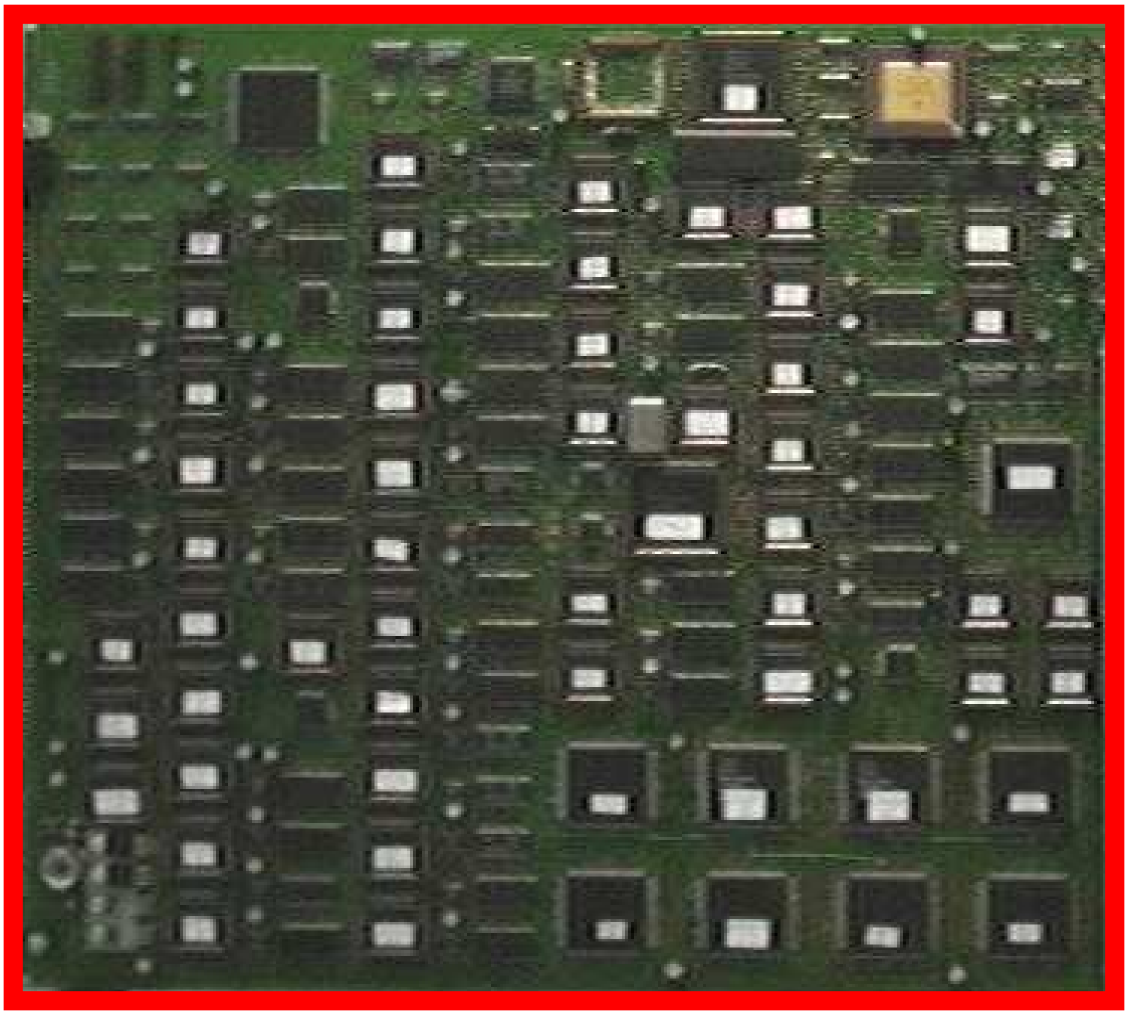
Track Parameter Unit

Documentation

Joachim Flammer

HERA-B

September 26, 2000



In this paper the general calculations and the Lookuptables are presented: Each Lookuptable transfers a set of input bits (which can be splitted to several input values) into 8 output bits. (For convenience we show always 18 bits - sometimes the LUT can take less bits.) Some of the calculations are split to several Lookuptables if the number of input bits exceeds the total possible number of input bits.
The description of the Lookuptables are related to the physical values they calculate. Therefore, some values take several Lookuptables which are described at once, some Lookuptables calculate several output values, therefore they appear several times[1]. In general the Lookuptable part can be split up in four parts:

- (A) Muon and Electron candidates part
- (B) Electron candidates part to apply the Bremsstrahlungs correction
- (C) High Pt candidates part
- (D) γ High E_T candidates part

The special Lookuptables for Electron candidates are also partly used by Muon candidates, however in this case they leave the value unchanged.
For Details about the meaning of the values and the calculations see also extrapolation chart.

General Explanation of coordinates:

- subscript refers to plane in which coordinates lies :
  pos_{Mag} refers to positions in Magnetplane ($Z_{Mag} = 4.50m$)
  pos_{TC2} refers to positions in layer TC2 ($Z_{TC2} = 12.96m$)
- superscript refers to special position (e.g. max = maximum position), 1 refers to intermediate values
- In- and outputbits are always stated from topmost bit (left) to 0th bit (right). x stands for a not used bit.

[1]They even may belong to different parts

Input Message

The TPU gets the message from the TFUs of the last tracking layer PC01. The message contains the following informations:

Value	# of bits	Coding
ξ	10	bit10Xi(XI,TC2)
η	9	bit9Eta(ETA,TC2)
$Dd\xi$	8	bit8Ddxi(DDXI,TPU)
$D\xi$	8	bit8Dxi(DXI,TC2)
W	2	floatW(W,Xi8to5,Eta7to3,ID,TC2)
FLAG	1	... see table below ...
ALL	1	... see table below ...
BX	8	
ID	1	... see table below ...
p	7	bit7P(P)
EBP	7	bit7P(EBP)
EBM	7	bit7(EBM)

- Definition of ID:
  ID specifies the Pretrigger type producing the message.

ID	Description
0	Electron
1	Muon
2	High P_T

- Defintion of FLAG:
  Flag distinguish between γ candidates with very high E_T and electrons (both coming from the ECAL Pretrigger).

FLAG	Description
0	-
1	High $E_T\gamma$ candidate

- Defintion of ALL:
  ALL distinguish between tracks with found hits in all TFU layers and tracks where not all layers had a hit (to be used in connection with FLAG)[2]

ALL	Description
0	All layers had hit
1	Not all layers had hit: High $E_T\gamma$ candidate

[2]Remember if no hit has been found in a TFU, the message is only forwarded if FLAG=1.

Output Message

The output message of the TPU contains the following informations:

Value	# of bits	Coding
ξ	10	... see input message ...
$Dd\xi$	2	bit2Ddxi(DDXI,$granularity_{PC01}$)
$D\xi$	8	... see input message ...
W	2	... see input message ...
FLAG	1	... see input message ...
ALL	1	... see input message ...
BX	8	... see input message ...
ID	1	... see input message ...
$\hat{P}_x = \frac{P_x}{P_z}$	10	floatPxhat(PXHAT)
$\hat{P}_y = \frac{P_y}{P_z}$	9	floatPyhat(PYHAT)
P_z	7	floatPz(PZ)
Q	1	$Q_{bit} = 0 : q = -1$ $Q_{bit} = 1 : q = +1$

Coding and Decoding, value ranges of output values from TPU[3]

$\boxed{P_z}$ (7 bits)

P_z is the z-component of the track momentum at the vertex. It is calculated in the TPU in LUTs 16 to 19 using a fieldmap aproximation. It is coded as a 7 bit value. The physical range of P_z to be coded is

$$5GeV \leq P_z \leq 200GeV$$

To determine the binning one has to take into account the measurement error in P_z due to the limited granularity of the detector. The measurement errors $\frac{\sigma(P_z)}{P_z^2}[GeV^{-1}]$ for the finest granularity, the Inner Tracker MSGC[4] is $7.5 \cdot 10^{-4} \cdot \frac{P^2}{GeV}$, for the inner part of the Outer Tracker[5], the error becomes $3 \cdot 10^{-3} \cdot \frac{P^2}{GeV}$ and for the outer part of the Outer Tracker[6] the error becomes $6 \cdot 10^{-3} \cdot \frac{P^2}{GeV}$.

