Jefferson Lab
Thomas Jefferson National Accelerator Facility

Machine Learning on FPGA

Sergey Furletov

Jefferson Lab

Joint GlueX-EIC-PANDA Machine Learning Workshop

25 Sep 2020

Outline

- Motivation
  - Examples from HEP experiments at LHC
  - EIC experiment
- ANN in FPGA (*)
- ML in FPGA for physicists
  - Experimental setup
  - Offline PID analysis with ML (root / TMVA)
  - Moving on to FPGA
    - Choosing FPGA for ML
  - Run ML on FPGA
- Optimization ML in FPGA with hls4ml package
- Global PID with multiple PID detectors
- Hardware solutions
- Outlook
- Live demonstration of the workflow (optional)

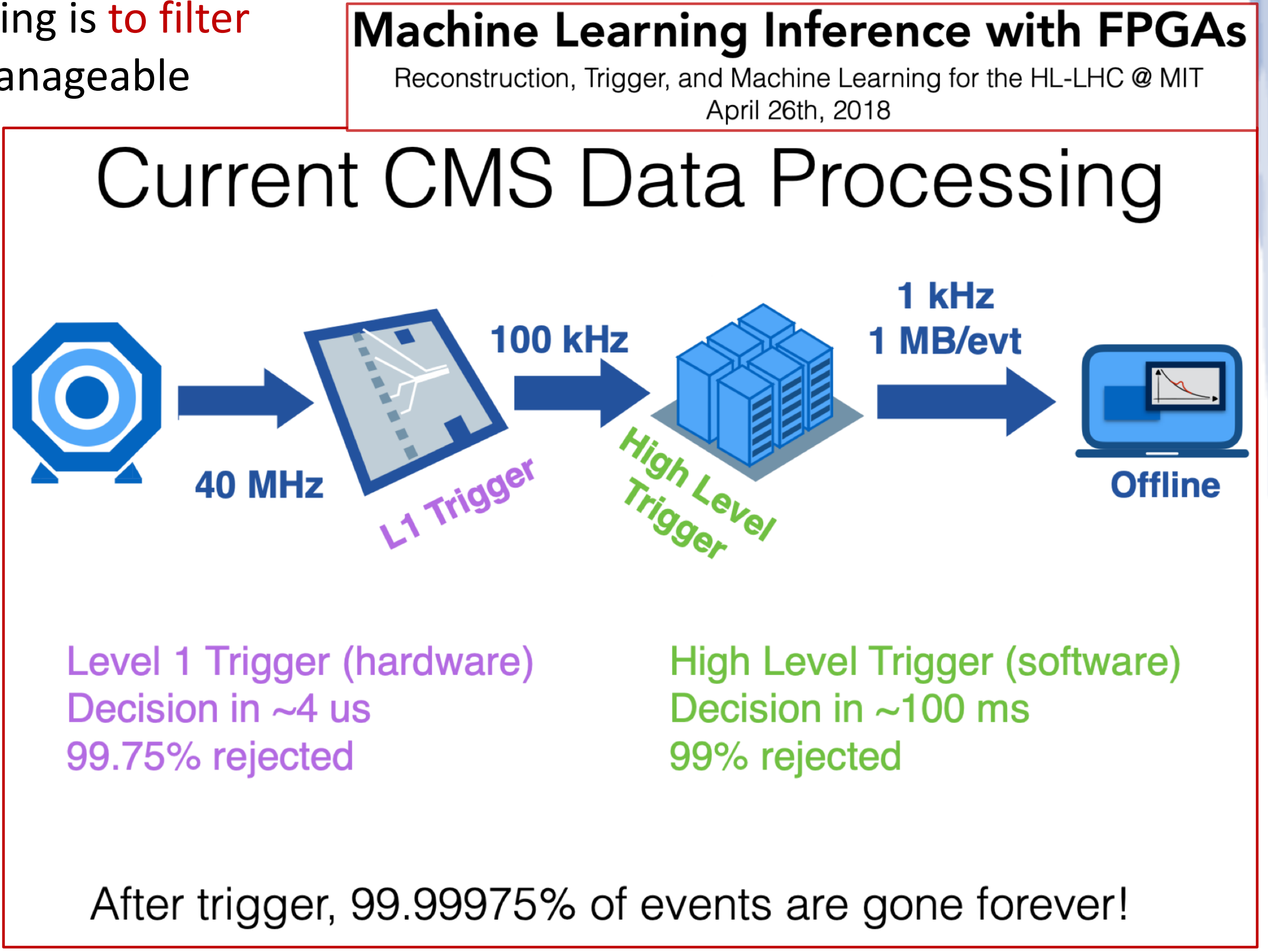
(*) Field Programmable Gate Array

Motivation

- *With increase of luminosity for accelerator colliders as well as a granularity of detectors for particle physics, more challenges fall on the readout system and data transfer from detector front-end to computer farm and long term storage.*
- *At the LHC, data rates at the CMS and ATLAS , are of the order of terabytes per second*
- *Modern (triggered) data acquisition systems (LHC, KEK, Fair) employ several stages for data reduction.*
- *Concepts of trigger-less readout and data streaming will produce large data volumes being read from the detectors.*
- *From a resource standpoint, it makes much more sense to perform data pre-processing and reduction at early stages of data streaming. Would allow to use information-rich data sets for event selection.*
- *Our project mostly inspired by work carried out at CERN , and progress in ML application on FPGA*

Rates example from CMS

- CMS bandwidth: Phase-2 ~50 Tb/s (1.8TB/s in Phase-1)
- The task of the real-time processing is to filter events to reduce data rates to manageable levels for offline processing.
- Level-1 typically uses custom hardware with ASICs or FPGAs (decision ~4 μs)
- The second stage of triggering, High Level Trigger (HLT), uses commercial CPUs to process the filtered data in software. (decision ~100 000 μs)



Machine Learning Inference with FPGAs

Reconstruction, Trigger, and Machine Learning for the HL-LHC @ MIT
April 26th, 2018

Current CMS Data Processing

Level 1 Trigger (hardware)
Decision in ~4 us
99.75% rejected

High Level Trigger (software)
Decision in ~100 ms
99% rejected

After trigger, 99.99975% of events are gone forever!

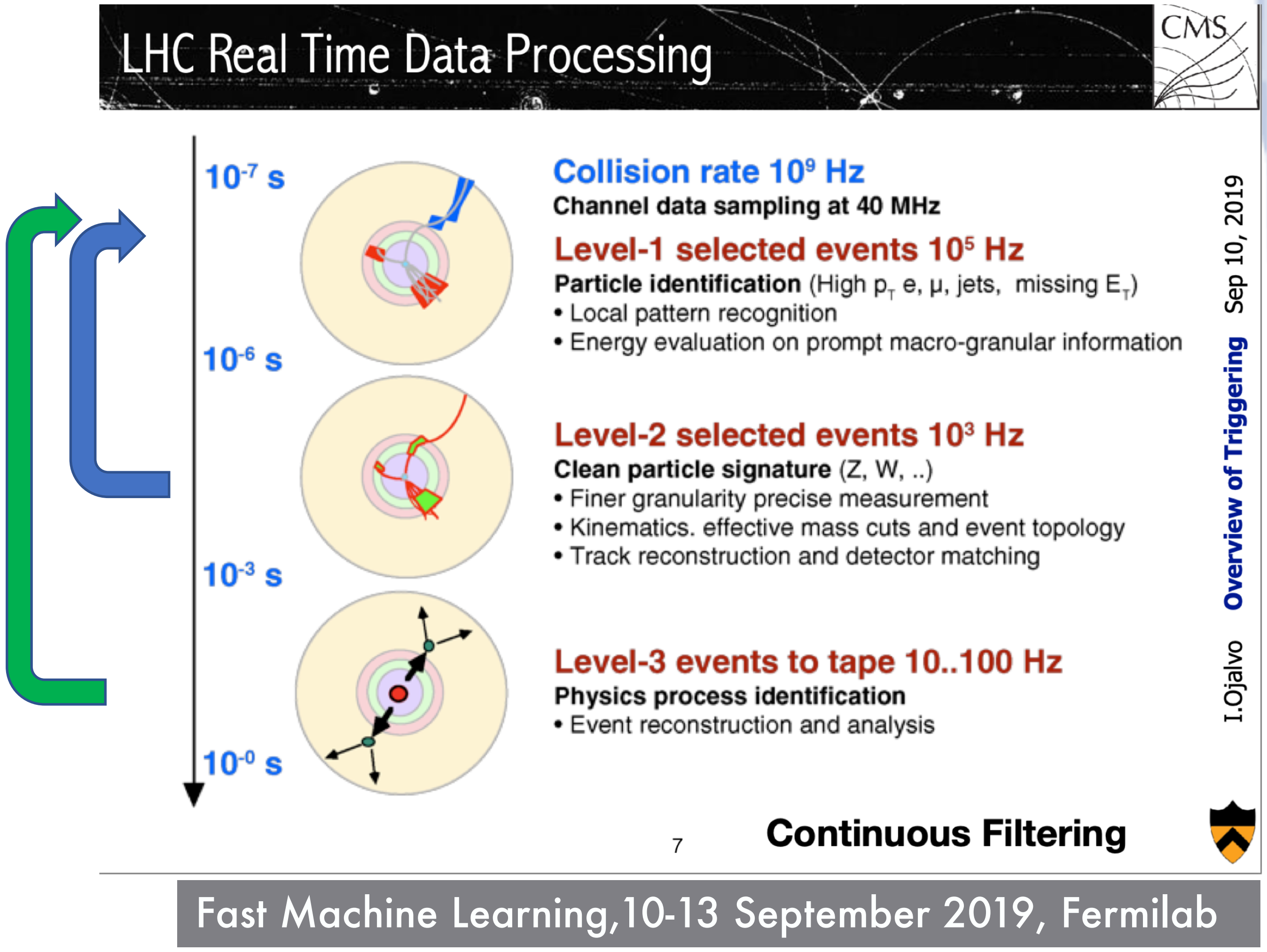
Rejection is mostly defined by cross section of interesting physics processes and trigger efficiency

Motivation ML on FPGA

- *The growing computational power of modern FPGA boards allows us to add more sophisticated algorithms for real time data processing.*
- *Many tasks could be solved using modern Machine Learning (ML) algorithms which are naturally suited for FPGA architectures.*

Level 1 works with Regional and sub-detector Trigger primitives

Using ML on FPGA many tasks from Level 2 and/or Level 3 can be performed at Level 1



LHC Real Time Data Processing

CMS

Collision rate 10^9 Hz
Channel data sampling at 40 MHz

Level-1 selected events 10^5 Hz
Particle identification (High p_T e, μ, jets, missing E_T)
- Local pattern recognition
- Energy evaluation on prompt macro-granular information

Level-2 selected events 10^3 Hz
Clean particle signature (Z, W, ..)
- Finer granularity precise measurement
- Kinematics. effective mass cuts and event topology
- Track reconstruction and detector matching

Level-3 events to tape 10..100 Hz
Physics process identification
- Event reconstruction and analysis

Continuous Filtering

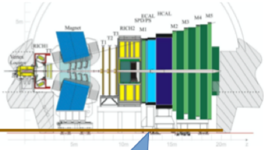
7

I.Ojalvo **Overview of Triggering** Sep 10, 2019

Fast Machine Learning,10-13 September 2019, Fermilab

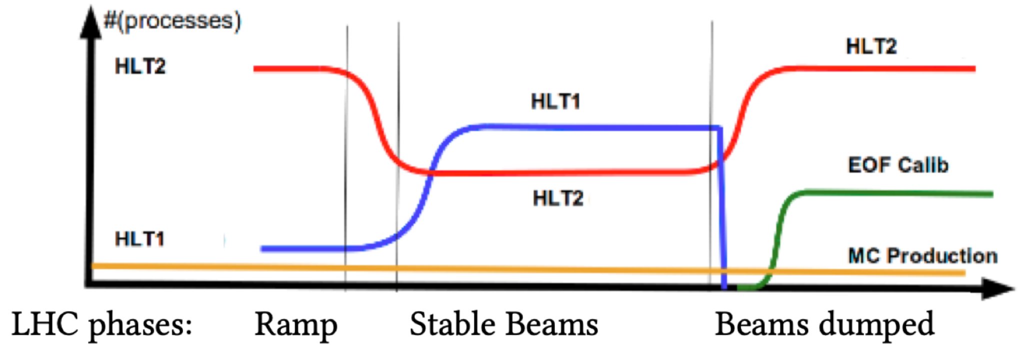
Example from LHCb Run2

LHCb event filter farm



- Event filter farm
  - 1600 nodes, up to 50000 concurrent processes
  - 12 PB disk storage
- High Level Trigger split in two stages:
  - HLT1 filters L0 output in real-time (partial event reconstruction)
  - Output buffered to local disks
  - HLT2 processes events in and out of fill (full event reconstruction)
- Optimal use of farm resources

Implemented for LHC Run 2



S. Stahl, 22.3.18 Real-time analysis with the LHCb trigger 6

ML on Xilinx FPGA

- *While the large numerical processing capability of GPUs is attractive, these technologies are optimized for high throughput, not low latency.*
- *FPGA-based trigger and data acquisition systems have extremely low, sub-microsecond latency requirements that are unique to particle physics.*
- *Machine learning methods are widely used and have proven to be very powerful in particle physics.*
- *However, exploration of the use of such techniques in low-latency FPGA hardware has only just begun.*

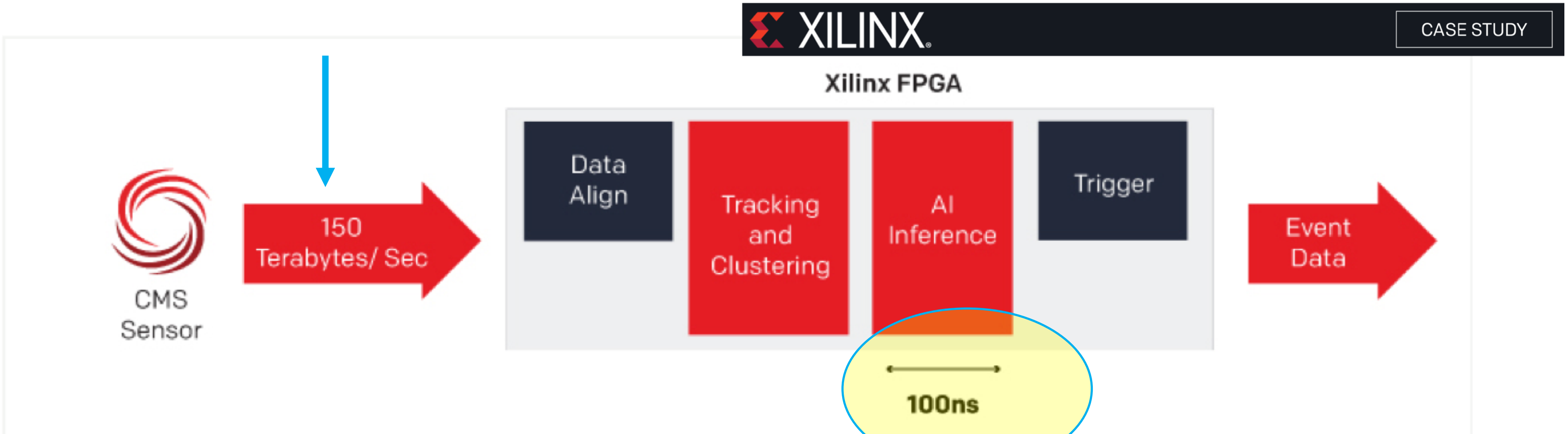


Figure 2: Compared to alternative devices such as GPUs and ASICs, FPGAs are the only viable choice for the event trigger processing because they provide extremely low latency. While the large numerical processing capability of GPUs is attractive, these technologies are optimized for high throughput, not low latency.

https://www.xilinx.com/publications/powered-by-xilinx/cerncasestudy-final.pdf

EIC rates estimation

Jefferson Lab
Thomas Jefferson National Accelerator Facility

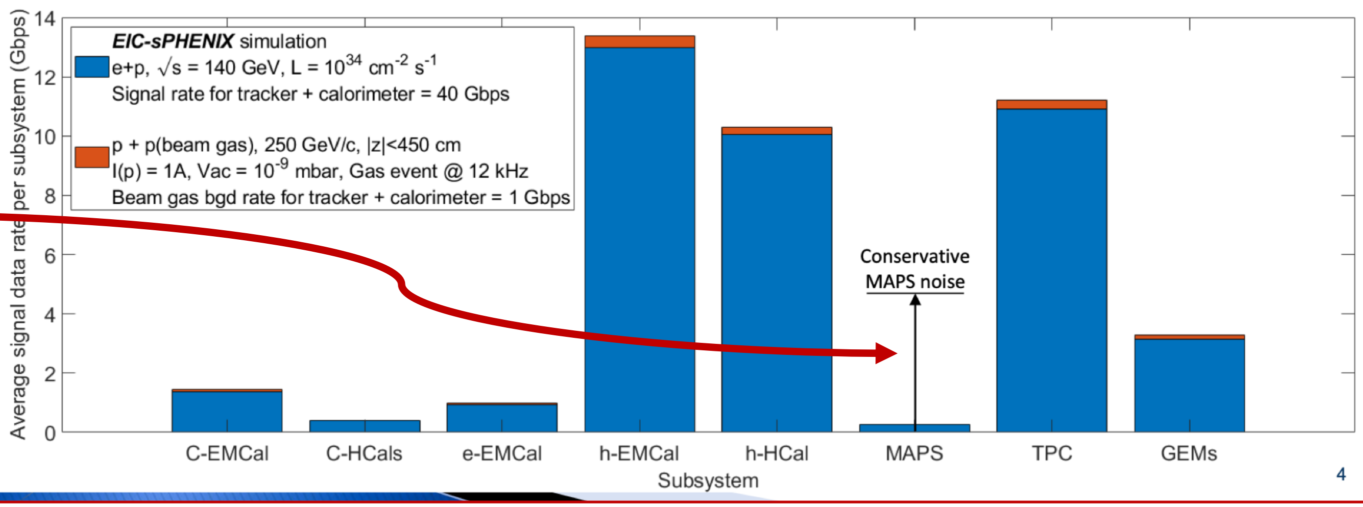
Jin Huang <jhuang@bnl.gov> YR kick-off meeting

- EIC moves to the concept of streaming readout.
- Need a large computer farm to handle streaming data.
- In terms of resources, it makes sense to perform data pre-processing and reduction at the early stages of data streaming.

