Data Path Interface Description

	Organisatie / Organization	Datum / Date
Auteur(s) / Author(s): Eric Kooistra	ASTRON	
Controle / Checked: Andre Gunst	ASTRON	
Goedkeuring / Approval: Andre Gunst	ASTRON	
Autorisatie / Authorisation: **Handtekening / Signature**	ASTRON	

Distribution list:

Group:	Others:
Andre Gunst Daniel van der Schuur Rajan Raj Thilak Jonathan Hargreaves (JIVE) Jing Xiang	

Document history:

Revision	Date	Author	Modification / Change
0.1	2010-07-12	Eric Kooistra	Ready for review.
0.2	2010-08-27	Eric Kooistra	The DP PHY frame interface now includes one valid idle word before the SFD. Use dp_fifo_fill instead of dp_frame_scheduler. Added multi test bench tb_tb_dp_packetizing. Added description of the flush feature of dp_frame_rd and the dedicated test bench tb_dp_frame_rd.
0.3	2010-11-04	Eric Kooistra	Use dp_fifo_sc instead of common_fifo_sc in the test bench.

Table of contents:

References:

1. “RSP Firmware Functional Specification”, LOFAR-ASTRON-SDD-001, W. Lubberhuizen
2. “RSP Firmware Design Description”, LOFAR-ASTRON-SDD-018, W. Lubberhuizen, Wietse Poiesz, Eric Kooistra
3. “Avalon Interface Specifications”, mnl_avalon_spec.pdf, www.altera.com
4. “Specification for module interfaces using VHDL records”, ASTRON-RP-380, Eric Kooistra
5. “About the ready signal of the streaming interface”, ASTRON-RP-382, Eric Kooistra
6. http://www.easics.com/webtools/crctool

Terminology:

ADC	Analogue to Digital Converter
Beamlet	A small beam, spanning one subband
BF	Beam Former
BRC	Boolean CRC (= err)
BS	Block Sync
BST	Beamlets Statistics
CRC	Cyclic Redundancy Check
Crosslet	A source for cross correlation, spanning one subband of a particular antenna
DP	Data Path
DSP	Digital Signal Processing
DST	Destination
DUT	Device Under Test
eof	End Of Frame
eop	End Of Packet
err	Error signal
FFT	Fast Fourier Transform
FIFO	First In First Out
FPGA	Field Programmable Gate Array
FSN	Frame Sequence Number
HDL	Hardware Description Language (typically VHDL or Verilog)
IC	Intergrated Circuit
I/O	Input/Output
IP	Internet Protocol, Intellectual Property
LOFAR	Low Frequency Array
MAC	Media Access Control
Nof	Number of
PFB	Poly-phase Filter Bank
PFS	Pre Filter Structure
PFT	Pipelined FFT
PHY	Physical layer
RAD	Ring Adder (module in LOFAR)
RSP	Remote Station Processing (board in LOFAR)
RTL	Register Transfer Level
SRC	Source
SFD	Start of Frame Delimiter
SISO	Source In Sink Out
Slice	Block of data, e.g. N real input samples, N/2 complex subband samples, < N/2 complex beamlet samples, where N corresponds to FFT size and the down sample factor of the PFB
sof	Start Of Frame
sop	Start Of Packet
SOPC	System On a Programmable Chip (Altera)
SOSI	Source Out Sink In
SS	Subband Select
SST	Subband Statistics
ST	Statistics
Subband	2/N-th of the input spectrum
VHDL	Very High Speed IC HDL
UDP	User Datagram Protocol
UTC	Coordinated Universal Time
XST	Crosslet Statistics

1 Introduction

1.1 Purpose

The Data Path (DP) interface provides definitions for packetizing and unpacketizing blocks of streaming data. The DP interface definition is suited for stream alignment and for point-to-point links. This document describes the DP interface and also the components in the VHDL library, called 'DP', that facilitate the use of the DP frames. The DP library is located in the UniBoard RadioNet FP7 SVN repository at:

https://svn.astron.nl/UniBoard_FP7/UniBoard/trunk/Firmware/modules/dp/

The DP interface definitions originate from the RAD module that is used in the LOFAR stations to transfer frames between the RSP boards.

The DP library contains several more stream processing components. These components are not described here but in [5].

