Parallel Programming and Heterogeneous Computing

Introduction

Max Plauth, Sven Köhler, Felix Eberhardt, Lukas Wenzel and Andreas Polze

Operating Systems and Middleware Group

Teaching Team (PhD Students)

- Max Plauth
  - GPU Accelerators
  - Virtualization of Accelerators

- Felix Eberhardt
  - NUMA Topologies
  - Dynamic Scale-Up/ Down

- Sven Köhler
  - On-Core Accelerators (SIMD, Crypto)
  - Programming Models for Accelerators

- Lukas Wenzel
  - Computer Architecture
  - FPGA Accelerators

Course Topics

A. The Parallelization Problem
   - □ Power wall, memory wall, Moore’s law
   - □ Terminology and metrics

B. Shared Memory Parallelism
   - □ Theory of concurrency, hardware today and in the past
   - □ Programming models, optimization, profiling

C. Heterogeneous Computing
   - □ On-Chip Accelerators (e.g. SIMD, special purpose accelerators, etc.)
   - □ External Accelerators (e.g. GPUs, FPGAs, etc.)

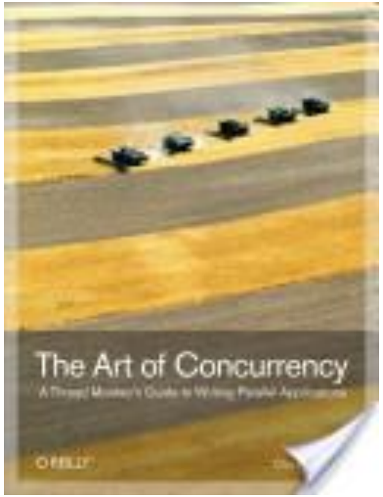
D. Shared Nothing Parallelism
   - □ Theory of concurrency, hardware today and in the past
   - □ Programming models, optimization, profiling

Course Design

This is a course about concepts, not a programming tutorial!

- Weekly lectures
- Practical assignments
- Oral exam ~30 minutes

- Literature list on course home page
- Based on a course by [Prof. Dr. Peter Tröger]



The Art of Concurrency: A Thread Monkey's Guide to Writing Parallel Applications

Clay Breshears
O'Reilly Media, Inc.
2009

Course Design
Lectures

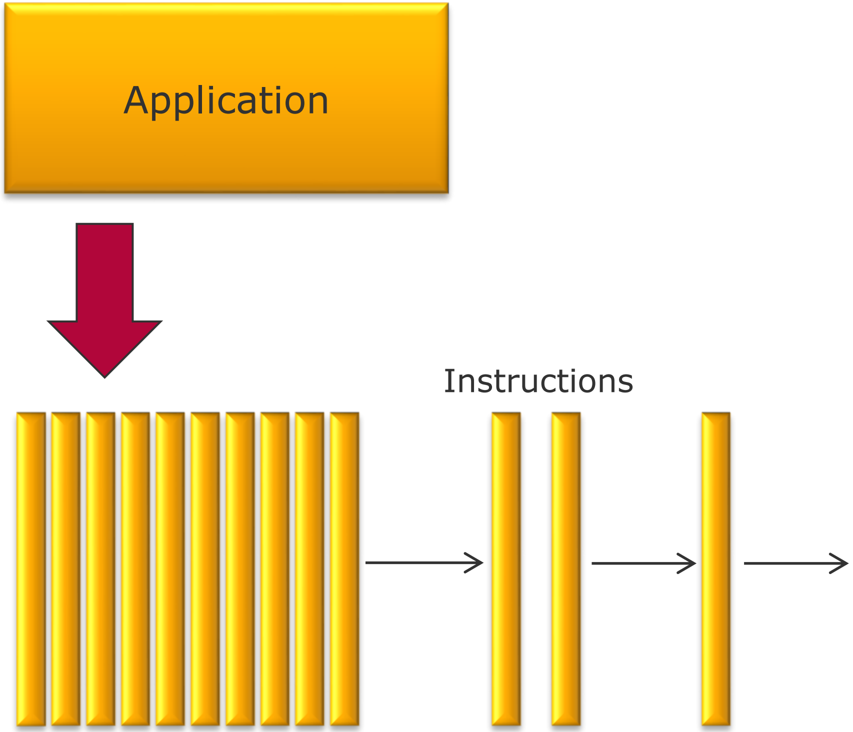
- Lectures are given in flipped classroom style
  - 75% time: Videos delivering lecture content
  - 25% time: Online meetings for recap sessions, questions and discussions
- Videos are posted every Wednesday on the website
- Meeting links for Thursday will also appear on the website

https://osm.hpi.de/parProg/2020/

Course Design Assignments

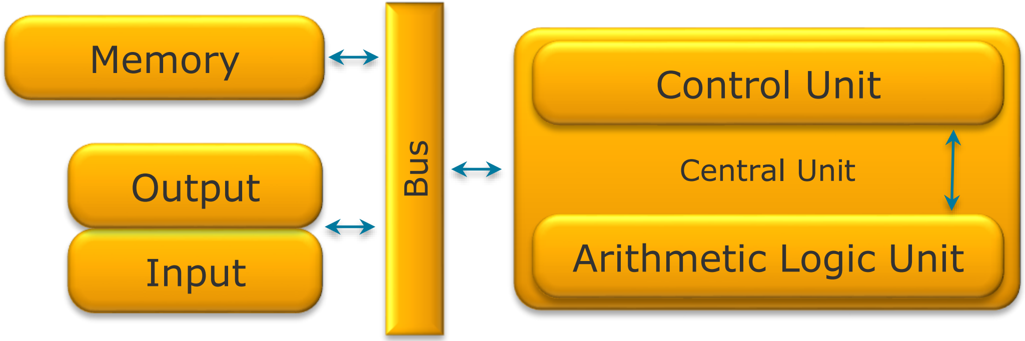
- Implementation of parallel algorithms to demonstrate the use of programming models discussed in the lecture
- Presented every ~3 Weeks
- Solved in teams of 2 persons
- Submission to https://osm.hpi.de/submit/
- At least 50% correct solutions enable participation in final exam

