1. Computing Technologies – Retrospect & Prospect

Computing Systems Architecture

Instructor: Azeez Bhavnagarwala

ECE 6913, Fall 2021

Computer Systems Architecture

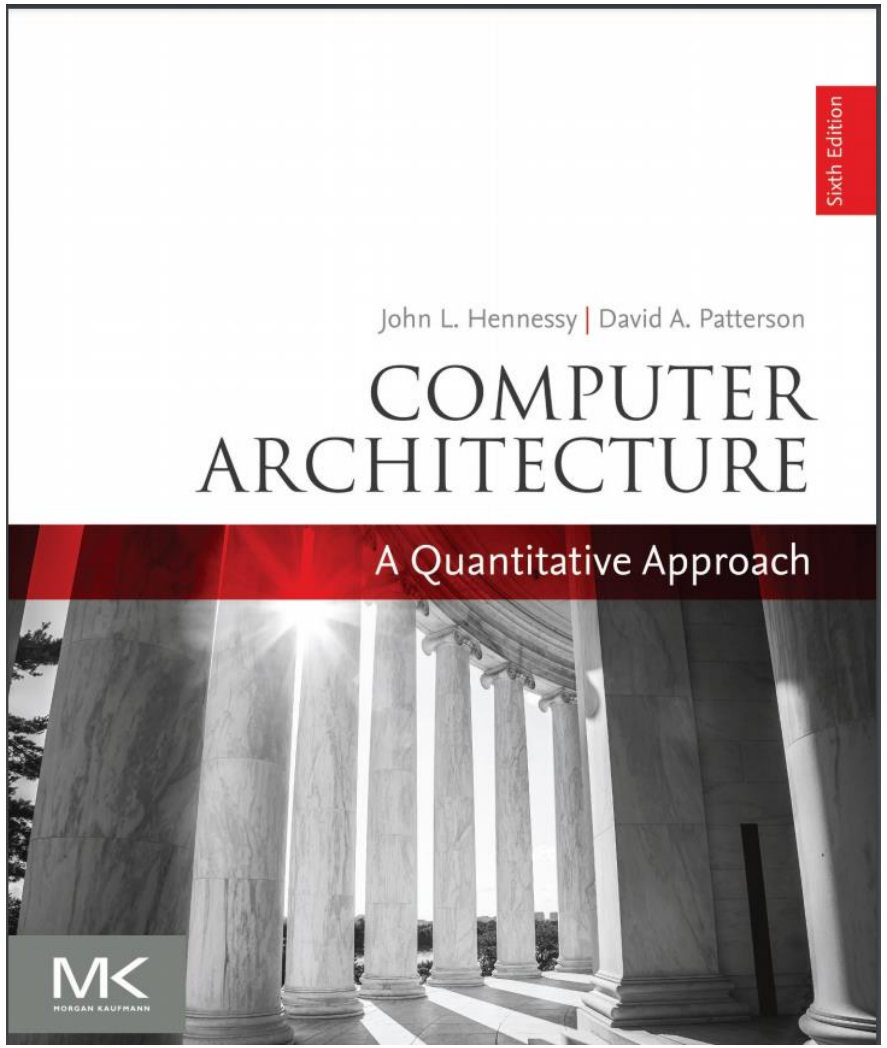
NYU Tandon School of Engineering

Course Highlights

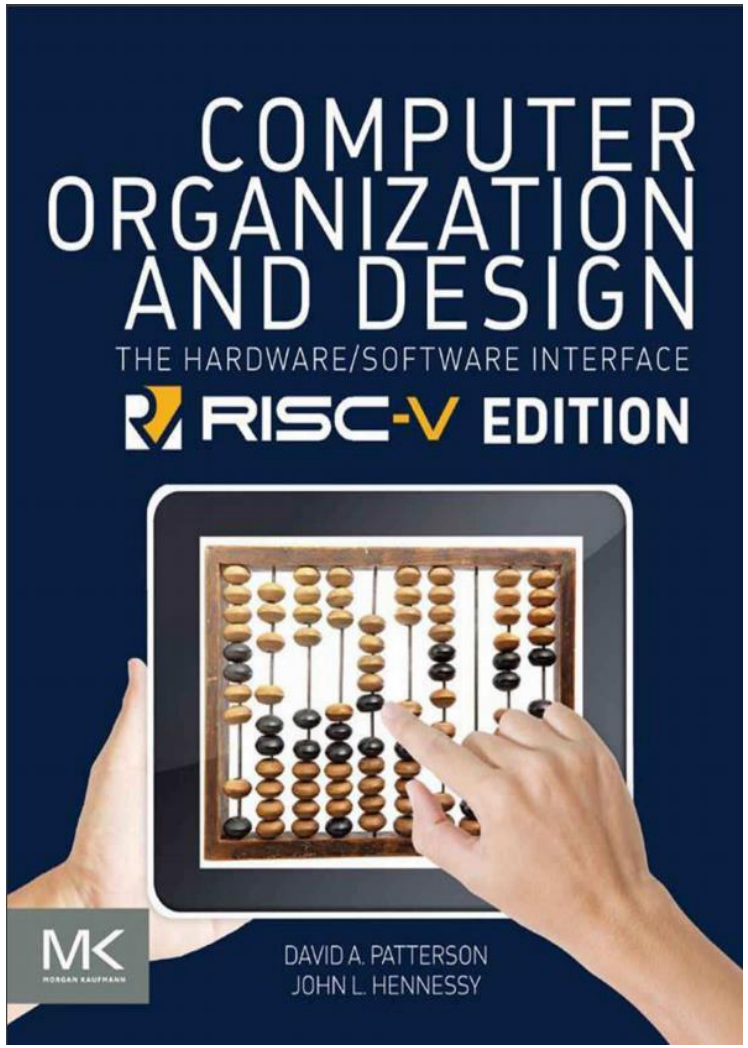
- Historical Context of Computing Technologies and their evolution
- Instruction Set Architecture – RISCV in focus and its comparison to older ISAs
- Floating Point Hardware Implementation. IEEE 754 Standard
- Classic 5-stage RISC Pipeline, Performance limitations from Hazards, solutions
- Memory Hierarchy – comparisons between Intel Quad Core i7 with the ARM Cortex A-53
- Parallel Processing, Flynn's taxonomy of SISD/MIMD/SIMD/SPMD machines, Multithreading and Multicore processors
- Domain Specific Architectures for AI workloads

Textbooks

Computer Architecture:
A Quantitative Approach" [6th Edition]



Computer Organization & Design,
Hardware-Software Interface – RISC-V Edition



Semester Schedule

Week	Date	ECE 6913 Content	Assignments
1	Sept 2	Quantitative Design & Analysis, Physical limits on scaling CMOS – End of Dennard Scaling and Moore's Law. Evolution of System architecture and cost with Open Source ISA, DSAs and Wafer Scale Engines	HW 1
2	Sept 9	Introduction to the RISC-V, RISC-V Instructions, Instruction formats, memory management	HW 2
3	Sept 16	Floating point Arithmetic for Computers – IEEE 754 representation,	HW 3
4	Sept 23	Pipelining: Basic & Intermediate concepts. Classic 5 stage RISC processor pipeline.	HW 4
5	Sept 30	Pipeline Hazards, Pipelining implementation.	HW 5 (RISCV project)
6	Oct 7	**Quiz 1**	
7	Oct 14	Open-Source RISCV ISA review. Comparisons with older ISAs. RV32FD – FP registers, FP load/stores/arithmetic, FP moves/converts, RV32C - Compressed Instructions, RV32V -Vector Extensions,	HW 6
8	Oct 21	Introduction to Memory Hierarchy: Memory technologies, Caches, Cache performance.	HW 7
9	Oct 28	Virtual Machines, Virtual Memory, FSM for simple cache controller, Cache coherence	HW 8
10	Nov 4	Review of ARM Cortex A-53 and the Intel quad Core i7 6700 Memory hierarchy	HW 9 Quiz 2 Review
11	Nov 11	**Quiz 2**	
12	Nov 18	Introduction to Parallel Processing SISD, MIMD, SIMD, SPMD, and Vector machines, Hardware Multithreading	HW 10 Publication Review
13	Nov 25	**Thanksgiving (no class)**	
14	Dec 2	Multicore and Other Shared Memory Multiprocessors	
15	Dec 9	Review for Final	
16	Dec 16	Final	

Weekly Schedule

ECE 6913	MON	TUE	WED	THU	FRI
9:00 -11:00	**Office Hours** CA1 Rm 808	**Office Hours** CA2 Rm 808	**Office Hours** CA3 Rm 808	**Office Hours** CA4 Rm 808	**Office Hours** CA5 Rm 808
11:00 – 1:30		**Office Hours** Instructor Rm 817		**Lecture** 370 Jay, Rm 202 Instructor	
2:00 – 4:00					
4:00 – 5:50					
6:00 – 9:00					
					Weekly HW due by 11:55 PM

Course Assistants

- Karan Parikh kap9580@nyu.edu
- Sahil Chitnis ssc9983@nyu.edu
- Kewal Jani kj2062@nyu.edu
- Zhiming Fan zf2035@nyu.edu
- Haotian Zheng hz2687@nyu.edu
- Shan Hao sh6206@nyu.edu

Course Prerequisites

- Basic knowledge of digital logic and Computer Architecture is assumed.
- If you have not taken an undergraduate level class on Computer Architecture, you will need to supplement course work with additional preparation
  - Review Ch 1, 2, 4, 5 (from Textbook 2 - RISC V edition) early in the semester

Course Structure

- Your performance in the course will be assessed via weekly HW assignments (**20%** of total grade)
  - Project (HW 5: 5% of grade) and
  - Publication Reviews (HW 10: 5% of grade),
- 2 Midterms (**50%** of total grade)
- Final (**25%** of total grade).
- Class Participation (**5%** of grade)
- There could also be (extra credit) pop-quizzes and (extra credit) HW assignments. Participation in these is highly encouraged.

Course Content

- Weekly lecture videos, slide sets and HW assignments, reading assignments to be made available on NYU Brightspace
- Homework Assignments are due weekly on Fridays 11:55 PM
- Please submit HW assignments as PDFs (from Word Docs) with your identifying information in the name of the file: ajb20_HW4.pdf
  - Not on hand-written sheets of paper.
  - Schedule pressures from projects/tests in other classes – please let me know in advance
  - Other challenges/concerns – can advocate for you.

NYU Policies on Academic Honesty

- NYU expects and requires its students to adhere to the highest standards of scholarship, research and academic conduct.
- Essential to the process of teaching and learning is the periodic assessment of students' academic progress through measures such as papers, examinations, presentations, and other projects.
- Academic dishonesty compromises the validity of these assessments as well as the relationship of trust within the community.
- Students who engage in such behavior will be subject to review and the possible imposition of penalties in accordance with the standards, practices, and procedures of NYU and its colleges and schools.
- Violations may result in failure on a particular assignment, failure in a course, suspension or expulsion from the University, or other penalties

Today's Agenda

- Historical developments in Computer technology and architecture from 'Tabulators' (1890) to Microprocessors (present)
- Classes of and markets for Computers as computer performance and cost improved by 3 orders of magnitude with innovations in Integrated circuits (CMOS) and in architecture (RISC) during the 17 years of the 'renaissance' period of 1986 – 2003
- Trends and limits of semiconductor technology as we enter an age where transistor density and cost scale 10x – 20x slower per year, Dennard scaling is dead and Moore's Law has considerably slowed.
  - Room for chip cost to scale with 'Open-source' RISC Instruction Sets & Design Automation – Design chips with 1 Engineer in < 30 minutes

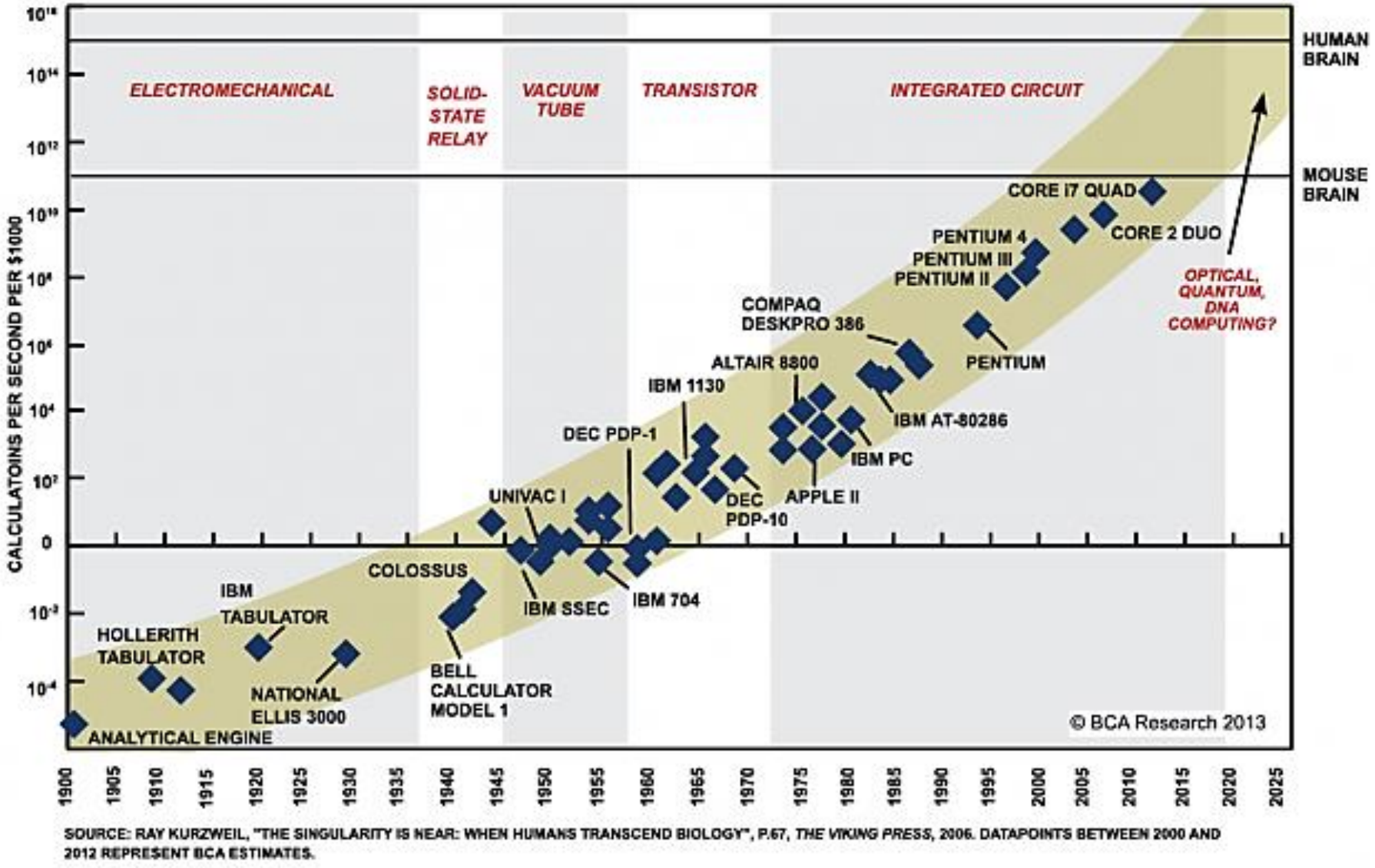
  https://www.darpa.mil/program/intelligent-design-of-electronic-assets
  - Room for chips to scale in number of functions – 2 trillion transistor, mouse-pad size chips:

  https://www.reuters.com/technology/cerebras-systems-connects-its-huge-chips-make-ai-more-power-efficient-2021-08-24/
  - Room for chip performance to scale: Domain Specific Architectures

  https://hc32.hotchips.org/assets/program/conference/day1/HC32-K1-Intel-Raja-2020.pdf (slides 17-18)
- Quantitative analysis of performance, power and reliability of a computer

