
Lecture 4

Estimating Energy Dissipation and Delay

Adapted from: Mark Horowitz
with contributions from Subhasish Mitra

Overview

Reading

W&H 4.4-4.4.2 – Power dissipation

W&H 2.4.4 – Leakage current

W&H 2.6 – Delay

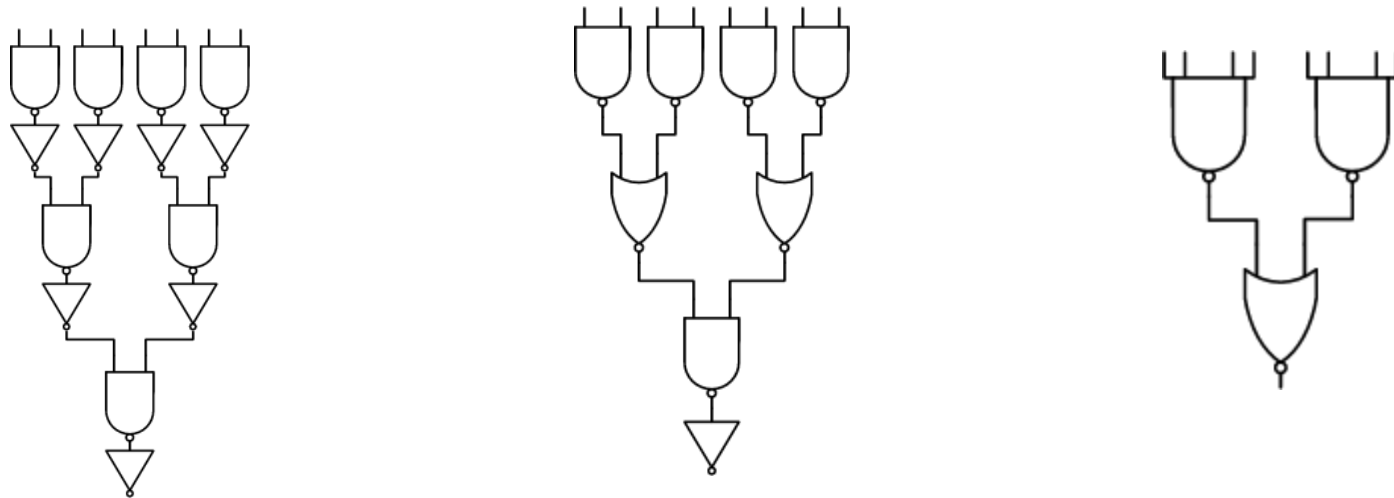
Introduction

There are many ways to construct any logical function. What separates a good design from a bad one is the amount of resources that the design consumes. In VLSI there are four kinds of resource you want to conserve: designer sanity, power, delay, area. These are listed roughly in order of importance.

This lecture will look at how one can estimate the energy and delay of primitive gates from the R and C that we discussed in the previous lecture. We will end this lecture by discussing the desire to work at a higher level of abstraction, so we can have a tool do most of this stuff.

How To Create Good Logic Functions

- There are many ways any function can be implemented
- These all produce the same function:



- In fact using our rules from last lecture
 - Should be able to make ANY inverting function in ONE gate!
(But this will turn out to be a bad idea.)

What To Optimize?

There are four resources you need to manage

- List them in most important to least important

1. Design time / designer sanity / design cost

- How long will it take you to create and **validate** design
- Simplicity goes a long way here
- Reuse is good too, but need to validate in new environment

2. Power dissipated / energy per operation

- Most designs today are running into power / energy limits
- Caused by scaling, and by the desire for portability
 - Energy is heavy, and no one wants a hot phone

What To Optimize, cont'd

3. Performance / delay / cycle time

- You system need to function at some rate
- You circuit will consume some time
- Need to allocate delay wisely (to minimize energy)
- Need to divide delay into multiple clock cycles (pipeline)

4. Area

- Transistors are small, but still take some space
- The larger the die
 - The more it costs
 - The larger capacitance the wires will have
 - Which means it will run slower and require more energy

Validation is the Main Task

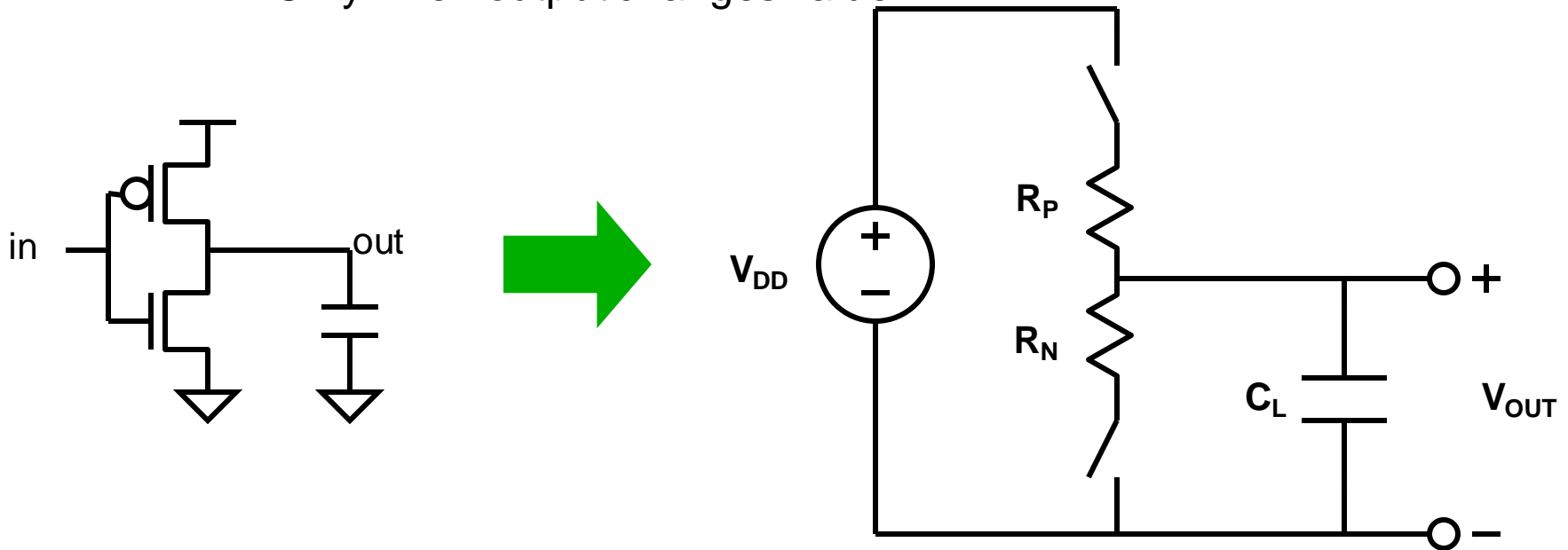
- Your design is going to be complex
 - Hard to build something that has 100M devices that is simple
- Complicated things are just complicated
 - It is hard to test all the corner cases
 - Look at software
 - It is hard to figure out whether they are really working
 - Shipping buggy hardware is a no-no
 - Unlike software where it can be a feature
 - Spend most of your effort making sure the chip works
- So you should design the hardware so you can test it
 - Functional and manufacturing test are both important

CMOS Power / Energy Dissipation

- Power = Energy dissipated / sec
- In CMOS have two sources of energy loss
 - Dynamic power (dominate)
 - Leakage power
- Easy to find overall chip power
 - Simple sum of power dissipated by each gate

Dynamic Power

- Major form of energy dissipation in CMOS circuits
- Capacitors are reactive devices
 - They don't dissipate any power
 - Energy dissipated when we charge/discharge the cap
 - Only when output changes value



The Hard Way To Solve The Problem

- Current flows from supply through PMOS to charge

$$V_{OUT} = V_{DD}(1 - e^{-t/R_P C})$$

$$I_{DD} = (V_{DD} - V_{OUT})/R_P$$

$$I_{DD} = V_{DD}/R_P(e^{-t/R_P C})$$

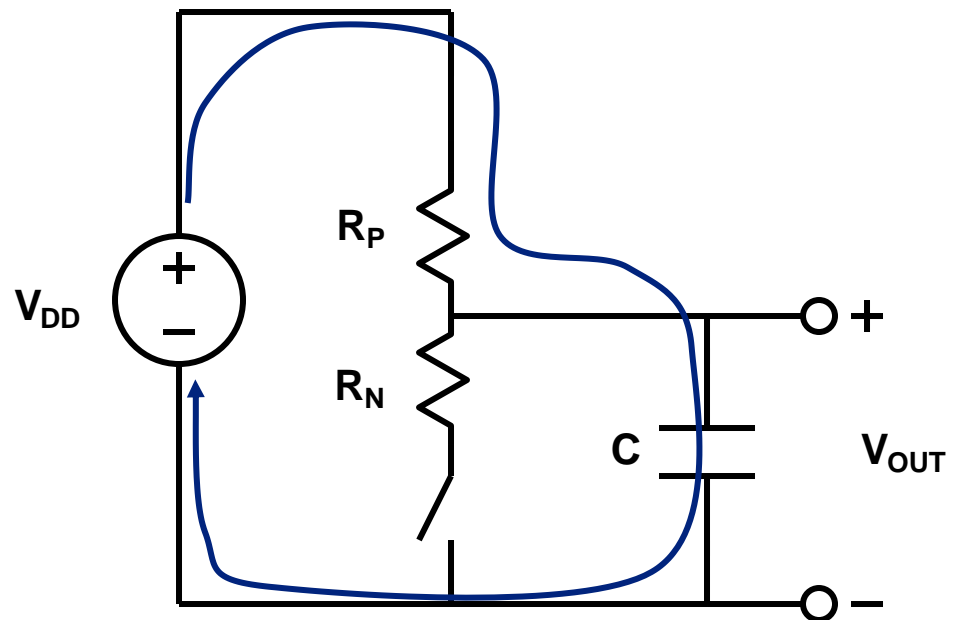
$$\text{Power} = I_{DD} * V_{DD}$$

$$\text{Avg Power} = \frac{\int_0^{t_{CYC}} V_{DD}^2 / R_P (e^{-t/R_P C}) dt}{t_{CYC}}$$

