R265: Advanced Topics in Computer Architecture

Seminar 1: Trends in Computer Architecture

Simon Moore

Based on slides from Robert Mullins

Introduction

- 1 operation per second to 100 TOPS in less than 100 years
- What challenges do computer architects face?
- An advanced exploration of key areas in computer architecture:
  - State-of-the-art Processor Design (Simon Moore)
  - Memory system design (Tim Jones)
  - Reliability (Tim Jones)
  - Specification and verification (Jonathan Woodruff)
  - 2 x Hardware security (Jonathan Woodruff + Simon Moore)
  - HW accelerators and HW accelerators for ML (Tim Jones)

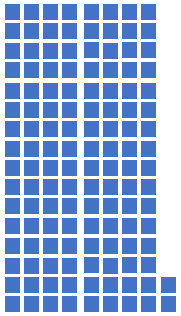
From sensors and smartphones to servers

An area optimised microcontroller core (e.g. Arm Cortex-M0)

1X

1 square represents the **area** of this core

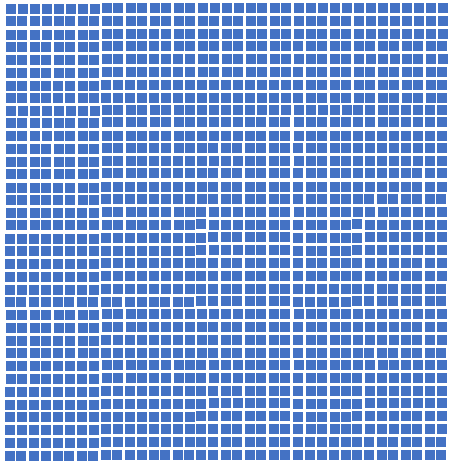
Mid-range 64-bit processor (e.g. Arm Cortex-A55). For smartphones, TVs, network infrastructure, …



130X

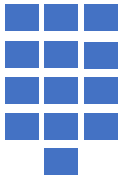
1 laptop or server class processor (e.g. A76 core with 512KB of L2 cache)

1380X

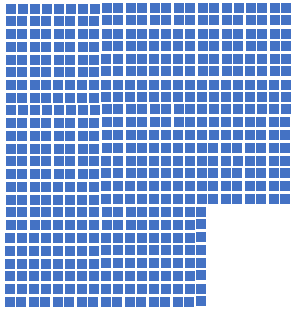


High-performance 32-bit core (e.g. Arm Cortex-M7)

Used in automotive, sensor hub and other embedded applications.



13X



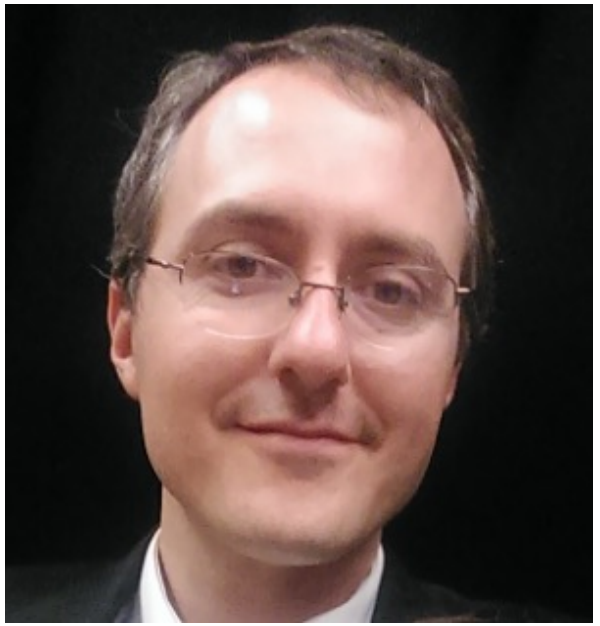
520X

High-performance processor (e.g. Arm Cortex-A73). For mobile and consumer devices.

Module convenors

Simon Moore

Timothy Jones



Jonathan Woodruff

Seminar format

- Student presentations x 3 (15 minutes each + 5 mins questions)
- Broader discussion of reading group topics (~30 minutes)
- Scene setting lecture for following week's topic (~20 minutes)

Assessment

- See https://www.cl.cam.ac.uk/teaching/2223/R265/
- Nothing for first week
- For week 2-8:
  - Each week, you will submit **either** an essay or presentation (submit via the R265 Moodle site)
  - Essays and presentations are marked and feedback will be provided, you will see an indication of the range of the mark (e.g. pass, merit, distinction).
  - The lowest mark is dropped and the remainder are scaled to give a final mark out of 100

Weekly essays

- Around 1500 words (1450-1650)
- Write an essay on two of the reading group papers
- Introduce the challenges the papers tackle, describe and clarify the important concepts, identify the key contributions the papers make.
- Critique the work, e.g. discuss cost, trade-offs and limitations, identify the strengths, weaknesses and any flaws in the work. Perhaps discuss how the ideas have been evaluated and questions that remain. Compare to and discuss relevant related work.
- What open questions and research questions remain? Do you have sound ideas for future work? Discuss relevant trends and future challenges.
- “Essays will be assessed for technical content, clarity, accurate critique, linkage of related work and sound proposals for future development.”
- Note 3-4 interesting ideas or questions to help stimulate our group discussion
- Do try to read around the subject, it is an opportunity to learn/explore.

Presentations

- 15 minutes + 5 minutes for questions
- Your presentations will be based on one of the papers from the reading group material or other work related to the week's theme
- Your chosen paper(s) must be agreed with the module convenor in advance
- You do not need to submit an essay when you will be presenting. Slides should be submitted via Moodle
- No more than 12 slides please
- Keep your slides simple (and text to a minimum)

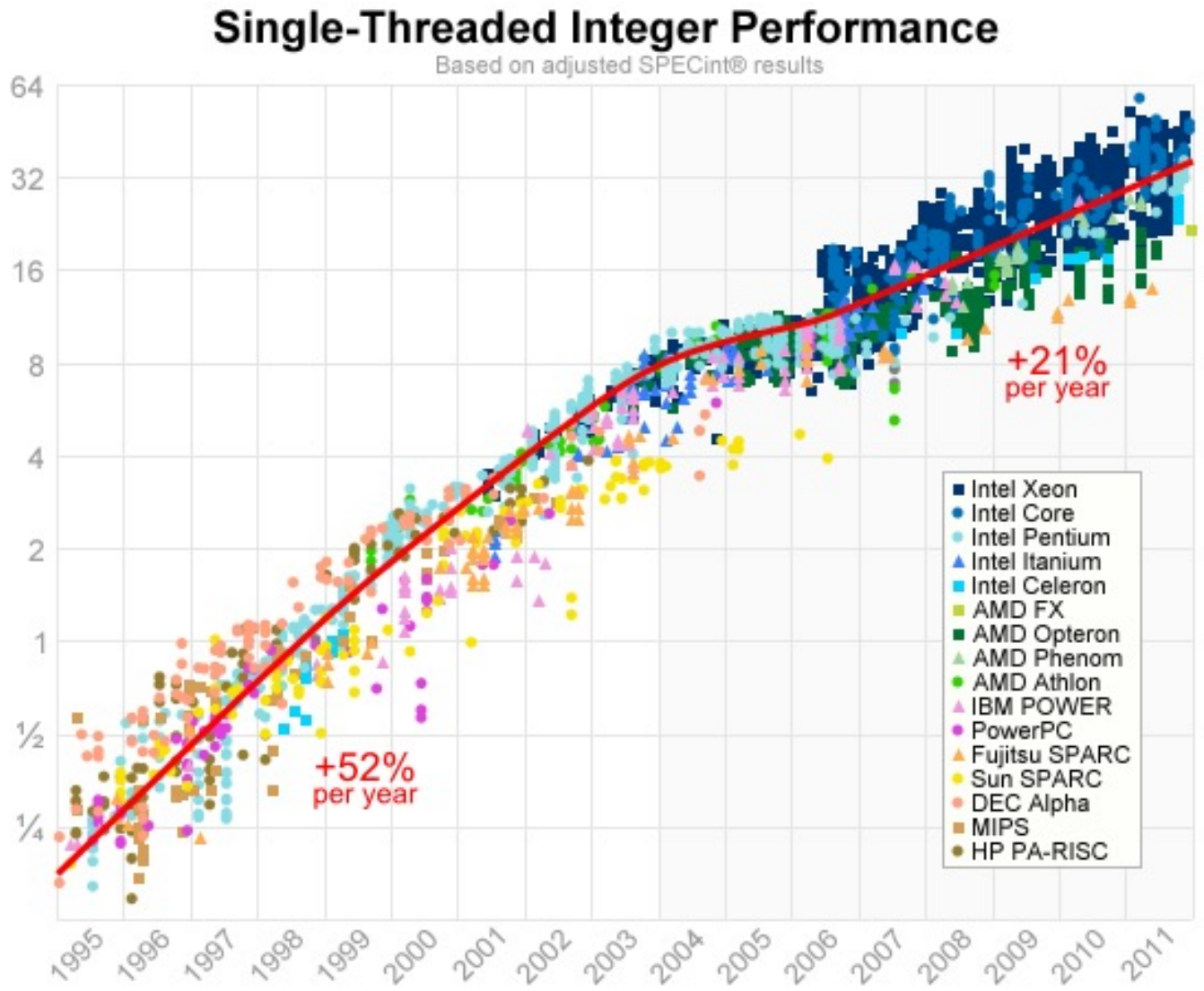
Trends in Computer Architecture

Time

Early computers	Gains from bit-level parallelism
Pipelining and superscalar issue	+ Instruction-level parallelism
Multicore / GPUs	+ Thread-level parallelism / data-level parallelism
Greater integration (large SoCs), heterogeneity and specialisation	+ Accelerator-level parallelism

Note: Memory hierarchy developments have also been significant. The memory hierarchy typically consumes a large fraction of the transistor budget.

