EP-ESE-BE

CLKS_IN
GBT_TX_IN
MGT_IN
GBT_RX_IN
GBT Bank
CLKS_OUT
GBT_TX_OU
MGT_OUT
GBT_RX_OU

GBT-FPGA User Guide

Version: 1.4

2016.07.21

DRAFT

The GBT-FPGA team: S. Baron (Sophie.Baron@cern.ch), J. Mendez (julian.mendez@cern.ch)

GBT-FPGA support: GBT-FPGA-support@cern.ch

Web site: *https://espace.cern.ch/GBT-Project/GBT-FPGA*

SVN repository: *https://svn.cern.ch/reps/ph-ese/be/gbt_fpga*

Document History

- **V1.4:** Added support for Xilinx Kintex Ultrascale
- **V1.3:** Added support for Arria 10
- **V1.22:** Added support for Vivado and removing 8b10b support
- **v1.21, 2015.02.26:** Added section 2.1.8. Completed sections 2.1.6.a, 2.4.1,d, 2.4.3.b, 2.4.4, 2.4.5, 2.4.6 and 2.5. Minor corrections.
- **v1.01, 2014.04.15:** Minor corrections.
- **v1.00, 2014.04.11:** First draft in the document history.

The GBT-FPGA team

- **Sophie Baron** (Sophie.Baron@cern.ch)
- **Julian Mendez (**julian.mendez@cern.ch**)**
- **GBT-FPGA-support** (GBT-FPGA-support@cern.ch)

Table of Contents

Document History 3

The GBT-FPGA team 3

Table of Contents 5

1. General Introduction 9

1.1. The GBT and the Versatile Link Projects 9

1.2. The GBT-FPGA Project Mandate 10

2. The GBT-FPGA Core 11

2.1. Introduction 11

2.1.1. GBT-FPGA Core Overview 11

2.1.2. Standard vs Latency-Optimized 12

2.1.3. Single-Link vs Multi-Link Instantiation 12

2.1.4. Data Frame & Encodings 13

2.1.5. Multiple Platforms 14

2.1.6. Typical Implementation Resources 14

2.1.6.a. Logic Resources Utilization: 14

2.1.6.b. Clocking Resources Utilization 15

2.1.7. Usual Data Latency 15

2.1.8. Clocks Phase & Data Latency Drift VS Temperature 15

2.2. Entities 17

2.2.1. GBT Bank 17

2.2.2. GBT Link 17

2.2.3. GBT Tx 18

2.2.4. GBT Rx 18

2.2.5. Multi-Gigabit Transceiver (MGT) 18

2.3. GBT-FPGA Project Files 19
2.3.1. Common vs Device Specific Source Files 19
2.3.2. The GBT-FPGA SVN Repository 19
2.4. Implementation 20
2.4.1. Getting Started 20
2.4.1.a. Check the Structure of the Folders 20
2.4.1.b. Open One of the GBT-FPGA Example Design Projects 20
2.4.1.c. Check the Source Files 21
2.4.1.d. Implement and Run one of the GBT-FPGA Example Design Projects 21
2.4.2. GBT Bank configuration 22
2.4.2.a. Using user setup package 22
2.4.2.b. Using gbt_bank generic parameters 23
2.4.3. Adding the GBT-FPGA Core to Your Project 23
2.4.3.a. Running TCL Scripts: Common Steps for Different FPGA vendors 23
2.4.3.b. Running TCL Scripts 25
2.4.4. The Particularities of the Latency-Optimized Version 26
2.4.4.a. Potential Uncertainty Points in the Latency-Optimized version 26
2.4.4.b. Unnecessary Clock Domains (UCD) 27
2.4.4.c. Non Latency Deterministic Components (NLDC) 28
2.4.4.d. Phase Relationship Between Clocks (PRBC) 28
2.4.4.e. Clock Frequency Multiplication/Division (CFMD) 29
2.4.4.f. Clock & Data Recovery (CDR) 29
2.4.4.g. Calibration 30
2.4.5. Clocks Scheme 31
2.4.5.a. Clocks Scheme for Single-Link Instantiation 31
2.4.5.b. Clocks Scheme for Multi-Link Instantiation 32
2.4.6. Timing Closure 33
2.4.6.a. Timing Closure in Xilinx FPGAs 33

2.4.6.b. Timing Closure in Altera FPGAs 33
2.4.7. Resets Scheme 34
2.5. Operating the GBT-FPGA Core 35
2.5.1. Clock Ports 35
2.5.1.a. Common Clock Ports 35
2.5.2. Device Specific Clock Ports 36
2.5.3. GBT Tx Ports 37
2.5.4. GBT Rx Ports 37
2.5.5. MGT Ports 38
3. Example Designs 45
3.1. Example Design Overview 45
3.2. Running the Example Designs 48
3.2.1. Running the Example Design in Xilinx FPGAs using ChipScope 48
3.2.2. Running the Example Design in Altera FPGAs using In-system Sources and Probes 48
4. FPGA Particularities & Hardware Recommendations 50
4.1. FPGA Particularities 50
4.1.1. Xilinx FPGAs 50
4.1.1.a. Virtex 6 50
4.1.1.b. Kintex 7 & Virtex 7 50
4.1.2. Altera FPGAs 50
4.1.2.a. Cyclone V GT 50
4.1.2.b. Stratix V GX 51
4.2. Hardware Recommendations 51
5. References 52
APPENDIX A: Frequently Asked Questions 54
APPENDIX B: GBT Bank Block Diagram 58
APPENDIX C: Known Issues 62

1. General Introduction

1.1. The GBT and the Versatile Link Projects

The GBTx is a radiation tolerant chip that can be used to implement multipurpose high speed (3.2-4.48 Gbps user bandwidth) bidirectional optical links for high-energy physics experiments.

Logically the link provides three "distinct" data paths for Timing and Trigger Control (TTC), Data Acquisition (DAQ) and Slow Control (SC) information. In practice, the three logical paths do not need to be physically separated and are merged on a single optical link as indicated in Figure 1. The aim of such architecture is to allow a single bidirectional link to be used simultaneously for data readout, trigger data, timing control distribution, and experiment slow control and monitoring. This link establishes a point-to-point, optical, bidirectional (two fibres), constant latency connection that can function with very high reliability in the harsh radiation environment typical of high energy physics experiments at LHC.

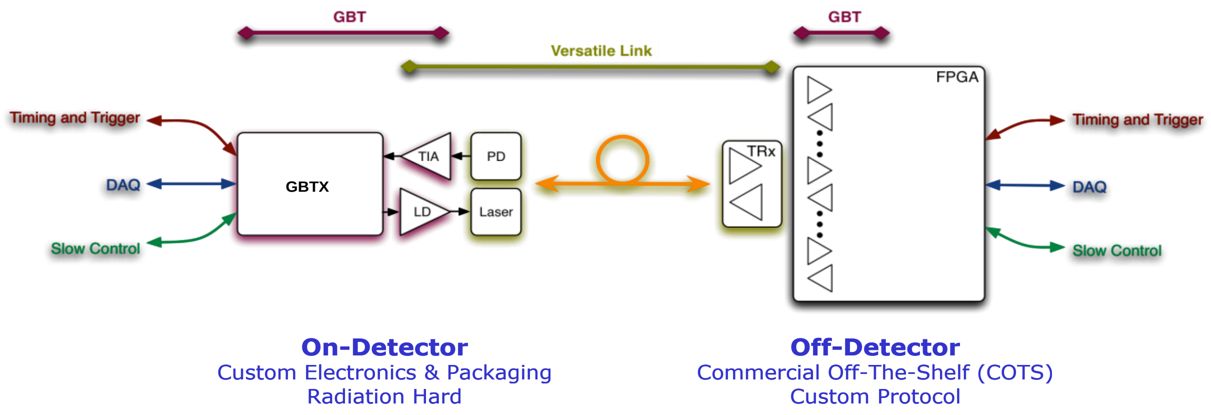


Figure 1: Link architecture with the GBT chip set and the Versatile Link opto-components

The development of the proposed link is conceptually divided into two distinct but complementary parts: the GBT link chips and the Versatile link opto components. The versatile link selects and qualifies appropriate fibres and opto-electronic components for use in radiation. The GBT develops and qualifies the required radiation hard ASICs.

The link is implemented by a combination of custom developed and Commercial-Off-The-Shelf (COTS) components. In the counting room the receiver and transmitters are implemented using COTS components and FPGA's. Embedded in the experiments, the receivers and transmitters are implemented by the GBT chipset and the Versatile Link optoelectronic components. This architecture clearly distinguishes between the counting room and front-end electronics because of the very different radiation environments. The on-detector front-end electronics works in a hostile radiation environment requiring custom made components. The counting room components operate in a radiation free environment and can be implemented by COTS components. The use of COTS components in the counting house allows this part of the link to take full advantage of the latest commercial technologies and components (e.g. FPGA) enabling efficient data concentration and data processing from many front-end sources to be implemented in very compact and cost efficient trigger and DAQ interface systems.

The firmware required to allow these FPGAs to communicate with the GBTx chipset over the versatile link is handled by the GBT-FPGA project.

1.2. The GBT-FPGA Project Mandate

Initiated in 2009 to emulate the GBTx serial link and test the first GBTx prototypes, the GBT-FPGA project developed first to provide the users with a basic "starter kit" allowing them to get used to the GBTx protocol. As the features of the GBTx ASIC inflated during design phase together with the users' requirements, the GBT-FPGA project followed naturally and grew up as well.

This GBT-FPGA core is now a full library, targeting FPGAs from ALTERA and XILINX, allowing the implementation of one or several GBT links of 2 different types: "**Standard**" or "**Latency-Optimized**" (providing low, fixed and deterministic latency either on Tx, Rx or on both). These links can be also configured to provide any encoding mode offered by the GBTx: the "**GBT-Frame**" mode (Reed-Solomon based) or the "**Wide-Bus**" mode (no encoding).

The GBT-FPGA core is freely available from SVN, and can be instantiated on Back-End FPGAs, but it can also be used as a GBTx emulator for the serial link. As such, the core is fully configurable in any of the above described options.

For obvious reasons, the GBT-FPGA core will not offer firmware versions for each FPGA type on the market. It targets the main vendors and the main series. The design effort on new series is foreseen to stop during 2014. The evolution of the core to follow-up the technology will thus depend on users' contributions.

2. The GBT-FPGA Core

2.1. Introduction

2.1.1. GBT-FPGA Core Overview

In order to facilitate the in-system implementation and the user support of the GBT-FPGA, the different components of the GBT-FPGA Core are integrated in a single module called "**GBT Bank**" (see Figure 2 and Figure 3).

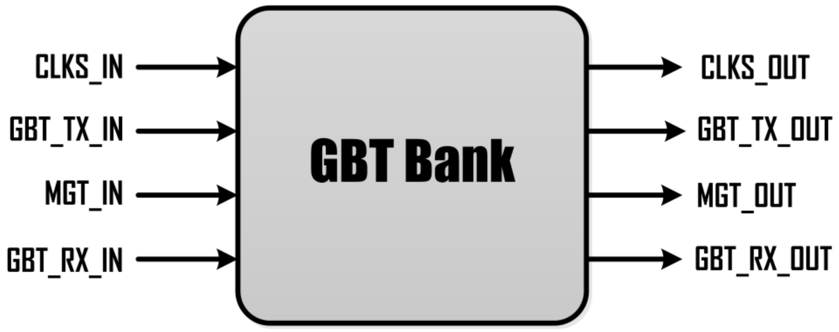
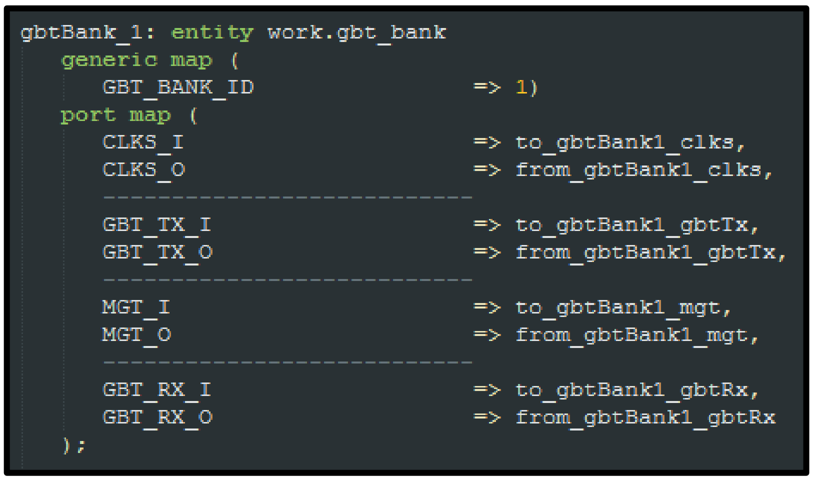


Figure 2: GBT Bank module

Most of these components are common for the different platforms. The GBT Bank may include several "**GBT Links**" (see 2.1.3). Each GBT Link is composed by a GBT Tx, a GBT Rx (both together will be referred to as "GBT Logic") and a Multi-Gigabit Transceiver (MGT). The clocking resources are external to the GBT Bank so the user can connect the different clocks as desired.



```
gbtBank_1: entity work.gbt_bank
   generic map (
      GBT_BANK_ID                 => 1)
   port map (
      CLKS_I                      => to_gbtBank1_clks,
      CLKS_O                      => from_gbtBank1_clks,
      ---------------------------
      GBT_TX_I                    => to_gbtBank1_gbtTx,
      GBT_TX_O                    => from_gbtBank1_gbtTx,
      ---------------------------
      MGT_I                       => to_gbtBank1_mgt,
      MGT_O                       => from_gbtBank1_mgt,
      ---------------------------
      GBT_RX_I                    => to_gbtBank1_gbtRx,
      GBT_RX_O                    => from_gbtBank1_gbtRx
   );
```

Figure 3: GBT Bank VHDL instantiation

The number of GBT Links of the GBT Bank as well as the two encoding schemes proposed by the GBTx ASIC ("GBT-Frame" (Reed-Solomon) and "Wide-Bus") and the two types of optimization ("Standard" and "Latency-Optimized"), may be selected at implementation time through a single file (**GBT User Setup File**).

2.1.2. Standard vs Latency-Optimized

Trigger related electronic systems in High Energy Physics (HEP) experiments, such as Timing Trigger and Control (TTC), require a fixed, low and deterministic latency in the transmission of the clock and data to ensure correct event building. On the other hand, other electronic systems that are not time critical, such as Data Acquisition (DAQ), do not need to comply with this requirement. The GBT-FPGA project provides two types of implementation for the transmitter and the receiver: the "**Standard**" version, targeted for non-time critical applications and the "**Latency-Optimized**" version, ensuring a fixed, low and deterministic latency of the clock and data (at the cost of a more complex implementation).

If latency is not an issue, the **Standard version** is strongly recommended.

Table 1: Standard VS Latency-Optimized

	Standard	Latency-Optimized
Latency	Non Fixed, Higher, Non Deterministic	Fixed, Low, Deterministic
Logic Resources Utilization	Low	Low
Clocking Resources Utilization	Low	**High**
Clock Domain Crossing	Don't Care	**Critical**
Implementation	Simple	Complex

2.1.3. Single-Link vs Multi-Link Instantiation

One of the aims of the GBT-FPGA Core is to facilitate the implementation of single-link and multi-link GBT-based systems. For that reason, the GBT-FPGA Core has been designed in such a way that for implementing either single-link or multi-link GBT-based systems, it is only necessary to set the number of desired GBT Links through the GBT User Setup File (see **Error! Reference source not found.**) and to provide the external required resources (e.g. clocking resources (see 2.4.5), reset resources (see 2.4.7), etc.).

The maximum number of GBT Links per GBT Bank is limited by the architecture of the targeted FPGA (see **Error! Reference source not found.**). The concept behind the design of the GBT Core is to keep each GBT Bank as an independent entity in terms of both logic and clocking resources. In applications that require more GBT Links than the number provided by a single GBT Bank, it is possible to add more GBT Links by instantiating GBT Banks in parallel. The maximum number of GBT Banks is also device dependent

Table 2: Maximum number of GBT Links per GBT Bank for different FPGAs.

Vendor	Device	Max. Number of GBT Links per GBt Bank	GBT Bank span into FPGA
Xilinx	Virtex 6	4	One GTX Quad
	7 Series	4	One GTX Quad
Altera	Cyclone V GT	3	Half GT Bank (3 GT channels)
	Stratix V GX	6	GX Bank (6 GX channels)
	Arria V GX	*4*	*Half GX Bank (3 GX channels) + 1 GX channel*(*)
	Arria 10 GX	6	GX Bank (6 GX channels)

(*) The extra GX channel is for spanning the four Application SFPs using one GBT Bank on the VFC-HPC.

2.1.4. Data Frame & Encodings

As previously mentioned, the GBT-FPGA supports the two available encoding schemes proposed by the GBTx.

The "**GBT-Frame**", shown in Figure 4, adopts the Reed-Solomon that can correct bursts of bit errors caused by Single Event Upsets (SEU). This encoding scheme can be used for Data Acquisition (DAQ), Timing Trigger & Control (TTC) and Experiment Control (EC).

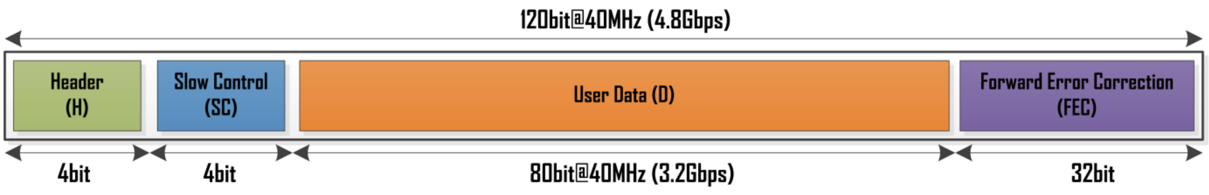


Figure 4: GBT-Frame encoding frame

For the "**Wide-Bus**", shown in Figure 5, the FEC field is fully replaced by user data at the cost of no error detection nor correction capability. This encoding scheme can **only** be used for DAQ and EC **in the uplink direction**.

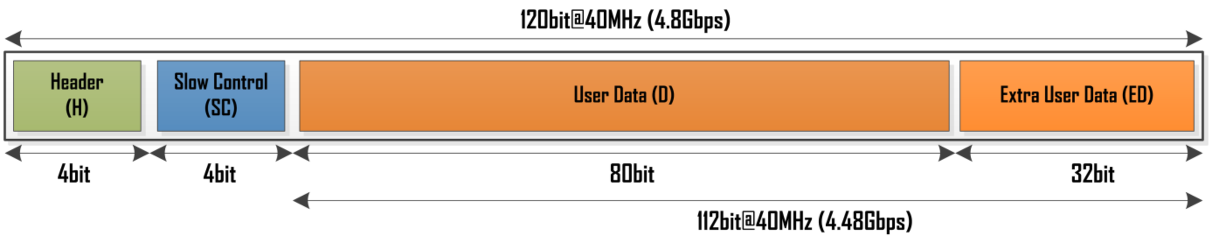


Figure 5: Wide-Bus encoding frame

2.1.5. Multiple Platforms

The GBT-FPGA Core already available for different FPGAs of the two main vendors, Xilinx and Altera.