Range of P_z [GeV]	Detector Part Pz Resolution [GeV]	$\frac{\sigma(P_z)}{P_z^2}[10^{-4}GeV^{-1}]$	Error range $\sigma(P_z)$ [GeV] for max. and min P value [7]
5 ... 10	60	0.15 (5) ... 0.6 (10)	
10 ... 20	30	0.3 (10) ...1.2 (20)	
20 ... 200	7.5	0.3 (20) ... 30 (200)	

An appropriate binning is hence:

$$4.5 \leq P_z < 10\text{GeV} \quad \rightarrow \quad P_z^{bit} = \text{int}\left(45 - \frac{200GeV}{P_z}\right) \tag{1}$$

$$10 \leq P_z < 20\text{GeV} \quad \rightarrow \quad P_z^{bit} = \text{int}\left(65 - \frac{400GeV}{P_z}\right) \tag{2}$$

$$20 \leq P_z \quad \rightarrow \quad P_z^{bit} = \text{int}\left(125 - \frac{1600GeV}{P_z}\right) \tag{3}$$

$$\Rightarrow \Sigma_{binning} \;=\; 25 + 20 + 80 = 125\,\text{bins}$$

The sizes of the bins are always smaller than the measurement errors, accordingly, 125 bins or 7 bits addresses are sufficient. The inverse coding is then:

$$0 \leq P_z^{bit} < 25 \quad \rightarrow \quad P_z = \frac{200GeV}{45 - P_z^{bit}} \tag{4}$$

$$25 \leq P_z^{bit} < 45 \quad \rightarrow \quad P_z = \frac{400GeV}{65 - P_z^{bit}} \tag{5}$$

$$45 \leq P_z^{bit} \quad \rightarrow \quad P_z = \frac{1600GeV}{125 - P_z^{bit}} \tag{6}$$

[3]for values just transferred by the TPU (e.g. $d\xi$) see TFU documentation.
[4]Granularity: 0.12 cm
[5]Granularity: 0.43 cm
[6]Granularity: 0.87 cm

$\boxed{\hat{P}_x}$ (10 bits)

$\hat{P}_x$ is $\frac{P_x}{P_z}$ at the vertex derived in the TPU. It is coded in 10 bits. The most significant bit is used as sign bit. The precision is determined by the precision of ξ in the center of the Magnet since it is derived from this quantity. The precision on ξ is 10 bits. The coding of this variable is done similar to the coding of ξ.[8]

$\boxed{\hat{P}_y}$ (9 bits)

$\hat{P}_y$ is $\frac{P_y}{P_z}$ at the vertex derived in the TPU. It is coded in 9 bits. The most significant bit is used as sign bit. This value is equal to η.[9]. Hence the same precision as for η is required.[10] $\boxed{\hat{P}_y}$

[8] see TFU documentation

[9] times a small correction due to the inhomogeneous magnetic field, which is not taken into account in the FLT

