Architecture at the end of Moore

Stefanos Kaxiras

Uppsala University

Conclusions

- There's a power problem and it seems bad
  - Nothing works really well (e.g., multicores)
  - Heterogeous architectures → specialized accelerators
    - Potential path forward to introduce novel devices
- Beware, though, of Amdahl's law:
  - Even if your devices accelerate a function 1000x for 1000x less power, the real-world benefits can be small
- Memory is an area where great advances can bring a revolution in CA
  - Need to break the Fast vs. Vast law
  - Can we embed functionality in memory ?
- Reliability: don't worry about ultra-reliable devices
  - Architects are starting to learn to live with them
    - Future architects could tolerate device errors (stochastic architectures) or (more conventionally) correct them

Outline

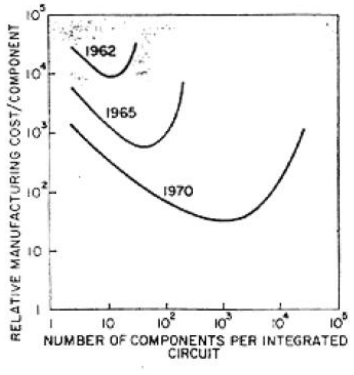
- Introduction
- The power problem
  - Multicores
  - Dark Silicon
  - Heterogeneous architectures
- Amdahl’s Law
  - Performance & Power
    - Moving data costs as much as computing
- The memory problem
  - Fast vs. Vast
  - Locality, Memory Hierarchy (Caches)
- Reliability
  - Near/sub-threshold operation, DIVA
  - Stochastic Architectures

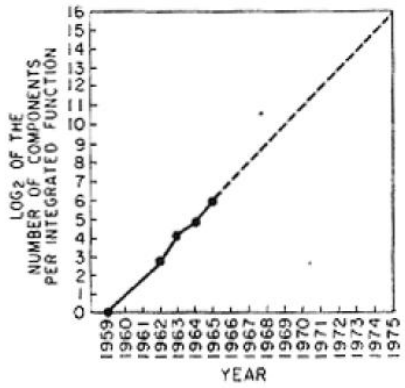
The Role of Architecture

- Architect’s Job:
  - Define the HW/SW interface
    - Layered approach in CS
    - Very hard to break but can be done if the benefit is right
  - Translate technology trends into performance
    - Power,
    - Reliability, …

Translate technology trends into performance:

- Major trend: Moore's Law
  - Scaling feature size, # transistors doubles with each generation (2 yrs)
  - Corollaries to Moore's Law:
    - Smaller transistors are faster → increase in operational frequency
    - Supply voltage and threshold voltage can also be scaled →smaller transistors are more power-efficient

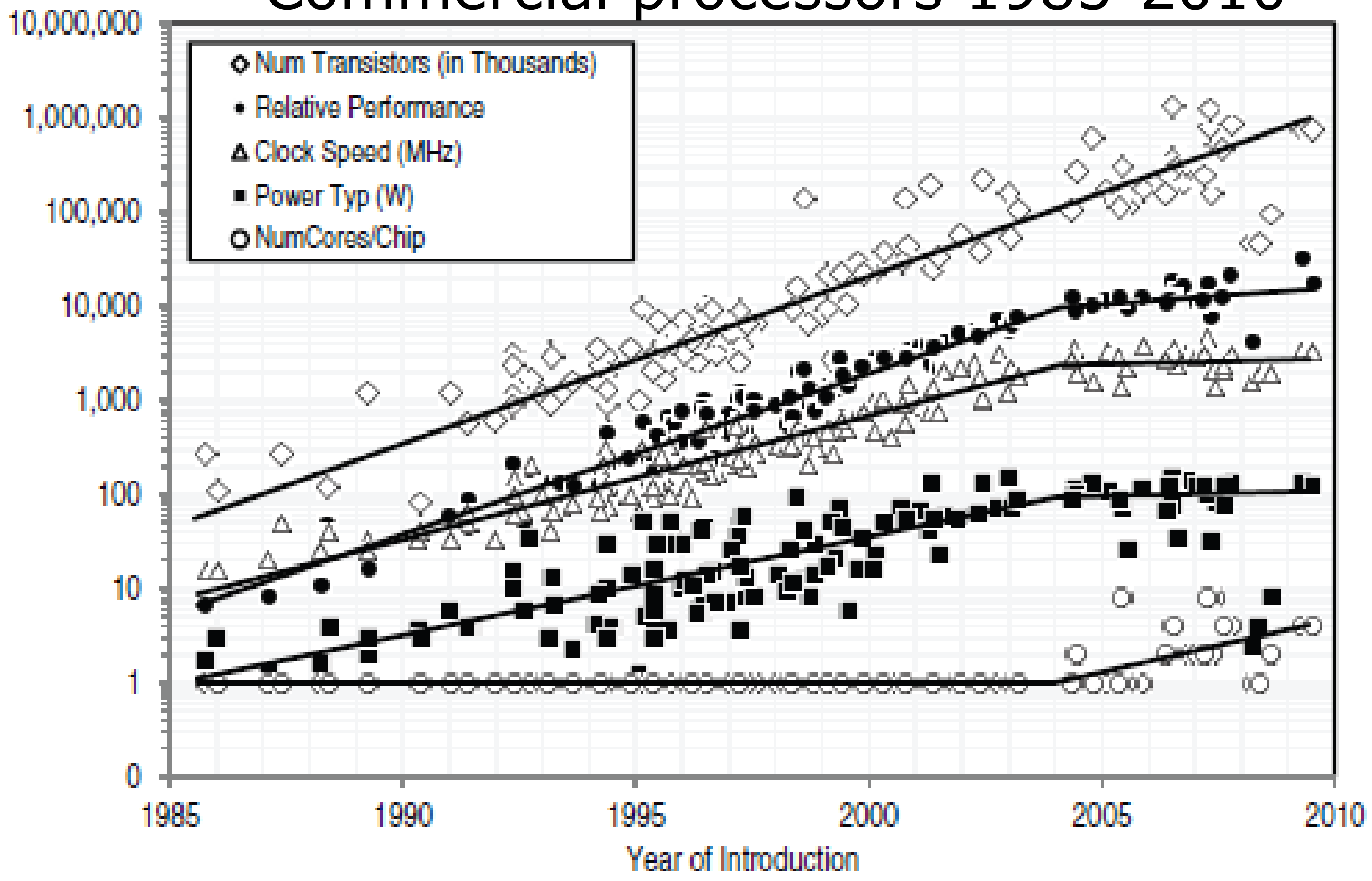




Moore: Electronics, Volume 38, Number 8, April 19, 1965

Moore's Law & Corollaries

Commercial processors 1985-2010



TINSTAAFL

Around 2004 we hit a big wall: Power Wall

… for two reasons:

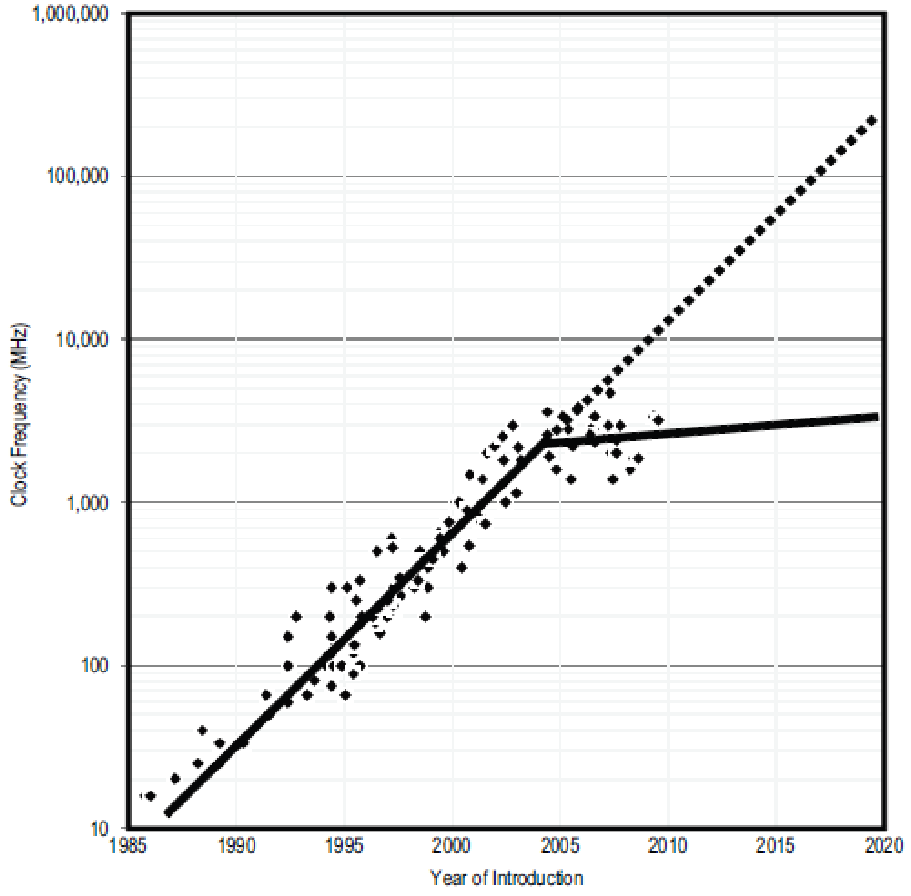
- Break-down of Dennard CMOS scaling:
  - V_{dd} scaling → lower V_{th} → exp. ↑leakage
  - Ideal switch vs. analog device
  - Today: leakage ~40% of total power
- Inefficient single-thread architectures:
  - Diminishing gains for exploding budgets
- Reaction: use Moore transistors for more cores

Performance

The Future of Computing Performance: Game Over or Next Level?
http://www.nap.edu/catalog/12980.html

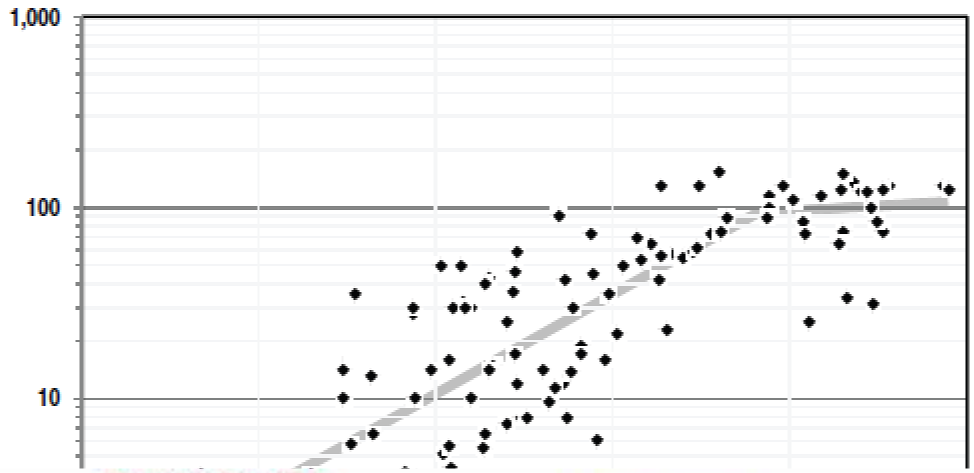
- Performance (1986 to 2008) SPECint2000
- Break in growth rate 2004
- Before 2004: 100x per decade
- Since 2004: 2x per decade
- By 2020: 1,000x "expectation gap" in single core performance

