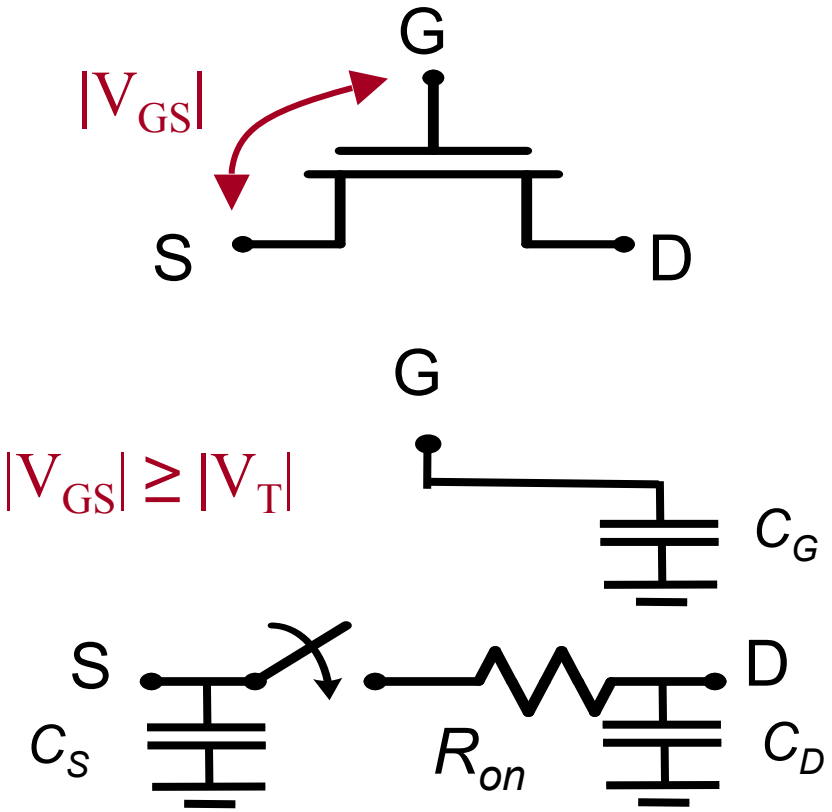
EECS 151/251A Spring 2019

Digital Design and Integrated Circuits

Instructor:
John Wawrzynek

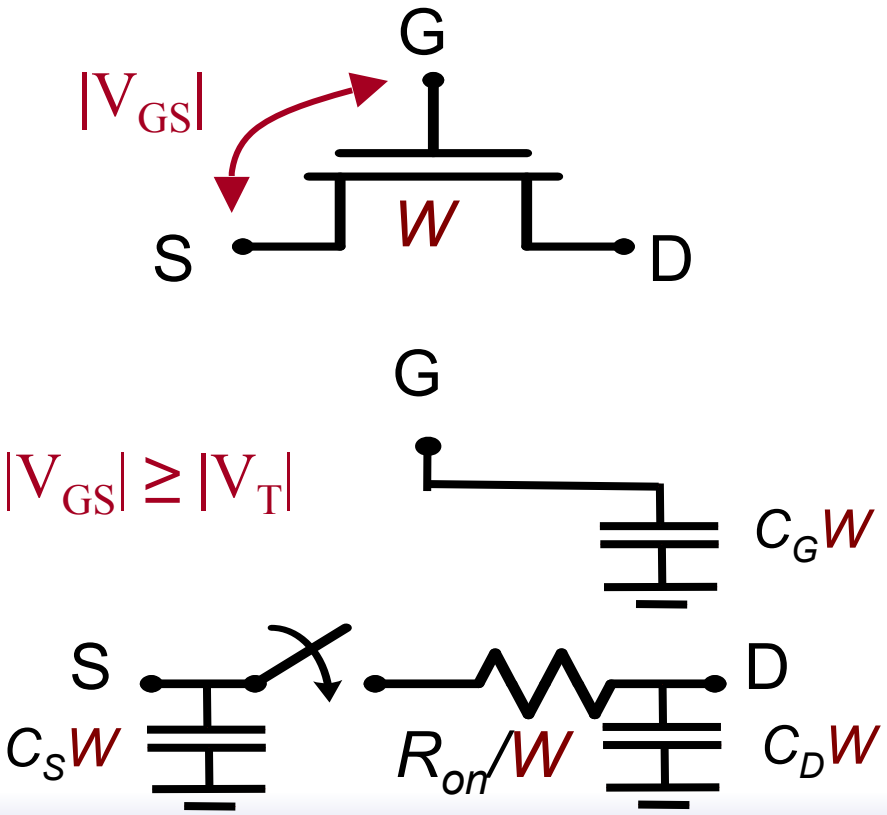
Lecture 10

The Switch – Dynamic Model (Simplified)



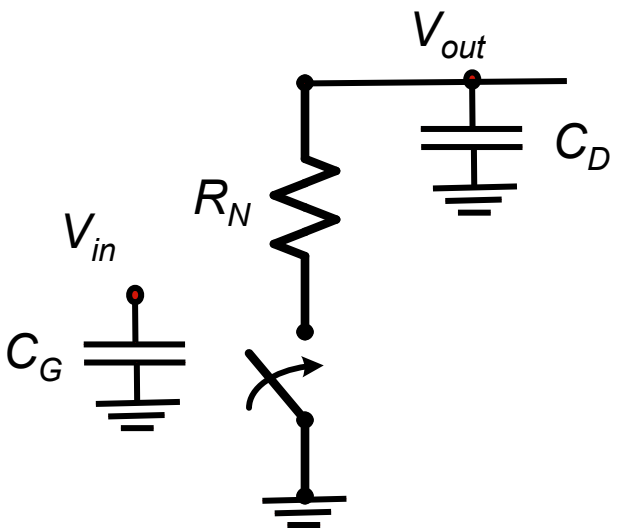
Switch Sizing

What happens if we make a switch W times larger (wider)