Running Applications



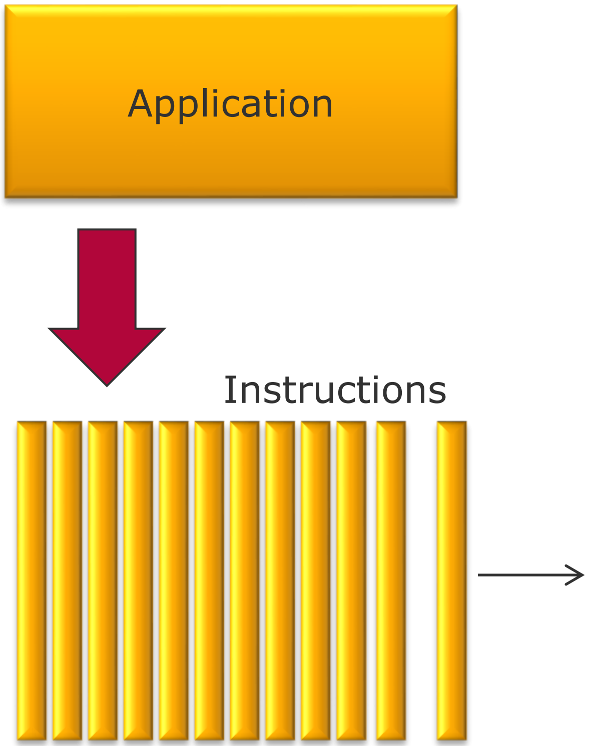
Machine Model

- First computers had fixed programs (electronic calculator)
- **von Neumann architecture** (1945, for EDVAC project)
  - Instruction set for control flows stored in memory
  - Program is treated as data, which allows the exchange of code during runtime and self-modification
  - Introduced the **von Neumann bottleneck**
- CPUs are built from logic gates, which are built from transistors
- Multiple CPUs (SMP) were always possible, but exotic



ParProg 2020
Introduction

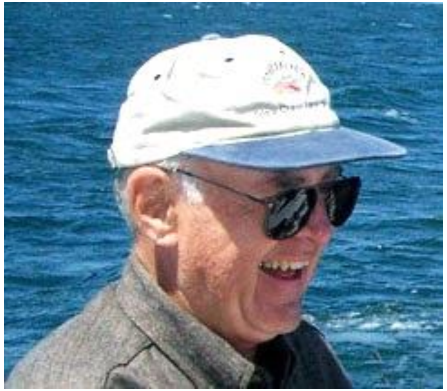
Three Ways of Doing Anything Faster [Pfister]



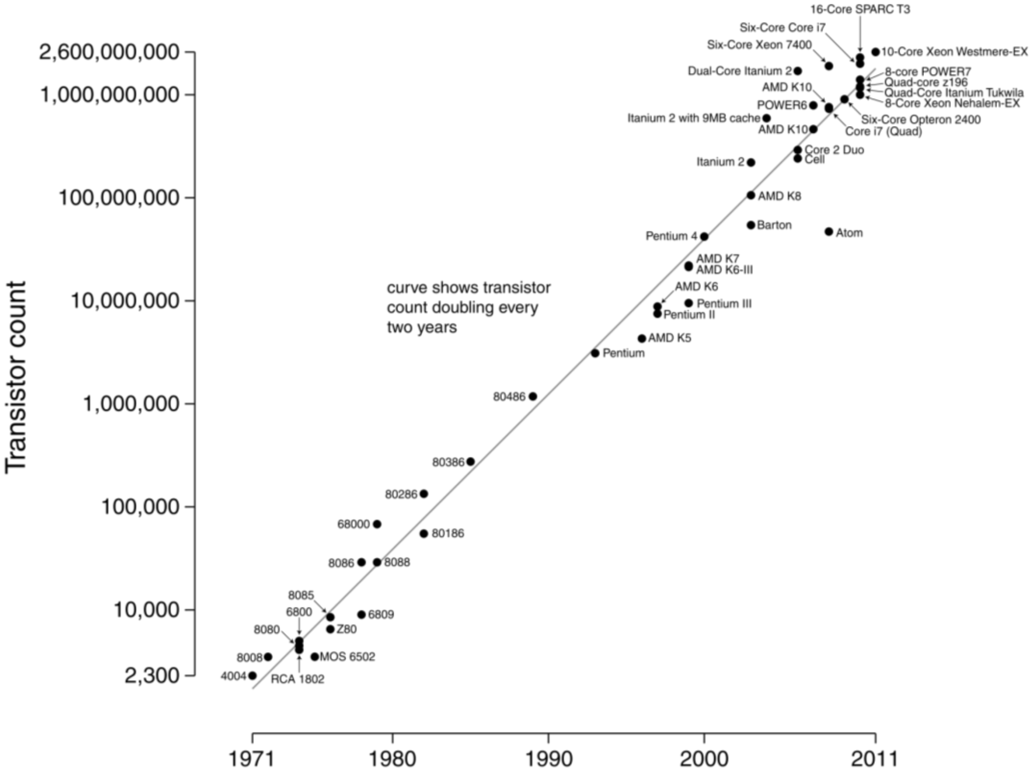
- Work Harder
  (clock speed)
- Work Smarter
  (optimization, caching)
- Get Help
  (parallelization)

Moore’s Law

- *„...the number of transistors that can be inexpensively placed on an integrated circuit is increasing exponentially, doubling approximately every two years. ...“ (Gordon Moore, 1965)*
  - □ Rule of exponential growth
  - □ Applied to many IT hardware developments
  - □ Sometimes misinterpreted as performance indication
  - □ Has become a self-fulfilling prophecy
  - □ Comes to an end within the next 5-10 years



Moore’s Law



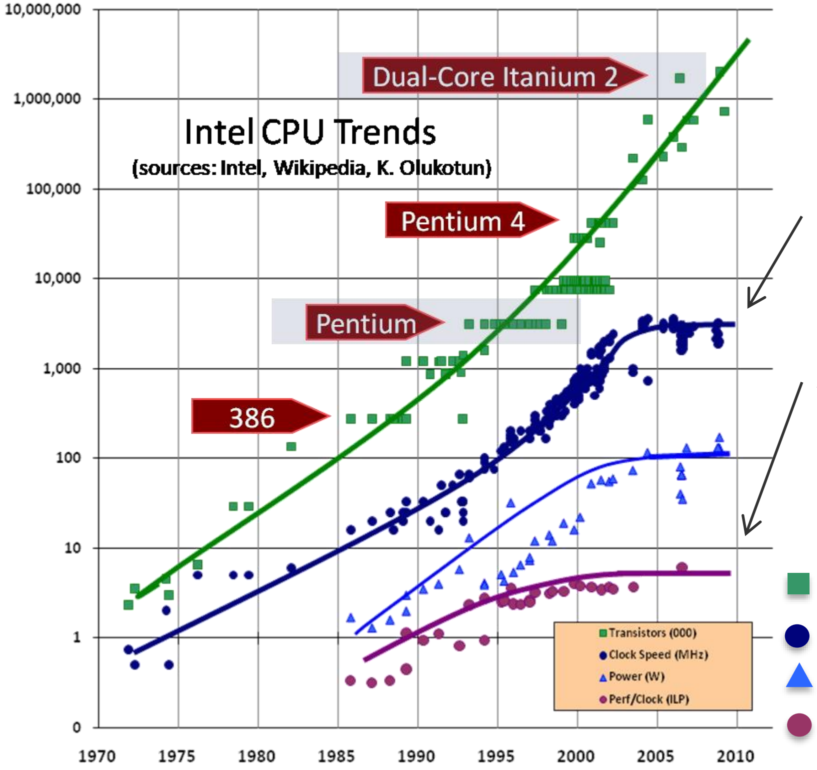
[Wikimedia]

Moore`s Law vs. Software

- Gate’s law: *“The speed of software halves every 18 months.”*
- Wirth’s law: *“Software is getting slower more rapidly than hardware becomes faster.”*
- May’s law: *“Software efficiency halves every 18 months, compensating Moore's Law.”*
- Jevons paradox:
  “Technological progress that increases the efficiency with which a resource is used tends to increase (rather than decrease) the rate of consumption of that resource.”
- Zawinski's Law of Software Envelopment:
  “Every program attempts to expand until it can read mail.
  Those programs which cannot so expand are replaced by ones which can.”

