Imperial College
London

The "Turing Tax" - reducing the interpretive bottleneck in computer architecture

UK Design Forum, Manchester, June 2025

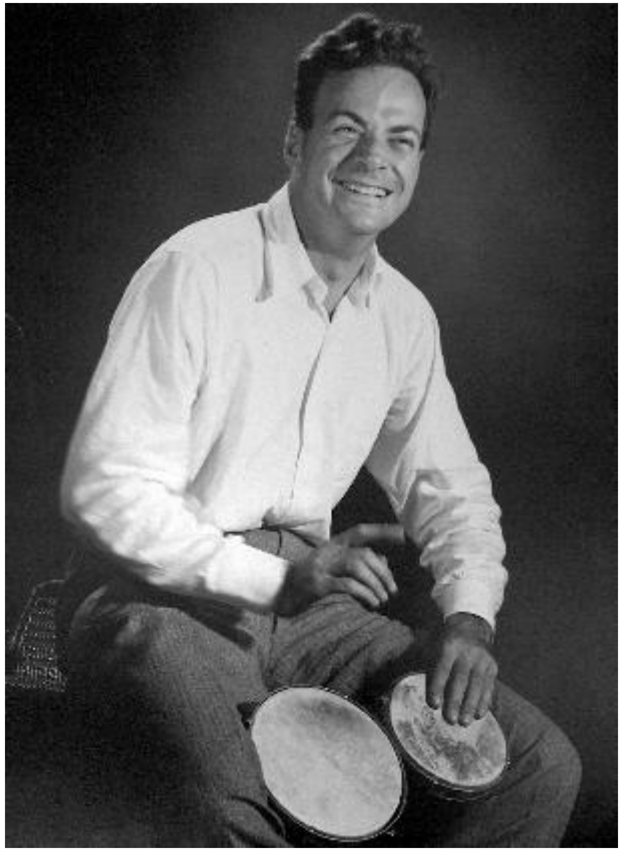
Paul H J Kelly
Group Leader, Software Performance Optimisation
Department of Computing, Imperial College London

This talk includes work done by or influenced by: David Ham (Imperial Maths), Andy Davison (Imperial Computing), Lawrence Mitchell (NVIDIA),Gerard Gorman (Imperial Earth Science Engineering), Peter Vincent (Imperial Aeronautics), Wayne Luk (Imperial Computing), Piotr Dudek (Manchester), Jim Whittaker (Huawei), Vassilios Chouliaras (Huawei), Martin Berger (Montanarius), Sam Ainsworth (Edinburgh) and more

And affiliated postdocs/PhD students: Luke Panayi, Jacky Wong, Nicholas Fry, Shinjeong Kim, Edward Stow, Yuncheng Lu, Shuang Liang, Riku Murai, George Bisbas

And by group alumni: Eduardo Carvalho (Valantis Labs), Marius Koch (NVIDIA), Navjot Kukreja (Senapt), Philippos Papaphilippou (U. Southampton), Edward Stow (TBC), Matthew Taylor, Sophia Vorderwuelbecke, Fabio Luporini (DevitoCodes), Graham Markall (NVIDIA), Florian Rathgeber (Google), Francis Russell (Quaisr), George Rokos (Intel), Tianjiao Sun (Orbit Markets), Thanasis Konstantinides (Cerebras), Carlo Bertolli (AMD), Doru Bercea (AMD), Michael Lange (ECMWF), Sajad Saeedi (UCL), Luigi Nardi (DBTune/Lund University), Navjot Kukreja (Senapt), Emanuele Vespa (Magic Leap), Miklos Homolya and more

And by collaborators: Mike Giles (Oxford), Gihan Mudalige (Warwick), Istvan Reguly (Pazmany Peter, Hungary), Matt Piggott (Imperial ESE), Spencer Sherwin, Chris Cantwell (Imperial Aero), Michelle Mills Strout (University of Arizona/HPE), Chris Krieger (Maryland), Cathie Olschanowsky (Colorado State University), Bruno Bodin (Yale-NUS), Richard Veras, Ram Ramanujam (Louisiana State University), Doru Thom Popovici (LBL), Franz Franchetti (CMU), Jan Hückelheim (Argonne), Freddie Witherden (Texas A&M), Chris–Kriton Skylaris (Southampton) Ridgway Scott (U Chicago), Lluis Guasch (Imperial Earth Science Engineering)



Feynmann: plenty of room at the bottom

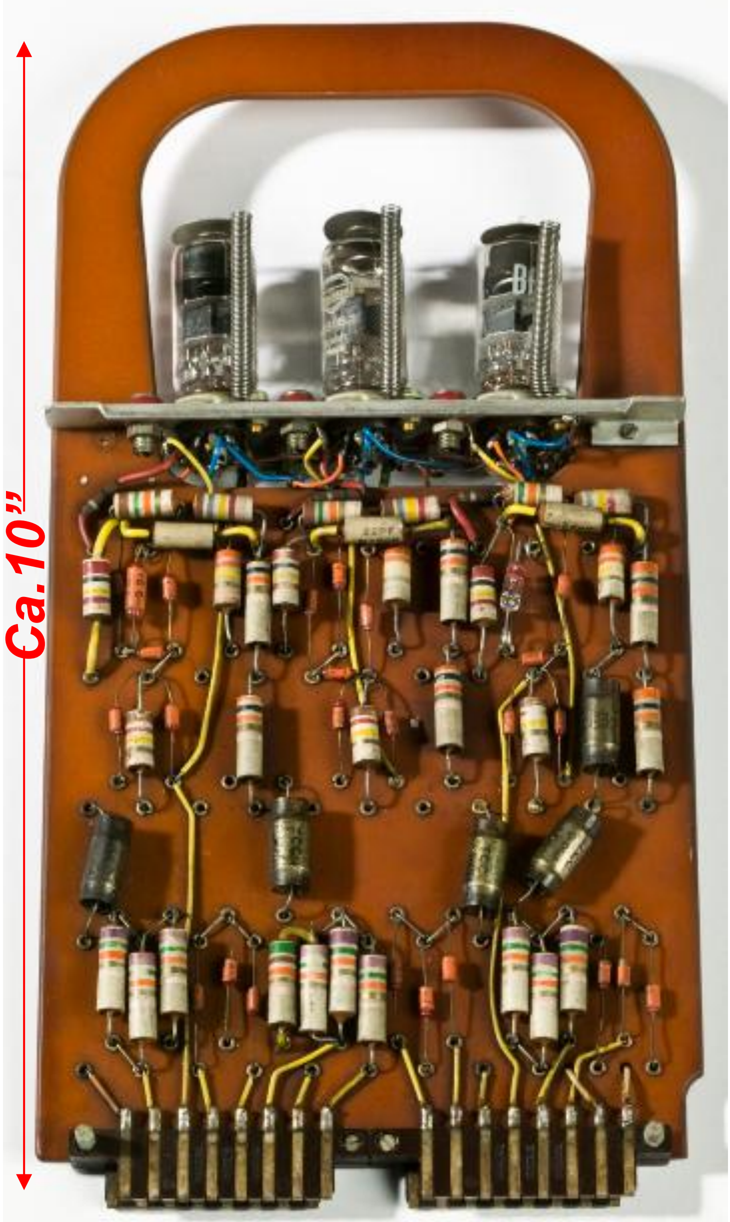
"…I do know that computing machines are very large; they fill rooms. Why can't we make them very small, make them of little wires, little elements – and by little, I mean little? For instance, the wires should be 10 or 100 atoms in diameter, and the circuits should be a few thousand angstroms across"

