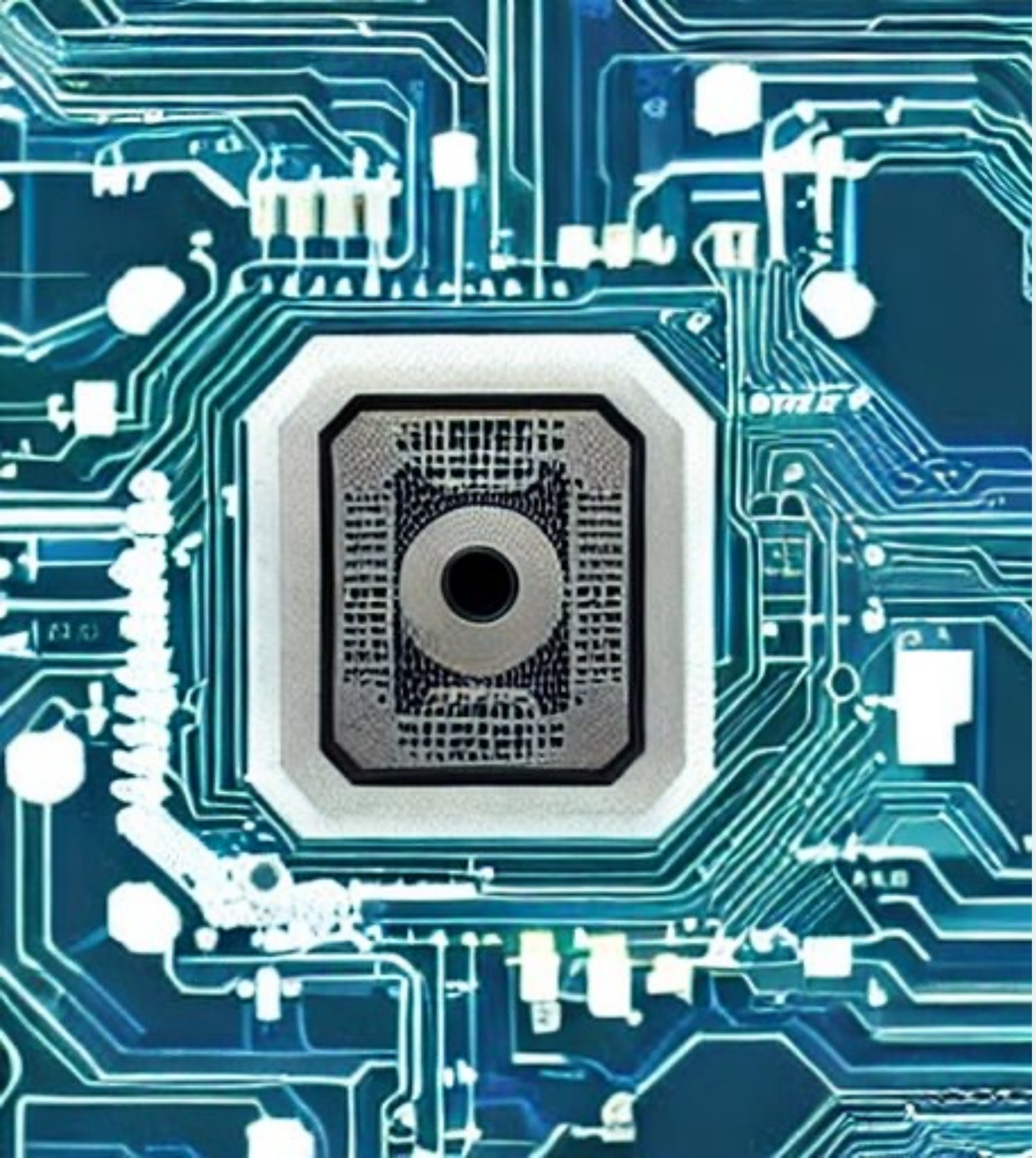


A Systematization of Knowledge study on

Hardware Specialization for AI/ML

Alenkruth Krishnan Murali (jht9sy)

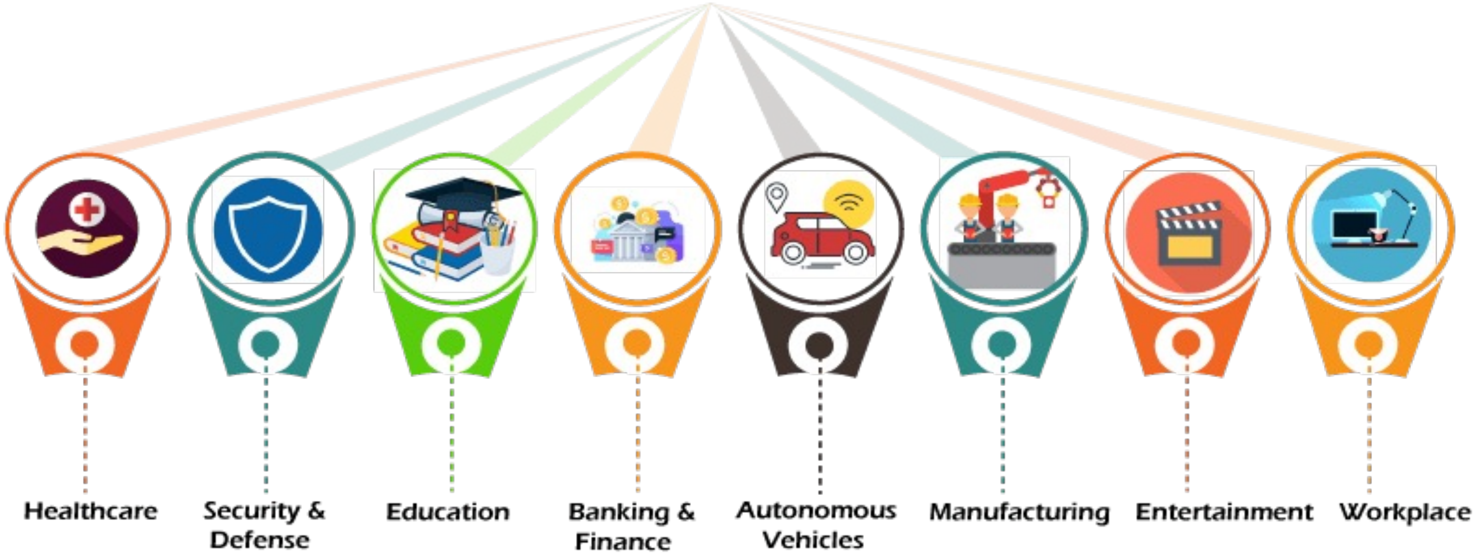
Khyati Kiyawat (vyn9mp)

Sabiha Tajdari (jvx2tt)

Outline

- Applications of AI/ML
- Basics of AI/ML
- Types of computations required
- Need for Specialization
- Current Accelerator Space
- Data Center AI Accelerators
- Edge AI Accelerators
- Past, Present and Future
- References

Applications of ML and AI



- NLP - Speech and Text (Voice assistants)
- Image recognition - Computer Vision
- Recommendation Systems
- Medical Diagnosis
- Placement and Routing (floorplanning)
- Code completion
- Attack detection and generation
- Accelerator design?

Basic CNN example

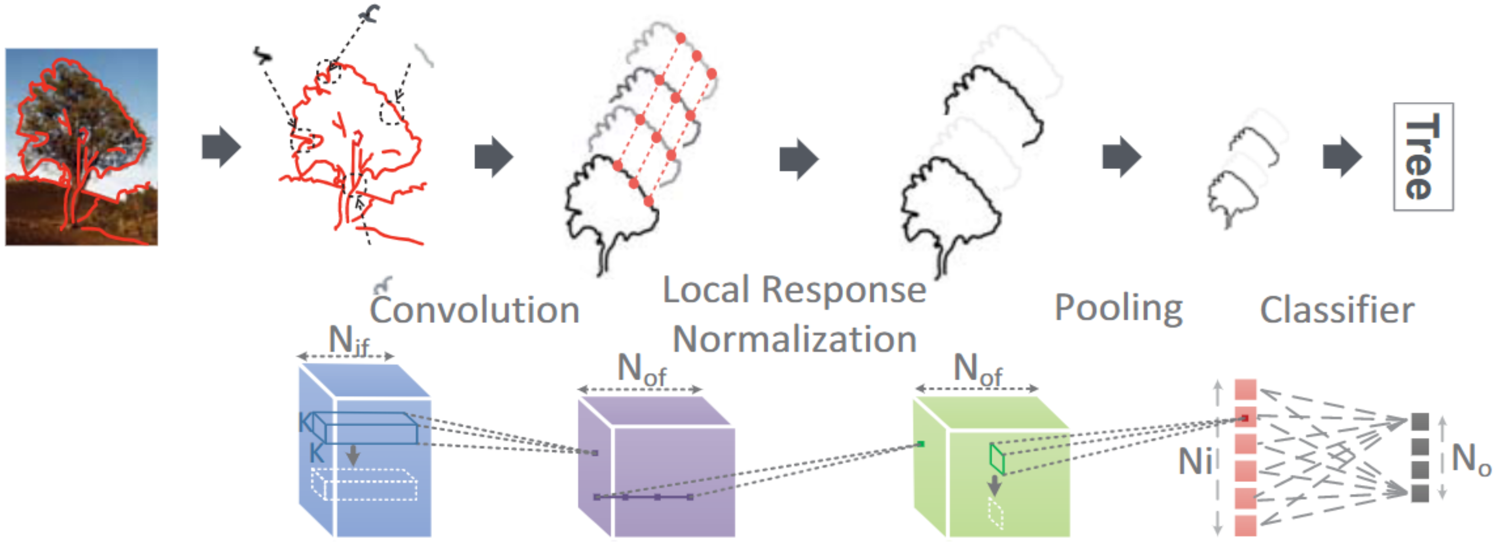


Figure 1: *The four layer types found in CNNs and DNNs.*

DaDianNao: A Machine-Learning Supercomputer[2015], Chen Y et al.

Basic CNN example

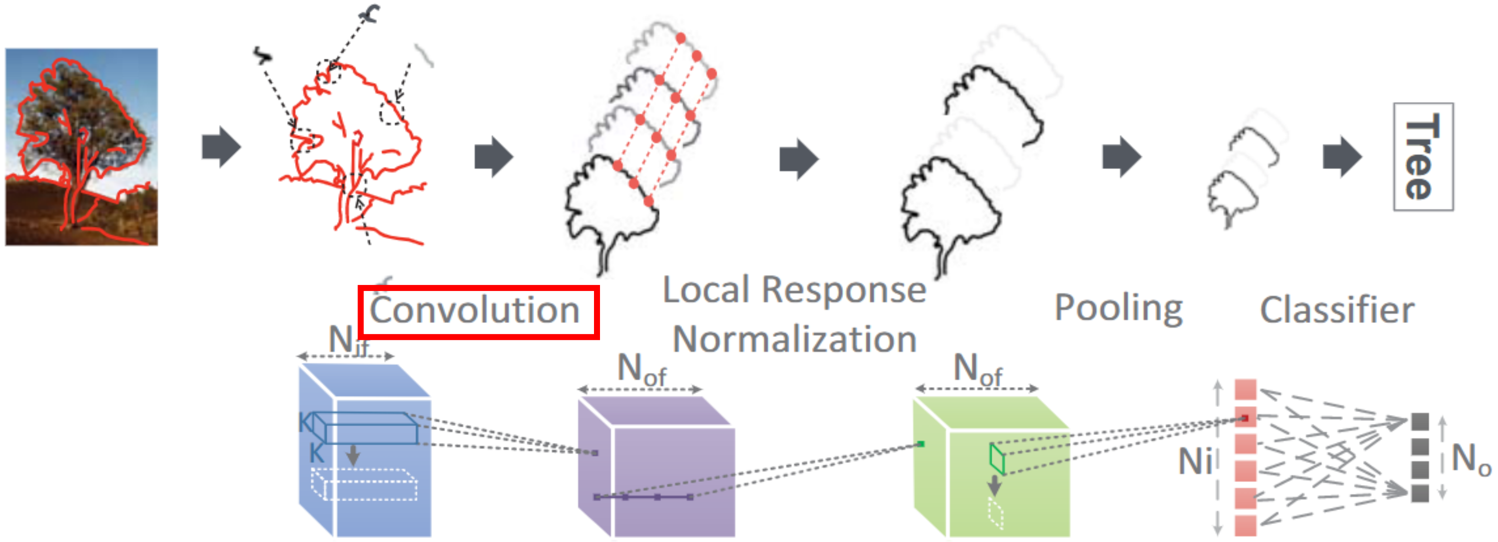


Figure 1: *The four layer types found in CNNs and DNNs.*

$$\boxed{out(x,y)^{f_o} = \sum_{f_i=0}^{N_{if}} \sum_{k_x=0}^{K_x} \sum_{k_y=0}^{K_y} w_{f_i,f_o}(k_x,k_y) * in(x+k_x, y+k_y)^{f_i}}$$

DaDianNao: A Machine-Learning Supercomputer[2015], Chen Y et al.

Basic CNN example

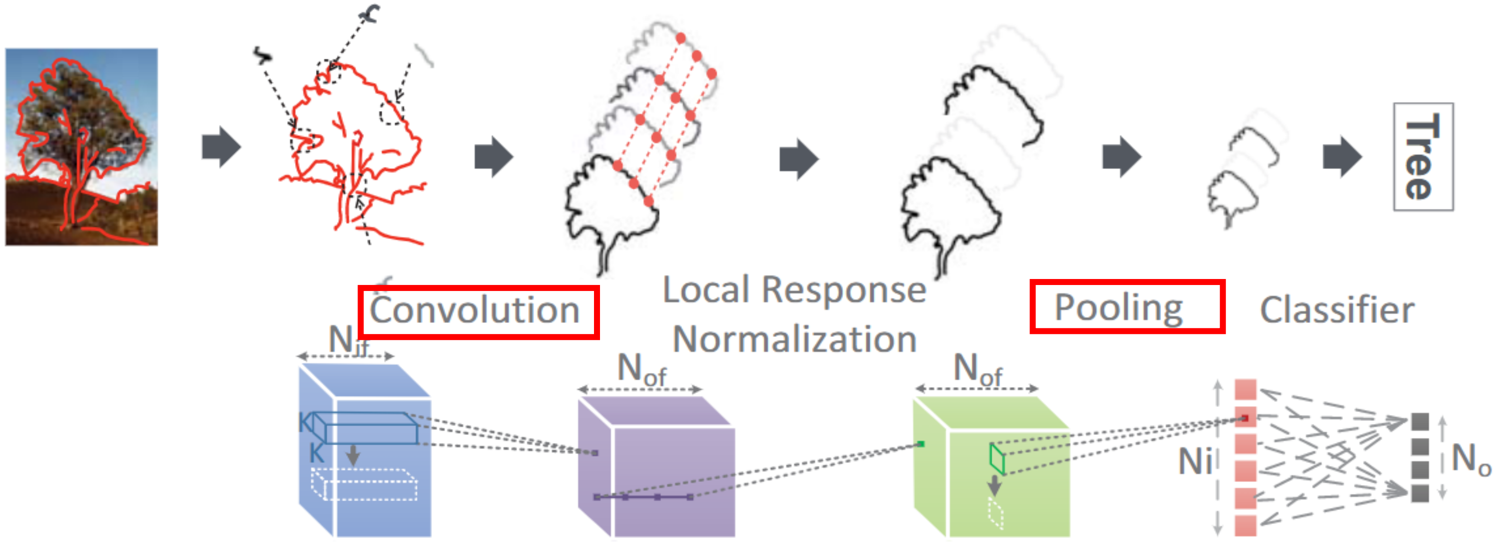


Figure 1: *The four layer types found in CNNs and DNNs.*

$$out(x,y)^{f_o} = \sum_{f_i=0}^{N_{if}} \sum_{k_x=0}^{K_x} \sum_{k_y=0}^{K_y} w_{f_i,f_o}(k_x,k_y) * in(x+k_x, y+k_y)^{f_i}$$

$$out(x,y)^{f} = \max_{0 \le k_x \le K_x, 0 \le k_y \le K_y} in(x+k_x, y+k_y)^{f}$$

DaDianNao: A Machine-Learning Supercomputer[2015], Chen Y et al.

Basic CNN example

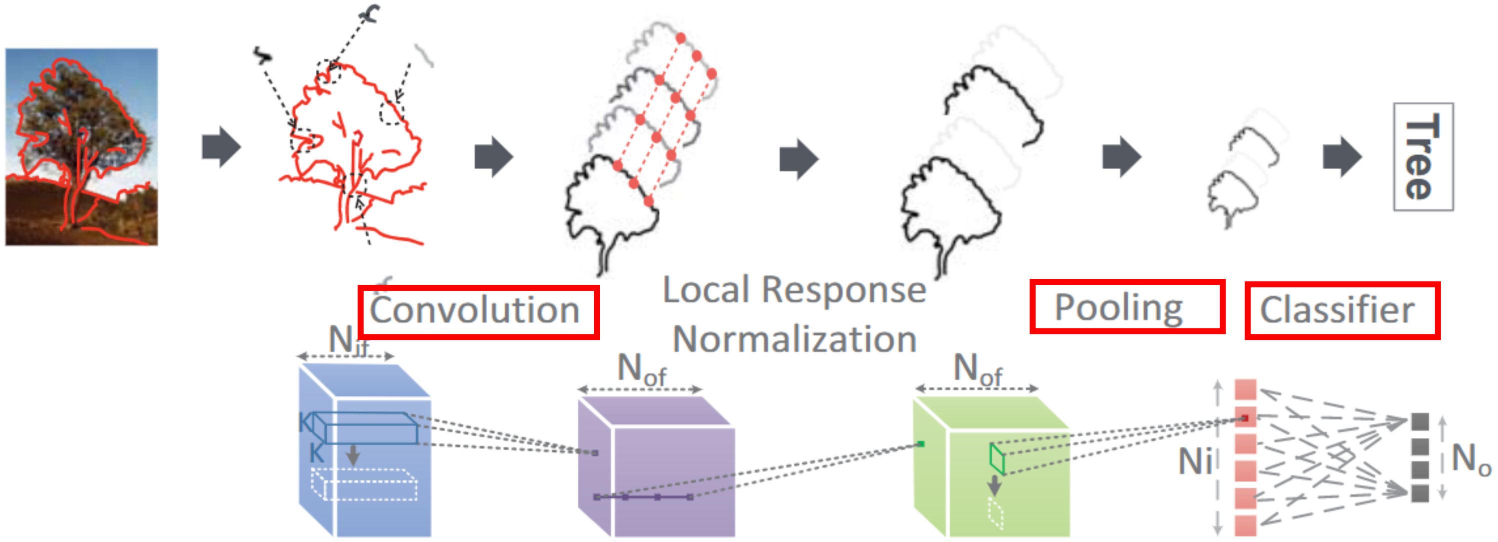


Figure 1: *The four layer types found in CNNs and DNNs.*

$$\boxed{out(j) = t\left(\sum_{i=0}^{N_i} w_{ij} * in(i)\right)}$$

$$\boxed{out(x,y)^{f_o} = \sum_{f_i=0}^{N_{if}} \sum_{k_x=0}^{K_x} \sum_{k_y=0}^{K_y} w_{f_i,f_o}(k_x,k_y) * in(x+k_x, y+k_y)^{f_i}}$$

$$\boxed{out(x,y)^f = \max_{0 \le k_x \le K_x, 0 \le k_y \le K_y} in(x+k_x, y+k_y)^f}$$

DaDianNao: A Machine-Learning Supercomputer[2015], Chen Y et al.