Signal data rate -> DAQ strategy

Note sPH-cQCD-2018-001: https://indico.bnl.gov/event/5283/ , Simulation: https://eic-detector.github.io/

- What we want to record: total collision signal ~ 100 Gbps @ 10^{34} cm^{-2} s^{-1}
  - Assumption: sPHENIX data format, 100% noise
  - Less than sPHENIX peak disk rate. 10^{-4} comparing to LHC collision
- Therefore, we could choose to stream out all EIC collisions data
- In addition, DAQ may need to filter out excessive beam background and electronics noise, if they become dominant.
  - Very different from LHC, where it is necessary to filter out uninteresting p+p collisions (CMS/ATLAS/LHCb) or highly compress collision data (ALICE)
  - Such filtering does not require real-time event reconstruction

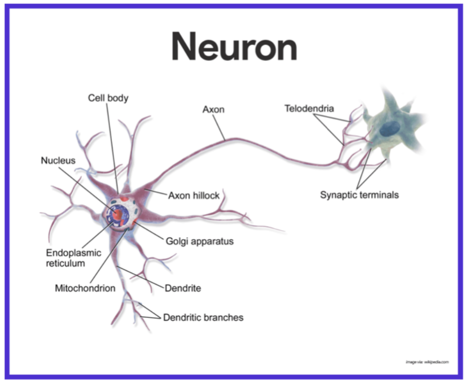


Belle II , 8 megapixels PXD produces ~200 Gbps @ 30 kHz trigger rate. (beam background, noise and synchrotron)

Neural network and FPGA

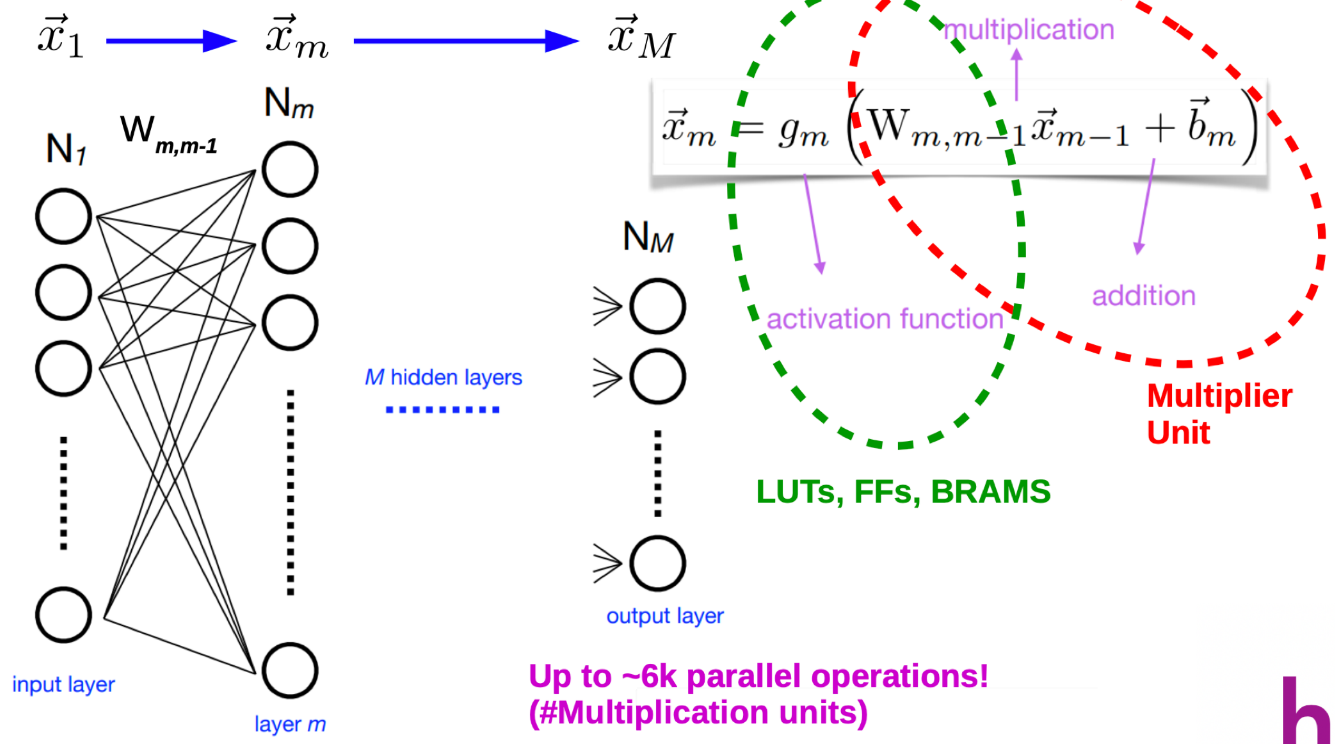
Artificial Neural Network

Image: https://nurseslabs.com/nervous-system/



Inference on an FPGA

Every clock cycle
(all layer operations can be performed simultaneously)

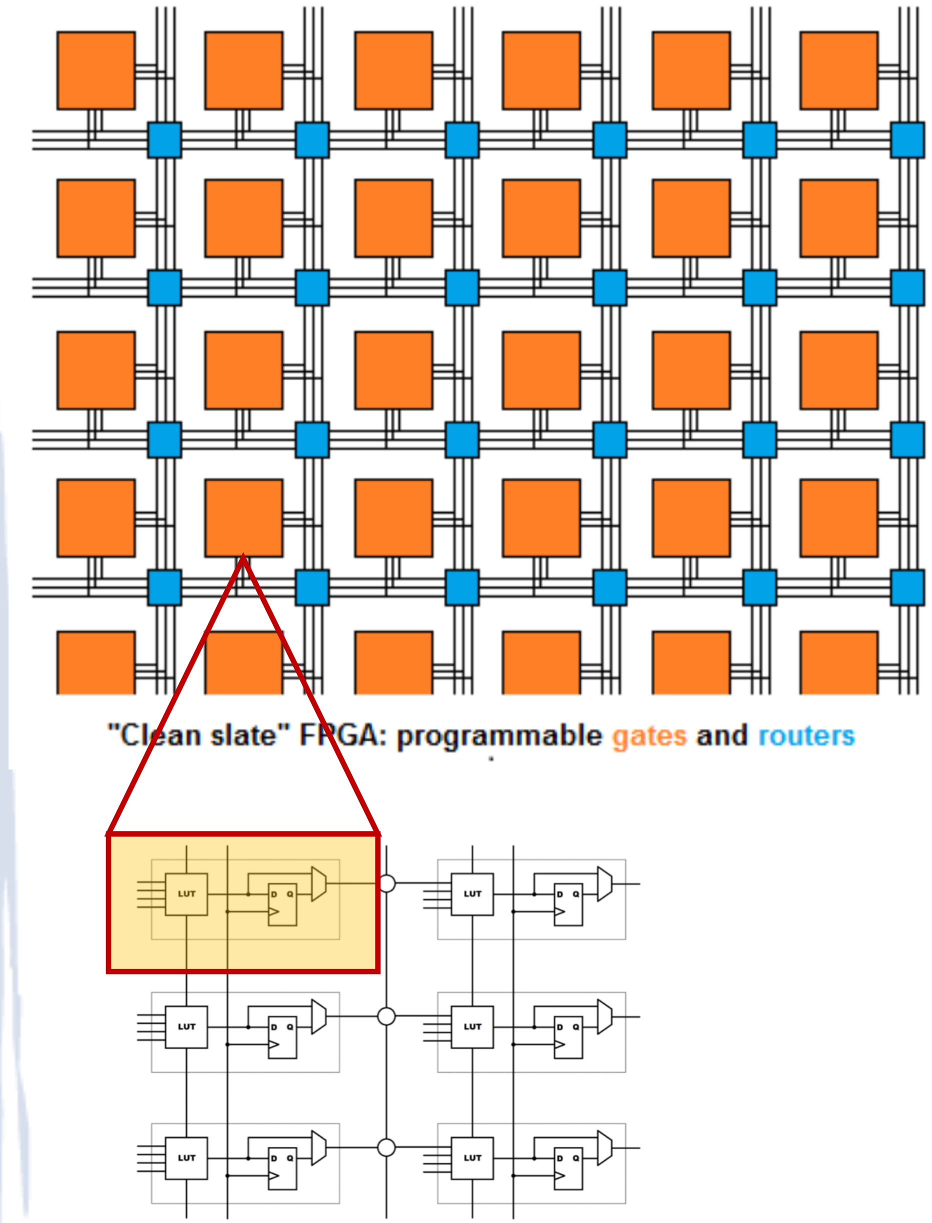


$$\vec{x}_m = g_m\left(W_{m,m-1}\vec{x}_{m-1} + \vec{b}_m\right)$$

Up to ~6k parallel operations!
(#Multiplication units)

IRIS-HEP th Febraury 13 , 2019 Dylan Rankin [MIT]

FPGA structure



- Fist FPGA have only programmable gates and routers: Field Programmable Gate Array .
- It can perform logical operation in parallel using LUTs and FFs.
- There are problems with the math operation required by the neural network.

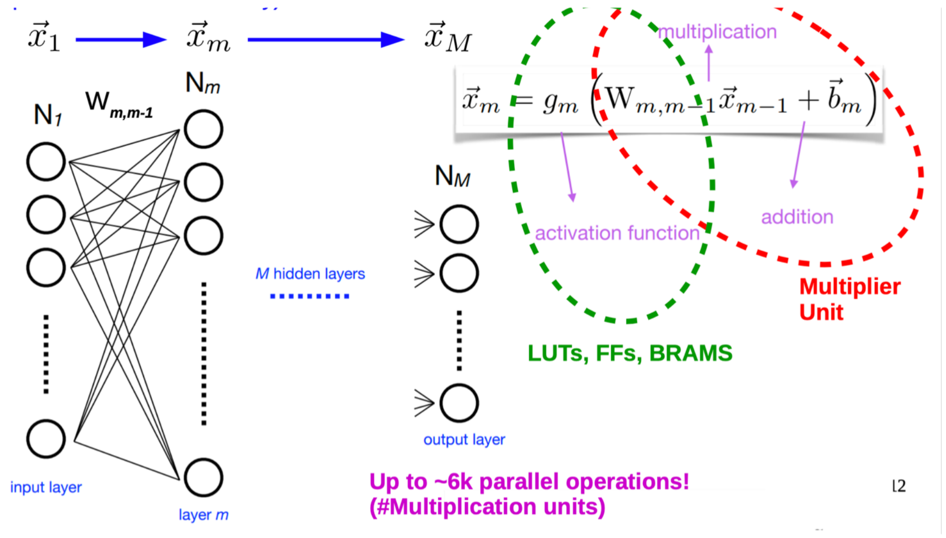
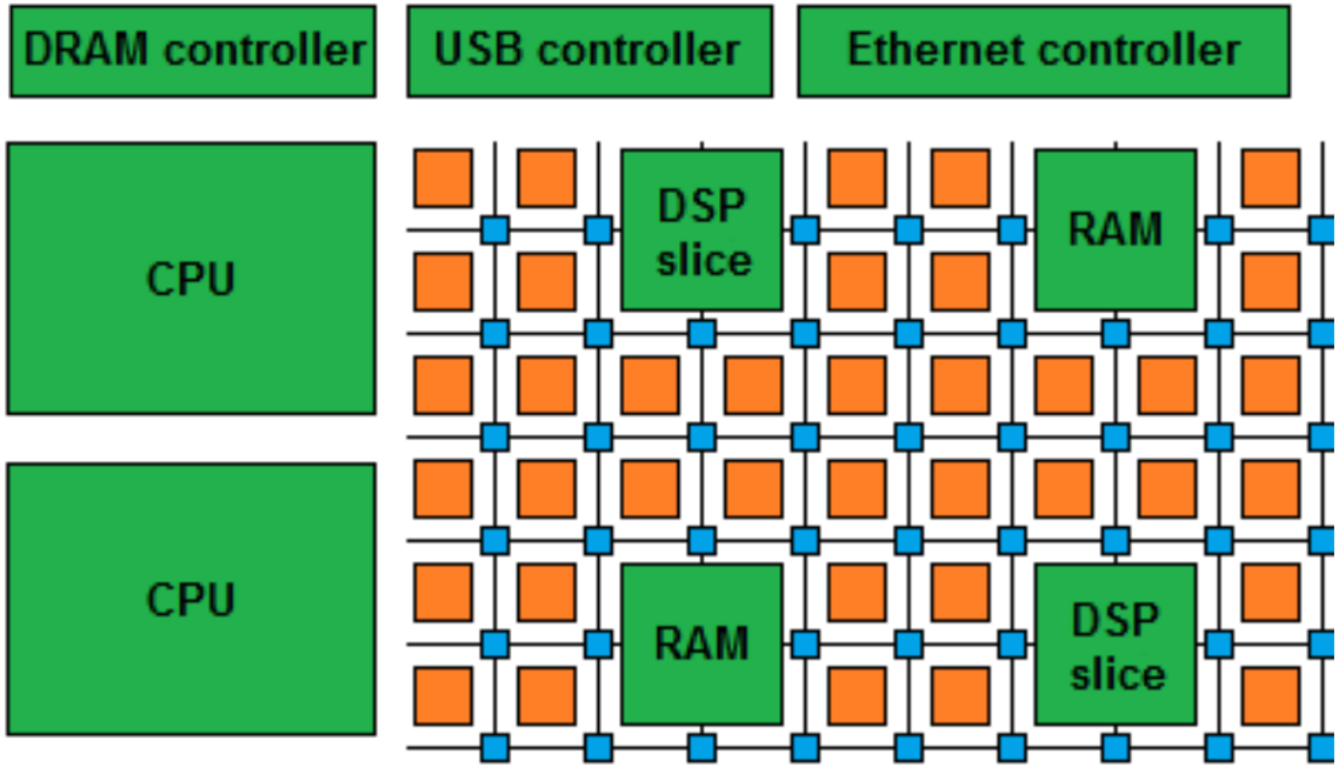


Image from: https://www.embeddedrelated.com/showarticle/195.php

Modern FPGA

- Modern FPGAs have DSP slices - specialized hardware blocks placed between gateways and routers that perform mathematical calculations.
- The number of DSP slices can be up to 6000-12000 per chip.
- In addition, they often have ARM cores implemented using non-programmable gates, or “soft processor core” MicroBlaze.



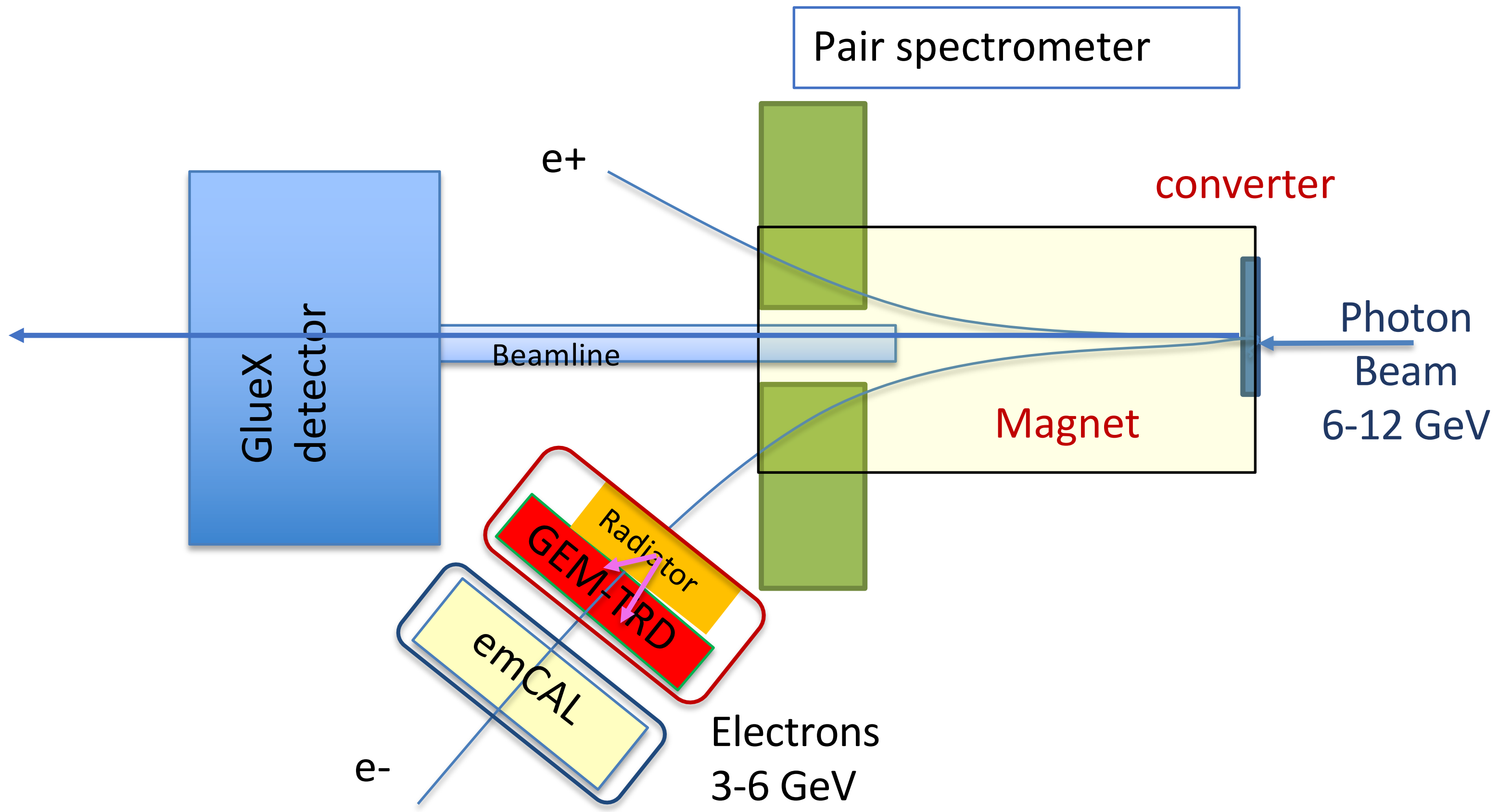
Modern FPGA: lots of hard, not-field-programmable gates

Image from: https://www.embeddedrelated.com/showarticle/195.php

Experimental setup and workflow

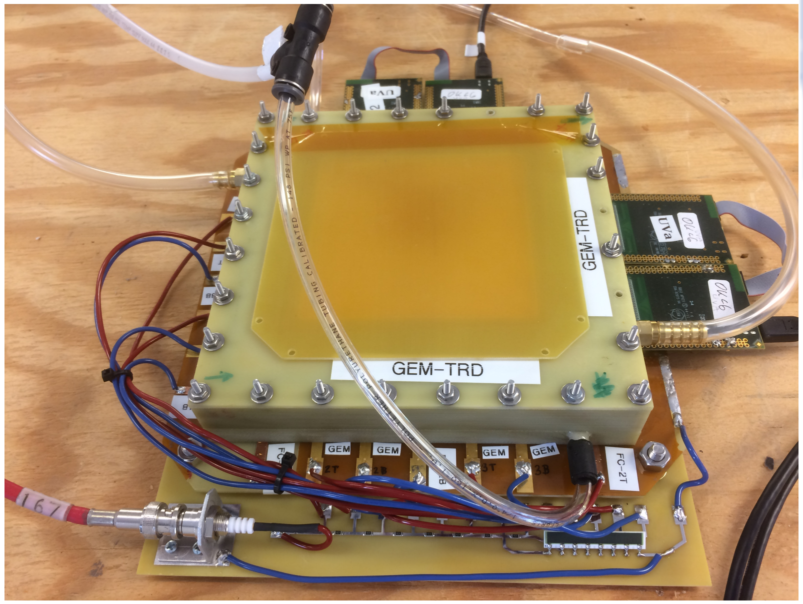
Beam setup at JLab Hall-D

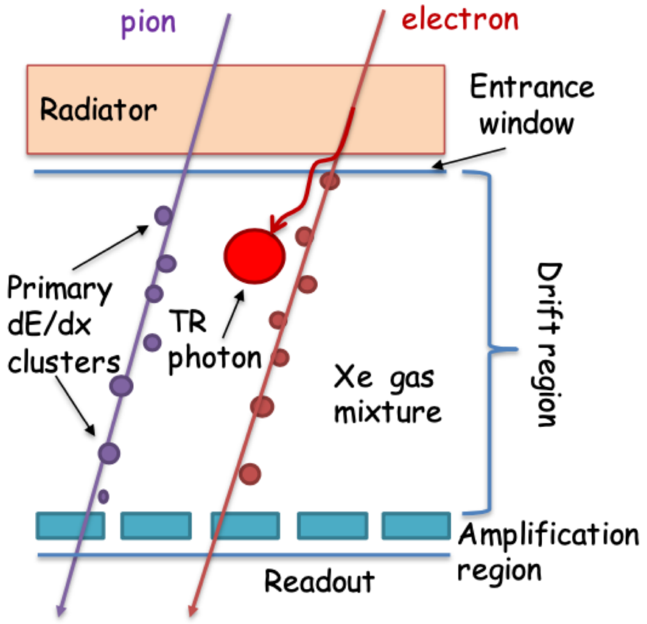
- *Tests were carried out using electrons with an energy of 3-6 GeV, produced in the converter of a pair spectrometer at the upstream of GlueX detector.*

