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

GPU & FPGA module training: Part-2

Week-6: Project: Re-designing RCT

Lecture-12: April 26th 2023

Varun Sharma

University of Wisconsin – Madison, USA

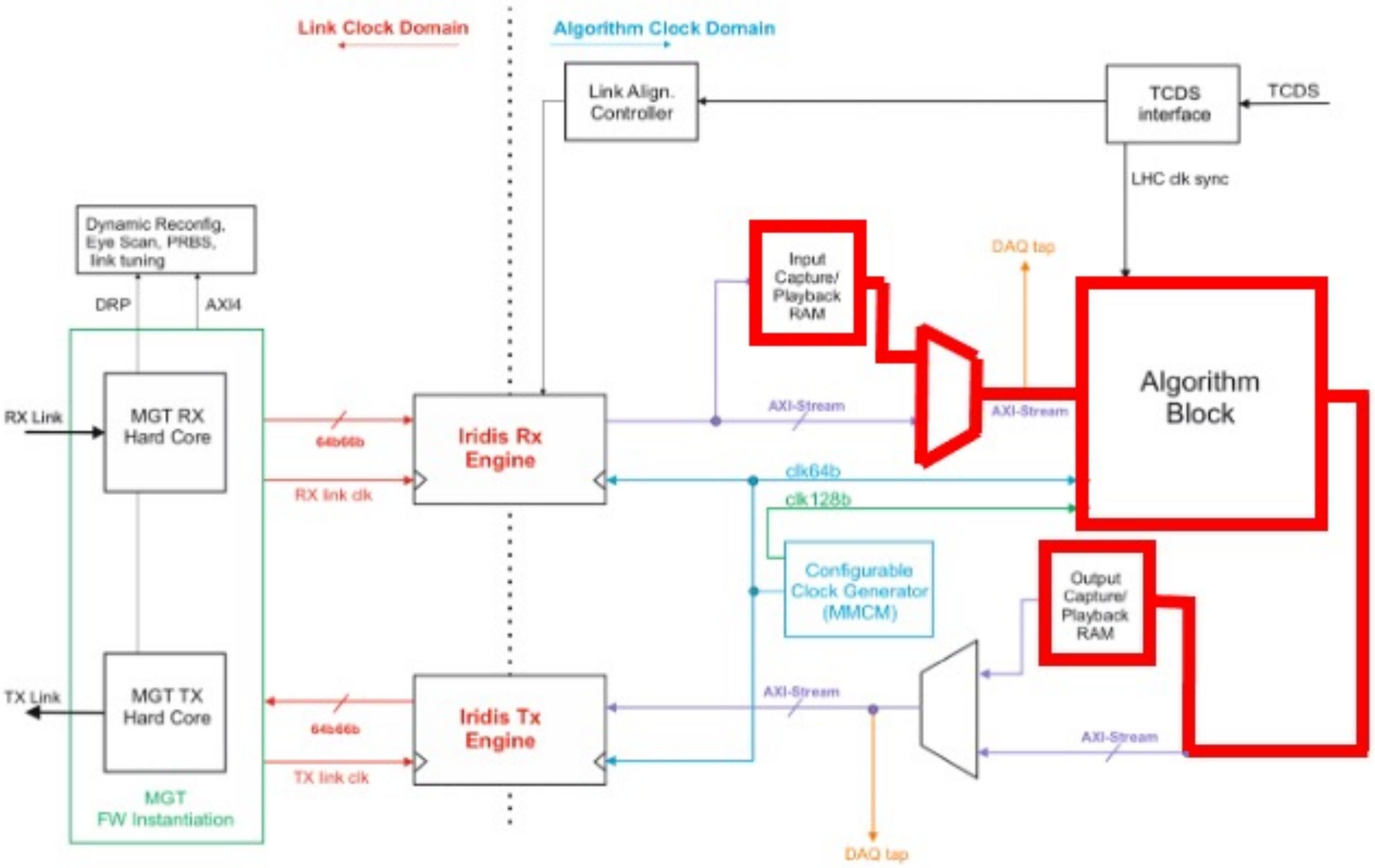
So Far…

- **FPGA and its architecture**
  - Registor/Flip-Flops, LUTs/Logic Cells, DSP, BRAMs
  - Clock Frequency, Latency
  - Extracting control logic & Implementing I/O ports
- **Parallelism in FPGA**
  - Scheduling, Pipelining, DataFlow
- **Vivado HLS**
  - Introduction, Setup, Hands-on for GUI/CLI, Introduction to Pragmas
  - Different Pragmas and their effects on performance
  - Practices to follow while writing HLS code – do's & don'ts
- **LHC and CMS Experiment: Level-1 Trigger System**

Today:
- More on Project:

APx Firmware Shell

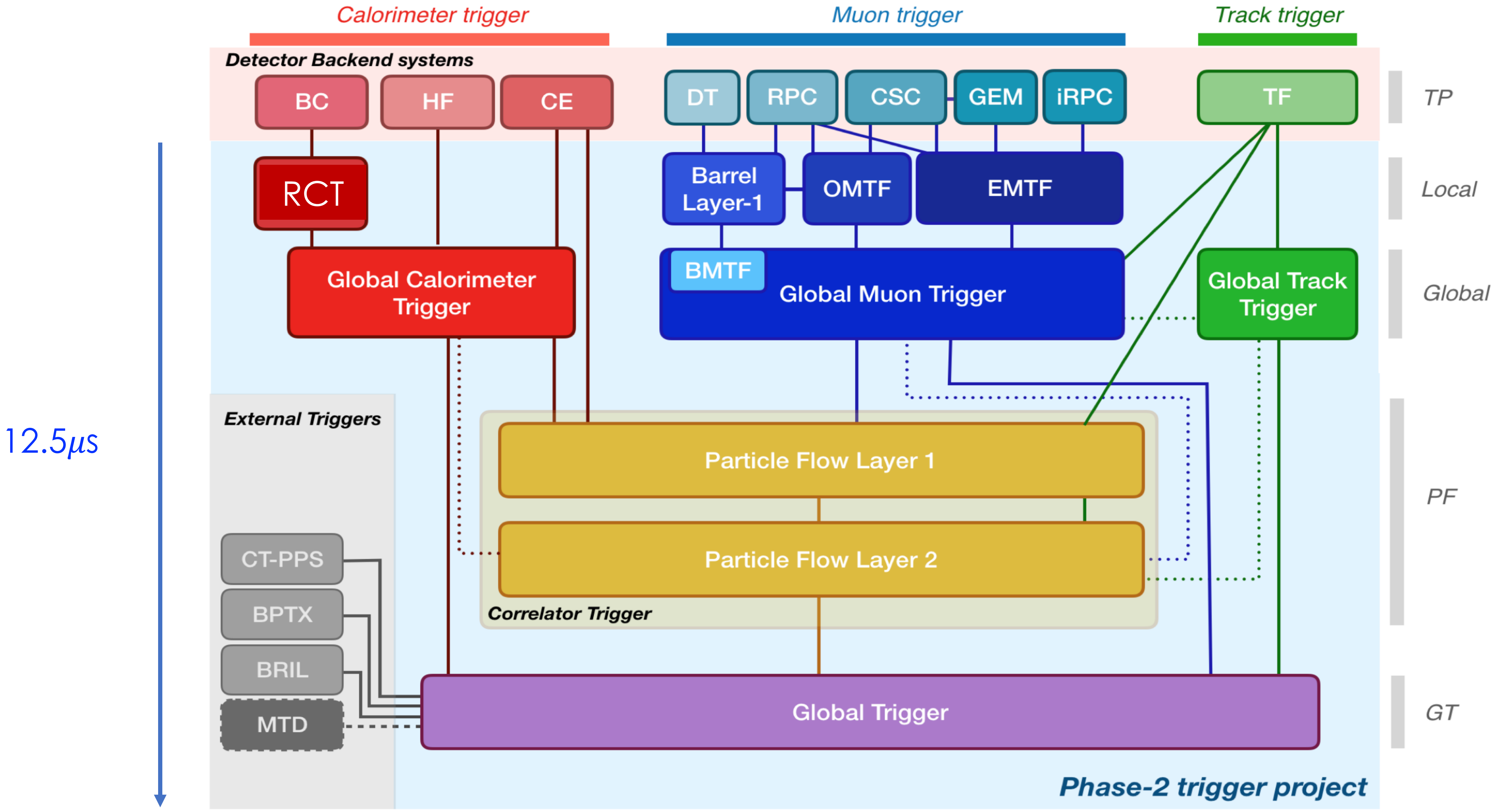
- Encoding/Decoding
- Current Protocol
  - 67b65b
  - 64b is data
  - 2 or 3b are control signals(Start, resets, debugging errors, etc ...)



