From T0 and CNS-1 to RISC-V and AI Hardware

Krste Asanovic
Professor, EECS Dept, UC Berkeley
Chairman of Board, RISC-V Foundation
Co-Founder and Chief Architect, SiFive

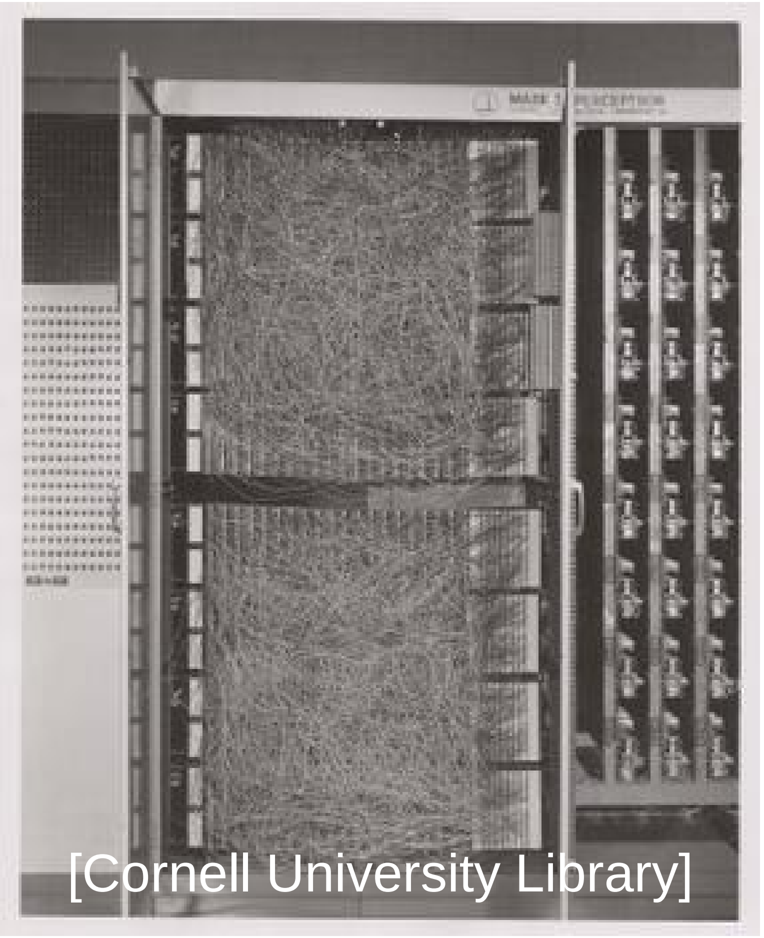
ICSI 30 Years of Innovation
October 5, 2018

Three Waves of Artificial Neural Networks

- 1950s/60s: Perceptrons (1 layer)
- 1980s/90s: Backpropagation, (2-3 layers)
- 2010s: Deep Neural Networks (3+ layers)

(Note, all ideas were developed much earlier than eventual popularity)

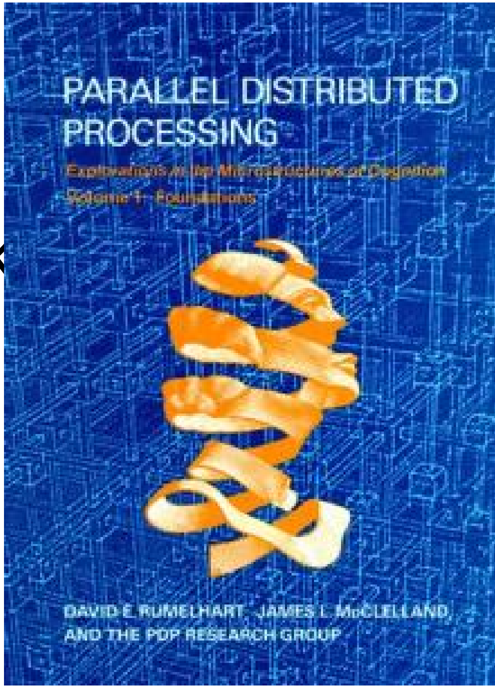
Single-Layer Perceptrons, 1950s-60s



- Single-layer perceptrons explored for image processing [Rosenblatt]
- Only a linearly separable classifier
  - Unable to capture interesting functions, e.g. XOR
- AI field moved from statistical to symbolic approaches in `70s-`80s

Multi-Layer Perceptrons and BackProp

- Influential PDP books published in 1986
- Two-layer backprop trained networks foun to work surprisingly well at many hard task
- Experts complained that results were unexplainable
- Training was extremely slow, so rush to build custom machines


PARALLEL DISTRIBUTED PROCESSING
Explorations in the Microstructure of Cognition
Volume 1: Foundations
DAVID E. RUMELHART, JAMES L. McCLELLAND, AND THE PDP RESEARCH GROUP

Ring Array Processor, (ICSI 1989)

(Nelson Morgan, Jim Beck, Phil Kohn, Jeff Bilmes)

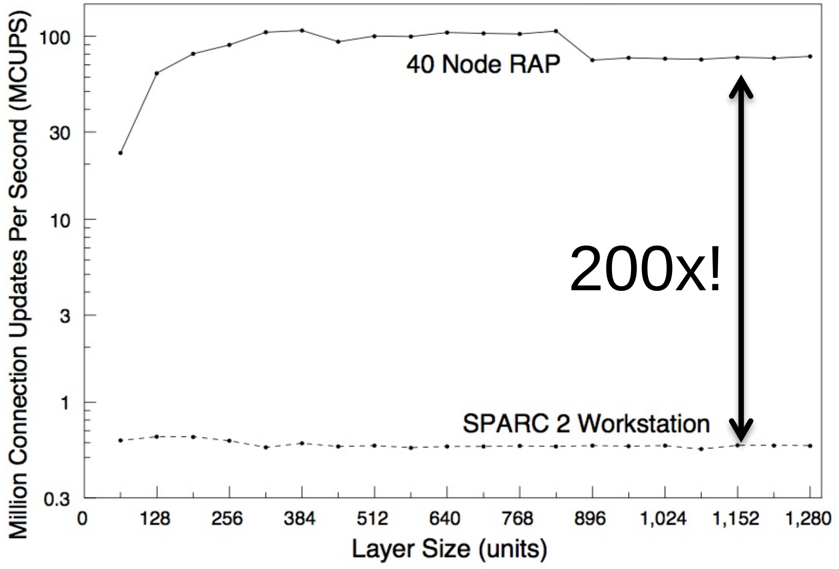
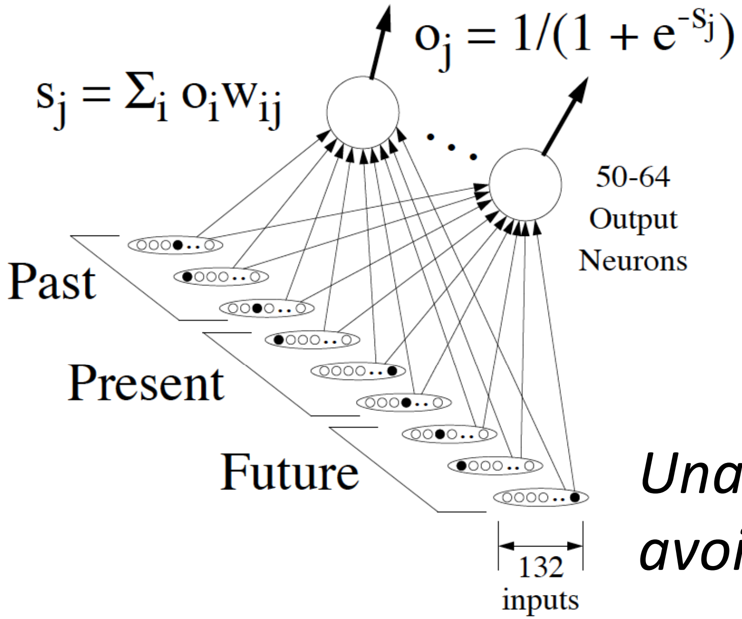


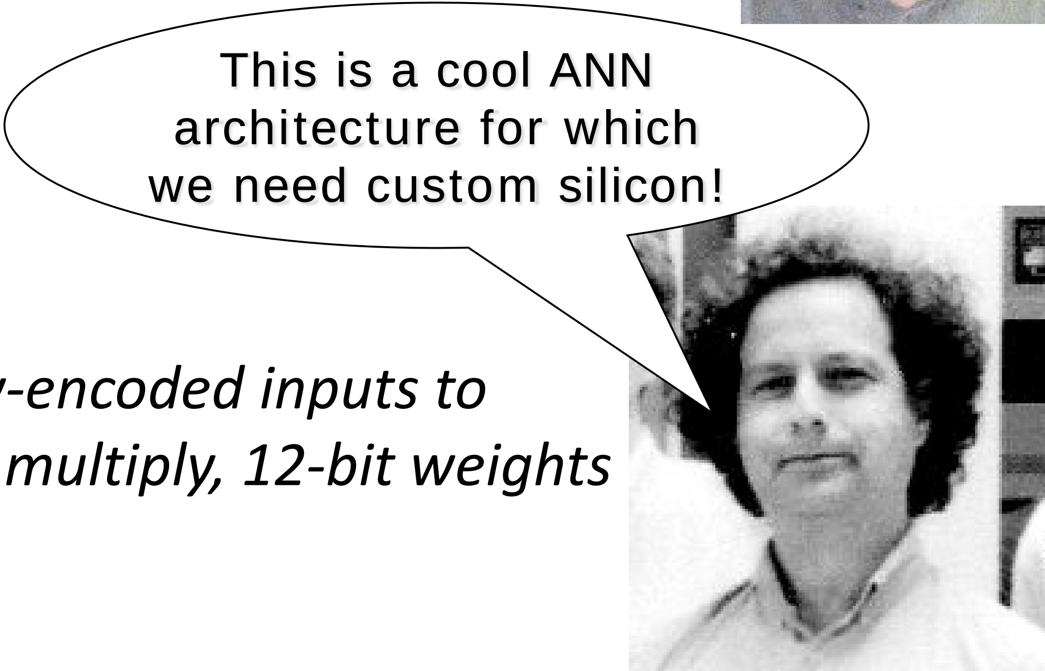
Fig. 3 RAP Performance for uniform layer size, one hidden layer.

