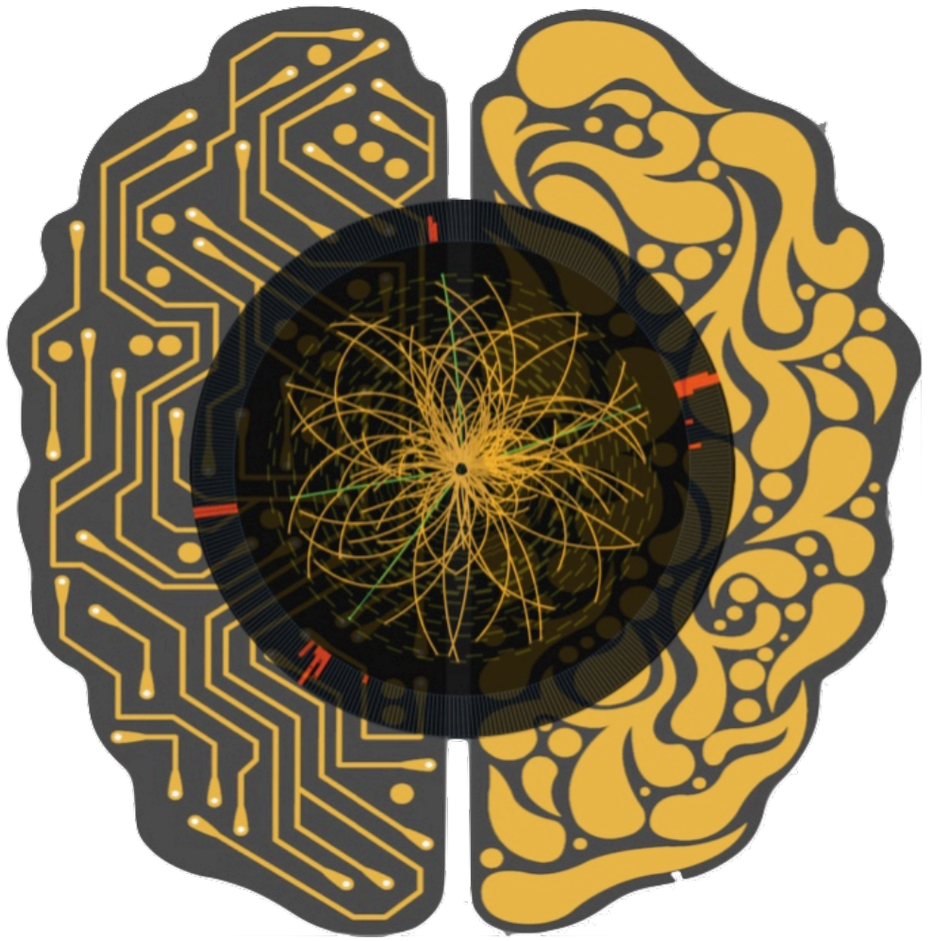
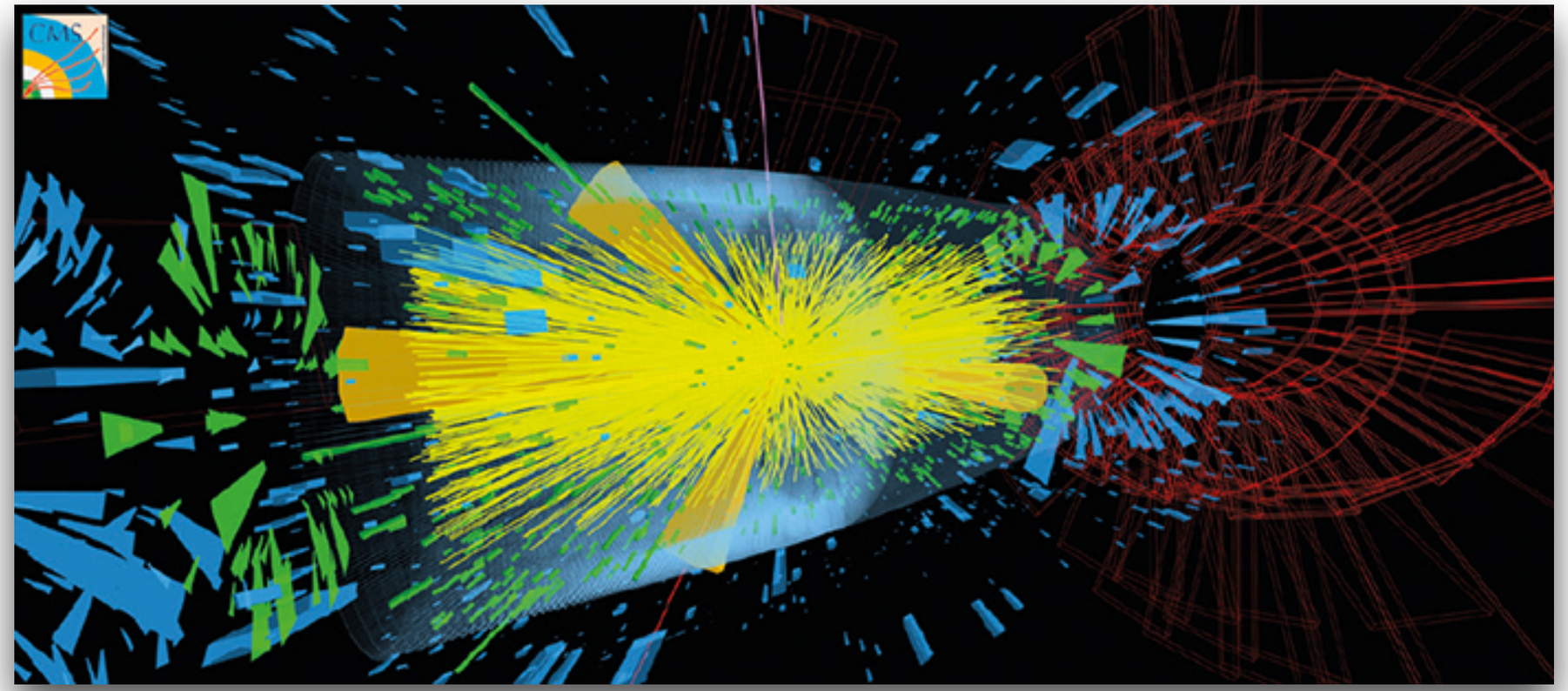


hls4ml Tutorial

Part I. Introduction

Fast Machine Learning Developer Bootcamp - Fermilab - 12 September 2019

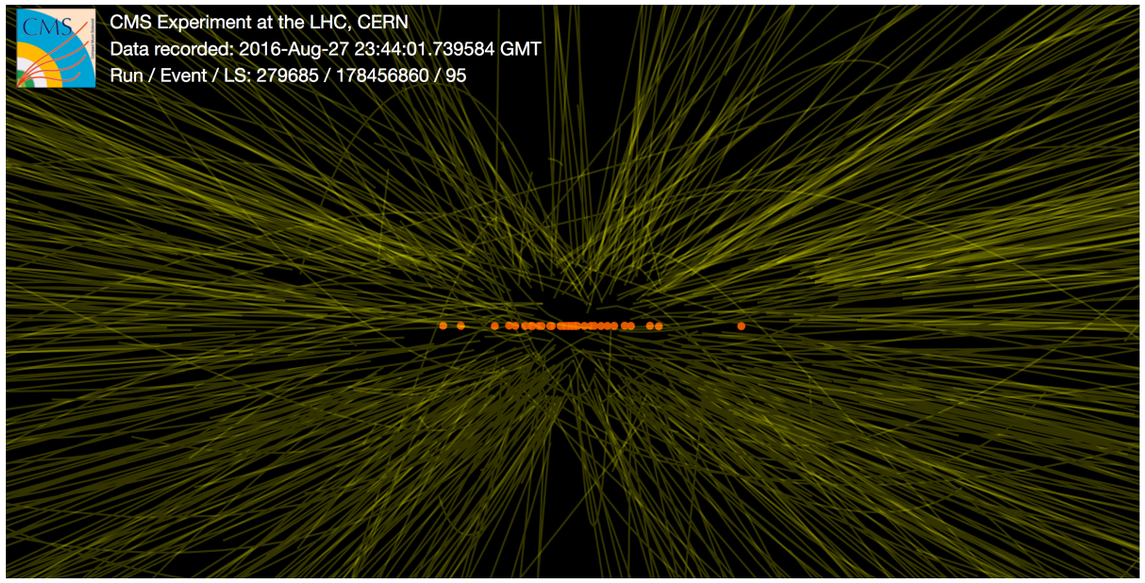


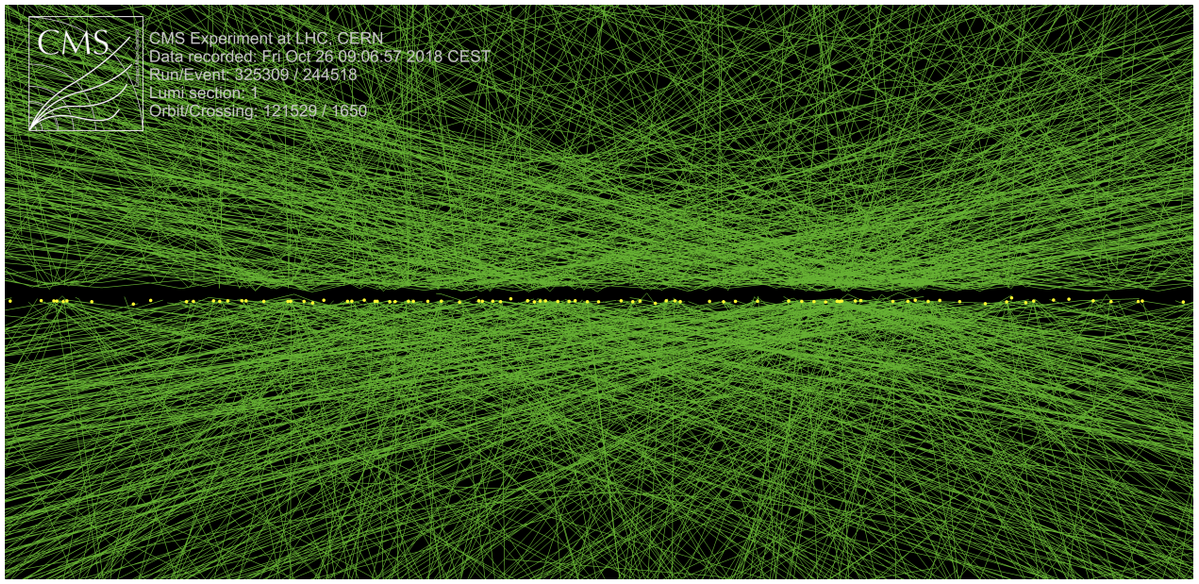
Motivation:

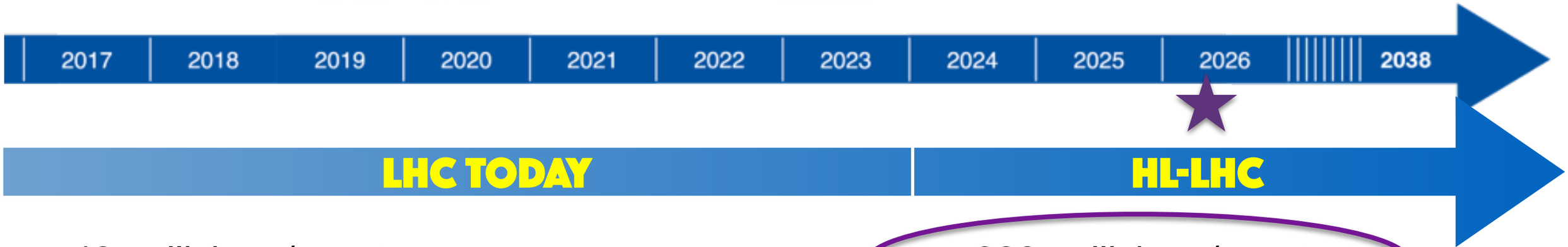
use cases at the LHC and beyond

Future challenges @ LHC

Extreme bunch crossing frequency of 40 MHz → extreme data rates O(100 TB/s)







