Very Large Scale Integration (VLSI)

Lecture 4

Dr. Ahmed H. Madian

Ah_madian@hotmail.com

Contents

- Delay estimation
  - Simple RC model
  - Penfield-Rubenstein Model
  - Logical effort
- Delay minimization techniques
  - Transistor sizing
  - Wiring sizing
  - Distributed drivers
  - Large driver
- Wiring techniques

Circuit characterization & performance

- Resistance estimation
- Capacitance estimation
- Inductance estimation
- Delay estimation
  - Simple RC model
  - Penfield-Rubenstein Model
- Delay minimization techniques
  - Transistor sizing
  - Distributed drivers
  - Driving large loads
- Wiring techniques

Delay Estimation

How to estimate the delay for your layout?

1- use a simulator

2- Solve the differential equation to get the exact delay

3- model your circuit using one of the defined delay models like RC

Signal Delay Time

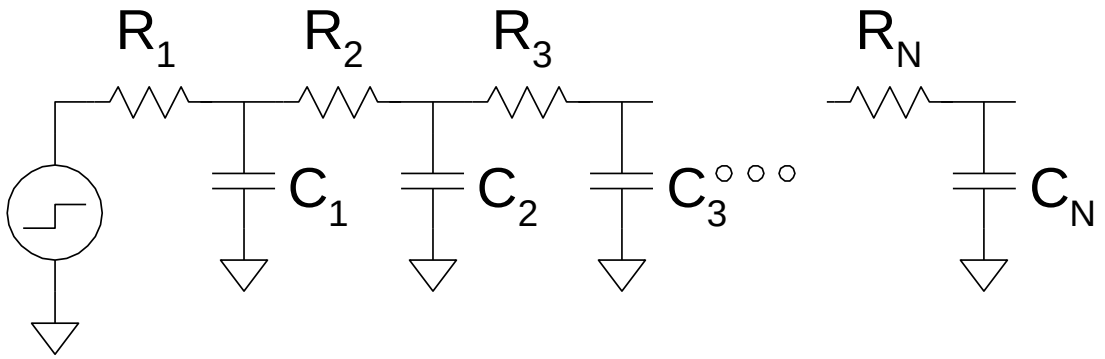
- Signal delay time is composed as follows
  - Gate delay time
  - Interconnection delay time
- due to minimization the delay times decreases
- the output impedance of buffers increases, thus the importance of interconnection delays increases
- So, signal delay time becomes less dependent on gate delay but more dependent on interconnection delay time

Elmore Delay

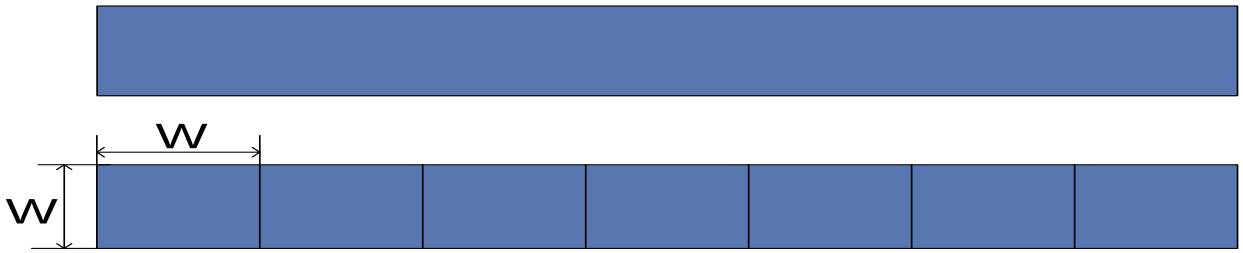
- ON transistors look like resistors
- Pull-up or pull-down network modeled as *RC ladder*
- Elmore delay of RC ladder

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

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



Routing Delay estimation (cont.)

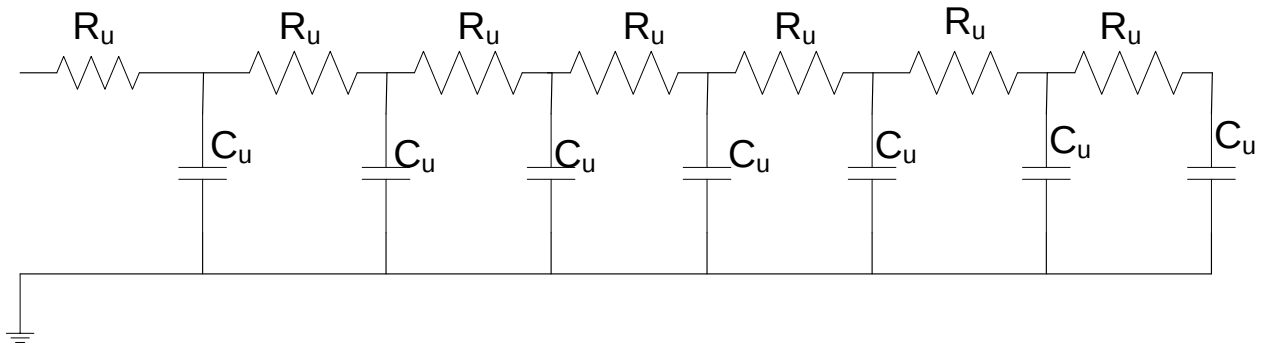


Elmore Delay:

$R_{total} = n\ R_U$

$C_{total} = n\ C_U$

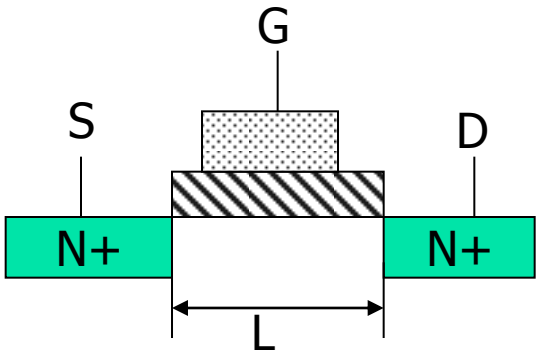
$\tau_{total} = n^2 R_U\ C_U$



Delay Estimation

- For MOSFET transistor,

$$\tau_{transit} = \frac{L}{velocity}$$



Velocity = μ.ζ

Where μ = mobility and ζ = electric filed = V_{DS}/L

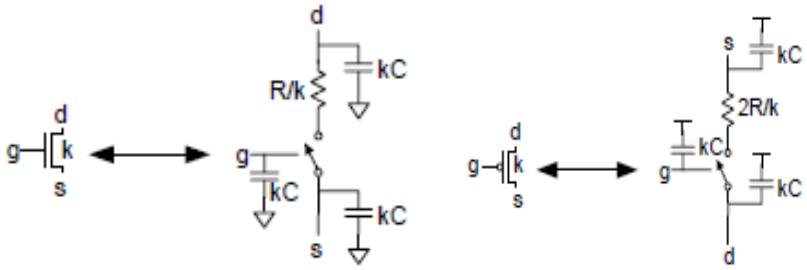
$$\tau_{transit} = \frac{L}{\mu \frac{V_{DS}}{L}} = \frac{L^2}{\mu V_{DD}}$$

For lower delay,

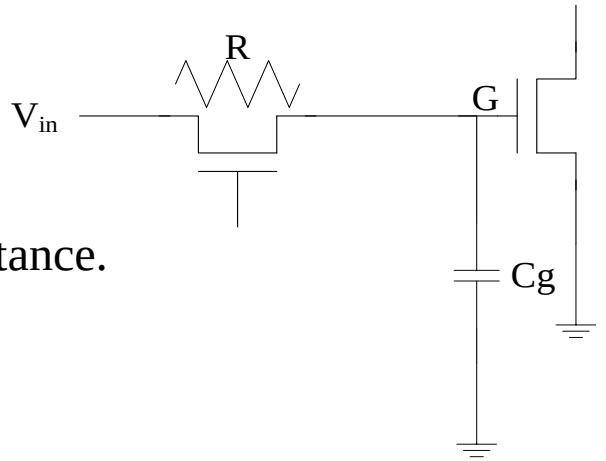
- V_{DD} may be increased but we have limit of break down
- L may be decreased but there's a problem with technology and Resistance

Delay estimation (cont.)

- Equivalent circuit used for MOSFET
  - 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 is inversely proportional to width



- Define a model for delay based on the unit delay
  - Treat every MOSFET as resistance.
  - Lump intermediate node capacitance with load capacitance.



$$\tau_u = R_S * C_{\text{Gate of min size transistor}}$$

Simple RC model

•Assume all pull –up /pull-down resistors are summed as R_{pu}/R_{pd} and all capacitance are summed in the output capacitance