- >60 years of exponential progress since then
- We're much closer to such limits
- Much debate about where they really lie
- What is clear is that we're a lot closer
- We are confronted more and more with fundamental physical concerns
- Particularly wrt communication latency, bandwidth and energy.

https://en.wikipedia.org/wiki/There's_Plenty_of_Room_at_the_Bottom

(1959, talk at the American Physical Society)

Ferranti Pegasus (1956-59)

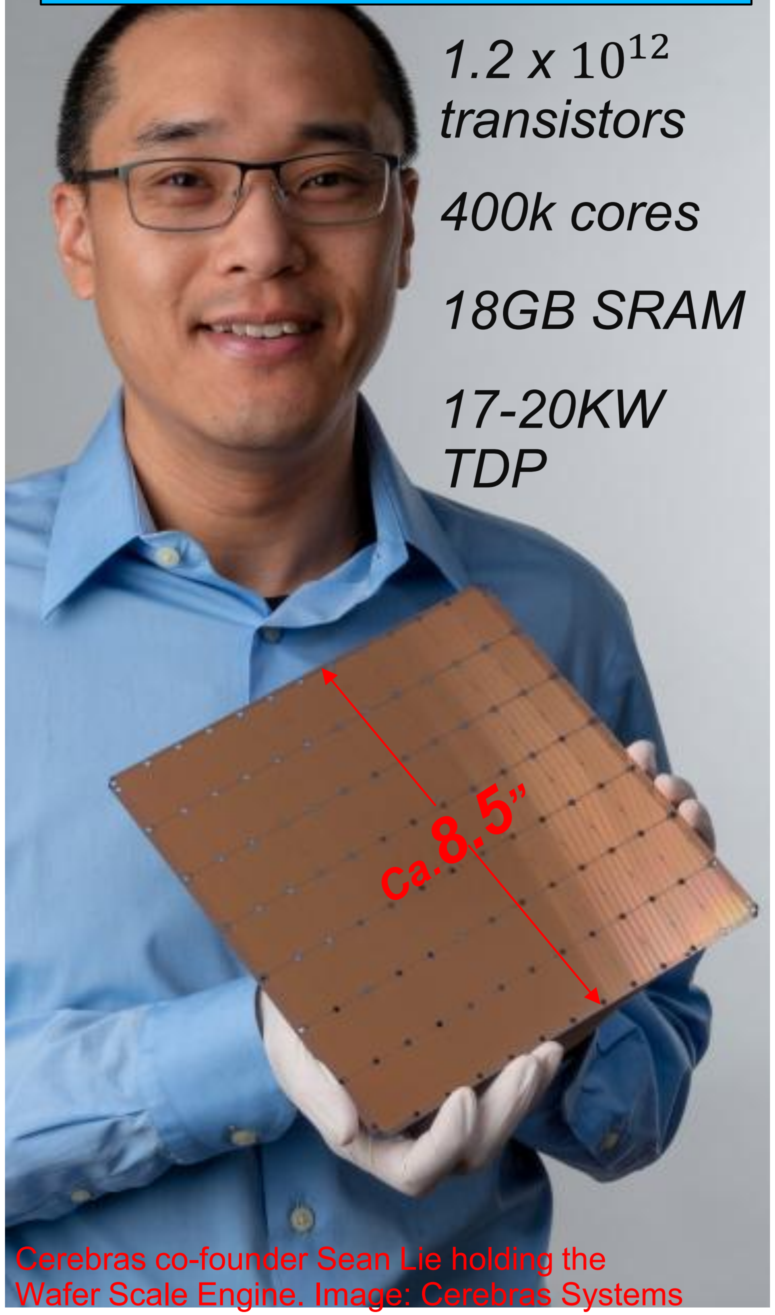


Ferranti Computer Systems Ltd Pegasus valve computer circuit board, c. 1964

https://blog.sciencemuseum.org.uk/the-pegasus-computer/

- Moore’s Law: “circuit density doubles every 18 months”
- 60 years =40x18months
- So Moore’s Law would predict 2^{40}= 10^{12} increase

Cerebras CS-1 (2020)



1.2 x 10^{12} transistors

400k cores

18GB SRAM

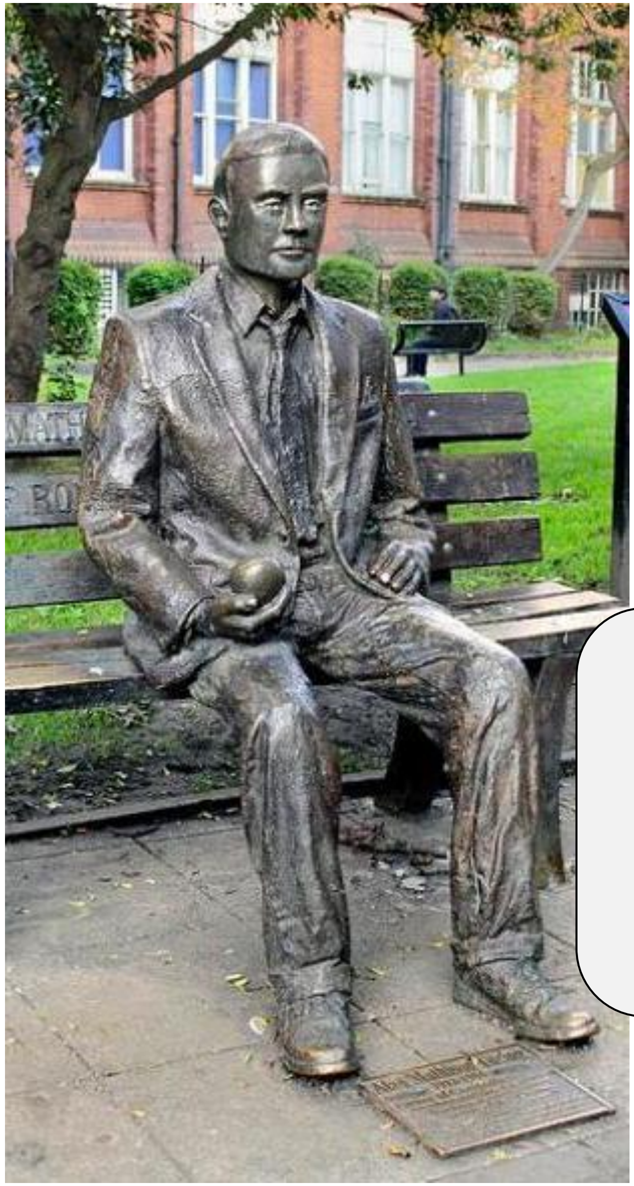
17-20KW TDP

Cerebras co-founder Sean Lie holding the Wafer Scale Engine. Image: Cerebras Systems

