


## Design considerations for UniBoard's Stratix IV transceivers

	Organisatie / Organization	Datum / Date
<b>Auteur(s) / Author(s):</b>  Daniel van der Schuur	ASTRON	25 May 2010
<b>Controle / Checked:</b>  Andre Gunst	ASTRON	21 October 2010
<b>Goedkeuring / Approval:</b>  Andre Gunst	ASTRON	21 October 2010
<b>Autorisatie / Authorisation:</b>  Handtekening / Signature  Andre Gunst	ASTRON	21 October 2010

© ASTRON 2010

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

### UniBoard

Doc.nr.: ASTRON-RP-386

Rev.: 2.0

Date: 21-10-2010

Class.: Public

## Distribution list:

---

Group:	Others:
Andre Gunst Eric Kooistra Rajan Raj Thilak Jonathan Hargreaves (JIVE) Ying Xiang	

## Document history:

---

Revision	Date	Chapter / Page	Modification / Change
0.1	2010-04-15	-	Draft
0.2	2010-04-19	-	BN_backplane interface section added, overclocked XAUI
0.3	2010-04-21	-	Minor changes
1.0	2010-04-21		Final
1.1	2010-05-14	-	Additions /changes in response to review
1.2	2010-05-18	-	Reconfig section, 8b10b encoding
1.3	2010-05-21	-	Offset cancellation, data rate & clock multiplication
1.4	2010-05-25	-	Minor changes
2.0	2010-10-21	-	Final

## Table of contents:

---

1	Introduction.....	5
2	Transceiver infrastructure .....	6
2.1	UniBoard.....	6
2.1.1	Back node – backplane interface (BN_BI) channel routing.....	6
2.1.2	Mesh (FN_BN) channel routing.....	6
2.1.3	Mesh (FN_BN) lane routing.....	7
2.1.4	Number and order of transceiver instances .....	8
2.1.5	Reconfiguration module.....	8
3	UniBoard transceiver clock sources.....	9
3.1	Clock Multiplier Unit (CMU) and Auxiliary Transmit PLL (ATX PLL).....	9
3.1.1	Available data rates .....	10
3.2	CMU (PMA-only) channels.....	10
4	XAUI IP cores.....	12
4.1	XAUI PCS IP core .....	12
4.2	Reduced XAUI (RXAUI) .....	12
4.2.1	Data switch .....	12
5	Comparison of transceiver configurations and modes.....	13
5.1	Available modes .....	13
5.2	Limitations.....	13
5.3	Overview.....	14
5.4	Resource usage .....	14
5.5	Hard + soft components combined.....	14
5.6	XAUI at higher data rates .....	14
5.6.1	Hard XAUI.....	15
5.6.2	Soft XAUI PCS IP .....	15
6	8b/10b encoding.....	16
6.1.1	CDR operation with 8b/10b coding.....	16
6.1.2	CDR operation without 8b/10b coding.....	17
7	Conclusion.....	18

## List of figures:

---

Figure 1 – UniBoard transceiver I/O.....	6
Figure 2 – Front node – Back node – Backplane infrastructure.....	7
Figure 3 – UniBoard transceiver clocking.....	9
Figure 4 – Transceiver architecture [4].....	10
Figure 5 – Clock and Data Recovery block.....	17

## List of tables:

---

Table 1 – Comparison of protocols, data rates and PCS types .....	14
---	----

## Terminology:

---

ATX	Auxiliary Transmit
BN	Back Node
BN_BI	Back node – Backplane Interface
CDR	Clock and Data Recovery
CMU	Clock Multiplier Unit
FIFO	First In First Out
FN	Front Node
FN_BN	Front Node – Back Node
FPGA	Field Programmable Gate Array
GbE	Giga Bit Ethernet
GXB	Gigabit Transceiver Block
PCF	Phase Compensation FIFO
PLL	Phase Locked Loop
PCS	Physical Coding Sublayer
PMA	Physical Media Attachment
RX	Receive
SI_FN	Serial Interface – Front Node
TX	Transmit

## References:

---

- [1] Marvell, RXAUI Interface and RXAUI Adapter Specifications, May 25, 2009
- [2] Altera, EP4SGX230.pdf – pin information for the Stratix IV GX EP4SGX230 device, v1.9 2009
- [3] Altera, AN516.pdf – 10-Gbps Ethernet Reference Design, November 2009
- [4] Altera, Stratix IV Device Handbook, November 2009
- [5] [https://svn.astron.nl/UniBoard\\_FP7/UniBoard/trunk/Firmware/modules/tr\\_nonbonded/](https://svn.astron.nl/UniBoard_FP7/UniBoard/trunk/Firmware/modules/tr_nonbonded/)

## 1 Introduction

The UniBoard is equipped with 8 Stratix IV EP4SGX230KF40 FPGA's, each containing 36 transceivers. This document provides an overview of the possibilities and limitations of these transceivers with regard to UniBoard and available protocols.

As it has been decided that the front side of the UniBoard will use 4 10 GbE interfaces per front node, this document will focus primarily on the mesh interconnect and the back node – backplane interfaces.

