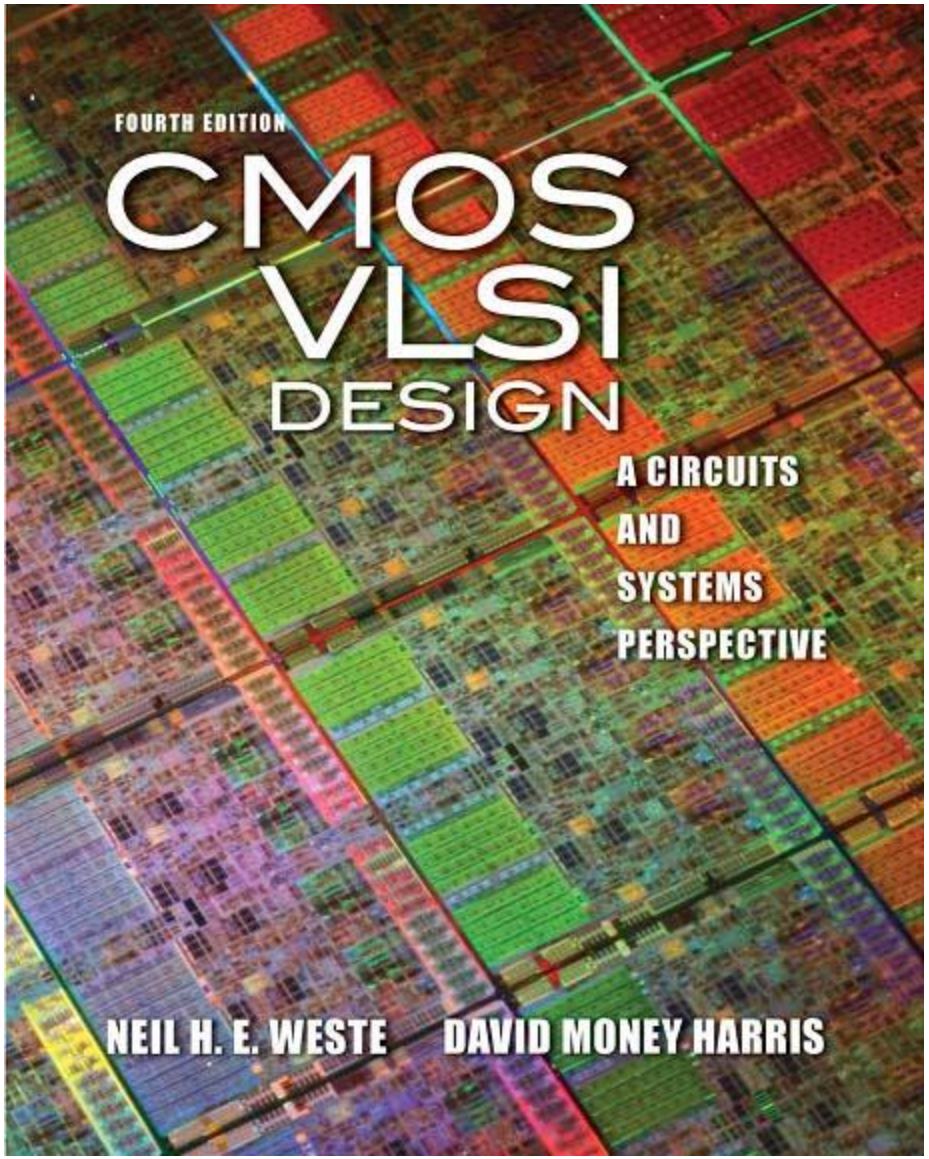


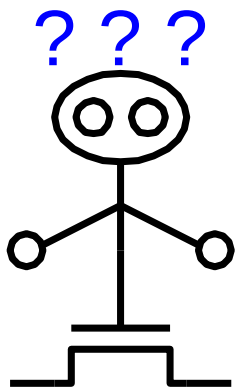
Lecture 6: Logical Effort

Outline

- Logical Effort
- Delay in a Logic Gate
- Multistage Logic Networks
- Choosing the Best Number of Stages
- Example
- Summary

Introduction

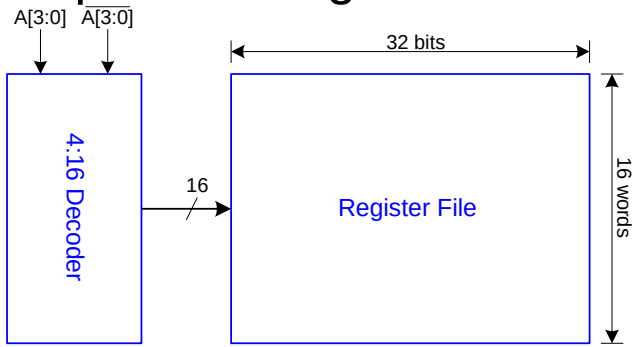
- Chip designers face a bewildering array of choices
  - What is the best circuit topology for a function?
  - How many stages of logic give least delay?
  - How wide should the transistors be?



