UC Berkeley
Teaching Professor
Dan Garcia

The Beauty and Joy of Computing

Concurrency

Koomey's Law – Efficiency 2x every 18mo

Prof Jonathan Koomey looked at 6 decades of data and found that energy efficiency of computers doubles roughly every 18 months. This is even more relevant as battery-powered devices become more popular. Restated, it says that for a fixed computing load, the amount of battery you need drops by half every 18 months (now 2.6 yrs). This was true before transistors!

en.wikipedia.org/wiki/Koomey%27s_law

Basic Definitions

Concurrency: Definition & Examples

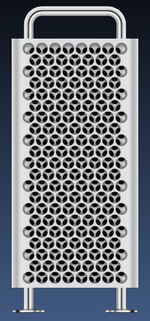
Definition: **A** property of computer systems in which several computations are executing simultaneously, and potentially interacting with each other.

Examples:

- Mouse cursor movement while Snap*!* calculates.
- Screen clock advances while typing in a text.
- Busy cursor spins while browser connects to server, waiting for response
- Walking while chewing gum

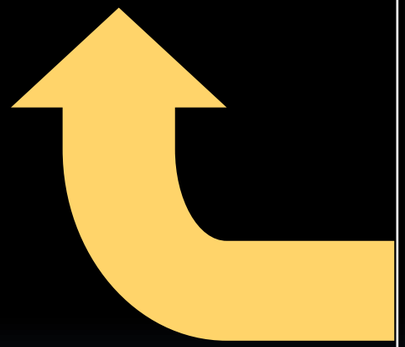
Concurrency & Parallelism

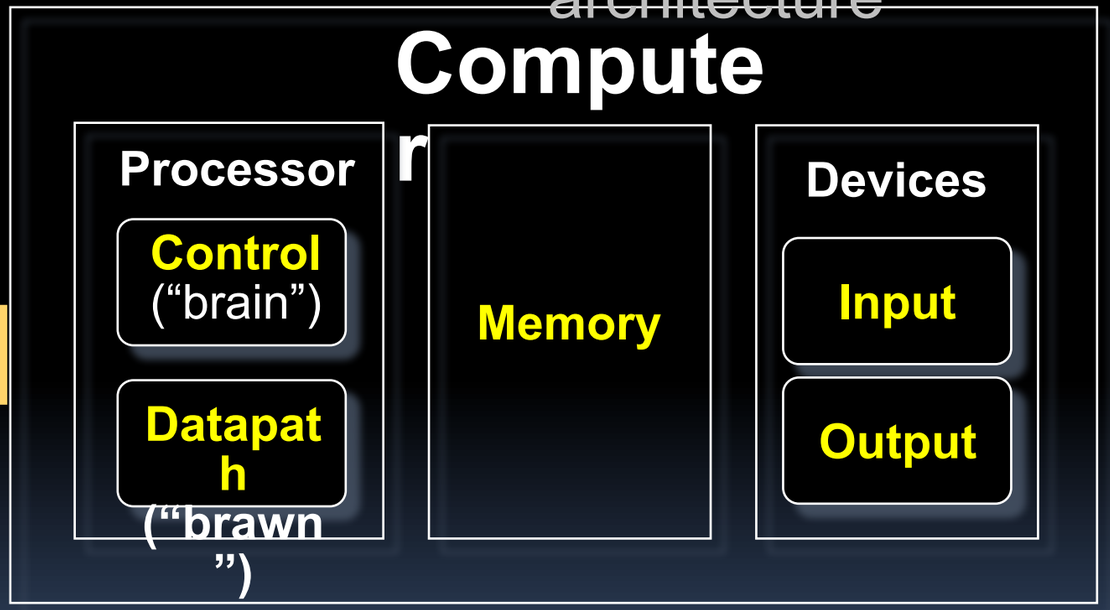
Intra-computer	Inter-computer
▪ Today's lecture	▪ Saving World w/ Computing
▪ Multiple computing "helpers" are cores <u>within one machine</u>	▪ Multiple computing "helpers" are <u>different machines</u>
▪ Aka "multi-core"	▪ Aka "distributed computing"
▯ Although GPU parallism is also "intra-computer"	▯ Grid & cluster computing