Turing ~~tax~~ tariffs

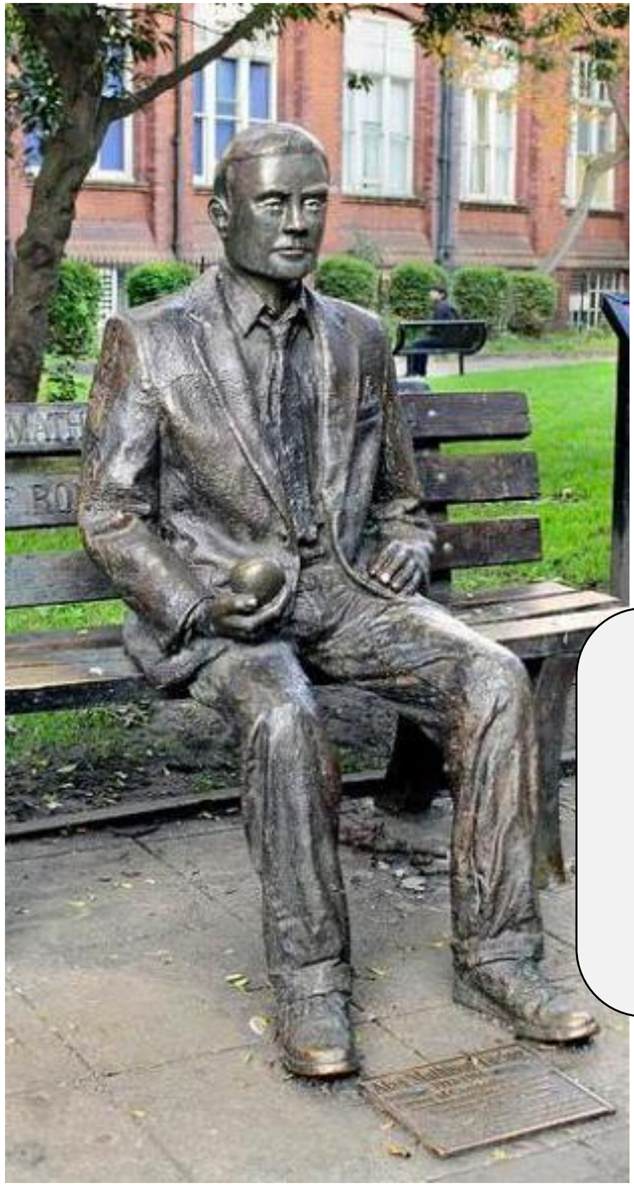
- Alan Turing realised we could use digital technology to implement any computable function
- He then proposed the idea of a “universal” computing device – a *single* device which, with the right program, can implement any computable function *without further configuration*
- “**Turing Tax**”, **or** “**Turing Tariffs**”: the overhead (performance, cost, or energy) of universality in this sense
- The performance (time/area/energy) difference between a **special-purpose** device and a **general-purpose** one
- **One of the fundamental questions of computer architecture is to how to reduce the Turing Tax**

Turing ~~tax~~ tariffs



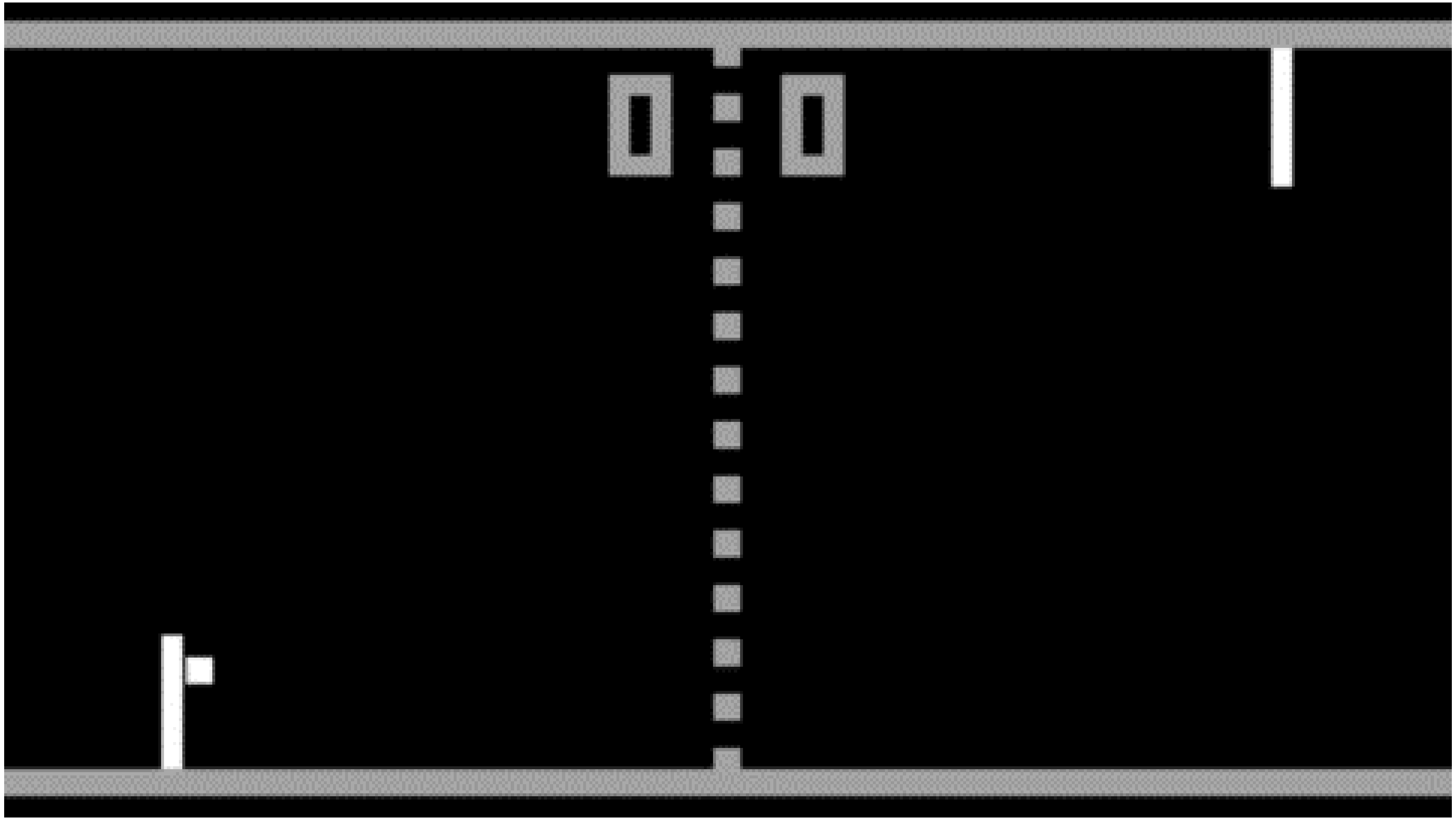
- Alan Turing realised we could use digital technology to implement any computable function
- He then proposed the idea of a "universal" computing device – a *single* device which, with the right program, can implement any computable function *without further configuration*
- "**Turing Tax**", **or** "**Turing Tariffs**": the overhead (performance, cost, or energy) of universality in this sense
- The performance (time/area/energy) difference between a **special-purpose** device and a **general-purpose** one
- **One of the fundamental questions of computer architecture is to how to evade the Turing Tax**

Turing ~~tax~~ tariffs



- Alan Turing realised we could use digital technology to implement any computable function
- He then proposed the idea of a “universal” computing device – a *single* device which, with the right program, can implement any computable function *without further configuration*
- “**Turing Tax**”, **or** “**Turing Tariffs**”: the overhead (performance, cost, or energy) of universality in this sense
- The performance (time/area/energy) difference between a **special-purpose** device and a **general-purpose** one
- **One of the fundamental questions of computer architecture is to how to reduce the Turing Tariffs**