- ~ 40 collisions/event
- ~ 10 sec/event processing time

- ~ 200 collisions/event

- more granular detector
- ~ minutes/event processing time
- flat budget for computing resources

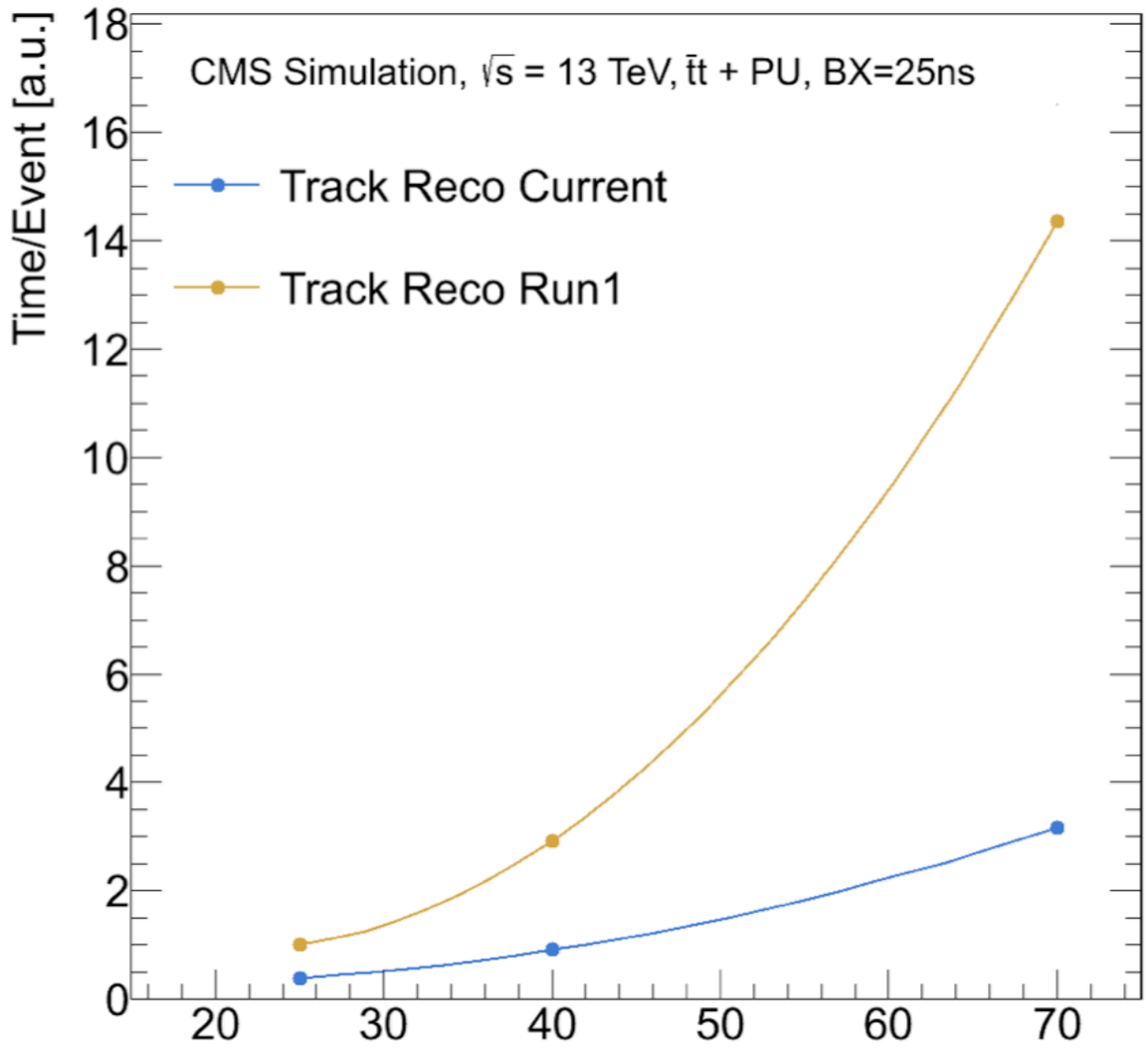
Future challenges @ HL-LHC

Modern machine learning methods might be the way out!

Current event reconstruction algorithms will not be sustainable

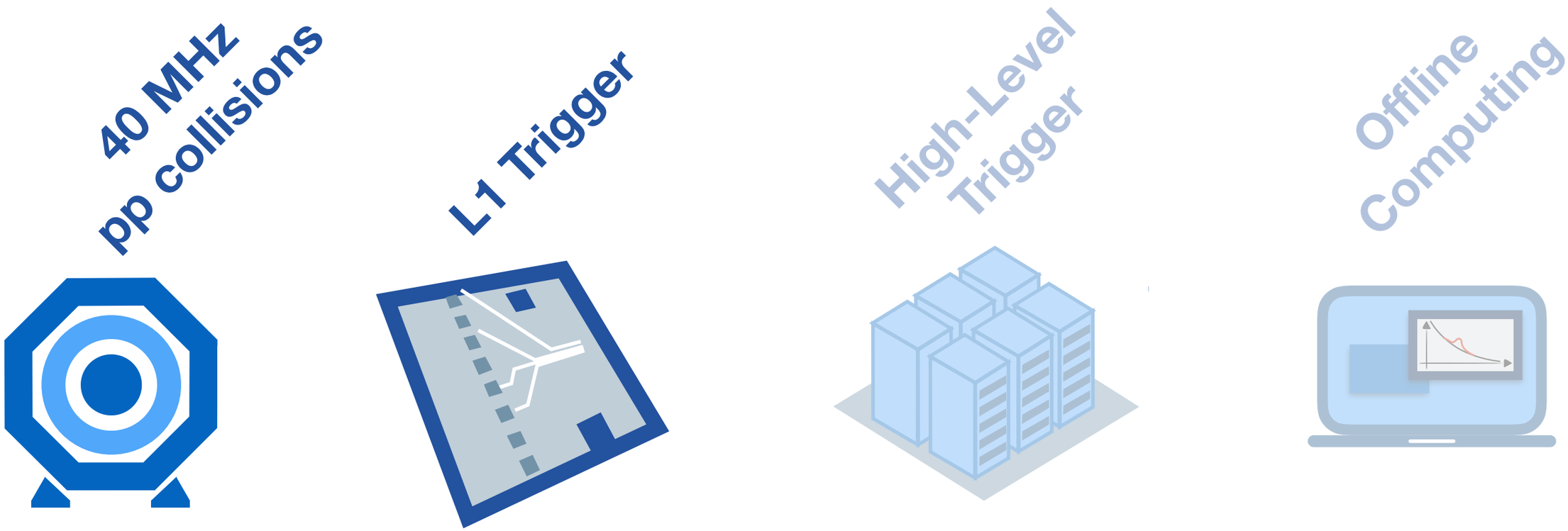
Recast instead the problem as a machine learning problem

- Excellent physics performance
- Intrinsically parallelizable → high speed
- Follow industry trends in developing new devices optimized for ML and speed the up the inference



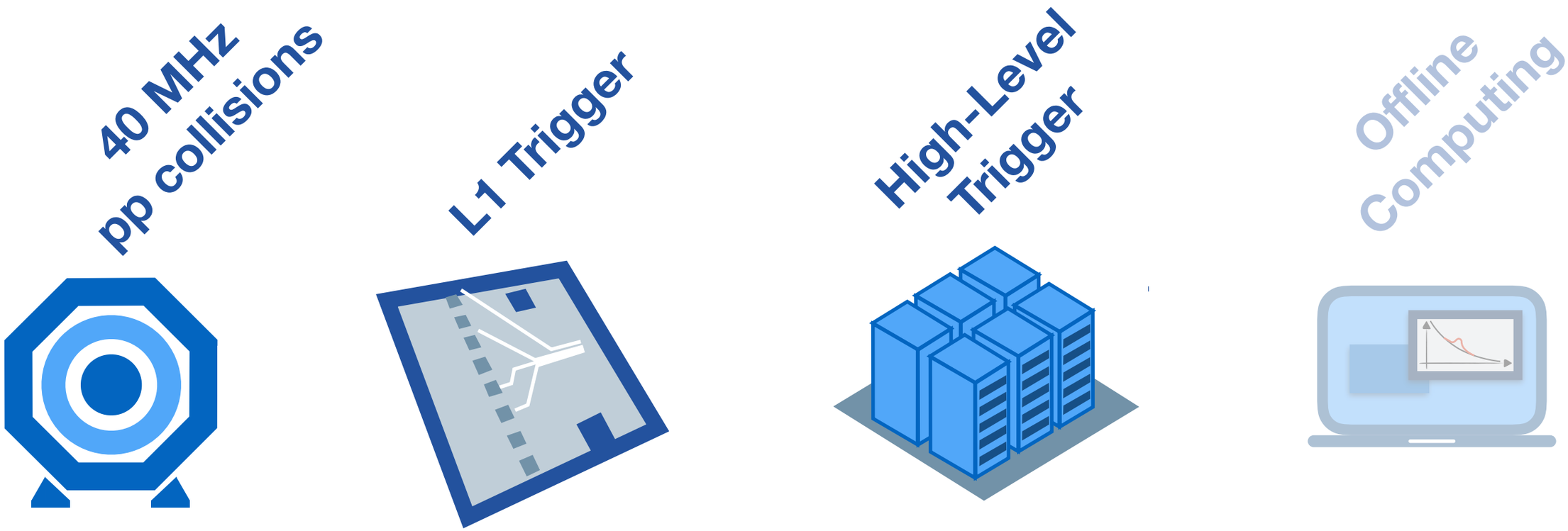
collisions/event ~ event complexity

The LHC big data problem



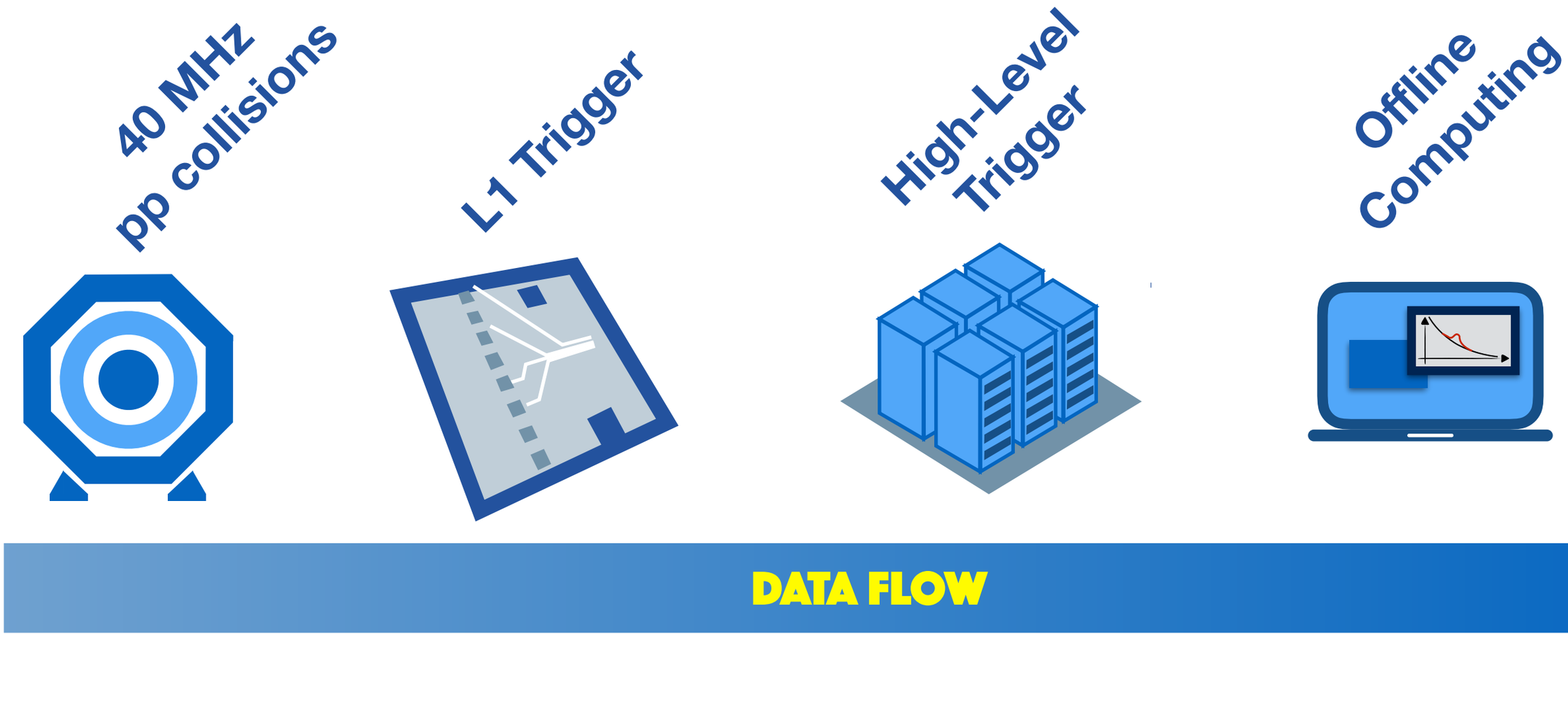
- 40 MHz in / 100 KHz out
- Absorbs 100s TB/s
- Trigger decision to be made in **~ 10 µs**
- Coarse local reconstruction
- FPGAs / Hardware implemented

