FPGA Applications on Nautilus: Physics Example

Mohammad Sada and Elham E Khoda

Sixth National Research Platform (6NRP) Workshop

January 28th, 2025

SAN DIEGO
SUPERCOMPUTER
CENTER

UC San Diego

Setup Instruction

We will use Jupyterhub for this session!

Follow instruction on GitHub:
https://github.com/nrp-nautilus/6nrp-hls4ml/tree/main

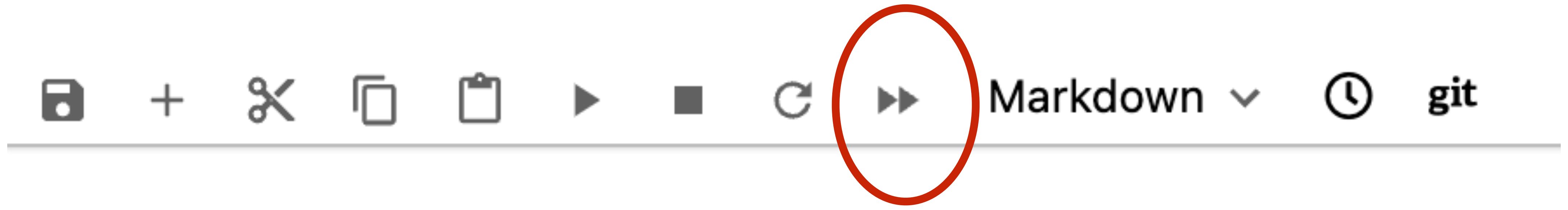
JupyterHub link:

- https://6nrp.nrp-nautilus.io/
- Log in with your university credentials via CILogon

Open and start running through “02_hls_conversion.ipynb” !

Run all the cells

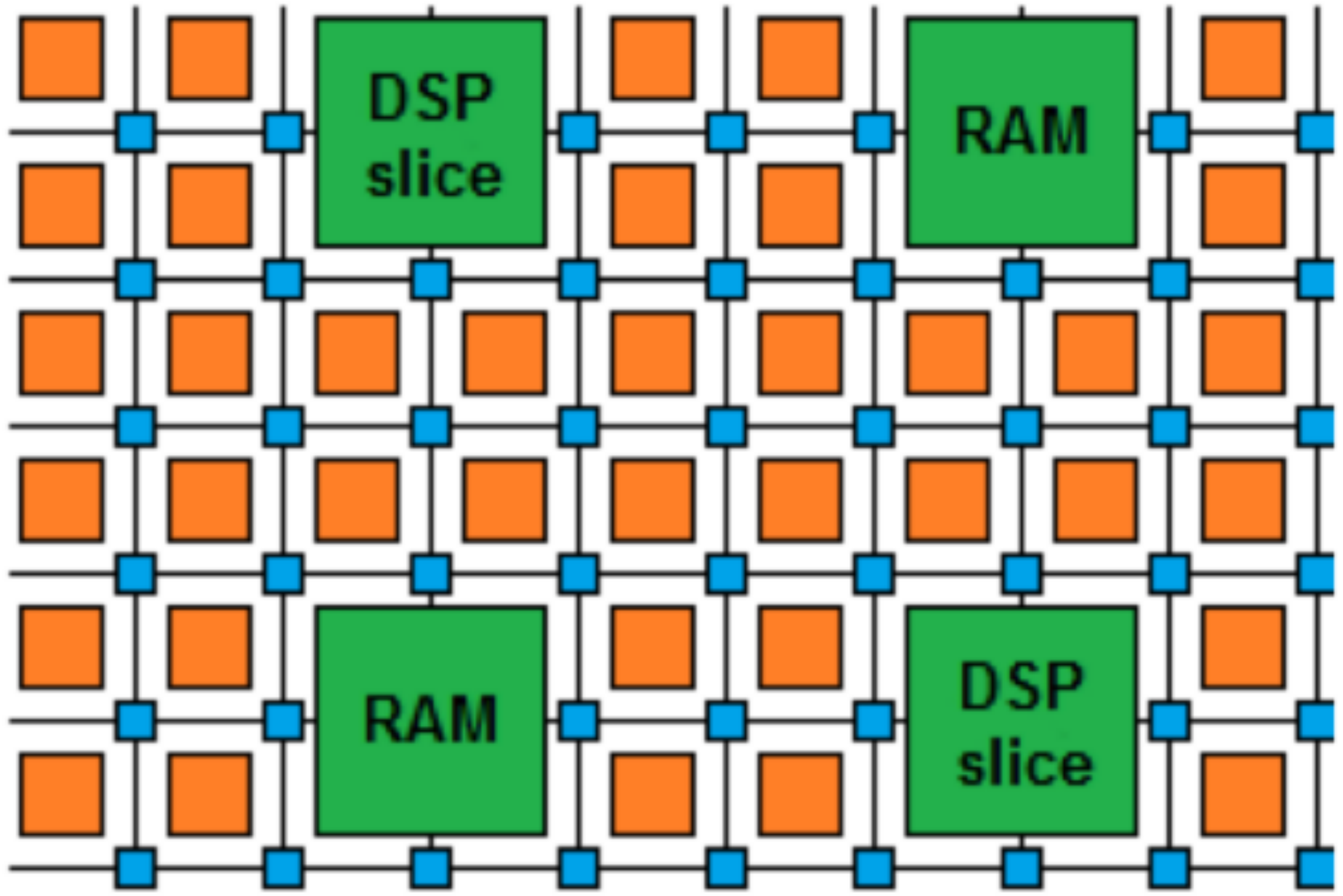
Hit this



What is an FPGA?

Field **P**rogrammable **G**ate **A**rrays (FPGAs) are reprogrammable integrated circuits

- Contain many different building blocks (‘resources’) which are connected together as you desire
- Originally popular for prototyping ASICs, but now also for high performance computing



Building blocks:

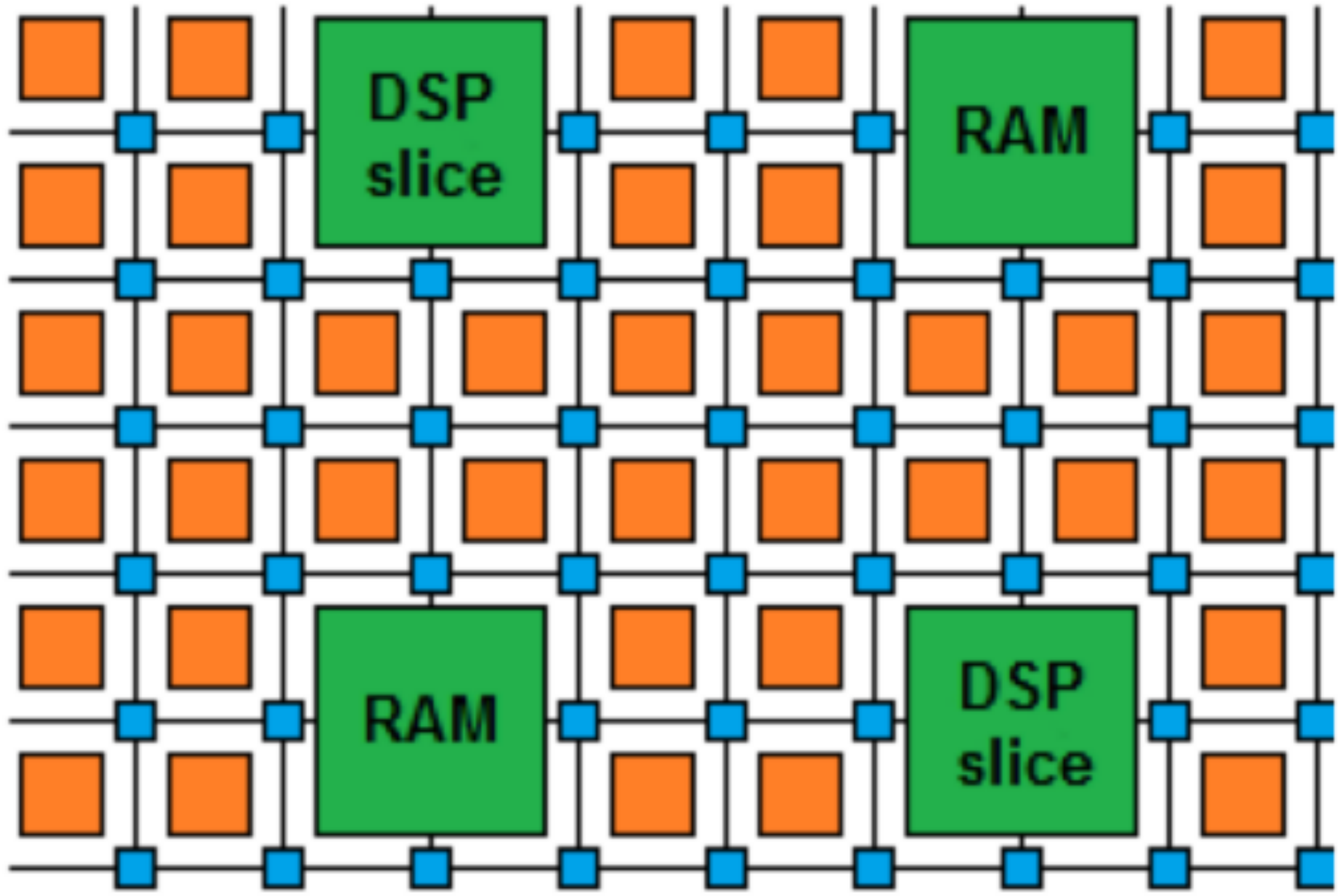
- **Multiplier units (DSPs)** [arithmetic]
- **Look Up Tables (LUTs)** [logic]
- **Flip-flops (FFs)** [registers]
- **Block RAMs (BRAMs)** [memory]

What is an FPGA?

