Harris

Introduction to CMOS VLSI Design (E158)

Lecture 4: Gates, Capacitance, and Simulation

David Harris

Harvey Mudd College

David_Harris@hmc.edu

Based on EE271 developed by Mark Horowitz, Stanford University

Overview

Reading

W&E 4.2-4.3 Capacitance (this is very detailed, more than we need)

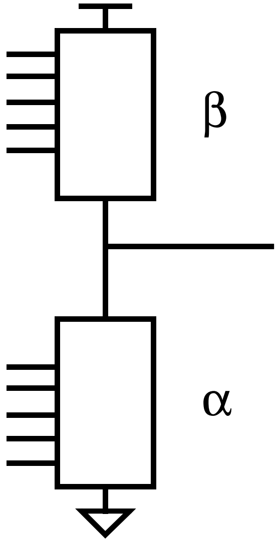
Introduction

Last lecture we built simple NAND and NOR gates. In fact, we can use switch networks to build a gate that implements any boolean function. The key is to realize a CMOS gate is just two switch networks, one to Vdd and one to Gnd. Practically, the kinds of gates that you can construct are limited by the need for stacks of series transistors, and their effect on gate performance. To better understand these issues we next look at capacitance, where it comes from, and how it affects the performance of gates (provides memory, and delay). The last part of the lecture will describe a method of simulating transistor level designs to ensure correctness. The labs will explore this further.

CMOS Gates

To build a logic gate $\bar{f}(x_1, \ldots, x_n)$, need to build two switch networks:

The pullup network connects the output to Vdd when f is false.



pMOS only, since only passes 1

The pulldown network connects the output to Gnd when f is true.

nMOS only, since only passes 0

Notice that the constraints on the two switch networks is just what we talked about for switch logic. The output must be driven $(f + \bar{f} = 1)$, and there can’t be conflicts $(f * \bar{f} = 0)$

Pulldown

$$\alpha(x_1, \ldots, x_n) = f(x_1, \ldots, x_n)$$

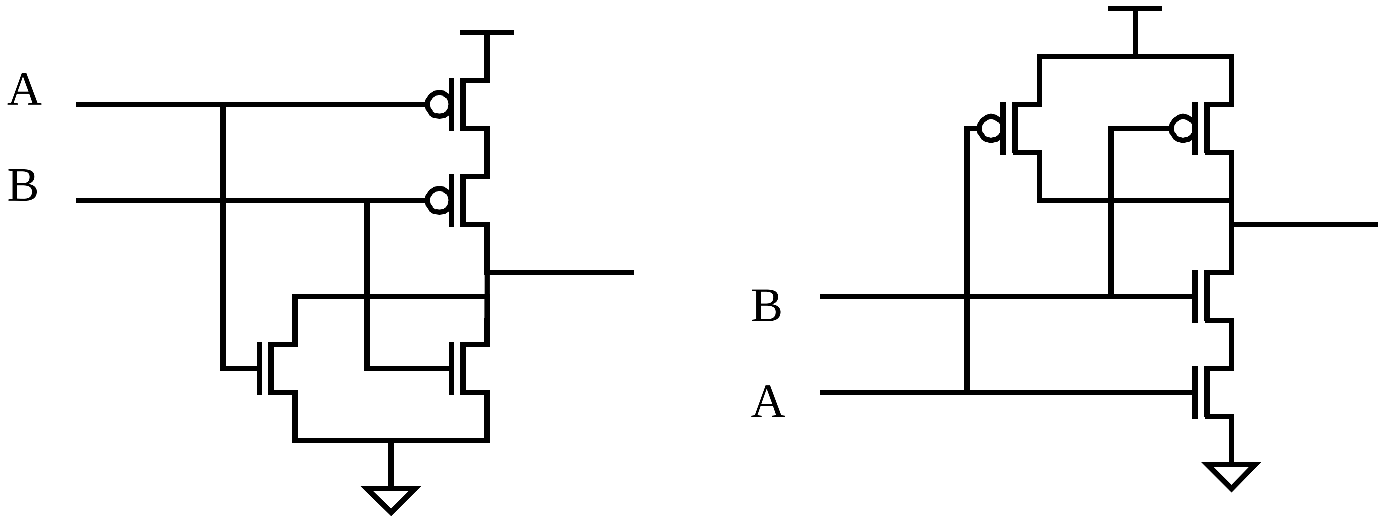
Pullup

$$\beta(\bar{x}_1, \ldots, \bar{x}_n) = \bar{f}(x_1, \ldots, x_n)$$ (since pMOS invert inputs)

CMOS Gate Examples

CMOS NAND and NOR gates

- Need to implement f using (x) and $\bar{f}$ using ($\bar{x}$)
- Series pulldown -> parallel pullup, parallel pulldown-> series pullup



- Easier to build N tree first, because easy to forget P tree inverts inputs.

Gates

The pullup network and pulldown network are duals of each other

- Dual of a function:
  - Exchange AND and ORs
- Example Duals
  - A B ; A + B
  - (A +B) C ; (A B) + C

For switch networks

- AND = series switches
- OR = parallel switches
- So
  - Parallel pulldown, serial pullup and vice versa

Why?