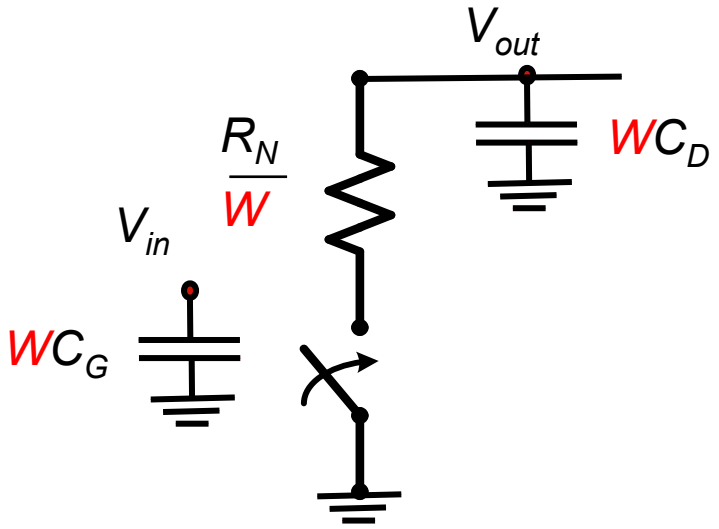


Switch Parasitic Model

The pull-down switch (NMOS)



Minimum-size switch

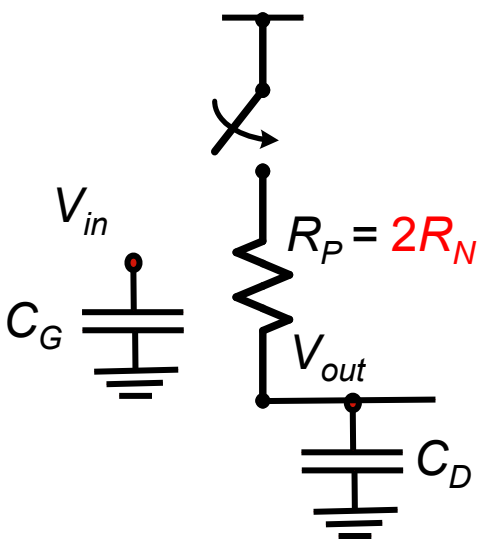


Sizing the transistor (factor W)

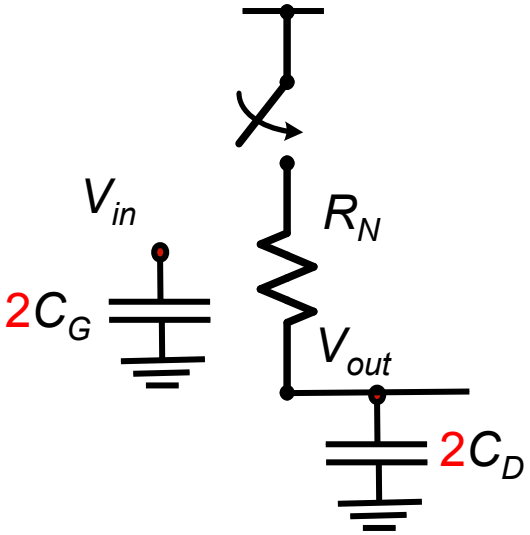
We assume transistors of minimal length (or at least constant length). R's and C's in units of per unit width.

Switch Parasitic Model

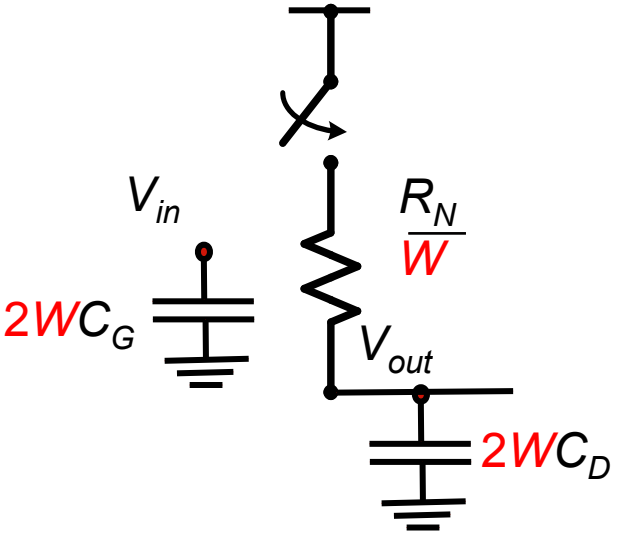
The pull-up switch (PMOS)



