CS598 LVK
Parallel Programming with Migratable Objects

Laxmikant (Sanjay) Kale

http://charm.cs.illinois.edu

Parallel Programming Laboratory

Department of Computer Science

University of Illinois at Urbana Champaign

ILLINOIS

UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

PARALLEL PROGRAMMING LAB

DEPT. OF COMPUTER SCIENCE, UNIVERSITY OF ILLINOIS

PPL UIUC

Special topics class

- We have an unexpectedly (but pleasantly) large class size
- No Teaching Assistants!

- A lot of responsibility for learning on you
- Also: it is a class where you learn by doing

Class Format

- Lectures
- Reading papers or chapters
- Homeworks
- Programming assignments: 5-6 tentative
- Semester project: groups of 4 students
- A midterm exam, but with a small fraction of grade; no final exam
- *Tentative* breakdown:

Homework	Program. Assignments	Midterm Exam	Project
15%	30%	15%	40%

Computers

- EWS workstations to begin with
- Later: Taub cluster
- (Hopes for Blue Waters are dashed with expected availability being probably too late for the class).
- Taub is large enough

Course organization

- On the web:
- Engineering wiki page will be set up
  - https://wiki.engr.illinois.edu/display/cs598lvk
- Backup page:
  - http://courses.engr.illinois.edu/cs598lvk
- Lecture notes
  - (always pdf, some also in ppt) will be posted online
- Homework assignments, updates: online
- Newsgroup: class.fa12.cs598lvk hosted at news.cs.illinois.edu
- You are responsible for checking these regularly

The Changing Landscape in Computing

Laxmikant (Sanjay) Kale
http://charm.cs.illinois.edu
Parallel Programming Laboratory
Department of Computer Science
University of Illinois at Urbana Champaign

ILLINOIS
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

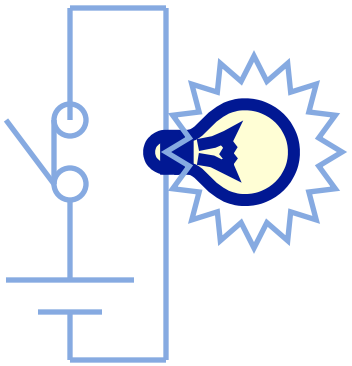
PARALLEL PROGRAMMING LAB
DEPT. OF COMPUTER SCIENCE, UNIVERSITY OF ILLINOIS

Computers

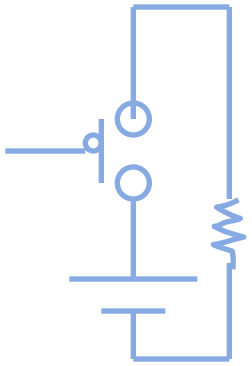
- We have been able to make a "Machine" that can do complex things
  - Add and multiply *really* fast
  - Weather forecast, design of medicinal drugs
  - Speech recognition, Robotics, Artificial Intelligence..
  - Web browsers, internet communication protocols
- What is this machine based on?

The Modest Switch

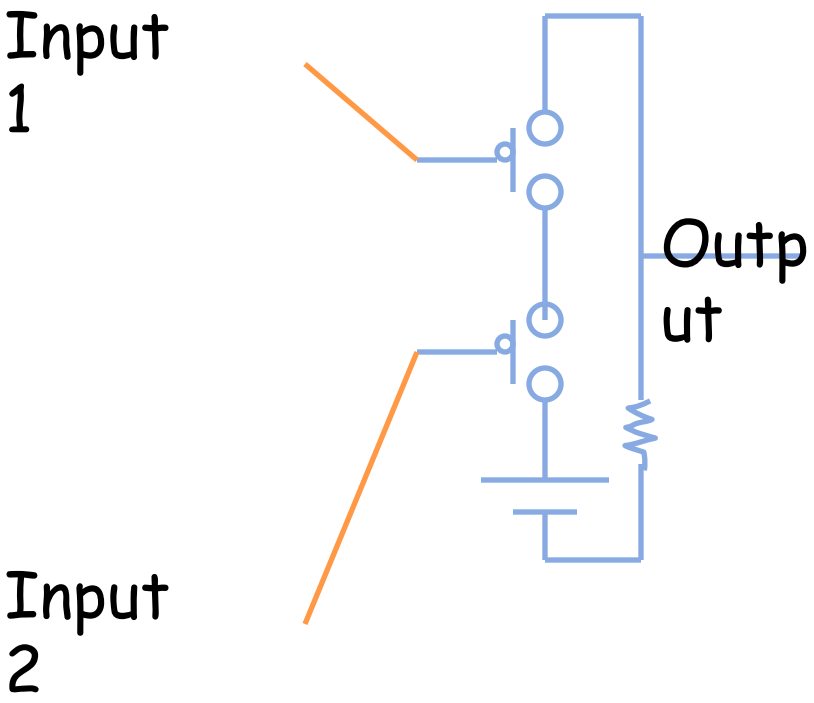
- All these capabilities are built from an extremely simple component:
  - A controllable *switch*
- The usual Electrical switch we use every day
  - The electric switch we use turns current on and off
  - But we need to turn it on and off by hand
  - The result of turning the switch on?
    - The "top end" in the figure becomes
    - raised to a high voltage
    - Which makes the current flow through the bulb



- The Controllable Switch
  - No hands
  - Voltage controls if the switch is on or off
  - High voltage at input: switch on
    - Otherwise it is off



