Firmware updates – May 2015 – v0.9

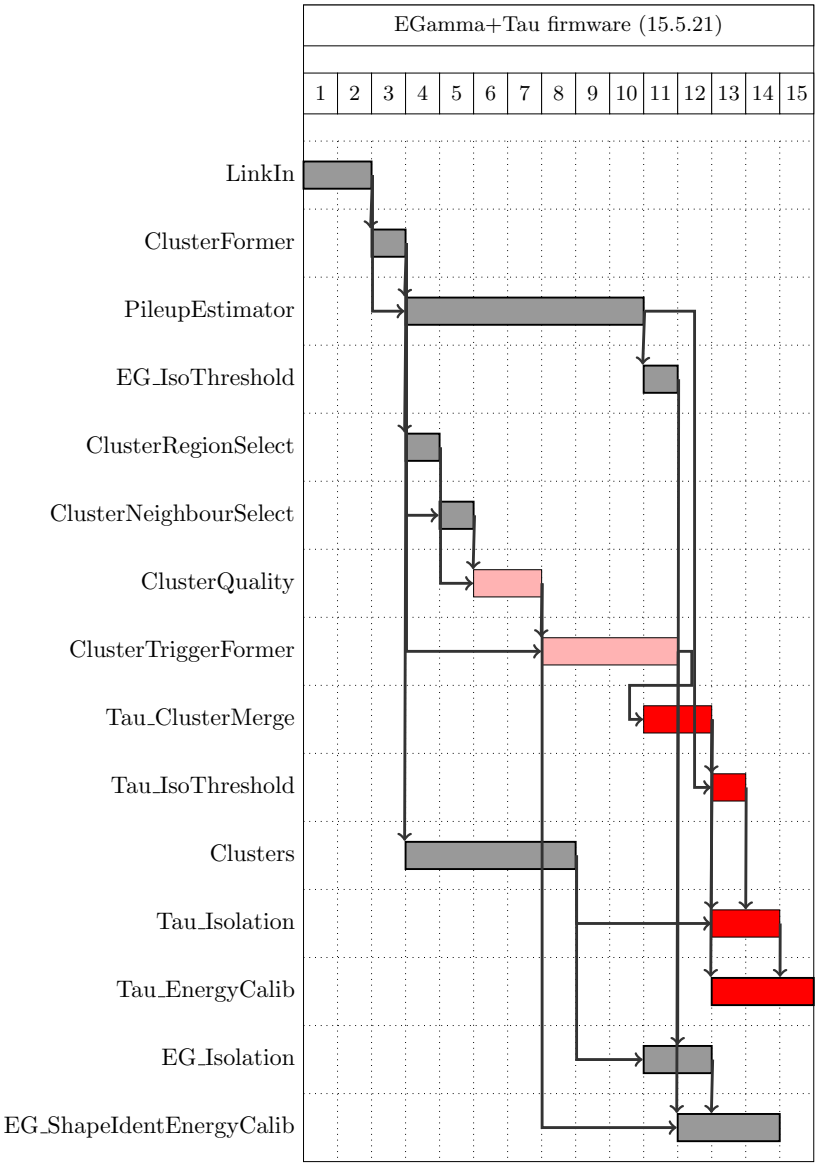

Figure 1: Timing and relation graph of the EGamma and Tau firmware. Each row corresponds to a given module of the algorithm, while the columns indicate the clock number. Gray modules are modules that don't require updates with respect to the current firmware implementation. Pink modules require updates or fixes. Red modules are currently not existing.

In the following, the updates to the current EGamma firmware are listed. Modifications of existing code are detailed in a pseudo-code form. Code to be removed is indicated in red, while code to be added is indicated in blue; black code doesn't need changes.

1 Reduction of the number of instances

The modifications related to the reduction of the number of potential candidates from 72 to 36 in each slice are not detailed here. It is just mentioned that this reduction can be done right after the seed filtering (section 2.2 below) and before the sharing (section 2.3).

