Traineeships in Advanced Computing for High Energy Physics (TAC-HEP)

FPGA module training

Lecture-11: April 14th 2026

Varun Sharma

University of Wisconsin – Madison, USA

Content

So far

- Motivation & Introduction
- Comparison: FPGAs/ASICs/GPU/CPU
- Domain specific Accelerators
- HLS setup and first example project
- Unsupported C/C++ constructs
- Data types
- HLS Pragmas
  - Interface
  - Array partition
  - Array Reshape
  - Pipeline
  - DataFlow
  - Allocation
  - Latency
  - Inline
  - Stable
  - Loop Unrolling
  - Loop_flatten
  - Loop_merge
  - Loop_tripcount

Today

- Project discussion

- You may choose to undertake the project individually or collaborate in a group of up to three or four members
- You are free to select one of the proposed projects
- **Alternatively**, you are welcome to propose your own idea

Project-1

Calorimeter Level-1 trigger system in real HEP experiment

Design and implement a **real-time calorimeter trigger pipeline** that processes streaming detector energy deposits and produces **physics trigger objects**.

1. receives calorimeter tower data,
2. performs local clustering,
3. sorts high-energy regions,
4. generates physics candidates,
5. Send out sorted physics candidates

The emphasis is on:

- **algorithm design**
- **Pragma Usage**
- **latency optimization**
- **Parallelization**

Project-1: Ingredients

1. *Input Data*

- Calorimeters (ECAL + HCAL) produces a grid of energy deposits arrange in η and ϕ.
  - E(η, ϕ)
- Use a grid of at least 30x60 in η x ϕ
- Each cel have integer ET with 10—12 bits worth of information per cell

2. *Clustering Algorithm*

- Implement a **sliding window** clustering:
  - Identify **local maxima** in the calorimeter grid.
  - Build **clusters (3x3, 5x5, 7x7)** by summing nearby cells – why would you need different sizes?
  - Thresholds for seed and sum
  - Keep the seed positions

3. *Sorting (try different algorithm)*

- Sort clusters by **transverse energy (ET)** or position
- Output top-N clusters

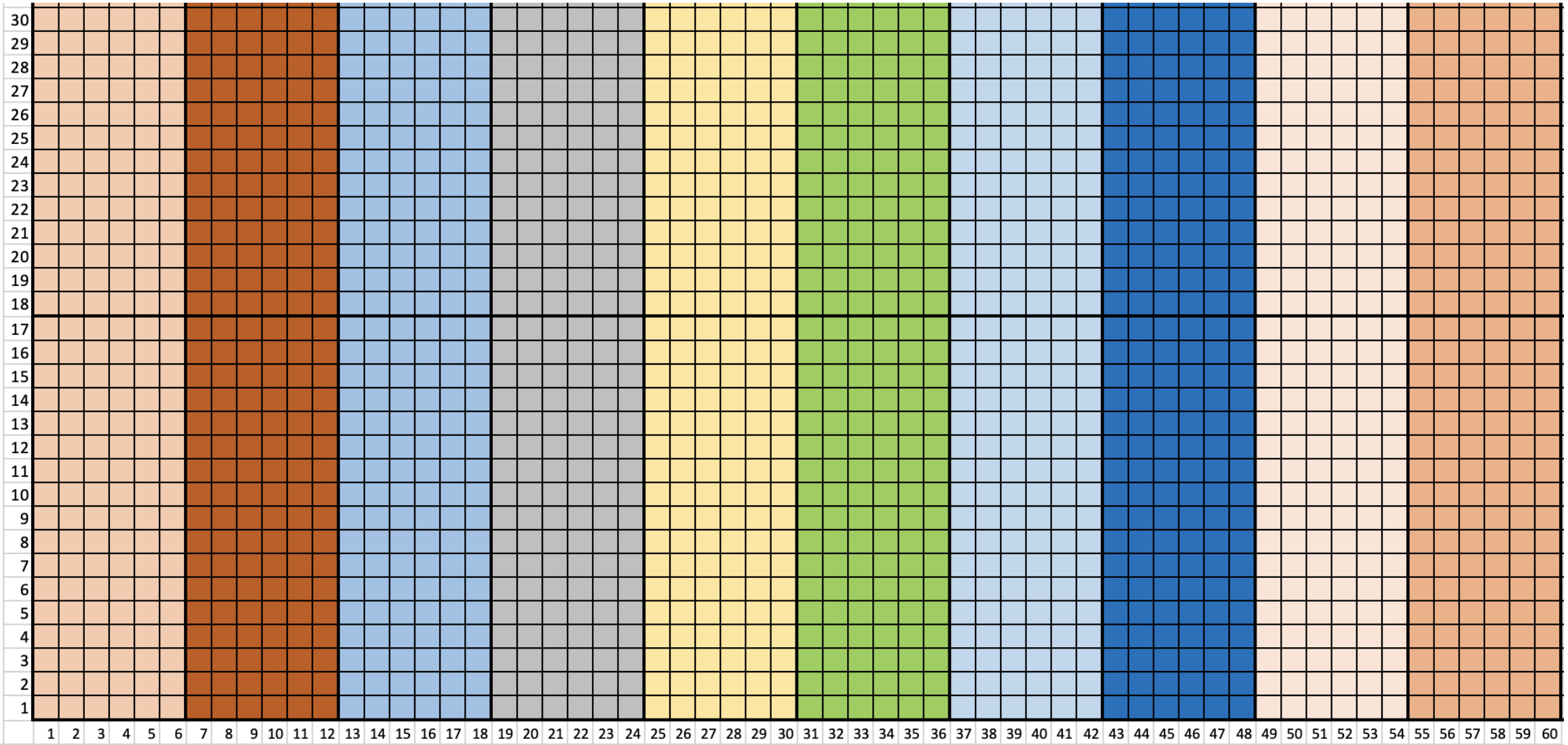
4. *Trigger Object Creation and send them out*