- Xilinx: Virtex 6, Kintex 7, Virtex 7
- Altera: Cyclone V GT, Stratix V GX, Arria 10 GX

These versions may be used as a reference by the user for porting the GBT-FPGA Core to other FPGA devices and vendors.

2.1.6. Typical Implementation Resources

2.1.6.a. Logic Resources Utilization:

The logic resources utilization of one GBT Bank instantiating one GBT Link is shown in Table 3 and Table 4 for Xilinx Kintex 7 and Altera Cyclone V GT respectively.

Table 3: Logic resources utilization in Xilinx Kintex 7

Xilinx (Kintex7: XC7K325T)		
Resources	**Standard (%)**	**Latency-Optimized (%)**
LUT		
FD_LD		
BMEM		
GTX		

Table 4: Logic resources utilization in Altera Stratix V GX (6 links in standard mode)

Altera (Stratix V GX: 5CGTFD9E5F35C7N)		
Resources	**Standard (%)**	**Latency-Optimized (%)**
ALM		
Register		
Mem (M10K)		
GT		

The logic resources utilization per GBT Link is very low compared with the amount of resources of the latest FPGAs.

2.1.6.b. Clocking Resources Utilization

The clocking resources are tightly related to the user designs:

- device,
- optimization modes (standard or latency optimized),
- number of links,
- general clocking scheme of the system,

Have a strong impact on the number of clocking resources (Buffers, PLLs etc) required for the implementation.

2.1.7. Usual Data Latency

A preliminary data latency study of the Latency-Optimized version has been carried out, obtaining the values showed in Table 5.

Table 5: Preliminary latency measurement results

Latency of the Latency-Optimized GBT Link (Virtex 6)			
	GBT Logic	**MGT (GTX)**	**Total**
GBT Link Tx	29.2ns	18.7ns	47.9ns
GBT Link Rx	54.2ns	28.2ns	82.4ns

2.1.8. Clocks Phase & Data Latency Drift VS Temperature

A fully automated testbench was used to verify the determinism in clock phase and data latency as well as to measure the drift of the clocks phase versus temperature, making sure that the previously mentioned drift does not affect the reliability of the system.

For the test, two FPGA-based development kits from Xilinx (KC705) were used, both loading a test firmware featuring the GBT-FPGA Core in "Latency-Optimized" version. One of the development kits was placed inside a thermal chamber whereas the other was placed outside the chamber, so only one board was affected by the temperature variation thus facilitating the identification of critical points.

Firstly, both systems were calibrated at 20 ºC and then the measurements were performed in two steps. In the first step, the skew between the different clocks and the latency between data paths were measured using an oscilloscope while performing power cycles of the boards (about 10.000 cycles) at constant temperature (20 ºC). In the second step, in addition to the previous test, repetitive temperature cycles from 10 to 60 to 10 ºC (in steps of 0.3 ºC) were performed on the board hosted within the thermal chamber. The temperature cycles were repeated during eight hours.

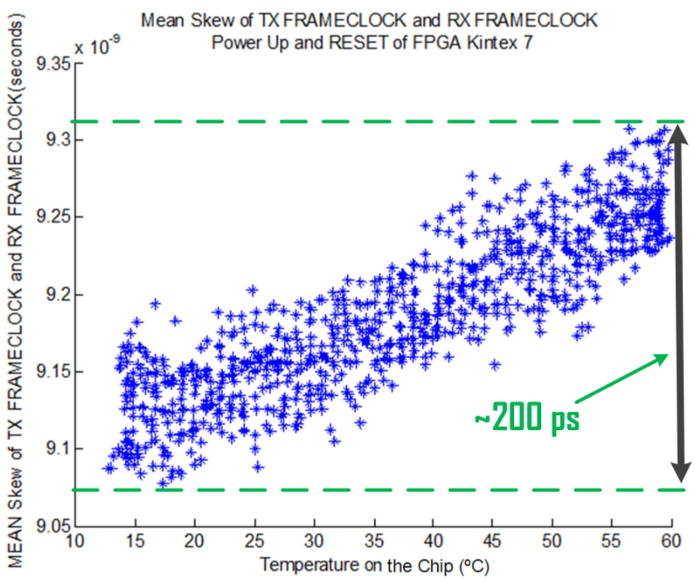


Figure 6: TX_FRAMECLK and RX_FRAMECLK phase drift VS temperature.

The result of the test showed a maximum clock and data phase uncertainty of 100 ps peak-to-peak between system powers up and a maximum phase drift of 200 ps for a temperature variation of 50 °C. An example of latency versus temperature measurement is shown in Figure 6.

Since temperature variations in the Back-End crates are foreseen to be very low and with a very slow drift. For that reason, the maximum phase drift expected during operation is about 20 ps, value within the specification for LHC experiments.

2.2. Entities

2.2.1. GBT Bank

The GBT Bank is the top module of the GBT-FPGA Core. That can integrates several GBT Links (see 2.1.3) and the different ports required for the operation of the GBT Links

A simplified block diagram of a GBT Bank instantiating two GBT Links is shown in Figure 7.

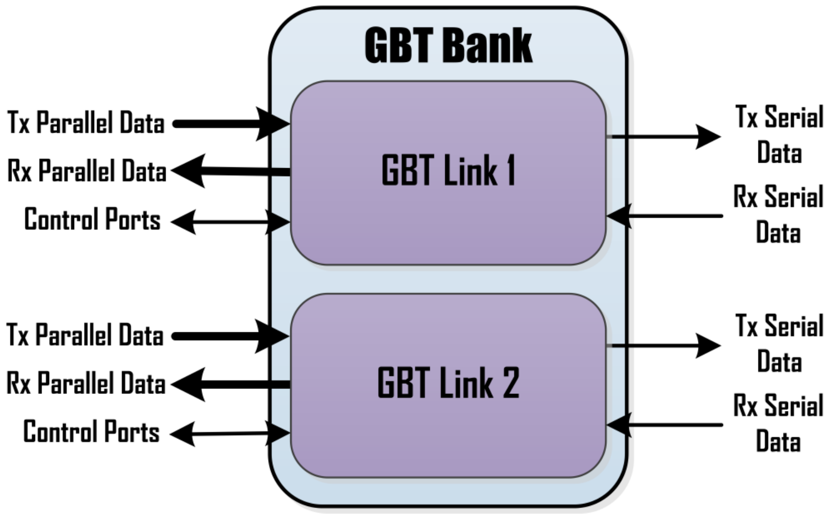


Figure 7: GBT Bank simplified block diagram

A detailed block diagram of the GBT Bank is shown in APPENDIX B:.

2.2.2. GBT Link

The GBT Link is the actual channel of the link. It is composed by a GBT Tx (that scrambles and encodes the transmitted parallel data), a Multi-Gigabit Transceiver (MGT) (that serializes, transmits, receives and de-serializes the data) and a GBT Rx (that aligns, decodes and descrambles the incoming data stream).

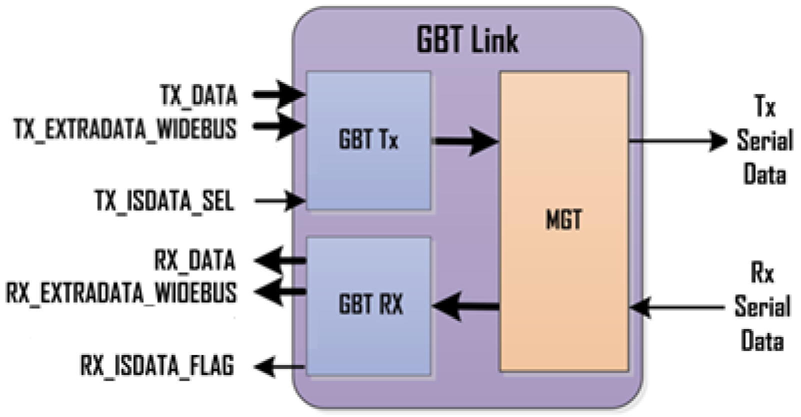


Figure 8: GBT Link simplified block diagram

2.2.3. GBT Tx

A simplified block diagram of the GBT Tx highlighting its main components is shown in Figure 9.

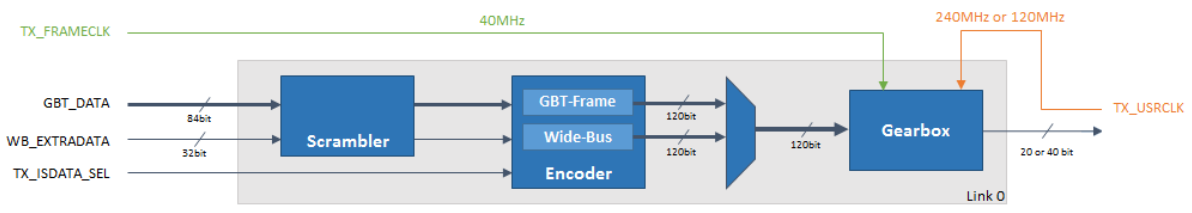


Figure 9: GBT Tx simplified block diagram

2.2.4. GBT Rx

A simplified block diagram of the GBT Rx highlighting its main components is shown in Figure 10.

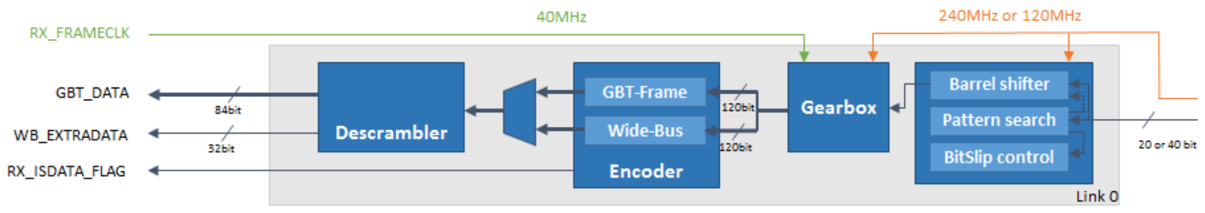


Figure 10: GBT Rx simplified block diagram

2.2.5. Multi-Gigabit Transceiver (MGT)

The Multi-Gigabit Transceiver (MGT) is a vendor and device specific Serializer/Deserializer (SerDes). In order to facilitate the integration with the GBT Logic, the MGT is wrapped with a HDL file (depicted by Figure 11) featuring a common interface with the GBT Logic. Besides this common interface, the MGT also has specific control ports. These ports are grouped in two "records" (a type in VHDL similar to "structures" in C programming), where the inputs named "MGT_I" and the outputs named "MGT_O" are common for all MGTs from the different FPGA vendors. These records are forwarded out from the GBT Bank thus allowing custom configuration of the MGT.

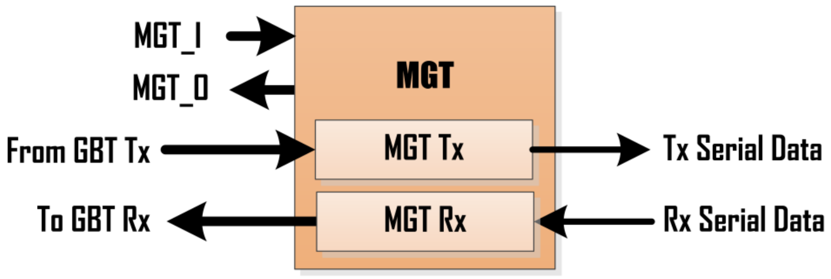


Figure 11: MGT simplified block diagram

2.3. GBT-FPGA Project Files

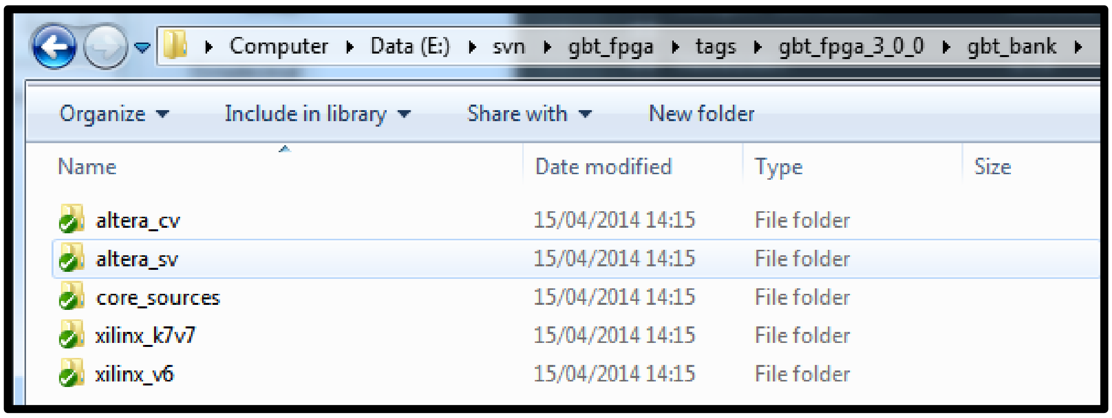
2.3.1. Common vs Device Specific Source Files

The concept behind the design of the GBT-FPGA Core is to provide a common firmware for the most common FPGAs. As a consequence, most of the source files of the GBT-FPGA Core are common to all FPGAs. These files are gathered in the same folder and classified in subfolders per functionality (gbt_rx, gbt_tx, mgt, etc.). The root folder of the common files of the GBT Bank is:

"... \gbt_bank\core_sources\..."

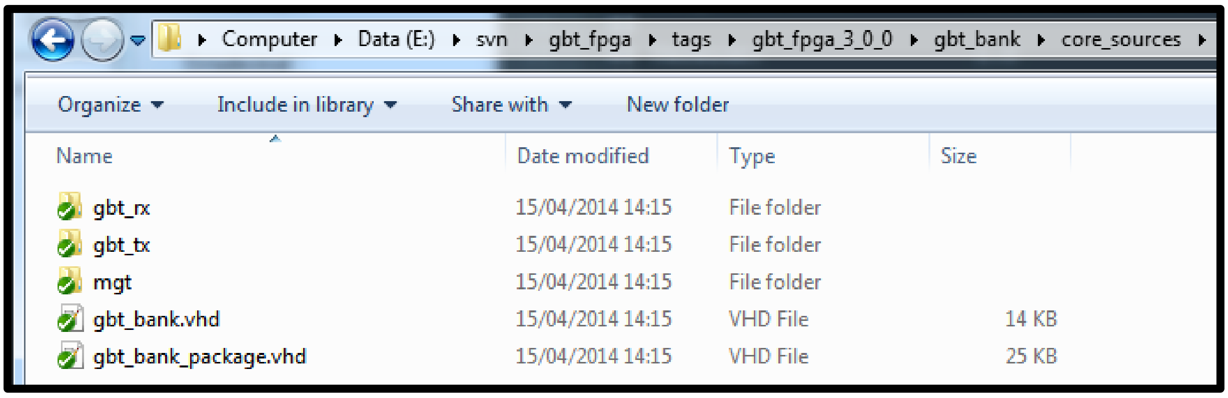
The rest of the files, that are vendor specific, are gathered per FPGA type, in separated folders. These files are also classified in subfolders following the same logic. They can be found in the folder:

"... \gbt_bank\<vendor>_<device>\..." (e.g. *"... \gbt_bank\xilinx_v6\..."*).


Computer ▸ Data (E:) ▸ svn ▸ gbt_fpga ▸ tags ▸ gbt_fpga_3_0_0 ▸ gbt_bank ▸
Organize
Include in library
Share with
New folder

Name	Date modified	Type	Size
altera_cv	15/04/2014 14:15	File folder	
altera_sv	15/04/2014 14:15	File folder	
core_sources	15/04/2014 14:15	File folder	
xilinx_k7v7	15/04/2014 14:15	File folder	
xilinx_v6	15/04/2014 14:15	File folder	

Figure 12: Common (in "core_source") and vendor specific source files folders


Computer ▸ Data (E:) ▸ svn ▸ gbt_fpga ▸ tags ▸ gbt_fpga_3_0_0 ▸ gbt_bank ▸ core_sources ▸
Organize
Include in library
Share with
New folder

Name	Date modified	Type	Size
gbt_rx	15/04/2014 14:15	File folder	
gbt_tx	15/04/2014 14:15	File folder	
mgt	15/04/2014 14:15	File folder	
gbt_bank.vhd	15/04/2014 14:15	VHD File	14 KB
gbt_bank_package.vhd	15/04/2014 14:15	VHD File	25 KB

Figure 13: Classification of files both in "core_sources" and in vendor specific folders

2.3.2. The GBT-FPGA SVN Repository

The source files, documentation and TCL scripts of the GBT-FPGA project are available in a CERN subversion (SVN) repository and supported by the members of the GBT-FPGA team.

https://svn.cern.ch/reps/ph-ese/be/gbt_fpga/tags

To access the SVN repository, the user may use any of the numerous open source SVN clients such as TortoiseSVN, shown in Figure 14.

The procedure of downloading and installing an SVN client (in this case TortoiseSVN) as well as accessing the GBT-FPGA SVN repository for downloading one official release of the GBT-FPGA Core is explained in the GBT-FPGA Video Tutorials that can be found in the web site of the GBT-FPGA project (https://espace.cern.ch/GBT-Project/GBT-FPGA).

Please note, that the official releases are stored in the "**tags**" folder.

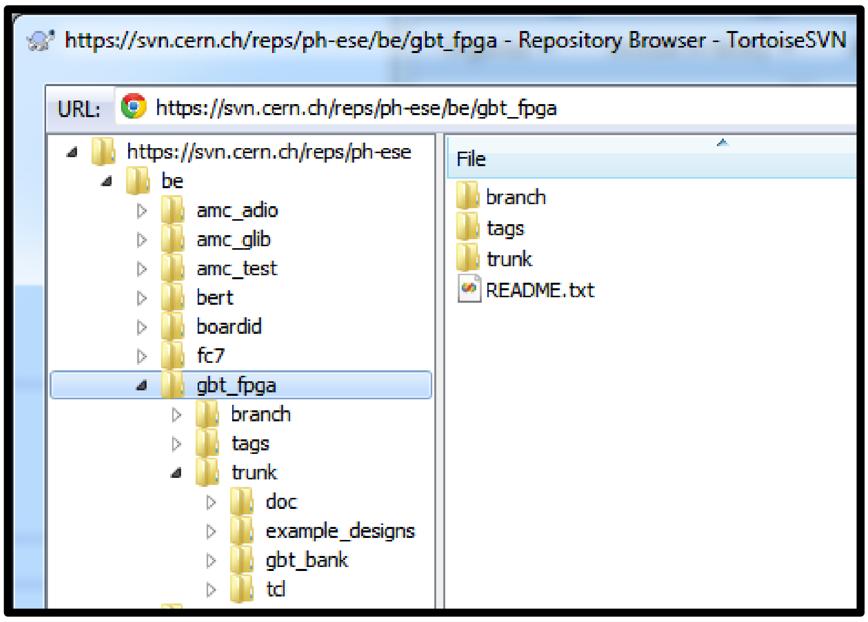

https://svn.cern.ch/reps/ph-ese/be/gbt_fpga - Repository Browser - TortoiseSVN
URL: https://svn.cern.ch/reps/ph-ese/be/gbt_fpga
https://svn.cern.ch/reps/ph-ese
be
amc_adio
amc_glib
amc_test
bert
boardid
fc7
gbt_fpga
branch
tags
trunk
doc
example_designs
gbt_bank
tcl
File
branch
tags
trunk
README.txt

Figure 14: GBT-FPGA SVN repository access through TortoiseSVN

One release (including all the FPGA types and reference designs) requires about 300 Mbytes of disk space.

