



Stage-2 Tau Algorithm: firmware-oriented pseudocode

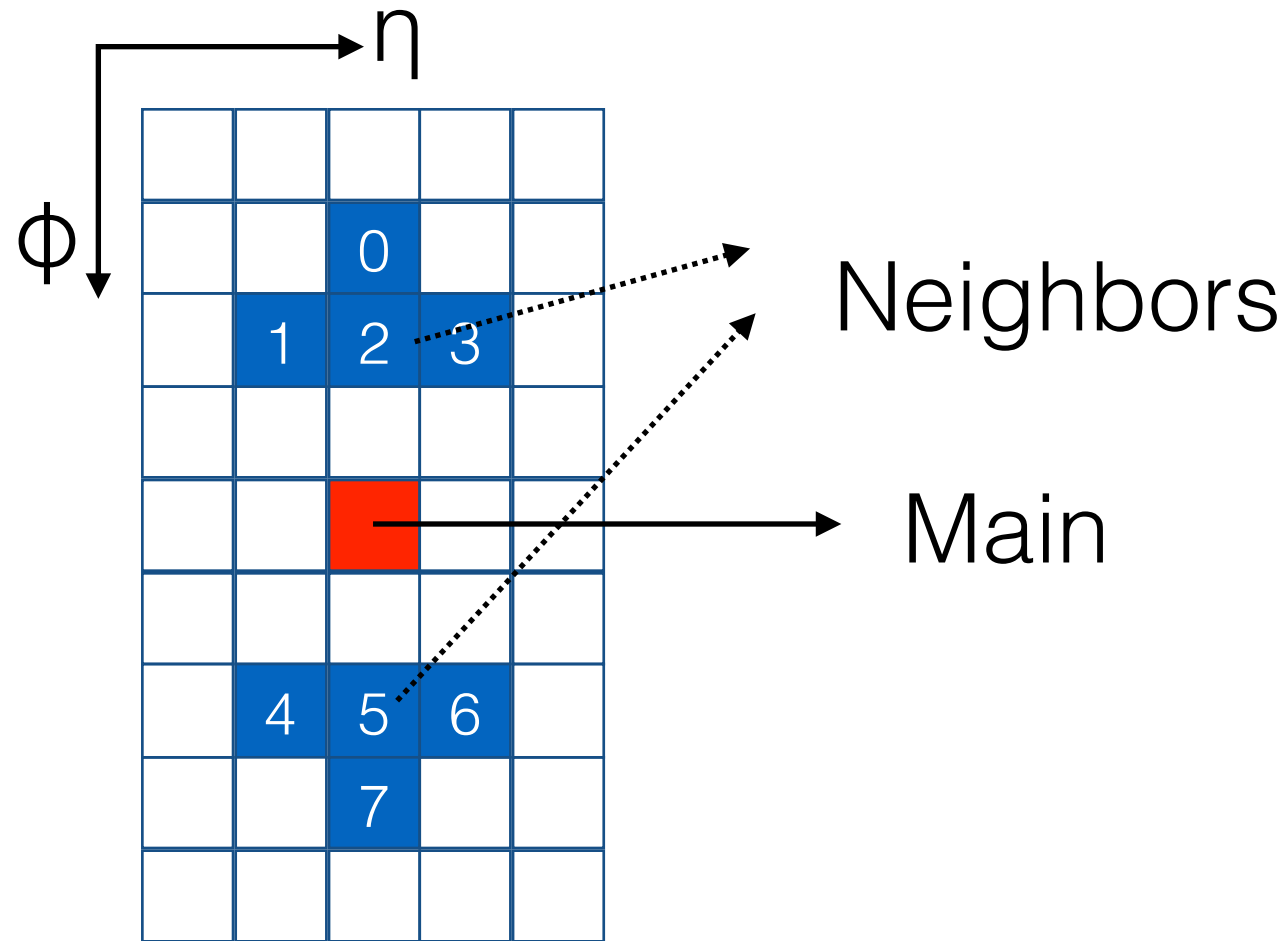
Luca Cadamuro

Laboratoire Leprince-Ringuet (École polytechnique)

Algorithm main steps

- Merging strategy → L1 tau candidate creation
 - 1: Neighbor search
 - 2: Merging
- 3: Calibration
- 4: Isolation
- 5: Shape veto

Introduction - 1



FLAGS DIFFERING FROM EG ALGO:

- + isTauSeed = 1 [DEFAULT]
- H/E
- FG bit

isTau seed is true if the corresponding cluster is the center of a tau L1 candidate (both merged and non merged)

- Squares indicate **clusters** (i.e., size can be larger than 1 TT)
 - the square indicates the cluster position, i.e. the cluster seed TT
- We work on formed clusters
- Neighbors clusters are searched only in the highlighted **pattern**

Introduction - 2

A. Need to find:

1. if at least 1 neighbor exist ←
2. if existing , the neighbor with the highest energy

This search can start earlier than energy comparison as cluster energy is computed in the last step of clustering algo
→ this information is available at an earlier stage

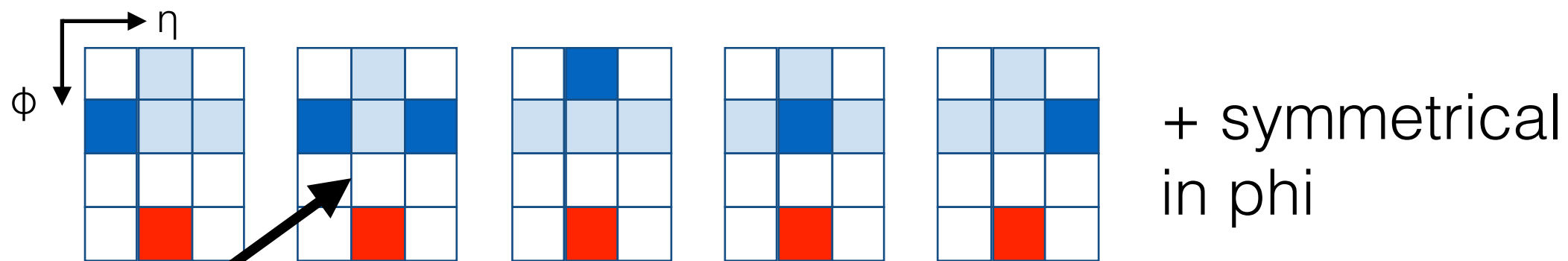
B. **IF** neighbor exist:

1. decide if it is main (if not, isTauSeed = false)
2. add main and neighbor energies if isTauSeed == true

C. **ELSE**

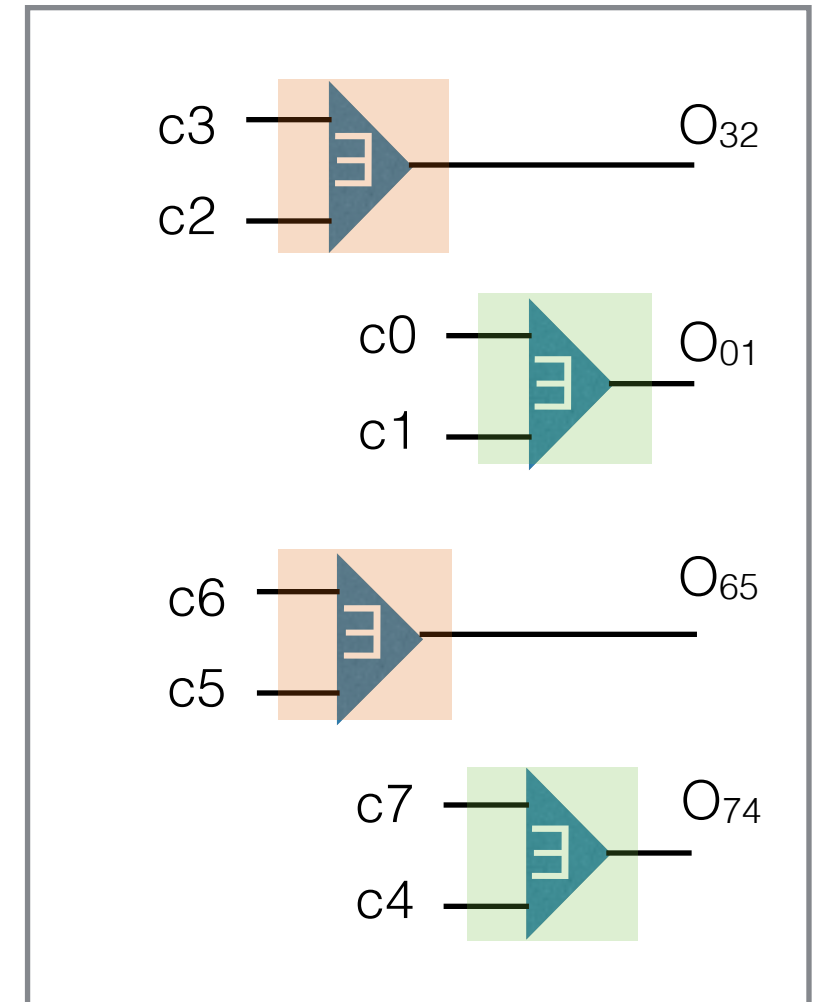
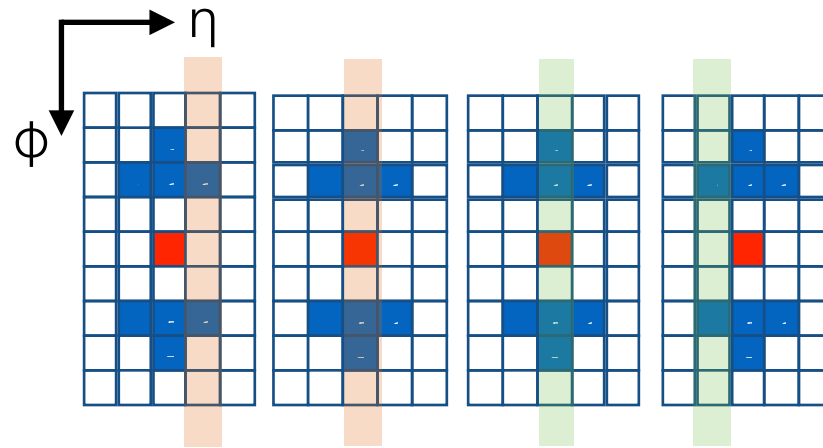
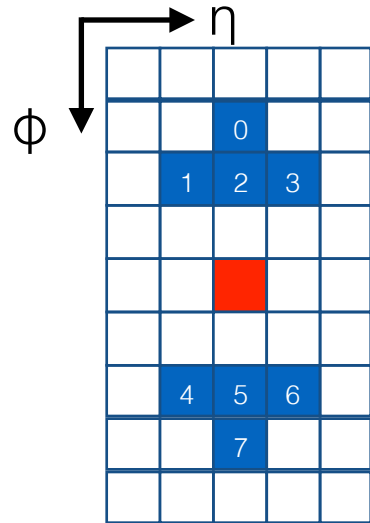
1. it will be a non-merged L1 tau candidate

