EN1600
Design and Implementation of VLSI Systems
Fall 2016

Lecture 10: Performance Optimization for Complex CMOS Gates

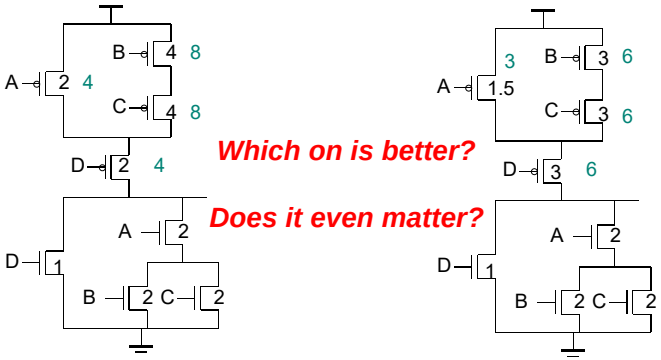
Reading: Chapter 4, sections 4.4-4.5 — Weste & Harris

October 12, 2016 — Prof. R. Iris Bahar

BROWN

Transistor sizing for a complex gate

OUT = !(D + A • (B + C))

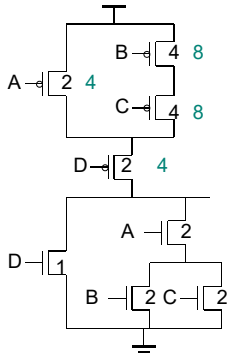


Which on is better?

Does it even matter?

Transistor sizing for a complex gate

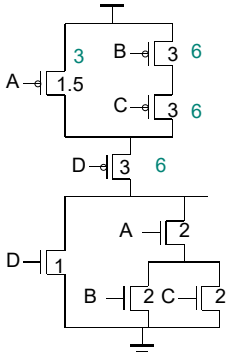
OUT = !(D + A • (B + C))



- Total Wp=24
- Worst case path resistance:
  - $Rp_{eff} = \beta(1/8 + 1/8 + 1/4)$
    $= 2 \times (1/2) = 1$
    (same as for an inverter)
- Shortest path resistance:
  - $Rp_{eff} = \beta(1/4 + 1/4)$
    $= 2 \times (1/2) = 1$
- Best case pull up resistance:
  - $Rp_{eff} = \beta\,[((1/8 + 1/8) \,||\, (1/4)) + 1/4]$
    $= \beta\,[1/8 + 1/4] = 3/8$
    (approx. more than twice as fast as worst case)

Transistor sizing for a complex gate

OUT = !(D + A • (B + C))



- Total Wp=21 (less area, intrinsic cap)
- Worst case path resistance:
  - $Rp_{eff} = \beta(1/6 + 1/6 + 1/6)$
    $= 2 \times (1/2) = 1$
    (same as for an inverter)
- Shortest path resistance:
  - $Rp_{eff} = \beta(1/3 + 1/6)$
    $= 2 \times (1/2) = 1$
- Best case pull up resistance:
  - $Rp_{eff} = \beta\,[((1/6 + 1/6) \,||\, (1/3)) + 1/6]$
    $= \beta\,[1/6 + 1/6] = 1/3$

(even better than shortest path first sizing!)

Creates larger disparity in delays as a function of inputs

BROWN

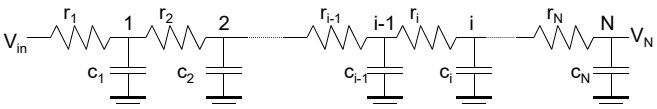
Homework #3

- Available on line later today
- Due Friday, October 21 by 5pm

- After today's lecture, you should be able to complete all but the last problem (on dynamic logic)
- Monday's lecture will finish covering dynamic logic

BROWN

Chain network Elmore delay



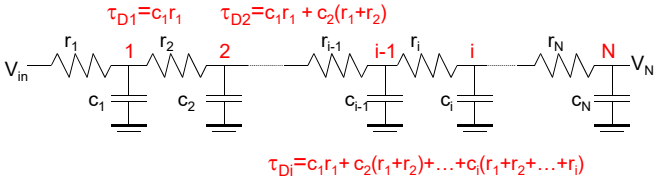
- A typical wire is a chain network with (simplified) Elmore delay of