Lets use them creatively



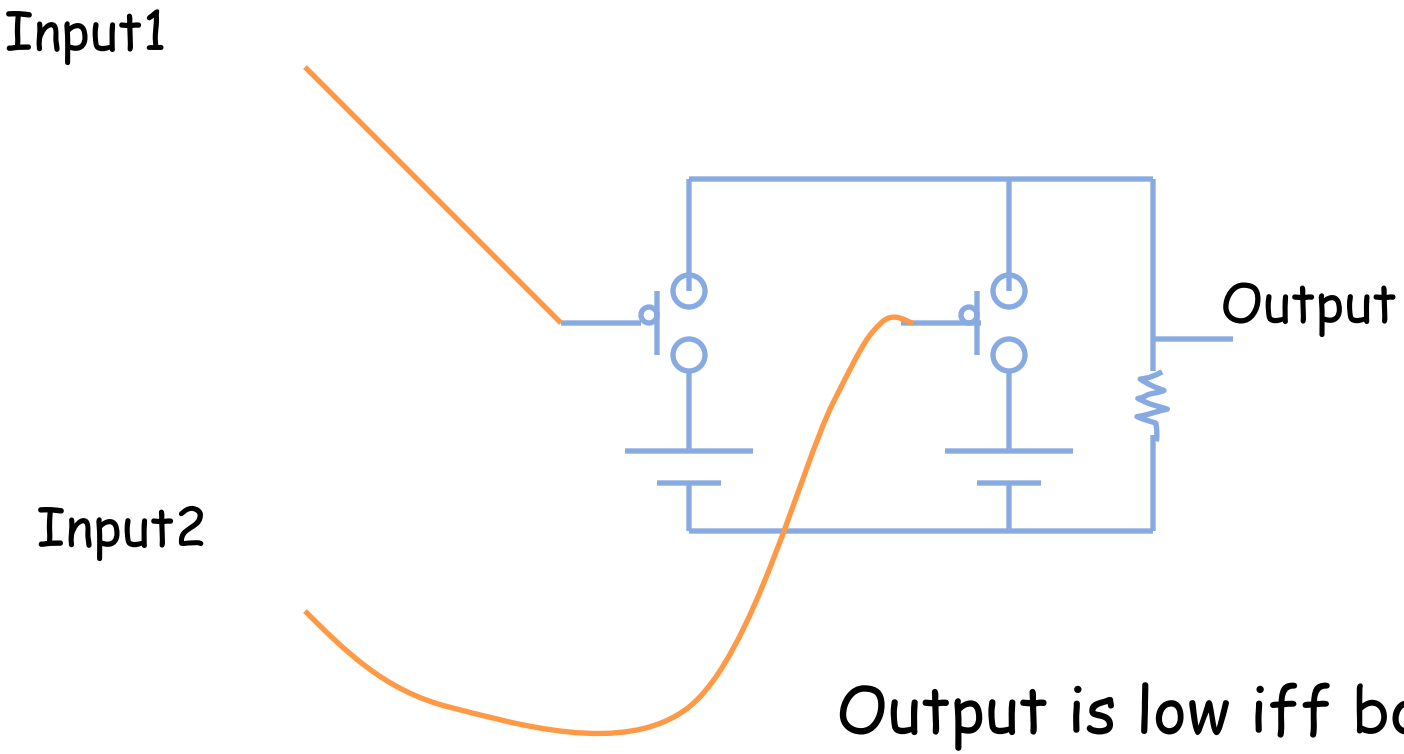
Output is high if both the inputs input1 AND input2 are high

If either of the inputs is low, the output is low.

This is called an AND gate

Now, can you make an OR gate with switches?

OR Gate

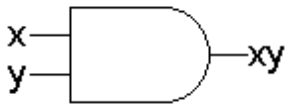
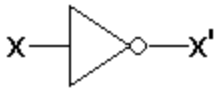


Output is low iff both inputs are low

I.e. Output is high if either of the inputs (or both) are high (input1 *OR* input2)

Basic Gates

- There are three basic kinds of logic gates

Operation:	AND of two inputs	OR of two inputs	NOT (complement) on one input
Logic gate:			

- Two Questions:
  - How can we implement such switches?
  - What can we build with Gates?
    - Adders, controllers, memory elements, computers!

How to make switches?

- Use mechanical power
- Use hydrolic pressure
- Use electromechanical switches (electromagnet turns the switch on)
- Current technology:
  - Semiconductor transistors
    - A transistor can be made to conduct electricity depending on the input on the 3rd input
  - CMOS "gates" (actually, switches)

Two properties of Switches and Gates:

Size

Switching and Propagation delay

Clock Speeds

- Since we can make transistors smaller
  - Which means smaller capacitances..
    - Imagine filling up "tanks" with "water" (electrons)
- We can turn them on or off faster
  - Which means we can make our computers go faster
  - Clock cycle is selected so that the parts of the computer can finish basic calculations within the cycle
  - And indeed:

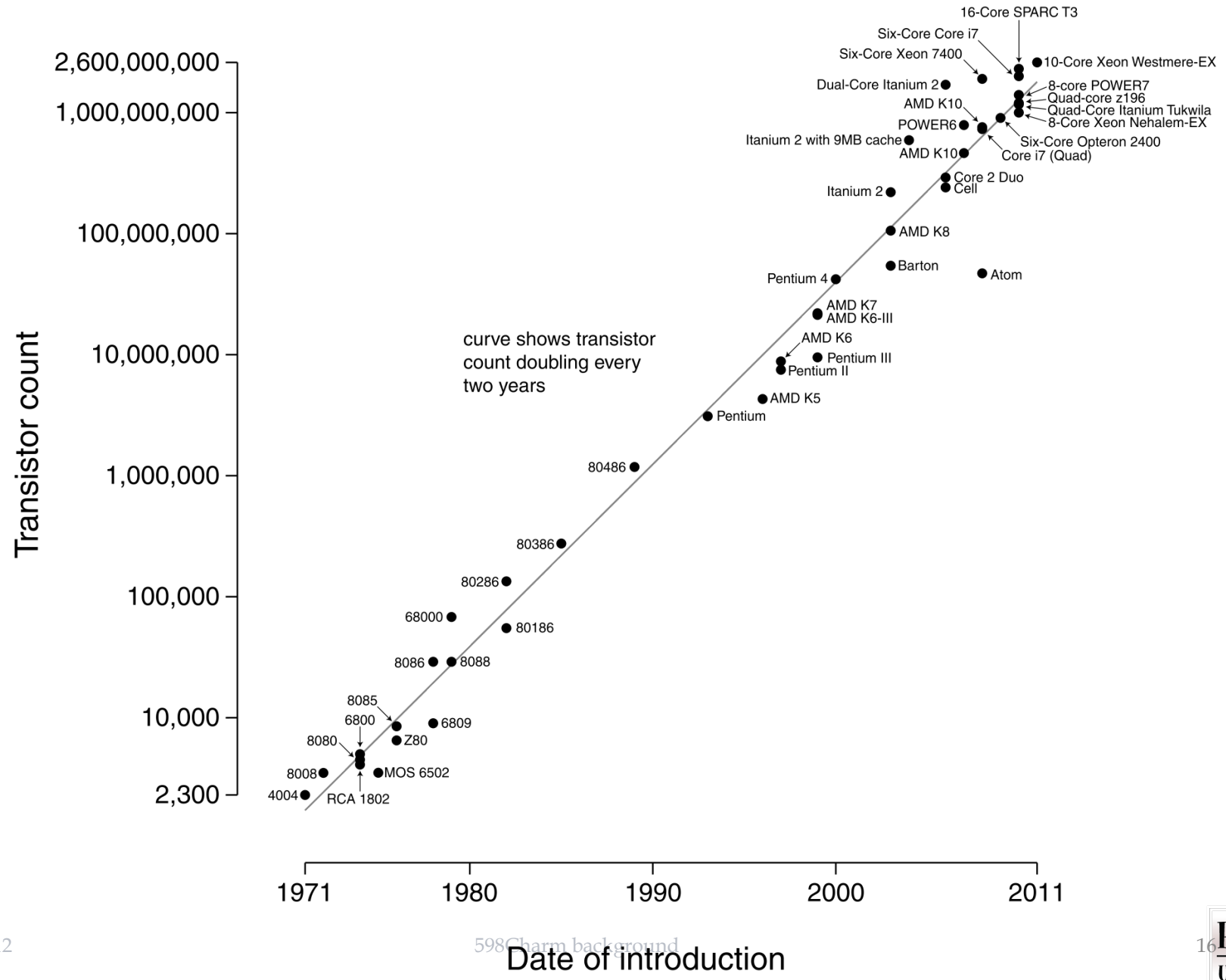
The Virtuous Cycle

- If you can make transistors smaller,
  - You can fit more of them on a chip
    - Cost per transistor decreases
  - AND: propagation delays get smaller
    - So they can run faster!
- Can you make them smaller?
  - Technological progress needed, but can be done
- This led to:
  - Cheaper and faster processors every year

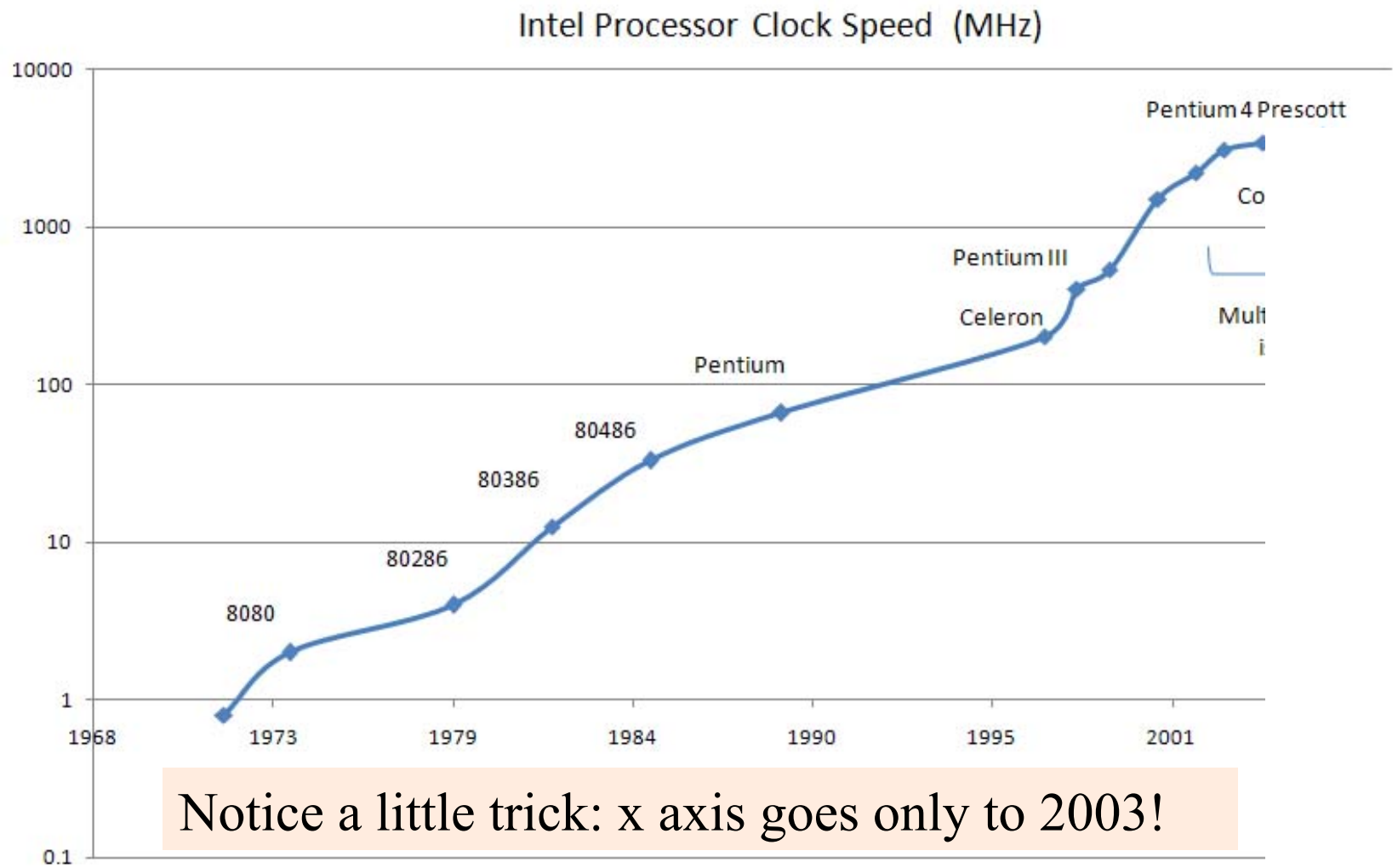
Moore's law

- Commonly (mis) stated as
  - "Computer Performance doubles every 18 months"
- Gordon Moore observed in 1965
  - "The complexity… has increased roughly a factor of two per year. [It] can be expected to continue… for at least 10 years"
  - Its about ***number of transistors*** per chip
- Funny thing is: it held true for 40+ years
  - And still going until 2020
  - "Self Fulfilling prophecy"