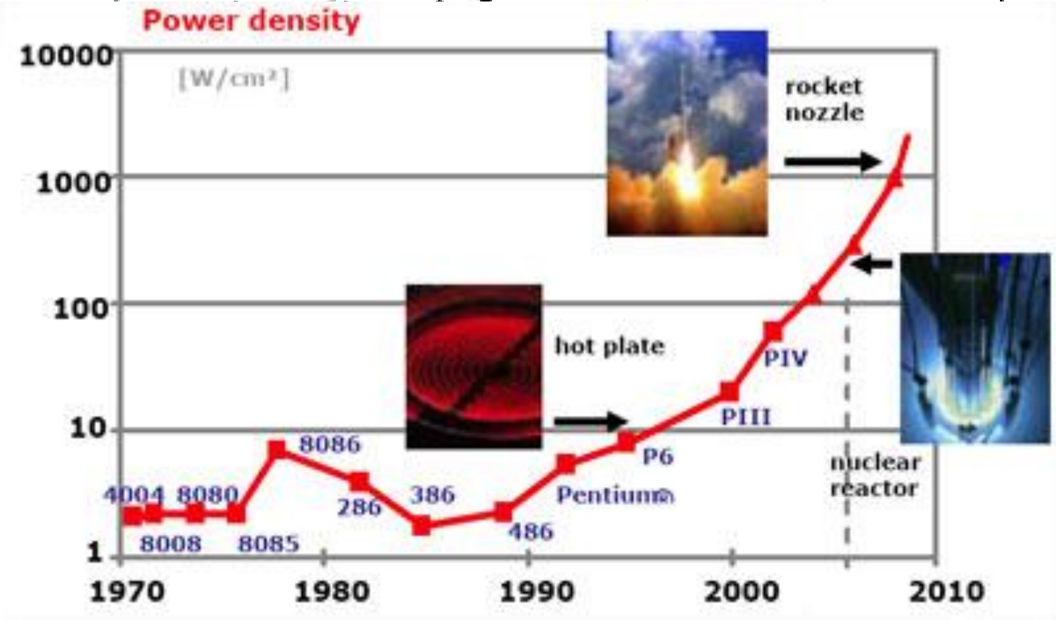
SUMMARY



Power

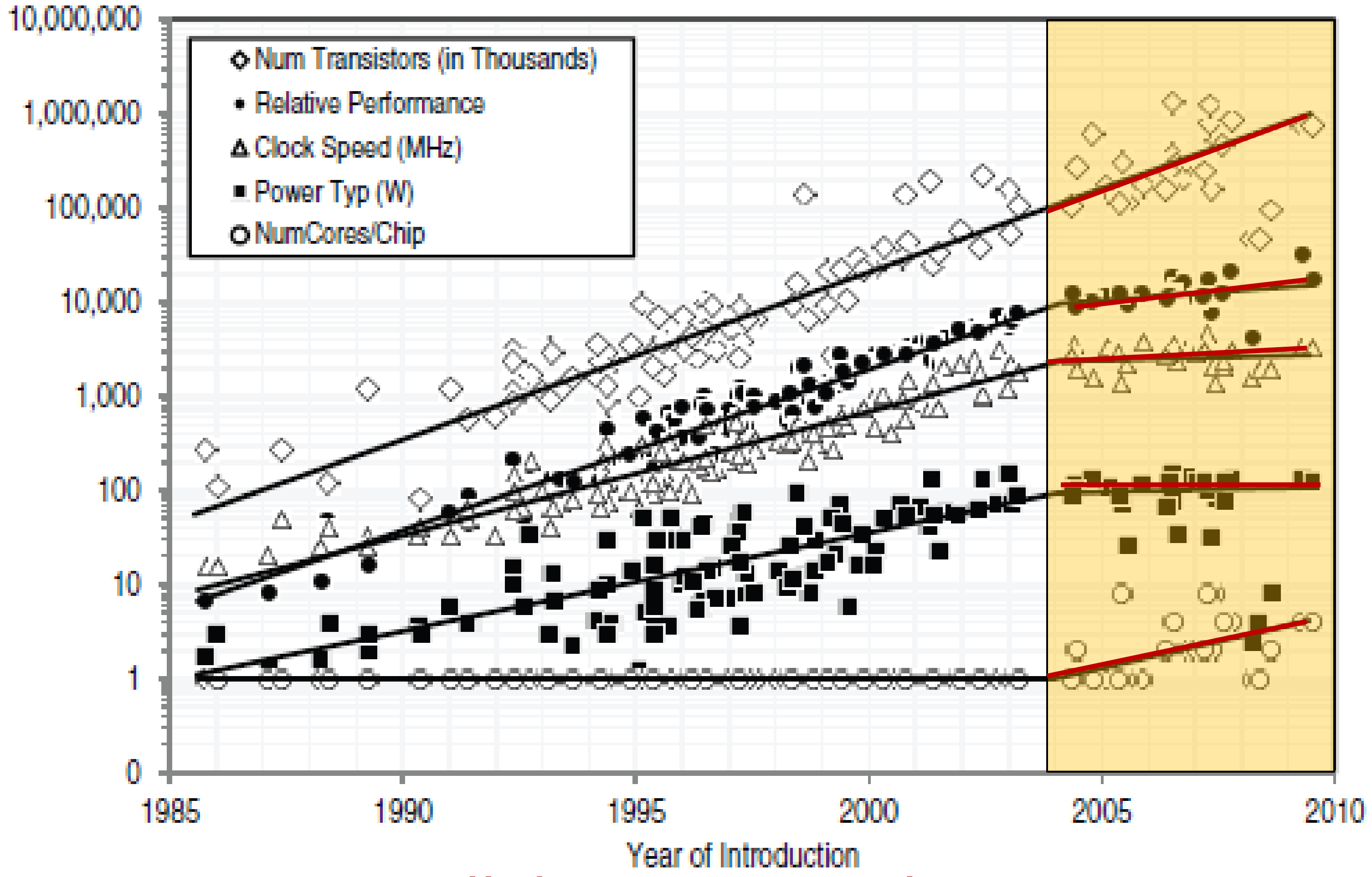
- Frequency scaling has stopped because of power
- Power Density
  - Thermal problems
  - Cooling?





Reaction:

MULTICORE ERA



Parallelism is now the norm.

The Switch to Multicores

- Reaction $\rightarrow$ Multicores (CMPs)
  - No need to scale single processor for performance
  - Use multiple processors to scale overall chip performance
- Constant energy dissipation per instruction
  - Use the growing number of transistors per chip to scale performance while staying within the limit of air-cooling.
- Solves the wire delay scaling problem
- But: coarse grain parallelism (Tasks)
  - Forces parallel programming to the mainstream
  - Difficult to get good speedups
  - Does not address the memory wall

Power defines performance

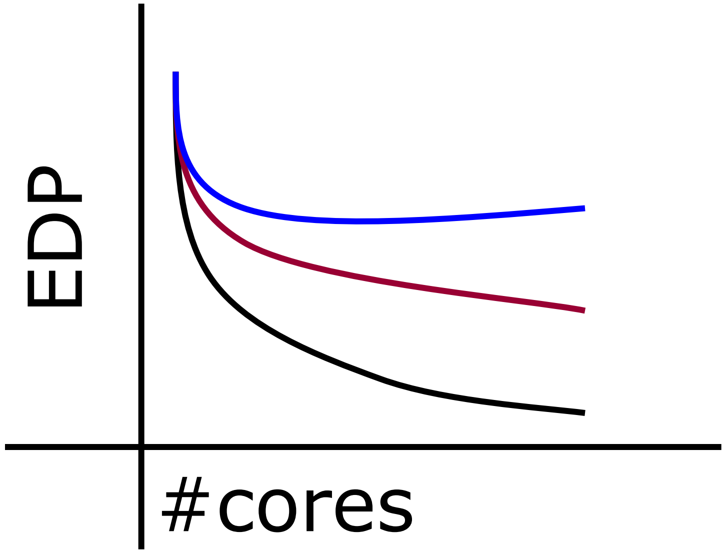
The Power Wall

- Another easy prediction: *Escalating multi-core designs will crash into the power wall just like single cores did due to escalating frequency*

