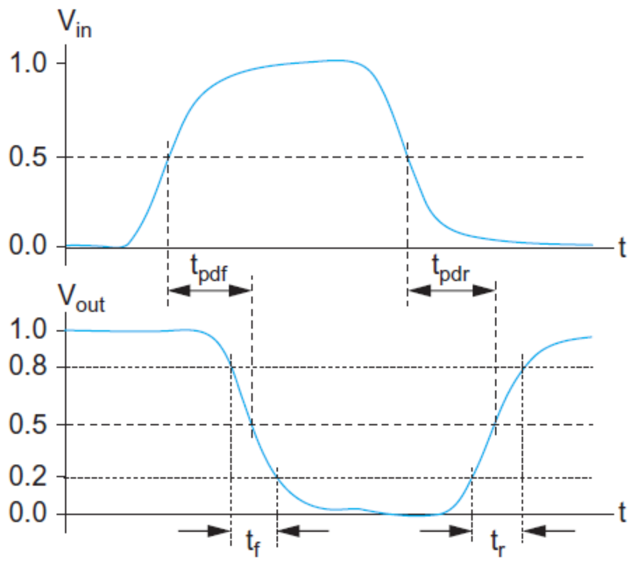
DELAY

Delay

- The two most common metrics for a good chip are speed and power.
- Delay and power are influenced as much by the wires as by the transistors.

Definitions

- *Propagation delay time*, *tpd* = maximum time from the input crossing 50% to the output crossing 50%
- *Contamination delay time*, *tcd* = minimum time from the input crossing 50% to the output crossing 50%
- *Rise time*, *tr* = time for a waveform to rise from 20% to 80% of its steady-state value
- *Fall time*, *tf* = time for a waveform to fall from 80% to 20% of its steady-state value
- *Edge rate*, *trf* = (*tr* + *tf*)/2



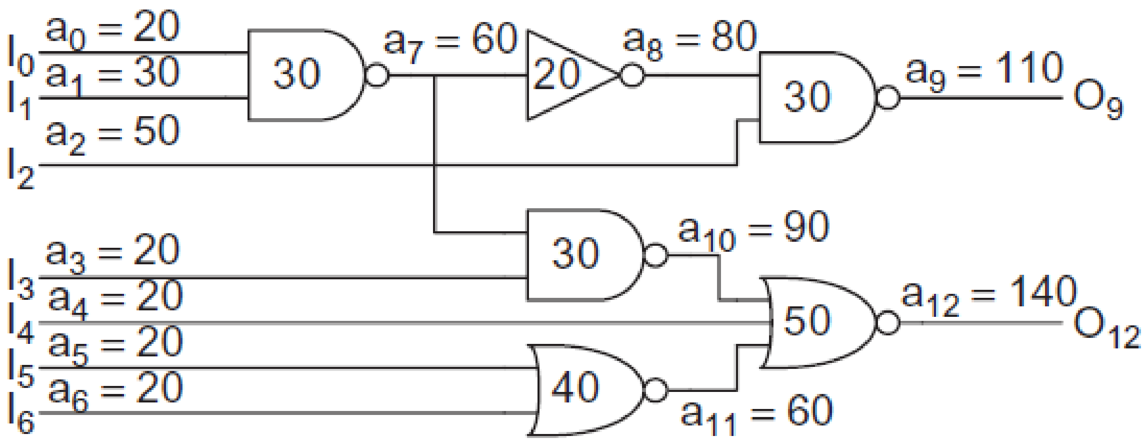
Arrival Time

- A *timing analyzer* computes the arrival times, i.e., the latest time at which each node in a block of logic will switch.
- The nodes are classified as inputs, outputs, and internal nodes.
- The user must specify the *arrival time* of inputs and the time data is required at the outputs.
- The arrival time a_i at internal node *i* depends on the propagation delay of the gate driving *i* and the arrival times of the inputs to the gate:

$$a_i = \max_{j \in fanin(i)} \left\{ a_j \right\} + t_{pd_i}$$

Slack

- The timing analyzer computes the arrival times at each node and checks that the outputs arrive by their required time.
- The *slack* is the difference between the required and arrival times.
- *Positive slack* means that the circuit meets timing.
- *Negative slack* means that the circuit is not fast enough.



Timing Optimization

- The critical paths can be affected at four main levels:
- The architectural/micro-architectural level
- The logic level
- The circuit level
- The layout level

Architectural Optimization

- This requires a broad knowledge of both the algorithms that implement the function and the technology being targeted,
- Such as how many gate delays fit in a clock cycle,
- How quickly addition occurs,
- how fast memories are accessed, and
- How long signals take to propagate along a wire.
- Trade-offs at the micro-architectural level include the number of pipeline stages, the number of execution units (parallelism), and the size of memories.

Logical Level Optimization

- Trade-offs include types of functional blocks (e.g., ripple carry vs. lookahead adders)
- The number of stages of gates in the clock cycle, and
- The fan-in and fan-out of the gates.
- The transformation from function to gates and registers can be done by experience, by experimentation, or, most often, by logic synthesis.
- No amount of skillful logic design can overcome a poor microarchitecture.

Circuit Level

- Once the logic has been selected, the delay can be tuned at the circuit level by choosing transistor sizes or using other styles of CMOS logic.
- Finally, delay is dependent on the layout.
- The floorplan (either manually or automatically generated) is of great importance because it determines the wire lengths that can dominate delay.
- Good cell layouts can also reduce parasitic capacitance.

Timing Constraint

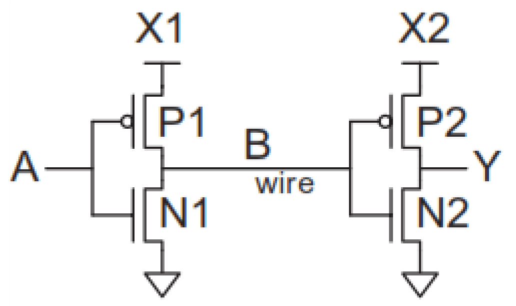
- Many RTL designers never venture below the microarchitectural level.
- A common design practice is to write RTL code, synthesize it (allowing the synthesizer to do the timing optimizations at the logic, circuit, and placement levels) and check if the results are fast enough.
- If they are not, the designer recodes the RTL with more parallelism or pipelining, or changes the algorithm and repeats until the timing constraints are satisfied.
- Timing analyzers are used to check *timing closure, i.e., whether the circuit meets all of the timing constraints.*
- Without an understanding of the lower levels of abstraction where the synthesizer is working, a designer may have a difficult time achieving timing closure on a challenging system.

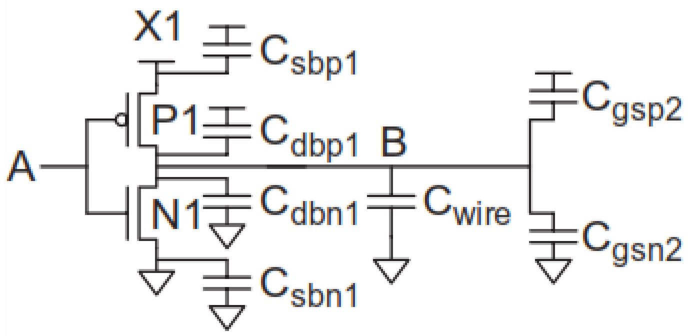
Transient Response

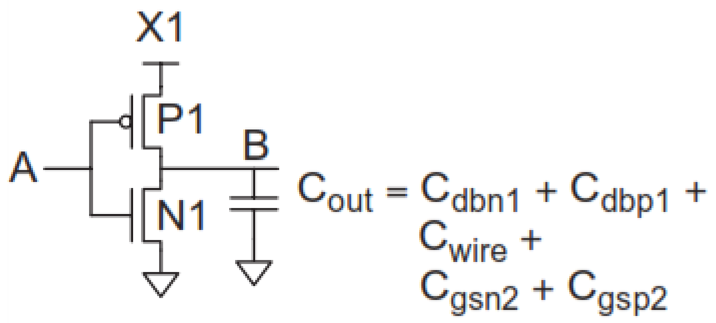
- The most fundamental way to compute delay is to develop a physical model of the circuit of interest
- Write a differential equation describing the output voltage as a function of input voltage and time, and solve the equation.
- The solution of the differential equation is called the *transient response, and the delay is the time when the output reaches* $V_{DD}/2$.
- The differential equation is based on charging or discharging of the capacitances in the circuit.
- The circuit takes time to switch because the capacitance cannot change its voltage instantaneously.
- If capacitance C charged with a current I.