- 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

Example

- 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?

Delay in a Logic Gate

- Express delays in process-independent unit
- Delay has two components: d = f + p
- *f*: *effort delay* = *gh* (a.k.a. stage effort)
  - Again has two components
- *g*: *logical effort*
  - Measures relative ability of gate to deliver current
  - $g \equiv 1$ for inverter
- *h*: *electrical effort* = C_{out} / C_{in}
  - Ratio of output to input capacitance
  - Sometimes called fanout
- *p*: parasitic delay
  - Represents delay of gate driving no load
  - Set by internal parasitic capacitance

$$d = \frac{d_{abs}}{\tau}$$

τ = 3RC

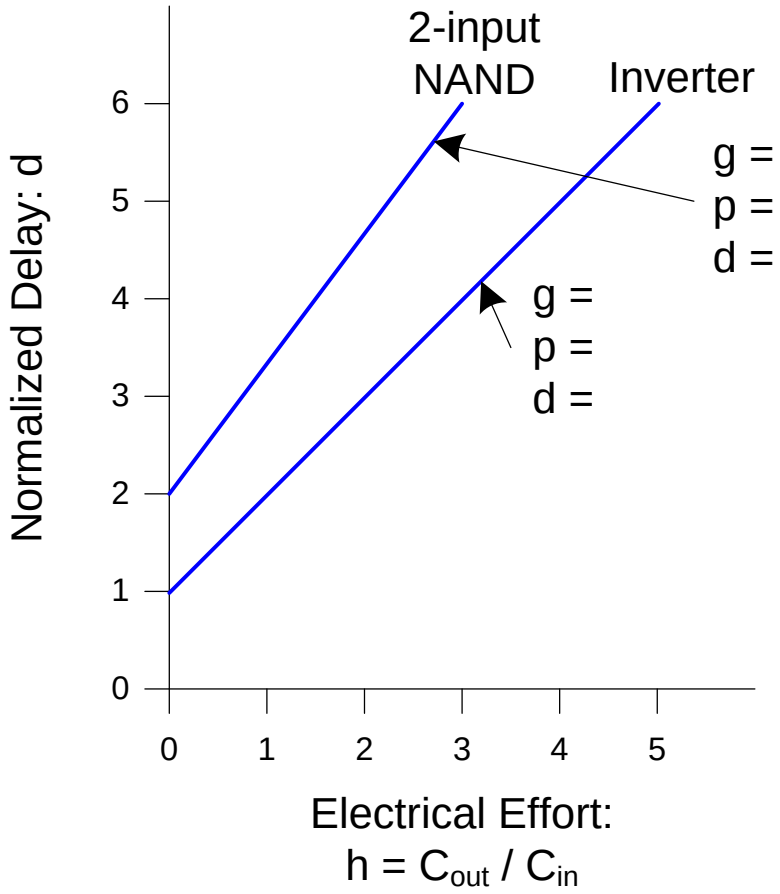
$\approx$ 3 ps in 65 nm process

60 ps in 0.6 μm process

Delay Plots

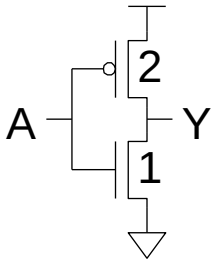
$d = f + p$

$= gh + p$

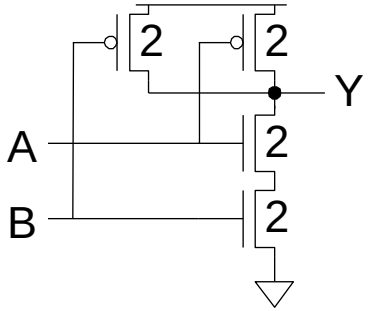


Computing Logical Effort

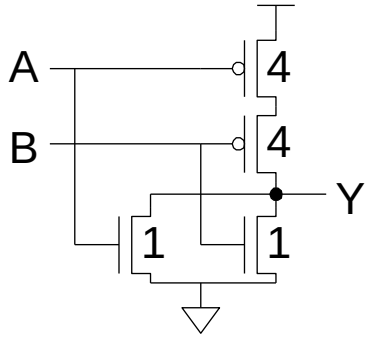
- DEF: *Logical effort is the ratio of the input capacitance of a gate to the input capacitance of an inverter delivering the same output current*.
- Measure from delay vs. fanout plots
- Or estimate by counting transistor widths



