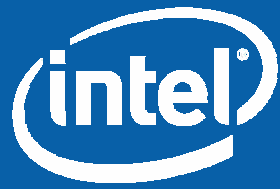


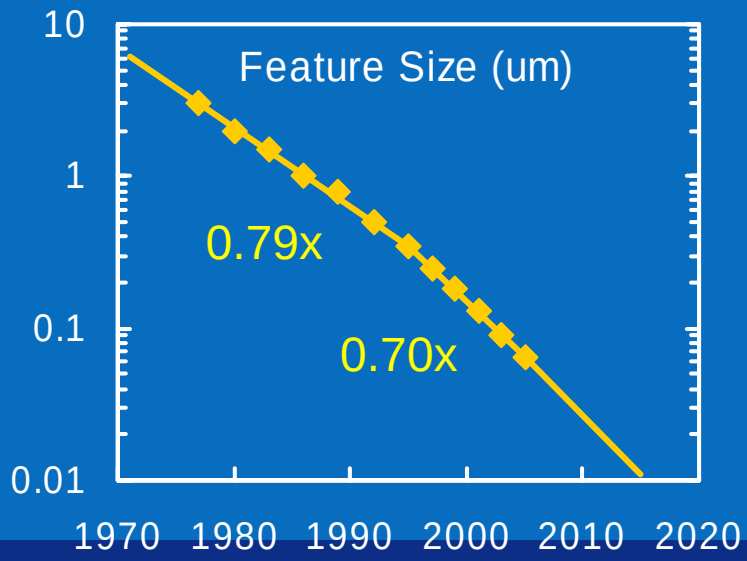
Why Intel is designing multi-core processors

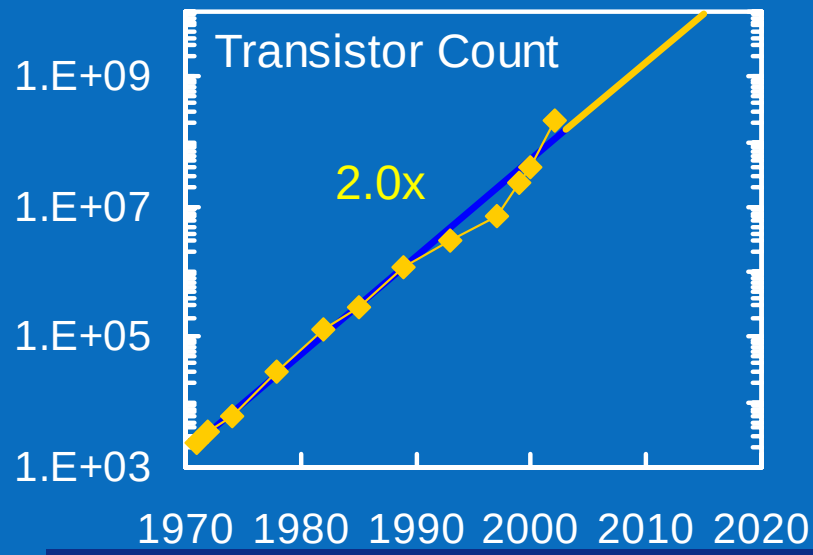
Geoff Lowney

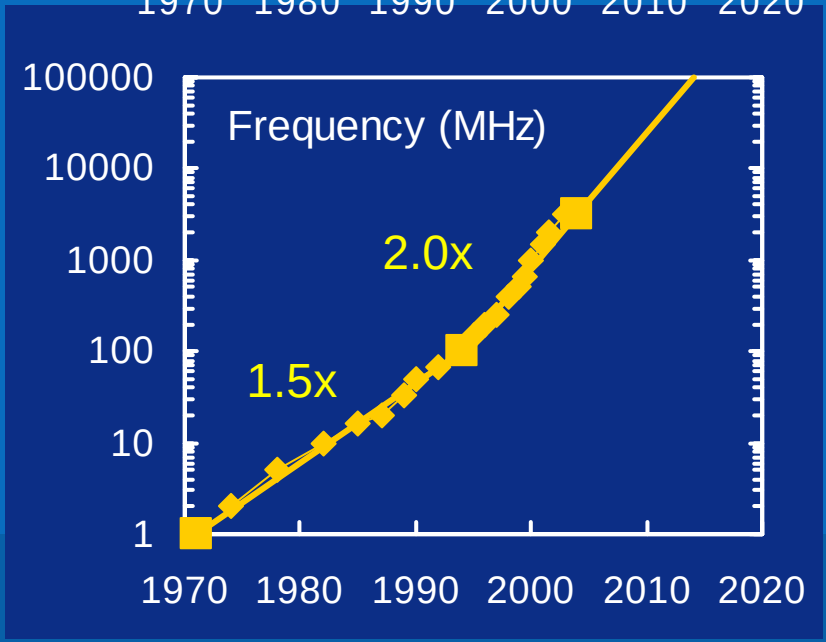
Intel Fellow, Microprocessor Architecture and Planning