- RAP Machine built for fast training of "big dumb" neural networks for speech recognition
- Ring of TMS320C30 floating-point DSPs
  - Each DSP providing 32MFLOPS (32-bit FP)
  - Four DSPs/board, up to 10 boards connected at once (>1GFLOP/s peak, 640MB DRAM)
  - Neural net training rate of >100MCUPS (million connection updates per second) on 10 boards
  - FPGA ring connection used for systolic all-all communication during training/inference
- Fast, flexible, but expensive
  - ~$100,000 each

Realization Group, ICSI, 1989

New naïve grad student joins Morgan's group to build custom ANN VLSI for speech training



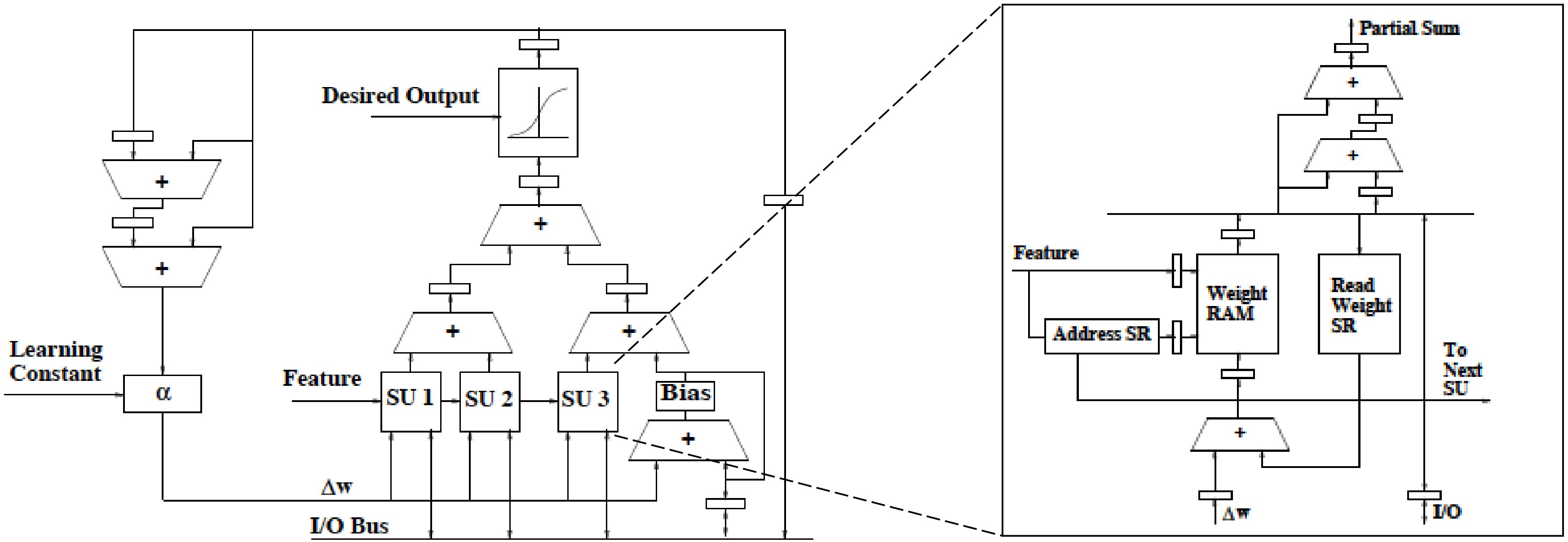

This is a cool ANN architecture for which we need custom silicon!

Unary-encoded inputs to avoid multiply, 12-bit weights

Training rule: $\Delta w_{ij} = -\alpha(o_j - d_j)\, o_i$

HiPNeT-1: (Highly Pipelined Network Trainer, 1990)

Krste Asanovic, Brian Kingsbury, Nelson Morgan, John Wawrzynek

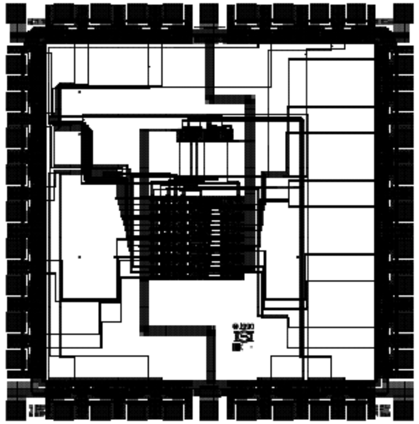


- Custom architecture for neural algorithm
- Ignores pipeline RAW hazards (net trains around them)
- Predicted 200MCUPS in 16mm^2 of 2µm CMOS running at 20MHz

The first few chips…

- MOSIS had a "TinyChip" program
  - $500 to fab a 2.2mmx2.2mm chip in 2µm CMOS

Sigmoid unit (Pawan Sinha)



8-bit datapath (Krste)

JTAG latches (Krste)

Multiplier (Brian)

24b Adder (Brian)

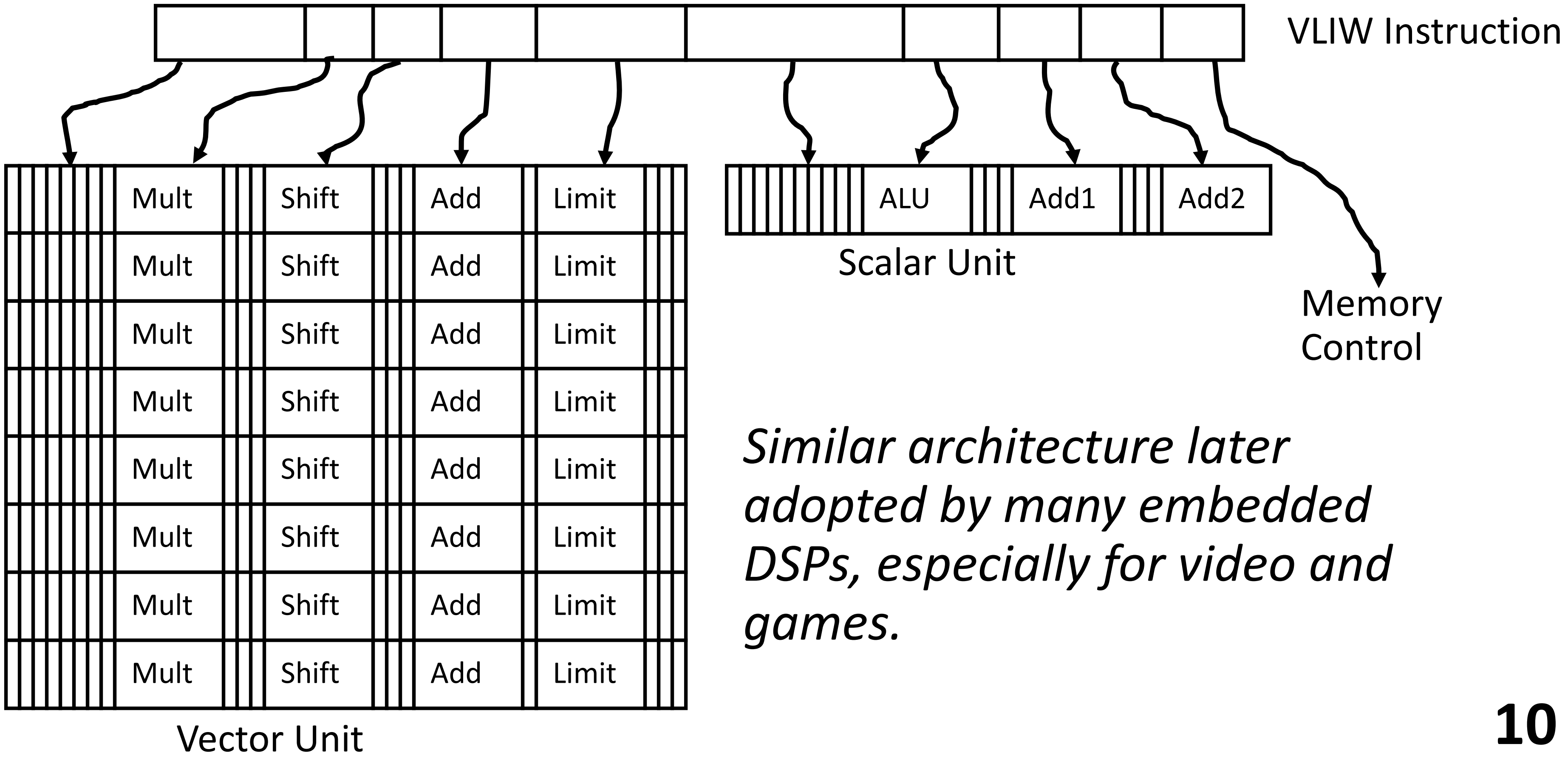
Regfile (Bertrand)

Meanwhile, back at the speech ranch…

There's this even cooler ANN architecture for which we need custom silicon!
And it doesn't look much like the last one. Can you build a different chip?