The LHC big data problem



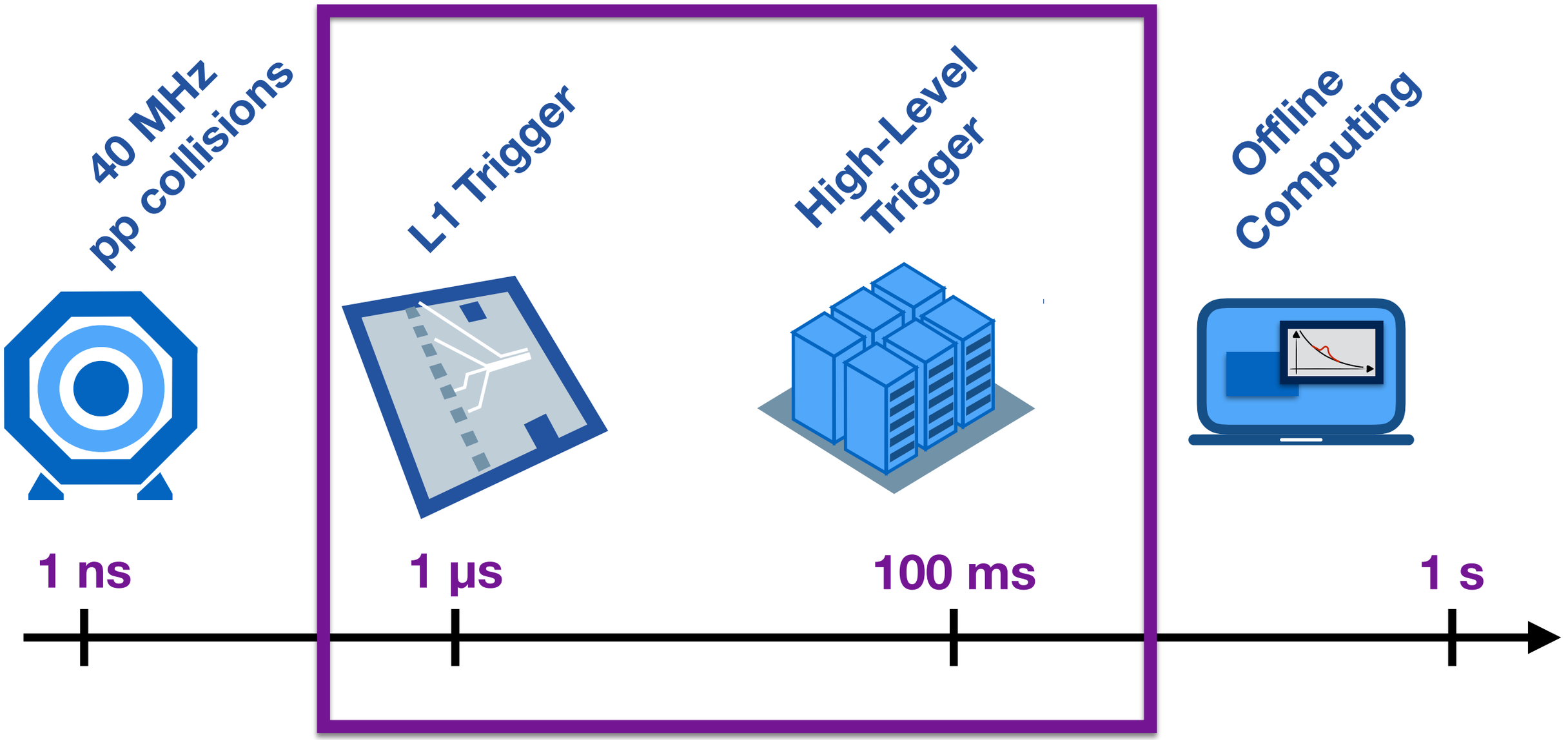
- 100 KHz in / 1 KHz out
- Output: ~ 500 KB/event
- Processing time **~ 300 ms**
- Simplified global reconstruction
- Software implemented on CPUs

The LHC big data problem

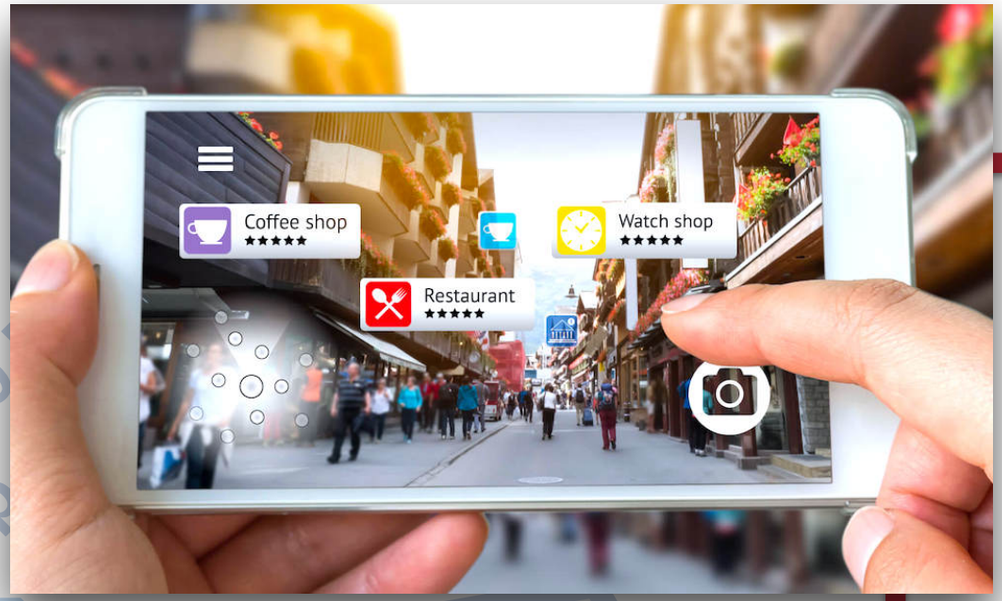


- Output: max. 1 MB/event
- Processing time **~ 20 s**
- Accurate global reconstruction
- Software implemented on CPUs

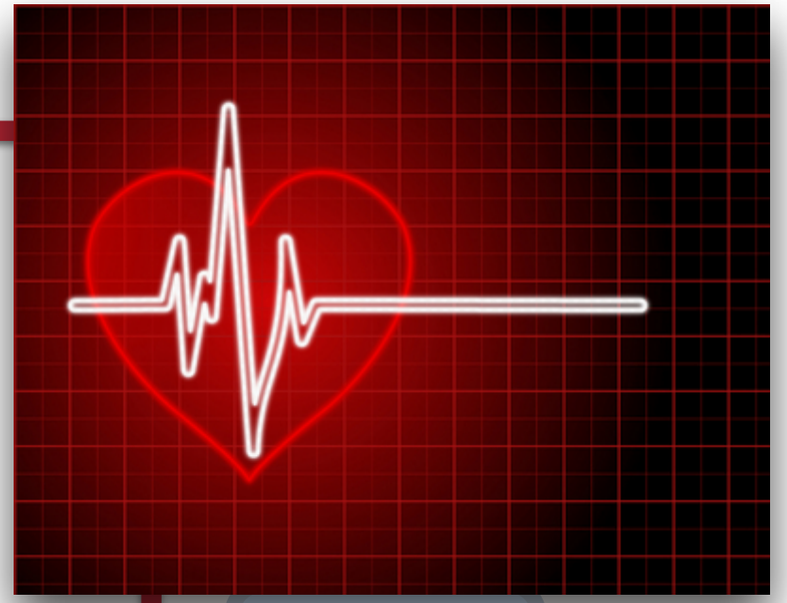
The LHC big data problem

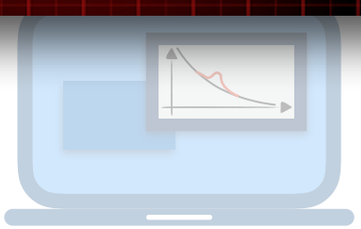


Deploy ML algorithms very early in the game
Challenge: strict latency constraints!



High-Level Trigger





100 ms

1 s

Beyond LHC

Ex: self-driving cars

A single self-driving test vehicle can produce **~ 30 TB/day**

There are over **250 million cars** on the road in the US alone

If **< 1% replaced by autonomous vehicles** by 2020

→ HUGE amount of data generated, not manageable by central servers!

Beyond LHC

Ex: self-driving cars

A single self-driving test vehicle can produce **~ 30 TB/day**

There are over **250 million cars** on the road in the US alone

If **< 1% replaced by autonomous vehicles** by 2020

→ HUGE amount of data generated, not manageable by central servers!

Need **edge computing architectures**, low power and small in size to run powerful data analytics programs oboard

NB: latency matters! even a few milliseconds of delay can result in an accident!

The stakes are too high to wait the answer from a distant cloud server.

FGPA Vs GPU: Autonomous Car Makers In Silicon Valley Have Definitely Chosen A Side When It Comes To AI Chips