2 ClusterQuality

2.1 ClusterThresholdQuality

The only update here is the corner trimming that should be removed.

Trimming updates

```
if(R1N<clusterThreshold and R1E<clusterThreshold) keepTower_R1NE=false
if(R1S<clusterThreshold and R1E<clusterThreshold) keepTower_R1SE=false
if(R1N<clusterThreshold and R1W<clusterThreshold) keepTower_R1NW=false
if(R1S<clusterThreshold and R1W<clusterThreshold) keepTower_R1SW=false
```

2.2 EGammaFilteringQuality

The updates in the filtering correspond to fixes that have been made in the emulator since the egamma algorithm specifications have been written. There are two main fixes:

- The conditions for the negative boundary should be the same as for the positive boundary.
- When looking at towers in the second ring (R2), clusters should be invalidated only if the tower in-between pass the clustering threshold.

There are in addition a few inconsistencies in the current VHDL code.
Fixes are listed below.

Filtering updates

```
//*****************************************************************//
// Positive eta
if(iEta>1)
  if(R2N  >  C and R1N >= clusterThreshold) invalidateCluster
  if(R1N  >  C) invalidateCluster
  if(R1NE >  C) invalidateCluster
  if(R1E  >  C) invalidateCluster
  if(R1SE >  C) invalidateCluster
  if(R1S  >= C) invalidateCluster
  if(R1SW >= C) invalidateCluster
  if(R1W  >= C) invalidateCluster
  if(R1NW >= C) invalidateCluster
  if(R2S  >= C) invalidateCluster
  if(R2S  >= C and R1S >= clusterThreshold) invalidateCluster
//*****************************************************************//
// Boundary positive eta
else if(iEta=1)
  if(R2N  >  C and R1N >= clusterThreshold) invalidateCluster
  if(R1N  >  C) invalidateCluster
  if(R1NE >  C) invalidateCluster
  if(R1E  >  C) invalidateCluster
  if(R1SE >  C) invalidateCluster
  if(R1S  >= C) invalidateCluster
```

```
  if(R1SW >  C) invalidateCluster
  if(R1W  >  C) invalidateCluster
  if(R1NW >  C) invalidateCluster
  if(R2S  >= C) invalidateCluster
  if(R2S  >= C and R1S >= clusterThreshold) invalidateCluster
//*****************************************************************//
// Negative eta (including boundary)
else if(iEta<0)
  if(R2N  >= C) invalidateCluster
  if(R2N  >= C and R1N >= clusterThreshold) invalidateCluster
  if(R1N  >= C) invalidateCluster
  if(R1NE >= C) invalidateCluster
  if(R1E  >= C) invalidateCluster
  if(R1SE >= C) invalidateCluster
  if(R1S  >  C) invalidateCluster
  if(R1SW >  C) invalidateCluster
  if(R1W  >  C) invalidateCluster
  if(R1NW >  C) invalidateCluster
  if(R2S  >  C and R1S >= clusterThreshold) invalidateCluster

//*****************************************************************//
```

2.3 EGammaSharingQuality

The updates in the sharing correspond also to fixes that have been made in the emulator since the egamma algorithm specifications have been written. They are mainly related to the way towers at $i\phi = \pm 2$ are shared between two seeds.

