

Update on TAB Progress

John Parsons

Nevis Labs, Columbia University

Feb. 15/2002

- Assumptions about ADC/FIR board
- ADC to TAB data links
- Progress on Trigger Algorithm Board (TAB)
- Urgent issues to be resolved
- Summary and conclusions

Introduction

- to allow a more thorough evaluation, have made certain assumptions to define a “strawman” architecture:
 - ◆ ADC+FIR
 - ▲ 32 channels/board
 - ▲ ≈ 80 ADC boards
 - ▲ I/P cable mapping groups neighboring eta, phi towers
 - ◆ fast copper ADC-TAB links
 - ◆ Trigger Algorithm Board (TAB)
 - ▲ assume processing 1 TT requires 5×5 towers
 - ▲ 1 TAB processes 4 eta X 32 phi $\Rightarrow$ 10 TABs
- effort has concentrated so far on TAB and implementation of sliding window algorithm (plus interface to ADC board)
 - ◆ tried to evaluate with flexibility wrt assumptions, and to identify where choices need to be made soon

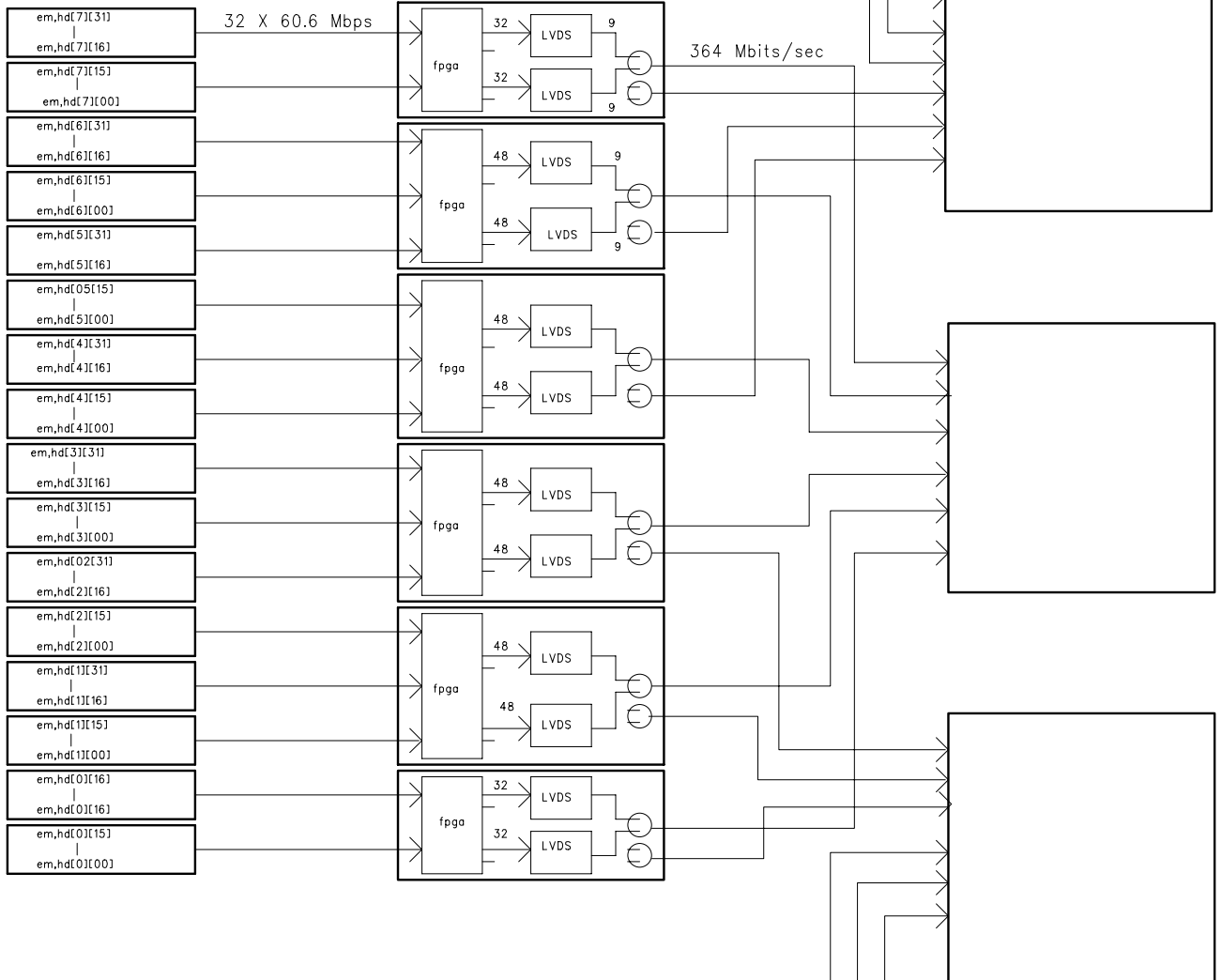
System Overview

System Interconnects

ADC-FIR BOARDS
(16 of 80 shown)

DATA CONCENTRATOR BOARDS

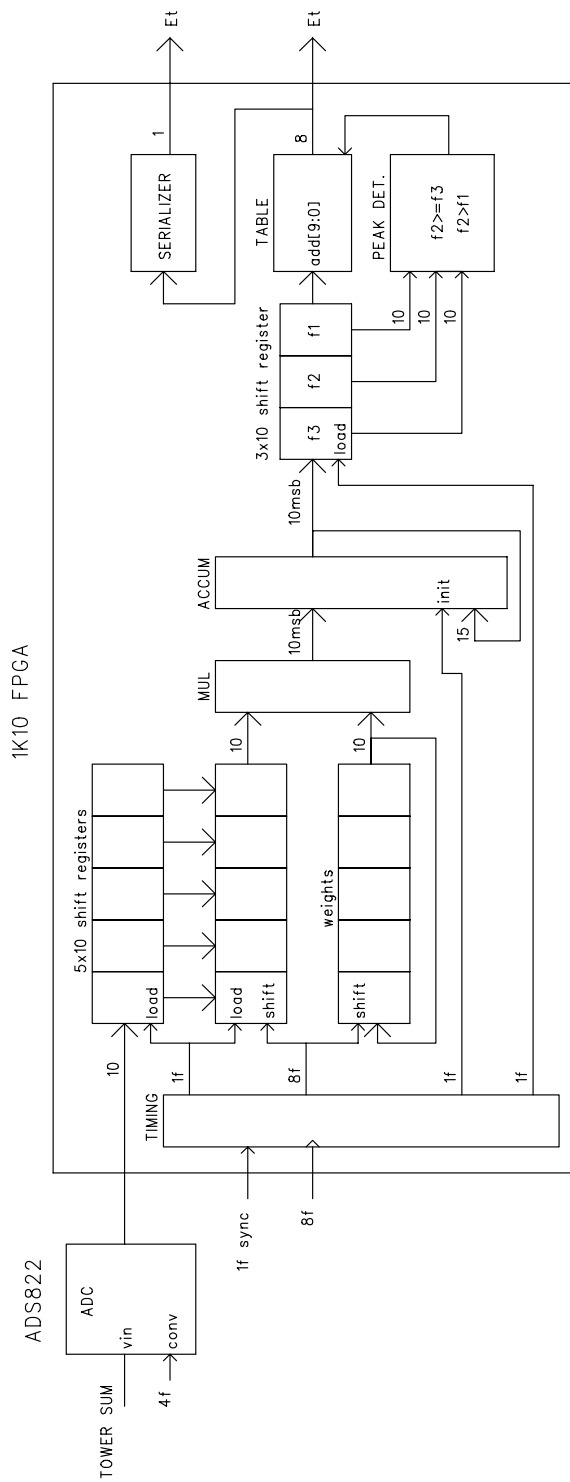
TRIGGER ALGORITHM BOARDS
(3 of 10 shown)



