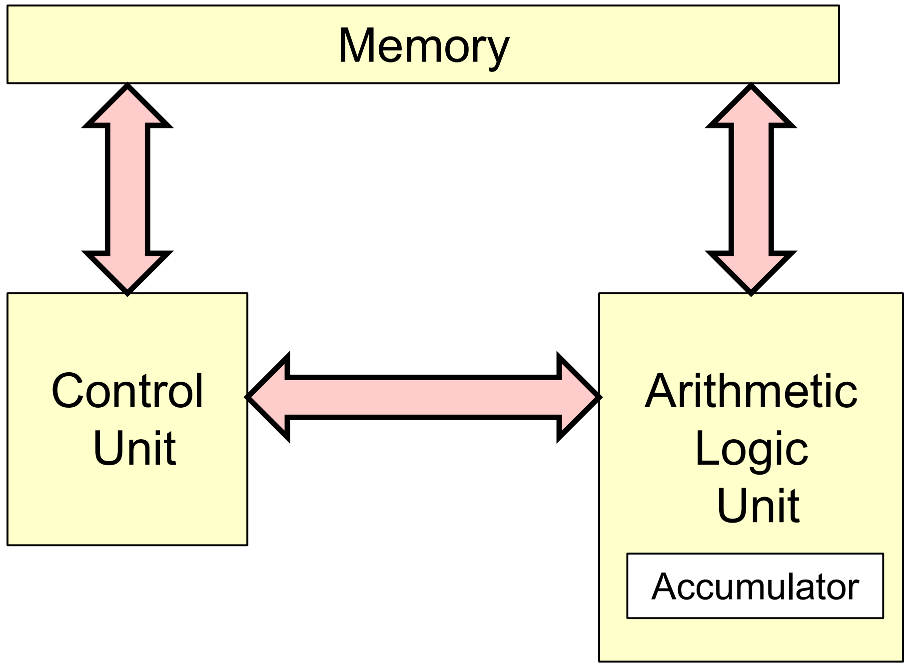
Parallel Programming:
Moore's Law and Multicore

Oregon State University

Mike Bailey

mjb@cs.oregonstate.edu

Von Neumann Architecture:
Basically the fundamental pieces of a CPU have not changed since the 1960s



Other elements:

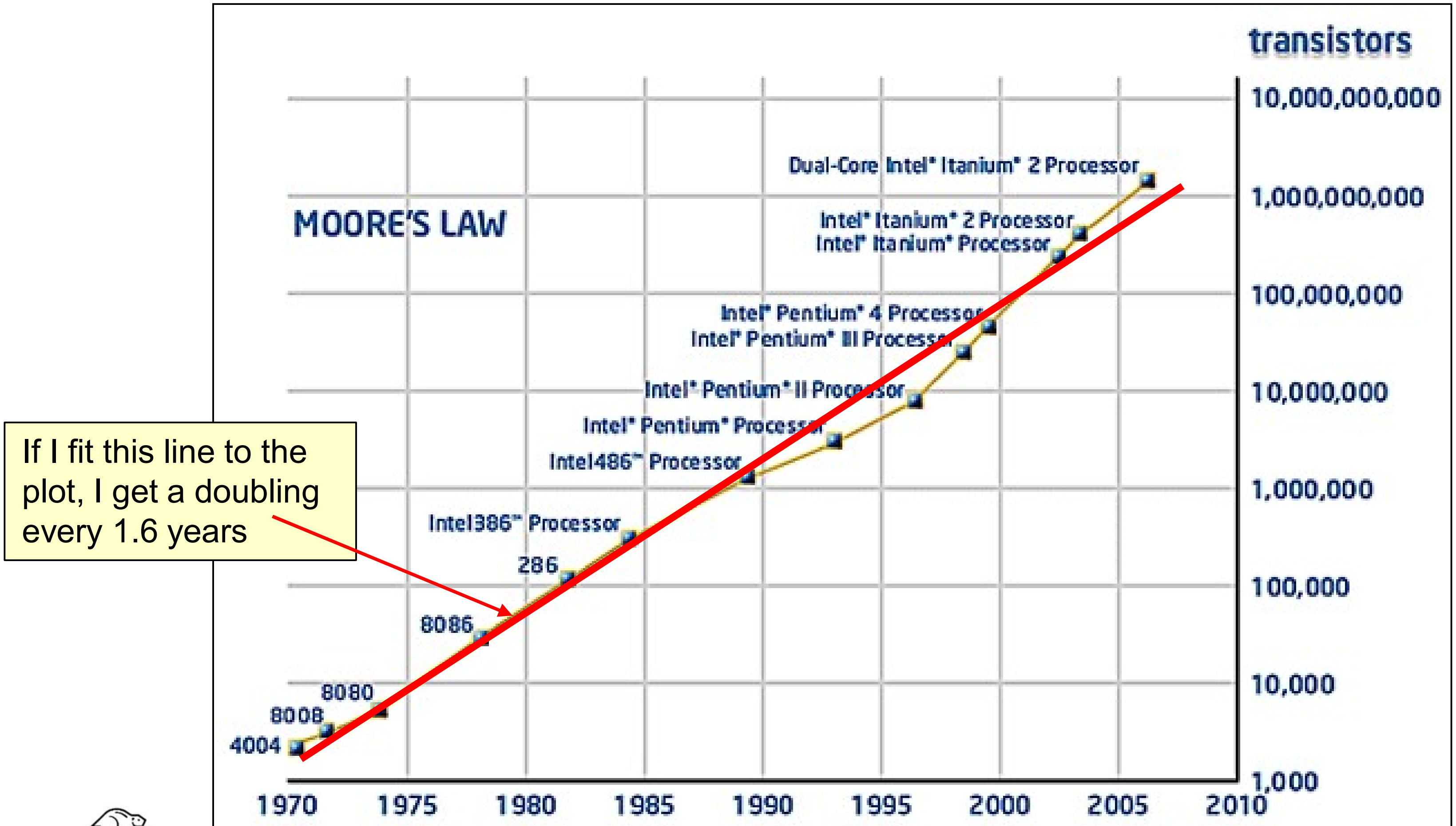
- **Clock**
- Registers
- Program counter
- Stack pointer

Oregon State University
Computer Graphics

Increasing Transistor Density -- Moore's Law

Note:
Log scale!

