Chris Perkins
06/14/2016

Implementation of QT Algorithm for STAR MTD : Run 2013

QT Code Version: 0x6c
MCS File: qt32b_l0_v6_c.mcs

Description:

This algorithm outputs the two largest TAC Pair Sums, each corrected by the TAC Pair Differences, from eight MTD modules and includes an ID for each TAC Pair Sum.

Each MTD module has an East and West end (J2 and J3), which are connected to either the first two or second two channels (2x ADC and 2x TAC) on any QT8 daughter card. This algorithm assumes the standard input configuration where channels 1-4 are signal inputs (ADC) and channels 5-8 are TAC inputs corresponding to channels 1-4.

An outline of the steps followed by this algorithm are listed below, with details of each step described later in this section:

1. Slew Correct each TAC Channel
2. Apply channel masks
3. Modified "Good Hit" Requirement
4. Form TAC Pair Sums (J2 + J3) and TAC Pair Differences (J2 – J3)
5. Correct TAC Pair Sums
6. Truncate Corrected TAC Pair Sums
7. Find two largest truncated, corrected TAC Pair Sums and assign IDs

A slew correction is applied to each TAC channel based on the value of the corresponding ADC channel. In the current implementation, there are a maximum of eight ADC bins. The ADC bin limits for each TAC channel can be defined independently. The ADC bin limits must cover the full available range of ADC values [0:4095] and must not overlap. Therefore any ADC value falls into exactly one ADC bin. The determination of which bin an ADC value falls into is done using the following logic:

Bin(X) = bin_limit(X-1) < ADC <= bin_limit(X)

Note that the lower limit of Bin(0) is hardwired to be 0, but the user has the ability to set all the other limits.

A slew correction offset is associated with each ADC bin of each channel. The slew correction offset is a signed integer with a range [-256 : 255]. The slew correction offset for this corresponding bin is then added to the raw TAC value. If the result is negative, a slew corrected TAC value of '0' is used. If the result is greater than 4095, a slew corrected TAC value of '4095' is used. This ensures that the slew corrected TAC values have the same range as the raw TAC values (ie [0:4095]).

The standard QT mask registers can be used for each channel to mask that channel from the trigger but retain the data in the datastream. Note that separate masks must be used for ADC and TAC channels. Also note that the channel masks are applied AFTER the slew correction.

This algorithm uses a modified "Good Hit" definition, which requires that the TAC value for a channel is greater than some **TAC_Min** and less than some **TAC_Max**. Note that this modified "Good Hit" definition doesn't include any requirement on the ADC value. For a module to be considered for the two largest TAC Pair Sums, both J2 and J3 must satisfy the "Good Hit" requirement for that module. If either the J2 or J3 input of a module doesn't satisfy the "Good Hit" requirement, the corrected TAC Pair Sum for that module will be '0'. If no modules satisfy the "Good Hit" requirement, both of the maximum TAC Pair Sums will be '0'.

After the TAC Difference (J2 – J3) is calculated, its value is checked to ensure that it is within some range. This range is specified in one register that designates the absolute value of the range. If the following is not true :

-1 * **TAC_Diff_Range_Abs** < TAC Difference < **TAC_Diff_Range_Abs**

then the corrected TAC Sum is set to zero. The value of **TAC_Diff_Range_Abs** is a 10 bit unsigned integer [0:1023].

The raw TAC Pair Sum (J2 + J3) is corrected according to the following:

Corrected_TAC_Sum = Raw_TAC_Sum + (Raw_TAC_Diff * **Correction_Factor** / 32) + **Correction_Offset_PairX** As previously stated, the corrected TAC Pair Sum is set to zero if either J2 or J3 doesn't satisfy the "Good Hit" requirement or the TAC Pair Difference falls outside the required range. The value of **Correction_Factor** is a 4 bit unsigned integer [0:15]. Each **Correction_Offset_PairX** is a 10 bit unsigned integer [0:1023].

The corrected TAC Pair Sum is then truncated to 10 bits. Bits [3:12] (starting from zero) are kept.

The two largest truncated, corrected TAC Pair Sums are then found and output. Each channel pair (MTD module) is assigned an ID as follows:

QT8A, ADC Ch0/1, TAC Ch4/5 = 1
QT8A, ADC Ch2/3, TAC Ch6/7 = 2
QT8B, ADC Ch0/1, TAC Ch4/5 = 1
QT8B, ADC Ch2/3, TAC Ch6/7 = 2
QT8C, ADC Ch0/1, TAC Ch4/5 = 1 or 2
QT8C, ADC Ch2/3, TAC Ch6/7 = 3 or 4
QT8D, ADC Ch0/1, TAC Ch4/5 = 3
QT8D, ADC Ch2/3, TAC Ch6/7 = 4

The first pair (J2, J3) of Daughter Card 'C' can either be assigned ID '1' or '2' depending on the sign of the TAC Difference. Similarly, the second pair of Daughter Card 'C' can either be assigned ID '3' or '4' depending on the sign of the TAC Difference. If (J2-J3) >= 0, the ID for Daughter Card 'C' is '1' or '3' (East). If (J2-J3) < 0, the ID for Daughter Card 'C' is '2' or '4' (West).