Sharing updates

```
//*****************************************************************//
// Positive eta
if(iEta>1)
  if R2N > C
    keepTower_R1NW = false
    keepTower_R1N  = false
    keepTower_R1NE = false
  if R2NNE > C
    keepTower_R1N  = false
    keepTower_R1NE = false
    keepTower_R2N  = false
    if(R1NE >= clusterThreshold) keepTower_R1E = false
  if R2SSE > C
    keepTower_R1S  = false
    keepTower_R1SE = false
    keepTower_R2S  = false
    if(R1SE >= clusterThreshold) keepTower_R1E = false
  if R2S >= C
    keepTower_R1SW = false
    keepTower_R1S  = false
    keepTower_R1SE = false
  if R2SSW >= C
    keepTower_R1SW = false
    keepTower_R1S  = false
    keepTower_R2S  = false
    if(R1SW >= clusterThreshold) keepTower_R1W = false
  if R2NNW >= C
    keepTower_R1NW = false
    keepTower_R1N  = false
    keepTower_R2N  = false
    if(R1NW >= clusterThreshold) keepTower_R1W = false
  if R3N > C
    keepTower_R1N = false
    if(R2N >= clusterThreshold) keepTower_R1N = false
    keepTower_R2N = false
  if R3NE > C
    keepTower_R1NE = false
    if(R2NNE >= clusterThreshold) keepTower_R1NE = false
    keepTower_R2N  = false
  if R3NW >= C
```

```
    keepTower_R1NW = false
    if(R2NNW >= clusterThreshold) keepTower_R1NW = false
    keepTower_R2N  = false
  if R3S >= C
    keepTower_R1S = false
    if(R2S >= clusterThreshold) keepTower_R1S = false
    keepTower_R2S = false
  if R3SE > C
    keepTower_R1SE = false
    if(R2SSE >= clusterThreshold) keepTower_R1SE = false
    keepTower_R2S  = false
  if R3SW >= C
    keepTower_R1SW = false
    if(R2SSW >= clusterThreshold) keepTower_R1SW = false
    keepTower_R2S  = false
  if R4N > C
    keepTower_R1N = false
    keepTower_R2N = false
    if(R3N >= clusterThreshold) keepTower_R2N = false
  if R4S >= C
    keepTower_R1S = false
    keepTower_R2S = false
    if(R3S >= clusterThreshold) keepTower_R2S = false
//*****************************************************************//
// Boundary Positive eta
if(iEta=1)
  if R2N > C
    keepTower_R1NW = false
    keepTower_R1N  = false
    keepTower_R1NE = false
  if R2NNE > C
    keepTower_R1N  = false
    keepTower_R1NE = false
    keepTower_R2N  = false
    if(R1NE >= clusterThreshold) keepTower_R1E = false
  if R2SSE > C
    keepTower_R1S  = false
    keepTower_R1SE = false
    keepTower_R2S  = false
    if(R1SE >= clusterThreshold) keepTower_R1E = false
  if R2S >= C
    keepTower_R1SW = false
    keepTower_R1S  = false
    keepTower_R1SE = false
  if R2SSW >= C
  if R2SSW > C
    keepTower_R1SW = false
    keepTower_R1S  = false
    keepTower_R2S  = false
    if(R1SW >= clusterThreshold) keepTower_R1W = false
  if R2NNW > C
    keepTower_R1NW = false
    keepTower_R1N  = false
    keepTower_R2N  = false
    if(R1NW >= clusterThreshold) keepTower_R1W = false
  if R3N > C
    keepTower_R1N = false
    if(R2N >= clusterThreshold) keepTower_R1N = false
    keepTower_R2N = false
  if R3NE > C
    keepTower_R1NE = false
    if(R2NNE >= clusterThreshold) keepTower_R1NE = false
    keepTower_R2N  = false
  if R3NW > C
    keepTower_R1NW = false
    if(R2NNW >= clusterThreshold) keepTower_R1NW = false
    keepTower_R2N  = false
  if R3S >= C
    keepTower_R1S = false
    if(R2S >= clusterThreshold) keepTower_R1S = false
    keepTower_R2S = false
  if R3SE > C
```

```
    keepTower_R1SE = false
    if(R2SSE >= clusterThreshold) keepTower_R1SE = false
    keepTower_R2S  = false
  if R3SW >= C
  if R3SW > C
    keepTower_R1SW = false
    if(R2SSW >= clusterThreshold) keepTower_R1SW = false
    keepTower_R2S  = false
  if R4N > C
    keepTower_R1N = false
    keepTower_R2N = false
    if(R3N >= clusterThreshold) keepTower_R2N = false
  if R4S >= C
    keepTower_R1S = false
    keepTower_R2S = false
    if(R3S >= clusterThreshold) keepTower_R2S = false
//*****************************************************************//