Processor Speed Development

[Herb Sutter, 2009]



„Work harder“

„Work smarter“

🟩 Transistors #

● Clock Speed (Mhz)

▲ Power (W)

● Perf/Clock (ILP)

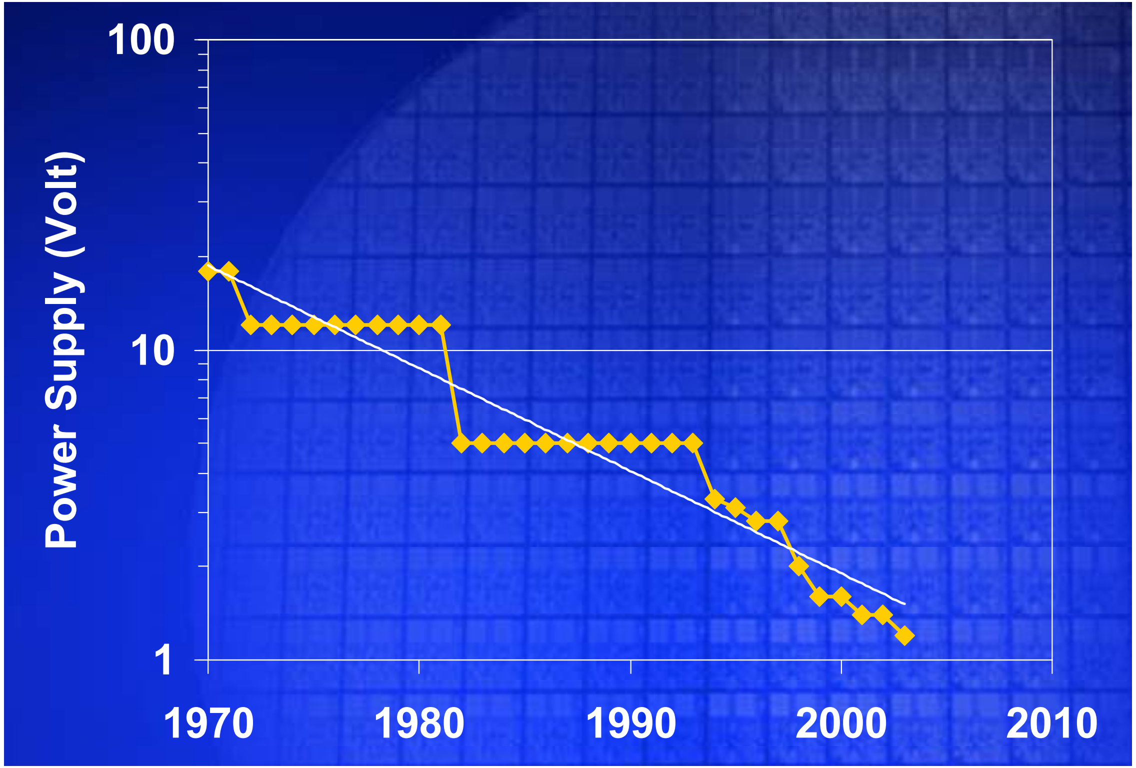
A Physics Problem

- Power: Energy needed per time unit
  - Power density: Watt/mm^2 → Cooling
- **Static power**: Leakage of transistors while being inactive
- **Dynamic power**: Energy needed to switch a gate

Dynamic Power ~
Number of Transistors (N) x Capacitance (C) x
Voltage^2 (V^2) x Frequency (F)

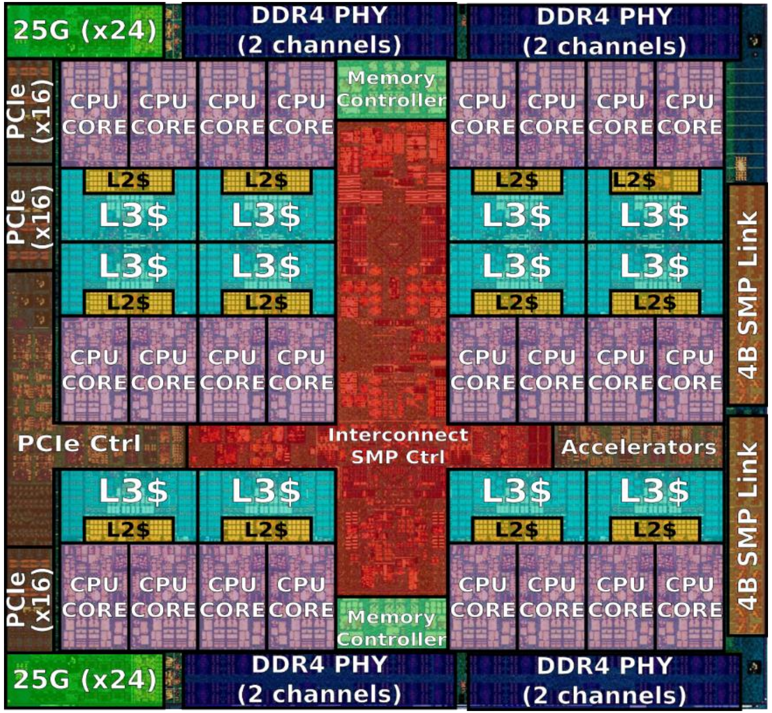
- Moore’s law: N goes up exponentially, C goes down with the size
- The trick
  - Bringing down V reduces energy consumption, quadratically
  - Don’t use all the N for gates (e.g. caches)
  - Keeps the dynamic power increase moderate
  - We can happily increase F with N for faster computation

Processor Supply Voltage



[Moore, ISSCC]

