

UniBoard transceiver module

	Organisatie / Organization	Datum / Date
Auteur(s) / Author(s): Daniel van der Schuur	ASTRON	24 September 2012
Controle / Checked: Eric Kooistra	ASTRON	
Goedkeuring / Approval: Andre Gunst	ASTRON	
Autorisatie / Authorisation: Handtekening / Signature Andre Gunst	ASTRON	

© ASTRON 2012
All rights are reserved. Reproduction in whole or in part is prohibited without written consent of the copyright owner.

UniBoard

Doc.nr.: ASTRON-RP-449
Rev.: 0.5
Date: 24-09-2012
Class.: Public

Distribution list:

Group:	Others:
Andre Gunst Eric Kooistra Harm-Jan Pepping Jonathan Hargreaves (JIVE) Salvatore Pirruccio (JIVE)	Gijs Schoonderbeek Sjouke Zwier Harro Verkouter (JIVE)

Document history:

Revision	Date	Author	Modification / Change
0.1	2011-01-31	Daniel van der Schuur	Draft
0.2	2011-03-03	Daniel van der Schuur	Added sections on the ST interface
0.3	2011-11-23	Daniel van der Schuur	Updated complete tr_nonbonded module
0.4	2011-11-28	Daniel van der Schuur	Minor changes.
0.5	2012-09-24	Daniel van der Schuur	Updated due to addition of mms_diagnostics and RX an TX alignment FSMs.

Table of contents:

1	Introduction.....	5
1.1	Purpose	5
1.2	Performance and limitations	5
1.3	Module overview.....	5
2	Hardware interface	6
2.1	Clock and reset signals	6
2.2	Parameters	7
2.3	Interface signals	7
3	Software interface	8
3.1	mms_diagnostics	8
3.2	tr_nonbonded.....	8
4	Design	10
4.1	Clock domains	10
4.2	Parameters	11
4.3	Interface signals	11
5	Implementation.....	14
5.1	Architecture.....	14
5.2	ALTGX Megafunction	14
5.2.1	Alignment of received data	15
6	Reference design	17
7	Verification.....	18
8	Validation.....	19
8.1	Design revisions	19
8.1.1	Front node and/or back node revisions	19
8.1.2	Back node only revisions	19
8.2	Dedicated C application.....	20
8.2.1	JTAG output.....	20
8.2.2	Interpreting the diagnostics result.....	20
8.2.3	Performed tests	21
9	Appendix – list of files.....	22
9.1	tr_nonbonded.....	22
9.2	unb_tr_nonbonded	22
9.3	Python programs	22

Terminology:

BN	Back Node
BN_BI	Back node – Backplane Interface
CMU	Clock Multiplier Unit
FIFO	First In First Out
FN	Front Node
FN_BN	Front Node – Back Node
FPGA	Field Programmable Gate Array
FSM	Finite State Machine
Nof	Number of
PCS	Physical Coding Sublayer
PHY	Physical layer
PMA	Physical Media Attachment
RX	Receive
SI_FN	Serial Interface – Front Node
SOPC	System On a Programmable Chip (Altera)
TX	Transmit
XGB	10 Gigabit Breakout board

References:

1. 'DP Streaming Module Description', ASTRON-RP-382, Eric Kooistra
2. 'Design considerations for UniBoard's Stratix IV Transceivers', ASTRON-RP-386, Daniel van der Schuur
3. 'Quartus II Handbook', quartusii_handbook.pdf, www.altera.com
4. 'Altera Stratix IV Device Handbook', July 2010, , www.altera.com
5. 'UniBoard Board Design', ASTRON RP-316, Gijs Schoonderbeek, Sjouke Zwier
6. \$UNB/Firmware/modules/diagnostics
7. www.altera.com, "Avalon Interface Specifications", mnl_avalon_spec.pdf
8. https://svn.astron.nl/UniBoard_FP7/UniBoard/trunk, the UniBoard FP7 SVN repository (\$UNB)
9. "Specification for module interfaces using VHDL records", ASTRON-RP-380, Eric Kooistra
10. "Common Library Memory and Register Component Descriptions", ASTRON-RP-415, Eric Kooistra
11. "UNB_Common Module Description", ASTRON-RP-426, Eric Kooistra
12. "UniBoard transceiver module: tr_nonbonded", ASTRON-RP-405, Daniel van der Schuur
13. "DIAG module description", ASTRON-RP-1313, Harm Jan Pepping
14. \$UNB/Software/python/README.txt

1 Introduction

1.1 Purpose

The UniBoard uses high speed Stratix IV transceivers to interface between FPGAs using the mesh (front node - back node or FN_BN) interconnect, and to interface to hardware via the back node to backplane interface (BN_BI). Details and background information can be obtained by reading [2] and [5]. This document describes a VHDL transceiver module, `mms_tr_nonbonded`, that can be used on the UniBoard to implement up to twelve transceiver links in basic non-bonded mode per FPGA side (left and/or right).

1.2 Performance and limitations

The module provides stable, error-free data (de)serialization at 6.25Gbps per transceiver link when interfaced via the UniBoard's FN_BN mesh. The BN_BI interface also supports a data rate of 6.25Gbps, however only via PCB traces. When CX4 cables are used, performance drops and is error-free up to 5Gbps. The Stratix IV transceivers support a data rate of 8.5Gbps, but this data rate requires the on-board crystals to be replaced with 425MHz ones, which is not compatible with XAUI and 10GbE. The data rate of 6.25Gbps is the maximum achievable using the installed 156.25MHz crystals. The mentioned data rates include the 8/10 encoding overhead.