PONG

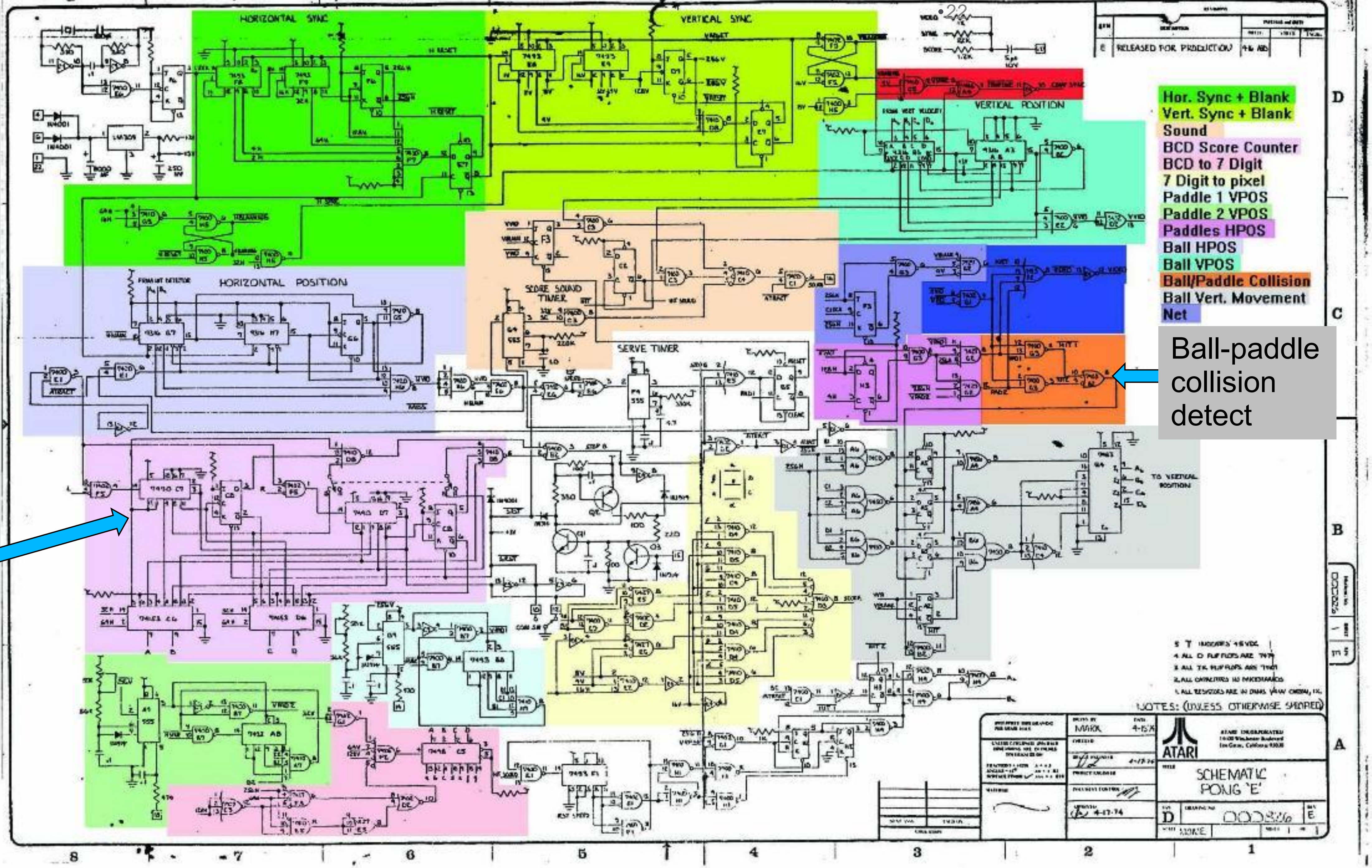

0
0

Circuit diagram for Atari's Pong game

Designed by Allan Alcorn in around 1972

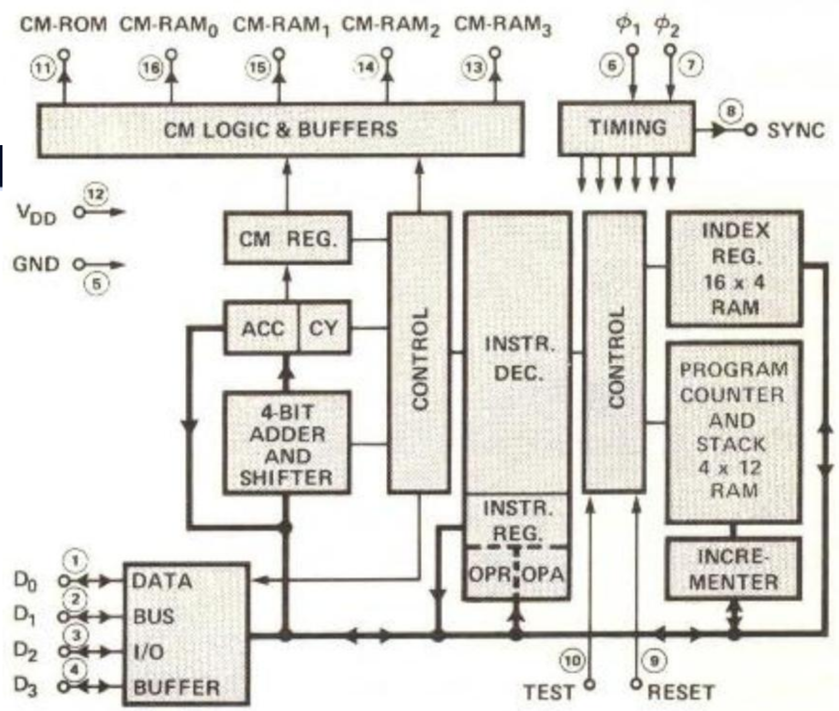
Alcorn hired Steve Jobs in 1974

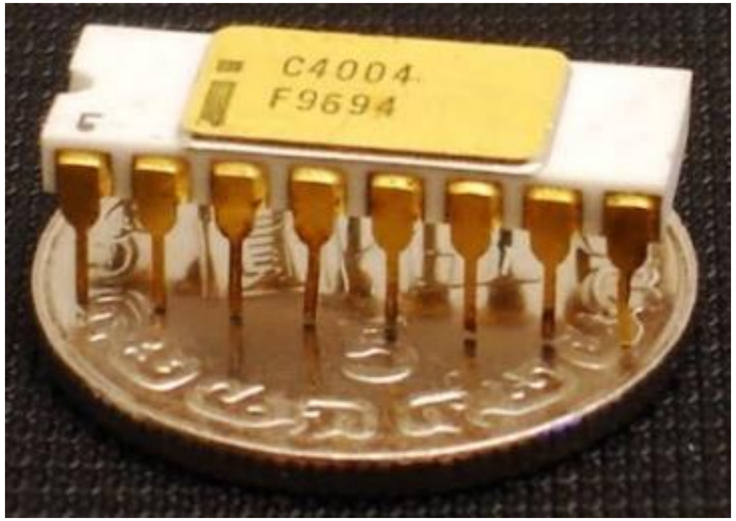
7490 TTL IC: "decade counter" for tracking the score (ca.45 transistors)