De Morgan’s Law / Duality

Remember DeMorgan’s Law?

$\overline{(a + b)} = \overline{a}\ \ \overline{b}$

$\overline{(a\ b)} \quad = \overline{a} + \overline{b}$

More generally the complement of a function can be obtained by replacing each variable / element with its complement, and exchanging the AND and OR operations

One of the most useful rules in boolean algebra

Can apply to arbitrarily complex expressions.

If element is not a single variable, then apply recursively to the expressions:

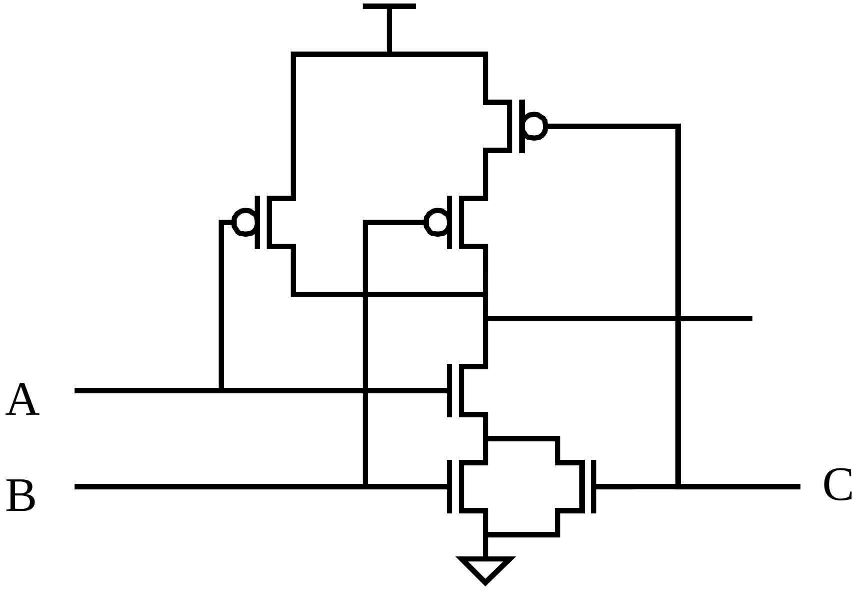
$\overline{(A{+}B)\ C} = \overline{(A + B)} + \overline{C} = (\overline{A}\ \ \ \overline{B}) + \overline{C}$

$\overline{(A\ B) + (C\ D)} = \overline{(A\ B)}\quad \overline{(C\ D)} = (\overline{A} + \overline{B})\ (\overline{C} + \overline{D})$

CMOS Gates

The pullup and pulldown switch networks are duals

- Since $\overline{f}(x_1, \ldots, x_n) = \text{DUAL}\ \{ f \}(\overline{x}_1, \ldots, \overline{x}_n)$, and pMOS invert inputs
- $\alpha(x_1, \ldots, x_n)$ is dual of $\beta(\overline{x}_1, \ldots, \overline{x}_n)$
- Example of a complex gate - $\overline{A*(B+C)}$ (called Or-And-Invert, OAI)

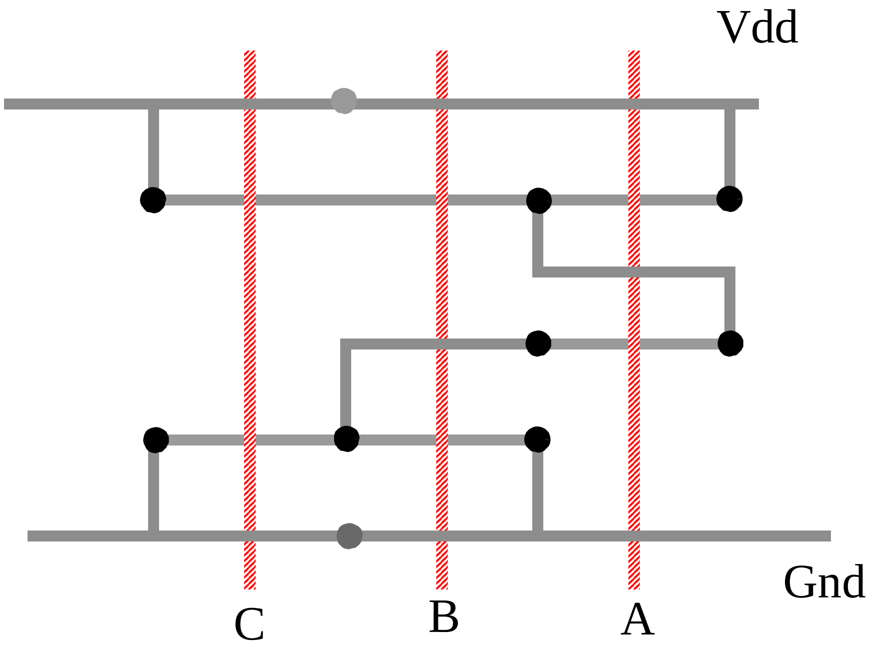


Notice that there are no real required ratio rules in CMOS; the pMOS transistors never fight against the nMOS transistors. But resistance is still an issue with the performance of the gate, and so you usually want the pulldown and pullup resistances to be similar. This resistance is also why gates with a large number (> 3) of series devices are bad.

