Introduction to CMOS VLSI Design

Logical Effort Part B

Original Lecture by Jay Brockman

University of Notre Dame Fall 2008

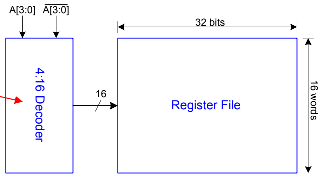
Modified by Peter Kogge Fall 2010,2011,2015, 2018

Based on lecture slides by David Harris, Harvey Mudd College

http://www.cmosvlsi.com/coursematerials.html

Review: Motivating Example

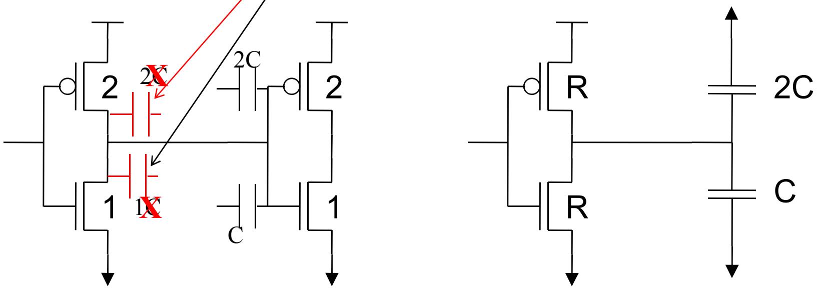
- ❑ Ben Bitdiddle is the memory designer for the Motoroil 68W86, an embedded automotive processor. Help Ben design the *decoder* for a 16x32 register file.



- ❑ Decoder specifications:
  - – 16 word register file
  - – Each word is 32 bits wide
  - – Each file bit presents load of 3 unit-sized transistors
  - – Both true & complementary address inputs A[3:0] available
  - – Each address input may drive 10 unit-sized transistors
- ❑ Ben needs to decide:
  - – How many stages to use in decoder driver output buffers?
  - – How large should each gate be?
  - – How fast can decoder operate?

Review: Ideal Gate Delay τ

- Imagine ***ideal unit inverter*** (*no* parasitic diffusion capacitance & unit on resistance) driving *identical inverter*



- τ = 3RC = Ideal Inverter Delay

Review: Linear Delay Model

$$\text{delay } d = p + f = p + hg$$

Prior delay had two parts

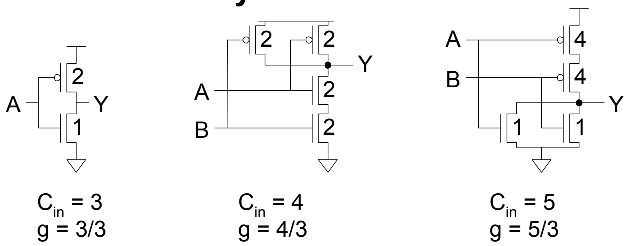
- *p: **Parasitic delay*** due to internal diffusion capacitance of gate
  - 3RC for inverter
  - *Independent* of load = sum of diffusion caps
- ***f: Effort delay*** due to load capacitance of gates being driven
  - 3h RC when driving h unit inverters
  - *Proportional* to total load capacitance
- = h*g if driving *identical* circuits (copies of itself)
  - h = # of copies of gate (also called "**Fanout**")
  - g = "**Logical Effort**" (function of gate complexity)
    - 3RC for inverter

Review: Normalized Linear Delay

- ❑ Remember
  - – 3RC = parasitic delay of unit inverter
  - – 3RC + 3hRC = delay if driving h inverter copies
- ❑ **Normalized delay**: divide delay by **3RC**
  - – Measure of how much "slower" a circuit is than an inverter
  - – = **1 + h** for inverter driving h identical inverters

Determining Logical Effort

- ❑ ***Logical effort g:*** ratio of input capacitance of a gate to input capacitance of an inverter *delivering the same output current* (pullup/down resistance).
- ❑ Measured from delay vs. fanout plots
- ❑ Or estimate by counting transistor widths
  - – **AND divide by 3 to "normalize"** to unit inverter



Question: does g change if we scale all the transistors wided?

Summary: (p.156)

Gate Type	Number of inputs														
	1			2			3			4			N		
Inverter	3	1	1												
NAND				4	2	4/3	5	3	5/3	6	4	6/3	N+2	N	(N+2)/3
NOR				5	2	5/3	7	3	7/3	9	4	9/3	2N+1	N	(2N+1)/3
TriState/mux		2	2		4	2		6	2		8	2		2N	2
XOR, XNOR					4	4,4		6	6,12,6		8	8,16,16,8			

(x,y,z) = Input Cap, p, g

*All inputs are **not** the same!*

Parasitic: delay driving 0 load (divided by 3RC)

Logical effort: Input Cap of gate / Input Cap of inverter of same current

Delay Plots

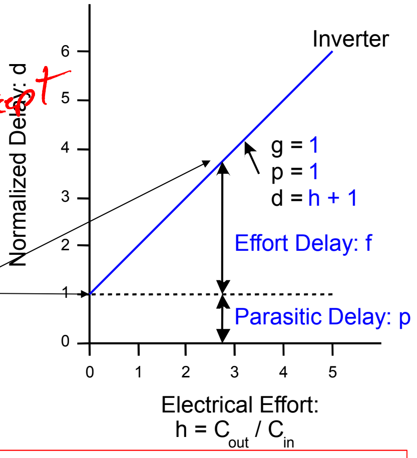
$d = f + p$

$= gh + p$

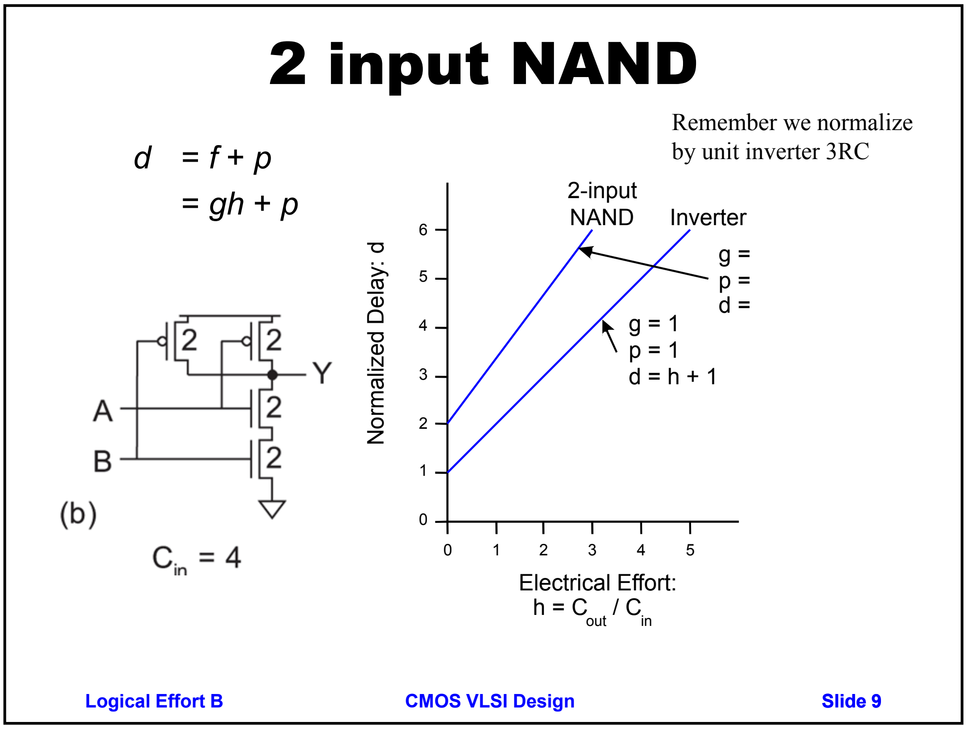
slope intercept

Lets take our invertor:

- d = h + 1
- p = 1 (y-intercept)
- g = 1 (slope)



What does "Normalized Delay" mean to designer?