[10] see TFU documentation

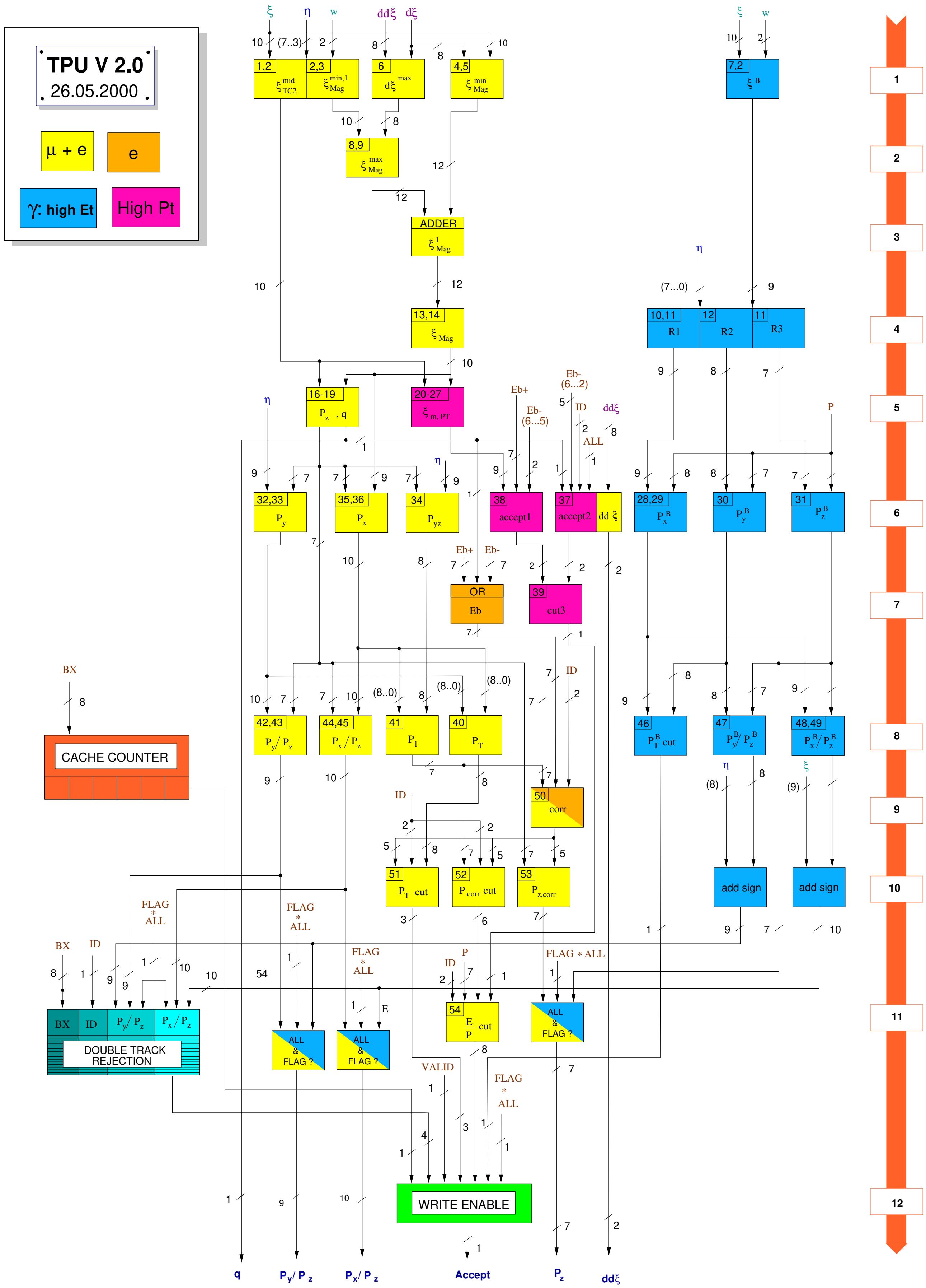


Figure 1: Float Diagram of TPU Lookuptables

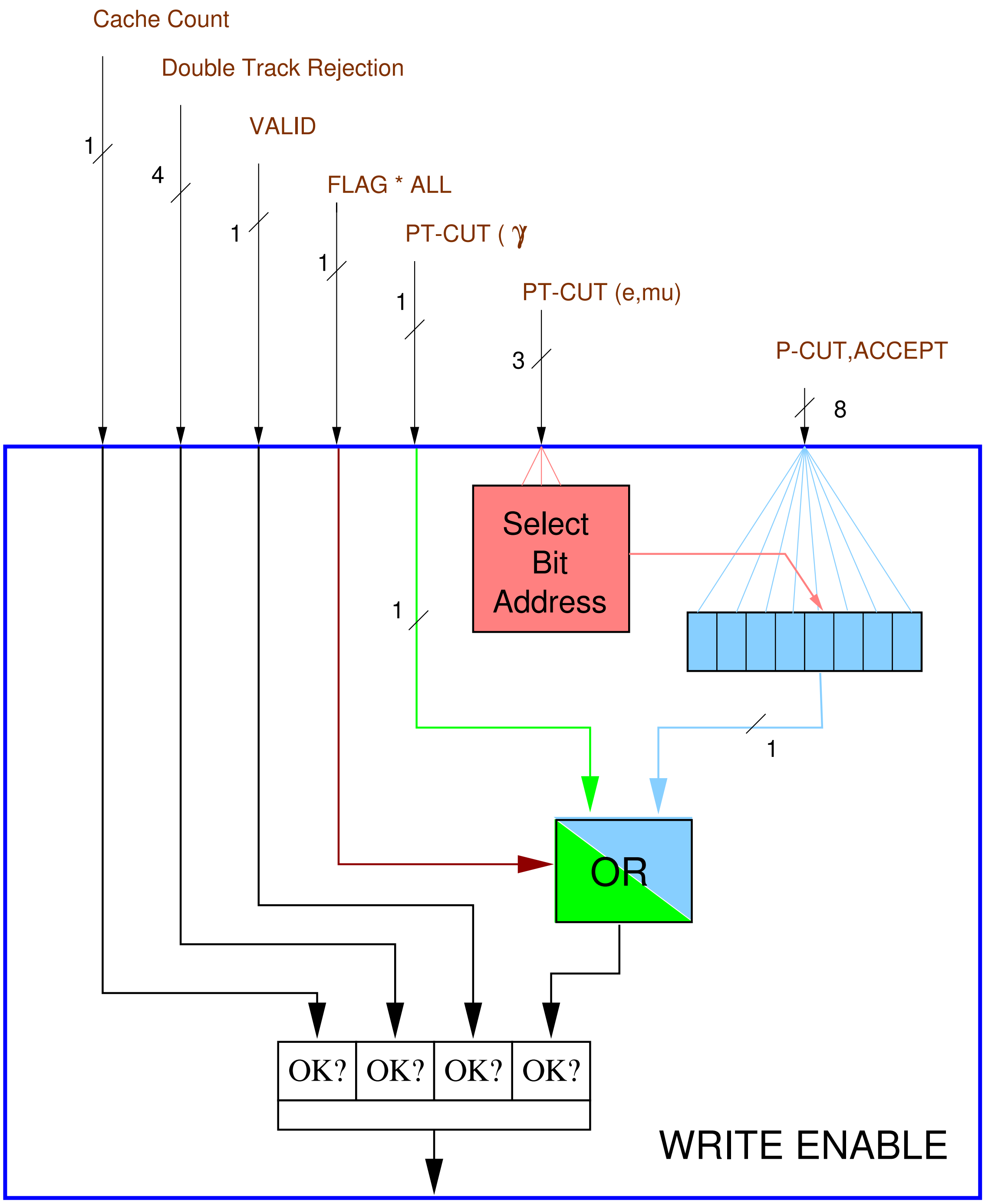


Figure 2: Write enable diagram of TPU

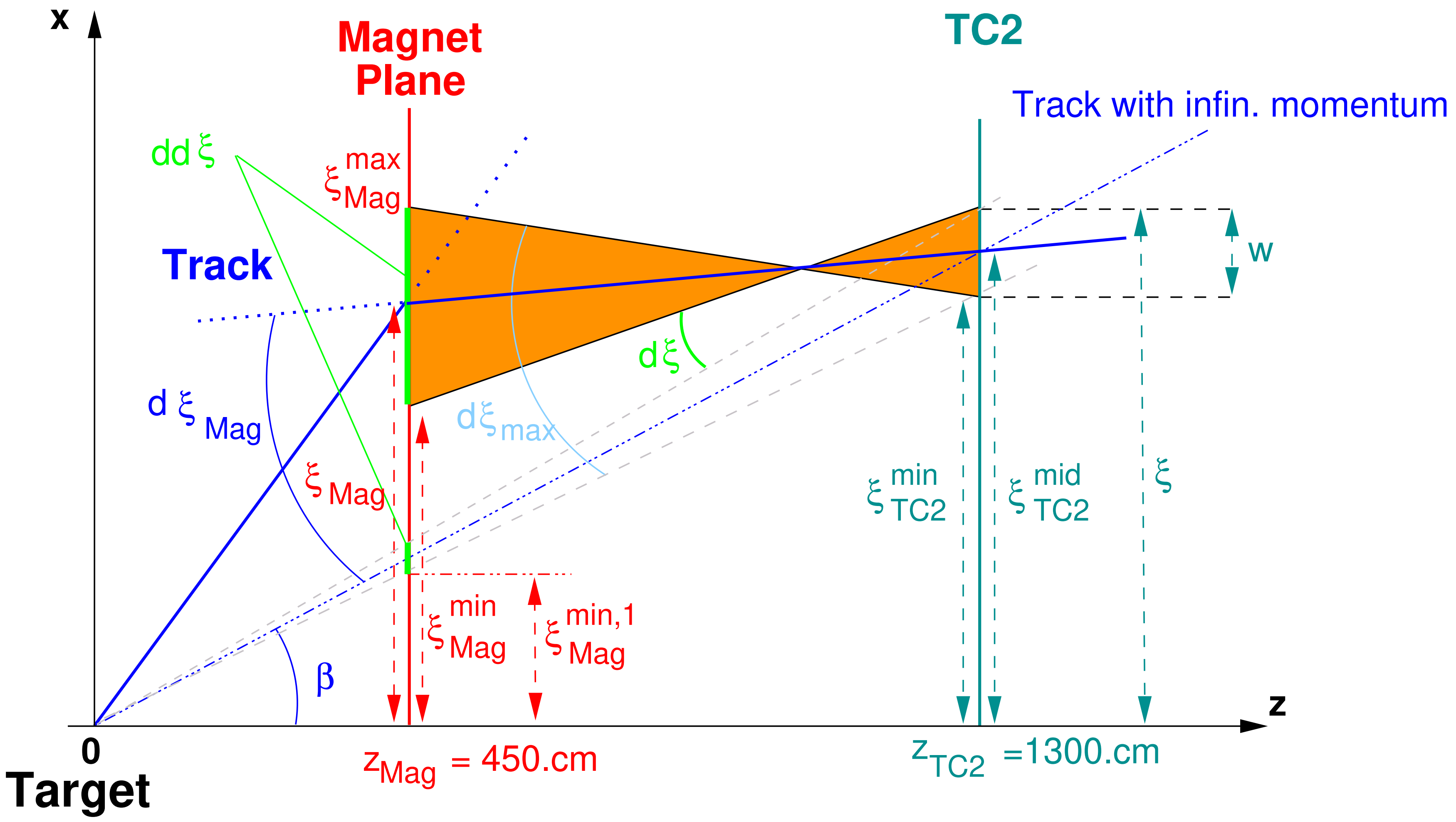


Figure 3: Extrapolation chart for TPU calculations

(A) Electron and Muon Part

LUT 1 and 2: ξ_{TC2}^{mid}

- Calculates the x position of the track in $TC2$.

$$\boxed{\xi_{TC2}^{mid} = \xi - \frac{W}{2}}$$

- Input: ξ (10bit) and η (bits 7.6.5.4.3):

17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
x	7	6	5	4	3	1	0	9	8	7	6	5	4	3	2	1	0
	η					W		ξ									

- Output: 10 bit number using the coding of $bit10Xi(\xi_{TC2}^{mid}, TC2)$:

LUT 1:

7	6	5	4	3	2	1	0
7	6	5	4	3	2	1	0
ξ_{TC2}^{mid}							

LUT 2 [11]:

7	6	5	4	3	2	1	0
x	x	x	8	9	8	9	8
			ξ^B	$\xi_{Mag}^{min,1}$		ξ_{TC2}^{mid}	

[11]see also γ-Part and $\xi_{Mag}^{min,1}$

LUT 2 and 3: $\xi_{Mag}^{min,1}$

- Calculates an intermediate position in the magnet plane to calculate the maximum position of the ROI in the magnet plane (done in LUT 8 and 9).

$$\boxed{\xi_{Mag}^{min,1} = (\xi - W) \cdot \frac{Z_{Mag}}{Z_{TC2}}}$$

- Input: W (2bit) and ξ (10bit)

17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
x	7	6	5	4	3	1	0	9	8	7	6	5	4	3	2	1	0
	η					W		ξ									

- Output: 10 bit number using the coding of $bit10Xi(\xi_{Mag}^{min,1}, TPU)$:

LUT 2[12]:

7	6	5	4	3	2	1	0
x	x	x	8	9	8	9	8
			ξ^B	$\xi_{Mag}^{min,1}$		ξ_{TC2}^{mid}	

LUT 3:

7	6	5	4	3	2	1	0
7	6	5	4	3	2	1	0
$\xi_{Mag}^{min,1}$							

[12] see also γ-Part and $\xi_{TC2}^{mid,1}$

LUT 4 and 5: ξ_{Mag}^{min}

- Calculates the minimum x position of the ROI in the magnet plane.

$$\boxed{\xi_{Mag}^{min} = \xi \cdot \frac{Z_{Mag}}{Z_{TC2}} + d\xi \cdot (Z_{TC2} - Z_{Mag})}$$

- Input: dξ (8 bit) and ξ (10 bit)

17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0
$d\xi$								ξ									

- $d\xi$ is coded as a pseudo-angle: distance in magnet plane divided by z-distance between magnet plane and TC2.

- Output: 12 bit number using the coding of $bit12XiMag$:

LUT 4:

7	6	5	4	3	2	1	0
7	6	5	4	3	2	1	0
ξ_{Mag}^{min}							

LUT 5:

7	6	5	4	3	2	1	0
x	x	x	x	11	10	9	8
ξ_{Mag}^{min}							

LUT 6: $d\xi^{max}$

$$\boxed{d\xi^{max} = \frac{dd\xi}{(Z_{TC2} - Z_{Mag})} + d\xi}$$

- $dd\xi$ is coded as a distance in the magnet plane. The LUT only takes Bit 0 to 5 for the calculations (see also LUT 37 for details).
- $d\xi$ is coded as a pseudo-angle: distance in magnet plane divided by z-distance between magnet plane and TC2.