The Computer Revolution

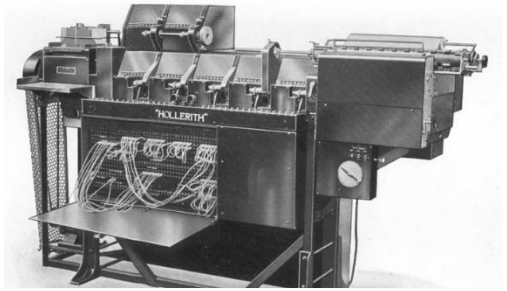
- Primary metric: Calculations per second per $1K
- Improved **18 orders of magnitude** in ~ **120 years**
- Unprecedented across **any industry**
- Impact to most other industries
- Created revenues, jobs, wealth for nations



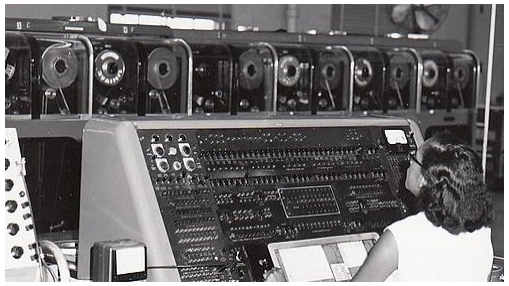
SOURCE: RAY KURZWEIL, "THE SINGULARITY IS NEAR: WHEN HUMANS TRANSCEND BIOLOGY", P.67, *THE VIKING PRESS*, 2006. DATAPOINTS BETWEEN 2000 AND 2012 REPRESENT BCA ESTIMATES.

Technology Evolution

Electromechanical → Vacuum Tubes → Transistor → Integrated Circuit

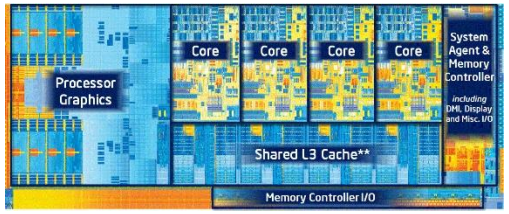


- IBM's Hollerith **Type 3 'Tabulator'**, 1932
- Basic function is to count and/or add from punched cards
- Could collect, count data more rapidly and accurately than manually
- *Electromechanical Technology – used 1890 - 1949*



- **Univac I**: 5000 vacuum tubes, 125kW, 1951
- Addition – 525 mS, multiplication 2150 mS. Memory - 1K words (12 char each)
- I/O with magnetic tape drives – 7,200 characters/sec
- Famously used by CBS in 1952, predicting Eisenhower win against Stevenson (favored in polls) *Vacuum Tube Technology – used 1945 - 1959*

- DEC's **Programmed Data Processor 1 (PDP 1)**, 1961
- 2,700 transistors and 3,000 diodes. 4Kb – 64Kb Magnetic Cores memory
- Arithmetic instructions use 2 memory cycles, 187 MHz
- *Transistor based computers – used 1959 - 1973*



- Intel's **Ivy Bridge i7 3770 Microprocessor,** 2012
- 1.4B transistors, integrated GPU alongside 4 performance cores, 4.2GHz
- *MOS Integrated Circuit based microprocessors – used 1971 - present*

What is Dennard Scaling ?

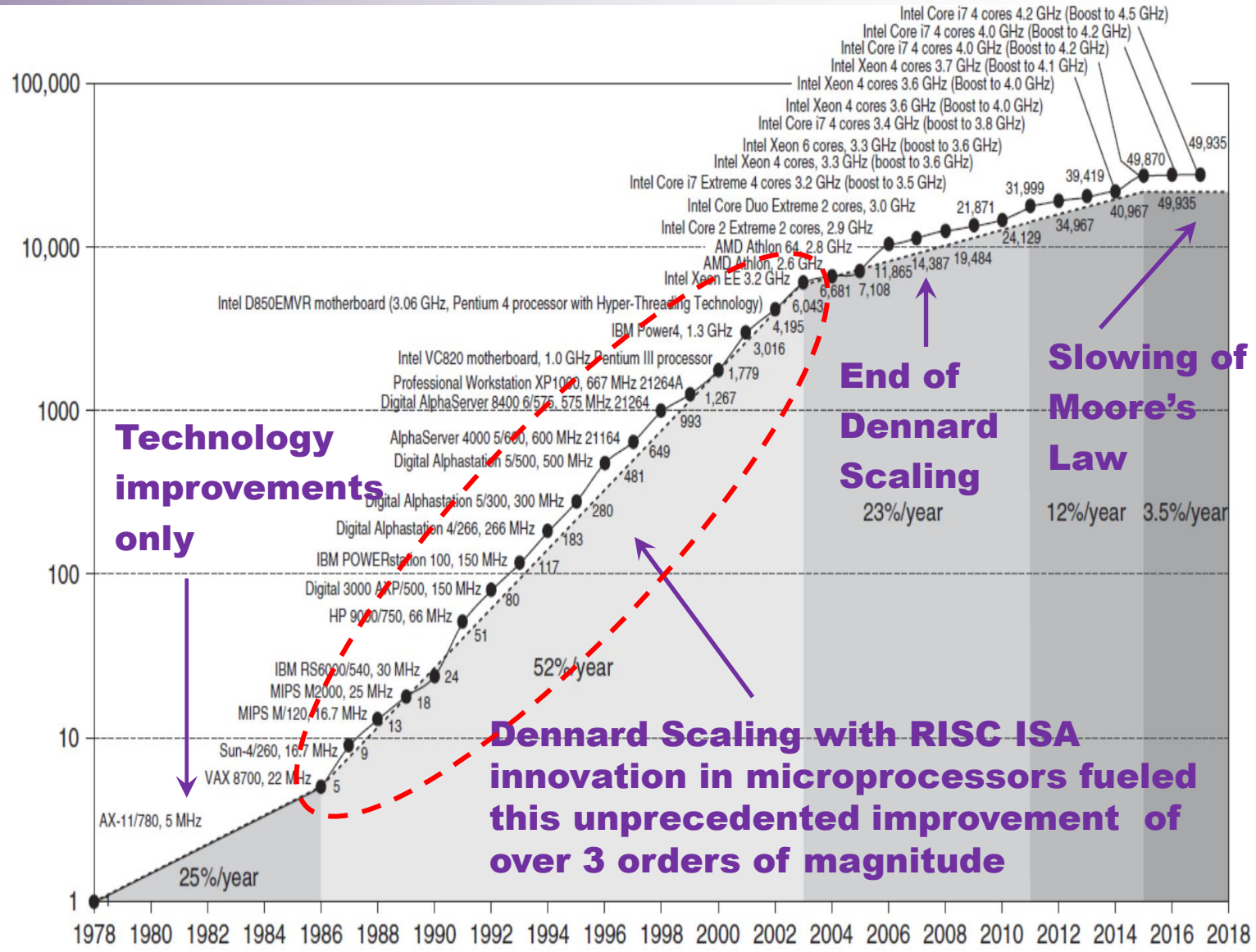
- Constant Electric Field scaling in MOS devices
  - Scale by 30% MOSFET geometries *along with* voltages & current to maintain constant electric field
  - Circuit (i) delays drop by geometry scaling factor (30%), (ii) Power drops by *square of* scaling factor (0.7 x 0.7 ~ 0.5) and (iii) area of circuit footprint (and cost) drops by *square of* scaling factor (~ 0.5). Maintain (iv) power density constant
  - **Unlike** previous technologies that consumed more power, area, cost as they delivered more performance (vacuum tubes, transistors… and also steam locomotives in the industrial age), single-chip CPUs (Microprocessors) delivered more performance at lower cost, power simultaneously with Dennard scaling

Constant field scaling

Parameter	Scaled parameter
Channel length (L)	$1/\alpha$
Width (W)	$1/\alpha$
Substrate doping (N_A)	α
Depletion layer thickness (d)	$1/\alpha$
Gate delay (τ_p)	$1/\alpha$
Load capacitance (C_L)	$1/\alpha$
Supply voltage (V)	$1/\alpha$
Gate oxide thickness (t_{ox})	$1/\alpha$
Current (I)	$1/\alpha$
Current density (J)	α
Transconductance (g_m)	1
Junction depth (x_j)	$1/\alpha$
Static power dissipation (P_{stat})	$1/\alpha^2$
Dynamic power dissipation (P_{dyn})	$1/\alpha^2$

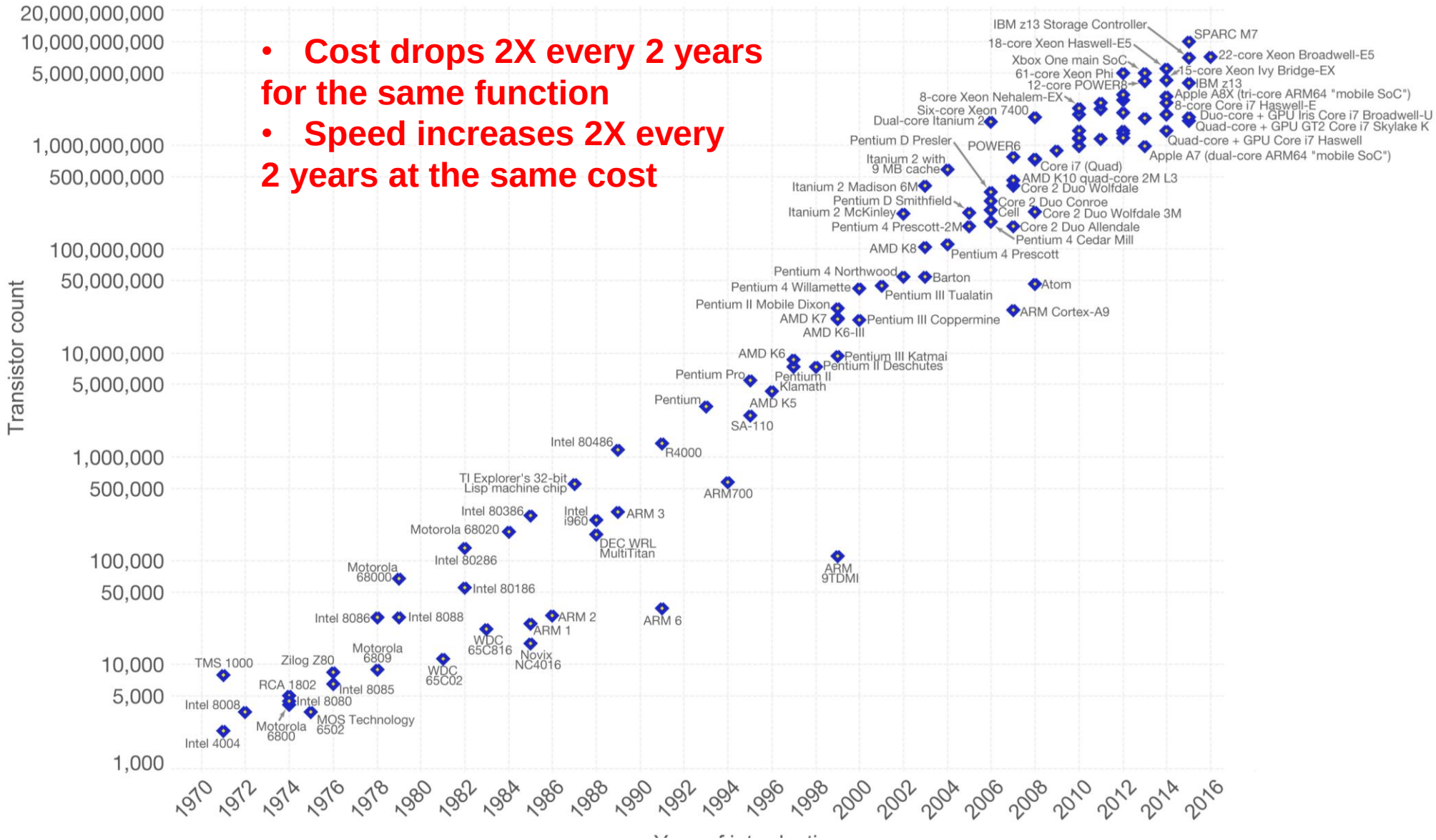
Dennard Scaling & ISA Innovation

- MOS transistor scaling increases their speed and lowers cost, power drain - with power density constant – Dennard Scaling
- Dennard scaling ended 2004 when operating voltages could not be scaled proportionately with transistor geometry
- Elimination of Assembly programming & creation of standardized vendor-independent OS (UNIX/Linux) *lowered cost and risk of developing new ISAs with simpler instructions*: RISC (mid 80's)
- RISC machines focused designers on 2 performance techniques:
  - Instruction level parallelism
  - Caches
- Intel translates x86 instructions to RISC internally at negligible overheads in high end CPUs
- Cost & power overheads of x86 translation in low-end applications helped ARM (RISC) dominate the mobile space