## 2 Transceiver infrastructure

### 2.1 UniBoard

Figure 1 depicts the UniBoard mesh infrastructure. Each front node connects to the 4 back nodes via the mesh interconnect. Each front node has four 10 Gigabit Ethernet connections (SI\_FN\_#), whereas a protocol has yet to be determined for the 16 channels (16\*TX, 16\*RX) exiting each back node.

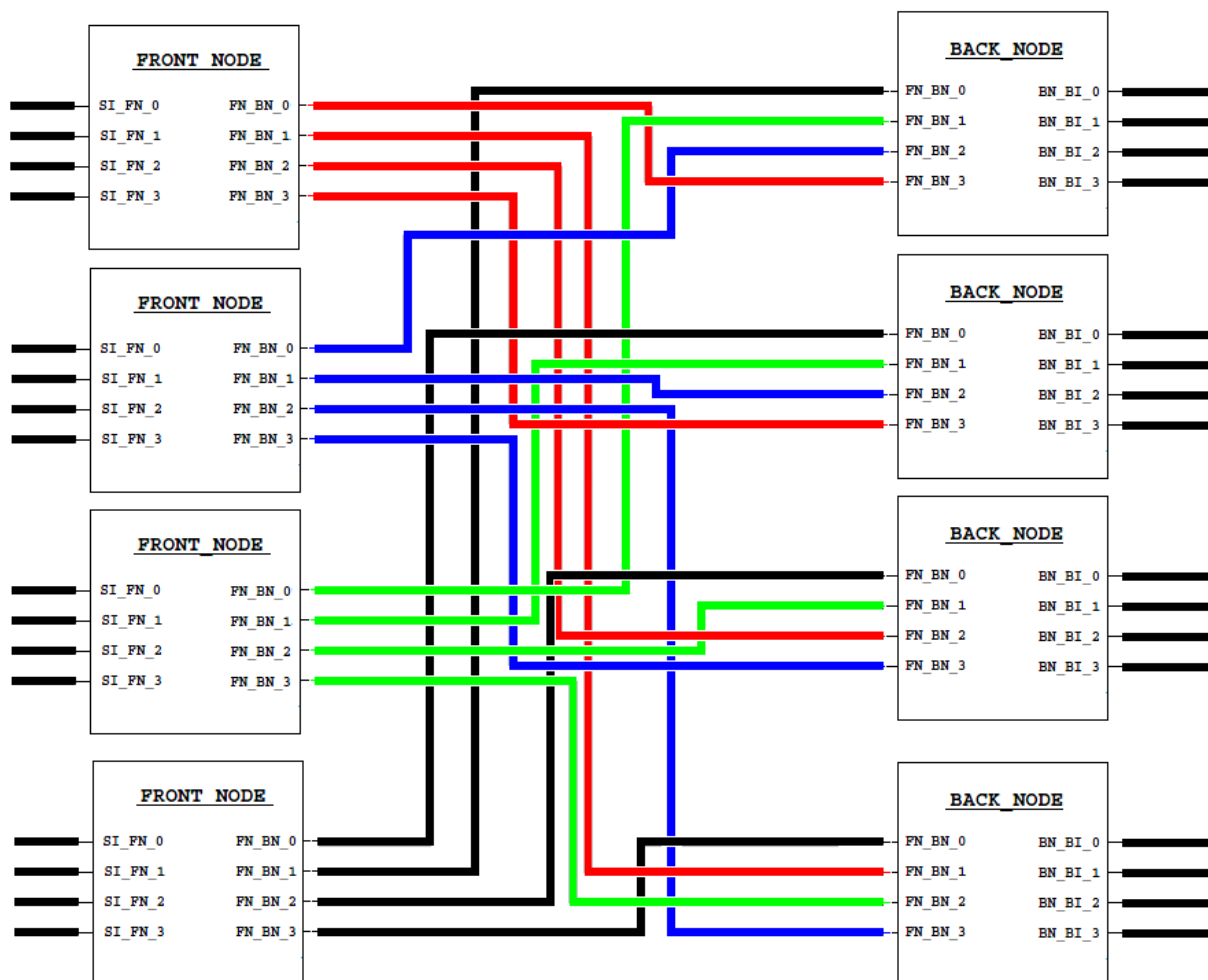


Figure 1 – UniBoard transceiver I/O

#### 2.1.1 Back node – backplane interface (BN\_BI) channel routing

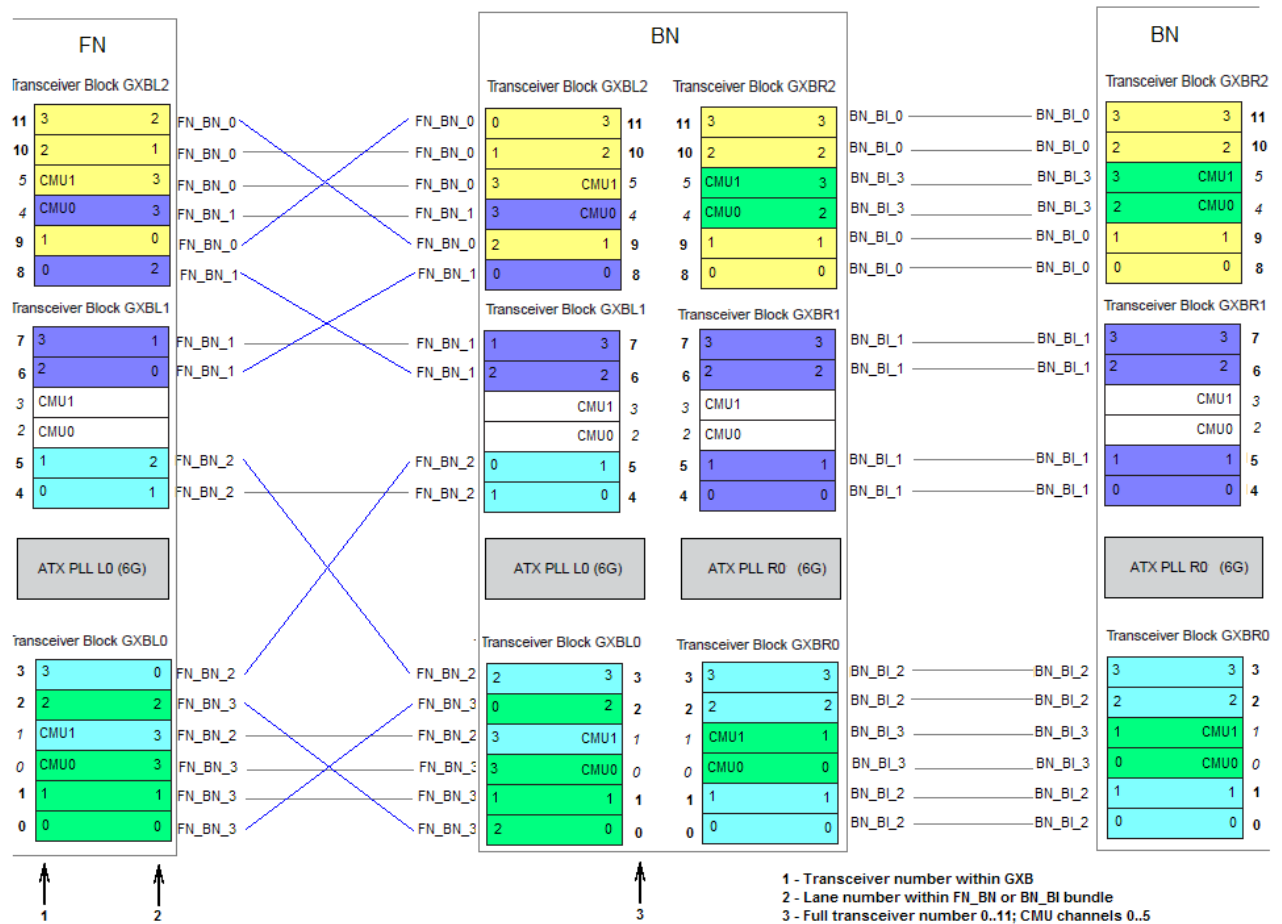
The back node – backplane interface (BN\_BI) consists of four bundles of four lanes, per FPGA. The bundles can interface via cables or backplane with other UniBoards or non-UniBoard hardware.

Bundle BN\_BI\_3 of each node consists of CMU transceivers only, the other three bundles are connected to four full transceivers each. The same applies to the SI\_FN bundles of the front nodes.

#### 2.1.2 Mesh (FN\_BN) channel routing

Each front node is connected to all four back nodes, by means of four bundles (FN\_BN\_#) of four lanes. As shown in Figure 1, this channel routing is not regular, i.e. FN\_BN\_0 connects to FN\_BN\_3 when looking at node 0 (upper front node), but this does not apply to all nodes. Each FN to BN routing is different. This will be addressed in the firmware, which contains a wrapper module to unify the channel routing. This adapter