$$\text{Avg Power} = C V_{DD}^2 f (1 - e^{-t_{CYC}/R_P C})$$

$$\text{Avg Power} \approx C V_{DD}^2 f \quad \text{if } t_{CYC} \gg R_P C$$

Not a function of R_P

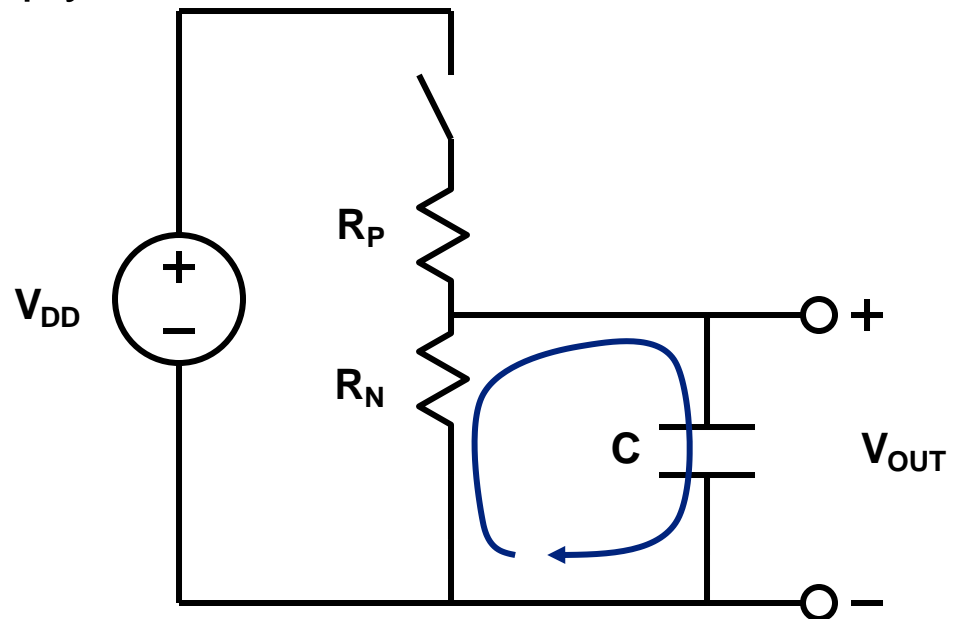


Capacitive Charge: 1->0 Transition

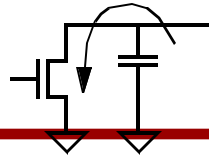
- Current flows from supply through NMOS to discharge
- Current path is entirely on chip
- No current drawn from supply

$$V_{OUT} = V_{DD} (e^{-t/RnC})$$

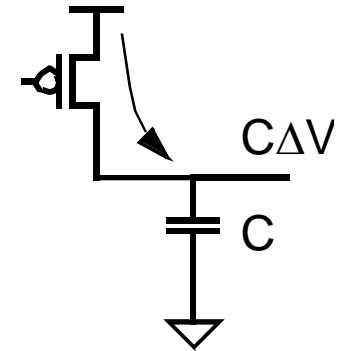
$$I_{DD} = 0$$



The Better Way



- Power = Energy/time
- Energy = $Q * V$
 - It takes energy to add charge to a voltage source, and the amount of energy needed is proportional to V , Q
- For a gate
 - When you charge up the output (0- \rightarrow 1), take a charge $C * V_{dd}$ out of the V_{dd} supply (since it is now on the output capacitance)
 - When you discharge the output (1- \rightarrow 0) that charge is returned to Gnd
 - Gate dissipated CV_{dd}^2 energy

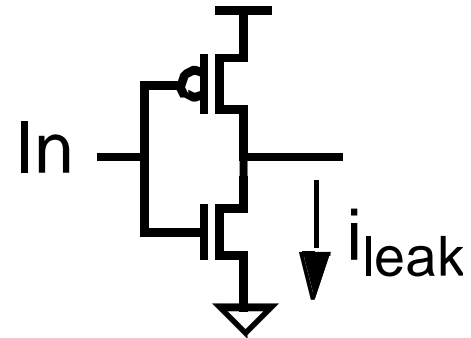


CMOS Dynamic Power

- For each 0->1->0 cycle of a node
 - Takes CV_{dd}^2 energy
 - If nodes don't transition, no energy is dissipated!
- Let α = # transitions / clock cycle
 - α is generally less than one for most circuits
- Power = $\frac{1}{2} \alpha CV_{dd}^2 F$
- WHAT ABOUT THE CLOCK NODE?
 - α is ? for a clock node

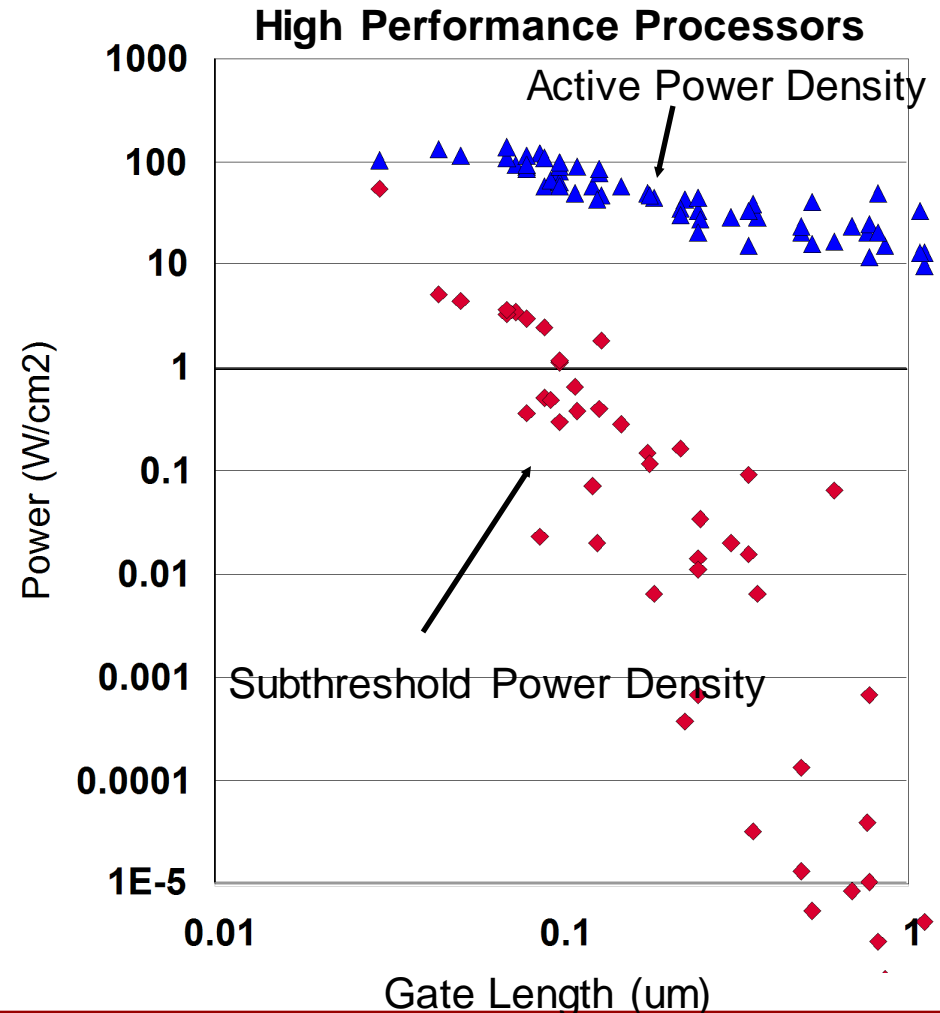
Leakage Power

- Transistors don't turn completely off
 - When V_{gs} is less than V_{th}
 - Both nMOS and pMOS leak
- Leakage depends on V_{th}
 - $I_{leak} = I_0 \exp(V_{th}/\kappa v_T)$
 - $v_T = kT/q = 25\text{mV}$ at room temp; $\kappa \sim 1.3$
 - Larger V_{th} decreases leakage
 - Drop 10x in current for each 100mV increase in V_{th}
 - But it also increases transistor resistance ($V_{dd}-V_{th}$ decreases)
- Previously we were scaling down V_{dd} and V_{th}
 - Which increased the leakage power
 - But allowed us to scale V_{dd} , which decreased total power



