

Jet and ring-sum algorithm technical documentation

Andrew W. Rose

Abstract

Presented within this document are the specifications and implementation details for the components of the Jet and Ring-sum algorithms as currently implemented for the TDR calorimeter trigger upgrade.

Contents

Abstract	1
Design rules	5
Overview.....	7
Data types: Towers.....	9
tTower record.....	9
Array types	9
Data types: Jets.....	10
tJet record	10
tComparison record.....	10
tJetMaxima record	10
tFilteredJet record	10
Array types	10
Data types: Ring sums	12
tRingSegment record.....	12
tMHTcoefficients record.....	12
Array types	12
Algorithm: Tower Forming	13
Files.....	13
Generics.....	13
I/O	13
Latency	13
Description	14
Algorithm: 1×9 Strip Forming	15
Files.....	15
Generics.....	15
I/O.....	15
Latency	15
Description	15
Algorithm: 9×1 Strip Forming	17
Files.....	17
Generics.....	17
I/O.....	17

Latency	17
Description	17
Algorithm: 9×1 Vector Strip Forming	19
Files.....	19
Generics.....	19
I/O.....	19
Latency	19
Description	20
Notes	20
Algorithm: 9×9 Jet Sum	21
Files.....	21
Generics.....	21
I/O.....	21
Latency	21
Description	21
Notes	22
Algorithm: 9×9 Jet Veto.....	23
Files.....	23
Generics.....	23
I/O.....	23
Latency	23
Description	24
Notes	25
Algorithm: Donut pile-up estimation (DEPRECATED).....	27
Files.....	27
Generics.....	27
I/O.....	27
Latency	28
Description	28
Notes	28
Algorithm: Donut pile-up estimation	29
Files.....	29
Generics.....	29
I/O.....	30
Latency	30
Description	30
Notes	30
Algorithm: Jet Overlap Filter	33
Files.....	33
Generics.....	33
I/O.....	33
Latency	34
Description	34
Notes	34
Algorithm: Pile-up subtraction	35
Files.....	35
Generics.....	35
I/O.....	35
Latency	36

Description	36
Notes	36
Algorithm: MHT coefficients	37
Files.....	37
Generics.....	37
I/O.....	37
Latency	37
Description	38
Algorithm: HT/MHT sums.....	39
Files.....	39
Generics.....	39
I/O.....	39
Latency	40
Description	40
Algorithm: ET/MET sums.....	41
Files.....	41
Generics.....	41
I/O.....	41
Latency	42
Description	42
Algorithm: HT/MHT and ET/MET accumulation.....	43
Files.....	43
Generics.....	43
I/O.....	43
Latency	43
Description	44
Notes	44
Algorithm: Jet Sorting.....	45
Files.....	45
Generics.....	45
I/O.....	45
Latency	45
Description	46
Notes	46
Algorithm: Accumulating Jet Sorting.....	47
Files.....	47
Generics.....	47
I/O.....	47
Latency	47
Description	48
Notes	48
Algorithm: Coordinate conversion	49
Files.....	49
Generics.....	49
I/O.....	49
Latency	49
Description	50
Notes	50

Design rules

The following strategy is preferred and used in the Jet and Ring-Sum algorithm firmware:

- At top level, there shall be no clocked logic, only instances of entities
- All data will always be accompanied by a local data-valid flag
- Global counters, resets or data-valid flags are forbidden
- All signals connecting two entities shall be “pipes”: that is, data shall be placed into a clocked pipeline before leaving an entity
- Data will be clocked along the pipeline in order of increasing cell index
- Data will be clocked out of any logic and will not be clocked into the zeroth cell of the pipeline
- The eta-0 boundary will be handled using the multiplex-on-input principle
- CamelCase will be used throughout the firmware
- BLOCK CAPITALS will be used for all VHDL keywords
- Types will be prefixed with a lower-case “t”
- Constants will be prefixed with a lower-case “c”
- All entity port names will be appended with “In” or “Out” to immediately identify the direction of data into or out of the algorithm

Overview

The Jet and Ring-sum algorithm and data flow is structured as shown in Figure 1.

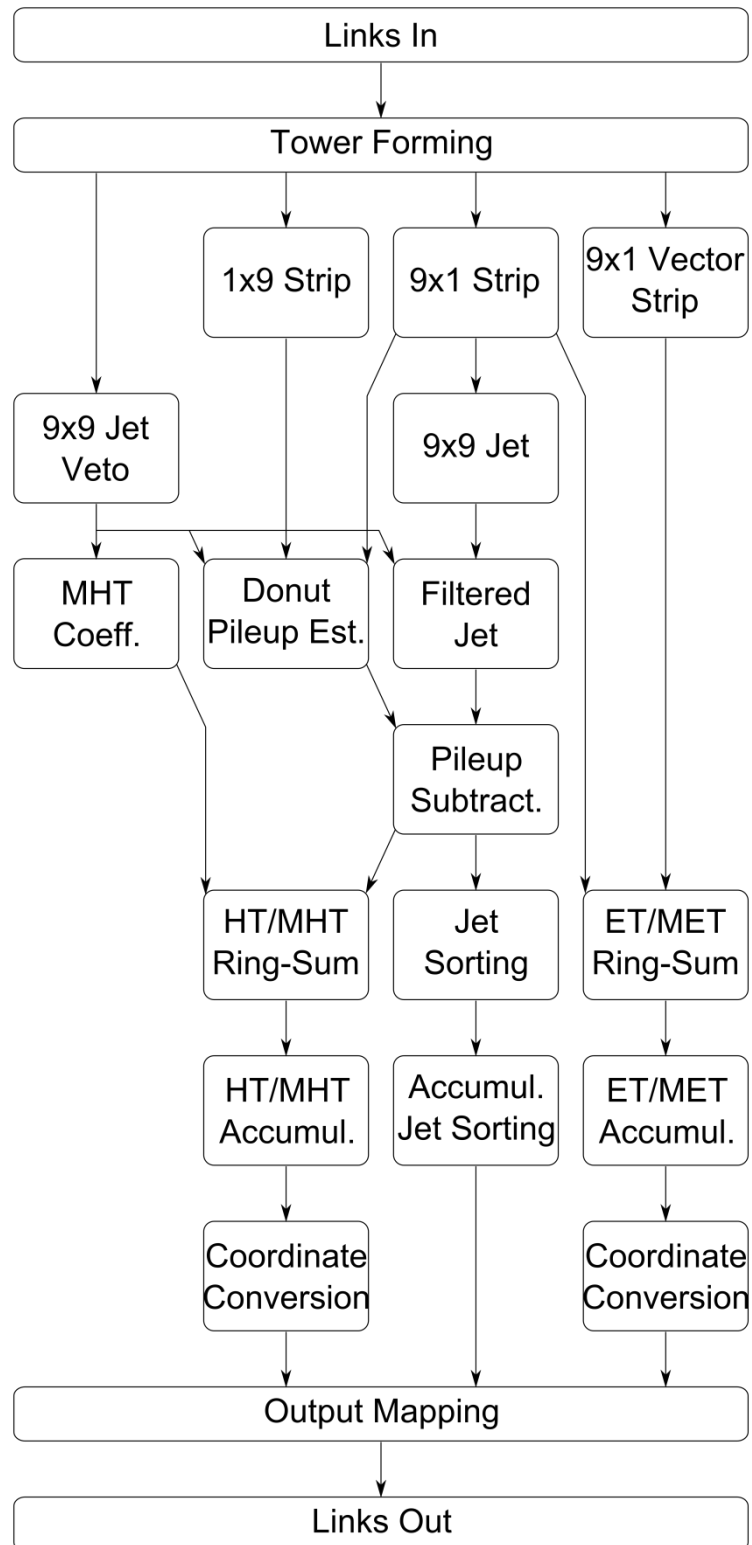


Figure 1: Algorithm- and data-flow of the Jet and Ring-Sum algorithms.