will remap the channel routing depending on its host FPGA ID, so the same mesh firmware can be downloaded onto all eight FPGAs.



**Figure 2 – Front node – Back node – Backplane infrastructure**

### 2.1.3 Mesh (FN\_BN) lane routing

Each FPGA has six transceiver blocks, three on each side. As the back nodes are rotated 180 degrees with respect to the front nodes, both the front nodes and the back nodes utilize the three left transceiver blocks to connect to the mesh. This is depicted in Figure 2, in which the numbers indicated by 1 are the transceiver numbers, and the numbers indicated by 2 represent the mesh or BN\_BI lane numbers. For instance, when looking at transceiver block 2 (GXBL2) of a front node, lanes 0 (FN\_BN\_0\_0) to 3 (FN\_BN\_0\_3) of bundle FN\_BN\_0 connect to transceivers (in order) 1, 2, 3 and CMU1. This mapping scheme, three full transceivers plus one CMU transceiver, applies to all four mesh bundles.

To aid in the design of the PCB layout, the BN transceiver connections of lanes 0 and 2 of each FN\_BN bundle have been swapped with respect to the FN transceiver connections. These swapped lanes (represented by the blue crossed lines) can cause a problem similar to the irregular channel routing discussed in the previous paragraph. To make the firmware more independent of the node it is ran on, a simple rewiring entity can be placed for all TX lanes to swap the data in question before it is sent (and swapped back by the mesh).