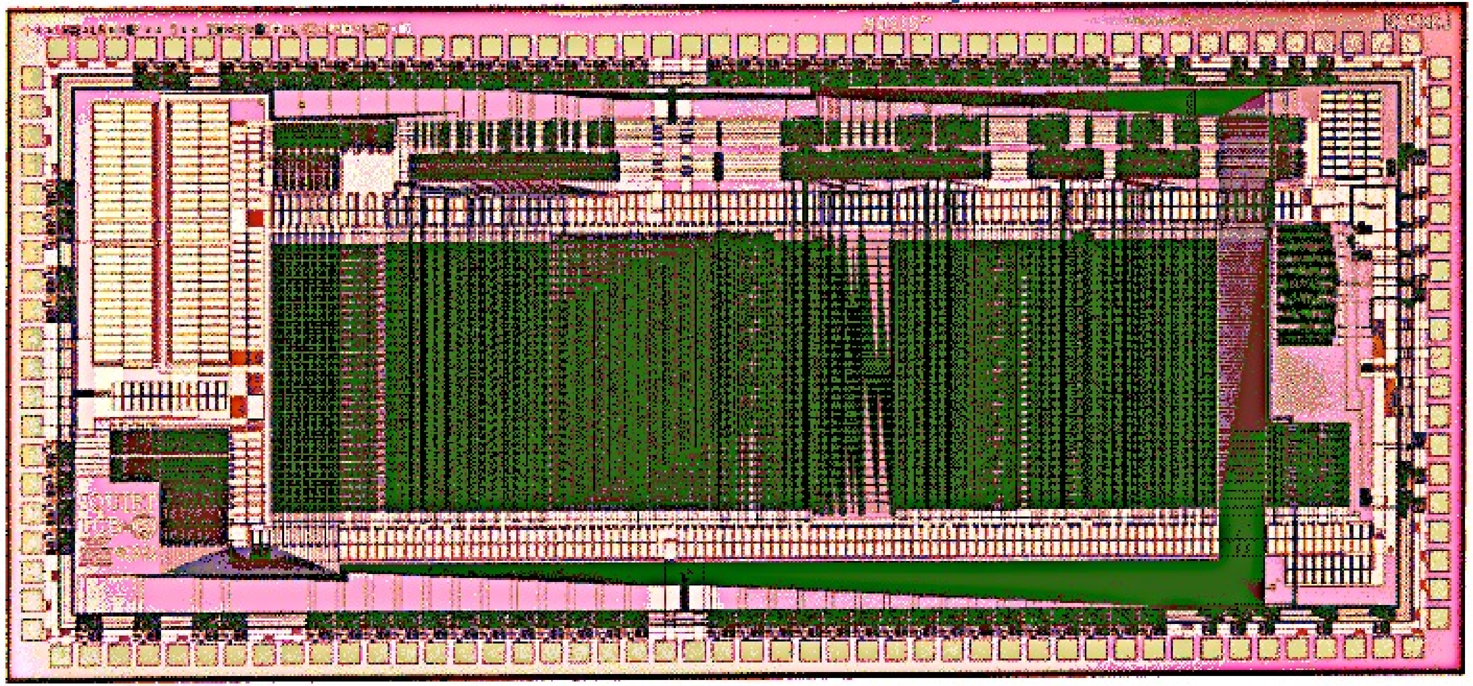
Time for a programmable architecture…

“Old” SPERT VLIW/SIMD Engine



Similar architecture later adopted by many embedded DSPs, especially for video and games.

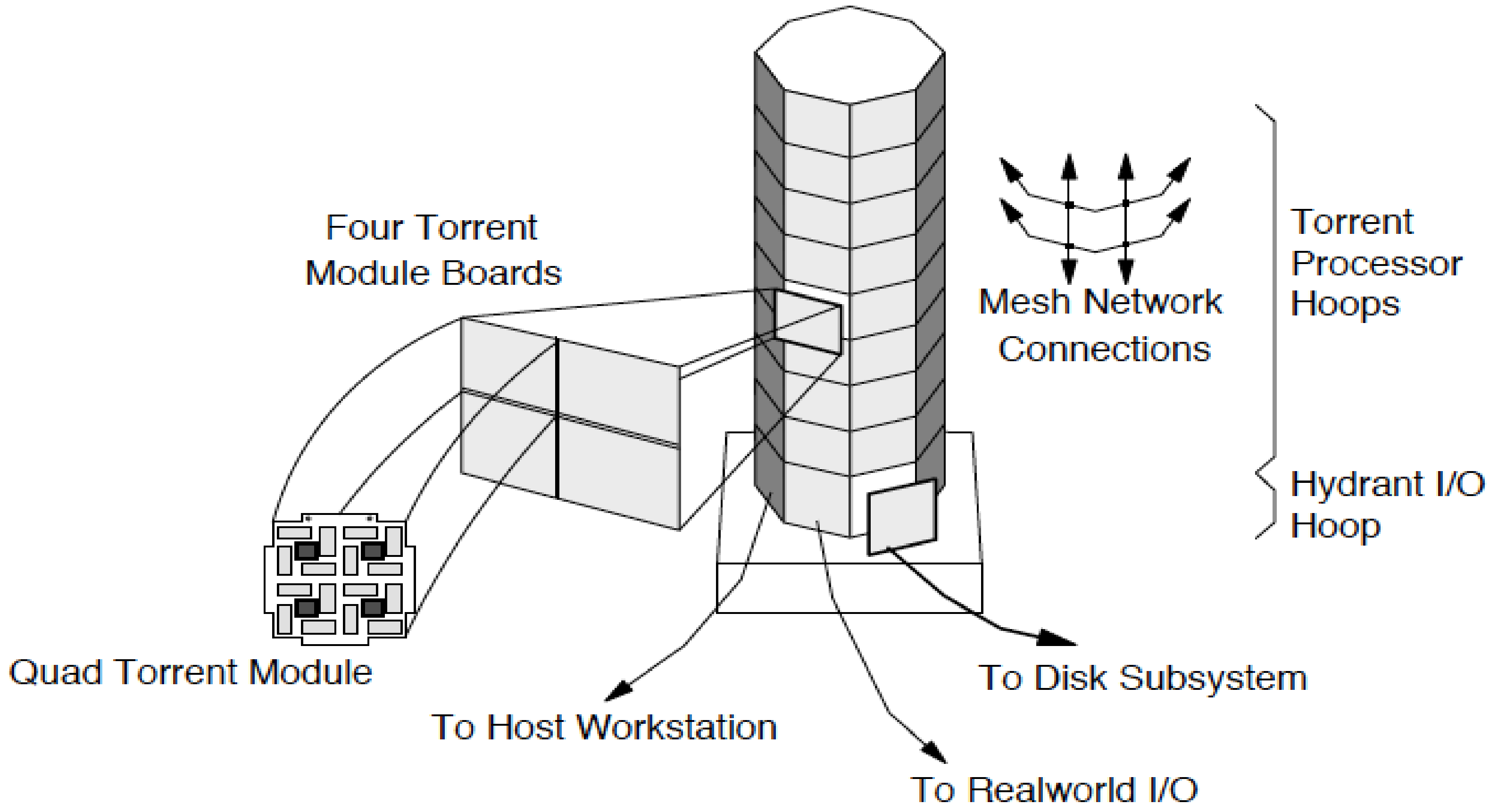
SQUIRT Test Chip, 1992



- 1.2µm CMOS, 2 metal layers
- 61,521 transistors, 8x4 mm^2, 400mW@5V, 50MHz
- 72-bit VLIW instruction word
- 16x32b register file, 24bx8b->32b multiplier, 32b ALU/shifter/clipper

CNS-1: Connectionist Network Supercomputer

(ICSI/UCB 1992-95)



CNS Research Group

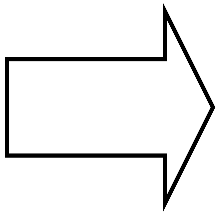
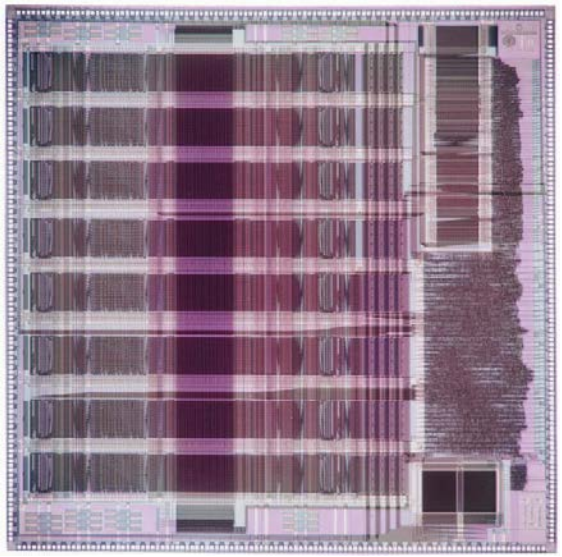
We abandoned SPERT VLIW

- VLIW means no upward compatibility
  - we wanted same ISA for CNS-1 to reuse software effort
- VLIW scalar compiler was complex
  - Simple VLIW hardware + complex VLIW compiler harder than complex RISC architecture + standard compiler
- Assembly code was tough to write
  - soon discovered this when writing test code and key loops
- VLIW format too rigid
  - hard to fit some operations into statically scheduled instruction slots (misaligned vector loads/stores, scatter/gathers)
- VLIW had too large an instruction-cache footprint
  - loop prologue/epilogue code plus unrolled loop body

Software, software, software,….