Minimum-size switch

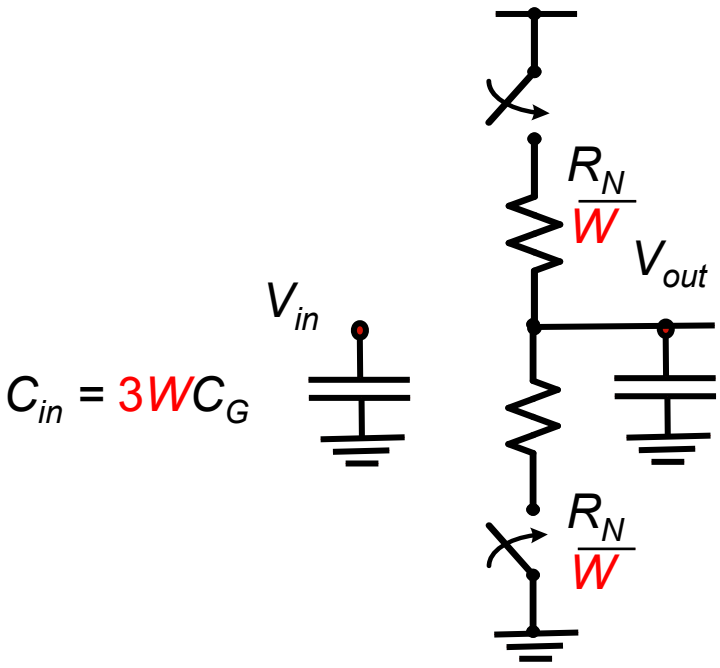


Sized for symmetry



General sizing

Inverter Parasitic Model



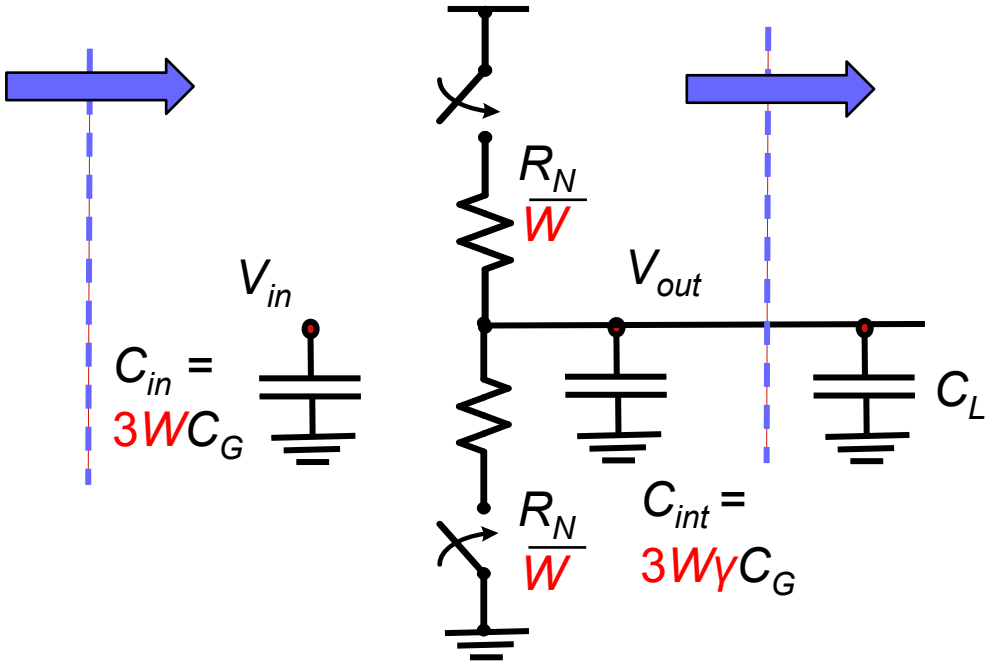
Drain and gate capacitance of transistor are **directly** related by process ($\gamma \approx 1$)

$$C_D = \gamma C_G$$

$$t_p = 0.69\left(\frac{R_N}{W}\right)(3W\gamma C_G) = 0.69(3\gamma)R_N C_G$$

Intrinsic delay of inverter independent of size

Inverter with Load Capacitance



$$t_p = 0.69(R_N/W)(C_{int} + C_L)$$

$$= 0.69(R_N/W)(3W\gamma C_G + C_L)$$

$$= 0.69(3\gamma R_N C_G)(1 + \frac{C_L}{\gamma C_{in}})$$

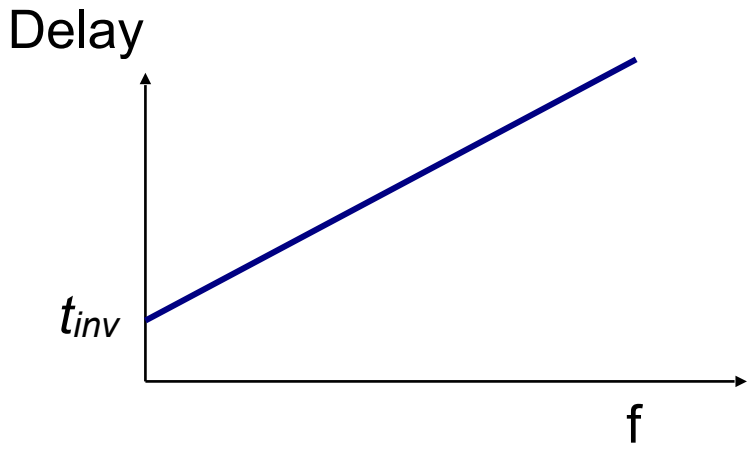
$$= t_{inv}(1 + \frac{C_L}{\gamma C_{in}}) = t_{p0}(1 + f/\gamma)$$

f = **fanout** = ratio between load and input capacitance of gate

Inverter Delay Model

Delay linearly proportional to fanout, f.
For f=0, delay is t_{inv}

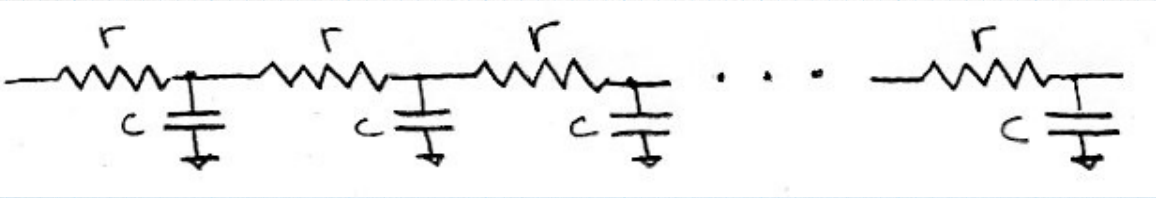
$$t_p = t_{p0}(1+f/\gamma)$$



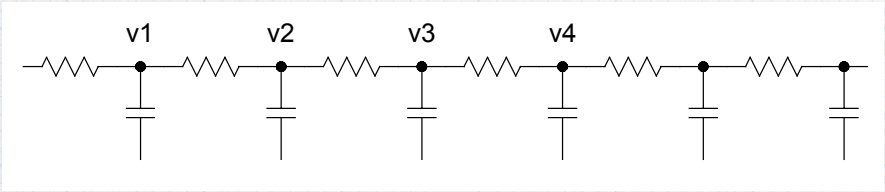
Question: how does transistor sizing (W) impact delay?