Typical computations for AI/ML

Typical computations for AI/ML

- Requires MAC operation for
  - feature extraction and classification
  - both of which are highly parallelizable

Typical computations for AI/ML

- Requires MAC operation for
  - feature extraction and classification
  - both of which are highly parallelizable
- Table along side shows types of instructions and operations required for AI accelerators [1].

Table 1: AI functions supported by the NNPA Instruction.

Operation Class	Function name
Query op	QAF
Elementwise ops	ADD, SUB, MUL, DIV, MIN, MAX
Activation ops	LOG, EXP, RELU, TANH, SIGMOID
RNN activation ops	LSTMACT, GRUACT
Normalization ops	SOFTMAX, BATCH NORMALIZATION
Pooling ops	AVERAGEPOOL2D, MAXPOOL2D
Systolic ops	FUSED CONVOLUTION,MATMUL-OP, MATMUL-BROADCAST-OP

1. AI Accelerator on IBM Telum Processor [2022], Lichtenau et el.

Typical computations for AI/ML

- Requires MAC operation for
  - feature extraction and classification
  - both of which are highly parallelizable
- Table along side shows types of instructions and operations required for AI accelerators [1].

Table 1: AI functions supported by the NNPA Instruction.

Operation Class	Function name
Query op	QAF
Elementwise ops	ADD, SUB, MUL, DIV, MIN, MAX
Activation ops	LOG, EXP, RELU, TANH, SIGMOID
RNN activation ops	LSTMACT, GRUACT
Normalization ops	SOFTMAX, BATCH NORMALIZATION
Pooling ops	AVERAGEPOOL2D, MAXPOOL2D
Systolic ops	FUSED CONVOLUTION,MATMUL-OP, MATMUL-BROADCAST-OP

These operations are not computationally complex but data intensive

1. AI Accelerator on IBM Telum Processor [2022], Lichtenau et el.

Need for HW specialization

Need for HW specialization

- History of processors
  - Leveraging the advancements in technology nodes (Moore’s law)
  - Single core
    - Thermal issues, stopped gaining from single cores
  - Super-scalar and Multi-core processor
    - Dark Silicon issue
  - Heterogenous processing to exploit performance based on workloads

Need for HW specialization

- History of processors
  - Leveraging the advancements in technology nodes (Moore’s law)
  - Single core
    - Thermal issues, stopped gaining from single cores
  - Super-scalar and Multi-core processor
    - Dark Silicon issue
  - Heterogenous processing to exploit performance based on workloads
- For optimum resource utilization and best performance
    - Need to design architectures based on workloads

Need for HW specialization

- History of processors
  - Leveraging the advancements in technology nodes (Moore's law)
  - Single core
    - Thermal issues, stopped gaining from single cores
  - Super-scalar and Multi-core processor
    - Dark Silicon issue
  - Heterogenous processing to exploit performance based on workloads
- For optimum resource utilization and best performance
    - Need to design architectures based on workloads
- Complex general purpose computation is not always the solution –

Need for HW specialization

- History of processors
  - Leveraging the advancements in technology nodes (Moore's law)
  - Single core
    - Thermal issues, stopped gaining from single cores
  - Super-scalar and Multi-core processor
    - Dark Silicon issue
  - Heterogenous processing to exploit performance based on workloads
- For optimum resource utilization and best performance
    - Need to design architectures based on workloads
- Complex general purpose computation is not always the solution –
  - Image and Video processing – GPUs

Need for HW specialization

- History of processors
  - Leveraging the advancements in technology nodes (Moore's law)
  - Single core
    - Thermal issues, stopped gaining from single cores
  - Super-scalar and Multi-core processor
    - Dark Silicon issue
  - Heterogenous processing to exploit performance based on workloads
- For optimum resource utilization and best performance
    - Need to design architectures based on workloads
- Complex general purpose computation is not always the solution –
  - Image and Video processing – GPUs
  - For scientific calculation with complex mathematical function, that are not supported by basic ALUs – we need co-processors like FPU (floating point unit), TMU – Trigonometric mathematical unit, etc.

Need for HW specialization

- History of processors
  - Leveraging the advancements in technology nodes (Moore's law)
  - Single core
    - Thermal issues, stopped gaining from single cores
  - Super-scalar and Multi-core processor
    - Dark Silicon issue
  - Heterogenous processing to exploit performance based on workloads
- For optimum resource utilization and best performance
    - Need to design architectures based on workloads
- Complex general purpose computation is not always the solution –
  - Image and Video processing – GPUs
  - For scientific calculation with complex mathematical function, that are not supported by basic ALUs – we need co-processors like FPU (floating point unit), TMU – Trigonometric mathematical unit, etc.
  - Sorting, genomics – need for circuits that are proficient in sequencing: Ex. automata processors

Need for HW specialization

- History of processors
  - Leveraging the advancements in technology nodes (Moore's law)
  - Single core
    - Thermal issues, stopped gaining from single cores
  - Super-scalar and Multi-core processor
    - Dark Silicon issue
  - Heterogenous processing to exploit performance based on workloads
- For optimum resource utilization and best performance
    - Need to design architectures based on workloads
- Complex general purpose computation is not always the solution –
  - Image and Video processing – GPUs
  - For scientific calculation with complex mathematical function, that are not supported by basic ALUs – we need co-processors like FPU (floating point unit), TMU – Trigonometric mathematical unit, etc.
  - Sorting, genomics – need for circuits that are proficient in sequencing: Ex. automata processors
  - ML/AI – MAC operations, highly parallel computation, reduce data movement

Memory wall

Memory wall

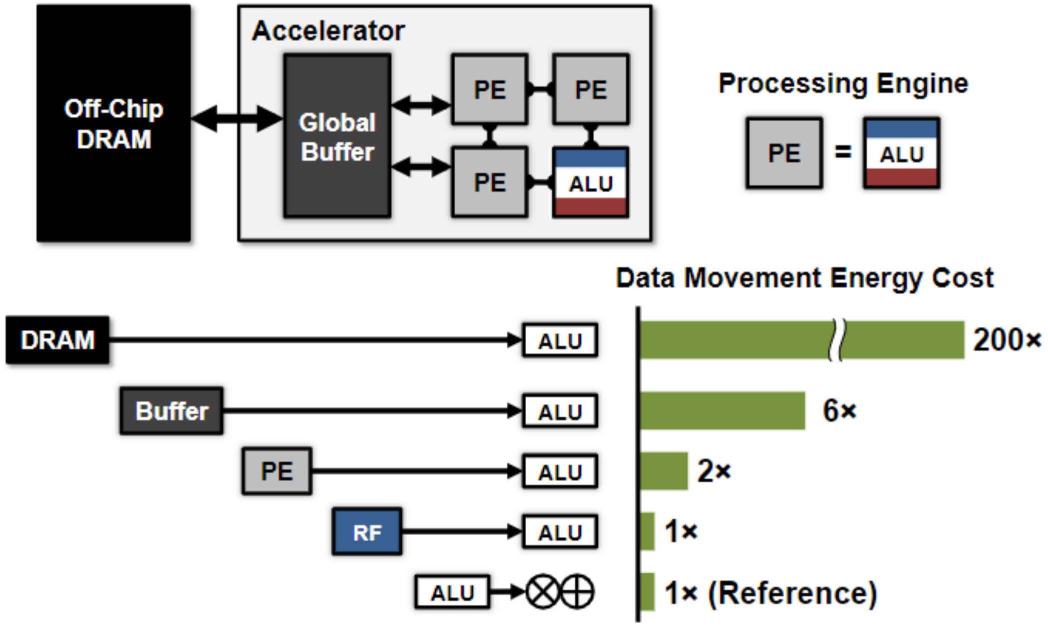


Fig. 7. Memory hierarchy and data movement energy [34].

Data movement from DRAM is the most energy expensive operation[1]

1. Hardware for Machine Learning: Challenges and Opportunities[2017], Sze et el.
2. Computing's energy problem (and what we can do about it)[2014], Mark H.

Memory wall

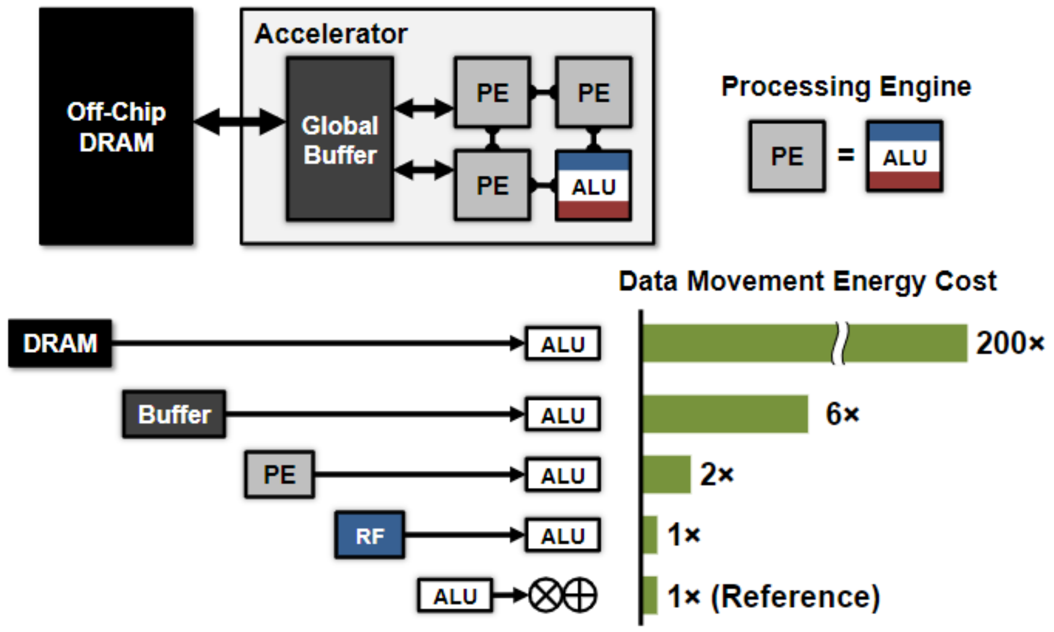


Fig. 7. Memory hierarchy and data movement energy [34].

Data movement from DRAM is the most energy expensive operation[1]

- Processor performance scales as technology scales, but the latency of DRAM access scales very slowly[2]

1. Hardware for Machine Learning: Challenges and Opportunities[2017], Sze et el.
2. Computing's energy problem (and what we can do about it)[2014], Mark H.

Memory wall

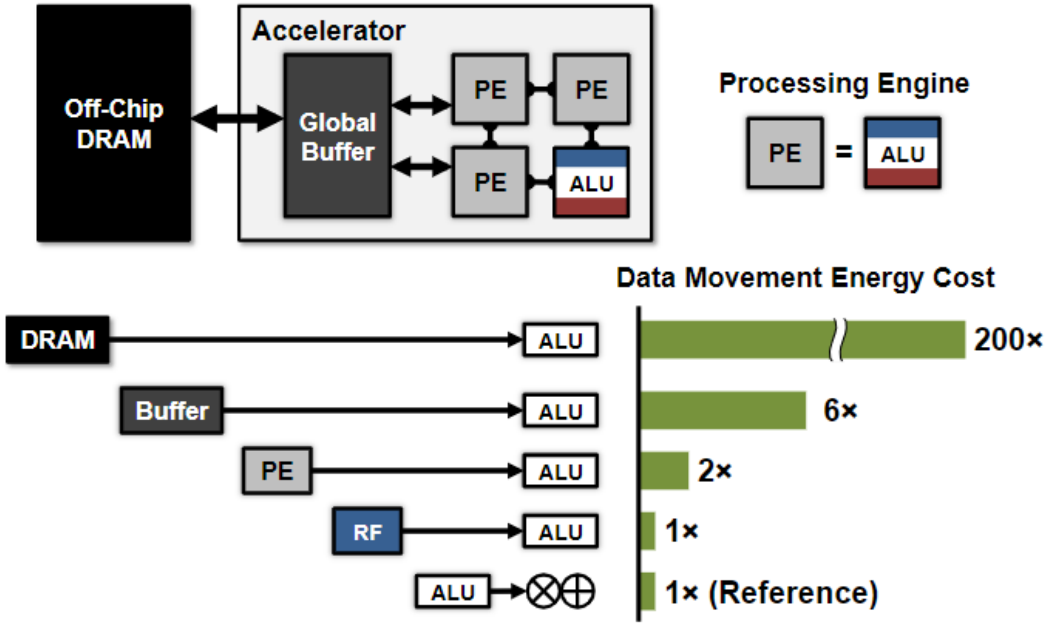


Fig. 7. Memory hierarchy and data movement energy [34].

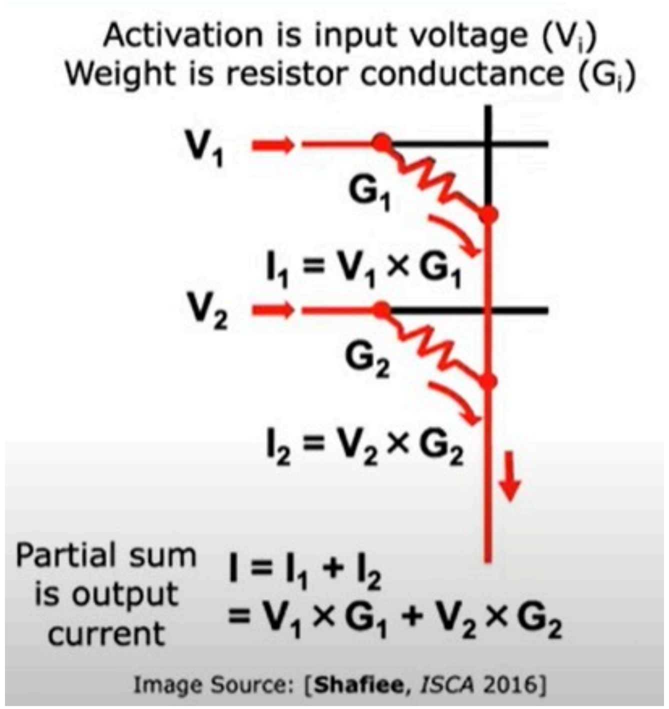
Data movement from DRAM is the most energy expensive operation[1]

- Processor performance scales as technology scales, but the latency of DRAM access scales very slowly[2]
- Why? DRAMs are traditionally designed for high transistor density, whereas compute units are designed for high performance

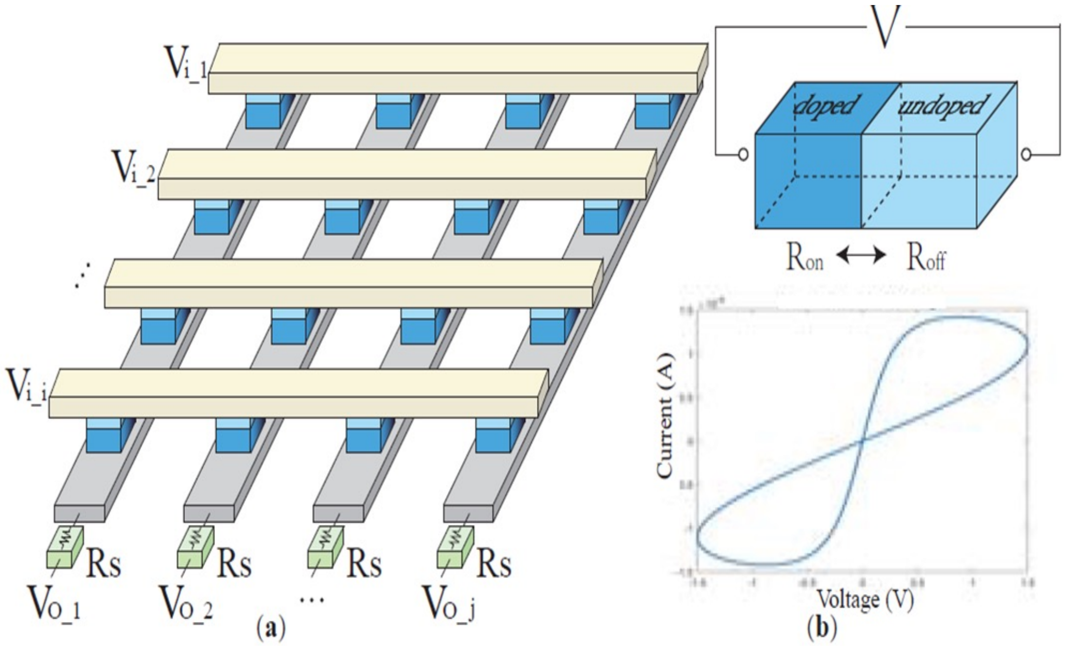
Paves way to research on
- when, where and how to compute
- improving memory bandwidth
- thinking of new memory technologies
  - Neuromorphic circuits, SNN, PiM

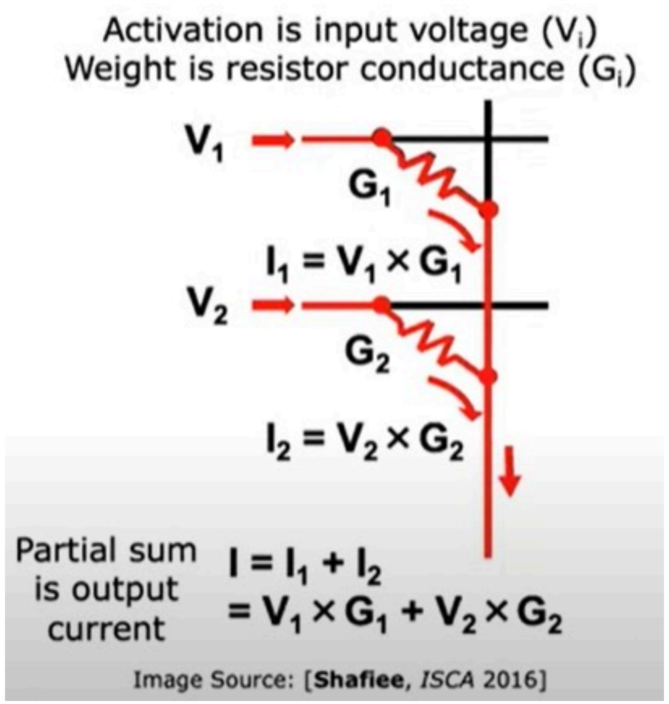
1. Hardware for Machine Learning: Challenges and Opportunities[2017], Sze et el.
2. Computing's energy problem (and what we can do about it)[2014], Mark H.

