

DIAG Module Description

	Organisatie / Organization	Datum / Date
Auteur(s) / Author(s): Eric Kooistra Harm Jan Pepping	ASTRON	24 February 2012
Controle / Checked: Andre Gunst	ASTRON	
Goedkeuring / Approval: Andre Gunst	ASTRON	
Autorisatie / Authorisation: Handtekening / Signature Andre Gunst	ASTRON	

© ASTRON 2011
All rights are reserved. Reproduction in whole or in part is prohibited without written consent of the copyright owner.

UniBoard

Doc.nr.: ASTRON-RP-1313
Rev.: 0.1
Date: 07-03-2012
Class.: Public

Distribution list:

Group:	Others:
Andre Gunst Eric Kooistra Daniel van der Schuur	Gijs Schoonderbeek Sjouke Zwier Harro Verkouter (JIVE) Jonathan Hargreaves (JIVE) Salvatore Pirruccio (JIVE)

Document history:

Revision	Date	Author	Modification / Change
0.1	2012-03-07	Harm Jan Pepping	Creation

Table of contents:

1	Introduction.....	5
1.1	Purpose	5
2	DIAG components	6
2.1	mms_diag_block_gen component.....	6
2.1.1	mms_diag_block_gen parameters	6
2.1.2	mms_diag_block_gen interface signals	6
2.1.3	mms_diag_block_gen software interface	7
2.1.4	mms_diag_block_gen mode of operation	8
2.1.5	mms_diag_block_gen design	9

Terminology:

DIAG	Diagnostics (VHDL module)
MM	Memory Mapped
ST	Streaming
UNB	https://svn.astron.nl/Uniboard_FP7/Uniboard/trunk/ , the Uniboard SVN repository

References:

1. 'DP Streaming Module Description', ASTRON-RP-382, Eric Kooistra
2. 'Detailed Design of the Digital Beamformer System for Apertif', ASTRON-RP-413, G. Schoonderbeek, A. Gunst, E. Kooistra
3. Digital Beamformer Module for APERTIF, ASTRON-RP-517, H.J. Pepping
4. \$UNB/Firmware/modules/common/
5. \$UNB/Firmware/modules/Lofar/st/

1 Introduction

1.1 Purpose

The DIAG module contains a set of components that can be used for performing diagnostics. The DIAG module is located in the Uniboard RadioNet FP7 SVN repository at \$UNB/Firmware/modules/Lofar/diag/.

2 DIAG components

2.1 mms_diag_block_gen component

The `mms_diag_block_gen` component is a packet generator that can send parameterized packets in the DP streaming format as defined in [1]. It is also possible to specify the period between two consecutive packets. The waveform data is stored in memory and can be uploaded via a mm interface.

2.1.1 mms_diag_block_gen parameters

The `mms_diag_block_gen` module has a list of generics that determine the number of outputs and the size of the waveform memory. The generics are listed in Table 1.

Generic	Type	Value	Description
<code>nof_output_streams</code>	POSITIVE	16	The number of data streams that the packet generator has to provide.
<code>buf_dat_w</code>	POSITIVE	32	Specifies the bitwidth of the memory that holds the waveform data.
<code>buf_addr_w</code>	POSITIVE	7	This number defines the address width of each waveform memory.
<code>sim</code>	BOOLEAN	FALSE	A generic to determine if the <code>mms_diag_block_gen</code> unit is used in simulation or used in synthesis. The <code>sim</code> generic is used to locate the files that contain the initial content for the waveform memories.

Table 1 `Mms_diag_block_gen` parameters

2.1.2 mms_diag_block_gen interface signals

The interface signals of the `mms_diag_block_gen` module are shown in Figure 1 and Table 2 lists a detailed overview of each signal.

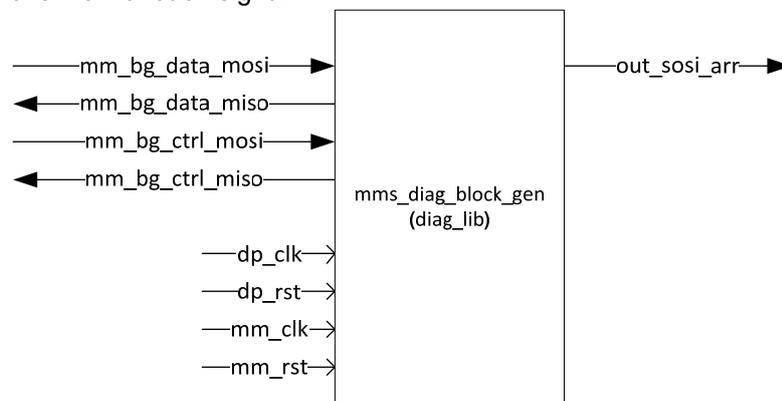


Figure 1 `mms_diag_block_gen` interface signals

Interface	Type	Size or Span	Description
<code>mm_bg_data_mosi</code>	<code>t_mm_mosi</code>	$\text{nof_output_streams} * 2^{\text{buf_addr_w}-1}$	The mosi part of the mm interface that contains the memory locations for the waveform data of every stream.
<code>mm_bg_data_miso</code>	<code>t_mm_miso</code>	$\text{nof_output_streams} * 2^{\text{buf_addr_w}-1}$	The miso part of the mm interface that contains the memory locations for the waveform data of every stream.

