

Versal Adaptive SoC Transceiver Subsystem v1.0

Product Guide

Vivado Design Suite

PG442 (v1.0) November 13, 2024

AMD Adaptive Computing is creating an environment where employees, customers, and partners feel welcome and included. To that end, we're removing non-inclusive language from our products and related collateral. We've launched an internal initiative to remove language that could exclude people or reinforce historical biases, including terms embedded in our software and IPs. You may still find examples of non-inclusive language in our older products as we work to make these changes and align with evolving industry standards. Follow this [link](#) for more information.





Table of Contents

- Chapter 1: Introduction..... 3**
 - Features..... 3
 - IP Facts.....4
- Chapter 2: Overview.....5**
 - Navigating Content by Design Process..... 5
 - Licensing and Ordering..... 5
- Chapter 3: Product Specification..... 6**
 - Designing with the Example Design..... 10
 - Reset Controller Helper Block.....10
 - Register Space..... 17
- Chapter 4: Design Flow Steps.....18**
 - Customizing and Generating the Subsystem..... 18
- Chapter 5: Example Design..... 31**
 - Limitations of the Example Design..... 32
 - Simulating the Example Design..... 33
- Appendix A: Debugging..... 35**
 - Finding Help with AMD Adaptive Computing Solutions..... 35
 - Debug Tools..... 36
 - Hardware Debug..... 36
 - Sample Tcl File for Import..... 37
- Appendix B: Additional Resources and Legal Notices..... 39**
 - Finding Additional Documentation..... 39
 - Support Resources..... 40
 - References.....40
 - Revision History.....40
 - Please Read: Important Legal Notices..... 41

Introduction

The AMD Versal™ Adaptive SoC GTY, GTYP, and GTM Transceivers Subsystem IP solution helps configure one or more serial transceivers. You can start from scratch, input your requirements, and generate valid configurations. Also, this wizard can produce an example design for simple simulation.

Features

The following are the key features of the wizard:

- Simple and intuitive feature selection flow.
- Onetime GUI entry for multiple line rate options to enable seamless dynamic rate switching without re-programming GT registers.
- Design entry through the IP catalog and IP integrator is supported for both AMD IP and custom IP.
- Synthesizable example design with configurable pseudo-random binary sequence (PRBS) data generator, checker, and link status indicator logic to quickly demonstrate core and transceiver functionality in simulation.

IP Facts

AMD LogiCORE™ IP Facts Table	
Subsystem Specifics	
Supported Device Family	AMD Versal™ Adaptive SoC
Supported User Interfaces	N/A
Resources	N/A
Provided with Subsystem	
Design Files	RTL
Example Design	Verilog
Test Bench	Verilog
Constraints File	Xilinx Design Constraints (XDC)
Simulation Model	Source HDL with SecureIP transceiver simulation
Supported S/W Driver	N/A
Tested Design Flows	
Design Entry	AMD Vivado™ Design Suite
Simulation	For supported simulators, see the <i>Vivado Design Suite User Guide: Release Notes, Installation, and Licensing</i> (UG973).
Synthesis	Vivado Synthesis
Support	
Release Notes and Known Issues	Master Answer Record: 75716
All Vivado IP Change Logs	Master Vivado IP Change Logs: 72775
Support web page	

Notes:

1. For a complete list of supported devices, see the AMD Vivado™ IP catalog.
2. For the supported versions of the tools, see the *Vivado Design Suite User Guide: Release Notes, Installation, and Licensing* ([UG973](#)).

Overview

Navigating Content by Design Process

AMD Adaptive Computing documentation is organized around a set of standard design processes to help you find relevant content for your current development task. You can access the AMD Versal™ adaptive SoC design processes on the [Design Hubs](#) page. You can also use the [Design Flow Assistant](#) to better understand the design flows and find content that is specific to your intended design needs. This document covers the following design processes:

- **Hardware, IP, and Platform Development:** Creating the PL IP blocks for the hardware platform, creating PL kernels, functional simulation, and evaluating the AMD Vivado™ timing, resource use, and power closure. Also involves developing the hardware platform for system integration. Topics in this document that apply to this design process include:
 - [Customizing and Generating the Subsystem](#)
 - [Chapter 5: Example Design](#)
-

Licensing and Ordering

This AMD LogiCORE™ IP module is provided at no additional cost with the AMD Vivado™ Design Suite under the terms of the [End User License](#).

For more information about this subsystem, visit the Transceiver Subsystem product web page.

Information about other AMD LogiCORE™ IP modules is available at the [Intellectual Property](#) page. For information about pricing and availability of other AMD LogiCORE IP modules and tools, contact your [local sales representative](#).

Product Specification

AMD Versal Adaptive SoC Transceiver Subsystem is a wrapper created around GT*_QUAD primitives. The wizard offers a highly adaptable AMD Vivado™ IDE-driven customization process. It allows for basic customization of transceiver modes, optional port enablement interface, and instantiation of clocking helper blocks. It initializes, configures, and links a GT*_QUAD primitive and includes simulation support to demonstrate various GT Quad features. The Wizard subsystem enables the sharing of multiple Interface IPs, with each supporting multiple line rates.

Versal Adaptive SoC Transceiver Subsystem supports the following:

- A single wizard subsystem core can instantiate a maximum of five GT*_QUAD primitive.
- A single wizard subsystem core can share a maximum of eight Interface IPs.
- Simplex and Duplex configurations support:
 1. Preset configurations for common standards.
 2. Full control of parameters for fine-tuning.
- Native support for dynamic line rate changing, eliminating the requirement for reprogramming or building an APB3 controller. Up to 16 line rates can be configured per channel in each direction.

Note: Multiple instances of wizard subsystem cores can be instantiated within a design as per the design requirements. The wizard subsystem is not restricted to use with the IP integrator for creating designs.

Wizard subsystem is created based on the older version of transceiver GT QUAD IP (gt_quad_base) that instantiates GT*_QUAD primitive. As illustrated in the following section, the Wizard subsystem is a wrapper around GT QUAD IPs, Reset FSM, and clocking helper blocks for various configurations.

