Development of PS-PL interface workflows for Zynq MPSoC

Emily Smith, Tong Ou, Fukun Tang, David Miller

Work done in part within the gFEX trigger system for ATLAS Phase I upgrade of the calorimeter trigger system

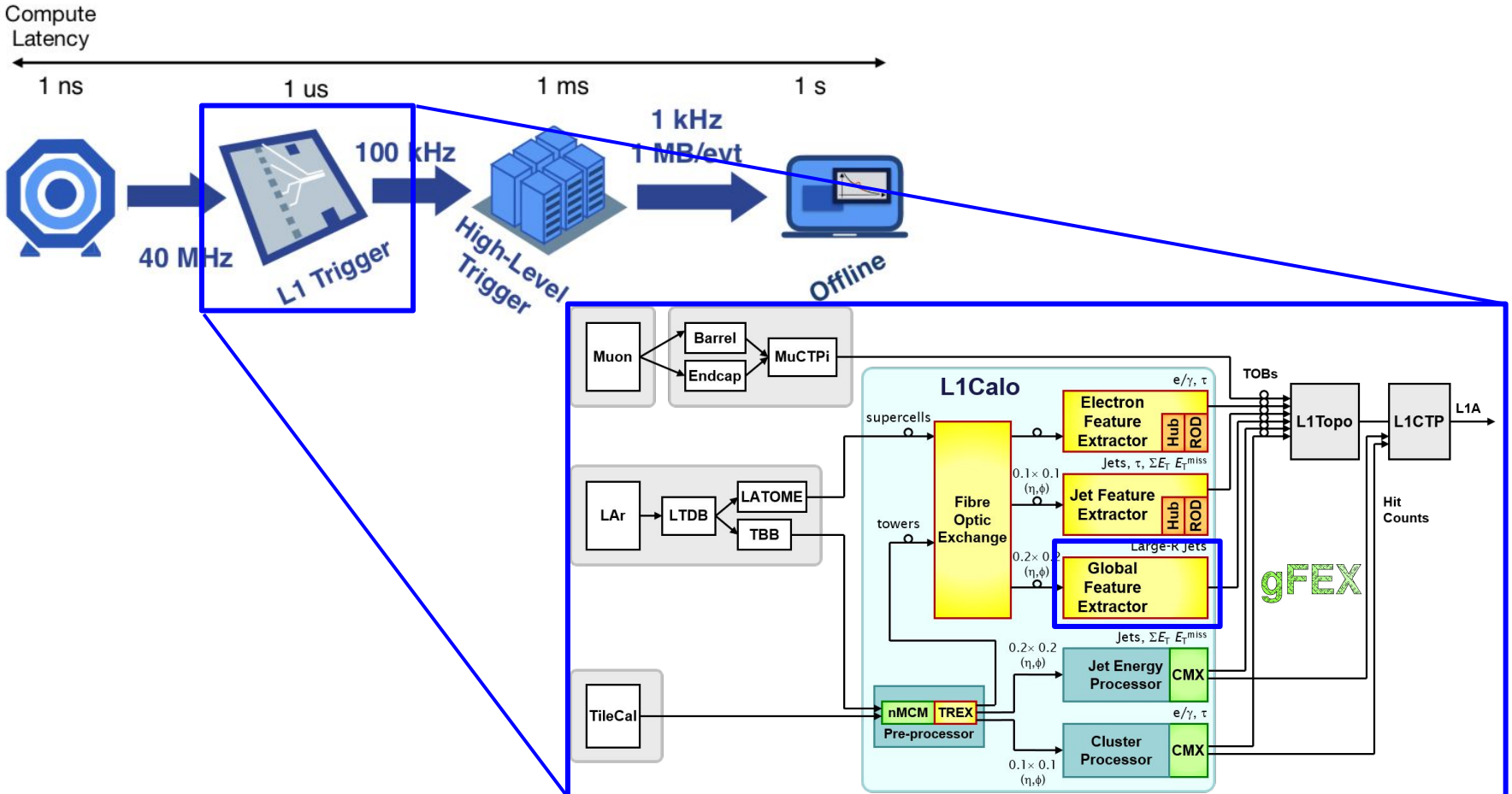
THE UNIVERSITY OF CHICAGO

ATLAS EXPERIMENT

Outline

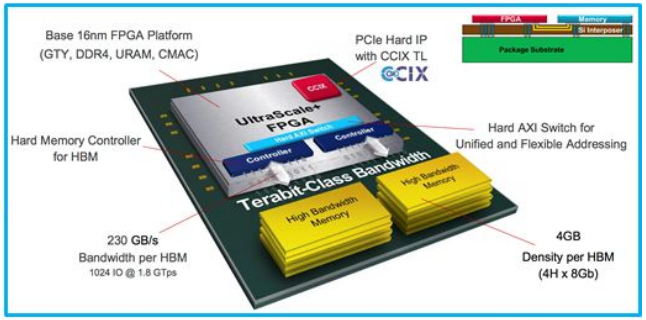
- gFEX use of Zynq+ Multi-Processor System-on-Chip (MPSoC)
- gFEX Operating System (OS) Development
- gFEX Processing System (PS) -- Programmable Logic (PL) Interfaces
- Machine Learning (ML) algorithms on Zynq & interactions with the PS

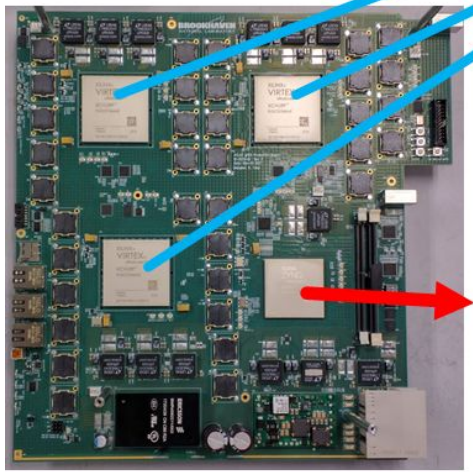
The ATLAS Calorimeter Trigger for LHC Run 3

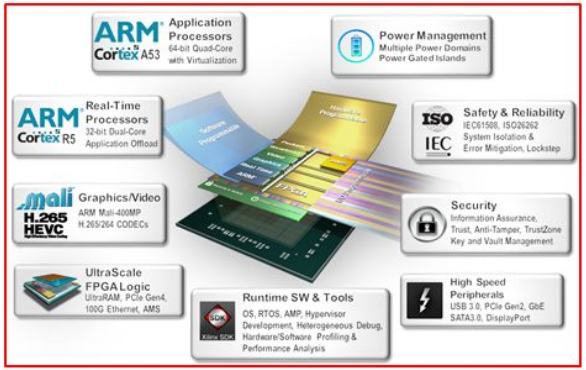


global Feature Extractor (gFEX)

