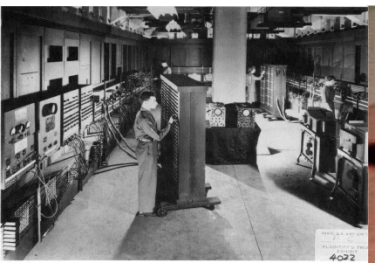
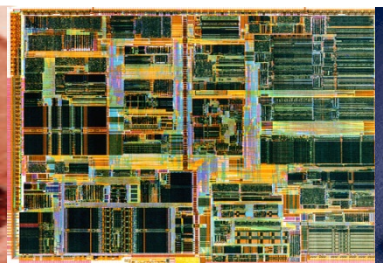
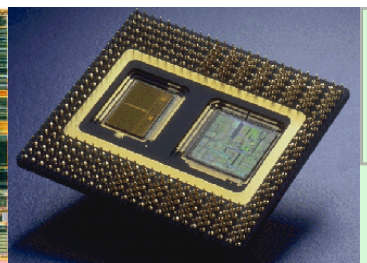
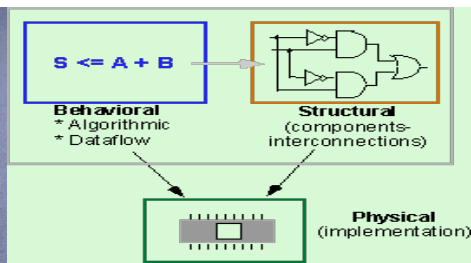
CPE 690: Introduction to VLSI Design

Lecture 10
Logical Effort and Multi-stage Logic Networks

Bryan Ackland
Department of Electrical and Computer Engineering
Stevens Institute of Technology
Hoboken, NJ 07030

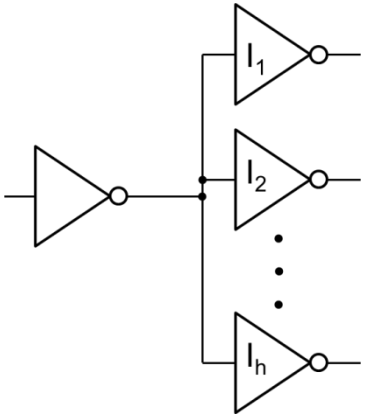
Adapted from Lecture Notes, David Mahoney Harris CMOS VLSI Design

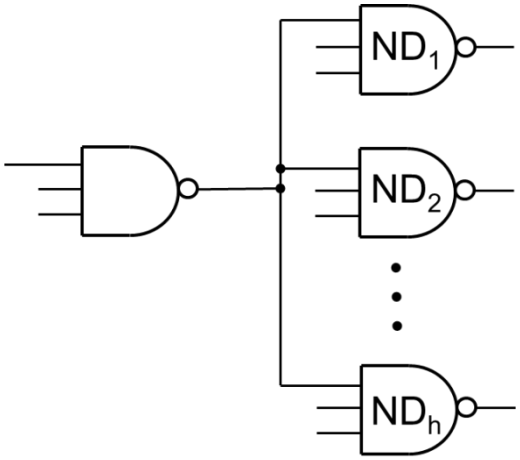


Linear Delay Model

- So far, we have seen how to estimate the delay of a gate in terms of its topology and fanout:





delay = (3+3h)RC *delay = (9+5h)RC*

or, in units of τ = delay of unloaded, unit size inverter (τ = 3RC):

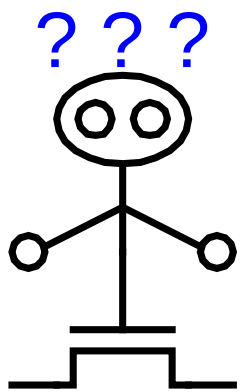
(~3ps in 65nm process, ~60 ps in 0.6μ process)

$d_{inv} = 1+h$ $d_{nand3} = 3+(5/3)h$

- How can we use this to optimize a network of gates?

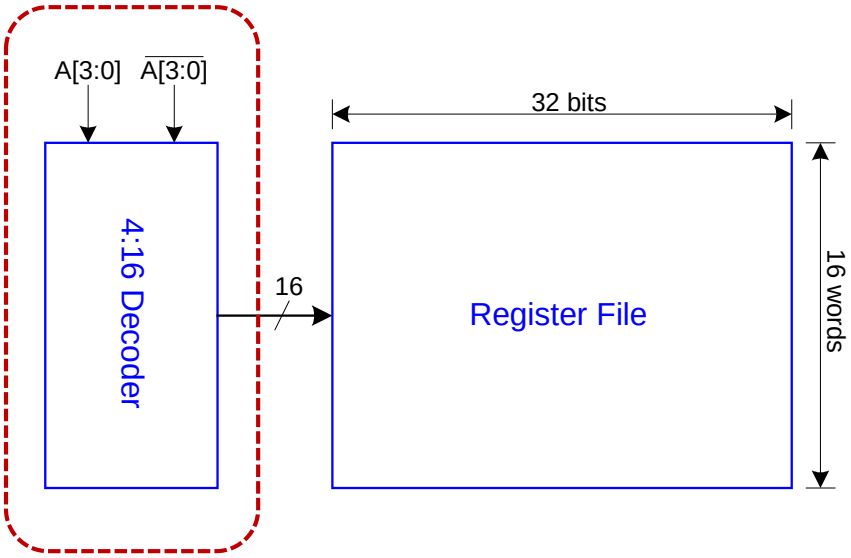
Logical Effort

- 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?
- Simulating or analyzing all the alternatives is impractical
- Logical effort is a method to make these decisions
  - Builds on our simple linear (RC) model of delay
  - Allows back-of-the-envelope calculations
  - Helps make rapid comparisons between alternatives
  - Provides intuitive understanding of network delay



Example

Suppose we need to design an address 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 (address) input may present a load of 10 unit-sized transistors
- We need to decide:
  - How many stages to use?
  - How large should each gate be?
  - How fast can decoder operate?

