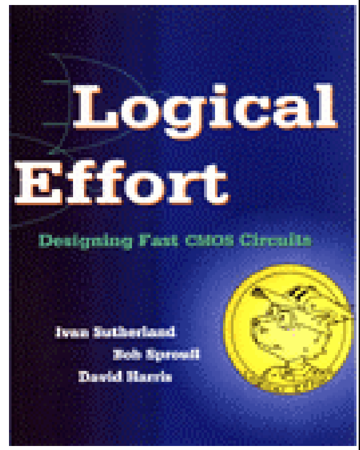
Estimating Delays

- Would be nice to have a "back of the envelope" method for sizing gates for speed
- Logical Effort
  - Book by Sutherland, Sproull, Harris
  - Chapter 1 is on our web page



Gate Delay Model

- First, normalize a model of delay to dimensionless units to isolate fabrication effects
  - $d_{abs} = d\ \tau$
  - τ is the delay of a minimum inverter driving another minimum inverter with no parasitics
  - In a 0.6u process, this is approx 40ps
  - Now we can think about delay in terms of d and scale it to whatever process we're building the circuit in

Gate Delay

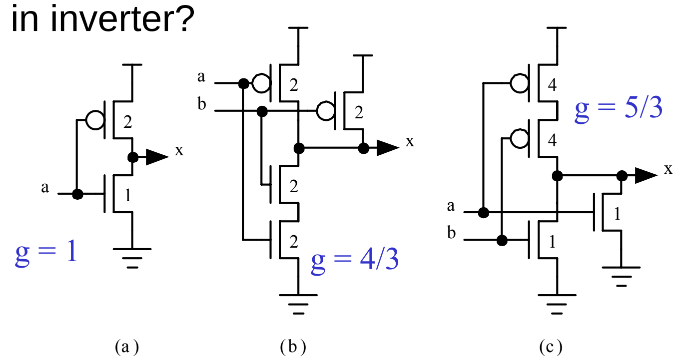
- Delay of a gate d has two components
  - A fixed part called *parasitic delay* p
  - A part proportional to the load on the output called the *effort delay* or *stage effort* f
  - Total delay is measured in units of τ, and is sum of these delays
  - $d = f + p$

Effort Delay

- The effort delay (due to load) can be further broken down into two terms
  - $f = g * h$
  - g = logical effort which captures properties of the gate's structure
  - h = electrical effort which captures properties of load and transistor sizes
    - $h = C_{out}/C_{in}$
    - C_{out} is capacitance that loads the output
    - C_{in} is capacitance presented at the input
  - So, $d = gh + p$

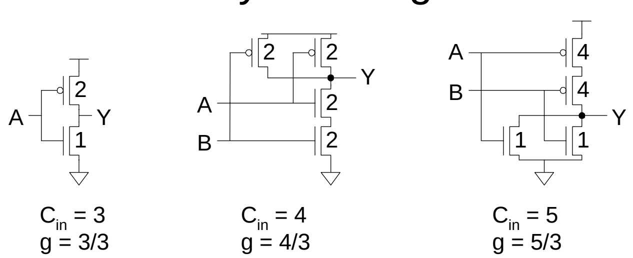
Logical Effort

- Logical effort normalizes the output drive capability of a gate to match a unit inverter
  - How much more input capacitance does a gate need to present to offer the same drive as in inverter?



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



Logical Effort of Other Gates

- Logical effort of common gates assuming that P/N size ratio is 2

Number of inputs

Gate Type	***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
MUX		2	2	2	2	2
XOR		4	12	32		

