Implementation and Algorithms for Vertex-QT-DSM-Tree

Eleanor Judd
Chris Perkins

February 22nd 2009
Update March 2nd 2009

There are some big changes to the Vertex DSM Tree for the 2009 running period. All the original digitizer boards (CDB) and the layer 0 DSM boards have been replaced by QT boards. Also, since the Vertex Position Detector (VPD) is designed to detect the same vertex as the BBC and ZDC, it has been added in to this part of the DSM tree.

1. Layer 0 QT Boards: BBQ_BB001:002

File: qt32b_l0_v5_3.mcs

Description:
This algorithm forms a 16bit ADC Sum and 12bit TAC Max. Only channels that satisfy a "good hit" requirement are included in the ADC Sum and TAC Max. 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 then TAC_MAX. The channel mask register can be used but note that ADC and TAC channels must each be masked individually.

Inputs:
QT8A: 4 PMT ADC, 4 PMT TAC
QT8B: 4 PMT ADC, 4 PMT TAC
QT8C: 4 PMT ADC, 4 PMT TAC
QT8D: 4 PMT ADC, 4 PMT TAC

Registers (1 Set Per Daughter Card)**:**
Alg. Reg. 0 (Reg 13): ADC_Threshold
Alg. Reg. 1 (Reg 14): TAC_MIN
Alg. Reg. 2 (Reg 15): TAC_MAX
Reg. 11: Channel Mask

LUT:
Timing adjustments/pedestal subtraction for each PMT

Action (21x RHIC Clock)**:**
1st: Mask channels and Latch inputs
If mask bit = 1, channel data = 0

2nd: For each PMT (4 per daughter board):
ADC above threshold: ADC > PMT_ADC_Thresh → Good_ADC
TAC above threshold: TAC > TAC_MIN → Good_TAC_MIN
TAC below threshold: TAC < TAC_MAX → Good_TAC_MAX

3rd: Make good_hits(0-3):
good_hit(i) = Good-ADC(i) && Good_TAC_MIN(i) && Good_TAC_MAX(i)

4th: Sum channels 0+1 subject to good hit requirements → Int_sum_0
Sum channels 2+3 subject to good hit requirements → Int_sum_1
Compare TAC channels 4, 5 subject to good hit requirements → Int_max_0
Compare TAC channels 6, 7 subject to good hit requirements → Int_max_1

5th: Sum Int_sum_0 + Int_sum_1 → Int_sum_2
Compare Int_max_0, Int_max_1 → Int_max_2

6th: Sum Int_sum_2 + Sum from previous daughters → ADC_Sum
Compare Int_max_2 to TAC Max from previous daughters → TAC_Max

7th: Latch Output Bits to next daughter or L0 FPGA
(0-15) : ADC_Sum
(16) : '0'
(17-28) : TAC_Max
(29-33) : '0'

Algorithm Latch: 1 or 2

L0 Output to DSM:
(0-15) : ADC Sum
(16-27) : TAC Max
(28-31) : '0'

2. Layer 1 DSM Board: BBC_BB101

The BB101 DSM board processes data from the BBC-small-tile detector. The algorithm receives ADC-sum and fastest-TAC data from the new QT boards. The ADC sums are compared to thresholds. A set of bits specified by the user is chosen from each incoming TAC value to send to the scaler system. In parallel, the TAC difference is calculated. The difference is set to zero if either of the two incoming TACS is zero, because a TAC value of zero implies there were no good hits on that side of the BBC.