Historical performance gains



Source: Jeff Preshing

Historical performance gains

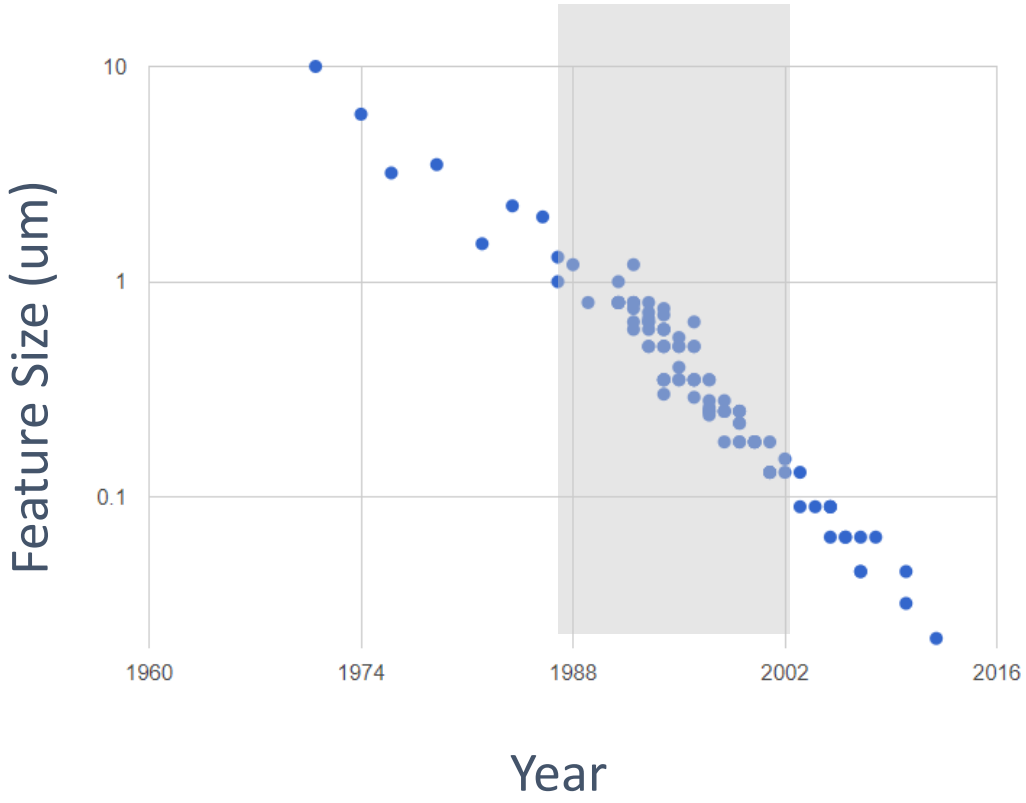
- From **1985-2002** microprocessor performance improved by ~800 times
- How was this possible?

- The “iron law” of processor performance:

Time = instructions x Clocks Per Instruction (CPI) x clock period
executed

Technology scaling: faster transistors

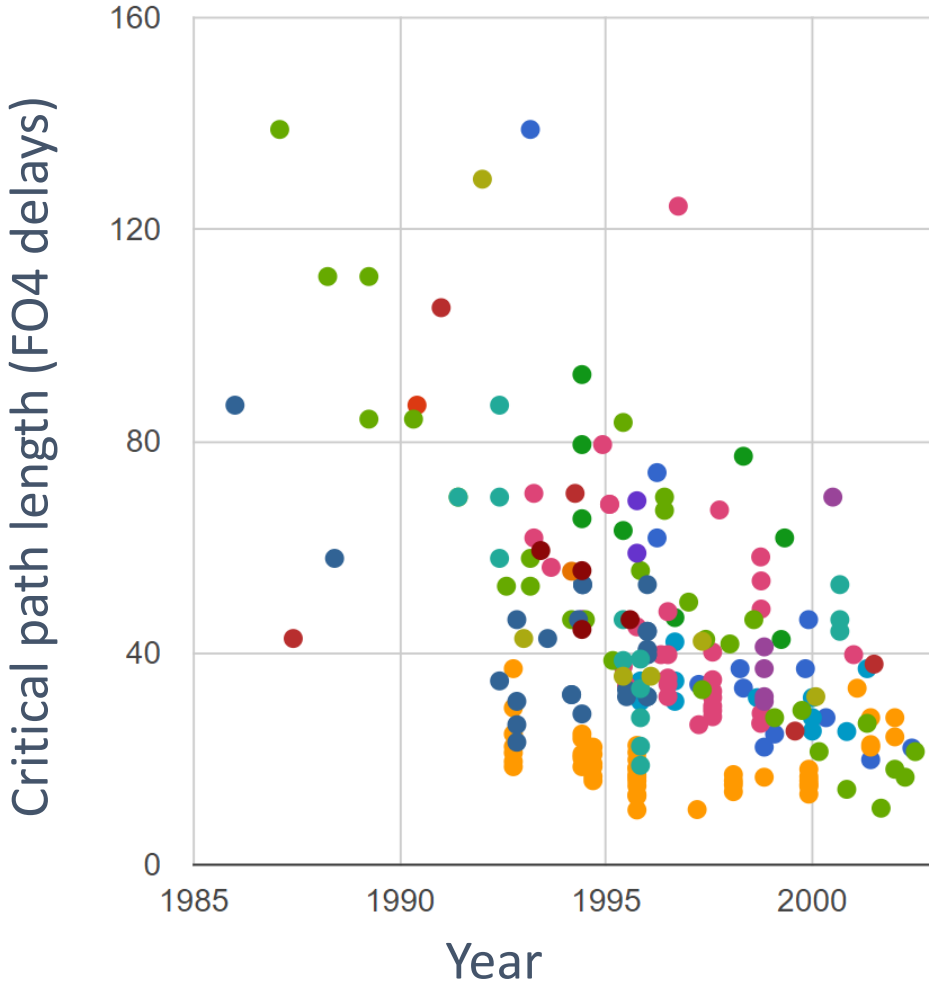
- From 1985-2002 we saw ~7 new process generations
- Scaling provides smaller and faster transistors. Performance improves ~1.4x per generation so for 7 generations we have: **~10x faster logic gates**



Source: Stanford CPU DB

A shorter critical path

- We can also try to reduce the number of gates on our critical path:
- This can be done by inserting additional registers to break complex logic into different “pipeline” stages
- Advances were also made that improved circuit-level design techniques
- The length of our critical paths reduced by ~**10x** (1985-2002)

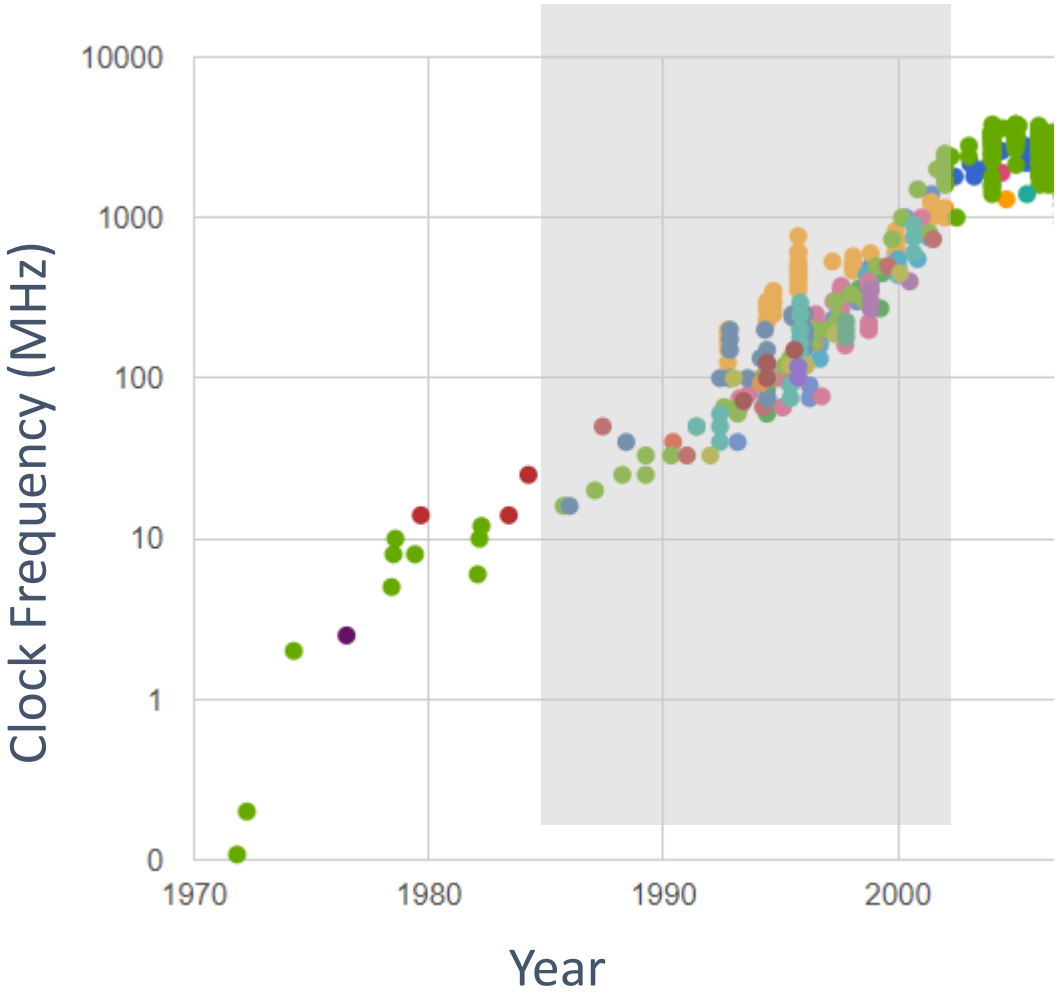


Source: Stanford CPU DB

Historical performance gains

Clock period

- Clock frequency improved quickly between 1985 and 2002:
- ~10x from faster transistors, and
- ~10x from pipelining and circuit-level advances.
- So overall, **~100X** of the total 800X gains came from reduced clock periods



