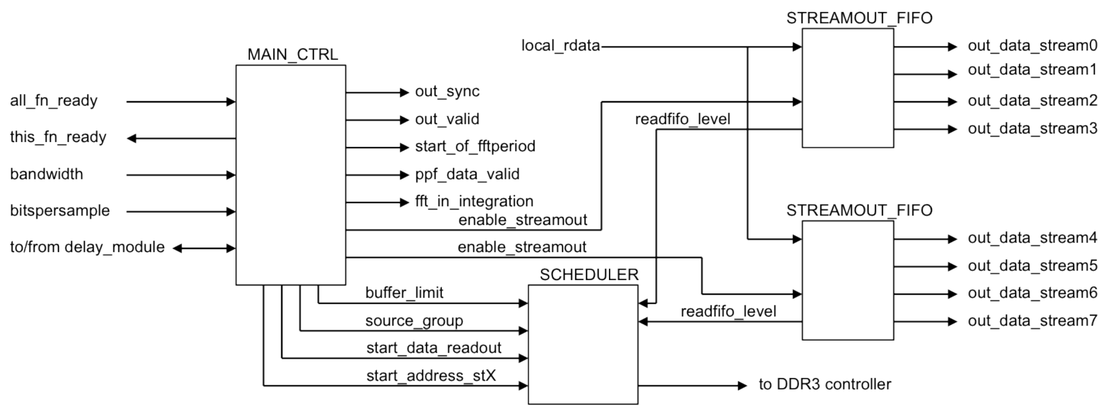
Read Side PKT_RX

The Read Side of PKT_RX reads data from DDR3 memory, reformats it and streams it to the filterbank module.

Architecture



Read Side block diagram

MAIN_CTRL module

The main controller performs the following tasks:

1. Keeps track of the requested number of integration batches.
2. Controls the integration and data streamout procedure.
3. Calculates the start addresses for every integration.
4. Controls communication with the delay module.
5. Provides control signals to filterbank and framer.

These tasks are explained in detail below.

Keeping track of the requested number of integration batches

The batch_size value, written through the MM-interface, indicates how many integrations are requested. This number is stored in a FIFO. This FIFO contains an 'empty' flag, which is asserted initially, but will be deasserted as soon as 2 batch_size values are written. This results in signal 'this_fn_ready' being asserted, which indicates that data processing can start. The batch_size value that will be read out of the FIFO is called 'numintegrations'.

Controlling the integration and data streamout procedure

When an active high 'all_fn_ready' signal is received, the main state machine is started. It reads a value from the batch_size FIFO, decrements the 'integration_count' and starts the process of streaming out data from the DDR3 memory by means of the assertion of the 'start_data_stream' signal. The state machine then transitions from IDLE to INTEGRATE.

In the INTEGRATE state, it checks for the 'end_cycle' signal to be asserted, indicating that the integration of a station pair is finished. If not all 4 station pairs have yet been processed, it starts the process of streaming out data from DDR3 again and waits for a new 'end_cycle' signal. When the 'end_cycle' signal is received for the last station pair, it checks whether the 'integration_count' is 0. If not, then it calculates the new start address for reading the DDR3 and starts streaming out data from the DDR3 again. If 'integration_count' is 0, then the integration process is terminated.

The data streamout process referred to above, is controlled by the streamout_ctrl state machine, which is triggered by the main_ctrl state machine. When the streamout_ctrl state machine receives a 'start_data_stream' signal, then it requests the delay module to send delay data for the appropriate station pair. When these delay figures are received, a down counter is loaded with a predefined value (1023) to allow the streamout FIFOs to be filled with DDR3 data. During count down, the counter value is checked against the 7 LSBs of the coarse delay values from the delay module. These 7 LSBs represent the sample offset. As soon as the counter value equals the sample offset value, the data starts streaming out of the streamout FIFOs, which is triggered by 'enable_streamout_stX'. This procedure is done separately for both stations of the station pair, since they will have different delay figures. The purpose of this procedure is to align the samples before they are applied to the filterbank.

After sample alignment has been done, as much data is streamed out to the filterbank as requested through the number of FFTs per integration.

<u>Calculating the start addresses for every integration</u>

The address per station from which the DDR3 is read is called: ‘start_address_st0’ and ‘start_address_st1’. This ‘start_address_stX’ is in units of 256 bit columns. It is the sum of two components: a base address which is incremented according to the number of FFTs processed and the delay figure, truncated to 128 samples. The figure below explains this.

delay_coarse_stX			sample offset		
27		7	6		0

Coarse delay

The calculation of ‘base_address’ and ‘start_address_stX’ for one station is explained here.