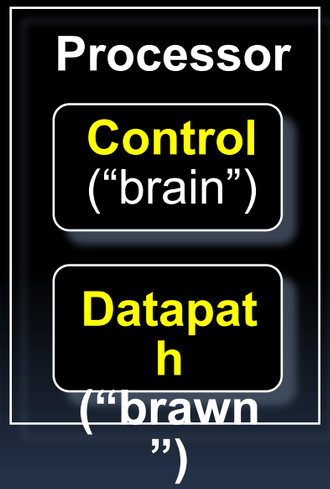
5 Components of a Computer

John von Neumann invented this architecture

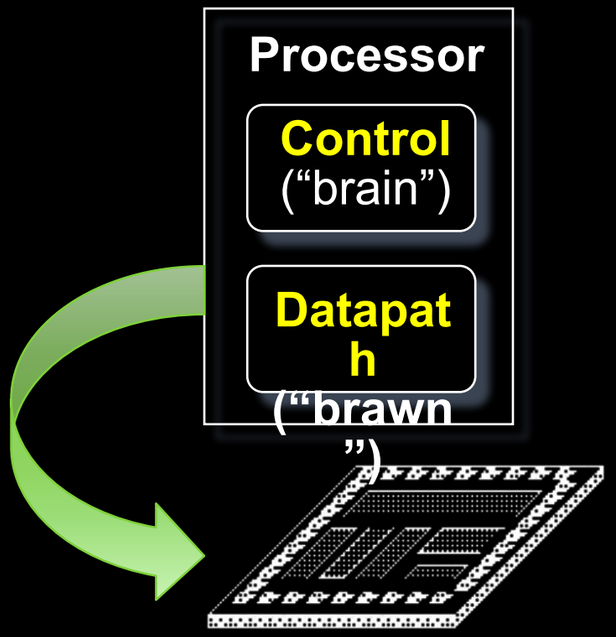




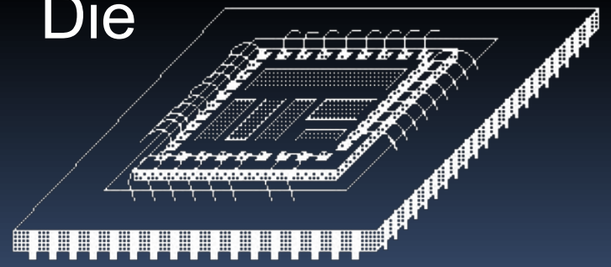
But what is INSIDE a Processor?



But what is INSIDE a Processor?



Bare Processor Die



Chip in Package

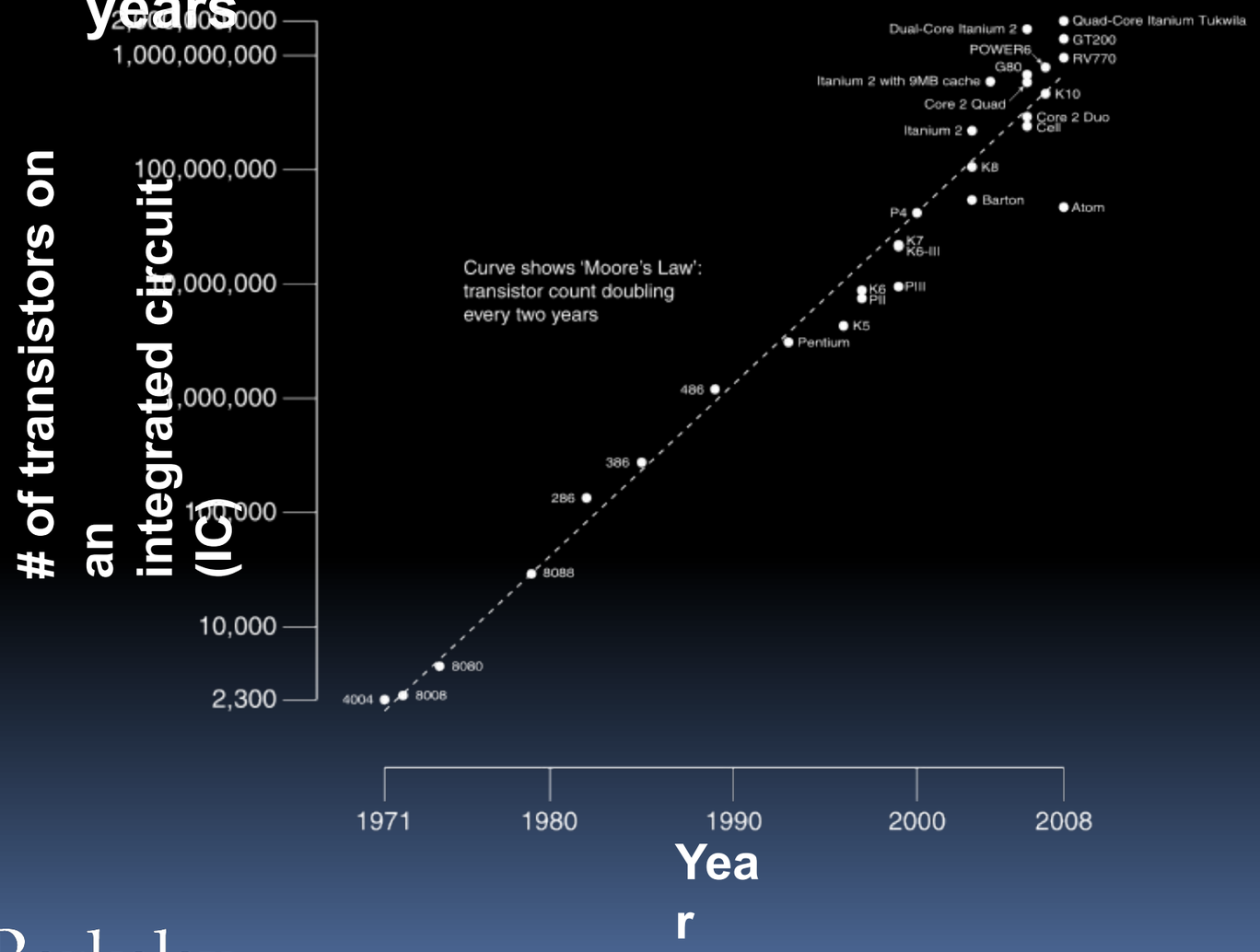
- Primarily Crystalline Silicon
- 1 mm – 25 mm on a side
- 2020 "feature size" (aka process) ~ 7 nm = 7 x 10^{-9} m (5nm, 3nm next!)
- Billions Trillions of transistors
- 3 - 15 conductive layers
- "MOSFET" (metal oxide semiconductor field-effect transistor) - most common