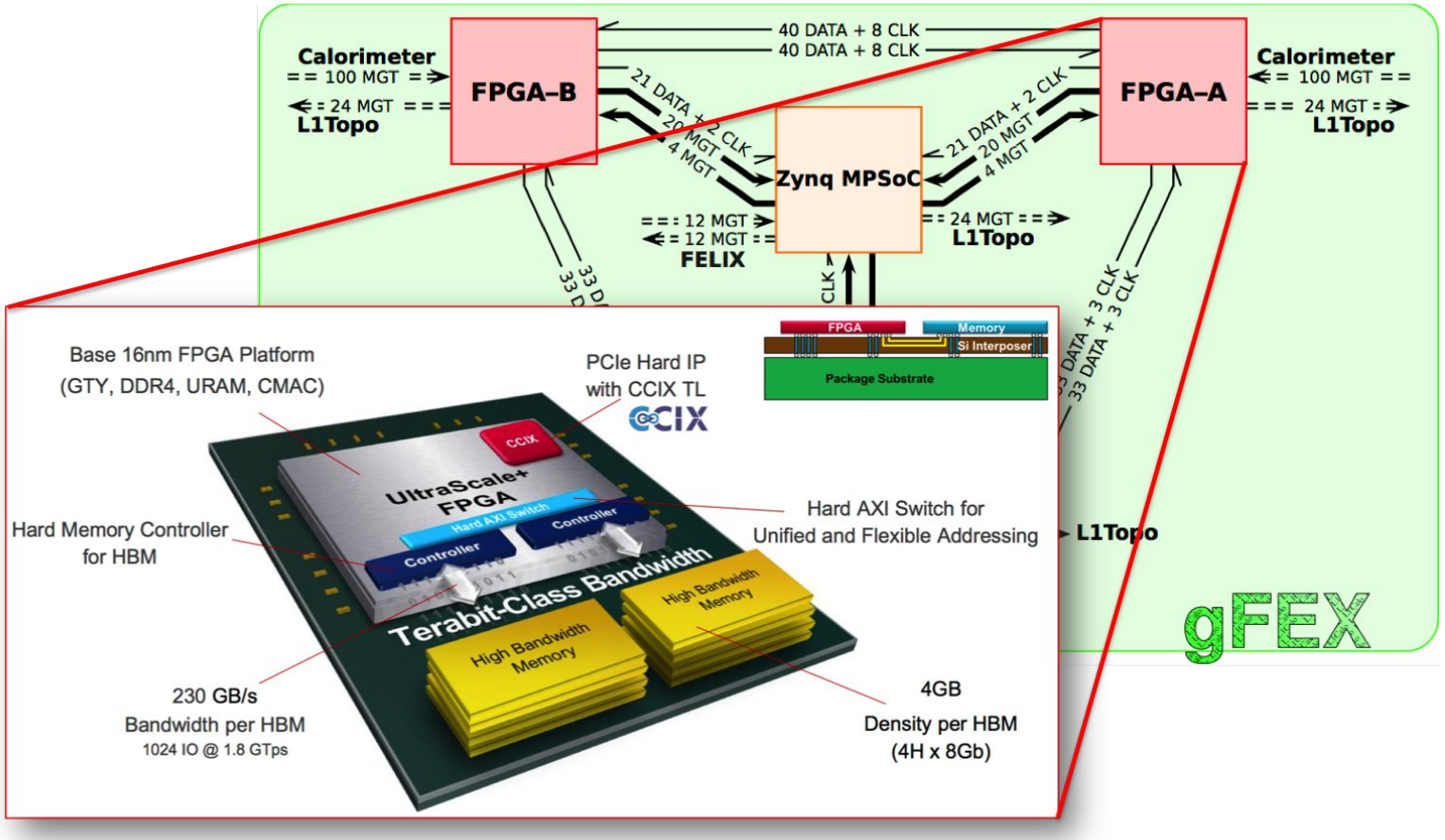
- Provide Large-radius jet triggers, MET, pile-up estimation, and other global observables for trigger
- 3 Ultrascale+ processor FPGAs (pFPGAs) covering different regions
- Zynq Ultrascale+ MPSoC for control and additional processes



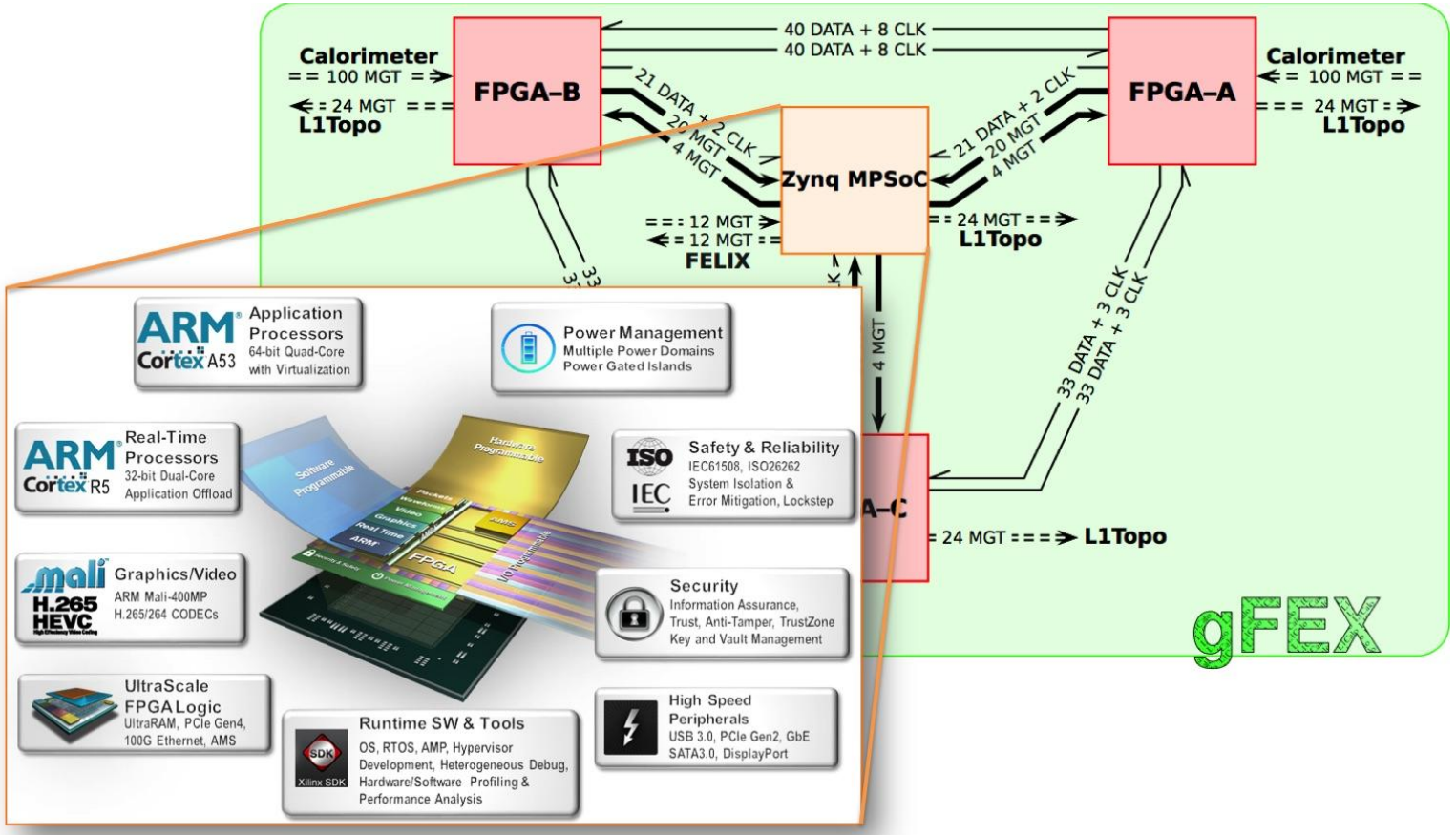




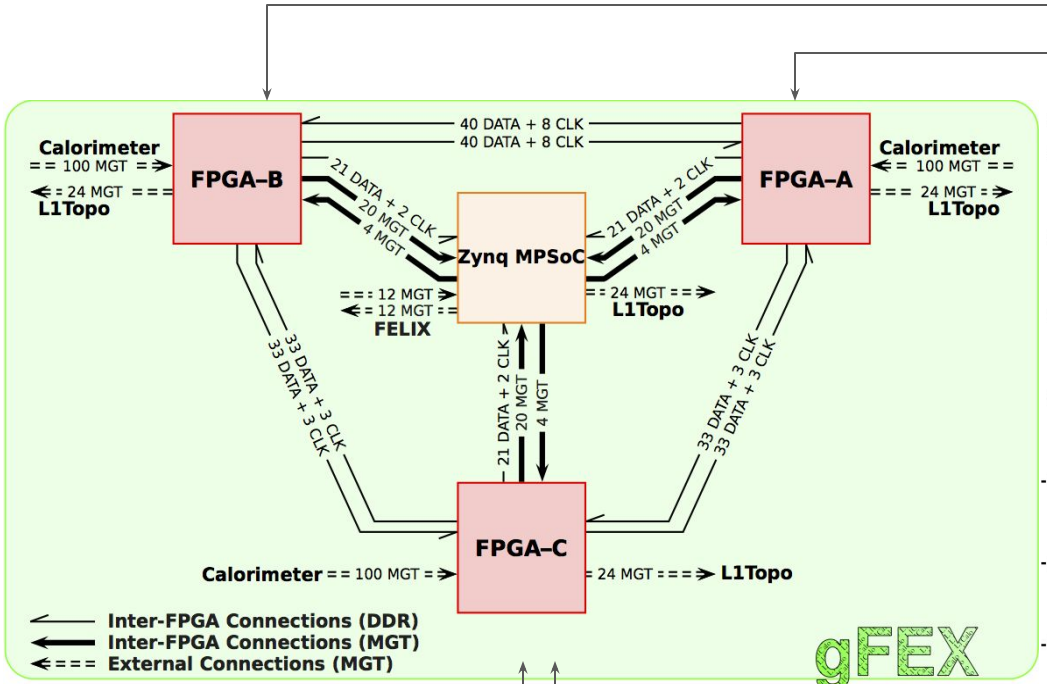
gFEX: processor FPGAs (Virtex UltraScale+ VU9P)