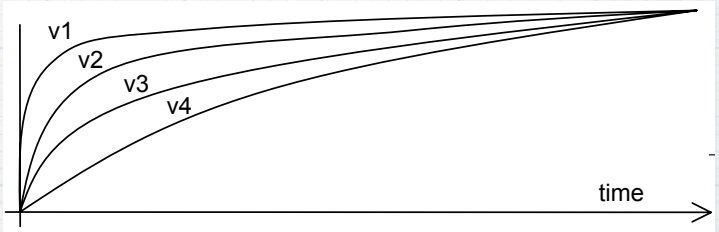
Adding Wires to gate delay

- Wires have finite resistance, so have distributed R and C:



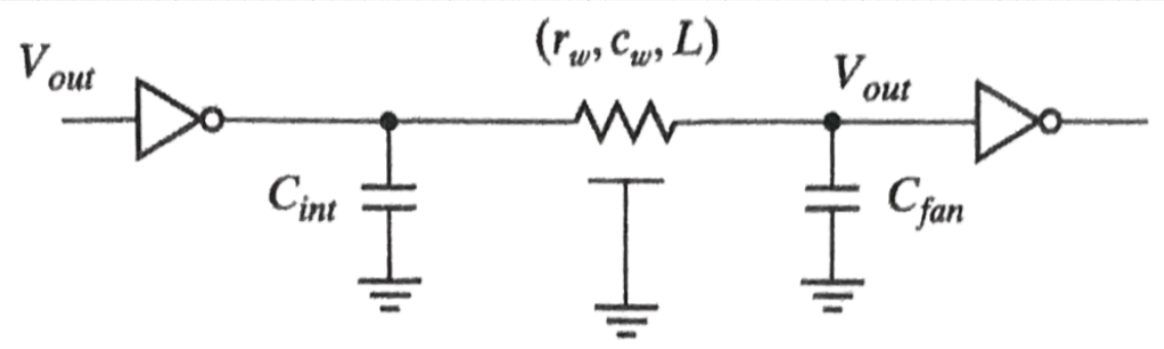
with r = res/length, c = cap/length, $\Delta t \propto rcL^2 \cong rc + 2rc + 3rc + \ldots$





- Wire propagation delay is around half of what it would be if R and W were "lumped": $t_p = 0.38(rL * cL) = 0.38rcL^2$

Gate Driving long wire and other gates



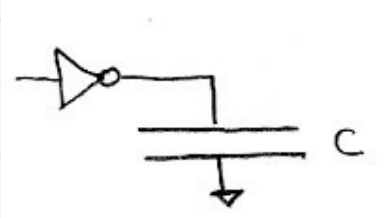
$$R_w = r_w L, \quad C_w = c_w L$$

$$t_p = 0.69R_{dr}C_{int} + 0.69R_{dr}C_w + 0.38R_wC_w + 0.69R_{dr}C_{fan} + 0.69R_wC_{fan}$$

$$= 0.69R_{dr}(C_{int} + C_{fan}) + 0.69(R_{dr}c_w + r_wC_{fan})L + 0.38r_wc_wL^2$$

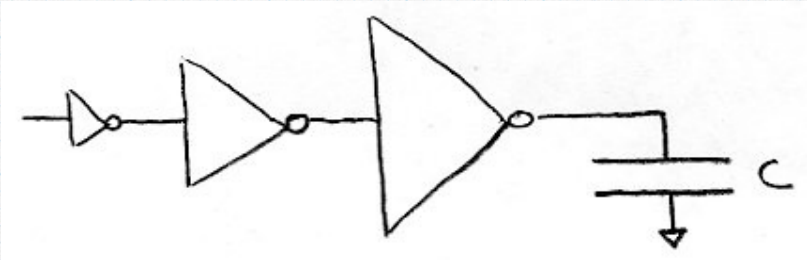
Driving Large Loads

- Large fanout nets: clocks, resets, memory bit lines, off-chip
- Relatively small driver results in long rise time (and thus large gate delay)



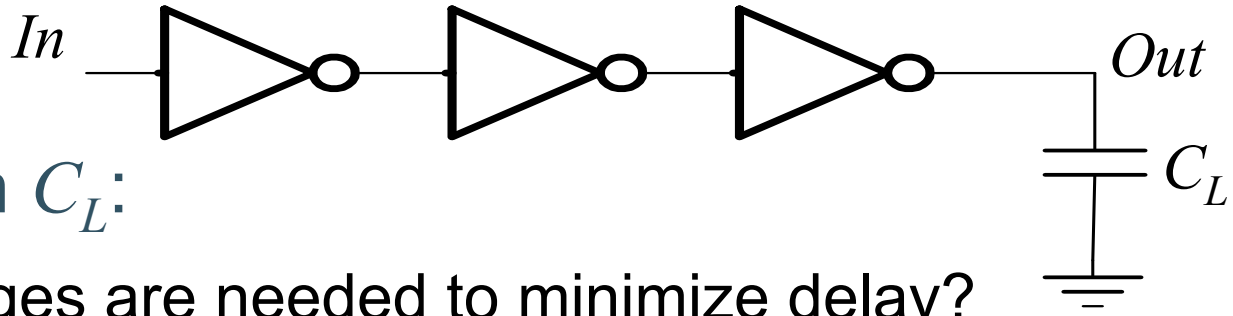
- Strategy:

Staged Buffers

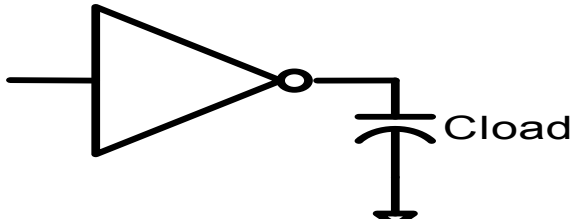


- How to optimally scale drivers?
- Optimal trade-off between delay per stage and total number of stages?

Driving Large Loads



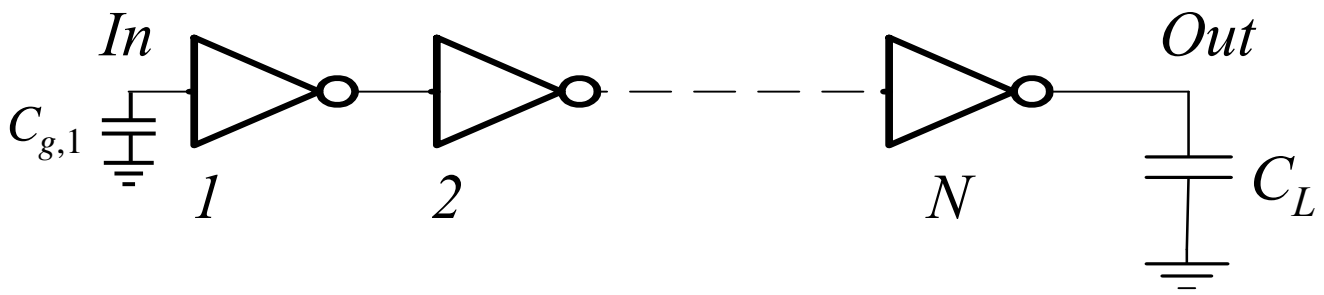
- For some given C_L:
  - How many stages are needed to minimize delay?
  - How to size the inverters?
- Get fastest delay if build one **very** big inverter
  - So big that delay is set only by self-loading