What is memristor



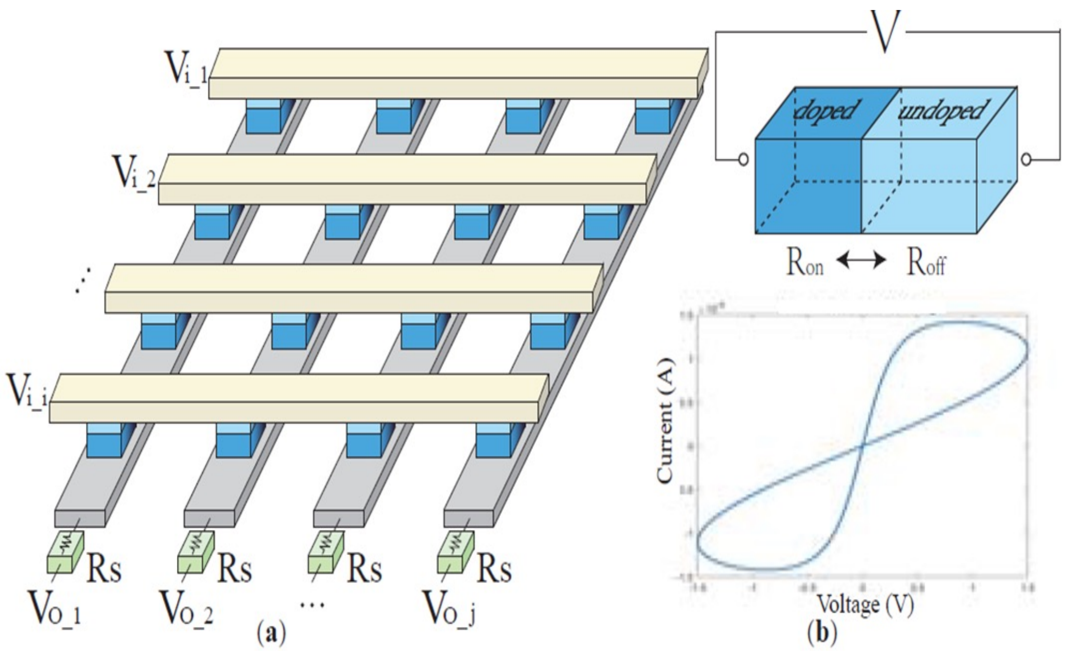
What is memristor

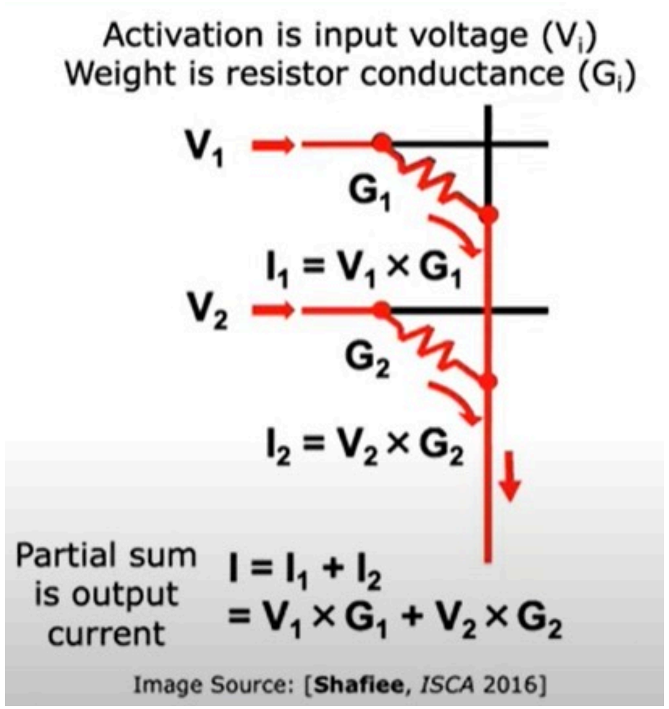




What is memristor

- Memristor is a hardware that can perform matrix multiplication with high speed
- By setting the voltage to the value of Ni and having the matrix W, the product of the matrix can be obtained by having the resistance Ri and reading the current
- Matrix multiplication is calculated in parallel => efficiency increase

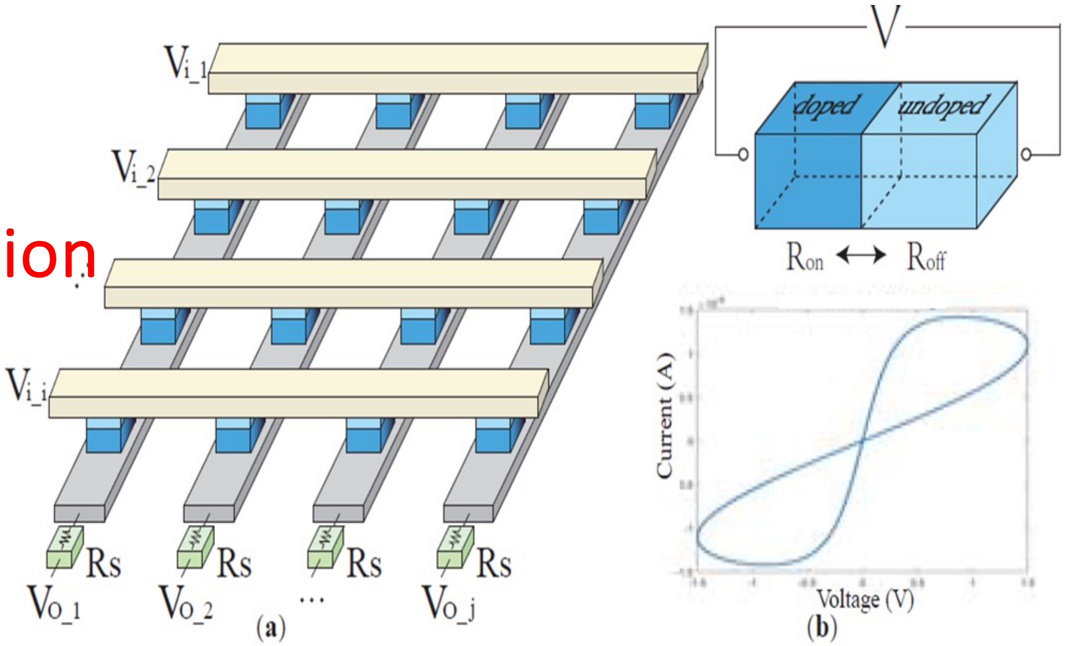


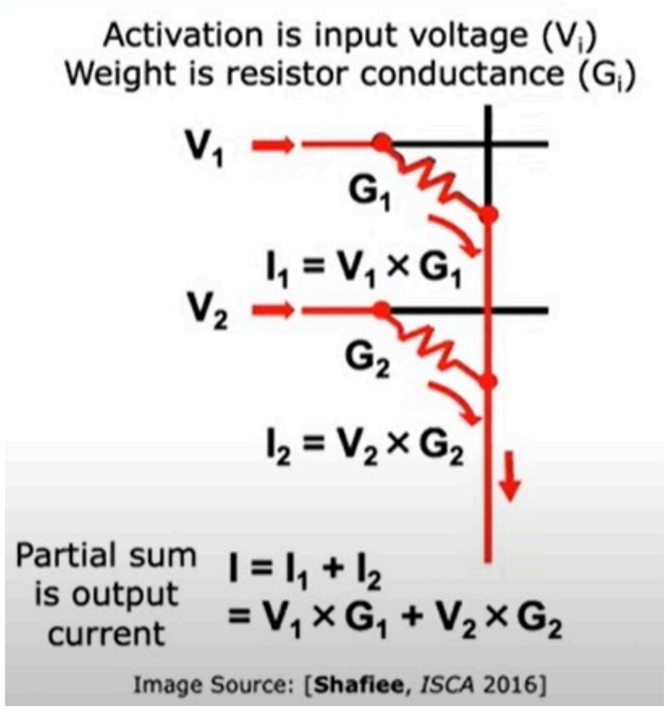


What is memristor

- Memristor is a hardware that can perform matrix multiplication with high speed
- By setting the voltage to the value of Ni and having the matrix W, the product of the matrix can be obtained by having the resistance Ri and reading the current
- Matrix multiplication is calculated in parallel => efficiency increase

High Power Consumption





How to Improve memristor Power

How to Improve memristor Power

- Quantization

=> To reduce data bits and fractions

How to Improve memristor Power

- Quantization

=> To reduce data bits and fractions

- Pruning

=> To reduce number of weights in learning

How to Improve memristor Power

- Quantization

=> To reduce data bits and fractions

Data loss

- Pruning

=> To reduce number of weights in learning

How to Improve memristor Power

- Quantization

=> To reduce data bits and fractions

Data loss

- Pruning

=> To reduce number of weights in learning

Accuracy

How to Improve memristor Power

- Quantization

=> To reduce data bits and fractions

Data loss

- Pruning

=> To reduce number of weights in learning

Accuracy

Solve This Problem Using ADMM

min F($\{w_i\}^N_{i=1}$, $\{b_i\}^N_{i=1}$) , W_i є P_i , b_i є Q_i
$\{w_i\}$,$\{w_i\}$
=> Find the Optimum Conductance State Levels
=> Remove Unnecessary Coefficients

How to Improve memristor Power

- Quantization

=> To reduce data bits and fractions

Data loss

- Pruning

=> To reduce number of weights in learning

Accuracy

Solve This Problem Using ADMM

$\min F(\{w_i\}^N_{i=1}, \{b_i\}^N_{i=1})$, $W_i \in P_i$, $b_i \in Q_i$
$\{w_i\},\{w_i\}$
=> Find the Optimum Conductance State Levels
=> Remove Unnecessary Coefficients

Accuracy
Fast and Parallel Computing
Power Usage

Current Accelerator Space

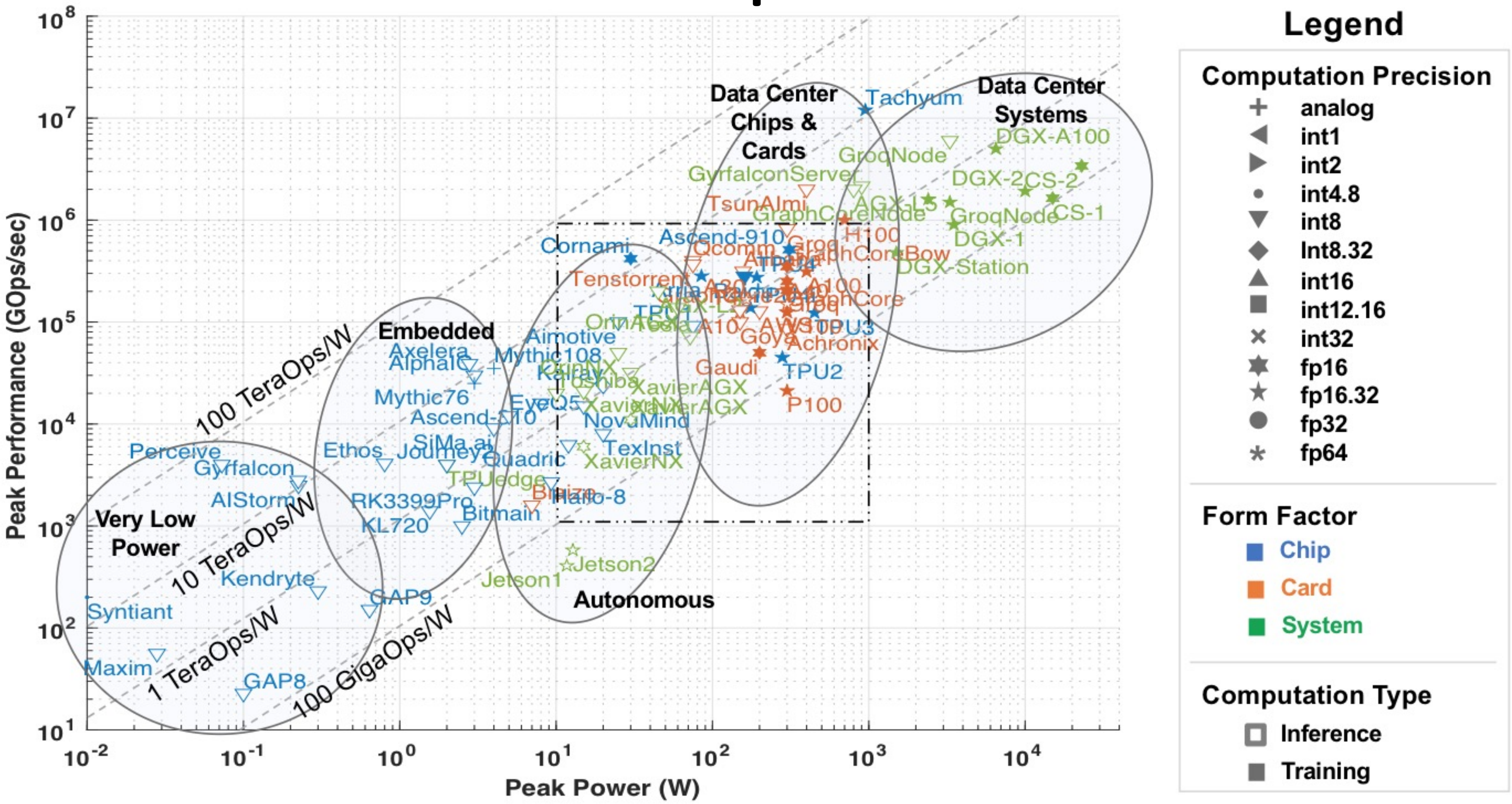


Fig. 2: Peak performance vs. power scatter plot of publicly announced AI accelerators and processors.

Figure from AI and ML Accelerator Survey and Trends [2022], Reuther et el.

Data Center AI Accelerators

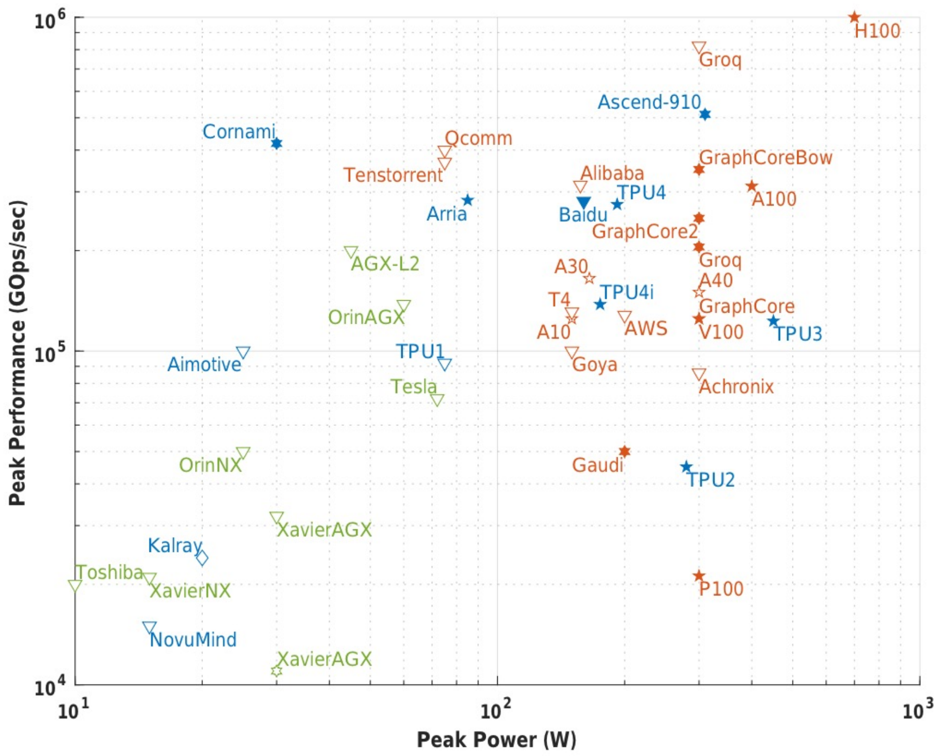


Fig. 3: Zoomed region of peak performance vs. power scatter plot.

- Requirements
- Train? Infer?
- The accelerators
  - ASIC
  - FPGA
  - CGRA?
  - PIM
  - GPU
- Our observations

Figure from AI and ML Accelerator Survey and Trends [2022], Reuther et el.

Requirements for Data Center AI Acceleration

Accelerated Computing with a Reconfigurable Dataflow Architecture[2021], SambaNova Systems.

Requirements for Data Center AI Acceleration

1. Latency (Service Level Objectives/Agreements[SLO/SLA])

Accelerated Computing with a Reconfigurable Dataflow Architecture[2021], SambaNova Systems.

Requirements for Data Center AI Acceleration

1. Latency (Service Level Objectives/Agreements[SLO/SLA])
2. Total Cost of Ownership (TCO)

Accelerated Computing with a Reconfigurable Dataflow Architecture[2021], SambaNova Systems.

Requirements for Data Center AI Acceleration

1. Latency (Service Level Objectives/Agreements[SLO/SLA])
2. Total Cost of Ownership (TCO)
3. Target select algorithms vs Target wide algorithm base

Accelerated Computing with a Reconfigurable Dataflow Architecture[2021], SambaNova Systems.

Requirements for Data Center AI Acceleration

1. Latency (Service Level Objectives/Agreements[SLO/SLA])
2. Total Cost of Ownership (TCO)
3. Target select algorithms vs Target wide algorithm base
4. Multi-tenancy and Isolation (sometimes, mixed workloads)

Accelerated Computing with a Reconfigurable Dataflow Architecture[2021], SambaNova Systems.

Requirements for Data Center AI Acceleration

1. Latency (Service Level Objectives/Agreements[SLO/SLA])
2. Total Cost of Ownership (TCO)
3. Target select algorith
4. Multi-tenancy and Isc

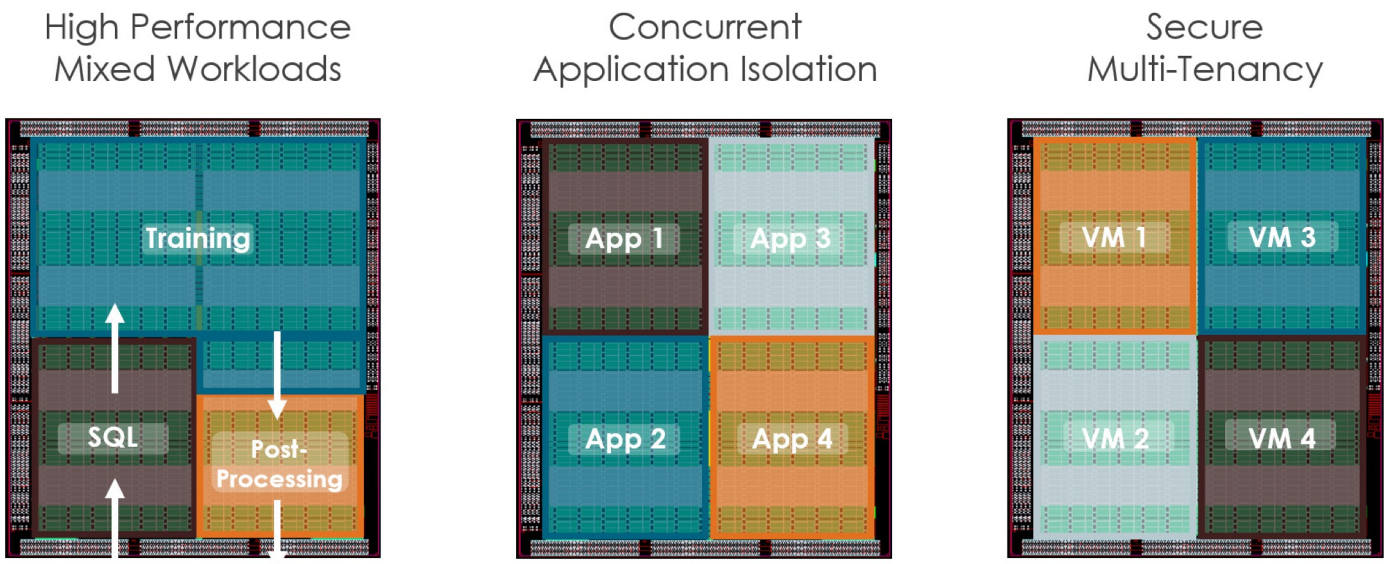


Figure 7 - Supporting multiple users or workloads simultaneously

Accelerated Computing with a Reconfigurable Dataflow Architecture[2021], SambaNova Systems.