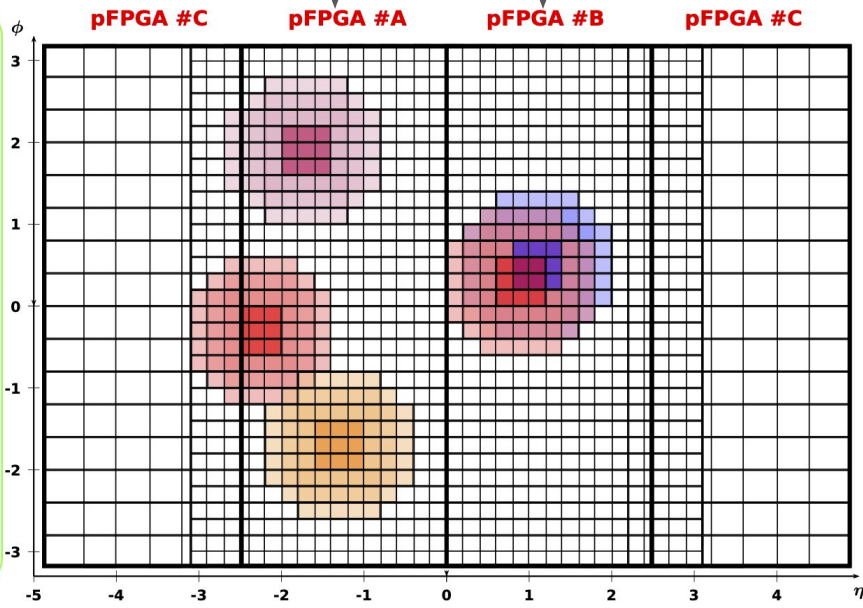


gFEX: Zynq MPSoC (UltraScale+ ZU19EG)



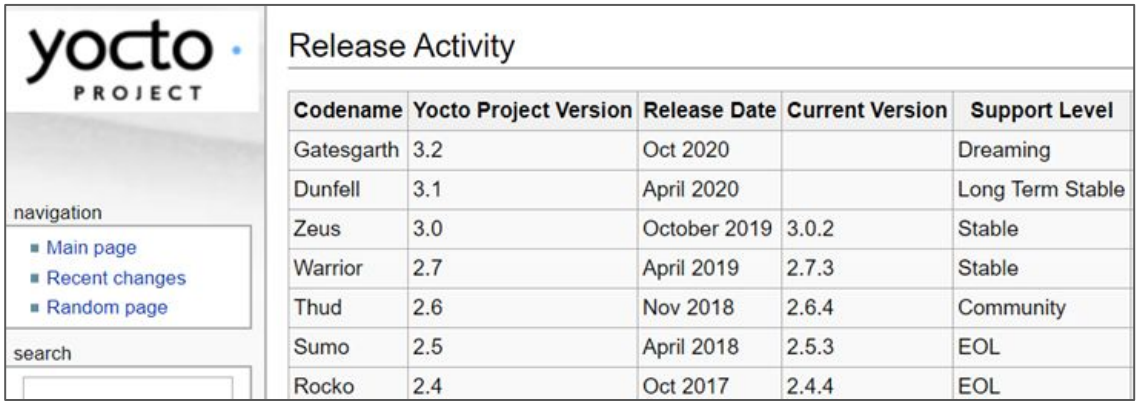
gFEX hardware view





gFEX Custom Operating System

- Custom operating system developed using the Yocto/Openembedded build engine and bitbake
- meta-l1calo custom layer for all our L1Calo needs (link)
- Previously were using Yocto release “rocko” upgrading to release “zeus”
  - Zeus offers python3 in addition to python 2
  - Still working on updating board definition files - lots of changes from Xilinx
- Recently added CI-flow for the OS build (see Michal’s talk later!)
- Plans to investigate centos as a possibility when personpower allows

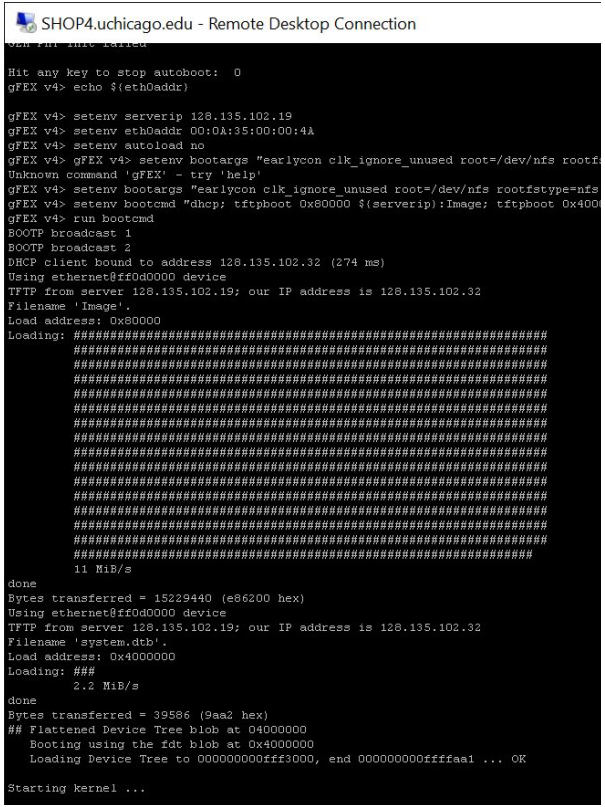


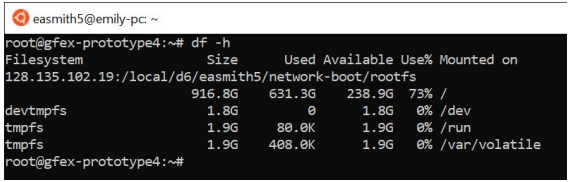
Release Activity

Codename	Yocto Project Version	Release Date	Current Version	Support Level
Gatesgarth	3.2	Oct 2020		Dreaming
Dunfell	3.1	April 2020		Long Term Stable
Zeus	3.0	October 2019	3.0.2	Stable
Warrior	2.7	April 2019	2.7.3	Stable
Thud	2.6	Nov 2018	2.6.4	Community
Sumo	2.5	April 2018	2.5.3	EOL
Rocko	2.4	Oct 2017	2.4.4	EOL

gFEX NetBoot with Zynq+ SoC

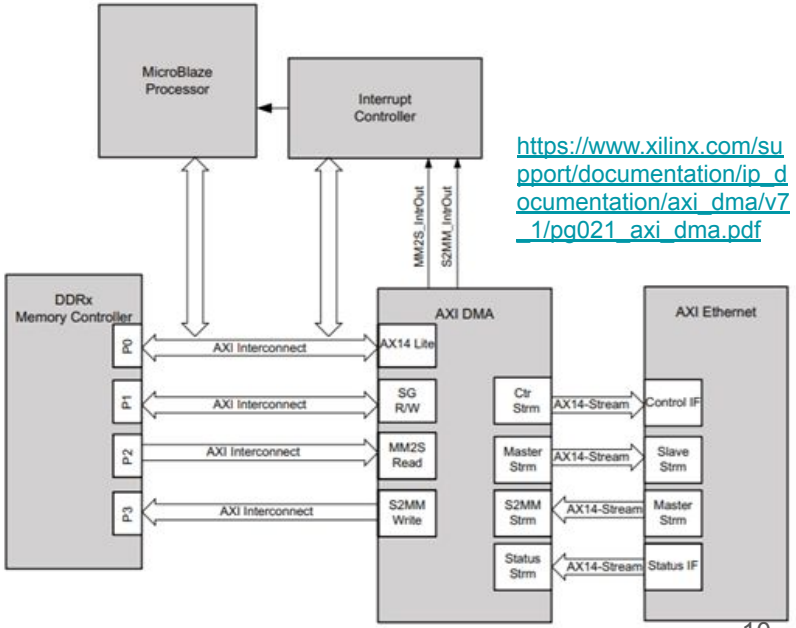
- Using First-Stage-Bootloader (FSBL) which
  - Loads bitstream
  - Configures the Zynq architecture processing system (PS) through u-boot
- Network Booting with:
  - TFTP server to fetch the kernel image, device tree, and bitfile
  - NFS server as the location of the rootfs (as opposed to a separate SD card partition)
- Allows for remote updating of the OS
- Implemented at Chicago and in progress in the STF at CERN





gFEX Monitoring & Control: Zynq+ SoC

- Primary task of the Zynq is to monitor the gFEX board and communicate info to central Detector Control System (DCS) and online monitoring framework
- Requires interaction between Zynq PS & PL
- Using fast DMA for PS-PL interfaces
  - Upgraded AXI-DMA vs AXI FIFO
  - Uses Xilinx provided IP for AXI Stream logic
  - Allows for faster transfer of data between FPGAs and the processing system
  - Requires driver & software on PS to implement