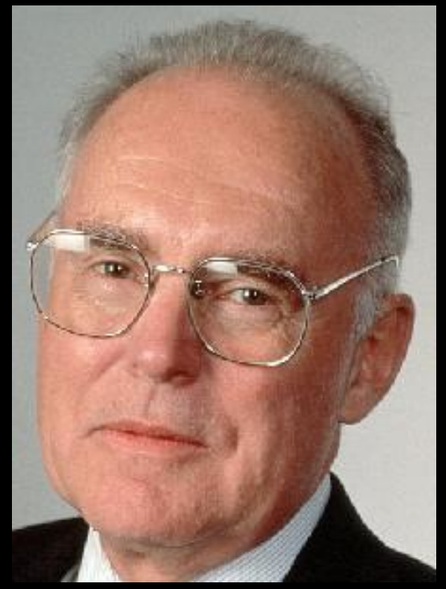
- **Package provides:**
  - spreading of chip-level signal paths to board-level
  - heat dissipation.
- **Ceramic or plastic with gold wires.**

Moore’s Law

Moore's Law

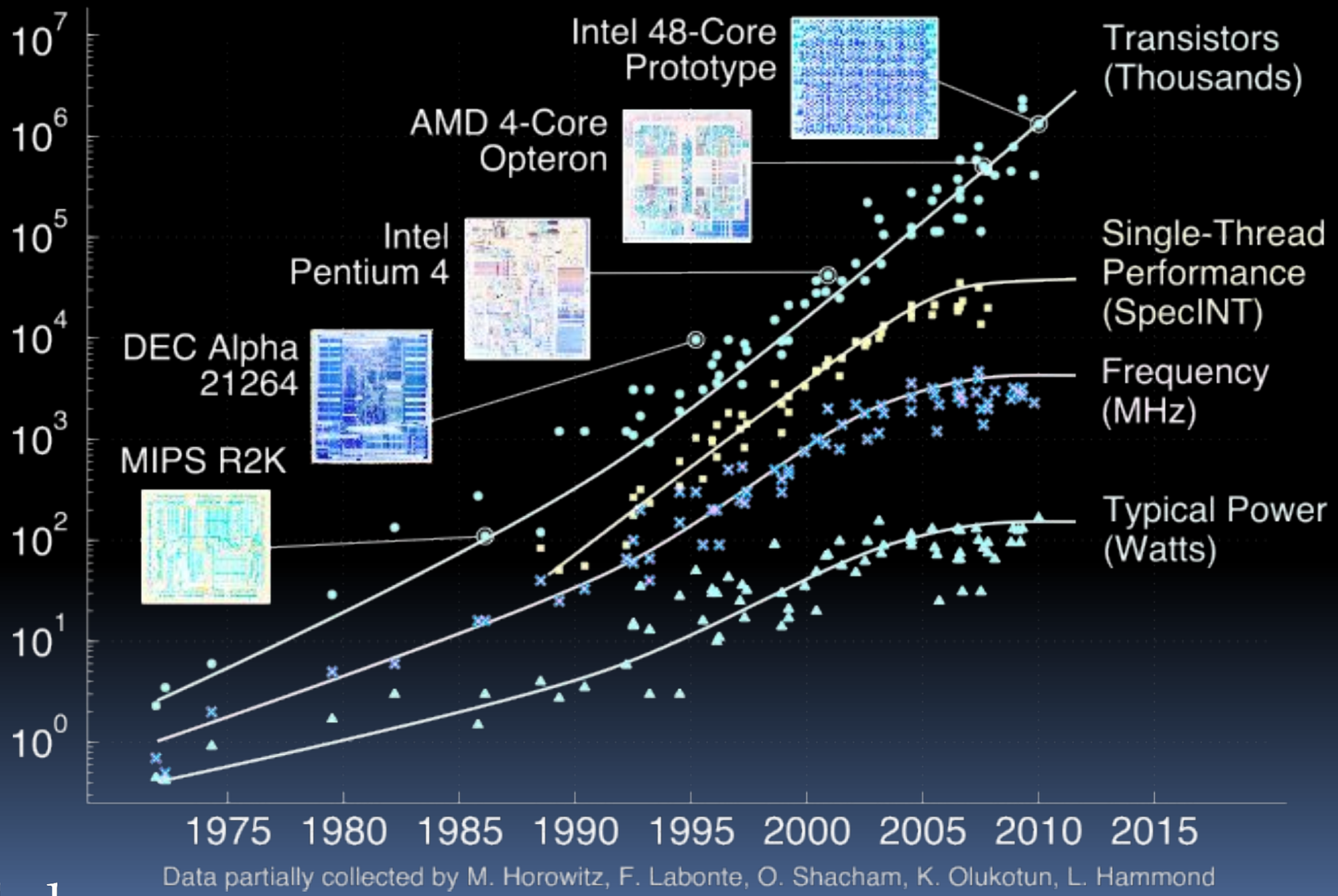
Predicts: 2X Transistors / chip every 2 years



