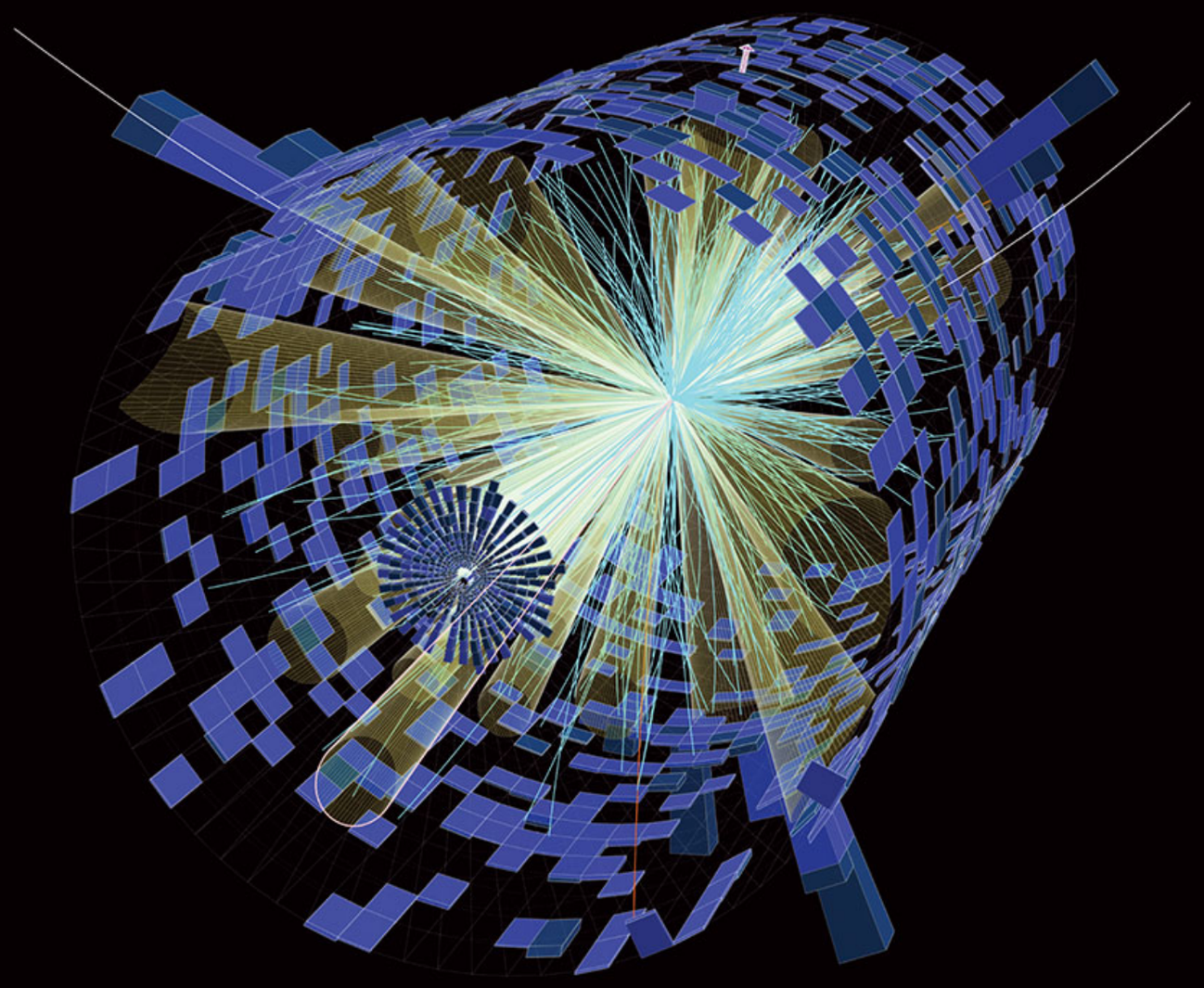
Ultra Low-latency and Low-area Machine Learning at the Edge

Jennifer Ngadiuba (Fermilab)

