University of Glasgow

School of Computing Science

Hardware Trends and their Implications

Advanced Operating Systems (M)

Lecture 2

Colin Perkins | https://csperkins.org/ | Copyright © 2018 | This work is licensed under the Creative Commons Attribution-NoDerivatives 4.0 International License. To view a copy of this license, visit http://creativecommons.org/licenses/by-nd/4.0/ or send a letter to Creative Commons, PO Box 1866, Mountain View, CA 94042, USA.

Lecture Outline

- Hardware trends and systems performance
  - Moore's law
  - Dennard Scaling
  - Power constraints
- System heterogeneity
  - CPU cores, memory, storage, and networking

CC BY ND

The PDP-11/40 and Unix

- The overwhelming majority of modern systems run some variant of Unix
  - macOS, iOS, Android, Linux
- Unix was designed in the early 1970s to run on PDP-11/40 minicomputers:
  - “The PDP-11/40 was designed to fit a broad range of applications, from small stand alone situations where the computer consists of only 8K of memory and a processor, to large multi-user, multi-task applications requiring up to 124K of addressable memory space. Among its major features are a fast central processor with a choice of floating point and sophisticated memory management, both of which are hardware options.”
    https://pdos.csail.mit.edu/6.828/2005/readings/pdp11-40.pdf
- Unix has proven surprisingly resilient and portable – is it still the right model?



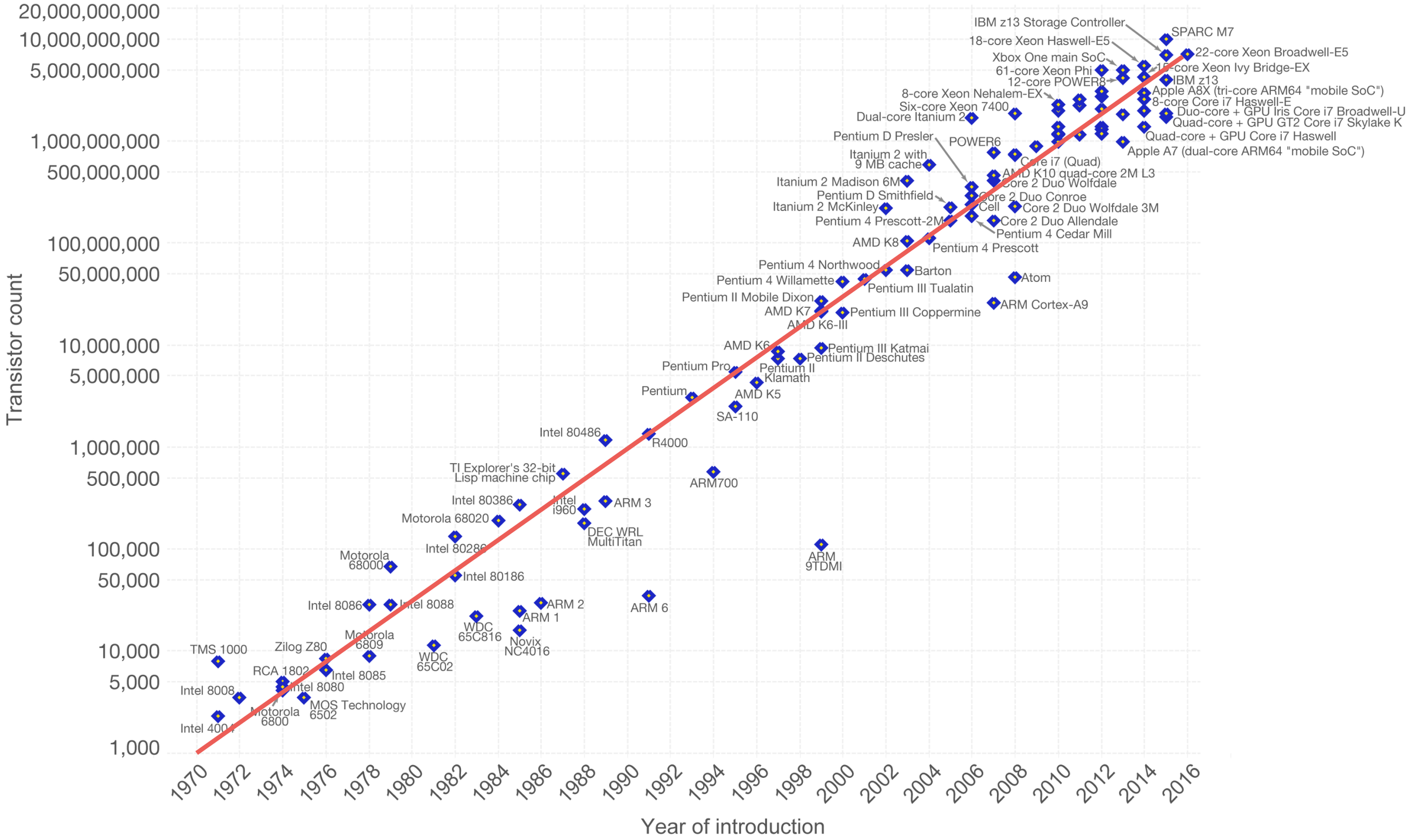
Source: Wikipedia

CC BY ND

Moore’s Law – The number of transistors on integrated circuit chips (1971-2016)

Our World in Data

Moore's law describes the empirical regularity that the number of transistors on integrated circuits doubles approximately every two years. This advancement is important as other aspects of technological progress – such as processing speed or the price of electronic products – are strongly linked to Moore's law.



Data source: Wikipedia (https://en.wikipedia.org/wiki/Transistor_count)
The data visualization is available at OurWorldinData.org. There you find more visualizations and research on this topic.

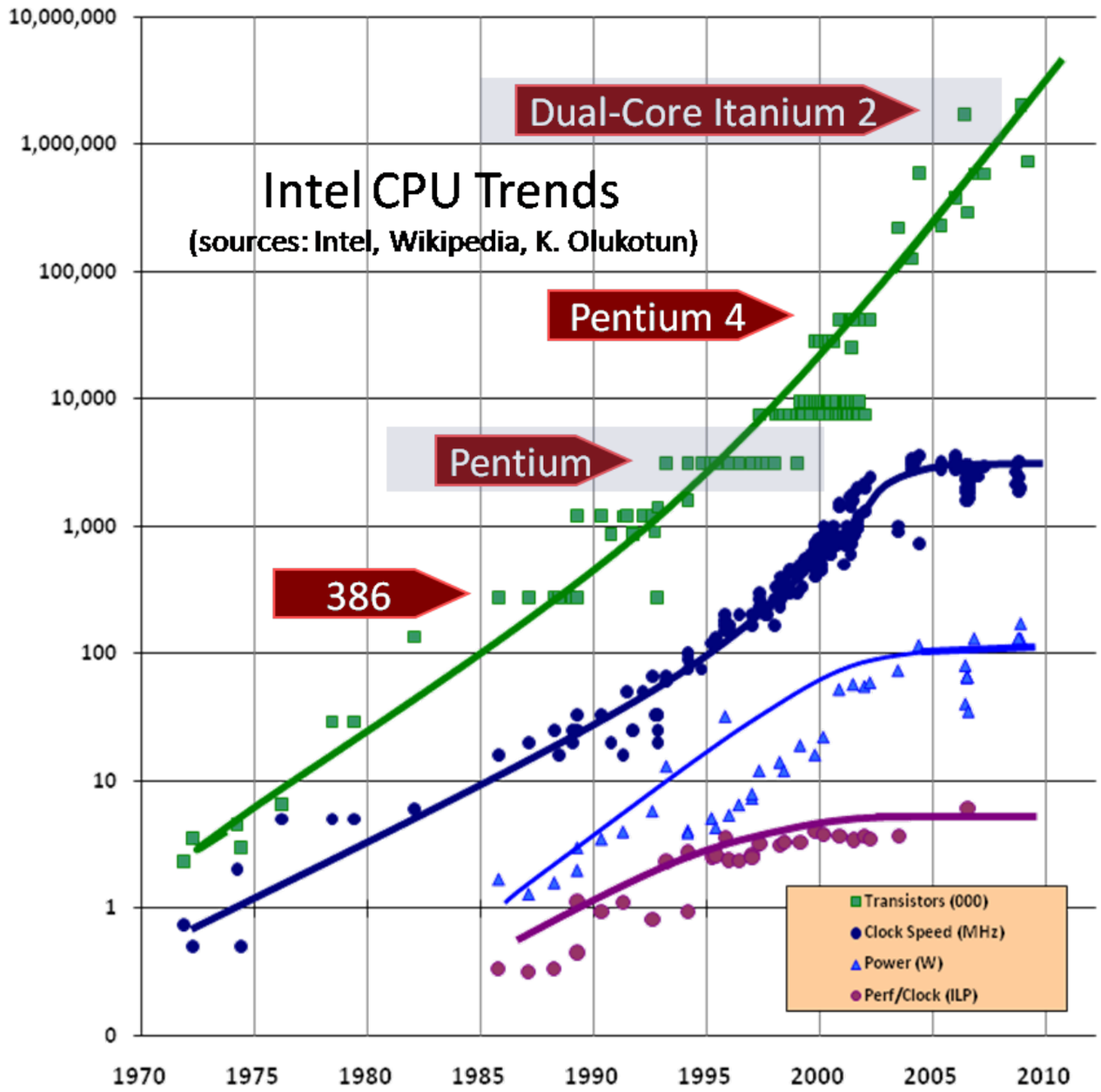
Licensed under CC-BY-SA by the author Max Roser.