$$I = C\frac{dV}{dt}$$

Capacitance for Inverter Delay Calculation



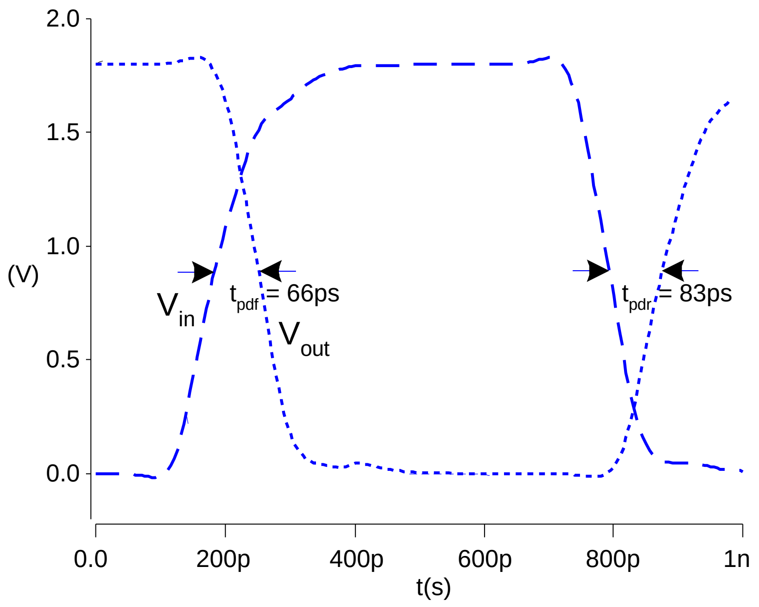



X1
P1
B
A
N1
$C_{out} = C_{dbn1} + C_{dbp1} + C_{wire} + C_{gsn2} + C_{gsp2}$

- An inverter *X1 driving another inverter X2 at the end of a* wire.
- Suppose a voltage step from 0 to V_{DD} is applied to node *A and we wish to* compute the propagation delay, t_{pdf}, through *X1, i.e., the delay from the input* step until node *B crosses* $V_{DD}/2$.

Simulated Inverter Delay

- Solving differential equations by hand is too hard
- SPICE simulator solves the equations numerically
  - Uses more accurate I-V models too!
- But simulations take time to write, may hide insight



Delay Estimation

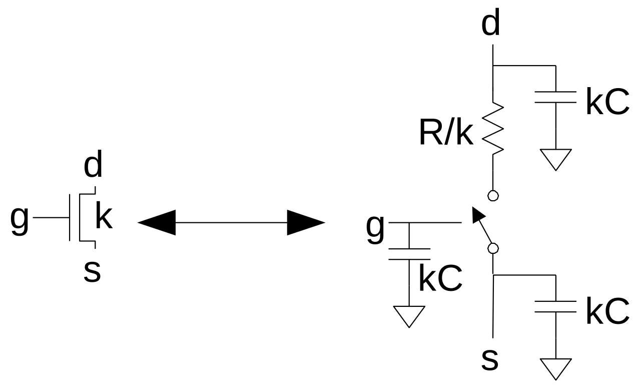
- We would like to be able to easily estimate delay
  - Not as accurate as simulation
- The step response usually looks like a 1st order RC response with a decaying exponential.
- Use RC delay models to estimate delay
  - C = total capacitance on output node
  - Use *effective resistance* R
  - So that t_{pd} = RC
- Characterize transistors by finding their effective R
  - Depends on average current as gate switches

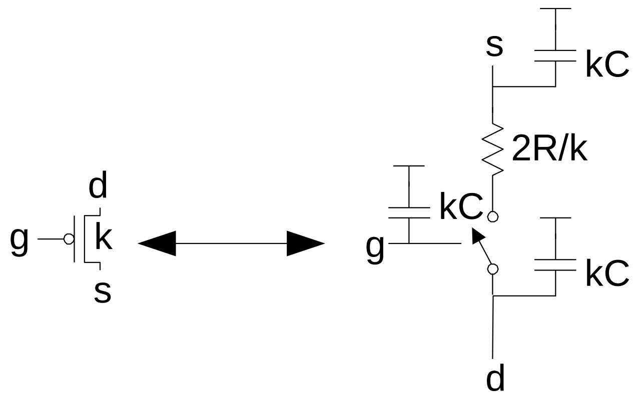
Effective Resistance

- Shockley models have limited value
  - Not accurate enough for modern transistors
  - Too complicated for much hand analysis
- Simplification: treat transistor as resistor
  - Replace $I_{ds}(V_{ds}, V_{gs})$ with effective resistance R
    - $I_{ds} = V_{ds}/R$
  - R averaged across switching of digital gate
- Too inaccurate to predict current at any given time
  - But good enough to predict RC delay

RC Delay Model

- Use equivalent circuits for MOS transistors
  - Ideal switch + capacitance and ON resistance
  - Unit nMOS has resistance R, capacitance C
  - Unit pMOS has resistance 2R, capacitance C
- Capacitance proportional to width(k)
- Resistance inversely proportional to width(k)