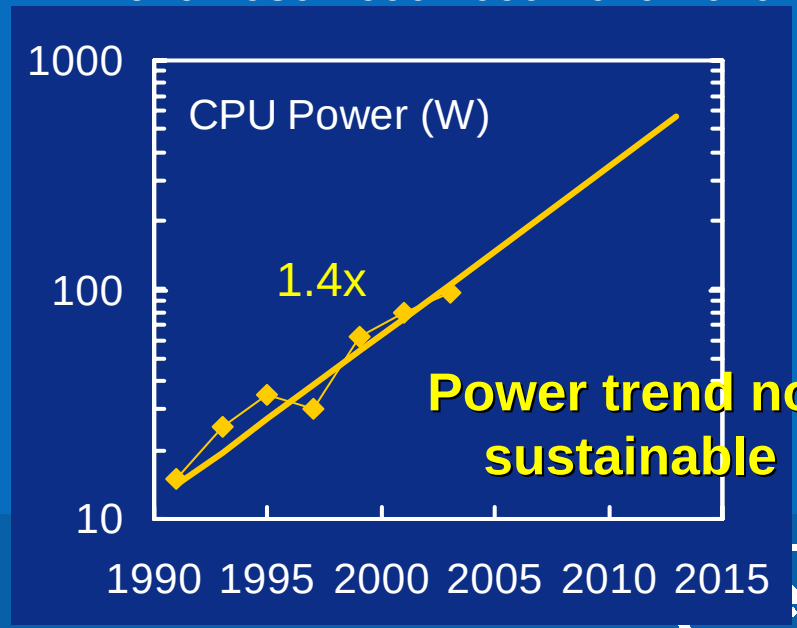
July, 2006

Moore's Law at Intel 1970-2005



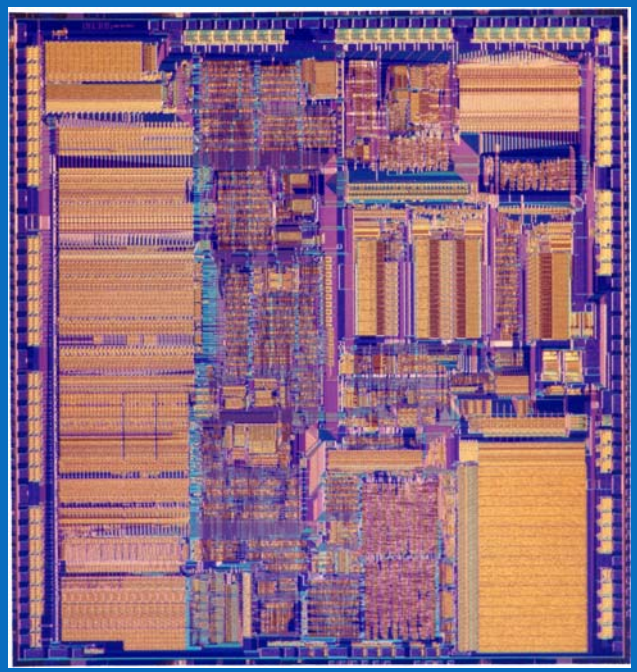


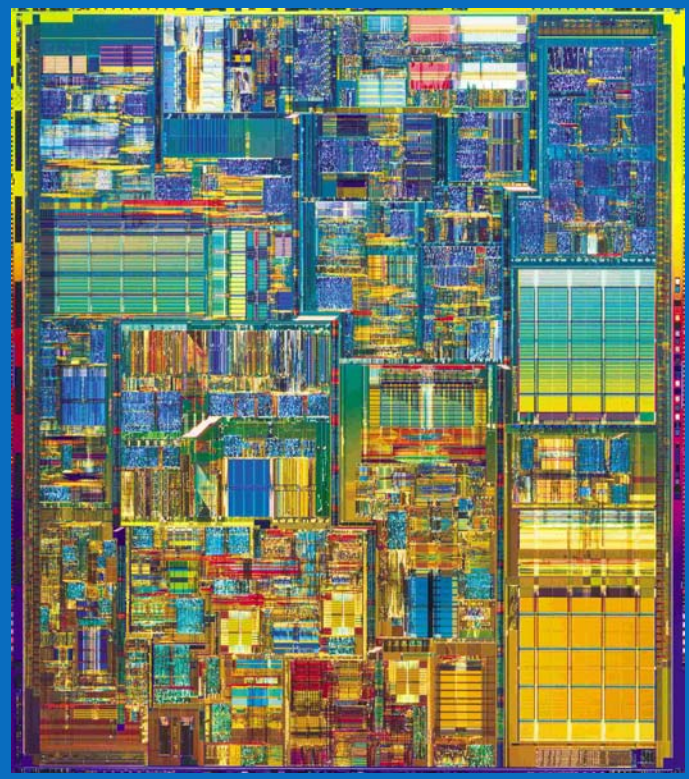




Pentium® 4 Processor

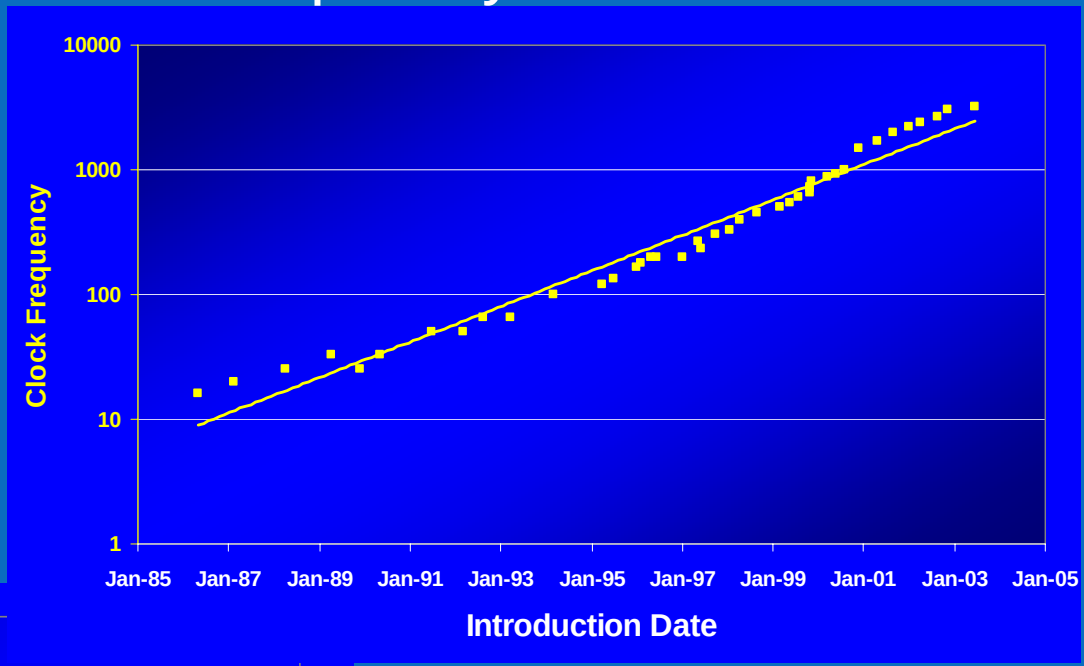
386 Processor



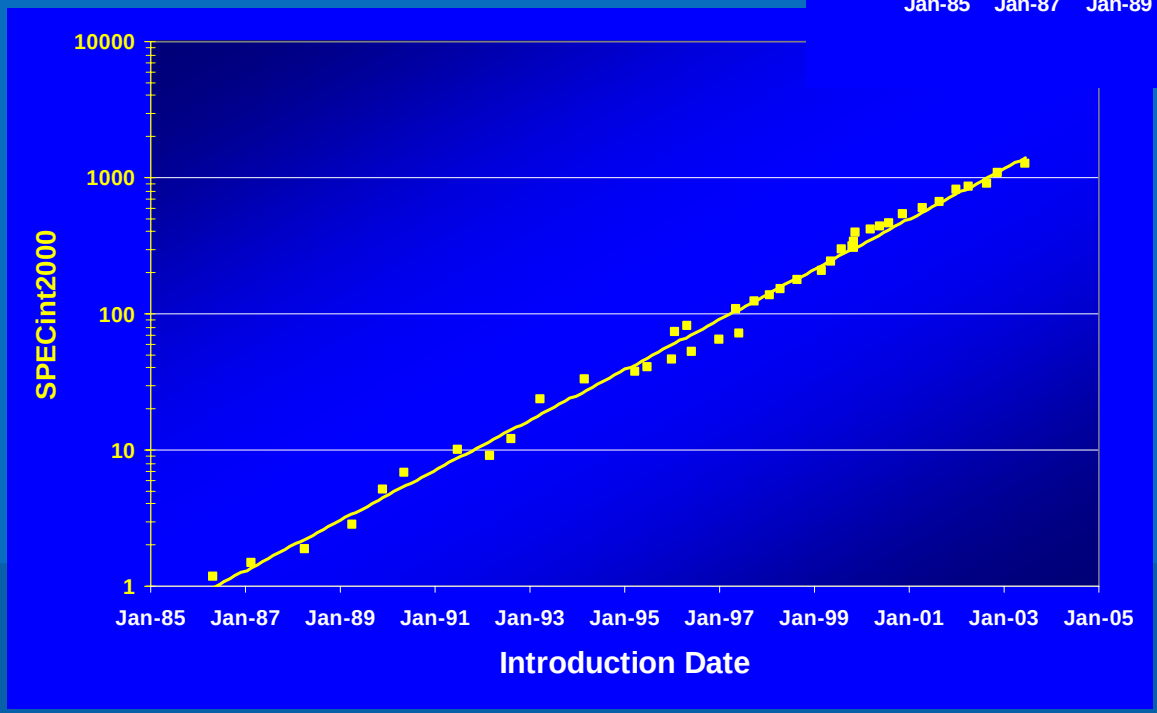


May 1986	17 Years	August 27, 2003
@16 MHz core	200x	@3.2 GHz core
275,000 1.5µ transistors	200x/11x	55 Million 0.13µ transistors
~1.2 SPECint2000	1000x	1249 SPECint2000