$R_P = 2.5R_n$

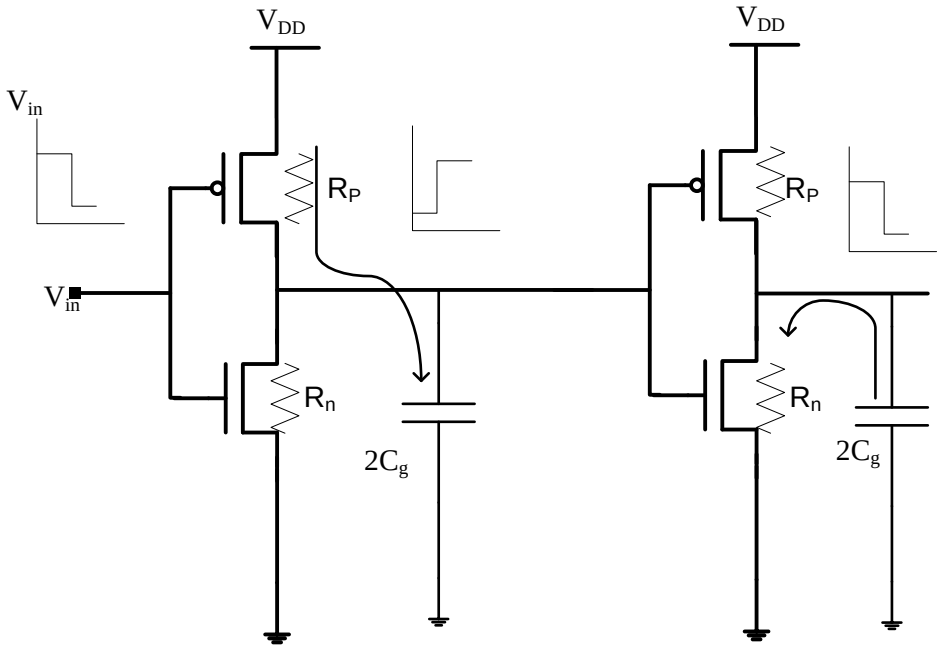
$\tau_u = R_n C_g$

$\tau_{inv1} = 2 R_P C_g$

$\tau_{inv1} = 5 R_n C_g$

$\tau_{inv2} = 2 R_n C_g$

Inverter pair delay = $\tau_{inv1} + \tau_{inv2} = 7 R_n C_g$

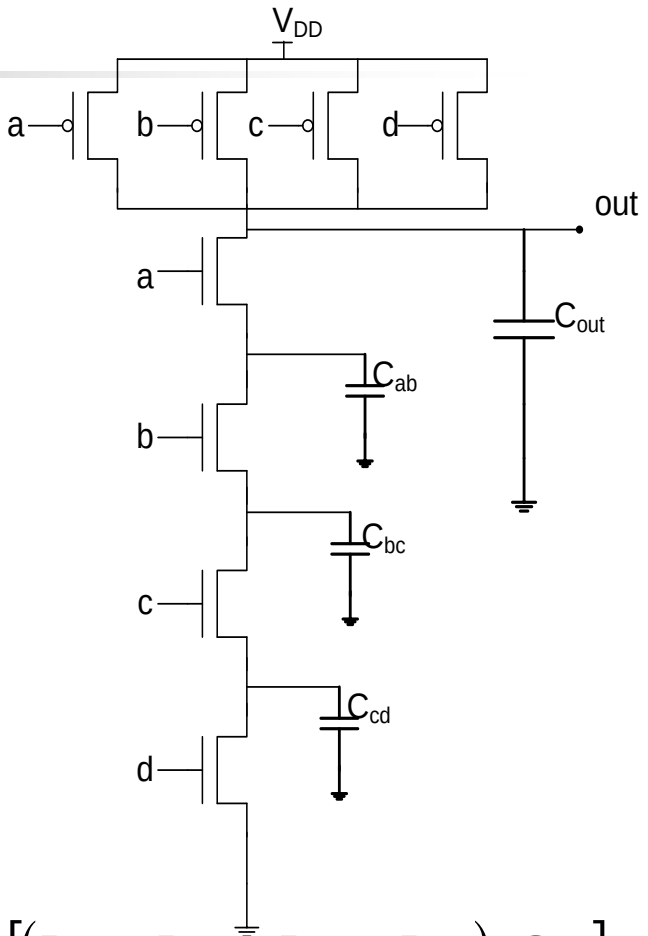


Elmore Delay Model

- Calculate the delays in generalized RC trees.
- For a group of transistors in series (as in NAND gate),

$$t_d = \sum_i R_i C_i$$

- Where R_i is the summed resistance from point i to power or ground and C_i is the capacitance at point i.
- Ex.: for 4 input NAND gate the fall time will be,



$$t_{df} = (R_{N1} x C_{cd}) + [(R_{N1} + R_{N2}) x C_{bc}] + [(R_{N1} + R_{N2} + R_{N3}) x C_{ab}] + [(R_{N1} + R_{N2} + R_{N3} + R_{N4}) x C_{out}]$$

Delay Components

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

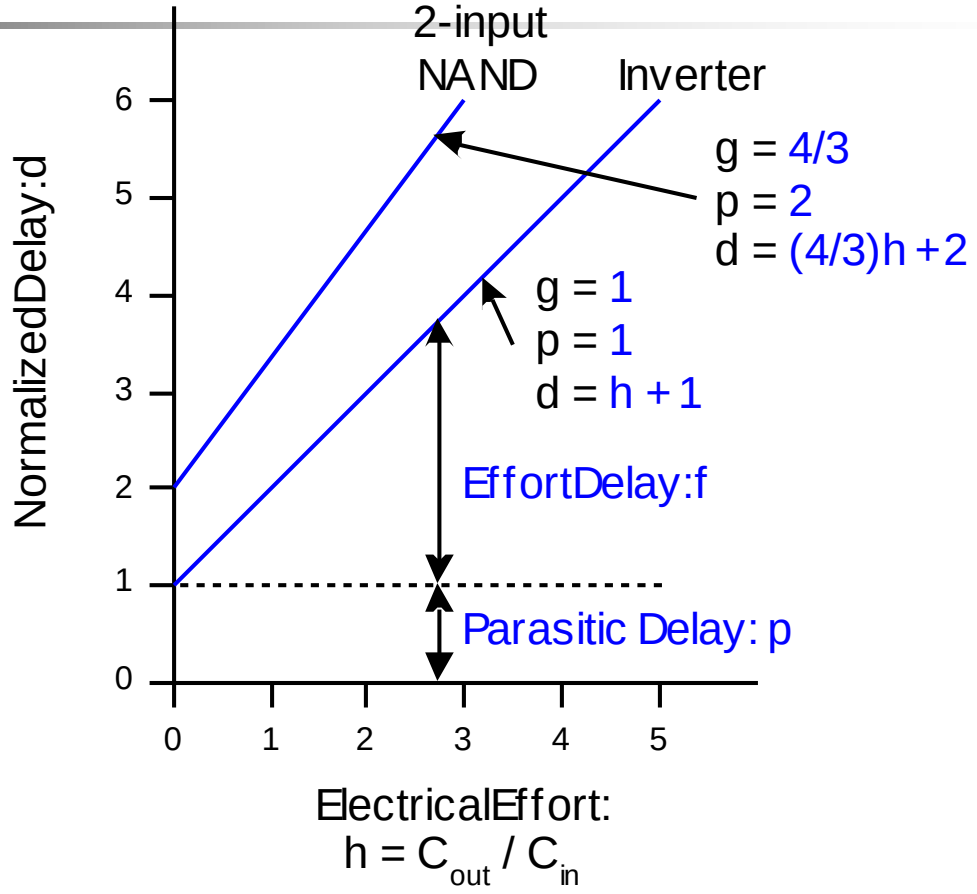
Logical Effort

- 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

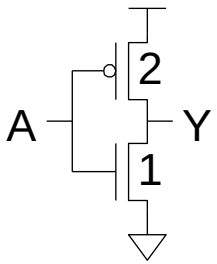
Delay Plots

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

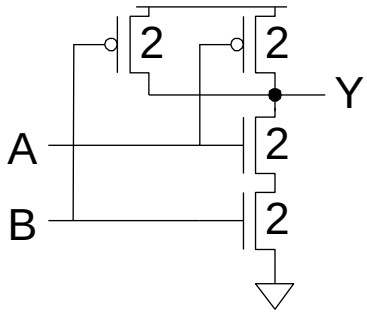


Computing Logical Effort

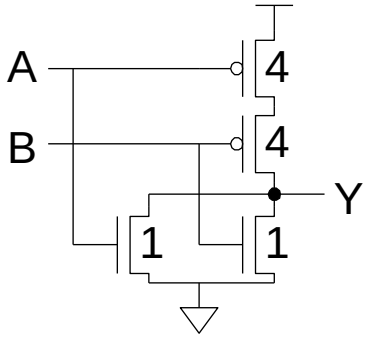
- DEF: *Logical effort is the ratio of the input capacitance of a gate to the input capacitance of an inverter delivering the same output current*.
- Measure from delay vs. fanout plots
- Or estimate by counting transistor widths