- Use cluster properties to generate **trigger candidates**:
  - Total transverse energy (HT)
  - Electron/Photon
  - Hadronic Jet

Task

- Partition the system into:
  - Input Buffering
  - Cluster Engine
  - Sorting Network
  - Global Trigger Builder
- Use the Pragma strategically
- Submit
  - Algorithm strategy & report
  - HLS Implementation
    - Which Pragma used and thoughts behind them
  - Performance report
  - How can you improve it further or add more functionality

Example: Barrel Calorimeter Segmentation

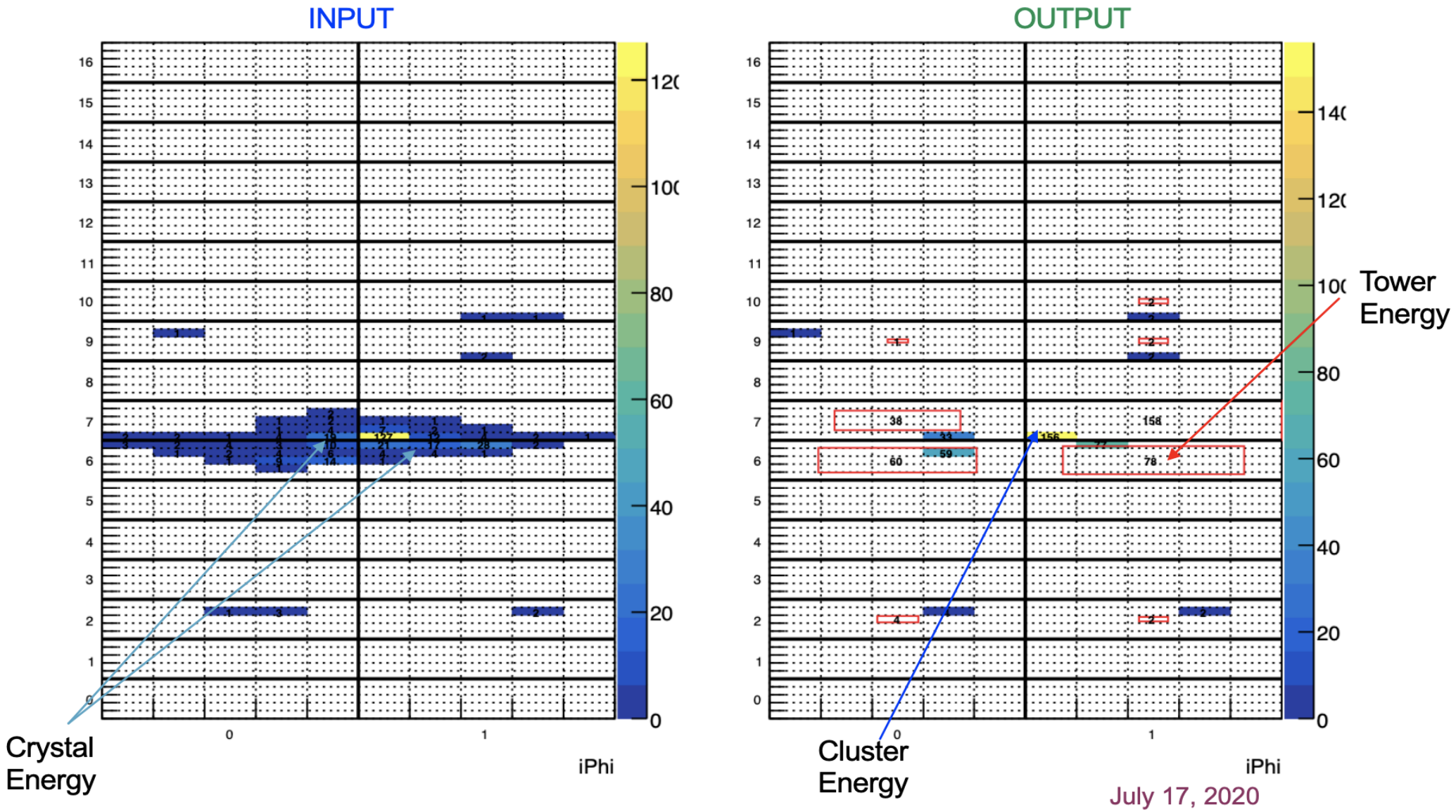


Isolation= 3x3/5x5 > 80%

5			
	13		
			9
2			

29

Example: Clustering



Project - 2

Real-time particle track reconstruction from simulated detector hits

- **Cluster** detector hits
- **Sort** them spatially or temporally
- Use a **Kalman filter** to estimate particle trajectories
- Output a list of candidate tracks in real-time

Ingredients

1. *Input Simulation*

- Take simulated tracker hits
  - (layer_id, x, y, z, time, charge) OR (layer_id, r, phi, z, time, charge)

2. *Clustering Algorithm*

- Group nearby hits in the same detector layer to form **cluster candidates** (e.g., based on distance or time)
- Clustering can use a simple approach or sliding window logic

3. *Sorting Hits*

- Sort hits by layer number, timestamp, or X/Y position to prepare for track fitting

4. *Track Fitting using Kalman Filter*

- Apply a Kalman filter to clustered and sorted hits to estimate track parameters (position, momentum, angle)

5. *Send tracks to output*

Project-3

Process data from a *Liquid Argon Time Projection Chamber (LArTPC)* used in neutrino experiments

- Identify and classify **neutrino interaction events** in real-time

Ingredients

1. ***Input Simulation***