Either pullup or pulldown will have stack height of about # the of inputs

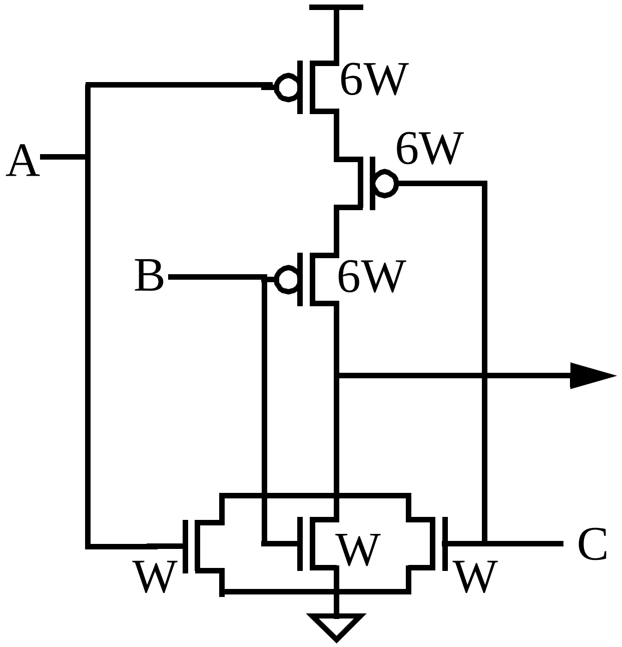
Stick Layout

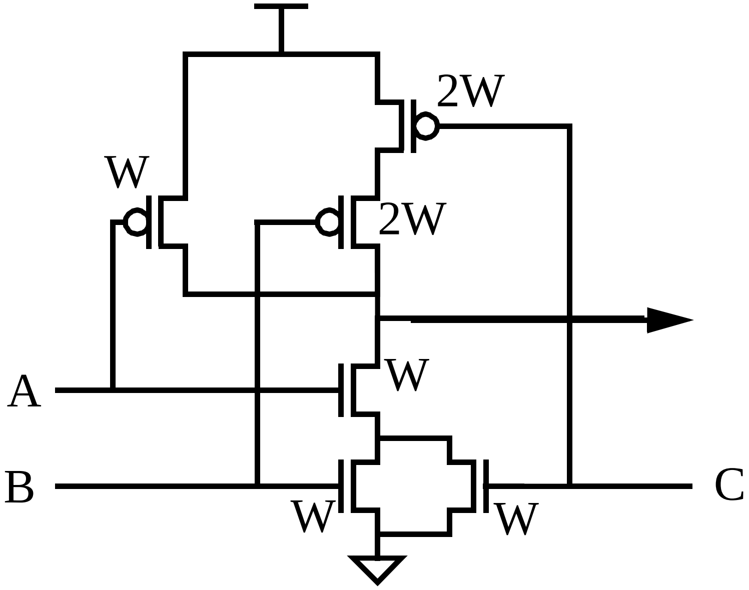
Layout of $\overline{A*(B+C)}$



+ Transistor Sizing in Static CMOS

Attempt to equalize pullup and pulldown resistance.





Sizing here only influences delay, not functionality. So, it can be varied.

Example: AOAOAOI gate

Sketch transistor-level schematic of gate to compute $Y = \overline{A+B(C+D(E+FG))}$

This is useful in adders. G4 = $G_3 + P_3(G_2+P_2(G_1+P_1G_0))$

Transistor widths?

Example: AOAOAOI gate stick diagram

Sketch stick diagram

Complex Gates

In theory can build any inverting logic function in a single gate

- Take the complement of the function
- Build a switch network out of nMOS devices and connect between Gnd and Out
- Build the dual switch network out of pMOS devices and connect between Vdd and Out

In practice the number of gate types is limited

- Want a finite number of gate types (need to design/test/layout them)
- One complex gate can be SLOWER than a couple smaller gates.

Let’s try to understand why this might be so…

Circuits

Resistance

- Relates current to voltage ($V = IR$)
- Wider transistors have lower resistance
- Series structures are not good for speed since the resistance of a series switch network is the sum of the transistor resistances.

But resistance is only part of the stuff you need to model circuits. The other important property is capacitance.

