Algorithms for Vertex QT-DSM Tree
RHIC 2016 dAu Run

Eleanor Judd

June 17, 2016

Change Log:

Date	Description
January 18, 2016	First change for 2016 heavy-ion data taking. The input bit map of the ZD101 algorithm has been updated to match the output bit map of the hybrid ZDC QT algorithm in its heavy-ion mode. There are no changes to the registers, the algorithm itself or the output bits.
March 3, 2016	The VPD logic has been changed to effectively calculate the mean good TAC value on each side instead of selecting the fastest good TAC value. In order to avoid the necessity of performing a generic, large integer division the QT boards pass the sum and hit count to the DSM tree, and the DSM algorithms multiply both the sums and the window cuts by the product of the hit counts. All of this logic is in VP101 so VT201 now just passes through ready-made ADC and TAC-in-window bits.
May 10, 2016	The BBC logic has been changed in BB101 to place a window cut on the TAC sum. This is in addition to the existing window cut on the TAC difference that is applied to VT201. A 3rd VPD TAC window cut, which was previously made and then discarded, is now passed through to the TCU. In order to make room for these 2 new TCU bits the Preceded logic has been dropped from VT201, and the 2 BBC-Large ADC bits have been combined.
June 17, 2016	The VP101 logic has been changed to add a VPD ADC coincidence bit in the output going to the scalers. There are no changes to the actual trigger bits going to VT201 and the TCU.

The Vertex branch of the DSM tree is used to locate the primary vertex of the RHIC beam collisions at STAR. All three relevant trigger detectors connect to this branch: Zero Degree Calorimeters (ZDC), Beam-Beam Counters (BBC) and the Vertex Position Detector (VPD). The raw detector signals are digitized and pre-processed in QT boards. The DSM tree is then used to calculate TAC differences and combine ADC information to produce (for example) minimum bias or ultra-peripheral triggers.

Layer 0 QT Boards: BBQ_BB001:002

There are two BBC small-tile QT boards: one processes data from the East side of the detector and the other from the West side. The algorithm has been changed in 2014 to add a slewing correction to the original logic. Please see the documentation provided by Chris Perkins for a detailed description of the new algorithm at http://www.star.bnl.gov/public/trg/TSL/Software/qt_v6_d_doc.pdf
The algorithm still forms a 16-bit ADC Sum and 12-bit 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 than TAC_MAX. The channel mask register can be used to force the algorithm to ignore certain channels, but note that ADC and TAC channels must each be masked individually.

1. Layer 1 DSM Boards: 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 QT boards. The ADC sums are compared to thresholds. In parallel, both the TAC difference and the TAC sum are 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. The sum is compared to two thresholds, which define a window. The result is true if the TAC sum is inside the window. It is false if the sum falls outside the window or if either of the two incoming TACS is zero.

RBT File: bbc_bb101_2016_a.rbt

Users: BB101

Inputs: Ch0/1 = QT Board BB001 (East)
Ch2/3 = QT Board BB002 (West)
Ch4/7 = Unused

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-TACsum-Min (13)
R3: BBCsmall-TACsum-Max (13)

Action:
1st Latch input

2nd Compare each ADC-sum to its threshold

Calculate: TAC difference = 4096 + TAC-E – TAC-W
Calculate: TAC sum = TAC-E + TAC-W
Define: Good-TAC-E = TAC-E > 0
Define: Good-TAC-W = TAC-W > 0

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.
Compare the TAC sum to its minimum and maximum values, as specified in registers #2 and #3. Combine the results to determine if the TAC sum is inside its specified window, i.e.:

(R2 < TAC Sum < R3) AND Good-TAC-E AND Good-TAC-W

4th Latch output

Output to VT201:

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

Scalers:

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

2. Layer 0 QT Boards: BBQ_BB003

There is just one BBC large-tile QT board and it receives data from both the East and West sides of the detector. The algorithm was written by Chris Perkins and is documented at http://www.star.bnl.gov/public/trg/TSL/Software/qt_v5_f_doc.pdf

3. Layer 1 DSM Board: BBC_BB102

The BB102 DSM board processes data from the BBC-large-tile detector. The algorithm receives a hit flag and fastest-TAC data for each of the East and West sides of the detector from the QT board. The hit flags indicate there was at least one good hit on each side, and they are just passed through to the output. 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_bb102_2010_b.rbt

Users: BB102

Inputs: Ch0/1 = QT Board BB003 (East and West)
Ch2/7 = Unused

From the QT board:
bits 0:11 = MAX TAC East (value of zero implies no good hits)
bits 12:23 = MAX TAC West
bit 24 = East hit
bit 25 = West hit

