



Titanium Interfaces

User Guide

UG-TiINTF-v2.7
February 2023
www.efinixinc.com



Contents

About the Interface Designer.....	iv
Chapter 1: Get Oriented.....	5
Interface Blocks.....	8
Package/Interface Support Matrix.....	9
Interface Block Connectivity.....	10
Designing an Interface.....	11
Create or Delete a Block.....	11
Using the Resource Assigner.....	12
Resource View.....	13
Importing and Exporting Assignments.....	13
Viewing the Package Pinout.....	16
Selecting a Pin.....	17
Browsing for Pins.....	18
Interface Designer Output Files.....	18
Scripting an Interface Design.....	19
Chapter 2: Device Settings.....	20
Configuration Interface.....	20
Enable Internal Configuration.....	20
About SEU Detection.....	20
Enable SEU Detection.....	21
SEU Detection Circuitry.....	22
Design Check: Configuration Messages.....	23
I/O Banks Interface.....	24
Titanium I/O Banks.....	24
Dynamic Voltage Support.....	25
Design Check: I/O Bank Messages.....	27
Chapter 3: Clock and Control Networks.....	28
Clock Sources that Drive the Global and Regional Networks.....	30
Configuring the Dynamic Clock Multiplexers.....	31
Driving both the Global and Regional Networks.....	31
Design Check: Clock Control Messages.....	32
Chapter 4: DDR Interface.....	35
About the DDR DRAM Interface.....	35
DDR Interface Designer Settings.....	41
Chapter 5: GPIO Interface.....	44
Types of GPIO.....	44
Features for HVIO and HSIO Configured as GPIO.....	46
Double-Data I/O.....	47
Programmable Delay Chains.....	48
About the HVIO Interface.....	48
About the HSIO Interface.....	50
HSIO Configured as GPIO.....	51
Using the GPIO Block.....	54
Using the GPIO Bus Block.....	57
Create a TX Serializer Interface.....	58
Create a RX Deserializer Interface.....	59
Design Check: GPIO Messages.....	60
Chapter 6: LVDS Interface.....	71

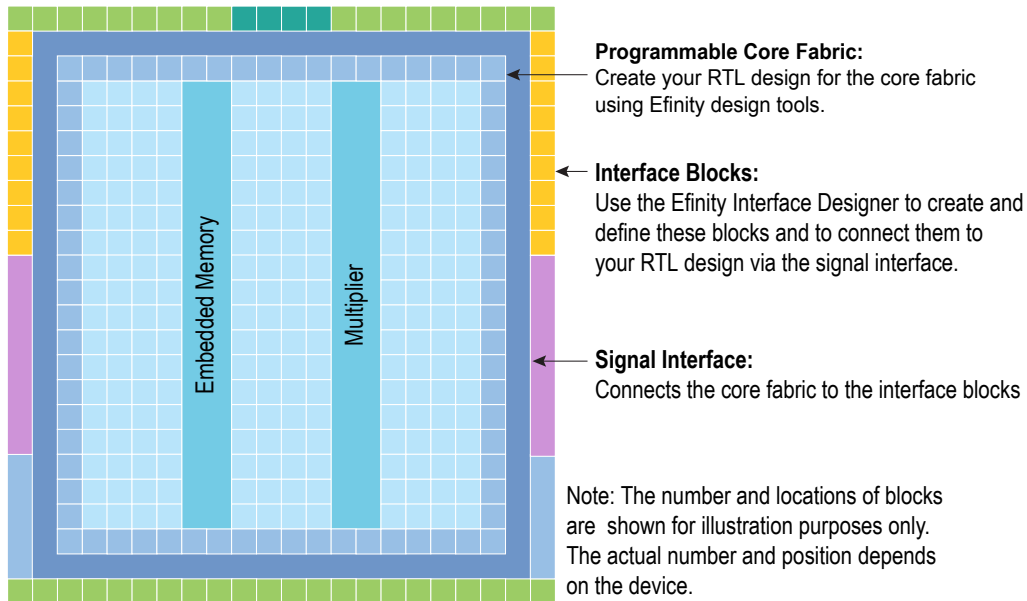
HSIO Configured as LVDS.....	71
Using the LVDS Block.....	77
Create an LVDS TX Interface.....	79
Create an LVDS RX Interface.....	81
Design Check: LVDS Errors and Warnings.....	82
Chapter 7: HyperRAM Interface.....	90
About the HyperRAM.....	90
Using the HyperRAM Interface.....	91
Chapter 8: JTAG User TAP Interface.....	93
JTAG Mode.....	93
Using the JTAG User TAP Block.....	96
Design Check: JTAG User Tap Errors and Warnings.....	97
Chapter 9: MIPI RX/TX Lane Interface.....	98
HSIO Configured as MIPI Lane.....	98
MIPI Groups by Package.....	103
Using the MIPI TX Lane or MIPI RX Lane Block.....	108
Create a MIPI TX Interface.....	109
Create a MIPI RX Interface.....	111
Design Check: MIPI Lane Messages.....	112
Chapter 10: MIPI D-PHY Interface.....	115
MIPI RX D-PHY.....	115
MIPI TX D-PHY.....	118
MIPI DPHY TX Interface Designer Settings.....	123
MIPI DPHY RX Interface Designer Settings.....	125
Chapter 11: PLL.....	127
About the PLL Interface.....	127
Using the PLL V3 Block.....	129
Using the PLL Clock Calculator.....	130
Understanding PLL Phase Shifting.....	131
Manually Configuring the PLL.....	131
Implementing a Zero-Delay Buffer.....	132
Design Check: PLL Errors.....	133
Chapter 12: Oscillator.....	138
Using the Oscillator Block.....	138
Design Check: Oscillator Errors.....	138
Chapter 13: SPI Flash Interface.....	140
About the SPI Flash Memory.....	140
Using the SPI Flash Interface.....	141
Chapter 14: Interface Floorplans.....	142
Icon Reference.....	146
Revision History.....	147

About the Interface Designer

Titanium FPGAs wrap a Quantum™-accelerated core with a periphery that sends signals out to the device pins. The core contains the logic, embedded memory, and multipliers. The device periphery includes blocks such as GPIO pins, LVDS, MIPI, DDR, and PLLs.

The tools in the Efinity® main window help you design the logic portion of your design. You use the Efinity Interface Designer to build the peripheral portion of your design.

Figure 1: Conceptual View of Interface Blocks



Get Oriented

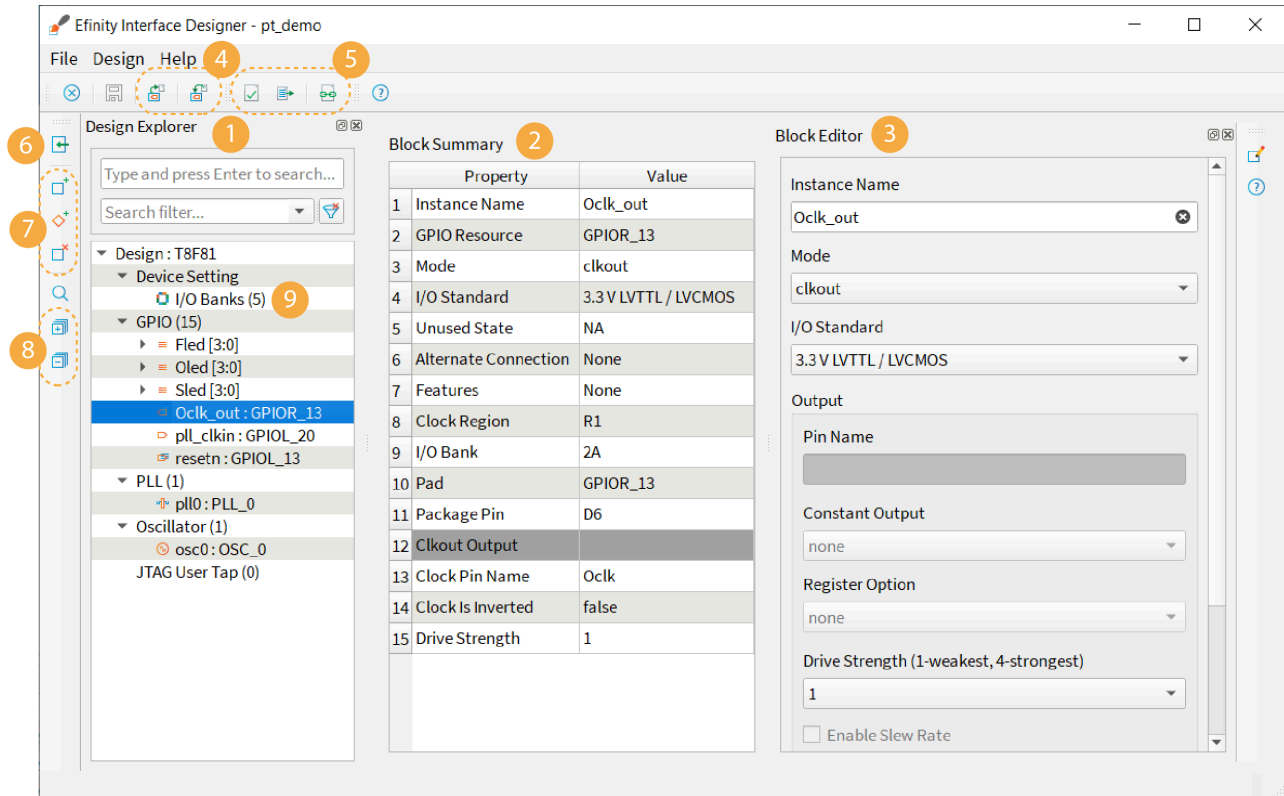
Contents:

- **Interface Blocks**
 - **Package/Interface Support Matrix**
 - **Interface Block Connectivity**
 - **Designing an Interface**
 - **Create or Delete a Block**
 - **Using the Resource Assigner**
 - **Viewing the Package Pinout**
 - **Interface Designer Output Files**
 - **Scripting an Interface Design**
-

The Interface Designer has four main sections:

- *Design Explorer*—Provides a list view of the interface blocks you have in your design organized by block type. It also includes device-wide settings for the I/O banks and configuration options. Select a block to display its summary and editor.
- *Block Summary*—Displays the current settings for the selected block.
- *Block Editor*—Provides options and settings for the selected block. The editor may have more than one tab, depending on the block.
- *Resource Assigner*—Provides an easy, tabular method for assigning resources. View by instance (default) or resource.

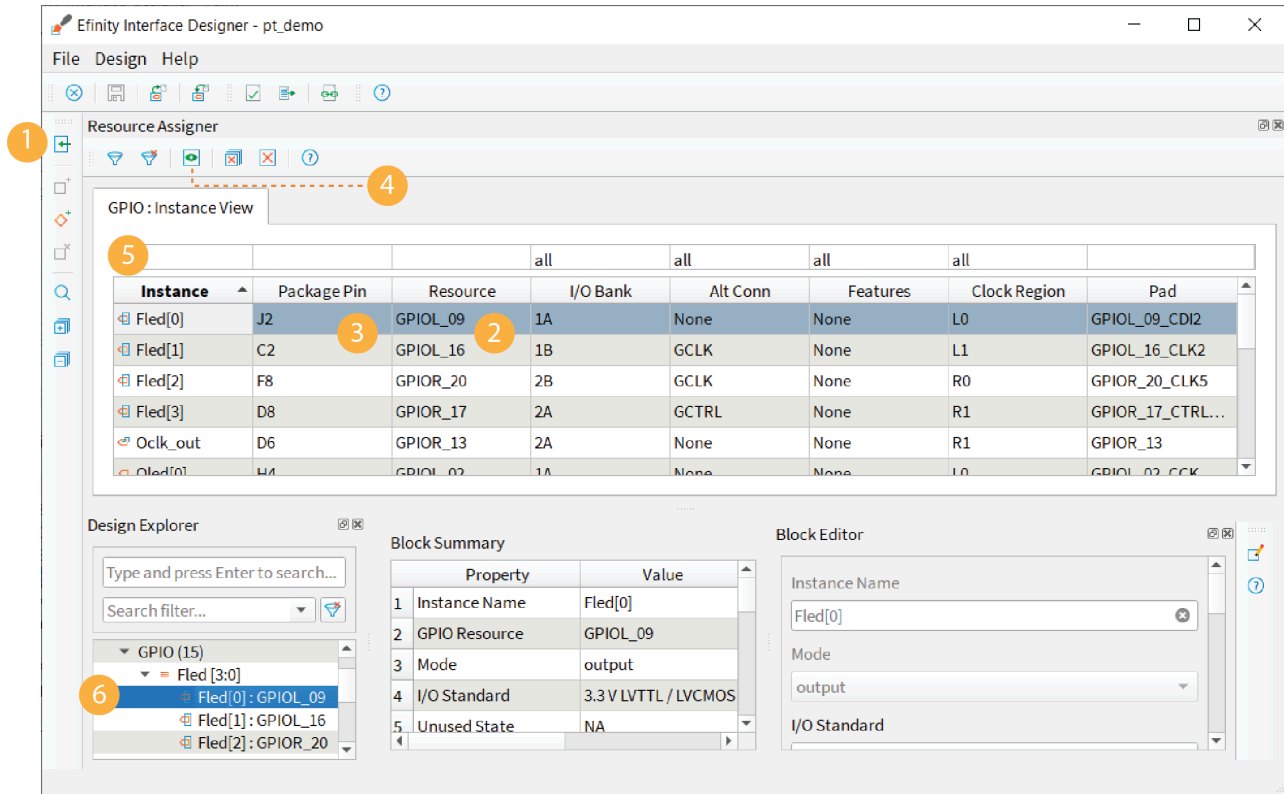
Figure 2: Interface Designer

**Notes:**

1. The Design Explorer shows the interface blocks in your design. They are organized by block type.
2. The block summary shows the settings for the block selected in the Design Explorer.
3. Use the Block Editor to add or change settings for the interface block.
4. You can import or export GPIO resource assignments using a **.csv** or **.isf** file.
5. Use the project management tools to perform design checks, view reports, generate constraints, etc.
6. Click Show/Hide Resource Assigner to toggle a tabular view of assignments.
7. Use the block tools to add or delete blocks and buses.
8. Expand or collapse the Design Explorer folders.
9. The number in parentheses shows the number of used blocks.

When you first open the Interface Designer for your project, the Design Explorer shows the Device Settings folder (with default settings) and empty folders for the interface blocks your chosen device supports. You need to add blocks as required for your design.

Figure 3: Resource Assigner



Notes:

1. Show or hide the Resource Assigner.
2. Double-click in the Resource cell to open the list of available resources.
3. Double-click in the Package Pin cell to open the list of available pins.
4. Click the Switch View button to toggle between Instance View and Resource View.
5. Type in the filter cell above the column you want to filter.
6. Selecting a block in the Design Explorer highlights it in the Resource Assigner.

Interface Blocks

Titanium FPGAs support a variety of interface blocks. The available blocks differ depending on which FPGA you target and the package. You need to assign a resource for every block you use.

The following table describes the interface blocks supported in the Efinity® software version 2022.2.

Table 1: Titanium Interface Block Support by Package

Interface	Ti35	Ti60	Ti90	Ti120	Ti180
DDR	-	-	J361, M361, J484, M484, F529, G529	J361, M361, J484, M484, F529, G529	J361, M361, J484, M484, F529, G529
GPIO	All	All	All	All	All
GPIO bus	All	All	All	All	All
I/O bank	All	All	All	All	All
JTAG User TAP	All	All	All	All	All
LVDS TX LVDS RX Bidirectional LVDS	All	All	All	All	All
MIPI DPHY	-	-	J361, M361, J484, L484, M484	J361, M361, J484, L484, M484	J361, M361, J484, L484, M484
MIPI TX Lane MIPI RX Lane	All	All	All	All	All
PLL (V3)	All	All	All	All	All
Oscillator	All	All	All	All	All

All interface blocks have an instance name that must be a unique identifier. When you add a new block, the Interface Designer gives the block a unique default name, which you can change.