Transistor Usage

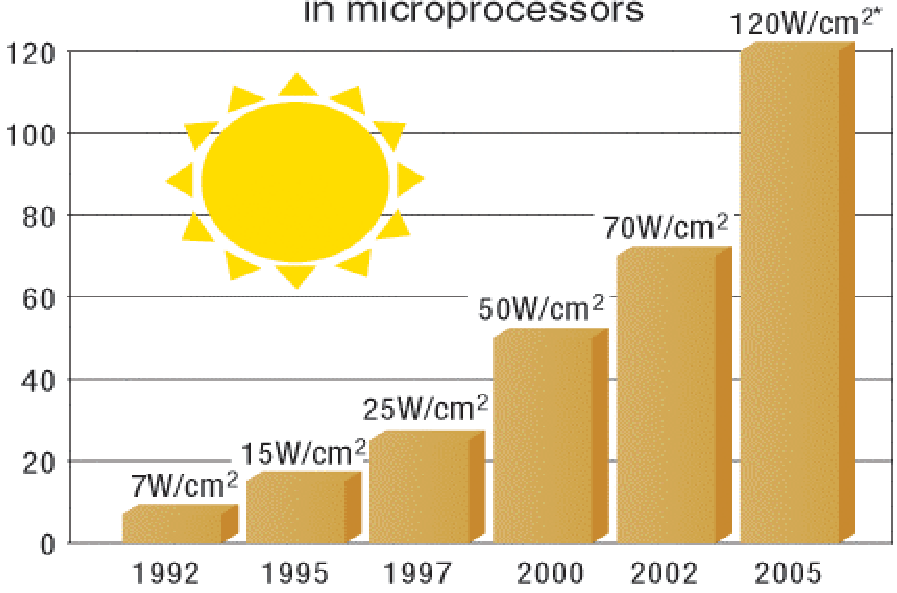


[https://en.wikichip.org/wiki/ibm/microarchitectures/power9]

Power Density

SIZZLING SEMICONDUCTORS

growth of watts per square centimeter in microprocessors



*Could be higher, depends on level of integration.

SOURCE: HEWLETT-PACKARD LABS

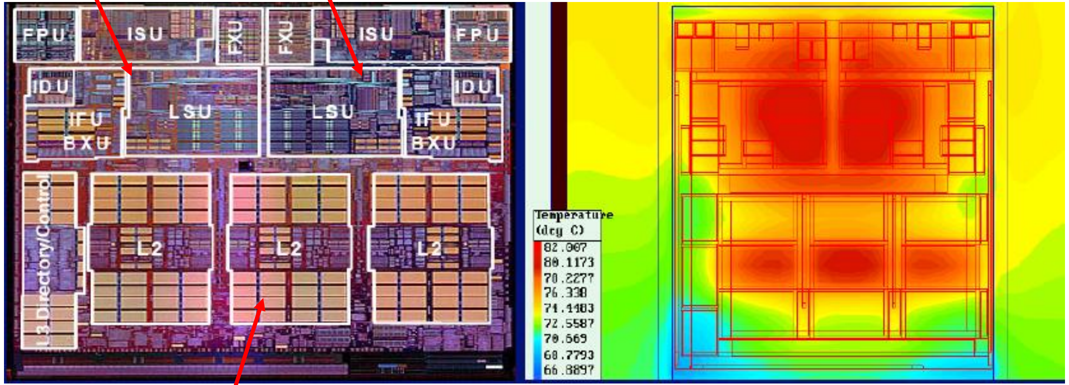
Power Density = Temperature

1st CPU

2nd CPU

[source: Devgan'05]

Power 4 server chip

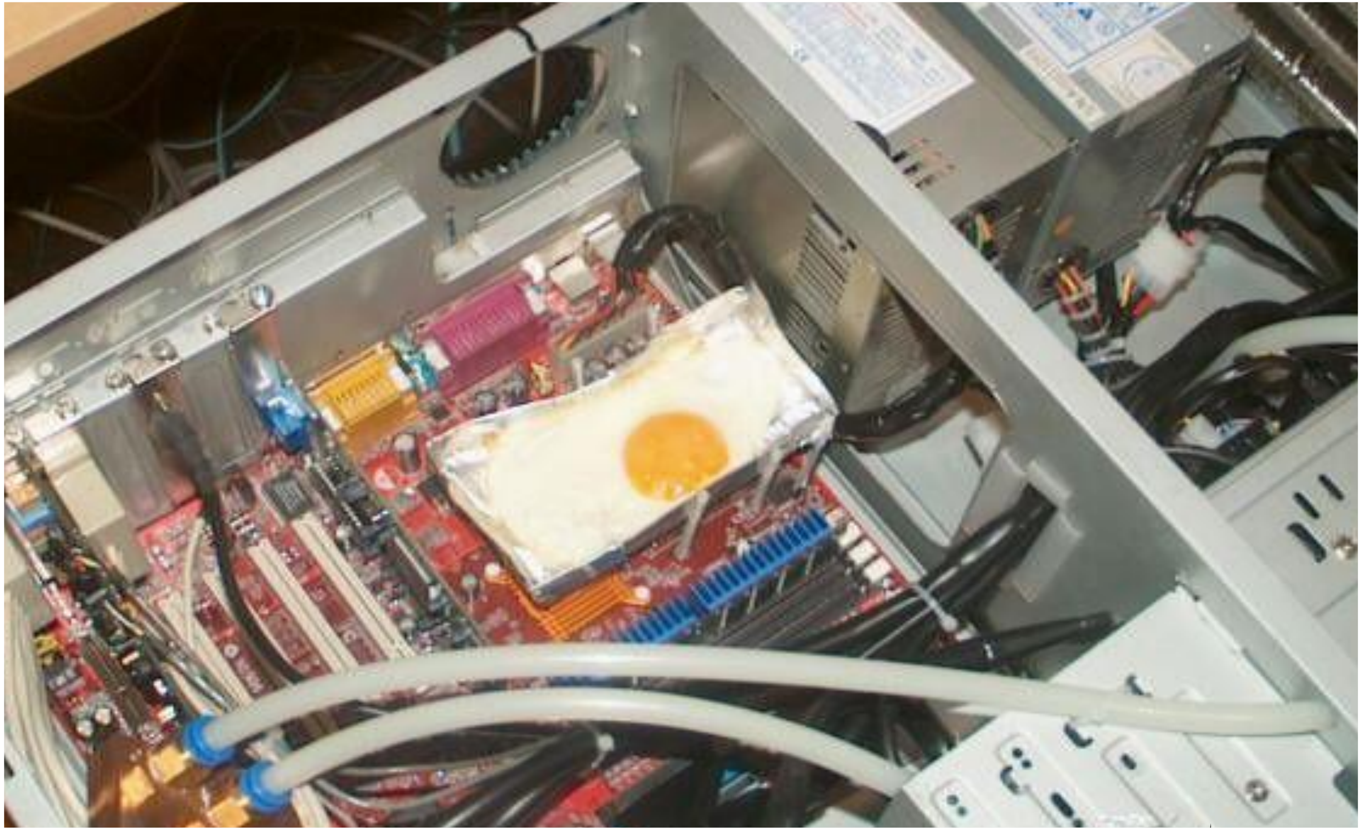


cache

thermal profile during runtime

- Higher temperature leads to
  - Increased transistor leakage
  - Decreased transistor speed
  - Higher failure probability

