CS 598CM: ML for Compilers and Architecture

Instructor: Charith Mendis

ILLINOIS

Brief Announcements

- **Paper Selections:** Due on today, presentations start Sept 12th
- **Paper Reviews:**
  - Due for each paper 2 days prior to the discussion date
  - Instructions will emailed before class next week
- **Campuswire:** Let's use Campuswire to discuss papers and more ML related work!

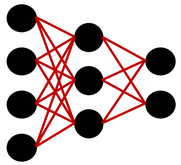
Recap

- Anatomy of a compiler optimization pass
- Domain Specific Languages

Lecture 4:
DSLs + ML in Architecture

Deep Learning Stack

Workload

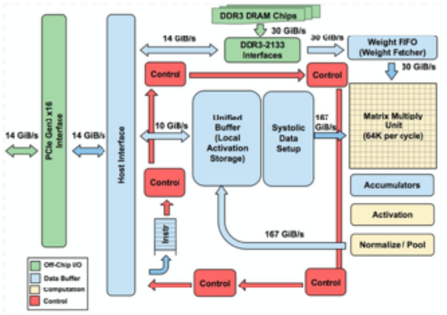


Language

TensorFlow

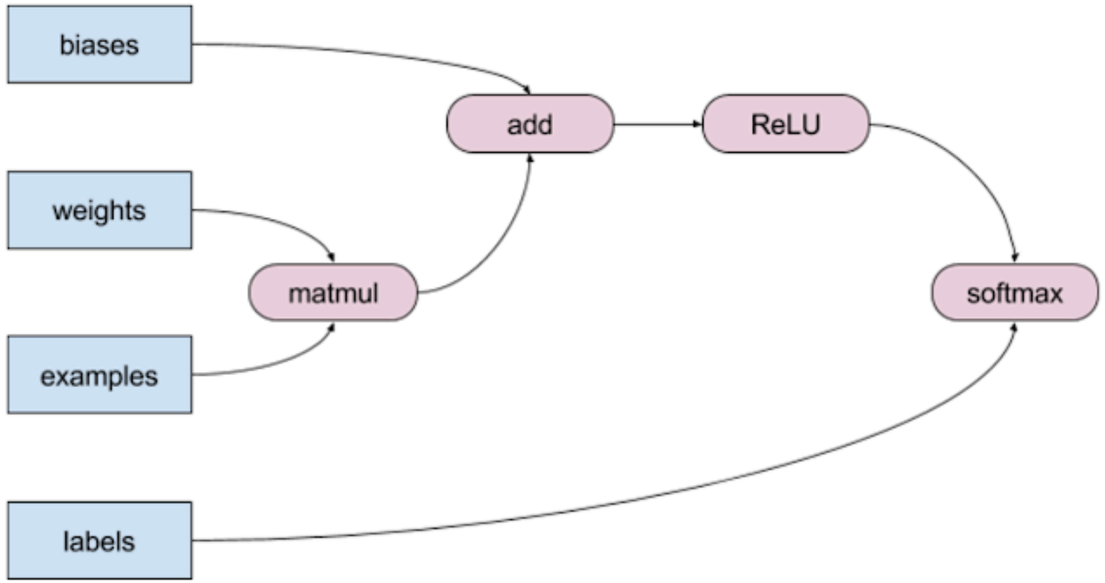
Compiler

Hardware



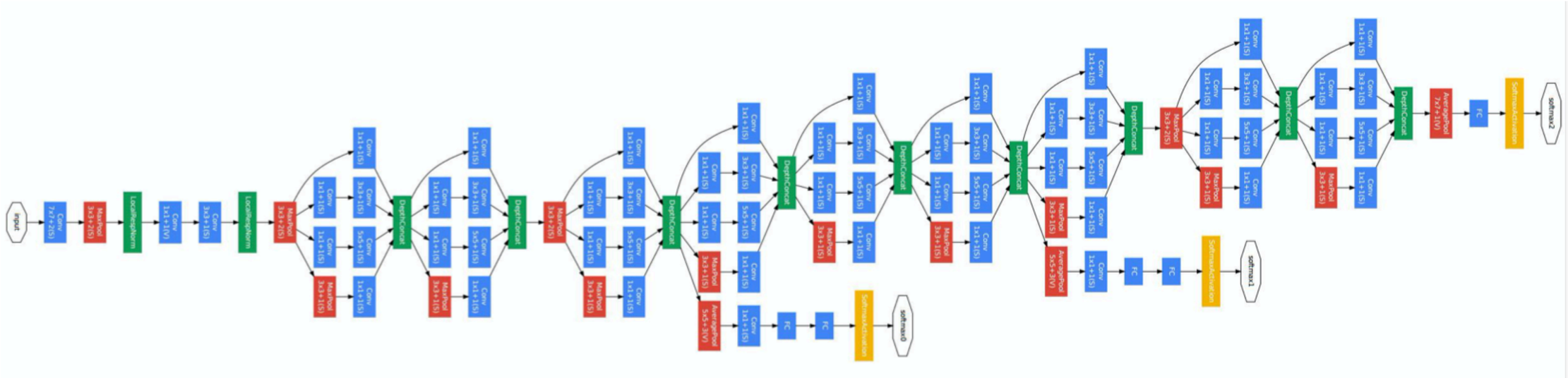
Computational Graphs

- A data-flow graph with tensor operations



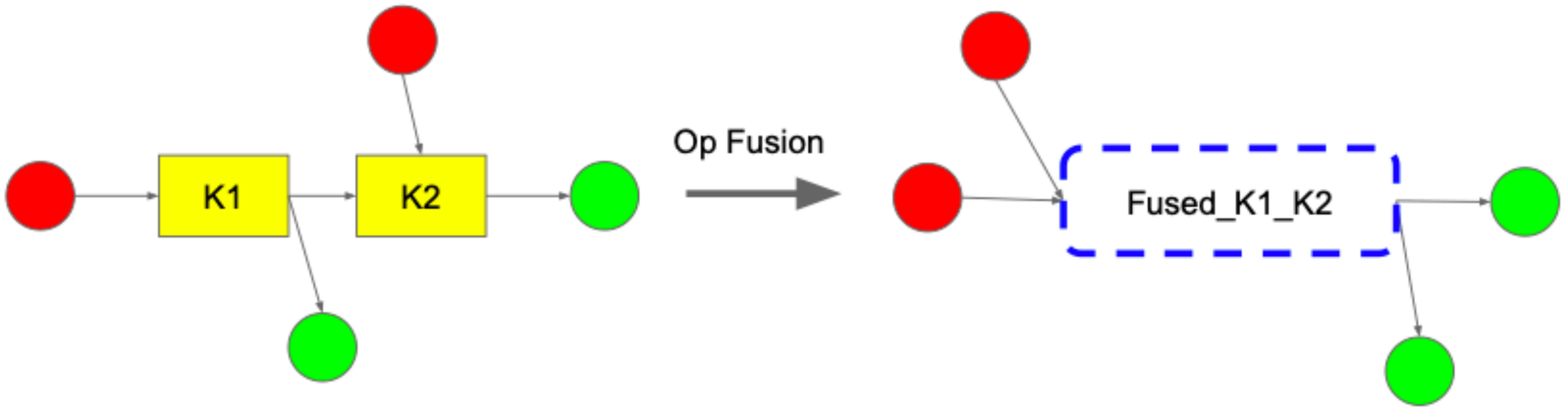
Example: Inception

Can grow to 1000s of nodes



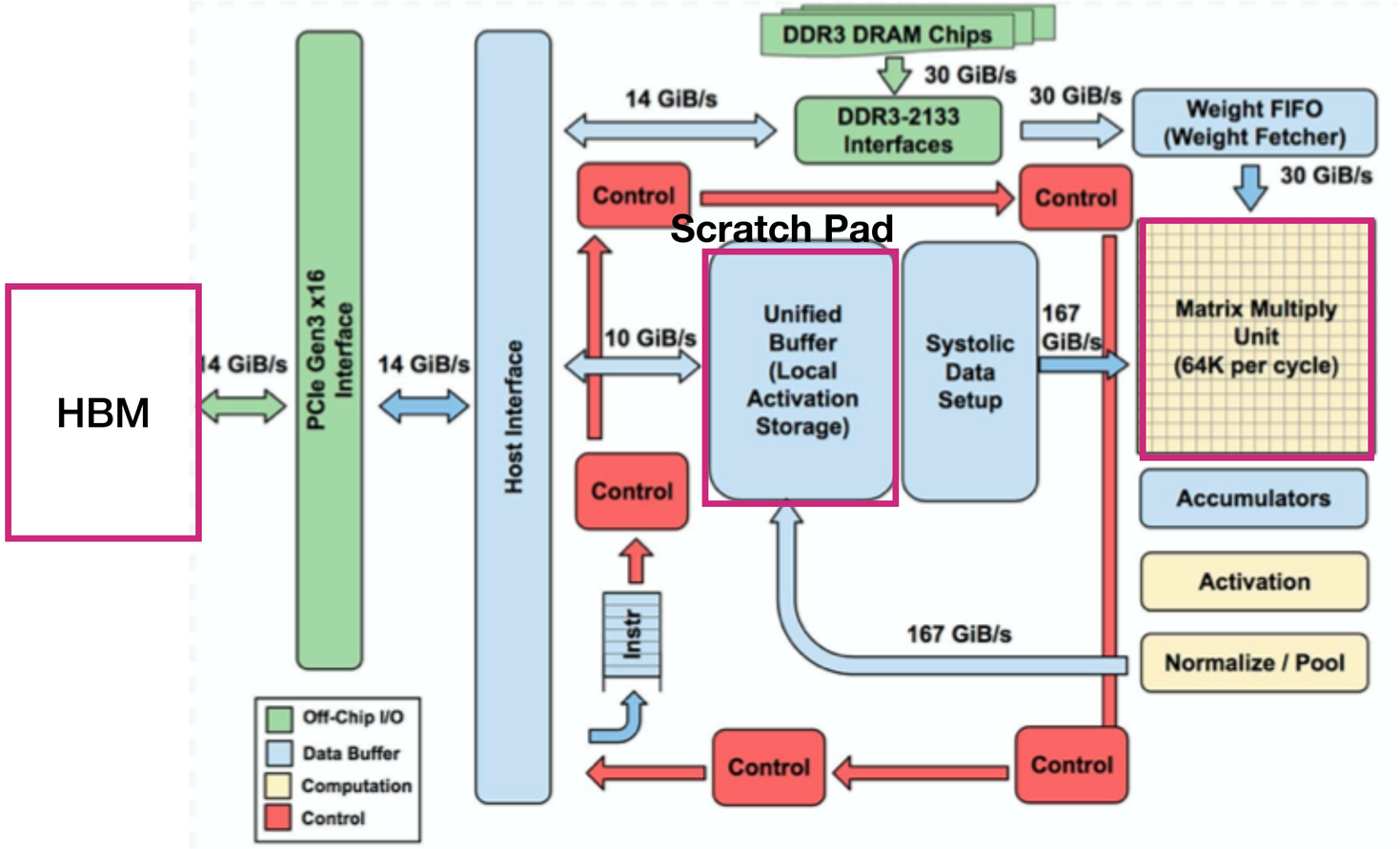
Szegedy et. al “Going Deeper with Convolutions”