Microprocessor Transistor Counts 1971-2011 & Moore's Law

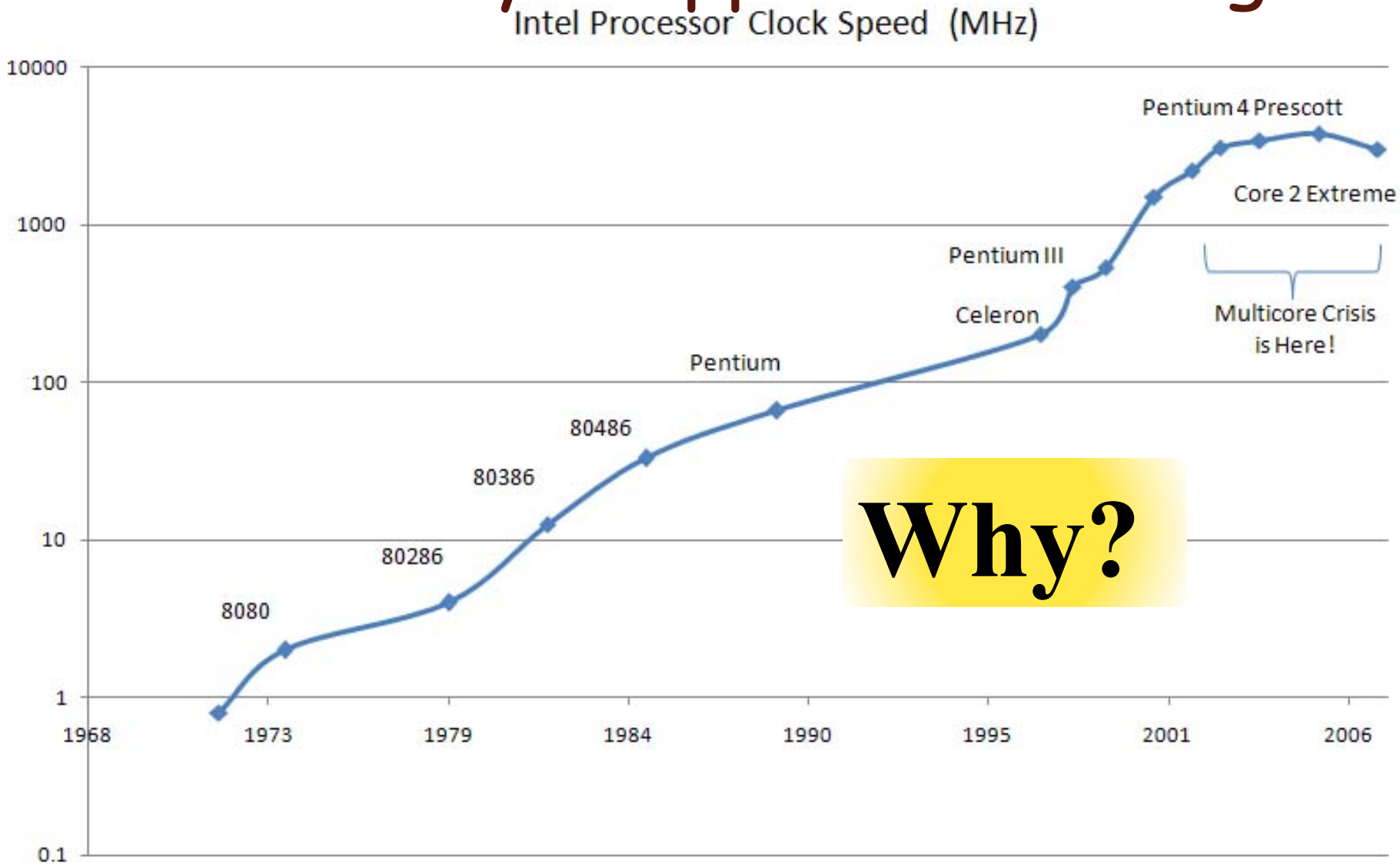


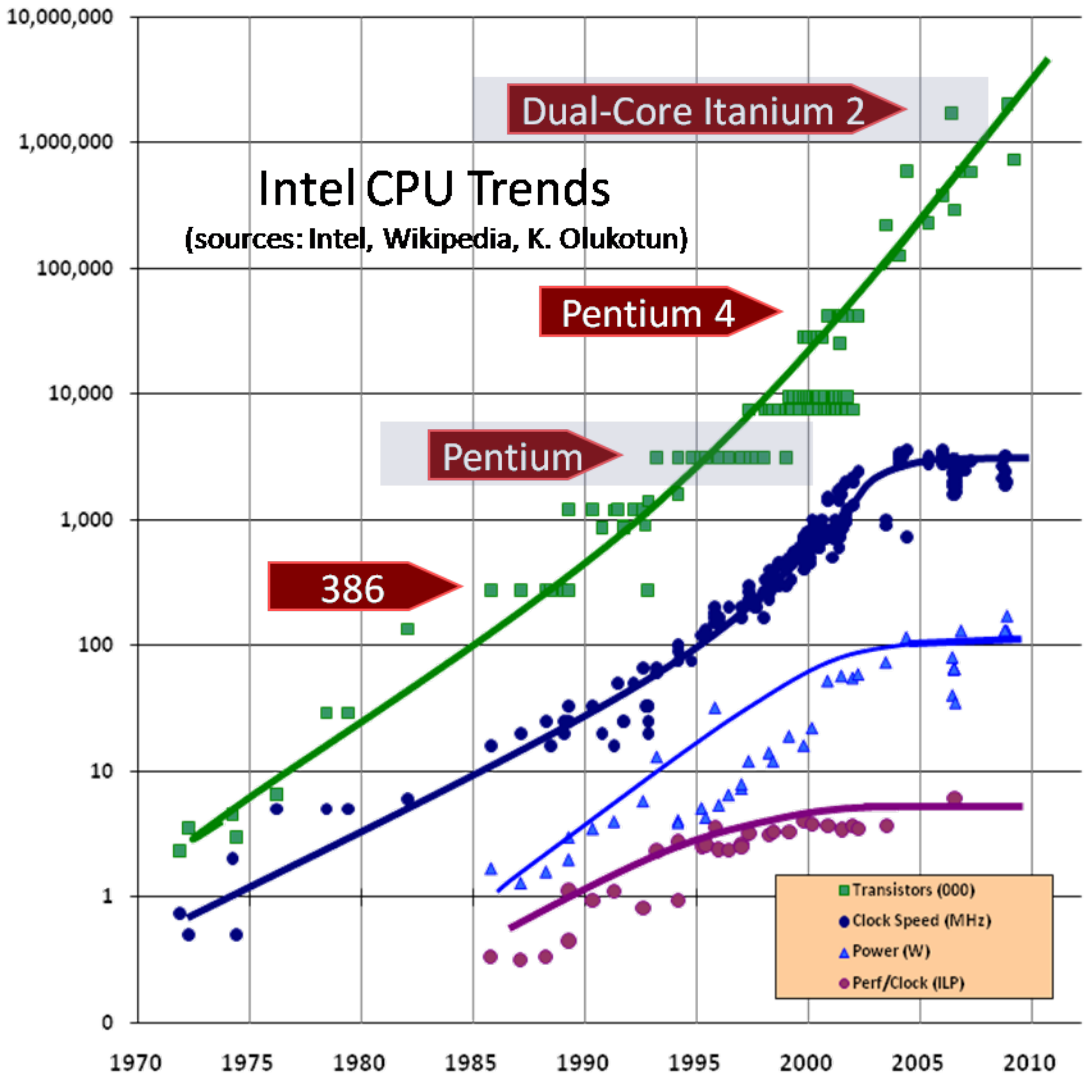
Clock Speeds Increased



Notice a little trick: x axis goes only to 2003!

