(*Lec 18*) Electrical Timing Issues: The Elmore Delay Model

- What you know…
  - Lots of synthesis for logic and for geometry
  - Ditto for verification--for logic
  - **Logical** timing abstraction: Static timing analysis, topological delay

- What you don't know…
  - How the geometric design of real, routed wires impacts delay
  - **Electrical** timing abstraction
  - We need to develop some usable notions of "delay" for use with layout algorithms: models simpler than a full simulation, but accurate enough

(Thanks to Larry Pileggi, for many cool slides & ideas here…)

Copyright Notice

Where Are We?

- For more accurate timing, need *electrical* wire delay estimation

	M	T	W	Th	F	
Aug	27	28	29	30	31	1
Sep	3	4	5	6	7	2
	10	11	12	13	14	3
	17	18	19	20	21	4
	24	25	26	27	28	5
Oct	1	2	3	4	5	6
	8	9	10	11	12	7
	15	16	17	18	19	8
	22	23	24	25	26	9
	29	30	31	1	2	10
Nov	5	6	7	8	9	11
	12	13	14	15	16	12
Thnxgive	19	20	21	22	23	13
	26	27	28	29	30	14
Dec	3	4	5	6	7	15
	10	11	12	13	14	16

Introduction
Advanced Boolean algebra
JAVA Review
Formal verification
2-Level logic synthesis
Multi-level logic synthesis
Technology mapping
Placement
Routing
Static timing analysis
Electrical timing analysis
Geometric data structs & apps

Nominal Deadlines…

Last 760 lecture (probably…

Thnxgive	19	20	21	22	23	13
	26	27	28	29	30	14
Dec	3	4	5	6	7	15
	10	11	12	13	14	16

HW5

6 PPT slide paper review

Proj 3 demos

- …and, this is clearly a bit extreme for the last week of class
  - Open to suggestions for moving *some* deadlines BACK *some*…
  - …but need to be careful not to mess up people with finals, early travel plans for break, etc

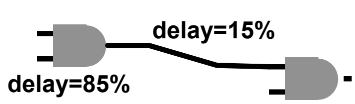
Timing Issues in Layout

- What's the problem?
  - Delays on signals due to wires no longer negligible
  - Modern designs must meet tight timing specifications
  - Layout tools must guarantee these timing specifications

- How have we addressed this so far in layout?
  - By ignoring it, mostly
  - Implicitly, qualitatively
    - We try to make layout area small
    - We try to make clusters close together
    - We try to make wires short
    - etc
    - All these are good things, but not the same as a *guarantee*...

Timing Issues: Impact of Interconnect

- IC technology trends



Mid 80s Scenario

Most of the input to output delay for 1 level of logic is due to gate delay

Wire delay is a very small component of the overall delay, ~18% here

Mid 90s Scenario

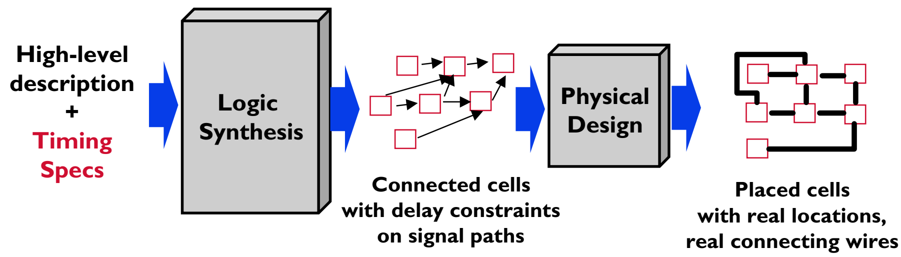
Half of the input to output delay for 1 level of logic is due to wire delay

Today's Scenario (example bad case)

Most of the input to output delay for 1 level of logic is due to wire delay

Timing Issues: Role of Layout Tools

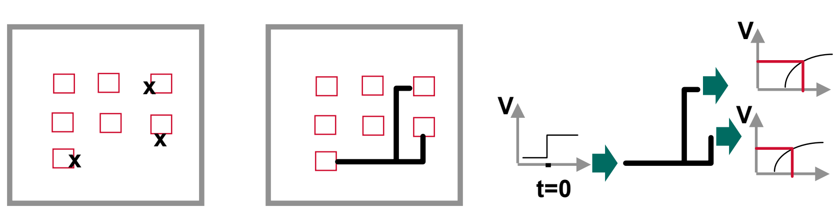
- Unfortunately, easy for layout tools to screw up the timing properties that "upstream" tools try to achieve



- Upstream tools
  - …may have no real, physical models for the placement or routing
  - Only have rough estimators to generate constraints on layout

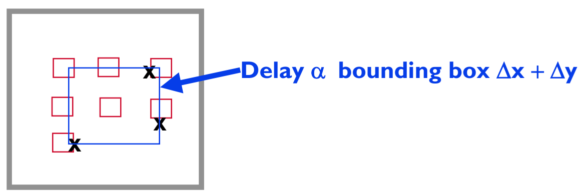
Basic Delay Modeling

- Let's focus in some detail on one important aspect of this overall timing optimization problem
- *Interconnect delay*
  - You do a placement, it puts the pins at a certain distance apart
  - So, you have to route a wire, it has an input-to-output delay
  - *Where* does the delay come from?
  - How accurately can we *predict* this delay?
  - How efficiently can we *model* this delay for use in layout tool?



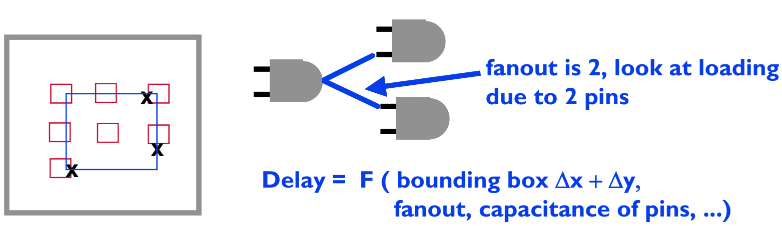
Sources of Delay: Model 1

- Delay = finite speed signal propagation through physical wires
- Model == *Length*
  - **Delay proportional to length**
  - **Shorter = better**
- Analysis
  - **Pro: This is really easy, qualitatively OK**
  - **Con: Not quantitatively accurate, extremely crude**



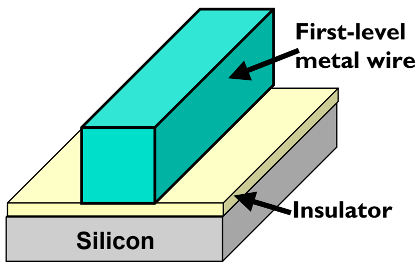
Sources of Delay: Model 2

- Add: Delay also affected by *circuit drive* limitations
- Model == *"Wire load"*
  - **Delay proportional to length, fanout, capacitance of the driven pins**
  - **Actually called "wire load models", usually model capacitance on a net**
- Analysis
  - **Pro: Qualitatively better**
  - **Con: Still focuses mostly on the pins, not on the wire; can be off by 3-5X**



Sources of Delay: Model 3

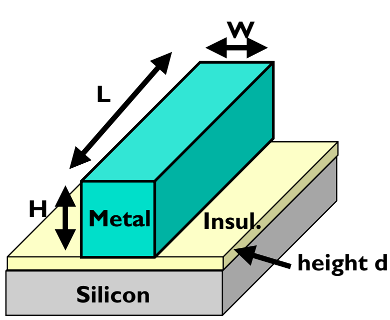
- Add: Delay comes from *parasitic loading* of the interconnect
  Depends critically on exact shape of the wired net
- Model == *Lumped Electrical Parameter*
  - Interconnect must be modeled as a circuit, analyzed as a circuit
- Why?



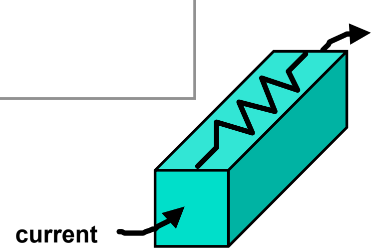
Interconnect geometry is now large relative to the devices themselves

Interconnect Models: RC Trees

- Let's see how to derive the most popular model used in layout applications for interconnect delay
- First: Interconnect -> Circuit

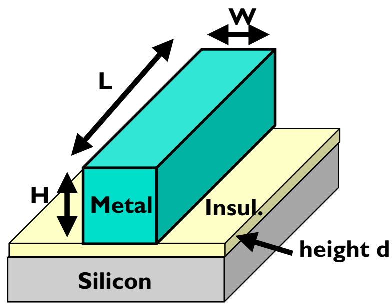


Metal wire has resistance = R to current flowing down its length

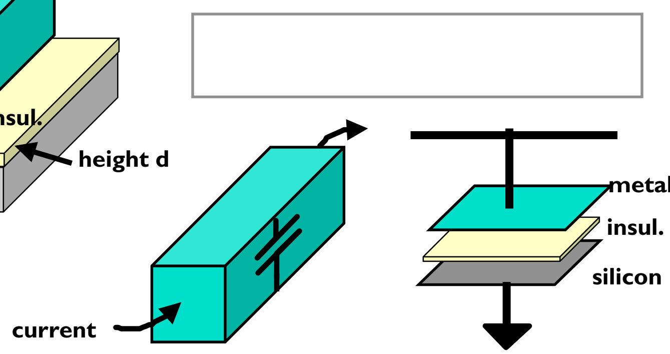


Toward RC Trees

- Interconnect -> Circuit

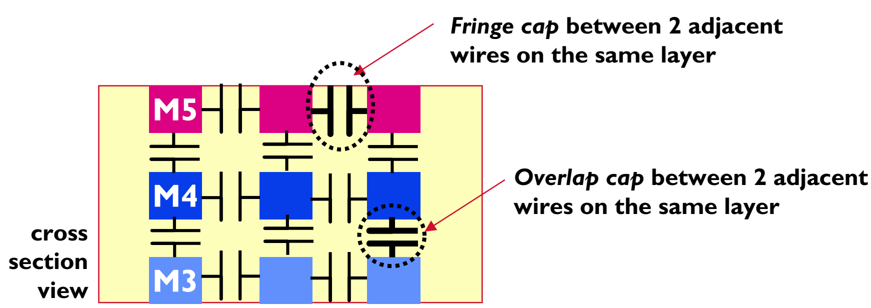


Metal wire has capacitance to silicon substrate, with insulator between



Aside: Metal Layer Capacitance

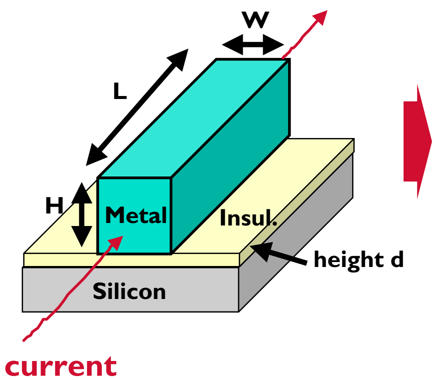
- Note: this view is *way* simplistic
  - You really get capacitance between any pair of conducting surfaces
  - So, in a multi-layer metal process you get Caps between all the layers
  - Vertically adjacent conductors create Overlap Cap.
  - Laterally adjacent conductors (wires *next* to you) create Fringe Cap.

