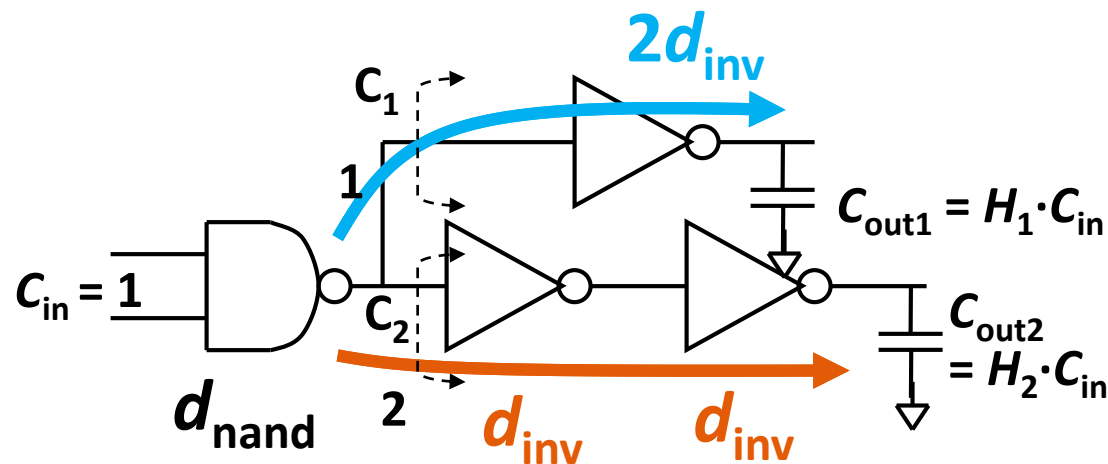
- $f_{\text{opt}}$  differs between two paths



Solving precisely, **example:**

- $2d_{\text{inv}} - p = C_{\text{out1}}/C_1$
- $(d_{\text{inv}} - p)^2 = C_{\text{out2}}/C_2$
- $d_{\text{nand}} = g_{\text{nand}}(C_1 + C_2)/C_{\text{in}}$
- $C_1 = C_{\text{out1}}/(2d_{\text{inv}} - p)$
- $C_2 = C_{\text{out2}}/(d_{\text{inv}} - p)^2$

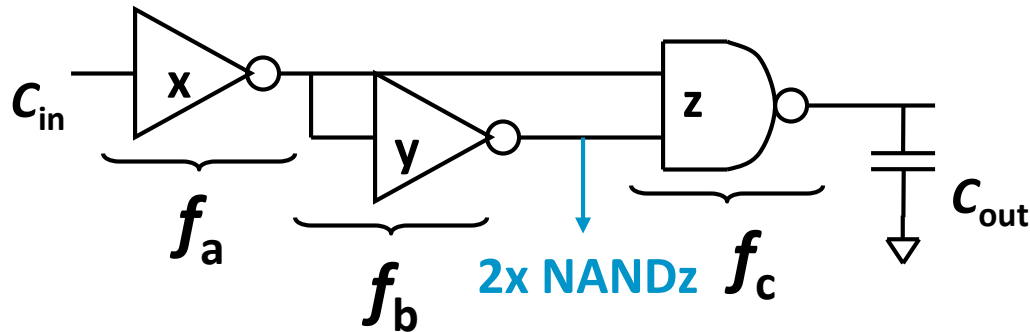
# Unequal-Length Branches (Not Easy)



- $D = g_{nand}(C_{out1}/(2d_{inv} - p) + C_{out2}/(d_{inv} - p)^2) + 2d_{inv}$
- Take partial derivative of  $\partial D/\partial(d_{inv})$
- Not easy so ignore  $p$  to simplify
  - If  $g_{NAND} = 4/3$ ,  $H_1 = H_2 = 3$  :  $d_{nand} = 2.35$ ,  $d_{inv} = 1.80$
  - The per stage  $f = g \cdot h$  (no  $p$ ) is different per stage
- Most branches are relatively long or not critical path

# Re-convergent Paths

- Recombined branches add complexity to logical effort
  - Typically constrains the sizing problem