Figure 1: Single Interface Single Quad

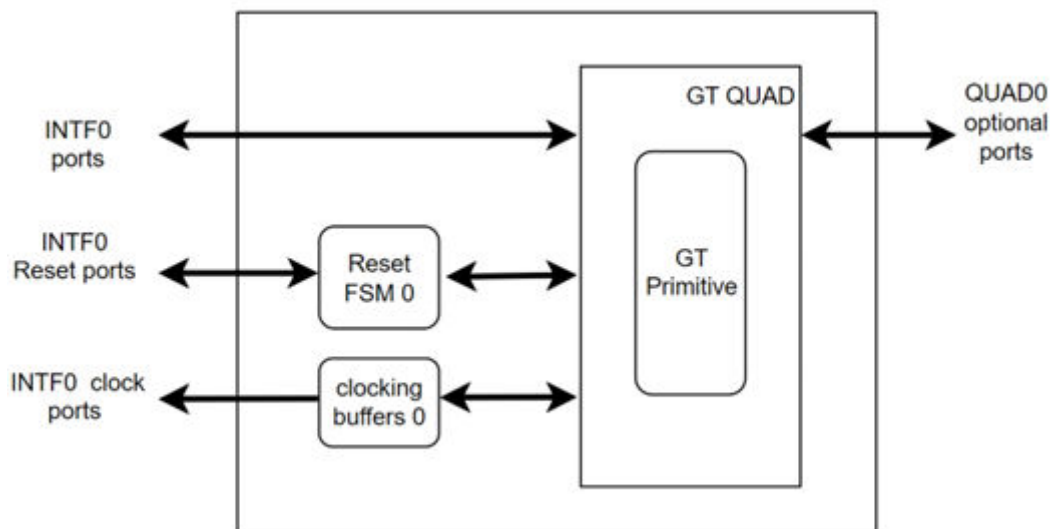


Figure 2: Multiple Interfaces Single Quad

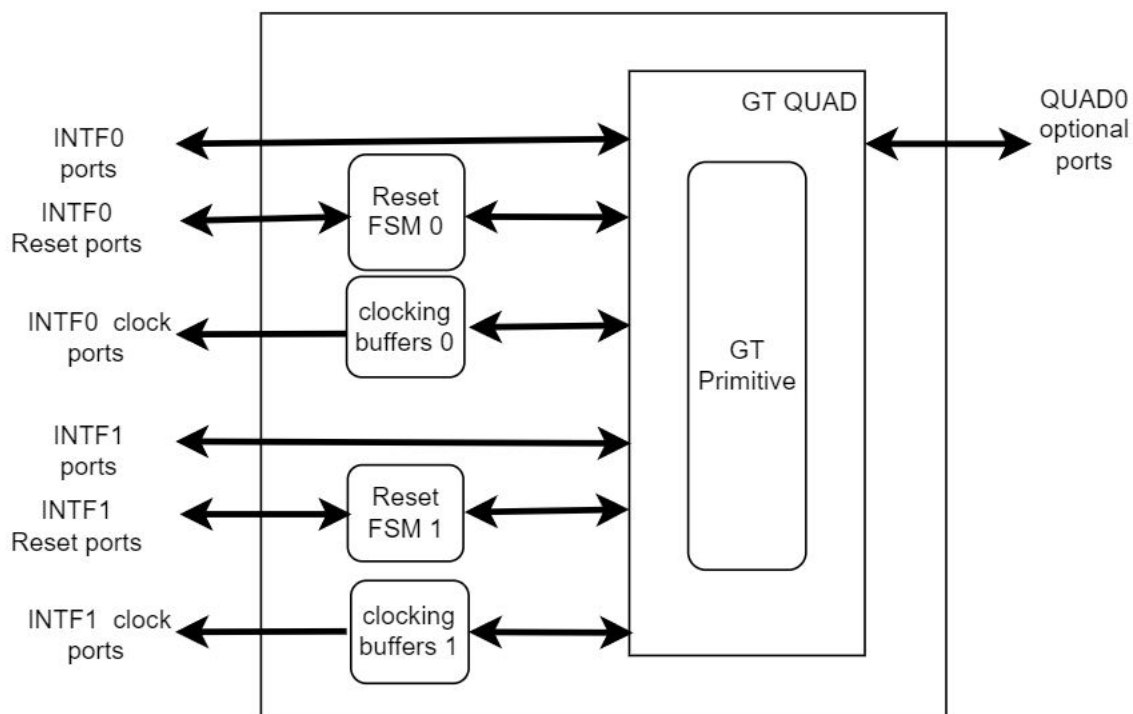


Figure 3: Single Interface Multi Quad

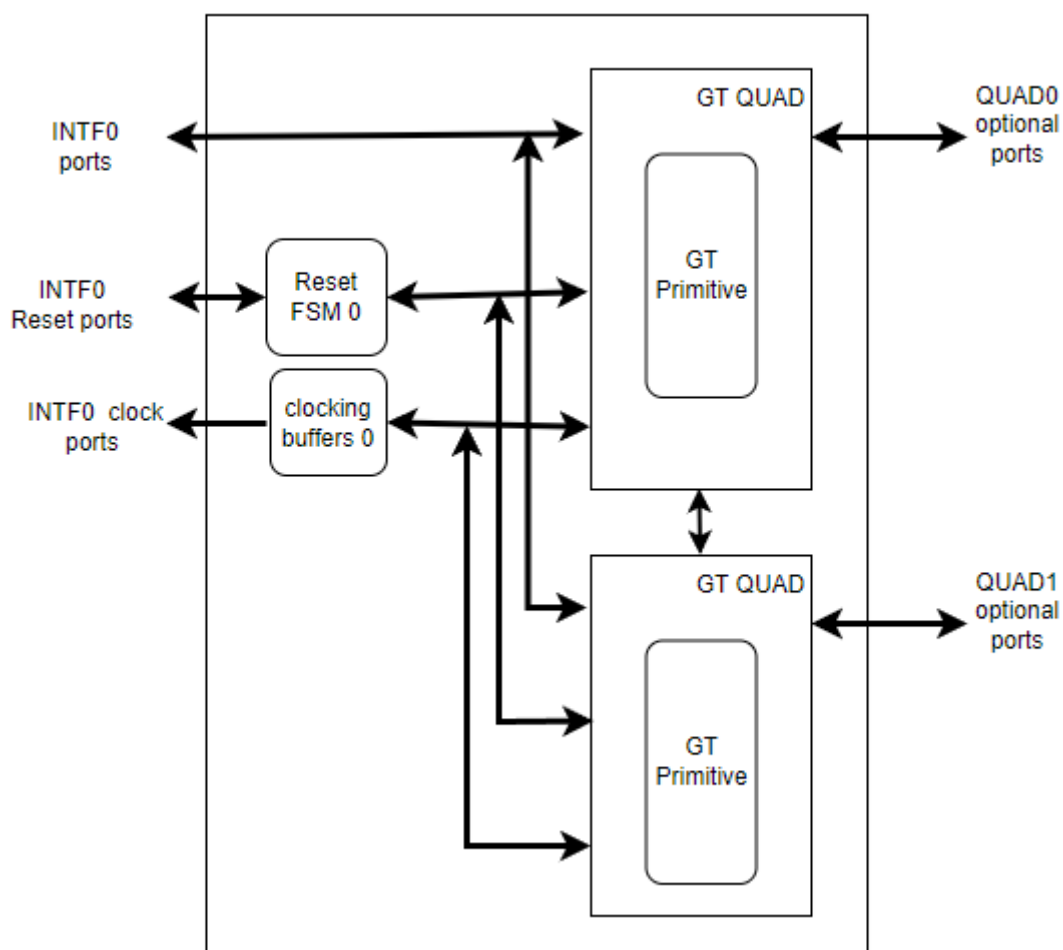
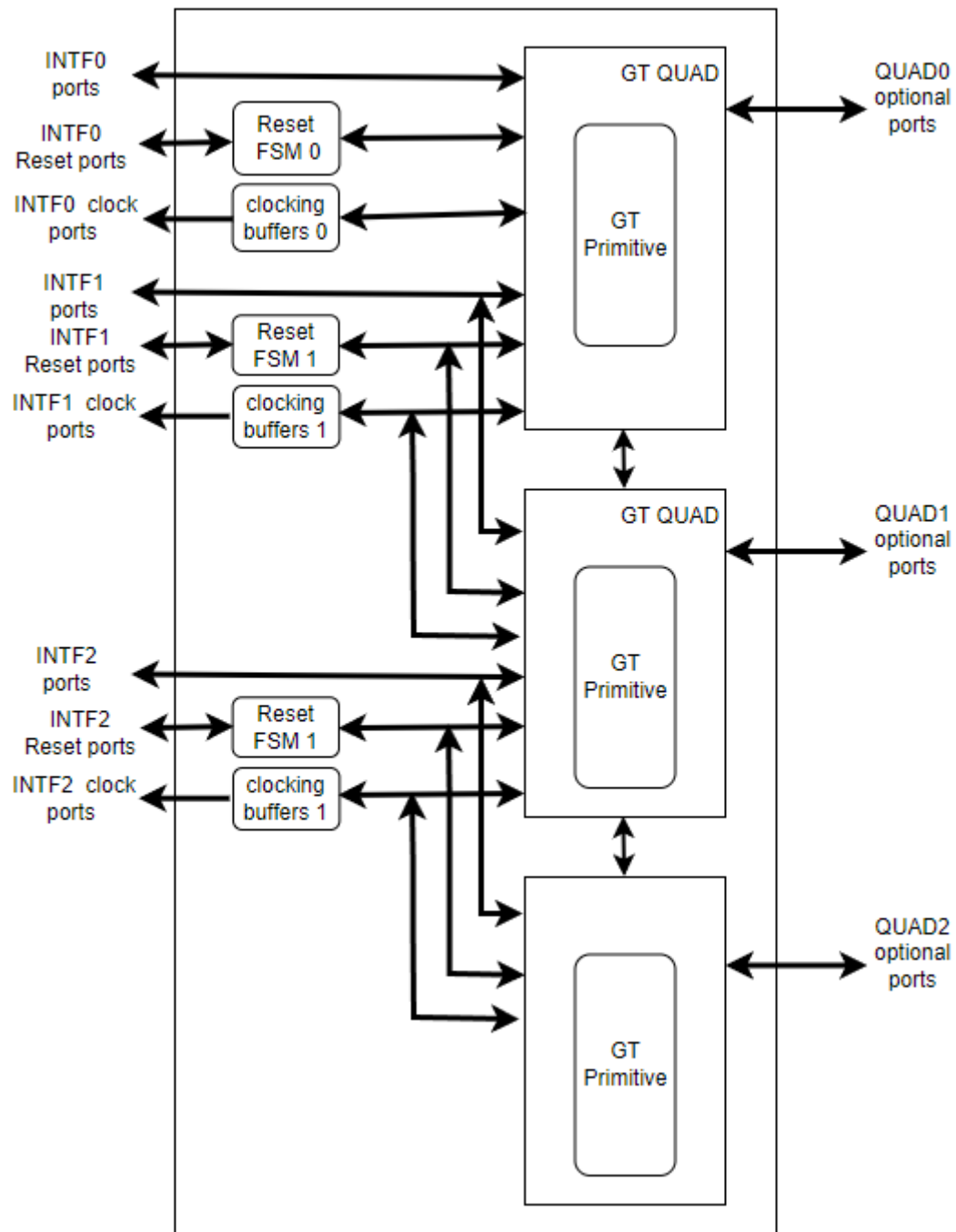


Figure 4: Multi Interface Multi Quad



For port list and definitions of this IP, see *Versal Adaptive SoC GTY and GTYP Transceivers Architecture Manual* ([AM002](#)) and *Versal Adaptive SoC GTM Transceivers Architecture Manual* ([AM017](#)).

Note: Starting 2024.2, Versal Adaptive SoC Transceiver Subsystem is the primary transceiver design tool for Versal devices. The support for AMD IP and IP integrator is limited in 2024.1. For configuring the Transceiver Wizard Subsystem properly in 2024.1, refer to Answer Record [36251](#).

Designing with the Example Design

An example design can be generated for any instance of the Wizard IP core. The example design instantiates the core instance, any helper blocks that you have chosen to locate in the example design, and the requisite reference clock buffers. The contents of the example design are customized to support the specific core customization. Use of the example design as a demonstration and as a starting point for integration into your system is suggested.

Reset Controller Helper Block

The Reset Controller helper block must be provided with the free running clock `gtwiz_reset_clk_freerun_in` that you have specified during IP customization. A single instance of the helper block is delivered with each interface added to the wizard subsystem.

Table 1: Reset Controller Helper Block Ports

Name	Direction	Width	Clock Domain	Description
<code>gtwiz_reset_clk_freerun_in</code>	IN	1	ASYNC	Free running clock, used to reset transceiver primitives. It should be toggling before device configuration.
<code>gtwiz_reset_all_in</code>	IN	1	ASYNC	Signal to reset the phase-locked loops (PLLs) and active data directions of transceiver primitives. The falling edge of an active-High, asynchronous pulse of at least one <code>gtwiz_reset_clk_freerun_in</code> period in the duration initializes the process.
<code>gtwiz_reset_tx_pll_and_data_path_in</code>	IN	1	ASYNC	Signal to reset the transmit data direction and associated PLLs of transceiver primitives. An active-High, asynchronous pulse of at least one <code>gtwiz_reset_clk_freerun_in</code> period in the duration initializes the process.

Table 1: Reset Controller Helper Block Ports (cont'd)

Name	Direction	Width	Clock Domain	Description
gtwiz_reset_tx_datapath_in	IN	1	ASYNC	Signal to reset the transmit data direction of transceiver primitives. An active-High, asynchronous pulse of at least one <code>gtwiz_reset_clk_freerun_in</code> period in the duration initializes the process.
gtwiz_reset_rx_pll_and_datapath_in	IN	1	ASYNC	Signal to reset the receive data direction and associated PLLs of transceiver primitives. An active-High, asynchronous pulse of at least one <code>gtwiz_reset_clk_freerun_in</code> period in the duration initializes the process.
gtwiz_reset_rx_datapath_in	IN	1	ASYNC	Signal to reset the receive data direction of transceiver primitives. An active-High, asynchronous pulse of at least one <code>gtwiz_reset_clk_freerun_in</code> period in the duration initializes the process.
gtpowergood_in	IN	1	ASYNC	Connected to GTPowerGood signals produced by the transceiver channel logic.
gtwiz_reset_userclk_tx_active_in	IN	1	ASYNC	For GTY, GTYP: Logical AND of all <code>TXPMARESETDONE</code> signals produced by the transceiver channel primitives. For GTM: Logical AND of all <code>TXPROGDIVRESETDONE</code> signals produced by the transceiver channel primitives.
gtwiz_reset_userclk_rx_active_in	IN	1	ASYNC	For GTY, GTYP: Logical AND of all <code>RXPMARESETDONE</code> signals produced by the transceiver channel primitives. For GTM: Logical AND of all <code>RXPROGDIVRESETDONE</code> signals produced by the transceiver channel primitives.
mst_tx_resetdone	IN	1	ASYNC	Logical AND of all <code>MSTTXRESETDONE</code> signals produced by transceiver channel primitives.
mst_tx_resetdone	IN	1	ASYNC	Logical AND of all <code>MSTRXRESETDONE</code> signals produced by the transceiver channel primitives.

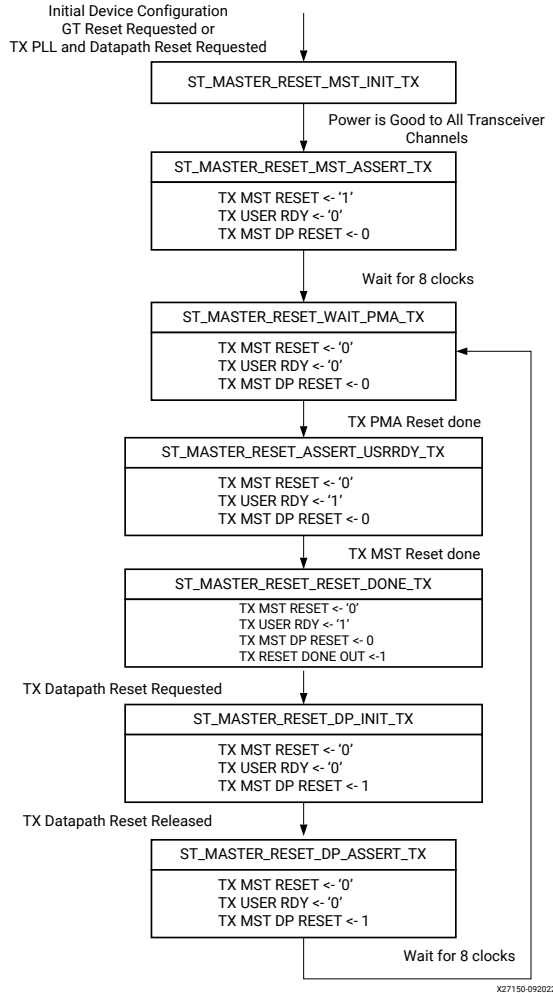
Table 1: Reset Controller Helper Block Ports (cont'd)

Name	Direction	Width	Clock Domain	Description
mst_tx_reset	OUT	1	gtwiz_reset_clk_freerun_in	Active-High signal fanned out to TXMSTRESET port of transceiver channels.
mst_rx_reset	OUT	1	gtwiz_reset_clk_freerun_in	Active-High signal fanned out to RXMSTRESET port of transceiver channels.
mst_tx_dp_reset	OUT	1	gtwiz_reset_clk_freerun_in	Active-High signal fanned out to TXMSTDATAPATHRESET port of transceiver channels.
mst_rx_dp_reset	OUT	1	gtwiz_reset_clk_freerun_in	Active-High signal fanned out to RXMSTDATAPATHRESET port of transceiver channels.
txuserddy_out	OUT	1	gtwiz_reset_clk_freerun_in	Active-High signal fanned out to TXUSERRDY port of all transceiver channel primitives.
rxuserddy_out	OUT	1	gtwiz_reset_clk_freerun_in	Active-High signal fanned out to RXUSERRDY port of all transceiver channel primitives.
gtwiz_reset_tx_done_out	OUT	1	gtwiz_reset_clk_freerun_in	Active-High indication that the transmitter reset sequence of transceiver primitives, as initiated by the reset controller helper block, is completed.
gtwiz_reset_rx_done_out	OUT	1	gtwiz_reset_clk_freerun_in	Active-High indication that the receiver reset sequence of transceiver primitives, as initiated by the reset controller helper block, is completed.
tx_clr_out	OUT	1	gtwiz_reset_clk_freerun_in	Active-High signal fanned out to clr_out port of clocking buffer module in transmitter clock path.
rx_clr_out	OUT	1	gtwiz_reset_clk_freerun_in	Active-High signal fanned out to clr_out port of clocking buffer module in receiver clock path.
tx_clrb_leaf_out	OUT	1	gtwiz_reset_clk_freerun_in	Active-Low signal fanned out to clrb_leaf port of clocking buffer module in transmitter path.
rx_clrb_leaf_out	OUT	1	gtwiz_reset_clk_freerun_in	Active-Low signal fanned out to clrb_leaf port of clocking buffer module in receiver path.

