EDMS Document Number
XXXXXXX

HPTD project URL
https://espace.cern.ch/HighPrecisionTiming

Date: 6 November 2019
Revision No. 1.0

Reference Note

Tx Phase Aligner for Xilinx transceivers

Abstract

This short reference note describes the usage of the Tx phase aligner core for Xilinx FPGA's. The technique behind the core was proposed by the Xilinx engineer Paolo Novellini [1] and the implementation here was made by the CERN HPTD team. An overview of the core and its typical applications (high determinism in phase over transceiver start-up, mesochronous clock domain crossing, using the rxrecclk for a cascaded timing distribution link) is given. Together with this core, are provided: an example design (GTH Kintex Ultrascale) containing a basic functional simulation (based on the Xilinx transceiver example design) and a hardware design for the KCU105 evaluation board.

This core is protocol-independent and can be used **in addition** to traditional IP cores as the GBT-FPGA, TTC-PON or the LpGBT-FPGA.

Prepared by	*Checked by*	*Approved by*
E. B. S. Mendes CERN/EP-ESE 1211 Geneva 23 Switzerland *eduardo.brandao.de.souza.mendes@cern.ch*	**S. Baron** CERN/EP-ESE 1211 Geneva 23 Switzerland *sophie.baron@cern.ch*	**-**

Document History

Rev. No	*Date*	*Pages*	*Description of Changes*
0.1	26 June 2018	*All*	First version
1.0	6 November 2019	*All*	Add fine phase-shifting capability

Contents

1 Technique overview

When the elastic buffer is enabled in a Xilinx FPGA transmitter, the phase between the clock used for the transmission (serializer clock) and the reference clock exhibits jumps after start-up (i.e. resets) with UI steps corresponding to dividers present in the PMA (physical medium attachment). This problem can be overcome with a technique proposed by the Xilinx engineer Paolo Novellini [1]. The technique employed here relies on two advanced blocks present in the transceiver.

- **Tx FIFO fill level flag (txbufstatus[0]):** A flag which indicates whether the Tx FIFO is more than half-full
- **Tx Phase Interpolator controller:** A very fine phase shifter able to shift the transmitter clock with a resolution in the order of *ps*

See your transceiver documentation [2] [3] [4] for more information about those blocks.

A simplified block diagram of the transceiver (focusing on the transmitter elastic buffer) is shown in figure 1. The technique presented in this paper consists in measuring the phase between the TXUSRCLK2 (FIFO write pointer) and the XCLK (FIFO read pointer, parallel clock of serializer) by means of the Tx FIFO fill level flag and performing an alignment (shifting the TXREFCLK with the Tx Phase Interpolator controller) on the level for which the FIFO is half-full. More details on algorithm and alignment flavours are provided in the section 4.

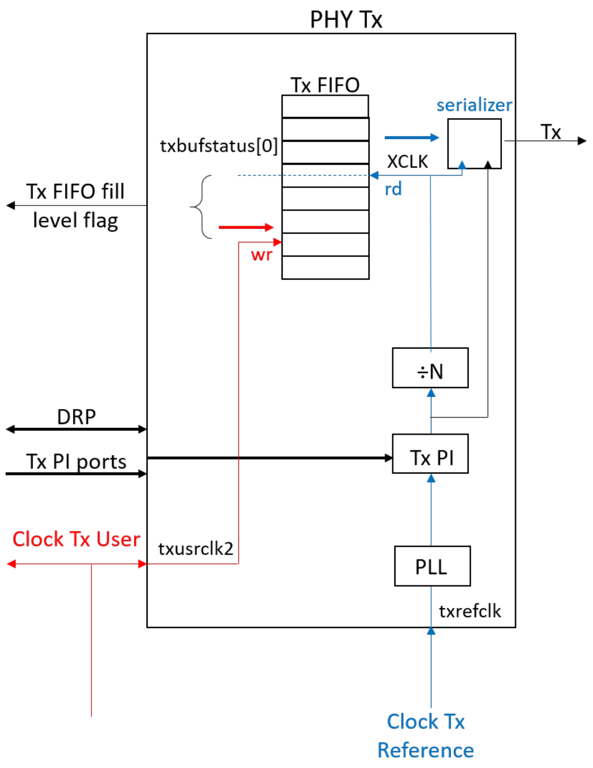


Figure 1: Simplified block diagram of Xilinx transmitter Multi-Gigabit-Transceiver (focusing on elastic buffer)

The proposed technique is able to provide us with a phase stability (between the TXREFCLK and Tx data) after resets better than when using the buffer-bypass technique (common technique for minimal latency variation for Xilinx transceivers in TTC-PON and GBT-FPGA projects). This was demonstrated in [5] and [6].

Another interesting aspect of this core is that the Tx phase interpolator can be used when using this block, allowing one to set a skew between links for a multi-lane implementation. If the buffer-bypass is enabled, the Tx phase interpolator cannot be controlled.

This solution can be, in principle, implemented for the following transceivers:

- **GTH 7 series:** not tested.
- **GTH Ultrascale:** tested with the Kintex Ultrascale in the evaluation board KCU105. Provided here as an example design.
- **GTY Ultrascale:** not tested.
- **GTH Ultrascale+:** tested with the Zynq Ultrascale+ in the evaluation board ZCU102.
- **GTY Ultrascale+:** tested with the Kintex Ultrascale+ in the evaluation board KCU116.

The core implementing this technique and presented in section 4 is provided together with a simulation targetting a GTH Ultrascale device and an example is offered for the KCU105 board. **Users are welcome to ask questions, provide feedback and share results with the HPTD team.**

It is important to note that this technique relies on the performance of features which are not fully specified by Xilinx. Values presented here are based on a measurement carried out by the HPTD team but shall not be seen as a specification.

2 High phase determinism after transceiver start-up

In the LHC experiments, radiation-hard ASICs (TTCrx, GBTx, LpGBT) are developed to receive timing, trigger and control information in the front-end from the back-end via high-speed serial links. Nowadays, the back-end transmitters are mainly based on FPGAs and therefore a typical link is shown in figure 2.