**Example:** ignore parasitics. Let  $x = C_{in} = 1$

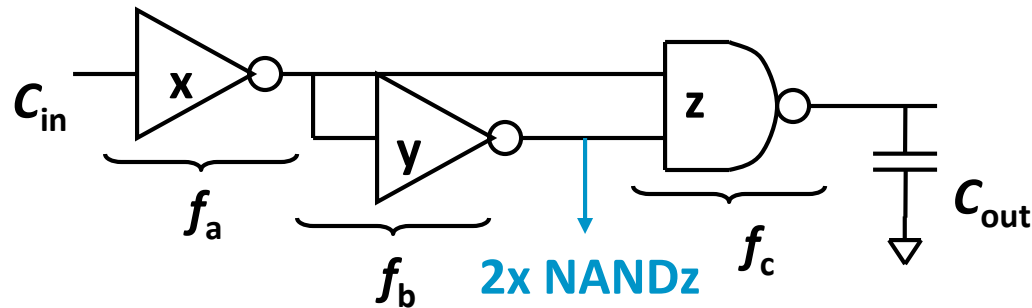
- $f_a = y + z \quad (x = 1)$

- $f_b = 3z/y$

- $f_c = g_{\text{NAND}} C_{out}/z$

**2 variables**

# Re-convergent Paths



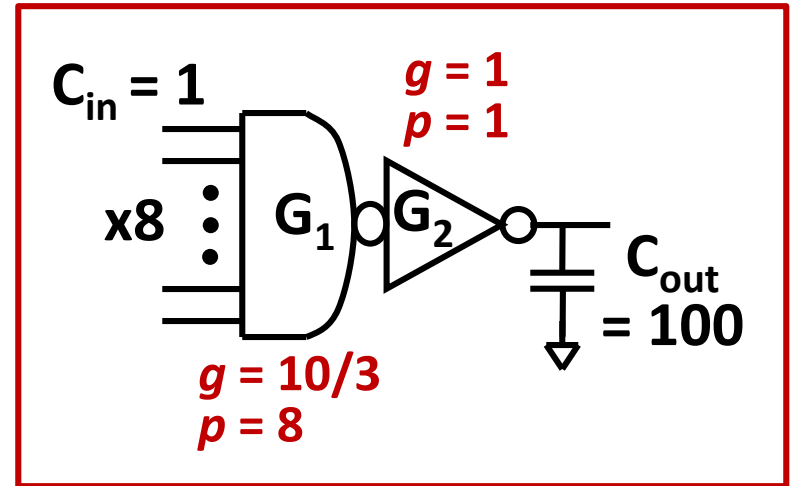
$$y+z = 3z/y = g_{NAND}C_{out}/z$$

- Constrain with  $f_a = f_b = f_c$
- **2 variables, 2 equations (directly solve)**
  - Any constraints are possible
- Increase variables by introducing more buffering (parallel to y)

# Example 4.9: Logic Optimization Example

- Symmetric 8-input AND ( $\beta = k = 2$ ,  $3W_0$  is unit  $C$ )
- Logical effort

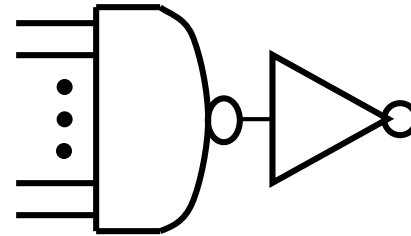
- $G = 10/3$ ,  $B = 1$ ,  $H = 100$
- $F = 333.3$
- For 2 stages,  $f_{\text{opt}} = 18.3$ 
  - $h_2 = 18.3$ ,  $C_{\text{in}2} = 5.5$ ,  $W_p = 11W_0$
  - $h_1 = 5.5$ ,  $C_{\text{in}1} = 1$ ,  $W_p = 0.6W_0$
  - Delay =  $36.6 + 9 = 45.6$



- Buffering
  - For 3 stages,  $f_{\text{opt}} = 7$  (1 extra inverter), Delay = 31
  - For 4 stages,  $f_{\text{opt}} = 4.3$  (2 extra inverters), **Delay = 28.2**
  - For 5 stages,  $f_{\text{opt}} = 3.19$  (closer to 3.6), Delay=28
  - For 6 stages,  $f_{\text{opt}} = 2.63$  (below 3.6), Delay=28.8