Until they stopped increasing!

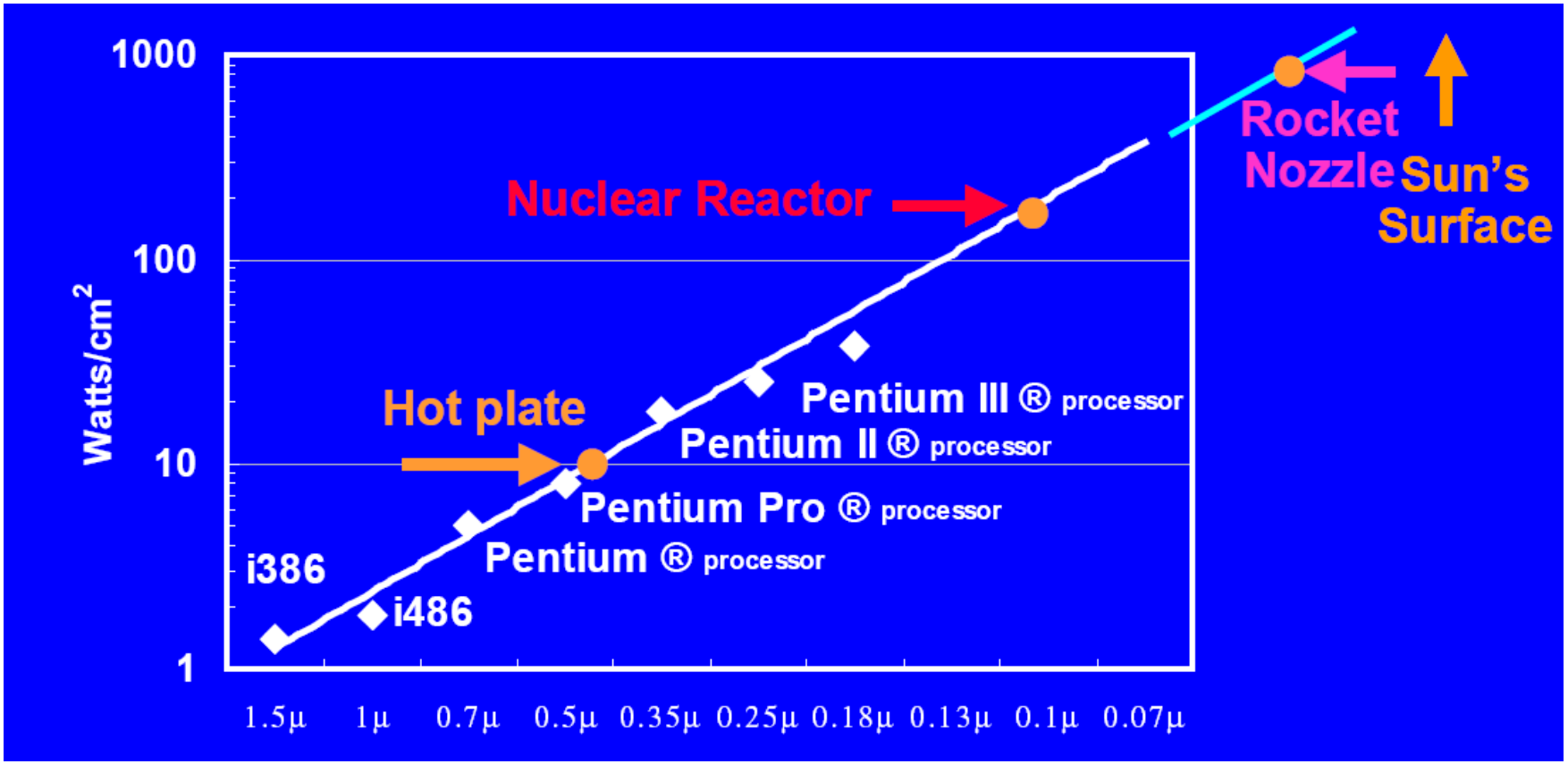




Source: Herb Sutter (orig. in DDJ)

Prediction in 1999

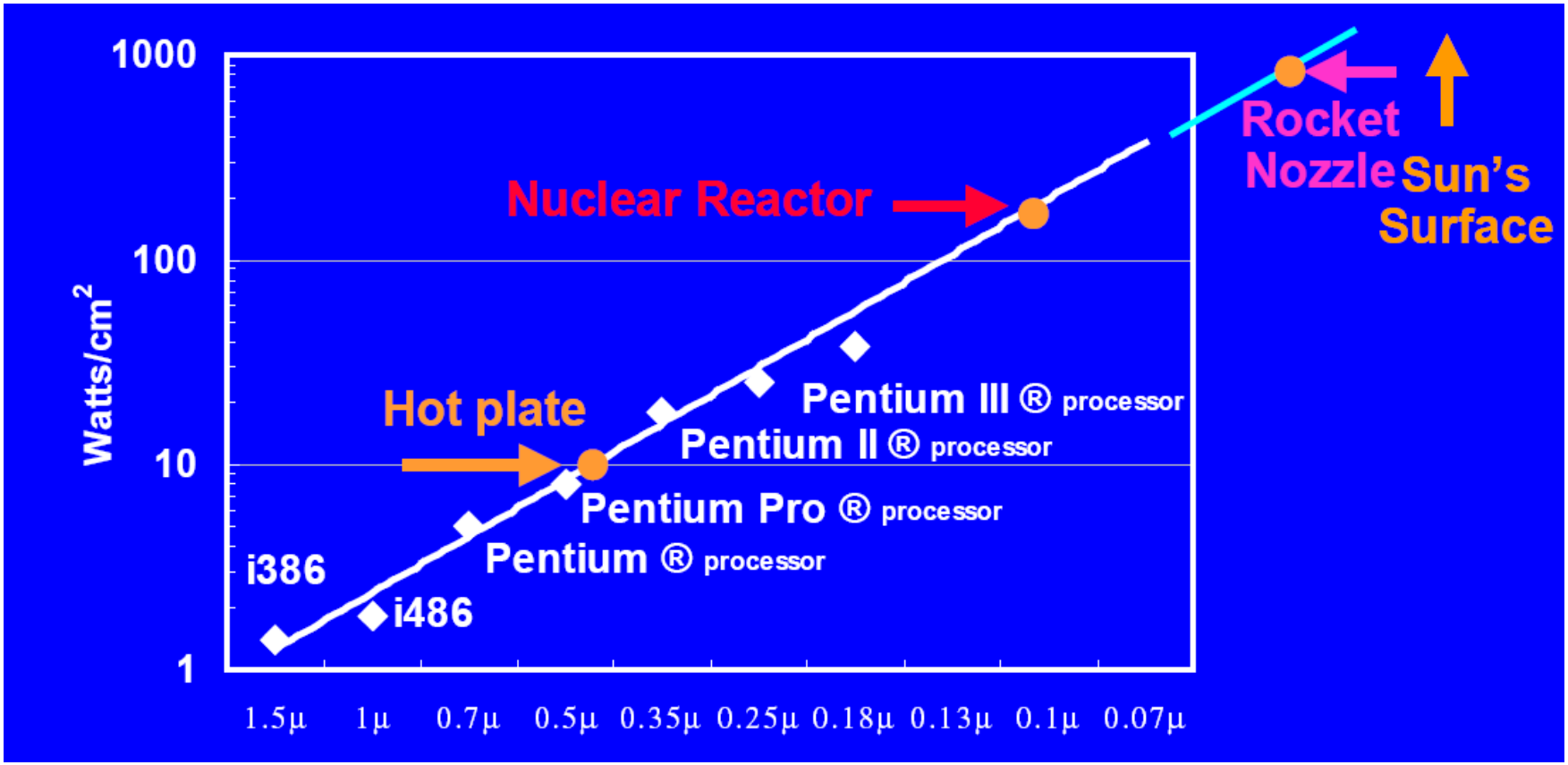
From Shekhar Borkar, Intel, at MICRO' 99