$$\tau_{DN} = \sum c_i r_{ii} = \sum^{N} c_i \sum^{i} r_j$$

- Where $\sum^{i} r_j = r_1 + r_2 + \ldots + r_i$

BROWN

Chain Network Elmore Delay



Elmore delay equation $\tau_{DN} = \sum c_i r_{ii} = \sum^{N} c_i \sum^{i} r_j$

If all resistors are equal size,

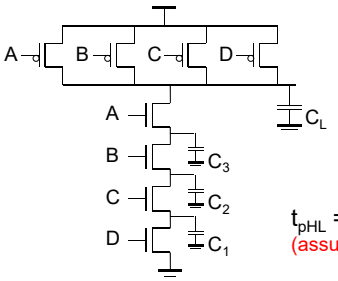
$$\tau_{Di}=c_1r_{eq}+ 2c_2r_{eq}+ 3c_3r_{eq}+\ldots+ ic_ir_{eq}$$

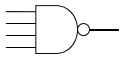
BROWN

Uses for the Elmore Delay Model

- Modeling the delay of a wire
- Modeling the delay of a series of pass transistors
- Modeling the delay of a pull-up and pull-down networks

Fanin considerations



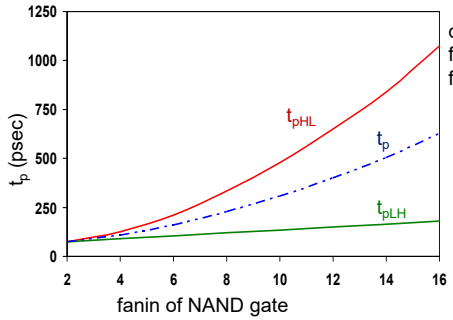


Distributed RC model
(Elmore delay)

$t_{pHL} = 0.69\ R_{eqn}(C_1+2C_2+3C_3+4C_L)$
(assuming all NMOS equally sized)

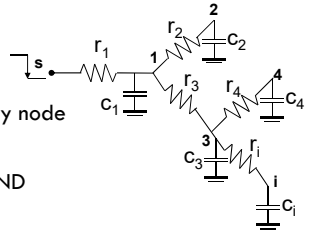
Propagation delay deteriorates rapidly as a function of fanin: quadratically in the worst case.

t_p as a function of fanin



- Gates with a fan-in greater than 4 should be avoided.

RC tree definitions



- RC tree characteristics
  - Unique path from source to any node
  - Single input (source), no loops
  - All caps have connection to GND
- Path resistance:

$$r_{ii} = \sum r_j \Rightarrow (r_j \in [\text{path}(s \rightarrow i)]$$

- Shared path resistance:

$$r_{ik} = \sum_{j=1}^{N} r_j \Rightarrow (r_j \in [\text{path}(s \rightarrow i) \cap \text{path}(s \rightarrow k)])$$

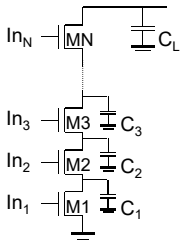
- A typical wire is a tree network with Elmore delay of:

$$\tau_{Di} = \sum_{k=1}^{N} c_k r_{ik}$$

Multiply each capacitance C_k by the shared resistance from source to k and source to i

Fast complex gates: techniques #0, #1

- Transistor sizing (i.e., scaling up all transistor in gate)
  - as long as fan-out capacitance dominates
- Progressive sizing



Distributed RC line

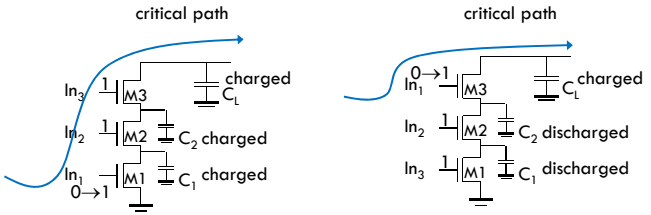
M1 > M2 > M3 > … > MN

(the FET closest to the *output* should be the smallest)