Note: After you re-name the block, press Enter or click Save to save the name.

Pin names are the top-level ports of the design implemented in the core that connect to the interface block. These names must be legal Verilog HDL or VHDL identifiers.

Package/Interface Support Matrix

Some interfaces are only available in certain packages. The following table describes which interfaces are supported in specific FPGA/package combinations for the Efinity® software v2022.2. Refer to the data sheet for package-dependent resources.

Table 2: Titanium Interface/Package Combinations Supported in Efinity® Software v2022.2

Package	Ti35	Ti60	Ti90, Ti120, Ti180
W64			
F100S3F2, F225			
J361, M361, J484, M484			
L484			
F529, G529			

Titanium Family Legend:

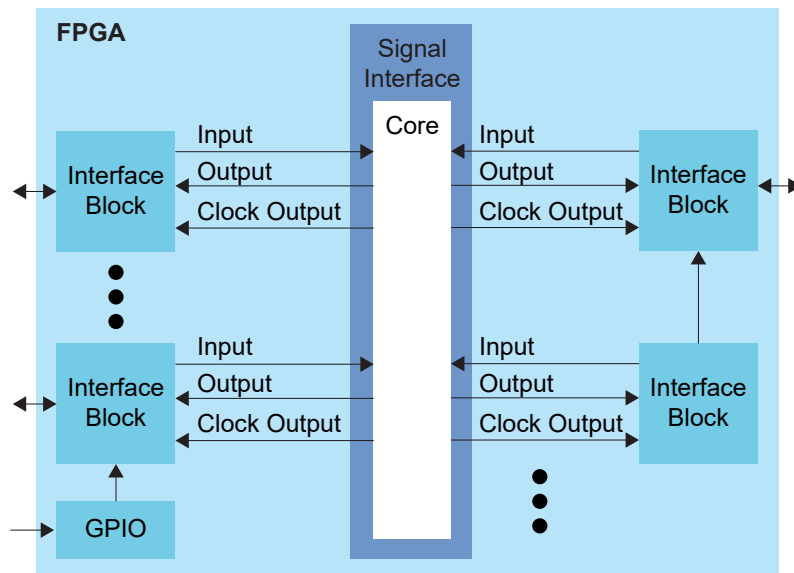


Interface Block Connectivity

The FPGA core fabric connects to the interface blocks through a signal interface. The interface blocks then connect to the package pins. The core connects to the interface blocks using three types of signals:

- *Input*—Input data or clock to the FPGA core
- *Output*—Output from the FPGA core
- *Clock output*—Clock signal from the core clock tree

Figure 4: Interface Block and Core Connectivity



GPIO blocks are a special case because they can operate in several modes. For example, in alternate mode the GPIO signal can bypass the signal interface and directly feed another interface block. So a GPIO configured as an alternate input can be used as a PLL reference clock without going through the signal interface to the core.

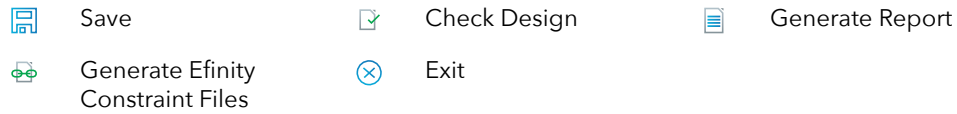
When designing for Titanium FPGAs, you create an RTL design for the core and also configure the interface blocks. From the perspective of the core, outputs from the core are inputs to the interface block and inputs to the core are outputs from the interface block.

The Efinity netlist always shows signals from the perspective of the core, so some signals do not appear in the netlist:

- GPIO used as reference clocks are not present in the RTL design, they are only visible in the interface block configuration of the Efinity® Interface Designer.
- The FPGA clock tree is connected to the interface blocks directly. Therefore, clock outputs from the core to the interface are not present in the RTL design, they are only part of the interface configuration (this includes GPIO configured as output clocks).

The following sections describe the different types of interface blocks in the Titanium. Signals and block diagrams are shown from the perspective of the interface, not the core.

Designing an Interface



Designing your interface is straightforward: add interface blocks, configure them, and then generate reports and constraints. The Efinity software uses the constraints during compilation to connect signals from the core to your interface.



Note: Refer to [Create or Delete a Block](#) on page 11 and [Interface Blocks](#) on page 8 for instructions on adding blocks and configuring them.

During the design process, you can generate reports, which are available in the Efinity® Results tab. When you generate reports, the software also saves your design.

Use the design checker to check the interface for errors and to ensure that your settings are valid. The Interface designer displays design issues in the message viewer window. You can also export design issues (**Design > Export Design Issues**) to generate a comma separated values (.csv) report to view the issues in a spreadsheet application. When you run the design checker, the software automatically saves your interface.

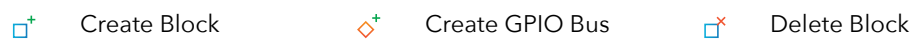
When you are done configuring your interface, click the Export Efinity Constraints Files button to export the interface constraints to your project. The software saves the design, checks it for errors, generates the interface reports and the interface constraint files.

Click Exit to close the Interface Designer and return to the Efinity® main window.



Note: You can leave the Interface Designer open while running the Efinity® software. However, if you make changes to the Efinity project, the Interface Designer is not updated until the next time you launch it.

Create or Delete a Block



To create a block:

1. Select the folder for the block type you want to create.
2. Click the Create Block button.

To create a GPIO bus, click the GPIO folder and then click the Create GPIO Bus button.

To delete a block, select the block name and click the Delete Block button.

Tip: Right-clicking a folder name opens a context-sensitive menu. From there you can choose **Create Block** (and **Create Bus** for GPIO).

Using the Resource Assigner



The Resource Assigner provides a tabular view of all GPIO resources in your chosen FPGA and information about them, such as whether they are used, the I/O bank, pad, and package pin, and the instance assigned to the resource.

- The **GPIO: Instance View** shows all GPIO instances in your project.
- The **GPIO: Resource View** shows all GPIO, LVDS, and MIPI RX or TX lane resources and the resources to which you assigned them.



Note: In the Efinity® software v2021.1, you can only view the resources used for LVDS and MIPI lanes in the Resource Assigner. You cannot change or assign resources in this view.

To assign a resource:

1. Open the Resource Assigner by clicking the Show/Hide Resource Assigner button. The software opens to the Instance View, which lists all instances in the design.



Note: Click Switch View to toggle between instance view and resource view.

2. In instance view, you can assign pins or resources to the instance. Double-click in the table cell for the item you want to assign. The software displays a drop-down list of available selections.
3. Select an unused resource, instance, or pin.



Note: If you select a used resource, instance, or pin, the software makes the new assignment, which replaces the previous assignment.

4. Press Enter.



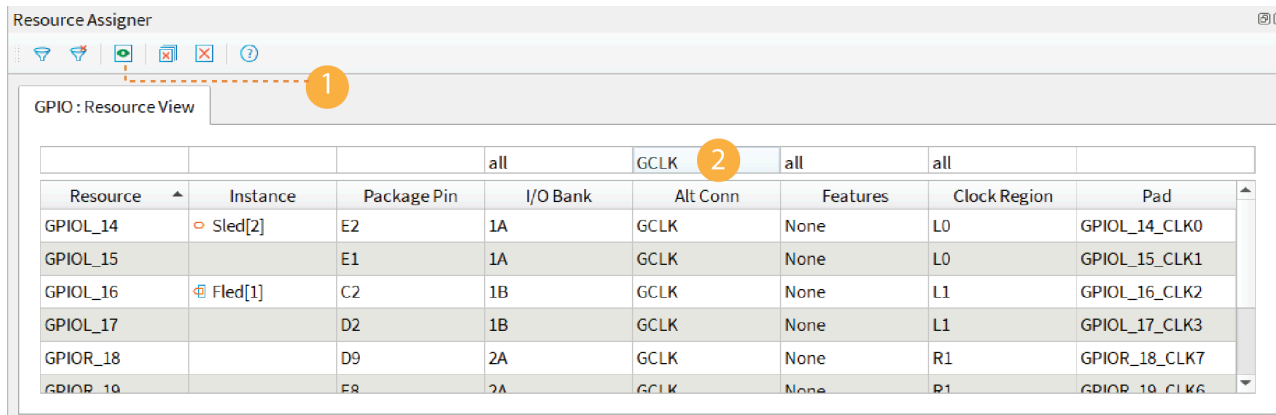
Note: Titanium: When using HSIO pins as GPIO, make sure to leave at least 1 pair of unassigned HSIO pins between any GPIO and HSIO pins in the same bank. This separation reduces noise. The Efinity software issues an error if you do not leave this separation.

Resource View

When assigning GPIO, sometimes you want to know which resource can be used as a global clock, global control, or other special function. You can look it up in the pin table for the FPGA and package you are targeting, but an easier way is to use the Resource View in the Resource Assigner.

1. Click the Switch View button to open the Resource View.
2. Double-click in the filter box above the **Alt Conn** column and choose the connection type, for example, **GCLK**.

Figure 5: Resource View



Importing and Exporting Assignments

Although it is nice to use a GUI for adding blocks, in some cases it may be easier to use another format. The Interface Designer lets you import and export assignments using an Interface Scripting File (.isf) or comma separated values (.csv) file.

When the software reads an imported .isf, it processes the entire imported file and shows any issues it found. The import only fails for catastrophic errors. The software:

- Creates new instances defined in the file that do not already exist in the GUI
- Overwrites assignments for existing instances with settings from the file
- Does not delete instances that are in the GUI but were not defined in the file

When the software reads an imported .csv file, it compares the imported assignments to the original assignments and reports any issues. If the software finds warnings, it displays them but allows you to finish the import. If it finds errors, it will not finish the import. When importing, the software:

- Deletes instances that you removed
- Creates newly defined instances
- Replaces instances you renamed with the new name



Learn more: For help understanding messages, refer to the "Design Check" topics in the Titanium Interfaces User Guide. These topics describe the messages the Interface Designer generates and gives suggestions on how to fix errors and warnings.

Interface Scripting File

The Interface Scripting File (.isf) contains all of the Python API commands to re-create your interface. You can export your design to an .isf, manipulate the file, and then re-import it back into the Efinity® software. Additionally, you can write your own .isf if desired.

In addition to using the API, you can export and import an .isf in the Interface Designer GUI. Click the Import GPIO or Export GPIO buttons and choose **Interface Scripting File (.isf)** under **Format**.

Example: Example Interface Scripting File

```
# Efinity Interface Configuration
# Version: 2020.M.138
# Date: 2020-06-26 14:22
#
# Copyright (C) 2017 - 2020 Efinix Inc. All rights reserved.
#
# Device: T8F81
# Package: 81-ball FBGA (final)
# Project: pt_demo
# Configuration mode: active (x1)
# Timing Model: C2 (final)

# Create instance
design.create_output_gpio("Fled",3,0)
design.create_inout_gpio("Sled",3,0)
design.create_output_gpio("Oled",3,0)
design.create_clockout_gpio("Oclk_out")
design.create_pll_input_clock_gpio("pll_clkkin")
design.create_global_control_gpio("resetsn")

# Set property, non-defaults
design.set_property("Fled","OUT_REG","REG")
design.set_property("Fled","OUT_CLK_PIN","Fclk")
design.set_property("Sled[0]","IN_PIN","")
design.set_property("Sled[0]","OUT_PIN","Sled[0]")
design.set_property("Sled[1]","IN_PIN","")
design.set_property("Sled[1]","OUT_PIN","Sled[1]")
design.set_property("Sled[2]","IN_PIN","")
design.set_property("Sled[2]","OUT_PIN","Sled[2]")
design.set_property("Sled[3]","IN_PIN","")
design.set_property("Sled[3]","OUT_PIN","Sled[3]")
design.set_property("Oclk_out","OUT_CLK_PIN","Oclk")

# Set resource assignment
design.assign_pkg_pin("Fled[0]","J2")
design.assign_pkg_pin("Fled[1]","C2")
design.assign_pkg_pin("Fled[2]","F8")
design.assign_pkg_pin("Fled[3]","D8")
design.assign_pkg_pin("Sled[0]","E6")
design.assign_pkg_pin("Sled[1]","G4")
design.assign_pkg_pin("Sled[2]","E2")
design.assign_pkg_pin("Sled[3]","G9")
design.assign_pkg_pin("Oled[0]","H4")
design.assign_pkg_pin("Oled[1]","J4")
design.assign_pkg_pin("Oled[2]","A5")
design.assign_pkg_pin("Oled[3]","C5")
design.assign_pkg_pin("Oclk_out","D6")
design.assign_pkg_pin("pll_clkkin","C3")
design.assign_pkg_pin("resetsn","F1")
```

.csv File for GPIO Blocks

For larger designs with lots of GPIO, it can be simpler to use a spreadsheet application to make assignments. The Resource Assigner allows you to import and export GPIO block assignments using a comma separated values (.csv) file. The .csv file includes the package pin and pad name, the instance name, and the mode. You can use this method for any type of GPIO, including LVDS pins used as GPIO or HSIO pins used as GPIO.

Table 3: Example GPIO .csv File

Package Pin-Pad Name	Instance Name	Mode
G5-GPIOL_00		
J4-GPIOL_01_SS_N		
H4-GPIOL02_CCK		
G4-GPIOL_03_CDI4	led[0]	output
F4-GPIOL04_CDI0	led[1]	output
J3-GPIOL_05_CDI5	rstn	input
H3-GPIOL_06_CDI1		
...		
(1)	led[6]	inout

When working with the .csv file:

- Add your assignments to the **Instance Name** and **Mode** columns.
- Do not modify the package pin-pad names.
- For the mode, specify: input, output, inout, clkout, or none



Note: You cannot make advanced settings such as alternate connections or registering. To make these settings, use the Block Editor.

When the software reads an imported .csv file, it performs a comparison between the .csv assignments and the original GPIO block assignments and reports any issues. If the software finds warnings, it displays them but allows you to finish the import. If it finds errors, it will not finish the import. When importing, the software:

- Deletes instances that you removed
- Creates newly defined instances
- Replaces instances you renamed with the new name

⁽¹⁾ Unassigned instances have a blank field for the Package Pin-Pad Name column.

Viewing the Package Pinout

The Package Planner provides a visual representation of the FPGA package pins. Each pin is color coded by function (such as GPIO, configuration, power, etc.) letting you easily see which package pin has which function. Additionally, you can highlight I/O banks, PLL reference clocks, global clocks, and global controls so you can quickly find a specific pin that has the feature you need. This tool is helpful when planning how to map the signals in your design to package pins.