Optimization: Operator Fusion



Zhou et. al “Transferable Graph Optimizers for ML Compilers”

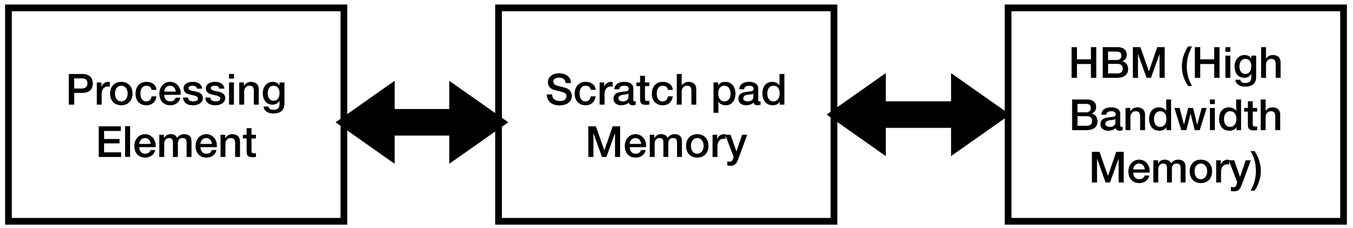
HW: Tensor Processing Unit



Processing Element

Jouppi et. al “In-Datacenter Performance Analysis of a Tensor Processing Unit”

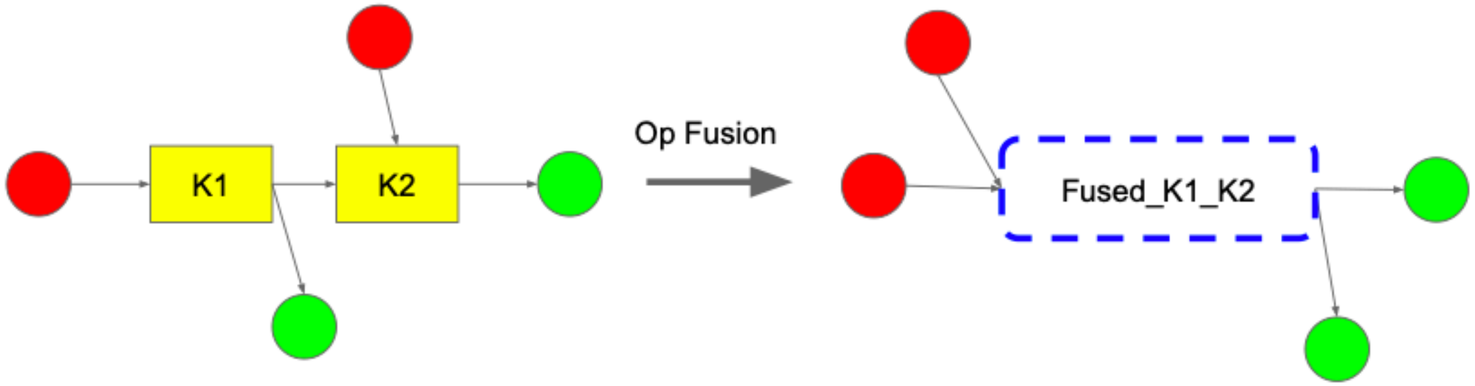
HW: Simplified Version



Size: scratch pad <<< HBM
Latency: scratch pad <<< HBM

- **Scratch pads are not caches**
- **Software Programmable**
- **Uses Direct Memory Access transfers**