The helper block follows the controller reset sequence and contains the following two state machines:

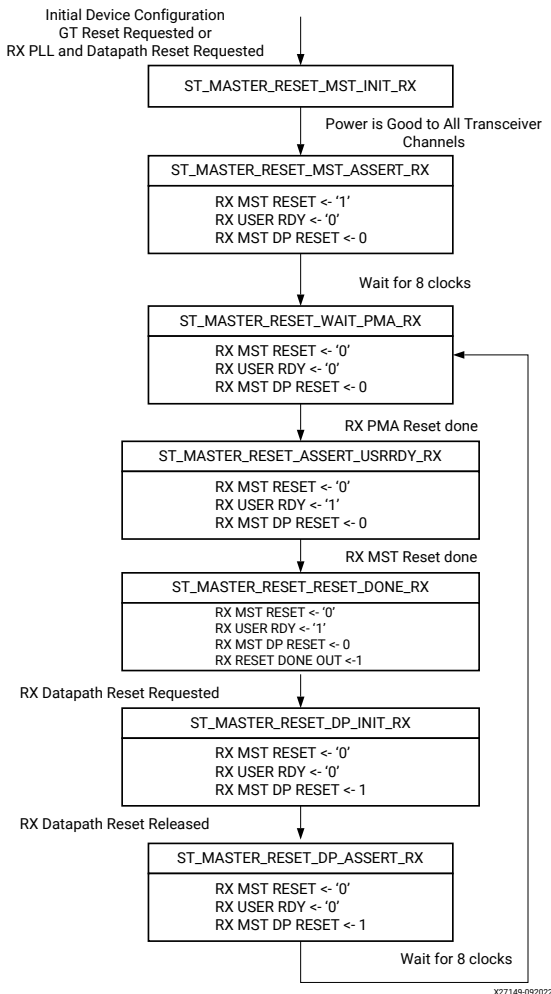
- **Transmitter Reset State Machine:** Resets the transmitter PLL and/or the transmitter datapath of all transceiver primitives and indicates their completion.

Figure 5: Transmitter Reset State Machine



- **Receiver Reset State Machine:** Resets the receiver PLL and/or the receiver datapath of all transceiver primitives and indicates their completion.

Figure 6: Receiver Reset State Machine



The `gtwiz_reset_all_in` input initiates both transmitter and receiver state machines. The transmitter and receiver reset state machines are independent of one another. Each can be initiated either directly through the user interface if needed or controlled by the `gtwiz_reset_all_in` input.

When the `gtwiz_reset_all_in` signal is activated, the transmitter and receiver state machines are initiated simultaneously. If the channel configuration is set up such that the receiver incoming data is dependent on the transmitter data, the receiver should go through a separate datapath reset after the `gtwiz_reset_all_in` has completed, or use the `reset_tx_pll_and_datapath` followed by `reset_rx_datapath`.

The following are reset ports on the subsystem boundary that are connected to the reset FSM module.

Table 2: Reset Ports on the Subsystem Boundary

Name	Direction	Description
INTF*_rst_all_in	IN	Connected to the <code>gtwiz_reset_all_in</code> port of the reset controller module.
INTF*_rst_tx_pll_and_datapath_in	IN	Connected to the <code>gtwiz_reset_tx_pll_and_datapath_in</code> port of the Reset controller module.
INTF*_rst_rx_pll_and_datapath_in	IN	Connected to the <code>gtwiz_reset_rx_pll_and_datapath_in</code> port of the Reset controller module.
INTF*_rst_tx_datapath_in	IN	Connected to the <code>gtwiz_reset_tx_datapath_in</code> port of the reset controller module.
INTF*_rst_rx_datapath_in	IN	Connected to the <code>gtwiz_reset_rx_datapath_in</code> port of the reset controller module.
INTF*_rst_tx_done_out	OUT	Connected to the <code>gtwiz_reset_tx_done_out</code> port of the reset controller module.
INTF*_rst_rx_done_out	OUT	Connected to the <code>gtwiz_reset_rx_done_out</code> port of the reset controller module.
INTF*_TX_clr_out	OUT	Connected to <code>tx_clr_out</code> port of the Reset controller module.
INTF*_RX_clr_out	OUT	Connected to the <code>rx_clr_out</code> port of the reset controller module.
INTF*_TX_clrb_leaf_out	OUT	Connected to the <code>tx_clrb_leaf_out</code> port of the reset controller module.
INTF*_RX_clrb_leaf_out	OUT	Connected to the <code>rx_clrb_leaf_out</code> port of the reset controller module.

Reset Sequencing and Other Services

The transmitter and receiver reset state machines implement the relevant master reset sequences as specified in the AMD Versal™ *Versal Adaptive SoC GTY and GTYP Transceivers Architecture Manual* (AM002) or *Versal Adaptive SoC GTM Transceivers Architecture Manual* (AM017). The reset controller helper block transceiver interface connects to the transceiver primitives. Following device configuration, no reset helper block reset inputs should be asserted until transceiver power is reported as good. The reset controller helper block internally holds all PLL and datapath resources in reset until `GTPOWERGOOD` is High from the GT Quad and then resets all transceiver resources by transitioning once through the transmitter and receiver state machines. As a result, you should wait for either the initial assertion of the `gtpowergood` port on the GT Wizard subsystem, or of both `INTF*_rst_tx_done_out` and `INTF*_rst_rx_done_out` before attempting subsequent resets of any kind.

Note: Assuming all requirements are met (`refclk` is stable) when and done values rises, the reset might take up to 150 ms to complete. The time taken varies depending on the GT configuration. This behavior holds true for the following configurations:

- Production rev silicon

- Full GT reset (TX/RX)
- All four channels / QUAD when used with the default configuration
- NPICLK = 300 MHz

Rate Change

The GT Wizard offers flexibility by enabling you to dynamically adjust the operating line rate. Based on the configuration provided in the GT Wizard, all attributes are set during the rate adjustment process. Rate changes can be executed by toggling the `INTF*_TX_ch_txrate` and `INTF*_RX_ch_rxrate` ports to any of the pre-configured rates available in the GT Wizard. You should wait for the assertion of the `INTF*_rst_tx_done_out` and `INTF*_rst_rx_done_out` signals as an indication that the current rate change process has been completed.

If the reference clock frequency remains unchanged, you can use the rate change ports for the adjustment process. For GTYE5 and GTYP variants, if the reference clock frequency is altered either on the same reference clock port or on a different one, you must ensure adherence to the rate change sequence outlined in *Versal Adaptive SoC GTY and GTYP Transceivers Architecture Manual* ([AM002](#)) by using GPI and GPO ports.

For GTME5 variant, the rate change sequence can be achieved by using rate ports on the GT Wizard subsystem even if the reference clock frequency is altered. GTME5 variant doesn't need GPI and GPO ports for rate change sequence unlike GTYE5 and GTYP variants. For additional details, refer to the rate change sequence outlined in *Versal Adaptive SoC GTM Transceivers Architecture Manual* ([AM017](#)).

Recommendations on the Reset Usage and Rate change for GT Wizard

- Resets do not propagate to GT Quad when the `gtpowergood` from GT Wizard is low
- The rate port of the GT Wizard should be tied to 0 until GT Power good is low
- When GT Reset is asserted to the GT Wizard, Reset dones from the GT Wizard indicate the reset sequence is completed successfully
- When rate port is changed on the GT Wizard, Reset dones from the GT Wizard indicate the rate change is successful
- It is recommended to assert the rate change only when the Reset dones from the GT Wizard are high
- It is recommended to assert the GT Reset only when the Reset dones from the GT Wizard are high
- It is not recommended to asserted GT Reset and rate change at the same time

Unsupported Features

- You cannot connect the Wizard Subsystem and GT Quad Base IP (*Versal Adaptive SoC Transceivers Wizard LogiCORE IP Product Guide* ([PG331](#))) to the same interface IP in a single design.

For the complete list of unsupported configurations, refer to Answer Record [36548](#).

Register Space

Download the Adaptive SoC GTM and Adaptive SoC GTY and GTYP register maps from the following links:

- [Adaptive SoC GTM register map](#)
- [Adaptive SoC GTY and GTYP register map](#)

Design Flow Steps

This section describes customizing and generating the subsystem, constraining the subsystem, and the simulation, synthesis, and implementation steps that are specific to this IP subsystem. More detailed information about the standard AMD Vivado™ design flows and the IP integrator can be found in the following Vivado Design Suite user guides:

- *Vivado Design Suite User Guide: Designing IP Subsystems using IP Integrator* ([UG994](#))
- *Vivado Design Suite User Guide: Designing with IP* ([UG896](#))
- *Vivado Design Suite User Guide: Getting Started* ([UG910](#))
- *Vivado Design Suite User Guide: Logic Simulation* ([UG900](#))

Customizing and Generating the Subsystem

This section includes information about using AMD tools to customize and generate the subsystem in the AMD Vivado™ Design Suite.

If you are customizing and generating the subsystem in the Vivado IP integrator, see the *Vivado Design Suite User Guide: Designing IP Subsystems using IP Integrator* ([UG994](#)) for detailed information. IP integrator might auto-compute certain configuration values when validating or generating the design. To check whether the values do change, see the description of the parameter in this chapter. To view the parameter value, run the `validate_bd_design` command in the Tcl console.

You can customize the IP for use in your design by specifying values for the various parameters associated with the IP subsystem using the following steps:

1. Select the IP from the IP catalog.
2. Double-click the selected IP or select the Customize IP command from the toolbar or right-click menu.

For details, see the *Vivado Design Suite User Guide: Designing with IP* ([UG896](#)) and the *Vivado Design Suite User Guide: Getting Started* ([UG910](#)).

Figures in this chapter are illustrations of the Vivado IDE. The layout depicted here might vary from the current version.

Customizing Vivado IDE Organization

The Wizard IP Customization AMD Vivado™ IDE is organized in four tabs:

- **Wrapper Configuration:** Provides customization options for fundamental wizard features, including the transceiver type, Number of Quads, and Number of interfaces. Also, it provides ability to select the Interface configurations such as direction, lanes, transmitter, and receiver settings.
- **Quad Interface Mapping:** Provides tabular view of mapping the interface to the transceiver lanes. Also provides options to choose the master clock source for each interface.
- **Structural Options:** Provides customization option to choose clocking helper block within the core or example design. Also provides options to enable the optional transceiver ports.
- **INTF TX/RX Optional Port:** Provides options to choose optional ports for each Interface. This page has multiple selection options in case of multiple interface configurations. Provides flexibility to enable the ports based on the user application

Review each of the available options and modify them as desired so that the resulting core instance meets your system requirements.

Component Name and Symbol

The name of the generated IP is set in the Component Name field. The default name is `gtwiz_versal_0`. This must be set to a name that is unique within your project. The IP symbol is shown on the left-hand side of the IP Customization Vivado IDE, and displays only enabled ports by default. The IP symbol is updated based on your selection in structural options and INTF TX/RX Optional Ports.