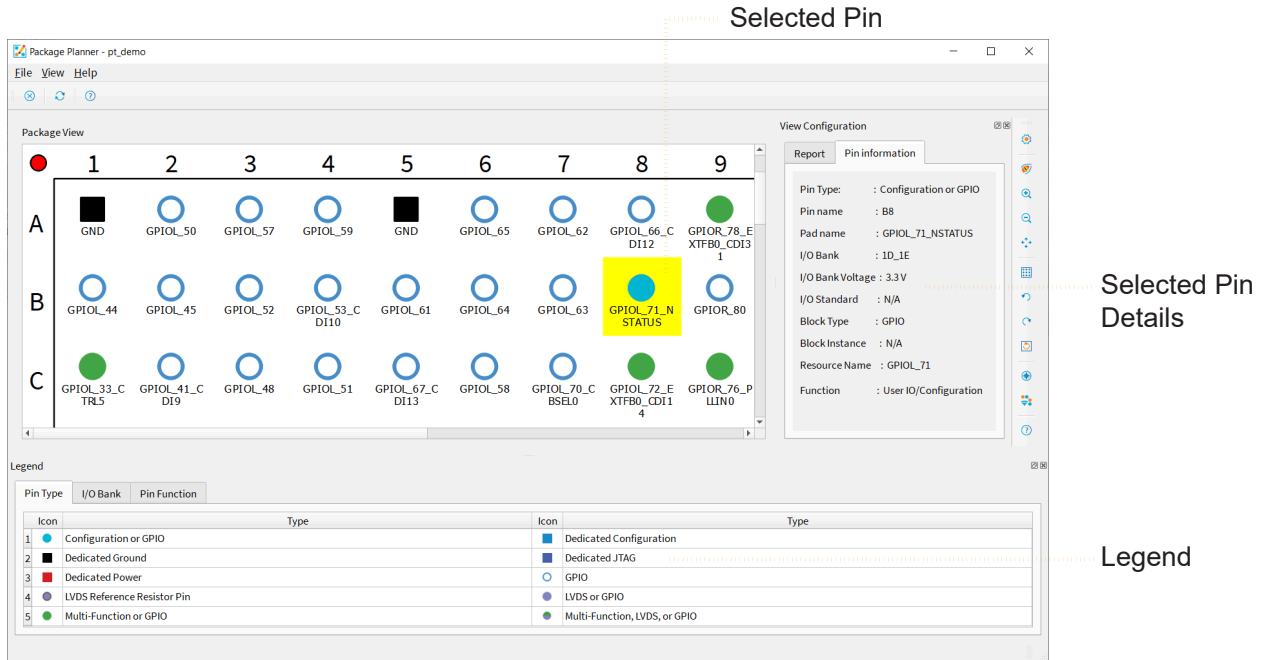
Figure 6: Package Planner

The screenshot shows the Package Planner application window titled "Package Planner - pt_demo". The main area displays a "256-ball FBGA Pinout Diagram" for device "T20F256". The pinout is shown as a grid with columns numbered 1-16 and rows lettered A-T. Each pin is represented by a colored circle with a label. A "Refresh the Pinout" button is located on the left side of the window. On the right, a "View Configuration" panel is open, showing options to toggle various features like "I/O bank group", "PLL Reference Clock", and "Global Control". A legend on the far right lists actions such as "Toggle Pin Configuration", "Zoom In", and "Open Help".

Selecting a Pin

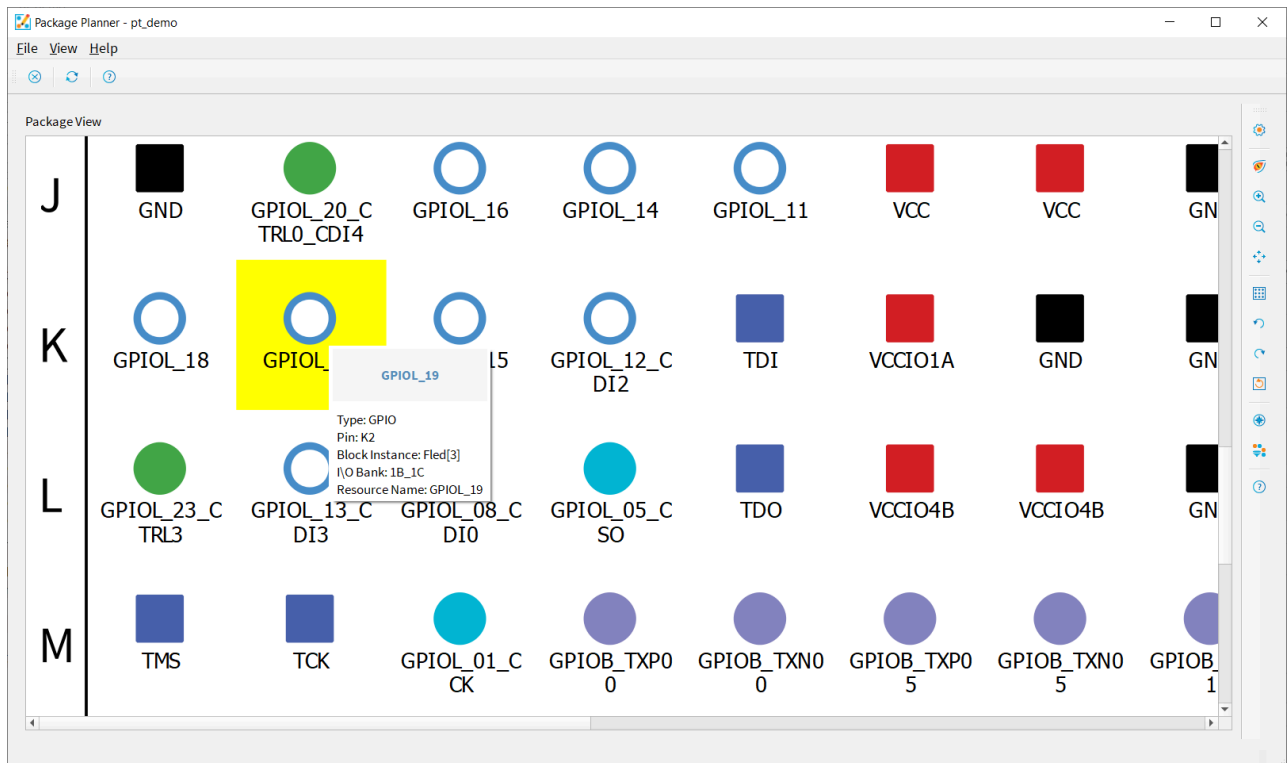
Click a pin in the pinout to highlight it. The **Pin Information** tab opens to show the details about the selected pin. Open the Legend to view the meaning of the pins' color coding.

Figure 7: Selected Pin



You can also hover over a pin for a quick view of the pin details.

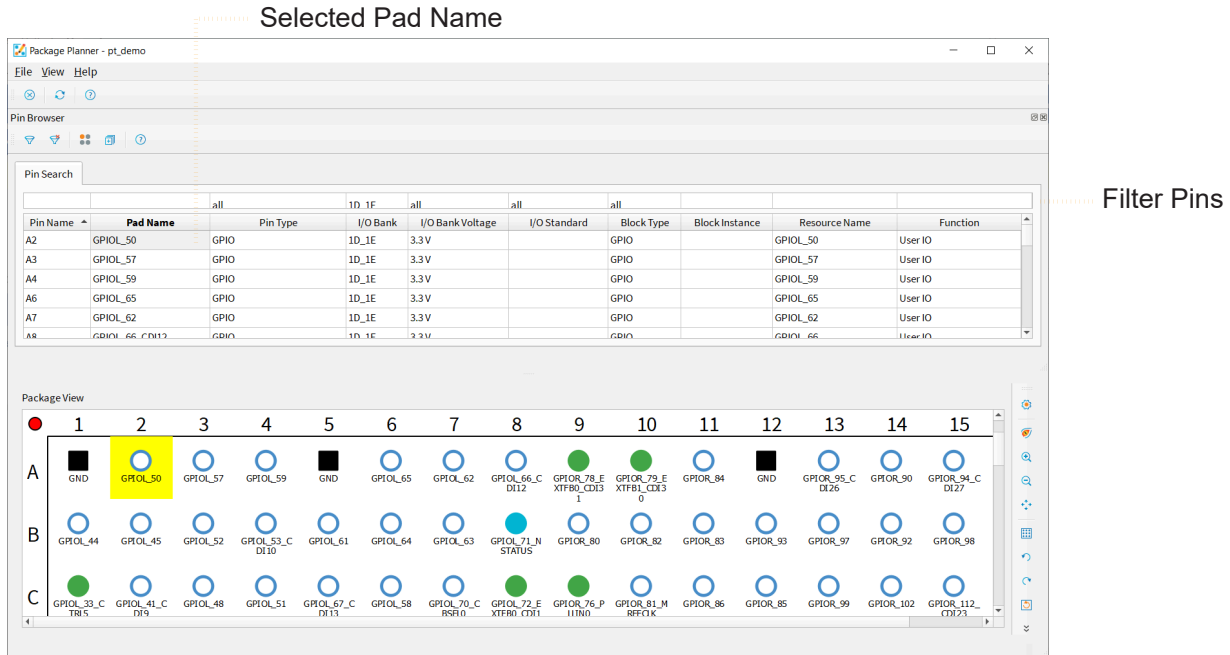
Figure 8: Pin Quick View



Browsing for Pins

The Package Planner has a **Pin Browser**, which has a table view similar to the **Resource Assigner**. You can filter pins and then select them in the **Pin Browser**. The selected pin is highlighted in the pinout.

Figure 9: Browsing for Pins



Interface Designer Output Files

When you generate constraint files, the Interface Designer creates the following output files. You can view them in the Interface section of the Result pane.

- **<project name>.interface.csv**—Constrains the FPGA design pins used in the interface between the core and the periphery.
- **<project name>.pt.rpt**—Provides information about the interface.
- **<project name>.pinout.csv**—Contains the board design pinout in CSV format.
- **<project name>.pinout.rpt**—Has the board design pinout in a nicely formatted text file format.
- **<project name>.pt_timing.rpt**—Timing report for the Titanium interface logic.
- **<project name>.pt.sdc**—Template SDC file to constrain the FPGA design pins based on the interface configuration.
- **<project name>_template.v**—Template Verilog HDL file defining the FPGA design pins based on the interface configuration.

Scripting an Interface Design

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics.⁽²⁾ Efinix distributes a copy of Python 3 with the Efinity[®] software to support point tools such as the Debugger and to allow users to write scripts to control compilation.

You use the Efinity[®] Interface Designer to build the peripheral portion of your design, including GPIO, LVDS, PLLs, MIPI RX and TX lanes, and other hardened blocks. Efinix provides a Python 3 API for the Interface Designer to let you write scripts to control the interface design process. For example, you may want to create a large number of GPIO, or target your design to another board, or export the interface to perform analysis. This user guide describes how to use the API and provides a function reference.



Learn more: Refer to the Python web site, www.python.org/doc, for detailed documentation on the language.



Learn more: For more information on using the Python API to script an interface, refer to the [Efinity Interface Designer Python API](#).

⁽²⁾ Source: [What Is Python? Executive Summary](#)

Device Settings

Contents:

- [Configuration Interface](#)
- [Design Check: Configuration Messages](#)
- [I/O Banks Interface](#)
- [Titanium I/O Banks](#)
- [Dynamic Voltage Support](#)
- [Design Check: I/O Bank Messages](#)

The Interface Designer has device-wide settings for I/O banks and configuration.

Configuration Interface

The Configuration device-wide setting lets you control or monitor configuration using the FPGA design implemented in the FPGA core.

Enable Internal Configuration

Efinix® FPGAs have an internal reconfiguration feature that allows you to control reconfiguration of the FPGA from within the FPGA design. Leave this feature disabled unless you want to use internal reconfiguration.

To enable internal reconfiguration:

1. Click **Device Setting > Configuration**.
2. In the Block Editor **Remote Update** tab, turn on **Enable Internal Reconfiguration Interface**.
3. Indicate the name of the clock pin that will control the internal reconfiguration.
4. Define the FPGA pins that the interface uses.
5. Save.



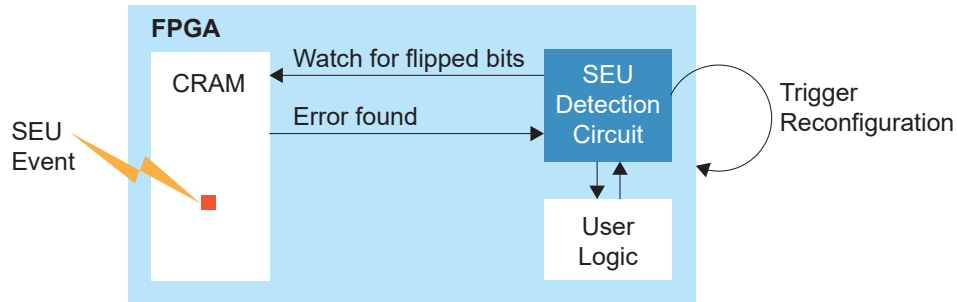
Note: Refer to [AN 010: Using Internal Reconfiguration in Trion and Titanium FPGAs](#) for instructions on how to use this feature.

About SEU Detection

An SEU happens when an environmental factor, such as background radiation, causes a digital circuit to malfunction. For FPGAs, the most frequent (and most worrisome) outcome of an SEU is that a CRAM bit is changed from its programmed value. Designs may not use every CRAM bit in the FPGA, so an SEU may or may not cause the FPGA to malfunction. However, in many situations the safest course of action is to assume that the FPGA's behavior is corrupted until it is reconfigured.

Titanium FPGAs contain built-in circuitry to help detect SEUs. This circuitry periodically monitors the FPGA's CRAM, detects if a CRAM value has changed from the programmed state, and sends status signals to user logic. The user logic can optionally trigger the FPGA to reconfigure using the SEU detection circuitry.

Figure 10: SEU Detection Circuitry



Titanium FPGAs can monitor the CRAM while the FPGA is operating normally in user mode; When the SEU detection circuitry is triggered, it calculates a 32-bit CRC value based on CRAM values and compares it to a CRC computed by the Efinity software and stored in the configuration bitstream. If the values are different, the SEU circuit determines an error has occurred and sends an error signal to the user logic. You can trigger the SEU detection circuitry automatically on a set interval, or manually using a signal.



Note: For Ti35 and Ti60 FPGAs, your design should be in an "idle" state before performing an SEU check.

Enable SEU Detection

To enable the SEU feature in the Efinity software:

1. Click **Device Setting > Configuration**.
2. Click the **SEU Detection** tab.
3. Turn on **Enable SEU Detection**.
4. Choose the mode, **auto** or **manual**.
 - In **auto** mode, you can specify the amount of time between SEU error checks in microseconds. Allowable values are 1 to 1650000.0 (default).
 - In **manual** mode, you specify the pin name that controls when the SEU error check happens.

Tip: For environment that have a higher risk of SEUs, you can set a shorter wait interval. However, the shorter the wait interval, the more power the system consumes. To use less power, choose a longer wait interval. To save even more power, you can use manual mode to only trigger the SEU detection circuitry when conditions require it.

5. Save.



Note: The software issues an error if you turn on SEU detection for Ti60ES FPGAs because they do not support SEU checking.

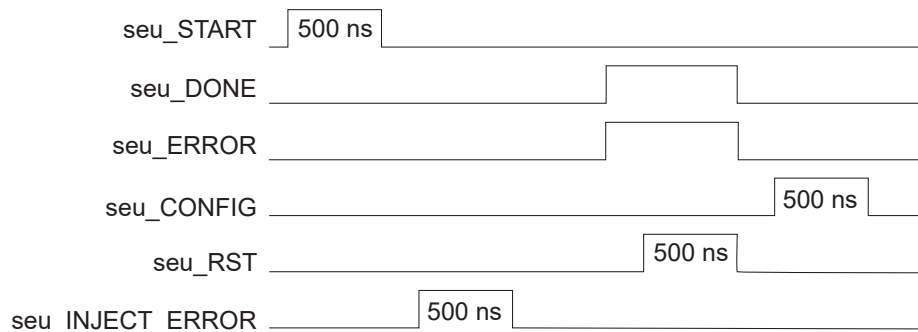
SEU Detection Circuitry

Your user logic connects to the SEU detection circuitry using the following pins.

