CS250
VLSI Systems Design

Lecture 4: Physical Realities: Beneath the Digital Abstraction, Part 1: Timing

Spring 2016

John Wawrzynek
with
Chris Yarp (GSI)

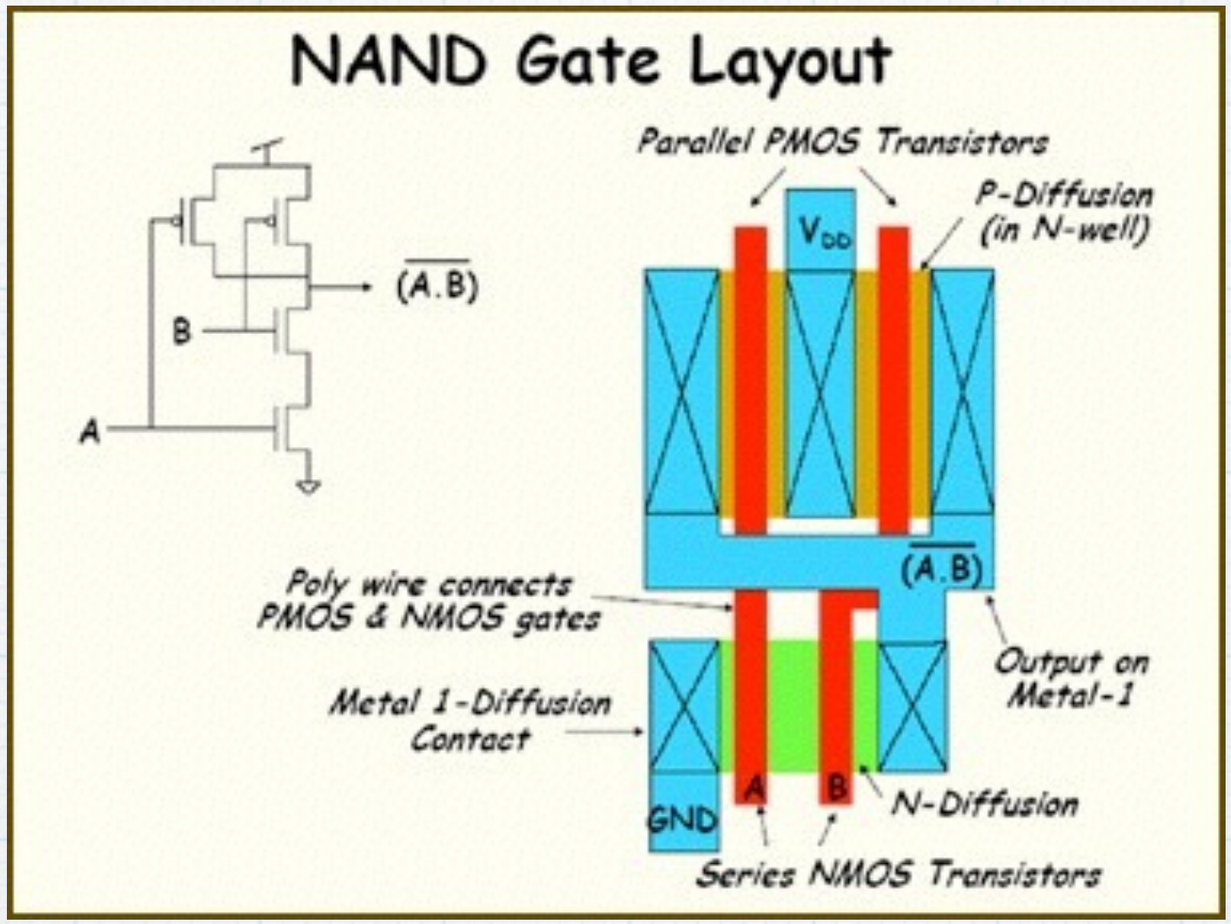
What do Computer Architects need to know about physics?

- Physics effect:

Area ⇒ cost
Delay ⇒ performance
Energy ⇒ performance & cost

- Ideally, zero delay, area, and energy. However, the physical devices occupy area, take time, and consume energy.
- CMOS process lets us build transistors, wires, connections, and we get capacitors, inductors, and resistors whether or not we want them.

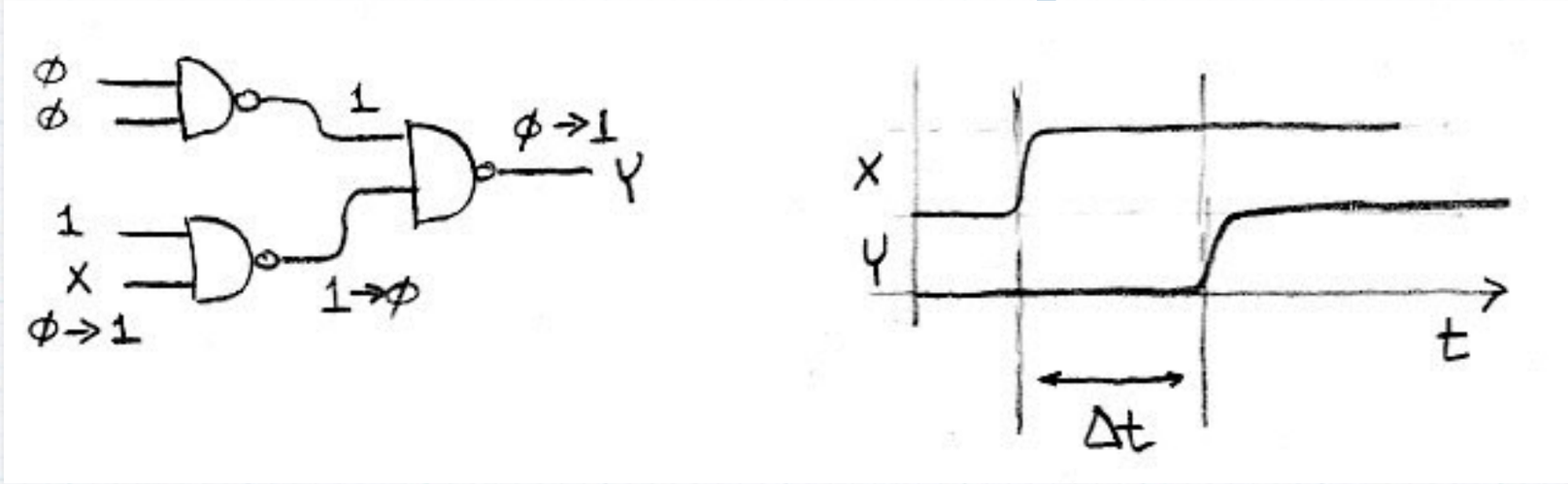
Physical Layout



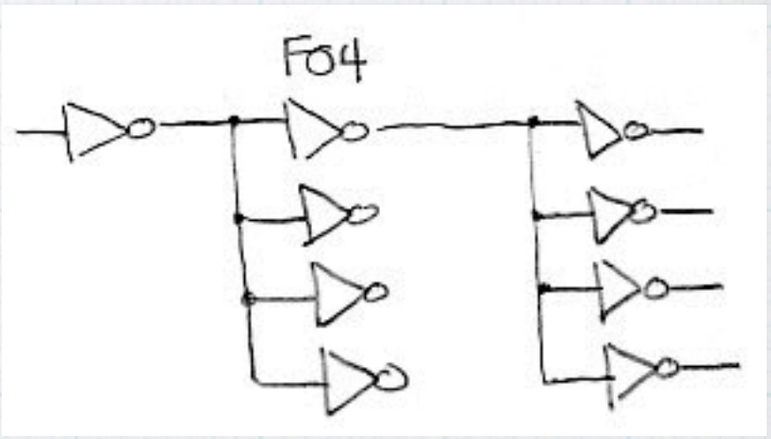
- "Switch-level" abstraction gives a good way to understand the **function** of a circuit.
  - nFET (g=1 ? short circuit : open)
  - pFET (g=0 ? short circuit : open)
- Understanding delay means going below the switch-level abstraction to transistor physics and layout details.

"Models should be as simple as possible,
but no simpler ..." Albert Einstein.

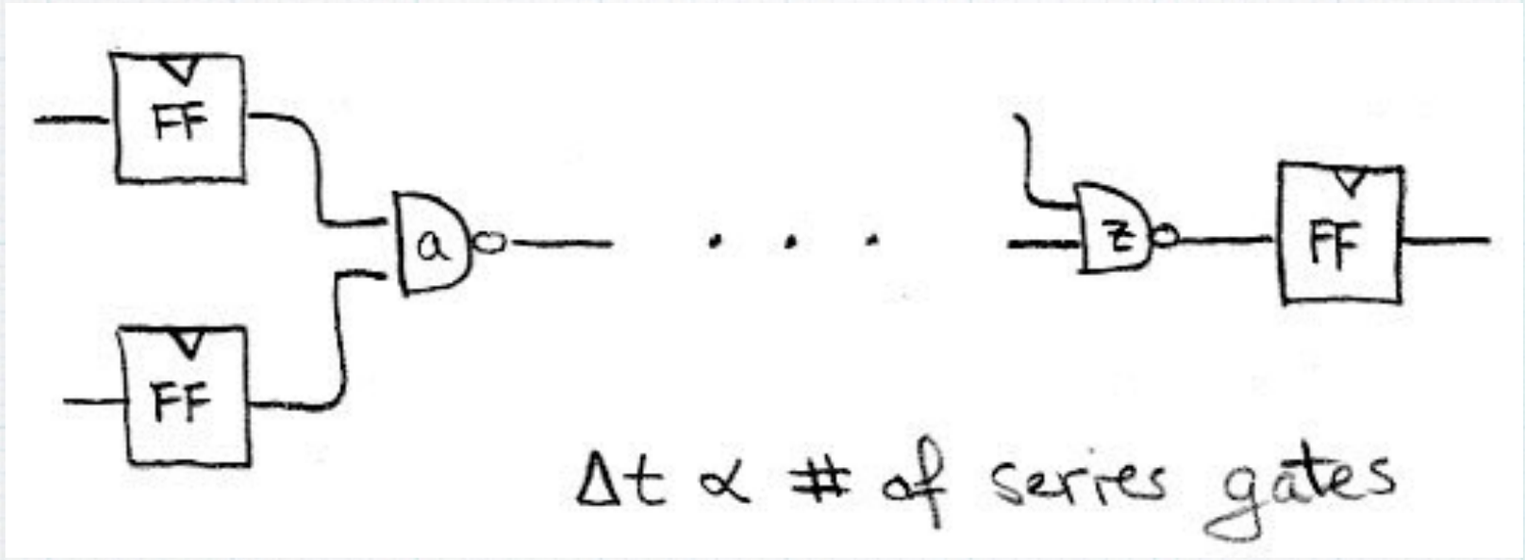
"Gate Delay"



- Modern CMOS gate delays on the order of a few picoseconds. (However, highly dependent on gate context.)
- Often expressed as FO4 delays (fan-out of 4) - as a process independent delay metric:
  - the delay of an inverter, driven by an inverter 4x smaller than itself, and driving an inverter 4x larger than itself.
  - For a 90nm process FO4 is around 20ps. Should be less than 10ps for our 32nm process.



"Path Delay"



- For correct operation:

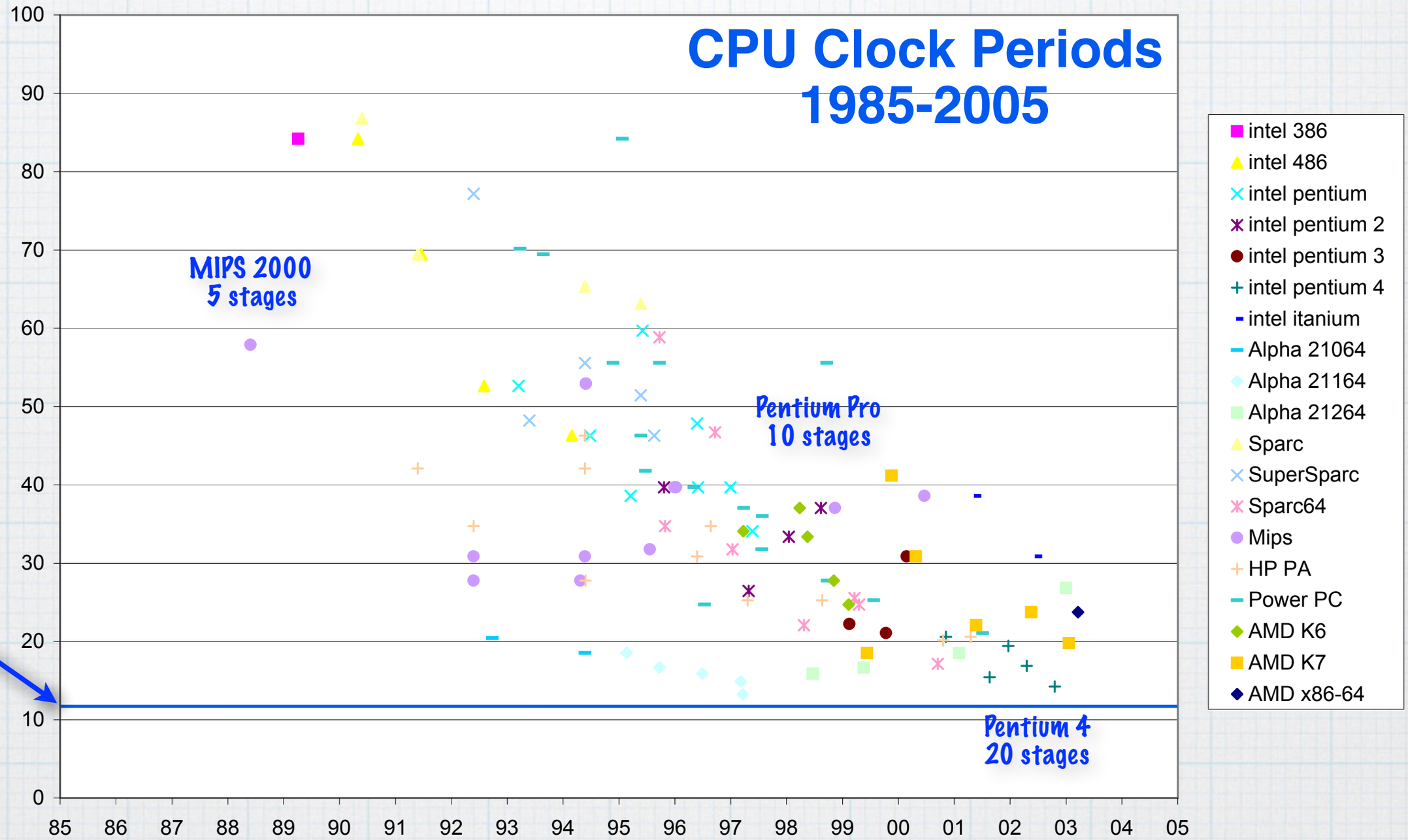
  Total Delay ≤ clock_period - FF_{setup_time} - $FF_{clk\_to\_q}$ - Clock_skew

  on all paths.

- High-speed processors critical paths have around 20 FO4 delays.

FO4 Delays per clock period

FO4 Delays



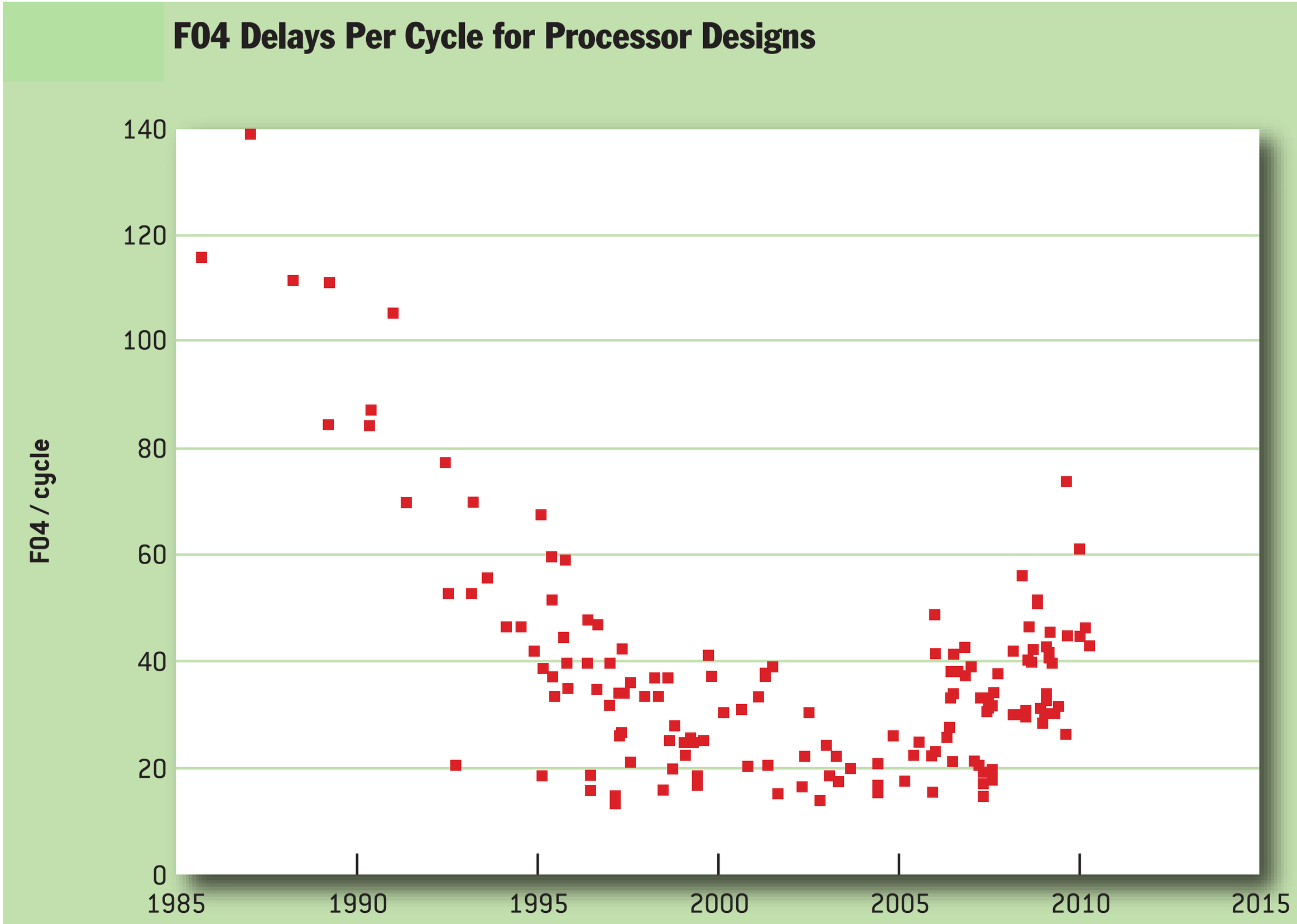
Thanks to Francois Labonte, Stanford

acmqueue

CPU DB: Recording Microprocessor History

With this open database, you can mine microprocessor trends over the past 40 years.

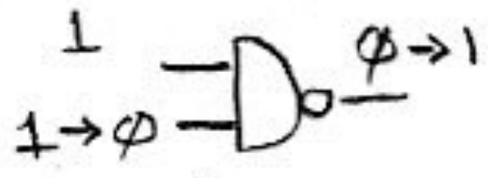
Andrew Danowitz, Kyle Kelley, James Mao, John P. Stevenson, Mark Horowitz, Stanford University

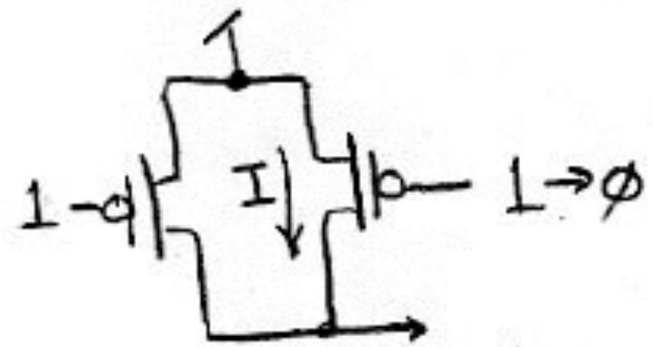


FO4 delay per cycle is roughly proportional to the amount of computation completed per cycle.

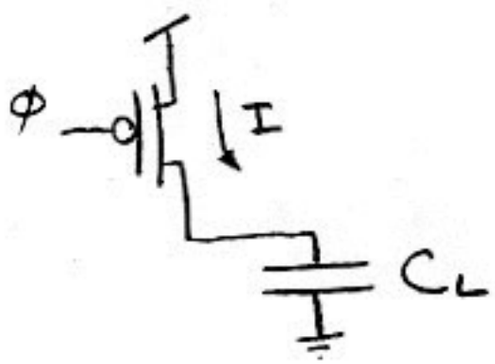
"Gate Delay"

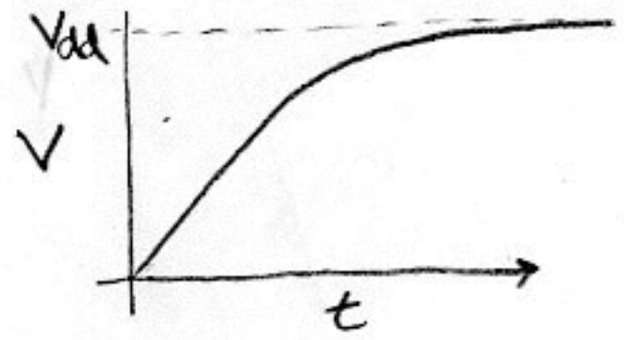
- What determines the actual delay of a logic gate?
- Transistors are not perfect switches - cannot change terminal voltages instantaneously.
- Consider the NAND gate:





  - Current (I) value depends on: process parameters, transistor size



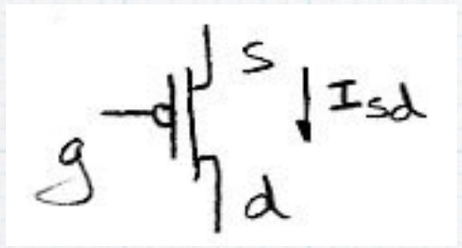


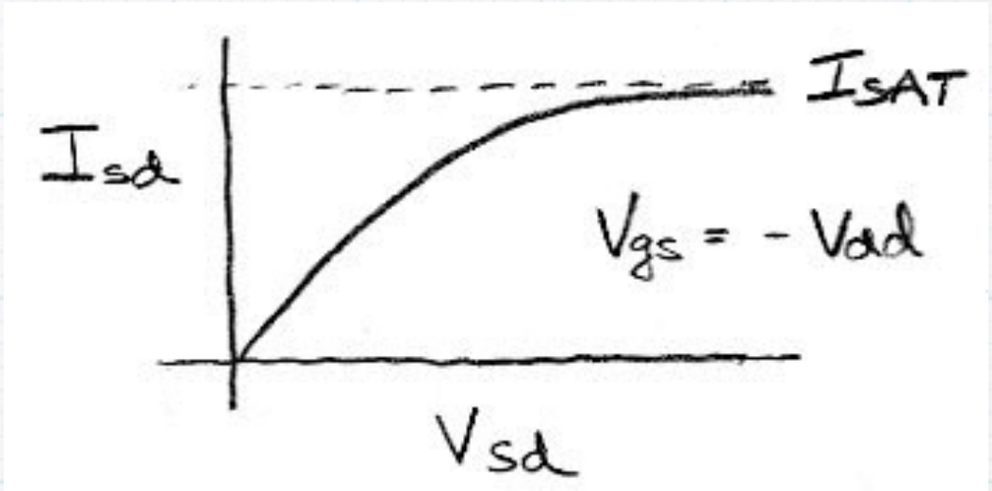
$\Delta \propto C_L / I$

- C_L models gate output, wire, inputs to next stage (Cap. of Load)
- C "integrates" I creating a voltage change at output