Capacitance

- Relates charge to voltage ($Q = CV$)
- Exists between any two conductors
- Causes delay in circuits ($t = RC$) and data storage (memory)

Capacitance Equations

Capacitors store charge

$Q = CV$ <- charge is proportional to the voltage on a node

This equation can be put in a more useful form

$$i = \frac{dQ}{dt} \Rightarrow i = C\frac{dV}{dt} \Rightarrow \frac{C\Delta V}{i} = \Delta t$$

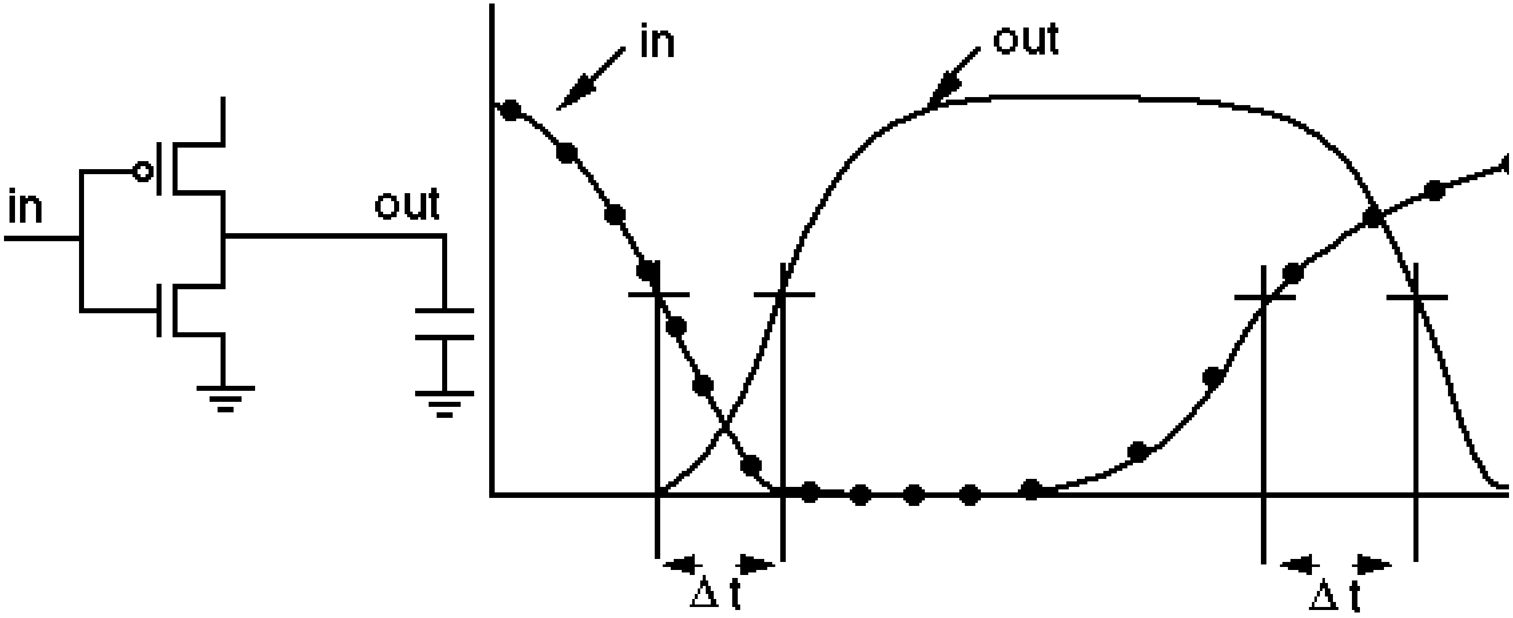
So to change the value of node (from 0 to 1 for example), the transistor or gate that is driving that node must charge (up, in our example) the capacitance associated with that node. The larger the capacitance, the larger the required charge, and the longer it will take to switch the node.

Since the current (i) through a transistor is approximately V/R_{trans}

$$\Delta t = \frac{C\Delta V}{i} = \frac{C\Delta V}{V/R} = R_{trans}C$$

Simple Delay Model

- Delay measured from 50% crossing point on input and output swings, because need the same point to allow additive composition of delays.
- Define R_{sq} of a transistor so RC gives the right delay values



- For our 1μ technology, nMOS = 13KΩ ∗ L/W, pMOS = 26KΩ ∗ L/W

Load Capacitance

C_{load} comes from three factors:

1. Gate capacitance of driven transistors.

2. Diffusion capacitance of source/drain regions connected to the wire.

3. Wire capacitance

Today, a 1.5μ technology is the really cheap technology that students use, and advanced processes are running at 0.5μ to 0.18μ. We will use 1μ technology numbers for this class.[1] This technology is different from the numbers in the book.

The ratio of the various numbers does not change much with technology, but the absolute numbers do vary. You should always find the correct numbers for the technology that you will use before starting a design. And, since you don't want to extract the C_{load} numbers by hand, make sure that the CAD tools have the right numbers too.

[1] The metric that I will use in class, resistance/square for transistors, and capacitance/micron don't change much with technology scaling. For a 0.25μ technology Rsq of a nMOS device is 15K, pMOS is 36K, which is similar to the 1μ numbers. The cap/micron numbers are nearly the same. The reason the gates get faster is that the cap/lambda goes down, so the cap of a 10:2 device scales down, while the resistance remains constant.

+ Calculating the Value of Capacitance

Two simple models

- Parallel Plate
- Cylindrical

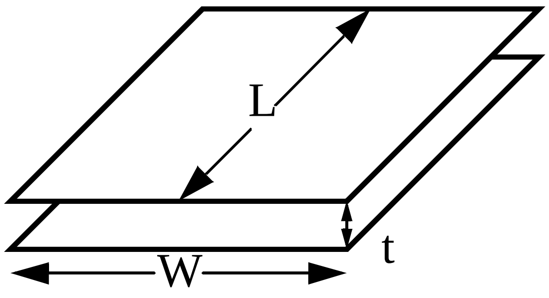
The capacitance of most real objects can be approximated by a combination of these two factors.