Colin Perkins | https://csperkins.org/ | Copyright © 2018

Moore’s Law and Hardware Trends

- Moore’s law has held up well
- Other features have not:
  - Clock speed
  - Performance per clock cycle
  - Power consumption

- Starting to reach physical limits



H. Sutter, “The Free Lunch Is Over: A Fundamental Turn Toward Concurrency in Software”, Dr. Dobb's Journal, 30(3), March 2005 (updated with 2009 data) http://www.gotw.ca/publications/concurrency-ddj.htm

More recent data in A. Danowitz et al, “CPU DB: Recording Microprocessor History”, Communications of the ACM, 55(4), April 2012, Pages 55-63. DOI: 10.1145/2133806.2133822 – shows the trends continue into 2012

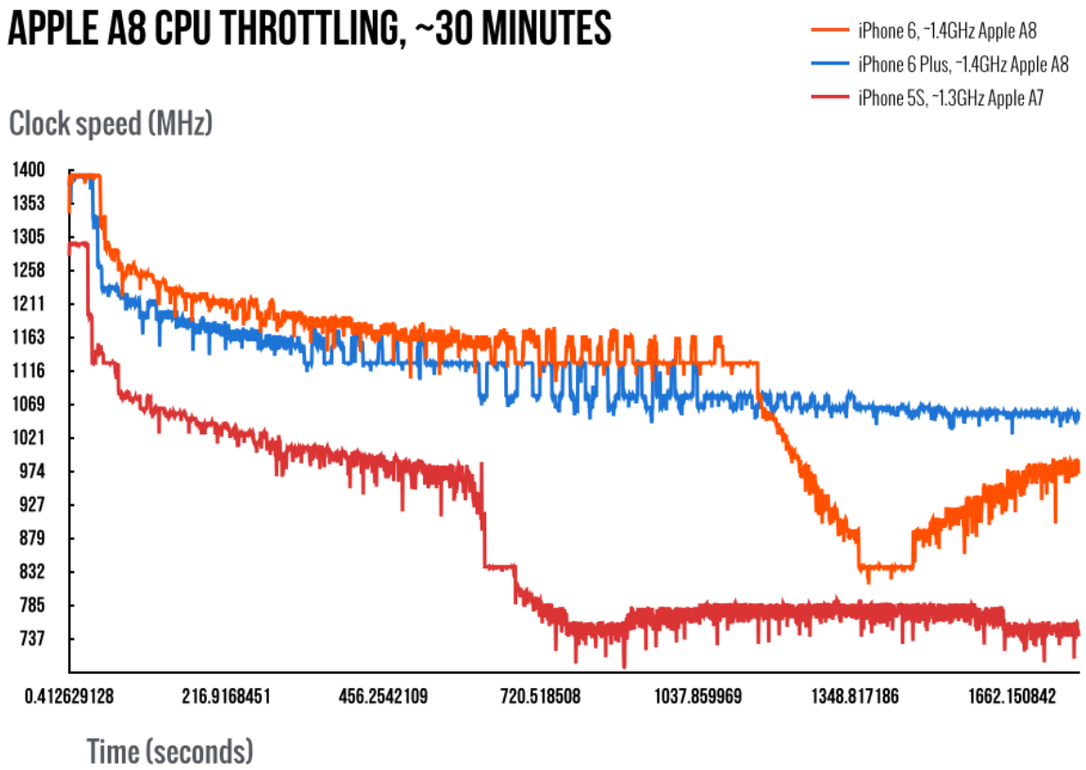
CC BY ND

Dennard Scaling

- Moore's law: transistor count, not performance
- Power consumption driven by Dennard scaling:
  - Power $\propto C \cdot F \cdot V^2$ where:
    C is capacitance
    F is clock frequency
    V is voltage
  - Size of transistor directly affects V and C
  - Implies smaller transistors (Moore's law) allows higher frequency for same power consumption
- Approximation that ignores *leakage current*
  - Limitations of semiconductor physics that dominate as sizes decreases – eventually became dominant source of power consumption
  - Dennard scaling equation no longer applies

CC BY ND

Thermal Throttling



Source: arstechnica.com

- CPU thermal throttling – cannot sustain performance
  - Graph is for Apple A8 SoC
  - Many other processors behave similarly
- Dark silicon – one consequence of Moore's law and breakdown of Dennard scaling; can't power the entire chip at once

CC BY ND

Implications of Hardware Trends

- Moore's law → how much longer?
- Breakdown in Dennard scaling:
  - Increased thermal throttling
  - Multicore systems
  - Power constraints – mobile *and* data centre

- Mobile devices have performance in bursts
- Data centres scale wider rather than faster

CC BY ND

Heterogeneity

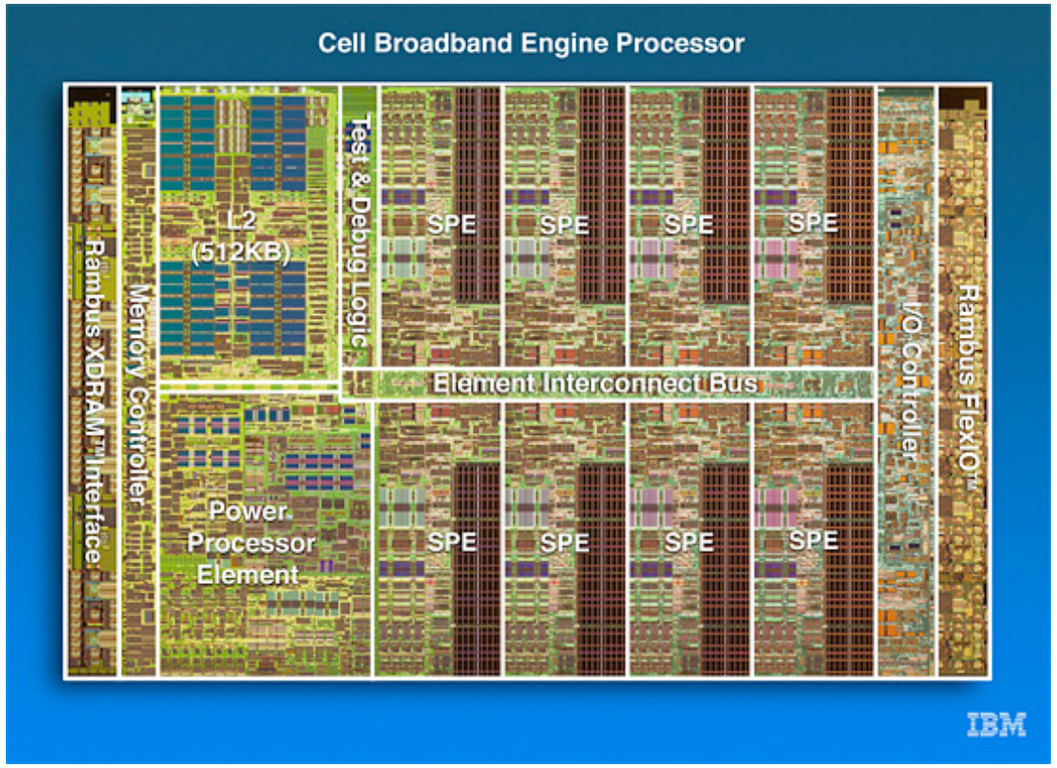
- Range of system designs increasing
  - Data centres
  - PCs and laptops
  - Mobile phones and tablets
  - Wearables
  - IoT