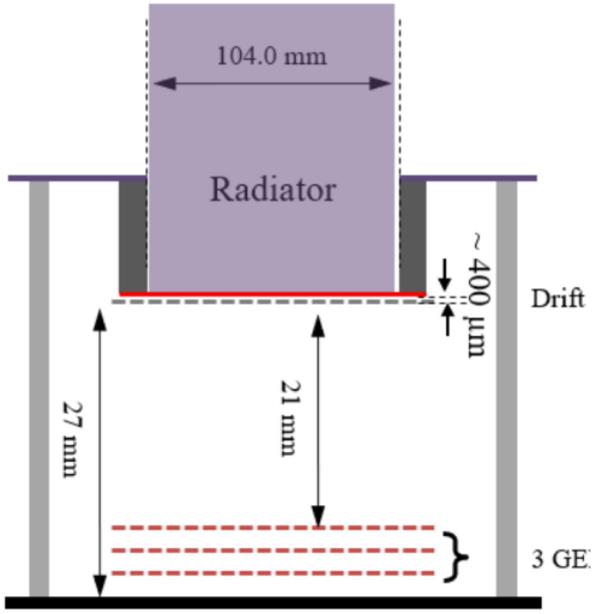


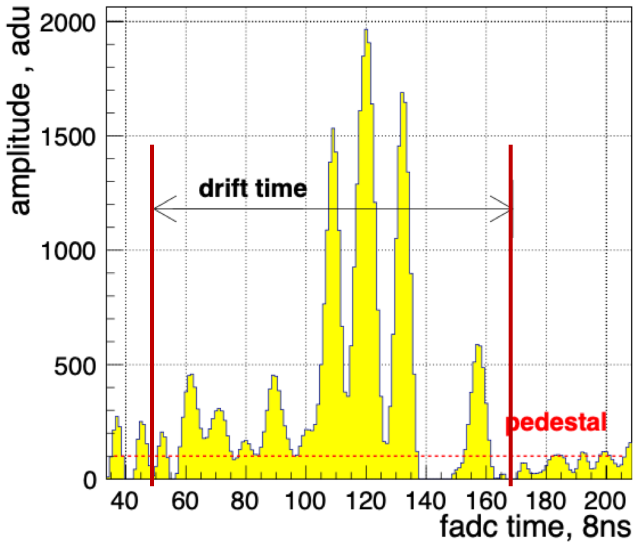
GEM-TRD prototype

- A test module was built at the University of Virginia
- The prototype of GEMTRD/T module has a size of 10 cm × 10 cm with a corresponding to a total of 512 channels for X/Y coordinates.
- The readout is based on flash ADC system developed at JLAB (fADC125) @125 MHz sampling.
- GEM-TRD provides e/hadron separation and tracking



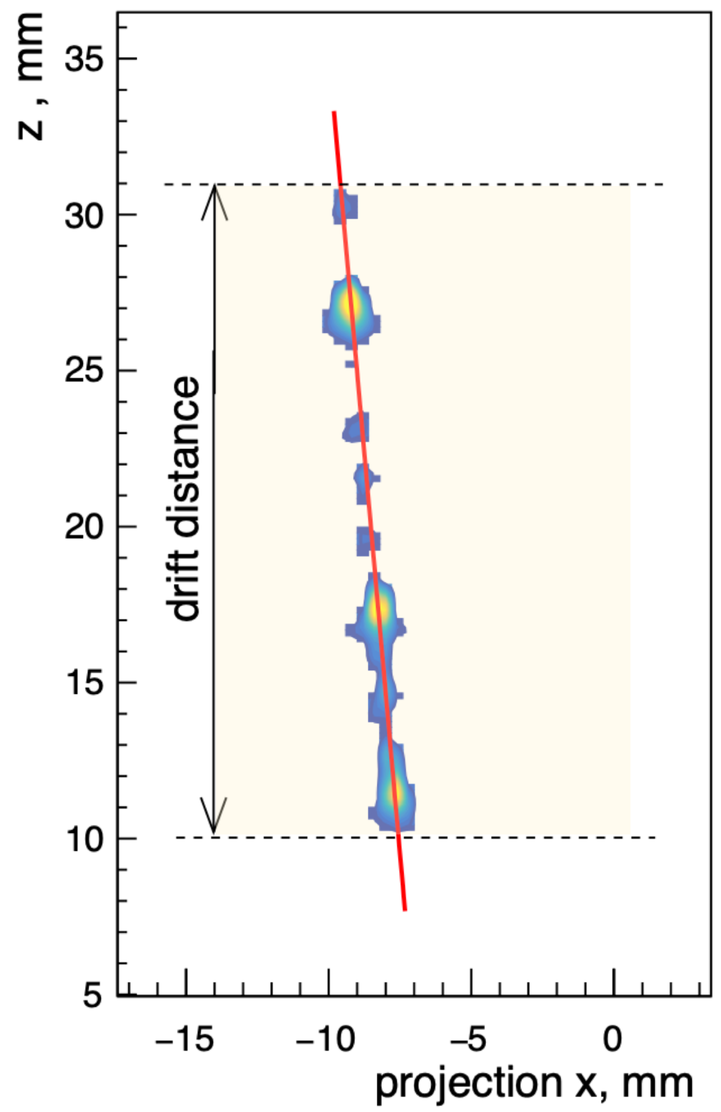


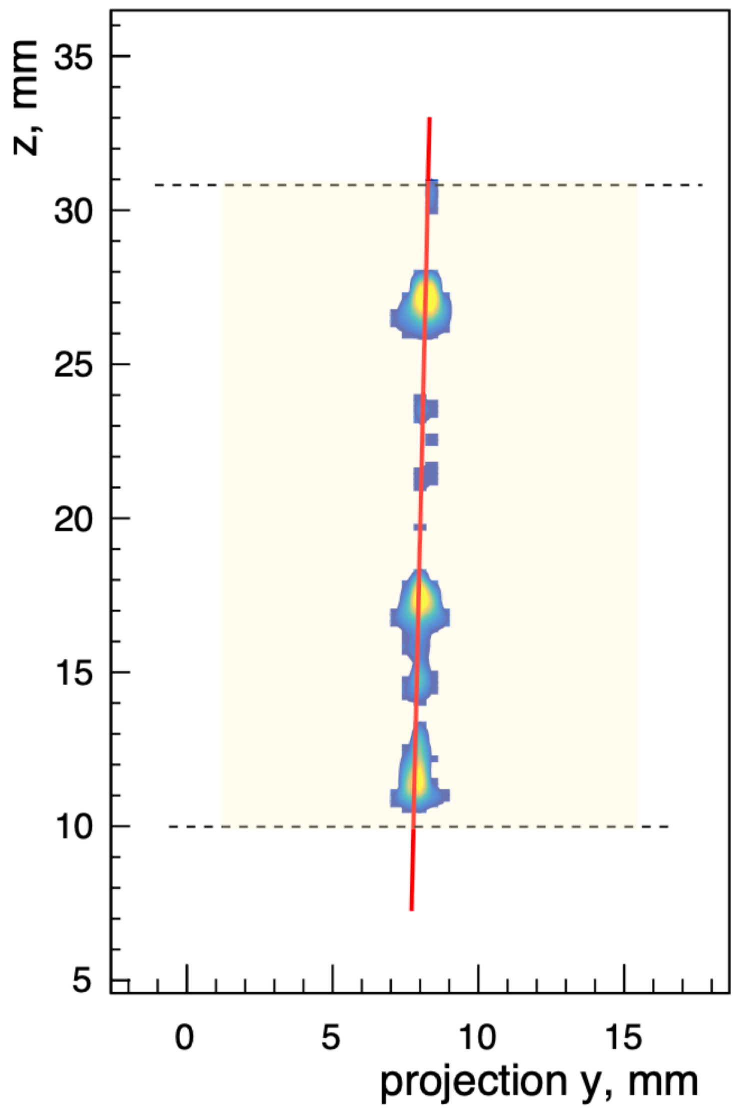




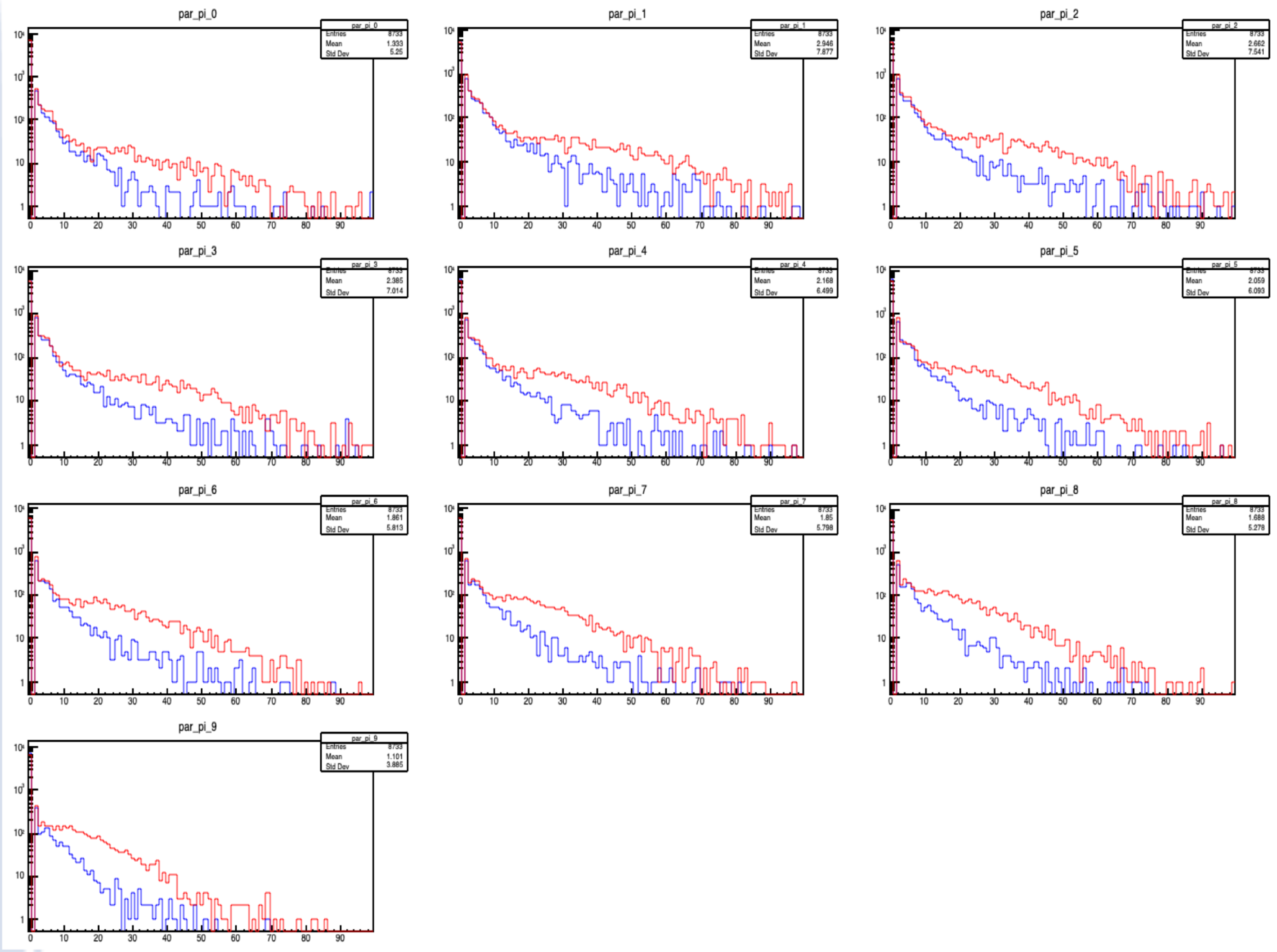
Tracking with GEM-TRD

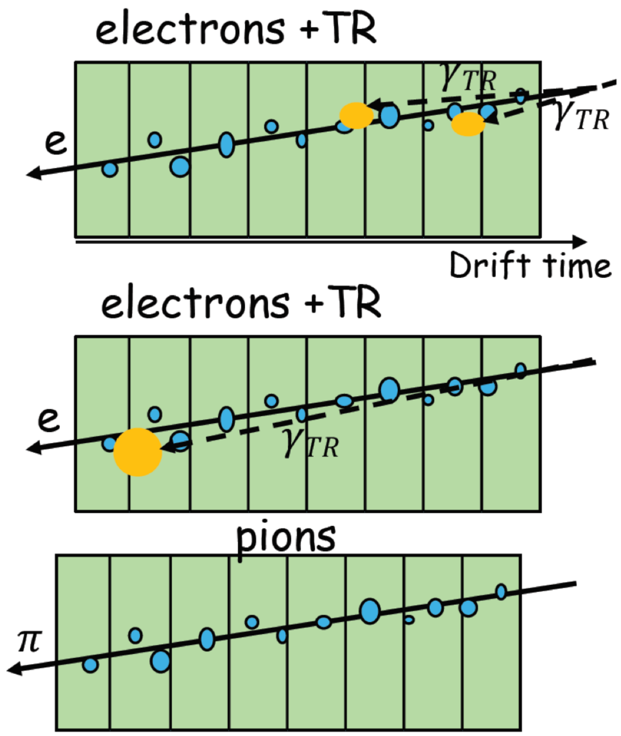
GEM-TRD can work as mini TPC, providing 3D track segments





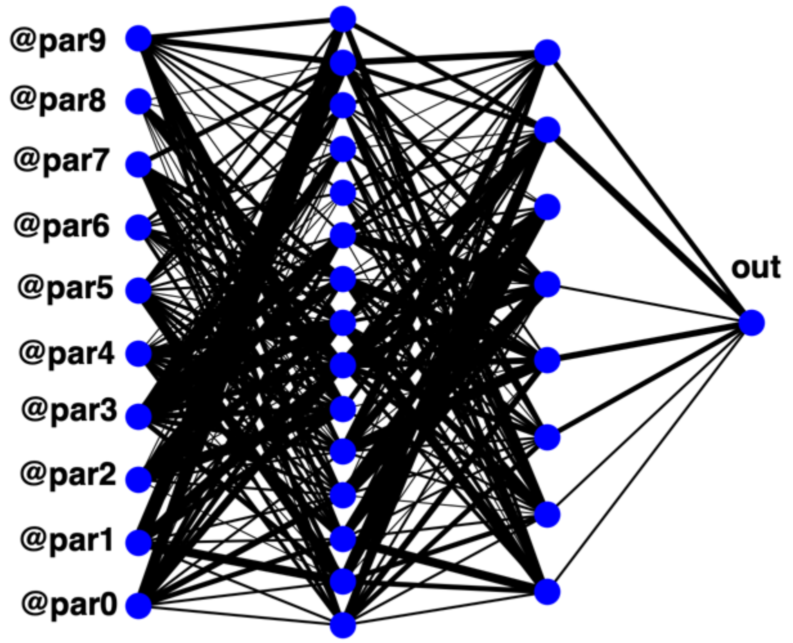
NN input parameters

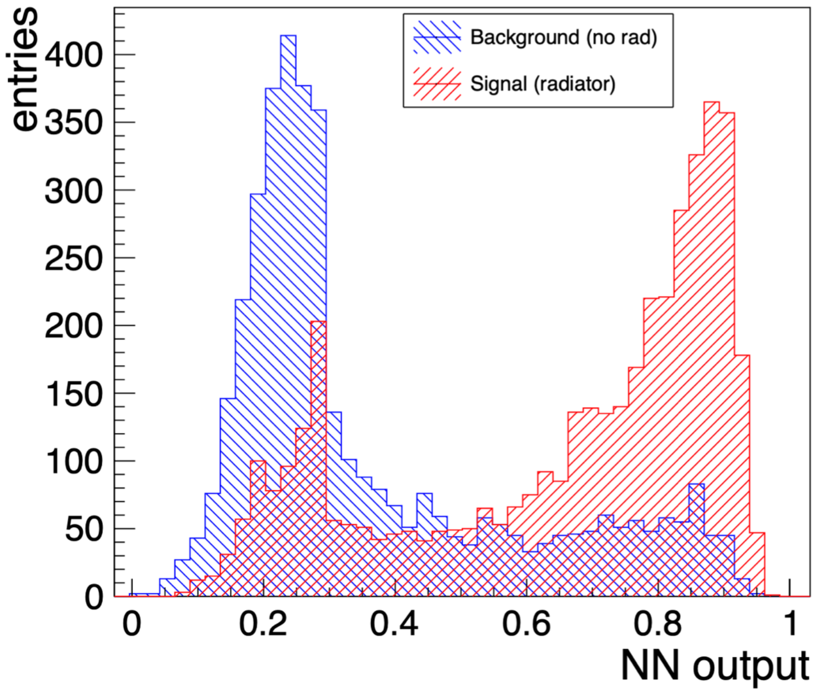




The last histogram represents the first time bin after entrance window with the most soft TR photons spectrum.

ML for GEMTRD offline analysis





- *For data analysis we used a neural network library provided by root /TMVA package : MultiLayerPerceptron (MLP)*
- *All data was divided into 2 samples: training and test samples*
- *Top right plot shows neural network output for single module:*
  - Red - electrons with radiator
  - Blue – electrons without radiator

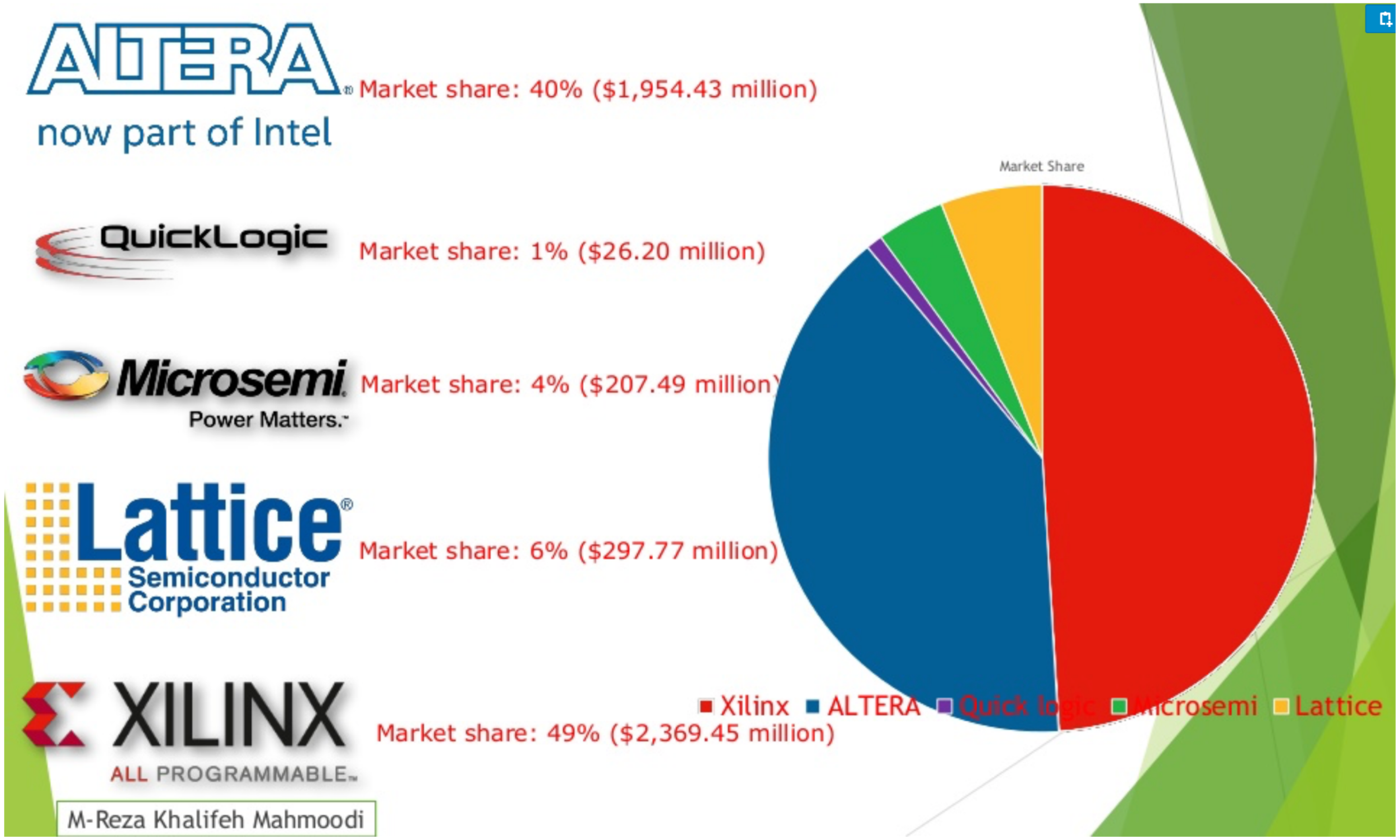
Moving forward

- *Offline analysis using ML looks promising.*
- *Can it be done in real time ?*
- *Here are some of the possible solutions :*
  - Computer farm.
  - CPU + GPU
  - CPU + FPGA
  - FPGA only
- *Steps for beginners to implement an FPGA solution:*
  - Select FPGA for application in ML
  - Export an offline trained neural network (NN) from root to C++ file.
  - Convert logical topology of NN coded in C++ to RTL structure of FPGA in VHDL or Verilog.
  - Optimize the NN for application in FPGA.
  - Create an I/O interface and configure FPGA.
  - Perform the test with hardware.