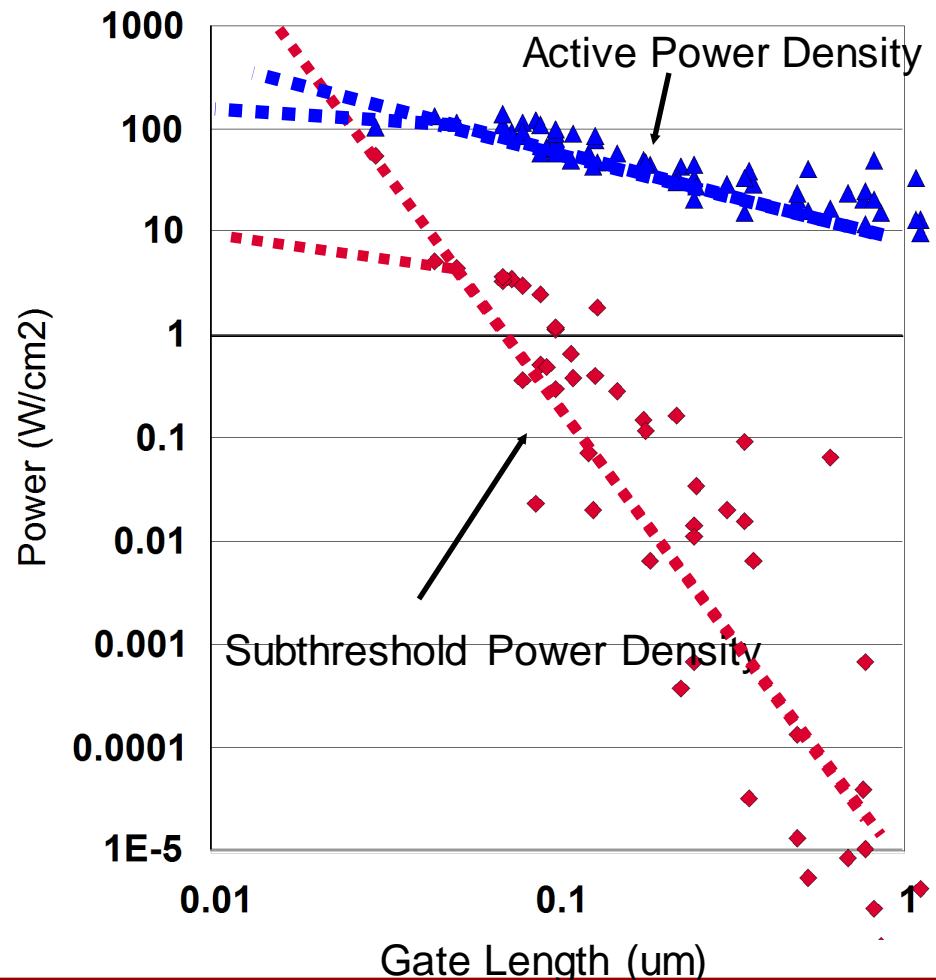
Power / Energy

- Historically:
 - Active (dynamic) power dominated chip power
 - Leakage power was small, but increased exponentially as V_{th} was scaled down (to allow V_{dd} to scale down)



What Won't Happen

- Now
 - We stopped scaling V_{th}
 - And V_{dd}
 - Leakage is ok
 - Issue is dynamic power
- Leakage current is not going to dominate total power dissipation.
- The raising leakage power was done to **reduce** the total power of the system.



Power Problem w/ Scaling

- Because of leakage, V_{th} is not scaling
 - This makes it harder to scaling V_{dd}
 - Lower V_{dd} means slower gates (explained next)
- If V_{dd} does not scale
 - In a new technology which is $\frac{1}{2}$ the feature size
 - Each gate will dissipate an energy
 - $\frac{1}{2} \alpha C V_{dd}^2$
 - Which if V_{dd} does not scale is $\frac{1}{2}$ previous energy
 - Cap will scale with technology
 - But we can fit 4x the number of gates
 - And we hope that the gates run faster
 - Total power will rise, unless we create better designs!

Gate Power / Energy Summary

- Power is energy / second
- In CMOS circuits
 - Energy is used to **change** output voltage
 - $\frac{1}{2} C_{\text{load}} V_{\text{dd}}^2$
 - Power is then energy * transactions/sec
 - $\frac{1}{2} C_{\text{load}} V_{\text{dd}}^2 \alpha F$ α = probability node changes each cycle
- Leakage is a constant power
 - $I_{\text{leak}} * V_{\text{dd}}$
 - I_{leak} is related to the total transistor width connected to Gnd or Vdd
 - Depends on whether the output is high (W to Gnd) or Low (W to Vdd)
- To reduce power
 - Low activity, low capacitance, low Vdd, and low leakage

Chip Power

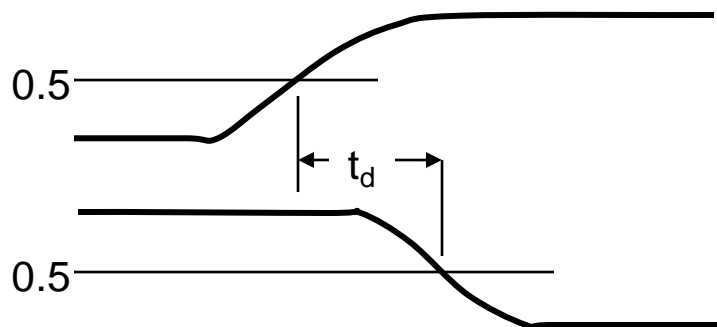
- Easy metric to estimate
 - Total power is simply the sum of all the gate power
- Only problem is to estimate transition probabilities
 - And wire capacitance if you don't have physical design

Chip Performance

- Need to determine gate performance
 - Define gate delay as:
 - Time from input crossing $V_{dd}/2$ to output crossing $V_{dd}/2$
 - Depends on RC time constant
- Sizing transistors in gates for equal rise and fall times
 - Need to worry about worst-case delay
- R will depend on V_{dd}
 - Important since might want to reduce V_{dd} to save power
- Chip performance is the critical path through logic gates
 - Which is a simple sum of **some** of the gates in the circuit

Gate Performance

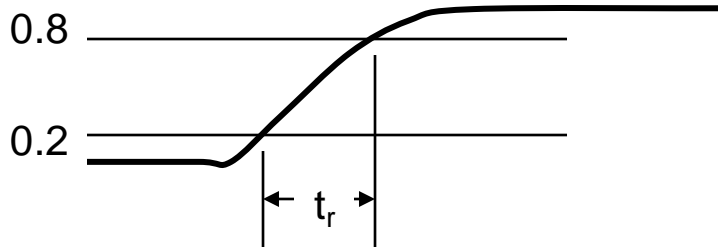
- Need a metric where we can easily add delay
 - Delay of a path should be $t_{d1} + t_{d2} + t_{d3} \dots$



Delay, t_d , is measured from 50% point to 50% point. Gives a measure that is composable. Define R_{sq} so $R_{trans}C$ product is roughly equal to the delay t_d

- For a single gate delay t_d
 - Delay is roughly equal to $R C$
 - But which R and which C ?

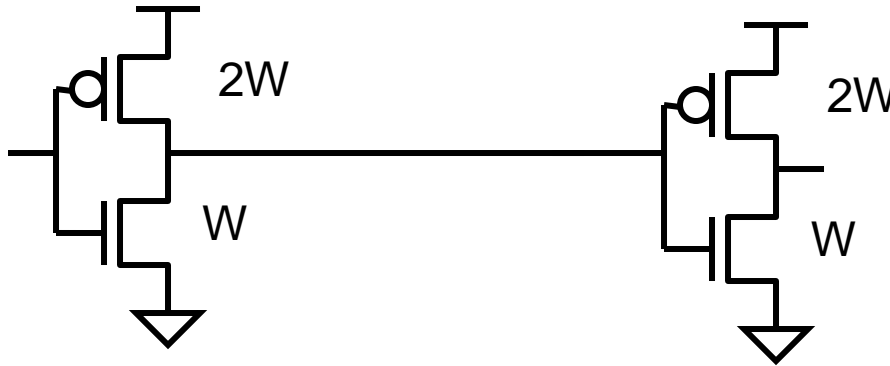
+ Aside: Rise/Fall Time



Rise time and fall time, t_r and t_f , are measured from 20% point to 80% point. But they are not generally used in digital design.

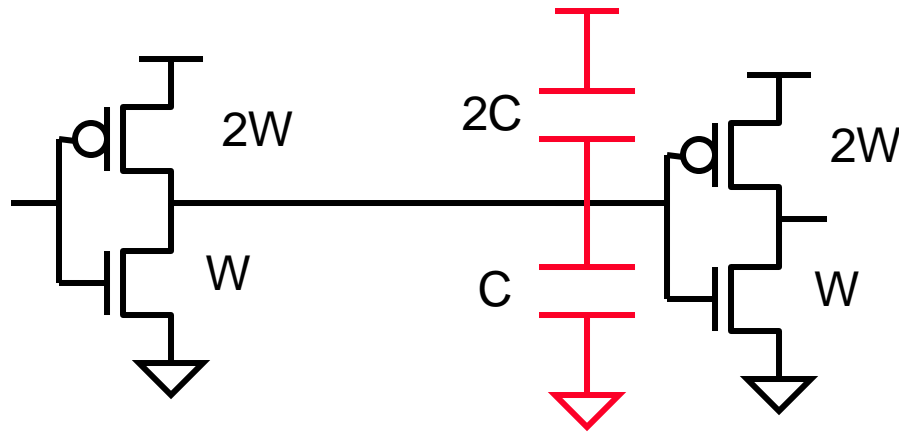
- Some people talk about rise and fall times
 - While these are interesting numbers
 - Can't add them to get path delays
 - They don't start/end at the same point
- We won't use them

Let's Look at a Simple Example



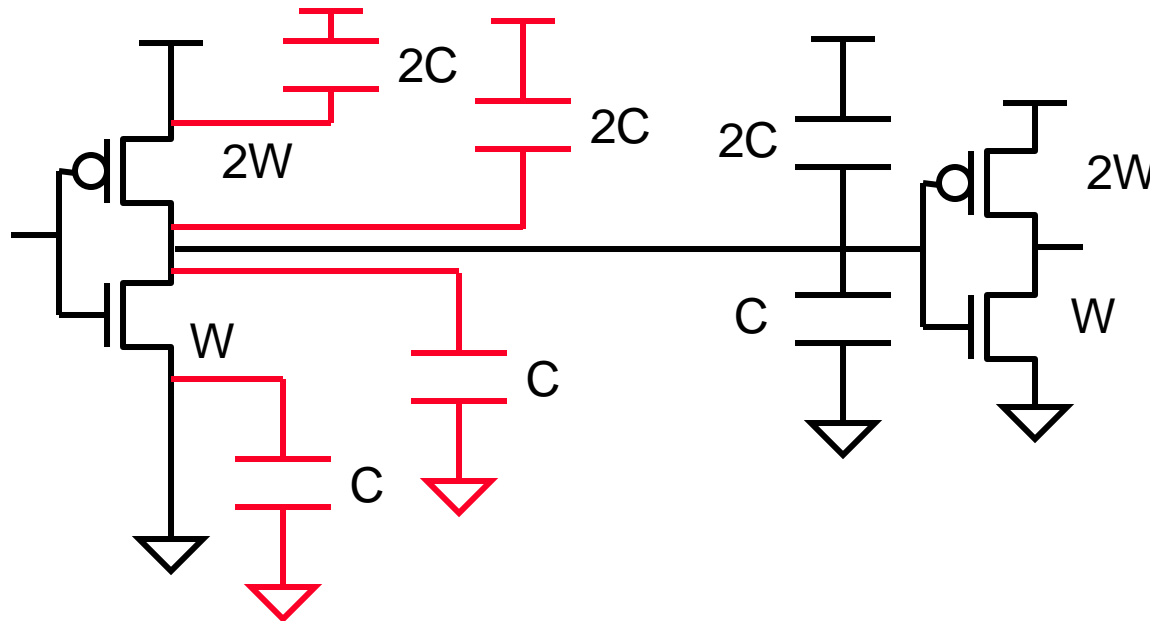
- Transistors are sized so
 - Pullup and pulldown resistances are matched
 - This is called a fanout of one, since it drives the same size gate
- Need to find resistance and C_{load}
 - Assume the inverter is driving an identical gate