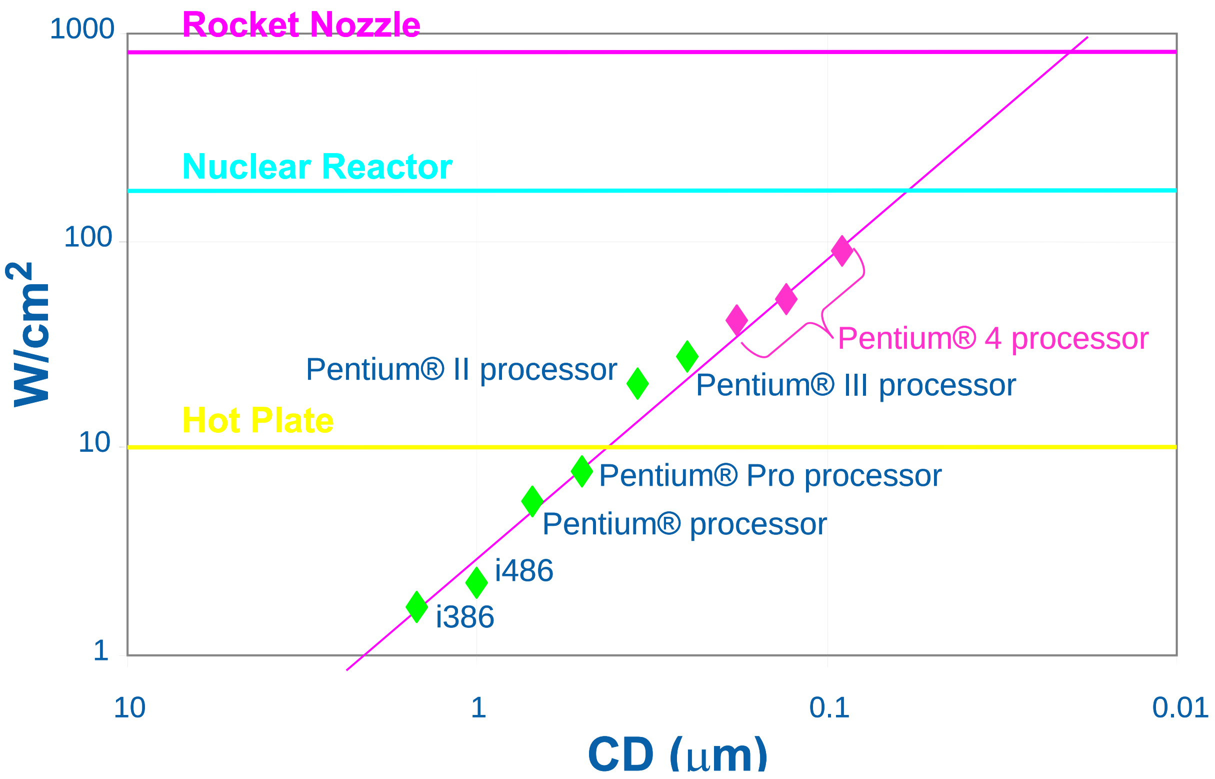
Power Density



[Kevin Skadron, 2007]

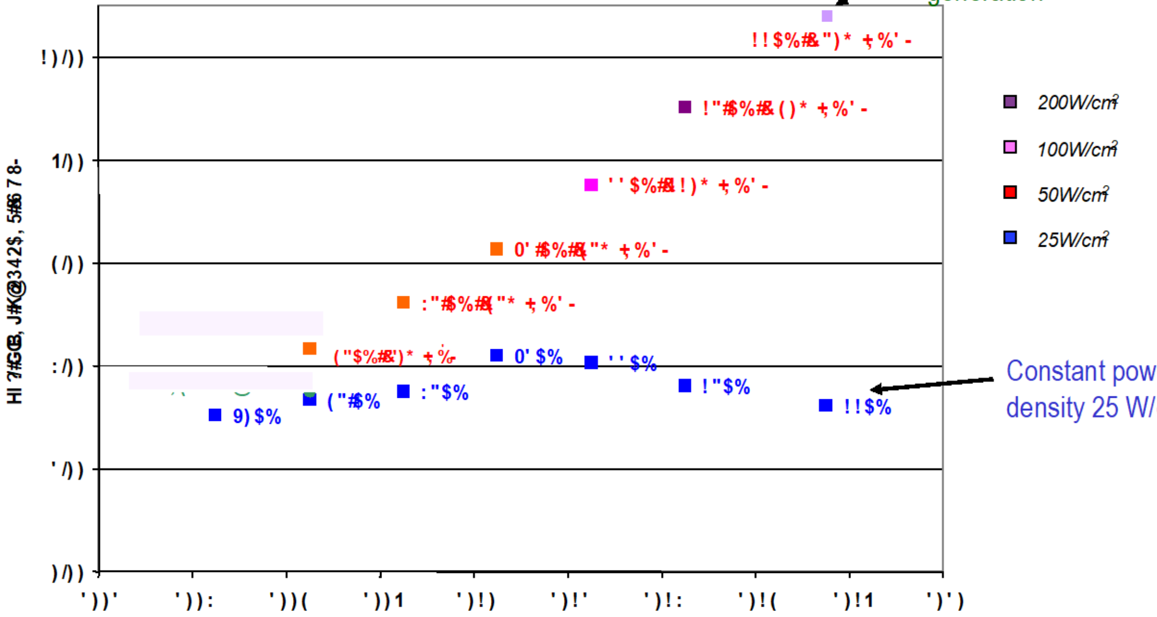
Power Density

[Taylor, 2009]