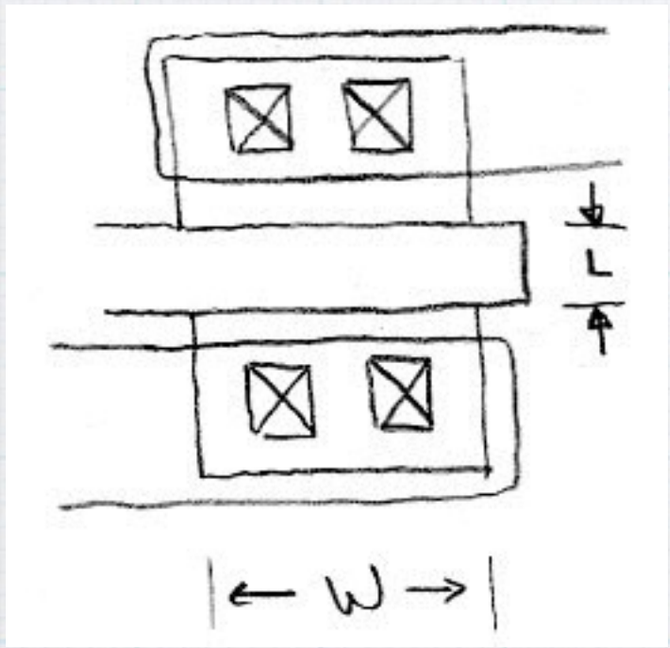
More on transistor Current

- Transistors act like a cross between a resistor and "current source"





- I_{SAT} depends on process parameters (higher for nFETs than for pFETs) and transistor size (layout):

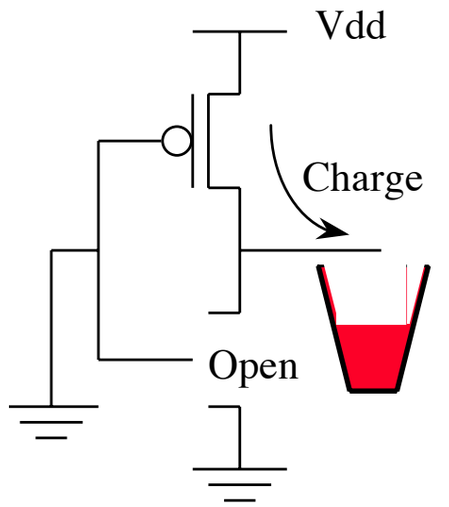


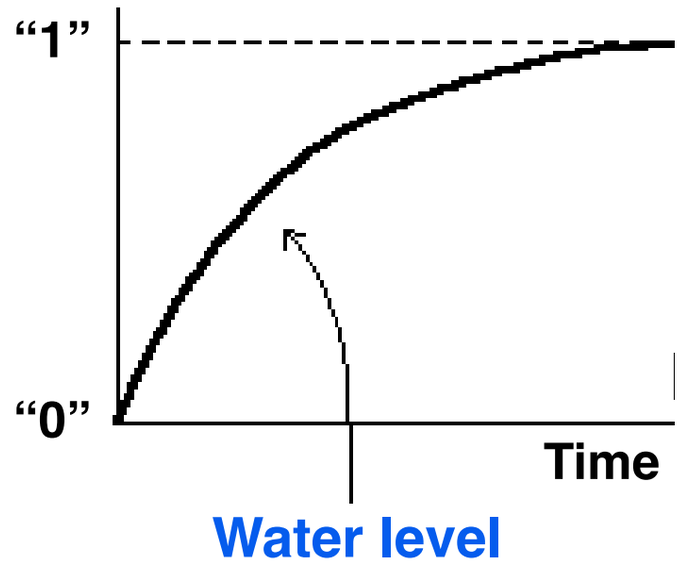
$$I_{SAT} \propto W/L$$

Transistors as water valves. (Cartoon physics)

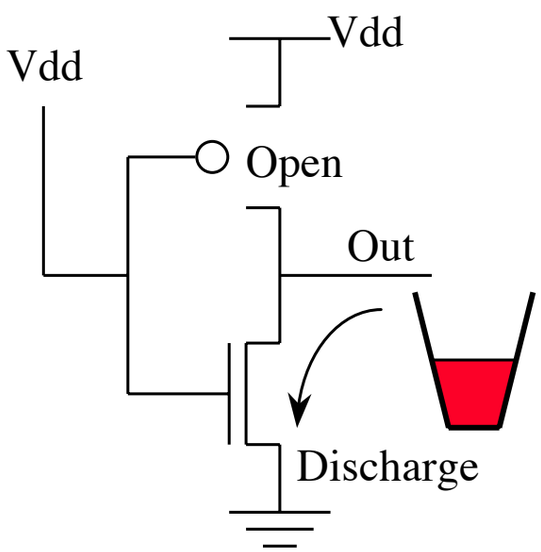
If electrons are water molecules,
transistor strengths (W/L) are pipe diameters,
and capacitors are buckets ...

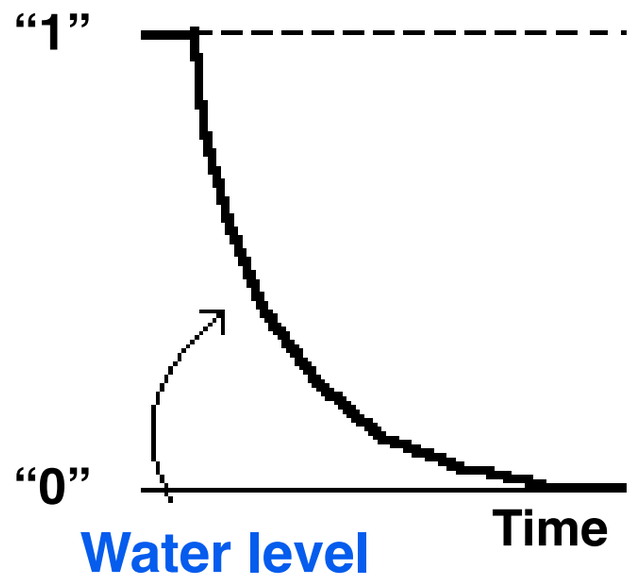
A "on" p-FET fills up the capacitor with charge.





A "on" n-FET empties the bucket.

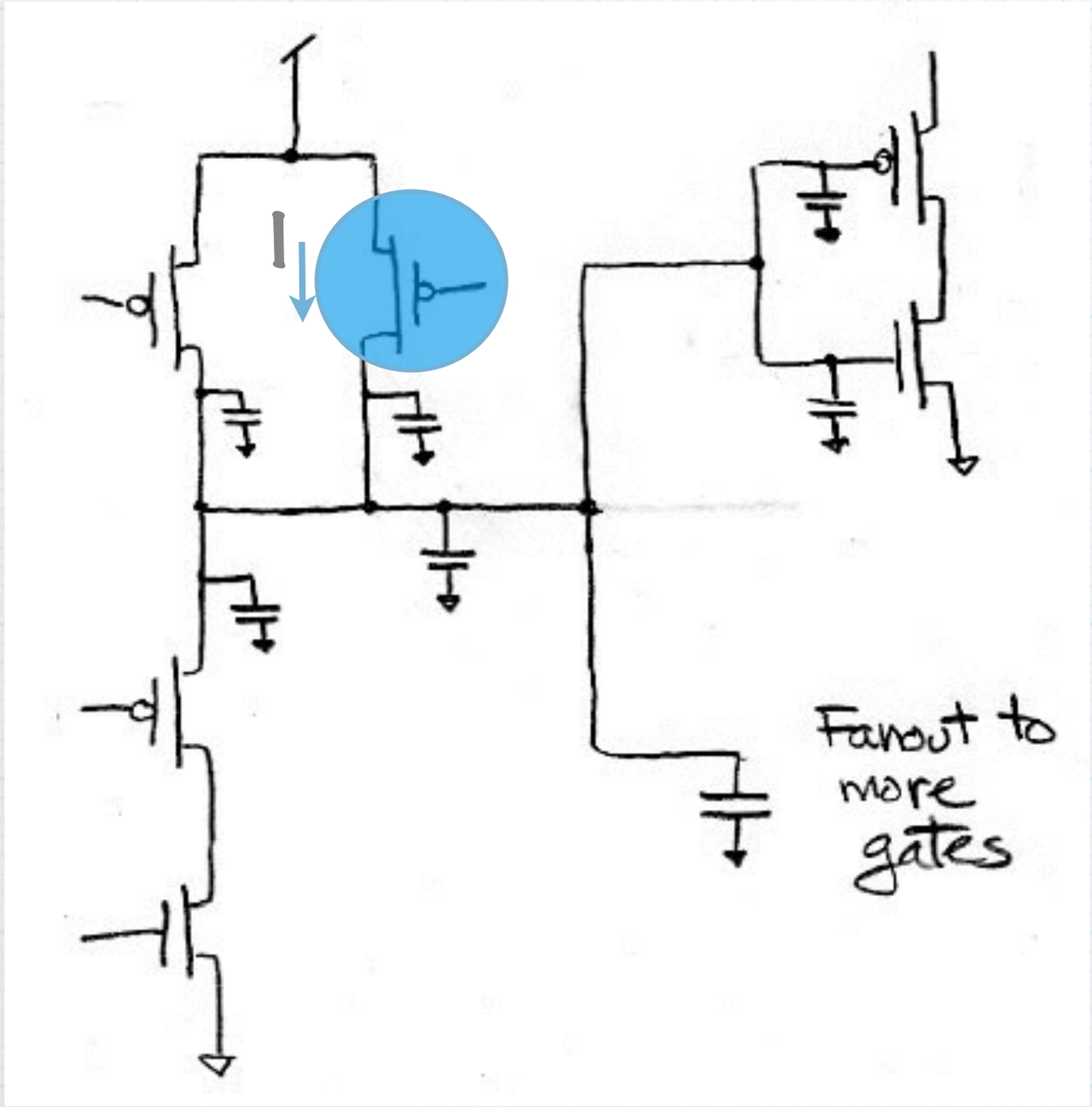




This model is often good enough ...

More on C_L

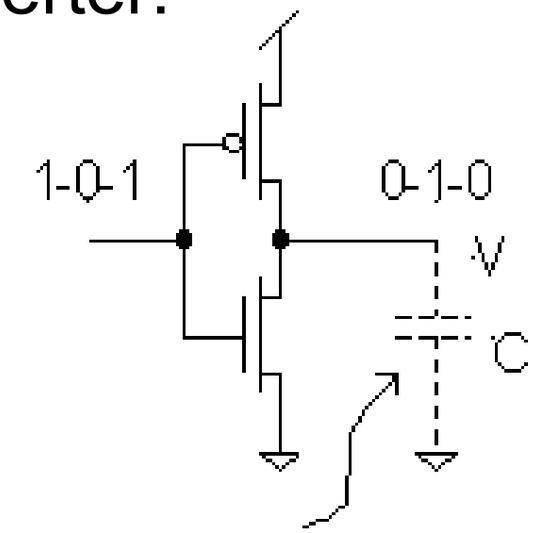
- Everything that connects to the output of a logic gate (or transistor) contributes capacitance:



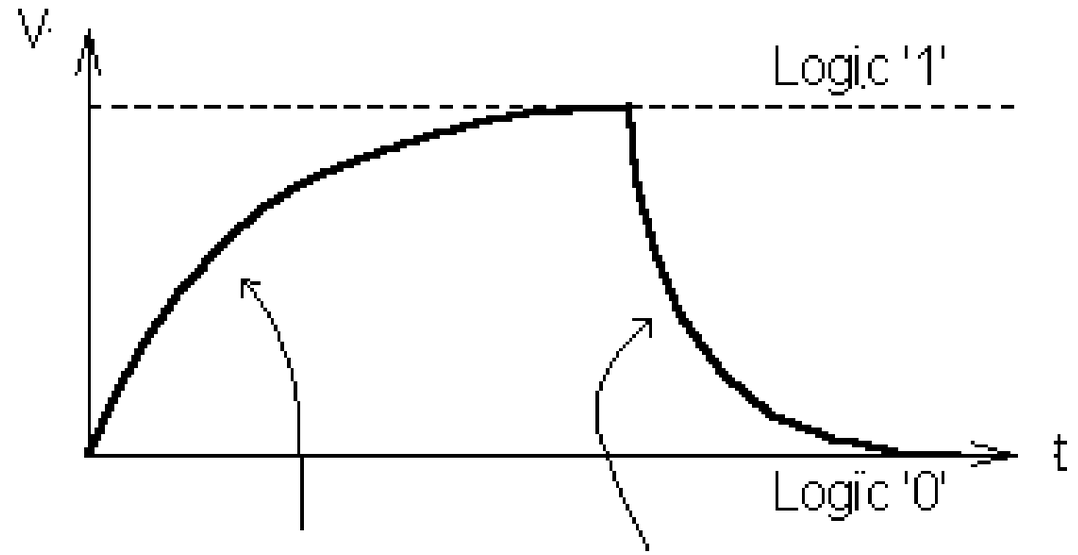
- Transistor drains
- Interconnection (wires/ contacts/vias)
- Transistor Gates

What is the bucket? A gate's "fan-out".

Inverter:



Models inputs to other gates & wire capacitance



rate dependent on pullup strength & C

rate dependent on pulldown strength & C

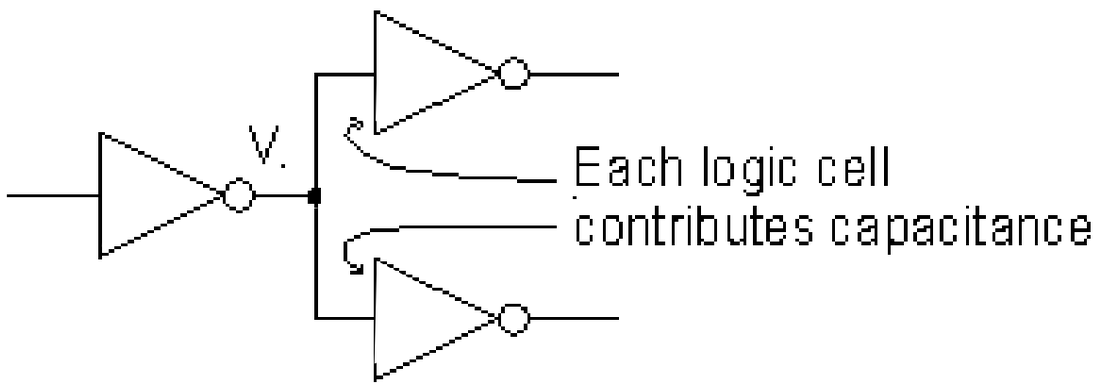
"Fan-out": The number of gate inputs driven by a gate's output.

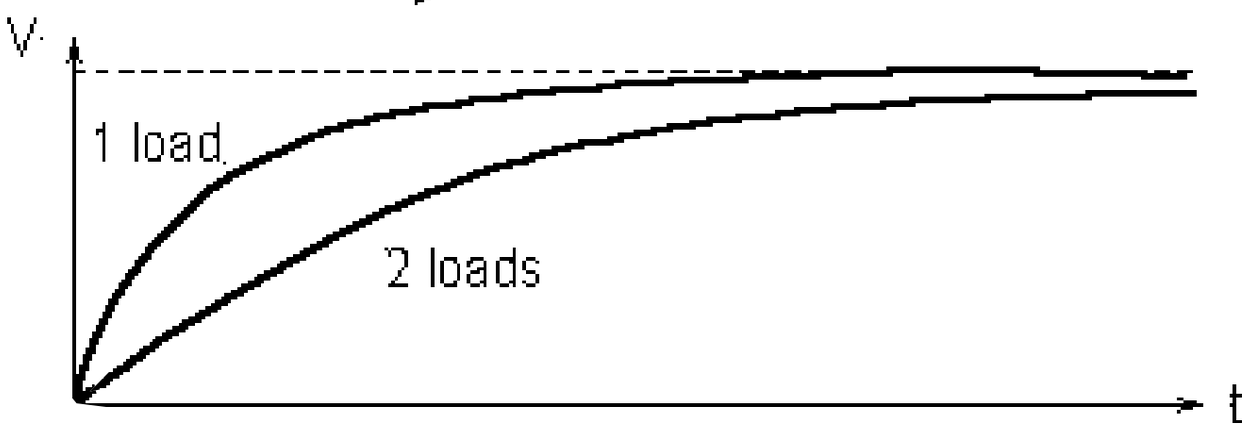
Driving other gates slows a gate down.

Driving wires slows a gate down.

Driving it's own parasitics slows a gate down.

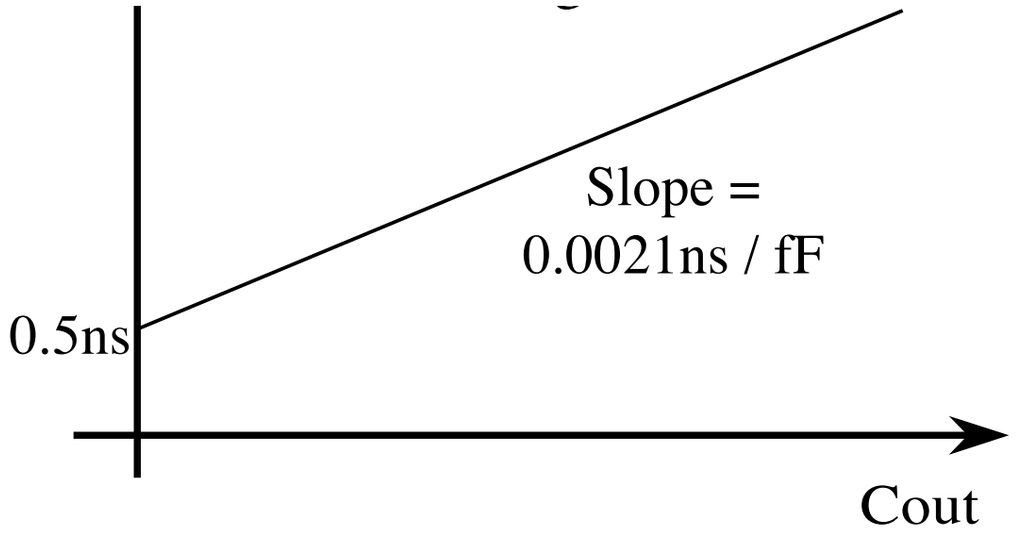
A closer look at fan-out ...



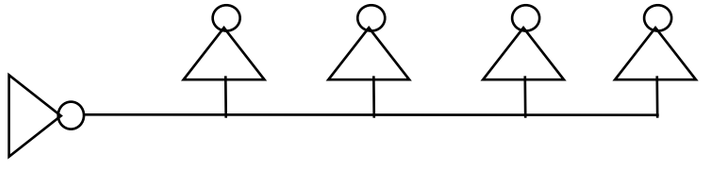


Driving more gates adds delay.

Linear model works for reasonable fan-out



FO4: Fanout of four delay.

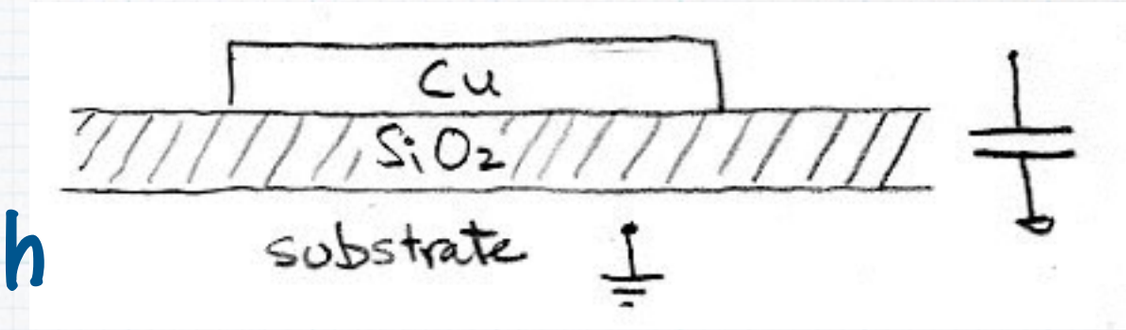


Delay time of an inverter driving 4 inverters.

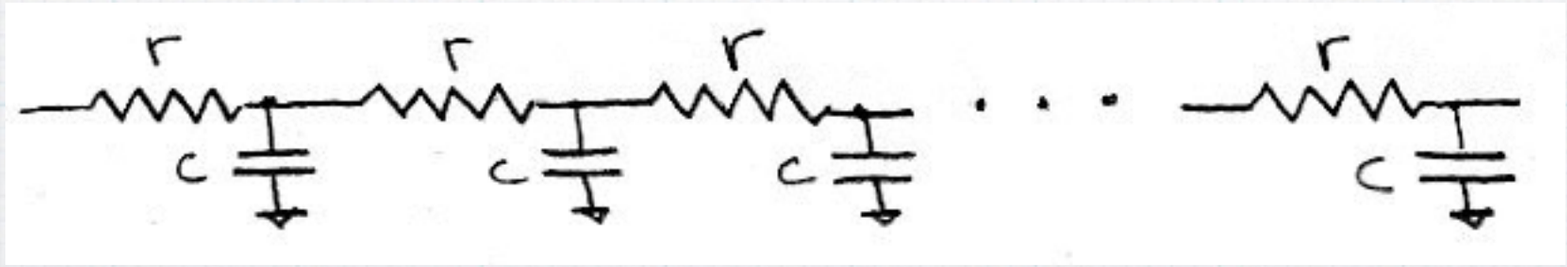
Wires

- So far, simple capacitors:

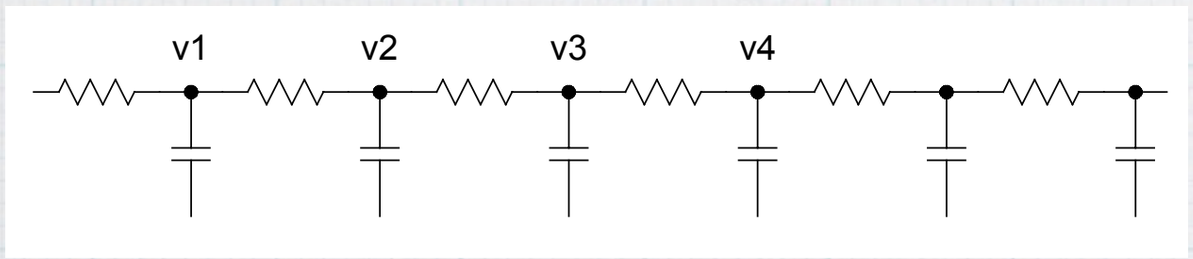
  $C \propto \text{Area} = \text{width} * \text{length}$

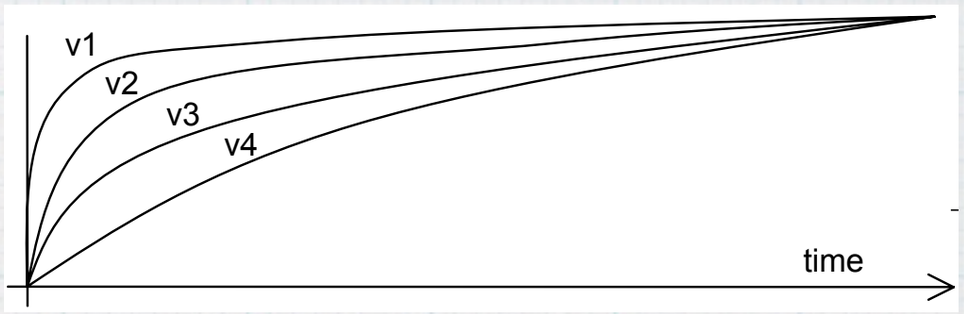


- Wires have finite resistance, so have distributed R and C:



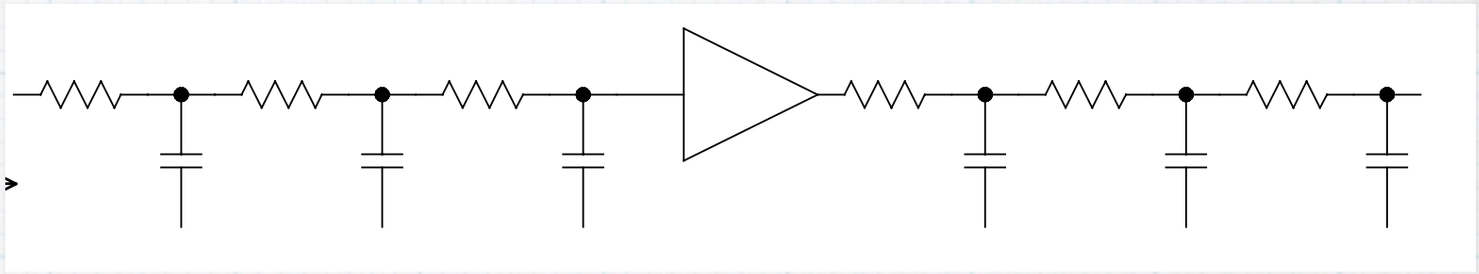
with r = res/length, c = cap/length, $\Delta \propto rcL^2 \cong rc + 2rc + 3rc + \ldots$

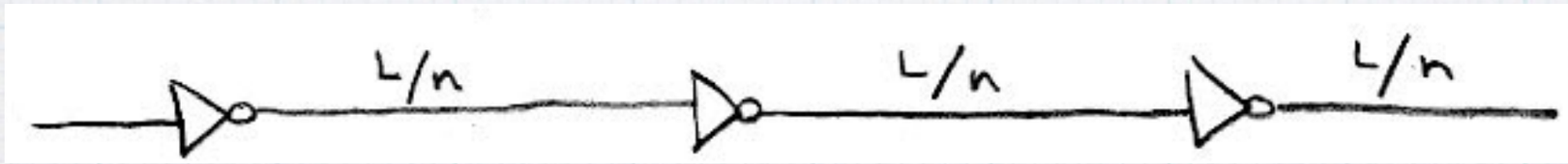




Wires

- For short wires (between gates) R is insignificant:
  (total wire RC delay << total gate delay)
- For long wires R becomes significant. Ex: busses, clocks, reset
- "rebuffering" helps

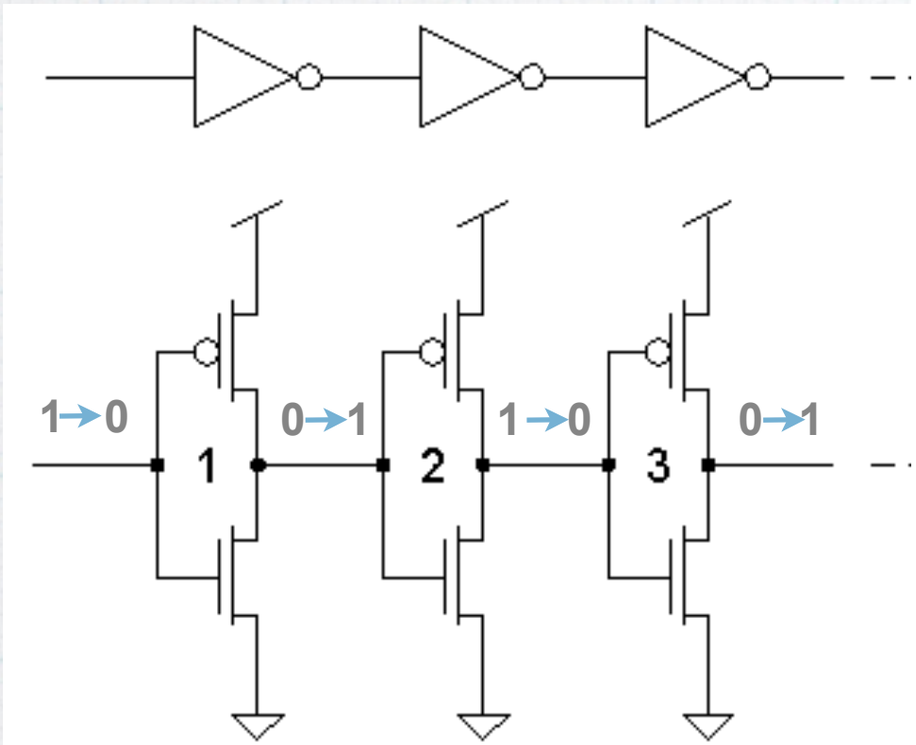


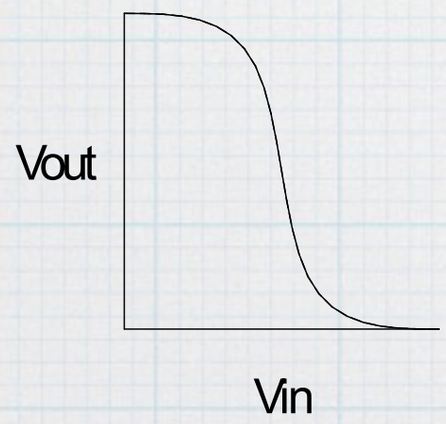


- Finding the correct number and spacing requires solving a quadratic optimization problem. Tradeoff fixed delay (overhead) of buffers with RC wire delay.

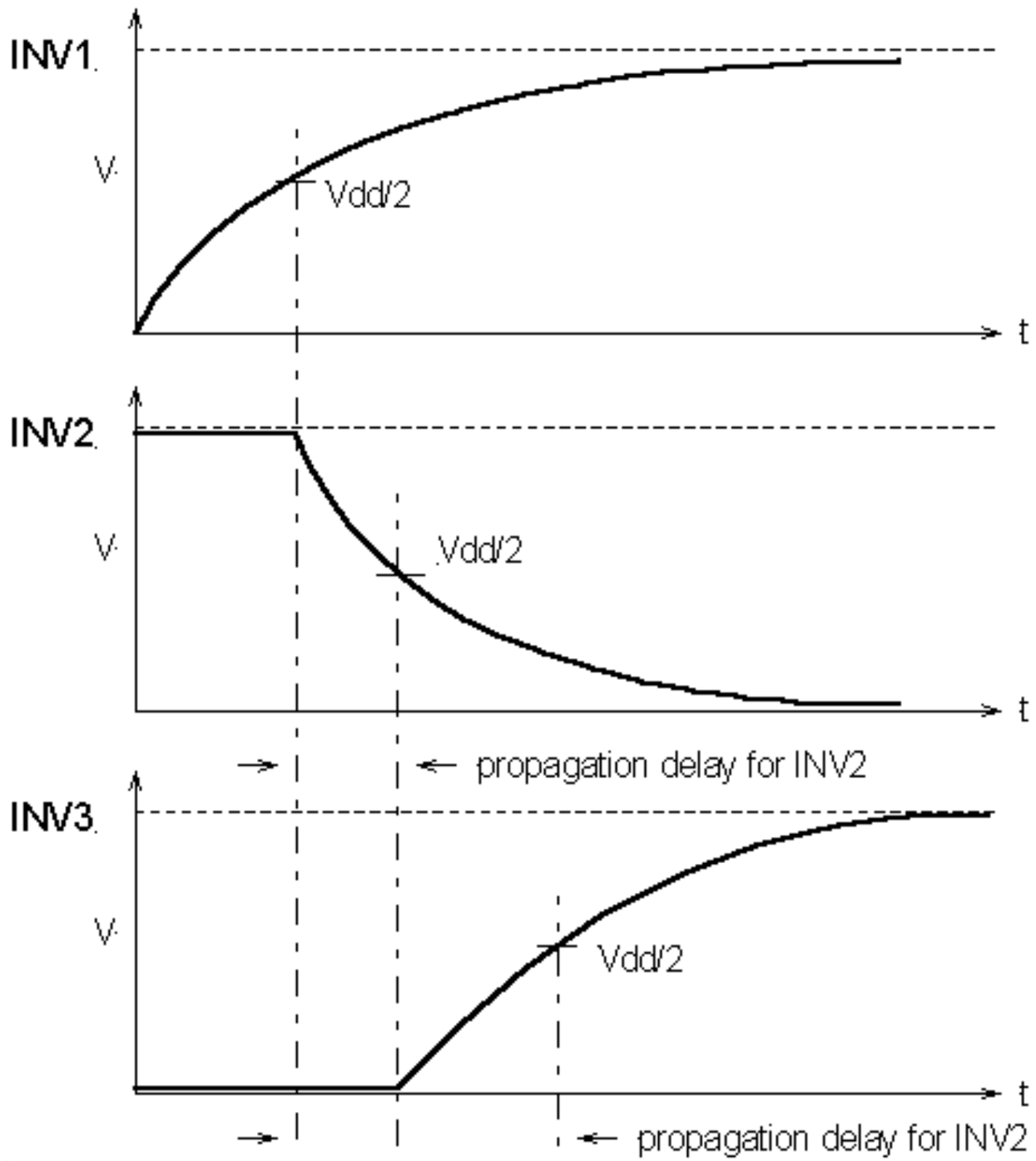
Turning Rise/Fall Delay into Gate Delay

- Cascaded gates:





"transfer curve" for inverter.

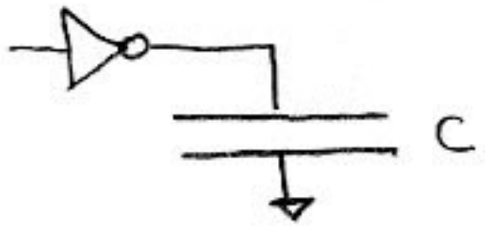


In general:

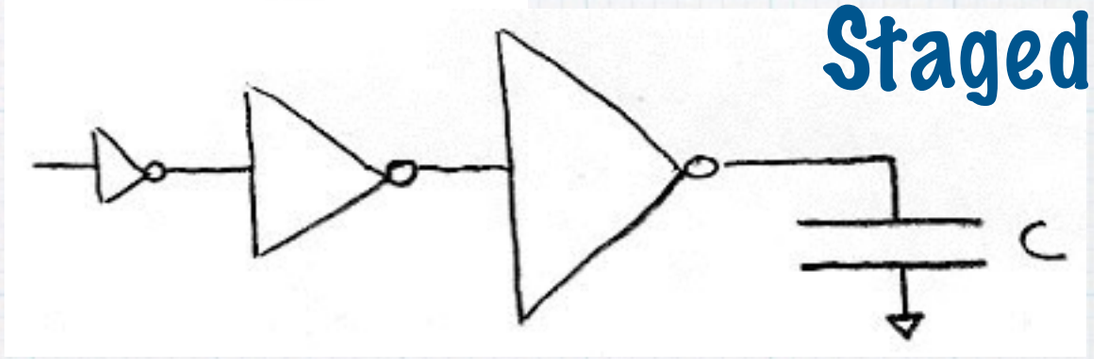
prop. delay = sum of individual prop. delays of gates in series.

Driving Large Loads

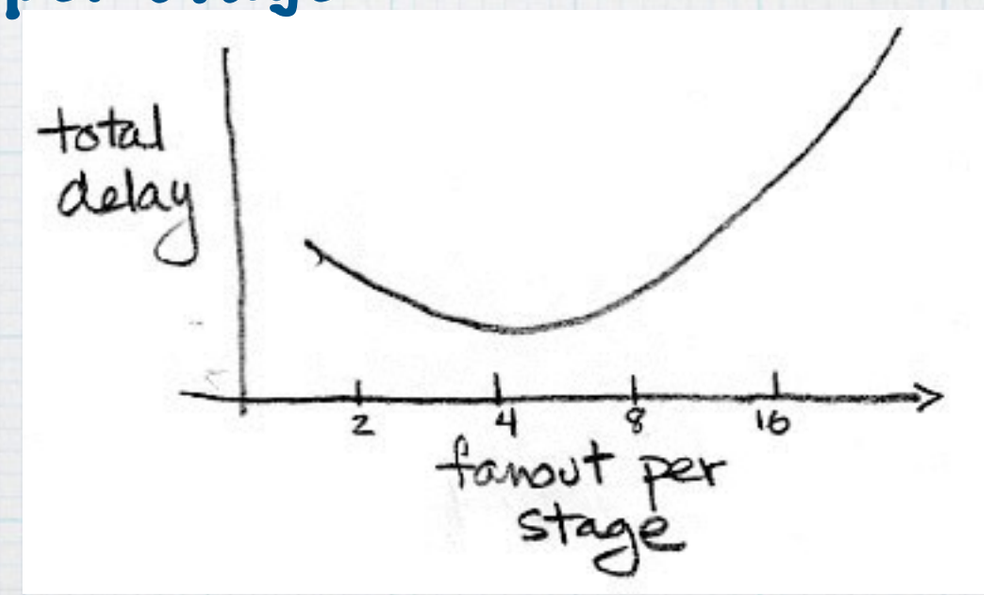
- Large fanout nets: clocks, resets, memory bit lines, off-chip
- Relatively small driver results in long rise time (and thus large gate delay)



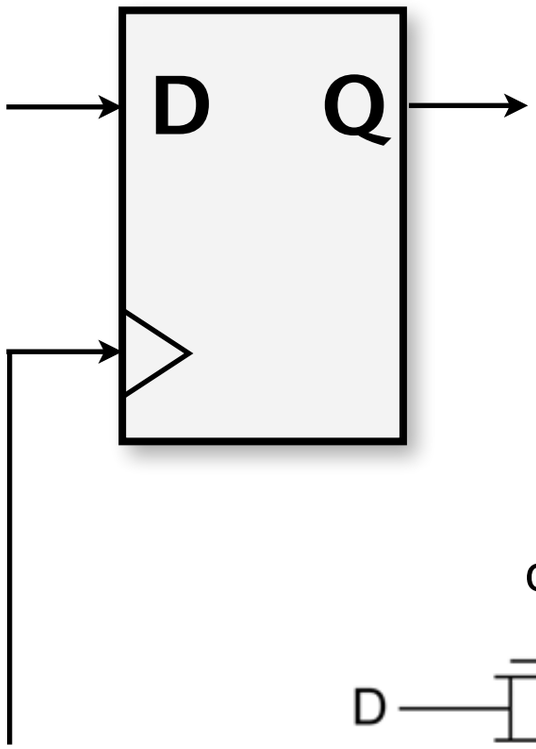
- Strategy: **Staged Buffers**



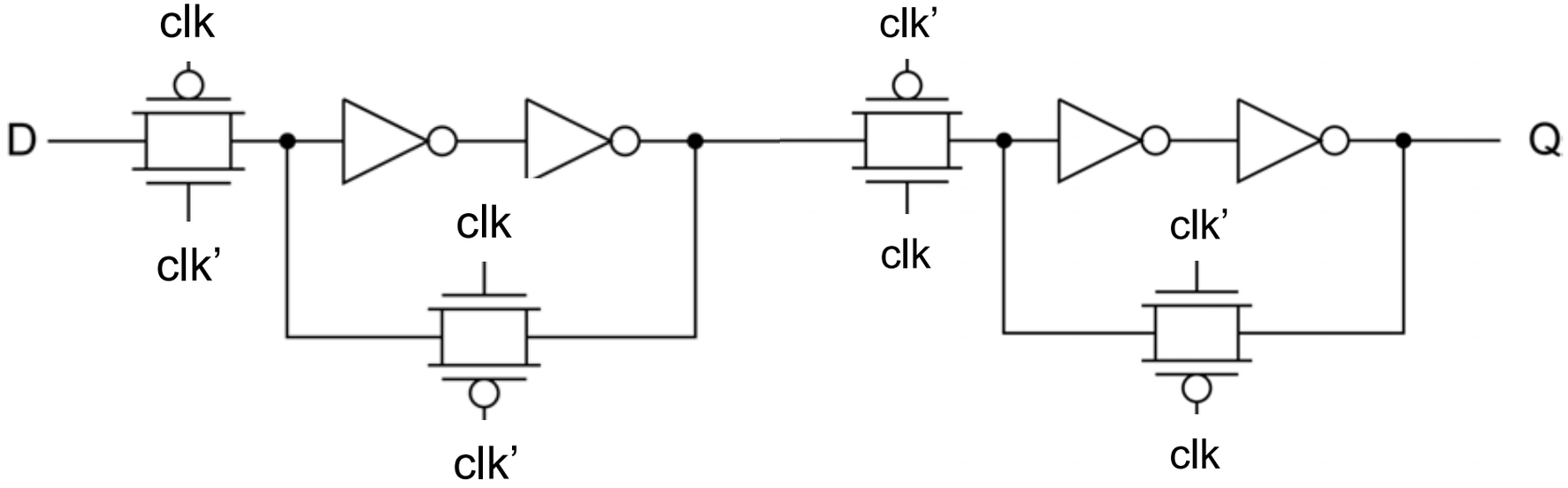
- Optimal trade-off between delay per stage and total number of stages ⇒ **fanout of ~4-6 per stage**



Recall: Positive edge-triggered flip-flop



A flip-flop "samples" right before the edge, and then "holds" value.

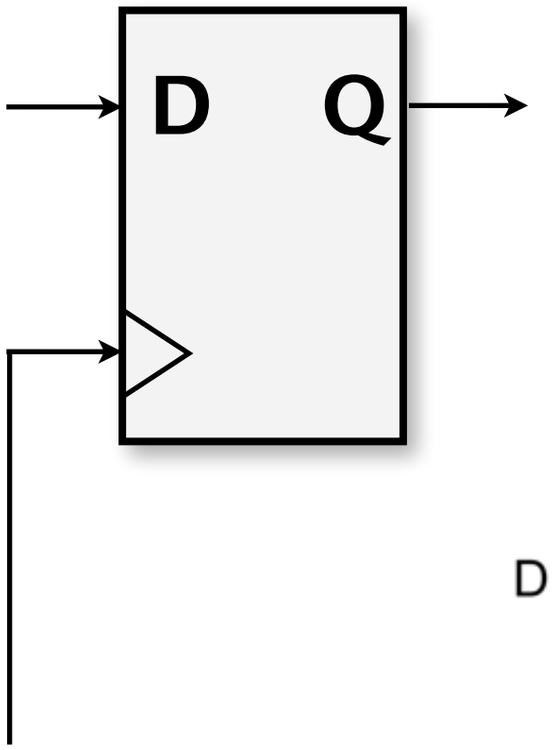


16 Transistors: Makes an SRAM look compact!

What do we get for the 10 extra transistors?

Clocked logic semantics.

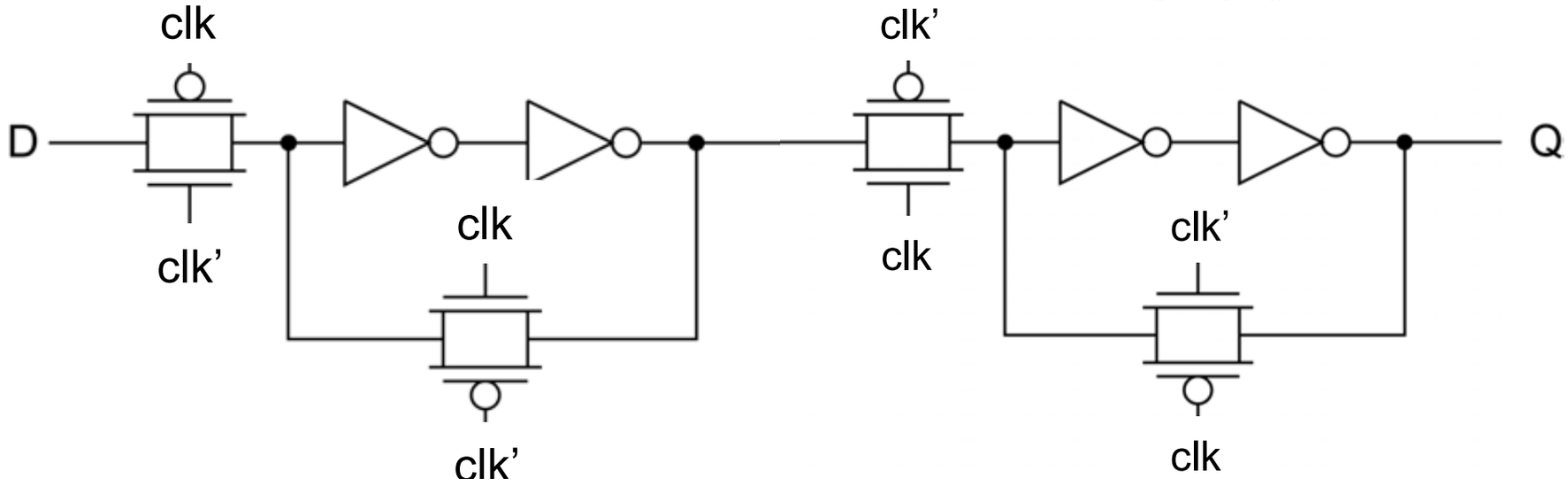
Sensing: When clock is low



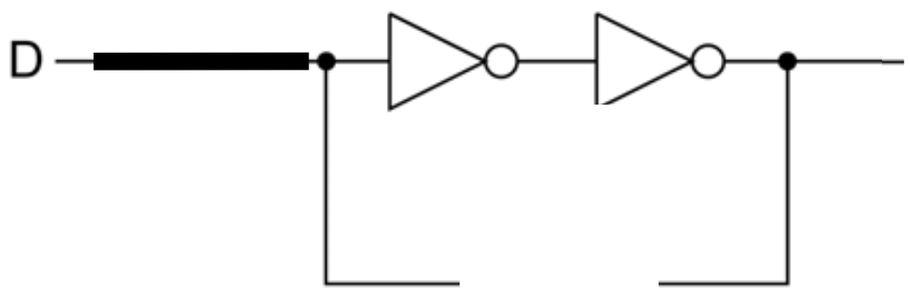
A flip-flop "samples" right before the edge, and then "holds" value.

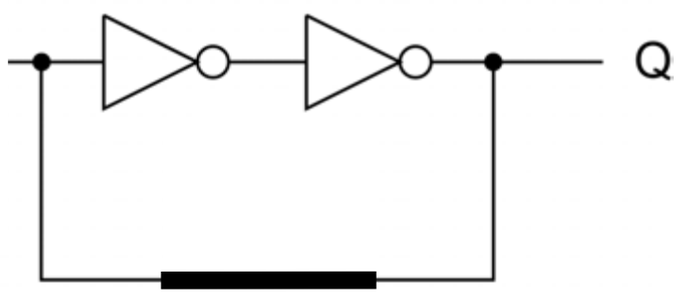
Sampling circuit

Holds value



clk = 0
clk' = 1

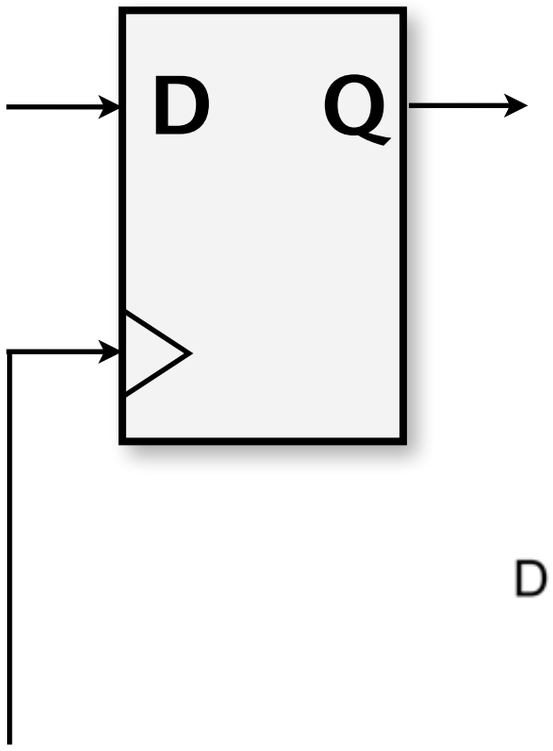




Will capture new value on posedge.

Outputs last value captured.