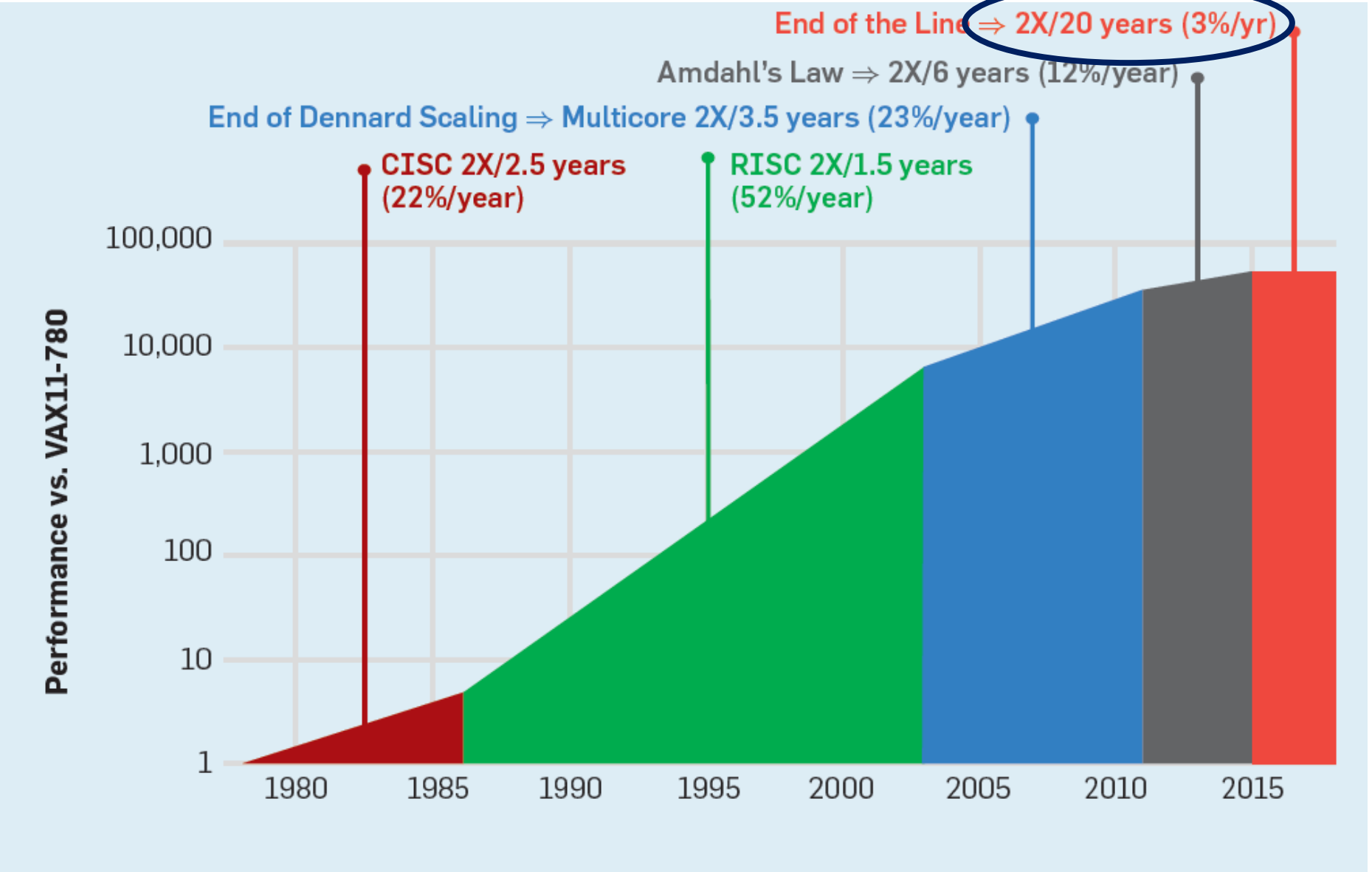


Moore's Law

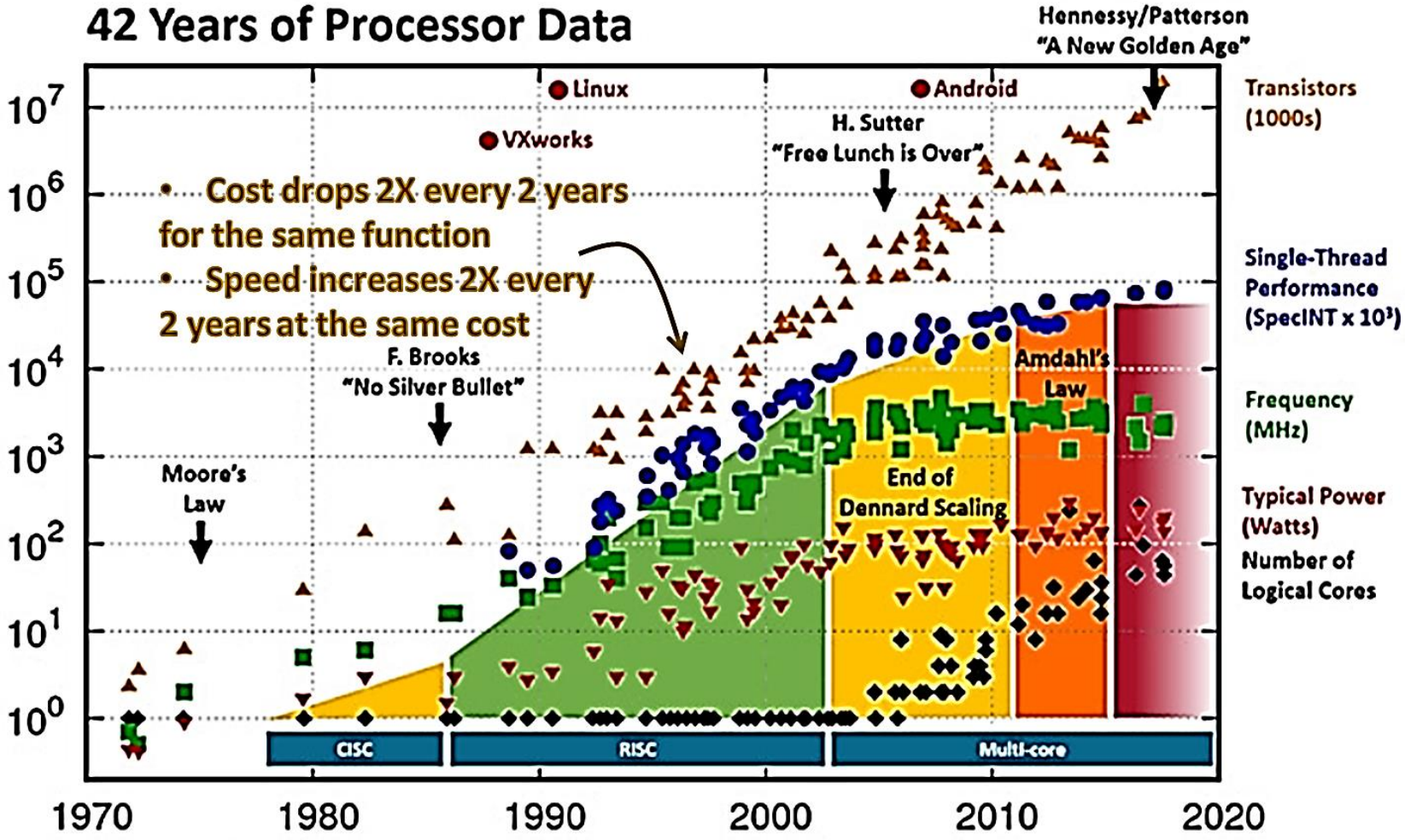
- **Cost drops 2X every 2 years for the same function**
- **Speed increases 2X every 2 years at the same cost**



End of Moore's Law?

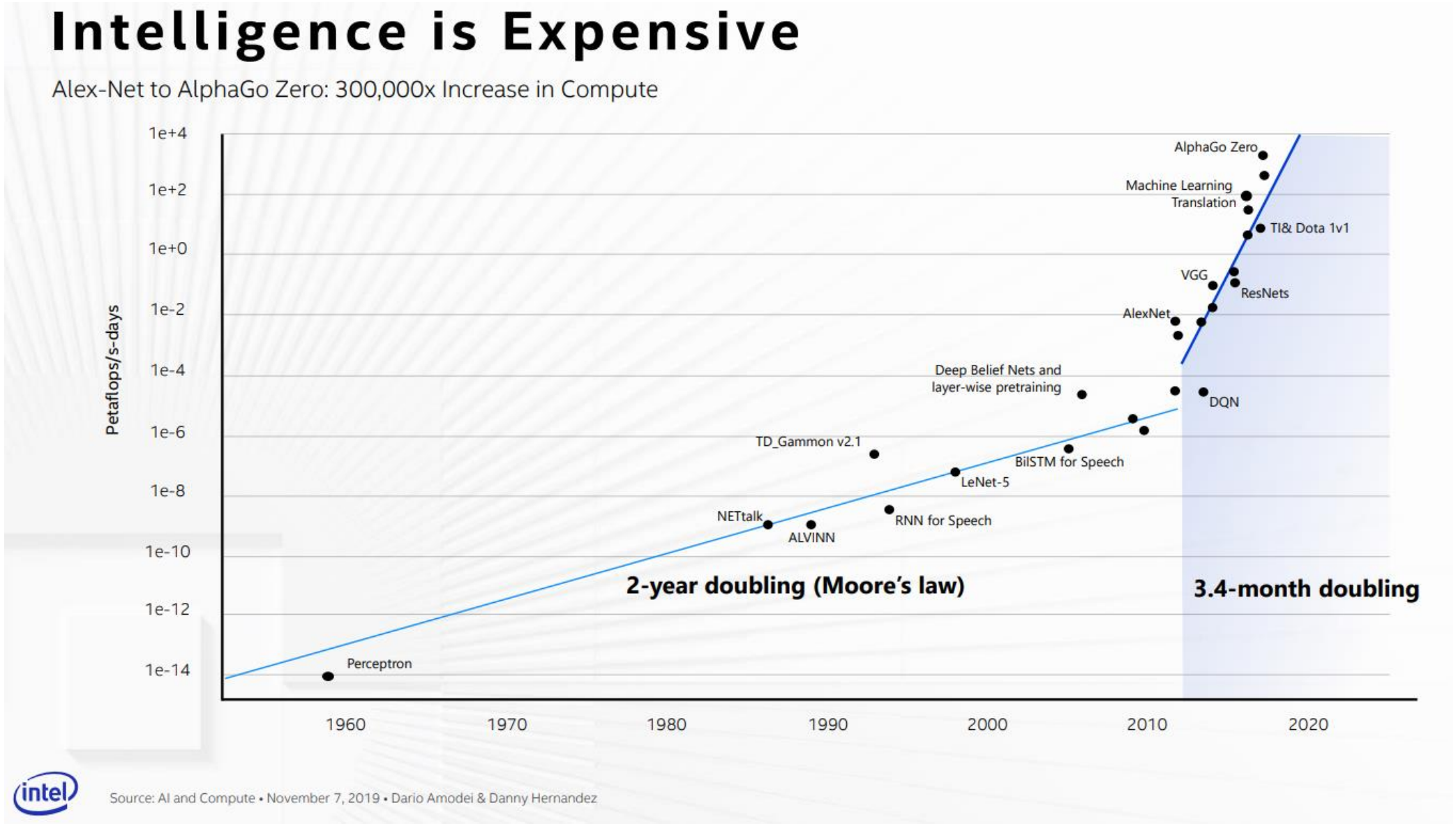


Chip Design – What is changing?



Hennessy and Patterson, Turing Lecture 2018, overlaid over "42 Years of Processors Data"
https://www.karlrupp.net/2018/02/42-years-of-microprocessor-trend-data/
Original data up to the year 2010 collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond, and C. Batten

Hot Chips 2020

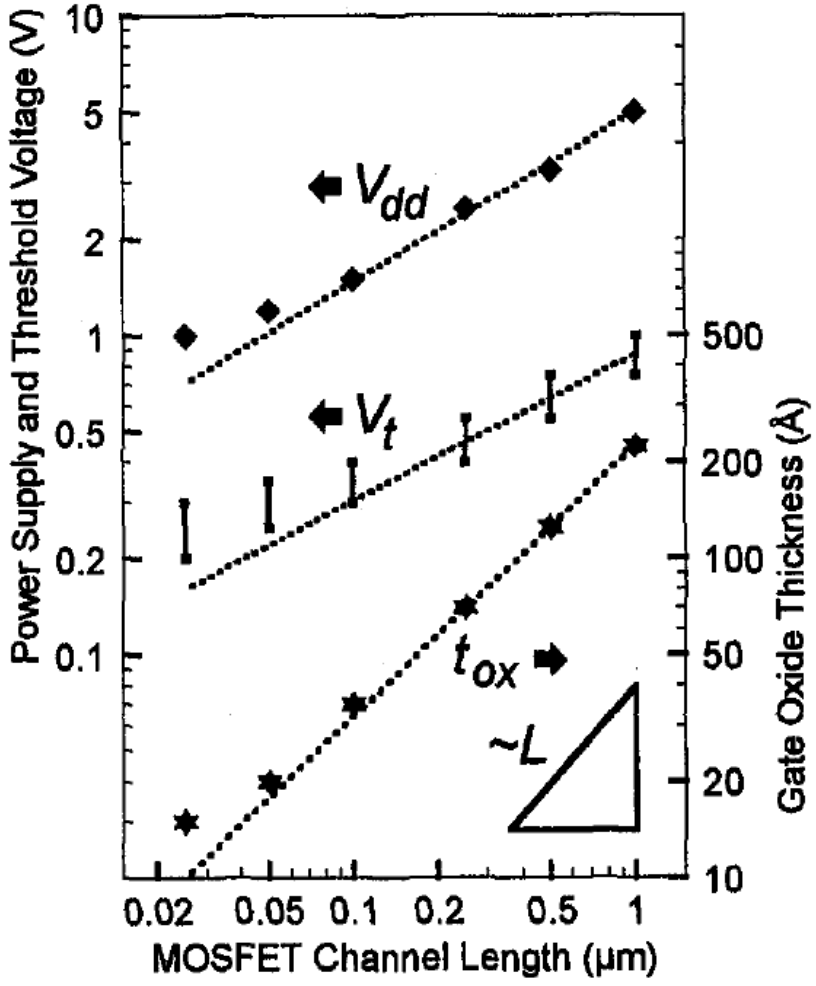


Why did Dennard Scaling end ?