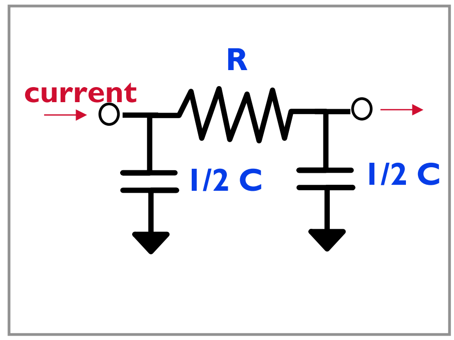


- We *won't* worry about all these different caps, just a single overlap cap

RC Trees

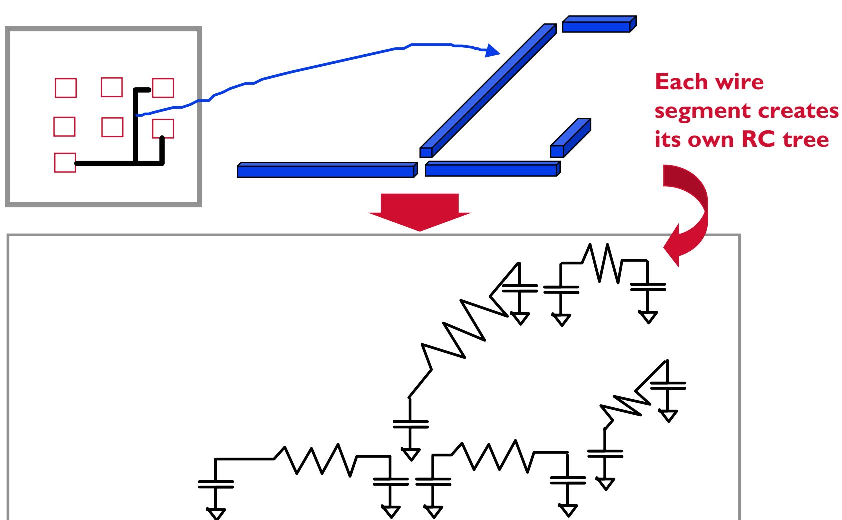
- Typical circuit model: Π **model ("pi" model)**
  - **Accounts for the resistance R and the capacitance C of wire segment**
  - **Symmetric (which is why we split the capacitance)**
  - **Small model, only need 2 numbers**





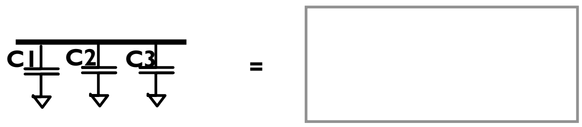
RC Trees

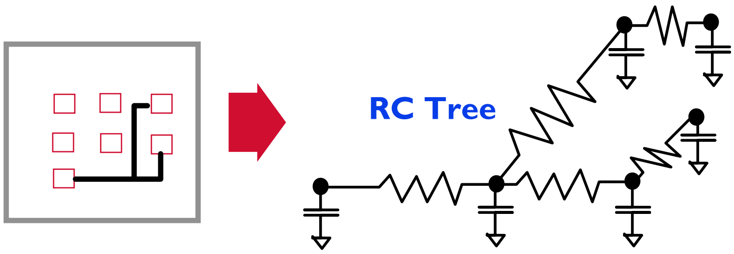
- Of course, that's just 1 segment of wire...



RC Trees

- Recall a simple rule from basic circuits (or physics)
  - Parallel capacitors can be replaced by 1 cap with Σ C

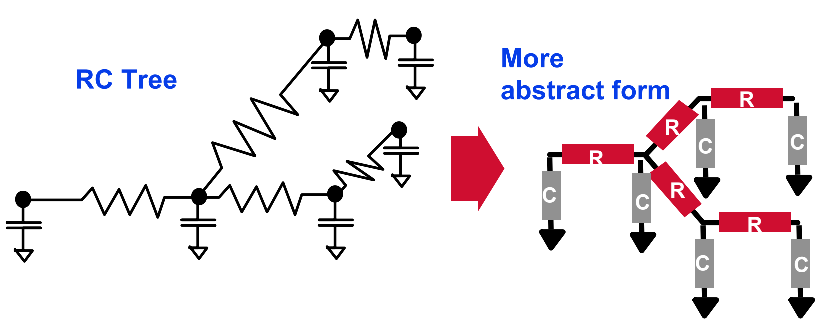




Note: each of the Rs, Cs in this tree are probably *different* numbers, since each depends on geometry of the segment

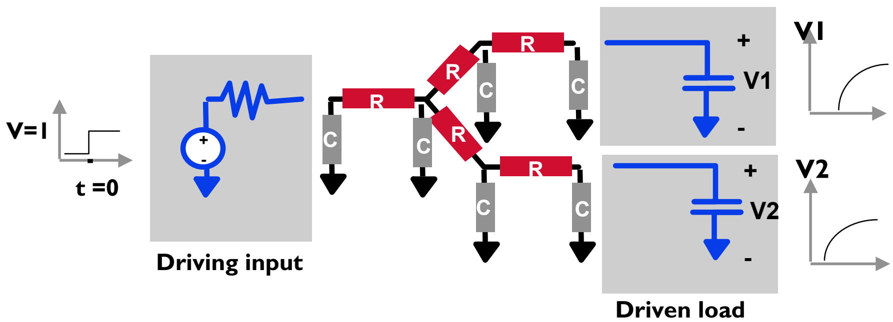
RC Trees

- RC Tree general form
  - A tree of resistors (no loops)
  - Root of tree is where signal is input
  - Leaves of tree are the driven outputs
  - Capacitors to ground at all intermediate nodes of the tree

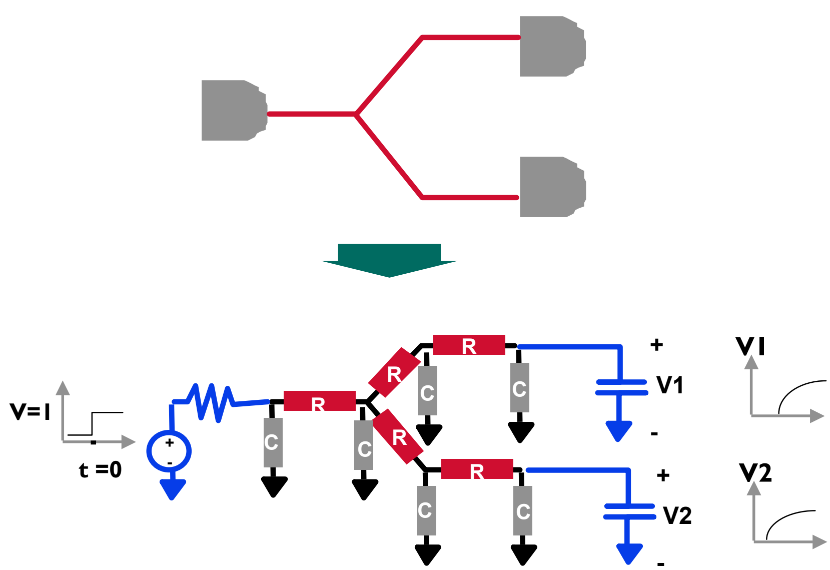


RC Trees: Delay Estimation

- OK, we can build them. What are they good for?
  - Turns out one can do fast, approx. delay estimation for an RC tree
  - Scenario
    - Voltage source + resistor as input at root (this models driving gate)
    - Capacitor as load at each leaf (each models a driven gate)



Summary: Gates + Wires -> RC Tree Circuits



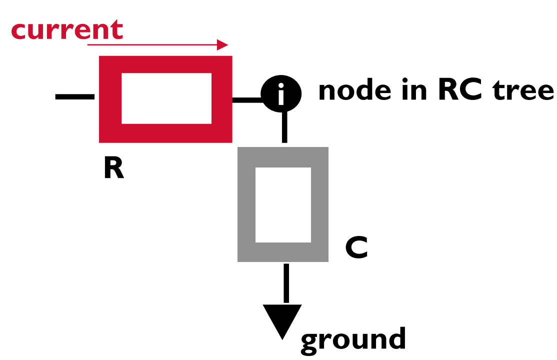
RC Trees: The Elmore Delay

- Famous delay formula called the "Elmore" delay
  - **Derived originally in the 40s for circuits applications**
  - **Resurrected in 80s by Penfield, Rubenstein, Horowitz for RC trees**
  - **Usually presented as a "magic formula" over the Rs and Cs...**

- Our goal
  - **Give the basic delay result, and explain how it's calculated and used**
  - **Apply the formula to a few illustrative examples**
  - **[Aside: Show how to derive the basic result--briefly-- since it's the most useful formula in the performance-based layout business (appendix)]**

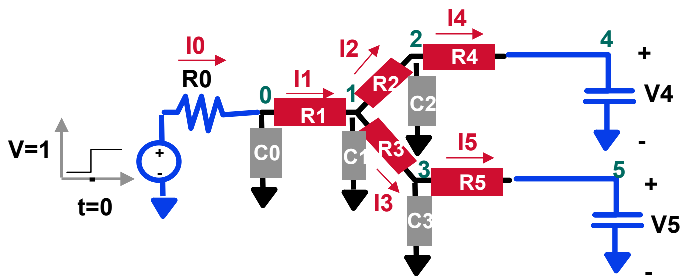
RC Trees: Labeling Convention

- Observe
  - **We combine ("lump") load capacitance with 1/2C from last segment**
  - **In RC tree, each R and each C may be *different***
  - **Give each a name: Ri feeds into node i, Ci hangs off node i**
  - **Label currents thru Ri as Ii**



RC Trees

- So, let's label our little example this way…
  - First the nodes (numbered 0 - 5)
  - Then all the currents thru the resistors (I0 - I5)

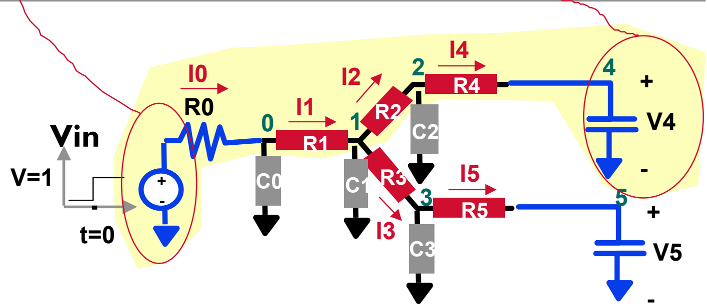


RC Trees: Elmore Delay

- What do we really want to get?
  - Approximate output waveforms, V4(t), V5(t), as efficiently as possible
- What do we know how to do? Can write Kirchhoff eqns here…