// Negative eta
if(iEta<-1)
  if R2N >= C
    keepTower_R1NW = false
    keepTower_R1N  = false
    keepTower_R1NE = false
  if R2NNE >= C
    keepTower_R1N  = false
    keepTower_R1NE = false
    keepTower_R2N  = false
    if(R1NE >= clusterThreshold) keepTower_R1E = false
  if R2SSE >= C
    keepTower_R1S  = false
    keepTower_R1SE = false
    keepTower_R2S  = false
    if(R1SE >= clusterThreshold) keepTower_R1E = false
  if R2S > C
    keepTower_R1SW = false
    keepTower_R1S  = false
    keepTower_R1SE = false
  if R2SSW > C
    keepTower_R1SW = false
    keepTower_R1S  = false
    keepTower_R2S  = false
    if(R1SW >= clusterThreshold) keepTower_R1W = false
  if R2NNW > C
    keepTower_R1NW = false
    keepTower_R1N  = false
    keepTower_R2N  = false
    if(R1NW >= clusterThreshold) keepTower_R1W = false
  if R3N >= C
    keepTower_R1N = false
    if(R2N >= clusterThreshold) keepTower_R1N = false
    keepTower_R2N = false
  if R3NE >= C
    keepTower_R1NE = false
    if(R2NNE >= clusterThreshold) keepTower_R1NE = false
    keepTower_R2N  = false
  if R3NW > C
    keepTower_R1NW = false
    if(R2NNW >= clusterThreshold) keepTower_R1NW = false
    keepTower_R2N  = false
  if R3S > C
    keepTower_R1S = false
    if(R2S >= clusterThreshold) keepTower_R1S = false
    keepTower_R2S = false
  if R3SE >= C
    keepTower_R1SE = false
    if(R2SSE >= clusterThreshold) keepTower_R1SE = false
    keepTower_R2S  = false
  if R3SW > C
    keepTower_R1SW = false
    if(R2SSW >= clusterThreshold) keepTower_R1SW = false
    keepTower_R2S  = false
  if R4N >= C
```

```
    keepTower_R1N = false
    keepTower_R2N = false
    if(R3N >= clusterThreshold) keepTower_R2N = false
  if R4S > C
    keepTower_R1S = false
    keepTower_R2S = false
    if(R3S >= clusterThreshold) keepTower_R2S = false
//*****************************************************************//
// Boundary Negative eta
if(iEta=-1)
  if R2N > C
  if R2N >= C
    keepTower_R1NW = false
    keepTower_R1N  = false
    keepTower_R1NE = false
  if R2NNE > C
  if R2NNE >= C
    keepTower_R1N  = false
    keepTower_R1NE = false
    keepTower_R2N  = false
    if(R1NE >= clusterThreshold) keepTower_R1E = false
  if R2SSE >= C
    keepTower_R1S  = false
    keepTower_R1SE = false
    keepTower_R2S  = false
    if(R1SE >= clusterThreshold) keepTower_R1E = false
  if R2S >= C
  if R2S > C
    keepTower_R1SW = false
    keepTower_R1S  = false
    keepTower_R1SE = false
  if R2SSW > C
    keepTower_R1SW = false
    keepTower_R1S  = false
    keepTower_R2S  = false
    if(R1SW >= clusterThreshold) keepTower_R1W = false
  if R2NNW > C
    keepTower_R1NW = false
    keepTower_R1N  = false
    keepTower_R2N  = false
    if(R1NW >= clusterThreshold) keepTower_R1W = false
  if R3N > C
  if R3N >= C
    keepTower_R1N = false
    if(R2N >= clusterThreshold) keepTower_R1N = false
    keepTower_R2N = false
  if R3NE > C
  if R3NE >= C
    keepTower_R1NE = false
    if(R2NNE >= clusterThreshold) keepTower_R1NE = false
    keepTower_R2N  = false
  if R3NW > C
    keepTower_R1NW = false
    if(R2NNW >= clusterThreshold) keepTower_R1NW = false
    keepTower_R2N  = false
  if R3S >= C
  if R3S > C
    keepTower_R1S = false
    if(R2S >= clusterThreshold) keepTower_R1S = false
    keepTower_R2S = false
  if R3SE >= C
    keepTower_R1SE = false
    if(R2SSE >= clusterThreshold) keepTower_R1SE = false
    keepTower_R2S  = false
  if R3SW > C
    keepTower_R1SW = false
    if(R2SSW >= clusterThreshold) keepTower_R1SW = false
    keepTower_R2S  = false
  if R4N > C
  if R4N >= C
    keepTower_R1N = false
    keepTower_R2N = false
```

```
    if(R3N >= clusterThreshold) keepTower_R2N = false
  if R4S >= C
  if R4S > C
    keepTower_R1S = false
    keepTower_R2S = false
    if(R3S >= clusterThreshold) keepTower_R2S = false
//*****************************************************************//
```

