CPE/EE 427, CPE 527
VLSI Design I
L11: Logical Effort

Department of Electrical and Computer Engineering
University of Alabama in Huntsville

Aleksandar Milenkovic (www.ece.uah.edu/~milenka)
www.ece.uah.edu/~milenka/cpe527-05F

Course Administration

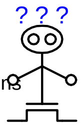
- Instructor: Aleksandar Milenkovic
  milenka@ece.uah.edu
  www.ece.uah.edu/~milenka
  EB 217-L
  Mon. 5:30 PM – 6:30 PM,
  Wen. 12:30 – 13:30 PM
- URL: http://www.ece.uah.edu/~milenka/cpe527-05F
- TA: Joel Wilder
- Labs: Lab#3: due 10/07/05
- Hws: Solutions in secure directory /scr (cpe427fall05, ?)
- Project: Proposals due 10/10/05
- Test I: 10/17/05
- Text: CMOS VLSI Design, 3rd ed., Weste, Harris
- Review: Chapters 1, 2, 3, 4;
- Today: Logical Effort (Chapter 4)

Outline

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

Introduction

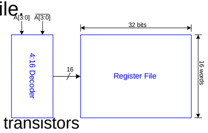
- Chip designers face a bewildering array of choices
  - What is the best circuit topology for a function?
  - How many stages of logic give least delay?
  - How wide should the transistors be?



- 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

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

$\tau =$ 3RC
$\approx$ 12 ps in 180 nm process
40 ps in 0.6 μm process

Delay in a Logic Gate

- Express delays in process-independent unit

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

- Delay has two components

$$d = f + p$$

Delay in a Logic Gate

- Express delays in process-independent unit

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

- Delay has two components

$$d = f + p$$

- *Effort delay* $f = gh$ (a.k.a. *stage effort*)
  - Again has two components

Delay in a Logic Gate

- Express delays in process-independent unit

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

- Delay has two components

$$d = f + p$$

- Effort delay $f = gh$ (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

Delay in a Logic Gate

- Express delays in process-independent unit

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

- Delay has two components

$$d = f + p$$

- Effort delay $f = gh$ (a.k.a. stage effort)
  - Again has two components
- h: *electrical effort* = C_{out} / C_{in}
  - Ratio of output to input capacitance
  - Sometimes called fanout

Delay in a Logic Gate

- Express delays in process-independent unit

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

- Delay has two components

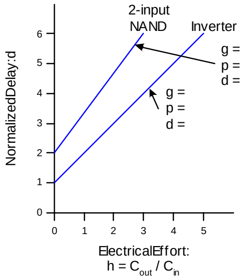
$$d = f + p$$

- Parasitic delay p
  - Represents delay of gate driving no load
  - Set by internal parasitic capacitance

Delay Plots

$d \;= f + p$

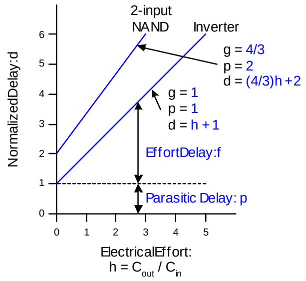
$\;\;\;= gh + p$



Delay Plots

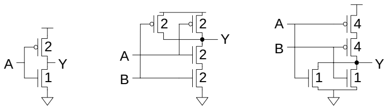
$$d = f + p$$
$$= gh + p$$

- What about NOR2?



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	

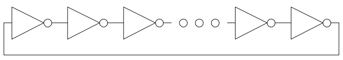
Catalog of Gates

- 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	

Example: Ring Oscillator

- Estimate the frequency of an N-stage ring oscillator



Logical Effort: g =

Electrical Effort: h =

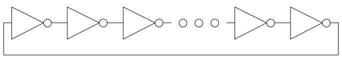
Parasitic Delay: p =

Stage Delay: d =

Frequency: f_{osc} =

Example: Ring Oscillator

- Estimate the frequency of an N-stage ring oscillator



Logical Effort: g = 1

Electrical Effort: h = 1

Parasitic Delay: p = 1

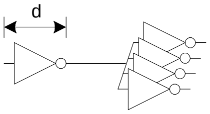
Stage Delay: d = 2

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

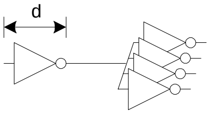
- Estimate the delay of a fanout-of-4 (FO4) inverter