Regional Calorimeter Trigger

TAC-HEP 2023

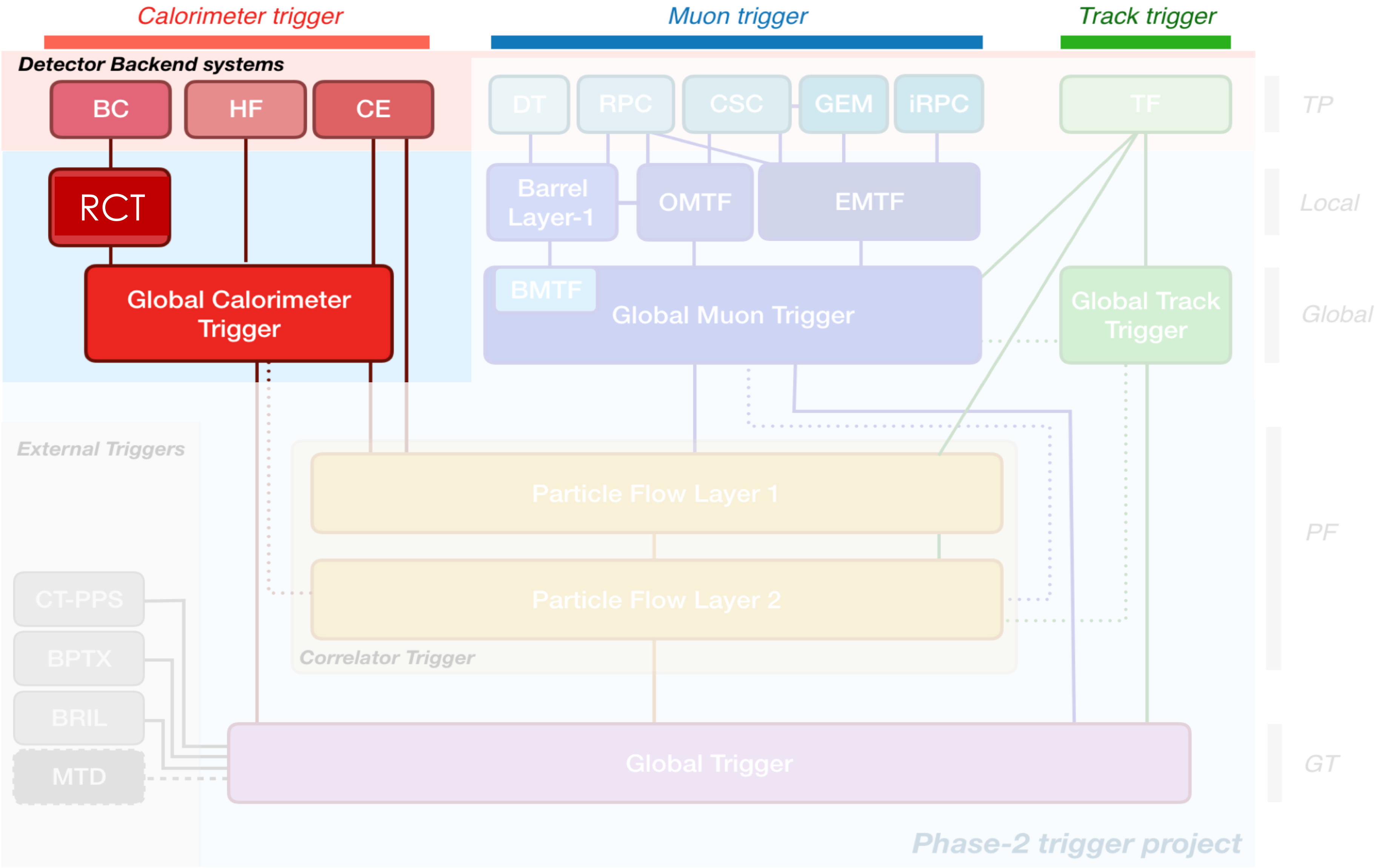
Phase-2 Level-1 Trigger



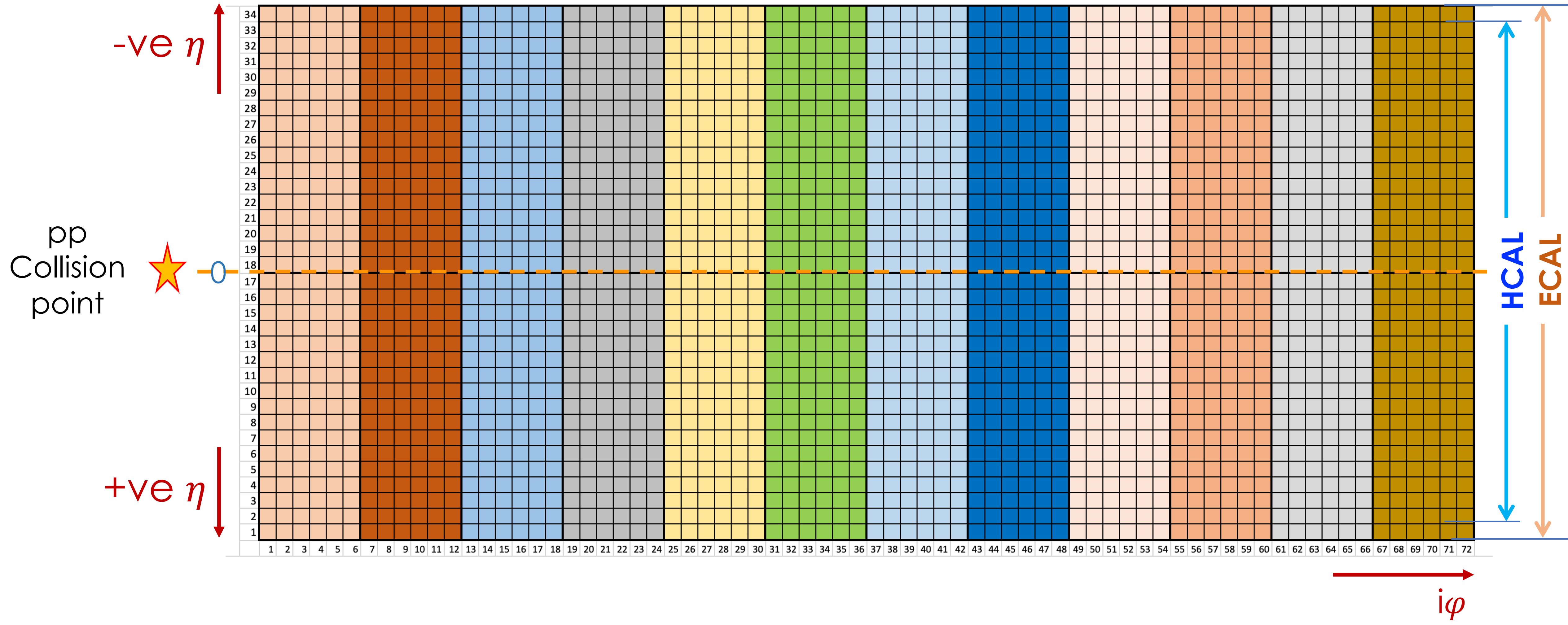
Phase-2 Calorimeter Level-1 Trigger

INPUTS:

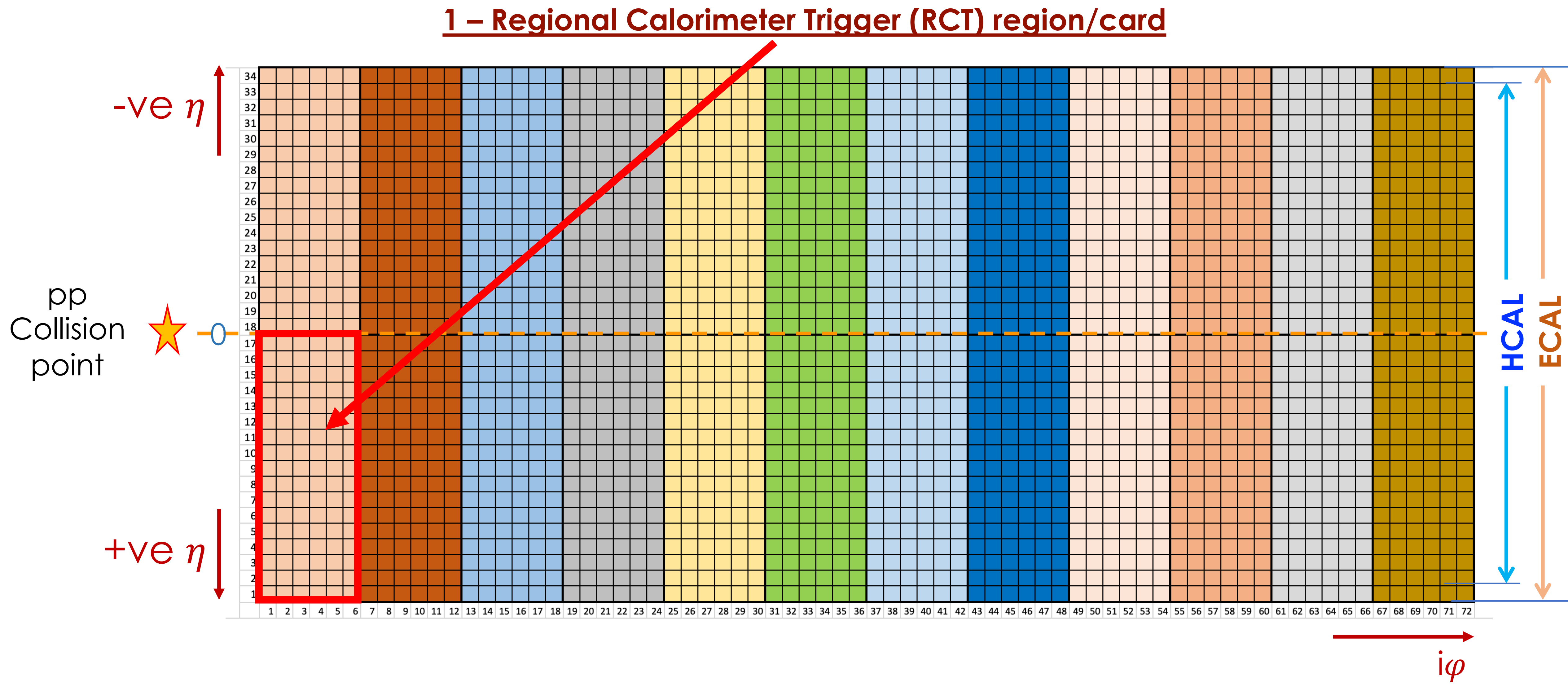
- ECAL crystals
- HCAL towers



Barrel Calorimeter Segmentation

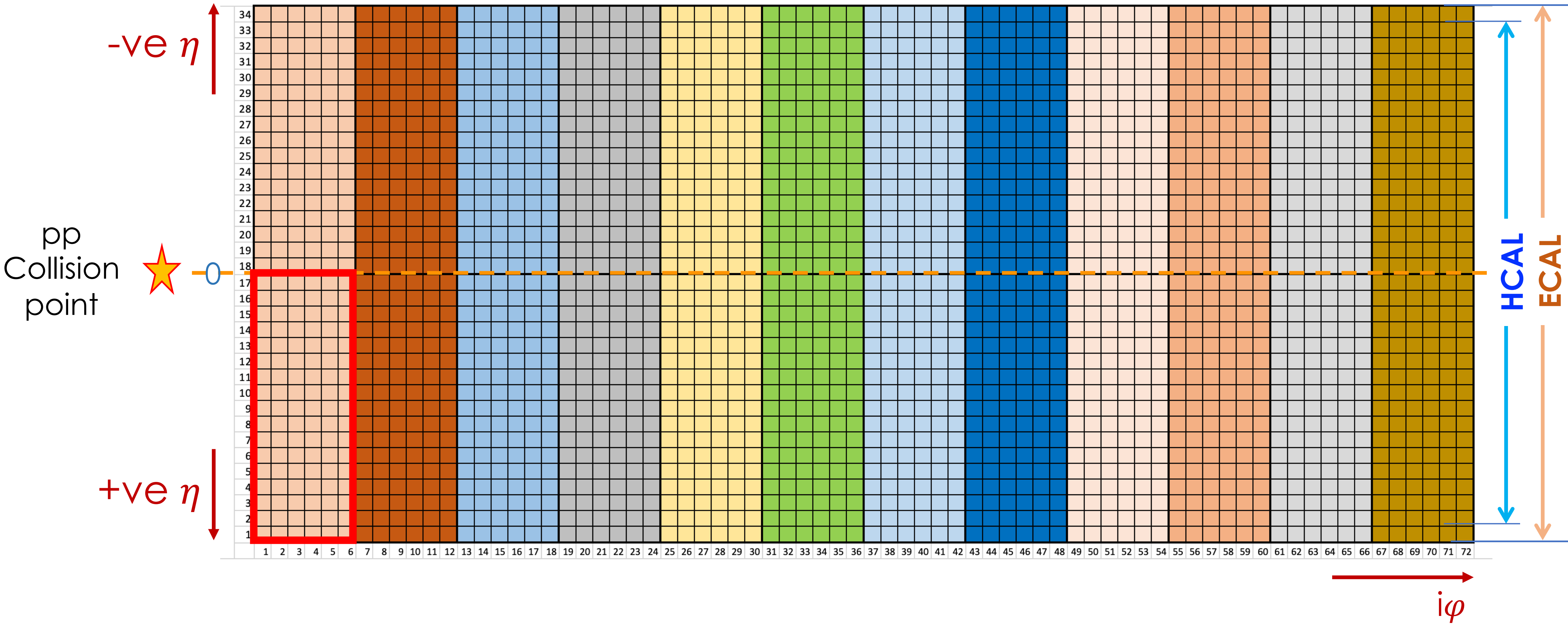


Barrel Calorimeter Segmentation



Barrel Calorimeter Segmentation

Full barrel Calorimeter: 24 RCT regions/cards



TAC-HEP 2023

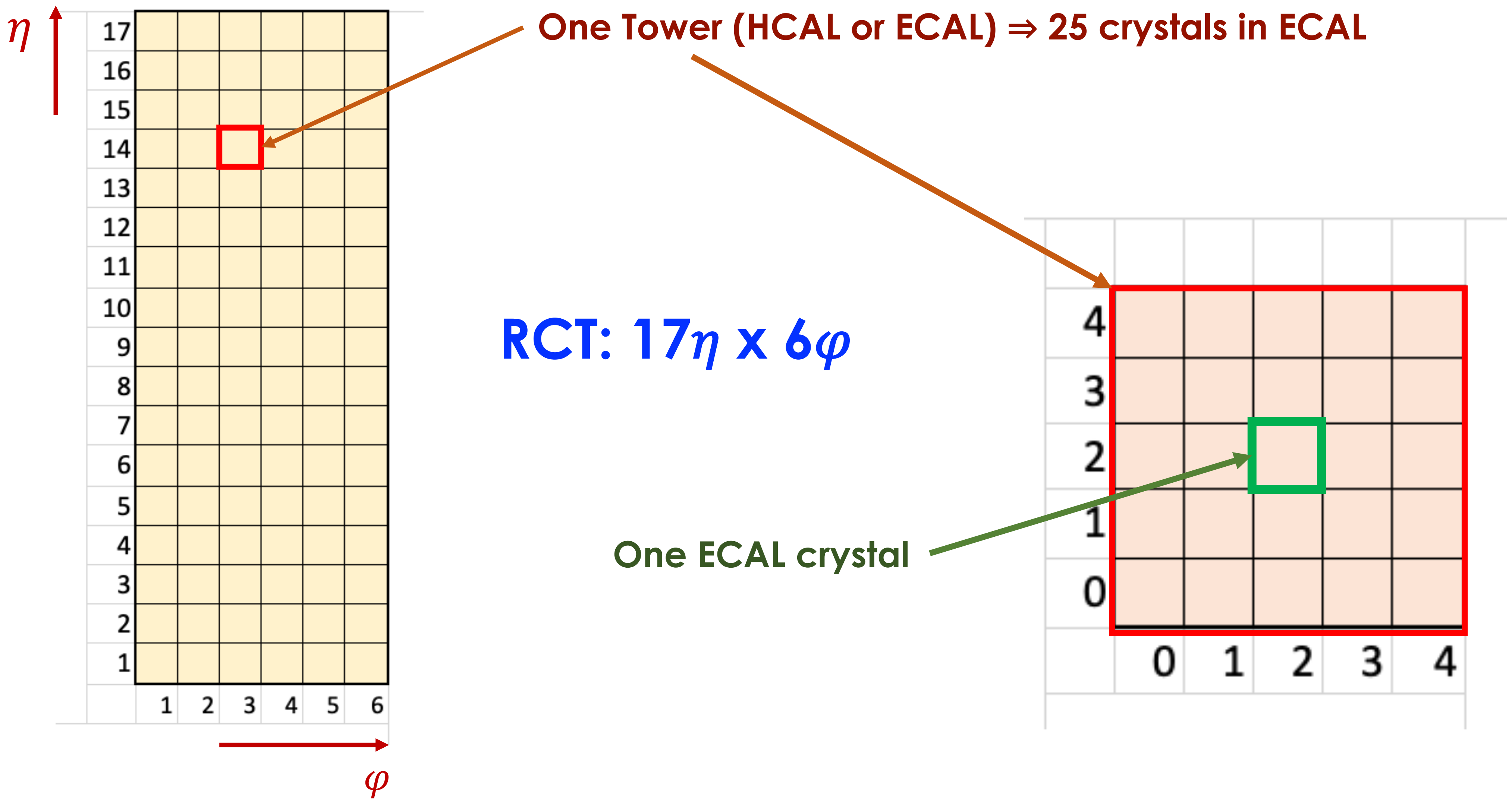
Project: Write & Synthesis an algorithm for single re-designed RCT