Table 4: SEU Detection Pins

GUI Option	Default Signal Name	Direction	Description
SEU Start Detection Pin Name	seu_START	Input	Manual mode only. To use this pin to initiate an SEU check, pulse this signal high for a minimum pulse of 500 ns.
Error Injection Pin Name	seu_INJECT_ERROR	Input	This signal forces an SEU error so you can test the how the SEU detection circuitry interacts with your user design during testing and development. To inject an error, pulse this signal high for a minimum pulse of 500 ns. Pull this signal low when you are not using it.
Error Reset Pin Name	seu_RST	Input	This pin sets the Error Status pin to low to clear the error and restart SEU detection monitoring. To reset SEU monitoring, pulse this signal high for a minimum pulse of 500 ns.
Error Status Pin Name	seu_ERROR	Output	If the FPGA detects an SEU, this signal goes high. It does not go low until you toggle the Error Reset pin.
Reconfiguration Pin Name	seu_CONFIG	Input	If the FPGA is using active configuration mode, you can use this pin to trigger the FPGA to reconfigure; the FPGA does not automatically reconfigure when an error is detected. Pulse this signal high for a minimum pulse width of 500 ns. If you are using passive configuration mode or JTAG configuration, you cannot trigger reconfiguration with this signal.
SEU Done Detection Pin Name	seu_DONE	Output	This signal goes high when the SEU detection circuitry has completed calculating the CRC and comparing it to the stored one. If an SEU occurred, this signal stays high until you reset the circuitry.

Figure 11: SEU Signal Waveform



Design Check: Configuration Messages

When you check your design, the Interface Designer applies design rules to your configuration settings. The following tables show some of the error messages you may encounter and explains how to fix them.

configuration_rule_clock (error)

Message	Internal Reconfiguration Interface is enabled but clock pin name is invalid
To fix	Enter a valid clock name.

seu_rule_start (error)

Message	Empty SEU Start Detection pin name found
To fix	Enter a valid start pin name.

seu_rule_interval (error)

Message	Invalid wait interval <#> microsec
To fix	Change the wait interval to one that is in range. Allowable values are 0 - 1677721.

seu_rule_error (error)

Message	Empty Error Status pin name found
To fix	Enter a valid error pin name.

seu_rule_param (error)

Message	Invalid parameters configuration: <feature list>
To fix	One of the parameters you set was incorrect. Review any other errors for details.

seu_rule_sample_device (error)

Message	SEU Detection is not supported in ES device
To fix	You get this error if you try to enable SEU for the Ti60ES. SEU is not supported in this FPGA.

I/O Banks Interface

The I/O Banks setting shows the device I/O banks and the I/O voltage each bank uses. Some I/O banks support multiple I/O standards, and you can specify which standard the bank uses. These settings determine the FPGA pinout requirements and timing values of the interface blocks. Some I/O banks can support multiple I/O standards as long as the I/O voltages of the different standards are compatible.

To set the I/O voltage for a bank:

1. Click **Device Setting > I/O Banks**.
2. In the Block Editor, select the I/O voltage for the bank.
You also select an I/O standard for GPIO blocks. The voltage you select for the I/O bank must be compatible with the settings you choose for any GPIO in this bank.
3. Save.



Note: The I/O banks and their legal configuration are device and package specific. Refer to the data sheet for your chosen FPGA for details on which I/O standards it supports.

Titanium I/O Banks

Efnix FPGAs have input/output (I/O) banks for general-purpose usage. Each I/O bank has independent power pins. The number and voltages supported vary by FPGA and package.

Some I/O banks are merged at the package level by sharing VCCIO pins, these are called merged banks. Merged banks have underscores () between banks in the VCCIO name (e.g., 1B_1C means VCCIO for bank 1B and 1C are connected). Some of the banks in a merged bank may not have available user I/Os in the package. The following table lists banks that have available user I/Os in a package.

Table 5: Titanium I/O Banks by Package for Ti35 and Ti60 FPGAs

Package	I/O Banks	Voltage (V)	Dynamic Voltage Support	DDIO Support	Merged Banks
W64	1A, 1B, 3B	1.2, 1.5, 1.8	-	All	1A_4B, 1B_2A, 2A_3A_3B_4A
F100S3F2	1A, 2A	1.2, 1.5, 1.8 ⁽³⁾	-	All	1A_4B, 2A_2B
	1B, 3A, 3B	1.2, 1.5, 1.8	-	All	3B_4A
	BL	1.8, 2.5, 3.0, 3.3	✓	All	-
F225	BL, TL, TR, BR,	1.8, 2.5, 3.0, 3.3	✓	All	-
	1A, 1B, 2A, 2B, 3A, 3B, 4A, 4B	1.2, 1.5, 1.8	-	All	-

⁽³⁾ The SPI flash memory's VCC is connected to VCCIO1A_4B. If you are using the SPI flash memory, drive the VCCIO1A_4B with a 1.8 V supply.

Table 6: Titanium I/O Banks by Package for Ti90, Ti120 and Ti180 FPGAs

Package	I/O Banks	Voltage (V)	Dynamic Voltage Support	DDIO Support	Merged Banks
J361, M361	2B, 2C, 3A, 3B, 4A, 4B, 4C	1.2, 1.5, 1.8	-	All	2A_2B, 3B_3C
	BL, TL, TR, BR	1.8, 2.5, 3.0, 3.3	✓	All	-
J484, M484	2B, 3A, 3B, 4A, 4B, 4C	1.2, 1.5, 1.8	-	All	2A_2B_2C, 3B_3C
	BL, TL, TR, BR	1.8, 2.5, 3.0, 3.3	✓	All	-
L484	2B, 3A, 3B, 4A, 4B, 4C	1.2, 1.5, 1.8	-	All	2A_2B_2C, 3B_3C
	BL, TL, TR, BR	1.8, 2.5, 3.0, 3.3	✓	All	-
F529, G529	2A, 2B, 2C, 3A, 3B, 3C, 4A, 4B, 4C	1.2, 1.5, 1.8	-	All	-
	BL, TL, TR, BR	1.8, 2.5, 3.0, 3.3	✓	All	-



Learn more: Refer to the FPGA pinout for information on the I/O bank assignments.

Dynamic Voltage Support

Titanium HVIO I/O banks support dynamic voltage shifting. This feature lets you change the voltage to the I/O bank during user mode. There are two methods for changing the voltage:

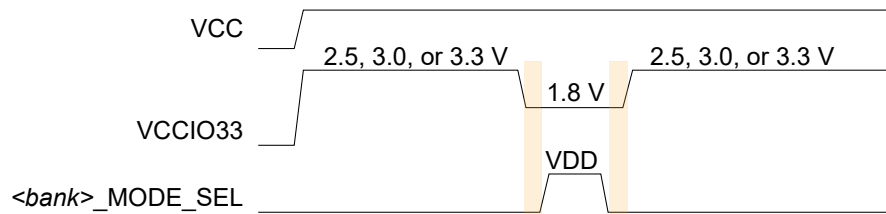
- If you are not using 1.8 V at all, you can simply change the voltage at any time. For example, you can change from 3.3 V to 2.5 V and then back to 3.3 V.
- If you are changing the voltage to 1.8 V, you must enable an option in the Interface Designer (which creates a mode select pin) and follow the timing requirements for the voltage change as described below.

To enable the option in the Interface Designer:

1. Choose **Device Setting > I/O Banks**.
2. Turn on **Enable Dynamic Voltage**.
3. Save.
4. Add the mode select pin (`<bank>_MODE_SEL`) to your RTL design.
5. Recompile.

When switching the voltage, pull down the voltage to 1.8 V for at least 1 ms before pulling the mode select signal high. Similarly, wait for at least 1 ms after releasing the mode select signal before changing the voltage back to the higher level.

Figure 12: VCCIO Switching Waveform (2.5, 3.0, or 3.3 V to 1.8 V)



Where <bank> is the bank name.
For example, for bank BR the pin
name is BR_MODE_SEL.

≥ 1 ms
The I/O performance cannot
be guaranteed during this period.

Design Check: I/O Bank Messages

When you check your design, the Interface Designer applies design rules to your configuration settings. The following tables show some of the error and warning messages you may encounter and explains how to fix them.

io_bank_rule_dyn_voltage (error)

Message	Bank <name> does not support dynamic voltage
To fix	Choose an HVIO I/O bank that supports dynamic voltage.

io_bank_rule_lvds (error)

Message	I/O Voltage has to be set to <#> when LVDS is used
To fix	Set the I/O bank voltage to 1.8 V if you are using pins in that bank for LVDS. Note: you cannot use MIPI lanes and LVDS pins in the same I/O bank because they require different voltages.

io_bank_rule_mipi_dphy (error)

Message	I/O Voltage has to be set to <#> when MIPI LANE is used
To fix	When the HSIO pins are used as MIPI lanes, the I/O bank voltage must be 1.2 V. Note: you cannot use MIPI lanes and LVDS pins in the same I/O bank because they require different voltages.

io_bank_rule_es_device (error)

Message	Unsupported 2.5 V in ES device
To fix	The Ti60ES FPGA does not support 2.5 V. Choose another voltage.

io_bank_rule_voltage_assignment (error)

Message	I/O Voltage <voltage> is not supported for the bank
To fix	Different I/O banks support different voltages. Choose a voltage that the I/O bank supports (refer to Titanium I/O Banks on page 24).

io_bank_rule_mode_sel (error)

Message	Empty Mode Select pin name found
To fix	Enter the name of the mode select pin.

io_bank_rule_vref (warning)

Message	It is not advisable to enable single-ended input SSTL/HSTL with LVDS Tx common mode in the following banks <banks>
To fix	An LVDS block with the Output Differential Type set to custom requires a VREF pin. A GPIO block using the single-ended SSTL or HSTL I/O standard also uses a VREF. The reference voltages for these standards is unlikely to be the same, so you should not use them in the same I/O bank. Instead, move one of the blocks to another bank.

Clock and Control Networks

Contents:

- **Clock Sources that Drive the Global and Regional Networks**
- **Configuring the Dynamic Clock Multiplexers**
- **Driving both the Global and Regional Networks**
- **Design Check: Clock Control Messages**

The clock and control network is distributed through the FPGA to provide clocking for the core's LEs, memory, DSP blocks, I/O blocks, and control signals. The FPGA has global signals that can be used as either clocks or control signals. The global signals are balanced trees that feed the whole FPGA.

The FPGA also has regional signals that can only reach certain FPGA regions, including the top or bottom edges. The FPGA has regional networks for the core, right interface, and left interface blocks. The top and bottom interface blocks have 1 regional clock network each. You can drive the right and left sides of each region independently. Each region also has a local network of clock signals that can only be used in that region.

The core's global buffer (GBUF) blocks drive the global and regional networks. Signals from the core and interface can drive the GBUF blocks.

Each network has dedicated enable logic to save power by disabling the clock tree. The logic dynamically enables/disables the network and guarantees no glitches at the output.

Figure 13: Global and Regional Clock Network Overview (Ti35, Ti60)

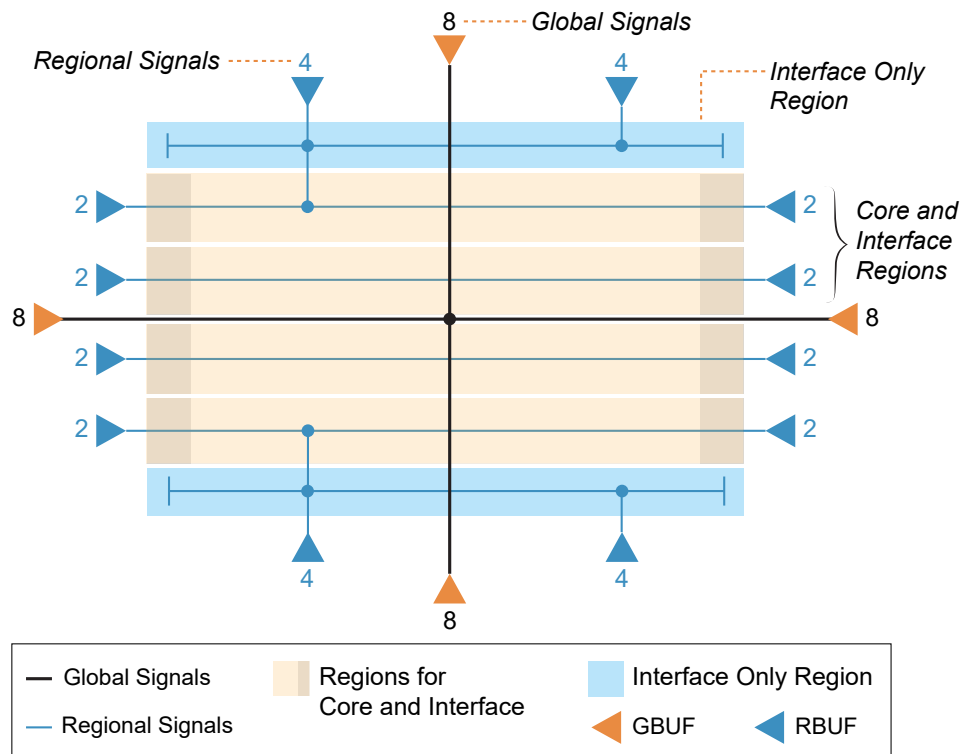
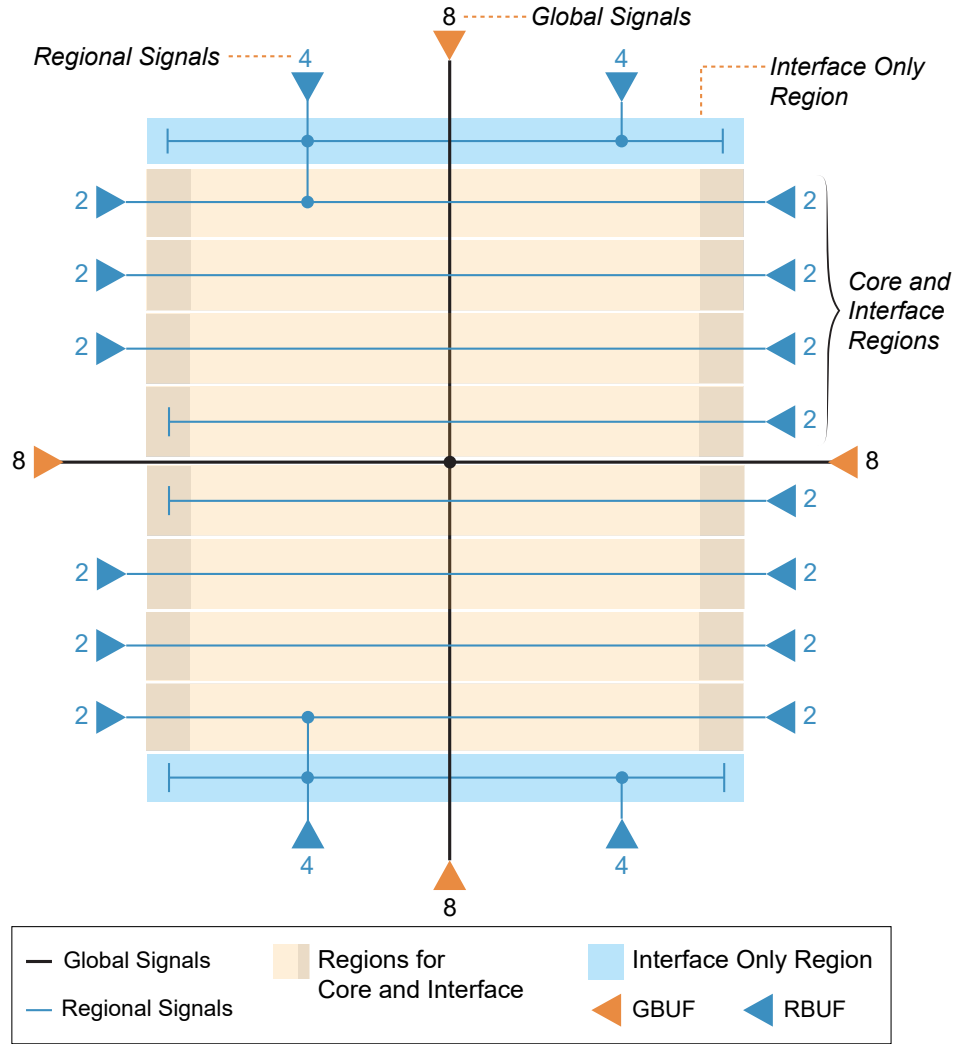


Figure 14: Global and Regional Clock Network Overview (Ti90, Ti120, Ti180)



Note: For detailed descriptions of the clock networks, refer to the data sheet.