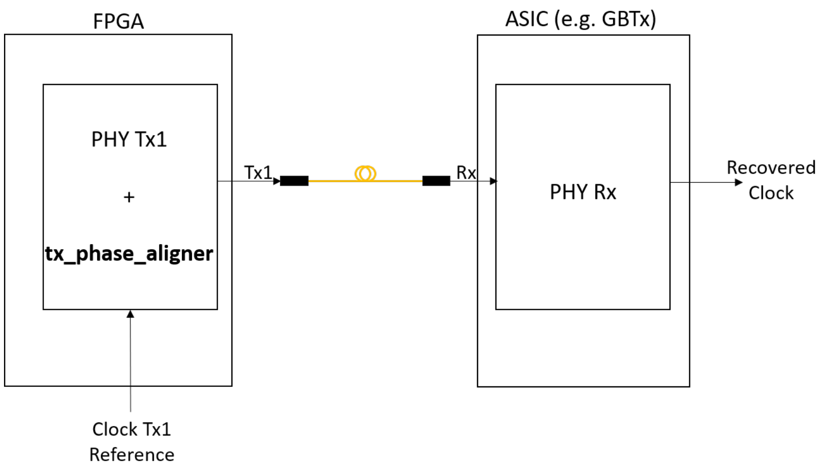


Figure 2: Single optical link from back-end board containing FPGA to front-end ASIC

A requirement for those links is a high phase determinism after system start-ups. The core presented here can be used in a back-end board containing a Xilinx FPGA to ensure this requirement. In order to cascade two FPGA-based links, some ideas are given in section 3.

3 Cascaded systems: using the RXRECCLK and mesochronous clock domain crossing (CDC)

The Ultrascale architecture possesses of an innovative functionality allowing one to output the recovered clock directly from the transceiver block using a reference clock pin without going through the fabric. The advantages of such a clock (when compared with a clock which goes through the fabric - e.g. RXUSRCLK2) are higher level of repeatability over implementations (a priori, its quality does not depend on the FPGA implementation) and a higher quality (not impacted by switching noise occurring in the fabric). On the other hand, the usage of this clock poses some challenges as it cannot be directly processed in the fabric.

After a transceiver start-up, the phase between the RXRECCLK and the RXUSRCLK2 (clock for which the GBT-FPGA, TTC-PON projects ensure minimal latency variation) can exhibit phase jumps with UI steps [6]. Therefore, if the RXRECCLK is cleaned by a PLL and used as a reference clock for a transmitter (with buffer-bypass) in a cascaded chain, the output of the transmitter will exhbit UI jumps with respect to the RXUSRCLK2. Therefore, a common technique is to output the RXUSRCLK2 as depicted in figure 3.
On the other hand, if the core here proposed is implemented and the TXUSRCLK2 of the cascaded transmitter uses the RXUSRCLK2 as depicted in figure 4 the UI jumps will be corrected.
Another challenge of fixed-phase cascaded systems is the proper clock domain crossing between RXUSRCLK2 and TXUSRCLK2 in a design implemented as depicted in figure 3. This problem is overcome by using the **tx_phase_aligner** core here proposed. Obs: RXRECCLK is only available for Ultrascale devices and therefore this is not applicable for 7-series devices.
Compared to a traditional cascading, the technique shown here has the following trade-off (advantages: green and disadvantages: red):

- Clock does not go through fabric

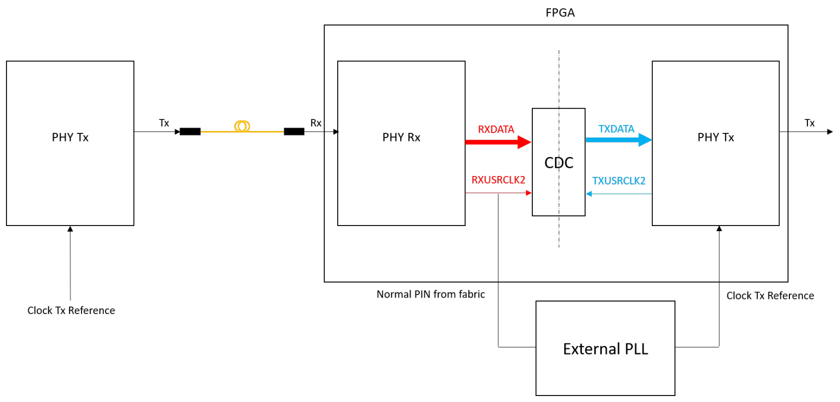


Figure 3: Traditional cascading (buffer-bypass, fixed-latency ensured by phase alignment of external PLL)

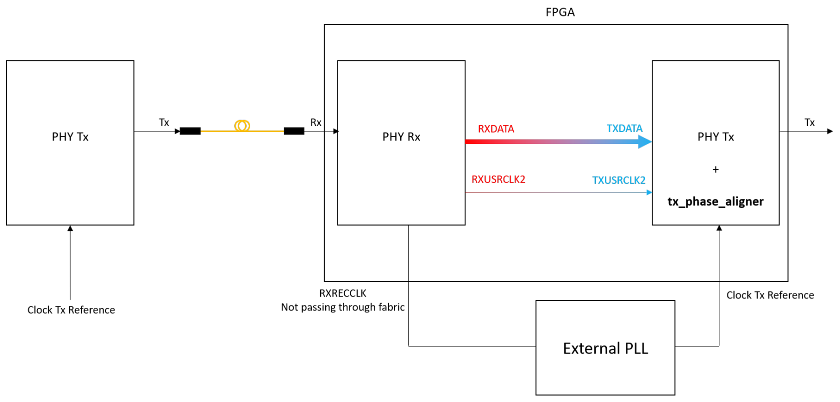


Figure 4: tx_phase_aligner cascading (elastic buffer used, RXRECCLK used as Tx reference clock)

- Clock domain crossing simplified
- Phase-determinism enhancement
- RXRECCLK is not fixed-phase as the correction is made at the Tx PI level

4 Core Overview

Architecture

The **tx_phase_aligner** architecture is depicted in figure 5. It is composed of three main blocks which have the following functionality:

- **fifo_fill_level_acc:** Phase measurement based on the Tx FIFO fill level flag (late/early flag) coming from the FPGA transceiver. The flag is averaged in order to provide us with a high resolution phase detection
- **tx_pi_ctrl:** Controller of transmitter phase based on the phase-interpolator
- **tx_phase_aligner_fsm:** Algorithm implemented as a finite state machine responsible for aligning the transmitter phase in the $0 \to 1$ transition of the Tx FIFO fill level flag

