CHAPTER 4

Circuit Characterization and Performance Estimation I

Outline

1. **Delay Estimation**
2. Logical Effort and Transistor Sizing
3. Power Dissipation
4. Interconnect
5. Wire Engineering
6. Design Margin
7. Reliability
8. Scaling

Transient Response

- DC analysis tells us V_{out} if V_{in} is constant
- Transient analysis tells us $V_{out}(t)$ if $V_{in}(t)$ changes
  - Requires solving differential equations
- Input is usually considered to be a step or ramp
  - From 0 to V_{DD} or vice versa

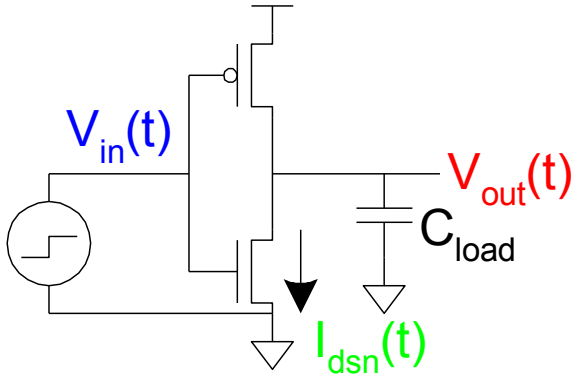
Inverter Step Response

- Find step response of inverter driving load cap

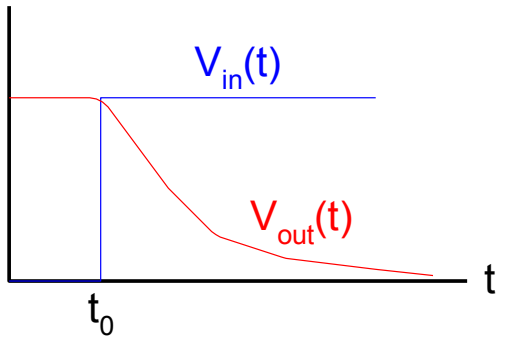
$$V_{in}(t) = u(t - t_0)V_{DD}$$

$$V_{out}(t < t_0) = V_{DD}$$

$$\frac{dV_{out}(t)}{dt} = -\frac{I_{dsn}(t)}{C_{load}}$$



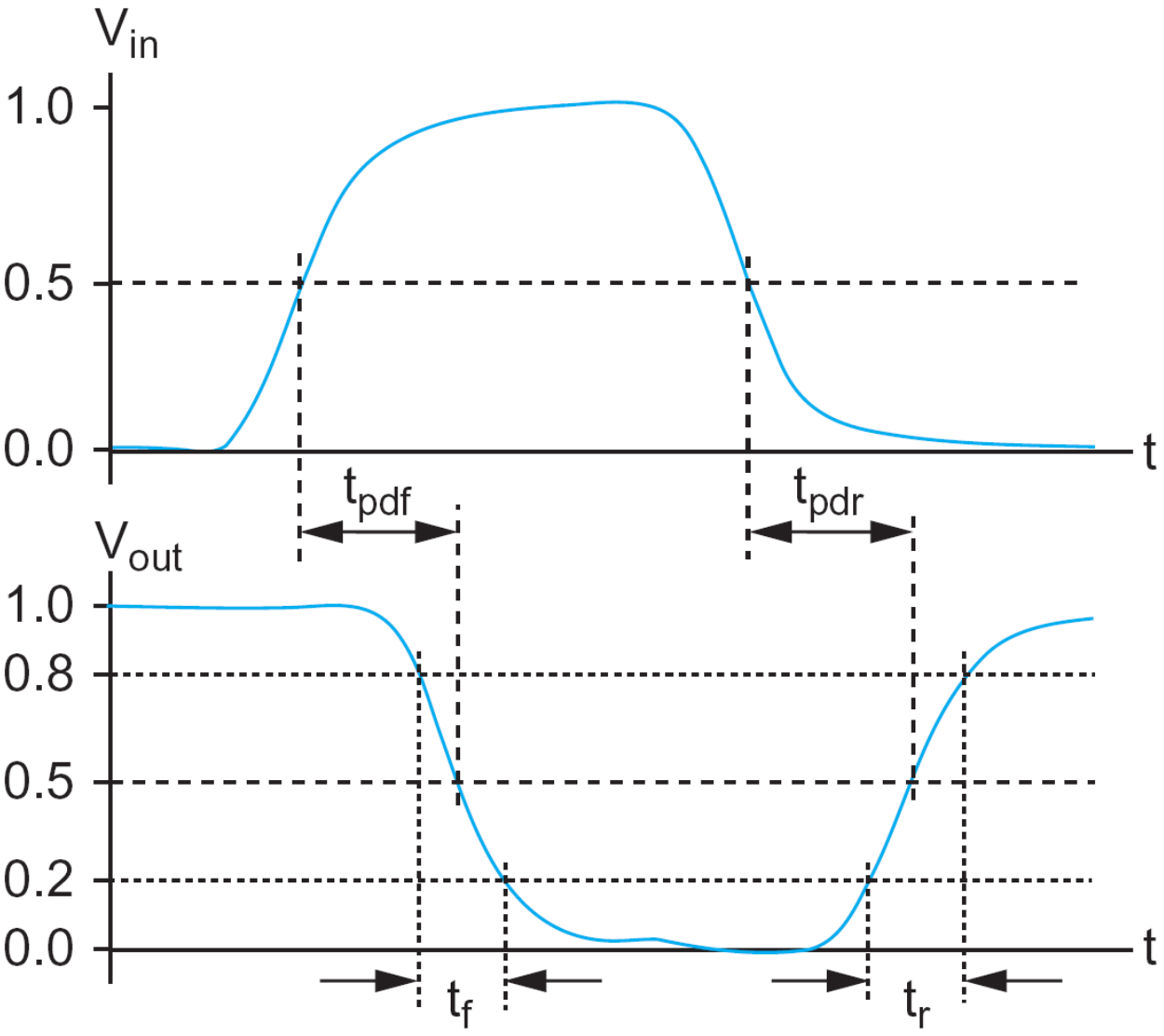
$$I_{dsn}(t) = \begin{cases} 0 & t \le t_0 \\ \frac{\beta}{2}(V_{DD} - V_t)^2 & V_{out} > V_{DD} - V_t \\ \beta\left(V_{DD} - V_t - V_{out}(t)/2\right)V_{out}(t) & V_{out} < V_{DD} - V_t \end{cases}$$



Delay Definitions (1/3)

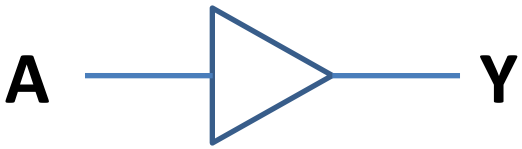
- $\mathbf{t_{pdr}}$: *maximum rising propagation delay*
  - From input to rising output crossing $V_{DD}/2$
- $\mathbf{t_{pdf}}$: *maximum falling propagation delay*
  - From input to falling output crossing $V_{DD}/2$
- $\mathbf{t_{pd}}$: *average propagation delay*
  - $t_{pd} = (t_{pdr} + t_{pdf})/2$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

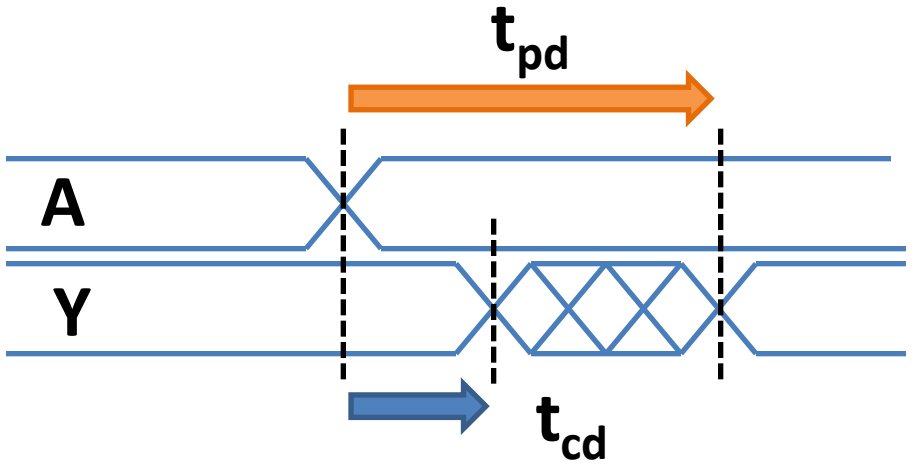
Delay Definitions (2/3)



Delay Definitions (3/3)

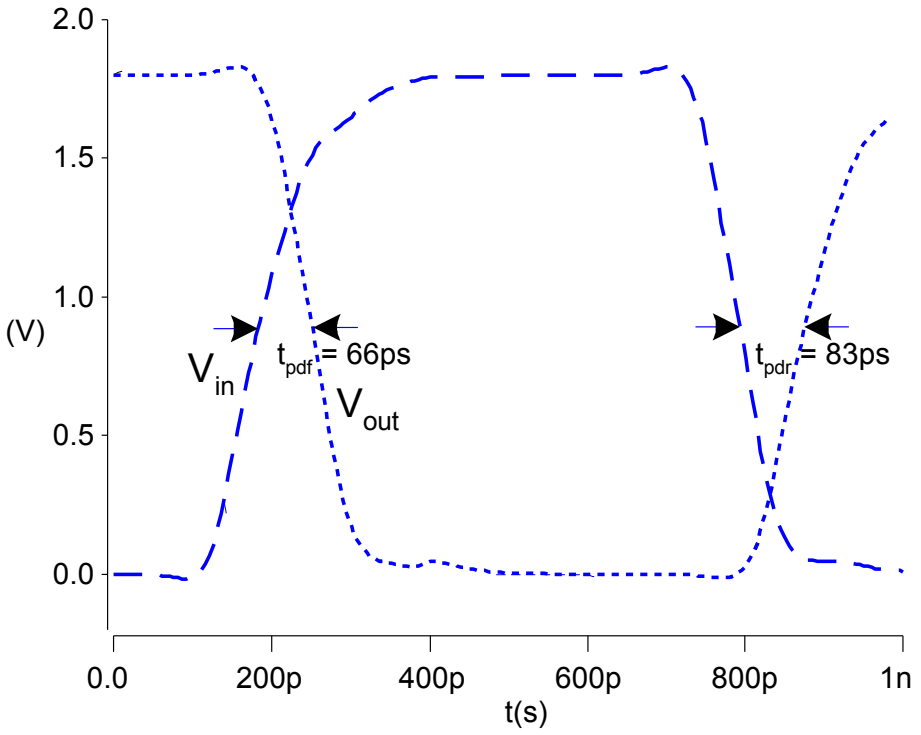
- $\mathbf{t_{cdr}}$: *minimum rising contamination delay*
  - From input to rising output crossing $V_{DD}/2$
- $\mathbf{t_{cdf}}$: *minimum falling contamination delay*
  - From input to falling output crossing $V_{DD}/2$
- $\mathbf{t_{cd}}$: *average contamination delay*
  - $t_{cd} = (t_{cdr} + t_{cdf})/2$