Logical Effort: g =
Electrical Effort: h =
Parasitic Delay: p =
Stage Delay: d =

Example: FO4 Inverter

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

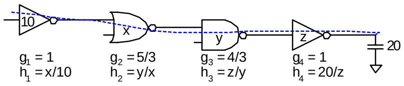


Logical Effort: g = 1
Electrical Effort: h = 4
Parasitic Delay: p = 1
Stage Delay: d = 5

The FO4 delay is about
200 ps in 0.6 μm process
60 ps in a 180 nm process
f/3 ns in an *f* μm process

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

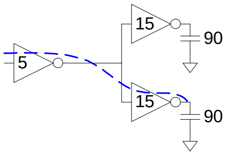


Multistage Logic Networks

- 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

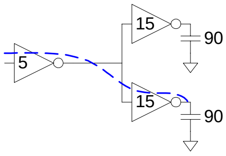
- No! Consider paths that branch:



G =
H =
GH =
h_1 =
h_2 =
F = GH?

Paths that Branch

- No! Consider paths that branch:



G = 1
H = 90 / 5 = 18
GH = 18
h_1 = (15 +15) / 5 = 6
h_2 = 90 / 15 = 6
F = $g_1 g_2 h_1 h_2$ = 36 = 2GH

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

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

- 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?

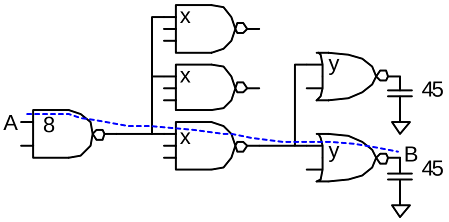
$$\hat{f} = gh = g\frac{C_{out}}{C_{in}}$$

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

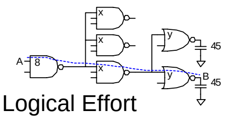
- Working backward, apply capacitance transformation to find input capacitance of each gate given load it drives.
- Check work by verifying input cap spec is met.

Example: 3-stage path

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

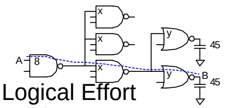


Example: 3-stage path



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

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

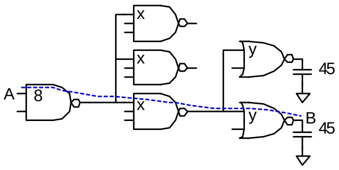
Delay D = 3*5 + 7 = 22 = 4.4 FO4

Example: 3-stage path

- Work backward for sizes

  y =

  x =

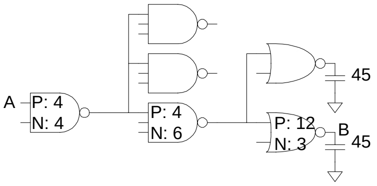


Example: 3-stage path

- Work backward for sizes

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

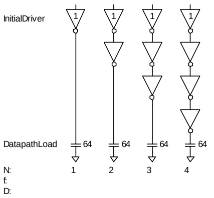
  x = (15*2) * (5/3) / 5 = 10



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 =

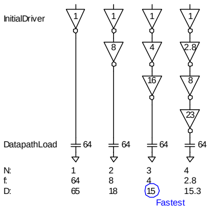


Best Number of Stages

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

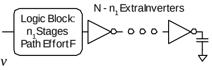
D = NF$^{1/N}$ + P

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



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

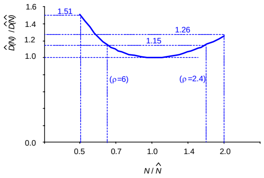
$$p_{inv} + \rho\left(1 - \ln \rho\right) = 0$$

Best Stage Effort

- $p_{inv} + \rho(1 - \ln\rho) = 0$ has no closed-form solution
- Neglecting parasitics (p_{inv} = 0), we find ρ = 2.718 (e)
- For p_{inv} = 1, solve numerically for ρ = 3.59

Sensitivity Analysis

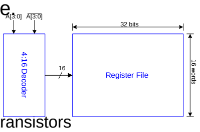
- How sensitive is delay to using exactly the best number of stages?