1.3 Module overview

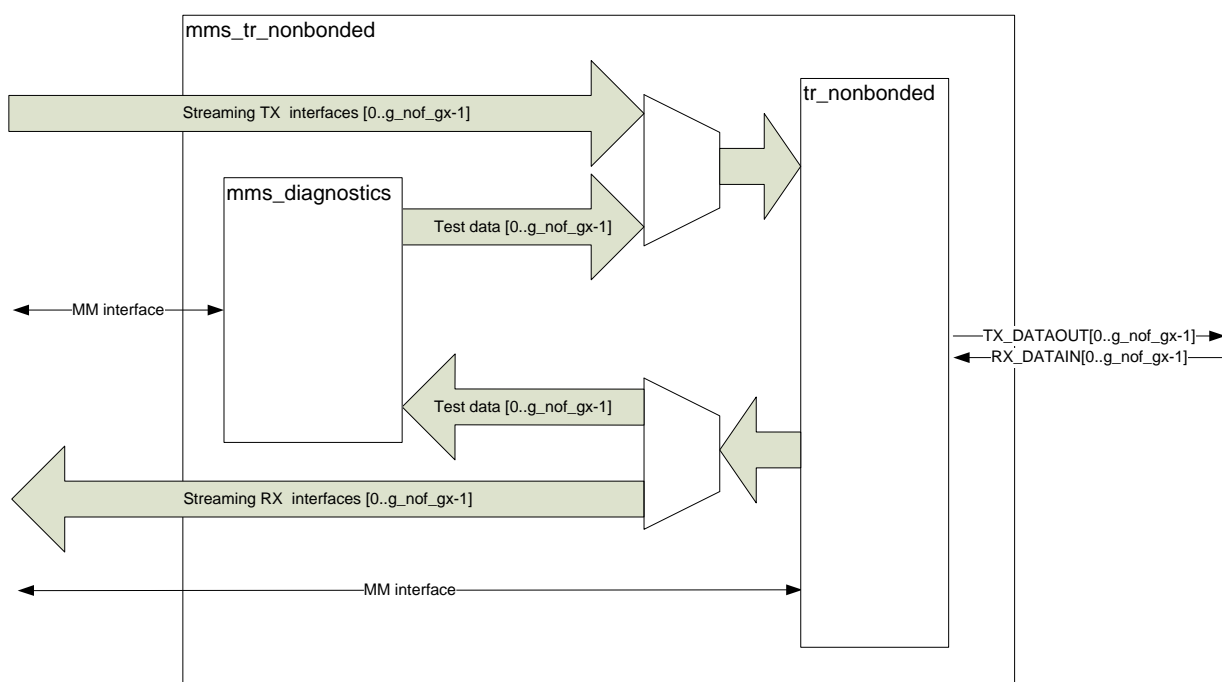


Figure 1 – mms_tr_nonbonded overview

Figure 1 shows the top-level of `mms_tr_nonbonded` and its main components. An internal `mms_diagnostics` instance is used to allow the user to send diagnostics data across the transceiver links instead of user data. This `mms_diagnostics` instance is controlled with an MM interface. See [13] for more information on the `mms_diagnostics` component.

The actual transceiver IP, Altera's ALTGX megafunction, is instantiated inside `tr_nonbonded`.

2 Hardware interface

2.1 Clock and reset signals

Figure 2 shows the clock and reset signals that exist in the tr_nonbonded module.

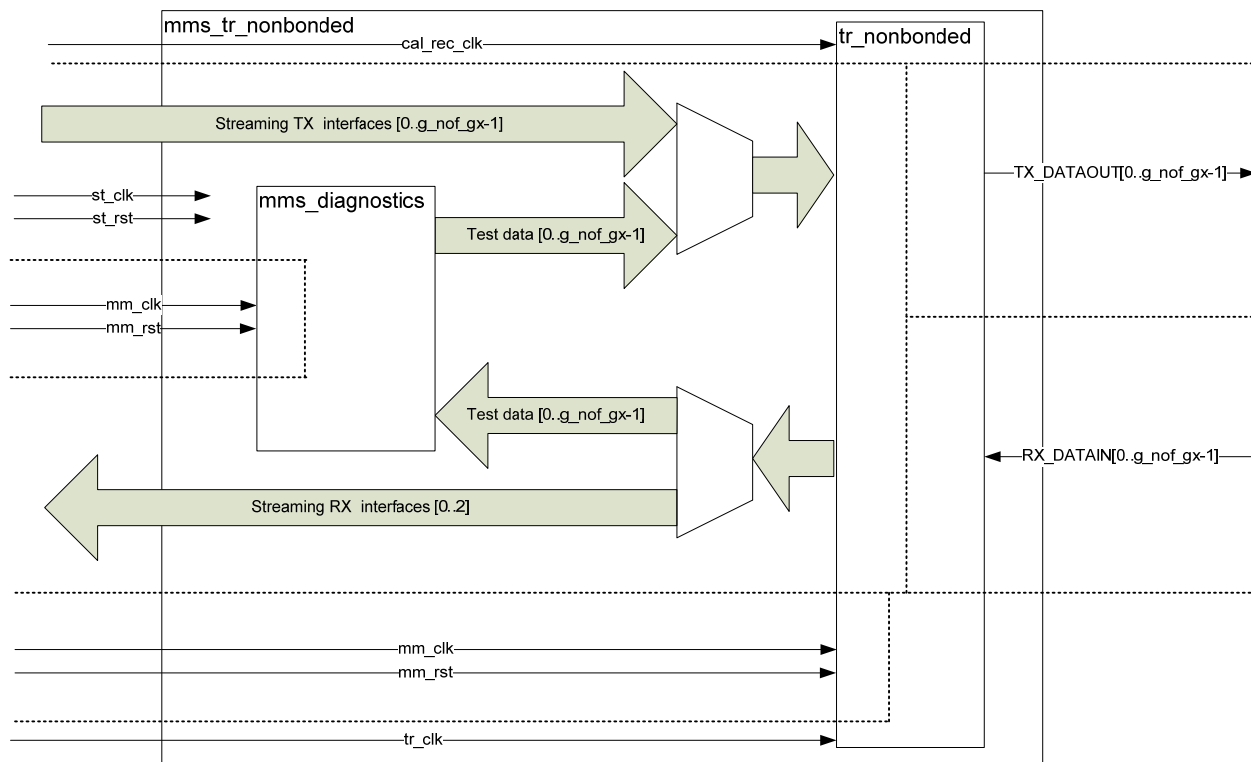


Figure 2 – mms_tr_nonbonded clock and reset signals

Table 1 summarizes the clock signals for mms_tr_nonbonded.

Name	Frequency (MHz)	Description
tr_clk	156.25	Reference clock for the ALTGX IP
st_clk	$F \leq (g_mbps/32) * (8/10)$ $F \leq 156.25$ (6250Mbps) $F \leq 125$ (5000Mbps) $F \leq 78,125$ (3125Mbps) $F \leq 62,5$ (2500Mbps)	Streaming domain clock input. The maximum frequency depends on the selected data rate (g_mbps which includes 8/10 encoding overhead), see the Parameters paragraph below.
mm_clk	User	MM clock for the memory-mapped bus shared with e.g. a NIOS II processor. Default is 50MHz.
cal_rec_clk	37,5 - 50	Used to clock the calibration block inside the transceiver PHY components. The frequency must be between 37.5 and 50MHz.

Table 1: mms_tr_nonbonded clock signals

2.2 Parameters

Available parameters for the tr_nonbonded module are listed in Table 2.