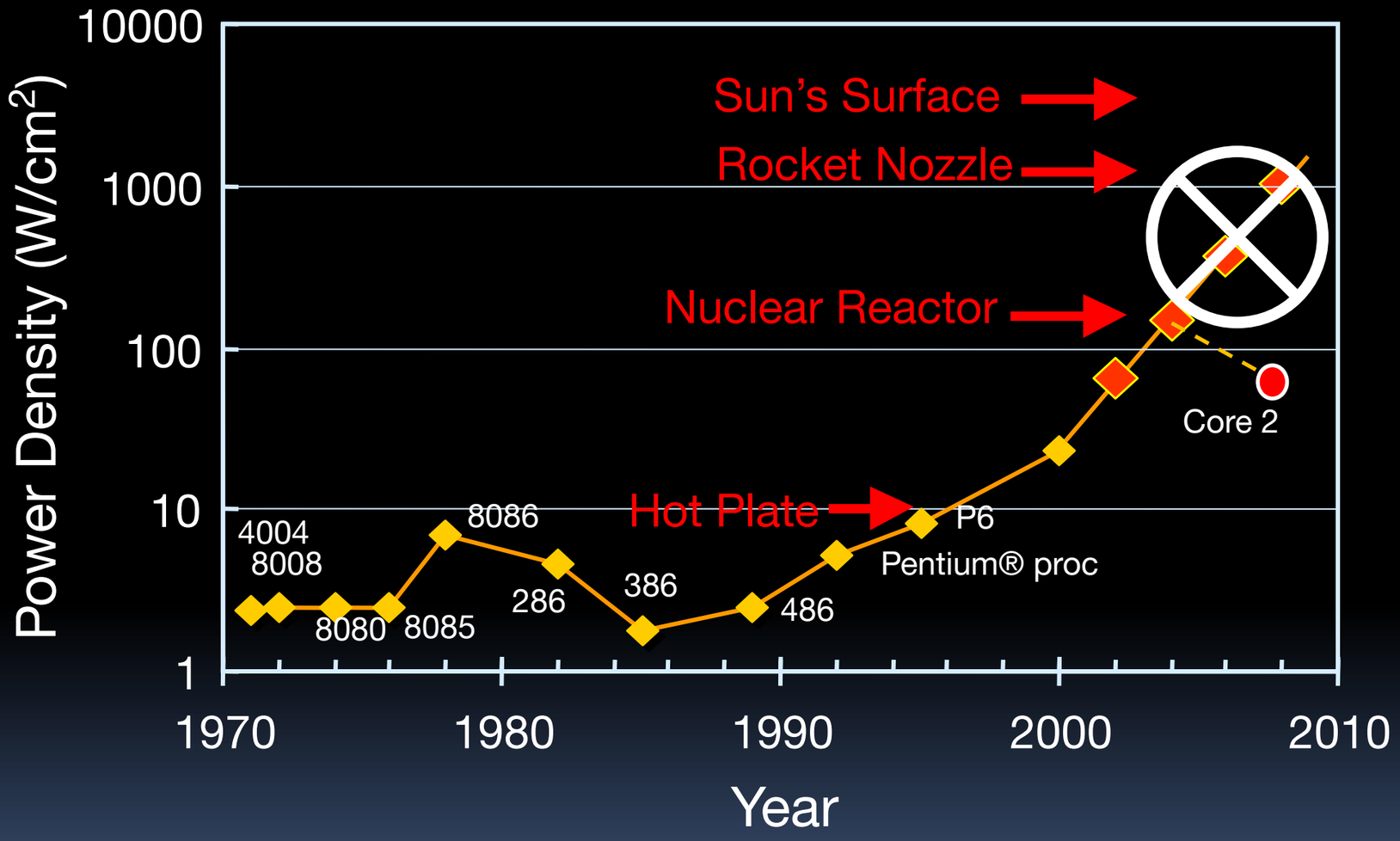


Gordon Moore
Intel Cofounder
B.S. Cal 1950!