ADC-FIR Board (1)

- assume 32 channels/board
 - ◆ I/P cable mapping groups eta, phi neighbors
- digitize with 10 bit ADC, at multiple of bc frequency of $f = 1/132 \text{ ns} \approx 7.6 \text{ MHz}$
 - ◆ reduce ADC latency
 - ◆ allow over-sampling in FIR (if required)
 - ◆ candidate device is **Burr Brown ADS822**
 - ▲ 10-bit, 40 MHz CMOS pipelined ADC
 - ▲ power is 190 mW @ 40 MHz
 - ▲ operate at $4f = 30.3 \text{ MHz}$
 - ▲ pipeline delay = 5 CLKs
 - ▲ for even lower latency, could use pin compatible 60 MHz ADS823 (\$8) or 70 MHz ADS824 (\$9)
 - ▲ Unit cost $\approx \$5$
- FPGA to apply FIR, conversion to 8-bit E_T , serialization of output data at $8f = 60.6 \text{ MHz}$
 - ◆ candidate device is **Altera EP1K10TC100-2**
 - ▲ FIR logic clocked at $8f = 60.6 \text{ MHz}$
 - ▲ Example with 5 samples:
 - utilization $\approx 84\%$ (logic), 16% (memory)
 - max. speed $\approx 67 \text{ MHz}$
 - ▲ Unit cost $\approx \$10$ (\$15 if use grade -1)

ADC-FIR Board (2)



ADC to TAB Links

- use high bandwidth LVDS serial links to keep cable plant manageable
 - ◆ eg. “Channel Link” chipset from National
 - ▲ 48:8 Serializer/Tx (DS90CR483)
 - ▲ 8:48 Rx/Deserializer (DS90CR484)
 - ▲ Unit cost = \$11 each (though for 1k quantity)
 - ◆ send 8 data bits on cable at rate of
 $7.6 \text{ MHz} \times 8 \text{ bits} \times 6 = 364 \text{ MHz}$
 - ◆ CLK sent on additional pair $\Rightarrow$ 9 pairs in total
 - ◆ chipset is rated up to $112 \times 6 = 672 \text{ MHz}$
(ATLAS L1 has demonstrated 480 MHz over 20m cables)
- two problems with indiv. cable per ADC board:
 - ◆ inefficient, since use only 32 of 48 data lines
 - ◆ each TAB (512 inputs) would require 16 cables, which take too much space to fit on (single width) 9U module
- to resolve these problems, consider merging data from several ADC boards into a Data Concentrator, which then drives the cable

Data Concentrator

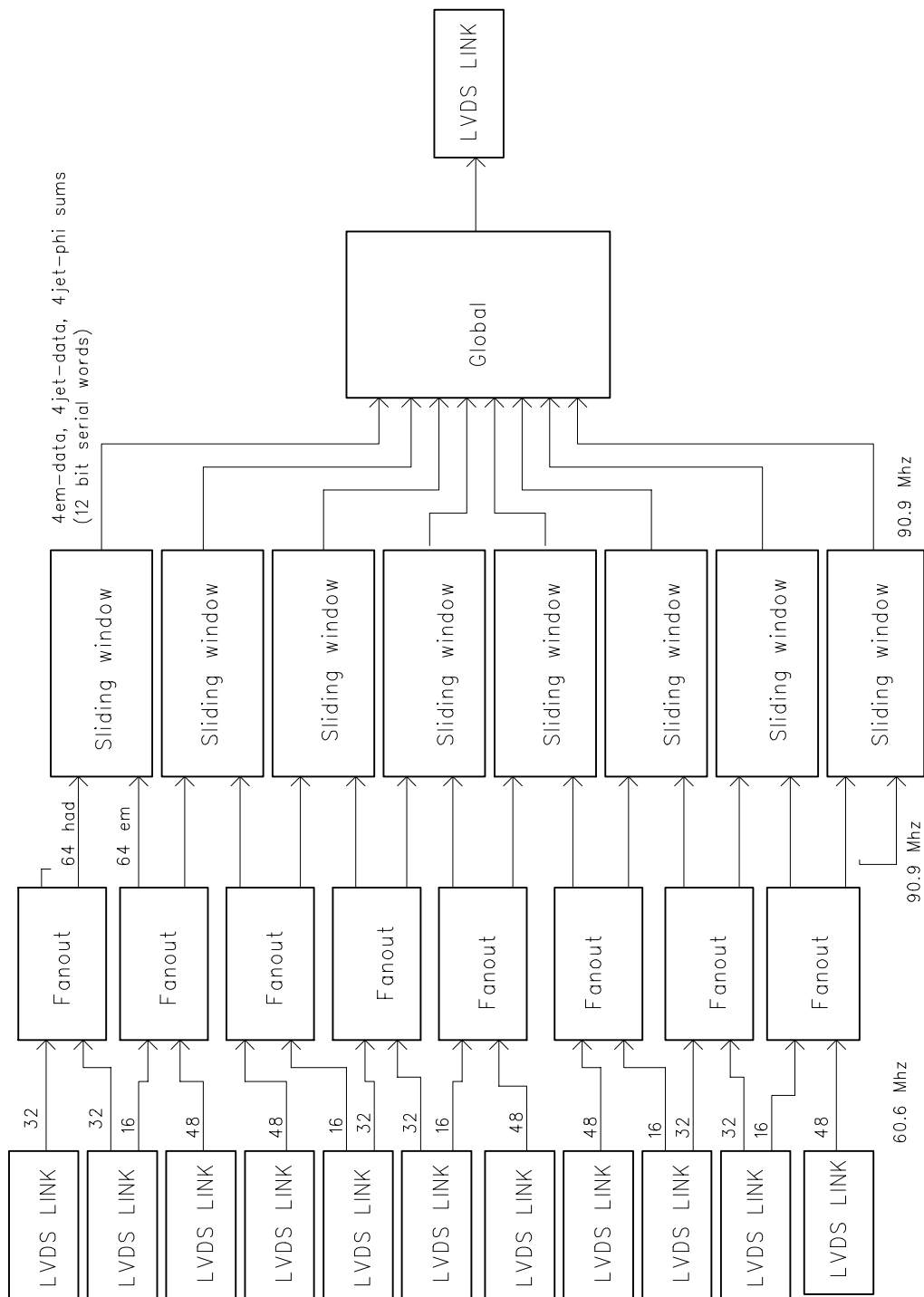
- several cable configurations can be considered
- one such possibility is:
 - ◆ collect data from 3 ADC boards (32 signals each at 60.6 MHz), for example over custom point-to-point P3 backplane
 - ◆ Data Concentrator re-synchs & merges the 3 data streams into 2 LVDS serialisers, and drives the resultant 16 data and 2 CLK signals over a 25-pair cable (extra pairs can be used for control fields)
 - ◆ each TAB (512 inputs) would require 6 such cables, which can fit on 9U VME front panel
- Also, due to overlap in sliding window, most TTs are needed on two separate TAB boards
- because of very high signal density in TAB crate, we propose performing this “fanout” at Data Concentrator (even though it doubles the number of cables)
- cable density at I/P to TAB is challenging, and ADC-TAB cabling scheme must be addressed with priority to allow design to continue