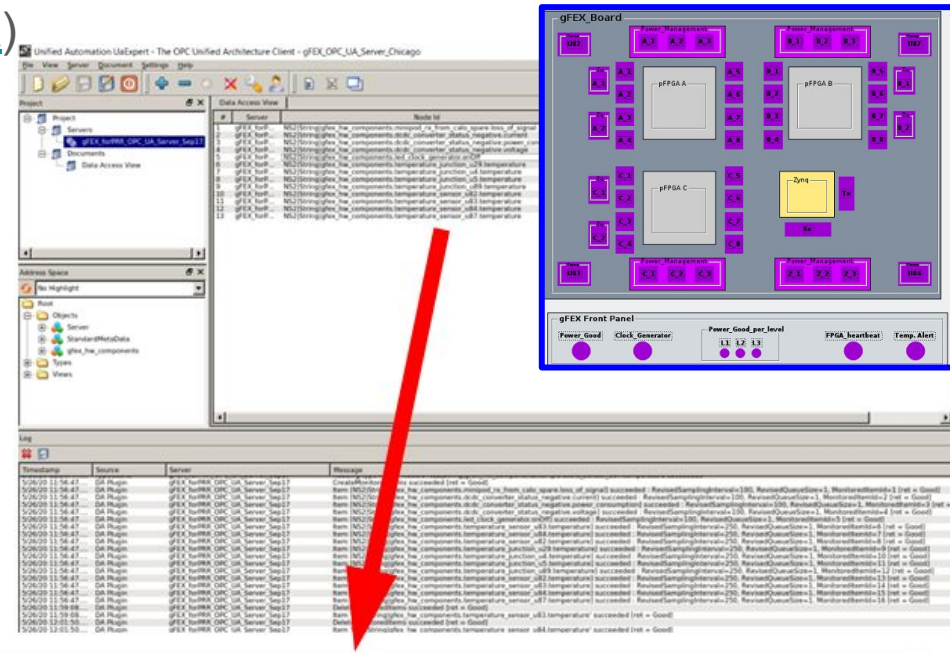


https://www.xilinx.com/support/documentation/ip_documentation/axi_dma/v7_1/pg021_axi_dma.pdf

Figure 3-1: **Typical MicroBlaze Processor System Configuration (AXI Ethernet)**

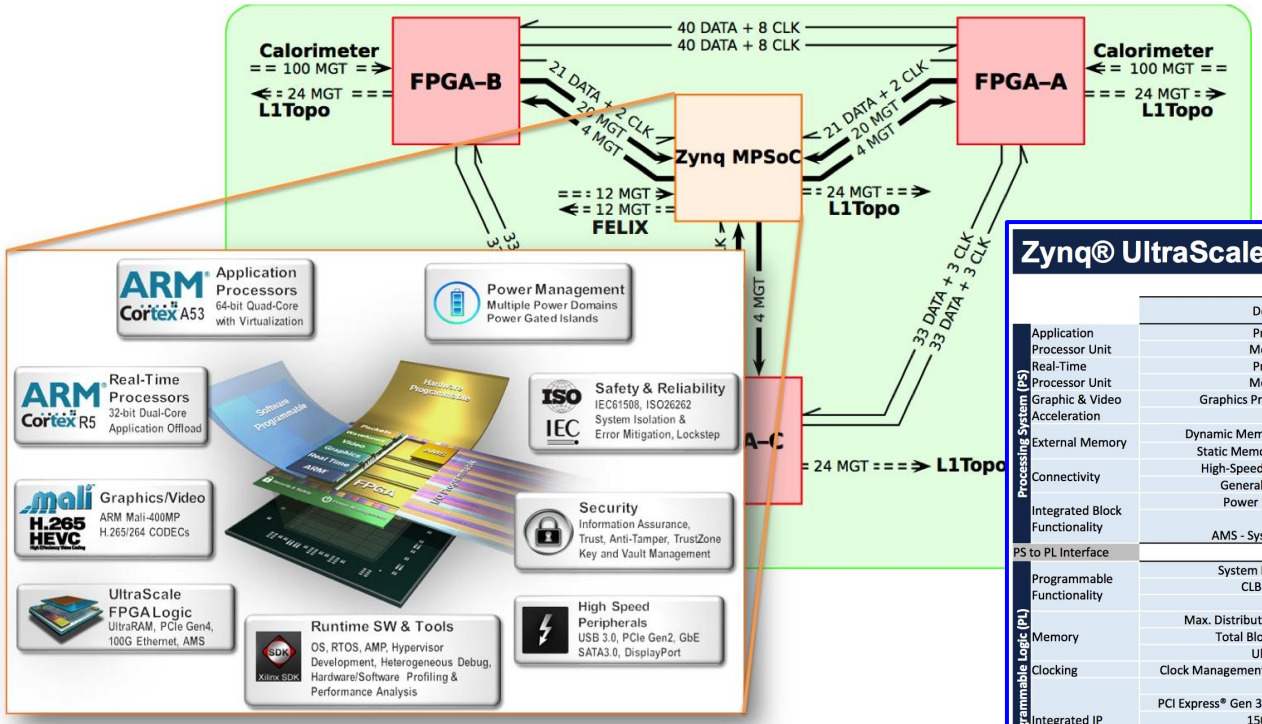
On-board Detector Control System (DCS) Server

- DCS Open Platform Communications Unified Architecture (OPC-UA) Server generated with quasar framework (link).
- gFEX OPC-UA server developed and integrated into the Yocto OS
  - Server built into python using Poverty module developed by quasar team
  - Integration into Yocto OS required the use of python3 for the Poverty module
- Updates to quasar version for gFEX OPC-UA project are in progress



gFEX_forP...	NS2|String|gfex_hw_components.temperature_junction_u89.temperature	temperature		Null
gFEX_forP...	NS2|String|gfex_hw_components.temperature_sensor_u82.temperature	temperature	21	Float
gFEX_forP...	NS2|String|gfex_hw_components.temperature_sensor_u83.temperature	temperature	20	Float
gFEX_forP...	NS2|String|gfex_hw_components.temperature_sensor_u84.temperature	temperature	20	Float
gFEX_forP...	NS2|String|gfex_hw_components.temperature_sensor_u87.temperature	temperature	25	Float

gFEX: Zynq MPSoC (UltraScale+ ZU19EG)