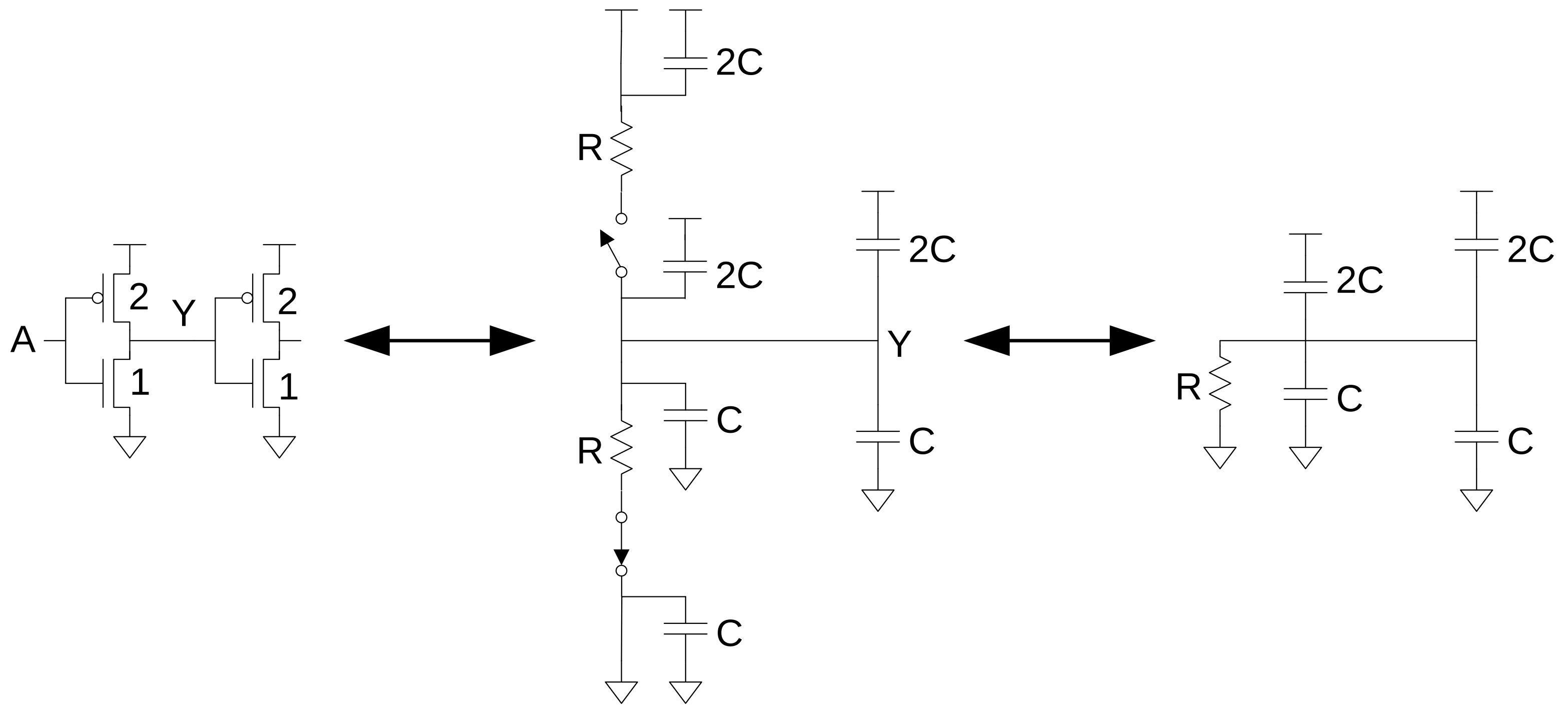


RC Values

- Capacitance
  - $C = C_g = C_s = C_d$ = 2 fF/μm of gate width in 0.6 μm
  - Gradually decline to 1 fF/μm in nanometer techs.
- Resistance
  - R ≈ 6 KΩ*μm in 0.6 μm process
  - Improves with shorter channel lengths
- Unit transistors
  - May refer to minimum contacted device (4/2 λ)
  - Or maybe 1 μm wide device
  - Doesn’t matter as long as you are consistent