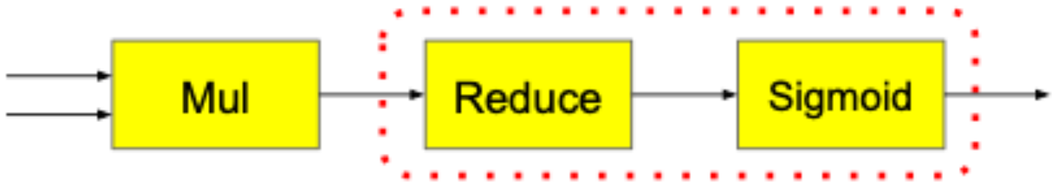
Programming Model



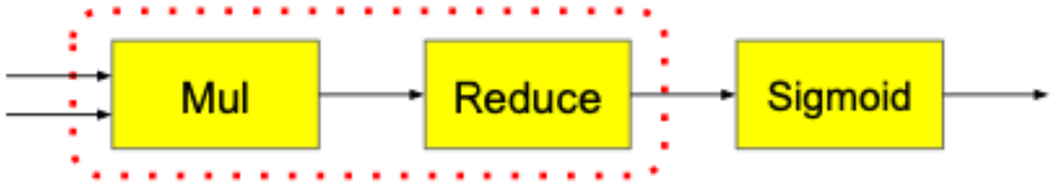
- K1 executes
- Writes back results to HBM
- K2 executes
- Writes back results to HBM

- Fused Operator Executes
- Writes back results to HBM
- Intermediates can be stored in HBM or Scratchpad