Wrapper Configuration Tab

IP customization options in the Wrapper Configuration tab are described in the following subsections:

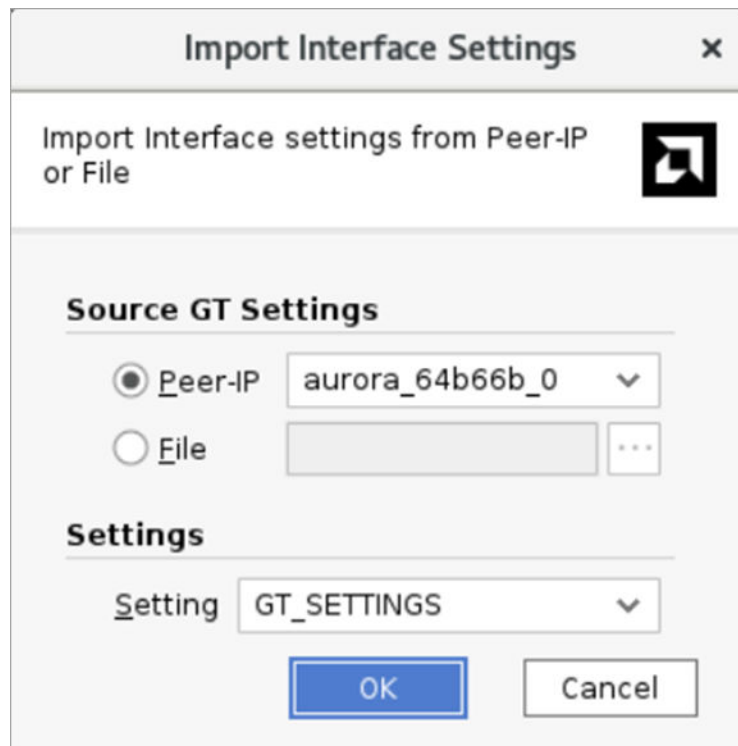
The following are customization options to configure each interface. These options are provided for each interface:

- **GT Direction:** Select GT direction options given in the drop-down menu.
- **Number of Lanes:** Select the number of lanes. Maximum value for lanes depends on the number of quads.

Note: In the case of GTME5 configurations requiring user data widths of 320/512, when the line rate is greater than 56 Gb/s, the number of lanes selected for interface are treated as the number of active PCS channels because both the PCS lanes are used. In a given GTME5 Dual, only 1 active PMA lane is possible for these rates. Refer to the configuration protocol GUI option for active PMA lane selection.

- **Import Settings Interface:** A widget provided to help you import the transceiver configuration settings. Import is triggered only when you open the import widget. The import feature populates a list of all valid settings for the project and provides you an option to choose the desired transceiver settings to use from a drop-down list. When you click OK, the corresponding settings are loaded to transceiver. Setting values loaded to the widget are not saved, default values are loaded next time you open the import widget. There are two options to import the transceiver source GT settings as shown in the following figure.

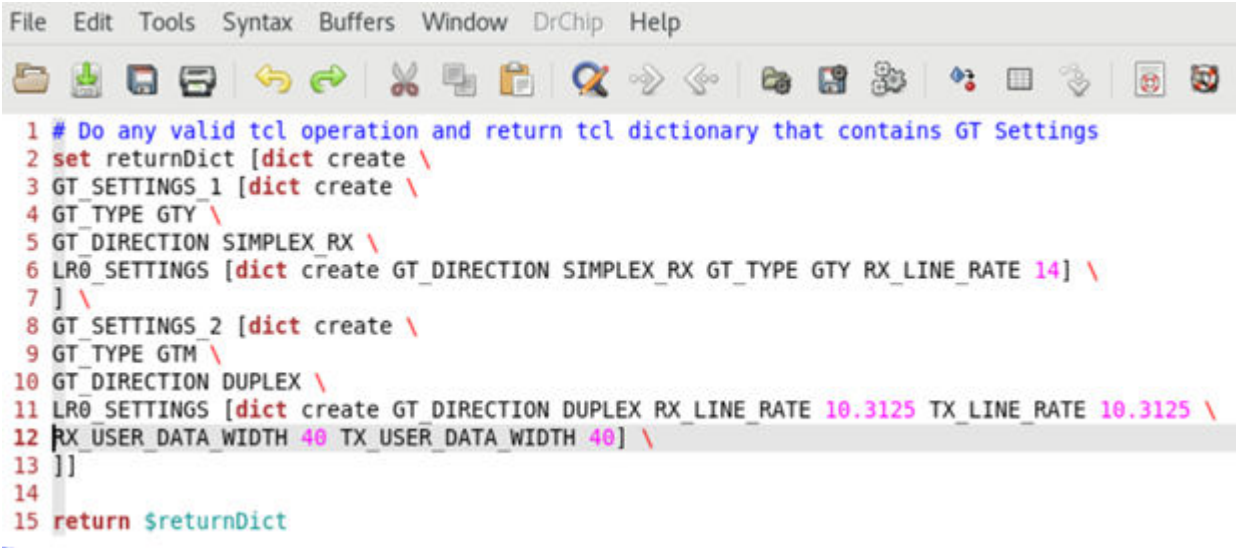
Figure 8: Import Interface Settings Dialog



Peer-IP: This selection populates a list of all AMD parent IPs added from the current Vivado project. As described earlier, transceiver settings are loaded to the parent IP when you select an IP and click OK. If the selected transceiver type and setting configuration is not compatible with the parent IP's, the widget throws an import error message.

File: This selection allows you to point to a local Tcl file that has the transceiver settings. After you provide the file path and click OK, the widget populates the transceiver settings from the local Tcl file. The following image shows an example illustrating the syntax and format of a Tcl file.

Figure 9: Tcl File Syntax for the Import Feature



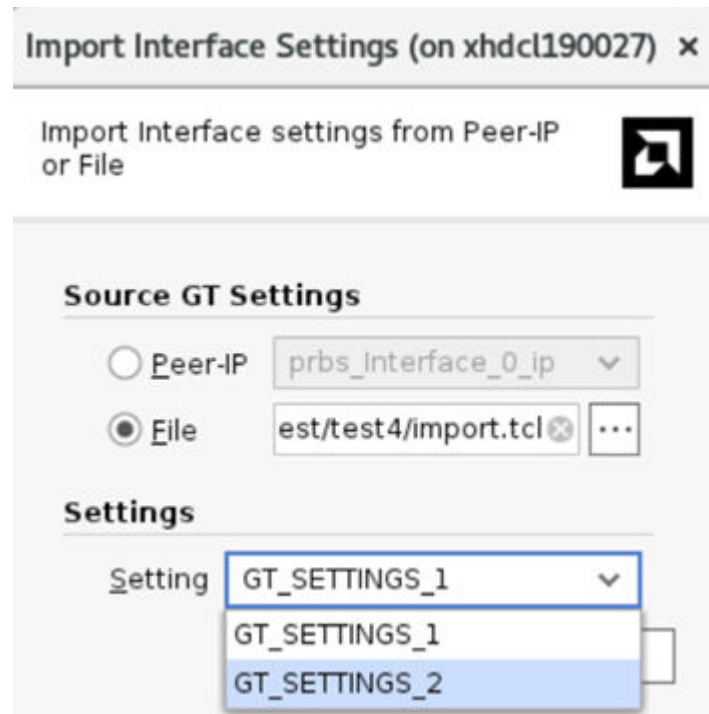
```

1 # Do any valid tcl operation and return tcl dictionary that contains GT Settings
2 set returnDict [dict create \
3 GT_SETTINGS_1 [dict create \
4 GT_TYPE GTY \
5 GT_DIRECTION SIMPLEX_RX \
6 LR_SETTINGS [dict create GT_DIRECTION SIMPLEX_RX GT_TYPE GTY RX_LINE_RATE 14] \
7 ] \
8 GT_SETTINGS_2 [dict create \
9 GT_TYPE GTM \
10 GT_DIRECTION DUPLEX \
11 LR_SETTINGS [dict create GT_DIRECTION DUPLEX RX_LINE_RATE 10.3125 TX_LINE_RATE 10.3125 \
12 RX_USER_DATA_WIDTH 40 TX_USER_DATA_WIDTH 40] \
13 ]]
14
15 return $returnDict

```

The Tcl file shown above has two settings that you can select using the settings drop down menu as follows.

Figure 10: Import Transceiver Settings from the Tcl File



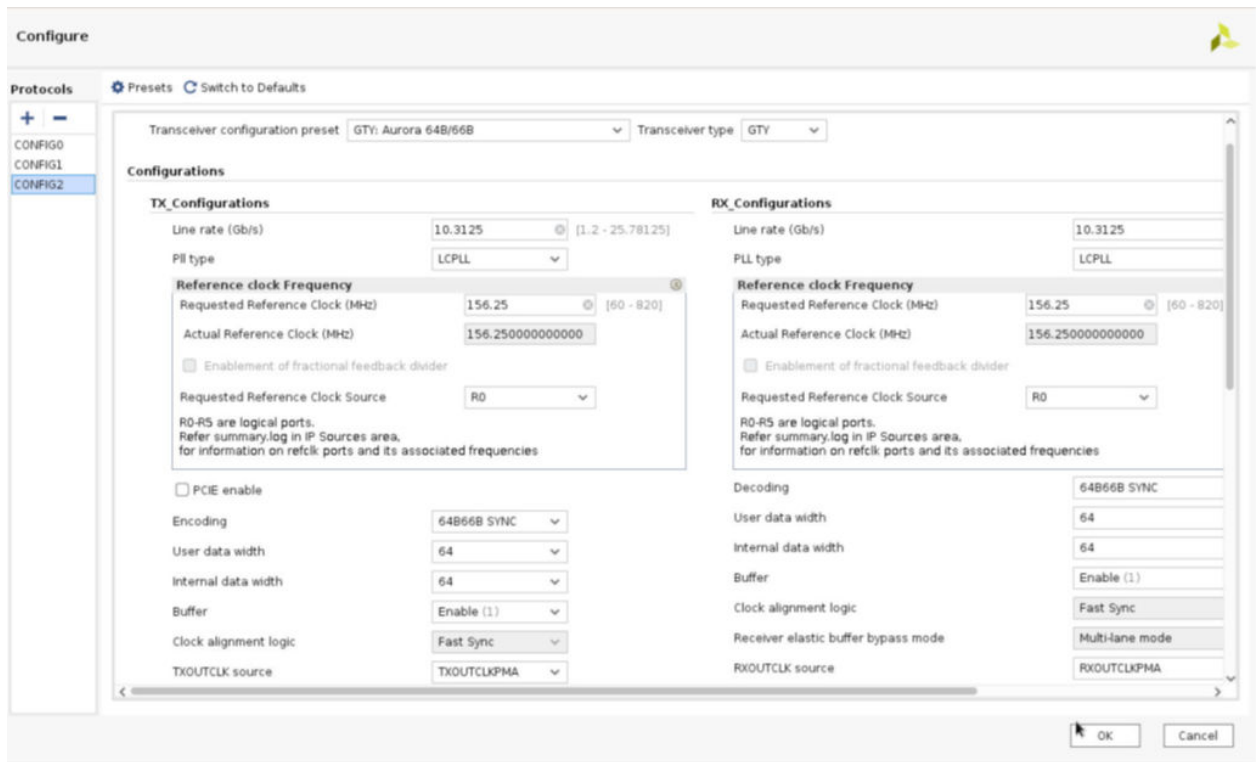
After you click OK, the respective transceiver settings can be seen from the Transceiver Configs Interface sub-GUI.

Note:

- Variables defined in the Tcl file must be defined as a key-value pair, that is in the form of a dictionary.
- `GT_TYPE` and `GT_DIRECTION` are mandatory keys. `LR*_SETTINGS` is an optional key. See the sample Tcl file image above.
- The value of `GT_TYPE` and `GT_DIRECTION` must match with the values configured in the wizard subsystem GUI.
- In the sample Tcl file image shown above, for the `LR_SETTINGS` value that is expected in dictionary format, where the key represent the parameter in the sub-GUI and the value represent the parameter value. For the keys that are not defined in the dictionary, default values are loaded to the transceiver configuration settings. For more information on `LR_SETTINGS` string, see [Sample Tcl File for Import](#).
- **Preset:** Standard protocol presets are supported. Selecting a preset loads corresponding transceiver settings in Sub GUI. Multi-line rate presets can be loaded in this Preset window. It populates multiple single line rate presets in the corresponding Sub GUI. For example, multi-line rate presets with the suffix MLR are added in preset lists.