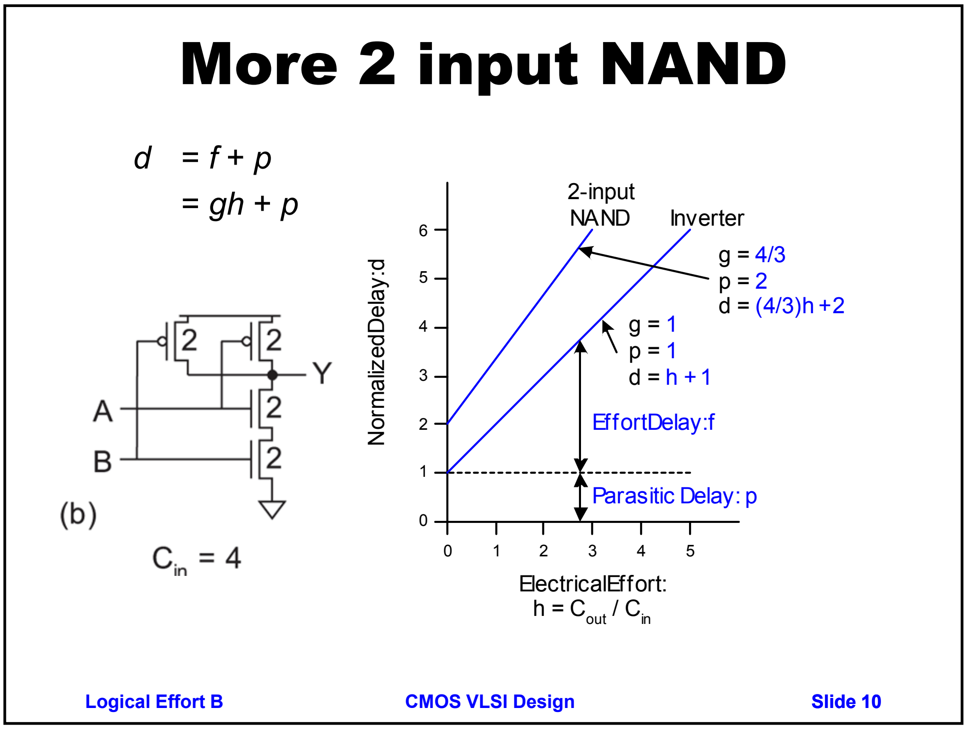


2 input NAND

$d = f + p$

$= gh + p$

Remember we normalize by unit inverter 3RC

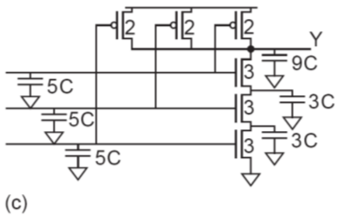


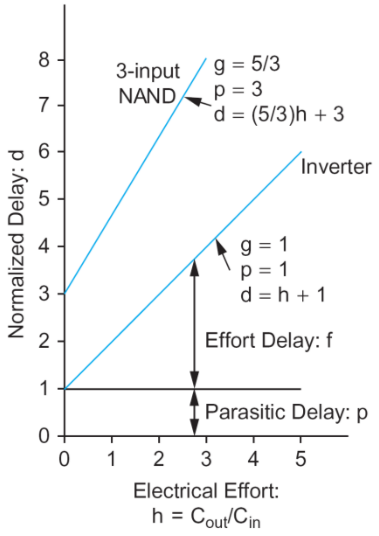
More 2 input NAND

$d = f + p$

$= gh + p$

3 Input NAND

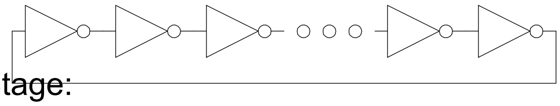




Eg: Ring Oscillator (p.158)

❑ Estimate the frequency of an N-stage ring oscillator

– N odd



Each Stage:

Logical Effort: g = 1

Electrical Effort: h = 1

Parasitic Delay: p = 1

Stage Delay: d = (1 + h) = (1 + 1) = 2

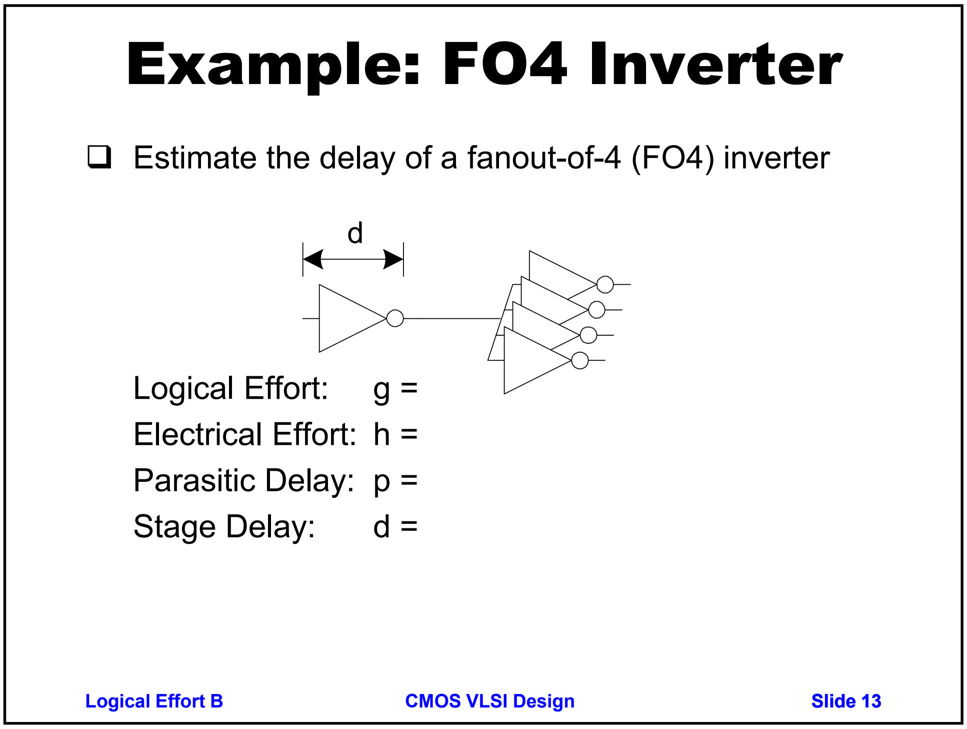
Total delay: Nd = 2N (normalized)

= 2Dτ (in seconds)

Overall Frequency: $f_{osc} = 1/(2N\tau)$

31 stage ring oscillator

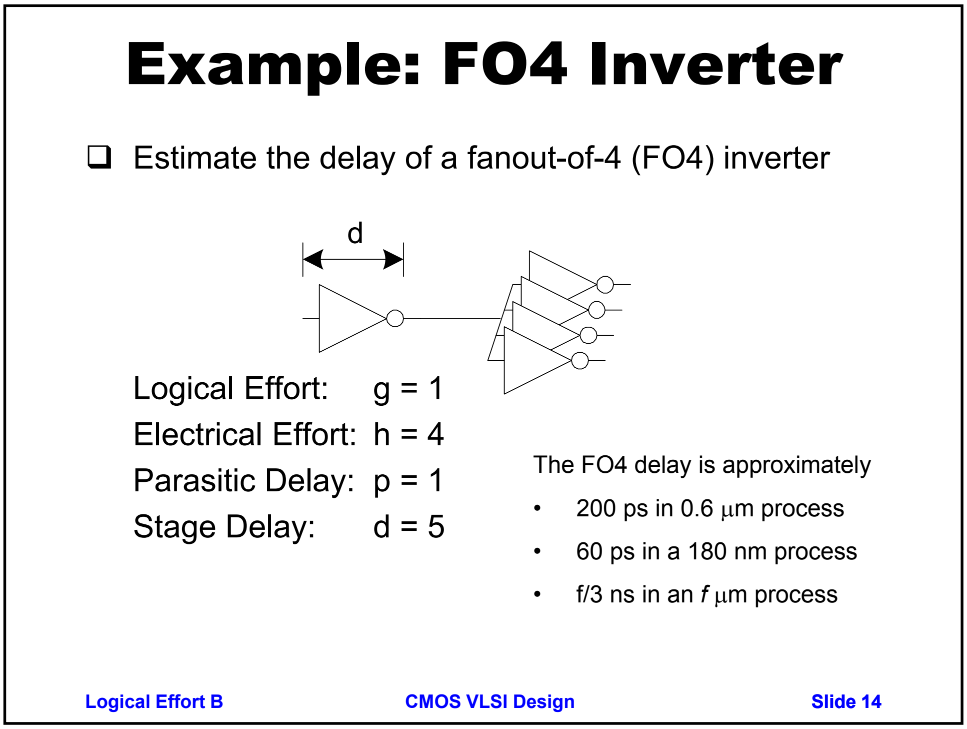
- 65nm process (3ps delay) = 2.7GHz
- 0.6 μm process ~ 200 MHz



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 approximately

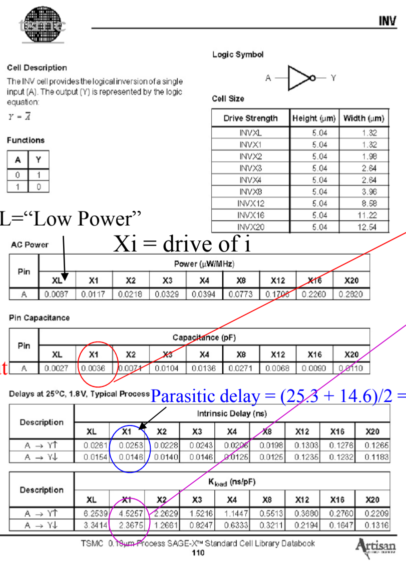
- 200 ps in 0.6 μm process
- 60 ps in a 180 nm process
- f/3 ns in an *f* μm process

Drive (p.159)

- ❑ Most libraries have multiple versions of common gates
  - – Named <gate-type>_#x
- ❑ Versions differ by “size” of transistors
  - – Denoted by # in name
  - – Called the gate version’s ***drive***
    - • Relative to unit inverter
- ❑ **Drive x** = C_{in} / g
- ❑ Thus delay = C_{out} / x + p
- ❑ Question: what’s speed difference between
  - – nand2_1x
  - – nand2_3x

(p. 160)

Figure 4.25 Artisan Components cell library datasheets. Reprinted with permission.
(180 nm process)



INV

Cell Description

The INV cell provides the logical inversion of a single input (A). The output (Y) is represented by the logic equation:

$Y = \overline{A}$

Logic Symbol

Functions

A	Y
0	1
1	0

Cell Size

Drive Strength	Height (μm)	Width (μm)
INVXL	5.04	1.32
INVX1	5.04	1.32
INVX2	5.04	1.98
INVX3	5.04	2.64
INVX4	5.04	2.64
INVX8	5.04	3.96
INVX12	5.04	8.58
INVX16	5.04	11.22
INVX20	5.04	12.54

XL=“Low Power”

Xi = drive of i

AC Power

Pin	Power (μW/MHz)								
	XL	X1	X2	X3	X4	X8	X12	X16	X20
A	0.0087	0.0117	0.0218	0.0329	0.0394	0.0773	0.1706	0.2260	0.2820

Pin Capacitance

Pin	Capacitance (pF)								
	XL	X1	X2	X3	X4	X8	X12	X16	X20
A	0.0027	0.0036	0.0071	0.0104	0.0136	0.0271	0.0068	0.0090	0.0110

Input cap

Delays at 25°C, 1.8V, Typical Process

Parasitic delay = (25.3 + 14.6)/2 = 20ps

Description	Intrinsic Delay (ns)								
	XL	X1	X2	X3	X4	X8	X12	X16	X20
A → Y↑	0.0261	0.0253	0.0228	0.0243	0.0206	0.0198	0.1303	0.1276	0.1265
A → Y↓	0.0154	0.0146	0.0140	0.0146	0.0125	0.0125	0.1235	0.1232	0.1183

Description	K_{load} (ns/pF)								
	XL	X1	X2	X3	X4	X8	X12	X16	X20
A → Y↑	6.2539	4.5257	2.2629	1.5216	1.1447	0.5513	0.3680	0.2760	0.2209
A → Y↓	3.3414	2.3675	1.2661	0.8247	0.6333	0.3211	0.2194	0.1647	0.1316

TSMC 0.18μm Process SAGE-X™ Standard Cell Library Databook
110

Artisan

Average (4.53+2.37)/2 = 3.45ns/pf

_1 delay driving h _1 inverters
= 20 + 3.6*h*3.45 ps

= 20 + 12.4h ps

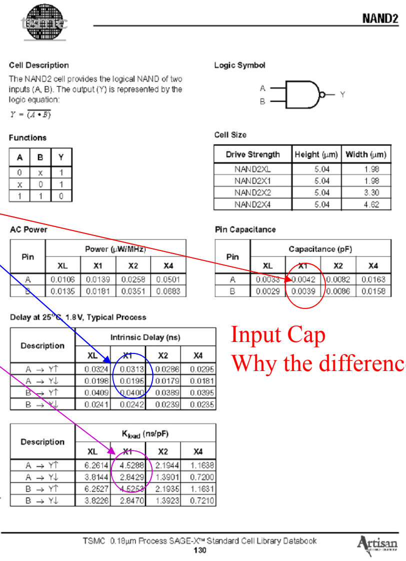
τ = 12.4ps
p_{inv} = 20ps = (20/12.4)τ
= 1.6 in normalized terms

(p. 160)

Figure 4.25 Artisan Components cell library datasheets. Reprinted with permission.

(180 nm process)

- Lets look at A input
- Parasitic delay = (31.3+19.5)/2
  = 25.4ps
- C_{in} = 4.2fF
- K_{load} = (4.53+2.84)/2 ns/pF
  = 3.69 ns/pF
  = 3.69 ps/fF
- t_{pd} = 25.4 +4.2*3.69*h ps
  = 25.4 + 15.5*h ps
- Normalizing by inverter 12.4ps
  - p = 25.4/12.4 = 2.05
  - g = 15.5/12.4 = 1.25
    - versus 4/3 = 1.33 from model



NAND2

Cell Description

The NAND2 cell provides the logical NAND of two inputs (A, B). The output (Y) is represented by the logic equation:

$Y = \overline{(A \bullet B)}$

Logic Symbol

Functions

A	B	Y
0	x	1
x	0	1
1	1	0

Cell Size

Drive Strength	Height (μm)	Width (μm)
NAND2XL	5.04	1.98
NAND2X1	5.04	1.98
NAND2X2	5.04	3.30
NAND2X4	5.04	4.62

AC Power

Pin	Power (μW/MHz)			
	XL	X1	X2	X4
A	0.0106	0.0139	0.0258	0.0501
B	0.0135	0.0181	0.0351	0.0683

Pin Capacitance

Pin	Capacitance (pF)			
	XL	X1	X2	X4
A	0.0033	0.0042	0.0082	0.0163
B	0.0029	0.0039	0.0086	0.0158

Delay at 25°C, 1.8 V, Typical Process

Description	Intrinsic Delay (ns)			
	XL	X1	X2	X4
A → Y↑	0.0324	0.0313	0.0286	0.0295
A → Y↓	0.0198	0.0195	0.0179	0.0181
B → Y↑	0.0409	0.0400	0.0389	0.0395
B → Y↓	0.0241	0.0242	0.0239	0.0235

Description	K_{load} (ns/pF)			
	XL	X1	X2	X4
A → Y↑	6.2614	4.5288	2.1944	1.1638
A → Y↓	3.8144	2.8429	1.3901	0.7200
B → Y↑	6.2527	4.5253	2.1935	1.1631
B → Y↓	3.8226	2.8470	1.3923	0.7210

TSMC 0.18μm Process SAGE-X™ Standard Cell Library Databook
130

Input Cap
Why the difference?

(p. 161)

Limitations to Linear Delay Model

- ❑ Input & output slopes:
  - not square
- ❑ Input arrival times: complex interactions
  - when 2 or more inputs change at same time
- ❑ Velocity Saturation:
  - We assume N transistors in series must be N times wider
  - But series transistors see less velocity saturation
    - & hence less resistance
- ❑ Voltage Dependencies:
  - $\tau \sim V^* V_{DD}/(V_{DD} - V_T)^\alpha$
- ❑ Gate/Source Dependencies:
  - We assumed gate caps terminate on a fixed rail
  - In reality to “middle” of channel
- ❑ Bootstrapping:
  - Transistors have “gate to drain” capacitance
  - Causes input to output “lifting”

Bootstrapping

Real-world: some cap from gate to drain

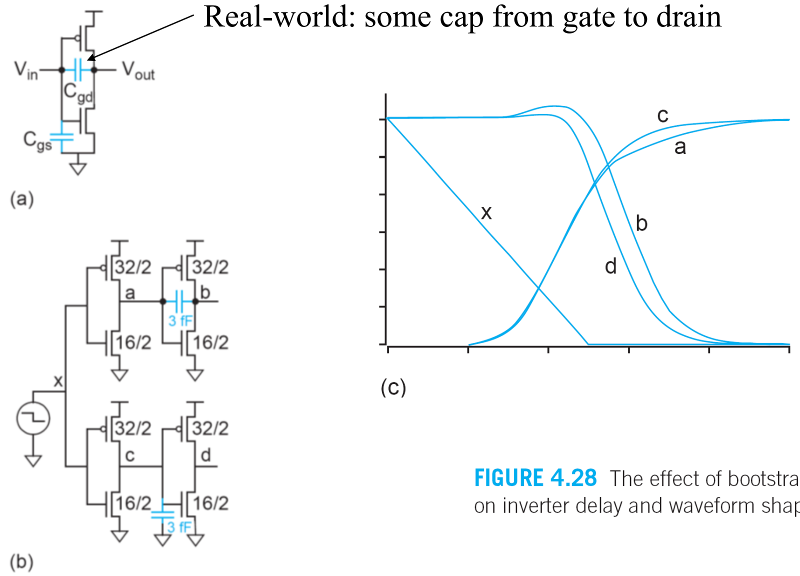
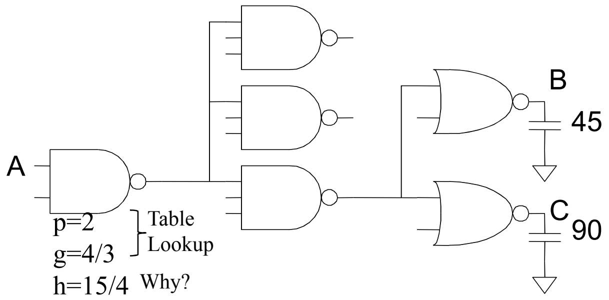


FIGURE 4.28 The effect of bootstrapping on inverter delay and waveform shape

Delay in Multi-Stage Circuits

A Sample Multi Stage Circuit



Gate Type	Number of inputs														
	1			2			3			4			N		
Inverter	3	1	1												
NAND				4	2	4/3	5	3	5/3	6	4	6/3	N+2	N	(N+2)/3
NOR				5	2	5/3	7	3	7/3	9	4	9/3	2N+1	N	(2N+1)/3
TriState/mux		2	2		4	2		6	2		8	2		2N	2
XOR, XNOR					4	4,4		6	6,12,6		8	8,16,16,8			
(x,y,z) = Input Cap, p, g															

Scaling Transistors

- ❑ What if all transistors in gate G got *wider* by k?
  - – Denote as gate “G(k)”
- ❑ **Parasitic delay** of G(k): delay of unloaded gate
  - – Diffusion capacitance increases by k
  - – Resistance decreases by k
  - – Result: *No change*
- ❑ **Effort delay**: ratio of load cap to input cap
  - – If drive same # of G(k) as before, *no change*
  - – If drive same # of G(1) as before, *decrease* by 1/k
  - – If drive k times as many G(1), *no change*
- ❑ Result: **fanout** to type G(1) gates *increases by k*
- ❑ YOU CAN DRIVE MORE GATES AT SAME SPEED!
  - – OR DRIVE SAME GATES FASTER

Design Question

- ❑ Given a signal path thru multiple gates
  - – Of different types
  - – And different #s on each output
- ❑ How do we select transistor scale factors?