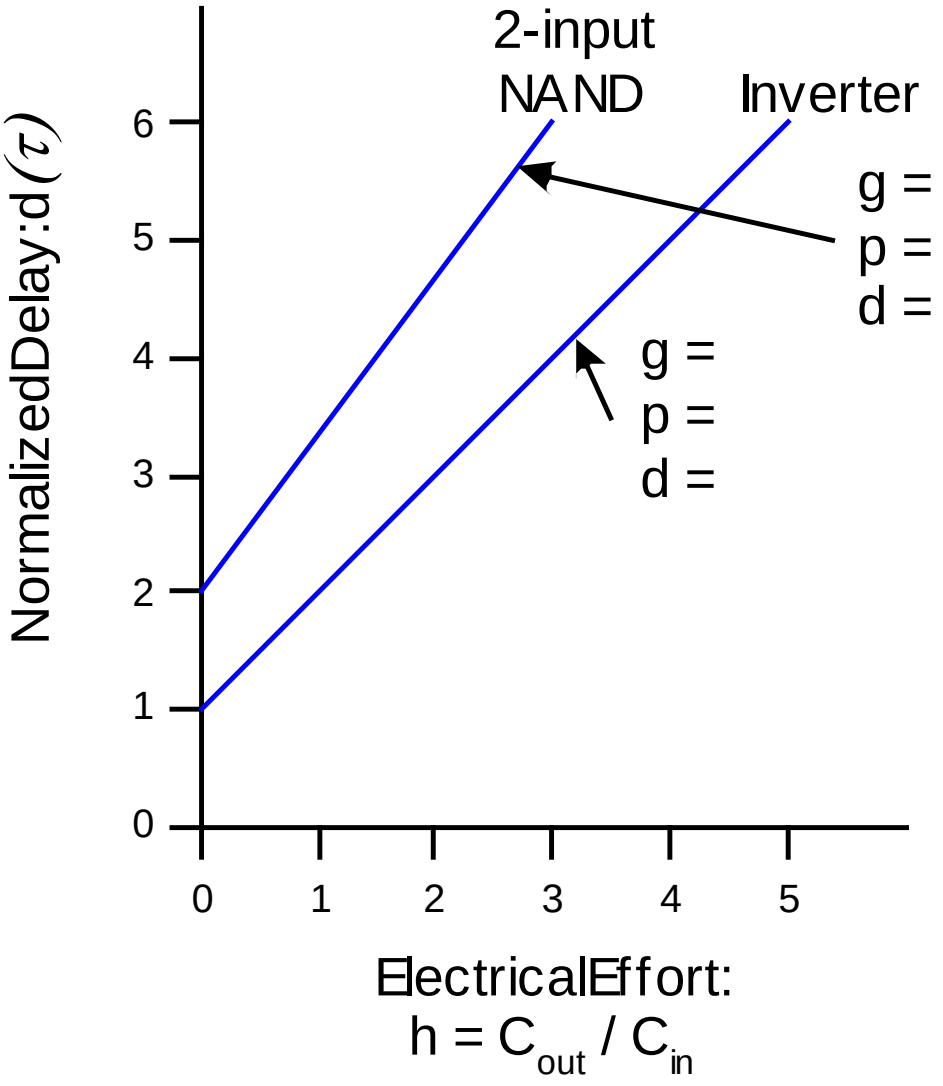
Delay in a Logic Gate

$d_{inv} = 1+h$ *(in units of* τ*)* $\qquad$ $d_{nand2} = 2+(4/3)h$

- Gate delay takes the form: $d = p + g.h = p + f$
- p: parasitic delay
  - Represents delay of gate driving no load (doing no useful work)
  - Set by internal parasitic capacitance
- f: *effort delay* = $g.h$ (a.k.a. stage effort)
  - Result of doing useful work (driving other gates)
- g: *logical effort*
  - Measures complexity of a gate (=1 for inverter)
  - input capacitance (*relative to inverter*) to deliver unit output current
- h: *electrical effort* = C_{out} / C_{in}
  - Ratio of output (load) capacitance to input capacitance
  - Sometimes called fanout