C_{in} = 3
g = 3/3



C_{in} = 4
g = 4/3



C_{in} = 5
g = 5/3

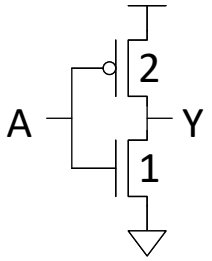
Catalog of Gates

- Logical effort of common gates

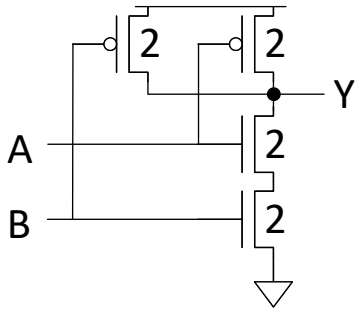
Gate type	Number of inputs				
	1	2	3	4	n
Inverter	1				
NAND		4/3	5/3	6/3	(n+2)/3
NOR		5/3	7/3	9/3	(2n+1)/3
Tristate / mux	2	2	2	2	2
XOR, XNOR		4, 4	6, 12, 6	8, 16, 16, 8	

Computing Parasitic Delay

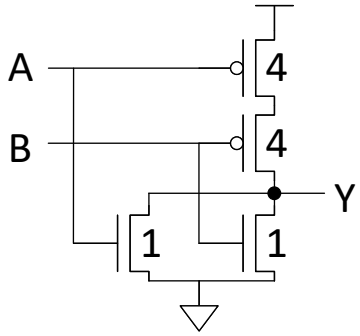
- DEF: The parasitic delay of a gate is the delay of the gate when it drives zero load.
- It can be estimated with Elmore RC delay models.
- An estimate is to count only diffusion capacitance on the output node (ignore internal node delays)



C_{out} = 3
p = 3/3



C_{out} = 6
p = 6/3



C_{out} = 6
p = 6/3

Catalog of Gates

- Parasitic delay of common gates
  - In multiples of p_{inv} (≈1)

Gate type	Number of inputs				
	1	2	3	4	n
Inverter	1				
NAND		2	3	4	n
NOR		2	3	4	n
Tristate / mux	2	4	6	8	2n
XOR, XNOR		4	6	8	

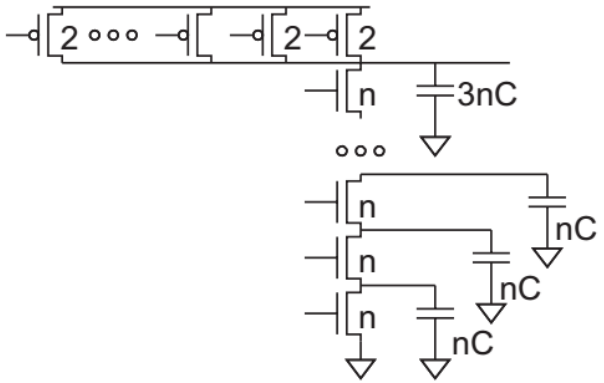
Parasitic Delay Note

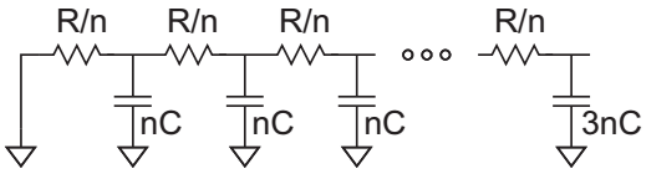
- This method of estimating parasitic delay is obviously crude.
- More refined estimates use the Elmore delay counting internal parasitics
- However, optimizing transistor size in a particular circuit is weakly dependent on the parasitic delay
- it is important to realize that parasitic delay grows more than linearly with the number of inputs in a real NAND or NOR circuit

Example: NAND Gate

- *n*-input NAND gate parasitic delay

$$t_{pd} = R(3nC) + \sum_{i=1}^{n-1}\left(\frac{iR}{n}\right)(nC) = \left(\frac{n^2}{2} + \frac{5}{2}n\right)RC \qquad (4.23)$$

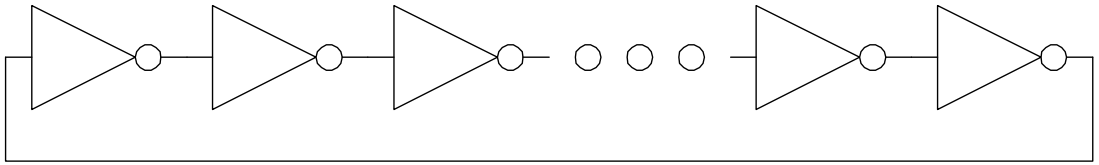




- In practice, it is rarely advisable to construct a gate with more than four to five series transistors.