Another concern regarding these swapped lanes was the possible implementation of XAUI on the mesh. Fortunately, these swapped lanes do not affect XAUI in terms of lane alignment and control words, as the alignment words are independent of the lanes they are carried on.

All transceivers of one side are also referred to by Altera (e.g. in the Stratix IV pin information files [2]) with the numbers 0..11 for the regular transceivers, and 0..5 for the CMU transceivers. On UniBoard, CMU0 and CMU1 of transceiver block 1 (GXBL1), or numbers 2 and 3 of the left side CMU transceivers, are not

Doc.nr.: ASTRON-RP-386

Rev.: 2.0

Date: 21-10-2010

Class.: Public

**UniBoard**

connected to the mesh. Also on the BN\_BI, the CMU channels of GLBR1 are not used as transceivers. The next chapter provides more details on these CMUs.

#### 2.1.4 Number and order of transceiver instances

To add transceivers to a VHDL firmware design, they must be generated by a Megafunction called ALTGX. When instantiating full (PMA+PCS) transceivers using the ALTGX Megafunction, one can use the following three approaches:

- Create one ALTGX instance containing the number of transceiver channels needed (e.g. 1x12)
- Create several ALTGX instances, each containing a number of channels (e.g. 3x4)
- Create a number of instances equal to number of channels needed (e.g. 12x1)

The first two approaches have limitations regarding the pin assignments to the serial transceiver I/O. For instance, if an ALTGX instance contains 4 channels, its channel I/O can only be mapped to pins that are physically connected to the same transceiver block. It is not possible to assign e.g. two channels to pins that connect to GXBL1, and the other two to pins that connect to GXBL2. This is because the MegaWizard generated logic can only control one transceiver block.

Similarly, if an ALTGX instance would contain five channels, one can divide the serial I/O over pins that connect to two transceiver blocks, because the MegaWizard generated control logic for two transceiver blocks (number of transceivers is greater than 4). The serial I/O cannot be divided over three transceiver blocks, so it cannot be divided equally over on side of the FPGA.

When generating one instance containing twelve transceivers, there is only one mapping possible, although the generated logic can control three transceiver blocks. This is caused by a second limitation: in this case, the first channel has to be mapped to physical transceiver 0 (connected to FN\_BN\_3[0]) and the last channel has to be connected to physical transceiver 11 (connected to FN\_BN\_0[2]). Not following this mapping scheme results in errors.

These limitations cause problems on UniBoard when flexible mapping of serial I/O is required. Therefore it is recommended to use one ALTGX instance for each individual channel. This provides full pin mapping flexibility and also facilitates the use of generics in VHDL, as only one MegaWizard generated file is instantiated for any number of transceivers.

#### 2.1.5 Reconfiguration module

For any number of (duplex or receiver-only) channels, a reconfiguration module (ALTGX\_RECONFIG) is necessary. This is a MegaWizard-generated module that provides offset cancellation (correcting analog voltage variations) for the receiver buffers and CDRs. This occurs once at power-up and is indicated by the assertion of the *busy* signal. The actual reconfiguration functionality is optional and can be used to reconfigure e.g. the data rate of each transceiver without having to reconfigure the FPGA.

Additionally, the reconfiguration module can be equipped with a feature called EyeQ control. This generates logic to control the EyeQ data samplers in each receiver, and a control interface that connects to user logic or a NIOS II processor.