Generic	Type	Description
g_nof_gx	NATURAL	The number of transceiver instances
g_mbps	NATURAL	Line data rate in Megabits per second, including 8/10 encoding overhead. Supported values: <ul style="list-style-type: none"> 6250 5000 3125 2500
g_tx, g_rx	BOOLEAN	Supported: <ul style="list-style-type: none"> g_tx=TRUE, g_rx=TRUE: g_nof_gx (duplex) transceiver instances g_tx=TRUE, g_rx=FALSE: g_nof_gx transmitter instances only g_tx=FALSE, g_rx=TRUE: g_nof_gx receiver instances only
g_tx_fifo_depth	NATURAL	Transmitter side FIFO depth in words (32-bit)
g_rx_fifo_depth	NATURAL	Receiver side FIFO depth in words (32-bit)

Table 2: tr_nonbonded parameters

2.3 Interface signals

The interface signals of mms_tr_nonbonded are listed in Table 3.

Interface	Type	Description
snk_in_arr	t_dp_sosi_arr	Array (g_nof_gx-1..0) of streaming TX data inputs, Data width = 32.
snk_out_arr	t_dp_siso_arr	Array (g_nof_gx-1..0) of flow control signals
src_out_arr	t_dp_sosi_arr	Array (g_nof_gx-1..0) streaming RX data outputs
src_in_arr	t_dp_siso_arr	Array (g_nof_gx-1..0) flow control signals
rx_datain	STD_LOGIC_VECTOR	Array of g_nof_gx serial receiver lanes
tx_dataout	STD_LOGIC_VECTOR	Array of g_nof_gx serial transmitter lanes
tr_nonbonded_mm_mosi	t_mem_mosi	MOSI interfaces to the mms_tr_nonbonded instance
tr_nonbonded_mm_miso	t_mem_miso	MISO interfaces from the mms_tr_nonbonded instance
diagnostics_mm_mosi	t_mem_mosi	MOSI interfaces to the mms_diagnostics instance
diagnostics_mm_miso	t_mem_miso	MISO interfaces from the mms_diagnostics instance

Table 3: interface signals

3 Software interface

An mms_tr_nonbonded instance provides MM buses to control its internal mms_diagnostics module, and provides MM control and monitoring of tr_nonbonded.

3.1 mms_diagnostics

[13] lists the register layout and contents of mms_diagnostics. Generic g_nof_streams equals the number of instantiated transceivers: g_nof_gx.

3.2 tr_nonbonded

Available registers of the tr_nonbonded module are listed in Table 4. All registers are 32 bits wide, however only the rx_dataout registers use the full 32 bits. The number of used bits of the remaining registers corresponds directly to the number of instantiated transceivers – g_nof_gx (bit 0 corresponds to transceiver 0, etc). This implies a maximum of 32 transceivers per tr_nonbonded module.

Name	Address (words)	Size (words)	Read/Write	Description
tx_align_en	0	1	W	'1' overrides any user data and puts an alignment pattern on the transmitter(g_nof_gx-1..0) parallel TX input.
tx_state	1	1	R	Indicates the state of each transmitter using 2 bits per transmitter, e.g. bits [1..0] = state of transmitter 0. TX state; normal FSM sequence: 00: Init 01: Sending alignment pattern 11: Allowing user data; normal state when TX is operating correctly. TX state; manually overridden 10: Manually forcing alignment pattern; tx_align_en='1'. Once user sets tx_align_en back to '0', tx_state will return to normal '11'.
rx_align_en	2	1	W	A '1' enables alignment for the corresponding receiver(g_nof_gx-1..0). Alignment requires the transmission of the alignment pattern on the connecting transmitter.
rx_state	1	1	R	Indicates the state of each receiver using 2 bits per receiver, e.g. bits [1..0] = receiver 0. RX state; normal FSM sequence: 00: Init 01: Byte aligning 10: Word aligning 11: Aligned; normal state when RX is operating correctly. When the FSM is in this state, the user can force the FSM to restart the alignment procedure by writing to the rx_align_en register.
rx_dataout_0	4	1	R	Parallel RX data out of transceiver 0.
rx_dataout_1	5	1	R	Parallel RX data out of transceiver 1.
rx_dataout_2	6	1	R	Parallel RX data out of transceiver 2.
rx_dataout_3	7	1	R	Parallel RX data out of transceiver 3.
rx_dataout_4	8	1	R	Parallel RX data out of transceiver 4.

Doc.nr.: ASTRON-RP-449
Rev.: 0.5
Date: 24-09-2012
Class.: Public

UniBoard

rx_dataout_5	9	1	R	Parallel RX data out of transceiver 5.
rx_dataout_6	10	1	R	Parallel RX data out of transceiver 6.
rx_dataout_7	11	1	R	Parallel RX data out of transceiver 7.
rx_dataout_8	12	1	R	Parallel RX data out of transceiver 8.
rx_dataout_9	13	1	R	Parallel RX data out of transceiver 9.
rx_dataout_10	14	1	R	Parallel RX data out of transceiver 10.
rx_dataout_11	15	1	R	Parallel RX data out of transceiver 11.

Table 4: tr_nonbonded module registers

4 Design

This chapter describes the `tr_nonbonded` module that is instantiated by `mms_tr_nonbonded`, focussing on the different clock domains, the FIFOs and the options to bypass these.

4.1 Clock domains

Figure 3 shows an overview of the `tr_nonbonded` module, in which one can distinguish four user provided clocks and two derived clocks (not including the serial clocks in `tx_dataout` and `rx_datain`). These clocks and their descriptions are listed in Table 1. The read sides of the TX FIFOs are clocked by the generated `tx_clk` signals, and the write sides of the RX FIFOs are clocked by the generated `rx_clk` signals. Figure 4 shows the clock domains in case no FIFOs are instantiated (`g_fifos = FALSE`), and the streaming interfaces access (TX) or emerge from (RX) the generated transceiver clock domains directly.