Requirements for Data Center AI Acceleration

1. Latency (Service Level Objectives/Agreements[SLO/SLA])
2. Total Cost of Ownership (TCO)
3. Target select algorithms vs Target wide algorithm base
4. Multi-tenancy and Isolation (sometimes, mixed workloads)

Accelerated Computing with a Reconfigurable Dataflow Architecture[2021], SambaNova Systems.

Requirements for Data Center AI Acceleration

1. Latency (Service Level Objectives/Agreements[SLO/SLA])
2. Total Cost of Ownership (TCO)
3. Target select algorithms vs Target wide algorithm base
4. Multi-tenancy and Isolation (sometimes, mixed workloads)
5. Scalability and Future Proofing

Accelerated Computing with a Reconfigurable Dataflow Architecture[2021], SambaNova Systems.

Requirements for Data Center AI Acceleration

1. Latency (Service Level Objectives/Agreements[SLO/SLA])
2. Total Cost of Ownership (TCO)
3. Target select algorithms vs Target wide algorithm base
4. Multi-tenancy and Isolation (sometimes, mixed workloads)
5. Scalability and Future Proofing
6. Power consumption and Cooling

Accelerated Computing with a Reconfigurable Dataflow Architecture[2021], SambaNova Systems.

Requirements for Data Center AI Acceleration

1. Latency (Service Level Objectives/Agreements[SLO/SLA])
2. Total Cost of Ownership (TCO)
3. Target select algorithms vs Target wide algorithm base
4. Multi-tenancy and Isolation (sometimes, mixed workloads)
5. Scalability and Future Proofing
6. Power consumption and Cooling
7. Programmability

Accelerated Computing with a Reconfigurable Dataflow Architecture[2021], SambaNova Systems.

Train? Infer?

TRAINING

1. Frequent memory updates (forward and back propagation)
2. Large models and parallelization constraints – distributed training is limited by off-chip bandwidth
3. Wider operands
4. Training is experimentation
5. Compute intensive

INFERENCE

1. Weights are only read once
2. Parallelization is easier
3. High precision is not a "need"
4. Inference is a one-time activity
5. Not as intensive as training

Some Common Strategies

Accelerated Computing with a Reconfigurable Dataflow Architecture[2021], SambaNova Systems.

Some Common Strategies

- High Bandwidth and High Capacity Memories to store weights

Accelerated Computing with a Reconfigurable Dataflow Architecture[2021], SambaNova Systems.

Some Common Strategies

- High Bandwidth and High Capacity Memories to store weights
- Matrix multiply and Vector operation units

Accelerated Computing with a Reconfigurable Dataflow Architecture[2021], SambaNova Systems.

Some Common Strategies

- High Bandwidth and High Capacity Memories to store weights
- Matrix multiply and Vector operation units
  - Systolic arrays and (explicit) dataflow architectures

Accelerated Computing with a Reconfigurable Dataflow Architecture[2021], SambaNova Systems.

Some Common Strategies

- High Bandwidth and High Capacity Memories to store weights
- Matrix multiply and Vector operation units
  - Systolic arrays and (explicit) dataflow architectures

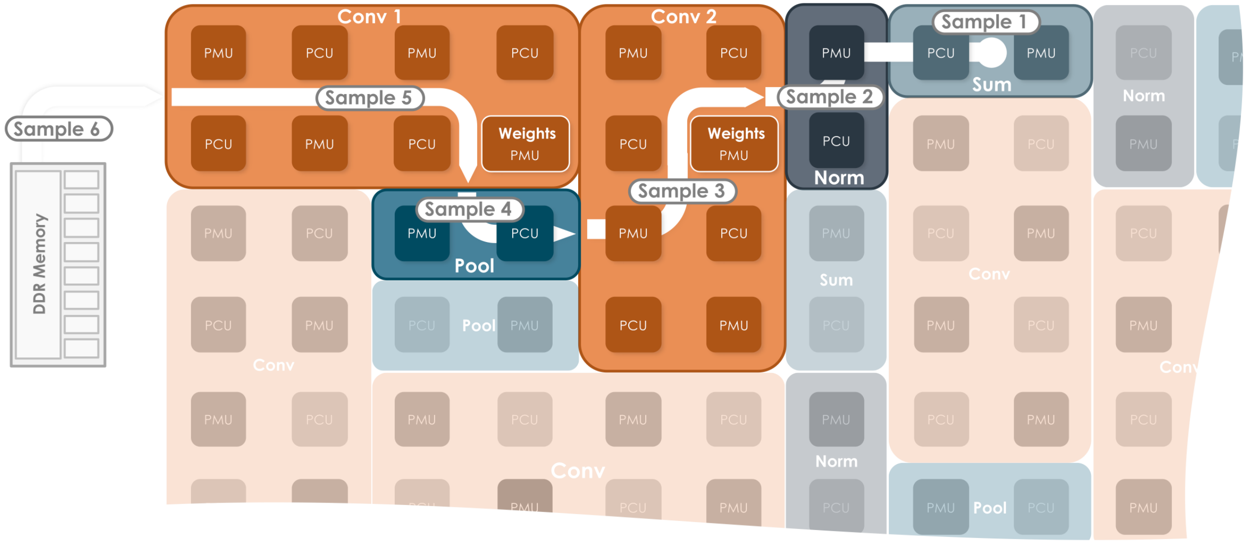


Figure 4 - RDU dataflow execution

Accelerated Computing with a Reconfigurable Dataflow Architecture[2021], SambaNova Systems.

Some Common Strategies

- High Bandwidth and High Capacity Memories to store weights
- Matrix multiply and Vector operation units
  - Systolic arrays and (explicit) dataflow architectures

Accelerated Computing with a Reconfigurable Dataflow Architecture[2021], SambaNova Systems.

Some Common Strategies

- High Bandwidth and High Capacity Memories to store weights
- Matrix multiply and Vector operation units
  - Systolic arrays and (explicit) dataflow architectures
- Large on-chip memories (weight pinning)

Accelerated Computing with a Reconfigurable Dataflow Architecture[2021], SambaNova Systems.

Some Common Strategies

- High Bandwidth and High Capacity Memories to store weights
- Matrix multiply and Vector operation units
  - Systolic arrays and (explicit) dataflow architectures
- Large on-chip memories (weight pinning)
- Data types of varied precision (tailored for AI)

Accelerated Computing with a Reconfigurable Dataflow Architecture[2021], SambaNova Systems.

Some Common Strategies

- High Bandwidth and High Capacity Memories to store weights
- Matrix multiply and Vector operation units
  - Systolic arrays and (explicit) dataflow architectures
- Large on-chip memories (weight pinning)
- Data types of varied precision (tailored for AI)
  - FP32, TF32, BF16, FP16, DLFLT16, UINT8, INT8 and maybe INT4

Accelerated Computing with a Reconfigurable Dataflow Architecture[2021], SambaNova Systems.

Some Common Strategies

- High Bandwidth and High Capacity Memories to store weights
- Matrix multiply and Vector operation units
  - Systolic arrays and (explicit) dataflow architectures
- Large on-chip memories (weight pinning)
- Data types of varied precision (tailored for AI)
  - FP32, TF32, BF16, FP16, DLFLT16, UINT8, INT8 and maybe INT4
  - Size of datatype influences power

Accelerated Computing with a Reconfigurable Dataflow Architecture[2021], SambaNova Systems.

Some Common Strategies

- High Bandwidth and High Capacity Memories to store weights
- Matrix multiply and Vector operation units
  - Systolic arrays and (explicit) dataflow architectures
- Large on-chip memories (weight pinning)
- Data types of varied precision (tailored for AI)
  - FP32, TF32, BF16, FP16, DLFLT16, UINT8, INT8 and maybe INT4
  - Size of datatype influences power
- SIMD execution, MIMD execution, SIMT execution!

Accelerated Computing with a Reconfigurable Dataflow Architecture[2021], SambaNova Systems.

Hardware and *Software* for AI/ML?

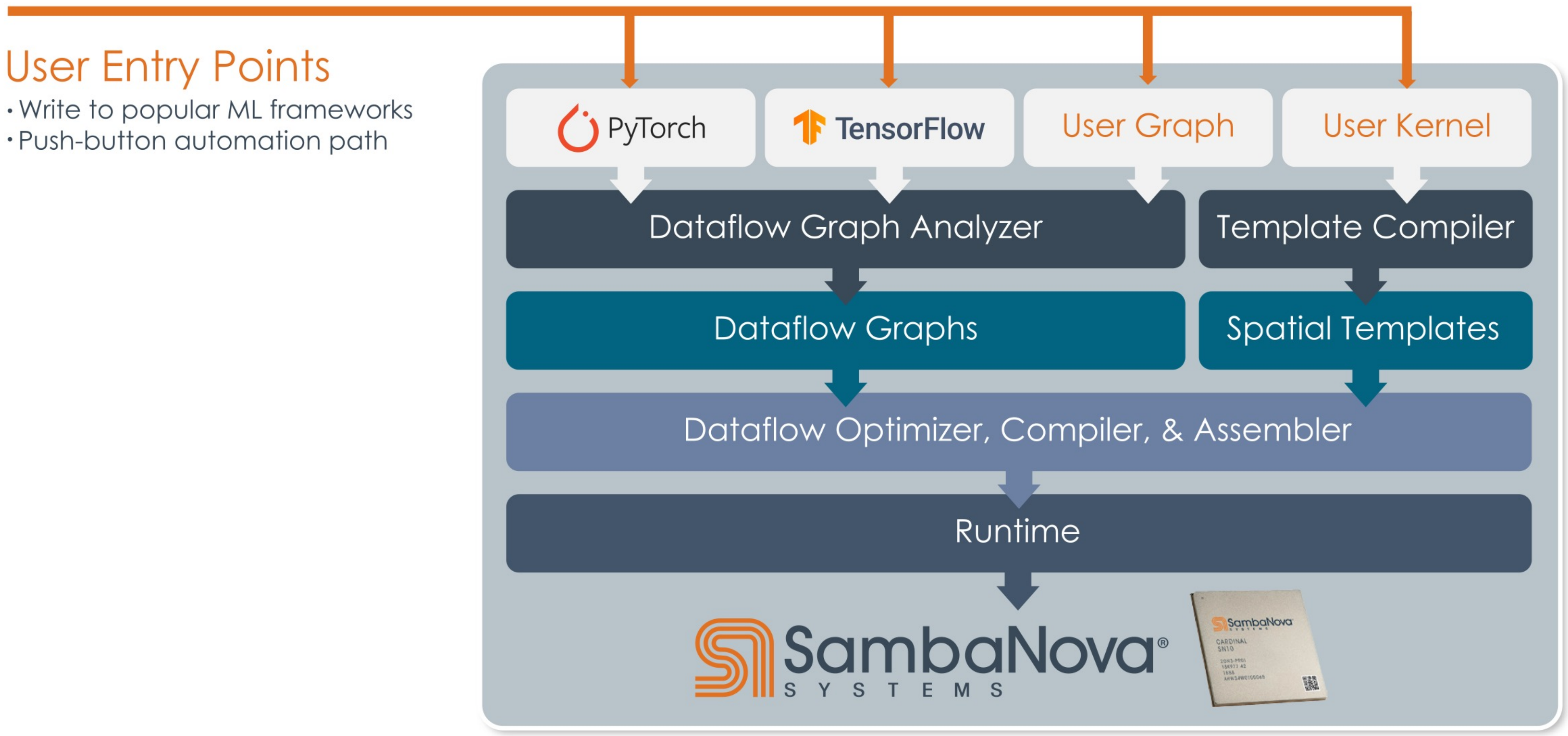


Figure 6 - SambaFlow components

Accelerated Computing with a Reconfigurable Dataflow Architecture[2021], SambaNova Systems.

Hardware and *Software* for AI/ML?

Safe to assume all the following works have a similar software stack.

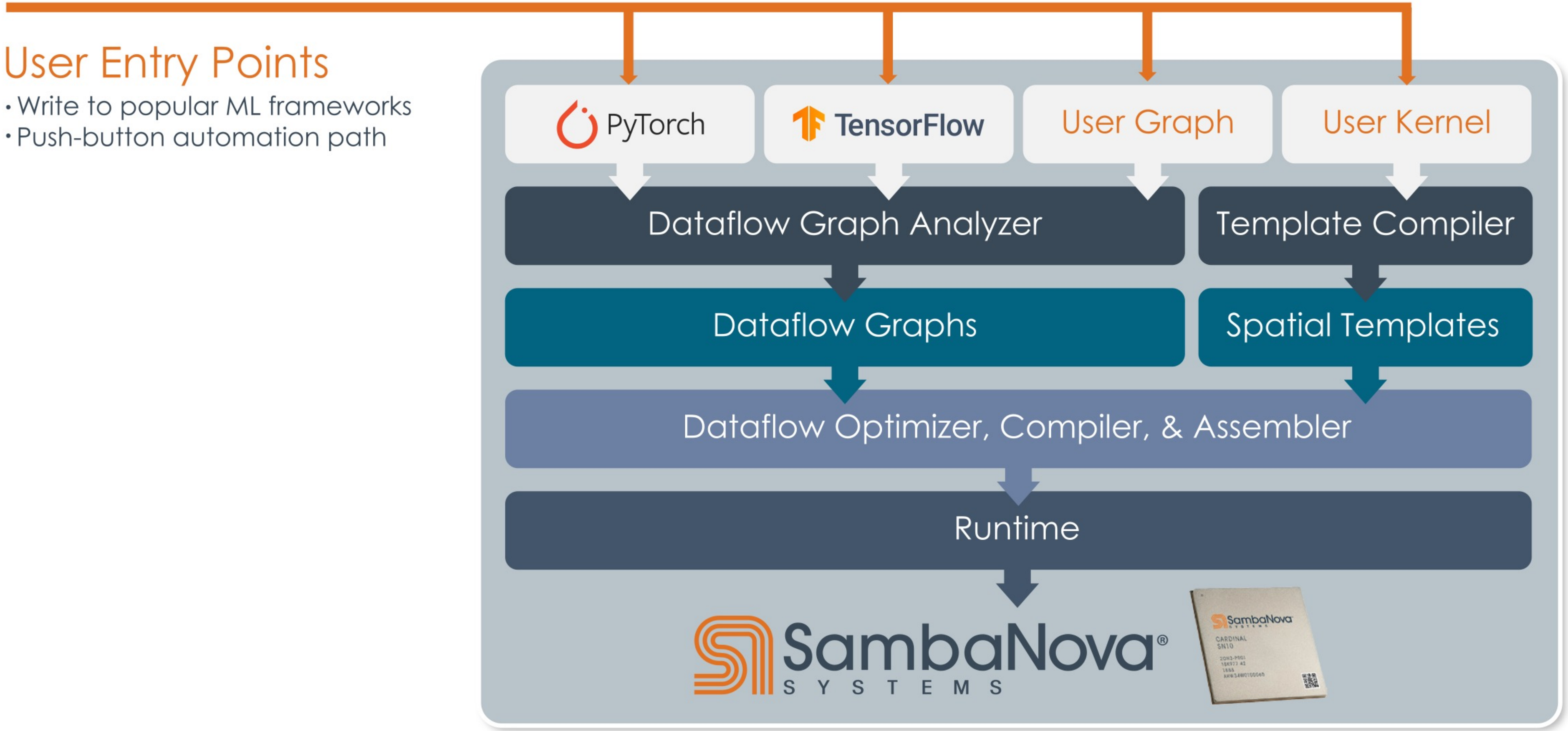


Figure 6 - SambaFlow components

Accelerated Computing with a Reconfigurable Dataflow Architecture[2021], SambaNova Systems.

DaDianNao - ASIC Accelerator

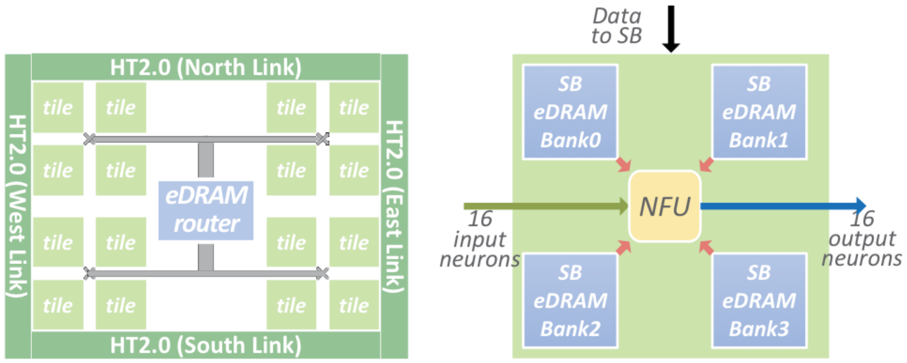


Figure 5: *Tile-based organization of a node (left) and tile architecture (right). A node contains 16 tiles, two central eDRAM banks and fat tree interconnect; a tile has an NFU, four eDRAM banks and input/output interfaces to/from the central eDRAM banks.*

- Training and Inference
- Tile based accelerator (recurring design choice)
- **Uses Embedded DRAM (eDRAM) instead of SRAMs for density**
- **Weights are pinned to the eDRAM - limits off-chip memory access.**
- More space to memory rather than compute

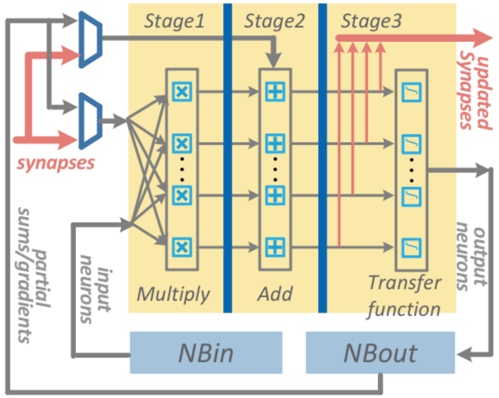


Figure 6: *The different (parallel) operators of an NFU: multipliers, adders, max, transfer function.*

- **Memory is split between tiles for high bandwidth.**
- More MACs than contemporary GPUs
- 36MB of eDRAM is insufficient for current ML model training.

DaDianNao: A Machine-Learning Supercomputer [ISCA 2014], Chen et al.

Google TPU v1 and v4i - ASIC Inference accelerators

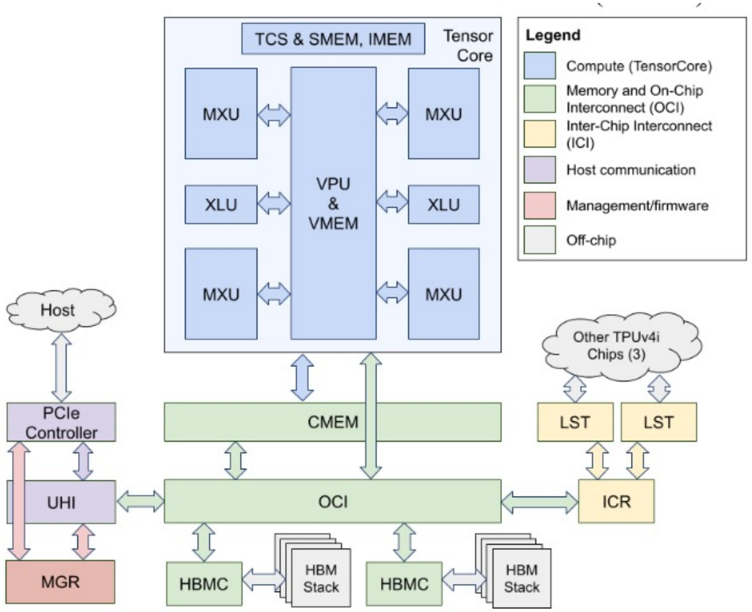


Figure 5. TPUv4i chip block diagram. Architectural memories are HBM, Common Memory (CMEM), Vector Memory (VMEM), Scalar Memory (SMEM), and Instruction Memory (IMEM). The data path is the Matrix Multiply Unit (MXU), Vector Processing Unit (VPU), Cross-Lane Unit (XLU), and TensorCore Sequencer (TCS). The uncore (everything not in blue) includes the On-Chip Interconnect (OCI), ICI Router (ICR), ICI Link Stack (LST), HBM Controller (HBMC), Unified Host Interface (UHI), and Chip Manager (MGR).