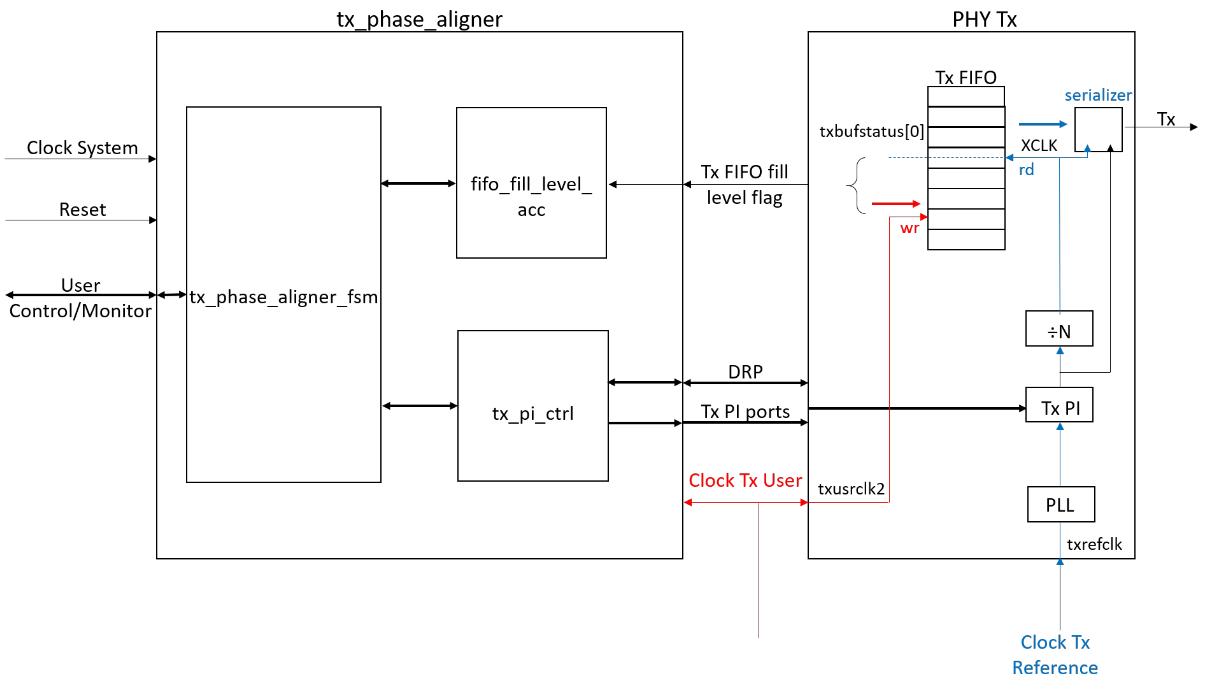


Figure 5: Block diagram tx_phase_aligner

A simplified overview of the state machine implemented by the **tx_phase_aligner_fsm** is shown in figure 6. In order to speed up the alignment process, a first coarse alignment is performed using a low resolution phase detection (the accumulator for the **fifo_fill_level_acc** is set to a low value) and coarse phase shifting (the steps of the **tx_pi_ctrl** block is set to a high value). This is followed by a fine alignment with a high resolution phase detection and fine step phase shifting.

The algorithm implemented can have two different flavours (FINE_ALIGNMENT and UI_ALIGNMENT) depending on the implementation requirements. The overview of the those flavours is given below.

- **1) FINE_ALIGNMENT:** At each reset, re-align transmitter with fine PI step

  $\rightarrow$Recommended for?

  applications not requiring a lower phase determinism (5-10 ps variation - based on our measurements) after resets

  applications using this block only as a CDC strategy with minimal latency variation

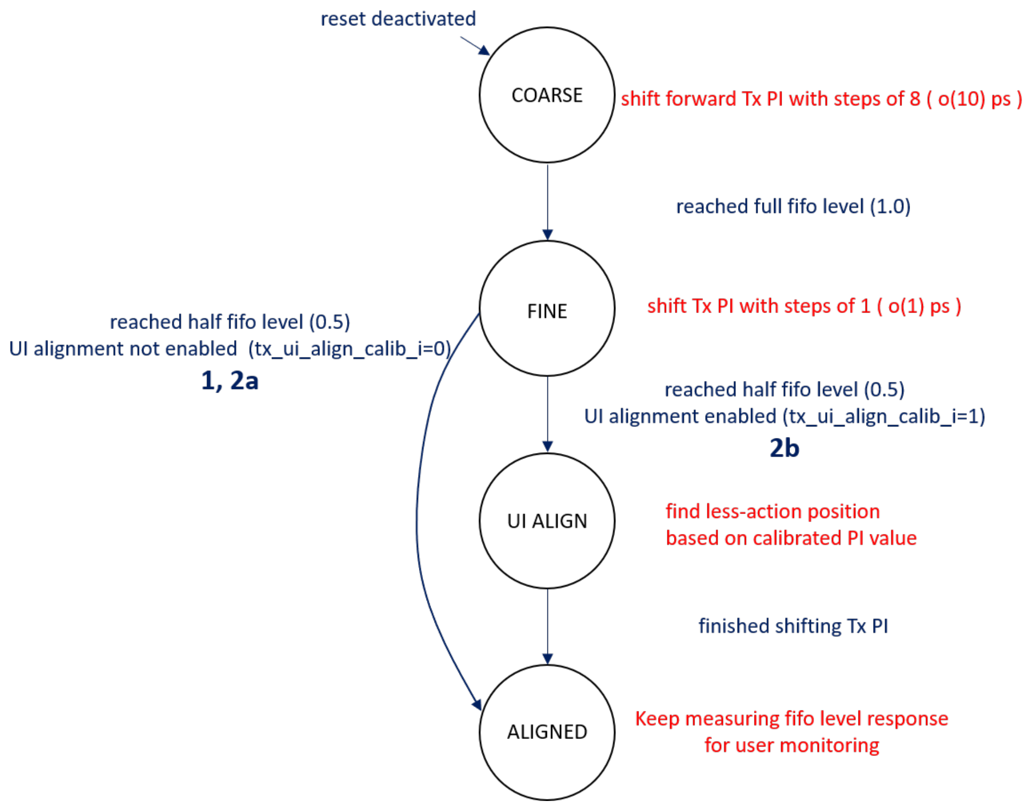

reset deactivated
COARSE
shift forward Tx PI with steps of 8 (o(10) ps)
reached full fifo level (1.0)
FINE
shift Tx PI with steps of 1 (o(1) ps)
reached half fifo level (0.5)
UI alignment not enabled (tx_ui_align_calib_i=0)
1, 2a
reached half fifo level (0.5)
UI alignment enabled (tx_ui_align_calib_i=1)
2b
UI ALIGN
find less-action position
based on calibrated PI value
finished shifting Tx PI
ALIGNED
Keep measuring fifo level response
for user monitoring