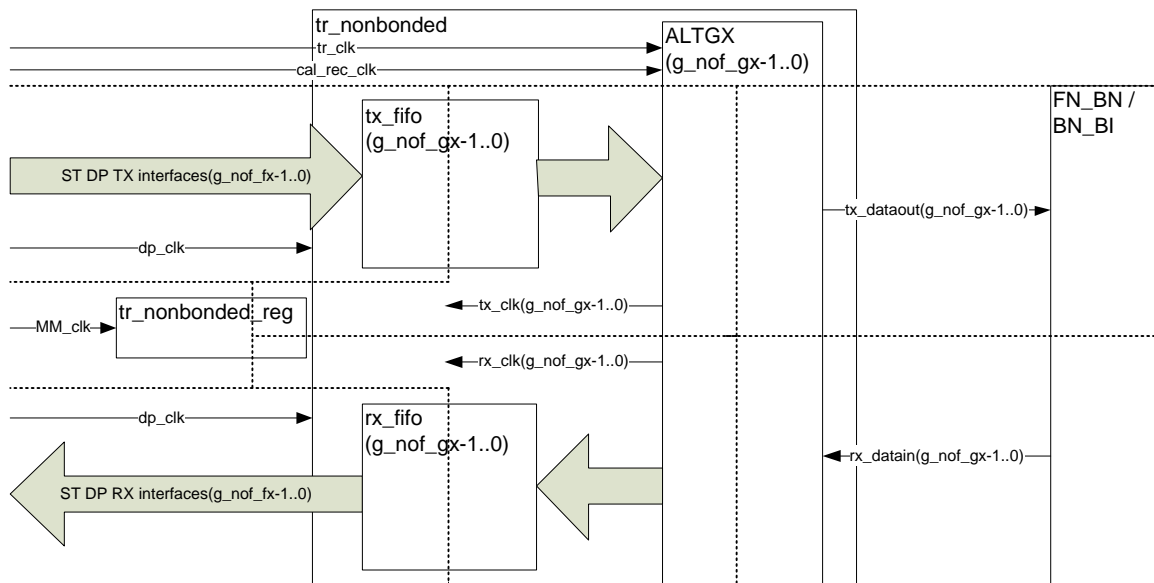


Figure 3: clock domains with FIFOs

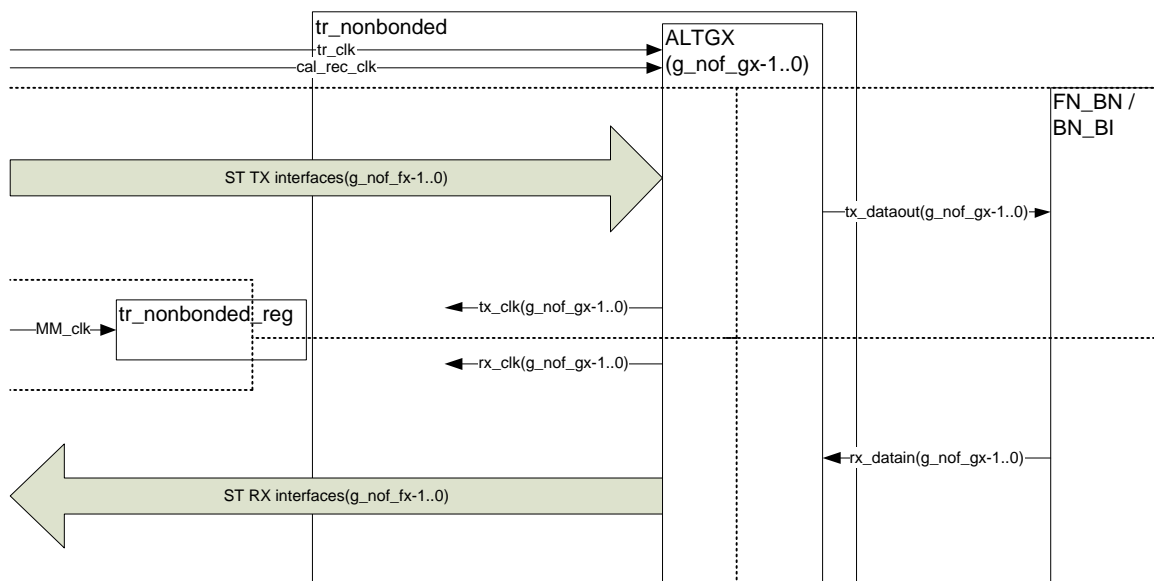


Figure 4: clock domains without FIFOs

Name	Frequency (MHz)	Description
tr_clk	156.25	Used to generate transmitter clock, also provides receiver reference clock.
tx_clk	156.25 (6250Mbps) 125 (5000Mbps) 78,125 (3125Mbps) 62,5 (2500Mbps)	Transmitter clock, one for each instantiated transceiver, derived from tr_clk. Used to clock in parallel transmitter data.
rx_clk	156.25 (6250Mbps) 125 (5000Mbps) 78,125 (3125Mbps) 62,5 (2500Mbps)	Receiver clock, one for each instantiated transceiver, derived from the clock that is recovered from the received serial stream. Used to clock out parallel receiver data.
mm_clk	user	MM clock for the memory-mapped bus shared with e.g. a NIOS II processor. Default = 50MHz.
dp_clk	user	Clock for the streaming interfaces on user side. Default = 200MHz.
cal_reconf_clk	37,5 - 50	Used to clock the ALTGX calibration block and the ALTGX_RECONFIG transceiver reconfiguration block. The frequency must lie between 37.5 and 50MHz [3].

Table 5: tr_nonbonded clock signals

4.2 Parameters

Available parameters when instantiating the tr_nonbonded module are listed in Table 2.

Generic	Type	Description
g_nof_gx	NATURAL	The number of instantiated transceivers. Limited to 32 as result of control register width, also limited by the FPGA device used. Defaults to 12, the maximum number of full-speed transceivers on one side of a UniBoard FPGA.
g_mbps	NATURAL	Transceiver data rate in Mbps. Supported: 2500, 3125, 5000, 6250
g_tx	BOOLEAN	Transmitters are instantiated when TRUE. The number of instances equals g_nof_gx.
g_rx	BOOLEAN	Receivers are instantiated when TRUE. The number of instances equals g_nof_gx.
g_fifos	BOOLEAN	Instantiate FIFOs between user clock domain (DP clock domain) and TX/RX clock domains.

Table 6: tr_nonbonded parameters

4.3 Interface signals

The interface signals of the tr_nonbonded module are shown in Figure 5. Table 3 lists the general specifications of these interfaces, while the next paragraphs provide more detailed information. The status and control signals of the tr_nonbonded module can be accessed via a memory mapped interface, using tr_nonbonded_reg, or can be used without tr_nonbonded_reg. The functions of the status and control signals between tr_nonbonded_reg and the tr_nonbonded module are listed in Table 8.

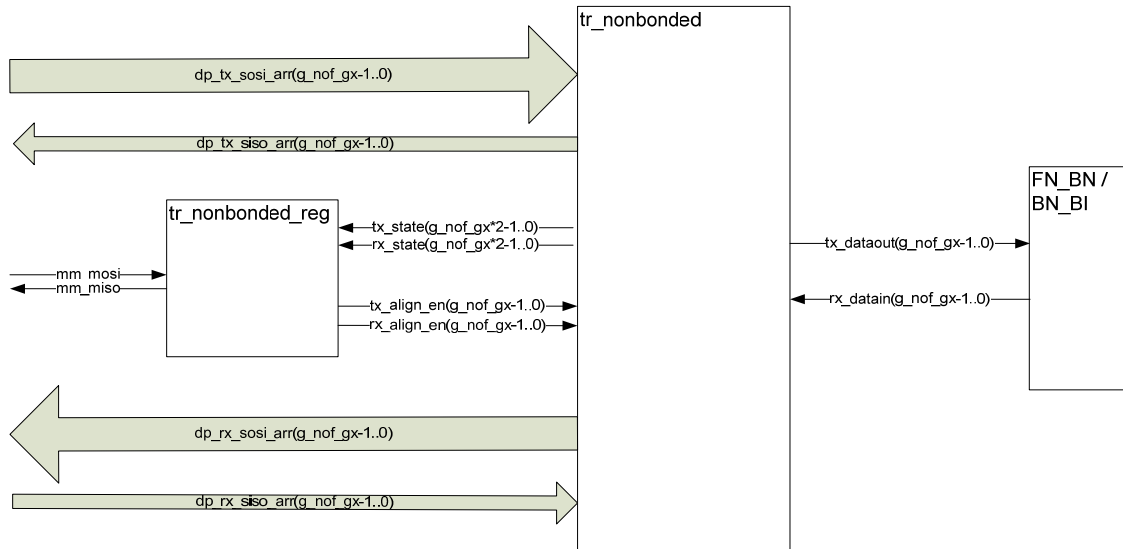


Figure 5: interface signals (with FIFOs)