Example: KVL around the loop from Vin to V4 to gnd

Vin - R0•I0 - R1•I1 - R2•I2 - R4•I4 - V4 = 0



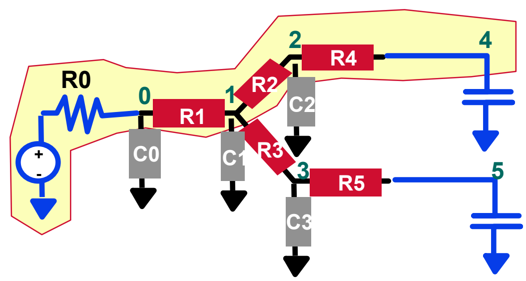
RC Trees: Elmore Delay

- Common *patterns* of resistor values in all these eqns

Vin - R0•I0 - R1•I1 - R2•I2 - R4•I4 - V4 = 0

- Can define some notation: R0k(i)
  - **R0k(i) is the sum of resistors you see walking back up the tree from node "k" to the root, that are *ALSO* on the path from root to node i**
  - **Called "upstream resistance" for node "k"**

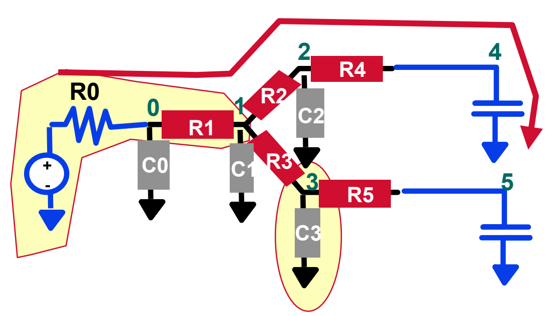
R04(4) = (R0 + R1 + R2 + R4)



RC Trees: Elmore Delay

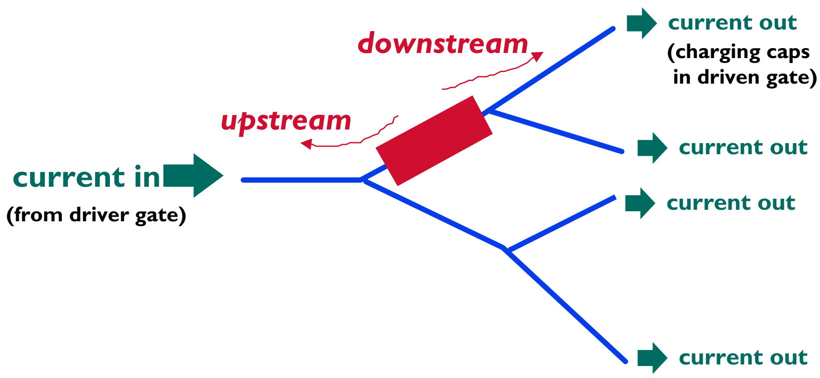
- More complex example of R0k(i)
  - **Only R0 and R1 are on both paths: from root->4, and from root->3**
  - **Turns out the derivation focuses on paths the charging currents take from driver (root) to the individual leaf nodes (load caps)**

R04(3) = (R0 + R1)



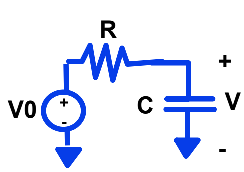
Aside: Stream Analogy

- Think of current like real water, flowing in tree
  - From any component of tree, if you look at what is happening back up toward the root, it's UPSTREAM
  - Look toward leaves, its DOWNSTREAM



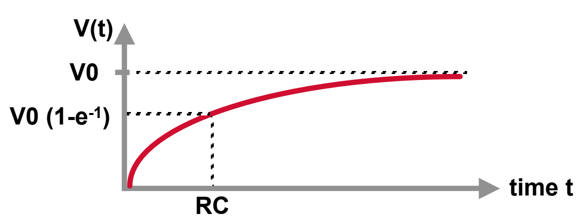
What Does Elmore Delay Try to Model?

- Recall: Apply a voltage step to a circuit with a capacitor…
  - Current starts to flow…eventually cap charges up, current stops flowing
  - Cap charges up to V0 here
  - Elmore tries to model output voltages with a single-time-constant exponential ramp voltage; trick is estimate a good "RC" for accuracy



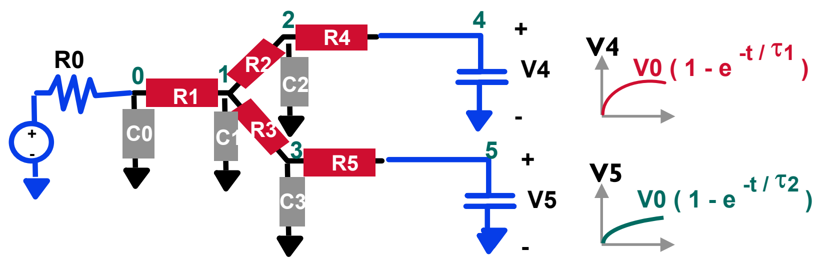
KVL: $V0 - R \cdot C \cdot dV/dt - V = 0$

Solve diff. eq: $V(t) = V0\,(1 - e^{-t/\boxed{RC}})$



What Does Elmore Delay Try to Model?

- We want an accurate time constant "τ" for each output
  - Can depend only on the Rs, Cs we know from the RC tree
  - *Different* for each output--a unique feature for Elmore model



RC Trees: The Elmore Delay

- This is the magic formula that we can derive

$$Vi(t) = V0(1 - e^{-t/\tau}) \qquad \tau = \sum_{\substack{\text{Nodes } k \\ \text{in RC tree}}} R0k \cdot Ck$$

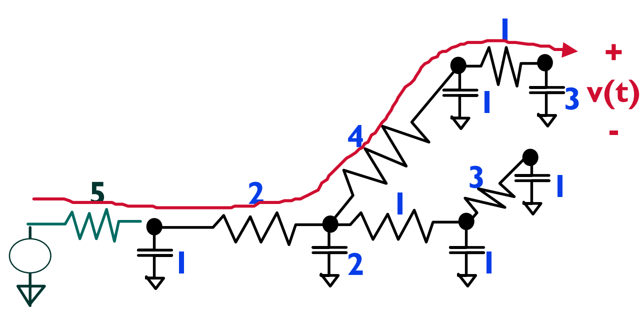
- τ is "the Elmore Delay"; recall:
  - We asked this: *what does this RC tree leaf voltage Vi(t) look like?*
  - We assumed this: *apply V0 step at t=0*
  - We also assumed: *can model voltage Vi(t) as 1 time constant,* $1 - e^{-t/\tau}$
  - Can derive this: $\tau = \Sigma_k$ R0k•Ck
- Note
  - A general formula for the time constant for the response at any leaf
  - Assume one time constant τ is a good approx for the actual delay

Observations

- Note
  - Basically says we can model the output at 1 leaf of an RC tree with an "equivalent circuit" that looks like 1 equivalent R, 1 eqv. C
  - We don't really know the R or the C though, just that RC = τ
  - Called a "one time constant" model (makes sense, eh?)
- Analysis
  - PRO: *Easy* to compute (can do it recursively by walking tree)
  - PRO: Gives you a *unique* delay for *each* output of the tree
  - PRO: Accounts for *all* the parasitics Rs, Cs of the interconnect
  - CON: It's still only a one time constant model; sometimes need > 1

Trick to Compute Elmore Delay Fast

- Do this:
  - Set $\tau = 0$; start walking down tree to the leaf node (arrow)
  - At each resistor, do τ += R • Σ (all caps downstream)



Delay =?

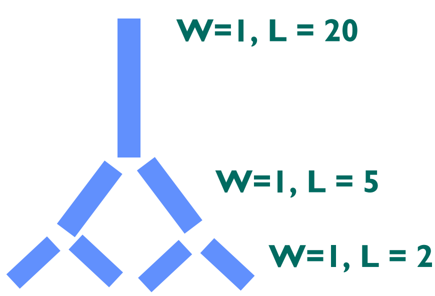
$$\text{delay(root->leaf)} = \sum_{\substack{\text{nodes i}\\ \text{from root}\\ \text{to leaf}}} \text{Ri} \cdot (\Sigma \text{ downstream caps})$$

Now What?

- The Elmore delay formulas are *immensely* useful
  - SImple enough for layout folks to use them in algorithms
  - Accurate enough that they beat simple length-based schemes
  - (Unfortunately, not so accurate that you can avoid later verification with what are called "higher order" models that incorporate more than one time constant)
- Applications
  - Let's look at a simple example and see how layout decisions affect actual delay, as measured with Elmore

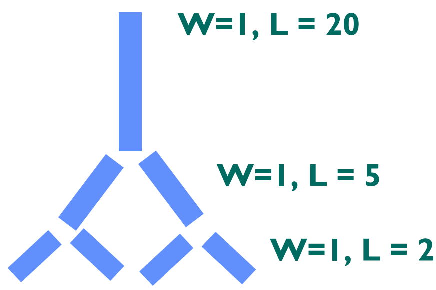
Elmore Example

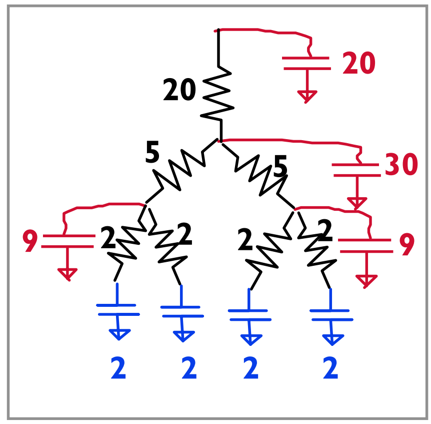
- Simple tree with 4 leaf nodes
  - Normalized parameters: r = 1 , c = 2
  - Just assume that for a segment, total R = r • L / W, C = c • W • L



Elmore Example

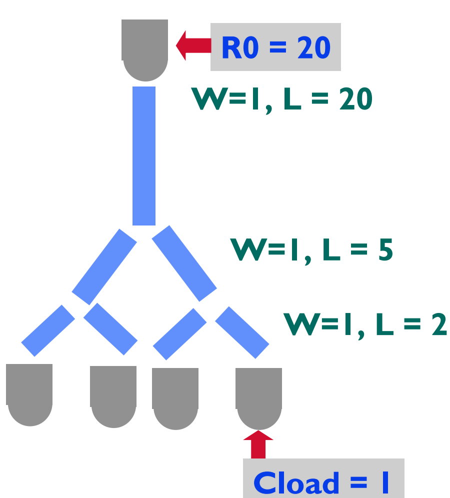
- RC Tree for the interconnect alone
  - Remember to add up caps each hanging off same node of ckt