Moore's Law and related curves

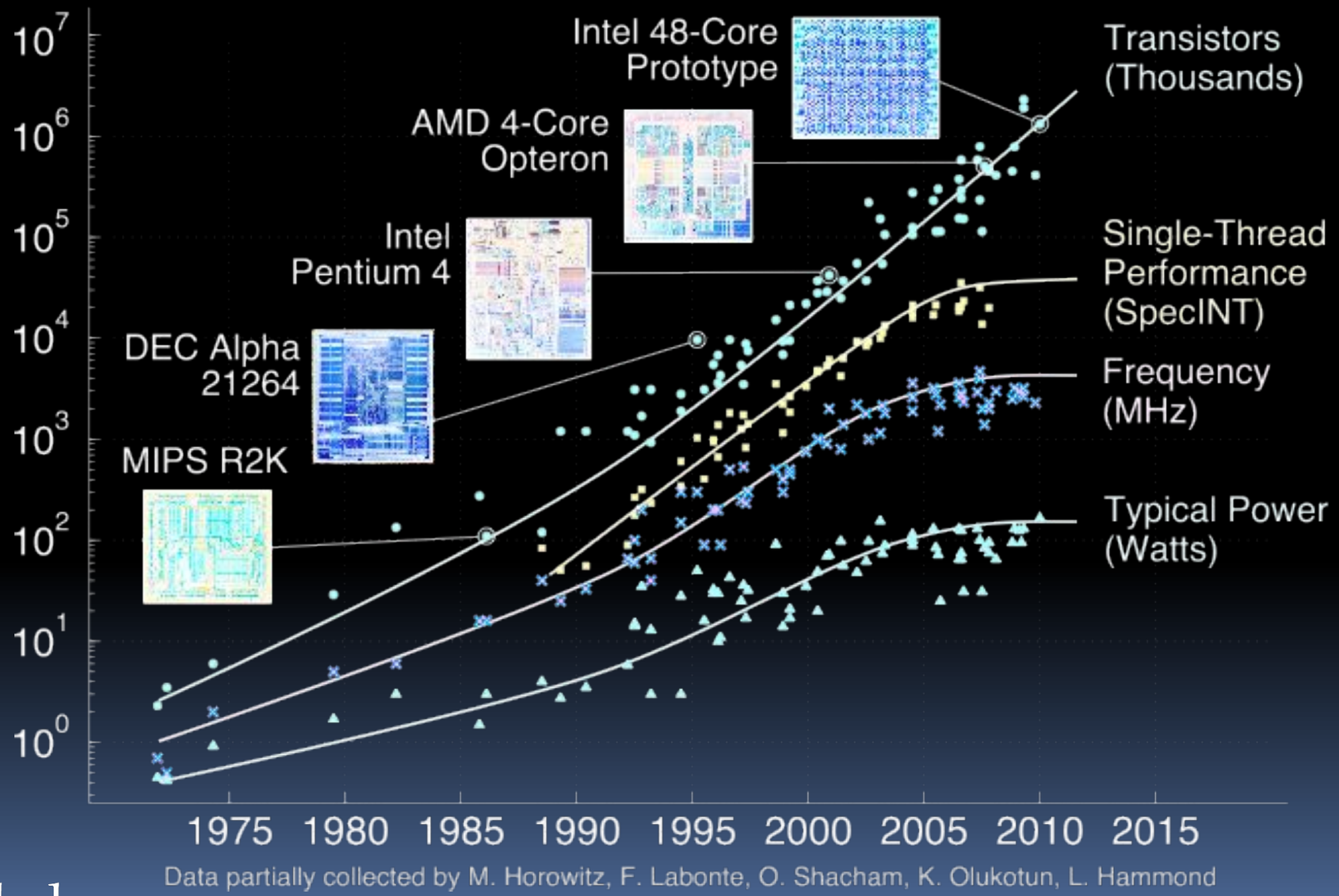


Power Density Prediction circa 2000

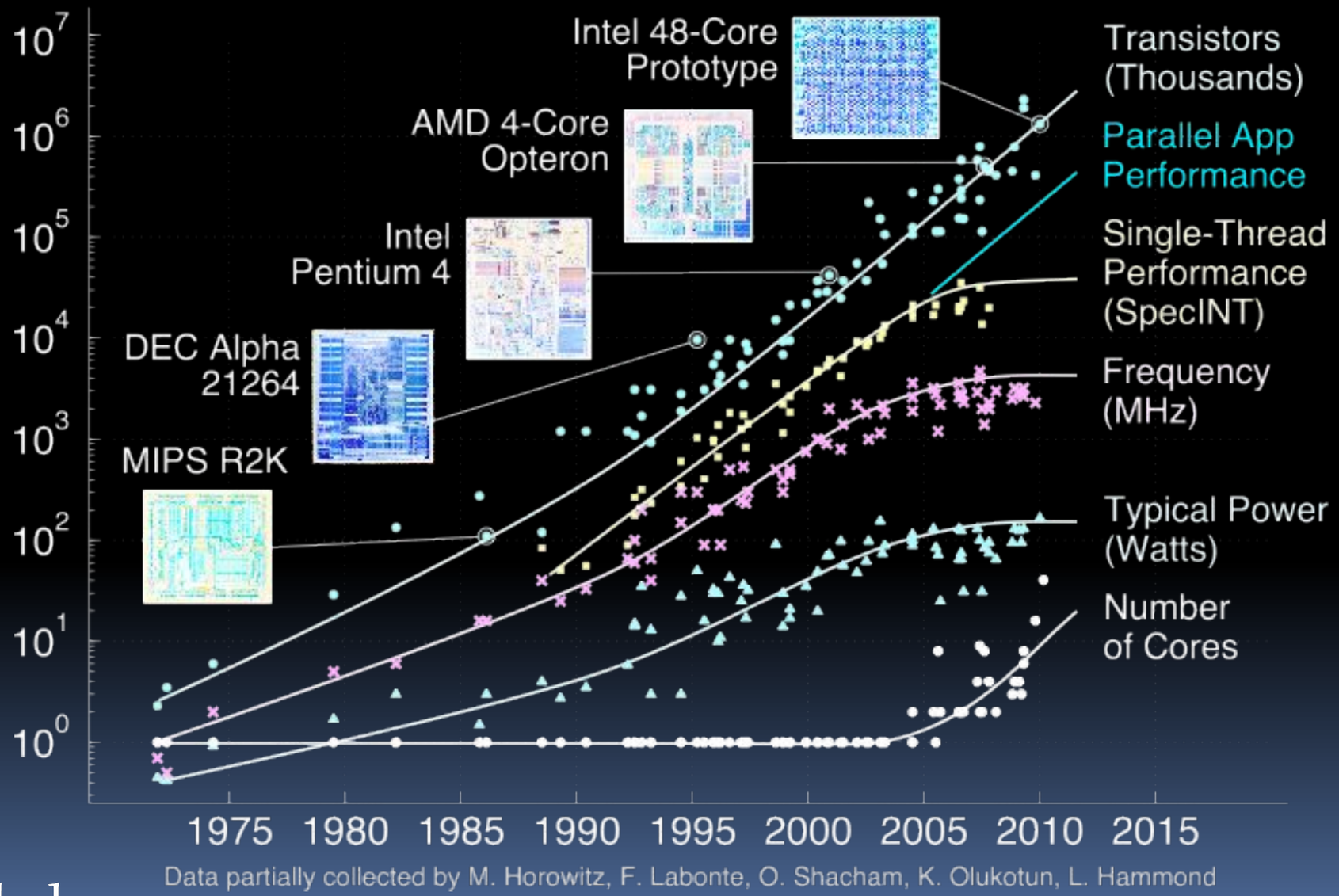


Source: S. Borkar (Intel)

Moore's Law and related curves

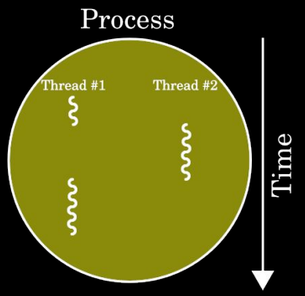


Moore's Law and related curves



Background: Threads

- A *Thread* stands for “thread of execution”, is a single stream of instructions