- Input: dξ (8 bit) and ddξ (8 bit)

17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
7	6	5	4	3	2	1	0	x	x	7	6	5	4	3	2	1	0
$d\xi$										$dd\xi$							

- Output: 8 bit number coded using the way of $d\xi$: $bit8Dxi(d\xi^{max}, TC2)$

LUT 6:

7	6	5	4	3	2	1	0
7	6	5	4	3	2	1	0
$d\xi^{max}$							

LUT 8 and 9: ξ_{Mag}^{max}

- Calculates the maximum x position of the ROI in the magnet plane.

$$\boxed{\xi_{Mag}^{max} = \xi_{Mag}^{min,1} + d\xi^{max} \cdot (Z_{TC2} - Z_{Mag})}$$

- Input: $\xi_{Mag}^{min,1}$ (10 bit) and dξ^{max} (8 bit)

17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0
$d\xi^{max}$								$\xi_{Mag}^{min,1}$									

- Output: 12 bit number using coding of $bit12XiMag$

LUT 8:

7	6	5	4	3	2	1	0
7	6	5	4	3	2	1	0
ξ_{Mag}^{max}							

LUT 9:

7	6	5	4	3	2	1	0
x	x	x	x	11	10	9	8
				ξ_{Mag}^{max}			

ADDER: ξ^1_{Mag}

- Input: ξ^{max}_{Mag} (12 bit) and ξ^{min}_{Mag} (12 bit)
- Only Adder - no LUT
- Calculations:

$$\boxed{\xi^1_{Mag} = \xi^{max}_{Mag} + \xi^{min}_{Mag}}$$

- Output: 12 bit uses special coding (see decoding function in Lut 13 and 14)

LUT 13 and 14: ξ_{Mag}

- Calculates the track x position in the magnet plane and transfers coding from 12 to 10 bit:

$$\boxed{\xi_{Mag} = \frac{\xi^1_{Mag}}{2}}$$

- Input: ξ^1_{Mag} (12 bit)

17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
x	x	x	x	x	x	11	10	9	8	7	6	5	4	3	2	1	0
						ξ^1_{Mag}											

- Output: 10 bit number using the coding of $bit10Xi(\xi_{Mag}, TPU)$

LUT 13:

7	6	5	4	3	2	1	0
7	6	5	4	3	2	1	0
ξ_{Mag}							

LUT 14:

7	6	5	4	3	2	1	0
x	x	x	x	x	x	9	8
						ξ_{Mag}	

LUT 16 to 19: P_z and Q

- Calculates the momentum P_{xz} in xz-direction by using an approximation field map (pxz2xdx) and extracts the charge:

$$\xi_{Mag}^{\infty} = \xi_{TC2}^{mid} \cdot \frac{Z_{Mag}}{Z_{TC2}}$$

$$\frac{d_x}{d_z} = \frac{\xi_{TC2}^{mid} - \xi_{Mag}}{Z_{TC2} - Z_{Mag}}$$

$\frac{d_y}{d_z} = 0$ (can be used as correction for pxz2xdx but is not known in this LUT)

$$P_{xz} = pxz2xdx(\xi_{Mag}, \frac{d_x}{d_z}, \frac{d_y}{d_z}, Z_{Mag})$$

$$P_z = \frac{P_{xz}}{\sqrt{(1 + (\frac{\xi_{Mag}}{Z_{Mag}})^2}}$$

if $\xi_{Mag} \geq \xi_{Mag}^{\infty} \rightarrow$ Electron : negative Charge q=-1 (Bit $Q = 0$)
if $\xi_{Mag} < \xi_{Mag}^{\infty} \rightarrow$ Positron : positive Charge q=+1 (Bit $Q = 1$)

- Input ξ_{TC2}^{mid} (10bit) and ξ_{Mag} (10 bit)

17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0
ξ_{Mag}								ξ_{TC2}^{mid}									

- LUT to be used depends on bit 8 and 9 of ξ_{Mag} (and is steered by switch in front of Luts):

bit 9	bit 8	LUT
0	0	16
0	1	17
1	0	18
1	1	19

- Output: 7 bit number for P_z using the coding $bit7Pz$ and 1 bit number Q:

	7	6	5	4	3	2	1	0
LUT 16, 17, 18, 19:	0	6	5	4	3	2	1	0
	Q	P_z						

LUT 32 and 33: P_y

- Calculates the y-projection of the momentum:

$$\boxed{P_y = P_z \cdot \eta}$$

- Input η (9 bit) and P_z (7 bit)

17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
x	x	8	7	6	5	4	3	2	1	0	6	5	4	3	2	1	0
		η									P_z						

- Remark: Due to redesign we have a special output of P_y. For LUT 42 and 43 the full 10bits of P_y are used, for LUT 40 only the bits 9..1!

- Output: 10bit number using coding $bit10Py$:

LUT32:

7	6	5	4	3	2	1	0
8	7	6	5	4	3	2	1
P_y							

LUT33:

7	6	5	4	3	2	1	0
x	x	x	x	x	x	0	9
						P_y	

LUT 34: P_{yz}

- Calculates the yz-projection of the momentum:

$$\boxed{P_{yz} = P_z \cdot \sqrt{1+\eta^2}}$$

- Input P_z (7 bit) and η (9 bit)

17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
x	x	8	7	6	5	4	3	2	1	0	6	5	4	3	2	1	0
		η									P_z						

- Output: 8bit number using the coding $bit8Pyz$.

LUT34:

7	6	5	4	3	2	1	0
7	6	5	4	3	2	1	0
P_{yz}							

LUT 35 and 36: P_x

- Calculates the x-projection of the momentum:

$$\boxed{P_x = P_z \cdot \frac{\xi_{Mag}}{Z_{Mag}}}$$

- Input P_z (7 bit) and ξ_{Mag} (10 bit)

17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
x	9	8	7	6	5	4	3	2	1	0	6	5	4	3	2	1	0
	ξ_{Mag}										P_z						

- Remark: Due to redesign we have a special output of P_x. For LUT 44 and 45 the full 10bits of P_y are used, for LUT 40 only the bits 9..1!

- Output: 10 bit number using the coding of $bit10Px$.

LUT35:

7	6	5	4	3	2	1	0
8	7	6	5	4	3	2	1
P_x							

LUT36:

7	6	5	4	3	2	1	0
x	x	x	x	x	x	0	9
						P_x	

LUT 37: $dd\xi$

- Reduces $dd\xi$ to 2 bits (more are not needed after TPU)
- The reduction is already done in the TFU of PC01 in the following way:

  6 Bits (Bit 5.4.3.2.1.0) are used to store the full $dd\xi$ information for the calculations in the TPU (see LUT 6).

  2 Bits (Bit 7.6) are used to store $dd\xi$ in such a special way that one can reconstruct $dd\xi_{out}$ out of this two bits and the rest of the FLT Message. The two bits of $dd\xi$ are calculated in the following way:

$$fraction = int(dd\xi \cdot \frac{1}{cellsize_{PC01}} \cdot \frac{Z_{TC02} - Z_{PC01}}{Z_{TC02} Z_{Mag}})$$

	$dd\xi_{out}$
$fraction \leq 1.5$	0
$1.5 \leq fraction \leq 2.5$	1
$2.5 \leq fraction \leq 3.5$	2
$fraction \geq 3.5$	3

- The TPU then just transfers bit 7 and 6 of $dd\xi$ as bit 1 and 0 of $dd\xi_{out}$.

- Input $dd\xi$ (7 bit) and E_b^- (bit 6..2) and Q (1 bit) and ALL (1 bit) and ID (2 bit)

17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
x	7	6	5	4	3	2	1	0	6	5	4	3	2	0	1	1	0
	$dd\xi$								E_b^-					Q	ALL	ID	

- Output: 2bit number $dd\xi_{out}$ using $bit2Ddxi$.

LUT37:

7	6	5	4	3	2	1	0
1	0	x	x	x	x	1	0
$Accept_B$						$dd\xi_{out}$	

LUT 40: P_T

- Calculates the transverse momentum:

$$\boxed{P_T = \sqrt{P_x^2 + P_y^2}}$$

- Input P_y (10 bit) and P_x (10 bit)

17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
8	7	6	5	4	3	2	1	0	8	7	6	5	4	3	2	1	0
P_x									P_y								

- Output: 8bit number using the coding of $bit8Pt$:

LUT40:

7	6	5	4	3	2	1	0
7	6	5	4	3	2	1	0
P_T							

LUT 41: P_1

- Calculates the total momentum

$$\boxed{P_1 = \sqrt{P_x^2 + P_{yz}^2}}$$

- Input P_{yz} (8 bit) and P_x (10 bit)

17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
x	7	6	5	4	3	2	1	0	8	7	6	5	4	3	2	1	0
	P_{yz}								P_x								

- Output: 7 bit number using the coding of $bit7P$.

