SKA Project Series
LFAA Station Beamformer structure

G. Comoretto[1], C. Belli[1]

[1]INAF - Osservatorio Astrofisico di Arcetri

Arcetri Technical Report N° 5/2016
05-dec-2016

Abstract

The LFAA tile processing modules (TPM) combine the signals from 256 dual-polarization antennas, in the frequency range 50-350 MHz, into a single *station beam*. Beamforming is performed hyerachically, with the TPM first combining the 16 antennas that are processed in the module, and summing this partial sum in a network packet that travels across the 16 TPMs composing a station.

This station beamformer performs also a corner turning operation, rearranging the channelized samples from the tile beamformer into sequences of contiguous time samples for a single channel at a time.

1 Introduction

The SKA Low Frequency Aperture Array (LFAA) is a synthesis radiotelescope, operating in the 50-350 MHz frequency range, located in the Murchinson shire in Western Australia. It is composed by a number (512 in the first project phase, SKA-1) of identical stations, disposed in a logarithmic spiral configuration.

Each LFAA station operates as a phased beam array, forming one or more electronically steered beams by applying a phase correction to each antenna signal. It is composed of 256 antennas, grouped into 16 tiles of 16 antennas each. Each tile is served by a *tile processing module* (TPM), that digitizes the signals from the 16 antennas and processes them. The tiles are combined together using a general purpose Ethernet network, to implement a distributed beamformer.

The LFAA beamformer combines the signals from the 256 antennas into up to 8 station beams, with a total bandwidth of 300 MHz (50–350 MHz frequency range for a single beam), dual polarization. The process is basically performed in three steps:

- The signal from each antenna is channelized in frequency, into 384 frequency channels with a channel spacing of 781.25 kHz.
- The tile beamformer combines together the signals from a *tile* of 16 antennas, one time sample at a time, by applying an appropriate phase and amplitude (tapering) factor to each channelized antenna sample. The tile beamformer also selects portions of the input band, to produce independent beams, and/or to process these portions in spectral zooming modes. All tiles in a station are beamformed to a common station phase center.
- The station beamformer combines the tile beams into a singe station beam. Mathematically this corresponds to a simple sum of the tile beams but, as they are produced by independent tile processing modules, the process involves data buffering and intra-TPM networking.

The LFAA tile beamformer is described in a different report[3].

1.1 General structure of the Tile Processing Module

The general structure of the TPM is shown in figure 1.

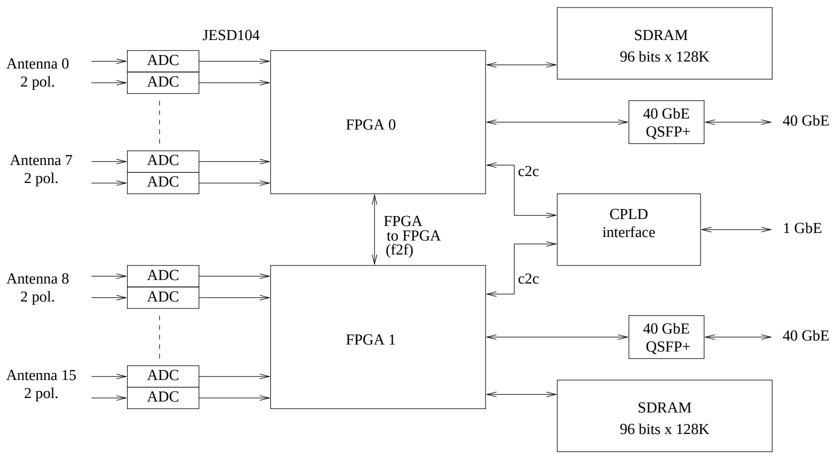

Figure 1: Tile Processing Module structure

The analog signals from 16 antennas, 2 polarizations are converted to digital by 16 dual-channel ADC. Two FPGAs process the signals from 8 antennas each, and are interconnected by a fast parallel LVDS data lane (FPGA-to-FPGA or f2f). Each FPGA is interfaced to an external memory bank, 96 bit wide and 128K word deep, and to a 40 GbE QSFP+ optical interface. Both FPGAs are controlled using a 1GbE interface, implemented in a CPLD.

The signal processing chain in a TPM is shown in figures 2 (general processing for the whole tile) and 3 (internal block diagram for the firmware in each FPGA). The ADCs are interfaced using a commercial JESD204 serial interface module. Signals are coarse aligned, to compensate for gross differences in cable length, channelized using an oversampling polyphase filterbank to a channel spacing of 781.25 kHz, and calibrated. The 8 antennas in each FPGA are beamformed in a tile beamformer. half of the bandwidth is then exchanged between the two FPGAs, and combined together in a full 16-antenna tile beam. Each FPGA from this point processes the signal for half the bandwidth.

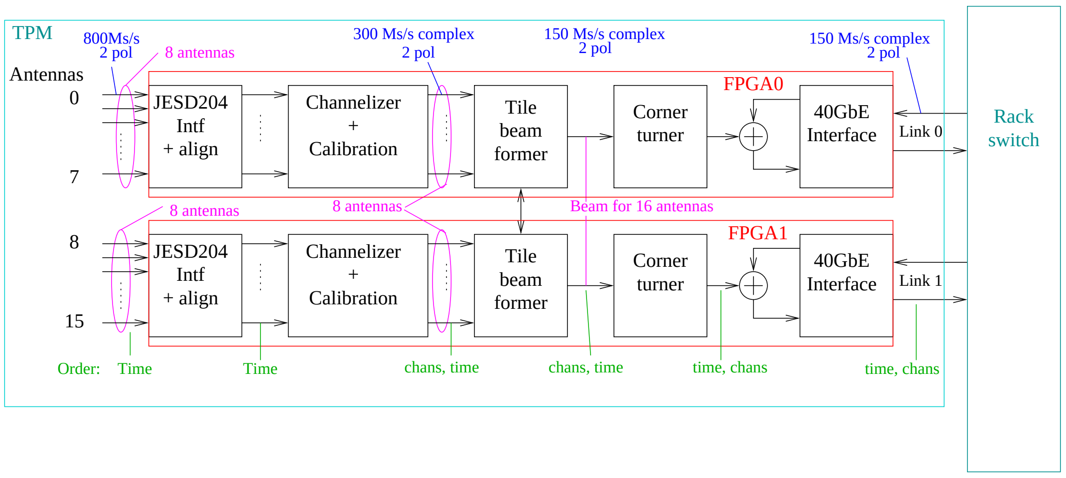

Figure 2: Signal processing in the tile processor

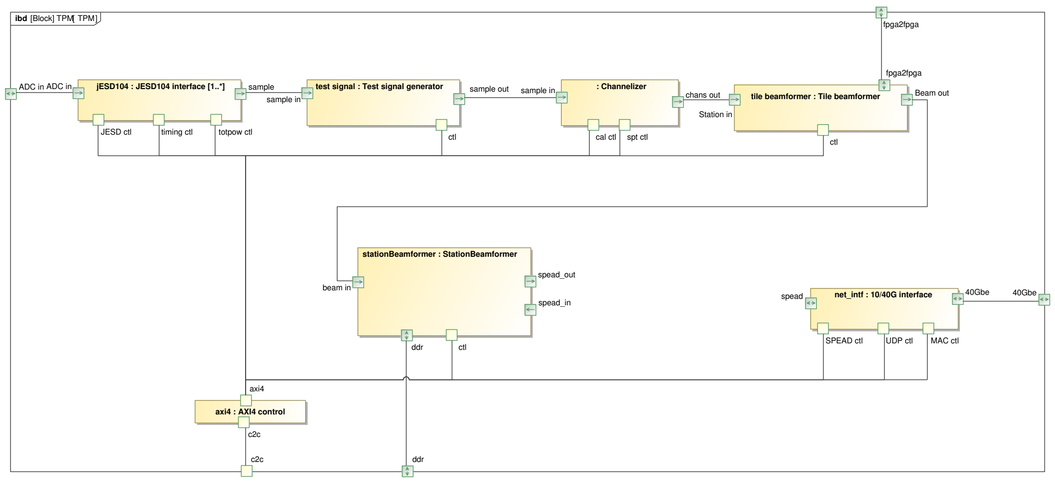

Figure 3: Internal block diagram for the data processing inside each TPM FPGA

1.2 General structure of the Station Beamformer

In the distributed station architecture a packet containing the station signal propagates in a daisy chain along the TPMs composing the station (figure 4), and each TPM adds to this packet the locally beamformed samples.