Simulated Inverter Delay

- Solving differential equations by hand is too difficult
- SPICE simulator solves the equations numerically
  - Use more accurate I-V models too
- But accurate simulations take time



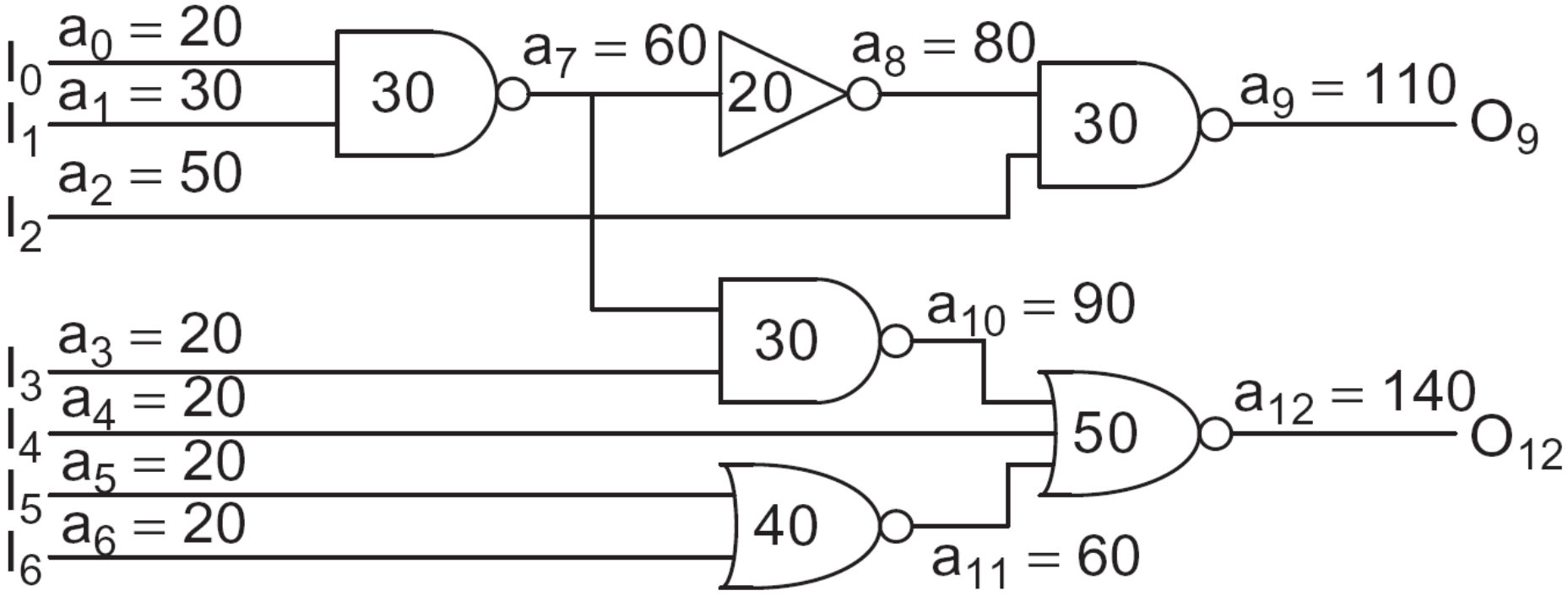
Delay Estimation I

- Estimate delay easily
  - Not as accurate as simulation
  - Easier to ask “What if?”
- The step response usually looks like a 1st 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 $\mathbf{t_{pd}}$ = RC
- Characterize transistors by finding their effective R
  - Depends on average current as gate switches

Delay Estimation II

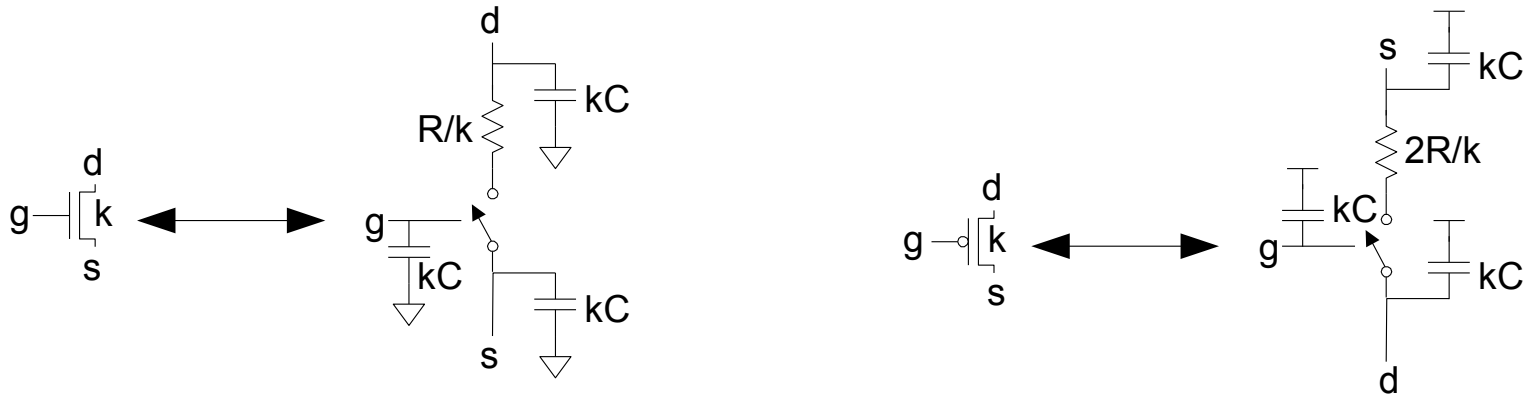
- **Critical path**：the signal path with the slowest (most critical) timing, it can be affected at 4 main levels.
- The **architectural/micro-architectural** level
  - Tradeoff of pipeline stages, number of execution units, and size of memory, it's the level with the most impact factor.
- The **logic** level
  - Tradeoff of functional block types, number of gate in the cycle, fan-in and fan-out number.
- The **circuit** level
  - Choosing transistor size and CMOS logic styles.
- The **layout** level
  - Determine floor plan, wire length, and check parasitic