- Scalability and heterogeneity
  - Due to increasing range of different applications
  - To make effective use of Moore's law, while subject to power constraints → optimise hardware for particular use cases, power only when needed

BY ND

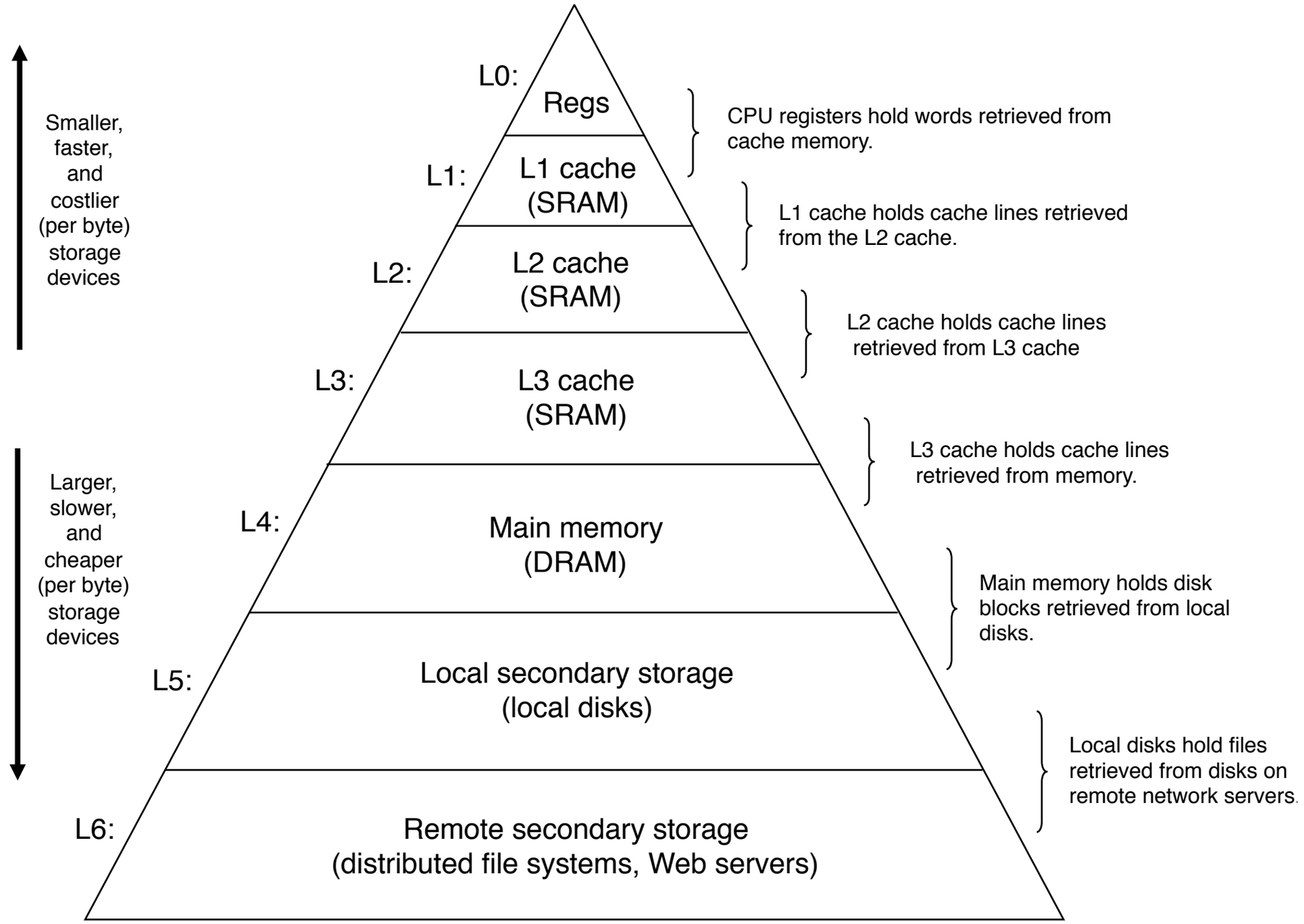
Heterogeneity: Processor Architecture

- Heterogeneous multiprocessor: CPU with multiple special purpose cores
  - Canonical example → Cell Broadband Engine
- Asymmetric processing capabilities
  - High-performance and low-power cores on a single die (e.g., ARM big.LITTLE model, with both Cortex A7 and A15 cores on-die)
  - GPU-like cores for graphics operations, with single program multiple data model vs. a more traditional multiple program multiple data model
  - Offload for crypto algorithms, TCP stack, etc.
- Asymmetric memory access models
  - Non-cache coherent
  - Cores explicitly do not share memory
- Common for mobile phones, games consoles, and other non-PC hardware