NOTE: there are limited possibilities for the number of valid neighbors

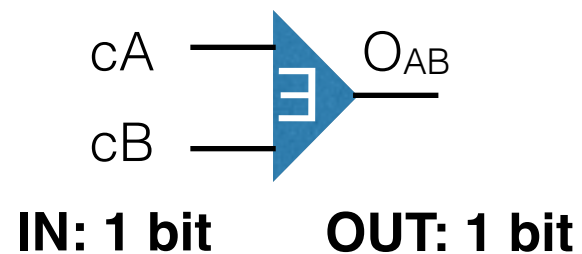


Only this will need more than one comparison on the same phi side

I. Neighbor search



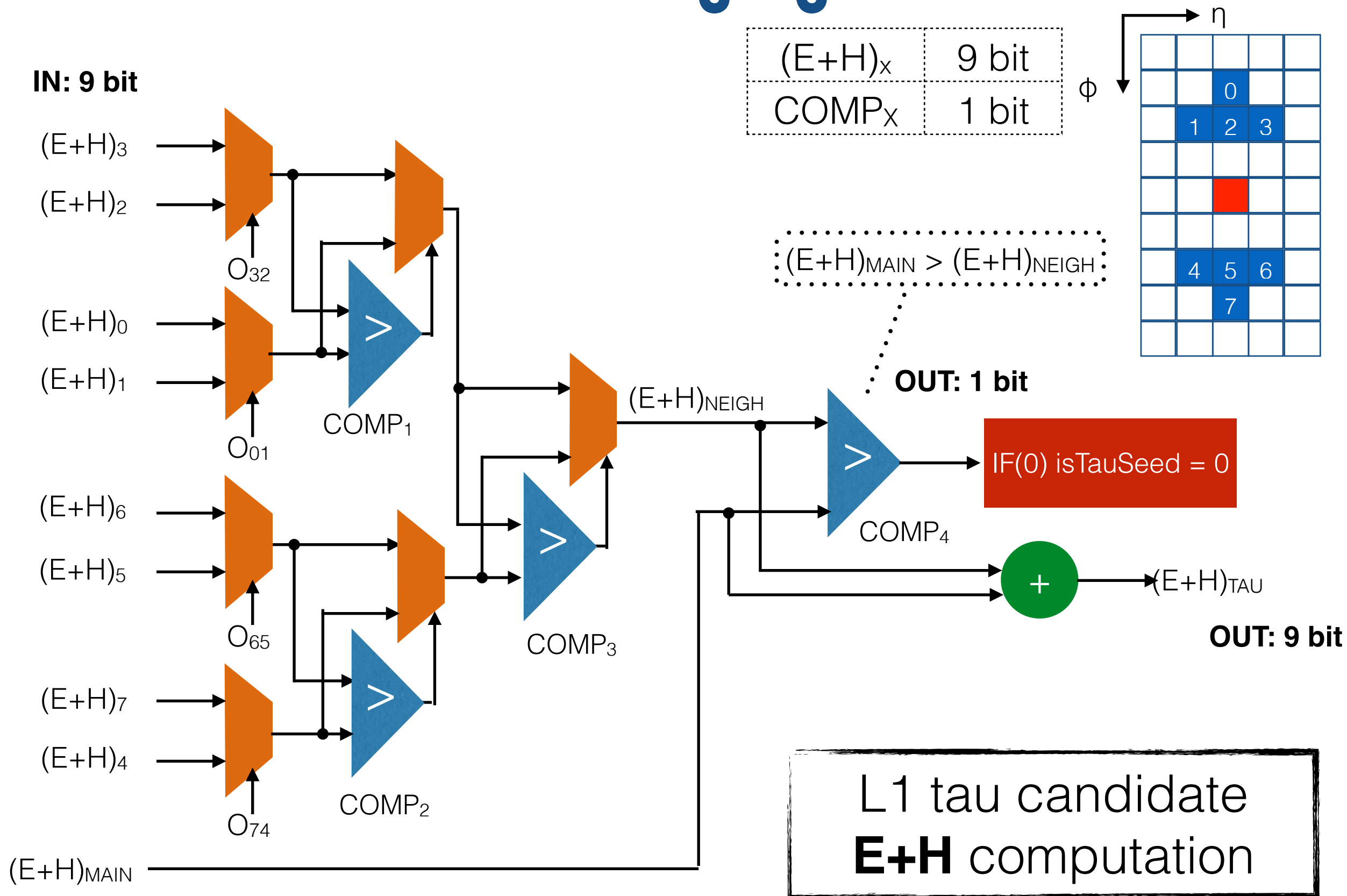
- Input comes from **ClusterQuality** (after Filtering)
- **cX** means the value of Cluster.center (i.e. the flag that states if a cluster exists or not at a certain position)
- Output bit is denoted as O_{AB}