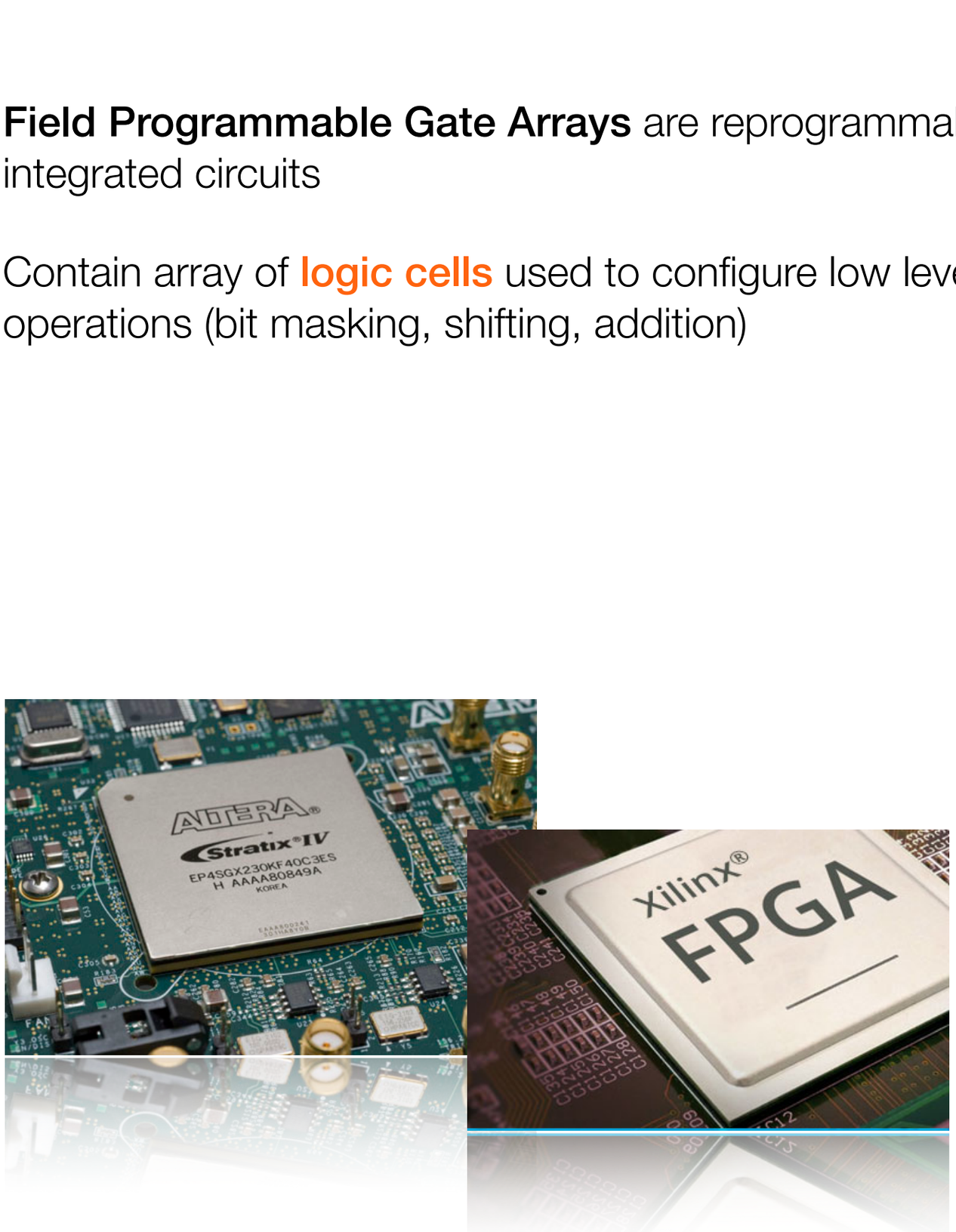
RICHA BHATIA - MAY 9, 2018

PEOPLE MIGHT HAVE DIFFERENT OPINION... BUT TODAY WE LEARN ABOUT FPGA & MACHINE LEARNING!

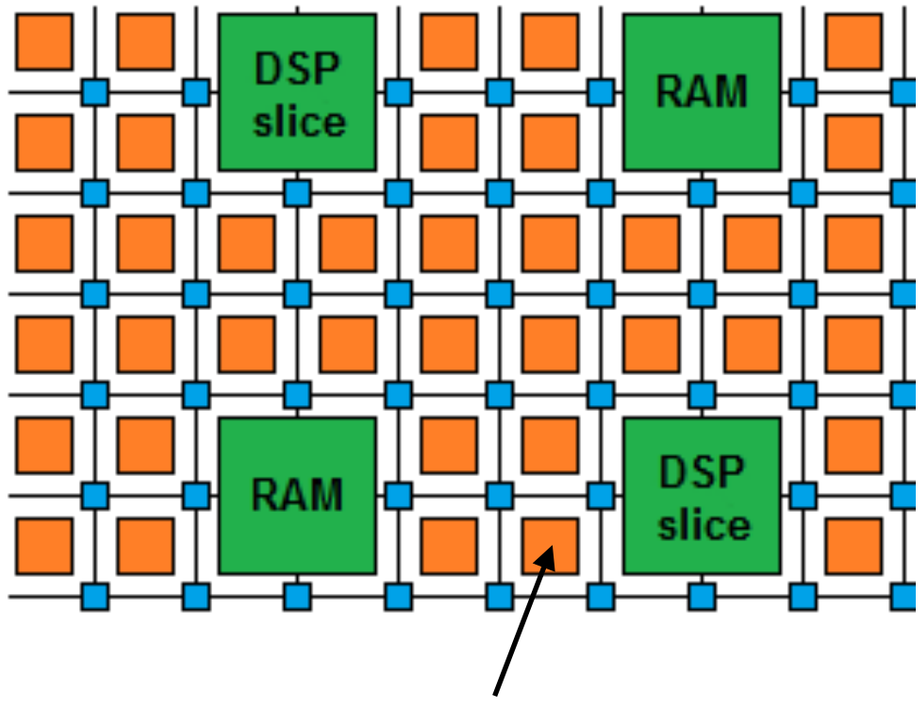
What are FPGAs?

Field Programmable Gate Arrays are reprogrammable integrated circuits

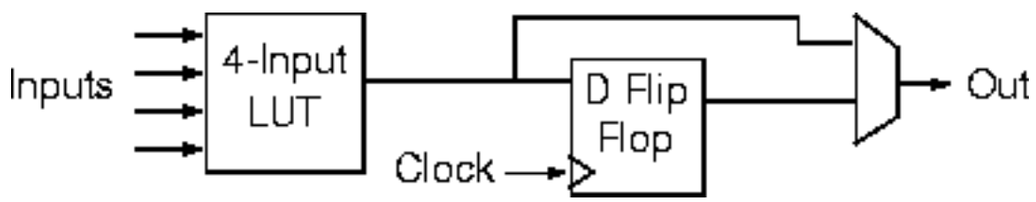
Contain array of **logic cells** used to configure low level operations (bit masking, shifting, addition)



FPGA diagram



Logic cell



Look-up table

(logic)

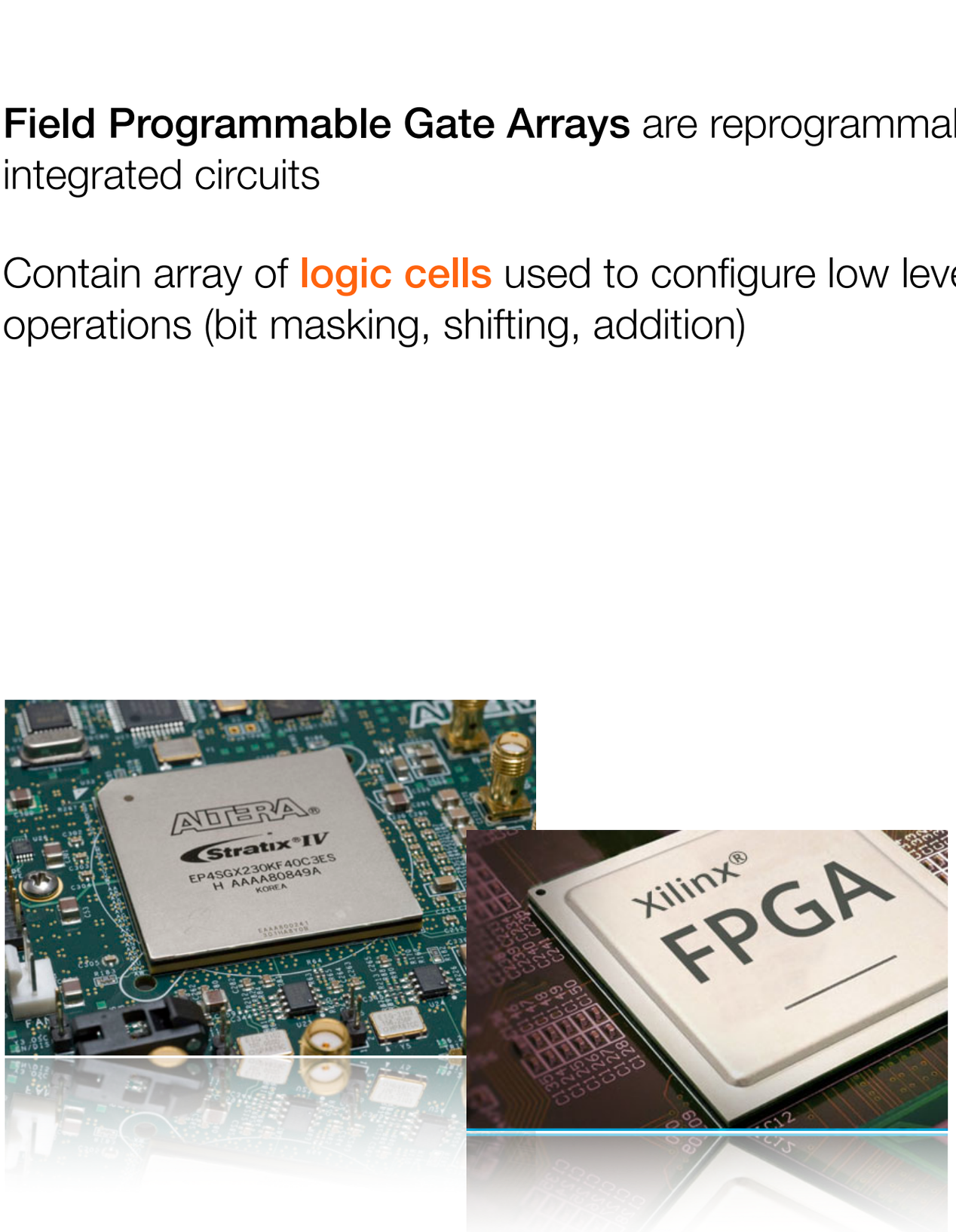
Flip-flop

(registers)

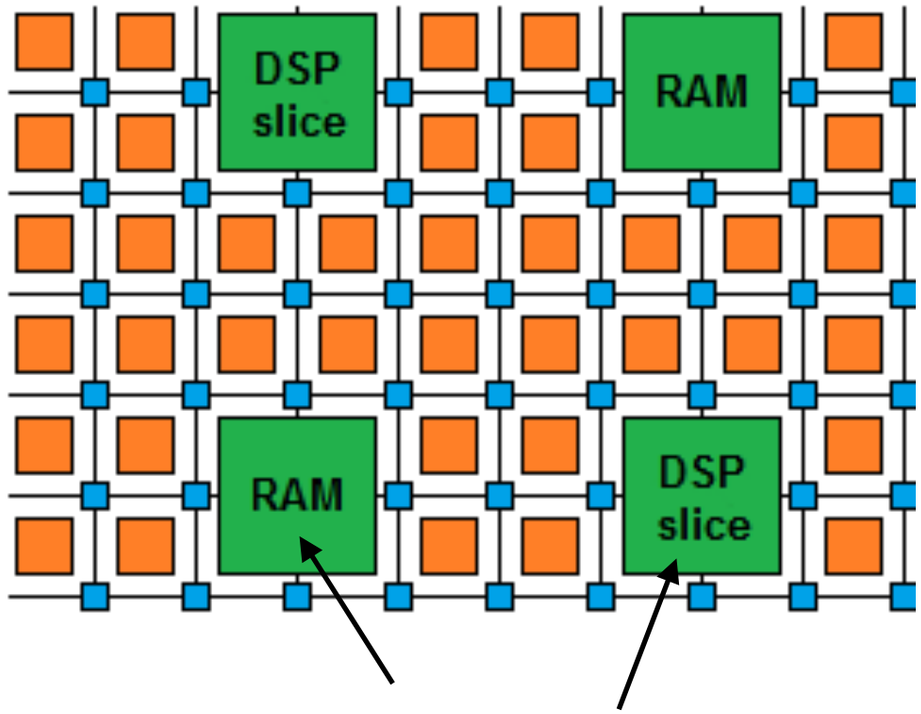
What are FPGAs?

Field Programmable Gate Arrays are reprogrammable integrated circuits

Contain array of **logic cells** used to configure low level operations (bit masking, shifting, addition)



FPGA diagram



Also contain embedded components:

Digital Signal Processors (DSPs):
logic units used for multiplications

Random-access memories (RAMs):
embedded memory elements

What are FPGAs?