Colin Perkins | https://csperkins.org/ | Copyright © 2018

Heterogeneity: Memory Hierarchy

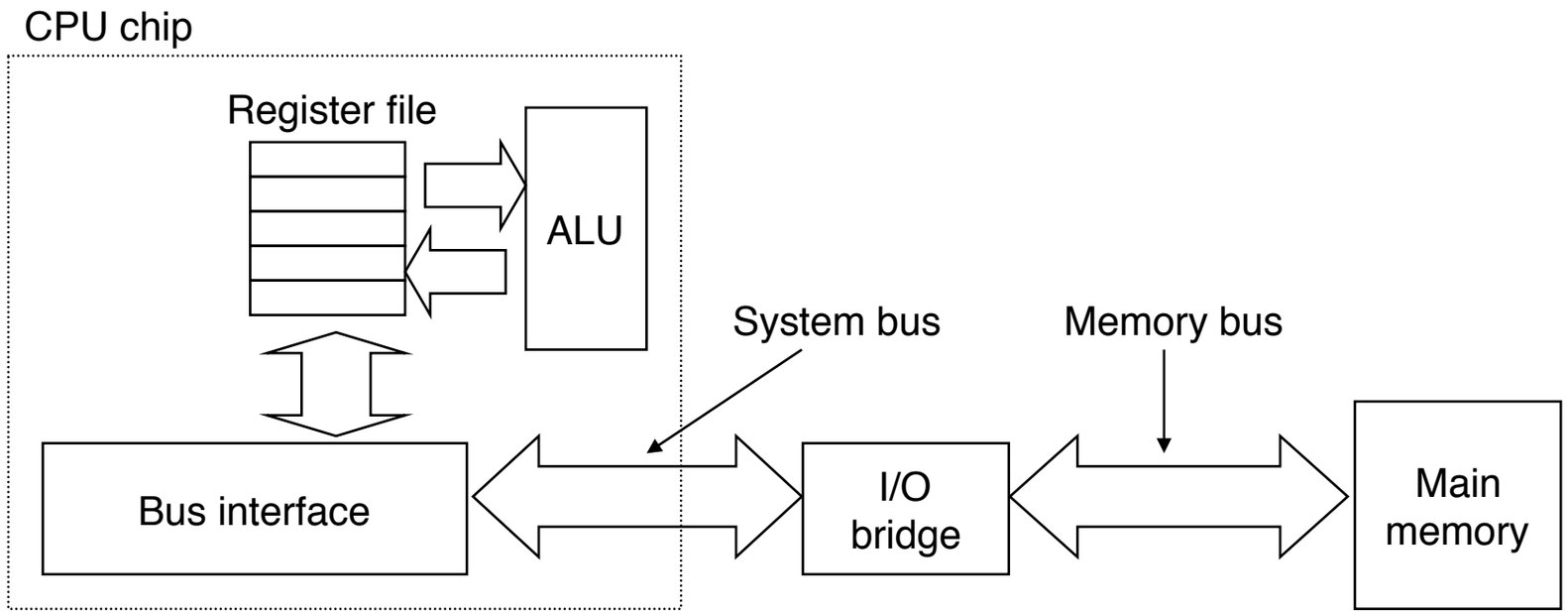


Source: Bryant & O'Hallaron, "Computer Systems: A Programmer's Perspective", 3rd Edition, Pearson, 2016
Fig. 6.21. http://csapp.cs.cmu.edu/3e/figures.html (website grants permission for lecture use with attribution)

BY ND

Heterogeneity: Memory Hierarchy

- The traditional view of memory in a computer system:



Source: Bryant & O'Hallaron, "Computer Systems: A Programmer's Perspective", 3rd Edition, Pearson, 2016
Fig. 6.6. http://csapp.cs.cmu.edu/3e/figures.html (website grants permission for lecture use with attribution)

- Note:
  - One CPU
  - One memory bus
  - One block of memory
- Uniform memory access

CC BY ND

Heterogeneity: Memory Hierarchy

- Organisation of a typical modern computer system:

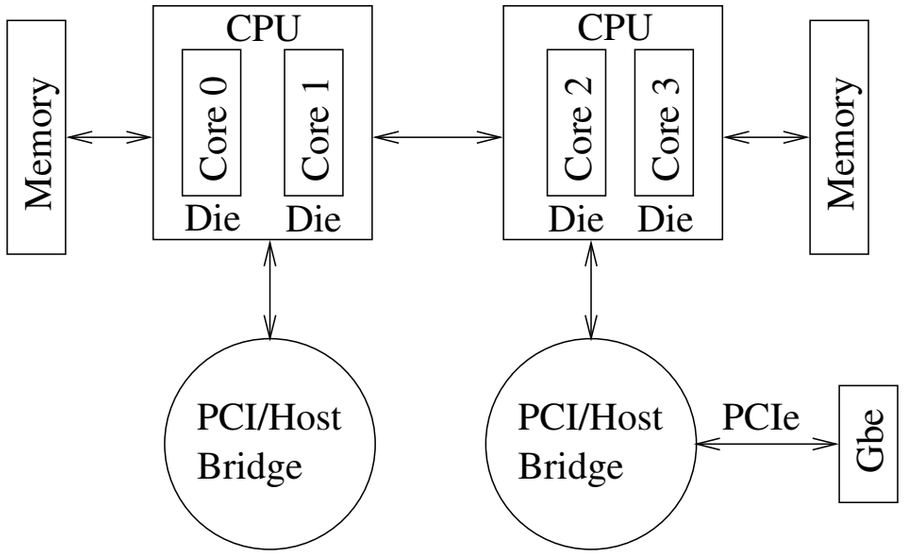


Figure 2. Structure of the AMD system

A. Schüpbach, *et al.*, Embracing diversity in the Barrelfish manycore operating system. Proc. Workshop on Managed Many-Core Systems, Boston, MA, USA, June 2008. ACM.