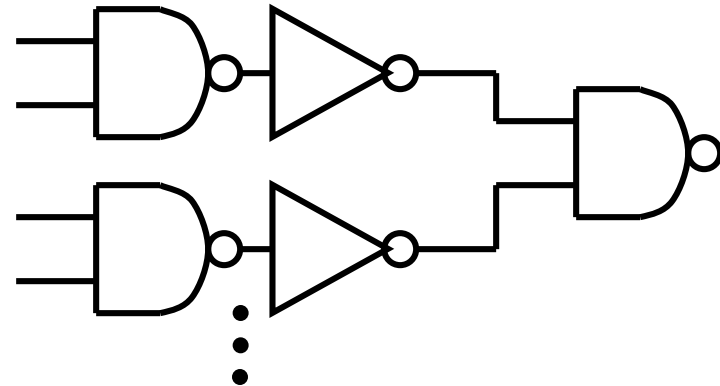
## Example 4.9: Other Implementations

- Many ways to implement this same function



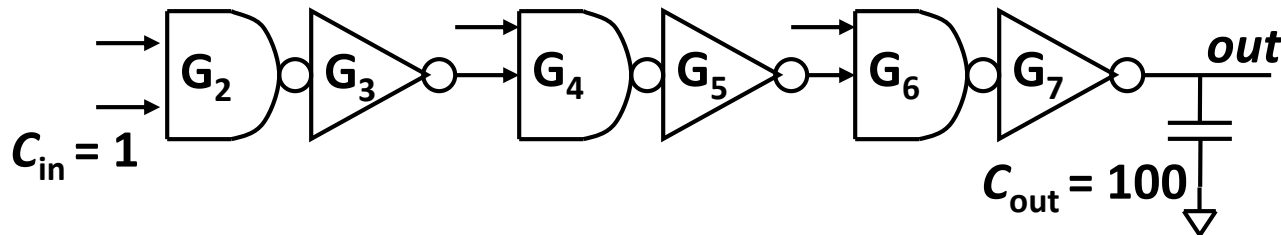
- Use a tree of fewer input AND gates

- $(A_0A_1)(A_2A_3)\dots$
- If multiple ANDs (as in a mem decoder), then partial results can be shared



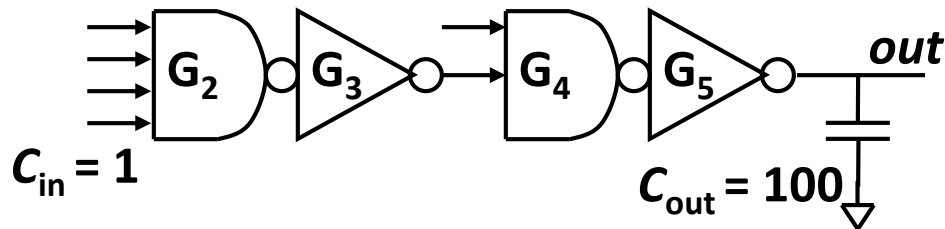


# Example 4.9: 2-input Implementation



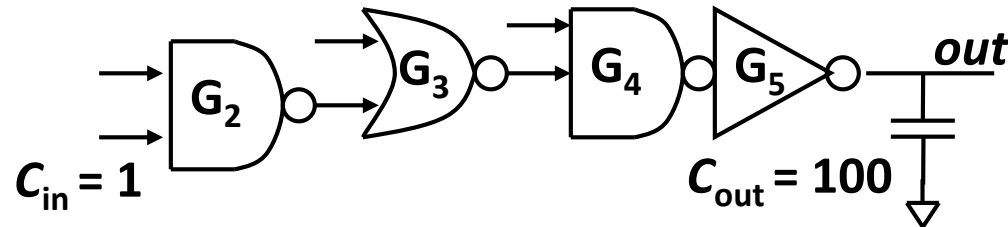
- $F = G \cdot B \cdot H = 4/3^3 \cdot 100 = 235$
- $f_{opt} = 2.48$  (too small)
- $D = 6 \cdot 2.48 + 3 \cdot 2 + 3 \cdot 1 = 14.88 + 9 = 23.88$ 
  - Still better than 8-input NAND
- Optimal sizing
  - $G_7$ :  $C_{in7} = 100g_{inv}/f = 40$ ,  $W_p = 80W_0$
  - $G_6$ :  $C_{in6} = C_{in7}g_{nand}/f = 21.5$ ,  $W_p = 32W_0$
  - $G_5$ :  $C_{in5} = C_{in6}g_{inv}/f = 8.7$ ,  $W_p = 17.4W_0$
  - $G_4$ :  $C_{in4} = C_{in5}g_{nand}/f = 4.7$ ,  $W_p = 7W_0$
  - $G_3$ :  $C_{in3} = C_{in4}g_{inv}/f = 1.88$ ,  $W_p = 3.8W_0$
  - Double check  $G_2$ :  $C_{in} = g_{nand}C_{in3}/f = 1$ ,  $W_p = 1.5W_0$