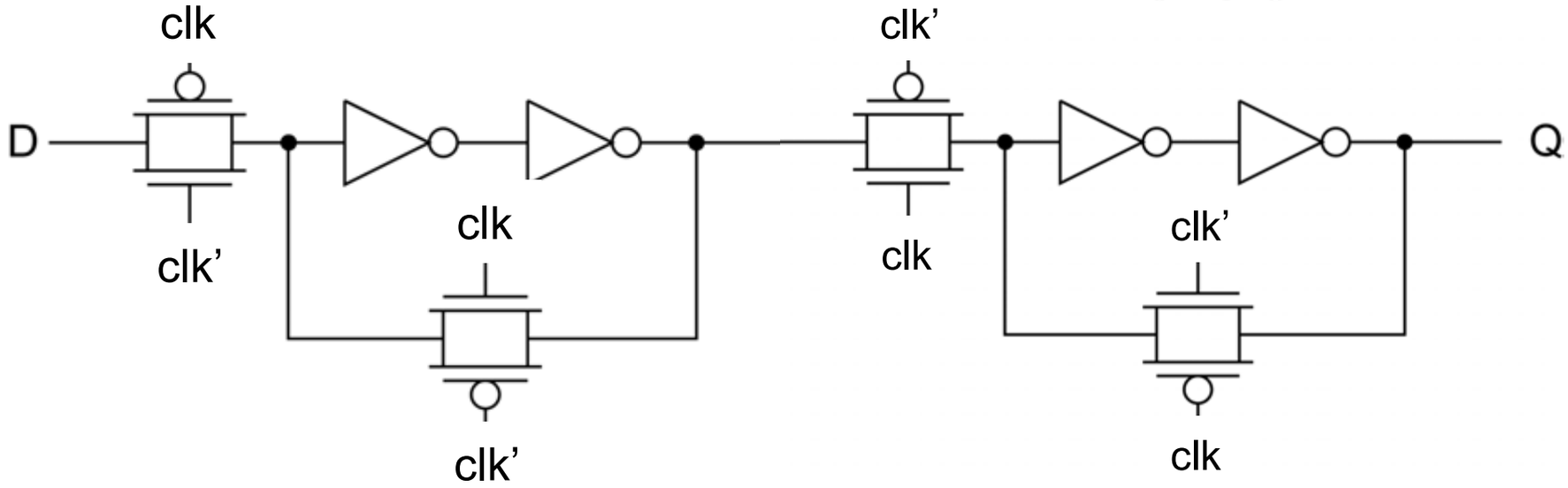
Capture: When clock goes high



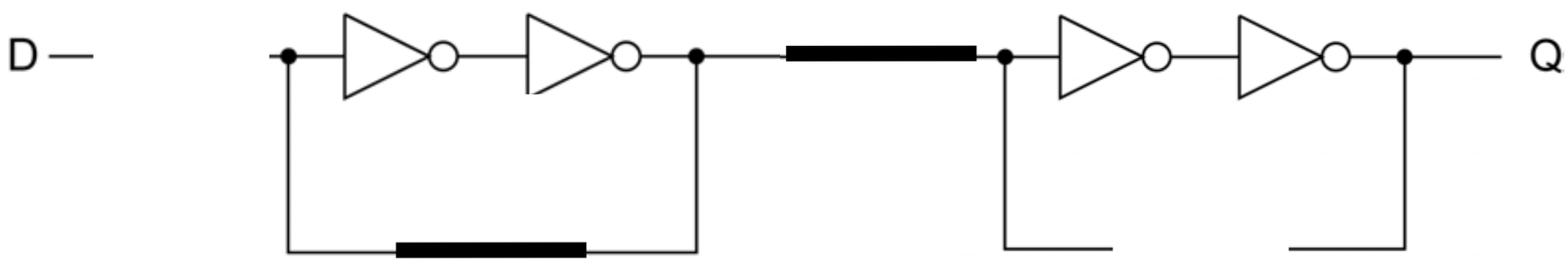
A flip-flop "samples" right before the edge, and then "holds" value.

Sampling circuit

Holds value



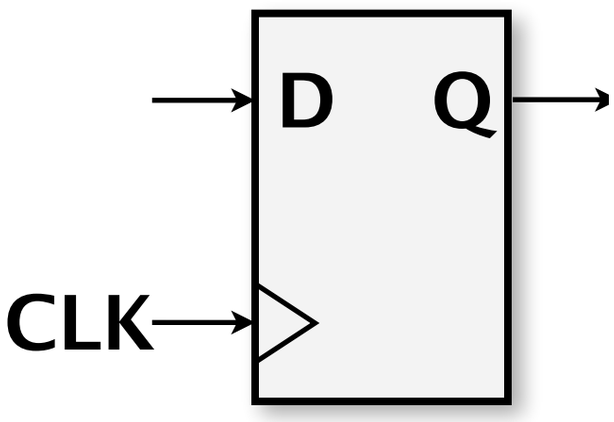
clk = 1
clk' = 0

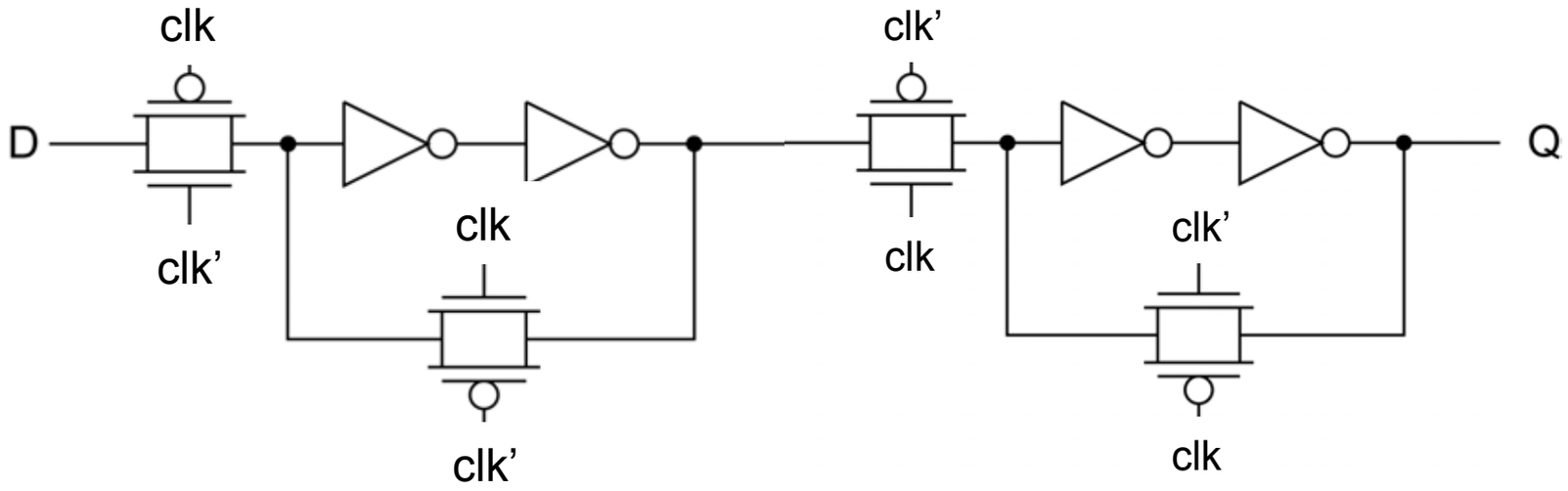


Remembers value just captured.

Outputs value just captured.

Flip Flop delays: clk-to-Q ? setup ? hold ?



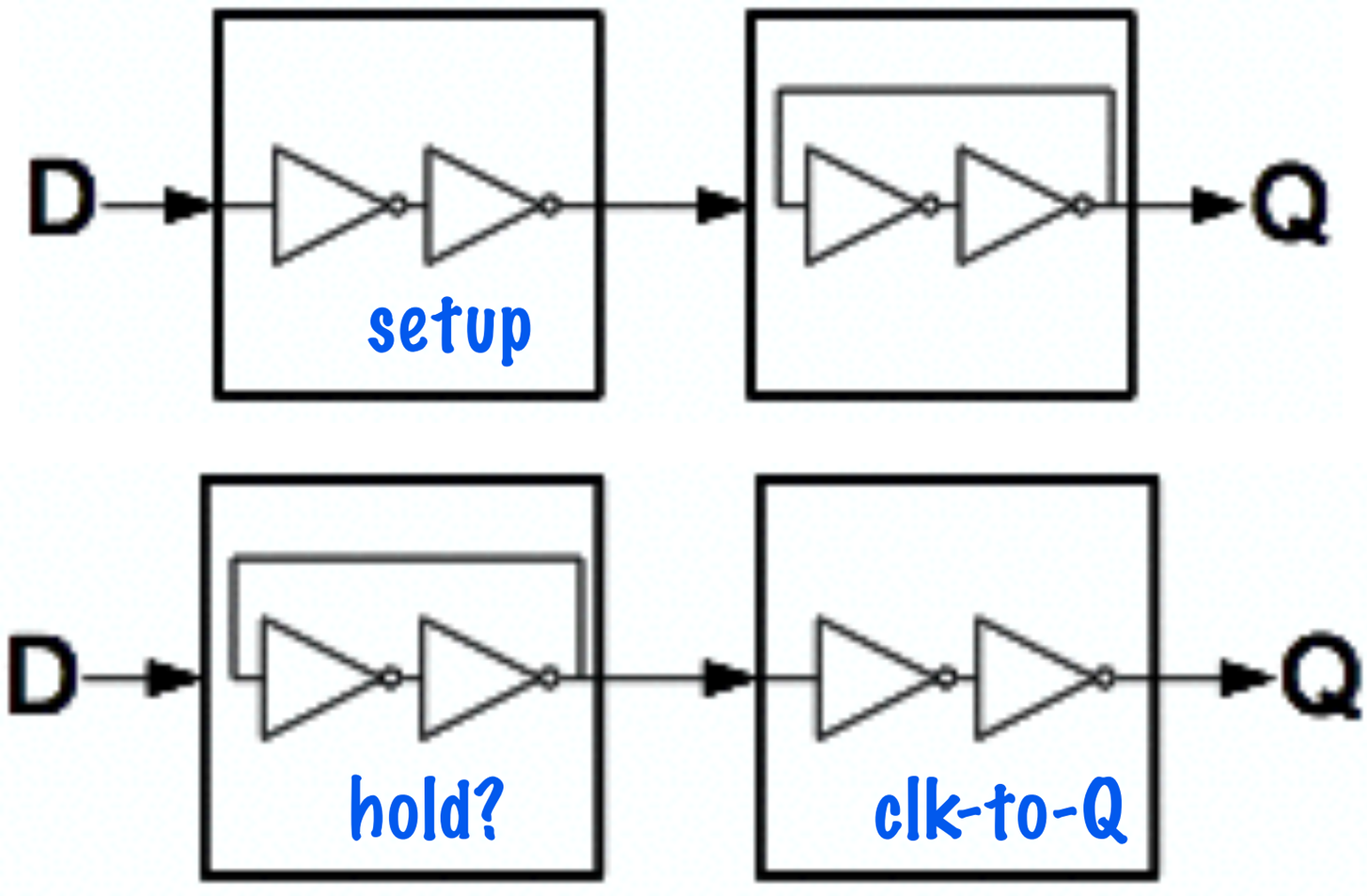


CLK == 0

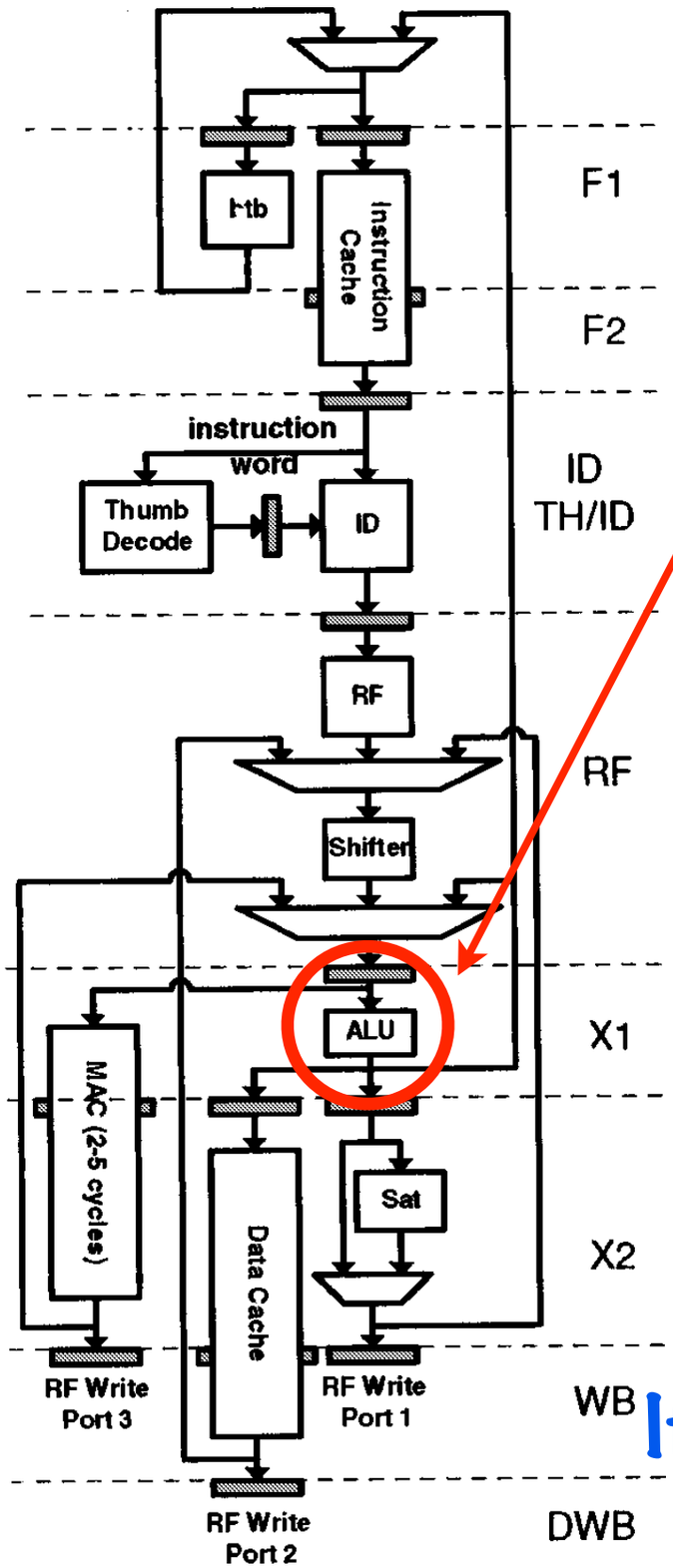
Sense D, but Q outputs old value.

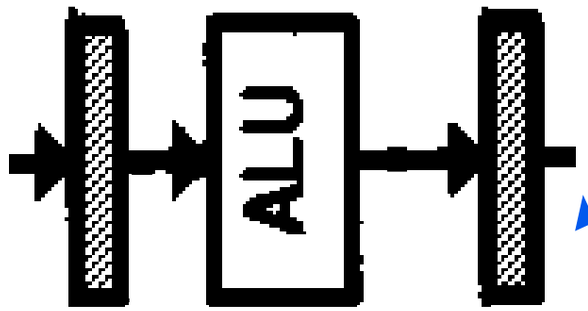
CLK 0->1

Capture D, pass value to Q



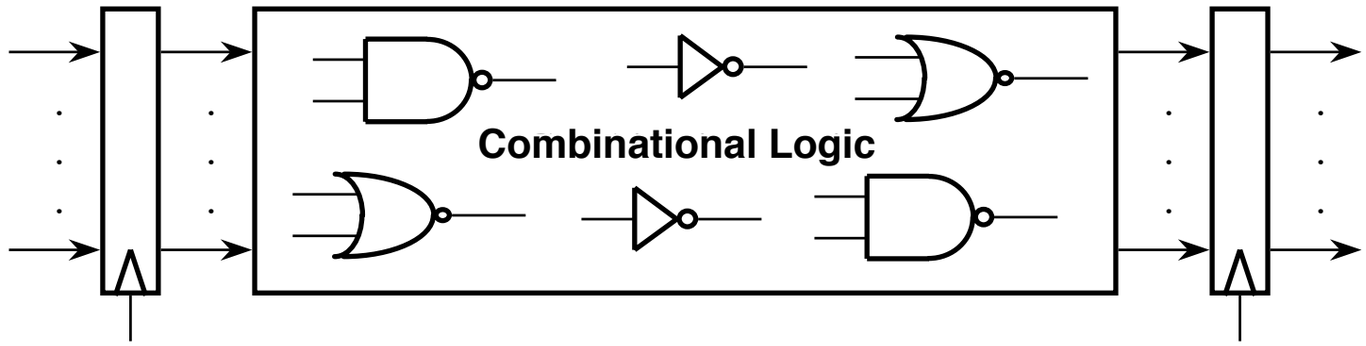
Timing Analysis and Logic Delay

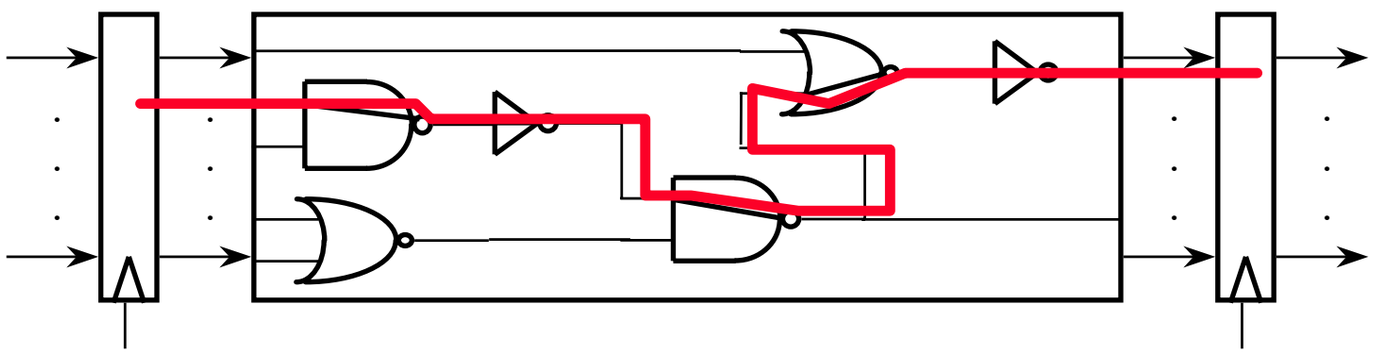




Register:

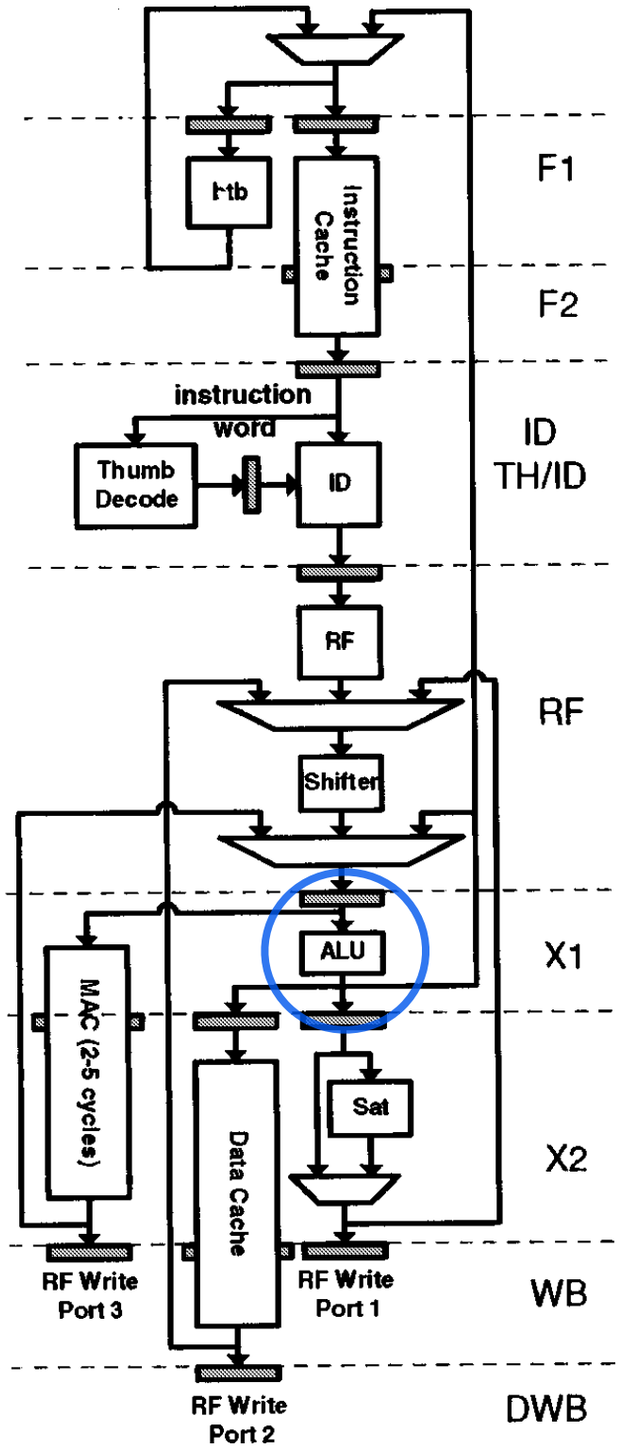
An Array of Flip-Flops

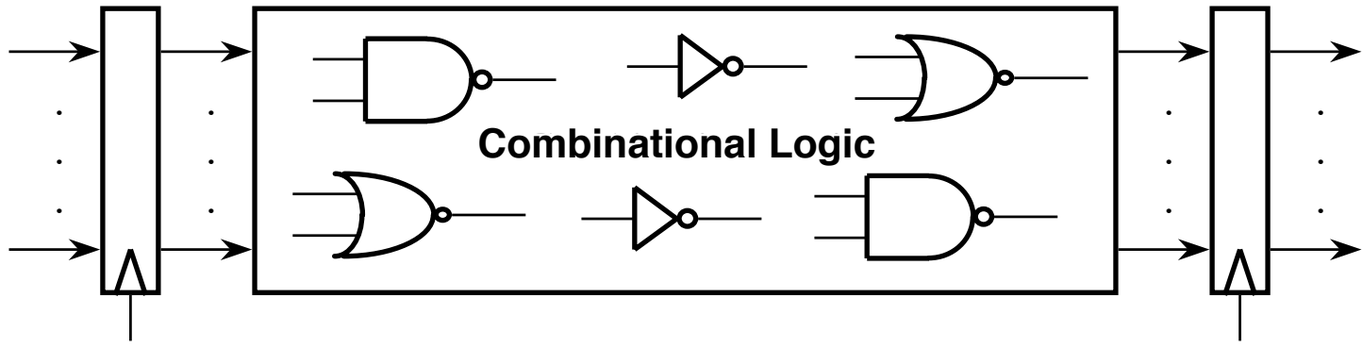


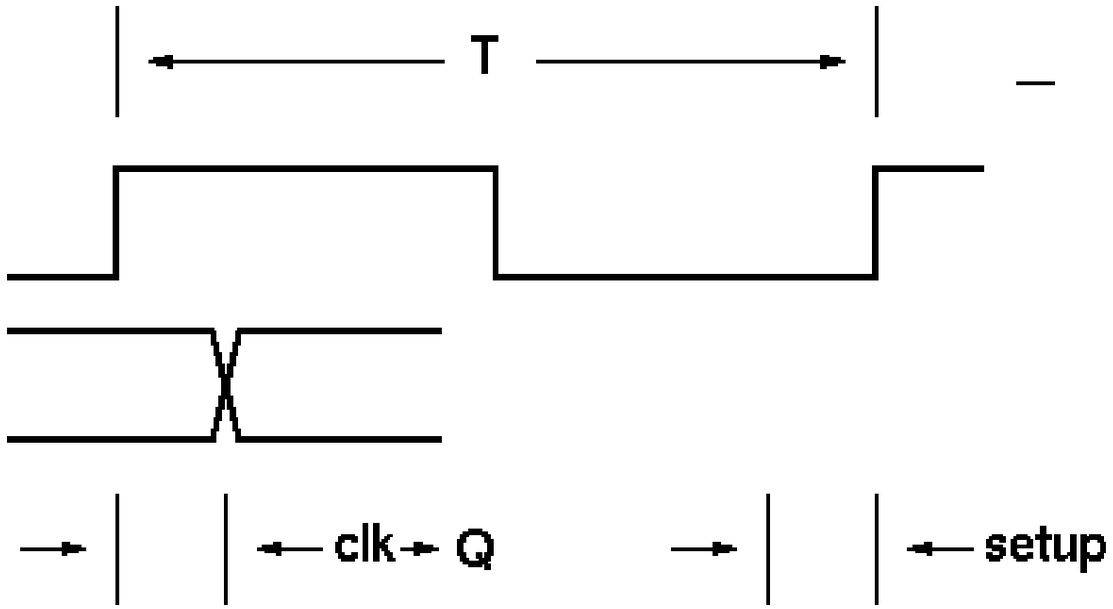


If our clock period T > worst-case delay through CL, does this ensure correct operation?