Figure 6: Algorithm FSM simplified

→ How user should connect ports (see table 1)?

tx_pi_phase_calib_i to all 0

tx_ui_align_calib_i to 0

- **2) UI_ALIGNMENT:** After an initial calibration, re-align the transmitter PI to the calibrated value

→Recommended for?

applications requiring a higher phase determinism (2-5 ps variation - based on our measurements) after resets

applications where the board FPGA is not subject to large temperature variations

→ What does it cost?

requires an initial calibration (automatically done by the core) during first reset

→ How user should connect ports (see table 1)?

a) during first reset (calibration):

tx_pi_phase_calib_i to all X (dont care)

tx_ui_align_calib_i to 0

b) during other resets (after calibration):

tx_pi_phase_calib_i to the value of **tx_pi_phase_o** after the first reset

tx_ui_align_calib_i to 1

*It is important to note that the **UI_ALIGNMENT** mode is based on a maximum likelihood hypothesis that the transceiver phase varies less than $\frac{UI}{2\times TXOUTDIV}$. Major phase variations can be monitored through the **tx_fifo_fill_pd_o** status port.*

*When using the mode **UI_ALIGNMENT**, it is recommended to monitor the port **tx_fifo_fill_pd_o** and perform a fine re-alignment in case this port gives all zeros or all ones.*

Ports / Attributes description

The ports and attributes of the **tx_phase_aligner** core are described in tables 1 and 2.

*Please note that it is recommended to keep the **tx_phase_aligner** core in a reset state while the transceiver reset procedure is not completed and to use the port **tx_aligned_o** as a reset for the transmitter user logic.*

Port	Direction	Clk Domain	Description	Functionality
clk_sys_i	in	—	System clock Connect to a free-running source	general
reset_i	in	clk_sys_i	Core synchronous reset Keep reset while MGT tx not ready	general
tx_aligned_o	out	clk_sys_i	Transmitter alignment procedure finished Use as reset for transmitter user logic	status
tx_pi_phase_calib_i[6::0]	in	clk_sys_i	UI alignment Tx PI calibrated phase	control
tx_ui_align_calib_i	in	clk_sys_i	Active high UI alignment Tx PI activate	control
tx_fifo_fill_pd_max_i[31::0]	in	clk_sys_i	Phase detector maximum accumulated value Recommended minimum 0x00040000	control
tx_fine_realign_i	in	clk_sys_i	Active on rising edge Repeats fine alignment procedure	control
ps_strobe_i	in	clk_sys_i	Active on rising edge Shifts phase of transmitter serial data (only use this signal once tx_aligned_o is one)	control
ps_inc_ndec_i	in	clk_sys_i	Increment or decrement phase	control
ps_phase_step_i[3::0]	in	clk_sys_i	Phase step in PI units	control
ps_done_o	out	clk_sys_i	Active on rising edge Phase shift is done	control
tx_pi_phase_o[6::0]	out	clk_sys_i	Tx PI phase after alignment	status
tx_fifo_fill_pd_o[31::0]	out	clk_sys_i	Phase detector current value Should be around half of **tx_fifo_fill_pd_max_i[31::0]** after alignment	status
clk_txusr_i	in	—	Tx user clock Connect to txusrclk2	transceiver
tx_fifo_fill_level_i	in	clk_txusr_i	Tx FIFO fill level flag Connect to txbufstatus[0]	transceiver
txpippmen_o	out	clk_txusr_i	Tx PI control See transceiver user guide for more information	transceiver
txpippmovrden_o	out	clk_txusr_i	same as above	transceiver
txpippmsel_o	out	clk_txusr_i	same as above	transceiver
txpippmstepsize_o[4::0]	out	clk_txusr_i	same as above	transceiver
txpippmen_o	out	clk_txusr_i	same as above	transceiver
drpaddr_o[8::0]	out	clk_sys_i	same as above	transceiver
drpen_o	out	clk_sys_i	same as above	transceiver
drpdi_o[15::0]	out	clk_sys_i	same as above	transceiver
drprdy_i	in	clk_sys_i	same as above	transceiver
drpdo_i[15::0]	in	clk_sys_i	same as above	transceiver
drpwe_o	out	clk_sys_i	same as above	transceiver

Table 1: **tx_phase_aligner** core ports description

Attribute	Type	Description	Functionality
g_DRP_NPORT_CONTROL	boolean	Tx PI control via PORT or DRP Port control was not yet tested so it is recommended to use DRP control	transceiver
g_DRP_ADDR_TXPI_PPM_CFG	std_logic_vector[8::0]	TXPI_PPM_CFG address See transceiver user guide for more information	transceiver

Table 2: **tx_phase_aligner** core attributes description

5 Reference design

Overview

A reference design available on GIT (*https://gitlab.cern.ch/HPTD/tx_phase_aligner*) targetting Ultrascale GTH transceivers. The reference design is based on the Xilinx example design for Ultrascale transceivers generated through the wizard [7]. Sections of the example design which were modified are commented with the *EBSM* signature to help the interested user in comparing the modifications made to the original example design code. A word aligner is also included in order to align the receiver and allow users testing the core in loopback.
The transceiver is generated with a data-rate of 10.24Gb/s (typical LpGBT-FPGA data-rate).

*The objective of the example design is to help users in understanding how to connect the **tx_phase_aligner** core to their design. The files provided in the example design are not fully supported by the HPTD team. Support is only given to the **tx_phase_aligner** core.*

A post-synthesis simulation is available, further discussed in the section 6. The simulation is done at a post-synthesis level in order to take into account the RX_SLIDE feature in PMA mode modified through the constraints, it has no correlation with the **tx_phase_aligner** core here presented. The example design targets the KCU105 evaluation board.

Resource usage

Typical resource usage of the **tx_phase_aligner** core is shown in the table below.