Gate Capacitance



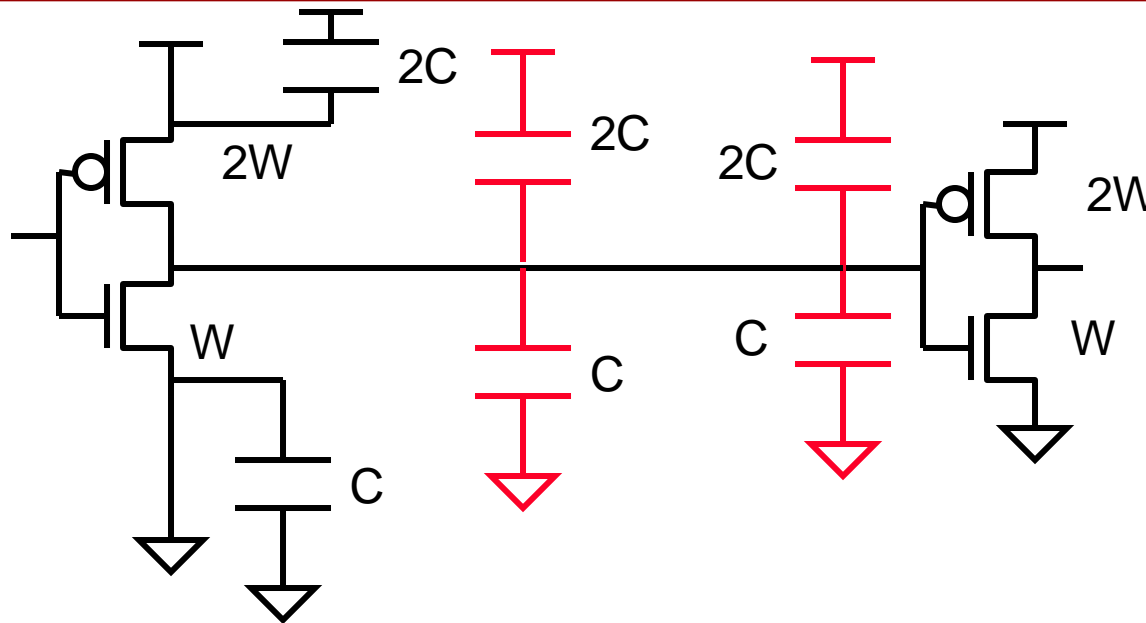
- Connected to a similar inverter
 - Gate capacitances of pullup and pulldown
 - Proportional to corresponding gate width
 - Here C = gate capacitance of a transistor of width W

Diffusion Capacitance



- In addition to gate caps of driven transistors,
 - Need to consider parasitic caps of driving transistors
 - Comparable to gate capacitance (between $\frac{1}{2}x$ and $1x$)
 - We will use $1x$

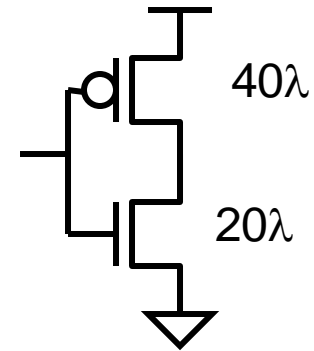
Cload



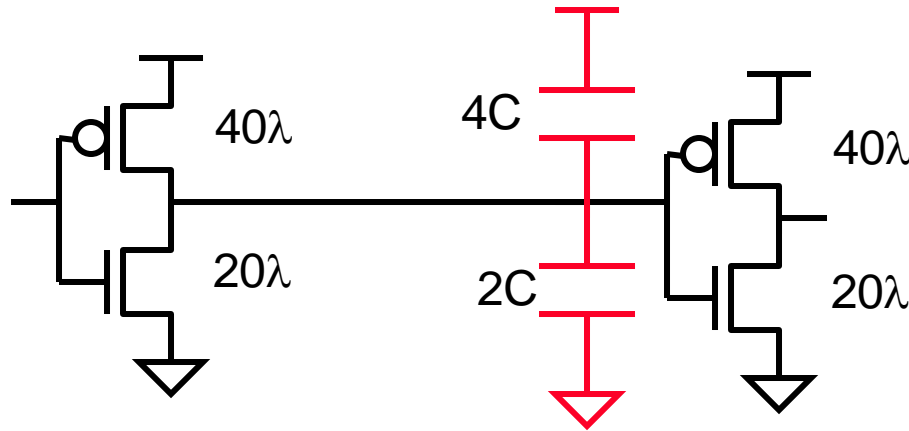
- Only the red colored ones matter
 - Since those are the only ones that change value
 - Total cap = $6C$ (assumed no interconnect cap)

Resistance of Transistor

- We will assume that pMOS $R_{sq} = 24K$
 - Rather than $26K$
 - Matches our 2 to 1 ratio
- $R_{down} = 12K * 2/20 = 1.2K$
- $R_{up} = 24K * 2/40 = 1.2K$

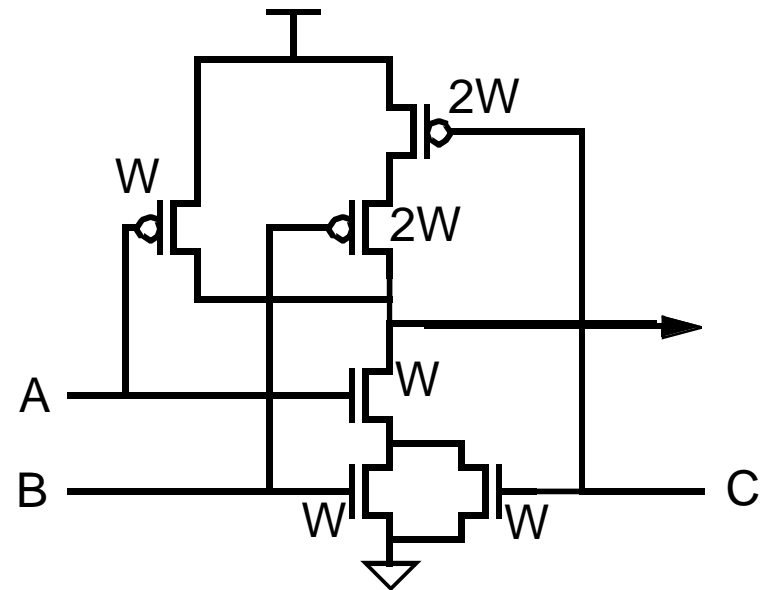
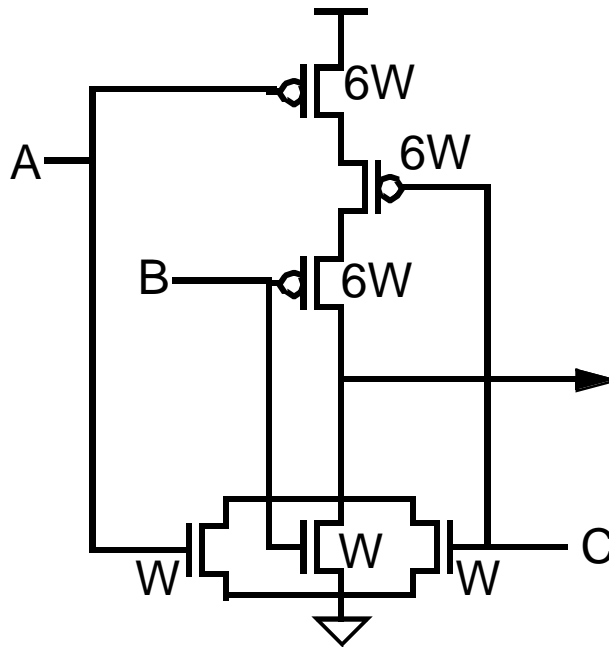


Delay



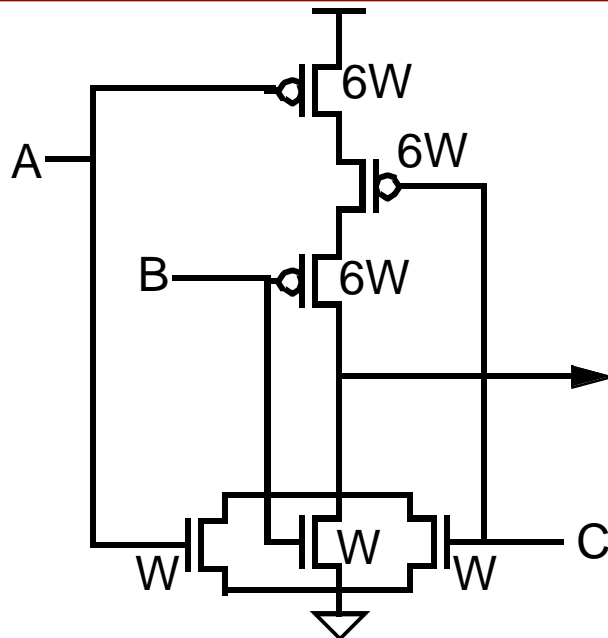
- $C = .5\mu\text{m} * 1.2 \text{ fF}/\mu\text{m} = 0.6\text{fF}$
 - Total is 3.6fF
- $RC = 1.2\text{K} * 3.6\text{fF} = 4.3\text{ps}$
 - That is pretty quick
- What happens if you double the size of all the transistors?