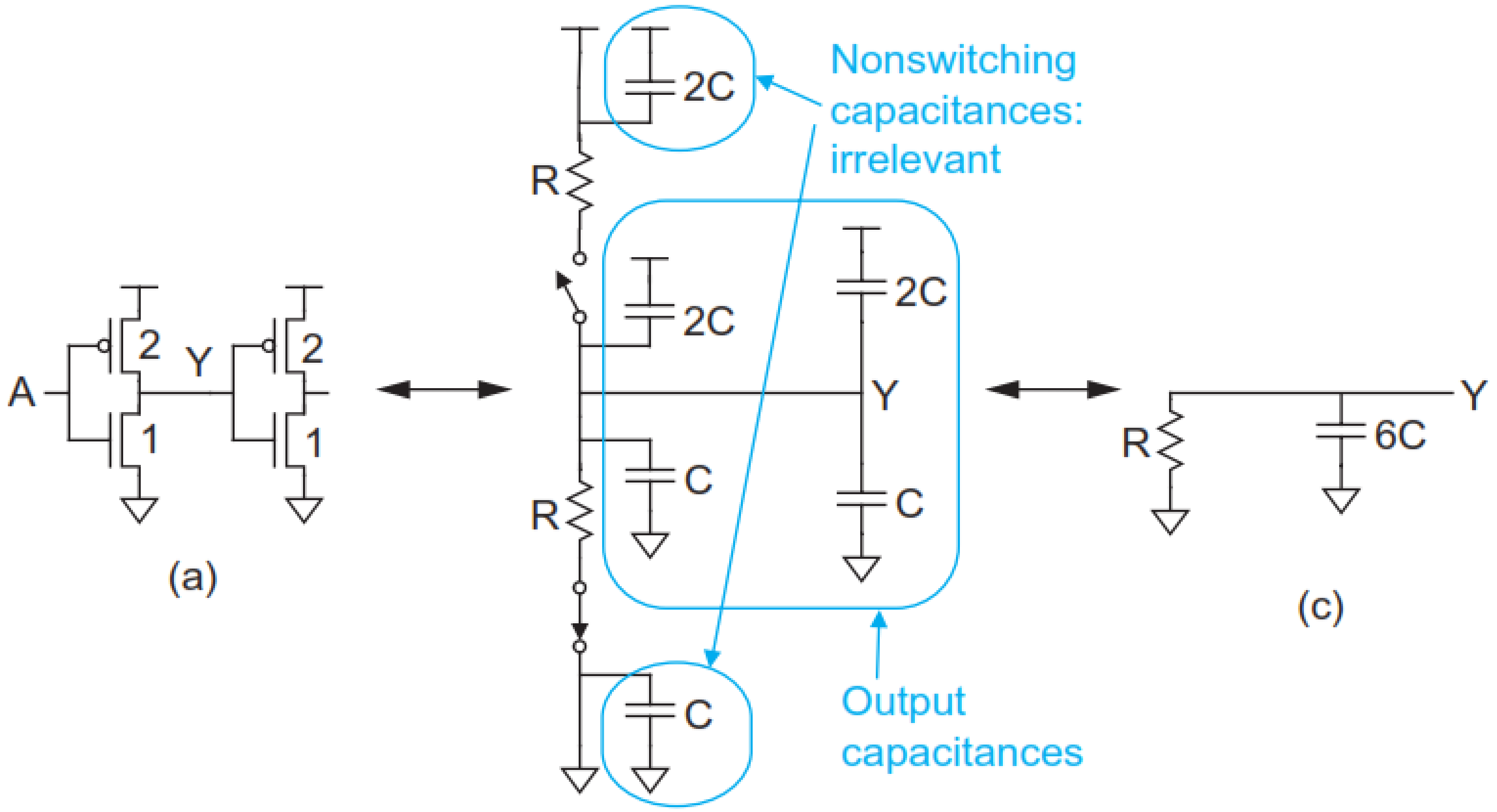
Inverter Delay Estimate

- Estimate the delay of a fanout-of-1 inverter



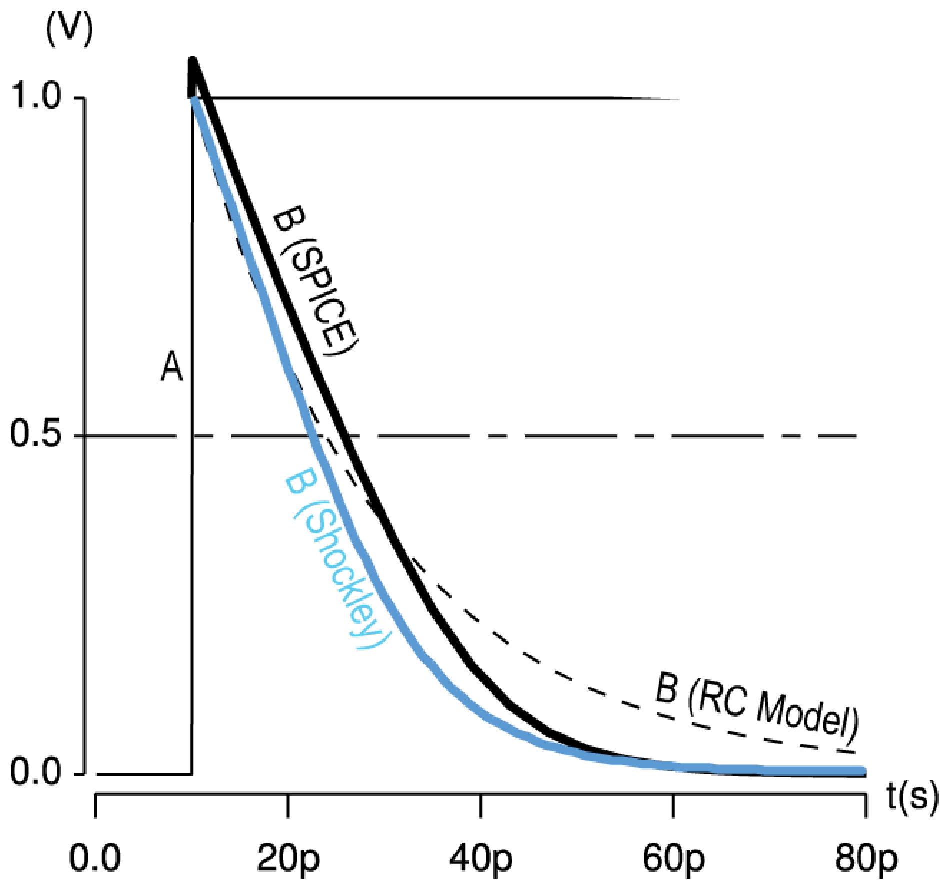
d = 6RC

RC Circuit

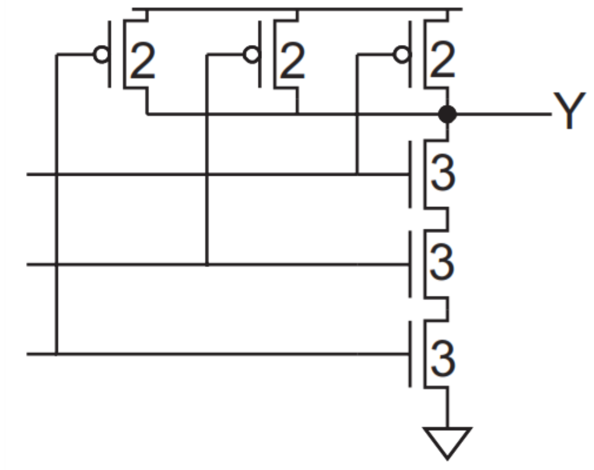


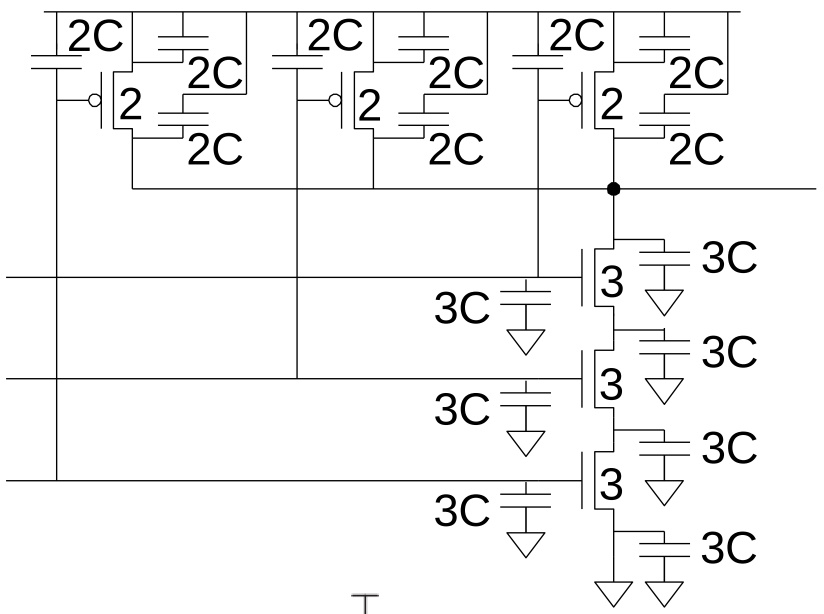
A
2
1
Y
2
1
(a)
R
2C
R
2C
2C
Y
C
C
C
Nonswitching capacitances: irrelevant
Output capacitances
R
6C
Y
(c)

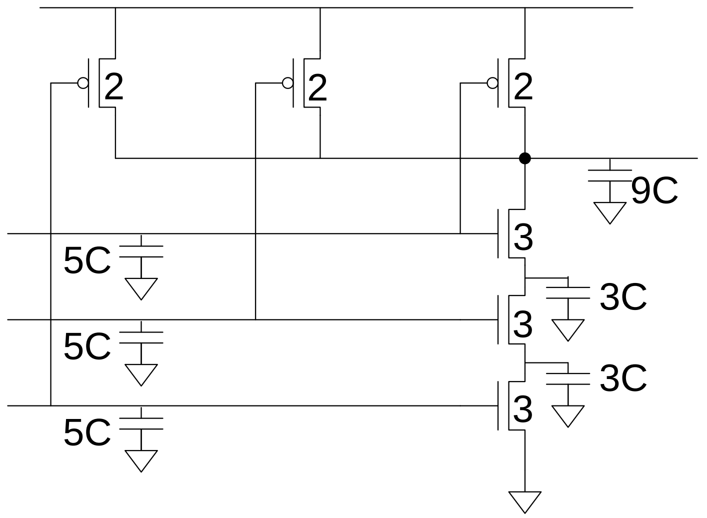
Delay Model Comparison

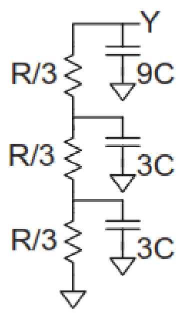


Example: 3-input NAND

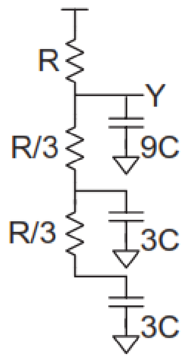








Falling



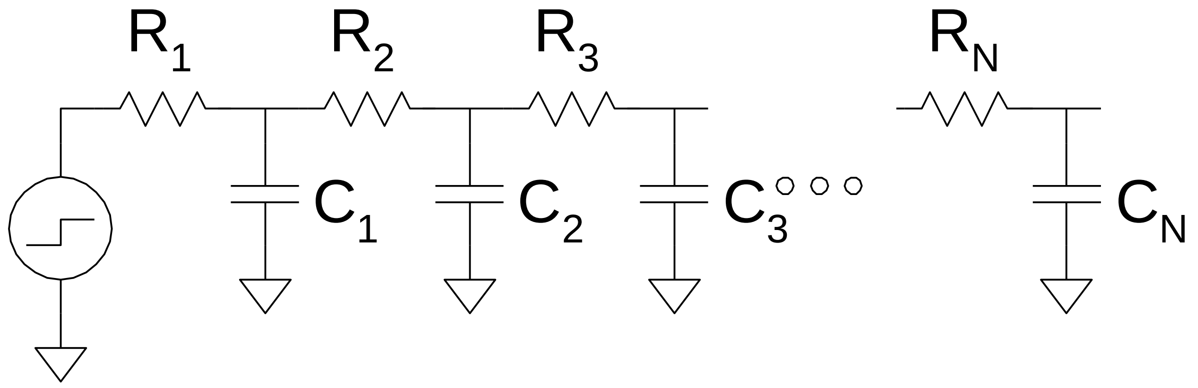
Rising

Elmore Delay

- ON transistors look like resistors
- Pullup or pulldown network modeled as *RC ladder*
- Elmore delay of RC ladder

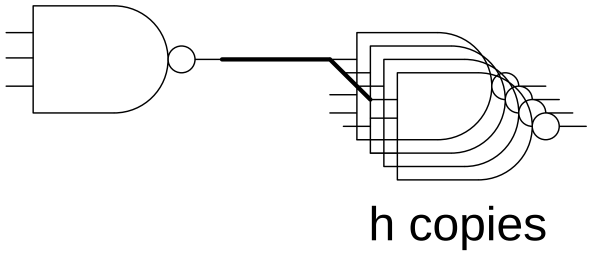
$$t_{pd} \approx \sum_{\text{nodes } i} R_{i-to-source} C_i$$

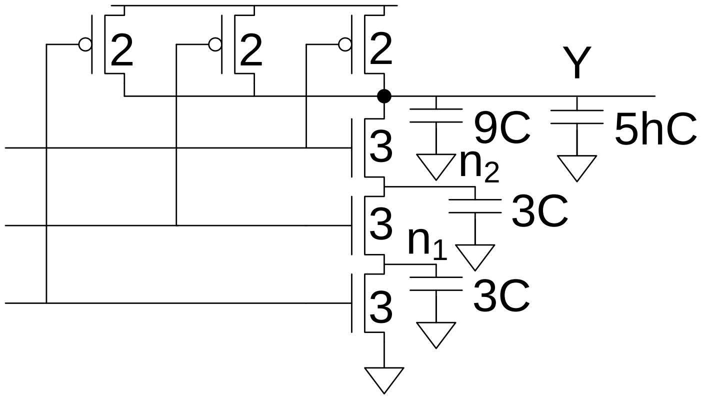
$$= R_1 C_1 + \left(R_1 + R_2\right) C_2 + \ldots + \left(R_1 + R_2 + \ldots + R_N\right) C_N$$