INFN School of Statistics 2022
May 15 – 20, 2022
Paestum, Italy



Fermilab

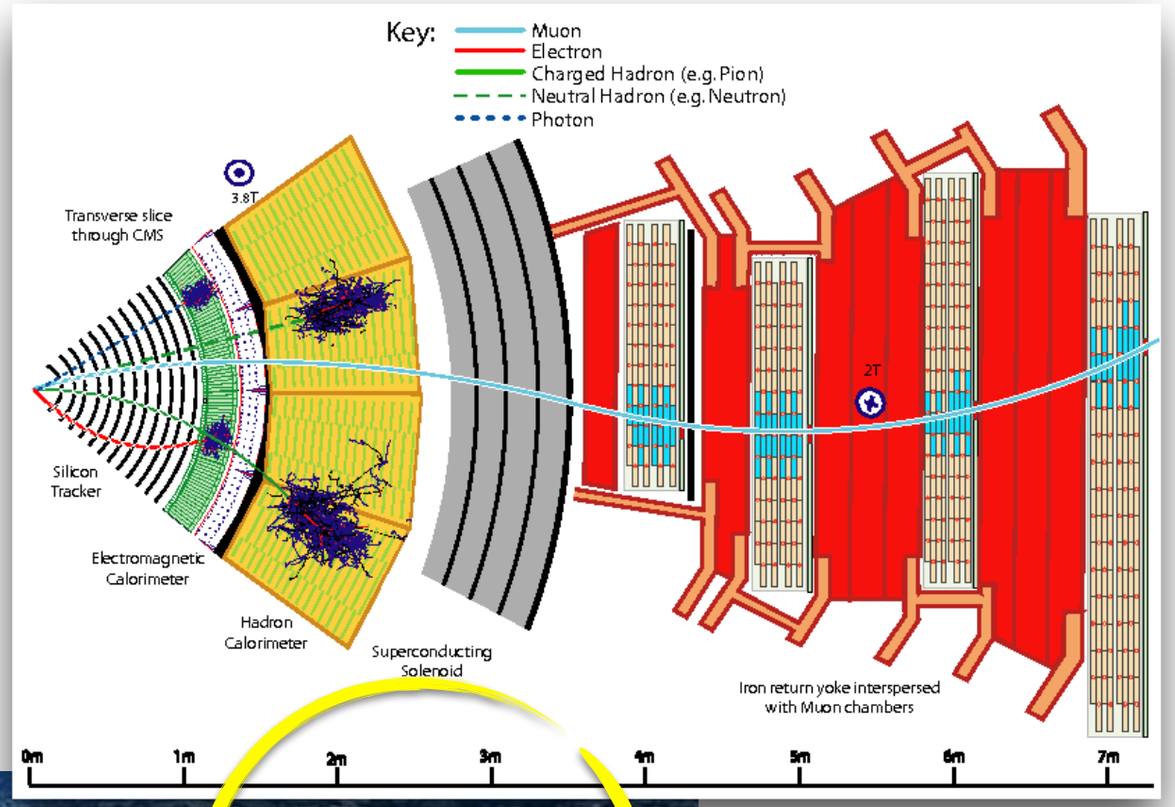
CERN

FastML Lab

Big data @ LHC

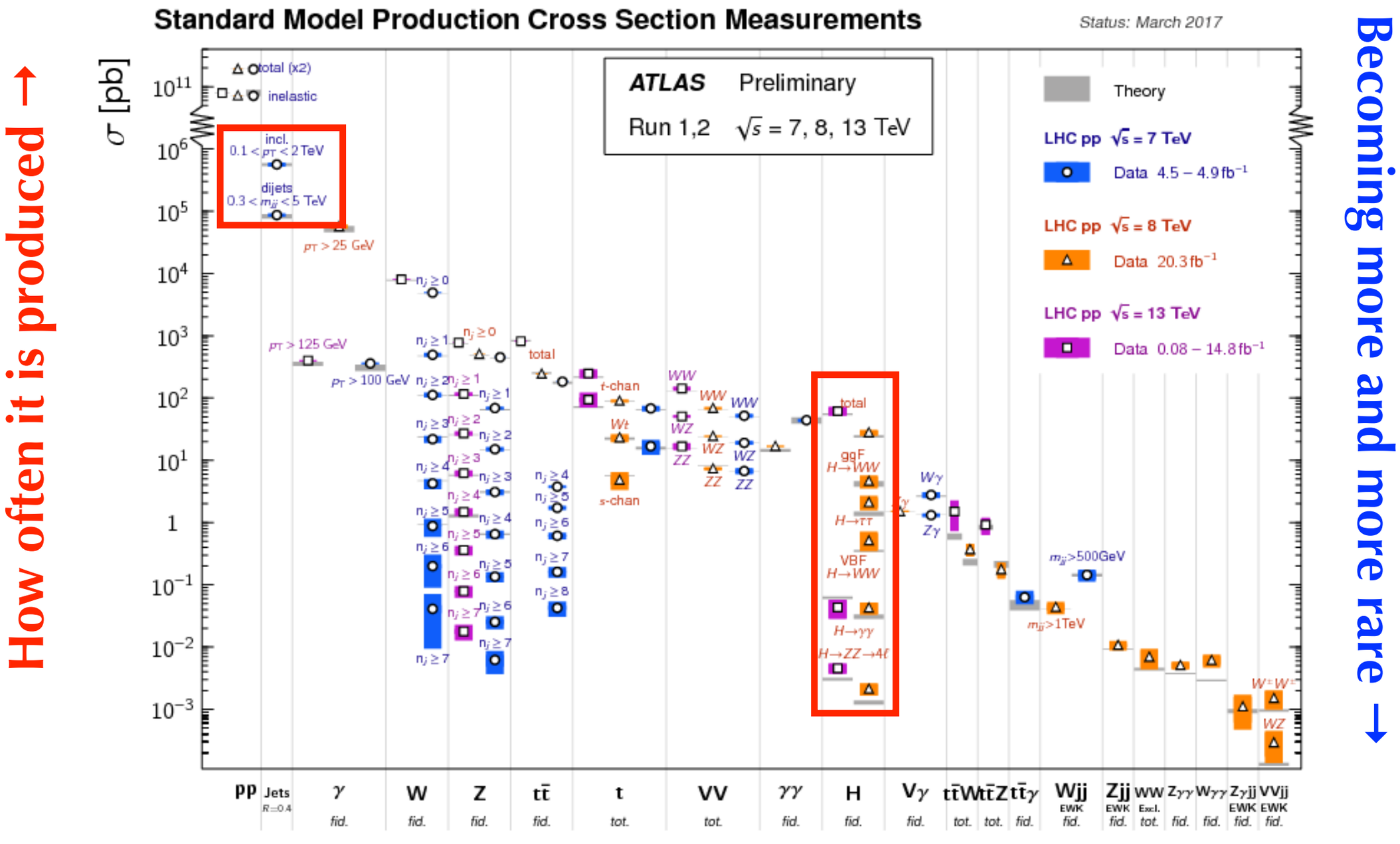
At the LHC the proton beams collide at a frequency of **40 MHz**
Each collision produces **O(10³) particles**
The detectors have **O(10⁸) sensors** used to detect these particles
Extreme **data rates of O(100 TB/s)!**

ex, Compact Muon Solenoid



Collisions which produce interesting products (ex: Higgs boson) are typically very rare

The probability of producing a Higgs boson is 5-9 orders of magnitude smaller than producing only jets



Data reduction workflow @ LHC

99.75% events rejected!

Hardware based
Runs on FPGAs in real time
O(μs) latency

99% events rejected!

Software based
Runs on CPUs in real time
O(100 ms) latency

Software based
Run on data centres (LHC grid)
O(s) computing time

No data stored before this line

40 MHz

100 KHz

L1 trigger

1 KHz

1 MB/evt

High-Level trigger

1 KHz

3-30 kB/evt

Offline reconstruction

Run by individuals on the grid

Data volume

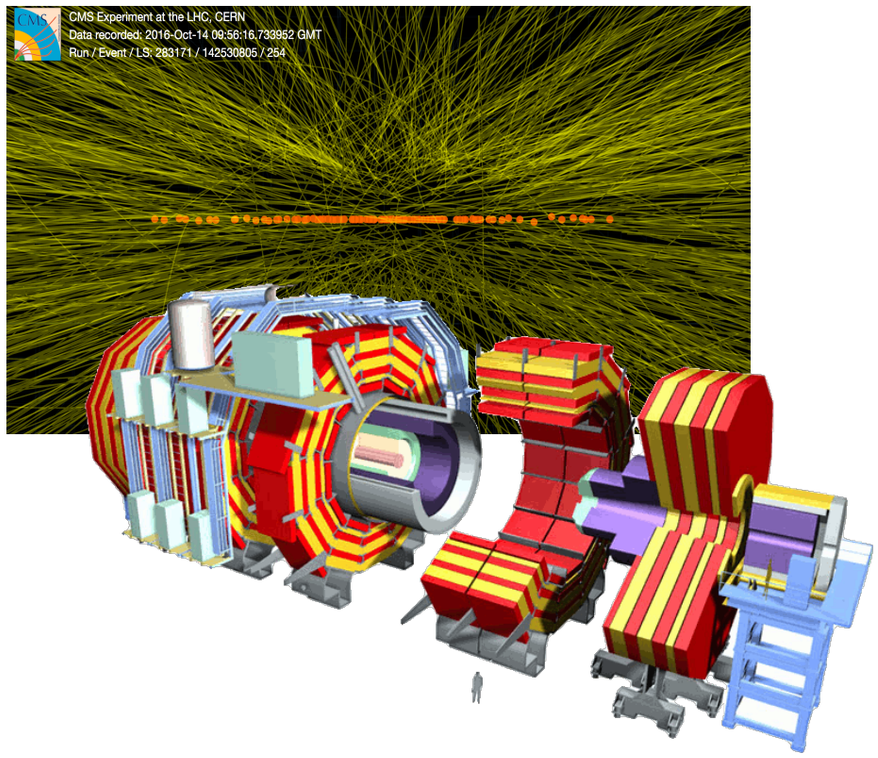
Computing time

Information

High-level analysis

Fast machine learning at the edge

- We must reduce data rates to manageable levels for offline processing and storage by filtering collision events → **triggering**
  - *not all collisions are interesting (ex, about 1 collision per trillion will produce a Higgs boson)*
- We must pick the good ones → **use deep learning for highest accuracy**
- We must do it as fast as one microsecond → **deploy FPGAs**
- Chip area (aka resources) limited as also needed by other physics tasks
- **Example of data processing at the edge** (close to the data source as opposed to cloud)