# Example 4.9: 4-input Implementation



- $F = B \cdot G \cdot H = 4/3 \cdot 6/3 \cdot 100 = 266$
- $f_{\text{opt}} = 4.04$  (a tad high)
- $D = 4 \cdot 4.04 + 1 \cdot 4 + 1 \cdot 2 + 2 \cdot 1 = 16 \therefore 16 + 8 = 24.16$ 
  - Slightly worse than 2-input! Due to self-loading
- Optimal sizing
  - $G_5$ :  $C_{\text{in}5} = 100 g_{\text{inv}} / f = 24.75$ ,  $W_p = 50W_0$
  - $G_4$ :  $C_{\text{in}4} = C_{\text{in}5} g_{\text{nand}} / f = 8.16$ ,  $W_p = 12.5W_0$
  - $G_3$ :  $C_{\text{in}3} = C_{\text{in}4} g_{\text{inv}} / f = 2.02$ ,  $W_p = 4W_0$
  - Double check  $G_2$ :  $C_{\text{in}} = g_{\text{nand}4} C_{\text{in}3} / f = 1$ ,  $W_p = 1.0W_0$

# Example 4.9: 2-NOR Implementation



- $F = BGH = 4/3^2 * 5/3 * 100 = 296$
- $f_{opt} = 4.14$  (a tad high)
- $D = 4 \cdot 4.14 + 3 \cdot 2 + 1 \cdot 1 = 16.56 + 7 = 23.6$  (Best one!)
- Optimal sizing
  - $G_5$ :  $C_{in5} = 100g_{inv}/f = 24.1$ ,  $W_p = 48W_0$
  - $G_4$ :  $C_{in4} = C_{in5}g_{nand}/f = 7.76$ ,  $W_p = 11.6W_0$
  - $G_3$ :  $C_{in3} = C_{in4}g_{nor}/f = 3.125$ ,  $W_p = 7.5W_0$
  - Double check  $G_2$ :  $C_{in} = g_{nand}C_{in3}/f = 1$ ,  $W_p = 1.5W_0$
- Lesson: use close to optimal  $f$ , use gates with small  $p$

# Logical Effort “Design Flow” (Min Delay)

---

- Compute the path effort:  $F = G \cdot B \cdot H$
- Find the best number of stages:  $N_{opt} \sim \log_4(F)$
- Compute the stage effort:  $f = (F)^{1/N}$
- Working from either end, determine gate sizes:

$$C_i = g_i \cdot \frac{b_i C_{i+1}}{f}$$

Reference: Sutherland, Sproul, Harris, “Logical Effort,” (Morgan-Kaufmann 1999)

# Summary of LE Design Methodology

---

- 1. Buffer non-critical paths with minimum-sized gates**
  - Minimize loading on critical path
  - Simplifies sizing of non-critical path
- 2. Estimate total effort along each path (without branching)**
- 3. Verify that the number of stages is appropriate**
  - Add inverters if  $f_{\text{opt}} > 5$
- 4. Assign branch ratio of each branch**
  - Estimate based on the ratio of the effort of the paths
  - Ignore paths that have little effect (i.e. min-sized)
  - Include wire capacitances
- 5. Compute delays for the design (include parasitic delay)**
  - Adjust branching ratios (especially with  $C_{\text{wire}}$ )
  - Repeat if necessary until delay meets specification
- 6. Re-optimize logic network if  $f_{\text{opt}}$  is small (Return to step 3)**