Flip-Flop delays eat into "time budget"



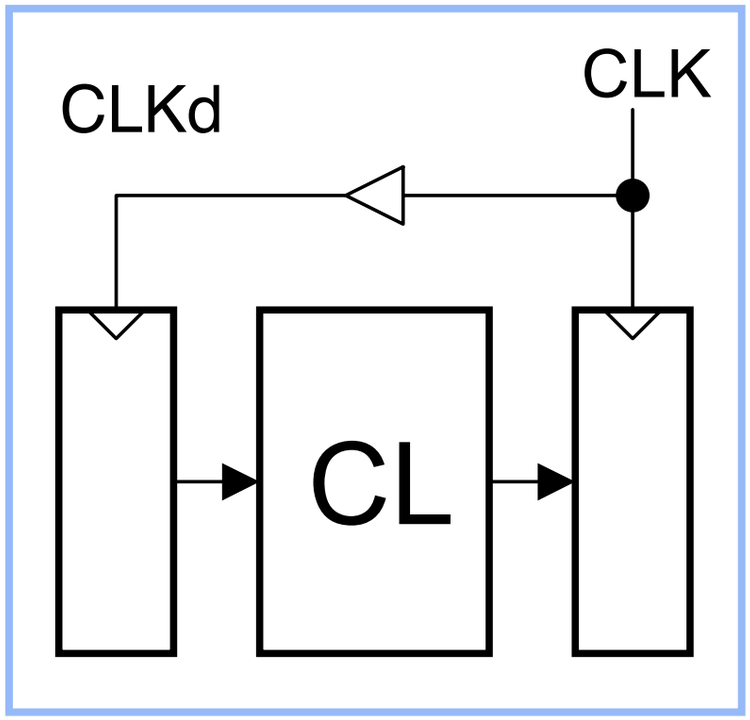




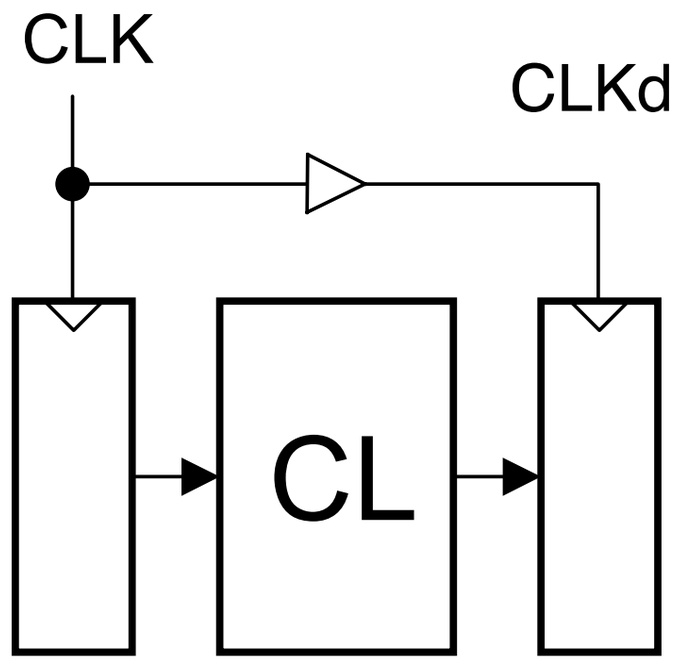
ALU "time budget"

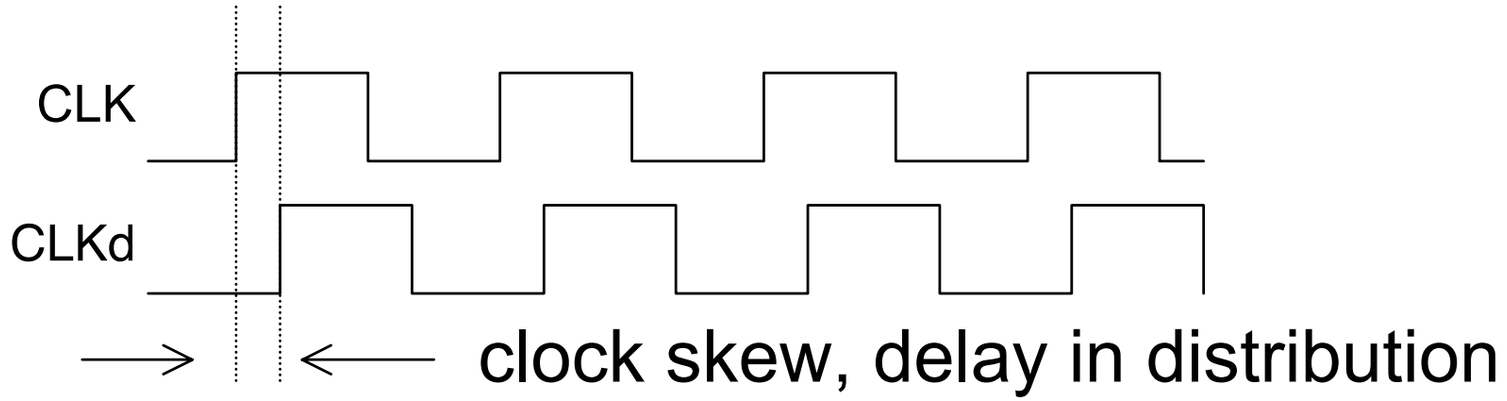
$$T \geq \tau_{clk \rightarrow Q} + \tau_{CL} + \tau_{setup}$$

Clock skew also eats into "time budget"



As T →0, which circuit fails first?





$$T \geq T_{CL} + T_{setup} + T_{clk \rightarrow Q} + \text{worst case skew.}$$

Grid

Tuned sector trees

Sector buffers

Delay

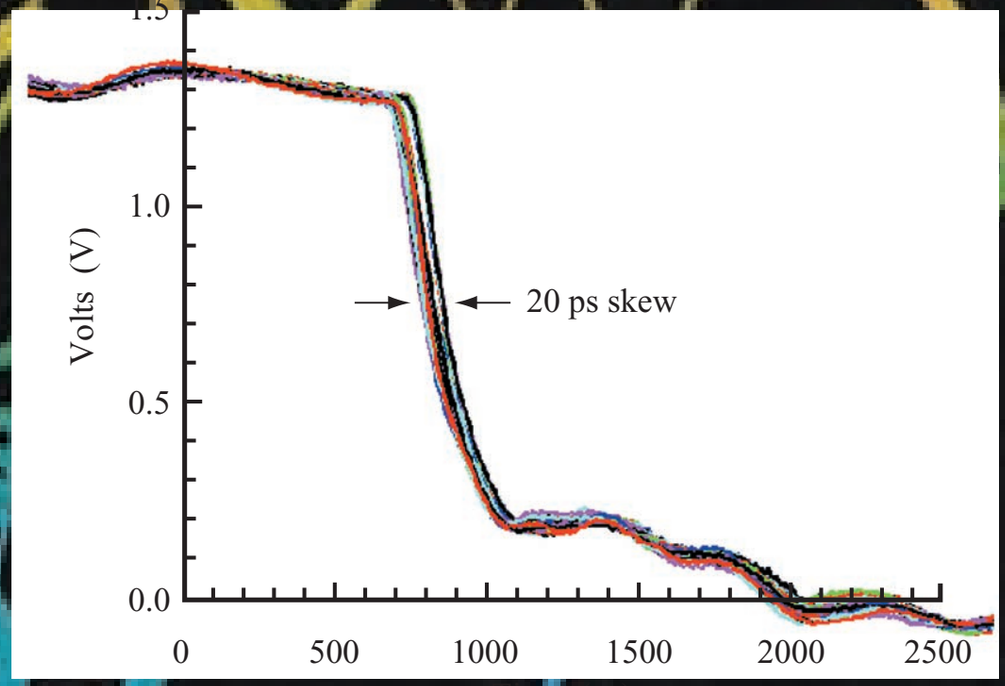
Clock Tree Delays, IBM "Power" CPU

Buffer level 2

Buffer level 1

x

y



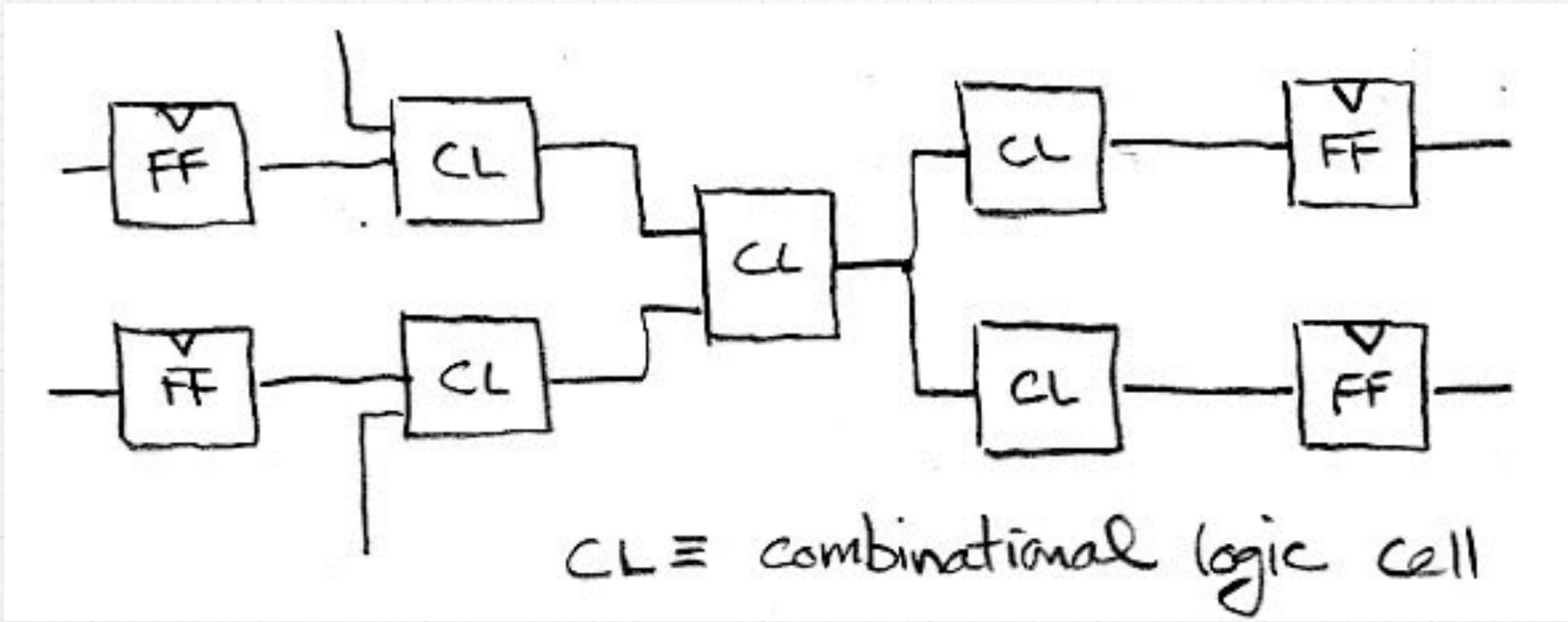
Delay

x

y

Clock Tree Delays, IBM Power

Components of Path Delay

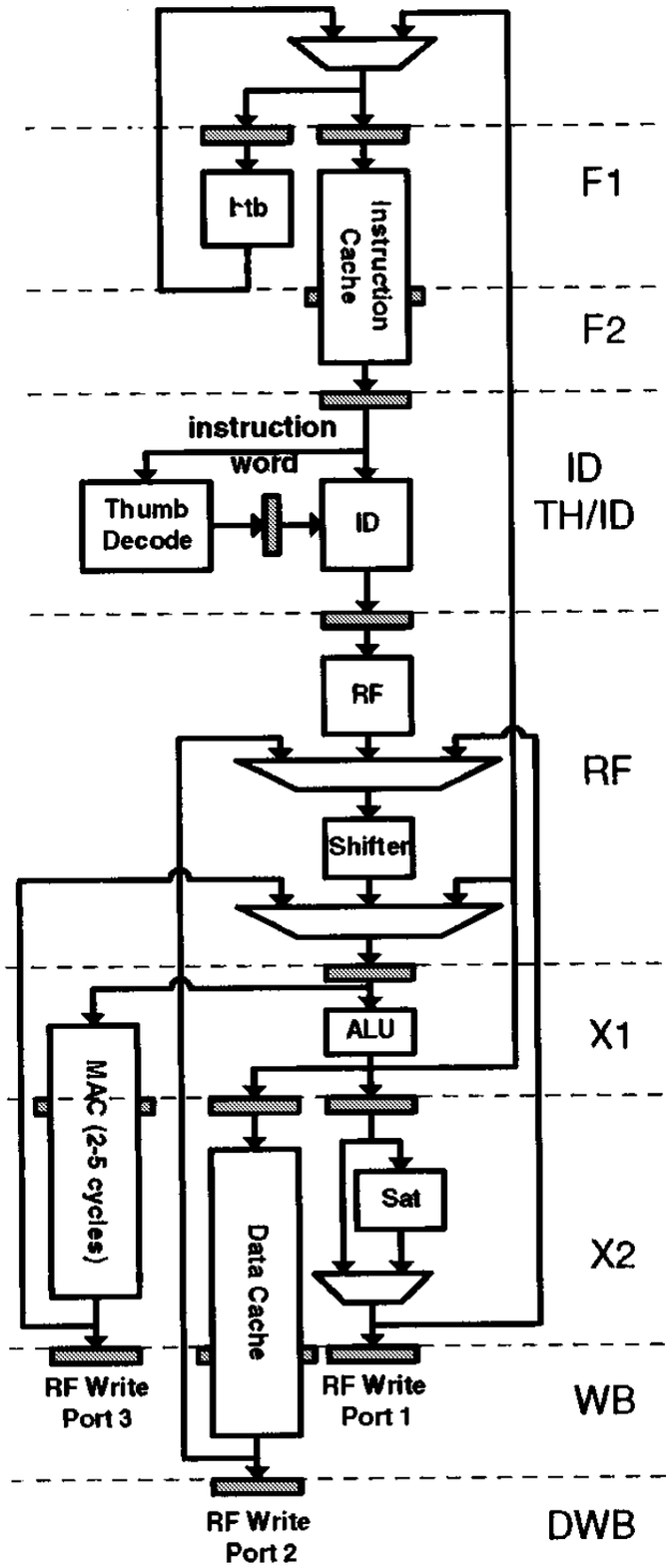


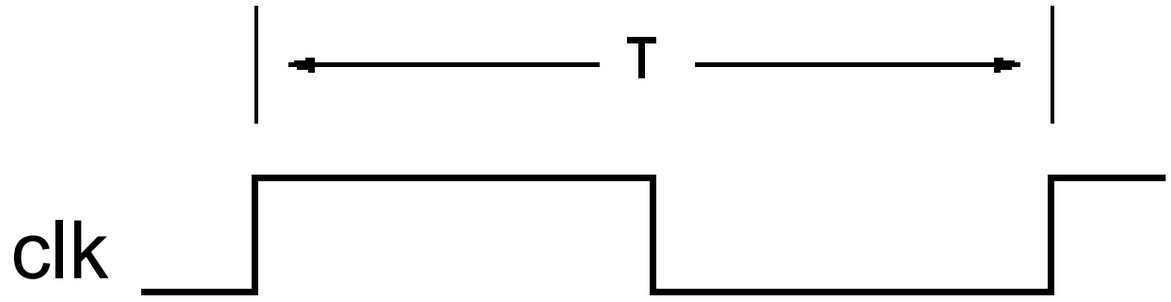
* # of levels of logic
* Internal cell delay
* wire delay
* cell input capacitance
* cell fanout
* cell output drive strength

Who controls the delay?

	foundary engineer (TSMC)	Library Developer (Aritsan)	CAD Tools (DC, IC Compiler)	Designer (Chris)
1. # of levels			synthesis	RTL
2. Internal cell delay	physical parameters	cell topology, trans sizing	cell selection	
3. Wire delay	physical parameters		place & route	layout generator
4. Cell input capacitance	physical parameters	cell topology, trans sizing	cell selection	instantiation
5. Cell fanout			synthesis	RTL
6. Cell drive strength	physical parameters	transistor sizing	cell selection	instantiation

From Delay Models to Timing Analysis



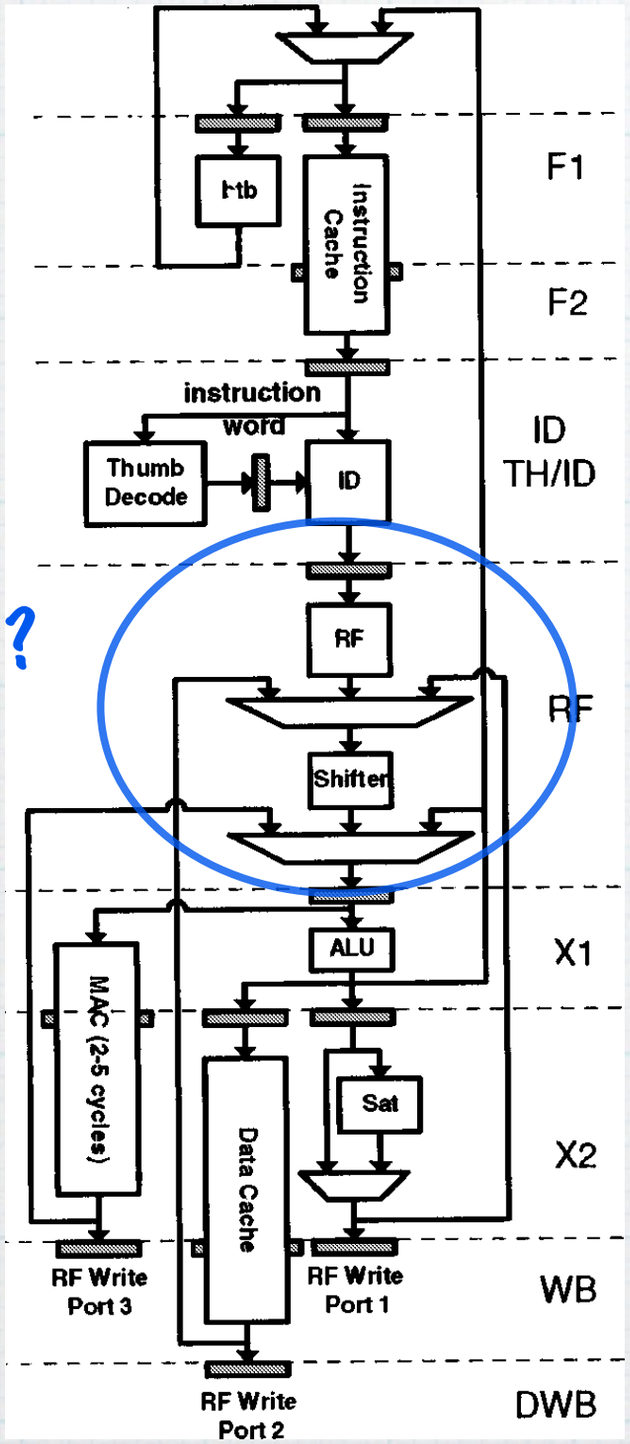


Timing Analysis

What is the smallest T that produces correct operation? Or, can we meet a target T?

f	T
1 MHz	1 µs
10 MHz	100 ns
100 MHz	10 ns
1 GHz	1 ns

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.

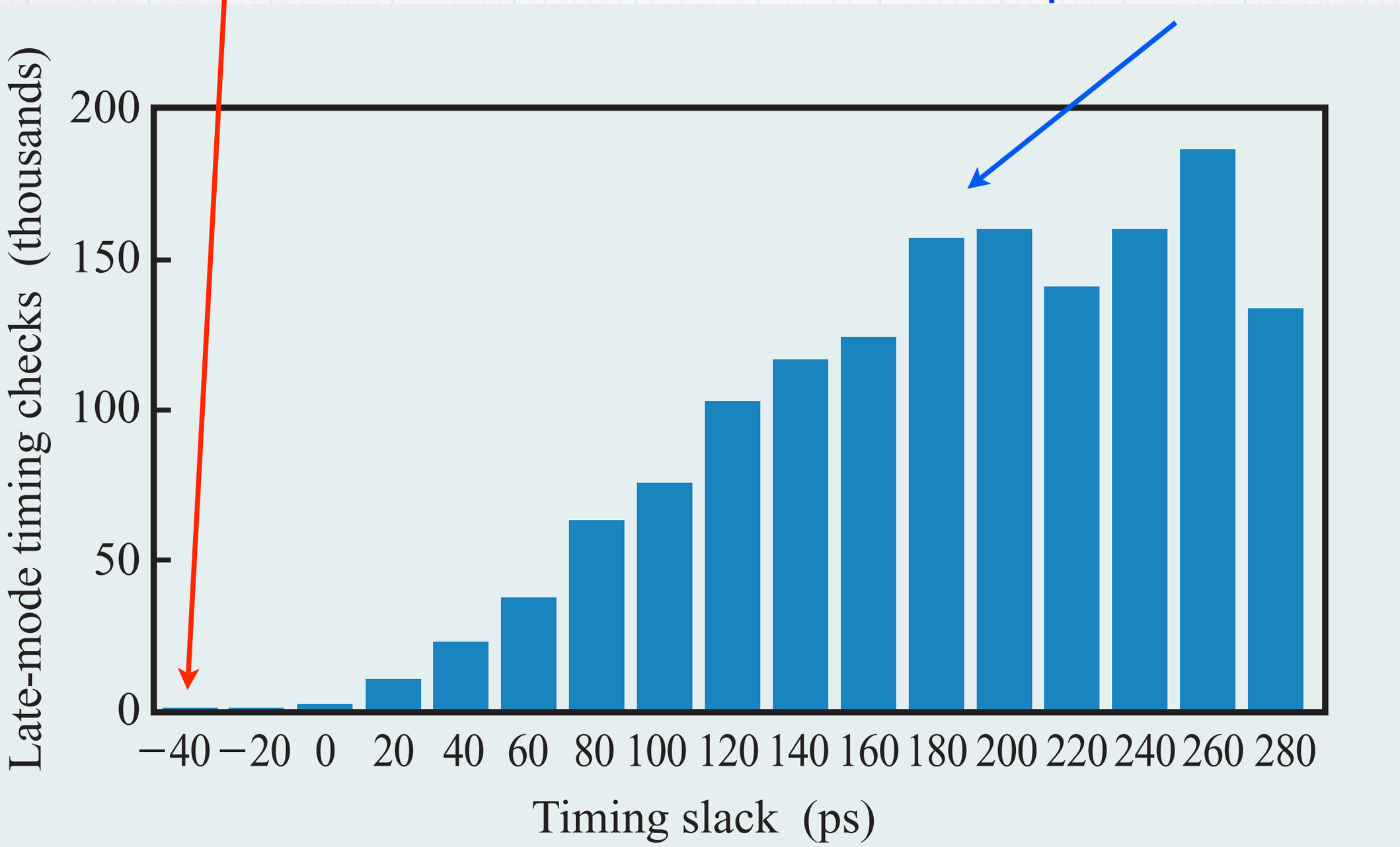
- Design tools help in the search.
- **Synthesis tools** work to meet clock constraint, report delays on paths,
  - Special **static timing analyzers** accept a design netlist and report path delays,
  - and, of course, **simulators** can be used to determine timing performance.

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 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

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

Circles are combinational logic, labelled with delays.