Data types: Towers

tTower record

EcalFG	STD_LOGIC	The Ecal finegrain flag
HcalFG	STD_LOGIC	An Hcal feature bit
Energy	UNSIGNED(8 DOWNT0 0)	The total E+H energy
Ecal	UNSIGNED(16 DOWNT0 0)	An estimate of the Ecal energy
Hcal	UNSIGNED(16 DOWNT0 0)	An estimate of the Hcal energy
DataValid	BOOLEAN	The data-valid flag propagated with the data

Array types

tTowersInPhi	ARRAY(cTowersInPhi-1 DOWNT0 0) OF tTower
tTowersInEtaPhi	ARRAY(cRegionsInEta-1 DOWNT0 0) OF tTowersInPhi
tTowersPipe	ARRAY(NATURAL RANGE <>) OF tTowersInEtaPhi

- Towers are formed at every site: Number of towers in phi and number of regions are fixed by geometry of the detector
- The length of the pipe is determined by final application, so set as an unbounded array and let the length be assigned at usage

Data types: Jets

tJet record

Energy	UNSIGNED(15 DOWNT0 0)	The energy of the 9x9 jet or part thereof
DataValid	BOOLEAN	The data-valid flag propagated with the data

tComparison record

Data	BOOLEAN	The result of a comparison or of a Boolean-logic expression
DataValid	BOOLEAN	The data-valid flag propagated with the data

tJetMaxima record

Max	UNSIGNED(8 DOWNT0 0)	The energy of the maximum-energy tower in a region
Phi	INTEGER RANGE 0 TO 8	The local phi-coordinate of the maximum-energy tower in a region
Eta	INTEGER RANGE 0 TO 3	The local eta-coordinate of the maximum-energy tower in a region
DataValid	BOOLEAN	The data-valid flag propagated with the data

tFilteredJet record

Energy	UNSIGNED(15 DOWNT0 0)	The energy of the 9x9 jet or part thereof
Phi	INTEGER RANGE 0 TO 71	The global phi-coordinate of the jet
Eta	INTEGER RANGE 0 TO 35	The global eta-coordinate of the jet
DataValid	BOOLEAN	The data-valid flag propagated with the data

Array types

tJetsInPhi	ARRAY(cTowersInPhi-1 DOWNT0 0) OF tJet
tJetsInEtaPhi	ARRAY(cRegionsInEta-1 DOWNT0 0) OF tJetsInPhi
tJetsPipe	ARRAY(NATURAL RANGE <>) OF tJetsInEtaPhi

- Jets are formed at every site: Number of towers in phi and number of regions are fixed by geometry of the detector
- The length of the pipe is determined by final application, so set as an unbounded array and let the length be assigned at usage

tComparisonsInPhi	ARRAY(cTowersInPhi-1 DOWNT0 0) OF tComparison
tComparisonsInEtaPhi	ARRAY(cRegionsInEta-1 DOWNT0 0) OF tComparisonsInPhi
tComparisonsPipe	ARRAY(NATURAL RANGE <>) OF tComparisonsInEtaPhi

- Jet veto calculation is performed at every site: Number of towers in phi and number of regions are fixed by geometry of the detector
- The length of the pipe is determined by final application, so set as an unbounded array and let the length be assigned at usage

tJetMaximalnPhi	ARRAY(cTowersInPhi-1 DOWNT0 0) OF tJetMaxima
tJetMaximalnEtaPhi	ARRAY(cRegionsInEta-1 DOWNT0 0) OF tJetMaximalnPhi
tJetMaximaPipe	ARRAY(NATURAL RANGE <>) OF tJetMaximalnEtaPhi

- Jet maxima-finding calculation is performed within the Jet veto calculation at every site: Number of towers in phi and number of regions are fixed by geometry of the detector
- The length of the pipe is determined by final application, so set as an unbounded array and let the length be assigned at usage

tFilteredJetsInPhi	ARRAY(NATURAL RANGE <>) OF tFilteredJet
tFilteredJetsInEtaPhi	ARRAY(cRegionsInEta-1 DOWNT0 0) OF tFilteredJetsInPhi((cTowersInPhi / 4) -1 DOWNT0 0)
tFilteredJetsPipe	ARRAY(NATURAL RANGE <>) OF tFilteredJetsInEtaPhi

- The number of filtered jets in phi is determined by final application, so set as an unbounded array and let the length be assigned at usage.
- The main use case for the Filtered-Jet type is in handling the overlap-filtered jets. In this case, the number of jets in phi and number of regions are fixed by geometry of the detector and by the size of the jet
- The length of the pipe is determined by final application, so set as an unbounded array and let the length be assigned at usage

Data types: Ring sums

tRingSegment record

Energy	UNSIGNED(19 DOWNT0 0)	The total energy of all the towers or jets in a ring around phi or part thereof
xComponent	SIGNED(19 DOWNT0 0)	The x-component of the energy of all the towers or jets in a ring around phi or part thereof
yComponent	SIGNED(19 DOWNT0 0)	The y-component of the energy of all the towers or jets in a ring around phi or part thereof
towersOverThreshold	UNSIGNED(11 DOWNT0 0)	A count of the number of towers over threshold in a ring around phi or part thereof
DataValid	BOOLEAN	The data-valid flag propagated with the data

tMHTcoefficients record

CosineCoefficients	SIGNED(9 DOWNT0 0)	The value cosine(phi) with range -511 to 511 stored as a 10-bit signed integer
SineCoefficients	SIGNED(9 DOWNT0 0)	The value sine(phi) with range -511 to 511 stored as a 10-bit signed integer

Array types

tRingSegmentsInPhi	ARRAY(cTowersInPhi-1 DOWNT0 0) OF tRingSegment
tRingSegmentsInEtaPhi	ARRAY(cRegionsInEta-1 DOWNT0 0) OF tRingSegmentsInPhi
tRingSegmentsPipe	ARRAY(NATURAL RANGE <>) OF tRingSegmentsInEtaPhi

- Ring Segments are formed at every site: Number of towers in phi and number of regions are fixed by geometry of the detector
- The length of the pipe is determined by final application, so set as an unbounded array and let the length be assigned at usage

tRingSegmentsInEta	ARRAY(cRegionsInEta-1 DOWNT0 0) OF tRingSegment
tRingSegmentsPipe2	ARRAY(NATURAL RANGE <>) OF tRingSegmentsInEta