Delay Plots

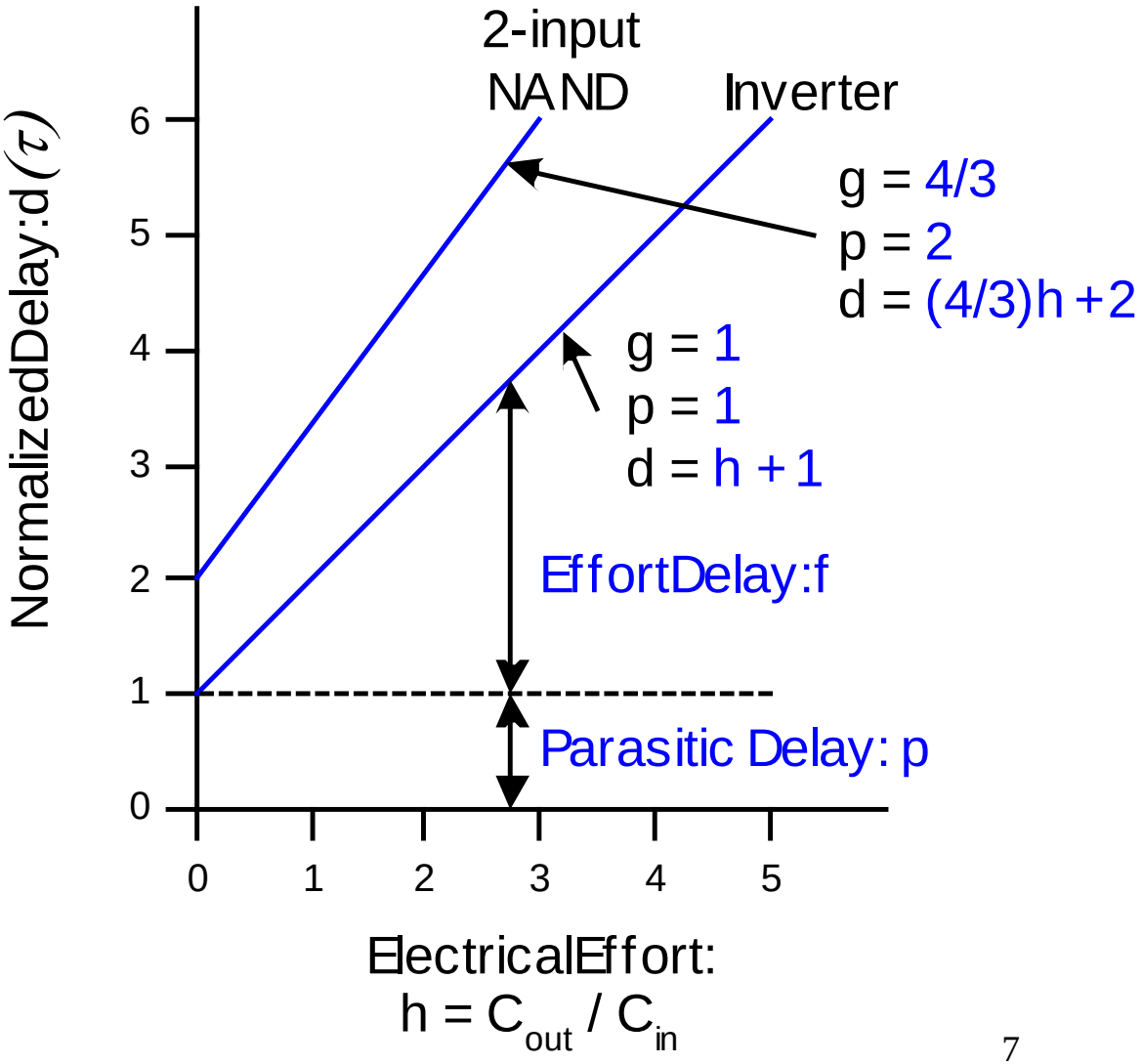
$$d = f + p$$
$$= g.h + p$$



Delay Plots

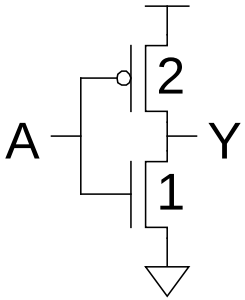
$$d = f + p$$
$$= g.h + p$$

What about NOR2?

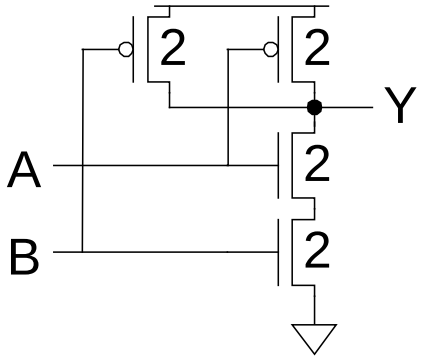


Logical Effort

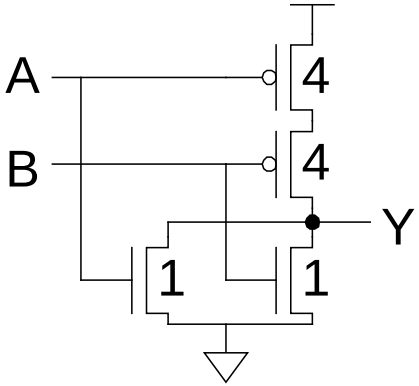
- *Logical effort of a gate is the ratio of the input capacitance of a gate to the input capacitance of an inverter delivering the same output current*.
- Property of topology of the gate
- Independent of “size” of the gate
- Can measure from delay vs. fanout plots
- *Or* estimate by counting transistor widths



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



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



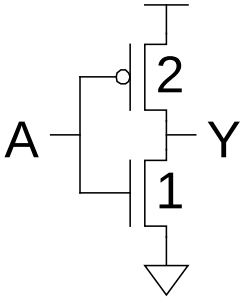
$C_{in} = 5C$
$g = 5/3$

Logical Effort of Common Gates

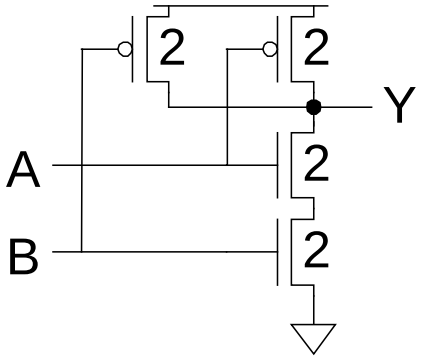
Logical Effort *g*	*Number of inputs*				
	1	2	3	4	n
Inverter	1				
NAND		4/3	5/3	6/3	(n+2)/3
NOR		5/3	7/3	9/3	(2n+1)/3
Tristate / mux	2	2	2	2	2
XOR, XNOR		4, 4	6, 12, 6	8, 16, 16, 8	

Parasitic Delay

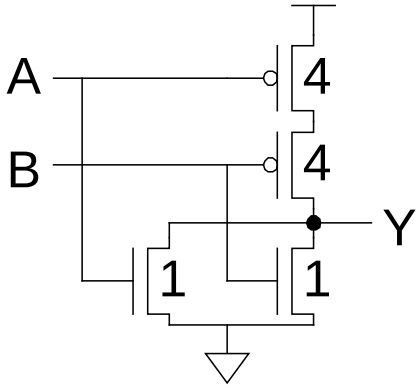
- *Parasitic delay of a gate is the delay of the gate when it drives zero load*
- Delay due to internal diffusion capacitance
- Property of topology of the gate
- Independent of "size" of the gate
- Usually just count diffusion capacitance on output node
  - but beware of high fan-in gates



$C_{diff} = 3C$
$p = 3/3$



$C_{diff} = 6C$
$p = 2$



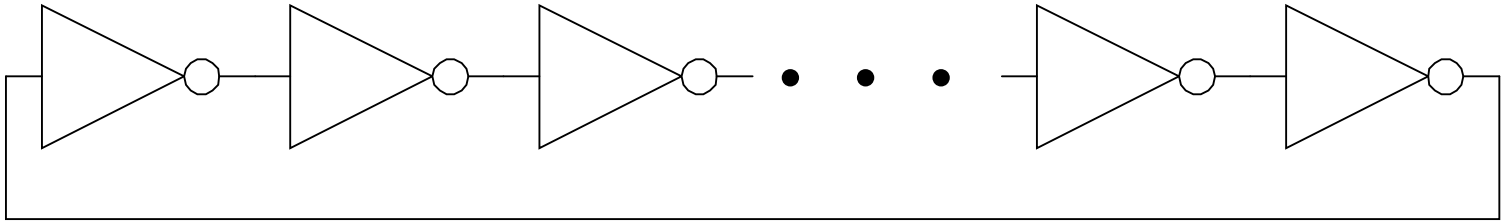
$C_{diff} = 6C$
$p = 2$

Parasitic Delay of Common Gates

Parasitic Delay *p*	*Number of inputs*				
	1	2	3	4	n
Inverter	1				
NAND		2	3	4	n
NAND (Elmore)		7/3	4	6	$(n^2+5n)/6$
NOR		2	3	4	n
Tristate / mux	2	4	6	8	2n
XOR, XNOR		4	6	8	

Example: Ring Oscillator

- Estimate the frequency of an N-stage ring oscillator:



Logical Effort: $g = 1$

Electrical Effort: $h = 1$

Parasitic Delay: $p = 1$

Stage Delay: $d = g.h+p = 2$

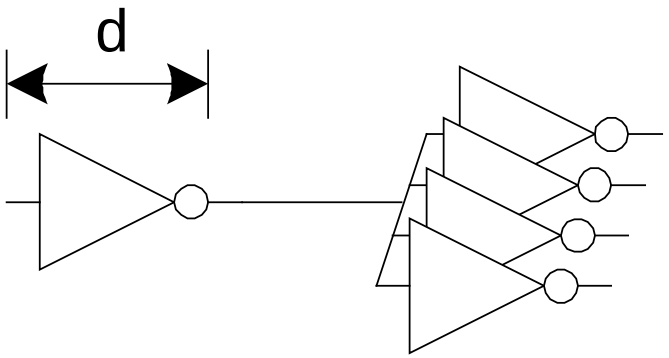
Frequency: $f_{osc} = 1/(2*N*d) = 1/(4N.\tau)$