Source: Stanford CPU DB

Instruction count

- Increased datapath width (e.g. 16-bit to 32-bit to 64-bit)
- Larger register files (fewer load/store instructions)
- More complex instructions?
- SIMD instructions

Clocks Per Instruction (CPI)

- Early machines were limited by transistor count. As a result they often required multiple clock cycles to execute each instruction (CPI >> 1)
- As transistor budgets improved we could aim to get closer to a CPI of 1
- This is easy if we don’t care at all about clock frequency
- Designing a high-frequency design with a good CPI is much harder. We need to keep our high-performance processor busy and avoid it stalling, which would increase our CPI. This requires many different techniques and costs transistors (area) and power.

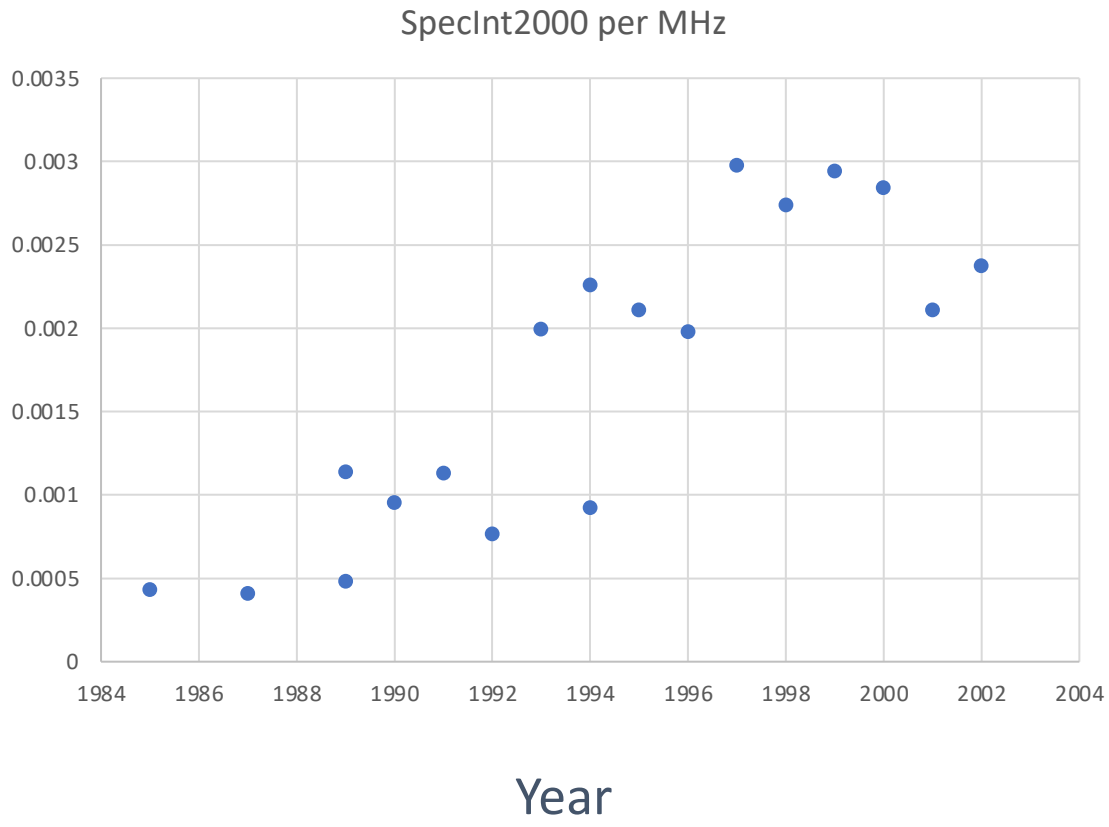
Clocks Per Instruction (CPI)

- Eventually industry was also able to fetch and execute multiple instructions per clock cycle. This reduced CPI to below 1
- When we fetch and execute multiple instructions together we often refer to **Instructions Per Cycle (IPC),** which is 1/CPI
- For instructions to be executed at the same time they must be independent.
- Again, growing transistor budgets were exploited to help find and exploit this **Instruction-Level Parallelism (ILP)**

IPC and instruction count

- Of the 800x improvement in performance (1985-2002), ~100x is from clock frequency improvements.
- The remaining gains (~8x) were from a reduction in instruction count, better compiler optimisations and improvements in IPC.

The graph to the right shows these improvements. It plots performance (SpecInt2000 benchmark performance per MHz for Intel processors against time)



Moore’s Law

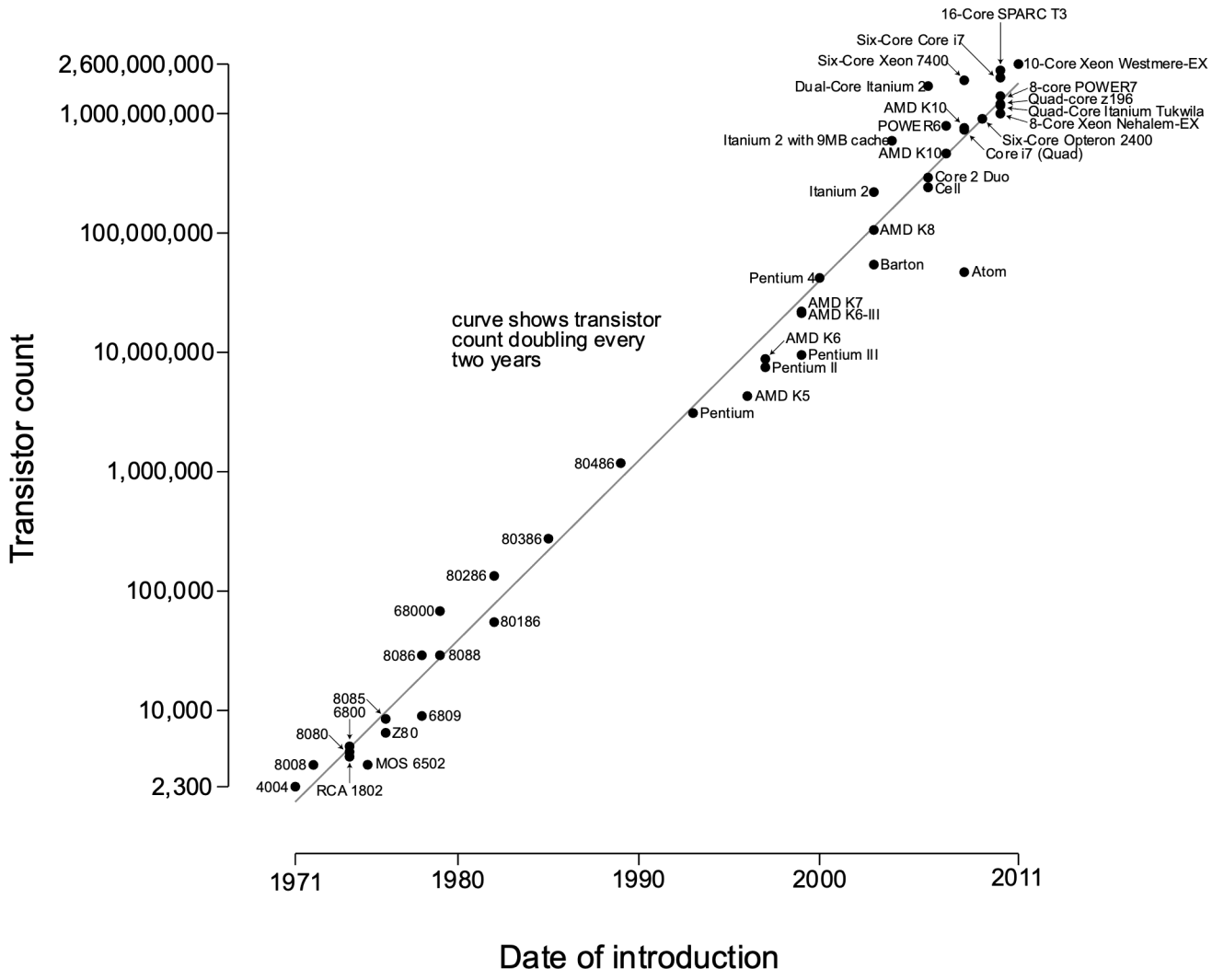
- Moore’s Law predicts that the number of transistors we can integrate onto a chip, for the same cost, doubles every 2 years

Gordon Moore and Robert Noyce at Intel in 1970

Source: IntelFreePress

Moore's Law

- Processor transistor budgets grew quickly as microarchitectures became more complex
- **1985 – Intel 386**
  275K transistors, die size =43mm^2
- **2002 – Intel Pentium 4**
  42M transistors, die size = 217mm^2