"Transistor density doubles every 1.5 years."

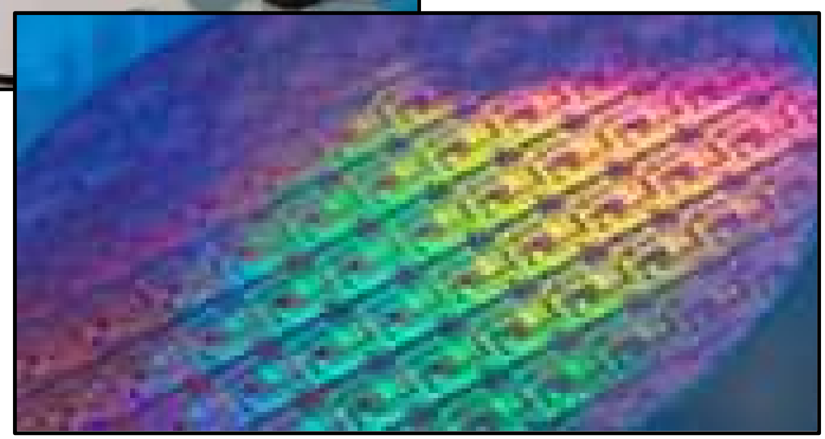


Source: http://www.intel.com/technology/mooreslaw/index.htm

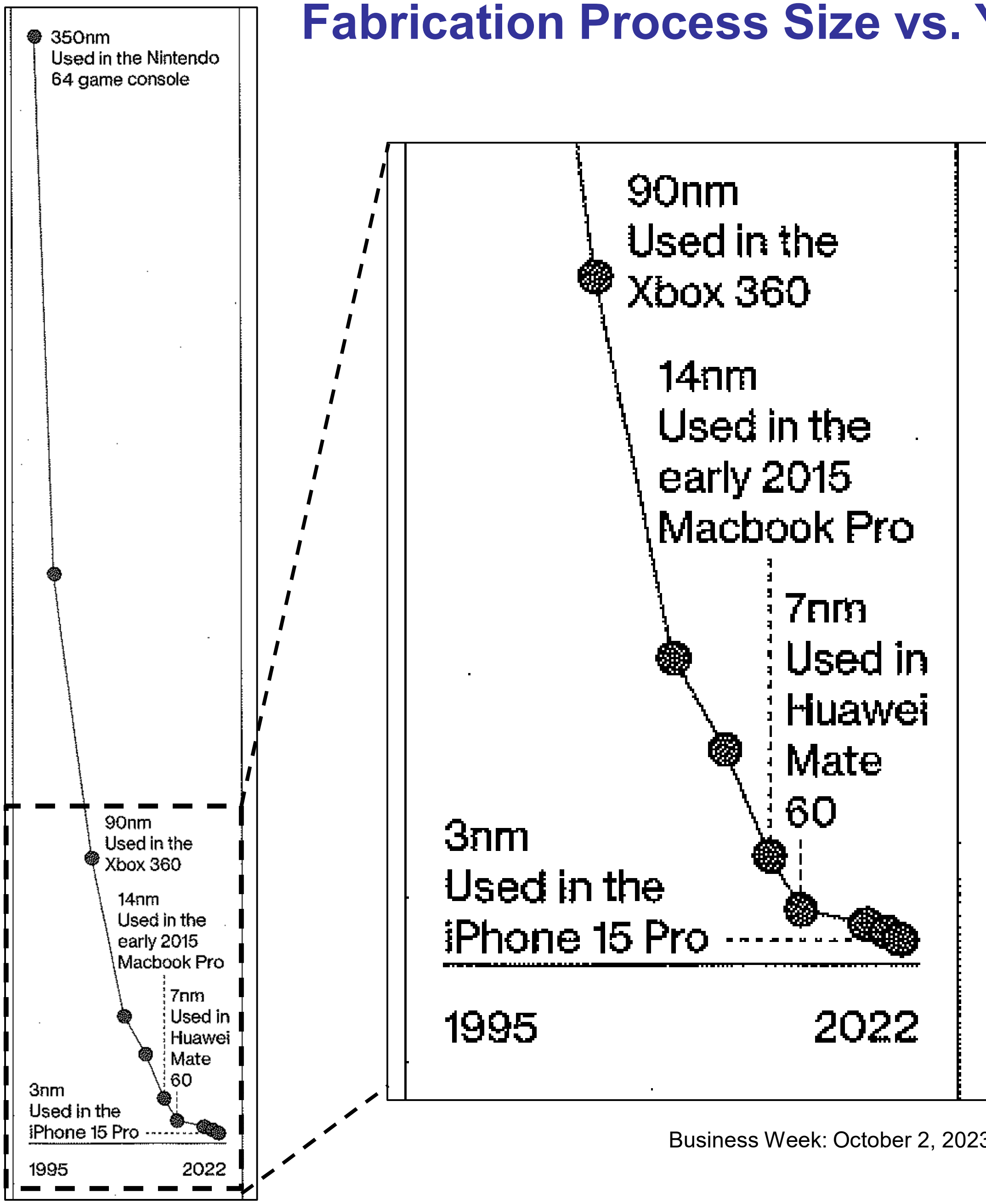
Oftentimes people have (*incorrectly*) equivalenced this to:
"Clock speed doubles every 1.5 years."

Increasing Transistor Density –
How Many Working Chips Can You fit on a Silicon Wafer?