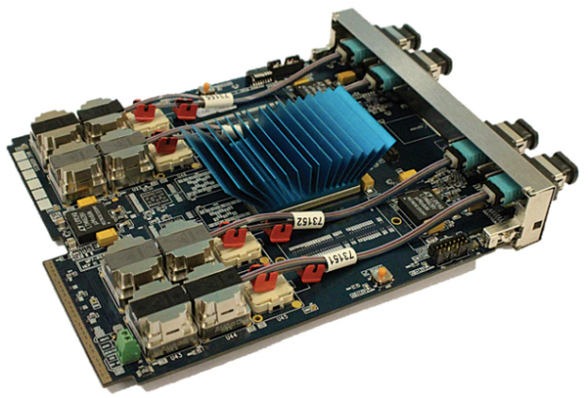


detector front end electronics

detector data

optical links

the trigger system
O(100) FPGAs



in: 40 MHz
out: 100 KHz

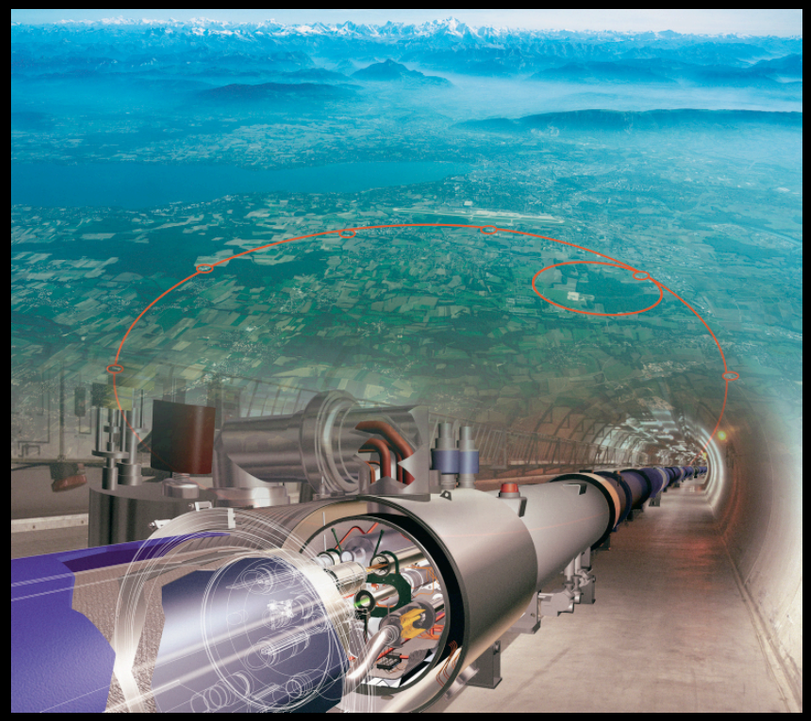
Edge computing: Particle physics vs Every day life

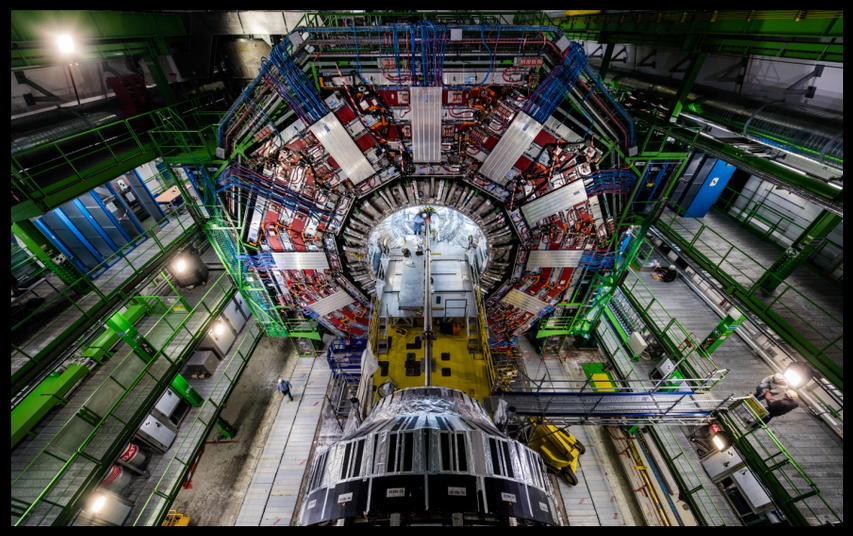
Latency

< O(1) µs

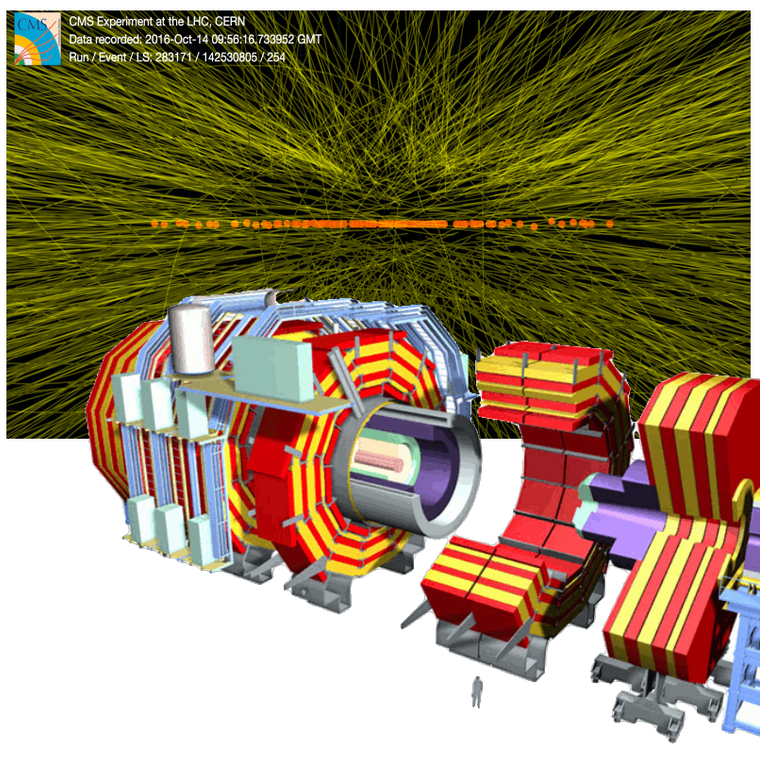
O(10) ms

O(1) s





How to fit DL model here?