Limits to single core performance

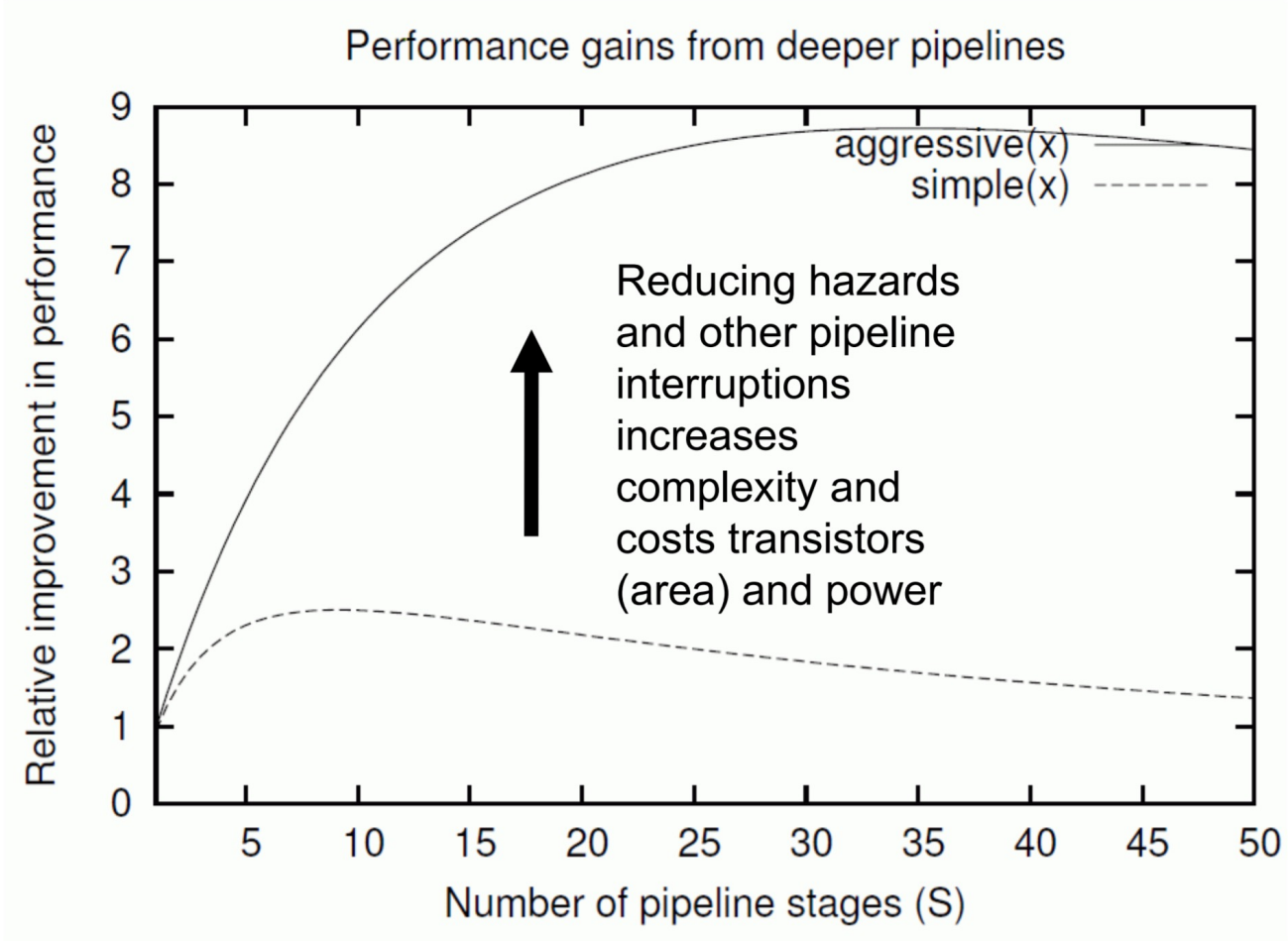
- **Limits to pipelining**
- Cost of interruptions grow, e.g. impact of cache misses and mispredicted branches
- Ultimately, some components are difficult or expensive to pipeline
- There are also practical limits to distributing very high-frequency clocks, registers represent a finite delay and we may struggle to balance logic between pipeline stages
- **Limits of Instruction-Level Parallelism (ILP)**
- Large amounts of ILP are very difficult to discover and exploit efficiently
- Our returns on investment quickly diminish, i.e. we must use more power and more transistors to expose and exploit ever smaller amounts of ILP.

Optimal pipeline depth

T = 5ns, penalty of interruption is (S-1)

Simple pipeline design
C= 300ps (clock/register overheads)
Pipeline interruption every 6 instructions

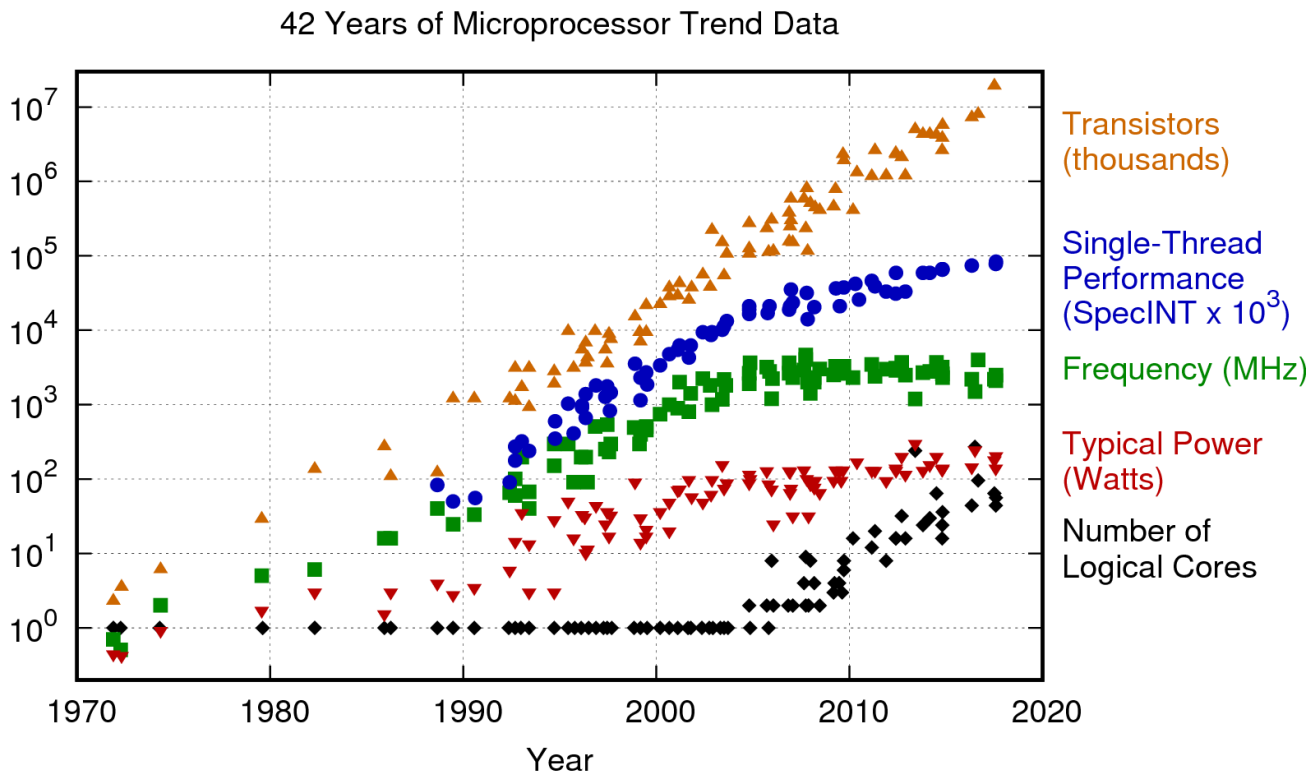
Aggressive pipeline design
C = 100ps
Pipeline interruption every 25 instructions



Source: Robert Mullins, University of Cambridge

Limits to single core performance

- **Power consumption**
- Historical performance gains have been impressive but power consumption also grew very quickly during the 1980s and 1990s
- This happened even with improvements in fabrication technology and reductions in supply voltage
- Power quickly became, and remains, a first order design constraint for all significant markets