2.4. Implementation

2.4.1. Getting Started

2.4.1.a. Check the Structure of the Folders

The first step after having downloaded the source files, documentation and TCL scripts of the GBT-FPGA project from the SVN repository as well as having read this user guide should be to go to the folder where the GBT-FPGA Core files were downloaded and check how the different source files are organized.

2.4.1.b. Open One of the GBT-FPGA Example Design Projects

The next step should be to open one of the GBT-FPGA example design projects. The procedure for opening a GBT-FPGA example design project is explained in section 3.

2.4.1.c. Check the Source Files

Once the GBT-FPGA example design project is open (in Altera projects it is also recommended to run the synthesis in order to generate the hierarchy of the source files), check the hierarchy and the content of the source files (which are extensively commented). The most important files from the user's point of view are explained next:

- The top level of the GBT Bank "**gbt_bank.vhd**" and its related package "**gbt_bank_package.vhd**". These two files show the external connectivity of the GBT Bank that is common for all FPGAs. These files can be found in the folder:

  "*… \gbt_bank\core_sources\*"

  Note that these two files shall not be modified by the user because they are part of the GBT-FPGA Core

- The GBT Bank package specific for the targeted FPGA "**<vendor>_<device>_gbt_bank_package.vhd**" (e.g. "xlx_v6_gbt_bank_package.vhd"). This file shows the external connectivity of the MGT. It can be found in the folder:

  "*… \gbt_bank\<vendor>_<device>*"

  Note that this file shall not be modified by the user because it is part of the GBT-FPGA Core

- The main file of the GBT-FPGA example design "**<vendor>_<device>_gbt_example_design.vhd**" (e.g. "xlx_v6_gbt_example_design.vhd") that contains the GBT Bank(s) and the rest of the modules. Through this file, it is possible to understand the interaction and connectivity between the GBT Bank and the other modules that compose the example design. This file can be found in the folder:

  "*… \example_designs\<vendor>_<device>\core_sources\*"

  (e.g." *… \example_designs\xilinx_v6 \core_sources*")

- The top level of the GBT-FPGA example design "**<hardware_platform>_gbt_example_design.vhd**" (e.g. "glib_gbt_example_design.vhd"). This file is basically a wrapper for interfacing the previously mentioned file ("<vendor>_<device>_gbt_example_design.vhd") with the FPGA-based card. This file is interesting because shows the connectivity between the example design and the hardware components of the FPGA-based card. This file can be found in the folder:

  "*… \example_designs\<vendor>_<device>\<hardware_platform>*"

  (e.g. "… \example_design\xilinx_v6\glib\")

2.4.1.d. Implement and Run one of the GBT-FPGA Example Design Projects

The last steps for getting familiar with the GBT-FPGA Core should be to implement and run the example design on the selected platform. More information about the procedures for implementing and running the GBT-FPGA example design is provided in sections **Error! Reference source not found.** and 3.2.

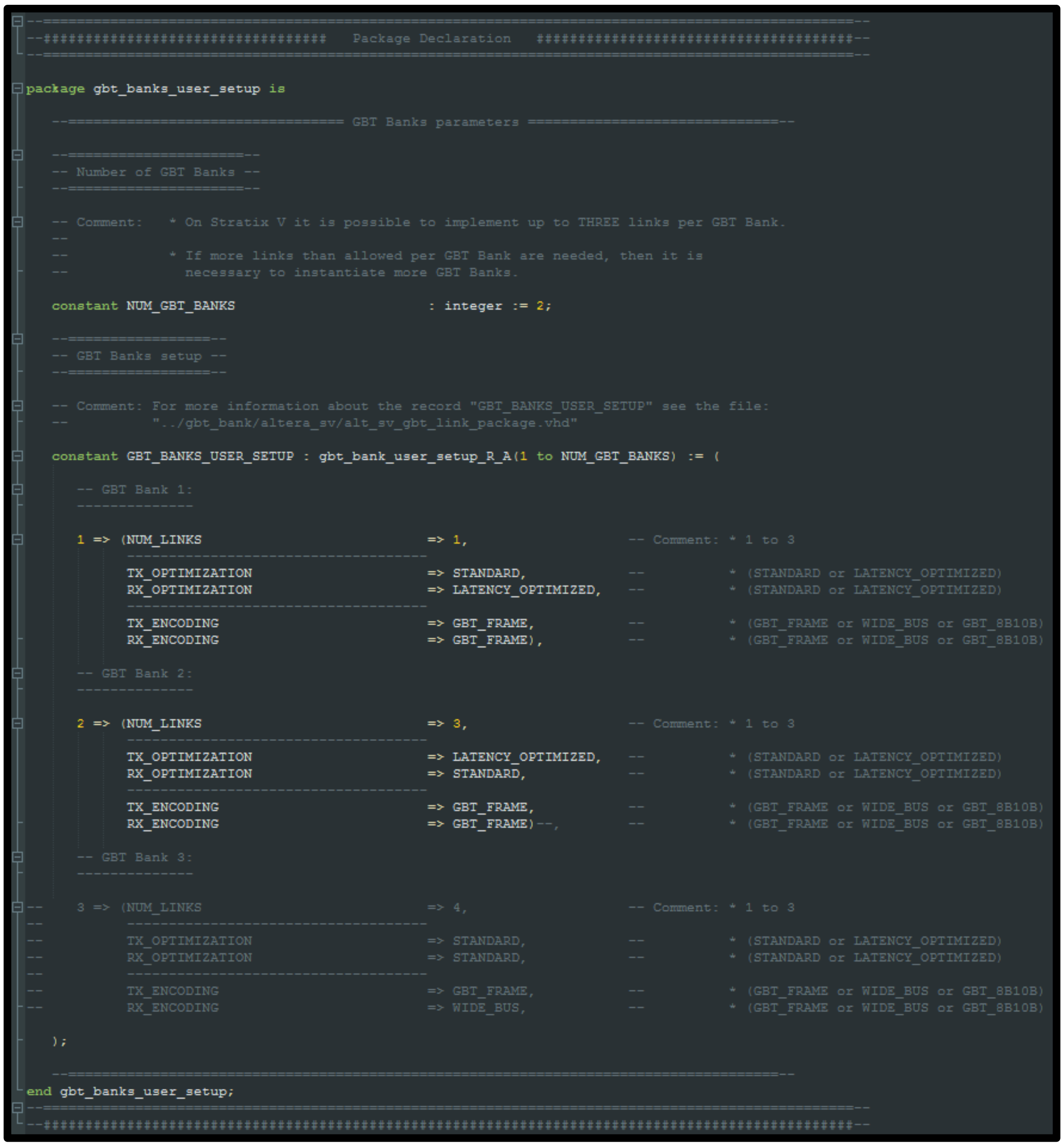
2.4.2. GBT Bank configuration

It exists 2 ways to configure the GBT bank implementation: using the user setup package located in "**<vendor>_<device>_gbt_banks_user_setup.vhd**" (e.g. "xlx_v6_gbt_banks_user_setup.vhd) or generic when implementing the gbt_bank module.

2.4.2.a. Using user setup package

All the constraints required by the GBT IP can be defined into the <vendor>_<device>_gbt_banks_user_setup.vhd dedicated file located into the "*… \gbt_bank\<vendor>_<device>*" folder. To use this method (used into the example design), the gbt bank module shall be mapped with only the GBT_BANK_ID parameter defined. Below is an example of configuration:

It is important to mention that only this file of the GBT Bank may be modified by the user



```
--=====================================================================================================--
--##################################   Package Declaration   #######################################--
--=====================================================================================================--

package gbt_banks_user_setup is

   --================================== GBT Banks parameters ==================================--

   --=====================--
   -- Number of GBT Banks --
   --=====================--

   -- Comment:   * On Stratix V it is possible to implement up to THREE links per GBT Bank.
   --
   --            * If more links than allowed per GBT Bank are needed, then it is
   --              necessary to instantiate more GBT Banks.

   constant NUM_GBT_BANKS                           : integer := 2;

   --==================--
   -- GBT Banks setup --
   --==================--

   -- Comment: For more information about the record "GBT_BANKS_USER_SETUP" see the file:
   --          "../gbt_bank/altera_sv/alt_sv_gbt_link_package.vhd"

   constant GBT_BANKS_USER_SETUP : gbt_bank_user_setup_R_A(1 to NUM_GBT_BANKS) := (

      -- GBT Bank 1:
      --------------

      1 => (NUM_LINKS                            => 1,                  -- Comment: * 1 to 3
            ------------------------------------
            TX_OPTIMIZATION                      => STANDARD,           --          * (STANDARD or LATENCY_OPTIMIZED)
            RX_OPTIMIZATION                      => LATENCY_OPTIMIZED,  --          * (STANDARD or LATENCY_OPTIMIZED)
            ------------------------------------
            TX_ENCODING                          => GBT_FRAME,          --          * (GBT_FRAME or WIDE_BUS or GBT_8B10B)
            RX_ENCODING                          => GBT_FRAME),         --          * (GBT_FRAME or WIDE_BUS or GBT_8B10B)

      -- GBT Bank 2:
      --------------

      2 => (NUM_LINKS                            => 3,                  -- Comment: * 1 to 3
            ------------------------------------
            TX_OPTIMIZATION                      => LATENCY_OPTIMIZED,  --          * (STANDARD or LATENCY_OPTIMIZED)
            RX_OPTIMIZATION                      => STANDARD,           --          * (STANDARD or LATENCY_OPTIMIZED)
            ------------------------------------
            TX_ENCODING                          => GBT_FRAME,          --          * (GBT_FRAME or WIDE_BUS or GBT_8B10B)
            RX_ENCODING                          => GBT_FRAME)--,       --          * (GBT_FRAME or WIDE_BUS or GBT_8B10B)

      -- GBT Bank 3:
      --------------

--    3 => (NUM_LINKS                            => 4,                  -- Comment: * 1 to 3
--          ------------------------------------
--          TX_OPTIMIZATION                      => STANDARD,           --          * (STANDARD or LATENCY_OPTIMIZED)
--          RX_OPTIMIZATION                      => STANDARD,           --          * (STANDARD or LATENCY_OPTIMIZED)
--          ------------------------------------
--          TX_ENCODING                          => GBT_FRAME,          --          * (GBT_FRAME or WIDE_BUS or GBT_8B10B)
--          RX_ENCODING                          => WIDE_BUS,           --          * (GBT_FRAME or WIDE_BUS or GBT_8B10B)

   );

   --==========================================================================================--
end gbt_banks_user_setup;
--=====================================================================================================--
--#####################################################################################################--
```

Figure 15: Content of the file "gbt_banks_user_setup".

2.4.2.b. Using gbt_bank generic parameters

In this case, the GBT_BANK_ID parameter shall not be defined (or be 0) when the gbt_bank module is instantiated. Only the others parameters must be defined. The figure 16 below is an example of instantiation using this method:

```
--==========--
-- GBT Bank--
--==========--

-- Comment: Note!! This example design instantiates two GBT Banks:
--
--          - GBT Bank 1: One GBT Link (Latency-Optimized GBT TX and Latency-Optimized GBT RX).

gbtBank: entity work.gbt_bank
   generic map (
      NUM_LINKS                                    => 1,                  -- Number of links (maximum depends on the device)
      TX_OPTIMIZATION                              => LATENCY_OPTIMIZED,  -- STANDARD or LATENCY_OPTIMIZED
      RX_OPTIMIZATION                              => LATENCY_OPTIMIZED,  -- STANDARD or LATENCY_OPTIMIZED
      TX_ENCODING                                  => GBT_FRAME,          -- GBT_FRAME or WIDE_BUS
      RX_ENCODING                                  => WIDE_BUS            -- GBT_FRAME or WIDE_BUS
   )
   port map (
      CLKS_I                                       => to_gbtBank_clks,
      CLKS_O                                       => from_gbtBank_clks,
      ----------------------------------------------
      GBT_TX_I                                     => to_gbtBank_gbtTx,
      GBT_TX_O                                     => from_gbtBank_gbtTx,
      ----------------------------------------------
      MGT_I                                        => to_gbtBank_mgt,
      MGT_O                                        => from_gbtBank_mgt,
      ----------------------------------------------
      GBT_RX_I                                     => to_gbtBank_gbtRx,
      GBT_RX_O                                     => from_gbtBank_gbtRx
   );
```

2.4.3. Adding the GBT-FPGA Core to Your Project

The Different files that compose the GBT Bank may be added to the user project by using TCL scripts. Note that clocking resources and the resets scheme files are not added to the user project by these scripts since they are application dependent. The naming of these scripts has the following format:

"<vendor_<device>_gbt_bank.tcl" (e.g. xlx_v6_gbt_bank.tcl)

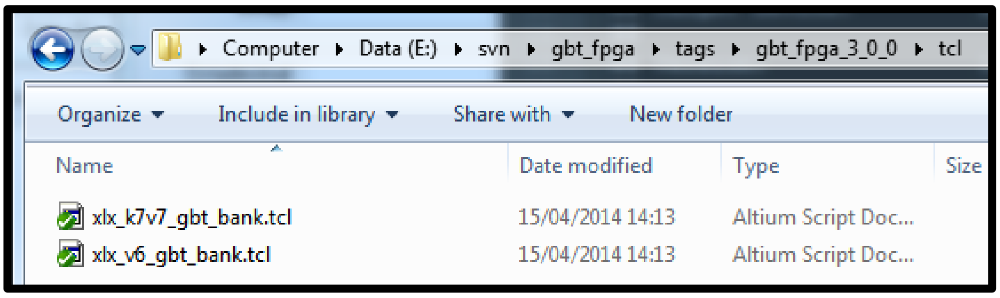
They can be found in the folder:

"…\tcl\"

2.4.3.a. Running TCL Scripts: Common Steps for Different FPGA vendors

This tutorial describes the steps to add the source files of the GBT-FPGA Core targeted to Xilinx Virtex 6, but these steps are equivalent for any other supported FPGA.

1. Go to the folder where the TCL scripts are located (see Figure 16).



Computer ▸ Data (E:) ▸ svn ▸ gbt_fpga ▸ tags ▸ gbt_fpga_3_0_0 ▸ tcl

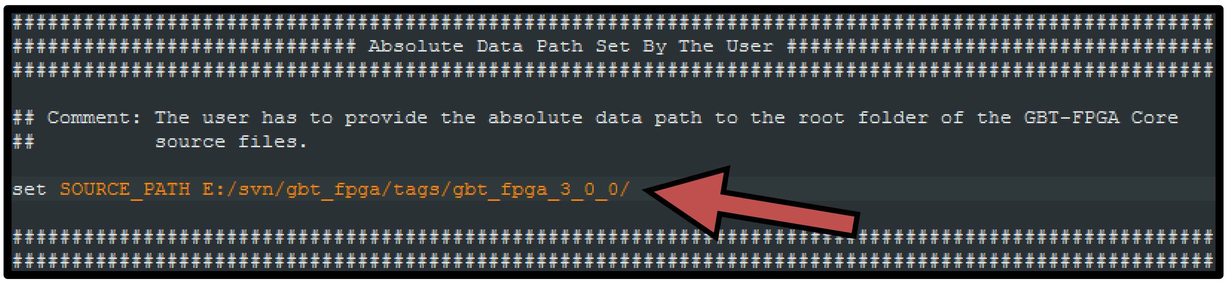
Organize ▾ Include in library ▾ Share with ▾ New folder

Name	Date modified	Type	Size
xlx_k7v7_gbt_bank.tcl	15/04/2014 14:13	Altium Script Doc...	
xlx_v6_gbt_bank.tcl	15/04/2014 14:13	Altium Script Doc...	

Figure 16: Content of the TCL scripts folder.

2. Open the script targeted to the selected FPGA (in this case "xlx_v6_gbt_bank.tcl").

3. Set the variable "SOURCE_PATH" with the absolute path of the root folder of the GBT-FPGA project on your computer (see Figure 17).



```
#####################################################################################################
############################# Absolute Data Path Set By The User ####################################
#####################################################################################################

## Comment: The user has to provide the absolute data path to the root folder of the GBT-FPGA Core
##          source files.

set SOURCE_PATH E:/svn/gbt_fpga/tags/gbt_fpga_3_0_0/

#####################################################################################################
#####################################################################################################
```

Figure 17: Modifying the absolute data path of the TCL script file.

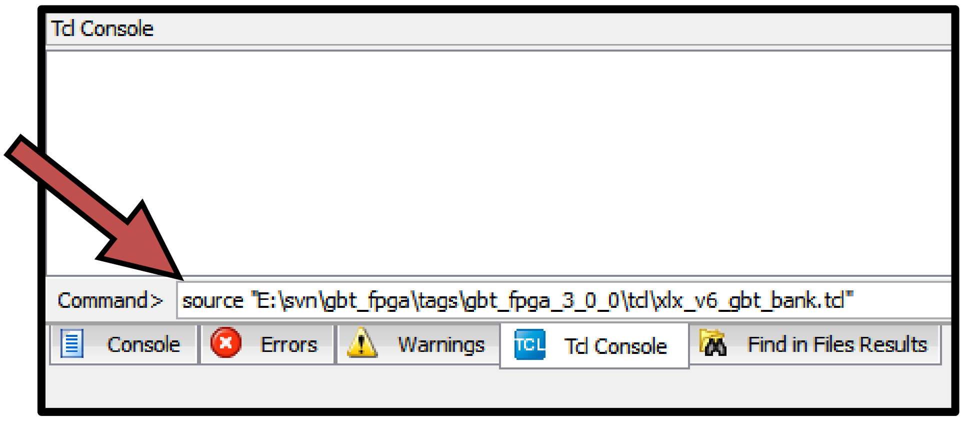
Save the modified script file and close it.

2.4.3.b. Running TCL Scripts

4. Open or create the project that will include the GBT-FPGA Core.

The GBT-IP sources can be sourced using the device dedicated TCL. The command below should be executed to run the script into the TCL console:

source "<absolute path to the folder of the TCL script>/<vendor>_<device>_gbt_bank.tcl"



```
source "E:\svn\gbt_fpga\tags\gbt_fpga_3_0_0\tcl\xlx_v6_gbt_bank.tcl"
```

Figure 18: Writing the TCL command for running the TCL script in the Tcl Console of ISE.