- Once a ring is fully formed, there is no phi information, so drop it: number of regions are fixed by geometry of the detector
- The length of the pipe is determined by final application, so set as an unbounded array and let the length be assigned at usage

tMHTcoefficientsInPhi	ARRAY((cTowersInPhi / 4) -1 DOWNT0 0) OF tMHTcoefficients
tMHTcoefficientsInEtaPhi	ARRAY(cRegionsInEta-1 DOWNT0 0) OF tMHTcoefficientsInPhi
tMHTcoefficientsPipe	ARRAY(NATURAL RANGE <>) OF tMHTcoefficientsInEtaPhi

- MHT coefficients are required for every filtered Jet: Number of towers in phi and number of regions are fixed by geometry of the detector and by the size of the jet
- The length of the pipe is determined by final application, so set as an unbounded array and let the length be assigned at usage

Algorithm: Tower Forming

Files

Top-level	links/LinksIn.vhd
Types	components/mp7_datapath/firmware/hdl/mp7_data_types.vhd projects/calol2/algorithm_components/firmware/hdl/towers/types_pkg.vhd
Pipes	projects/calol2/algorithm_components/firmware/hdl/towers/TowerPipe.vhd
Internal entities	projects/calol2/algorithm_components/firmware/hdl/towers/TowerFormer.vhd projects/calol2/algorithm_components/firmware/hdl/towers/TowerSum.vhd projects/calol2/algorithm_components/firmware/cgn/multiplier9Ux7U.xco
Utilities	projects/calol2/algorithm_components/firmware/hdl/towers/functions_pkg.vhd

Generics

Pipe lengths	towerPipeOutLength
Pipe Offsets	-
Other	-

I/O

Inputs	Name	linksIn
	Type	ldata((cNumberOfLinksIn-1) DOWNT0 0)
	Description	Raw link data in. Note that this is “flat” data, not a pipe
Outputs	Name	towerPipeOut
	Type	tTowersPipe((towerPipeOutLength-1) DOWNT0 0)
	Description	Formatted towers, organized by eta-half/phi

Latency

Clock cycles	2
BX	$\frac{1}{3}$

Description

- GENERATE loop over all links and for each link,
 - Split word into two 16-bit words
 - Pass each 16-bit word to TowerFormer
 - Assign output of TowerFormers to individual towers
- The TowerFormer entity then
 - Extracts energy and feature bits from 16-bit word
 - Estimates the Ecal and Hcal components of the tower based on the total energy and the E-over-H ratio

Algorithm: 1×9 Strip Forming

Files

Top-level	projects/calol2/algorithm_components/firmware/hdl/jets/Strip1x9Former.vhd
Types	projects/calol2/algorithm_components/firmware/hdl/jets/types_pkg.vhd
Pipes	projects/calol2/algorithm_components/firmware/hdl/jets/JetPipe.vhd
Internal entities	projects/calol2/algorithm_components/firmware/hdl/jets/JetSum.vhd
Utilities	projects/calol2/algorithm_components/firmware/hdl/jets/functions_pkg.vhd

Generics

Pipe lengths	towerPipeInLength stripPipeOutLength
Pipe Offsets	-
Other	-

I/O

Inputs	Name	towerPipeIn
	Type	tTowersPipe(towerPipeInLength-1 DOWNT0 0)
	Description	Formatted towers organized by eta-half/phi
Outputs	Name	strips1x9SumPipeOut
	Type	tJetsPipe(stripPipeOutLength-1 DOWNT0 0)
	Description	Strips 1 in phi by 9 in eta, organized by eta-half/phi

Latency

Clock cycles	6
BX	1

Description

- GENERATE loop over all towers in phi and both halves in eta and for each site
 - Sum three neighbouring towers in eta into a strip 1×3
 - Sum three non-overlapping neighbouring strip 1×3 objects in eta into a strip 1×9

Algorithm: 9×1 Strip Forming

Files

Top-level	projects/calol2/algorithm_components/firmware/hdl/jets/Strip9x1Former.vhd
Types	projects/calol2/algorithm_components/firmware/hdl/jets/types_pkg.vhd
Pipes	projects/calol2/algorithm_components/firmware/hdl/jets/JetPipe.vhd
Internal entities	projects/calol2/algorithm_components/firmware/hdl/jets/JetSum.vhd
Utilities	projects/calol2/algorithm_components/firmware/hdl/jets/functions_pkg.vhd

Generics

Pipe lengths	towerPipeInLength stripPipeOutLength
Pipe Offsets	-
Other	-

I/O

Inputs	Name	towerPipeIn
	Type	tTowersPipe(towerPipeInLength-1 DOWNT0 0)
	Description	Formatted towers organized by eta-half/phi
Outputs	Name	strips9x1SumPipeOut
	Type	tJetsPipe(stripPipeOutLength-1 DOWNT0 0)
	Description	Strips 9 in phi by 1 in eta, organized by eta-half/phi

Latency

Clock cycles	2
BX	1/3

Description

- GENERATE loop over all towers in phi and both halves in eta and for each site
 - Sum three neighbouring towers in phi into a strip 3×1
 - Sum three non-overlapping neighbouring strip 3×1 objects in eta into a strip 9×1

Algorithm: 9×1 Vector Strip Forming

Files

Top-level	projects/calol2/algorithm_components/firmware/hdl/rings/VectorStrip9x1Former.vhd
Types	projects/calol2/algorithm_components/firmware/hdl/rings/types_pkg.vhd
Pipes	projects/calol2/algorithm_components/firmware/hdl/rings/RingPipe.vhd
Internal entities	projects/calol2/algorithm_components/firmware/cgn/multiplier9Ux10S.xco projects/calol2/algorithm_components/firmware/hdl/rings/RingSegmentSum.vhd
Utilities	projects/calol2/algorithm_components/firmware/hdl/rings/functions_pkg.vhd

Generics

Pipe lengths	towerPipeInLength stripPipeOutLength
Pipe Offsets	-
Other	vectorTowerShift

I/O

Inputs	Name	towerPipeIn
	Type	tTowersPipe(towerPipeInLength-1 DOWNT0 0)
	Description	Formatted towers organized by eta-half/phi
	Name	energyThresholdIn
	Type	UNSIGNED(10 DOWNT0 0)
	Description	Energy threshold which will be used in the Towers-over-Threshold calculation
Outputs	Name	strips9x1SumPipeOut
	Type	tRingSegmentsPipe(stripPipeOutLength-1 DOWNT0 0)
	Description	Ring-segments 9 in phi by 1 in eta, organized by eta-half/phi

Latency

Clock cycles	4
BX	2/3

Description

- GENERATE loop over all towers in phi and both halves in eta and for each site
 - Compile-time calculate Sin(phi) and Cos(phi) from GENERATE parameter
 - Multiply energy by Sin(phi) and Cos(phi) to get x- and y-components of the vector sum
 - Right-shift the x- and y-components of the vector sum by vectorTowerShift bits and truncate the least-significant bits
 - Compare the tower energy to the input “energyThresholdIn” and, if greater than or equal to the threshold, label the tower as being over threshold.
- GENERATE loop over 1/3 towers in phi and both halves in eta and for each site
 - Sum three neighbouring towers in phi into a strip 3×1
- GENERATE loop over 1/9 towers in phi and both halves in eta and for each site
 - Sum three neighbouring strip 3×1 objects in eta into a strip 9×1

Notes

Unlike the 1×9 Strip Forming and 9×1 Strip Forming, the 9×1 Vector Strip Forming does not create 72 overlapping 9-tower strips, but 8 non-overlapping 9-tower strips.