- Run at high frequency - *O(100 MHz)*
  - Can compute outputs in O(ns)

- Low-level Hardware Description Language for programming
  Verilog/VHDL

- Possible to translate C/C++ → Verilog/VHDL using High **Level Synthesis (HLS)** tools

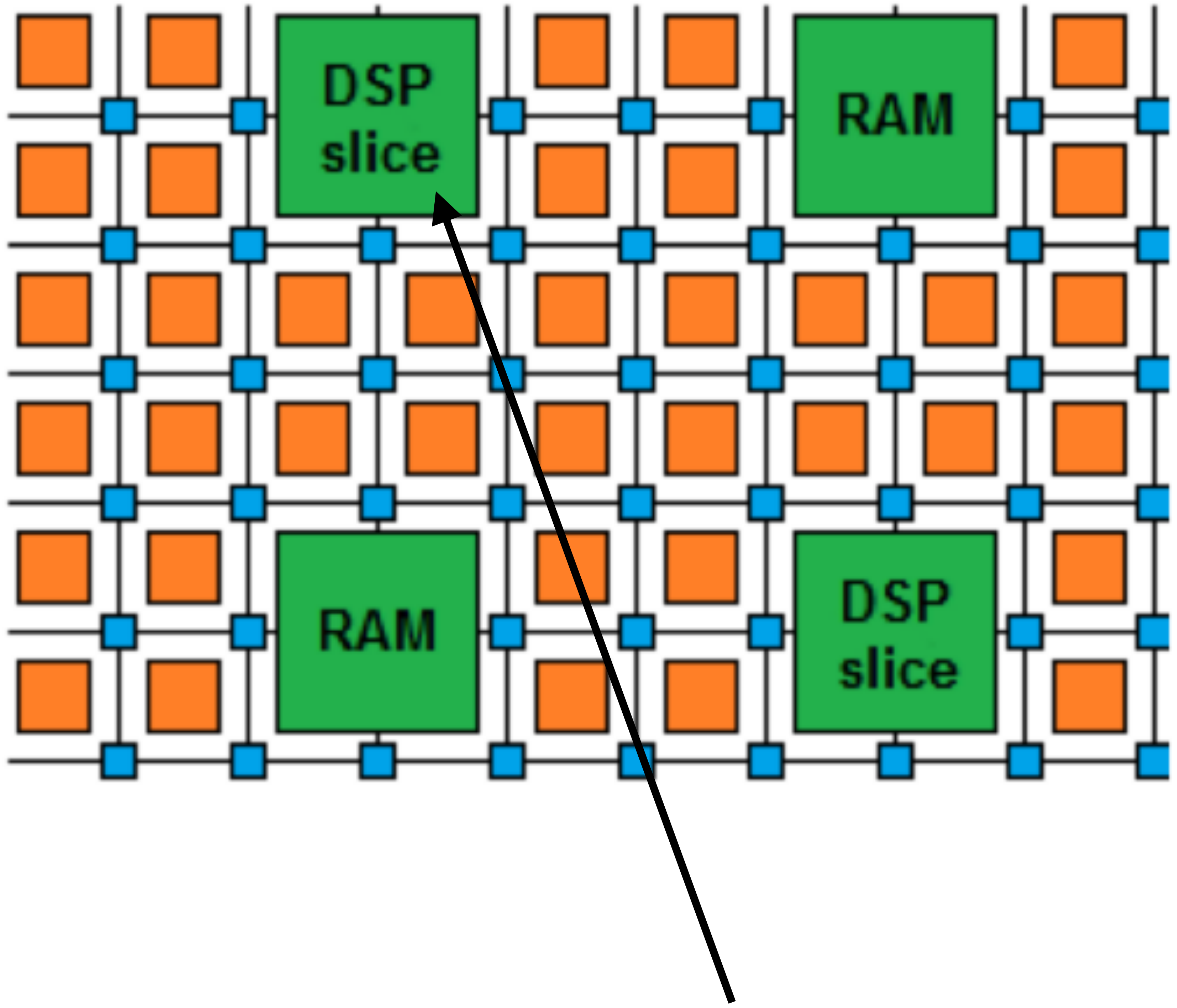


Building blocks:

– **Multiplier units (DSPs)** [arithmetic]
– **Look Up Tables (LUTs)** [logic]
– **Flip-flops (FFs)** [registers]
– **Block RAMs (BRAMs)** [memory]

What is an FPGA?

- DSPs (Digital Signal Processor) are specialized units for multiplication and arithmetic
- DSPs are often the most scarce for NNs
- Faster and more efficient than using LUTs for these types of operations



DSP
(multiplication)

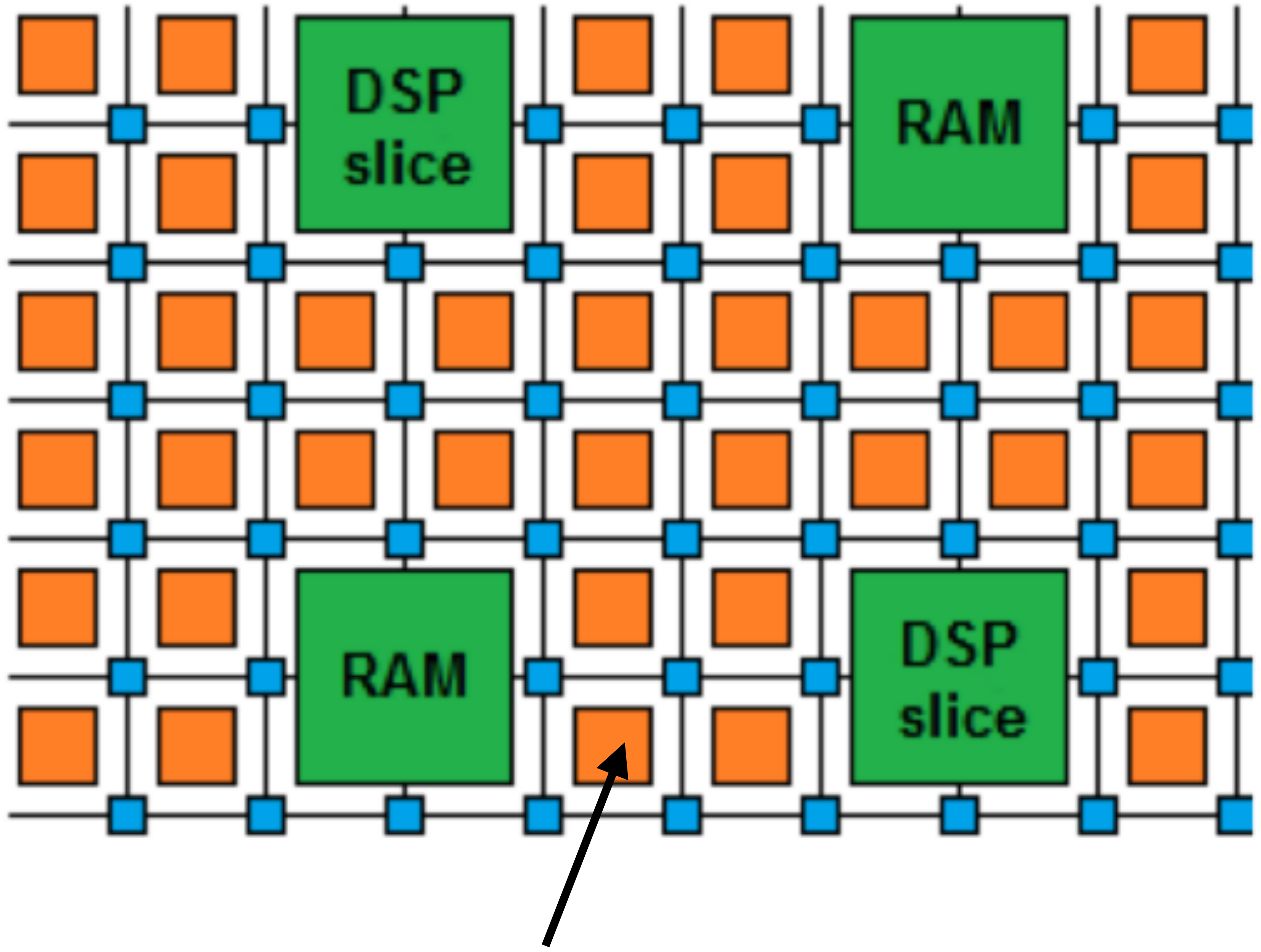
Building blocks:

– **Multiplier units (DSPs)** [arithmetic]
– **Look Up Tables (LUTs)** [logic]
– **Flip-flops (FFs)** [registers]
– **Block RAMs (BRAMs)** [memory]

What is an FPGA?

- Logic cells / Look Up Tables perform arbitrary functional operations on small bit-width inputs (2-6)
  - boolean, arithmetic
  - small memories

- Flip-Flops register data in time with the clock pulse

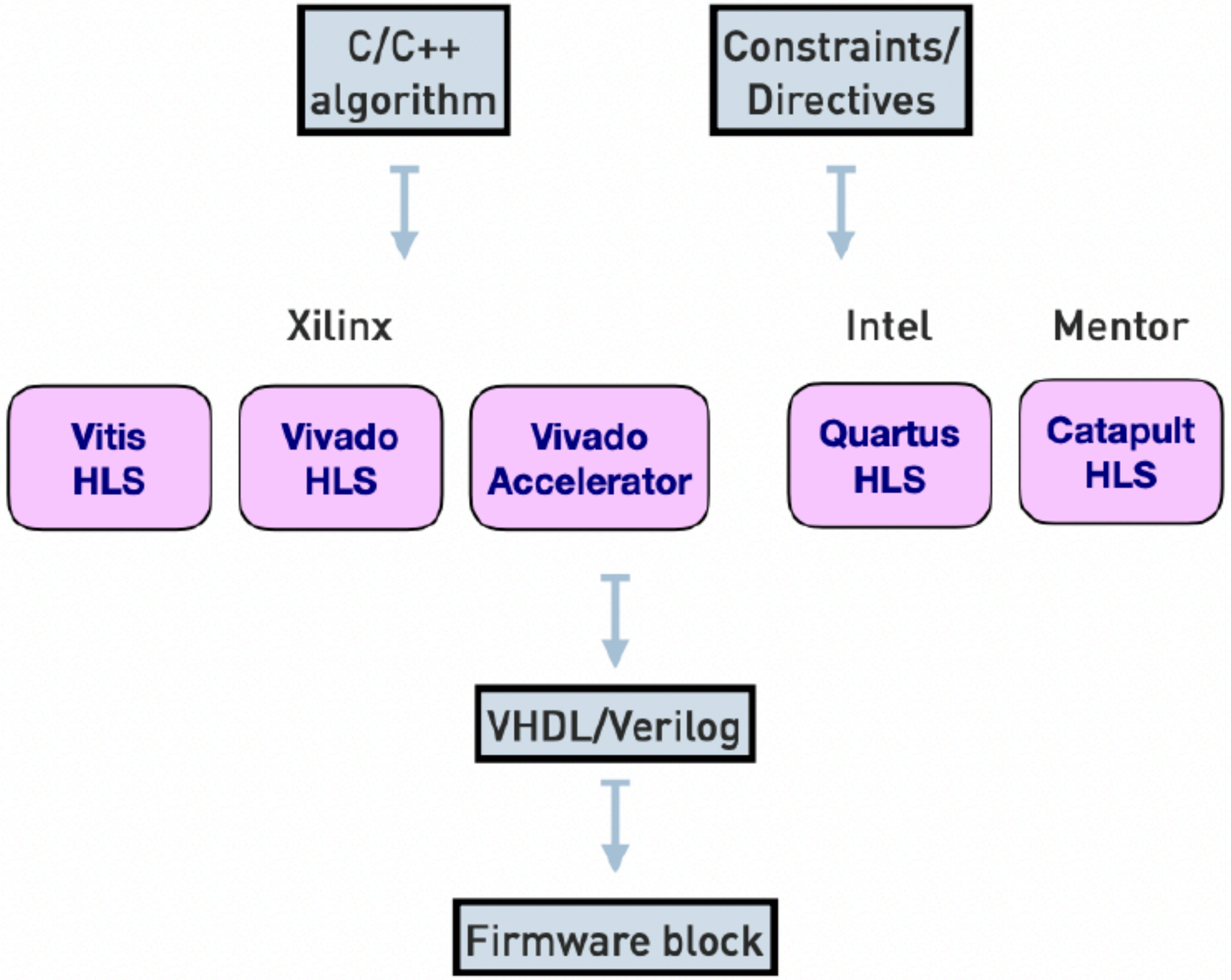


Logic cell

Building blocks:

– Multiplier units (DSPs) [arithmetic]
– Look Up Tables (LUTs) [logic]
– Flip-flops (FFs) [registers]
– Block RAMs (BRAMs) [memory]