RCT Algorithm

- Take energy input from Calorimeter
- Cluster ECAL energy for each tower
  - Keep the position (eta, phi) of the seed (highest energetic) crystal
- Add HCAL tower energy to clustered tower

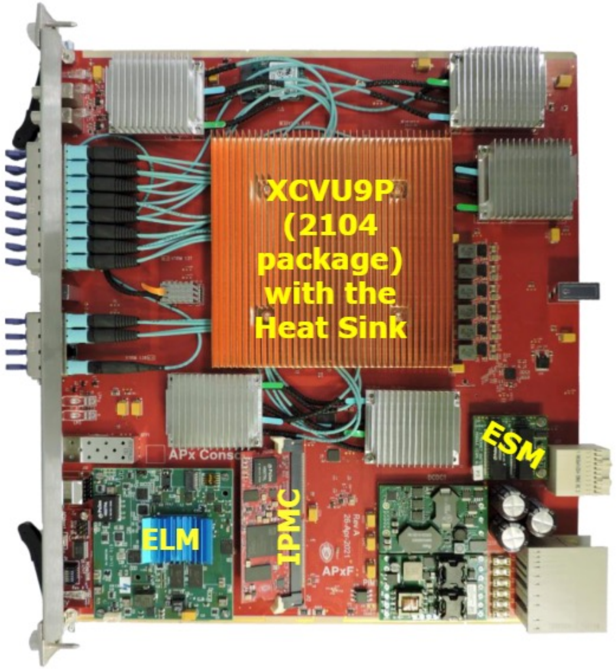
- Keep unclustered for respective tower
- Stitch clusters on boundary
- Sort highest 12 clusters and send as output
  - Cluster ET
  - Position of towers and corresponding seed crystal within tower
  - Unclustered Energy

Single RCT


η
17
16
15
14
13
12
11
10
9
8
7
6
5
4
3
2
1
1 2 3 4 5 6
φ
One Tower (HCAL or ECAL) ⇒ 25 crystals in ECAL
RCT: 17η x 6φ
One ECAL crystal
4
3
2
1
0
0 1 2 3 4

Physics bandwidth vs Algo Clock @ 25G

LHC BC Clock [MHz]	40.08
Word Bit Size	66
Line Rate [Gbps]	**25.78125**
Max Theoretical Words/Bx	9.74613



Bx Frame Length (TM interval)	**1**	**1**	**1**
Words/Frame	**7**	**8**	**9**
Equiv. Words/Bx	7.00	8.00	9.00
Equiv. Bits/Bx	448	512	576
Data Rate [Gbps]	18.52	21.16	23.81
Filler Rate [Gbps]	7.26	4.62	1.97
Average Filler Words/Bx	2.75	1.75	0.75
Average Filler Words/Orbit	9787.22	6223.22	2659.22
Average Filler Words/Frame	2.75	1.75	0.75
Payload Bits/Frame	**448**	**512**	**576**
Algo Clock @ 64b i/f [MHz]	**280.56**	**320.64**	**360.72**

Algorithm

1. Input per tower

- ECAL: 25 crystal energies
  - Each crystal 16b, total: 400b
- HCAL: tower energy
  - Each tower: 16b

1 - ECAL crystals = 16b	10 ET + 5 timing + 1 Spike
1 - HCAL towers = 16b	10 ET + 6 feature bits

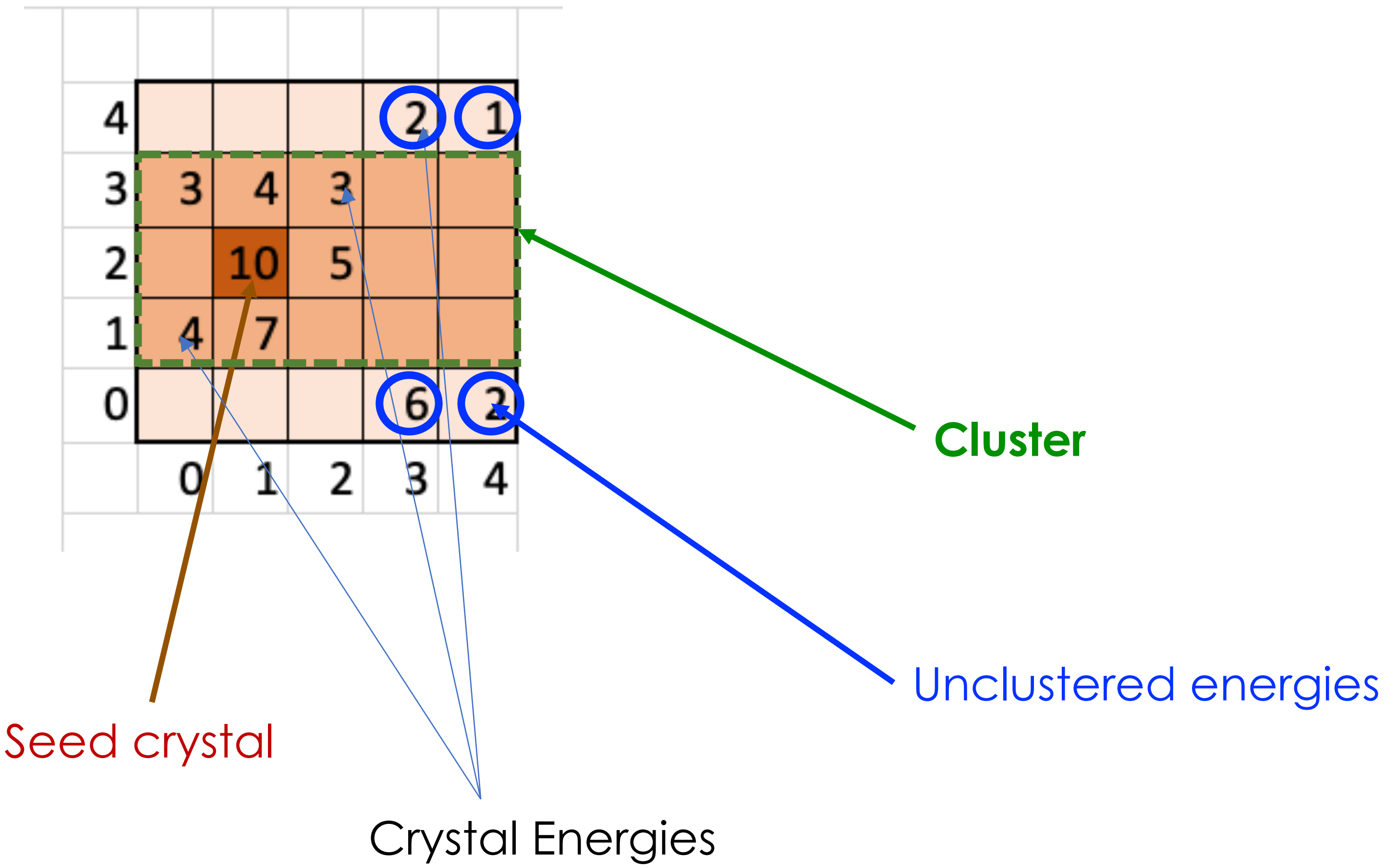
- Total input: 416b (ECAL + HCAL)
- 1 link can have 576 bits
- Lets assume 1 link carries 1 tower worth of information (416 b)
- We need total of 102 input links