LUT 41:

7	6	5	4	3	2	1	0
7	6	5	4	3	2	1	0
P_1							

LUT 42 and 43: $\hat{P}_y (= \eta)$

- Calculates $\hat{P}_y$:

$$\boxed{\hat{P}_y = \frac{P_y}{P_z} (= \eta)}$$

- Input P_y (10 bit) and P_z (7 bit)

17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
x	0	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1
	P_y	P_z							P_y								

- Output: 9bit number using the coding $bit9Pyhat$

LUT 42:

7	6	5	4	3	2	1	0
7	6	5	4	3	2	1	0
$\hat{P}_y$							

LUT 43:

7	6	5	4	3	2	1	0
x	x	x	x	x	x	x	8
							$\hat{P}_y$

LUT 44 and 45: $\frac{P_x}{P_z}$

- Calculates $\frac{P_x}{P_z}$:

$$\boxed{\hat{P}_x = \frac{P_x}{P_z}}$$

- Input P_x (10 bit) and P_z (7 bit)

17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
x	0	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1
	P_x	P_z							P_x								

- Output: 10 bit number using the coding of $10Pxhat$:

LUT 44:

7	6	5	4	3	2	1	0
7	6	5	4	3	2	1	0
$\hat{P}_x$							

LUT 45:

7	6	5	4	3	2	1	0
x	x	x	x	x	x	9	8
						$\hat{P}_x$	

LUT 51: $P_T cut$

- Applies P_T cut and steers with the output which bit of the Lookuptable 54 is used for write enable check:
  (In brackets nominal cut values).
  if (ID = $electron$)

if $P_T >$ TPU_CUT_PT_ELECTRON (500 MeV)

Output =

2	1	0
1	1	1

else

Output =

2	1	0
0	0	0

else if (ID = $muon$)

if $P_T >$ TPU_CUT_PT_MUON (500 MeV)

Output =

2	1	0
1	1	1

else

Output =

2	1	0
0	0	0

else if (ID = $highP_T$)

to be done

- Input P_T (8 bit) and ID (2 bit) and Corr (5 bit):

17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
x	x	x	7	6	5	4	3	2	1	0	1	0	4	3	2	1	0
			P_T								ID		$Corr$				

- Output: 3 bit number:

LUT 51:

7	6	5	4	3	2	1	0
x	x	x	x	x	2	1	0
					$P_T cut$		

LUT 52: $P_{corr} cut$

- Applies the P cut on the Momentum corrected by the Bremsstrahlungs correction.
- If Track is accepted, LUT transfers absolut value of P_1 Momentum (not P_{corr} (!)) to LUT 54 (for $\frac{E}{P}$ cut which is done with uncorrected P).
(In brackets nominal cut values).

$$P_{corr} = P_1 \cdot Corr$$

if (TPU_MIN_P_CUT (5 GeV) $\leq P_{Corr} \leq$ TPU_MAX_P_CUT (200 GeV)):

$$\text{Output} = \mid P_1 \mid$$

else

$$\text{Output} = 0$$

- The TPU_MIN_P_CUT and TPU_MAX_P_CUT cut can be set for each particle ID seperately.

- Input P_1 (6 bit) and ID (2 bit) and Corr (5 bit):

17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
x	x	x	x	6	5	4	3	2	1	0	1	0	4	3	2	1	0
				P_1							ID		$Corr$				

- Output: 8 bit number using coding of $bit6P$ (without sign):

	7	6	5	4	3	2	1	0
LUT 52:	x	6	5	4	3	2	1	0
		$\mid P_1 \mid$						

LUT 53: $P_{z,corr}$

- Corrects P_z by Bremsstrahlungs lost:

$$\boxed{P_{z,corr} = P_z \cdot corr}$$

- Input P_z (7 bit) and Corr (5 bit):

17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
x	x	x	x	x	x	6	5	4	3	2	1	0	4	3	2	1	0
						P_z							$Corr$				

- Output: 7 bit number using the coding $bit7Pz$:

	7	6	5	4	3	2	1	0
LUT 53:	x	6	5	4	3	2	1	0
		$P_{z,corr}$						

LUT 54: $\frac{E}{P}$ *cut*

- Applies the $\frac{E}{|P_1|}$ cut for electrons

(In brackets nominal cut values)
if $\mid P_1 \mid < 0$

Output =

7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0

else if (ID = $electron$)

if (TPU_MIN_E_P (0.7) $\leq \frac{E}{|P_1|} \leq$ TPU_MAX_E_P (1.3)):

Output =

7	6	5	4	3	2	1	0
1	1	1	1	1	1	1	0

else

Output =

7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0

else if (ID = $highP_T$)

to be done ....

if (cut3 = 1)

Output =

7	6	5	4	3	2	1	0
1	1	1	1	1	1	1	0

else

Output =

7	6	5	4	3	2	1	0
1	1	1	1	1	1	1	0

- Input ID (2 bit) and P (7 bit) and P_{corr} (7 bit) and cut3 (1bit)

17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
x	6	5	4	3	2	1	0	0	6	5	4	3	2	1	0	1	0
	P							$Cut3$	$\mid P_1 \mid$							ID	

- Output: 7 bit number

LUT 54:

7	6	5	4	3	2	1	0
7	6	5	4	3	2	1	0
$P_{cut,accept}$							

(B) Bremsstrahlungscorrection for Electron Part

OR: Decide on E_{b+}, E_{b-}

- Depending on Q take $E_{b+} or E_{b-}$

$$\begin{array}{l} \text{if } Q = 0 \text{ (Electron)} \rightarrow E_b = E_{b-} \\ \text{if } Q = 1 \text{ (Positron)} \rightarrow E_b = E_{b+} \end{array}$$

- Output: 7 bit number: Energie using the coding $bit7P$

LUT 50: corr

- Calculates the Bremsstrahlungs correction factor:

$$\begin{array}{ll} \text{if (ID} = electron\text{):} & Corr = 1 + \frac{Eb}{P_1} \\ \text{else} & Corr = 1 \end{array}$$

- Input: P_1 (7 bit) and E_b (7 bit) and ID (2 bit)

17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
x	x	6	5	4	3	2	1	0	6	5	4	3	2	1	0	1	0
		P_1							E_b							ID	

- Output: 5 bit number using the coding $bit5Corr$

LUT 50:

7	6	5	4	3	2	1	0
x	x	x	4	3	2	1	0
			$Corr$				

(C) γ High E_t Part

All the calculation are done first without using the sign. The sign of P_x^B and P_y^B is added in Pipestep 10 and 11. The calculations basically do a straight line extrapolation and determine P_x^B, P_y^B and P_z^B out of this straight, from the target coming, line since Photons are not affected by the magnetic field.

LUT 2 and 7: ξ^B

- Calculates the x position in $TC2$

$$\xi^{B,W} = \xi(bitXi) - \xi(bitXi - 1) \text{ (get granularity)}$$

$$sign = \frac{\xi}{|xi|}$$

$$\xi^B = \mid \xi + (sign \cdot W \cdot \frac{\xi^{B,W}}{2}) \mid$$

- In this LUT we have the problem that W can only be decoded if we know ξ and η. However this LUT has no information about η.

- Input: ξ (10bit) and η (bits 7.6.5.4.3):

17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
x	x	x	x	x	x	1	0	9	8	7	6	5	4	3	2	1	0
(η)						W		ξ									

- Output: 9 bit number using $bit9Xi1$:

LUT 2 [13]:

7	6	5	4	3	2	1	0
x	x	x	8	9	8	9	8
			ξ^B	$\xi_{Mag}^{min,1}$		ξ_{TC2}^{mid}	

LUT 7:

7	6	5	4	3	2	1	0
7	6	5	4	3	2	1	0
ξ^B							

[13] see also Muon/Electron-Part

LUT 10 and 11: R_1

- Calculates the projection factor for projecting P on x-direction:

$$\boxed{R_1 = \frac{\xi^B}{\sqrt{\xi^{B2} + (1+\eta^2)\cdot Z_{TC2}^2}}}$$

$$(= \frac{X_{TC2}}{\sqrt{X_{TC2}^2 + Y_{TC2}^2 + Z_{TC2}^2}})$$

- Input: η (8 bit - without sign) and ξ^B (9 bit)

17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
x	7	6	5	4	3	2	1	0	8	7	6	5	4	3	2	1	0
	η								ξ^B								

- Output: 9 bit number using coding $bit9R1$

LUT 10:

7	6	5	4	3	2	1	0
7	6	5	4	3	2	1	0
R_1							

LUT 11:

7	6	5	4	3	2	1	0
8	6	5	4	3	2	1	0
R_1	R_3						

LUT 12: R_2

- Calculates the projection factor for projecting P on y-direction:

$$\boxed{R_2 = \frac{\eta \cdot Z_{TC2}}{\sqrt{\xi^{B^2} + (1+\eta^2) \cdot Z_{TC2}^2}}}$$

$$(= \frac{Y_{TC2}}{\sqrt{X_{TC2}^2 + Y_{TC2}^2 + Z_{TC2}^2}})$$

- Input: η (8 bit - without sign) and ξ^B (9 bit)

17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
x	7	6	5	4	3	2	1	0	8	7	6	5	4	3	2	1	0
	η								ξ^B								

- Output: 8 bit number using coding $bit9R2$

LUT 12:

7	6	5	4	3	2	1	0
7	6	5	4	3	2	1	0
R_2							

LUT 11: R_3

- Calculates the projection factor for projecting P on z-direction:

$$\boxed{R_3 = \frac{Z_{TC2}}{\sqrt{\xi^{B^2} + (1+\eta^2) \cdot Z_{TC2}^2}}}$$

$$(= \frac{Z_{TC2}}{\sqrt{X_{TC2}^2 + Y_{TC2}^2 + Z_{TC2}^2}})$$

- Input: η (8 bit - without sign) and ξ^B (9 bit)

17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
x	7	6	5	4	3	2	1	0	8	7	6	5	4	3	2	1	0
	η								ξ^B								

- Output: 7 bit number using coding $bit9R3$

LUT 11:

7	6	5	4	3	2	1	0
8	6	5	4	3	2	1	0
R_1	R_3						

LUT 28 and 29: P_x^B

- Calculates absolute value[14] of Momentum in x-direction:

$$\boxed{P_x^B = \mid P \cdot R_1 \mid}$$

- Input: R_1 (9 bit) and P (7 bit)

17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
x	x	8	7	6	5	4	3	2	1	0	6	5	4	3	2	1	0
		R_1									P						

- Output: 9 bit number using coding of $bit9PxB$

LUT 28:

7	6	5	4	3	2	1	0
7	6	5	4	3	2	1	0
P_x^B							

LUT 29:

7	6	5	4	3	2	1	0
x	x	x	x	x	x	x	8
							P_x^B

LUT 30: P_y^B

- Calculates absolute value[15] of Momentum in y-direction:

$$\boxed{P_y^B = \mid P \cdot R_2 \mid}$$

- Input: R_2 (8 bit) and P (9 bit)

17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
x	x	x	7	6	5	4	3	2	1	0	6	5	4	3	2	1	0
			R_2								P						

- Output: 8 bit number using coding of $bit8PyB$

LUT 30:

7	6	5	4	3	2	1	0
7	6	5	4	3	2	1	0
P_y^B							

[14] The sign is added later in pipe 10 to $\hat{P}_x$.
[15] The sign is added later in pipe 10 to $\hat{P}_y$.

LUT 31: P_z^B

- Calculates Momentum in z-direction:

$$\boxed{P_z^B = P \cdot R_3}$$

- Input: R_3 (7 bit) and P (7 bit)

17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
x	x	x	x	6	5	4	3	2	1	0	6	5	4	3	2	1	0
				R_3							P						

- Output: 7 bit number using coding of $bit7Pz$

	7	6	5	4	3	2	1	0
LUT 31:	x	6	5	4	3	2	1	0
		P_z^B						

LUT 46: $P_{T,cut}$

- Calculates transverse momentum P_T and applies P_T cut:

$$\boxed{P_T = \sqrt{{P_x^B}^2 + {P_y^B}^2}}$$

$$P_T \geq \text{TPU\_CUT\_PT\_GAMMA (3.5 GeV) ?}$$

- Input: P_x^B (bit 8) and P_y^B (bit9)

17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
x	7	6	5	4	3	2	1	0	8	7	6	5	4	3	2	1	0
	P_y^B								P_x^B								

- Output: 1 bit number Accept (1 = Accept, 0 = Not Accept)

	7	6	5	4	3	2	1	0
LUT 46:	x	x	x	x	x	x	x	0
								Accept

LUT 47: $\frac{P_y^B}{P_z^B}$

- Calculates $\hat{P_y^B} = \frac{P_y^B}{P_z^B}$:

$$\boxed{\hat{P_y^B} = \frac{P_y^B}{P_z^B}}$$

- Input: P_z^B (7 bit) and P_y^B (8 bit)

17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
x	7	6	5	4	3	2	1	0	x	x	6	5	4	3	2	1	0
	P_y^B										P_z^B						

- Output: 7 bit number using coding of $bit7pyhat$

	7	6	5	4	3	2	1	0
LUT 47:	7	6	5	4	3	2	1	0
	$\hat{P_y^B}$							

LUT 48 and 49: $\frac{P_x^B}{P_z^B}$

- Calculates $\hat{P_x^B} = \frac{P_x^B}{P_z^B}$:

$$\boxed{\hat{P_x^B} = \frac{P_x^B}{P_z^B}}$$

- Input: P_x^B (9 bit) and P_z^B (7 bit)

17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
x	x	6	5	4	3	2	1	0	8	7	6	5	4	3	2	1	0
		P_z^B							P_x^B								

- Output: 9 bit number using coding $bit9Pxhat$

	7	6	5	4	3	2	1	0
LUT 48:	7	6	5	4	3	2	1	0
	$\hat{P_x^B}$							

	7	6	5	4	3	2	1	0
LUT 49:	x	x	x	x	x	x	x	8
								$\hat{P_x^B}$

$\boxed{\textbf{Add Sign to } \hat{P_y^B}}$

- Adds sign to $\frac{P_y^B}{P_z^B}$ by combining topmost bit (bit 8) of η to $\frac{P_y^B}{P_z^B}$
- Input: $\frac{P_y^B}{P_z^B}$ (8 bit)
- Output: $\hat{P_y^B} = \frac{P_y^B}{P_z^B}$ (9 bit)

$\boxed{\textbf{Add Sign to } \hat{P_x^B}}$

- Adds sign to $\frac{P_x^B}{P_z^B}$ by combining topmost bit (bit 9) of ξ to $\frac{P_x^B}{P_z^B}$
- Input: $\frac{P_x^B}{P_z^B}$ (9 bit)
- Output: $\hat{P_x^B} = \frac{P_x^B}{P_z^B}$ (10 bit)

(D) High P_t Part

LUT 20 - 27: $\xi_{Mag,Pt}$

- **to be done**

- Input: ξ_{Mag} (10 bit) and ξ^{mid}_{Tc2} (10 bit)

17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0
ξ_{Mag}								ξ^{mid}_{Tc2}									

- LUT to be used depends on bit 8 and 9 of ξ_{Mag}:

bit 9	bit 8	LUT A	LUT B
0	0	20	24
0	1	21	25
1	0	22	26
1	1	23	27

- Output: 9 bit number: **to be done**

LUT A (20 - 23):

7	6	5	4	3	2	1	0
7	6	5	4	3	2	1	0
$\xi_{Mag,PT}$							

LUT B (24 - 27):

7	6	5	4	3	2	1	0
x	x	x	x	x	x	x	8
							$\xi_{Mag,PT}$

LUT 37: AcceptB

- **to be done**

- Input $dd\xi$ (7 bit) and E_b^- (bit 6..2) and Q (1 bit) and ALL (1 bit) and ID (2 bit)

17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
x	7	6	5	4	3	2	1	0	6	5	4	3	2	0	1	1	0
	$dd\xi$								E_b^-					Q	ALL	ID	

- Output: 2bit number $Accept_B$ using ????????

LUT37:

7	6	5	4	3	2	1	0
1	0	x	x	x	x	1	0
$Accept_b$						$dd\xi$	

LUT 38: AcceptA

- **top be done**

- Input: ξ_m^{PT} (9 bit) and E_b^- (bit 6.5) and E_b^+ (7 bit)

17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
x	7	6	5	4	3	2	1	0	6	5	6	5	4	3	2	1	0
	ξ_m^{PT}								E_b^-		E_b^+						

- Output: 2 bit number depending on bit 8 of ξ_m^{PT} output bits 0 and 1 or 2 and 3 are used:

bit 8	used output bits
0	1 and 0
1	3 and 2

The output bits used are selcted outside the LUT.

LUT 38:

7	6	5	4	3	2	1	0
x	x	x	x	3	2	1	0
				$Accept_a$			

LUT 39: Cut3

- **to be done**
- Input: $Accept_1$ (2 bit) and $Accept_2$ (2 bit)

17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
x	x	x	x	x	x	x	x	x	x	x	x	x	x	1	0	1	0
														$Accept_b$		$Accept_a$	

- Output: 1 bit number: Accept = 1 / Not accept = 0 ?

(E) Double Track Rejection

A special double track comparator is implemented in the TPU to recognize and suppress double tracks. Double tracks can be produced by the Track Finding Units (TFU) if a track is considered as two seperate tracks in one tracking layer. A track is considered as a double track if its parameter $\hat{P}_x$, $\hat{P}_y$, BX and ID are identical.
The comparator compares each incoming track in pipeline stage 9 with the tracks in the following pipeline steps. Since the TPU processor pipeline ends with the pipeline stage 12, a buffer register has been implemented containing the track parameters of the last 18 accepted tracks. For each pipeline step, the comparator consists of four individual comparators: $\hat{P}_x$-, $\hat{P}_y$-, BX- and ID-Comparator. Each comparator is used to compare the input from the pipeline stage P9 with the stored TPU Pipeline or -buffer register. Figure 4 shows schematically the comparison of $\hat{P}_x$.
For $\hat{P}_x$ and $\hat{P}_y$, $\hat{P}_x^B$ and $\hat{P}_y^B$ from Lookuptables 47,48 and 49 are taken for the comparison (high Et part) if ALL and $FLAG$ are 1, otherwise $\hat{P}_x$ and $\hat{P}_y$ from Lookuptable 42 to 45 are taken. The comparator for $\hat{P}_x$ and $\hat{P}_y$ not only checks for the exact identity, but also tolerates a ± 1 deviation to compensate for the ± 1 tolerance of $\hat{P}_x$, $\hat{P}_y$. For BX and ID an exact identity is required.
If the present track differs from all register entries a separate flag for each variable ($\hat{P}_x$, $\hat{P}_y$, BX and ID) is send to the Write Enable buffer.