T0: First Vector Microprocessor (1995)

- Vector supercomputers (e.g., Crays) very successful in scientific computing, clean programming model

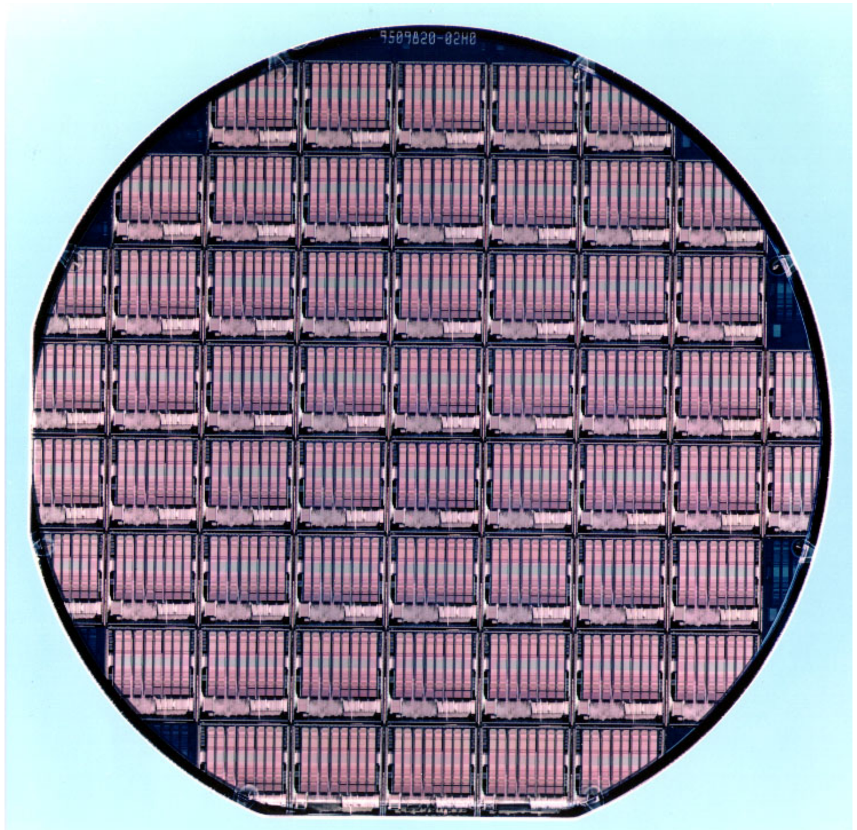



Add a vector coprocessor to a standard MIPS RISC scalar processor, all on one chip, for neural net training

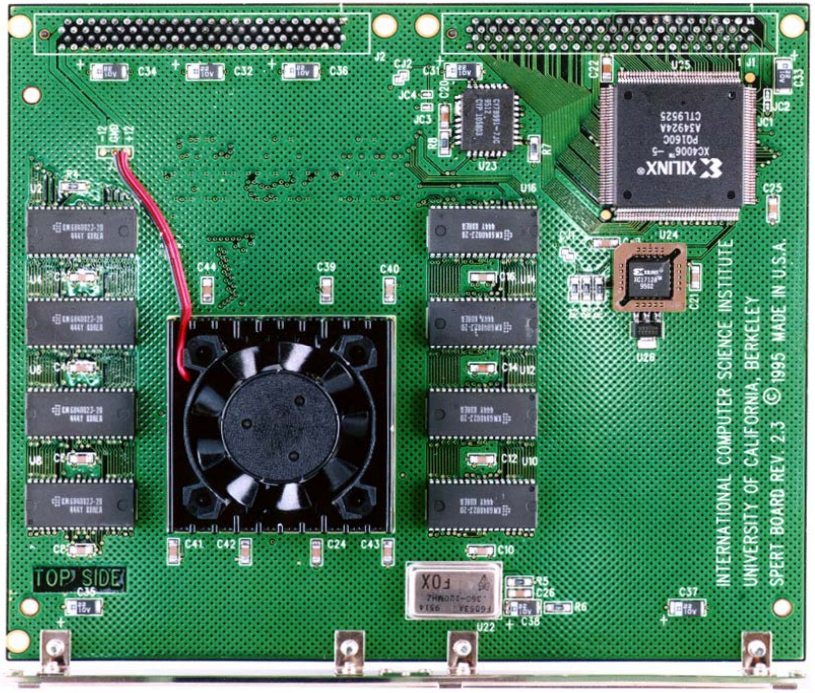
System Design Choices

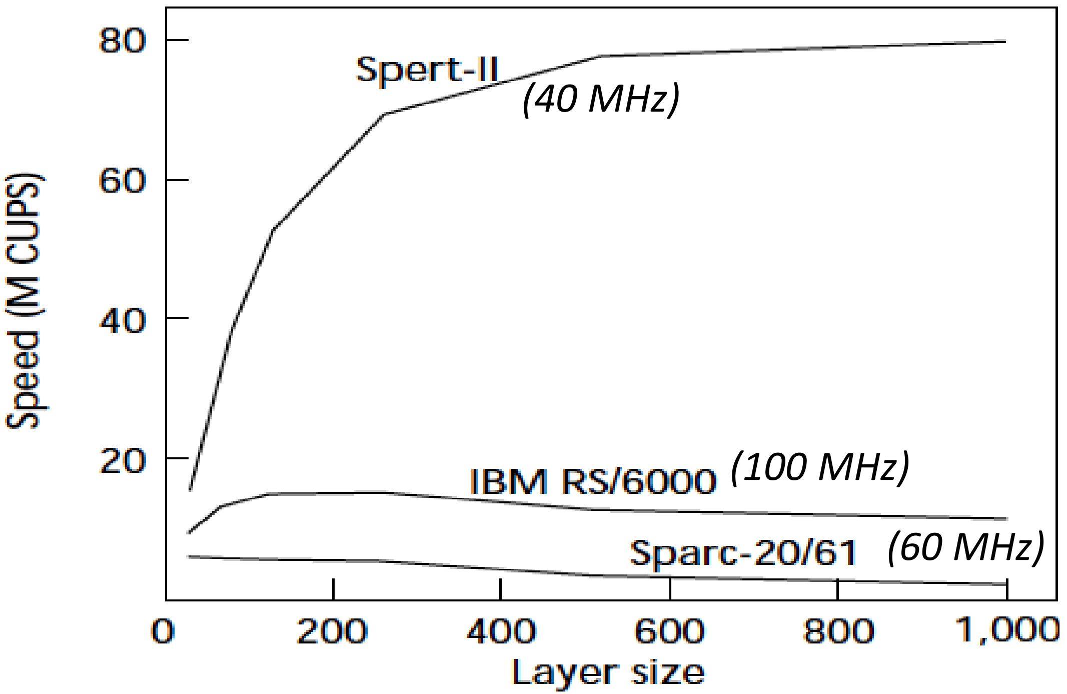
- Which standard RISC?
  - Considered SPARC, HP PA, PowerPC, and Alpha
  - Chose MIPS because: simplest, good software tools, Unix desktop workstations for development, and a 64-bit extension path
- Buy or build a MIPS core?
  - Commercial MIPS R3000 chips had coprocessor interface
  - No MIPS soft cores (no Verilog or synthesis tools yet)
  - Decided to roll our own
    - Vector coprocessor would have played havoc with caches
    - Coprocessor interface too inefficient
    - Commercial chip plus glue logic would blow our size and power budgets (to fit inside workstation)
    - Couldn’t simulate whole system in our environment

Brian, Krste, and a Torrent wafer



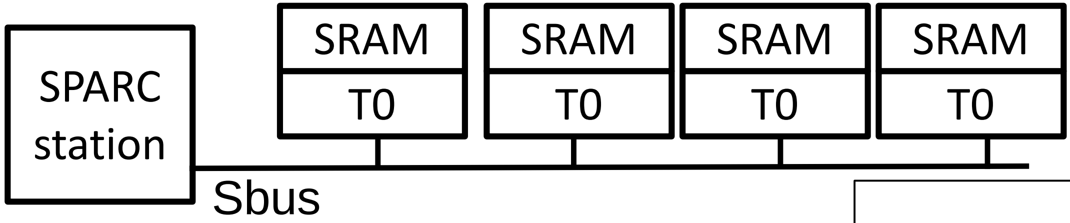
SPERT-II / T0 Vector Microprocessor (1995)


INTERNATIONAL COMPUTER SCIENCE INSTITUTE
UNIVERSITY OF CALIFORNIA, BERKELEY © 1995 MADE IN U.S.A.
SPERT BOARD REV. 2.3
TOP SIDE
XILINX



- Boards shipped to 9 international sites
- Used as production research platform for nine years
  - last time powered up for work in 2004!

TetraSpert (1997): faster training



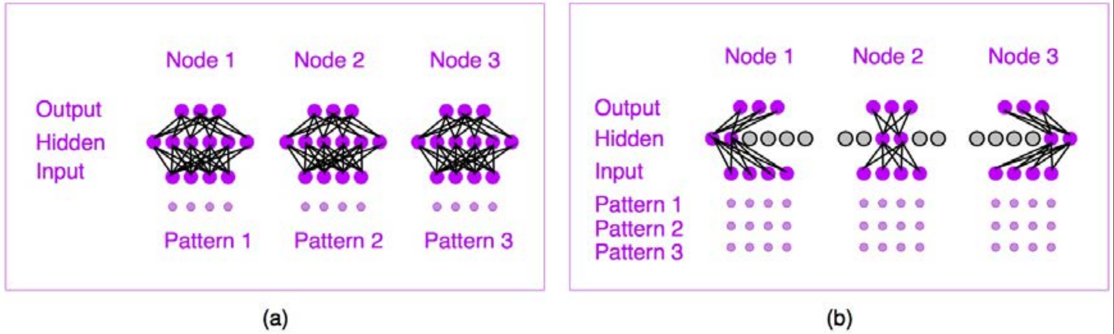


Figure 2: Pattern Parallel (a) and Network Parallel (b) distribution strategies.