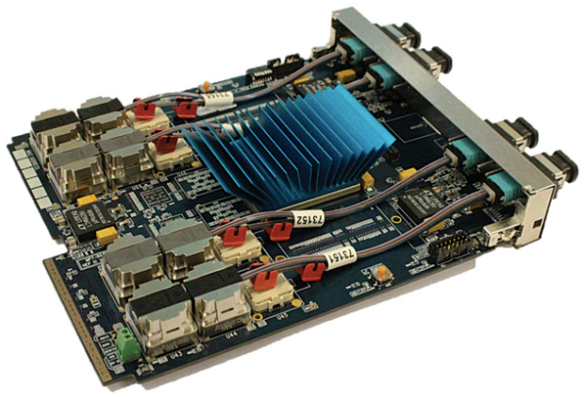


detector front end electronics

detector data

optical links

the trigger system
O(100) FPGAs



in: 40 MHz
out: 100 KHz

Ultra-low latency and low-area with

high level synthesis for machine learning

PYTORCH

Keras
TensorFlow
PyTorch
...

model

compressed model

Usual ML
software workflow

Vivado™ HLS

hls4ml

Co-processing kernel

HLS conversion

HLS project

Custom firmware design

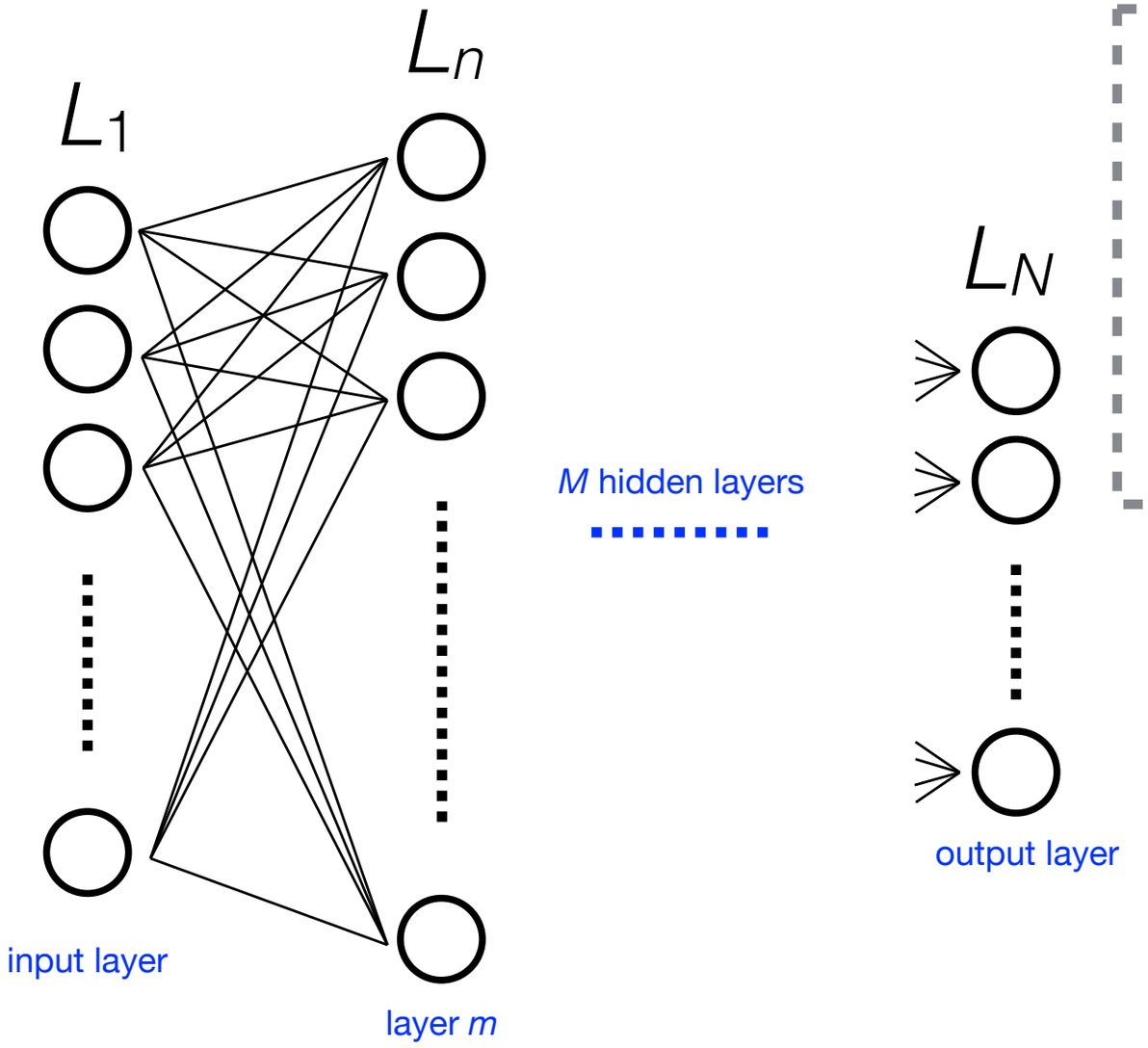
tune configuration
precision
reuse/pipeline

K + TensorFlow

ONNX

https://fastmachinelearning.org/hls4ml/

Neural Network Inference



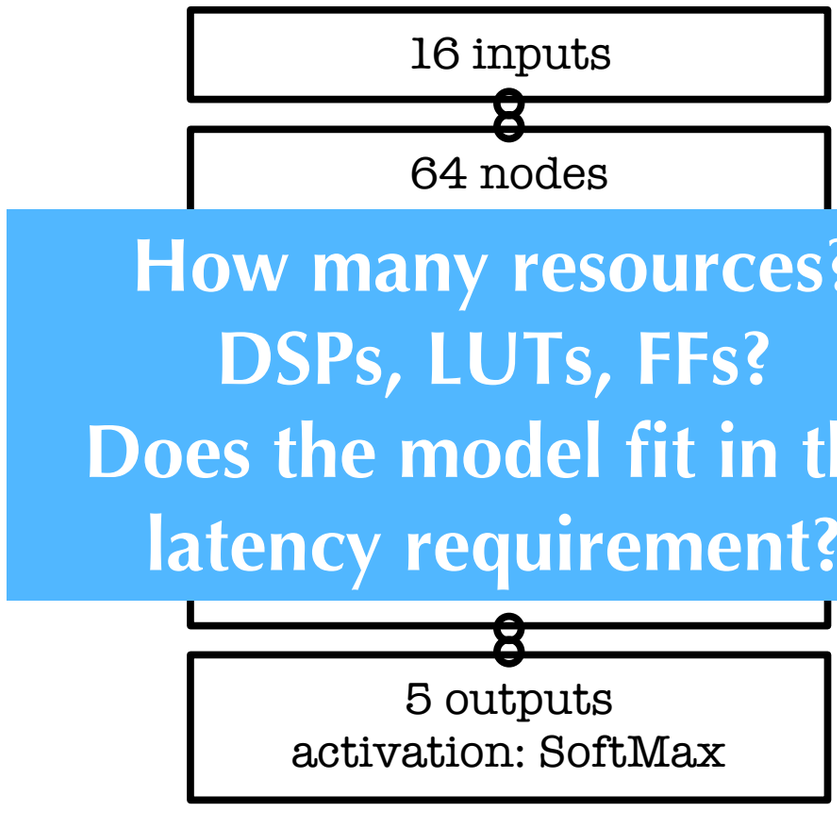
$$\mathbf{x}_n = g_n(\mathbf{W}_{n,n-1}\mathbf{x}_{n-1} + \mathbf{b}_n)$$

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 requirement?