mm_bg_ctrl_mosi	t_mm_mosi	0x8	The mosi part of the mm interface that contains the control registers for the mms_diag_block_gen unit.
mm_bg_ctrl_miso	t_mm_miso	0x8	The miso part of the mm interface that contains the control registers for the mms_diag_block_gen unit.
out_sosi_arr	t_dp_sosi_arr	nof_output_streams	Array of data streams that contain the configured blocks with the waveform data.
dp_clk	std_logic	na	Datapath clock
dp_rst	std_logic	na	Datapath reset
mm_clk	std_logic	na	Memory mapped interface clock
mm_rst	std_logic	na	Memory mapped interface reset

Table 2 mms_diag_block_gen interface signals

2.1.3 mms_diag_block_gen software interface

The software interface of the mms_diag_block_gen module consists of two address spans: a control span and a data span. The control span is used to configure and control the streams and the data span is used to upload the data that represents the waveform.

2.1.3.1 mms_diag_block_gen control span

The control registers of the mms_diag_block_gen apply to all output streams. The mms_diag_block_gen control span contains the following registers:

Name	Address (words)	Bit Width	Read/Write	Description
enable	0x0	1	r/w	The waveform enable register. A logic '1' will enable the data streams, starting with the assertion of sop. A logic '0' will stop the outgoing data streams after the current packet is finished. The stream will end with the assertion of eop.
samples_per_packet	0x1	16	r/w	This register specifies the number of samples in a packet.
gapsize	0x2	16	r/w	Register represents the time between two consecutive packets. Time is expressed in dp_clk cycles.
blocks_per_sync	0x3	16	r/w	This register specifies the sync interval in number of blocks.
mem_low_address	0x4	8	r/w	Register contains the pointer to the first address of the memory from which the waveform generator reads the data.
mem_high_address	0x5	8	r/w	Register contains the pointer to the last address of the memory from which the waveform generator reads the data.
bsn_init_low	0x6	32	r/w	The 32 lsb's of the initial value of the bsn.
bsn_init_high	0x7	16	r/w	The 16 msb's of the initial value of the bsn.

Table 3 mms_diag_block_gen control span

2.1.3.2 mms_diag_block_gen data span

The mms_diag_block_gen data span contains the entrance to the memory locations that hold the data that represent the waveforms. The number of registers is defined by $\text{nof_output_streams} * 2^{\text{buf_addr_w}}$. The mms_diag_block_gen module provides complex data to the output-streams and therefore complex data is stored in the memory. The format is shown in Figure 2.

31

0

data.imaginary[15..0]	data.real[15..0]
-----------------------	------------------

Figure 2 Waveform data format

The registers of the mms_diag_block_gen data span are listed in Table 4. The shown registers are based on a mms_diag_block_gen instantiation where nof_output_streams = 16, buf_dat_w = 32 and buf_addr_w = 7. Note that not all registers are listed in order to save space.

Name	Address (words)	Bit Width	Read/Write	Description
bg_str0_d0	0x0	32	r/w	Complex data word 0 of stream 0.
bg_str0_d1	0x1	32	r/w	Complex data word 1 of stream 0.
bg_str0_d2	0x2	32	r/w	Complex data word 2 of stream 0.
-----	-----	---	----	-----
bg_str0_d126	0x7E	32	r/w	Complex data word 126 of stream 0.
bg_str0_d127	0x7F	32	r/w	Complex data word 127 of stream 0.
bg_str1_d0	0x80	32	r/w	Complex data word 0 of stream 1.
bg_str1_d1	0x81	32	r/w	Complex data word 1 of stream 1.
bg_str1_d2	0x82	32	r/w	Complex data word 2 of stream 1.
-----	-----	---	----	-----
bg_str1_d126	0xFE	32	r/w	Complex data word 126 of stream 1.
bg_str1_d127	0xFF	32	r/w	Complex data word 127 of stream 1.
bg_str2_d0	0x100	32	r/w	Complex data word 0 of stream 2.
bg_str2_d1	0x101	32	r/w	Complex data word 1 of stream 2.
-----	-----	---	----	-----
-----	-----	---	----	-----
-----	-----	---	----	-----
bg_str14_d126	0x77E	32	r/w	Complex data word 126 of stream 14.
bg_str14_d127	0x77F	32	r/w	Complex data word 127 of stream 14.
bg_str15_d0	0x780	32	r/w	Complex data word 0 of stream 15.
bg_str15_d1	0x781	32	r/w	Complex data word 1 of stream 15.
bg_str15_d2	0x782	32	r/w	Complex data word 2 of stream 15.
-----	-----	---	----	-----
bg_str15_d126	0x7FE	32	r/w	Complex data word 126 of stream 15.
bg_str15_d127	0x7FF	32	r/w	Complex data word 127 of stream 15.