Algorithm: 9×9 Jet Sum

Files

Top-level	projects/calol2/algorithm_components/firmware/hdl/jets/JetFormer.vhd
Types	projects/calol2/algorithm_components/firmware/hdl/jets/types_pkg.vhd
Pipes	projects/calol2/algorithm_components/firmware/hdl/jets/JetPipe.vhd
Internal entities	projects/calol2/algorithm_components/firmware/hdl/jets/JetSum.vhd
Utilities	projects/calol2/algorithm_components/firmware/hdl/jets/functions_pkg.vhd

Generics

Pipe lengths	stripPipeInLength jetPipeOutLength
Pipe Offsets	-
Other	jetFormingOffset

I/O

Inputs	Name	strip9x1PipeIn
	Type	tJetsPipe(stripPipeInLength-1 DOWNT0 0)
	Description	Strips 9 in phi by 1 in eta, organized by eta-half/phi
Outputs	Name	jet9x9PipeOut
	Type	tJetsPipe(jetPipeOutLength-1 DOWNT0 0)
	Description	Energy sums 9 in phi by 9 in eta, organized by eta-half/phi

Latency

Clock cycles	6
BX	1

Description

- GENERATE loop over all towers in phi and both halves in eta and for each site
 - Sum three neighbouring 9×1 strips in eta into a 9×3 object
 - Sum three non-overlapping neighbouring 9×3 objects in eta into a 9×9 jet

Notes

Because the pipes for the 9x1 strips are long anyway because of the donut pile-up subtraction and because the 9x9 sum is not needed until the jet veto is calculated, I have added a generic “jetFormingOffset” which delays the forming of the 9x9 sum until it is needed so that both the inputs (9x1 strips) and outputs (9x9 strips) are being stored in pipes simultaneously, thereby unnecessarily using resources.

Algorithm: 9×9 Jet Veto

Files

Top-level	projects/calol2/algorithm_components/firmware/hdl/jets/JetOverlapCriteria.vhd
Types	projects/calol2/algorithm_components/firmware/hdl/jets/types_pkg.vhd
Pipes	projects/calol2/algorithm_components/firmware/hdl/jets/JetPipe.vhd projects/calol2/algorithm_components/firmware/hdl/jets/JetComparisonPipe.vhd projects/calol2/algorithm_components/firmware/hdl/jets/JetMaximaPipe.vhd
Internal entities	-
Utilities	projects/calol2/algorithm_components/firmware/hdl/jets/functions_pkg.vhd

Generics

Pipe lengths	towerPipeInLength overlapVetoPipeOutLength
Pipe Offsets	-
Other	-

I/O

Inputs	Name	towerPipeIn
	Type	tTowersPipe(towerPipeInLength-1 DOWNT0 0)
	Description	Formatted towers organized by eta-half/phi
Outputs	Name	overlapVetoPipeOut
	Type	tComparisonsPipe(overlapVetoPipeOutLength-1 DOWNT0 0)
	Description	Boolean specifying whether the central tower of the region 9 in phi by 9 in eta has been vetoed by another tower in the region, organized by eta-half/phi

Latency

Clock cycles	11
BX	1 5/6

Description

- GENERATE loop over all towers in phi and both halves in eta and for each site
 - For each group of 3 neighbouring towers in phi, compare the towers and find the maximum energy.
 - Create a 3-tower maxima object which stores the maximum tower energy and the relative position of the maximum tower in phi (0-2)
 - For each group of 3 non-overlapping neighbouring 3-tower maxima objects in phi, compare the energy of each and find the maximum energy.
 - Create a 9-tower maxima object which stores the maximum tower energy and the relative position of the maximum tower in phi (0-8)
 - For each group of 3 neighbouring 9-tower maxima objects in eta, identify the maxima object satisfying the most stringent test criteria (as specified in Figure 2 and Figure 3). Because of the complexity of the logic here:
 - The comparisons of energies and phi are performed in one clock step
 - The combinatorial logic is performed in a second clock step **setting a variable**
 - The variable then selects the appropriate value for assignments to a 9×3-tower maxima object which stores the maximum tower energy, the relative position of the maximum tower in phi (0-8) and the relative position of the maximum tower in eta (0-2).
 - For each group of 3 non-overlapping neighbouring 9×3-tower maxima objects in eta, set the veto flag if any of the following are true:
 - The maxima in the central region is not the central tower
 - The maxima in either the left or right regions is greater than the maxima in the central region
 - The maxima in either the left or right regions is equal to the the maxima in the central region and the maxima lies within the “<=” region as shown in Figure 2.

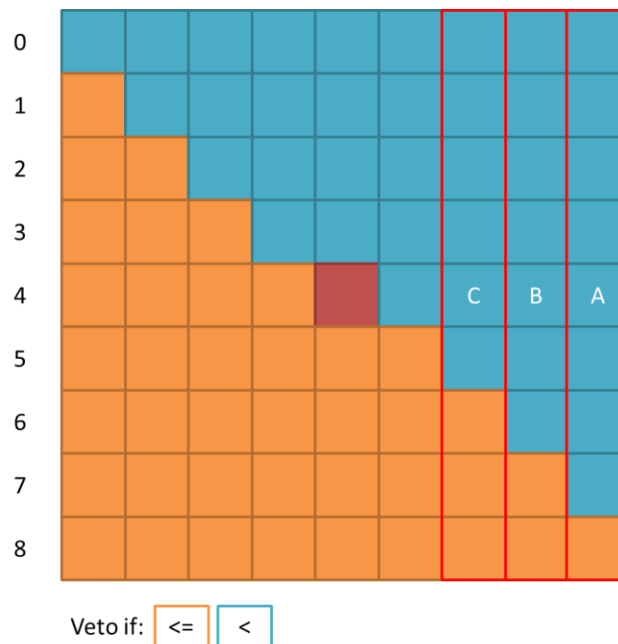



Figure 2: Geometry of the Jet Overlap Criteria for the 9×3 regions. The numbers down the left hand side indicate relative tower position in phi.