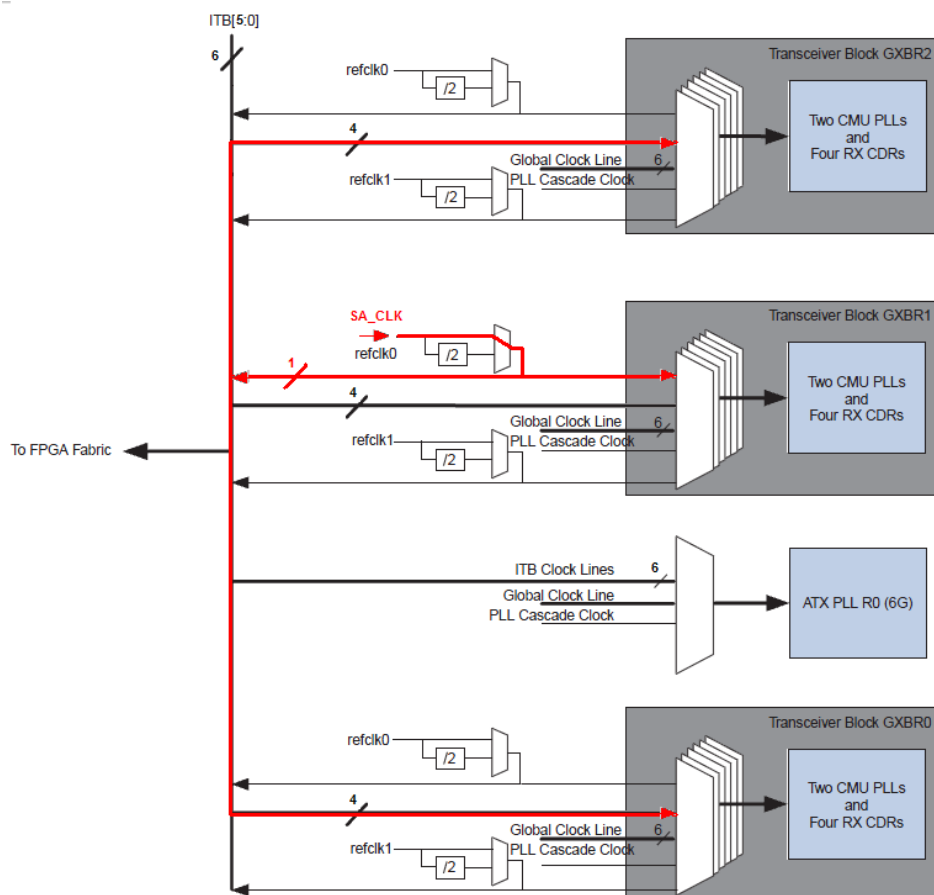
## 3 UniBoard transceiver clock sources

### 3.1 Clock Multiplier Unit (CMU) and Auxiliary Transmit PLL (ATX PLL)

As mentioned in the previous section, the CMU channels of transceiver blocks 1 are not used as transceivers. On UniBoard, two 156.56MHz reference clock signals are connected to two dedicated input pins on each FPGA:

- SA\_CLK for right side transceiver blocks, connected to the ref\_clk input of CMU2 (CMU0 of GXBR1)
- SB\_CLK for left side transceiver blocks, connected to the ref\_clk input of CMU2 (CMU0 of GXBL1)

These clock signals provide the reference clocks for the receivers, and are multiplied to provide the high speed transmitter clocks. This can be done by a CMU, or the ATX PLL: signals connected to any CMU's input reference clock pin can be forwarded to the ATX PLL on the same side of the device. These ATX PLLs support data rates up to 6.5Gbps. If the maximum data rate (8.5Gbps) is required, one must use a CMU PLL to create the high speed clock.



**Figure 3 – UniBoard transceiver clocking**

As shown in figure Figure 3, transceivers can use an input reference clock from the following sources:

- Its own reference clock (refclk0 or refclk1) pin
  - On UniBoard, only the transceiver PLLs of one transceiver block of each side (GXBR1 or GXBL1) can be clocked using this as input.
- Inter-Transceiver Block (ITB) clock lines

- Highlighted red in figure [ref], the ITB lines provide a way of carrying the clock signal that is generated in one transceiver block to the other transceiver blocks, or the ATX PLL. This is the way all UniBoard transceivers can be clocked by the SA\_CLK and SB\_CLK input clocks.
- Global clock lines
  - Global clock lines can be used to connect to dedicated differential clock inputs, located in non-transceiver banks, to the transceiver or ATX PLLs.
- PLL Cascade clock
  - This option can be used to provide input reference clocks generated by regular PLLs.

### 3.1.1 Available data rates

The following data rates (Mbps) are supported using an input reference clock of 156.25MHz, as is the case on UniBoard:

- 1250 [8x]
- 1562.5 [10x]
- 2500 [16x]
- 3125 [20x]
- 3906.25 [25x]
- 5000 [32x]
- 6250 [40x]

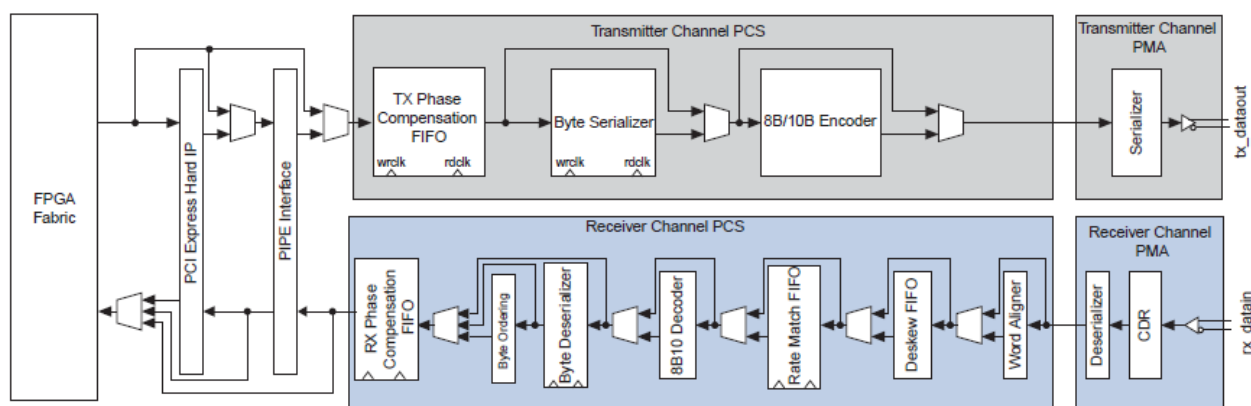
To run the CMU channels at their maximum data rate of 6500 Mbps, a range of input reference frequencies can be used, such as 130 Mhz and 162.5 Mhz.