Operator Fusion



1

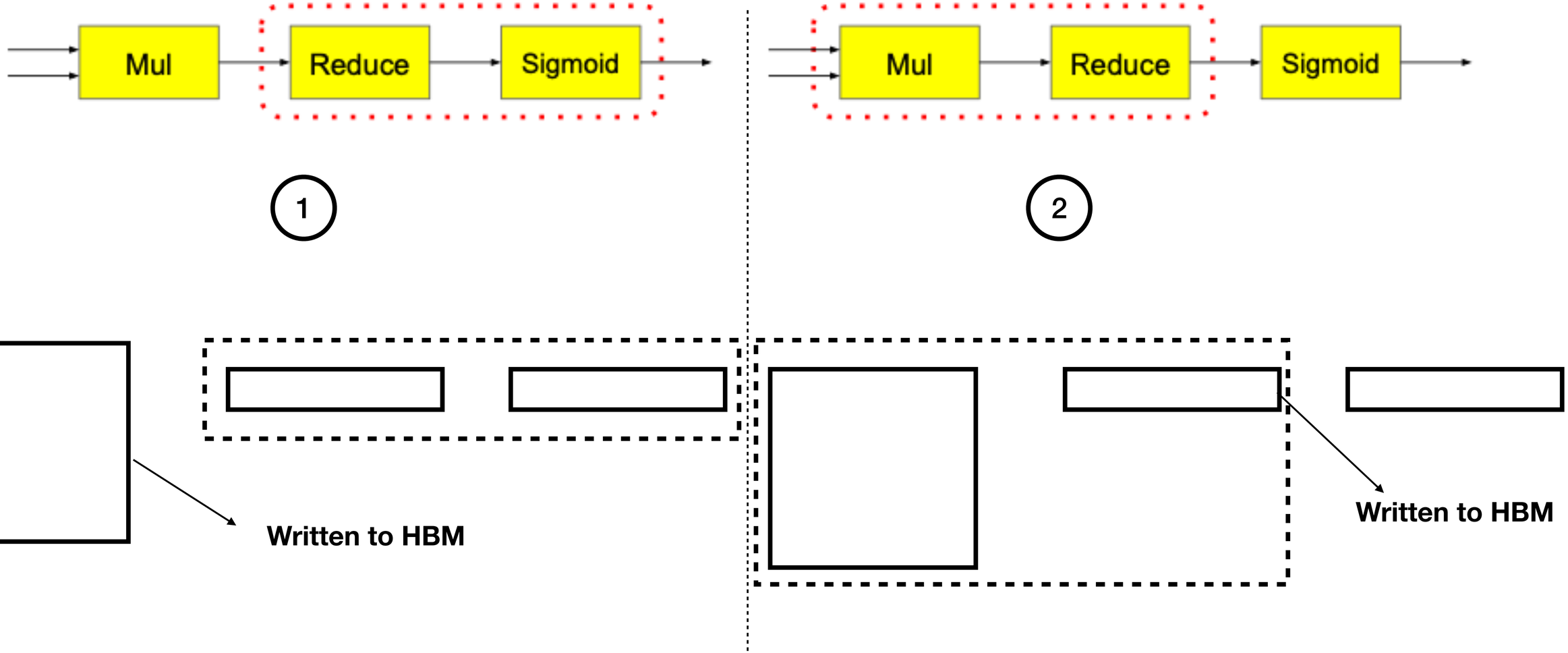


2

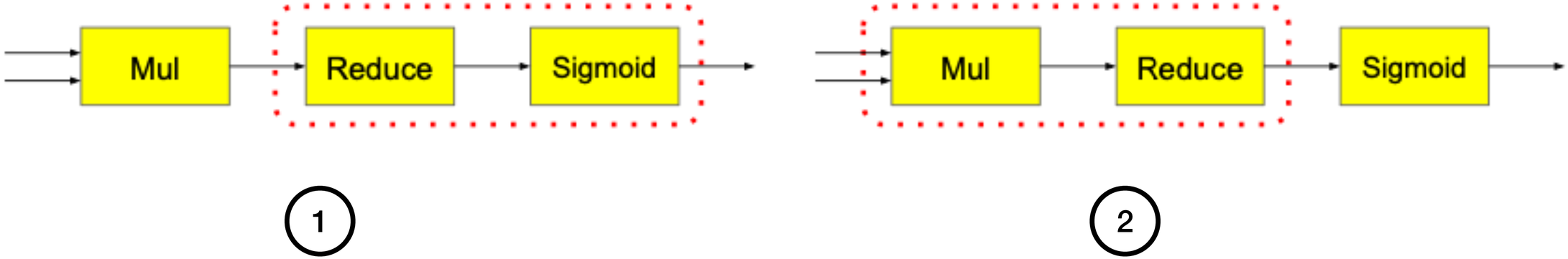
Which is faster?

Usually 2

Operator Fusion



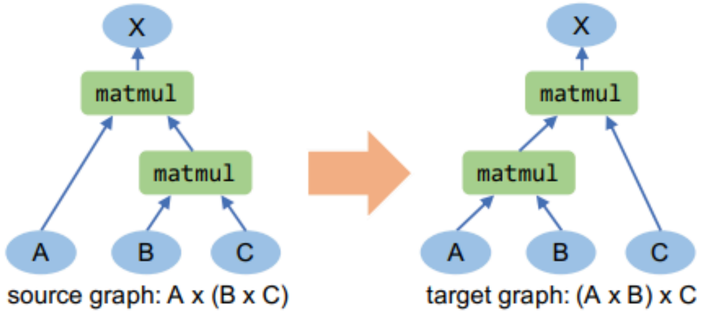
Operator Fusion



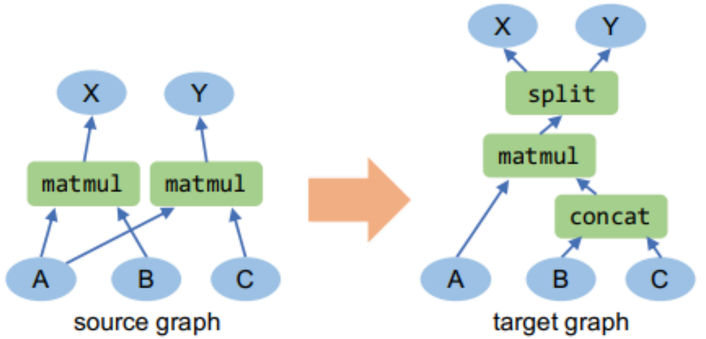
Fusion is not always profitable!
Typical type II Optimization

Operator Fusion

Graph Simplifications



(a) Associativity of matrix multiplication.



(b) Fusing two matrix multiplications using concatenation and split.

- expressed as computational graph rewrites

TASO [SOSP'19]