Example: Ring Oscillator

- Estimate the frequency of an N-stage ring oscillator



Logical Effort: g =

Electrical Effort: h =

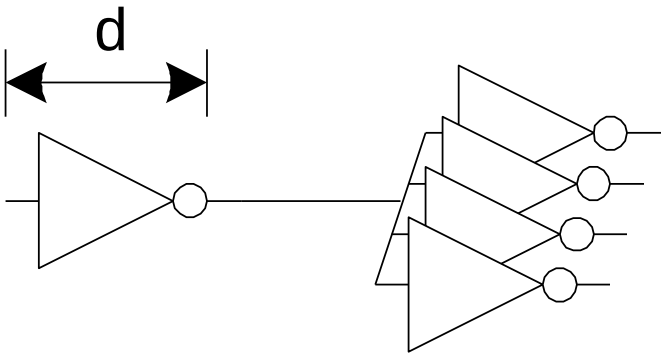
Parasitic Delay: p =

Stage Delay: d =

Frequency: f_{osc} =

Example: FO4 Inverter

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



Logical Effort: g =

Electrical Effort: h =

Parasitic Delay: p =

Stage Delay: d =

Drive

- A standard cell library contains multiple sizes of each common gate.
- The sizes are typically labeled with their drive.
- For example, a unit inverter may be called inv_1x. An inverter of eight times unit size is called inv_8x. A 2-input NAND that delivers the same current as the inverter is called nand2_1x.
- If we redefine a unit inverter to have one unit of input capacitance, then the drive of an arbitrary gate is:

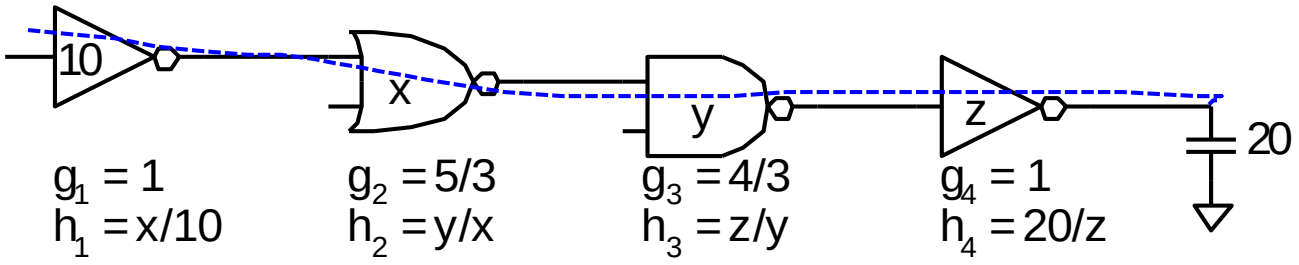
$$x = \frac{C_{in}}{g} \quad \rightarrow \quad d = \frac{C_{out}}{x} + p$$

Limitations to the Linear Delay Model

- Input and Output Slopes: inputs do not change abruptly
- Input arrival times: not all inputs arrive at the same instance
- Velocity saturation: Series transistors see less velocity saturation and hence have lower resistance than we estimate
- Voltage Dependence: delay will vary if the supply or threshold voltage is changed
- Gate-Source Capacitance: is not zero for internal nodes
- Bootstrapping: gate-drain capacitances couples input/output which increases the delay.

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 =

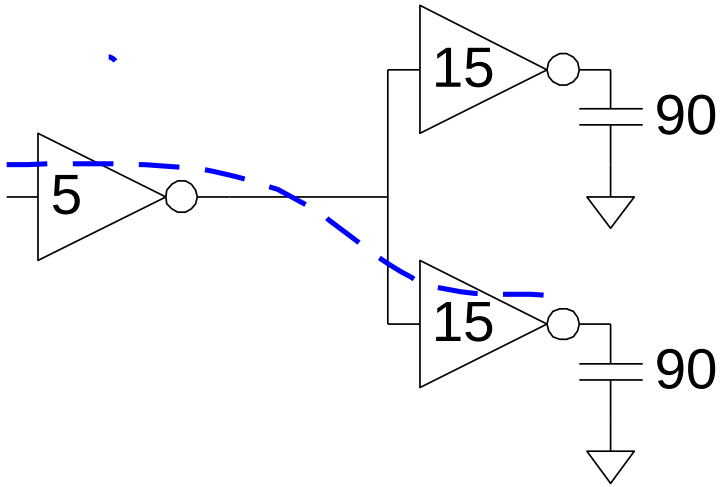
H =

GH =

h_1 =

h_2 =

F =



Branching Effort

- Introduce *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

- 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?