Can reduce delay by more than 20%; decreasing gains as technology shrinks

Fast complex gates: technique #2

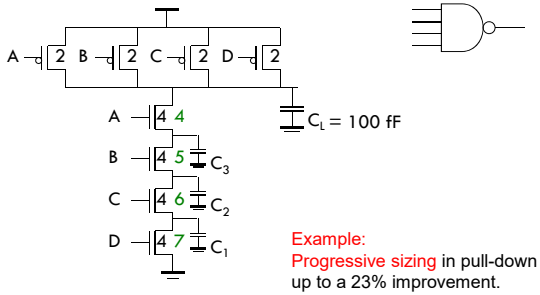
- Input re-ordering
  - when not all inputs arrive at the same time



delay determined by time to discharge C_L, C_1 and C_2

delay determined by time to discharge C_L

Sizing and ordering effects



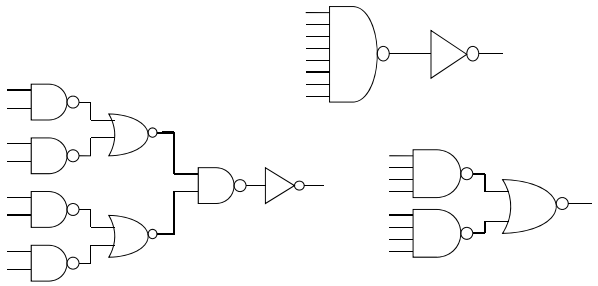
Example:
Progressive sizing in pull-down chain gives up to a 23% improvement.

Input ordering saves 6%
critical path A – 23%
critical path D – 17%

Fast complex gates: technique #3

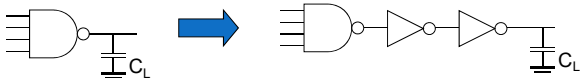
- Alternative logic structures

F = ABCDEFGH



Fast complex gates: technique #4

- Isolating fan-in from fan-out using buffer insertion



- *Real lesson:* optimizing the propagation delay of a gate in isolation is misguided

Technique #5 : Logical Effort

- The optimum effective fan-out for a chain of *N* inverters driving a load C_L is $f = \sqrt[N]{C_L / C_{in}}$
  - Set *N* such that the fan-out per stage is around 4, whenever possible (FO4)
- Can we generalize this approach (logical effort) to any gate?
  - The inverter equation is
    $t_p = t_{p0}(1 + C_{ext}/\gamma C_g) = t_{p0}(1 + f/\gamma)$
    we can generalize it to…

$$t_p = t_{p0}(p + g\,f/\gamma)$$

  - t_{p0} is the intrinsic delay of an inverter
  - *f* is the effective fan-out (C_{ext}/C_g) – also called the electrical effort
  - *p* is the ratio of the intrinsic delay of the gate relative to a simple inverter (a function of the gate topology and layout style): parasitic delay
  - *g* is the logical effort

Intrinsic delay term, p

- The more involved the structure of the complex gate, the higher the intrinsic delay compared to an inverter

Gate Type	P
Inverter	1
n-input NAND	n
n-input NOR	n
n-way mux	2n
XOR, XNOR	$n\,2^{n-1}$

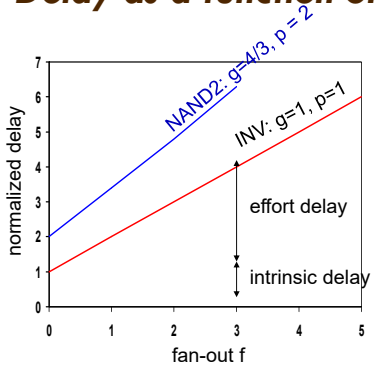
Ignoring second order effects such as internal node capacitances

Logical effort term, g

- **g** represents the fact that, for a given load, complex gates have to work harder than an inverter to produce a similar (speed) response
  - complex gates have higher input capacitance → worse output current

Gate Type	g (for 1 to 4 input gates)			
	1	2	3	4
Inverter	1			
NAND		4/3	5/3	(n+2)/3
NOR		5/3	7/3	(2n+1)/3
mux		2	2	2
XOR		4	12	

Delay as a function of fanout



Gate effort: h=fg

- The slope of the line is the logical effort of the gate (g)
- The y-axis intercept is the intrinsic delay (t_{p0})
- What are 2 ways to reduce delay?
  - Adjust the effective fanout (by scaling up transistor sizes)
  - Choose a gate with a different logical effort

Path delay of logic gate network

- Total path delay through a combinational logic block

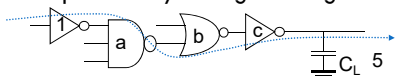
$$t_p = \sum t_{p,i} = t_{p0} \sum(p_i + (f_i\, g_i)/\gamma)$$

- Using the same analysis as for the inverter we find that each stage should bear the same gate effort

$$f_1 g_1 = f_2 g_2 = \ldots = f_N g_N \quad \text{or}$$

$$g_1 C_{ext,1}/C_{g,1} = g_2 C_{ext,2}/C_{g,2} = \ldots = g_N C_L/C_{g,N}$$

- Optimize the path delay through the logic network



How do we optimally size gates a, b, and c?