1.2 Background

The need to packetize data arises when data has to be transferred between nodes or sites, or when it has to be stored for offline processing later on. Packetizing data offers for example:

- The start of frame delimiter (SFD) in a frame allows finding the start of a frame without the need of a separate start of frame (SOF) signal in parallel.
- The header of a frame can contain an UTC time stamp which makes the processing of the data latency independent.
- Adding a CRC to a frame allows for error detection on links between nodes.

Inside a node the processing of data typically also occurs in blocks. Therefore inside the FPGA also some level of packetizing is used. However inside the FPGA the header and error information can be represented by parallel signals along with the data for easier handling:

- sync
- fsn[]
- valid
- sof
- eof
- err / brc

These signals (except for sync) are also supported by the Avalon streaming interface [3] source-output-sink-input (SOSI) signals [4].

The DP packetizing components do not (yet) support the back pressure via the ready signal [5], so ready='1'. The ready signal is the Avalon streaming interface source-input-sink-output signal (SISO). Back pressure support is not necessarily needed, because temporary rate differences between the source and the sink can be handled with FIFOs and by throttling the output rate. Long term rate differences (which would cause FIFO overflow) can not occur because then the sink is too slow anyway.

2 DP packetizing levels

Packetizing data concerns several levels of marking blocks of streaming data. The data path (DP) interface describes three levels of packetizing data:

- The DP data interface with data block control signals
- The DP frame interface with a frame sequence number field for frame alignment and an error field
- The DP PHY frame interface for transport over point to point physical links and with a CRC

The following sections describe these packetizing levels of the DP interface.

2.1 The DP data interface

2.1.1 Time series data

In a data path the first digital source is typically an ADC. From an ADC the data first consists of a continuous stream of samples whereby every sample clock cycle contains valid data. One of the first things to do is to couple real time information to the ADC samples by means of an external PPS (pulse per second) signal. The assumption is that the sample clock is in lock with the PPS and therefore also with UTC, if the PPS is in lock with UTC. Then the external PPS can be used to start an internal sync. After that, with for example a 200 MHz sample clock, this sync signal pulses high for one clock cycle every 200M samples. This PPS to sync function is called Block Sync (BS) in [2]. The sync pulse can be regarded as a primitive precursor of the 32-bit UTC time stamp that can be transported as data header information at a higher packetizing level e.g. via UDP/IP packets.

The data interface consists of the following signals: sync, valid and data, for both the inputs and the outputs as shown in Figure 1.

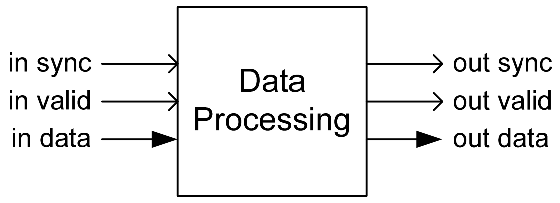

Figure 1: Data path processing block interface [2]

The data is the signal that will contain the actual data values that are fed into the processing blocks, and are sent out of the processing blocks. Data can have any width needed to represent the values that have to be transported on it. Data can consist of one or more concurrent values at a given clock cycle. For example, ADC samples can be transported using just one instance of data, while complex results produced by the frequency separation would require 2 data instances (real and imaginary).

The valid is a boolean value, that is used to indicate if the data signal contains data or is idle for every clock cycle. By using the valid, the interface is not limited to ‘always on’, there can be idle spaces between data. Hence the data path processing is data driven meaning that it only acts when a clock cycle carries a sample as indicated by the valid signal.

The sync signal is also a boolean value indicating that the next data valid is the first sample of a sync interval. The sync is passed along with the data to maintain synchronisation with external time, independent of digital processing delays. If for some data processing block the input sync occurs out of order with its internal counters, then the input sync must be followed (this indicates some internal logic error, so should never occur).

The processing latency does not disturb the synchronisation, but it can impact the required size of FIFOs when different data paths need to be combined, especially if the data has to go through many hops from node to node. For example in LOFAR the beam former [1] works as an adder chain, whereby each node adds its local beam data to the sum beam data that is transported from node to node. The downstream nodes in the beam former chain then have to hold their local beam data until the sum has arrived.

Figure 2 shows the general definition of valid data and the sync pulse. Typically the sample clock rate equals the system clock rate, so then once high the data valid will stay high for all clock cycles. The sync pulse indicates that the next valid data is the first data sample of the next sync interval. Hence signals *a*, *b*, *c* and *d* in Figure 2 are all allowed sync signals.