- Inference only accelerators targeted for MLP, RNN-LSTM and CNN (in v1 - 2015)
- Support for BERT, transformer encoder and LSTM decoder, Wave RNN (2020).

TPU v4i details:

- 128MB common memory allows reuse of weights during inference
- Inference in batches
- **Systolic array Matrix Multiplication - 4x 128x128**
- **XLA compiler compiles the NN models.**
- **322b VLIW ISA**
- **175W TDP - Air cooled**
- on-chip interconnect
- bf16/int8

In-Datacenter Performance Analysis of a Tensor Processing Unit (ISCA 2017), Jouppi et al.
Ten lessons from Three Generations Shaped Google's TPUv4i (ISCA 2021), Jouppi et al.

Google TPU v2 and v3: ASIC Training Accelerators

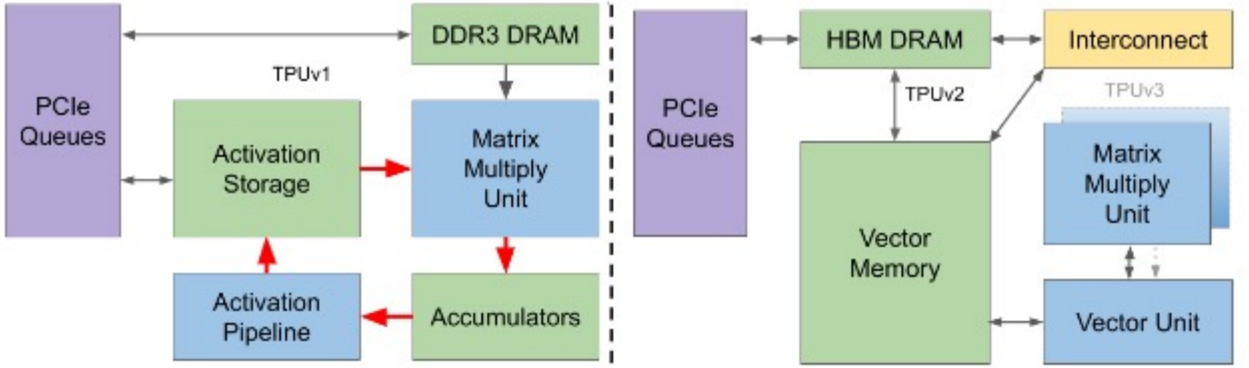


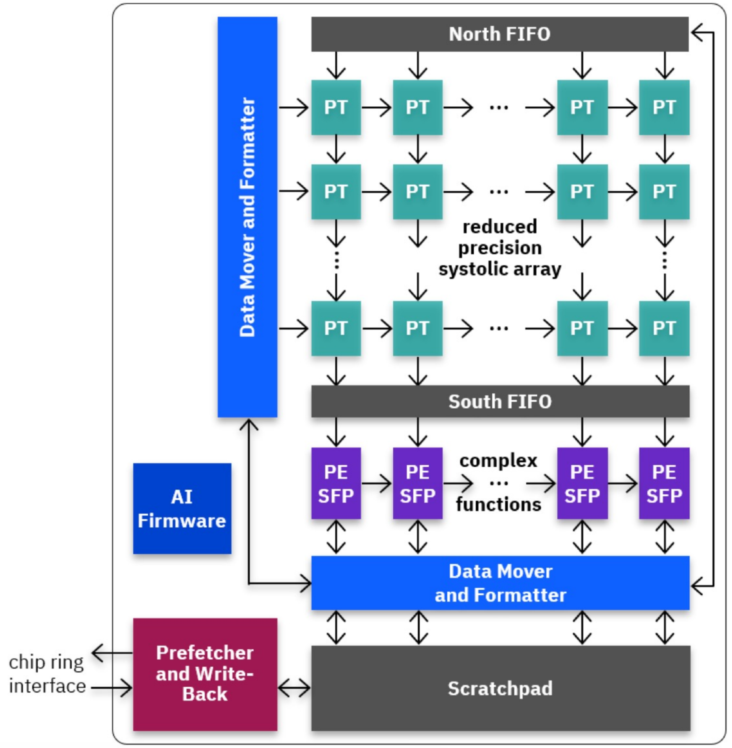
Figure 1. TPUv1 block diagram (left) vs TPUv2/v3.

v2 and v3 were built around the v1 inference chip.

1. **Larger matrix multiply unit**
2. **High Bandwidth Memory with a vector scratch pad memory (SRAM)**
3. **Support for bf16 (Matrix Multiplication), fp32 (accumulation)**
4. Activation pipeline is replaced with a more **general purpose vector compute unit** for training
5. **On and off-chip networking for parallelization at scale. (2 cores per chip)**
6. XLA compiler support. Same VLIW ISA

The Design Process of Google's Training Chips: TPUv2 and TPUv3 [2021], Norrie et al.

AI Accelerator on IBM Telum - ASIC



- On-chip accelerator for inference.
- **Primarily for privacy and latency concerns**
- All threads on the multicore can offload
- Interfaced shared L2 cache
  - **Coherency is maintained with firmware**
  - Like atomic instructions
  - Suitable when models are large than L1
- **Generic accelerator - for multiple models**
- **DLFLT16 support. Big Endian**
- Firmware updates and Firmware controls offloading
- NNPA - Neural Network Processing Assist instructions
- No influence on power and clock frequency
- **Accessed in a per request basis, no chaining.**

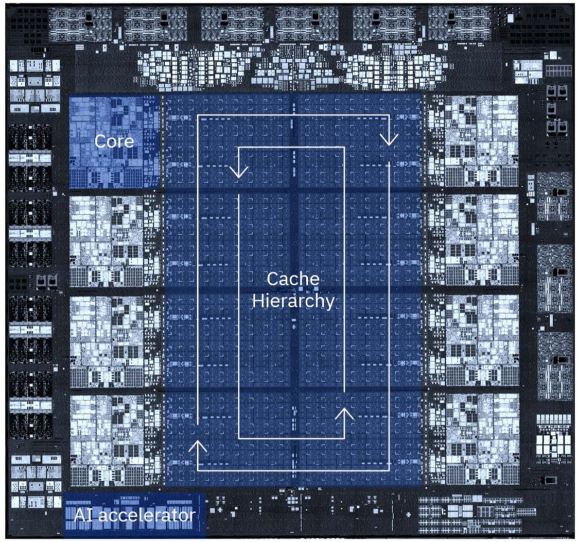
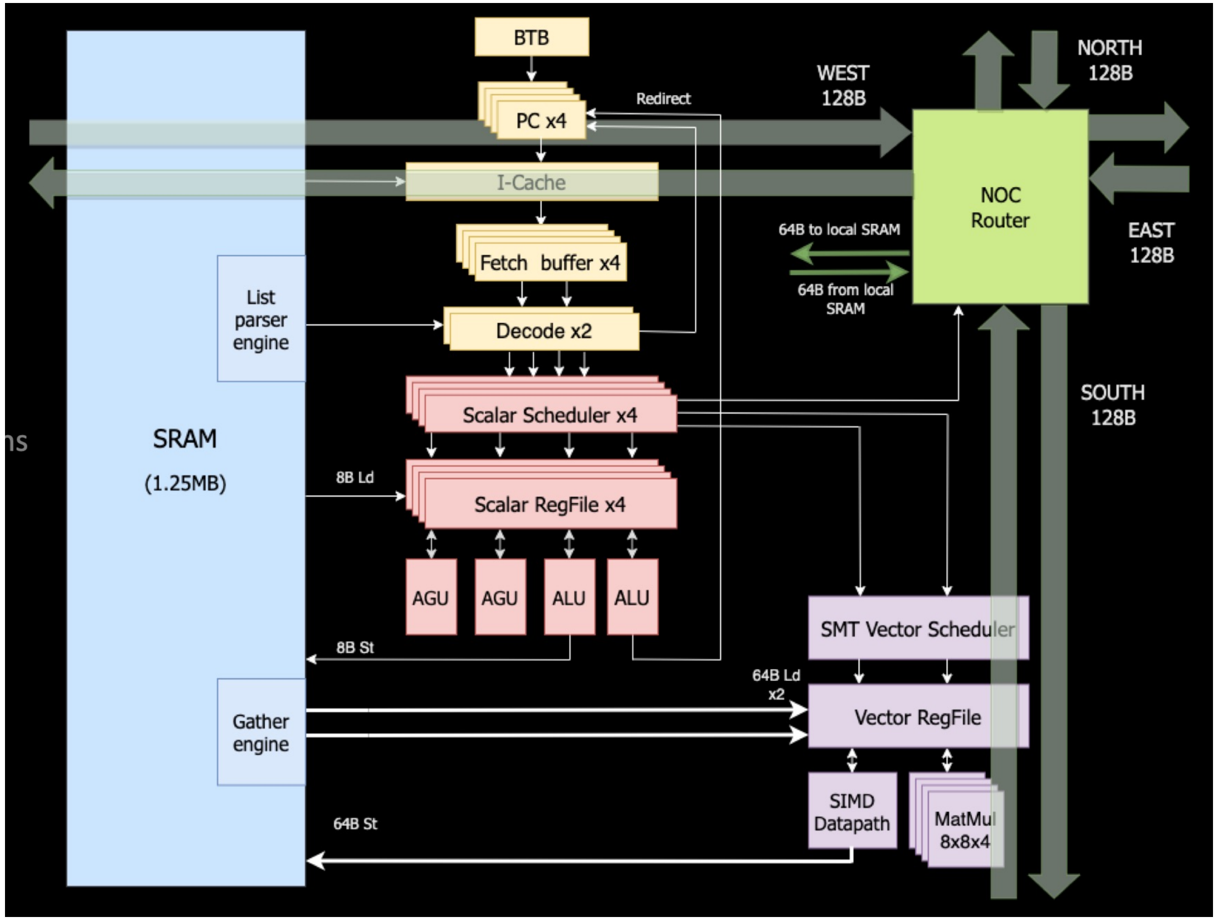


Figure 1: Telum chip die photo highlighting the optimized core, the new cache hierarchy and the AI accelerator.

AI Accelerator on IBM Telum Processor [2022], Lichtenau et el.

Tesla Dojo: CPU with AI capabilities - ASIC



- Middle ground Between General Purpose CPU and Application Specific ASIC
- **Uses a custom ISA tuned for ML**
- **Wide SMT Vector and Scalar functional Units**
- Large SRAM
- Built to Scale
- **NOC Router designed for throughput - relies more on data movement than local storage**.
- **VM, coherency and other poorly scaling features were omitted.**

Related:
Tesla had its FSD chip - inference in Autonomous vehicles which is very similar to TPUv1.

DOJO: The microarchitecture of Tesla's Exa-Scale Computer [2022], Talpes et al.

Project Brainwaves - FPGA Inference Accelerator

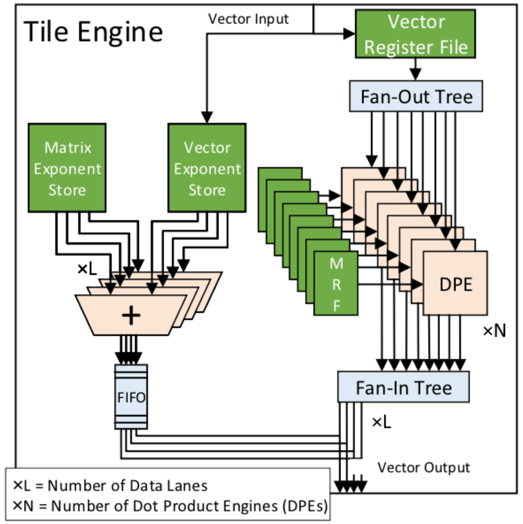


Fig. 5. Matrix-vector tile engine microarchitecture.

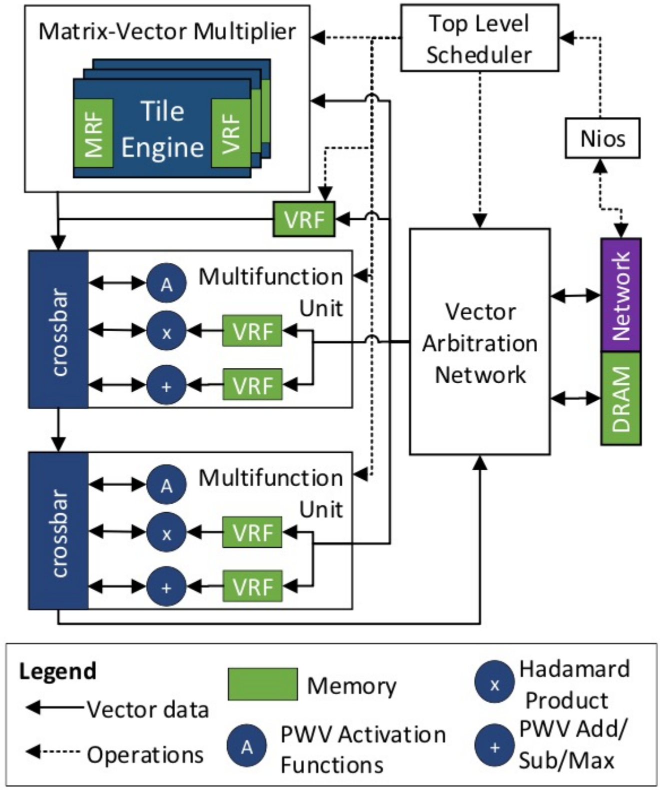
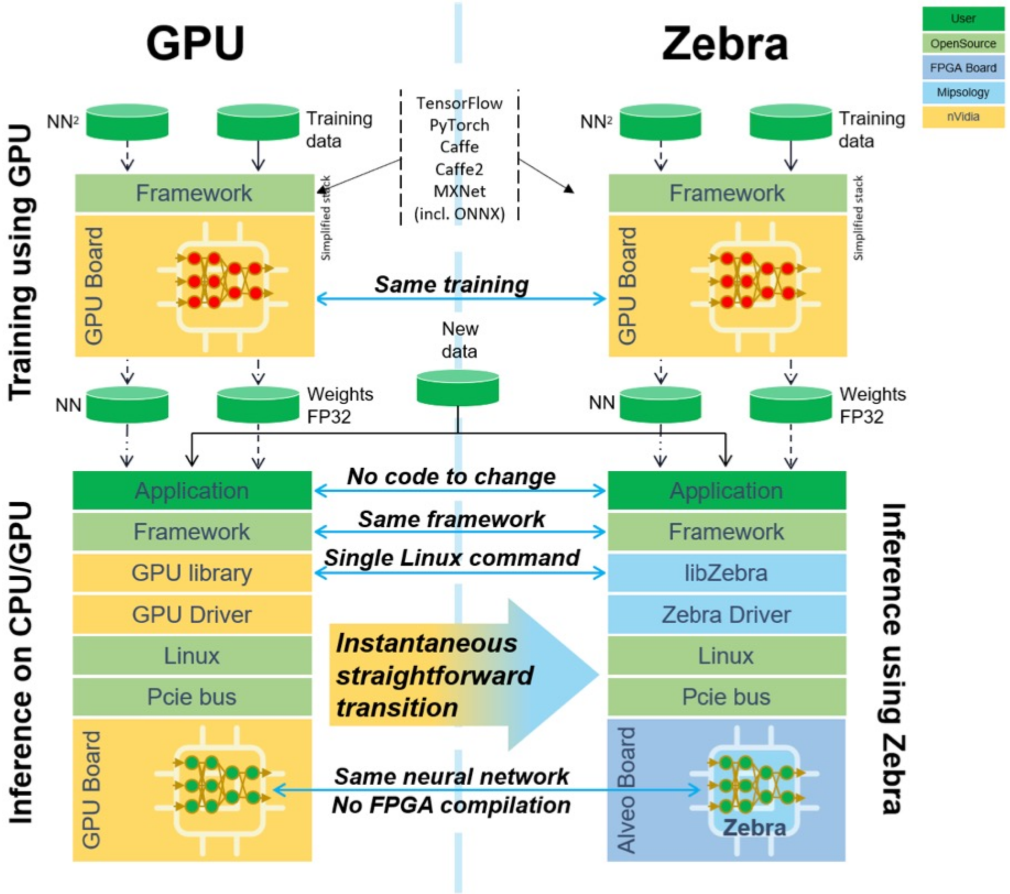


Fig. 3. Microarchitecture overview.

- **FPGA based Neural Processing Unit (NPU)**
  - **Configurable during compilation**
  - **High Flexibility**
- **Batch Size = 1 - Low Latency**
- DNN accelerator
- **Single threaded SIMD ISA**
  - **Spawns millions of primitive ops**
  - **Minimal compiler support needed**
- **Matrix-Vector Multiplication** and not Matrix-Matrix
- Dataflow architecture (instruction chaining)
- Model Pinning
- Sub-banked memory (Similar to DaDianNao)
- 125W on Stratix 10 FPGAs

A Configurable Cloud-ScaleDNN Processor for Real-Time AI [ISCA 2018], Fowers et al.