A>B>C	A
A>B=C	A
A>C>B	A
A>C=B	DUPLICATE
A=B>C	MaxPhi(A,B)
A=B=C	MaxPhi(A,B,C)
A=C>B	MaxPhi(A,C)
A=C=B	DUPLICATE
B>A>C	B
B>A=C	B
B>C>A	B
B>C=A	DUPLICATE
B=A>C	DUPLICATE
B=A=C	DUPLICATE
B=C>A	MaxPhi(B,C)
B=C=A	DUPLICATE
C>A>B	C
C>A=B	C
C>B>A	C
C>B=A	DUPLICATE
C=A>B	DUPLICATE
C=A=B	DUPLICATE
C=B>A	DUPLICATE
C=B=A	DUPLICATE



A>B>C	A
A>B=C	A
A>C>B	A
A=B>C	MaxPhi(A,B)
A=B=C	MaxPhi(A,B,C)
A=C>B	MaxPhi(A,C)
B>A>C	B
B>A=C	B
B>C>A	B
B=C>A	MaxPhi(B,C)
C>A>B	C
C>A=B	C
C>B>A	C

Figure 3: Left: The exhaustive list of permutations of the relative energies of A, B and C, where A, B and C are defined in Figure 2, and which of A, B and C would represent the most stringent condition for comparison with the central tower. Right: the same list with duplicates removed for clarity.

MaxPhi is a function which identifies which strip has its maximum furthest from the diagonal in the positive phi direction.

Notes

The extreme lengths gone to within this algorithm are necessary to prevent an 80-way fan-in to a single point at every site and to then do 80-non reusable comparisons. As it stands each 3×1 comparison is reused three times, each 9×1 comparison reused three times and each 9x3 comparison reused 3 times and the biggest fan-in is only ever 3:1.

Algorithm: Donut pile-up estimation (DEPRECATED)

Files

Top-level	projects/calol2/algorithm_components/firmware/hdl/jets/PileUpEstimator.vhd
Types	projects/calol2/algorithm_components/firmware/hdl/jets/types_pkg.vhd
Pipes	projects/calol2/algorithm_components/firmware/hdl/jets/JetPipe.vhd
Internal entities	projects/calol2/algorithm_components/firmware/hdl/jets/JetBitonicSort.vhd
Utilities	projects/calol2/algorithm_components/firmware/hdl/jets/functions_pkg.vhd

Generics

Pipe lengths	strip9x1PipeInLength strip1x9PipeInLength pileUpPipeOutLength
Pipe Offsets	-
Other	-

I/O

Inputs	Name	strip9x1PipeIn
	Type	tJetsPipe(strip9x1PipeInLength-1 DOWNT0 0)
	Description	Strips 9 in phi by 1 in eta, organized by eta-half/phi
	Name	strip1x9PipeIn
	Type	tJetsPipe(strip1x9PipeInLength-1 DOWNT0 0)
Outputs	Description	Strips 1 in phi by 9 in eta, organized by eta-half/phi
	Name	pileUpPipeOut
	Type	tJetsPipe(pileUpPipeOutLength-1 DOWNT0 0)
	Description	An energy sum representing the estimated pileup in a region of size 9 in phi by 9 in eta, organized by eta-half/phi

Latency

Clock cycles	5
--------------	---

BX	5/6
----	-----

Description

- GENERATE loop over all towers in phi and both halves in eta and for each site
 - Select the four strips which abut the Jet on each side
 - Bitonic-sort the four strips by energy (see section “Algorithm: Jet Sorting”)
 - Add the middle two strips in the sorted list and left-shift by 2 (Multiply by 4) to compensate for relative area of strips and jet (should really be 4.5).

Notes

Since the Bitonic sort acts on “FilteredJet” objects, rather than simply the “Jet” objects, we must use the conversion function “ToFilteredJet” and define the local-type “tFilteredJetsInputsInPhi” to hold a 4-entry input for the Bitonic sort.

Algorithm: Donut pile-up estimation

Files

Top-level	projects/calol2/algorithm_components/firmware/hdl/jets/PileUpEstimator.vhd
Types	projects/calol2/algorithm_components/firmware/hdl/jets/types_pkg.vhd
Pipes	projects/calol2/algorithm_components/firmware/hdl/jets/JetPipe.vhd
Internal entities	projects/calol2/algorithm_components/firmware/hdl/jets/JetBitonicSort.vhd
Utilities	projects/calol2/algorithm_components/firmware/hdl/jets/functions_pkg.vhd

Generics

Pipe lengths	jetVetoPipeInLength strip9x1PipeInLength strip1x9PipeInLength pileUpPipeOutLength
Pipe Offsets	stripPipeOffset
Other	-

I/O

Inputs	Name	strip9x1PipeIn
	Type	tJetsPipe(strip9x1PipeInLength-1 DOWNT0 0)
	Description	Strips 9 in phi by 1 in eta, organized by eta-half/phi
	Name	strip1x9PipeIn
	Type	tJetsPipe(strip1x9PipeInLength-1 DOWNT0 0)
	Description	Strips 1 in phi by 9 in eta, organized by eta-half/phi
	Name	jetVetoPipeIn
	Type	tComparisonsPipe(jetVetoPipeInLength-1 DOWNT0 0)
	Description	Boolean specifying whether the central tower of the region 9 in phi by 9 in eta has been vetoed by another tower in the region, organized by eta-half/phi
Outputs	Name	pileUpPipeOut
	Type	tJetsPipe(pileUpPipeOutLength-1 DOWNT0 0)
	Description	An energy sum representing the estimated pileup in a region of size 9 in phi by 9 in eta, organized by eta-half/phi

Latency

Clock cycles	?
BX	?

Description

- GENERATE loop over all towers in phi and both halves in eta and for each site
 - Select the four strips which abut the Jet on each side
 - Apply the jet veto filter to calculate the donut only for jets which are not vetoed
 - Bitonic-sort the four strips by energy (see section “Algorithm: Jet Sorting”)
 - Add the middle two strips in the sorted list and left-shift by 2 (Multiply by 4) to compensate for relative area of strips and jet (should really be 4.5).

Notes

Since the Bitonic sort acts on “FilteredJet” objects, rather than simply the “Jet” objects, we must use the conversion function “ToFilteredJet” and define the local-type “tFilteredJetsInputsInPhi” to hold a 4-entry input for the Bitonic sort.

By applying the veto before donut creation, we reduce the number of 4-input sorts by a factor of 4. The now-deprecated donut pile-up estimation accounted for 25% of the algorithm resource usage, purely because of the sorts.

Since we must now wait for the veto to arrive before we can do the sort + sum, this has had the effect of delaying the pile-up estimation by 3 clock-cycles compared to the now-deprecated donut pile-up estimation which calculated the estimates prior to the formation of the veto.