Interface	Type	Description
tx_sosi_arr	t_dp_sosi_arr	Array (g_nof_gx-1..0) of streaming TX data ports, source to sink (tr_nonbonded module). When no FIFOs are used, these streaming interfaces connect directly to the transmitters. When FIFOs are used, this array remains unconnected.
tx_siso_arr	t_dp_siso_arr	Array (g_nof_gx-1..0) of flow control signals from tr_nonbonded TX ports, sink to source. When no FIFOs are used, these streaming interfaces connect directly to the transmitters. When FIFOs are used, this array remains unconnected.
rx_sosi_arr	t_dp_sosi_arr	Array (g_nof_gx-1..0) streaming RX data ports, source (tr_nonbonded module) to sink. When no FIFOs are used, these interfaces connect directly to the receivers. When FIFOs are used, this array is still available to allow the user to view the RX data as it is before it enters the FIFO.
rx_siso_arr	t_dp_siso_arr	Array (g_nof_gx-1..0) flow control signals to tr_nonbonded RX ports, sink to source. When no FIFOs are used, these interfaces connect directly to the receivers. When FIFOs are used, this array remains unconnected.
dp_tx_sosi_arr	t_dp_sosi_arr	Array (g_nof_gx-1..0) of streaming TX data ports, source to sink (tr_nonbonded module). These streaming interfaces connect to the write sides of the TX FIFOs .
dp_tx_siso_arr	t_dp_siso_arr	Array (g_nof_gx-1..0) of flow control signals from tr_nonbonded TX ports, sink to source. These interfaces provide flow control from the write sides of the TX FIFOs .
dp_rx_sosi_arr	t_dp_sosi_arr	Array (g_nof_gx-1..0) streaming RX data ports, source (tr_nonbonded module) to sink. These streaming interfaces connect to the read sides of the RX FIFOs .
dp_rx_siso_arr	t_dp_siso_arr	Array (g_nof_gx-1..0) flow control signals to tr_nonbonded RX

		ports, sink to source. These interfaces provide flow control to the read sides of the RX FIFOs .
tx_dataout	STD_LOGIC_VECTOR (g_nof_gx-1..0)	Array of transmitter serialized data outputs.
rx_datain	STD_LOGIC_VECTOR (g_nof_gx-1..0)	Array of receiver serialized data inputs.

Table 7: interface signals

Interface	Type	Description
tx_state	STD_LOGIC_VECTOR (g_nof_gx*2-1..0)	Indicates the state of each transmitter using 2 bits per transmitter, e.g. signals [1..0] = state of transmitter 0. TX state; normal FSM sequence: 00: Init 01: Sending alignment pattern 11: Allowing user data; normal state when TX is operating correctly. TX state; manually overridden 10: Manually forcing alignment pattern; tx_align_en='1'. Once user sets tx_align_en back to '0', tx_state will return to normal '11'.
rx_state	STD_LOGIC_VECTOR (g_nof_gx*2-1..0)	Indicates the state of each receiver using 2 bits per receiver, e.g. signals [1..0] = receiver 0. RX state; normal FSM sequence: 00: Init 01: Byte aligning 10: Word aligning 11: Aligned; normal state when RX is operating correctly. When the FSM is in this state, the user can force the FSM to restart the alignment procedure by writing to the rx_align_en register.
tx_align_en	STD_LOGIC_VECTOR (g_nof_gx-1..0)	'1' overrides any user data and puts an alignment pattern on the transmitter(g_nof_gx-1..0) parallel TX input.
rx_align_en	STD_LOGIC_VECTOR (g_nof_gx-1..0)	A rising edge enables alignment for a receiver(g_nof_gx-1..0). Full alignment to a received alignment pattern requires this signal(for each receiver) to be asserted twice: <ul style="list-style-type: none"> The first time performs word alignment The second time performs byte ordering

Table 8: tr_nonbonded status and control signals

5 Implementation

5.1 Architecture

Figure 6 shows the internal architecture of the tr_nonbonded module. Its main components are the used IP block (one to twelve ALTGX instances), the required reset circuitry (phy_tx_rst and phy_rx_rst), the FIFOs and logic to control alignment and validation of data. User data is fed into the TX FIFOs via an array of streaming interfaces, and is clocked out of the RX FIFO via an identical array of streaming interfaces.