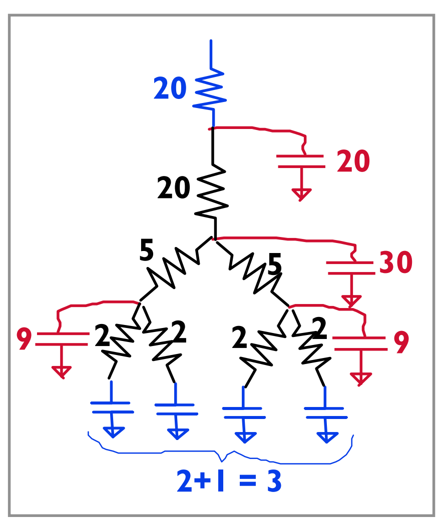




Elmore Example

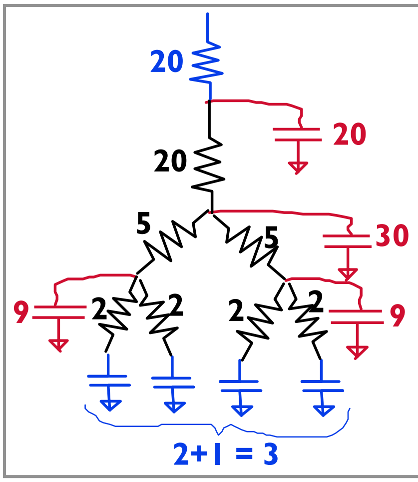
- Add driver and driven gates





Elmore Example

- OK: what's the delay to each leaf ?
  - Since symmetric, only need to compute 1 path



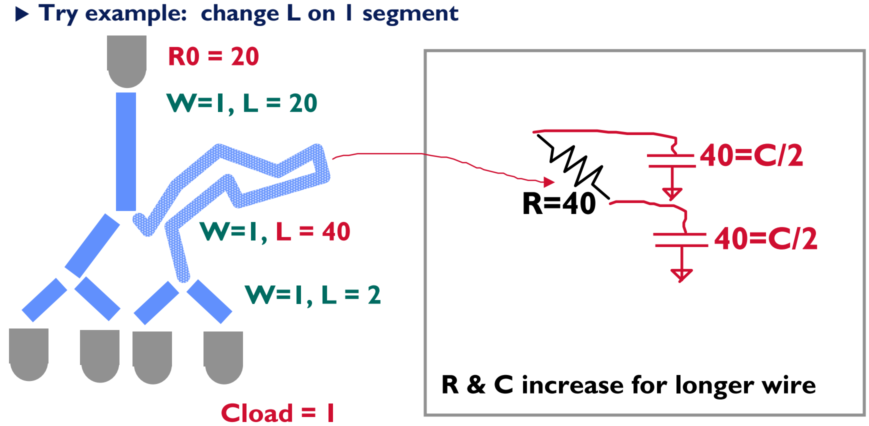
Remember the trick:

1. Set τ = 0, walk from root to leaf

2. At each resistor, do
τ += R • Σ (all caps downstream)

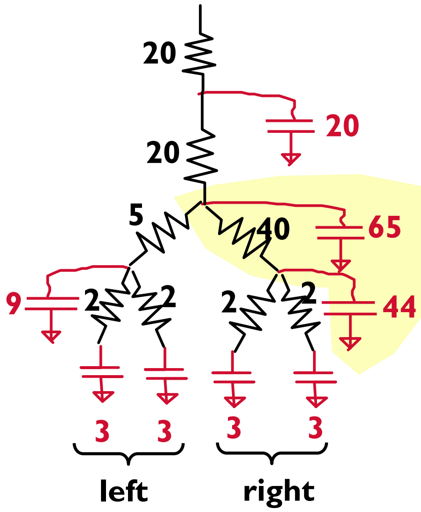
New Elmore Example

- What can *layout* (ie, placement, routing) do to wiring?
  - Change the length of a wire
  - Change the width of a wire (a very recent degree of freedom to use...)
  - Try example: change L on 1 segment



New Elmore Example

- OK, now what is delay to each leaf?



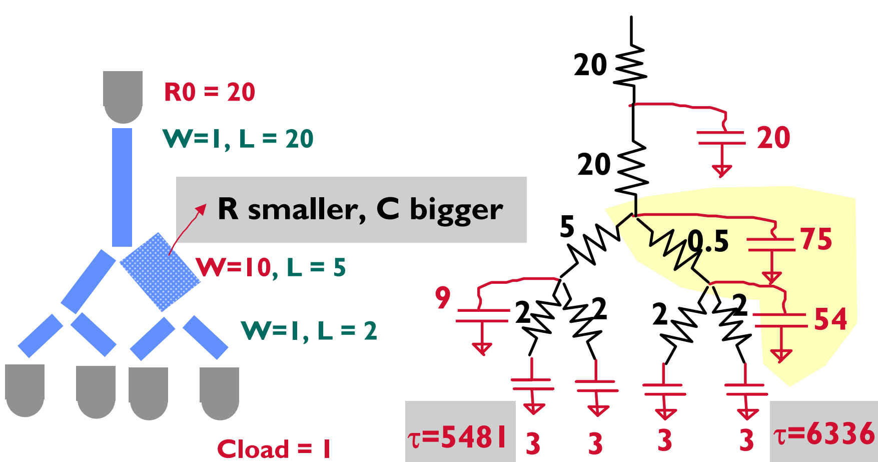
Right side:
τ=7606

Left side:
τ=5681

Note:
Extra C of longer wire even loads the left side of tree, upping the delay

New Elmore Example, version 2

- How about instead we change W=width on 1 segment?



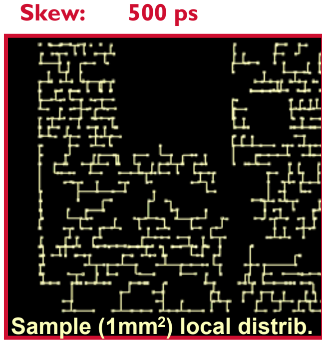
Elmore Applications

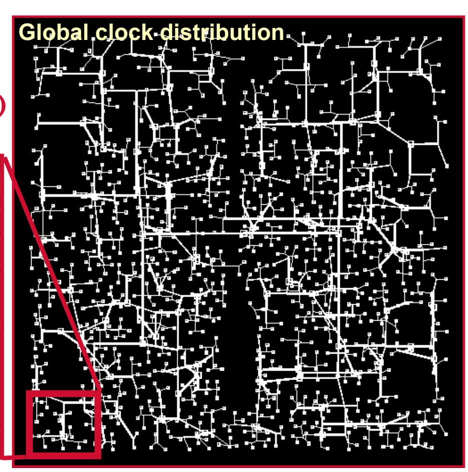
- Do people really use this delay metric?
  - Yes!
- Verification
  - It’s easy to compute, gives a semi-real delay to each leaf node in an RC tree, allows us to see how wire “shape” affects per-leaf delay
  - So, can use it for verification
- Synthesis (of layout)
  - Since it is easy to see how length change of width change affect per-leaf delay, this becomes an optimizable “degree of freedom” in some apps
  - Good example: clock trees

Clock Trees: ~Same Delay To Each Leaf

- Clock is huge global net (1000s of leaf nodes)
  - Each leaf is a latch, want ~same delay from root->latch; max(arrival time difference at latches) is called “skew”, want this small

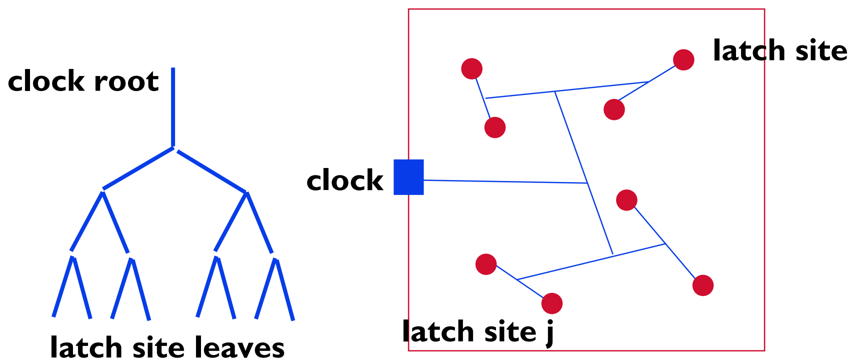
Source:	IBM
Size:	16,818 latches
Tech:	0.35 um
Freq:	200 MHz (T=5 ns)
Skew:	500 ps





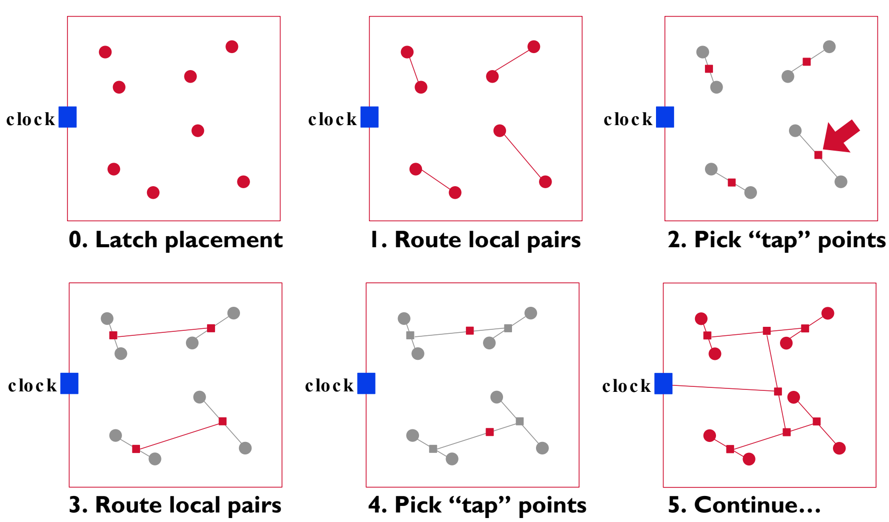
Clock Tree Routing

- It's a very specialized kind of routing, to optimize skew
  - Basically a recursive process, which tries to match delays to each subtree of the clock



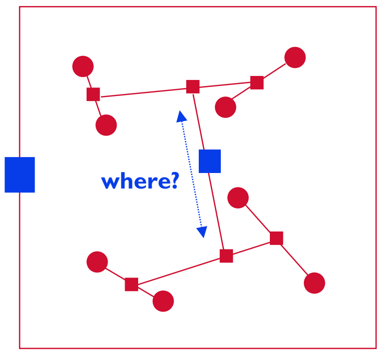
Clock Tree Routing

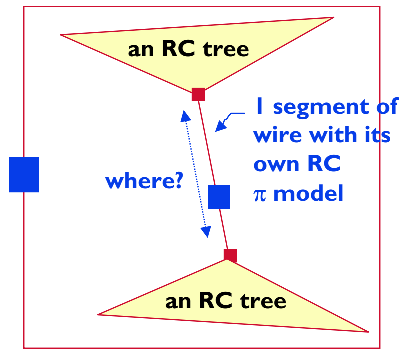
- Example: bottom-up construction



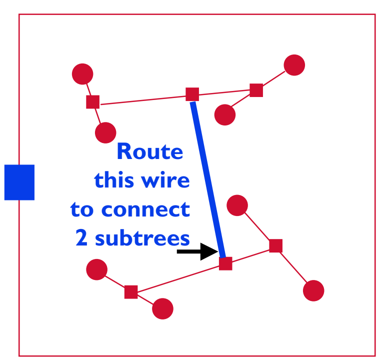
Delay Optimization Problem

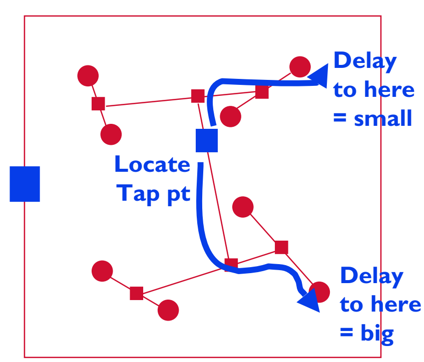
- **Proper location of "tap" points to balance delay to sub-trees**
  - **You have 2 routed clock "subtrees". You want to connect them, so you route a wire between them.**
  - **But, where do you put the connection--the "tap" point--on this wire, so that delay down each each subtree is matched?**