Chuck Moore, AMD, Micro 2008

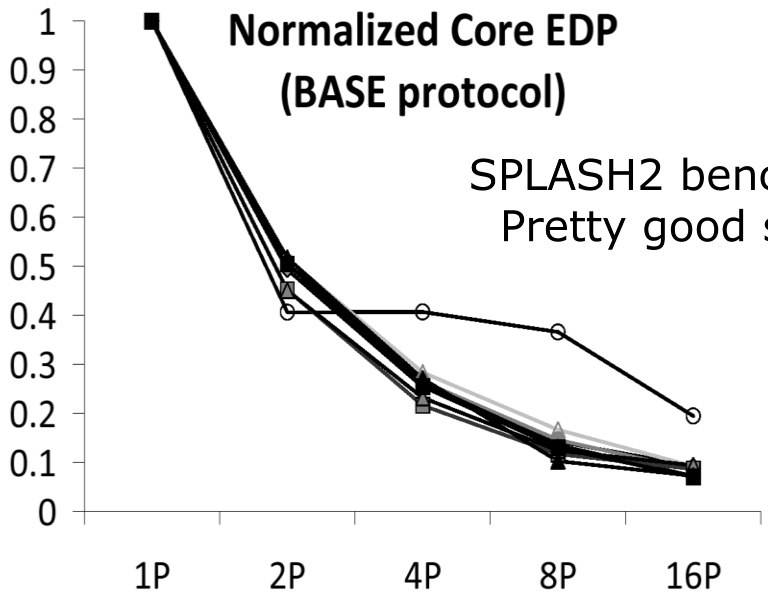
We’ve been saying:

EDP == ENERGY-DELAY PRODUCT
A metric of efficiency that gives equal weight to energy and performance. LOWER IS BETTER.



- Parallelism alone won’t solve the power problem → bound to happen again
- Example: energy efficiency scaling for parallel apps
  - EDP *should* scale with # cores
  - Sub-linear speedups
  - Communication overheads (network energy)

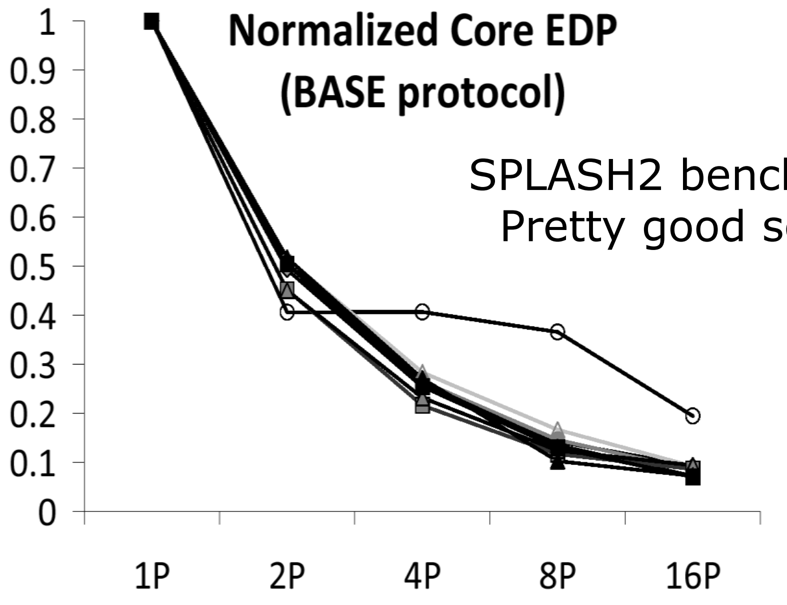
Power-Inefficiency of Coherence



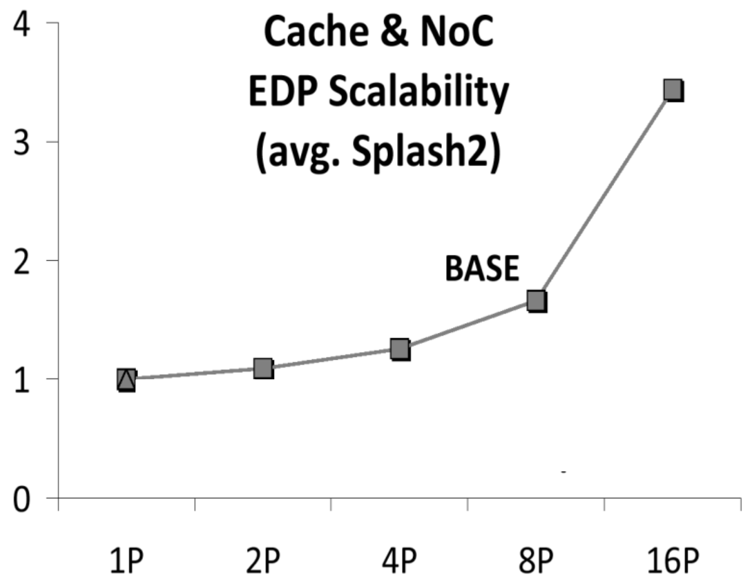
SPLASH2 benchmarks
Pretty good scaling

[Kaxiras & Keramidas, “SARC Coherence: Scaling Directory Coherence in Performance and Power,” IEEE MICRO Sep. ‘10.]

Power-Inefficiency of Coherence



SPLASH2 benchmarks
Pretty good scaling



[Kaxiras & Keramidas, "SARC Coherence: Scaling Directory Coherence in Performance and Power," IEEE MICRO Sep. '10.]

Some grim predictions: Dark Silicon

[Esmaeilzadeh et al. ISCA'11]

device scaling × core scaling × multicore scaling = { Opt. # cores / Speedup / % dark silicon }

ITRS & conservative predictions

Pareto curve of core perf. vs. TDP (avg. all SPEC)

Parsec benchmarks Scaling constrained by Amdahl's Law

- Dark silicon predictions:
  - Regardless of chip organization and topology, multicore scaling is power limited:
    - 22nm → 21% of chip must be powered off
    - 8nm → 50%.
  - 2024: 7.9x, 24 times less than 2x per 1.5 year