Trigger Algorithm Board (TAB)

- aim to cover 4 eta X 32 phi in single TAB
 - ◆ 10 TAB boards in total system
- assuming 5X5 towers required to evaluate a given TT, number of input signals per TAB is
inputs = 8 eta X 32 phi X 2 (EM,HAD) = 512
- basic architecture (see next slide)
 - ◆ LVDS Rx/Deserialisers
 - ◆ “Fanout” FPGAs
 - ◆ “Sliding Window” FPGAs
 - ▲ apply sliding window algo.’s for EM and jet objects
 - ▲ perform partial E_T sums
 - ◆ “Global” FPGA(s)
 - ▲ summarize window results
 - ▲ perform partial E_T , E_T^x and E_T^y sums

TAB Architecture

4eta-32phi TAB



“Fanout” FPGAs

- each chip has:
 - ◆ 64 serial input streams at $8f = 60.6$ MHz
 - ◆ 128 serial output streams at $12f = 90.9$ MHz
- functionality required:
 - ◆ align all signals in time
 - ◆ pad 8-bit TT E_T 's with zeroes to 12 bits
 - ▲ allows more dynamic range in summing trees
 - ◆ switch serial transmission frequency from 60.6 MHz to 90.9 MHz
 - ▲ costs 1 b.c. latency

(might do all 3 above in Window FPGA instead)

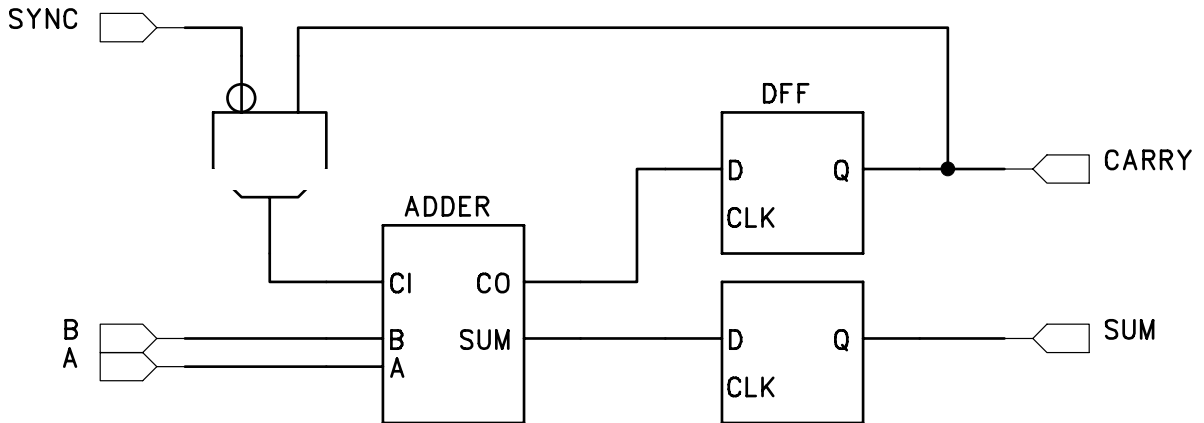
- ◆ perform two-fold fanout of signals
 - ▲ required by window overlaps
 - ◆ allow VME loading of test data for TAB standalone diagnostics
- candidate device = Altera EP1K50FC484-3
 - ◆ Unit cost = \$33

“Sliding Window” FPGAs

- aim to cover 4 eta X 4 phi in single FPGA
 - ◆ 8 Sliding Window FPGAs per TAB
- assuming 5X5 towers required to evaluate a given TT, number of input signals per FPGA is
inputs = 8 eta X 8 phi X 2 (EM,HAD) = 128
- to minimize data duplication and routing, perform both EM and jet algorithms in the same FPGA
 - ◆ with these assumptions, Fanout FPGA must provide X2 fanout only
- basic FPGA design philosophy
 - ◆ operate algorithms bit-serially in order to minimize FPGA resources required
 - ◆ operate logic at 12f = 90.9 MHz and fully pipeline in order to maintain low latency