In the case of equal Corrected TAC Pair Sums between multiple channel pairs, the resulting ID value can be found using the pseudo-code listed later on. The two pairs within one QT8 daughter card are first ordered and IDs assigned (see pseudo-code

below). The maximums are then found between the locally ordered sums and the incoming sums from previous daughters (also see pseudo-code below).

Inputs:

QT8A :

Ch 0/1 : MTD Position 1 or 2 Module J2/J3 ADC
Ch 2/3 : MTD Position 5 or 4 Module J2/J3 ADC
Ch 4/5 : MTD Position 1 or 2 Module J2/J3 TAC
Ch 6/7 : MTD Position 5 or 4 Module J2/J3 TAC

QT8B :

Ch 0/1 : MTD Position 1 or 2 Module J2/J3 ADC
Ch 2/3 : MTD Position 5 or 4 Module J2/J3 ADC
Ch 4/5 : MTD Position 1 or 2 Module J2/J3 TAC
Ch 6/7 : MTD Position 5 or 4 Module J2/J3 TAC

QT8C :

Ch 0/1 : MTD Position 3 Module J2/J3 ADC
Ch 2/3 : MTD Position 3 Module J2/J3 ADC
Ch 4/5 : MTD Position 3 Module J2/J3 TAC
Ch 6/7 : MTD Position 3 Module J2/J3 TAC

QT8D :

Ch 0/1 : MTD Position 1 or 2 Module J2/J3 ADC
Ch 2/3 : MTD Position 5 or 4 Module J2/J3 ADC
Ch 4/5 : MTD Position 1 or 2 Module J2/J3 TAC
Ch 6/7 : MTD Position 5 or 4 Module J2/J3 TAC

Registers (1 Set Per Daughter Card, GUI Register Name in **bold**)**:**

Alg. Reg. 1 (Reg 14): "Good Hit" **TAC_Min** (12 bits, unsigned)
Alg. Reg. 2 (Reg 15): "Good Hit" **TAC_Max** (12 bits, unsigned)
Alg. Reg. 3 (Reg 16): **Correction_Factor** (4 bits, unsigned)
Alg. Reg. 4 (Reg 17): **Correction_Offset_Pair0** (10 bits, unsigned)
Alg. Reg. 5 (Reg 18): **Correction_Offset_Pair1** (10 bits, unsigned)
Alg. Reg. 6 (Reg 19): **TAC_Diff_Range_Abs** (10 bits, unsigned)

LUT:

TAC timing adjustment/ADC Pedestal subtraction for each channel

Slew Correction: 8 ADC Bins/Slew Correction Offsets per TAC channel

Algorithm Latch: 4

L0 Output to DSM:

(0-9) : Maximum Corrected TAC Pair Sum (J2 + J3)
(10-11) : Maximum Corrected TAC Pair ID
(12-15) : '0'
(16-25) : Second Maximum Corrected TAC Pair Sum (J2 + J3)
(26-27) : Second Maximum Corrected TAC Pair ID
(28-31) : '0'

Pseudo-code for ordering local QT8 Pair Sums and assigning IDs,

Daughter Cards 'A', 'B', 'D' :

```
If (LOCAL_SUM0 > LOCAL_SUM1) then
    LOCAL_MAX_SUM0 = LOCAL_SUM0
    LOCAL_MAX_ID0 = '1' for 'A' and 'B',  '3' for 'D'
    LOCAL_MAX_SUM1 = LOCAL_SUM1
    LOCAL_MAX_ID1 = '2' for 'A' and 'B',  '4' for 'D'
Else
    LOCAL_MAX_SUM0 = LOCAL_SUM1
    LOCAL_MAX_ID0 = '2' for 'A' and 'B',  '4' for 'D'
    LOCAL_MAX_SUM1 = LOCAL_SUM0
    LOCAL_MAX_ID1 = '1' for 'A' and 'B',  '3' for 'D'
End if;
```

