LWA-SV Memo 1

LWA-SV F-engine firmware overview

D. Price
22nd February 2017

Introduction

The LWA-SV digital signal processing system is known as the Advanced Data Processor (ADP). It consists of:

- 32x CASPER ADC16x250-8 digitizer cards (a total of 32x16=512 inputs).
- 16x CASPER ROACH2 FPGA processing boards (Xilinx Virtex-6 SX475T FPGA)
- Mellanox SX1024 10/40GbE switch
- 6x GPU servers (ASUS ESC4000 G3 server)
- Clock synthesizer (Valon 5008), PPS, 16-way splitters and lots of cables.

The digitization, channelization, and channel selection are done on the ROACH2 boards, the firmware of which is written using the CASPER / MATLAB / Simulink / Xilinx ISE toolflow[1].

This memo gives an overview of LWA-SV firmware, by providing a walkthrough of the firmware's Simulink diagram. It is intended that this document is read while clicking through the Simulink diagram.

Firmware overview

The LWA-SV digital frontend digitizes, channelizes, selects a target band, and sends them it over packetized 10 Gb Ethernet to the GPU servers. Identical firmware runs on all 16 ROACH2 boards

[1] https://casper.berkeley.edu/wiki/Main_Page

in parallel, with each board digitizing 32 inputs (for a total of 512). Data output from each board can be identified by setting an ID register, which appears in the packet headers.

Digitization

Digitization of the 32 inputs is conducted using dual ADC16-250-8 cards, which use the Hitite HMCAD1511 ADC chip, running at clock speed 205 Msample/s at 8-bit.

Coarse Delay

After digitization, coarse delay of the input signals is conducted. The coarse delay allows integer delay of ADC inputs, from values of 4-1024 cycles. The purpose of this delay is to account for cable delays and ADC calibration, which can misalign boards by +/- 1 clock cycle.

Channelization

Channelization is performed via a 8192-point FFT, either with or without a polyphase filterbank FIR frontend. The polyphase filterbank (PFB) is a 4-tap, 8192-branch, Hamming-windowed frontend that can be bypassed if a raw FFT is desired.

Requantization to 4 bits

After channelization, data are requantized to 4 bits (4b real, 4b imag). Bit selection is done via multiplication with a scaling coefficient (entered via a shared BRAM) slicing the bottom bits, then rounding to 4 bits.

Packetization

Finally, channel selection and packetization is performed. Channel selection is done by setting a ‘start channel’ and ‘stop channel’ register. Any channels within this range are written to a packet buffer. The packetizer then generates packet headers, and reads the selected channels into each packet’s data payload.

The size of packets is configured by setting several registers. A maximum of ~¼ of the band can be handled by each 10 GbE port, i.e. about 1024 channels. The maximum number of channels in a single packet is roughly 240, so the target band must be broken into several ‘subbands’. For ADP, 6 subbands are used, one for each compute node.

Packet structure

Packets are standard UDP packets with jumbo frames (up to 8192B); the precise size of packets depends on firmware configuration. After the standard UDP headers, there is a 128-bit packet header that describes the packet payload. We have coined this CHIPS: the *common high-throughput interferometer packet specification*.

Packet header

The packet header is as follows, from MSB to LSB:

Name	Data type	Description
roach_id	U8	ID of ROACH board. For LWA-SV, 1-16
gbe_id	U8	ID of GbE output port. Either 0 or 1
n_chan_per_sub	U8	Number of channels in the packet (subband) <240
n_subband	U8	Number of subbands total (configurable)
subband_id	U8	ID of this subband.
(spare)	U8	Currently unused
first_chan	U16	ID of first channel in packet
sequence_id	U64	Unique ID given to each subband (i.e. FFT window)

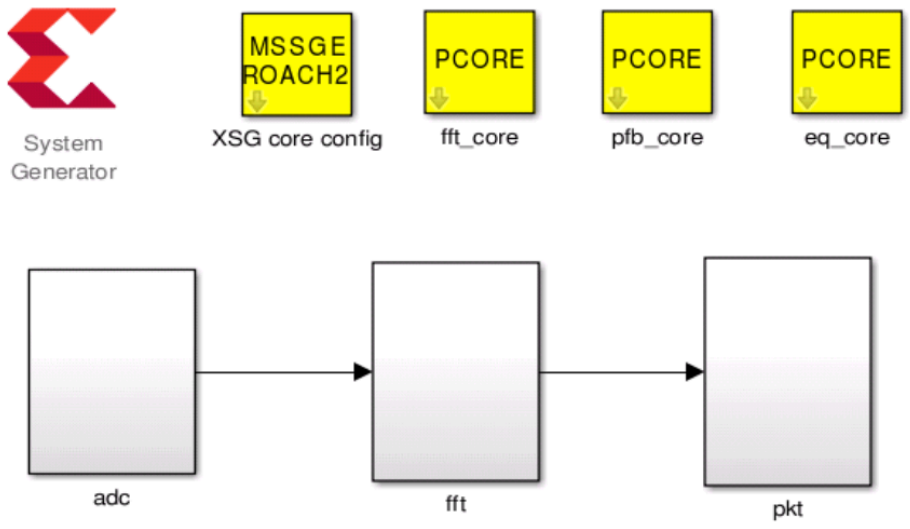
Packet data payload

Each packet contains n_chan_per_sub channels, for all 32 inputs. Each channel is 4-bit real, 4-bit imaginary. Axes are [*channel, antpol, real_imag*], where *antpol* runs 0-31, and channel runs 0-*n_chan_per_sub*. Each channel totals 256 bits (32 antpol x (4 re + 4 imag)), so the total packet size in bytes is:

$$(128 + 256 \times n\_chan\_per\_sub\,)\,/\,8$$

The user needs to make sure the packet size is <8192B, and also that <1024 channels total are selected.