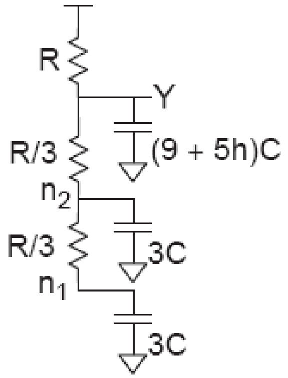


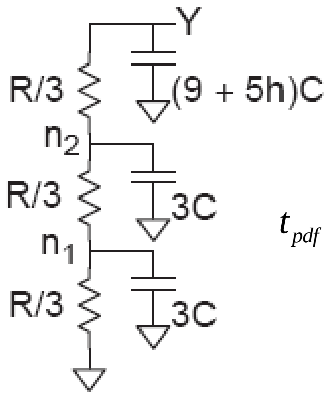
Example: 3-input NAND

- Estimate worst-case rising and falling delay of 3-input NAND driving *h* identical gates.









$$t_{pdf} = (3C)\left(\tfrac{R}{3}\right) + (3C)\left(\tfrac{R}{3} + \tfrac{R}{3}\right) + \left[(9+5h)C\right]\left(\tfrac{R}{3} + \tfrac{R}{3} + \tfrac{R}{3}\right)$$
$$= (11+5h)RC$$

$$t_{pdr} = (9+5h)RC$$

Delay Components

- Delay has two parts
  - *Parasitic delay*
    - 9 or 11 RC
    - Independent of load
  - *Effort delay*
    - 5h RC
    - Proportional to load capacitance

LOGICAL EFFORT

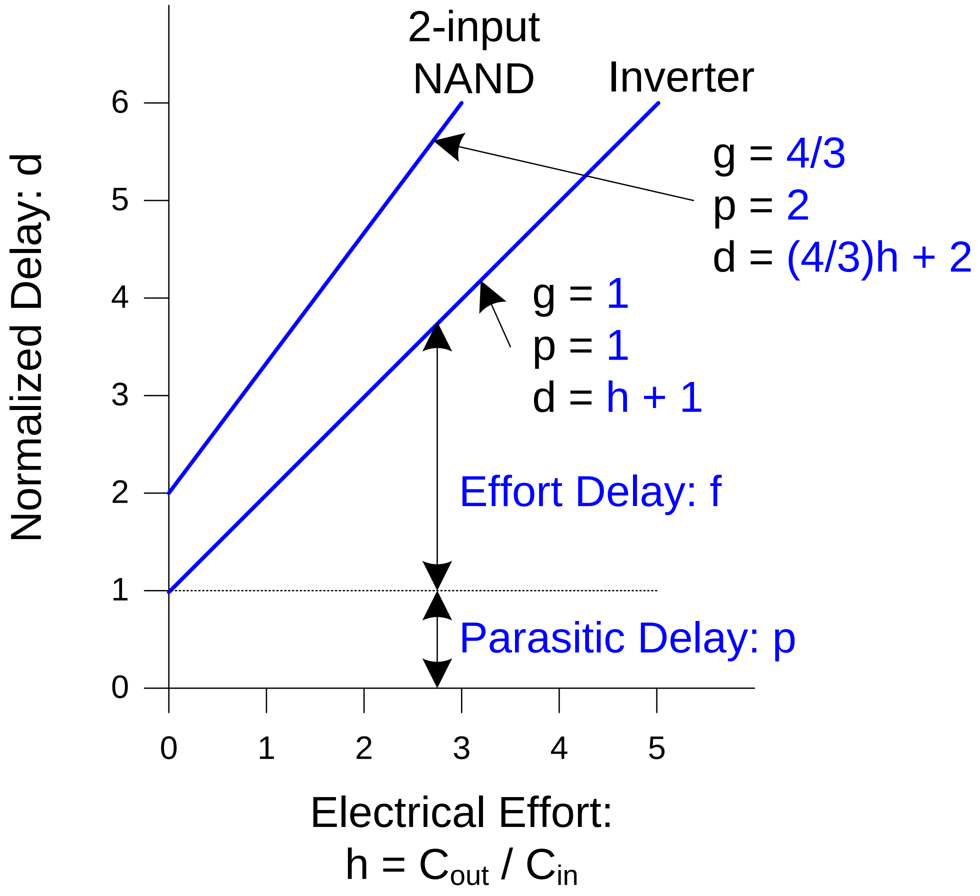
Delay in a Logic Gate

- 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

Delay Plots

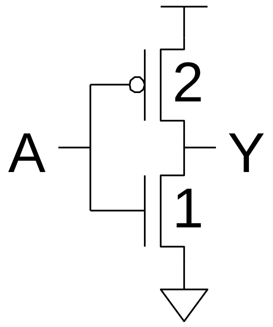
- $d = f + p$

  $= gh + p$

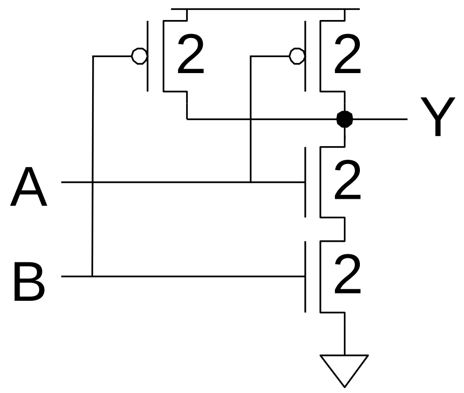


Computing Logical Effort

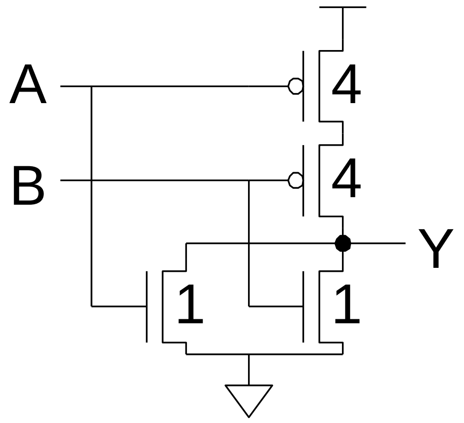
- DEF: *Logical effort is the ratio of the input capacitance of a gate to the input capacitance of an inverter delivering the same output current*.
- Measure from delay vs. fanout plots
- Or estimate by counting transistor widths



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



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



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

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	

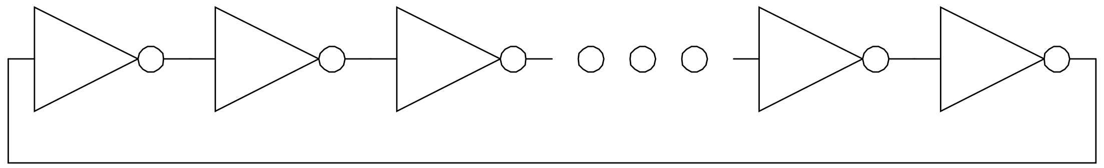
Catalog of Gates

- Parasitic delay of common gates
  - In multiples of p_{inv} ($\approx$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	