FPGA Programming



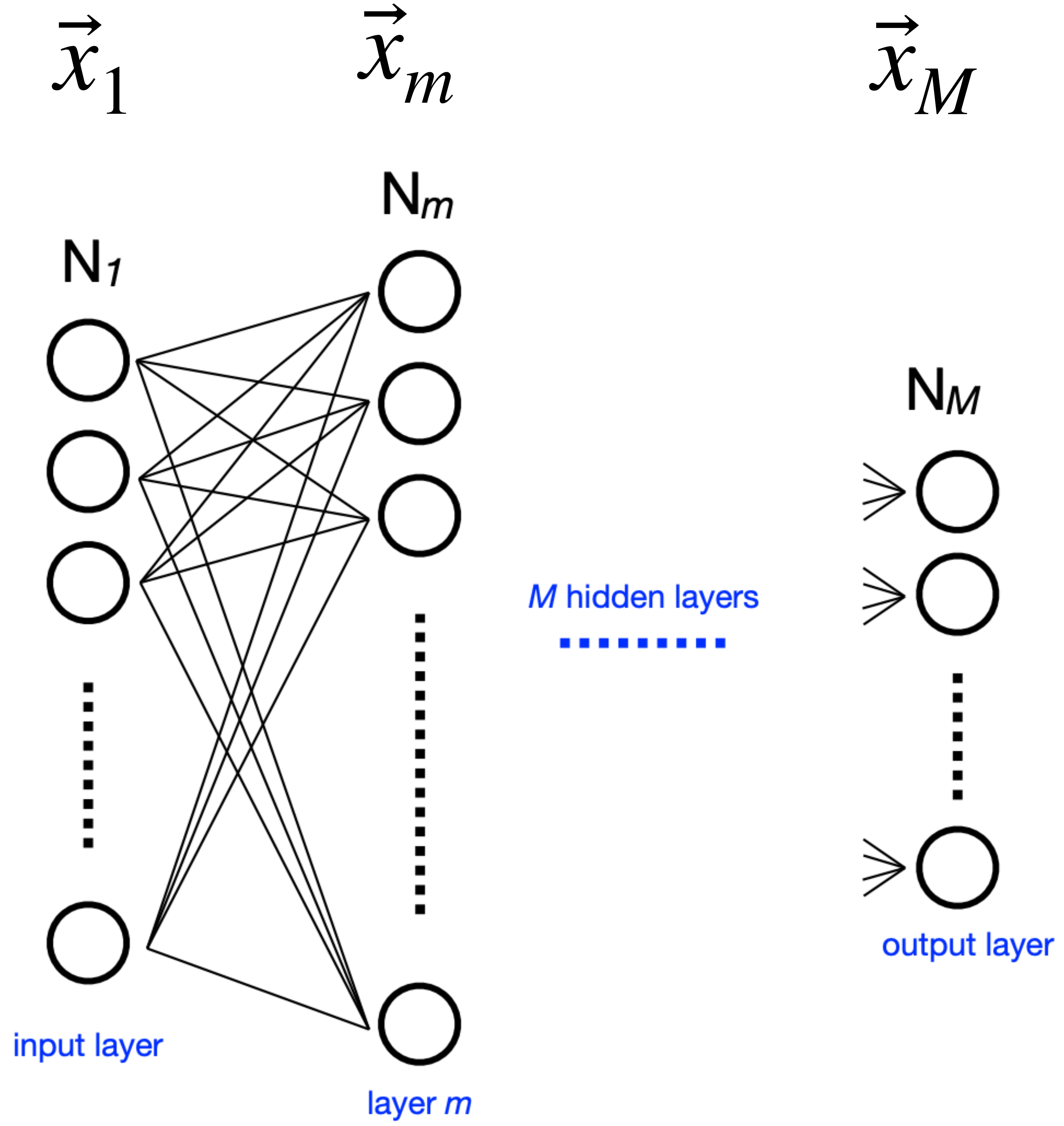
1. Write C-style code of function

2. Pass to a high-level synthesis (HLS) tool

3. HLS translates to hardware-description language (HDL)

4. Build firmware

Neural Network Inference on FPGA



multiplication

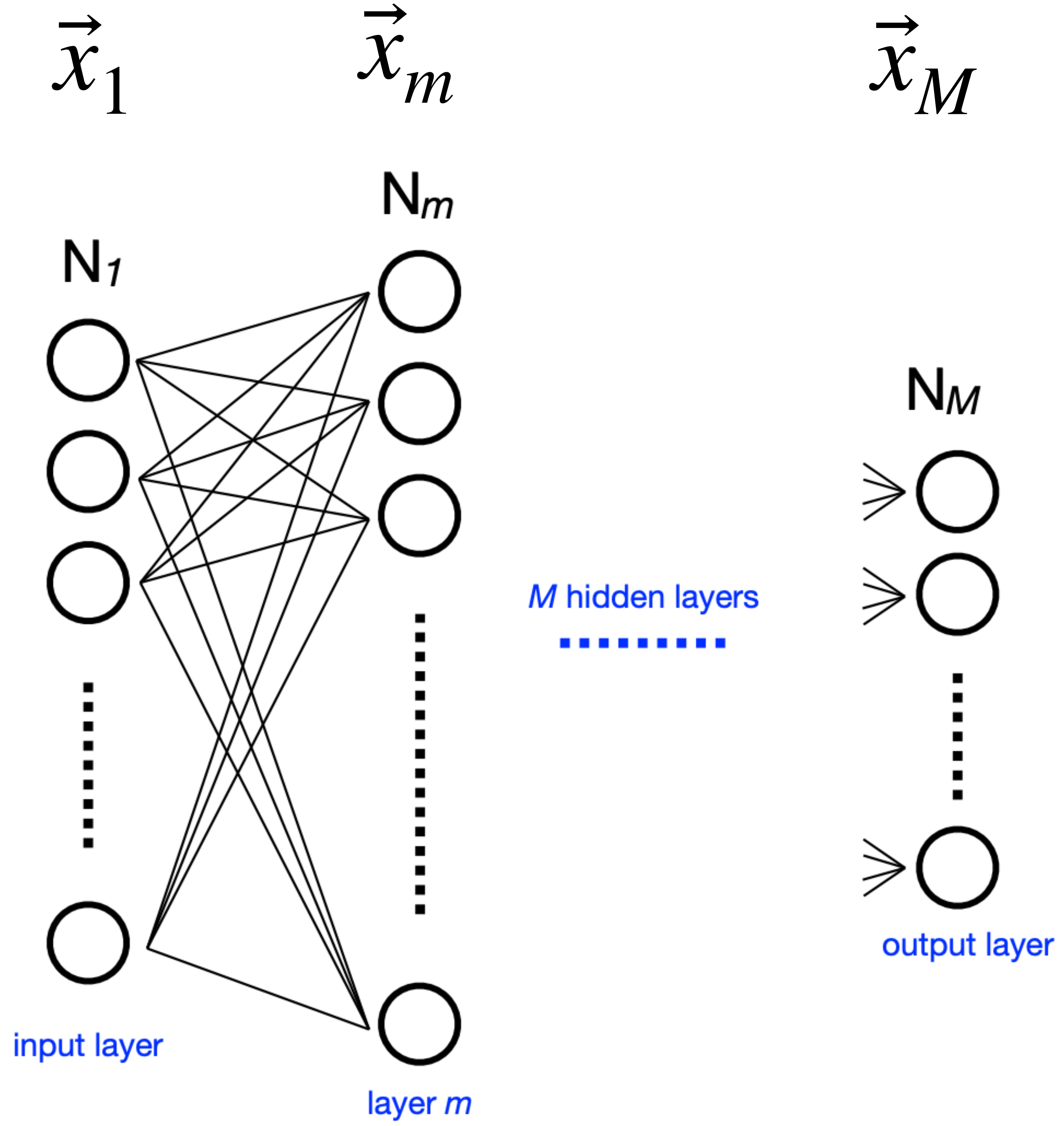
$$\vec{x}_m = g_m\left(W_{m,m-1}\vec{x}_{m-1} + \vec{b}_m\right)$$