2.4 EGammaNeighborQuality

No change.

3 ClusterTriggerFormer

Both egamma and tau cluster formers are grouped in one single module, with a common part and two specific parts, in order to limit the duplication of sums to be performed. The tau part is basically the same as the old egamma trigger former. For the new egamma part an additional trimming is added on top of the left–right trimming. It depends on the cluster flags as well as on the ieta position and is implemented via a LUT (with 12 bits in inputs and 7 bits in output). This additional trimming makes the trigger former module more complicated due to the necessity to compute first the left–right flag before trimming the cluster flags and computing the final cluster energy from the trimmed cluster flags. It requires more adders as well as more latency from reading the trimming LUT and recomputing partial sums. The updates are detailed below.

Cluster trigger former

```
//*****************************************************************//
// Common tower zero suppression
C    = (C    if keepTower_C    else 0)
R1N  = (R1N  if keepTower_R1N  else 0)
R1NE = (R1NE if keepTower_R1NE else 0)
R1E  = (R1E  if keepTower_R1E  else 0)
R1SE = (R1SE if keepTower_R1SE else 0)
R1S  = (R1S  if keepTower_R1S  else 0)
R1SW = (R1SW if keepTower_R1SW else 0)
R1W  = (R1W  if keepTower_R1W  else 0)
R1NW = (R1NW if keepTower_R1NW else 0)
R2N  = (R2N  if keepTower_R2N  else 0)
R2S  = (R2S  if keepTower_R2S  else 0)
//*****************************************************************//
// Common partial sums
SUMLEFT   = R1NW + R1W + R1SW
SUMCENTRE = R1N  + C   + R1S
SUMRIGHT  = R1NE + R1E + R1SE
SUMEXTEND = R2N  + R2S
//*****************************************************************//
// Common left flags
if iEta<0
  LeftFlag = SUMLEFT > SUMRIGHT
if iEta>0
  LeftFlag = SUMLEFT >= SUMRIGHT
//*****************************************************************//
// Common fine position -- Phi
SUM3UP   = R1NW + R1N + R1NE
SUM3DOWN = R1SW + R1S + R1SE
SUMUP    = SUM3UP + R2N
SUMDOWN  = SUM3DOWN + R2S
if SUMUP>SUMDOWN
  PhiFinePos = UPPER
elsif SUMUP<SUMDOWN
   PhiFinePos = LOWER
else
  PhiFinePos = CENTER
```

```
//*****************************************************************//
// Common fine position -- Eta
if SUMLEFT>SUMRIGHT
  EtaFinePos = LEFT
elsif SUMLEFT=SUMRIGHT
  EtaFinePos = CENTER
else
  EtaFinePos = RIGHT
//*****************************************************************//
// Tau final sum
SUMLEFTRIGHT = (SUMLEFT if LeftFlag else SUMRIGHT)
SUMGLOBAL = SUMLEFTRIGHT + SUMCENTRE + SUMEXTEND
//*****************************************************************//
// EGamma Trimming LUT
keepTower_R1NWorNE = (keepTower_R1NW if LeftFlag else keepTower_R1NE)