Note: If the TCL console is disabled, you can enable it by selecting the option into the development tool (E.g.: "Views > Utility Windows > TCL Console" for Quartus and "View > Panel > TCL Console" for ISE and Vivado).

2.4.4. The Particularities of the Latency-Optimized Version

As previously mentioned in section 2.1.2, Trigger related electronic systems in High Energy Physics (HEP) experiments, such as TTC links, require a fixed, low and deterministic latency in the transmission of the clock and data to ensure correct event building. On the other hand, other electronic systems that are not time critical, such as DAQ, do not need to comply with this requirement. The unification of TTC, SC and DAQ functionality on the new Rad-Hard Optical Link for Experiments simplifies the topology and reduces the number of optical links. However, this topology introduces new technical challenges related to the reference clock that will have to be recovered from the incoming data stream.

The GBT-FPGA project provides two types of implementation for the transmitter and the receiver: the "Standard" version, targeted for non-time critical applications and the "Latency-Optimized" version, ensuring a fixed, low and deterministic latency of the clock and data (at the cost of a more complex implementation). The main differences between the Standard and the Latency-Optimized versions are presented in Table 1: Standard VS Latency-Optimized.

This section focuses on the Potential Uncertainty Points (PUPs) of the GBT Bank when implementing the Latency-Optimized version and explains the methods applied by the GBT-FPGA team in order to guarantee the latency determinism in both clock and data. Then the importance of the system calibration, that ensures the correct sampling of the data, is emphasized.

2.4.4.a. Potential Uncertainty Points in the Latency-Optimized version

Achieving fixed and deterministic phase in clocks as well as low, fixed and deterministic latency in data transmission/reception when implementing the Latency-Optimized version of the GBT-FPGA core, requires identifying and properly managing the different PUPs of the GBT-FPGA core-based system.

Since these uncertainty points are implementation and configuration dependent, dealing with systems featuring the Latency-Optimized version of the GBT-FPGA core may become very challenging, especially in multi-GBT Link configurations.

The different PUPs within the GBT Bank have been already properly managed by the firmware, being only necessary to constraint some critical paths of the GBT Link(s) during in-system implementation (these constraints are shown in the GBT-FPGA example design). On the other hand, the external PUPs have to be managed by the user (although the GBT-FPGA project provides example designs including custom modules for this purpose). It is important to mention that the methods applied for achieving latency determinism in the Latency-Optimized version are based on general concepts of logic design, so most of them are common to the different FPGAs supported by the GBT-FPGA project.

The block diagram shown in Figure 19: Internal and external PUPs of the GBT Bank. Depicts a GBT Bank featuring one GBT Link, highlighting the internal and external PUPs, which are explained in the following subsections.

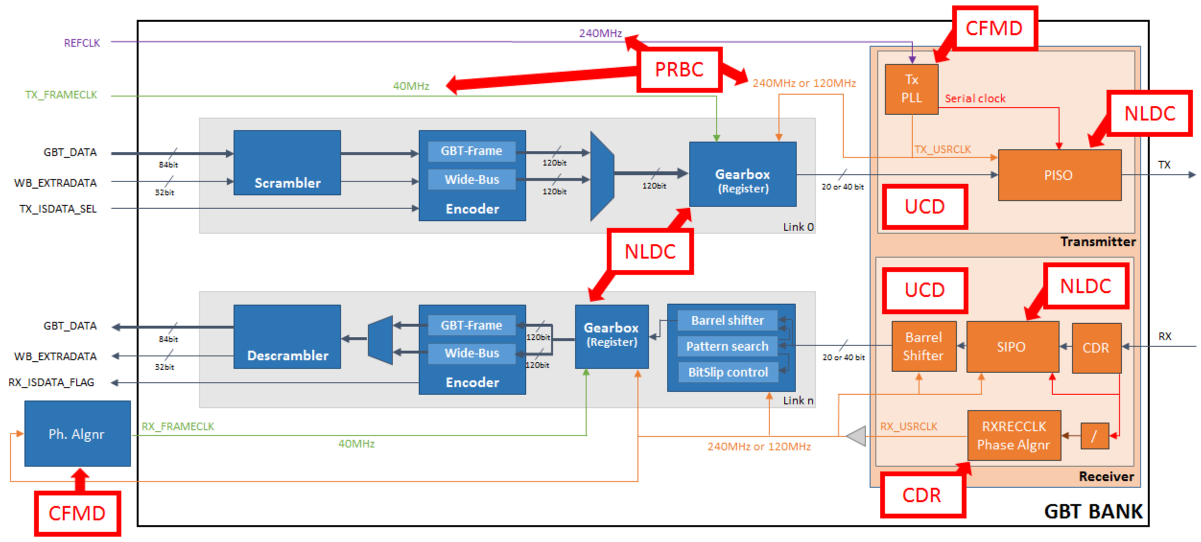


Figure 19: Internal and external PUPs of the GBT Bank.

2.4.4.b. Unnecessary Clock Domains (UCD)

One of the most delicate points in logic design and a source of latency uncertainty issues is the clock domain crossing (CDC). In order to minimize the impact of CDCs in the GBT-FPGA Core, the number of clock domains has been reduced as much as possible.

The most critical components of the GBT-FPGA Core in terms of CDCs are the MGTs due to their complexity, containing a high number of clock domains, and to the fact that they are hard IP blocks provided by the FPGA vendor (so the user can only set the different parameters of the core). The number of internal clock domains within the MGT has been minimized by properly selecting the frequency of the different clocks and the width of the data buses. It is important to mention that the internal architecture of the MGT varies between the different models so each case has to be studied independently.

As an example of MGT clock domains complexity, the internal structure of a GTX TX in a Xilinx Virtex 6 is shown in Figure 20. This single GTX structure can have up to 4 separate clock domains.

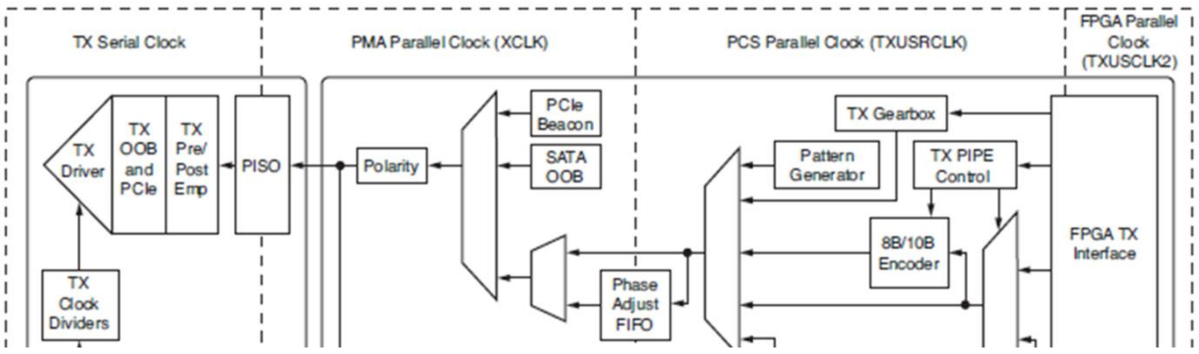


Figure 20: Internal structure of GTX TX in Xilinx Virtex 6.

2.4.4.c. Non Latency Deterministic Components (NLDC)

Although the number of clock domains in the GBT-FPGA Core has been minimized, not all can be merged so the boundary crossing of the required clock domains has to be achieved in such a way that ensures low and deterministic latency of data.

The common approach for CDC of multi-bit signals in high throughput data paths is the utilisation of memories (such as asynchronous FIFO or DPRAM), due to their high performance, reliability and simplicity of use (this approach is used in the Standard version). However, the utilisation of these blocks has a penalty in terms of latency, which may be not deterministic and also requires undesired extra clock cycles. Due to this penalty, in the Latency-Optimized version, the memory-based CDC used in the Standard version is replaced by a register-based approach. This ensures low and deterministic latency in data at the **cost of calibration after system implementation**, for guaranteeing the integrity of the data. The details of the constraints on the phase between clocks are discussed below and the calibration is detailed in section 2.4.4.g.

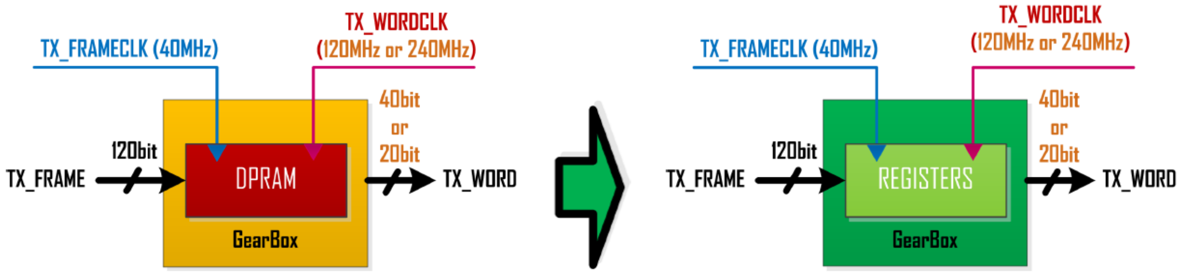


Figure 21: Gearbox DPRAM-based (left) VS Gearbox Registers-based (right).

One example of Non Latency Deterministic Components in the GBT-FPGA Core is the GBT TX GearBox (see Figure 21), where the scrambled and encoded data (a 120-bit frame), generated at 40 MHz in the TX_FRAMECLK domain ("Bunch Clock" in LHC Experiments), is divided into several words when crossing to the TX_WORDCLK domain (the MGT TX clock). The number and width of the words as well as the frequency of TX_WORDCLK are device dependent and it may be either three words of 40 bits at 120 MHz or six words of 20 bits at 240 MHz.

2.4.4.d. Phase Relationship Between Clocks (PRBC)

As previously mentioned, the phase relationship between the different clocks is another constraint when implementing the Latency-Optimized version of the GBT-FPGA Core. It must be fixed in order to achieve latency determinism in both clock and data as well as set to a correct value for proper data sampling.

The fixed phase relationship requires the utilisation of a **single clock source (generally the Bunch Clock)**, that is further multiplied, divided and recovered from the data stream for generating the different clock domains. In some cases, the devices in charge of deriving this single clock source (clock synthesizers, etc.) have been carefully selected and characterized. These devices offer the required features for the fixed and deterministic phase application (such as clock feedbacks (zero-delay), synchronous dividers, etc.) and are controlled by the firmware when necessary. In other cases, when the component does not comply with the requirements of the Latency-Optimized version (e.g. asynchronous "Byte Serializer" clock divider in Altera Cyclone V GT and Stratix V GX transceivers , most common PLLs, etc.), the firmware monitors and controls the phase of the derived clocks for ensuring the phase determinism.

The correct phase value of the different clocks is selected during the calibration procedure, explained in section 2.4.4.g.

2.4.4.e. Clock Frequency Multiplication/Division (CFMD)

Multiplying the frequency of a source clock and then dividing it again to the original frequency (e.g. creating the RX_FRAMECLK clock from the RX_WORDCLK) will lead to a non-deterministic phase difference between the divided and source clocks. This issue is due to the fact that the rising edge of the divided clock may lock onto any of the rising edges of the multiplied clock (see Figure 22: Example of Clock Frequency Multiplication/Division (CFMD).).

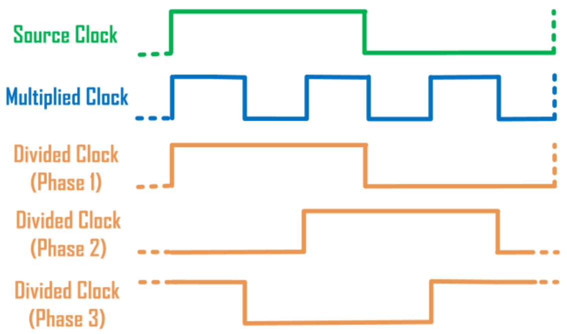


Figure 22: Example of Clock Frequency Multiplication/Division (CFMD).

In order to avoid this uncertainty in the phase of the divided clock, it is necessary to generate a reference signal that indicates when the rising edge of the source clock occurs and then, shift the phase of the divided clock to lock on the reference signal. In the GBT-FPGA Core, each time that the header of the data frame (which is aligned with the rising edge of TX_FRAMECLK) is detected by the GBT RX control logic, a flag signal ("header_flag") is asserted, so it may be used as a reference for the phase alignment of the RX_FRAMECLK (which is a recovered TX_FRAMECLK).

2.4.4.f. Clock & Data Recovery (CDR)

A particular case of CFMD is the Clock and Data Recovery (CDR), where both clock and data have to be properly managed in order to avoid uncertainty issues.

In the GBT-FPGA Core case, clock and data from the incoming serial stream at 4.8 Gbps are separated during CDR procedure in the MGT RX of the GBT Link. Then, the frequency of the serial clock is divided by a certain factor N to generate RX_WORDCLK (N equals 10 when RX_WORDCLK at 240MHz or N equals 20 when RX_WORDCLK at 120MHz). Due to the utilisation of Dual Data Rate (DDR) by the CDR, the recovered clock may lock onto either of the edges (rising and falling) of the serial clock. This gives 2xN possible phases for the recovered clock and also, the same number of possible bitslips in the recovered data. This clock and data uncertainty issue is worked out by using a custom monitor logic embedded into the GBT Link which shifts both clock and data to ensure a constant line latency.

2.4.4.g. Calibration

In any logic design for FPGA, the phase of the different internal clocks as well as the length of the combinatorial data paths between flip-flops may vary after re-implementation. In addition, the phase of the different clocks may also vary with physical changes in the system (e.g. fibres replacement, etc.). For this reason, when dealing with systems featuring register-based CDC, this uncertainty in clock phase and data propagation delay may lead to errors and latency uncertainty in the data.

In the worst case, the launch clock (e.g. TX_FRAMECLK) and the latch clock (e.g. TX_WORDCLK) are wrongly aligned and data is sampled in metastability zone, thus presenting errors. Furthermore, there is also the possibility of sampling data correctly after implementation, but this sampling is done very close to the metastability zone, so a slight phase shift of the launch and/or latch clocks (due to temperature variations, etc.) will result in errors or even in a phase jump of one latch clock period for the data.

For ensuring the reliability of the system, a calibration process is required after any re-implementation or physical changes in the system that varies clock and/or data delays. The calibration consists of adjusting the phase of the launch and/or the latch clocks to have the latch clock in the middle of the sampling window to guarantee a comfortable margin.

This calibration can be made using the PH_ALIGNED flag from the TX Gearbox. This flag goes high when no error are detected for a transmission of 50 gbt words. The calibration consists in getting the minimum and the maximum deviation between both TX_FRAMECLK and MGT_REFCLK for a good transmission in order to know the shifting window and to **shift the MGT_REFCLK** to be in the middle of it.

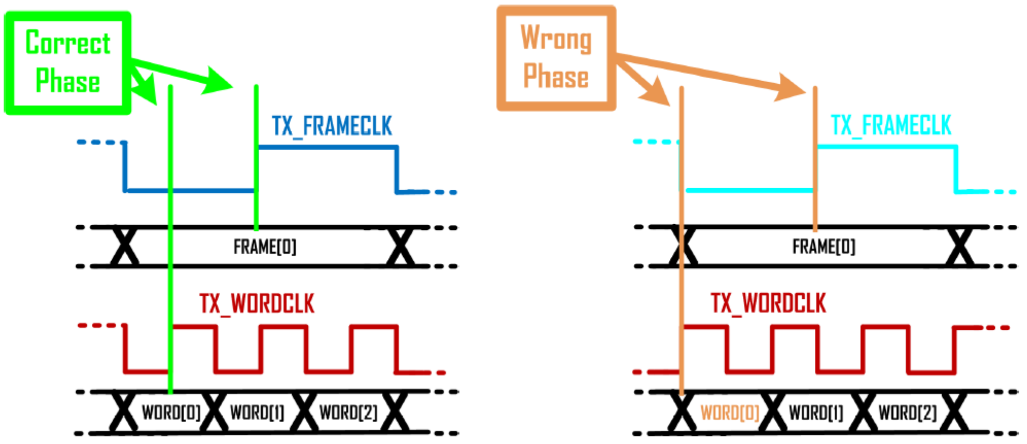


Figure 23: Correct Sampling Point (left) VS wrong Sampling Point (right).

Please note that after calibration, further compensations (e.g. due to variation in fiber length) are not required since the Back-End crates as well as the fibres between the counting room and the detector are not subject to large temperature variations.

2.4.5. Clocks Scheme

The frequencies of the internal clock domains and the widths of the different data paths of the GBT Core have been optimized in order to minimize the utilization of clocking resources of the targeted FPGA. Moreover, the common structure of the GBT Core for different FPGAs is also present in the clocking scheme, thus facilitating the in-system implementation and the user support.

When implementing the standard version of the GBT Core, the different GBT Links from the same GBT Bank may share the clocking resources required thus minimizing even more the utilization of clocking resources of the targeted FPGA.

It is important to mention that the clocks scheme of the latency-optimized version is far more complicated than the clocks scheme of the standard version, requiring dedicated clock phase monitors and aligners for the correct functioning of the GBT Links (examples of these clocking resources are provided with the example design). In addition to the extra complexity, the clocking resources utilization of the latency-optimized version in multi-link implementations is higher compared with the standard-version counterpart, since the dedicated clock phase monitors and aligners cannot be shared by the different GBT Links from the same GBT Bank.

2.4.5.a. Clocks Scheme for Single-Link Instantiation

- Clocks scheme for single-link implementation of standard version

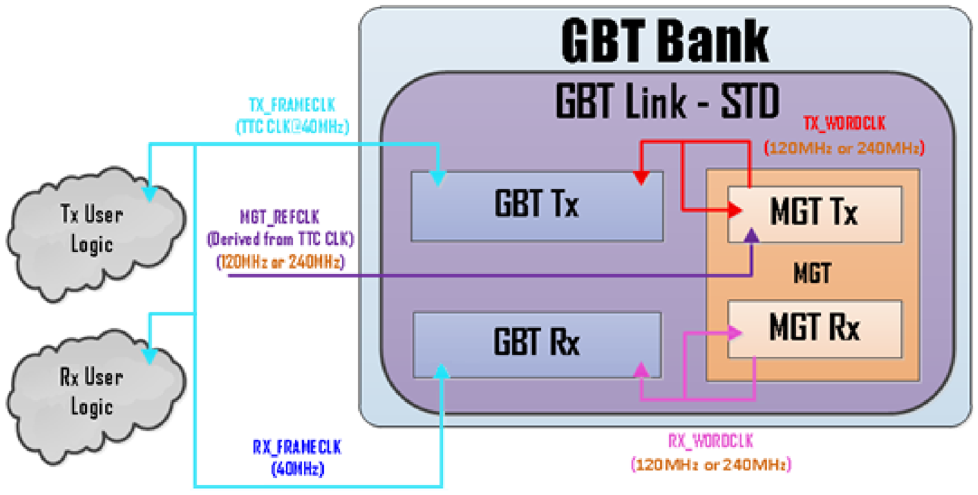


Figure 24: Example of clocks scheme for single-link implementation of standard version.