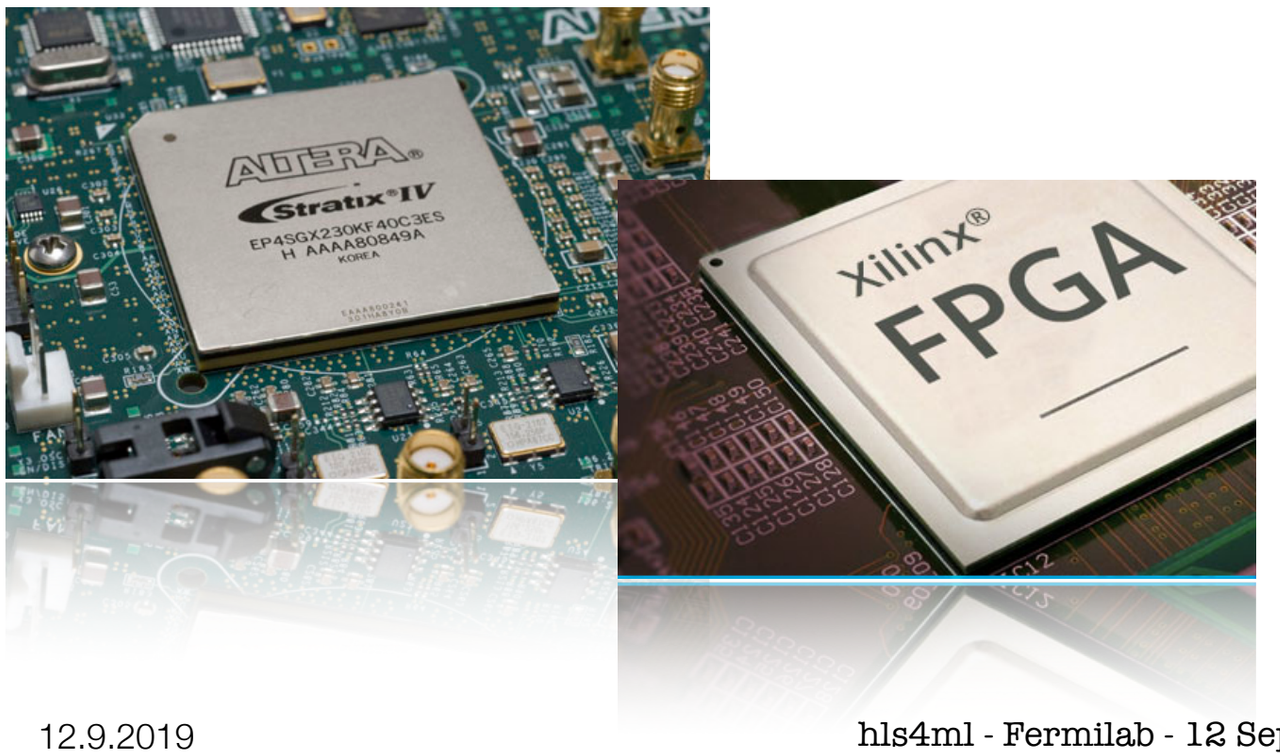
Field Programmable Gate Arrays are reprogrammable integrated circuits

Contain array of **logic cells** embedded with **DSPs**, **BRAMs**, etc.

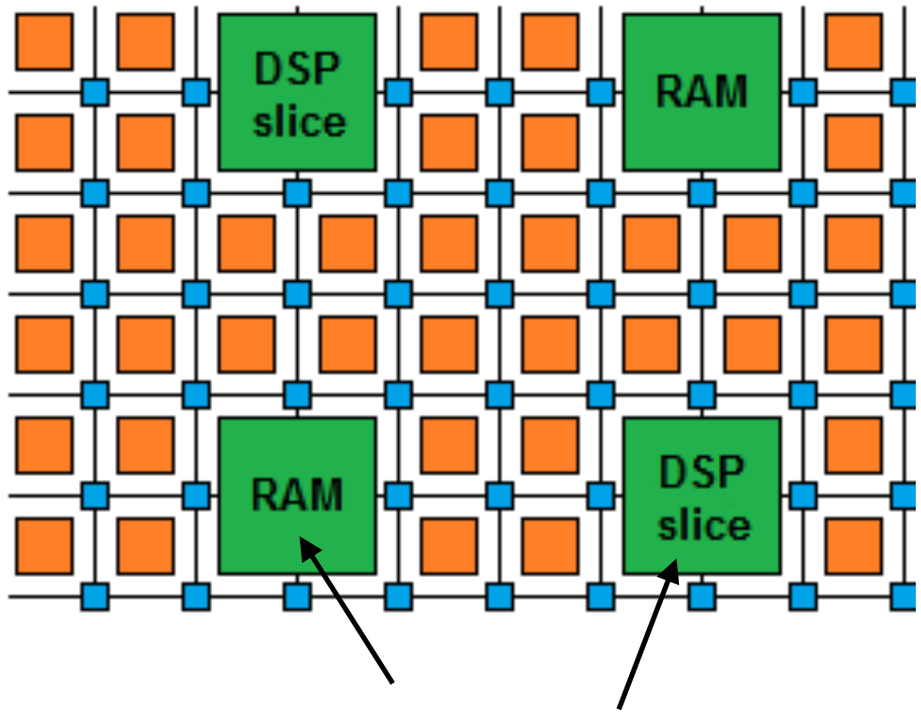
High speed input/output to handle the large bandwith

Support **highly parallel** algorithm implementations

Low power (relative to CPU/GPU)



FPGA diagram



Digital Signal Processors (DSPs): logic units used for multiplications

Random-access memories (RAMs): embedded memory elements

Flip-flops (FF) and look up tables (LUTs) for additions

How are FPGAs programmed?

Hardware Description Languages

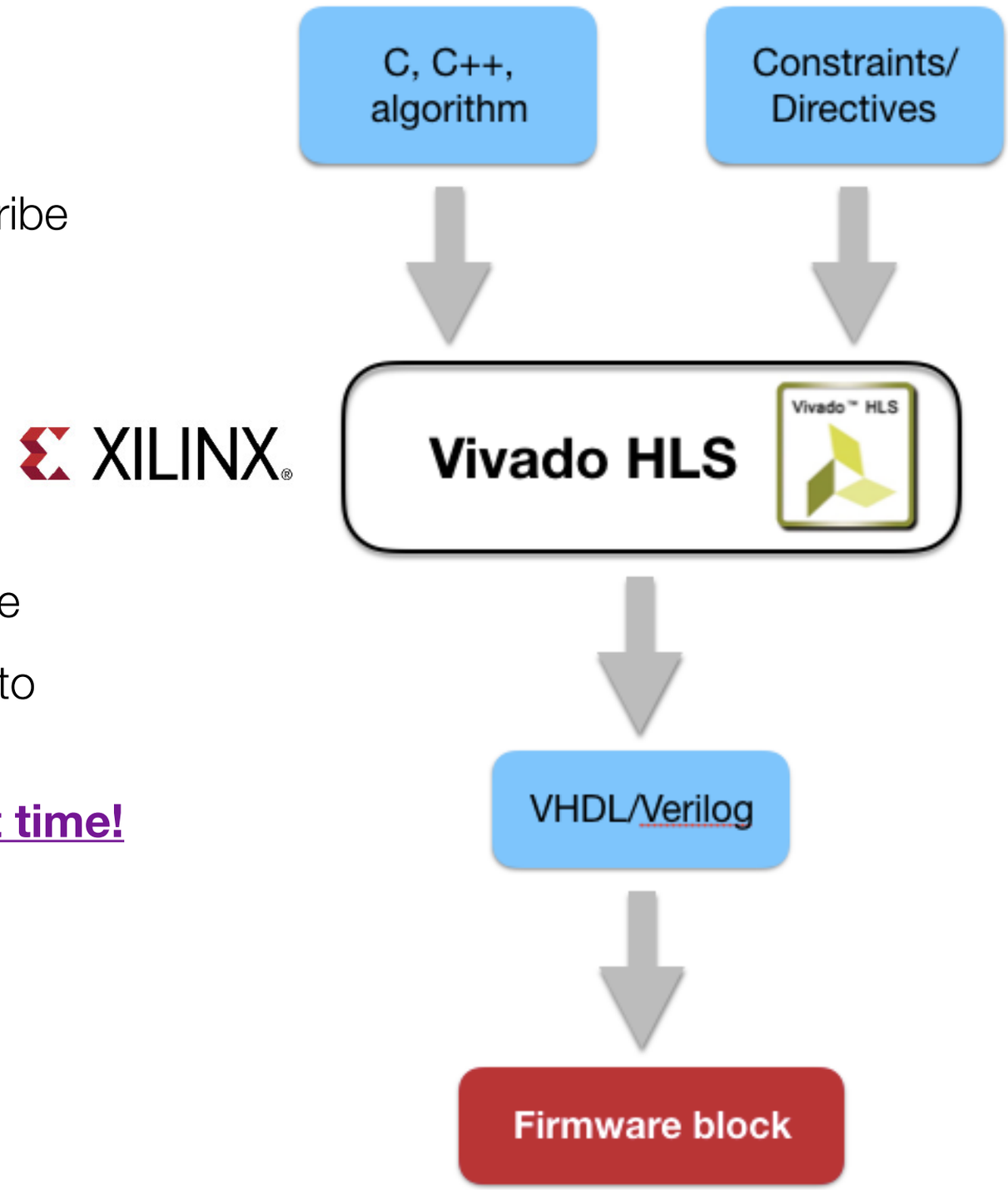
HDLs are programming languages which describe electronic circuits

High Level Synthesis

generate HDL from more common C/C++ code

pre-processor directives and constraints used to optimize the timing

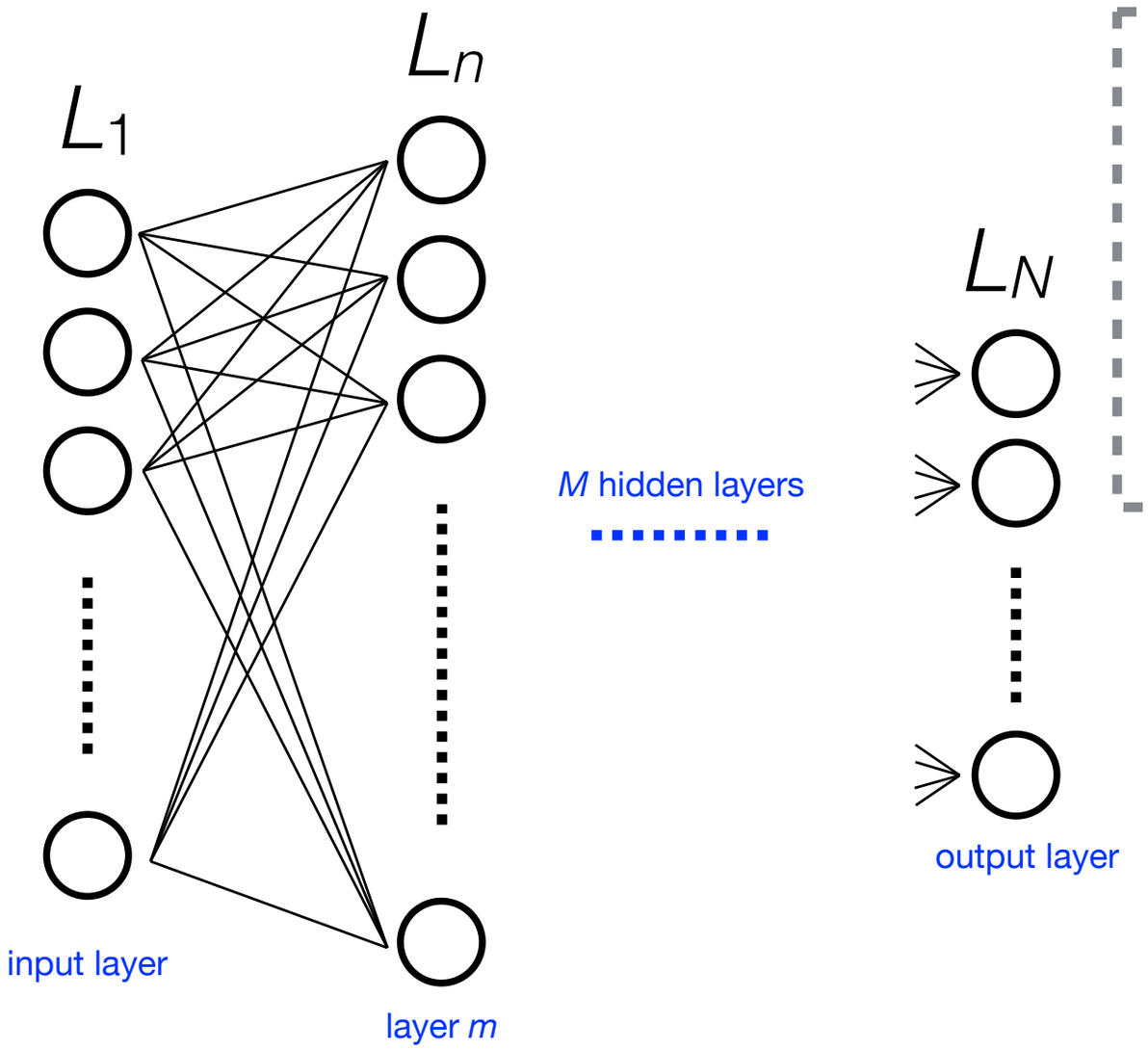
drastic decrease in firmware development time!