Table 4 mms_diag_block_gen data span

2.1.4 mms_diag_block_gen mode of operation

The way to operate the mms_diag_block_gen module is straight forward. First upload the data of the waveforms, then configure the stream and finally enable the streams by writing a logic '1' to the "enable" register. Figure 3 shows a diagram that shows the timing of a single output stream. The stream is configured with the following values:

```
enable = 1
samples_per_packet = 8
blocks_per_sync = 2
gapsize = 8
mem_low_adrs = 0
mem_high_adrs = 11
bsn_init_low = 41
bsn_init_high = 0
```

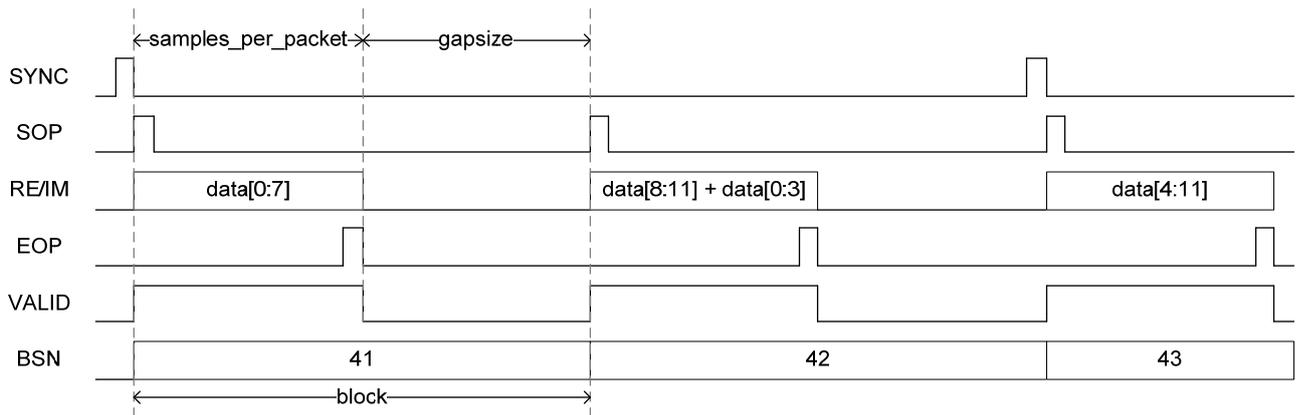


Figure 3 Timing diagram mms_diag_block_gen

2.1.5 mms_diag_block_gen design

A schematic overview of the mm_diag_block_gen design is shown in Figure 4 (nof_output_streams = 4). The common_mem_mux (common_lib) module is used to connect multiple dual port rams in case the mms_diag_block_gen module is configured to have more than one output stream (in Figure 4 only the first element of the array is drawn). A dual ported ram instance (also from the common_lib) functions as memory buffer for the data that will be send in the packets.

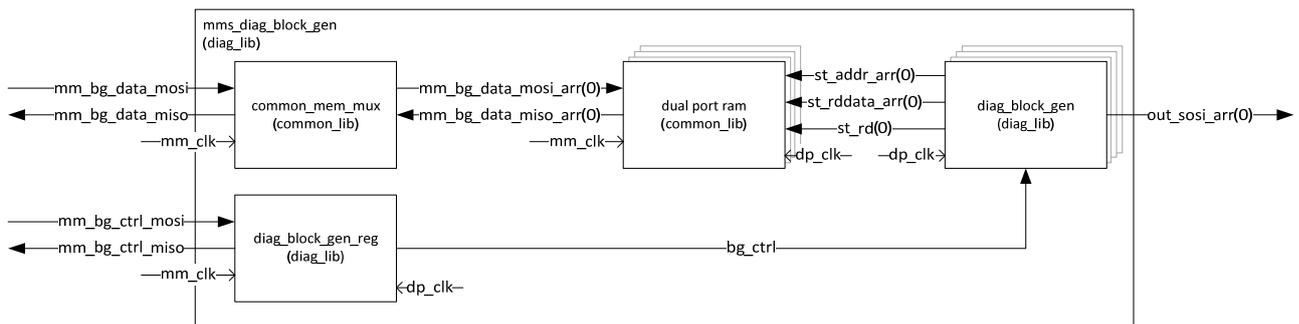


Figure 4 mms_diag_block_gen schematic overview

The diag_block_gen (diag_lib) responds to the settings of the bg_ctrl input port and when enabled it reads data from the dual ported ram and composes blocks on the out_sosi_arr that complies with the DP streaming format. The diag_block_gen_reg unit receives the control values via the mm interface and provides these values to the bg_ctrl record. The bg_ctrl record is defined in the diag_pkg (diag_lib).