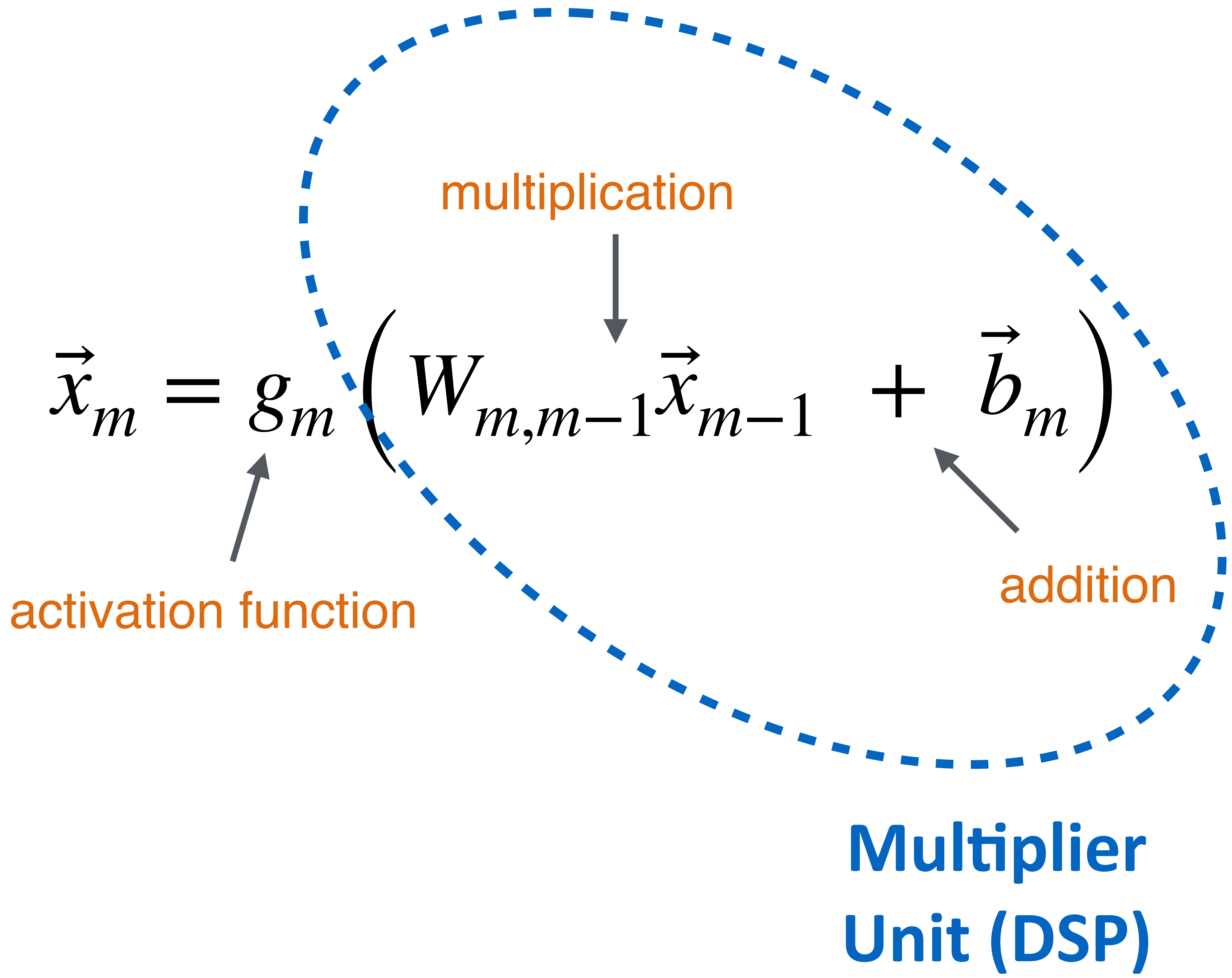
activation function

addition

Credit: Dylan Rankin

Neural Network Inference on FPGA





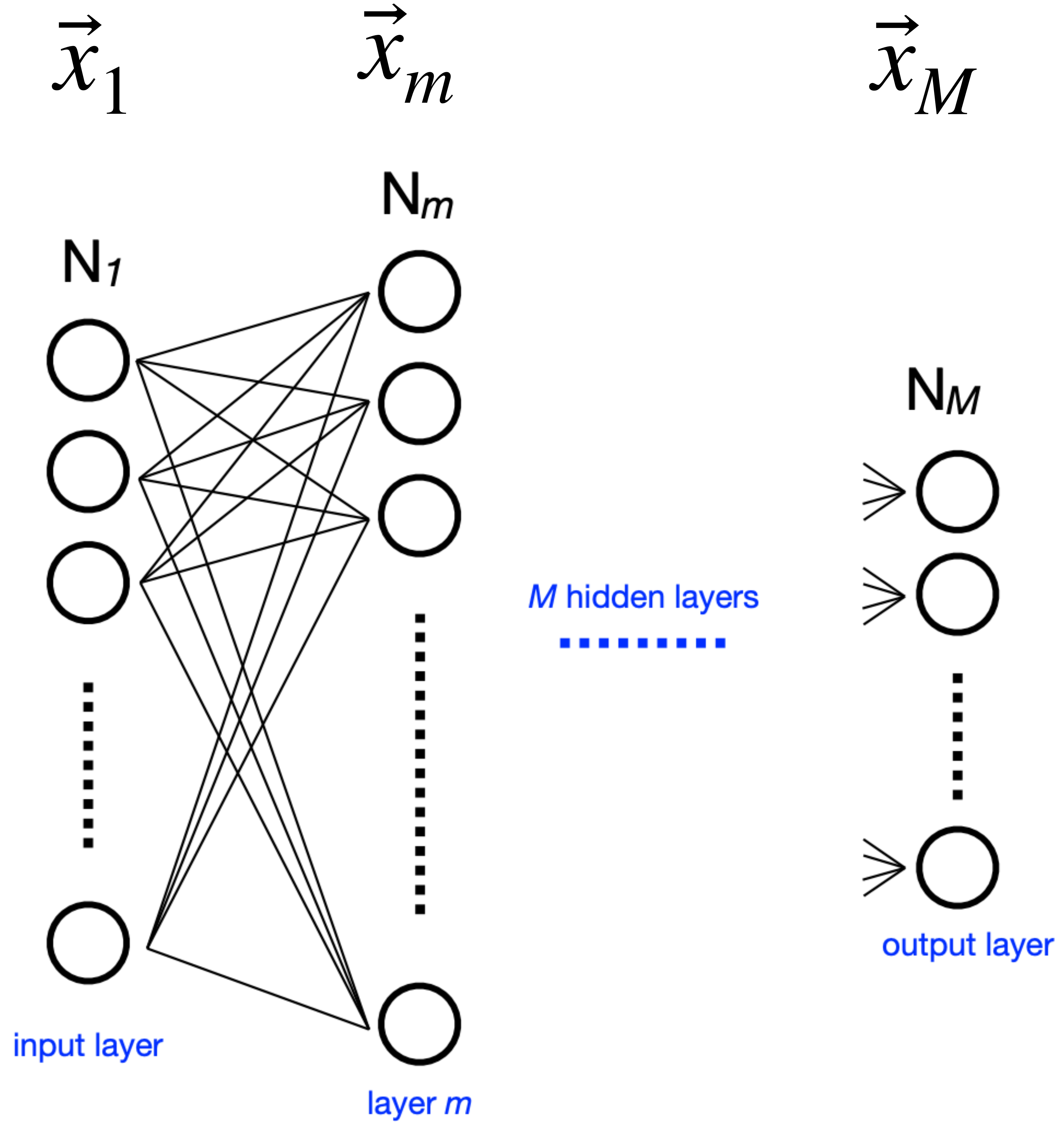
$$\vec{x}_m = g_m\left(W_{m,m-1}\vec{x}_{m-1} + \vec{b}_m\right)$$

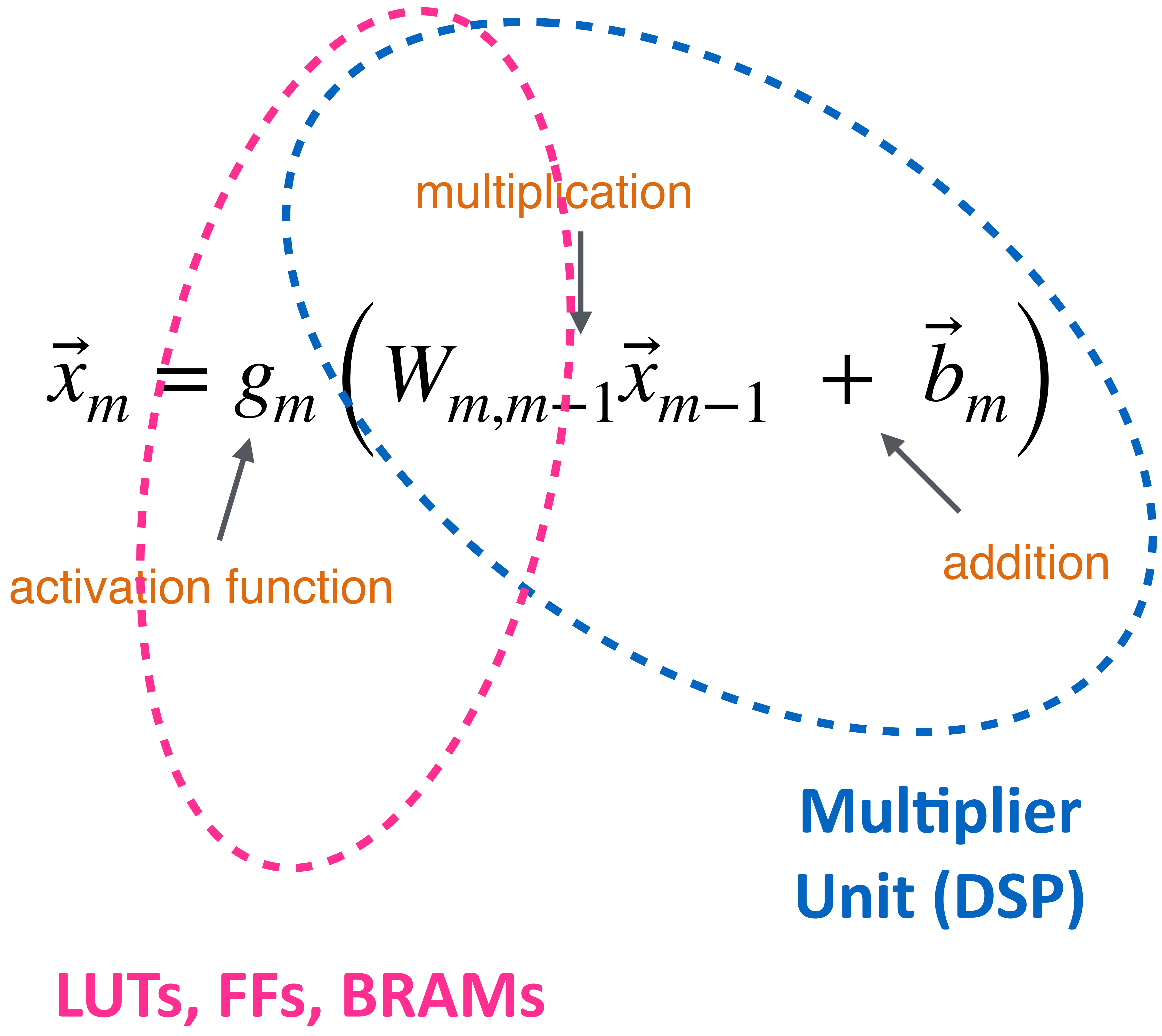
Multiplier Unit (DSP)

up to ~6k parallel operation (VU9P)

Credit: Dylan Rankin

Neural Network Inference on FPGA





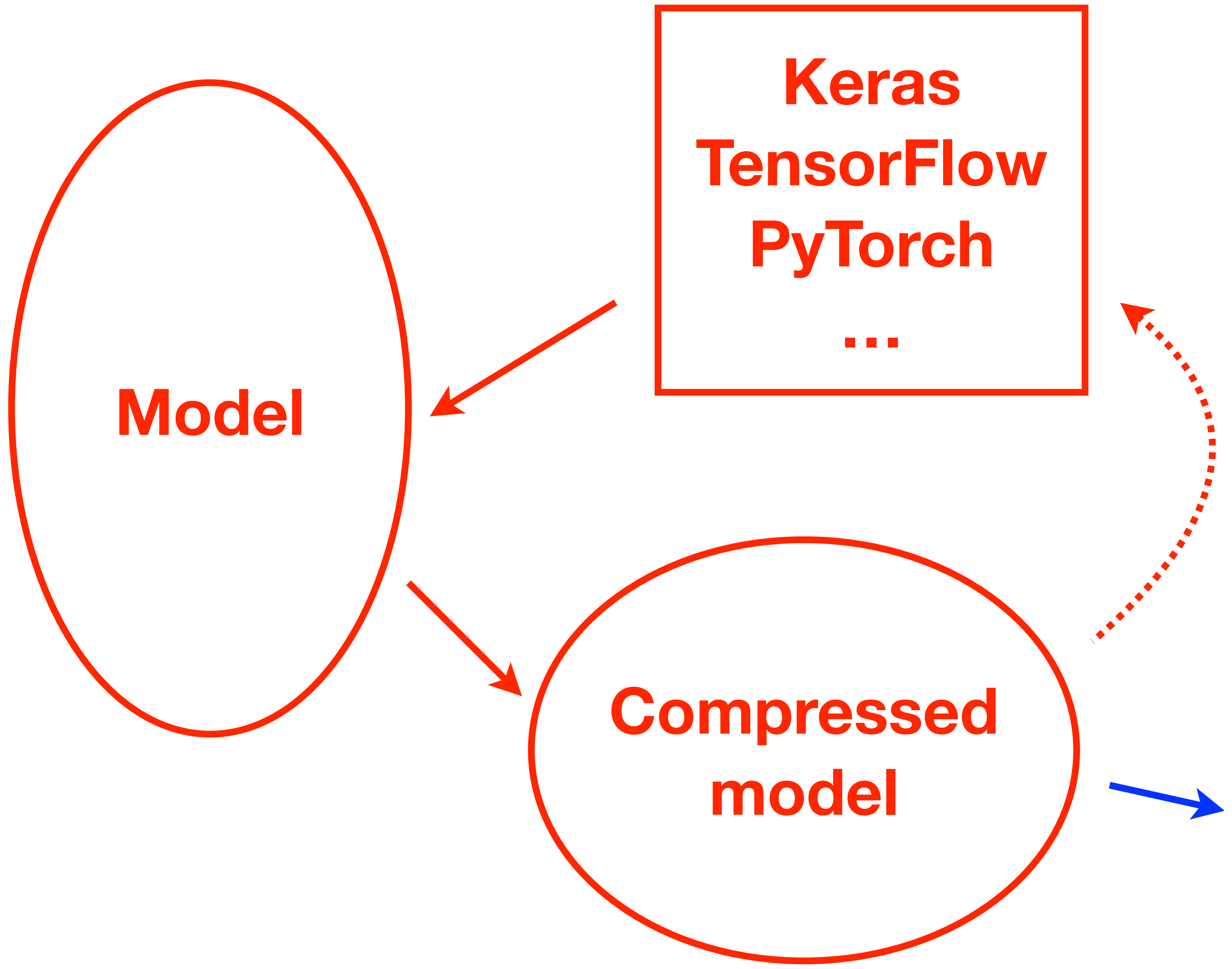
$$\vec{x}_m = g_m\left(W_{m,m-1}\vec{x}_{m-1} + \vec{b}_m\right)$$