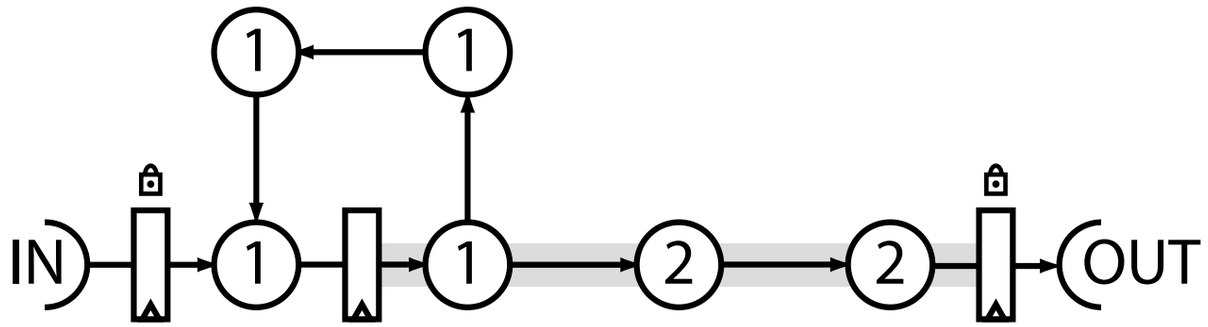


Figure 1: A small graph before retiming. The nodes represent logic delays, with the inputs and outputs passing through mandatory, fixed registers. The critical path is 5.

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

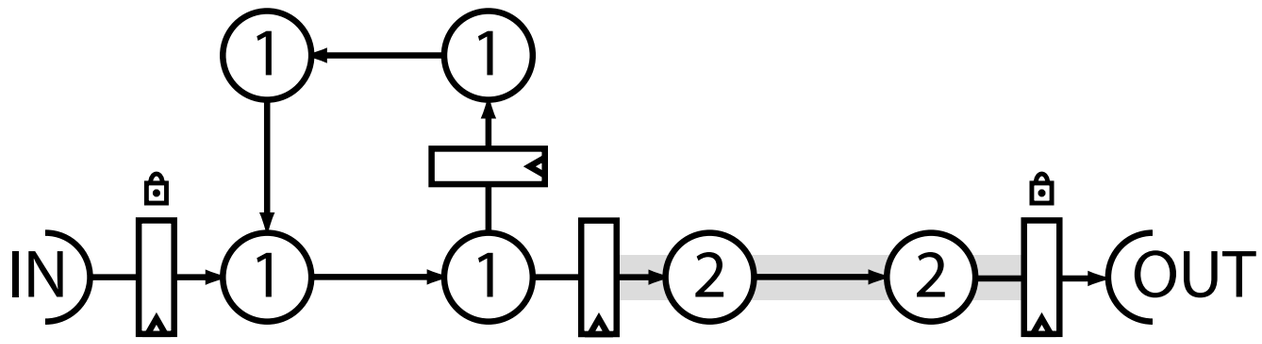


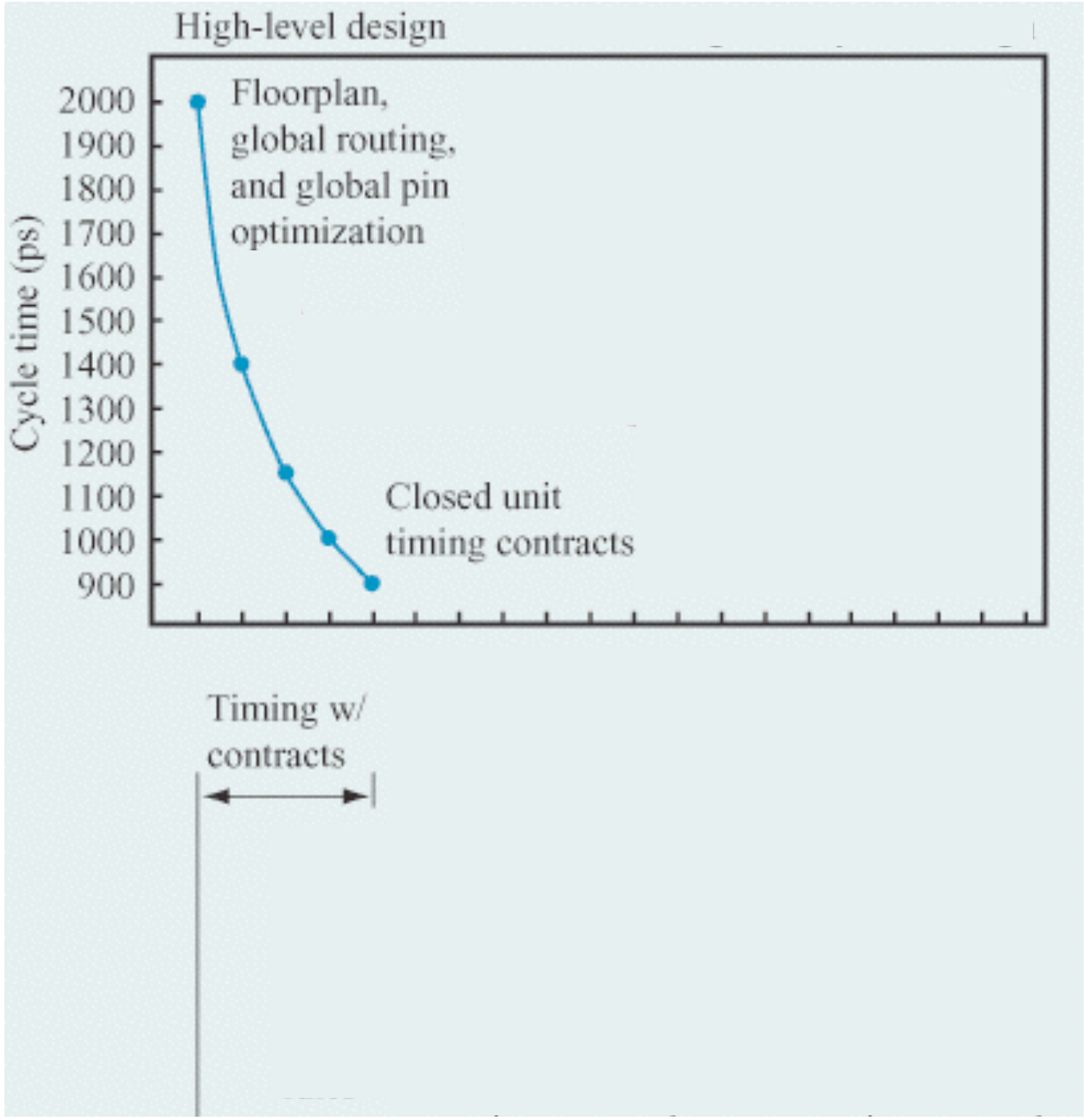
Figure 2: The example in Figure 2 after retiming. The critical path is reduced from 5 to 4.

"Technology X" can do this in simple cases.

Power 4: Timing Estimation, Closure

Timing Estimation

Predicting a processor's clock rate early in the project

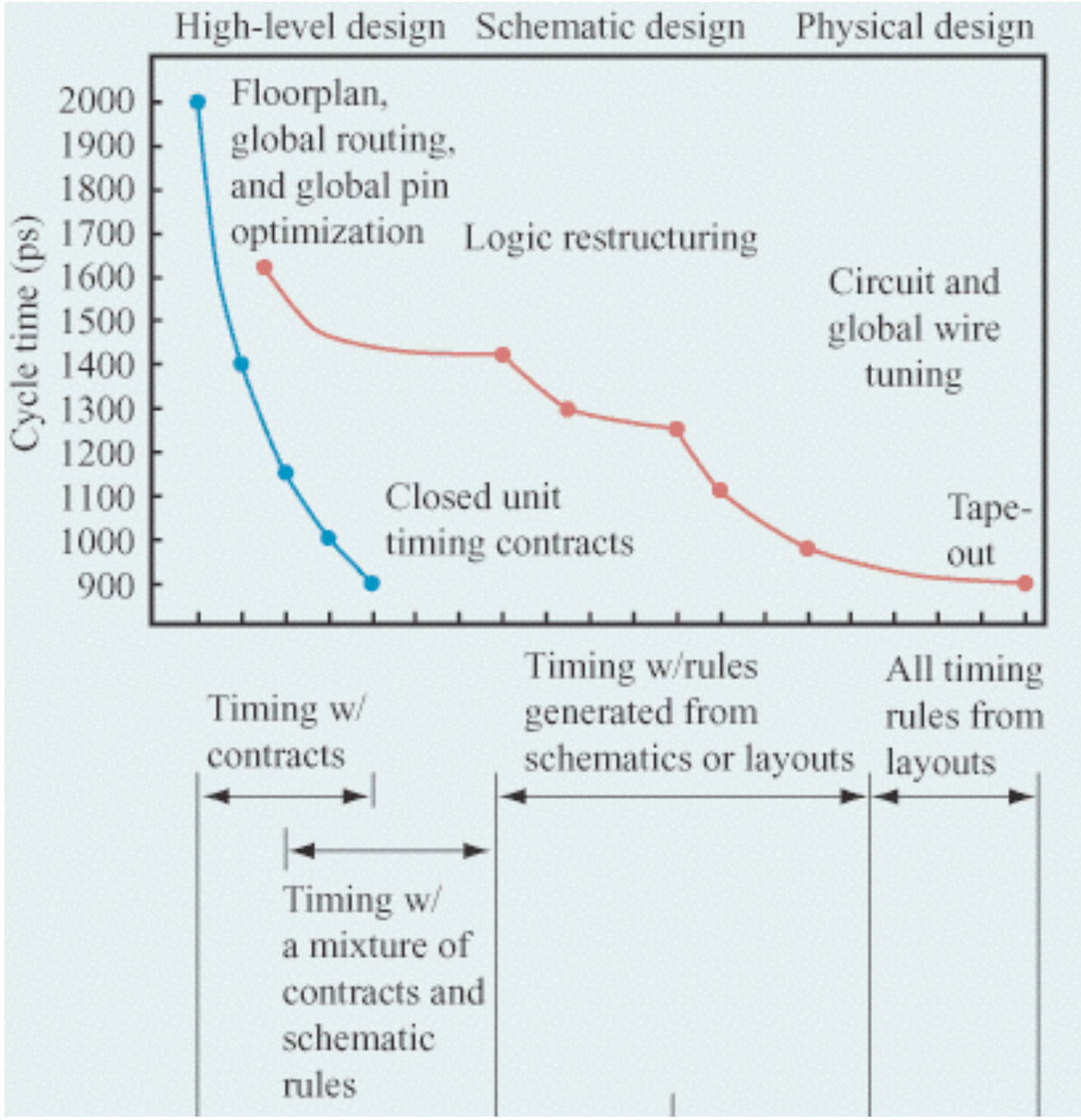


From "The circuit and physical design of the POWER4 microprocessor", IBM J Res and Dev, 46:1, Jan 2002, J.D. Warnock et al.

Power 4: Timing Estimation, Closure

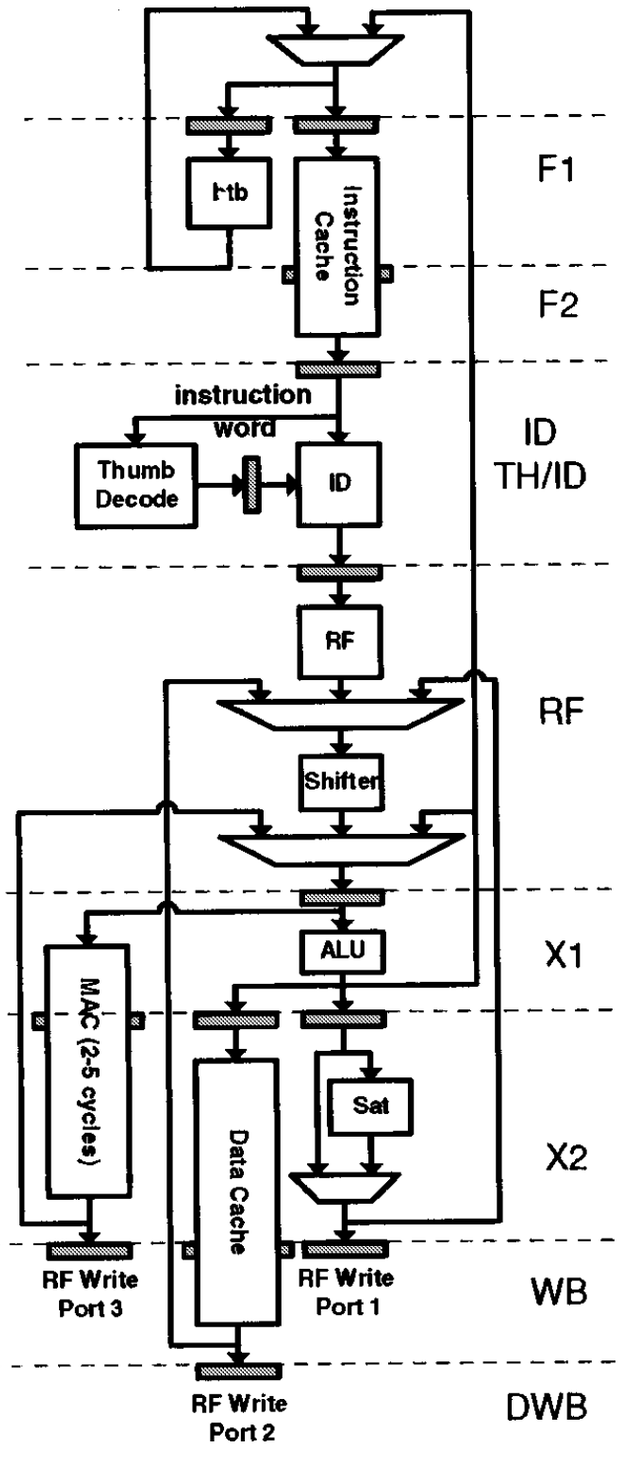
Timing Closure

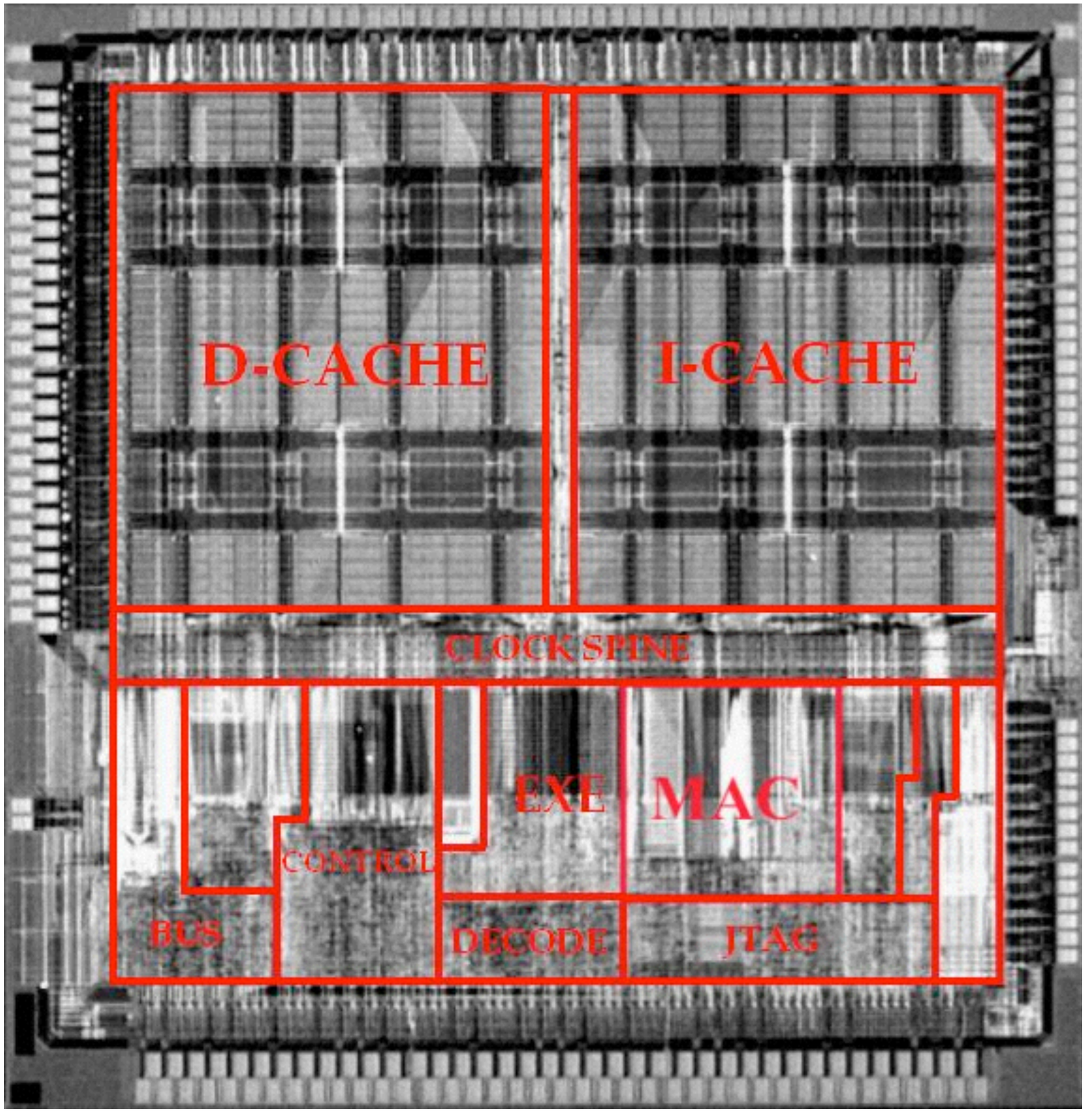
Meeting (or exceeding!) the timing estimate



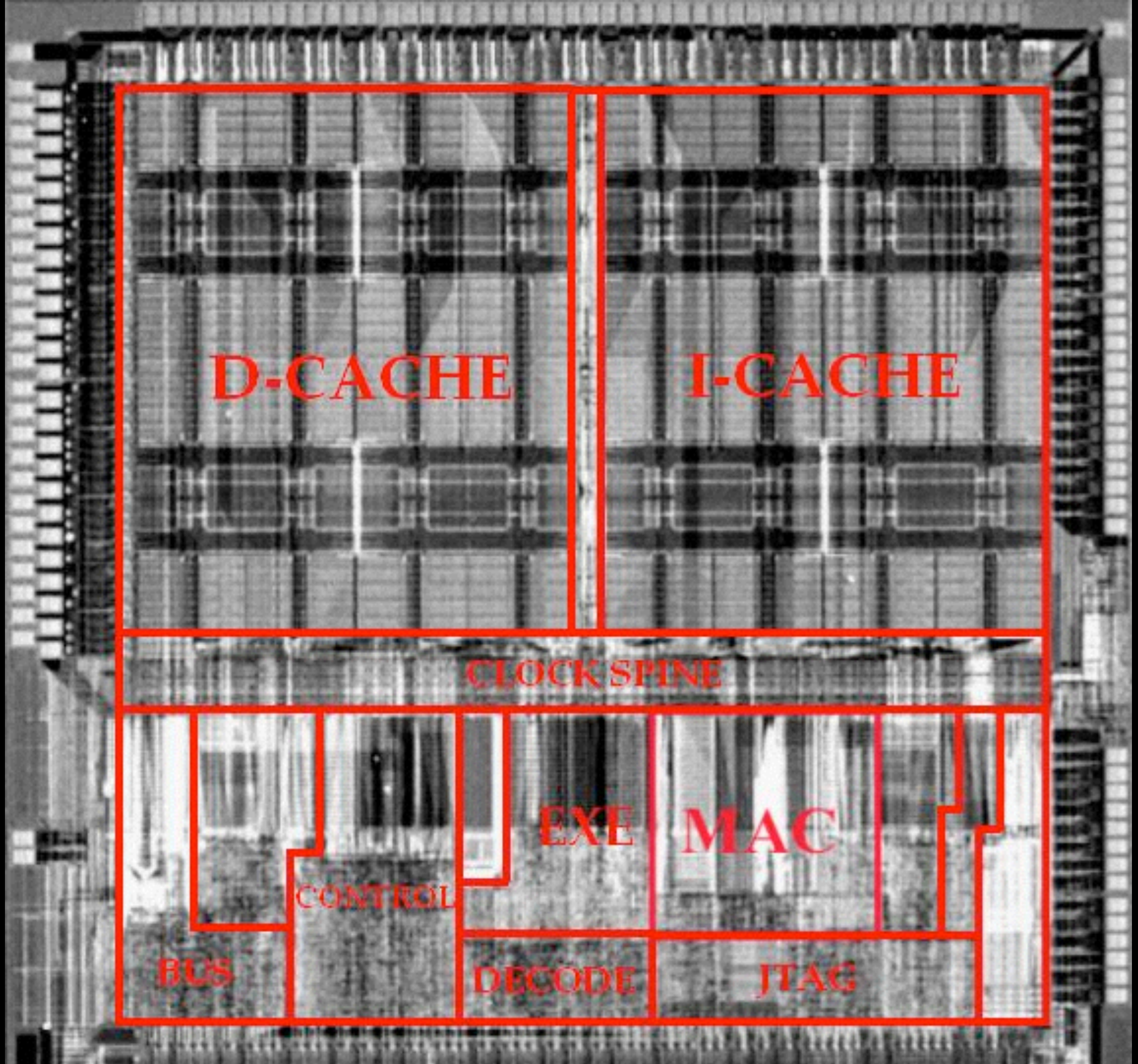
From "The circuit and physical design of the POWER4 microprocessor", IBM J Res and Dev, 46:1, Jan 2002, J.D. Warnock et al.

Floorplaning: essential to meet timing.