End of Dennard Scaling

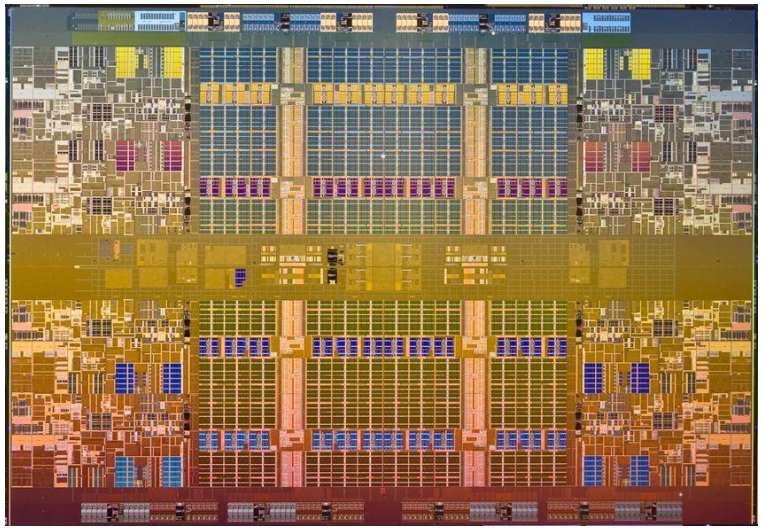
From 2004 onward, CMOS voltages (Vdd) and currents *could not scale with MOS geometry* due to *leakage (slide 49), variability and performance, constraints*

- Microprocessors had to use multiple efficient processors operating at lower voltages (and higher gate delays) instead of a single inefficient processor at unscaled voltage with faster circuits
- Trade off circuit speed for lower voltage and make up for circuit performance loss with higher throughput performance using multiple cores

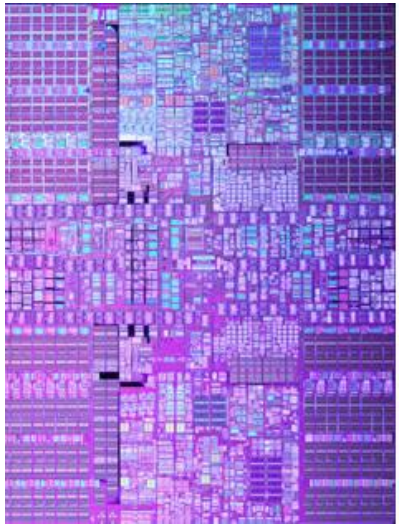


End of Dennard Scaling

- **Architecture Impacts**
  - In 2004 Intel cancelled high-performance uniprocessor projects and joined others on the road to higher throughput performance with multiple processors per chip
  - First roll-out of *Dual Cores*: Xeon (Intel, 2004), Power5,6 (IBM, 2003,2006)
  - *Historic switch* from relying on Instruction Level Parallelism (ILP) alone to Data Level Parallelism, Thread Level Parallelism and Request Level Parallelism
  - Compilers and Hardware conspire to *exploit ILP implicitly without* Programmers attention.
  - DLP, TLP and RLP are *explicitly parallel* requiring restructuring of the application to exploit *explicit parallelism*
  - In most cases it is burdensome for programmers



Intel, Dual core Xeon, 2004



IBM, Dual core, Power6 2005

Classes of Parallelism

- Parallelism at multiple levels drives computer design across all classes of computers – with energy & cost being the primary constraints
- Classes of application parallelism:
  - Task-Level Parallelism (TLP) exists because tasks of work are created that can operate independently and in parallel
  - Data-Level Parallelism (DLP) exists because there are many data items that can be operated on by the same instruction at the same time
- The above classes of parallelism can be exploited 4 ways:
  - Instruction-Level Parallelism (ILP): exploits DLP at modest levels with compiler help using ideas like pipelining and speculative execution
  - Vector architectures/Graphic Processor Units (GPUs): exploits DLP by applying a single instruction to a collection of data in parallel
  - Thread-Level Parallelism: exploits DLP or TLP in a tightly coupled hardware model that allows for interaction between parallel threads
  - Request-Level Parallelism: exploits parallelism among largely decoupled tasks specified by the programmer or OS

Impacts from Dennard Scaling + RISC

- Enhanced capability available to Users
  - Microprocessors outperformed Supercomputers from less than 20 years before they were introduced
  - MOS scaling enabled Microprocessors to improve performance *and* lower power, cost unlike previous computer & memory technologies
- New classes of computers emerged
  - Increasing chip speed *and* lowering chip cost created *pervasive new markets* for chips ($ 33B in 1987 to over $ 200B in 2004)
  - Multiple energy-performance-cost design points (over $ 460B in 2018)
  - Manifesting in Warehouse Scale, Servers, Personal Computers, Personal Mobile Devices, Embedded Computers/IoT… and now in Accelerators

Classes of Computers

Feature	Personal mobile device (PMD)	Desktop	Server	Clusters/warehouse-scale computer	Internet of things/ embedded
Price of system	$100–$1000	$300–$2500	$5000–$10,000,000	$100,000–$200,000,000	$10–$100,000
Price of microprocessor	$10–$100	$50–$500	$200–$2000	$50–$250	$0.01–$100
Critical system design issues	Cost, energy, media performance, responsiveness	Price-performance, energy, graphics performance	Throughput, availability, scalability, energy	Price-performance, throughput, energy proportionality	Price, energy, application-specific performance

- Explosive and unprecedented (52% per year) improvement in performance, cost during the renaissance period of 1986 – 2003 from Dennard scaling and ISA innovation created new markets for computing and several classes of computers to fill these – that were not possible with previous technologies such as vacuum tubes or discrete transistors.
- The unique character of the semiconductor industry where both performance and cost improved during this period enabled microprocessors (single chip CPUs) to become as pervasive as they are today

1. Embedded Computers

- Widest range of processing power, cost (1 ¢ → $ 100s) & Largest class of computers – widest range of applications.
- Processors in your car, TV, microwave, washer, wi-fi router etc. Users do not even realize they are using a 'computer'
- Designed to run only one application – hardwired to do so as a component of a system
- Reliability achieved through simplicity – emphasis on doing one function as perfectly as possible.
- Stringent power and cost constraints on performance
- Do not have the ability to run third-party software

2. Internet of Things

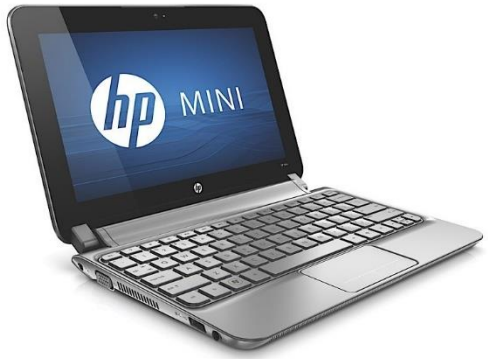
- IoT refers to embedded computers connected to the internet, typically wirelessly
- With sensors, actuators & wireless connectivity IoT devices collect useful data and interact with physical world
- Myriad of ‘smart’ applications result: smart watches, smart thermostats, smart cars, smart homes, smart grids etc.

3. Personal Mobile Devices

- Wireless devices with multimedia user interface
- Cell phones, Tablet computers with web-based and media-oriented applications
- Primary design constraint: cost, energy efficiency
- Energy efficiency of processor not just driven by battery lifetime and weight – also by cost of chip packaging (plastic Vs ceramic) and the need to remove heat without a fan
- Energy efficiency and form factor of PMD require use of (more expensive) Flash instead of magnetic disks
- PMD processors are considered embedded computers, but also are platforms that run externally developed software – like PCs

4. Desktop Computing

- First & possibly still the largest market for processors in $
- Spans low-end netbooks (<$300) to heavily configured desktops (>$2500)
- Since 2008, more than half of PCs have been battery powered laptops
- Single user, low cost, 3rd party software
- General purpose, variety of software, programs
- Subject to cost/performance tradeoff



5. Servers

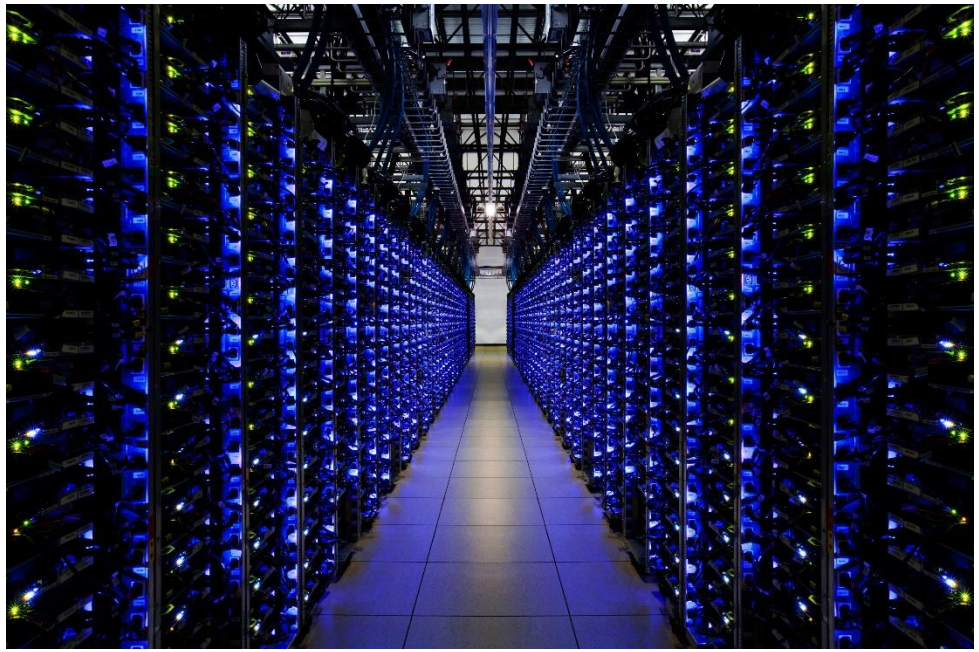
- Role of servers: provide larger scale and more reliable file and computing services
- Backbone of large scale enterprise computing replaced the traditional mainframe
- Availability is critical – servers running ATM machines for Banks and airline reservation systems 24/7 – failure can be catastrophic

Application	Cost of downtime per hour	Annual losses with downtime of		
		1% (87.6 h/year)	0.5% (43.8 h/year)	0.1% (8.8 h/year)
Brokerage service	$4,000,000	$350,400,000	$175,200,000	$35,000,000
Energy	$1,750,000	$153,300,000	$76,700,000	$15,300,000
Telecom	$1,250,000	$109,500,000	$54,800,000	$11,000,000
Manufacturing	$1,000,000	$87,600,000	$43,800,000	$8,800,000
Retail	$650,000	$56,900,000	$28,500,000	$5,700,000
Health care	$400,000	$35,000,000	$17,500,000	$3,500,000
Media	$50,000	$4,400,000	$2,200,000	$400,000

Costs rounded to nearest $100,000 of an unavailable system are shown by analyzing the cost of downtime (in terms of immediately lost revenue), assuming three different levels of availability, and that downtime is distributed uniformly.

6. Warehouse-Scale Computers/Clusters

- A cluster is a collection of desktop computers or servers connected together by a local area network to *act as a single larger computer*.
- A warehouse-scale computer (WSC) is a cluster comprised of *tens of thousands of servers*.
- The cost may be on the order of $150M for the building, electrical and cooling infrastructure, the servers, and the networking equipment that houses *50,000 to 100,000 servers*.
- A WSC can be used to provide *internet services*.
  - search - Google
  - social networking - Facebook
  - video sharing - YouTube
  - online sales - Amazon
  - cloud computing services - Rackspace
  - and many more applications



Computer Design

- Determine what attributes are important for the Computer
- Design to maximize performance & energy efficiency within cost power & availability constraints
  - Instruction set design
  - Functional Organization
  - Logic Design
  - Implementation
    - Integrated circuit design
    - Packaging
    - Power delivery
    - Cooling
- Optimize across compilers & OS to logic design & packaging

Instruction Set Architecture

- ISA serves as the *interface* between software and hardware
- Software that has been written for an ISA *can run on different implementations* of the same ISA: mobile device, laptop, server
- Enables *binary compatibility between different CPUs* using the same ISA to be easily achieved
- An ISA defines everything a machine language programmer needs to know in order to program a computer.
- Realization of an ISA is called an *Implementation* with 2 components:
  - *Organization* (also called '*microarchitecture*'): high level aspects – memory system, memory interconnect, CPU (where arithmetic, logic, branching, data transfer etc. is done) design. Examples: AMD Opteron and Intel Core i7: same ISA but different pipeline and cache organization
  - *Hardware*: logic design and packaging technology. Examples: Intel Core i7 and Intel Xeon E7: Same ISA, similar organization but different clock rates and different memory systems with Xeon 7 more effective for servers
- The architecture of a computer covers all 3 aspects: ISA, microarchitecture and hardware.

Classification of Instruction Set Architectures

- **Classification by memory access:**
  - Two popular versions of general purpose register architectures where instruction operands are either *registers* or *memory locations are:* (1) *register-memory ISAs* and (2) *load-store ISAs*.
  - Register-memory ISAs such as the x86 ISA *can access memory as part of many instructions* and load-store ISAs such as ARMv8 or RISC-V can *access memory only with load and store instructions*
  - All ISAs announced since 1985 are load-store ISAs
- **Classification by architectural complexity:**
  - **CISC** (Complex Instruction Set Computer) ISAs have *many specialized instructions*, some of which may only be rarely used in practical programs
  - **RISC** (Reduced Instruction Set Computer) simplifies the processor by efficiently *implementing only the instructions that are frequently used* in programs, while the less common operations are implemented as subroutines, having their resulting additional processor execution time offset by infrequent use

CISC

- CISC: Single instruction executes *several low-level operations*
  - For e.g., Load from memory, an arithmetic operation, a memory store
  - Multi-step operations, addressing modes *within single instructions*
  - 'CISC' *retroactively* coined in contrast to 'RISC'
  - CISC became a catch-all term where *arithmetic instructions also perform memory accesses* meaning *anything that's not a load-store* (RISC) architecture
  - It's *not the number* of instructions, *nor the complexity* of the implementation or of the instructions themselves - that define CISC
- The compact nature of CISC results in
  - *Smaller* program sizes
  - *Fewer* (slow) main memory accesses
  - Tremendous saving on the *cost of computer memory and disc storage*, as well as *faster execution*
  - good *programming productivity* in assembly language, as high level languages such as Fortran or Algol were not always available or appropriate

CISC

- PDP-10, PDP-8, Intel 80386, Intel 4004, Motorola 68000, Sys z mainframe, Burroughs B5000, VAX etc.
  - *Vary wildly* in the number, sizes, and formats of **instructions**, the number, types, and sizes of registers, and the available data types
  - Are all in the CISC category because they have "load-operate" instructions that load and/or store memory contents *within the same instructions that perform the actual calculations*

RISC

- A *small* set of *simple* and *general* instructions, rather than a large set of complex and specialized instructions
- RISC allows *fewer cycles per instruction* (CPI) than a complex instruction set computer (CISC)
  - AKA …'Relegate Interesting Stuff to the Compiler'
  - RISC designs use *uniform instruction length*
  - Employ *strictly separate* load/store-instructions (can access memory only with load/store instructions)
- First RISC system: IBM 801 design - began in 1975 by John Cocke and was completed in 1980
- Most popular RISC processors: from ARM – *in over 16.7B chips shipped in 2016 alone* – over 50X the total number of x86 chips ever
- RISC-V: modern ISA developed at UC Berkeley: Free Open ISA – opportunity to enable *ultra-low-cost ubiquitous computing in IoT devices* – with over 60 industry members part of the RISC-V foundation including AMD, Google, HP, IBM, Microsoft, Nvidia, Qualcomm, Samsung and Western Digital

Break !