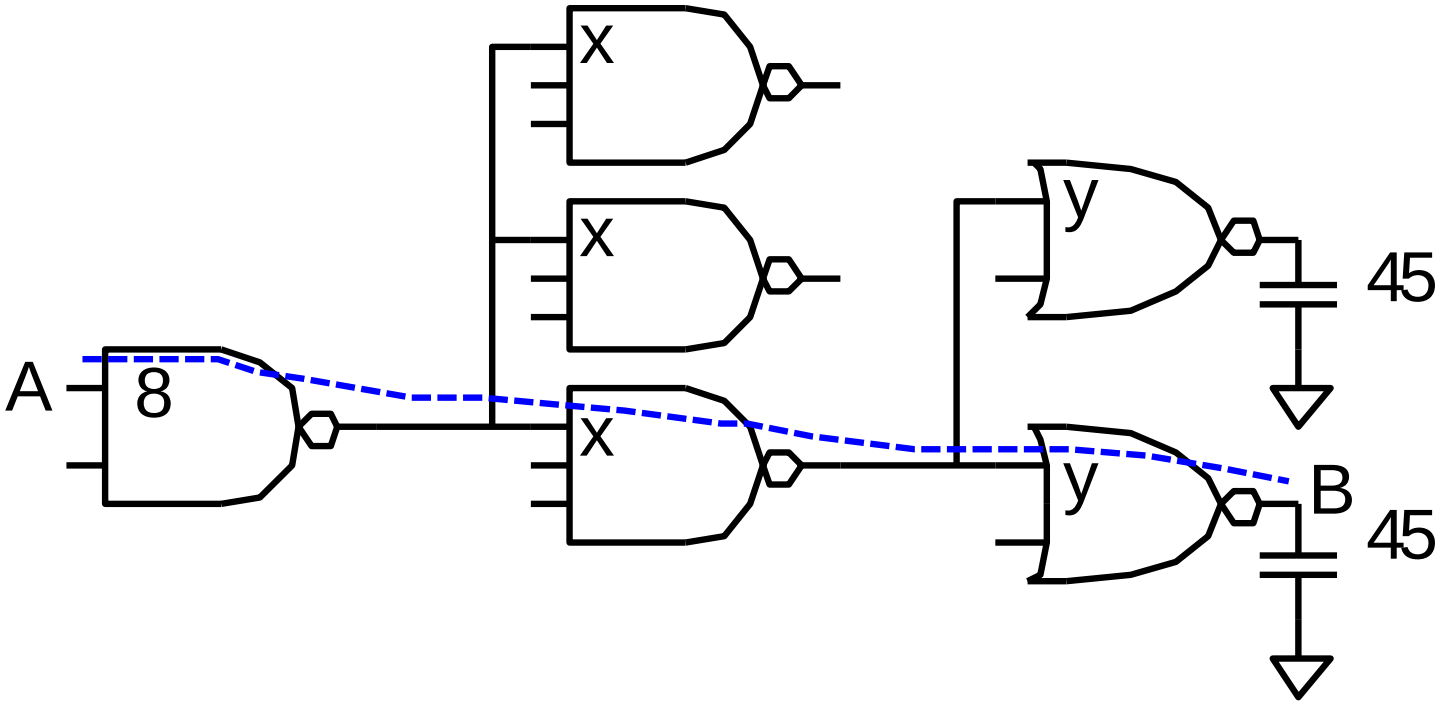
$$\hat{f} = gh = g\frac{C_{out}}{C_{in}}$$

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

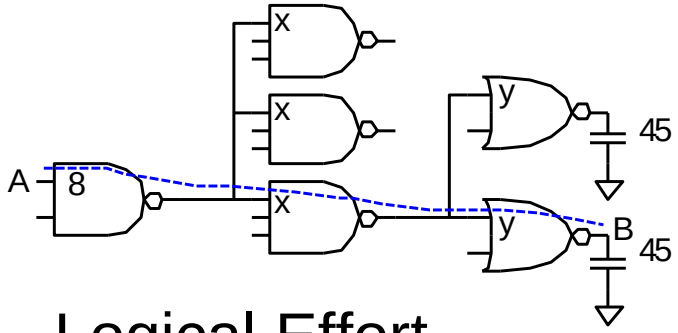
- Working backward, apply capacitance transformation to find input capacitance of each gate given load it drives.
- Check work by verifying input cap spec is met.