Efficient ML design for FPGAs

FPGAs provide huge flexibility

Performance depends on how well you take advantage of this

Constraints:
Input bandwidth
FPGA resources
Latency

<u>Tricks needed to boost ML inference efficiency:</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

Check out our tutorial with notebooks at:
https://cern.ch/ssummers/hls4ml-tutorial

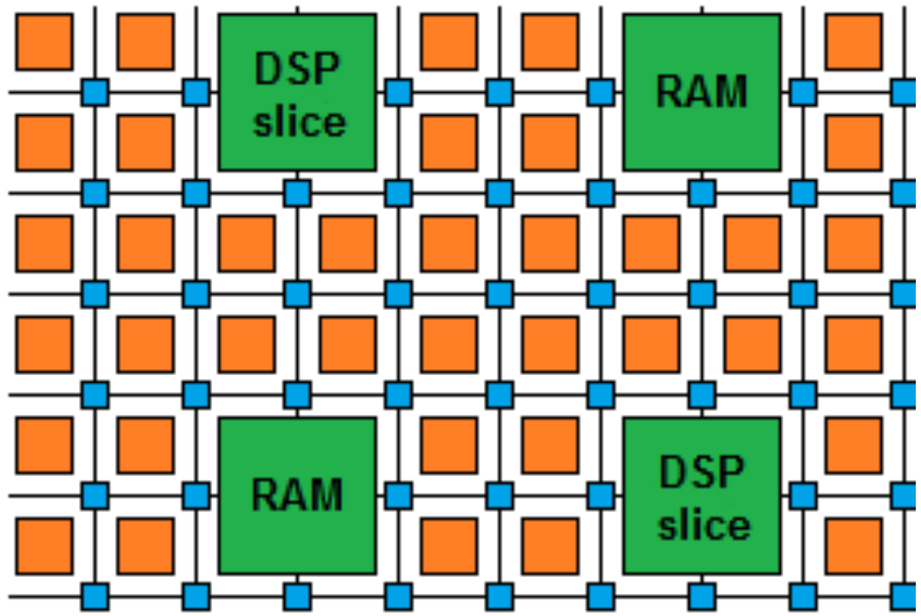
Bonus:
Intro on FPGAs

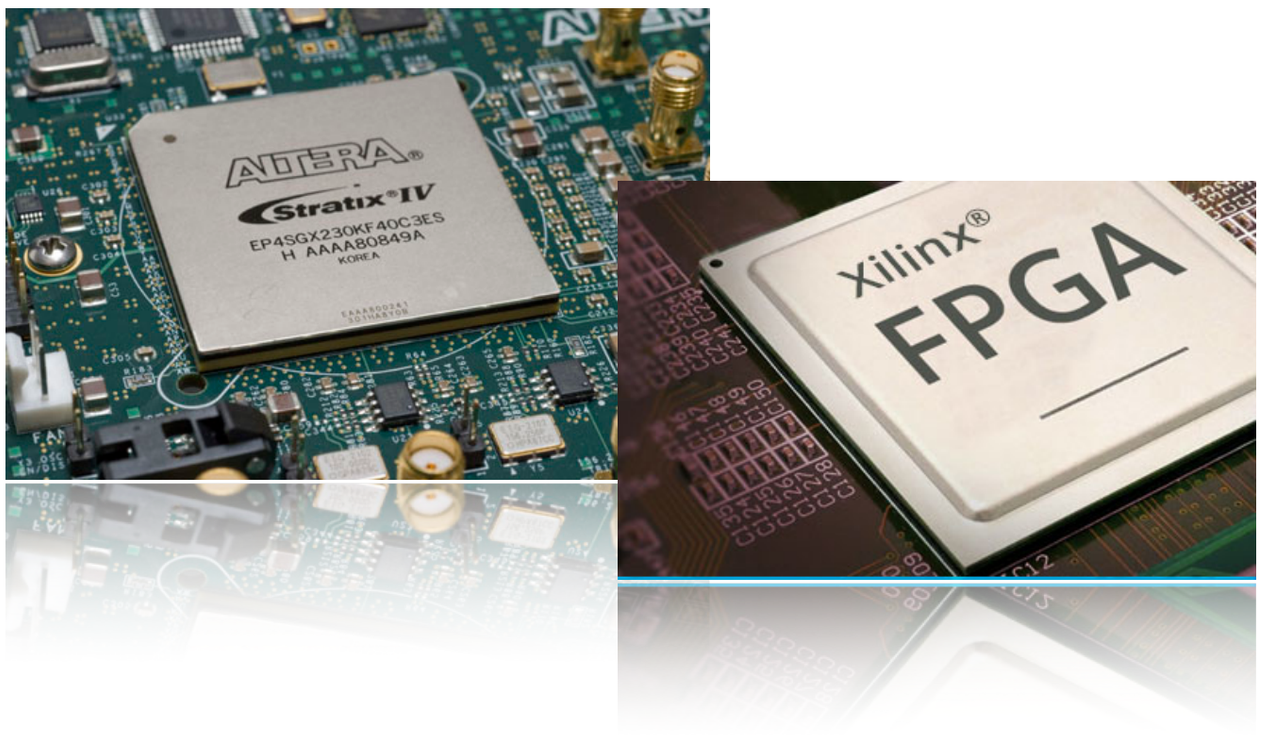
What are FPGAs?

Field Programmable Gate Arrays
are reprogrammable integrated circuits

Contain many different building blocks ('resources') which are connected together as you desire

FPGA diagram



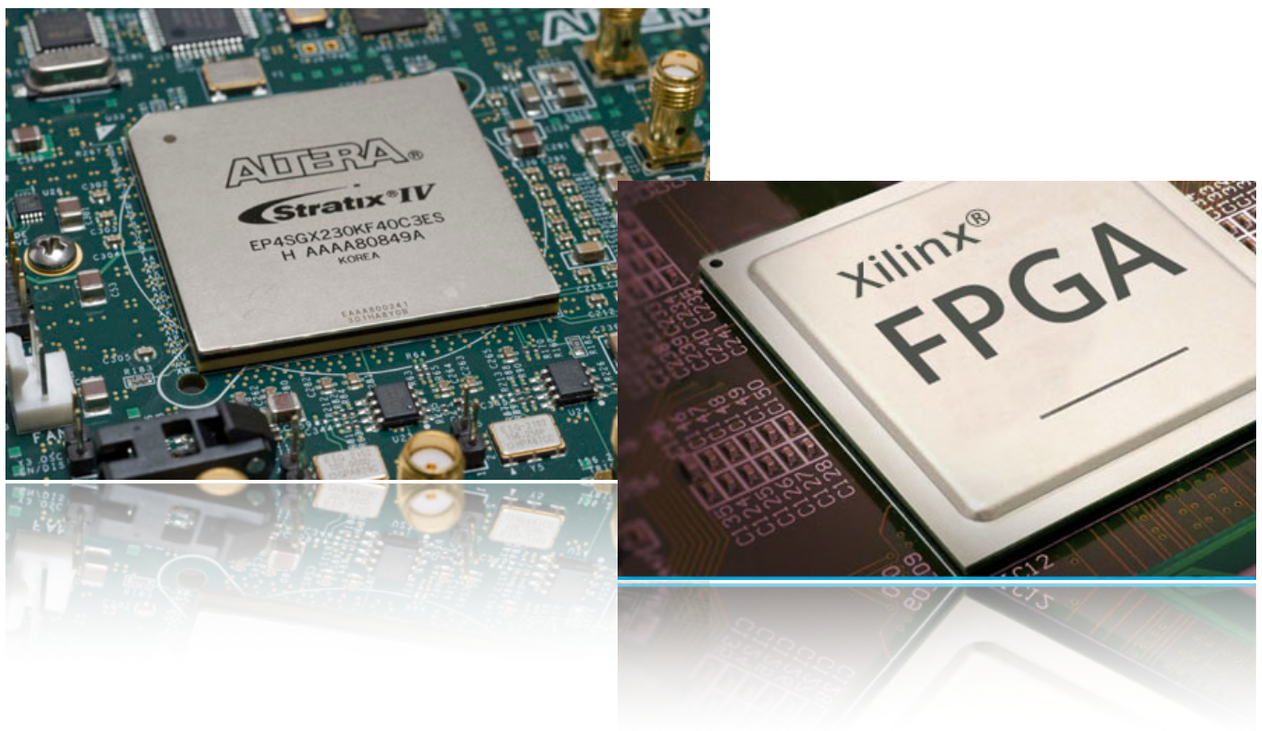


What are FPGAs?