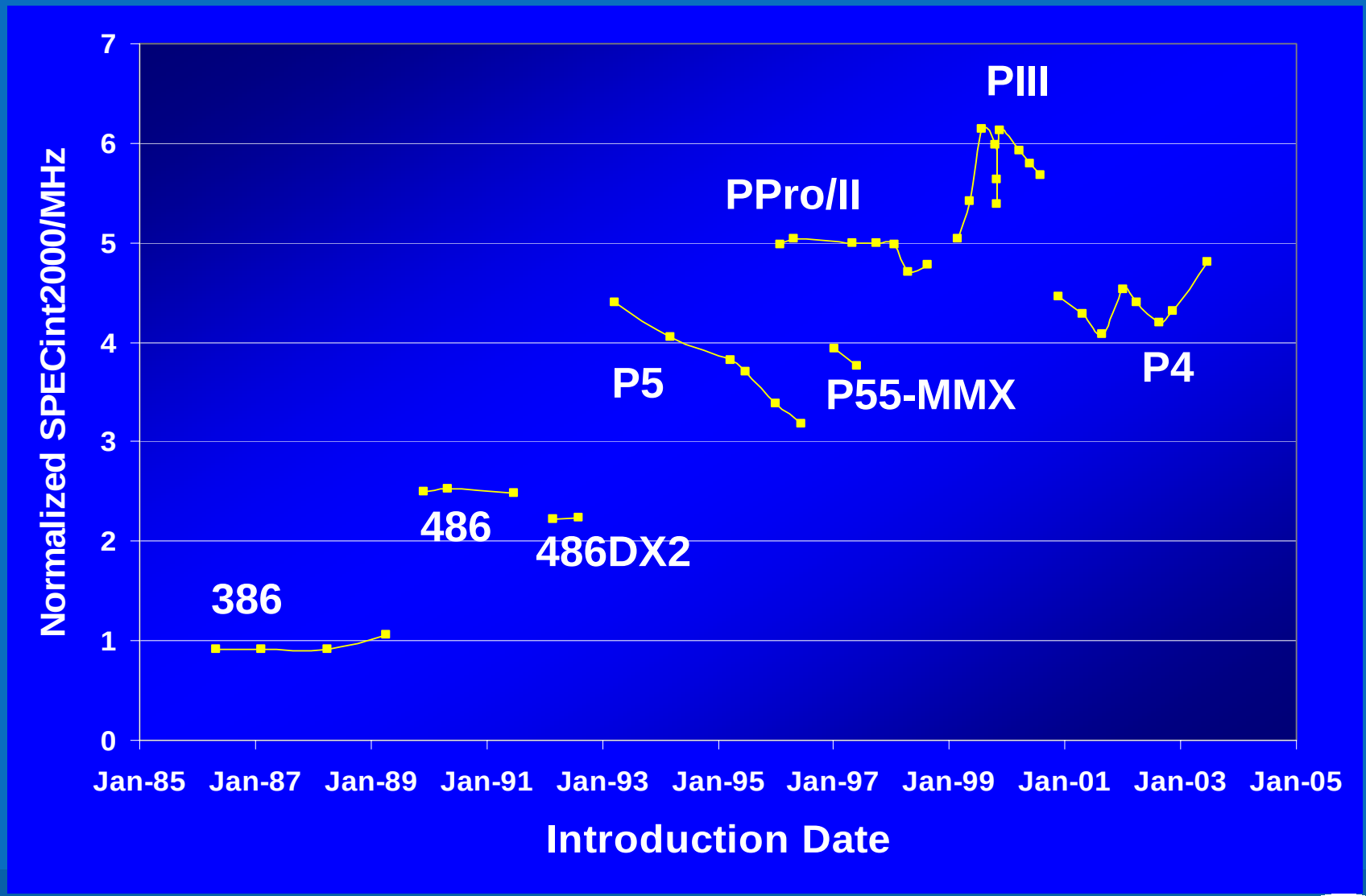
Frequency: 200x

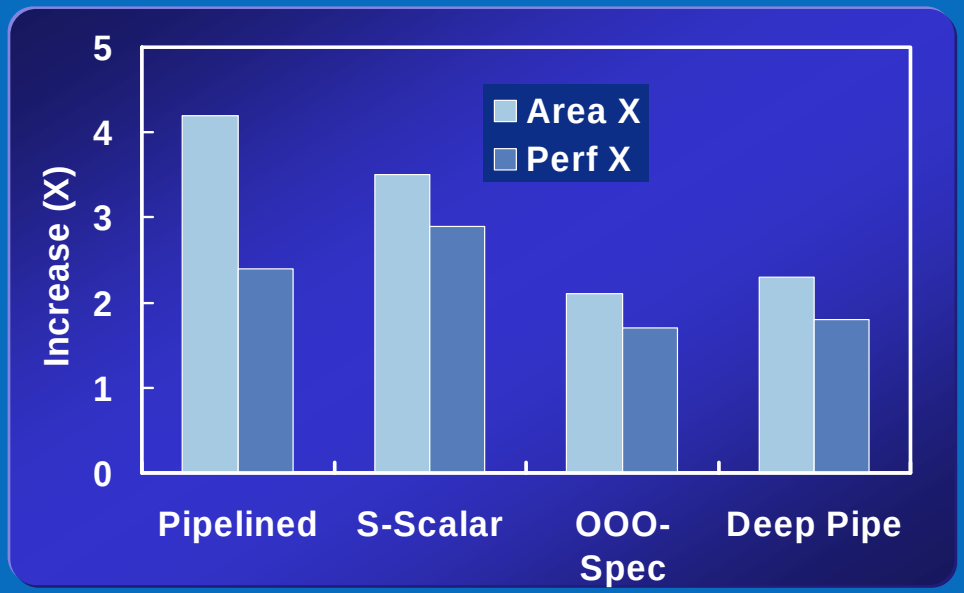


Performance: 1000x



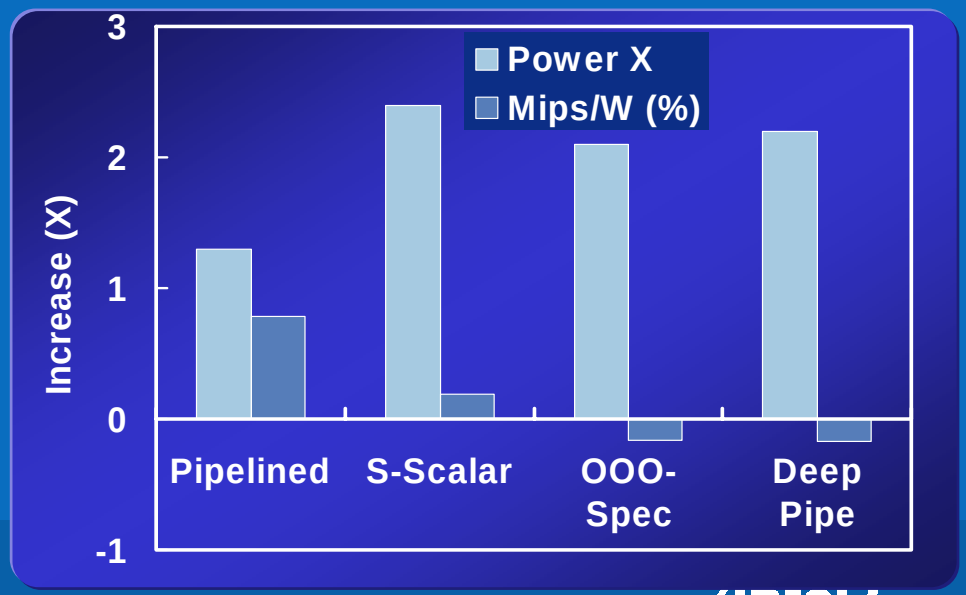
SPECint2000/MHz (normalized)



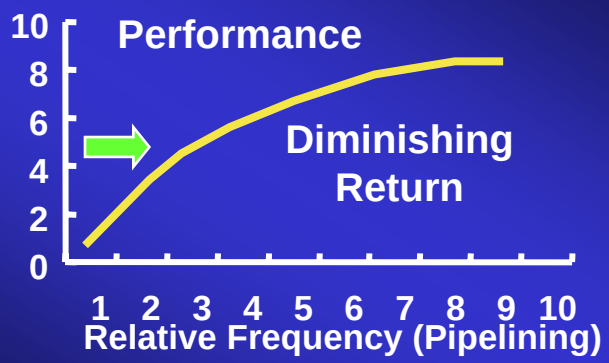


Performance scales with area**.5

Power efficiency has dropped



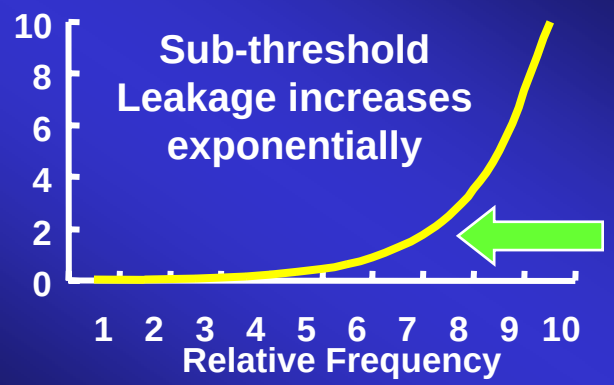
Pushing Frequency



Pipeline & Performance

Maximized frequency by
- **Deeper pipelines**
- **Pushed process to the limit**

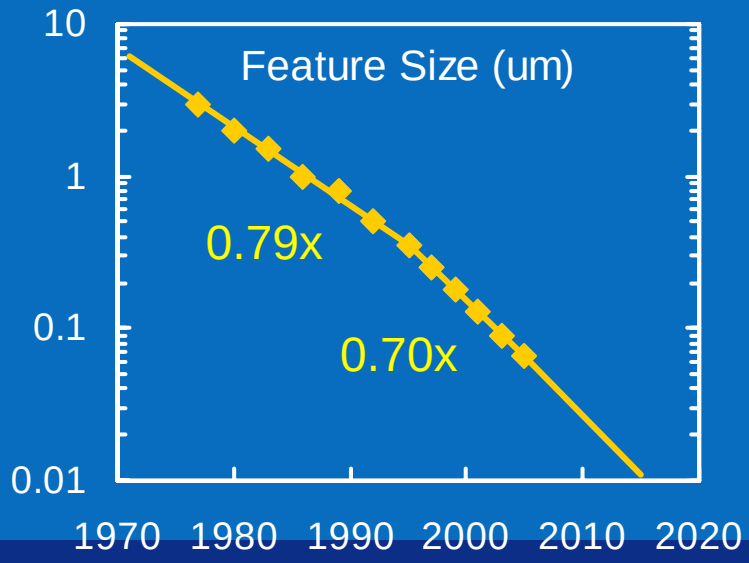
- **Dynamic power increases with frequency**
- **Leakage power increases with reducing Vt**

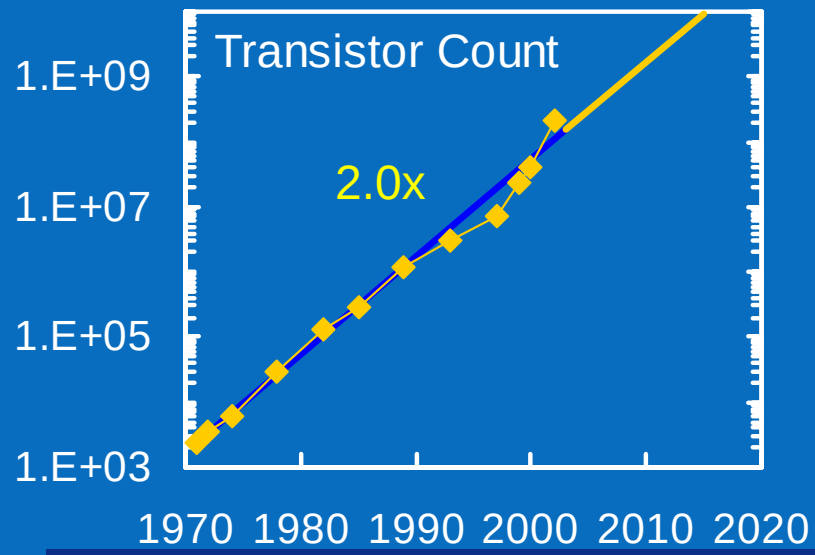


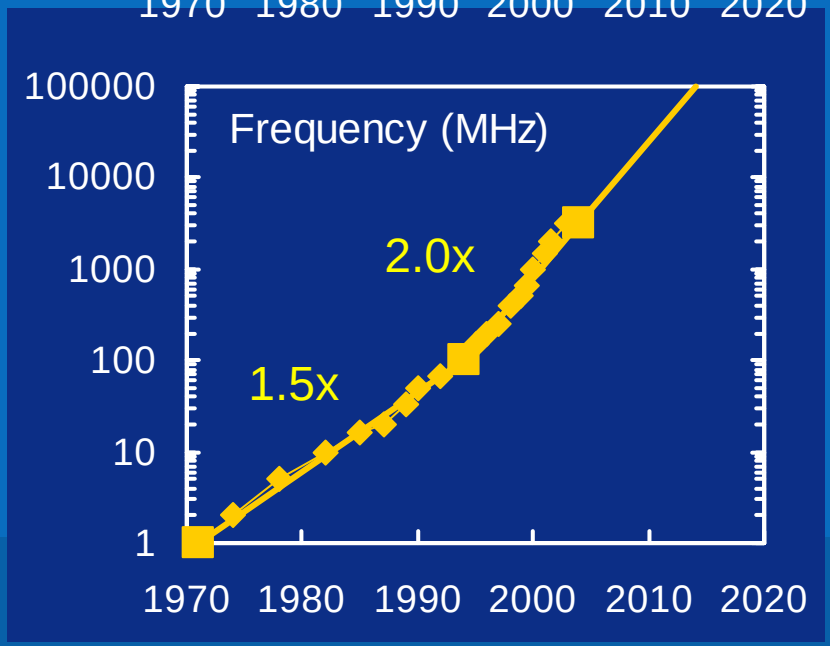
Process Technology

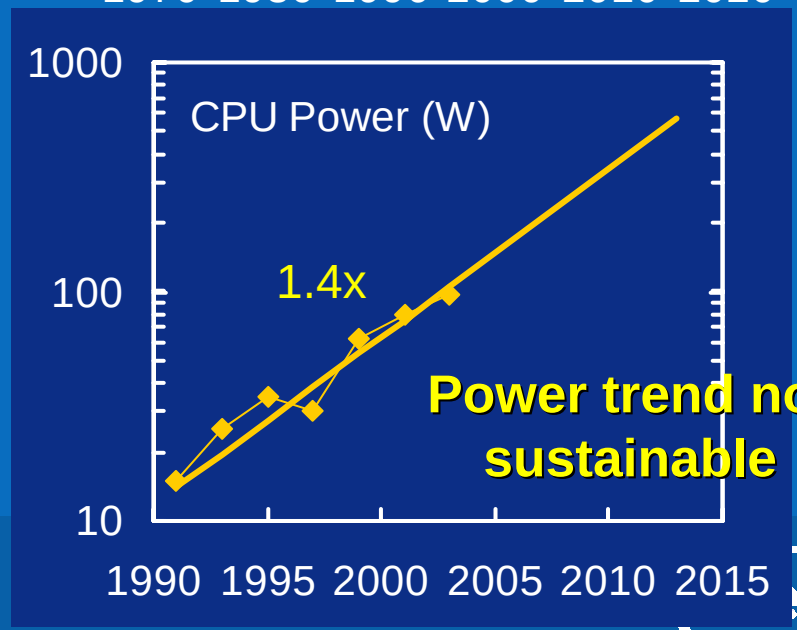
Diminishing return on performance. Increase in power

Moore's Law at Intel 1970-2005









Reducing power with voltage scaling

Power = Capacitance * Voltage**2 * Frequency

Frequency ~ Voltage in region of interest

Power ~ Voltage ** 3

10% reduction of voltage yields

- 10% reduction in frequency
- 30% reduction in power
- Less than 10% reduction in performance