- Parallel Plate[1]

$$C = \frac{\varepsilon\, L * W}{t}$$

Fixed by technology

$$C = C_{per\_square\_micron} * W * L$$

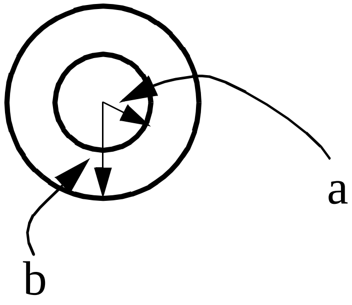


[1] The capacitance can be found by solving Laplace’s equation. For an infinite parallel plate capacitor, the E-field does not vary in the vertical direction, and hence the voltage is proportional to the thickness.

+ Cylindrical Capacitance

This is the model of capacitance between two cylinders[1]

$$C = \underbrace{\boxed{\frac{2\,\pi\,\varepsilon}{\ln(b/a)}}}_{\text{Constant}}\, L$$

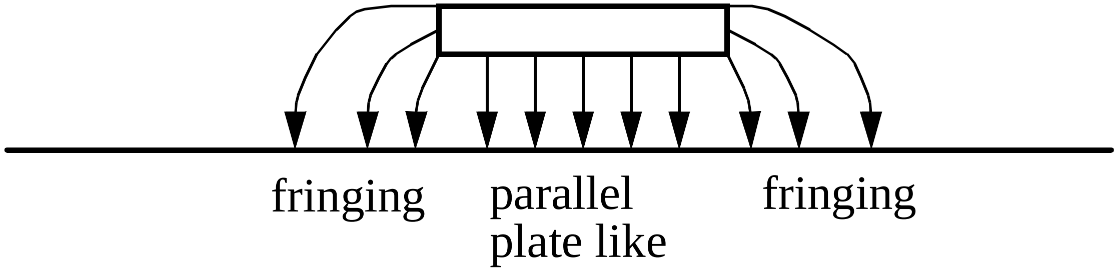


This gives a capacitance that is proportional to length. It is also not very sensitive to the ratio of b/a, so making b much larger than a still does not reduce the capacitance much.

This type of capacitance can be used to model fringing fields of wires – this is the capacitance from the edge of the wire.

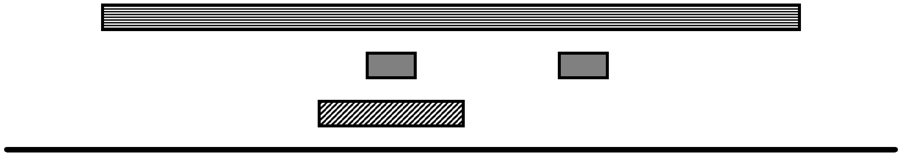
[1] The result can be found by solving Laplace’s equation in cylindrical coordinates. In this space the E- field falls off as 1/r (where r = b/a), and thus the voltage (integral of the field) varies as the log of the distance.

Real Wires



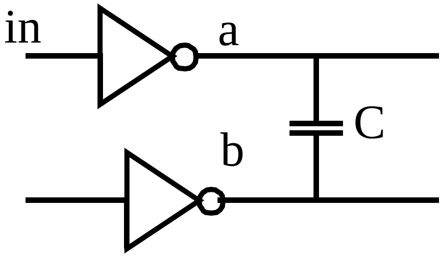
So, wires have two components to capacitance, one that is proportional to the wire’s area, and the other proportional to the wire’s perimeter. **For minimum width metal wires, the edge component is much larger than the area component**, so forgetting the edge is a large error.

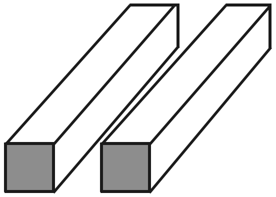
The area capacitance depends on the thickness of the oxide between the capacitor plates, and that thickness depends on what is below it.



+ Coupling Capacitance

Capacitance is mostly between two wires, not between a wire and ground





Coupling capacitance makes analysis more complex:

- It creates noise issues
  - ‘a’ changing will cause noise on ‘b’
- It makes delay calculations harder
- If ‘a’ and ‘b’ transition at the same time in same direction
  - ΔV across the cap will be zero, and it won’t affect the delay
- If ‘a’ and ‘b’ transition at the same time in opposite direction
  - ΔV across the cap will be 2V, and it will look like a grounded cap of 2C

Wire Capacitance

Most CAD systems have tools that take care of all this complexity by using large tables of numbers, one for each type of legal layer crossing. The tools take the capacitance numbers, multiply by the correct area and perimeter coefficients and then add all the numbers together.

That is far too much work for us to do by hand. Also it requires that the layout be finished to get the capacitance numbers. We often want to estimate the capacitance numbers to size transistors from either crude layout, or layout estimates.