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

Example, 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 =

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

Gate Sizes & Delay

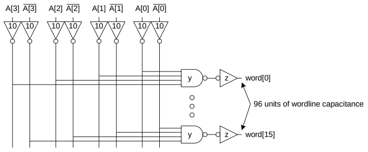
Logical Effort: G =

Path Effort: F =

Stage Effort: $\hat{f} =$

Path Delay: $D =$

Gate sizes: z = y =



Gate Sizes & Delay

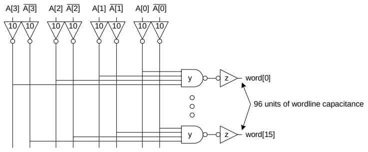
Logical Effort: G = 1 * 6/3 * 1 = 2

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



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

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
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 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

Wires

Outline

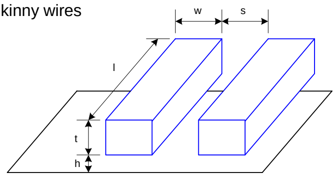
- Introduction
- Wire Resistance
- Wire Capacitance
- Wire RC Delay
- Crosstalk
- Wire Engineering
- Repeaters

Introduction

- Chips are mostly made of wires called *interconnect*
  - In stick diagram, wires set size
  - Transistors are little things under the wires
  - Many layers of wires
- Wires are as important as transistors
  - Speed
  - Power
  - Noise
- Alternating layers run orthogonally

Wire Geometry

- Pitch = w + s
- Aspect ratio: AR = t/w
  - Old processes had AR << 1
  - Modern processes have AR ≈ 2
    - Pack in many skinny wires

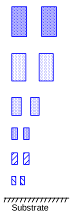


Layer Stack

- AMI 0.6 μm process has 3 metal layers
- Modern processes use 6-10+ metal layers
- Example:

  Intel 180 nm process
- M1: thin, narrow (< 3λ)
  - High density cells
- M2-M4: thicker
  - For longer wires
- M5-M6: thickest
  - For V_{DD}, GND, clk

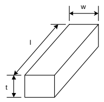
Layer	**T** (nm)	**W** (nm)	**S** (nm)	**AR**
6	1720	860	860	2.0
	1000			
5	1600	800	800	2.0
	1000			
4	1080	540	540	2.0
	700			
3	700	320	320	2.2
	700			
2	700	320	320	2.2
	700			
1	480	250	250	1.9
	800			



Wire Resistance

☐ ρ = *resistivity* (Ω*m)

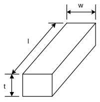
$R =$



Wire Resistance

- ρ = *resistivity* (Ω*m)

$$R = \frac{\rho}{t}\frac{l}{w}$$

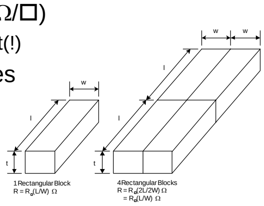


Wire Resistance

- ρ = *resistivity* (Ω*m)

$$R = \frac{\rho}{t}\frac{l}{w} = R_{\square}\frac{l}{w}$$

- $R_{\square}$ = *sheet resistance* (Ω/☐)
  - ☐ is a dimensionless unit(!)
- Count number of squares
  - R = $R_{\square}$ * (# of squares)



Choice of Metals

- Until 180 nm generation, most wires were aluminum
- Modern processes often use copper
  - Cu atoms diffuse into silicon and damage FETs
  - Must be surrounded by a diffusion barrier

Metal	**Bulk resistivity (μΩ*cm)**
Silver (Ag)	1.6
Copper (Cu)	1.7
Gold (Au)	2.2
Aluminum (Al)	2.8
Tungsten (W)	5.3
Molybdenum (Mo)	5.3

Sheet Resistance

- Typical sheet resistances in 180 nm process

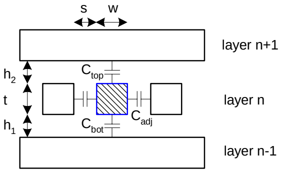
Layer	**Sheet Resistance (Ω/☐)**
Diffusion (silicided)	3-10
Diffusion (no silicide)	50-200
Polysilicon (silicided)	3-10
Polysilicon (no silicide)	50-400
Metal1	0.08
Metal2	0.05
Metal3	0.05
Metal4	0.03
Metal5	0.02
Metal6	0.02

Contacts Resistance

- Contacts and vias also have 2-20 Ω
- Use many contacts for lower R
  - Many small contacts for current crowding around periphery

Wire Capacitance

- Wire has capacitance per unit length
  - To neighbors
  - To layers above and below
- $C_{total} = C_{top} + C_{bot} + 2C_{adj}$