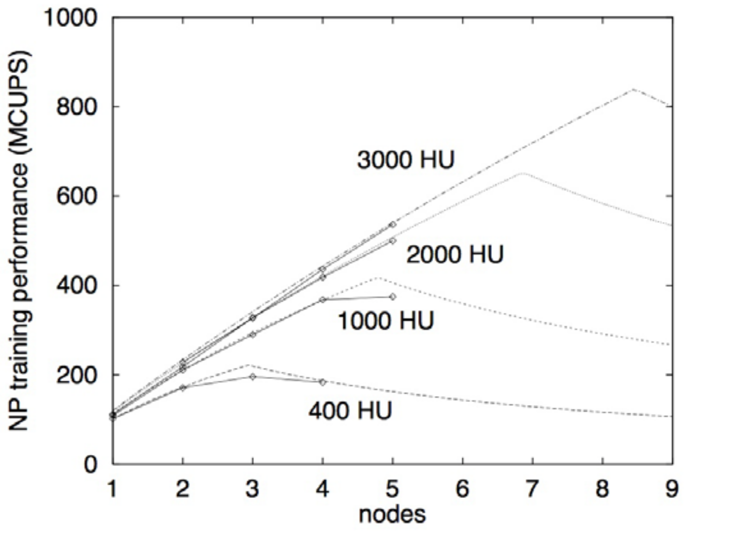
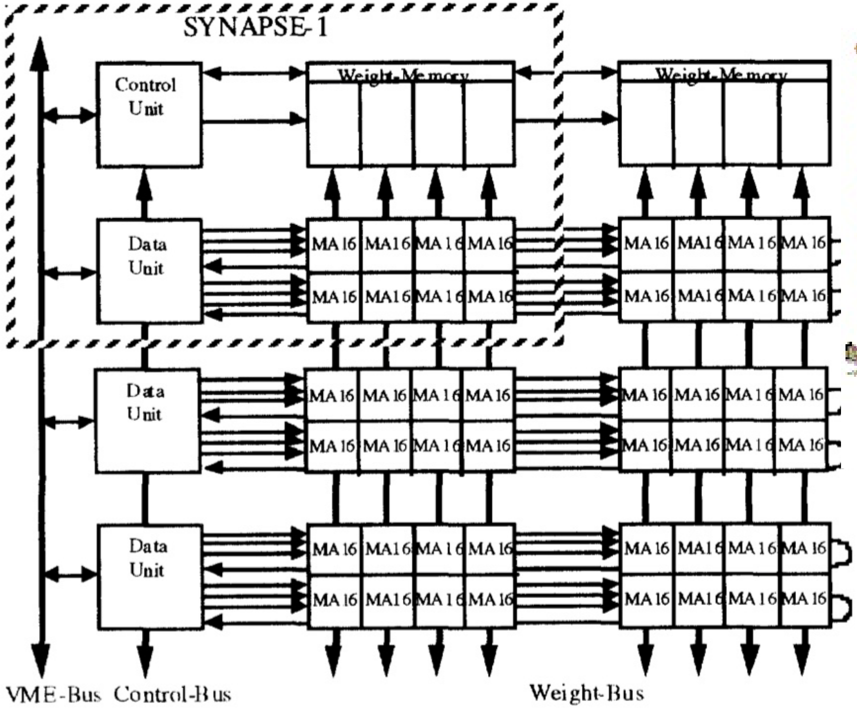


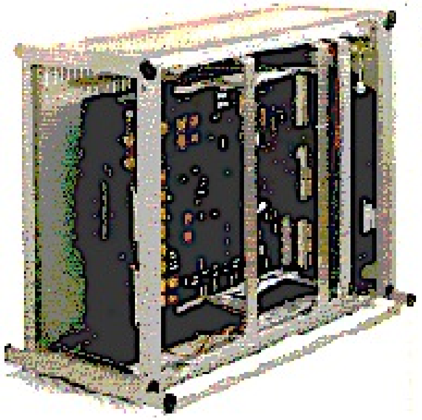
Figure 5: Multi-Spert NP training performance with optimized pattern p

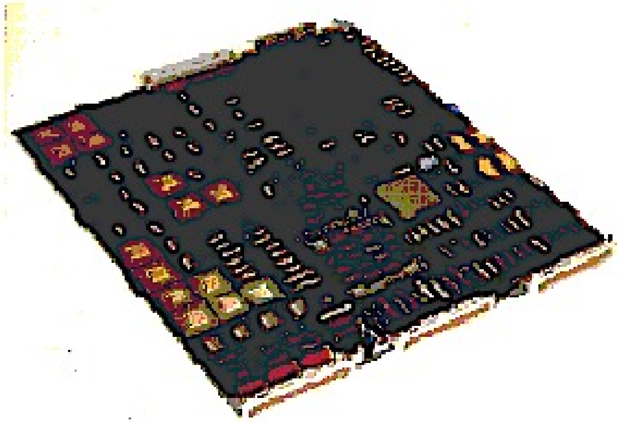
[Faerber, Asanovic, IEEE ICAAPP 1997]

Siemens SYNAPSE-1 (1992-5)

- Systolic matrix-multiply engine (16b*16b)
- Four levels of program control (68000s + microcode)

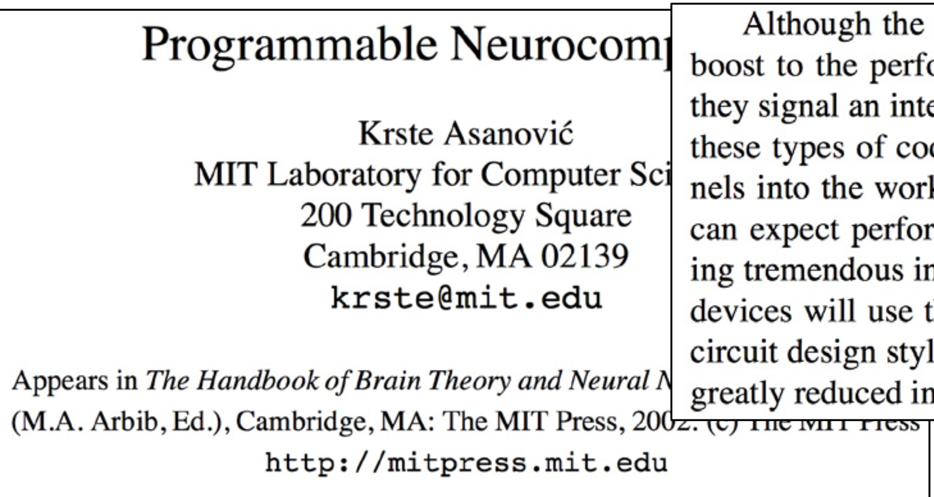



SYNAPSE-1


Control Unit

What Happened to '90s Neurocomputers?

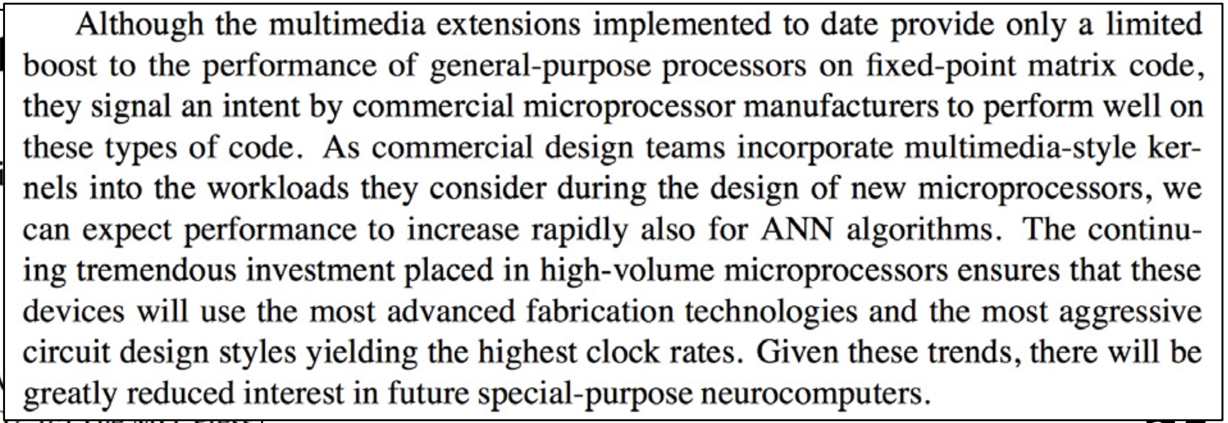
- Very small market
- Neural networks faded in popularity
  - some kept working on them
- Moore's Law scaling favored general-purpose processors
- In 1996, Intel introduced MMX