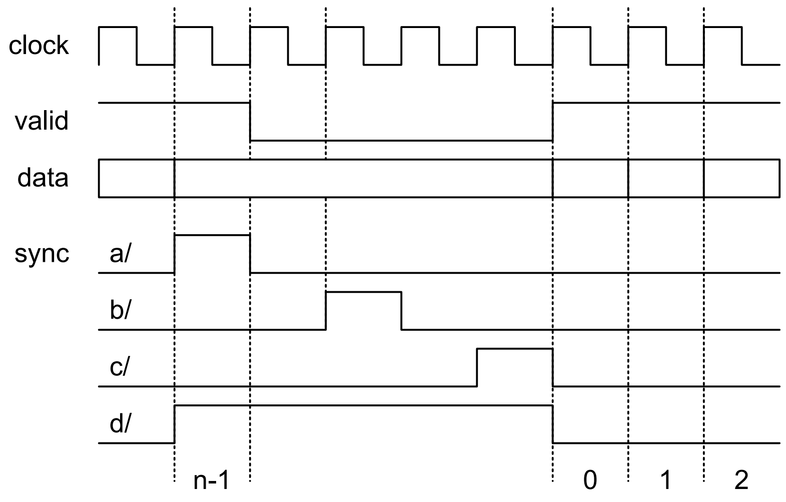

Figure 2: Timing of DP data interface signals valid and sync [2]

2.1.2 Block series data

One of the first processing steps in a data path is a PFB (Poly Phase Filterbank). After the PFB the time series data becomes block series data whereby each block contains the number of spectral subband samples that depends on the size of the FFT in the filterbank. After the PFB still all clock cycles contain valid subband data as shown at top of Figure 3.

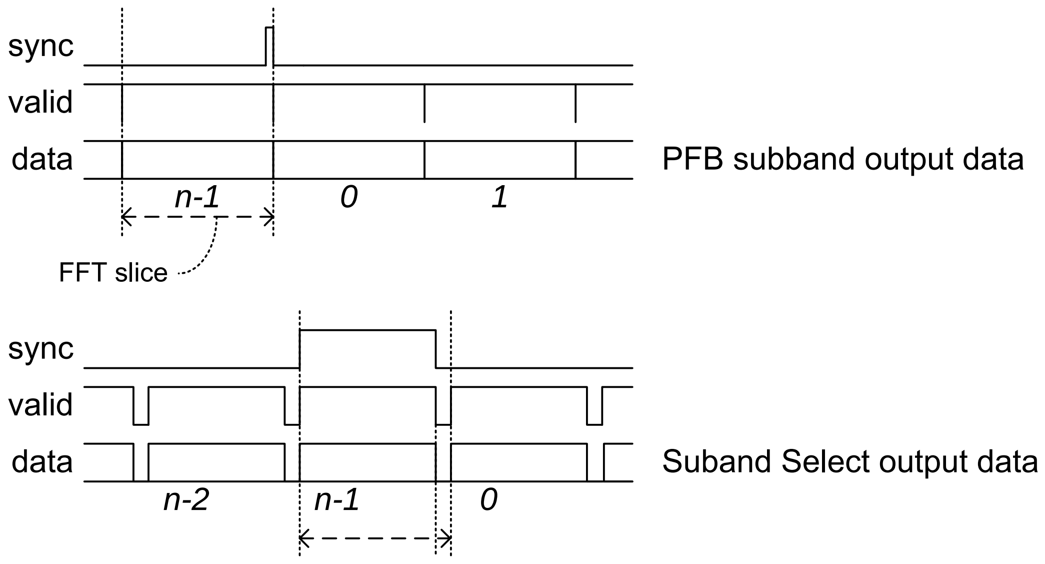

Figure 3: Timing of PFB subband data (top) and subband select data (bottom) [2]

Typically not all subbands have to be processed further. Therefore the next step is to select a set of subbands for further processing (e.g. beam forming and correlation) as shown at bottom Figure 3. Note that each subband sample after the subband select has its own sync pulse, because they all belong to the same FFT slice. By selecting somewhat less subbands than that fit in an FFT slice the subband selection allows to create some data invalid cycles between data valid blocks. This gap of data invalid cycles can be used to add a header and a tail (CRC) to the data blocks to create data frames.