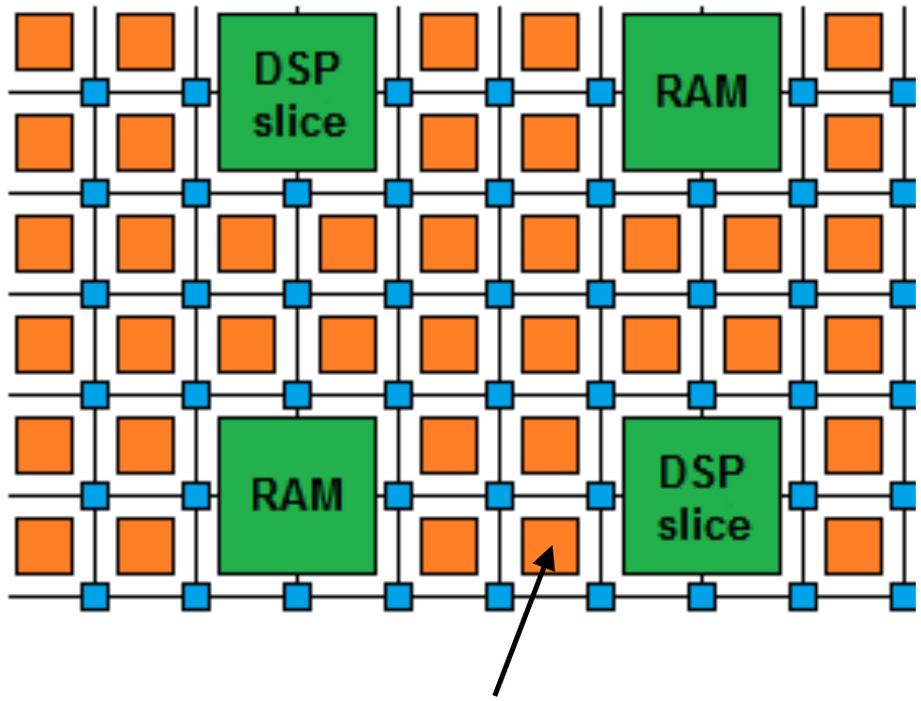
Field Programmable Gate Arrays
are reprogrammable integrated circuits

Look Up Tables (LUTs) perform arbitrary functions on small bitwidth inputs (2-6 bits)
→ used for boolean operations, arithmetics, memory

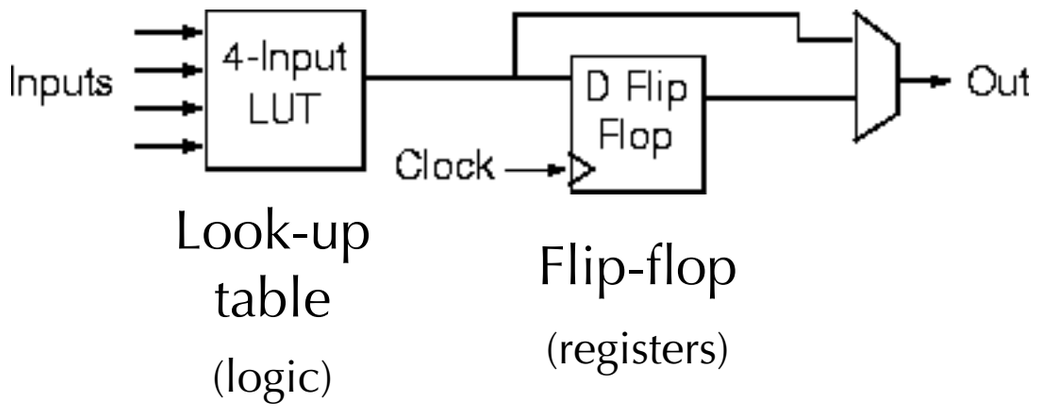
Flip-flops register data in time with the clock pulse