Mipsology with Xilinx FPGAs - FPGA Inference Accelerator



- CNN Inference
- **Software Stack** which is aware of underlying FPGA
- Accelerates the common layers
- **Pre-compiled FPGA binaries** - optimzed for workload
- Zero FPGA knowledge required
- Scalable for N FPGA boards

Deep Learning Inferencing with Mipsology using Xililnx ALVEO™ on Dell EMC Infrastructure [2019]

Mozart: Reuse Exposed DataFlow - CGRA

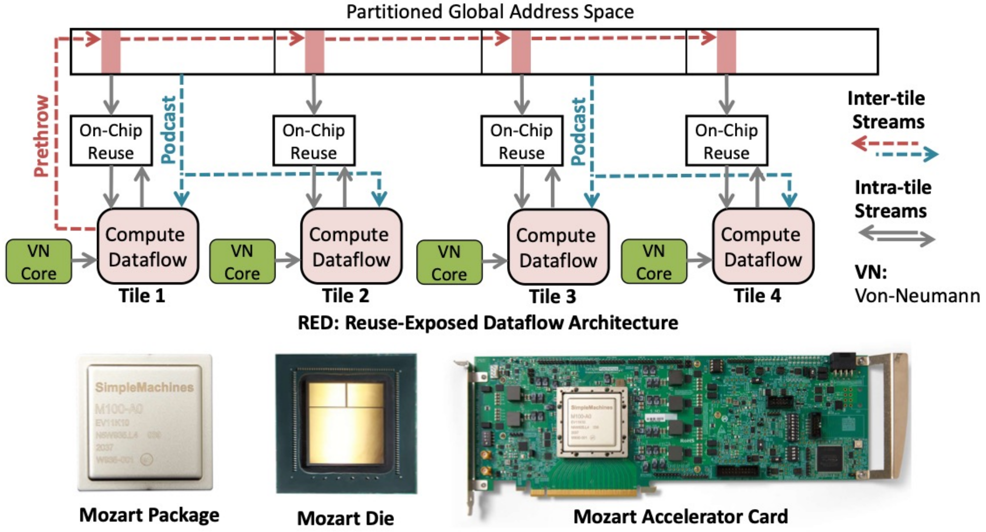
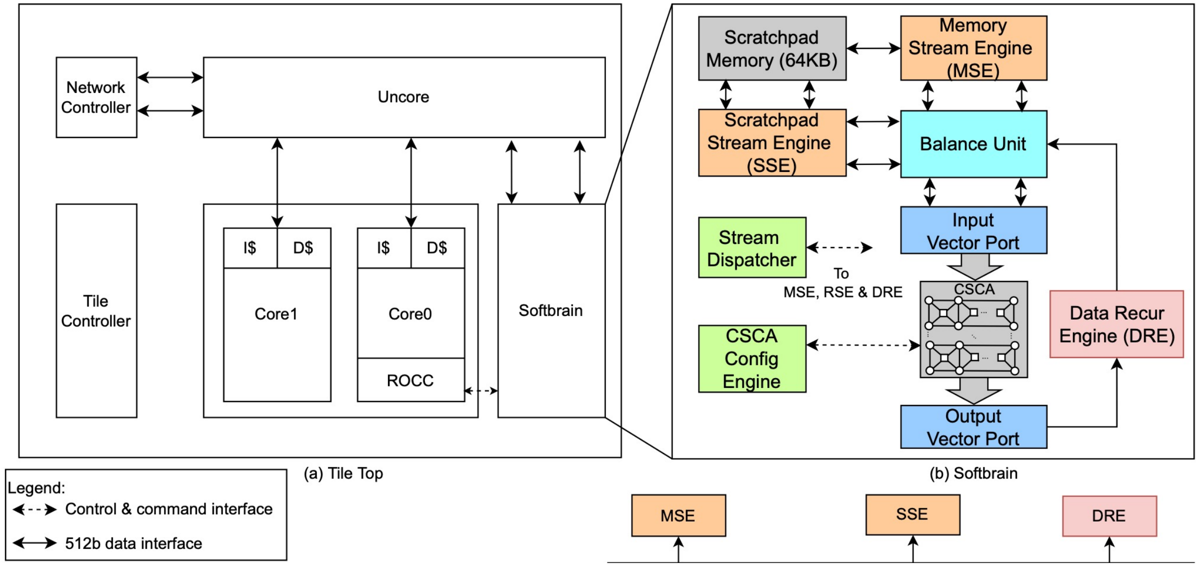


Figure 1: Mozart Hardware and its Reuse-Exposed Dataflow Architecture (RED)



- **Parallelism in Smaller Batches (N=4)** - Accelerate features General to ML and not just DNN/CNN
- Reuse Data to maximum extent
- **Streaming processor - gather and scatter-esque instructions**
- **Configurable Circuit Switch Compute Array (CSCA) - 8 FUs - Dataflow Architecture**
- **Software for configuration of CSCA based on the model**.

The Mozart Reuse Exposed Dataflow Processor for AI and Beyond [2022], Sankaralingam et al.

SambaNova: Reconfigurable Dataflow - CGRA

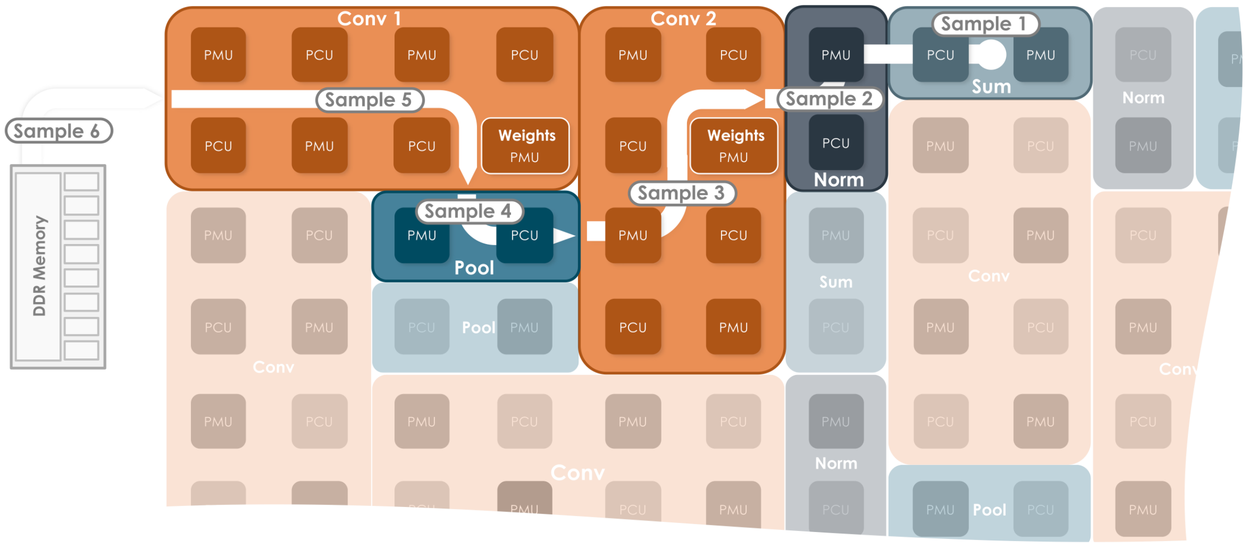
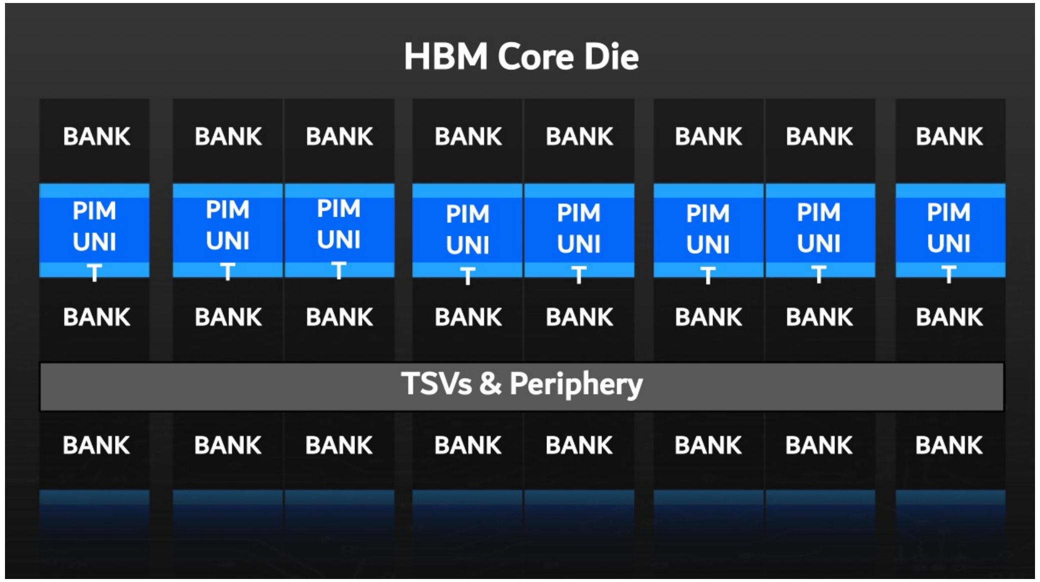


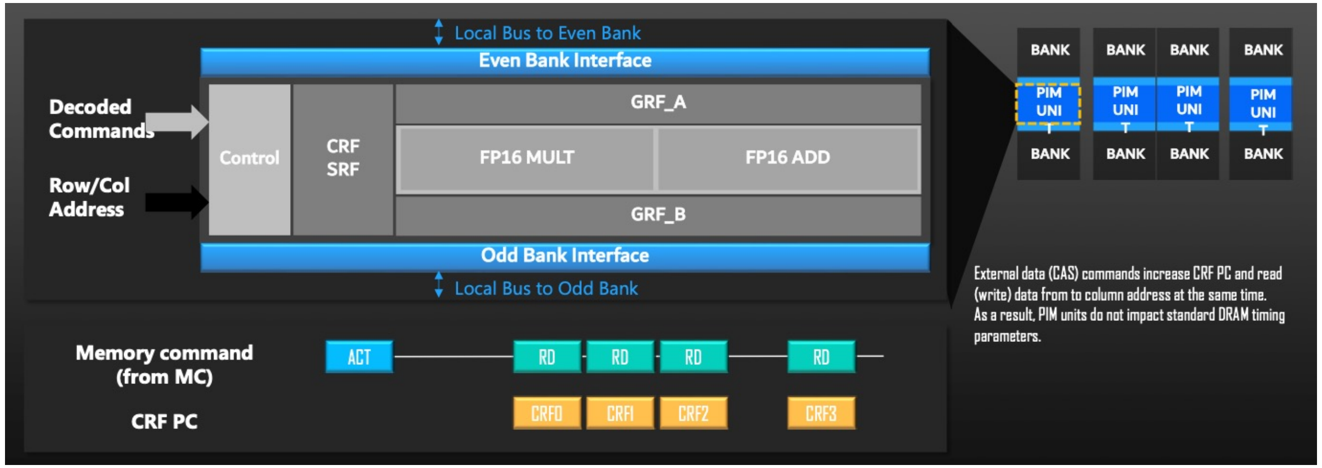
Figure 4 - RDU dataflow execution

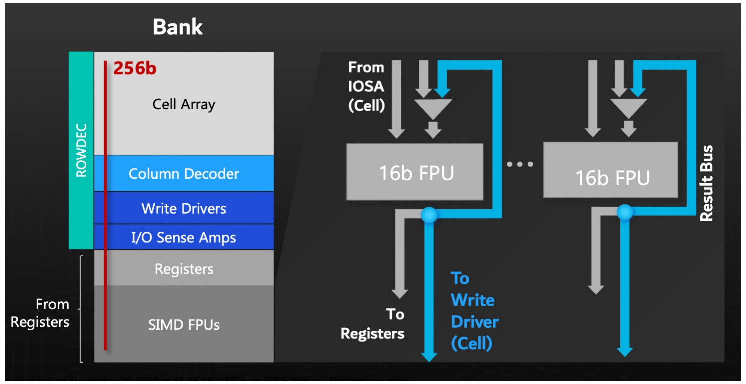
- Train and Infer
- Accelerate **multiple workloads** (not just A/ML)
- Support **pre/post processing** of data
- ISA tuned for Dataflow architecture
- **Reconfigurable DataFlow Unit(RDU)** with interconnects
- **PCU** - Pattern Compute Unit - **SIMD**
- **PMU** - Pattern Memory Unit - **SRAM**
- **Symbaflow software stack - TensorFlow to RDU**
- Scalable

Accelerated Computing with a Reconfigurable Dataflow Architecture[2021], SambaNova Systems.

Samsung Aquabolt-XL - HBM2 with PIM







- **Programmable PIM execution unit at the I/O boundary of a HBM2 bank**
  - Bank-level parallelism: **access multi banks/FPUs in a lockstep manner**
  - Same formfactor as non-PIM counterpart - no redesign of DRAM core
- **SIMD FPUs in the PIM block** - Custom RISC 32b ISA
- **Software stack** to exploit new capabilities.

Aquabolt-XL: Samsung HBM2-PIM with in-memory processing for ML accelerators and beyond [2021], Kim et al.

GPU - Graphics Processing Uni

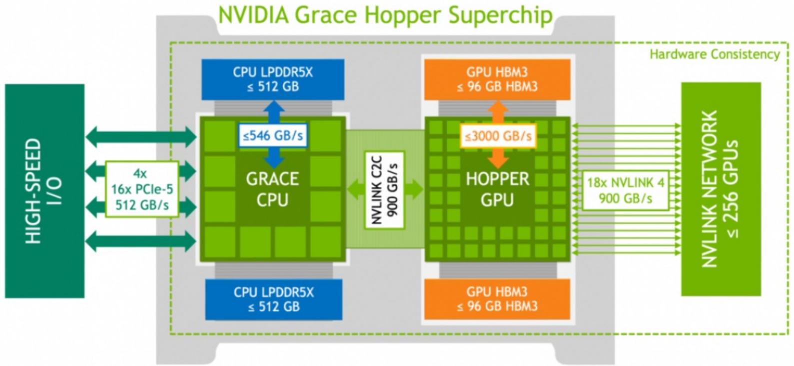


Figure 2. NVIDIA Grace Hopper Superchip logical overview

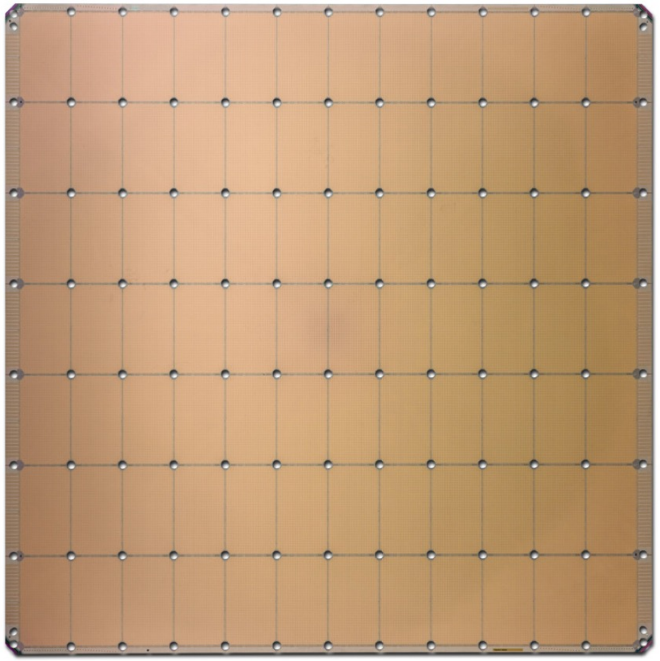
Figure 7. GH100 Streaming Multiprocessor (SM)

- Streaming Multiprocessor (SM)
- **Dynamic programming Instructions (DPX)**
- **Tensor Memory Accelerator** (Data transfer between local and global memory)
- **Transformer Engine**
- **HBM3** and **PCIe 5**
- NVLink 4 - 900GBps (Multi GPU IO)
- NVSwitch - 13.6 Tbits/sec (Between GPUs in clusters, datacenters)
- SXM for high power and High Bandwidth (alternate to PCIe)

Clustering of Accelerators a key idea with models scaling.

NVIDIA H100 Tensor Core GPU Architecture [2022], Nvidia.

Cerebras Systems: Wafer Scale Engine 2



Cerebras WSE-2
2.6 Trillion Transistors
46,225 mm² Silicon

Largest GPU
54.2 Billion Transistors
826 mm² Silicon

Figure 2. The Cerebras WSE-2 and the largest Graphics Processing Unit in comparison

- Use the **ENTIRE WAFER!**
- Interconnects - Swarm
- No Multiplying by zero - sparsity aware
- Sparse Linear Algebra Compute (SLAC) cores

	Cerebras WSE-2	A100	Cerebras Advantage
Chip size	46,225 mm²	826 mm²	**56 X**
Cores	850,000	6,912 + 432	**123 X**
On chip memory	40 Gigabytes	40 Megabytes	**1,000 X**
Memory bandwidth	20 Petabytes/sec	1,555 Gigabytes/sec	**12,862 X**
Fabric bandwidth	220 Petabits/sec	600 Gigabytes/sec	**45,833 X**

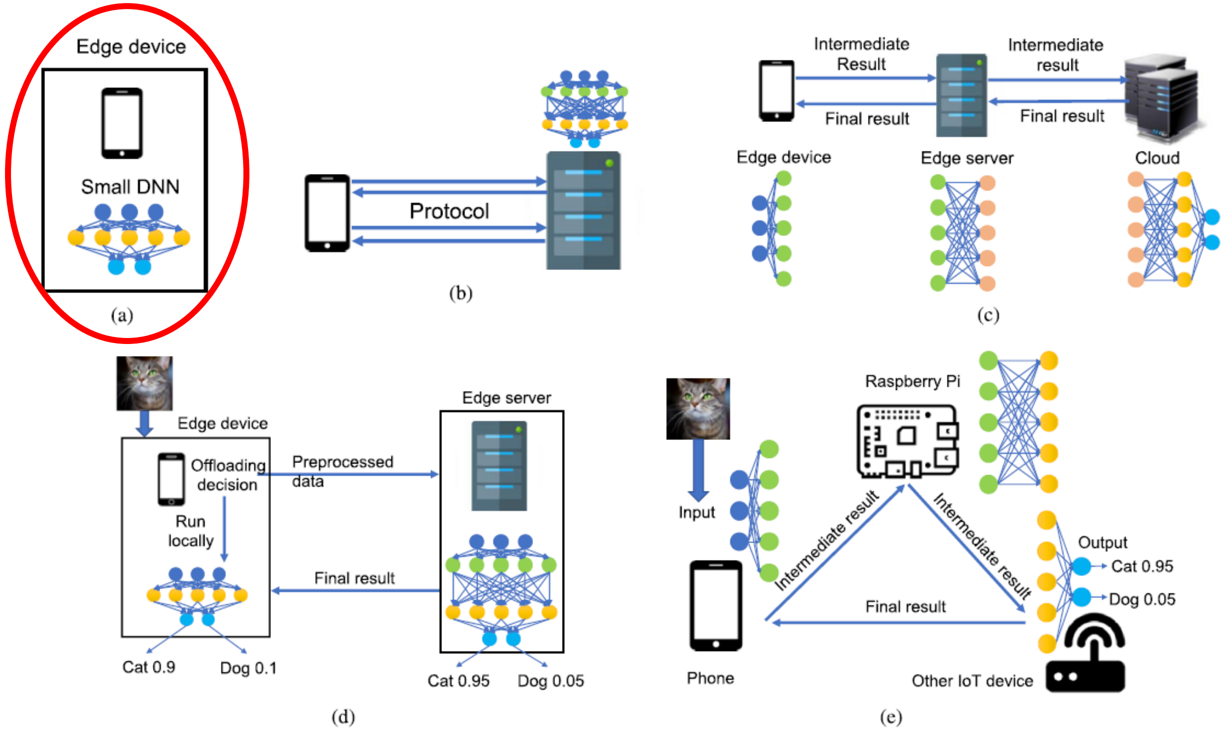
Table 1. Overview of the magnitude of advancement made by the Cerebras WSE-2.