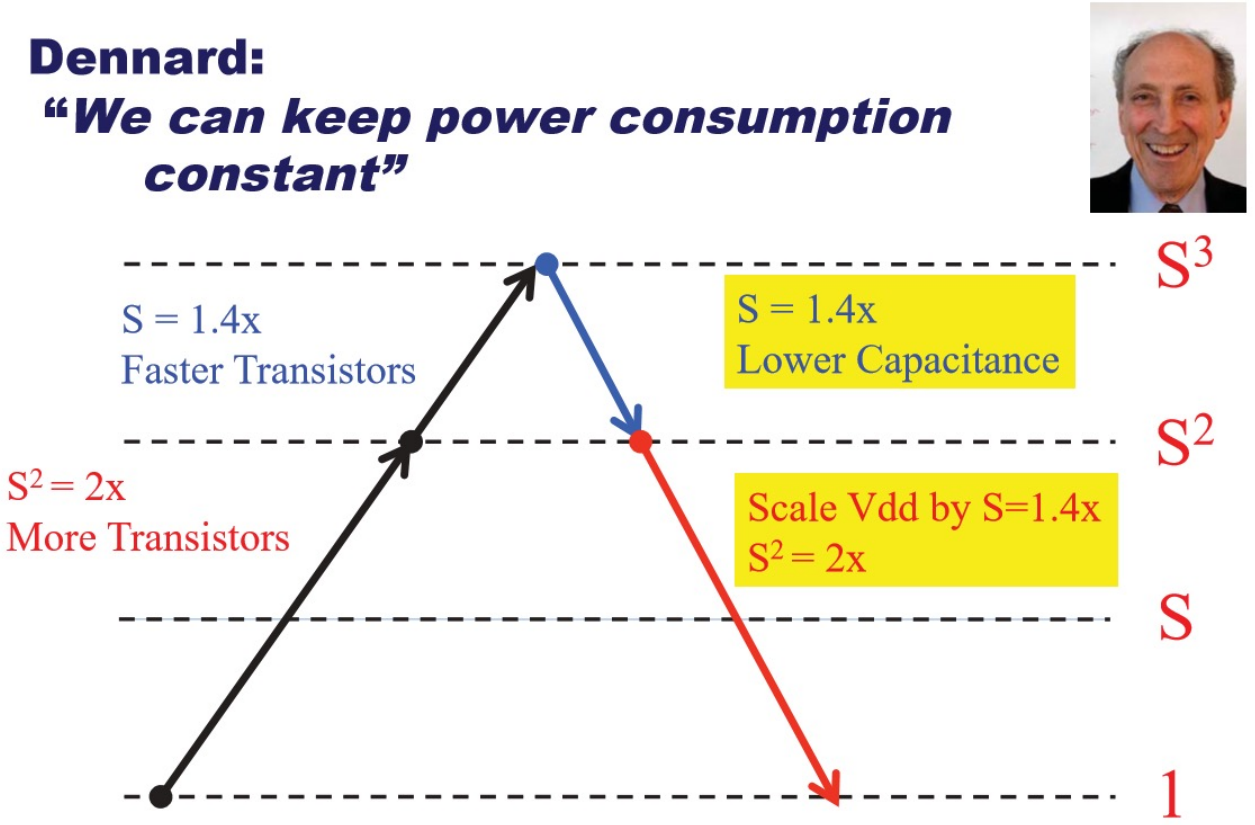


https://www.karlrupp.net/2018/02/42-years-of-microprocessor-trend-data/

Dennard and Post-Dennard scaling

Property	Dennard	Post-Dennard
Δ Quantity	S^2	S^2
Δ Frequency	S	S
Δ Capacitance	1/S	1/S
Δ Power	1	S^2
Δ Util = 1/Power	1	1/S^2

For a fixed power budget, the total chip utilisation has to fall.
Leaving us with so-called "**dark silicon**"



See "A Landscape of the New Dark Silicon Design Regime", Michael Taylor, IEEE Micro, Sept/Oct. 2013

Limits to single core performance

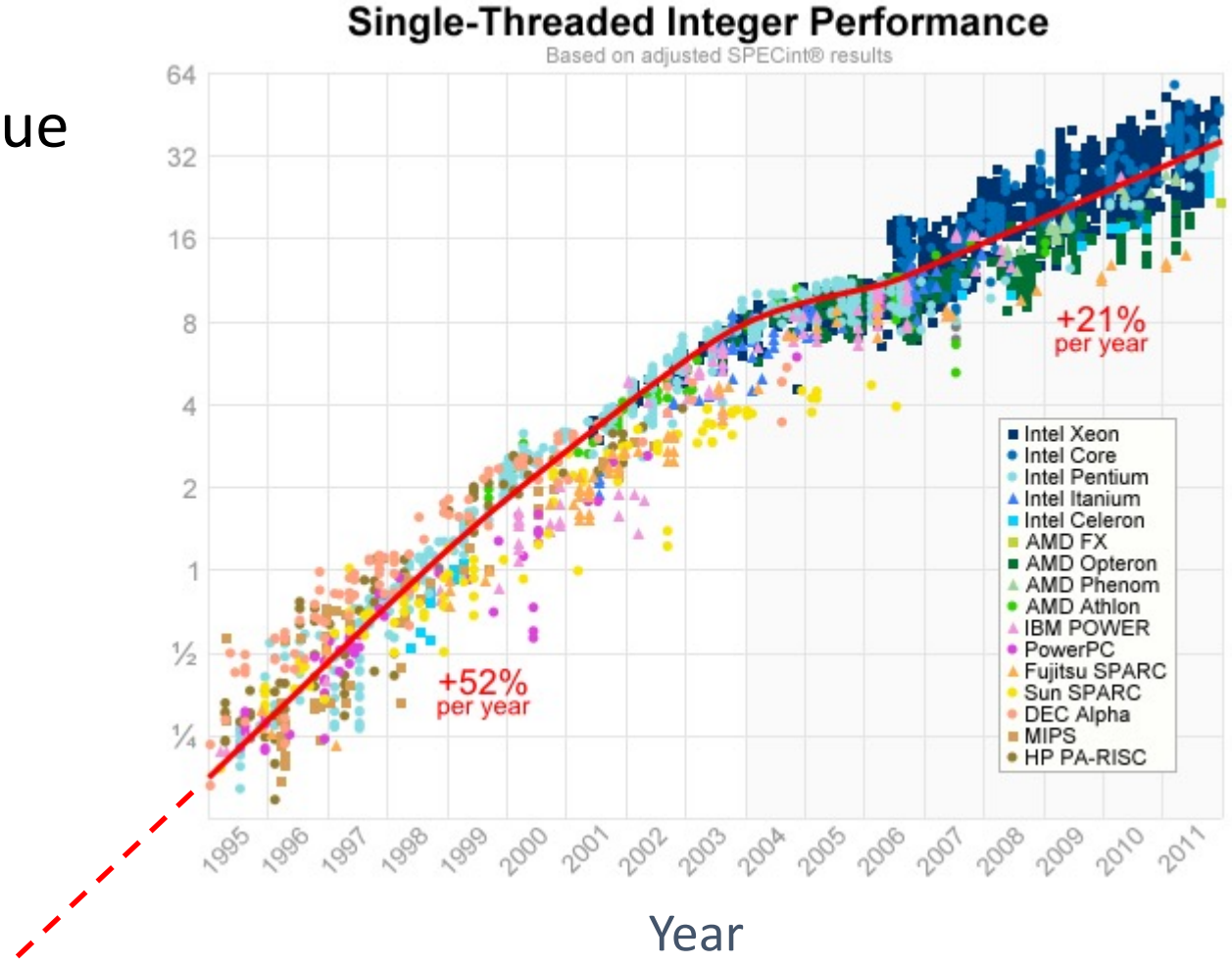
- **On-chip wiring**
  - Wire delays scale relatively poorly compared to logic delays
  - This limits the amount of state reachable in one clock cycle
  - Unfortunately, this limits the performance of large complex processors

Slowing single-core performance gains

To summarise, sustaining single core performance gains became difficult due to:

- The limits of pipelining
- The limits of Instruction-Level Parallelism (ILP)
- Power consumption
- The performance of on-chip wires

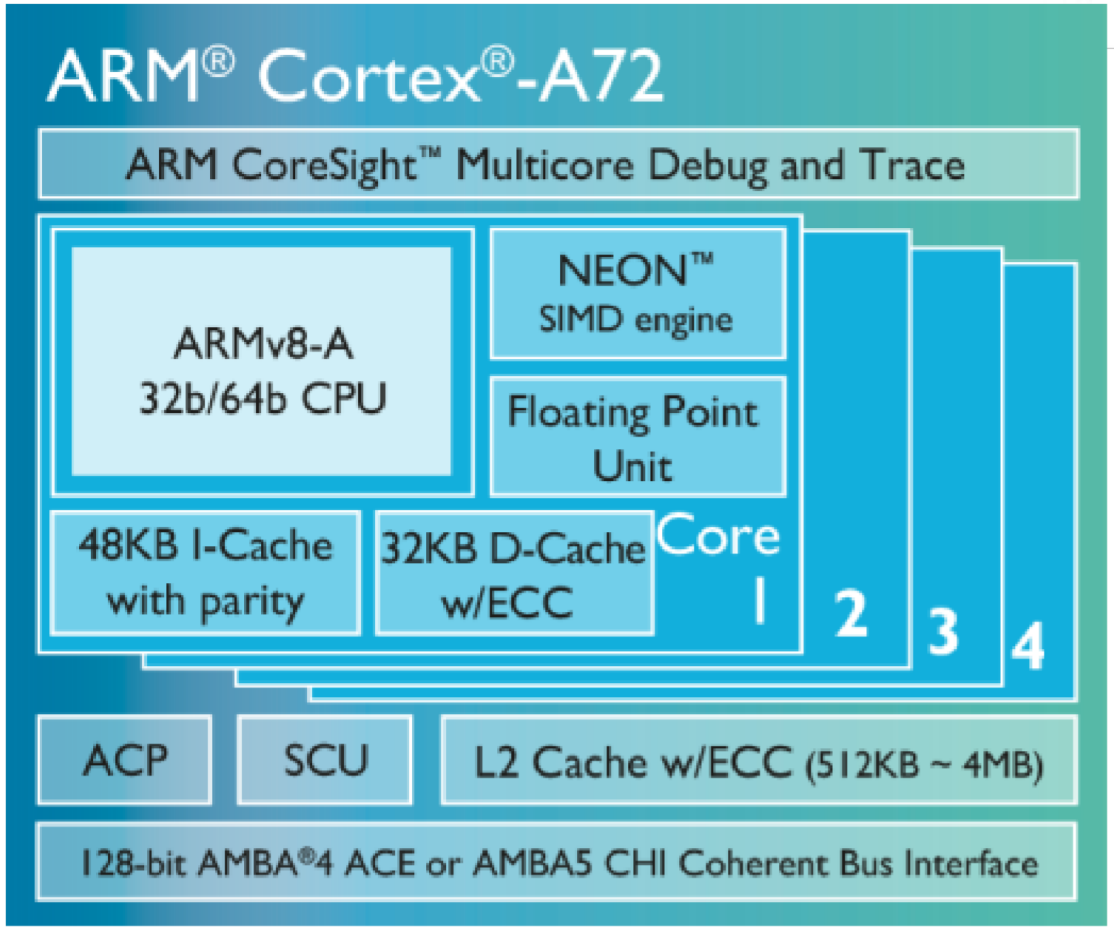
As a result performance gains slowed from 52% to 21% per year for the highest performance processors