- ❑ Answer: analyze/select “input capacitance”
  - – And then adjust transistor scaling to give you that value

MultiStage Logic Networks

(p. 163)

Relative Input Capacitance

(based on gate design & **transistor size**)

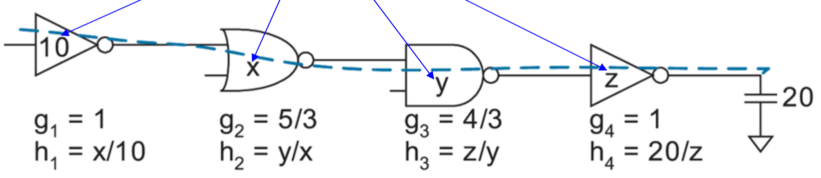
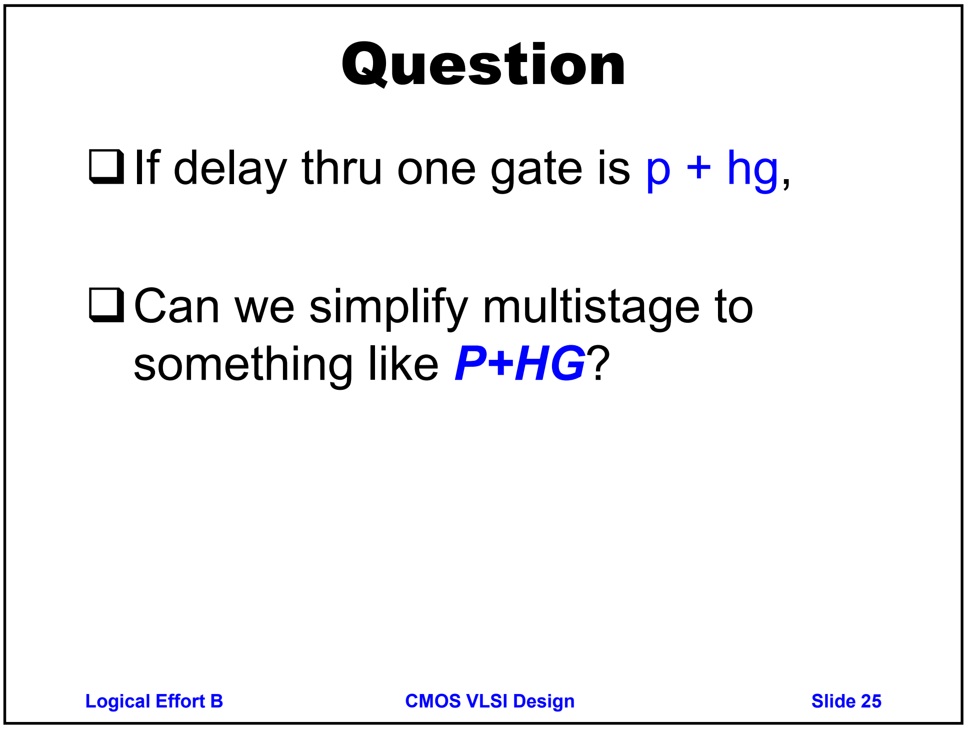


FIG 4.29 Multistage logic network

g_i = logical effort to drive a gate of type i = input cap/cap of inverter

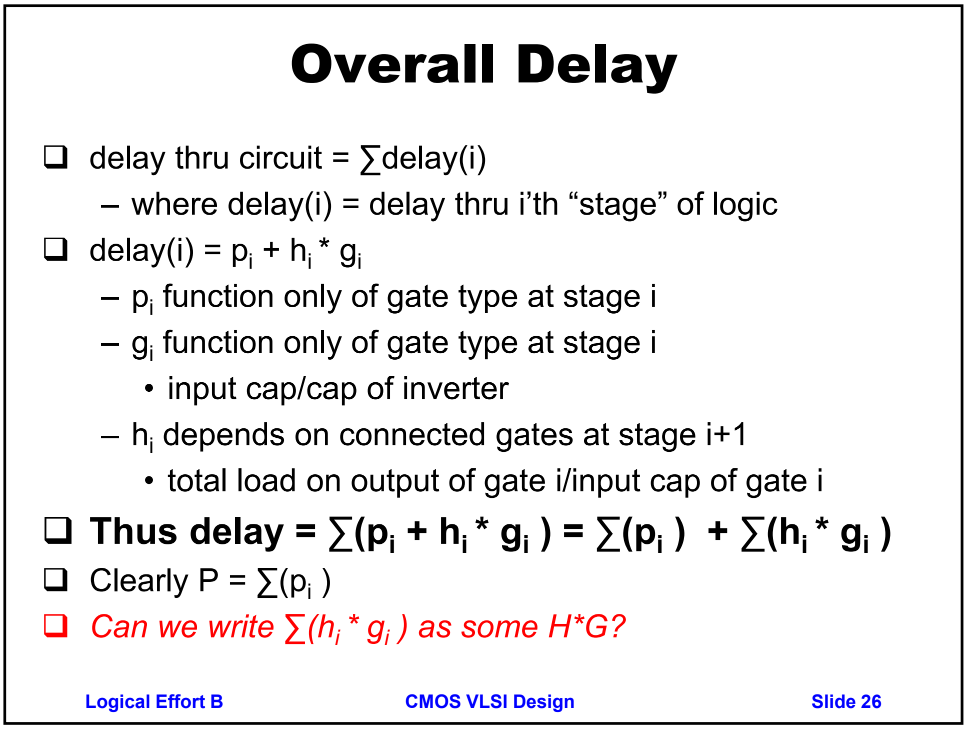
h_i = fanout of gates of type i = load cap/input cap



Question

□ If delay thru one gate is $p + hg$,

□ Can we simplify multistage to something like ***P+HG***?



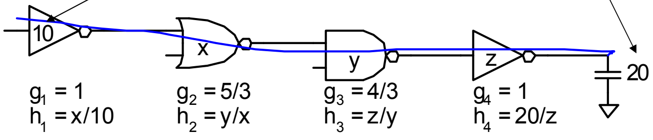
Overall Delay

- □ delay thru circuit = ∑delay(i)
  - – where delay(i) = delay thru i'th "stage" of logic
- □ delay(i) = $p_i + h_i * g_i$
  - – p_i function only of gate type at stage i
  - – g_i function only of gate type at stage i
    - input cap/cap of inverter
  - – h_i depends on connected gates at stage i+1
    - total load on output of gate i/input cap of gate i
- □ **Thus delay = $\sum(p_i + h_i * g_i) = \sum(p_i) + \sum(h_i * g_i)$**
- □ Clearly P = $\sum(p_i)$
- □ *Can we write $\sum(h_i * g_i)$ as some H*G?*

(p. 164)

Definitions

- ❑ ***Path Logical Effort*** $G = \prod g_i$
- ❑ ***Path Electrical Effort*** $H = \dfrac{C_{\text{out-path}}}{C_{\text{in-path}}}$
- ❑ ***Path Effort*** $F = \prod f_i = \prod g_i h_i$



Question: Can we write F = GH?

Can We State F =GH?

- ❑ **No!** Consider paths that **branch**:
  - – Not all load at one stage is on path to output
- ❑ Individual terms

  $g_1 = g_2 = 1$ (inverters)

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

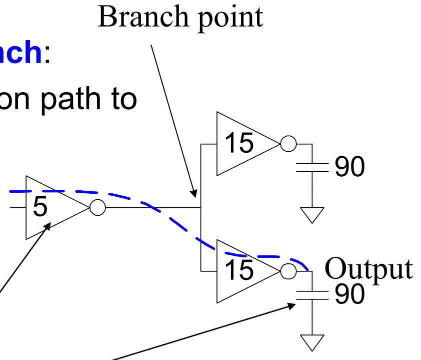
  $h_2 = 90 / 15 = 6$
- ❑ Path Terms

  $G = \Pi g_i = 1 \times 1 = 1$

  $H = C_{out}/C_{in} = 90 / 5 = 18$

  Thus $GH = 18$

  Versus $F = g_1 h_1 g_2 h_2 = 36 =$ **2GH != GH**



Question: What did GH overlook?