Example: Bad Tap Point Location

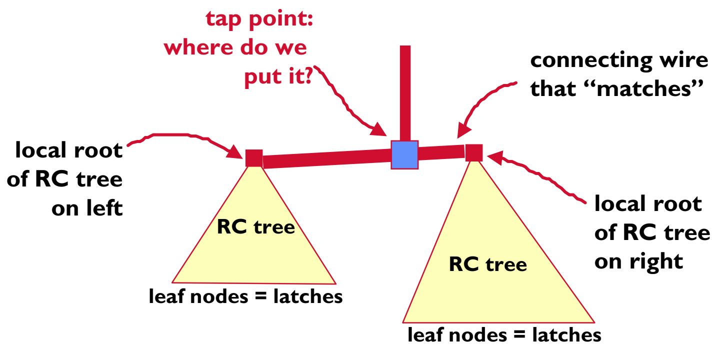




A bad tap point location gives unequal delays down each side of the clock, into each subtree

This is a Geometric/Delay Optimization Task

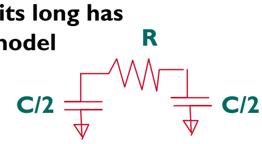
- Let us redraw for clarity
  - You already have 2 complete RC trees going down to latches
  - You have decided to "match" the local "roots" of these 2 trees
  - You will connect with a straight wire (you hope)
  - Problem: Where to put the tap point to equalize the Elmore delay on each side?



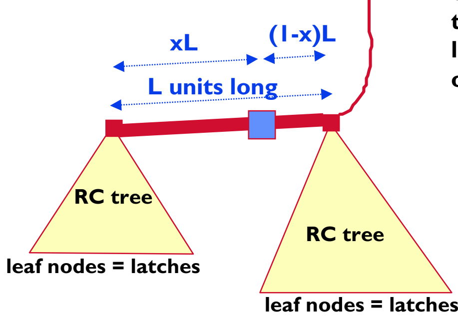
Nice Solution: Exact Zero Skew Algorithm

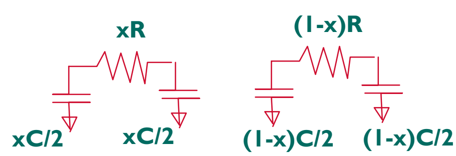
- Look closely at an RC model of this situation

Wire L units long has an R C π model



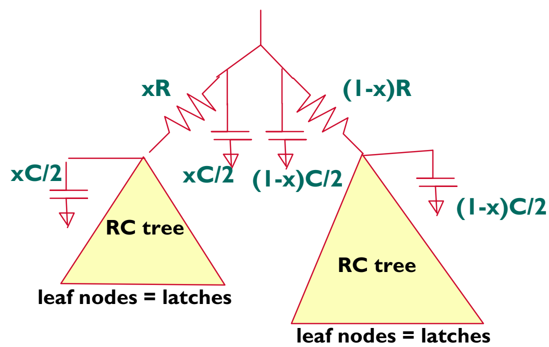
Since both R, C directly proportional to length L, it's easy to model the left segment of len xL, and right segment of len (1-x) L also as 2 π models





Exact Zero Skew

- So what have we got?
  - Complete RC model for the 2 subtrees, and the connecting (match) wire
  - In terms of a variable x that we don't know, that tells us where to tap
  - Goal: Elmore delay down to left latch sites == Elmore delay to right



Elmore Hacking

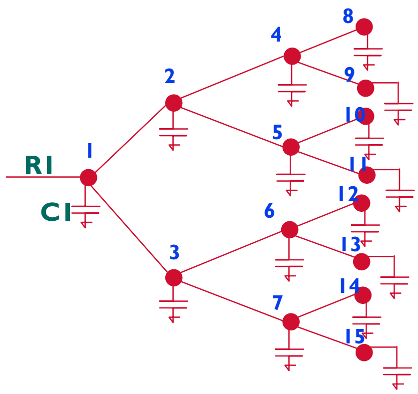
- Recall
  - Delay (RC) from root to leaf in an RC tree was calculated like this:

$$\text{delay(root->leaf)} = \sum_{\substack{\text{nodes } i\\ \text{from root}\\ \text{to leaf}}} Ri \cdot (\Sigma \text{ downstream capacitance} = Cdi)$$

- Can also define delay from root to an internal node j
  - Delay (RC) from root to internal node j is similar:

$$\text{delay(root -> j)} = \sum_{\substack{\text{nodes } i\\ \text{from root}\\ \text{to } j}} Ri \cdot (\Sigma \text{ downstream capacitance} = Cdi)$$

Elmore Hacking

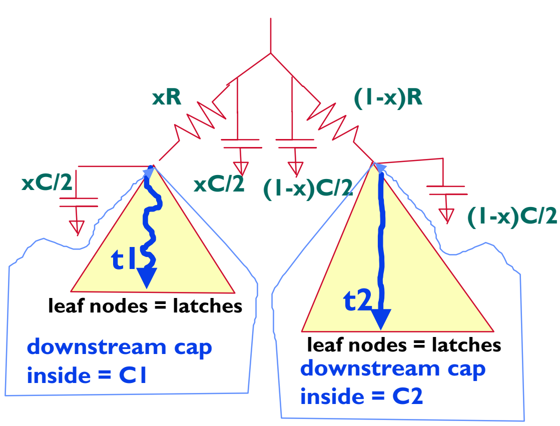


- Delay root -> 8?

- Delay root to 6?

Exact Zero Skew

- So, we can now write delays for our 2 matched trees
  - Assume delay for left tree from its root is t1, for right tree = t2
  - Assume total cap inside left tree = C1, for right tree C2



Delay to left:

xR(xC/2 + C1) +t1

Delay to right:

(1-x)R[(1-x)C/2 + C2)] + t2

Exact Zero Skew

- What do we want to accomplish here?
  - Delay to the left = delay to the right
  - So, we equate the 2 delays, and we get 1 equation in 1 unknown, x

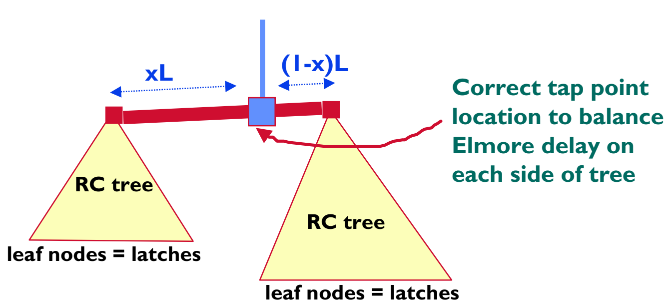
$$xR(xC/2 + C1) + t1 = (1-x)R[(1-x)C/2 + C2)] + t2$$

  - Can solve this analytically, get a unique x solution

$$x = \frac{(t2 - t1) + R[C2 + C/2)}{R(C + C1 + C2)}$$

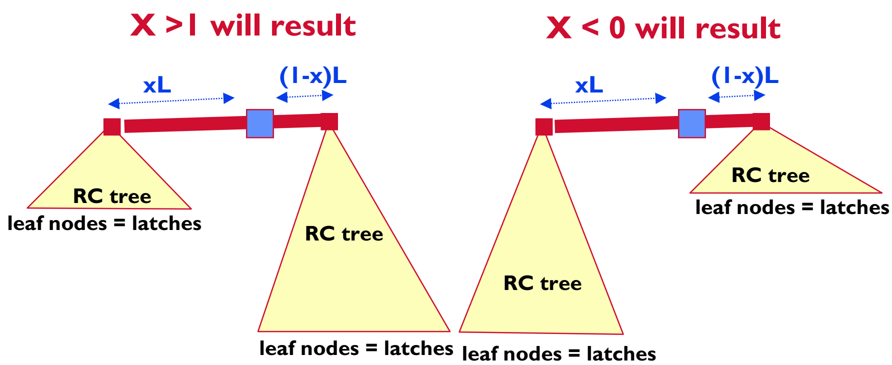
Exact Zero Skew

- Interpretation
  - Value of x tells us where to put the tap point on the matching wire
  - If we put xL units of wire on left, (1-x)L on right, then Elmore delays balance -- assuming that Elmore delays inside each subtree, from subtree root to each leaf in each subtree, also balance
  - Can get "exact zero skew" this way -- hence name of algorithm



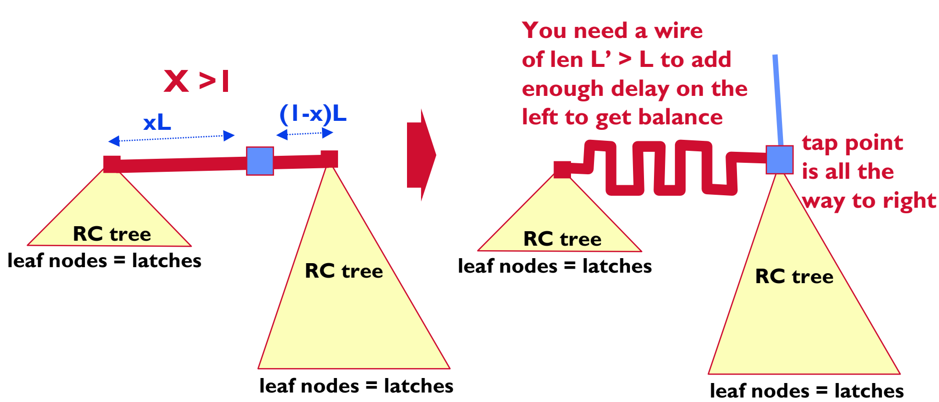
Exact Zero Skew: One Complication…

- You *want* x to come out 0 <= x <= 1
  - ***But it might not…!***
  - **Why not? If the trees are too unbalanced there IS NO tap point that will balance the Elmore delay!**