https://cs.stanford.edu/~padon/taso-sosp19.pdf

Operator Placement

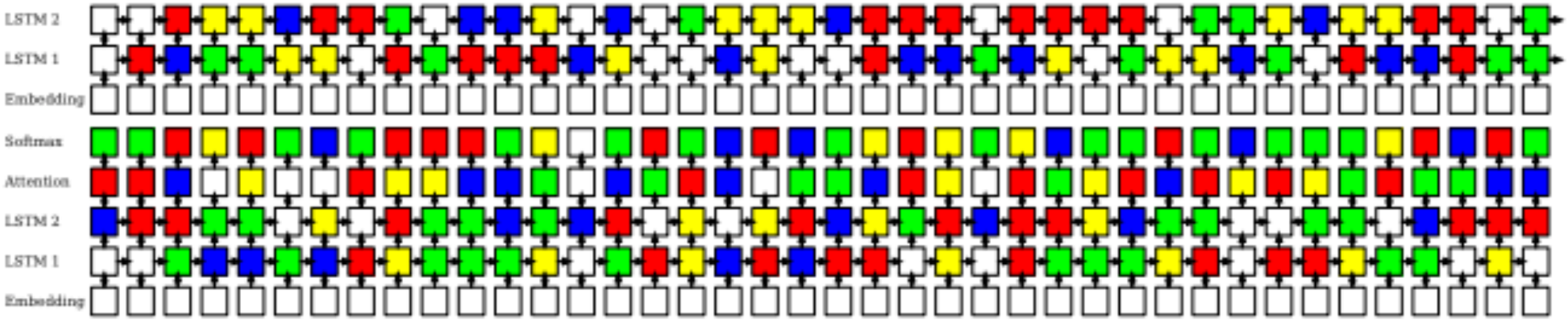


Figure 4. RL-based placement of Neural MT graph. Top: encoder, Bottom: decoder. Devices are denoted by colors, where the transparent color represents an operation on a CPU and each other unique color represents a different GPU. This placement achieves an improvement of 19.3% in running time compared to the fine-tuned expert-designed placement.

11/03 Reading

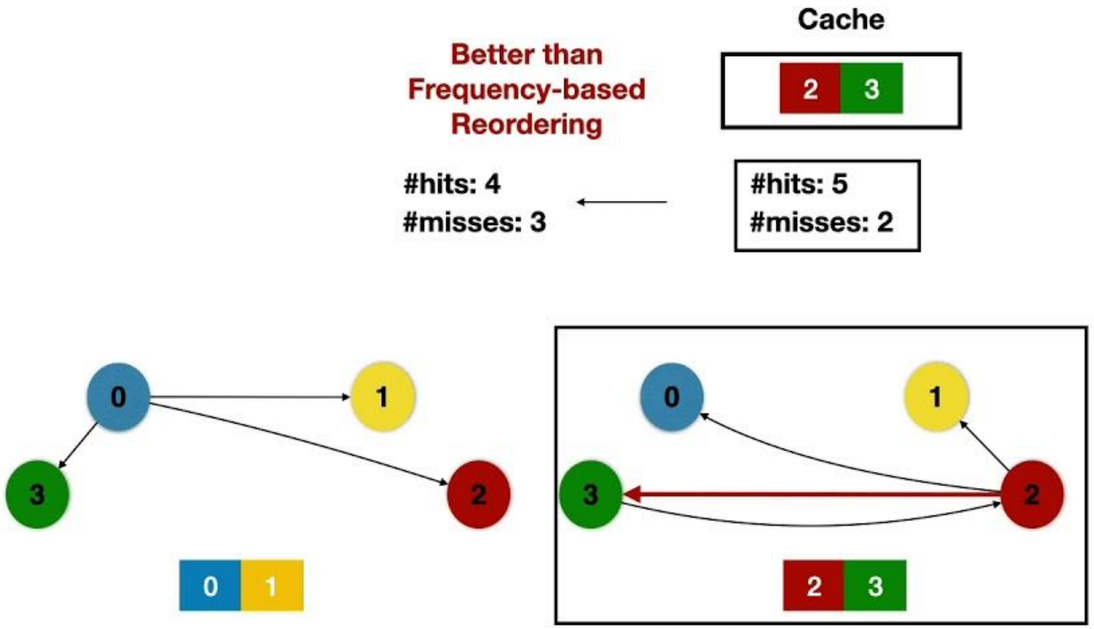
Mirhoseini et. al “Device Placement Optimization with Reinforcement Learning”

Graph Computing

- Models computations on graph structured data
- Vertex centric or Edge centric
- Plenty of domain specific optimizations
  - Push / Pull optimizations
  - Vertex Reordering
  - Graph Segmentation etc…

Optimizations on Graphs

Graph Processing

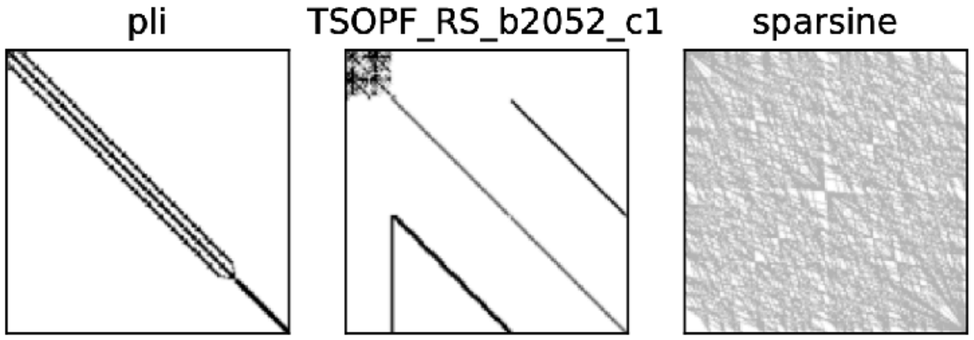


Auto-tuning for graphs

- We have a separate lecture on auto-tuning
- Challenging
  - Highly input sensitive (e.g. power-law graphs vs road graphs)
  - Dependent on the graph algorithm (e.g. Page rank vs BFS)
  - Dependent on hardware (e.g. GPUs vs CPUs)
- **Meng et. al “A pattern based algorithmic autotunes for graph processing on GPUs”**