We use today **Xilinx Vivado HLS [*]**

[*] *https://www.xilinx.com/support/documentation/sw_manuals/xilinx2014_1/ug902-vivado-high-level-synthesis.pdf*

Neural network inference



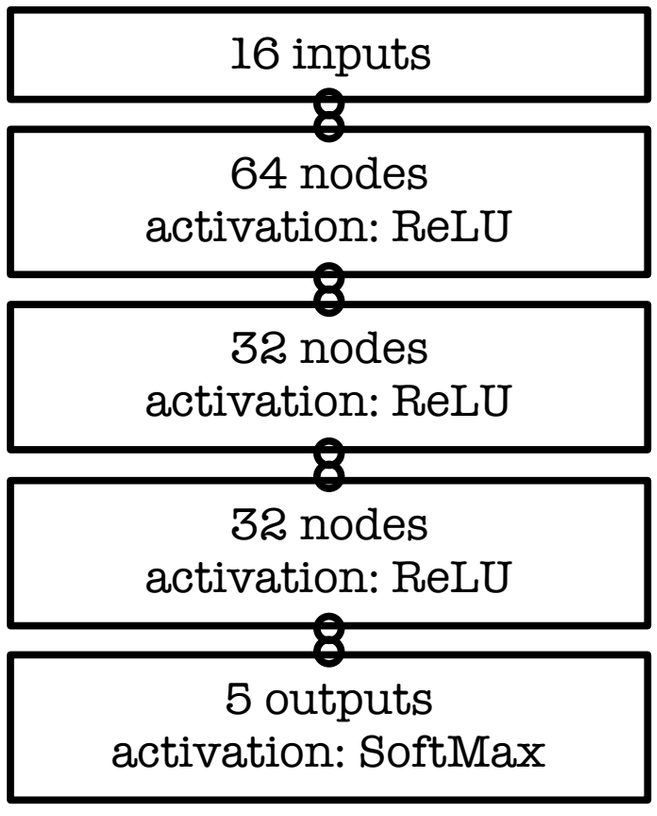
$$\mathbf{x}_n = g_n(\mathbf{W}_{n,n-1}\mathbf{x}_{n-1} + \mathbf{b}_n)$$

activation function — precomputed and stored in BRAMs

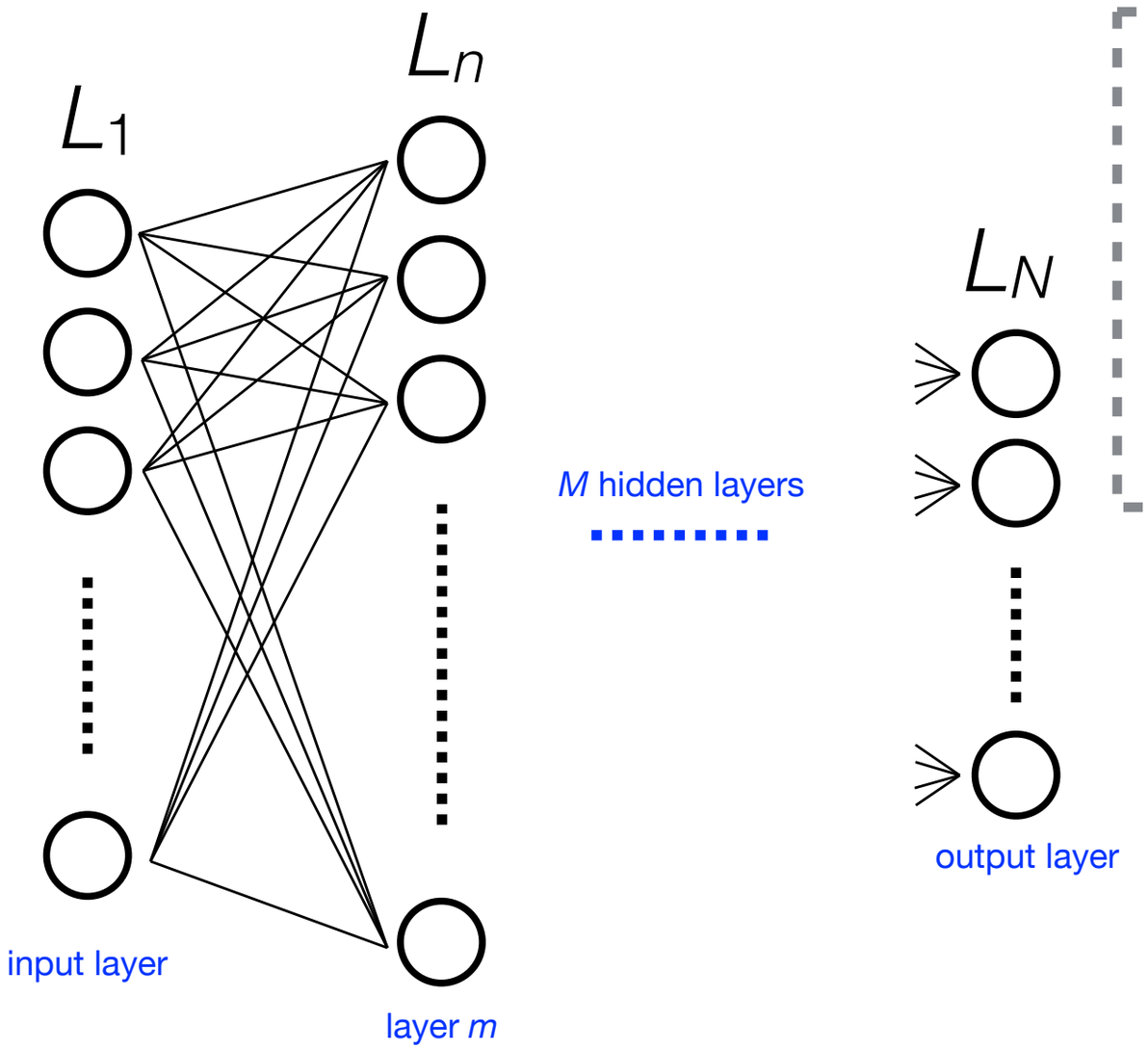
multiplication — DSPs

addition — logic cells

$$N_{\text{multiplications}} = \sum_{n=2}^{N} L_{n-1} \times L_n$$



Neural network inference



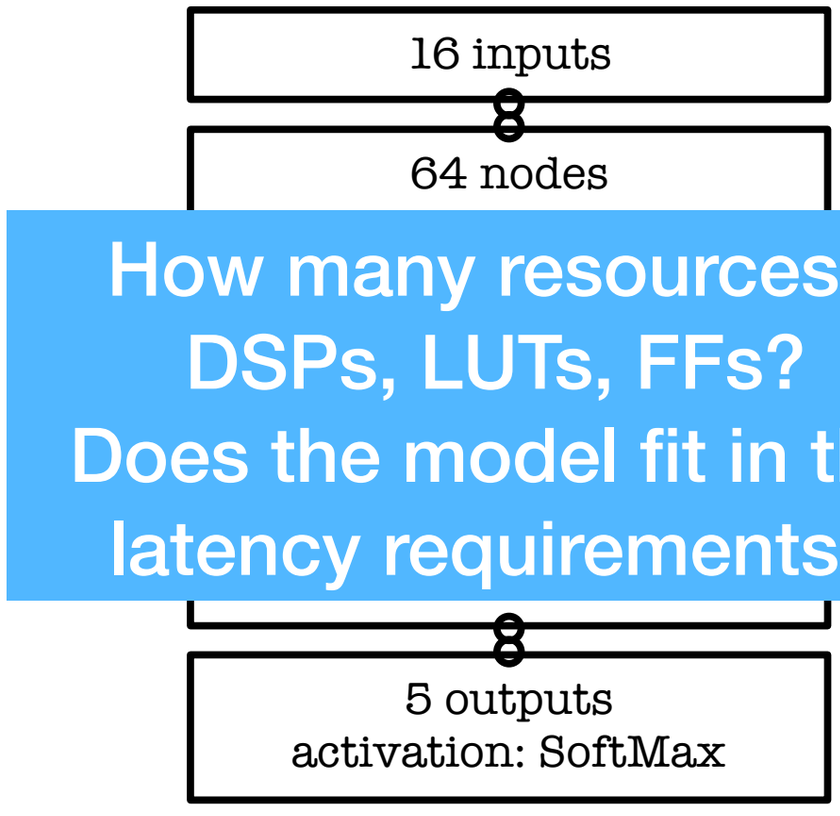
$$\mathbf{x}_n = g_n(\mathbf{W}_{n,n-1}\mathbf{x}_{n-1} + \mathbf{b}_n)$$

activation function
precomputed and stored in BRAMs

multiplication
DSPs

addition
logic cells

$$N_{\text{multiplications}} = \sum_{n=2}^{N} L_{n-1} \times L_n$$

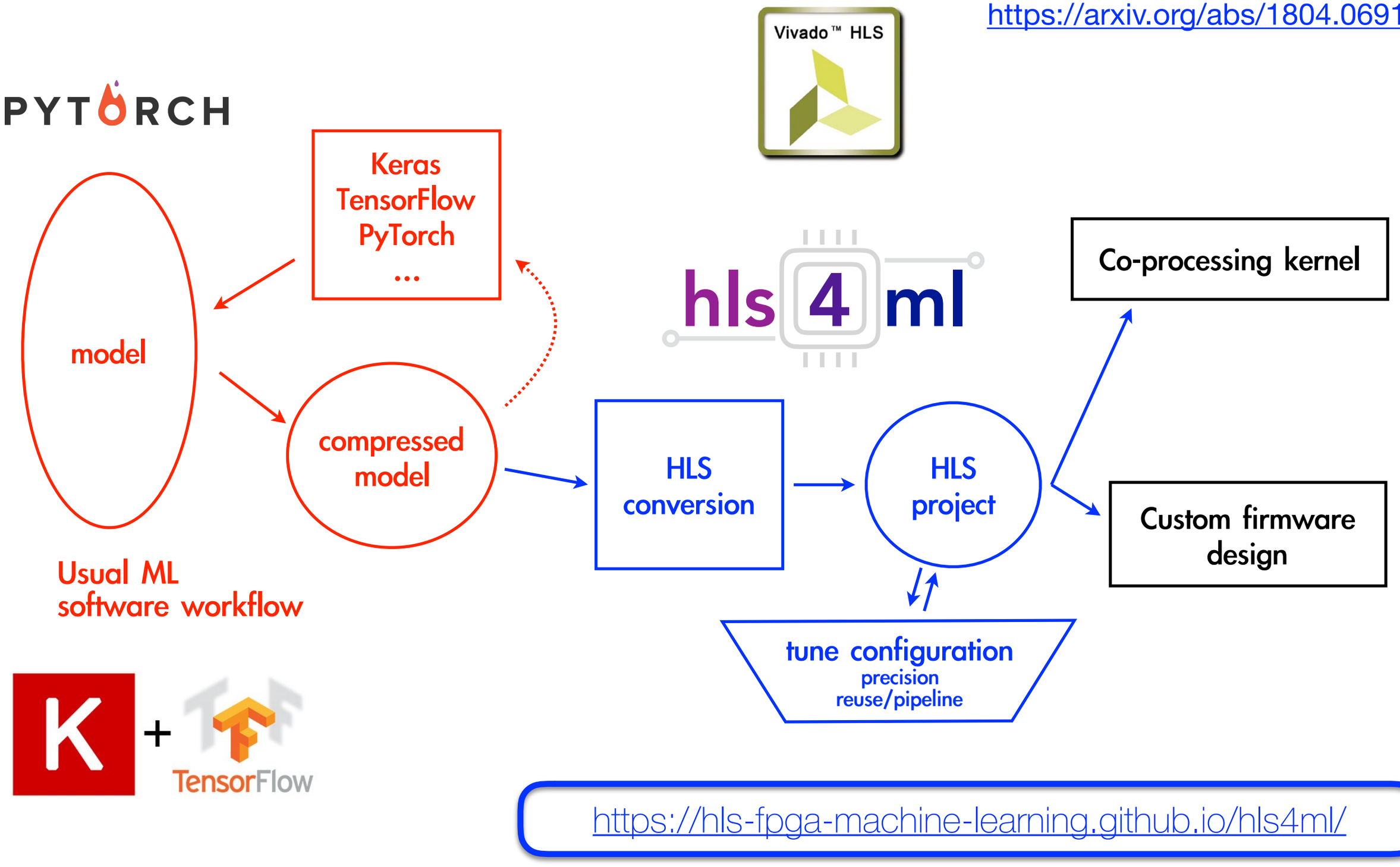


How many resources?
DSPs, LUTs, FFs?
Does the model fit in the latency requirements?