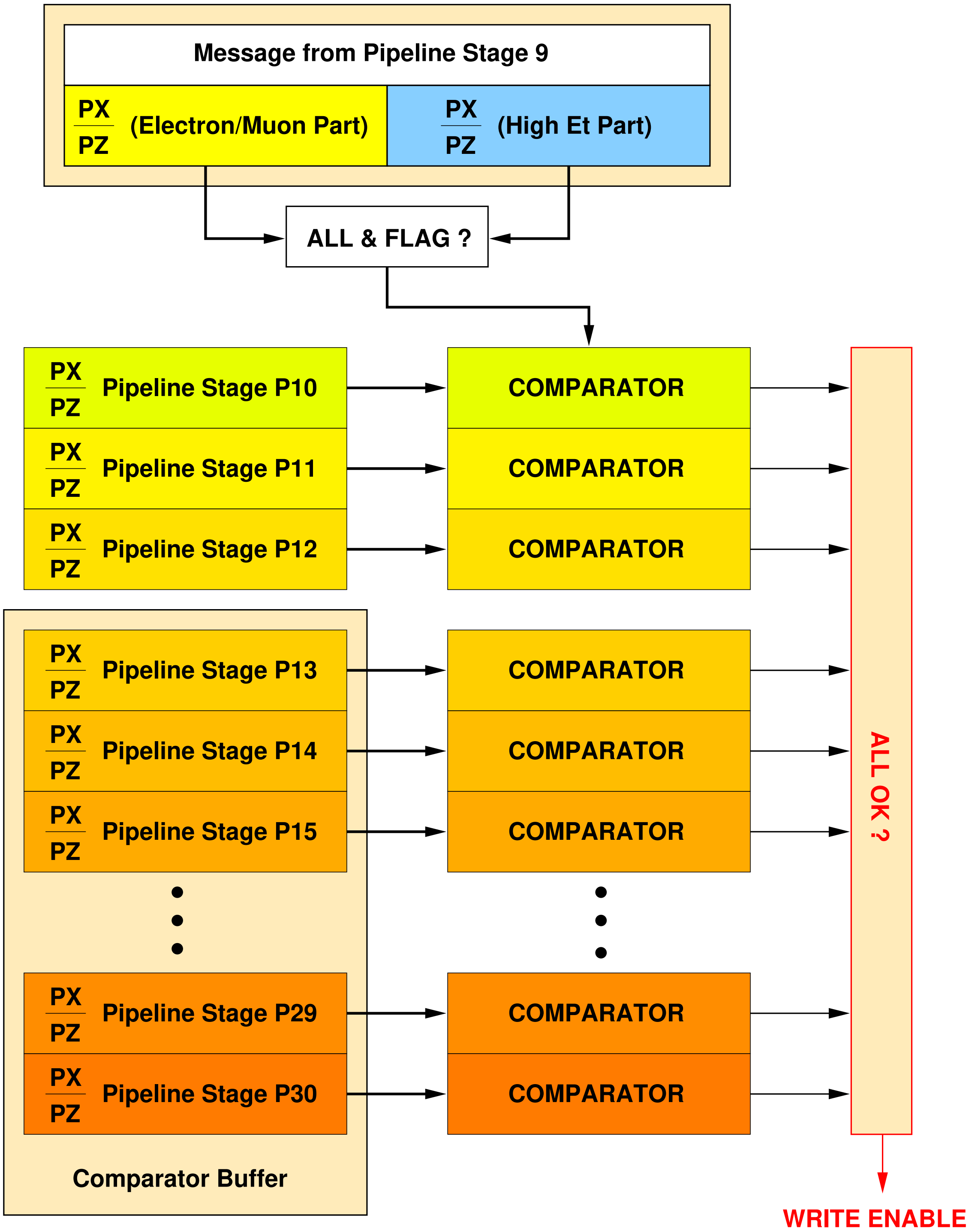


Figure 4: Schematic view of the double track rejection for $\hat{P}_x$ of the TPU.

(F) Histogram Cache Operation

The Track Paramter Unit includes a histogram cache to suppress tracks, if the number of tracks belonging to a certain BX number exceeds a programmable limit. Suppression of these tracks increases the overall performance of the First Level Trigger. The cache exists of eight cache entry pairs.
The TPU cache histogram compares the BX number of accepted tracks with the cache entries. Bits 2,1 and 0 of the BX number are taken to select the cache entry pair to be used for the comparison and bits 7 to 3 are used for the comparison. If the BX number has not been found in one of the two cache entries, the BX entry of the Cache Counter which has not been used the last, is replaced by the new BX number (bit 7 to 3), the valid field is set to true and the histogram counter is set to one.
If the BX number has been found, the number of the Cache Counter is increased by one and the value of the Cache Counter is compared with the cut register[16]. If the value of the cut register is exceeded, the track suppression signal is generated and send to the Write Enable buffer.
Because of the pipelined operation of the TPU processor and the histogram cache, there is a delay between the entry write and the generation of the track suppression signal. Therefore the cut limit depends on the order of the tracks. Figure 5 shows a schematic view of the Histogram Cache.

To ensure that not two different BX numbers, belonging to BX_a and BX_a+256 are counted up in the same cache, an additional Cache entry clear operation is included. It ensures that each buffer containing a BX number BX_a is cleared just before $BX_a + 256$ messages are processed by the TPU.
To apply this clearing, the Cache Counter looks at the most significant BX bits 7 and 6 of the actual number and compares them to an internal Buffer. When the TPU is started, an internal two bit Clear Buffer is set to 00. If the TPU processes a message with the most significant bits 7 and 6 equal to 00, each cache entry containing a BX number having its two most significant bit equal to 01 is cleared and the internal Clear Buffer is increased by one (to 01). If it receives messages, having the two most significant bits not equal to 00, no operation is done.
If the Clear Buffer is increased by one, the TPU now compares and clears the Buffer in the same way with the new Clear buffer value 01. Figure 6 shows an example of the clearing process.

To ensure correct operations, even at very low rate, an additional clear operation can be enabled. This additional clear operation uses 4 Cache counters, having a size of 4 bits and which are driven by a cycle time of 320ns.

- Counter 0 for entries having a $BX\mathbf{0}XXXX\mathbf{0}XXX$[17]
- Counter 1 for entries having a $BX\mathbf{0}XXXX\mathbf{1}XXX$
- Counter 2 for entries having a $BX\mathbf{1}XXXX\mathbf{0}XXX$
- Counter 3 for entries having a $BX\mathbf{1}XXXX\mathbf{1}XXX$

[16] The maximal number of tracks to be taken for one BX.
[17] In the following description, X always stands for any bit meaning 0 or 1.