cA	cB	O_{AB}
1	0	0
0	1	1
0	0	0
1	1	NEVER

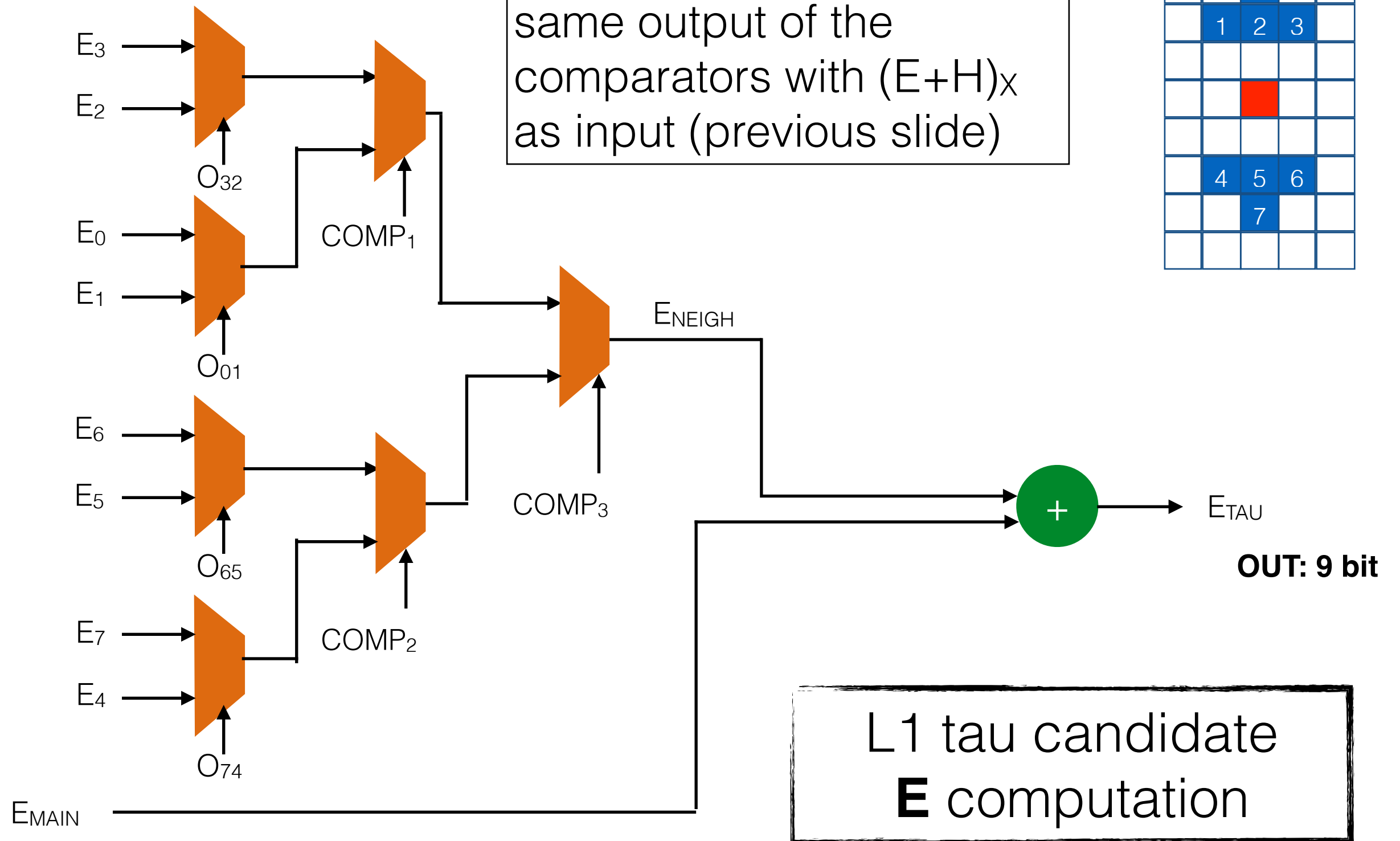
out: 0 \rightarrow cA
 out: 1 \rightarrow cB
**if both are zero,
 force to 0 as default**

II. Merging



II. Merging

IN: 9 bit

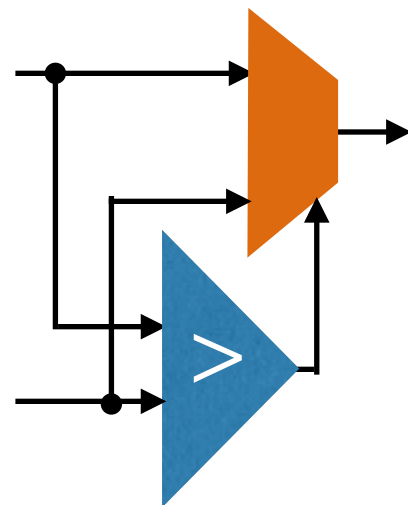
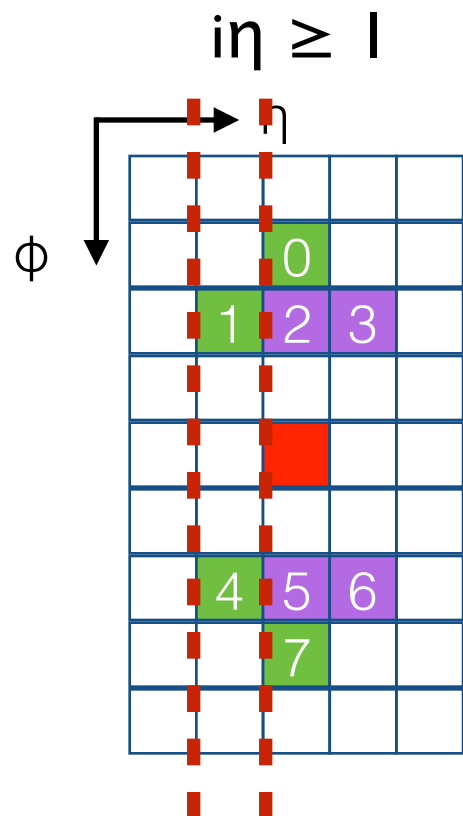


II. Merging [comments]

Symbols used in the previous slide and comments:

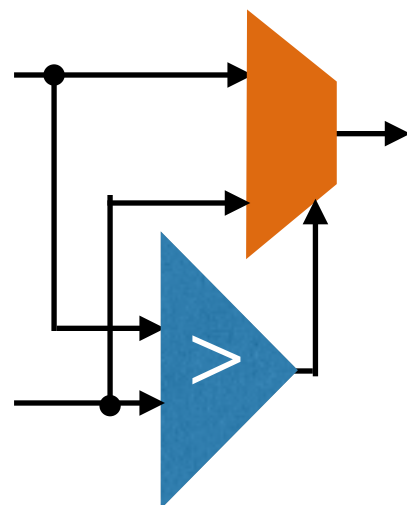
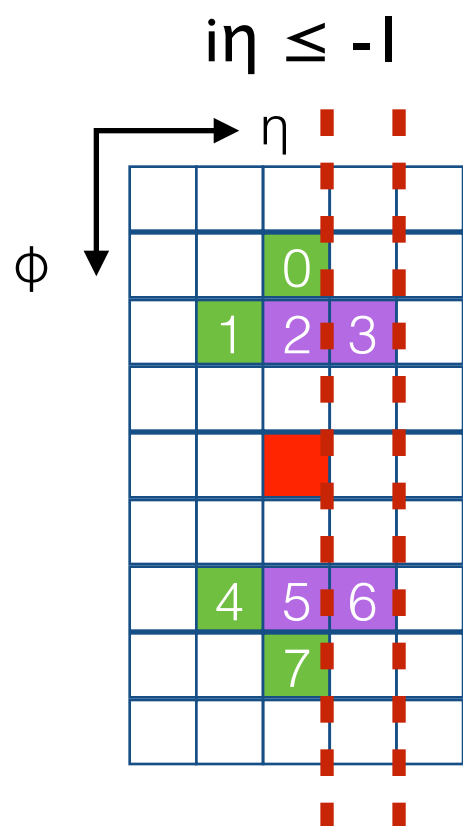
- $(\mathbf{E+H})_x$ denotes the full tau cluster energy resulting from **Tau_ClusterTriggerFormer**
- \mathbf{O}_{xy} denotes the mux output from neighbor search phase
- **NOTE:** if a cluster does not exist (“center” is zero) $(\mathbf{E+H})_x$ and E_x **must be 0** so that the mux selects this value
- Each comparator consists in some “>” and “>=” operation according to the position as in the EG algo (see next slide)
- The current cluster (i.e. the one centered in the computing unit in use) is denoted as “MAIN”
- If no neighbor exists, everything is fine because E_{NEIGH} is 0 and therefore
 - will fail the energy comparison
 - nothing is added to E_{MAIN}