   - Simulated ionization signals from neutrino interactions
2. ***Clustering***
   - Apply a **sliding window** or any other clustering to group adjacent hits (both spatial and temporal).
3. ***Sorting***
   - Sort hits or clusters by time (drift) or position to prepare for reconstruction
   - Sort and filter (maybe Kalman filter) clusters by time/space/energy.
4. ***Classification***
   - Simple rules: straight tracks = muons; diffuse = electrons; high energy = neutral pion decay?
   - Add a neural net to make classification
5. ***Send out sorted, classified events***

Project-4

Multi blob Video tagging with Cluster sorting

- Keep 40x40 frame
- From a video frame, detect different colored pixels (Red/Green/Blue)
- Group same color neighbouring red pixels into clusters
- Calculate Cluster properties
- Sort clusters by size
- Output tags like:
  - Cluster 0: color=red, size=12 -> Large_RED_Object
  - Cluster 1: color=Green, size=3 -> Small_Green_Object

Objective

- ***Implement algorithms***
- ***Optimize the latency and resource utilization using all possible pragma***
- ***Synthesize the code***
- ***Feel free to be more adventurous and add more features to algorithms***

What all pragma can we use?

```
#include <ap_int.h>
#include <hls_stream.h>

#include "example.h"

void example(din_t A[N][N], din_t B[N][N], din_t C[N][N]) {

 for (size_t i = 0; i < N; ++i) {
  for (size_t j = 0; j < N; ++j) {
   C[i][j]=0;
//#pragma HLS UNROLL factor = 4
   for (size_t k = 0; k < N; ++k) {

    C[i][j] += A[i][k] * B[k][j];
   }
  }
 }
}
```

HLS Pragmas:

- Interface
- Array Partition
- Array reshape
- Pipeline
- Dataflow
- Latency
- Allocation
- Stable
- Inline
- Unroll
- Aggregate
- Expression_balance
- Performance
- Protocol

TAC-HEP 2026

Questions?