keepTower_R1WorE   = (keepTower_R1W  if LeftFlag else keepTower_R1E)
keepTower_R1SWorSE = (keepTower_R1SW if LeftFlag else keepTower_R1SE)
// Since cluster flags are computed here it is not necessary
// to recompute them in ShapeIdentEnergyCalib
ClusterFlag(6) = keepTower_R1NWorNE
ClusterFlag(5) = keepTower_R1WorE
ClusterFlag(4) = keepTower_R1SWorSE
ClusterFlag(3) = keepTower_R2N
ClusterFlag(2) = keepTower_R2S
ClusterFlag(1) = keepTower_R1N
ClusterFlag(0) = keepTower_R1S
iEtaCompress = absoluteValue(iEta)
ClusterFlagTrim = LUT_Trim(ClusterFlag, iEtaCompress) // input: 7+5 bits, output: 7 bits
//*****************************************************************//
// EGamma trimmed partial sums
R1NTRIM  = (R1N  if ClusterFlagTrim(1) else 0)
R1NETRIM = (R1NE if ClusterFlagTrim(6) else 0)
R1ETRIM  = (R1E  if ClusterFlagTrim(5) else 0)
R1SETRIM = (R1SE if ClusterFlagTrim(4) else 0)
R1STRIM  = (R1S  if ClusterFlagTrim(0) else 0)
R1SWTRIM = (R1SW if ClusterFlagTrim(4) else 0)
R1WTRIM  = (R1W  if ClusterFlagTrim(5) else 0)
R1NWTRIM = (R1NW if ClusterFlagTrim(6) else 0)
R2NTRIM  = (R2N  if ClusterFlagTrim(3) else 0)
R2STRIM  = (R2S  if ClusterFlagTrim(2) else 0)
SUMLEFTRIGHTTRIM   = (R1NWTRIM + R1WTRIM + R1SWTRIM if LeftFlag else R1NETRIM + R1ETRIM + R1SETRIM)
SUMCENTRETRIM = R1NTRIM + C + R1STRIM
SUMEXTENDTRIM = R2NTRIM + R2STRIM
//*****************************************************************//
// EGamma final sum
SUMGLOBAL = SUMLEFTRIGHTTRIM + SUMCENTRETRIM + SUMEXTENDTRIM
```

4 Clusters

Sums performed here should not include towers at |iEta| $\geq$ 28. The easiest way to do this (only one file to modify) may be to force to zero the 9×1, 5×1 and 2×1 sums if |iEta| $\geq$ 28, which is what is done below.

Isolation energy sums

```
//*****************************************************************//
// 9x5 tower sums
EHcluster9x1 = EHCluster9x1Former(TowerPipeForClusters)
EHcluster9x1 = EHcluster9x1 if |iEta|<28 else 0
EHcluster9x3 = EHclusterPx3Former(EHcluster9x1Pipe)
EHcluster9x5 = EHclusterXx3Former(EHcluster9x3, EHcluster9x1Pipe)
//*****************************************************************//
// 5x2 tower sums
Ecluster5x1 = ECluster5x1Former(TowerPipeForClusters)
Ecluster5x1 = Ecluster5x1 if |iEta|<28 else 0
Ecluster5x2 = ECluster5x2Former(Ecluster5x1Pipe)
// Selection of Ecluster5x2Left and Ecluster5x2Right from Ecluster5x2Pipe not shown here
//*****************************************************************//
// 2x1 tower sums
```

```
Hcluster2x1 = HCluster2x1Former(TowerPipeForClusters)
Hcluster2x1 = Hcluster2x1 if |iEta|<28 else 0
```

5 TauMergeNeighSearch

Checks adjacent cluster seeds: only max one at time can exist, outputs a bit for each pair according to this information (if none of the element of the pair exist, default value is 0).