Path delay (equation derivation)

- The path logical effort, $G = \prod g_i$
- Path effective fanout (path electrical effort) is $F = C_L/C_{g1}$ [1]
- The branching effort accounts for fan-out to other gates in the network: $b = (C_{on\text{-}path} + C_{off\text{-}path})/C_{on\text{-}path}$
- The path branching effort is then $B = \prod b_i$

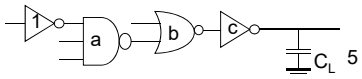
  …and the total path effort is then $H = GFB$ [1]
- The gate effort that minimizes path delay is $h = \sqrt[N]{H}$
- So, the minimum delay through the path is

$$D = t_{p0}\left(\sum_{j=1}^{N} p_j + \frac{N\left(\sqrt[N]{H}\right)}{\gamma}\right)$$

[1] Note the textbook swaps the definitions of F,H and f_i, h

Path delay of logic gate network (cont.)

- For gate i in the chain, its size is determined by $h = f_i g_i b_i$



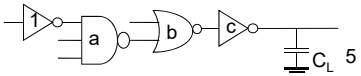
- For this network what do we need to compute h?
  - $F = C_L/C_{g1} = 5$
  - $G = 1 \times 5/3 \times 5/3 \times 1 = 25/9$
  - $B = 1$ (no branching)
  - $H = GFB = 125/9$, so the optimal stage effort is $h = \sqrt[4]{H} = 1.93$
  - Fanout factors are computed as $f_i = h/(g_i \cdot b_i)$. Since $b_i = 1$ we have:

    $f_1 = h/g_1 = 1.93$, $f_2 = 1.93 / (5/3) = 1.16$,

    $f_3 = 1.93/(5/3) = 1.16$, $f_4 = 1.93$

Path delay of logic gate network (cont.)

- Given f_i for each gate i in the chain, what is the final sizing?

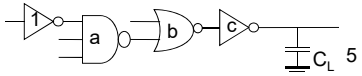


- $f_1 = 1.93$, $f_2 = 1.16$, $f_3 = 1.16$, $f_4 = 1.93$
- So the gate sizes are (*working from outputs to inputs*):
  - c: $f_4 = c_{ext,4}/c_{g,4} = c_L/c_{g,4} = 1.93$ ➔ $c_{g,4} = 5/1.93 = 2.59$
  - b: $f_3 = c_{ext,3}/c_{g,3} = c_{g,4}/c_{g,3} = 1.16$ ➔ $c_{g,3} = 2.59/1.16 = 2.23$
  - a: $f_2 = c_{ext,2}/c_{g,2} = c_{g,3}/c_{g,2} = 1.16$ ➔ $c_{g,2} = 2.23/1.16 = 1.93$
  - g_1: $f_1 = c_{ext,1}/c_{g,1} = c_{g,2}/c_{g,1} = 1.93$ ➔ $c_{g,1} = 1.93/1.93 = 1.00$

BROWN

Path delay of logic gate network (cont.)

- So what are the actually scaling sizes of the gates?



- Consider again the intrinsic capacitance values we calculated
  - $c_{g,4}$=5/1.93=2.56, $c_{g,3}$=2.59/1.16=2.23, $c_{g,2}$=2.23/1.16=1.93, $c_{g,1}$=1.93/1.93=1.00
  - These are relative to a *minimum sized inverter*, so we need to adjust to the gate type:
  - $c_{g,4}$ = 2.56 ➔ gate c is 2.56X size of minimum sized inverter, so S_c=2.56 since gate c is an inverter as well.
  - $c_{g,3}$ = 2.23 ➔ gate b is 2.23X size of minimum sized nor. Minimum sized nor is 5/3 as big as min sized inv so S_b = 2.23 X 3/5 = 1.34 (i.e., NOR is 1.34X size of minimum sized NOR)
  - $c_{g,2}$ = 1.93 ➔ gate a is 1.93X size of minimum sized NAND3. Minimum sized nand3 is 5/3 as big as min sized nand so S_a = 1.93X 3/5 = 1.16 (i.e. NAND is 1.16X size of minimum sized NAND3)