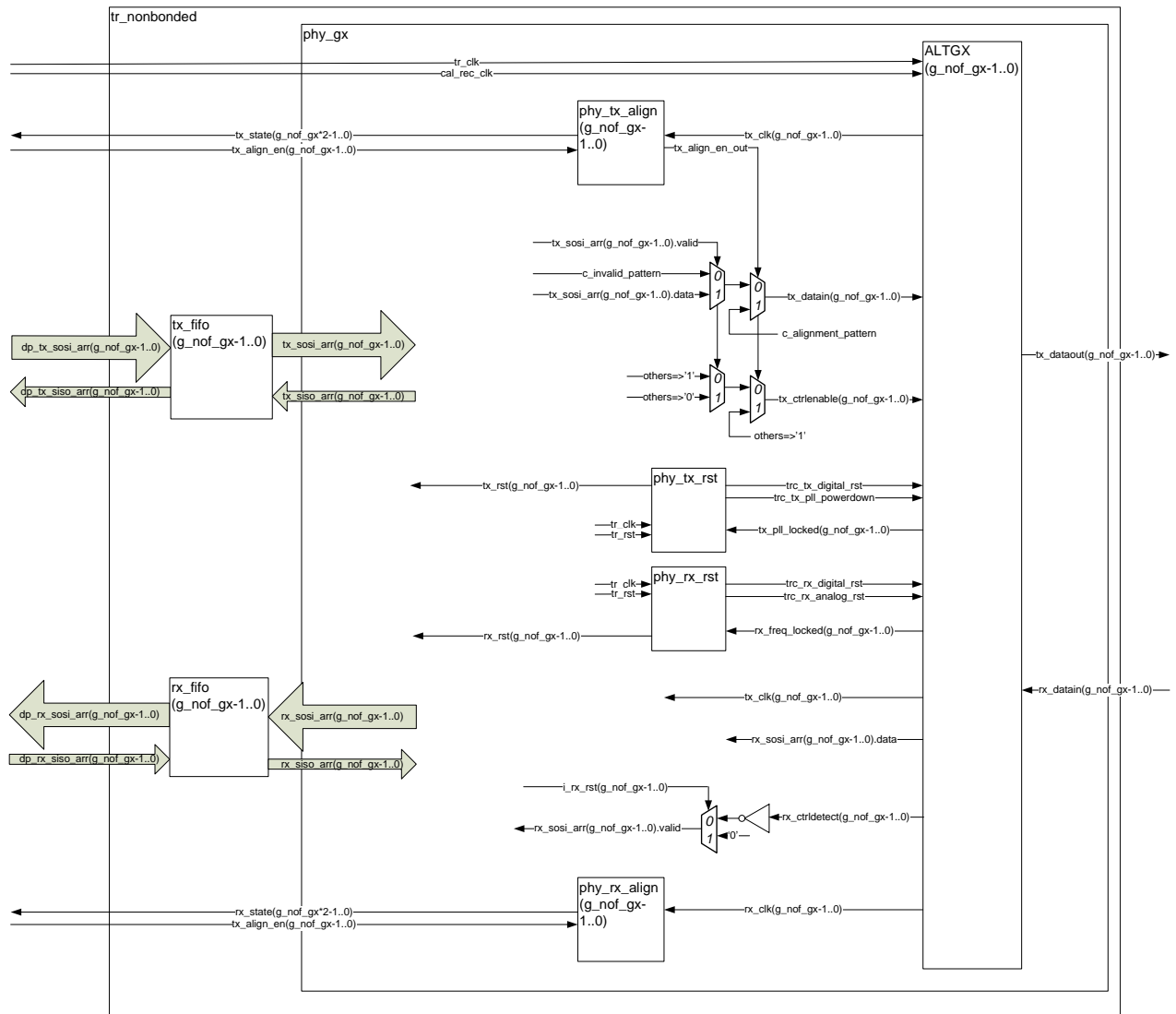


Figure 6: tr_nonbonded architecture

5.2 ALTGX Megafunction

The ALTGX Megafunction is instantiated g_nof_gx times. Quartus will map the transceiver instances onto the 3 available transceiver blocks (so up to four transceivers per block) on one side of the FPGA. The side (left = FN_BN or right = BN_BI interface) depends on which pins the user assigned the tr_clk and the serial I/O pins of tr_nonbonded.

5.2.1 Alignment of received data

Upon serialization of parallel data, the word boundary is lost. To restore this word boundary, an alignment block is used on the receiver side. Besides that, depending on when the system comes out of reset, the byte order may not match the original byte order. This problem is addressed by the *byte ordering block*, available as part of the Altera ALTGX IP block. This paragraph describes the alignment procedure and these two necessary blocks.

5.2.1.1 Word alignment

When a 32-bit width is used for the fabric-transceiver interface, as illustrated in Figure 7, the 32-bit data is byte serialized into half the width at twice the frequency. The resulting 16 bits are fed into the 8b/10b encoder that encodes the MSB and LSB into 10-bits each and outputs these 20 bits to the serializer. On the receiver side, the word aligner is located before the 8b/10b decoder and thus looks for word alignment patterns that are 8b/10b encoded. This word alignment pattern, 10 (single width) or 20 (double width) bits wide, is entered as a parameter in the corresponding ALTGX Megafunction field (see Figure 8).

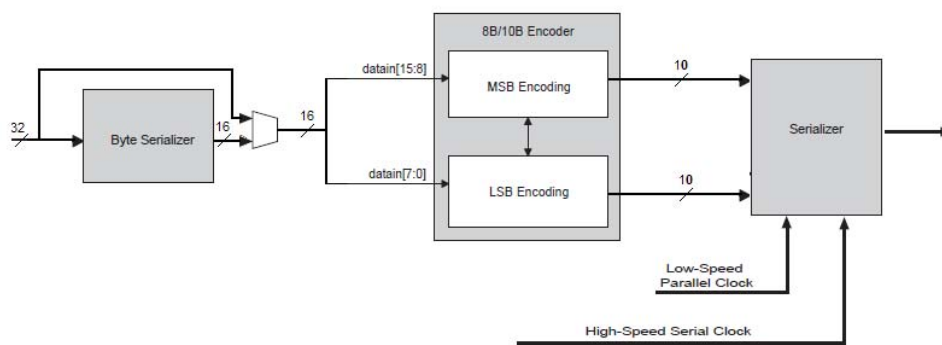


Figure 7: Transmitter data path within ALTGX

Word Aligner

- ☒ Use manual word alignment mode
 - When should the word aligner realign?
 - ☐ Realign continuously while 'rx_enapatternalign' is high
 - ☒ Realign at the rising edge of 'rx_enapatternalign'
- ☐ Use manual bitslipping mode
- ☐ Use the automatic synchronization state machine mode
 - Number of continuous valid code groups received to reduce the error count by 1: 1
 - Number of erroneous code groups(error count) received to lose sync: 1
 - Number of valid code groups received to achieve sync: 1

What is the word alignment pattern length? 20

What is the word alignment pattern? 01011111000010111100 SF0BC

- ☐ Flip word alignment pattern bits
- ☐ Enable run-length violation checking with a run length of 40
- ☐ Enable word aligner output reverse bit ordering
- ☐ Create 'rx_syncstatus' output port for pattern detector and word aligner
- ☒ Create 'rx_patterndetect' port to indicate pattern detected
- ☐ Create 'rx_invpolarity' to enable word aligner polarity inversion
- ☐ Create 'rx_revbyteorderwa' to enable receiver symbol swap
- ☐ Create 'rx_bitslipboundaryselectout' port to indicate the number of bits slipped in the word aligner

Figure 8: Word alignment

The 20-bit default alignment pattern for the tr_nonbonded module is:

```
0011110100 0011111010 = 8b/10b encoded, LS Bits right
K28.0      K.28.5      = Comma symbol followed by control word
0xBC      0x1C        = 8b/10b decoded
```

This corresponds to 0xBC1C after 8b/10b decoding at the receiver end and before 8b/10b encoding at the transmitter end. The comma symbol (K.28.5 or 0xBC) differs from the second byte so their places cannot be swapped during alignment. The second byte (0x1C) is also used in the byte ordering block described in the next paragraph.