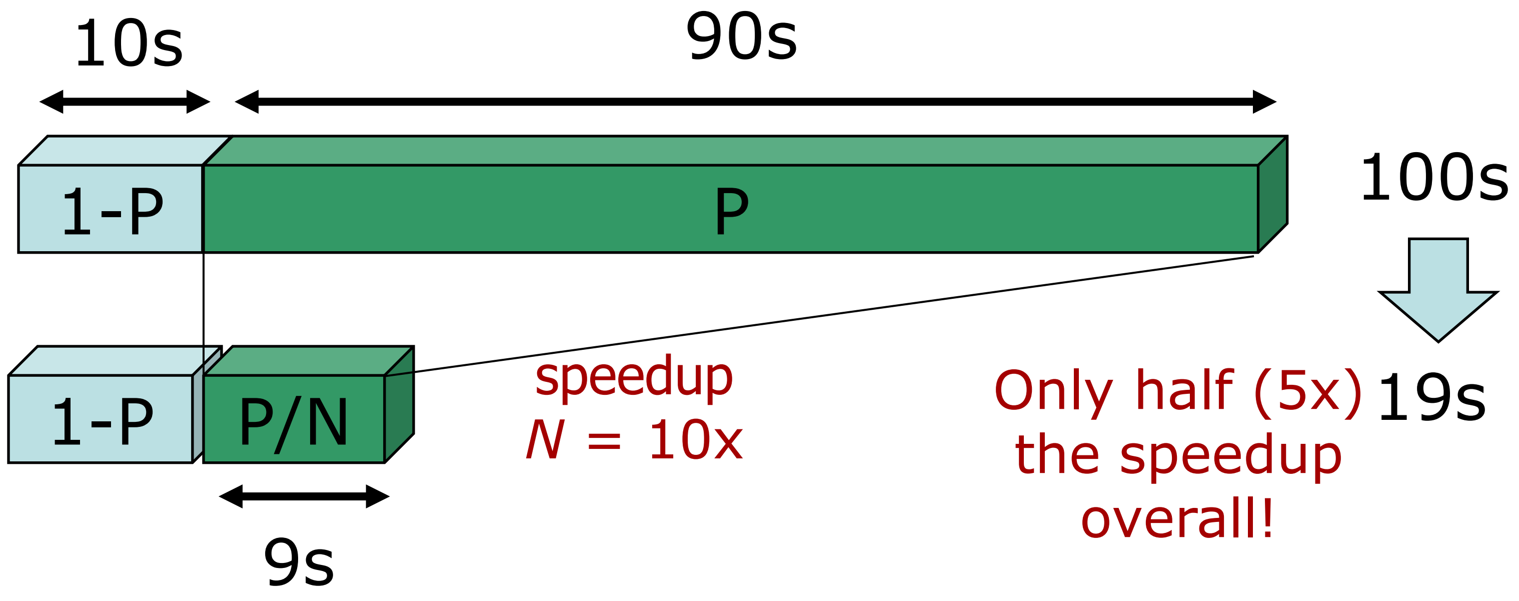
Specialization: Heterogeneous Architectures

- Power-efficient "accelerators" for specific functions: graphics, encryption, communications, application-specific accelerators …
- Have lots of them on the chip, powered off most of the time
- Pathway to introduce accelerators based on new devices

Outline

Accelerators and Amdahl's Law

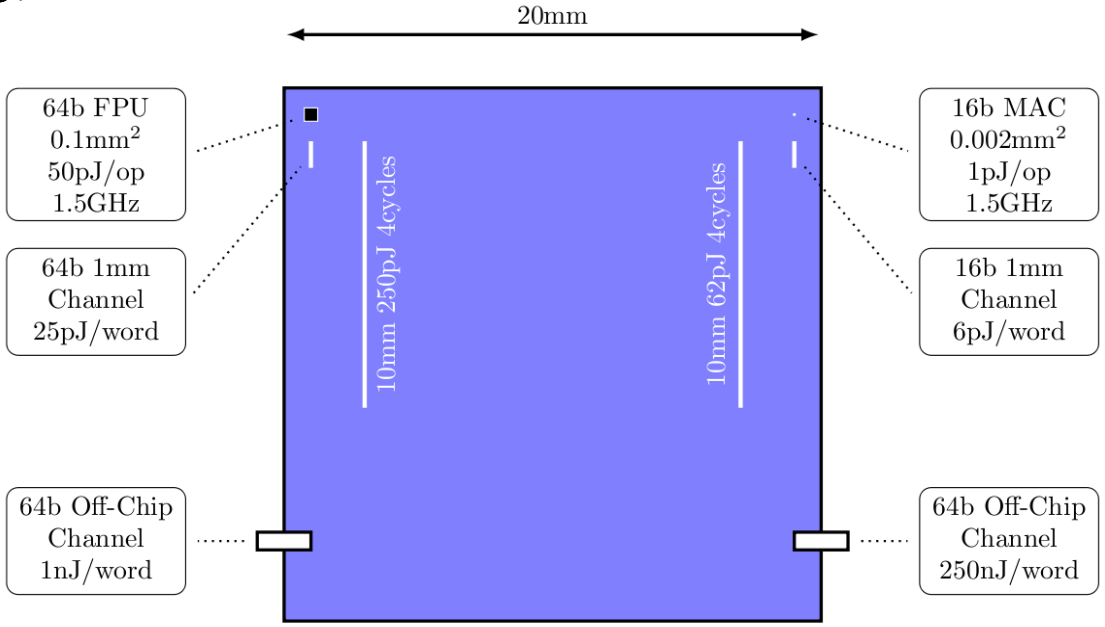
- Accelerating only part of a problem leads to smaller speed-ups (Amdahl's law):



- Speedup = 1 / [(1-P) + P/N]
- Limits the scaling of accelerators

Moving data

- Key limitation in GPU performance → power consumption
- Ops vs. data transfer over large distances on/off chip
- Compare area and energy:
  - 16-bit MAC
  - 64-bit FPU
  - channels



Bill Dally, International Conference on Supercomputing 2010

- FP: 10x more energy-efficient than moving a word .5 die length (e.g. from the LL-cache)
- 16b MAC: 100x
- Off-chip: 40x more!
- ALSO: Wire delays do not scale as fast as transistor speeds

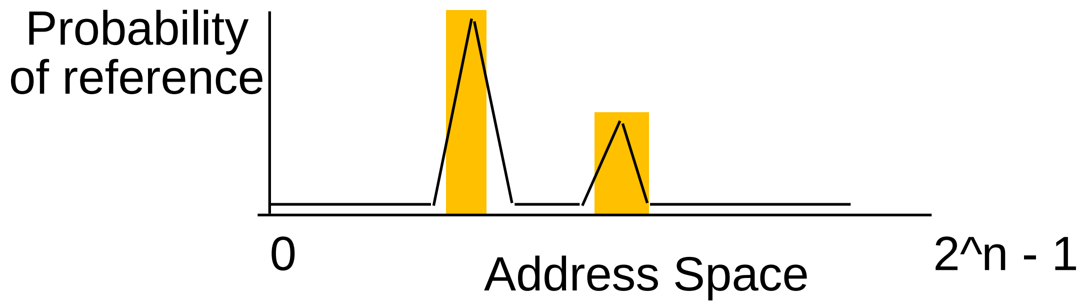
Outline

The Problem with Memory

- Memory → VAST or FAST
  - Flip-Flops (registers)
  - 6-T SRAM (cache)
  - Capacitor-based DRAM (main memory)
- Fundamental characteristic of Si-based memory: bigger → slower
  - Bigger arrays: wire delay (word-lines, bit-lines)
  - The denser the technology the slower it is to detect the logic values stored (sense-amps)