- Likely not the solution you're interested in
  - Someone has to drive this inverter…

Delay Optimization

- First assume given:
  - A fixed number of inverters
  - The size of the first inverter
  - The size of the load that needs to be driven
- What is the minimal delay of the inverter chain



- Delay for the *j-th* inverter stage:

$$t_{p,j} = t_{p0}\left(1 + \frac{C_{g,j+1}}{\gamma C_{g,j}}\right) = t_{p0}(1 + f_j/\gamma)$$

- Total delay of the chain:

$$t_p = \sum_{j=1}^{N} t_{p,j} = t_{p0} \sum_{j=1}^{N} \left(1 + \frac{C_{g,j+1}}{\gamma C_{g,j}}\right), C_{g,N+1} = C_L$$

Optimum Delay and Number of Stages

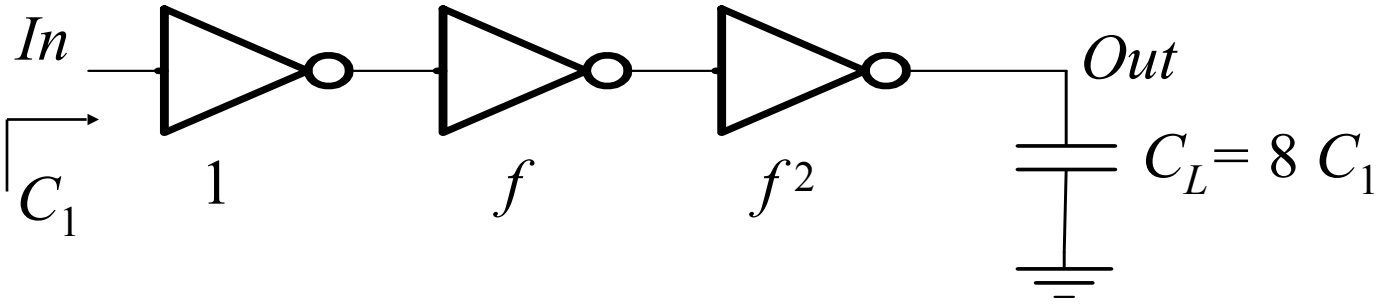
- Each inverter should be sized up by the same factor f with respect to the preceding gate
- Therefore each stage has the same delay
- Given $C_{g,1}$ and C_L

$$f = \sqrt[N]{C_L / C_{g,1}} = \sqrt[N]{F}$$

- Where F represents the overall fan-out of the circuit
- The minimal delay through the chain is

$$t_p = N t_{p0}(1 + \sqrt[N]{F}/\gamma)$$

Example



C_L/C_1 has to be evenly distributed across $N = 3$ stages:

Delay Optimization

- Now assume given:
  - The size of the first inverter
  - The size of the load that needs to be driven
- Minimize delay by finding optimal number and sizes of gates
- So, need to find N that minimizes:

$$t_p = Nt_{p0}(1 + \sqrt[N]{F}/\gamma), \quad F = C_L/C_{g,1}$$

Finding optimal fanout per stage

$$t_p = Nt_{p0}(1 + \sqrt[N]{F}/\gamma), \quad F = C_L/C_{g,1}$$

- Differentiate w.r.t. N and set = 0:

$$\gamma + \sqrt[N]{F} - \frac{\sqrt[N]{F}\, lnF}{N} = 0$$

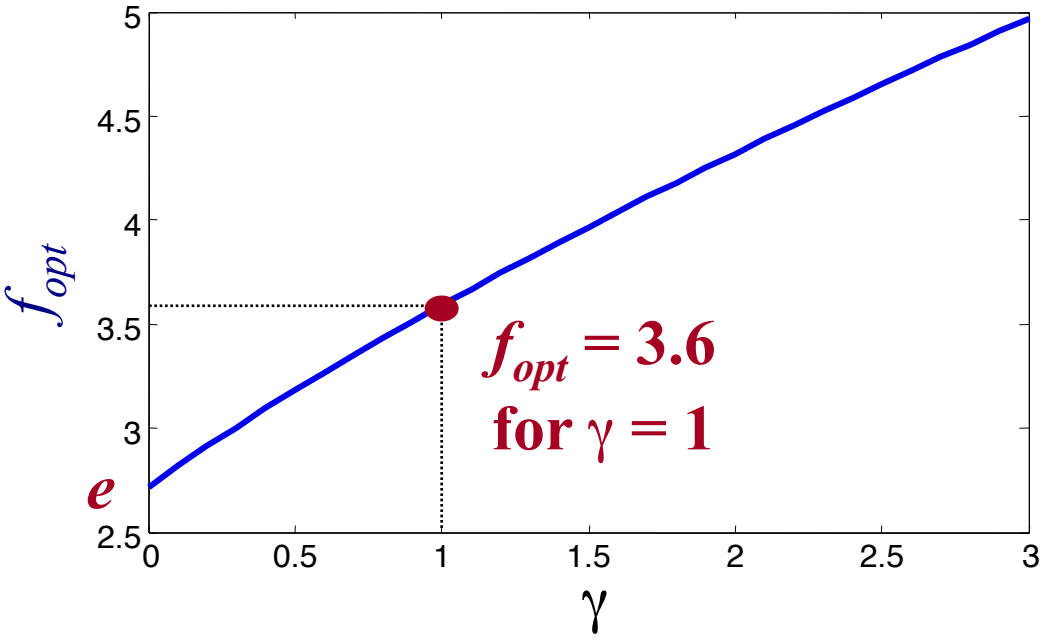
$$\Rightarrow \quad \boxed{f = e^{(1+\gamma/f)}}$$