31 stage ring oscillator in 0.6 μm process: τ = 60ps, f = 135 MHz

65nm process: τ = 3ps, f = 2.7GHz

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 = g.h+p = 5

The FO4 delay is about

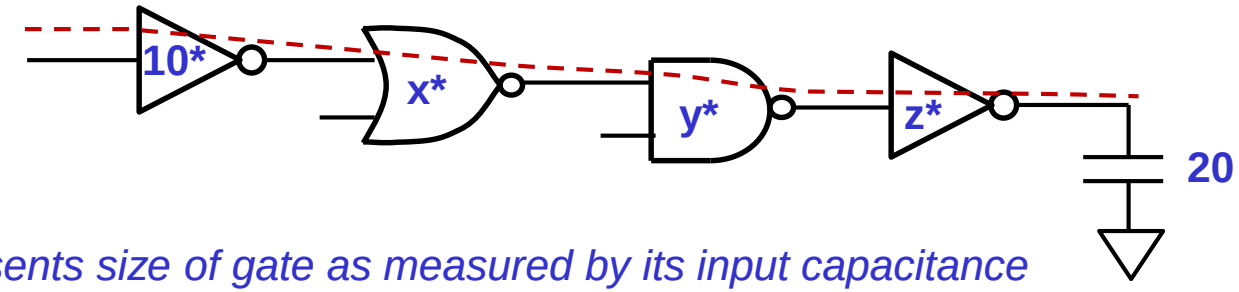
300 ps in 0.6 μm process

60 ps in a 180 nm process

15 ps in an *65* nm process

Multistage Logic Networks

- What if we have a multi-stage path with
  - a mix of different gate types along the path
  - a known load capacitance at the end of the path
  - a limitation on the input capacitance to the path

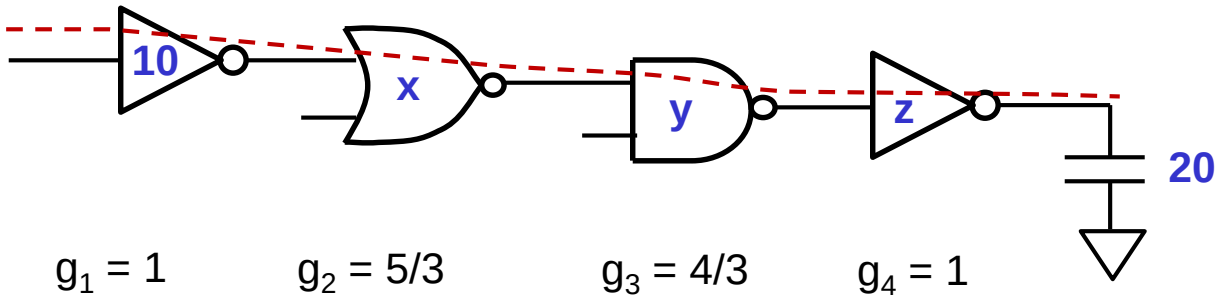


** represents size of gate as measured by its input capacitance*

- How can we size these gates to minimize the delay?
- We generalize the concept of logical effort from single gate to a multistage path

Path Effort

- *Define Path Logical Effort* $G = \prod g_i$ = 20/9
- *Define Path Electrical Effort* $H = \frac{C_{\text{out-path}}}{C_{\text{in-path}}}$ = 20/10 = 2
- *Define Path Effort* $F = \prod f_i = \prod g_i h_i$ = 40/9

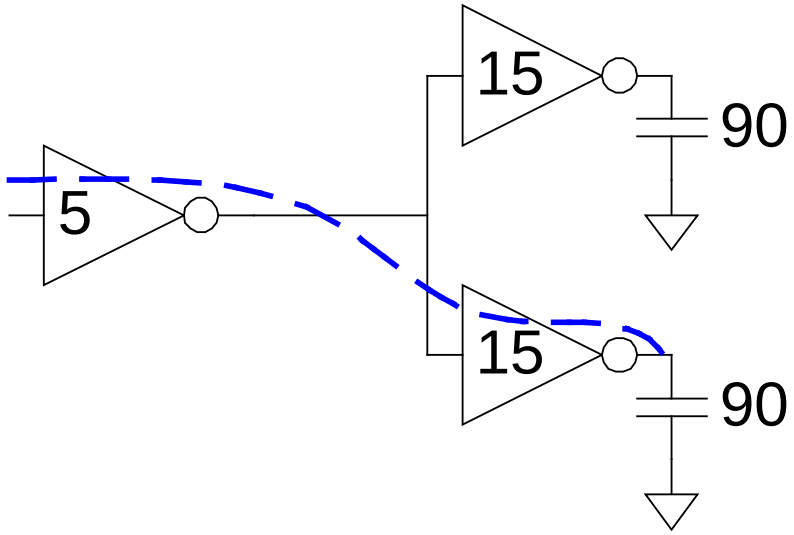


$g_1 = 1$	$g_2 = 5/3$	$g_3 = 4/3$	$g_4 = 1$
$h_1 = x/10$	$h_2 = y/x$	$h_3 = z/y$	$h_4 = 20/z$
$f_1 = x/10$	$f_2 = 5y/3x$	$f_3 = 4z/3y$	$f_4 = 20/z$

- Can we write F = G.H ?