Example: 3-stage path

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



Example: 3-stage path



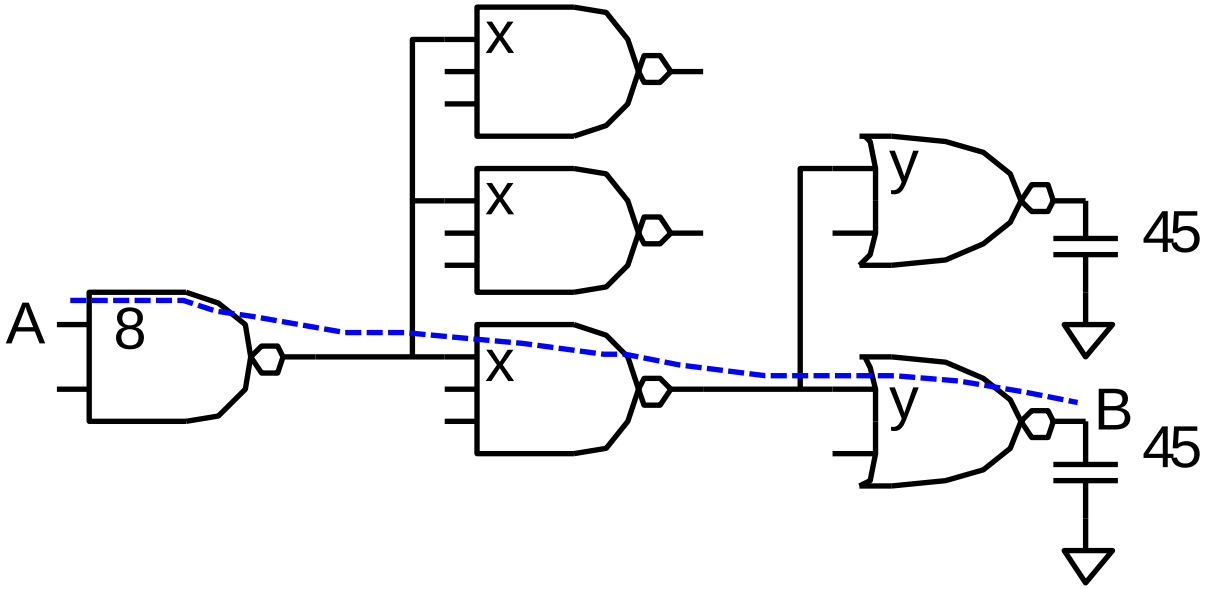
Logical Effort	G =
Electrical Effort	H =
Branching Effort	B =
Path Effort	F =
Best Stage Effort	$\hat{f} =$
Parasitic Delay	P =
Delay	D =

Example: 3-stage path

- Work backward for sizes

  y =

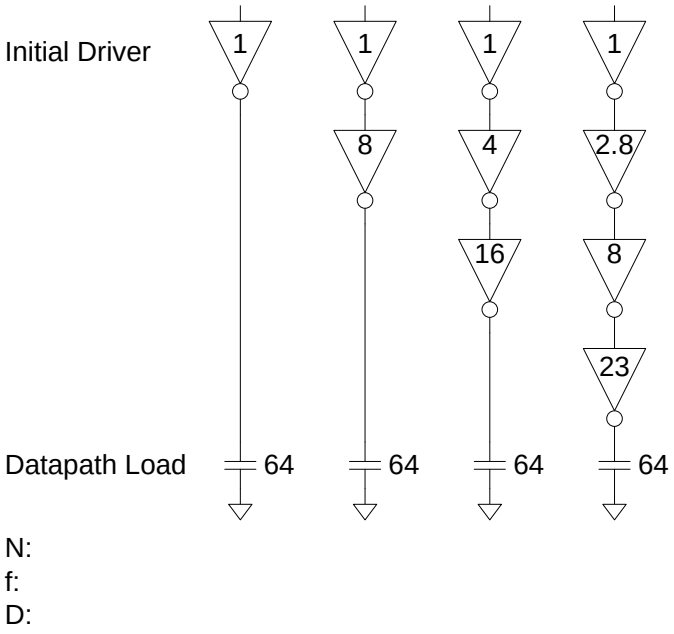
  x =



Best Number of Stages

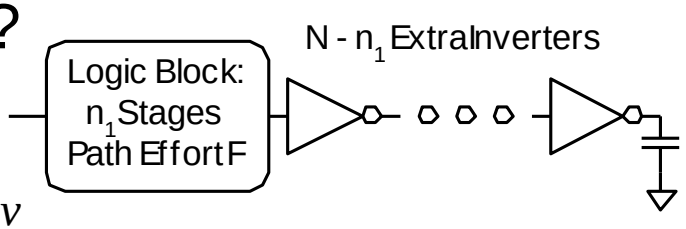
- How many stages should a path use?
  - Minimizing number of stages is not always fastest
- Example: drive 64-bit datapath with unit inverter

D =



Derivation

- Consider adding inverters to end of path
  - How many give least delay?