- Closed form only if : $\gamma = 0 \Rightarrow N = ln(F), f = e$

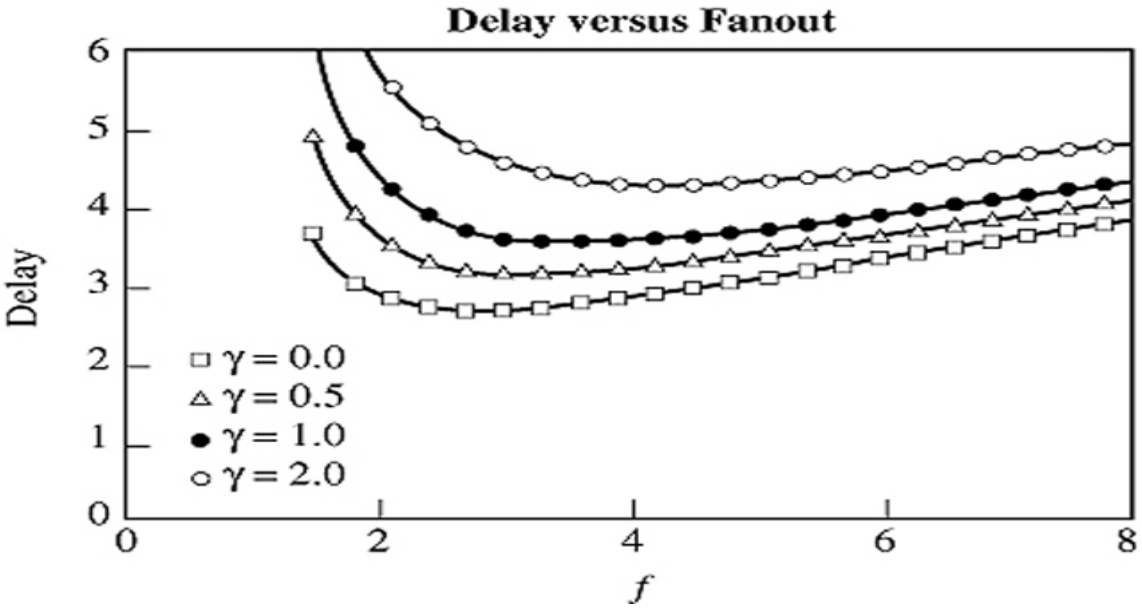
Optimum Effective Fanout f

- Optimum f for given process defined by γ

$$f = e^{(1+\gamma/f)}$$



In Practice: Plot of Total Delay



[Hodges, p.281]

- Why the shape?
- Curves very flat for $f > 2$
  - Simplest/most common choice: $f = 4$

Normalized Delay As a Function of F

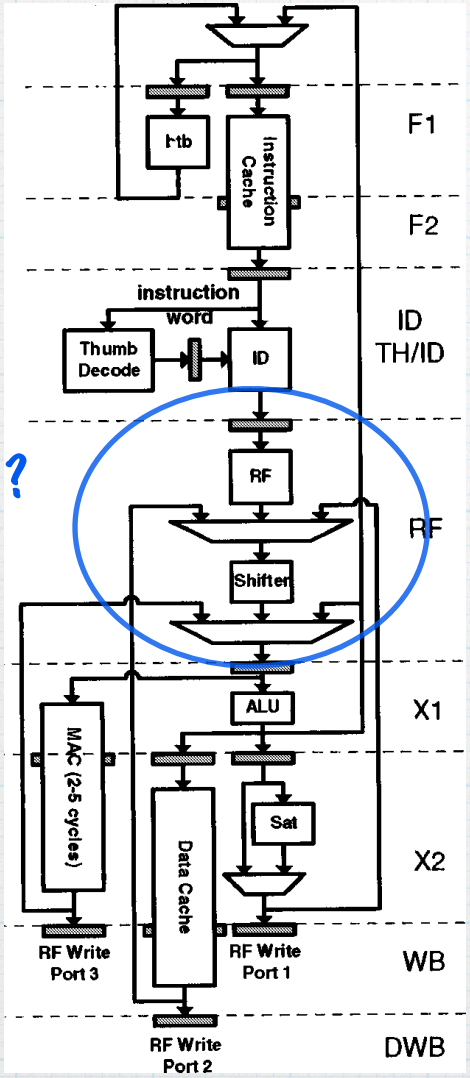
$$t_p = Nt_{p0}(1 + \sqrt[N]{F}/\gamma), \quad F = C_L/C_{g,1}$$

F	Unbuffered	Two Stage	Inverter Chain
10	11	8.3	8.3
100	101	22	16.5
1000	1001	65	24.8
10,000	10,001	202	33.1

($\gamma = 1$)

[Rabaey: page 210]

Timing Closure: Searching for and beating down the critical path



Must consider all connected register pairs, paths, plus from input to register, plus register to output.