Algorithm: Jet Overlap Filter

Files

Top-level	projects/calol2/algorithm_components/firmware/hdl/jets/JetOverlapFilter.vhd
Types	projects/calol2/algorithm_components/firmware/hdl/jets/types_pkg.vhd
Pipes	projects/calol2/algorithm_components/firmware/hdl/jets/JetPipe.vhd projects/calol2/algorithm_components/firmware/hdl/jets/FilteredJetPipe.vhd projects/calol2/algorithm_components/firmware/hdl/jets/JetComparisonPipe.vhd
Internal entities	-
Utilities	projects/calol2/algorithm_components/firmware/hdl/jets/functions_pkg.vhd

Generics

Pipe lengths	jetSumPipeInLength jetVetoPipeInLength filteredJetsPipeOutLength
Pipe Offsets	SumPipeOffset VetoPipeOffset
Other	-

I/O

Inputs	Name	jetSumPipeIn
	Type	tJetsPipe(jetSumPipeInLength-1 DOWNT0 0)
	Description	Energy sums 9 in phi by 9 in eta, organized by eta-half/phi
	Name	jetVetoPipeIn
	Type	tComparisonsPipe(jetVetoPipeInLength-1 DOWNT0 0)
Outputs	Description	Boolean specifying whether the central tower of the region 9 in phi by 9 in eta has been vetoed by another tower in the region, organized by eta-half/phi
	Name	filteredJetsPipeOut
	Type	tFilteredJetsPipe(filteredJetsPipeOutLength-1 DOWNT0 0)
	Description	Energy sums 9 in phi by 9 in eta + global coordinate, central tower guaranteed to be the highest energy, organized by eta-half/phi

Latency

Clock cycles	2
--------------	---

BX	1/3
----	-----

Description

- GENERATE loop over 1/4 towers in phi and both halves in eta and for each site
 - Select the jet for output based on which if any of the four overlapping jet sites was not vetoed.
 - Assign the value of phi to the jet for output based on the GENERATE parameter

Notes

For the now-deprecated pile-up estimation, this same code is used to select both the 9×9 energy sum and the pile-up subtraction estimate based on the jet veto bit so that the pile-up subtraction is only done on the filtered jets, rather than on the 3/4 jets which will be rejected by the overlap filtering anyway. In the current pile-up estimation scheme, the veto is applied explicitly within the pile-up estimation step, before the estimate is made. For a global pile-up estimation, this step would again be required.

The overlap filtering must assign an explicit phi-value to the candidates or else their position will be lost. This could be a relative position (0-3), but since the candidates will go into an 18-way sort after this, I assign global phi (0-71).

Algorithm: Pile-up subtraction

Files

Top-level	projects/calol2/algorithm_components/firmware/hdl/jets/PileUpSubtraction.vhd
Types	projects/calol2/algorithm_components/firmware/hdl/jets/types_pkg.vhd
Pipes	projects/calol2/algorithm_components/firmware/hdl/jets/FilteredJetPipe.vhd
Internal entities	-
Utilities	projects/calol2/algorithm_components/firmware/hdl/jets/functions_pkg.vhd

Generics

Pipe lengths	filteredJetsPipeInLength filteredPileUpPipeInLength pileUpSubtractedJetsPipeOutLength
Pipe Offsets	-
Other	-

I/O

Inputs	Name	filteredJetsPipeIn
	Type	tFilteredJetsPipe(filteredJetsPipeInLength-1 DOWNT0 0)
	Description	Energy sums 9 in phi by 9 in eta + global phi pos., central tower guaranteed to be the highest energy, organized by eta-half/phi
	Name	filteredPileUpPipeIn
	Type	tFilteredJetsPipe(filteredPileUpPipeInLength-1 DOWNT0 0)
	Description	An energy sum representing the estimated pileup in a region of size 9 in phi by 9 in eta, organized by eta-half/phi
Outputs	Name	pileUpSubtractedJetsPipeOut
	Type	tFilteredJetsPipe(pileUpSubtractedJetsPipeOutLength-1 DOWNT0 0)
	Description	Corrected energy sums 9 in phi by 9 in eta + global coordinate, central tower guaranteed to be the highest energy, organized by eta-half/phi

Latency

Clock cycles	1
--------------	---

BX	1/6
----	-----

Description

- GENERATE loop over 1/4 towers in phi and both halves in eta and for each site
 - Convert both the Jet Sum energy and the Pileup estimate to integer form
 - Subtract the Pileup estimate from the Jet Sum energy
 - If the result is less than zero, set result to zero
 - If the result is greater than or equal to zero, output the result as an unsigned

Notes

No offsets are provided for the two input types, since there is freedom in the preceding algorithms to adjust the timings such that both inputs become valid simultaneously

Algorithm: MHT coefficients

Files

Top-level	projects/calol2/algorithm_components/firmware/hdl/rings/MHTcoefficients.vhd
Types	projects/calol2/algorithm_components/firmware/hdl/rings/types_pkg.vhd
Pipes	projects/calol2/algorithm_components/firmware/hdl/jets/JetComparisonPipe.vhd projects/calol2/algorithm_components/firmware/hdl/rings/MHTcoefficientsPipe.vhd
Internal entities	-
Utilities	projects/calol2/algorithm_components/firmware/hdl/rings/functions_pkg.vhd

Generics

Pipe lengths	jetVetoPipeInLength MHTcoefficientsPipeOutLength
Pipe Offsets	jetVetoPipeOffset
Other	-

I/O

Inputs	Name	jetVetoPipeIn
	Type	tComparisonsPipe(jetVetoPipeInLength-1 DOWNT0 0)
	Description	Boolean specifying whether the central tower of the region 9 in phi by 9 in eta has been vetoed by another tower in the region, organized by eta-half/phi
Outputs	Name	MHTcoefficientsPipeOut
	Type	tMHTcoefficientsPipe(MHTcoefficientsPipeOutLength-1 DOWNT0 0)
	Description	Global trigonometric coefficients of the non-vetoed jet (if any), organized by eta-half/phi

Latency

Clock cycles	1
BX	1/6

Description

- GENERATE loop over $1/4$ towers in ϕ and both halves in η and for each site
 - Compile-time calculate $\text{Sin}(\phi)$ and $\text{Cos}(\phi)$ from GENERATE parameter
 - Select the coefficient for output based on which, if any, of the four overlapping jet sites was not vetoed.

Algorithm: HT/MHT sums

Files

Top-level	projects/calol2/algorithm_components/firmware/hdl/rings/HTandMHTsums.vhd
Types	projects/calol2/algorithm_components/firmware/hdl/rings/types_pkg.vhd
Pipes	projects/calol2/algorithm_components/firmware/hdl/rings/MHTcoefficientsPipe.vhd projects/calol2/algorithm_components/firmware/hdl/rings/RingPipe.vhd
Internal entities	projects/calol2/algorithm_components/firmware/cgn/multiplier16Ux10S.xco projects/calol2/algorithm_components/firmware/hdl/rings/RingSegmentSum.vhd
Utilities	projects/calol2/algorithm_components/firmware/hdl/rings/functions_pkg.vhd

Generics

Pipe lengths	pileUpSubtractedJetsPipeInLength MHTcoefficientsPipeInLength ringPipeOutLength
Pipe Offsets	-
Other	vectorJetShift