Paths that Branch

- No! Consider paths that branch:



G = 1

H = 90 / 5 = 18

G.H = 18

h_1 = (15 +15) / 5 = 6

h_2 = 90 / 15 = 6

F = $(g_1.h_1).(g_2.h_2)$ = 36 = 2G.H

Branching Effort

- Introduce *branching effort*
  - Accounts for branching between stages in path
  - Define $b_i = \dfrac{C_{out:on\_path} + C_{out:off\_path}}{C_{out:on\_path}}$
  - Define path branching effort $B = \prod b_i$

- Now we compute the path effort

  F = G.B.H

Note:

$$\prod h_i = BH$$

Paths that Branch

- Now, including branching effort:

$G = 1$

$H = 90 / 5 = 18$

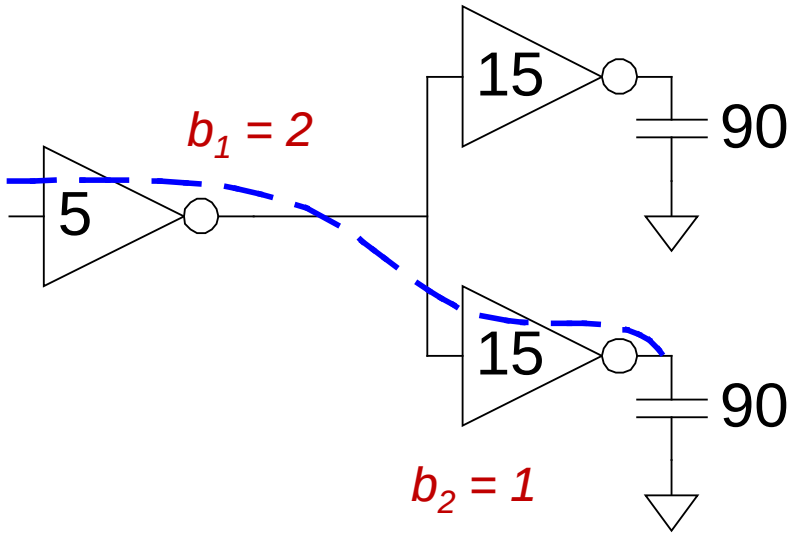
$B = 2.1 = 2$

$B.G.H = 36$

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

$h_2 = 90 / 15 = 6$

$F = (g_1.h_1).(g_2.h_2) = 36 = B.G.H$



Multistage Delays

- Now, remember that delay of each gate is:

  $d = g.h + p = f + p$

- So, total path delay is:

$$D = \sum_i (f_i + p_i)$$

- Define Path Effort Delay: $D_F = \sum f_i$

- Define Path Parasitic Delay: $P = \sum p_i$

- Path Delay $D = \sum d_i = D_F + P$

Minimizing Path Delay

- We want to set the size of the individual gates along the path so as to minimize:

$$D = \sum d_i = D_F + P$$

- Remember P is independent of gate size.

- We want to minimize $\sum_i f_i$ given $\prod_i f_i = F$ *(a constant)*

- 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

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

Minimum Path Delay

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

This is the key result of logical effort:

- Find fastest delay doesn't require calculating gate sizes
- Can estimate *D* knowing :
  - number of stages (N)
  - path effort (depends on stage topologies)
  - parasitic gate delays (depends on stage topologies)
- Compare to optimization by simulation

Setting Gate Sizes

- How wide should the gates be for minimum delay?

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

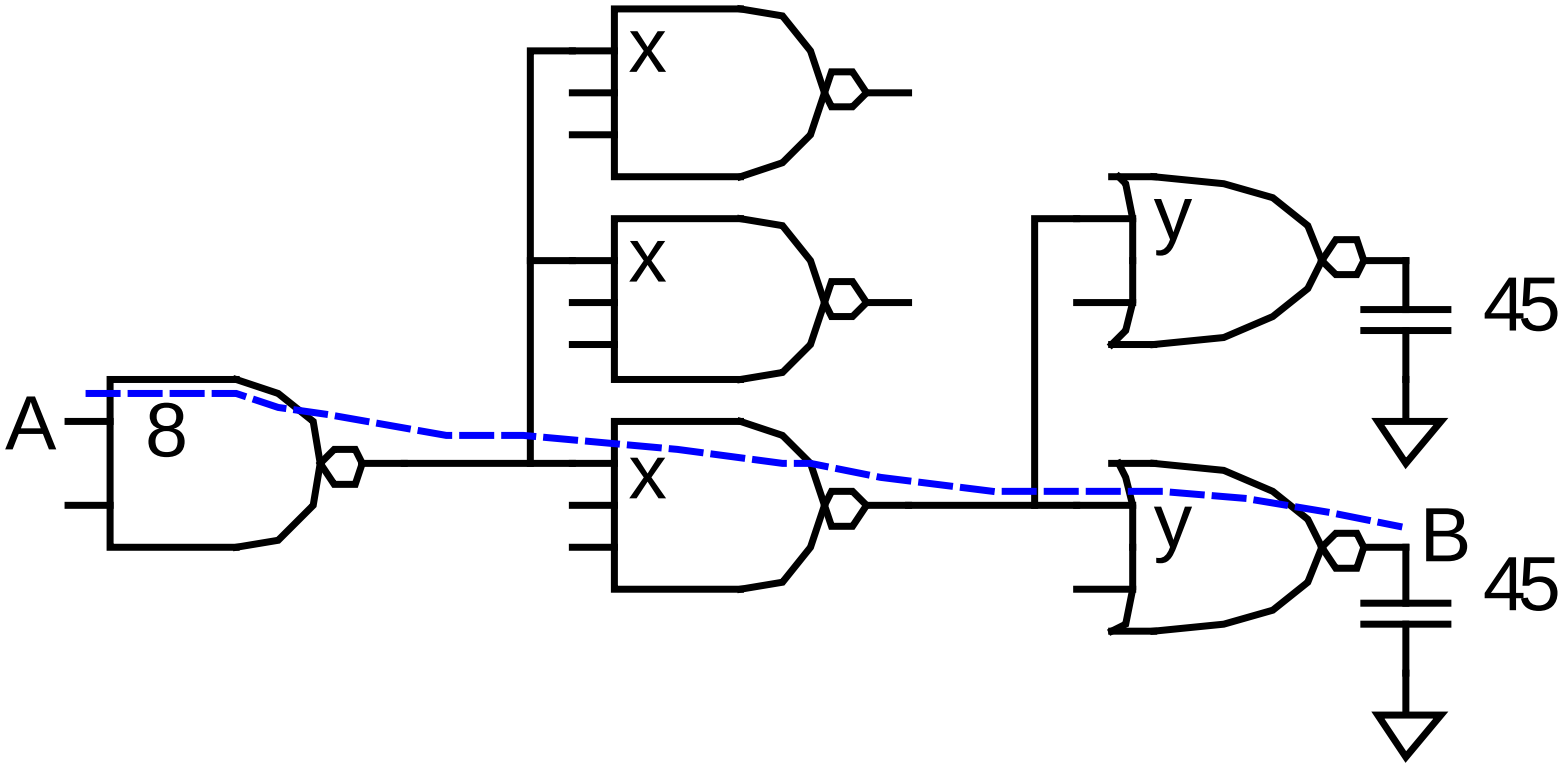
$$\Rightarrow C_{in_i} = \frac{g_i C_{out_i}}{\hat{f}}$$

- Working backward, apply capacitance transformation to find input capacitance of each gate given load it drives.
  - Input capacitance determines gate size