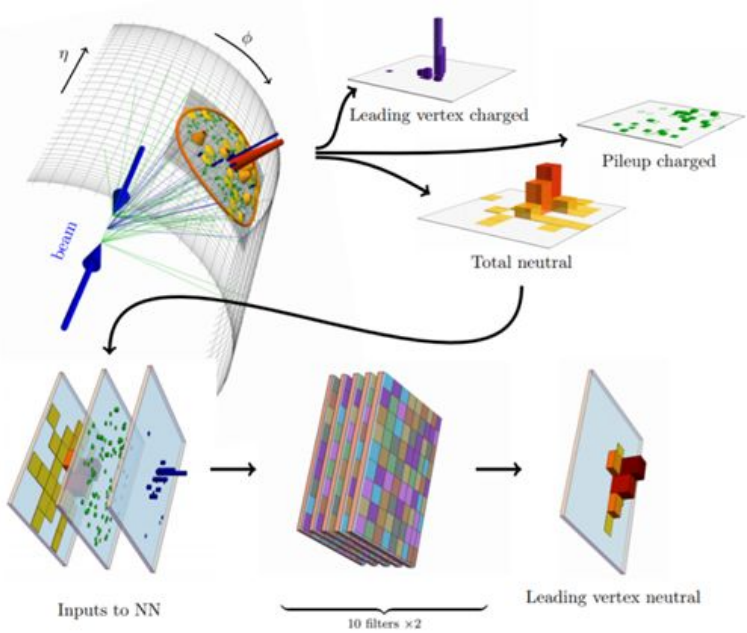
Zynq® UltraScale+™ MPSoCs: EG Devices

			Smarter Control and Vision					Smarter Network					
	Device Name[1]	ZU2EG	ZU3EG	ZU4EG	ZU5EG	ZU7EG	ZU6EG	ZU9EG	ZU15EG	ZU11EG	ZU17EG	ZU19EG
Processing System (PS)	Application Processor Unit	Processor Core	Quad-core ARM® Cortex™-A53 MPCore™ up to 1.5GHz										
		Memory w/ECC	L1 Cache 32KB I / D per core, L2 Cache 1MB, on-chip Memory 256KB										
	Real-Time Processor Unit	Processor Core	Dual-core ARM Cortex-R5 MPCore™ up to 600MHz										
		Memory w/ECC	L1 Cache 32KB I / D per core, Tightly Coupled Memory 128KB per core										
	Graphic & Video Acceleration	Graphics Processing Unit	Mali™-400 MP2 up to 667MHz										
		Memory	L2 Cache 64KB										
	External Memory	Dynamic Memory Interface	x32/x64: DDR4, LPDDR4, DDR3, DDR3L, LPDDR3 with ECC										
		Static Memory Interfaces	NAND, 2x Quad-SPI										
	Connectivity	High-Speed Connectivity	PCIe® Gen2 x4, 2x USB3.0, SATA 3.1, DisplayPort, 4x Tri-mode Gigabit Etherne										
		General Connectivity	2xUSB 2.0, 2x SD/SDIO, 2x UART, 2x CAN 2.0B, 2x I2C, 2x SPI, 4x 32b GPIO										
	Integrated Block Functionality	Power Management	Full / Low / PL / Battery Power Domains										
		Security	RSA, AES, and SHA										
		AMS - System Monitor	10-bit, 1MSPS - Temperature, Voltage, and Current Monitor										
PS to PL Interface			12 x 32/64/128b AXI Ports										
Programmable Logic (PL)	Programmable Functionality	System Logic Cells (K)	103	154	192	256	504	469	600	747	653	926	1,143
		CLB Flip-Flops (K)	94	141	176	234	461	429	548	682	597	847	1,045
		CLB LUTs (K)	47	71	88	117	230	215	274	341	299	423	523
	Memory	Max. Distributed RAM (Mb)	1.2	1.8	2.6	3.5	6.2	6.9	8.8	11.3	9.1	8.0	9.8
		Total Block RAM (Mb)	5.3	7.6	4.5	5.1	11.0	25.1	32.1	26.2	21.1	28.0	34.6
		UltraRAM (Mb)	-	-	14.0	18.0	27.0	-	-	31.5	22.5	28.7	36.0
	Clocking	Clock Management Tiles (CMTs)	3	3	4	4	8	4	4	4	8	11	11
	Integrated IP	DSP Slices	240	360	728	1,056	1,728	1,973	2,520	3,528	2,928	1,590	1,968
		PCI Express® Gen 3x16 / Gen4x8	-	-	2	2	2	-	-	-	4	4	5
		150G Interlaken	-	-	-	-	-	-	-	-	1	2	4
		100G Ethernet MAC/PCS w/RS-FEC	-	-	-	-	-	-	-	-	2	2	4
		AMS - System Monitor	1	1	1	1	1	1	1	1	1	1	1
	Speed Grades	Extended[2]	-1 -2L -3					-1 -2L -3					
		Industrial	-1 -1L -2					-1 -1L -2					

Notes:
1. For full part number details, see the Ordering Information section in DS891, *Zynq UltraScale+ MPSoC Overview*.
2. -2LE (Tj = 0°C to 110°C). For more details, see the Ordering Information section in DS891, *Zynq UltraScale+ MPSoC Overview*.

Zynq+ FPGA Utilization in gFEX context

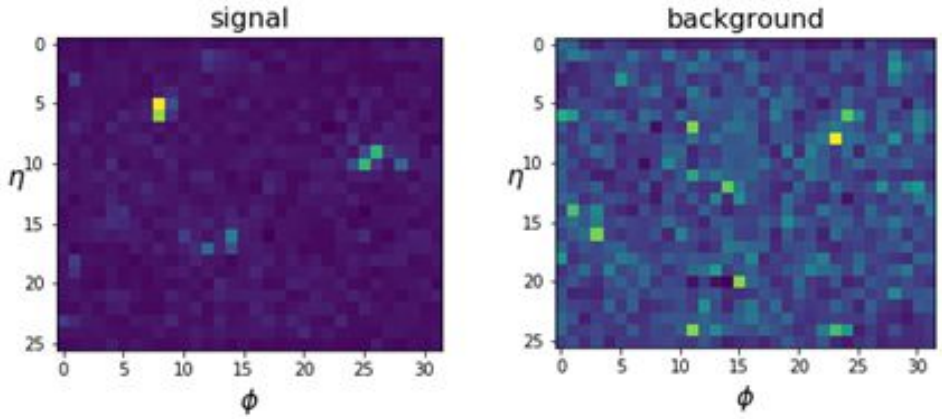
- **Under utilized!**
- Additional tasks under consideration:
  - On-board high-level monitoring (histogramming, data curation, etc)
  - ***Real-time processing and analysis***
- Calorimeter is particularly well suited for representation in the form of “images”
  - e.g. calorimeter trigger information often transmitted as pixelated energy
  - Convolutional Neural Networks (CNN) provide a well-studied and appropriate machine learning (ML) approach to classifying images of this form



Patrick T. Komiske, Eric M. Metodiev, Benjamin Nachman, Matthew D. Schwartz: arXiv:1707.08600

Calorimeter Trigger Event Classification

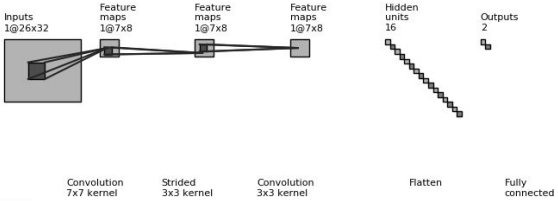
- Calorimeter energy deposits in η-ϕ space create a 26 x 32 pixel “image”
- Use a convolutional neural network (CNN) to classify each “image” (event) as either signal or background
- Implementation with Vivado High-Level Synthesis (HLS) tools

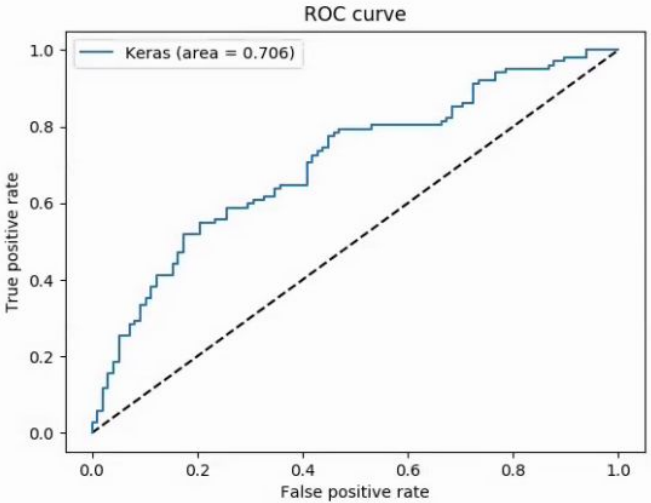


Example: Convolutional Neural Network

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	(None, 26, 32, 1)	0
conv2d_1 (Conv2D)	(None, 7, 8, 1)	26
conv2d_2 (Conv2D)	(None, 7, 8, 1)	10
conv2d_3 (Conv2D)	(None, 7, 8, 1)	10
flatten_1 (Flatten)	(None, 56)	0
dense_1 (Dense)	(None, 16)	912
dense_2 (Dense)	(None, 2)	34