Intel



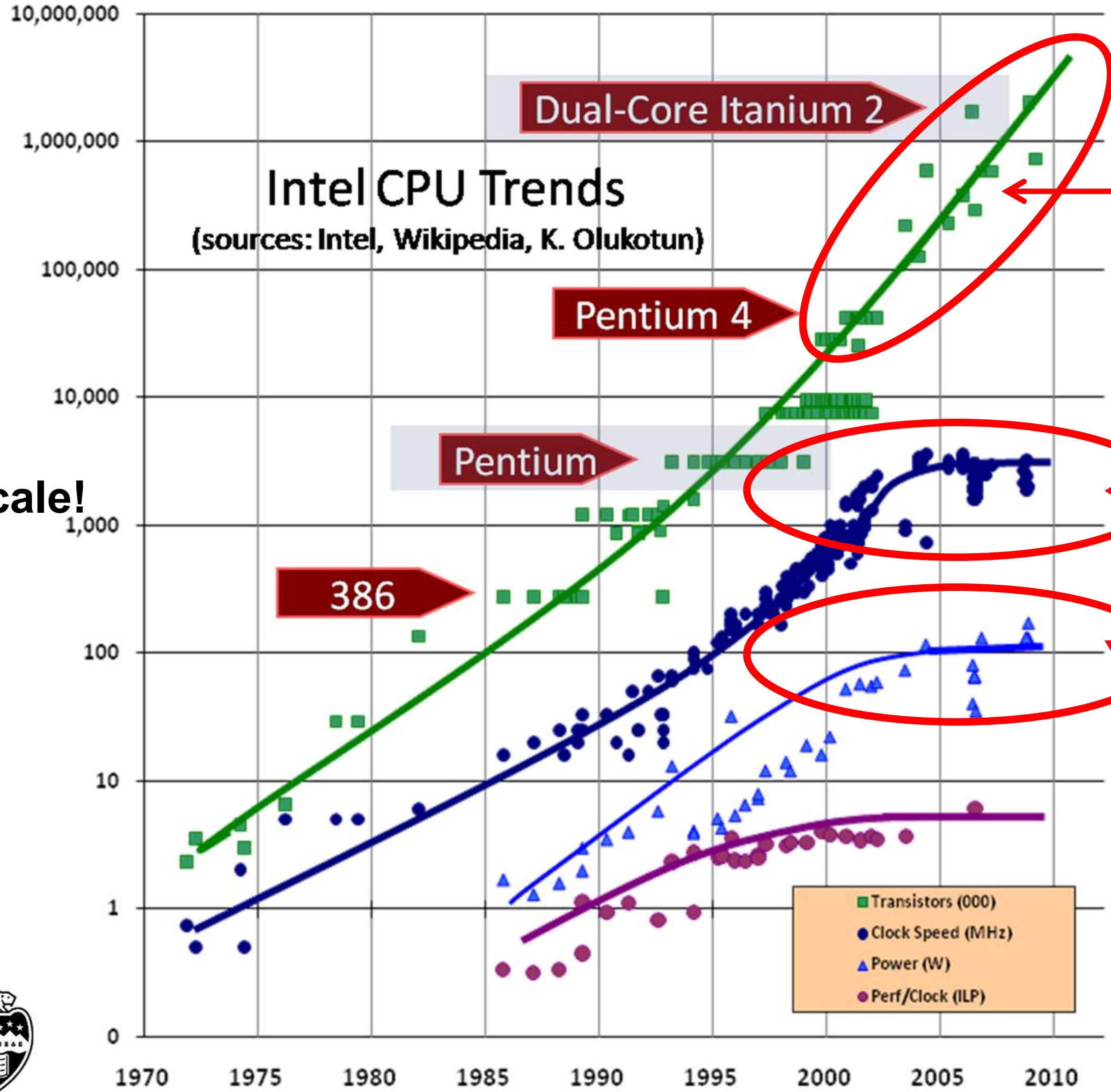
Fabrication Process Size vs. Year



Business Week: October 2, 2023

Increasing Clock Speed?

Note:
Log scale!



Transistor count

This is what Moore's Law really deals with!