Example: Ring Oscillator

- Estimate the frequency of an N-stage ring oscillator

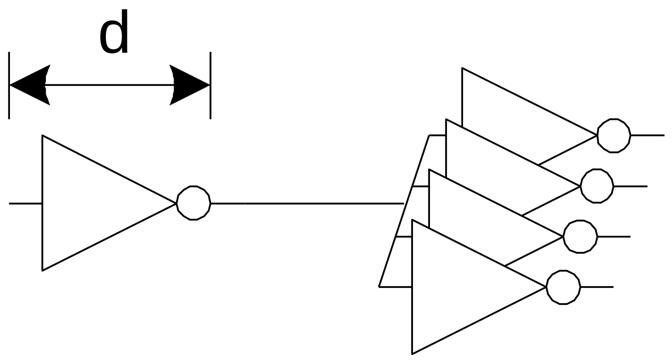


Logical Effort:	$g = 1$
Electrical Effort:	$h = 1$
Parasitic Delay:	$p = 1$
Stage Delay:	$d = 2$
Frequency:	$f_{osc} = 1/(2*N*d) = 1/(4N)$

31 stage ring oscillator in 0.6 μm process has frequency of ~ 200 MHz

Example: FO4 Inverter

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

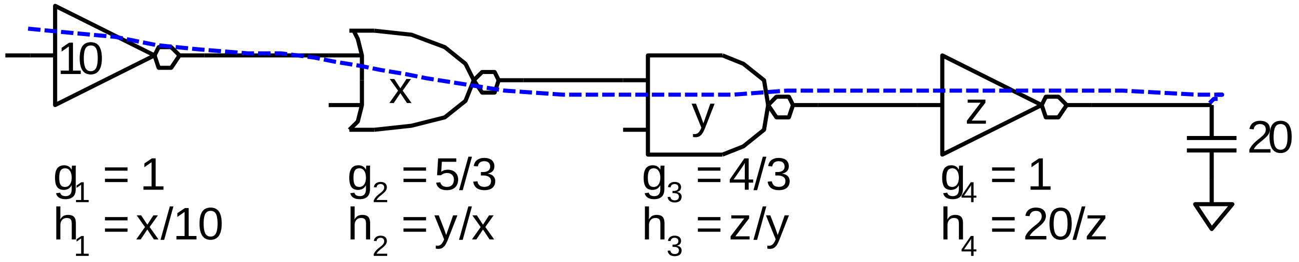


Logical Effort:	$g = 1$
Electrical Effort:	$h = 4$
Parasitic Delay:	$p = 1$
Stage Delay:	$d = 5$

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*

$$F = \prod f_i = \prod g_i h_i$$



$g_1 = 1$
$h_1 = x/10$

$g_2 = 5/3$
$h_2 = y/x$

$g_3 = 4/3$
$h_3 = z/y$

$g_4 = 1$
$h_4 = 20/z$

Multistage Logic Networks

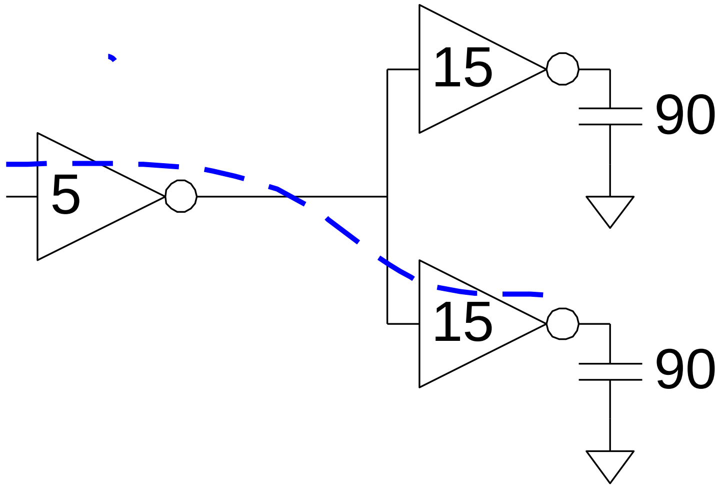
- Logical effort generalizes to multistage networks
- *Path Logical Effort* $G = \prod g_i$
- *Path Electrical Effort* $H = \frac{C_{out-path}}{C_{in-path}}$
- *Path Effort*

$$F = \prod f_i = \prod g_i h_i$$

- Can we write F = GH?

Paths that Branch

G	= 1
H	= 90 / 5 = 18
GH	= 18
h_1	= (15 +15) / 5 = 6
h_2	= 90 / 15 = 6
F	$= g_1 g_2 h_1 h_2$



Branching Effort

- Introduce *branching effort*
  - Accounts for branching between stages in path

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

$$B = \prod b_i$$

- Now we compute the path effort
  - F = GBH

Note:

$$\prod h_i = BH$$

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

Gate Sizes

- How wide should the gates be for least delay?

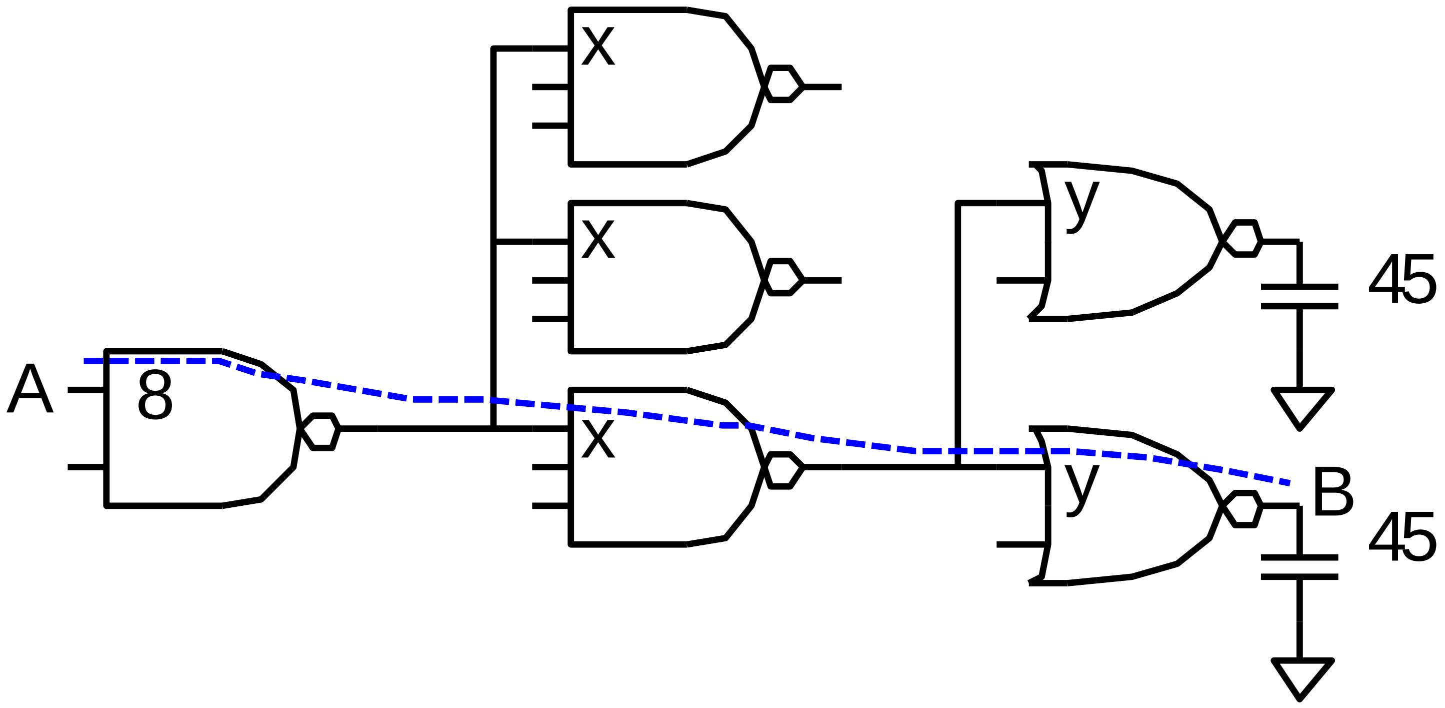
$$\hat{f} = gh = g\frac{C_{out}}{C_{in}}$$

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

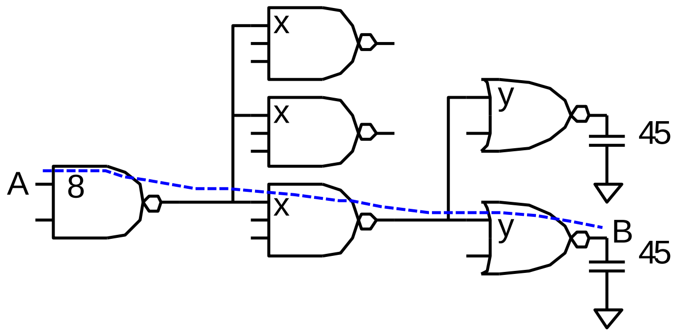
- Working backward, apply capacitance transformation to find input capacitance of each gate given load it drives.
- Check work by verifying input cap spec is met.

