Chris Perkins
01/29/2019

Implementation of QT Algorithm for STAR ZDC: Combined Proton-Proton and Heavy-Ion Algorithms

QT Code Version: 0x6f
MCS File: qt32b_l0_v6_f.mcs
qt32c_l0_v6_f.mcs

Description:

This algorithm combines the STAR ZDC Proton-Proton (pp) and Heavy-Ion (HI) QT algorithms into one algorithm. The pp algorithm follows from QT code v62 while the Heavy-Ion algorithm follows from QT code v6A. No changes were made to either algorithm except for internal bit timing to align output bits. Note that the "Good Hit" requirement (or non-requirement) is retained from the original algorithms and is different between the pp and Heavy-Ion algorithms.

In addition to combining the two algorithms, the east and west sides can be configured independently to use either algorithm. The Algorithm Select register must be the same for both East QT Daughter Cards (A & B). This register must also be the same for both West QT Daughter Cards (C & D).

Note that as a result of the addition of the TAC Gain Control register the register map in the QTC system is different from the original QTB version.

Proton-Proton Algorithm:

This algorithm compares various ADC sums to thresholds, passes two separate partial TAC values (East and West), and passes two separate partial ADC sums (East and West). To choose this algorithm set the Algorithm Select register to '0' for East or West.

Only channels that satisfy a "Good Hit" requirement are included in all parts of this algorithm (ADC sums for threshold comparison, TAC output, ADC sum output). A "Good Hit" is defined as one where the ADC value is greater than some threshold and the corresponding TAC value is greater than TAC_MIN and less than TAC_MAX. The channel mask register can be used but note that ADC and TAC channels must each be masked individually.

Note that only the first two ADC and TAC channels are used on each daughter card. The other channels will show up in the datastream but are not considered in the trigger decision.

The first sum considered is channel 0 + 1 on each daughter card. This is compared to Pair_Threshold and one bit per daughter card is output. The second sum considered is channel 0 + 1 on daughter A plus channel 0 + 1 on daughter B. A similar sum is calculated from channels 0 + 1 on daughter C plus channels 0 + 1 on daughter D. These sums are compared to Sum_Threshold and two bits in total are output from each QT32 (East and West).

There are two separate partial TAC values output: the upper 10 bits (2-11) from the first TAC channel on daughter A and the first TAC channel on daughter C, both subject to the “Good Hit” requirement on the full TAC value.

This algorithm also outputs the upper three bits (11-13) from the (channel 0+1)$_{AB}$ sum and the (channel 0+1)$_{CD}$ sum.

Note that this algorithm uses the direct path from Daughter B to the L0 FPGA. In order to achieve the correct bit alignment at the L0 FPGA the data from Daughter B must be delayed appropriately. This delay setting is packed into the upper bits of the Algorithm Latch register.

The masks should be set to 0x02 on daughters A and C, and 0x00 on daughters B and D. This will mask out the second ADC channel on daughters A and C only and are compatible with the Heavy-Ion algorithm so that channel masks don’t need to change for different collision species. This effectively makes the Pair thresholds as follows:

Pair A: ZDC E1	(East Front)
Pair B: ZDC E2 + E3	(East Back)
Pair C: ZDC W1	(West Front)
Pair D: ZDC W2 + W3	(West Back)

And the Sum thresholds as follows:

Sum A+B: ZDC E1 + E2 + E3	(East Sum)
Sum C+D: ZDC W1 + W2 + W3	(West Sum)

Heavy-Ion Algorithm:

This algorithm compares the East and West Analog Sums to four different thresholds, compares the East and West Attenuated Analog Sums to two thresholds, and outputs the upper bits of the E1TAC and W1TAC signals if they are within some range. To choose this algorithm set the Algorithm Select register to ‘1’ for East or West.

This algorithm does **not** use the “Good Hit” requirements.

The upper 10 bits of the E1TAC signal (Daughter A, ch4) is passed on to L0 if:

TAC_MIN < E1TAC < TAC_MAX

Otherwise, 0x000 is passed on to L0 for E1TAC. An equivalent condition is required to pass on the upper 10 bits of W1TAC (Daughter C, ch4). Note that the “Good Hit” ADC threshold register is not used in this algorithm.

The ESum, WSum, ESumA, and WSumA threshold bits have no requirements on their TAC signals; a threshold bit is ‘1’ if the corresponding channel is greater than the corresponding threshold with no other requirements.