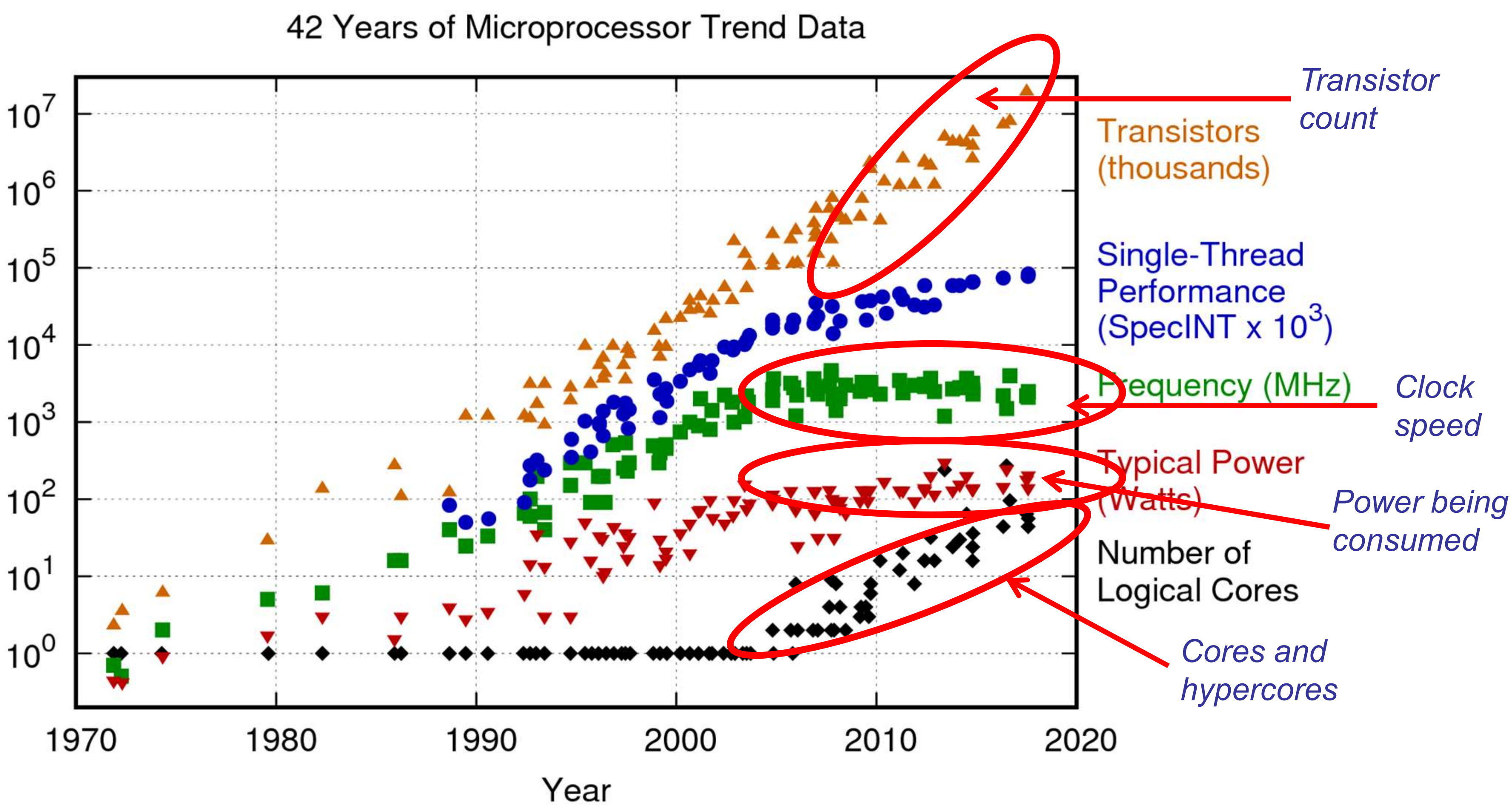
Clock speed

This is *not* what Moore's Law really deals with!

Power being consumed

Intel

Increasing Clock Speed?



Original data up to the year 2010 collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond, and C. Batten
New plot and data collected for 2010-2017 by K. Rupp

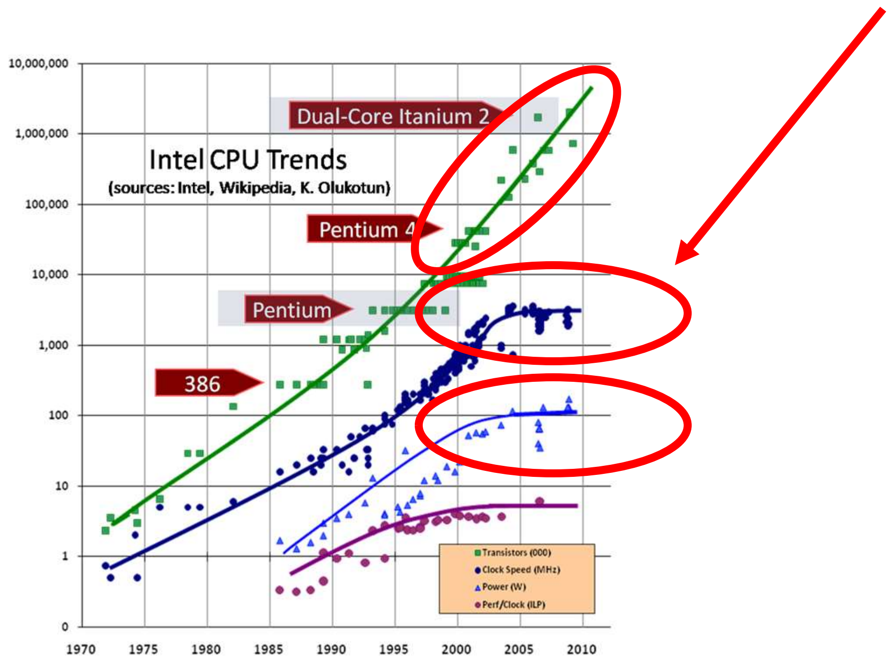
Oregon State University
Computer Graphics

Source: Karl Rupp

Moore's Law

- Fabrication process size ("gate pitch") has fallen from 65 nm, to 45 nm, to 32 nm, to 22 nm, to 16 nm, to 11 nm, to 8 nm. This translates to more transistors on the same size die.

- From 1986 to 2002, processor performance increased an average of 52%/year, but then virtually plateaued.



Oregon State University Computer Graphics

Clock Speed and Power Consumption

1981	IBM PC	5 MHz
1995	Pentium	100 MHz
2002	Pentium 4	3000 MHz (3 GHz)
2007		3800 MHz (3.8 GHz)
2009		4000 MHz (4.0 GHz)

Clock speed has hit a plateau, largely because of power consumption and power dissipation.