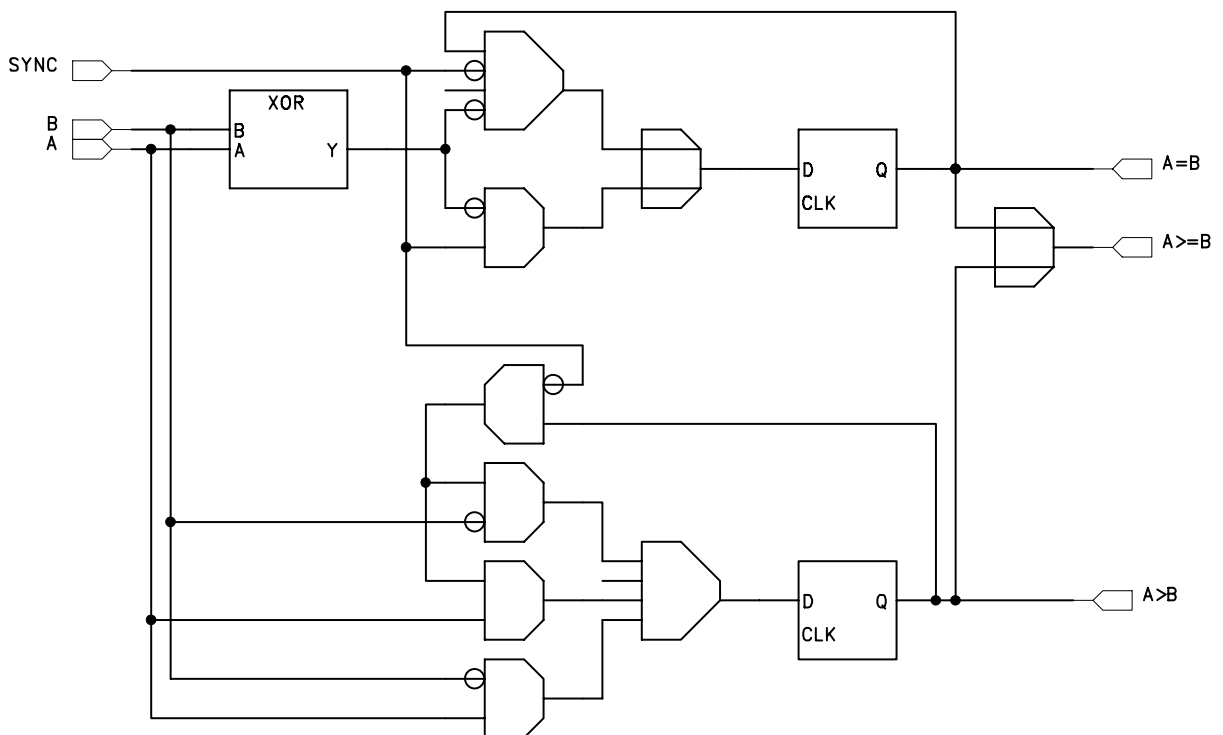
Example bit-serial operators

Serial adder



- SYNC is signal which separates one 12-bit serial word (ie. data from one b.c.) from the next

Serial comparator



```
#inputs=2np           =128
#2x2 rois=(n-2)(p-2) =36
#outputs=(n-4)(p-4)   =16
```

em and had towers

phi	0 7	1 7	2 7	3 7	4 7	5 7	6 7	7 7
	0 6	1 6	2 6	3 6	4 6	5 6	6 6	7 6
	0 5	1 5	2 5	3 5	4 5	5 5	6 5	7 5
	0 4	1 4	2 4	3 4	4 4	5 4	6 4	7 4
	0 3	1 3	2 3	3 3	4 3	5 3	6 3	7 3
	0 2	1 2	2 2	3 2	4 2	5 2	6 2	7 2
	0 1	1 1	2 1	3 1	4 1	5 1	6 1	7 1
	0 0	1 0	2 0	3 0	4 0	5 0	6 0	7 0

eta

36 roi sums

A 10x10 grid with two highlighted 2x2 squares. The top-right square contains the numbers 6, 7, 7, 6. The bottom-left square contains the numbers 1, 2, 2, 1.

16 isolation sums

remapped roi energys

0 5	1 5	2 5	3 5	4 5	5 5
0 4	1 4	2 4	3 4	4 4	5 4
0 3	1 3	2 3	3 3	4 3	5 3
0 2	1 2	2 2	3 2	4 2	5 2
0 1	1 1	2 1	3 1	4 1	5 1
0 0	1 0	2 0	3 0	4 0	5 0

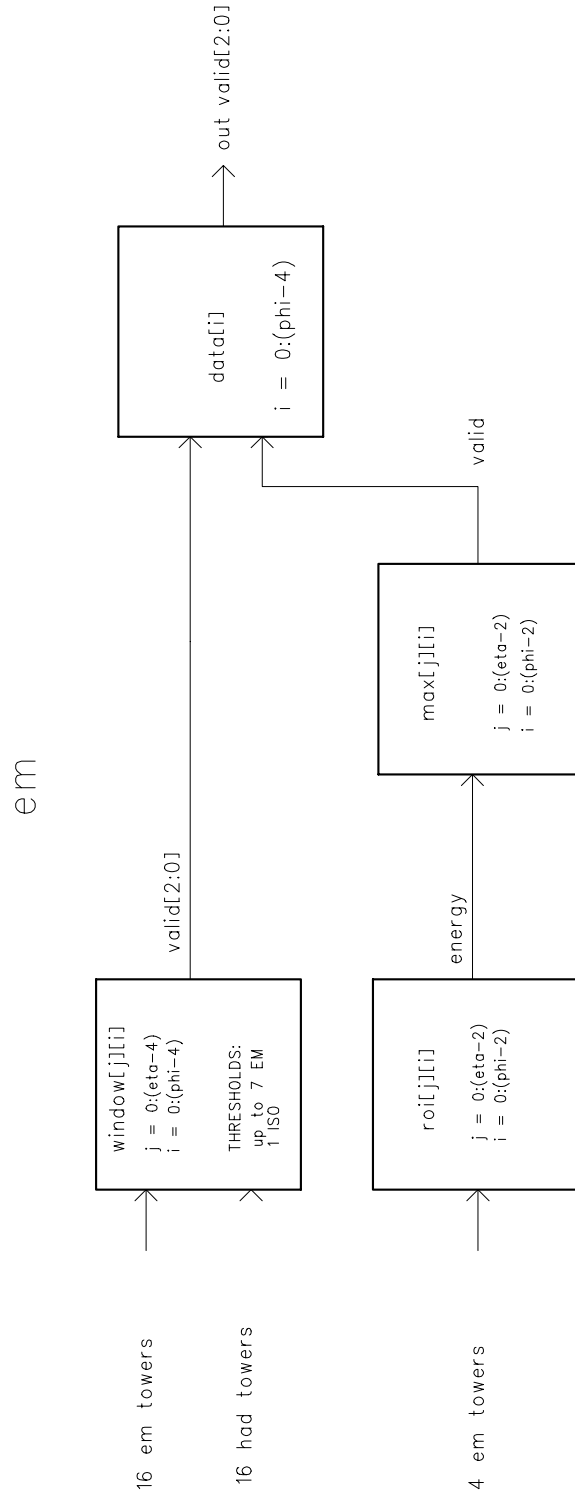
test energy for max

				3 5	4 5	5 5
				3 4	4 4	5 4
				3 3	4 3	5 3
0 2	1 2	2 2				
0 1	1 1	2 1				
0 0	1 0	2 0				