What Saves Us …

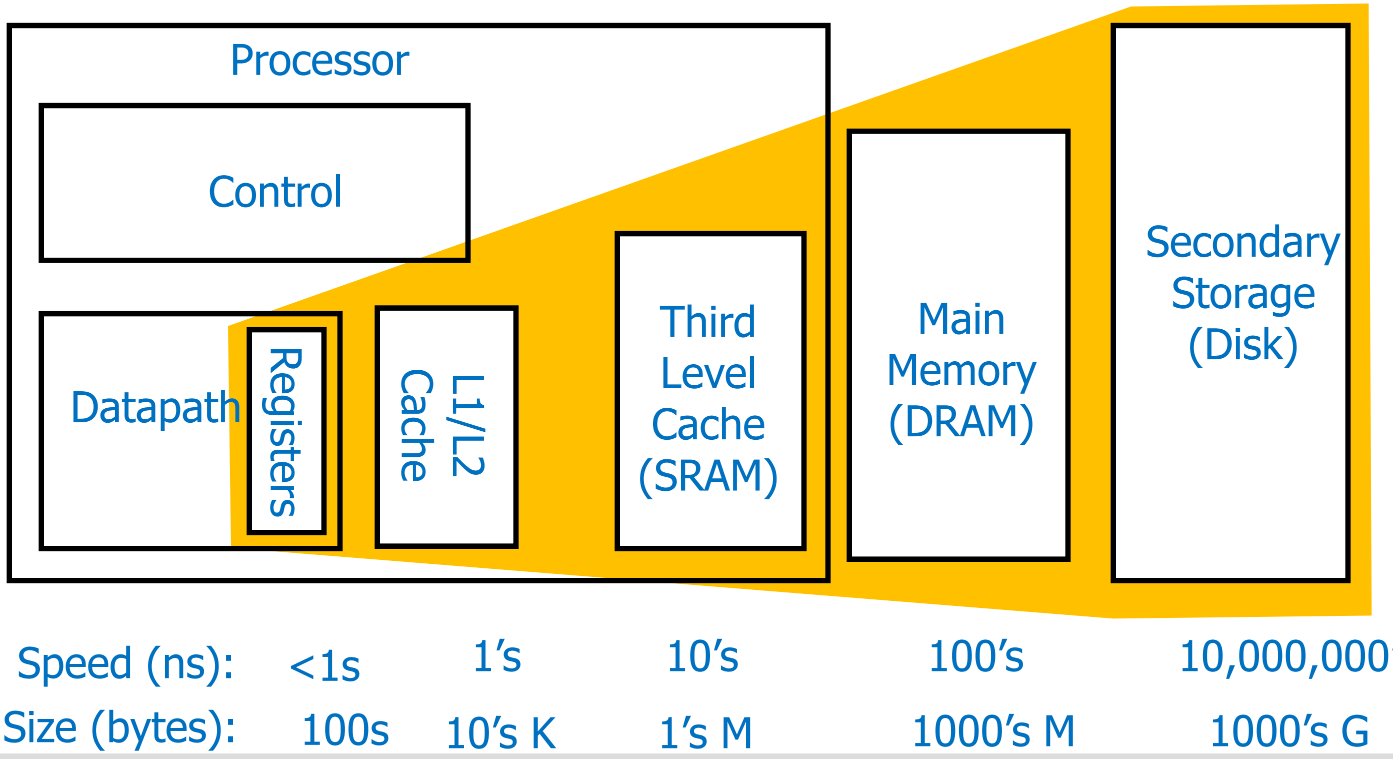
- The *Principle of Locality*:
  - Program accesses a relatively small portion of the address space at any instant of time.



- Memory Hierarchy:
  - Provide the illusion of a fast, large, and cheap memory system -- most of the time.