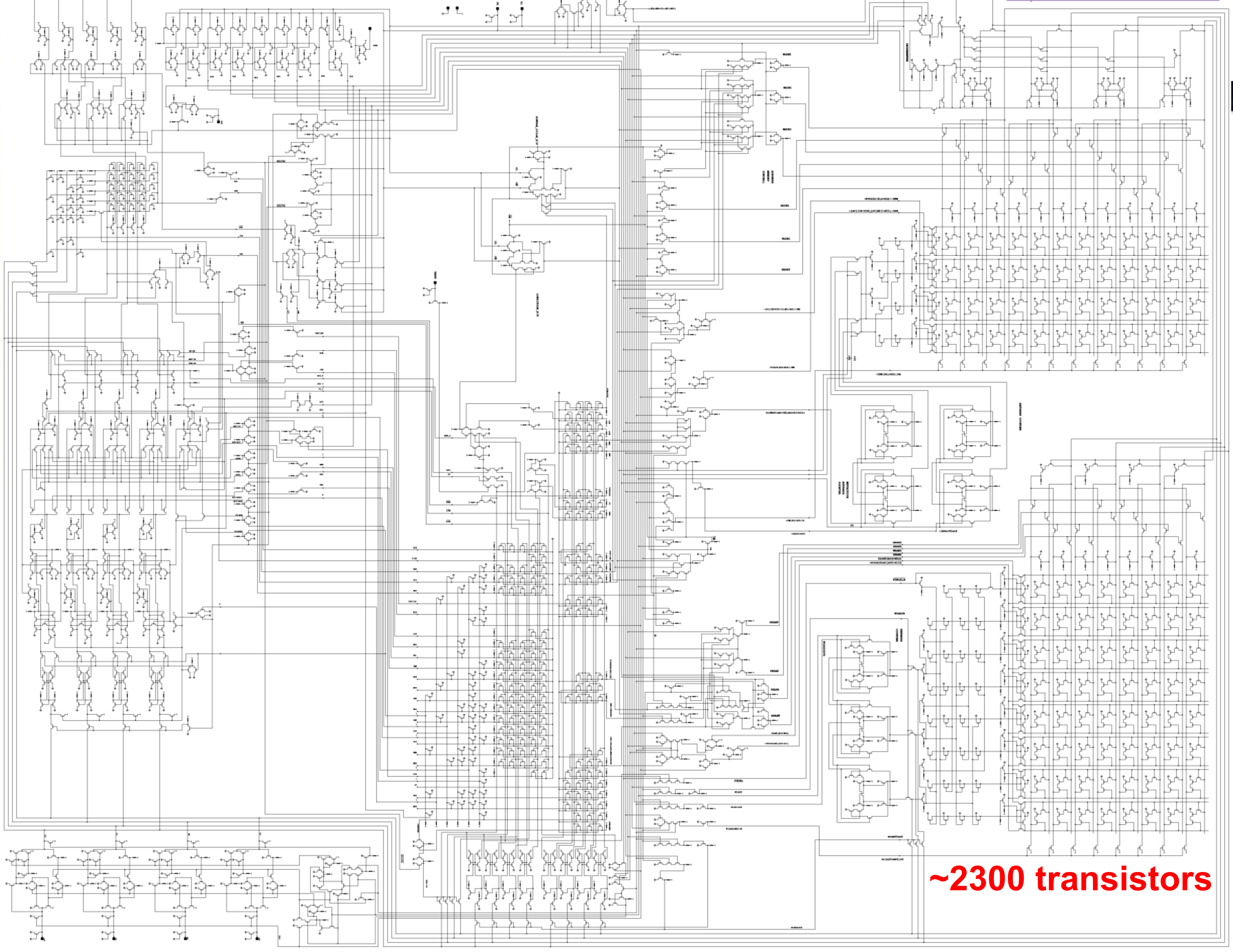


Ball-paddle collision detect

https://www.pong-story.com/LAWN_TENNIS.pdf https://www.4004.com/







~2300 transistors

Circuit diagram for the first commercially-available microprocessor, Intel’s 4004 (1971)

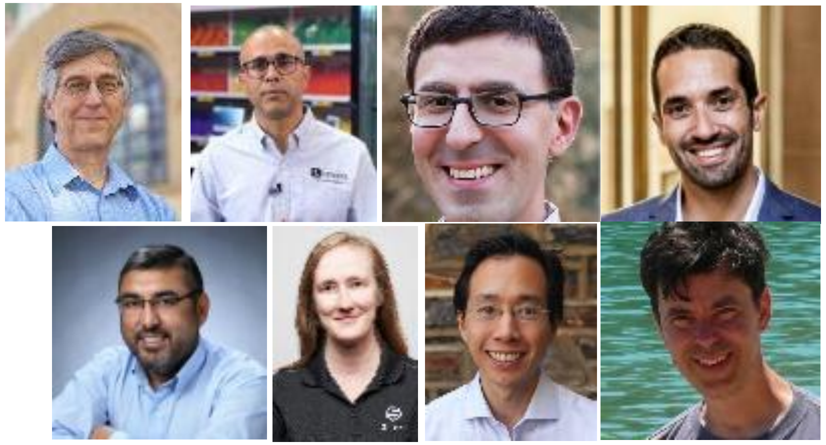
- **So how big is the Turing Tax?**

- Suppose you have a job that you can do on a single CPU core
- But you invest in a fully-specialised ASIC instead

- **How much faster?**
- **How much smaller?**
- **How much less energy?**

Example: H264 encoding

- 2010 article: Hameed, Qadeer, Wachs, Azizi, Solomatnikov, Lee, Richardson, Kozyrakis & Horowitz: **Understanding sources of inefficiency in general-purpose chips**. ISCA'10



- Intel's hand-coded implementation of H.264 encoding for Pentium 4 for 1280x720 HD:
  - 11fps
  - 122mm^2
  - 2023mJ/frame

- ASIC implementation of H.264 encoding for 1280x720 HD:
  - 30fps
  - 8mm^2
  - 4mJ/frame

(Chen, Chien, Huang, Tsai, Chen, Chen, & Chen: Analysis and architecture design of an HDTV720p 30 frames/s H.264/AVC encoder. IEEE Trans. Cir. and Sys. for Video Technol. 16, 6 (September 2006))

Turing tariffs

- Fetch-execute is the original Turing tariff
- FPGAs pay Turing tariffs in the reconfigurable fabric