LUT: 1:1

Registers:

R0: BBClarge-EastTAC-select (3)

0 => select bits 0:6
1 => select bits 1:7
…
5 => select bits 5:11

R1: BBClarge-WestTAC-select (3)

Same value definitions as for R0

Action:

1st Latch input

2nd Delay hit bits to 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(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 R0) and TAC-W (see register R1)

3rd Zero out TAC difference if either Good-TAC-E or Good-TAC-W is false, otherwise just delay TAC difference to the 4th step
Use R0 to select the TAC-E scaler bits:

If (R0 = 0) then chose TAC-E-scaler-0
Else if (R0 = 1) then chose TAC-E-scaler-1
Etc…

Do the same for West side, using R1 to control the selection.

4th Latch output

Output to VT201:

(0-12) TAC difference
(13) Unused
(14) East hit
(15) West hit

Scalers:

(0-6) selected bits of TAC-E
(7-13) selected bits of TAC-W
(14) East hit

(15) West hit

4. Layer 0 QT Boards: BBQ_VP001:002

There are two VPD QT boards: one processes data from the East side of the detector and the other from the West side. The algorithm now forms a truncated 12-bit ADC Sum, a full-range 16-bit Sum of slew- and noise-corrected TAC values and a "good hit" count. Only channels that satisfy the "good hit" requirements are included in the Sums. 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. Please see the documentation for a detailed description of the new algorithm at http://www.star.bnl.gov/public/trg/TSL/Software/qt_v7_6_doc.pdf

5. Layer 1 DSM Board: BBC_VP101

The VP101 DSM board processes data from the VPD detector. The algorithm receives ADC- and TAC sum data as well as hit counts from the QT boards. The ADC sums are compared to thresholds. The TAC sum and hit counts are combined to effectively cut on the difference of the means while avoiding the necessity of implementing a generic large integer division. The standard calculation is:

$$\left| \frac{\sum E}{N(E)} - \frac{\sum W}{N(W)} \right| < threshold$$

This can be re-written as:

$$\left| \left(N(W) \sum E \right) - \left(N(E) \sum W \right) \right| < threshold * N(E)N(W)$$

Performing multiplication in the FPGA is easier than division, so the 2nd equation is used. NOTE: This algorithm takes 4 extra ticks of the 4xRHIC clock, which is used by the FPGA, to complete all these calculations.

RBT File: bbc_vp101_2016_b.rbt

Users: VP101

Inputs: Ch0/3 = Unused
Ch4/5 = QT Board VP003 (East)
Ch6/7 = QT Board VP004 (West)

From each QT board:
bits 0:11 = ADC Sum
bits 12:15 = Hit Count
bits 16:31 = TAC Sum

LUT: 1:1

Registers:
Five registers, all thresholds can be set independently

R0: VPD-EastADCsum_th (12 bits)
R1: VPD-WestADCsum_th (12)
R2: VPD-MeanDiff-window1 (12)
R3: VPD-MeanDiff-window2 (12)
R4: VPD-MeanDiff-window3 (12)

Action:

1st Latch input

2nd Compare each ADC sum to its threshold
Calculate: $N(W)\sum E$ $N(E)\sum W$ and $N(E)N(W)$

3rd Delay the ADC sum threshold bits to the 8th step
Calculate: $Diff = [N(W)\sum E - N(E)\sum W]$
Calculate: $Th1 = R2N(E)N(W)$ and the same for R3 and R4

4th Compare the difference to the registers, i.e.:
$$Window1 = Diff < Th1$$
Do the same for Windows 2 and 3.

5th – 7th Extra time for calculations

8th Latch output

Output to VT201:

(0) VPD Mean-TAC difference inside Window-1
(1) VPD Mean-TAC difference inside Window-2
(2) VPD Mean-TAC difference inside Window-3
(3:13) Unused
(14) ADC-sum-E > th0
(15) ADC-sum-W > th0

Scalers:

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

6. Layer 0 QT Board: BBQ_ZD001

The single ZDC QT board receives signals from both the East and West sides of the ZDC. The algorithm can be configured to compare the ZDC-Front and ZDC-Back ADC values, and their digital sum, to a threshold (the "proton" logic) or to compare the analog sum to multiple thresholds (the "heavy ion" logic). It should be noted that the proton logic uses the "good hit" requirement for all hits (ADC > threshold and associated TAC in window) but the heavy ion logic does not. In the current setup both sides are configured to use the heavy-ion logic. The algorithm was written by Chris Perkins and is documented at
http://www.star.bnl.gov/public/trg/TSL/Software/qt_v6_f_doc.pdf

7. Layer 1 DSM Board: BBC_ZD101

The ZD101 DSM board processes data from the ZDC detector. The algorithm receives TAC data from the ZD001 QT board. 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 ZDC. A user-specified set of bits is then chosen to be passed on to VT201. In addition, the algorithm also receives the results of comparing sums to thresholds. Most of those threshold bits are passed through to VT201 unmodified. Two thresholds, one from each side, are protected by a "preceded" signal in order to deal with after-pulsing in the ZDC. This allows the user to zero out those two thresholds for a certain number of RHIC clock ticks after a ZDC coincidence is detected. In addition two other thresholds are combined (OR) to create a bit that can be used to define central triggers.