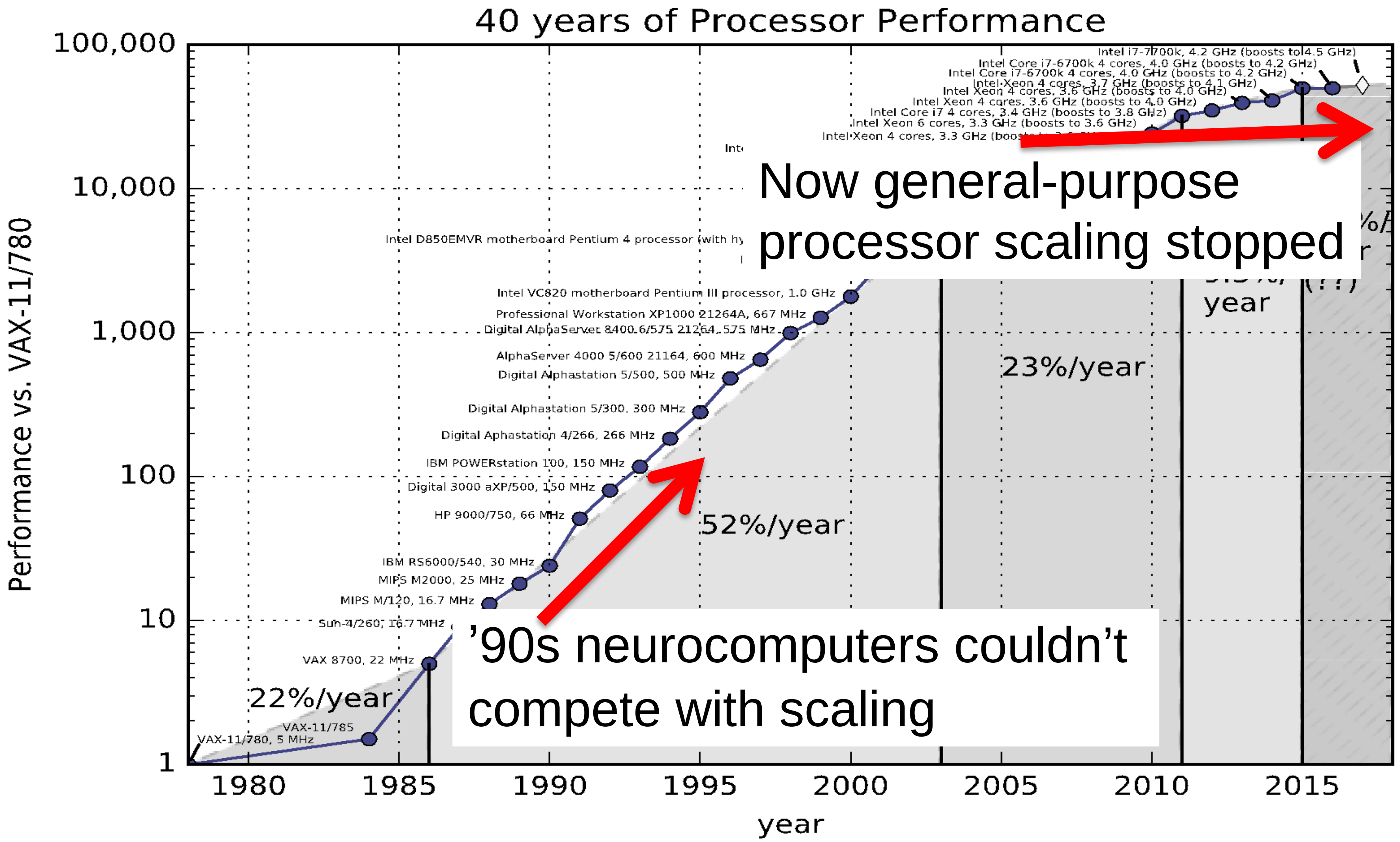
Programmable Neurocomp

Krste Asanović
MIT Laboratory for Computer Sci
200 Technology Square
Cambridge, MA 02139
krste@mit.edu

Appears in *The Handbook of Brain Theory and Neural N*
(M.A. Arbib, Ed.), Cambridge, MA: The MIT Press, 2002. (c) The MIT Press
http://mitpress.mit.edu



Although the multimedia extensions implemented to date provide only a limited boost to the performance of general-purpose processors on fixed-point matrix code, they signal an intent by commercial microprocessor manufacturers to perform well on these types of code. As commercial design teams incorporate multimedia-style kernels into the workloads they consider during the design of new microprocessors, we can expect performance to increase rapidly also for ANN algorithms. The continuing tremendous investment placed in high-volume microprocessors ensures that these devices will use the most advanced fabrication technologies and the most aggressive circuit design styles yielding the highest clock rates. Given these trends, there will be greatly reduced interest in future special-purpose neurocomputers.



Now general-purpose processor scaling stopped

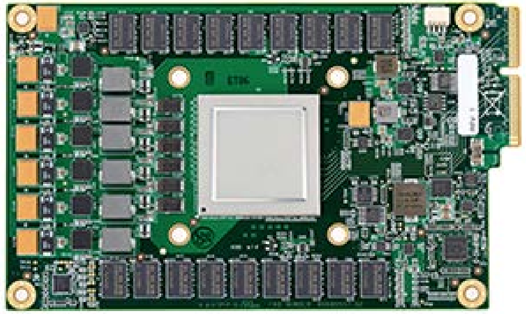
’90s neurocomputers couldn’t compete with scaling

[Hennessy & Patterson, 2017]

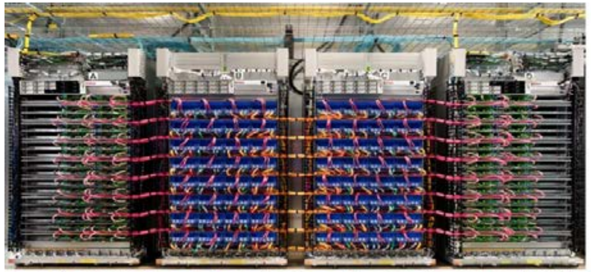
General-Purpose GPUs (GP-GPUs)

- In 2006, Nvidia introduced GeForce 8800 GPU supporting a new programming language: CUDA
  - "Compute Unified Device Architecture"
  - Subsequently, broader industry pushing for OpenCL, a vendor-neutral version of same ideas.
- Idea: Take advantage of GPU computational performance and memory bandwidth to accelerate some kernels for general-purpose computing
- Attached processor model: Host CPU issues data-parallel kernels to GP-GPU for execution
- Over time, became the fastest standard way of performing neural network training