- **Transceiver Config Interface:** The Sub GUI configuration tab provides customization options for fundamental transceiver features, and transmitter and receiver settings. Each Sub GUI configuration corresponds to a target transceiver configuration referred to as `CONFIG<0/1...>` in the Protocols panel. The target configuration could be selected by controlling the `ch*_txrate[7:0]` or `ch*_rxrate[7:0]` ports on the wizard. Multiple line rate settings require multiple Sub GUI configurations. The following figure displays the tab:

Figure 11: Transceiver Config Interface

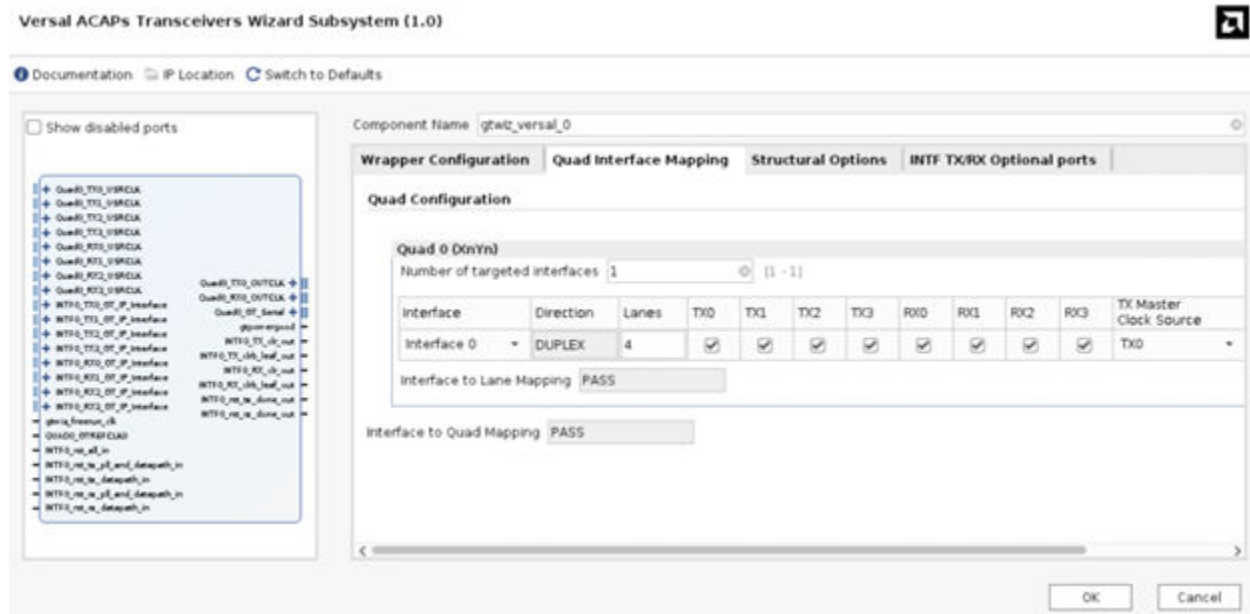


- **Interface Configuration Check:** Read-only GUI option provides guidance to the user if the information provided in the Wrapper Configuration tab is correct.

Quad Interface Mapping Tab

This page provides customization options to assign the lanes of each quad to various interfaces defined in the wrapper configuration tab. The GUI layout expands based on the number of Quads selected in the wrapper configuration tab. The customization options to map quads to each interface are provided. The following are customization GUI options provided for each Quad:

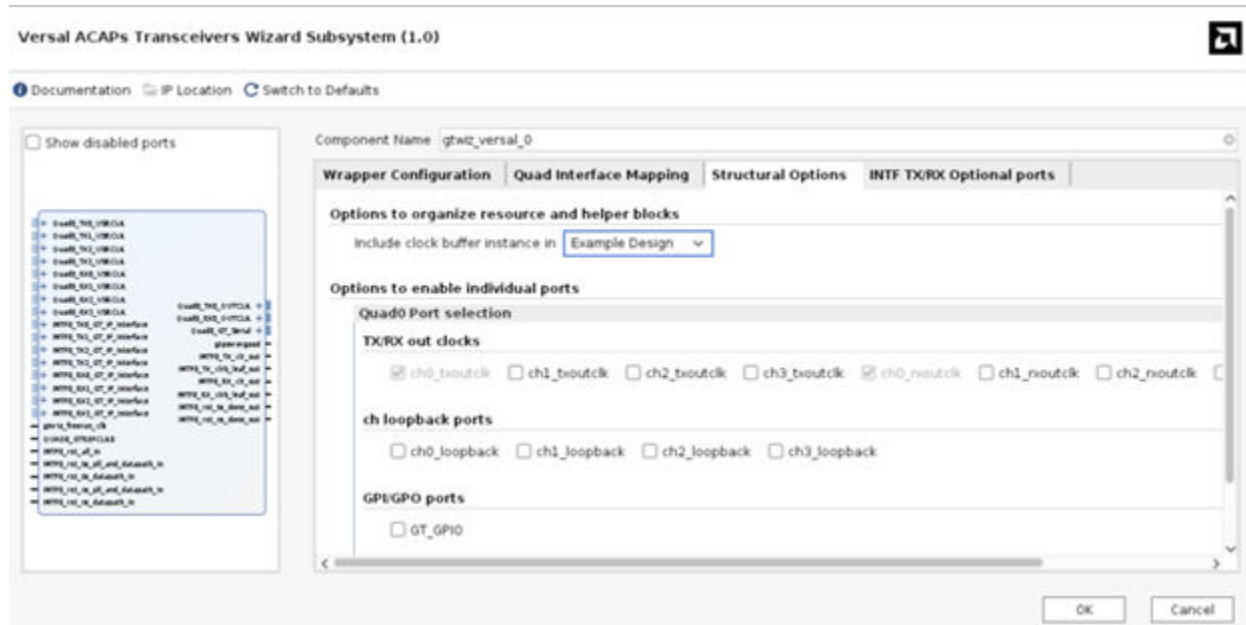
Figure 12: Quad Interface Mapping Tab



- **Number of targeted Interfaces:** Enter the number of interfaces to be mapped to the current Quad. The table expands based on the number of targeted interfaces entry. Customization layout provides options to map the lanes of current quad to the targeted interfaces. The direction of each interface is auto populated based on the information provided in the wrapper configuration. The lanes for each interface need to be provided as customization input.
- **TX Master Clock source:** Select desired master clock source options given in the dropdown menu.
- **Interface to lane Mapping:** Read-only GUI option guides the user on mapping multiple interfaces to the single quad. The customization GUI checks to evaluate if the selected interfaces can be packed in the quad selected based on the physical resources. If the checks fail, the error message is provided to guide the users to provide the correct customization options.
- **Interface to Quad Mapping:** Read-only GUI option provides guidance to the user if the information provided in the Quad Interface Mapping tab is correct.

Structural Options Tab

Figure 13: Structural Options Tab

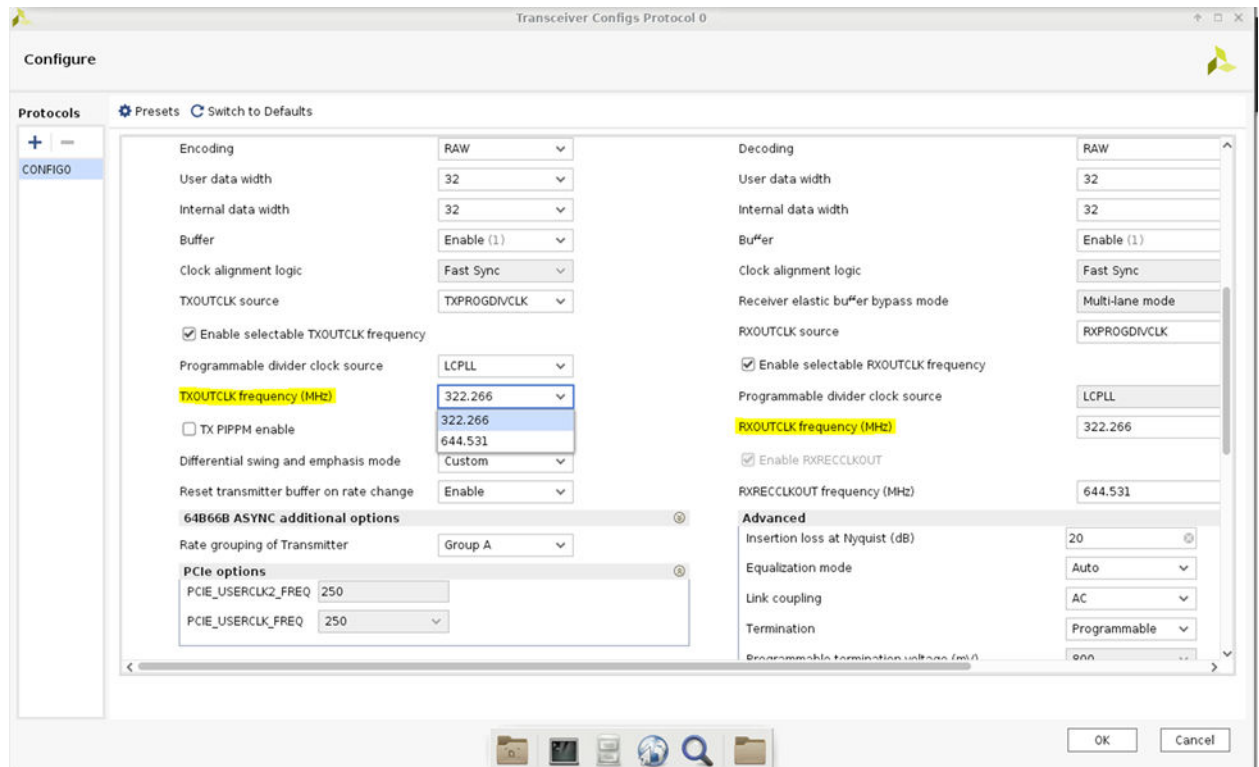


- **Include Clock Buffer instance in:**

IP customization options to include the clocking buffer in the IP core or example design. Based on the transceiver settings, this customization option can be restricted to only Example Design. The following are the configurations where the clocking buffer instance must be in the example design:

1. Receiver Elastic Buffer Bypass Mode is set to Single Lane Mode for any of the interfaces.
2. If the OUTCLK frequency is modified by the user from the default value as shown in the following figure. It is because the value of BUFG_GT dividers in the `outclk` to `usrclk` path of the transceiver needs to be adjusted accordingly.