Power Density

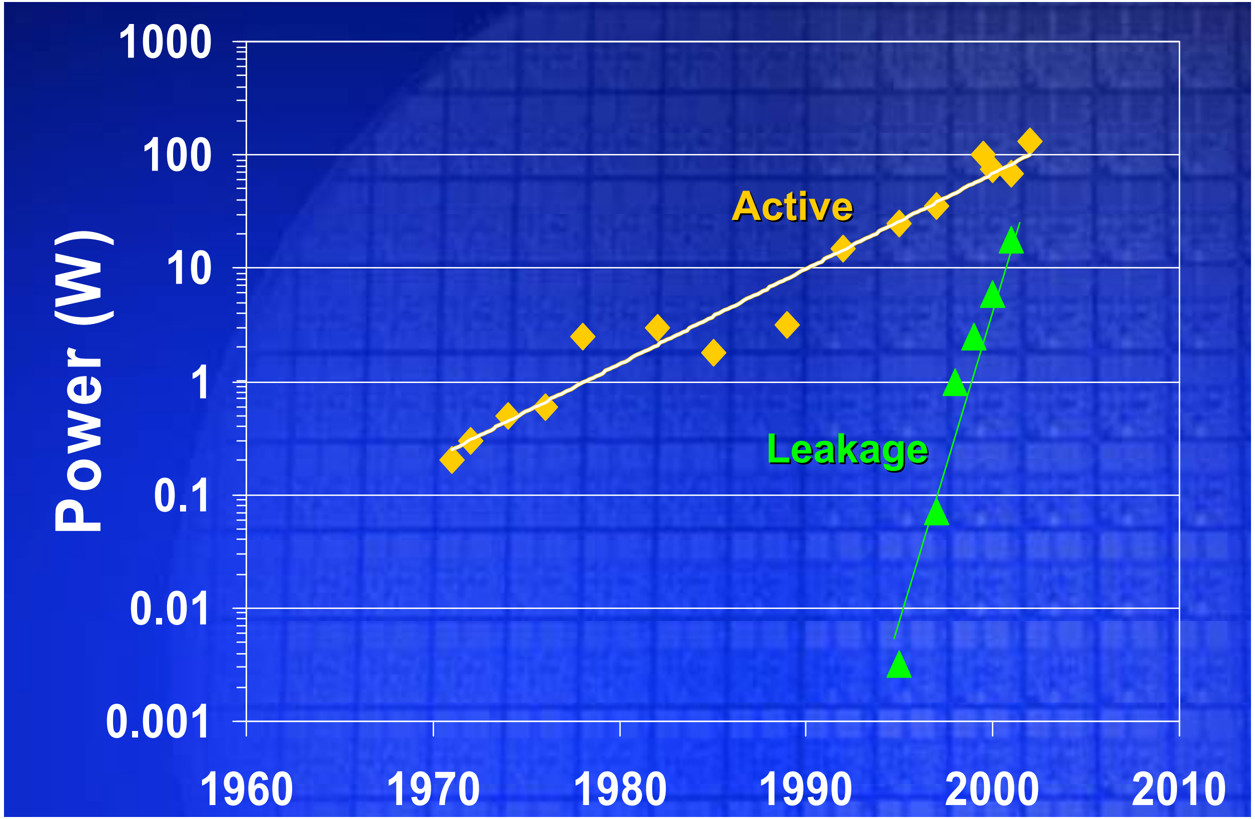
Constant performance improvement of 20% per generation



Constant power density 25 W/cm^2

Source: *D. Frank, C. Tyberg,* IBM Research

Second Problem: Leakage Increase



[www.ieeeghn.org]

A Physics Problem

Dynamic Power = N x C x V^2 x F

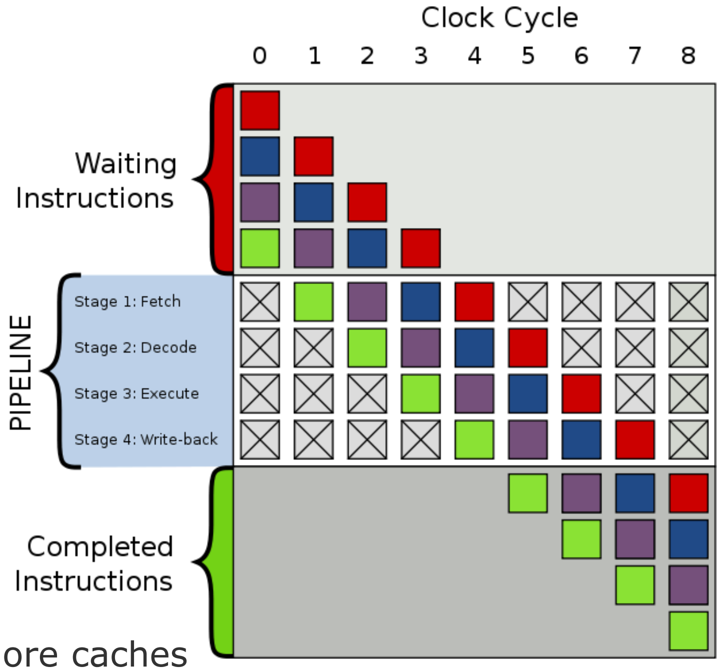
- Even if we would keep F constant
  - N continues to increase exponentially → dynamic power
  - Increasing N sums up to more leakage → static power
- Cooling performance is constant (100-125 Celsius)
  - Static and dynamic power consumption has a limit
- Further reducing V for compensating an additionally increased F
  - Also makes the transistors slower
  - We can’t do that endlessly, 0V is the limit
  - Strange physical effects
- Increasing the frequency is no longer possible
  → **“Power Wall”**
- Ok, so let’s use the additional N for smarter processors

Instruction Level Parallelism

- Increasing transistor count was also used for more gate logic in **instruction level parallelism (ILP)**
  - **Instruction pipelining**
    - Overlapped execution of serial instructions
  - **Superscalar execution**
    - Multiple execution units are used in parallel
  - **Out-of-order execution**
    - Reorder instructions that have no data dependency
  - **Speculative execution**
    - Control flow speculation, memory dependence prediction, branch prediction
- Today's processors are packed with ILP logic

The ILP Wall

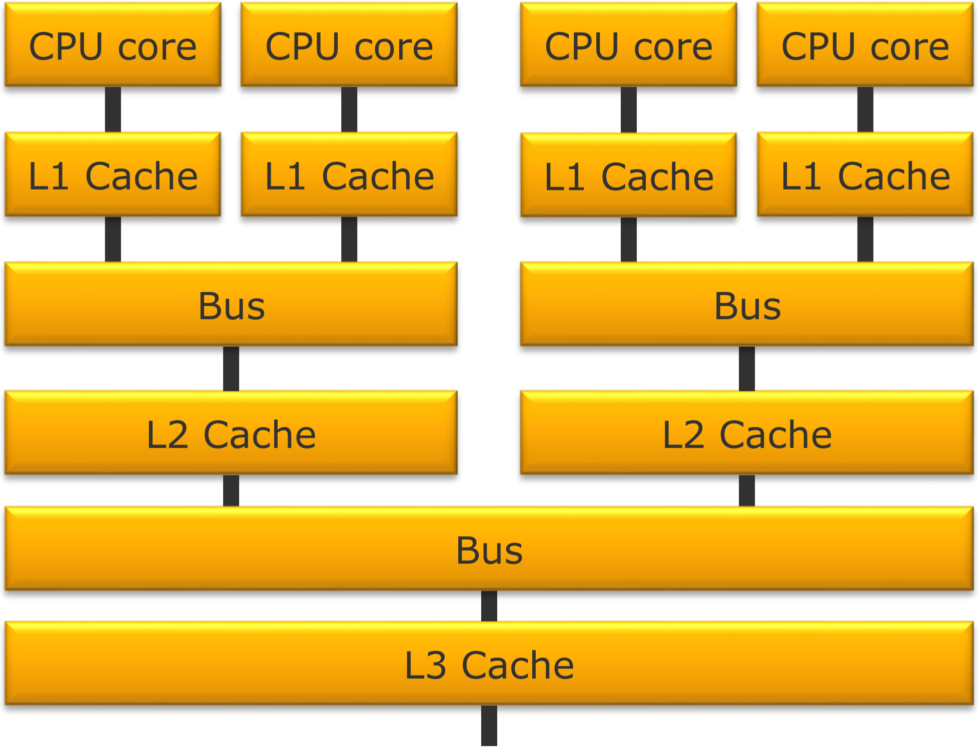
- No longer cost-effective to dedicate new transistors to ILP mechanisms
- Deeper pipelines make the power problem worse
- High ILP complexity effectively reduces the processing speed for a given frequency (e.g. mispredictions)
- More aggressive ILP technologies too risky for products due to unknown real-world workloads
- → **"ILP wall"**
- Ok, so let's use the additional N for more caches