Since most of the wires are minimum width, we will use an effective capacitance per running micron of length, assuming an average number of wire crossings. This number will include both the area (plate) and perimeter (fringe) capacitance terms. Diffusion is treated almost the same, but the width for diffusion we assume to be the extension of a transistor drain, and the length should therefore be the width of the transistor. This assumes that diffusion is only used to make a connection to a transistor, which is normally the case since the diffusion capacitance is so large.

Simple Capacitance Numbers

Want to have numbers that make it easy to estimate the capacitance

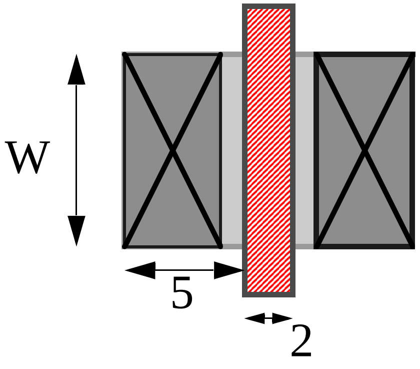
- Want the estimates to depend on the fewest number of parameters
- Willing to make some approximations

For wires

- Most wires are minimum width
- Large edge component of capacitance anyhow
- So makes sense to measure capacitance per unit length

For transistors

- Gate length is usually minimum (2λ, 1μ), width varies
- Diffusion region kept small, size depends on transistor width
- Give cap of these region as capacitance per unit transistor width



Rule of Thumb Capacitance Table

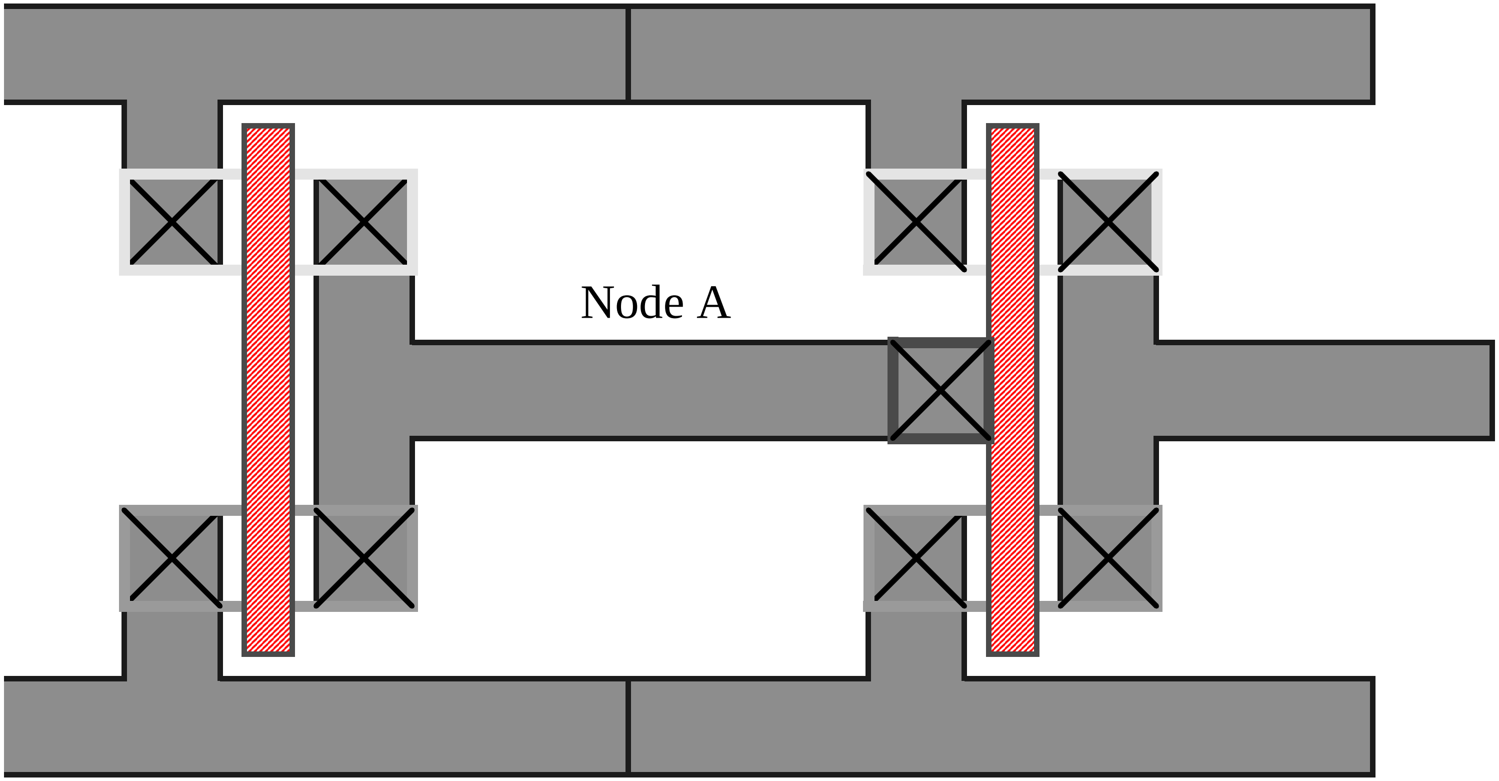
Table 1:

Transistor Cap	Capacitance per μ of transistor W
gate (poly over diff)	**2.0 fF**/μ
ndiff (5λ or 6λ wide)	**2.0 fF**/μ
pdiff (5λ or 6λ wide)	**2.0 fF**/μ

Wire Cap	Capacitance per unit length	Length when $C = C_{inv}$
poly wiring (2λ wide)	**0.2 fF**/μ	40μ
metal1 (3λ or 4λ wide)	**0.3 fF**/μ	27μ ~30μ
metal2 (3λ or 4λ wide)	**0.2 fF**/μ	40μ

C_{inv} is 8fF, the input capacitance of a 4λ:2λ nMOS, 4λ:2λ pMOS inverter

Example



Node A: 2 diffusion regions each 2μ (4λ), 2 gate regions each 2μ, 16μ M1 (12λ vertical, 20λ horizontal), 12λ poly= 2*2μ*2fF/μ + 2*2μ*2fF/μ +16μ *0.3fF/μ + 6μ*0.2fF/μ = 8fF + 8fF + 4.8fF+1.2fF = 22fF

Another Example

What if:

PMOS 16/2

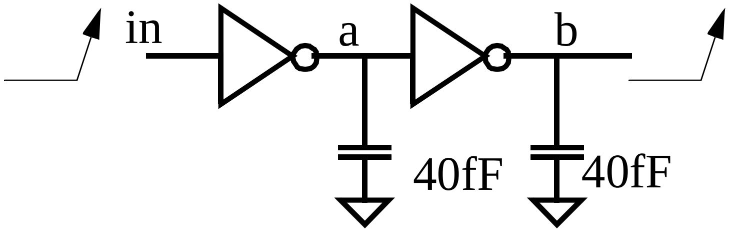
NMOS 8/2

Additional 50 microns of Metal2 between the gates

C =

Timing Example

Assume that all transistors are 4:2λ



40fF includes the diffusion and gate cap

When the ‘in’ rises, ‘a’ will fall:

delay = RC = 13K/2 * 40fF = 0.26ns (nMOS transistor is on)

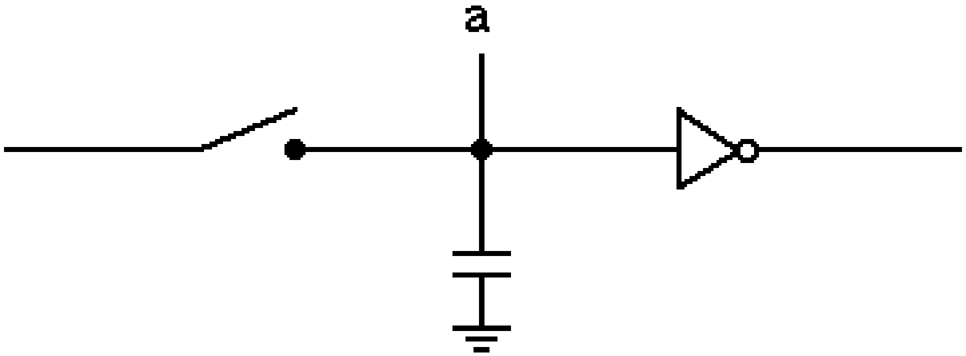
When ‘a’ falls ‘b’ will rise:

delay = RC = 26K/2 * 40fF = 0.52ns (pMOS transistor is on)

Total delay from ‘in’ to ‘b’ = 0.26ns + 0.52ns = 0.8ns

Dynamic Charge Storage

What happens to the value on node ‘a’ when the switch disconnects?



When the switch is off, the current driving the capacitor is zero

- i = CdV/dt

  dV/dt = 0, so the voltage remains unchanged

- The value remains unchanged

  That is, when you stop driving a node its value remains unchanged, and remains almost the same until it is driven again. This is the good part of capacitance.

+ Charge Leakage

There is no leakage current from the gate of a MOS transistor but the source/drain terminals do have a small leakage current.

Leakage current is very small, usually picoAmps

- Charge will leak away, but very slowly

  Storage times are usually about **1 second** at Room Temp

- Leakage is temp sensitive

  Doubles every 10^{o}C

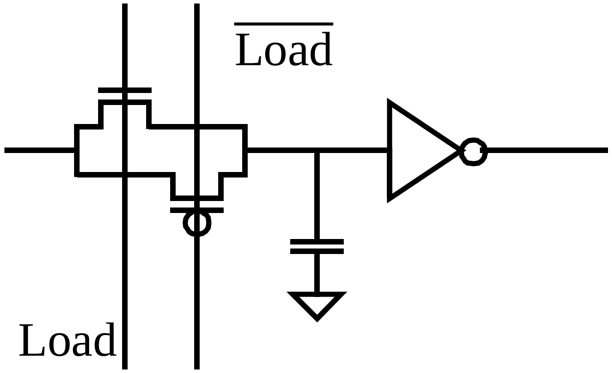
  10ms at 70^{o}C

Leakage is much slower than the clock rate. Dynamic store is ok, if the node is reloaded every few clocks. If you can’t guarantee when you will reload the storage node, you had better use a different storage element.