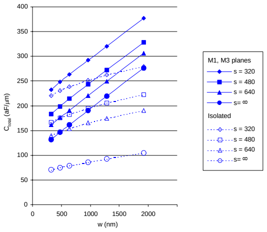


Capacitance Trends

- Parallel plate equation: C = εA/d
  - Wires are not parallel plates, but obey trends
  - Increasing area (W, t) increases capacitance
  - Increasing distance (s, h) decreases capacitance
- Dielectric constant
  - $\varepsilon = k\varepsilon_0$
- $\varepsilon_0 = 8.85 \times 10^{-14}$ F/cm
- k = 3.9 for SiO_2
- Processes are starting to use low-k dielectrics
  - k ≈ 3 (or less) as dielectrics use air pockets

M2 Capacitance Data

- Typical wires have ~ 0.2 fF/µm
  - Compare to 2 fF/µm for gate capacitance

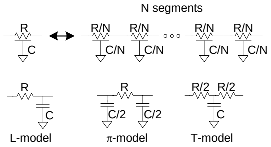


Diffusion & Polysilicon

- Diffusion capacitance is very high (about 2 fF/µm)
  - Comparable to gate capacitance
  - Diffusion also has high resistance
  - Avoid using diffusion *runners* for wires!
- Polysilicon has lower C but high R
  - Use for transistor gates
  - Occasionally for very short wires between gates

Lumped Element Models

- Wires are a distributed system
  - Approximate with lumped element models



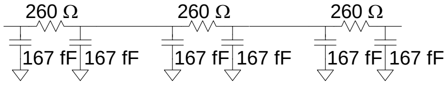
- 3-segment π-model is accurate to 3% in simulation
- L-model needs 100 segments for same accuracy!
- Use single segment π-model for Elmore delay

Example

- Metal2 wire in 180 nm process
  - 5 mm long
  - 0.32 µm wide
- Construct a 3-segment π-model
  - $R_\square$ =
  - $C_{permicron}$ =

Example

- Metal2 wire in 180 nm process
  - 5 mm long
  - 0.32 µm wide
- Construct a 3-segment π-model
  - $R_\square$ = 0.05 Ω/□ => R = 781 Ω
  - $C_{permicron}$ = 0.2 fF/µm => C = 1 pF

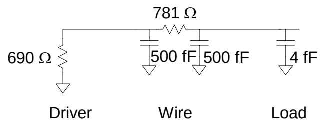


Wire RC Delay

- Estimate the delay of a 10x inverter driving a 2x inverter at the end of the 5mm wire from the previous example.
  - R = 2.5 kΩ*μm for gates
  - Unit inverter: 0.36 μm nMOS, 0.72 μm pMOS
  - t_{pd} =

Wire RC Delay

- Estimate the delay of a 10x inverter driving a 2x inverter at the end of the 5mm wire from the previous example.
  - R = 2.5 kΩ*μm for gates
  - Unit inverter: 0.36 μm nMOS, 0.72 μm pMOS
  - t_{pd} = 1.1 ns

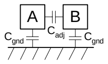


Crosstalk

- A capacitor does not like to change its voltage instantaneously.
- A wire has high capacitance to its neighbor.
  - When the neighbor switches from 1-> 0 or 0->1, the wire tends to switch too.
  - Called capacitive *coupling* or *crosstalk*.
- Crosstalk effects
  - Noise on nonswitching wires
  - Increased delay on switching wires

Crosstalk Delay

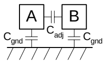
- Assume layers above and below on average are quiet
  - Second terminal of capacitor can be ignored
  - Model as Cgnd = Ctop + Cbot
- Effective Cadj depends on behavior of neighbors
  - Miller effect



B	ΔV	$C_{eff(A)}$	MCF
Constant			
Switching with A			
Switching opposite A			

Crosstalk Delay

- Assume layers above and below on average are quiet
  - Second terminal of capacitor can be ignored
  - Model as Cgnd = Ctop + Cbot
- Effective Cadj depends on behavior of neighbors
  - Miller effect

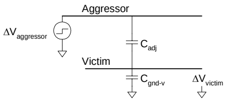


B	ΔV	$C_{eff(A)}$	MCF
Constant	V_{DD}	$C_{gnd} + C_{adj}$	1
Switching with A	0	C_{gnd}	0
Switching opposite A	$2V_{DD}$	$C_{gnd} + 2\ C_{adj}$	2

Crosstalk Noise

- Crosstalk causes noise on nonswitching wires
- If victim is floating:
  - model as capacitive voltage divider

$$\Delta V_{victim} = \frac{C_{adj}}{C_{gnd-v} + C_{adj}} \Delta V_{aggressor}$$

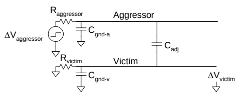


Driven Victims

- Usually victim is driven by a gate that fights noise
  - Noise depends on relative resistances
  - Victim driver is in linear region, agg. in saturation
  - If sizes are same, $R_{aggressor}$ = 2-4 x R_{victim}