Model walkthrough

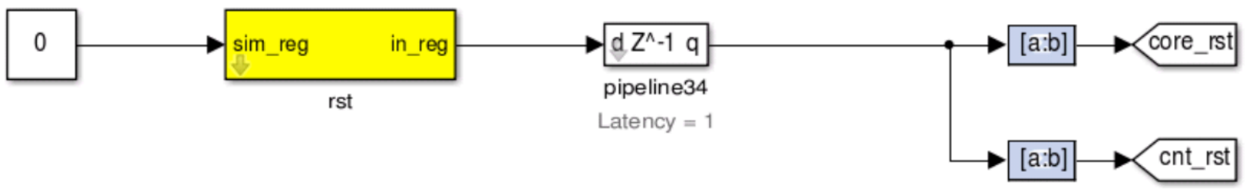


The model consists of three top-level blocks:

- **ADC** – this contains the ADC16x250 digitizer yellow block, and reset / synchronization pulse logic.
- **FFT** – this contains the polyphase filterbank implementation for each of 32 inputs, and the post-channelization 4-bit requantization logic.
- **PKT** – this contains 10 GbE Ethernet packetization logic, including channel selection.

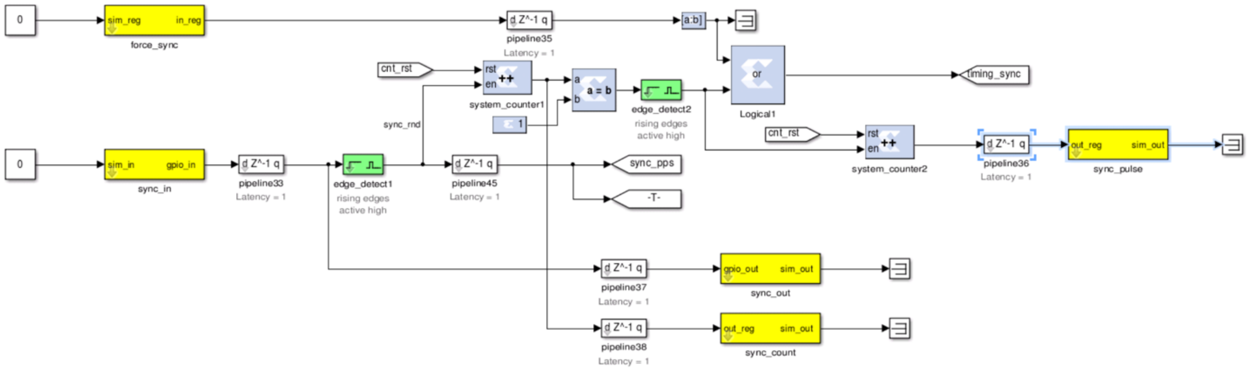
ADC top-level block

Reset logic



Main reset register *adc_rst* provides a reset line for ‘cores’ (i.e. logic blocks), and counter values. The *adc_rst* line should be set high (e.g. 0b11) and then low (0b00) to trigger a reset.

Pulse-per-second and sync pulse

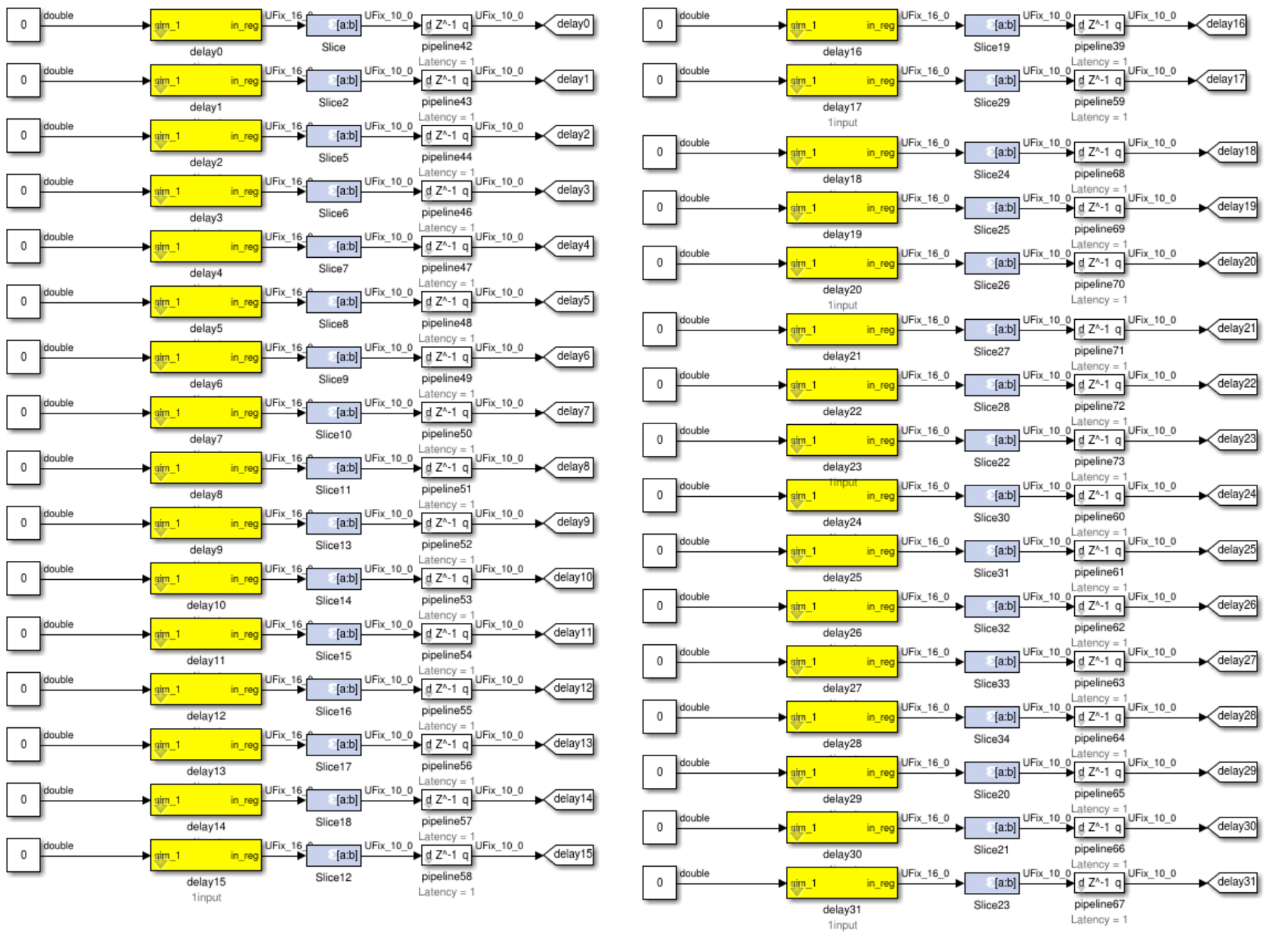


This logic is used to derive a ‘synchronization pulse’, that is required to reset internal logic of many CASPER blocks. Only one sync pulse will be generated and passed to the *timing_sync* goto block, and a counter reset is required to trigger a new pulse to be propagated on *timing_sync.*