C_{in} = 3
g = 3/3



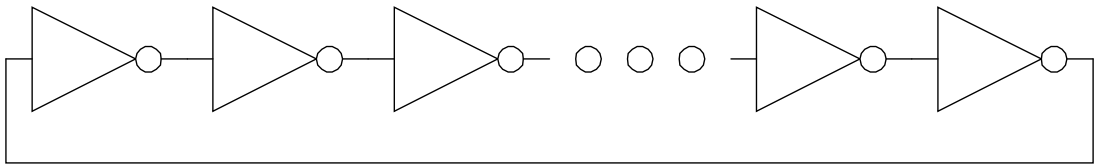
C_{in} = 4
g = 4/3



C_{in} = 5
g = 5/3

Example: Ring Oscillator

- Estimate the frequency of an N-stage ring oscillator



Logical Effort: g =

Electrical Effort: h =

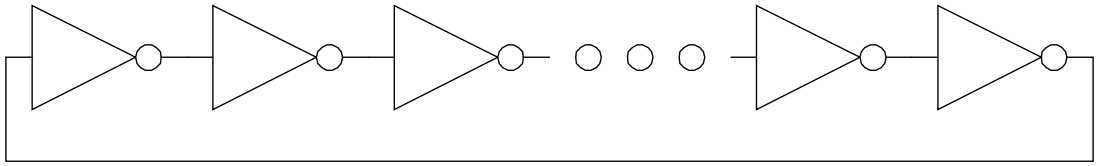
Parasitic Delay: p =

Stage Delay: d =

Frequency: f_{osc} =

Example: Ring Oscillator

- Estimate the frequency of an N-stage ring oscillator



31 stage ring oscillator in 0.6 μm process has frequency of ~ 200 MHz

Logical Effort: g = 1

Electrical Effort: h = 1

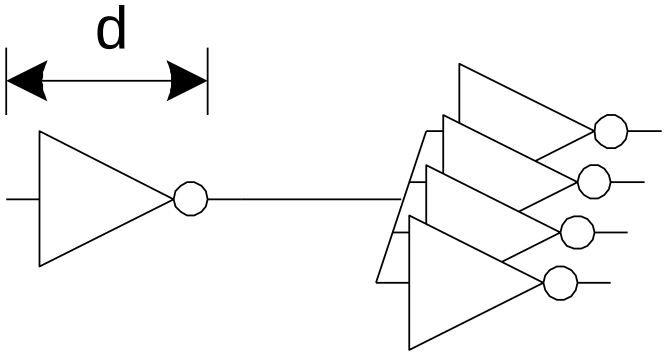
Parasitic Delay: p = 1

Stage Delay: d = 2

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

Example: FO4 Inverter

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



Logical Effort: g =

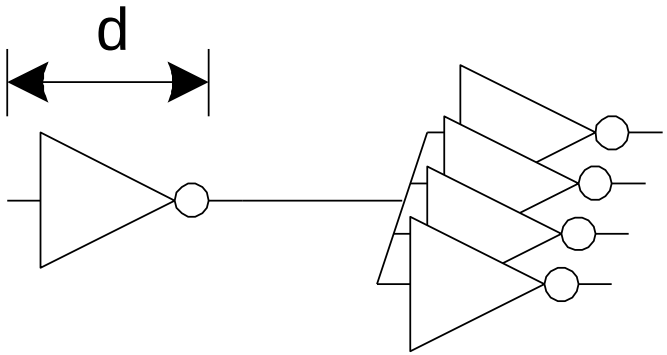
Electrical Effort: h =

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

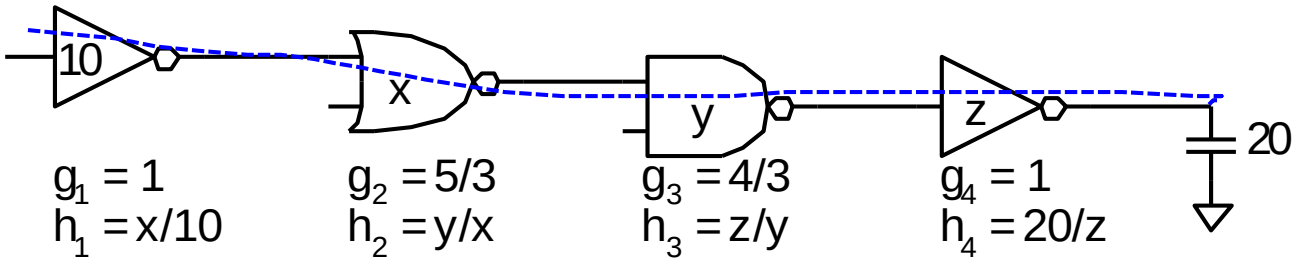
200 ps in 0.6 μm process

60 ps in a 180 nm process

f/3 ns in an *f* μm 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 =

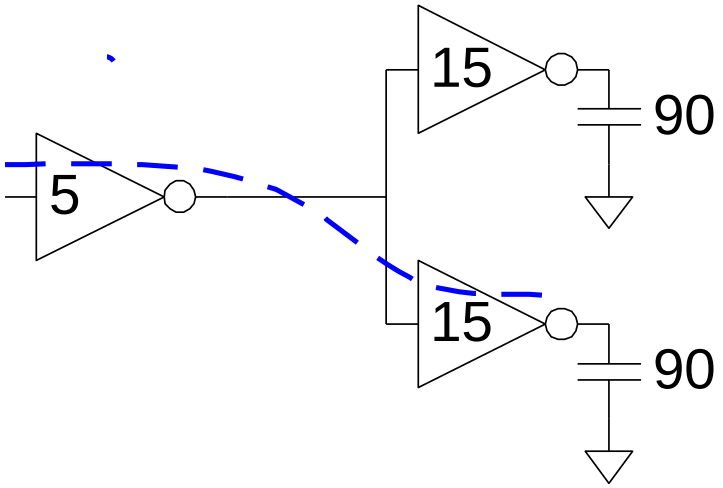
  H =

  GH =

  h_1 =

  h_2 =

  F = GH?



Paths that Branch

- No! Consider paths that branch:

$G = 1$

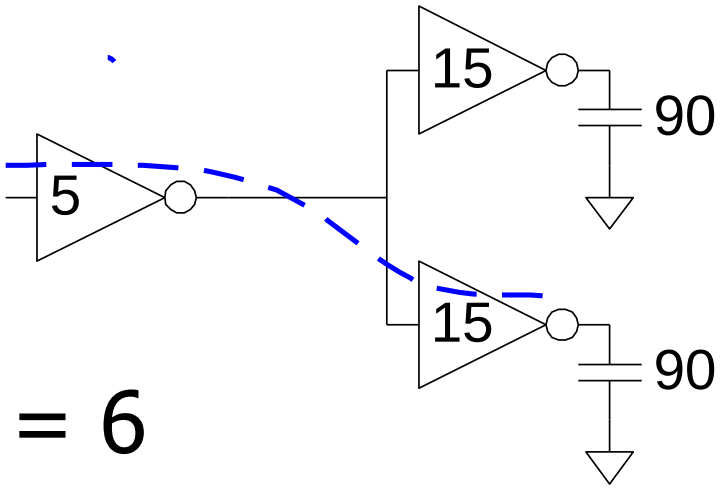
$H = 90 / 5 = 18$

$GH = 18$

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

$h_2 = 90 / 15 = 6$

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



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?

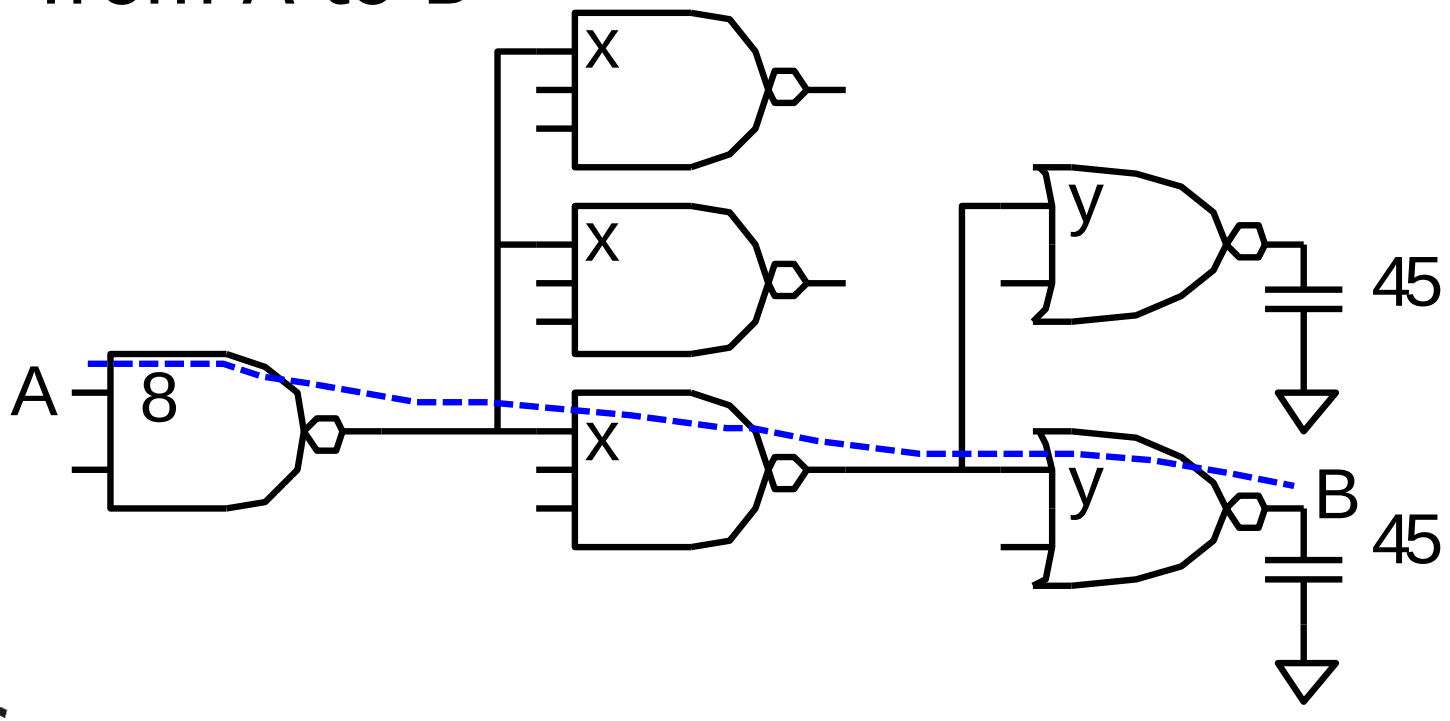
$$\hat{f} = gh = g\frac{C_{out}}{C_{in}}$$

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

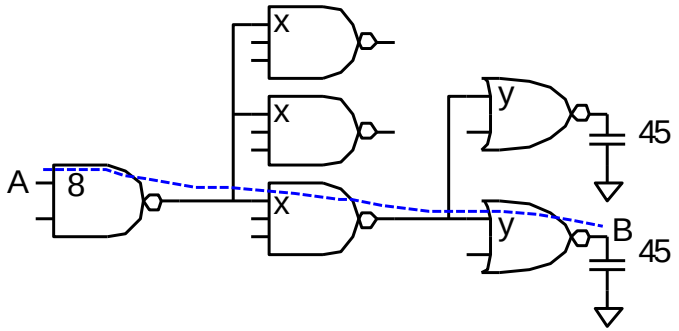
- Working backward, apply capacitance transformation to find input capacitance of each gate given load it drives.
- Check work by verifying input cap spec is met.