RBT File: bbc_zd101_2016_a.rbt

Users: ZD101

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

From the QT board:
bits 0:9 = West-1 TAC
bits 10:19 = East-1 TAC
bit 20 = West sum > threshold-0
bit 21 = West sum > threshold-1
bit 22 = West sum > threshold-2
bit 23 = West sum > threshold-3
bit 24 = West attenuated sum > threshold-4 (Unused in this algorithm)
bit 25 = West attenuated sum > threshold-5 (Unused in this algorithm)
bit 26 = East sum > threshold-0
bit 27 = East sum > threshold-1
bit 28 = East sum > threshold-2
bit 29 = East sum > threshold-3
bit 30 = East attenuated sum > threshold-4 (Unused in this algorithm)
bit 31 = East attenuated sum > threshold-5 (Unused in this algorithm)

LUT: 1:1

Registers:
R0: ZDC-TACdiff-select (2 bits)
0 => select bits 0:8
1 => select bits 1:9
2 => select bits 2:10
R1: ZDC-EastTAC-select (3)
0 => select bits 0:4
1 => select bits 1:5
…
5 => select bits 5:9
R2: ZDC-WestTAC-select (3)
Same value definitions as for R1

R3: ZDC-deadtime (4 bits)

Action:

1st Latch input

2nd Delay threshold-0,-1 and -2 bits to the 4th step.
Zero out a copy of the threshold-0 bits if the Preceded bit is set, i.e.
Protected-West-th0 = West-th0 and NOT Preceded
Protected-East-th0 = East-th0 and NOT Preceded
Combine the threshold-3 bits:
East_or_West = East-th3 or West-th3
Calculate: TAC difference = 1024 + TAC-E – TAC-W
Define: Good-TAC-E = TAC-E > 0, same for West side
For the scaler output, 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) or (9)
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 Delay the protected threshold bits and the East_or_West bit to the 4th step.
For the VT201 output use R0 to select the TAC difference bits, including overflow logic and the "good" TAC cut, i.e.:
Diff-overflow-0 = TAC-diff(9) or (10)
Diff-overflow-1 = TAC-diff(10)
If (Good-TAC-E = 0 or Good-TAC-W = 0) then output = 0
Else if (R0 = 0)
If (Diff-overflow-0 = 1) then output = 511
Else output = TAC-diff(0:8)
Else if (R0 = 1)
If (Diff-overflow-1 = 1) then output = 511
Else output = TAC-diff(1:9)
Else
Output = TAC-diff(2:10)
For the scaler output 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.
For the Preceded logic check for a ZDC coincidence:
Coincidence = Protected-West-th0 and Protcted-East-th0 and
Good-TAC-W and Good-TAC-E
If there is a coincidence then initialize a counter to R3-1. Allow it to count down to zero at a rate of one count per tick of the RHIC clock. Set the "Preceded" bit to one while the counter is counting.
NOTE: If R3 = 0 then the Preceded logic is disabled.

4th Latch output

Output to VT201:

(0-8) TAC difference
(9) Protected-sum-W > th0
(10) Sum-W > th1
(11) Sum-W > th2
(12) Protected-sum-E > th0
(13) Sum-E > th1
(14) Sum-E > th2
(15) Sum-E > th3 or Sum-W > th3

Scalers:

(0-4) selected bits of TAC-E
(5-9) selected bits of TAC-W
(10) Protected-sum-W > th0
(11) Sum-W > th0
(12) Protected-sum-E > th0
(13) Sum-E > th0
(14) ZDC coincidence
(15) Sum-E > th3 or Sum-W > th3

8. Layer 2 Vertex DSM Board: L1-VT201

All results of the Vertex tree are brought into the Vertex DSM. Threshold bits are passed on to the TCU either as individual bits or in in combinations. In parallel windows are placed around each TAC difference, and the "inside window" bits get passed through to the TCU and the scaler system. A minimum bias bit, based on an OR of information from all 4 detectors is created.