The following data rates, ranging from 7000 up to the maximum data rate of 8500 Mbps, require a specific input reference clock frequency (and a CMU PLL):

- 7000 - 350 Mhz [20x]
- 7500 - 375 MHz [20x]
- 8000 - 400 Mhz [20x]
- 8500 - 425 Mhz [20x]

### 3.2 CMU (PMA-only) channels

A transceiver block (GXB) of the Stratix IV EP4SGX230KF40 contains six transceivers. Two of these are CMU (PMA-only) channels, meaning that they do not have a PCS. The PCS contains a byte (de)serializer that doubles the data width. With the data width doubled, the data rate to/from the FPGA fabric can be halved. Lacking this byte (de)serializer, the highest possible data rate for CMU channels is 6.5Gbps ( $16 * 10/8 * 325$ ), as the transceiver-FPGA fabric frequency limit lies at 325MHz. See Figure 4 for details.



**Figure 4 – Transceiver architecture [4]**

Any functionality provided by the PCS in full transceiver channels, has to be implemented in fabric when using the CMU channels as transceivers. This includes:

Doc.nr.: ASTRON-RP-386  
 Rev.: 2.0  
 Date: 21-10-2010  
 Class.: Public

- Phase compensation FIFO
- 8b/10b encoder/decoder (available in MegaWizard)
- Byte and word alignment
- Channel alignment for bonded modes such as XAUI

Altera does provide a 'soft' PCS that implements these features in fabric. This IP core will be discussed in the next section.

## 4 XAUI IP cores

### 4.1 XAUI PCS IP core

Altera provides a XAUI PCS core as part of the 10Gb Ethernet IP core. Using the 10Gb Ethernet Megafunction, the XAUI PCS core can be generated as a stand-alone module. Using this core has the following pros and cons:

Pros:

- Provides full PCS functionality
- Built-in reset sequence state machine
- Allows combining transceiver channels of multiple GXB's into one XAUI channel
- Can be used with RXAUI core to double throughput

Cons:

- Requires more logic cells

### 4.2 Reduced XAUI (RXAUI)

The prementioned RXAUI core is freely available from OpenCores or Marvell [1] as open-source Verilog code. It is a XAUI<->RXAUI adapter that converts XAUI to RXAUI and vice versa. RXAUI uses two lanes at 6.25Gbps instead of four lanes at 3.125Gbps (XAUI).

This core is written by Marvell to support their future product line of RXAUI Ethernet transceivers, and is tagged at OpenCores as:

- Mature, Design done
- FPGA proven
- ASIC proven

#### 4.2.1 Data switch

Compared to XAUI, one disadvantage of using RXAUI is the fact that it provides two XAUI interfaces between each node instead of one. Depending on the application, this might mean some extra logic (data switch) has to be coded to distribute data over these two XAUI bundles.

## 5 Comparison of transceiver configurations and modes

There are several problems introduced by the combination of using CMU channels and the scattering of lanes across multiple transceiver blocks. This chapter discusses these problems and the possible solutions.

### 5.1 Available modes

The ALTGX Megafunction supports a number of available protocols. Of these protocols, the following three protocols are of interest:

- Basic
- Basic (PMA only)
- XAUI

Basic simply means no protocol is maintained other than 8b/10b encoding, which is not mandatory. This simplifies its usage and does not constrain achievable data rates by protocol.

### 5.2 Limitations

Disadvantages of using PMA-only (CMU) channels:

- PCS components must be implemented in fabric
- Data rate limited to 6.5Gbps
- Hard XAUI is not possible
- Combination hard+soft XAUI is not possible (appears to be a Quartus issue)
- Adaptive Equalization feature not available

Disadvantages of using bundles of lanes across multiple transceiver blocks:

- Hard XAUI is not possible

One solution to the above would be to use the full channels only:

- FN\_BN:
  - 4 bundles of 3 lanes
  - 1 lane of each FN\_BN bundle remains unused
  - No bonded modes possible
- BN\_BI:
  - 3 bundles of 4 lanes
  - 1 BN\_BI bundle remains unused
  - 3 bonded (XAUI) channels possible

For the mesh, when only full transceivers are used, the theoretical data throughput limit lies at  $4 \times 4 \times 3 \times 8.5 \times 8/10 = 326.4$  Gbps. This is calculated the following way:

$$[nof\ FPGAs] * [nof\ bundles] * [nof\ active\ transceivers\ per\ bundle] * [data\ rate\ per\ transceiver] * [coding\ scheme]$$

For the BN\_BI the same maximum speed applies, except when XAUI is used: three XAUI channels per node offer a throughput of  $4 \times 3 \times 4 \times 3.125 \times 8/10 = 120$  Gbps.

To address the above limitations, and be able to use all channels, including the four remaining CMU channels:

- The PCS and channel alignment functionality could be implemented by hand (if XAUI is not desired), or

- The XAUI PCS IP core could be used.

Both would result in more logic cells being occupied, but with all transceivers employed, the maximum throughput in any mode (Basic, XAUI, RXAUI) would be higher.