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 =

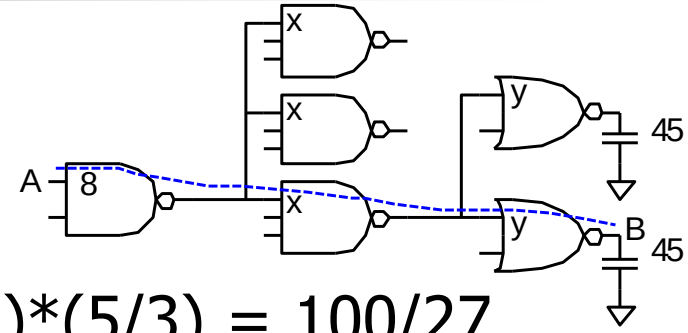
Path Effort F =

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

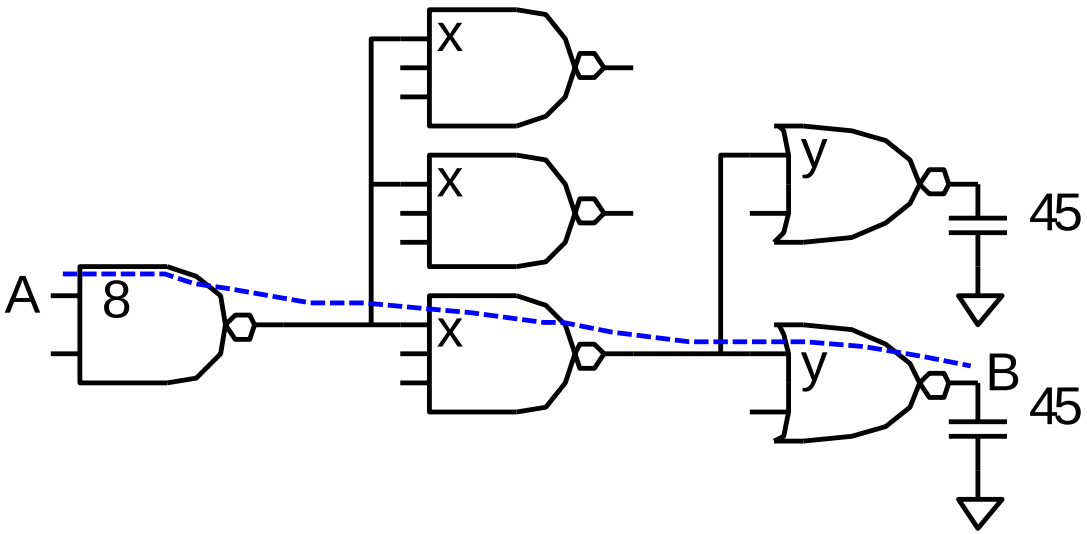
Delay D = 3*5 + 7 = 22 = 4.4 FO4

Example: 3-stage path

- Work backward for sizes

  y =

  x =

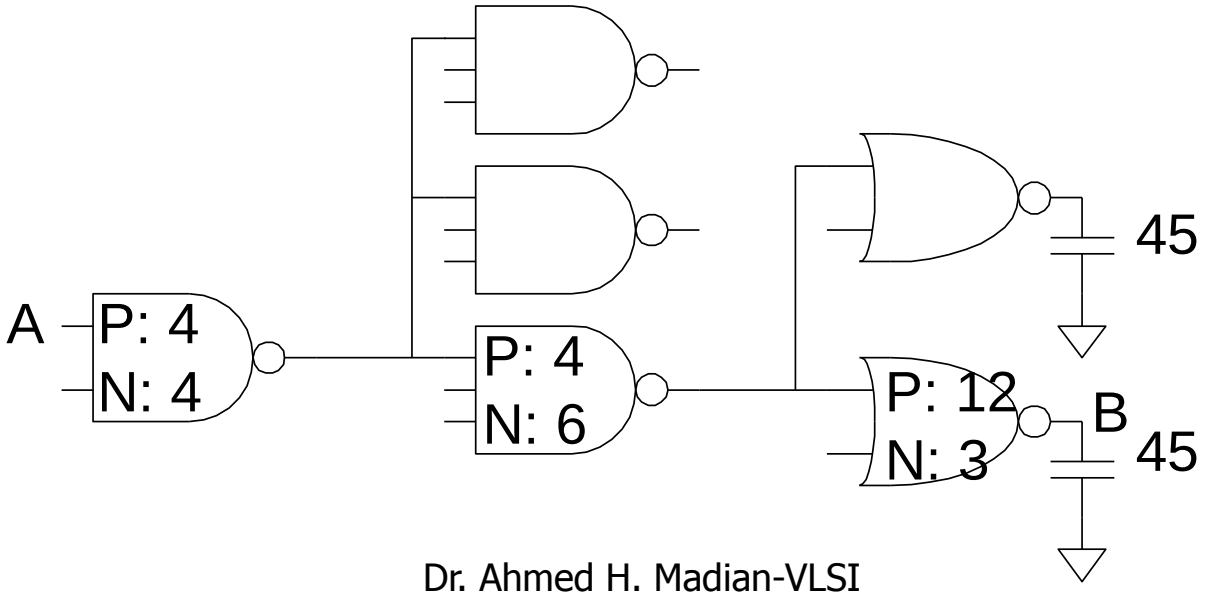


Example: 3-stage path

- Work backward for sizes

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

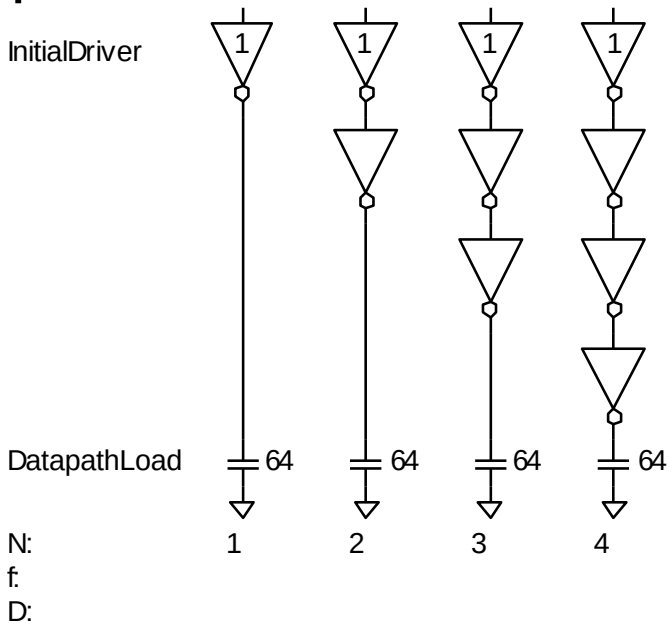
  x = (15*2) * (5/3) / 5 = 10



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

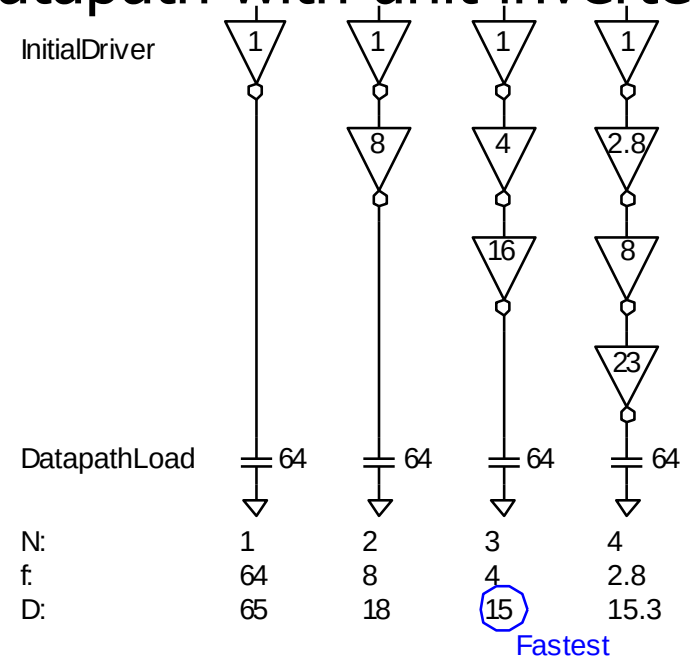
D =



Best Number of Stages

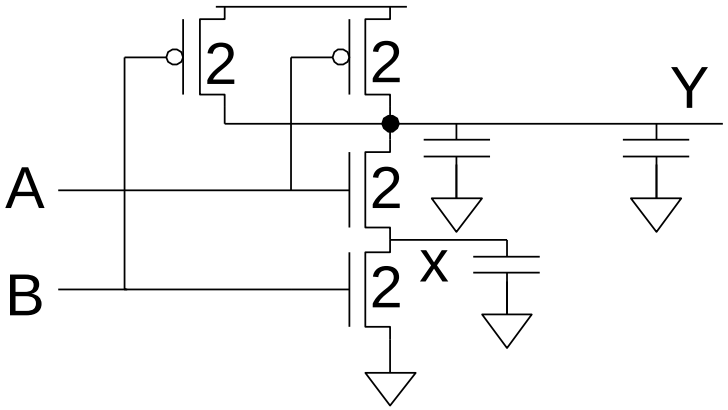
- How many stages should a path use?
  - Minimizing number of stages is not always fastest
- Example: drive 64-bit datapath with unit inverter

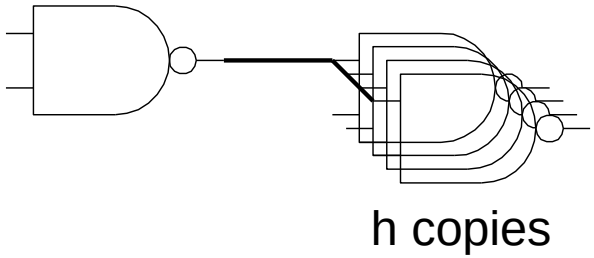
$$D = NF^{1/N} + P$$
$$= N(64)^{1/N} + N$$



Example: 2-input NAND

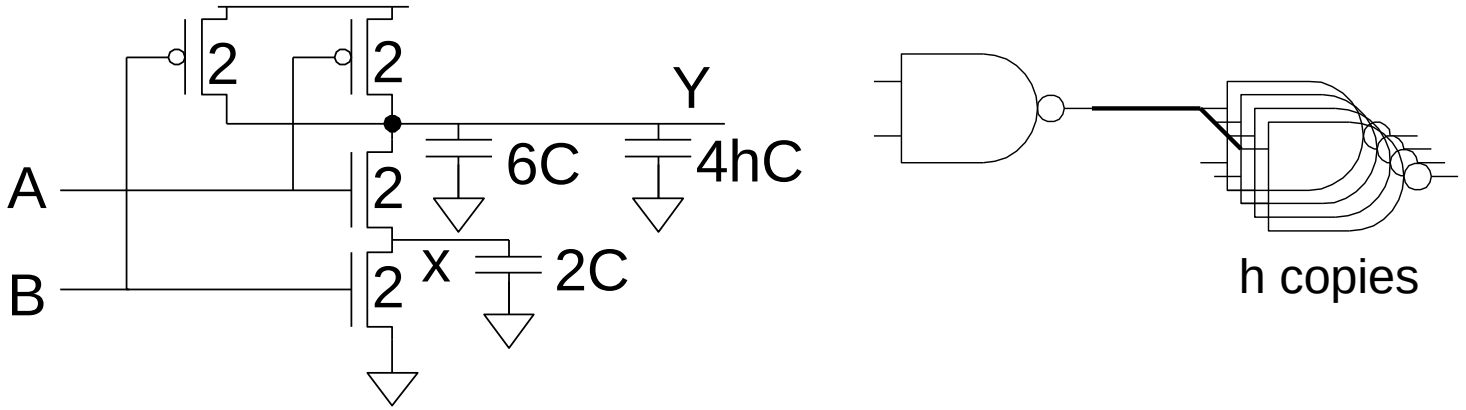
Estimate worst-case rising and falling delay of 2-input NAND driving *h* identical gates.