So, the chips were getting too hot

Prediction in 1999

From Shekhar Borkar, Intel, at MICRO' 99



So, the chips were getting too hot

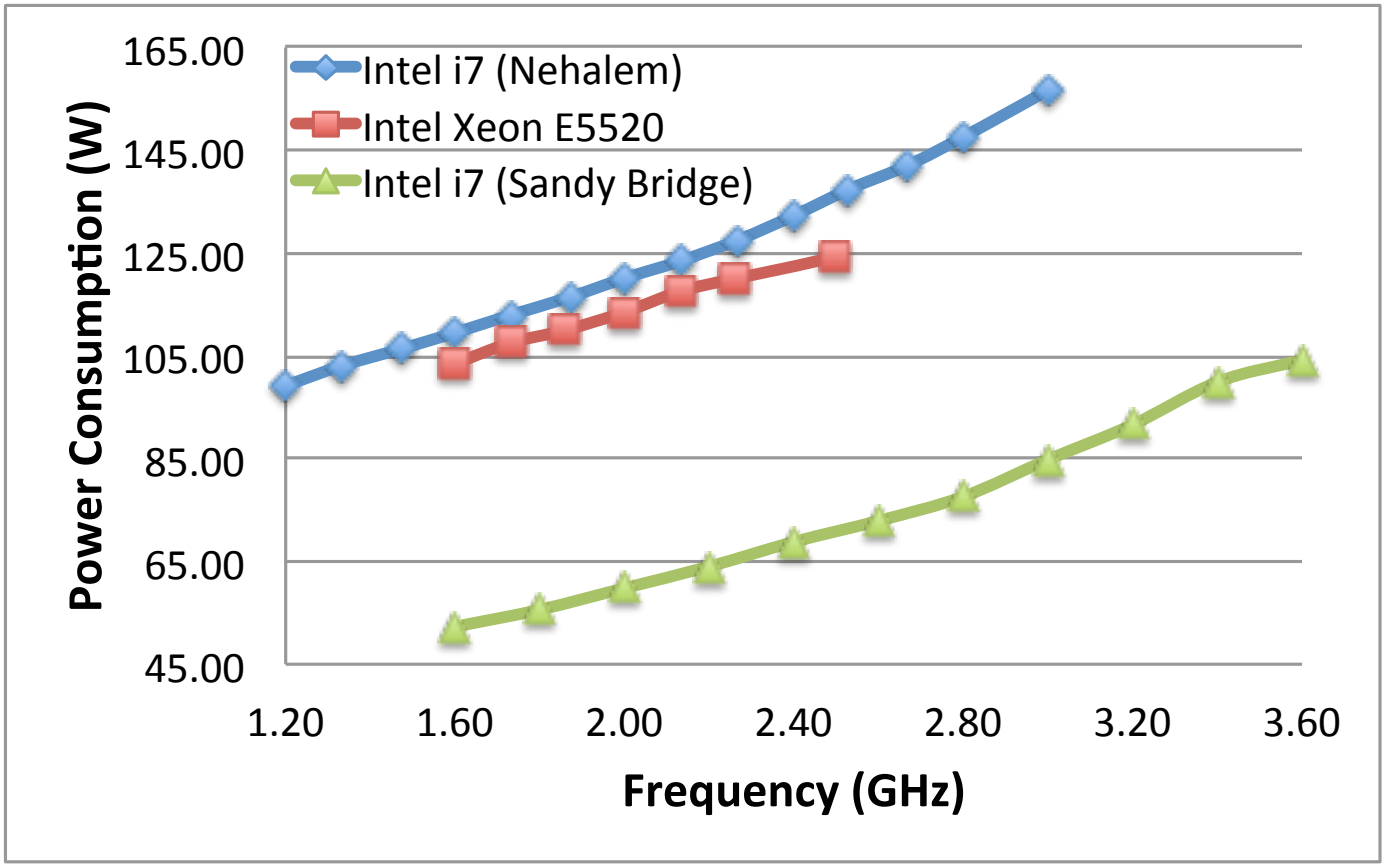
Bill Wulf in 1978

- William Wulf is one of the most influential computer scientist
- Visited IISc Bangalore around 1978..
  - When I was a grad student
- Talked about his parallel computing projects
  - C.mmp
  - Stated motivations: Sequential processors cannot keep getting faster forever, because of physical limitations. We need many processors working in parallel to compute faster
- But engineers kept making it go faster
  - Until now