FPGA market

Jefferson Lab
Thomas Jefferson National Accelerator Facility

Since Intel acquired Altera, Xilinx was left as the only major FPGA company in the market. Xilinx has ~50% market share while Altera (Intel) has ~37% and Lattice Semiconductor has a 10% market share.



ALTERA now part of Intel — Market share: 40% ($1,954.43 million)

QuickLogic — Market share: 1% ($26.20 million)

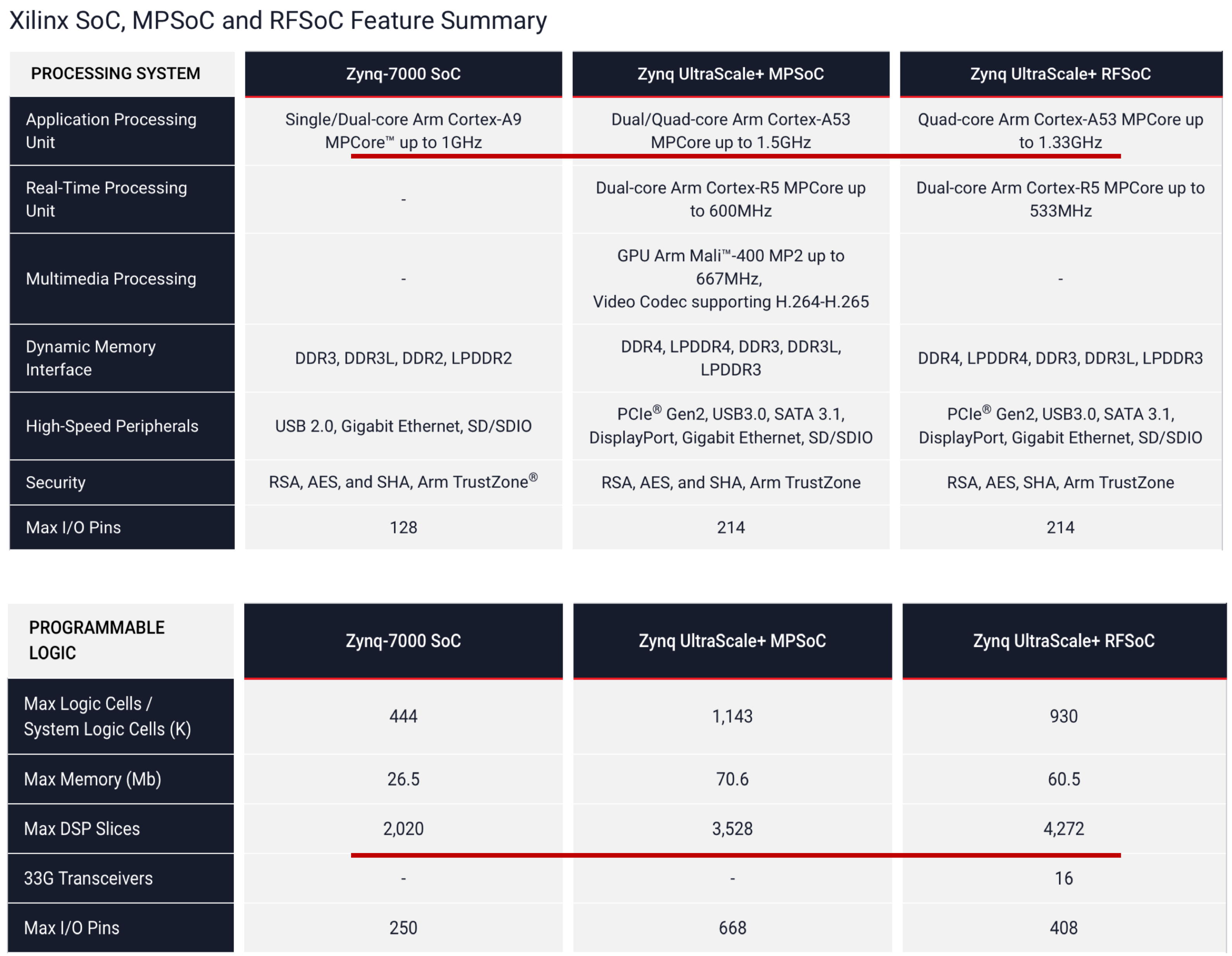
Microsemi Power Matters. — Market share: 4% ($207.49 million)

Lattice Semiconductor Corporation — Market share: 6% ($297.77 million)

XILINX ALL PROGRAMMABLE. — Market share: 49% ($2,369.45 million)

Xilinx Zynq FPGA family

Jefferson Lab
Thomas Jefferson National Accelerator Facility



Xilinx SoC, MPSoC and RFSoC Feature Summary

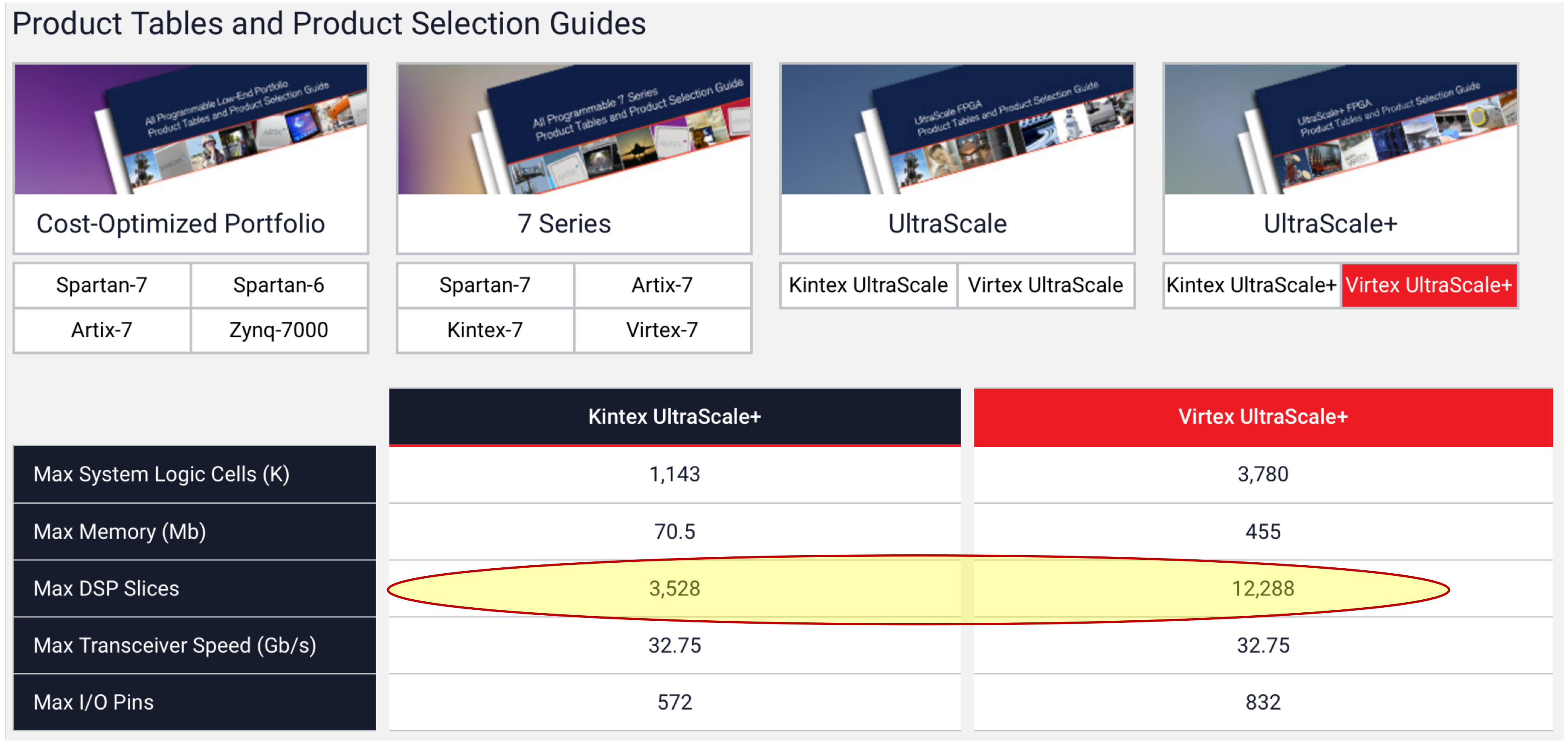
PROCESSING SYSTEM	Zynq-7000 SoC	Zynq UltraScale+ MPSoC	Zynq UltraScale+ RFSoC
Application Processing Unit	Single/Dual-core Arm Cortex-A9 MPCore™ up to 1GHz	Dual/Quad-core Arm Cortex-A53 MPCore up to 1.5GHz	Quad-core Arm Cortex-A53 MPCore up to 1.33GHz
Real-Time Processing Unit	-	Dual-core Arm Cortex-R5 MPCore up to 600MHz	Dual-core Arm Cortex-R5 MPCore up to 533MHz
Multimedia Processing	-	GPU Arm Mali™-400 MP2 up to 667MHz, Video Codec supporting H.264-H.265	-
Dynamic Memory Interface	DDR3, DDR3L, DDR2, LPDDR2	DDR4, LPDDR4, DDR3, DDR3L, LPDDR3	DDR4, LPDDR4, DDR3, DDR3L, LPDDR3
High-Speed Peripherals	USB 2.0, Gigabit Ethernet, SD/SDIO	PCIe® Gen2, USB3.0, SATA 3.1, DisplayPort, Gigabit Ethernet, SD/SDIO	PCIe® Gen2, USB3.0, SATA 3.1, DisplayPort, Gigabit Ethernet, SD/SDIO
Security	RSA, AES, and SHA, Arm TrustZone®	RSA, AES, and SHA, Arm TrustZone	RSA, AES, SHA, Arm TrustZone
Max I/O Pins	128	214	214

PROGRAMMABLE LOGIC	Zynq-7000 SoC	Zynq UltraScale+ MPSoC	Zynq UltraScale+ RFSoC
Max Logic Cells / System Logic Cells (K)	444	1,143	930
Max Memory (Mb)	26.5	70.6	60.5
Max DSP Slices	2,020	3,528	4,272
33G Transceivers	-	-	16
Max I/O Pins	250	668	408

Xilinx UltraScale+ family

Xilinx Ultrascale+ has the most DSP on a chip.

But it lacks a real processor on a chip.



Product Tables and Product Selection Guides

Cost-Optimized Portfolio		7 Series		UltraScale		UltraScale+	
Spartan-7	Spartan-6	Spartan-7	Artix-7	Kintex UltraScale	Virtex UltraScale	Kintex UltraScale+	Virtex UltraScale+
Artix-7	Zynq-7000	Kintex-7	Virtex-7				

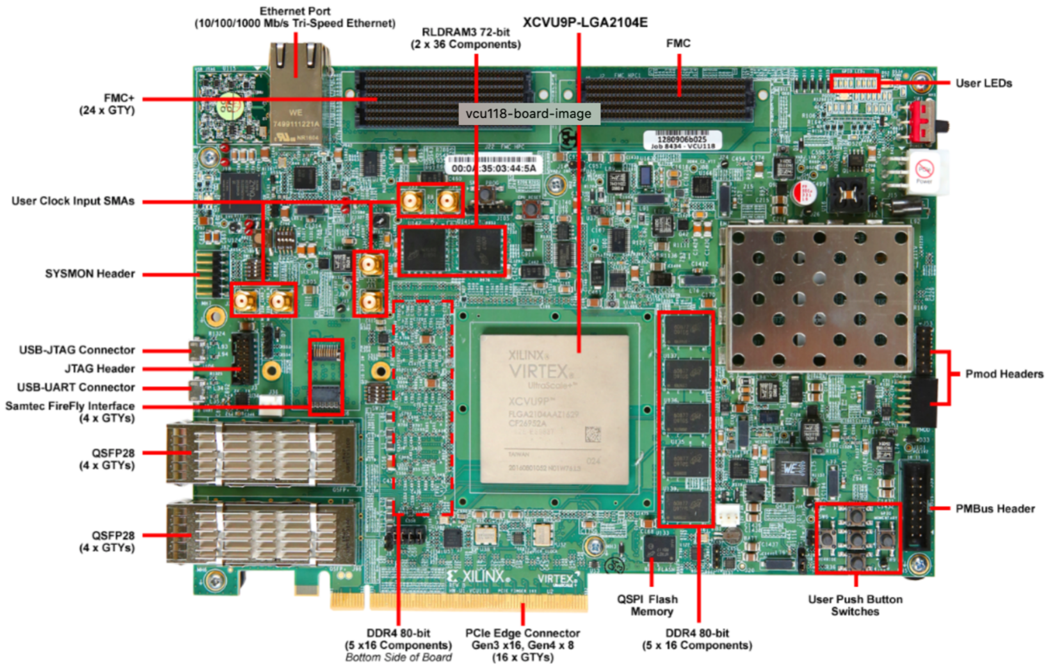
	Kintex UltraScale+	Virtex UltraScale+
Max System Logic Cells (K)	1,143	3,780
Max Memory (Mb)	70.5	455
Max DSP Slices	3,528	12,288
Max Transceiver Speed (Gb/s)	32.75	32.75
Max I/O Pins	572	832

Xilinx Virtex® UltraScale+™

Jefferson Lab
Thomas Jefferson National Accelerator Facility

- *At an early stage in this project, as hardware to test ML algorithms on FPGA , we use a standard Xilinx evaluation boards rather than developing a customized FPGA board. These boards have functions and interfaces sufficient for proof of principle of ML-FPGA.*
- *The Xilinx evaluation board includes the Xilinx XCVU9P and 6,840 DSP slices. Each includes a hardwired optimized multiply unit and collectively offers a peak theoretical performance in excess of 1 Tera multiplications per second.*
- *Second, the internal organization can be optimized to the specific computational problem. The internal data processing architecture can support deep computational pipelines offering high throughputs.*
- *Third, the FPGA supports high speed I/O interfaces including Ethernet and 180 high speed transceivers that can operate in excess of 30 Gbps.*

Featuring the **Virtex® UltraScale+™ XCVU9P-L2FLGA2104E FPGA**

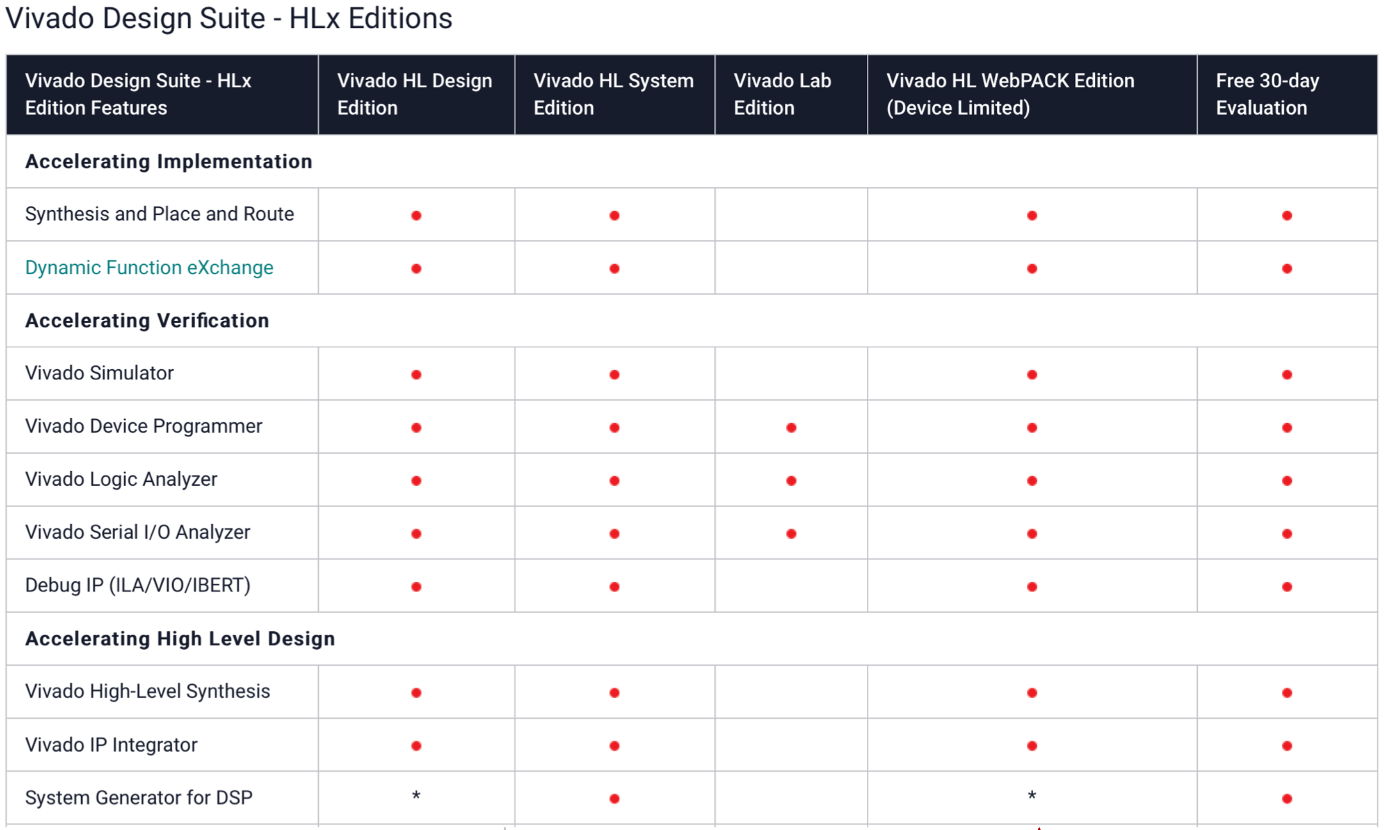

Ethernet Port (10/100/1000 Mb/s Tri-Speed Ethernet)
RLDRAM3 72-bit (2 x 36 Components)
XCVU9P-LGA2104E
FMC
User LEDs
FMC+ (24 x GTY)
vcu118-board-image
User Clock Input SMAs
SYSMON Header
USB-JTAG Connector
JTAG Header
USB-UART Connector
Samtec FireFly Interface (4 x GTYs)
QSFP28 (4 x GTYs)
QSFP28 (4 x GTYs)
Pmod Headers
PMBus Header
QSPI Flash Memory
User Push Button Switches
DDR4 80-bit (5 x16 Components) Bottom Side of Board
PCIe Edge Connector Gen3 x16, Gen4 x 8 (16 x GTYs)
DDR4 80-bit (5 x 16 Components)

Xilinx Virtex® UltraScale+™

Evaluation board XCVU9P includes software license (node locked & device-locked) with 1 year of updates.