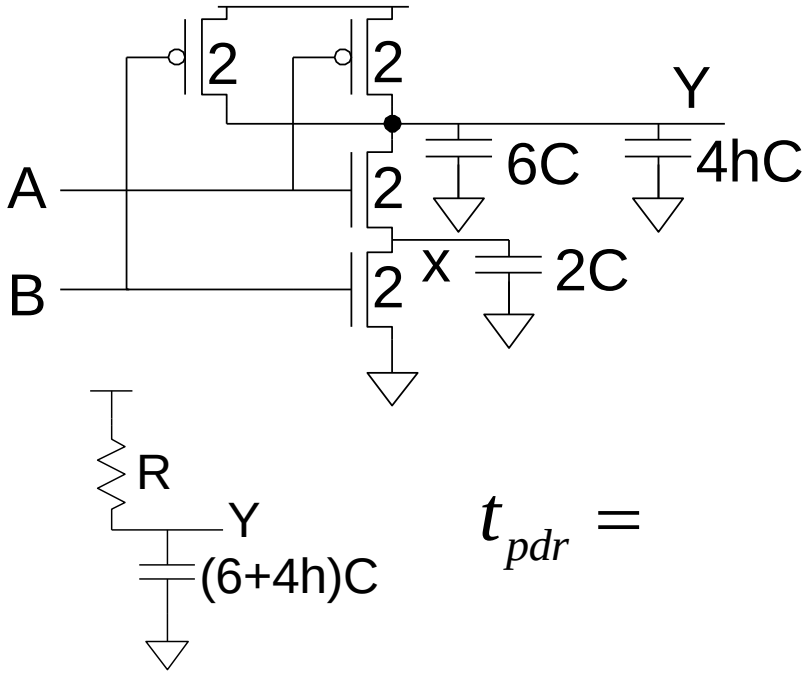
Example: 2-input NAND

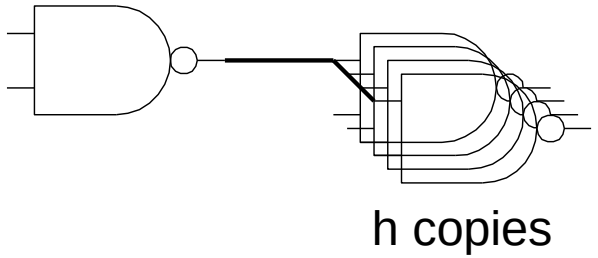
- Estimate rising and falling propagation delays of a 2-input NAND driving *h* identical gates.



Example: 2-input NAND

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

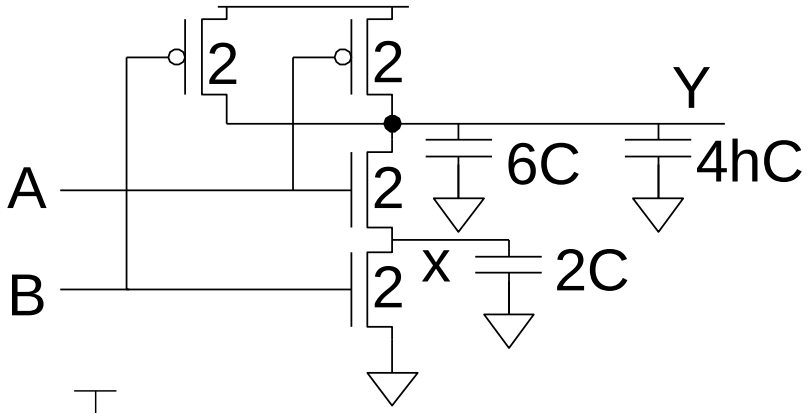


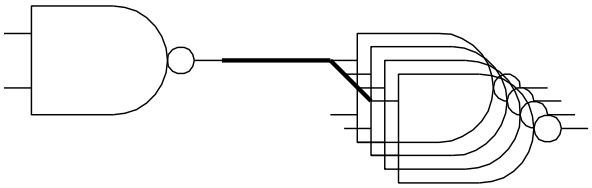


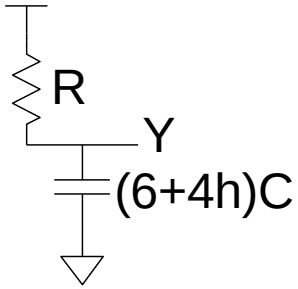
$$t_{pdr} =$$

Example: 2-input NAND

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



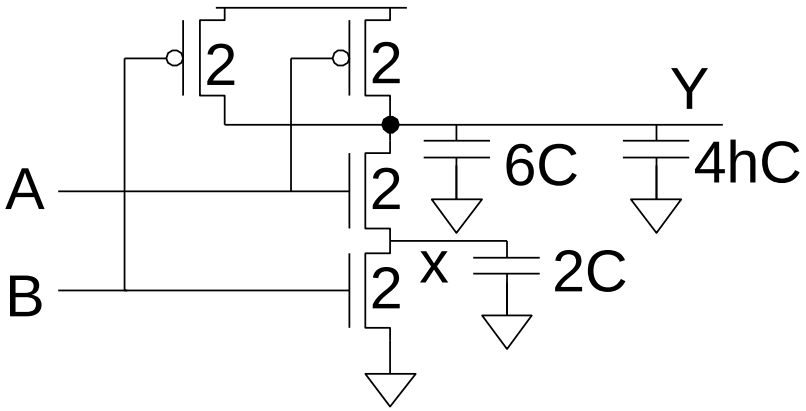


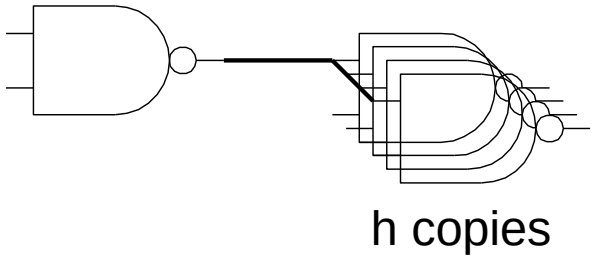


$$t_{pdr} = (6+4h)RC$$

Example: 2-input NAND

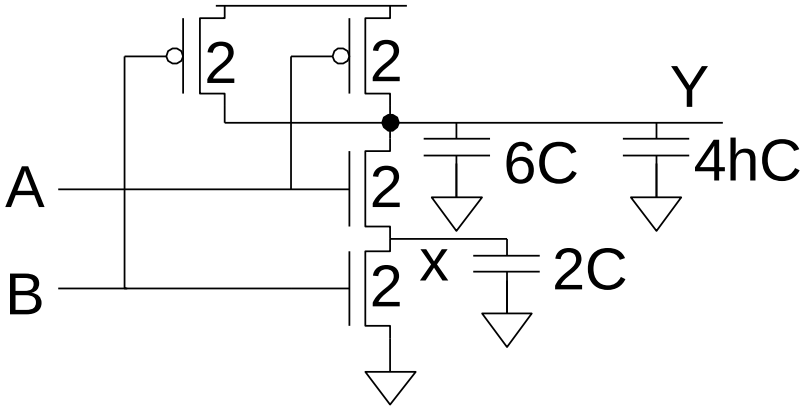
- Estimate rising and falling propagation delays of a 2-input NAND driving *h* identical gates.