RBT File: bbc_bb101_2009_a.rbt

Users: BB101, VP101

Inputs: Ch0/1 = QT Board BB001 (East)
Ch2/3 = QT Board BB002 (West)

From each QT board:
bits 0:15 = ADC-Sum
bits 16:27 = Max TAC (Value of zero implies NO good hits)

LUT: 1:1

Registers:

Four registers, all thresholds can be set independently
R0: BBCsmall-EastADCsum_th (16 bits)
R1: BBCsmall-WestADCsum_th (16)
R2: BBCsmall-EastTAC-select (3)
0 => select bits 0:6
1 => select bits 1:7
…
5 => select bits 5:11
R3: BBCsmall-WestTAC-select (3)
Same value definitions as for R2

Action:

1st Latch input

2nd Compare each ADC-sum to its threshold
Calculate: TAC difference = 4096 + TAC-E – TAC-W
Define: Good-TAC-E = TAC-E > 0, same for West side
Make all possible bit selections from TAC-E and TAC-W, including overflow logic. For example:
TAC-E-overflow-0 = TAC-E(7), (8), (9), (10) or (11)
If (TAC-E-overflow-0 = 1) then TAC-E-scaler-0 = 127
Else TAC-E-scaler-0 = TAC-E(0:6)
Same logic for all possible bit selections from TAC-E (see description of register R2) and TAC-W

3rd Delay ADC-sum threshold bits
Zero out TAC difference if either Good-TAC-E or Good-TAC-W is false, otherwise just delay TAC difference
Use R2 to select the TAC-E scaler bits:
If (R2 = 0) then chose TAC-E-scaler-0
Else if (R2 = 1) then chose TAC-E-scaler 1
Etc…
Do the same for West side, using R3 to control the selection.

4th Latch output

Output to VT201:

(0-12) TAC difference
(13) Unused
(14) ADC-sum-E > th0
(15) ADC-sum-W > th0

Scalers:

(0-6) selected bits of TAC-E
(7-13) selected bits of TAC-W
(14) ADC-sum-E > th0
(15) ADC-sum-W > th0

3. Layer 0 QT Boards: BBQ_BB003

The layer 0 DSM board for the large-tile BBC been replaced with a QT board this year. This algorithm is not implemented yet.

4. Layer 1 DSM Board: BBC_BB102

The BB102 DSM board receives large-tile BBC data from 1 QT board. This algorithm is not implemented yet.

5. Layer 0 QT Boards: BBQ_VP001:004

The VPD layer 0 DSM boards have been replaced with QT boards this year. See documentation above for BBQ_BB001:002.

6. Layer 1 DSM Board: BBC_VP101

The VP101 DSM board receives VPD data from 4 QT boards. However, currently only 2 of those boards (VP001 and VP002) produce data that needs to be analyzed by the trigger system. The logic needed to do this analysis is the same as that used by the BB101 algorithm, so VP101 will use the BB101 algorithm. It should be noted that both output cables from VP101 go to VT201, so the scaler system will not receive the data on the second output cable from VP101.

7. Layer 0 QT Board: BBQ_ZD001

File: qt32b_l0_v5_3.mcs

Description:

This algorithm compares various ADC sums to thresholds and finds two separate TAC Max values.

Only channels that satisfy a "good hit" requirement are included in sums for threshold comparisons and TAC Max determination. 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 then 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 total are output from each QT32.

There are two separate TAC Max values output: one from TAC channels on daughters A and B, another from TAC channels on daughters C and D.

Note that this algorithm uses the direct path from Daughter B to the L0 FPGA.

The default masks for Run 9 (as of 090302) are 0xEE on daughters A and C, and 0xCC on daughters B and D. This makes the Pair thresholds as follows:

Pair A: ZDC E1
Pair B: ZDC E2+E3
Pair C: ZDC W1
Pair D: ZDC W2+W3

And the Sum thresholds as follows:

Sum A+B: ZDC E1+E2+E3
Sum C+D: ZDC W1+W2+W3

And the TAC Max values as follows:

TAC Max A,B: Max(ZDC E1,E2,E3)
TAC Max C,D: Max(ZDC W1,W2,W3)

Inputs:

QT8A: 2 PMT ADC (ch 0,1), 2 PMT TAC (ch 4,5)
QT8B: 2 PMT ADC (ch 8,9), 2 PMT TAC (ch 12,13)
QT8C: 2 PMT ADC (ch 16,17), 2 PMT TAC (ch 20,21)
QT8D: 2 PMT ADC (ch 24,25), 2 PMT TAC (ch 28,29)

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

Alg. Reg. 0 (Reg 13): ADC_Threshold
Alg. Reg. 1 (Reg 14): TAC_MIN
Alg. Reg. 2 (Reg 15): TAC_MAX
Alg. Reg. 3 (Reg 16): Pair_Threshold
Alg. Reg. 4 (Reg 17): Sum_Threshold (only valid on daughters B,D)
Reg. 11: Channel Mask

LUT:

Timing adjustments/pedestal subtraction for each PMT

Algorithm Latch: 1

Action (21x RHIC Clock)**:**

1st: Mask channels and Latch inputs
If mask bit = 1, channel data = 0

2nd: For each PMT (2 per daughter board):
ADC above threshold: ADC > PMT_ADC_Thresh → Good_ADC
TAC above threshold: TAC > TAC_MIN → Good_TAC_MIN
TAC below threshold: TAC < TAC_MAX → Good_TAC_MAX

3rd: Make good_hits(0-1):
good_hit(i) = Good-ADC(i) && Good_TAC_MIN(i) && Good_TAC_MAX(i)

4th: Sum ADC channels 0+1 subject to good hit requirements → Int_sum_0
Compare TAC channels 4, 5 subject to good hit requirements → Int_max_0

5th: Compare Int_sum_0 to Pair_Threshold → Pair_Good
Add Int_sum_0 to sum from previous daughter (input bits0-12) → Int_sum_1
(Note: This result is ignored on daughters A,C)
Compare Int_max_0 to max from previous daughter (input bits17-28) → Int_max_1
(Note: This result is ignored on daughters A,C)

6th: Compare Int_sum_1 to Sum_Threshold → Sum_Good
(Note: This result is ignored on daughters A,C)

7th: Latch Output Bits to next daughter or L0 FPGA

if(daughter A)

(0-12) : Int_sum_0 (Pair ADC Sum A)
(13-16) : '0'
(17-28) : Int_max_0 (Pair TAC Max A)
(29-32) : '0'
(33) : Pair_Good (A)

else if(daughter B)

(0-12) : '0'
(13-16) : Int_max_1 (bits8-11) (TAC Max A,B)
(17-30) : '0'
(31) : Sum_Good (A+B)
(32) : Pair_Good (B)
(33) : Pair_Good (A) (Passed from previous daughter)
Level0_Out : Int_max_1 (bits0-7) (TAC Max A,B)

else if(daughter C)

(0-12) : Int_sum_0 (Pair ADC Sum C)
(13-16) : TAC Max(bits8-11) (A,B) (Passed from previous daughter)
(17-28) : Int_max_0 (Pair TAC Max C)
(29) : '0'
(30) : Pair Good (C)
(31) : Sum_Good (A+B) (Passed from previous daughter)
(32) : Pair_Good (B) (Passed from previous daughter)
(33) : Pair_Good (A) (Passed from previous daughter)

else if(daughter D)

(0-11) : Int_max_1 (TAC Max C,D)
(12) : '0'
(13-16) : TAC Max(bits8-11) (A,B) (Passed from previous daughter)
(17-27) : '0'
(28) : Sum_Good (C+D)
(29) : Pair Good (D)
(30) : Pair Good (C) (Passed from previous daughter)
(31) : Sum_Good (A+B) (Passed from previous daughter)
(32) : Pair_Good (B) (Passed from previous daughter)
(33) : Pair_Good (A) (Passed from previous daughter)

L0 Output to DSM:

(0-11) : TAC Max (C,D)
(12-23) : TAC Max (A,B)
(24) : Sum Good (C+D)
(25) : Pair Good (D)
(26) : Pair Good (C)
(27) : Sum Good (A+B)
(28) : Pair Good (B)
(29) : Pair Good (A)
(30-31) : '0'

8. Layer 1 DSM Board: BBC_ZD101

The ZD101 DSM board processes data from the ZDC detector. The algorithm receives fastest-TAC data from the new QT boards. It also receives the results of comparing ADC sums to thresholds. Those threshold bits are passed through to VT201 unmodified. A set of bits specified by the user is chosen from each incoming TAC value to send to the scaler system. This is a variation on the logic used in the BB101 algorithm. The difference is that a smaller set of bits is available to send to the scaler system, so the user has a larger number of subsets to choose from. In parallel, the TAC difference is calculated. The difference is set to zero if either of the two incoming TACS is zero, because a TAC value of zero implies there were no good hits on that side of the ZDC. A user-specified set of bits is then chosen to be passed on to VT201.

RBT File: bbc_zd101_2009_b.rbt

Users: ZD101

Inputs: Ch0/1 = QT Board ZD001
Ch2:7 = Unused

From the QT board:
bits 0:11 = Max TAC, West
bits 12:23 = Max TAC, East
bit 24 = West ADC sum > threshold
bit 25 = Back West ADC > threshold
bit 26 = Front West ADC > threshold
bit 27 = East ADC sum > threshold
bit 28 = Back East ADC > threshold
bit 29 = Front East ADC > threshold
bits 30:31 = Unused

LUT: 1:1

Registers:
Three selection registers
R0: ZDC-TACdiff-select (2 bits)
0 => select bits 0:9
1 => select bits 1:10
2 => select bits 2:11
3 => select bits 3:12
R1: ZDC-EastTAC-select (3)
0 => select bits 0:4
1 => select bits 1:5
…
7 => select bits 7:11
R2: ZDC-WestTAC-select (3)
Same value definitions as for R1

Action:

1st Latch input

2nd Delay ADC-sum threshold bits to the 4th step.
Calculate: TAC difference = 4096 + TAC-E – TAC-W
Define: Good-TAC-E = TAC-E > 0, same for West side
Make all possible bit selections from TAC-E and TAC-W, including overflow logic. For example:
TAC-E-overflow-0 = TAC-E(5), (6), (7), (8), (9), (10) or (11)
If (TAC-E-overflow-0 = 1) then TAC-E-scaler-0 = 31
Else TAC-E-scaler-0 = TAC-E(0:4)
Same logic for all possible bit selections from TAC-E (see description of register R1) and TAC-W

3rd Use R0 to select the TAC difference bits for VT201, including overflow logic and the "good" TAC cut, i.e.:
Diff-overflow-0 = TAC-diff(10), (11) or (12)
Diff-overflow-1 = TAC-diff(11) or (12)
Diff-overflow-2 = TAC-diff(12)
If (Good-TAC-E = 0 or Good-TAC-W = 0) then output = 0
Else if (R0 = 0)
If (Diff-overflow-0 = 1) then output = 1023
Else output = TAC-diff(0:9)
Else if (R0 = 1)
If (Diff-overflow-1 = 1) then output = 1023
Else output = TAC-diff(1:10)
Etc…
Use R1 to select the TAC-E scaler bits:
If (R1 = 0) then chose TAC-E-scaler-0
Else if (R1 = 1) then chose TAC-E-scaler 1
Etc…
Do the same for the West side using R2 to control the selection.