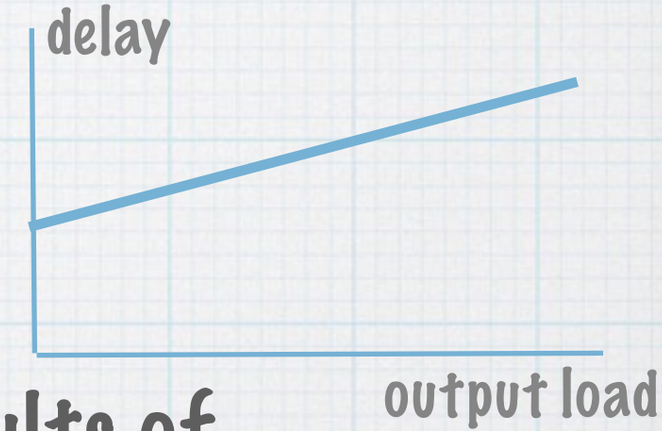


(Intel XScale 80200)


D-CACHE
I-CACHE
CLOCK SPINE
EXE
MAC
CONTROL
BUS
DECODE
JTAG

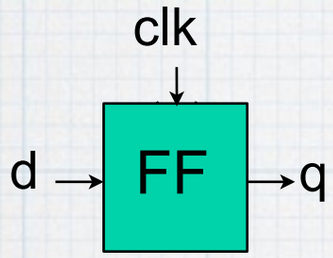
Timing Analysis Tools

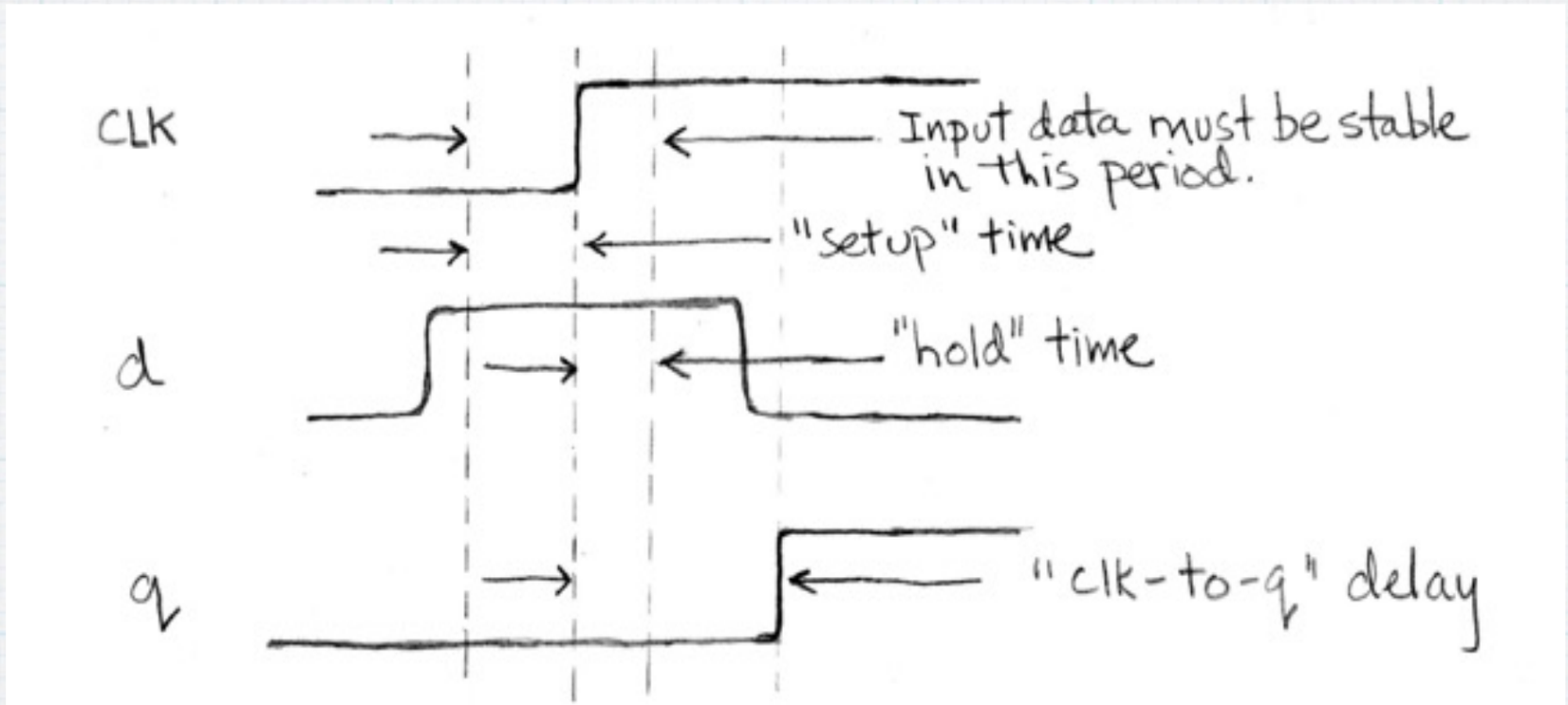
- Static Timing Analysis: Tools use delay models for gates and interconnect. Traces through circuit paths.
  - Cell delay model capture
    - For each input/output pair, internal delay (output load independent)
    - output dependent delay



- 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.

Hold-time Violations





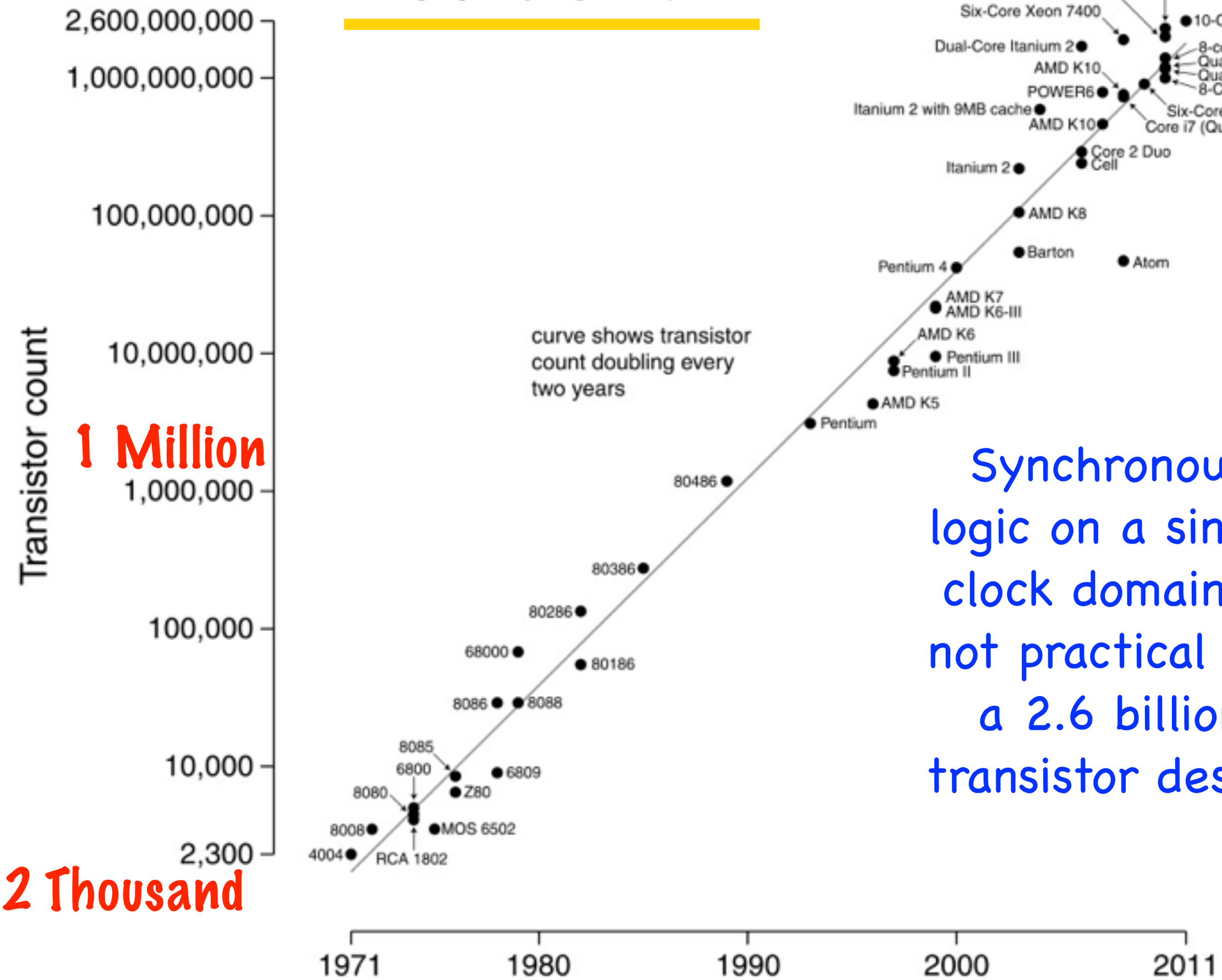
- Some state elements have positive hold time requirements.
  - How can this be?
- Fast paths from one state element to the next can create a violation. (Think about shift registers!)
- CAD tools do their best to fix violations by inserting delay (buffers).
  - Of course, if the path is delayed too much, then cycle time suffers.
  - Difficult because buffer insertion changes layout, which changes path delay.

Moore's Law

2.6 Billion

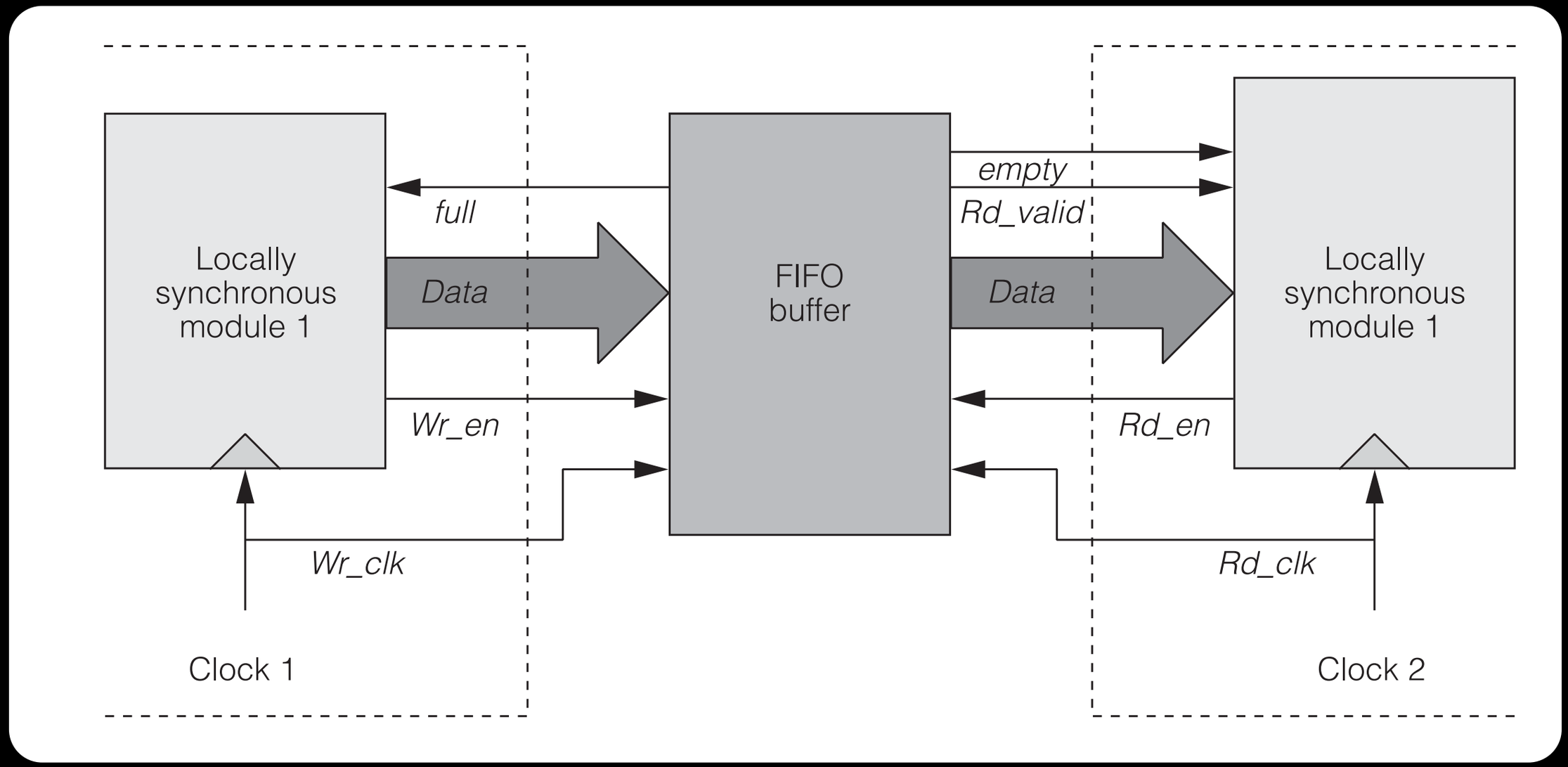
1 Million

2 Thousand



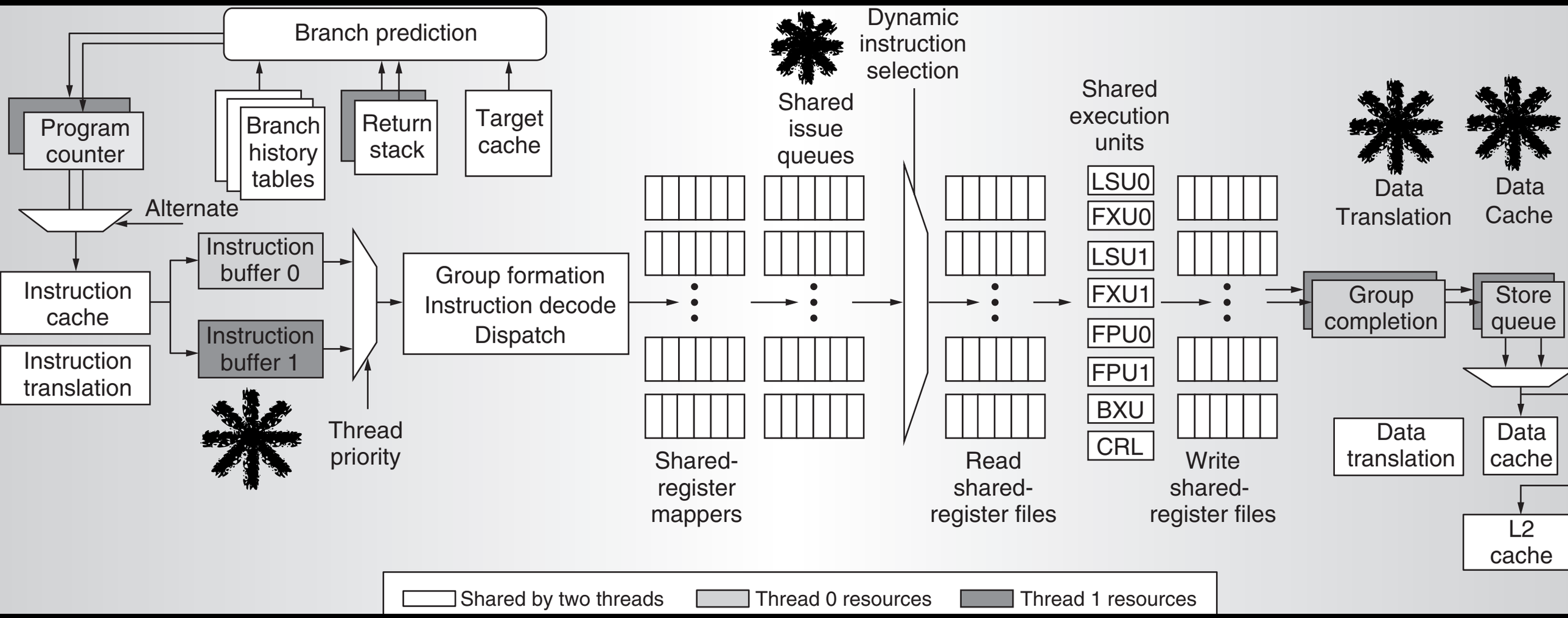
Synchronous logic on a single clock domain is not practical for a 2.6 billion transistor design

GALS: Globally Asynchronous, Locally Synchronous



Synchronous modules typically 50K-1M gates, so that the synchronous logic approach works well without requiring heroics. Examples ...

IBM Power 5 CPU - Dynamically Scheduled

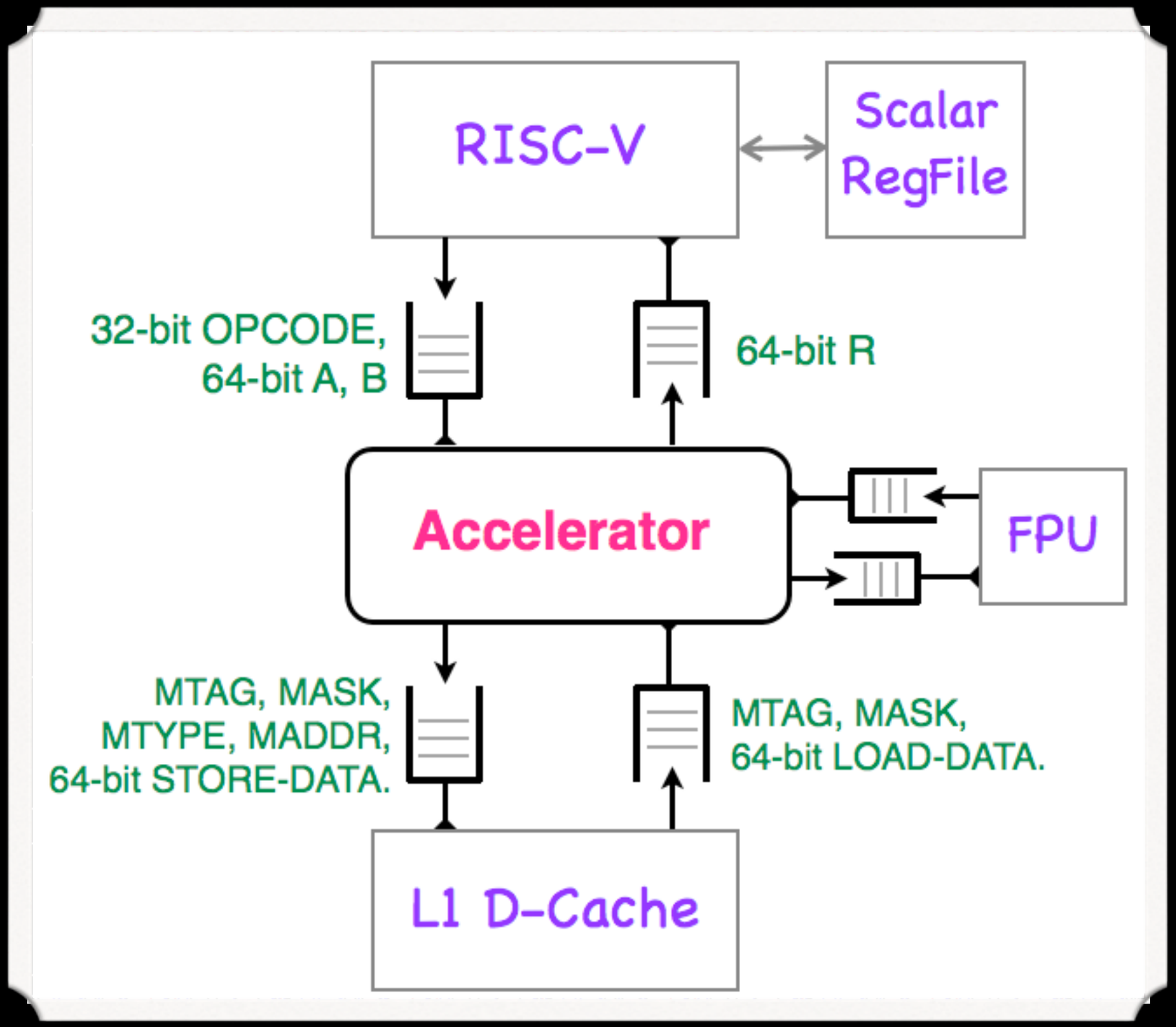


Stars denote FIFOs that create separate synchronous domains. An example of how architecture and circuits work together.

Rocket uses GALS for accelerator interface

Your project interfaces with the RISC-V pipeline and the memory system using FIFOs.

Your timing closure is independent of the CPU logic domain.



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

End of Physical Realities part 1 Timing

Simple exercises for gaining intuition about timing for your process + EDA tools.

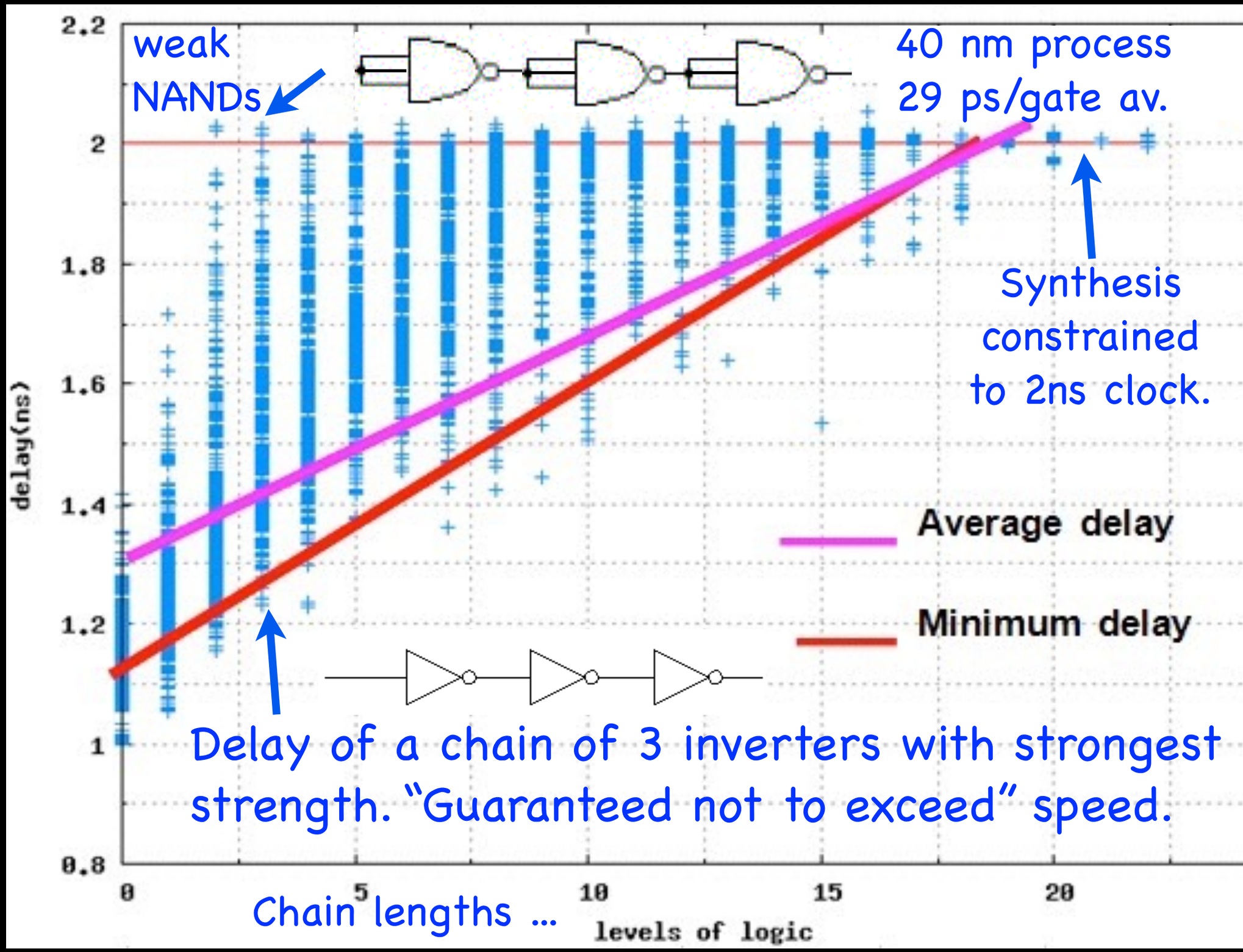
Know Your Numbers

Thanks to Bhupesh Dasila, Open-Silicon Bangalore

Synthesize gate chains using hand-specified library cells

Exercises cell library and place and route tools.

Lets you know how many levels of logic you can use in the best case.