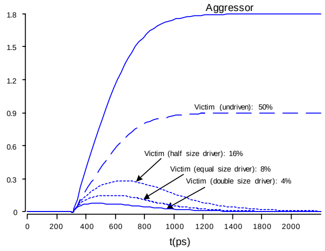
$$\Delta V_{victim} = \frac{C_{adj}}{C_{gnd-v} + C_{adj}} \frac{1}{1+k} \Delta V_{aggressor}$$

$$k = \frac{\tau_{aggressor}}{\tau_{victim}} = \frac{R_{aggressor}\left(C_{gnd-a} + C_{adj}\right)}{R_{victim}\left(C_{gnd-v} + C_{adj}\right)}$$



Coupling Waveforms

- Simulated coupling for $C_{adj} = C_{victim}$



Noise Implications

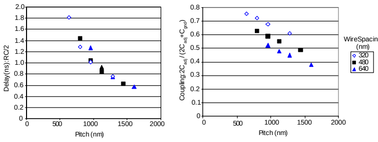
- *So what* if we have noise?
- If the noise is less than the noise margin, nothing happens
- Static CMOS logic will eventually settle to correct output even if disturbed by large noise spikes
  - But glitches cause extra delay
  - Also cause extra power from false transitions
- Dynamic logic never recovers from glitches
- Memories and other sensitive circuits also can produce the wrong answer

Wire Engineering

- Goal: achieve delay, area, power goals with acceptable noise
- Degrees of freedom:

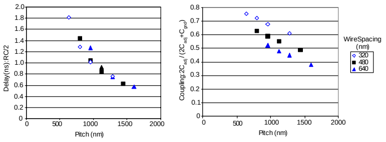
Wire Engineering

- Goal: achieve delay, area, power goals with acceptable noise
- Degrees of freedom:
  - Width
  - Spacing



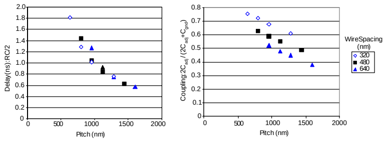
Wire Engineering

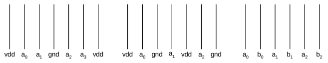
- Goal: achieve delay, area, power goals with acceptable noise
- Degrees of freedom:
  - Width
  - Spacing
  - Layer



Wire Engineering

- Goal: achieve delay, area, power goals with acceptable noise
- Degrees of freedom:
  - – Width
  - – Spacing
  - – Layer
  - – Shielding



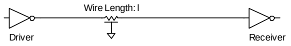


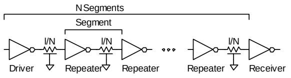
Repeaters

- R and C are proportional to *l*
- RC delay is proportional to l^2
  - – Unacceptably great for long wires

Repeaters

- R and C are proportional to *l*
- RC delay is proportional to l^2
  - – Unacceptably great for long wires
- Break long wires into N shorter segments
  - – Drive each one with an inverter or buffer



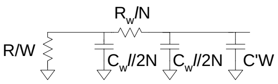


Repeater Design

- How many repeaters should we use?
- How large should each one be?
- Equivalent Circuit
  - – Wire length *l/N*
    - Wire Capaitance C_w*l/N, Resistance R_w*l/N
  - – Inverter width W (nMOS = W, pMOS = 2W)
    - Gate Capacitance C'*W, Resistance R/W

Repeater Design

- How many repeaters should we use?
- How large should each one be?
- Equivalent Circuit
  - – Wire length *l*
    - Wire Capacitance C_w*l, Resistance R_w*l
  - – Inverter width W (nMOS = W, pMOS = 2W)
    - Gate Capacitance C'*W, Resistance R/W



Repeater Results

- Write equation for Elmore Delay
  - – Differentiate with respect to W and N
  - – Set equal to 0, solve

$$\frac{l}{N} = \sqrt{\frac{2RC'}{R_w C_w}}$$

$$\frac{t_{pd}}{l} = \left(2+\sqrt{2}\right)\sqrt{RC'R_w C_w}$$

~60-80 ps/mm

in 180 nm process

$$W = \sqrt{\frac{RC_w}{R_w C'}}$$