- Clocks scheme for single-link implementation of latency-optimized version

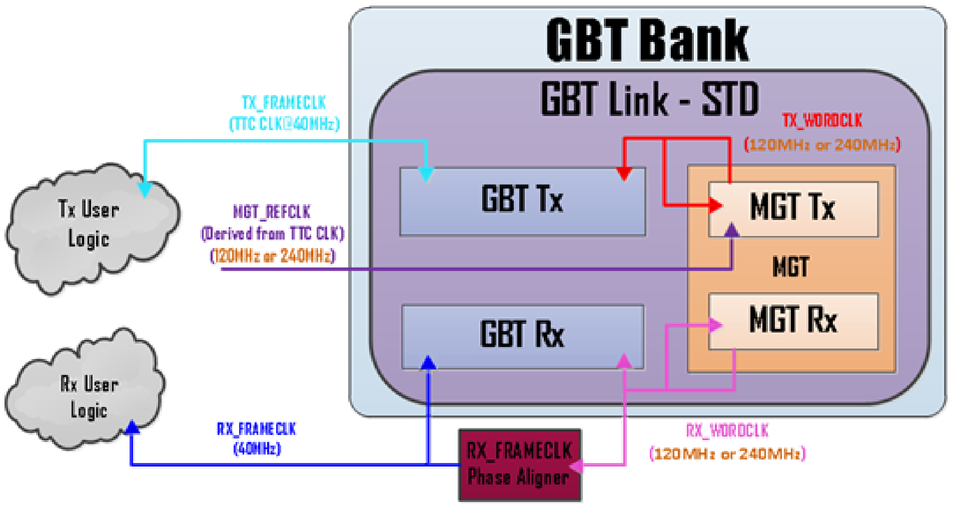

GBT Bank
GBT Link - STD
TX_FRAMECLK (TTC CLK@40MHz)
TX_WORDCLK (120MHz or 240MHz)
Tx User Logic
MGT_REFCLK (Derived from TTC CLK) (120MHz or 240MHz)
GBT Tx
MGT Tx
MGT
MGT Rx
GBT Rx
Rx User Logic
RX_FRAMECLK (40MHz)
RX_FRAMECLK Phase Aligner
RX_WORDCLK (120MHz or 240MHz)

Figure 25: Example of clocks scheme for single-link implementation of latency-optimized version.

2.4.5.b. Clocks Scheme for Multi-Link Instantiation

- Clocks scheme for multi-link implementation of latency-optimized version:

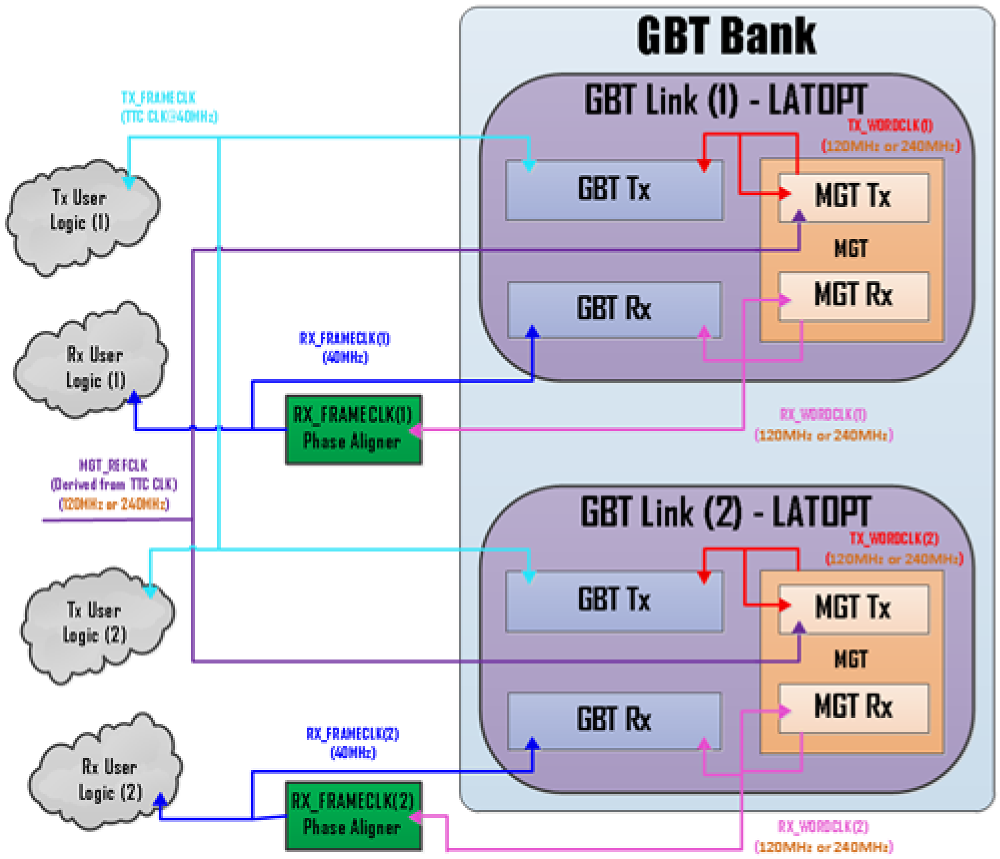

GBT Bank
GBT Link (1) - LATOPT
TX_FRAMECLK (TTC CLK@40MHz)
TX_WORDCLK(1) (120MHz or 240MHz)
Tx User Logic (1)
GBT Tx
MGT Tx
MGT
MGT Rx
GBT Rx
Rx User Logic (1)
RX_FRAMECLK(1) (40MHz)
RX_FRAMECLK(1) Phase Aligner
RX_WORDCLK(1) (120MHz or 240MHz)
MGT_REFCLK (Derived from TTC CLK) (120MHz or 240MHz)
GBT Link (2) - LATOPT
TX_WORDCLK(2) (120MHz or 240MHz)
Tx User Logic (2)
GBT Tx
MGT Tx
MGT
MGT Rx
GBT Rx
Rx User Logic (2)
RX_FRAMECLK(2) (40MHz)
RX_FRAMECLK(2) Phase Aligner
RX_WORDCLK(2) (120MHz or 240MHz)

Figure 26: Example of clocks scheme for multi-link implementation of latency-optimized version.

2.4.6. Timing Closure

When implementing a GBT-FPGA Core-based system, as when implementing any other system on an FPGA, it is recommended to use timing and floorplanning constraints in order to guide the EDA tool during the implementation process, thus facilitating the timing closure and obtaining more precise results in the static timing analysis report.

The syntax and particularities of the different constraints may vary from one FPGA vendor (or EDA tool) to another, but since the architecture of the GBT-FPGA Core is common for all FPGAs, most of the different paths and blocks to be constrained are also common for all FPGAs.

It is important to mention that since in the latency-optimized version, the different clock domains crossings are based on registers and some critical components require a fixed location in order maintain the same timing after reimplementation (see 6), it is necessary to provide extra constraints that are not required in the standard version.

Table 6: Extra constrains in the latency-optimized version.

Type of Constraint	Critical Path or Logic Element	Comment
Timing	Scrambler to Gearbox Data Path (GBT TX)	Recommended propagation delay < 20 ns
	Gearbox to Scrambler Data Path (GBT RX)	Recommended propagation delay < 20 ns
Floorplanning	Sampling Registers in Clock Phase Monitors	Lock placement close to the source

2.4.6.a. Timing Closure in Xilinx FPGAs

The different constraints applied to the GBT-FPGA example designs for Xilinx devices may be used as a reference by the user. Both the timing and floorplanning constraints can be found in the "User Constraints Files" (UCF) or (XDC) files in the folder:

"…\example_designs\xilinx_<device>\<*hardware_platform*>\ucf\" or "…\<*hardware_platform*>\xdc\"

(e.g. …\example_designs\xilinx_k7v7\kc705\ucf\ or …\kc705\xdc\)

2.4.6.b. Timing Closure in Altera FPGAs

The different constraints applied to the GBT-FPGA example designs for Altera devices may be used as a reference by the user.

The timing constraints can be found in the "Synopsys Design Constraint" (SDC) file in the folder:

"…\example_designs\altera_<device>\<*hardware_platform*>\sdc\"

(e.g. …\example_designs\alteara_sv\amc40\sdc\)

Whereas the floorplaning constraints can be found in the Quartus II Setings File (QSF) in the folder:

"…\example_designs\altera_<device>\<*hardware_platform*>\qii_project\"

(e.g. …\example_designs\alteara_sv\amc40\qii_project\)

2.4.7. Resets Scheme

The reset sequence of the GBT-FPGA Core is divided into four steps, in order to properly initialize the different components of the link (see Figure 27).

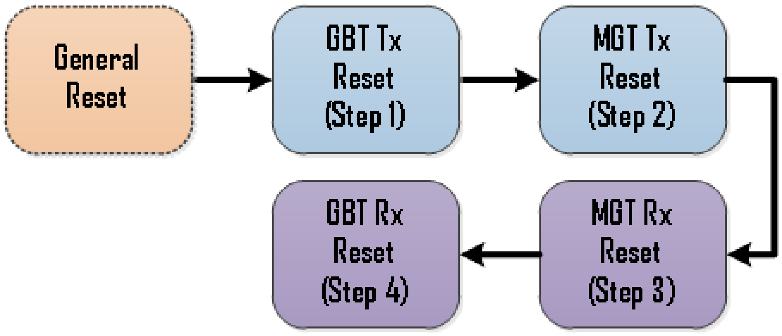


Figure 27: The reset sequence of the GBT-FPGA Core

In the transmitter part, the Tx user logic and the GBT Tx are reset first (step 1), in order to provide stable data to the MGT Tx before being reset in step 2.

Note!! When implementing the latency-optimized version, the resets of the transmitter part of the GBT-FPGA Core must be synchronous with TX_FRAMECLK in order to properly align the internal clock domains of GBT Tx.

In the receiver part of the GBT-FPGA Core, the MGT Rx must be reset (step 3) when the initialization of the transmitter of the incoming data stream is concluded and the transmitted data is stable, so the Clock and Data Recovery (CDR) block of the MGT Rx can properly separate the recovered clock from the incoming data stream. Once the MGT Rx is properly delivering the received data and the recovered clock (RX_WORDCLK), the reset of GBT Rx is deasserted (step 4) so the received data is decoded and delivered to the Rx user logic. Please note that the reset of GBT Rx may be also used for resetting Rx user logic

The GBT-FPGA project provides two reset blocks that combined execute the reset sequence of the GBT-FPGA Core. The first block is a General Reset that resets the second block, named GBT Bank Reset, which sequentially performs the four steps of the reset sequence.

Please note that these two reset blocks are used in the GBT-FPGA example designs provided by the GBT-FPGA Project, but since they are not part of the GBT-FPGA Core, they are not added to the HDL project by the TCL scripts.

- The General Reset block file "<vendor>_<device>_reset.vhd" (e.g. xilinx_v6_reset.vhd) can be found in the folder:
  - Xilinx:

  "…\example_designs\xilinx_<device>\core_sources\" (e.g. "…\example_designs\xilinx_v6\core_sources\")

  - Altera:

"…\example_designs\altera_<device>\core_sources\reset\" (e.g. "…\example_designs\altera_sv\core_sources\")

- The GBT Bank Reset block file "gbt_bank_reset.vhd" can be found in the folder:

"…\example_designs\core_sources\"

2.5. Operating the GBT-FPGA Core

The operation of the GBT-FPGA Core is done through the different ports of the GBT Bank that control and interface with the user logic the instantiated GBT Links. While most of these ports are common for all FPGAs, some other ports are device specific.

- The common ports of the GBT Bank are declared in the file "gbt_bank_package.vhd", that can be found in the folder:

  "...\gbt_bank\core_sources\"

- The device specific ports of the GBT Bank are declared in the file "<vendor>_<device>_gbt_bank_package.vhd" (e.g. altera_cv_gbt_bank_package.vhd), that can be found in the folder:

  "...\gbt_bank\<vendor>_<device>\" (e.g. "...\gbt_bank\altera_cv\")

The different ports of the GBT Bank are organized into four categories (Clocks, GBT Tx. GBT Rx and MGT) and grouped in records, In order to facilitate the in-system implementation.

2.5.1. Clock Ports

The different clocks of the GBT Bank are forwarded in and out through the ports of the records CLKS_I and CLKS_O respectively.

Please note that the user must provide the external clocking resources (e.g. PLL, etc.) when required.

(n) One port per GTB Link of the GBT Bank (*) In standard version, GBT Links can be clocked by the TX_FRAMECLK

2.5.1.a. Common Clock Ports

Table 7: Common clock ports

Port	Dir	Clock Domain	Description
CLKS_I.tx_frameClk(n)	In	TX_FRAMECLK(n)	40MHz clock provided by the user (TTC clock)
CLKS_I.rx_frameClk(n)	In	RX_FRAMECLK(n)	40MHz clock derived from "rx_wordClk(n)"(*)

2.5.2. Device Specific Clock Ports

- Xilinx Virtex 6 specific clock ports:

Table 8: Xilinx Virtex 6 specific clock ports

Port	Dir	Clock Domain	Description
CLKS_I.mgt_clks.tx_refClk	In	MGT_TX_REFCLK	240MHz reference clock for MGT Tx
CLKS_I.mgt_clks.rx_refClk	In	MGT_RX_REFCLK	240MHz reference clock for MGT Rx
CLKS_I.mgt_clks.drp_dClk	In	DRP CLK	MGT DRP reference clock (see DS152)
CLKS_O.mgt_clks.tx_wordClk (n)	Out	TX_WORDCLK(n)	240MHz clock from MGT Tx (n)
CLKS_O.mgt_clks.rx_wordClk(n)	Out	RX_WORDCLK(n)	240MHz clock from MGT Rx (n)

- Xilinx Kintex 7 & Virtex 7 specific clock ports:

Table 9: Xilinx Kintx 7 & Virtex 7 specific clock ports

Port	Dir	Clock Domain	Description
CLKS_I.mgt_clks.mgtRefClk	In	MGT_REFCLK	120MHz reference clock for MGT
CLKS_I.mgt_clks.mgtRstCtrlRefClk	In	TX_FRAMECLK(I)	40MHz reference clock for MGT initialization FSMs
CLKS_I.mgt_clks.drpClk	In	DRP CLK	MGT DRP reference clock (see DS182 or DS183)
CLKS_I.mgt_clks. cpllLockDetClk	In	CPLLLockDetClk	CPLL locked detection clock (120MHz)
CLKS_O.mgt_clks.tx_wordClk (n)	Out	TX_WORDCLK(n)	120MHz clock from MGT Tx (n)
CLKS_O.mgt_clks.rx_wordClk(n)	Out	RX_WORDCLK(n)	120MHz clock from MGT Rx (n)

- Xilinx Kintex Ultrascale specific clock ports:

Table 10: Xilinx Kintx Ultrascale specific clock ports

Port	Dir	Clock Domain	Description
CLKS_I.mgt_clks.mgtRefClk	In	MGT_REFCLK	120MHz reference clock for MGT
CLKS_I.mgt_clks.mgtRstCtrlRefClk	In	TX_FRAMECLK(I)	40MHz reference clock for MGT initialization FSMs
CLKS_I.mgt_clks.drpClk	In	DRP CLK	MGT DRP reference clock (see DS182 or DS183)
CLKS_O.mgt_clks.tx_wordClk (n)	Out	TX_WORDCLK(n)	120MHz clock from MGT Tx (n)
CLKS_O.mgt_clks.rx_wordClk(n)	Out	RX_WORDCLK(n)	120MHz clock from MGT Rx (n)

- Altera Cyclone V GT specific clock ports:

Table 11: Altera Cyclone GT specific clock ports

Port	Dir	Clock Domain	Description
CLKS_I.mgt_clks. mgtRefClk	In	MGT_REFCLK	120MHz reference clock for MGT
CLKS_I.mgt_clks. txFrameClk	In	TX_FRAMECLK(I)	40MHz clock provided by the user (TTC clock)
CLKS_O.mgt_clks.tx_wordClk(n)	Out	TX_WORDCLK(n)	120MHz clock from MGT Tx (n)
CLKS_O.mgt_clks.rx_wordClk(n)	Out	RX_WORDCLK(n)	120MHz clock from MGT Rx (n)

- Altera Stratix V GX specific clock ports:

Table 12: Altera Stratix GX specific clock ports

Port	Dir	Clock Domain	Description
CLKS_I.mgt_clks. mgtRefClk	In	MGT_REFCLK	120MHz reference clock for MGT Tx
CLKS_I.mgt_clks. txFrameClk	In	TX_FRAMECLK(1)	40MHz reference clock for MGT initialization FSMs
CLKS_O.mgt_clks.tx_wordClk(n)	Out	TX_WORDCLK(n)	120MHz clock from MGT Tx (n)
CLKS_O.mgt_clks.rx_wordClk(n)	Out	RX_WORDCLK(n)	120MHz clock from MGT Rx (n)

- Altera Arria 10 GX specific clock ports:

Table 13: Altera Arria 10 specific clock ports

Port	Dir	Clock Domain	Description
CLKS_I.mgt_clks. mgtRefClk	In	MGT_REFCLK	240MHz reference clock for MGT Tx
CLKS_I.mgt_clks. txFrameClk	In	TX_FRAMECLK(1)	40MHz reference clock for MGT initialization FSMs
CLKS_O.mgt_clks.tx_wordClk(n)	Out	TX_WORDCLK(n)	240MHz clock from MGT Tx (n)
CLKS_O.mgt_clks.rx_wordClk(n)	Out	RX_WORDCLK(n)	240MHz clock from MGT Rx (n)

2.5.3. GBT Tx Ports

The different ports of GBT Tx are common for all FPGAs and are grouped as inputs and outputs in the records GBT_TX_I and GBT_TX_O respectively.

Table 14: GBT Tx ports

Port	Dir	Clock Domain	Description
GBT_TX_I(n).reset	In	Asynchronous	Assert to '1' to reset the GBT Tx (n) logic
GBT_TX_I(n).isDataSel	In	TX_FRAMECLK(n)	Selects header sent by GBT Tx (n) ('1' -> DATA | '0' -> IDLE')
GBT_TX_I(n).data	In	TX_FRAMECLK(n)	84-bit user data bus of GBT Link Tx (n)
GBT_TX_I(n).extraData_wideBus	In	TX_FRAMECLK(n)	32-bit Wide-Bus extra user data of GBT Link Tx (n)
GBT_TX_O(n).latOptGbtBank_tx	Out	Static	Indicates version of GBT Bank Tx ('0' -> STD | '1' -> LATOP)
GBT_TX_O(n). txGearboxAligned_o	Out	TX_FRAMECLK(n)	Indicates status for tx clock domain crossing (LATOPT)
GBT_TX_O(n). txGearboxAligned_done	Out	TX_FRAMECLK(n)	'1' when the previous flag is computed (LATOPT)

2.5.4. GBT Rx Ports

The different ports of GBT Rx are common for all FPGAs and are grouped as inputs and outputs in the records GBT_RX_I and GBT_RX_O respectively.