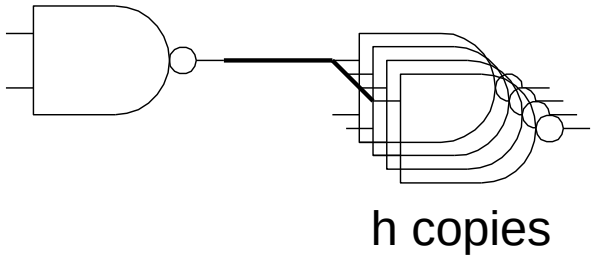


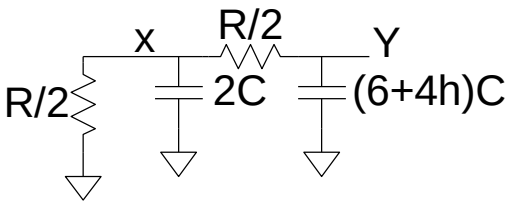


Example: 2-input NAND

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



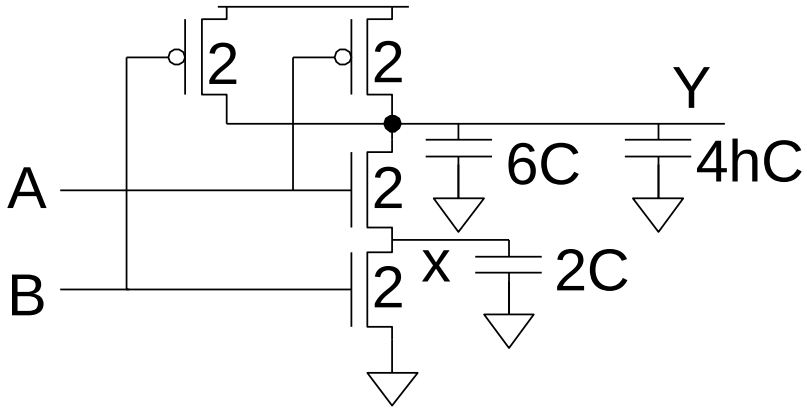


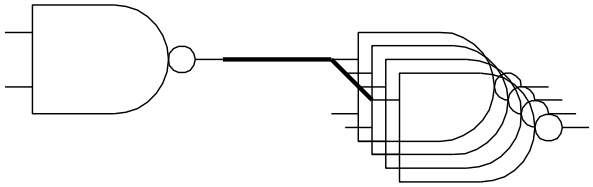


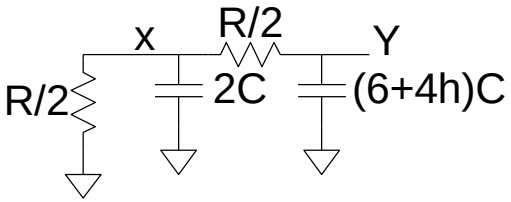
$$t_{pdf} =$$

Example: 2-input NAND

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



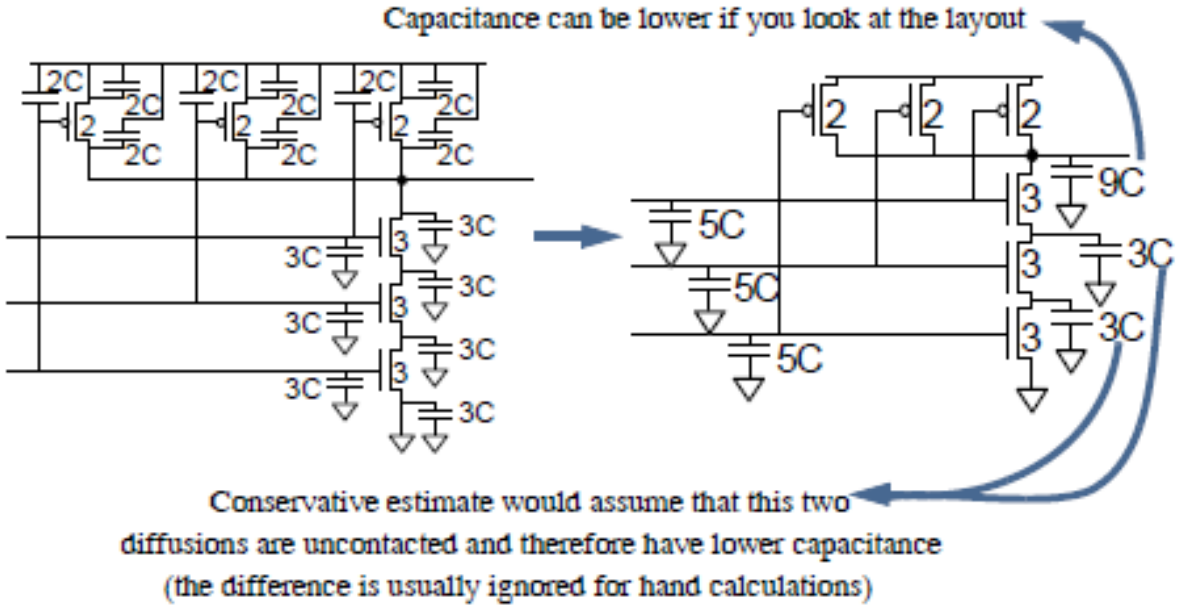




$$t_{pdf} = (2C)\left(\frac{R}{2}\right) + \left[(6+4h)C\right]\left(\frac{R}{2}+\frac{R}{2}\right)$$
$$= (7+4h)RC$$

Delay Estimation (cont.)

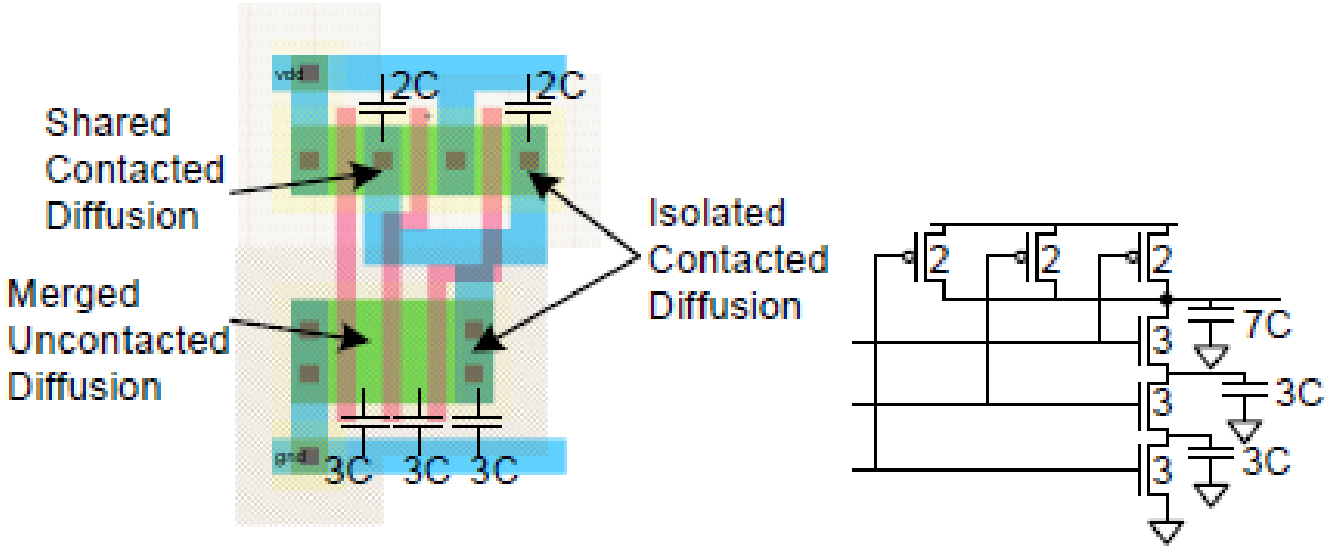
- 3 input NAND gate with it's gate and diffusion capacitances (assuming all nodes are contacted). Estimation at schematic levels will be different if you look at the layouts.



Layout models

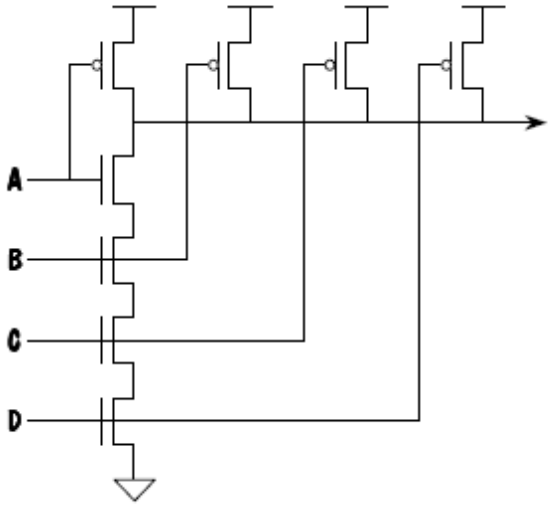
Good layout minimizes the diffusion area.

3 input NAND shown here, shares one diffusion contact, thus lowering the output capacitance by 2C. Contact diffusions are assumed.



Body effect & delay

- If A goes from 0 to 1 while B, C and D are 1, then all the intermediate nodes in the pulldown chain have already been discharged and the top MOSFET sees only a small body effect.
- If D goes from 0 to 1 while A, B and C are 1, then the intermediate nodes are all one V_t below V_{dd} and the upper MOSFETs see a larger body effect.



Circuit characterization & performance

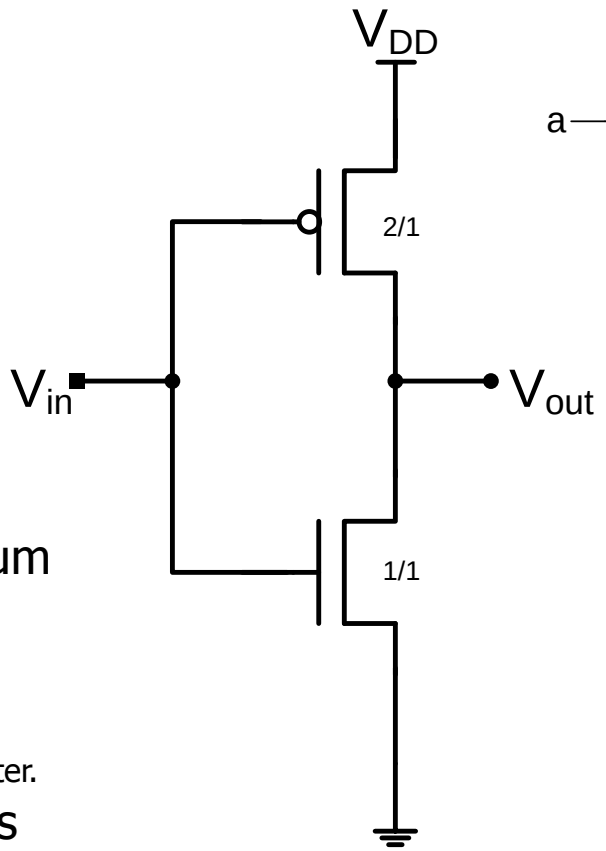
- Delay estimation
  - Simple RC model
  - Penfield-Rubenstein Model
- Delay minimization techniques
  - Transistor sizing
  - Wiring sizing
  - Distributed drivers
  - Driving large loads
- Wiring techniques