I/O

Inputs	Name	pileUpSubtractedJetsPipeIn
	Type	tFilteredJetsPipe(pileUpSubtractedJetsPipeInLength-1 DOWNT0 0)
	Description	Corrected energy sums 9 in phi by 9 in eta + global coordinate, central tower guaranteed to be the highest energy, organized by eta-half/phi
	Name	MHTcoefficientsPipeIn
	Type	tMHTcoefficientsPipe(MHTcoefficientsPipeInLength-1 DOWNT0 0)
	Description	Global trigonometric coefficients of the non-vetoed jet (if any), organized by eta-half/phi
Outputs	Name	ringPipeOut
	Type	tRingSegmentsPipe2(ringPipeOutLength-1 DOWNT0 0)
	Description	Scalar and vector sum of jet energy, organized by eta-half

Latency

Clock cycles	4
--------------	---

BX	2/3
----	-----

Description

- GENERATE loop over 1/4 towers in phi and both halves in eta and for each site
 - Multiply jet energy by the trigonometric coefficients from the corresponding MHTcoefficient object to get x- and y-components of vector
 - Right-shift the x- and y-components of the vector sum by vectorJetShift bits and truncate the least-significant bits
 - Store as a 4x1 ring-segment object
- GENERATE loop over 1/12 towers in phi and both halves in eta and for each site
 - Sum three non-overlapping neighbouring 4x1 ring-segment objects in phi into a 12x1 ring-segment objects
- GENERATE loop over 1/36 towers in phi and both halves in eta and for each site
 - Sum three non-overlapping neighbouring 12x1 ring-segment objects in phi into a 36x1 ring-segment objects
- GENERATE loop over 1/72 towers in phi and both halves in eta and for each site
 - Sum two non-overlapping neighbouring 36x1 ring-segment objects in phi into a 72x1 ring-segment objects

Algorithm: ET/MET sums

Files

Top-level	projects/calol2/algorithm_components/firmware/hdl/rings/ETandMETsums.vhd
Types	projects/calol2/algorithm_components/firmware/hdl/jets/types_pkg.vhd projects/calol2/algorithm_components/firmware/hdl/rings/types_pkg.vhd
Pipes	projects/calol2/algorithm_components/firmware/hdl/jets/JetPipe.vhd projects/calol2/algorithm_components/firmware/hdl/rings/RingPipe.vhd
Internal entities	projects/calol2/algorithm_components/firmware/hdl/rings/RingSegmentSum.vhd
Utilities	projects/calol2/algorithm_components/firmware/hdl/rings/functions_pkg.vhd

Generics

Pipe lengths	stripPipeInLength vectorStripPipeInLength ringPipeOutLength
Pipe Offsets	stripOffset vectorStripOffset
Other	-

I/O

Inputs	Name	strips9x1PipeIn
	Type	tJetsPipe(stripPipeInLength-1 DOWNT0 0)
	Description	Strips 9 in phi by 1 in eta, organized by eta-half/phi
	Name	vectorStrips9x1PipeIn
	Type	tRingSegmentsPipe(vectorStripPipeInLength-1 DOWNT0 0)
	Description	Ring-segments 9 in phi by 1 in eta, organized by eta-half/phi
Outputs	Name	ringPipeOut
	Type	tRingSegmentsPipe2(ringPipeOutLength-1 DOWNT0 0)
	Description	Scalar and vector sum of tower energy, organized by eta-half

Latency

Clock cycles	7
--------------	---

BX	1 1/6
----	-------

Description

- GENERATE loop over 1/9 towers in phi and both halves in eta and for each site
 - Map the total energy of a region of 9x1 towers in phi from the 9x1 strip object, and the components and towers over threshold from the 9x1 vector-strip objects into a single container object
- GENERATE loop over 1/18 towers in phi and both halves in eta and for each site
 - Sum two non-overlapping neighbouring 9x1 ring-segment objects in phi into an 18x1 ring-segment objects
- GENERATE loop over 1/36 towers in phi and both halves in eta and for each site
 - Sum two non-overlapping neighbouring 18x1 ring-segment objects in phi into a 36x1 ring-segment objects
- GENERATE loop over 1/72 towers in phi and both halves in eta and for each site
 - Sum two non-overlapping neighbouring 36x1 ring-segment objects in phi into a 72x1 ring-segment objects

Algorithm: HT/MHT and ET/MET accumulation

Files

Top-level	projects/calol2/algorithm_components/firmware/hdl/rings/RingAccumulators.vhd
Types	projects/calol2/algorithm_components/firmware/hdl/rings/types_pkg.vhd
Pipes	projects/calol2/algorithm_components/firmware/hdl/rings/RingPipe.vhd
Internal entities	-
Utilities	projects/calol2/algorithm_components/firmware/hdl/rings/functions_pkg.vhd

Generics

Pipe lengths	ringPipeInLength accumulatedRingPipeOutLength
Pipe Offsets	-
Other	scalarSumShift vectorSumShift

I/O

Inputs	Name	ringPipeIn
	Type	tRingSegmentsPipe2(ringPipeOutLength-1 DOWNT0 0)
	Description	Scalar and vector sum of tower energy, organized by eta-half
Outputs	Name	accumulatedRingPipeOut
	Type	tRingSegmentsPipe2(ringPipeOutLength-1 DOWNT0 0)
	Description	Scalar and vector sum of tower energy, organized by eta-half, accumulated over the half-barrel

Latency

Clock cycles	1
BX	1/6

Description

- Reset the accumulation if the incoming data is not valid
- The scalar sum of the incoming ring segment is right-shifted by scalarSumShift bits and the least-significant bits truncated
- The vector-sum components of the incoming ring segment are right-shifted by vectorSumShift bits and the least-significant bits truncated
- Otherwise, if the data is valid add it to existing accumulation and mark as valid

Notes

The truncations make sure that the accumulated sum doesn't overflow the 20-bit signed-type used to store the value and bring us back into the realm of sanity.

Algorithm: Jet Sorting

Files

Top-level	projects/calol2/algorithm_components/firmware/hdl/jets/JetBitonicSort.vhd
Types	projects/calol2/algorithm_components/firmware/hdl/jets/types_pkg.vhd
Pipes	projects/calol2/algorithm_components/firmware/hdl/jets/FilteredJetPipe.vhd
Internal entities	-
Utilities	projects/calol2/algorithm_components/firmware/hdl/jets/functions_pkg.vhd projects/calol2/algorithm_components/firmware/hdl/common/functions_pkg.vhd

Generics

Pipe lengths	filteredJetsPipeInLength sortedJetsPipeOutLength
Pipe Offsets	-
Other	-

I/O

Inputs	Name	filteredJetsPipeIn
	Type	tFilteredJetsPipe(filteredJetsPipeInLength-1 DOWNT0 0)
	Description	Corrected energy sums 9 in phi by 9 in eta + global coordinate, central tower guaranteed to be the highest energy, organized by eta-half/phi
Outputs	Name	sortedJetsPipeOut
	Type	tFilteredJetsPipe(sortedJetsPipeOutLength-1 DOWNT0 0)
	Description	Corrected energy sums 9 in phi by 9 in eta + global coordinate, central tower guaranteed to be the highest energy, organized by eta-half/sort-index

Latency

Clock cycles	14
BX	2 1/3

Description

- A fully-recursive implementation of a truncating, any-size bitonic sort of jet candidates. For more information on bitonic sorting algorithms, see
 - http://en.wikipedia.org/wiki/Bitonic_sorter
 - <http://www.iti.fh-flensburg.de/lang/algorithmen/sortieren/bitonic/oddn.htm>

Notes

VHDL (or at least the ISE VHDL parser) does not like recursive constructs; they have to be carefully constructed to work at all.