Transistor Sizing inside CMOS Gates

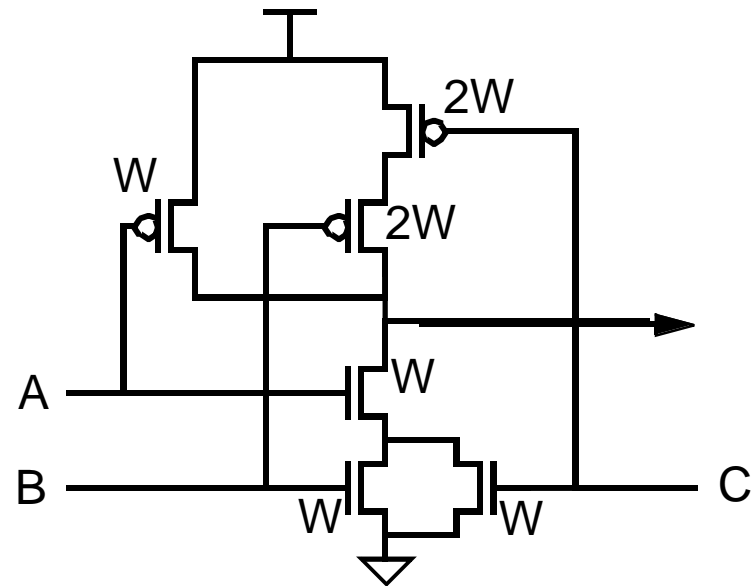


- Attempt to equalize **WORST-CASE** pull-up and pull-down resistance
 - For parallel transistors, worst case when **only one** of the parallel transistors is on – **Worst-case limits delay**
 - Worst-case pullup and pulldown resistances match for above gates

Resistance: nMOS vs. pMOS



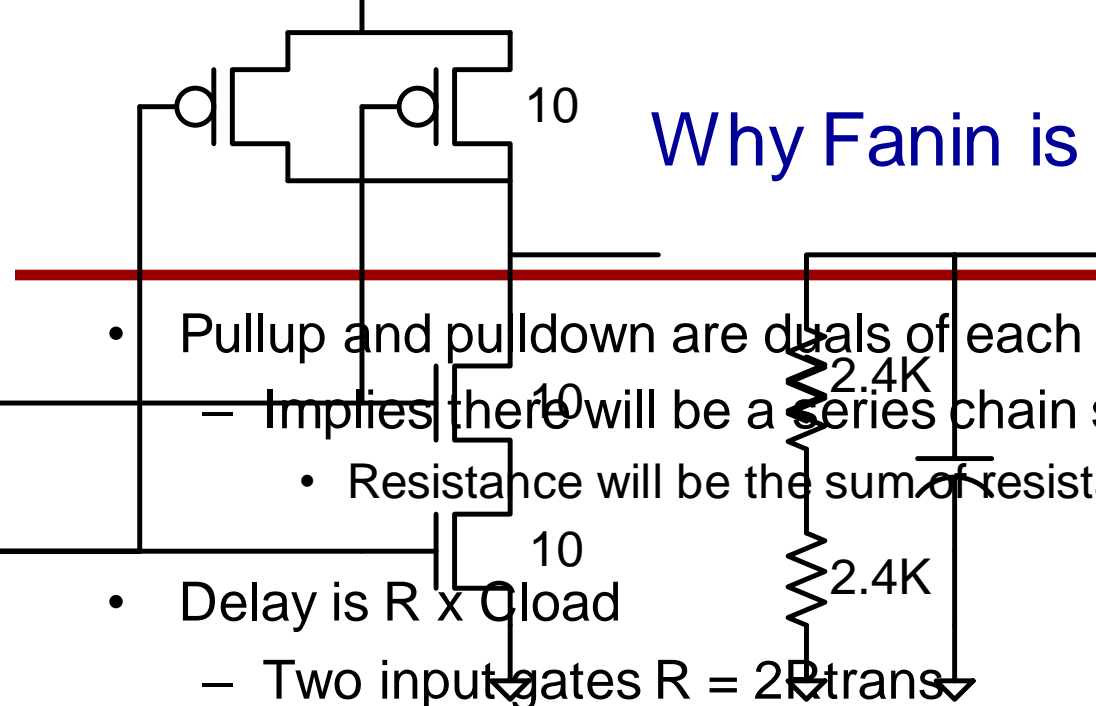
Pulldown: R for each transistor
Pullup: $R/3$ (not $R/6$) for each transistor



Pulldown: R for each transistor
Pullup: $2R$ for W transistor, R for each $2W$ transistor

- R corresponds to $R_{\text{trans},n}$ of an NMOS transistor with width W
 - Does it matter that one gate's resistance is R , and the other is $2R$?

Why Fanin is Bad

- 
- The image contains two circuit diagrams. The left diagram shows a stack of three inverters connected in series, with the number '10' next to each, representing fan-in. The right diagram shows a pullup network (resistor) and a pulldown network (transistor) connected to a node, with the number '10' next to the transistor, representing fan-in. The pullup network is a resistor labeled '2.4K' and the pulldown network is a transistor labeled '2.4K'.
- Pullup and pulldown are duals of each other
 - Implies there will be a series chain somewhere
 - Resistance will be the sum of resistances
 - Delay is $R \times C_{load}$
 - Two input gates $R = 2R_{trans}$
 - Three input gates $R = 3R_{trans}$
 - Often called stack effect
 - Can deal with resistance
 - Make transistors wider
 - But will still lead to slower gates
 - Limit stacks to around 3 transistors in series

Why Wires Are Important

- The capacitance of wires is much larger than min size transistor
- Tend to make transistors larger (W larger) to drive wires
 - Reduce R so RC product is smaller
 - But this increases the C_g of the transistor
 - This increases the power of the circuit
- If you estimate the wire length wrong
 - You will have large load where you did not expect
 - R C for this wire will then be large, and node will be slow
 - You have a small load where you did not expect
 - Wasting power by having transistors larger than needed
- When wires are too long, their resistance becomes a problem too

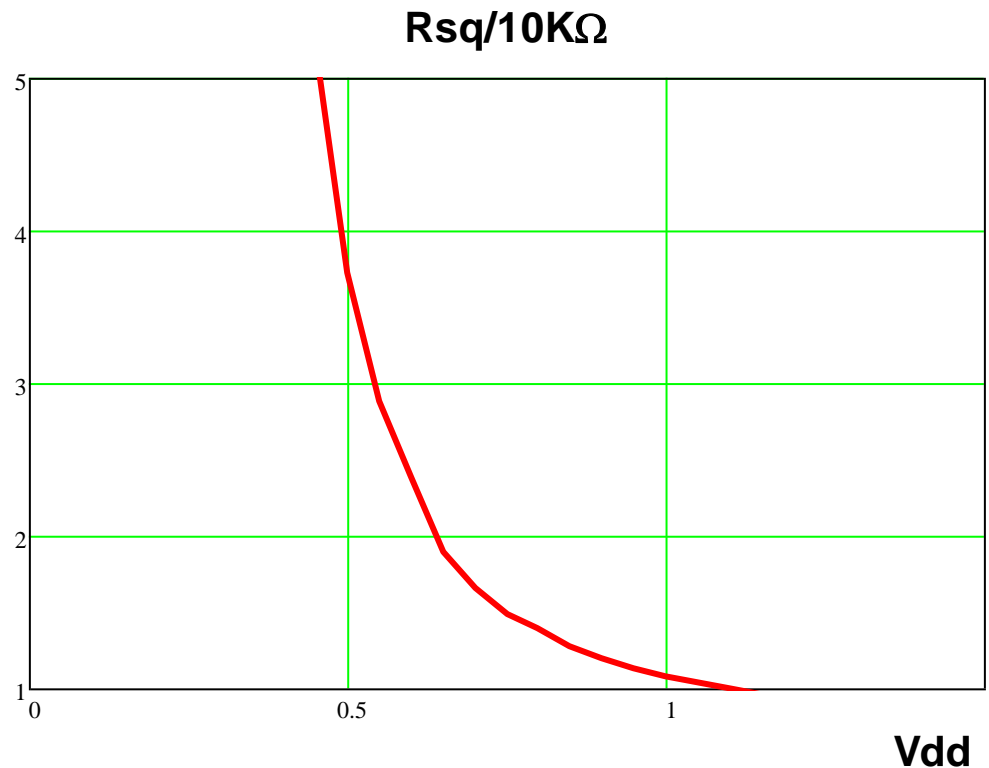
Gate Performance Summary

- Gate speed depends on two factors — resistance and cap
 - Resistance is set by size of driving transistors, and # in series.
 - Capacitance is set by the wiring and the fanout.
- Faster circuits
 - Low fanin and low fanout
 - Transistor capacitance larger than wire capacitance
 - So short wires and larger transistors are good
 - High V_{dd} and low V_{th} (low R_{trans})
- Notice that many of these are in conflict with low energy

