

Firmware updates – May 2015 – v0.4

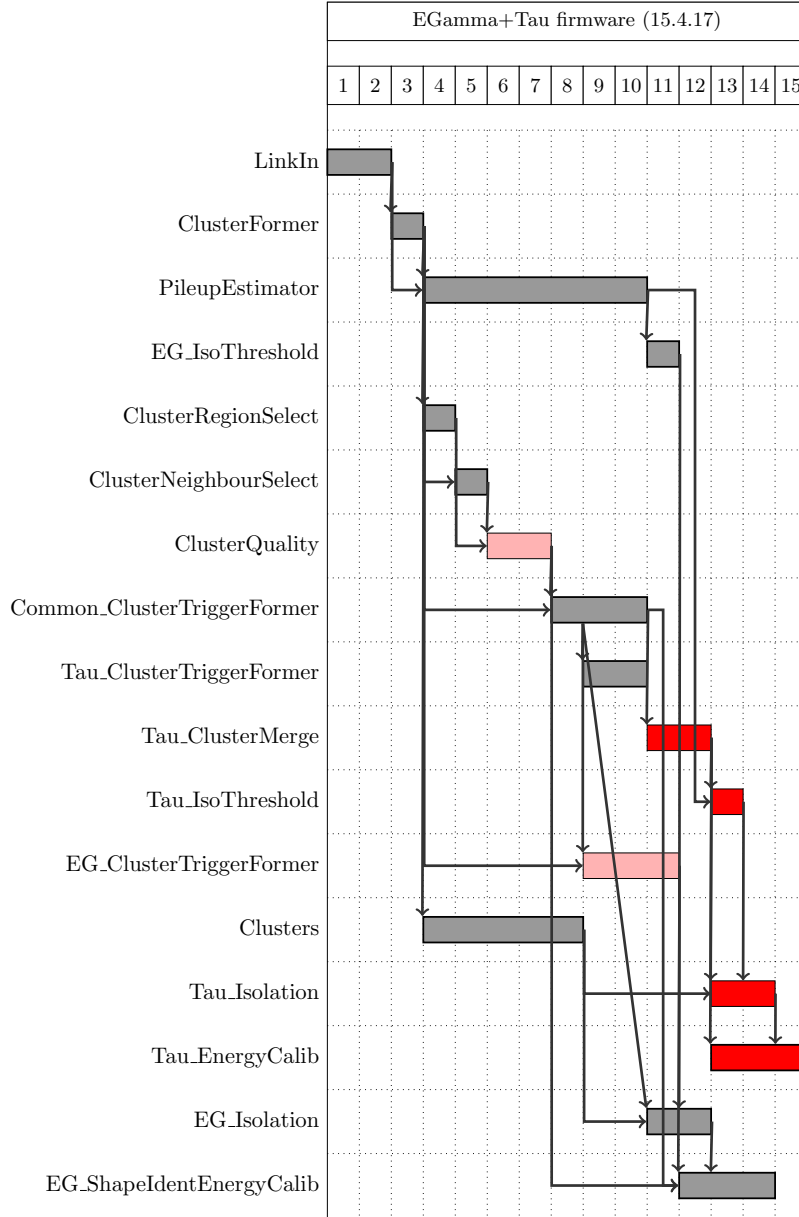


Figure 1: Timing and relation graph of the EGamma and Tau firmware. Each row corresponds to a given module of the algorithm, while the columns indicate the clock number. Gray modules are modules that don't require updates with respect to the current firmware implementation. Pink modules require updates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

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
//*****
// Negative eta
else if(iEta<0)
else if(iEta<-1)
  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
//*****//
// Central boundary
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
//*****//

```

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
        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
//*****//
// 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
        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
//*****//

```

2.4 EGammaNeighborQuality

No change.

3 ClusterTriggerFormer

The logic to create final clusters is currently grouped in one single module called EClusterTriggerFormer. Part of the logic in this module is common to egamma and tau objects, while the second part should be split in two modules specific to egamma and taus. It results in three modules, with one among them being common to egamma and tau.

3.1 CommonClusterTriggerFormer

This module is common to egamma and tau objects. It computes the left/right flag as well as the cluster fine position and no changes are needed with respect to what already exists.

```
----- Common cluster trigger former -----
//*****
// 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)
//*****
// Partial sums
SUMLEFT  = R1NW + R1W + R1SW
SUMCENTRE = R1N + C + R1S
SUMRIGHT = R1NE + R1E + R1SE
SUMEXTEND = R2N + R2S
//*****
// Left flags
if iEta<0
    LeftFlag = SUMLEFT > SUMRIGHT
if iEta>0
    LeftFlag = SUMLEFT >= SUMRIGHT
//*****
// Fine position -- Phi
SUM3UP   = R1NW + R1N + R1NE
SUM3DOWN = R1SW + R1S + R1SE
SUMUP    = SUM3UP + R2N
SUMDOWN  = SUM3DOWN + R2S
if SUMUP>SUMDOWN
    PhiFinePos = UPPER
elseif SUMUP<SUMDOWN
    PhiFinePos = LOWER
else
    PhiFinePos = CENTER
//*****
// Fine position -- Eta
if SUMLEFT>SUMRIGHT
    EtaFinePos = LEFT
elseif SUMLEFT=SUMRIGHT
    EtaFinePos = CENTER
else
    EtaFinePos = RIGHT
```


3.2 TauClusterTriggerFormer

The tau-specific module is a copy of the second part of the current ClusterTriggerFormer module. Since most of the processing is performed in CommonClusterTriggerFormer, only the final energy sum has to be computed here.

```
----- Tau cluster trigger former -----  
//*****  
// Final sum  
SUMLEFTRIGHT = (SUMLEFT if LeftFlag else SUMRIGHT)  
SUMGLOBAL = SUMLEFTRIGHT + SUMCENTRE + SUMEXTEND
```

3.3 EGClusterTriggerFormer

An additional trimming is added on top of the left-right trimming. This additional trimming is specific to egamma clusters. 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 should require one more adder as well as more latency from reading the trimming LUT and recomputing partial sums. The updates are detailed below.

```
----- EG cluster energy sums -----  
//*****  
// 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  
//*****  
// 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  
//*****  
// 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

```