II. Merging [comparator structure - 1]



mux is commanded by the discriminator so that quantities (E, E+H) related to the first input [on the left of the operator in the table] pass if condition is true, else those related to the second input [on the right in the table] pass

$in \geq 1$			
COMP ₁	(E+H) ₂₃	>	(E+H) ₀₁
COMP ₂	(E+H) ₅₆	>	(E+H) ₄₇
COMP ₃	(E+H) ₀₁₂₃	>	(E+H) ₄₅₆₇

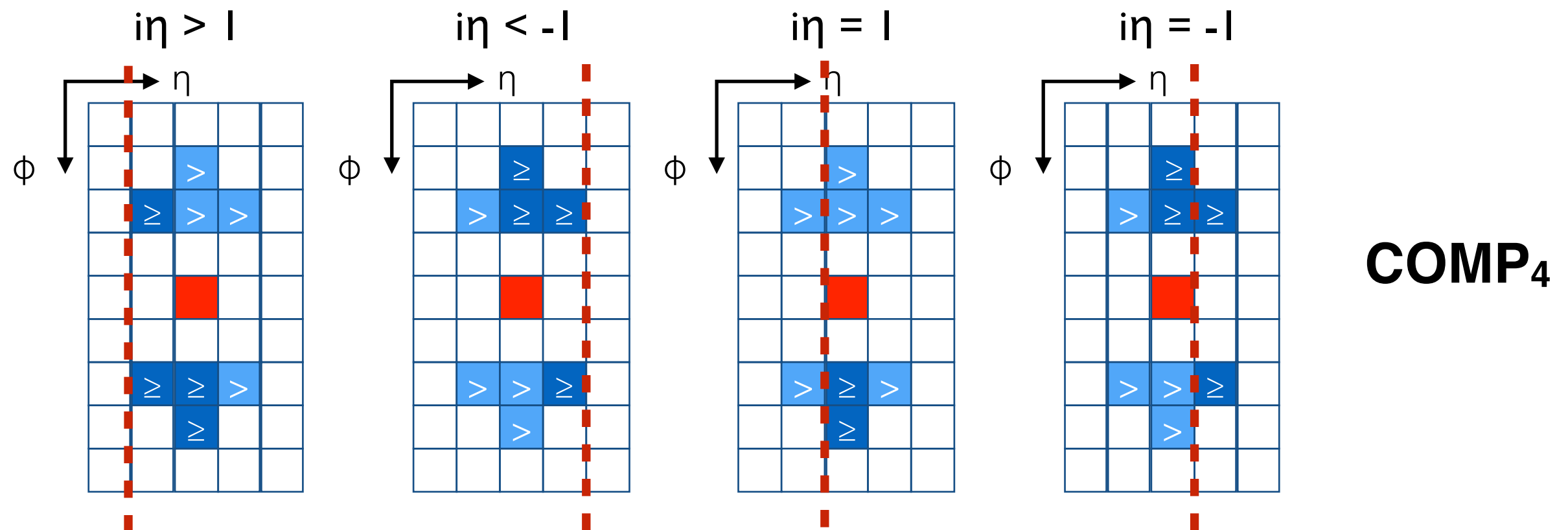


See slide 4: the only comparison on two existing clusters is 1-3, 4-6, so the most central position is favored. Phi values choice is arbitrary

$in \leq -1$			
COMP ₁	(E+H) ₂₃	≥	(E+H) ₀₁
COMP ₂	(E+H) ₅₆	≥	(E+H) ₄₇
COMP ₃	(E+H) ₀₁₂₃	≥	(E+H) ₄₅₆₇

II. Merging [comparator structure - 2]

For COMP_4 (MAIN vs NEIGH comparison) the same logic used in the EG algo is applied



III. Energy calibration

Calibration is computed as a linear function of the hadronic and EM energy deposited (energy of the full cluster)

- a module to compute E , H (had and EM energy) of the cluster is needed

$$E_{calib} = (aE_{EM} + bE_{had} + p) \cdot d(i\eta) \quad (\text{if } E_{EM} > 0)$$

$$E_{calib} = (cE_{had} + q) \cdot d(i\eta) \quad (\text{if } E_{EM} = 0)$$

- **a, b, c** depend on $E_{uncal} = E_{EM} + E_{had}$ and on the barrel/endcap cluster position
- **p, q** depend on the barrel/endcap cluster position
- **d** depends on the position of the cluster in $i\eta$ units