The alignment pattern is input in the Megafunction in reverse, the way it is sent after serialization:

```
0101111100 0010111100 (0x5F0BC) = 8b/10b encoded, LS Bits left (reversed)
```

5.2.1.2 Byte ordering

As described in the previous paragraph, the 0x1C byte is used for byte ordering. The byte ordering block (available as option in the ALTGX Megafunction, see Figure 9) uses the synchronization status output of the word aligner to start the byte ordering – this is necessary because the byte deserializer at the receiver could output the bytes in the wrong order. When it finds byte ordering pattern 0x1C at the LSB position, it considers the data to be aligned. Otherwise, it inserts a padding pattern (set to 0x00) to effectively shift the data onto the LSB position.

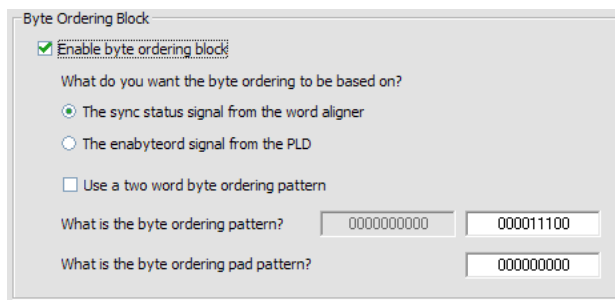


Figure 9: Byte ordering options

6 Reference design

A reference design, unb_tr_nonbonded, instantiates mms_tr_nonbonded along with yet another mms_diagnostics module (besides the mms_diagnostics module inside mms_tr_nonbonded). This second mms_diagnostics module emulates an external user data stream, whereas the internal mms_diagnostics is able to override this user data stream with a test data stream that can be verified independently from the user stream.

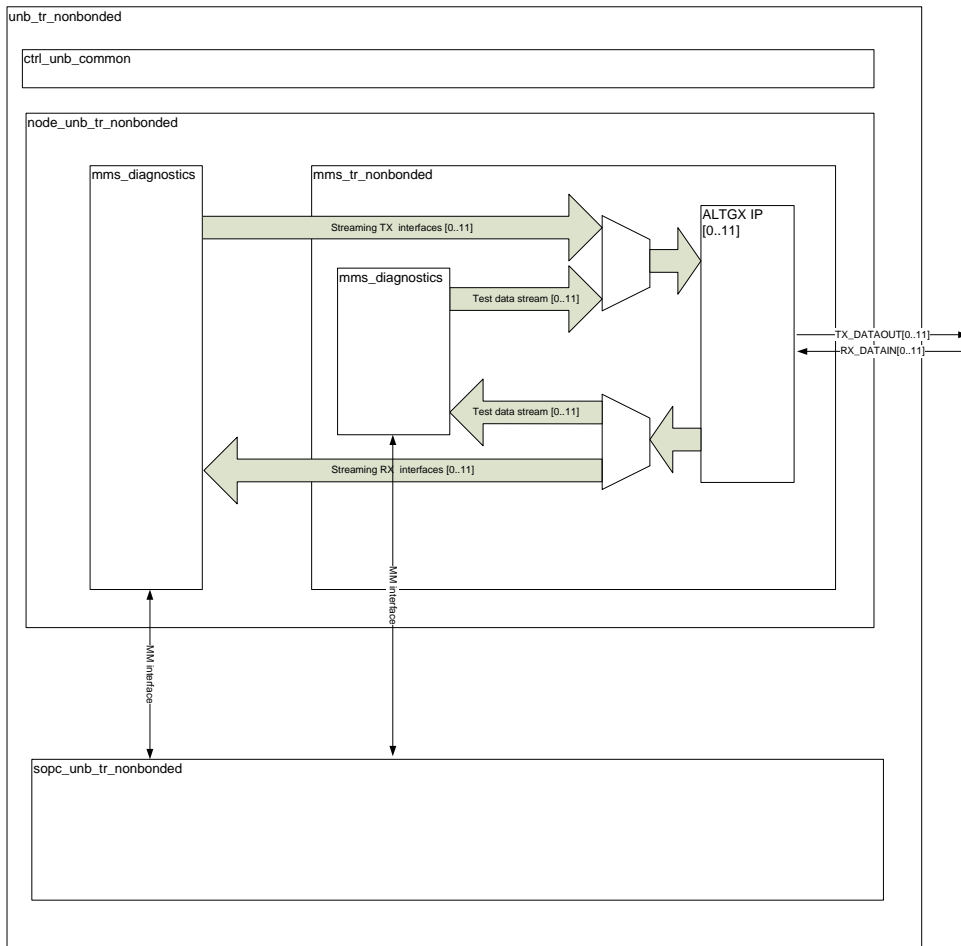


Figure 10 - design unb_tr_nonbonded

The second instance of mms_diagnostics operates in exactly the same way as the internal instance, with the exception that the internal one, when enabled, overrides the external diagnostics stream.

The NIOS II processor in the SOPC can run a dedicated C application, see Chapter 9, or unb_osy to allow Python programs to target the design.

There are three revisions of this design, which are discussed in Chapter 8.

7 Verification

Verification of the tr_nonbonded module is done using the diagnostics module to generate and verify data. This basic scheme is used in three levels:

- tb_tr_nonbonded
 - uses VHDL stimuli
- tb_node_unb_tr_nonbonded
 - uses VHDL functions to access tr_nonbonded module via dedicated MM bus
 - uses VHDL functions to access diagnostics module via dedicated MM bus
- tb_unb_tr_nonbonded
 - conducts the same MM accesses through one shared MM bus using a C application (main.c) running on a NIOS II processor

In addition, two instances of tb_node_unb_tr_nonbonded are verified in 'multi test bench'-test bench:

- tb_tb_node_unb_tr_nonbonded
 - uses two instances of tb_node_unb_tr_nonbonded

8 Validation

The unb_tr_nonbonded design instantiates one mms_tr_nonbonded instance, of which the transceivers can be connected to the UniBoard's FN_BN interface or BN_BI interface. As these interfaces are permanently connected to the FPGA's left transceiver blocks in case of the FN_BN interface, and to the right transceiver blocks in case of the BN_BI interface, two separate Quartus revisions are necessary for each interface. This is caused by the fact that Quartus requires a design containing all transceiver pins in the QSF (for both the FN_BN and BN_BI) to use all of these pins. Therefore, a unb_* revision can run on any node and connects the tr_nonbonded module to the FPGA's transceiver blocks that use the FN_BN interface, and the bn_* revision can run on the back nodes only because it instantiates a tr_nonbonded module on the BN_BI side. The next paragraphs list the properties of each existing Quartus project revision.