For each ϕ ring information is regrouped into 32 computation units that are denoted here with UC. Inside each unit, a bit tells if the cluster is in the upper or lower part: here it is denoted with UC-X.UP (if UP == true it's in the upper part, if it is false it's in the lower part).

Symbols: starting from the central UC and moving north, the two adjacent units are UCN and UC2N. Moving south, they are UCS and UC2S. If it is in the right (east) ring a suffix "E" is added, if it is to the west (left) a "W" is added: e.g. UCNE is the unit that is one step north and one east. See the sketch in figure 2.

The four output bits are O32, O01, O65, O74 (refer to tau algo slides). Default is 0 for all.

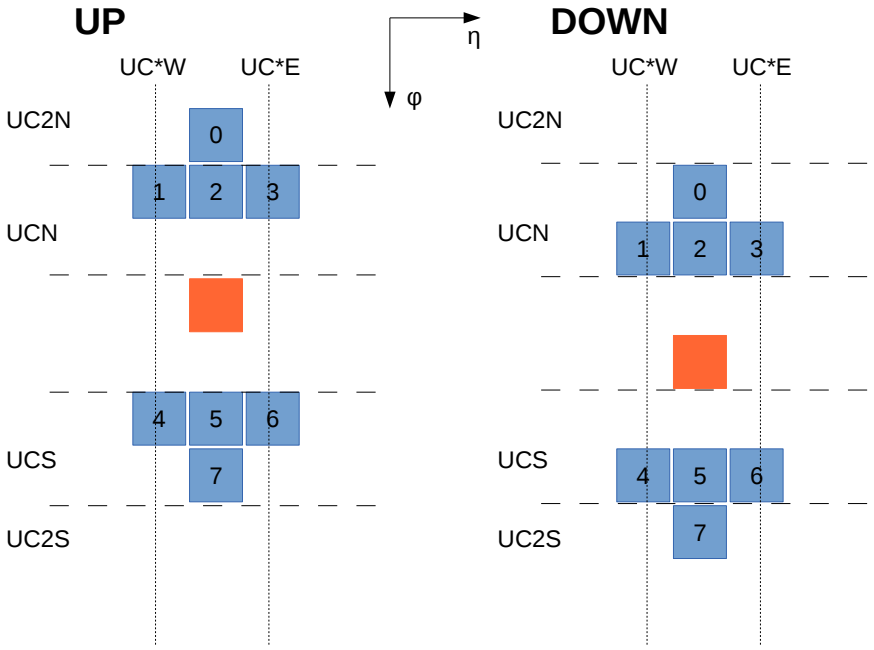

Figure 2: Scheme of computation unit (UC) and neighbour naming for the two cases of central cluster UP (left) and DOWN (right)

Tau merging neighbours existence check

```
//*****************************************************************//
// Init all to default values
O32 = O01 = O65 = O74 = 0
if UP:   // the central cluster seed is in the upper part
  // north side
  if UCN.Center == 1:
    O32 = 1
  if UCNW.Center == 1:
    O01 = 1
  // south side
```

```
  if UCS.UP == 1 && UCS.Center == 1:
    O65 = 1
  if UCSW.UP == 0 && UCSW.Center == 1:
    O74 = 1
else:   // the central cluster seed is in the lower part
  // north side
  if UCN.UP == 0 && UCN.Center == 1:
    O32 = 1
  if UCNW.UP == 1 && UCNW.Center == 1:
    O01 = 1
  // south side
  if UCS.Center == 1:
    O65 = 1
  if UCSW.Center == 1:
    O74 = 1
OUTPUT (O32, O01, O65, O74)
```