Table 15: GBT Rx ports

Port	Dir	Clock Domain	Description
GBT_RX_I(n). reset	In	Asynchronous	Assert to '1' for resetting the GBT Rx (n) logic
GBT_RX_I(n). rxFrameClkReady	In	Asynchronous	Assert to '1' when RX_FRAMECLK is ready
GBT_RX_O(n). latOptGbtBank_rx	Out	Static	Indicates version of GBT Bank Rx ('0' -> STD | '1' -> LATOP)
GBT_RX_O(n). ready	Out	RX_FRAMECLK(n)	When '1' indicates that GBT Rx (n) is ready
GBT_RX_O(n). bitSlipNbr	Out	RX_WORDCLK(n)	Indicates nbr of bitslips done by GBT Rx (n) for aligning data
GBT_RX_O(n). header_flag	Out	RX_WORDCLK(n)	When '1' indicates that a header is detected by GBT Rx (n)

GBT_RX_O(n). header_lockedFlag	Out	RX_WORDCLK(n)	When '1' indicates that the header is locked by GBT Rx (n)
GBT_RX_O(n). isDataFlag	Out	RX_FRAMECLK(n)	Indicates header at GBT Rx (n) ('1' -> DATA | '0' -> IDLE')
GBT_RX_O(n). data	Out	RX_FRAMECLK(n)	84-bit user data bus of GBT Link Rx (n)
GBT_TX_O(n). extraData_wideBus	Out	RX_FRAMECLK(n)	32-bit Wide-Bus extra user data of GBT Link Rx (n)

2.5.5. MGT Ports

The different ports of the MGT are device specific and are grouped as inputs and outputs in the records MGT_I and MGT_O respectively.

- Xilinx Virtex 6 specific MGT ports:

Table 16: Xilinx Virtex 6 specific MGT ports

Port	Dir	Clock Domain	Description
MGT_O.mgtLink(n).tx_p	Out	MGT SERIAL CLK	Positive serial Tx pin of MGT (n)
MGT_O.mgtLink(n).tx_n	Out	MGT SERIAL CLK	Negative serial Tx pin of MGT (n)
MGT_I.mgtLink(n).rx_p	In	MGT SERIAL CLK	Positive serial Rx pin of MGT (n)
MGT_I.mgtLink(n).rx_n	In	MGT SERIAL CLK	Negative serial Rx pin of MGT (n)
MGT_I.mgtLink(n).loopBack	In	Asynchronous	Selects loopback mode (see UG366 page 123)
MGT_I.mgtLink(n).tx_reset	In	Asynchronous	Assert to '1' for resetting the MGT Tx (n)
MGT_I.mgtLink(n).rx_reset	In	Asynchronous	Assert to '1' for resetting the MGT Rx (n)
MGT_I.mgtLink(n).tx_syncReset	In	Asynchronous	Assert to '1' for resetting the sync of MGT Tx (n) LATOPT
MGT_I.mgtLink(n).rx_syncReset	In	Asynchronous	Assert to '1' for resetting the sync of MGT Rx (n) LATOPT
MGT_O.mgtLink(n).tx_resetDone	Out	Asynchronous	When '1' indicates that MGT Tx (n) reset is done
MGT_O.mgtLink(n).rx_resetDone	Out	Asynchronous	When '1' indicates that MGT Rx (n) reset is done
MGT_O.mgtLink(n).tx_pllLkDet	Out	Asynchronous	When '1' indicates that MGT Tx PLL (n) is locked
MGT_O.mgtLink(n).rx_pllLkDet	Out	Asynchronous	When '1' indicates that MGT Rx PLL (n) is locked
MGT_I.mgtLink(n). rxBitSlip_enable	In	Asynchronous	Set to '1' for enabling the Rx bitslip ctrl of MGT Rx (n)
MGT_I.mgtLink(n). rxBitSlip_ctrl	In	Asynchronous	Selects Rx bitslip (n) mode ('0' -> AUTO | '1' -> MANUAL)
MGT_I.mgtLink(n). rxBitSlip_nbr	In	Asynchronous	Sets number of bitslips when MANUAL mode of Rx bitslip (n)
MGT_I.mgtLink(n). rxBitSlip_run	In	Asynchronous	Assert to '1' for triggering the Rx bitslip ctrl of MGT Rx (n)
MGT_I.mgtLink(n). rxBitSlip_oddRstEn	In	Asynchronous	Automatic reset control using the bitslip number
MGT_I.mgtLink(n). rxBitSlip_oddRstNbr	Out	Asynchronous	Number of reset performed by the BitSlip monitoring FSM
MGT_O.mgtLink(n).ready	Out	Asynchronous	When '1' indicates that MGT (n) is ready
MGT_O.mgtLink(n).rxWordClkReady	Out	RX_WORDCLK(n)	When '1' indicates that RX_WORDCLK(n) is ready
MGT_I.mgtLink(n).conf_diff	In	Asynchronous	Sets differential swing of MGT Tx (n) (see UG366 page 173)
MGT_I.mgtLink(n).conf_pstEmph	In	Asynchronous	Sets Post-Emphasis of MGT Tx (n) (see UG366 page 173)
MGT_I.mgtLink(n).conf_preEmph	In	Asynchronous	Sets Pre-Emphasis of MGT Tx (n) (see UG366 page 173)
MGT_I.mgtLink(n).conf_eqMix	In	Asynchronous	Sets equalization of MGT Rx (n) (see UG366 page 193)
MGT_I.mgtLink(n).conf_txPol	In	Asynchronous	Sets polarity of MGT Tx (n) (see UG366 page 167)
MGT_I.mgtLink(n).conf_rxPol	In	Asynchronous	Sets polarity of MGT Rx (n) (see UG366 page 212)
MGT_I.mgtLink(n).drp_dAddr	In	DRP CLK	DRP address bus MGT (n) (see UG366 page 125)
MGT_I.mgtLink(n).drp_dEn	In	DRP CLK	DRP enable of MGT (n) (see UG366 page 125)
MGT_I.mgtLink(n).drp_dI	In	DRP CLK	DRP write data bus of MGT (n) (see UG366 page 125)

MGT_I.mgtLink(n).drp_dWe	In	DRP CLK	DRP write enable of MGT (n) (see UG366 page 125)
MGT_O.mgtLink(n).drp_dRdy	Out	DRP CLK	DRP ready of MGT (n) (see UG366 page 125)
MGT_O.mgtLink(n).drp_dRpDo	Out	DRP CLK	DRP read data bus of MGT (n) (see UG366 page 125)
MGT_I.mgtLink(n).prbs_txEn	In	TX_WORDCLK(n)	Sets PRBS generator of MGT Tx (n) (see UG366 page 163)
MGT_I.mgtLink(n).prbs_rxEn	In	RX_WORDCLK(n)	Sets PRBS checker of MGT Rx (n) (see UG366 page 214)
MGT_I.mgtLink(n).prbs_forcErr	In	TX_WORDCLK(n)	Assert to '1' to inject an error in PRBS of MGT Tx (n)
MGT_I.mgtLink(n).prbs_errCntRst	In	RX_WORDCLK(n)	Assert to '1' to reset the PRBS error cntr of MGT Rx (n)
MGT_O.mgtLink(n).prbs_rxErr	Out	RX_WORDCLK(n)	When '1' indicates an PRBS error found by MGT Rx (n)

- Xilinx Kintex 7 & Virtex 7 specific MGT ports:

Table 17: Xilinx Kintx 7 & Virtex 7 specific MGT ports

Port	Dir	Clock Domain	Description
MGT_O.mgtLink(n).tx_p	Out	MGT SERIAL CLK	Positive serial Tx pin of MGT (n)
MGT_O.mgtLink(n).tx_n	Out	MGT SERIAL CLK	Negative serial Tx pin of MGT (n)
MGT_I.mgtLink(n).rx_p	In	MGT SERIAL CLK	Positive serial Rx pin of MGT (n)
MGT_I.mgtLink(n).rx_n	In	MGT SERIAL CLK	Negative serial Rx pin of MGT (n)
MGT_I.mgtLink(n).loopBack	In	Asynchronous	Selects loopback mode (see UG476 page 88)
MGT_I.mgtLink(n).tx_reset	In	Asynchronous	Assert to '1' for resetting the MGT Tx (n)
MGT_I.mgtLink(n).rx_reset	In	Asynchronous	Assert to '1' for resetting the MGT Rx (n)
MGT_O.mgtLink(n).tx_resetDone	Out	Asynchronous	When '1' indicates that MGT Tx (n) reset is done
MGT_O.mgtLink(n).rx_resetDone	Out	Asynchronous	When '1' indicates that MGT Rx (n) reset is done
MGT_O.mgtLink(n).cpllLock	Out	Asynchronous	When '1' indicated that the CPLL is locked
MGT_O.mgtLink(n).tx_fsmResetDone	Out	TX_WORDCLK(n)	When '1' indicates that MGT Tx (n) initialization is done
MGT_O.mgtLink(n).rx_fsmResetDone	Out	RX_WORDCLK(n)	When '1' indicates that MGT Rx (n) initialization is done
MGT_O.mgtLink(n).rxCdrLock	Out	Asynchronous	When '1' indicates that the CDR of MGT Rx (n) is locked
MGT_I.mgtLink(n). rxBitSlip_enable	In	Asynchronous	Set to '1' for enabling the Rx bitslip ctrl of MGT Rx (n)
MGT_I.mgtLink(n). rxBitSlip_ctrl	In	Asynchronous	Selects Rx bitslip (n) mode ('0' -> AUTO | '1' -> MANUAL)
MGT_I.mgtLink(n). rxBitSlip_run	In	Asynchronous	Sets nbr of bitslips when MANUAL mode of Rx bitslip (n)
MGT_I.mgtLink(n). rxBitSlip_run	In	Asynchronous	Assert to '1' for triggering the Rx bitslip ctrl of MGT Rx (n)
MGT_I.mgtLink(n). rxBitSlip_oddRstEn	In	Asynchronous	Automatic reset control using the bitslip number
MGT_I.mgtLink(n). rxBitSlip_oddRstNbr	Out	Asynchronous	Number of reset performed by the BitSlip monitoring FSM
MGT_O.mgtLink(n).ready	Out	Asynchronous	When '1' indicates that MGT (n) is ready
MGT_O.mgtLink(n).rxWordClkReady	Out	RX_WORDCLK(n)	When '1' indicates that RX_WORDCLK(n) is ready
MGT_O.mgtLink(n).rx_phMonitor	Out	Asynchronous	Buffer Bypass related port (see UG476 page 231)
MGT_O.mgtLink(n).rx_phSlipMonitor	Out	Asynchronous	Buffer Bypass related port (see UG476 page 231)
MGT_I.mgtLink(n).conf_diffCtrl	In	Asynchronous	Sets differential swing of MGT Tx (n) (see UG476 page 146)
MGT_I.mgtLink(n).conf_postCursor	In	Asynchronous	Sets Post-Emphasis of MGT Tx (n) (see UG476 page 146)
MGT_I.mgtLink(n).conf_preCursor	In	Asynchronous	Sets Pre-Emphasis of MGT Tx (n) (see UG476 page 146)
MGT_I.mgtLink(n).conf_txPol	In	Asynchronous	Sets polarity of MGT Tx (n) (see UG476 page 139)
MGT_I.mgtLink(n).conf_rxPol	In	Asynchronous	Sets polarity of MGT Rx (n) (see UG476 page 212)
MGT_I.mgtLink(n).drp_addr	In	DRP CLK	DRP address bus MGT (n) (see UG476 page 89)
MGT_I.mgtLink(n).drp_en	In	DRP CLK	DRP enable of MGT (n) (see UG476 page 89)
MGT_I.mgtLink(n).drp_di	In	DRP CLK	DRP write data bus of MGT (n) (see UG476 page 89)
MGT_I.mgtLink(n).drp_we	In	DRP CLK	DRP write enable of MGT (n) (see UG476 page 89)
MGT_O.mgtLink(n).drp_rdy	Out	DRP CLK	DRP ready of MGT (n) (see UG476 page 89)
MGT_O.mgtLink(n).drp_do	Out	DRP CLK	DRP read data bus of MGT (n) (see UG476 page 89)
MGT_I.mgtLink(n).prbs_txSel	In	TX_WORDCLK(n)	Sets PRBS generator of MGT Tx (n) (see UG476 page 135)
MGT_I.mgtLink(n).prbs_rxSel	In	RX_WORDCLK(n)	Sets PRBS checker of MGT Rx (n) (see UG476 page 212)
MGT_I.mgtLink(n).prbs_txForceErr	In	TX_WORDCLK(n)	Assert to '1' to inject an error in PRBS of MGT Tx (n)
MGT_I.mgtLink(n).prbs_rxCntReset	In	RX_WORDCLK(n)	Assert to '1' to reset the PRBS error cntr of MGT Rx (n)
MGT_O.mgtLink(n).prbs_rxErr	Out	RX_WORDCLK(n)	When '1' indicates an PRBS error found by MGT Rx (n)

- Xilinx Kintex Ultrascale specific MGT ports:

Table 18: Xilinx Kintx Ultrascale specific MGT ports

Port	Dir	Clock Domain	Description
MGT_O.mgtLink(n).tx_p	Out	MGT SERIAL CLK	Positive serial Tx pin of MGT (n)
MGT_O.mgtLink(n).tx_n	Out	MGT SERIAL CLK	Negative serial Tx pin of MGT (n)
MGT_I.mgtLink(n).rx_p	In	MGT SERIAL CLK	Positive serial Rx pin of MGT (n)
MGT_I.mgtLink(n).rx_n	In	MGT SERIAL CLK	Negative serial Rx pin of MGT (n)
MGT_I.mgtLink(n).loopBack	In	Asynchronous	Selects loopback mode (see UG476 page 88)
MGT_I.mgtLink(n).tx_reset	In	Asynchronous	Assert to '1' for resetting the MGT Tx (n)
MGT_I.mgtLink(n).rx_reset	In	Asynchronous	Assert to '1' for resetting the MGT Rx (n)
MGT_O.mgtLink(n).tx_resetDone	Out	Asynchronous	When '1' indicates that MGT Tx (n) reset is done
MGT_O.mgtLink(n).rx_resetDone	Out	Asynchronous	When '1' indicates that MGT Rx (n) reset is done
MGT_O.mgtLink(n).tx_fsmResetDone	Out	TX_WORDCLK(n)	When '1' indicates that MGT Tx (n) initialization is done
MGT_O.mgtLink(n).rx_fsmResetDone	Out	RX_WORDCLK(n)	When '1' indicates that MGT Rx (n) initialization is done
MGT_I.mgtLink(n). rxBitSlip_enable	In	Asynchronous	Set to '1' for enabling the Rx bitslip ctrl of MGT Rx (n)
MGT_I.mgtLink(n). rxBitSlip_ctrl	In	Asynchronous	Selects Rx bitslip (n) mode ('0' -> AUTO | '1' -> MANUAL)
MGT_I.mgtLink(n). rxBitSlip_run	In	Asynchronous	Sets nbr of bitslips when MANUAL mode of Rx bitslip (n)
MGT_I.mgtLink(n). rxBitSlip_run	In	Asynchronous	Assert to '1' for triggering the Rx bitslip ctrl of MGT Rx (n)
MGT_I.mgtLink(n). rxBitSlip_oddRstEn	In	Asynchronous	Automatic reset control using the bitslip number
MGT_I.mgtLink(n). rxBitSlip_oddRstNbr	Out	Asynchronous	Number of reset performed by the BitSlip monitoring FSM
MGT_O.mgtLink(n).ready	Out	Asynchronous	When '1' indicates that MGT (n) is ready
MGT_O.mgtLink(n).rxWordClkReady	Out	RX_WORDCLK(n)	When '1' indicates that RX_WORDCLK(n) is ready
MGT_I.mgtLink(n).conf_diffCtrl	In	Asynchronous	Sets differential swing of MGT Tx (n) (see UG476 page 146)
MGT_I.mgtLink(n).conf_postCursor	In	Asynchronous	Sets Post-Emphasis of MGT Tx (n) (see UG476 page 146)
MGT_I.mgtLink(n).conf_preCursor	In	Asynchronous	Sets Pre-Emphasis of MGT Tx (n) (see UG476 page 146)
MGT_I.mgtLink(n).conf_txPol	In	Asynchronous	Sets polarity of MGT Tx (n) (see UG476 page 139)
MGT_I.mgtLink(n).conf_rxPol	In	Asynchronous	Sets polarity of MGT Rx (n) (see UG476 page 212)
MGT_I.mgtLink(n).drp_addr	In	DRP CLK	DRP address bus MGT (n) (see UG476 page 89)
MGT_I.mgtLink(n).drp_en	In	DRP CLK	DRP enable of MGT (n) (see UG476 page 89)
MGT_I.mgtLink(n).drp_di	In	DRP CLK	DRP write data bus of MGT (n) (see UG476 page 89)
MGT_I.mgtLink(n).drp_we	In	DRP CLK	DRP write enable of MGT (n) (see UG476 page 89)
MGT_O.mgtLink(n).drp_rdy	Out	DRP CLK	DRP ready of MGT (n) (see UG476 page 89)
MGT_O.mgtLink(n).drp_do	Out	DRP CLK	DRP read data bus of MGT (n) (see UG476 page 89)

- Altera Cyclone V GT specific MGT ports:

Table 19: Altera Cyclone GT specific MGT ports - Common

Port	Dir	Clock Domain	Description
MGT_I.mgtCommon.reconf_reset	In	Reconf clk	Reconfig. reset
MGT_I.mgtCommon.reconf_clk	In	Reconf clk	Reconfig. clock
MGT_I.mgtCommon.reconf_avmm_addr	In	Reconf clk	Reconf. Avalon MM bus (address)
MGT_I.mgtCommon.reconf_avmm_read	In	Reconf clk	Reconf. Avalon MM bus (read)
MGT_I.mgtCommon.reconf_write	In	Reconf clk	Reconf. Avalon MM bus (write)
MGT_I.mgtCommon.reconf_writedata	In	Reconf clk	Reconf. Avalon MM bus (writedata)
MGT_O.mgtCommon.reconf_readdata	Out	Reconf clk	Reconf. Avalon MM bus (readdata)
MGT_O.mgtCommon.reconf_waitrequest	Out	Reconf clk	Reconf. Avalon MM bus (waitrequest)