Credit: Dylan Rankin

Design Exploration with hls4ml

JINST 13, P07027 (2018)

- hls4ml for scientists or ML experts to translate ML algorithms into RTL firmware

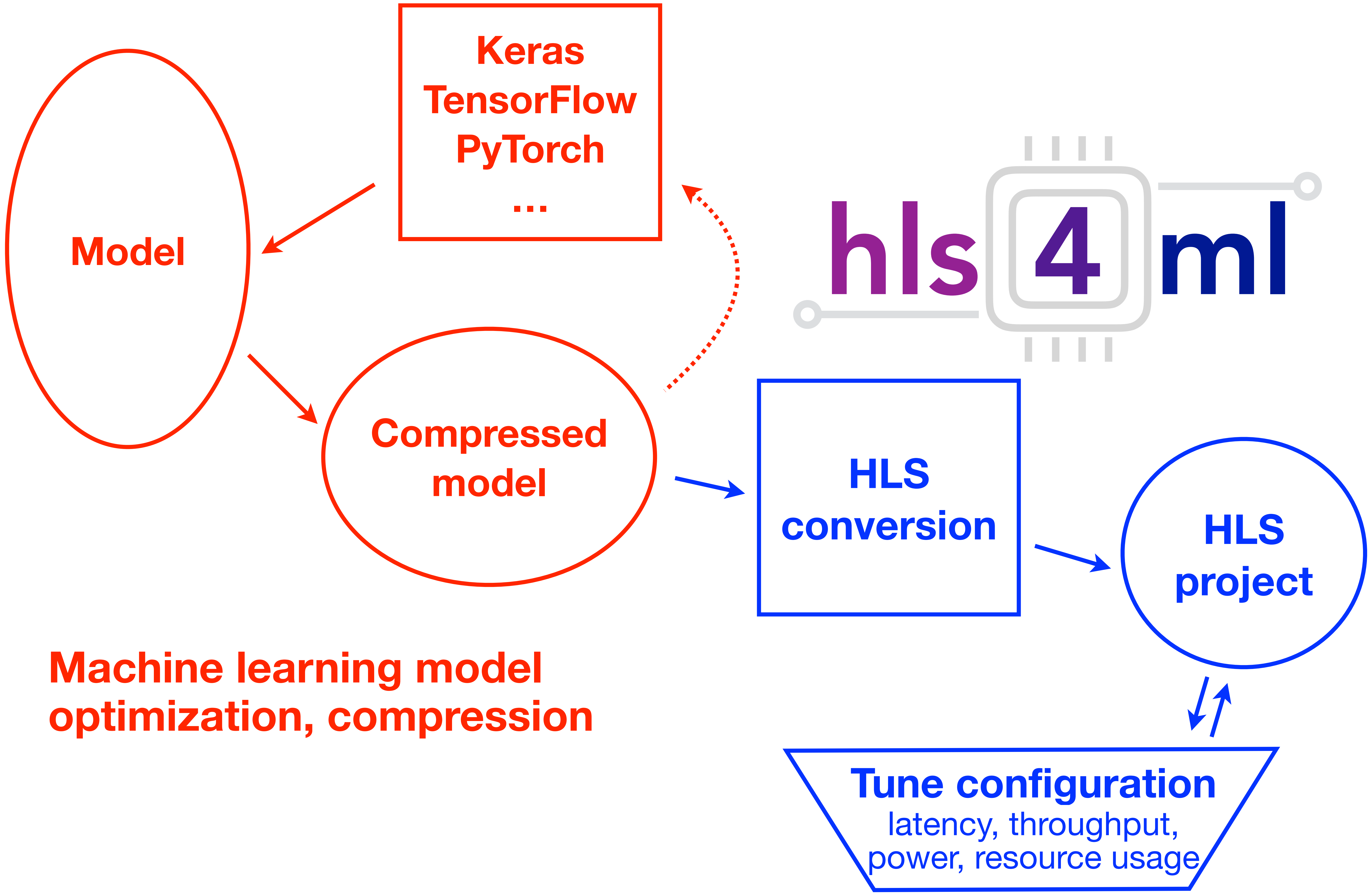


Machine learning model optimization, compression

Design Exploration with hls4ml

JINST 13, P07027 (2018)

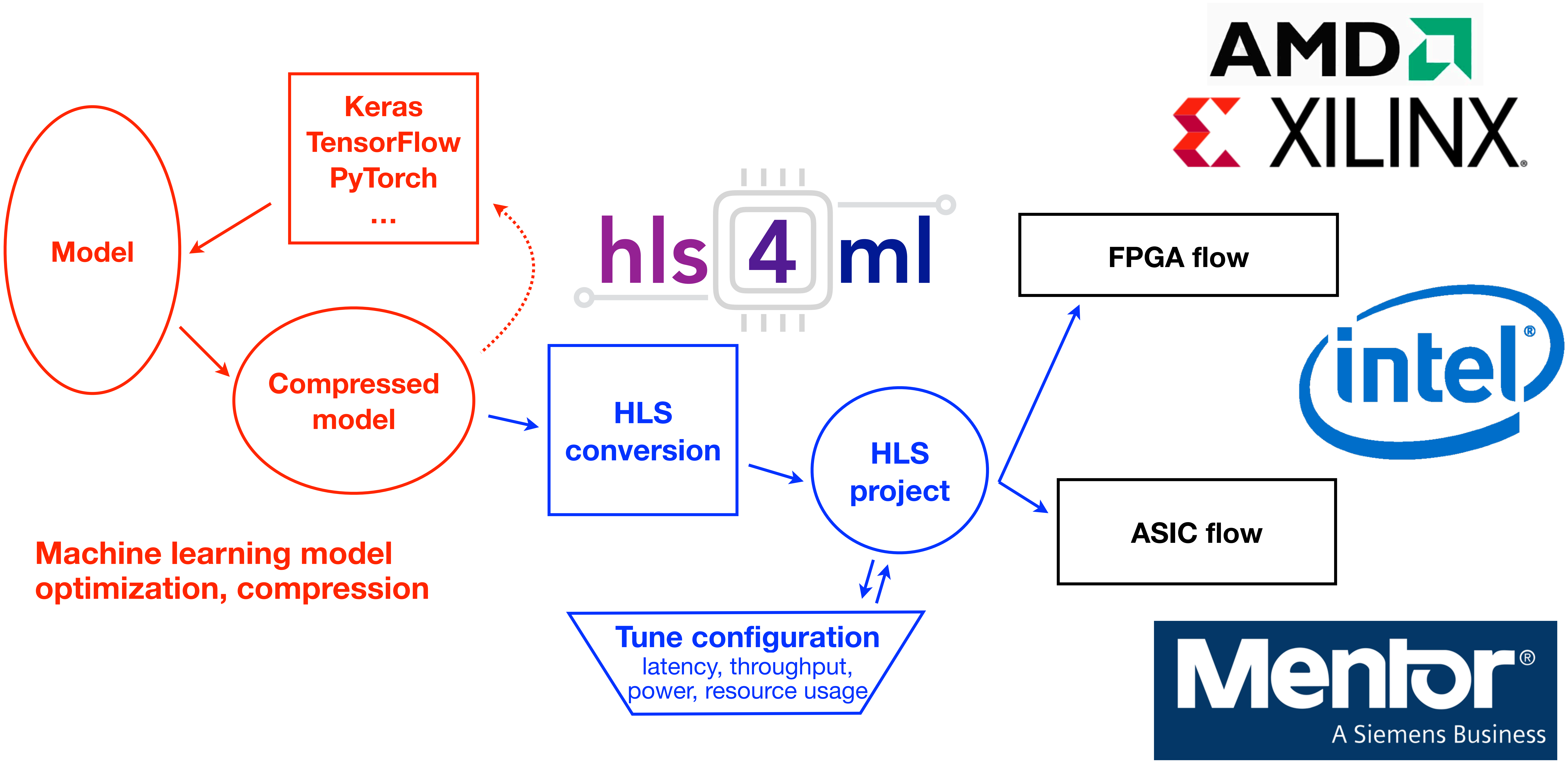
- hls4ml for scientists or ML experts to translate ML algorithms into RTL firmware



Design Exploration with hls4ml

JINST 13, P07027 (2018)

- hls4ml for scientists or ML experts to translate ML algorithms into RTL firmware



High-Level Syntenthesis for Machine Learning

https://fastmachinelearning.org/hls4ml/
arXiv:2103.05579

A software interface for implementing Neural Networks on an FPAG

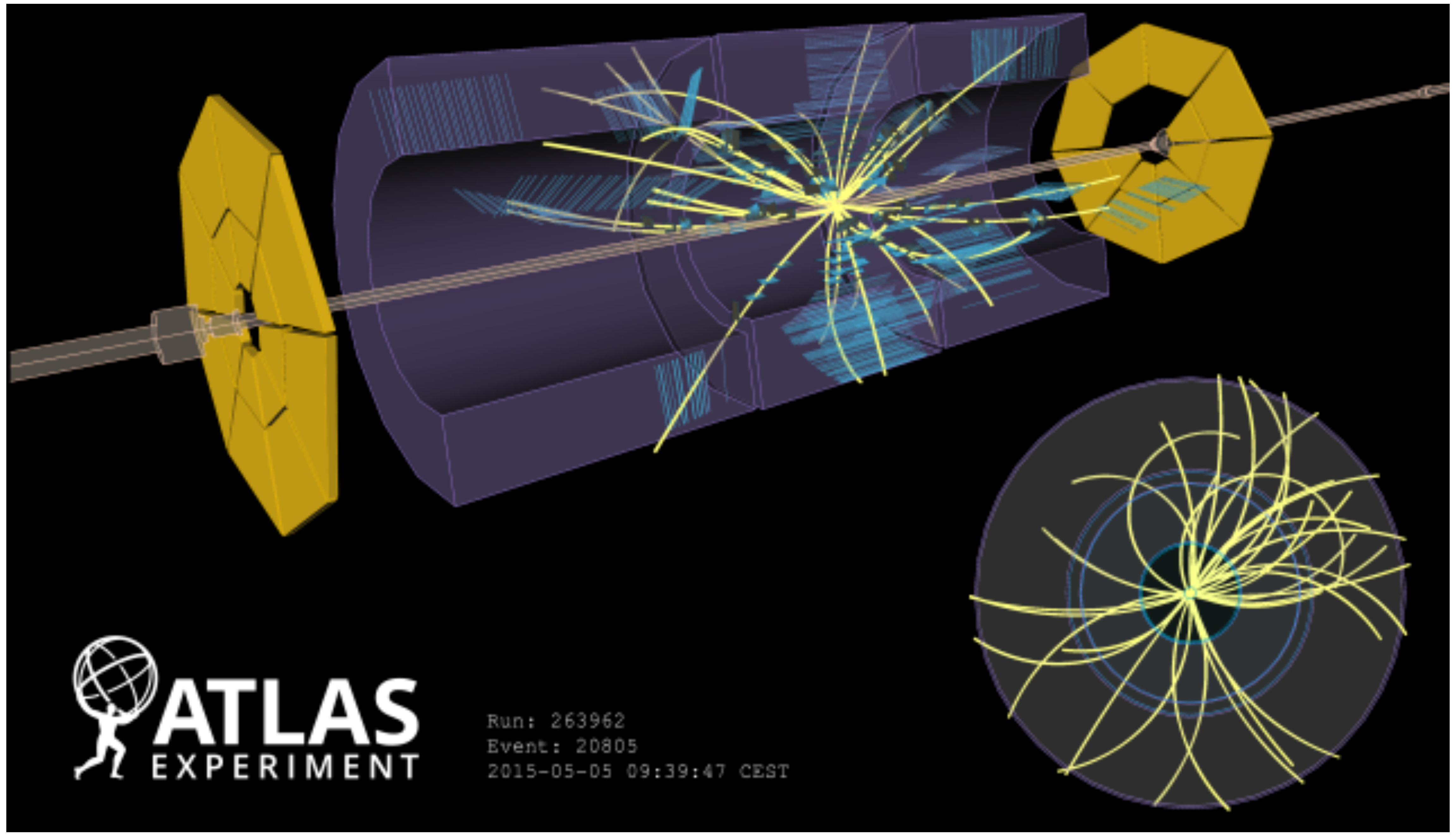
- Supports many common layer like DNN, CNN, RNN, GNN, Transformers, etc
- Support for different backends: Vivado/Vitis, oneAPI, Catapult, Quartus, etc