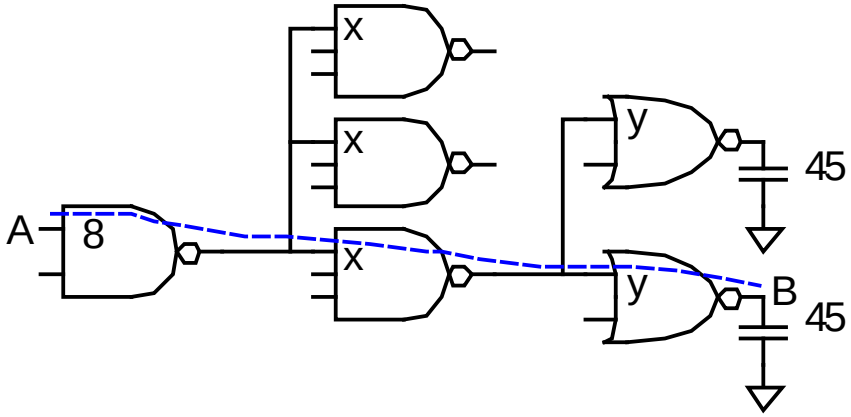
- Check work by verifying input cap spec is met.

Example: 3-stage path

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



Example: Minimum Path Delay



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 = G.B.H = 125
Best Stage Effort	$\hat{f} = \sqrt[3]{F} = 5$
Parasitic Delay	P = 2+3+2 = 7
Minimum Delay	D = 3*5 + 7 = 22 τ (=4.4 FO4)

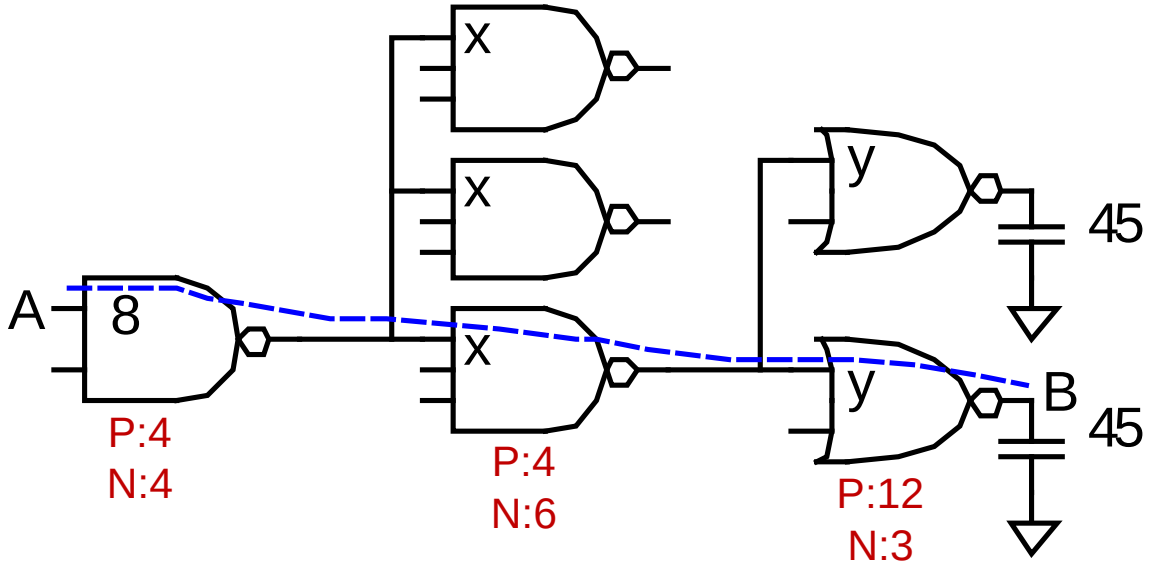
Example: Calculating Gate Sizes

- Work backward for sizes: $C_{in_i} = \frac{g_i C_{out_i}}{\hat{f}}$

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

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

A_{load} = ((4/3)*(10+10+10))/5 = 8 ✔ (agrees with original spec.)