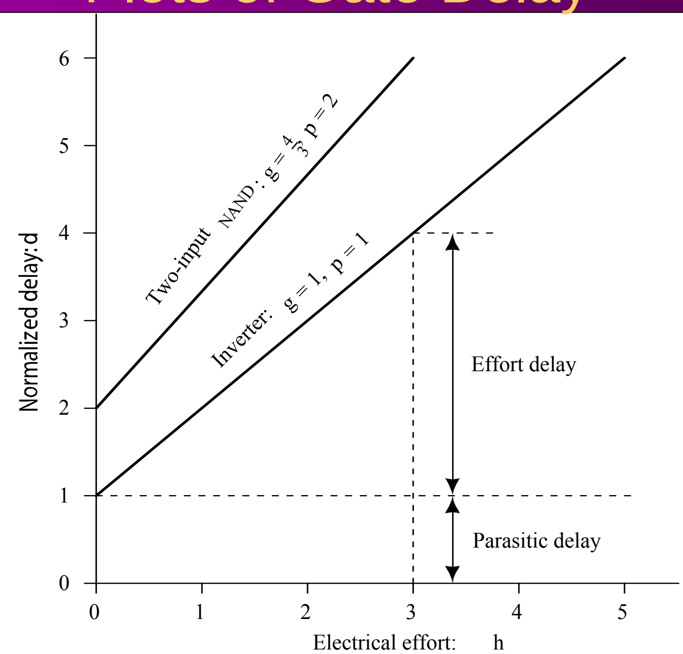
Electrical Effort

- Value of logical effort g is independent of transistor size
  - It's related to the ratios and the topology
- Electrical effort h captures the drive capability of the transistors via sizing
  - Electrical effort $h = C_{out}/C_{in}$
  - Note that as transistor sizes for a gate increase, h decreases because C_{in} goes up

Parasitic Delay

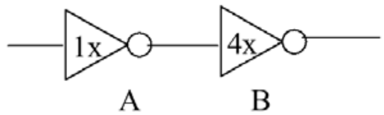
- Parasitic delay p is caused by the internal capacitance of the gate
  - It's constant and independent of transistor size
  - As you increase the transistor size, you also increase the cap of the gate/source/drain areas which keeps it constant
  - For our purposes, normalize p_{inv} to 1
    - N-input NAND = $n*p_{inv}$
    - N-input NOR = $n*p_{inv}$
    - N-way mux = $2n*p_{inv}$
    - XOR = $4*p_{inv}$

Plots of Gate Delay



Delay Estimation

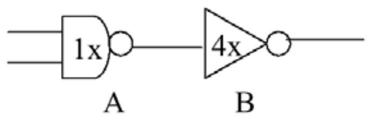
Delay Estimation



Remember, τ in
Our process ~ 40ps

A_delay = g*h + p = 1*(CinB/CinA) + 1
= 1*(4*CinA/CinA) + 1 = 4 + 1 = 5 time units

~200ps

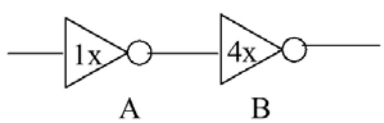


A_delay = g*h + p = (4/3)*(CinB/CinA) + 2*1
Cin_B = 4*3 = 12. Cin_A = 4
A_delay = (4/3)*(12/4) + 2 = 4 + 2 = 6 units ~240ps

Nand2 worse because of higher parasitic delay than inverter.
Note that g*h term was same for both because NAND2 sized to provide same current drive.

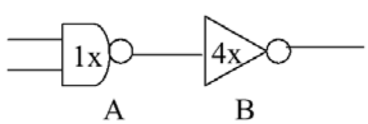
Delay Estimation

Delay Estimation



Remember, τ in
Our process ~ 40ps

A_delay = g*h + p = 1*(CinB/CinA) + 1 ~200ps
= 1*(4*CinA/CinA) + 1 = 4 + 1 = 5 time units



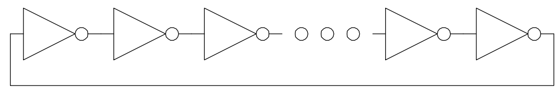
τ in 180nm = ~ 12ps
FO4 Inverter delay = 60ps
FO4 NAND delay = 72ps

A_delay = g*h + p = (4/3)*(CinB/CinA) + 2*1
Cin_B = 4*3 = 12. Cin_A = 4
A_delay = (4/3)*(12/4) + 2 = 4 + 2 = 6 units ~240ps

Nand2 worse because of higher parasitic delay than inverter.
Note that g*h term was same for both because NAND2 sized to provide same current drive.