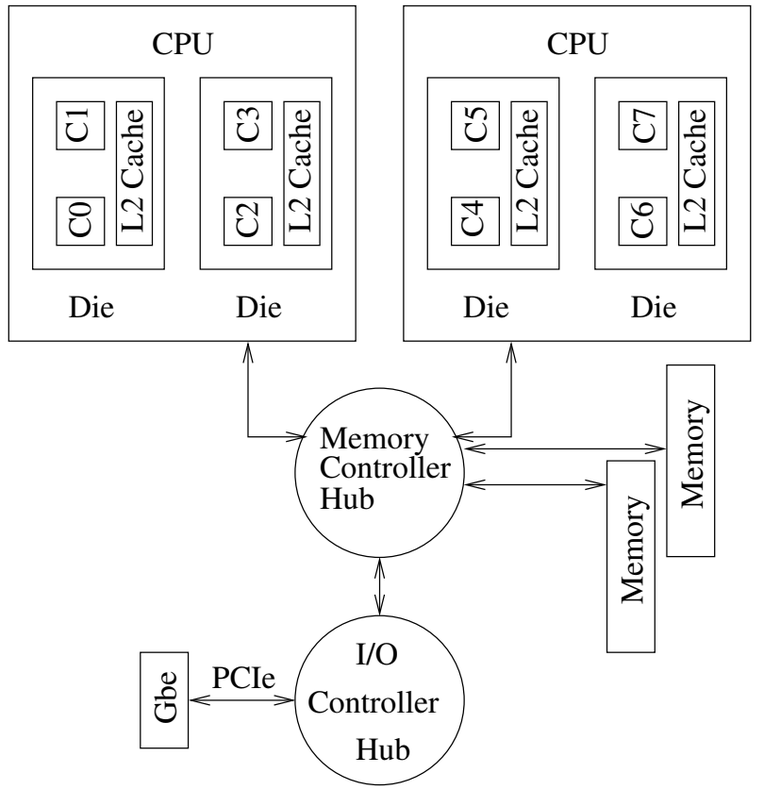


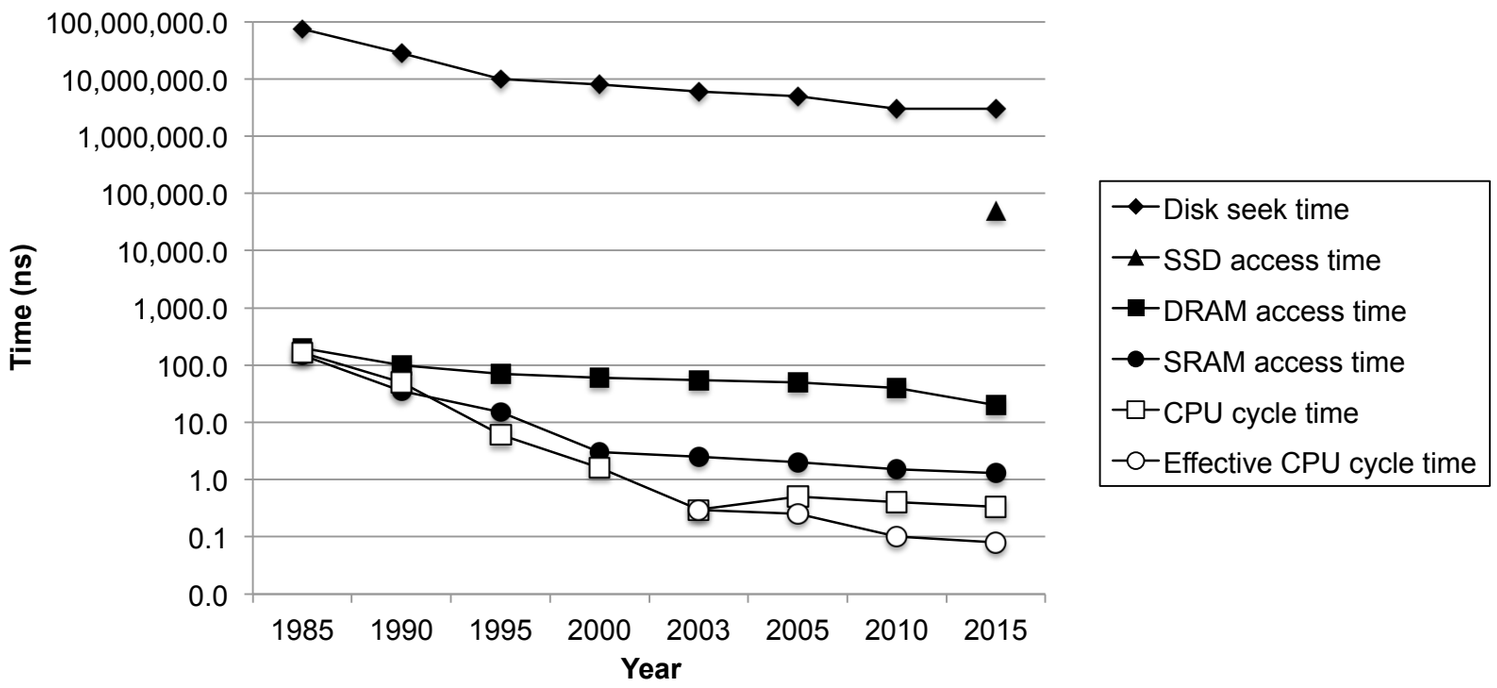
Figure 1. Structure of the Intel system

- Non-uniform memory access (NUMA)
  - Large on-chip cache memory; main memory off-chip, accessed via interconnect
  - Heterogeneity unavoidable, due to the physical layout of the hardware – some memory is physically closer to some CPU chips than to others
  - Cache coherency protocols maintain random access illusion
  - Memory access latency varies depending on which core is accessing which memory bank

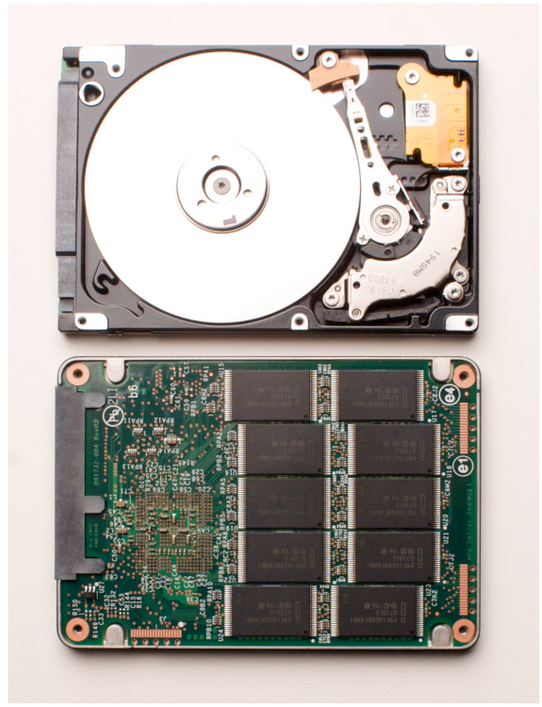
CC BY ND

Heterogeneity: Storage

- Hard disks → solid state disks
  - No moving parts
  - True random access – hard disk seek time varies with physical movement of disk heads
  - Lower latency, higher throughput
  - Lower power consumption
  - Wear levelling and block-level FEC – limited number of write cycles
  - Sophisticated controller hidden in the SSD – embedded computer, not device driver



Source: Bryant & O'Hallaron, "Computer Systems: A Programmer's Perspective", 3rd Edition, Pearson, 2016
Fig. 6.16. http://csapp.cs.cmu.edu/3e/figures.html (website grants permission for lecture use with attribution)



Source: Intel

CC BY ND

Heterogeneity: Networking

- Network performance has caught up with CPU performance
  - DMA-based ring-buffer interface to NIC
  - Zero copy from packet reception to user space
  - Protocol header processing and kernel data structures are a significant cost – difficult to keep up with line rate processing of small packets on high speed links
  - Pushing towards different device driver and network APIs – sockets API a bottleneck

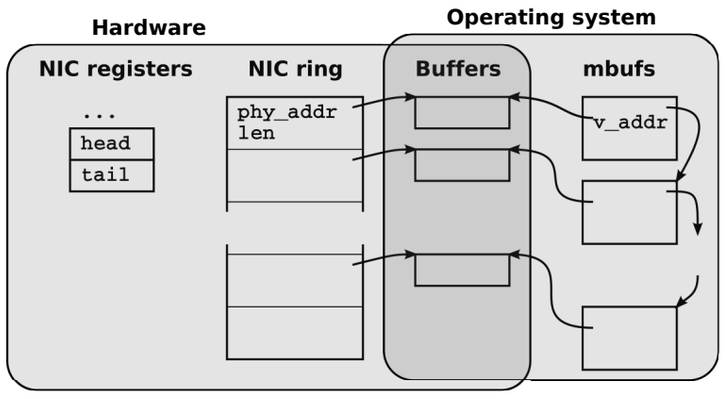


Figure 1: Typical NIC's data structures and their relation with the OS data structures.

From: L. Rizzo, "netmap: a novel framework for fast packet I/O", Proc. USENIX Annual Technical Conference, Boston, MA, USA, June 2012.

- Increasing range of interface types: wired and wireless
- Wireless and mobility expose the limitations of TCP congestion control and service model
- Multiple interfaces, wireless, and mobility push towards multipath protocols, protocols for path discovery and edge-to-network communication, and new APIs

CC BY ND

Summary

- Systems are increasingly heterogenous, and quite different to those used when the OS was designed

- How should change the programming model?
- Are we using appropriate programming languages, tools, and operating systems architectures?