RBT File: l1_vt201_2016_b.rbt

Users: VT201

Inputs: Ch 0 = BB101
Ch 1 = BB102
Ch 2 = ZD101
Ch 3 = Unused
Ch 4 = VP101
Ch 5:7 = Unused

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

From Large tile BBC-DSM BB102
(0-12) Large tile TAC-Difference
(13) Unused
(14/15) East/West hit

From ZDC DSM ZD101
(0-8) TAC difference

(9) Protected-sum-W > th0
(10) Sum-W > th1
(11) Sum-W > th2
(12) Protected-sum-E > th0
(13) Sum-E > th1
(14) Sum-E > th2
(15) Sum-E > th3 or Sum-W > th3

From VPD-DSM VP101
(0) VPD Mean-TAC Difference inside Window-1
(1) VPD Mean-TAC Difference inside Window-2
(2) VPD Mean-TAC Difference inside Window-3
(3/13) 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 (9)
R5: ZDC-TACdiff-Max (9)
R6: Minimum-Bias-Select (4)
R7: BBClarge-Combo-Sel (1)

Action

1st Latch inputs

2nd Delay all the threshold bits that need to go to the TCU to the 4th step.
Delay a 2nd copy of the ZDC-th0 bits, the threshold bits from BBC-small and BBC-large and the VPD-window-1 bit to the 3rd step.
Delay a copy of the ZDC and BBC-small TAC difference to the 4th step.
Combine the ZDC (un-protected) th1 and th2 bits to make windows on the East and West sides separately, i.e.:
ZDC-E-Window = Sum-E > th1 and not Sum-E > th2
ZDC-W-Window = Sum-W > th1 and not Sum-W > th2
Compare each of the 3 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 Make the following combinations of ZDC bits:
ZDC-COINC = Protected-sum-W>th0 and Protected-sum-E>th0
ZDC-UPC = ZDC-E-Window and ZDC-W-Window
Combine the two BBC-large threshold bits using R7 to determine if the combination is an AND (R7 = 1) or an OR (R7 = 0)
Combine the results of the TAC difference comparisons to determine if each TAC difference is inside its specified window, e.g.:

ZDC-Tdiff = R4 < ZDC TAC difference < R6

Combine the results of the TAC difference comparisons and the ADC threshold bits to make the minimum bias bit, using R6 to turn each component on/off, i.e.:

MB = (R6(0) and BBC-S-Tdiff and BBC-S-E>th0 and BBC-S-W>th0) or
(R6(1) and BBC-L-Tdiff and BBC-L-E>th0 and BBC-L-W>th0) or
(R6(2) and ZDC-Tdiff) or
(R6(3) and VPD-Win1)

4th Latch Outputs

Output to TCU:

Bit	Name	Description
Bit 0	BBC-TAC	BBC small-tile TAC difference in window
Bit 1	BBC-E	BBC small-tile East ADC sum > threshold
Bit 2	BBC-W	BBC small-tile West ADC sum > threshold
Bit 3	VPD-TAC2	VPD Mean-TAC difference in window-2
Bit 4	BBC-Sum	BBC small-tile TAC sum in window
Bit 5	BBC-L	BBC large-tile East hit AND/OR West hit
Bit 6	ZDC-TAC	ZDC TAC difference in window
Bit 7	ZDC-E	ZDC Protected-sum-East > th0
Bit 8	ZDC-W	ZDC Protected-sum-West > th0
Bit 9	ZDC-UPC	ZDC East in window (th1 and th2) AND ZDC West in window (th1 and th2)
Bit 10	ZDC-EW	ZDC Sum-East > th3 or ZDC Sum-West > th3
Bit 11	Minimum-Bias	At least one selected TAC difference in window
Bit 12	VPD-TAC3	VPD Mean-TAC difference in window-3
Bit 13	VPD-TAC	VPD Mean-TAC difference in window-1
Bit 14	VPD-E	VPD East ADC sum > threshold
Bit 15	VPD-W	VPD West ADC sum > threshold

Output to Scalers

Bit	**Description**
Bit 0	BBC small-tile TAC difference in window
Bits 1:4	4 MSB of BBC small-tile TAC difference
Bit 5	BBC large-tile TAC difference in window
Bit 6	ZDC TAC difference in window
Bit 7	ZDC Protected-sum-East > th0 AND ZDC Protected-Sum-West > th0
Bits 8:10	3 MSB of ZDC TAC difference
Bit 11	VPD Mean-TAC difference in window-1
Bit 12	VPD Mean-TAC difference in window-2
Bit 13	VPD Mean-TAC difference in window-3
Bits 14:15	Unused