Extensive Efforts in Custom AI Chips

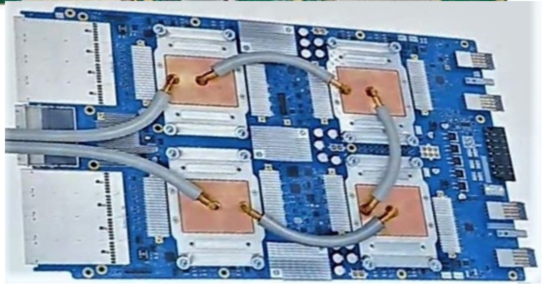


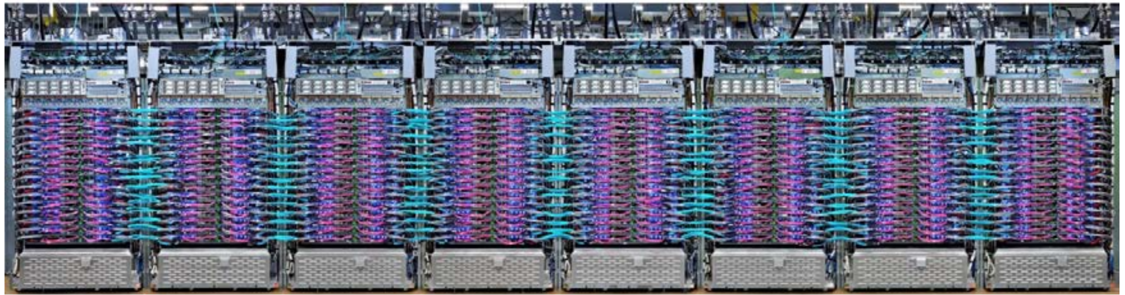
Google TPU



TPUv2

TPUv3

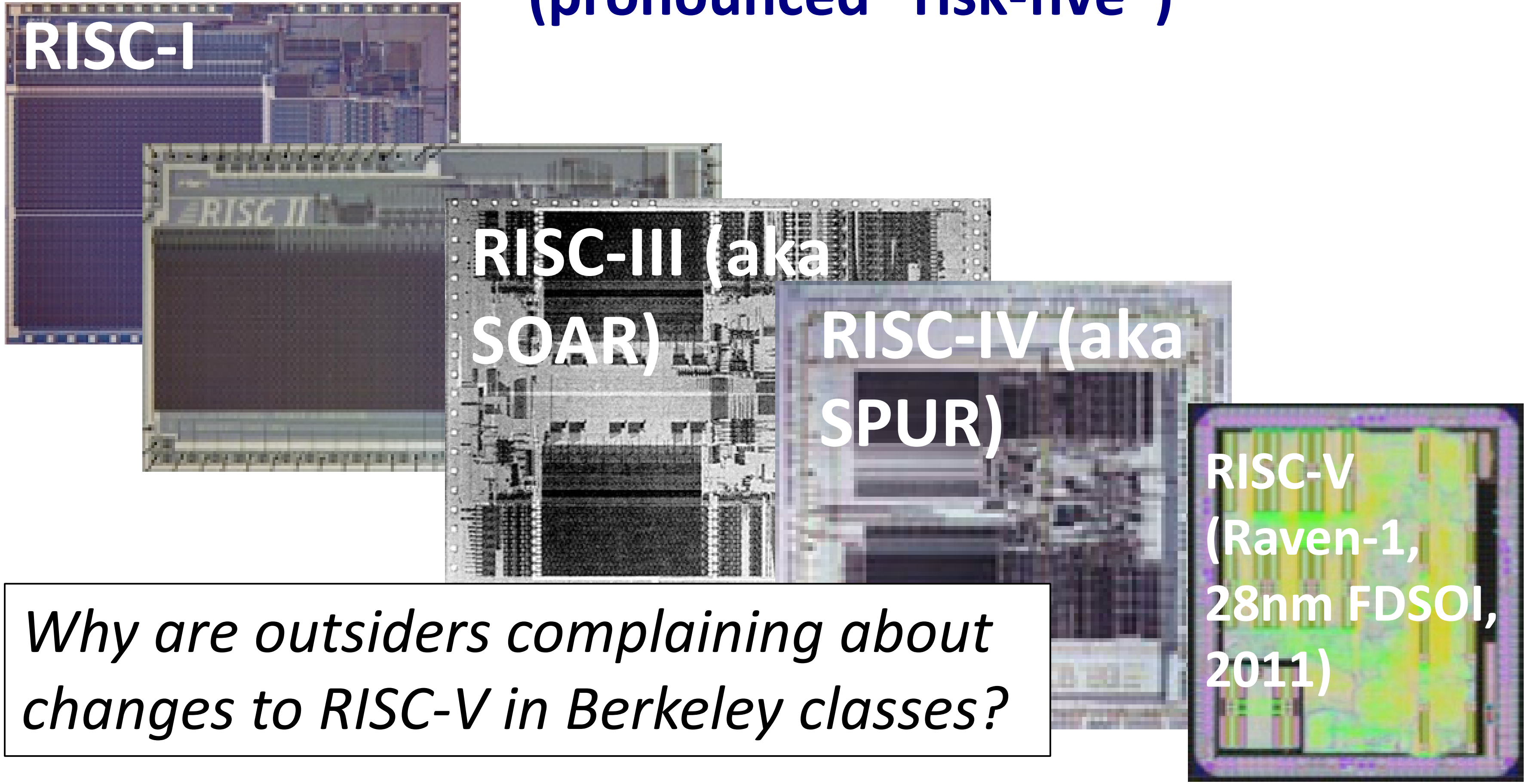




“Let’s fill a reticle with reduced-precision vector processing, add high-bandwidth local memory, and attach multiple to a server to accelerate neural network training.”

~100 AI Hardware Startup Pitches

RISC-V
(pronounced “risk-five”)

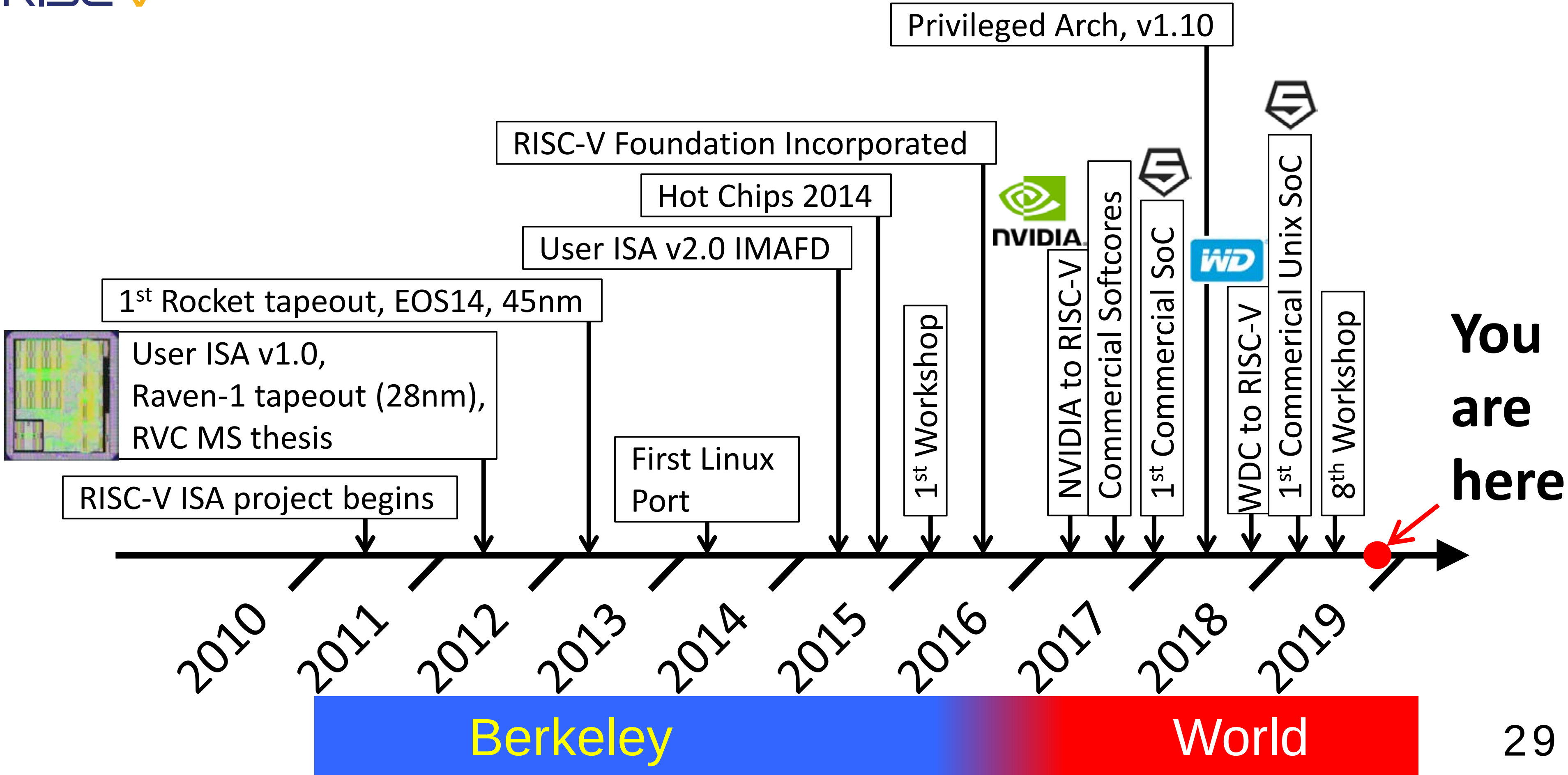


Why are outsiders complaining about changes to RISC-V in Berkeley classes?

What is RISC-V?

- A high-quality, license-free, royalty-free RISC ISA specification originally from UC Berkeley
- Standard maintained by non-profit RISC-V Foundation
- Suitable for all types of computing system, microcontrollers to supercomputers
- Numerous proprietary and open-source cores
- Experiencing rapid uptake in industry and academia
- Supported by growing shared software ecosystem
- A work in progress…

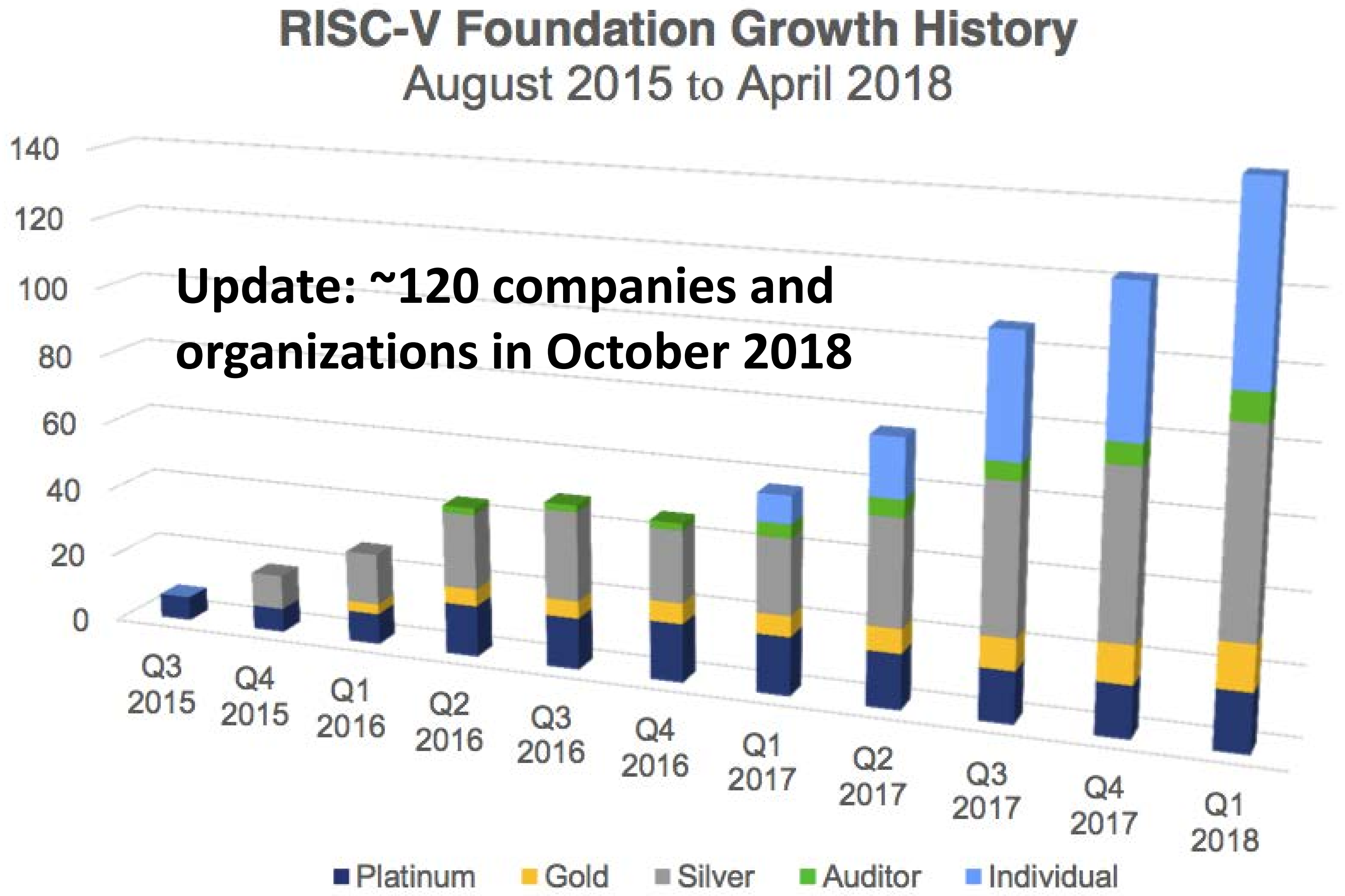
RISC-V Timeline



Berkeley Architecture Research
antmicro EMBEDDED SYSTEMS
bluespec
cortus
SKY
Rambus Cryptography Research
Mellanox TECHNOLOGIES
NXP
DRAPER
Google
Mentor A Siemens Business
Micron
Microsemi
HUAWEI
NVIDIA.
SAMSUNG
IBM
Western Digital.
SiFive
QUALCOMM
LATTICE SEMICONDUCTOR