FPGA diagram



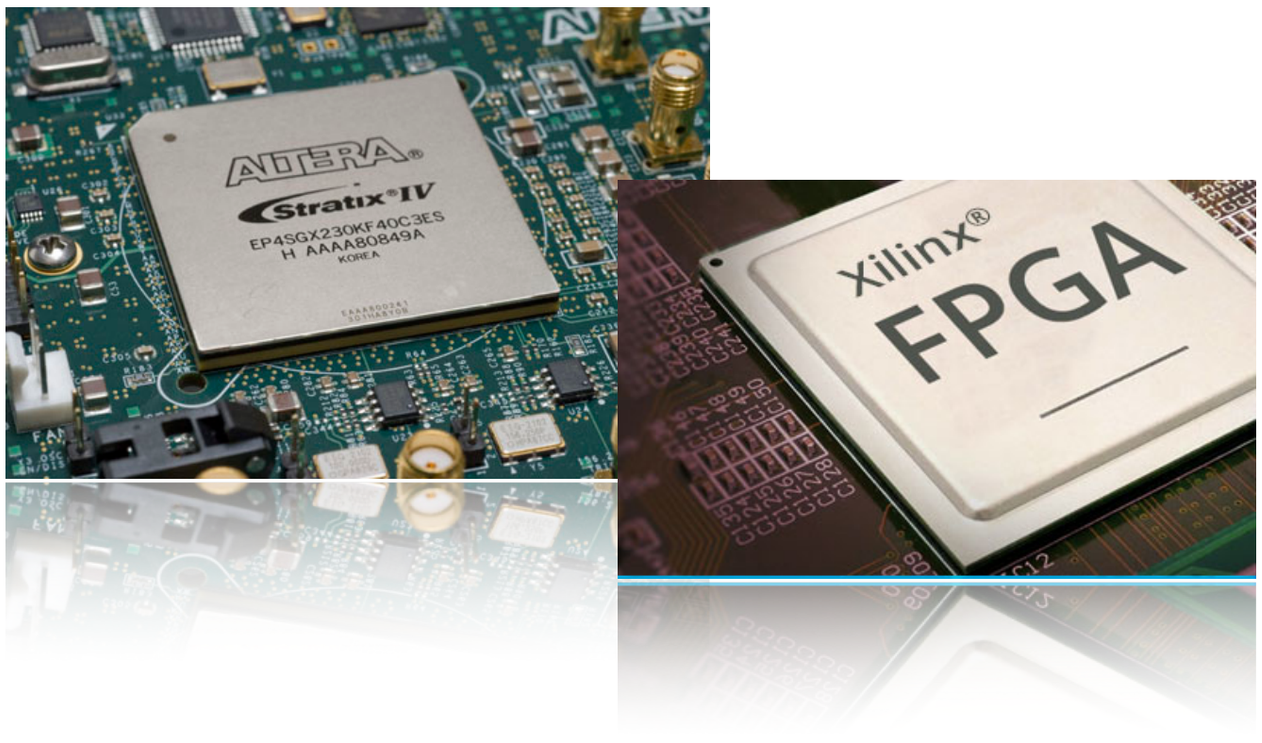
Logic cell



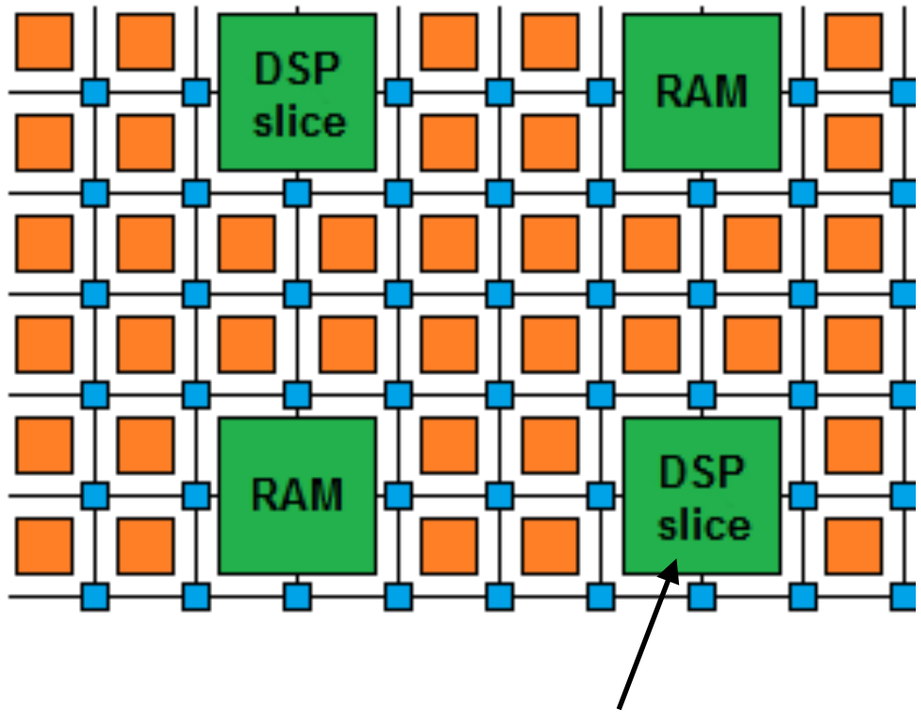
What are FPGAs?

Field Programmable Gate Arrays
are reprogrammable integrated circuits

DSPs are specialized units for multiplication and arithmetic
→ faster and more efficient than LUTs for these type of operations
→ for deep learning, they are often the most precious resource



FPGA diagram



Also contain embedded components:

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

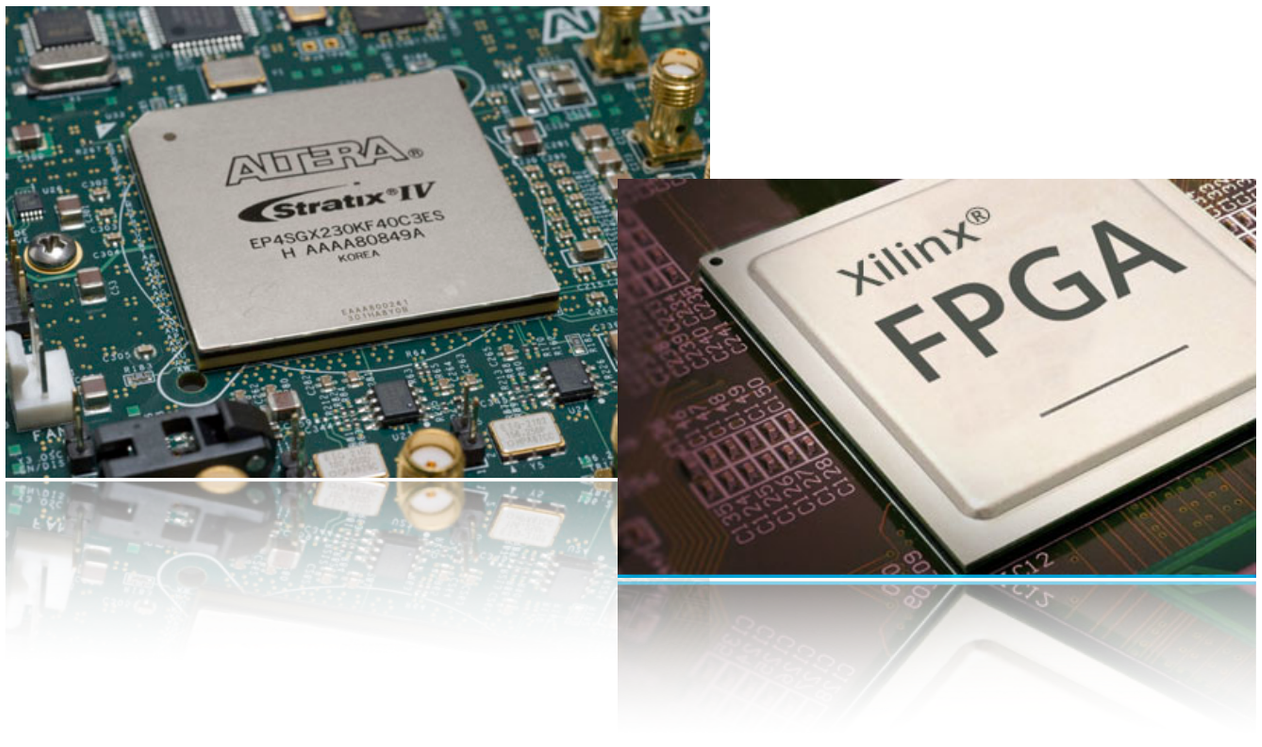
What are FPGAs?

Field Programmable Gate Arrays
are reprogrammable integrated circuits

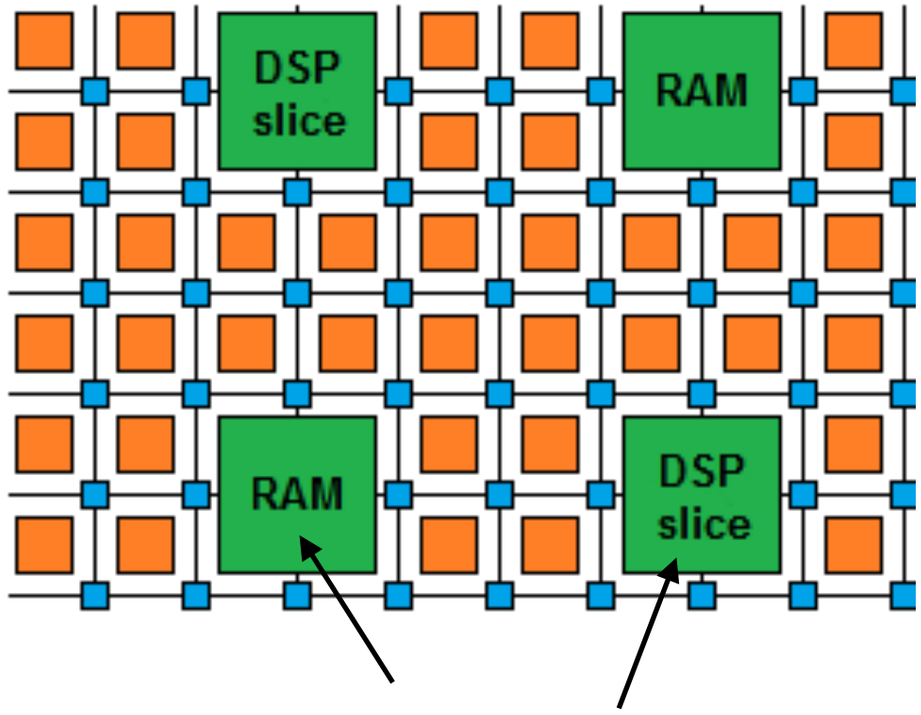
BRAMs are small, fast memories (ex, 18 Kb each)