Hence for data path processing the PFB and subband select (SS) form a natural way of creating blocks of data with gaps in between. Given an arbitrary block size other, more digital logic, approaches for creating gaps in a data stream are e.g.:

- Pass the data on via two streams, whereby each stream alternately contains a valid block
- Pack the data into larger words and lead these through a single clock FIFO to remove the invalid cycles during the block
- Write the data into a dual clock FIFO and read the FIFO with a somewhat faster clock

2.1.3 Block control signals

When processing blocks of data inside the FPGA then it is useful to have a sof (start of frame, block) and an eof (end of frame, block) signal to avoid having to run a block sample counter at each processing stage. The sof and the eof are valid when the valid is active. In addition the blocks need a frame sequence number (FSN) and an error signal.

The FSN can be regarded as a timestamp that is not a unique UTC time stamp yet, because it has an arbitrary width (typically the same as the data width). The MSbit of the FSN marks the sync and the lower bits of the FSN count the data blocks and are reset to zero at the sync. Within a sync interval the FSN count may wrap several times. However the FSN count range should be sufficient to allow two data streams to be aligned based on their FSN being equal. The FSN signal is valid at the sof.

Inside an FPGA the error signal is typically not needed, because the design is expected to be functionally correct and the FPGA logic is assumed to work error free. However the error signal is useful inside the FPGA to propagate e.g. CRC errors that occurred on received frames from outside links. Rather than passing on a true CRC inside the FPGA the error information is passed on as a boolean CRC signal (called err or brc), where by 0 is OK and > 0 indicates an error. If necessary the error signal can be an error number. The error signal is valid at the eof.

Figure 4 shows all DP data interface signals. Note that data[], fsn[], and err[] preferably hold their value until they have to change again.

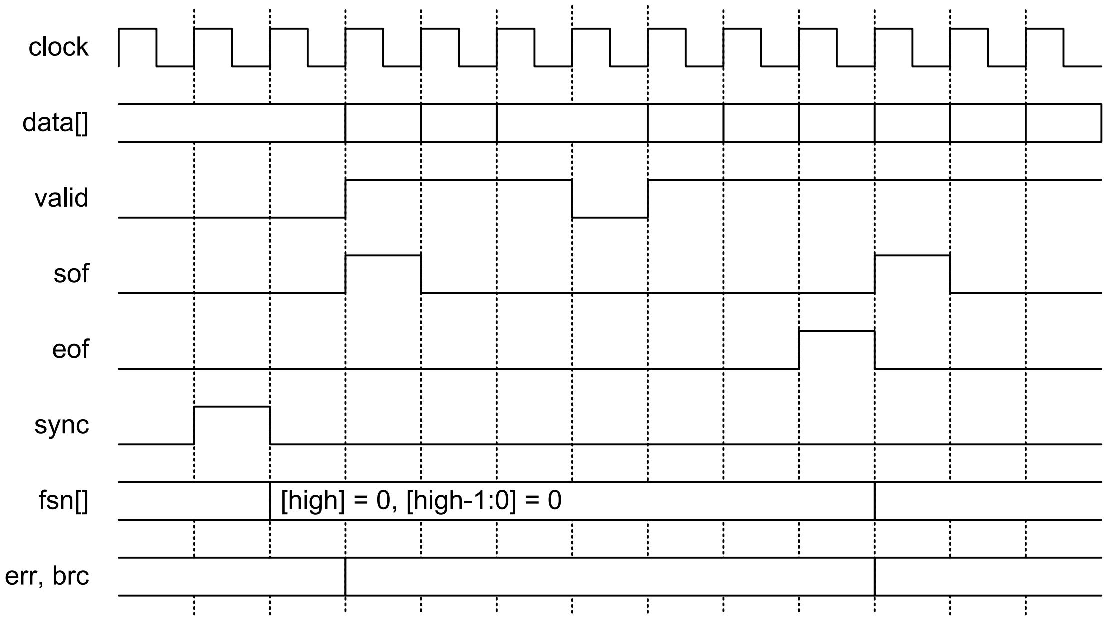

Figure 4: Timing of all DP data interface signals including sof, eof, fsn[] and err signals [2]

2.1.4 Packing and unpacking data

If the data width on an external link does not fit the user data width then it is necessary to pack and unpack the data to be able to make efficient use of the capacity of the external link. Figure 5 shows how user data words gets packed into PHY link words that have a 3/2 larger width.