Total params: 992
Trainable params: 992
Non-trainable params: 0

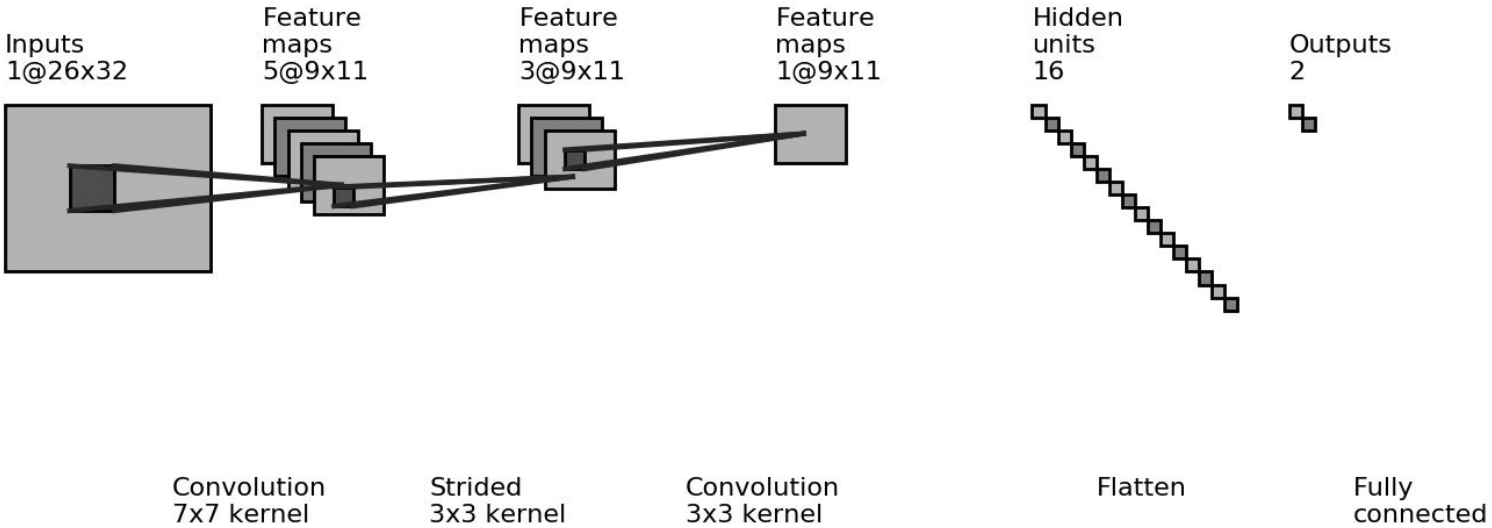




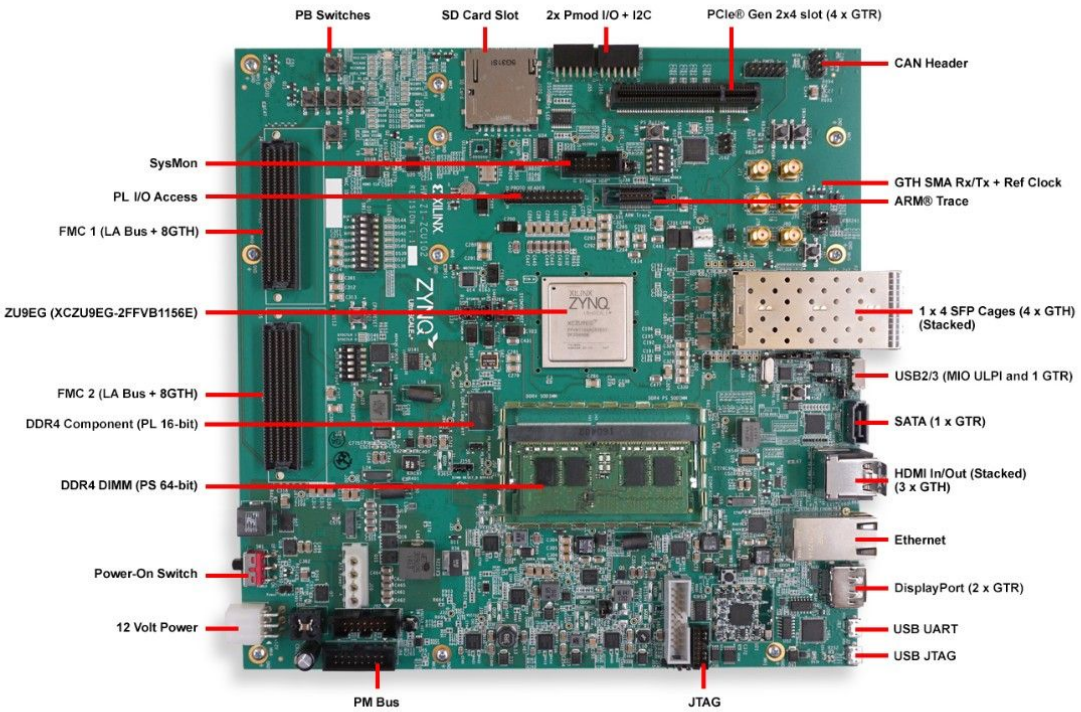
Benchmarked with process H → bb

Multi-filter Convolutional Neural Network

- Neural network with multiple filters for each convolutional layer is powerful in signal identification.



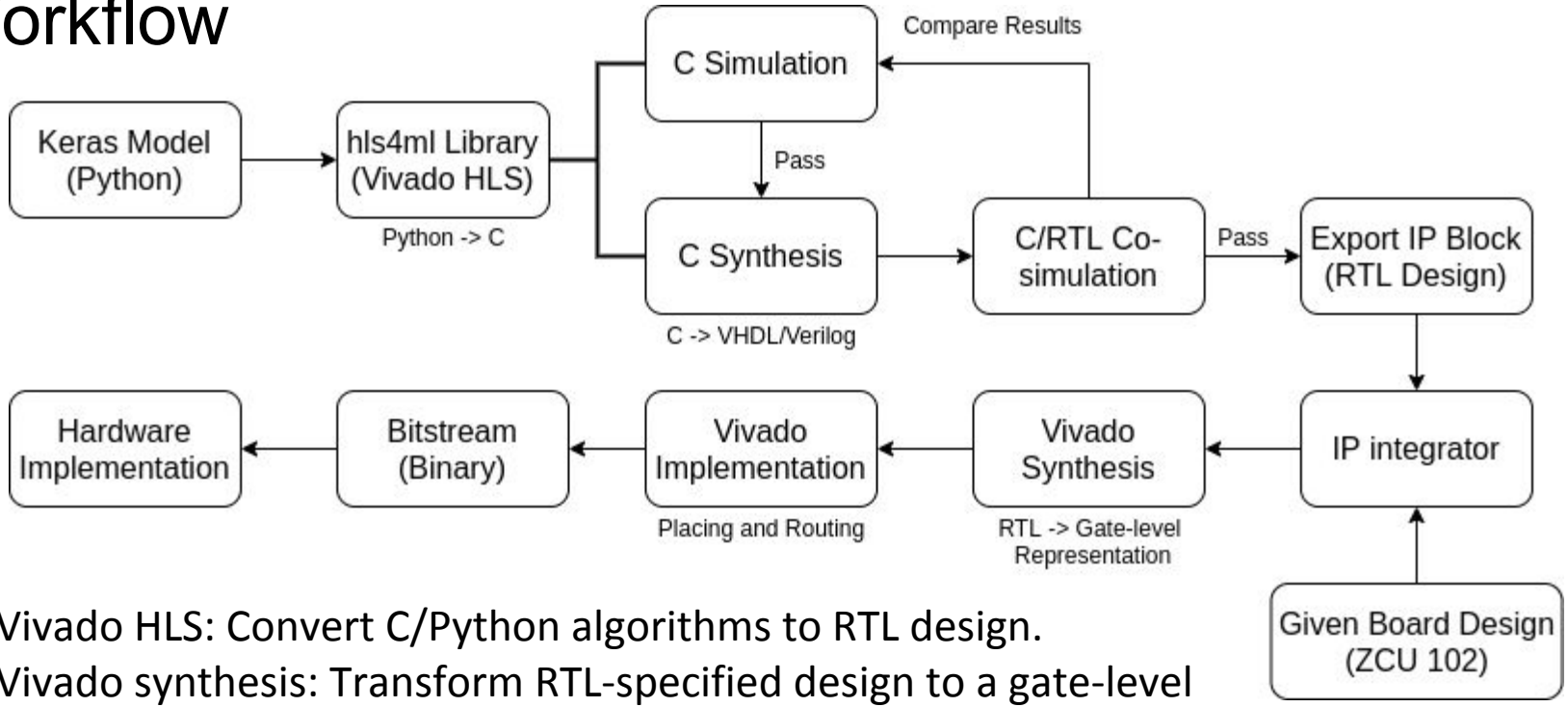
Zynq UltraScale+ MP SoC ZCU102 *(ZU9EG Zynq)*



Embedded environment combines power of CPU and FPGA.