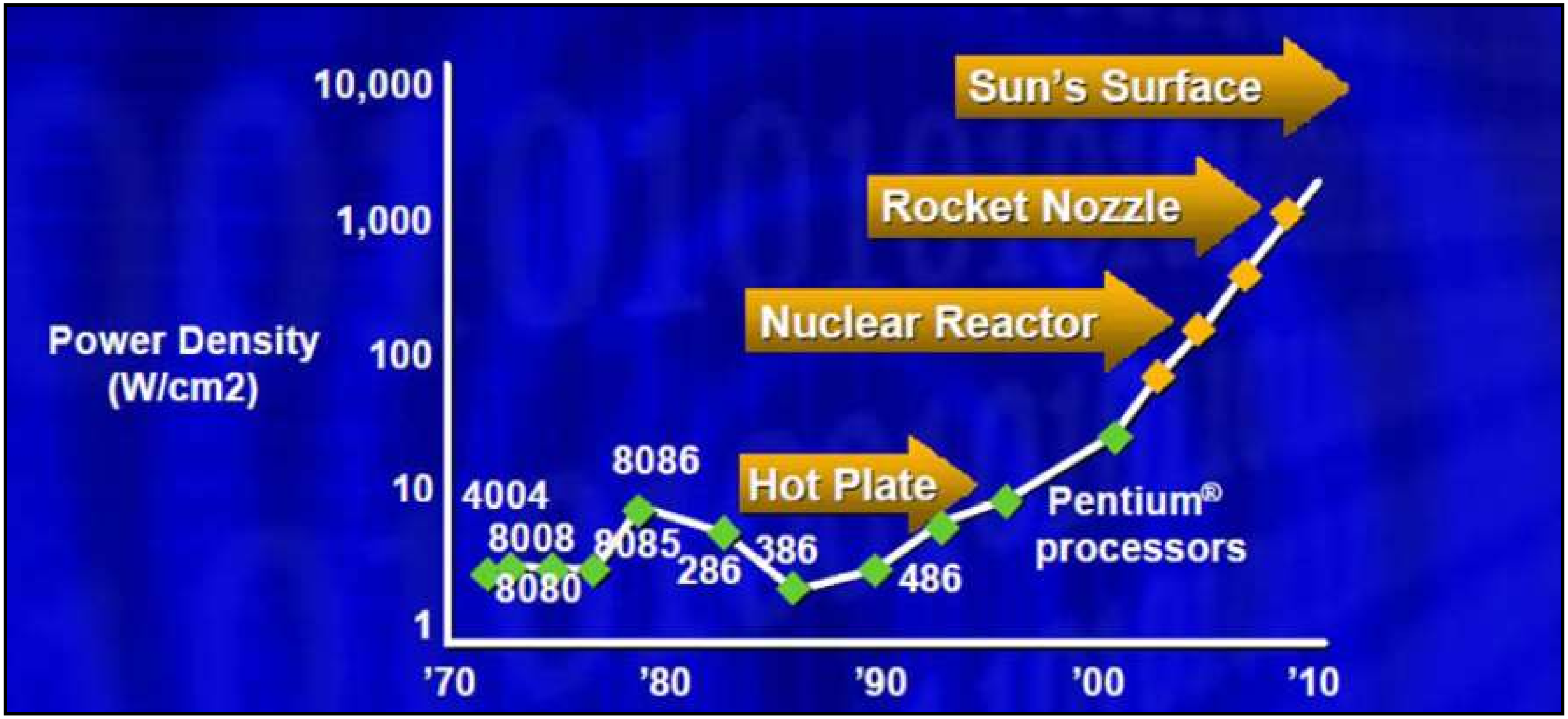
$$PowerConsumption \propto ClockSpeed^2$$

Yikes!

is-proportional-to

Once consumed, that power becomes *heat*, which much be *dissipated* somehow. In general, compute systems can remove around 150 $^{watts}/_{cm^2}$ without resorting to exotic cooling methods.

What Kind of Power Density Dissipation Would it Have Taken to Keep up with Clock Speed Trends?



Intel

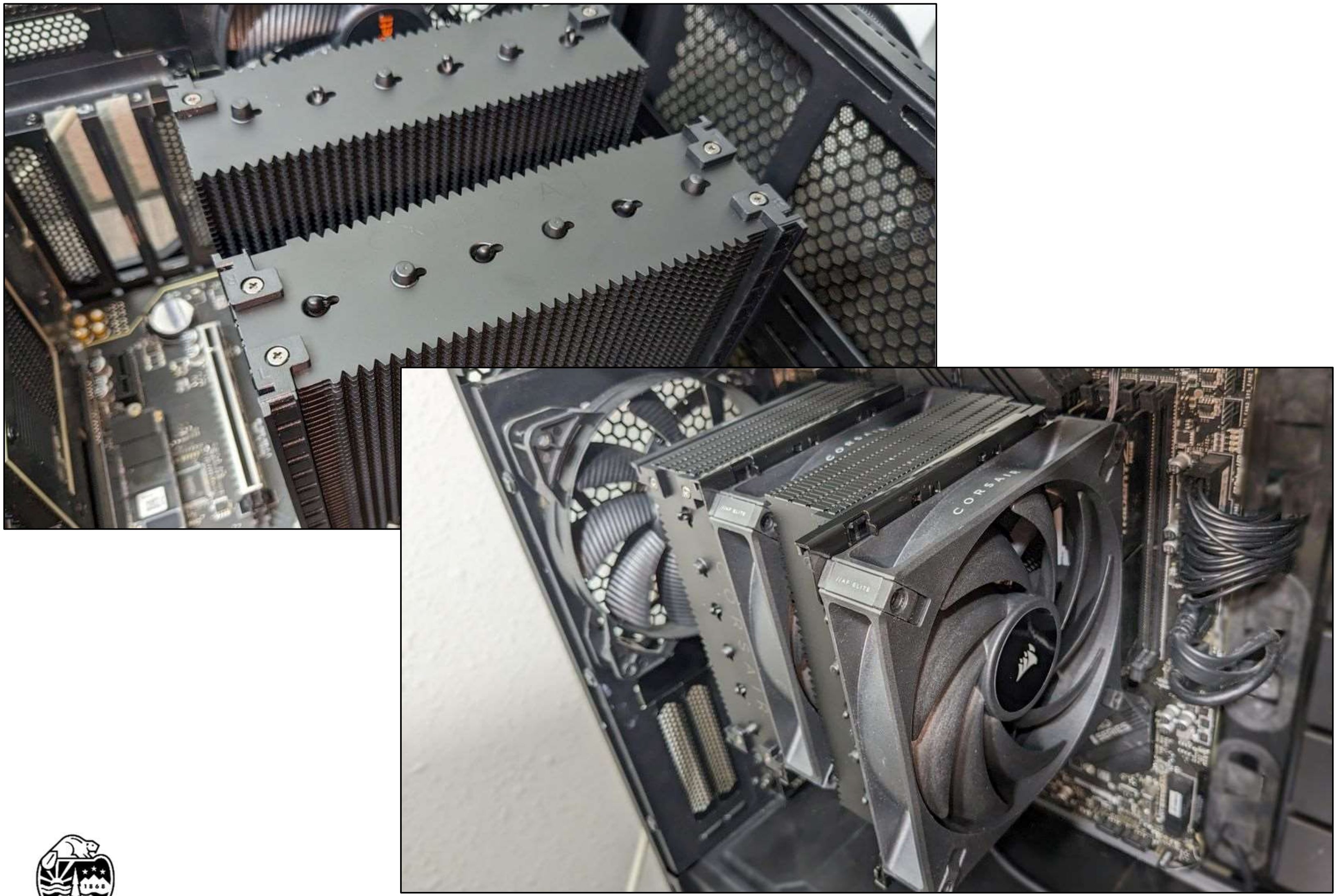
Oregon State University
Computer Graphics