Delay minimization techniques

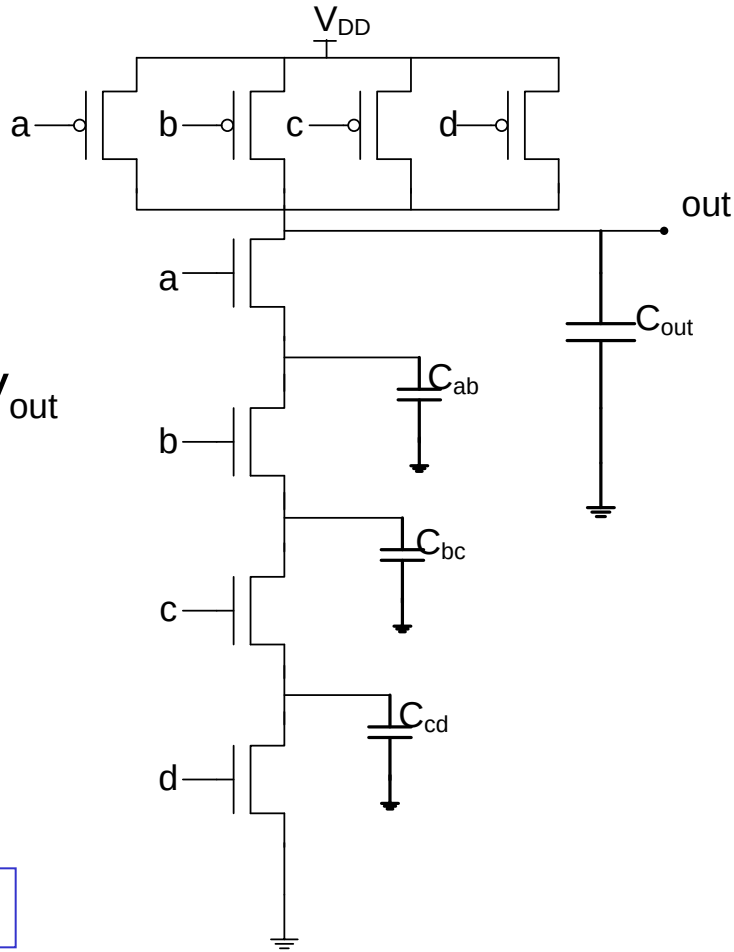
Transistor sizing

- Transistor sizing for minimum delay
- Each Series transistors has

  $(W/L) = 3(W/L)_{basic\ inverter.}$
- each parallel transistors has $(W/L) = (W/L)_{basic\ inverter}$



Basic inverter

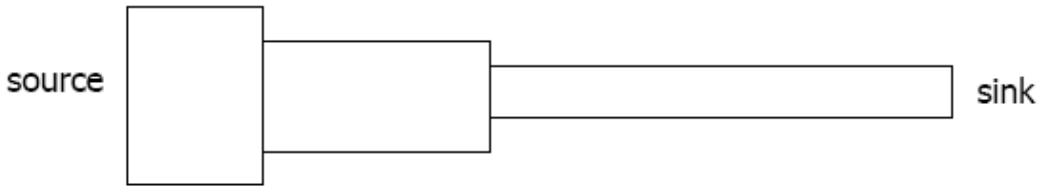


Wire sizing

- Wire length is determined by layout architecture, but we can choose wire width to minimize delay.
- width can vary with distance from driver to adjust the resistance which drives downstream capacitance.

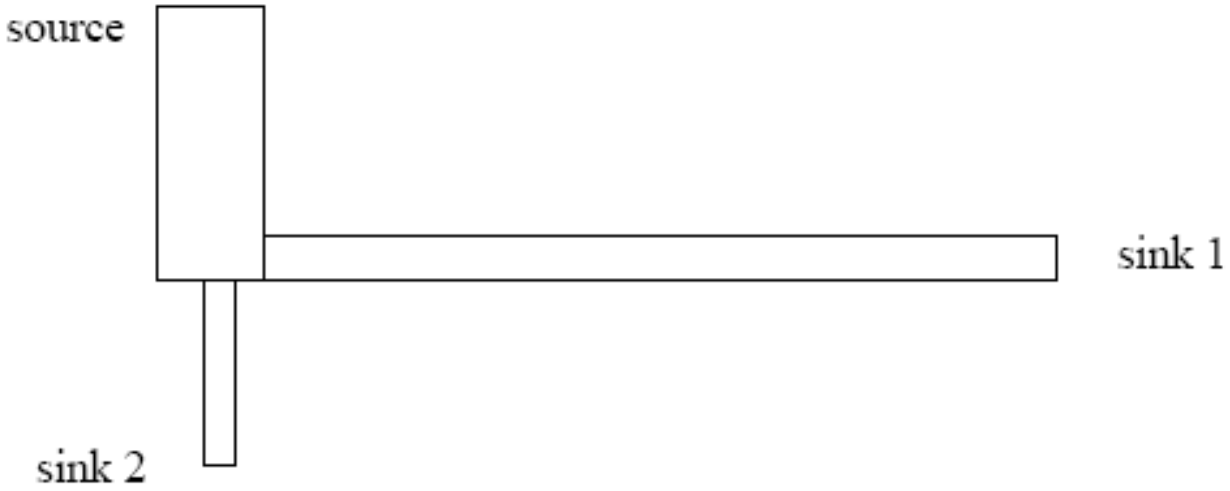
Optimal wire sizing

- Widening the wire reduces the resistance, but increases its capacitance.
- Wire with minimum delay has an exponential taper.
- Optimal tapering improves delay by about 8%.
- Can approximate optimal tapering with a few rectangular segments.



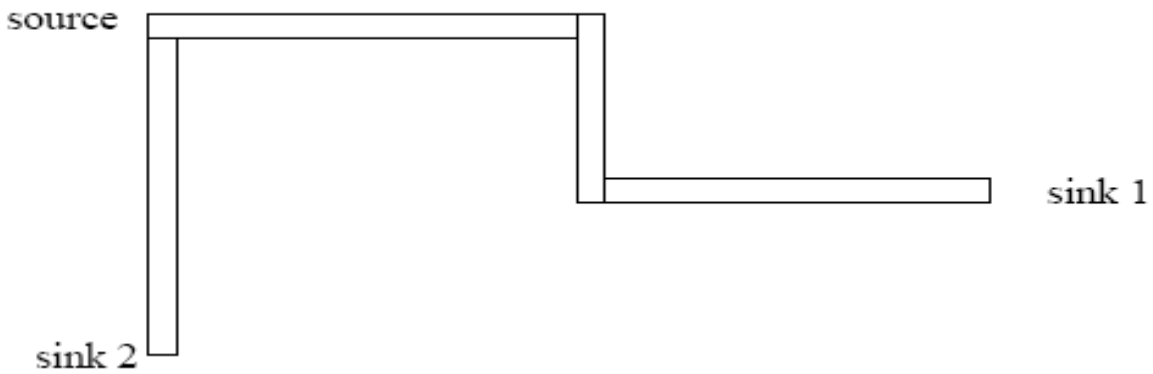
Tapering of wiring trees

- Different branches of tree can be set to different lengths to optimize delay.



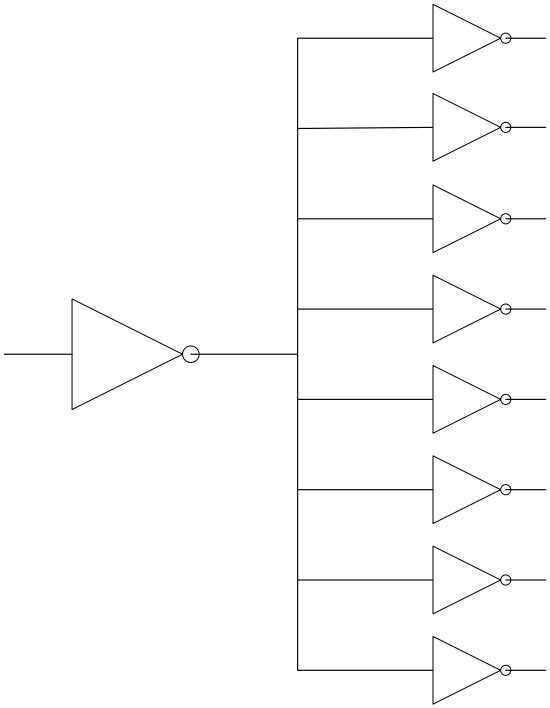
Spanning tree

- A spanning tree has segments that go directly between sources and sinks.

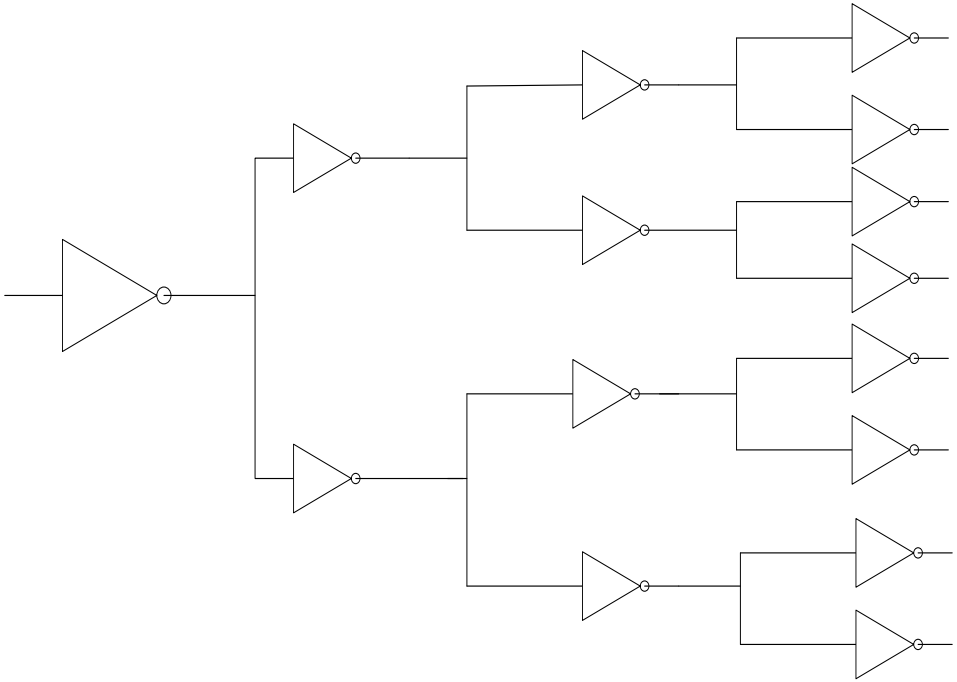


Distributed driver

- Decrease output load capacitance



Undistributed : $C_{load} = 8C_g$

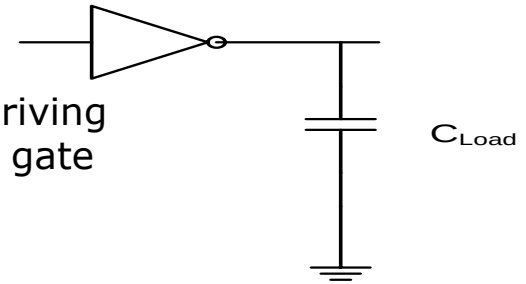


Distributed: $C_{load} = 6C_g$

Driving large loads

- If large loads have to be driven, the delay may increase drastically. Large loads are output capacitances, clock trees, etc.