Rule of Thumb

Voltage	Frequency	Power	Performance
1%	1%	3%	0.66%

Dual core with voltage scaling

A 15% Reduction In Voltage Yields

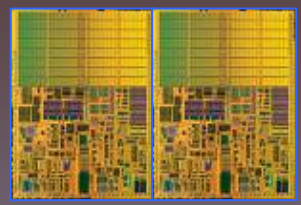
RULE OF THUMB

Frequency Reduction	Power Reduction	Performance Reduction
15%	45%	10%

SINGLE CORE

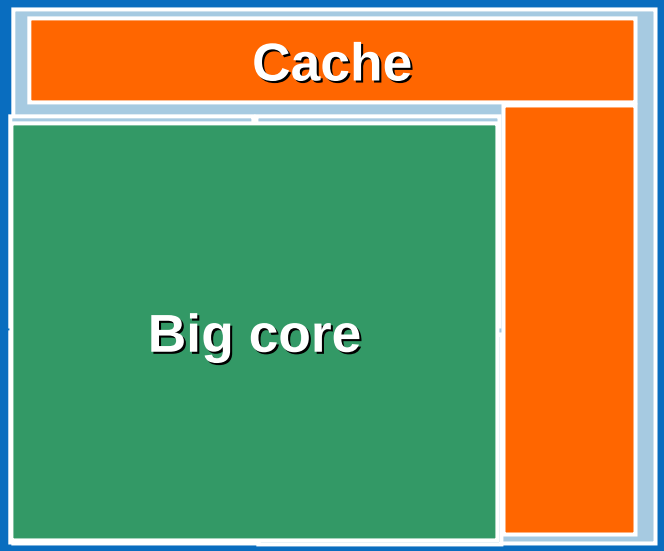
Area = 1
Voltage = 1
Freq = 1
Power = 1
Perf = 1

DUAL CORE



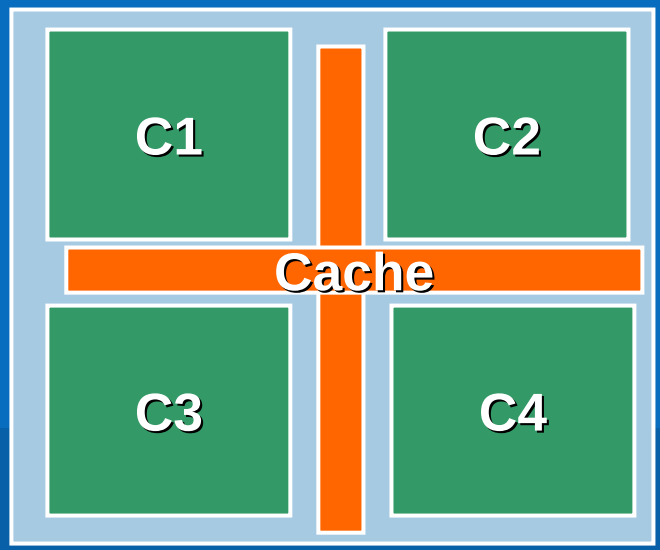
Area = 2
Voltage = 0.85
Freq = 0.85
Power = 1
Perf = ~1.8

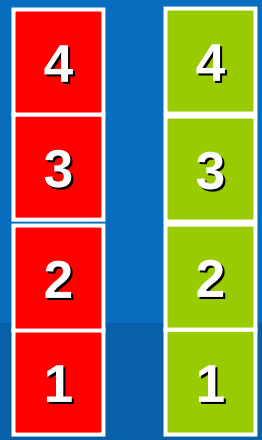
Multiple cores deliver more performance per watt



Power = ¼

Performance = 1/2





Many core is more power efficient

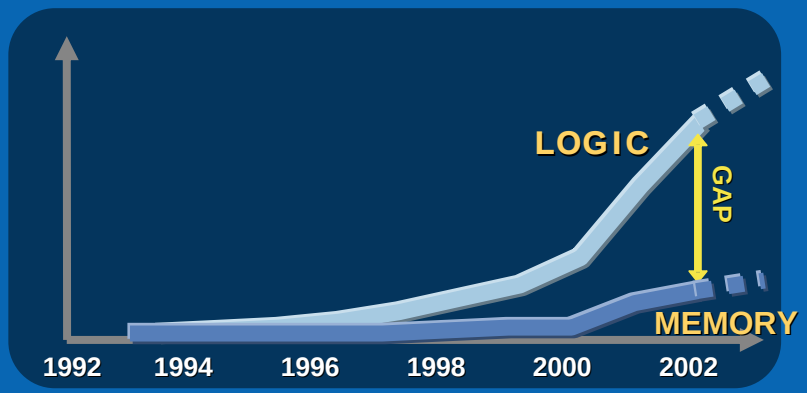
Power ~ area

Single thread performance ~ area.5**

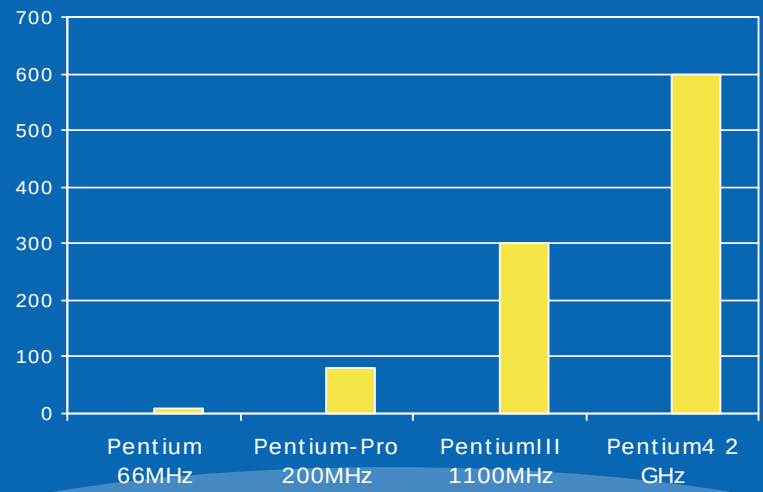
intel

Memory Gap

Growing Performance Gap



Peak Instructions Per DRAM Access



Reduce DRAM access with large caches

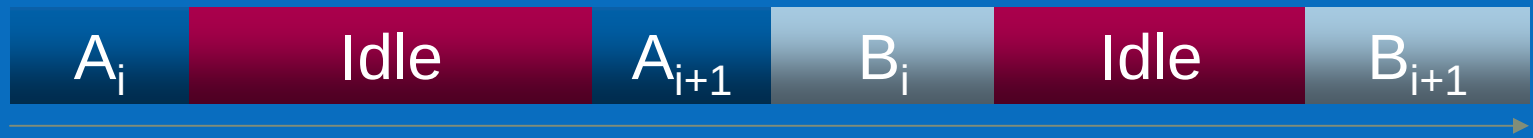
Extra benefit: power savings. Cache is lower power than logic