Critical Path

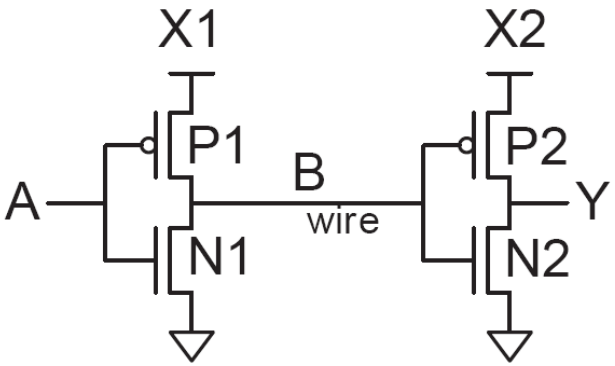


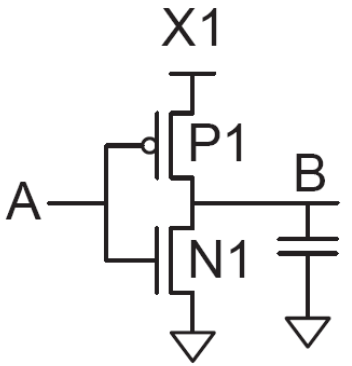
RC Delay Models

- 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

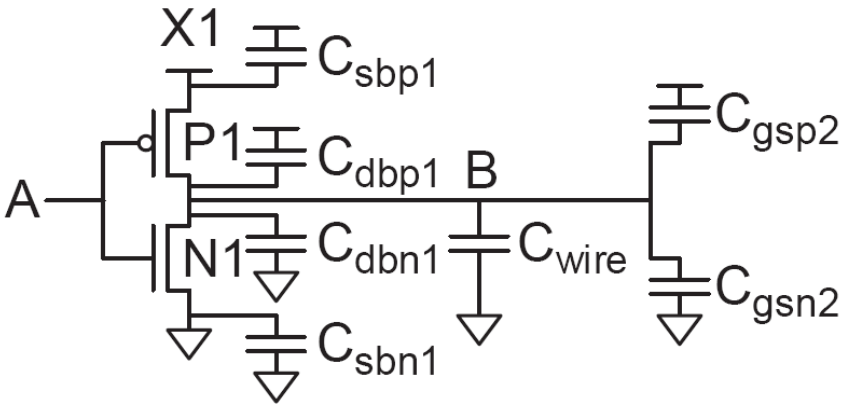


Example: Inverter

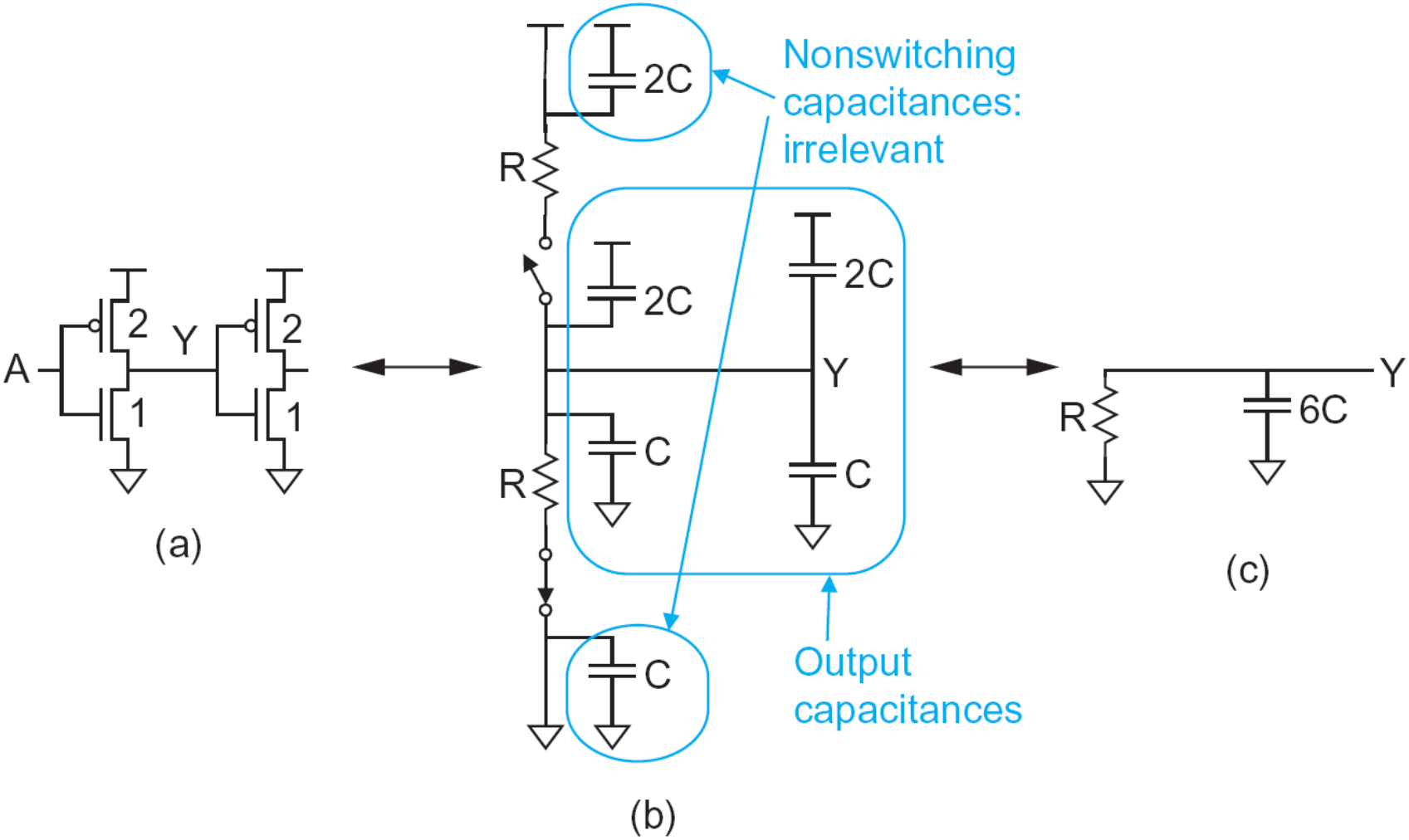




$C_{out} = C_{dbn1} + C_{dbp1} + C_{wire} + C_{gsn2} + C_{gsp2}$

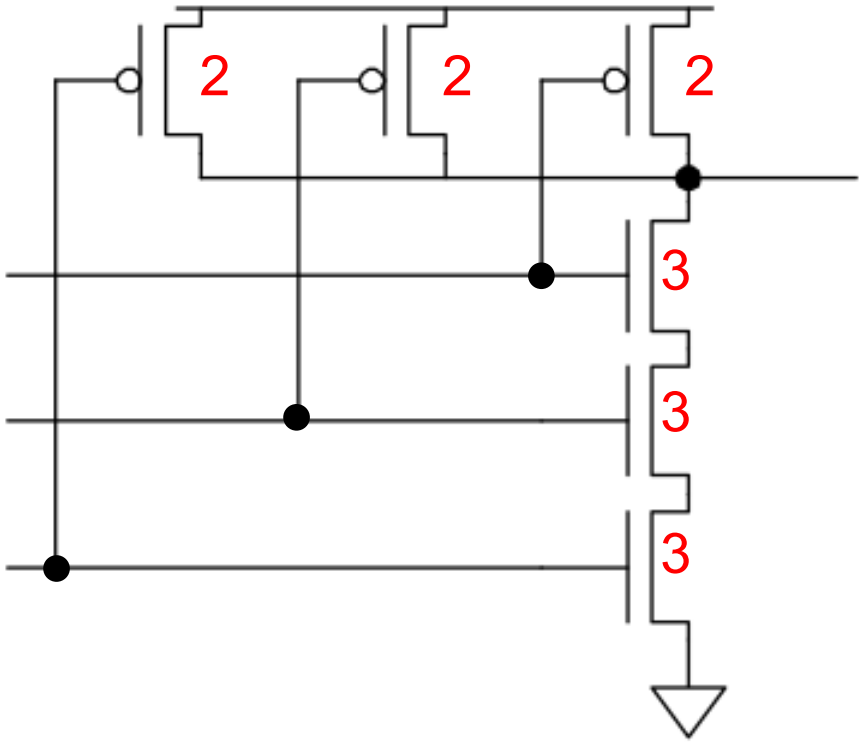


Example: Inverter



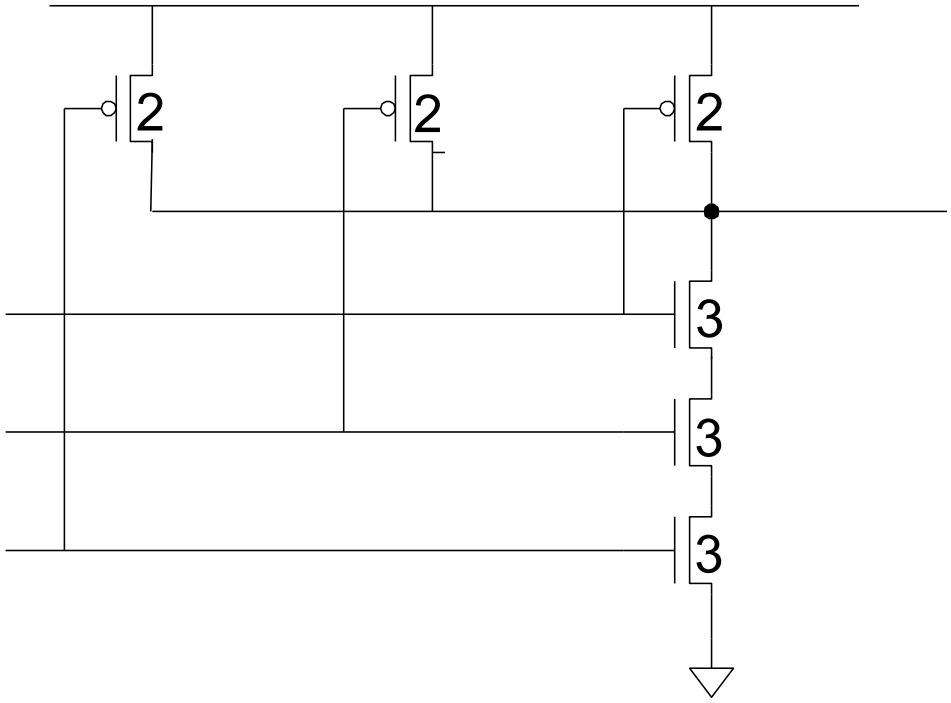
Example: 3-input NAND

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



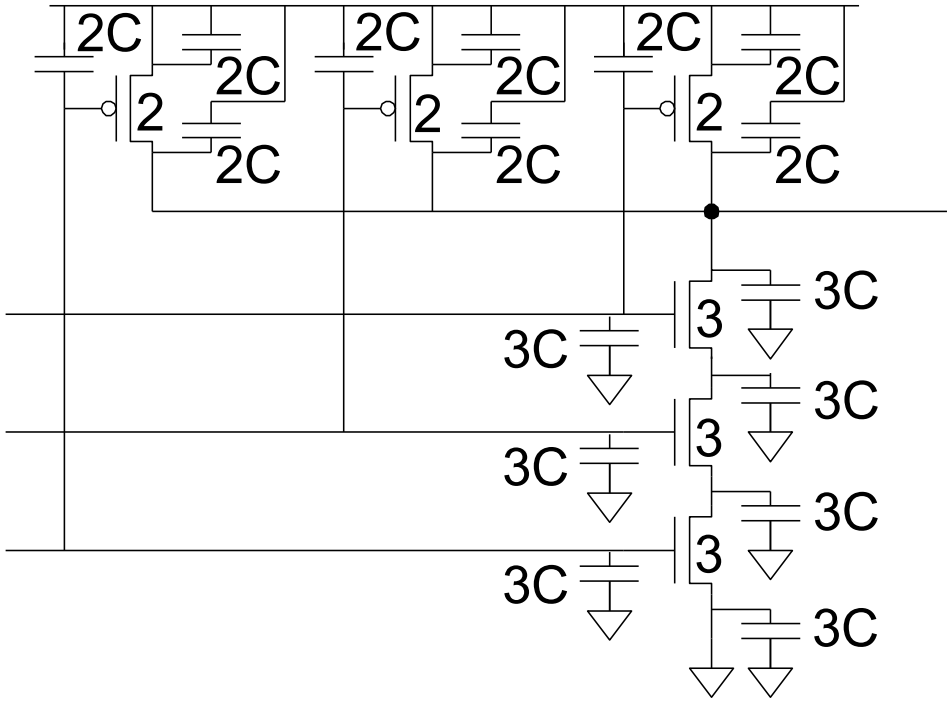
3-input NAND Caps

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



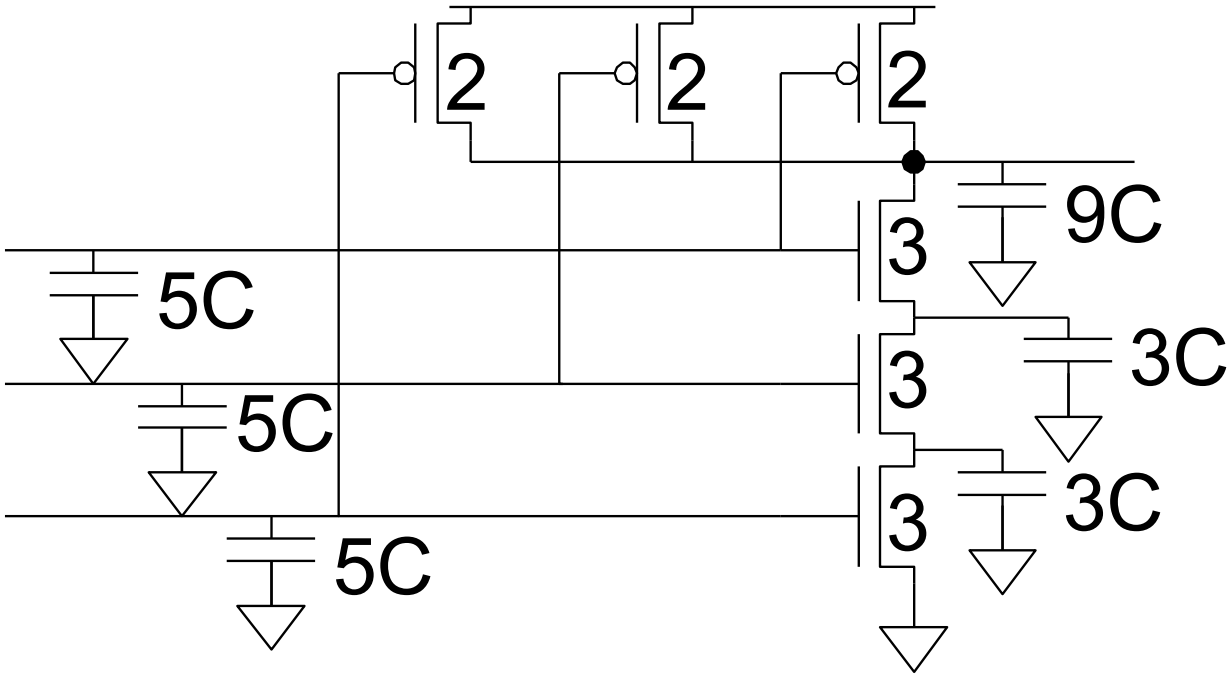
3-input NAND Caps

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

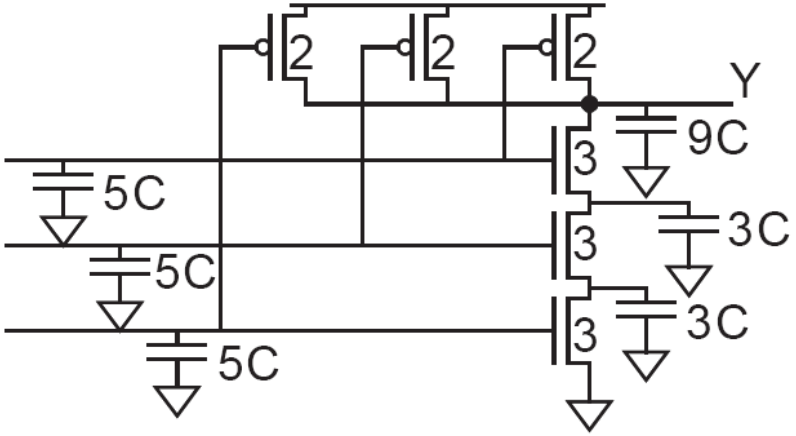


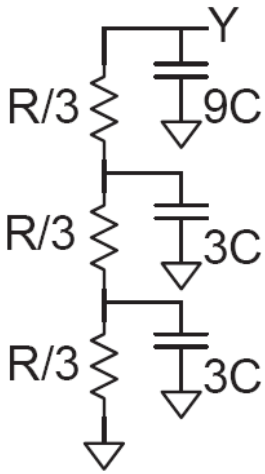
3-input NAND Caps

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

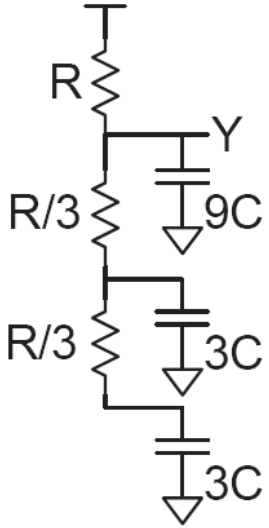


Delay of 3-input NAND





Falling

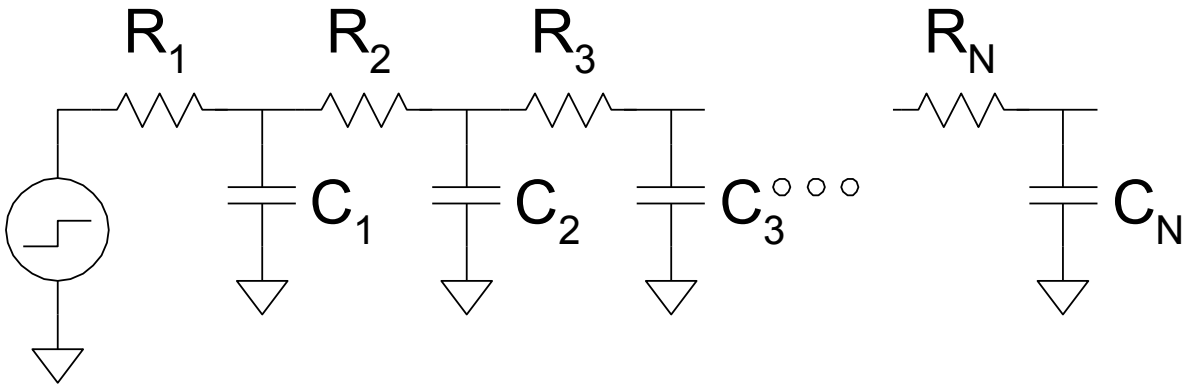