- CPU: Optimized for rapid text switching. Excels at performing varied tasks.
- FPGA: Emulate digital logic at speeds analogous to hard-wired circuit. Excels at performing repetitive tasks.

Workflow



- Vivado HLS: Convert C/Python algorithms to RTL design.
- Vivado synthesis: Transform RTL-specified design to a gate-level representation.
- Vivado implementation: Placing and routing the netlist onto device resources.
- In this project, BOOT.BIN for hardware implementation is generated by Petalinux.

C synthesis report

- C synthesis is the process of translating C program to RTL design formatted in HDL (VHDL/Verilog).

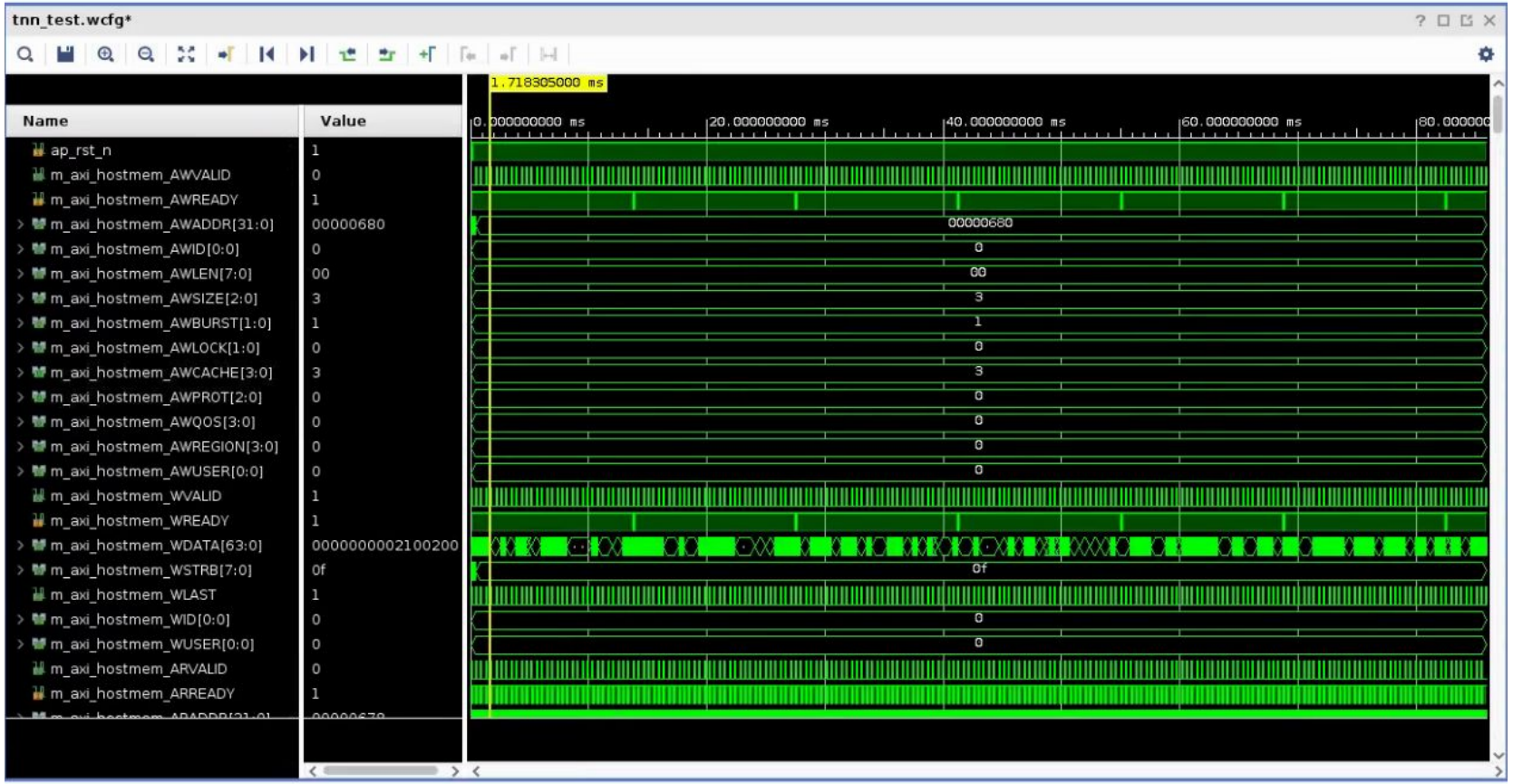
```
================================================================
== Performance Estimates
================================================================
+ Timing:
    * Summary:
    +--------+---------+----------+------------+
    |  Clock |  Target | Estimated| Uncertainty|
    +--------+---------+----------+------------+
    |ap_clk  | 5.00 ns | 5.702 ns |   0.62 ns  |
    +--------+---------+----------+------------+

+ Latency:
    * Summary:
    +---------+---------+----------+----------+-------+-------+---------+
    |  Latency (cycles) |  Latency (absolute) |    Interval   | Pipeline|
    |   min   |   max   |    min   |    max   |  min  |  max  |   Type  |
    +---------+---------+----------+----------+-------+-------+---------+
    |    41768|    41768| 0.238 ms | 0.238 ms |  41768|  41768|   none  |
    +---------+---------+----------+----------+-------+-------+---------+
```

```
================================================================
== Utilization Estimates
================================================================
* Summary:
+-----------------+---------+-------+--------+--------+-----+
|       Name      | BRAM_18K| DSP48E|   FF   |   LUT  | URAM|
+-----------------+---------+-------+--------+--------+-----+
|DSP              |        -|      -|       -|       -|    -|
|Expression       |        -|      -|       -|       -|    -|
|FIFO             |        -|      -|       -|       -|    -|
|Instance         |      270|    838|   43089|   89157|    0|
|Memory           |        -|      -|       -|       -|    -|
|Multiplexer      |        -|      -|       -|     201|    -|
|Register         |        -|      -|    3037|       -|    -|
+-----------------+---------+-------+--------+--------+-----+
|Total            |      270|    838|   46126|   89358|    0|
+-----------------+---------+-------+--------+--------+-----+
|Available        |     1824|   2520|  548160|  274080|    0|
+-----------------+---------+-------+--------+--------+-----+
|Utilization (%)  |       14|     33|       8|      32|    0|
+-----------------+---------+-------+--------+--------+-----+
```

C/RTL co-simulation

- C/RTL co-simulation is useful for validating the RTL design got from C synthesis.
- It uses the C testbench to generate input stimuli, with which an RTL testbench is generated and RTL simulation is executed.
- The output from the RTL simulation is re-applied to the C test bench to check the results.