Clock Sources that Drive the Global and Regional Networks

The Titanium global and regional networks are highly flexible and configurable. Clock sources can come from interface blocks, such as GPIO or PLLs, or from the core fabric.



Note: For more information on the clock sources that can drive the global and regional networks, refer to the data sheet.

Table 7: Clock Sources that Drive the Global and Regional Networks

Source	Description
GPIO	Supports GCLK and RCLK. (Only the P resources support this connection type).
LVDS RX	Supports GCLK and RCLK.
MIPI D-PHY RX and TX	Ti90, Ti120, Ti180: Can drive the word clock onto the global and regional clock networks.
MIPI RX Lane (configured as clock lane)	Supports GCLK (default) and RCLK. You can only use resources that are identified as clocks.
PLL	Output clocks 0 - 3 connect to the global network. Output clock 4 only connects to the regional network in the top or bottom interface regions (depending on the location of the PLL) and can only drive interface blocks on the top or bottom of the FPGA. Ti35, Ti60: Output clocks 0 - 3 connect to the global network. Output clock 4 only connects to the regional network in the top or bottom interface regions (depending on the location of the PLL) and can only drive interface blocks on the top or bottom of the FPGA. Ti90, Ti120, Ti180: All output clocks connect to the global network. Refer to Driving the Regional Network for the PLL clocks that drive the regional network.
Oscillator	Connects to global buffer. ⁽⁴⁾
Core	Signals from the core logic can drive the global or regional network.



Note: Some clock sources can drive both the global and regional networks. See [Driving both the Global and Regional Networks](#) on page 31 for instructions on using both.

⁽⁴⁾ The Ti60ES oscillator can only drive the core.

Configuring the Dynamic Clock Multiplexers

You configure the dynamic clock multiplexers in the Interface Designer. Expand **Device Setting** > **Clock/Control Configuration** and then click the multiplexer for the bottom, left, right, or top.

Table 8: Global Buffer Configuration

Option	Choices	Description
Core Clock <i>n</i> Pin Name	User defined	Specify a clock from the core that drives the global buffer. You can use up to four.
Enable Dynamic Mux 0 Enable Dynamic Mux 7	On or off	Turn on this option to enable the dynamic multiplexer. The clock multiplexers (static and dynamic) are numbered from 0 to 7. The dynamic ones are 0 and 7.
Dynamic Clock Mux Select [1:0] Bus Name	User defined	Specify the bus that controls the dynamic multiplexer.
Dynamic Clock Pin Name	User defined	Specify the clock name that drives the core RTL design.
Dynamic Clock Input <i>n</i>	List of sources ⁽⁵⁾ or None	Choose the clock source. Unassigned indicates that you have not yet used that resource. Choose None if you want to leave an input unassigned. A clock source must be assigned to the dynamic clock multiplexer input 0.
Independently Connect to Core	On or off	Enable this option if you want the clock source to also be available to the core.

Driving both the Global and Regional Networks

In some situations you may want a clock source to drive both the global and regional clock networks. In this case, you use the **rclk** connection type. To drive both a regional and a global, make these settings in the Interface Designer:

1. Create your clock source (e.g., a GPIO).
2. For the **Connection Type**, choose **rclk**.
3. In the Resource Assigner, assign a resource that has RCLK shown as an alternate connection.
4. Note the letter after GPIO in the resource name. This letter indicates the side of the FPGA. GPIOB = bottom, GPIOT = top, etc.
5. Click **Device Setting** > **Clock/Control Configuration**.
6. Select the bottom, left, right, or top as determined in step 4.
7. Click the **Regional Buffers** tab.
8. Choose the regional buffer according to the **Driving the Regional Network** section of the data sheet. When you select a buffer, the resource for that buffer displays under **Assignment**.
9. Specify a **Global Pin Name** to drive the global network.

⁽⁵⁾ Refer to the Driving the Global Network topic in the Titanium data sheet for the list of available clock sources.

Design Check: Clock Control Messages

When you check your design, the Interface Designer applies design rules to your clock and control settings. The following tables show some of the error and warning messages you may encounter and explains how to fix them.

clock_rule_capacity (error)

Message	Cannot connect to more than <int> different clocks per region (40 rows) on left and right and <int> clocks on the top or bottom
To fix	You cannot have more than 32 clocks (GPIO configured in clkout mode) coming from the core. You need to remove some clocks.
Message	Cannot connect to more than <int> different clocks per region (40 rows) on left and right
To fix	You are using more clocks that are available for the local region. Remove some clocks. See "Driving the Local Network" in the data sheet.
Message	Cannot connect to more than <int> different clocks on top and bottom
To fix	You are using more clocks than are available on the top and bottom local regions. Remove some clocks. See "Driving the Local Network" in the data sheet.

clock_rule_max_count (error)

Message	Number of core clock used exceeds max limit of <int>
To fix	You cannot have more than 32 clocks (GPIO configured in clkout mode) coming from the core. You need to remove some clocks.

clock_rule_lvds_rx_clock_source (error)

Message	The following PLL instance has output clocks driving LVDS Rx instance on different sides pair - [<PLL instance>: [<clock name>(<LVDS instance>)]]
To fix	The PLL output clocks go to multiple LVDS RX on different sides of the FPGA. The fast clock and slow clock can only drive I/O from the same left/right or top/bottom sides. For example, TR_PLL generates rx_fastclk and rx_slowclk for LVDS in the right-side bank. These clocks can also drive the LVDS channel on the left bank. However, they cannot drive LVDS in the top or bottom banks.

clock_rule_pll_ref_clock_lvds_rx (error)

Message	The following PLL instance has reference clock that does not match the side of the LVDS Rx instance driven by its output clocks - [<PLL instance>: [<clock name>(<LVDS instance>)]]
To fix	Choose the PLL reference clock that is on the same side as the LVDS that the output clock is driving. For example, if TR_PLL is driving LVDS on the right side, the PLL external source clock should also come from the I/O on the right side.

clock_rule_undefined_name (warning)

Message	No clock source defined
To fix	All clocks in the periphery must be defined in the Interface Designer (GPIO clock, oscillator, PLL, LVDS GCLK, MIPI D-PHY CLK). This warning indicates that you have not defined it as an interface block. If the clock is generated in the core you can ignore this warning.

clkmux_rule_clocks_routed (error)

Message	Unrouted pins driving inputs of clock mux <ins name>:<inputs not routeable>
To fix	The software tries to route all of the clocks according to the scheme shown in "Driving the Global Network" in the data sheet. If it cannot find a mapping, it issues this error. Reassign the instances to other resources or try using a different PLL output clock (if they are not all are assigned).
Message	Some inputs of clock mux <ins name> were not routed
To fix	The software tries to route all of the clocks according to the scheme shown in "Driving the Global Network" in the data sheet. If it cannot find a mapping for some of the pins, it issues this error. Reassign the instances to other resources or try using a different PLL output clock (if they are not all are assigned).

clkmux_rule_core_clock_pin (error)

Message	Core clock pin can only be used with dynamic mux enabled
To fix	Clocks from the core can only drive the dynamic clock multiplexers (see "Driving the Global Network" in the data sheet). Enable a dynamic mux (0 or 7) and assign the core clock to it (Configuring the Dynamic Clock Multiplexers on page 31).

clkmux_rule_core_clock_static_mux (error)

Message	Core clock pin <name> not allowed to route through static mux output
To fix	Clocks that come from the core can only connect to dynamic multiplexer input (see "Driving the Global Network" in the data sheet). You need to add the clock to a dynamic mux.

clkmux_rule_dynamic_clock_pin (error)

Message	Dynamic clock pin names for <both dynamic muxes/dynamic mux 0/7> <are/is> empty
To fix	You need to specify the pin name. Go to Device Setting > Clock/Control Configuration > <region> > Global Buffers tab .

clkmux_rule_dynamic_clock_select_pin (error)

Message	Dynamic clock select pin names for <both dynamic muxes/dynamic mux 0/7> <are/is> empty
To fix	You need to specify the pin name. Go to Device Setting > Clock/Control Configuration > <region> > Global Buffers tab .

clkmux_rule_global_regional_pin (error)

Message	Missing global pin name for a regional connection that also connects to global buffer
To fix	If you want to use a clock source to drive <i>both</i> the global and regional networks, you need to specify the name of the clock that drives the global network. See Driving both the Global and Regional Networks on page 31.
Message	Global and regional buffer connections require unique pin name
To fix	If you are using both the global and regional buffers, you need to specify unique pin names for each one.

clkmux_rule_global_regional_resource (error)

Message	Regional buffer resource <name> does not support global connection
To fix	Some clock sources cannot connect to the global network, e.g., PLL CLKOUT4. Look in the Resource Assigner Alt Conn column to find a different resource that can connect.

clkmux_rule_pll_output_clock (error)

Message	Found the following PLL output clock routed multiple times: <PLL output>
To fix	PLL clocks connect to clock muxes on two sides of the device. Only one of these connections may be used at a time. Make sure you have not chosen a PLL clock as a dynamic clock source and enabled it to independently drive the core on two different clock muxes.

clkmux_rule_regional_conn_type (error)

Message	Regional buffer instance <name> requires connection type to be set to rclk
To fix	If you are a clock source to drive the regional network, you need to choose rclk as the Connection Type in the Input tab. (see Input Mode on page 55)

clkmux_rule_type_config (error)

Message	Resource <name> configured as MIPI LANE Rx not allowed to connect to dynamic mux <0/7> Resource <name> configured as GPIO/LVDS not allowed to connect to dynamic mux <0/7> Invalid output clock configuration of <name> connected to index <int> of dynamic mux <0/7> Assigned core clock pin name at index <int> of dynamic mux <0/7> is empty Resource <name> assigned to input <int> of dynamic mux <0/7> but no valid configured instance found
To fix	You get this error if you have not configured the clock source correctly to connect to the dynamic mux. For example, you can configure the same resource as GPIO, LVDS, or a MIPI lane. If you configure it as a MIPI lane, then you cannot connect it to the dynamic mux. Only GPIO or LVDS can connect. See "Driving the Global Network" in the data sheet for the sources that can drive the dynamic multiplexer.
Message	Resource <name> with instance <name> assigned to RBUF# has invalid configuration Resource <name> with instance <name> can only connect to regional buffer input # through output clock 4
To fix	You can only connect the PLL output to regional buffers on the top and bottom. Refer to "Driving the Regional Network" in the data sheet.

DDR Interface

Contents:

- [About the DDR DRAM Interface](#)
- [DDR Interface Designer Settings](#)

Some Titanium FPGAs have a hardened IP interface block to communicate with off-the-shelf memories. Refer to the [Package/Interface Support Matrix](#) on page 9 to find out if your FPGA supports DDR.

About the DDR DRAM Interface



Important: All information is preliminary and pending definition.

The DDR PHY interface supports LPDDR4 and LPDDR4x memories with x16 or x32 DQ widths and a memory controller hard IP block. The memory controller provides two full-duplex AXI4 buses to communicate with the FPGA core.



Note: The DDR PHY and controller are hard blocks; you cannot bypass the DDR DRAM memory controller to access the PHY directly for non-DDR memory controller applications.

Figure 15: DDR DRAM Block Diagram

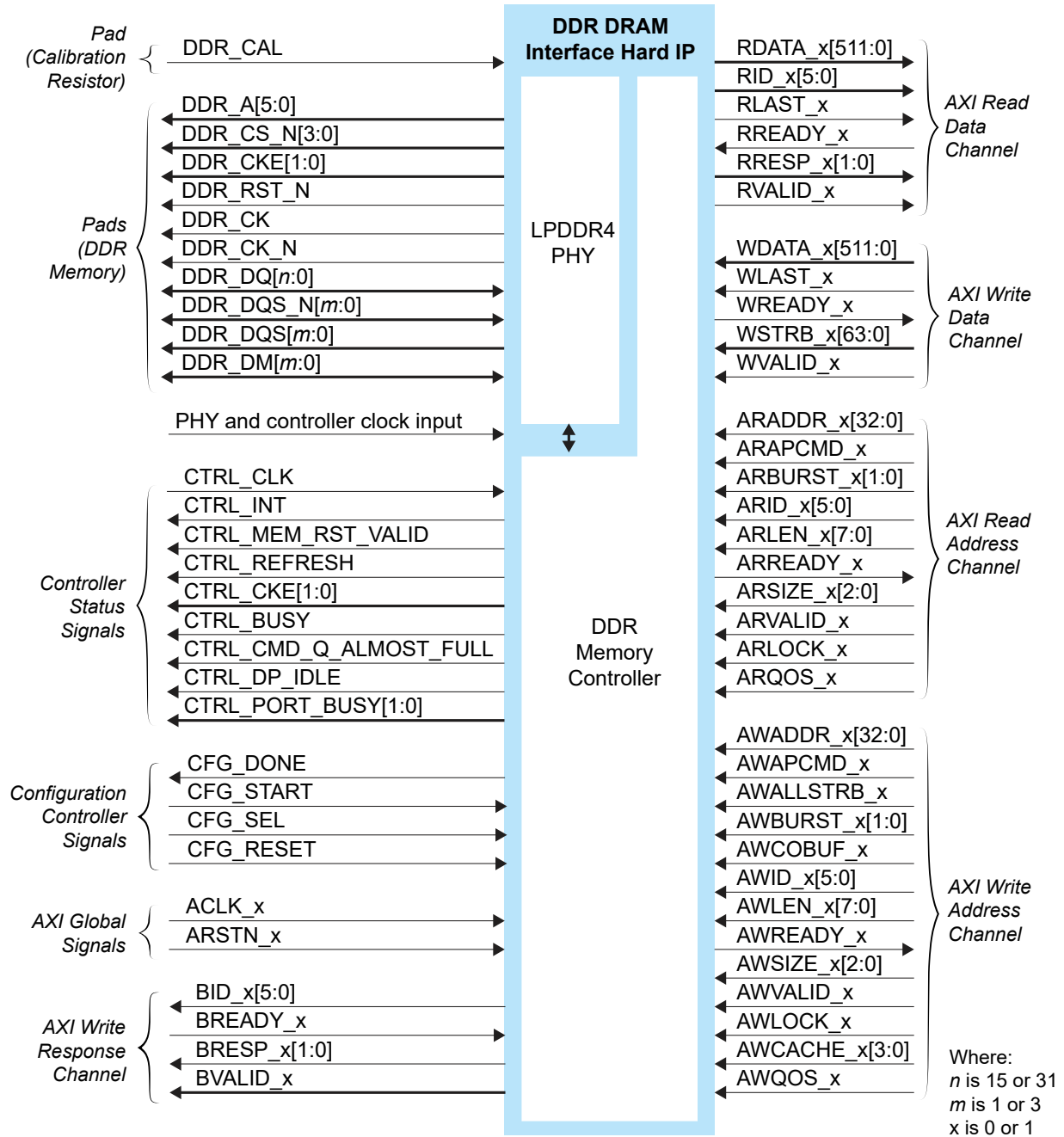
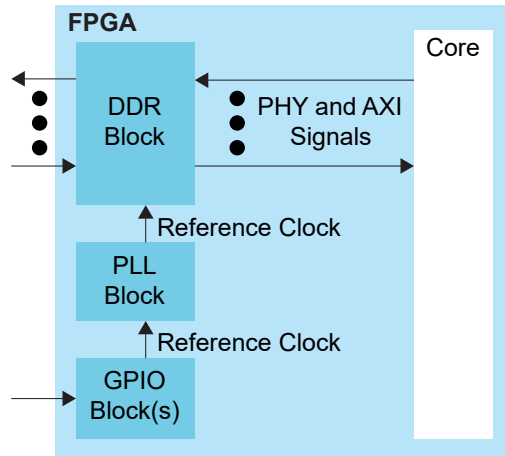


Figure 16: DDR DRAM Interface Block Diagram



Note: The PLL reference clock must be driven by I/O pads. The Efinity® software issues a warning if you do not connect the reference clock to an I/O pad. (Using the clock tree may induce additional jitter and degrade the DDR performance.) Refer to [About the PLL Interface](#) on page 127 for more information about the PLL block.