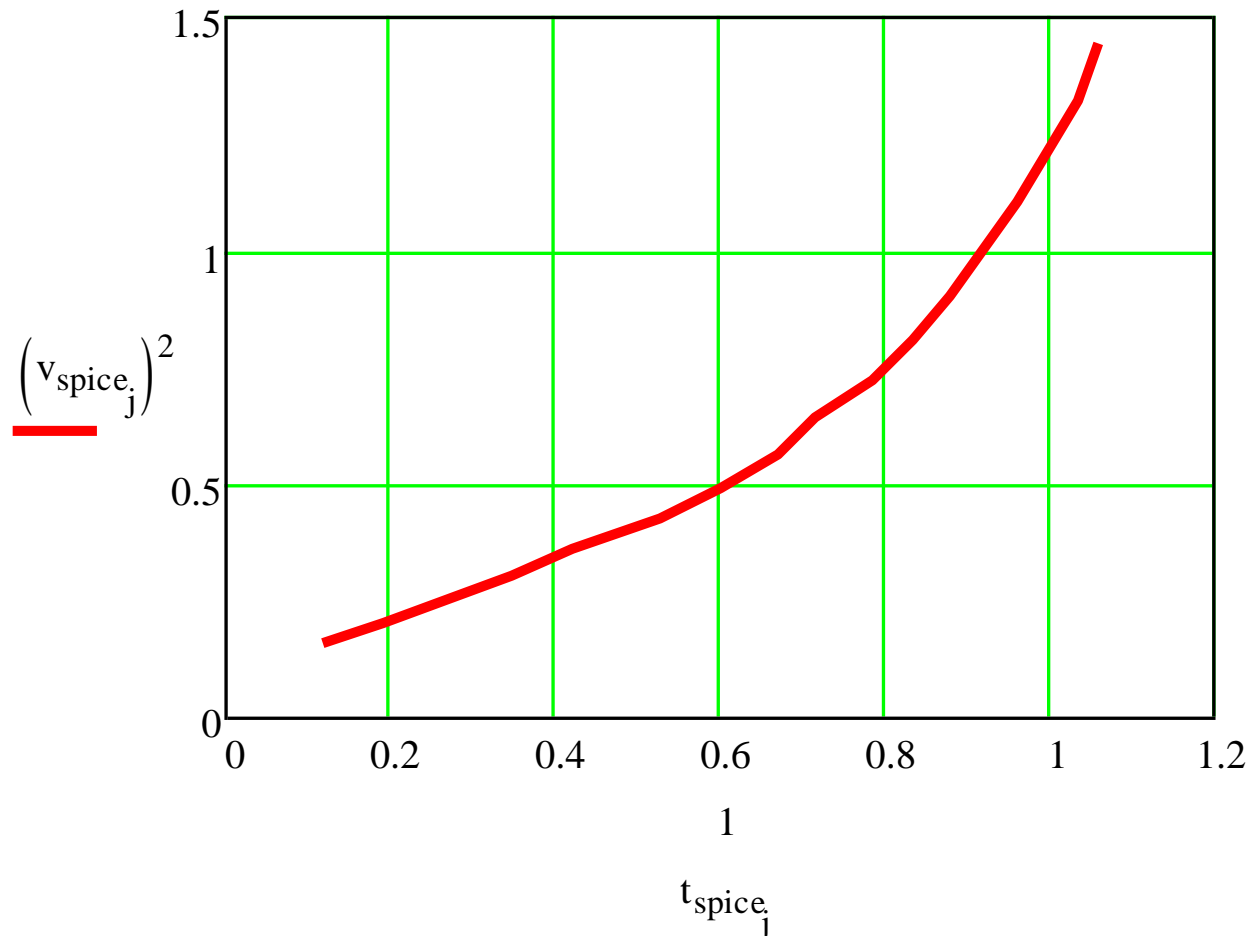
Notice Trade-off Between Speed / Power

- Low power wanted low Vdd, small devices

- Fast circuits want
 - High Vdd (low R)
 - Large devices
 - Compared to Cw



CMOS Speed Energy Trade-off (Changing Vdd)

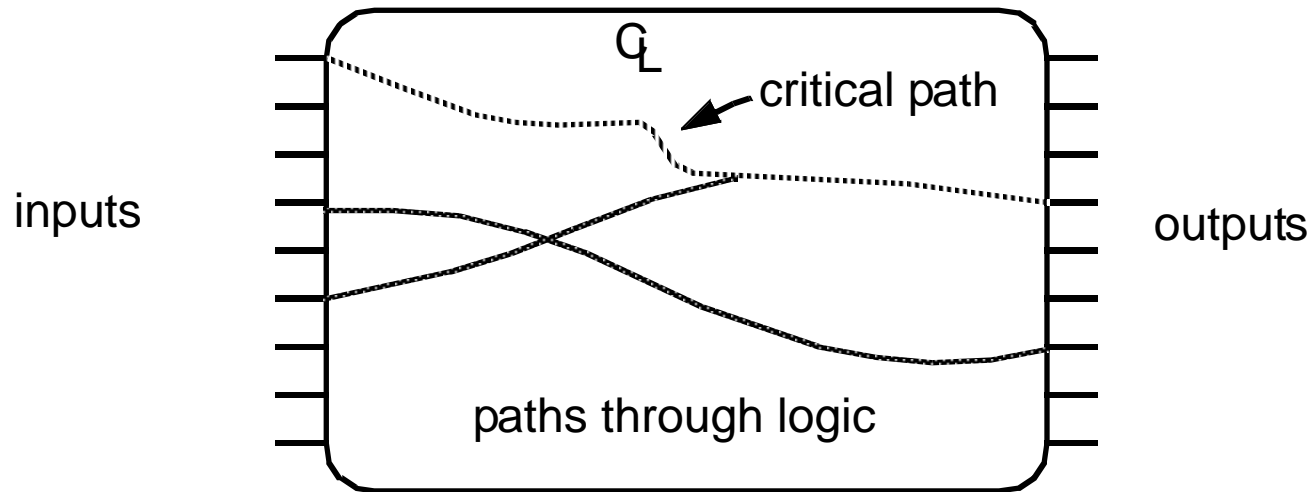


System Performance

- System performance is a little tricky to calculate
 - Even when all the gate delays are known
- Issue is that we are building a synchronous system
 - Runs at some clock frequency
 - Need to ensure that all flop inputs settle before next clock
- Clock then depends on the worse-case path through the logic
 - This is often called the critical path
 - Many tools, timing analysis tools can find these paths
- System performance harder to calculate than power

Critical Paths

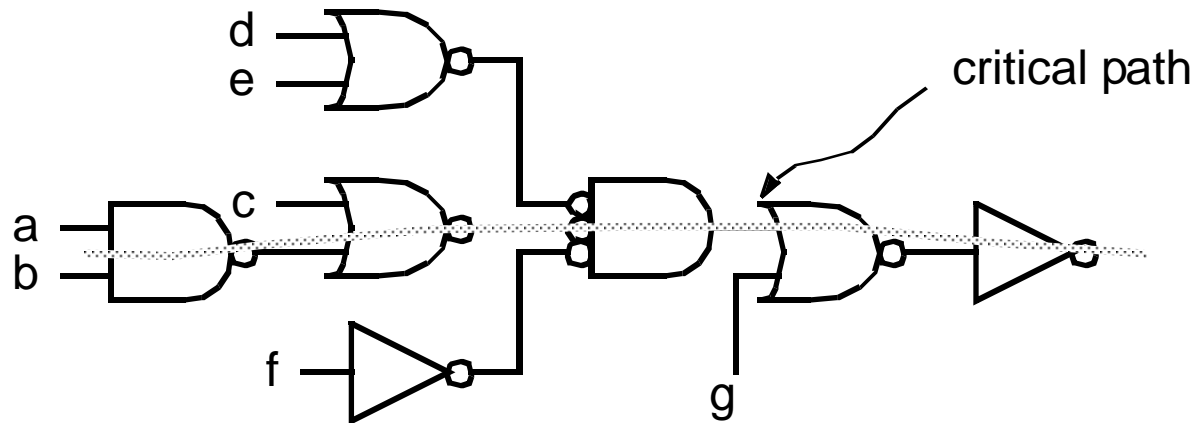
- There are many signal paths through combinational logic.



- Not all the paths have the same delay
 - Path from input to latest output is called the critical path
 - It is this path that will slow down machine, since clock has to wait for all the outputs to be valid.

Critical Paths

- Look at the function
 - $((a b)' + c) (d + e) f + g$



- If all the inputs change at the same time
 - Worst-case path will probably be from the inputs $\{a, b\}$
 - Speeding up g won't help much

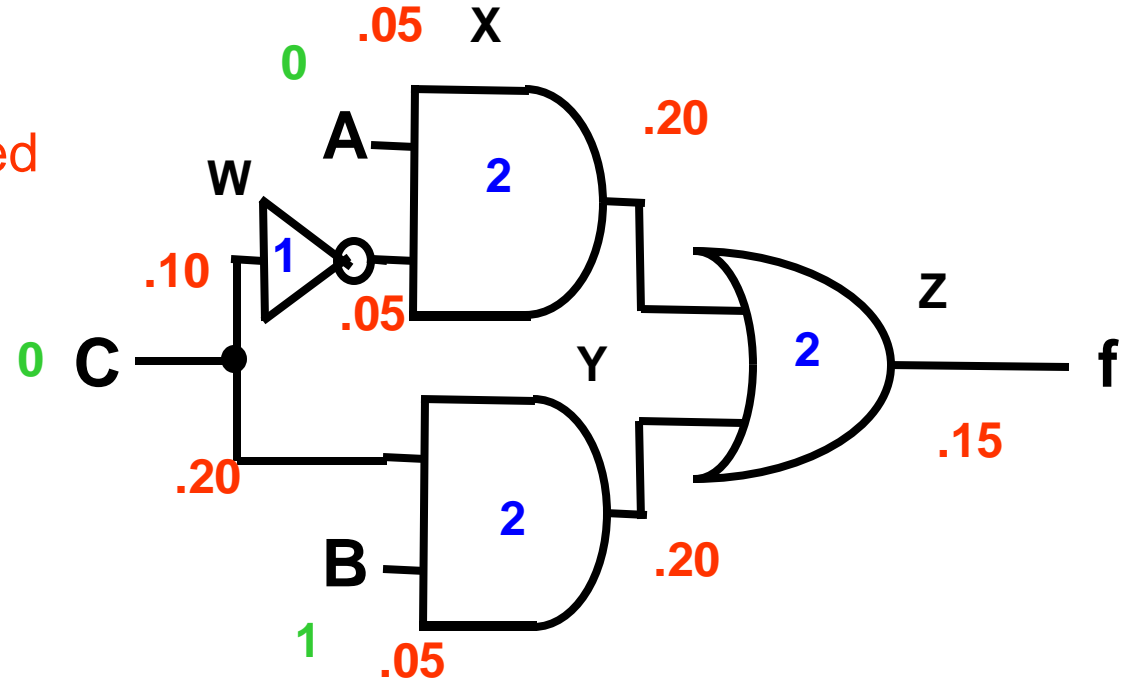
Static Timing Analysis

- Timing analysis
 - Goal 1: find worst possible delay in a circuit (over all possible input transitions)
 - e.g., worst circuit delay $\leq 1\text{ns}$
 - Goal 2: find nets and gates which contribute to worst possible delay
- Too many paths in a circuit (billions)
 - Avoid enumerating paths – Static Timing Analysis (STA)
 - Not as accurate as electrical circuit simulation
 - Much faster than electrical circuit simulation