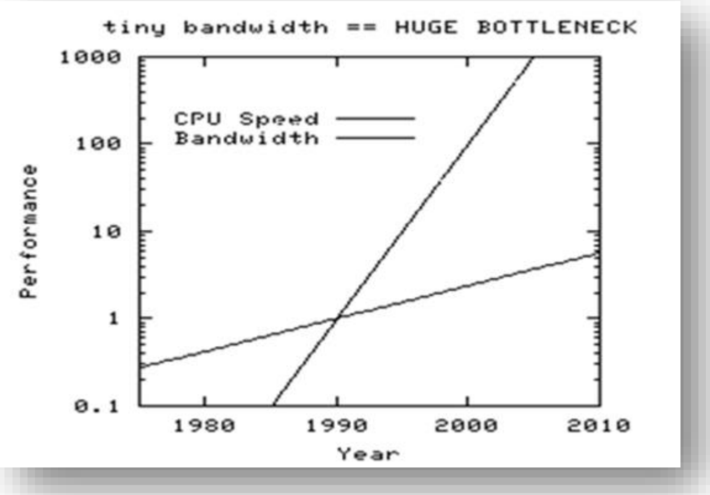
[Wikipedia]

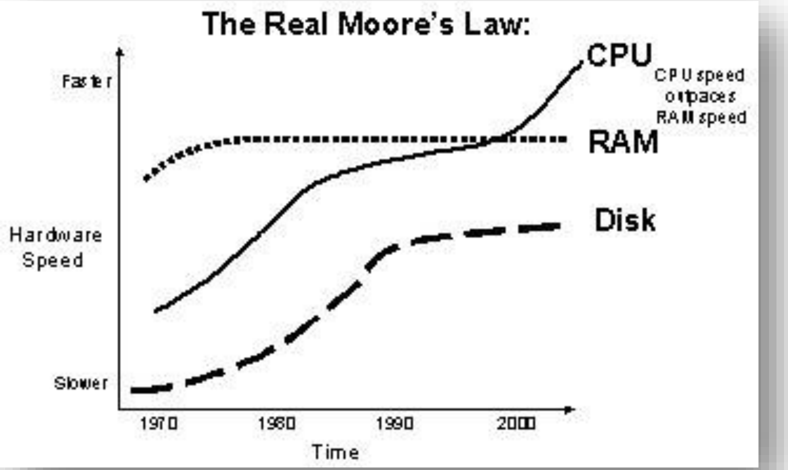
ParProg 2020 Introduction

Memory Hierarchy



Memory Wall


tiny bandwidth == HUGE BOTTLENECK
CPU Speed
Bandwidth
Performance
1000
100
10
1
0.1
1980
1990
2000
2010
Year

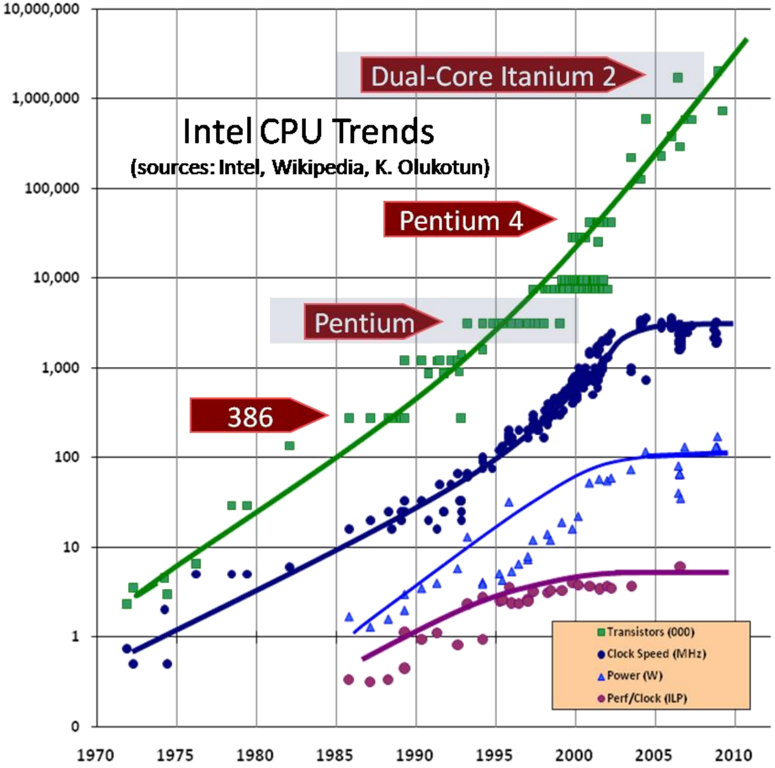

The Real Moore's Law:
CPU
CPU speed outpaces RAM speed
RAM
Disk
Faster
Hardware Speed
Slower
1970
1980
1990
2000
Time

Memory Wall

- Sandia National Labs investigated the speedup achievable by increasing parallelism (ILP, multiple processors) in 2009
- Example: Number of clerks behind a supermarket counter
  - Two clerks can serve more customers than one
  - 4 ? 8 ? 16 ? 32 ? 64 ? ... 1000 ?
- The problem: Shared memory is ‚shared‘
  - Memory bandwidth
    - Memory transfer speed is limited by the power wall
    - Memory transfer size is limited by the power wall
    - Putting memory into the processor is too costly
  - Bus contention
- Another problem: Memory need kept the pace of CPU speedup
- → **“Memory wall”**

Processor Speed Development

- Clock speed curve flattened in 2003
  - Heat
  - Power consumption
  - Leakage
- 3-4 GHz since 2001 (!)
- Speeding up the serial instruction execution through clock speed improvements no longer works
- We stumbled into the **Many-Core Era**



[Herb Sutter, 2009]