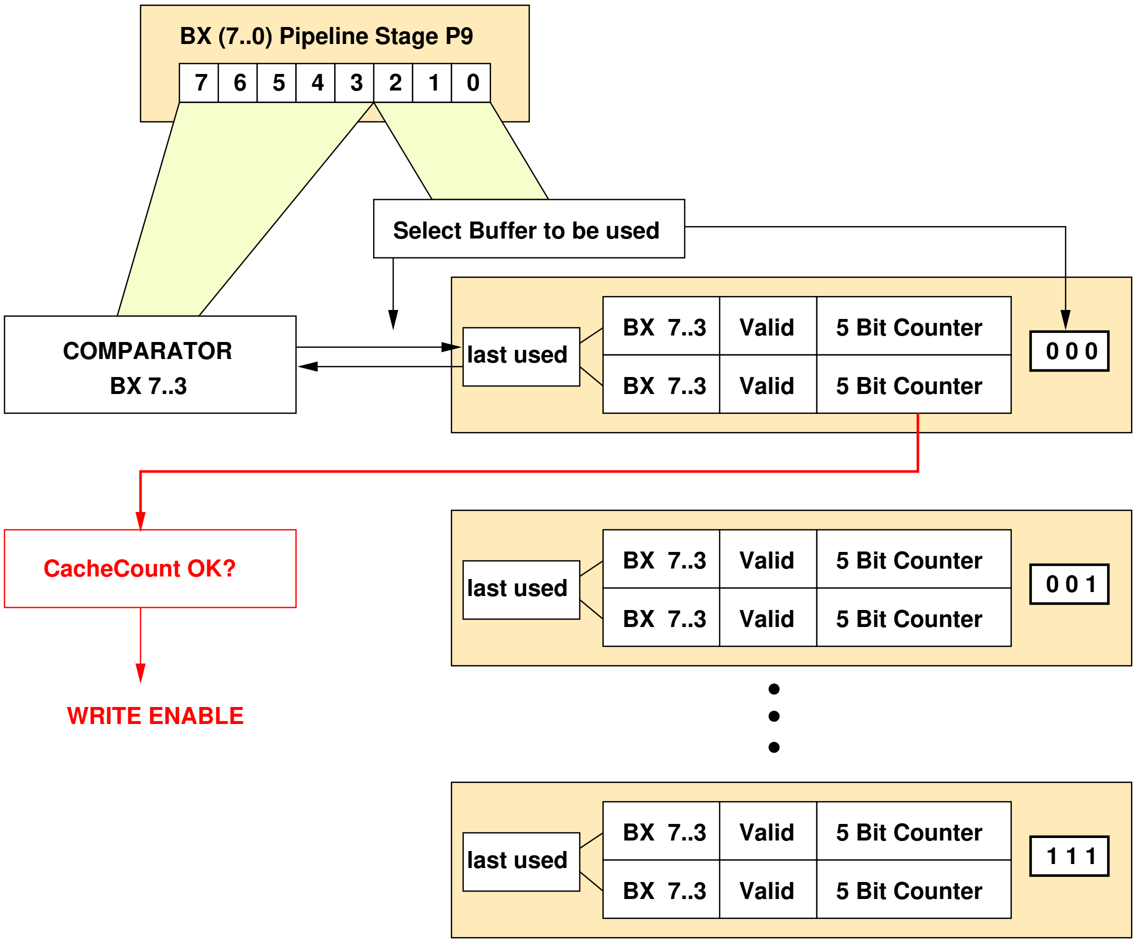


Figure 5: Schematic view of the Histogram Cache of the TPU.

This means, each buffer is increased by one after 320 ns, giving a total amount of $16 * 320ns = 5120ns$ cycling time. If a track, having a $BX\mathbf{0}XXXX\mathbf{0}XXX$ is processed by the TPU, the Counter 0 is reseted, if it processes a BX number $\mathbf{1}XXXX\mathbf{0}XXX$, Counter 3 is reseted and so on. After resetting a counter, the counter starts to count again starting from 0.
If a counter reaches ist maximal value (1111), all Cache entries having the corresponding BX number a cleared, eg. if Counter 3 reaches 1111, all Cache entries with a BX **1** X X X X **1** X X X are cleared.
This leads to the clearing of a buffer with eg. $BX\mathbf{0}XXXX\mathbf{0}XXX$ the latest 5,1 μs after the last $BX\mathbf{0}XXXX\mathbf{0}XXX$ has been processed by the TPU. Figure 7 shows an example of the operation.

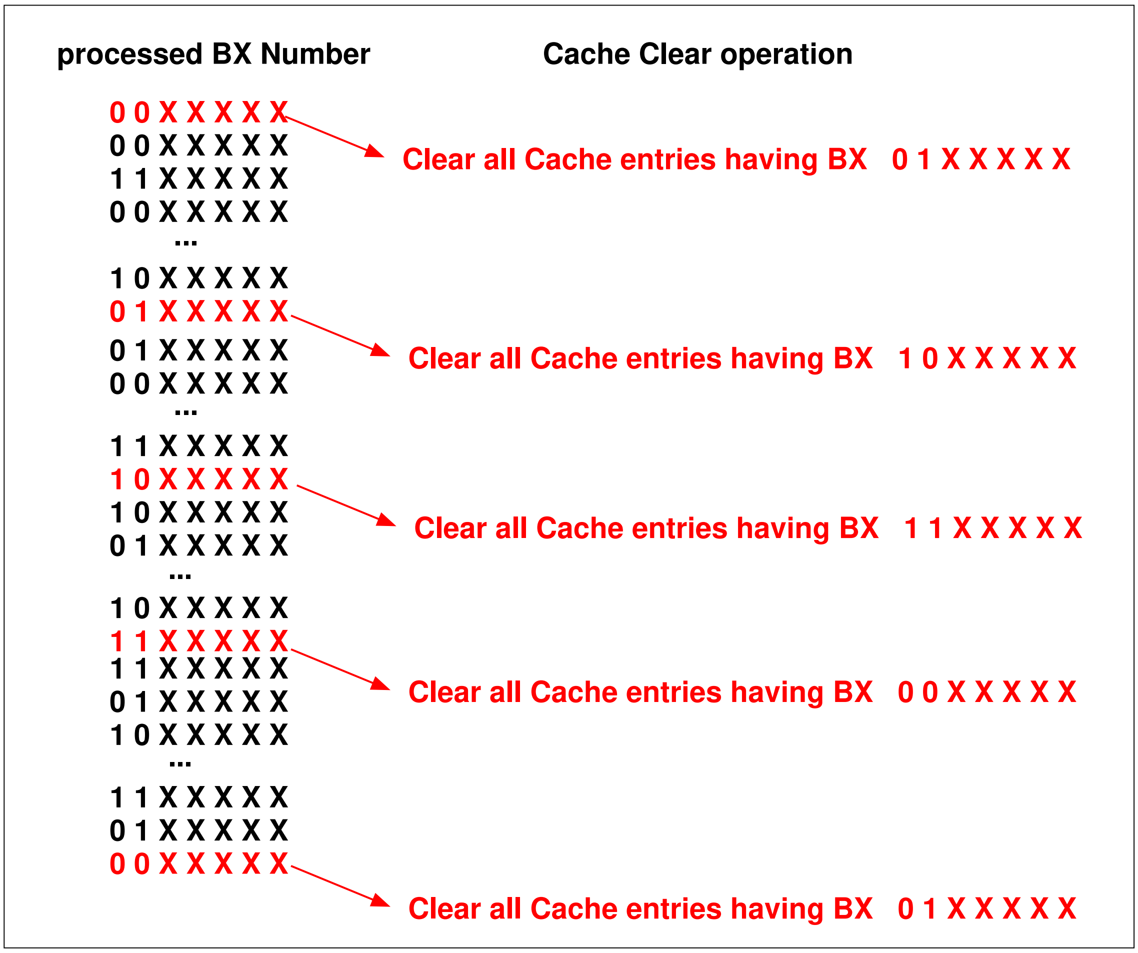


Figure 6: Example of the clear operation of the Histogram Cache.

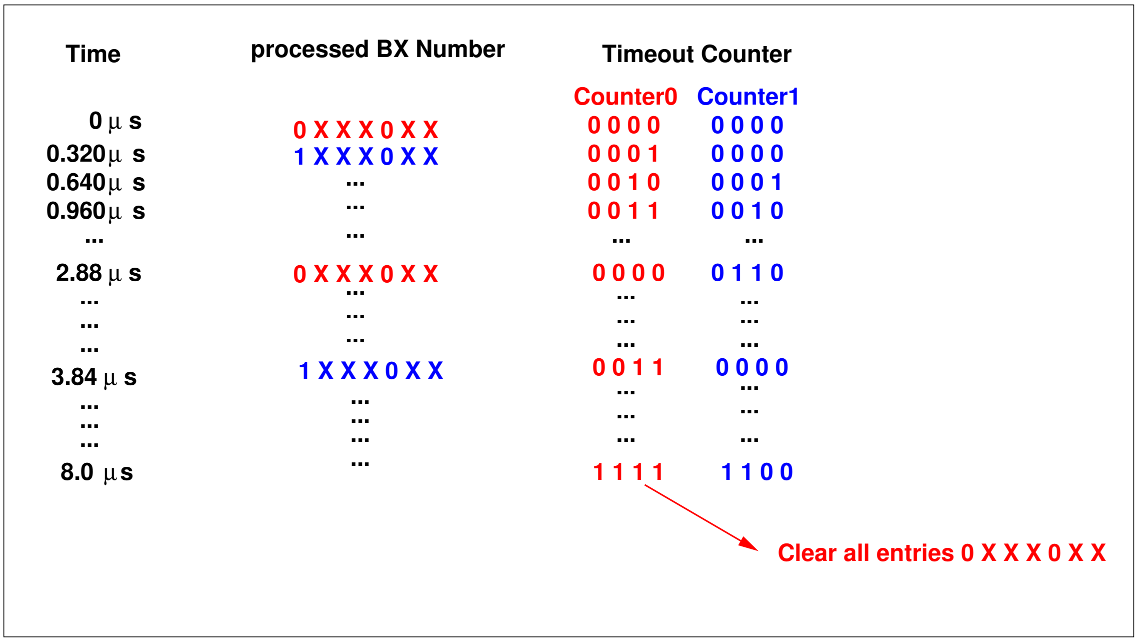


Figure 7: Example of the timeout clear operation of the Histogram Cache.