Calculating ‘base_address’:

- To read data for 1 FFT, 16 words of 256 bits must be read from the DDR3 memory.
- Integration time, ‘tintegrate_in_ffts’, is expressed in number of FFTs.
- So ‘base_address’ := ‘base_address’ + (‘tintegrate_in_ffts’ x 16)
- The updated ‘base_address’ might exceed the buffer size, in which case ‘base_address’ must be corrected.
- IF ‘base_address’ > ‘base_address_limit’ – 1 THEN
          ‘base_address’ := ‘base_address’ – ‘base_address_limit’
  END IF

Calculation of ‘start_address_stX’:

- IF ‘delay_coarse_stX’ < 0 THEN
          ‘delay_coarse_stX’ := ‘delay_coarse_stX’ + ‘base_address_limit’
  END IF
- ‘delay_coarse_stX’ := ‘delay_coarse_stX’ + ‘base_address’
- IF ‘delay_coarse_stX’ > ‘base_address_limit’ – 1 THEN
          ‘delay_coarse_stX’ := ‘delay_coarse_stX’ – ‘base_address_limit’
  END IF
- ‘start_address_stX’ = ‘delay_coarse_stX’

So ‘start_address_stX’ is basically ‘coarse_delay_stX’ + ‘base_address’.

The value for ‘base_address_limit’ depends on the bandwidth of a subband and the number of bits per sample. In this case, the values are 16 MHz and 2 bps respectively. This results in:

$$\frac{16\,MHz \times 2\,(Nyquist) \times 2\,bps \times 4\,s}{256\,(word\ size)} = 10^6$$

<u>Controlling communication with the delay module</u>

The signals between the MAIN_CTRL module and the delay module are generated (outputs) and evaluated (inputs) in several places inside MAIN_CTRL. (It is set up such that the earlier established interface between fn_delay_module and pkt_rx is kept the same). Detailed information can be found in the VHDL code.

<u>Providing control signals to filterbank and framer</u>

The control signals to the filterbank are ‘out_sync’ and ‘out_valid’. They follow the same rules as sync and valid signals in other places in the design. The exact implementation can be found in the VHDL code.

The data from the filterbank to the framer is accompanied by control signals that are generated in PKT_RX. These signals are ‘start_of_fftperiod’, ‘ppf_data_valid’ and ‘fft_in_integration’. They arrive in time with the data at the framer, taking into account the latency through the filterbank. The exact implementation can be found in the VHDL code.

SCHEDULER module

The SCHEDULER contains the READOP module which controls the actual reading from DDR3 memory. When it receives the ‘start_data_readout’ signal from the MAIN_CTRL module, it copies the current ‘start_address_stX’ values and requests read access to the DDR3 controller. When access is granted, it does 8 read cycles of 256 bits (4 subbands, 2 polarizations) for both stations. The data read from

DDR3 is stored in the streamout FIFOs (explained in the next section). As long as ‘start_data_readout’ is active and the streamout FIFOs are less than half full, it keeps reading from DDR3.

When either one of the streamout FIFOs is half full, filling that FIFO is suspended, until the half full flag is asserted again. If ‘start_data_readout’ is low, indicating that all data for a particular integration has been read from DDR3, the state machine in READOP transitions back to IDLE.

The read address that is used to read from DDR3 is built as follows:

26:25	24	23:22	21	20:0
source_group	station_count	band_count	pol_count	column_count

Where: source_group is an input to READOP, station_count, band_count and pol_count are generated within READOP and column_count is an internal copy of ‘start_address_stX’, that gets updated during the process.

STREAMOUT_FIFO module

When the FIFO in this module has been given some time to fill up (1023 clock cycles) and the down counter as described in MAIN_CTRL module section reaches the value of ‘delay_coarse_stX(6:0)’, the state machine in STREAMOUT_FIFO module is started. This is done through ‘enable_streamout’ being asserted.

The state machine transitions from IDLE to READ_FIFO, in which state 8 read cycles of 256 bits each are done. The order of the data that is read is: band_A_pol_0, band_B_pol_0, band_C_pol_0, band_D_pol_0, band_A_pol_1, band_B_pol_1, band_C_pol_1, band_D_pol_1. This data is stored in registers.

Then the state machine transitions from READ_FIFO to STREAMOUT. A counter is started that counts off 256 clocks. During these clocks, data is streamed out from the registers. On even count values band_A and band_B registers are streamed out, 2 bits at a time (being 1 sample) and on odd count values band_C and band_D registers are streamed out, 2 bits at a time (being 1 sample). Hence, 128 clock cycles are needed to completely stream out the data from the registers. The 2 bits data is mapped according to an encoding scheme that ensures maximum distance between the 4 possible values of the samples.

At the end of the stream out process, the ‘enable_streamout’ signal is checked. If still asserted, new data is read form the FIFO, otherwise the process is terminated and the state machine transitions back to IDLE.