2. Cluster these energies

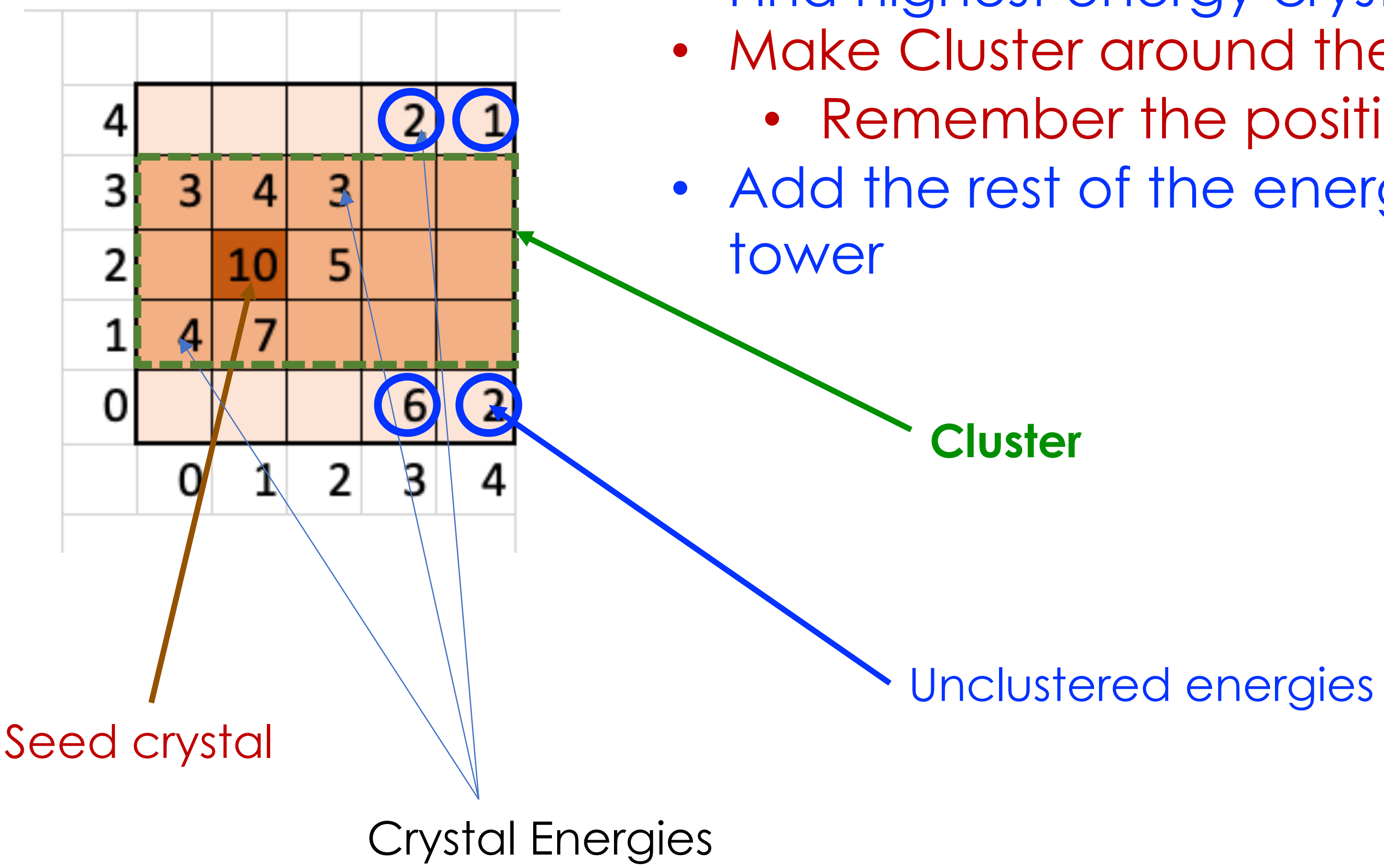
TAC-HEP 2023

Clustering energies in ECAL

Clustering

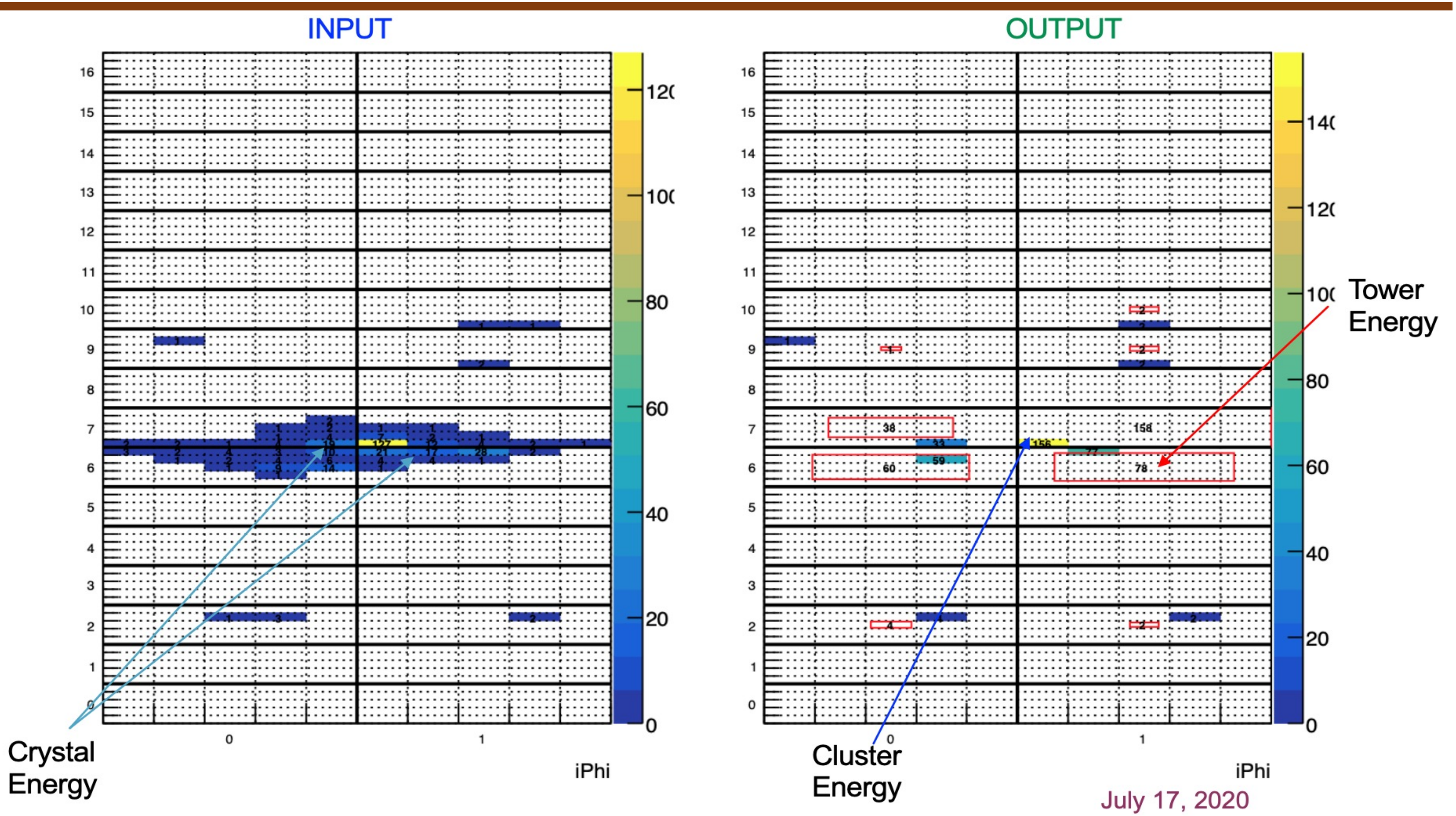

4
3
2
1
0
0
1
2
3
4
2
1
3
4
3
10
5
4
7
6
2
Cluster
Unclustered energies
Seed crystal
Crystal Energies

Clustering

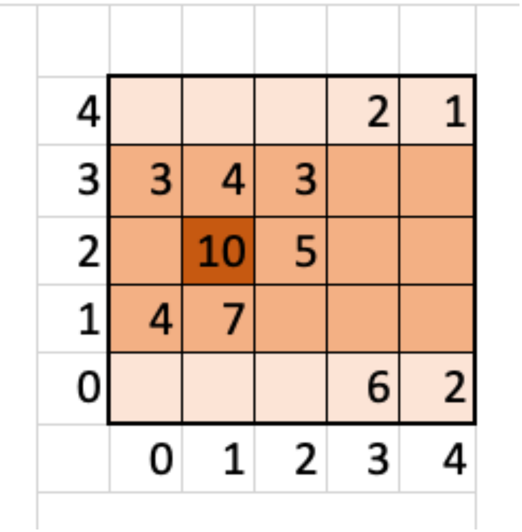


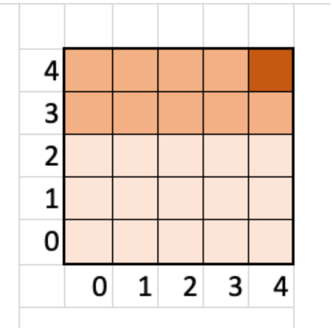
- Find highest energy crystal, often referred as seed crystal
- Make Cluster around the seed (3η x 5φ)
  - Remember the position of seed crystal
- Add the rest of the energy to unclustered energy for the tower