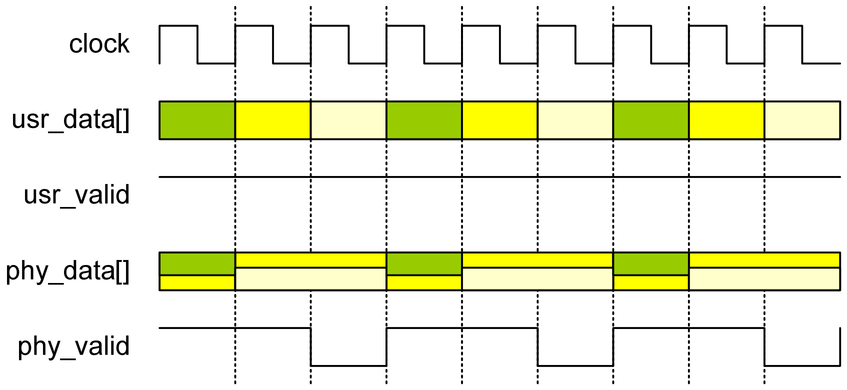

Figure 5: Packing data with 3 user words per 2 PHY link words [2]

2.2 The DP frame interface

Figure 6 shows the definition of the DP frame and the timing of the DP frame interface signals. The DP data interface signals of Figure 4 are used to packetize the data into the DP frame format of Figure 6. Similar a DP frame can be unpacketized into the DP data interface signals. The DP frame consists of the data words preceded by the FSN word and appended with the error or brc word. Although Figure 6 shows valid active high during the entire frame, not every clock cycle has to carry valid data during a frame.

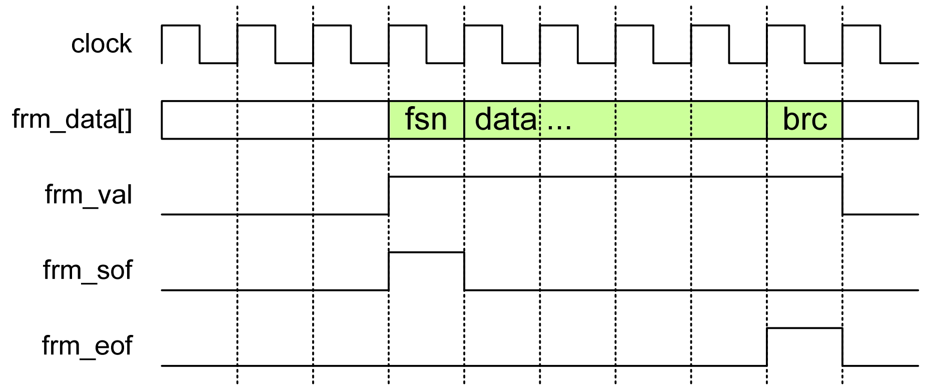

Figure 6: Definition of the DP frame and timing of the DP frame interface signals [2]

The DP frame definition allows alignment of streams (using the FSN) and error propagation (via the brc). Data processing operations can directly take pace on the data in the DP frame; often it is the not necessary to un-frame the DP frame to the DP data interface format.

2.3 The DP PHY frame interface

To transfer the DP frames over a PHY link all parallel block control signals need to be passed on together with the data in a single PHY data stream. This results in the following definitions:

- The sof is passed on as a combination of an idle word and a start of frame delimiter (SFD) word. In the ‘RAD’ library in [2] the SFD was called ‘magic’.
- The fsn[] passed on after the SFD and then followed by the PHY data words.
- The frame length that is marked by the eof is fixed per SFD word, so there is no need for a length field
- The err or brc is propagated via a true CRC word after the data if it is OK or the inverted CRC if indicates an error.
- For a simple PHY link the data valid has to be active during the entire frame. However if the PHY link supports e.g. 8b/10b encoding, then the data valid can be passed on as a unique PHY code word. This then allows a frame to have data invalid cycles between SFD and CRC, because they can be recognized as invalid at the receiver.

Figure 7 shows the definition of the DP PHY frame and the timing of the DP PHY frame interface.