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

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

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

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

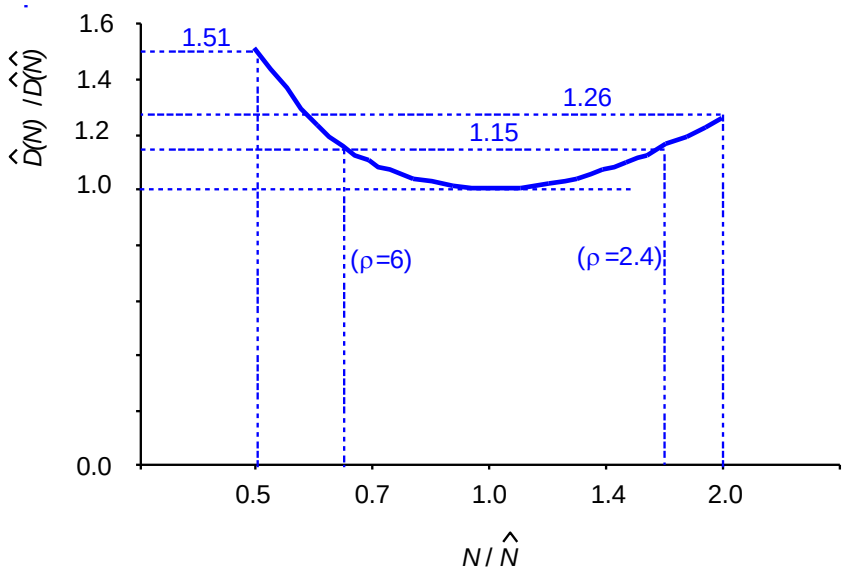
Best Stage Effort

- $p_{inv} + \rho(1 - \ln \rho) = 0$ has no closed-form solution

- Neglecting parasitics ($p_{inv} = 0$), we find $\rho = 2.718$ (e)
- For $p_{inv} = 1$, solve numerically for $\rho = 3.59$

Sensitivity Analysis

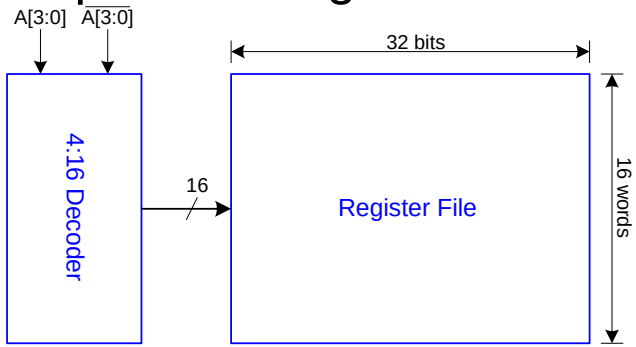
- How sensitive is delay to using exactly the best number of stages?



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

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?

Number of Stages

- Decoder effort is mainly electrical and branching

  Electrical Effort: H =

  Branching Effort: B =

- If we neglect logical effort (assume G = 1)

  Path Effort: F =

  Number of Stages: N =

- Try a -stage design

Gate Sizes & Delay

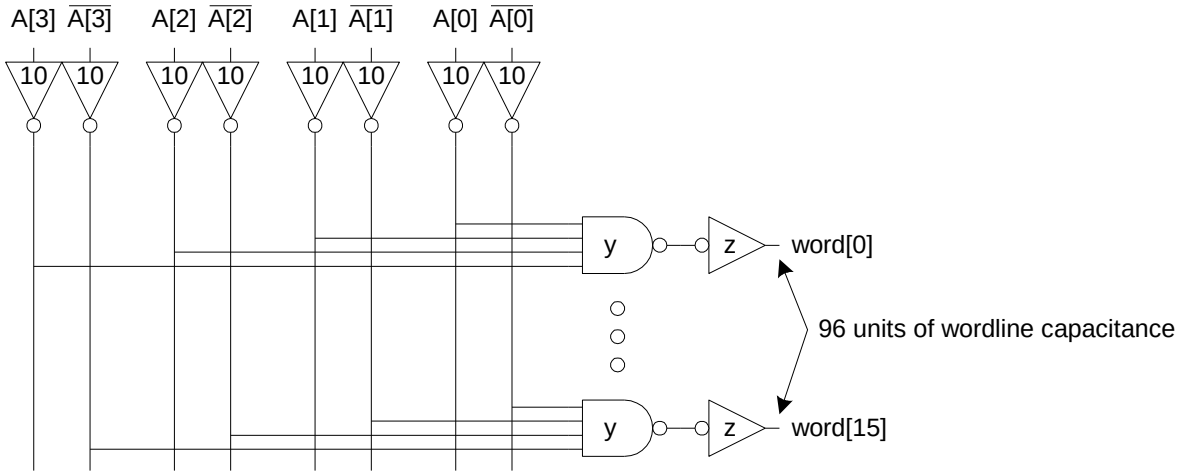
Logical Effort: G =

Path Effort: F =

Stage Effort: $\hat{f} =$

Path Delay: $D =$

Gate sizes: z = y =



Comparison

- 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	$h = \frac{C_{out}}{C_{in}}$	$H = \frac{C_{out\text{-}path}}{C_{in\text{-}path}}$
branching effort	$b = \frac{C_{on\text{-}path} + C_{off\text{-}path}}{C_{on\text{-}path}}$	$B = \prod b_i$
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 $F = GBH$
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 $C_{in_i} = \frac{g_i C_{out_i}}{\hat{f}}$