Ack: Prof. Igor Markov, Univ. of Michigan

Timing Analysis Inputs

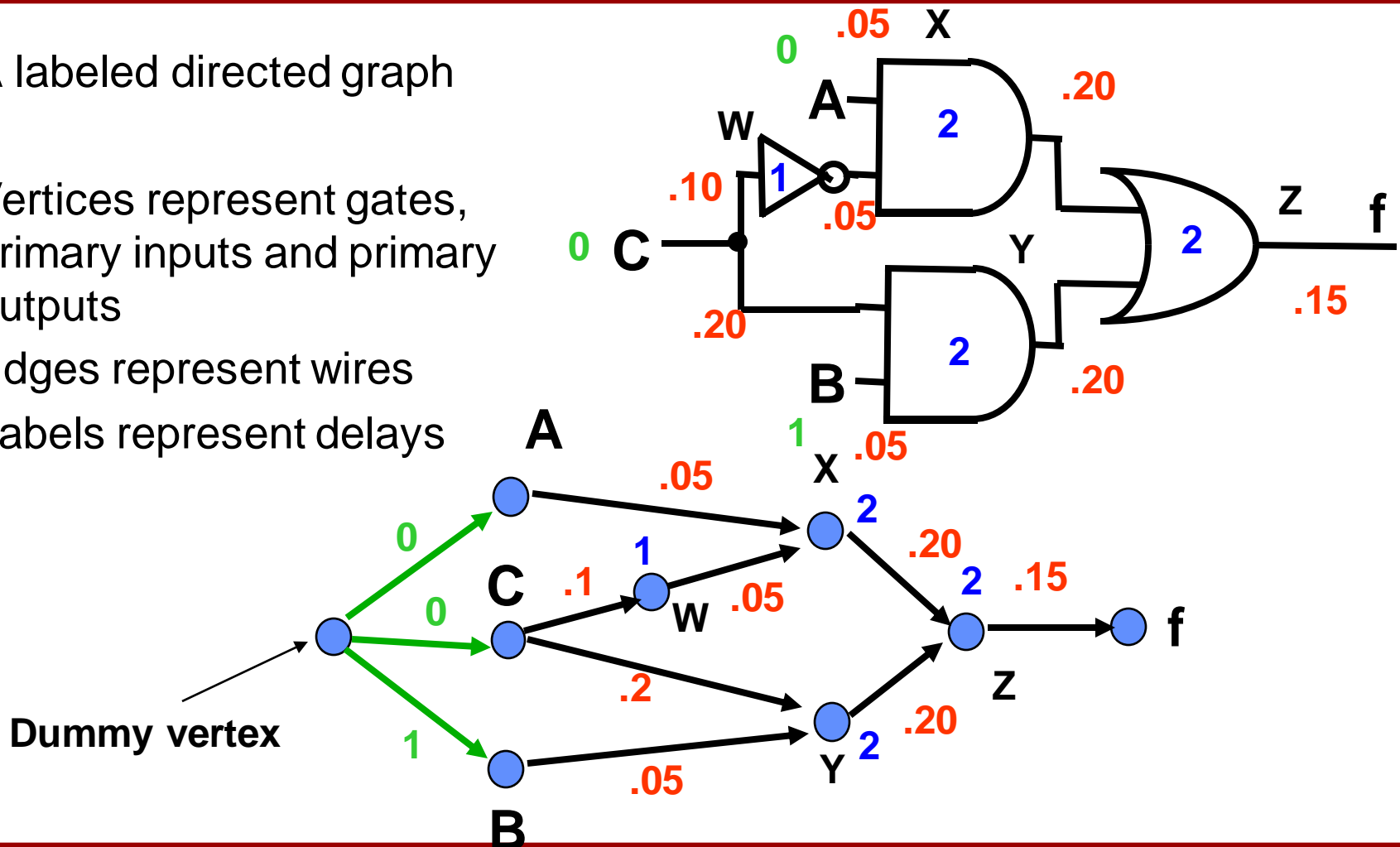
- Arrival time in green
- Interconnect delay in red
- Gate delay in blue



Ack: Prof. Igor Markov, Univ. of Michigan

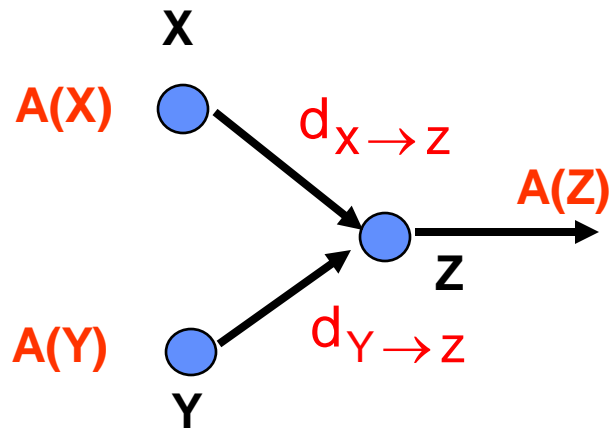
Timing Graph

- A labeled directed graph
- Vertices represent gates, primary inputs and primary outputs
- Edges represent wires
- Labels represent delays



Actual Arrival Time

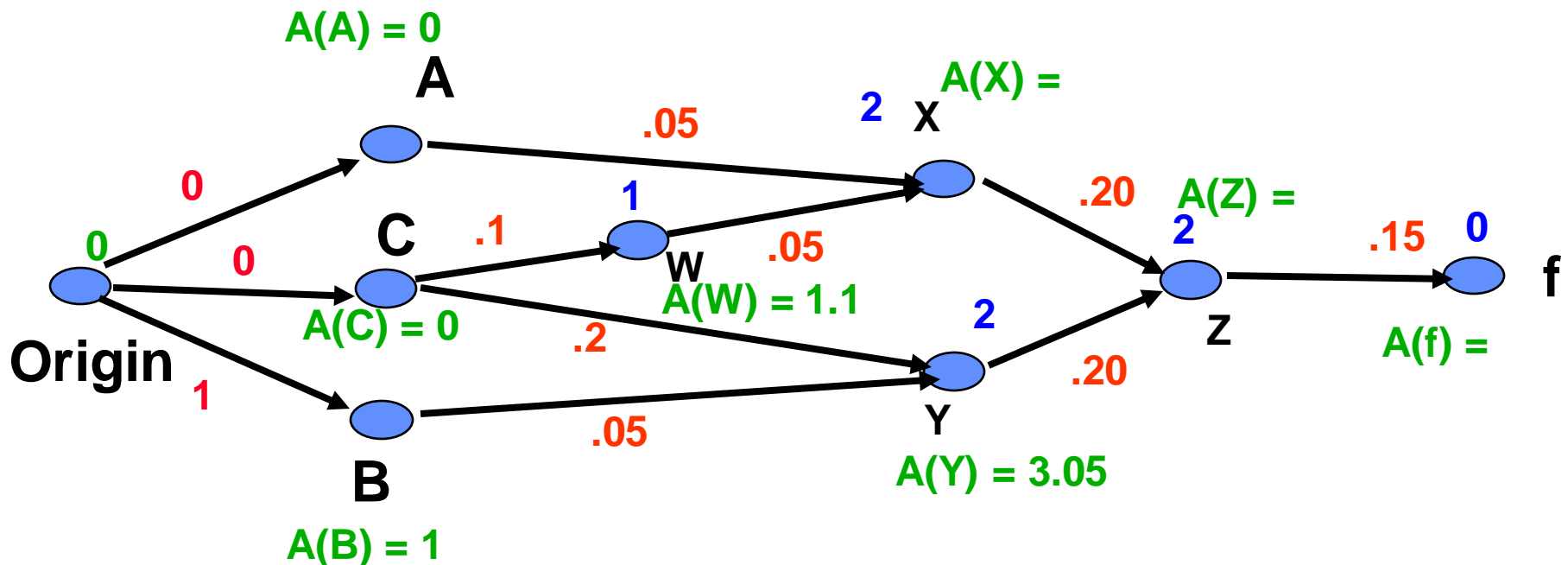
- Actual arrival time $A(v)$ for a vertex v is latest time that signal can arrive at v (actually the OUTPUT of v when v corresponds to a gate)
- Start from inputs (input arrival times given)



$$A(Z) = \text{Delay}(Z) + \text{MAX} (\{A(X) + d_{X \rightarrow Z}\}, \{A(Y) + d_{Y \rightarrow Z}\})$$