Example: 3-stage path

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



Example: 3-stage path



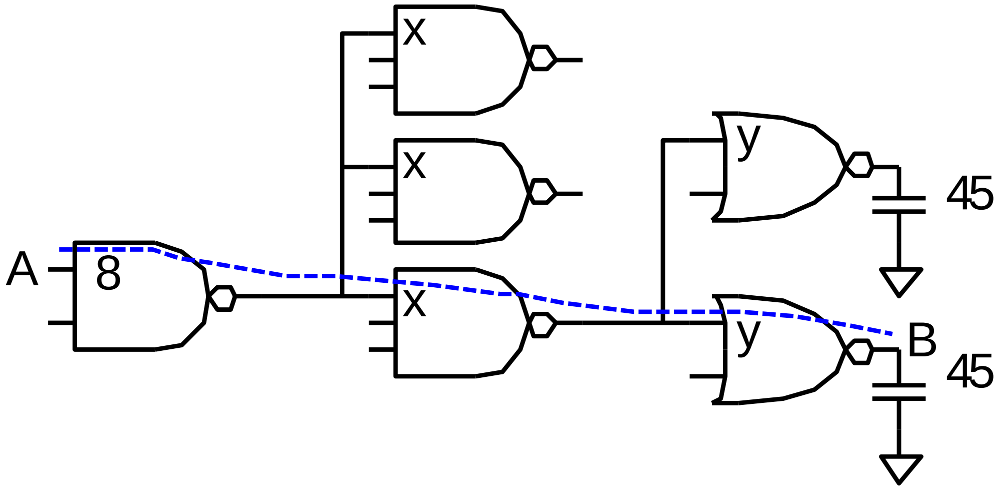
Logical Effort	G = (4/3)*(5/3)*(5/3) = 100/27
Electrical Effort	H = 45/8
Branching Effort	B = 3 * 2 = 6
Path Effort	F = GBH = 125
Best Stage Effort	$\hat{f} = \sqrt[3]{F} = 5$
Parasitic Delay	P = 2 + 3 + 2 = 7
Delay	D = 3*5 + 7 = 22 = 4.4 FO4

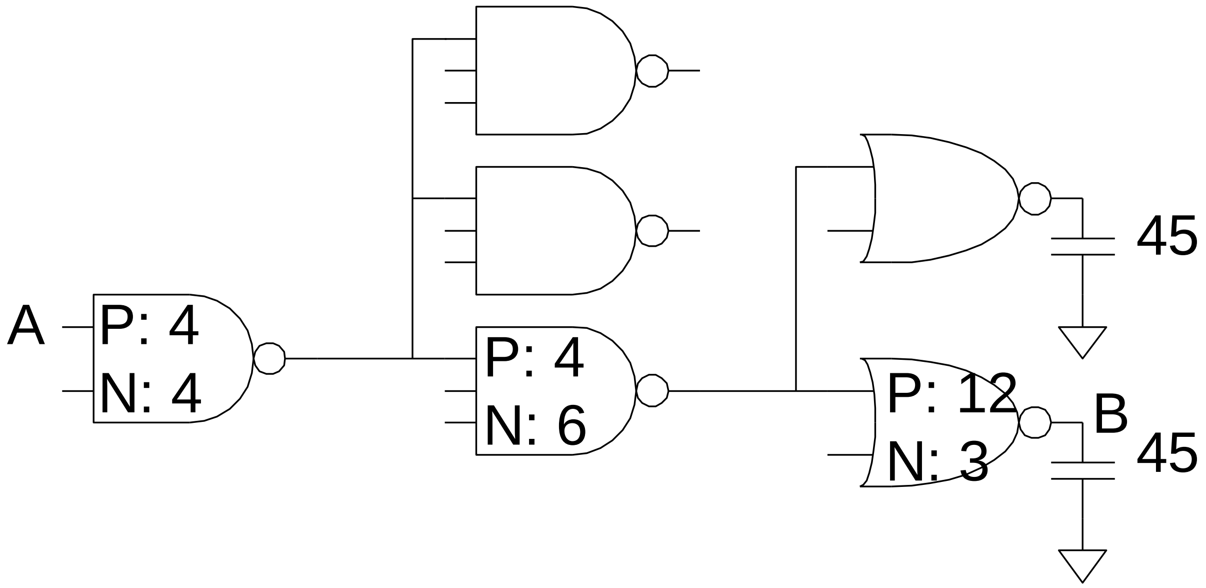
Example: 3-stage path

- Work backward for sizes

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

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





Example: 3-stage path

- We should check that the initial 2-input NAND gate should have a size of
  - (10+10+10)*(4/3)/5 = 8 as desired
  - {(3 fanout)*g for 2input NAND}/F
- The NAND2 gate delay is
  - $d_1 = g_1h_1 + p_1 = (4/3)*(10+10+10)/8 + 2 = 7$
- The NAND3 gate delay is
  - $d_2 = g_2h_2 + p_2 = (5/3)*(15+15)/10 + 3 = 8$
- The NOR3 gate delay is
  - $d_3 = g_3h_3 + p_3 = (5/3) * (45/15) + 2 = 7$
- Hence path delay = $d_1 + d_2 + d_3 = 22$

$p_1 = 2$ because of 2 input NAND

$p_2 = 3$ because of 3 input NAND

Point to Note

- Inexperience designer: wider transistor offer more current and thus try to make circuit faster by using bigger gates
- Increasing the size of any of the gates except the first one only makes the circuit slower
- For example:
  - Increasing the size of the NAND3 make the NAND3 faster but make the NAND2 slower
    - Resulting a net speed loss
  - Increasing the size of the initial NAND2 gate does speed up the circuit under consideration
  - However, it presents a larger load on the path that computes input A, making that path slower
- Hence, it is crucial to have a specification of not only the load the path must drive but also the maximum input capacitance the path may present

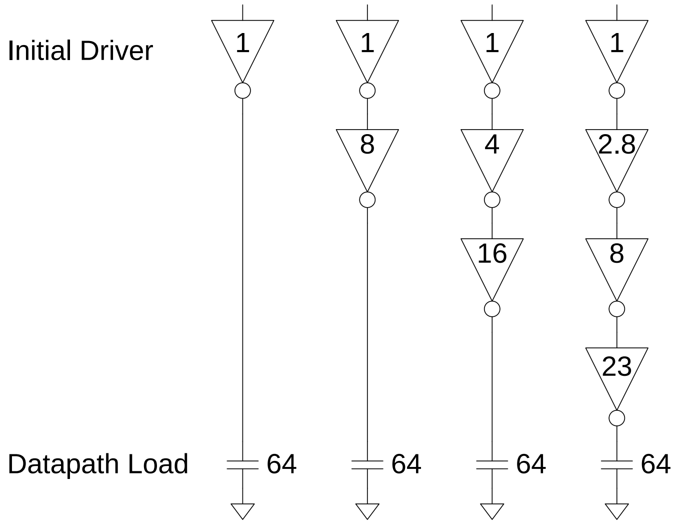
Choosing the Best Number of Stages

- There are many different topologies that implement a particular logic function
- Logical effort tells us that
  - NANDs are better than NORs and
  - that gates with few inputs are better than gates with many
- Logic designer sometimes estimate delay by counting the number of stages of logic, assuming each stage has constant delay
- This is potentially misleading because it implies that the fastest circuits are those that use the fewest stages of logic
- Of course the gate delay actually depends on the electrical effort, so some times using fewer stages results in more delay