Example: Ring Oscillator

- Estimate the frequency of an N-stage ring oscillator



Logical Effort: g =

Electrical Effort: h =

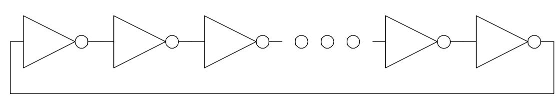
Parasitic Delay: p =

Stage Delay: d =

Period of osc =

Example: Ring Oscillator

- Estimate the frequency of an N-stage ring oscillator



Logical Effort: g = 1

Electrical Effort: h = 1

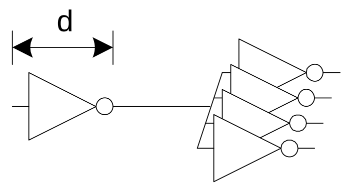
Parasitic Delay: p = 1

Stage Delay: d = 2 so d_{abs} = 80ps

Period: 2*N*d_{abs} = 4.96ns, Freq = ~200MHz

Example: FO4 Inverter

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



Logical Effort: $g =$

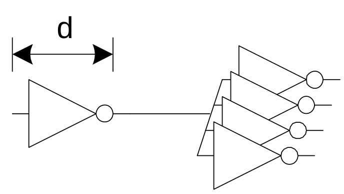
Electrical Effort: $h =$

Parasitic Delay: $p =$

Stage Delay: $d =$

Example: FO4 Inverter

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



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

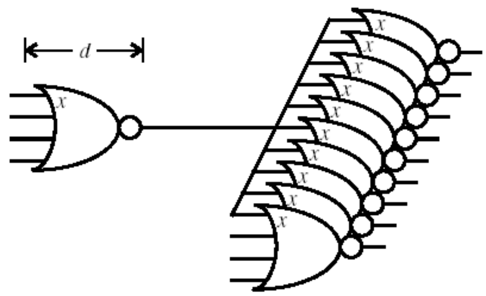
Logical Effort: $g = 1$

Electrical Effort: $h = 4$

Parasitic Delay: $p = 1$

Stage Delay: $d = gh + p = 5$

Delay Estimation

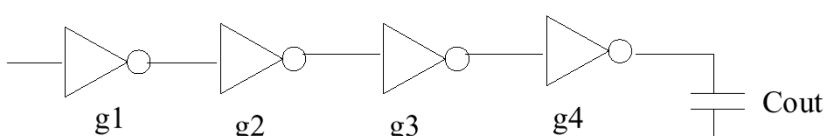


- If Cin = x, Cout = 10x, thus h = 10
- g = 9/3 = 3
- d = gh + p = 3*10 + 4*1 = 34 (1360 ps)

Multi Stage Delay

MultiStage Delay

- Recall rule of thumb that said to balance the delay at each stage along a critical path
- Concepts of logical effort and electrical effort can be generalized to multistage paths



Path logical effort = g1*g2*g3 *g4

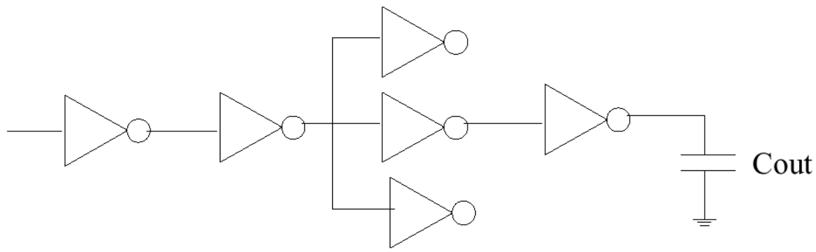
In general, Path logic effort $G = \Pi\, g(i)$

Path electrical effort $H = Cout / Cin_{first\_gate}$

Must remember that electrical effort only is concerned with effect of logic network on input drivers and output load.

Off-Path Load

Off Path Load



Off path load will divert electrical effort from the main path, must account for this. Define a *branching effort b* as:

$$b = (C_{on\_path} + C_{off\_path}) / C_{on\_path} = \frac{C_{total}}{C_{useful}}$$

The branching effort will modify the electrical effort needed at that stage. The branch effort *B* of the path is:

$$B = \Pi\, b(i)$$

Summary – multistage 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?