RISC-V in the News

Business

RISC-V grows globally as an alternative to Arm and its license fees

DEAN TAKAHASHI @DEANTAK DECEMBER 11, 2019 5:30 AM

RISC-V Gaining Traction

429 Shares | 162 | 48 | 110

Experts at the Table: Extensible instruction-set architecture is drawing attention from across the industry and supply chain.

JULY 30TH, 2020 - BY: ED SPERLING

DESIGNLINES | MCU DESIGNLINE

Can Arm Survive RISC-V Challenge?

By Nitin Dahad 02.13.2019 4

Technologies.org

A Whole New World: Apple, Qualcomm, ARM and RISC-V

July 25, 2019 By Technologies.org — Leave a Comment

Home › Computer › News

Alibaba XT910 RISC-V Core Faster Than Kirin 970 SoC; Threat To ARM?

By Vivek | Published: Wednesday, August 19, 2020, 12:18 [IST]

Samsung to Use SiFive RISC-V Cores for SoCs, Automotive, 5G Applications

by Anton Shilov on December 12, 2019 11:00 AM EST

DESIGNLINES | MCU DESIGNLINE

RISC-V on the Verge of Broad Adoption

By Rich Quinnell 02.14.2019 1

TECHNOLOGIES > EMBEDDED REVOLUTION

SiFive Lands Another $60 Million in Funding to Challenge Arm

Domain Specific Accelerators Will Drive Vector Processing on RISC-V

By Charlie Cheng, Andes Technology (May 26, 2020)

Open-source computing

A new blueprint for microprocessors challenges the industry's giants

RISC-V is an alternative to proprietary designs

DESIGNLINES | MCU DESIGNLINE

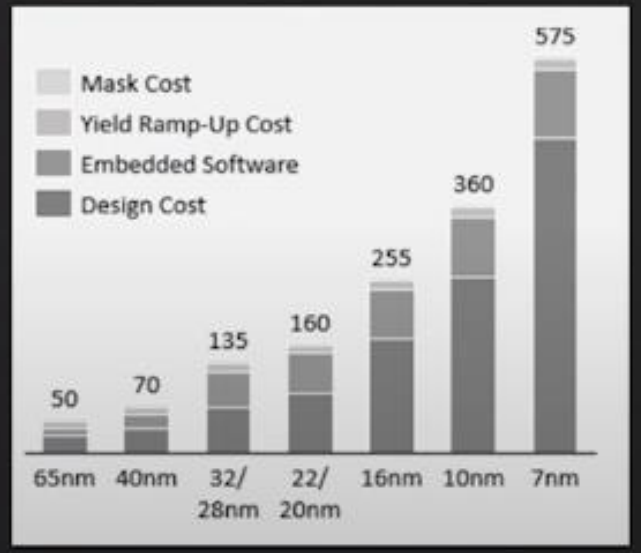
Arm China CEO: Good or Gone?

By EE Times China 06.10.2020 0

Owning Chip Development – 3 Problems

Cost

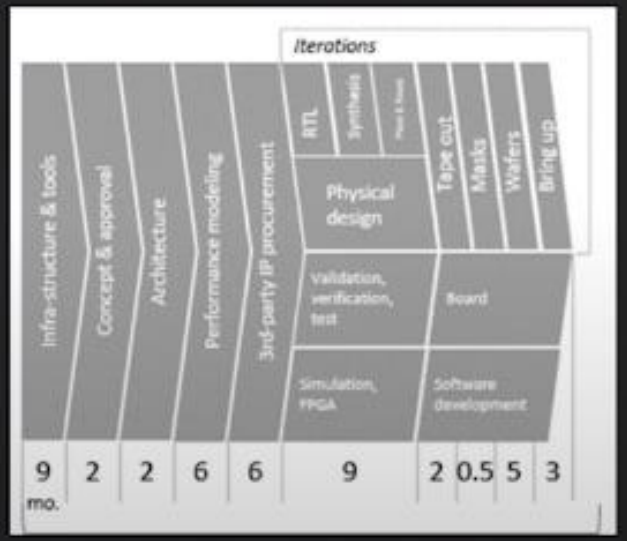
Chip Development Too Costly



$500M+ for 7nm

Time

Development Cycle Too Long



2 – 4 years

Expertise

Too Many Experts Needed

14+ Disciplines

Lessons from Software

- Software successful because they were able to overcome these 3 barriers: Cost, Time and Expertise
- Instagram: $1B valuation with 13 employees (total)
- Software companies have experts in 3-4 areas not 30
- They only write software at the top – the software stack is already provided (open source) by the industry

New Chip Architecture Options – already available from RISC V

- Modify the core
- Drop a floating point unit.
- Drop a system port.
- Add Vector Extensions.
- Switch between 32 and 64-bit architectures.
- For Edge chips, an extremely small microprocessor with 64-bit vectors
- End product is verified RTL - sold as IP to customer who then proceeds with his own flow for Design, Verification & Test

Why this Flexibility/Reconfigurability?

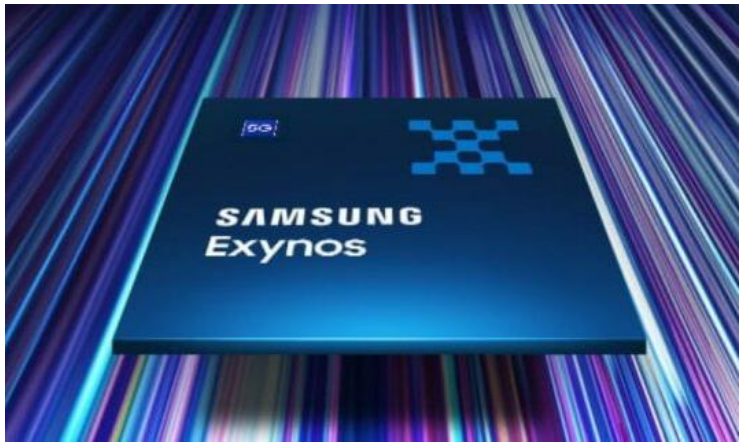
- A third of the Power consumed by an equivalent ARM core
- Much smaller chip (smaller cost and higher chip yields) – meets the needs of the Application
- Much faster chip because wire lengths , gate loads are smaller – time to market is faster due to simpler design
- The modularity of RISCV ISA enables these options – that the (much older) ARM ISA has no ability to deliver

Impact on Market

- "I predict, in about 3 years, you will definitely see RISC-V based cell phones", SiFive Chair, June 2019
- "But is that like a Nokia candy bar phone?" – John Cooley, DAC Panel MC, June 2019

Six months later: December 12, 2019 11:00 AM EST

- https://www.anandtech.com/show/15228/samsung-to-use-riscv-cores
- "Samsung disclosed the use SiFive's RISC-V cores for upcoming chips for a variety of applications. The company is joining a growing list of leading high-tech companies that have adopted the RISC-V architecture"



THE WALL STREET JOURNAL.

TECH

China's Chip-Independence Goals Helped by U.S.-Developed Tech

New approach to semiconductors, developed partly with the Pentagon's backing, could threaten Intel and Arm's dominance

The standard is winning global interest, and early users include Chinese online retail and tech giant Alibaba Group Holding Ltd. , which developed what some industry insiders consider the highest-performance RISC-V chip in production. Alibaba has said it is using that chip in its data centers to perform artificial intelligence calculations, and is selling versions of it.

Widespread adoption of RISC-V, pronounced "risk five," could open the door for China to accelerate efforts to end its dependence on Western chip technology, threaten licensing revenues for Arm and could aid the rise of new rivals to incumbent chip makers, industry executives and analysts said.

https://www.wsj.com/articles/chinas-chip-independence-goals-helped-by-u-s-developed-tech-11610375472

RISC-V boom from edge AI says Facebook's chief AI scientist

October 15, 2020 //By Nick Flaherty

A boom in low cost edge AI chips using the RISC-V technology is coming says Facebook's chief AI scientist Yann LeCun

The move to RISC-V for running neural networks for edge AI applications is accelerated by the proposed takeover of ARM by Nvidia, says Yann LeCun, chief AI scientist at Facebook speaking at the Innovation Day of French research lab CEA-Leti.

https://www.eenewseurope.com/news/risc-v-boom-edge-ai-says-facebooks-chief-ai-scientist

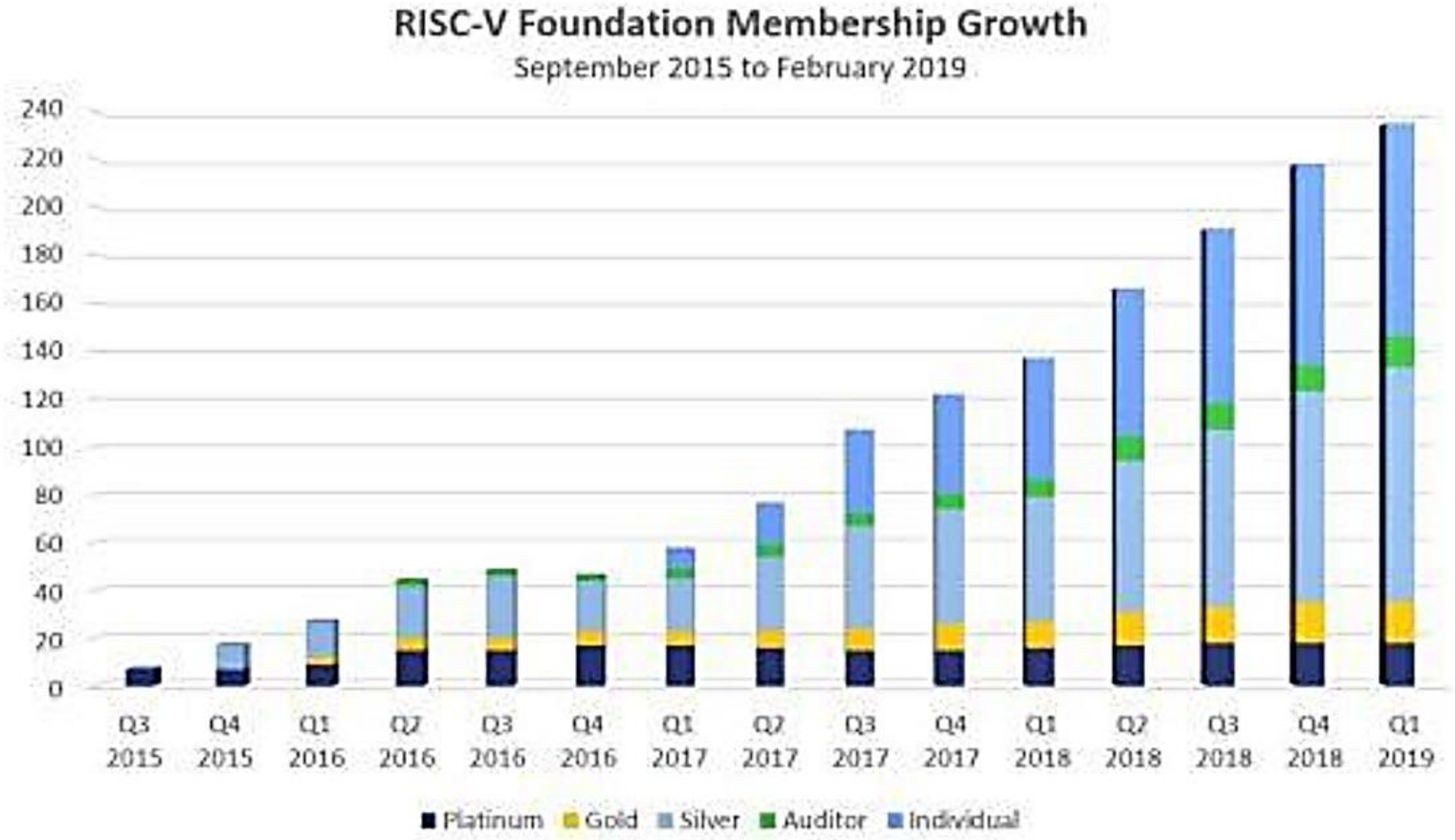
Opportunities & Markets enabled by RISC V

- Emergence of *'Open Source'* processor cores for the first time with RISC-V –*lowers cost, barriers to market entry and enables flexibility of SoC design* by way of customizing ISA extensions - unavailable previously
- Emergence of markets for *'Internet of Things'* demanding (1) ultra-low cost (2) interoperability and (3) Upwards of 20B - 50B devices/year
- Emergence of an unprecedented need for energy efficiency to support *performance demanding AI workloads* in the wireless space

Why RISC V?

- **Open Source**, no licensing fees, ultra-low cost enabler for pervasive sub-$10 IoT systems – 10s of billions of these
- Previously, system developers either **must choose** a *proprietary CPU architecture*, often optimized to a specific application space, ***or*** *design their own CPU architecture*.
- With their own CPU design, *developers give up the extensive support ecosystems that established CPUs have developed.*
- the **RISC-V Foundation** (`riscv.org`) maintains and drives community development & debug of the modular, open source, RISC-V processor instruction set architecture (ISA)
- Aims to meet application needs spanning *embedded systems to server farms* and beyond – *without requiring $ millions in licensing fees that ARM, for example, charges for the use of its proprietary ISA*
- The *base instructions are frozen* and optional extensions which have been approved (by the foundation) are also frozen. Because of the stability of the ISA, software development can confidently be applied to RISC-V

0 → 500+ in 5 years



Anybody *not* here?

Trends in Semiconductor Technology

- If an ISA is to prevail, it must be designed to survive changes in underlying component technology
- To plan for the evolution of the computer, the architect must be aware of changes in implementation technology
- 5 implementation technologies that change rapidly are critical:
  - *Integrated Circuits:* Transistor counts increased 40%-55% per year (Moore's Law) Moore's Law is no more – with transistor count now doubling ever 20 years instead.
  - *DRAM:* Growth of DRAM densities has slowed from quadrupling every 3 years to doubling in every 5 years. It does not appear 32Gb DRAM is likely to happen.
  - *Flash memory*: standard storage in PMDs its capacity doubles (and price/Gb drops) every 2 years. Data retention has declined from 10 years to 2-3 and redundancy requirements for yield have climbed to over 30% (1 redundant bit for every 3 bits in chip)
  - *Magnetic disk*: from its high of doubling every year, disk improvement has slowed to 5% per year.