transistor widths
assume C in units of C_g

Optimizing Number of Stages

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

$G = 1$

$H = 64$

$B = 1$

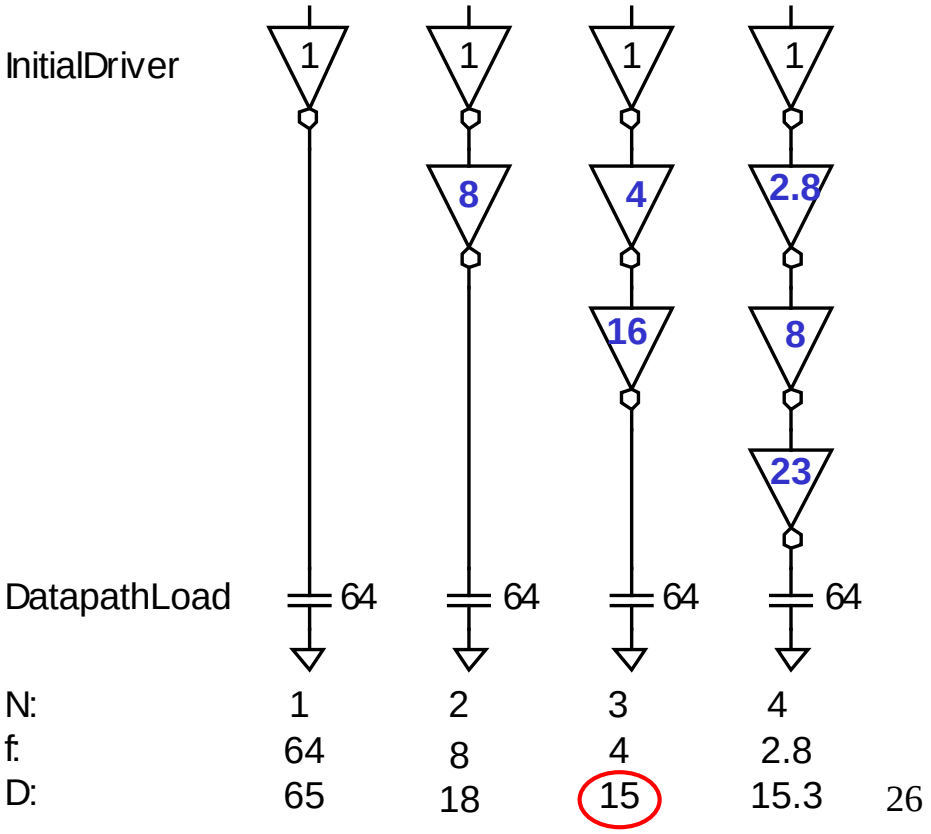
$F = 64$

$P = N$

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

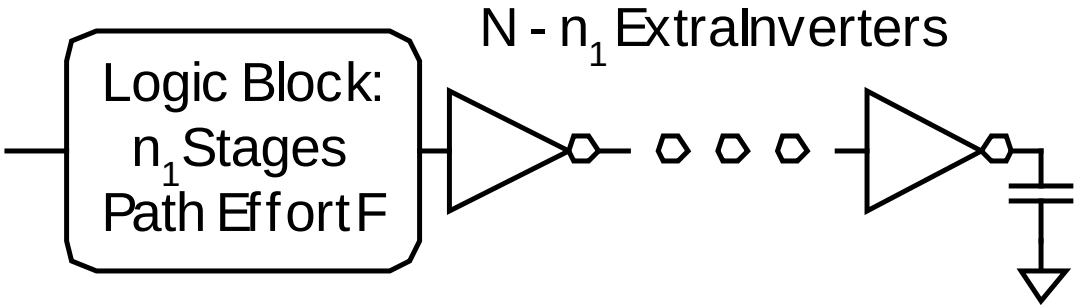
$D = NF^{1/N} + P$

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



Adding Stages to Logical Function

- Separate out optimization of logic function from the optimization of number of stages by adding extra inverters to output



- Add (N - n_1) inverters to output to create N-stage network
- Extra inverters do not change path logic effort *F*, but they do add parasitic delay
- Optimum delay of extended path is:

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

Optimizing Extended Path

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

- Differentiate with respect to N and set to zero to determine optimum value for N:

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

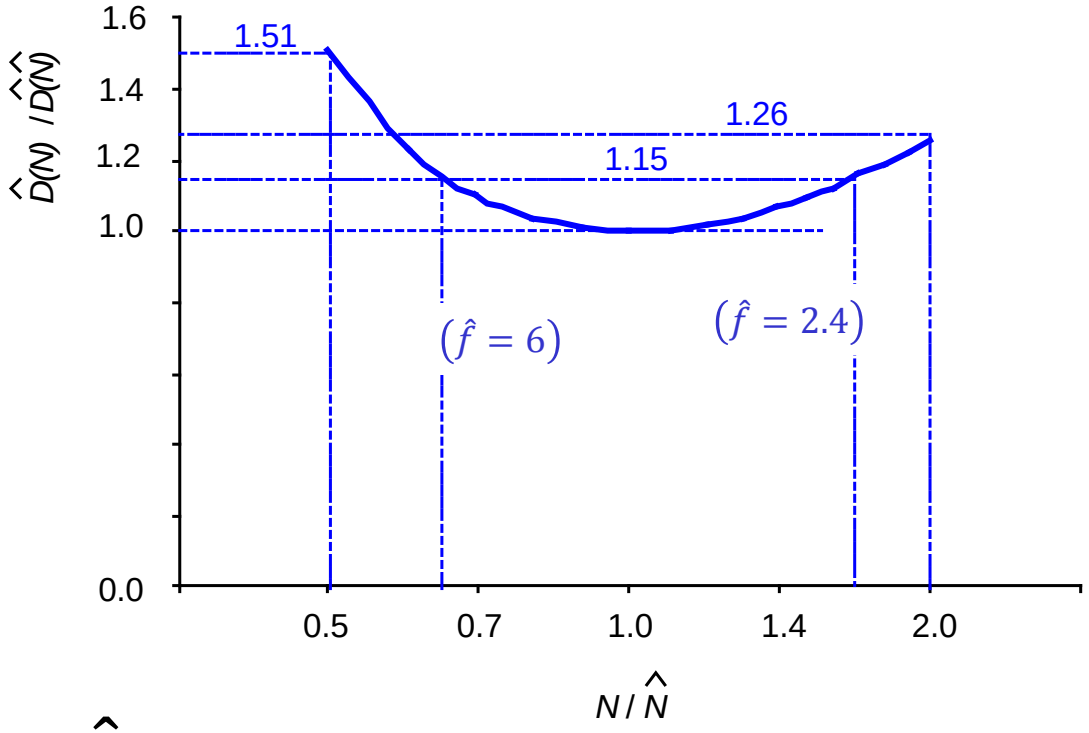
- Define optimal stage effort $\hat{f}(N) = F^{\frac{1}{N}}$ (note $\hat{f} \equiv \rho$ in textbook)
  - then delay minimized when:

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

- No closed form solution for $\hat{f}$ as a function of p_{inv}
- Neglecting parasitics ($p_{inv} = 0$), we find $\hat{f} = 2.718\ (e)$
- For $p_{inv} = 1$, solve numerically for $\hat{f} = 3.59$

Sensitivity Analysis

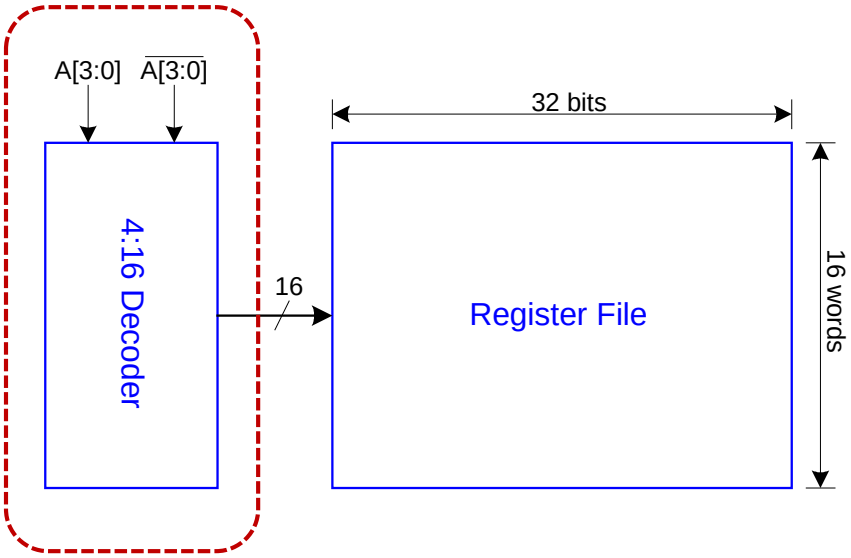
- How sensitive is delay to using exactly the best number of stages? Fortunately, not very.