The traveling packet is formatted according to a streamlined version of the SPEAD protocol ([5]). The data payload is preceded by a fixed header, conveying informations about the payload content, i.e. the frequency and time interval represented, the antenna ID, and the number of participating TPMs. The header is inserted and decoded by a separate module, and these ancillary informations are provided on a separate interface for each packet.

To reduce the number of Ethernet interfaces, the beamformer core modules for both half of the bandwidth can be implemented into only one FPGA. The corner turner is implemented in each FPGA and, for the FPGA without a 40 GbE interface, it is connected with the corresponding beamformer core on the other FPGA using the FPGA-to-FPGA interface. The resulting structure is shown in figure 5. This architecture

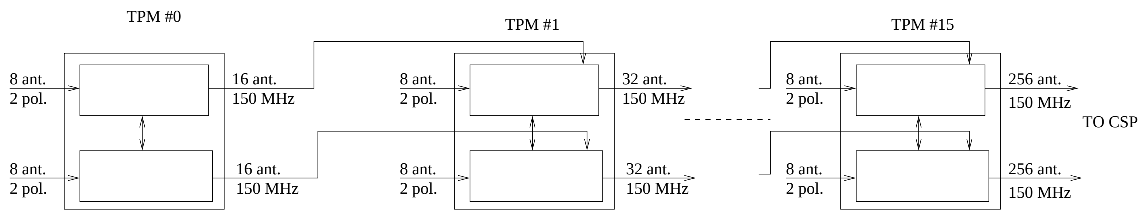

Figure 4: Beamforming for a LFAA station. Each Tile processing Module beamforms 16 antennas, 2 polarizations; TPM's are arranged in a daisy chain for complete beamforming of 256 antennas

will not be implemented in the first version of the firmware, usend in the SKA verification array, but is necessary in the SKA1 implementation to fit the allocated cost and power budgets.

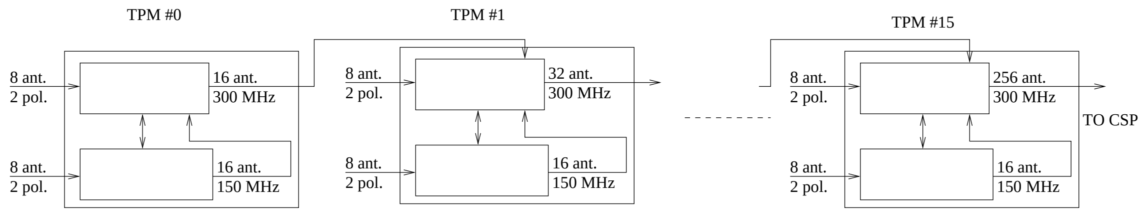

Figure 5: Modified structure for the LFAA station beamforming. To reduce the number of Ethernet interconnect, only one of the two FPGA interconnections is used

The station beamformer is composed of four major blocks (fig. 6):

- a corner turner and buffer memory, based on the external SDRAM memory, that reorganizes the order of the input samples;
- the station beamformer core, that adds the locally generated tile beam to the traveling station beamformer packet;
- a final reformatting module, used only in the last tile of the station chain, that re-samples the beamformed data and buffered into frames with the correct format expected by the CSP.
- a control and monitor interface

The tile beamformer produces frames (tile frames) containing all frequency channels and beams for a single sample time. The CSP must receive sequences of consecutive samples for a single frequency channel at a time, in order to perform a fine channelization of each beamformer coarse channel. The sequence must be significantly longer than the fine channelize filter length, and successive sequences must overlap for this length, in order to provide a continuous fine channelized sample sequence. The station beamformer must therefore generate the beamformed samples in the correct order required by the CSP. Details on this order is given in section 2.

This function is basically a transpose of the time-frequency sample matrix, and is performed in a *corner turner*, a large memory block implemented as an external SDRAM. The individual frequency channels are written to the SDRAM in sparse order, and read back as continuous memory blocks.

The memory read circuitry is also used to implement a synchronization between frames. As the propagation delay along the chain can grow to a relatively large value, up to several microseconds, the local tile packet is read from the memory only when the traveling packet is received, thus minimizing the need for packet buffering. The cornerturner is described in detail in section 3.

The station beamformer is a relatively simple unit. It must synchronism the two packets to be added, and add them together. It is described in detail in section 4.

Data samples from the tile beamformer are represented as 12+12 bit complex quantities. Input frames contain samples for 196 frequency channels, one time, two polarizations. No timing signals are provided with the packet, as the packet time increases monotonically by 1080 ns for each packet. The first received packet corresponds to a deterministic delay after the first captured sample in the ADC input interface.

The traveling packet containing the beam in construction is composed of 16+16 bit complex quantities, to allow for signal growth without the need of intermediate, multiple quantizations or overflow control. Packet is formatted in a standard SPEAD[5] packet, with the header described in the LFAA-CSP ICD and in table

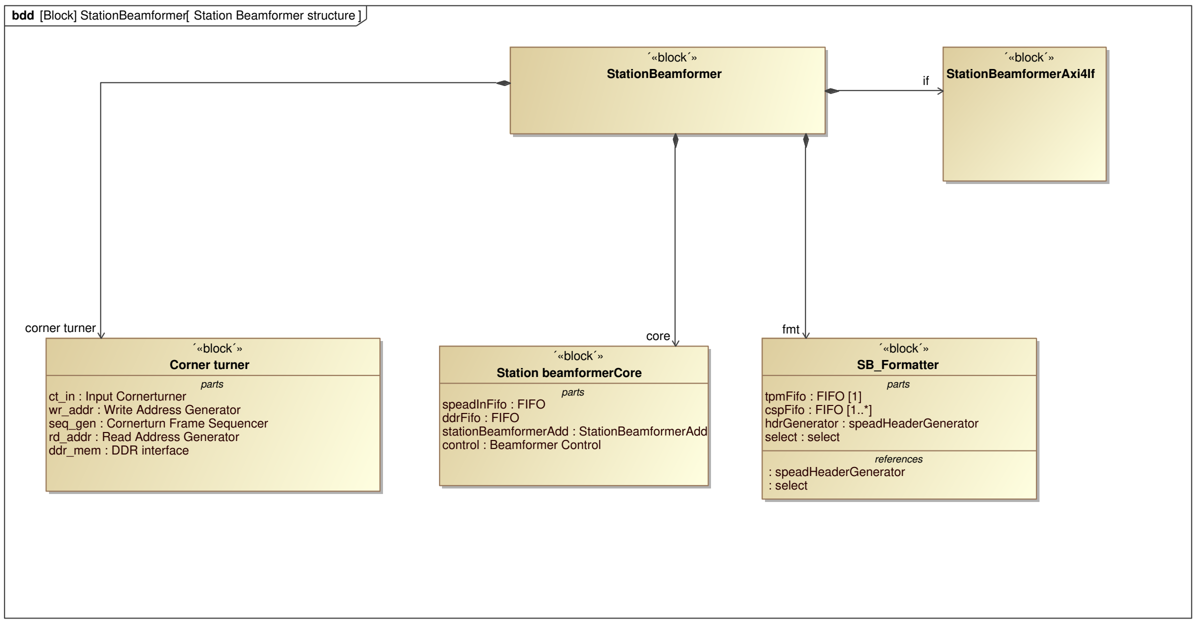

Figure 6: Static block diagram for the station beamformer structure

2. A total of 256 time samples, for 2 polarizations, and 4 frequency channels are stored in a 8192 byte packet payload. The packet sequence number is the only required information for the beamforming process, as it contains the time and the channel ID for the first sample in the packet.