TAC-HEP 2026

Extra Slides

TAC-HEP 2026

HLS Setup on cmstrigger02

https://github.com/varuns23/TAC-HEP-FPGA/blob/main/hls-setup/readme.md

Setup

Setup Port Forwarding for RDP

- From your local machine, run:
  - `ssh -L 3389:localhost:3389 -J <username>@login.hep.wisc.edu <username>@cmstrigger02.hep.wisc.edu`
- Keep this terminal open - it maintains the RDP tunnel

Connect Using Remote Desktop (RDP Client)

- Use Microsoft's RDP client (called `Windows App`) available for macOS and Windows.
- **Setup:** Download & install the `Windows App`, Open the app and click the **+** icon, then select `Add PC`
- **Configure:** Enter the IP address: `localhost:3389` or `127.0.0.1:3389`
- **Connect:** Double-click the PC icon, enter your UW computing and , and click **Connect**
- You should now see the remote desktop of `cmstrigger02`

https://github.com/varuns23/TAC-HEP-FPGA/blob/main/hls-setup/readme.md

Setup workign Directory

After logging in via RDP, open a terminal inside the remote desktop

- `mkdir -p /scratch/`whoami`` (if not there already)
- `cd /scratch/`whoami``
  - **For Vitis HLS:**
    - `Source /opt/Xilinx/Vitis/2020.1/settings64.sh`
    - `cd /scratch/`whoami``
    - `vitis_hls`
  - **For Vivado HLS:**
    - `Source /opt/Xilinx/Vivado/2020.1/settings64.sh`
    - `cd /scratch/`whoami``
    - `vivado_hls`

Jargons

- **ICs - Integrated chip:** assembly of hundreds of millions of transistors on a minor chip
- **PCB:** Printed Circuit Board
- **LUT - Look Up Table aka 'logic'** - generic functions on small bitwidth inputs. Combine many to build the algorithm
- **FF - Flip Flops** - control the flow of data with the clock pulse. Used to build the pipeline and achieve high throughput
- **DSP - Digital Signal Processor** - performs multiplication and other arithmetic in the FPGA
- **BRAM - Block RAM** - hardened RAM resource. More efficient memories than using LUTs for more than a few elements
- **PCIe or PCI-E - Peripheral Component Interconnect Express**: is a serial expansion bus standard for connecting a computer to one or more peripheral devices
- **InfiniBand** is a computer networking communications standard used in high-performance computing that features very high throughput and very low latency
- **HLS** - High Level Synthesis - compiler for C, C++, SystemC into FPGA IP cores
- **HDL** - Hardware Description Language - low level language for describing circuits
- **RTL** - Register Transfer Level - the very low level description of the function and connection of logic gates
- **FIFO** – First In First Out memory
- **Latency** - time between starting processing and receiving the result
  - Measured in clock cycles or seconds
- **II - Initiation Interval** - time from accepting first input to accepting next input

Terminology

- **HLS file:** C/C++ code that will be synthesised and run on FPGA
- **Test bench (TB) file:** C/C++ code that is run to test the HLS code. It calls the HLS functions and can run tests on their output, e.g. C asserts
- **Tcl scripts:** set of tcl instructions executed by the Vivado HLS shell

- **Synthesis:** C/C++ ➞ HDL lang (VHDL/Verilog)
- **Project:** Collection of HLS and test bench (TB) files
  - Has a top-level function name that is the starting point for synthesis
- **Solution:** specific implementation of project
  - Runs on a specific device at a specific clock frequency
- **C simulation:** HLS+TB files are compiled with gcc against HLS headers and lib and plainly run as any other executable
- **C/RTL cosimulation:** synthesized HLS code is run on simulator and results tested on the C/C++ test bench