And speaking of "exotic methods", Intel currently holds the record for Clock Speed: 9.117 GHz

Tom's Hardware

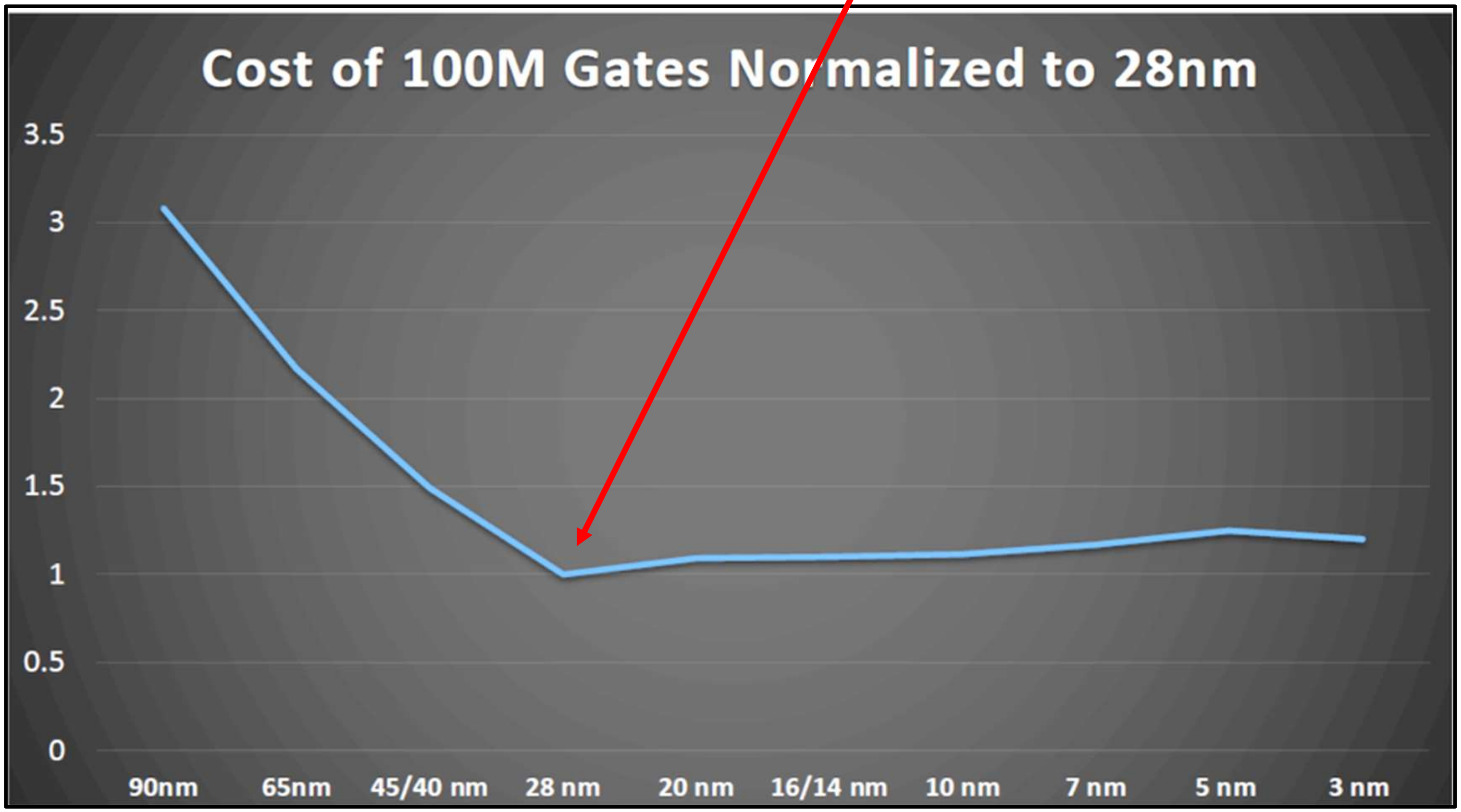
Cooled with liquid helium (-231 ºC = -394 ºF).
They had to use liquid helium because liquid nitrogen wasn't cold enough. Wow.

There Are More Sensible Cooling Solutions



Corsair's A115 Dual-Tower Air Cooler

Interestingly, this Size Reduction was Resulting in Cheaper Chips until the Fab Size Hit 28nm



Toms Hardware: February 3, 2024

This has nothing to do with our Moore's Law discussion, I just found it interesting…

MultiCore -- Multiprocessing on a Single Chip

So, to summarize:

Moore's Law of transistor density is still going, but the "Moore's Law" of clock speed has hit a wall. Now what do we do?

We keep packing more and more transistors on a single chip, but don't increase the clock speed. Instead, we increase computational throughput by using those transistors to pack multiple processors onto the same chip.

This is referred to as ***multicore***.

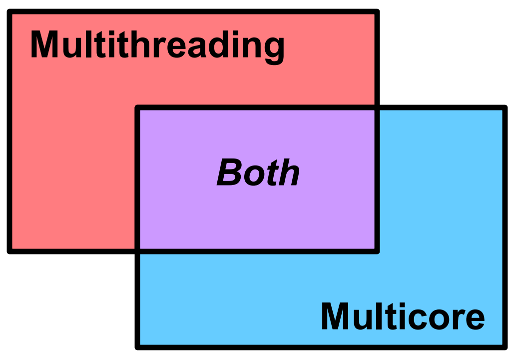
Vendors have also reacted by adding SIMD floating-point units on the chip as well. We will get to that later.