Branching Effort

- ❑ Introduce ***Branching Effort B***
  - – Accounts for signal branching between stages in path

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

$$B = \prod b_i \qquad \prod h_i = BH$$

- ❑ Now we compute the ***Path Effort*** **F = GBH**
- ❑ Or **delay = P + GBH**

Redo

- ❑ Individual terms

g_1 = g_2 = 1 (inverters)

h_1 = (15 +15) / 5 = 6

h_2 = 90 / 15 = 6

b_1 = (15+15)/15 = 2

b_2 = 90/90 =1

- ❑ Path Terms

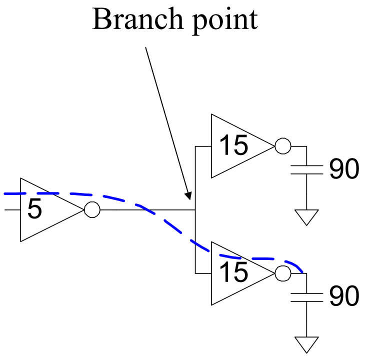
G = Πg_i = 1x1 = 1

H = C_{out}/C_{in} = 90 / 5 = 18

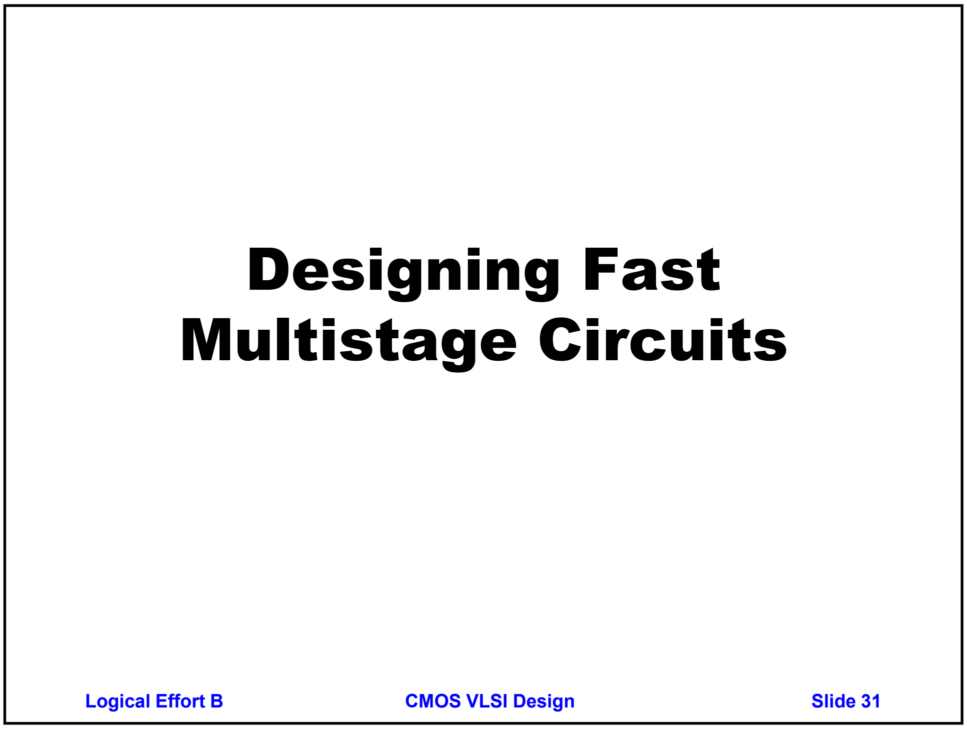
B = 2x1 = 2

Thus GBH = 36

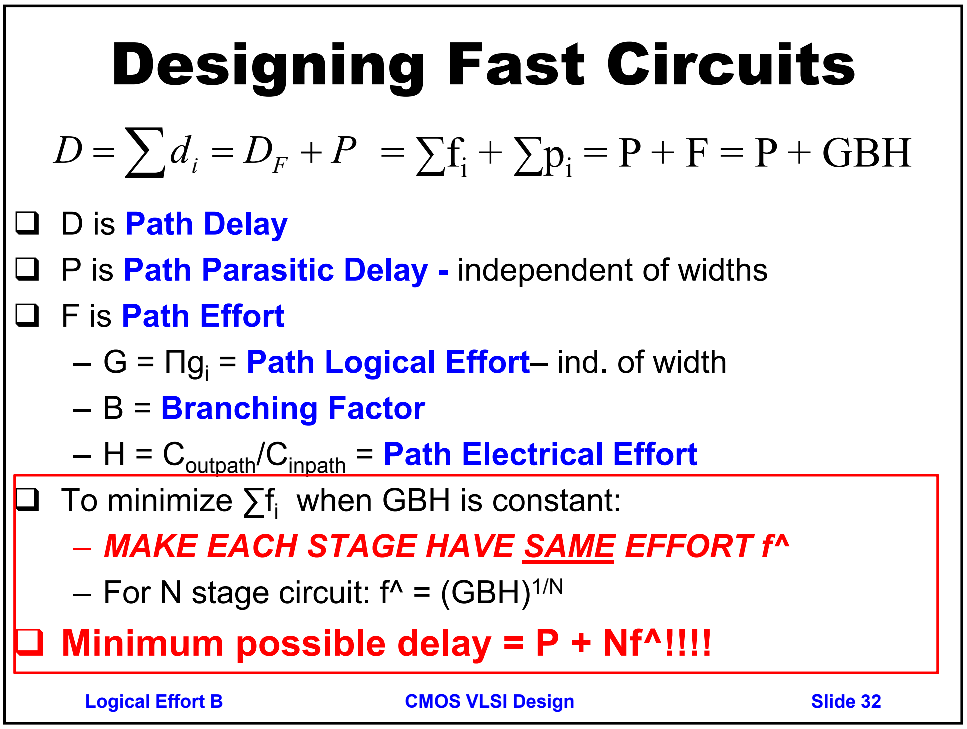
Versus F = $g_1h_1g_2h_2$ = 36!



In this case: exact match



Designing Fast Multistage Circuits



Designing Fast Circuits

$$D = \sum d_i = D_F + P \quad = \sum \mathrm{f_i} + \sum \mathrm{p_i} = \mathrm{P} + \mathrm{F} = \mathrm{P} + \mathrm{GBH}$$

❑ D is **Path Delay**

❑ P is **Path Parasitic Delay -** independent of widths

❑ F is **Path Effort**

- G = Πg_i = **Path Logical Effort**– ind. of width
- B = **Branching Factor**
- H = $C_{outpath}/C_{inpath}$ = **Path Electrical Effort**

❑ To minimize $\sum f_i$ when GBH is constant:

- ***MAKE EACH STAGE HAVE SAME EFFORT f^***
- For N stage circuit: f^ = $(GBH)^{1/N}$

❑ **Minimum possible delay = P + Nf^!!!!**

How Do We Find Gate Sizes to Reach this Equal Effort?

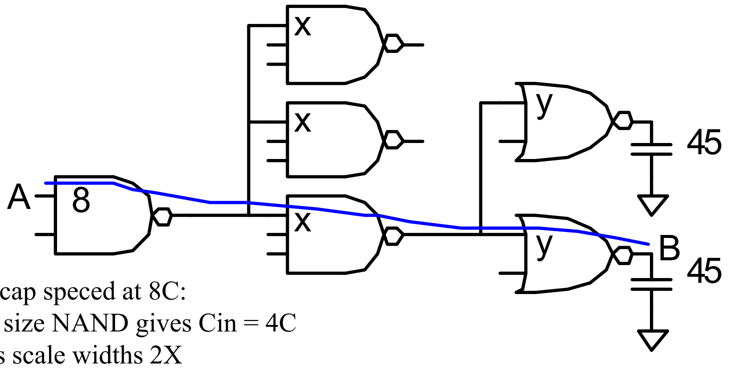
- ❑ How wide should gates be for least delay?
- ❑ Typically C_{in} of first gate is pre-specified
- ❑ Working backward from load,
  - – apply transformation to find input capacitance of each gate, given the load it drives.
  - – Use this for the load capacitance on previous stage
- ❑ Check work by verifying input cap spec i
- ❑ Then use input cap of each gate to size each gate

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

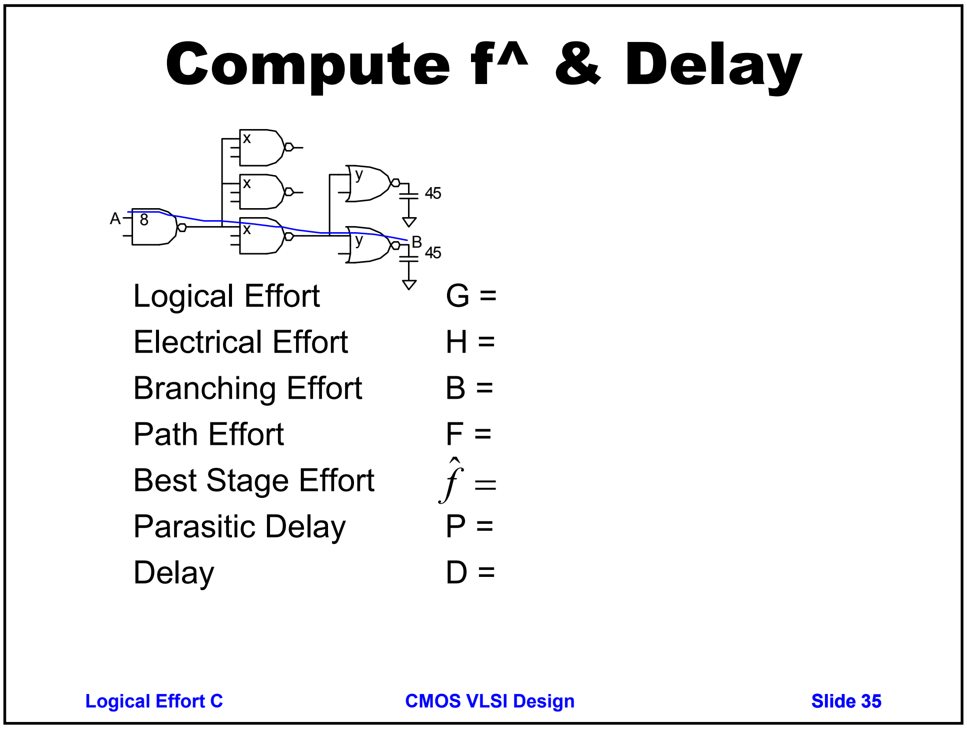
$$\Rightarrow C_{in_i} = \frac{g_i C_{out_i}}{\hat{f}}$$