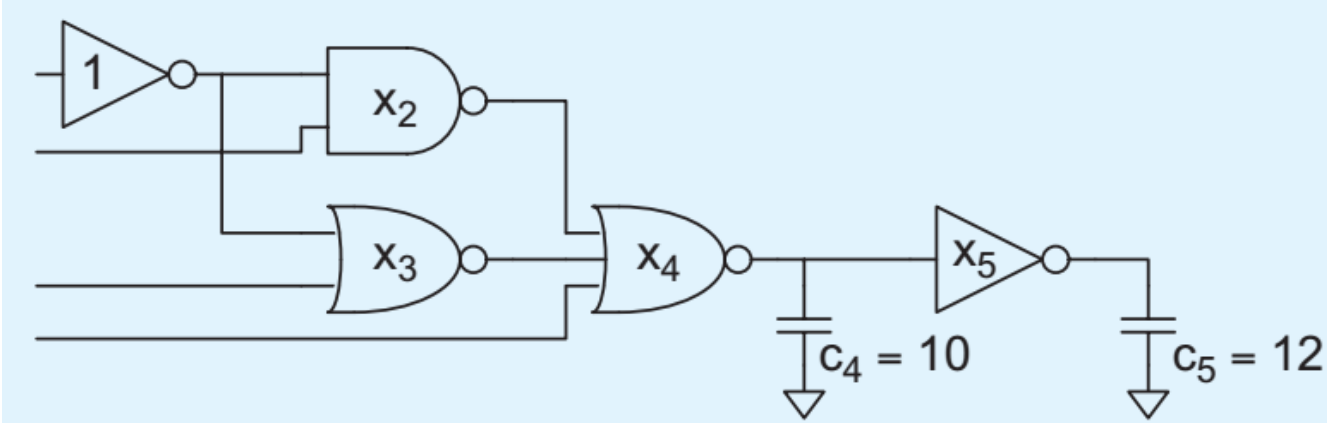
Limits of Logical Effort

- 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

Iterative Solutions for Sizing

- Not all logical effort problems can be solved using the systematic steps.
- Example:

For the following circuit, write an expression for the arrival time of the output as a function of the gate drives. Determine the sizes to achieve minimum delay.



Solution

- The input capacitance of a gate with logical effort g and drive x is $C_{in} = gx$

$$
\begin{aligned}
d_1 &= 1 + \tfrac{4}{3}x_2 + \tfrac{5}{3}x_3 \\
d_2 &= 2 + \tfrac{7}{3}\frac{x_4}{x_2} \\
d_3 &= 2 + \tfrac{7}{3}\frac{x_4}{x_3} \\
d_4 &= 3 + \frac{x_5}{x_4} + \frac{10}{x_4} \\
d_5 &= 1 + \frac{12}{x_5}
\end{aligned}
$$

- The arrival time can be expressed as:

$$a_1 = d_1$$
$$a_2 = a_1 + d_2$$
$$a_3 = a_1 + d_3$$
$$a_4 = \max\{a_2, a_3\} + d_4$$
$$a_5 = a_4 + d_5 = d_1 + \max\{d_2, d_3\} + d_4 + d_5$$

- Use an automatic solver to choose the drives to minimize the latest arrival time

Stage (i)	x_i	f_i	c_{in}	d_i	a_i
1: INV	1	4.85	1	5.85	5.85
2: NAND2	1.62	4.85	2.16	6.85	12.70
3: NOR2	1.62	4.85	2.70	6.85	12.70
4: NOR3	3.37	4.85	7.86	7.85	20.55
5: INV	6.35	1.89	6.35	2.89	23.44

Example Observations

- In paths that branch, each fork should contribute equal delay.
- If one fork were faster than the other, it could be downsized to reduce the capacitance it presents to the stage before the branch.
- The stage efforts, *f*, are equal for each gate in paths with no fixed capacitive loads, but may change after a load.
- To minimize delay, upsize gates on nodes with large fixed capacitances to reduce the effort borne by the gate, while only slightly increasing the effort borne by the predecessor.

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