Clustering


INPUT
OUTPUT
iPhi
iPhi
Crystal Energy
Cluster Energy
Tower Energy
July 17, 2020

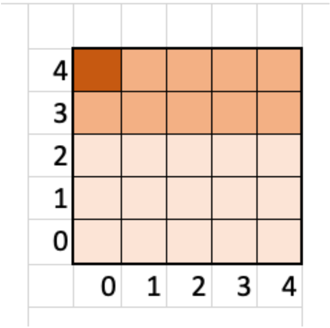
Clustering





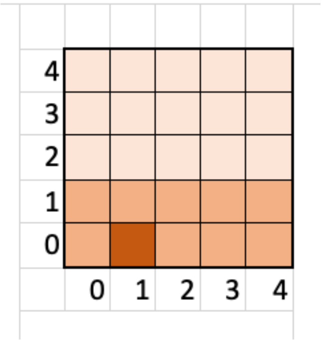
2η x 5φ

cEta:4, cPhi:4



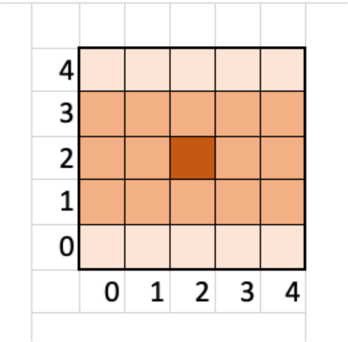
2η x 5φ

cEta:4, cPhi:0



2η x 5φ

cEta:0, cPhi:1



3η x 5φ

cEta:2, cPhi:2

Algorithm

1. Input per tower

- ECAL: 25 crystal energies
  - Each crystal 16b, total: 400b
- HCAL: tower energy
  - Each tower: 16b

1 - ECAL crystals = 16b	10 ET + 5 timing + 1 Spike
1 - HCAL towers = 16b	10 ET + 6 feature bits

- Total input: 416b (ECAL + HCAL)
- 1 link can have 576 bits
- Lets assume 1 link carries 1 tower worth of information (416 b)
- We need total of 102 input links

2. Cluster these energies

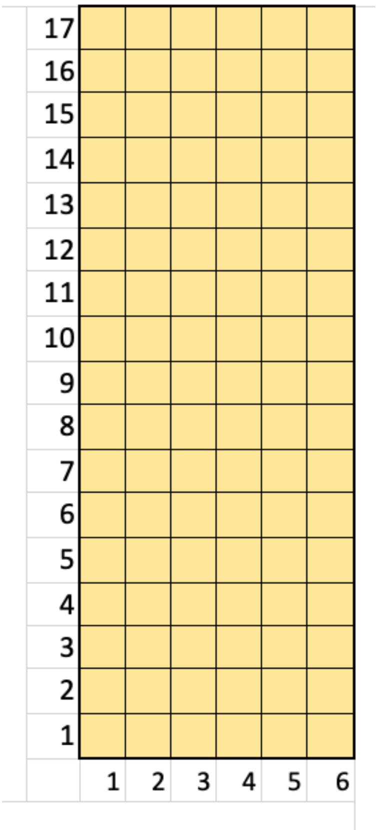
- Make clusters for each of these towers

Algorithm

1. **Input per tower**
2. **Cluster ECAL energies for each tower**
   - **Divide the RCT card further to make life simple**

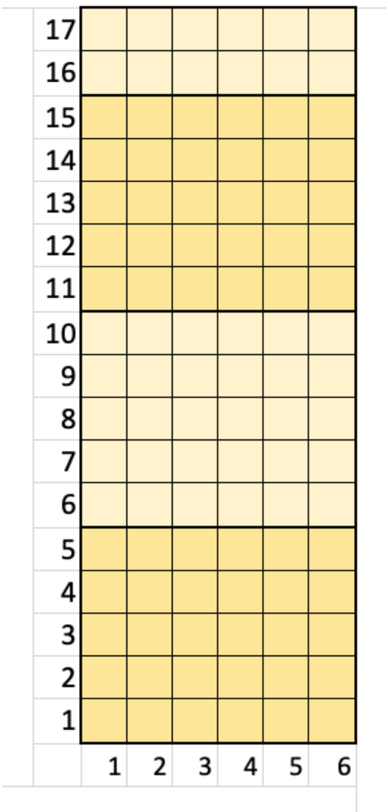
Divide & Rule

Case-1



1 RCT region: 17η x 6φ

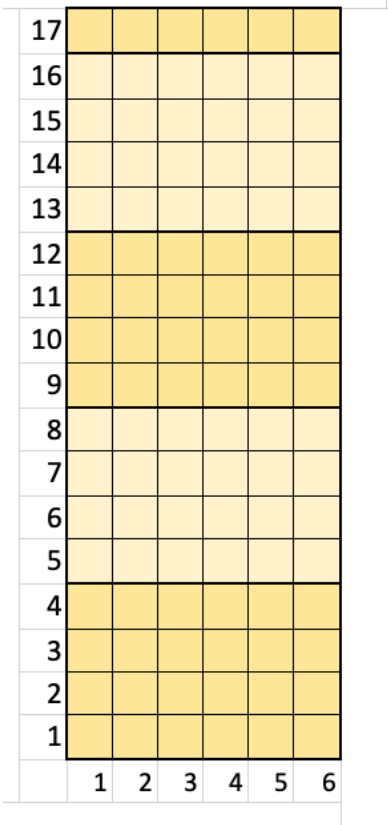
Case-2



1 RCT region: 17η x 6φ

- 4 sub-regions:
- 3: 5η x 6φ + 1: 2η x 6φ

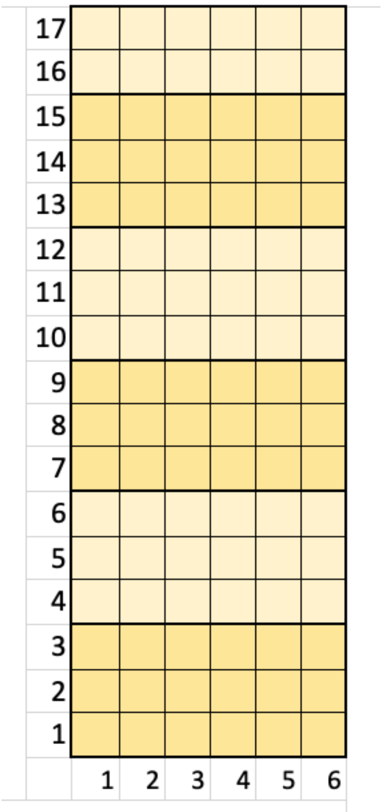
Case-3



1 RCT region: 17η x 6φ

- 5 sub-regions:
- 4: 4η x 6φ + 1: 1η x 6φ

Case-4



1 RCT region: 17η x 6φ

- 6 sub-regions:
- 5: 3η x 6φ + 1: 2η x 6φ

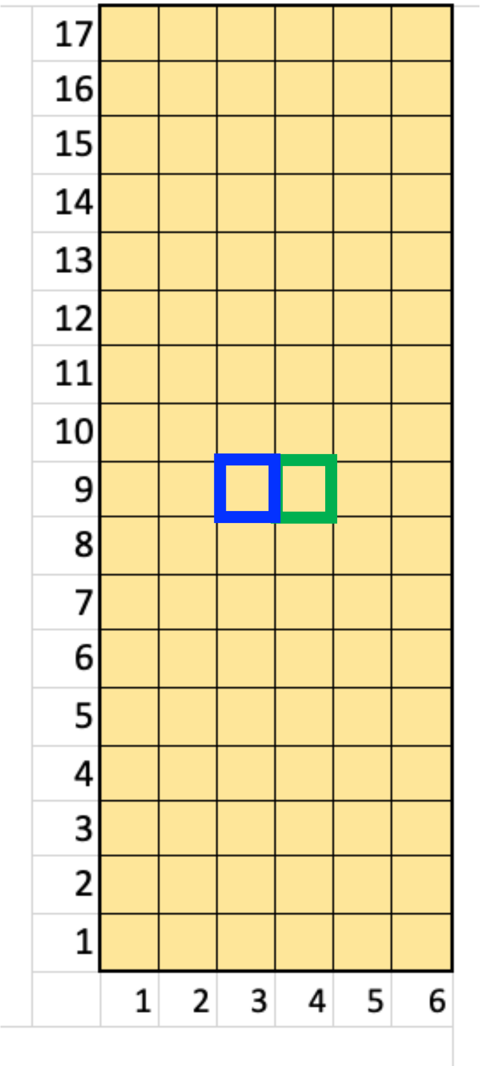
Choose wisely

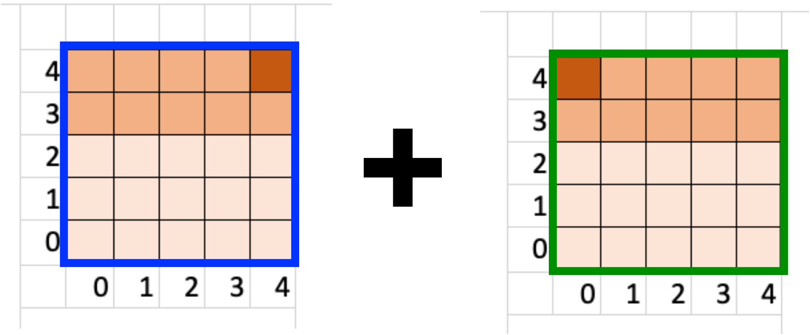
- Case-1
  - 102 cluster
- Case-2
  - 30 or 12 cluster per sub-region
- Case-3
  - 24 or 6 cluster per sub-region
- Case-4
  - 18 or 12 cluster per sub-region