Branching Effort

- Remember *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

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

This is a key result of logical effort

- Find fastest possible delay
- Doesn't require calculating gate sizes

Minimizing Path Delay

The absolute delay will have the parasitic delays of each stage summed together.

However, can *focus on just Path effort F* for minimization purposes since parasitic delays are constant.

For an N-stage network, *the path delay is least when each stage in the path bears the same stage effort.*

$$f(\min) = g(i) * h(i) = F^{1/N}$$

Minimum achievable path delay

$$D(\min) = N * F^{1/N} + P$$

Note that if N=1, then $d = f + p$, the original single gate equation.

Choosing Transistor Sizes

Remember that the stage effort h(i) is related to transistor sizes.

$f(\min) = g(i) * h(i) = F^{1/N}$

So

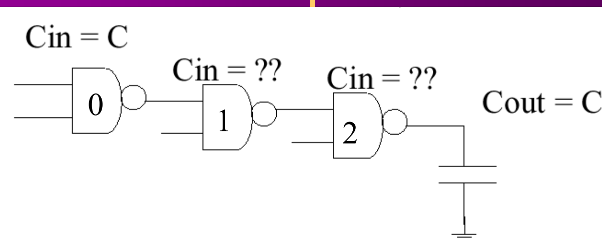
$h(i)\ \min = F^{1/N} / g(i)$

To size transistors, start at end of path, and compute:

$Cin(i) = gi * Cout(i) / f(\min)$

Once Cin(i) is know, can distribute this among transistors of that stage.

Example



Size the transistors of the nand2 gates for the three stages shown.

Path logic effort = G = g0 * g1 * g2 = 4/3 * 4/3 * 4/3 = 2.37

Branching effort B = 1.0 (no off-path load)

Electrical effort H = Cout/Cin = C/C = 1.0

Min delay achievable $= 3*(G*B*H)^{1/3} + 3(2*pinv)$

$= 3*(2.37*1*1)^{1/3} + 3(2*1.0) = 10.0$

$minD = N*F^{1/N} + P$

Example, continued

The effort of each stage will be:

$$f\,min = (G*B*H)^{1/3} = (2.37*1.0*1.0)^{1/3} = 1.33 = 4/3$$

Cin of last gate should equal:

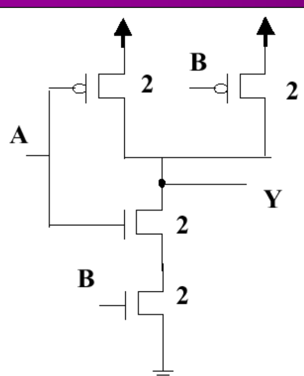
$$\text{Cin last gate (min)} = gi * \text{Cout}(i) \; / \; f(min)$$
$$= 4/3 * C / (4/3) = C$$

Cin of middle gate should equal:

$$\text{Cin middle gate} = gi * \text{Cin last gate} / f(min)$$
$$= 4/3 * C / (4/3) = C$$

All gates have same input capacitance, distribute it among transistors.

Transistor Sizes for Example



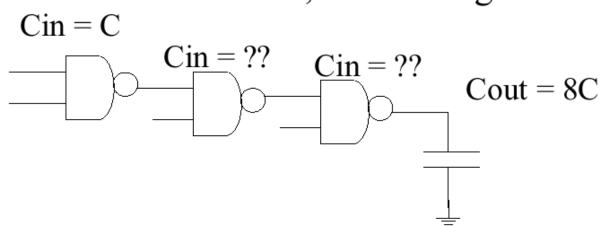
Where gate capacitance of

2 *W *L Mosfet = C/2

Choose W accordingly.

Another Example, Larger Load

Let Load = 8C, what changes?



Size the transistors of the nand2 gates for the three stages shown.

Path logic effort = G = g0 * g1 * g2 = 4/3 * 4/3 * 4/3 = 2.37

Branching effort B = 1.0 (no off-path load)

Electrical effort H = Cout/Cin = **8**C/C = **8.0**

Min delay achievable $= 3^* (G^*B^*H)^{1/3} + 3\ (2^*pinv)$

$= 3\ {}^*(2.37^*1^*\mathbf{8})^{1/3} + 3\ (2^*1.0) = \mathbf{14.0}$

8C Load Example Cont.