CLB LUTs	CLB Registers	CARRY8	F7 Muxes
135	172	15	3

Table 3: **tx_phase_aligner** core typical resource usage for Ultrascale implementation

*The **tx_phase_aligner** core incurs a minor latency increase of the transceiver. Check your transceiver Tx latency values for an estimation. It should be noted though that in a mesochronous cascaded mode this increase might be compensated by the latency decrease of not implementing the CDC in the fabric.*

Getting started and tools

The example design and core was tested using **Vivado 2016.2**. After loading the core from GIT, run the script **run_script_tcl.bat** which will automatically re-create the project and launch a full compilation. Once the compilation is finished, a simulation will be launched.

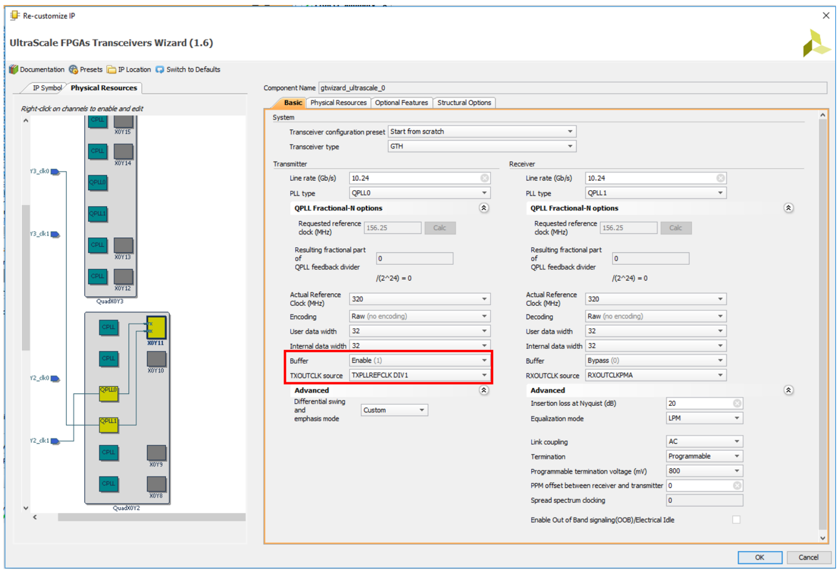
The hierarchy of files available on GIT is shown below:

run_script_tcl.bat (launch tx_phase_aligner_proj.tcl - assumes Vivado default installation path)
tx_aligner_proj.tcl (re-creates project)
- scripts/
– sim/
— tx_phase_aligner_simu.tcl (executes simulation and force VIO values)
— tx_phase_aligner_simu.wcfg (waveform for simulation example design)
- source/
– constrs/
— imports/
—- example_design/
—– gtwizard_ultrascale_0_example_top.xdc (timing and physical constraints based on example design)
– sim/
— imports/

—- example_design/
—– gtwizard_ultrascale_0_example_top_sim.v (top test-bench based on example design)
– synth/
— imports/
—- design_tx_aligner/
—– tx_phase_aligner.vhd (**tx_phase_aligner** core top-level)
—– fifo_fill_level_acc.vhd (**tx_phase_aligner** core)
—– tx_phase_aligner_fsm.vhd (**tx_phase_aligner** core)
—– tx_pi_ctrl.vhd (**tx_phase_aligner** core)
—- example_design/
—– gtwizard_ultrascale_0_example_top.v (top wrapper based on example design)
—– gtwizard_ultrascale_0_example_bit_synchronizer.v (see transceiver example design)
—– gtwizard_ultrascale_0_example_checking_raw.v (see transceiver example design)
—– gtwizard_ultrascale_0_example_init.v (see transceiver example design)
—– gtwizard_ultrascale_0_example_reset_synchronizer.v (see transceiver example design)
—– gtwizard_ultrascale_0_example_stimulus_raw.v (see transceiver example design)
—– gtwizard_ultrascale_0_example_wrapper.v (see transceiver example design)
—– gtwizard_ultrascale_0_example_wrapper_functions.vh (see transceiver example design)
—– gtwizard_ultrascale_0_prbs_any.v (see transceiver example design)
—– rx_word_aligner.vhd (word aligner implemented by HPTD for example design)
— ip/
—- gtwizard_ultrascale_0.xcix (transceiver ip core)
—- gtwizard_ultrascale_0_vio_0.xcix (VIO ip core based on example design)

Transceiver configuration

The configuration of the transceiver in the example design for the **tx_phase_aligner** core is depicted in figures 7 and 8.

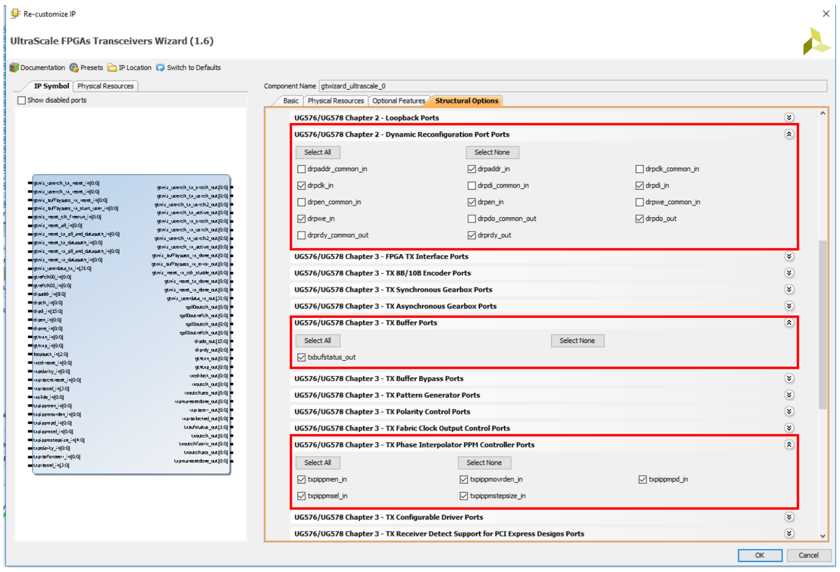


Component Name gtwizard_ultrascale_0

System
Transceiver configuration preset: Start from scratch
Transceiver type: GTH