Algorithm

1. **Input per tower**
2. **Cluster ECAL energies for each tower**
   - **Divide the RCT card further to make life simple**
3. **Stitch together the clusters for neighbouring towers and add HCAL energies for respective towers**

Stitching





Add energies if seed crystal are neighbor for neighboring towers

Choose wisely

- Case-1
  - 102 cluster
  - **Send 12 output tower**
- Case-2
  - 30 or 12 cluster per sub-region
  - **Send out 7 towers per sub-region (28 total towers)**
- Case-3
  - 24 or 6 cluster per sub-region
  - **Send out 6 towers per sub-region (30 total towers)**
- Case-4
  - 18 or 12 cluster per sub-region
  - **Send out 5 towers per sub-region (30 total towers)**

Algorithm

1. Input per tower
2. Cluster ECAL energies for each tower
   - Divide the RCT card further to make life simple
3. Stitch together the clusters for neighbouring towers
4. Sort the final list of towers

Sorting

Bitonic Sorter

- Comparison-based sorting algorithm that can run in parallel
- Converts random sequence of numbers into a bitonic sequence
  - One that monotonically increases, then decreases

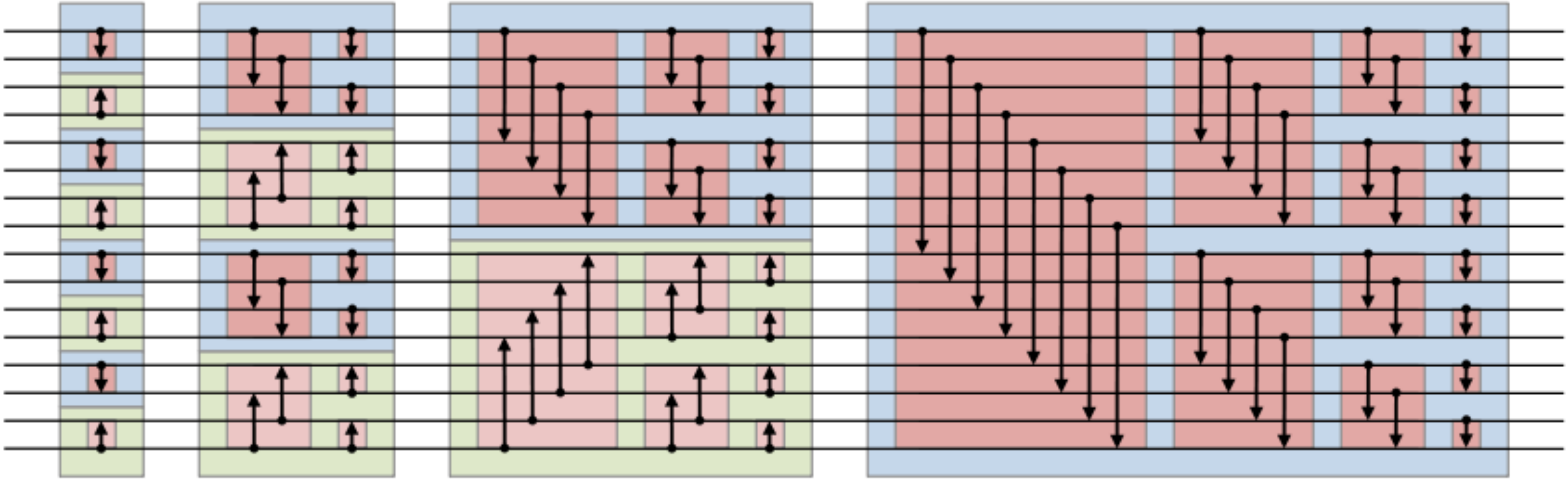


Fig: Bitonic sort with 16 elements

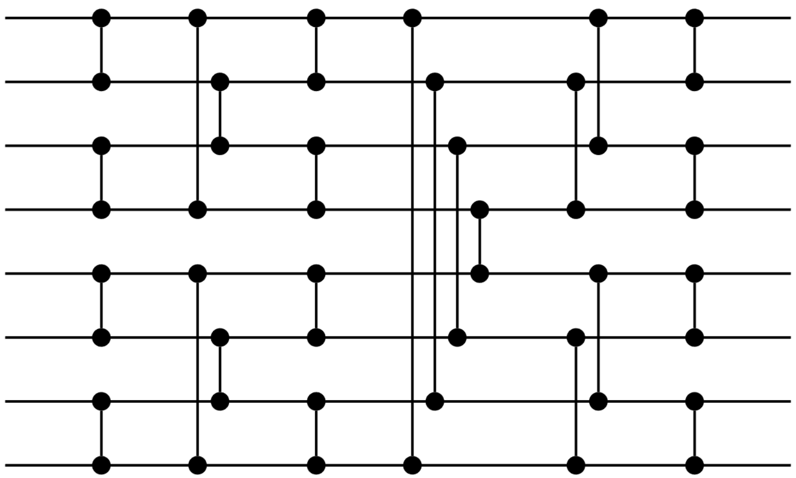
Algorithm

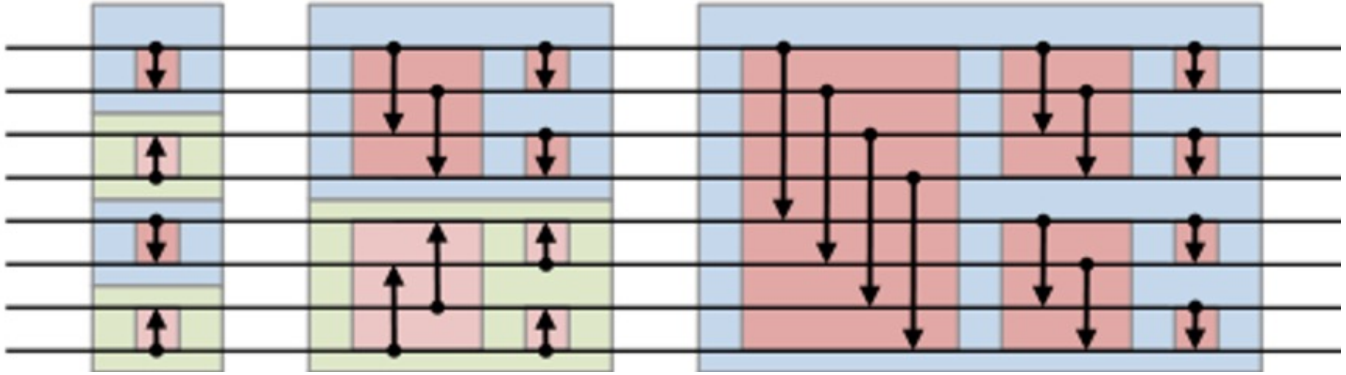
- Inputs: random set of $2n = 2^k$ (k is +ve integer) numbers
  - Every pair of elements is bitonic
- Bitonic sequences of size-2 are merged to create ordered lists of size 2
  - At the end of this stage of merging, we have N/4 bitonic sequence of size 4
- Bitonic sequence of size-4 are merged into sorted sequence of size 4, leading to N/8 bitonic sequence of size 8 and so on…
- Given an unordered sequence of $2n$, we have $log_2\ 2n$ stages

Bitonic Sorter

- Bitonic sort is a classic parallel algorithm for sorting & thus for FPGAs
- No. of comparisons: $O(n\log^2(n))$
- Suitable for hardware implementation:
  - Parallel implementation
  - Compare elements in a predefined sequence & doesn't depend on data
  - Only be done for elements: 2^n