The last tile beamformer provides a reformatting of the packet, in order to generate SPEAD packets with the exact format required by the CSP. In particular samples for different channels are split into separate packets, and all the informations in the SPEAD header are filled in. Data samples are resampled as 8+8 complex quantities, 2 polarizations, and 2048 consecutive sample times. This module is described in detail in section 5.

The module is controlled via an AXI4lite interface. A number of registers can be written to control the behavior of the beamformer module, as described in section 6.1. The input and output data streams use the AXI4 streaming protocol.

2 LFAA timing

A brief summary of the general timing for a LFAA integration is given here. A more detailed description can be found in [2].

The station beamformer generates the overall sequence of beamformed sample frames required by the CSP to correctly perform an integration. Each sample must be correctly time tagged, with an accuracy that does not degrade the time sampling accuracy at the ADC.

The TPM receives ADC samples at 800 MS/s, and channelizes them into oversampled coarse channels at a sample rate of 926 kHz (1080 ns sample spacing). These samples are then packed into UDP jumbo packets of $\approx$ 8 KB each. At the end of the station beamforming process, samples are requantized into 8+8 bit samples, and transmitted to the CSP in frames of 2048 samples, 2 polarizations, one frequency channel. Thus each frame represents exactly 2.21184 ms of data. Data transfer across the station beamformer occurs into shorter packets, of 256 samples each, as each packet carries 4 frequency channels and uses 16 + 16 bit samples. All the description in this section uses however CSP packets, as they represent the minimum available granularity.

The minimum correlator dump time is 0.9 s, i.e. it is longer than the amount of data that can be stored in the TPM memory (see section 3.1). Thus the actual elementary integration time is a fraction of 0.9 s, and with the assumptions of this memo it is $0.9/4 = 0.225$ s. The closest number of CSP frames to approximate this time is 102, corresponding to 225.6 ms. A different number of frames can be chosen, under program control.

The resulting timing for the station beamformer is shown in fig. 7. 102 frames (plus a number depending on the fine channelizer filter preload time) are sent to the CSP in contiguous blocks, corresponding to 225.6 ms of integration time. This time is basically dictated by the available memory on the TPM boards, and could change in the future, being always a simple fraction of 0.9 s. 4 such blocks (with an overlap equal to twice the filter preload time) compose an integration time of 902.43072 ms, that reasonably approximates 0.9 s.

Timing details depend on fine channelizer implementation. Here we will assume that the fine channelizer can process has a number of channels that is a power of 2, with a maximum of 4096 channels. In this way, even at the higher possible resolution the basic block for the fine channelizer would correspond to 2 frames, and there would always be an integer number of finely channelized samples in a block of 102 frames.

It should be possible to include a filter preload time, if required by the CSP. In this case each block for a given frequency channel will begin with a repetition of the previous block, and the total number of frames would be increased accordingly.

The filter preload time $n_f t_f$ depends on the fine filter architecture, the actual frequency resolution and the filter specifications. Therefore it is not assumed that the number of frames n_f required for preloading is an integer. The block length, however, is constrained to an integer number of frames, so a total of $2n_f$ frames (rounded up) is added to the block length. Transmitted blocks are spaced exactly 102 frames (225.60768 ms), with a small overlap to account for n_f.

With the current design, the preload time would be excessively long and this feature is not required at the moment.

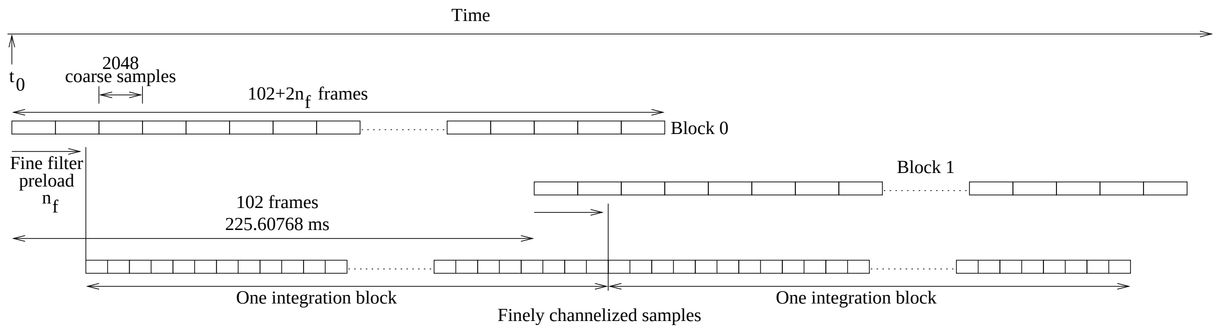

Figure 7: Timing for coarse channelized frames to the CSP. Samples are sent in frames of 2048 samples each. 4 blocks of 102 frames represent the minimum integration time of 0.9 s

3 Cornerturner

The cornerturner receives the tile beamformed samples in sequences of one time value, all frequency channels, and returns them in sequences of one frequency channel (or few frequency channels), many time values. It is basically a memory that is written in a random order, and read back in an (almost) sequential order. Due to the large amount of data to be stored, external SDRAM (DDR3 or DDR4) is used.

Several problems must be considered:

- SDRAM is intrinsically a block devices: memory access is efficient only if data is read or written in blocks above a minimum size
- The amount of data that can be stored in the corner turner memory is limited by the memory size
- Although the TPM FPGAs have a bank of 96 bits each, the standard Xilinx IP core implementing the SDRAM interface is available only for bank sizes up to 64 bits. Until this limitation is waived, only 64 bits are accessible in each bank.

The first problem is managed by using a small corner turner both before (*input corner turner*) and after (*output corner turner*) the DDR memory. The input corner turner stores a sufficient number of frames in order to build a *memory burst frame* of sufficient length. This is composed of successive samples (from successive tile beamformer frames) for a limited number of frequency channels. This number of channels is retrieved simultaneously during memory read, and beamformed in parallel. At the end of the beamformer chain, each frequency channel is stored into a different IP packet, in the output corner turner that is part of the output formatter, and sent sequentially on the 40 GbE link to the CSP. This requires that the CSP is organized in such a way to be able to process simultaneously more than one frequency channel, e.g. in different parallel sections. Actually a CSP architecture composed of independent parallel sections take advantage of such an organization.

The amount of internal FPGA memory, that must be used for the input and output cornerturners, is limited. The SDRAM memory must be addressed in bursts of a minimum of 8 transactions, but the memory speed decreases dramatically for such short bursts. The memory burst length, for the design considered here, is the product of the input and output cornerturner depths, so these three parameters are closely related. For these considerations, we have used a memory burst length of 32 consecutive words, with 8 consecutive samples (input cornerturner of depth 8) and 4 parallel channels (output cornerturner of depth 4).