Tolerate memory latency with multiple threads

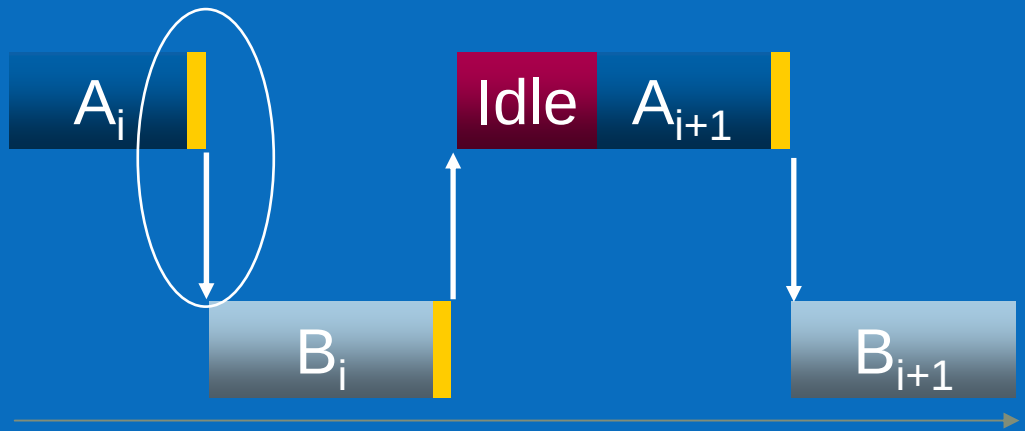
- Multiple cores
- Hyper-threading

Multi-threading tolerates memory latency

Serial Execution



Multi-threaded Execution

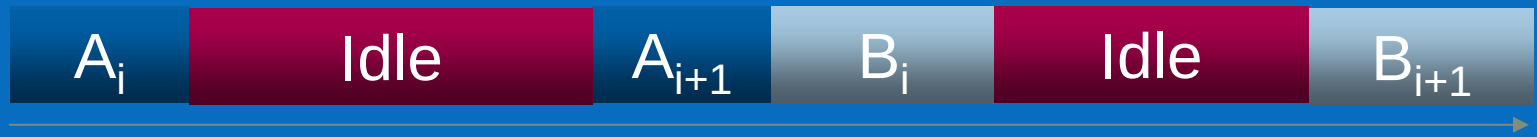


Execute thread B while thread A waits for memory

Multi-core has a similar effect

Multi-core tolerates memory latency

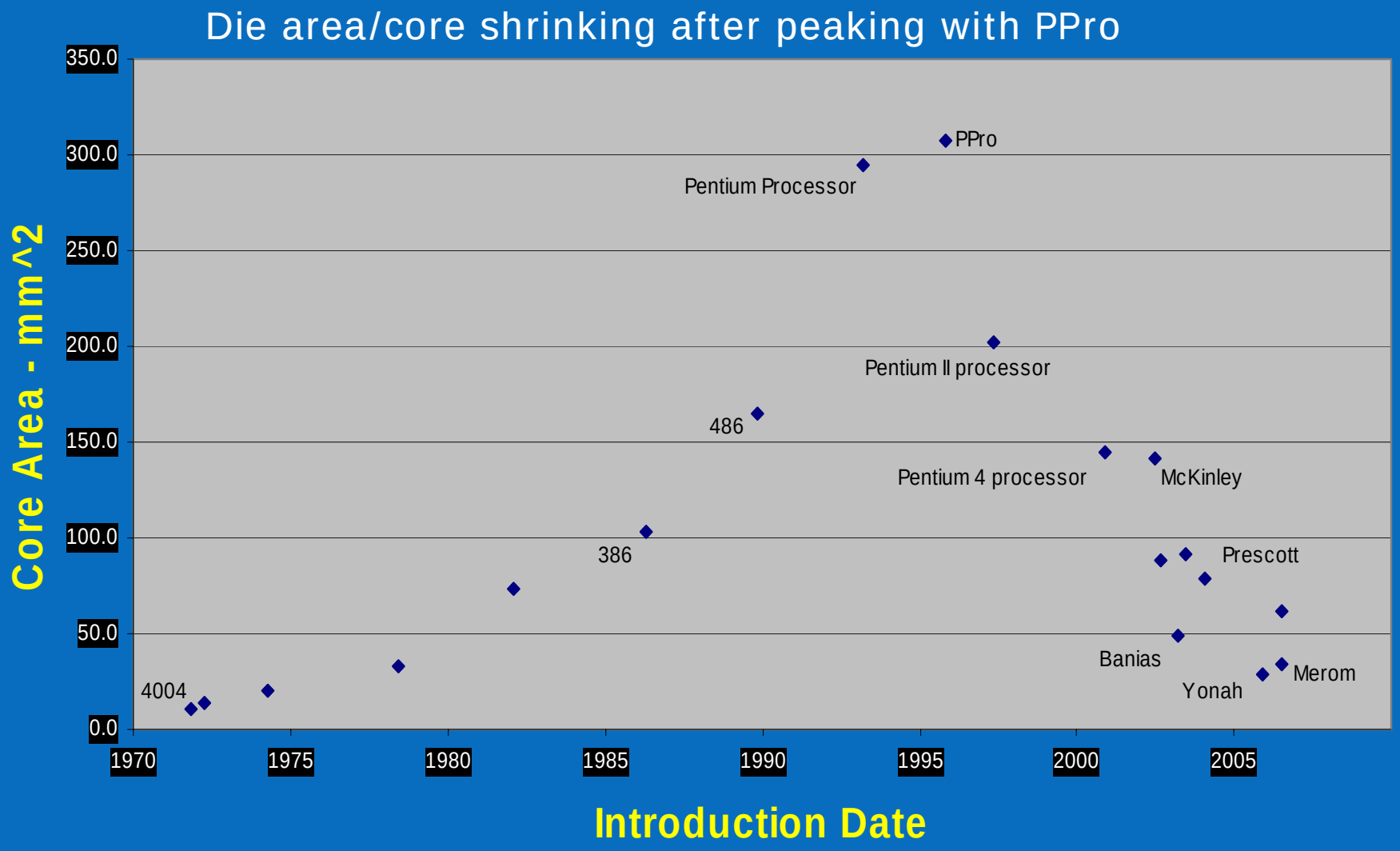
Serial Execution



Multi-core Execution

Execute thread A and B simultaneously

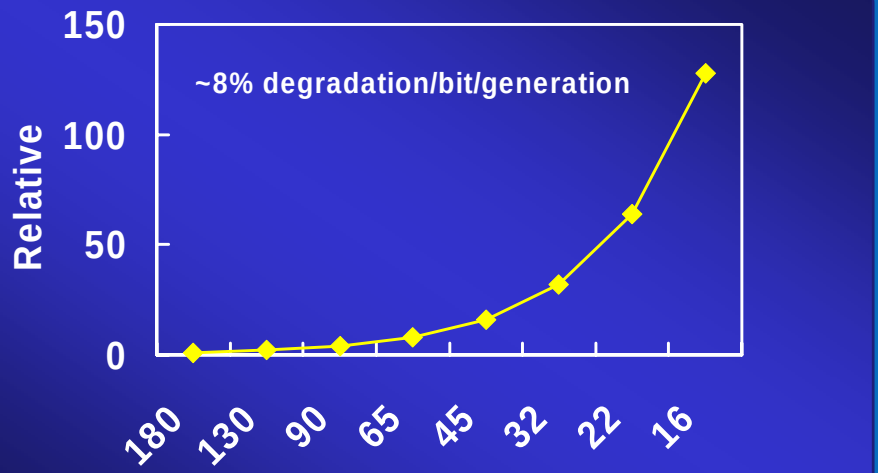
Core Area (with L1 caches) Trend



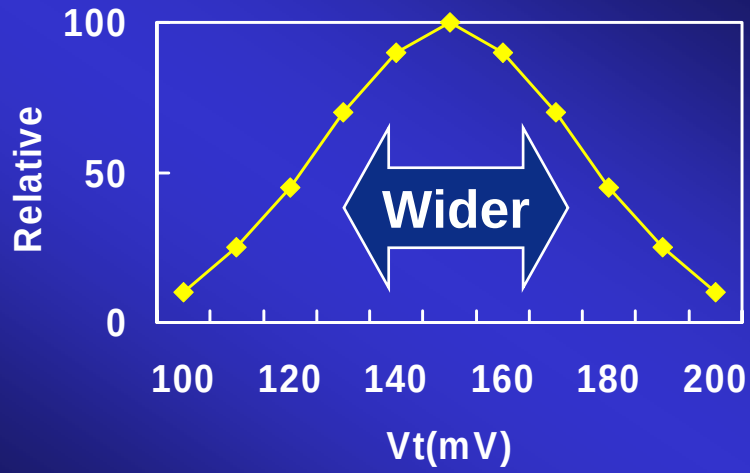
Why shrinking? Diminishing returns on performance.

Interconnect. Caches. Complexity.

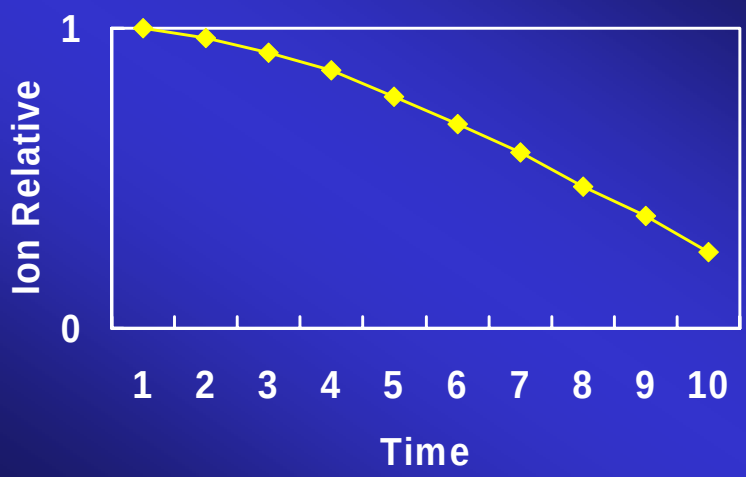
Reliability in the long term



Soft Error FIT/Chip (Logic & Mem)



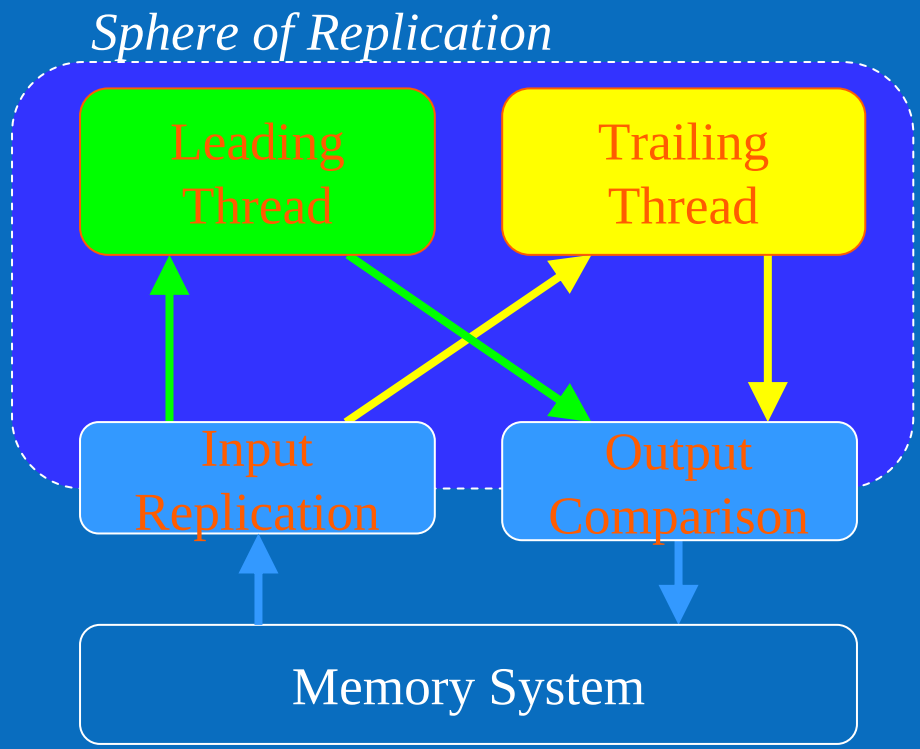
Extreme device variations



Time dependent device degradation

In future process generations, soft and hard errors will be more common.

Redundant multi-threading: an architecture for fault detection and recovery

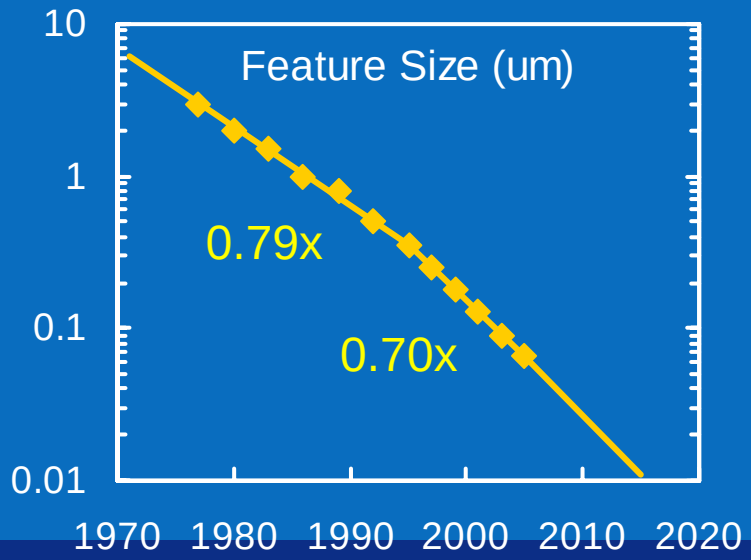


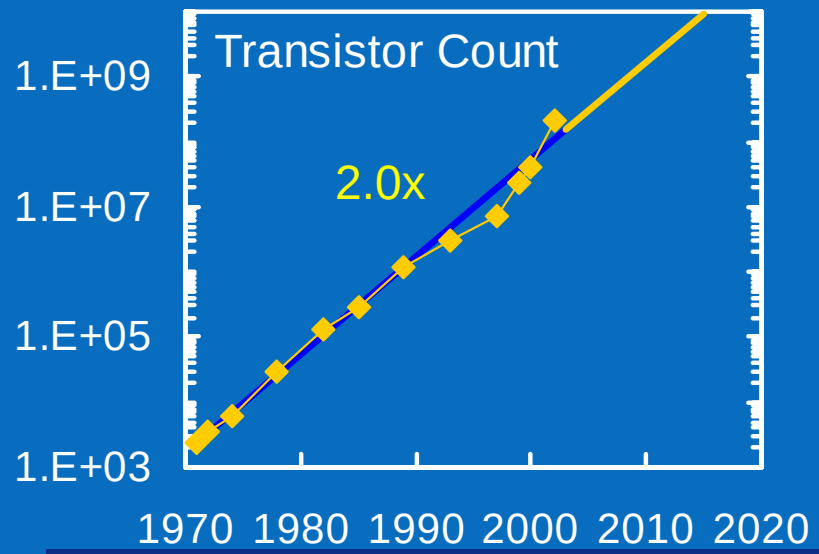
Two copies of each architecturally visible thread