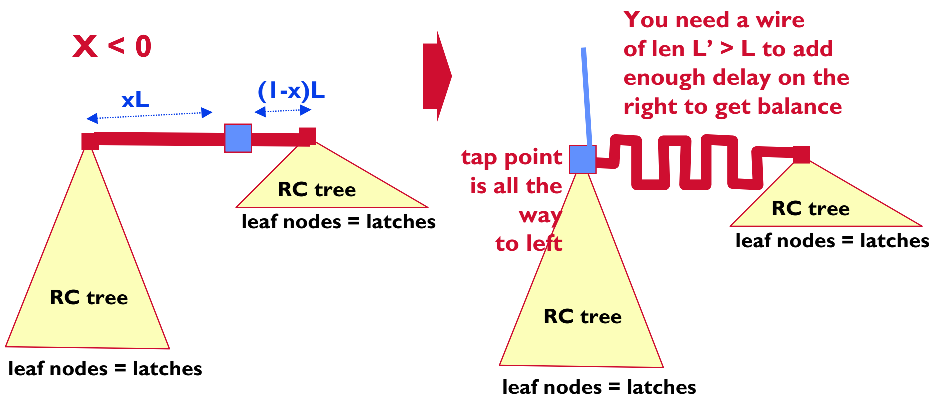
Exact Zero Skew

- Interpretation
  - **The trees are so unbalanced that a minimum length wire connecting the 2 roots of the subtrees is NOT LONG ENOUGH to balance delays**
  - **X<0 or X>1 tells us: *add more wirelength (more C, really) to balance trees.***



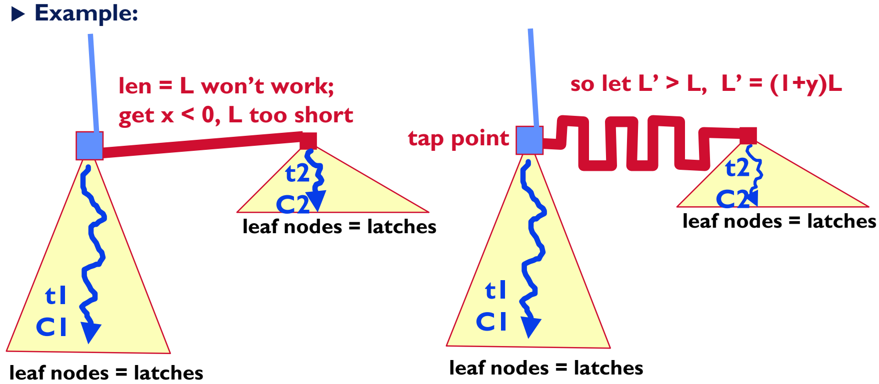
Exact Zero Skew

- Ditto for X < 0



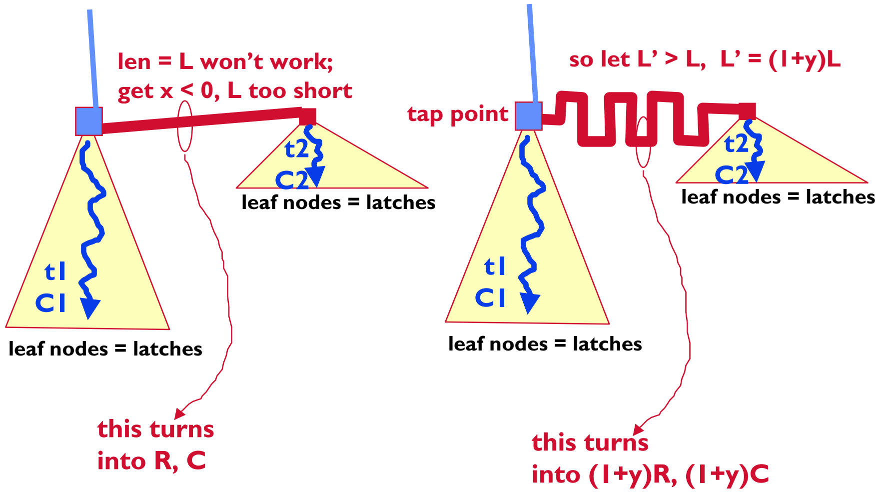
Exact Zero Skew

- New problem
  - If 0 < x < 1, you put in a minimum length straight wire to connect to 2 subtree roots, and then you solve for x fraction for where to tap it
  - If not, you have to solve for the new L' > L that adds enough extra delay so that the delays balance.
  - Example:



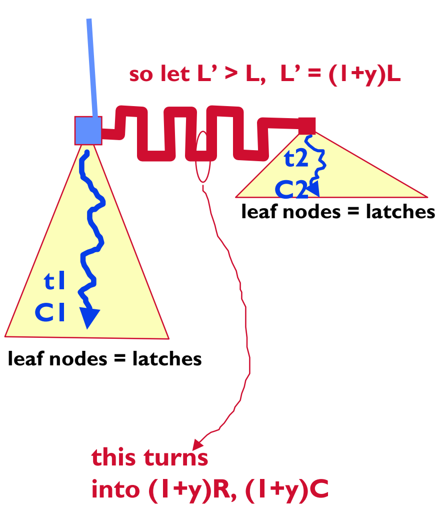
Exact Zero Skew

- Look at R, C for the 2 different segments



Exact Zero Skew

- Can again do this analytically



Get delay on left;

get delay on right;

equate:

$$t1 = (1+y)R\,[\,(1+y)C/2 + C2\,] + t2$$

solve for (y)

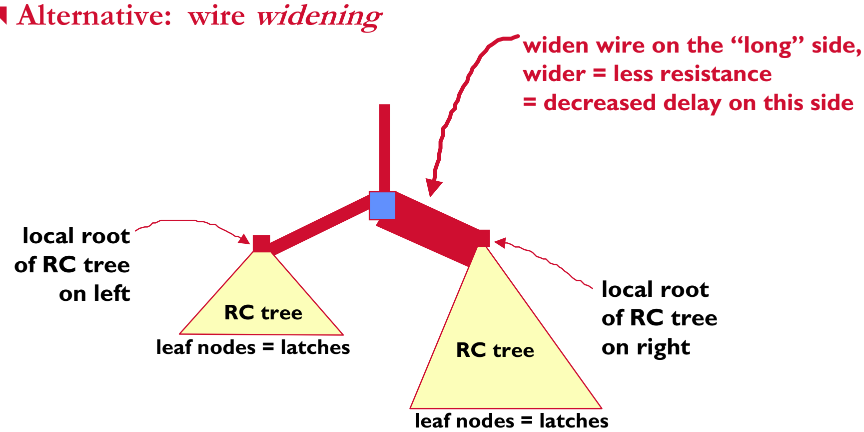
(DO it – not too hard)

Exact Zero Skew

- Can similarly solve for when x>1…
  - **Basically the same answer, with t1 and t2, C1 and C2 switched**
- Utility
  - **If you use a recursive, bottom up approach to geometrically route tree…**
  - **Cool idea is : *at every point where you make a wiring/tapping decision, you strive for perfectly balanced Elmore delay to both subtrees. Can solve analytically for this.***
  - **If all the Elmore delays perfectly balanced, you get: *Exact Zero Skew***

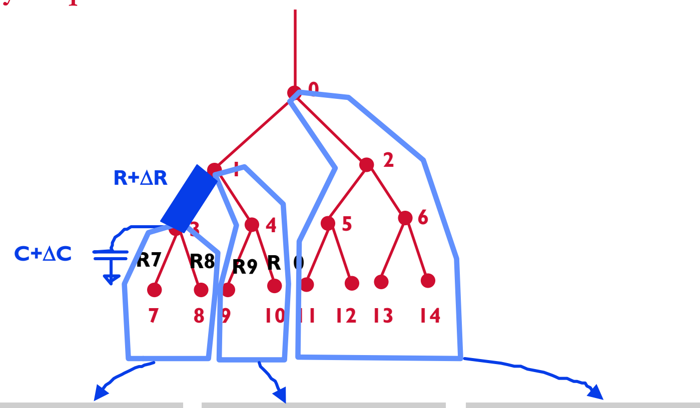
Clock Balancing: By Wire Widening

- Picking right tap point, maybe adding wire is not *only* way
- Alternative: wire *widening*



Widening in a Clock Tree

- Summary of qualitative effects



Delay gets smaller:
R got smaller

Delay gets bigger:
C got bigger

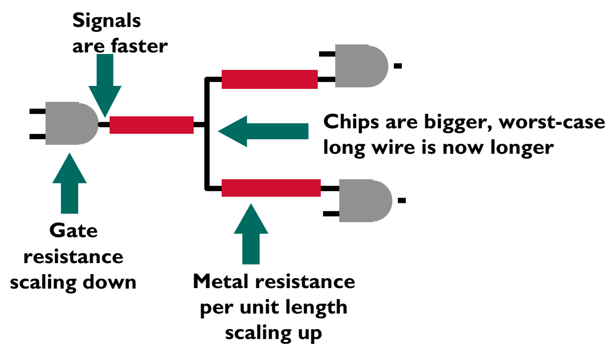
Delay gets bigger, but *by less*:
C got bigger

Summary

- Interconnect increasingly responsible for chip speed
  - Technology is scaling to smaller sizes
  - Chips are being designed to run faster
- Layout tools responsible for part of timing guarantee
  - Upstream tools handle levels of logic, etc
  - Physical design tools responsible for partitioning, placement, routing
  - All of these impact wire length and distribution
- Individual wires modeled as complex circuits
  - From a layout view, RC tree is the nicest, most useful model
  - Elmore delay is easiest to compute delay estimator for 1 in->out
  - Can get the Elmore delay with a little very basic circuits
  - There are sophisticated estimators beyond Elmore...
  - Can use for both verification, and for layout optimizations (eg clock)

Appendix: Why the Delay Trends?