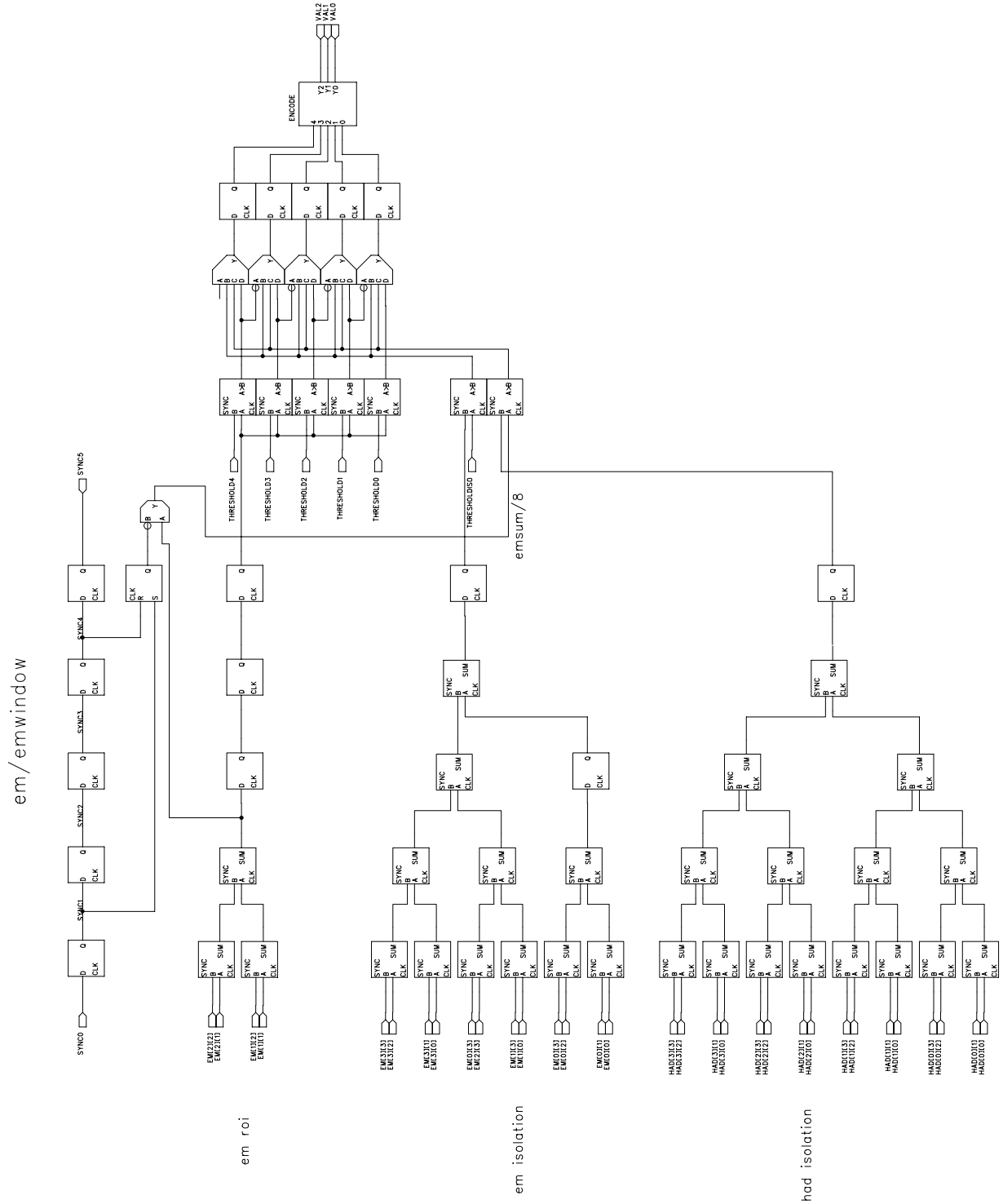
output energy

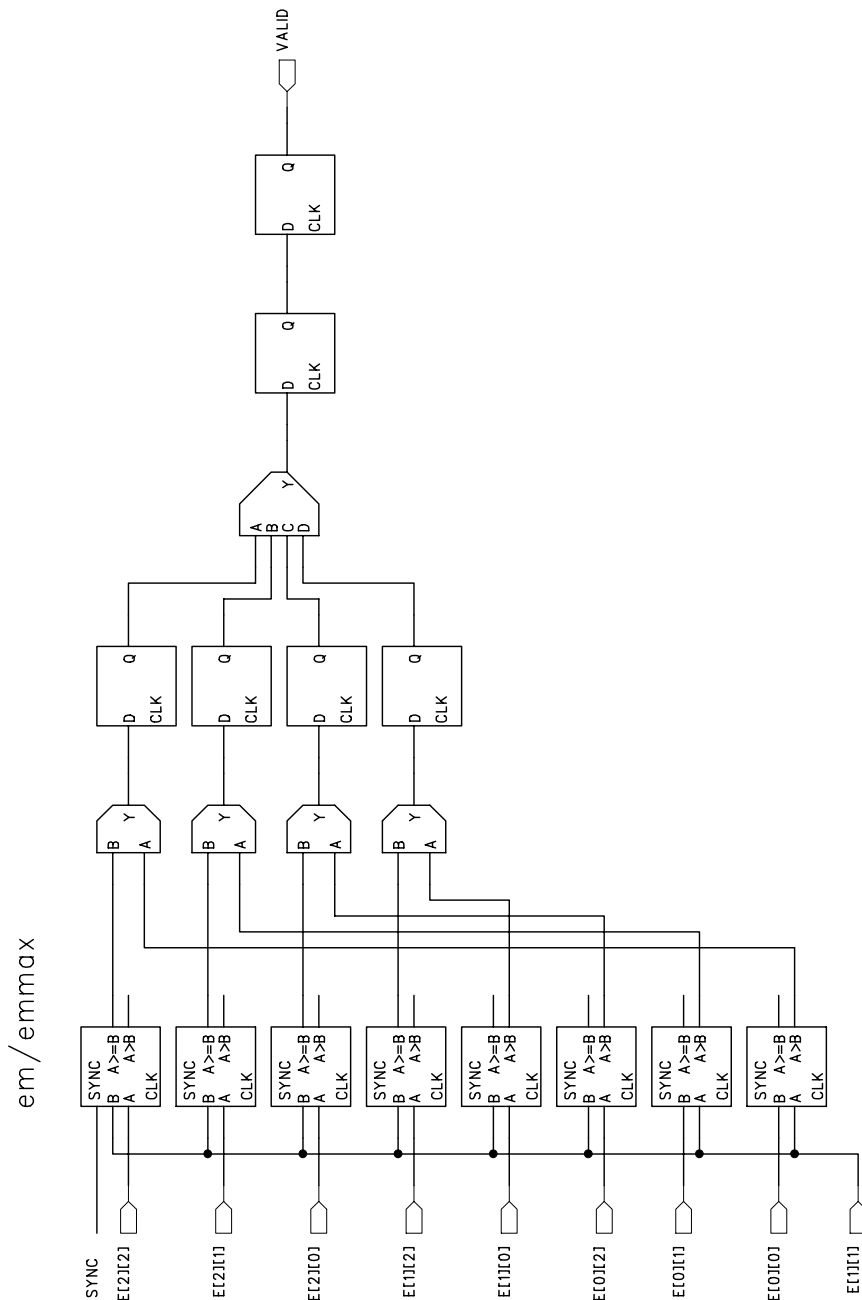
0 3	1 3	3 3	3 3
0 2	1 2	2 2	3 2
0 1	1 1	2 1	3 1
0 0	1 0	2 0	3 0