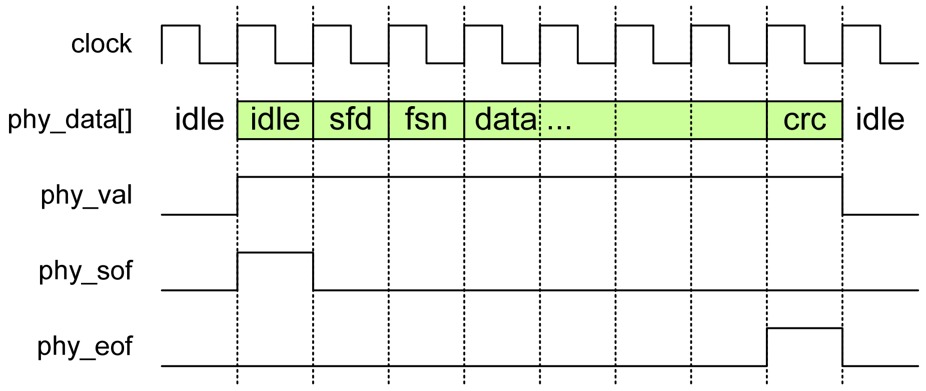

Figure 7: Definition of the DP PHY frame and timing of the DP PHY frame interface signals [2]

The PHY sof and eof are not transferred as such on the PHY link but internally in the FPGA it is useful to have them. The FSN is resized sign-extended to the DP PHY frame data width to preserve the MSBit sync information. The DP PHY frame interface includes one valid idle word before the SFD. This ensures that subsequent DP functions will preserve the idle-SFD word combination for detecting the frame. Other invalid words between DP PHY frames may be idle words, but this is not required. In the 'RAD' library in [2] the phy_sof marked the SFD, now it marks the idle word just before the SFD as shown in Figure 7.

2.3.1 Comparison with the Ethernet frame

Table 1 compares the DP PHY frame defined in Figure 7 with the definition of the Ethernet frame.

Ethernet field	**DP PHY field**	**Comment**
preamble	idle	-
SFD	SFD	In DP PHY the SFD can also be used to distinguish different frame streams
DST MAC	-	DP PHY is point to point so no addressing
SRC MAC	-	DP PHY is point to point so no addressing
Type/length	SFD	In DP PHY the SFD can be used as a frame type and the length is then fixed
-	FSN	Defined in DP PHY to support data frame alignment between two streams
Data	Data	DP PHY supports arbitrary data widths
CRC	CRC	In DP PHY the CRC width is resized to the data width

Table 1: Comparison of the DP PHY frame with the Ethernet frame

3 DP packetizing components

This section describes the packetizing components in the DP library. More detailed information can be found in the VHDL source files in the Uniboard_FP7 SVN repository.

Most of the DP packetizing components were ported from LOFAR where they were in the 'RAD' library [2]. For CRC calculations code from the Easics web site [6] is used.

3.1 Overview of the DP library

Figure 8 on the next page show all data packetizing components from the DP library in a single test bench called tb_dp_packetizing.vhd.

The procedure proc_dp_gen_block_dat() generates a data signal stream with counter data conform Figure 2.

The component dp_phy_link models the PHY link between two FPGA nodes, e.g. a transceiver link. The latency of the link is modelled by delay g_latency. If the real link has e.g. 8b/10b coding, then it can transmit fill words. This is modelled by g_valid_support = TRUE. When the input valid is low then a fill word is transmitted that can not occur in the user data. At the receiver the fill words can be passed on through output valid low. If the real link has no coding, then this is modelled by g_valid_support = FALSE and output valid will be high always.

The procedure proc_dp_verify_data () verifies that the output data equals the original user data. It does this by simply checking that the output data is a continuous stream of counter values with correct increment, else an error message is reported.

A VHDL assert statement checks the received error field to report if there occurred and CRC error. A CRC error can be forced by running the test bench via 'do tb_dp_packetizing.do' in the Modelsim simulator.

Different variations of the tb_dp_packetizing test bench can easily be simulated in parallel by means of the tb_tb_dp_packetizing test bench. This multi test bench has multiple instances of tb_dp_packetizing with different generic mappings to try different word widths and packing ratios.