adc_sync_in is a GPIO connection that is connected to a pulse-per-second signal derived from GPS. A global goto *sync_pps* is used in the packetizer to make sure values only change on a PPS, not mid-second.

The *sync_out* GPIO propagates the sync pulse to the GPIO output (this is not used at LWA-SV). The *sync_count* register stores a count of how many PPS have occurred since the last counter reset. The *sync_pulse* register stores how many timing sync pulses have been sent.

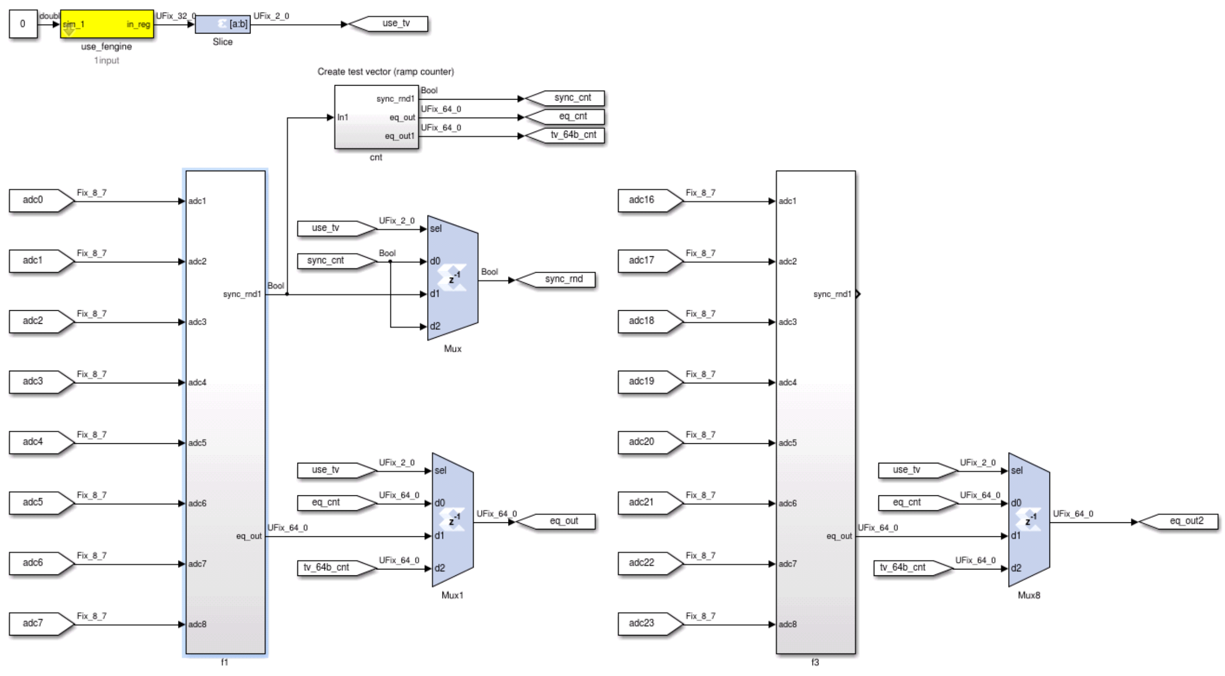
Coarse delay



Each input can be delayed by up to 1024 clock cycles, by writing to *adc_delayX*, where X runs [0, 31]. A minimum value of 4 should be used due to the BRAM-based implementation (values below 3 may result in a delay of 1021 cycles due to wrapping).