Overview of EM Algorithm



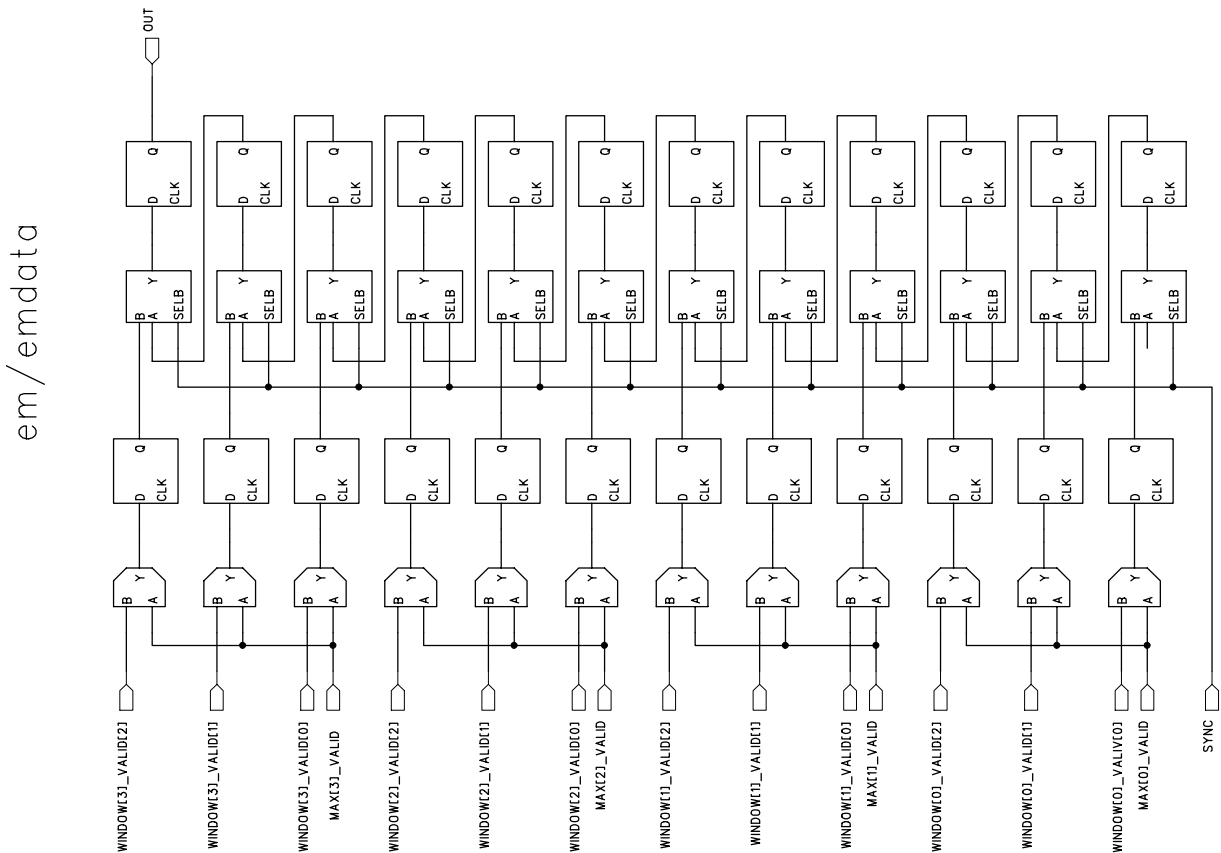
EM Window Schematic





“EM Data” Schematic

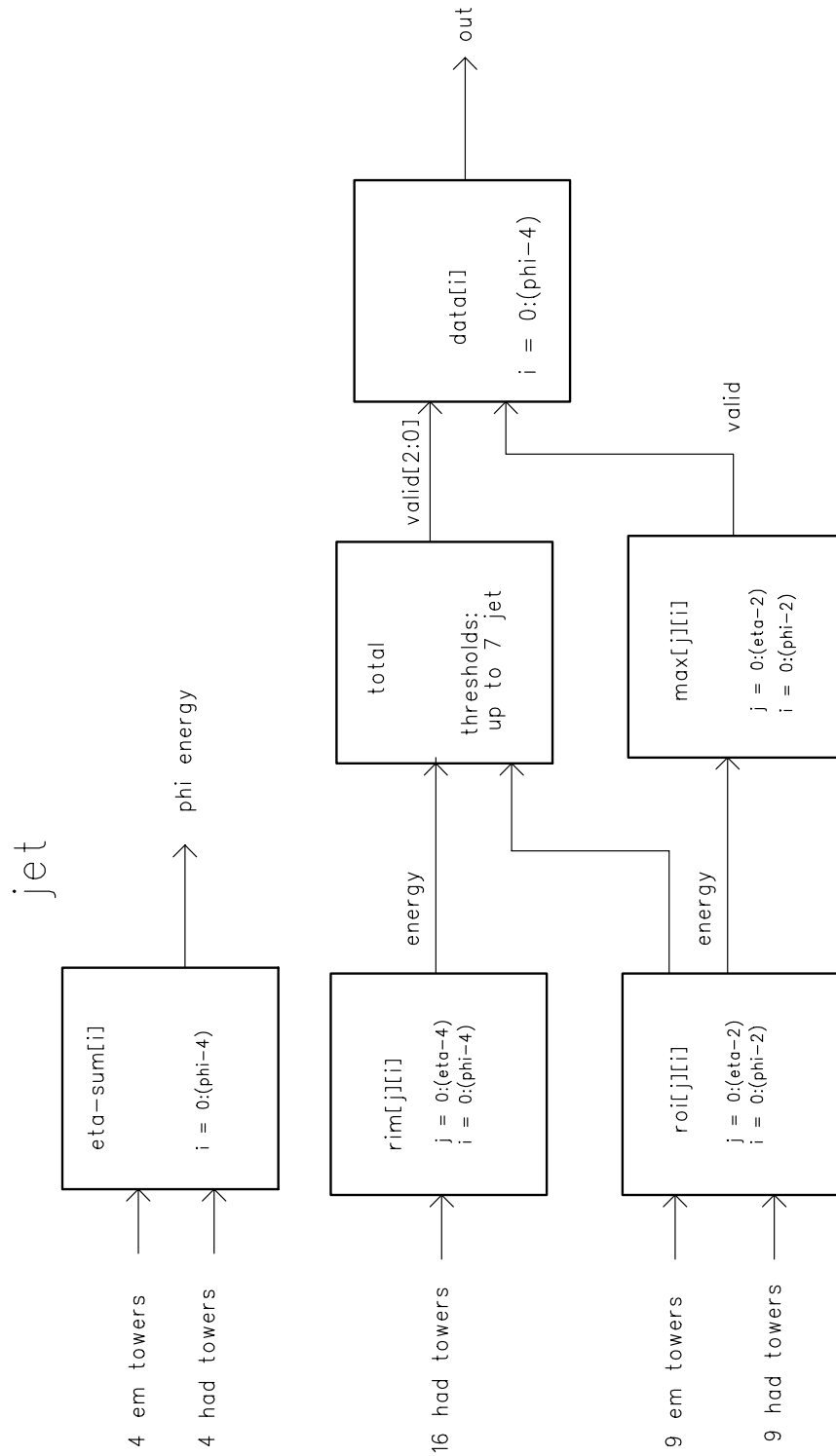
- condition threshold bits with local max. VALID
- merge 3-bit threshold data from 4 TT's and serialize output into one 12-bit serial stream
 - ◆ serialization costs 1 b.c. latency



- each FPGA handles $4 \times 4 = 16$ TTs
 - ◆ EM algorithm output is 4 12-bit serial words, encoding highest threshold passed by possible isolated EM objects in each TT