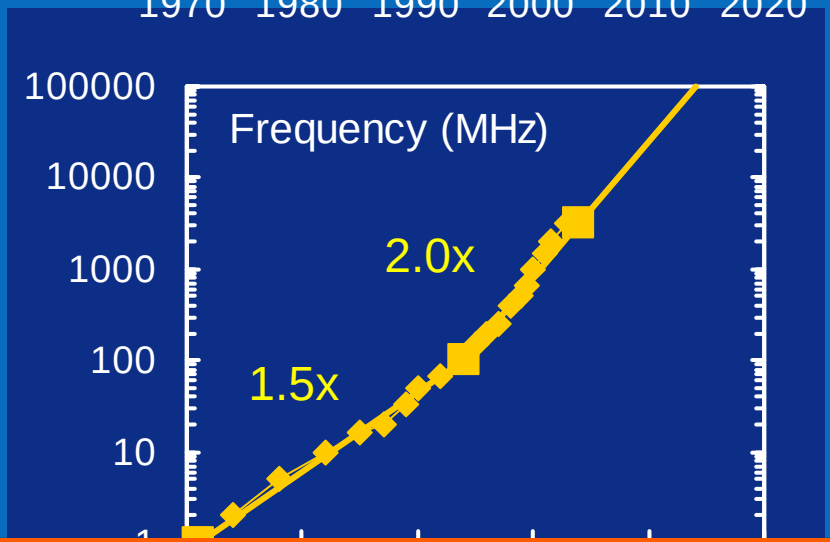
Compare results: signal fault if different

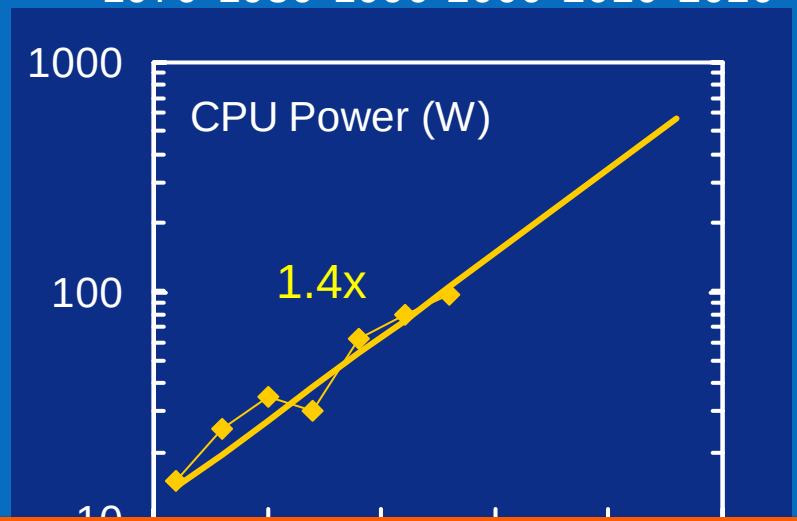
Multi-core enables many possible designs for redundant threading.

Moore's Law at Intel 1970-2005





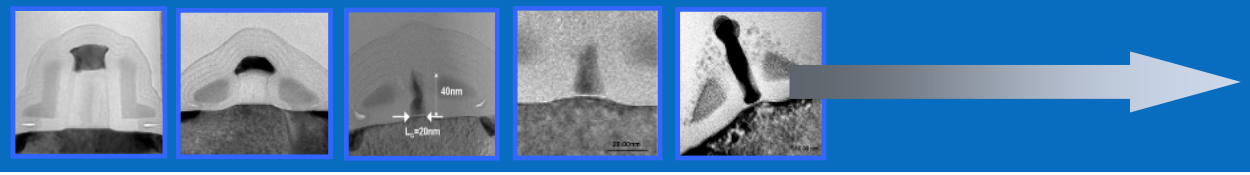




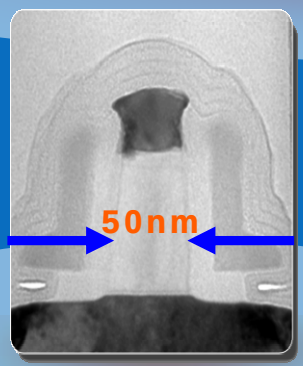
Multi-core addresses power, performance, memory, complexity, reliability

Moore's Law will provide transistors

Intel process technology capabilities

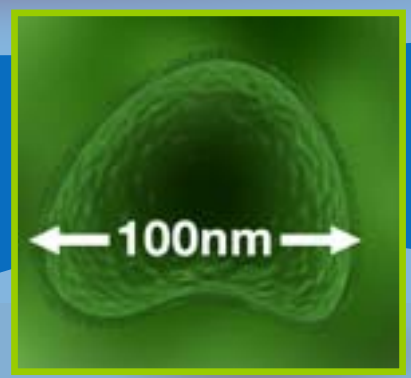


High Volume Manufacturing	2004	2006	2008	2010	2012	2014	2016	2018
Feature Size	90nm	65nm	45nm	32nm	22nm	16nm	11nm	8nm
Integration Capacity (Billions of Transistors)	2	4	8	16	32	64	128	256



Transistor for 90nm Process

Source: Intel



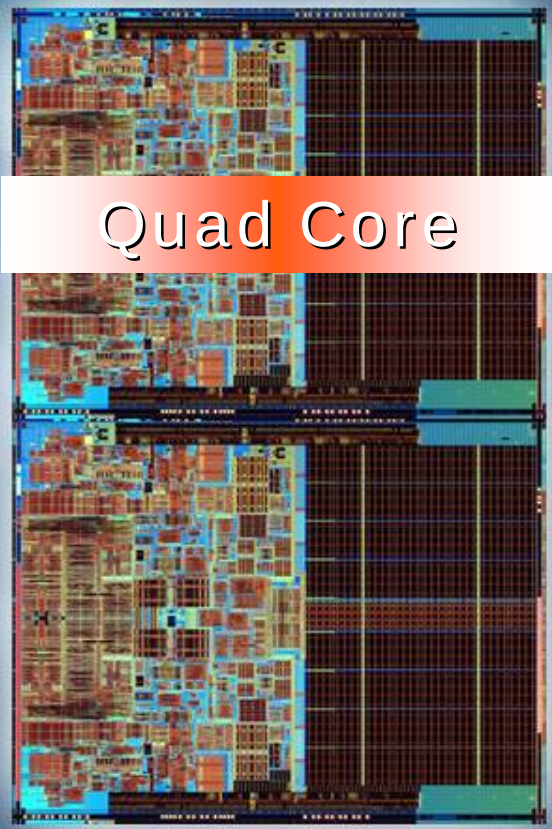
Influenza Virus

Source: CDC

Use transistors for multiple cores, caches, and new features.

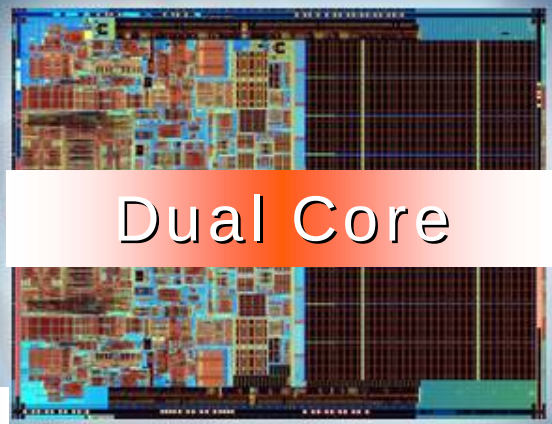
Multi-Core Processors

CLOVERTOWN



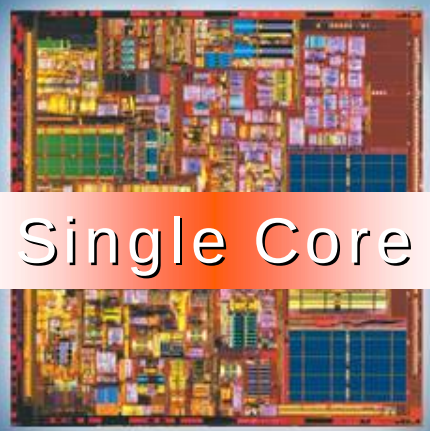
Quad Core

WOODCREST



Dual Core

PENTIUM® M



Single Core

Future Architecture: More Cores

Open issues

Cores

- How many?
- What size?
- Homogeneous?
- Heterogeneous?