Rising

Elmore Delay Model

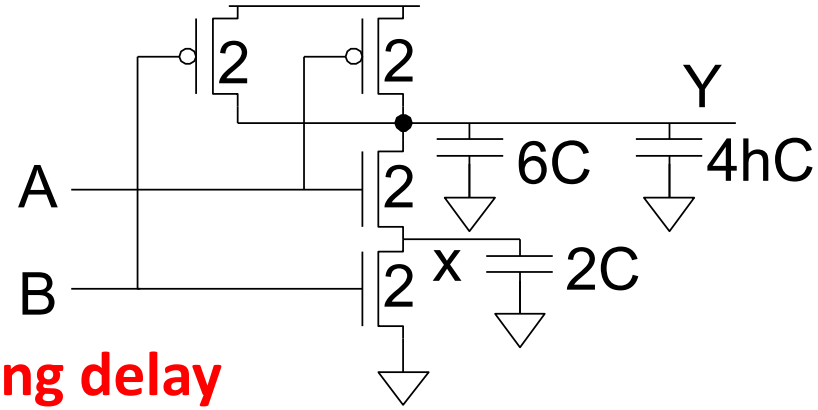
- On transistors look like resistors
- Pullup or pulldown network can be modeled as RC ladder
- Elmore delay model of an RC ladder

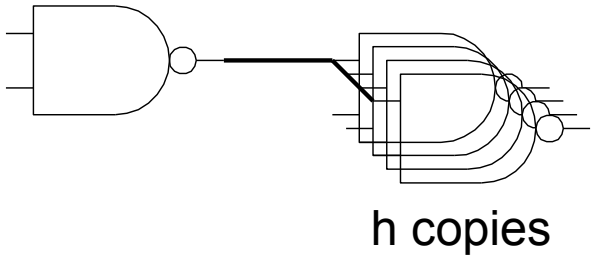
$$t_{pd} = \sum_i R_{n-i} C_i = \sum_{i=1}^{N} C_i \sum_{j=1}^{N} R_j$$



Example: 2-input NAND

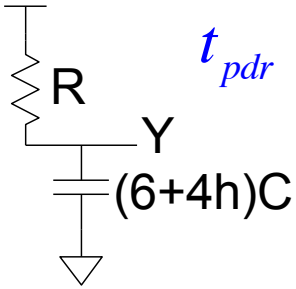
- Estimate worst-case rising and falling delay of 2-input NAND driving ***h*** identical gates



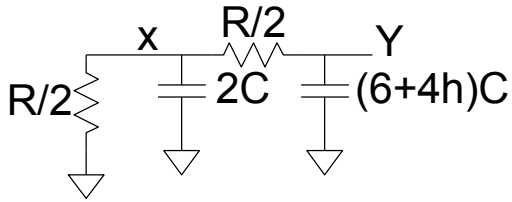


rising delay
(A=1, B=1->0 only 1 pMOS ON)

falling delay
(A=1, B=0->1, X,Y=1->0)



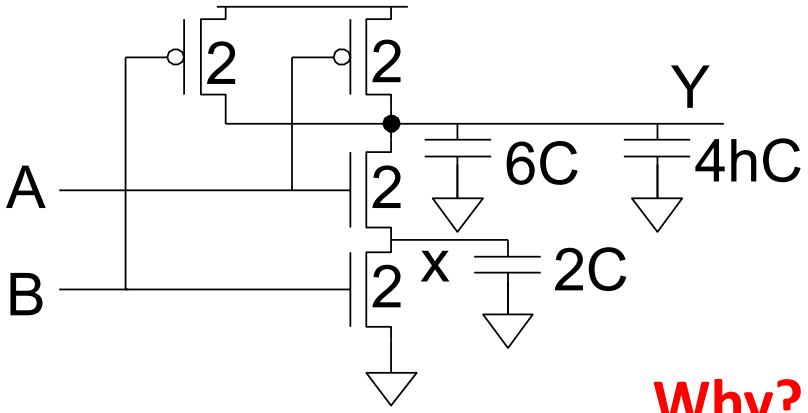
$$t_{pdr} = (6+4h)RC$$

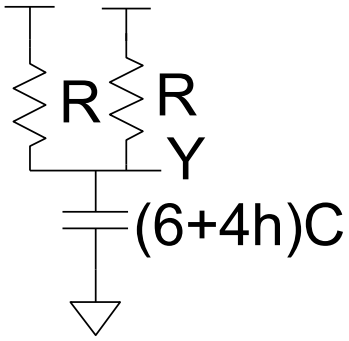


$$t_{pdf} = (2C)\left(\frac{R}{2}\right) + \left[(6+4h)C\right]\left(\frac{R}{2}+\frac{R}{2}\right) = (7+4h)RC$$

Contamination Delay

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





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

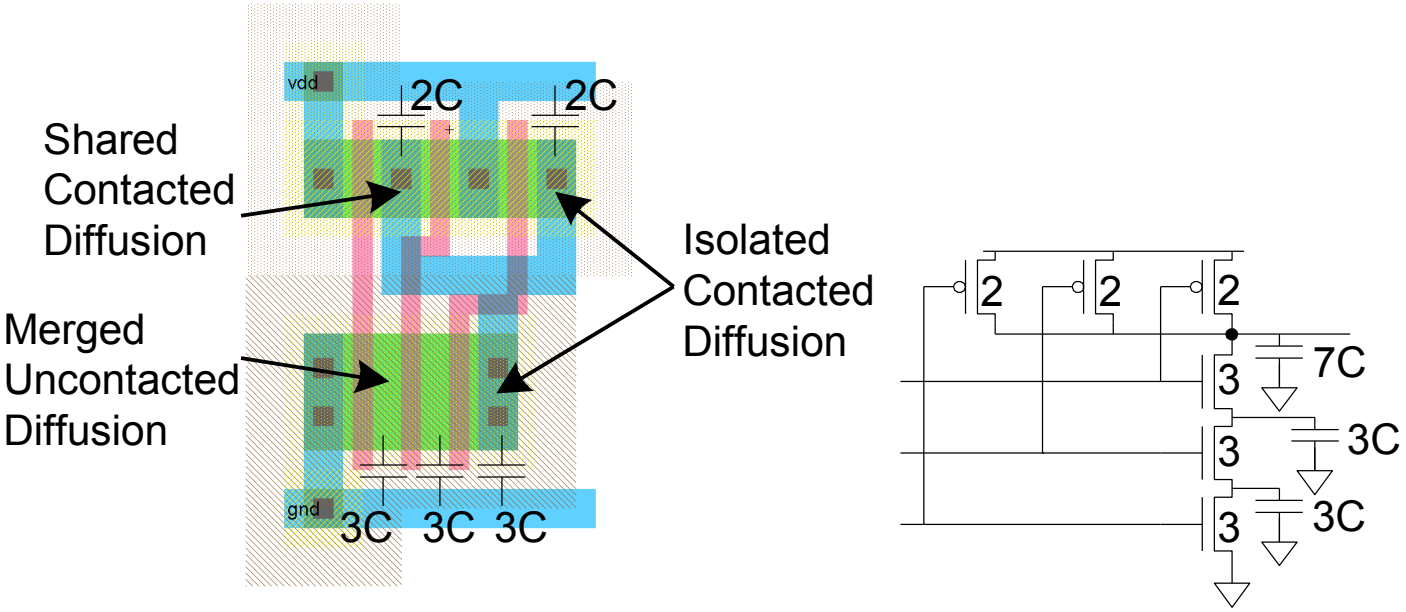
Why?