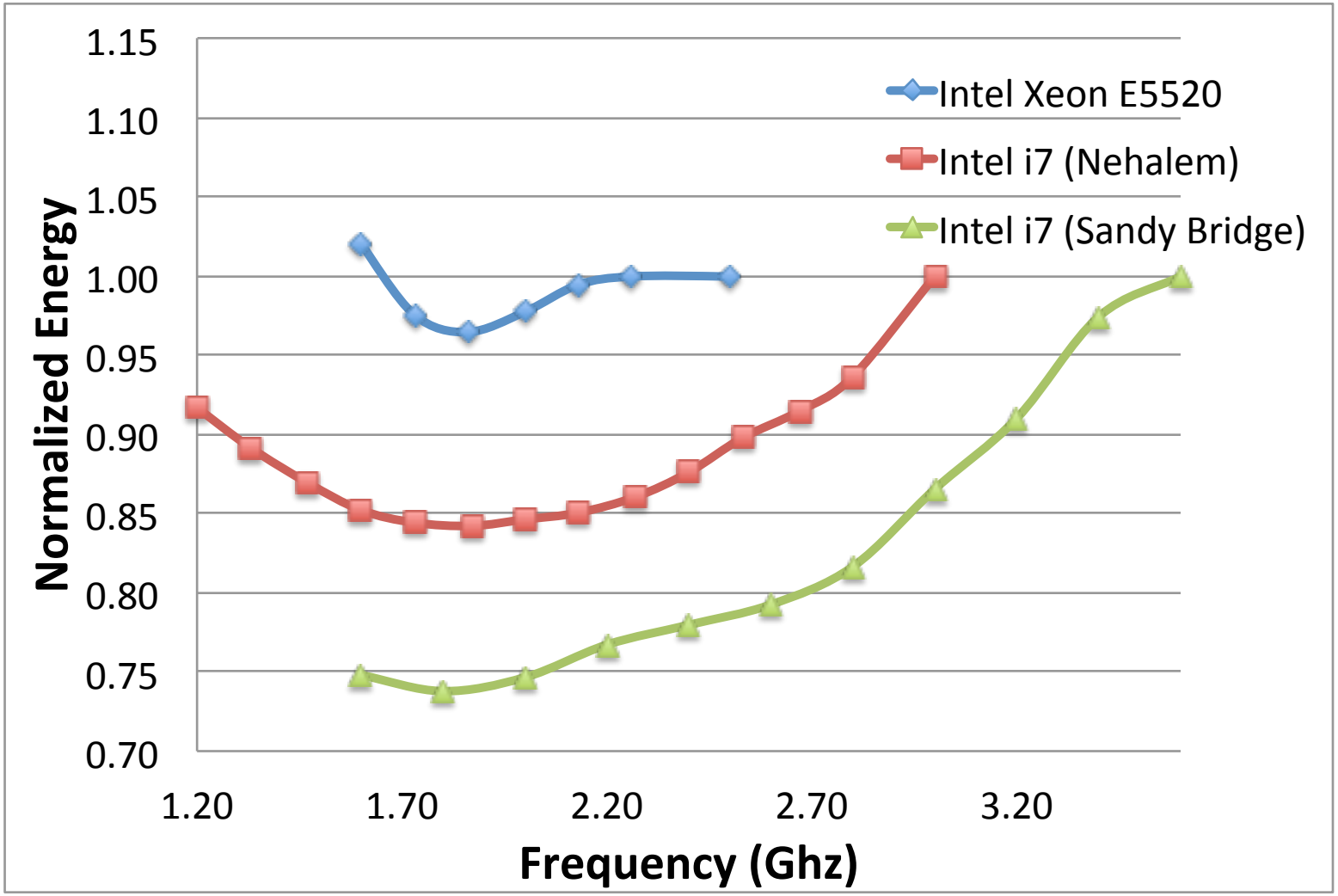
Frequency and Power

- There is a cubic relationship between them!
  - So, if frequency doubles, dynamic power increases 8-fold!
  - Static power is still a significant part of the total power
- So, frequencies stalled around 2-3 GHz
  - Which is plenty fast

Power vs Frequency on a given processor

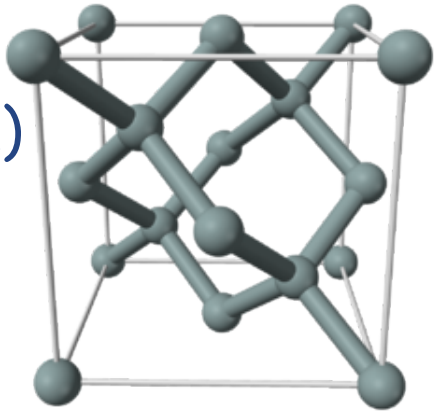


Energy vs Frequency on a given processor



Number of Transistors/chip?

- Well, they will keep on growing for the next ten years
  - May be a bit slowly
- Current technology is 32 or 22 nanometers
- We may go to 9 or 5 nanometers feature size
  - i.e. gap between two wires (as a simple definition)
- For comparison:
  - Distance between a carbon and a Hydrogen atom is 1 Angstrom = 0.1 nanometer!
  - Silicon-Silicon bonds are longer
  - 5 A° lattice spacing (image: wikipedia)
    - i.e. 0.5 nanometer
  - So, we are close to atomic units!



Consequence

- We will get to over 50 billion transistors/chip!
- What to do with them?
- Put more processors on a chip
- Beginning of the multicore era:
  - Number of cores per chip doubles every X years
    - X= 2? 3?

Change is changing

- To summarize:
  - We had been used to computers becoming faster every year.. That "change" was a constant
  - The change is: that the speeds are no longer changing..
- So, Lets think about what happens over the next 10 years
- And later:
  - What happens *after* ten years, when even the number of transistors don't increase any more

Why didn't we witness disruption?

- Why haven't we seen the effect of this disruption already?
  - After all the clocks stop speeding after 2003
- Better integration :
  - Intel's Nehalem architecture
- 4-way multicores could show useful performance gains for the users
  - Running multiple programs simultaneously
  - Browse, Index mail, virus scans, …
- The real questions will appear in near future

Specialized processors

- It was discovered that somewhat specialized processors can be made to compute (some things) faster, within the same technology
  - Vector instructions (SSE):
    - "Do these 4 additions" as a single instruction
  - IBM's Cell processor (Toshiba, Sony)
  - NVIDIA GPGPU
  - Intel "maybe" LRB, MIC
- It was assumed that people will not be willing to program specialized processors
  - NVIDIA proved them wrong
  - And time was right

Threat to business models

- The previous were just ways in which processors can keep getting faster
- But will people buy them?
- Intel/AMD/.. Business model is fueled by people buying new machines every few years
- Why do people buy new machines?
  - New apps need faster processors

Two problems

- Maybe we have all the speed we need..
  - I.e. for all the apps that we need
  - Nyah..
- Maybe 8-16 cores is all that you need
  - We are still seeing improvements because
    - We use multiple programs on the desktop
    - Browsers can do multiple things: get data, draw pictures, ..
  - But now, we have enough power.. Right?
- So, unless one (or more) parallel "killer app" appears, the market will stop growing

What if we stop here?