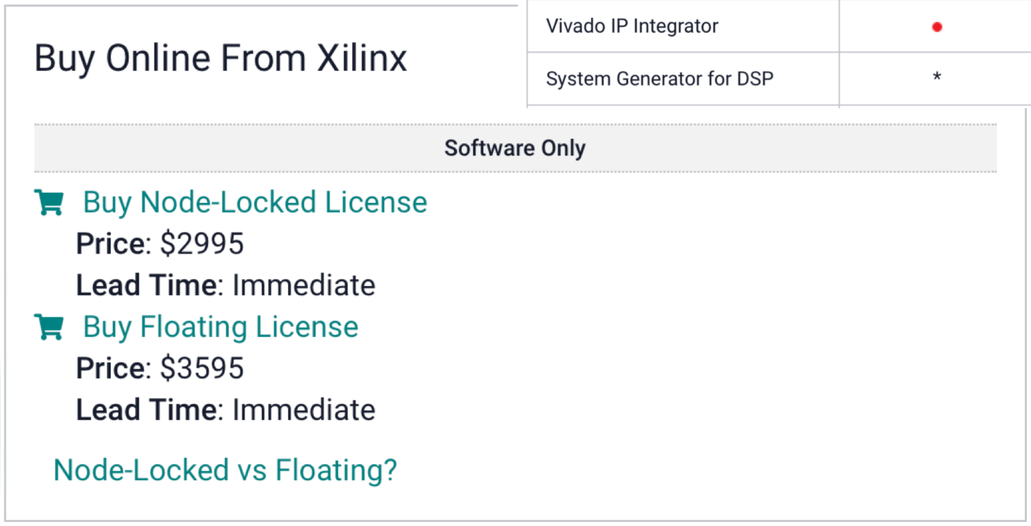
Xilinx design software

Jefferson Lab
Thomas Jefferson National Accelerator Facility



Vivado Design Suite - HLx Editions

Vivado Design Suite - HLx Edition Features	Vivado HL Design Edition	Vivado HL System Edition	Vivado Lab Edition	Vivado HL WebPACK Edition (Device Limited)	Free 30-day Evaluation
Accelerating Implementation					
Synthesis and Place and Route	●	●		●	●
Dynamic Function eXchange	●	●		●	●
Accelerating Verification					
Vivado Simulator	●	●		●	●
Vivado Device Programmer	●	●	●	●	●
Vivado Logic Analyzer	●	●	●	●	●
Vivado Serial I/O Analyzer	●	●	●	●	●
Debug IP (ILA/VIO/IBERT)	●	●		●	●
Accelerating High Level Design					
Vivado High-Level Synthesis	●	●		●	●
Vivado IP Integrator	●	●		●	●
System Generator for DSP	*	●		*	●



Buy Online From Xilinx

Software Only

Buy Node-Locked License
Price: $2995
Lead Time: Immediate

Buy Floating License
Price: $3595
Lead Time: Immediate

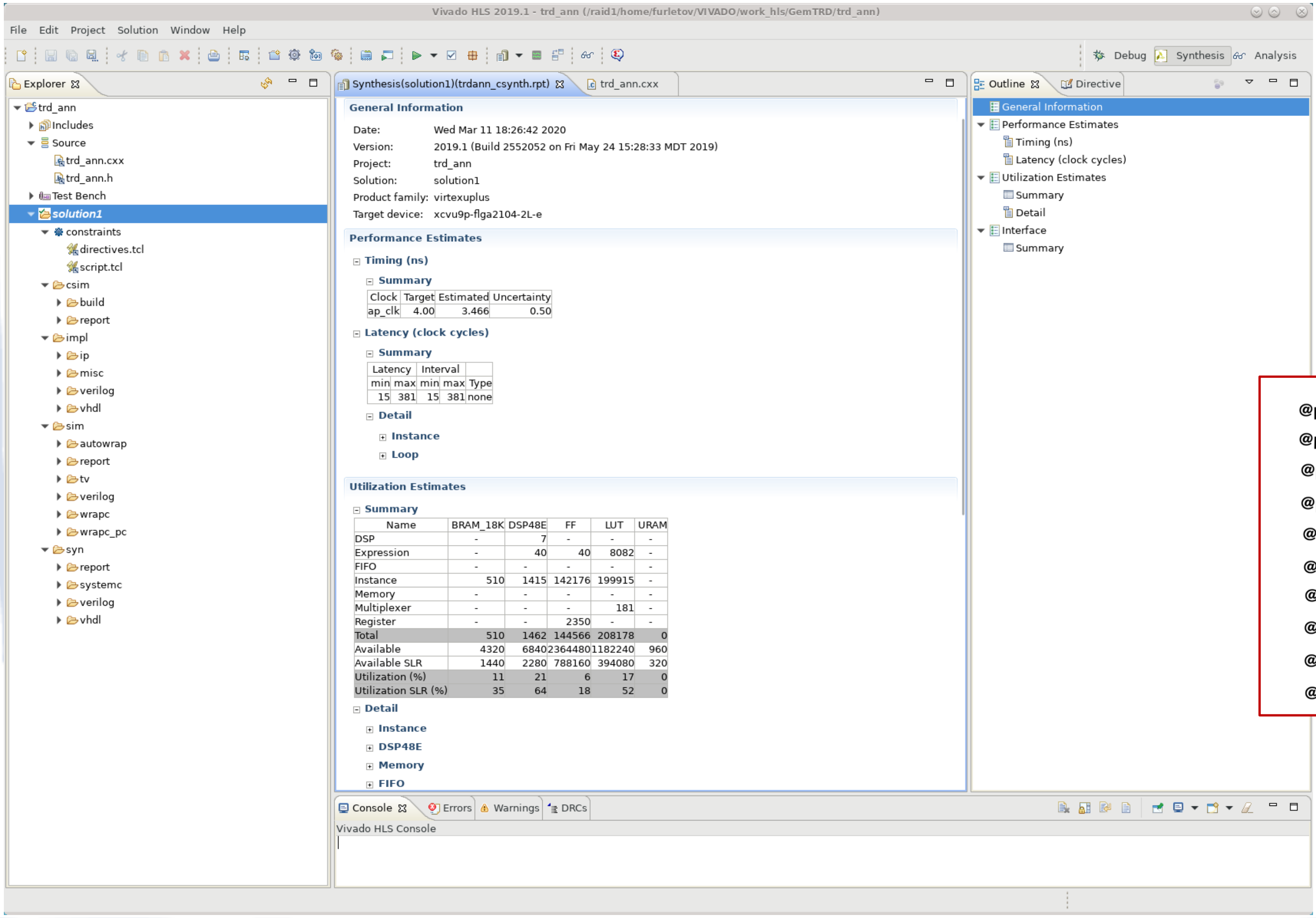
Node-Locked vs Floating?

WebPACK is free, but device limited, and does not support XCVU9P

Xilinx High-Level Synthesis

Jefferson Lab
Thomas Jefferson National Accelerator Facility

The Xilinx Vivado HLS (High-Level Synthesis) tool provides a higher level of abstraction for the user by synthesizing functions written in C,C++ into IP blocks, by generating the appropriate ,low-level, VHDL and Verilog code. Then those blocks can be integrated into a real hardware system.



General Information

Date:	Wed Mar 11 18:26:42 2020
Version:	2019.1 (Build 2552052 on Fri May 24 15:28:33 MDT 2019)
Project:	trd_ann
Solution:	solution1
Product family:	virtexuplus
Target device:	xcvu9p-flga2104-2L-e

Performance Estimates

Timing (ns)

Summary

Clock	Target	Estimated	Uncertainty
ap_clk	4.00	3.466	0.50

Latency (clock cycles)

Summary

Latency		Interval		
min	max	min	max	Type
15	381	15	381	none

Detail

Instance

Loop

Utilization Estimates

Summary

Name	BRAM_18K	DSP48E	FF	LUT	URAM
DSP	-	7	-	-	-
Expression	-	40	40	8082	-
FIFO	-	-	-	-	-
Instance	510	1415	142176	199915	-
Memory	-	-	-	-	-
Multiplexer	-	-	-	181	-
Register	-	-	2350	-	-
Total	510	1462	144566	208178	0
Available	4320	6840	2364480	1182240	960
Available SLR	1440	2280	788160	394080	320
Utilization (%)	11	21	6	17	0
Utilization SLR (%)	35	64	18	52	0

Detail

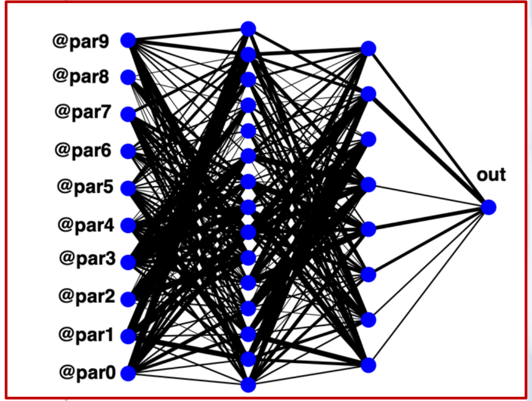
Instance

DSP48E

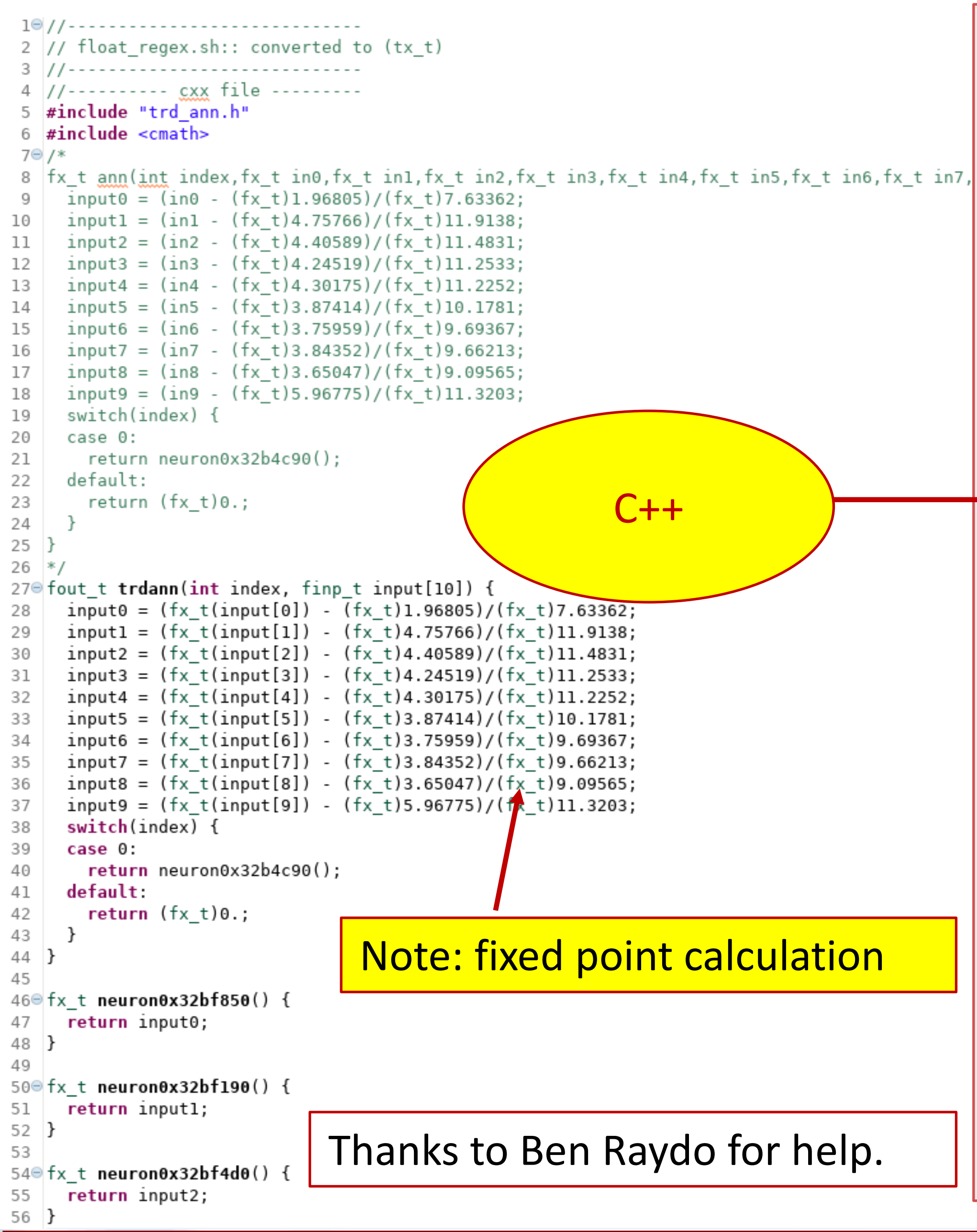
Memory

FIFO

The C/C++ code of the trained network is used as input for Vivado_HLS.



Xilinx HLS: C++ to Verilog



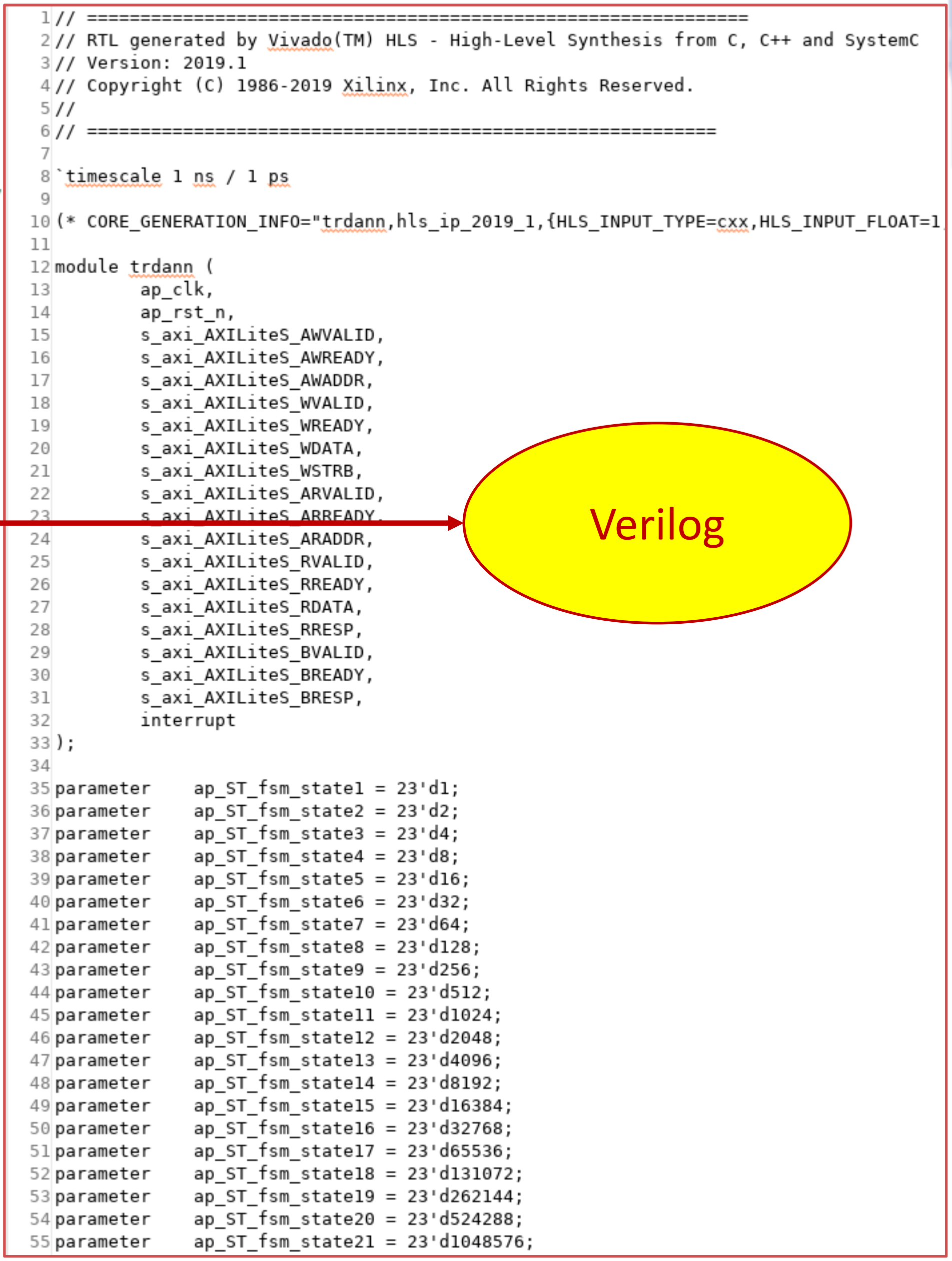
```
//-----------------------------
// float_regex.sh:: converted to (tx_t)
//-----------------------------
//---------- cxx file ---------
#include "trd_ann.h"
#include <cmath>
/*
fx_t ann(int index,fx_t in0,fx_t in1,fx_t in2,fx_t in3,fx_t in4,fx_t in5,fx_t in6,fx_t in7,
  input0 = (in0 - (fx_t)1.96805)/(fx_t)7.63362;
  input1 = (in1 - (fx_t)4.75766)/(fx_t)11.9138;
  input2 = (in2 - (fx_t)4.40589)/(fx_t)11.4831;
  input3 = (in3 - (fx_t)4.24519)/(fx_t)11.2533;
  input4 = (in4 - (fx_t)4.30175)/(fx_t)11.2252;
  input5 = (in5 - (fx_t)3.87414)/(fx_t)10.1781;
  input6 = (in6 - (fx_t)3.75959)/(fx_t)9.69367;
  input7 = (in7 - (fx_t)3.84352)/(fx_t)9.66213;
  input8 = (in8 - (fx_t)3.65047)/(fx_t)9.09565;
  input9 = (in9 - (fx_t)5.96775)/(fx_t)11.3203;
  switch(index) {
  case 0:
    return neuron0x32b4c90();
  default:
    return (fx_t)0.;
  }
}
*/
fout_t trdann(int index, finp_t input[10]) {
  input0 = (fx_t(input[0]) - (fx_t)1.96805)/(fx_t)7.63362;
  input1 = (fx_t(input[1]) - (fx_t)4.75766)/(fx_t)11.9138;
  input2 = (fx_t(input[2]) - (fx_t)4.40589)/(fx_t)11.4831;
  input3 = (fx_t(input[3]) - (fx_t)4.24519)/(fx_t)11.2533;
  input4 = (fx_t(input[4]) - (fx_t)4.30175)/(fx_t)11.2252;
  input5 = (fx_t(input[5]) - (fx_t)3.87414)/(fx_t)10.1781;
  input6 = (fx_t(input[6]) - (fx_t)3.75959)/(fx_t)9.69367;
  input7 = (fx_t(input[7]) - (fx_t)3.84352)/(fx_t)9.66213;
  input8 = (fx_t(input[8]) - (fx_t)3.65047)/(fx_t)9.09565;
  input9 = (fx_t(input[9]) - (fx_t)5.96775)/(fx_t)11.3203;
  switch(index) {
  case 0:
    return neuron0x32b4c90();
  default:
    return (fx_t)0.;
  }
}

fx_t neuron0x32bf850() {
  return input0;
}

fx_t neuron0x32bf190() {
  return input1;
}

fx_t neuron0x32bf4d0() {
  return input2;
}
```

C++

Note: fixed point calculation

Thanks to Ben Raydo for help.