CC BY ND

Implications of Hardware Changes for Systems Programming

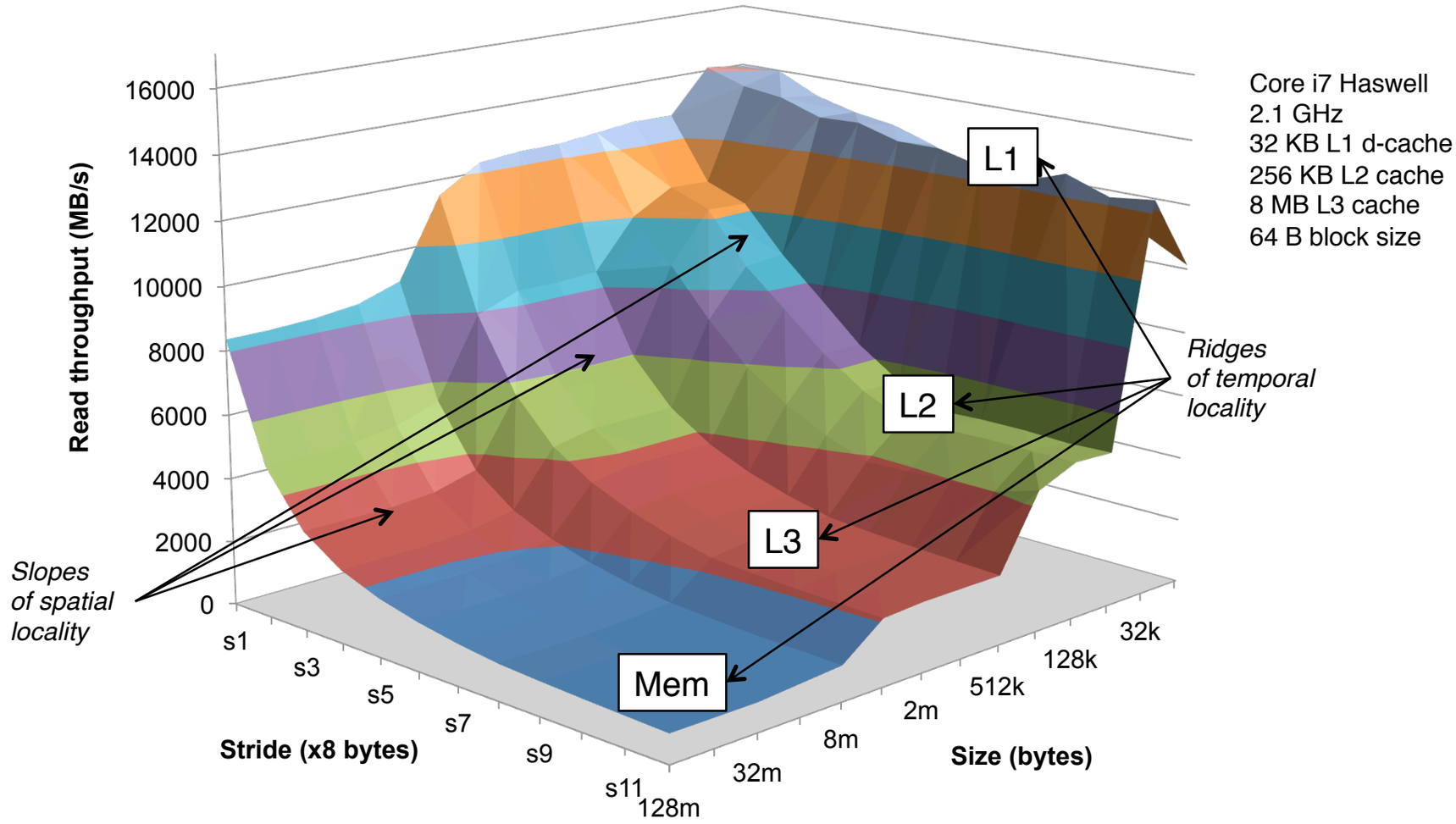
Implications for Systems Programming

- What are implications of hardware changes for systems programming?
  - Memory hierarchy
  - Non-uniform memory access
  - Concurrency
  - Solid state storage
  - High-performance networking

CC BY ND

Implications of Memory Hierarchy

- Memory isn't flat → access time depends on data location



Smaller values of **stride** indicate data with better spatial locality; **size** is the total amount of data accessed

Source: Bryant & O'Hallaron, "Computer Systems: A Programmer's Perspective", 3rd Edition, Pearson, 2016
Fig. 6.41. http://csapp.cs.cmu.edu/3e/figures.html (website grants permission for lecture use with attribution)

CC BY ND

Implications of Memory Hierarchy

- Data should be packed so objects accessed close in time are located close in memory
  - RAM = Random Access Memory – *but not every access takes equal time*
- Importance of caching and data layout
  - Bringing an object into CPU cache also brings in adjacent objects
    - "cache line size" → 64 bytes, typically
  - Reading from main memory >1000x slower than reading from L1 cache, (generally – highly system dependent)
- Implications of pointer chasing on performance
- Choice of programming language matters
  - A systems language must allow precise control over data layout, memory allocation, and use of pointers – C, C++, Rust, but not Java, C#, etc.

CC BY ND

Implications of NUMA

- Traditional hardware gives appearance of a uniform flat memory, shared between cores
  - Complex cache coherency protocols and memory models
  - Varying degrees of success in hiding the diversity
- Increasingly an illusion, maintained by underlying inter-core network
  - e.g., AMD HyperTransport, Intel QuickPath
  - Point-to-point communications/switching network with message passing protocol

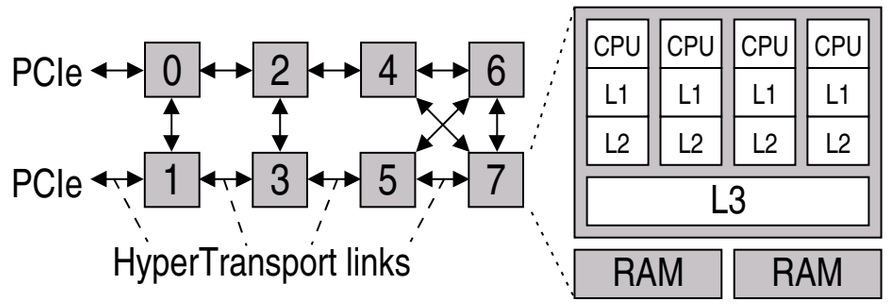


Figure 2: Node layout of an 8×4-core AMD system

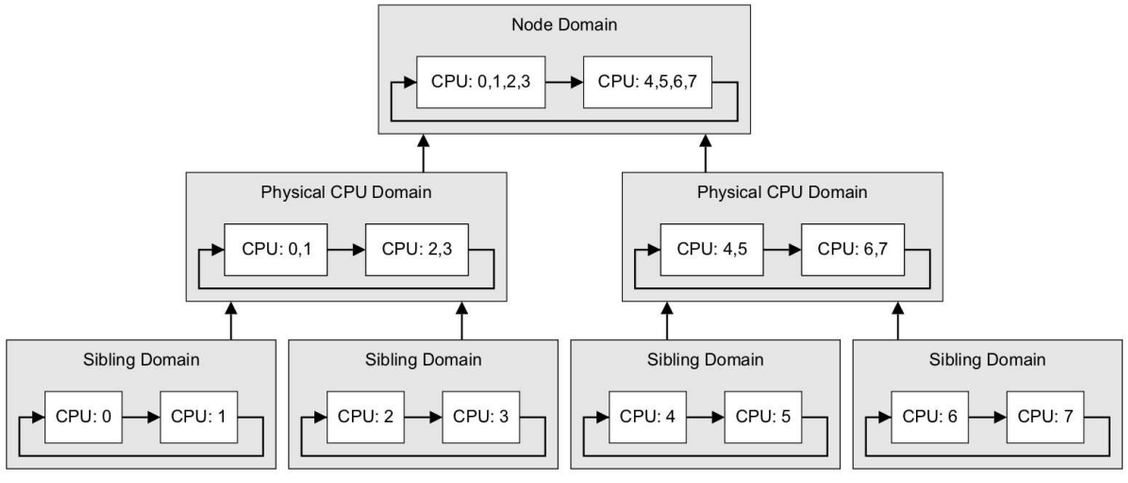
Baumann *et al*, "The Multikernel: A new OS architecture for scalable multicore systems", Proc. ACM SOSP 2009. DOI 10.1145/1629575.1629579

- Processors increasingly exposing heterogeneity – how should it be visible to software?

CC BY ND

Implications of NUMA: Scheduling

- Scheduler must be aware of CPU topology and memory hierarchy to assign threads to physically close processors
  - Avoid blocking on long-distance memory access delays
- Three layers of topology in Linux
  - Hyper-threading within a core (siblings)
  - Cores within a physical CPU package
  - Physical CPUs within a NUMA node
- Load balancing between nodes essential for performance
  - CPU intensive tasks should be separate
  - Communicating threads should be close
  - Monitor and periodically rebalance – heuristic driven, hard to formulate a general policy
  - Linux has `sched_setaffinity()` to bind thread to a particular set of CPUs, to allow manual optimisation



M. Bligh *et al.*, Linux on NUMA systems. Proc. Ottawa Linux Symposium, July 2004

Colin Perkins | https://csperkins.org/ | Copyright © 2018

Implications of NUMA: Memory Allocation

- Locality-aware memory allocation
  - Memory is discontiguous between nodes – partitioned address space
  - Threads should allocate memory local to the node on which they execute; essentially an independent memory management subsystem per node
    - The `malloc()` API is not sufficient in itself – cannot ensure that related data accessed by multiple threads is allocated in memory that is located on the same node, and cannot place allocations on particular nodes (pinning threads to particular nodes can help here)
- Replication of kernel memory on multiple cores
  - Read-only memory regions accessed by multiple threads should be replicated across nodes
    - e.g., the kernel code, shared libraries such as libc
  - Requires support from VM system, to map a virtual address to a different physical address on each core
  - Un-copy-on-write to collapse replicas down to single page if write occurs