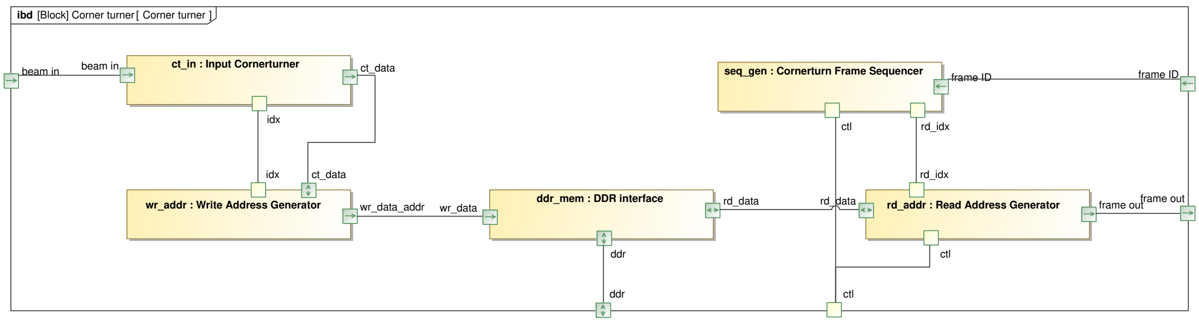
ibd [Block] Corner turner[Corner turner]
beam in
beam in
ct_in : Input Cornerturner
ct_data
idx
idx
ct_data
wr_addr : Write Address Generator
wr_data_addr
wr_data
ddr_mem : DDR interface
ddr
ddr
rd_data
rd_data
seq_gen : Cornerturn Frame Sequencer
frame ID
frame ID
ctl
rd_idx
rd_idx
rd_addr : Read Address Generator
frame out
frame out
ctl
ctl

Figure 8: Cornerturner internal block diagram

The cornerturner structure is shown in fig. 8. The input frames are stored in the input cornerturner. When 8 frames are available, the Write Address Generator fetches blocks of 8 samples by 4 frequency channels and stores them in the appropriate locations in SDRAM. The DDR interface manages the multiplexing of read and write operations, and the generation of memory addresses. The actual low level interface is implemented using the proprietary IP block for the used FPGA family. This interface is external to the station beamformer module, and the interface uses a vendor-independent data structure.

The cornerturner frame sequencer generates the appropriate sequences of frame read operations, for the first tile in the chain, or waits for a packet from the SPEAD interface. These requests are sent to a Read Address Generator, that reads the packet from memory.

3.1 Memory size

The sample data rate from the beamformer is equal to $n_p \cdot B \cdot O_f$, with $n_p = 2$ are the polarizations, $O_f = 32/27$ is the oversampling factor, and $B = 150$ MHz is the bandwidth, providing the inter-FPGA connection has been used to share the input bandwidth, or $B = 300$ MHz otherwise.

The sample width is either 24 bits (12+12 bits complex) that would be very efficient if 96 bit DDR words could be used. In the present instantiation of the DDR interface, using 24 bits in a 64 bit SDRAM wastes half of the on-board memory. Disabling the unused chips anyway reduces the required power.

Assuming $B = 150$ MHz and 12+12 bit samples, the total bit rate would be 11.4 Gb/s, or 178 MT/s for a 48 bit memory bus (each for read and write, 356 MT/s total). The memory can support 1600 MT/s maximum, so even a low efficiency in the memory usage can be tolerated.

Memory is organized as 128 M-words, thus assuming dual buffering the maximum time that can be stored in a frame is about 0.36 s. Considering the simple addressing scheme described in section 3.4, only 75% of the memory space can be used, reducing the maximum time to 0.283 s.

3.2 Architecture

The general architecture of the corner turner/station beamformer is shown in more detail in figure 9.

The input frames from the channelizer are rearranged in a small corner turner memory. Rearranged memory bursts are then fed into a rate-change FIFO, that provide the memory controlled with uninterrupted data for the duration of each memory write burst. The address generator provides a write address depending on the input corner turner status, and on the input frame count.

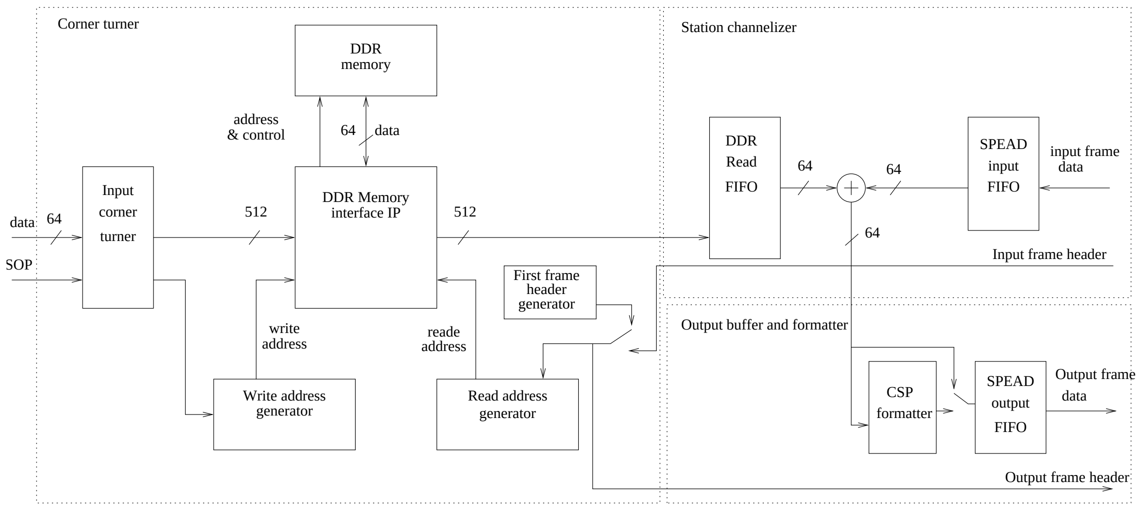

Figure 9: General architecture of the corner turner

Data are read from memory when a frame is received from the Ethernet interface. This frame corresponds to a station beam that is being assembled traveling in sequence across all the station TPMs. SPEAD frame header is used to locate the corresponding data in DDR memory, and to read it. The locally formed beam is summed to the station beam, and the result is sent to the next TPM.

The first TPM in the chain directly generates the station beam frames, and sends them to the UDP interface, without waiting for a SPEAD packet to be received. The last TPM performs a final format change, to produce the frames to be sent to the CSP.

3.3 Input corner turner

The input corner turner is the component that receives frames of channelized data from the tile beamformer, and organizes them into memory bursts of sufficient length to efficiently use the memory.

DDR memory accesses occur in bursts, i.e. sets of read/write operations on contiguous memory addresses. The memory addressing protocol requires several clock cycles, and a new address can be sent only after a predetermined number of clock cycles from the previous one.

The minimum length for a DDR memory access is 8 words, i.e. 64 bytes [1]. Longer burst size typically result in a better memory efficiency, with optimum length around 32–64 words. This exact length depends on the memory controller IP, that is typically a vendor specific package with no possibility for adjustments.

The smallest unit of beamformed data is composed of 2 complex numbers, i.e. 2 polarizations for one time sample and one frequency channel. With 16 bit representation this corresponds to 8 bytes, or a DDR word. 12 bit samples result in a 48 bit word size, but we decided to use a whole word anyway, with the upper 16 bits set to zero, to avoid the need of a complex packing/unpacking scheme and of burst sizes that are not a power of 2.

By storing 8 frames in a local buffer it is possible to create sequences of 8 consecutive samples for the same channel. Groups of channels (4 in the current implementation) are combined together in order to create a burst of sufficient length. These frequency channels will be separated in an output corner turner, placed in the formatter in the last TPM for a station chain (see section 5.1).

In figure 10 the sequences of samples arriving from the time beamformer and sent to the SDRAM memory are shown. Samples arrive from the tile beamformer in the order described in the upper part of the figure. Each cell in the figure represents a dual polarization complex sample (48 bits). Samples arrive in frames of one time step, all frequency channels and are stored in memory in this order. Each of the two FPGAs process 192 of the total 384 frequency channels, so each frame is 192×8 bytes long.

When a full set of 8 frames (time steps) is stored into memory, these samples are read back in the order listed in the lower part of the figure. Groups of 32 dual polarization samples, corresponding to a memory burst of 32 SDRAM words, are read in groups of 8 words and sent to the memory controller at the required speed. The memory read is performed at the speed dictated by the memory controller, in its clock domain. Each group of 8 words is then transferred to the SDRAM memory at a 8x clock speed.