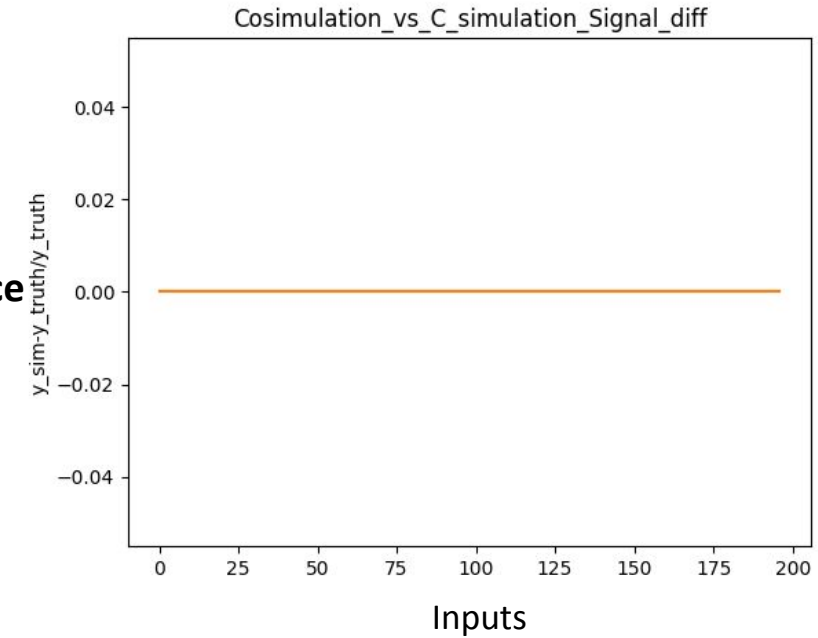
RTL simulation waveforms



Compare with C simulation results

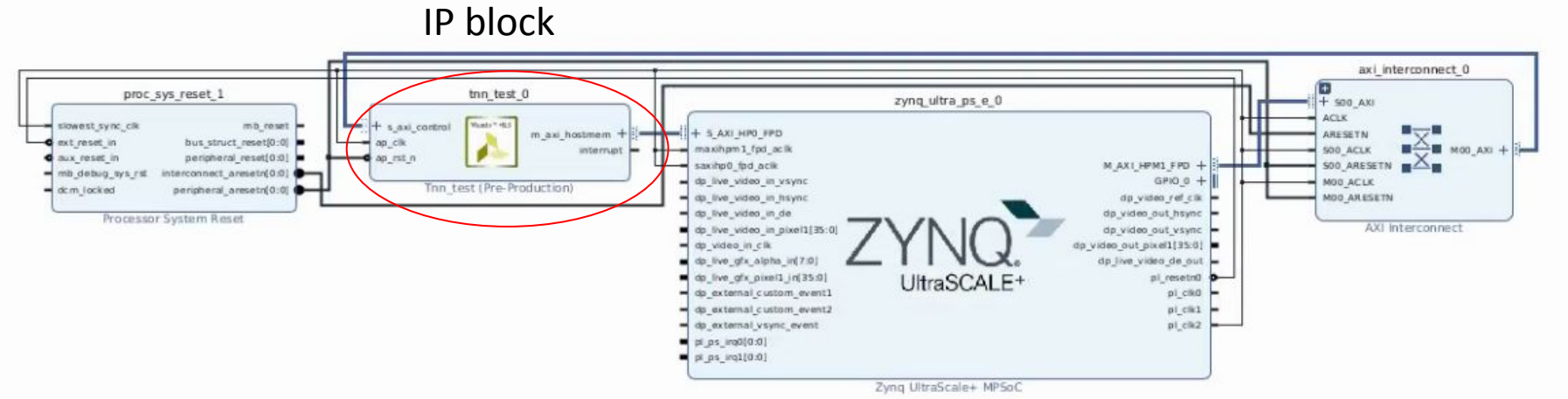
- C/RTL co-simulation results are compared with C simulation results. The RTL design is valid if they are consistent.

Relative difference



IP integration

- The IP block is exported from HLS and integrated into a given board design (ZCU102). The entire board design is shown below.



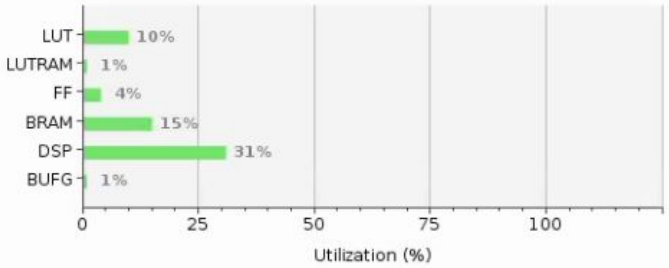
IP block is connected to other blocks via axi interfaces.

Implementation reports

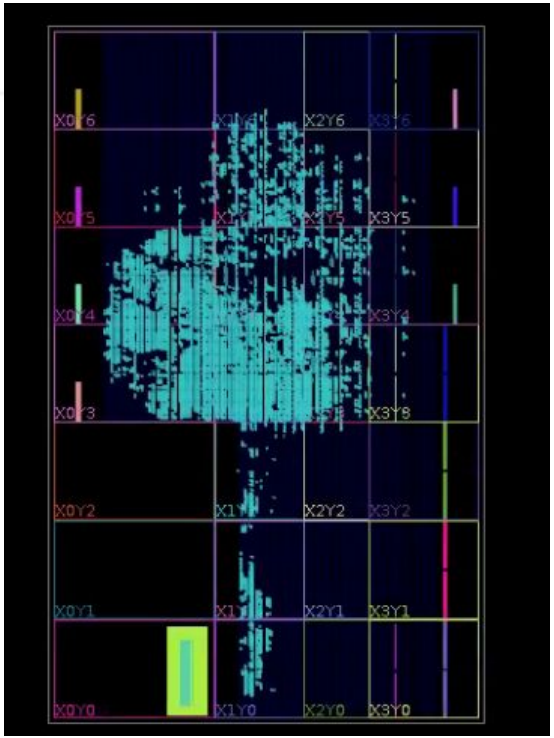
- Resources utilization

Summary

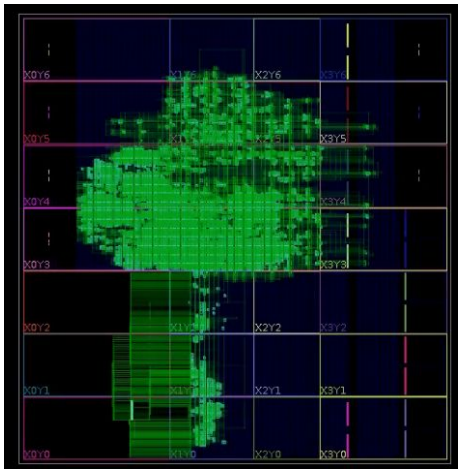
Resource	Utilization	Available	Utilization %
LUT	28767	274080	10.50
LUTRAM	702	144000	0.49
FF	21293	548160	3.88
BRAM	136.50	912	14.97
DSP	787	2520	31.23
BUFG	1	404	0.25



- Placement results



- Routing resources



Conclusions

- Systems on Chips are extremely powerful devices
- Our implementation of the Zynq MPSoC has allowed us to do many things
  - Run an OS
  - Control & Monitoring
- Have many options for increased usage
  - Increase in availability and dynamicity of monitoring
  - Additional algorithms
- Convolutional Neural Network demonstrated
  - To be seen what the final implementations and applications are
  - “Parasitic” or Level 1.5 trigger? Monitoring capabilities?

Backup

Synthesis and implementation reports

- Synthesis is the process of transforming an RTL-specified design into a gate-level

Design Timing Summary

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	0.798 ns	Worst Hold Slack (WHS):	-0.216 ns	Worst Pulse Width Slack (WPWS):	1.166 ns
Total Negative Slack (TNS):	0.000 ns	Total Hold Slack (THS):	-265.270 ns	Total Pulse Width Negative Slack (TPWS):	0.000 ns
Number of Failing Endpoints:	0	Number of Failing Endpoints:	3240	Number of Failing Endpoints:	0
Total Number of Endpoints:	53250	Total Number of Endpoints:	53250	Total Number of Endpoints:	23024

Timing constraints are not met.

- Implementation is the process of placing and routing the netlist onto device resources, within the logical, physical, and timing constraints of the design (optimizations occurring here make timing constraints met)

Design Timing Summary

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	0.186 ns	Worst Hold Slack (WHS):	0.010 ns	Worst Pulse Width Slack (WPWS):	1.167 ns
Total Negative Slack (TNS):	0.000 ns	Total Hold Slack (THS):	0.000 ns	Total Pulse Width Negative Slack (TPWS):	0.000 ns
Number of Failing Endpoints:	0	Number of Failing Endpoints:	0	Number of Failing Endpoints:	0
Total Number of Endpoints:	52074	Total Number of Endpoints:	52074	Total Number of Endpoints:	22497

All user specified timing constraints are met.