If XAUI is used on the mesh, only soft PCS can be used. This is because XAUI with hard PCS can only be used if all four XAUI lanes are connected to the same transceiver block.

Connecting the RXAUI adapter between the XAUI PCS core and the PMAs would double the UniBoard's mesh and backplane data rate (in XAUI mode) from 160 Gbps to 320 Gbps. This increase in speed enables fast XAUI, optionally carrying 10GbE, as a possible mesh and backplane protocol in case the standard XAUI data rate proves to be insufficient.

### 5.3 Overview

Table 1 gives an overview of the achievable data rates in different modes. The PCS column shows whether or not it is possible to use the hard PCS, a (hand-coded) soft PCS or the XAUI soft PCS IP for a particular mode.

	Configuration			GX type	Mode	PCS	Gbps/GX	Total throughput	Total throughput (Overclocked)	Resource usage (one FPGA side)	
	GX/side	Bundles	GX/bundle							ALUTS	Registers
FN_BN	12	4	3	PMA+PCS	Basic	Hard PCS	8,5	326,4	n.a.	2,0%	2,7%
	16	4	4	PMA	Basic	Soft PCS	6,5	332,8	n.a.	-	-
	16	4	4	PMA	RXAUI	XAUI PCS IP	6,25	320	332,8	-	-
	16	4	4	PMA	XAUI	XAUI PCS IP	3,125	160	332,8	5,6%	4,8%
	16	4	4	PMA	10GbE	XAUI PCS IP	3,125	160	332,8	17,8%	16,4%
BN_BI	12	3	4	PMA+PCS	Basic	Hard PCS	8,5	326,4	n.a.	2,0%	2,7%
	12	3	4	PMA+PCS	XAUI	Hard PCS	3,125	120	144	-	-
	12	3	4	PMA+PCS	XAUI	XAUI PCS IP	3,125	120	249,6	4,2%	3,6%
	16	4	4	PMA	Basic	Soft PCS	6,5	332,8	n.a.	-	-
	16	4	4	PMA	RXAUI	XAUI PCS IP	6,25	320	332,8	-	-
	16	4	4	PMA	XAUI	XAUI PCS IP	3,125	160	332,8	5,6%	4,8%
	16	4	4	PMA	10GbE	XAUI PCS IP	3,125	160	332,8	17,8%	16,4%

**Table 1 – Comparison of protocols, data rates and PCS types**

### 5.4 Resource usage

Table 1 provides the resource usage of the hard PCS and the XAUI PCS IP core in varying configurations. The hard PCS resource usage results from the synthesis of the tr\_nonbonded module [5], and includes reset and alignment logic, 12 transceivers, 12 TX pseudo random number generators and 12 RX monitors. The resource usage of the XAUI PCS IP is taken from the 10-Gbps Ethernet Reference Design document [3].

### 5.5 Hard + soft components combined

As mentioned in paragraph 5.2, it is not (yet) possible to use a combination of hard and soft XAUI cores. It is however possible to combine full channels with PMA-only channels, as long as they are used in a non-bonded mode. As using PMA-only channels requires certain components to be implemented in fabric, and introduces some timing issues, this setup only adds value when the data rate of a full-channel-only configuration turns out below requirements.

### 5.6 XAUI at higher data rates

When interfacing between UniBoards or UniBoard FPGAs, it is not necessary to limit the data rate to the 3.125Gbps defined by the protocol. There are two ways to implement XAUI. The following paragraphs list the possibilities of running XAUI at higher speeds when using XAUI with soft or hard PCS.

### 5.6.1 Hard XAUI

The ALTGX MegaWizard provides a clean way to raise the data rate to 3.75Gbps per channel, provided that an input frequency of 187.5MHz is available.

### 5.6.2 Soft XAUI PCS IP

The soft XAUI PCS IP, that can be generated using the 10GbE Megafunction, does not directly support higher speeds. However, it is possible to access the ALTGX Megafunctions used by the 10GbE Megafunction. In these ALTGX Megafunctions, the speed can be set to a maximum of 6.5Gbps (because PMA-only channels are used). Everything that is instantiated by these ALTGX Megafunctions will be updated accordingly. This includes the ATX PLL output and the RX CDR PLL. The soft PCS itself however, lies the highest in the hierarchy and is therefore not updated with the new speed settings. This means that the PMA and its clock sources should be correctly configured for the higher speed, i.e. 6.25Gbps, but the PCS would still work at 3.125Gbps (20 bit SERDES \* 156.25MHz), making it incompatible with the PMA. A solution to this could be to clock the PCS with twice the frequency of the PMA clock, i.e. 312.5MHz, which is overclocking instead of configuring.

## 6 8b/10b encoding

An 8b/10b encoder/decoder can be MegaWizard-generated and placed in fabric, or the PCS 8b/10b encoder/decoder can be used. If achievable data rates are lower than necessary, the 8b/10b coding could be entirely omitted. This would only apply to the mesh interface, because all FPGAs on the UniBoard use the same source for their transceiver reference clocks. When interfacing with other hardware (via the BN\_BI interface), the reference clocks of receivers and transmitters will differ from each other in terms of phase offset and drift, necessitating 8b/10b encoding to provide enough transitions for clock extraction at the receiver.