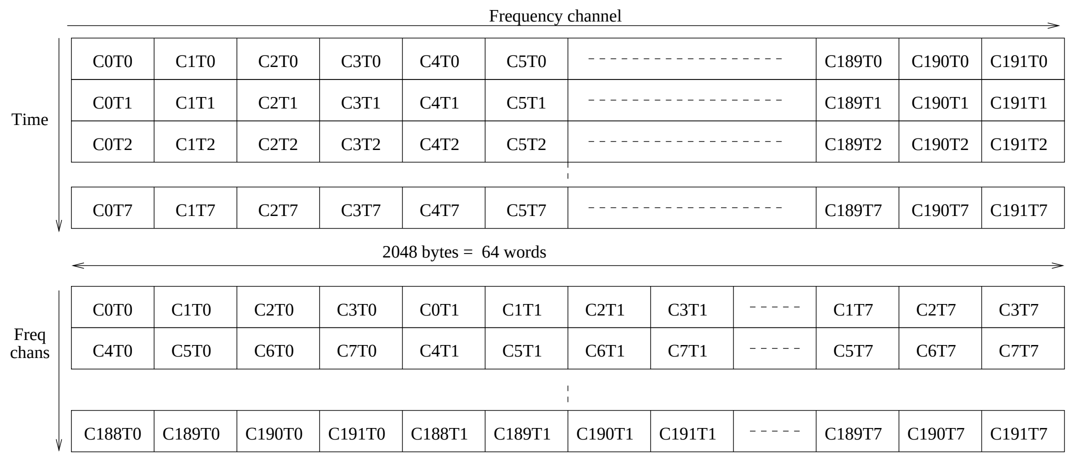
Frequency channel
Time
C0T0 C1T0 C2T0 C3T0 C4T0 C5T0 C189T0 C190T0 C191T0
C0T1 C1T1 C2T1 C3T1 C4T1 C5T1 C189T1 C190T1 C191T1
C0T2 C1T2 C2T2 C3T2 C4T2 C5T2 C189T2 C190T2 C191T2
C0T7 C1T7 C2T7 C3T7 C4T7 C5T7 C189T7 C190T7 C191T7
2048 bytes = 64 words
Freq chans
C0T0 C1T0 C2T0 C3T0 C0T1 C1T1 C2T1 C3T1 C1T7 C2T7 C3T7
C4T0 C5T0 C6T0 C7T0 C4T1 C5T1 C6T1 C7T1 C5T7 C6T7 C7T7
C188T0 C189T0 C190T0 C191T0 C188T1 C189T1 C190T1 C191T1 C189T7 C190T7 C191T7

Figure 10: Sequence of samples at the input and at the output of the SDRAM input cornerturner

3.4 SDRAM write address generation

The TPM SDRAM memory uses DDR3 memories with a word length of 96 bits. Due to limitations of the memory controller IP, only 64 bits can be used. In the current implementation, the standard 64 bit memory controller is used, but the upper 16 bits transferred are always zero.

Each word represents a single time sample for one frequency channel, 2 polarizations, i.e. 2 consecutive 64 bit words correspond to one time sample for 2 consecutive frequency channels. As seen above, samples are written in the DDR in blocks of 8 frequency channels, for a total of 8 memory words. SDRAM access is organized in rows, with each row containing 2^{10} words (10 bits of addressing). Data is written to the memory in bursts of 32 words each, i.e. 8 contiguous samples for 4 frequency channels, as in figure 10. Thus a SDRAM row represents 256 samples of channelized data.

DDR3 memory chips in the TPM contain 8 banks of 16384 rows each, for a total of 128 M-words. Successive memory bursts store groups of 4 frequency channels, for a total of 48 bursts, into different row and bank addresses. All bursts start at an address that is a multiple of 32.

A SDRAM location is written by opening a row, with an ACCESS command, writing a burst of a multiple of 8 words, and closing the row with a PRECHARGE command. This requires a considerable amount of time. Therefore, for memory efficiency, contiguous bursts are sent to different memory banks, in order to interleave the SDRAM ACCESS, WRITE and PRECHARGE operations. An optimal write sequence would overlap the operations, by placing these operations in the correct order. Unfortunately this is not possible with most commercial SDRAM controller IPs, that are optimized for generic, relatively long memory bursts and not for a specific addressing sequence. As a consequence the overall efficiency of the memory write is limited to around 25% of the maximum theoric speed.

With this scheme each of the 8 available banks contains 6 contiguous blocks of samples. Each block is a contiguous addressing range, containing all time samples for 4 frequency channels. In theory it would be possible to use up to 2730 rows for each block (700 thousand samples, 0.75 s of data). Partitioning the SDRAM into 8 blocks results in a much easier addressing, but the SDRAM capacity is reduced to 2^{19} samples, or 566 ms of data. In the first case, the SDRAM corner turner could generate contiguous sample blocks of 0.3 s (including double buffering), i.e. 1/3 of the basic integration time of the CSP, while in the second the sample block should be reduced to 0.225 s, i.e. 1/4 of the CSP integration time. We adapted the second addressing scheme. This is summarized in table 1.

A simple write addressing scheme for the SDRAM is thus:

- SDRAM address space is divided into blocks of 2048 columns, i.e. 2^{21} consecutive words. Bank address is considered separately.
- Each memory block has a starting address that depends on the frequency channel F. The SDRAM bank number is specified using both the SDRAM bank address and the upper 3 bits of the row address, as shown in figure 11.
- The time of a sample is given by the 21 least significant bits of the address, with the 2 LSB not considered (as each sample is 4 words long).
- Time wraps around these 21 address bits.

Burst length	32	words
Frequency channels per row	4	
Time samples per burst	8	
Time samples per SDRAM row	256	
Frequency channels per SDRAM	192	
Frequency channels per SDRAM bank	24	
Max contiguous rows per channel	2730	
Used contiguous rows per channel	2048	
Max contiguous time interval	699	ms
Used contiguous time interval	566	ms

Table 1: Parameters of the corner turner SDRAM memory

- Each frame of 8 time samples begin at a given offset from the start of each memory block, with an offset increment of 32 words between frames.

This is summarized in figure 11. The time sample index $t_{18} : t_0$ is mapped to bits 23:5 of the memory address, with cyclic wraparound. Channel index bits $c_8 : c_2$ are mapped to the remaining bits, as shown in figure. Bit c_0 (odd/even channels) corresponds to the two FPGAS, as odd and even channels are processed in different FPGAs. Different mapping of the channel index bits can be used, depending on the detailed processing scheme in the CSP.

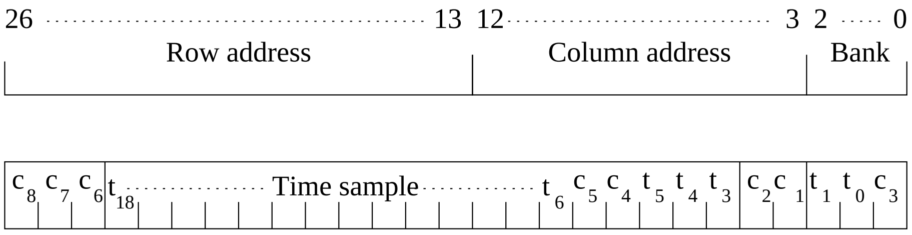
26 ····· 13 12 ····· 3 2 ····· 0
Row address
Column address
Bank
c8 c7 c6 t18 ····· Time sample ····· t6 c5 c4 t5 t4 t3 c2 c1 t1 t0 c3

Figure 11: DDR SDRAM address scheme, for 27 bit addressing, showing mapping for time sample index $t_{18} : t_0$ and channel index bits $c_8 : c_1$

3.5 Simplified version of the cornerturner

In an initial phase, for testing purpose only, the cornerturner will not be implemented. Only frames of 128 channels per FPGA will be processed (200 MHz of effective processed band). Each frame will be stored in a contiguous memory block, of 128 words. 8 frames are stored in a memory column (1024 words), and only one bank is used. The resulting address scheme is shown in figure 12.