  - A program / process can split, or fork itself into separate threads, which can (in theory) execute simultaneously.
  - An easy way to describe/think about parallelism



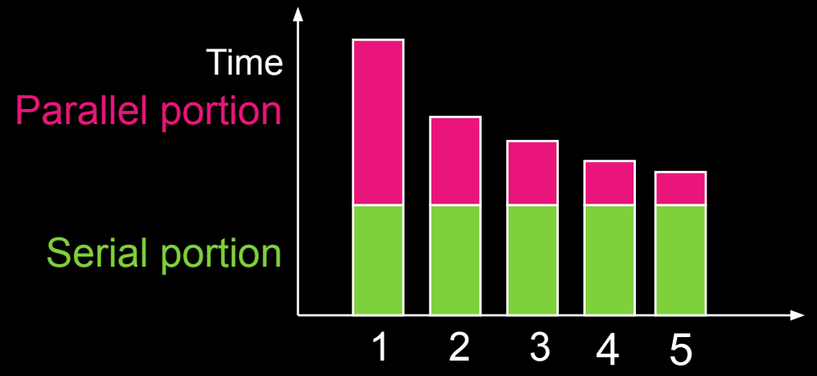
- With a single core, a single CPU can execute many threads by *Time Sharing*

- *Multithreading* is running multiple threads through the same hardware

Limitations

Speedup Issues : Amdahl's Law