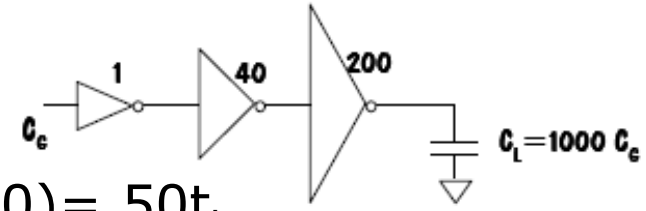
$$t_d = t_{inv}\,(C_L/C_G)$$



Where t_{inv}=is the average delay of a minimum-sized inverter driving another minimum sized inverter, C_L = load capacitance& C_G = gate capacitance

$$C_L = 1000C_G,\ \text{then}\ t_d = 1000.t_{inv}$$

A possibility to reduce the delay, is to use a sequence of n scaled inverters, but not the optimum delay:



$$t_d = t_{inv}\,(40/1) + t_{inv}\,(200/40) + t_{inv}\,(1000/200) = 50t_{inv}$$

Driving large loads (cont.)

- We may decrease the delay by using larger transistors to decrease the resistance.
  - Scaling transistors by factor S results in:

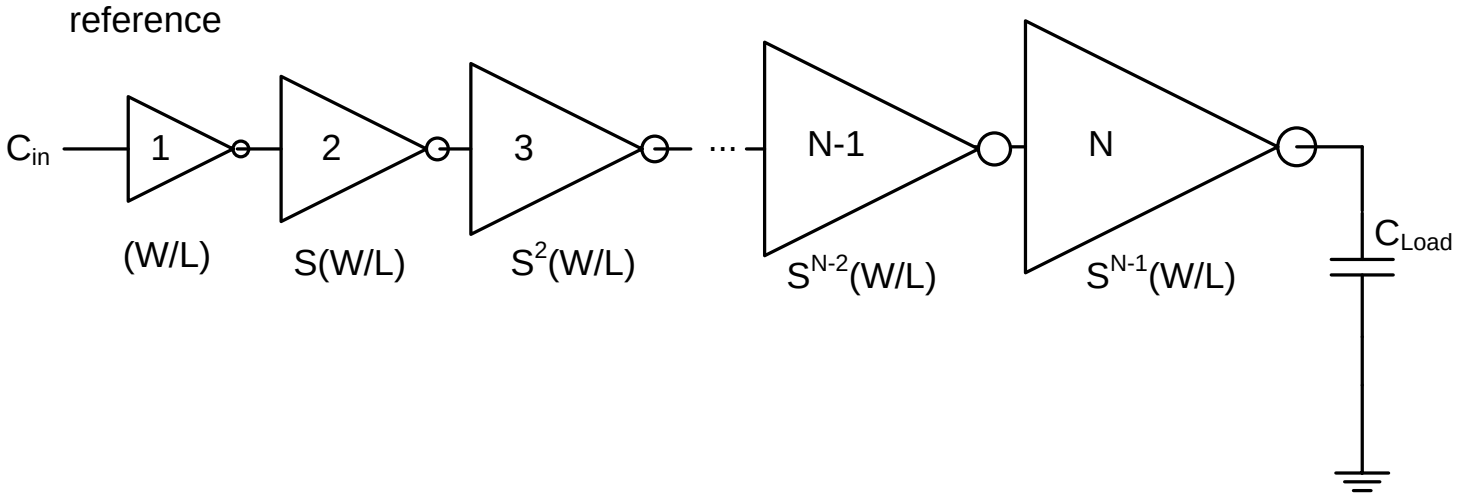
$$\left(\frac{W}{L}\right)_{scaled} = S\left(\frac{W}{L}\right)_{normal}$$

$$R_{scaled} = \frac{R_{scaled}}{S}$$

$$t_r = 2.2R_{scaled}\,(C_{in} + C_{load})$$

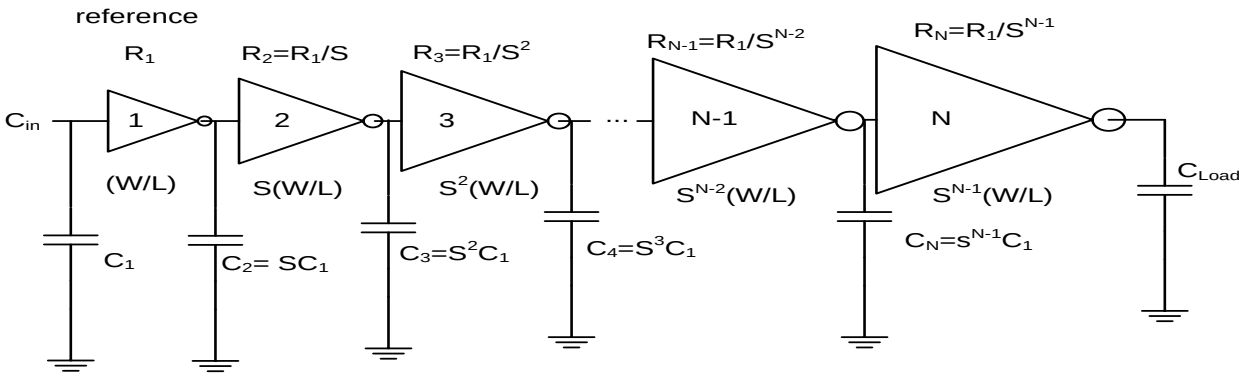
- But scaling the transistor also affects the input capacitance of the transistor:
  - $C_{in,scaled} = S \cdot C_{in}$

Driving large loads (cont.)



- The problem is if input signal is placed at inverter 1 what's the number of stages N and the scaling factor S that will minimize the time needed for the signal to reach C_{load}

Driving large loads (cont.)



- $t_d = t_1+t_2+t_3+\ldots+t_N$

$t_d = R_1C_2+ R_2C_3 + R_3C_4 +\ldots+R_{N-1}C_N+R_NC_{load}$

$C_{load} = S^N C_1$

$$t_d = R_1SC_1 + \frac{R_1}{S}S^2C_1 + \frac{R_1}{S^2}S^3C_1 + \frac{R_1}{S^3}S^4C_1 + \ldots + \frac{R_1}{S^{N-2}}S^{N-1}C_1 + \frac{R_1}{S^{N-1}}S^NC_1$$

$$t_d = N(SR_1C_1)$$

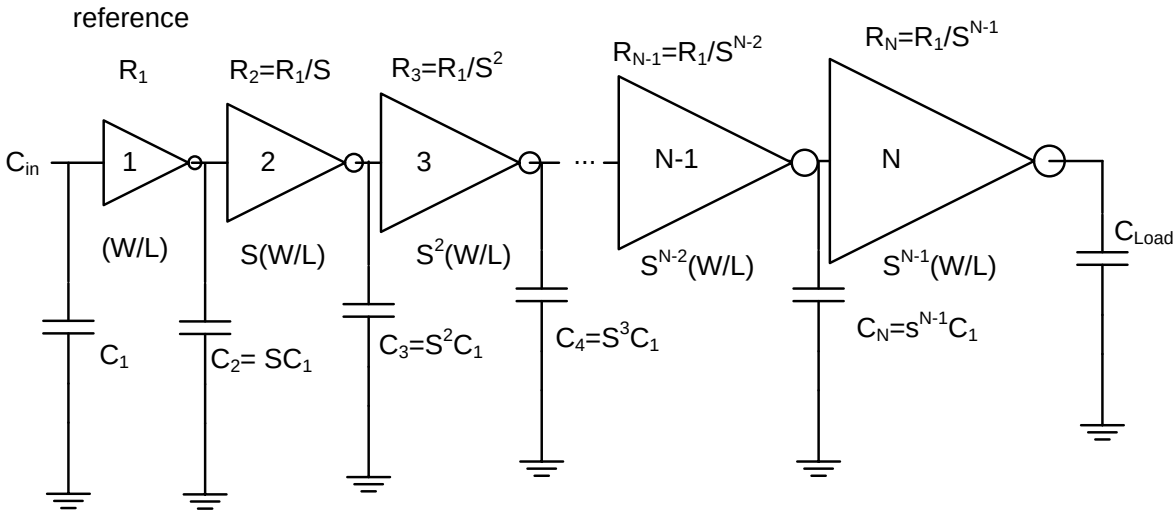
Driving large loads (cont.)

$$t_d = N(SR_1C_1)$$

$$C_{Load} = S^N C_1$$

$$N = \frac{\ln\left(\frac{C_{Load}}{C_1}\right)}{\ln(S)}$$

$$t_d = \frac{\ln\left(\frac{C_{Load}}{C_1}\right)}{\ln(S)}(SR_1C_1)$$



For minimum delay, $dt_d/dS = 0$; this yields,

$S = e = 2.71$

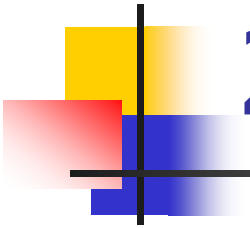
Example

we want to drive a load capacitor of value C_L = 10pF. The input stage is defined as C_1=20fF and has k_1=200μA/V^2. calculate the number of stages N to minimize the delay.

solution

$$N = \frac{\ln \frac{C_{Load}}{C_1}}{\ln(S)} = \frac{\ln(\frac{10x10^{-12}}{20x10^{-15}})}{\ln(e)} = \ln(500) = 6.21$$

We will select N = 6 to obtain non-inverting chain



2nd assignment

- Check the web site for Assignment 3.
- Due date next Saturday.

Refs.

- David Harris, Logical effort lecture notes, Hurvey mid college, 2004.
- CMOS VLSI Design, 4th edition
- Introduction to VLSI, 2nd edition