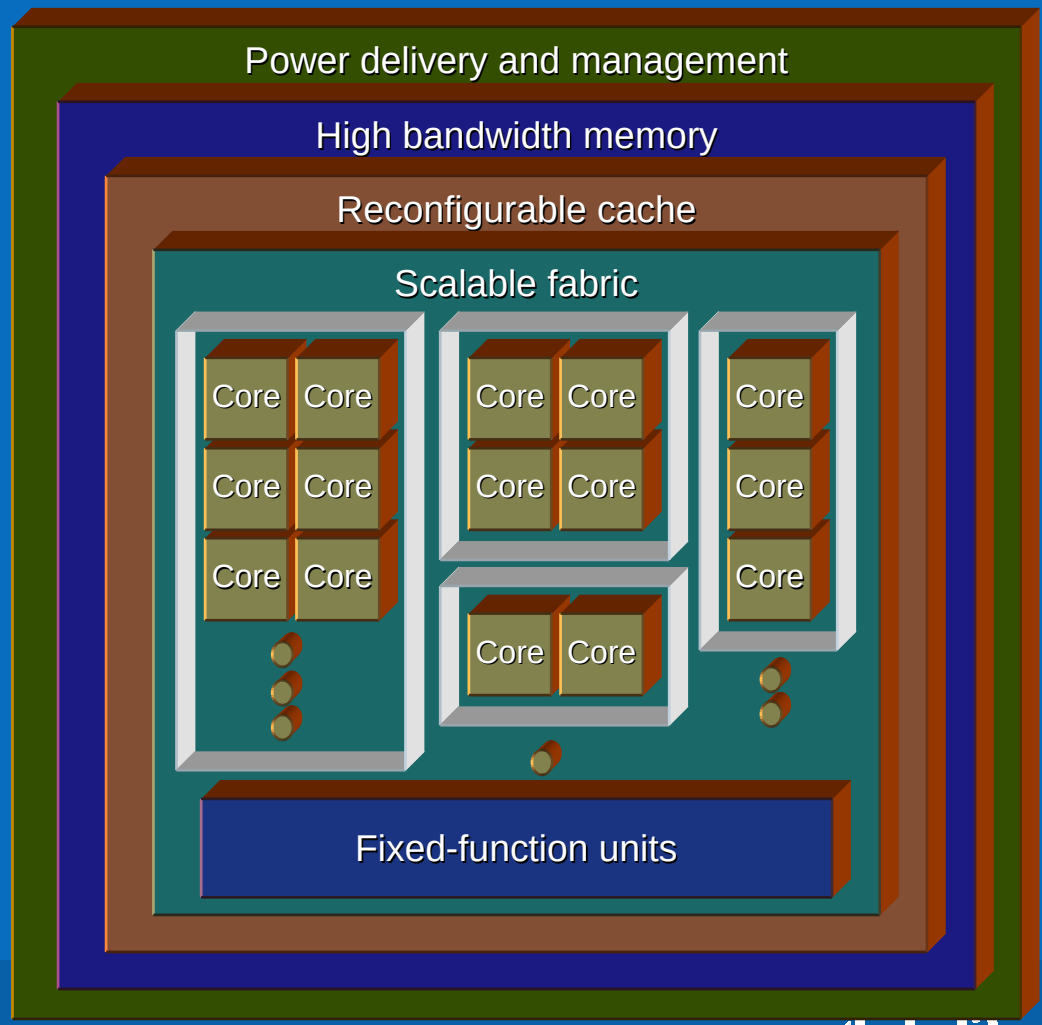
On-die Interconnect

- Topology
- Bandwidth

Cache hierarchy

- Number of levels
- Sharing
- Inclusion

Scalability



The Importance of Threading

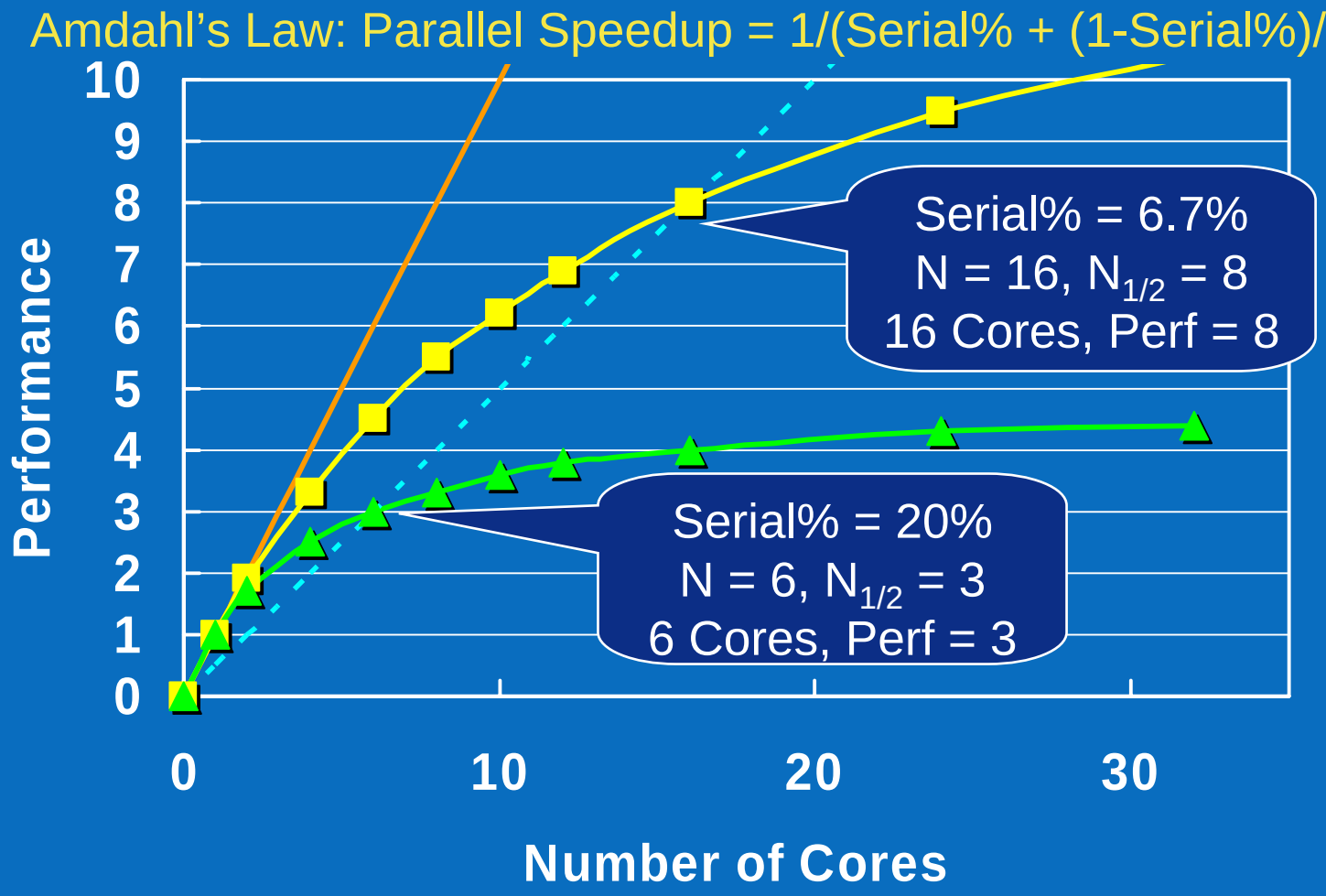
Do Nothing: Benefits Still Visible

- Operating systems ready for multi-processing
- Background tasks benefit from more compute resources
- Virtual machines

Parallelize: Unlock the Potential

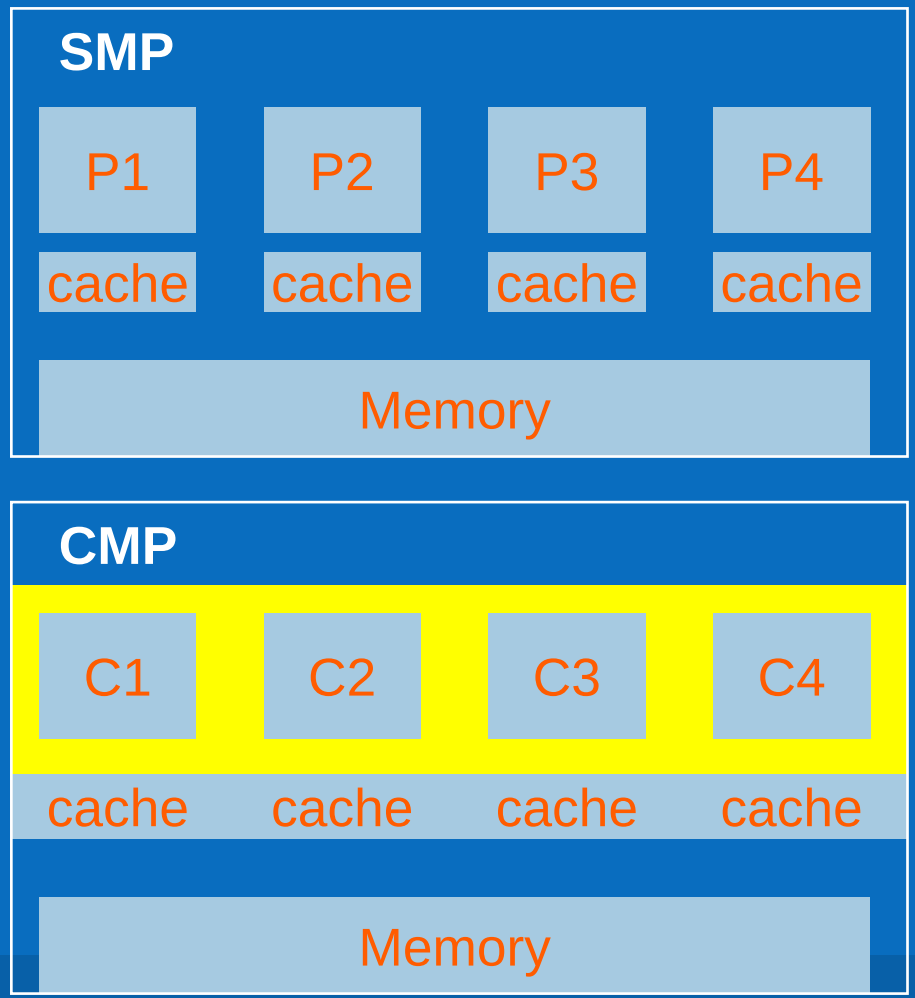
- Native threads
- Threaded libraries
- Compiler generated threads

Performance Scaling



Parallel software key to Multi-core success

How does Multicore Change Parallel Programming?