Figure 14: Transceiver Configuration Example

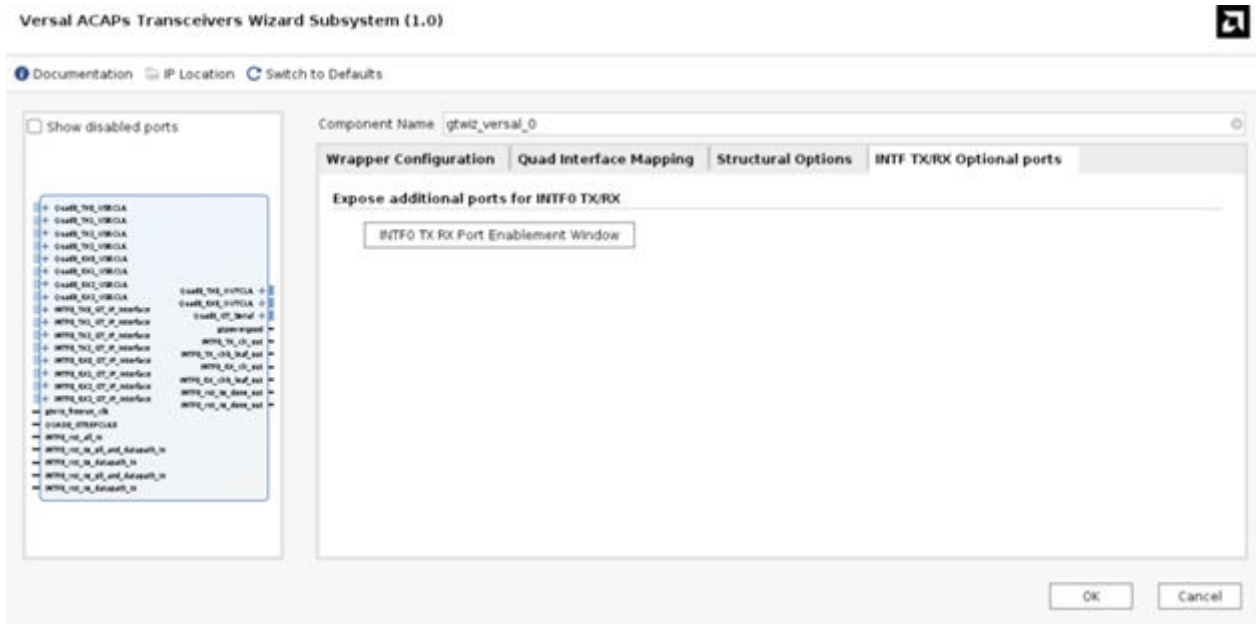


3. The Master clock source selection in the Quad Interface Mapping Tab is customized as None.
- **Options to Enable Individual Ports:** This tab provides customization options for enabling Quad-level ports according to your design preferences. Based on the configuration outlined in the wrapper configuration and quad interface mapping tab, mandatory ports are automatically enabled in the Structural Options tab. Customization options that are incompatible with the provided configuration are disabled. Optional ports are displayed in the GUI for you to enable based on your design requirements.

INTF TX/RX Optional Ports Tab

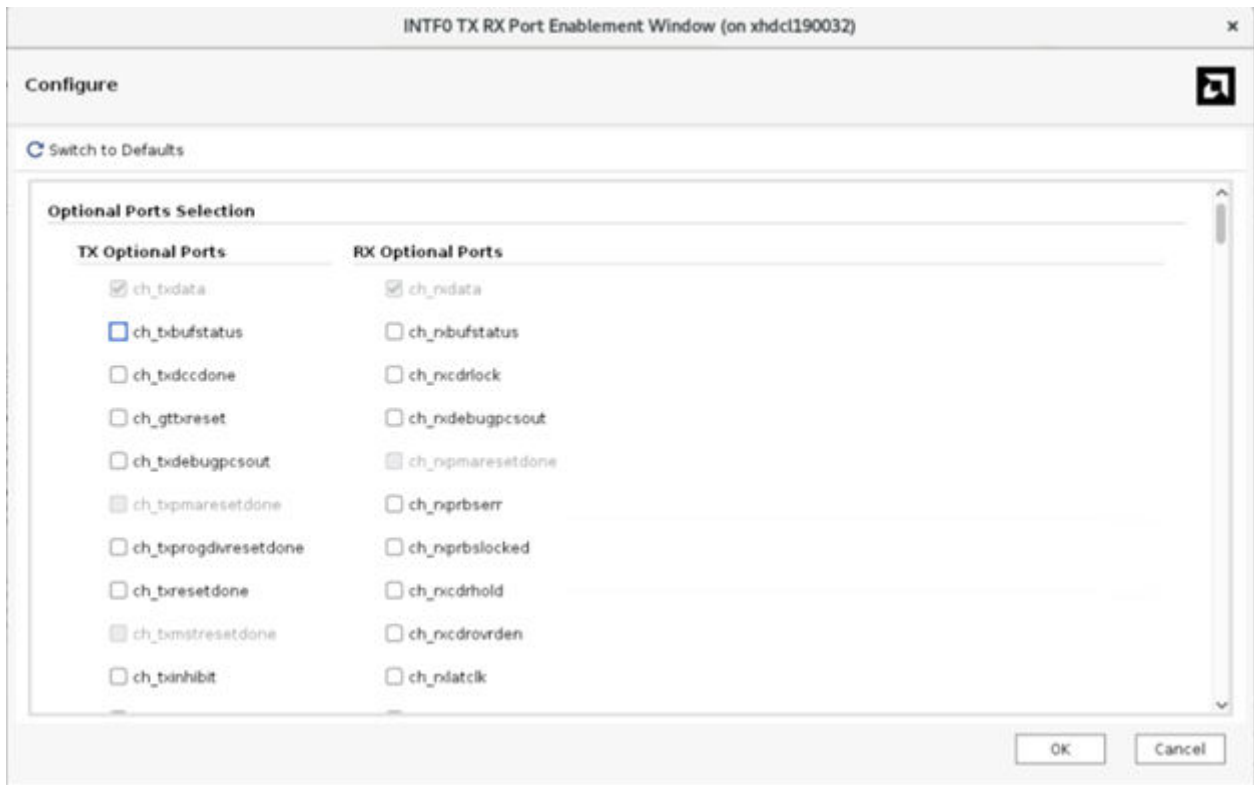
The selection options expand based on the number of interfaces selected in the wrapper configuration. The options are common for all the lanes of an interface. The customization widget provides options to enable the ports on the TX/RX interfaces. The customization of ports is disabled in the IP integrator (IP integrator) and all the ports are enabled by default. IP integrator handles the unconnected ports by driving the ports with default values.

Figure 15: INTF TX/RX Optional Ports Tab



Based on the Interface configuration, the ports which are mandatory for the configuration are enabled. The ports which are invalid for the configuration are disabled for enablement. Users can enable the optional ports according to the design requirement.

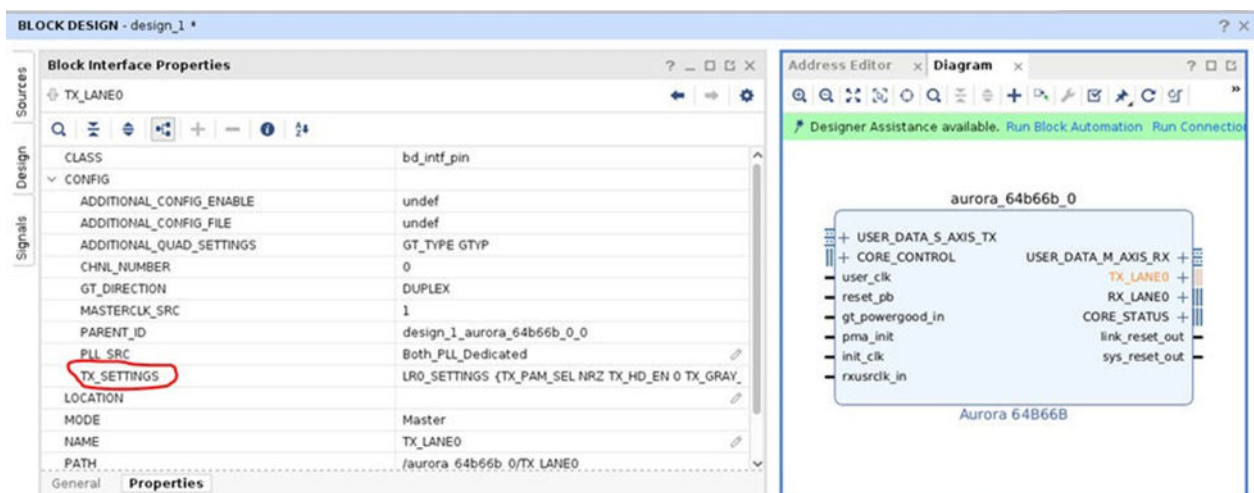
Figure 16: INTF TX/RX Port Enablement Window



Propagation in IP Integrator (IPI)

In IPI, Versal Wizard Subsystem supports propagation for Transceiver Config Interface settings when integrated with AMD Parent IP. The AMD Parent IP hosts the Transceiver Config Interface settings on the TX and RX IP integrator Interface as shown in the following figure.

Figure 17: Block Interface Properties



The following is the example of TX_SETTINGS for the Aurora IP

Figure 18: TX Settings Example for Aurora IP

```
1 TX_SETTINGS {TX_PAM_SEL NRZ TX_GRAY_BYP true TX_GRAY_LITTLEENDIAN true TX_PRECODE_BYP true TX_PRECODE_LITTLEENDIAN false TX_LINE_RATE 3.5 TX_PLL_TYPE LCPLL TX_REFCLK_FREQUENCY 156.250 TX_ACTUAL_REFCLK_FREQUENCY 156.250000062365 TX_FRACN_ENABLED true TX_FRACN_NUMERATOR 0 TX_REFCLK_SOURCE R0 TX_DATA_ENCODING 64B66B_SYNC TX_USER_DATA_WIDTH 64 TX_INT_DATA_WIDTH 64 TX_BUFFER_MODE 1 TX_BUFFER_BYPASS_MODE Fast_Sync TX_PIPM_ENABLE false TX_OUTCLK_SOURCE TXOUTCLKPMA TXPROGDIV_FREQ_ENABLE false TXPROGDIV_FREQ_SOURCE LCPLL TXPROGDIV_FREQ_VAL 322.265625 TX_DIFF_SWING_EMPH_MODE CUSTOM TX_64B66B_SCRAMBLER false TX_64B66B_ENCODER false TX_64B66B_CRC false TX_RATE_GROUP A TX_LANE_DESCRIPTOR HDMI_ENABLE false TX_BUFFER_RESET_ON_RATE_CHANGE_ENABLE GT_TYPE GTY}
```

Based on the Wizard subsystem configuration and the connections between AMD Parent IP and Wizard system, the settings are propagated to Wizard subsystem if the Transceiver Config Interface widget is in AUTO mode in IP integrator (IP integrator). The Import Interface settings customization option is disabled when the Transceiver Config Interface settings are propagated from AMD Parent IP. If the Transceiver Config Interface is set to MANUAL, you can provide the settings through manual entry in the GUI or using Import Settings Interface feature.

Note: The propagation feature is supported in Wizard subsystem only for Transceiver Config Interface widget. The other customization options provided should be manually configured by the user based on the design requirements.

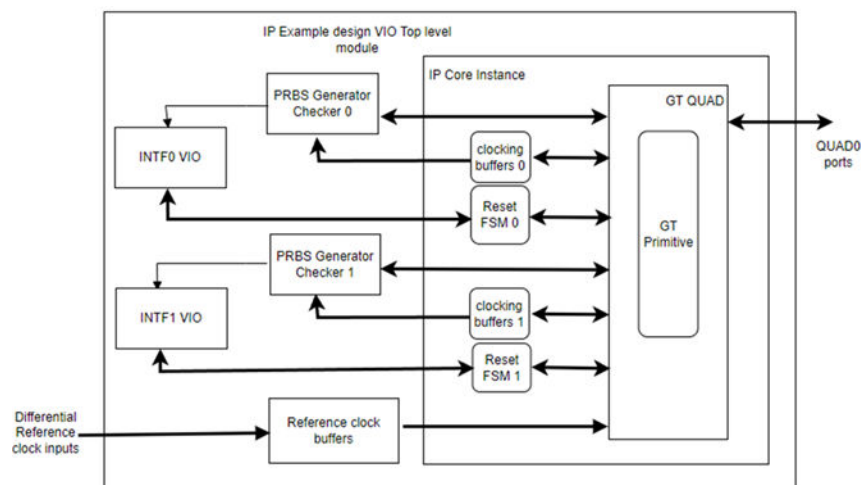
Example Design

This chapter provides information about the example design in the AMD Vivado™ Design Suite.

An example design can be generated for any instance of the Transceivers Wizard IP core. After you customize and generate a core instance, select Open IP Example Design option for that instance. A separate Vivado project opens with the wizard example design as the top-level module. The example design instantiates the customized core, add `prbs_generator_checker_ip` and does the connections for the given configuration.

The following figure shows the hierarchy and simplified representation of the wizard subsystem of the example design top level module, which is used for simulation and implementation.