Scaling of Transistor Performance & Wires

- While *Transistor performance improves linearly* with scaling geometries, *transistor area shrinks quadratically*
- This higher rate of improvement in density (area drops by 50% each technology node/generation) *enabled architects to scale processors from 4-bits to 8, 16, 32 and 64 bits*
- More recently, density improvements *enabled multiple processors on chip, SIMD units and other innovations* in speculative execution and caches
- Wire delay is a product of its resistance and its capacitance – which increases considering *all dimensions of a wire scale with transistor dimensions except wire length* – making chip performance, power delivery and clock speed *increasingly limited by on-chip wires*

Trends in Power & Energy in Integrated Circuits

- **Energy is the biggest challenge facing computer designers for all classes of computers**

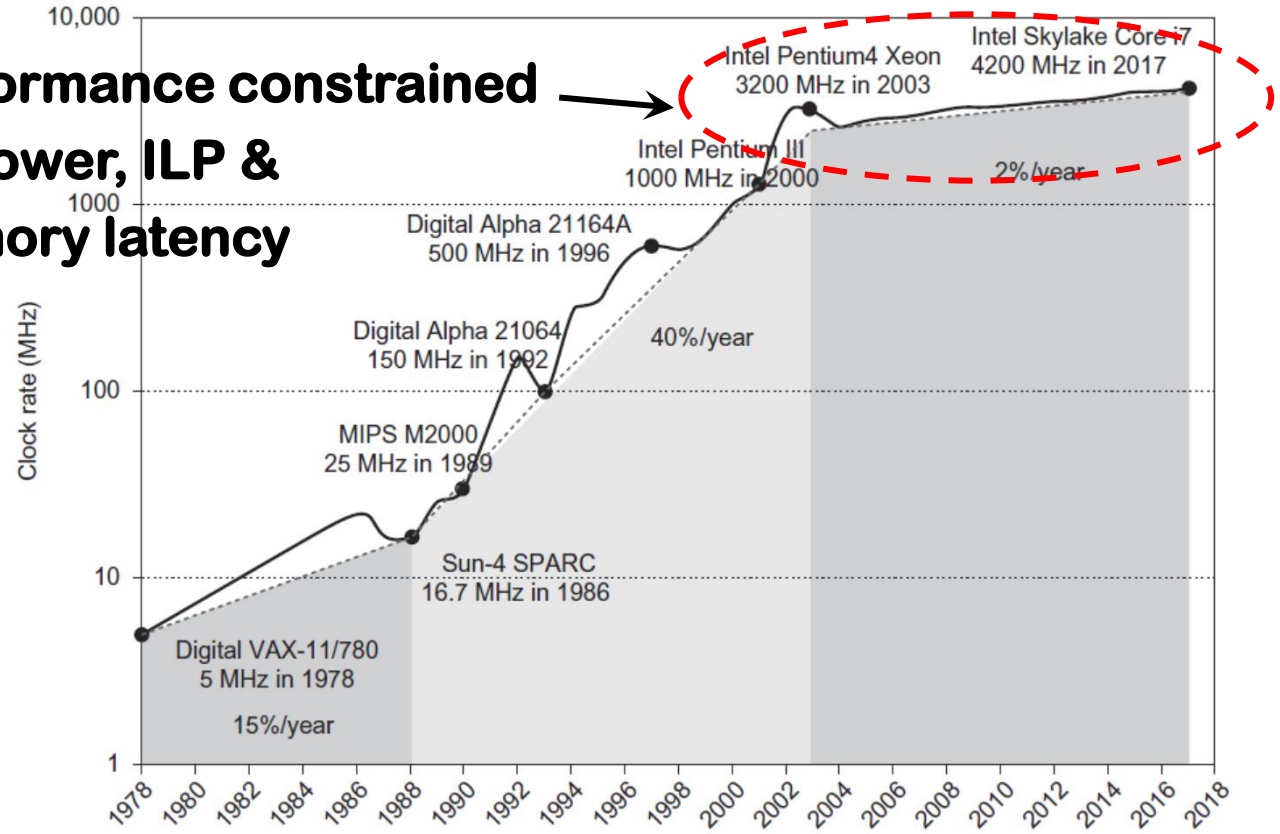
3 concerns:

- Meeting *maximum power demand*: If a processor draws more power than can be delivered to it, voltage in power grid of chip will drop - potentially causing processor circuits to fail.
- *Thermal Design Power* (TDP) is the **sustained power dissipation** from the processor – determines the cooling requirement. TDP is neither peak power (which is 1.5x higher) nor is it average power (which is lower)
- *Power delivery of a system* is designed to **exceed TDP (to meet Peak Power demand)** and the Cooling system is designed to exceed TDP as well.
- Failure to provide adequate cooling will raise junction temperature of MOS devices resulting in device failure and possibly damage
- *Energy & Energy efficiency*: **Best metrics** for all classes of computers – the electricity bill of a WSC and the battery lifetime of a PMD are both determined by energy consumed

System	Chip	TDP	Idle power	Busy power
General-purpose	Haswell E5-2699 v3	504 W	159 W	455 W
Graphics processor	NVIDIA K80	1838 W	357 W	991 W
Custom ASIC	TPU	861 W	290 W	384 W

Energy & Power within a Microprocessor

- The **Power Wall**
  - We *cannot lower voltage* further (see Slides on Dennard Scaling)
  - We *cannot remove more heat* (see next slide on why)
- How else can we improve performance?
- In CMOS IC technology

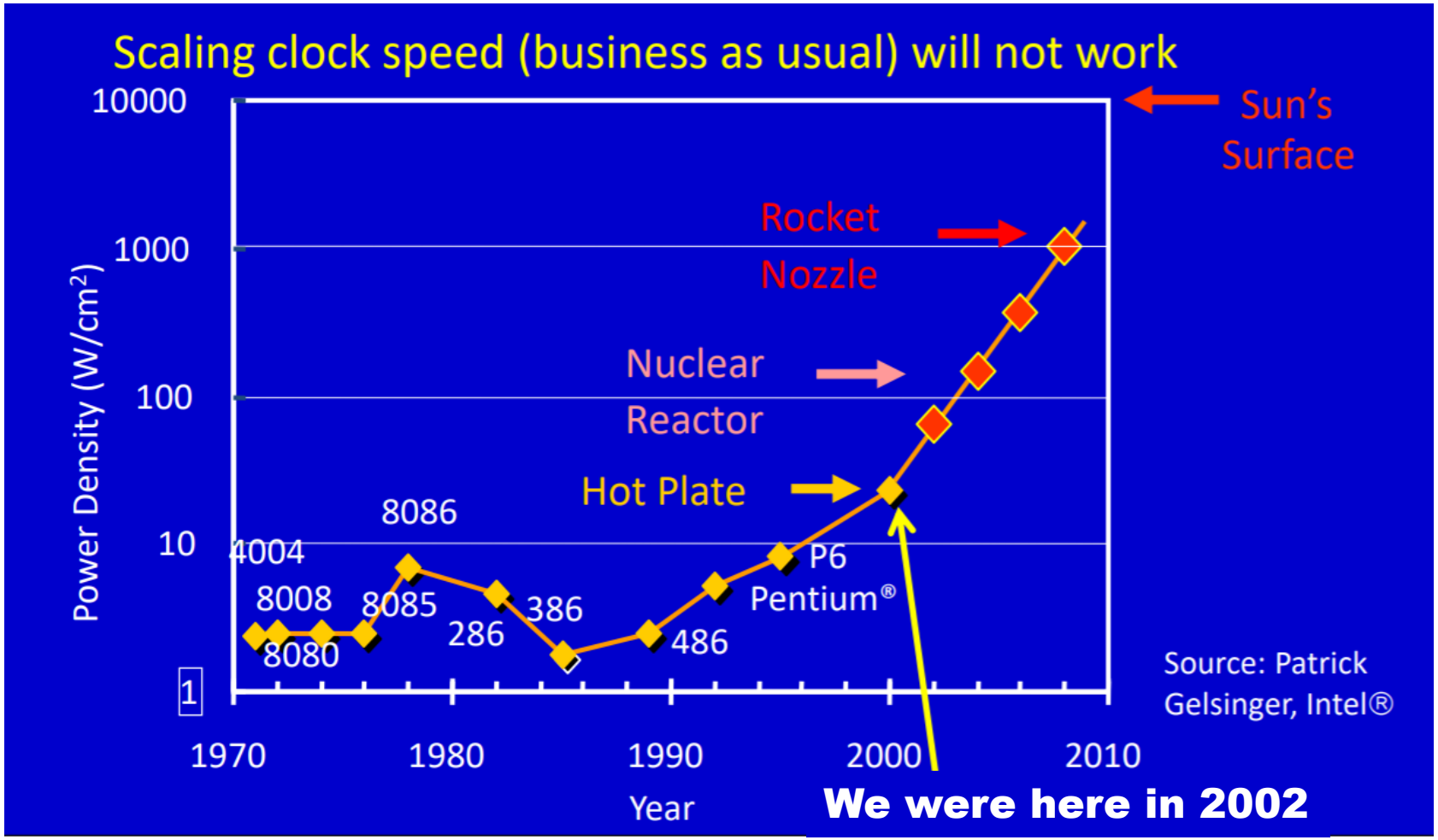


$$\text{Power} = \text{Capacitive load} \times \text{Voltage}^2 \times \text{Frequency}$$

×30 | 5V → 1V | ×1000

Power Density limits on Microprocessor Scaling

- Dennard scaling ended 2004
- Voltages and currents *could not scale with geometry anymore*
- Under constant voltage scaling (pursued by industry post 2004), ***power density increases as the cube of scaling factor***.
- Heat removal with more sophisticated and expensive packaging possible, but not for much longer, as red diamonds in graph show



Reducing Power

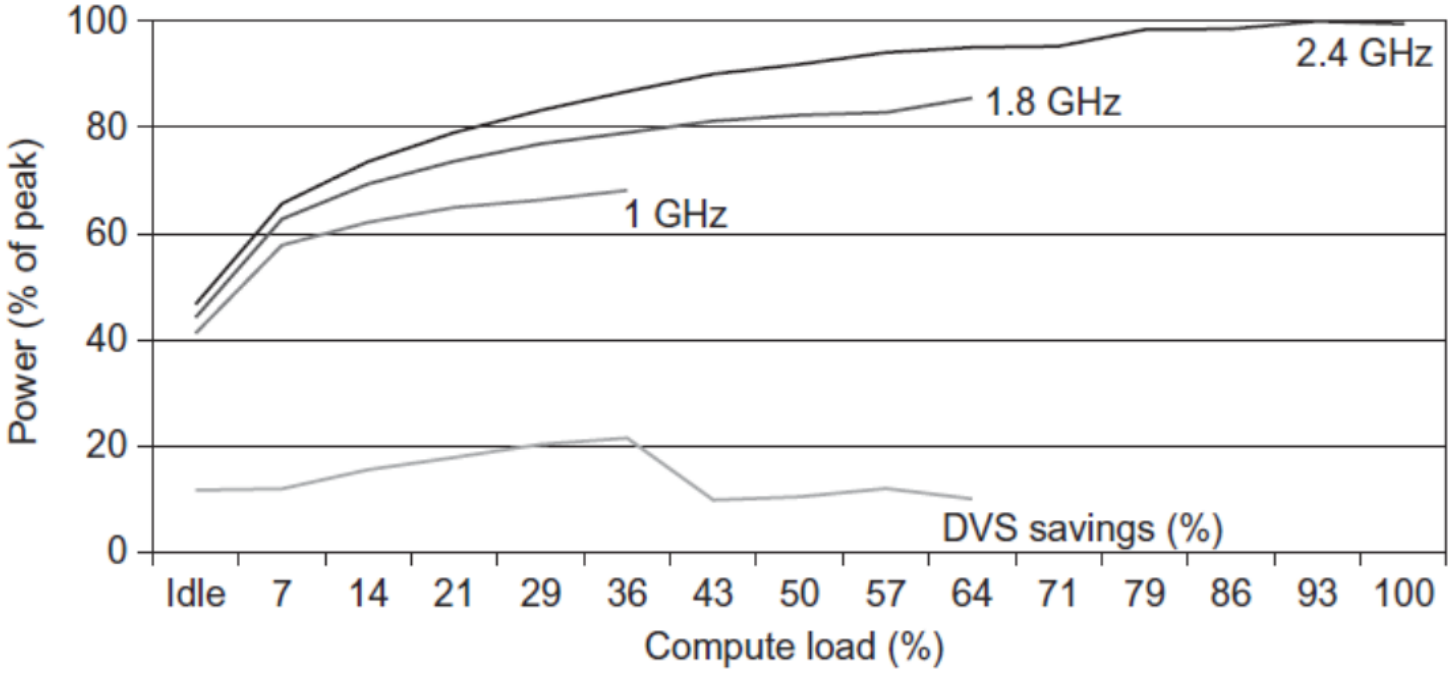
- Suppose a new CPU has
  - 85% of capacitive load of old CPU
  - 15% voltage and 15% frequency reduction

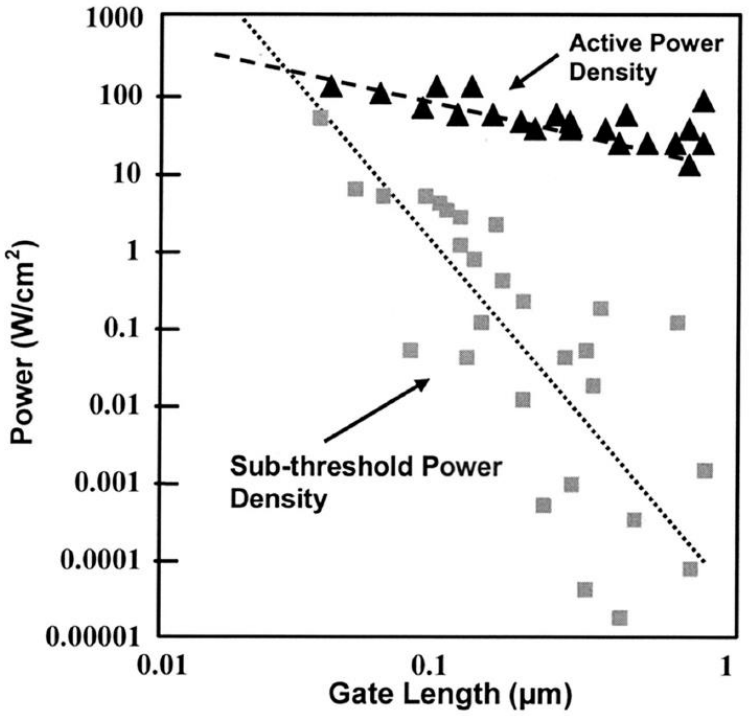
$$\frac{P_{new}}{P_{old}} = \frac{C_{old} \times 0.85 \times (V_{old} \times 0.85)^2 \times F_{old} \times 0.85}{C_{old} \times V_{old}^{\ 2} \times F_{old}} = 0.85^4 = 0.52$$

- If Voltage and Frequency do not scale (prev. slide) what happens?
- $\frac{P_{new}}{P_{old}} = \frac{C_{old} \times 0.85 \times (V_{old})^2 \times F_{old}}{C_{old} \times (V_{old})^2 \times F_{old}} = 0.85$
- How else can we improve performance?