Source: Jeff Preshing

Multicore processors

- From ~2005 multicore designs became mainstream
- The number of cores on a single chip increased over time
- Clock frequencies increased more slowly
- Individual cores were designed to be as power efficient as possible



e.g. 4 x Arm Cortex-A72 processors, each with their own L1 caches and a shared L2 cache

Multi-core processors

Exploiting multiple cores comes with its own set of challenges and limitations:

- Power consumption may still limit performance
- We need to write scalable and correct parallel programs to exploit them
- Amdahl's law
- On-chip and off-chip communication will limit performance gains
  - Off-chip bandwidth is limited and may throttle our many cores
  - Cores also need to communicate to maintain a coherent view of memory

Specialisation

- Today we often need to look beyond general-purpose programmable processors to meet our design goals
- We trade flexibility for efficiency
- We remove the ability to run all programs and design for a narrow workload, perhaps even a single algorithm
- These “accelerators” can be 10-1000x better than a general-purpose solution in terms of power and performance

Specialisation

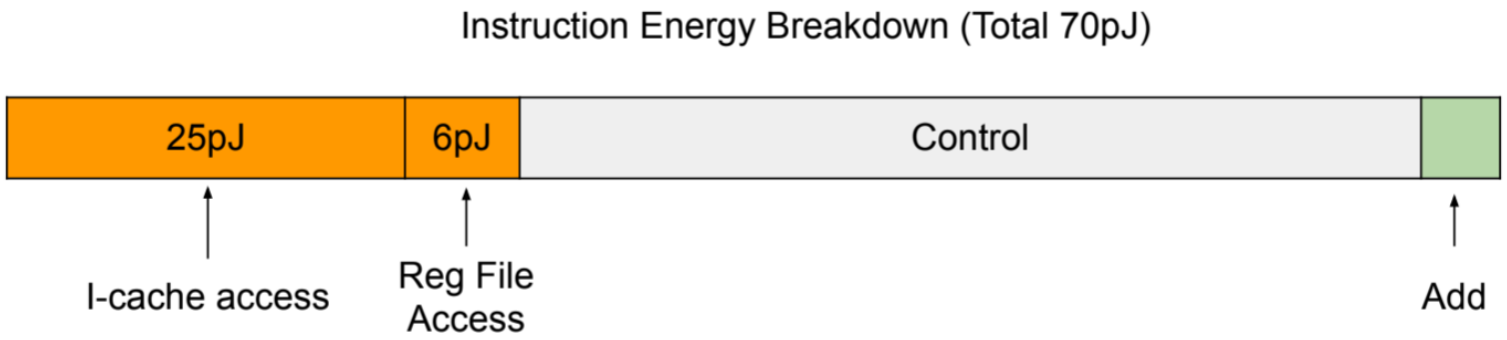
What does specialisation allow us to do?

- Remove infrequently used parts of the processor
- Tune instruction set for common operations or replace with hardwired control
- Exploit forms of parallelism abundant in the application(s) – we often see a specialised processing element and local memory reproduced many times.
- Instantiate specialised memories and tune their widths and sizes
- Provide specialised interconnect between components
- Optimise data-use patterns

Specialisation

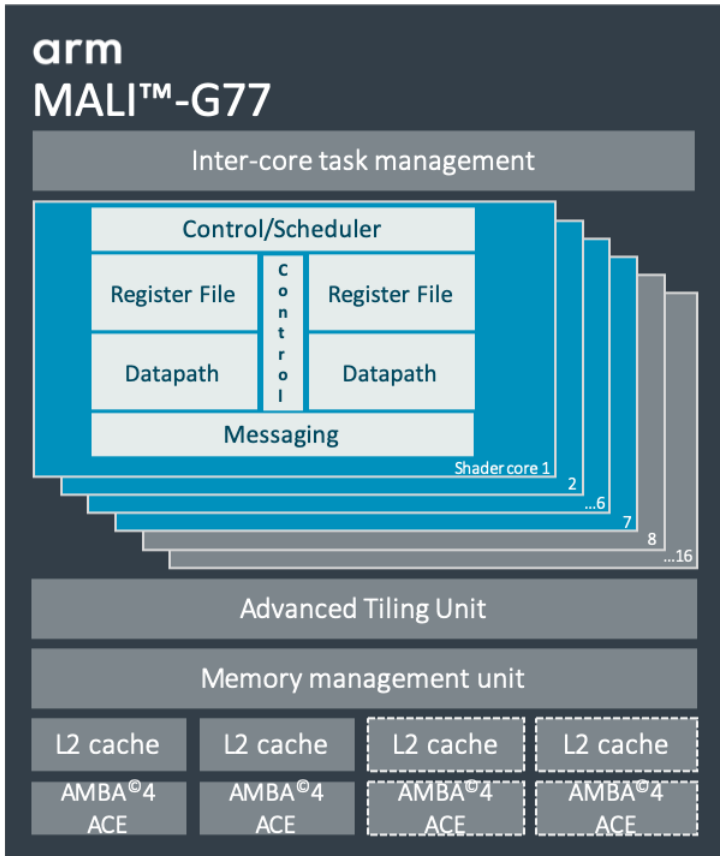
Floating Point Arithmetic	
FAdd	
16 bit	0.4pJ
32 bit	0.9pJ
FMult	
16 bit	1pJ
32 bit	4pJ

Memory	
Cache	(64 bit)
8KB	10pJ
32KB	20pJ
1MB	100pJ
DRAM	1.3 - 2.6nJ

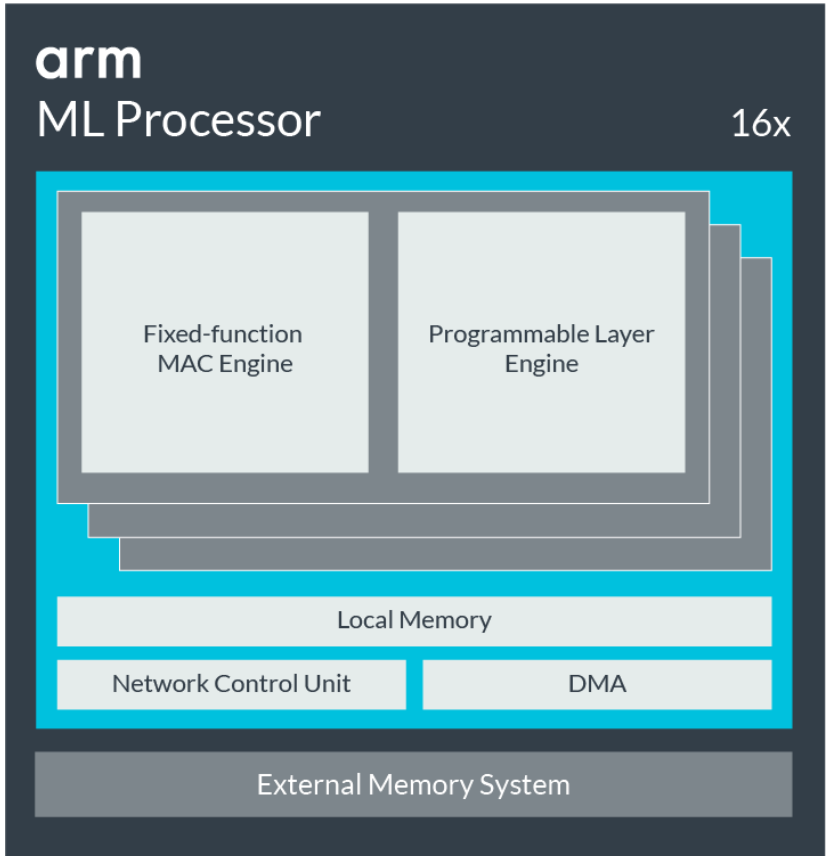


Data assumes a 45nm process @0.9V, source: “Computing’s energy problem (and what we can do about it)”, Mark Horowitz, ISSCC 2014

Specialisation



Graphics Processing Unit (GPU)