Table 20: Altera Cyclone GT specific MGT ports - GBT Link

Port	Dir	Clock Domain	Description
MGT_O.mgtLink(n).txSerialData	Out	MGT SERIAL CLK	Serial Tx pin of MGT (n)
MGT_I.mgtLink(n).rxSerialData	In	MGT SERIAL CLK	Serial Rx pin of MGT (n)
MGT_I.mgtLink(n).loopBack	In	Asynchronous	Assert to '1' for loopback mode (see CV-53006 page 6-1)
MGT_I.mgtLink(n).tx_reset	In	Asynchronous	Assert to '1' for resetting the MGT Tx (n)
MGT_I.mgtLink(n).rx_reset	In	Asynchronous	Assert to '1' for resetting the MGT Rx (n)
MGT_O.mgtLink(n).tx_ready	Out	Asynchronous	When '1' indicates that MGT Tx (n) is ready
MGT_O.mgtLink(n).rx_ready	Out	Asynchronous	When '1' indicates that MGT Rx (n) is ready
MGT_O.mgtLink(n).ready	Out	Asynchronous	When '1' indicates that MGT (n) is ready
MGT_I.mgtLink(n).tx_polarity	In	Asynchronous	Assert to '1' for inverting polarity of MGT Tx (n)
MGT_I.mgtLink(n).rx_polarity	In	Asynchronous	Assert to '1' for inverting polarity of MGT Rx (n)
MGT_I.mgtLink(n).rxBitSlip_enable	In	Asynchronous	Assert to '1' to enable the Rx bitslip ctrl of MGT Rx (n)
MGT_I.mgtLink(n).rxBitSlip_ctrl	In	Asynchronous	Selects Rx bitslip (n) mode ('0' -> AUTO | '1' -> MANUAL)
MGT_I.mgtLink(n).rxBitSlip_nbr	In	Asynchronous	Sets nbr of bitslips when MANUAL mode of Rx bitslip (n)
MGT_I.mgtLink(n). rxBitSlip_run	In	Asynchronous	Assert to '1' for triggering the Rx bitslip ctrl of MGT Rx (n)
MGT_I.mgtLink(n). rxBitSlip_oddRstEn	In	Asynchronous	Automatic reset control using the bitslip number
MGT_I.mgtLink(n). rxBitSlip_oddRstNbr	Out	Asynchronous	Number of reset performed by the BitSlip monitoring FSM
MGT_O.mgtLink(n).rxWordClkReady	Out	RX_WORDCLK(n)	When '1' indicates that RX_WORDCLK(n) is ready
MGT_O.mgtLink(n).rxIsLocked_toRef	Out	Asynchronous	When '1' indicates that MGT Rx (n) is locked to data
MGT_O.mgtLink(n).rxIsLocked_toData	Out	Asynchronous	When '1' indicates that MGT Rx (n) is locked to MGT RefClk
MGT_O.mgtLink(n).txCal_busy	Out	MGMT_CLK	When '1' indicates that MGT Tx (n) calibration is active
MGT_O.mgtLink(n).rxCal_busy	Out	MGMT_CLK	When '1' indicates that MGT Rx (n) calibration is active

- Altera Stratix V GX specific MGT ports:

Table 21: Altera Stratix V GX specific MGT ports - Common

Port	Dir	Clock Domain	Description
MGT_I.mgtCommon.reconf_reset	In	Reconf clk	Reconfig. reset
MGT_I.mgtCommon.reconf_clk	In	Reconf clk	Reconfig. clock
MGT_I.mgtCommon.reconf_avmm_addr	In	Reconf clk	Reconf. Avalon MM bus (address)
MGT_I.mgtCommon.reconf_avmm_read	In	Reconf clk	Reconf. Avalon MM bus (read)
MGT_I.mgtCommon.reconf_write	In	Reconf clk	Reconf. Avalon MM bus (write)
MGT_I.mgtCommon.reconf_writedata	In	Reconf clk	Reconf. Avalon MM bus (writedata)
MGT_O.mgtCommon.reconf_readdata	Out	Reconf clk	Reconf. Avalon MM bus (readdata)
MGT_O.mgtCommon.reconf_waitrequest	Out	Reconf clk	Reconf. Avalon MM bus (waitrequest)

Table 22: Altera Stratix V GX specific MGT ports - GBT Link

Port	Dir	Clock Domain	Description
MGT_O.mgtLink(n).txSerialData	Out	MGT SERIAL CLK	Serial Tx pin of MGT (n)
MGT_I.mgtLink(n).rxSerialData	In	MGT SERIAL CLK	Serial Rx pin of MGT (n)
MGT_I.mgtLink(n).loopBack	In	Asynchronous	Assert to '1' for loopback mode (see CV-53006 page 6-1)
MGT_I.mgtLink(n).tx_reset	In	Asynchronous	Assert to '1' for resetting the MGT Tx (n)
MGT_I.mgtLink(n).rx_reset	In	Asynchronous	Assert to '1' for resetting the MGT Rx (n)
MGT_O.mgtLink(n).tx_ready	Out	Asynchronous	When '1' indicates that MGT Tx (n) is ready
MGT_O.mgtLink(n).rx_ready	Out	Asynchronous	When '1' indicates that MGT Rx (n) is ready
MGT_O.mgtLink(n).ready	Out	Asynchronous	When '1' indicates that MGT (n) is ready
MGT_I.mgtLink(n).tx_polarity	In	Asynchronous	Assert to '1' for inverting polarity of MGT Tx (n)
MGT_I.mgtLink(n).rx_polarity	In	Asynchronous	Assert to '1' for inverting polarity of MGT Rx (n)
MGT_I.mgtLink(n).rxBitSlip_enable	In	Asynchronous	Assert to '1' to enable the Rx bitslip ctrl of MGT Rx (n)
MGT_I.mgtLink(n).rxBitSlip _ctrl	In	Asynchronous	Selects Rx bitslip (n) mode ('0' -> AUTO | '1' -> MANUAL)
MGT_I.mgtLink(n).rxBitSlip_nbr	In	Asynchronous	Sets nbr of bitslips when MANUAL mode of Rx bitslip (n)
MGT_I.mgtLink(n).rxBitSlip _run	In	Asynchronous	Assert to '1' for triggering the Rx bitslip ctrl of MGT Rx (n)
MGT_I.mgtLink(n). rxBitSlip_oddRstEn	In	Asynchronous	Automatic reset control using the bitslip number
MGT_I.mgtLink(n).rxBitSlip_oddRstNbr	Out	Asynchronous	Number of reset performed by the BitSlip monitoring FSM
MGT_O.mgtLink(n).rxWordClkReady	Out	RX_WORDCLK(n)	When '1' indicates that RX_WORDCLK(n) is ready
MGT_O.mgtLink(n).rxIsLocked_toRef	Out	Asynchronous	When '1' indicates that MGT Rx (n) is locked to data
MGT_O.mgtLink(n).rxIsLocked_toData	Out	Asynchronous	When '1' indicates that MGT Rx (n) is locked to MGT RefClk
MGT_O.mgtLink(n).txCal_busy	Out	MGMT_CLK	When '1' indicates that MGT Tx (n) calibration is active
MGT_O.mgtLink(n).rxCal_busy	Out	MGMT_CLK	When '1' indicates that MGT Rx (n) calibration is active

- Altera Arrai 10 GX specific MGT ports:

Table 23: Altera Stratix V GX specific MGT ports - Common

Port	Dir	Clock Domain	Description
MGT_I.mgtCommon.reconf_reset	In	Reconf clk	Reconfig. reset
MGT_I.mgtCommon.reconf_clk	In	Reconf clk	Reconfig. clock
MGT_I.mgtCommon.reconf_avmm_addr	In	Reconf clk	Reconf. Avalon MM bus (address)
MGT_I.mgtCommon.reconf_avmm_read	In	Reconf clk	Reconf. Avalon MM bus (read)
MGT_I.mgtCommon.reconf_avmm_write	In	Reconf clk	Reconf. Avalon MM bus (write)

MGT_I.mgtCommon.reconf_avmm_writedata	In	Reconf clk	Reconf. Avalon MM bus (writedata)
MGT_O.mgtCommon.reconf_avmm_readdata	Out	Reconf clk	Reconf. Avalon MM bus (readdata)
MGT_O.mgtCommon.reconf_avmm_waitrequest	Out	Reconf clk	Reconf. Avalon MM bus (waitrequest)

Table 24: Altera Arria 10 GX specific MGT ports - GBT Link

Port	Dir	Clock Domain	Description
MGT_O.mgtLink(n).txSerialData	Out	MGT SERIAL CLK	Serial Tx pin of MGT (n)
MGT_I.mgtLink(n).rxSerialData	In	MGT SERIAL CLK	Serial Rx pin of MGT (n)
MGT_I.mgtLink(n).loopBack	In	Asynchronous	Assert to '1' for loopback mode (see CV-53006 page 6-1)
MGT_I.mgtLink(n).tx_reset	In	Asynchronous	Assert to '1' for resetting the MGT Tx (n)
MGT_I.mgtLink(n).rx_reset	In	Asynchronous	Assert to '1' for resetting the MGT Rx (n)
MGT_O.mgtLink(n).tx_ready	Out	Asynchronous	When '1' indicates that MGT Tx (n) is ready
MGT_O.mgtLink(n).rx_ready	Out	Asynchronous	When '1' indicates that MGT Rx (n) is ready
MGT_O.mgtLink(n).ready	Out	Asynchronous	When '1' indicates that MGT (n) is ready
MGT_I.mgtLink(n).tx_polarity	In	Asynchronous	Assert to '1' for inverting polarity of MGT Tx (n)
MGT_I.mgtLink(n).rx_polarity	In	Asynchronous	Assert to '1' for inverting polarity of MGT Rx (n)
MGT_I.mgtLink(n).rxBitSlip_enable	In	Asynchronous	Assert to '1' to enable the Rx bitslip ctrl of MGT Rx (n)
MGT_I.mgtLink(n).rxBitSlip _ctrl	In	Asynchronous	Selects Rx bitslip (n) mode ('0' -> AUTO | '1' -> MANUAL)
MGT_I.mgtLink(n).rxBitSlip_nbr	In	Asynchronous	Sets nbr of bitslips when MANUAL mode of Rx bitslip (n)
MGT_I.mgtLink(n).rxBitSlip _run	In	Asynchronous	Assert to '1' for triggering the Rx bitslip ctrl of MGT Rx (n)
MGT_I.mgtLink(n). rxBitSlip_oddRstEn	In	Asynchronous	Automatic reset control using the bitslip number
MGT_I.mgtLink(n).rxBitSlip_oddRstNbr	Out	Asynchronous	Number of reset performed by the BitSlip monitoring FSM
MGT_O.mgtLink(n).rxWordClkReady	Out	RX_WORDCLK(n)	When '1' indicates that RX_WORDCLK(n) is ready
MGT_O.mgtLink(n).rxIsLocked_toRef	Out	Asynchronous	When '1' indicates that MGT Rx (n) is locked to data
MGT_O.mgtLink(n).rxIsLocked_toData	Out	Asynchronous	When '1' indicates that MGT Rx (n) is locked to MGT RefClk
MGT_O.mgtLink(n).txCal_busy	Out	MGMT_CLK	When '1' indicates that MGT Tx (n) calibration is active
MGT_O.mgtLink(n).rxCal_busy	Out	MGMT_CLK	When '1' indicates that MGT Rx (n) calibration is active

3. Example Designs

3.1. Example Design Overview

Besides the GBT-FPGA Core source files, the GBT-FPGA project delivers example designs for some selected FPGA-based cards and the most common FPGA devkits. These example designs follow the same concept of unification as the GBT-FPGA Core, so all of them share the same structure and many of the source files (only vendor specific files are not shared). This scheme facilitates the user support and the port of these example designs to other FPGA-based cards if needed.

Although all these example designs follow the same structure, there are two different types of example designs in terms of number of GBT Links implemented. All of the example designs are single-link except a multi-link example design for the AMC40, an FPGA-based card featuring an Altera Stratix V GX that has been developed at *Centre de Physique des Particules de Marseille* (CPPM).

Table 25: Available Reference Designs per Vendor/device

VENDOR	FPGA type	Reference design	Manufacturer	Available	Configuration
ALTERA	Cyclone V GT	GBTx_SAT board	CERN	Soon	1 GBT bank, 1 GBT link
		Cyclone V GT dekit	ALTERA	Yes	1 GBT bank, 1 GBT link
	Stratix V GX	AMC40	CPPM – LHCb - ALICE	Yes	2 GBT banks, 3 GBT links (see below)
XILINX	Virtex 6	ML605	Xilinx	Yes	1 GBT bank, 1 GBT link
		GLIB	CERN	Yes	1 GBT bank, 1 GBT link
	Kintex 7	KC705	XILINX	Yes	1 GBT bank, 1 GBT link
		FC7	CERN- CMS	Yes	1 GBT bank, 1 GBT link
	Virtex 7	VC707	XILINX	Yes	1 GBT bank, 1 GBT link
	Kintex Ultrascale	KCU105	XILINX	Yes	1 GBT bank, 1 GBT link

The **single-link example design**, shown in Figure 28, consists of a GBT Bank implementing one GBT Link. The Tx is connected to a pattern generator whilst the Rx is connected to a pattern checker. The serial lanes of the GBT Link may be connected in loopback but also to any other GBT compatible device. Regarding the versions of the GBT-FPGA Core, it is possible to implement the two versions since the example design features the components needed for implementing the Latency-Optimized version and these components are also compatible with the Standard version (just change the option in the GBT User Setup File. Besides the version of the GBT-FPGA Core, the user may also chose the type of encoding (GBT-Frame or Wide-Bus).

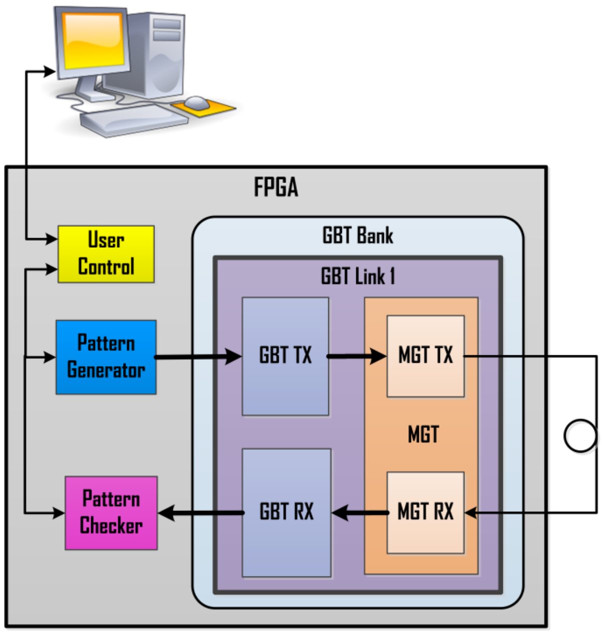


Figure 28: GBT-FPGA single-link example design simplified block diagram

The **multi-link example design** (only available for AMC40 (Stratix V GX)), shown in Figure 29, consists of four GBT Links implemented into two GBT Banks. The GBT Bank 1 implements one GBT Link configured with Standard Tx and Latency-Optimized Rx whilst the GBT Bank 2 implements three GBT Links configured with Latency-Optimized Txs and Standard Rxs. This example design has been optimized for the previously mentioned configuration, but the GBT Banks will also operate in fully Standard version.

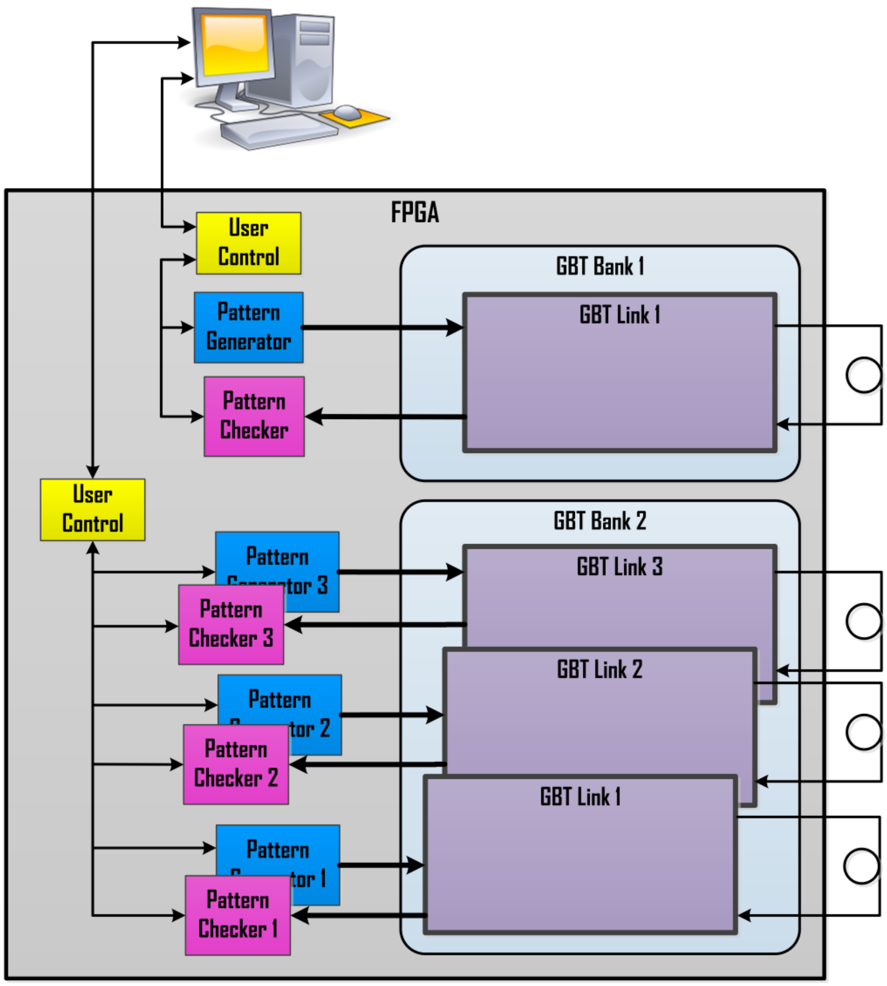


Figure 29: GBT-FPGA multi-link example design simplified block diagram

Like in the single-link example design, the Txs are connected to respective pattern generators whilst the Rxs are connected to respective a pattern checkers. The serial lanes of the GBT Links may be connected in loopback but also to any other GBT compatible devices. Regarding logic and clocking resources, the two GBT Banks are independent, only sharing the MGT reference clock for simplicity reasons. Besides the version of the GBT-FPGA Core, the user may also chose the type of encoding (GBT-Frame or Wide-Bus). It is important to mention that **this example design may be used as a reference for implementing multi-GBT Link systems in other Stratix V GX FPGA-based cards and even in different model/vendor FPGA-based cards** due to the similarities between the different GBT-FPGA Core implementations.

The control of the example designs is done through a PC using HDL cores and software provided by the FPGA vendors (Xilinx's ChipScope and Altera's In-System Sources and Probes/SignalTap II).