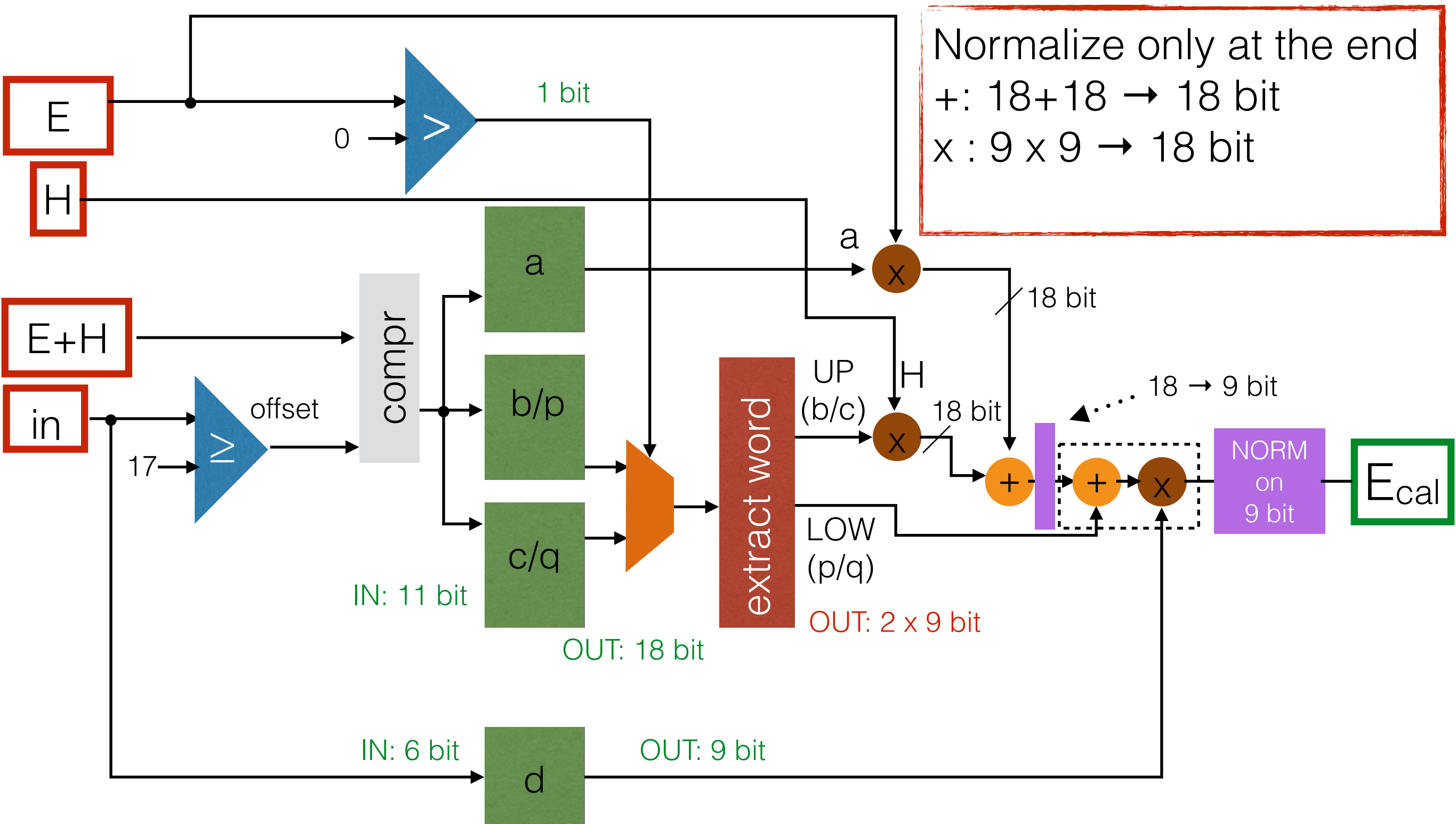
Coefficient values are stored in a LUT.

In the case of merging, input E_{uncal} , E_{EM} , E_{had} are all computed as the sum of the energies of the two merged clusters.