Sparse Computations

- Few primitive kernels used heavily in ML as well as in traditional HPC.
  - Sparse Matrix Vector Multiplication (SpMV)
  - Sparse Matrix Dense Matrix Multiplication (SpMM)
  - Sampled Dense-Dense Matrix Multiplication (SDDMM)



Jaeyeon et. al. "WACO: Learning Workload-Aware Co-optimization of the Format and Schedule of a Sparse Tensor Program”

ML in Architecture

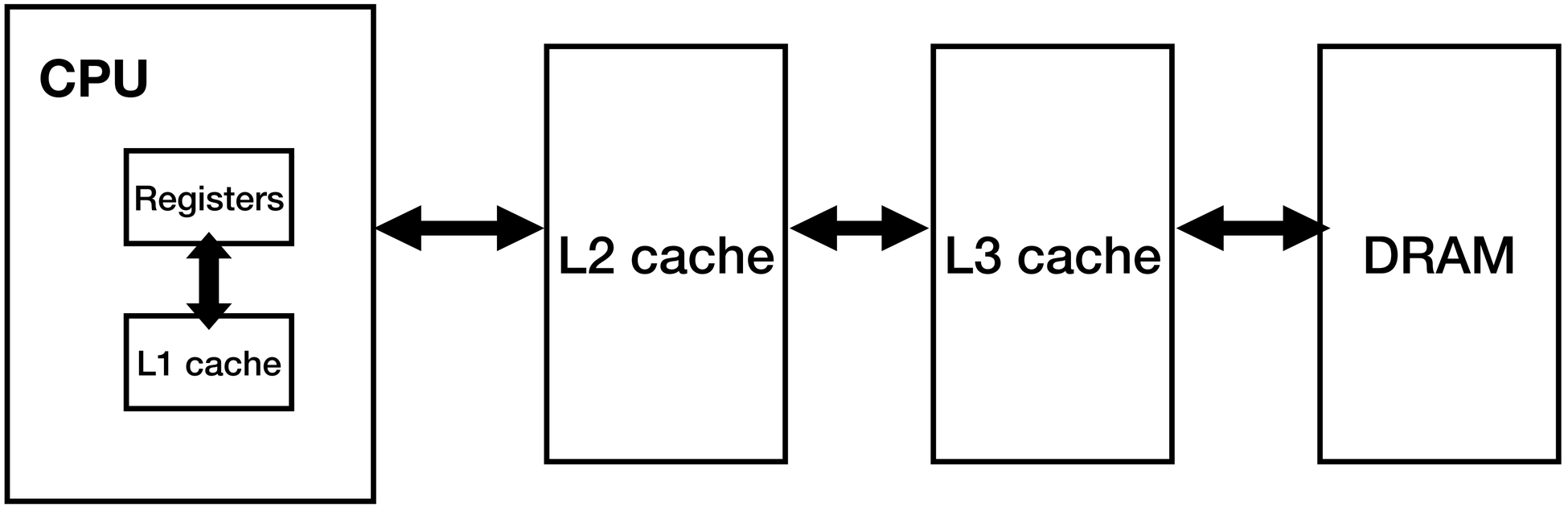
Memory Subsystem

- Usually CPU arithmetic capacity is far greater than the memory bandwidth or latency

- We can optimize memory performance by exploiting
  - Spatial Locality
  - Temporal Locality

- This gave rise to caches and cache hierarchies

Caches



Size: Registers < L1 cache < L2 cache < L3 cache < DRAM

Latency: Registers < L1 cache < L2 cache < L3 cache < DRAM

Caches

- Hold a portion of the data from memory for faster access
- The space is caches is limited, so determining what data it holds is important
- We want to maximize cache-hit rate
- Cache replacement policies are important in determining the cache-hit rate
- **11/30** - we are going to discuss learned replacement policies

ML in Architecture: good idea?