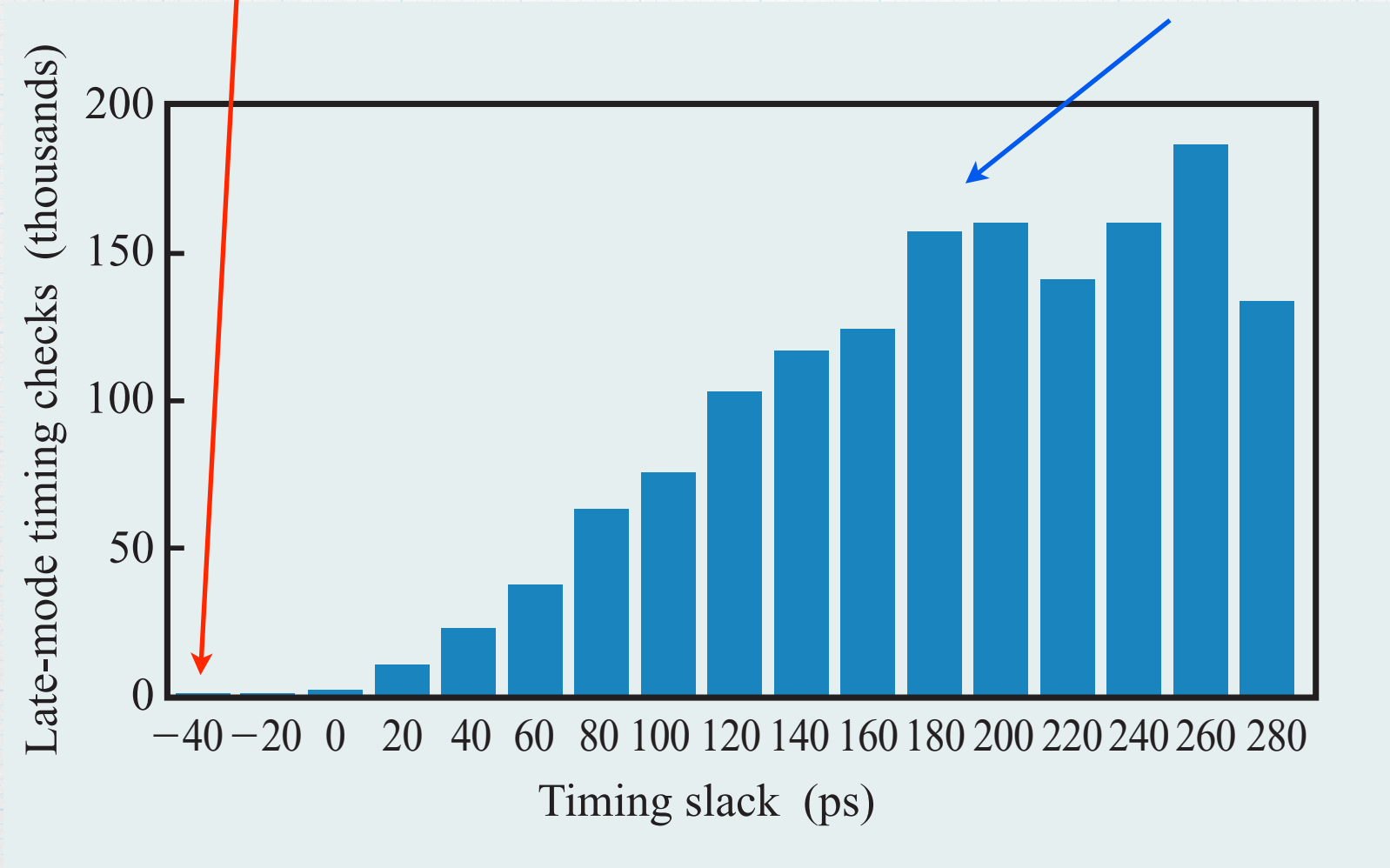
- Design tools help in the search.
- **Synthesis tools** work to meet clock constraint, report delays on paths,
  - Special **static timing analyzers** accept a design netlist and report path delays,
  - and, of course, **simulators** can be used to determine timing performance.

Tools that are expected to **do something** about the timing behavior (such as synthesizers), also include provisions for specifying input arrival times (relative to the clock), and output requirements (set-up times of next stage).

Timing Analysis, real example

The critical path

Most paths have hundreds of picoseconds to spare.



From "The circuit and physical design of the POWER4 microprocessor", IBM J Res and Dev, 46:1, Jan 2002, J.D. Warnock et al.

Timing Optimization

As an ASIC/FPGA designer you get to choose:

- The algorithm
- The Microarchitecture (block diagram)
- The RTL description of the CL blocks (number of levels of logic)
- Where to place registers and memory (the pipelining)
- Overall floorplan and relative placement of blocks

How to retime logic

Circles are combinational logic, labelled with delays.

Critical path is 5. We want to improve it without changing circuit semantics.

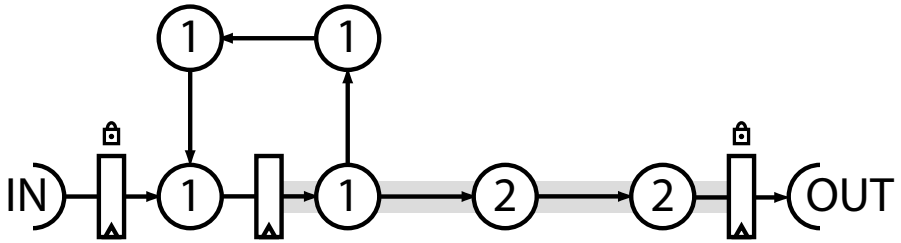


Figure 1: A small graph before retiming. The nodes represent logic delays, with the inputs and outputs passing through mandatory, fixed registers. The critical path is 5.

Add a register, move one circle. Performance improves by 20%.