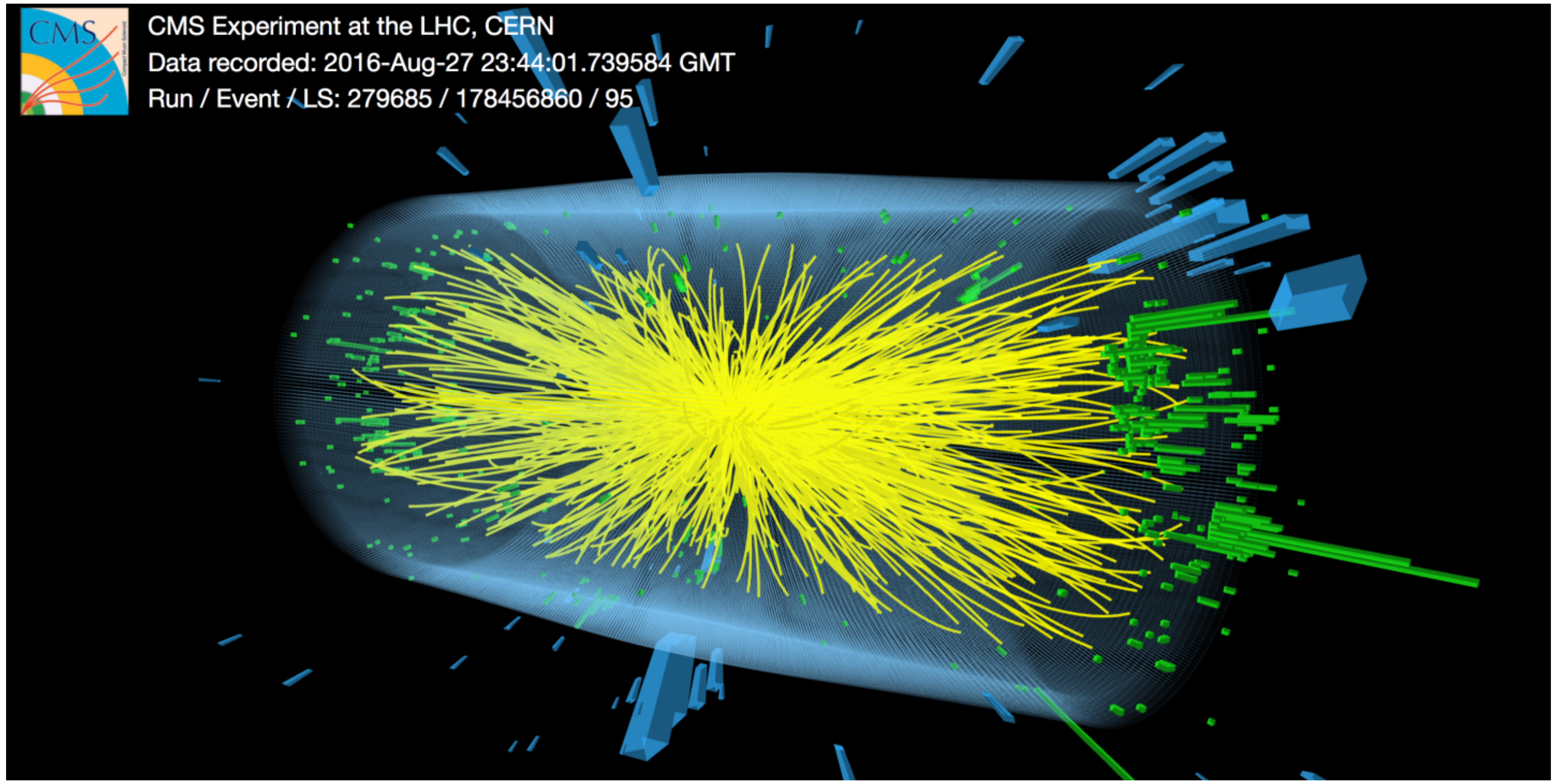
Official hls4ml tutorials:
https://fastmachinelearning.org/hls4ml-tutorial/README.html

Example Case Study: Physics Application

Particle physics Jet classification

Physics Case: Particle collisions

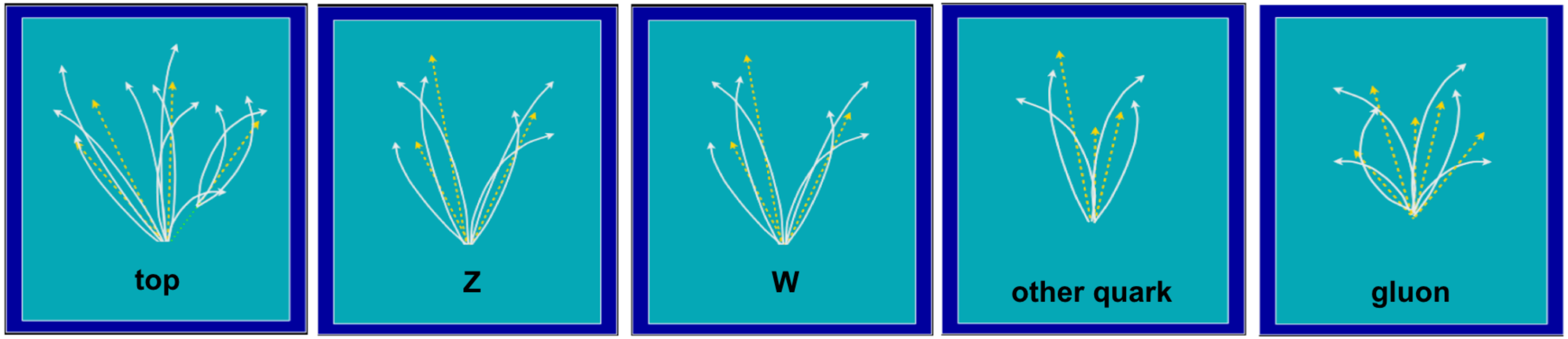



CMS
CMS Experiment at the LHC, CERN
Data recorded: 2016-Aug-27 23:44:01.739584 GMT
Run / Event / LS: 279685 / 178456860 / 95

Physics Case: Jet tagging

Study a multi-classification task to be implemented on FPGA:
discrimination between highly energetic (boosted) ***q, g, W, Z, t*** initiated *jets*