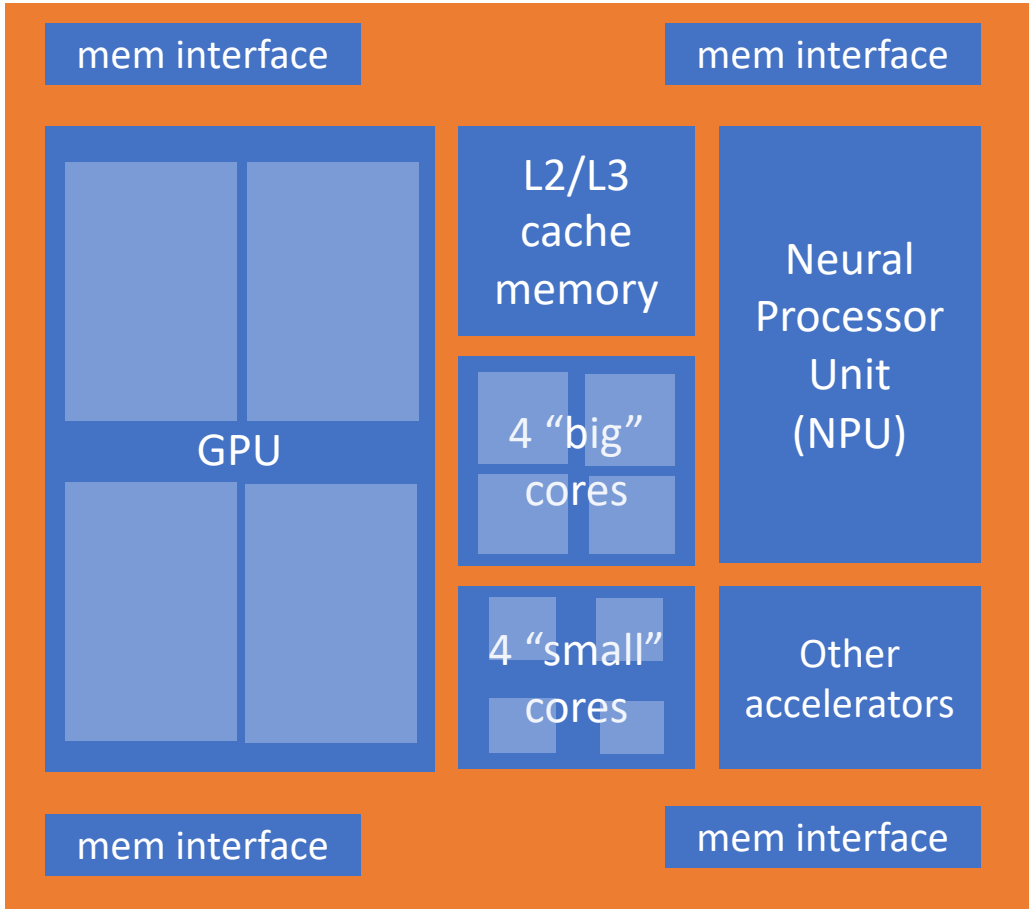
Neural Processor Unit (NPU)

Limits to specialisation

- There are new costs associated with designing each new accelerator
- The chip, or “ASIC”, produced may only be competitive in a smaller target market, reducing profitability
- Specialisation reduces flexibility
  - The logic invested in specialised accelerators is no longer general-purpose
  - Algorithm changes may render specialised hardware obsolete
- Once we’ve specialised, further gains may be difficult to achieve
  - Specialisation isn’t immune to the concept of diminishing returns

Today's System-on-Chip designs (SoCs)

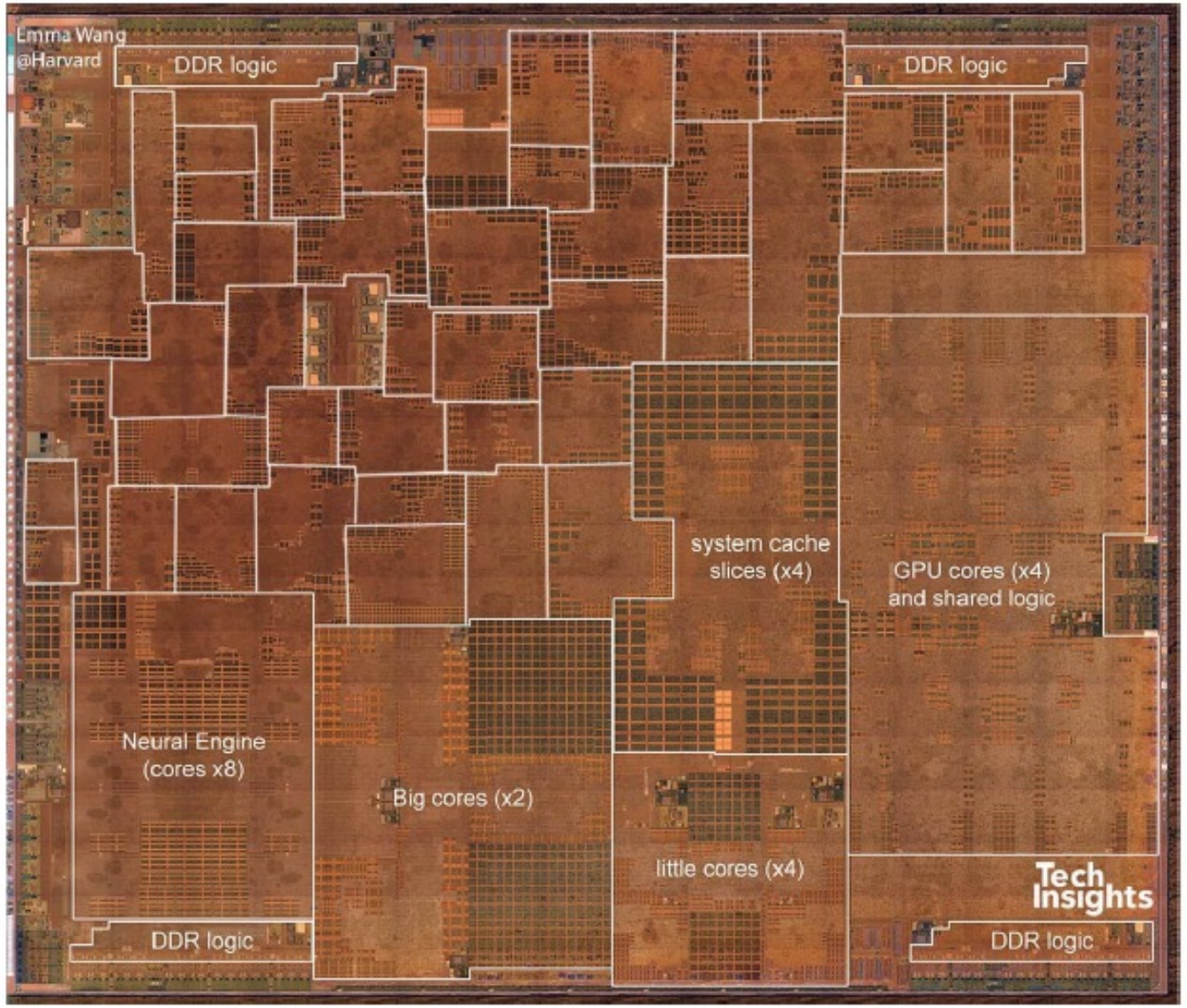
- A modern mobile phone SoC (2019) may contain more than 7 billion transistors
- It will integrate:
  - Multiple processor cores
  - a GPU
  - a large number of specialised accelerators
  - Large amount of on-chip memory
  - High bandwidth interfaces to off-chip memory



A high-level block diagram of a mobile phone SoC

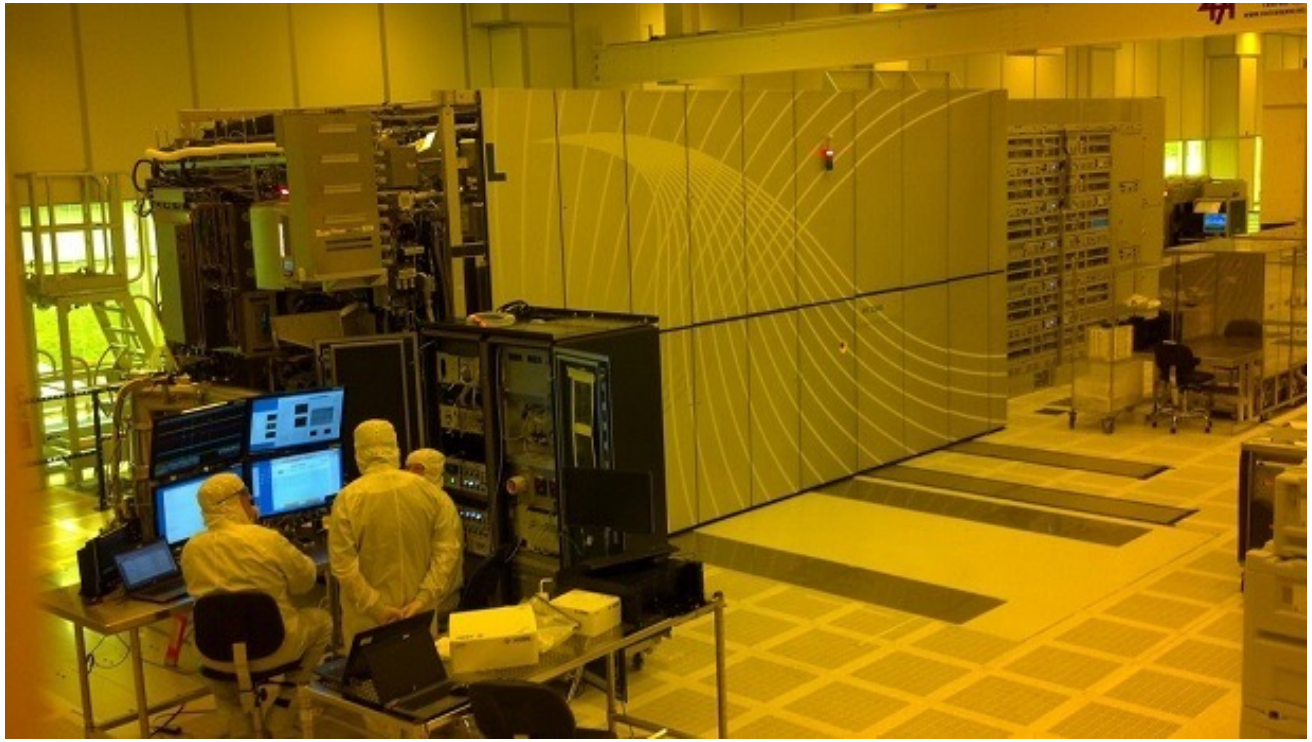
Apple A12 SoC

- 2019
- 7nm TSMC process
- 83 mm^2
- 40+ accelerators
- 2 big Arm cores,
  4 little Arm cores,
  GPU and NPU



State-of-the-art fabs: 7nm and beyond

- Oct 2019: TSMC ramping 7nm+ process towards commercial availability
- Uses Extreme Ultraviolet (EUV) lithography for critical layers
- Will be used for AMD Zen 3, Kirin 990 5G mobile processor:
  - 10+ billion transistors
  - 8 Arm cores, 5G support, NPU, ….