Example: 3-stage path

- ❑ Select input capacitance x and y for least delay from A to B, given capacitance on A is 8C



Input cap speced at 8C:
- • Min size NAND gives Cin = 4C
- • Thus scale widths 2X
- • or n-type are 2X2 = 4 wide
- • p-type are also 2x2 = 4 wide

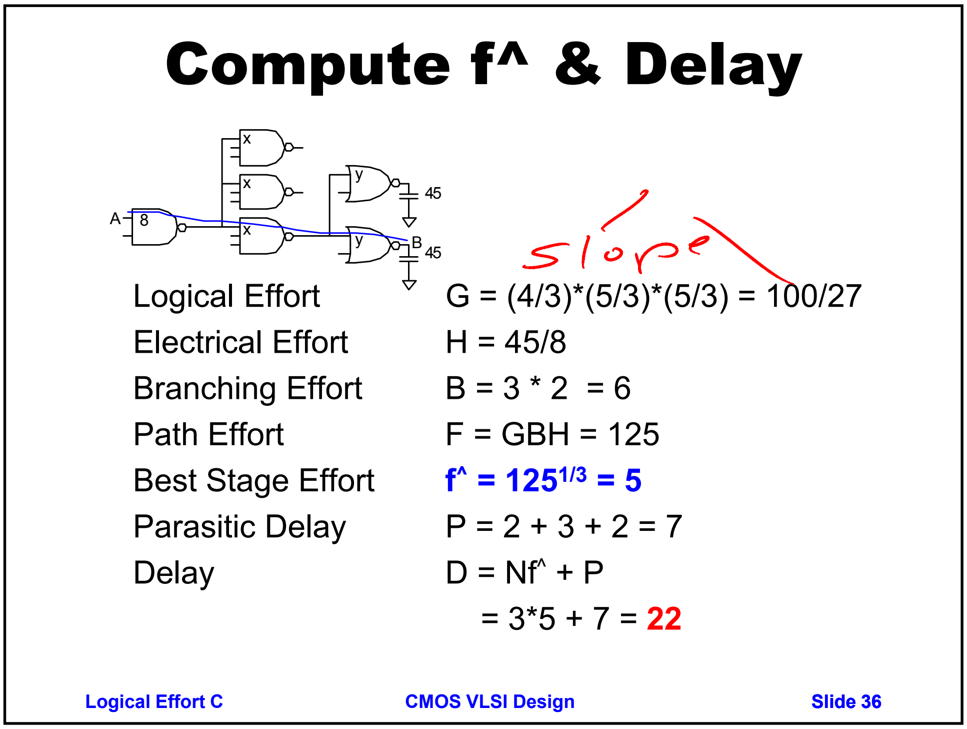


Compute f^ & Delay

A
8
x
x
x
y
y
45
B
45

Logical Effort	G =
Electrical Effort	H =
Branching Effort	B =
Path Effort	F =
Best Stage Effort	$\hat{f} =$
Parasitic Delay	P =
Delay	D =

Logical Effort C CMOS VLSI Design Slide 35



Compute f^ & Delay

A
8
x
x
x
y
y
45
B
45

slope

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	**f^ = $125^{1/3}$ = 5**
Parasitic Delay	P = 2 + 3 + 2 = 7
Delay	D = Nf^ + P = 3*5 + 7 = **22**

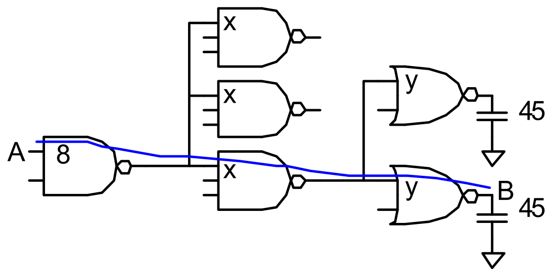
Logical Effort C CMOS VLSI Design Slide 36

Work Backwards

- ❑ Work backward for sizes using $C_{in[i]} = C_{out[i]*}g_i/f^{\wedge}$

  y =

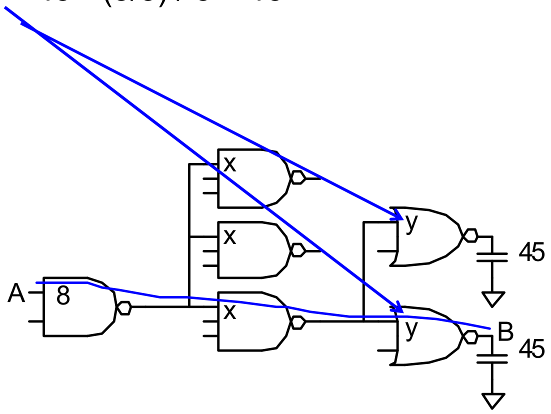
  x =



Work Backwards

- ❑ Work backward for sizes using $C_{in[i]} = C_{out[i]*}g_i/f^{\wedge}$

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

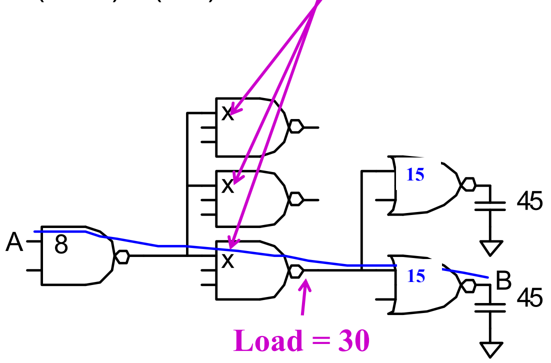


Work Backwards

- ❑ Work backward for sizes using $C_{in[i]} = C_{out[i]*}g_i/f^\wedge$

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

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



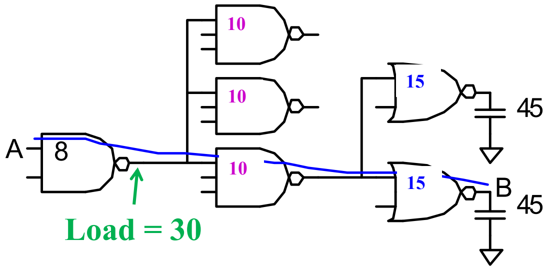
Work Backwards

- ❑ Work backward for sizes using $C_{in[i]} = C_{out[i]*}g_i/f^\wedge$

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

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

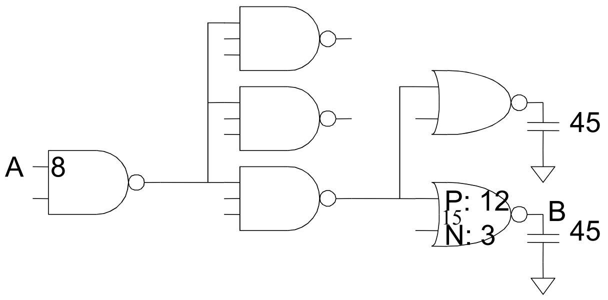
  A = (10+10+10)*(4/3)/5 = 8 AND IT CHECKS!



Size Last Gate

- ❑ Now size last stage: x = 15
  - – 2 input NOR has unit input cap of 5
- ❑ To get an input cap of 15=> widths 15/5 = 3X unit!
  - – => p-type are 3*4 = 12 wide

    => n-type are 3*1 = 3 wide

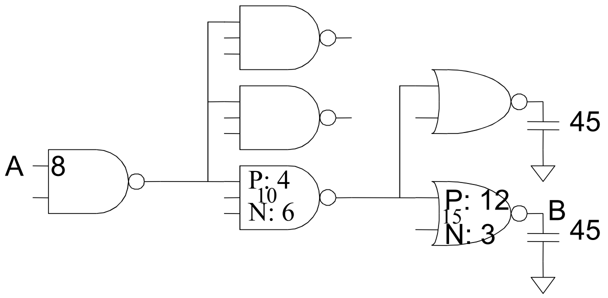


Size Middle Gate

- ❑ Now size 2nd stage 3 input NAND

  y = 10;
  - – unit input cap of 5 => widths 10/5 = 2X unit!
  - – p:n ratio of 2:3 => p-type are 2*2 = 4 wide

    => n-type are 3*2 = 6 wide



Size First Gate

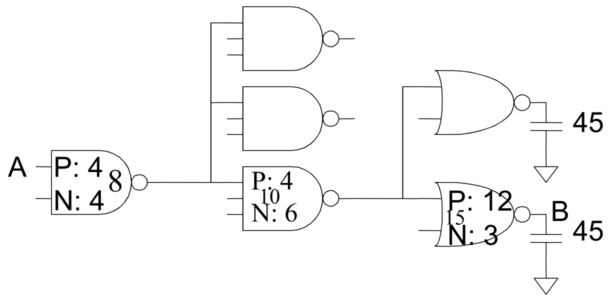
- ❑ Now size 1st stage

  Cin = 8; 2 input NAND has

  – unit input cap of 4 => widths 8/4 = 2X unit!