MultiCore and Multithreading

Multicore, even without multithreading too, is still a good thing. It can be used, for example, to allow multiple programs on a desktop system to always be executing concurrently.

Multithreading, even without multicore too, is still a good thing. Threads can make it easier to logically have many things going on in your program at a time, and can absorb the dead-time of other threads.

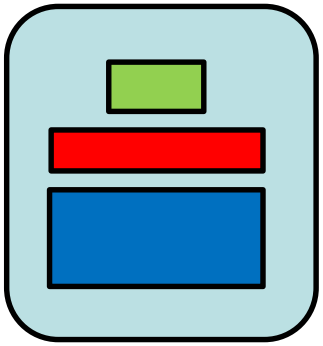
But, the big gain in performance is to use *both* to speed up a *single program*. For this, we need *a **combination of both multicore and multithreading***.



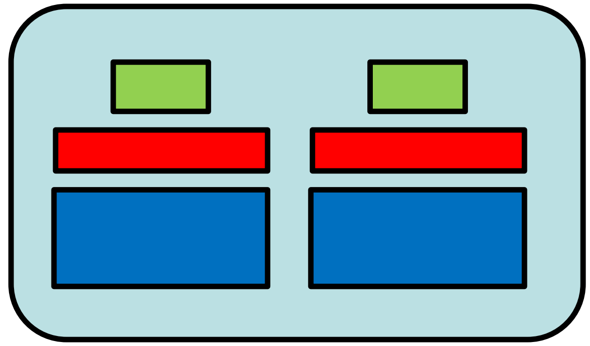
Multicore is a very hot topic these days. It would be hard to buy a CPU that doesn't have more than one core. We, as programmers, get to take advantage of that.

We need to be prepared to convert our programs to run on ***MultiThreaded Shared Memory Multicore*** architectures.

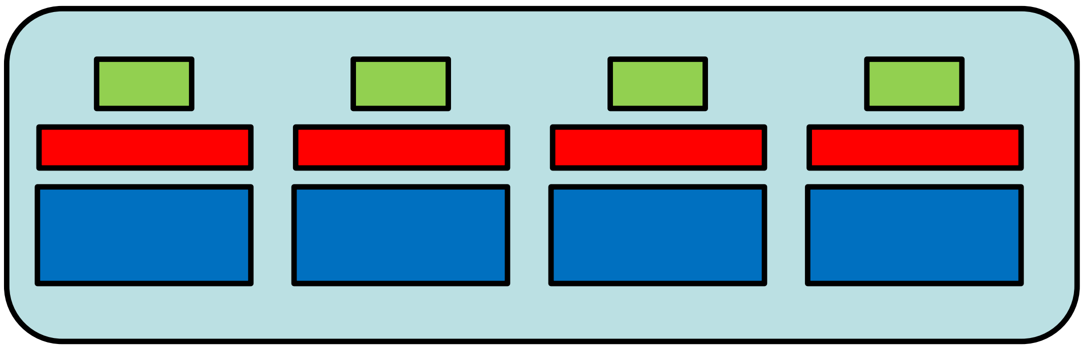
Each of the Multiple Cores keeps its own State