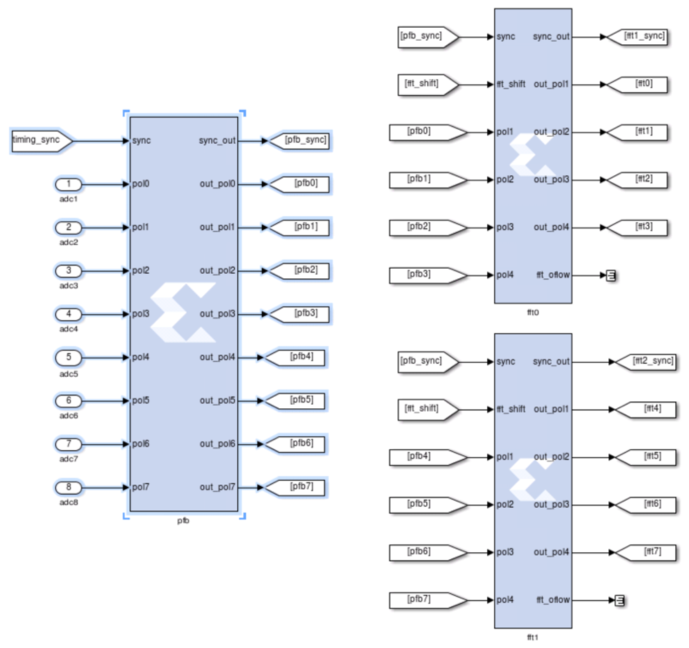
FFT top-level block

Sub-level PFB blocks

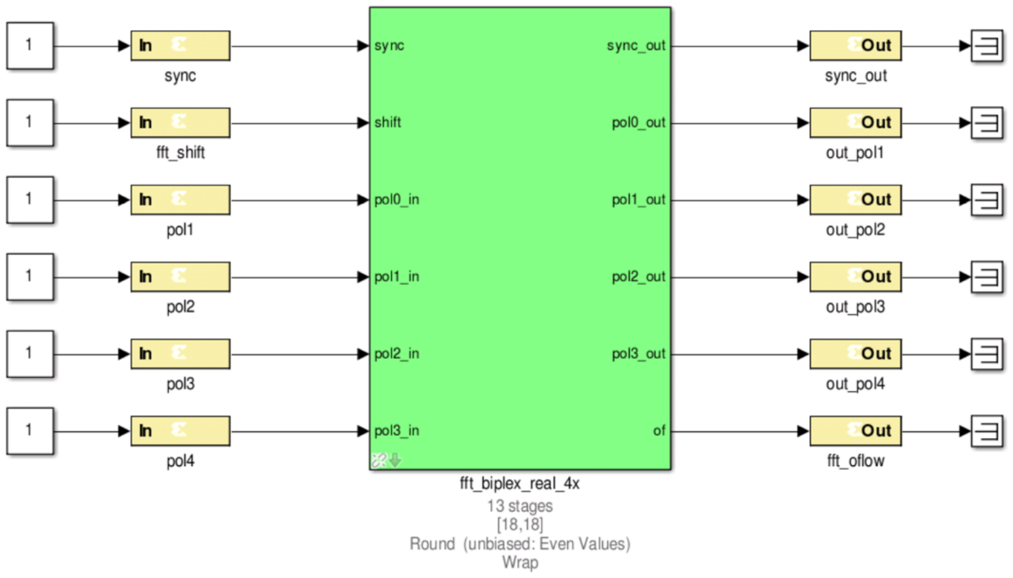


The PFB/FFT for the 32 input signals is split into 4 sub-blocks, *f0 – f3.* The output of each is a *sync_rnd* signal, which is the propagated sync pulse, and *eq_out,* which is all 8 requantized 4-bit input crammed together into a 64-bit word.

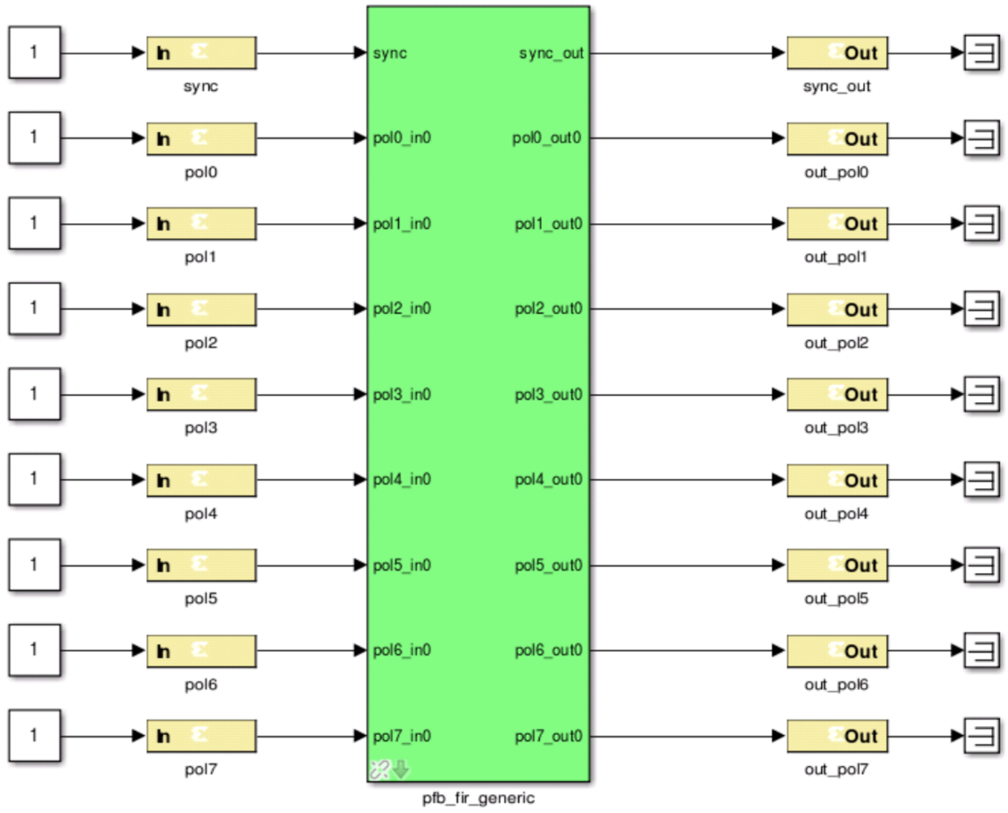
Two test vectors can be selected instead of FFT data using the *fft_use_fengine* register. A value of 0 selects a 64-bit counter that counts to 8192 to be input instead of the FFT data. This is useful for debugging channel offset issues. A value of 2 selects a 64-bit counter that counts to 2^64, this is useful for comparing board synchronization. Setting this to 1 (i.e. True) will select the actual FFT output.