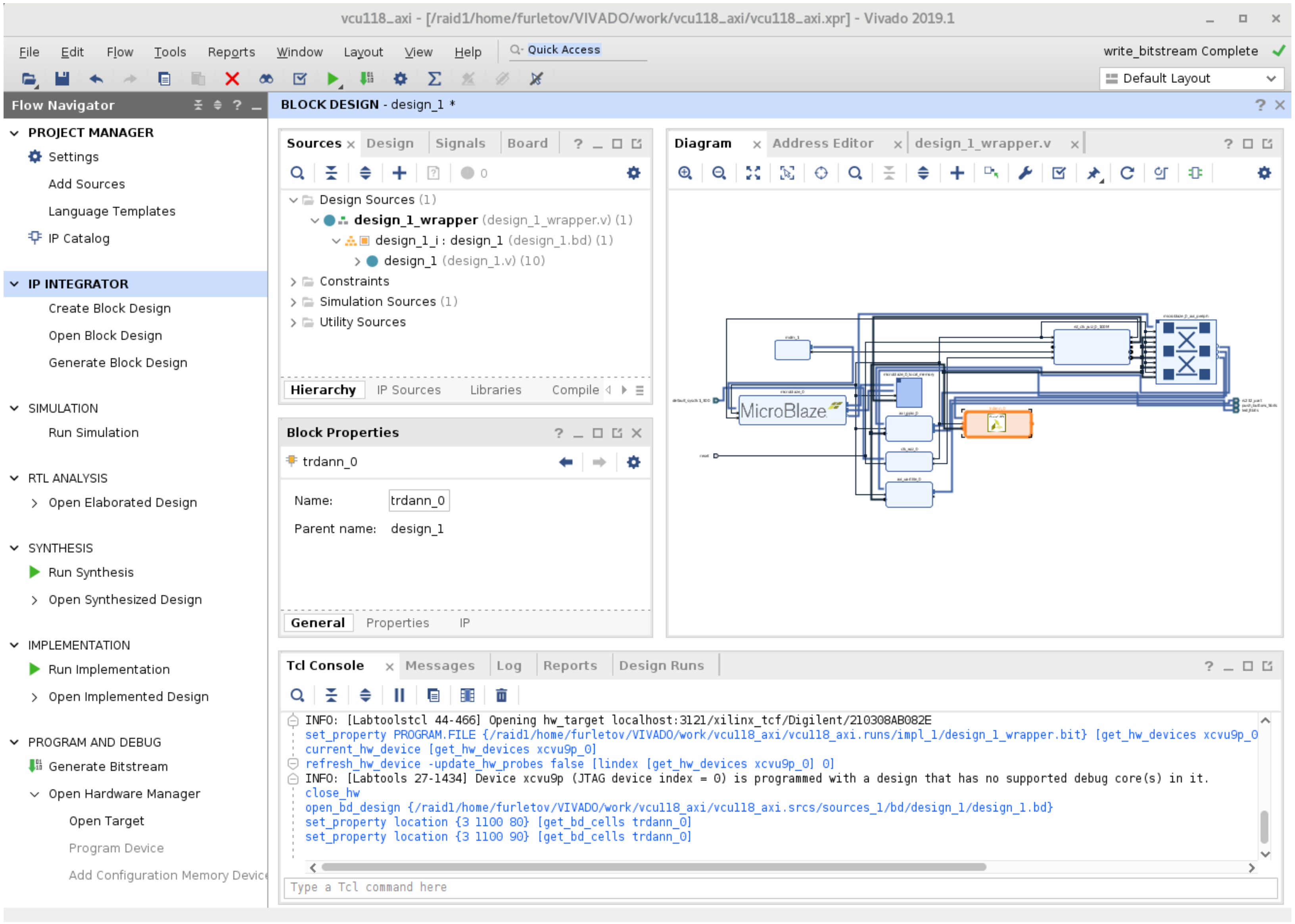
Verilog



```
// ==============================================================
// RTL generated by Vivado(TM) HLS - High-Level Synthesis from C, C++ and SystemC
// Version: 2019.1
// Copyright (C) 1986-2019 Xilinx, Inc. All Rights Reserved.
//
// ===========================================================

`timescale 1 ns / 1 ps

(* CORE_GENERATION_INFO="trdann,hls_ip_2019_1,{HLS_INPUT_TYPE=cxx,HLS_INPUT_FLOAT=1

module trdann (
        ap_clk,
        ap_rst_n,
        s_axi_AXILiteS_AWVALID,
        s_axi_AXILiteS_AWREADY,
        s_axi_AXILiteS_AWADDR,
        s_axi_AXILiteS_WVALID,
        s_axi_AXILiteS_WREADY,
        s_axi_AXILiteS_WDATA,
        s_axi_AXILiteS_WSTRB,
        s_axi_AXILiteS_ARVALID,
        s_axi_AXILiteS_ARREADY,
        s_axi_AXILiteS_ARADDR,
        s_axi_AXILiteS_RVALID,
        s_axi_AXILiteS_RREADY,
        s_axi_AXILiteS_RDATA,
        s_axi_AXILiteS_RRESP,
        s_axi_AXILiteS_BVALID,
        s_axi_AXILiteS_BREADY,
        s_axi_AXILiteS_BRESP,
        interrupt
);

parameter    ap_ST_fsm_state1 = 23'd1;
parameter    ap_ST_fsm_state2 = 23'd2;
parameter    ap_ST_fsm_state3 = 23'd4;
parameter    ap_ST_fsm_state4 = 23'd8;
parameter    ap_ST_fsm_state5 = 23'd16;
parameter    ap_ST_fsm_state6 = 23'd32;
parameter    ap_ST_fsm_state7 = 23'd64;
parameter    ap_ST_fsm_state8 = 23'd128;
parameter    ap_ST_fsm_state9 = 23'd256;
parameter    ap_ST_fsm_state10 = 23'd512;
parameter    ap_ST_fsm_state11 = 23'd1024;
parameter    ap_ST_fsm_state12 = 23'd2048;
parameter    ap_ST_fsm_state13 = 23'd4096;
parameter    ap_ST_fsm_state14 = 23'd8192;
parameter    ap_ST_fsm_state15 = 23'd16384;
parameter    ap_ST_fsm_state16 = 23'd32768;
parameter    ap_ST_fsm_state17 = 23'd65536;
parameter    ap_ST_fsm_state18 = 23'd131072;
parameter    ap_ST_fsm_state19 = 23'd262144;
parameter    ap_ST_fsm_state20 = 23'd524288;
parameter    ap_ST_fsm_state21 = 23'd1048576;
```

Xilinx Vivado

Vivado Design Suite is a software suite produced by Xilinx for synthesis and analysis of HDL designs.



vcu118_axi - [/raid1/home/furletov/VIVADO/work/vcu118_axi/vcu118_axi.xpr] - Vivado 2019.1

write_bitstream Complete

BLOCK DESIGN - design_1 *

Design Sources (1)
- design_1_wrapper (design_1_wrapper.v) (1)
  - design_1_i : design_1 (design_1.bd) (1)
    - design_1 (design_1.v) (10)

Constraints
Simulation Sources (1)
Utility Sources

Block Properties: trdann_0

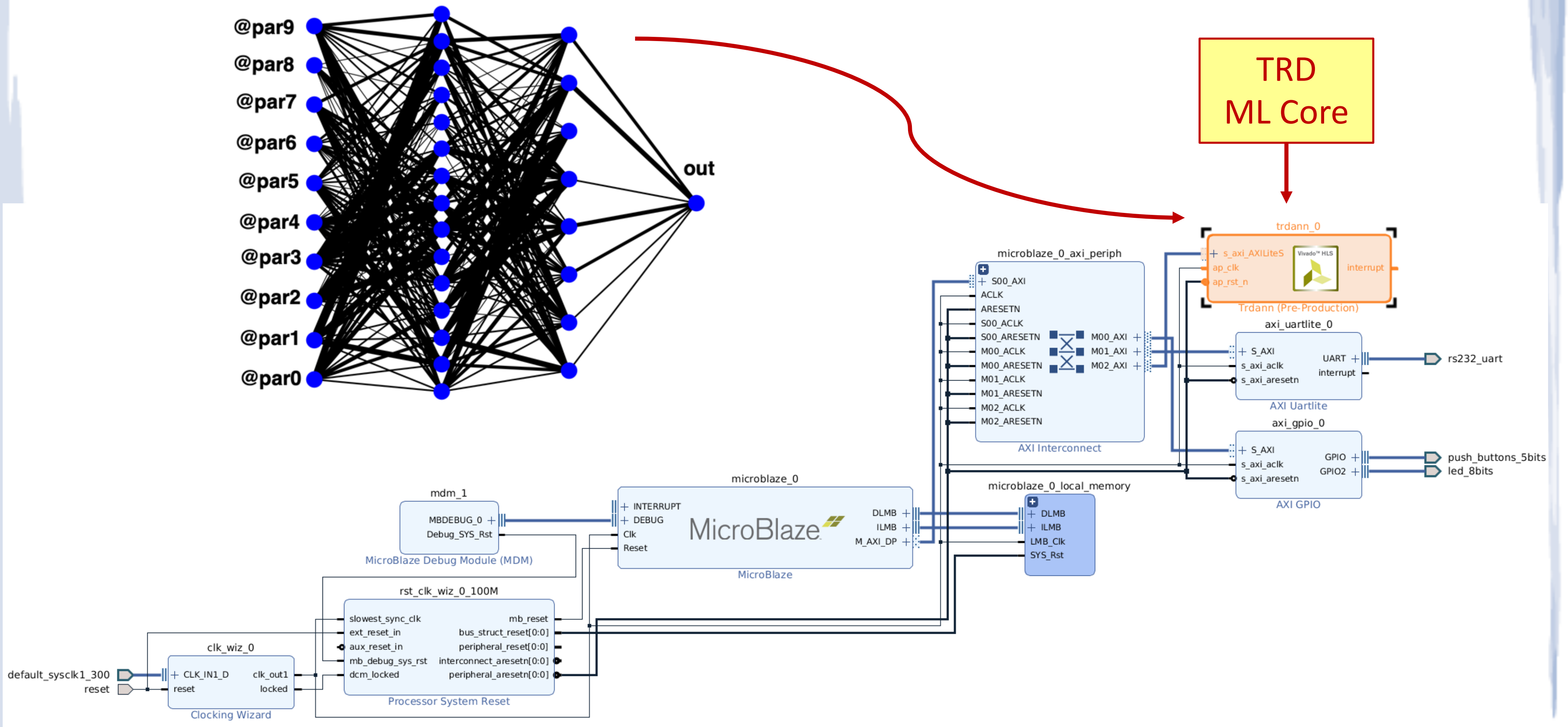
Name: trdann_0

Parent name: design_1

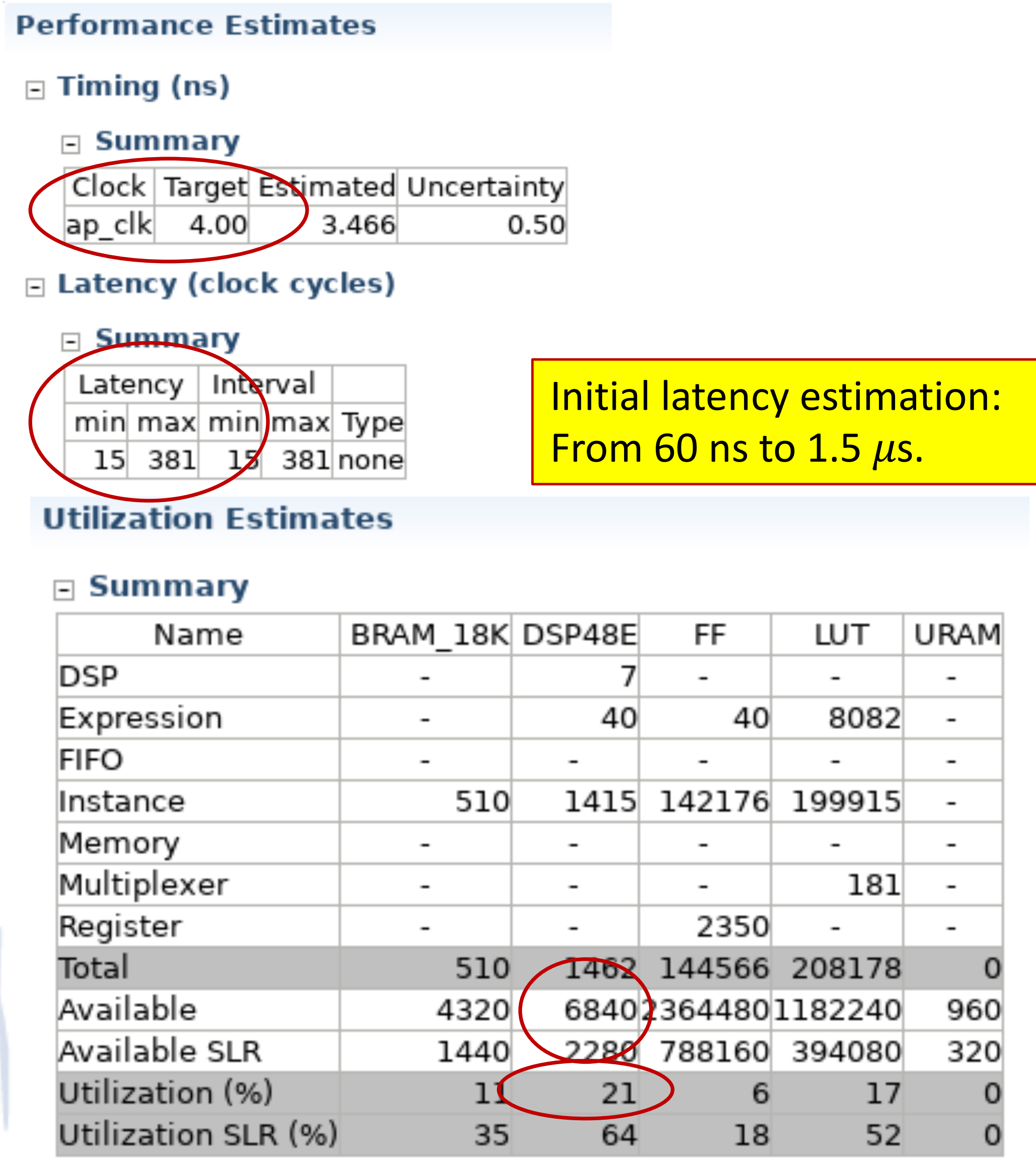
```
INFO: [Labtoolstcl 44-466] Opening hw_target localhost:3121/xilinx_tcf/Digilent/210308AB082E
set_property PROGRAM.FILE {/raid1/home/furletov/VIVADO/work/vcu118_axi/vcu118_axi.runs/impl_1/design_1_wrapper.bit} [get_hw_devices xcvu9p_0
current_hw_device [get_hw_devices xcvu9p_0]
refresh_hw_device -update_hw_probes false [lindex [get_hw_devices xcvu9p_0] 0]
INFO: [Labtools 27-1434] Device xcvu9p (JTAG device index = 0) is programmed with a design that has no supported debug core(s) in it.
close_hw
open_bd_design {/raid1/home/furletov/VIVADO/work/vcu118_axi/vcu118_axi.srcs/sources_1/bd/design_1/design_1.bd}
set_property location {3 1100 80} [get_bd_cells trdann_0]
set_property location {3 1100 90} [get_bd_cells trdann_0]
```

ML FPGA Core for TRD

- *Using HLS significantly decreases development time. (at the cost of lower efficiency of use of FPGA resources)*



Vivado implementation report



Performance Estimates

Timing (ns)

Summary

Clock	Target	Estimated	Uncertainty
ap_clk	4.00	3.466	0.50

Latency (clock cycles)

Summary

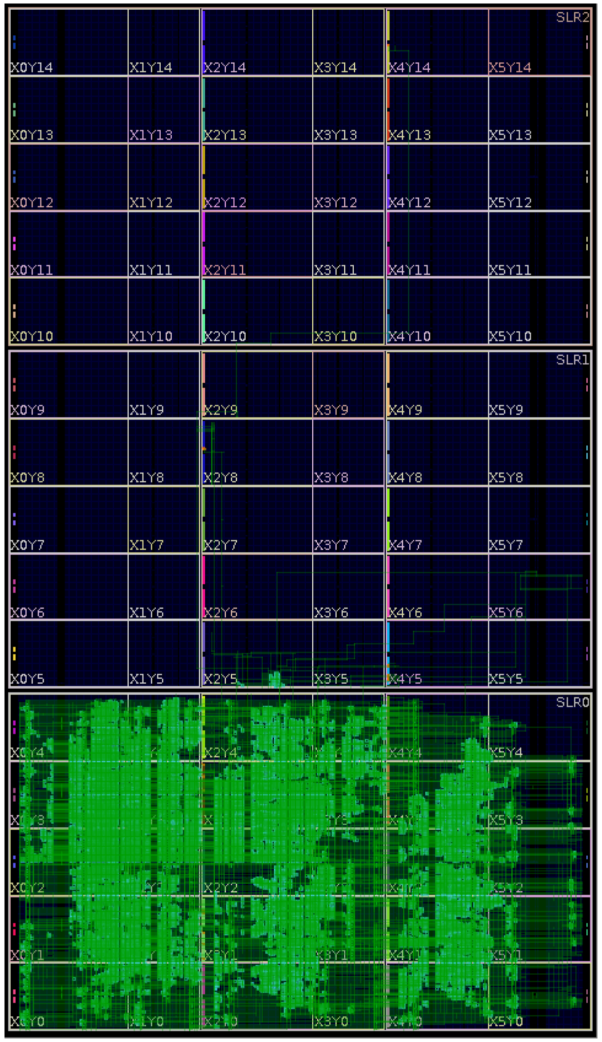
Latency		Interval		
min	max	min	max	Type
15	381	15	381	none

Utilization Estimates

Summary

Name	BRAM_18K	DSP48E	FF	LUT	URAM
DSP	-	7	-	-	-
Expression	-	40	40	8082	-
FIFO	-	-	-	-	-
Instance	510	1415	142176	199915	-
Memory	-	-	-	-	-
Multiplexer	-	-	-	181	-
Register	-	-	2350	-	-
Total	510	1462	144566	208178	0
Available	4320	6840	2364480	1182240	960
Available SLR	1440	2280	788160	394080	320
Utilization (%)	11	21	6	17	0
Utilization SLR (%)	35	64	18	52	0

Initial latency estimation:
From 60 ns to 1.5 μs.