Today you are going to implement a NN on FPGA with this package:

high level synthesis for machine learning

https://arxiv.org/abs/1804.06913



https://hls-fpga-machine-learning.github.io/hls4ml/

Efficient NN design for FPGAs

FPGAs provide huge flexibility

Performance depends on how well you take advantage of this

Constraints:

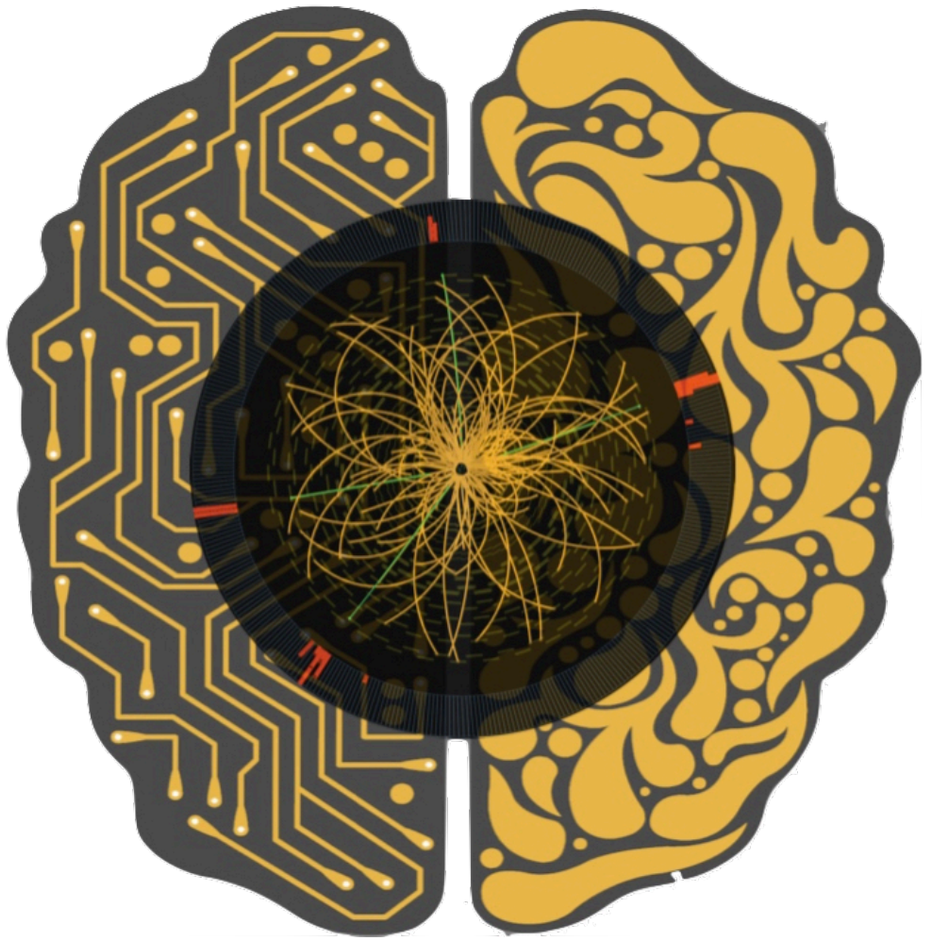
- Input bandwidth
- FPGA resources
- Latency

<u>Today you will learn how to optimize your project through:</u>

- **compression:** reduce number of synapses or neurons
- **quantization:** reduces the precision of the calculations (inputs, weights, biases)
- **parallelization**: tune how much to parallelize to make the inference faster/slower versus FPGA resources

Today's hls4ml hands on

- First part:
  - take confidence with the package, its functionalities and design synthesis by running with one of the provided trained NN
  - learn how to read out an estimate of FPGA resources and latency for a NN after synthesis
  - learn how to optimize the design with quantization and parallelization
- Second part:
  - learn how to run the design on Amazon Web Services FPGAs with SDAccel
  - timing and resources studies after running on real FPGA
- Third part:
  - learn how to do model compression and its effect on the FPGA resources/latency



hls4ml Tutorial

Part I. Introduction - Hands On

Fast Machine Learning Developer Bootcamp - Fermilab - 12 September 2019

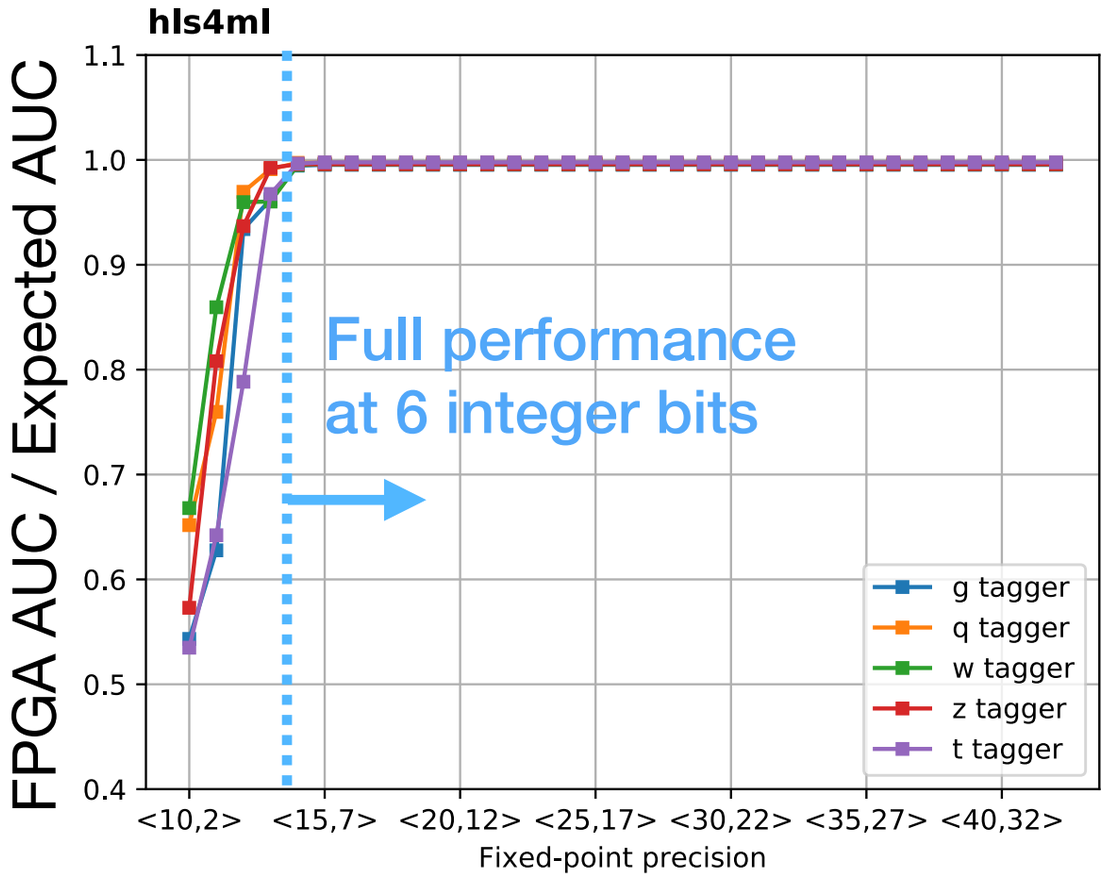
Efficient NN design: quantization

ap_fixed<width,integer>

- Quantify the performance of the classifier with the AUC
- Expected AUC = AUC achieved by 32-bit floating point inference of the neural network