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

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

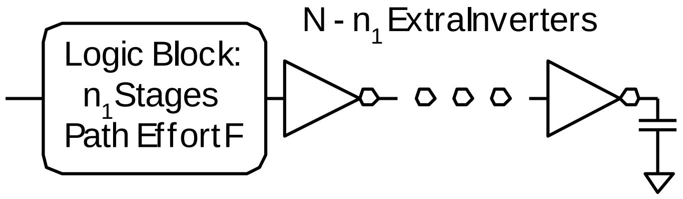


N:	1	2	3	4
f:	64	8	4	2.8
D:	65	18	15	15.3
			Fastest	

Best Number of Stages

- From the above example
  - The 3 stage design is faster and is much superior to a single stage
  - If an even number of inversions were required
    - The two- or four- stage designs are promising
    - The four stage design is slightly faster
    - But two stage design is requires significant area and power

- 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 $\rho = F^{\frac{1}{N}}$

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

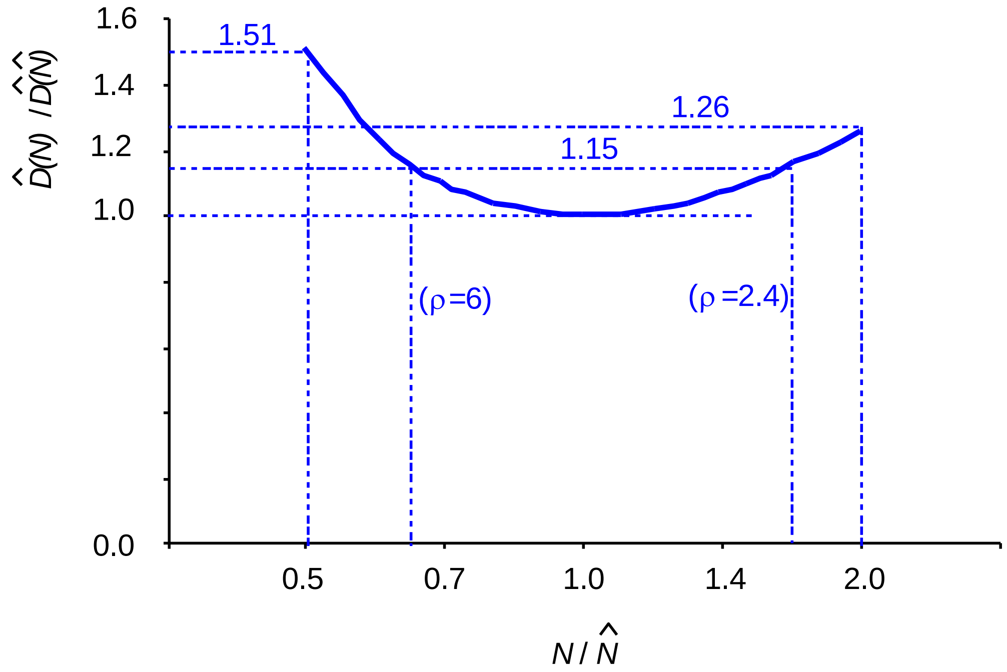
Best Stage Effort

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

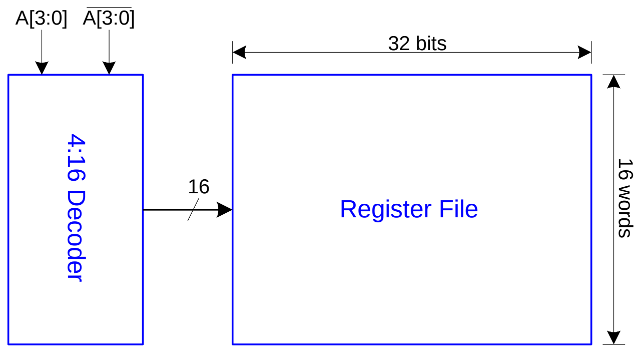
- How sensitive is delay to using exactly the best number of stages?



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

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 address 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?

Decoder Design

- A 2^N word decoder consists of 2^N N-input AND gates
- Then problem is reduced to designing a suitable 4-input ANG gate.
- The output load on a word line is 32 bits with three units of capacitance each, or 96 units.
- Therefore, the path electrical effort is $H = 96/10 = 9.6.$
- *Each address is used to compute* half of the 16 word lines;
- its complement is used for the other half.
- Therefore,
- a B = 8-way branch is required somewhere in the path.

chicken-and egg dilemma

- We need to know the path logical effort to calculate the path effort and best number of stages.
- However, without knowing the best number of stages, we cannot sketch a path and determine the logical effort for that path.
- There are two ways to resolve the dilemma.
- One is to sketch a path with a random number of stages, determine the path logical effort, and then use that to compute the path effort and the actual number of stages.
- The path can be redesigned with this number of stages, refining the path logical effort.
- If the logical effort changes significantly, the process can be repeated.
- Alternatively, we know that the logic of a decoder is rather simple, so we can ignore the logical effort (assume *$G = 1$).*
- *Then we can proceed with our design, remembering that the best number* of stages is likely slightly higher than predicted because we neglected logical effort.

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
- (INV-NAND4-INV).

Gate Sizes & Delay

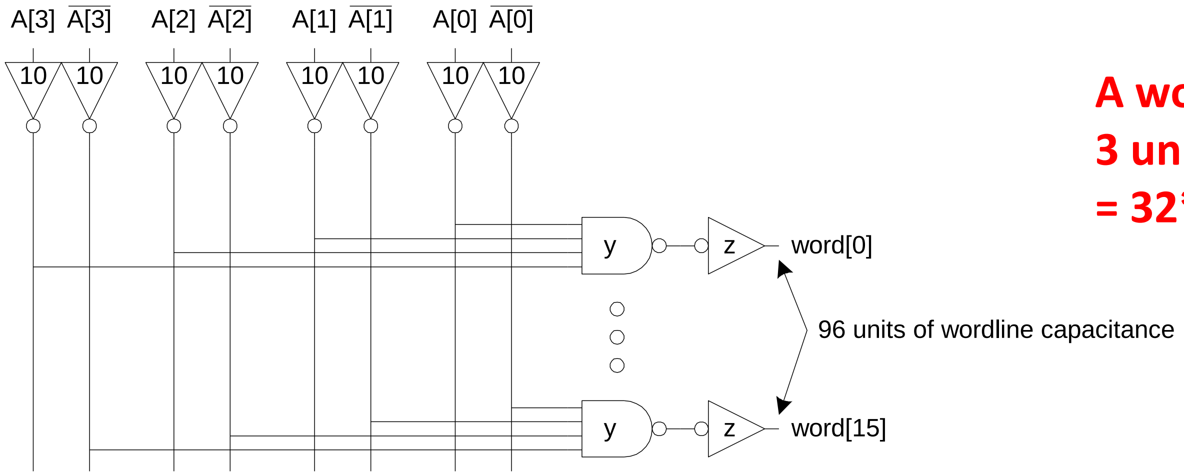
Logical Effort: G = 1 * 6/3 * 1 = 2

Path Effort: F = GBH = 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



A word line of 32-bits with 3 units of capacitance each = 32*3 = 96 units

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	$h = \frac{C_{\text{out}}}{C_{\text{in}}}$	$H = \frac{C_{\text{out-path}}}{C_{\text{in-path}}}$
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 $F = GBH$
2) Estimate best number of stages $N = \log_4 F$
3) Sketch path with N stages $D = NF^{\frac{1}{N}} + P$
4) Estimate least delay $\hat{f} = F^{\frac{1}{N}}$
5) Determine best stage effort
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