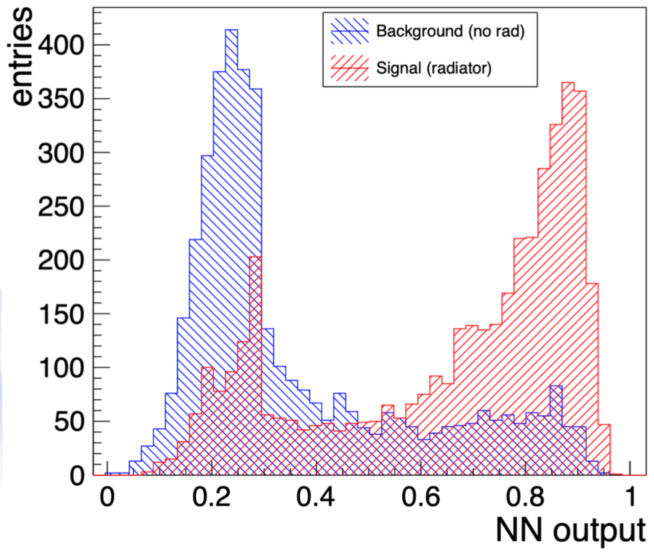


Test ML FPGA

Test tools:

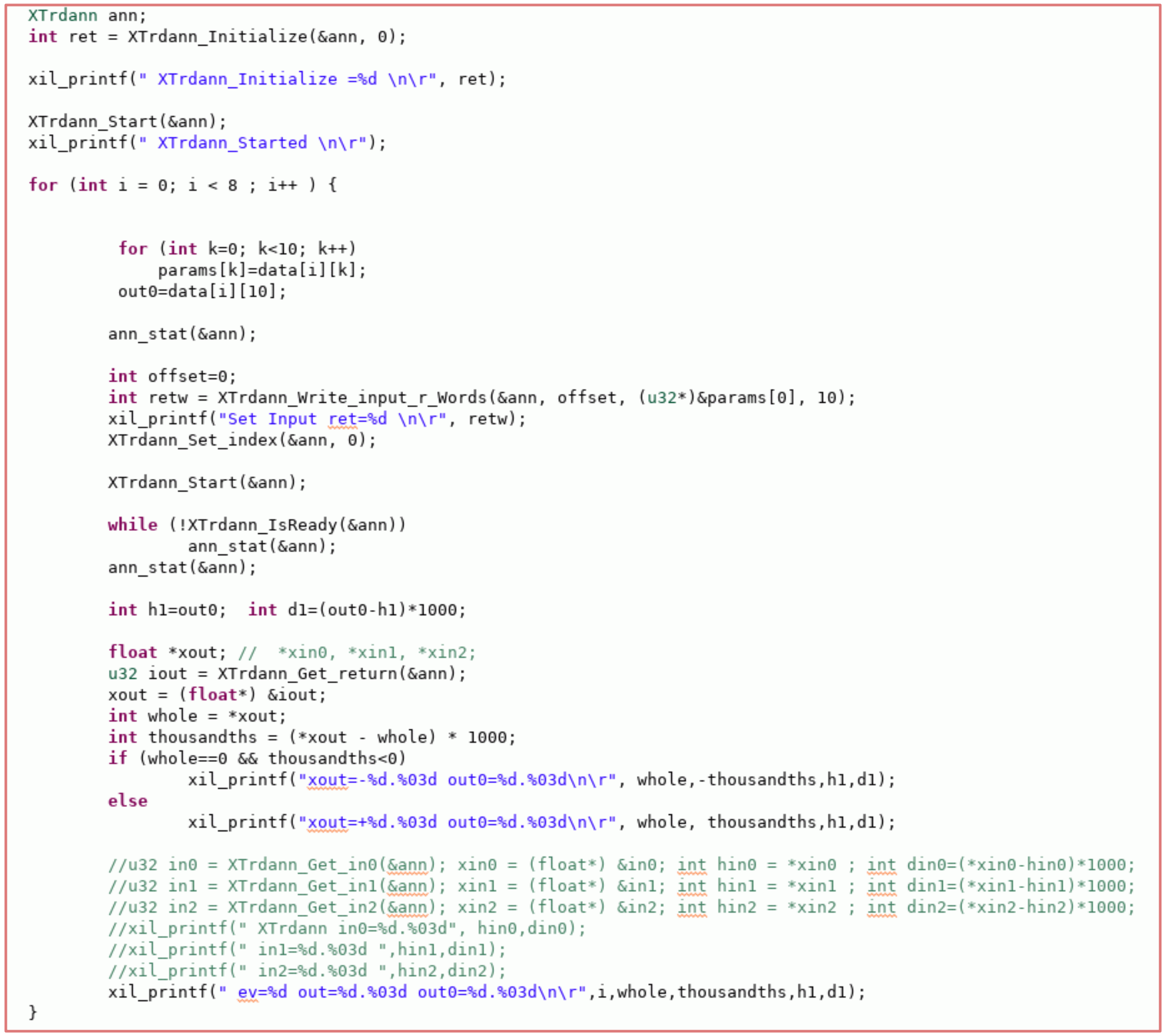
1. Vivado SDK
2. Petalinux

```
ev=0 out=0.192 out0=0.197
ev=1 out=0.192 out0=0.197
ev=2 out=0.233 out0=0.236
ev=3 out=0.192 out0=0.197
ev=4 out=0.165 out0=0.169
ev=5 out=0.192 out0=0.196
ev=6 out=0.462 out0=0.470
ev=7 out=0.187 out0=0.191
```



C++ code for test :

XTrdann ann; // create an instance of ML core.

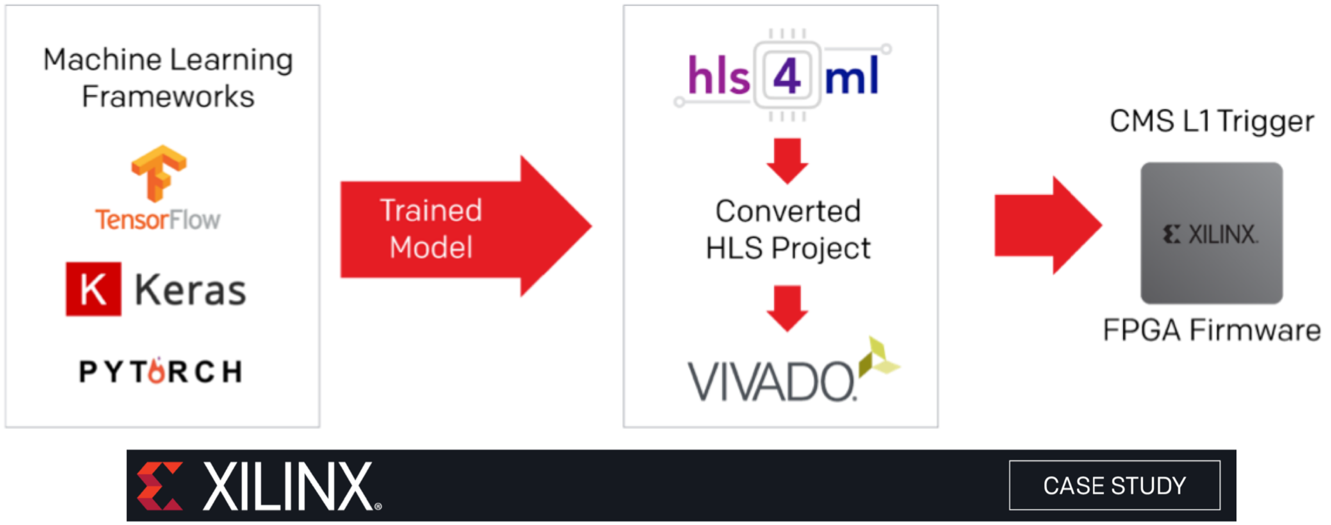


```
XTrdann ann;
int ret = XTrdann_Initialize(&ann, 0);

xil_printf(" XTrdann_Initialize =%d \n\r", ret);

XTrdann_Start(&ann);
xil_printf(" XTrdann_Started \n\r");

for (int i = 0; i < 8 ; i++ ) {


         for (int k=0; k<10; k++)
             params[k]=data[i][k];
         out0=data[i][10];

        ann_stat(&ann);

        int offset=0;
        int retw = XTrdann_Write_input_r_Words(&ann, offset, (u32*)&params[0], 10);
        xil_printf("Set Input ret=%d \n\r", retw);
        XTrdann_Set_index(&ann, 0);

        XTrdann_Start(&ann);

        while (!XTrdann_IsReady(&ann))
                ann_stat(&ann);
        ann_stat(&ann);

        int h1=out0;  int d1=(out0-h1)*1000;

        float *xout; //  *xin0, *xin1, *xin2;
        u32 iout = XTrdann_Get_return(&ann);
        xout = (float*) &iout;
        int whole = *xout;
        int thousandths = (*xout - whole) * 1000;
        if (whole==0 && thousandths<0)
                xil_printf("xout=-%d.%03d out0=%d.%03d\n\r", whole,-thousandths,h1,d1);
        else
                xil_printf("xout=+%d.%03d out0=%d.%03d\n\r", whole, thousandths,h1,d1);

        //u32 in0 = XTrdann_Get_in0(&ann); xin0 = (float*) &in0; int hin0 = *xin0 ; int din0=(*xin0-hin0)*1000;
        //u32 in1 = XTrdann_Get_in1(&ann); xin1 = (float*) &in1; int hin1 = *xin1 ; int din1=(*xin1-hin1)*1000;
        //u32 in2 = XTrdann_Get_in2(&ann); xin2 = (float*) &in2; int hin2 = *xin2 ; int din2=(*xin2-hin2)*1000;
        //xil_printf(" XTrdann in0=%d.%03d", hin0,din0);
        //xil_printf(" in1=%d.%03d ",hin1,din1);
        //xil_printf(" in2=%d.%03d ",hin2,din2);
        xil_printf(" ev=%d out=%d.%03d out0=%d.%03d\n\r",i,whole,thousandths,h1,d1);
}
```

hls4ml software

- ***`hls4ml`*** is a software package for creating HLS implementations of neural networks
- Supports common layer architectures and model software
- Highly customizable output for different latency and size needs
- Simple workflow to allow quick translation to HLS

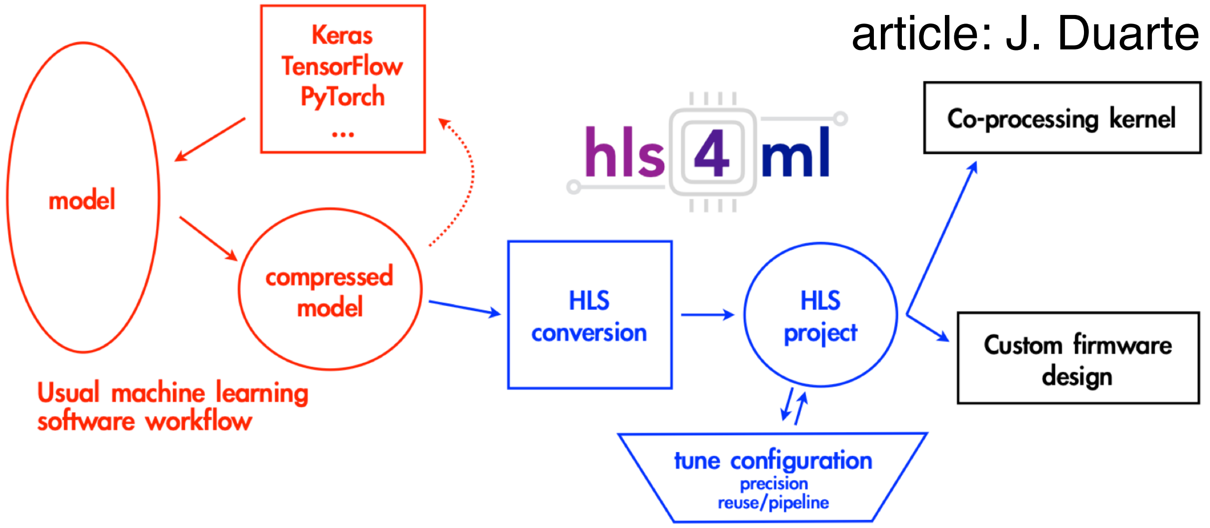
https://fastmachinelearning.org/hls4ml/

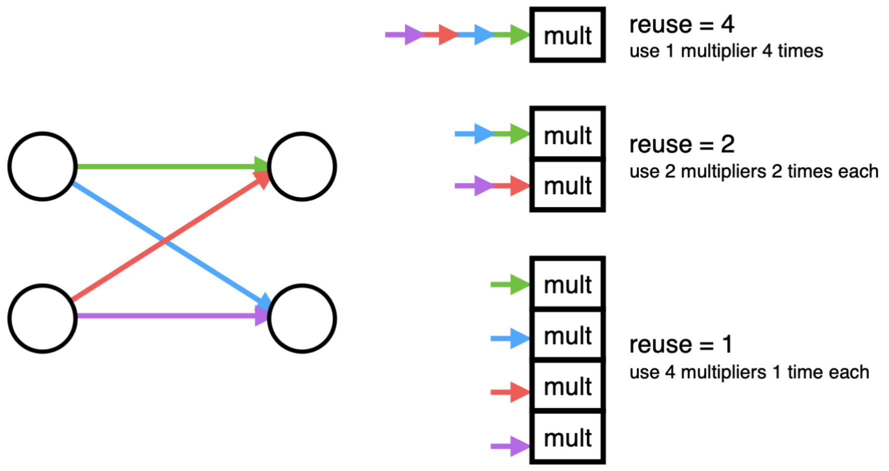


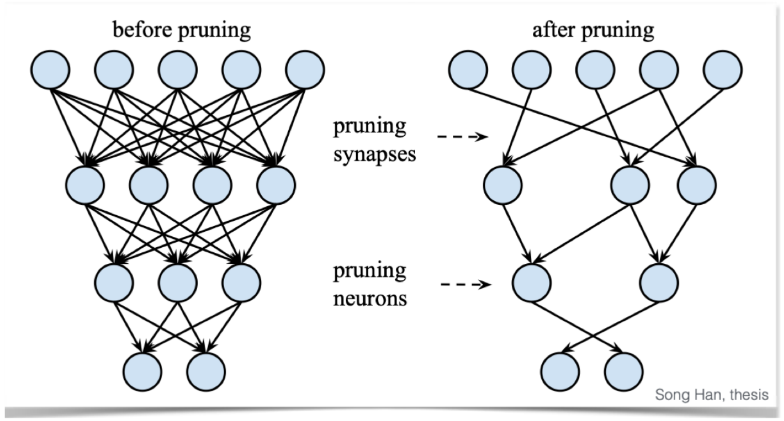
Optimization with hls4ml package

- A package hls4ml is developed based on High-Level Synthesis (HLS) to build machine learning models in FPGAs.

article: J. Duarte *et al* 2018 *JINST* **13** P07027



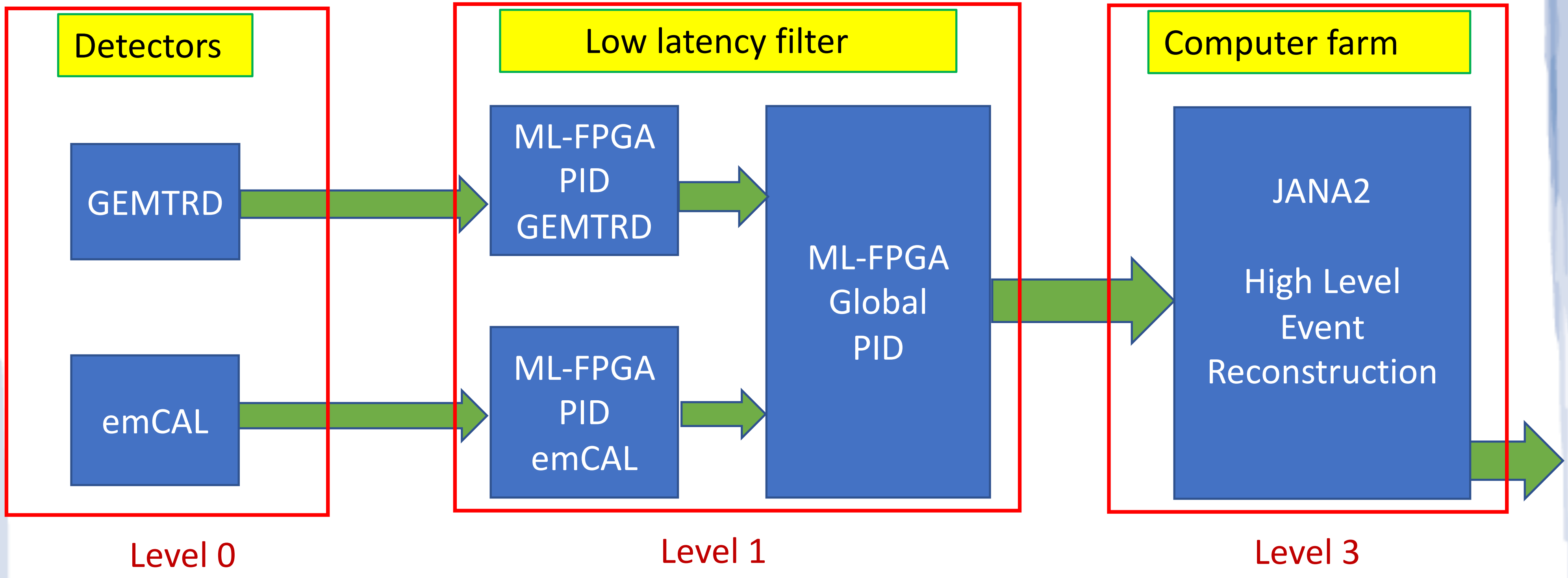


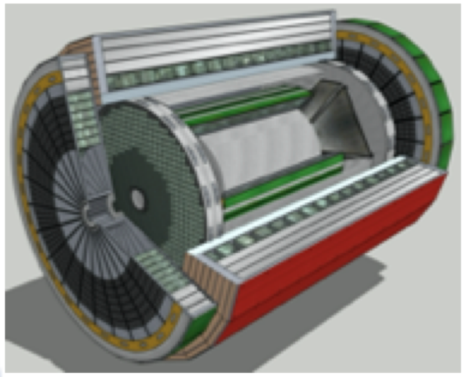


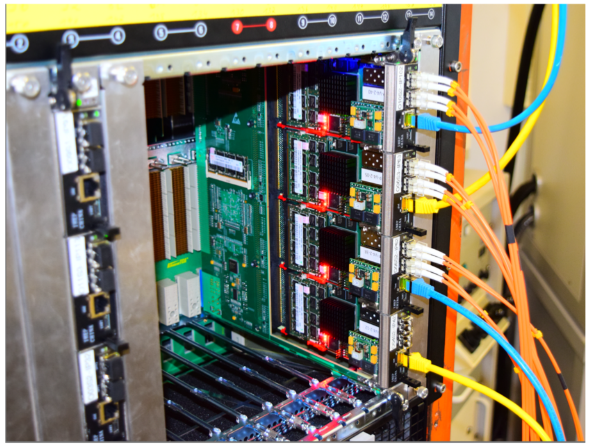
Project scope

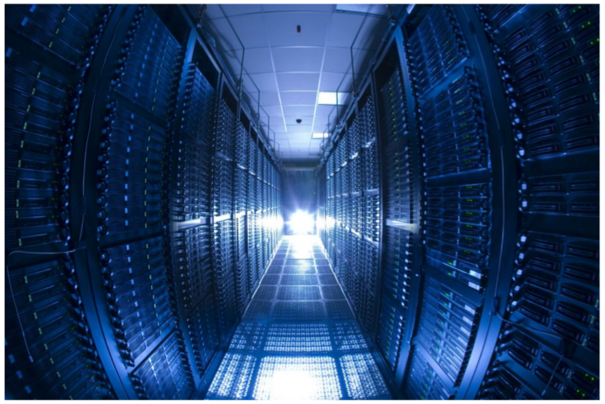
- *To demonstrate the operating principle of the ML FPGA, we propose to use the existing setup of the ongoing EIC detector R&D project (eRD22) "GEM based Transition radiation detector (TRD) and tracker".*
- *To test the "global PID" performance we plan to integrate the EIC calorimeter prototype (3x3 modules) into the ML-FPGA setup.*
- *Preprocessed data from both detectors including decision on the particle type will be transferred to another ML-FPGA board with neural network for global PID decision.*
- *Real beam testing is planned in Hall D, where there is already a test beam site that can be used for testing the prototype GEM-TRD, ECAL and, if possible, Modular RICH detectors: this is an important part of the project to evaluate a "global PID" .*
- *For the GEM-TRD project we already use offline Machine Learning tools (JETNET, ROOT-based TMVA), and the results can be used for validation of the proposed implementation of FPGA-based neural networks .*

Filter design proposal









Images from the Internet are for illustration of scale only.