8.1 Design revisions

8.1.1 Front node and/or back node revisions

8.1.1.1 Unb_tr_nonbonded

- FN_BN interface
- Tr_clk = SB_CLK
- Transmitters and receivers
- QSF sets g_mbps to g_mbps_fn_bi (= 6250Mbps by default)

8.1.1.2 Unb_tx_nonbonded

- FN_BN interface
- Tr_clk = SB_CLK
- Transmitters only
- QSF sets g_mbps to g_mbps_fn_bi (= 6250Mbps by default)
- QSF sets g_rx to FALSE

8.1.1.3 Unb_rx_nonbonded

- FN_BN interface
- Tr_clk = SB_CLK
- Receivers only
- QSF sets g_mbps to g_mbps_fn_bi (= 6250Mbps by default)
- QSF sets g_tx to FALSE

8.1.2 Back node only revisions

8.1.2.1 Bn_tr_nonbonded

- BN_BI interface
- Tr_clk = SA_CLK
- Transmitters and receivers
- QSF sets g_mbps to g_mbps_bn_bi (= 5000Mbps by default)

8.1.2.2 Bn_tx_nonbonded

- BN_BI interface
- Tr_clk = SA_CLK
- Transmitters only
- QSF sets g_mbps to g_mbps_bn_bi (= 5000Mbps by default)
- QSF sets g_rx to FALSE

8.1.2.3 Bn_rx_nonbonded

- BN_BI interface
- Tr_clk = SA_CLK
- Receivers only
- QSF sets g_mbps to g_mbps_bn_bi (= 5000Mbps by default)
- QSF sets g_tx to FALSE

Python programs tc_tr_nonbonded.py, tc_tr_nonbonded_dgn.py and a utility script can be used to target these designs. In addition, the dedicated C application main.c can be used. The results of the first tests with this C application are discussed in paragraph 8.2.

8.2 Dedicated C application

This paragraph describes the test results that were obtained using the dedicated C application.

8.2.1 JTAG output

The output printed by unb_tr_nonbonded closely reflects the actions performed by the NIOS (IOWR and IORD). As an example, the following is the JTAG output of back node 0 that receives data from front node 0 only.

UNB_TR_NONBONDED: 28 Oct 2011

*Waiting for TX to be ready.
Waiting for RX to be ready.
TX: sending alignment pattern.
RX: Disabling alignment.
RX: Enabling alignment.
RX: Disabling alignment.
RX: Enabling alignment.
TX: Disabling alignment.
Enabling diagnostics sink to flush any RX data.
Resetting diagnostics sink.
Enabling diagnostics source.
Diag valid : fff
Diag result: 1ff
Diag valid : fff
Diag result: 1ff*

8.2.2 Interpreting the diagnostics result

The printed diag valid and diag_result values will vary between 0 and 12 (decimal). The values are printed in hexadecimal format. For example, a diag valid and result value indicating all 12 receivers are interpreting the received data as valid (the 0x is not printed), followed by the binary representation, one bit for each receiver:

*Diag valid: 0xFFFF = 111111111111 (12 valid input streams)
Diag result 0x0 = 000000000000 (12 OK results)*

On the FB_BN interface, each front node is connected to each back node with 3 transceiver links. This means the result should be interpreted as follows:

*Diag valid: 0xFFFF = 111.111.111.111
Diag result 0x0 = 000.000.000.000*

The least significant bit maps to transceiver 0. One can distinguish four transceiver bundles this way, one bundle of 3 coming from each connected node. The connection scheme is summarized in Table 9, which shows which front node connects to which back node, and via which interface.

Front node	Transceivers	Front node interface	Back node interface	Transceivers	Back node
0	0,1,2	FN_BN_0	FN_BN_3	9,10,11	0
	3,4,5	FN_BN_1	FN_BN_1	3,4,5	3
	6,7,8	FN_BN_2	FN_BN_2	6,7,8	2
	9,10,11	FN_BN_3	FN_BN_3	9,10,11	1
1	0,1,2	FN_BN_0	FN_BN_2	6,7,8	0
	3,4,5	FN_BN_1	FN_BN_2	6,7,8	1
	6,7,8	FN_BN_2	FN_BN_3	9,10,11	3
	9,10,11	FN_BN_3	FN_BN_3	9,10,11	2
2	0,1,2	FN_BN_0	FN_BN_1	3,4,5	0
	3,4,5	FN_BN_1	FN_BN_1	3,4,5	1
	6,7,8	FN_BN_2	FN_BN_1	3,4,5	2
	9,10,11	FN_BN_3	FN_BN_2	6,7,8	3
3	0,1,2	FN_BN_0	FN_BN_0	0,1,2	1
	3,4,5	FN_BN_1	FN_BN_0	0,1,2	0
	6,7,8	FN_BN_2	FN_BN_0	0,1,2	2
	9,10,11	FN_BN_3	FN_BN_0	0,1,2	3

Table 9: Transceiver connections FN_BN

8.2.3 Performed tests

The tr_nonbonded module was tested on the UniBoard using design unb_tr_nonbonded and revision bn_tr_nonbonded. The tests ran error-free for 16 hours (total test duration per test). The maximum data rate tested on the mesh was 6250Mbps. This data rate was also tested on the BN_BI interface using a prototype PCB to loop back a back node's transmitters to its receivers, for several hours. During this time, no errors occurred. The same test was repeated using CX4 cables instead of PCB traces, connecting back nodes bn0 to bn1 and bn2 to bn3. Errors occurred after which another test was performed at a lower data rate of 5000Mbps. This test ran error-free for 16 hours. Finally, the last test was repeated, connecting the back nodes of one UniBoard to the back nodes of another UniBoard. This test also ran error-free for 16 hours.

9 Appendix – list of files

9.1 tr_nonbonded

The tr_nonbonded directory contains the source code for the tr_nonbonded module and the test benches. It is located at:

`$UNB/Firmware/modules/tr_nonbonded/src/vhdl`

The pre-generated ALTGX and ALTGX_RECONFIG files are available from:

`$UNB/Firmware/modules/tr_nonbonded/build/synth/quartus`

9.2 unb_tr_nonbonded

The files for synthesis and simulation of the unb_tr_nonbonded design are in the following directory:

`$UNB/Firmware/designs/unb_tr_nonbonded`

In addition, a C main program to run on the design is located at:

`$UNB/Firmware/software/apps/unb_tr_nonbonded`

9.3 Python programs

The test cases that target design unb_tr_nonbonded or one of its revisions:

`$UNB/Software/python/peripherals/tc_tr_nonbonded.py`
`$UNB/Software/python/peripherals/tc_tr_nonbonded_dgn.py`

The main peripherals used by these scripts are found in:

`$UNB/Software/python/peripherals/pi_tr_nonbonded.py`
`$UNB/Software/python/peripherals/pi_diagnostics.py`

A utility script is located at:

`$UNB/Software/python/peripherals/util_pi_tr_nonbonded.py`

More information on targeting simulations using Python can be obtained by reading [14].