Jet = collimated ‘spray’ of particles



t→bW→bqq 3-prong jet

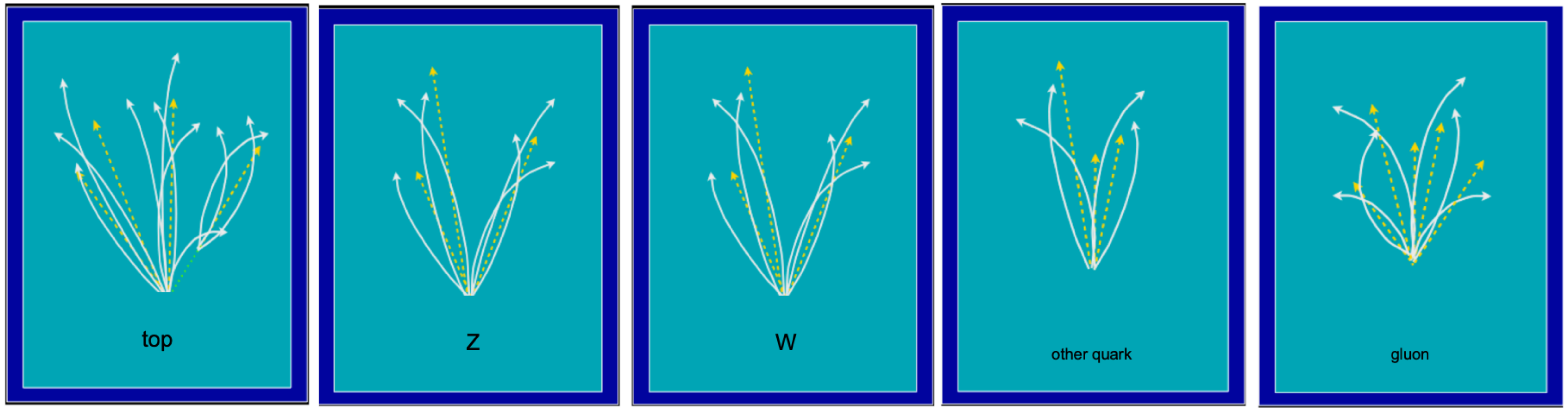
Z→qq 2-prong jet

W→qq 2-prong jet

Reconstructed as one massive jet with substructure

q/g background no substructure and/or mass ~ 0

Physics Case: Jet tagging



t→bW→bqq

Z→qq

W→qq

q/g background

Observables

m_{mMDT}

$N_2^{\beta=1,2}$

$M_2^{\beta=1,2}$

$C_1^{\beta=0,1,2}$

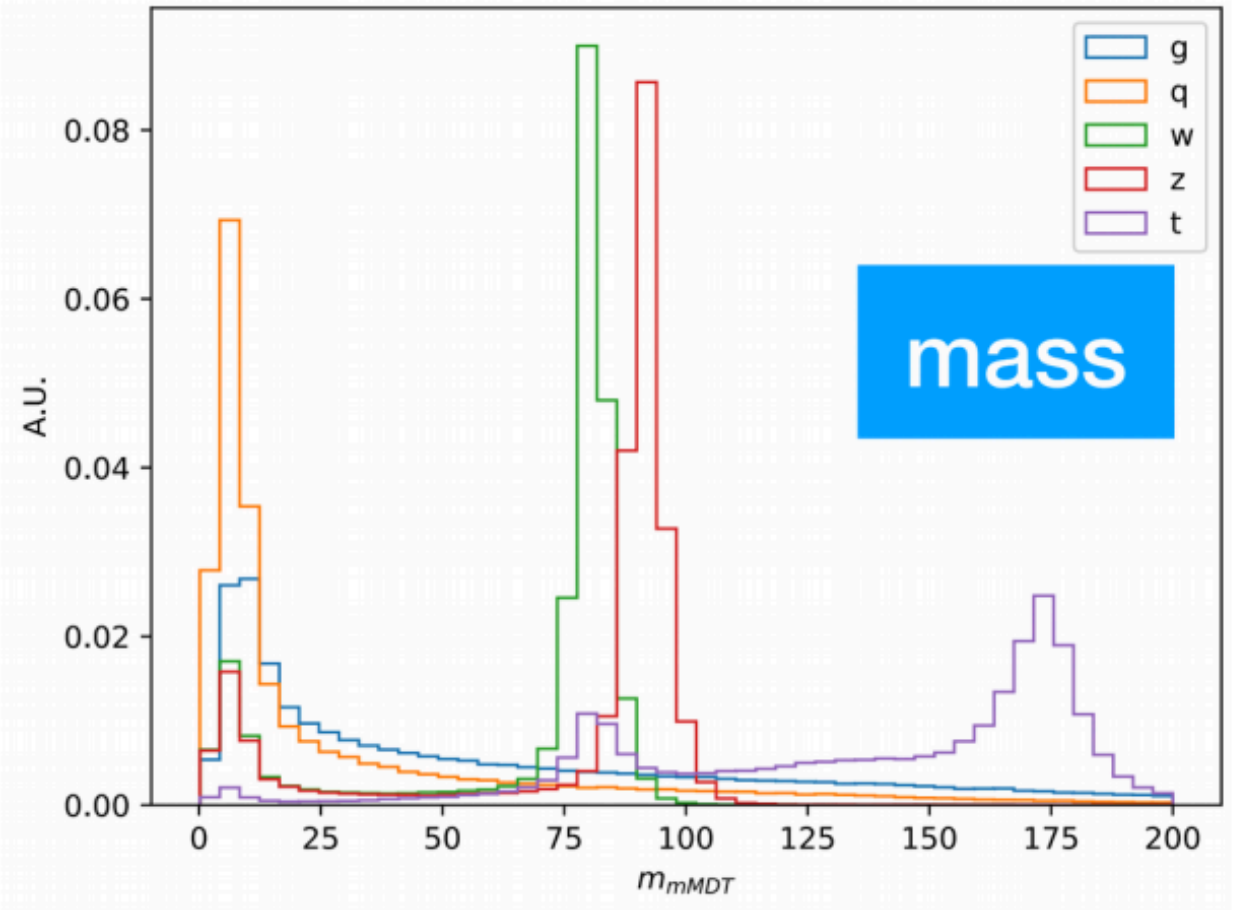
$C_2^{\beta=1,2}$

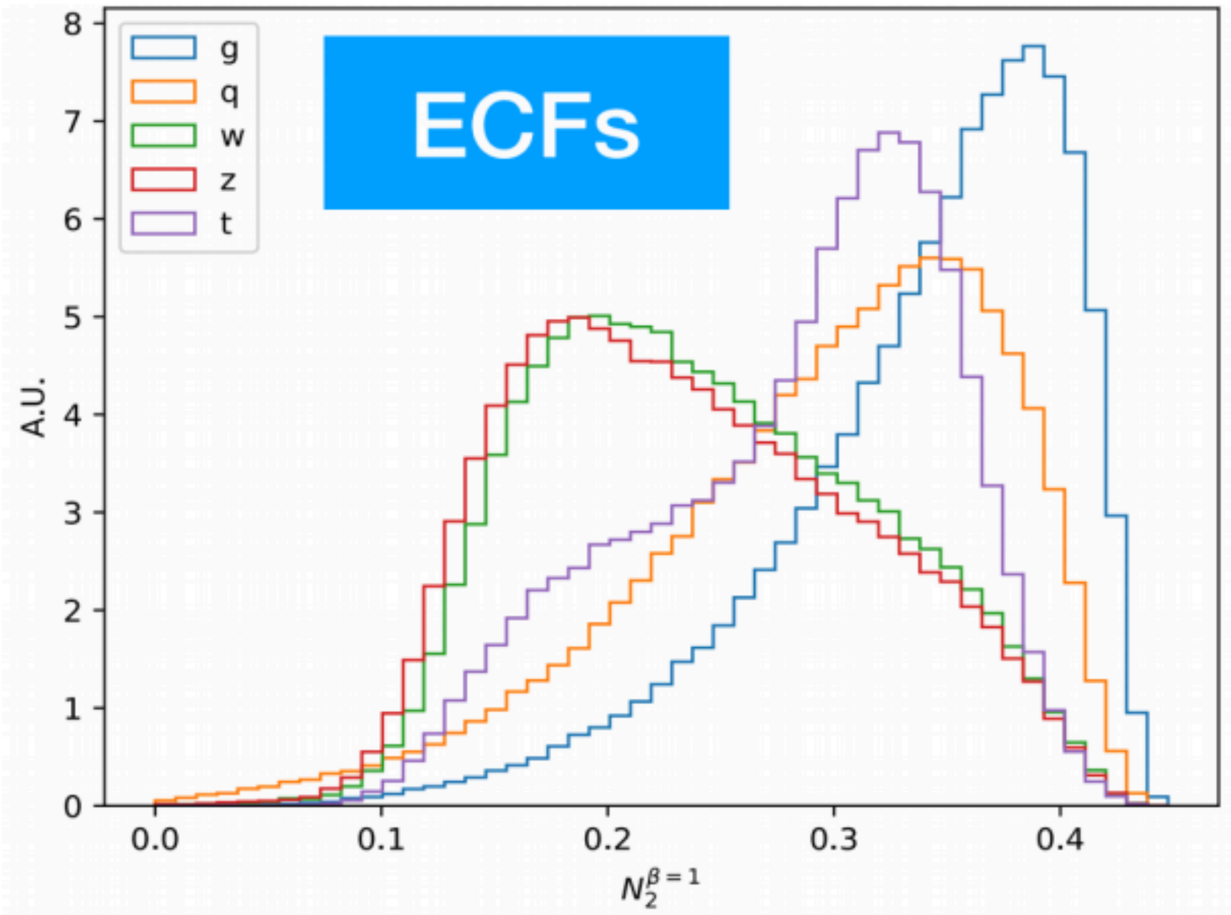
$D_2^{\beta=1,2}$

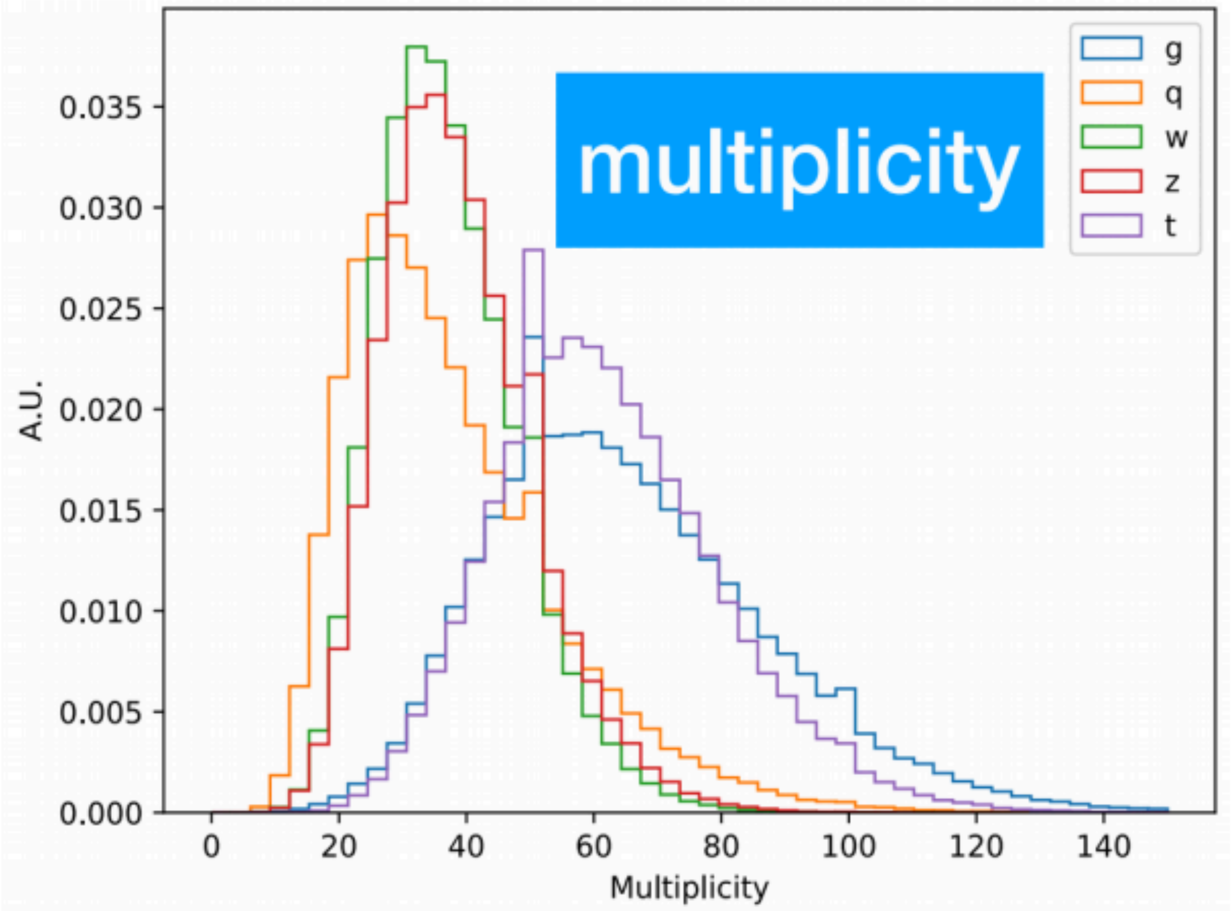
$D_2^{(\alpha,\beta)=(1,1),(1,2)}$

$\sum z \log z$

Multiplicity







Let’s Practice

Follow instruction on GitHub:
https://github.com/nrp-nautilus/6nrp-hls4ml/tree/main

JupyterHub link:

- https://6nrp.nrp-nautilus.io/
- Log in with your university credentials via CILogon

Extra Slides

Quantization

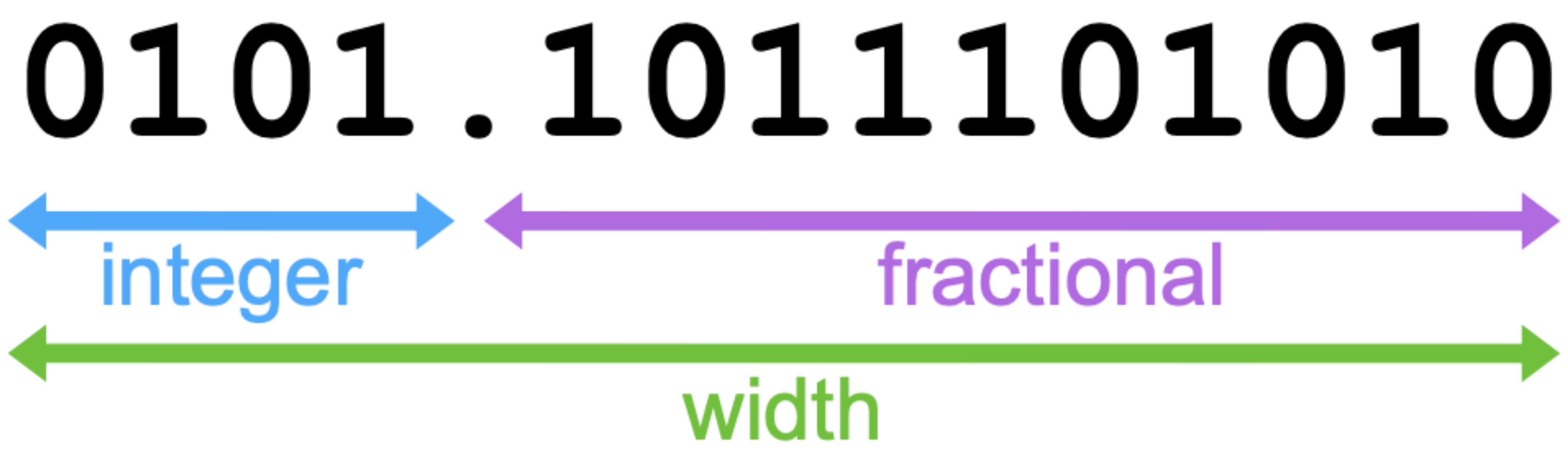
Quantization – Reducing the bit precision used for NN arithmetic

Why this is necessary?