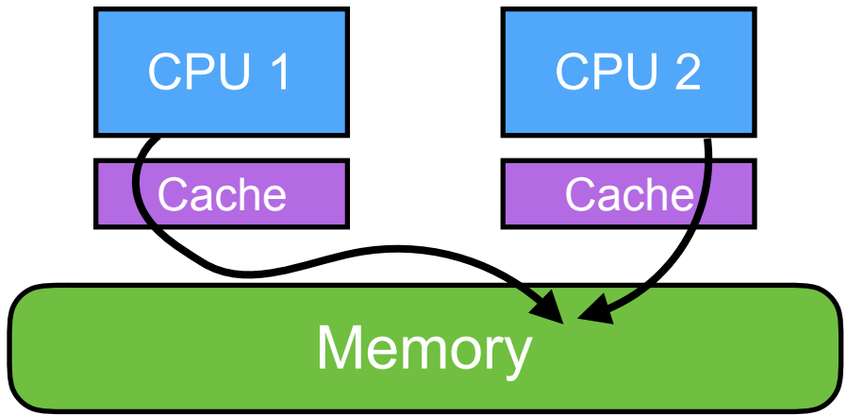
CC BY ND

Implications of Concurrency

- Issues with NUMA closely tied to those with concurrency

- Typical concurrent programming model: threads with shared mutable state
  - Note: introduction of threads into Unix was controversial – this isn't much appreciated now

- Difficult on multicore systems – caches give each core a different view of memory state
  - If reading from cache, potential for other core to have a different value cached
  - If reading from memory, latency penalty counters benefit of parallelism



CC BY ND

Cache Coherence in Concurrent Systems

- Shared mutable state → processors must have a coherent view of memory
  - At least, at synchronisation points defined by the language memory model
  - Potentially significant performance gains to be achieved by partitioning memory between processors
  - Scheduling threads that share memory on processors that share cache can result in significant speedup; equally – significant slowdowns occur when unrelated data sharing a cache line is accessed by multiple cores
    - Access by one core invalidates the cache, causing a flush to main memory and reload; the other core then accesses, and the data is flushed back – ping-pong occurs
    - Can causes slowdowns of many orders of magnitude
- Significant implementation complexity
- Complex programming model

CC BY ND

Is a Shared Memory Model Reasonable?

- Which is cheaper – message passing or shared memory?
  - Graph shows shared memory costs (1-8 cores, SHM1...SHM8) and message passing costs (MSG1...MSG8) for a 4 x quad-core server, with AMD HyperTransport
  - Cost of cache coherency protocols increases with the number of cores – messages can be cheaper, depending on the architecture
- Which is easier to program?
  - Shared-state concurrency is notoriously hard to program (locks, etc.)
  - Systems that avoid shared mutable state are frequently cited as easier to reason about
- For how long will it be reasonable to maintain illusion of shared memory?
  - For the hardware?
  - For the programmer?

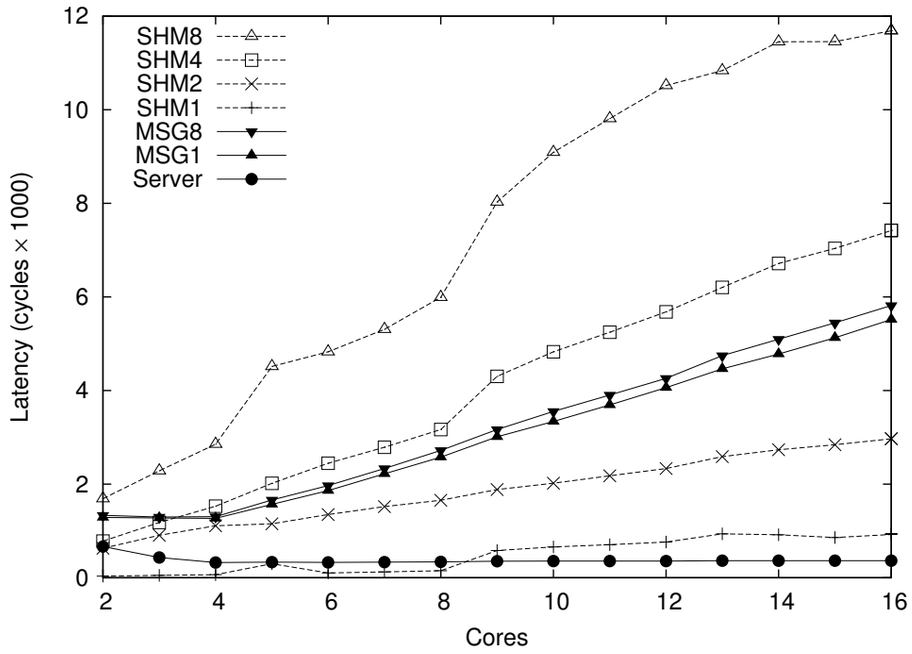


Figure 3: Comparison of the cost of updating shared state using shared memory and message passing.

Baumann *et al.*, "The Multikernel: A new OS architecture for scalable multicore systems", Proc. ACM SOSP 2009. DOI 10.1145/1629575.1629579

See also: F. Gaud *et* al., "Challenges of Memory Management on Modern NUMA Systems", Communications of the ACM, Vol. 58, Num.12, pp.59-66. Dec. 2015, DOI: 10.1145/2814328

CC BY ND

Implications of Solid State Storage

- Properties of storage are changing
  - Hard disks → solid state disks
  - Importance of data layout changing:
    - Was a performance optimisation
    - Becoming critical for correctness

Colin Perkins | https://csperkins.org/ | Copyright © 2018

Behaviour of hard disks

- Electro-mechanical device
  - Spinning disk platters
  - Moving read/write heads
  - Seek time, rotation → latency
- Poor random access performance: layout of data on disk important for good performance
  - File systems seek to avoid fragmentation within a file
  - Place frequently accessed files physically close to each other

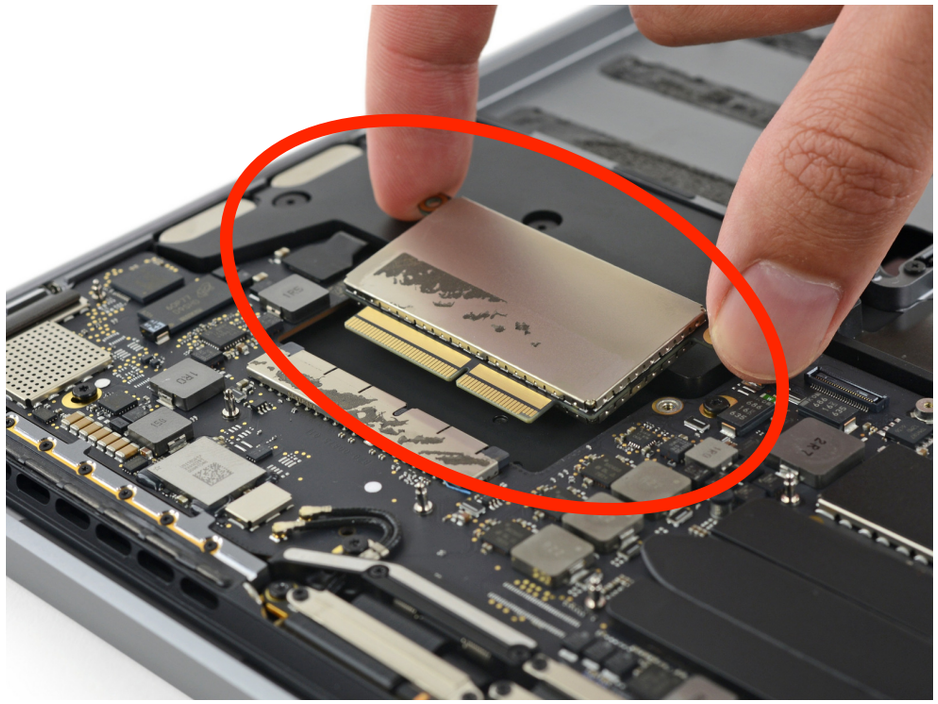


- Similar issues to the memory cache hierarchy, with much higher latency

Colin Perkins | https://csperkins.org/ | Copyright © 2018

Behaviour of solid state disks

- Non-volatile flash memory – no moving parts
- Much lower latency – much better random access performance
  - No need to wait for physical rotation of disk or movement of the heads
  - Location of files on disk unimportant for performance – file system complexity can be reduced
- Limited write cycles per block
  - Physical location of data on SSD matters for correctness – repeatedly writing to the same block will cause failures
  - Usually handled by SSD firmware without knowledge of the operating system
    - Firmware complexity becoming an issue – SSDs essentially run their own operating systems
    - Beneficial for operating system to tell the SSD “this data is deleted” rather than just discard it