No change in fundamental programming model

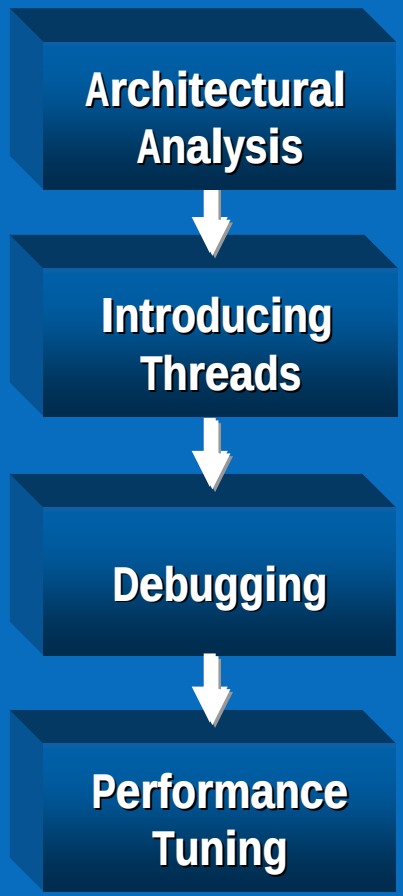
Synchronization and communication costs greatly reduced

- Makes it practical to parallelize more programs

Resources now shared

- Caches
- Memory interface
- Optimization choices may be different

Threading for Multi-Core



Intel® VTune™ Analyzers

Intel® C++ Compiler

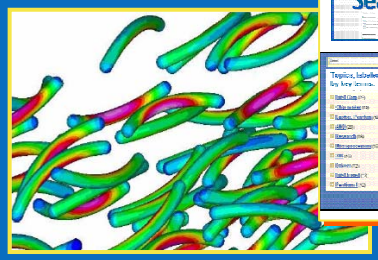
Intel® Thread Checker

Intel® Thread Profiler

Intel has a full set of tools for parallel programming

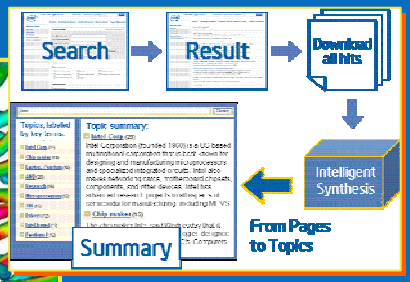
The Era Of Tera

Terabytes of data. Teraflops of performance.



Scientific Simulation

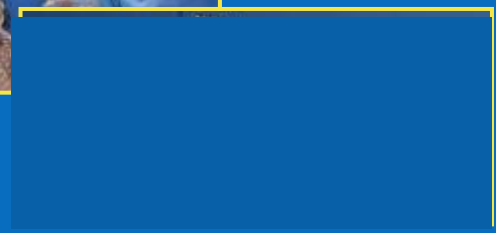
Courtesy Tsinghua University HPC Center



Text Mining

Improved Medical care

Machine vision



When personal computing finally becomes ***personal***

Immersive 3D entertainment

Source: Steven K. Feiner, Columbia University.

Interactive learning

Tele-present meetings

Virtual realities

Computationally intensive applications of the future will be highly parallel

21st Century Computer Architecture

Slide from
Patterson 2006

Old CW: Since cannot know future programs, find set of old programs to evaluate designs of computers for the future

- E.g., SPEC2006

What about parallel codes?

- Few available, tied to old models, languages, architectures, …

New approach: Design computers of future for numerical methods important in future

Claim: key methods for next decade are 7 dwarves (+ a few), so design for them!

- Representative codes may vary over time, but these numerical methods will be important for > 10 years

Patterson, UC Berkeley, also predicts importance of parallel applications.

Phillip Colella's "Seven dwarfs"

Slide from
Patterson 2006

High-end simulation in the physical sciences = 7 numerical methods:

Structured Grids (including locally structured grids, e.g. Adaptive Mesh Refinement)

Unstructured Grids

Fast Fourier Transform

Dense Linear Algebra

Sparse Linear Algebra

Particles

Monte Carlo

If add 4 for embedded, covers all 41 EEMBC benchmarks

8. Search/Sort
9. Filter
10. Combinational logic
11. Finite State Machine

Note: Data sizes (8 bit to 32 bit) and types (integer, character) differ, but algorithms the same

Slide from "Defining Software Requirements for Scientific Computing", Phillip Colella, 2004

Well-defined targets from algorithmic, software, and architecture standpoint

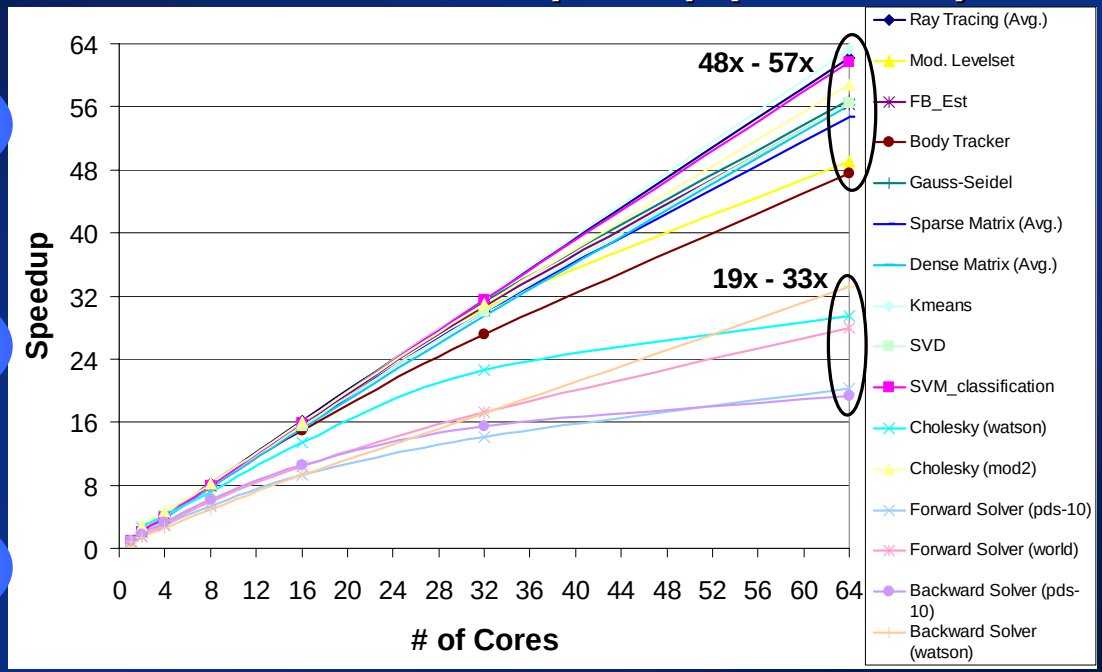
Note scientific computing, media processing, machine learning, statistical computing share many algorithms

Workload Acceleration

Recognition

Mining

Synthesis



Group I – Scale well with increasing core count

Examples: Ray tracing, body tracking, physical simulation

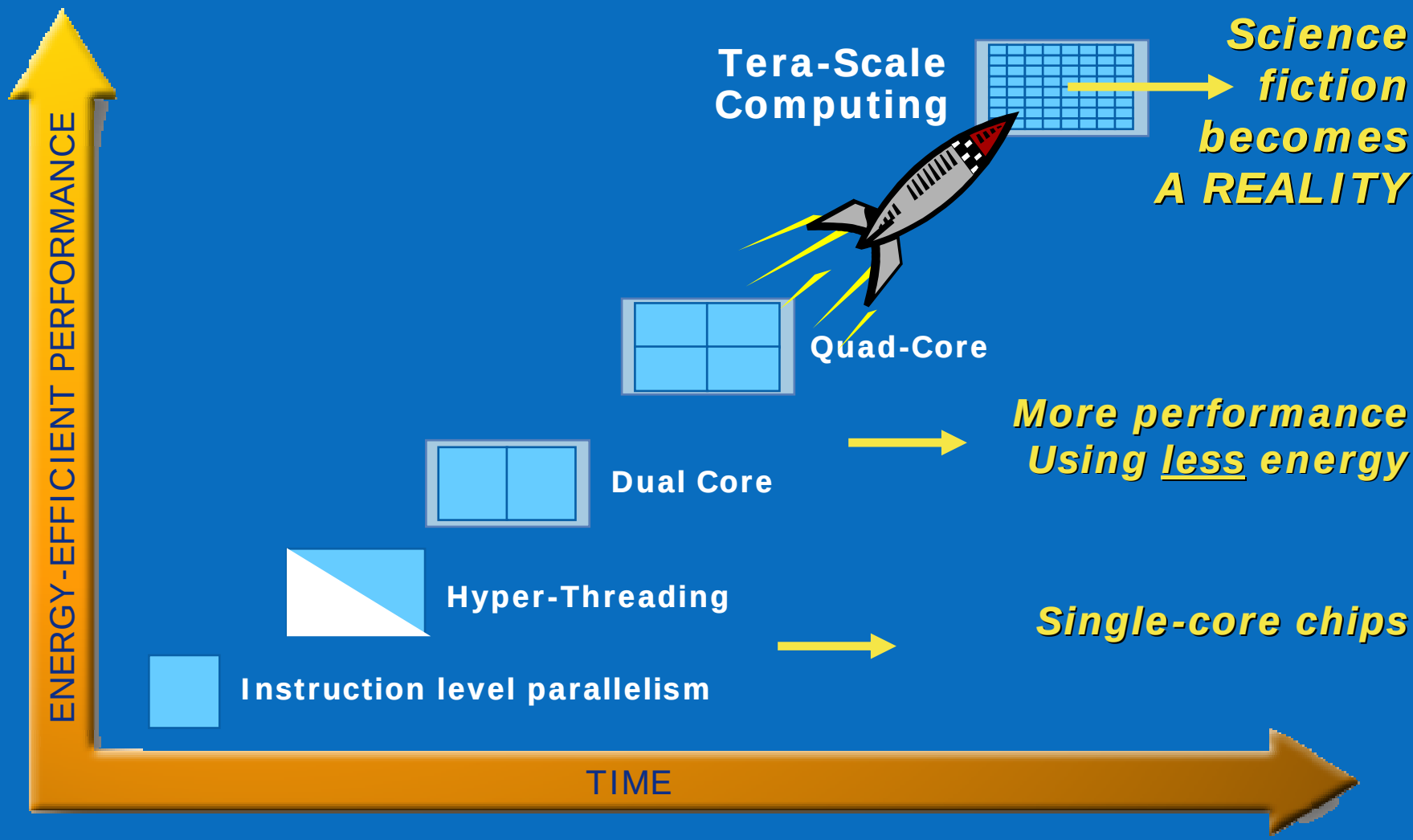
Group II – Worst-case scaling examples, yet still scale

Examples: Forward & backward solvers

Source: Intel Corporation

Tera-Leap to Parallelism:

Energy Efficient Performance



Summary

Technology is driving Intel to build multi-core processors.

- Power
- Performance
- Memory latency
- Complexity
- Reliability

Parallel programming is a central issue.

Parallel applications will become mainstream