- Applications can almost never be completely parallelized; some serial code remains



- **Amdahl's law:** (s is serial % of program, C is # of cores)

$$Speedup(C) = \frac{Time(1)}{Time(C)} \leq \frac{1}{s + \frac{1-s}{C}} \leq \frac{1}{s} \quad as\ C \rightarrow \infty$$

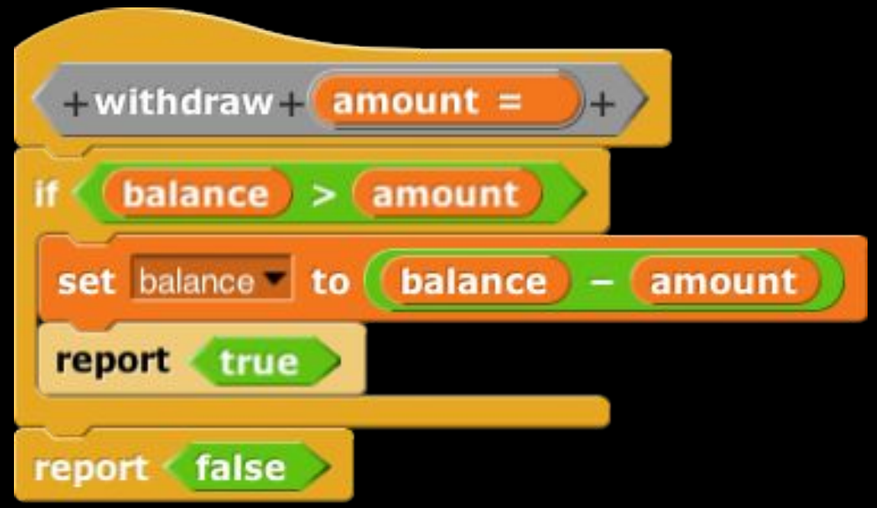
- Even if the parallel portion of your application speeds up perfectly, performance is limited by the sequential portion