The effort of each stage will be:

$f\ min = (G^*B^*H)^{1/3} = (2.37^*1.0^*\mathbf{8})^{1/3} = \mathbf{2.67} = \mathbf{8/3}$

Cin of last gate should equal:

Cin last gate (min) = gi * Cout (i) / f(min)

= 4/3 * **8C** / (**8/3**) = 4C

Cin of middle gate should equal:

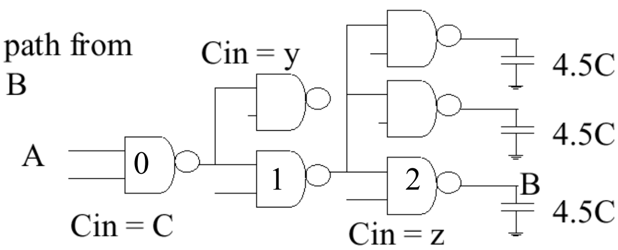
Cin middle gate = gi * Cin last gate / f(min)

= 4/3 * **4C**/ (**8/3**) = 2C

Note that each stage gets progressively larger, as is typical with a multi-stage path driving a large load.

Example 1.6 from Chap 1

Size path from
A to B



Path logic effort G = g0 * g1 * g2 = 4/3 * 4/3 * 4/3 = 2.37
Branch effort, 1st stage = (y+y)/y = 2.
Branch effort, 2nd stage = (z+z+z)/z = 3
Path Branch effort B = 2 * 3 = 6.
Path electrical effort H = Cout/Cin = 4.5C/C = 4.5
Path stage effort = F = G*B*H = 2.37*6*4.5 = 64.

Min delay = $N(F)^{1/N} + P = 3*(64)^{1/3} + 3(2pinv)$ = 18.0 units

Example 1.6 Continued

Stage effort of each stage should be:
$f(min) = (F)^{1/N} = (GBH)^{1/N} = (64)^{1/3} = 4$

Determine Cin of last stage:

Cin(z) = g * Cout / f(min) = 4/3 * 4.5C / 4 = 1.5 C

Determine Cin of middle stage:

Cin(y) = g * (3*Cin(z))/ f(min) = 4/3 * (3*1.5C) / 4 = 1.5C

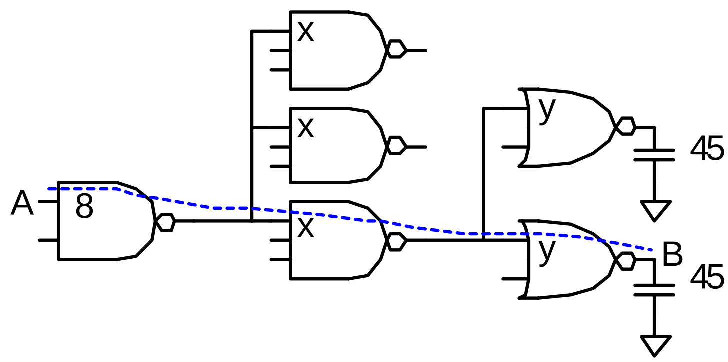
Is first stage correct?

Cin(A) = g * (2*Cin(y))/f(min) = 4/3 *(2*1.5C)/4 = C.

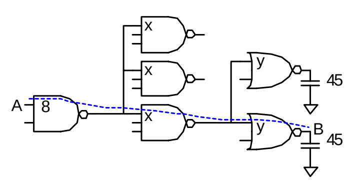
Yes, self-consistent.