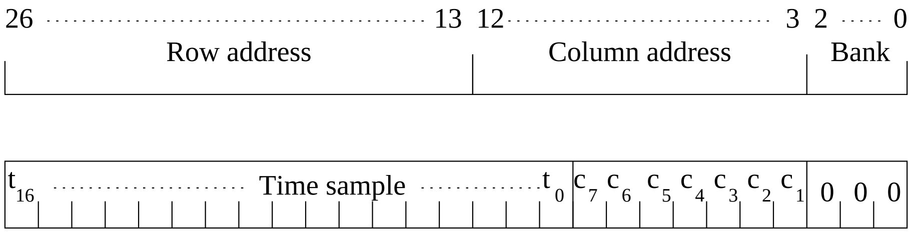

Figure 12: DDR SDRAM address scheme, for linear frame addressing, showing mapping for time sample index $t_{18} : t_0$ and channel index bits $c_8 : c_1$

The "cornerturner" is thus simply a FIFO that stores a frame and changes the bus width and clock domain from that of the tile beamformer to that of the DDR.

3.6 SDRAM read address generator

The SDRAM is read any time a SPEAD frame is received from the 40 GbE interface. The frame header specifies the starting time and the frequency channel of the frame samples. From these, a starting address and bank number is computed, according to the scheme in figure 11. The starting memory address is arbitrary. For efficiency reasons, it is advisable that the starting address is always a multiple of 32 words, i.e. it corresponds to a write burst.

The memory read generator then generates a sequence of read operations, that are interleaved with the write operations. A state machine ensures a balance of read and write operations to avoid possible losses of data. Read operations are always contiguous, to increase the SDRAM speed. By reading a whole row in a single read operation, the speed efficiency is around 85-90%, resulting in an overall SDRAM usage efficiency (read+write) around 35-40%.

3.7 Frame sequencer

The first TPM in the station chain does not receive any frame. A state machine then simulates a set of dummy received frames, with null samples and a complete set of time interval and frequency channels. Both these quantities and the frame rate can be programmed, in order to reduce the generated data traffic by the complete beamformer. Frame generation is started when a complete set of samples, for the whole elementary integration time, is available in SDRAM memory.

The sequence of read operations is configurable using the control interface. Parameters that can be controlled are:

- The first and last frame to be processed. This effectively determines the time interval processed by the telescope.
- The integration length, in CSP frames. The number of generated TPM frames is 8 times this number, as each CSP frame (2048 samples) is composed of 8 consecutive TPM frames (256 samples).
- The spacing between successive integrations, in CSP frames. This is equal to the integration length minus the overlap between integrations.
- The number of consecutive coarse channels sent to the CSP, expressed as a power of 2. This value is currently fixed to 4, but can be increased up to 64 channels if the CSP is capable of analyzing more than 8 channels (4 from each FPGA) in parallel.
- A *guard time* between successive SPEAD packets. As the fetch time of a packet from SDRAM is shorter than the time required to send it over the Ethernet interface, this time must be set long enough to avoid an overrunning condition. Back-pressure from the Ethernet interface is not a safe method to determine the packet transmission rate, because an overrun condition may occur downstream the station beamformer chain.

The sequencer fetches a whole integration period in blocks of 4 channels and 256 time samples, for consecutive time samples. When all blocks have been sent for a group of 4 channels, the next 4 channels are sent, until all channels are processed.

In the output cornerturner (section 5.1) these 4 channels are separately buffered into 4 CSP frames, that are sent sequentially when filled. Therefore the CSP receives 8 frames (4 frames from each FPGA) for the same time interval and for 8 consecutive coarse channels, then 8 frames from the next time interval, and this repeats for the whole integration period. The 8 frames can be addressed to separate destinations, for a distributed CSP.

If more than 8 frequency channels can be processed in parallel by the CSP [1] the sequence is slightly more complex:

- 8 consecutive frames are fetched for a group of 4 channels. These will be combined into 4 separate CSP SPEAD frames for the CSP, in the output cornerturner.
- The same 8 frames, for other 4 channels, are fetched to produce other 4 CSP frames. The process is repeated until all channels that can be processed in parallel by the CSP are fetched and sent.
- The sequence is repeated for the next 8 time frames, until all the integration period is processed.
- The whole sequence is repeated for another block of channels, until all channels have been processed.

[1] the current version of the CSP is composed of 16 parallel units, so the optimum number of parallel processed channels would be 16.

4 Beamformer core

The beamformer fetches from the cornerturner a memory packet of contiguous time samples, for 4 frequency channels and 256 time samples, and propagates it through the daisy chain. The station TPM chain is built dynamically, by providing each tile the Ethernet address of the next tile in the chain.

The first tile in the chain generates the frames autonomously, as described in section 3.7. In this case the beamformer waits for the cornerturner to have stored a complete time sequence in the DDR memory. In the other cases, the beamformer waits for a frame from the SPEAD decoder and, using the SPEAD packet number as an index, fetches the same packet from the cornerturner. Both packets are buffered in a FIFO. The packet from the DDR memory is fetched at a higher speed with respect to that from the Ethernet interface, but has a relatively high latency due to the possibility of a conflict with an ongoing memory write.

When both packets are available, they are fed to an added module, that adds together corresponding samples from the two frames. The adder checks for bad samples, and flag as bad the corresponding output samples. This means that a bad sample in *any antenna* in the station causes the corresponding sample in the station beam to be flagged as bad.

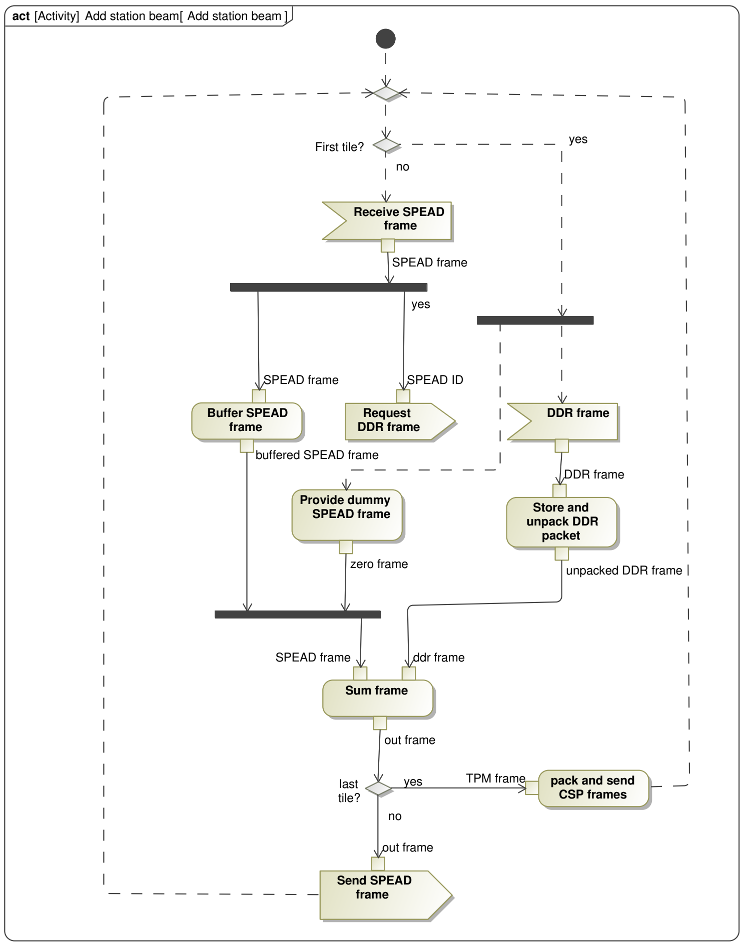

Figure 13: Beamformer flow diagram

5 Output formatter

The data frame is reformatted in a UDP packet together with the received frame header. Packets are sent using the SPEAD format, that is composed of a header and a payload (heap). The actual header is built in a separate module, but the required values are provided as a structure (VHDL *record*) that is generated inside the formatter module. The SPEAD formatter takes these values, and uses them to build the header. The UDP header is also built by this external formatter, using a destination index specified in the `t_id` field of the AXI4 stream. Each index corresponds to an appropriate combination of IP address and UDP port. Index 3 is used for the next TPM in the chain, and indexes 4-15 for up to 12 different destinations in the CSP.