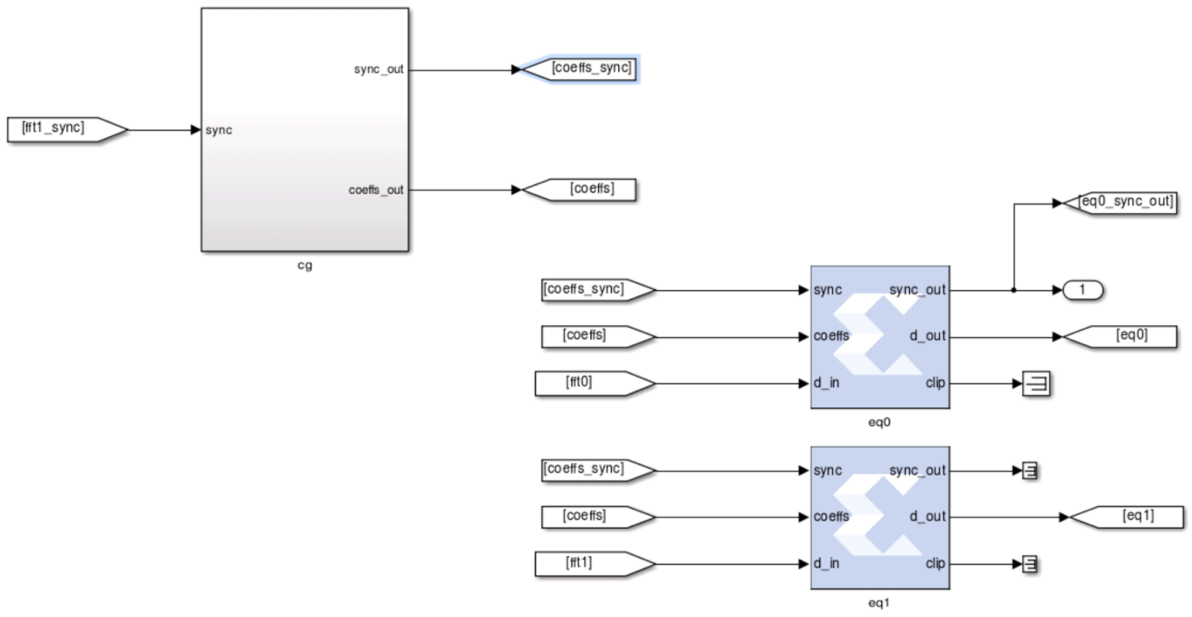
Inside each sub-block is 1x PFB FIR frontend, and 2x FFTs. Both of these are blackboxed.



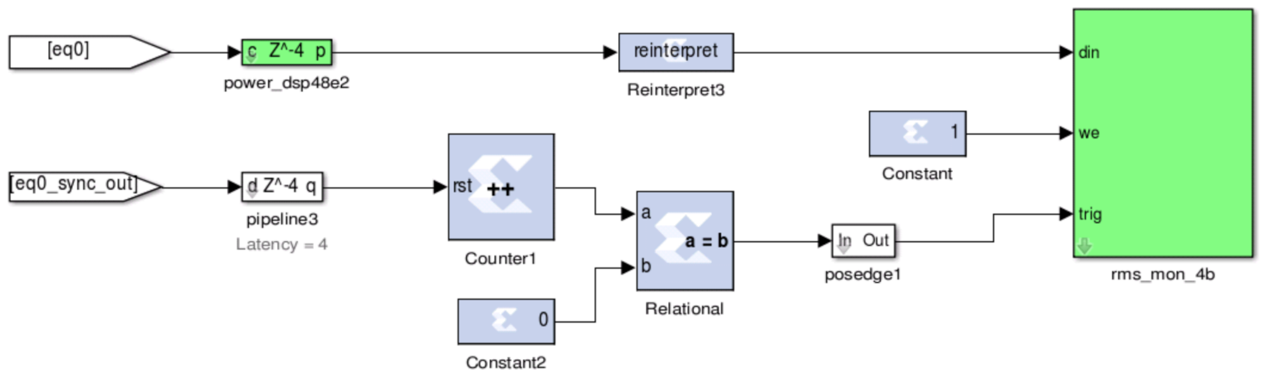
The core of each fft sub-block is of course an FFT. This is blackboxed (so in a separate simulink diagram), but has 13 stages (2^13 point FFT), and is a **biplex_real_4x** CASPER block.



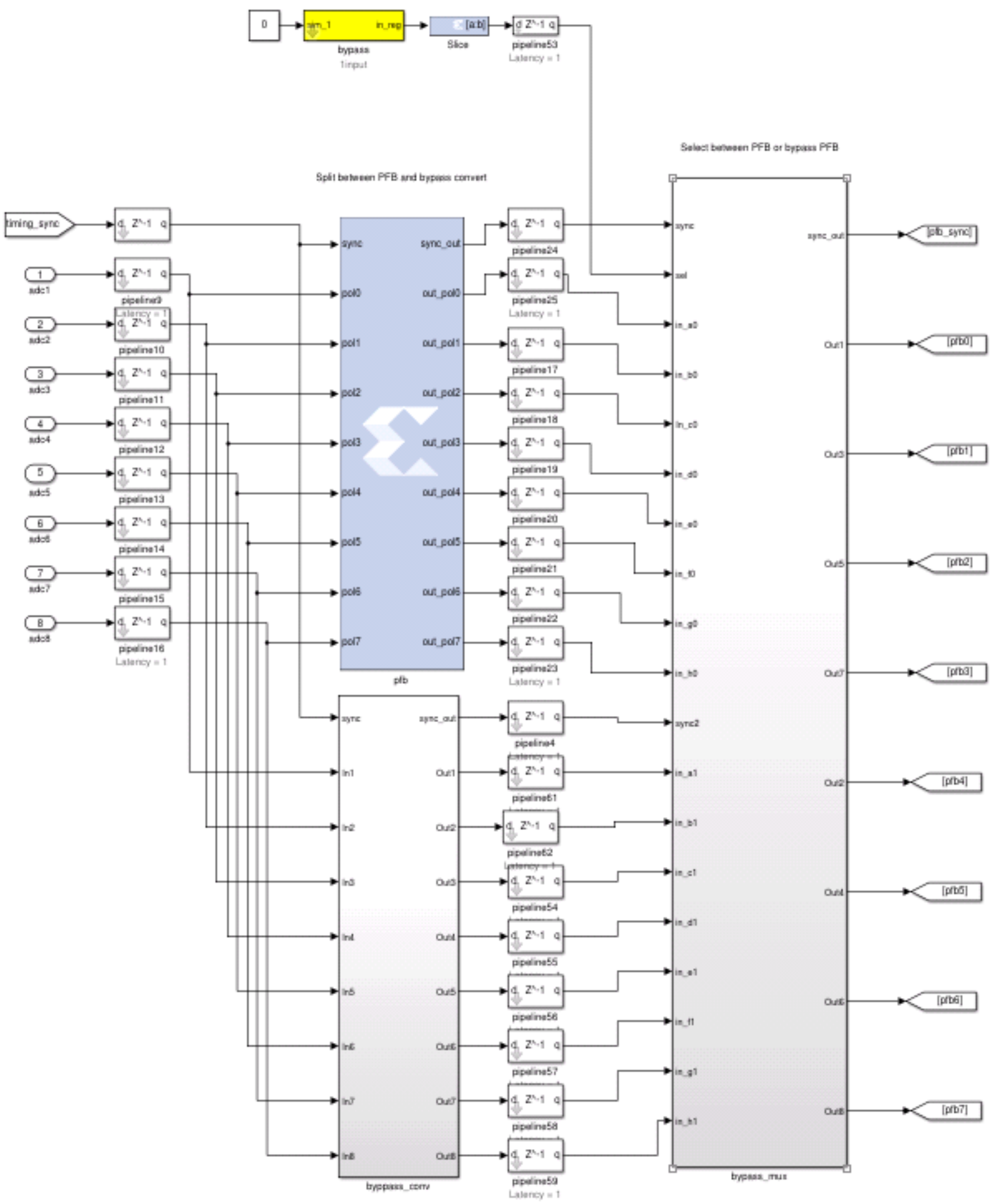
As the F-engine is a polyphase filterbank, the FFT is preceeded by an FIR low-pass prototype filter. This is also blackboxed.