Table 9: DDR DRAM Pads

Signal	Direction	Description
DDR_A[5:0]	Output	Address signals to the DRAM.
DDR_CS_N[3:0]	Output	Chip select to the DRAM.
DDR_CKE[1:0]	Output	Active-high clock enable signals to the DRAM.
DDR_RST_N	Output	Active-low reset signal to the DRAM.
DDR_CK	Output	Differential clock signals to the DRAM.
DDR_CK_N	Output	
DDR_DQ[n:0]	Bidirectional	Data bus to/from the memories. For writes, the FPGA drives these signals. For reads, the memory drives these signals. These signals are connected to the DQ pins on the memories. n is 15 or 31 depending on the Data Width setting.
DDR_DQS_N[m:0]	Bidirectional	Differential data strobes to/from the memories. For writes, the FPGA drives these signals. For reads, the memory drives these signals. These signals are connected to the DQS pins on the memories. m is 1 or 3 depending on the DQ width.
DDR_DQS[m:0]	Bidirectional	
DDR_DM[m:0]	Bidirectional	Active-high data-mask signals to the memories. m is 1 or 3 depending on the DQ width.

Table 10: Calibration Resistor Pad

Signal	Direction	Description
DDR_CAL	Input	Calibration resistor connection. Connect to the ground through a 240 Ω resistor on your board.

Table 11: Controller Status Signals

Signal	Direction	Clock Domain	Description
CTRL_CLK	Input	N/A	Clock for controller status signals.
CTRL_INT	Output	N/A	Controller detects Interrupt.
CTRL_MEM_RST_VALID	Output	N/A	Controller has been reset.
CTRL_REFRESH	Output	CTRL_CLK	Indicate controller is executing refresh command.
CTRL_CKE[1:0]	Output	CTRL_CLK	Delayed 'control_cke' from the controller, indicating that the memory is in self-refresh or power down mode.
CTRL_BUSY	Output	CTRL_CLK	Controller is busy reading data.
CTRL_CMD_Q_ALMOST_FULL	Output	CTRL_CLK	Command queue reached 'q_fullness' parameter.
CTRL_DP_IDLE	Output	CTRL_CLK	Datapath is idle.
CTRL_PORT_BUSY[1:0]	Output	CTRL_CLK	Indicate if port is reading data.

Table 12: Configuration Controller Signals

Signal	Direction	Description
CFG_RESET	Input	Active-high configuration controller reset. Asserting this signal also resets the DDR controller, PHY and the DRAM device.
CFG_START	Input	Start the configuration controller.
CFG_DONE	Output	Indicates the configuration controller is done
CFG_SEL	Input	Tie this input to low to enable the configuration controller.

Table 13: AXI4 Global Signals (Interface to FPGA Core Logic)

Signal	Direction	Clock Domain	Description
ACLK_x	Input	N/A	AXI4 clock inputs.
ARSTN_x	Input	ACLK_x	Active-low reset signal to the AXI interface.

Table 14: AXI4 Write Response Channel Signals (Interface to FPGA Core Logic)

Signal x is 0 or 1	Direction	Clock Domain	Description
BID_x[5:0]	Output	ACLK_x	Response ID tag. This signal is the ID tag of the write response.
BREADY_x	Input	ACLK_x	Response ready. This signal indicates that the master can accept a write response.
BRESP_x[1:0]	Output	ACLK_x	Read response. This signal indicates the status of the read transfer.
BVALID_x	Output	ACLK_x	Write response valid. This signal indicates that the channel is signaling a valid write response.

Table 15: AXI4 Read Data Channel Signals (Interface to FPGA Core Logic)

Signal x is 0 or 1	Direction	Clock Domain	Description
RDATA_x[511:0]	Output	ACLK_x	Read data.
RID_x[5:0]	Output	ACLK_x	Read ID tag. This signal is the identification tag for the read data group of signals generated by the slave.
RLAST_x	Output	ACLK_x	Read last. This signal indicates the last transfer in a read burst.
RREADY_x	Input	ACLK_x	Read ready. This signal indicates that the master can accept the read data and response information.
RRESP_x[1:0]	Output	ACLK_x	Read response. This signal indicates the status of the read transfer.
RVALID_x	Output	ACLK_x	Read valid. This signal indicates that the channel is signaling the required read data.

Table 16: AXI4 Write Data Channel Signals (Interface to FPGA Core Logic)

Signal x is 0 or 1	Direction	Clock Domain	Description
WDATA_x[511:0]	Input	ACLK_x	Write data.
WLAST_x	Input	ACLK_x	Write last. This signal indicates the last transfer in a write burst.
WREADY_x	Output	ACLK_x	Write ready. This signal indicates that the slave can accept the write data.
WSTRB_x[63:0]	Input	ACLK_x	Write strobes. This signal indicates which byte lanes hold valid data. There is one write strobe bit for each eight bits of the write data bus.
WVALID_x	Input	ACLK_x	Write valid. This signal indicates that valid write data and strobes are available.

Table 17: AXI4 Read Address Signals (Interface to FPGA Core Logic)

Signal x is 0 or 1	Direction	Clock Domain	Description
ARADDR_x[32:0]	Input	ACLK_x	Read address. It gives the address of the first transfer in a burst transaction.
ARBURST_x[1:0]	Input	ACLK_x	Burst type. The burst type and the size determine how the address for each transfer within the burst is calculated. 'b01 = INCR 'b10 = WRAP
ARID_x[5:0]	Input	ACLK_x	Address ID. This signal identifies the group of address signals.
ARLEN_x[7:0]	Input	ACLK_x	Burst length. This signal indicates the number of transfers in a burst.
ARREADY_x	Output	ACLK_x	Address ready. This signal indicates that the slave is ready to accept an address and associated control signals.
ARSIZE_x[2:0]	Input	ACLK_x	Burst size. This signal indicates the size of each transfer in the burst.
ARVALID_x	Input	ACLK_x	Address valid. This signal indicates that the channel is signaling valid address and control information.
ARLOCK_x	Input	ACLK_x	Lock type. This signal provides additional information about the atomic characteristics of the transfer.
ARAPCMD_x	Input	ACLK_x	Read auto-precharge.
ARQOS_x	Input	ACLK_x	QoS identifier for read transaction.

Table 18: AXI4 Write Address Signals (Interface to FPGA Core Logic)

Signal x is 0 or 1	Direction	Clock Domain	Description
AWADDR_x[32:0]	Input	ACLK_x	Write address. It gives the address of the first transfer in a burst transaction.
AWBURST_x[1:0]	Input	ACLK_x	Burst type. The burst type and the size determine how the address for each transfer within the burst is calculated.
AWID_x[5:0]	Input	ACLK_x	Address ID. This signal identifies the group of address signals.
AWLEN_x[7:0]	Input	ACLK_x	Burst length. This signal indicates the number of transfers in a burst.
AWREADY_x	Output	ACLK_x	Address ready. This signal indicates that the slave is ready to accept an address and associated control signals.
AWSIZE_x[2:0]	Input	ACLK_x	Burst size. This signal indicates the size of each transfer in the burst.
AWVALID_x	Input	ACLK_x	Address valid. This signal indicates that the channel is signaling valid address and control information.
AWLOCK_x	Input	ACLK_x	Lock type. This signal provides additional information about the atomic characteristics of the transfer.
AWAPCMD_x	Input	ACLK_x	Write auto-precharge.
AWQOS_x	Input	ACLK_x	QoS identifier for write transaction.
AWCACHE_x[3:0]	Input	ACLK_x	Memory type. This signal indicates how transactions are required to progress through a system.
AWALLSTRB_x	Input	ACLK_x	Write all strobes asserted. The DDR controller only supports a maximum of 16 AXI beats for write commands using this signal.
AWCOBUF_x	Input	ACLK_x	Write coherent bufferable selection.

Clock Input to PHY and Controller

PLL_TL2 CLKOUT3 and CLKOUT4 are clocks to drive the DDR PHY and controller. The CLKOUT3 drives the DDR PHY and must run at half of the PHY data rate (for example, 2000 Mbps requires a 1000 MHz clock). CLKOUT4 drives the DDR controller and must run at a quarter of the PHY data rate (for example, 2000 Mbps requires a 500 MHz clock).

You only need to instantiate PLL_TL2 with CLKOUT3 and CLKOUT4, enabled with the required frequencies. The Efinity software connects the clocks to the DDR PHY and controller automatically.

DDR Interface Designer Settings

The following tables describe the settings for the Titanium DDR block in the Interface Designer.

Table 19: Base Tab

Parameter	Choices	Notes
Instance Name	User defined	Indicate the DDR instance name. This name is the prefix for all DDR signals.
DDR Resource	None, DDR_0	Only one resource available.
Data Width	16, 32	Choose the DQ width. Default: 32 (16 for M361, M484, F529 packages)
Memory Density	2G, 3G, 4G, 6G, 8G, 12G, 16G	Choose the memory density per channel. Default: 4G
Physical Rank	1, 2	Default: 1
Memory Type	LPDDR4, LPDDR4x	Default: LPDDR4

Table 20: Advanced Options Tab (FPGA Settings)

Option	Choices	Notes
DQ Pull-Down Drive Strength (Ohm)	34.3, 40, 48, 60, 80, 120, 240	Default: 48
DQ Pull-Down ODT (Ohm)	34.3, 40, 48, 60, 80, 120, 240, High-Z	Default: 60
DQ Pull-Up Drive Strength (Ohm)	34.3, 40, 48, 60, 80, 120, 240	Default: 48
DQ Pull-Up ODT (Ohm)	34.3, 40, 48, 60, 80, 120, 240, High-Z	Default: High-Z
VREF Range Selection	Range 0, Range 1	Default: Range 0
VREF Setting (% of VDDQ)	<p>LPDDR4, Range 0: 5.40 - 38.42 (step: 0.26)</p> <p>LPDDR4, Range 1: 11.90 - 48.222 (step: 0.286)</p> <p>LPDDR4x, Range 0: 11.60 - 49.70 (step: 0.3)</p> <p>LPDDR4x, Range 1: 21.20 - 59.30 (step: 0.3)</p>	<p>Default:</p> <p>LPDDR4, Range 0: 21.78</p> <p>LPDDR4, Range 1: 29.918</p> <p>LPDDR4x, Range 0: 30.5</p> <p>LPDDR4x, Range 1: 40.1</p>

Table 21: Advanced Options Tab (Memory Mode Register Settings)

Option	Choices	Notes
Burst Length	BL = 16 Sequential, BL = 16 or 32 Sequential, BL = 32 Sequential	Default: 16 Sequential
CA Bus Receiver On-Die-Termination for CS0/CS1	Disable, RZQ/1, RZQ/2, RZQ/3, RZQ/4, RZQ/5, RZQ/6	Default: Disable
DQ Bus Receiver On-Die-Termination for CS0/CS1	Disable, RZQ/1, RZQ/2, RZQ/3, RZQ/4, RZQ/5, RZQ/6	Default: Disable
Pull-Down Drive Strength (PDDS) for CS0/CS1	RFU, RZQ/1, RZQ/2, RZQ/3, RZQ/4, RZQ/5, RZQ/6	Default: RZQ/6
CA VREF Setting Range Selection	RANGE [0], RANGE [1]	Default: RANGE [1]
CA VREF Settings (% of VDD2)	RANGE [0]: 10 - 30 (step: 0.4) RANGE [1]: 22 - 42 (step: 0.4)	Default: RANGE [0]: 27.2 RANGE [1]: 27.2
DQ VREF Setting Range Selection	RANGE [0], RANGE [1]	Default: RANGE [1]
DQ VREF Settings (% of VDDQ)	RANGE [0]: 10 - 30 (step: 0.4) RANGE [1]: 22 - 42 (step: 0.4)	Default: RANGE [0]: 27.2 RANGE [1]: 27.2
CK ODT CS0/CS1 Enabled for Non-terminating Rank	Override Disabled, Override Enabled	Default: Override Disabled
CS ODT CS/CS1 Enabled for Non-terminating Rank	Override Disabled, Override Enabled	Default: Override Disabled
CA ODT CS/CS1 Termination Disable	Obeys ODT_CA Bond Pad, Disabled	Default: Override Obeys ODT_CA Bond Pad

Table 22: Config Controller Tab

Option	Choices	Notes
<description> Pin Name	User defined	Configuration and control pins. You can use the default names or specify your own.

Table 23: Advanced Options Tab (Memory Timing Settings)

Option	Choices	Notes
tCCD, CAS-to-CAS Delay (cycles)	8 - 31	Default: 8
tCCDMW, CAS-to-CAS Delay Masked Write (cycles)	32 - 63	Default: 32
tFAW, Four-Bank Activate Window (ns)	40 - 100	Default: 40
tPPD, Precharge to Precharge Delay (cycles)	4 - 7	Default: 4
tRAS, Row Active Time (ns)	42 - 100	Default: 42
tRCD, RAS-to-CAS Delay (ns)	18 - 100	Default: 18
tRPab, Row Precharge Time (All Banks) (ns)	21 - 100	Default: 21
tRPpb, Row Precharge Time (Single Bank) (ns)	18 - 100	Default: 18
tRRD, Active Bank-A to Active Bank-B (ns)	10 - 100	Default: 10
tRTP, Internal Read To Precharge Delay (ns)	7.5 - 100	Default: 7.5
tSR, Minimum Self Refresh Time (ns)	15 - 100	Default: 15
tWR, Write Recovery Time (ns)	18 - 60	Default: 18
tWTR, Write-To-Read Delay (ns)	10 - 60	Default: 10

Table 24: AXI 0 and AXI 1 Tabs

Parameter	Choices	Notes
Enable Target 0 Enable Target 1	On or off	Turn on to enable the AXI 0 interface. Turn on to enable the AXI 1 interface.
AXI Width	256, 512	Choose the AXI bit width. Default: 512
AXI Clock Input Pin Name	User defined	Specify the name of the AXI input clock pin.
Invert AXI Clock Input	On or off	Turn on to invert the AXI clock.
AXI Reset Pin Name	User defined	Specify the name of the AXI reset clock pin or use the default.
Read Address Channel tab Write Address Channel tab Write Response Channel tab Read Data Channel tab Write Data Channel tab	User defined	These tabs defines the AXI signal names for the channels. Efinix recommends that you use the default names.