In particular “xxx : entity work.yyy” type instantiations do not work, nor do COMPONENT declarations at global scope: COMPONENT declarations must be made in the ARCHITECTURE preamble.

I have modified the standard bitonic sort so that an output size is specified. Once an entry is guaranteed to fail to make it into the output list, it is discarded so as to save resources and latency. Setting the output size to the input size keeps all candidates.

A non-power-of-two bitonic sort has different sized sorts on each branch. Different sized sorts have different latencies, and so each branch must be latency compensated. The latency compensation is automatically calculated at compile-time in this implementation, the latency itself being calculated by a recursive function call on the GENERIC parameters.

Algorithm: Accumulating Jet Sorting

Files

Top-level	projects/calol2/algorithm_components/firmware/hdl/jets/JetBitonicSort.vhd
Types	projects/calol2/algorithm_components/firmware/hdl/jets/types_pkg.vhd
Pipes	projects/calol2/algorithm_components/firmware/hdl/jets/FilteredJetPipe.vhd
Internal entities	-
Utilities	projects/calol2/algorithm_components/firmware/hdl/jets/functions_pkg.vhd projects/calol2/algorithm_components/firmware/hdl/common/functions_pkg.vhd

Generics

Pipe lengths	sortedJetsPipeInLength accumulatedSortedJetsPipeOutLength
Pipe Offsets	-
Other	-

I/O

Inputs	Name	sortedJetsPipeIn
	Type	tFilteredJetsPipe(filteredJetsPipeInLength-1 DOWNT0 0)
	Description	Corrected energy sums 9 in phi by 9 in eta + global coordinate, central tower guaranteed to be the highest energy, organized by eta-half/sort-index
Outputs	Name	accumulatedSortedJetsPipeOut
	Type	tFilteredJetsPipe(sortedJetsPipeOutLength-1 DOWNT0 0)
	Description	Corrected energy sums 9 in phi by 9 in eta + global coordinate, central tower guaranteed to be the highest energy, organized by eta-half/sort-index, accumulated over the half-barrel

Latency

Clock cycles	18
BX	3

Description

- An accumulating sort which takes a sorted list and returns a sorted list integrated over time.
- The generic bitonic sort functionality is reused within this module.
- Because sorting has a finite latency, accumulation is complicated. The implementation used here is an interleaved accumulating sort, where the outputs are fed into a pipeline and the interleaved sorts then merged into a final sorted list. This is shown in Figure 4.

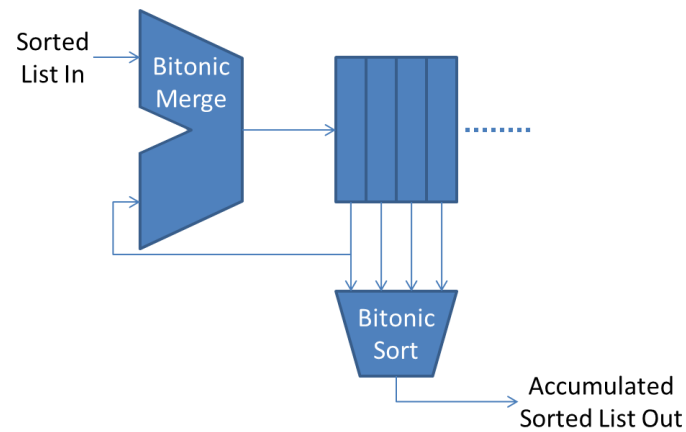


Figure 4 : The architecture of the accumulating bitonic sort showing the mechanism used to compensate for the latency of merging two sorted lists

Notes

VHDL (or at least the ISE VHDL parser) does not like recursive constructs; they have to be carefully constructed to work at all.

In particular “xxx : entity work.yyy” type instantiations do not work, nor do COMPONENT declarations at global scope: COMPONENT declarations must be made in the ARCHITECTURE preamble.

I have modified the standard bitonic sort so that an output size is specified. Once an entry is guaranteed to fail to make it into the output list, it is discarded so as to save resources and latency. Setting the output size to the input size keeps all candidates.

A non-power-of-two bitonic sort has different sized sorts on each branch. Different sized sorts have different latencies, and so each branch must be latency compensated. The latency compensation is automatically calculated at compile-time in this implementation, the latency itself being calculated by a recursive function call on the GENERIC parameters.

The period of interleaving and thus size of the final sort depends on the latency of the initial sort, which in-turn depends on the size of the initial sort. All sizes and latencies are automatically calculated and invisible to the user.

Algorithm: Coordinate conversion

Files

Top-level	projects/calol2/algorithm_components/firmware/hdl/rings/CoordinateConversion.vhd
Types	projects/calol2/algorithm_components/firmware/hdl/rings/types_pkg.vhd
Pipes	projects/calol2/algorithm_components/firmware/hdl/rings/RingPipe.vhd projects/calol2/algorithm_components/firmware/hdl/rings/PolarRingPipe.vhd
Internal entities	-
Utilities	projects/calol2/algorithm_components/firmware/hdl/rings/functions_pkg.vhd

Generics

Pipe lengths	ringPipeInLength polarRingPipeOutLength
Pipe Offsets	-
Other	-

I/O

Inputs	Name	ringsPipeIn
	Type	tFilteredRingsPipe(ringsPipeInLength-1 DOWNT0 0)
	Description	Scalar and vector sum of tower energy, organized by eta-half, accumulated over the half-barrel
Outputs	Name	polarRingPipeOut
	Type	tPolarRingSegmentPipe(polarRingPipeOutLength-1 DOWNT0 0)
	Description	Scalar and vector sum of tower energy, organized by eta-half, accumulated over the half-barrel, in polar coordinates

Latency

Clock cycles	?
BX	?

Description

- An initial angle and a sign bit are set according to the quadrant in which the Cartesian vector components lie. The sign bit specifies the direction of the rotation from the initial angle.
- The magnitude of the vector components is taken so that the coordinate rotation is performed in the primary quadrant.
- Eight, pipelined, coordinate-rotation steps are applied, each updating the size of the Cartesian components and the angle. The y-component of the Cartesian coordinates is used as the control-variable and the x-component is, therefore, the component being updated for the magnitude. The constants needed for the rotation are calculated at compile-time.
- The magnitude (x-component) is multiplied by a constant value (which depends on the number of rotation steps and a scaling factor) and then right-shifted by the scaling factor. The value of the constant is calculated at compile-time.

Notes

This CORDIC algorithm is based on python code (albeit modified to handle the four-quadrant input) available here:

<http://code.activestate.com/recipes/576792-polar-to-rectangular-conversions-using-cordic/>

CORDICs are magic:

<http://en.wikipedia.org/wiki/CORDIC>