- Technical advantage (of Intel, AMD, NVIDIA) is no longer an advantage
- Processor chips become commodity
  - Get manufactured wherever it is cheap
- Innovation shifts elsewhere
- Or more starkly stated:
  - Innovation in computing stops

Alternative: Parallelism

- If we find killer apps that need all the parallel power we can bring to bear
  - With 50B transistors, at least 100+ processor cores on each chip
- There is a tremendous competitive advantage to building such a killer app
  - So, given our history, we will find it
- What are the enabling factors:
  - Finding the application areas.
  - Parallel programming skills

A few candidate areas

- Some obvious ones:
- Image processing:
  - Find all pictures of my daughter with a cat from all my albums
    - Already exists and will improve.. Parallelism is easy

More interesting areas

- Speech recognition:
  - almost perfect already
  - But speaker dependent, minor training, and needs non-noisy environment
  - Frontier: speaker independent recognition with non-controlled environment
- Broadly: Artificial intelligence
- (Of course, HPC)

All Programming Becomes Parallel Programming

Parallel programming Skills

- So, all machines will be (are?) parallel
- So, almost all programs will be parallel
  - True?
- There are 10 million programmers in the world
  - Approximate estimate
- All programmers must become parallel programmers
  - Right? What do you think?

One way: Sequential-Parallel Separation

- Our own example is a language called "Charisma"
  - or a simpler variant: structured dagger
- A charisma Script captures parallel interactions
- Sequential methods
  - consume data given to them, compute, and "publish" data
  - without knowing where it goes
- You can decompose development into a team of parallel experts and domain experts..
  - No "Joe Shmoe"s

Another Idea

- Parallel programming is difficult
  - in large part because of race conditions, and non-deterministic behavior
  - Things may be complete in different orders than we expected
  - Its like relativity: no notion of "simultaneity" on a parallel machine
  - Result: a bug that manifests itself once in 10,000 runs
- So, Outlaw non-determinism
  - Not quite like the Indiana legislature and Pi
  - Develop programming models that don't allow it
  - Need to interoperate with languages that do
- Our Examples:
  - MSA (Multiphase Shared Arrays), Charisma

Programming Models innovations

- Expect a lot of novel programming models
  - There is scope for new languages, unlike now
  - Only Java broke thru after C/C++
- This is good news:
  - If you are a computer scientist wanting to develop new languages
- Bad news:
  - If you are a application developer
- DO NOT WAIT FOR "AutoMagic" Parallelizing compiler!

Digging Deeper

- First law of holes:
  - If you in a hole, stop digging!
- But we are in a hole, and we cannot help but dig!
- Let me explain

Latency to Memory

- To process data, we must bring it to the processor and take the resulting data back to memory
- DRAM, the main inexpensive memory chip
  - Once you supply an address to it, it gets you the data after 50-ish nanoseconds
    - Doesn't improve that much over time: 80 -> 30 ns
  - A single core clock is 2 GHz: it beats twice in a nanosecond!
  - So, you are working with someone who is 100 times slower..
  - Not just that: a single core can do 4+ additions/cycle
  - We are talking about putting hundreds of cores on a chip?!

Latency vs bandwidth

- Imagine you are putting a fire out
  - Only buckets, no hose
  - 100 seconds to walk with a bucket from water to fire, (and 100 to walk to walk back)
  - But if you form a bucket brigade
    - (needs people and buckets)
  - You can deliver a bucket every 10 seconds
    - So, latency is 100 or 200 seconds, but bandwidth is 0.1 buckets per second.. Much better
  - Whats more: you can increase bandwidth:
    - Just make more lines of bucket brigade

Bandwidth can be increased

- "Only" need resources
- But technology is going to help
- More pins can be added to chips
- 3D stacking of memory can increase bandwidth further

Exploiting bandwidth

- Need methods that translate latency problems to bandwidth problems
- The difference with bucket brigade analogy:
  - Data dependencies
  - If what mixture to ask in the next bucket depends on what happened using the last one
  - Solution: concurrency

Architectural methods

- For translating latency problems to bandwidth problems
- Ask for nearby data: Cache hierarchies
- Ask for data in bulk (prefetch) and operate on it in bulk : cell processor
- OR
  - Every cycle: work on one problem, send its memory request out, and switch to another problem
  - GPGPU (and before that, MTA, ..)

Impact on segments

- Mobile
- Laptop/desktop
- Small clusters
- Supercomputers

Mobile computing

- Clients will get more powerful
  - Within same form factors..
  - Of course, well connected
- Your children will tell their children wistfully about text messaging..
  - Because speech-to-text (and may be brain-to-screen) may become the norm
- Headsets will whisper the name of the person walking towards you..
- Robots?

Laptops/Desktops

- This is the big question mark
- Depends on the parallel killer app
- Of course, speech recognition, video processing will be there, as in mobile

Small Clusters

- Probably some of the biggest impact
- Broadening of the market
- Every company/department can afford a very powerful (100 TF? PF?) cluster
- All kinds of activities can be computerized
  - Intel's example:
    - fashion designers examining how a cloth will drape over a body, and how it will move
    - Via simulation
- Operations Research
- Business Strategies via AI support

Supercomputers

- Exascale will be reached by 2020
  - May be 50 MW, and 10^18 ops/s
- I expect
  - Will create breakthroughs in Science and Engineering of great societal impact
  - Biomedicine, materials,
  - Astronomy, Physics: theories
  - Engineering design of better artifacts
  - Control Nuclear fusion (fission) may solve energy problems
  - Climate??
- If society deems it beneficial, technology can be developed for beyond-exascale (1000 Eflops?)

Next Era: End of Moore's Law

- 10-15 years from now
- No more increase in performance from a general purpose chip!
- What can we predict about this era?
  - First, innovation would shift to functional specialization
    - would have started happening already
  - Next, innovation will shift to application areas, and molecular sciences: biomedical (nanobots?), materials,
  - Another 5-10 years, you can develop CSE applications knowing that machine won't change under your feet
    - Maybe

Caution: predicting future

- Remember:
  - 1900 or so: "End of Science" predicted
  - 1990 or so: "End of History" predicted

Summary

- Times are changing:
  - I.e. they are getting more stagnant!
- Those who can "get" parallel, will have an advantage
- If killer parallel app doesn't arrive, progress will stall
- Complete "stasis" after 15-20 years..
  - But then such things have been predicted before
  -