3.2. Running the Example Designs

3.2.1. Running the Example Design in Xilinx FPGAs using ChipScope

The procedure for running the example designs for Xilinx FPGAs is explained in the GBT-FPGA Video Tutorials that can be found in the web site of the GBT-FPGA project (https://espace.cern.ch/GBT-Project/GBT-FPGA).

3.2.2. Running the Example Design in Altera FPGAs using In-system Sources and Probes

The procedure for running the example designs for Altera FPGAs is explained in the GBT-FPGA Video Tutorials that can be found in the web site of the GBT-FPGA project (https://espace.cern.ch/GBT-Project/GBT-FPGA).

4. FPGA Particularities & Hardware Recommendations

4.1. FPGA Particularities

4.1.1. Xilinx FPGAs

4.1.1.a. Virtex 6

- The frequency of the MGT reference clock is 240MHz.
- When using the Latency-Optimized Rx version, the RX_FRAMECLK must be aligned in order to maintain the correct phase (a RX_FRAMECLK phase aligner is provided with the GBT-FPGA example design).

4.1.1.b. Kintex 7, Virtex 7 & Kintex Ultrascale

- The frequency of the MGT reference clock is 120MHz.
- When using the Latency-Optimized Rx version, the RX_FRAMECLK must be aligned in order to maintain the correct phase (a RX_FRAMECLK phase aligner is provided with the GBT-FPGA example design).

4.1.2. Altera FPGAs

4.1.2.a. Cyclone V GT

- The frequency of the MGT reference clock is 120MHz.
- When using the Latency-Optimized Tx version, the phase of the TX_WORDCLK may be 0deg or 180deg with respect to the phase of the MGT REFCLK due to a phase uncertainty issue after MGT reset, present in all Altera GT transceivers of the V series. For that reason it is necessary the utilization of a phase monitoring that resets the MGT Tx when the phase of the TX_WORDCLK is not de one desired (the TX_WORDCLK monitor is already integrated into the Latency-Optimized MGT).
- When using the Latency-Optimized Tx version, the TX_FRAMECLK may need to be aligned in order to find the correct sampling point in the gearbox of the GBT Tx when crossing from TX_FRAMECLK to TX_WORDCLK domains (a TX_FRAMECLK phase aligner is provided with the GBT-FPGA example design)
- When using the Latency-Optimized Rx version, the RX_FRAMECLK must be aligned in order to maintain the correct phase (a RX_FRAMECLK phase aligner is provided with the GBT-FPGA example design).

Note for Latency-Optimized version in Cyclone V GT: achieving timing closure was a really difficult task in Cyclone V GT FPGAs (which are low-end devices). For that reason, the latency-optimized mode was removed for this FPGA.

4.1.2.b. Stratix V GX

- The frequency of the MGT reference clock is 120MHz.
- When using the Latency-Optimized Tx version, the phase of the TX_WORDCLK may be 0deg or 180deg with respect to the phase of the MGT REFCLK due to a phase uncertainty issue after MGT reset, present in all Altera GT transceivers of the V series. For that reason it is necessary the utilization of a phase monitoring that resets the MGT Tx when the phase of the TX_WORDCLK is not de one desired (the TX_WORDCLK monitor is already integrated into the Latency-Optimized MGT).
- When using the Latency-Optimized Tx version, the TX_FRAMECLK may need to be aligned in order to find the correct sampling point in the gearbox of the GBT Tx when crossing from TX_FRAMECLK to TX_WORDCLK domains (a TX_FRAMECLK phase aligner is provided with the GBT-FPGA example design)

When using the Latency-Optimized Rx version, the RX_FRAMECLK must be aligned in order to maintain the correct phase (a RX_FRAMECLK phase aligner is provided with the GBT-FPGA example design).

4.1.2.c. Arria 10 GX

- The frequency of the MGT reference clock is 240MHz.
- When using the Latency-Optimized Rx version, the RX_FRAMECLK must be aligned in order to maintain the correct phase (a RX_FRAMECLK phase aligner is provided with the GBT-FPGA example design).

4.2. Hardware Recommendations

- When using the Latency-Optimized version, the MGT reference clock must have very low jitter. The allowed values may be consulted in the datasheet of the targeted FPGA.
- Achieving timing closure may be complicated in low-end FPGAs (e.g. Cyclone V GT), mainly when implementing multi-GBT Link systems. For that reason it is recommended the utilisation of high-end FPGA (e.g. Virtex 6) when implementing multi-GBT Link systems.

5. References

- **The GBT Project Home Page:**

  https://espace.cern.ch/GBT-Project

- **The GBTx ASIC User Guide**

  https://espace.cern.ch/GBT-Project/GBTX/Manuals/gbtxManual.pdf

- **The GBT-FPGA Project Home Page**

  https://espace.cern.ch/GBT-Project/GBT-FPGA

- ***The GBT_FPGA: one unified core for multiple users***, ppt, M. Barros Marin, Feb 2014

  https://svnweb.cern.ch/cern/wsvn/ph-ese/be/gbt_fpga/tags/doc/GBT-FPGA_students-fellows_ManoelBarrosMarin_2014_02_05.pdf

- ***The GBT_FPGA project***, M. Barros Marin, S. Baron, ACES 2014

  https://svnweb.cern.ch/cern/wsvn/ph-ese/be/gbt_fpga/tags/doc/gbt_fpga_aces2014_poster.pdf

- **The GBT encoding scheme:** "An Error-Correcting Line Coding ASIC for a HEP Rad-Hard Multi-GigaBit Optical Link", G. Papotti, Proc. 2nd Conference on Ph.D. Research in Microelectronics and Electronics (PRIME 2006), Otranto (Lecce), Italy, 12-15 June 2006, pp.225-8.

- **The GBT-FPGA Core: Features and Challenges**, M. Barros Marin et al., 2015 JINST 7 P01075.

APPENDIX A: Frequently Asked Questions

- **I am a new user: how do I start?**

See 2.4.1.

- **What is the difference between a GBT Bank and a GBT Link?**

The GBT Bank is the main module of the GBT-FPGA Core. Each GBT Bank may include several GBT Links (up to four in Xilinx FPGAs or up to three in Altera FPGAs).

The GBT Link is the actual channel of the link. It is composed by a GBT Tx (that scrambles and encodes the transmitted parallel data), a Multi-Gigabit Transceiver (MGT) (that serializes, transmits, receives and de-serializes the data) and a GBT Rx (that aligns, decodes and descrambles the incoming data stream).

- **Where can I choose the version I want to implement?**

The different parameters of the GBT Bank, such as optimization or frame type, are set through the GBT User Setup File (see **Error! Reference source not found.**). This file can be found in:

"… \gbt_bank\<vendor>_<device>\<vendor>_<device>_gbt_banks_user_setup.vhd"

(e.g. *"…/gbt_bank\altera_cv\alt_cv_gbt_banks_user_setup.vhd"*)

- **Where can I define the frame type?**

The different parameters of the GBT Bank, such as optimization or frame type, are set through the GBT User Setup File (see **Error! Reference source not found.**). This file can be found in:

"… \gbt_bank\<vendor>_<device>\<vendor>_<device>_gbt_banks_user_setup.vhd"

(e.g. *"…/gbt_bank\altera_cv\alt_cv_gbt_banks_user_setup.vhd"*)

- **How can I instantiate several GBT links in my FPGA?**

GBT links are grouped in GBT Banks. Each GBT Bank may have up to 4 GBT Links in Xilinx FPGAs or up to 3 GBT Links in Altera FPGAs. The user may instantiate several GBT Banks in parallel. The maximum number of GBT Banks is device dependant.

The example design for Altera Stratix V GX can be used as a reference of multi-GBT Links implementation. It instantiates one GBT Bank with one GBT Link and a second GBT Bank with three GBT Links.

- **How many GBT links can I fit within an FPGA?**

The maximum number of GBT Links is device dependent.

When implementing the Standard version in Xilinx or Stratix V GX devices, it is possible to have one GBT Link per MGT of the FPGA. In Cyclone V GT, at least one MGT Tx has to be used as reference clock by the other MGTs of the FPGA.

When implementing the Latency-Optimized version, the number of GBT Link also depends of the clocking resources available.

- **What is the difference between the Standard and the Latency Optimized versions?**

See 2.1.2.

- **Why the Latency Optimized version is so tricky?**

The use of registers for Cross Domain Crossing instead of FIFOs or DPRAMs and the dynamic realignment of the phase of the clocks in order to maintain the latency low, fixed and deterministic, introduce new technical challenges. Extra logic and clocking resources as well as timing and floorplanning constraints are needed for performing the clock alignments and for monitoring the correct phase relationship between clocks in some critical paths, thus complicating the implementation of the system.

- **What is critical for using a Latency Optimized version?**

The two main critical points when implementing the latency-optimized version of the GBT-FPGA are metastability issues and clocking resources availability.

In order to avoid **metastability issues** due to the register-based Clock Domain Crossing (CDC) of the latency-optimized version, the quality of the reference clocks in terms of jitter is very important. In addition, it is also necessary to ensure the correct relationship between the different clocks in the critical paths (using timing and floorplannig constraints, phase aligners, phase monitoring, etc.). Please note that the drift of the different clocks due to voltage or temperature variations may also lead to metastability issues so further study of the system under different conditions is recommended.

The other critical point when using the latency-optimized version is the intensive use of **clocking resources** due to the numerous clocks domains generated and the need of controlling the phase relationship between them.

- **Can I mix Standard and Latency-Optimized links within one MGT bank?**

No, all GBT Links of the same GBT Bank will share the same configuration.

- **Can I have one GBT link with the Rx in Standard version and the Tx in Latency-Optimized version (or reversely)?**

Yes, it is possible to use Latency-Optimized Tx and Standard Rx and vice versa.

Note that all GBT Links of the same GBT Bank will share the same configuration.

- **How can I integrate only the files I need in my project?**

The GBT-FPGA team provides TCL scripts (see 2.4.3) for adding the GBT-FPGA sources files into the project of the vendor tool (ISE in Xilinx and Quartus II in Altera).

- **How to use the current GBT-FPGA core for an FPGA which is not on the list.**

Most of the HDL files are vendor agnostic so they can be used as is in FPGA devices with enough resources. These files can be found in the folder:

"… \gbt_bank\core_sources\"

The rest of the files, that are vendor specific, must be regenerated for the targeted FPGA. These files can be found in the folder:

"*… \gbt_bank\<vendor>_<device>*" (e.g. "*… \gbt_bank\xilinx_v6*")

- ***How to properly reset the GBT link?***

The GBT Link has to be reset sequentially.

In the **Tx side**, the GBT Tx has to be reset first and then the MGT Tx.

In the **Rx side**, the MGT Rx has to be reset first and then the GBT Rx.

- **What is the best suited FPGA on the GBT link point of view?**

There is not a best suited FPGA on the GBT Link point of view.

When using the **Standard version**, any high-end FPGA (e.g. Virtex 6, Stratix V GX) is a good candidate for hosting a GBT Link-based system.

When using the **Latency-Optimized version**, the differences between the Multi-Gigabit Transceivers (MGT) and the clocking resources of the different vendors may lead to a preferred candidate for certain GBT Link based applications.

Example 1: Xilinx FPGAs featuring GTX transceivers do not need any monitoring in the Tx side whilst Altera FPGAs featuring GT transceivers need extra logic for monitoring the TX_WORDCLK.

Example 2: The dynamic phase alignment of Altera fPLL allows independent phase values per output clock whilst the dynamic phase alignment of Xilinx MMCMs applies the same phase value to all clock outputs that are shifted.

- **What are the particularities of the Virtex6 for the Latency-Optimized version?**

See 4.1.1.a.

- **What are the particularities of the Kintex 7 for the Latency-Optimized version?**

The same GBT-FPGA core is shared by Kintex 7 and Virtex 7 so the particularities may be applied to both families of FPGAs.

See 4.1.1.b.

- **What are the particularities of the Virtex7 for the Latency-Optimized version?**

The same GBT-FPGA core is shared by Kintex 7 and Virtex 7 so the particularities may be applied to both families of FPGAs.

See 4.1.1.b.

- **What are the particularities of the Cyclone V GT for the Latency-Optimized version?**

See 4.1.2.a.

- **What are the particularities of the Stratix V GX for the Latency-Optimized version?**

See 4.1.2.b.

- **Which FPGA will be targeted in the future?**

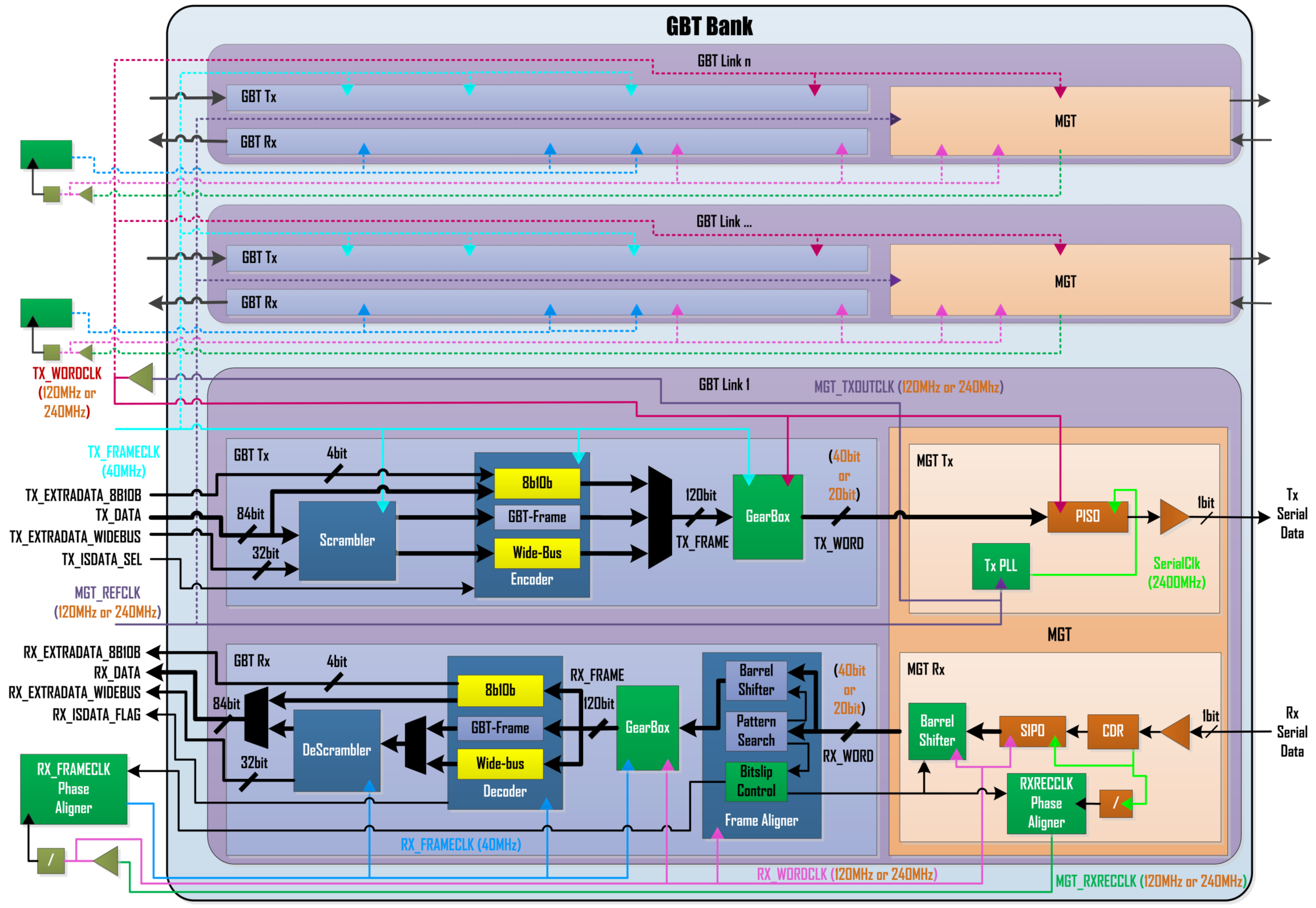
For obvious reasons, the GBT-FPGA core will not offer firmware versions for each FPGA type on the market. It targets the main vendors and the main series. The design effort on new series is foreseen to stop during 2014. The evolution of the core to follow-up the technology will thus depend on users' contributions. If you aim to port the GBT-FPGA core to an FPGA or a reference design which is not yet in the list of supported devices, and you are willing to share your work, you are very much welcome to contact the GBT-FPGA support team (GBT-FPGA-support@cern.ch): they will try to integrate it to further releases.

- **What type of support can I expect from the GBT-FPGA team?**

For obvious reasons, the GBT-FPGA team will not infinitely develop new firmware versions for each FPGA type on the market. The design effort (from the GBT-FPGA team) on new series is foreseen to stop during 2014, but the support will remain active on the official release available on the SVN, provided that the GBT-FPGA core itself has not been modified by the user.

The evolution of the core to follow-up the technology will depend on users' contributions. If you aim to port the GBT-FPGA core to an FPGA or a reference design which is not yet in the list of supported devices, and you are willing to share your work, you are very much welcome to contact the GBT-FPGA support team (GBT-FPGA-support@cern.ch): they will try to integrate it to further releases.

APPENDIX B: GBT Bank Block Diagram



GBT Bank
GBT Link n
GBT Tx
GBT Rx
MGT
GBT Link ...
GBT Tx
GBT Rx
MGT
TX_WORDCLK (120MHz or 240MHz)
GBT Link 1
MGT_TXOUTCLK (120MHz or 240Mhz)
TX_FRAMECLK (40MHz)
GBT Tx
4bit
84bit
32bit
Scrambler
8b10b
GBT-Frame
Wide-Bus
Encoder
120bit
TX_FRAME
GearBox
(40bit or 20bit)
TX_WORD
MGT Tx
PISO
Tx PLL
SerialClk (2400MHz)
1bit
Tx Serial Data
TX_EXTRADATA_8B10B
TX_DATA
TX_EXTRADATA_WIDEBUS
TX_ISDATA_SEL
MGT_REFCLK (120MHz or 240MHz)
MGT
RX_EXTRADATA_8B10B
RX_DATA
RX_EXTRADATA_WIDEBUS
RX_ISDATA_FLAG
GBT Rx
4bit
84bit
32bit
DeScrambler
8b10b
GBT-Frame
Wide-bus
Decoder
RX_FRAME
120bit
GearBox
Barrel Shifter
Pattern Search
Bitslip Control
Frame Aligner
(40bit or 20bit)
RX_WORD
MGT Rx
Barrel Shifter
SIPO
CDR
1bit
Rx Serial Data
RXRECCLK Phase Aligner
/
RX_FRAMECLK Phase Aligner
/
RX_FRAMECLK (40MHz)
RX_WORDCLK (120MHz or 240MHz)
MGT_RXRECCLK (120MHz or 240MHz)

APPENDIX C: Known Issues