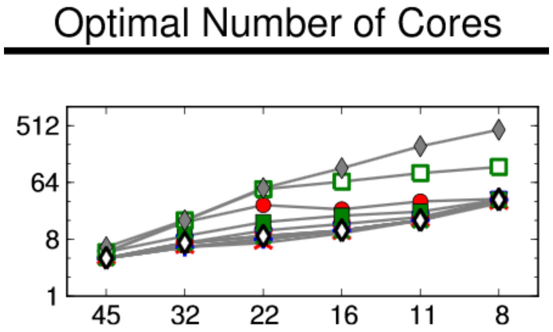
Conventional Wisdoms Replaced

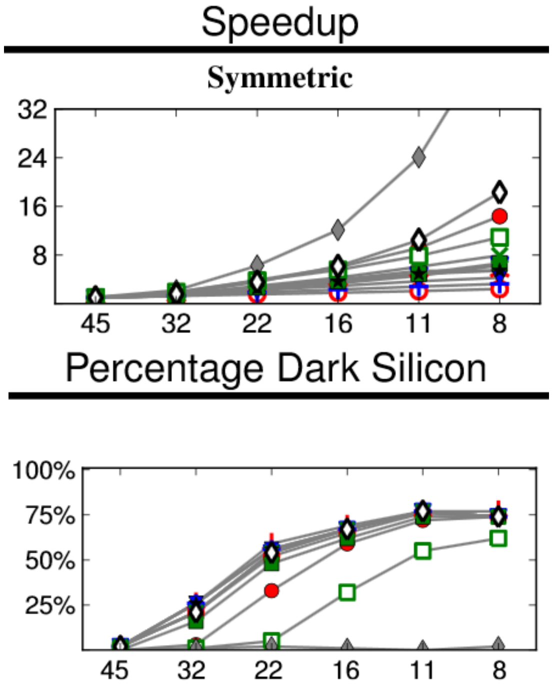
Old Wisdom	New Wisdom
Power is free, transistors are expensive	„Power wall“
Only dynamic power counts	Static leakage makes 40% of power
Multiply is slow, load-and-store is fast	„Memory wall“
Instruction-level parallelism gets constantly better via compilers and architectures	„ILP wall“
Parallelization is not worth the effort, wait for the faster uniprocessor	Performance doubling might now take 5 years due to physical limits
Processor performance improvement by increased clock frequency	Processor performance improvement by increased parallelism

Power Wall 2.0

- Power consumption increases with Moore‘s law, even under constant frequencies
- Cooling is a constant factor
  - Maximum temperature of 100-125 C
  - Hot spots make it worse
- Next-generation processors need to use less power
  - Lower the frequencies
  - Dynamic frequencies scaling (see latest Intel products)
  - Minimize ‚power per bit of I/O‘ [Skadron 2007]
  - Better cache locality, stop moving stuff around
  - Start to use specialized co-processors and accelerators

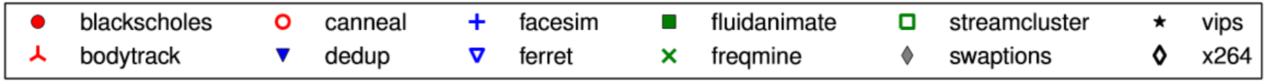
Power Wall 2.0 = Dark Silicon





“Dark Silicon and the End of Multicore Scaling”

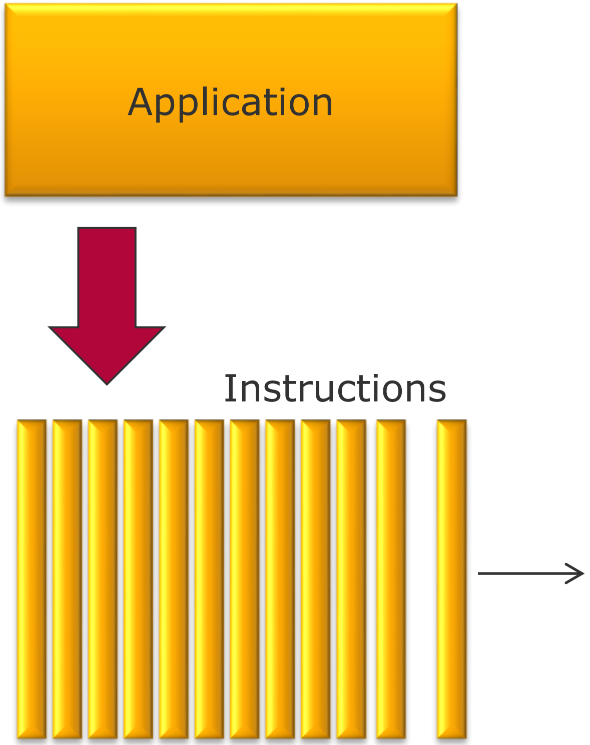
by Hadi Esmaeilzadeh, Emily Blem, Renée St. Amant, Karthikeyan Sankaralingam, Doug Burger



The Situation

- Hardware people
  - Number of transistors N is still increasing
  - Building larger caches no longer helps (memory wall)
  - ILP is out of options (ILP wall)
  - Voltage / power consumption is at the limit (power wall)
    - Some help with dynamic scaling approaches
  - Frequency is stalled (power wall)
  - Only possible offer is to use increasing N for more cores
- For faster software in the future ...
  - Speedup must come from the utilization of an increasing core count, since F is now fixed
  - Software must participate in the power wall handling, to keep F fixed
  - Software must tackle the memory wall

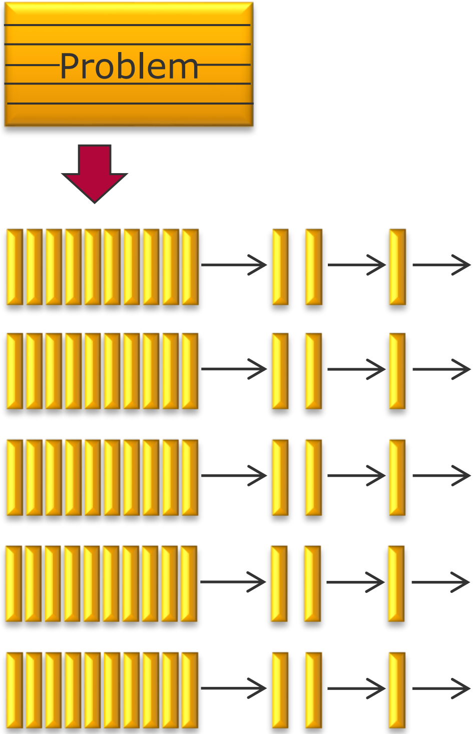
Three Ways of Doing Anything Faster [Pfister]



- Work Harder
  (clock speed)
- Work Smarter
  (optimization, caching)
- **Get Help
  (parallelization)**

Getting Help

- Parallelization not only in computer science
  - Building construction, car manufacturing, large companies
- The basic idea is easy to understand
- Meanwhile tons of options for parallel processing
  - Languages, execution environments, patterns
- Parallelism is a hardware property that must be exploited by software
  - *„A parallel computer is a set of processors that are able to work cooperatively to solve a computational problem.“* (Foster 1995)



HPI Hasso Plattner Institut
Digital Engineering • Universität Potsdam

Thank you
for your attention!