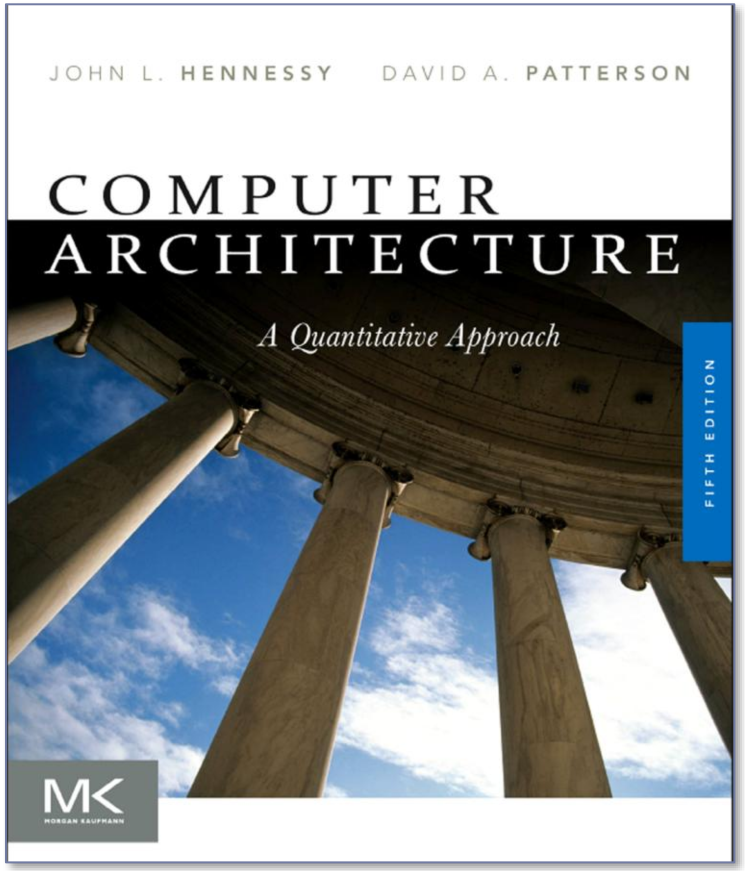
- Registers are a Turing Tariff
  - Because if we know the program's dataflow, we can use wires and latches to pass data from functional unit to functional unit

- Memory
  - But if we can stream data from where it's produced to where it's used, maybe we don't need so much RAM?

- Cache
  - If we know exactly when the reuse will occur, we can program movement to and from local fast memory explicitly

- Floating-point arithmetic:
  - If we know the dynamic range of expected values…

Turing tariffs – how architects pay



- Fetch-execute, decode
- Registers, forwarding
- Dynamic instruction scheduling, cracking, packing, renaming
- Cache tags
- Cache blocks
- Cache coherency
- Prefetching
- Branch prediction
- Speculative execution
- Address translation
- Store-to-load forwarding, write combining, address decoding, ECC, DRAM refresh
- Mis-provisioning: unused bandwidth, unusable FLOPs, under-used accelerators

Basically the whole computer architecture textbook

How architects avoid Turing tariffs

- SIMD: amortise fetch-execute over a vector or matrix of operands
- Predication, VLIW, EPIC, register rotation
- Macro-instructions: FMA, crypto, conflict-detect, custom ISAs

- Streaming dataflow: FPGAs, CGRAs, systolic arrays
- Circuit switching instead of packet switching
- DMA, long cache lines: move a lot of data with one instruction

- Non-temporal loads/stores, explicit prefetch instructions
- Scratchpads ("shared memory" in CUDA)
- Message passing, instead of cache-coherent shared memory

- Texture caches, interpolation, decompression

- Fine-grain multithreading to avoid pipeline hazards, hide latency

- **And many more ideas…. *Your ideas?***

How *compilers* avoid Turing tariffs

Generating code to avoid the need for interpretive mechanisms in hardware:

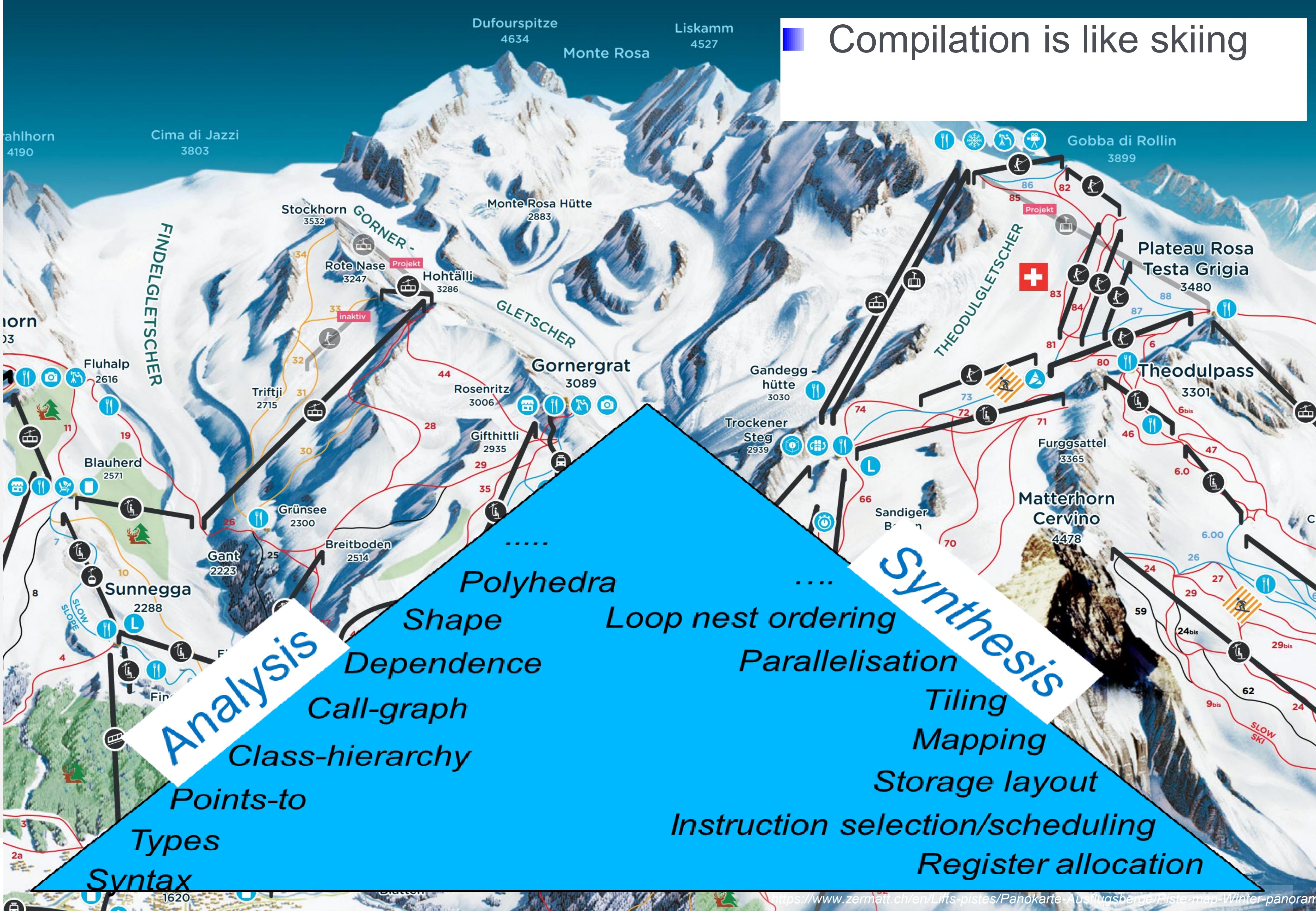
- Vectorisation
- Static instruction scheduling
- Offloading
- Predication
- Message aggregation
- Synchronisation minimization

Generating code that is specialized for a specific purpose:

- Function inlining, type disambiguation, object inlining
- Specialisation: metaprogramming, JIT, metatracing

Compilation is like skiing

- What about **compilers ?**
- **Is there a Turing Tax for compilers too?**



Analysis

- Syntax
- Types
- Points-to
- Class-hierarchy
- Call-graph
- Dependence
- Shape
- Polyhedra
- .....