Informations in the header includes (at least):

- A packet counter
- Elementary integration block number
- Starting sample time, number of samples
- Starting frequency channel number
- Number of antennas beamformed so far
- Station ID

The SPEAD header is composed as in table 2, both for TPM packets and CSP packets, but in TPM packets words 3–7 are not filled in. The heap counter is used to locate the packet in the DDR memory.

Word	ID	Description
0	–	SPEAD Header word, 0x5304401000000008
1	0x0001	Heap counter: Logical chan ID & Packet counter
2	0x0004	Packet payload length
3	0x1027	Reference time (s)
4	0x1600	Timestamp (ns)
5	0x1011	Center frequency (Hz)
6	0x3000	CSP channel info: beam ID & frequency ID
7	0x3001	Antenna info: sub-array ID & station ID & n. of antennas
8	0x3300	Sample vector: pointer to 1^{st} sample in packet

Table 2: SPEAD header format

The meaning of individual entries are:

Header word: The word specifies that this is a SPEAD packet, and provides some informations on the SPEAD dialect used. In particular these packets use SPEAD version 4, with an item pointer width of 64 bits, item ID width of 16 bits, and 8 items in the header.

Heap counter: The word is required to be unique for a given packet. It is composed by the ID of the first logic channel, in bits 47–32, and a packet counter that increases monotonically and continuously from module reset. These values are used to fetch the appropriate packet in the SDRAM memory, and to generate the other informations in the CSP formatter.

Reference time: This value is specified by the control interface. It specifies an integer number of seconds in the standard UNIX time format. It represents, together with the timestamp, the time of the first sample in the packet.

Timestamp: It is derived from the Packet counter, and express the time, relative to the reference time, of the first sample in the packet, in ns.

Center Frequency: It represents the center frequency of the channel, expressed in Hz. Is equal to the $\Delta f(f_i + 1/2)$, where $\Delta f = 781250$ Hz and f_i is the frequency ID.

Channel info: Is composed of the beam ID, in bits 31–16, and the frequency ID, in bits 15–0. These values are derived from the logical channel ID, in a way similar to that used for the tile beamformer[3].

Antenna info: Specifies the sub-array ID for this station, in bits 39–32, the station ID, in bits 31–16, and the number of antennas composing the station, in bits 15–0. All these values are provided by the control interface.

The SPEAD formatter is implemented in a separate module. It receives a `t_spead_header` structure, together with the AXI4 stream with the data, containing the above informations. The structure is defined as:

Most of the required fields are provided by the program interface, i.e. directly by the control program. The module provides only:

Field	Size	Use
`sync_time`	32	Time in Unix seconds
`time_stamp`	40	Start time in ns
`sub_array_id`	4	Subarray ID
`station_id`	16	Station ID
`nof_included_antennas`	10	N. of antennas
`ant_start_id`	8	Antenna start ID
`nof_included_channels`	8	N. of channels
`channel_start_id`	16	Channel start freq
`tpm_id`	8	Tile processing module ID

Table 3: Definition of the fields for the t_spead_header structure (subject to to changes)

- The number of antennas, incrementing the number received from the previous TPM
- The start time in ns, by multiplying the frame number by the frame period (1080 ns)
- the channel start frequency, by multiplying the channel number by the channel spacing.

These values are required only for the packets sent to the CSP (section 5.1). All values, apart from the heap counter, can be left blank in the packets along the TPM chain.

5.1 CSP formatting

The last TPM in the chain must perform a more complex set of operations, in order to reformat the packet in a format adequate for the CSP.

The accumulated beamformed samples must be requantized to the final sample size (usually 8 bits), with proper treatment of overflows. Some statistics may be collected on the packets (e.g. number of overflowing samples, total power). The requantized samples must then be separated according to the frequency channel, and typically more than one incoming packet must be accumulated to generate a packet for the UDP. For example, having input and output packets of the same size, with 4 channels per packet and 16 bits/sample at the input, 8 input packets are required to generate 4 output packets.

Each frequency channel is then sent to a different IP address and/or port, in order to use the LFAA network switch structure to route a subset of the LFAA band for several stations to one correlator slice. The IP address/port is provided by the external UDP packetizer, and is selected by specifying an index in the `user_data` field of the AXI4 stream packet. The index is 0 for packets along the TPM chain, and 1–4 for packets to be sent to the TPM.

Channels must be sent to the Ethernet switch in a rotated order that depends on the station number, in order not to block the switch. For example station 0 could send packets for channels 0 to 7 in straight order, station 1 in the order [1–7, 0], station 2 in the order [2–7, 0, 1] and so on.

The UDP-IP sequencer must also be programmed with the number of frequency channels to send in parallel, the starting channel to transmit in the round robin sequence (to reduce/avoid switch conflicts), the idle time between packets, and the IP address/port for each of the channels.

6 Implementation

The top module is called `Station_beamformer`. It is highly configurable, even if at the moment most parameters can be modified only in a limited range. It interfaces with:

- The tile beamformer, using an AXI4 stream interface
- The SPEAD decoder (receiver) and formatter (transmitter), using an AXI4 stream interface supplemented by a SPEAD header structure
- The DDR memory, using a custom logical interface
- The control system, using an AXI4lite slave interface.

The top level entity is `station_beamformer`. It is parametrized with the generics:

Name	Type	Description	Default
		Input signal	
`g_in_data_w`	INTEGER	bits per sample (real) in input samples	12
`g_in_frame_length`	INTEGER	input frame length (number of channels)	192
`g_in_nof_frames`	INTEGER	Frames in input cornerturner	8
`g_values_per_sample`	INTEGER	2 polarizations, complex	4
		DDR	
`g_ddr_word_size`	INTEGER	Bits in DDR word (not necessarily all used)	64
`g_ddr_tmf`	INTEGER	Time multiplexing factor for DDR	8
`g_addr_w`	INTEGER	Total DDR address space	29
`g_row_w`	INTEGER	Row address space	14
`g_col_w`	INTEGER	Column address space	10
`g_bank_w`	INTEGER	8 DDR banks	3
`g_burst_len`	INTEGER	Number of memory words per read/write burst	32
		SPEAD in/out	
`g_tpm_nof_chans`	INTEGER	Frequency channels in a TPM output frame	4
`g_tpm_frame_len`	INTEGER	TPM frame length (samples)	256
`g_tpm_data_w`	INTEGER	bits per sample (real) in SPEAD stream	16
		Output frame	
`g_csp_nof_chans`	INTEGER	Frequency channels in a CSP output frame	1
`g_csp_frame_len`	INTEGER	CSP Frame length (samples)	2048
`g_csp_data_w`	INTEGER	Data width for CSP samples	8
`g_architecture`	STRING	Whether DDR part, SPEAD part or both	`ddr_spead`

The input and output signals belong to four clock domain:

- the DSP clock, for the input frames. In the TPM the DSP clock is 200 MHz.
- the DDR clock affects the DDR interface. For the iTPM the base DDR clock is 200 MHz, that together with a DDR time multiplexing factor of 8 allows for a data transfer rate of 1600 MT/s (800 MHz DDR clock). Even if the DDR clock has the same frequency of the DSP clock, it is not safe to assume they have a fixed phase relationship.
- the SPEAD interface clock. The SPEAD formatter could provide a cross boundary FIFO for the input and output frames. If this is not the case, this clock is the same used in the Ethernet interface, i.e. 156.25 MHz. If the formatter can accept different clock rates, a clock equal to the DDR clock is used.
- the AXI4lite clock, for the control interface.