- **Qualitative answer**
  - **Signals propagate through the physical materials of gates, wires with finite delay**
  - **Wires, gates getting physically smaller, but interactions of the low-level technology parameters is complicated...**

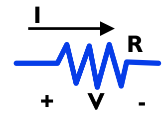


Deriving the Elmore Delay

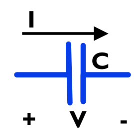
- **From first principles**
  - **Avoid complex linear system theoretic math**
  - **Want to do this with plain old Kirchhoff laws and some basic circuit analysis, and some simple calculus**
- **Turns out to be not too hard**
  - **Though it does turn on a few representation tricks for the algebra that are not obvious…**

RC Trees: Back to Circuit Basics

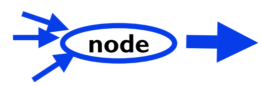
- How resistors work
  - V = IR



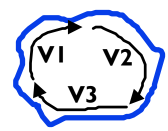
- How capacitors work
  - I = C dV/dt



- Kirchhoff's current law
  - Current is conserved, Σ(current into node) = Σ(current out of node)

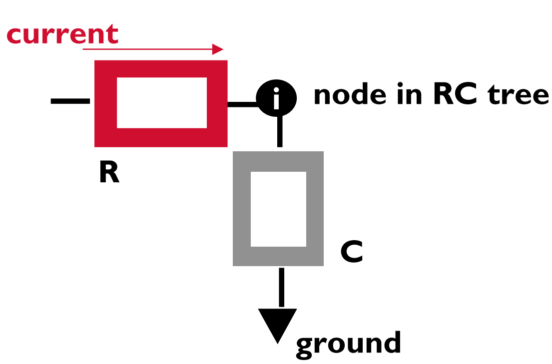


- Kirchhoff's voltage law
  - Σ (voltage drop around closed circuit loop) = 0



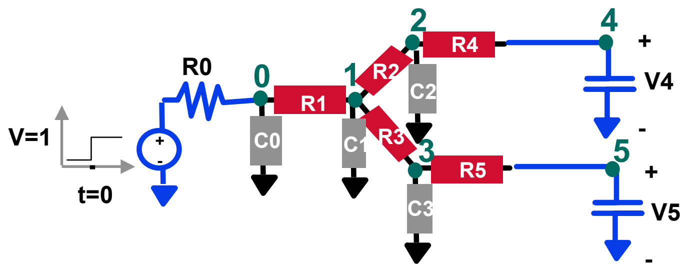
RC Trees:

- Observe
  - Combine ("lump") load capacitance with 1/2C from last segment
  - In RC tree, each R and each C may be different
  - Give each a name: Ri feeds into node i, Ci hangs off node i
  - Label currents thru Ri as Ii



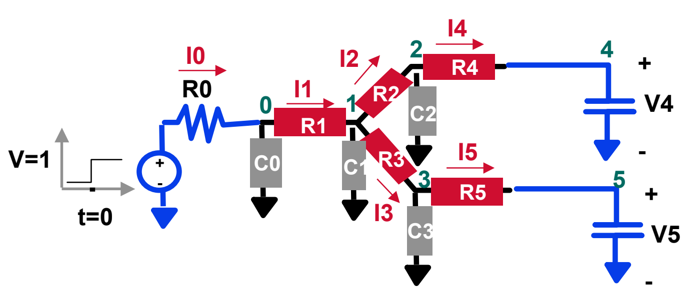
RC Trees

- So, let's label our little example this way… first the nodes



RC Trees

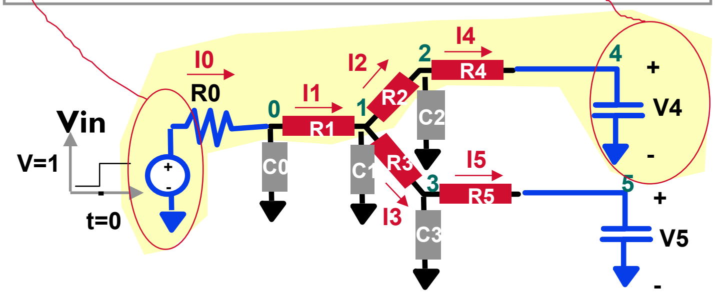
- Now, let's label all the currents thru resistors too



RC Trees: Elmore Delay

- What do we really want to get?
  - Approx. output waveforms, V4(t), V5(t), as efficiently as possible
- First: write KVL from input to an output, say V4

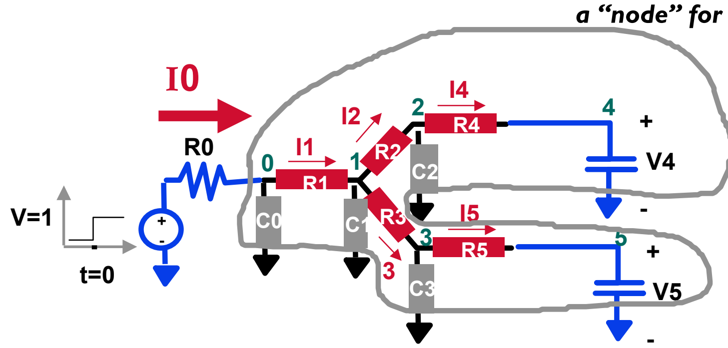
$$Vin - R0 \cdot I0 - R1 \cdot I1 - R2 \cdot I2 - R4 \cdot I4 - V4 = 0$$



RC Trees: Elmore Delay

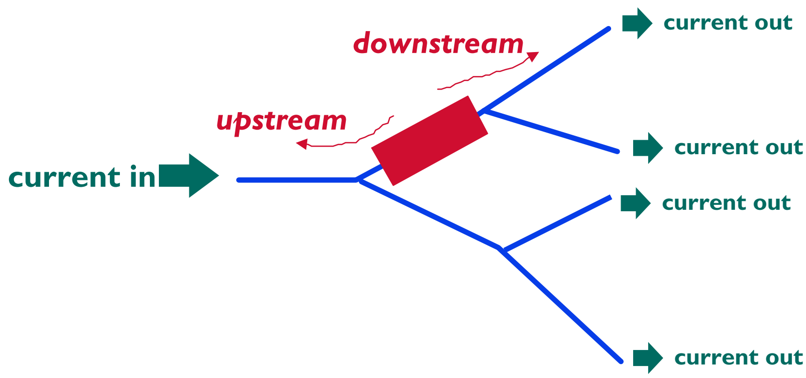
- OK, so what are the currents Ii?
  - Trick involving KCL observation
  - Look at current I0, it flows into "downstream" part of tree
  - What flows out of this part of the tree?
    - *Only current thru capacitors to ground*

This is still like a "node" for KCL



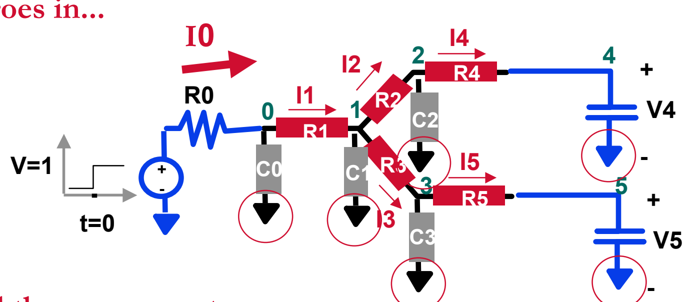
Aside: Stream Analogy

- Think of current like real water, flowing in tree
  - From any component of tree, if you look at what is happening back up toward the root, it's UPSTREAM
  - Look toward leaves, its DOWNSTREAM

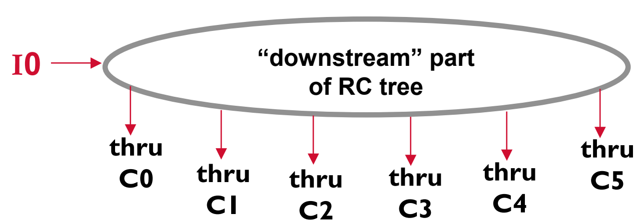


RC Trees: Elmore Delay

- I0 goes in...

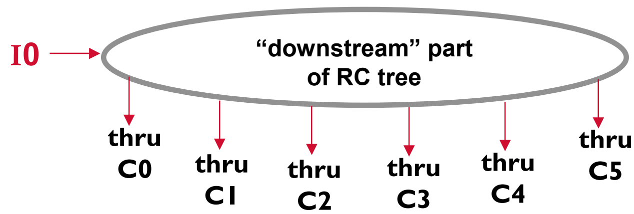


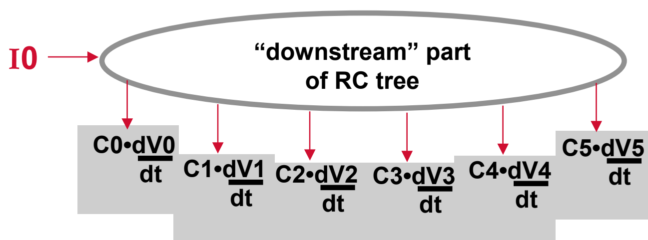
- ...all these come out



RC Trees

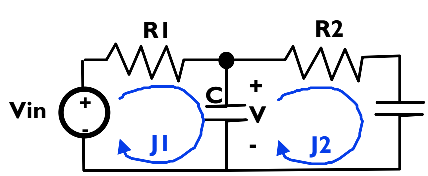
- Can we write an equation for these currents out?





RC Trees: Elmore Delay

- Suggests a change in strategy
  - Let’s try to express *everything* interesting in the circuit using only combinations of the *currents* thru these capacitors
  - Let’s call current thru Ck as Jk (and we know Jk = Ck•dVk/dt)
- Idea
  - Use superposition in the form of mesh analysis
  - Currents add up in each branch of the circuit

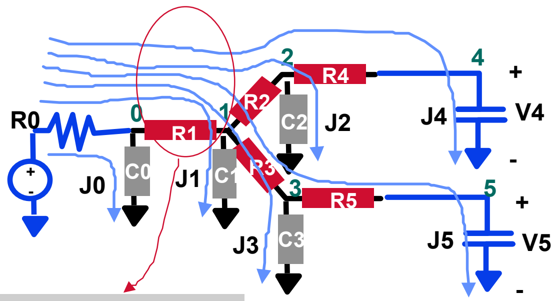


What’s current thru cap C? J1-J2

What’s KCL at top of C? J1 - J2 - C*dV/dt

RC Trees: Elmore Delay

- Let's relabel using only Jk currents thru caps
- Observe
  - Each current has a unique path, root to ground
  - Total current thru any resistor = $\boxed{\Sigma \text{ (Jk thru downstream caps)}}$
  - Ex: R1



currents = J1 + J2 +J3 +J4 +J5

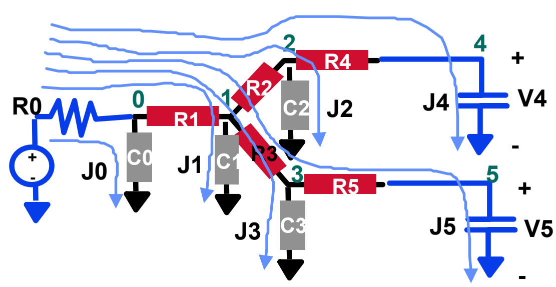
RC Trees: Elmore Delay

- Let's write KVL from Vin to V4 again

Vin - R0•(J0+J1+J2+J3+J4+J5) - R1•(J1+J2+J3+J4+J5)
- R2•(J2+J4) - R4•(J4) - V4 = 0

- Let's factor it over the J's instead of the R's

Vin - J0•(R0) - J1•(R0+R1) - J2•(R0+R1+R2) - J3•(R0+R1)
- J4•(R0+R1+R2+R4) - J5•(R0+R1) - V4 = 0

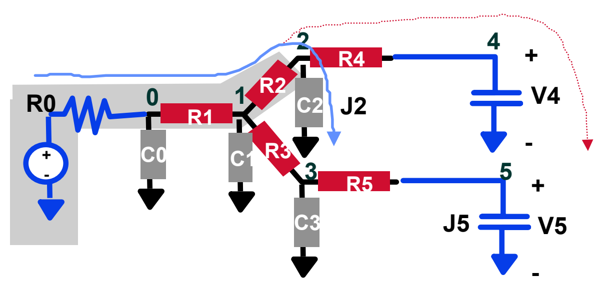


RC Trees: Elmore Delay

- What are these "sums of R's" on each J?
  - "Upstream" resistance on the unique path from root to V4 seen by the current Jk thru each capacitor Ck

Vin - J0•(R0) - J1•(R0+R1) - J2•(R0+R1+R2) - J3•(R0+R1) - J4•(R0+R1+R2+R4) - J5•(R0+R1) - V4 = 0

  - Define this as **R0k;** rewrite above as $Vin - \Sigma_k\ R0k \bullet Jk - V4 = 0$



RC Trees: Elmore Delay

- Swell, but we still *don't* have V4(t)...
  - Replace Jk by Ck•dVk/dt

$$Vin(t) - \Sigma_k\ R0k \bullet Ck \bullet dVk/dt - V4(t) = 0$$

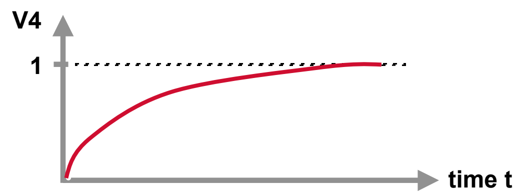
  - Assume Vin(t) is a 1 V step applied at time = 0; rearrange

$$1 - V4(t) = \Sigma_k\ R0k \bullet Ck \bullet dVk/dt$$

- Problems
  - We don't know V4(t) -- it's what we want to solve for
  - We don't know all those C dV/dt derivatives at leaves either
  - We need a couple of tricks to get around these...