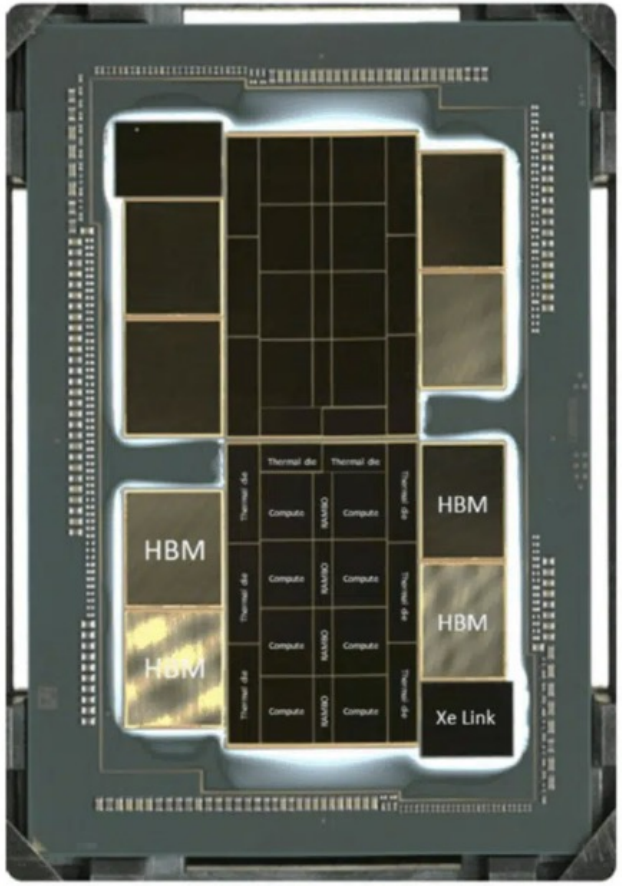
IBM, EUV lithography for 5nm
(NanoWire/NanoSheet – “gate-all-around” MOSFETs)

The future – the end of Moore's Law?

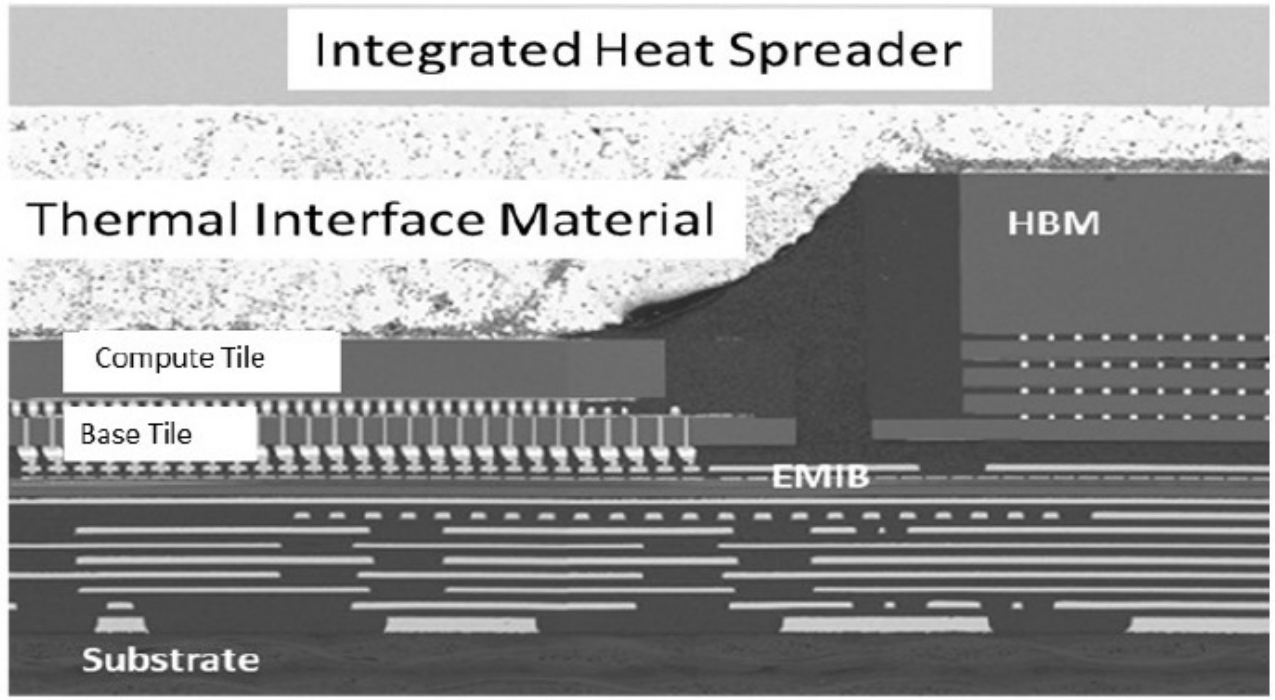
- The end of Moore's Law has been predicted many times
- Scaling has perhaps slowed in recent years but transistor density continues to improve
- Eventually 2D dimensional scaling will have to slow
  - We are ultimately limited by the number of atoms!
- Where next?
  - Interesting new packaging options, e.g. chip stacking and chiplets
  - Going 3D - Future designs may take advantage of multiple layers of transistors on a single chip (monolithic 3D). Note: the gains are linear rather than exponential
  - New types of memory (interesting compute in memory ideas, e.g. for ML)
  - New materials and devices

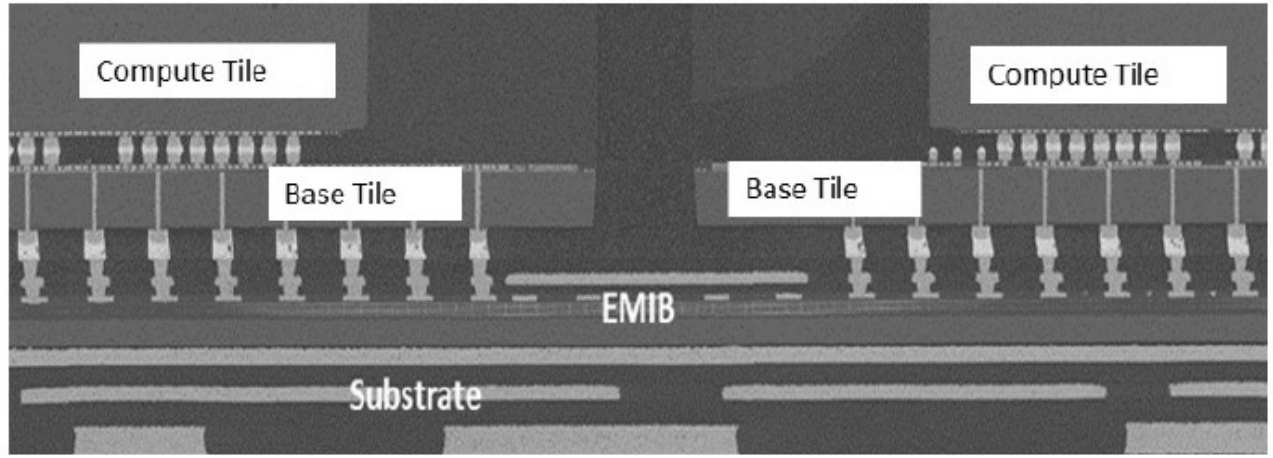
Intel: Ponte Vecchio HPC GPU

- 47 functional tiles or chiplets
- They exploit TSMC (5nm and 7nm) and Intel (7nm) processes. 5 different process nodes in total
- Stacking exploits Through Stack Vias (TSVs) and micro bumps
- Co-EMIB = silicon chips embedded in package substrate to interconnect dies
- HBM memory is again built using die stacking techniques to place many layers of DRAM on top of a memory controller die



Integration	Foveros + EMIB
Power Envelope	600W
Transistor count	> 100B
Total Tiles	63 (**47 functional** + 16 thermal Tiles)
HBM count	8
Package Form factor	77.5 x 62.5 mm (4844 mm^2)
Platforms	3 platforms
IO	4x16 90G SERDES, 1x16 PCIe Gen5
Total Silicon	3100 mm^2 Si
Silicon footprint	2330 mm^2 Si footprint
Package layers	11-2-11 (24 layers)
2.5D Count	11 2.5D connections
Resistance	0.15 mΩ R_{path}/tile
Package pins	4468 pins
Package Cavity	186 mm^2 x4 cavities


Integrated Heat Spreader
Thermal Interface Material
HBM
Compute Tile
Base Tile
EMIB
Substrate


Compute Tile
Compute Tile
Base Tile
Base Tile
EMIB
Substrate