NATIONAL RESEARCH UNIVERSITY

Computer Architecture and Operating Systems
Lecture 16: Domain-specific architectures. Tensor Processing Unit.

Andrei Tatarnikov

atatarnikov@hse.ru
@andrewt0301

Motivation

- Modern performance tuning techniques:
  - Deep memory hierarchy
  - Wide SIMD units
  - Deep pipelines
  - Branch prediction
  - Out-of-order execution
  - Speculative prefetching
  - Multithreading
  - Multiprocessing
- Further improvement:
  - Domain-specific architectures

Guidelines for DSAs

- Use **dedicated memories** to minimize data movement
- Invest resources into **more arithmetic units** or bigger memories
- Use the easiest form of **parallelism** that matches the domain
- Reduce **data size and type** to the simplest needed for the domain
- Use a **domain-specific** programming language

Deep Neural Networks

- Inpired by neuron of the brain
- Computes non-linear “activation” function of the weighted sum of input values
- Neurons arranged in layers
- Most practitioners will choose an existing design
  - Topology
  - Data type
- Training (learning):
  - Calculate weights using backpropagation algorithm
  - Supervised learning: stochastic graduate descent
- Inference: use neural network for classification

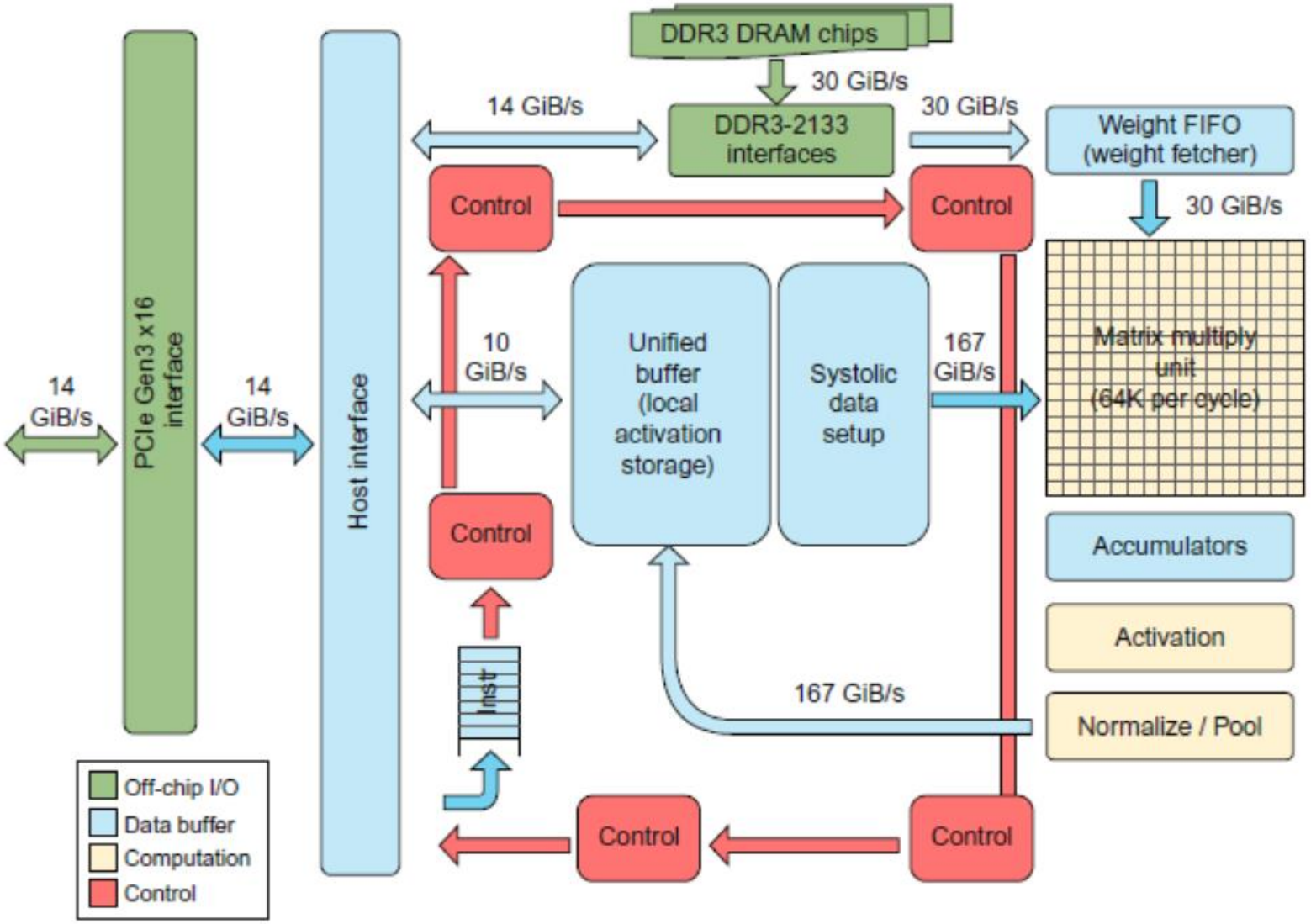
Convolutional Neural Network

- Batches:
  - Reuse weights once fetched from memory across multiple inputs
  - Increases operational intensity
- Quantization
  - Use 8- or 16-bit fixed point
- Summary:
  - Need the following kernels:
    - Matrix-vector multiply
    - Matrix-matrix multiply
    - Stencil
    - ReLU
    - Sigmoid
    - Hyperbolic tangeant

Tensor Processing Unit

- Google's DNN ASIC
- 256 x 256 8-bit matrix multiply unit
- Large software-managed scratchpad
- Coprocessor on the PCIe bus