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 =

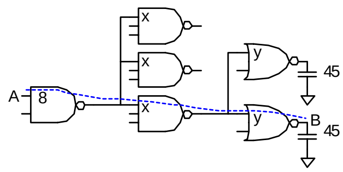
Path Effort F =

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 $P = 2 + 3 + 2 = 7$

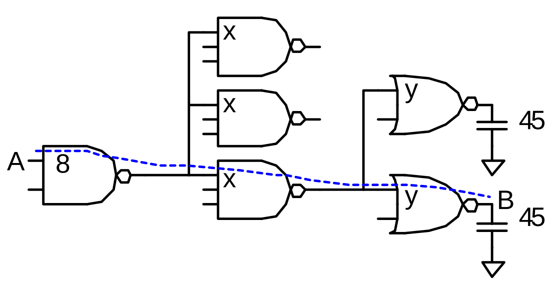
Delay $D = 3*5 + 7 = 22 = 4.4$ FO4

Example: 3-stage path

- Work backward for sizes

  y =

  x =

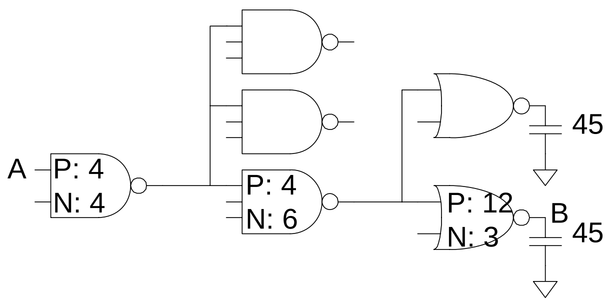


Example: 3-stage path

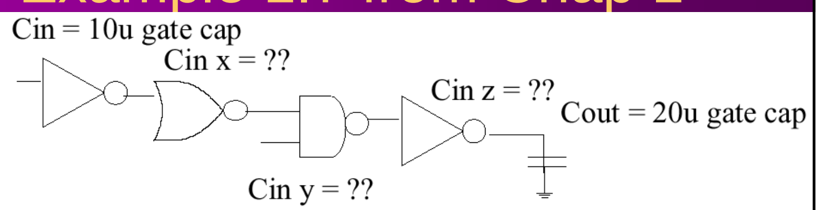
- Work backward for sizes

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

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



Example 1.7 from Chap 1



Path logic effort G = g0 * g1 * g2 * g3 = 1*5/3 * 4/3 * 1 = 20/9
Path Branch effort B = 1
Path electrical effort H = Cout/Cin = 20/10 = 2
Path stage effort = F = G*B*H = (20/9)*1*2 = 40/9

For Min delay, each stage has effort $(F)^{1/N} = (40/9)^{1/4} = 1.45$

z = g * Cout/f(min) = 1*20/1.45 = 14
y = g * Cin(z)/f(min) = 4/3 * 14 / 1.45 = 13
x = g * Cin(y)/f(min) = 5/3 * 13 / 1.45 = 15

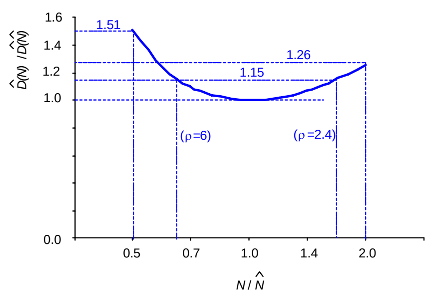
Note: Don't care about parasitics for gate sizing, only if you want to know absolute delay…

Misc. Comments

- Note that you never size the first gate
  - This gate is assumed to be fixed
  - If you were allowed to size it, the algorithm would try to make it as large as possible
- This is an estimation algorithm
  - Authors claim that sizing a gate by 1.5x too big or small still results in a path delay within 15% of minimum

Sensitivity Analysis

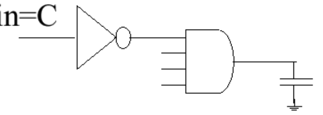
- How sensitive is delay to using exactly the best number of stages?