An example



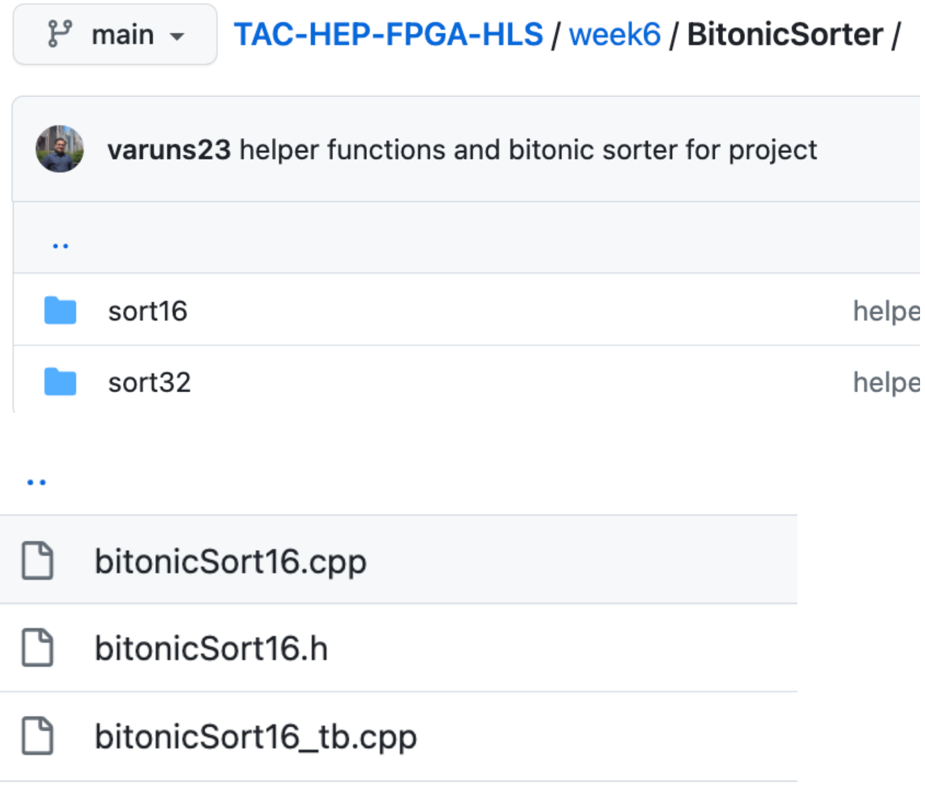


3	3		3	2		2	2	**2**
9	9		2	3		3	3	**3**
2	7		7	7		5	5	**4**
7	2		9	9		4	4	**5**
5	5		8	8		8	7	**6**
6	6		6	6		6	6	**7**
4	8		5	5		7	8	**8**
8	4		4	4		9	9	**9**

Ready to use code

https://github.com/varuns23/TAC-HEP-FPGA-HLS/tree/main/week6/BitonicSorter

```
#include "bitonicSort16.h"

//Main CAE block (compare and exchange)
GreaterSmaller AscendDescend(const din_t &x, const din_t &y){
#pragma HLS PIPELINE II=9
#pragma HLS INLINE
    GreaterSmaller s;

    s.greater = (x > y) ? x : y;
    s.smaller = (x > y) ? y : x;

    return s;
}

void FourinSmallFir(const din_t &x0, const din_t &x1, const din_t &x2, const din_t &x3,
                                        din_t &y0, din_t &y1, din_t &y2, din_t &y3){
   #pragma HLS PIPELINE II=9
   #pragma HLS INLINE
    GreaterSmaller res;
        res = AscendDescend(x0, x2);
        y0 = res.smaller; y2 = res.greater;

        res = AscendDescend(x1, x3);
        y1 = res.smaller; y3 = res.greater;
}

void FourinGreatFir(const din_t &x0, const din_t &x1, const din_t &x2, const din_t &x3,
                                        din_t &y0, din_t &y1, din_t &y2, din_t &y3){
   #pragma HLS PIPELINE II=9
   #pragma HLS INLINE
    GreaterSmaller res;
        res = AscendDescend(x0, x2);
        y0 = res.greater; y2 = res.smaller;

        res = AscendDescend(x1, x3);
        y1 = res.greater; y3 = res.smaller;
}

```



https://github.com/varuns23/TAC-HEP-FPGA-HLS/blob/main/week6/BitonicSorter/sort16/bitonicSort16.cpp

Algorithm

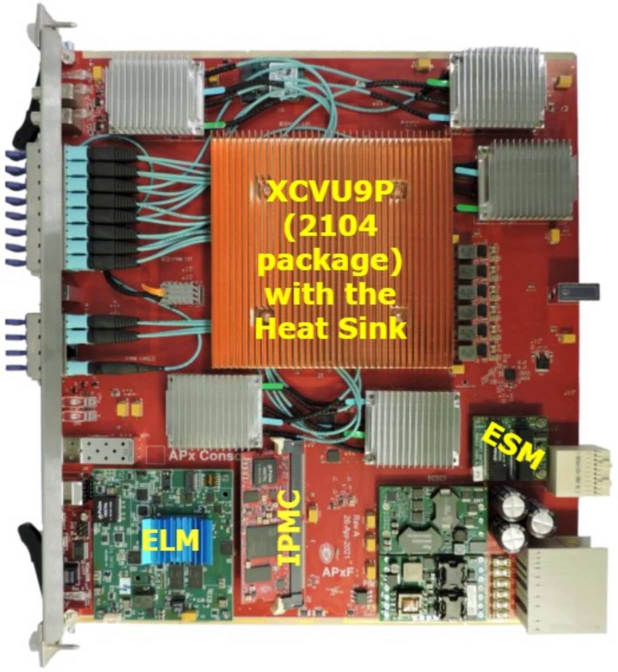
1. Input per tower
2. Cluster ECAL energies for each tower
   - Divide the RCT card further to make life simple
3. Stitch together the clusters for neighbouring towers
4. Sort the final list
5. Send just 12 towers per RCT region

Output

- 12 towers
- Information per tower
  - Clustered energy (32b)
  - Unclustered energy (32b)
  - Seed crystal position (cEta, cPhi)
  - Tower position (tEta, tPhi)
  - HoE (Ratio of HCAL/ECAL energies)

Contrainsts to use

- Target Clock Period
  - 2.7778ns
- Uncertainty
  - 30%
- Target Device
  - `xcvu9p-flgc2104-1-e`



Some functions to refer from

https://github.com/varuns23/TAC-HEP-FPGA-HLS/tree/main/week6

```
uint16_t getPeakBinOf5(uint16_t et[NCrystalsPerEtaPhi], uint16_t etSum);

bool getClustersInTower(uint16_t crystals[NCrystalsPerEtaPhi][NCrystalsPerEtaPhi],
    uint16_t *peakEta,
    uint16_t *peakPhi,
    uint16_t *largeClusterET,
    uint16_t *smallClusterET);

bool mergeClusters(uint16_t ieta1, uint16_t iphi1, uint16_t itet1, uint16_t icet1,
    uint16_t ieta2, uint16_t iphi2, uint16_t itet2, uint16_t icet2,
    uint16_t *eta1, uint16_t *phi1, uint16_t *tet1, uint16_t *cet1,
    uint16_t *eta2, uint16_t *phi2, uint16_t *tet2, uint16_t *cet2);

void stitchNeighbors(bool stitch, Tower Ai, Tower Bi, Tower &Ao, Tower &B);
```

- algo_top.cpp
- algo_top.h
- helperFunctions.cpp
- helperFunctions.h

Summary: Project

Write an algorithm to cluster ECAL and HCAL energies for Regional Calorimeter Trigger using HLS and synthesis the results

1. Input per tower (ECAL + HCAL)
2. Cluster ECAL energies for each tower
   - Divide the RCT card further to make life simple
3. Stitch together the clusters for neighbouring towers
4. Sort the final list
5. Send just 12 towers per RCT region

TAC-HEP 2023

Questions?

TAC-HEP 2023

Acknowledgement

Lectures are compiled using content from Xilinx's public pages/examples or different user guides

TAC-HEP 2023

Additional material

Assignment submission

- Where to submit:
  - https://pages.hep.wisc.edu/~varuns/assignments/TAC-HEP/
- Use your login machine credentials
- Submit one file per week
- Try to submit by following week's Tuesday

Correct Time

From 03.28.2023 onwards

- Tuesdays: 9:00-10:00 CT / 10:00-11:00 ET / 16:00-17:00 CET
- Wednesday: 11:00-12:00 CT / 12:00-13:00 ET / 18:00-19:00 CET

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
- **DRCs -** Design Rule Checks
- **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

Assignment Week-3

- Use target device: **xc7k160tfbg484-2**
- Clock period of 10ns

1. Execute the code (lec5Ex2.tcl) using CLI (slide-25) and compare the results with GUI results for C-Simulation, C-Synthesis

2. Vary following parameters for two cases: high and very high values and compare with 1 for both CLI and GUI
    - Variable: "samples"
    - Variable: "N"

3. Run example lec3Ex2a

Assignment Week-4

1. Do a matrix multiplication of two 1-dimensional arrays – A[N]*B[N], where N > 5
   a) Report synthesis results without any pragma directives
   b) Add as many pragma directives possible
      i. Report any conflicts (if reported in logs) between two pragmas

2. Compare the analysis perspective (Performance) for different case shared today
3. For Array_partitioning, instead of using complete, use block and cyclic with different factors

Assignment Week-5

1. Do exercise mention on slide-24
2. A matrix multiplication using two for loops and compare results for pragma loop_flatten & unroll
3. Write a simple program doing arithmetic operations(+, -, *, /, %) between two variable use of arbitrary precision to compare results between stand c/c++ data types and using ap_(u)int<N>
4. Write a program using an array with N(=10/15/20) elements and then restructure the code with a struct having N-data member. Compare the results of two programs