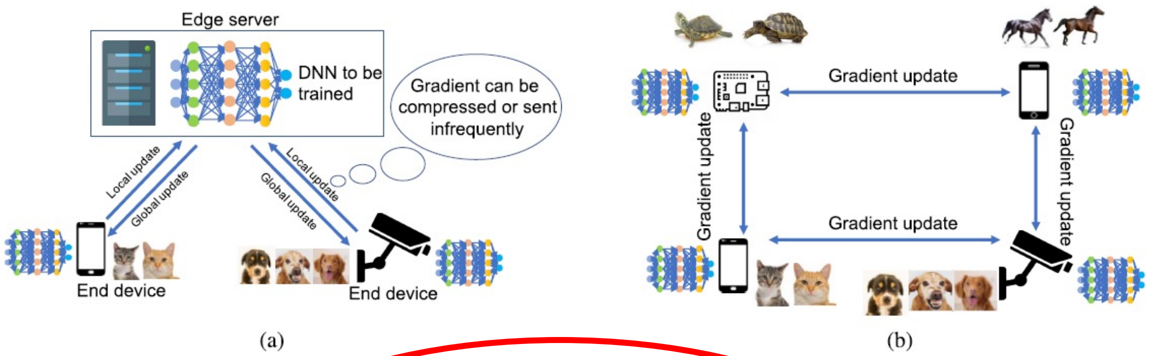
Cerebras Systems: Achieving Industry Best AI Performance Through A Systems Approach [2021]

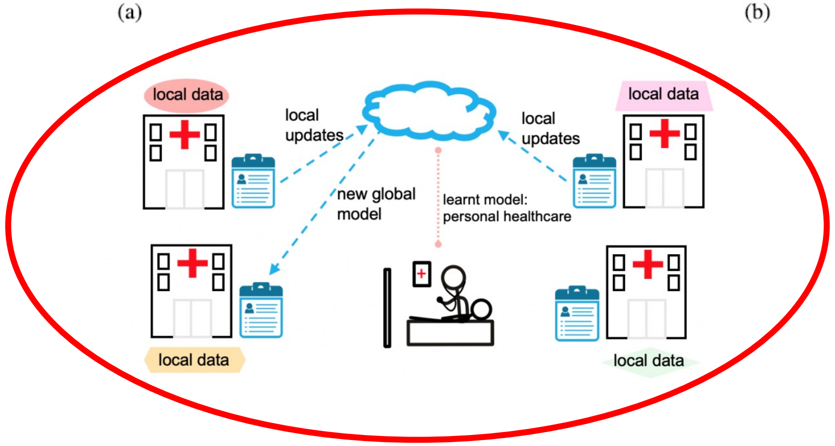
Observations

- Custom **software stack** and not just hardware
- Exploit parallelism everywhere - Batched inference and distributed training
- Generic acceleration (Accelerator Startups)
- Abstraction of programming (Accelerator startups)
- Resurgence of explicit dataflow architectures.
- Large on-chip memories (memory wall is real: PIM solutions)
- Value pinning and reconfigurable processing cores/programmable cores
- On-chip interconnects and off-chip networks - Scalability with model scaling
- Custom data types
- Some on-chip accelerators due to customer needs.

AI/ML accelerators for edge computing







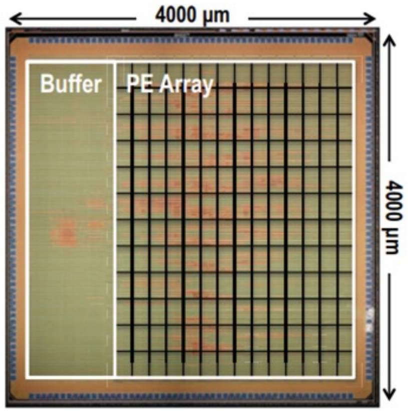
Key Metrics for edge/embedded AI/ML accelerators

- TOPS/W (Tera Operations per Watt)
- Accuracy
  - Quality of results
  - Throughput – analytics on high volume data
  - Latency – autonomous navigation
- Hardware cost
  - Designing and Manufacturing
- Flexibility
  - Due to evolving DNN models
- Scalability
- Privacy

Hardware for Machine Learning: Challenges and Opportunities[2017], Sze et el.

Eyeriss

- A very tiny low power AI accelerator chip ~278nW
- Is a spatial architecture
  - Use dataflow processing
  - To facilitate efficient data reuse
- Focused on accelerating DNN
- Identified convolution is the most important
  - Takes 90% - 99% of computation and runtime



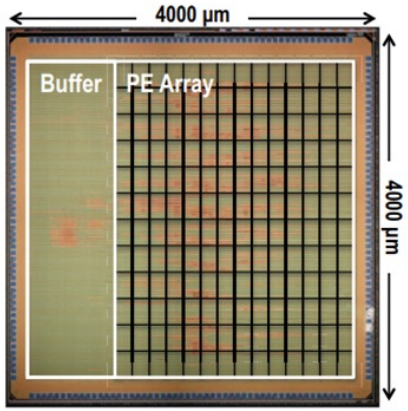
Process	65nm CMOS
# of PEs	168
RF Size/PE	0.5 kB
Buffer Size	108 kB
Clock Rate	200 MHz
Precision	16-bit Fixed-Point
Area	~278nW

Figure 4. Die photo and spec of the Eyeriss chip [41].

Eyeriss: A Spatial Architecture for Energy-Efficient Dataflow for Convolutional Neural Networks[2016], Sze et al.

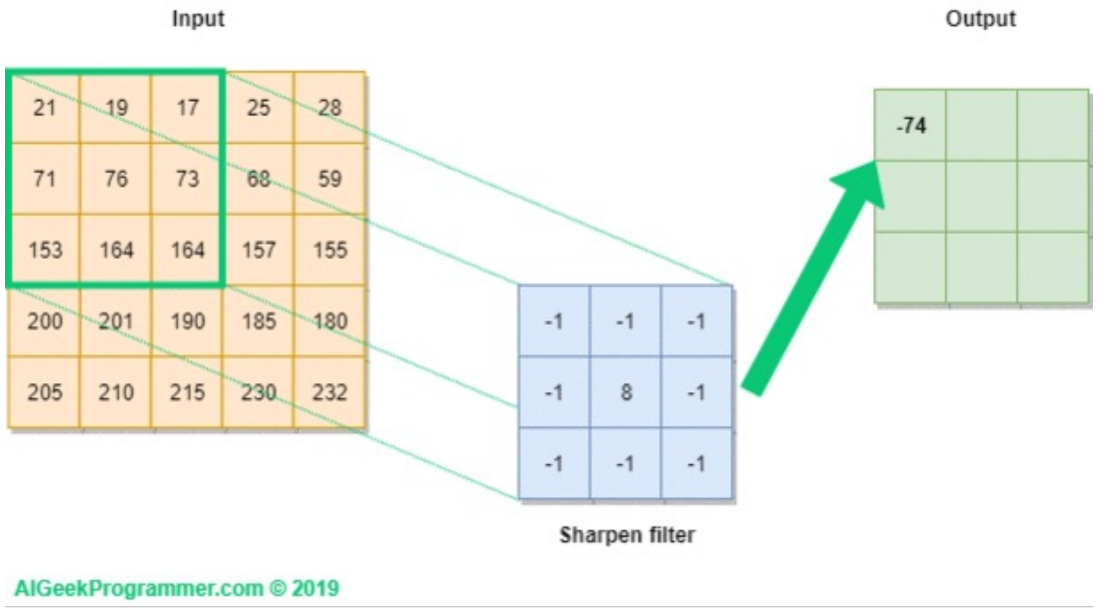
Eyeriss

- A very tiny low power AI accelerator chip ~278nW
- Is a spatial architecture
  - Use dataflow processing
  - To facilitate efficient data reuse
- Focused on accelerating DNN
- Identified convolution is the most important
  - Takes 90% - 99% of computation and runtime



Process	65nm CMOS
# of PEs	168
RF Size/PE	0.5 kB
Buffer Size	108 kB
Clock Rate	200 MHz
Precision	16-bit Fixed-Point
Area	~278nW

Figure 4. Die photo and spec of the Eyeriss chip [41].



Eyeriss: A Spatial Architecture for Energy-Efficient Dataflow for Convolutional Neural Networks[2016], Sze et al.

Eyeriss

- Dataflow implementation
  - Output Stationary
  - Weight Stationary
  - No Local data reuse
    - All the input and filter data come from global buffer
    - All the partial sums and output written to global buffer
- Eyeriss proposed novel dataflow implementation
  - Row-stationary approach
  - High input reuse (filter, feature maps)
  - Minimize partial sum accumulation cost

Eyeriss v2: A Flexible Accelerator for Emerging Deep Neural Networks on Mobile Devices [2018], Chen Y, et al

Eyeriss

- Dataflow implementation
  - Output Stationary
  - Weight Stationary
  - No Local data reuse
    - All the input and filter data come from global buffer
    - All the partial sums and output written to global buffer
- Eyeriss proposed novel dataflow implementation
  - Row-stationary approach
  - High input reuse (filter, feature maps)
  - Minimize partial sum accumulation cost

Eyeriss V2

- Designed for sparse and compact DNN models

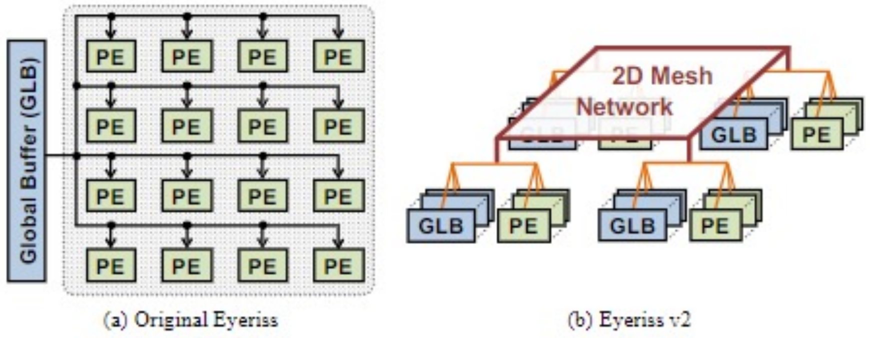


Fig. 5. Comparison of the architecture of original Eyeriss and Eyeriss v2.

Eyeriss v2: A Flexible Accelerator for Emerging Deep Neural Networks on Mobile Devices [2018], Chen Y, et al

Eyeriss

- Dataflow implementation
  - Output Stationary
  - Weight Stationary
  - No Local data reuse
    - All the input and filter data come from global buffer
    - All the partial sums and output written to global buffer
- Eyeriss proposed novel dataflow implementation
  - Row-stationary approach
  - High input reuse (filter, feature maps)
  - Minimize partial sum accumulation cost

Eyeriss V2

- Designed for sparse and compact DNN models
- Proposed highly on-chip network, called hierarchy mesh
  - To adapt to different amount of data-reuse, bandwidth, and utilization

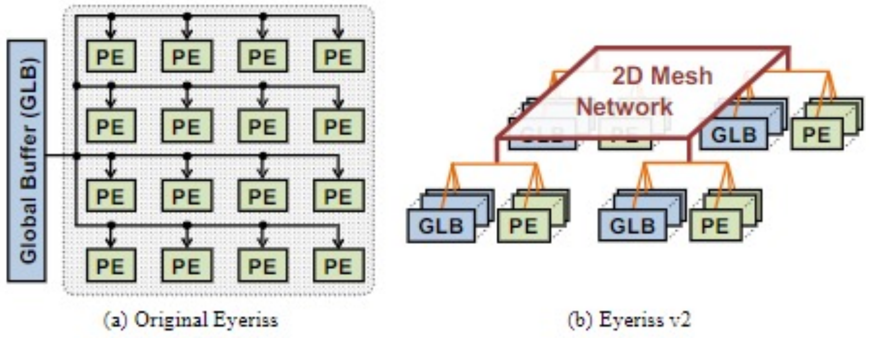


Fig. 5. Comparison of the architecture of original Eyeriss and Eyeriss v2.

Eyeriss v2: A Flexible Accelerator for Emerging Deep Neural Networks on Mobile Devices [2018], Chen Y, et al

Eyeriss

- Dataflow implementation
  - Output Stationary
  - Weight Stationary
  - No Local data reuse
    - All the input and filter data come from global buffer
    - All the partial sums and output written to global buffer
- Eyeriss proposed novel dataflow implementation
  - Row-stationary approach
  - High input reuse (filter, feature maps)
  - Minimize partial sum accumulation cost

Eyeriss V2

- Designed for sparse and compact DNN models
- Proposed highly on-chip network, called hierarchy mesh
  - To adapt to different amount of data-reuse, bandwidth, and utilization
- 12.6X faster and 2.5X energy efficient than the original Eyeriss

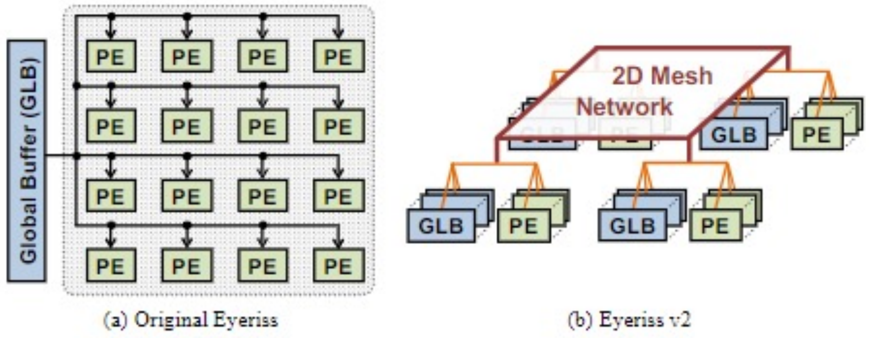


Fig. 5. Comparison of the architecture of original Eyeriss and Eyeriss v2.

Eyeriss v2: A Flexible Accelerator for Emerging Deep Neural Networks on Mobile Devices [2018], Chen Y, et al

Eyeriss

- Dataflow implementation
  - Output Stationary
  - Weight Stationary
  - No Local data reuse

reduce data movement, storage, and computation,
reuse as much as possible

- implementation
  - Row-stationary approach
  - High input reuse (filter, feature maps)
  - Minimize partial sum accumulation cost

Eyeriss V2

- Designed for sparse and compact DNN models
- Proposed highly on-chip network, called hierarchy mesh

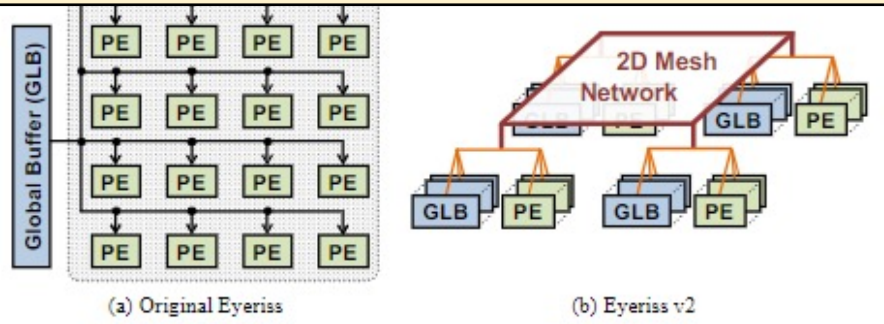


Fig. 5. Comparison of the architecture of original Eyeriss and Eyeriss v2.

Eyeriss v2: A Flexible Accelerator for Emerging Deep Neural Networks on Mobile Devices [2018], Chen Y, et al

Minerva

Minerva: Enabling Low-Power, Highly-Accurate Deep Neural Network Accelerators[2016], Reagen B. et al.

Minerva

- Designed to deploy DNNs in power-constrained environment

Minerva: Enabling Low-Power, Highly-Accurate Deep Neural Network Accelerators[2016], Reagen B. et al.

Minerva

- Designed to deploy DNNs in power-constrained environment
- Automated co-design flow

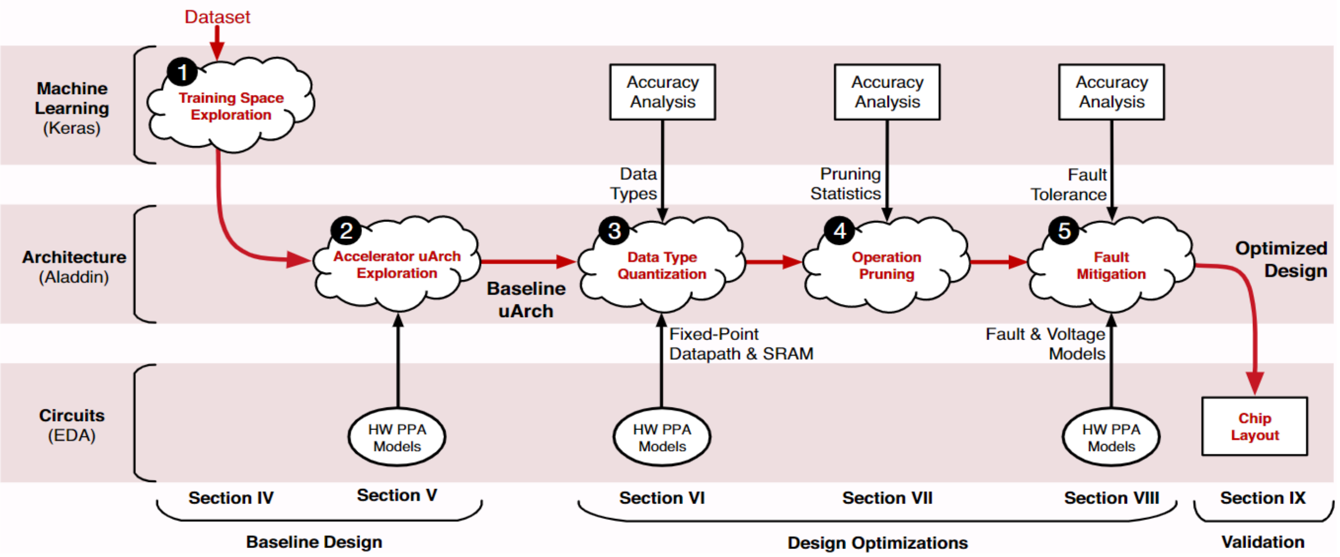


Figure 2: The five stages of Minerva. Analysis details for each stage and the tool-chain are presented in Section 3.

Minerva: Enabling Low-Power, Highly-Accurate Deep Neural Network Accelerators[2016], Reagen B. et al.

Minerva

- Designed to deploy DNNs in power-constrained environment
- Automated co-design flow

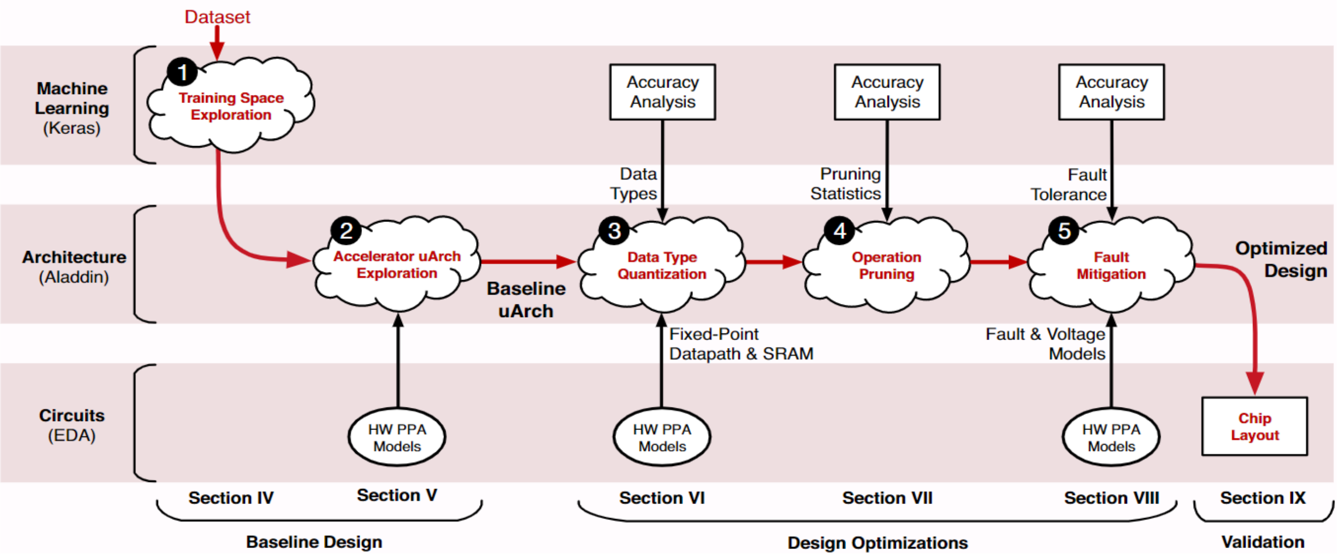


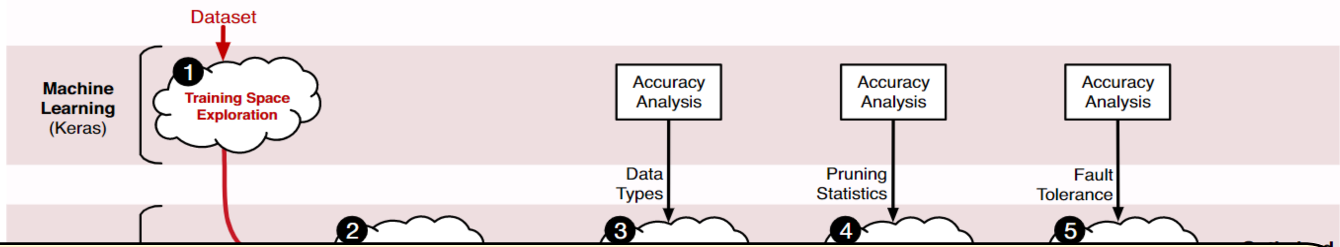
Figure 2: The five stages of Minerva. Analysis details for each stage and the tool-chain are presented in Section 3.