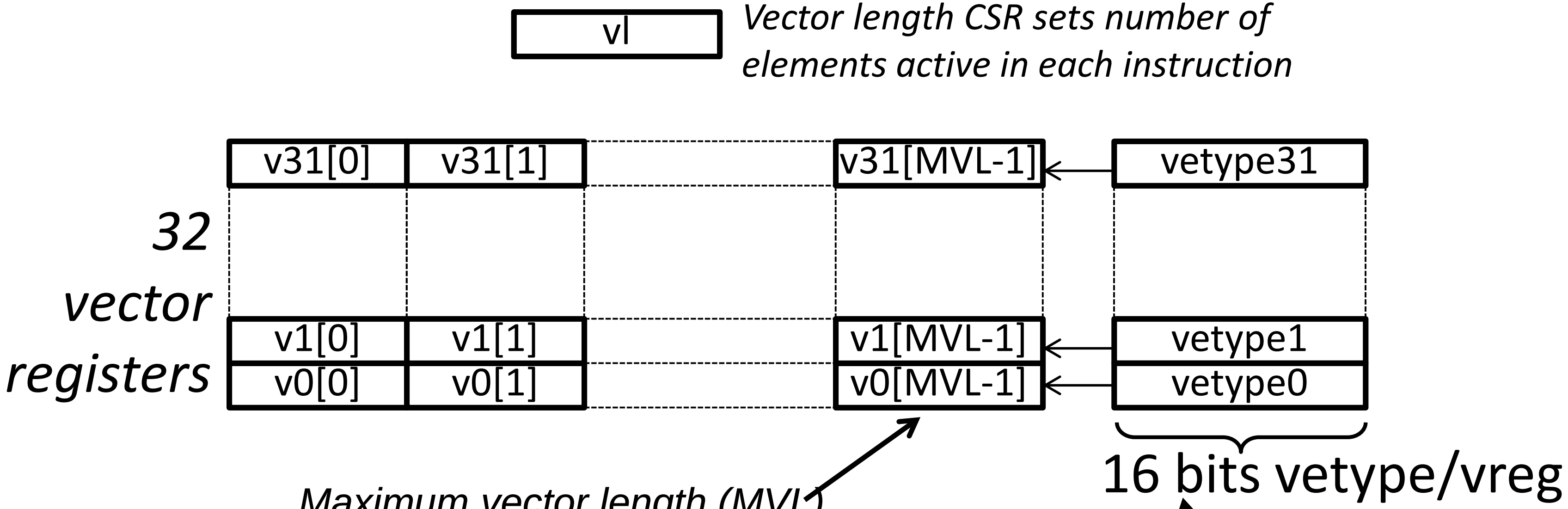
RISC-V Foundation: 180+ Members

ESPRESSIF
MEDIATEK
ANDES TECHNOLOGY
LAUTERBACH DEVELOPMENT TOOLS
runtime.io
ICT
BAE SYSTEMS
Esperanto Technologies
VeriSilicon
inside secure DRIVING TRUST
GRAY RESEARCH
IDT
INDIAN INSTITUTE OF TECHNOLOGY MADRAS
ISTUARY INNOVATION GROUP
中科院计算所 INSTITUTE OF COMPUTING TECHNOLOGY, CAS
NUS National University of Singapore
谨晟电子科技 WWW.GLORYSPACE.CN
GOWIN PROGRAMMING FOR FUTURE
GRIDOOG "Eyes on the Grid, Eyes on the Buck"
SIEMENS
INTRINSIX
ETHzürich
MIT CSAIL
SEGGER It simply works!
ASHLING THE DEVELOPMENT SYSTEMS COMPANY
codasip
Roa Logic Silicon Proven IP for FPGA and ASIC
GREENWAVES TECHNOLOGIES
SH CONSULTING
MINIMA PROCESSOR
Rumble Development
TRINAMIC
XtremeEDA
VectorBlox embedded supercomputing
UBILITE Hello IoT World
SEAGATE
EMDALO TECHNOLOGIES
EMBECOSM®
SECURE RF Securing the Internet of Things®
BERKELEY LAB
Syntacore™ Custom cores and tools
DOVER MICROSYSTEMS
lowRISC
TechanaLye
imperas
优矽科技 UC TECH IP
Technolution
ultrasoc
csem
PRINCETON UNIVERSITY
Blockstream
dxcorr
GLOBALFOUNDRIES®
expresslogic



Update: ~120 companies and organizations in October 2018

RISC-V Vector Extension Overview



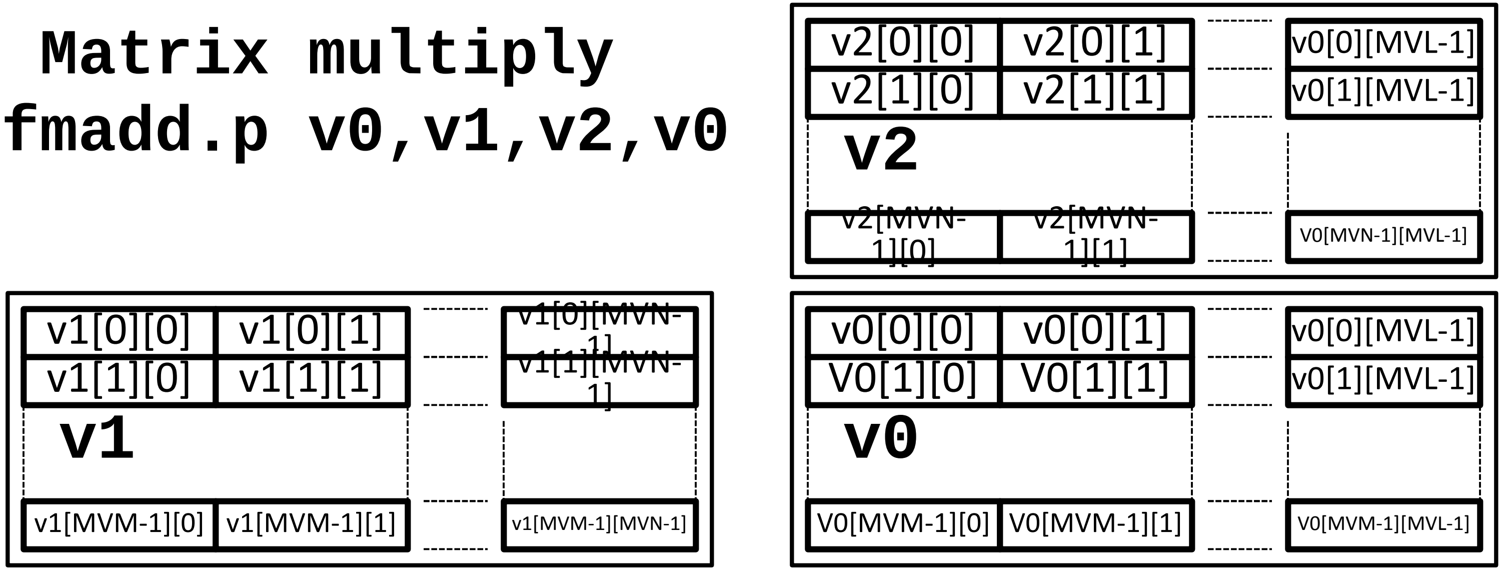
Vector length CSR sets number of elements active in each instruction

Maximum vector length (MVL) depends on implementation, number of vector registers used, and type of each element.

Vetype sets type of element in each vector register (e.g., 32-bit integer, 16-bit floating-point)

RISC-V 2-D Vector Extension

```
# Matrix multiply
vfmadd.p v0,v1,v2,v0
```



- Vector registers configured as 2D matrices
- Single instructions for matrix multiply/ convolutions

RISC-V for AI Accelerators

- RISC-V designed originally as basis for custom accelerators
- Simplify software by using one simple base ISA on all cores
  - Where you need high-performance Unix-capable core to run operating system, build a superscalar OoO core
  - Where wanted VLIW for microcode scheduling, build wide in-order superscalar
  - Where wanted low-precision SIMD, use standard vector extensions
  - Where want to take advantage of 2D optimizations (e.g., systolic matrix multiply, convolution), use 2D vector extensions (in progress)
  - Secret-sauce weight compression/number format? add custom extensions!
  - Where need interrupt-responsive I/O management core, build embedded core

RISC-V

- Same memory model, synchronization primitives, compiler tool flow (C-struct packing), debug, tracing,...