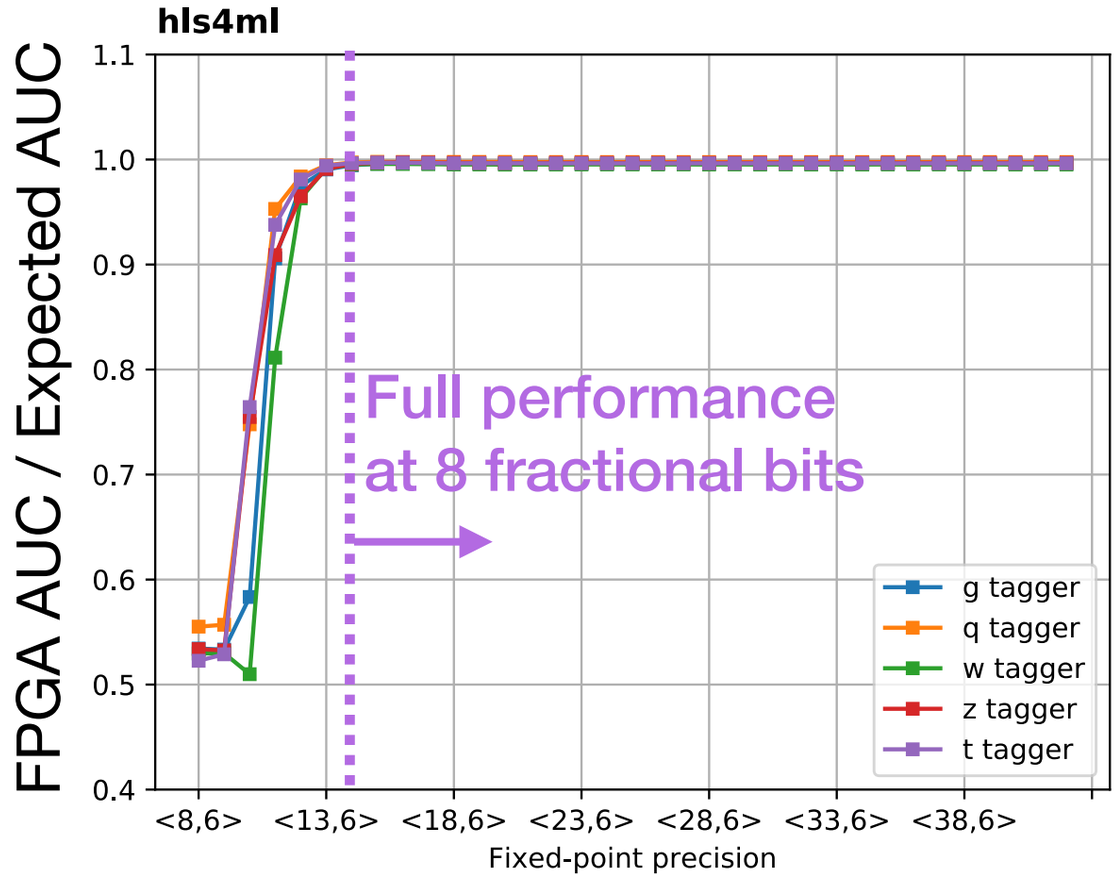
Scan integer bits

Fractional bits fixed to 8



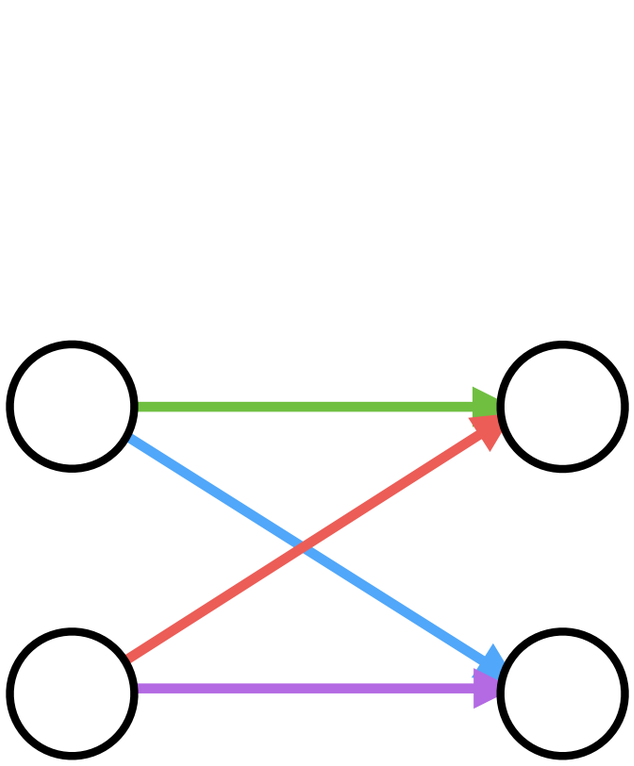
Scan fractional bits

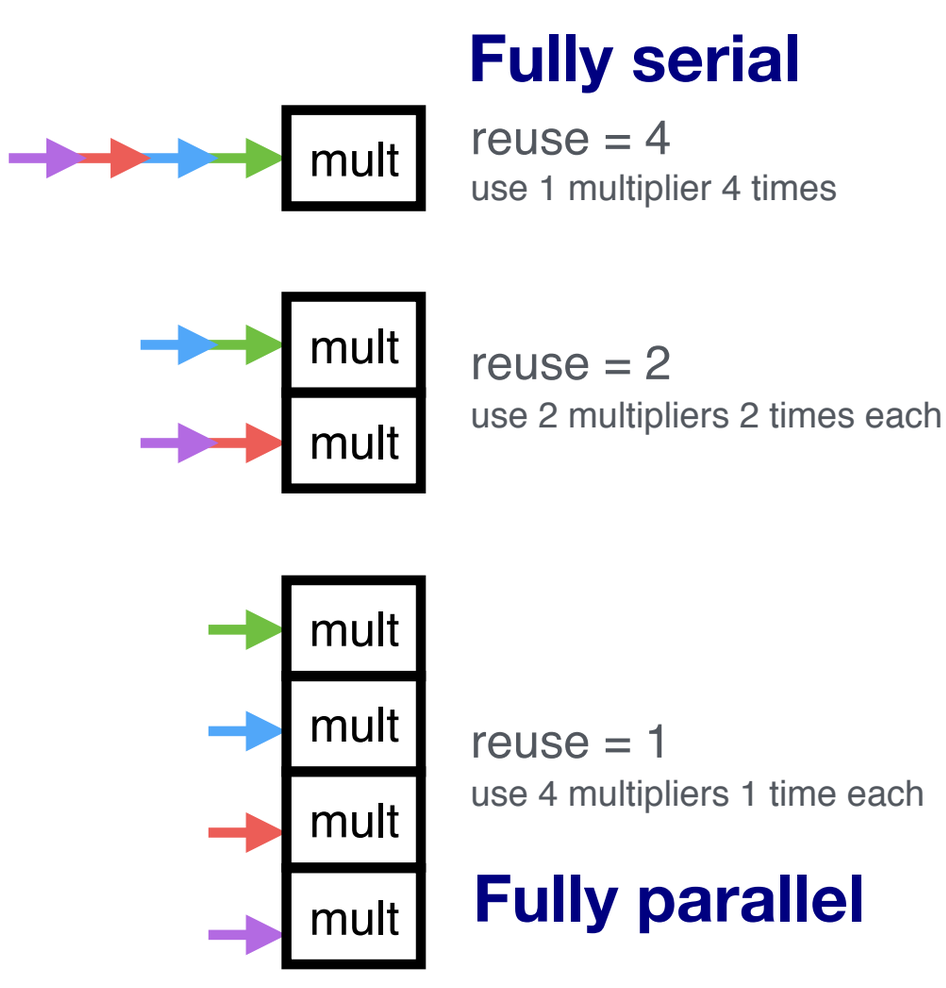
Integer bits fixed to 6



Efficient NN design: 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





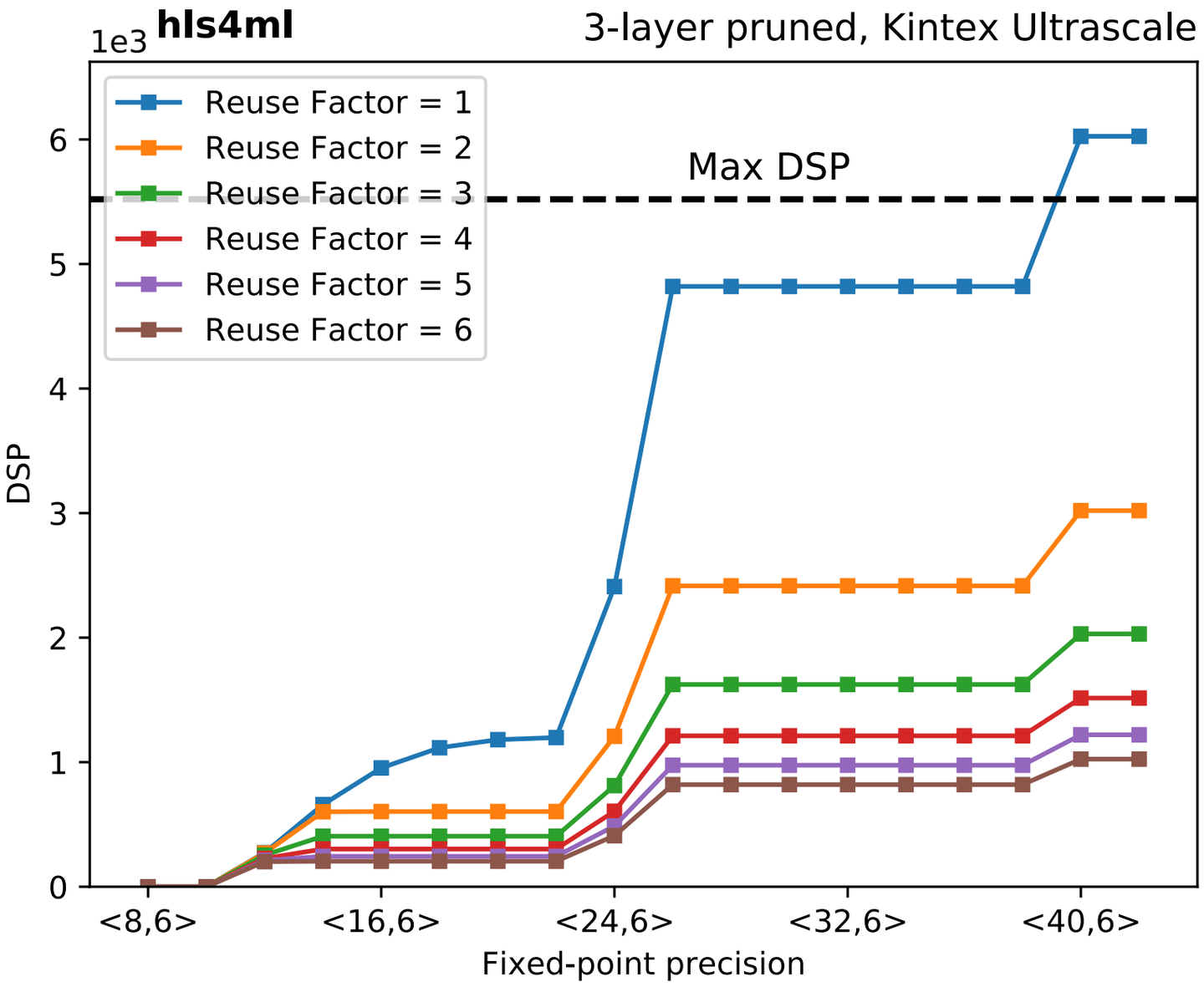
Fully serial

Fully parallel

Less resources/
Less throughput

Reuse factor: how much to parallelize operations in a hidden layer

Parallelization: DSPs usage



Fully parallel
Each mult. used 1x

Each mult. used 2x

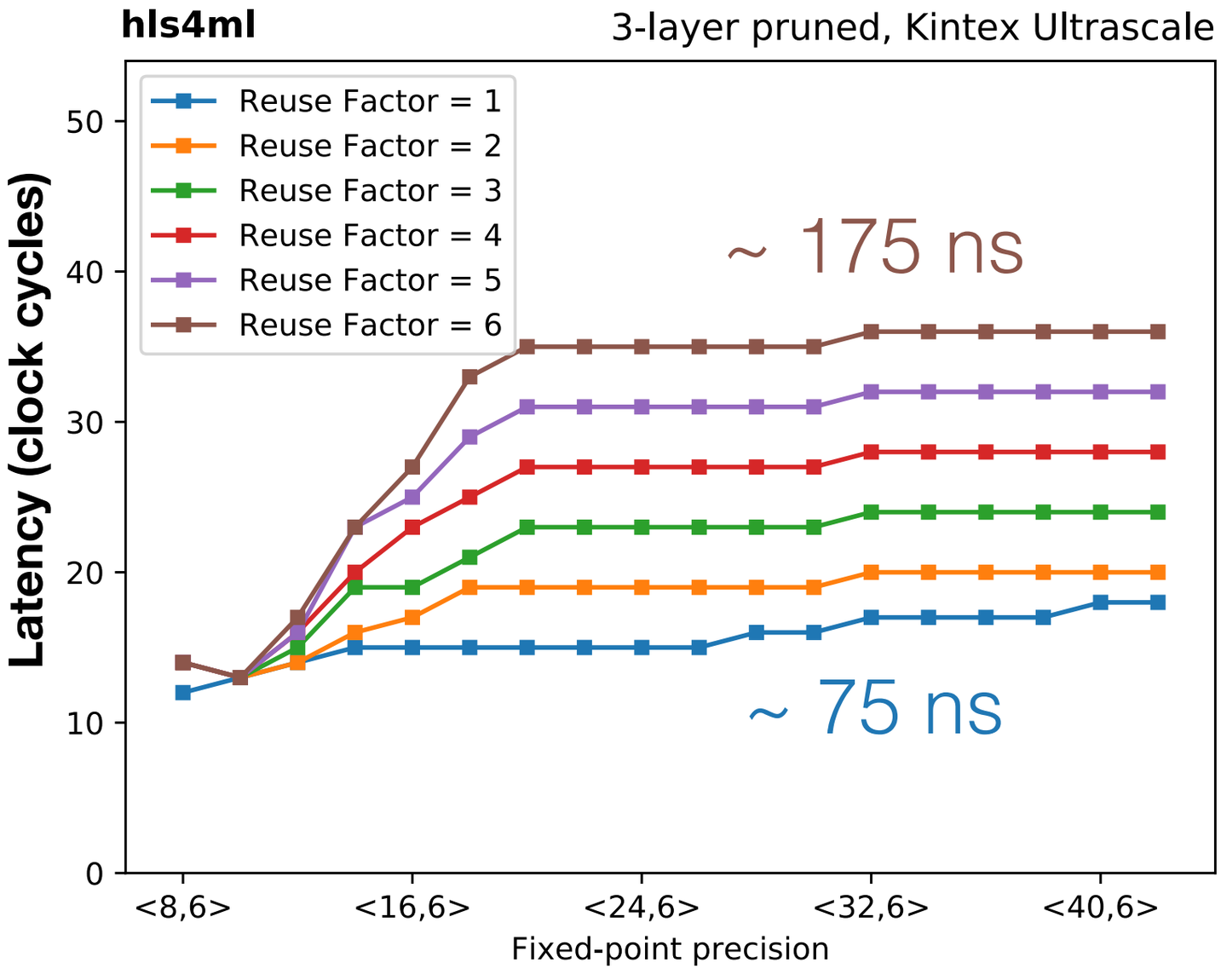
Each mult. used 3x

⋮

Parallelization: Timing

Latency of layer m

$$L_m = L_{\text{mult}} + (R-1) \times II_{\text{mult}} + L_{\text{activ}}$$



Each mult. used 6x

Each mult. used 3x

Fully parallel
Each mult. used 1x

Longer latency

More resources

Exercise

- https://github.com/FPGA4HEP/course_material/blob/fml/part1_hls4ml_intro.md