- $2.4 < \hat{f} < 6$ gives delay within 15% of optimal
  - Many designers use $\hat{f} = 4$ as the standard stage effort
  - Hence importance of FO4 Inverter as a representative logic gate delay

Revisit Decode Design Example

Suppose we need to design an address decoder for a register file:

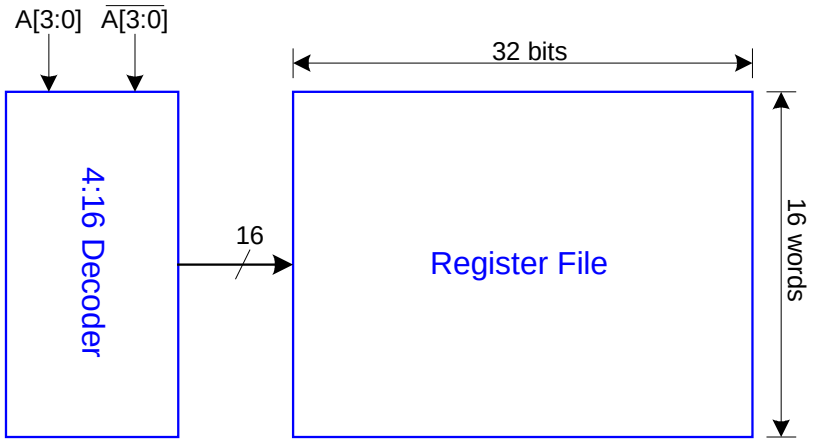


- Decoder specifications:
  - 16 word register file
  - Each word is 32 bits wide
  - Each bit presents load of 3 unit-sized transistors
  - True and complementary address inputs A[3:0]
  - Each input may present a load of 10 unit-sized transistors

- We need to decide:
  - How many stages to use?
  - How large should each gate be?
  - How fast can decoder operate?

Estimate Number of Stages

- Decoder specifications:
  - 16 word register file
  - Each word is 32 bits wide
  - Each bit presents load of 3 unit-sized transistors
  - Each input may drive 10 unit-sized transistors



$H = (3 \times 32)/10 = 9.6$

Each input address line drives 8 decoder gates, so

$B = 8$

There will be something like NAND4 to decode each address, so assume

$G = 2$

then, path effort $F = G.B.H = 153.6$

assuming $\hat{f} = 4$, would give $N = \log_4(F) = \log_4(153.6) = 3.63$

Let's try a 3-stage design!

Complete the Design

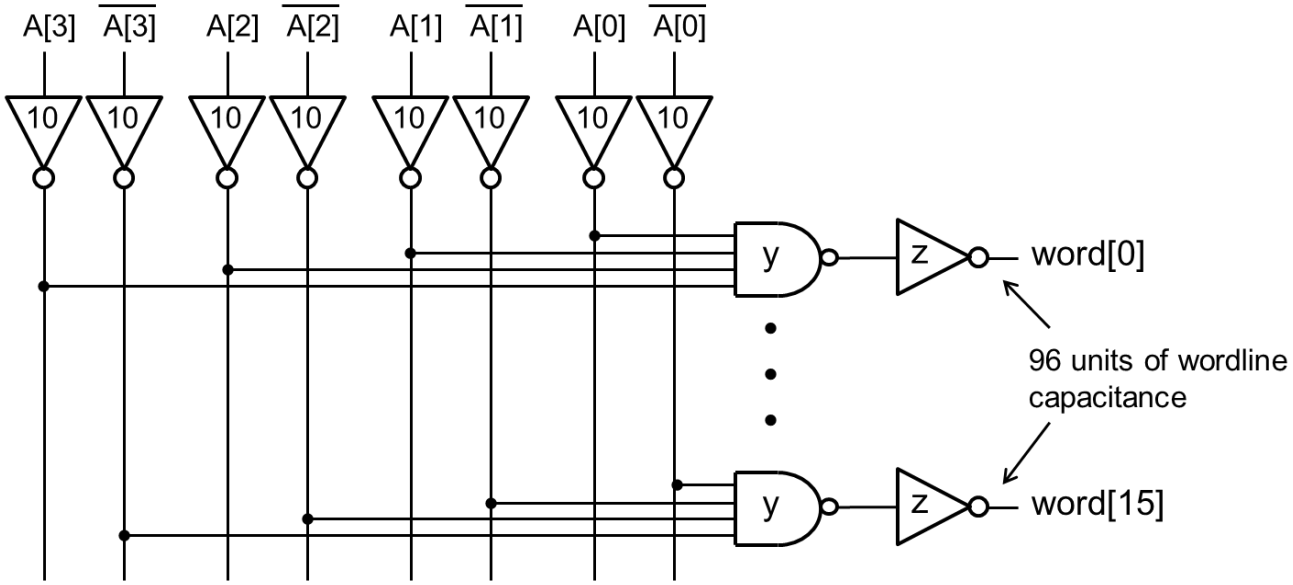
Logical Effort: G = 1 * 6/3 * 1 = 2

Path Effort: F = GBH = 153.6

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



Compare Alternative Solutions

- Compare many alternatives with a spreadsheet
- $D = N(76.8\ G)^{1/N} + P$

Design	N	G	P	D
NOR4	1	3	4	234
NAND4-INV	2	2	5	29.8
NAND2-NOR2	2	20/9	4	30.1
INV-NAND4-INV	3	2	6	22.1
NAND4-INV-INV-INV	4	2	7	21.1
NAND2-NOR2-INV-INV	4	20/9	6	20.5
NAND2-INV-NAND2-INV	4	16/9	6	19.7
INV-NAND2-INV-NAND2-INV	5	16/9	7	20.4
NAND2-INV-NAND2-INV-INV-INV	6	16/9	8	21.6

Review of Definitions

Term	Stage	Path
number of stages	1	N
logical effort	g	$G = \prod g_i$
electrical effort	$h = \frac{C_{\text{out}}}{C_{\text{in}}}$	$H = \frac{C_{\text{out-path}}}{C_{\text{in-path}}}$
branching effort	$b = \frac{C_{\text{on-path}} + C_{\text{off-path}}}{C_{\text{on-path}}}$	$B = \prod b_i$
effort	$f = gh$	$F = GBH$
effort delay	f	$D_F = \sum f_i$
parasitic delay	p	$P = \sum p_i$
delay	$d = f + p$	$D = \sum d_i = D_F + P$

Review Method of Logical Effort

1) Compute path effort $F = GBH$

2) Estimate best number of stages $N = \log_4 F$

3) Sketch path with N stages

4) Estimate least delay $D = NF^{\frac{1}{N}} + P$

5) Determine best stage effort $\hat{f} = F^{\frac{1}{N}}$

6) Find gate sizes $C_{in_i} = \frac{g_i C_{out_i}}{\hat{f}}$

Limits of Logical Effort

- "Chicken and egg" problem
  - Need path to compute G
  - But don't know number of stages without G
- Simplistic delay model
  - Neglects input rise time effects
- Interconnect
  - Iteration required in designs with wire
- Maximum speed only
  - Not minimum area/power for constrained delay

Summary

- Logical effort is useful for thinking about 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
  - requires practice to master