  – p:n ratio of 1:1 => p-type are 2*2 = 4 wide

  => n-type are 2*2 = 4 wide



Checking Delay

- ❑ Let's check delay

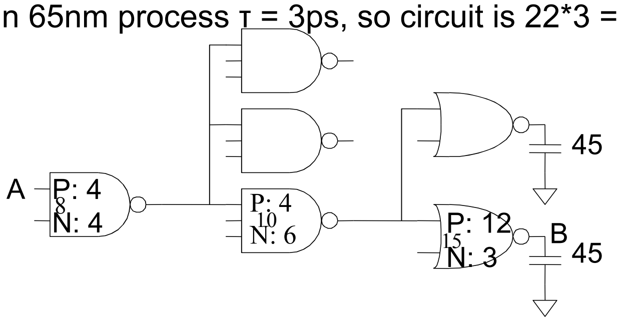
  – $d_1 = g_1h_1 + p_1 = \{(10+10+10)/8\}*(4/3) + 2 = 7$

  – $d_2 = g_2h_2 + p_2 = \{(15+15)/10\}*(5/3) + 3 = 8$

  – $d_3 = g_3h_3 + p_3 = \{45/15\}*(5/3) + 2 = 7$

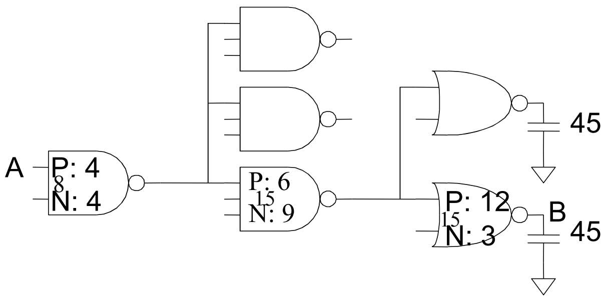
  – delay = 7 + 8 + 7 = 22

  – in 65nm process τ = 3ps, so circuit is 22*3 = 66ps



What If We Try to Tweak?

- ❑ What if we made stage 2 even bigger (to be faster)
  - – $d_2 = g_2h_2 + p_2$ = {(15+15)/15}*(5/3) + 3 = 6.3 (*faster*)
  - – but $d_1 = g_1h_1 + p_1$ = {(15+15+15)/8}*(4/3) + 2 = 9.5 (*slower*)
  - – and $d_3 = g_3h_3 + p_3$ = {45/15}*(5/3) + 2 = 7 (no change)
  - – delay = 9.5 + 6.3 + 7 = ***22.8 > 22 – SLOWER CIRCUIT***

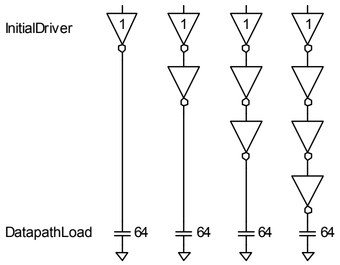


Choosing Best # of Stages

- ❑ Many logic functions have multiple possible circuits (**topologies**)
- ❑ Goal: select topology, est. delay, & size transistors
- ❑ We know in general
  - – NANDs better than NORs
  - – Gates with fewer inputs better than more inputs
- ❑ **Typical shortcut: estimate delay by # of stages**
  - – **Assuming constant "gate delay"**
  - – **and thus shorter paths are faster**
- ❑ ***THIS IS NOT ALWAYS TRUE!***
  - – ***Eg: Adding inverters at end with increasing sizes can speed up circuit, esp. when high load***

Example (p. 166)

- ❑ How many stages should a path use?
  - – Minimizing number of stages is not always fastest
- ❑ Example: drive 64-bit datapath with unit inverter
  - – Each bit eqvt to unit inverter in load



N:	1	2	3	4
f:				
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

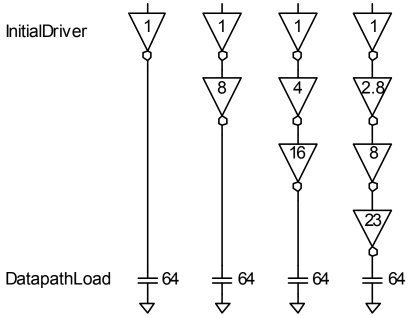
H = 64

G = 1

F = HG = 64

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

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



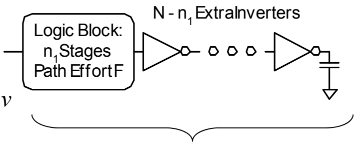
N:	1	2	3	4
f:	64	8	4	2.8
D:	65	18	15	15.3

Fastest

General Derivation

- ❑ Consider **adding inverters** to end of n_1 stage path
  - – How many give least delay?

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



N total stages with (N-n_1) Inverters
• do not change logical effort
• do add parasitic delay

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

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

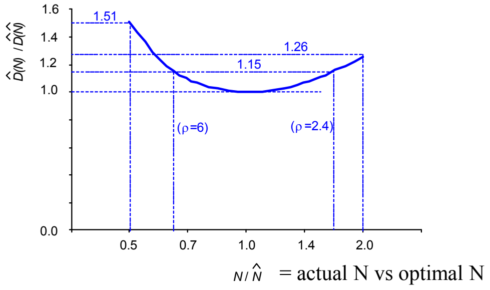
$$p_{inv} + \rho\left(1 - \ln \rho\right) = 0$$

Best Stage Effort

- ❑ $p_{inv} + \rho\left(1 - \ln \rho\right) = 0$ has no closed-form solution
- ❑ Neglecting parasitics (p_{inv} = 0), we find ρ = 2.718 (e)
- ❑ For p_{inv} = 1, solve numerically for ρ = 3.59
- ❑ Again,
  - – these ρ values are best logical effort per stage
  - – when you have $\hat{N} = \log_{\rho} F$ stages

Sensitivity Analysis

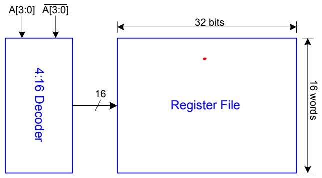
- □ How sensitive is delay to using exactly the best number of stages?



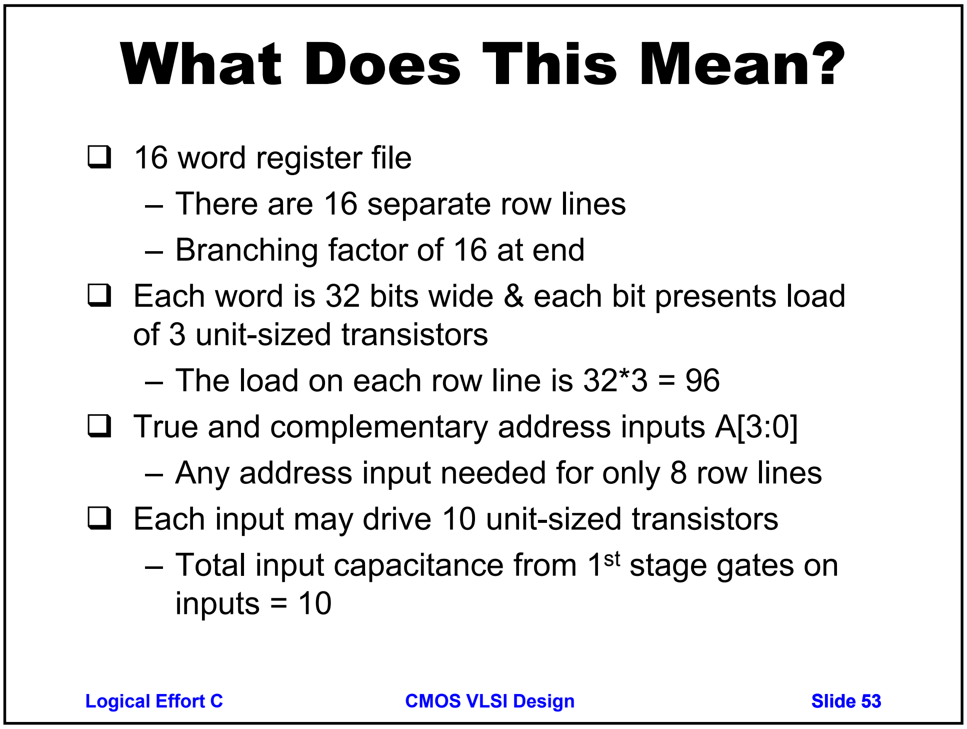
- □ 2.4 < ρ < 6 gives delay within 15% of optimal
  - – We can be sloppy!
  - – **Book likes ρ = 4**

1st 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.