Transmitter		Receiver	
Line rate (Gb/s)	10.24	Line rate (Gb/s)	10.24
PLL type	QPLL0	PLL type	QPLL1
QPLL Fractional-N options		**QPLL Fractional-N options**	
Requested reference clock (MHz)	156.25	Requested reference clock (MHz)	156.25
Resulting fractional part of QPLL feedback divider	0 /(2^24) = 0	Resulting fractional part of QPLL feedback divider	0 /(2^24) = 0
Actual Reference Clock (MHz)	320	Actual Reference Clock (MHz)	320
Encoding	Raw (no encoding)	Decoding	Raw (no encoding)
User data width	32	User data width	32
Internal data width	32	Internal data width	32
Buffer	Enable (1)	Buffer	Bypass (0)
TXOUTCLK source	TXPLLREFCLK DIV1	RXOUTCLK source	RXOUTCLKPMA
Advanced		**Advanced**	
Differential swing and emphasis mode	Custom	Insertion loss at Nyquist (dB)	20
		Equalization mode	LPM
		Link coupling	AC
		Termination	Programmable
		Programmable termination voltage (mV)	800
		PPM offset between receiver and transmitter	0
		Spread spectrum clocking	0
		Enable Out of Band signaling(OOB)/Electrical Idle	☐

Figure 7: transceiver wizard configuration (1)



Component Name gtwizard_ultrascale_0

UG576/UG578 Chapter 2 - Loopback Ports

UG576/UG578 Chapter 2 - Dynamic Reconfiguration Port Ports
Select All | Select None

☐ drpaddr_common_in	✓ drpaddr_in	☐ drpclk_common_in
✓ drpclk_in	☐ drpdi_common_in	✓ drpdi_in
☐ drpen_common_in	✓ drpen_in	☐ drpwe_common_in
✓ drpwe_in	☐ drpdo_common_out	✓ drpdo_out
☐ drprdy_common_out	✓ drprdy_out	

UG576/UG578 Chapter 3 - FPGA TX Interface Ports
UG576/UG578 Chapter 3 - TX 8B/10B Encoder Ports
UG576/UG578 Chapter 3 - TX Synchronous Gearbox Ports
UG576/UG578 Chapter 3 - TX Asynchronous Gearbox Ports

UG576/UG578 Chapter 3 - TX Buffer Ports
Select All | Select None
✓ txbufstatus_out

UG576/UG578 Chapter 3 - TX Buffer Bypass Ports
UG576/UG578 Chapter 3 - TX Pattern Generator Ports
UG576/UG578 Chapter 3 - TX Polarity Control Ports
UG576/UG578 Chapter 3 - TX Fabric Clock Output Control Ports

UG576/UG578 Chapter 3 - TX Phase Interpolator PPM Controller Ports
Select All | Select None

✓ txpippmen_in	✓ txpippmovrden_in	✓ txpippmpd_in
✓ txpippmsel_in	✓ txpippmstepsize_in	

UG576/UG578 Chapter 3 - TX Configurable Driver Ports
UG576/UG578 Chapter 3 - TX Receiver Detect Support for PCI Express Designs Ports

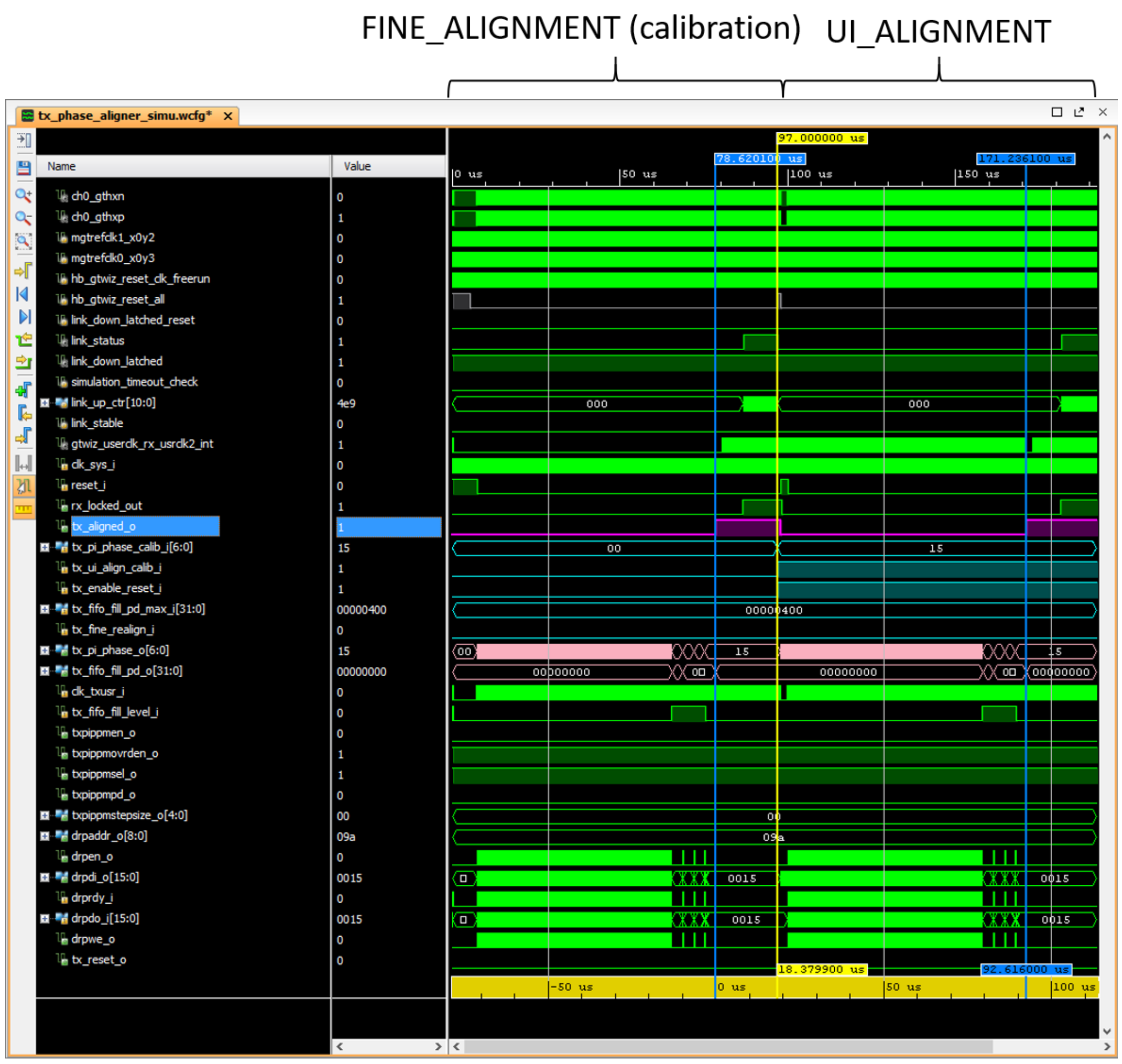
Figure 8: transceiver wizard configuration (2)

It should be noted that for a cascaded configuration (c.f. in the section 3), the user should connect the TXUSRCLK and TXUSRCLK2 to the RXOUTCLK from another receiver channel through a BUFG_GT.