Figure 19: Subsystem Example Design



The purpose of the Wizard IP example design is to:

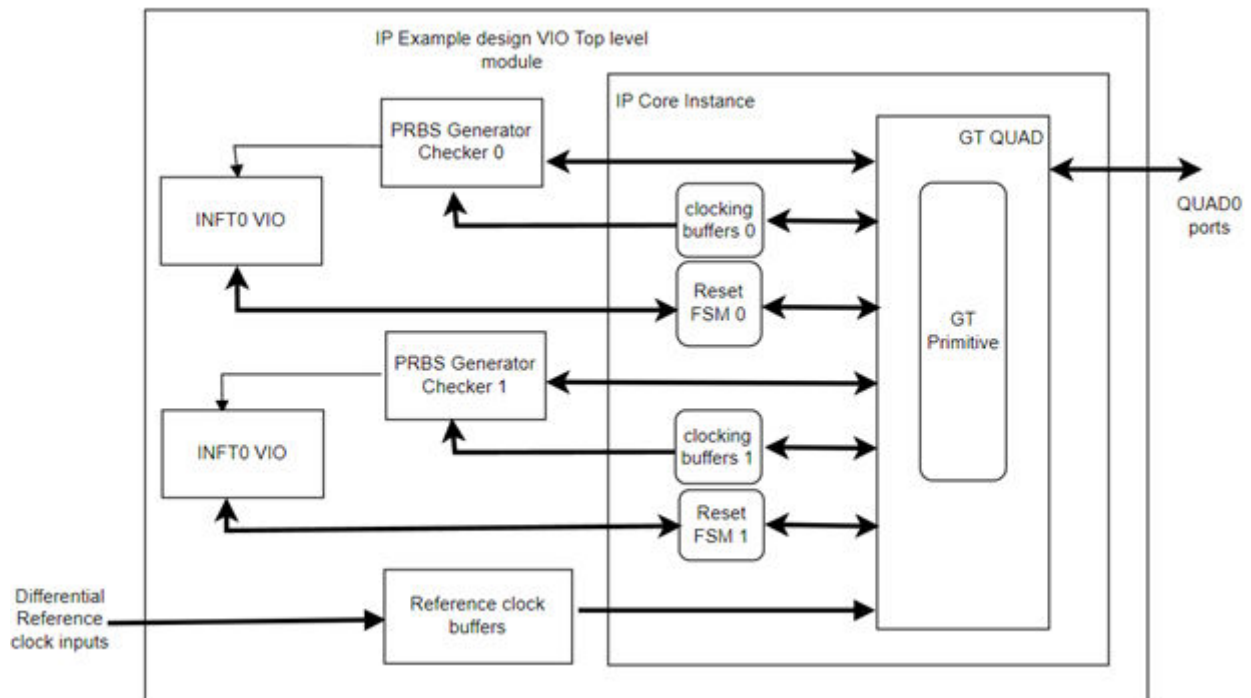
- Provide a simple demonstration of the customized core instance operating in simulation through the use of a link status indicator based on PRBS generators and checkers.
- Provide a starting point for integrating the customized core into your system, including reference clock buffers.

The `prbs_generator_checker` contains configurable PRBS generator and checker modules per transceiver channel that enable simple data integrity testing and resulting link status reporting. The example design is also synthesizable.

The example design uses sample GT Quad and GT REFCLK location for the device selected. The XDC needs to be updated based on the board layout if the expectation is to program the PDI.

The example design has additional top-level wrapper with the instantiation of VIOs to exercise the resets, rate change, and check the status of reset done and link status. The top-level VIO wrapper is suited when the example design is targeted for any board and the behavior of the wizard subsystem can be exercised using the VIOs.

Figure 20: Example Design File Structure for VIO Instantiated Top File



Limitations of the Example Design

The example design is the recommended means of simulating or implementing an instance of the Wizard IP core outside the context of your own system. It is quite simple to use, and you need to understand the following limitations:

- The example design does not implement specific protocols to generate or check data. Fundamentally, raw PRBS data is generated and checked.

- When the example design is simulated using the provided test bench, each transceiver channel is looped back from the serial data transmitter to the receiver. Data integrity can only be properly checked if the transmitter and receiver are configured for the same line rate and to use the same data coding. No rate adjustment schemes are used. If the transmitter and receiver line rates or data coding are configured differently in your system, cross-couple two appropriately customized core instances and check for data integrity in hardware or in your own test bench. In such a setup, the transmitter of core instance A is rate and coding matched to the receiver of core instance B, and vice-versa.
- The example design generation is limited for certain configurations. When user tries to generate the example design for those configurations, an error dialog box is displayed with the relevant message as shown below:

```
Example design is not supported:  
Example design is not supported as requested number of reference clocks  
are not routable in example design. To override this, please set the  
parameter EN_EXDES_REFCLK_CHECK_BYPASS to true on the IP instance and  
then open the example design. User need to take care of the Reference  
clock placement.
```

The following are the scenarios where example design generation is restricted:

- For PCIe-related configurations
- For GTM Simplex configuration
- When the selected part doesn't have transceivers that can be accessed by PL logic
- If the requested reference clock sources exceed the number of reference clock sources available for the selected part
- If the clock buffers are instantiated in the example design and the respective master clock source ports are not enabled in the core

Simulating the Example Design

To simulate an instance of the Wizard IP core, first open its example design. In the example project, select **Run → Simulation → Run Behavioral Simulation** in the Vivado Integrated Design Environment (IDE) to start a behavioral simulation. See Lab 6 in *Vivado Design Suite Tutorial: Logic Simulation* ([UG937](#)) for GTME5 integer ports related simulation updates.

The example design simulation test bench provides the requisite free-running clock and transceiver reference clock signals, and a reset all pulse to the reset controller FSM input ports. This stimulus is sufficient to enable the helper blocks to bring up the remainder of the system. After some time, the transceiver PLL(s) achieves lock, allowing the reset controller helper block finite state machines to complete the full reset sequence. After the completion of the reset sequence, you can observe the example stimulus module transmitting data. A short time later,

the example checking module begins to search for data alignment and checks for data integrity, which is in turn used by the link status logic to drive the link status indicator. In multi-line rate configurations, the example design switches to the next line rate and again tries to achieve lock and other Reset Done signals. You can observe that when `*rate_sel[3:0]` ports are toggled, their corresponding `*outclk` values change after `*resetdone` is asserted high.

Simulation Waveform Showing Rate Change

Figure 21: Simulation Waveform Showing Rate Change



The rate port should be driven with 0 until the initial `INTF*_rst_tx/rx_done_out` is asserted. The default value of `INTF*_rst_tx/rx_done_out` is high until `gtpowergood` is asserted high and goes low when `ch*_tx/rxmstreset` is applied to the quad. Finally, it measures the expected user clk frequency to ensure the expected line rate is achieved. After every rate change, it prints the following message.

```

INTFO RATE CHANGE INITIATED
INTFO CONFIG1 :::: TX RESET DONE asserted
INTFO CONFIG1 :::: TXUSRCLK Frequency = 375.059
INTFO CONFIG1 :::: TX_USER CLOCK IS PROPER. TX_USER_CLOCK is 375.059 and EXPECTED is [371.25:378.75]
INTFO CONFIG1 :::: RX RESET DONE asserted
INTFO CONFIG1 :::: RXUSRCLK Frequency = 375.059
INTFO CONFIG1 :::: RX_USER CLOCK IS PROPER RX_USER_CLOCK POST PLL LOCK is 375.059 and EXPECTED is [371.25:378.75]
INTFO POST CONFIG1 LINK STABLE TXUSRCLK Frequency = 375.059
INTFO CONFIG1 :::: POST LINK STABLE RXUSRCLK Frequency = 375.059

```

Finally, the following messages are printed to the transcript, which indicates that the test is passed.

```

INTFO RATE CHANGE INITIATED
INTFO CONFIG3 :::: TX RESET DONE asserted
INTFO CONFIG3 :::: TXUSRCLK Frequency = 250.039
INTFO CONFIG3 :::: TX_USER CLOCK IS PROPER. TX_USER_CLOCK is 250.039 and EXPECTED is [247.5:252.5]
INTFO CONFIG3 :::: RX RESET DONE asserted
INTFO CONFIG3 :::: RXUSRCLK Frequency = 250.039
INTFO CONFIG3 :::: RX_USER CLOCK IS PROPER RX_USER_CLOCK POST PLL LOCK is 250.039 and EXPECTED is [247.5:252.5]
INTFO POST CONFIG3 LINK STABLE TXUSRCLK Frequency = 250.039
INTFO CONFIG3 :::: POST LINK STABLE RXUSRCLK Frequency = 250.039
Time : 290518 ns PASS: simulation completed
** Test completed successfully

```

Debugging

This appendix includes details about resources available on the AMD Support website and debugging tools.

If the IP requires a license key, the key must be verified. The AMD Vivado™ design tools have several license checkpoints for gating licensed IP through the flow. If the license check succeeds, the IP can continue generation. Otherwise, generation halts with an error. License checkpoints are enforced by the following tools:

- Vivado Synthesis
- Vivado Implementation
- write_bitstream (Tcl command)



IMPORTANT! IP license level is ignored at checkpoints. The test confirms a valid license exists. It does not check IP license level.

Finding Help with AMD Adaptive Computing Solutions

To help in the design and debug process when using the subsystem, the [Support web page](#) contains key resources such as product documentation, release notes, answer records, information about known issues, and links for obtaining further product support. The [Community Forums](#) are also available where members can learn, participate, share, and ask questions about AMD Adaptive Computing solutions.

Documentation

This product guide is the main document associated with the subsystem. This guide, along with documentation related to all products that aid in the design process, can be found on the [AMD Adaptive Support web page](#) or by using the AMD Adaptive Computing Documentation Navigator. Download the Documentation Navigator from the [Downloads page](#). For more information about this tool and the features available, open the online help after installation.

Technical Support

AMD Adaptive Computing provides technical support on the [Community Forums](#) for this AMD LogiCORE™ IP product when used as described in the product documentation. AMD Adaptive Computing cannot guarantee timing, functionality, or support if you do any of the following:

- Implement the solution in devices that are not defined in the documentation.
- Customize the solution beyond that allowed in the product documentation.
- Change any section of the design labeled DO NOT MODIFY.

To ask questions, navigate to the [Community Forums](#).

Debug Tools

There are many tools available to address Transceiver Subsystem design issues. It is important to know which tools are useful for debugging various situations.

Vivado Design Suite Debug Feature

The AMD Vivado™ Design Suite debug feature inserts logic analyzer and virtual I/O cores directly into your design. The debug feature also allows you to set trigger conditions to capture application and integrated block port signals in hardware. Captured signals can then be analyzed. This feature in the Vivado IDE is used for logic debugging and validation of a design running in AMD devices.

The Vivado logic analyzer is used to interact with the logic debug LogiCORE IP cores, including:

- ILA 2.0 (and later versions)
- VIO 2.0 (and later versions)

See the *Vivado Design Suite User Guide: Programming and Debugging* ([UG908](#)).

Hardware Debug

Hardware issues can range from link bring-up to problems seen after hours of testing. This section provides debug steps for common issues. The AMD Vivado™ debug feature is a valuable resource to use in hardware debug. The signal names mentioned in the following individual sections can be probed using the debug feature for debugging the specific problems.

General Checks

Ensure that all the timing constraints for the core were properly incorporated from the example design and that all constraints were met during implementation.

- Does it work in post-place and route timing simulation? If problems are seen in hardware but not in timing simulation, this could indicate a PCB issue. Ensure that all clock sources are active and clean.
- If using MMCMs in the design, ensure that all MMCMs have obtained lock by monitoring the `locked` port.
- If your outputs go to 0, check your licensing.