Also under the FFT block is a 4-bit requantizer. This takes the 18_re_18_im signal from the FFT and converts it down to 4_re_4_im in the range [-7, 7]. In order to select the relevant bits, the signal is pre-multiplied on a per-channel basis, by a value written to a shared BRAM *fft_f[0]_cg_bpass_bram*.

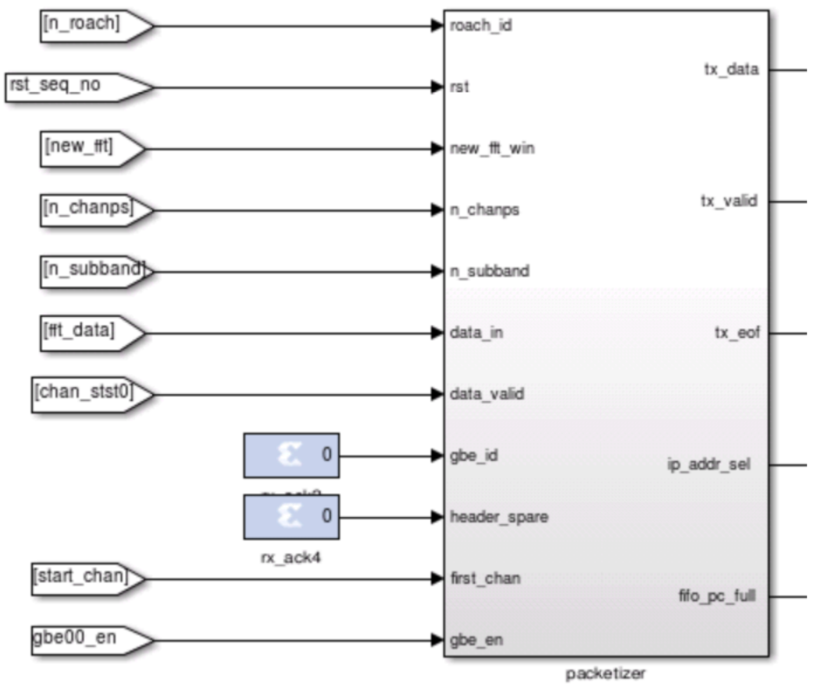


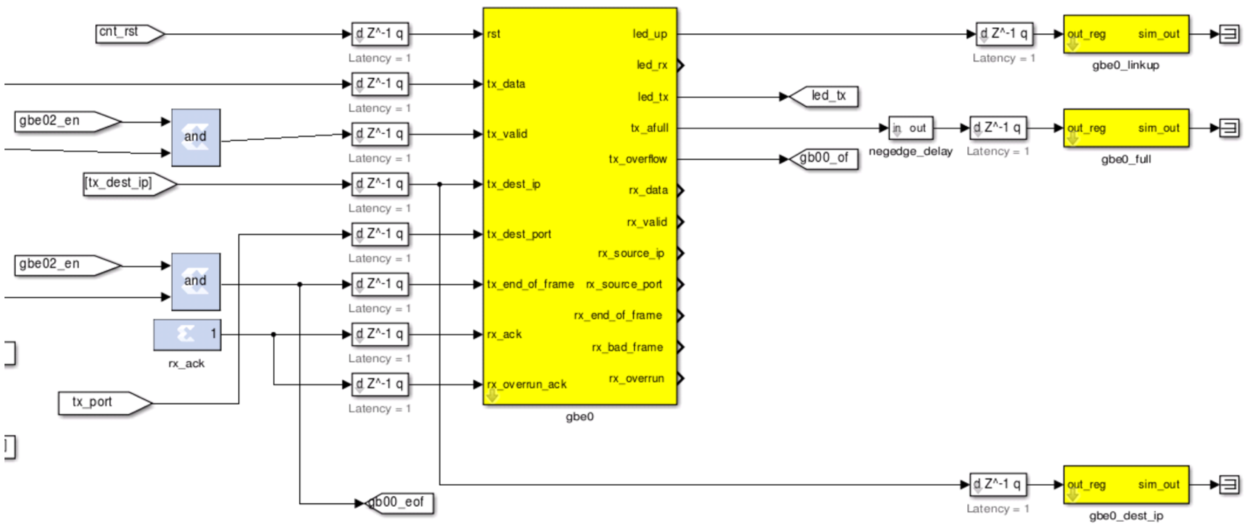
The final item in under a FFT block is a shared BRAM *fft_f0_rms_mon_4b,* which is essentially a 4-bit, single integration spectrometer. It allows the 4-bit quantization value to quickly tuned by providing an instantaneous snapshot of quantized values.