6 Simulation test-bench

The simulation test-bench is controlled by a TCL script (**tx_phase_aligner_simu.tcl**). The typical waveform after the simulation is finished is shown in the figure 9. The simulation is divided into two steps:

- **FINE_ALIGNMENT:** A first fine alignment is performed and the user can see the core tx aligned (marker at 78us).
- **UI_ALIGNMENT:** After the first alignment, the Tx PI position aligned is used as a calibration value (marker at 97us) and a reset is performed. The simulation finishes when the second alignment is done (marker at 171us).



Name	Value
ch0_gthxn	0
ch0_gthxp	1
mgtrefclk1_x0y2	0
mgtrefclk0_x0y3	0
hb_gtwiz_reset_clk_freerun	0
hb_gtwiz_reset_all	1
link_down_latched_reset	0
link_status	1
link_down_latched	1
simulation_timeout_check	0
link_up_ctr[10:0]	4e9
link_stable	0
gtwiz_userclk_rx_usrclk2_int	1
clk_sys_i	0
reset_i	0
rx_locked_out	1
tx_aligned_o	1
tx_pi_phase_calib_i[6:0]	15
tx_ui_align_calib_i	1
tx_enable_reset_i	1
tx_fifo_fill_pd_max_i[31:0]	00000400
tx_fine_realign_i	0
tx_pi_phase_o[6:0]	15
tx_fifo_fill_pd_o[31:0]	00000000
clk_txusr_i	0
tx_fifo_fill_level_i	0
txpippmen_o	0
txpippmovrden_o	1
txpippmsel_o	1
txpippmpd_o	0
txpippmstepsize_o[4:0]	00
drpaddr_o[8:0]	09a
drpen_o	0
drpdi_o[15:0]	0015
drprdy_i	0
drpdo_i[15:0]	0015
drpwe_o	0
tx_reset_o	0

Figure 9: tx_phase_aligner simulation overview

*Please note that the behaviour of the txbufstatus[0] flag behaviour has some limitations in simulation as jitter is not included. Also, the value of the port **tx_fifo_fill_pd_max_i** was set to 0x00000400 to speed-up the simulation, it is recommended to keep this port at least to 0x00040000.*

7 Hardware test-bench on the KCU105 board

In order to bring-up the hardware design, the following actions are needed:

- **Si570 on-board programming:** Receiver reference clock, program to 320MHz through board system controller.
- **FMC_LPC_GBTCLK0:** Transmitter reference clock, connect to a 320MHz generator. This design does not share the tx and rx reference clock for test purposes, it shall not be seen as a limitation of the **tx_phase_aligner** core.
- **Tx-Rx:** Loopback Tx $\rightarrow$ Rx SMA MGT pins for full link implementation. Connect one of the Tx pairs to a scope to measure the Reference clock $\rightarrow$ Tx phase consistency.

Other connections are also available for the interested user. The clock RXRECCLK is connected to the SMA REF CLK pins and the clock RXUSRCLK2 is connected to the USER_SMA_GPIO_P.

An example of connection is shown in the figures 10 and 11.

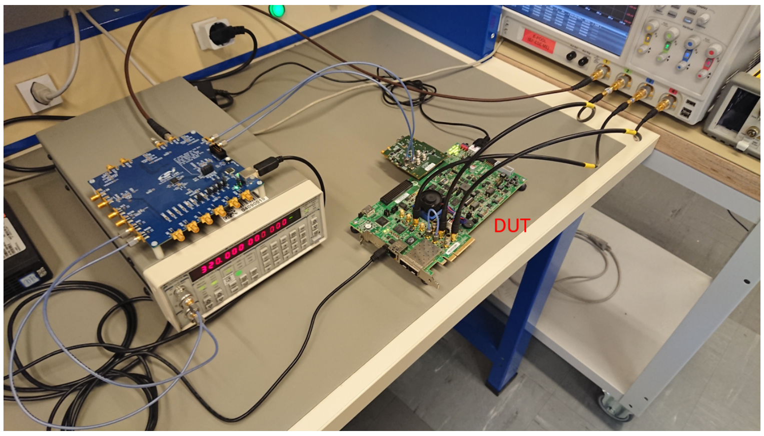


Figure 10: Hardware-test bench overview

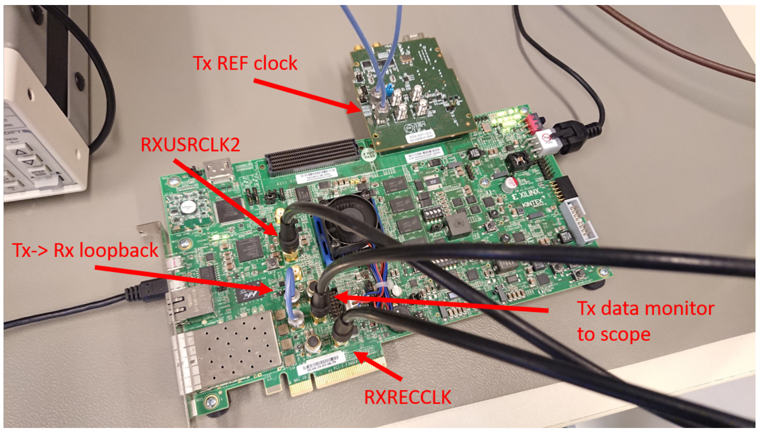
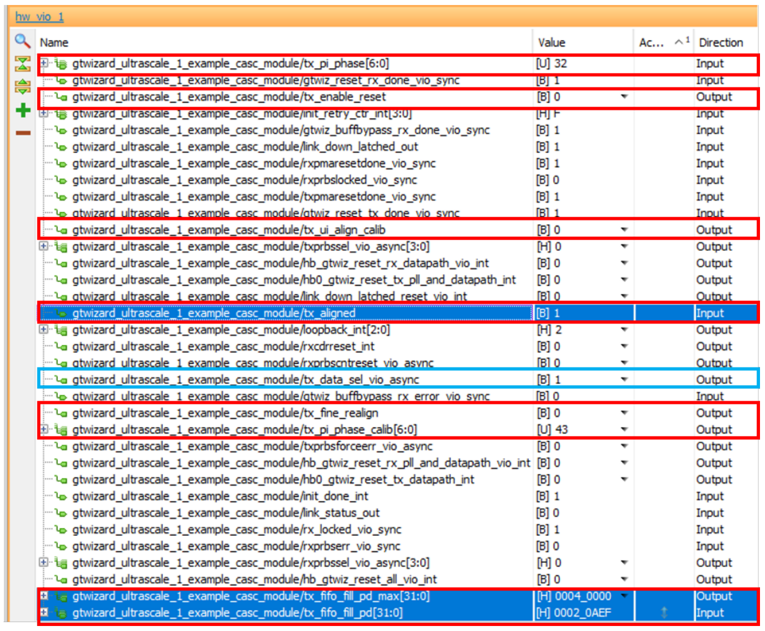