Table 25: Controller Status Tab

Option	Choices	Notes
Invert Controller Status Clock Pin	On or off	Turn on if you want to invert the clock. Default: off
<description> Pin Name	User defined	Controller status pins. You can use the default names or specify your own.

GPIO Interface

Contents:

- [Types of GPIO](#)
- [Features for HVIO and HSIO Configured as GPIO](#)
- [About the HVIO Interface](#)
- [About the HSIO Interface](#)
- [HSIO Configured as GPIO](#)
- [Using the GPIO Block](#)
- [Using the GPIO Bus Block](#)
- [Create a TX Serializer Interface](#)
- [Create a RX Deserializer Interface](#)
- [Design Check: GPIO Messages](#)

Titanium FPGAs have general-purpose I/O (GPIO) pins that allow the FPGA to communicate with other components on your circuit board. When you create your RTL design in the Efinity® software, you use the Interface Designer to add GPIO blocks for each input, output, or bi-directional pin in your design.

Titanium GPIO pins have various features, depending on the position of the pin and which package you are using. Refer to the Resource Assigner in the Interface Designer for the features of the GPIO pin you want to use.

- GPIO that provide normal functionality
- GPIO with the double-data I/O (DDIO) feature that can capture twice the data
- HSIO as GPIO where the HSIO pin acts as a GPIO

The following sections describe the GPIO interface and how to use it in your design.

Types of GPIO

The Titanium FPGA supports two types of GPIO:

- *High-voltage I/O (HVIO)*—Simple I/O blocks that can support single-ended I/O standards.
- *High-speed I/O (HSIO)*—Complex I/O blocks that can support single-ended and differential I/O functionality.

The I/O logic comprises three register types:

- *Input*—Capture interface signals from the I/O before being transferred to the core logic
- *Output*—Register signals from the core logic before being transferred to the I/O buffers
- *Output enable*—Enable and disable the I/O buffers when I/O used as output

The HVIO supports the following I/O standards.

Table 26: HVIO Supported Standards

Standard	VCCIO33 (V)	When Configured As
LVTTTL 3.3 V	3.3	GPIO
LVTTTL 3.0 V	3.0	GPIO
LVC MOS 3.3 V	3.3	GPIO
LVC MOS 3.0 V	3.0	GPIO
LVC MOS 2.5 V	2.5	GPIO
LVC MOS 1.8 V	1.8	GPIO



Important: Efinix recommends that you limit the number of 3.0/3.3 V HVIO as I/O or output to 6 per bank to avoid switching noise. The Efinity[®] software issues a warning if you exceed the recommended limit.

The HSIO supports the following I/O standards.

Table 27: HSIO Supported I/O Standards

Standard	VCCIO (V)		VCCAUX (V)	VREF (V)	When Configured As
	TX	RX			
LVC MOS 1.8 V	1.8	1.8	1.8	-	GPIO
LVC MOS 1.5 V	1.5	1.5	1.8	-	GPIO
LVC MOS 1.2 V	1.2	1.2	1.8	-	GPIO
HSTL/Differential HSTL 1.8 V SSTL/Differential SSTL 1.8 V	1.8	1.8	1.8	0.9	GPIO
HSTL/Differential HSTL 1.5 V SSTL/Differential SSTL 1.5 V	1.5	1.5, 1.8 ⁽⁶⁾	1.8	0.75	GPIO
HSTL/Differential HSTL 1.2 V SSTL/Differential SSTL 1.2 V	1.2	1.2, 1.5, 1.8 ⁽⁶⁾	1.8	0.6	GPIO
LVDS/RSDS/mini-LVDS	1.8	1.5, 1.8 ⁽⁶⁾	1.8	-	LVDS
Sub-LVDS	1.8	1.5, 1.8 ⁽⁶⁾	1.8	-	LVDS
MIPI/SLVS	1.2	1.2	1.8	-	MIPI Lane

The differential receivers are powered by VCCAUX, which gives you the flexibility to choose the VCCIO you want to use. However, you must comply to the requirements stated in the previous table.

⁽⁶⁾ To prevent pin leakage, you must ensure that the voltage at the pin does not exceed VCCIO.

Features for HVIO and HSIO Configured as GPIO

The following table describes the features supported by HVIO and HSIO configured as GPIO.

Table 28: Features for HVIO and HSIO Configured as GPIO

Feature	HVIO	HSIO Configured as GPIO
Double-data I/O (DDIO)	✓	✓
Dynamic pull-up	-	✓
Pull-up/Pull-down	✓	✓
Slew-Rate Control	-	✓
Variable Drive Strength	✓	✓
Schmitt Trigger	✓	✓
1:4 Serializer/Deserializer (Full rate mode only)	-	✓
Programmable Bus Hold	-	✓
Static Programmable Delay Chains	✓	✓
Dynamic Programmable Delay Chains	-	✓

Table 29: GPIO Modes

GPIO Mode	Description
Input	<p>Only the input path is enabled; optionally registered. If registered, the input path uses the input clock to control the registers (positively or negatively triggered).</p> <p>Select the alternate input path to drive the alternate function of the GPIO. The alternate path cannot be registered.</p> <p>In DDIO mode, two registers sample the data on the positive and negative edges of the input clock, creating two data streams.</p>
Output	<p>Only the output path is enabled; optionally registered. If registered, the output path uses the output clock to control the registers (positively or negatively triggered).</p> <p>The output register can be inverted.</p> <p>In DDIO mode, two registers capture the data on the positive and negative edges of the output clock, multiplexing them into one data stream.</p>
Bidirectional	<p>The input, output, and OE paths are enabled; optionally registered. If registered, the input clock controls the input register, the output clock controls the output and OE registers. All registers can be positively or negatively triggered. Additionally, the input and output paths can be registered independently.</p> <p>The output register can be inverted.</p>
Clock output	Clock output path is enabled.

During configuration, all GPIO pins are configured in weak pull-up mode.

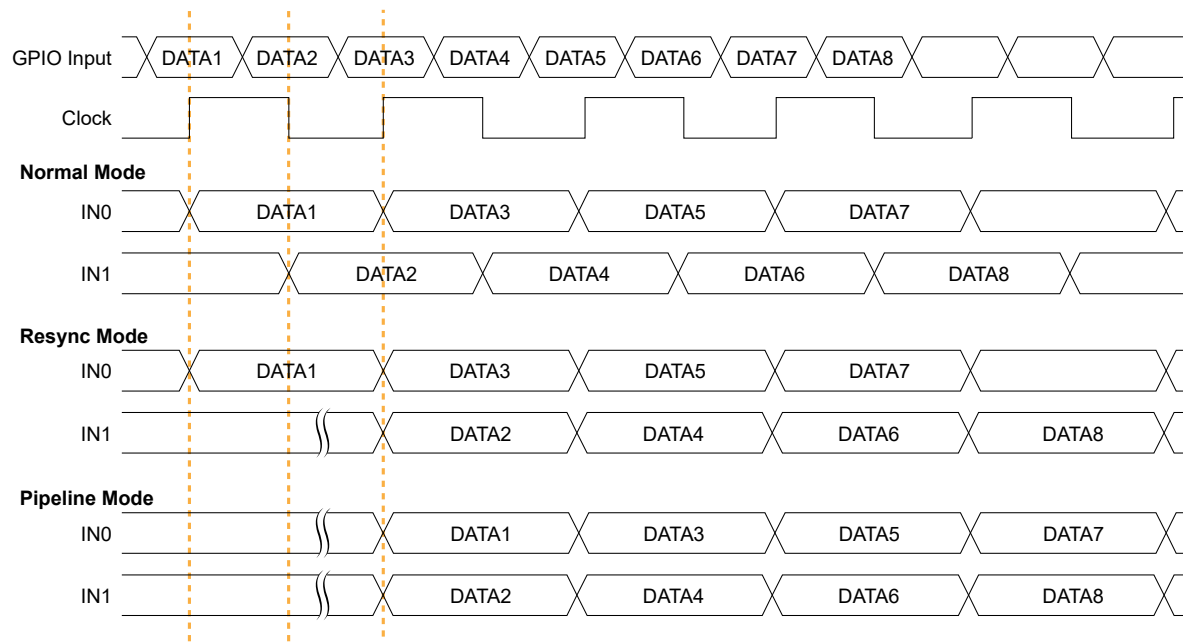
During user mode, unused GPIO pins are tristated and configured in weak pull-up mode. You can change the default mode to weak pull-down in the Interface Designer.

Double-Data I/O

Titanium FPGAs support double data I/O (DDIO) on input and output registers. In this mode, the DDIO register captures data on both positive and negative clock edges. The core receives 2 bit wide data from the interface.

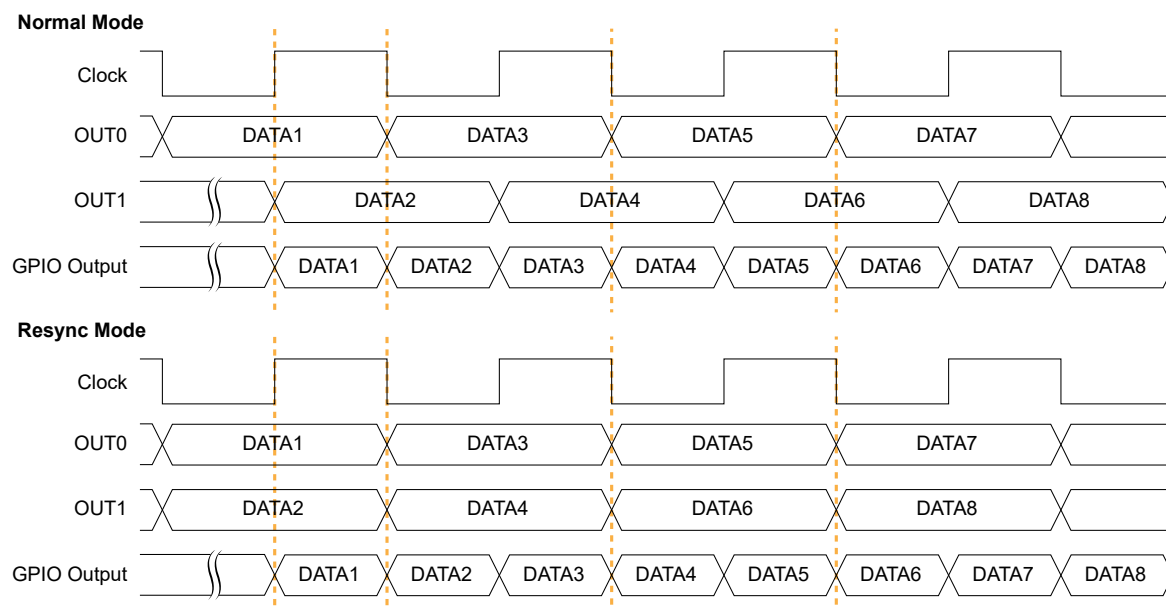
In normal mode, the interface receives or sends data directly to or from the core on the positive and negative clock edges. In resync and pipeline mode, the interface resynchronizes the data to pass both signals on the positive clock edge only.

Figure 17: DDIO Input Timing Waveform



In resync mode, the IN1 data captured on the falling clock edge is delayed one half clock cycle.
In the Interface Designer, IN0 is the HI pin name and IN1 is the LO pin name.

Figure 18: DDIO Output Timing Waveform



In the Interface Designer, OUT0 is the HI pin name and OUT1 is the LO pin name.

Programmable Delay Chains

The HSIO configured as GPIO supports programmable delay chain. In some cases you can use static and dynamic delays at the same time.

Table 30: Programmable Delay Support

Delay Type	GPIO Type	Description
Static	Single-ended	16 steps with approximately 60 ps of delay per step, both input and output paths.
	Differential RX	64 steps of approximately 25 ps delay per step. You cannot use the static and dynamic delay at the same time. Available only on P input of the HSIO pin pair.
	Differential TX	64 steps with approximately 25 ps of delay per step.
Dynamic	Single ended and differential	64 steps with approximately 25 ps of delay per step, input only. Only available on input (RX) delay; available only on P input of the HSIO pin pair.

About the HVIO Interface

The HVIOs are grouped into banks. Each bank has its own VCCIO33 that sets the bank voltage for the I/O standard. Each HVIO consists of I/O logic and an I/O buffer. I/O logic connects the core logic to the I/O buffers. I/O buffers are located at the periphery of the device.

Figure 19: HVIO Interface Block

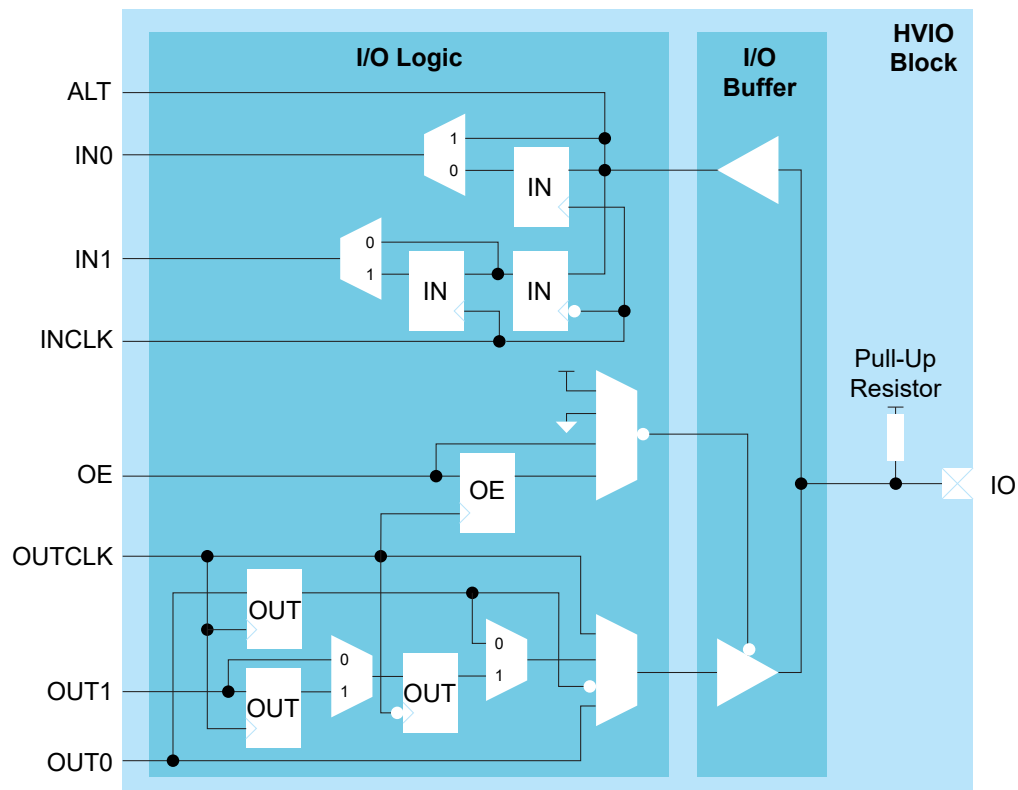


Table 31: HVIO Signals (Interface to FPGA Fabric)

Signal	Direction	Description
IN[1:0]	Output	Input data from the HVIO pad to the core fabric. IN0 is the normal input to the core. In DDIO mode, IN0 is the data captured on the positive clock edge (HI pin name in the Interface Designer) and IN1 is the data captured on the negative clock edge (LO pin name in the Interface Designer).
ALT	Output	Alternative input connection (in the Interface Designer, Register Option is none). HVIO only support pll_clkln as the alternative connection.
OUT[1:0]	Input	Output data to HVIO pad from the core fabric. OUT0 is the normal output from the core. In DDIO mode, OUT0 is the data captured on the positive clock edge (HI pin name in the Interface Designer) and OUT1 is the data captured on the negative clock edge (LO pin name in the Interface Designer).
OE	Input	Output enable from core fabric to the I/O block. Can be registered.
OUTCLK	Input	Core clock that controls the output and OE registers. This clock is not visible in the user netlist.
INCLK	Input	Core clock that controls the input registers. This clock is not visible in the user netlist.