Another reason why the 8b/10b scheme is needed for the BN\_BI is the possible build-up of charge in transmission cables due to the transmission of long streams of non-transient data, causing bit errors.

Omitting 8b/10b encoding would have the following consequences:

- 20% less overhead
- Less fabric occupied if PMA-only channels are used
- Wizard-generated word alignment no longer available
- Byte alignment (PCS) block no longer available
- No support for protocols requiring 8b/10b coding scheme (e.g. XAUI)
- CDR operation requires attention
- Adaptive Equalization feature no longer available

### 6.1.1 CDR operation with 8b/10b coding

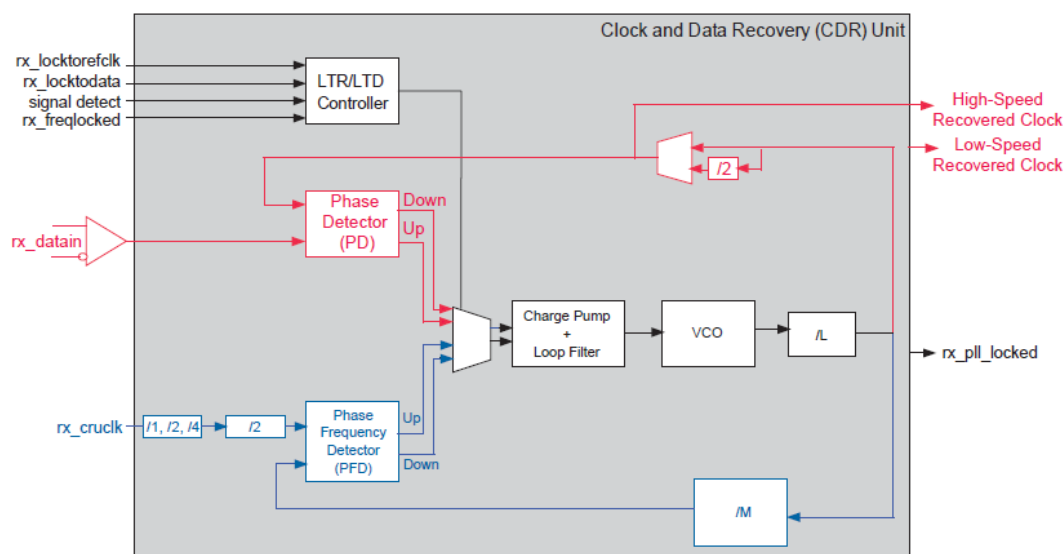
The Clock and Data Recovery block (CDR) is located in the receiver PMA. It operates in either of the following modes:

- LTR: Lock To Reference mode
- LTD: Lock To Data mode

To establish a transceiver link, the CDR must be in LTD mode (depicted in red in Figure 5) and use the recovered clock to sample the data. Before LTD mode is possible, the CDR must first successfully lock to the reference clock (LTR mode, depicted in blue).

In LTR mode, the CDR locks to the input reference clock (rx\_crucclk in Figure 5). Once locked, the CDR must be set to LTD mode. This can be done automatically, or manually, using input signals like rx\_locktodata. In LTD mode, the CDR extracts the clock from the incoming serial data. This recovered clock high speed clock, and its divided low speed recovered clock are forwarded to the other receiver blocks.





**Figure 5 – Clock and Data Recovery block**

## 6.1.2 CDR operation without 8b/10b coding

Without 8b/10b encoding, the CDR can lock to the reference clock (LTR mode) as it would with 8b/10b encoding. However, the LTD mode would require sufficient transitions in the input serial stream to do the initial locking. After the CDR is locked to data, and remains in LTD mode, the PLL feedback loop must keep providing a correct clock, even when the serial input remains low for a long time. Whether or not the PLL will remain stable depends on the type of phase detector and charge pump.

## 7 Conclusion

A comparison of all possible modes for both the FN\_BN and the BN\_BI interface shows that, in theory, high data rates (250 – 330Gbps) can be achieved in any mode.

RXAUI is an interesting add-on, but if XAUI works properly at higher speeds (up to 6.5Gbps per transceiver), RXAUI no longer has any advantages.

The soft XAUI PCS IP offers the most flexible XAUI implementation, giving us the options to use RXAUI and overclocked XAUI, both also enabling 20 Gigabit Ethernet (2\*10 in RXAUI or 1\*20 in overclocked XAUI mode) from node to node.

When XAUI is not required, nearly the same data rates can be achieved using the transceivers in Basic non-bonded mode, if the transceivers work properly at 8.5Gbps each. In that case, 12 full transceivers are used per FPGA side. The four CMU (PMA only) transceivers remain unused.

In case the achievable data rates turn out lower than necessary, the four remaining CMU channels per side could be incorporated into the design to provide more throughput. Also, omitting the 8b/10b encoding scheme allows some play regarding achievable data rates, at the expense of 8b/10b dependant protocol (e.g. XAUI) compliance.