Source: iFixit/CC-BY-NC-SA 3.0

CC BY ND

Implications for filesystems

- Most existing filesystems spend considerable effort to ensure good performance
  - Optimising placement of files
  - Avoiding fragmentation
- Now largely unnecessary with solid state storage
- Data integrity with limited write cycles now a more critical issue for filesystem design

CC BY ND

Implications of Performant Networking

- Modern network interfaces outperform the network stack
  - Multiple receive queues for concurrent connections
  - DMA to several cores simultaneously
  - Can process small packets at line rate – 64 byte packets on 10Gbps links gives very little time for the operating system to process each packet

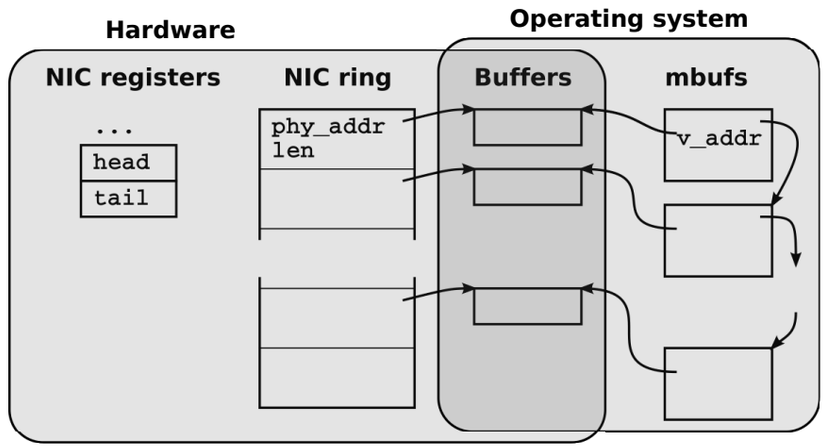


Figure 1: Typical NIC's data structures and their relation with the OS data structures.

Source: L. Rizzo. netmap: a novel framework for fast packet I/O. In Proceedings of the USENIX Annual Technical Conference, Boston, MA, USA, June 2012. USENIX.

- Traditional Berkeley sockets interface, with system call per packet, cannot deliver data at required rate
- Alternatives:
  - Operating system bypass – netmap, DPDK
  - New APIs – StackMap

CC BY ND

Conclusions

- The way we build operating systems has to change
  - Memory hierarchy, caching, NUMA, concurrency → threading unsuitable
  - Solid state storage → rethink file system internals
  - Performant networking → rethink network APIs and protocol design

Colin Perkins | https://csperkins.org/ | Copyright © 2018

Further Reading

- C. Mack, “The Multiple Lives of Moore’s Law”, IEEE Spectrum, Vol. 52, Num. 4, April 2015. DOI: 10.1109/MSPEC.2015.7065415
- R. Courtland, “The molten tin solution”, IEEE Spectrum, 53(11):28–41, November 2016. DOI: 10.1109/MSPEC.2016.7607024



MOORE'S LAW 50 YEARS

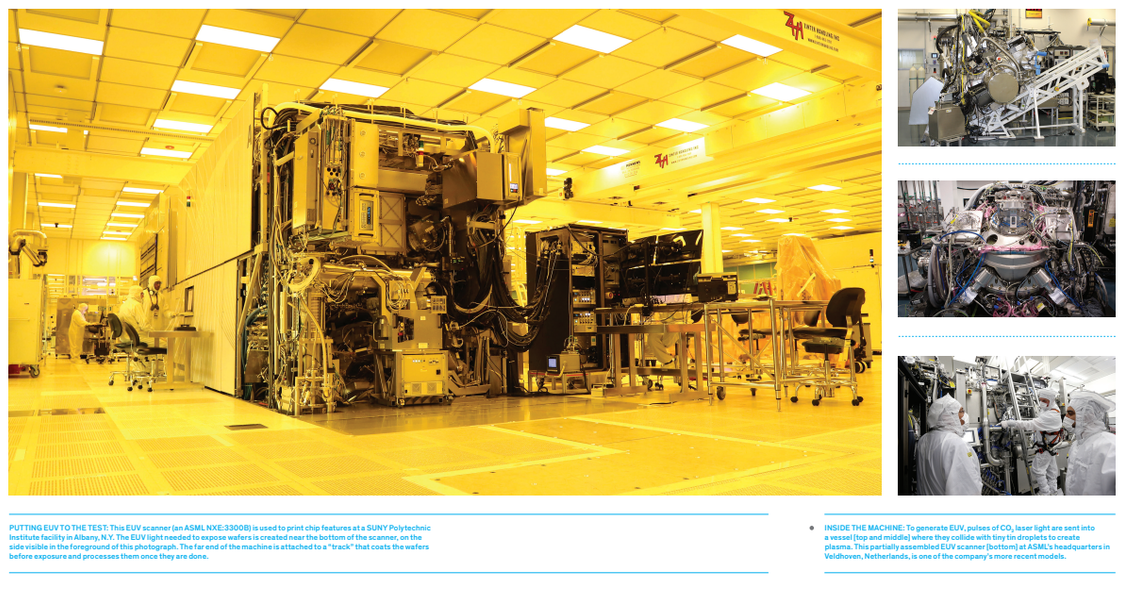
The MULTIPLE LIVES of MOORE'S LAW

Why Gordon Moore's grand prediction has endured for 50 years

By CHRIS MACK

A HALF CENTURY AGO, a young engineer named Gordon E. Moore took a look at his fledgling industry and predicted big things to come in the decade ahead. In a four-page article in the trade magazine *Electronics*, he foresaw a future with home computers, mobile phones, and automatic control systems for cars. All these wonders, he wrote, would be driven by a steady doubling, year after year, in the number of circuit components that could be economically packed on an integrated chip. • A decade later, the exponential progress of the integrated circuit—later dubbed “Moore’s Law”—showed no signs of stopping. And today it describes a remarkable, 50-year-long winning streak that has given us countless forms of computers, personal electronics, and sensors. The impact of Moore’s Law on modern life can’t be overstated. We can’t take a plane ride, make a call, or even turn on our dishwashers without encountering its effects. Without it, we would not have found the Higgs boson or created the Internet. • But what exactly is Moore’s Law, and why has it been so successful? Is it evidence of technology’s inevitable and unstoppable march? Or does it simply reflect a unique time in engineering history, when the special properties of silicon and a steady series of engineering innovations conspired to give us a few decades of staggering computational progress?

SPECTRUM.IEEE.ORG | NORTH AMERICAN | APR 2015 | 31



The Molten Tin Solution

BY RACHEL COURTLAND

After a 30-year journey, EUV chip-printing technology is now poised to help keep Moore’s Law on track

PUTTING EUV TO THE TEST: This EUV scanner (an ASML NXE:3300B) is used to print chip features at a SUNY Polytechnic Institute facility in Albany, N.Y. The EUV light needed to expose wafers is created near the bottom of the scanner, on the side visible in the foreground of this photograph. The far end of the machine is attached to a “track” that coats the wafers before exposure and processes them once they are done.

INSIDE THE MACHINE: To generate EUV, pulses of CO_2 laser light are sent into a vessel [top and middle] where they collide with tiny tin droplets to create plasma. This partially assembled EUV scanner [bottom] at ASML’s headquarters in Veldhoven, Netherlands, is one of the company’s more recent models.

28 | NOV 2016 | NORTH AMERICAN | SPECTRUM.IEEE.ORG

LEFT: IBM RESEARCH; RIGHT: ASML (3)

SPECTRUM.IEEE.ORG | NORTH AMERICAN | NOV 2016 | 29

Colin Perkins | https://csperkins.org/ | Copyright © 2018