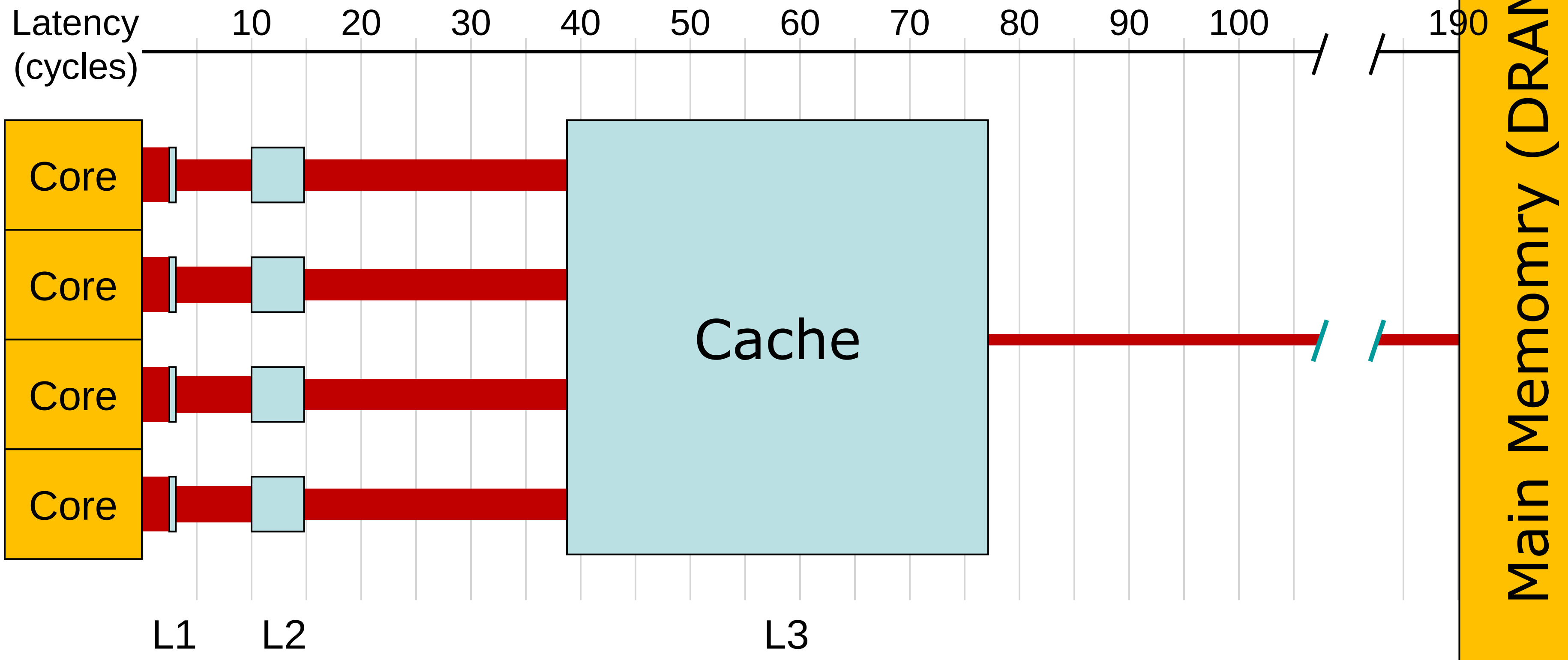
Memory Hierarchy

- By taking advantage of the principle of locality:
  - Present the user with as much memory as is available in the cheapest technology.
  - Provide access at the speed offered by the fastest technology.



Multicore Memory Hierarchy

Intel Nehalem 3GHz (2009)



D. Molka, et. al., *Memory Performance and Cache Coherency Effects on an Intel Nehalem Multiprocessor System*, PACT 2009.

Modern processors go through hoops for performance

- Instruction-Level-Parallelism
- Out-of-Order (OoO) execution
- Very expensive to scale:

  Width (# of instructions in parallel) & Instruction Window (pool of instructions to choose from for OoO)

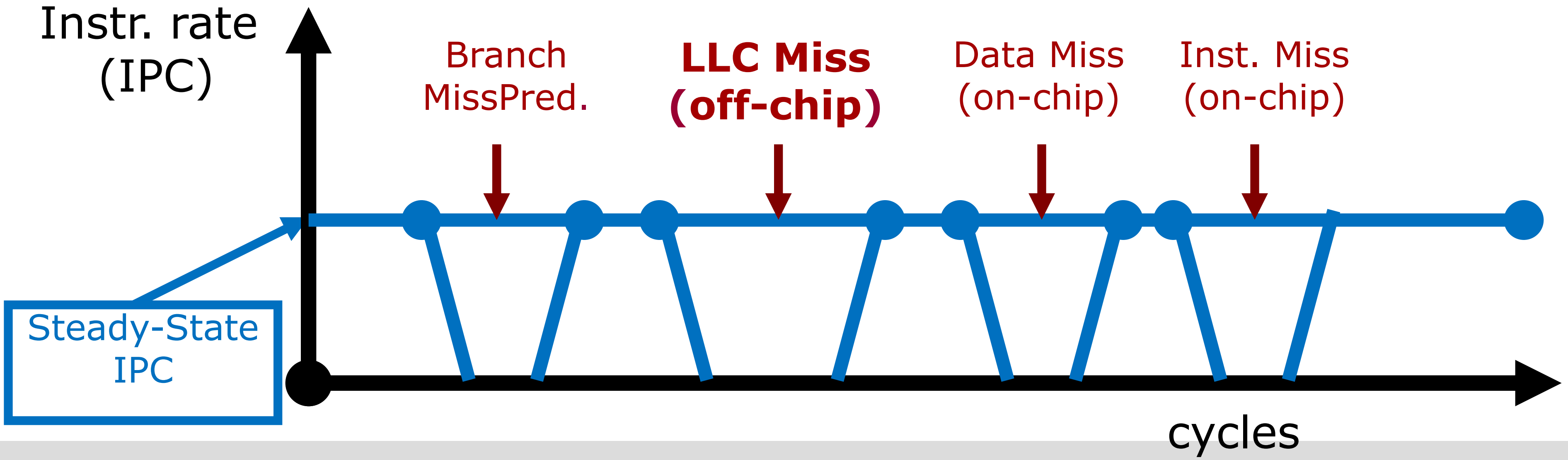
  - Power grows exponentially to these parameters
  - Power dissipation scales as performance raised to the 1.73 power
  - Pentium 4 ~ 6x i486 performance; 24x power!

OoO Requirements

- OoO requires tremendous support
  - Transistor budget → M transistors
  - Complexity
- Architectural techniques required:
  - Large Instruction Windows 100s instr. (associative searches)
  - Branch prediction → Speculative execution
  - Large register files and Register renaming
  - Load/Store queues (associative searches)
  - Reorder buffer (sequential semantics)
  - Checkpointing of state (atomic semantics)

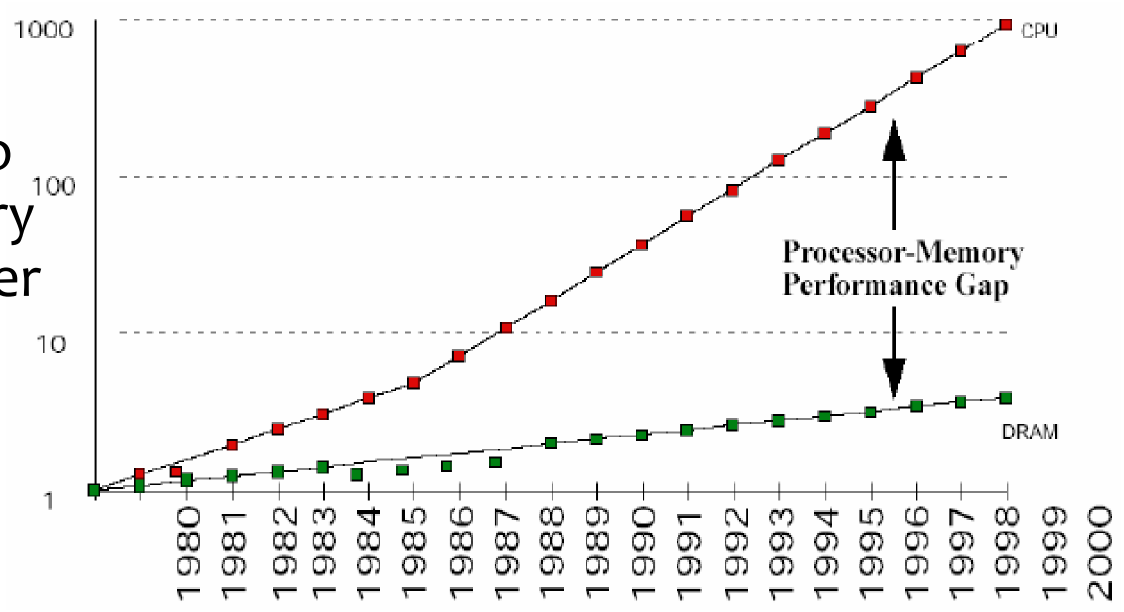
OoO Performance

- Interval-based models → break the execution time of a program to intervals
  - Steady-state intervals: the IPC is limited by the machine width and program's ILP
  - Miss-intervals: introduce stall cycles due to branch mispredictions, on-chip instruction/data misses, LLC misses (off-chip misses)