- 2.4 < ρ < 6 gives delay within 15% of optimal
  - We can be sloppy!
  - I like ρ = 4

Evaluating Different Options

Cin=C

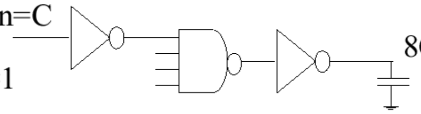


8C

The problem

Cin=C

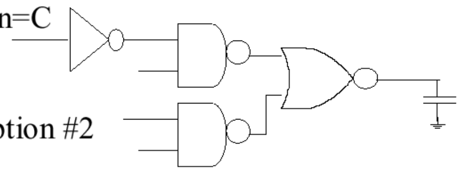
Option #1



8C

Cin=C

Option #2

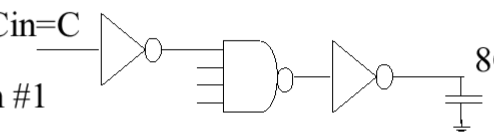


8C

Option #1

Cin=C

Option #1

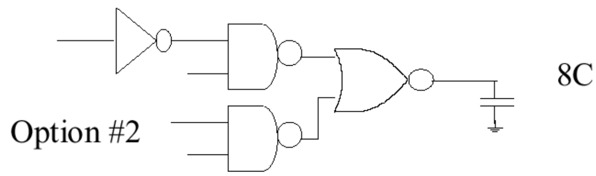


8C

Path logic effort G = g0 * g1 * g2 = 1*6/3 * 1 = 2
Path Branch effort B = 1
Path electrical effort H = Cout/Cin = 8C/C = 8
Path stage effort = F = G*B*H = 2*1*8 = 16

Min delay: $= N^* (F)^{1/N} + P$
$= 3 *(16)^{1/3} + (pinv + 4*pinv + pinv)$
$= 3 *(2.5) + 6 = 13.5$

Option #2



Path logic effort G = g0 * g1 * g2 = 1*4/3 * 5/3 = 20/9
Path Branch effort B = 1
Path electrical effort H = Cout/Cin = 8C/C = 8
Path stage effort = F = G*B*H = 20/9*1*8 = 160/9

Min delay: $= N^* (F)^{1/N} + P$
$= 3 *(160/9)^{1/3} + (pinv + 2*pinv + 2*pinv)$
$= 3 * 2.6 + 5 = 12.8$

Option #2 appears to be better than Option #1, by a slight margin.

How many stages?

- Consider three alternatives for driving a load 25 times the input capacitance
  - One inverter
  - Three inverters in series
  - Five inverters in series
- They all do the job, but which one is fastest?

How many stages?

- In all cases: G = 1, B = 1, and H = 25
- Path delay is $N(25)^{1/N} + N P_{inv}$
  - N = 1, D = 26 units
  - N = 3, D = 11.8 units
  - N = 5, D = 14.5 units
- Since N=3 is best, each stage will bear an effort of $(25)^{1/3} = 2.9$
  - So, each stage is ~3x larger than the last
  - In general, the best stage effort is between 3 and 4 (not e as often stated)
    - The e value doesn't use parasitics…

Choosing the Best # of Stages

- You can solve the delay equations to determine the number of stages N that will achieve the minimum delay
  - Approximate by $Log_4 F$

Path Effort F	***Best N***	***Min Delay D***	***Stage effort f***
0-5.83	1	1.0-6.8	0-5.8
5.83-22.3	2	6.8-11.4	2.4-4.7
22.3-82.2	3	11.4-16.0	2.8-4.4
82.2-300	4	16.0-20.7	3.0-4.2
300-1090	5	20.7-25.3	3.1-4.1
1090-3920	6	25.3-29.8	3.2-4.0

Example

- String of inverters driving an off-chip load
  - Pad cap and load = 40pf
  - Equivalent to 20,000 microns of gate cap
  - Assume first inverter in chain has 7.2u of input cap
  - How many stages in inv chain?
- H = 20,000/7.2 = 2777
- From the table, 6 stages is best
- Stage effort = $f = (2777)^{1/6} = 3.75$
- Path delay D = 6*3.75 +6*Pinv = 28.5
  - D = 1.14ns if τ = 40ps

Summary

- Compute path effort $F = GBH$
- Use table, or estimate $N = \log_4 F$ to decide on number of stages
- Estimate minimum possible delay $D = NF^{1/N} + \Sigma p_i$
- Add or remove stages in your logic to get close to N
- Compute effort at each stage $f = F^{1/N}$
- Starting at output, work backwards to compute transistor sizes $C_{in} = (g_i/f)C_{out}$

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 ~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