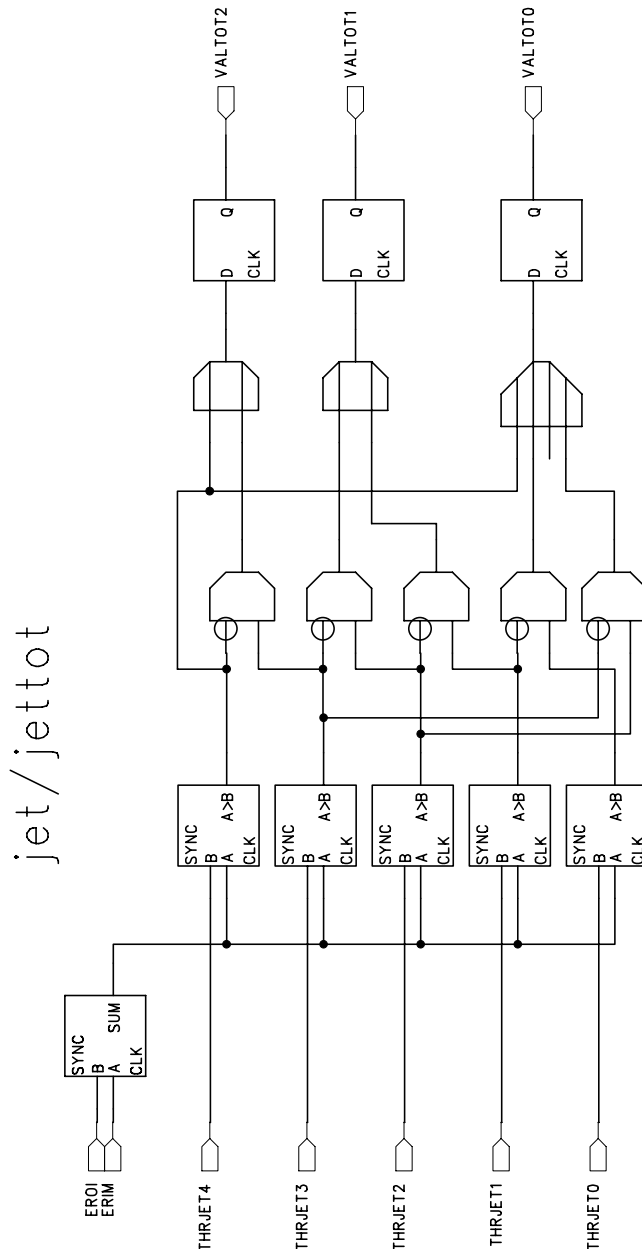
total energy			
0 3	1 3	3 3	3 3
0 2	1 2	2 2	3 2
0 1	1 1	2 1	3 1
0 0	1 0	2 0	3 0

Overview of Jet Algorithm



“Jet Total” Schematic

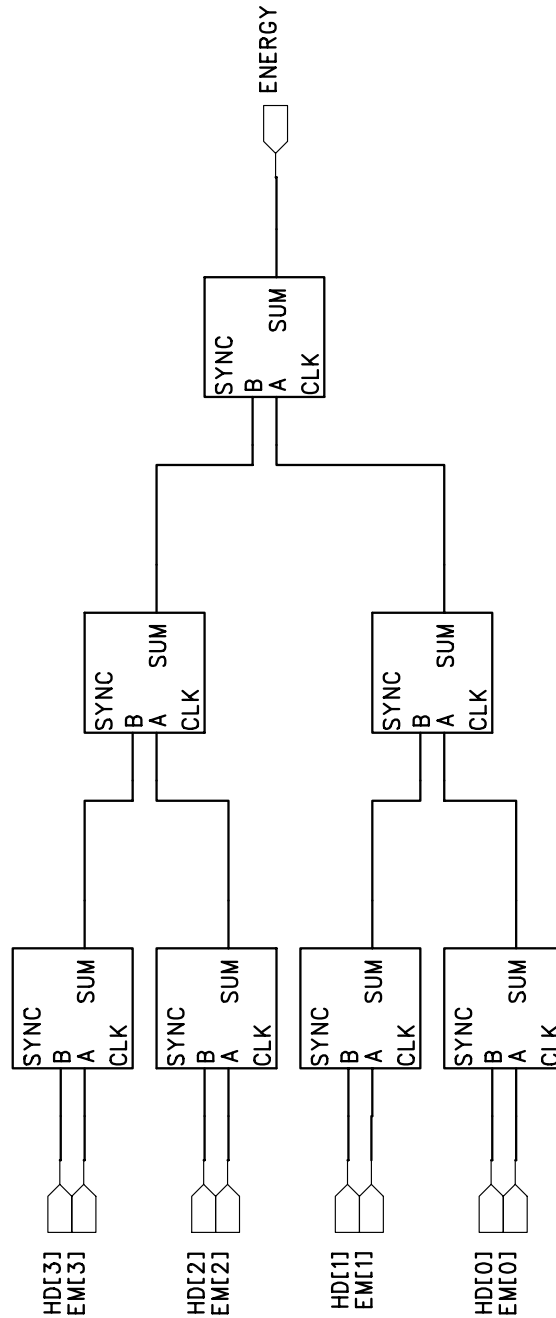
- combine 3X3 ROI and “rim” to get E_T in 5X5
- compare against up to 7 thresholds, and encode highest threshold passed onto 3 bits



“Jet eta sum” Schematic

- for input to E_T and E_T^{miss} , compute partial 12-bit E_T sums over eta at fixed phi

jet/etasum



Sliding Window Implementation

- logic, as described, has been coded and simulated
- with 4X4 TT's/FPGA, and 5X5 TT's needed to evaluate any TT, candidates include:
 - ◆ EP1K100FC256-3 (unit cost = \$46)
 - ▲ BUT LC utilization = 91% ⇒ VERY LITTLE flexibility
 - ◆ EP20K160EQC240-3 (unit cost = \$94)
 - ▲ Utilization: LCells = 71%, Mem = 0%
 - ▲ Max. speed = 133 MHz
 - ◆ EP20K200EBC356-3 (unit cost = \$130)
 - ▲ LCell utilization = 55%
- code structured to allow quick check of impact of changing assumptions
 - ◆ eg. What if need 7X7 to evaluate any TT??
 - ▲ # inputs increases from 128 to 200
 - ▲ # Lcells required increases by 33%
 - ⇒ 20K200 with 73% utilization and 120 MHz max. speed
 - ⇒ most difficult issue with 7X7 arises not from FPGA considerations, but from cabling to TAB (each TAB then requires 640 inputs)