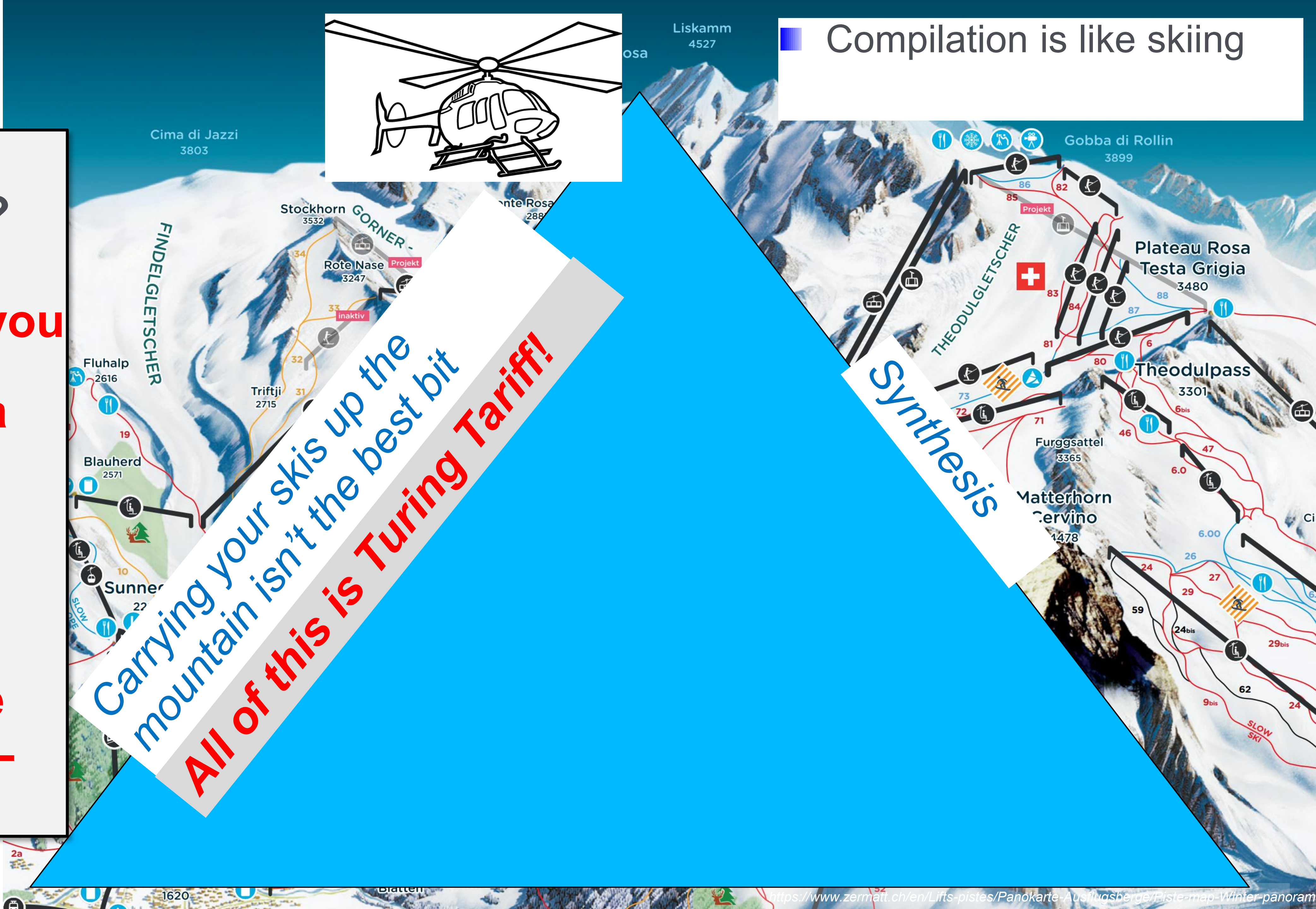
Synthesis

- Register allocation
- Instruction selection/scheduling
- Storage layout
- Mapping
- Tiling
- Parallelisation
- Loop nest ordering
- ….

https://www.zermatt.ch/en/Lifts-pistes/Panokarte-Ausflugsberge/Piste-map-Winter-panoram

Compilation is like skiing

- What about **compilers?**
- **The price you pay for coding in a general-purpose language**
-
- **When you could have used a DSL**



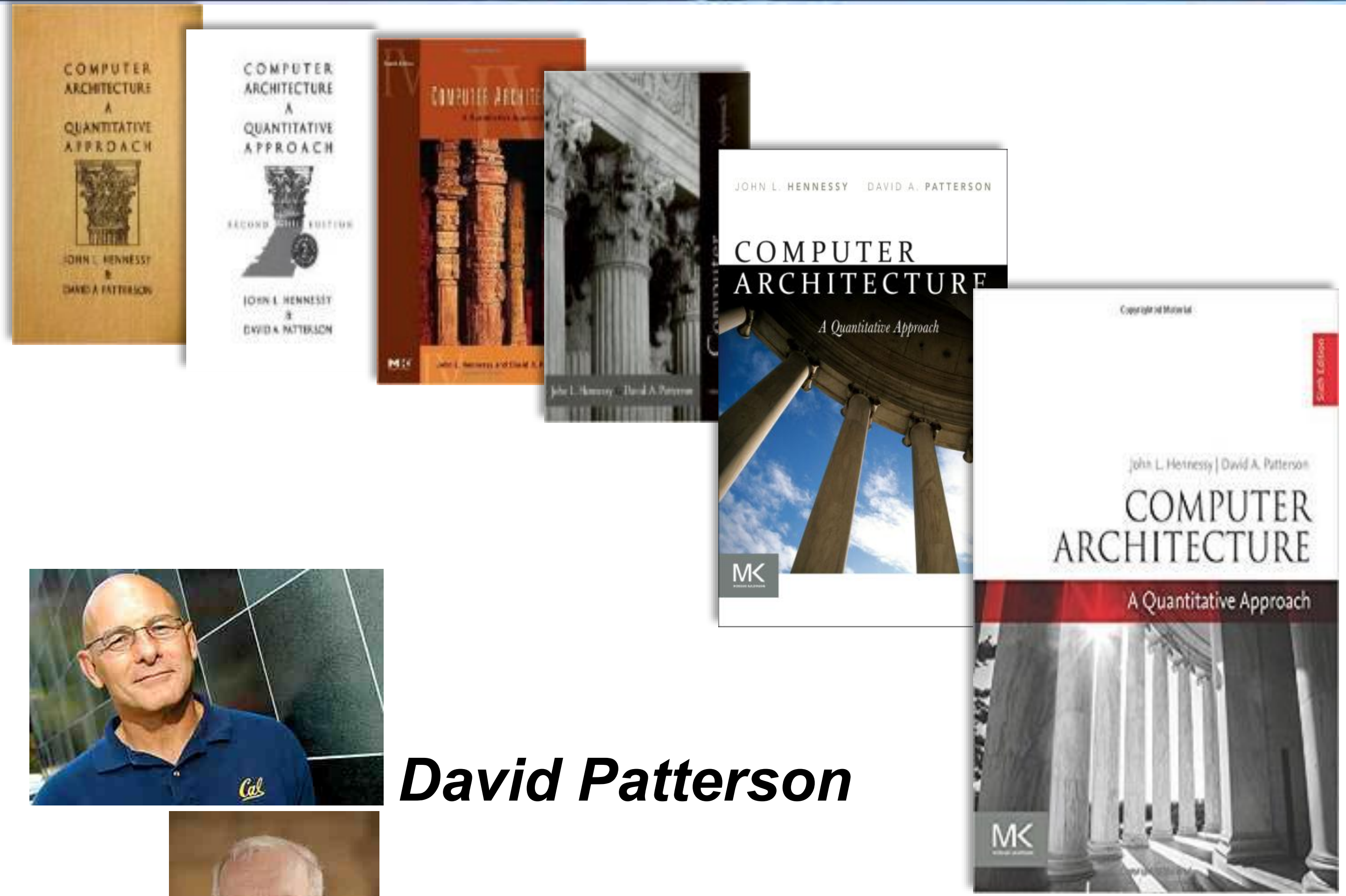
Carrying your skis up the mountain isn't the best bit

All of this is Turing Tariff!