- Floating-point operations (32 bit numbers) on an FPGA consumes large resources
- Not necessary to do it for desired performance

- **hls4ml uses fixed-point representation for all computations**
  - Operations are integer ops, but we can represent fractional values

ap_fixed<width bits, integer bits>



0101.1011101010

integer

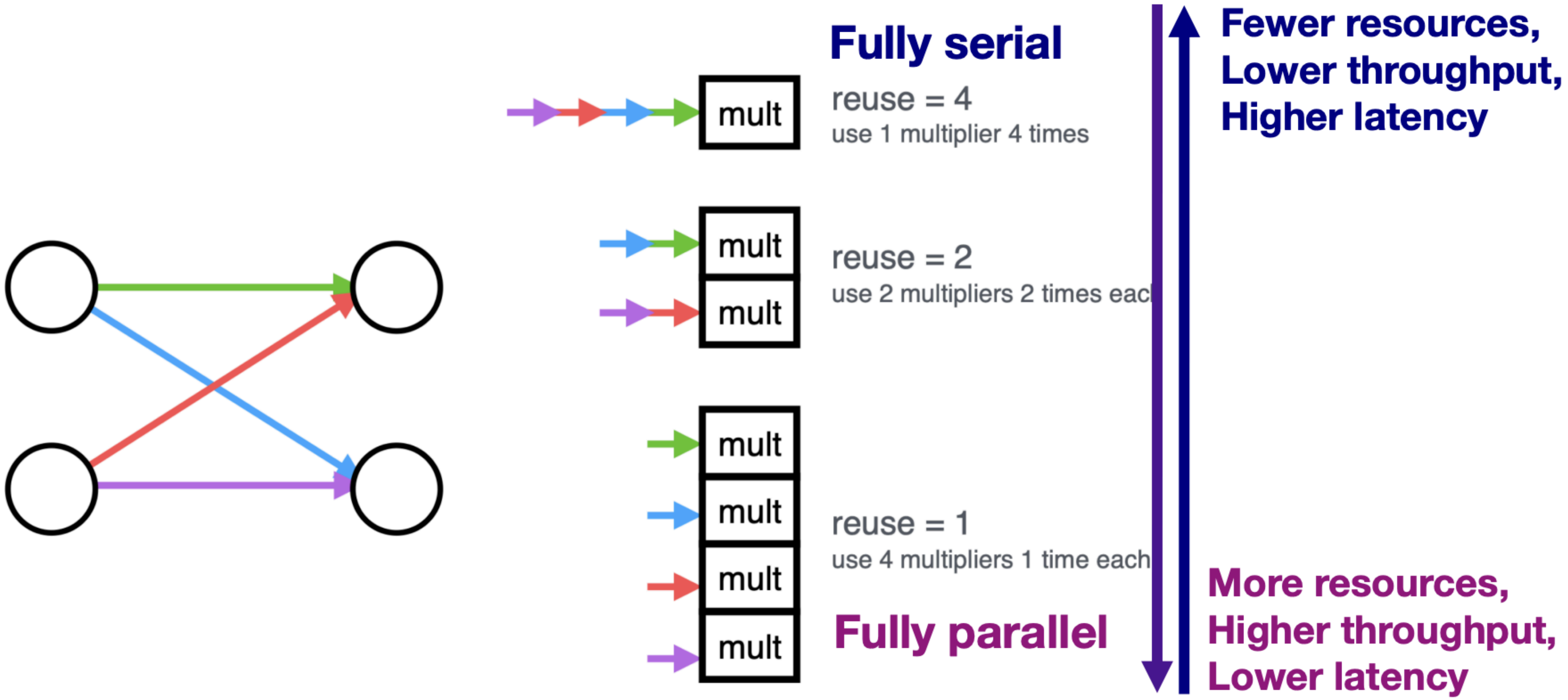
fractional

width

Parallelization

- Trade-off between latency and FPGA resource usage determined by the parallelization of the calculations in each layer

- Configure the “reuse factor” = number of times a multiplier is used to do a computation



Fully serial

reuse = 4
use 1 multiplier 4 times

reuse = 2
use 2 multipliers 2 times eac

reuse = 1
use 4 multipliers 1 time each

Fully parallel

**Fewer resources,
Lower throughput,
Higher latency**

**More resources,
Higher throughput,
Lower latency**

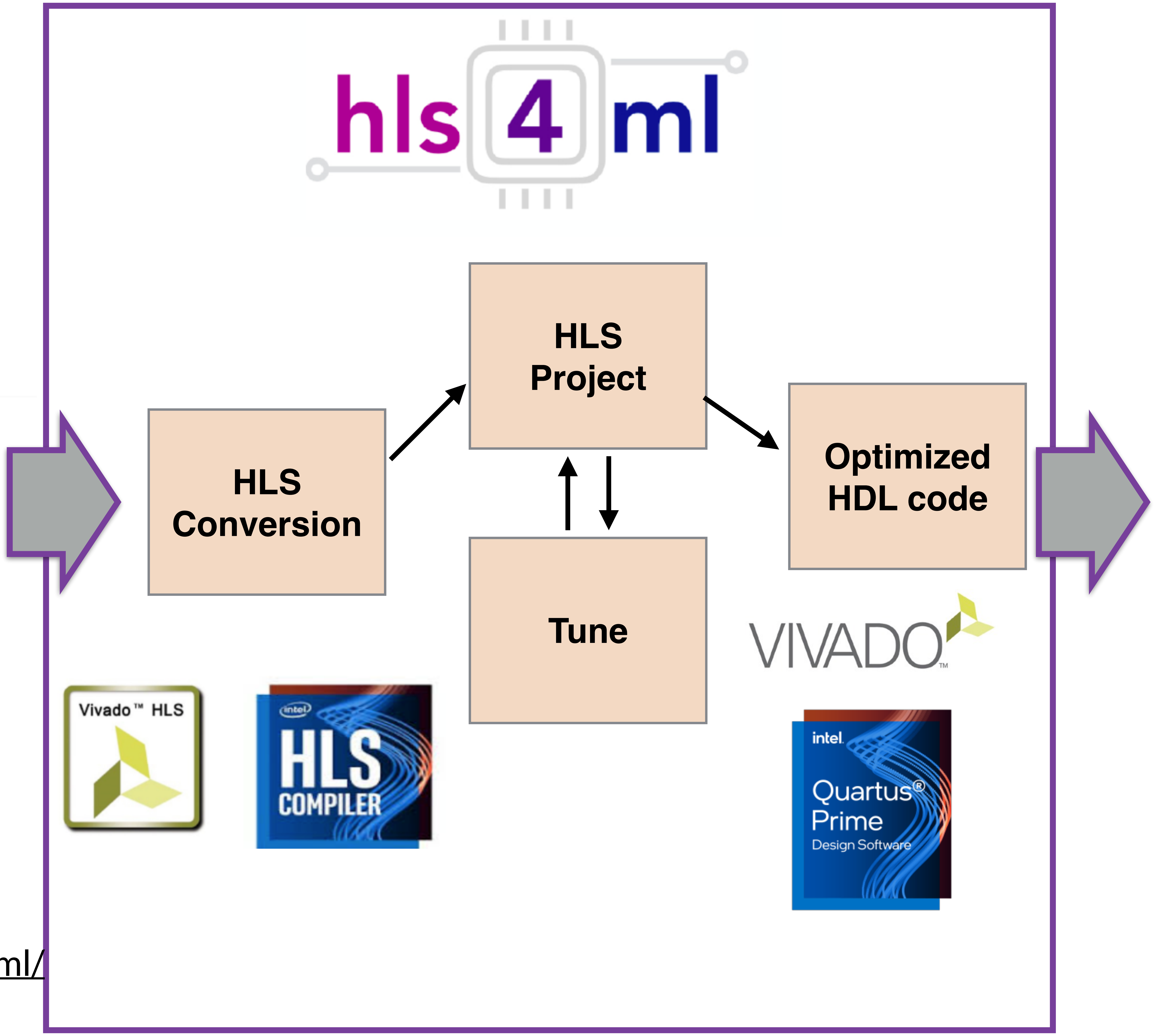
High-Level Synthesis for Machine Learning

Usual ML software workflow

(Q) K + TensorFlow

(Q) ONNX

PYTORCH

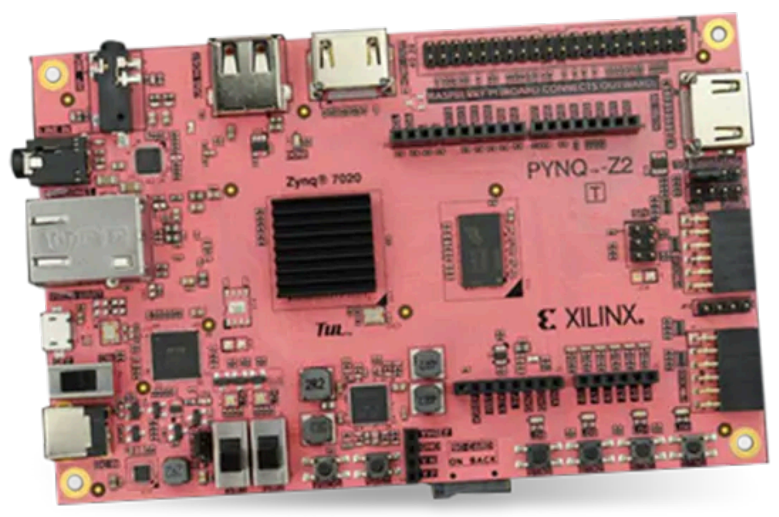


HLS Conversion

HLS Project

Tune

Optimized HDL code



Bit Files for FPGA

https://fastmachinelearning.org/hls4ml/
arXiv:2103.05579