Station 1,2,3 chunks are written to this fifo. If write\_en isn’t working, problem upstream (means data word is already improperly formed). Could read\_en problem cause the 9-chip chunks of stations 1,2,3 loss?

Station 0 bypasses this block so any problem here cannot account for lost station 0 wedges.

Upon LATCH, determine what phase of clock to use for FPHX words

Determine if data lines should be blocked (SHIFT)

This seems ruled out because (1) some wedges are lost in middle of data taking (not LATCH then) (2) shift\_in, shift\_en are only enabled for FPGAs A and C

Write is bit\_0, bit\_16 of data word.

read\_en is always ‘1’

Write is bit\_0, bit\_16 of data word.

read\_en derived from EMPTYs (parallel\_arbiter)

Since read\_en is tied to ‘1’ can only be problem here if somehow the data word is garbled and it doesn’t find bit\_0, bit\_16 set to ‘1’ (?). But that would mean a problem up-stream

Passively or’s data. This is where chips are first grouped in chunks of 8, 9 (stations 1,2,3) or 10 (station 0). Doesn’t seem like this could have a problem since it is a passive piece of code unless somehow the pieces that are being or’ed are colliding with each other (?).

write\_en from decoder array round-robin

Deserialize FPHX data

Output data according to write\_en from decoder\_array round-robin. Can this somehow get out of sync?

latch

shift\_in, shift\_en

output\_fifo\_block

Station data or’ed together (stations 1-3 have 3 groups of data)

fifo\_block

output\_arbiter

or\_array

deser\_array

pf\_array

roc\_block