- Power reduction by using
  - Fine-grained, heterogenous data type optimization
  - Selective pruning
  - Lowering SRAM voltage and domain-aware fault mitigation

Minerva: Enabling Low-Power, Highly-Accurate Deep Neural Network Accelerators[2016], Reagen B. et al.

Minerva

- Designed to deploy DNNs in power-constrained environment
- Automated co-design flow



Approximate to reduce power consumption within the required accuracy limits

Figure 2: The five stages of Minerva. Analysis details for each stage and the tool-chain are presented in Section 3.

- Power reduction by using
  - Fine-grained, heterogenous data type optimization
  - Selective pruning
  - Lowering SRAM voltage and domain-aware fault mitigation

Minerva: Enabling Low-Power, Highly-Accurate Deep Neural Network Accelerators[2016], Reagen B. et al.

AI-RISC processor

AI-RISC: Scalable RISC-V Processor for IoT Edge AI applications[2022], Vaibhav V.

AI-RISC processor

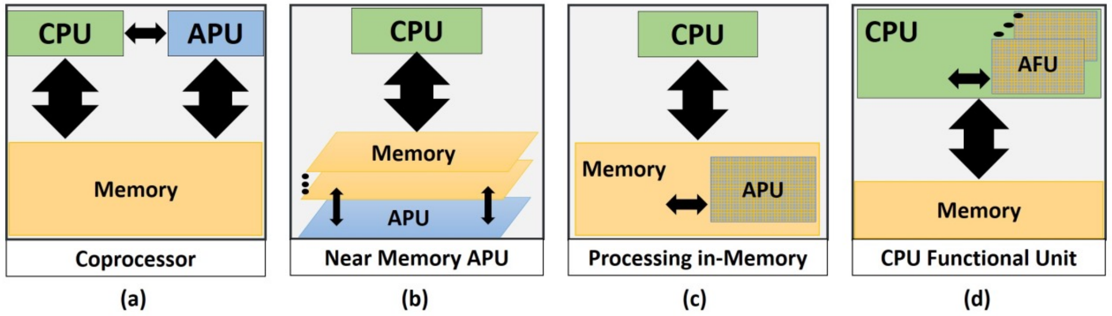


Figure 1.4: Different AI Processing Unit (APU) types and their integration with the CPU hardware

AI-RISC: Scalable RISC-V Processor for IoT Edge AI applications[2022], Vaibhav V.

AI-RISC processor

- Introduces AFU: Tightly integrated **A**I **F**unctional **U**nits

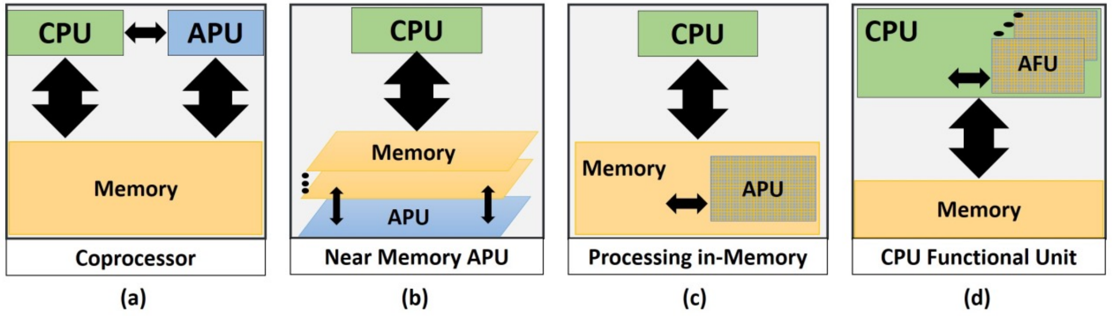


Figure 1.4: Different AI Processing Unit (APU) types and their integration with the CPU hardware

AI-RISC: Scalable RISC-V Processor for IoT Edge AI applications[2022], Vaibhav V.

AI-RISC processor

- Introduces AFU: Tightly integrated **A**I **F**unctional **U**nits
- Inspired from history of adding instructions and functional units for commonly used operations
  - Ex: floating-point arithmetic became popular and eventually we have floating point instructions and units inside the processor

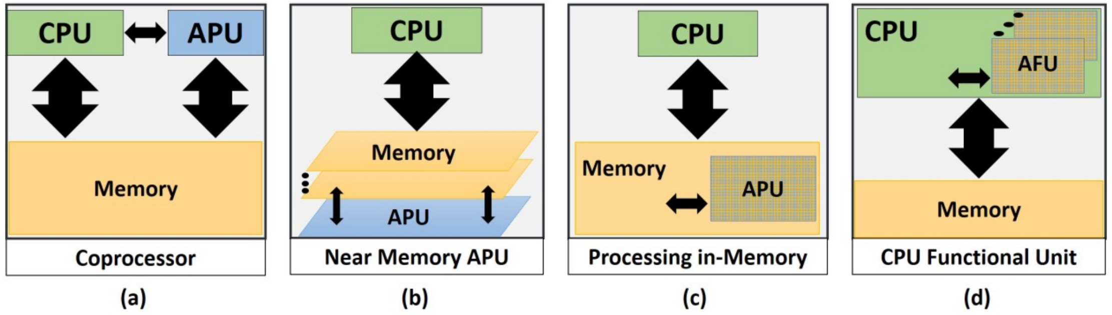


Figure 1.4: Different AI Processing Unit (APU) types and their integration with the CPU hardware

AI-RISC: Scalable RISC-V Processor for IoT Edge AI applications[2022], Vaibhav V.

AI-RISC processor

- Introduces AFU: Tightly integrated **A**I **F**unctional **U**nits
- Inspired from history of adding instructions and functional units for commonly used operations
  - Ex: floating-point arithmetic became popular and eventually we have floating point instructions and units inside the processor
- Reduces complexity required to design
  - bus interfaces
  - separate ISA for a decoupled AI accelerator

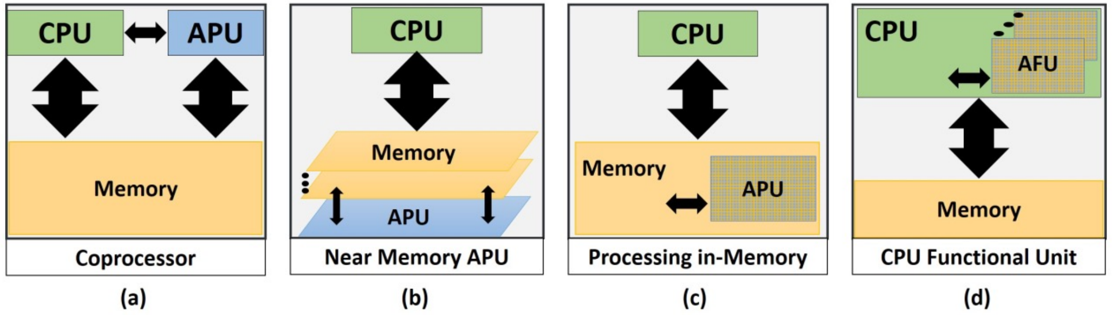


Figure 1.4: Different AI Processing Unit (APU) types and their integration with the CPU hardware

AI-RISC: Scalable RISC-V Processor for IoT Edge AI applications[2022], Vaibhav V.

AI-RISC processor

- Introduces AFU: Tightly integrated <u>A</u>I <u>F</u>unctional <u>U</u>nits
- Inspired from history of adding instructions and functional units for commonly used operations
  - Ex: floating-point arithmetic became popular and eventually we have floating point instructions and units inside the processor
- Reduces complexity required to design
  - bus interfaces
  - separate ISA for a decoupled AI accelerator

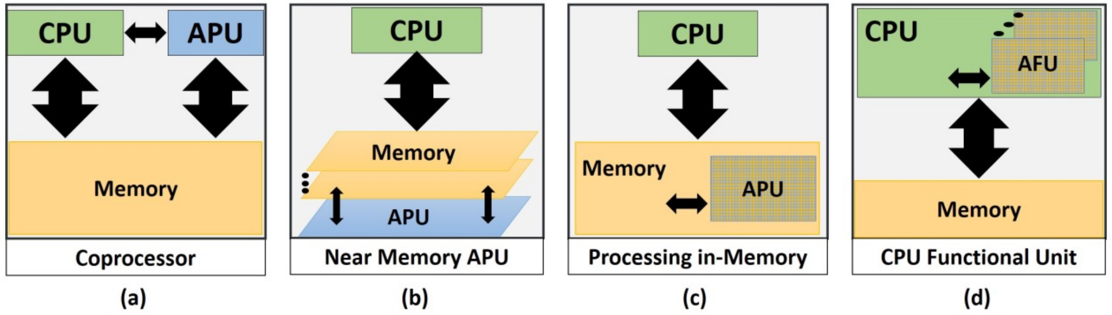


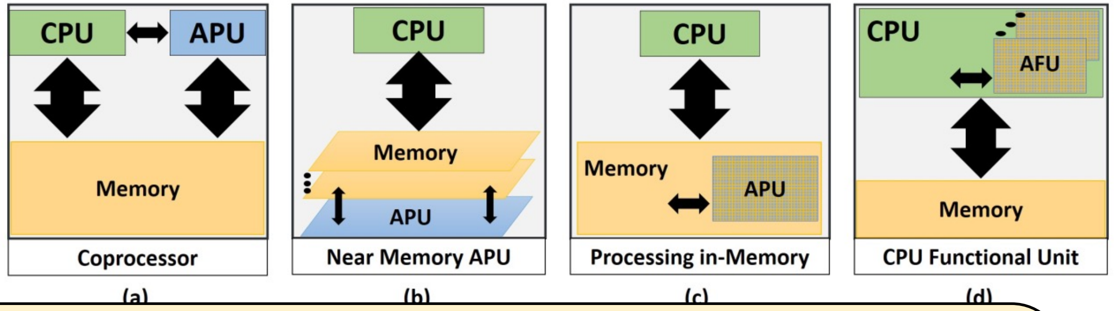
Figure 1.4: Different AI Processing Unit (APU) types and their integration with the CPU hardware

- Such tight integration is good for edge but not for cloud
  - While data-centers AI accelerators run models with more than 100 billion parameters
  - tinyML has introduced much smaller models for edge devices i.e. around 100K

AI-RISC: Scalable RISC-V Processor for IoT Edge AI applications[2022], Vaibhav V.

AI-RISC processor

- Introduces AFU: Tightly integrated **A**I **F**unctional **U**nits
- Inspired from history of adding instructions and functional units for commonly used operations



Specialized ISAs are a good option

- design
  - bus interfaces
  - separate ISA for a decoupled AI accelerator

models with more than 100 billion parameters

- tinyML has introduced much smaller models for edge devices i.e. around 100K

AI-RISC: Scalable RISC-V Processor for IoT Edge AI applications[2022], Vaibhav V.

AI-RISC Processor (contd.)

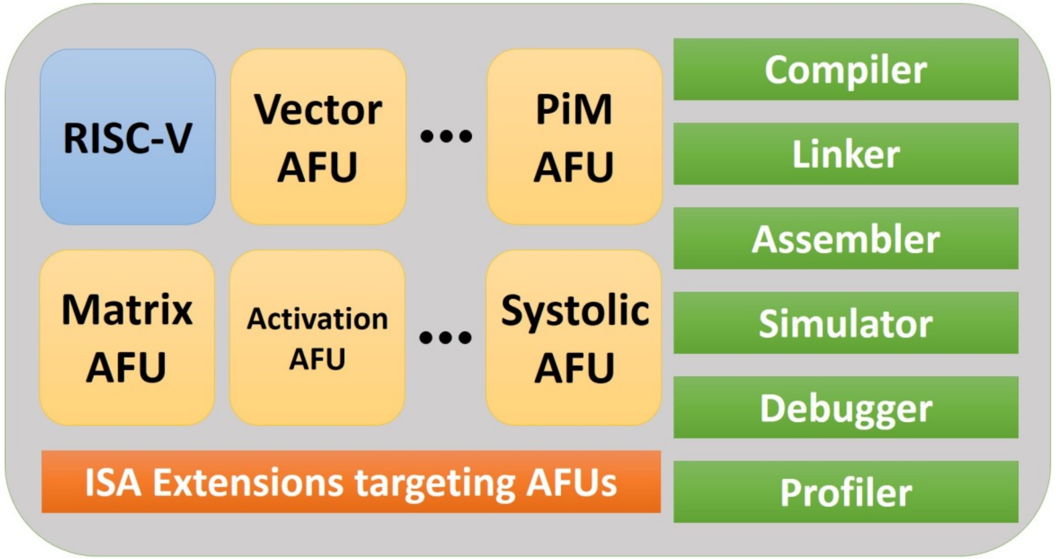


Figure 1.5: Top level overview of the AI-RISC processor proposed in this dissertation .

- Issues with ISA-extension
  - Compatibility across the stack
  - Need joint efforts by the community
  - Agile design infrastructure
- Issues with scalability
- Future work
  - Tape-out AI RISC processor
  - Integrate it with LiteX (open source SoC builder)
  - End-to-end framework from DSL to GDSII layout

Federated Learning in Edge

- **Federated learning**
- Preserve data-privacy
- Less data movement
- More efficient/less power consumption

- Can be implemented using coarse grained reconfigurable array (CGRA) that act like a bunch of mini-accelerators and can be dynamically configured for a particular use case
  - Prof. David Atienza (EPFL)

Observation

Observation

Critical path reduction

- Eyeriss identified the most common operation – convolution
- Identified the bottleneck -> it's the data movement, solved it by exploring strategies of data-reuse
- Eyeriss V2 discussed about a novel NoC approach to keep the PE's utilization high
- Minerva tried to optimize the MAC operations by compressing the data, pruning the data : remove the operations that don't impact the results dramatically
- Find a way to integrate it with the existing or new framework
- AI-RISC is trying to build an end-to-end framework from quick tape-out of AI chips from the Domain specific language description

Major dilemmas

Generalization ⟷	Specialization
Bring data to compute ⟷	Bring compute to data
Specialized Components ⟷	Specialized Instructions
Online learning ⟷	Offline learning

Major dilemmas

Generalization	Specialization
Bring data to compute	Bring compute to data
Specialized Components	Specialized Instructions
Online learning	Offline learning

Major dilemmas

Generalization ⟷ Specialization

Bring data to compute ⟷ Bring compute to data

Specialized Components ⟷ Specialized Instructions

Online learning ⟷ Offline learning

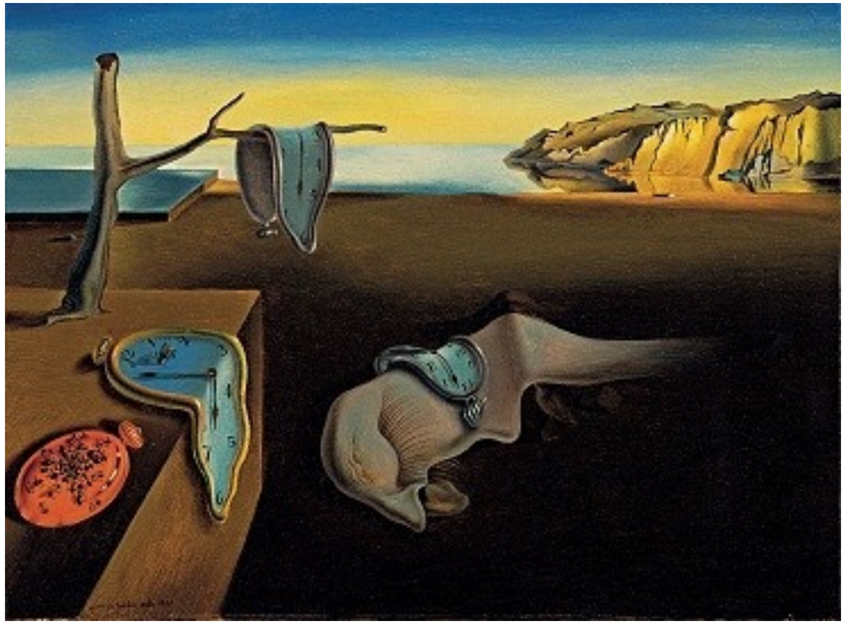
Decisions to be made based on trade-offs and workloads

Design Space Exploration

One ring that unites and rules them all

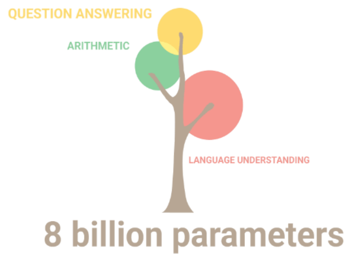
- Aladdin
  - A Pre-RTL, Power-Performance Accelerator Simulator Enabling Large Design Space Exploration of Customized Architecture
  - For all types of accelerator
- Scale-SIM (specially for AI accelerators)
- Accelergy – early stage energy estimator
- Timeloop – Performance simulator and DNN mapping tool
- PIMulator-NN - Simulator designed for PiM based NN accelerators
- Once you have designed, how do you integrate, so we also have
  - Open source RISC-V toolchains
  - ESP (more flexible than chipyard)
  - Chipyard, Firesim

Past and Present



- Cambrian explosion of Accelerators
  - Novel techniques
  - Solutions across the stack
- Diverse target application and workloads
- Limited works on data privacy
- Carbon footprint of computing! Large datacenters chugging power.
- Reinventing the wheel multiple times
- Missing a community effort (industry-academia gap)
  - Proprietary solutions in the industry
  - Novel ideas from academia unadopted

Future



- Larger models require more data (>540B parameters) - PIM is one solution
- Collaboration between within Academia and Industry
  - Privacy aware ML models and architectures
  - Homomorphic encryption
  - Better ML algorithms and newer technologies - Spiking Neural Networks
  - HW aware Neural Architecture Search (NetAdapt)
  - Improved process nodes and processing paradigms (eFPGA, low power CGRAs).
- Environmental impact of Computing needs to be studied - global warming!
- Inference and training at scale
- The AI Accelerator Wall?
  - Are we relying on performance through transistor scaling?
  - What is the performance boost that we get from intelligent design?

Questions?

Thank you!