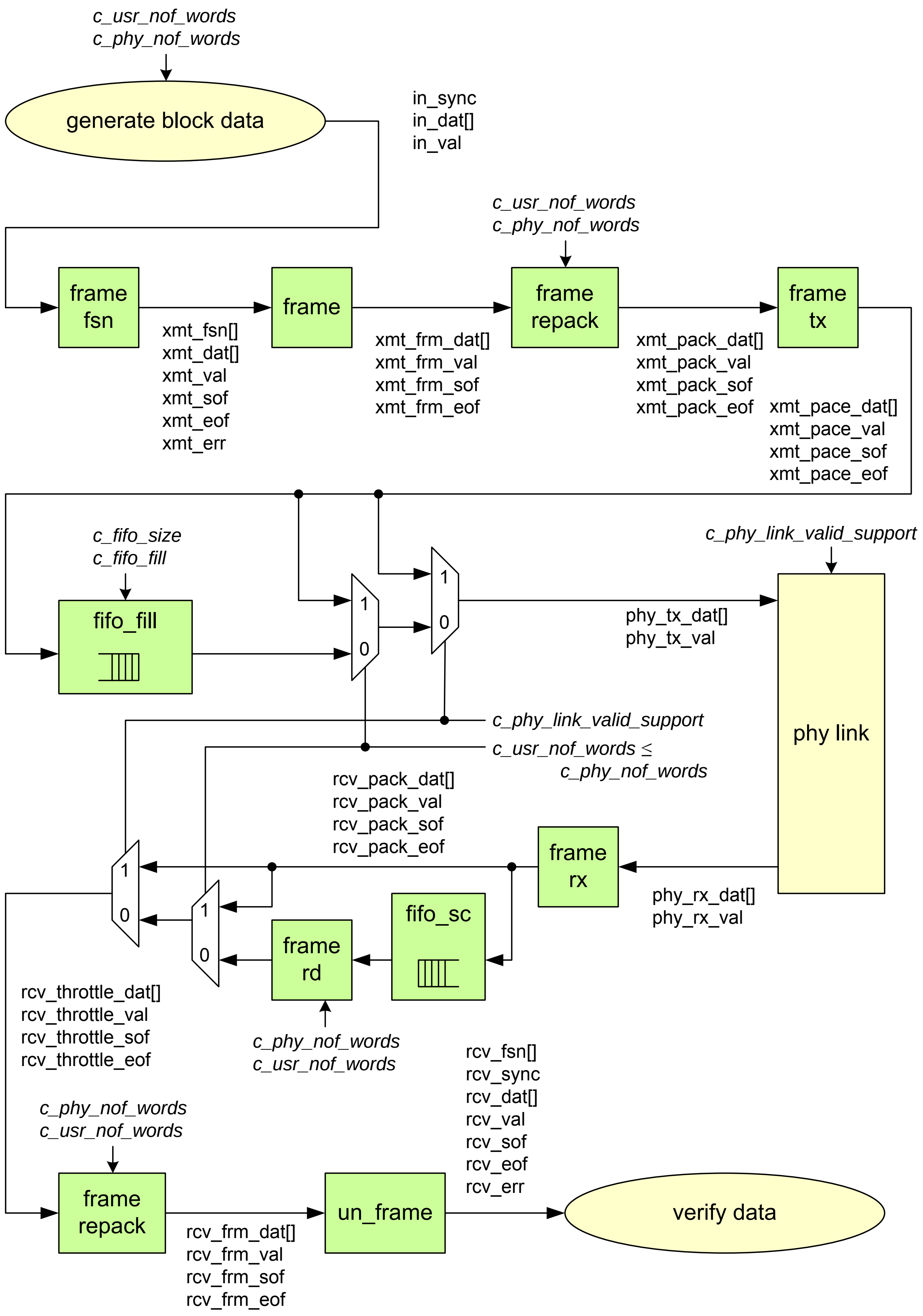

Figure 8: Test bench for DP packetizing and unpacketizing

3.2 DP library components

The following sections describe the DP library components that are used in the tb_dp_packetizing test bench to show and verify the DP packetizing components.

3.2.1 dp_frame_fsn

The dp_frame_fsn component determines start and end of data blocks conform Figure 4 and introduces the data path FSN with sync and block sequence number information. The number of valid data words per block is set by g_block_size. The minimum gap between blocks is 4 cycles to allow space for the SFD, FSN and CRC words of a DP PHY frame and one idle word in between DP PHY frames. The sync timing of the dp_frame_fsn requires at least one invalid cycle before the data block starts, so it does not allow schemes c and d in Figure 2.

3.2.2 dp_frame

The dp_frame component converts the DP data interface signals into the DP frame format conform Figure 6.

3.2.3 dp_unframe

The dp_unframe component does the inverse of dp_frame. It converts a DP frame into the DP data signals conform Figure 4.