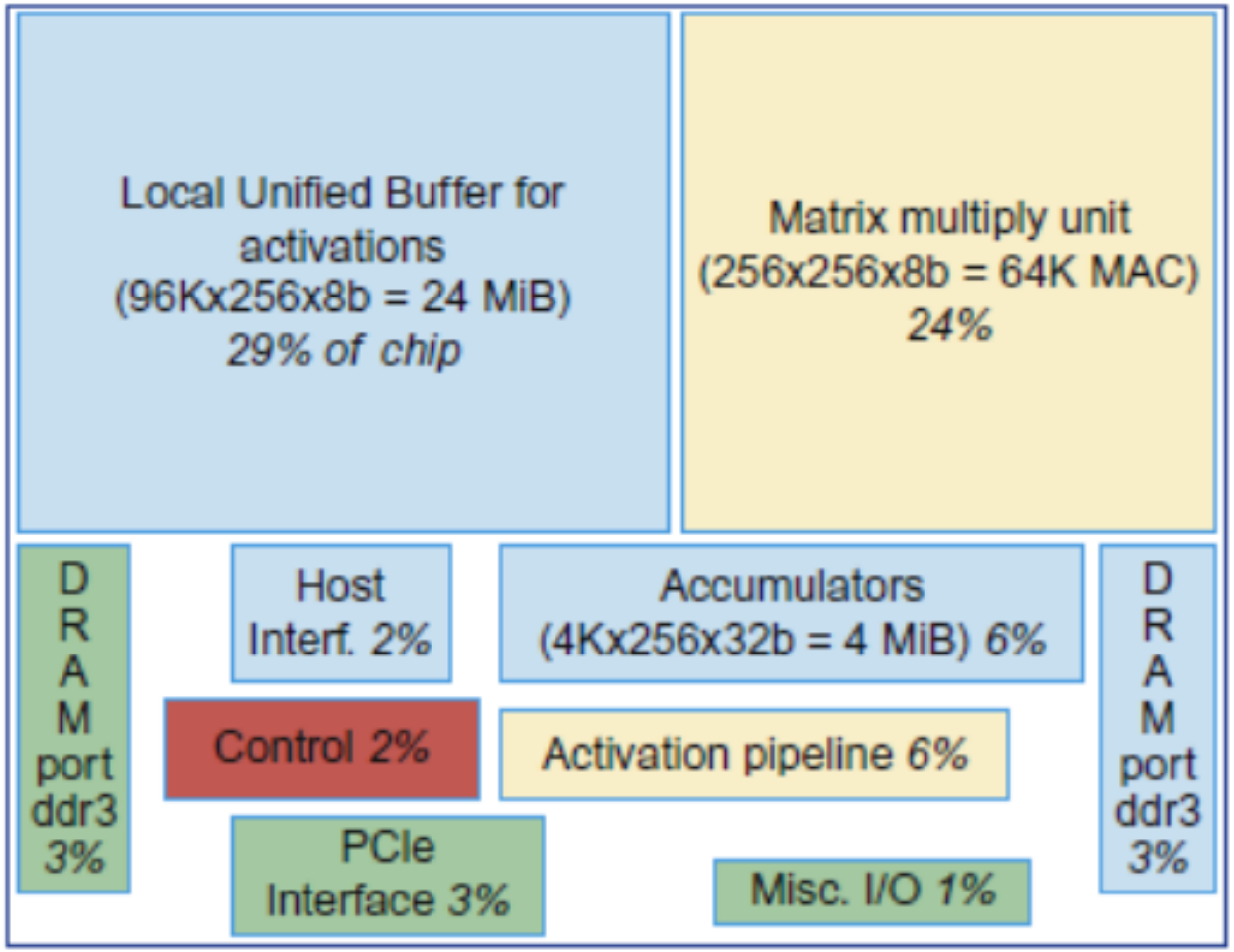
Tensor Processing Unit

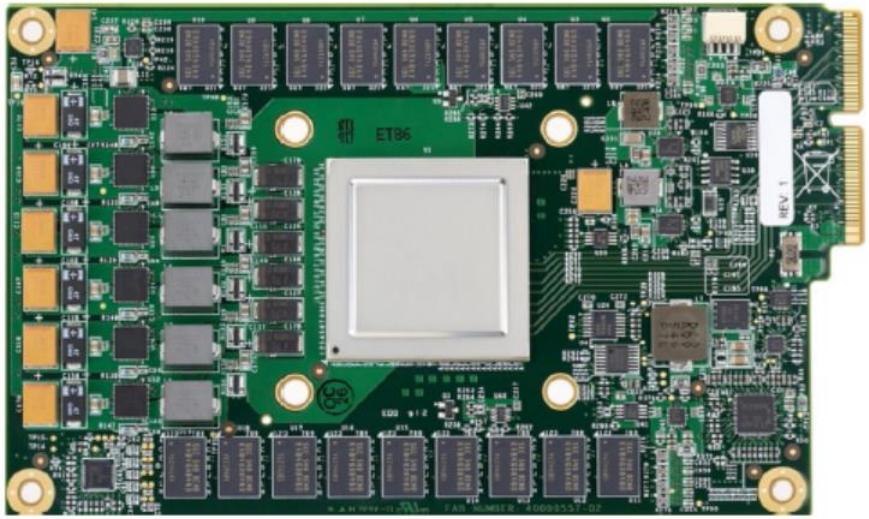


TPU ISA

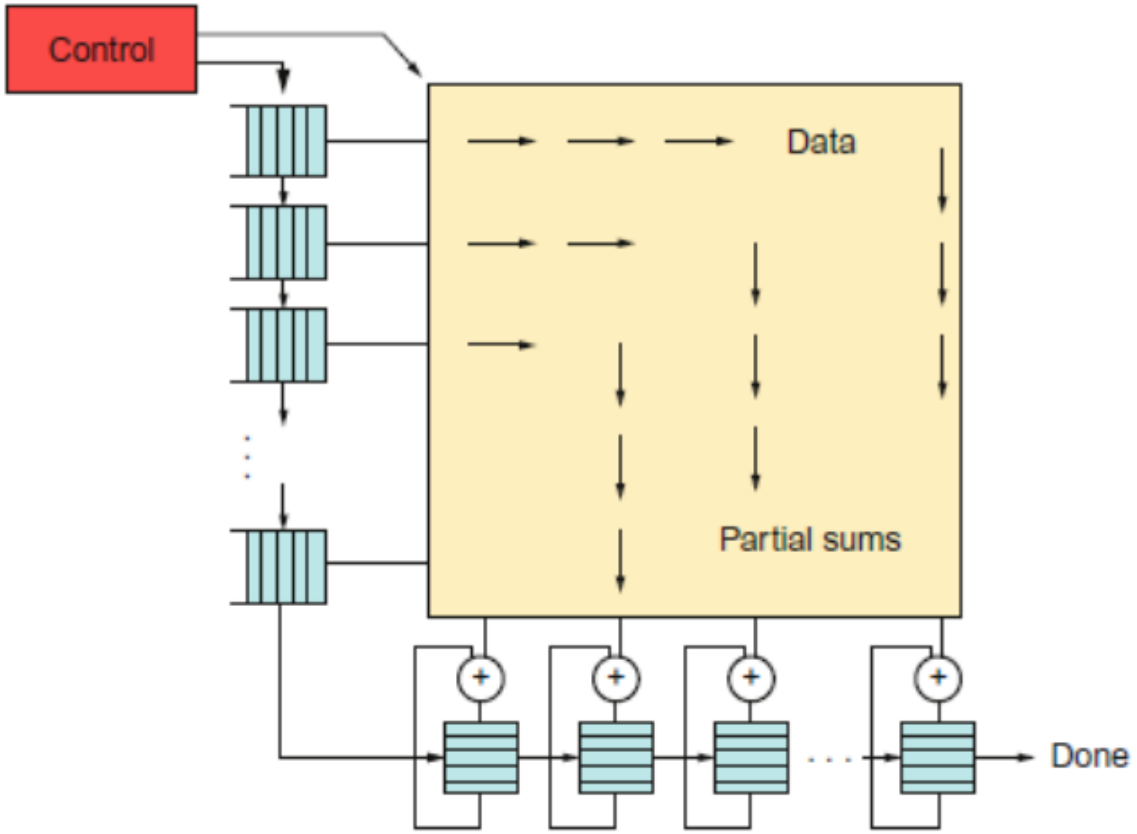
- **Read_Host_Memory**
  - Reads memory from the CPU memory into the unified buffer
- **Read_Weights**
  - Reads weights from the Weight Memory into the Weight FIFO as input to the Matrix Unit
- **MatrixMatrixMultiply/Convolve**
  - Perform a matrix-matrix multiply, a vector-matrix multiply, an element-wise matrix multiply, an element-wise vector multiply, or a convolution from the Unified Buffer into the accumulators
  - takes a variable-sized B*256 input, multiplies it by a 256x256 constant input, and produces a B*256 output, taking B pipelined cycles to complete
- **Activate**
  - Computes activation function
- **Write_Host_Memory**
  - Writes data from unified buffer into host memory