Sample Tcl File for Import

```
set returnDict [dict create \
GT_SETTINGS_1 [dict create \
GT_TYPE GTY \
GT_DIRECTION DUPLEX \
LRO_SETTINGS [dict create GT_DIRECTION DUPLEX GT_TYPE GTY INS_LOSS_NYQ 20
INTERNAL_PRESET None OOB_ENABLE false PCIE_ENABLE false PCIE_USERCLK2_FREQ
250 PCIE_USERCLK_FREQ 250 PRESET None RESET_SEQUENCE_INTERVAL 0
RXPROGDIV_FREQ_ENABLE false RXPROGDIV_FREQ_SOURCE LCPLL RXPROGDIV_FREQ_VAL
322.265625 RXRECCLK_FREQ_ENABLE false RXRECCLK_FREQ_VAL 0 RX_64B66B_CRC
false RX_64B66B_DECODER false RX_64B66B_DESCRAMBLER false
RX_ACTUAL_REFCLK_FREQUENCY 156.250000000000 RX_BUFFER_BYPASS_MODE Fast_Sync
RX_BUFFER_BYPASS_LANE MULTI RX_BUFFER_MODE 1
RX_BUFFER_RESET_ON_CB_CHANGE ENABLE RX_BUFFER_RESET_ON_COMMAALIGN DISABLE
RX_BUFFER_RESET_RATE_CHANGE ENABLE RX_CB_DISP_0_0 false RX_CB_DISP_0_1
false RX_CB_DISP_0_2 false RX_CB_DISP_0_3 false RX_CB_DISP_1_0 false
RX_CB_DISP_1_1 false RX_CB_DISP_1_2 false RX_CB_DISP_1_3 false RX_CB_K_0_0
false RX_CB_K_0_1 false RX_CB_K_0_2 false RX_CB_K_0_3 false RX_CB_K_1_0
false RX_CB_K_1_1 false RX_CB_K_1_2 false RX_CB_K_1_3 false RX_CB_LEN_SEQ 1
RX_CB_MASK_0_0 false RX_CB_MASK_0_1 false RX_CB_MASK_0_2 false
RX_CB_MASK_0_3 false RX_CB_MASK_1_0 false RX_CB_MASK_1_1 false
RX_CB_MASK_1_2 false RX_CB_MASK_1_3 false RX_CB_MAX_LEVEL 1 RX_CB_MAX_SKEW
1 RX_CB_NUM_SEQ 0 RX_CB_VAL_0_0 00000000 RX_CB_VAL_0_1 00000000
RX_CB_VAL_0_2 00000000 RX_CB_VAL_0_3 00000000 RX_CB_VAL_1_0 00000000
RX_CB_VAL_1_1 00000000 RX_CB_VAL_1_2 00000000 RX_CB_VAL_1_3 00000000
RX_CC_DISP_0_0 false RX_CC_DISP_0_1 false RX_CC_DISP_0_2 false
RX_CC_DISP_0_3 false RX_CC_DISP_1_0 false RX_CC_DISP_1_1 false
RX_CC_DISP_1_2 false RX_CC_DISP_1_3 false RX_CC_KEEP_IDLE DISABLE
RX_CC_K_0_0 false RX_CC_K_0_1 false RX_CC_K_0_2 false RX_CC_K_0_3 false
RX_CC_K_1_0 false RX_CC_K_1_1 false RX_CC_K_1_2 false RX_CC_K_1_3 false
RX_CC_LEN_SEQ 1 RX_CC_MASK_0_0 false RX_CC_MASK_0_1 false RX_CC_MASK_0_2
false RX_CC_MASK_0_3 false RX_CC_MASK_1_0 false RX_CC_MASK_1_1 false
RX_CC_MASK_1_2 false RX_CC_MASK_1_3 false RX_CC_NUM_SEQ 0 RX_CC_PERIODICITY
5000 RX_CC_PRECEDENCE ENABLE RX_CC_REPEAT_WAIT 0 RX_CC_VAL
000000000000000000000000000000000000000000000000000000000000000000000000
0000 RX_CC_VAL_0_0 00000000 RX_CC_VAL_0_1 00000000 RX_CC_VAL_0_2 00000000
RX_CC_VAL_0_3 00000000 RX_CC_VAL_1_0 00000000 RX_CC_VAL_1_1 00000000
RX_CC_VAL_1_2 00000000 RX_CC_VAL_1_3 00000000 RX_COMMA_ALIGN_WORD 1
RX_COMMA_DOUBLE_ENABLE false RX_COMMA_MASK 0000000000 RX_COMMA_M_ENABLE
```

```

false RX_COMMA_M_VAL 1010000011 RX_COMMA_PRESET NONE RX_COMMA_P_ENABLE
false RX_COMMA_P_VAL 0101111100 RX_COMMA_SHOW_REALIGN_ENABLE true
RX_COMMA_VALID_ONLY 0 RX_COUPLING AC RX_DATA_DECODING RAW RX_EQ_MODE AUTO
RX_FRACN_ENABLED false RX_FRACN_NUMERATOR 0 RX_FRACN_OVRD false RX_GRAY_BYP
true RX_GRAY_LITTLEENDIAN true RX_HD_EN 0 RX_INT_DATA_WIDTH 32 RX_JTOL_FC
6.1862627 RX_JTOL_LF_SLOPE -20 RX_LINE_RATE 10.3125 RX_OUTCLK_SOURCE
RXOUTCLKPMA RX_PAM_SEL NRZ RX_PLL_TYPE LCPLL RX_PPM_OFFSET 0 RX_PRECODE_BYP
true RX_PRECODE_LITTLEENDIAN false RX_RATE_GROUP A RX_REFCLK_FREQUENCY
156.25 RX_REFCLK_SOURCE R0 RX_SLIDE_MODE OFF RX_SSC_PPM 0 RX_TERMINATION
PROGRAMMABLE RX_TERMINATION_PROG_VALUE 800 RX_USER_DATA_WIDTH 32
TXPROGDIV_FREQ_ENABLE false TXPROGDIV_FREQ_SOURCE LCPLL TXPROGDIV_FREQ_VAL
322.265625 TX_64B66B_CRC false TX_64B66B_ENCODER false TX_64B66B_SCRAMBLER
false TX_ACTUAL_REFCLK_FREQUENCY 156.250000000000 TX_BUFFER_BYPASS_MODE
Fast_Sync TX_BUFFER_MODE 1 TX_BUFFER_RESET_ON_RATE_CHANGE ENABLE
TX_DATA_ENCODING RAW TX_DIFF_SWING_EMPH_MODE CUSTOM TX_FRACN_ENABLED false
TX_FRACN_NUMERATOR 0 TX_FRACN_OVRD false TX_GRAY_BYP true
TX_GRAY_LITTLEENDIAN true TX_HD_EN 0 TX_INT_DATA_WIDTH 32
TX_LANE_DESKEW_HDMI_ENABLE false TX_LINE_RATE 10.3125 TX_OUTCLK_SOURCE
TXOUTCLKPMA TX_PAM_SEL NRZ TX_PIPM_ENABLE false TX_PLL_TYPE LCPLL
TX_PRECODE_BYP true TX_PRECODE_LITTLEENDIAN false TX_RATE_GROUP A
TX_REFCLK_FREQUENCY 156.25 TX_REFCLK_SOURCE R0 TX_USER_DATA_WIDTH 32 ] \
]]

```

Additional Resources and Legal Notices

Finding Additional Documentation

Technical Information Portal

The AMD Technical Information Portal is an online tool that provides robust search and navigation for documentation using your web browser. To access the Technical Information Portal, go to <https://docs.amd.com>.

Documentation Navigator

Documentation Navigator (DocNav) is an installed tool that provides access to AMD Adaptive Computing documents, videos, and support resources, which you can filter and search to find information. To open DocNav:

- From the AMD Vivado™ IDE, select **Help** → **Documentation and Tutorials**.
- On Windows, click the **Start** button and select **Xilinx Design Tools** → **DocNav**.
- At the Linux command prompt, enter `docnav`.

Note: For more information on DocNav, refer to the *Documentation Navigator User Guide* ([UG968](#)).

Design Hubs

AMD Design Hubs provide links to documentation organized by design tasks and other topics, which you can use to learn key concepts and address frequently asked questions. To access the Design Hubs:

- In DocNav, click the **Design Hubs View** tab.
- Go to the [Design Hubs](#) web page.

Support Resources

For support resources such as Answers, Documentation, Downloads, and Forums, see [Support](#).

References

These documents provide supplemental material useful with this guide:

1. Versal Adaptive SoC Transceivers Wizard LogiCORE IP Product Guide ([PG331](#))
2. Vivado Design Suite User Guide: Designing IP Subsystems using IP Integrator ([UG994](#))
3. Vivado Design Suite User Guide: Designing with IP ([UG896](#))
4. Vivado Design Suite User Guide: Getting Started ([UG910](#))
5. Vivado Design Suite User Guide: Logic Simulation ([UG900](#))
6. Vivado Design Suite User Guide: I/O and Clock Planning ([UG899](#))
7. Vivado Design Suite User Guide: Programming and Debugging ([UG908](#))
8. Versal Adaptive SoC GTY and GTYP Transceivers Architecture Manual ([AM002](#))
9. Vivado Design Suite Tutorial: Logic Simulation ([UG937](#))
10. Vivado Design Suite User Guide: Release Notes, Installation, and Licensing ([UG973](#))
11. Versal Adaptive SoC GTM Transceivers Architecture Manual ([AM017](#))
12. Versal Adaptive SoC Transceiver Wizard to Versal Adaptive SoC Transceiver Subsystem Design Migration Guide ([GitHub link](#))

Revision History

The following table shows the revision history for this document.

Section	Revision Summary
11/13/2024 Version 1.0	
Chapter 3: Product Specification	Updated information on multiple quad configuration support.
Unsupported Features	Updated this section.
Wrapper Configuration Tab	Updated this section.
Chapter 5: Example Design	Updated this section.
Sample Tcl File for Import	Added this section.

Section	Revision Summary
References	Updated this section.
06/13/2024 Version 1.0	
Initial release.	N/A

Please Read: Important Legal Notices

The information presented in this document is for informational purposes only and may contain technical inaccuracies, omissions, and typographical errors. The information contained herein is subject to change and may be rendered inaccurate for many reasons, including but not limited to product and roadmap changes, component and motherboard version changes, new model and/or product releases, product differences between differing manufacturers, software changes, BIOS flashes, firmware upgrades, or the like. Any computer system has risks of security vulnerabilities that cannot be completely prevented or mitigated. AMD assumes no obligation to update or otherwise correct or revise this information. However, AMD reserves the right to revise this information and to make changes from time to time to the content hereof without obligation of AMD to notify any person of such revisions or changes. THIS INFORMATION IS PROVIDED "AS IS." AMD MAKES NO REPRESENTATIONS OR WARRANTIES WITH RESPECT TO THE CONTENTS HEREOF AND ASSUMES NO RESPONSIBILITY FOR ANY INACCURACIES, ERRORS, OR OMISSIONS THAT MAY APPEAR IN THIS INFORMATION. AMD SPECIFICALLY DISCLAIMS ANY IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, OR FITNESS FOR ANY PARTICULAR PURPOSE. IN NO EVENT WILL AMD BE LIABLE TO ANY PERSON FOR ANY RELIANCE, DIRECT, INDIRECT, SPECIAL, OR OTHER CONSEQUENTIAL DAMAGES ARISING FROM THE USE OF ANY INFORMATION CONTAINED HEREIN, EVEN IF AMD IS EXPRESSLY ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

AUTOMOTIVE APPLICATIONS DISCLAIMER

AUTOMOTIVE PRODUCTS (IDENTIFIED AS "XA" IN THE PART NUMBER) ARE NOT WARRANTED FOR USE IN THE DEPLOYMENT OF AIRBAGS OR FOR USE IN APPLICATIONS THAT AFFECT CONTROL OF A VEHICLE ("SAFETY APPLICATION") UNLESS THERE IS A SAFETY CONCEPT OR REDUNDANCY FEATURE CONSISTENT WITH THE ISO 26262 AUTOMOTIVE SAFETY STANDARD ("SAFETY DESIGN"). CUSTOMER SHALL, PRIOR TO USING OR DISTRIBUTING ANY SYSTEMS THAT INCORPORATE PRODUCTS, THOROUGHLY TEST SUCH SYSTEMS FOR SAFETY PURPOSES. USE OF PRODUCTS IN A SAFETY APPLICATION WITHOUT A SAFETY DESIGN IS FULLY AT THE RISK OF CUSTOMER, SUBJECT ONLY TO APPLICABLE LAWS AND REGULATIONS GOVERNING LIMITATIONS ON PRODUCT LIABILITY.

Copyright

© Copyright 2024 Advanced Micro Devices, Inc. AMD, the AMD Arrow logo, Versal, Vivado, and combinations thereof are trademarks of Advanced Micro Devices, Inc. Other product names used in this publication are for identification purposes only and may be trademarks of their respective companies.