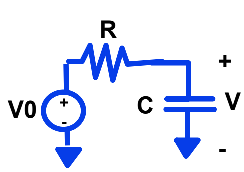
RC Trees: Elmore Delay

- Trick: what does V4(t) actually do, as a waveform?
  - Step back for a moment and think: what will V4(t) look like?
  - Answer: some exponential ramp rising from 0V to a 1V asymptote
  - Why? The 1V step input supplies current to charge capacitors in the RC tree; eventually they all charge up, current stops flowing, voltages become constant



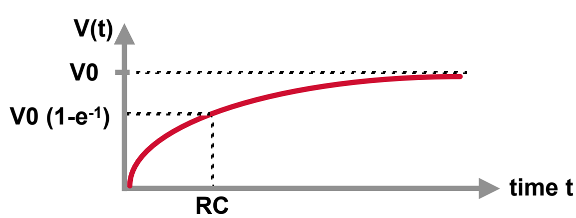
RC Trees: Elmore Delay

- Recall: Apply a voltage step to a circuit with a capacitor…
  - Current starts to flow…
  - Eventually the cap charges up, and current stops flowing
  - Cap charges up to V0 here
  - Current I eventually goes to 0



KVL: $V0 - R \cdot C \cdot dV/dt - V = 0$

Solve diff. eq: $V(t) = V0\ (1 - e^{-t/RC})$



RC Trees: Elmore Delay

- OK, but we have a whole *tree* of Rs and Cs…
- Trick: let's integrate both sides to get rid of those derivatives
  - Look at our expression for 1 - V4(t)
  - Integrate it, from 0 to ∞

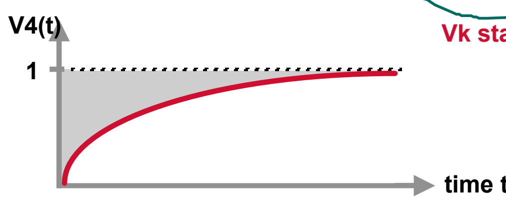
$$1 - V4(t) = \Sigma_k \, R0k \bullet Ck \bullet dVk(t)/dt$$

$$\int_0^\infty (1 - V4(t))dt = \int_0^\infty \Sigma_k \, R0k \bullet Ck \bullet dVk/dt$$

$$\text{“area above curve”} = \Sigma_k \, R0k \bullet Ck \bullet Vk \Big|_0^\infty = \Sigma_k \, R0k \bullet Ck \bullet 1 - 0$$

Vk -> 1 as t -> ∞

Vk starts uncharged



RC Trees: Elmore Delay

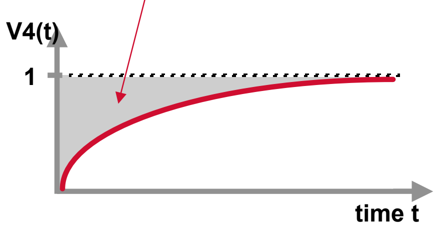
- Are we getting anywhere? Yes...

$$1 - V4(t) = \Sigma_k \, R0k \bullet Ck \bullet dVk(t)/dt$$

$$\boxed{\int_0^\infty (1 - V4(t))dt} = \int_0^\infty \Sigma_k \, R0k \bullet Ck \bullet dVk/dt$$

$$= \Sigma_k \, R0k \bullet Ck \bullet Vk \Big|_0^\infty \quad \boxed{= \Sigma_k \, R0k \bullet Ck}$$

Aha! This is what we need, a simple expression for this integral involving only quantities we know.



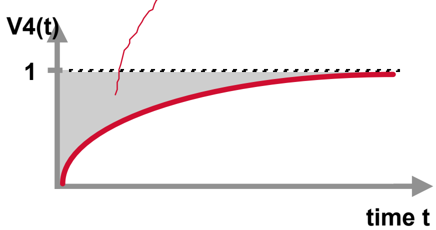
RC Trees: Elmore Delay

- Turns out this is enough for our needs
  - Let's assume that V4(t) follows an exponential rise, just like a circuit with a single **R** and a single **C**; let τ = **R•C** here.
  - So, we shall assume that

$$V4(t) = 1 - e^{-t/\tau}$$

  - ..but we don't know τ. But we do know the area above V4(τ)!

$$\int_0^\infty (1 - V4(t))dt = \int_0^\infty [1 - (1 - e^{-t/\tau})]dt = \Sigma_k \; R0k \bullet Ck$$



Solve for τ

$$\boxed{\Sigma_k \; R0k \bullet Ck = \tau}$$

RC Trees: The Elmore Delay

- This is the magic formula that we want

$$\boxed{V4(t) = 1 - e^{-t/\tau} \qquad \tau = \Sigma_k \; R0k \bullet Ck}$$

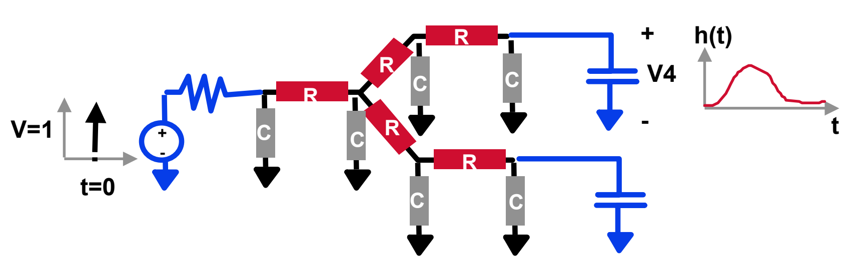
- τ is "the Elmore Delay"; recall:
  - We asked this: *what does this RC tree leaf voltage Vi(t) look like?*
  - We assumed this: *apply 1V step at t=0*
  - We also assumed: *can model voltage Vi(t) as 1 time constant,* $1 - e^{-t/\tau}$
  - We derived this: $\tau = \Sigma_k$ R0k•Ck
- Note
  - A general formula for the time constant for the response at any leaf
  - (Nothing in top eqn is really specific to node 4, except which resistors)
  - Assume one time constant τ is a good approx for the actual delay

Observations

- **Note**
  - **Basically says we can model the output at 1 leaf of an RC tree with an "equivalent circuit" that looks like 1 equivalent R, 1 eqv. C**
  - **We don't really know the R or the C though, just that RC = τ**
  - **Called a "one time constant" model (makes sense, eh?)**
- **Analysis**
  - **PRO: *Easy* to compute (can do it recursively by walking tree)**
  - **PRO: Gives you a *unique* delay for *each* output of the tree**
  - **PRO: Accounts for *all* the parasitics Rs, Cs of the interconnect**
  - **CON: It's still only a one time constant model; sometimes need > 1**

Elmore Delay: Circuits Aside

- **That magic τ is actually derivable several other ways**
  - **Recall that for any linear system (circuit) you can characterize it by it's impulse response, denoted h(t), which is what comes out when you put in a Dirac $\delta(\tau)$**



Elmore Delay: Circuits Aside

- **Turns out you can see more in frequency domain**
  - Use the Laplace transform, which turns differential eqns into plain, old algebraic equations

$$F(s) = \int_0^{\infty} f(t)\, e^{-st}\, dt$$

$$H(s) = \int_0^{\infty} h(t)\, e^{-st}\, dt = \int_0^{\infty} h(t)\, [1 + (-st)/1! + (-st)^2/2! + \ldots]\, dt$$

$$= \underbrace{\int_0^{\infty} h(t)dt}_{\text{0th moment of } h(t)} + (-s) \underbrace{\int_0^{\infty} t \bullet h(t)\, dt}_{\text{1st moment of } h(t)} + (-s)^2 \underbrace{\int_0^{\infty} t^2 \bullet h(t)\, dt}_{\text{2nd moment of } h(t)} + \ldots$$

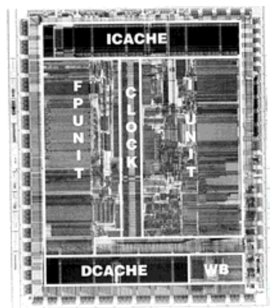
= Elmore delay Σ_k R0k•Ck

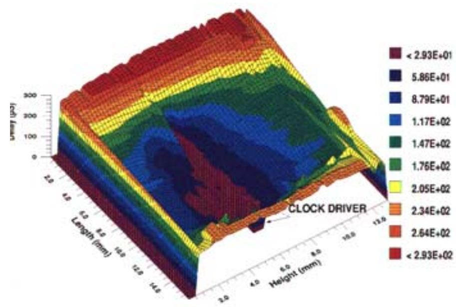
Elmore Delay: Circuits Aside

- **Elmore delay uses the 1st moment of h(t) to approximate the response of the circuit to a voltage step applied at t=0**
  - 1 moment gives you 1 time constant, so you follow 1 exp rise
- **What happens if you want more accuracy?**
  - You need to use more of these moments in your approximation
  - Technique called "moment matching"
  - Assumes you can get 'em, then "curve fit" a response waveform
  - Best known algorithms for doing it?
    - AWE: Asymptotic Waveform Eval., [Rohrer & Pillage TCAD90]
    - Lots of follow-on work to this
    - You need to use some subtle circuits ideas to get more than the first moment, stuff beyond our self-imposed I=C•dV/dt limit

Circuit Aside: AWE Example

- Evaluation of clock signal network on DEC Alpha
  - 1st generation ALPHA chip, clock analyzed using AWE techniques
  - This allows us to get a more accurate delay than Elmore, using more than one time constant





Arrival time of clock (ps)
as function of position on chip;
Note clock driver is in chip center