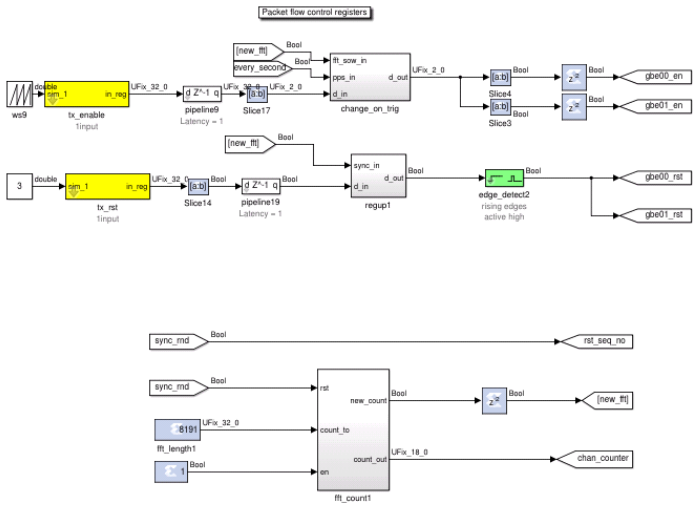
There is now a register *fft_f[0]_bypass*, which can be used to bypass the polyphase FIR frontend.

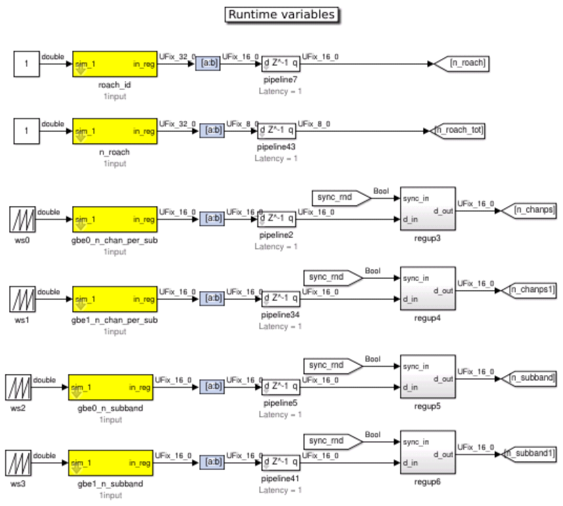
PKT top-level block

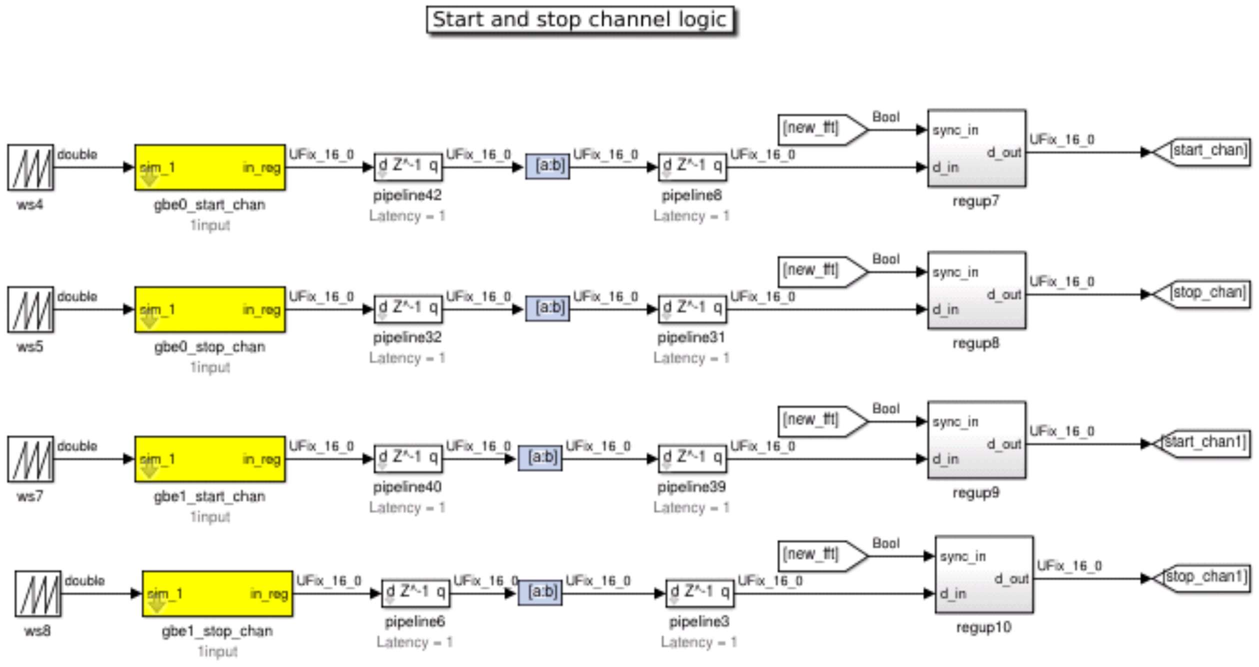




The packetizer selects channels to be sent out over 10GbE and sends them to the compute nodes.





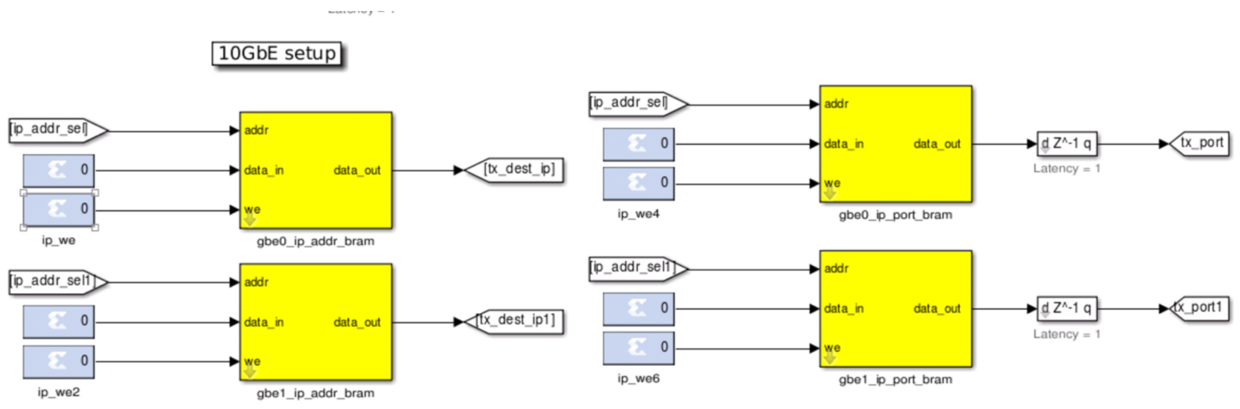


The packetizer has a whole bunch of user-configurable registers. The change_on_trig and regup blocks are used so that values only change at the start of a PPS/FFT/sync window.

