Parallel Programming: Moore’s Law and Multicore

Mike Bailey
mjb@cs.oregonstate.edu

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License
Von Neumann Architecture:
Basically the fundamental pieces of a CPU have not changed since the 1960s

Other elements:
- Clock
- Registers
- Program counter
- Stack pointer
Increasing Transistor Density -- Moore’s Law

“Transistor density doubles every 1.5 years.”

Note: Log scale!

If I fit this line to the plot, I get a doubling every 1.6 years


Oftentimes people have (incorrectly) equivalenced this to: “Clock speed doubles every 1.5 years.”
Increasing Clock Speed?

Intel CPU Trends
(sources: Intel, Wikipedia, K. Olukotun)

Note:
Log scale!

Transistor count

Clock speed

Power being consumed

This is what Moore’s Law really deals with!

This is not what Moore’s Law really deals with!
Increasing Clock Speed?

42 Years of Microprocessor Trend Data

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

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

\[ \text{PowerConsumption} \propto \text{ClockSpeed}^2 \]

Yikes!

Once consumed, that power becomes *heat*, which much be dissipated somehow. In general, compute systems can remove around 150 watts/cm without resorting to exotic cooling methods.
And, speaking of “exotic”, AMD set the world record for clock speed (8.429 GHz) using a Liquid Nitrogen-cooled CPU.

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

Source: Intel
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, 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*?

- 1 core, 1 state
- 1 core, 2 states, with Hyperthreading
- 2 cores, 2 states
- 2 cores, 4 states, with Hyperthreading
- 4 cores, 4 states
Four Cores with Two Hyperthreads per Core

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