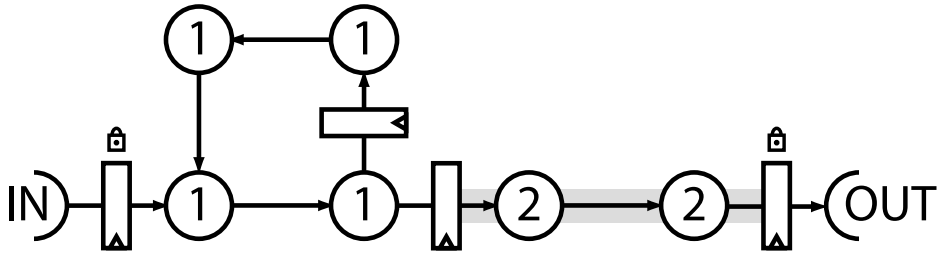
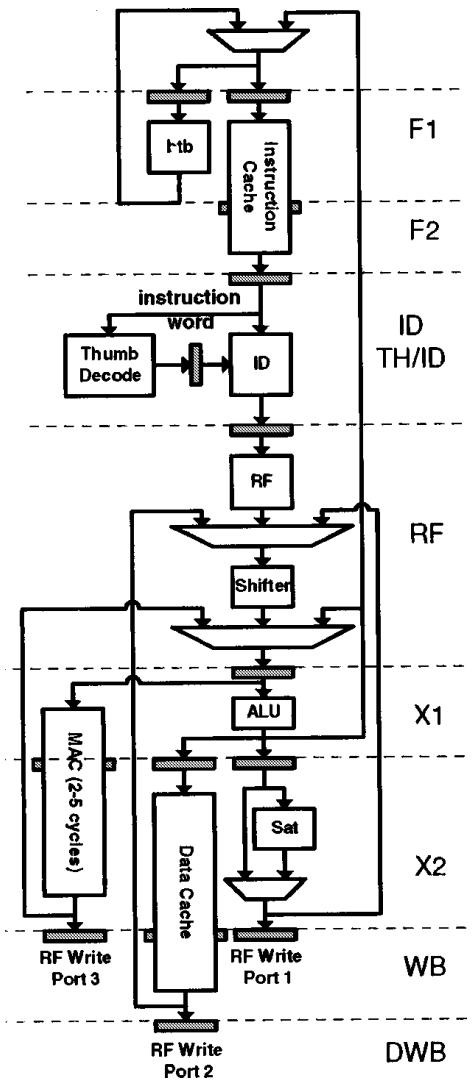
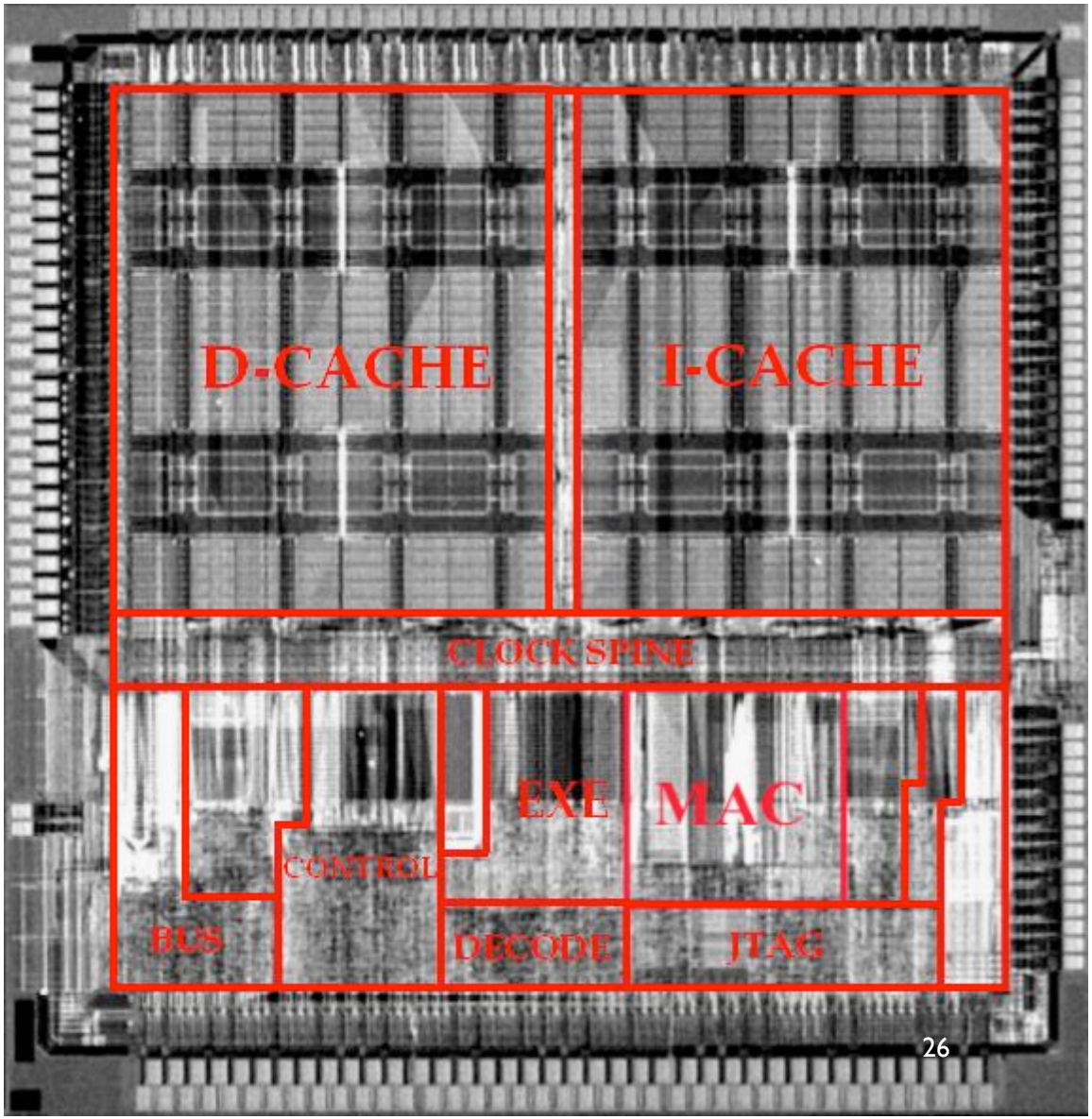


Figure 2: The example in Figure 2 after retiming. The critical path is reduced from 5 to 4.

Logic Synthesis tools can do this in simple cases.

Floorplaning: essential to meet timing.

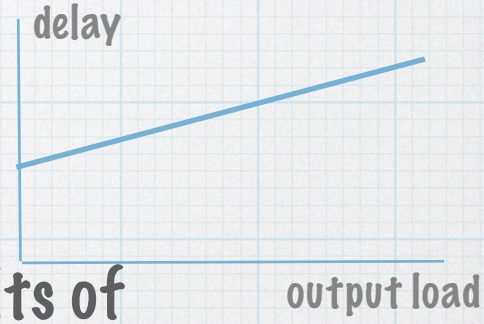




(Intel XScale 80200)

Timing Analysis Tools

- Static Timing Analysis: Tools use delay models for gates and interconnect. Traces through circuit paths.
  - Cell delay model capture
    - For each input/output pair, internal delay (output load independent)
    - output dependent delay



- Standalone tools (PrimeTime) and part of logic synthesis.
- Back-annotation takes information from results of place and route to improve accuracy of timing analysis.
- DC in "topographical mode" uses preliminary layout information to model interconnect parasitics.
  - Prior versions used a simple fan-out model of gate loading.

Conclusion

- Timing Optimization: You start with a target on clock period. What control do you have?
- Biggest effect is RTL manipulation.
  - i.e., how much logic to put in each pipeline stage.
  - We will be talking later about how to manipulate RTL for better timing results.
- In most cases, the tools will do a good job at logic/circuit level:
  - Logic level manipulation
  - Transistor sizing
  - Buffer insertion
  - But some cases may be difficult and you may need to help
- The tools will need some help at the floorpan and layout