The module ports are:

Name	Direction	Type	Description
`dsp_clk`	IN	`STD_LOGIC`	DSP input clock
`dsp_in_mosi`	IN	`t_axi4s_mosi`	and data
`ddr_clk`	IN	`STD_LOGIC`	DDR clock
`ddr_sosi`	OUT	`t_ddr_sosi`	and logical interface
`ddr_siso`	IN	`t_ddr_siso`	
`spead_clk`	IN	`STD_LOGIC`	SPEAD interface clock
`spead_rx_mosi`	IN	`t_axi4s_mosi`	SPEAD receiver signals
`spead_rx_miso`	OUT	`t_axi4s_miso`	
`spead_rx_frame_id`	IN	`STD_LOGIC_VECTOR(48)`	RX SPEAD counter
`spead_tx_mosi`	OUT	`t_axi4s_mosi`	SPEAD transmitter signals
`spead_tx_miso`	IN	`t_axi4s_miso`	
`spead_tx_frame_id`	OUT	`STD_LOGIC_VECTOR(48)`	TX SPEAD counter
`spead_tx_header`	OUT	`t_spead_header`	TX header data
`axi4_clk`	IN	`STD_LOGIC`	Control interface clock
`axi4_rstn`	IN	`STD_LOGIC`	and signals
`axi4lite_mosi`	IN	`t_axi4lite_mosi`	
`axi4lite_miso`	OUT	`t_axi4lite_miso`	

The input stream comes from the tile beamformer. It is composed of frames, of `g_in_frame_length` samples per frame (or less). Each frame is composed of 48 bit words, for 4 (complex, 2 pol.) samples, each of `g_in_data_w` bits.

The DDR is described by generics in the DDR group. Each memory access is composed of `g_burst_length` words of size `g_word_size`, with the interface joining together, in a single burst, requests with contiguous addresses. The Xilinx interface accepts requests for 8 words, i.e. for a single clock cycle in the multiplexed data bus (`g_ddr_tmf` words wide). The efficiency for such short bursts is very low, so we always transmit multiple requests (4 with the default shown above) in a burst.

The `t_ddr_sosi` and `t_ddr_siso` structures are described in the DDR interface library developed as part of this project, and summarized in the table below.

Field	Size	Use
`t_ddr_sosi`		
`req`	1	Request: '1' if transaction requested
`rnw`	1	1=read, 0=write
`adr`	27	Address, in row,column,bank order
`wdat_be`	64	byte enable for write
`wdat`	$64 * 8$	Write data (64 bytes, 8 memory words)
`autoprecharge`	1	Assert precharge at end of transaction
`t_ddr_siso`		
`rdy`	1	Interface ready
`rdat`	$64 * 8$	Read data (64 bytes, 8 memory words)
`rdat_vld`	1	Read data valid

A command is issued by raising `req`, and is accepted if both `req` and `rdy` are asserted.

The SPEAD interface uses an AXI4 stream supplemented by a `t_spead_header` structure, described in table 3.

6.1 Programming interface

The beamformer is controlled using a set of writeable registers. Most of the registers are write-only, i.e. they cannot be read back. All registers with bit fields can be read back, to allow for read-modify-write operations on a single bit field without the need of knowing the content of the remaining fields.

The actual interface is generate automatically from a XML register description. The procedure generates a VHDL structure with named fields corresponding to the registers, and a symbolic address mapping that is loaded in the VHDL design, used by the low level programming code. The register mapping is shown in table 4. Register address is subject to changes, but as all programming and VHDL coding use the symbolic names, this is transparent to the developer.

List of acronyms

ADC: Analog to Digital Converter

ADU: Analog to Digital Unit: the amplitude of one ADC quantization step

ALMA: Atacama Large Millimetre Array

CSP: Central Signal Processor

DDR: Double Data Rate: Implementations of DRAM using both clock edges for data transfer. DDR3 and DDR4 versions of the standard are used in the design

DRD: Document Requirements Descriptions

DSP: Digital Signal Processing

EMC: Electromagnetic Compatibility

EMI: Electromagnetic Interface

ENOB: Equivalent number of bits

FFT: Fast Fourier Transformation

FPGA: Field Programmable Gate Array

GBE: Giga Bit Ethernet

HDL: High Level Design Language

ICD: Interface Control Document

Addr.	Bits	R/W	ID	Description
0x00	31-0	R	`date_code`	Compile date (format YYYYMMDD)
0x04	2-0	RW	`control`	Control register
	0	RW	`reset`	Module reset
	1	RW	`first_tile`	Set if this is the 1st tile in the chain
	2	RW	`last_tile`	Set if this is the last tile in the chain
0x08	31-0	RW	`frame_timing`	Frame timing and length parameters
	11-0	RW	`int_block_len`	Integration block length, in CSP blocks
	23-12	RW	`int_block_ovl`	Integration block overlap, in CSP blocks
	26-24	RW	`frame_rate`	Time between SPEAD frames, in μs
	31-28	RW	`csp_frame_size`	CSP frame size, in TPM frames, minus 1
0x0c	15-0	W	`csp_scaling`	Scaling of the beamformed samples, before CSP requantizaton
0x10	31-0	W	`start_frame`	First frame processed
0x14	31-0	W	`last_frame`	Last frame processed
0x18	31-0	R	`current_frame`	Current frame received by the tile beamformer
0x20	31-0	W	`ref_epoch_lo`	Reference epoch, lower 32 bits
0x24	15-0	W	`ref_epoch_hi`	Reference epoch, higher 16 bits
0x28	31-0	W	`start_time`	Time of 1st sample in integration, in ns from reference epoch
0x2c	31-0	RW	`frame_id`	Frame ID parameters
	7-0	RW	`antenna_index`	Number of antennas in the beam
	15-8	RW	`tpm_id`	ID number of the TPM
	31-16	RW	`station_id`	ID number of the station

Table 4: Control registers for the station beamformer module

IICD: Internal Interface Control Document

INAF: National Institute for Astrophysics

I/O: Input/Output

LFAA: Low Frequency Aperture Array Element or Consortium

MATLAB: MATLAB simulation language and application

M&C: Monitor and Control

NRC: National Research Council (Canada)

OX: Oxford University

PIP: Physical Implementation Proposal

RFI: Radio Frequency Interference

RS: Requirement Specification

SDP: Science Data Processing

SDRAM: Syncronous Dynamic Random Access Memory: Standard for bursting, fast memory. DDR3 and DDR4 implementations of SDRAM are used in the design

SPEAD: Standard Protocol for Exchange of Astronomic Data

SKA: Square Kilometre Array

SKAO: SKA Organization (or office)

SW: Software

TBC: To be confirmed

TBD: To be decided

TPM: Tile Processing Module

WBS: Work Breakdown Structure

References

[1] UltraScale Architecture-Based FPGAs Memory Interface Solutions v6.0, *LogiCORE IP Product Guide*, Vivado Design Suite, **PG150** (2014).
`http://www.xilinx.com/support/documentation/ip_documentation/mig/v6_0/pg150-ultrascale-mis.pdf`

[2] G. Comoretto: "LFAA timing and observation sequence", in preparation (2016).

[3] G. Comoretto: "LFAA Tile Beamformer structure", INAF - Osservatorio Astrofisico di Arcetri Internal Report no. 2/2015

[4] "SKA1 LFAA to CSP ICD", SKA Office document no. SKA-TEL-SKO-0000142 (2015)

[5] J. Manely, M.Welz, A.Parson, S. Ratcliffe, R. van Rooyen: "SPEAD: Streaming Protocol for Exchanging of Astronomical Data", SKA-SA internal memo, 2010/10/07

Contents

List of Tables

List of Figures