Figure 11: Hardware-test bench specific connections

The hardware test-bench is controlled by a VIO. The different control and status are shown in the figure 12. The red boxes show the **tx_phase_aligner** control and status. The user can modify the transmitted data pattern through the **tx_data_sel_vio_async** bit (0=PRBS-31 + 2b word header; 1=320MHz clock pattern).



hw_vio_1

Name	Value	Ac...	Direction
gtwizard_ultrascale_1_example_casc_module/tx_pi_phase[6:0]	[U] 32		Input
gtwizard_ultrascale_1_example_casc_module/gtwiz_reset_rx_done_vio_sync	[B] 1		Input
gtwizard_ultrascale_1_example_casc_module/tx_enable_reset	[B] 0		Output
gtwizard_ultrascale_1_example_casc_module/init_retry_ctr_int[3:0]	[H] F		Input
gtwizard_ultrascale_1_example_casc_module/gtwiz_buffbypass_rx_done_vio_sync	[B] 1		Input
gtwizard_ultrascale_1_example_casc_module/link_down_latched_out	[B] 1		Input
gtwizard_ultrascale_1_example_casc_module/rxpmaresetdone_vio_sync	[B] 1		Input
gtwizard_ultrascale_1_example_casc_module/rxprbslocked_vio_sync	[B] 0		Input
gtwizard_ultrascale_1_example_casc_module/txpmaresetdone_vio_sync	[B] 1		Input
gtwizard_ultrascale_1_example_casc_module/gtwiz_reset_tx_done_vio_sync	[B] 1		Input
gtwizard_ultrascale_1_example_casc_module/tx_ui_align_calib	[B] 0		Output
gtwizard_ultrascale_1_example_casc_module/txprbssel_vio_async[3:0]	[H] 0		Output
gtwizard_ultrascale_1_example_casc_module/hb_gtwiz_reset_rx_datapath_vio_int	[B] 0		Output
gtwizard_ultrascale_1_example_casc_module/hb0_gtwiz_reset_tx_pll_and_datapath_int	[B] 0		Output
gtwizard_ultrascale_1_example_casc_module/link_down_latched_reset_vio_int	[B] 0		Output
gtwizard_ultrascale_1_example_casc_module/tx_aligned	[B] 1		Input
gtwizard_ultrascale_1_example_casc_module/loopback_int[2:0]	[H] 2		Output
gtwizard_ultrascale_1_example_casc_module/rxcdrreset_int	[B] 0		Output
gtwizard_ultrascale_1_example_casc_module/rxprbscntreset_vio_async	[B] 0		Output
gtwizard_ultrascale_1_example_casc_module/tx_data_sel_vio_async	[B] 1		Output
gtwizard_ultrascale_1_example_casc_module/gtwiz_buffbypass_rx_error_vio_sync	[B] 0		Input
gtwizard_ultrascale_1_example_casc_module/tx_fine_realign	[B] 0		Output
gtwizard_ultrascale_1_example_casc_module/tx_pi_phase_calib[6:0]	[U] 43		Output
gtwizard_ultrascale_1_example_casc_module/txprbsforceerr_vio_async	[B] 0		Output
gtwizard_ultrascale_1_example_casc_module/hb_gtwiz_reset_rx_pll_and_datapath_vio_int	[B] 0		Output
gtwizard_ultrascale_1_example_casc_module/hb0_gtwiz_reset_tx_datapath_int	[B] 0		Output
gtwizard_ultrascale_1_example_casc_module/init_done_int	[B] 1		Input
gtwizard_ultrascale_1_example_casc_module/link_status_out	[B] 0		Input
gtwizard_ultrascale_1_example_casc_module/rx_locked_vio_sync	[B] 1		Input
gtwizard_ultrascale_1_example_casc_module/rxprbserr_vio_sync	[B] 0		Input
gtwizard_ultrascale_1_example_casc_module/rxprbssel_vio_async[3:0]	[H] 0		Output
gtwizard_ultrascale_1_example_casc_module/hb_gtwiz_reset_all_vio_int	[B] 0		Output
gtwizard_ultrascale_1_example_casc_module/tx_fifo_fill_pd_max[31:0]	[H] 0004_0000		Output
gtwizard_ultrascale_1_example_casc_module/tx_fifo_fill_pd[31:0]	[H] 0002_0AEF		Input

Figure 12: tx_phase_aligner VIO overview

References

[1] P. Novellini, The control of phase and latency in the xilinx transceivers, Xilinx **Xilinx** (2017) .

[2] Xilinx, 7 series fpgas gtx/gth transceivers, User Guide **UG476** (2016) .

[3] Xilinx, Ultrascale architecture gth transceivers, User Guide **UG576** (2017) .

[4] Xilinx, Ultrascale architecture gty transceivers, User Guide **UG578** (2017) .

[5] S. Baron and E. Mendes, Common project on precision timing: High-precision timing distribution for hl-lhc, CERN **ACES** (2018) .

[6] E. Mendes and S. Baron, Hpt ip core for high-speed links using xilinx fpgas, CERN **HPTD Working Group Meeting** (2018) .

[7] Xilinx, Ultrascale fpgas transceivers wizard v1.6, Vivado Design Suite **PG182** (2017) .