Make sure you don’t build these storage elements by accident (by leaving a node floating).

Latches

A simple dynamic latch



We will talk more about this later

A switch can be made by using a full CMOS transmission gate

- No degraded levels (like from using just a single nMOS pass transistor)
- But two control signals needed

Problems with Transistor Switch Circuits

If you obey all the rules for switch logic, the circuits generally work well.

- Make sure there are no floating outputs, no drive conflicts
- Sometimes you make errors …

Switches are bidirectional

- Generally, designers would like to think that information only flows one direction, but the transistors don't care about the intent.
- Leads to errors, some pretty subtle because they all depend on the capacitance ratio.
- Look at two problems with capacitance

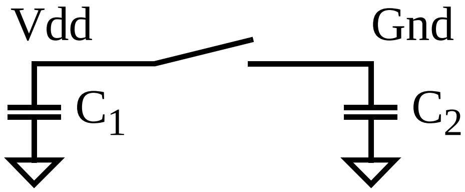
  Charge sharing (pure)

  Charge sharing (driven)

Charge Sharing

What happens when you connect two capacitors together

(without any connection to a supply)



- Charge must be conserved

  C_1 Vdd = (C_1 + C_2) V_{final}

  V_{final} = Vdd * C_1/(C_1 +C_2)

- So either

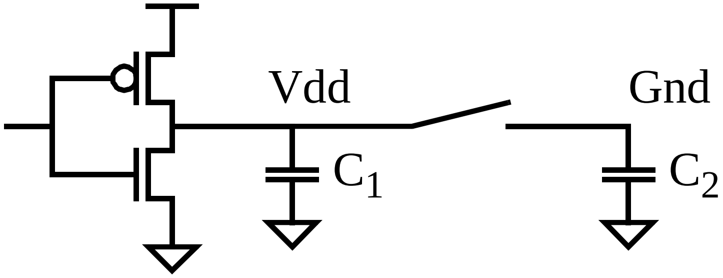
  C1 >> C2; both nodes become 1   >> requires approx 4x ratio

  C2 >> C1; both nodes become 0

  otherwise you will get an undefined value.

Driven Charge Sharing

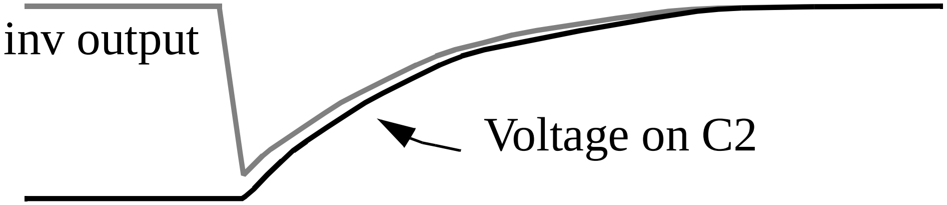
You can get charge sharing even if you are driving the node



If the resistance of the switch is small compared to the driving transistor, and C2 is larger than C1, then there is momentarily a resistive divider.

- C2 will first drive C1 to Gnd

Then (more slowly) the pMOS will drive both capacitors high



Need Switch-Level Simulation

Need some way to check the circuits to see if we built them correctly

- Nice if the method could handle all legal switch circuits.
- Want to find common bugs in circuits, yet not produce false errors.
- Be fast and easy to use.

Tool should answer the questions:

- Does this pile of transistors do the logic function that I want?
- Are there any sneak paths, floating outputs?

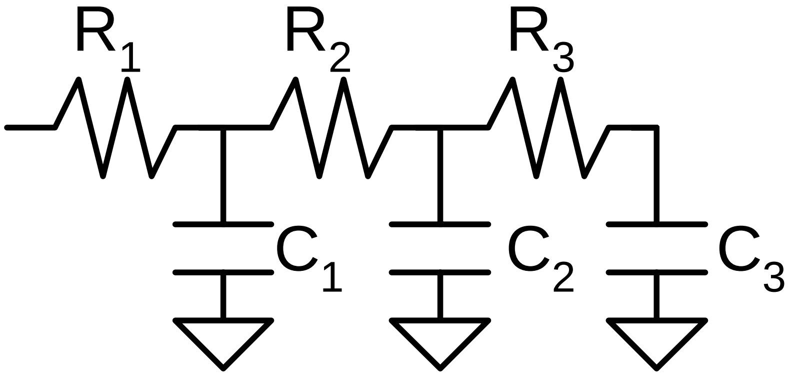
Switch-level simulation is one good way to answer these questions.

- Uses the same type of model that we have been talking about in class:
- Nodes are modelled as capacitors
- Values on the nodes are 0,1,X
- Transistor is modelled as a switch

Elmore Delay Model

Return to estimating delay.

Calculate delay of RC network:



Elmore Delay Model

For these examples, ignore wire capacitance.

Estimate the delay of:

- inverter driving no load
- inverter driving identical inveter
- inverter driving four identical inverters
- 2-input NAND gate driving no load
- 2-input NAND gate driving four identical gates