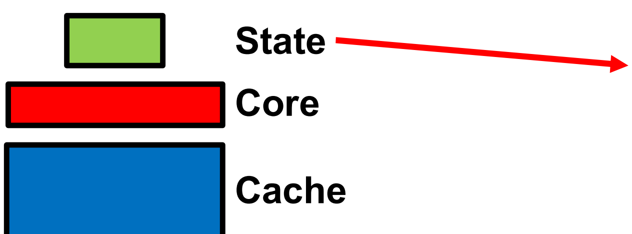
1 core, 1 state



2 cores, 2 states

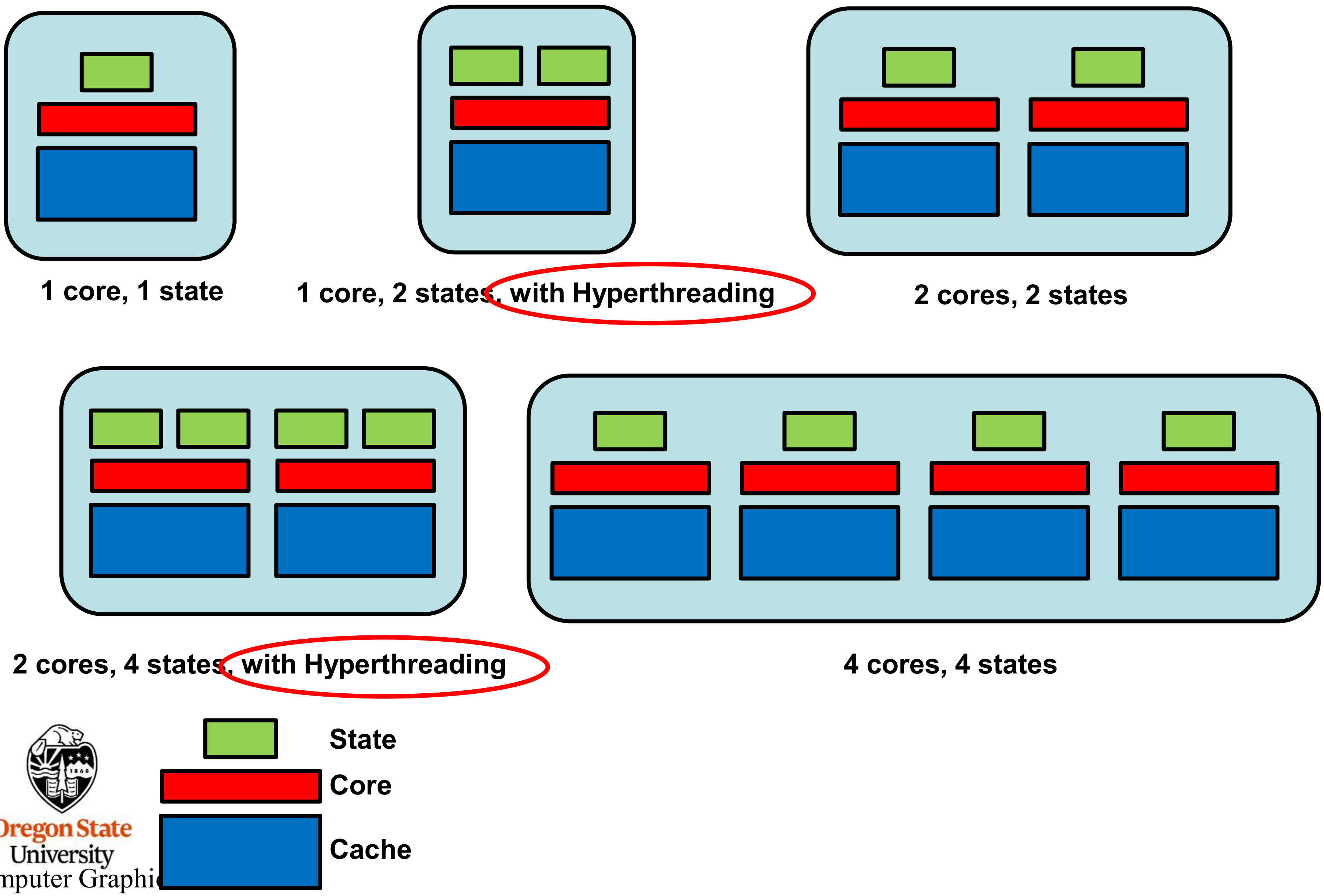


4 cores, 4 states

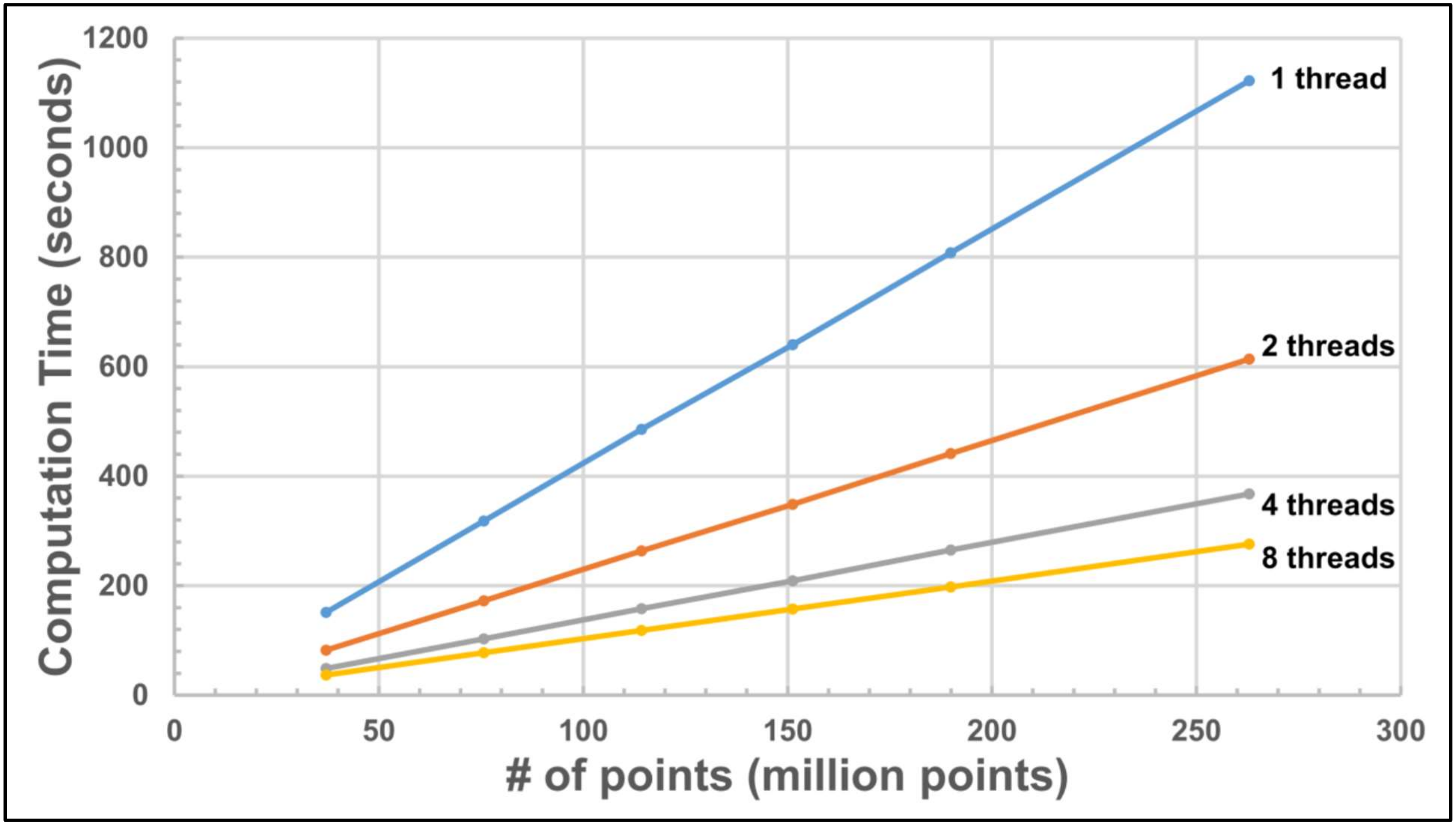


- Registers
- Program Counter
- Stack Pointer

So, if that's what Multicore is about, what is *Hyperthreading*?



Four Cores with Two Hyperthreads per Core



Source: Erzhuo Che

Note that this is upside-down from our usual convention. Sorry. I got this from someone else.