$t_{cdf} = (6+4h)RC$

Latest input should be connected to transistor closest to the output

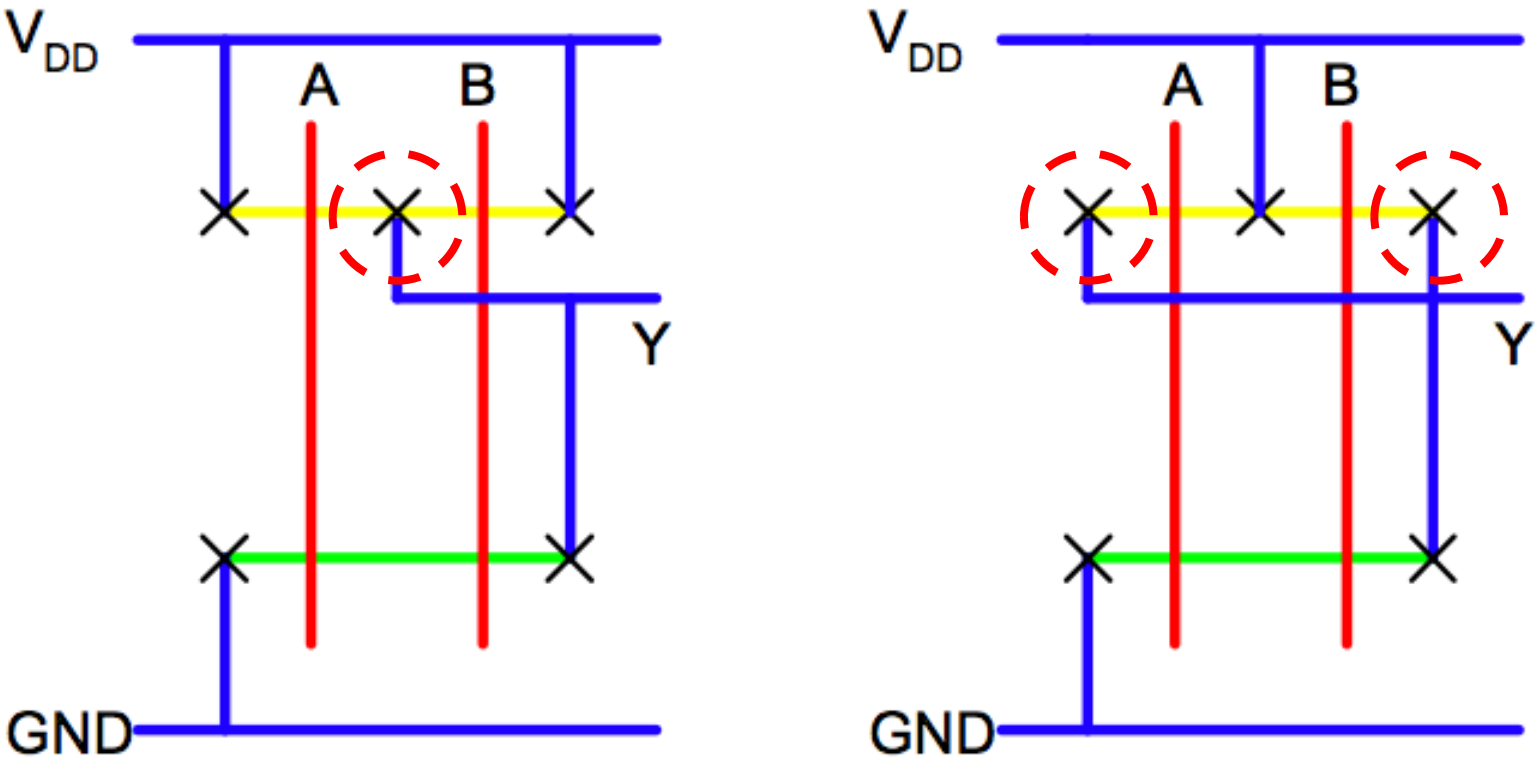
Diffusion Capacitance

- Good layout minimizes diffusion area
- Example: NAND3 layout shares diffusion contact
  - Reduce output capacitance by 2C
  - Merged un-contacted diffusion might help too



Layout Comparison

- Which layout is better



Delay Components

- Parasitic delay
  - 6 or 7 RC
  - Independent of load
- Effort delay
  - 4h RC
  - Proportional to load capacitance

$$t_{pdr} = (6 + 4h)RC$$

$$t_{pdf} = (7 + 4h)RC$$

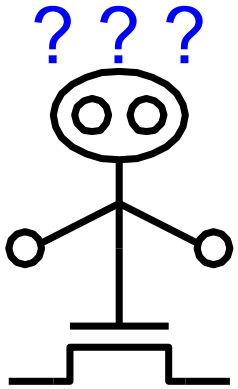
Outline

1. Delay Estimation
2. **Logical Effort and Transistor Sizing**
3. Power Dissipation
4. Interconnect
5. Wire Engineering
6. Design Margin
7. Reliability
8. Scaling

Introduction

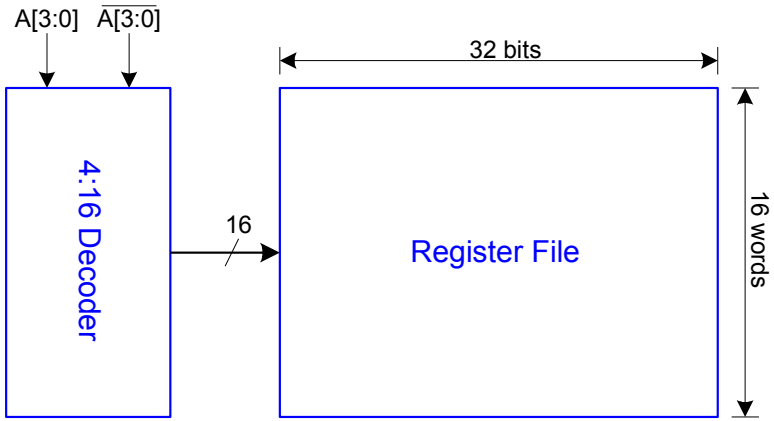
- Chip designers face a bewildering array of choices
  - What is the best circuit topology for a function?
  - How many stages of logic give least delay?
  - How wide should the transistors be?

- 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 Motoroil 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

$\approx$ 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* $f = gh$ (or *stage effort*)
  - Again has two components

Propagation 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 $f = gh$ (or 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

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

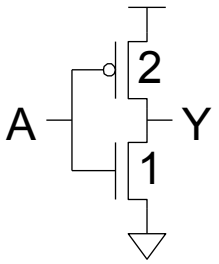
- Delay has two components

$$d = f + p$$

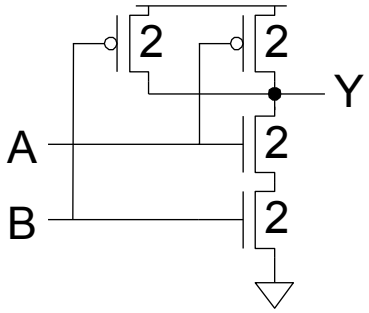
- Parasitic delay *p*
  - Represents delay of gate driving no load
  - Set by internal parasitic capacitance

Computing Logical Effort

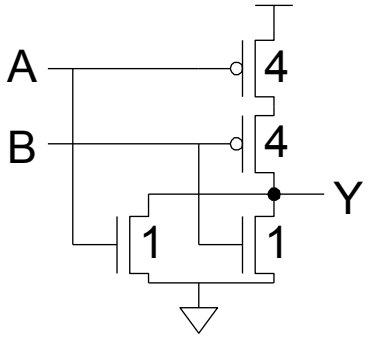
- The ratio of the input capacitance of a gate to the input capacitance of an inverter delivering the same output current
- Measure from delay vs. fanout plots or
- Estimate by counting transistor widths