4th Latch output

Output to VT201:

(0-9) TAC difference
(10) ADC-sum-E > th0
(11) ADC-sum-W > th0
(12) Front-ADC-E > th0
(13) Back-ADC-E > th0
(14) Front-ADC-W > th0
(15) Back-ADC-W > th0

Scalers:

(0-4) selected bits of TAC-E
(5-9) selected bits of TAC-W
(10) ADC-sum-E > th0
(11) ADC-sum-W > th0
(12) Front-ADC-E > th0
(13) Back-ADC-E > th0
(14) Front-ADC-W > th0
(15) Back-ADC-W > th0

9. Layer 2 Vertex DSM Board: L1-VT201

All threshold bits of the Vertex tree from the large and small-tile BBC, the ZDC and the VPD are brought into the Vertex DSM. They are passed on to the last DSM and the TCU. In parallel all four TAC differences are brought into the Vertex DSM. Windows are placed around each TAC difference, and the "inside window" bits get passed through to the last DSM and the scaler system. The four MSB of the TAC difference from the BBC small-tiles, the ZDC and the VPD are also in the scaler output.

RBT File: l1_vt201_2009_a.rbt

Users: VT201

Inputs: Ch 0 = BB101
Ch 1 = BB102
Ch 2 = ZD101
Ch 3 = Unused
Ch 4 = VP101 (JP1)
Ch 5 = VP101 (JP6)
Ch 6/7 = Unused

From Small tile BBC-DSM BB101
(0-12) Small tile TAC-Difference
(13) Unused
(14/15) Small tile ADC East/West sum > th0

From Large tile BBC-DSM BB102
(0-2) Large tile TAC-Difference
(13) Unused
(14/15) Large tile ADC East/West sum > th0

From ZDC DSM ZD101
(0-9) ZDC TAC-Difference
(10) ZDC East ADC sum > th0
(11) ZDC West ADC sum > th0
(12) ZDC East Front ADC > th0
(13) ZDC East Back ADC > th0
(14) ZDC West Front ADC > th0
(15) ZDC West Back ADC > th0

From VPD-DSM VP101
(0-12) VPD TAC-Difference
(13) Unused
(14/15) VPD ADC East/West> th0

LUT: Either 1-to-1 or TAC-difference range conversion

Registers:

R0: BBCsmall-TACdiff-Min (13 bits)
R1: BBCsmall-TACdiff-Max (13)
R2: BBClarge-TACdiff-Min (13)
R3: BBClarge-TACdiff-Max (13)
R4: ZDC-TACdiff-Min (10)
R5: ZDC-TACdiff-Max (10)
R6: VPD-TACdiff-Min (13)
R7: VPD-TACdiff-Max (13)

Action

1st Latch inputs

2nd Delay all 12 input threshold bits to the 4th step.
Delay a copy of the BBC-small, VPD and ZDC TAC difference to the 4th step.
Compare each of the 4 TAC differences to its minimum and maximum value, as specified in the relevant registers. The logic looks for the TAC difference to be greater than the minimum and less than the maximum;

3rd Combine the results of the TAC difference comparisons to determine if each TAC difference is inside its specified window, e.g.:
ZDC-TAC-diff-in-window = R4 < ZDC TAC difference < R5

4th Latch Outputs

Output to LD301 or new TCU:

(0) BBC small-tile TAC difference in window
(1) BBC small-tile East ADC sum > th
(2) BBC small-tile West ADC sum > th
(3) BBC large-tile TAC difference in window
(4) BBC large-tile East ADC sum > th
(5) BBC large-tile West ADC sum > th
(6) ZDC TAC difference in window
(7) ZDC East ADC sum > th
(8) ZDC West ADC sum > th
(9) ZDC East Front ADC > th
(10) ZDC East Back ADC > th
(11) ZDC West Front ADC > th
(12) ZDC West Back ADC > th
(13) VPD TAC difference in window
(14) VPD East ADC sum > th
(15) VPD West ADC sum > th

Scalers:

(0) BBC small-tile TAC difference in window
(1:4) 4 MSB of BBC small-tile TAC difference
(5) BBC large-tile TAC difference in window
(6) ZDC TAC difference in window
(7:10) 4 MSB of ZDC TAC difference
(11) VPD TAC difference in widnow
(12-15) 4 MSB of VPD TAC difference