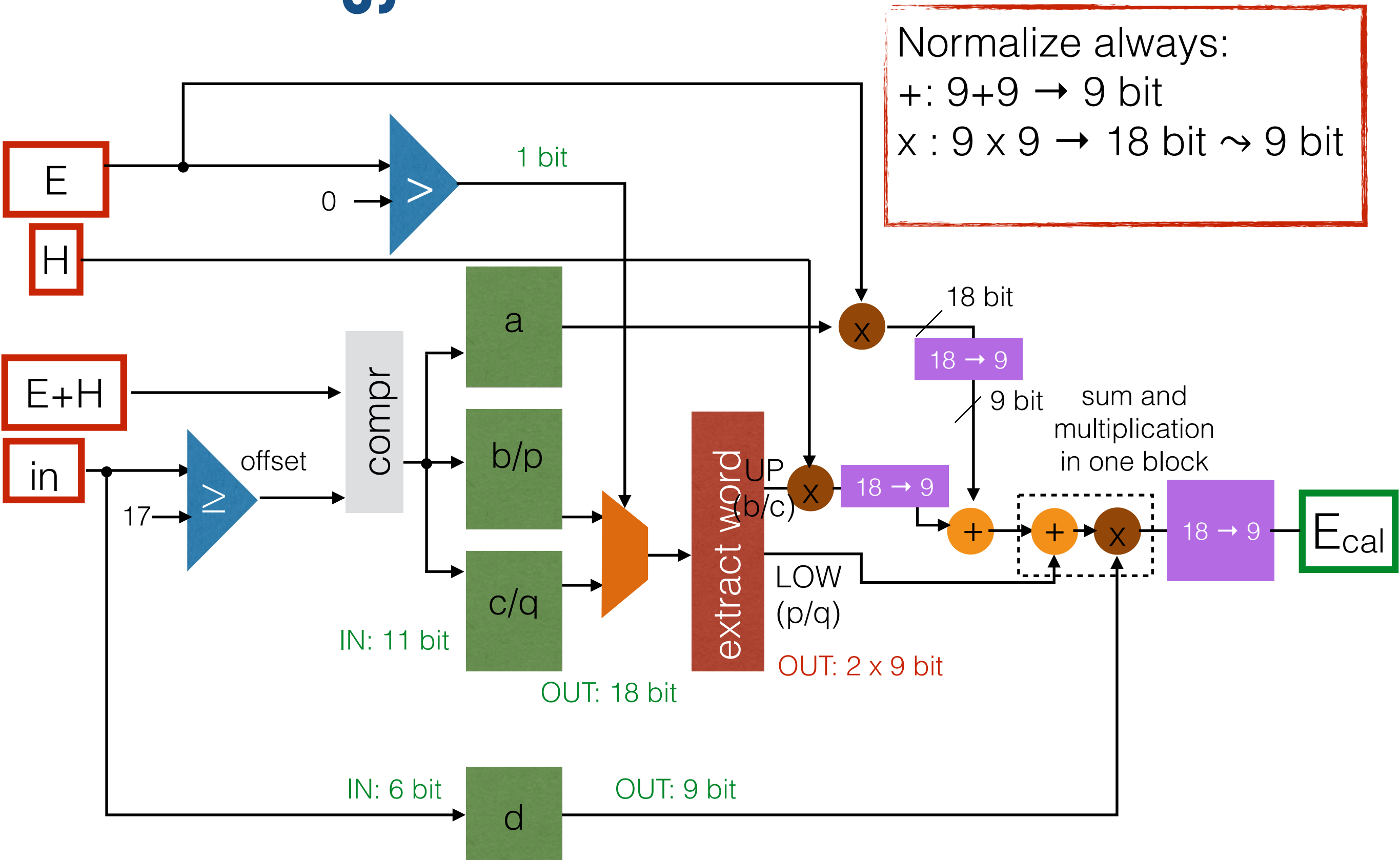
III. Energy calibration

- Input is E , H , $E+H$ of the full tau cluster
- In `Tau_ClusterTriggerFormer`, need to compute $E+H$ and E over the full cluster
 - H is retrieved as $(E+H) - E$

III. Energy calibration – data flow – str. 1



III. Energy calibration – data flow – str. 2



III. Calibration coefficients LUT

- Must contain 3 coefficients
 - A, B, and C
- Either B or C is used for calibration of a cluster; choice is done depending if E_{EM} is zero or not
- Two possibilities to encode the coefficients:
 1. One word with [A, B, C]; extract either B or C depending on the value of E_{EM}
 - This is the best approach for resources usage if enough bits are available for the LUT output
 2. Code A and [B, C] in two different LUT. Extract B or C according to an offset given by E_{EM}
 - In this way also the constants P and Q could be stored in the same word as B or C

IV. Isolation

- Isolation is computed as:

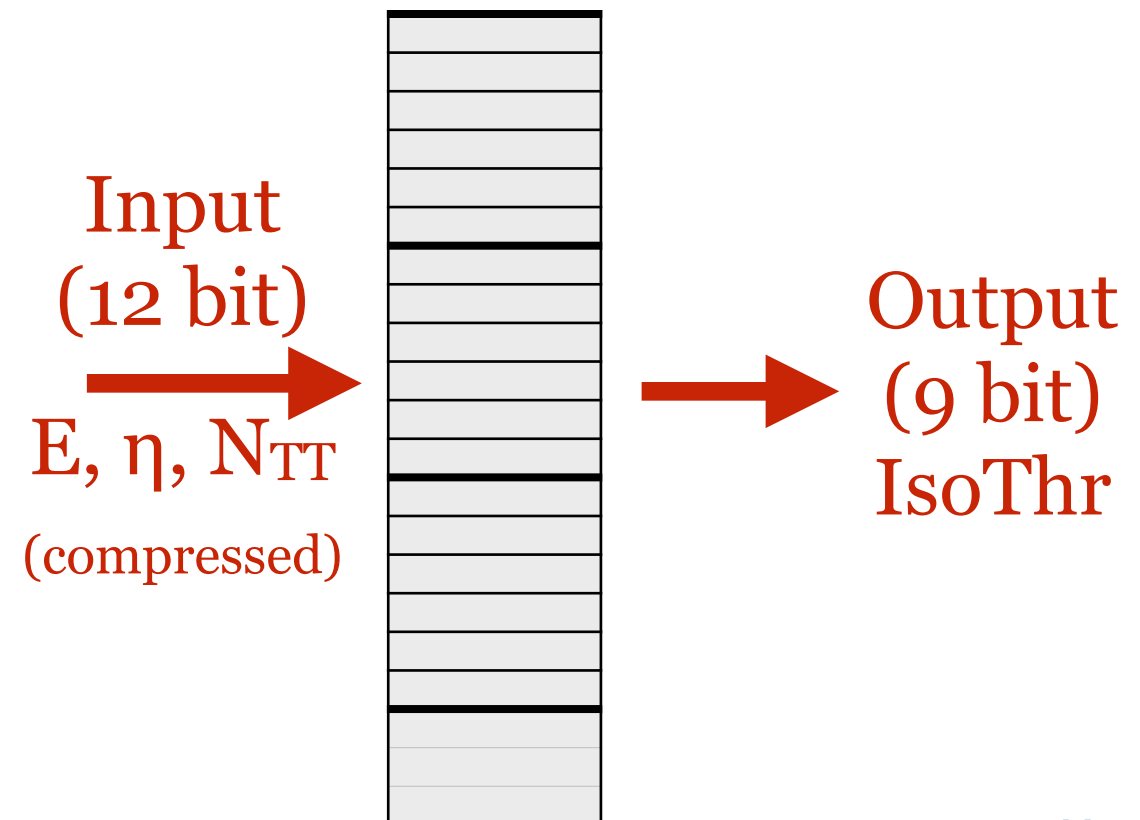
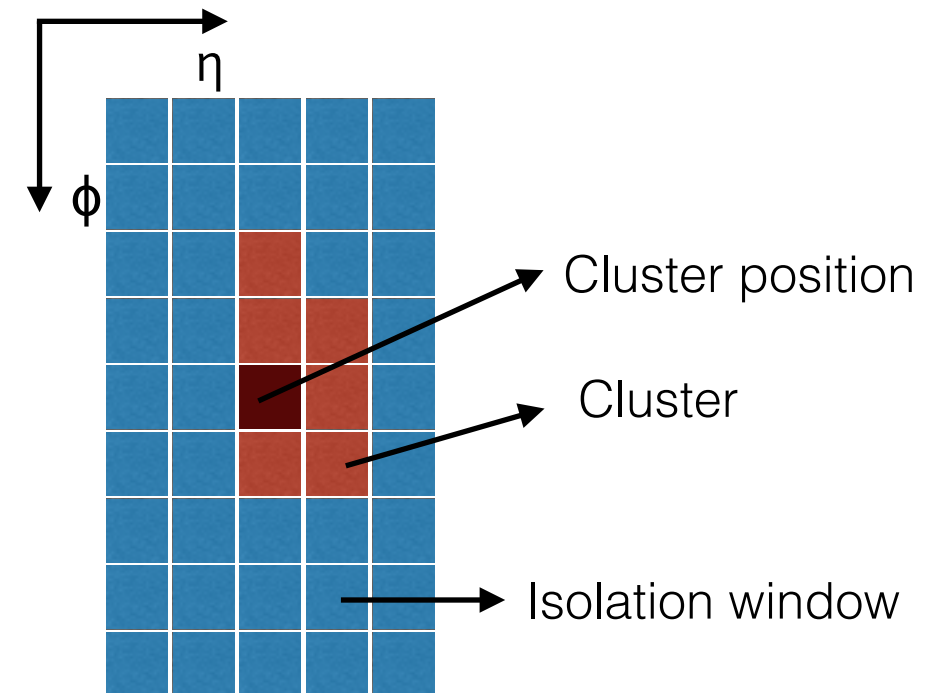
$$E_{5 \times 9} - E_{uncal} \leq IsoThr(E_{uncal}, N_{TT}, \eta)$$