Speedup Issues : Overhead

- Even assuming no sequential portion, there's…
  - Time to think how to divide the problem up
  - Time to hand out small "work units" to workers
  - All workers may not work equally fast
  - Some workers may fail
  - There may be contention for shared resources
  - Workers could overwriting each others' answers
  - You may have to wait until the last worker returns to proceed (the slowest / weakest link problem)
  - There's time to put the data back together in a way that looks as if it were done by one

Parallel Programming is Hard!

- What if two people were calling withdraw at the same time?
  - E.g., balance=100 and two withdraw 75 each
  - Can anyone see what the problem *could* be?
  - This is a race condition
- In most languages, this is a problem.
  - In Snap*!*, the system doesn't let two of these run at once.



"Non-Deterministic" Parallel Code

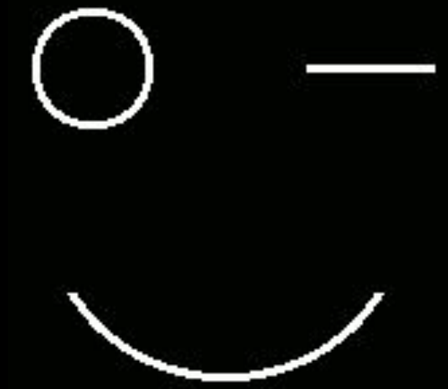
- Two (or more) scripts are running at the same time, BUT we don't know what order they will be run in!

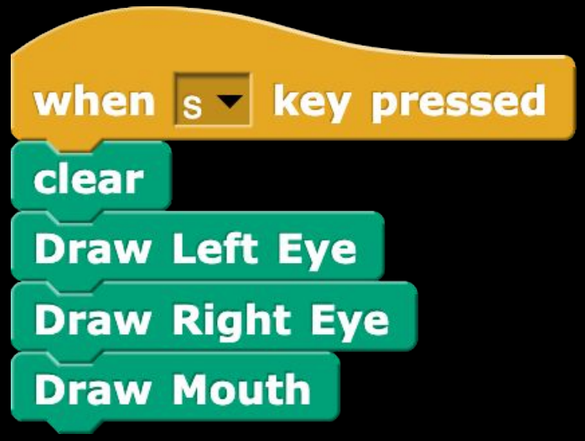
- This will not happen in Snap*!* because it will swap between running script A and script B and won't let both of them run at once. (Phew!)
  - …but if you have some explicit calls to random in your code & use these for timers, it can happen!

Race Condition Demonstration!

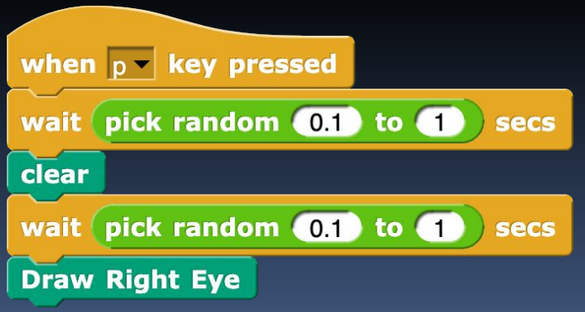
- We want this code to draw a cute winky-face, but there's a problem with parallelizing it! How many different faces could it draw?

Serial





Parallel

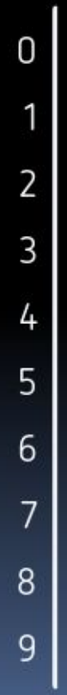


When poll is active, respond at **pollev.com/ddg**

Text **DDG** to **22333** once to join

L23

How many different faces could the parallel code draw?



Another concurrency problem…

deadlock

- Two people need to draw a graph but there is only one pencil and one ruler.
  - One grabs the pencil
  - One grabs the ruler
  - Neither release what they hold, waiting for the other to release
- Livelock also possible
  - Movement, no progress

Summary

- "Sea change" of computing because of inability to cool CPUs means we're now in multi-core world
- This brave new world offers lots of potential for innovation by computing professionals, but challenges persist