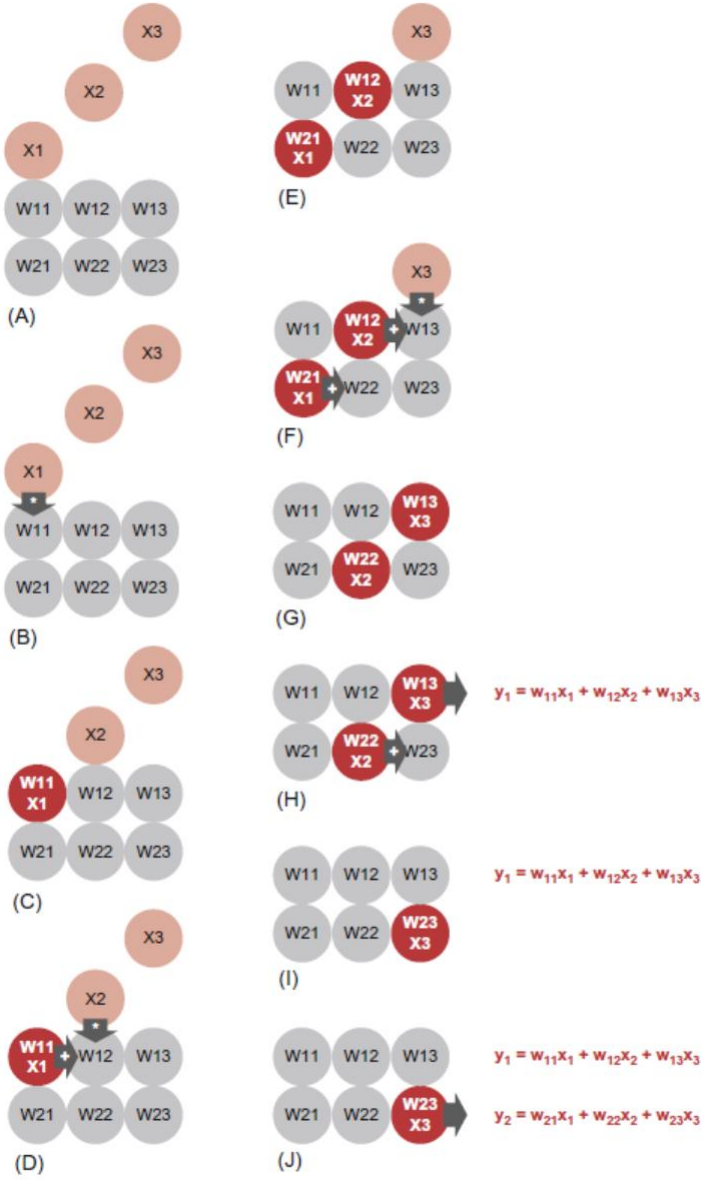
Tensor Processing Unit





TPU ISA


Control
Data
Partial sums
Done


X3
X2
X1
W11 W12 W13
W21 W22 W23
(A)
(B)
(C)
(D)
(E)
(F)
(G)
(H)
(I)
(J)
$y_1 = w_{11}x_1 + w_{12}x_2 + w_{13}x_3$
$y_2 = w_{21}x_1 + w_{22}x_2 + w_{23}x_3$

The TPU and the Guidelines

- Use dedicated memories
  - 24 MiB dedicated buffer, 4 MiB accumulator buffers
- Invest resources in arithmetic units and dedicated memories
  - 60% of the memory and 250X the arithmetic units of a server-class CPU
- Use the easiest form of parallelism that matches the domain
  - Exploits 2D SIMD parallelism
- Reduce the data size and type needed for the domain
  - Primarily uses 8-bit integers
- Use a domain-specific programming language
  - Uses TensorFlow

Any Questions?

```
            .text
__start:    addi t1, zero, 0x18
            addi t2, zero, 0x21
cycle:      beq t1, t2, done
            slt t0, t1, t2
            bne t0, zero, if_less
            nop
            sub t1, t1, t2
            j cycle
            nop
if_less:    sub t2, t2, t1
            j cycle
done:       add t3, t1, zero
```