“Global” FPGA

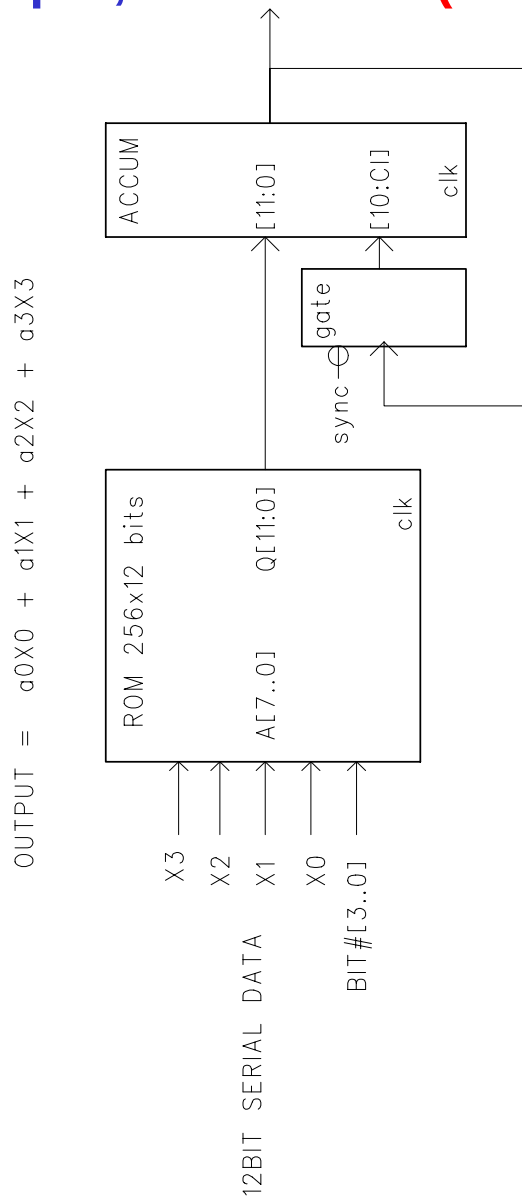
- from each of 8 Sliding Window FPGAs, receive:
 - ◆ 4 12-bit streams of encoded EM data
 - ◆ 4 12-bit streams of encoded jet data
 - ◆ 4 12-bit E_T sums over eta at fixed phi

⇒ total of $8 \times 12 = 96$ 12-bit serial inputs
- for entire TAB, calculate and serially output 12-bit results for $\sum E_T$, $\sum E_T^x$, $\sum E_T^y$
 - ◆ apply x,y weights bit-serially using LUT stored in ROM (see next slide)
- “summarize” EM, jet data to reduce output data volume
 - ◆ eg. count number of EM/jet objects above each of the corresponding thresholds (?)

(need to detail what information is needed at L1 and L2, and for the L1 track match logic)
- candidate device = EP20K160EQC240-1
 - ◆ -1 speed grade probably needed (due to Accumulator, which is not bit serial)
 - ◆ Unit cost = \$264
 - ◆ LUTs utilize $\approx 60\%$ of available 81k memory bits

$E_T^{x,y}$ calculations

- results of single-bit weighted sums pre-computed and stored in LUT in FPGA ROM
- Accumulator (with shift) sums single bit results
- before output, re-serialize (**costs 1 b.c.**)



THE ROM STORES ALL POSSIBLE SUMS OF A0 TO A3 FOR THE 12 SERIAL BITS.
THE 12BIT ACCUM OUTPUT IS FEDBACK TO THE INPUT, SHIFTED 1 BIT POSITION.

TAB Latency Considerations

- **Fanout FPGA**
 - ◆ 1 b.c. for changing serialization frequency
- **Sliding Window FPGA**
 - ◆ pipelined logic involves a total of ≈ 10 stages, each of $132/12 = 11$ ns $\Rightarrow < 1$ b.c.
 - ◆ 1 b.c. for serializing output streams
- **Global FPGA**
 - ◆ 1 b.c. for $E_T^{x,y}$ calculations
 - ◆ 1 b.c. for serializing output streams
- **Total TAB latency ≈ 5 b.c. = 660 ns**
(expect comparable number from ADC/FIR)
 - ◆ can provide lot of time for track match logic
 - ◆ Global CAL L1 board will presumably have to store CAL L1 information before transmission to Framework, in order to wait for other detectors

Global L1CAL Board

- one Global L1CAL board for entire system
- from each of 10 TABs, receives:
 - ◆ 12-bit E_T , E_T^x , E_T^y sums
 - ◆ “summarized” EM/jet data
- calculate E_T^{miss}
 - ◆ finishes summing (takes $4 \times 11 \text{ ns} = 44 \text{ ns}$)
 - ◆ use multipliers to calculate $(E_T^{\text{miss}})^2$
- FPGAs used to determine (and store until the correct time) the ‘AND/OR’ terms for transmission to the L1 Framework
- while no detailed design work has yet been done, it is clear this board is less technically challenging than the TAB

Urgent Issues

- to proceed much further with TAB design, some issues need to be resolved:
 - ◆ size of region required/TT (ie. 5X5 or 7X7)
 - ▲ # inputs/TAB is either 512 or 640
 - ▲ # inputs/Window FPGA is either 128 or 200
 - ▲ data fanout is either 2 or (in some cases) 3
 - ▲ ADC-TAB cabling looks very different
 - ▲ these are two VERY different scenarios, and we must choose one SOON in order to proceed (my view: given significant increase in cost and complexity, choice of 7X7 should require strong physics case)
 - ◆ interfaces to track match, L1, L2
 - ▲ see next slide
 - ◆ details of trigger algorithm
 - ▲ less critical now, since FPGAs provide a lot of flexibility (provided we allow some “headroom”)
 - ▲ However, if we foresee LARGE additions/changes to the algo. (eg. addition of τ trigger), need to take into account in choice of FPGA sizes
- [Comment: it would appear to be possible to add a τ trigger without a large impact on complexity/cost.]

Interfaces

- so far, have concentrated on implementation of Sliding Window algorithm
- need to start “folding in” interface requirements
 - ◆ L1 CAL-track match
 - ▲ what summary of EM info. is required, and with what granularity?
 - ▲ could come from Window FPGAs directly, from Global FPGA, or from Global CAL board
 - ◆ L1 trigger framework
 - ▲ look at generation/timing of And/Or terms
 - ◆ L2
 - ▲ what information is required?
 - ▲ eg. if E_T needed for each TT, could be stored using on-chip memory in Window FPGAs
 - ◆ SCL
 - ▲ CLK, L1Accept
- while use of FPGAs for algorithms provides a lot of flexibility, issues such as “which cables are interconnecting which boards” need to be frozen early in design phase
 - ◆ need to proceed soon with interface definition

Summary and Conclusions

- we have investigated a TAB architecture to implement the Sliding Window algorithms for iso. EM and jet objects for 4 eta X 32 phi TT's
 - ◆ 4 X 4 TTs can be processed in 20K160 (\$94/chip)
 - ▲ 20K200 (\$130/chip) might be preferable if want to be able to make large change, such as adding τ trigger
 - ◆ total TAB latency ≈ 5 b.c. (660 ns)
- proceeding much further with TAB design requires making some decisions
 - ◆ 5X5 vs 7X7 area required around each TT
 - ◆ def'n of ADC-Concentrator-TAB cabling scheme
 - ◆ Def'n of interfaces of trk match, L1, L2, etc.