Techniques to Lower Processor Power

- *Clock Gating:* Turn off clock for inactive modules by gating clocks to their Flip Flops
- *Dynamic Voltage Frequency Scaling:* Periods of low activity do not require high speed clocks or high voltages. Use higher Voltage and/or Frequency only when workloads require
- *Power Gating*: Static (leakage) power increases rapidly as MOS threshold voltages scale to recover circuit speed loss at lower operating voltages. Power gating lowers leakage currents 10X

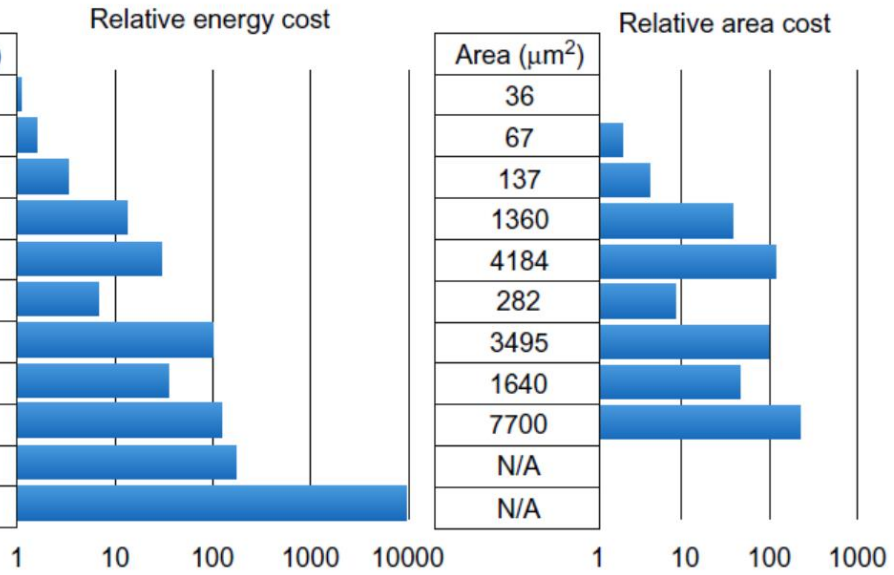




Shift in Architecture because of Energy Limits

- **Energy & Area costs of building blocks reveal:**
  - 32b FP Add **uses 30X energy** of 8b integer Add, and **60X area** as well
  - 32b DRAM access uses 20000X energy as 8b Add [*Graphcore*: recalculate data rather than access it from memory]
  - Small SRAM 125X more energy efficient than DRAM
  - DSA options to save energy:
    - *Graphcore*: recalculate data rather than access from memory
    - *Nvidia GPUs*: Reuse data locally in RF arrays with optimized Row Stationary Data flows to minimize off-chip DRAM access

Operation:	Energy (pJ)	Area (μm^2)
8b Add	0.03	36
16b Add	0.05	67
32b Add	0.1	137
16b FB Add	0.4	1360
32b FB Add	0.9	4184
8b Mult	0.2	282
32b Mult	3.1	3495
16b FB Mult	1.1	1640
32b FB Mult	3.7	7700
32b SRAM Read (8KB)	5	N/A
32b DRAM Read	640	N/A



Dependability

- Module reliability is a measure of time to failure from a reference initial instant
- MTTF (mean time to failure) is a reliability measure
- Reciprocal of MTTF is failure rate - FIT (Failures in Time)
- Service interruption is measured as a Mean time to Repair (MTTR)
- Mean time between failures (MTBF) is widely used and is a sum of MTTF and MTTR
- Overall failure rate of a module equals the sum of the failure rates of its components
- Module availability is ratio of MTTF to MTBF [MTTF/(MTTF + MTTR)]

Example

- Failure rate of component (failures in time) = 1/MTTF of component
- Failure rate of system = sum of failure rates of components
- Using values in given Table,
- Failure rate of Computer

Component	Mean time to failure (in units of 10^6 hours)	Number of these
CPU	8	1
Memory stick	6	3
Hard drive	1	2
Power supply	4	1

$= (10^{-6}) x [(1/8) + 3x(1/6) + 2(1) + (1/4)]$

$= (10^{-6}) x [0.125 + 0.5 + 2 + 0.25]$

$= 2.875 \times 10^{-6}$ per hour

$= 2.875 \times 10^{-6} \times 24 \times 365$ per year

$= 2.5185 \times 10^{-2}$ per year

MTTF of computer = 1/(failure rate) = 39.71 years

- Failure rate of cluster of 144 computers

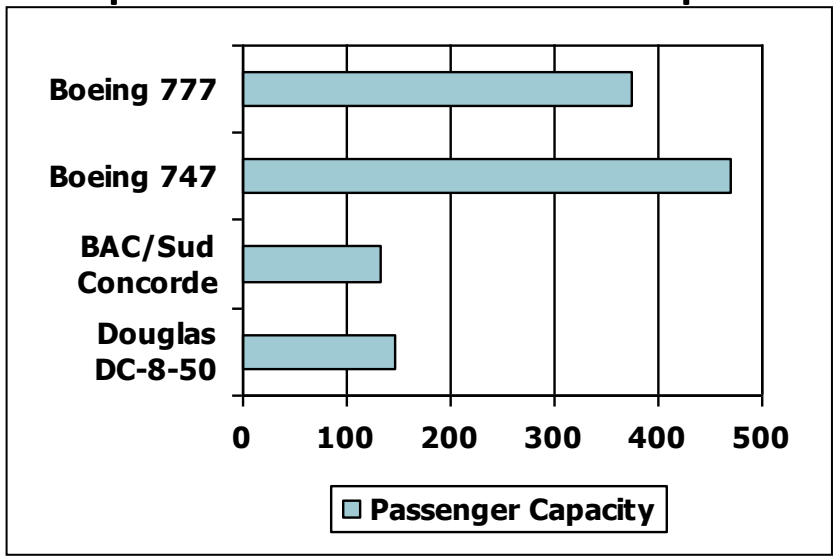
$= 2.5185 \times 10^{-2} \times 144$

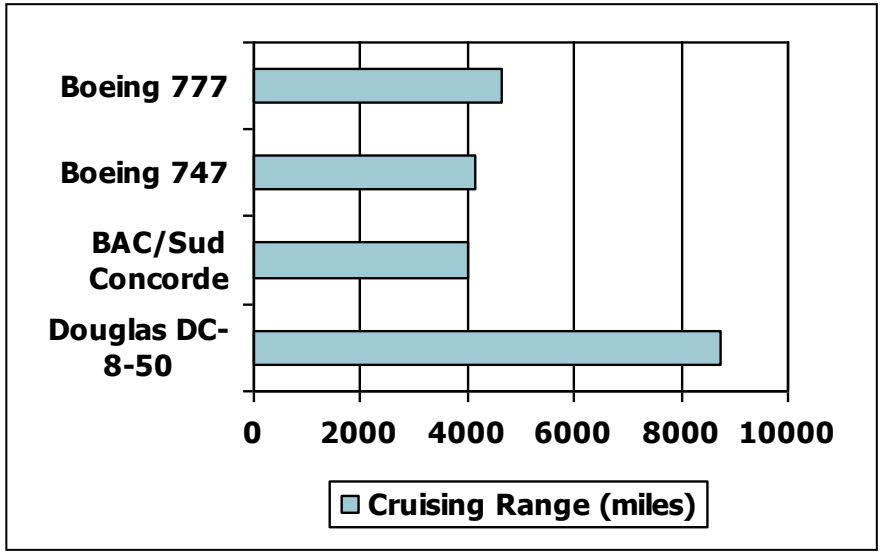
= 3.63 computers per year

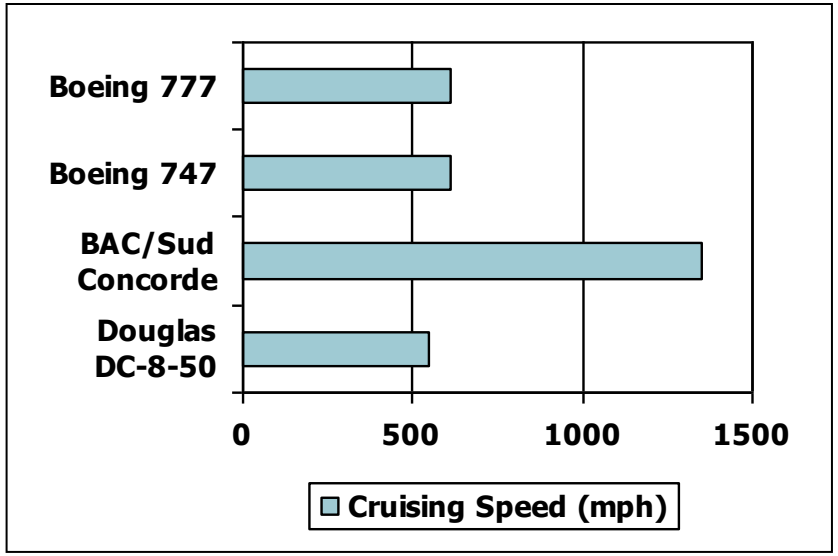
Only 140 computers working at end of first year

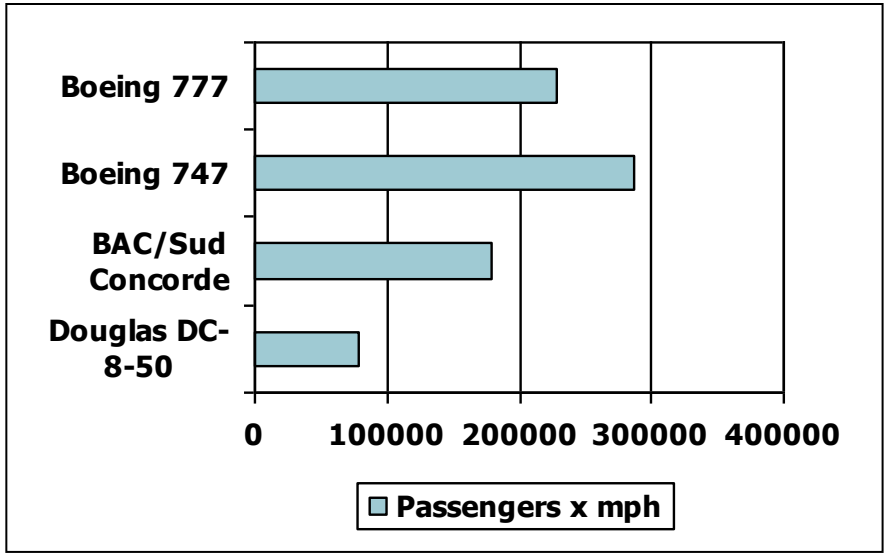
Defining Performance

- Which airplane has the best performance?









Relative Performance

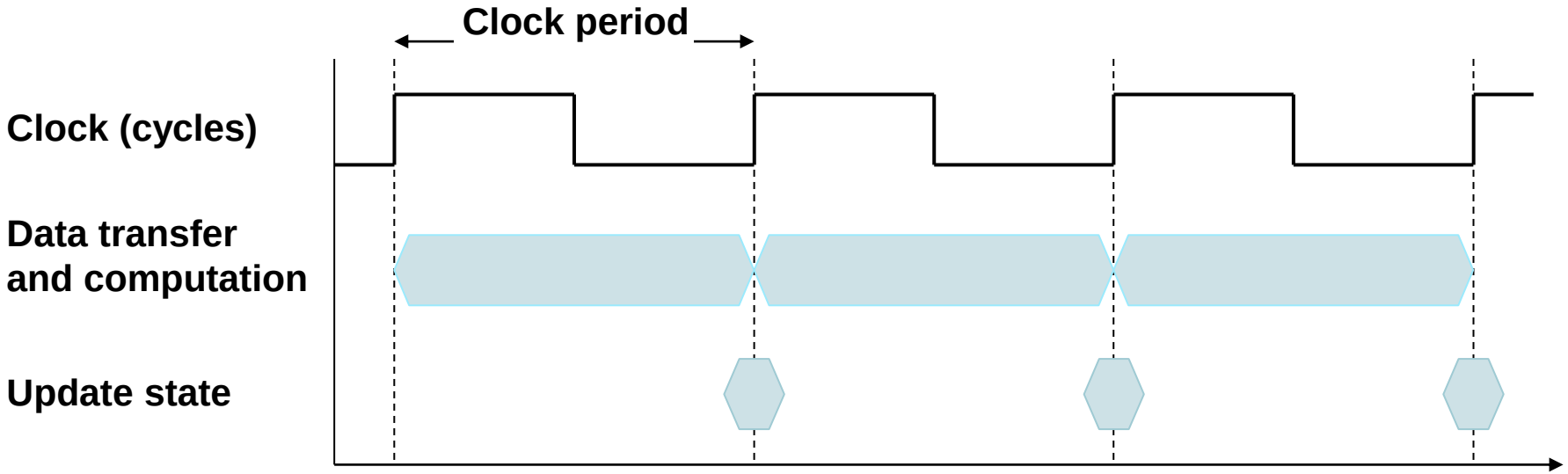
- Define Performance = 1/Execution Time
- "X is *n* time faster than Y"

$$\text{Performance}_X / \text{Performance}_Y$$
$$= \text{Execution time}_Y / \text{Execution time}_X = n$$

- Example: time taken to run a program
  - 10s on A, 15s on B
  - Execution Time_B / Execution Time_A
    = 15s / 10s = 1.5
  - So A is 1.5 times faster than B

CPU Clocking

- Operation of digital hardware governed by a constant-rate clock



- Clock period: duration of a clock cycle
  - e.g., 250ps = 0.25ns = 250×10^{-12}s
- Clock frequency (rate): cycles per second
  - e.g., 4.0GHz = 4000MHz = 4.0×10^{9}Hz

CPU Time

$$\text{CPU Time} = \text{CPU Clock Cycles} \times \text{Clock Cycle Time}$$

$$= \frac{\text{CPU Clock Cycles}}{\text{Clock Rate}}$$

- Performance improved by
  - Reducing number of clock cycles
  - Increasing clock rate
  - Hardware designer must often trade off clock rate against cycle count

CPU Time Example

- Computer A: 2GHz clock, 10s CPU time
- Designing Computer B
  - Aim for 6s CPU time
  - Can do faster clock, but causes 1.2 × clock cycles
- How fast must Computer B clock be?

$$\text{Clock Rate}_B = \frac{\text{Clock Cycles}_B}{\text{CPU Time}_B} = \frac{1.2 \times \text{Clock Cycles}_A}{6\text{s}}$$

$$\text{Clock Cycles}_A = \text{CPU Time}_A \times \text{Clock Rate}_A$$

$$= 10\text{s} \times 2\text{GHz} = 20 \times 10^9$$

$$\text{Clock Rate}_B = \frac{1.2 \times 20 \times 10^9}{6\text{s}} = \frac{24 \times 10^9}{6\text{s}} = 4\text{GHz}$$