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



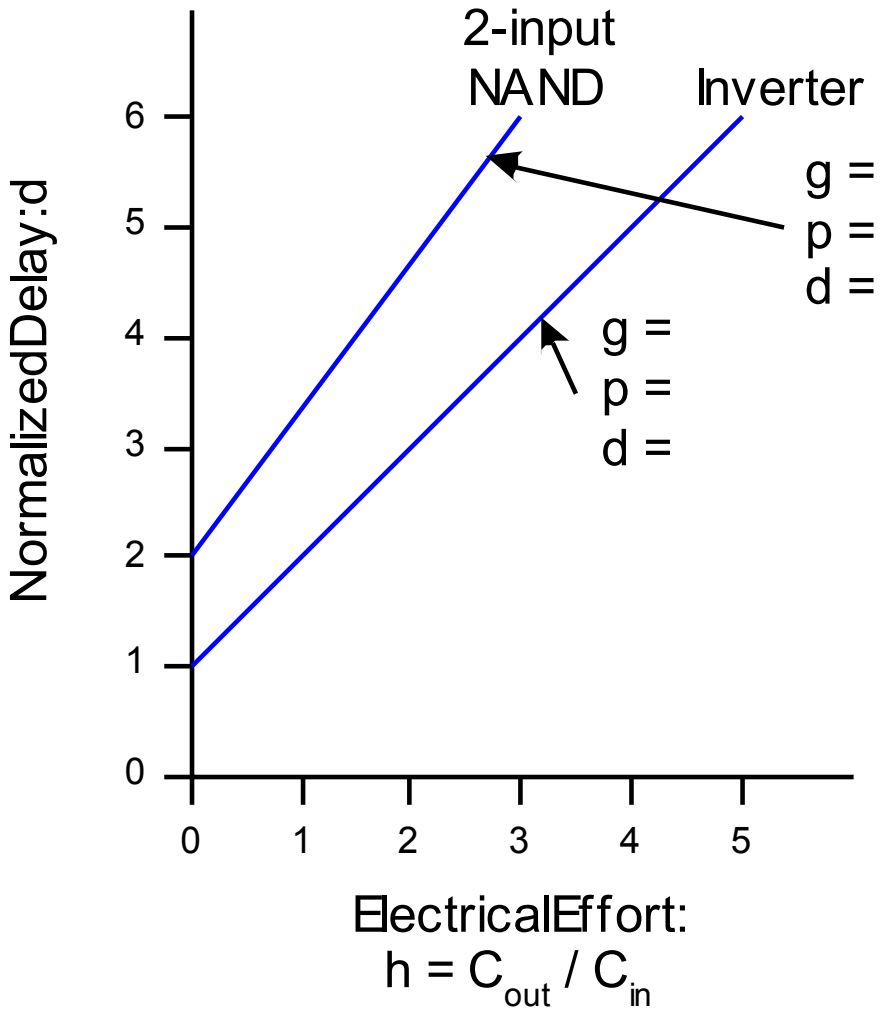
$C_{in} = 4$
$g = 4/3$



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

Delay Plots

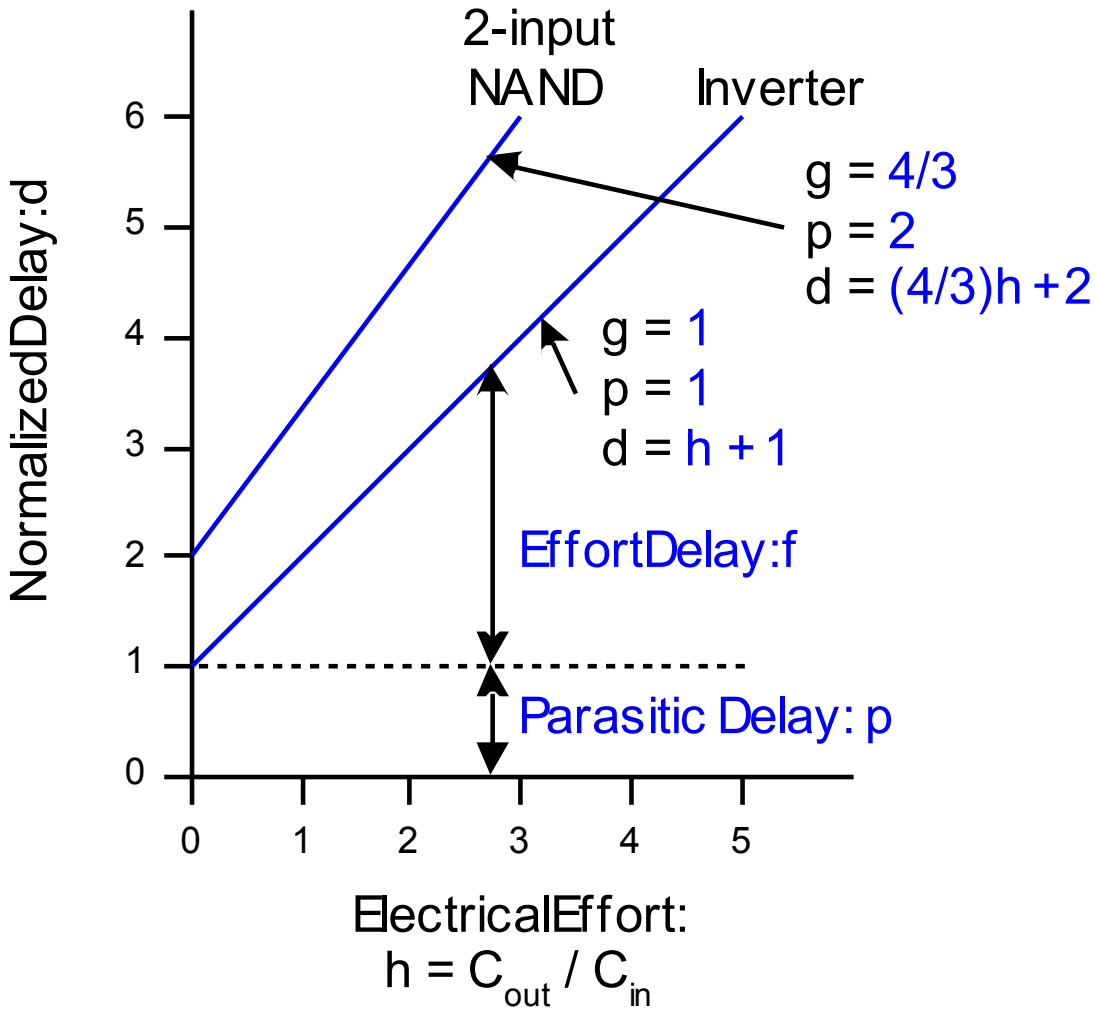
$$d = f + p$$
$$= gh + p$$



Delay Plots

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

- What about NOR2?



Catalog of Gates

- Logic effort of common gates

$$g_i = \frac{C_{in-i}}{C_{in-inv}} = \frac{C_{in-i}}{3C}$$

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
XOR, XNOR		4, 4	6, 12, 6	8, 16, 16, 8	

Catalog of Gates

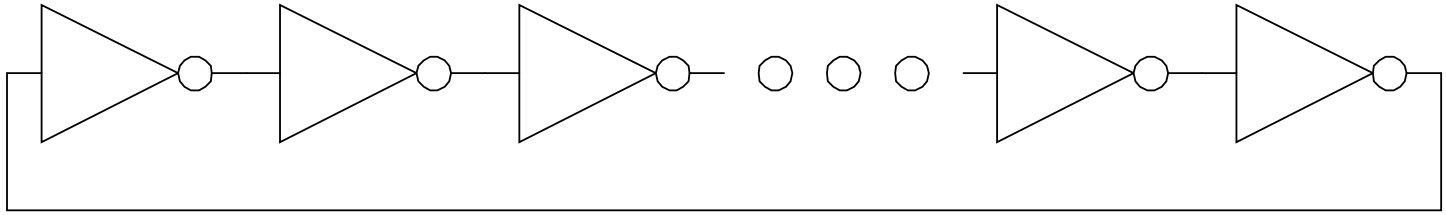
- Parasitic delay of common gates
  - In multiples of P_{inv}(~1)

$$p_i = \frac{C_{p-i}}{C_{p-inv}} = \frac{C_{p-i}}{3C}$$

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



Logic Effort: g = 1

Electrical Effort: h = 1

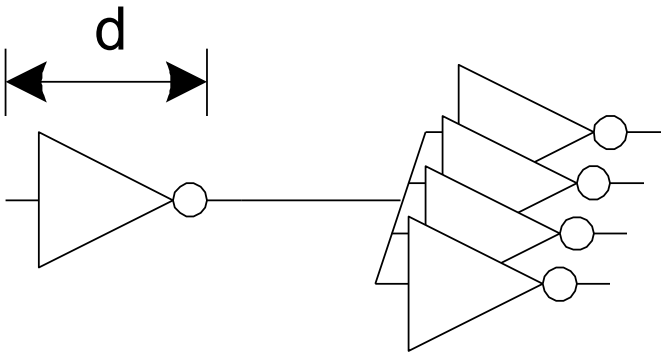
Parasitic Delay: p = 1

Stage Delay: d = 2

Frequency: $f_{osc} = \frac{1}{2Nd} = \frac{1}{4N}$

Example: FO4 Inverter

- Estimate the delay of a fanout-of-4 (FO4) inverter
  - *FO4 delay for a process (ps) is ⅓ to ½ of the channel length (nm). Ex. 180nm : FO4 = 60~90ps → highly sensitive to process, voltage, temperature variation.*

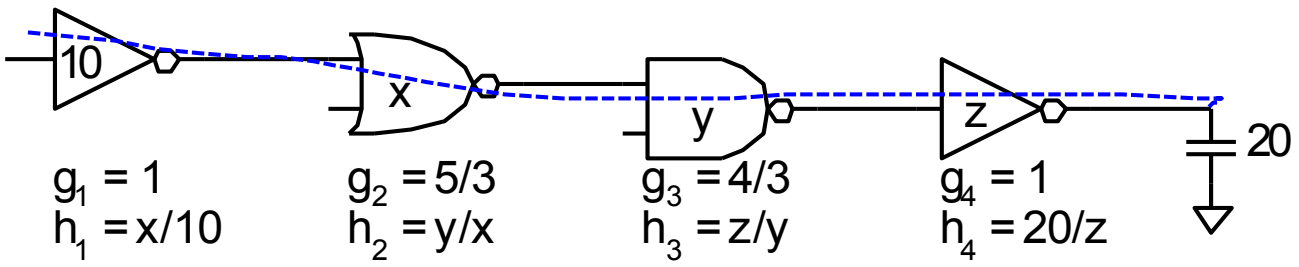


Logic Effort: g = 1
Electrical Effort: h = 4
Parasitic Delay: p = 1
Stage Delay: d = 5

Multistage Logic Networks

- Logic effort generalizes to multistage networks
- Path logical effort $G = \prod g_i$
- Path electrical effort $H = C_{out-path} / C_{in-path}$
- Path effort $F = \prod f_i = \prod g_i h_i$

F=GH?



Paths that Branch

- No! Consider paths that branch

$G = 1$

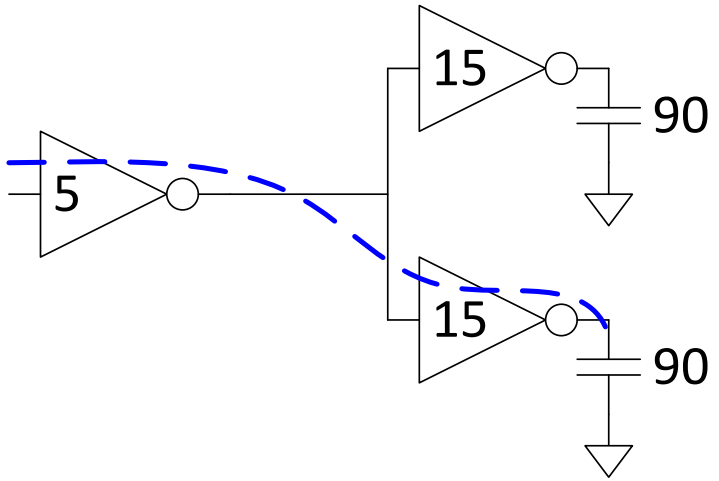
$H = 90/5 = 18$

$GH = 18$

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

$h_2 = 90/15 = 6$

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



Branching Effort

- Accounts for branching between stages in path
  - *Branching effort* $b = \dfrac{C_{\text{on path}} + C_{\text{off path}}}{C_{\text{on path}}}$
  - *Path ranching effort* $B = \prod b_i$

Note: $\prod h_i = BH$

- Now we compute 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

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

- This is a key result of logic effort
  - Find fastest possible delay
  - Doesn't require calculating gate size

Gate Size

- How wide should the gates be for least delay?

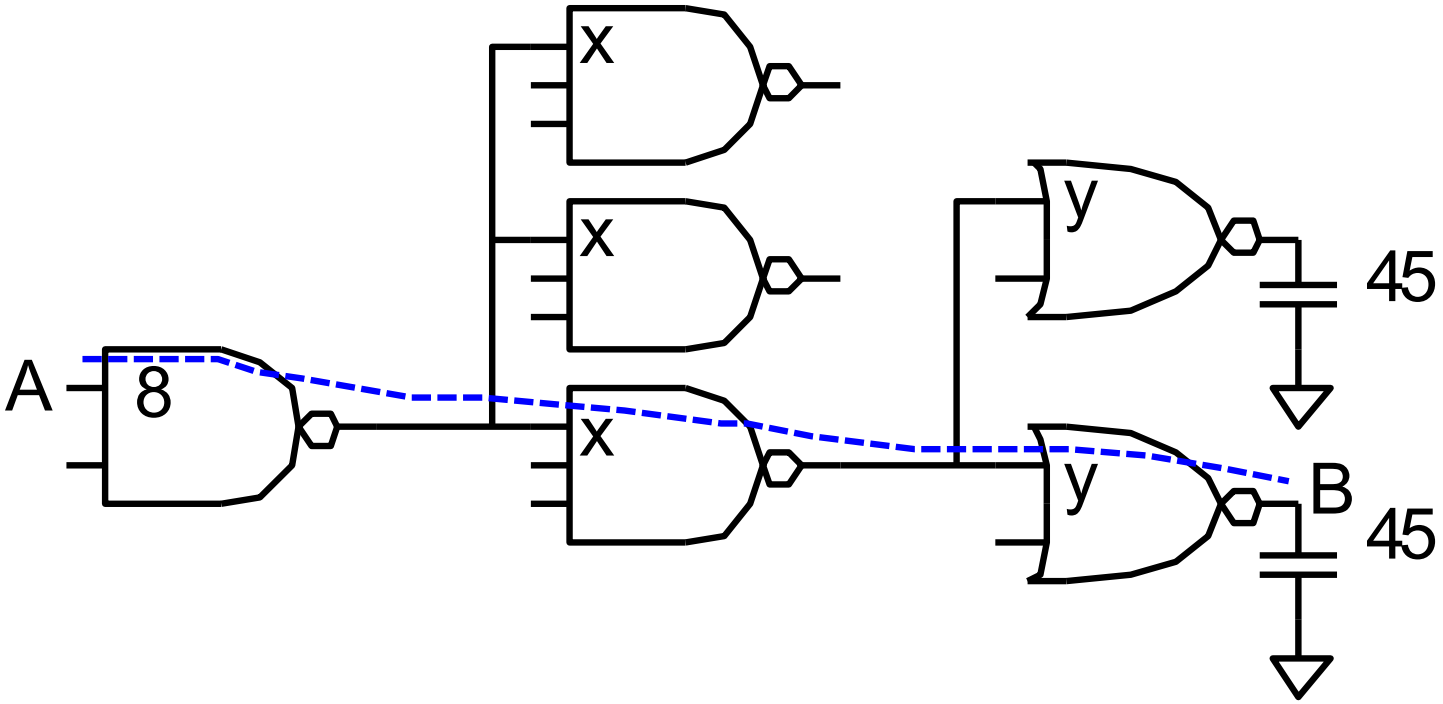
$$\hat{f} = gh = g\frac{C_{out}}{C_{in}}$$