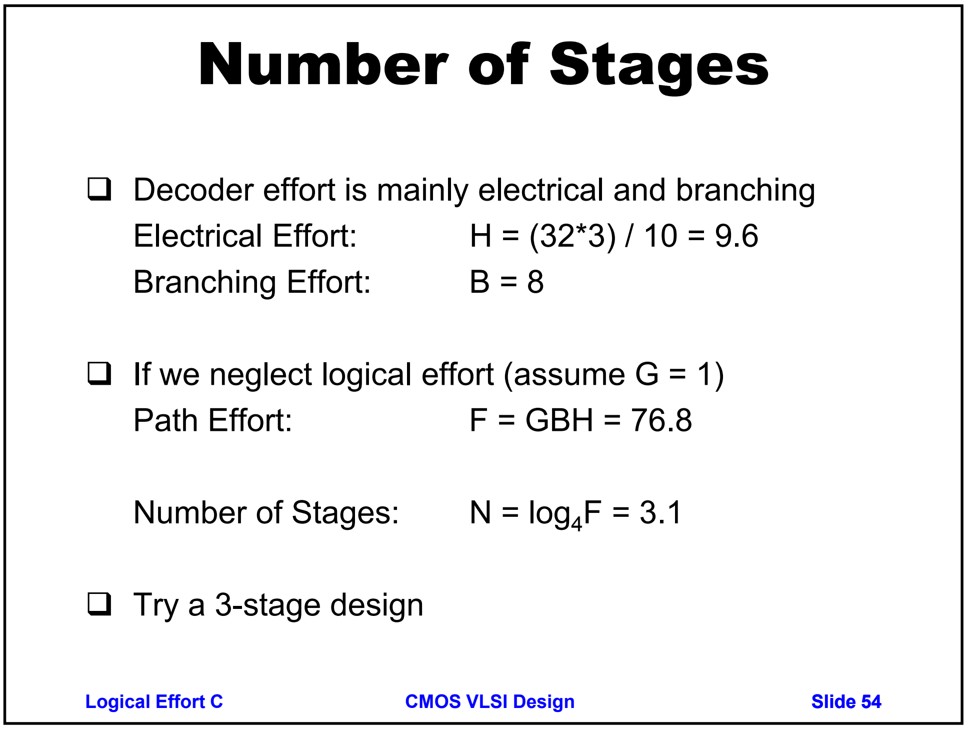


- □ Decoder specifications:
  - – 16 word register file
  - – Each word is 32 bits wide
  - – Each bit presents load of 3 unit-sized transistors
  - – True and complementary address inputs A[3:0]
  - – Each input may drive 10 unit-sized transistors
- □ Ben needs to decide:
  - – How many stages to use?
  - – How large should each gate be?
  - – How fast can decoder operate?



What Does This Mean?

- ❑ 16 word register file
  - – There are 16 separate row lines
  - – Branching factor of 16 at end
- ❑ Each word is 32 bits wide & each bit presents load of 3 unit-sized transistors
  - – The load on each row line is 32*3 = 96
- ❑ True and complementary address inputs A[3:0]
  - – Any address input needed for only 8 row lines
- ❑ Each input may drive 10 unit-sized transistors
  - – Total input capacitance from 1st stage gates on inputs = 10



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 Gate Sizes & Delay

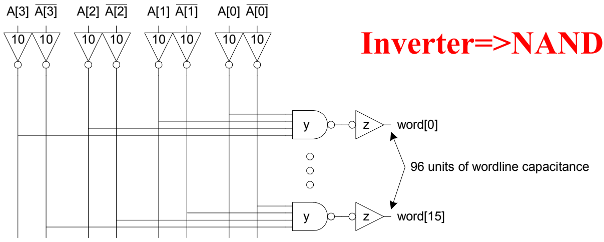
Logical Effort: G = 1 * 6/3 * 1 = 2

Path Effort: F = GBH = 2*8*9.6 =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



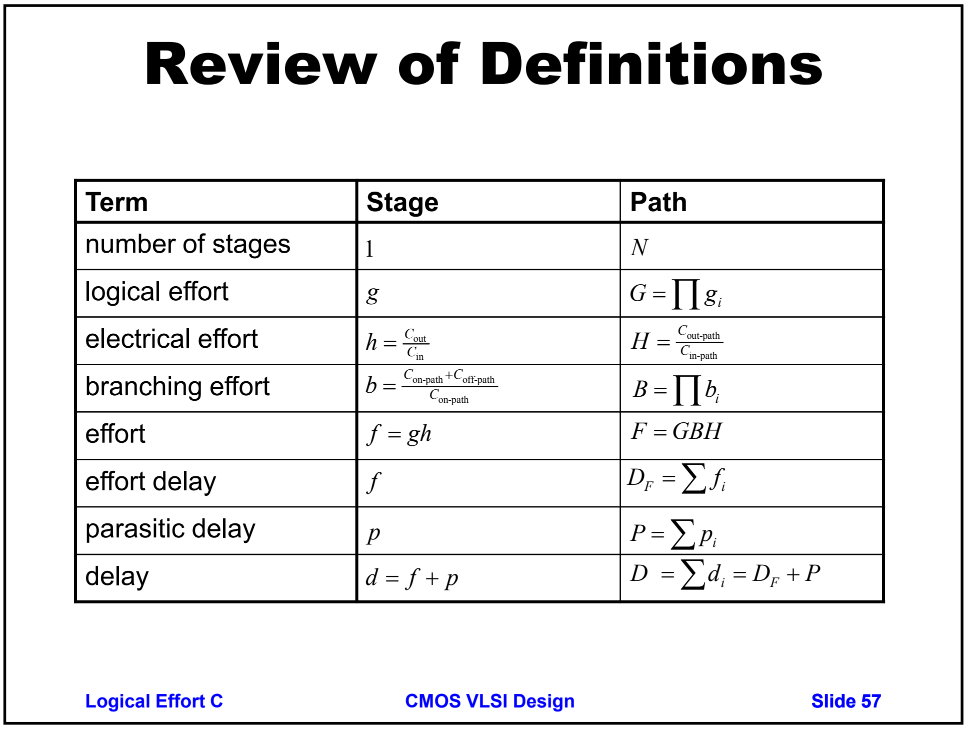
Inverter=>NAND=>Inverter

Comparison

- Compare many alternatives with a spreadsheet

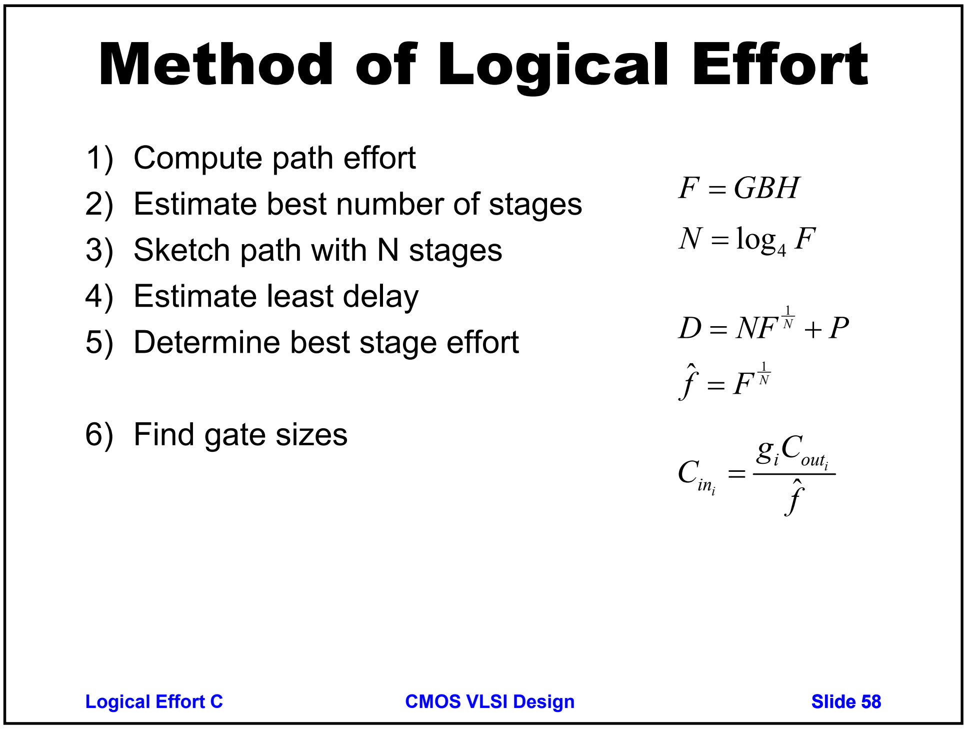
Design	N	G	P	D
NAND4-INV	2	2	5	29.8
NAND2-NOR2	2	20/9	4	30.1
INV-NAND4-INV	3	2	6	22.1
NAND4-INV-INV-INV	4	2	7	21.1
NAND2-NOR2-INV-INV	4	20/9	6	20.5
NAND2-INV-NAND2-INV	4	16/9	6	19.7
INV-NAND2-INV-NAND2-INV	5	16/9	7	20.4
NAND2-INV-NAND2-INV-INV-INV	6	16/9	8	21.6

Fastest



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
2) Estimate best number of stages
3) Sketch path with N stages
4) Estimate least delay
5) Determine best stage effort

6) Find gate sizes

$$F = GBH$$

$$N = \log_4 F$$

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

$$\hat{f} = F^{\frac{1}{N}}$$

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

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
- ❑ 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