Daughter Cards 'C' :

```
If (LOCAL_SUM0 > LOCAL_SUM1) then
    LOCAL_MAX_SUM0 = LOCAL_SUM0
    If (TAC_DIFF0 >= 0) then
        LOCAL_MAX_ID0 = '1'
    Else
        LOCAL_MAX_ID0 = '2'
    End if;
    LOCAL_MAX_SUM1 = LOCAL_SUM1
    If (TAC_DIFF1 >= 0) then
        LOCAL_MAX_ID1 = '3'
    Else
        LOCAL_MAX_ID1 = '4'
    End if;
Else
    LOCAL_MAX_SUM0 = LOCAL_SUM1
    If (TAC_DIFF1 >= 0) then
        LOCAL_MAX_ID0 = '3'
    Else
        LOCAL_MAX_ID0 = '4'
    Endif;
    LOCAL_MAX_SUM1 = LOCAL_SUM0
    If (TAC_DIFF0 >= 0) then
        LOCAL_MAX_ID1 = '1'
    Else
        LOCAL_MAX_ID1 = '2'
    End if;
End if;
```

Pseudo-code for ordering between daughter cards:

First Highest Corrected TAC Pair Sum :

```
If (LOCAL_MAX_SUM0 > INPUT_MAX_SUM0) then
    OUTPUT_SUM0 = LOCAL_MAX_SUM0
    OUTPUT_ID0 = LOCAL_MAX_ID0
Else
    OUTPUT_SUM0 = INPUT_MAX_SUM0
    OUTPUT_ID0 = INPUT_MAX_ID0
End if;
```

Second Highest Corrected TAC Pair Sum :

```
If (LOCAL_MAX_SUM0 > INPUT_MAX_SUM0) then
    If (LOCAL_MAX_SUM1 > INPUT_MAX_SUM0) then
        OUTPUT_SUM1 = LOCAL_MAX_SUM1
        OUTPUT_ID1 = LOCAL_MAX_ID1
    Else
        OUTPUT_SUM1 = INPUT_MAX_SUM0
        OUTPUT_ID1 = INPUT_MAX_ID0
    End if;
Else
    If (LOCAL_MAX_SUM0 > INPUT_MAX_SUM1) then
        OUTPUT_SUM1 = LOCAL_MAX_SUM0
        OUTPUT_ID1 = LOCAL_MAX_ID0
    Else
        OUTPUT_SUM1 = INPUT_MAX_SUM1
        OUTPUT_ID1 = INPUT_MAX_ID1
    End if;
End if;
```

Actions:

Tick	QT8A	QT8B	QT8C	QT8D
1	**Latch Inputs**	**Latch Inputs**	**Latch Inputs**	**Latch Inputs**
2	**Find/Latch ADC Bins** **Delay TAC Values**	**Find/Latch ADC Bins** **Delay TAC Values**	**Find/Latch ADC Bins** **Delay TAC Values**	**Find/Latch ADC Bins** **Delay TAC Values**
3	**Calculate/Latch** **Slew Corrected TACs**	**Calculate/Latch** **Slew Corrected TACs**	**Calculate/Latch** **Slew Corrected TACs**	**Calculate/Latch** **Slew Corrected TACs**
4	**Overflow/Underflow/Mask** **Slew Corrected TACs**	**Overflow/Underflow/Mask** **Slew Corrected TACs**	**Overflow/Underflow/Mask** **Slew Corrected TACs**	**Overflow/Underflow/Mask** **Slew Corrected TACs**
5	TAC > R1 →**TAC_MIN_GOOD** TAC < R2 →**TAC_MAX_GOOD** Ch4 + Ch5 → **Sum1** Ch6 + Ch7 → **Sum2** Ch4 – Ch5 → **Diff1** Ch6 – Ch7 → **Diff2** Ch5 > Ch4 → **Diff1_Neg** Ch7 > Ch6 → **Diff2_Neg**	TAC > R1 →**TAC_MIN_GOOD** TAC < R2 →**TAC_MAX_GOOD** Ch4 + Ch5 → **Sum1** Ch6 + Ch7 → **Sum2** Ch4 – Ch5 → **Diff1** Ch6 – Ch7 → **Diff2** Ch5 > Ch4 → **Diff1_Neg** Ch7 > Ch6 → **Diff2_Neg**	TAC > R1 →**TAC_MIN_GOOD** TAC < R2 →**TAC_MAX_GOOD** Ch4 + Ch5 → **Sum1** Ch6 + Ch7 → **Sum2** Ch4 – Ch5 → **Diff1** Ch6 – Ch7 → **Diff2** Ch5 > Ch4 → **Diff1_Neg** Ch7 > Ch6 → **Diff2_Neg**	TAC > R1 →**TAC_MIN_GOOD** TAC < R2 →**TAC_MAX_GOOD** Ch4 + Ch5 → **Sum1** Ch6 + Ch7 → **Sum2** Ch4 – Ch5 → **Diff1** Ch6 – Ch7 → **Diff2** Ch5 > Ch4 → **Diff1_Neg** Ch7 > Ch6 → **Diff2_Neg**
6	Diff1 * Factor → **Mult1** Diff2 * Factor → **Mult2** Delay : **Sum1, Sum2, Diff1, Diff2** **Diff1_Neg, Diff2_Neg,** **Good_Hits**	Diff1 * Factor → **Mult1** Diff2 * Factor → **Mult2** Delay : **Sum1, Sum2, Diff1, Diff2** **Diff1_Neg, Diff2_Neg,** **Good_Hits**	Diff1 * Factor → **Mult1** Diff2 * Factor → **Mult2** Delay : **Sum1, Sum2, Diff1, Diff2** **Diff1_Neg, Diff2_Neg,** **Good_Hits**	Diff1 * Factor → **Mult1** Diff2 * Factor → **Mult2** Delay : **Sum1, Sum2, Diff1, Diff2** **Diff1_Neg, Diff2_Neg,** **Good_Hits**
7	**Calculate/Latch** **Corrected TAC Sums →** **Sum1_Corr, Sum2_Corr** Delay : **Diff1_Neg, Diff2_Neg**	**Calculate/Latch** **Corrected TAC Sums →** **Sum1_Corr, Sum2_Corr** Delay : **Diff1_Neg, Diff2_Neg**	**Calculate/Latch** **Corrected TAC Sums →** **Sum1_Corr, Sum2_Corr** Delay : **Diff1_Neg, Diff2_Neg**	**Calculate/Latch** **Corrected TAC Sums →** **Sum1_Corr, Sum2_Corr** Delay : **Diff1_Neg, Diff2_Neg**
8	**Truncate Corrected TAC Sums** Delay : **Diff1_Neg, Diff2_Neg**	**Truncate Corrected TAC Sums** Delay : **Diff1_Neg, Diff2_Neg**	**Truncate Corrected TAC Sums** Delay : **Diff1_Neg, Diff2_Neg**	**Truncate Corrected TAC Sums** Delay : **Diff1_Neg, Diff2_Neg**
9	Order Local Corrected TAC Sums and assign IDs	Corrected Sum→**Sum_Del1** Diff_Neg→ **Diff_Neg_Del1**	Corrected Sum→**Sum_Del1** Diff_Neg→ **Diff_Neg_Del1**	Corrected Sum→**Sum_Del1** Diff_Neg→ **Diff_Neg_Del1**
10	Find Global MAX TAC Sums	Corrected Sum→**Sum_Del2** Diff_Neg→ **Diff_Neg_Del2**	Corrected Sum→**Sum_Del2** Diff_Neg→ **Diff_Neg_Del2**	Corrected Sum→**Sum_Del2** Diff_Neg→ **Diff_Neg_Del2**
11	**Latch Out MAX Sums/IDs**	Corrected Sum→**Sum_Del3** Diff_Neg→ **Diff_Neg_Del3**	Corrected Sum→**Sum_Del3** Diff_Neg→ **Diff_Neg_Del3**	Corrected Sum→**Sum_Del3** Diff_Neg→ **Diff_Neg_Del3**
12	-	**Latch In Sums/IDs** Order Local Delayed (Del3) Corrected TAC Sums and assign IDs	Corrected Sum→**Sum_Del4** Diff_Neg→ **Diff_Neg_Del4**	Corrected Sum→**Sum_Del4** Diff_Neg→ **Diff_Neg_Del4**
13	-	Find Global MAX TAC Sums	Corrected Sum→**Sum_Del5** Diff_Neg→ **Diff_Neg_Del5**	Corrected Sum→**Sum_Del5** Diff_Neg→ **Diff_Neg_Del5**
14	-	**Latch Out MAX Sums/IDs**	Corrected Sum→**Sum_Del6** Diff_Neg→ **Diff_Neg_Del6**	Corrected Sum→**Sum_Del6** Diff_Neg→ **Diff_Neg_Del6**

Tick	QT8A	QT8B	QT8C	QT8D
15	**-**	**-**	**Latch In Sums/IDs** Order Local Delayed (Del6) Corrected TAC Sums and assign IDs	Corrected Sum→**Sum_Del7** Diff_Neg→ **Diff_Neg_Del7**
16	**-**	**-**	Find Global MAX TAC Sums	Corrected Sum→**Sum_Del8** Diff_Neg→ **Diff_Neg_Del8**
17	**-**	**-**	**Latch Out MAX Sums/IDs**	Corrected Sum→**Sum_Del9** Diff_Neg→ **Diff_Neg_Del9**
18	**-**	**-**	**-**	**Latch In Sums/IDs** Order Local Delayed (Del9) Corrected TAC Sums and assign IDs
19	-	-	-	Find Global MAX TAC Sums
20	-	-	-	**Latch Out MAX Sums/IDs**