Memory vs. Processor Performance Trends

- Memory performance
  - 1978: Used to be faster to get the data from memory than to add them together
  - 2008: Fetching a number from memory ~ 600 additions
  - The Memory WALL
  - Memory ultimately limits the performance we can get from frequency scaling

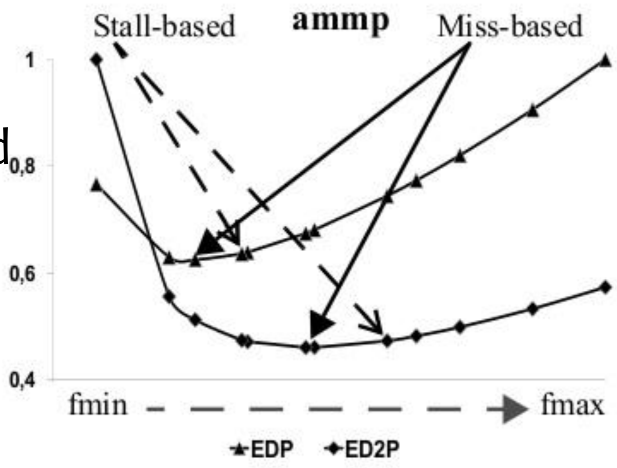


“A Case for Intelligent RAM: IRAM,” by David Patterson et. al.

Implications on Power

- Memory Hierarchy is not only a performance optimization but also a power optimization → brings useful data close to CPU, reduces data transfers over long wires
- Memory behavior, at first order, defines both performance and power
  - CPU stall cycles → *slack* that can be exploited by reducing operational frequency → DVFS
- OoO execution inefficiency + memory stalls = power inefficiency

EDP: energy delay product (lower is better)



Where Locality Fails ?

- Many interesting problems have little locality:
  - Sparse matrix, …
  - Sorting,
  - Traversing complex data structures,
  - Indexing,
  - Data mining,
  - Google search,
  - Going over vast data sets performing very little computation,
  - Parallel programs with lots of communication
- Supercomputers of the past made it a point not to rely on locality
  - CRAYs, Cray T3E, … → no caches!
- Digital content created in 2010 ~ 1000 EB ! (1 billion TB)

Some Possible Directions:

- Processing in Memory (PIM)
  - Failed in the past, mainly because tried to do computation in memory → slow and not so dense
- IRAM (Kozyrakis et al. ISCA 1997)
  - Tried to combine DRAMS and vector processing
  - Exploits tremendous DRAM bandwidth
- IPSTASH (Micro 2004)
  - SRAM+little logic to execute IP-Lookup in route-table memory
  - More power-efficient than TCAM (content addressable)

- Very dense (VAST) memories with simple processing capabilities
  - Comparisons, make simple decisions, move data
- Very high connectivity/bandwidth

Outline

- Introduction
- The power problem
  - Multicores
  - Dark Silicon
  - Heterogeneous architectures
- Amdahl's Law
  - Performance & Power
    - Moving data costs as much as computing
- The memory problem
  - Fast vs. Vast
  - Locality, Memory Hierarchy (Caches)
- Reliability
  - Near/sub-threshold operation, DIVA
  - Stochastic Architectures

Systems will rely on unreliable components

Reliability

- Architecture obsessed with reliability
  - Big margins (voltages, frequencies, …)
  - Triple redundancy
  - Error Correction Codes (ECC)
- Power vs. Reliability → reducing supply voltage makes HW prone to failures
- Variability (speed, leakage) increases with scaling
- Errors can be detected or tolerated

Detecting Errors

- Operating at Sub-Critical Voltages:
  - Lower Vsupply below the critical voltage for a certain frequency

- Razor flip-flop (Ernst et al. Micro 2003):
  - Double sample and detect timing errors
  - Correct errors at the architectural level (flush the pipeline and restart)
  - Implemented by ARM
  - Still … preserves reliability

Tolerating Errors: Stochastic Architectures

(R. Kumar et al.)

Insight#1:

- A large class of emerging client-side (in field) applications have inherent algorithmic/cognitive noise tolerance.
  - Processors can be optimized for very low-power instead of always preserving correctness.
    - Errors tolerated by the applications instead of spending power in detecting/correcting errors at the circuit/architecture level.

Insight#2:

- If processor designed to make errors gradually instead of catastrophically, significant power savings possible
  - E.g., when input voltage is decreased below critical voltage (voltage overscaling). for power reduction.

Embracing Unreliability

- What if we could embrace uncertainty (unreliability) ?
  - Difficult for the general case
  - But could find “killer” apps where it does not matter
- Example 1: Fault Tolerance in Cortical Microachitectures (ISCA 2011)
  - Shows how faults in GPUs are irrelevant for bio-inspired algorithms
- Example 2: Operating at Sub-Critical Voltages but allowing errors to happen (FP7 LPGPU project).
  - Many operations in GPUs can afford errors → reduced QoS
  - Others cannot → these are protected by having two supply voltages (critical & sub-critical)

Recap

- Unreliable components might not be so bad
- While reliability has been a must in architecture, recently we are exploring algorithms and architectures that do not demand it
- Depending on the application it may be too expensive to guarantee reliability
- Good news for novel devices!

Summary

- "Around 2004, 50 years of exponential improvement in the performance of sequential computers ended."
  K.Olukotun and L. Hammond, "The Future of Microprocessors," *ACMQueue*,Vol. 3(7), 2005
  - Power (increase in leakage, architectural inefficiency), Wire delay
- Multicores force the world to think parallel
  - But parallel programming hard, most apps not scalable?
  - Highly parallel accelerators → power efficiency; but beware of Amdahl's Law
- Memory is still the major limiting factor for performance
  - Immediate impact: any technology that can improve on that …
  - Maybe some applications need to run in memory
- Reliability was and is a must in many cases, but new algorithms and architectures are now considered

Summary

- Novel devices → great potential for power efficiency
  - E.g., move quantum properties not charge
- What can we architect?
  - Direct replacement for switches?
  - Specialized apps/accelerators
  - Radically new devices?
- Unreliability the new reality