Inputs:

QT8A : E1 (ch0), Masked out (ch1), ESum (ch2), ESumA (ch3), E1TAC (ch4)
QT8B : E2 (ch0), E3 (ch1)
QT8C : W1 (ch0), Masked out (ch1), WSum (ch2), WSumA(ch3), W1TAC (ch4)
QT8D : W2 (ch0), W3 (ch2)

Registers (1 Set Per Daughter Card)**:**

QT Reg. 11: Channel Mask
QT8A: 0x02
QT8B: 0x00
QT8C: 0x02
QT8D: 0x00

Alg. Reg. 0 (Reg 13):	ZDC_QT_ADC_Th	(pp Only)
Alg. Reg. 1 (Reg 14):	ZDC_QT_TAC_Min	(pp & HI)
Alg. Reg. 2 (Reg 15):	ZDC_QT_TAC_Max	(pp & HI)
Alg. Reg. 3 (Reg 16):	ZDC_QT_E/W_HI_Th0	(only Daughters A and C)
Alg. Reg. 4 (Reg 17):	ZDC_QT_E/W_HI_Th1	(only Daughters A and C)
Alg. Reg. 5 (Reg 18):	ZDC_QT_E/W_HI_Th2	(only Daughters A and C)
Alg. Reg. 6 (Reg 19):	ZDC_QT_E/W_HI_Th3	(only Daughters A and C)
Alg. Reg. 7 (Reg 20):	ZDC_QT_E/Watten_HI_Th0	(only Daughters A and C)
Alg. Reg. 8 (Reg 21):	ZDC_QT_E/Watten_HI_Th1	(only Daughters A and C)
QTB		
Alg. Reg. 9 (Reg 22):	ZDC_QT_E/W_pp_Front/Back_Th	(all Daughters)
Alg. Reg. 10 (Reg 23):	ZDC_QT_E/W_pp_Total_Th	(only Daughters B and D)
Alg. Reg. 11 (Reg 24):	ZDC_QT_E/W_Alg_Select	(all Daughters, 0=pp, 1=HI)
QTC		
Alg. Reg. 9 (Reg 22):	N/A – used for TAC Gain Control	
Alg. Reg. 10 (Reg 23):	ZDC_QT_E/W_pp_Front/Back_Th	(all Daughters)
Alg. Reg. 11 (Reg 24):	ZDC_QT_E/W_pp_Total_Th	(only Daughters B and D)
Alg. Reg. 12 (Reg 25):	ZDC_QT_E/W_Alg_Select	(all Daughters, 0=pp, 1=HI)

LUT:

TAC timing adjustment/ADC Pedestal subtraction for each channel

Algorithm Latch:

QT8A: 0x0
QT8B: 0x100
QT8C: 0x0
QT8D: 0x1

L0 Output to DSM:

(0-9)	:	West TAC (Daughter C, ch4) (Upper 10 bits)
(10-19)	:	East TAC (Daughter A, ch4) (Upper 10 bits)
(20-25)	:	West Sum/Threshold Bits (see below)
(26-31)	:	East Sum/Threshold Bits (see below)

Sum/Threshold Bits : pp Algorithm :

West :

(20-22)	:	West ADC Sum (bits 11-13) (Daughters C+D)
(23)	:	West Pair Good (C)
(24)	:	West Pair Good (D)
(25)	:	West Sum Good (C+D)

East :

(26-28)	:	East ADC Sum (bits 11-13) (Daughters A+B)
(29)	:	East Pair Good (A)
(30)	:	East Pair Good (B)
(31)	:	East Sum Good (A+B)

Sum/Threshold Bits : Heavy-Ion Algorithm :

West :

(20)	:	WSum > Th0
(21)	:	WSum > Th1
(22)	:	WSum > Th2
(23)	:	WSum > Th3
(24)	:	WSumA > atten_Th0
(25)	:	WSumA > atten_Th1

East :

(26)	:	ESum > Th0
(27)	:	ESum > Th1
(28)	:	ESum > Th2
(29)	:	ESum > Th3
(30)	:	ESumA > atten_Th0
(31)	:	ESumA > atten_Th1

Actions:

Tick	QT8A	QT8B	QT8C	QT8D
1	**Latch Inputs**	**Latch Inputs**	**Latch Inputs**	**Latch Inputs**
2	"Good Hit" ADC & TAC Th **Delay ADC & TAC (del1)**	"Good Hit" ADC & TAC Th **Delay ADC & TAC (del1)**	"Good Hit" ADC & TAC Th **Delay ADC & TAC (del1)**	"Good Hit" ADC & TAC Th **Delay ADC & TAC (del1)**
3	**Latch "Good Hit" ADC & TAC** **Latch "Good TAC" TAC** ADC_X > Th → **ADC_Th_X$_A$**	**Latch "Good Hit" ADC & TAC**	**Latch "Good Hit" ADC & TAC** **Latch "Good TAC" TAC** ADC_X > Th → **ADC_Th_X$_C$**	**Latch "Good Hit" ADC & TAC**
4	Good Ch0 + Good Ch1 → **(0+1)$_A$** **Local_TAC$_A$_del1** **ADC_Th_X$_A$_del1**	Good Ch8 + Good Ch9 → **(0+1)$_B$**	Good Ch0 + Good Ch1 → **(0+1)$_C$** **Local_TAC$_C$_del1** **ADC_Th_X$_C$_del1**	Good Ch8 + Good Ch9 → **(0+1)$_D$**
5	(0+1)$_A$ > Th → **Pair_Th$_A$** **(0+1)$_A$_del1** **Local_TAC$_A$_del2** **ADC_Th_X$_A$_del2**	(8+9)$_B$ > Th → **Pair_Th$_B$** **(8+9)$_B$_del1**	(0+1)$_C$ > Th → **Pair_Th$_C$** **(0+1)$_C$_del1** **Local_TAC$_C$_del2** **ADC_Th_X$_C$_del2**	(8+9)$_D$ > Th → **Pair_Th$_D$** **(8+9)$_D$_del1**
6	**Latch Out :** **Local_TAC$_A$_del2** **pp:** **(0+1)$_A$_del1** **Pair_Th$_A$** **HI:** **ADC_Th_X$_A$_del2**	**Pair_Th$_B$_del1** **(8+9)$_B$_del2**	**Latch Out :** **Local_TAC$_C$_del2** **pp:** **(0+1)$_C$_del1** **Pair_Th$_C$** **HI:** **ADC_Th_X$_C$_del2**	**Pair_Th$_D$_del1** **(8+9)$_D$_del2**
7	-	**Pair_Th$_B$_del2** **(8+9)$_B$_del3** **Latch In:** **TAC$_A$** **(0+1)$_A$** **Pair_Th$_A$** **ADC_Th_X$_A$**	-	**Pair_Th$_D$_del2** **(8+9)$_D$_del3** **Latch In:** **TAC$_C$** **(0+1)$_C$** **Pair_Th$_C$** **ADC_Th_X$_C$**
8	-	(0+1+8+9)$_{AB}$ → **Sum$_{AB}$** **TAC$_A$_del1** **Pair_Th$_A$_del1** **Pair_Th$_B$_del3** **ADC_Th_X$_A$_del1**	**-**	(0+1+8+9)$_{CD}$ → **Sum$_{CD}$** **TAC$_C$_del1** **Pair_Th$_C$_del1** **Pair_Th$_D$_del3** **ADC_Th_X$_C$_del1**
9	-	(0+1+8+9)$_{AB}$ > Th → **Sum_Th$_{AB}$** **Sum$_{AB}$_del1** **Pair_Th$_A$_del2** **Pair_Th$_B$_del4** **ADC_Th_X$_A$_del2** **TAC$_A$_del2**	-	(0+1+8+9)$_{CD}$ > Th → **Sum_Th$_{CD}$** **Sum$_{CD}$_del1** **Pair_Th$_C$_del2** **Pair_Th$_D$_del4** **ADC_Th_X$_C$_del2** **TAC$_C$_del2**

Tick	QT8A	QT8B	QT8C	QT8D
10	**-**	**TAC$_A$_del3 (bits 0-7)** **Latch Out :** **TAC$_A$_del2 (bits 8-9)** **pp :** **Sum$_{AB}$_del1** **SumTh$_{AB}$** **Pair_Th$_A$_del2** **Pair_Th$_B$_del4** **HI :** **ADC_Th_X$_A$_del2**	-	**Sum$_{CD}$_del2** **Sum_Th$_{CD}$_del1** **Pair_Th$_C$_del3** **Pair_Th$_D$_del5** **ADC_Th_X$_C$_del3** **TAC$_C$_del3**
11	**-**	**TAC$_A$_del4 (bits 0-7)**	**Latch In :** **East Bits (8 bits)**	**Sum$_{CD}$_del3** **Sum_Th$_{CD}$_del2** **Pair_Th$_C$_del4** **Pair_Th$_D$_del6** **ADC_Th_X$_C$_del4** **TAC$_C$_del4**
12	**-**	**TAC$_A$_del5 (bits 0-7)**	**Latch Out :** **East Bits (8 bits)**	**Sum$_{CD}$_del4** **Sum_Th$_{CD}$_del3** **Pair_Th$_C$_del5** **Pair_Th$_D$_del7** **ADC_Th_X$_C$_del5** **TAC$_C$_del5**
13	**-**	**TAC$_A$_del6 (bits 0-7)**	**-**	**Latch In :** **East Bits (8 bits)** **Sum$_{CD}$_del5** **Sum_Th$_{CD}$_del4** **Pair_Th$_C$_del6** **Pair_Th$_D$_del8** **ADC_Th_X$_C$_del6** **TAC$_C$_del6**
14	-	**Latch Out : TAC$_A$_del6** **to L0 FPGA**	**-**	**Latch Out :** **East Bits (8 bits)** **TAC$_C$** **pp :** **Sum$_{CD}$** **Sum_Th$_{CD}$** **Pair_Th$_C$** **Pair_Th$_D$** **HI :** **ADC_Th_X$_C$**