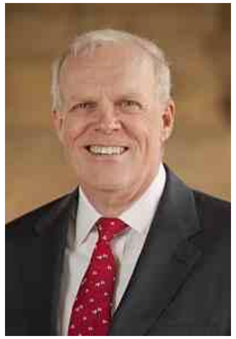
Synthesis

https://www.zermatt.ch/en/Lifts-pistes/Panokarte-Ausflugsberge/Piste-map-Winter-panoram

Computer architecture – the book


COMPUTER ARCHITECTURE A QUANTITATIVE APPROACH
JOHN L. HENNESSY & DAVID A. PATTERSON
SECOND EDITION
JOHN L. HENNESSY DAVID A. PATTERSON
COMPUTER ARCHITECTURE
A Quantitative Approach
MK
John L. Hennessy | David A. Patterson
COMPUTER ARCHITECTURE
A Quantitative Approach
MK

David Patterson



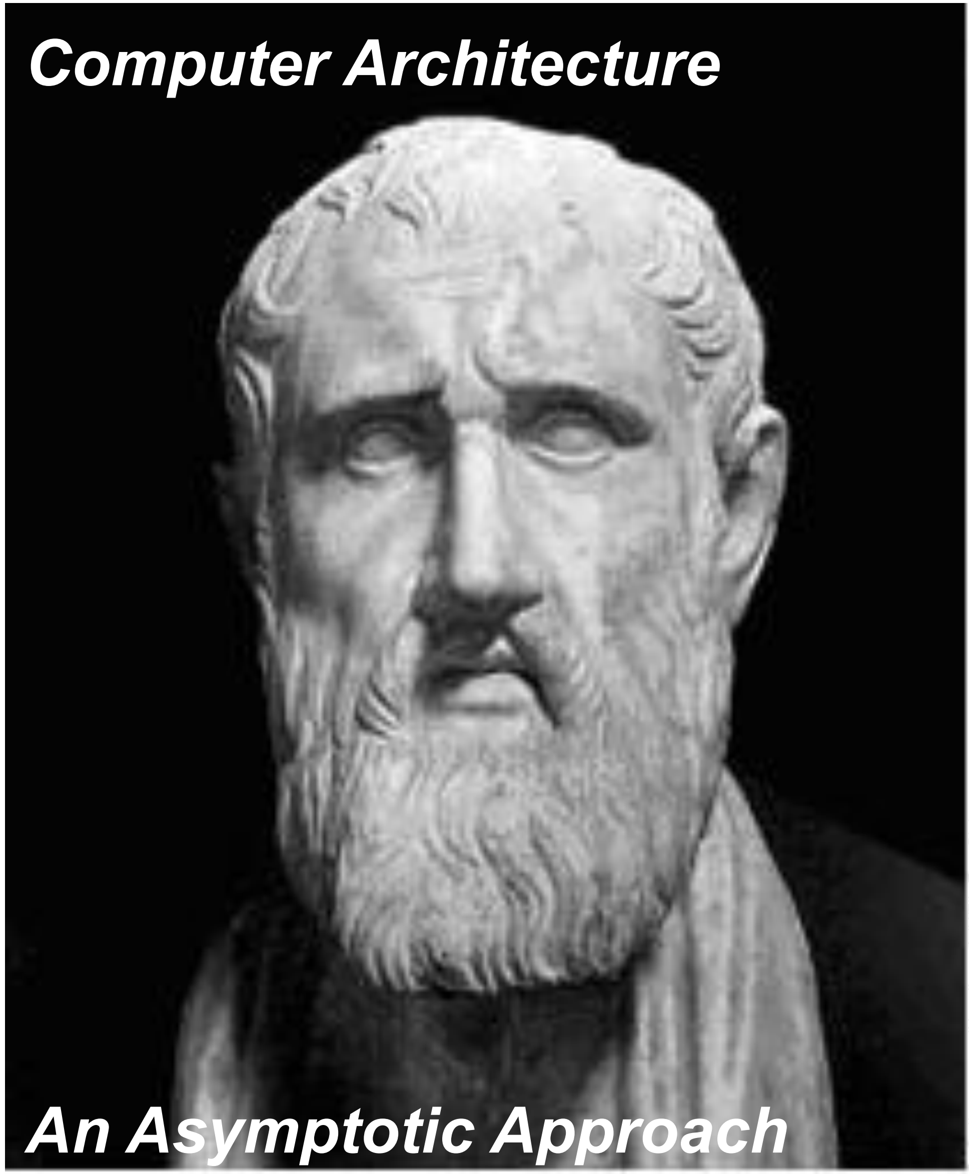
John Hennessy

- Computer Architecture: A Quantitative Approach
- Six editions since 1990
- Revolutionary landmark book brought experimental discipline to processor design
- Almost entirely devoid of theory

Computer architecture – the future?

COMPUTER ARCHITECTURE A QUANTITATIVE APPROACH
JOHN L. HENNESSY & DAVID A. PATTERSON
SECOND EDITION
JOHN L. HENNESSY DAVID A. PATTERSON
COMPUTER ARCHITECTURE
A Quantitative Approach
MK
John L. Hennessy | David A. Patterson
COMPUTER ARCHITECTURE
A Quantitative Approach
MK

- A manifesto
- For computer architecture at the end of Moore’s Law
- Where we confront fundamental physical constraints
- Where we have to account for fundamental costs
- **Where architectural efficiency is paramount**



https://commons.wikimedia.org/wiki/File:ZenoPlain.png

(Statue is of Zeno of Citium, perhaps the patron saint of asymptotics)

Acknowledgements

Partly funded by

- UKRI On-Sensor Computer Vision EP/Y020499/1
- UKRI Reliable and Robust Quantum Computing EP/W03221X/1
- UKRI SysGenX: Composable software generation for system-level simulation at Exascale EP/W026066/1
- UKRI Efficient Cross-Domain DSL Development for Exascale EP/W007789/1
- EPSRC “PAMELA” Programme Grant (EP/K008730/1)
- UKRI Gen X: ExCALIBUR working group on Exascale continuum mechanics through code generation (EP/V001493/1)
- EPSRC “MAPDES” project (EP/I00677X/1)
- EPSRC “PSL” project (EP/I006761/1)
- Rolls Royce and the TSB through the SILOET programme
- EPSRC “PRISM” Platform Grants (EP/I006761/1, EP/R029423/1)
- EPSRC “Custom Computing” Platform Grant (EP/I012036/1)
- AMD, ARM, Arup, Codeplay, Huawei, IBM, Maxeler Technologies, Microsoft, VMWare