→ more efficient than LUTs when large memory is required

Modern FPGAs have ~100 Mb of BRAMs, chained together as needed



FPGA diagram



Also contain embedded components:

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

Random-access memories (RAMs): embedded memory elements

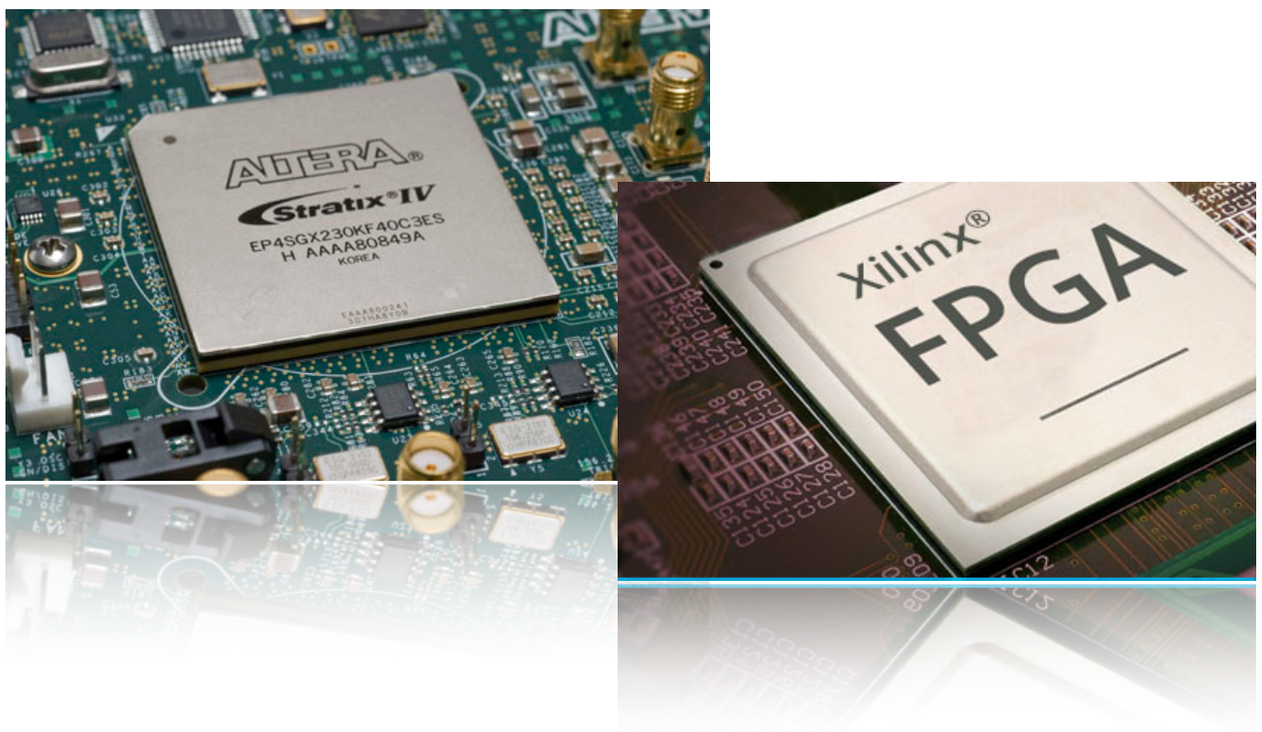
What are FPGAs?

Field Programmable Gate Arrays
are reprogrammable integrated circuits

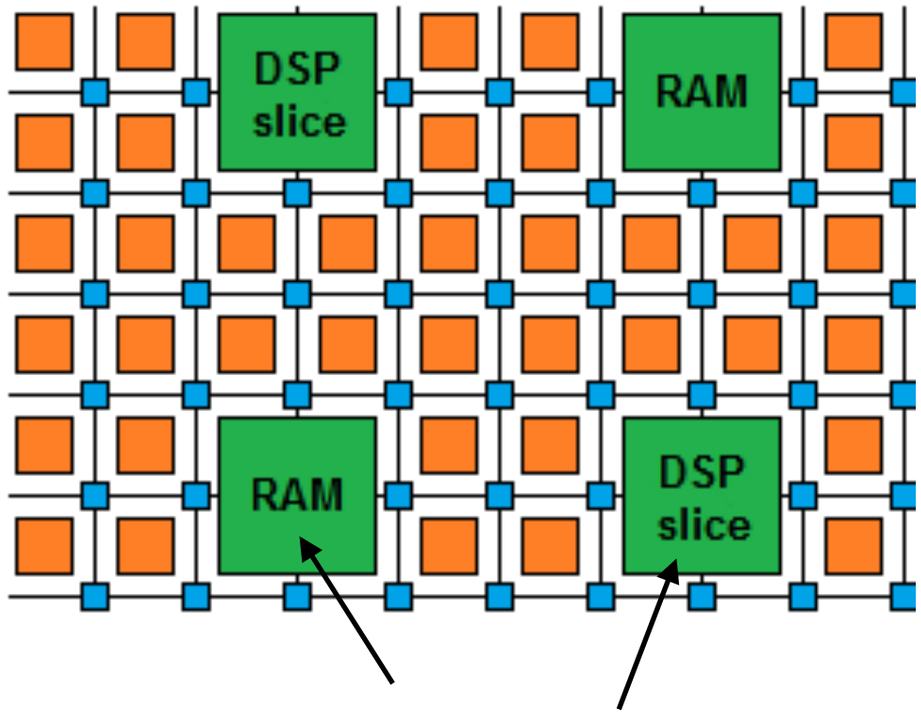
Contain array of **logic cells** embedded with **DSPs**, **BRAMs**, etc.

Support **highly parallel** algorithm implementation

Low power per Op (relative to CPU/GPU)



FPGA diagram



Also contain embedded components:

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

Random-access memories (RAMs): embedded memory elements

Why are FPGAs fast?

- Fine-grained / resource parallelism
  - use the many resources to work on different parts of the problem simultaneously
  - allows us to achieve **low latency**
- Most problems have at least some sequential aspect, limiting how low latency we can go
  - but we can still take advantage of it with…
- **Pipeline parallelism**
  - instruct the FPGA to work on different data simultaneously
  - allows us to achieve **high throughput**

Like a production line for data…

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!

See Xilinx Vivado HLS, Intel HLS, Catapult HLS