The threshold depends on the cluster energy and η as well as from the pile-up indicator N_{TT} .

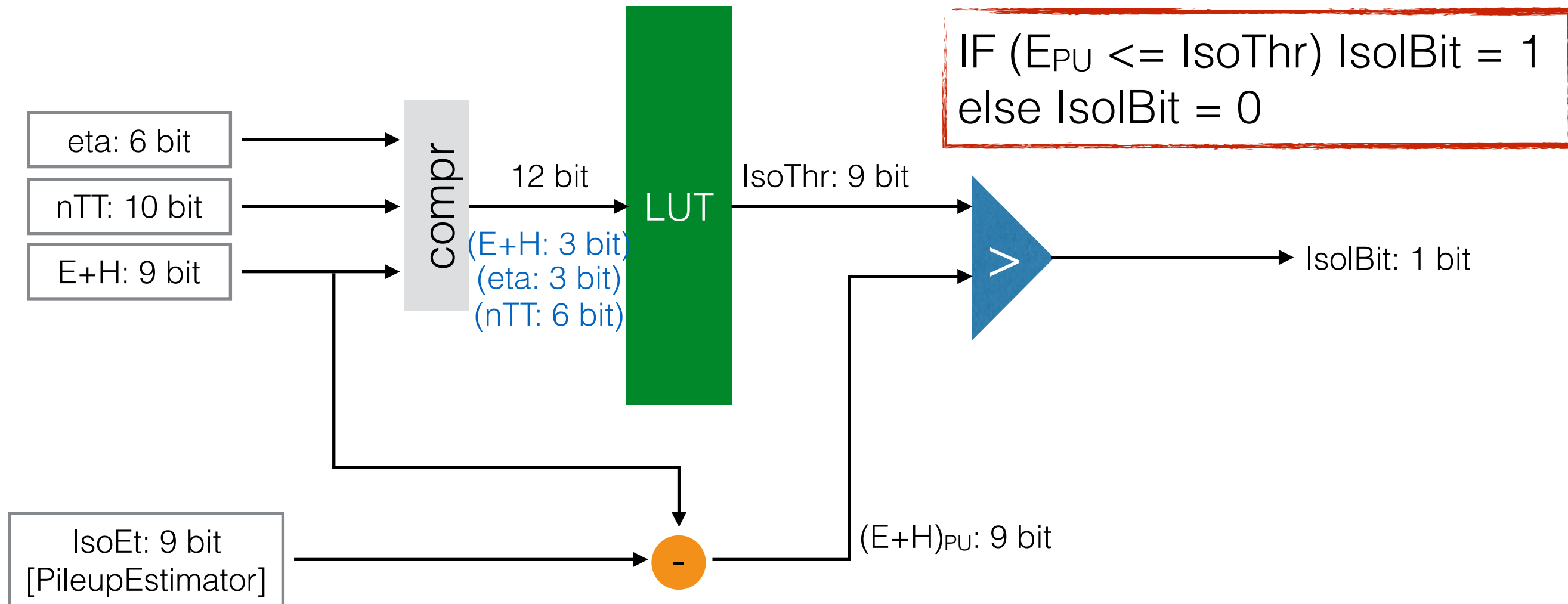
Coded in a LUT. It could be also possible to input just (E, η) in two LUTs to get A, B coefficients to multiply externally by N_{TT} (IsoThr is a linear function of N_{TT} : $IsoThr = A + B N_{TT}$)

N_{TT} = num. of towers with $E > 0$
in the region $\eta \in [-4, +4]$

- The isolation window has a size 5x9 in (η, ϕ) and is **always** centered around the cluster position (computed as the initial cluster seed position)
- Compression has already been developed and tested



IV. Isolation: data flow



V. Shape veto

- Cluster shape is coded in the same way as EG algorithm
- For merged clusters, only the main cluster shape is used at the moment
- Vetoed shapes encoded in a LUT
 - a fixed set of shapes is used at the moment
 - maybe can be optimized as a function of eta (+ other variables?)
- Shape veto is applied at the very end of the tau algorithm (after isolation)
 - we can gain some resources by putting it before isolation computation?