3.2.4 dp_frame_repack

The dp_frame_repack component performs frame data packing or unpacking at DP frame level. Figure 9 shows the block diagram of dp_frame_repack. Internally it operates at DP data level using dp_repack as described in section 2.1.4.

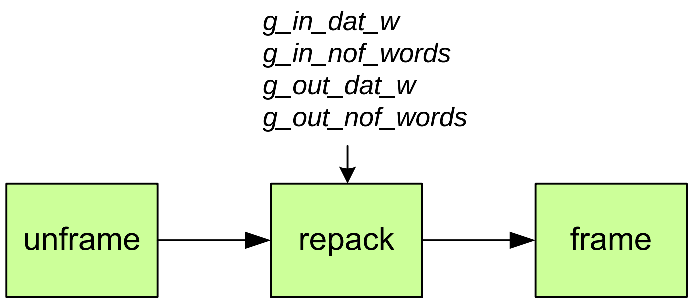

Figure 9: Block diagram of dp_frame_repack

The dp_repack component repacks g_in_nof_words of width g_in_dat_w into g_out_nof_words of width g_out_dat_w whereby g_in_nof_words * g_in_dat_w ≤ g_out_nof_words * g_out_dat_w. The dp_repack works both as packer and as unpacker. If g_in_nof_words=g_out_nof_words then the input data is simply passed on directly to the output data via wires. The number of input words indicated by the input sof and input eof must be a multiple of g_in_nof_words, because padding is not supported. The input valid active duty-cycle must be ≤ g_in_nof_words/g_out_nof_words, because the dp_frame_repack component does not buffer the input.

3.2.5 dp_frame_tx

The dp_frame_tx component converts a DP frame into a DP PHY frame conform Figure 7

3.2.6 dp_frame_rx

The dp_frame_rx component detects a DP PHY frame and converts it into a DP frame conform Figure 6. The dp_frame_rx uses the combination of idle and SFD to detect a frame. Therefore DP PHY frames must be separated by at least one idle word. The idle word is defined as 0x…555. The default SFD is defined as 0x…554, but more SFD can be defined to distinguish different DP PHY streams that share a PHY link.

3.2.7 dp_frame_scheduler

The dp_frame_scheduler component can multiplex 1 or more DP PHY frame streams on to one PHY link. Each input is buffered by a FIFO and selected in a round robin scheme. The dp_frame_scheduler is not used in the tb_dp_packetizing test bench, but can be verified using the tb_dp_frame_scheduler test bench. However dp_frame_scheduler is legacy derived from LOFAR rad_frame_scheduler. If a multiplexer is needed then instead of dp_frame_scheduler it is now better to use dp_mux [4], because dp_mux supports back pressure and passes the input port number on as channel number. The tb_dp_frame_scheduler test bench can also verify dp_mux, this shows that dp_mux can replace dp_frame_scheduler. By means of the multi test bench tb_tb_dp_frame_scheduler it is possible to verify dp_mux and some variations of dp_frame_scheduler in a single simulation run.

3.2.8 dp_fifo_fill and dp_fifo_sc

The dp_fifo_fill and dp_fifo_sc [4] are used in the test bench if the user data needs to be packed into less PHY data words. That then causes data invalid cycles in the DP PHY frame stream which can not be passed on via the PHY link if the PHY link does not support data valid. The FIFO in the dp_fifo_fill is then filled to a sufficient level before the FIFO is read, to ensure that the transmitted DP PHY frame will have data valid asserted during the entire frame. The dp_fifo_sc provides streaming interface signals to a common fifo_sc.

3.2.9 dp_frame_rd

The dp_frame_rd component can read a frame from a FIFO at a throttled rate ≤ 1. In the tb_dp_packetizing test bench a rate < 1 is needed if the user data was packed into less PHY data words. The throttled rate then equals the packing ratio.

The dp_frame_rd can also flush a frame from the FIFO, output valid, sof and eof then remain low. A frame that does not start with a sof gets flushed too, this behaviour is verified with the dedicated tb_dp_frame_rd test bench.

4 Conclusion

The DP library provides a set of components that support packetizing data according to the DP interface definitions.

- The DP data interface definition provides the signals for data block control.
- The DP frame interface definition facilitates stream alignment by means of an FSN.
- The DP PHY frame interface definition suits point to point PHY links with arbitrary data width and error detection or propagation using a CRC.

The test bench tb_dp_packetizing shows an end to end application example and verifies the DP packetizing components.