Instruction Count and CPI

$$\text{Clock Cycles} = \text{Instruction Count} \times \text{Cycles per Instruction}$$

$$\text{CPU Time} = \text{Instruction Count} \times \text{CPI} \times \text{Clock Cycle Time}$$

$$= \frac{\text{Instruction Count} \times \text{CPI}}{\text{Clock Rate}}$$

- Instruction Count for a program
  - Determined by program, ISA and compiler
- Average cycles per instruction
  - Determined by CPU hardware
  - If different instructions have different CPI
    - Average CPI affected by instruction mix

CPI Example

- Computer A: Cycle Time = 250ps, CPI = 2.0
- Computer B: Cycle Time = 500ps, CPI = 1.2
- Same number of instructions
- Which is faster, and by how much?

$$\text{CPU Time}_A = \text{Instruction Count} \times \text{CPI}_A \times \text{Cycle Time}_A$$
$$= I \times 2.0 \times 250\text{ps} = I \times 500\text{ps}$$

A is faster…

$$\text{CPU Time}_B = \text{Instruction Count} \times \text{CPI}_B \times \text{Cycle Time}_B$$
$$= I \times 1.2 \times 500\text{ps} = I \times 600\text{ps}$$

$$\frac{\text{CPU Time}_B}{\text{CPU Time}_A} = \frac{I \times 600\text{ps}}{I \times 500\text{ps}} = 1.2$$

…by this much

CPI in more detail

- If different instruction classes take different numbers of cycles

$$\text{Clock Cycles} = \sum_{i=1}^{n} (\text{CPI}_i \times \text{Instruction Count}_i)$$

- Weighted average CPI

$$\text{CPI} = \frac{\text{Clock Cycles}}{\text{Instruction Count}} = \sum_{i=1}^{n} \left(\text{CPI}_i \times \underbrace{\frac{\text{Instruction Count}_i}{\text{Instruction Count}}}_{\text{Relative frequency}} \right)$$

CPI Example

- Alternative compiled code sequences using instructions in classes A, B, C

Class	A	B	C
CPI for class	1	2	3
IC in sequence 1	2	1	2
IC in sequence 2	4	1	1

- Sequence 1: IC = 5
  - Clock Cycles
    = 2×1 + 1×2 + 2×3
    = 10
  - Avg. CPI = 10/5 = 2.0
- Sequence 2: IC = 6
  - Clock Cycles
    = 4×1 + 1×2 + 1×3
    = 9
  - Avg. CPI = 9/6 = 1.5

Performance Summary

$$\text{CPU Time} = \frac{\text{Instructions}}{\text{Program}} \times \frac{\text{Clock cycles}}{\text{Instruction}} \times \frac{\text{Seconds}}{\text{Clock cycle}}$$

- Performance depends on
  - Algorithm: affects IC, possibly CPI
  - Programming language: affects IC, CPI
  - Compiler: affects IC, CPI
  - Instruction set architecture: affects IC, CPI, T_c

Focus on the Common Case

- Design trade-offs require favouring the more frequent case over the infrequent case
- Impact of the improvement is higher if favoured case is more frequent
- Instruction Fetch and Decode used more frequently than multiplier
- If a database server has 50 storage devices for every processor storage dependability will dominate system dependability
- Adding 2 numbers we can expect overflow, but if it is rare, performance is optimized by optimizing the normal case

Example Problem 1

- Consider three different processors P1, P2, and P3 executing the **same instruction set**. P1 has a 3 GHz clock rate and a CPI of 1.5. P2 has a 2.5 GHz clock rate and a CPI of 1.0. P3 has a 4.0 GHz clock rate and has a CPI of 2.2.

  a. Which processor has the highest performance expressed in instructions per second?

  b. If the processors each execute a program in 10 seconds, find the *number of cycles* and the *number of instructions*.

  c. We are trying to *reduce the execution time by 30%,* but this leads to an *increase of 20% in the CPI*. What clock rate should we have to get this time reduction?

1a

IPS = Cycles per Second/Cycles per Instruction

A measure of throughput – or rate of doing work

- Performance of P1: 3GHz/1.5 = 2 x 10^9 Inst/sec
- Performance of P2: 2.5GHz/**1.0** = 2.5 x 10^9 Inst/sec
- Performance of P3: 4GHz/2.2 = 1.8 x 10^9 Inst/sec
- **CPI** can be as relevant to processor performance as clock frequency
- *Faster clocks may not always be a good thing* – higher power dissipation, worse reliability, worse coupling noise… and not the best rate of processing instructions!

1b

- # of cycles = Cycles per second x time (in seconds)
- Cycles of P1: 3 GHz x 10 s = 30 B cycles
- Cycles of P2: 2.5GHz x 10 s = 25 B cycles
- Cycles of P3: 4 GHz x 10s = 40 B cycles

Assuming a metric of wall clock time to execute a given benchmark program,

P3 consumed more clock cycles – more power to do the same work

P2 consumed the least number of clock cycles to do the same work

Lower CPI translates into higher productivity and higher energy efficiency as a result

1b contd..

- # of Instructions = Cycles / CPI
- # of Instructions of P1: 30 B cycles/1.5 Cycles per Instruction = 20B
- # of Instructions of P2: 25 B cycles/1 Cycles per Instruction = 25B
- # of Instructions of P3: 40 B cycles/2.2 Cycles per Instruction = 18.18B

1c

- *Lower execution time* trades off with *higher CPI & higher* F_{CLK}
- Assuming 30% reduction in execution time requires 20% higher CPI

Instructions x CPI_new / ET_new = Fclk_new

Fclk_new P1 = 20 B x 1.8 / 7s = 5.14 GHz

Fclk_new P2 = 25 B x 1.2 / 7s = 4.28 GHz

Fclk_new P1 = 18.18 B x 2.6 / 7s = 6.75 GHz

High CPI processors require even higher Clock rates to get the same % improvement in execution time

Example Problem 2

Compilers can have a profound impact on the performance of an application. Assume that for a program, compiler A results in a dynamic instruction count of 1.0E9 and has an execution time of 1.1 s, while compiler B results in a dynamic instruction count of 1.2E9 and an execution time of 1.5 s.

a. Find the average CPI for each program given that the processor has a clock cycle time of 1 ns.

b. Assume the compiled programs run on two different processors. If the execution times on the two processors are the same, how much faster is the clock of the processor running compiler A's code versus the clock of the processor running compiler B's code?

c. A new compiler is developed that uses only 6.0E8 instructions and has an average CPI of 1.1. What is the speedup of using this new compiler versus using compiler A or B on the original processor?

2a

- CPI = ETime x Fclk / Instr Count
- Compiler A CPI = 1.1s x 1GHz / 1 B = 1.1
- Compiler B CPI = 1.5s x 1GHz / 1.2 B = 1.25

On a given machine with **a given clock frequency**, different compilers that generate machine instructions using the **same instruction set architecture** *can differentiate* in achieving **lower CPI** and **higher performance** as a result

2b

- Assume the **processors are different** and **Execution times are now the same**
- How much faster is clock running B's code Vs clock running A's code?

Fclk_B / F_clk_A = [IC_B x CPI_B] / [IC_A x CPI_A]

= [1.2 B x 1.25] / [1 B x 1.1]

= 1.36

2c

- New compiler uses only 0.6 B instructions, CPI = 1.1
- Speedup Versus A or B on the original processor:
- Instr Count x CPI = #cycles in original processor
- TA / Tnew = Speedup Vs A: 0.66 B cycles Vs 1 B x 1.1 or 1.1 B cycles

= 1.1 / 0.6 = 1.67

- TB/Tnew = 0.66 B cycles Vs 1.2 B x 1.25

= 1.5 / 0.66 = 2.27

Example Problem 3

Consider two **different implementations** of the **same instruction set architecture**. The instructions can be divided into four classes according to their CPI (classes A, B, C, and D). P1 with a clock rate of 2.5 GHz and CPIs of 1, 2, 3, and 3, and P2 with a clock rate of 3 GHz and CPIs of 2, 2, 2, and 2.

a. What is the global CPI for each implementation?

b. Given a program with a dynamic instruction count of 1.0E6 instructions divided into classes as follows: 10% class A, 20% class B, 50% class C, and 20% class D, **which is faster: P1 or P2?**

c. Find the clock cycles required in both cases.

3a,b

INSTR CLASS →	A: 10%	B: 20%	C: 50%	D: 20%
P1 CPI	1	2	3	3
P2 CPI	2	2	2	2

- CPI of P1 = 0.1 X 1 + 0.2 X 2 + 0.5 X 3 + 0.2 X 3 = 2.6
- CPI of P2 = 0.1 X 2 + 0.2 X 2 + 0.5 X 2 + 0.2 X 2 = 2.0

ET_P1 = [CPI / FCLK] X IC = [2.6 / 2.5G] X 1M = 1.04 ms

ET_P2 = [CPI / FCLK] X IC = [2.0 / 3G] X 1M = 0.66 ms

P2 is faster!

3c

- Clock cycles required = CPI x IC
- P1: 2.6 x 1M = 2.6M cycles
- P2: 2.0 x 1M = 2.0M cycles

Example Problem 4

The **Pentium 4** Prescott processor, released in 2004, had a clock rate of **3.6 GHz** and voltage of 1.25 V. Assume that, on average, it consumed **10 W of static power and 90 W of dynamic power**. The **Core i5** Ivy Bridge, released in 2012, has a clock rate of **3.4 GHz** and voltage of 0.9 V. Assume that, on average, it consumed **30 W of static power and 40 W of dynamic power**.

a. For each processor find the average capacitive loads.

b. Find the percentage of the total dissipated power comprised by static power and the ratio of static power to dynamic power for each technology.

c. If the total dissipated power is to be reduced by 10%, how much should the voltage be reduced to maintain the same leakage current? Note: power is defined as the product of voltage and current.

4a, b

- $DP = F \times \frac{1}{2} CV^2$
- $C = 2DP/[V^2F]$

P4: $2 \times 90W/[\ 1.5625 \times 3.6G\ ] = 3.2 \times 10^{-8}$ F

i5: $2 \times 40W/[0.81 \times 3.4G] = 2.9 \times 10^{-8}$ F

P4: 10W/100W = 10%

i5: 30W/70W = 42.9%

Leakage current increases as # of transistors on chip increases exponentially

Example Problem 5

Instruction	Frequency	Cycles per Instruction
ALU operations	30%	1
Load	20%	2
Store	10%	2
Branches	20%	3
Floating point operations	20%	5

a. What is the overall CPI of this machine?

b. If the CPU runs at 750MHz, what is the MIPS rating of this machine? For this question, count floating point operations in the MIPS rating.

c. Consider improving this computer's performance by enhancing the speed of the floating point instructions. What is the best possible overall speedup that we could obtain?

5a, b, c

- **CPI overall**

= 0.3 x 1 + 0.2 x 2 + 0.1 x 2 + 0.2 x 3 + 0.2 x 5

= 0.3 + 0.4 + 0.2 + 0.6 + 1.0 = 2.5

- **MIPS (millions of instructions per second)**

= Clock rate/CPI = $750 \times 10^6 / 2.5 = 3 \times 10^8$

= 300 MIPS

- **Biggest increase in CPI contributed by floating point instructions (need more cycles per instruction)**

Improvements in CPI of floating-point instruction CPI = infinite, i.e., CPI of FP instructions → 0

How much does the CPI of machine improve?

Speedup = CPI old / CPI new

CPI new =

0.3 x 1 + 0.2 x 2 + 0.1 x 2 + 0.2 x 3 + 0.2 x 0 = 1.5

Speedup = 2.5/1.5 = 1.667

Example Problem 6

Two enhancements, E1 and E2, with the following speedups are proposed for a new architecture:

Speedup1 = 10

Speedup2 = 5

Only one of the enhancements is usable at any point in time (maybe because they use some of the same hardware).

a. If E1 can be used 20% of the time and E2 can be used 10% of the time, what would be the overall speedup?

b. If the percentage of time that E1 can be used decreased to 15%, what percentage of the time would the use of E2 have to be to get the same overall speedup as in part (a)?

c. Suppose we are free to choose between E1 or E2, whenever we want (the percentages of time for using E1 or E2 can be varied as desired, but in total cannot be more than 100% of the time). What would be the maximum achievable overall speedup?

6 a, b, c

a. Speedup = Te old / Te new

Te / [20% (Te/10) + 10% (Te/5) + 70% x (Te/Te)]

= 1 / [0.02 + 0.02 + 0.7] = 1/0.74 = 1.35

b. 1/[0.15/10 + x/5 + (0.85 -x)] = 1 / [0.74]

[0.15/10 + x/5 + (0.85 -x) = 0.74

x = 0.125 / 0.8 = 0.15625

Enhancement 2 would need to increase its percentage time from 10% to 15.625% to make up for a decrease in time of Enhancement 1 from 20% to 15%

c. speedup = Te / [100% x (Te/10)] = 10

Example Problem 7

- Suppose a program (or a program task) takes 1 billion instructions to execute on a processor running at 2 GHz. Suppose also that 50% of the instructions execute in 3 clock cycles, 30% execute in 4 clock cycles, and 20% execute in 5 clock cycles. What is the execution time for the program or task?

Instruction	Frequency	Cycles per Instruction
A	50%	3
B	30%	4
C	20%	5

Problem 7

Average Cycles Per Instruction (CPI) of the Program

= 0.5 x 3 + 0.3 x 4 + 0.2 x 5 = **3.7**

1 billion instructions x CPI = number of cycles required by Program = 3.7×10^9

at 2 GHz, *one clock cycle* consumes = 1 / [2×10^9] seconds or 0.5×10^{-9} seconds or 0.5 nanoseconds

So, 3.7×10^9 cycles consumes $3.7 \times 10^9 \times 0.5 \times 10^{-9}$ seconds

= 3.7 x 0.5 = 1.85 seconds

Example Problem 8

- Suppose the processor in the previous example is redesigned so that all instructions that initially executed in 5 cycles now execute in 4 cycles. Due to changes in the circuitry, the clock rate has to be decreased from 2.0 GHz to 1.9 GHz. What is the overall percentage improvement?

Instruction	Frequency	Cycles per Instruction
A	50%	3
B	30%	4
C	20%	4

P8

Now, the Average Cycles Per Instruction (CPI) of the Program

$= 0.5 \times 3 + 0.3 \times 4 + 0.2 \times 4 = 3.5$

So,

1 billion instructions x CPI = number of cycles required by Program = 3.5×10^9

at 1.9 GHz,

one clock cycle consumes = $1 / [\ 1.9 \times 10^9]$ seconds or 0.526315×10^{-9} seconds

So,

3.5×10^9 cycles consumes $3.5 \times 10^9 \times 0.526315 \times 10^{-9}$ seconds = 1.8421025 sec

so improvement is 1.85/1.8421025 = 1.0042872 or ~ 0.43% improvement