Ack: Prof. Igor Markov, Univ. of Michigan

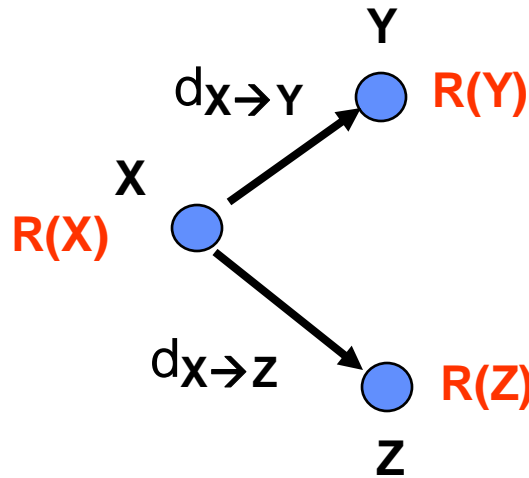
Actual Arrival Time : Example



Ack: Prof. Igor Markov, Univ. of Michigan

Required Time

- Required time of node v : $R(v)$
- Start from outputs (output required time should be given)

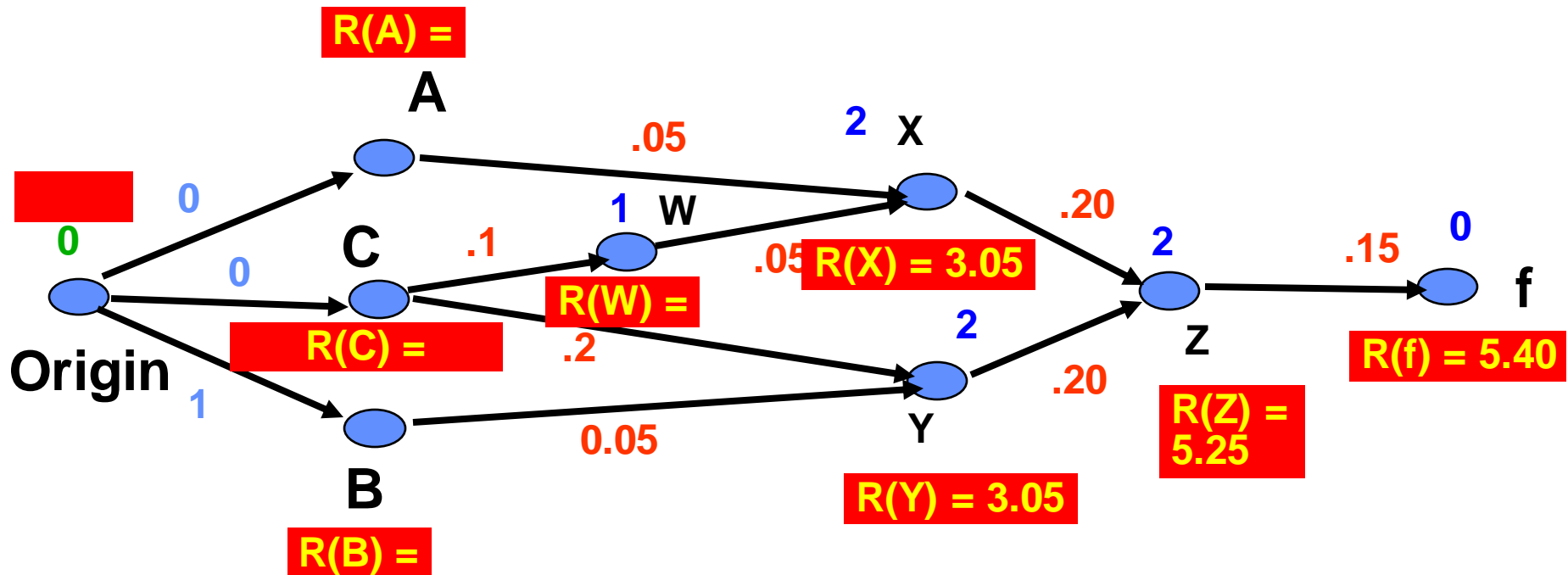


$$R(X) = \text{MIN} (\{R(Y) - \text{Delay}(Y) - d_{X \rightarrow Y}\}, \{R(Z) - \text{Delay}(Z) - d_{X \rightarrow Z}\})$$

Ack: Prof. Igor Markov, Univ. of Michigan

Required Time Propagation: Example

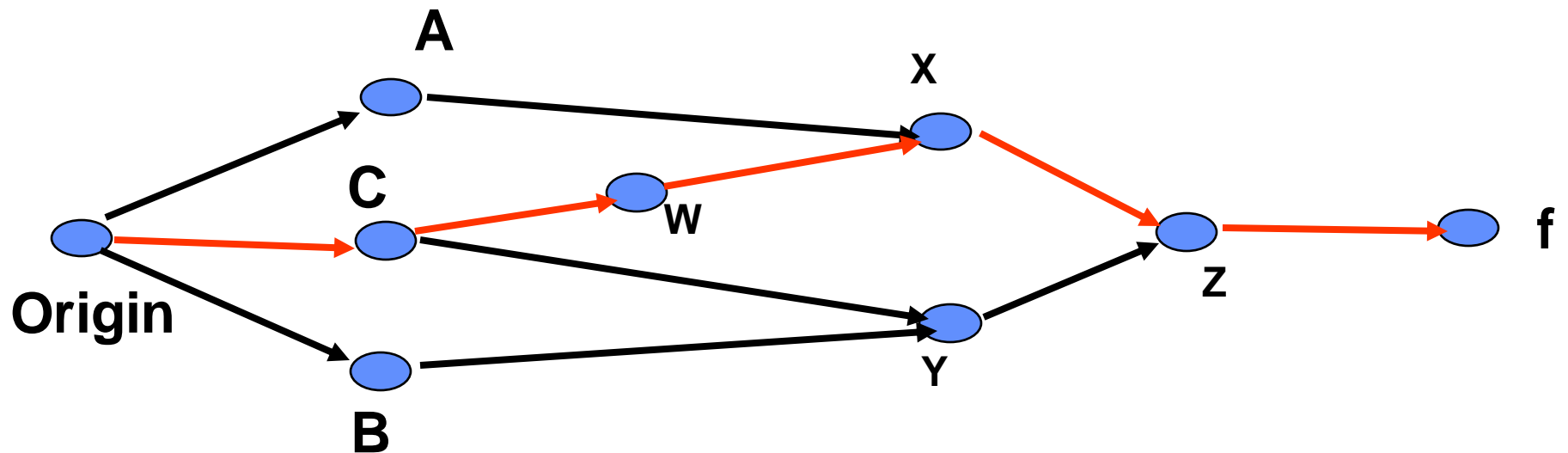
- Assume required time at output $R(f) = 5.40$
- Propagate required times backwards



Timing Slacks using STA

- At each vertex X compute $slack(X) = R(X) - A(X)$
 - Negative slack somewhere \Rightarrow timing violation
(i.e., some path that does not meet timing constraints)
 - No negative slack \Rightarrow timing constraints are satisfied
(all paths meet timing constraints)
- Didn't have to enumerate paths
- The number of possible paths can be exponential (e.g., in an array multipliers)
 - Path enumeration does not scale, but
 - STA takes linear time \rightarrow very fast

Timing Slack Computation



Ack: Prof. Igor Markov, Univ. of Michigan

Why Is Slack Computation Important?

- Suppose you have negative slacks on some paths
 - You know which paths to focus on to satisfy delay of the circuit
- Suppose you want to make a circuit faster
 - You know which paths to focus on (paths with very little slack)
- Suppose you have a circuit with paths with large slack values
 - Can potentially make the W's of the transistors on those paths smaller
 - Can reduce power
 - Can reduce area

Dealing with a Slow Input: Factoring

- Any logical function can be made fast for a single input
 - Factor the logic and determine two answers
 - Answer if input was 1, and answer if input was 0
 - Real output can be found in one mux delay
 - This input selects the correct output
 - This technique is used all the time in real circuits!
 - Remember it
- Formally to make $f()$ evaluate fast for x
 - $f(x,y,z) = [x \wedge f(1,y,z)] \vee [x' \wedge f(0,y,z)]$
 - Precompute $f(1,y,z)$ and $f(0,y,z)$
 - Any logical function can be converted to normal form by repeated factoring

Area Objective

- Size metric depends heavily on implementation technology
- In board level design
 - Memory is very cheap (one chip holds tons)
- In FPGAs, memories are more expensive (but getting cheaper)
- In VLSI ASICs
 - Memories are pretty dense, as they are regular structures
 - Random gates are dominated by wires
- Remember:
 - Chips sometimes have minimum allowable area. Smaller is not necessarily better. As long as you fit in a certain die size you may be OK!

Managing the Trade-offs

Need to manage conflicts between these resources

- Higher performance
 - Can mean a trickier design
 - More validation, more risk
 - Can mean bigger gates
 - More energy and area
- Custom cell design / layout
 - Reduce area, energy and improve performance
 - Increases design time, and circuit risk
- Reusing a design
 - Lower risk, lower design cost
 - Reduced functionality, area cost for functions not using

Design Time / Complexity

- This is the number one issue today
 - Reason a new custom design costs over \$100M to produce
- When your chip has a minimum of 100M devices on it
 - Tend to be very conservative, since every device must work
 - Even in the presence of fab variations, and noise
 - Move to bullet proof design styles
 - Static CMOS gates, previously validated (std cells)
- Move to higher level design input
 - Verilog / System Verilog
 - Some people are moving to High Level Synthesis (HLS)

Choosing Gates Is Not the Real Problem

- Turns out that tools will help us figure out the gate stuff

But then our problems move to higher levels

- How to create a good
 - 32 bit adder
 - Priority encoder
 - Multiplier
 - Floating point adder
 - Floating point multiplier
- Some of these tools can handle too
 - But there are always higher levels

You Still Need To Understand The Problem Even When Tools Do The Work

- Tools are very powerful
 - Much more effective in the “right” hands
- You need to understand the problem
 - To set and balance objectives
 - To get out of local minima that always occur
 - Work on the larger problem
- We will look at some of these issues in the next lecture