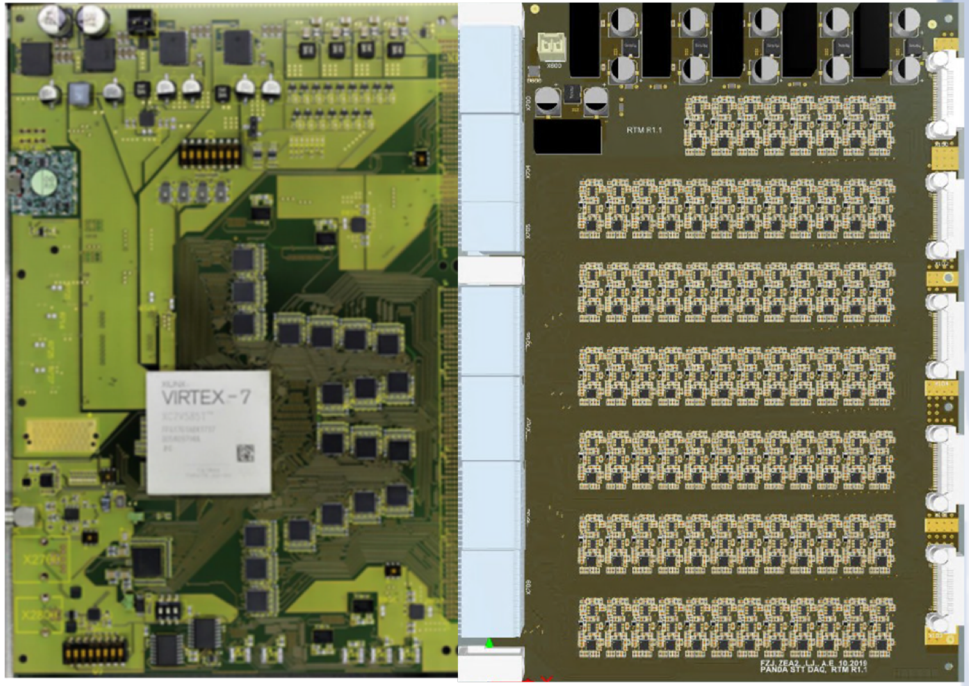
ADC based DAQ for PANDA STT

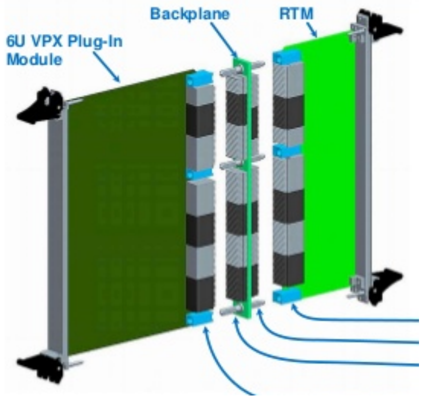
Jefferson Lab
Thomas Jefferson National Accelerator Facility

Level 0 Open VPX Crate

ADC based DAQ for PANDA STT (one of approaches):

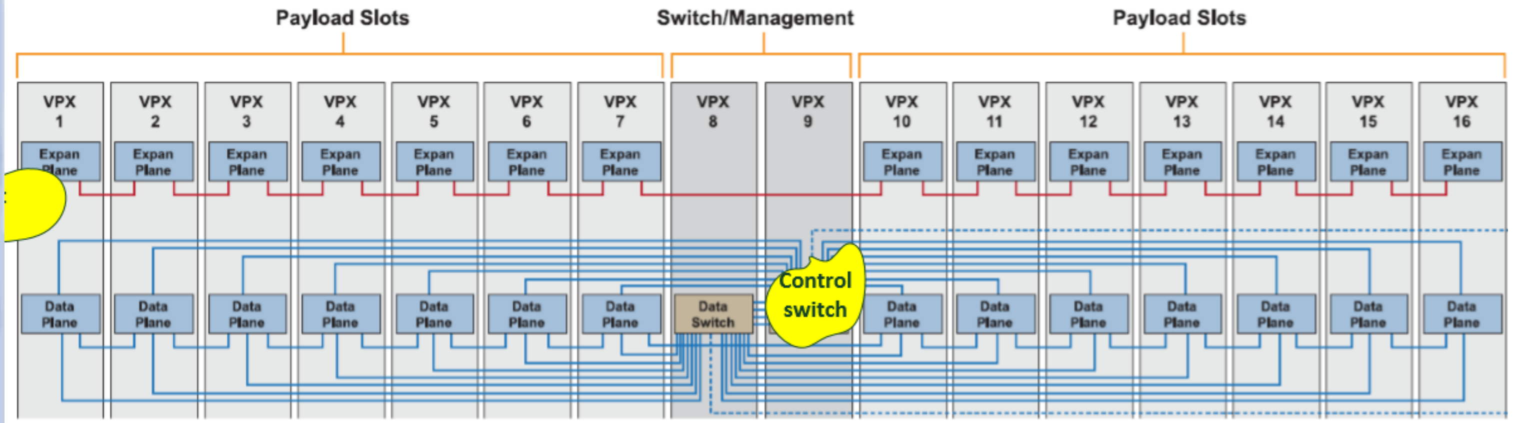
- 160 channels (shaping, sampling and processing) per payload slot, 14 payload slots+2 controllers;
- **totally 2200 channels per crate**;
- time sorted output data stream (arrival time, energy,...)
- noise rejection, pile up resolution, base line correction, ..





- 40 4-channel ADCs (configurable up to 1 GSPS);
- Single Virtex7 FPGA

- 160 Amplifiers;
- 5 connectors for 32-pins samtec cables



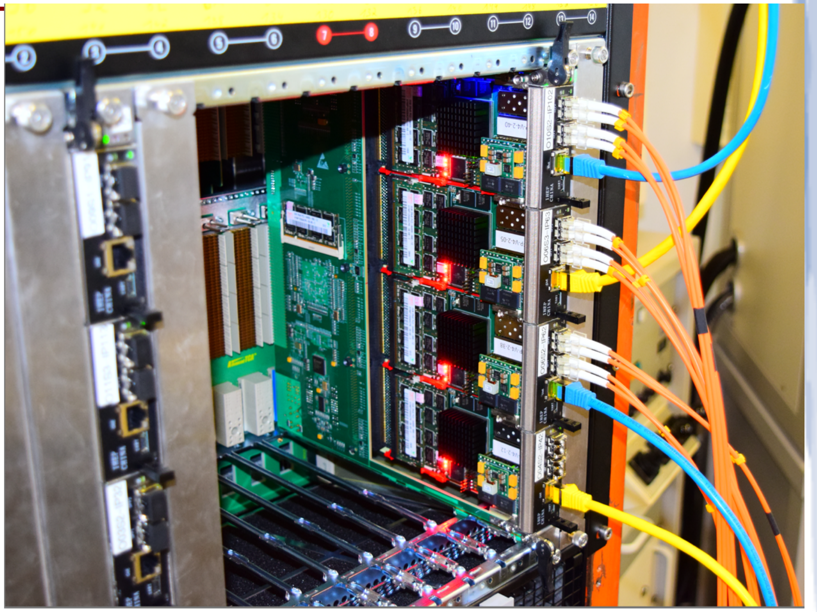
Powerful Backplane up to 670 GBs

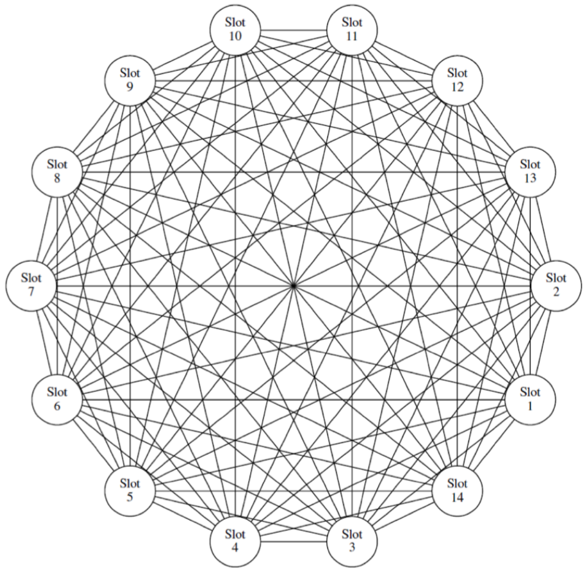
L. Jokhovets, P Kulessa ..

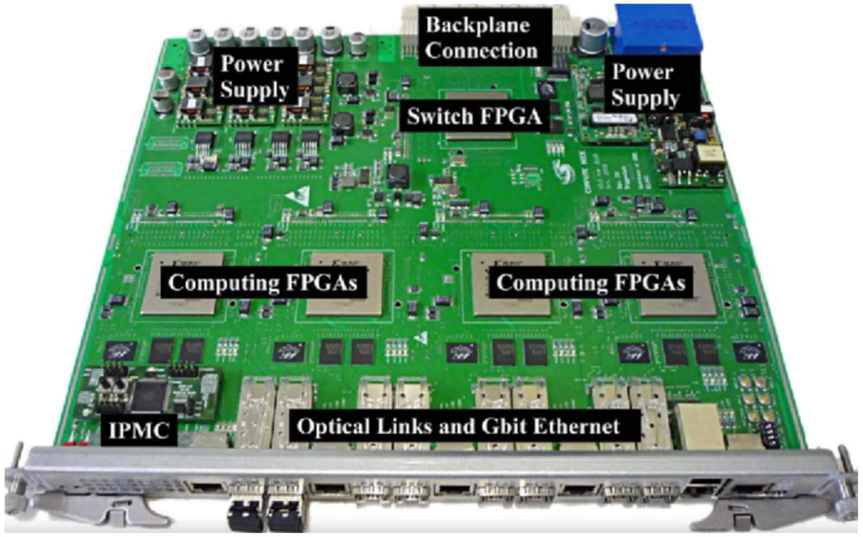
JÜLICH Forschungszentrum

Compute Node (PXD,Belle II)

- The pixel detector of Belle II with its ~ 8 million channels will deliver data at rate of 22 Gbytes/s for a trigger rate of 30 kHz
- A hardware platform capable of processing this amount of data is the ATCA based Compute Node. (Advanced Telecommunications Computing Architecture).
- A single ATCA crate can host up to 14 boards interconnected via a full mesh backplane.
- Each AMC board is equipped with 4 Xilinx Virtex-5 FX70T FPGA.







Outlook

- *An FPGA-based Neural Network application would offer online event preprocessing and allow for data reduction based on physics at the early stage of data processing.*
- *This can reduce the size of the high level trigger computer farm and data traffic.*
- *Open-source hls4ml software tool with Xilinx® Vivado® High Level Synthesis (HLS) accelerates machine learning neural network algorithm development.*
- *FPGA provides extremely low-latency neural-network inference on the order of 100 nanoseconds.*
- *The ultimate goal is to build a real-time event filter based on physics signatures.*

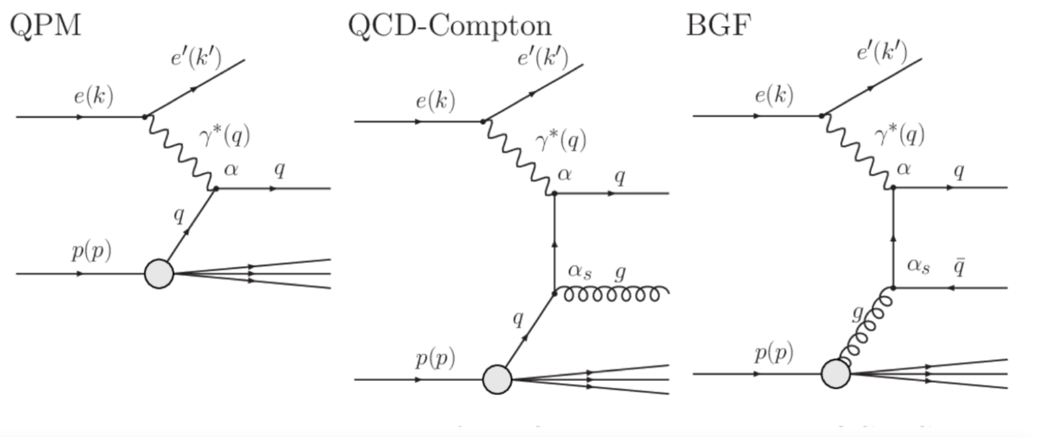


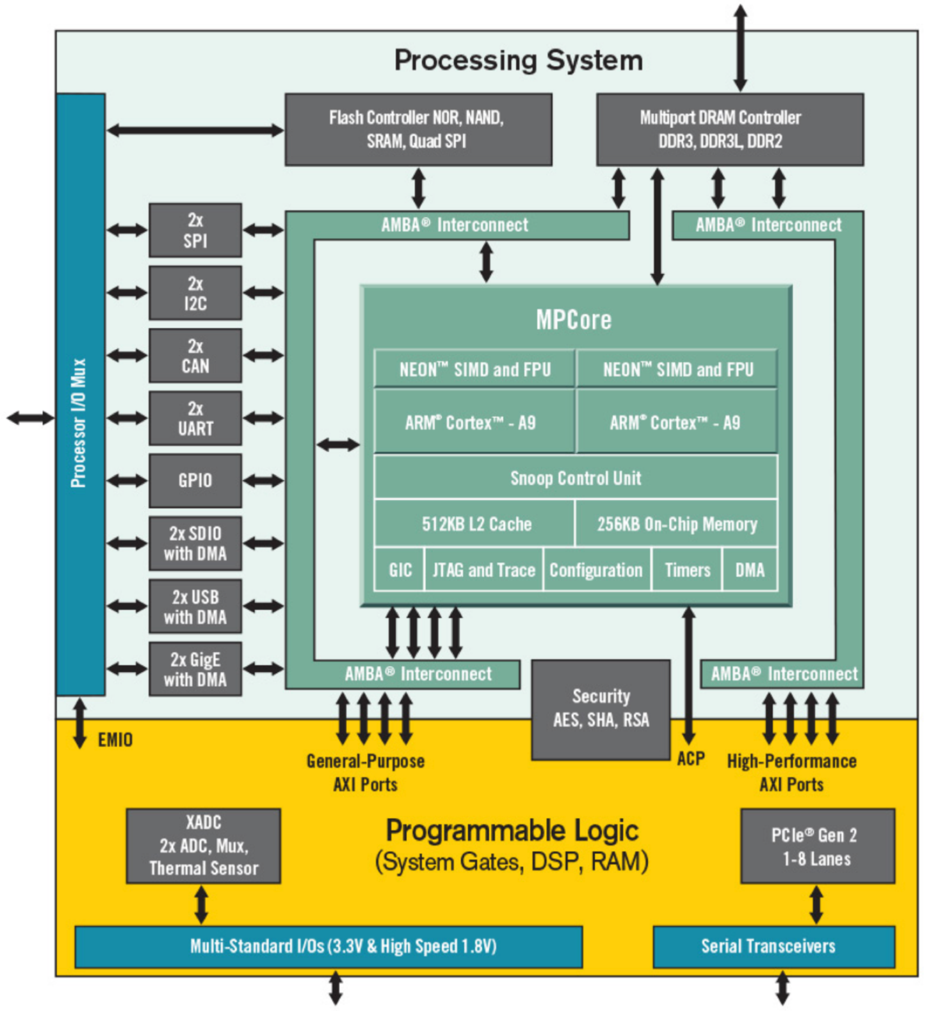
Figure 2.1: Feynman diagrams of the Quark Parton Model, QCD-Compton and Boson Gluon Fusion processes in NC DIS.

Published in 2007
Measurement of multijet events at low x_{Bj} and low Q^2 with the ZEUS detector at HERA
T. Gosau

Jefferson Lab
Thomas Jefferson National Accelerator Facility

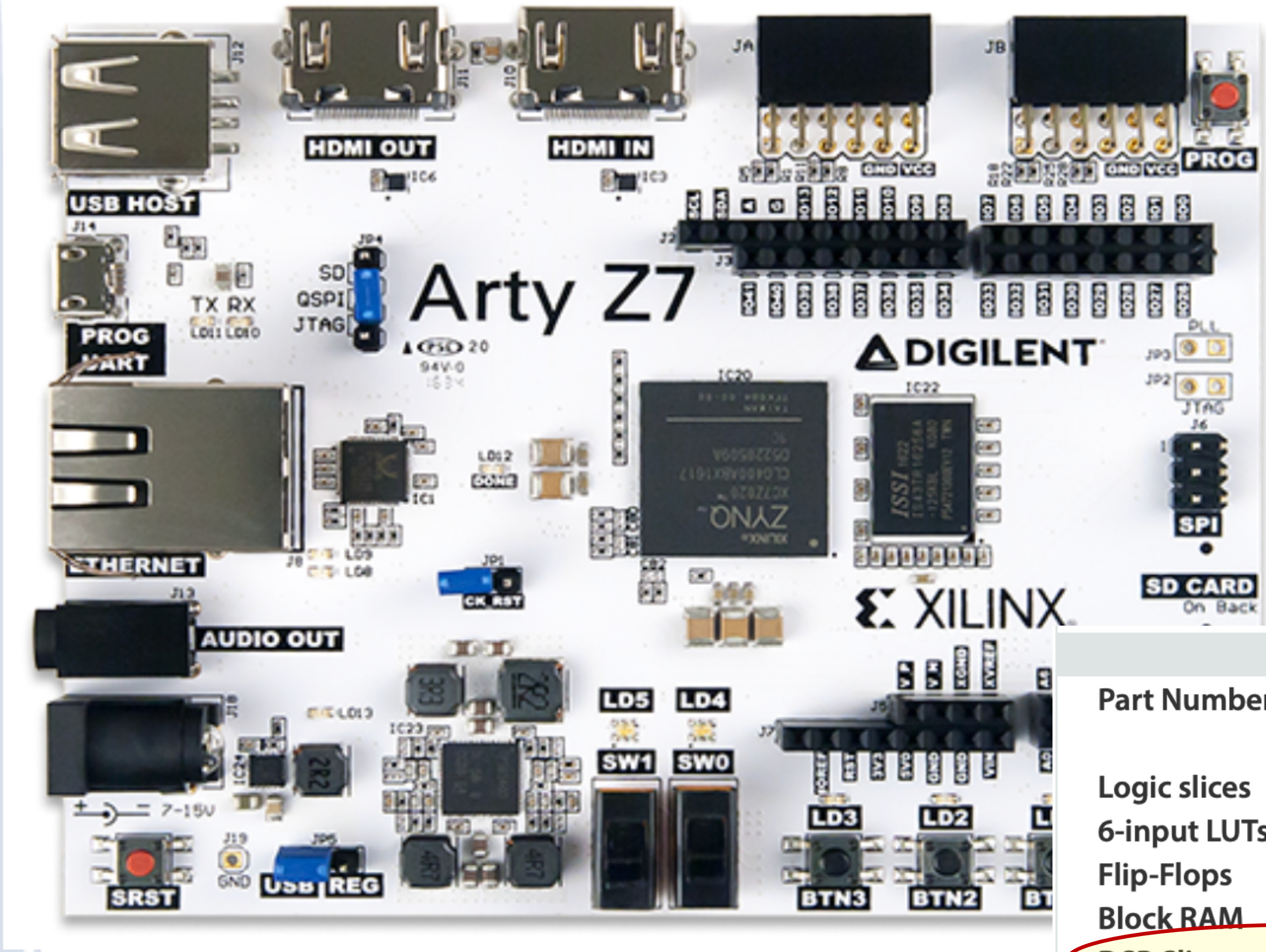
Backup

Xilinx Zynq 7000 architecture



Zynq-7000
Zynq-7000 devices are equipped with dual-core ARM Cortex-A9 processors integrated with 28nm Artix-7 or Kintex®-7 based programmable logic for excellent performance-per-watt and maximum design flexibility. With up to 6.6M logic cells and offered with transceivers ranging from 6.25Gb/s to 12.5Gb/s, Zynq-7000 devices enable highly differentiated designs for a wide range of embedded applications including multi-camera drivers assistance systems and 4K2K Ultra-HDTV.

Arty Z7 (Zynq-7000)



Key FPGA Specifications	
Part Number	XC7Z020-1CLG400C (XC7Z010-1CLG400C*)
Logic slices	13,300 (4,400*)
6-input LUTs	53,200 (17,600*)
Flip-Flops	106,400 (35,200*)
Block RAM	630 KB (270 KB*)
DSP Slices	220 (80*)
Clock Resources	Zynq PLL with 4 outputs 4 PLLs (2 PLLs*) 4 MMCMs (2 MMCMs*) 125 MHz external clock

Xilinx Virtex® UltraScale+™

Jefferson Lab
Thomas Jefferson National Accelerator Facility