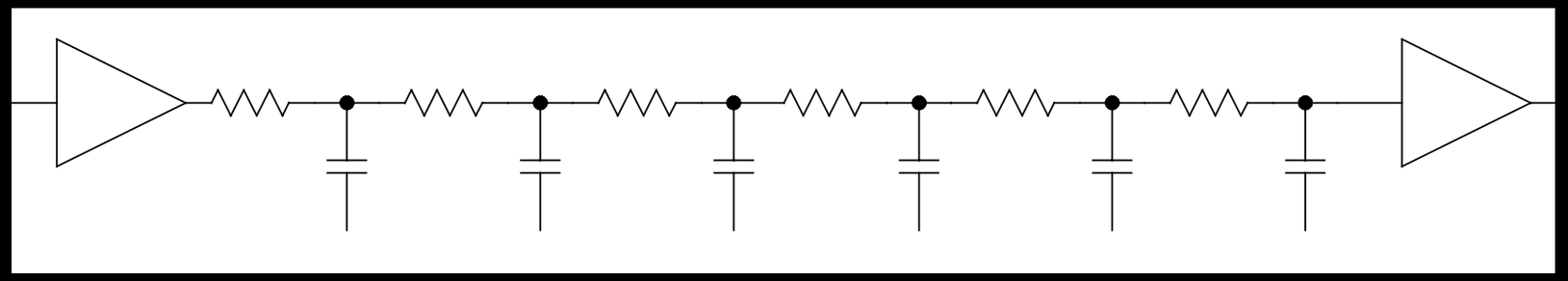


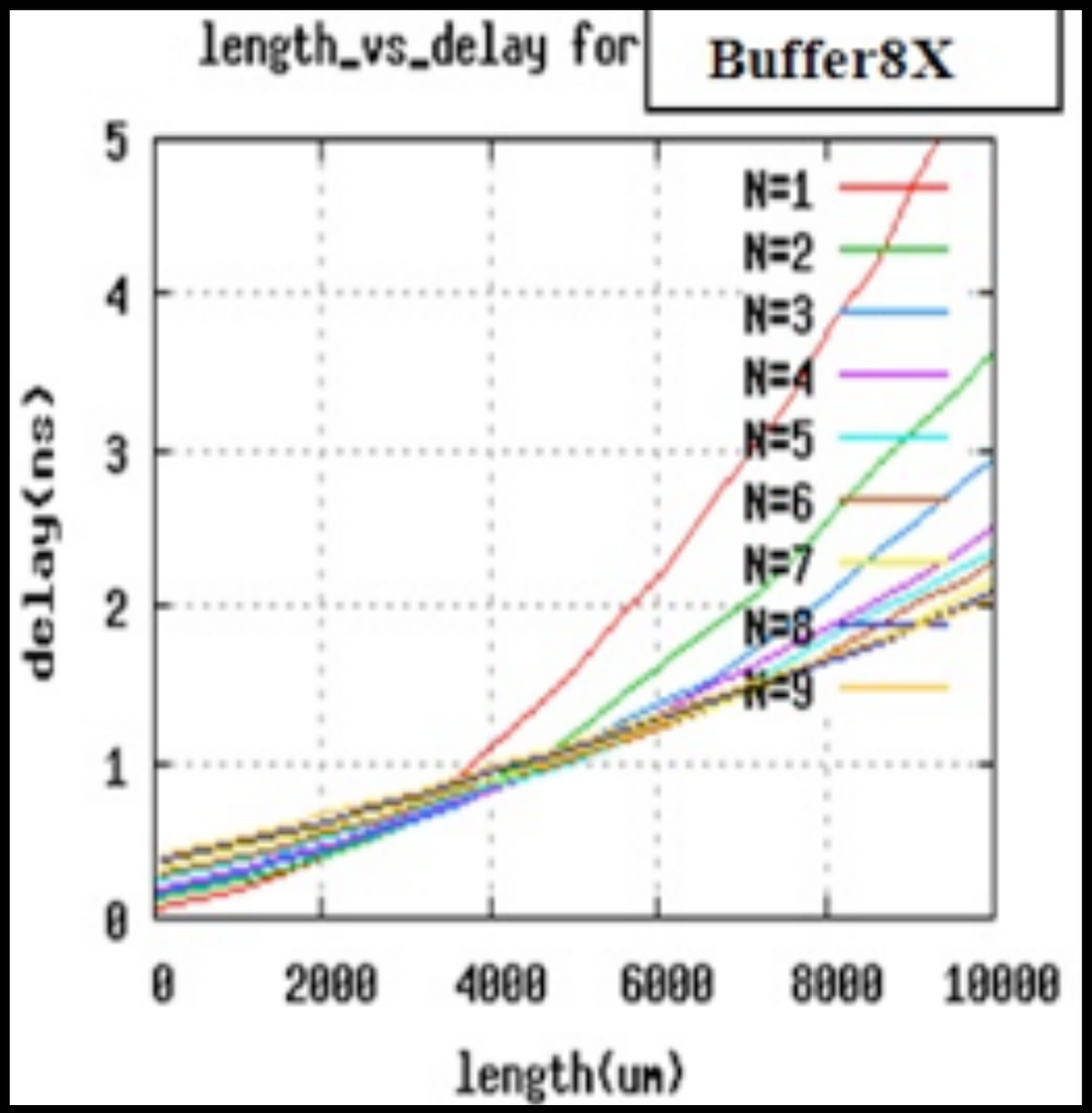
Helps you "see through" ... "Technology X".

Force P&L to drive a long wire with a known buffer cell.

Vary driver strength, wire length, metal layer.

Shows the maximum distance two gates can be placed and still meet your clock period.





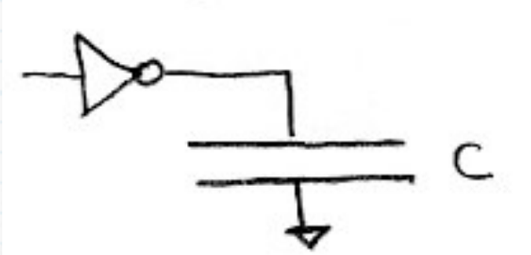
Distributed RC is the square of the length is clearly seen!

Delay/length = 195.8ps

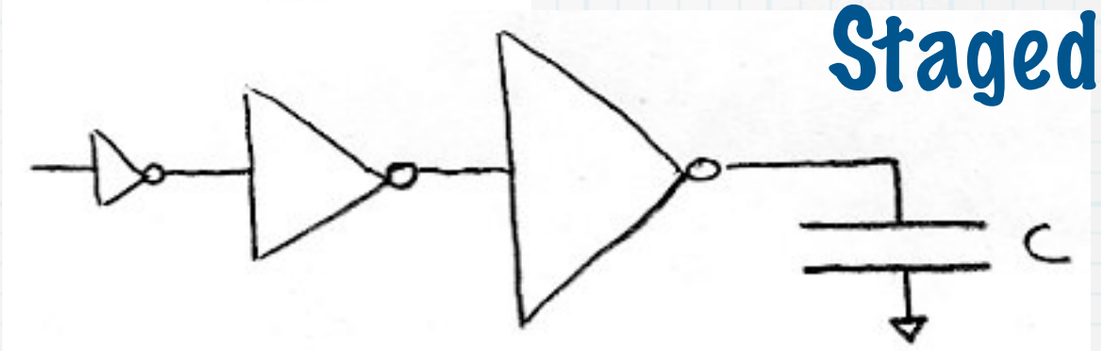
buffer type	delay/length (ps/mm)	area (um^2)	leakagePower (nW)
Buffer8X	195.8	2.8	1565.6
Buffer12X	175.7	3.9	2199.0
Buffer16X	165.2	5.1	3037.7
Buffer20X	162.0	6.2	3693.8

Driving Large Loads

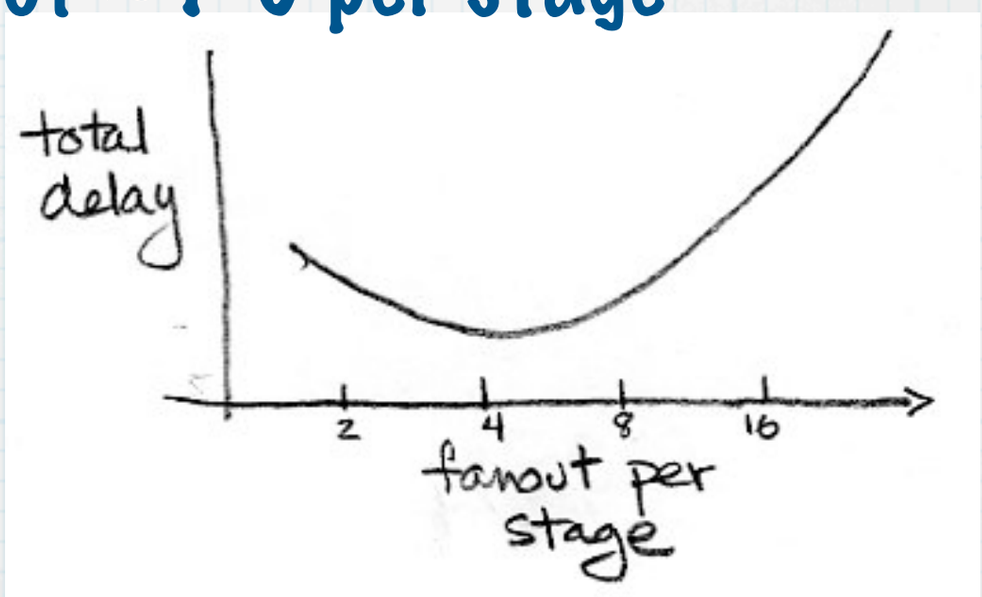
- Large fanout nets: clocks, resets, memory bit lines, off-chip
- Relatively small driver results in long rise time (and thus large gate delay)



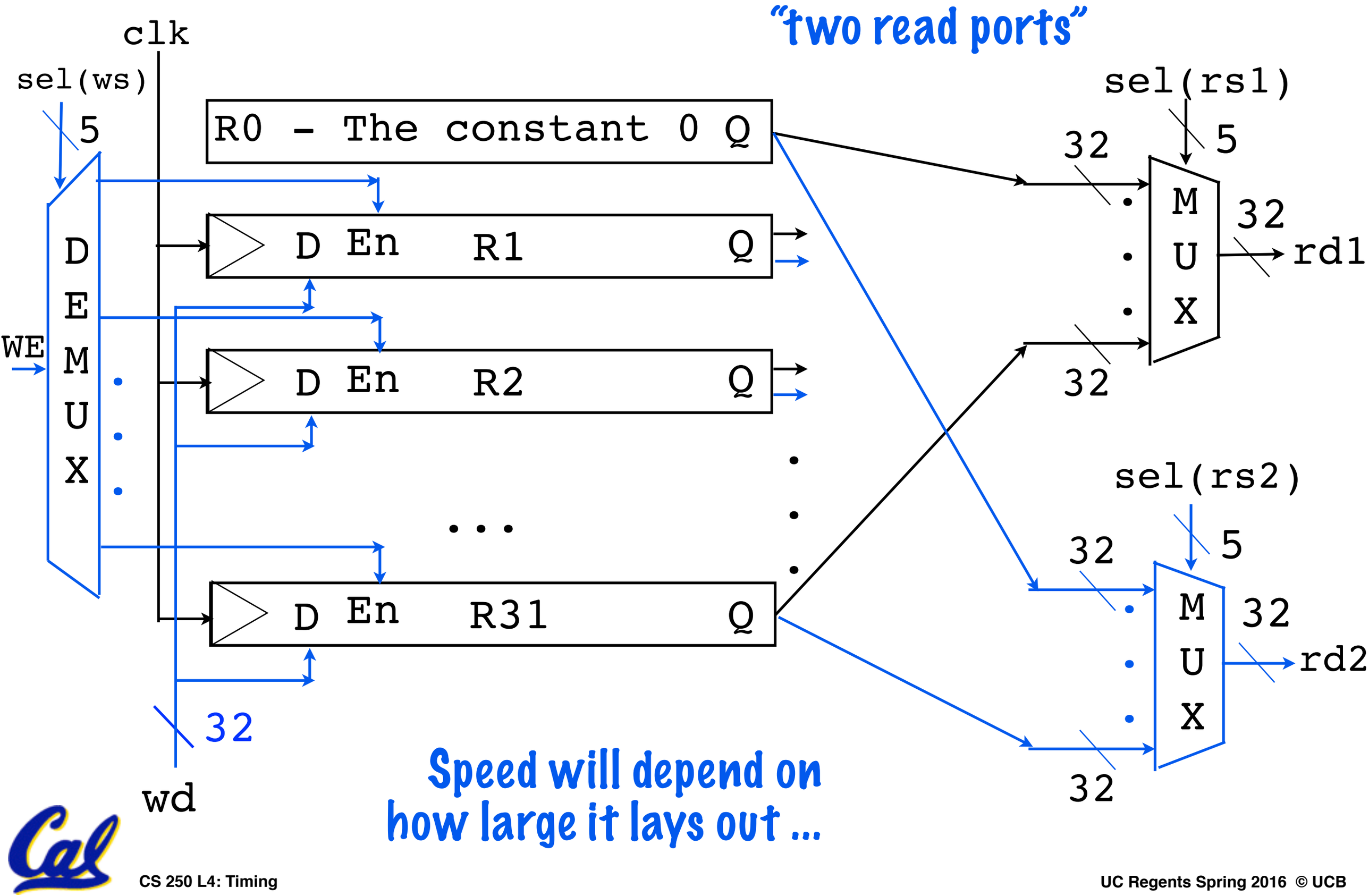
- Strategy: **Staged Buffers**



- Optimal trade-off between delay per stage and total number of stages ⇒ **fanout of ~4-6 per stage**



Register file: Synthesize, or use SRAM?



Synthesized, custom, and SRAM-based register files, 40nm

For small register files, logic synthesis is competitive.

Not clear if the SRAM data points include area for register control, etc.

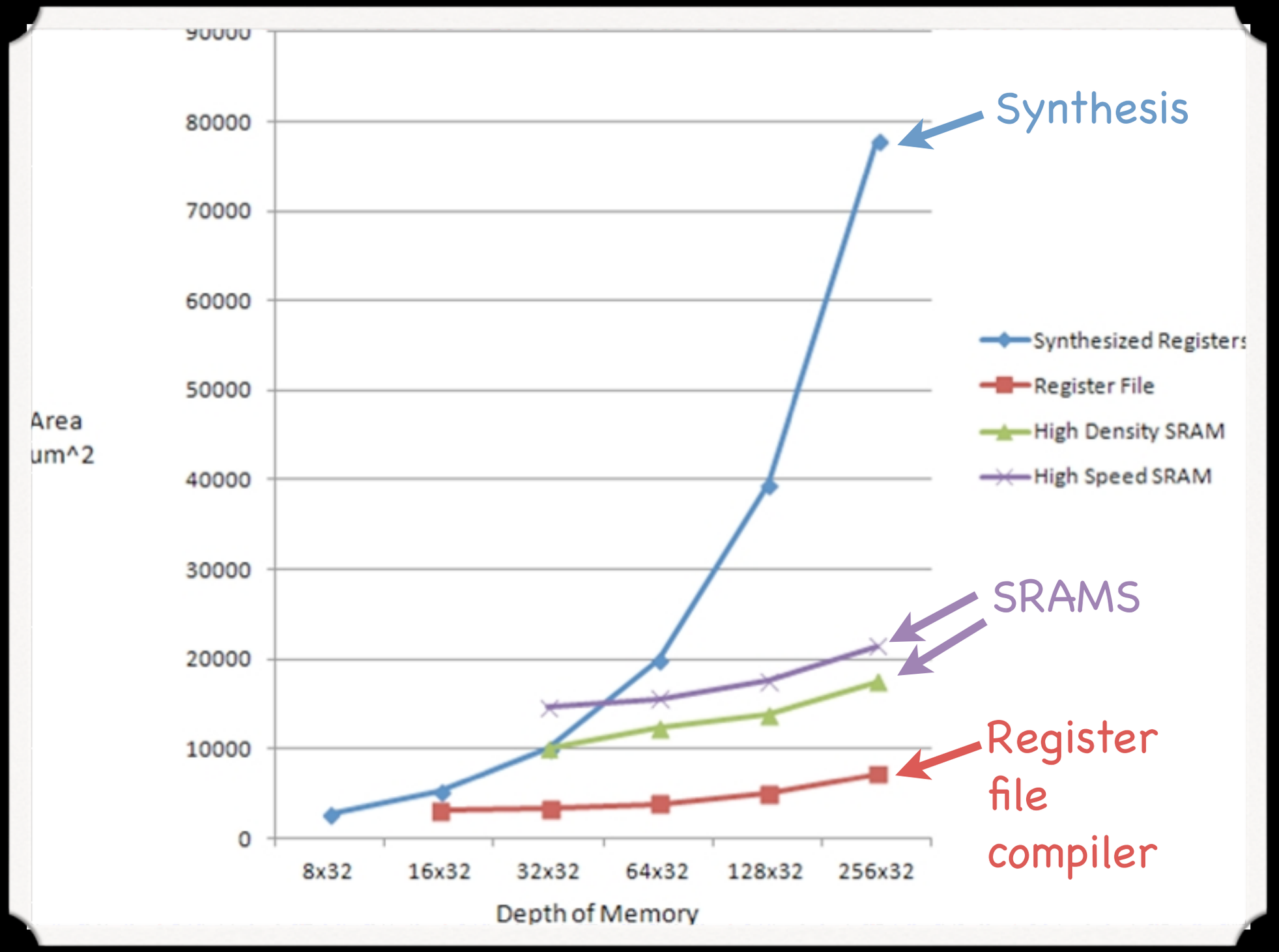
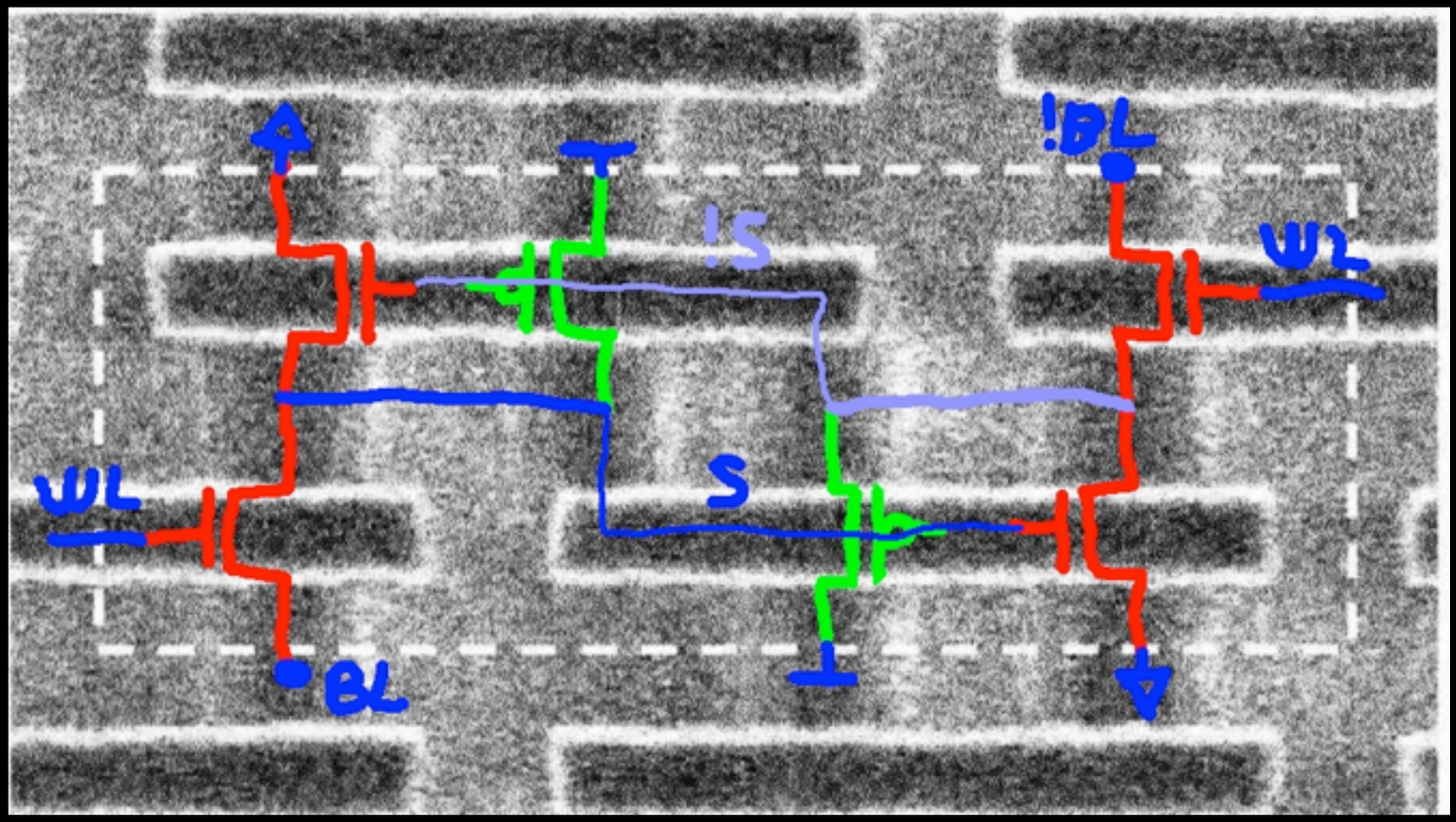


Figure 3: Using the raw area data, the physical implementation team can get a more accurate area estimation early in the RTL development stage for floorplanning purposes. This shows an example of this graph for a 1-port, 32-bit-wide SRAM.

Bhupesh Dasila

Today: Timing insights for your project

What we're not doing. If this class was EE 241 and your project was an SRAM:



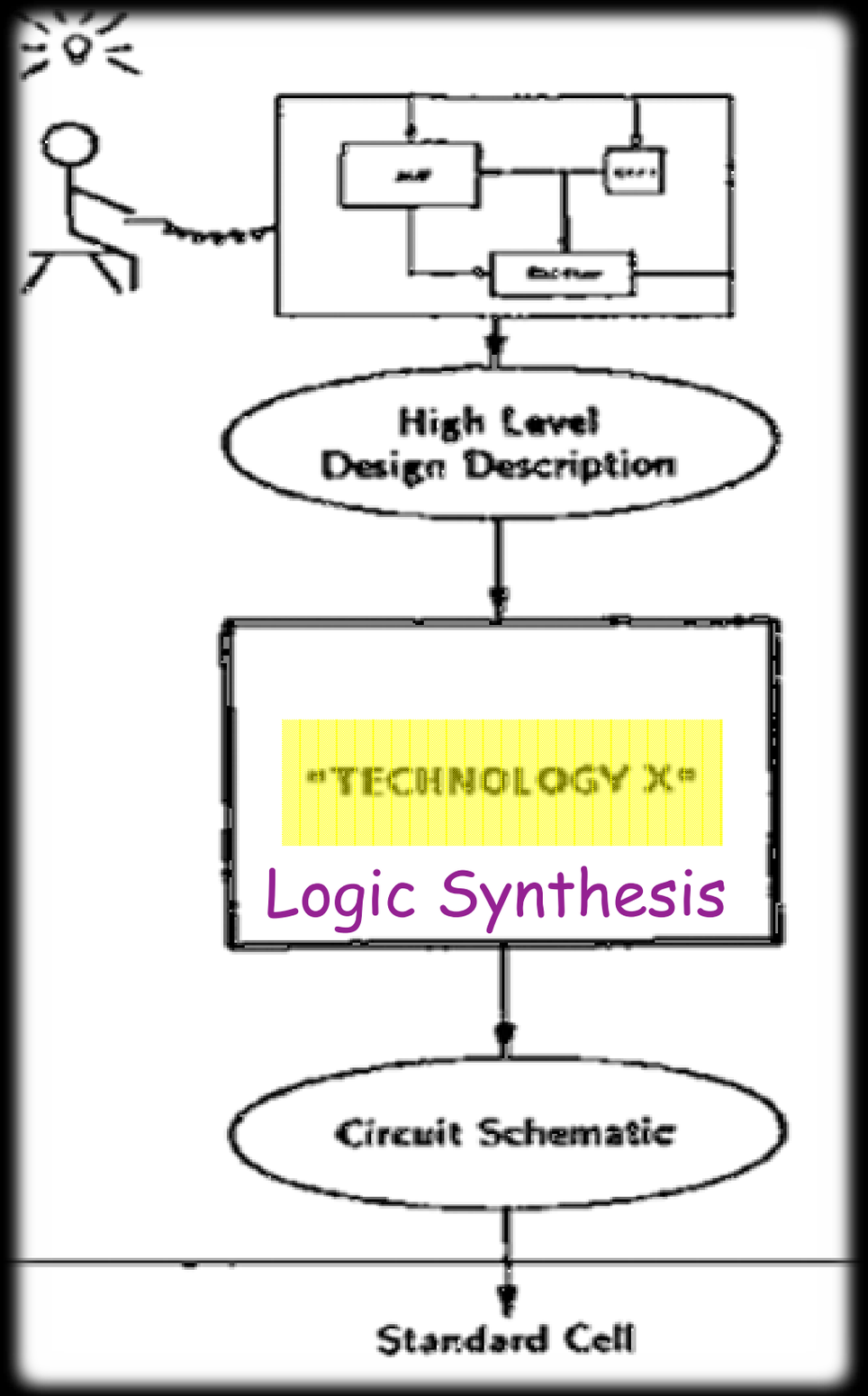
You could see through down to the layout.

Timing? Use SPICE on this hand-drawn schematic.

Technology X: The CS 250 timing challenge.

- What we are doing --->
- If your accelerator is too slow ... two options:

Top-down: Rework high-level micro-architecture. Let Technology X keep its job. **Today.**



Bottom-up: Take control away from logic synthesis. Use HDL as textual schematic. Also, use command-line tool flags.

Sometimes necessary. Ben is the expert, ask in discussion section.