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

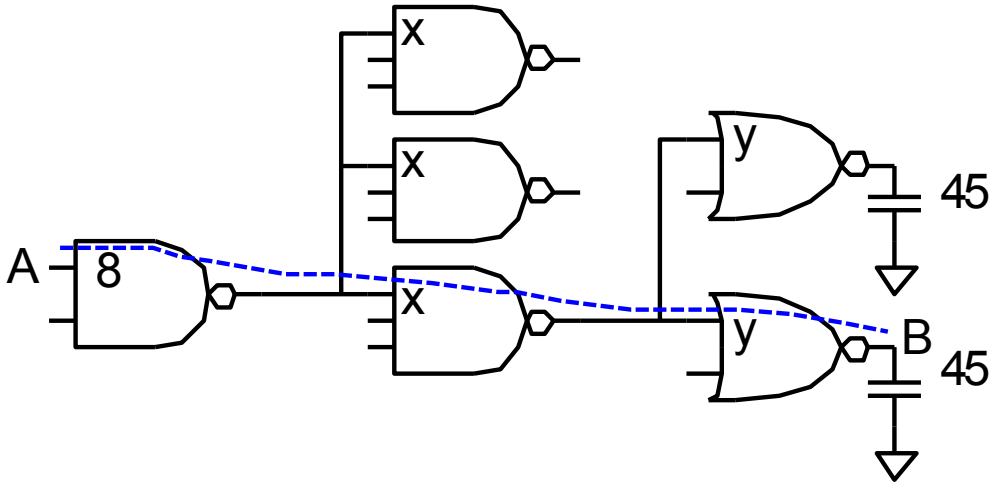
- Working backward, apply capacitance transformation to find input capacitance of each gate with given load it drives
- Check work by verifying input cap spec is met

Example: 3-stage path

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



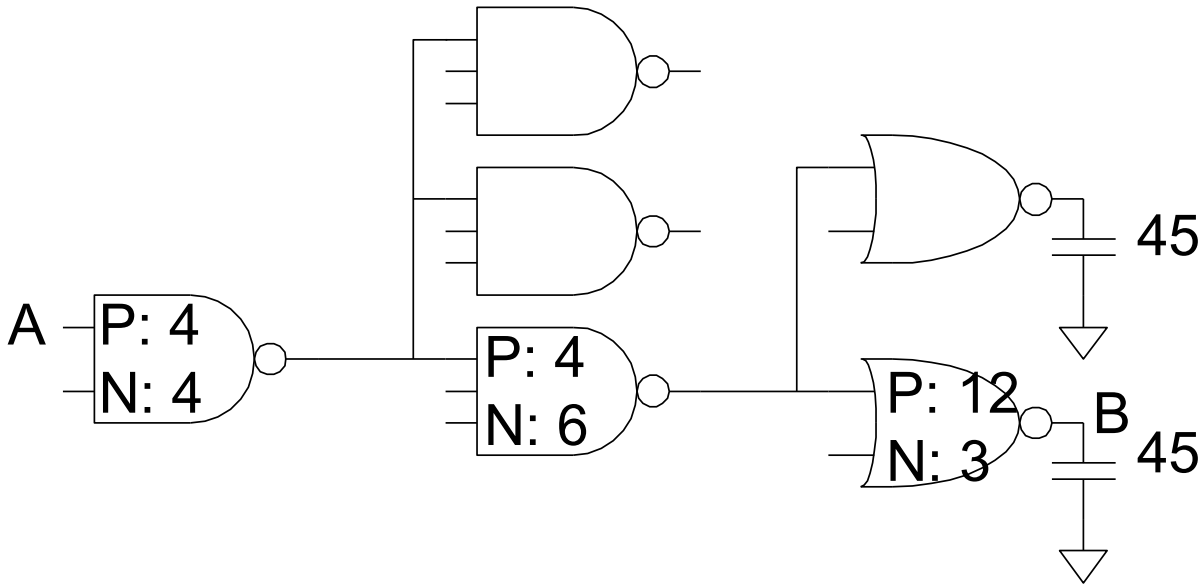
Example: 3-stage path



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 FO4

Example: 3-stage path

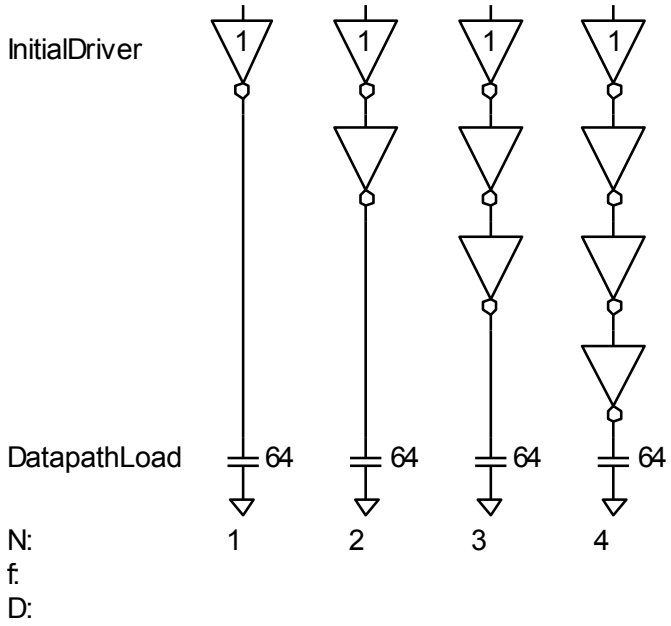
- Work backward for sizes
  - y=45*(5/3)/5=15
  - x=(15*2)*(5/3)/5=10



Best Number of Stages

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

D =

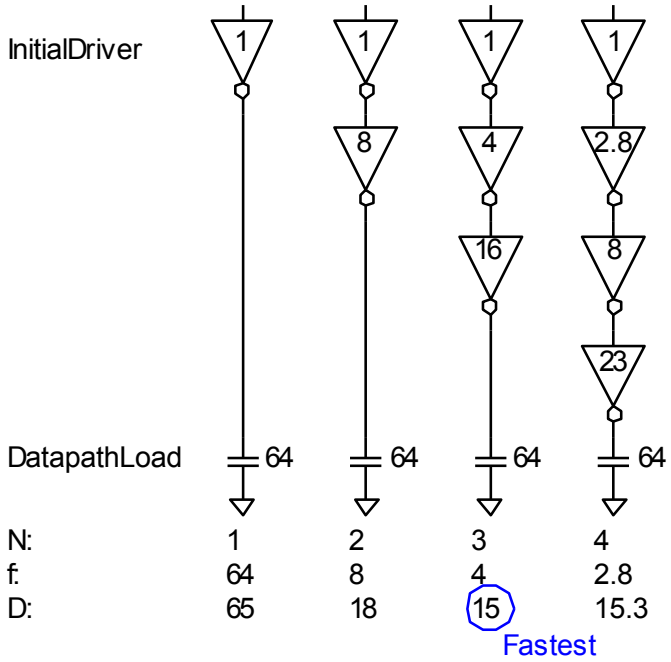


Best Number of Stages

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

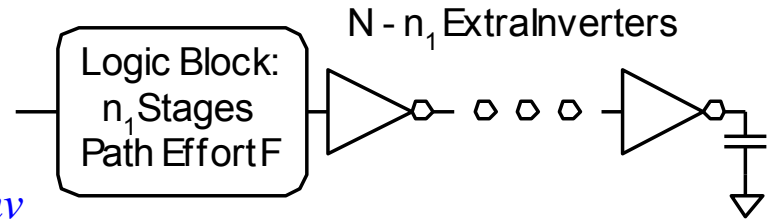
$$D = NF^{1/N} + P$$

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



Derivation

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



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

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

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

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

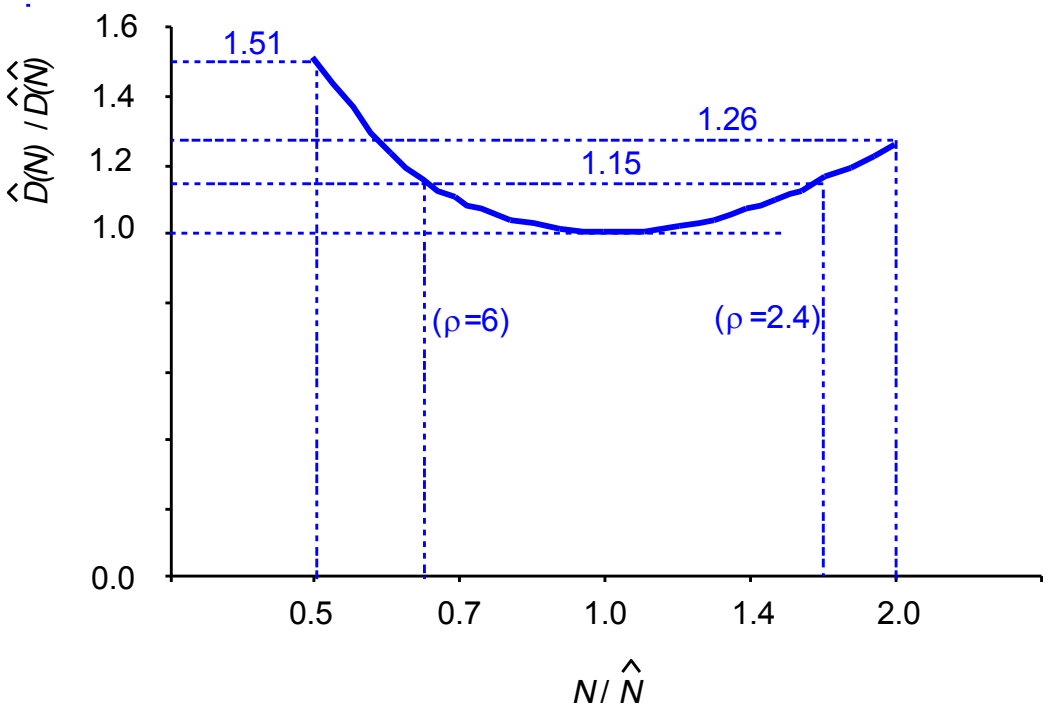
Best Stage Effort

- $p_{inv} + \rho(1 - \ln \rho) = 0$ has no closed-form solution
- Neglecting parasitics (p_{inv}=0), we define