- Depends
  - Assume you are including a Neural Network (NN) in HW
    - Implementing NN in takes area
    - Adds execution latency / clock cycles get longer
    - May be a too heavy of a hammer
  - Designing new hardware: Great Choice!
    - We are reading papers on **11/14** and **11/16** about Design Space Exploration

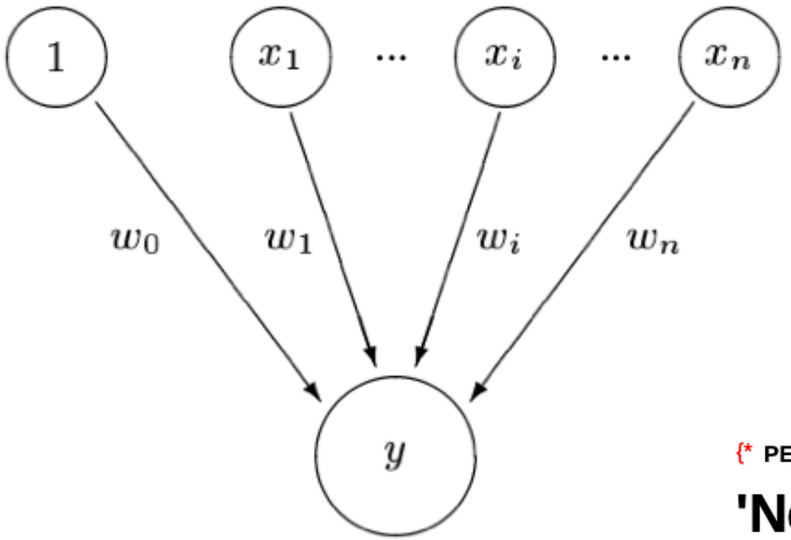
Branch Prediction

- Determines which instructions to fetch on a conditional branch
- If wrong instructions are fetched and executed entire processor pipeline should be flushed — Costly!
- Early work on using ML for branch prediction, **Jimenez and Lin “Dynamic Branch Prediction with Perceptrons”**

HPCA 2019 Test of Time award

Dynamic Branch Prediction with Perceptrons

- Why perceptrons? Can be efficiently implemented in hardware



$$y = w_0 + \sum_{i=1}^{n} x_i w_i .$$

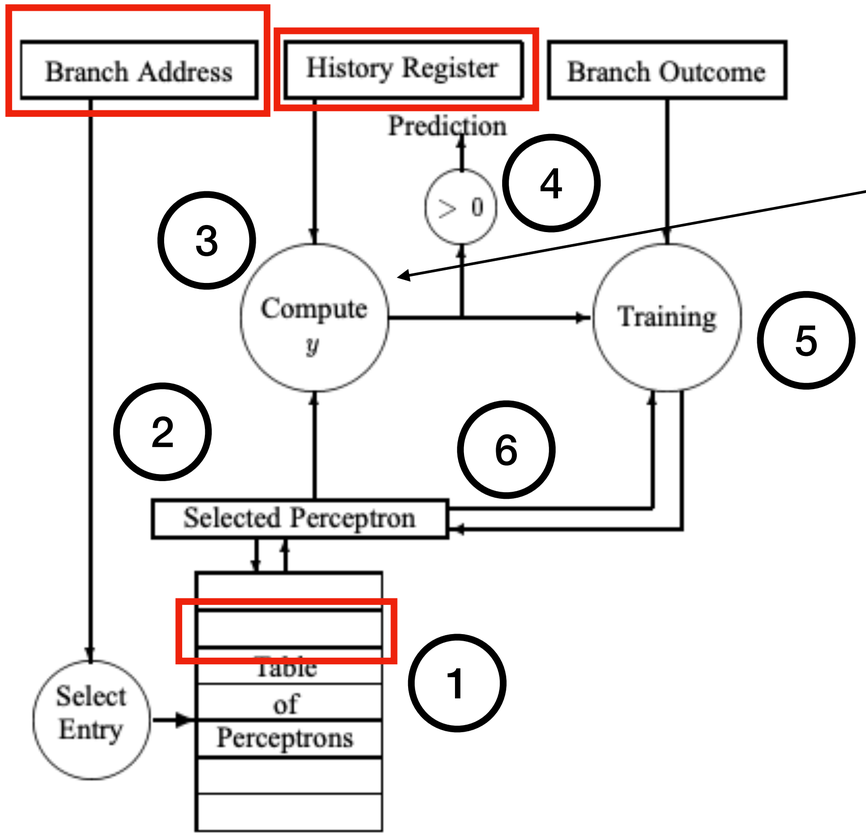
{* PERSONAL TECH *}

'Neural network' spotted deep inside Samsung's Galaxy S7 silicon brain

Secrets of Exynos M1 cores spilled

https://www.theregister.com/2016/08/22/samsung_m1_core/

Dynamic Branch Prediction with Perceptrons



$$y = w_0 + \sum_{i=1}^{n} x_i w_i.$$

```
if sign(y_out) ≠ t or |y_out| ≤ θ then
        for i := 0 to n do
                w_i := w_i + t x_i
        end for
end if
```

Dynamic Branch Prediction with Perceptrons

```
if sign(y_out) ≠ t or |y_out| ≤ θ then
        for i := 0 to n do
                w_i := w_i + t x_i
        end for
end if
```

If the predictor is right

No training needed!

If the predictor is wrong or within threshold

W_i ↑ When t and x_i same sign

W_i ↓ When t and x_i opposite signs

Any Questions?