The user-configurable registers are:

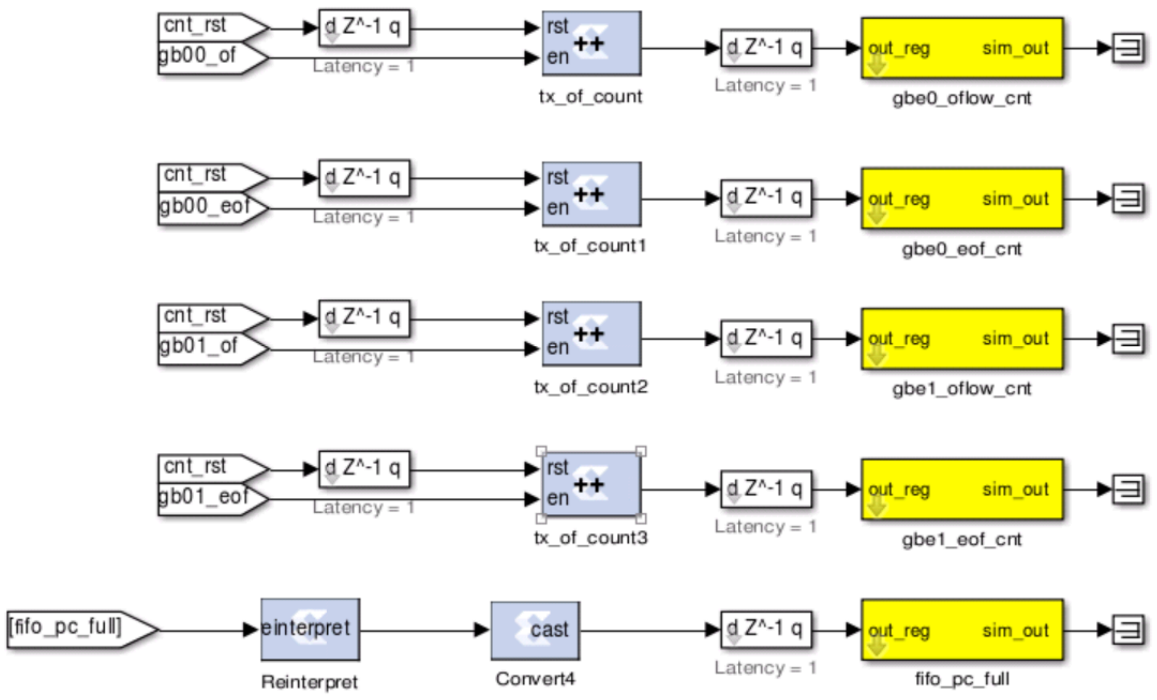
Packetizer control registers

pkt_roach_id	Roach ID number. Should run 1-16, used to identify ROACH
pkt_gbe0n_chan_per_sub	Number of channels per subband for gbe0 packetizer. Allowable values are 10-144 *Updates only after adc_rst*
pkt_gbe1_n_chan_per_sub	Number of channels per subband for gbe1 packetizer. Allowable values are 10-144 *Updates only after adc_rst*
pkt_gbe0_n_subband	Number of subbands for gbe0 packetizer. Total number of channels sent will be n_subband x n_chan_per_sub. Allowable values are 1-32 *Updates only after adc_rst*
pkt_gbe1_n_subband	Number of subbands for gbe1 packetizer. Total number of channels sent will be n_subband x n_chan_per_sub. Allowable values are 1-32 *Updates only after adc_rst*
pkt_gbe0_start_chan	Start channel (lowest channel in range) for gbe0. Allowable values 10-4000 *Updates at the start of each FFT window*
pkt_gbe0_start_chan	Start channel (lowest channel in range) for gbe1. Allowable values 10-4000 *Updates at the start of each FFT window*
pkt_gbe0_stop_chan	Stop channel (highest channel in range) for gbe0. Allowable values 20-4095 *Updates at the start of each FFT window*
pkt_gbe1_stop_chan	Stop channel (highest channel in range) for gbe1. Allowable values 20-4095 *Updates at the start of each FFT window*
pkt_tx_enable	Enable data flow. Controls data flow for both gbe0 and gbe1. LSB is for gbe0, LSB+1 is gbe1. To turn both on, write a value 0b11 = 3. To turn on gbe0 write 0b01=1 or for gbe1 0b10=2 *Updates at the first FFT window after a PPS*
pkt_ts_rst	Reset 10 GbE cores. Only needs to set high once, during INI, to configure the 10GbE core.



Packetizer control BRAMS

pkt_gbe0_ip_addr_bram **pkt_gbe1_ip_addr_bram**	List of IP address to send to. This should be the same length as the n_subbands; each subband is sent to a corresponding IP in this list.
pkt_gbe0_ip_port_bram **pkt_gbe1_ip_port_bram**	List of IP ports to send to. This should be the same length as the ip_addr_bram.



Packetizer output registers

pkt_gbe0_oflow_cnt **pkt_gbe1_oflow_cnt**	This register will be >0 if there's overflows happening in the 10GbE core. This probably means that your n_subbands * n_chan_per_sub is too high. Only used in debugging.
pkt_gbe0_eof_cnt **pkt_gbe1_oeof_cnt**	Count of how many end of frame (EOF) have passed. More simply, the number of packets send out over the 10GbE link. Only used in debugging.
pkt_gbe0_linkup **pkt_gbe1_linkup**	A register that shows if the 10GbE core is configured and the link is up. Returns 1 if up, 0 if down.