  ρ = 2.718 (e)
- For p_{inv}=1, solve numerically for ρ = 3.59

Sensitivity Analysis

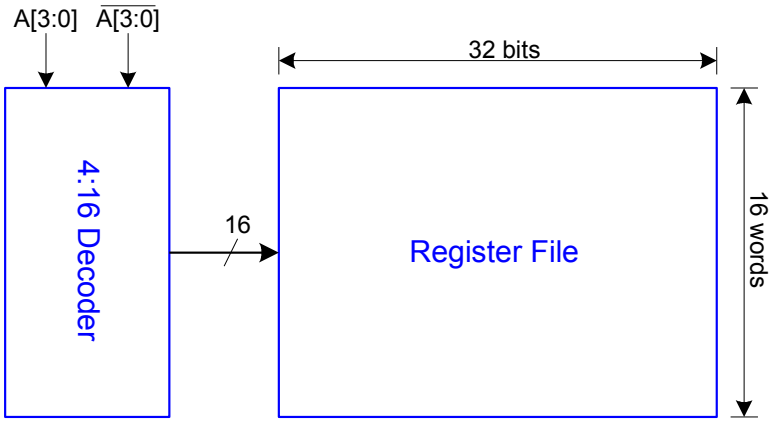
- How sensitive is delay to using exactly the best number of stages?



- 2.4< ρ <6 gives delay with 15% of optimal
  - 4 is a convenient choice

Example, Revisited

- Ben Bitdiddle is the memory designer for the Motoroil 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?

Number of Stages

- Decoder effort is mainly electrical and branching

  Electrical Effort: H =

  Branching Effort: B =

- If we neglect logical effort (assume G = 1)

  Path Effort: F =

  Number of Stages: N =

Number of Stages

- Decoder effort is mainly electrical and branching

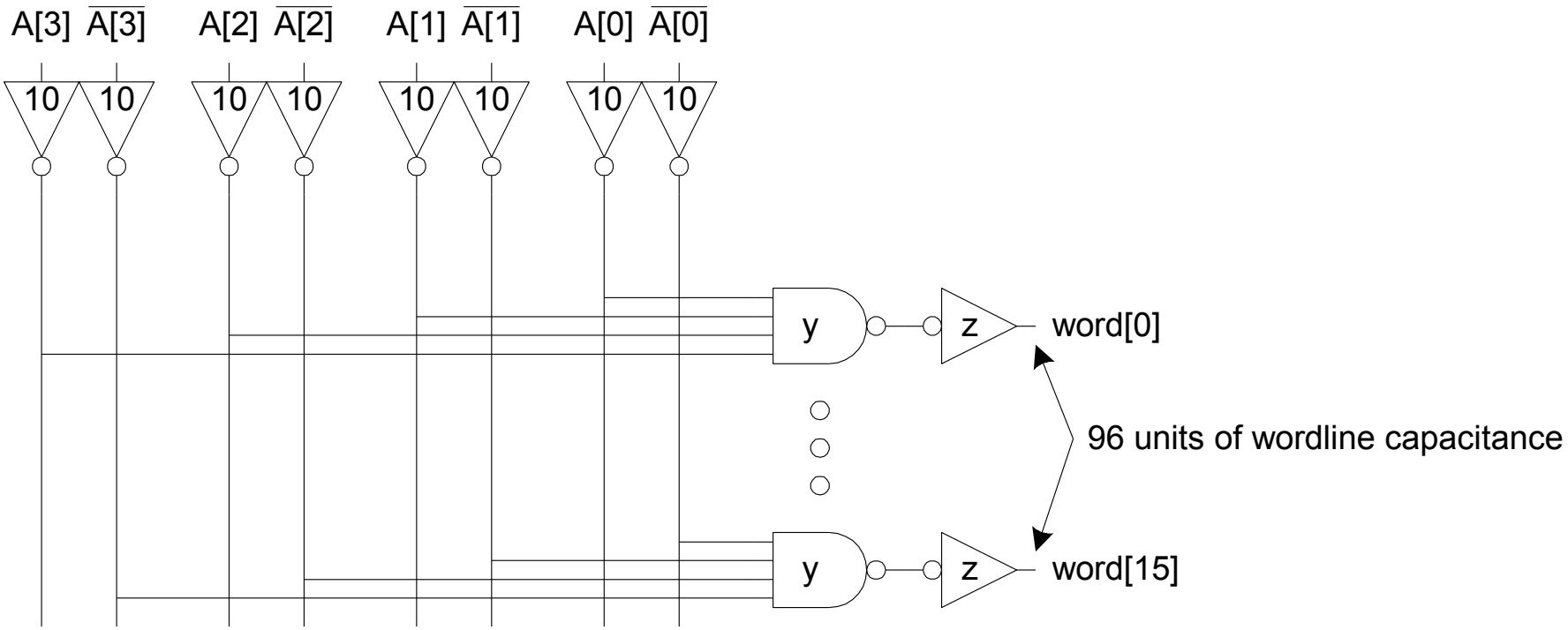
  Electrical Effort: H = (32*3) / 10 = 9.6

  Branching Effort: B = 8
- If we neglect logical effort (assume G = 1)

  Path Effort: F = GBH = 76.8

  Number of Stages: $N = \log_4 F = 3.1$
- Try a 3-stage design

3 Stage 4:16 Decoder



Gate Sizes & Delay

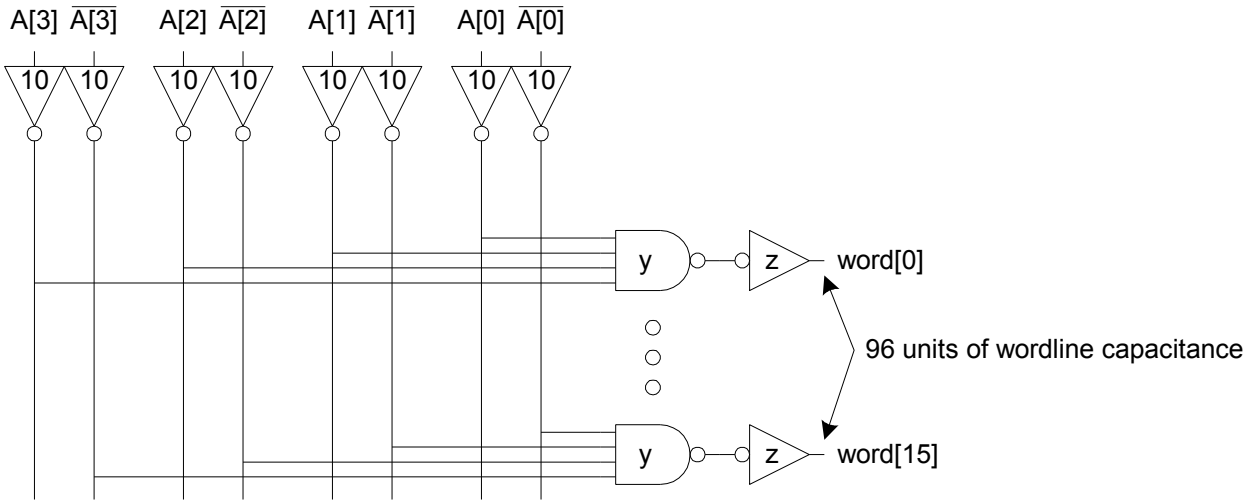
Logical Effort: G =

Path Effort: F =

Stage Effort: $\hat{f} =$

Path Delay: $D =$

Gate sizes: z = y =



Gate Sizes & Delay

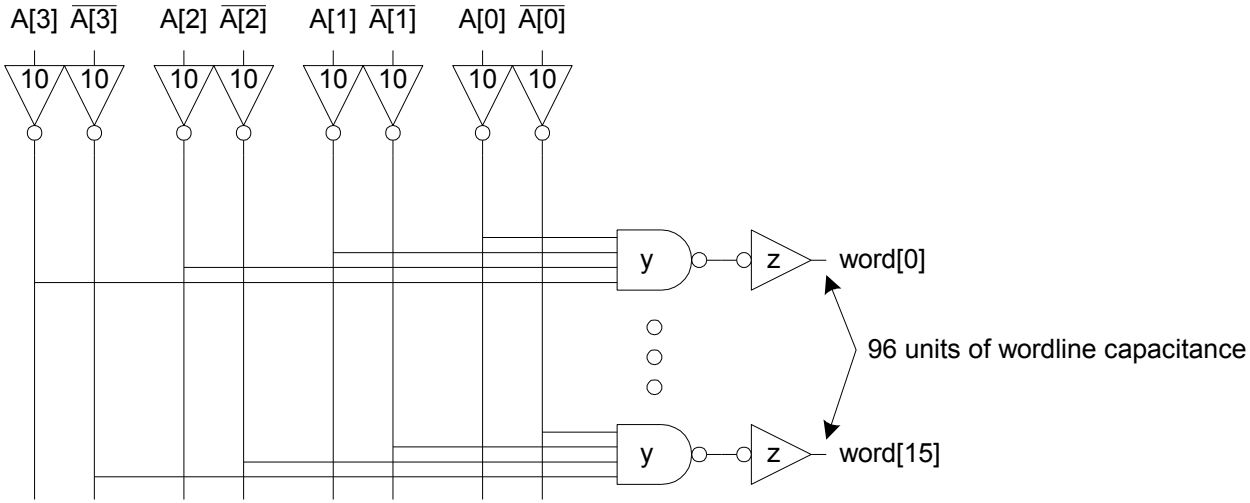
Logical Effort: G = 1 * 6/3 * 1 = 2

Path Effort: F = GBH = 154

Stage Effort: $\hat{f} = F^{1/3} = 5.36$

Path Delay: $D = 3\hat{f} + 1 + 4 + 1 = 22.1$

Gate sizes: z = 96*1/5.36 = 18 y = 18*2/5.36 = 6.7



Comparison

- Compare many alternatives with a spreadsheet

Design	N	G	P	D
NAND4-INV	2	2	5	29.8
NAND2-NOR2	2	20/9	4	30.1
INV-NAND4-INV	3	2	6	22.1
NAND4-INV-INV-INV	4	2	7	21.1
NAND2-NOR2-INV-INV	4	20/9	6	20.5
NAND2-INV-NAND2-INV	4	16/9	6	19.7
INV-NAND2-INV-NAND2-INV	5	16/9	7	20.4
NAND2-INV-NAND2-INV-INV-INV	6	16/9	8	21.6

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, velocity saturation & body effect …
- No Interconnect account
  - 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 ~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