Table 32: HVIO Pads

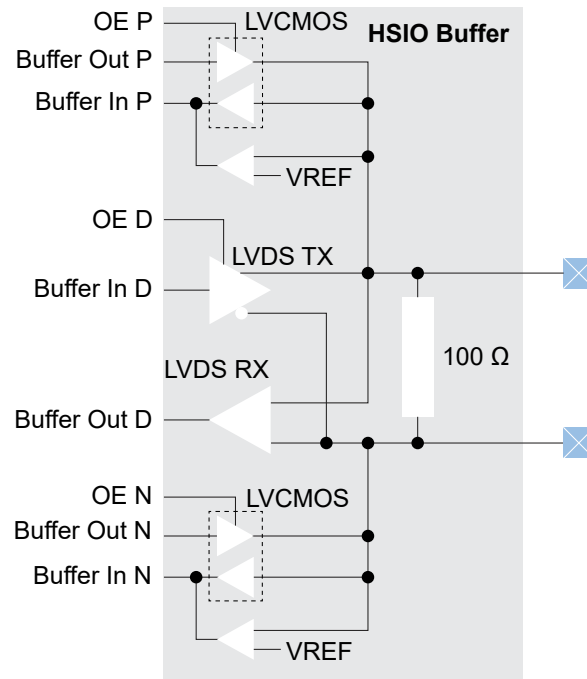
Signal	Direction	Description
IO	Bidirectional	HVIO pad.

About the HSIO Interface

Each HSIO block uses a pair of I/O pins as one of the following:

- *Single-ended HSIO*—Two single-ended I/O pins (LVCMOS, SSTL, HSTL)
- *Differential HSIO*—One differential I/O pins:
 - Differential SSTL and HSTL
 - LVDS—Receiver (RX), transmitter (TX), or bidirectional (RX/TX)
 - MIPI lane I/O—Receiver (RX) or transmitter (TX)

Figure 20: HSIO Buffer Block Diagram



Important: When you are using an HSIO pin as a GPIO, make sure to leave at least 1 pair of unassigned HSIO pins between any GPIO and LVDS or MIPI lane pins. This rule applies for pins on each side of the device (top, bottom, left, right). This separation reduces noise. The Efinity software issues an error if you do not leave this separation.

HSIO Configured as GPIO

You can configure each HSIO block as two GPIO (single-ended) or one GPIO (differential).

Figure 21: I/O Interface Block

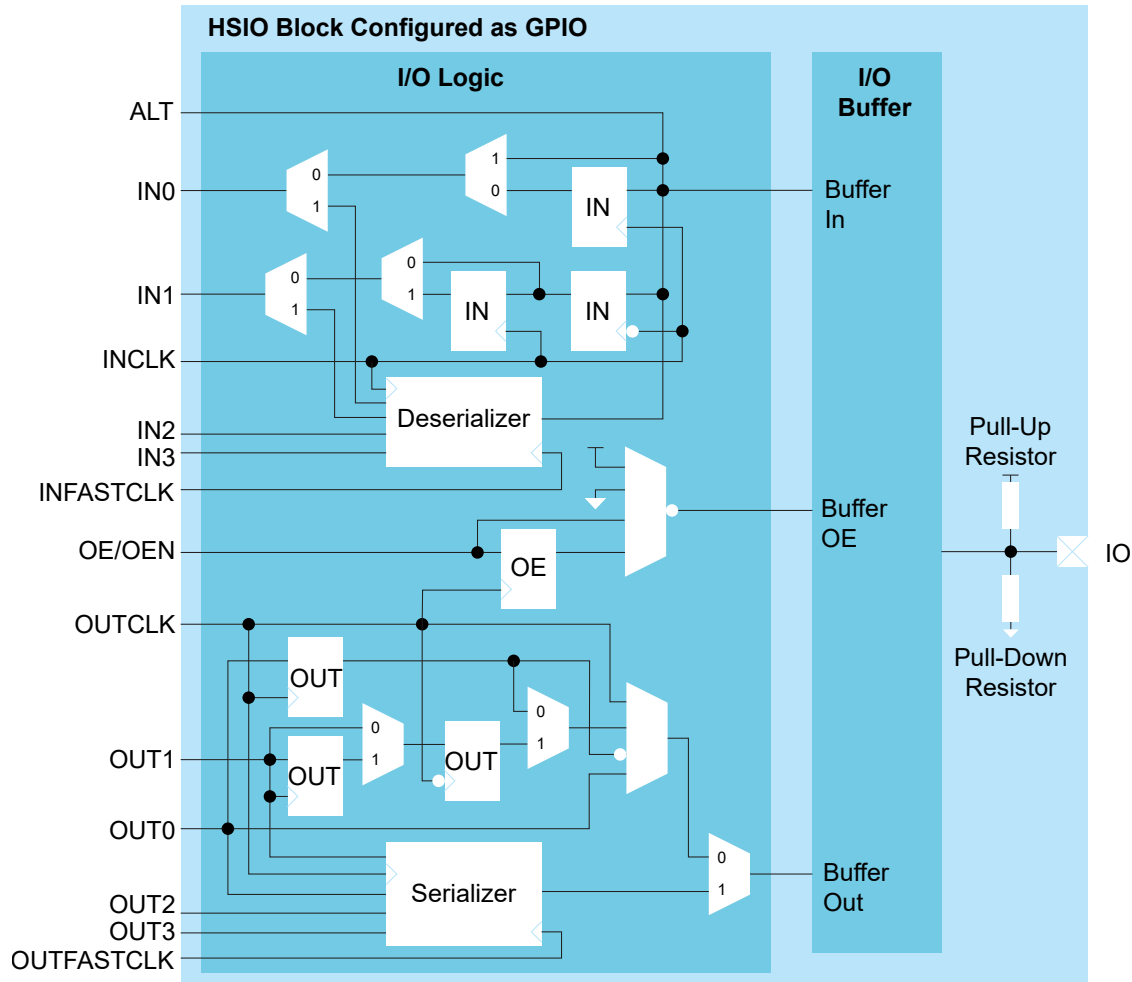


Table 33: HSIO Block Configured as GPIO Signals (Interface to FPGA Fabric)

Signal	Direction	Description
IN[3:0]	Output	Input data from the pad to the core fabric. IN0 is the normal input to the core. In DDIO mode, IN0 is the data captured on the positive clock edge (HI pin name in the Interface Designer) and IN1 is the data captured on the negative clock edge (LO pin name in the Interface Designer). When using the deserializer, the first bit is on IN0 and the last bit is on IN3.
ALT	Output	Alternative input connection for GCLK, PLL_CLKIN, RCLK, PLL_EXTFB, and VREF. (In the Interface Designer, Register Option is none).
OUT[3:0]	Input	Output data to GPIO pad from the core fabric. OUT0 is the normal output from the core. In DDIO mode, OUT0 is the data captured on the positive clock edge (HI pin name in the Interface Designer) and OUT1 is the data captured on the negative clock edge (LO pin name in the Interface Designer). When using the serializer, the first bit is on OUT0 and the last bit is on OUT3.
OE/OEN	Input	Output enable from core fabric to the I/O block. Can be registered. OEN is used in differential mode. Drive it with the same signal as OE.
DLY_ENA	Input	(Optional) Enable the dynamic delay control.
DLY_INC	Input	(Optional) Dynamic delay control. When DLY_ENA = 1, 1: Increments 0: Decrements
DLY_RST	Input	(Optional) Reset the delay counter.
OUTCLK	Input	Core clock that controls the output and OE registers. This clock is not visible in the user netlist.
OUTFASTCLK	Input	Core clock that controls the output serializer.
INCLK	Input	Core clock that controls the input registers. This clock is not visible in the user netlist.
INFASTCLK	Input	Core clock that controls the input serializer.

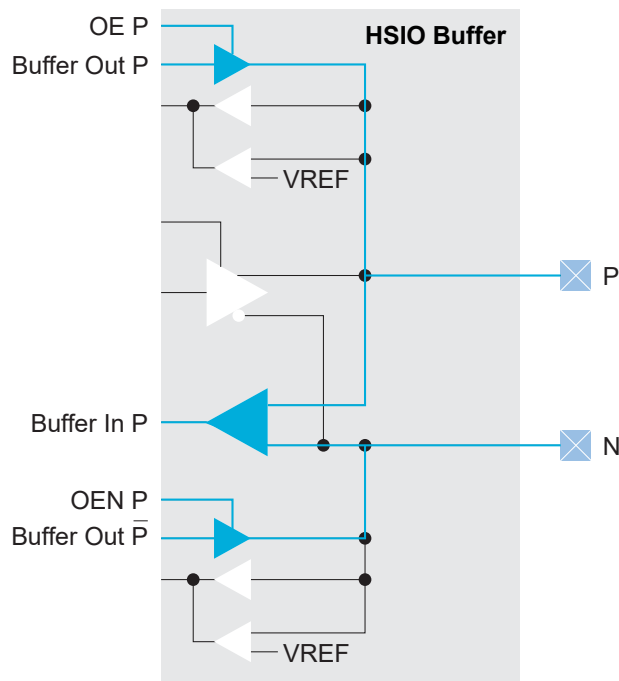
Table 34: GPIO Pads

Signal	Direction	Description
IO (P and N)	Bidirectional	GPIO pad.

The signal path from the pad through the I/O buffer changes depending on the I/O standard you are using. The following figures show the paths for the supported standards. The blue highlight indicates the path.

When using an HSIO with the differential HSTL or differential SSTL standard, you must use both GPIO resources in the HSIO. You use the core interface pins associated with the P resource.

Figure 24: I/O Buffer Path for Differential HSTL and SSTL



Using the GPIO Block

This block defines the functionality of the general-purpose I/O (GPIO) pins. The mode you select determines the GPIO capabilities and which settings you can configure. GPIO modes are: input, output, inout, clkout, and none.

You can assign GPIO to HVIO or HSIO resources. These resources support different I/O standards and have different features. When you check the interface design, the software compares your selections to the resource you assigned to the GPIO block. If the resource does not support your selection(s), the software reports it.

Create a GPIO

To create a new GPIO block, select GPIO in the Design Explorer and then click the Create Block button.

1. Specify the instance name.
2. Choose the Mode (input, output, inout, clkout, or none).
3. Set the options as described in the following sections.
4. **Assign a resource for the signal using the Resource Assigner.**



Note: You can set the default state of unused GPIO. Click the **GPIO(n)** category under Design Explorer. In the Block Editor to the right, select the unused state (**input with weak pull up** or **input with weak pull down**).




Note: When using HSIO pins as GPIO, make sure to leave at least 1 pair of unassigned HSIO pins between any GPIO and HSIO pins in the same bank. This separation reduces noise. The Efinity software issues an error if you do not leave this separation.

Input Mode

Use **input** mode for input signals.

Table 35: Input Mode Options

Option	Choices	Description								
Connection Type	normal, gclk, pll_clkin, pll_extfb, mipi_clkin, rclk, vref	<p>Some pins have alternate functions, and you use this option to choose the function. (This option only applies to pins that have alternate functions. Refer to the data sheet for your FPGA for pin information.) For example, a PLL can use a GPIO with an alternate connection type as a reference clock.</p> <p> Note: If you set the connection type to pll_clkin or mipi_clkin, the signal is also available as a regular input to the core.</p>								
Register Option	register, none	<p>Choose whether the input is registered.</p> <p>If you choose register:</p> <ul style="list-style-type: none"> Define an input clock pin name. Turn clock inversion on or off. Under Double Data I/O Option, select one of the following: <table border="1" data-bbox="690 829 1421 1155"> <tbody> <tr> <td>none</td> <td>Do not use double data I/O.</td> </tr> <tr> <td>normal</td> <td>Data is passed to the core on both the positive and negative clock edges</td> </tr> <tr> <td>resync</td> <td>Data is resynchronized to pass both data signals on the positive clock edge. <code><pin name>_hi</code> is the positive edge and <code><pin name>_lo</code> is the negative edge.</td> </tr> <tr> <td>pipeline</td> <td>Similar to resync except the data for both signals starts on the same clock edge.</td> </tr> </tbody> </table> <ul style="list-style-type: none"> To use the deserializer, turn on Enable Deserialization and specify the serial clock pin name. (You cannot use the deserializer with DDIO.) 	none	Do not use double data I/O.	normal	Data is passed to the core on both the positive and negative clock edges	resync	Data is resynchronized to pass both data signals on the positive clock edge. <code><pin name>_hi</code> is the positive edge and <code><pin name>_lo</code> is the negative edge.	pipeline	Similar to resync except the data for both signals starts on the same clock edge.
none	Do not use double data I/O.									
normal	Data is passed to the core on both the positive and negative clock edges									
resync	Data is resynchronized to pass both data signals on the positive clock edge. <code><pin name>_hi</code> is the positive edge and <code><pin name>_lo</code> is the negative edge.									
pipeline	Similar to resync except the data for both signals starts on the same clock edge.									
Pull Option	none, weak pullup, weak pulldown, dynamic	<p>Specify if you want a pull option.</p> <p>If you choose dynamic, you must also specify the Dynamic Pull Up Enable Pin Name.</p>								
Enable Schmitt Trigger	On or off	Optionally enable a Schmitt trigger.								
Enable Bus Hold	On or off	Optionally enable a bus hold.								
Static Delay Setting	Integer 0-15	<p>For single-ended only. Choose the amount of static delay, each step adds approximately 60 ps of delay.</p> <p>You can only set the static delay for individual signals, not buses.</p>								
	0 - 63	<p>For differential only: 64 steps with approximately 25 ps of delay per step.</p> <p>You cannot use the static delay and dynamic delay simultaneously.</p>								
Enable Dynamic Delay	On or off	<p>For inputs, 64 steps with approximately 25 ps of delay per step. If you enable this option, specify the enable, reset, and control pins as well as the clock pin.</p> <p>You can only set the dynamic delay for individual signals, not buses.</p> <p>A clock is required when using the dynamic delay. If the input register is used, the clock is the same as the input register. Otherwise, you have to define a clock in the dynamic delay group box.</p>								

Output Mode

Use **output** mode for output signals.

Table 36: Output Mode Options

Option	Choices	Description						
Constant Output	none, 1, 0	Choose whether the output is VCC (1) or GND (0). Otherwise, leave this option as none.						
Register Option	none, register, inv_register	<p>Choose whether the output is registered or has an inverted register.</p> <p>If you choose register:</p> <ul style="list-style-type: none"> Define an output clock pin name. Turn clock inversion on or off. Under Double Data I/O Option, select one of the following: <table border="1" data-bbox="732 657 1424 909"> <tbody> <tr> <td>none</td> <td>Do not use double data I/O.</td> </tr> <tr> <td>normal</td> <td>Data is passed to the core on both the positive and negative clock edges</td> </tr> <tr> <td>resync</td> <td>Data is resynchronized to pass both data signals on the positive clock edge. <pin name>_hi is the positive edge and <pin name>_lo is the negative edge.</td> </tr> </tbody> </table> <ul style="list-style-type: none"> To use the serializer, turn on Enable Serialization and specify the serial clock pin name. (You cannot use the serializer with DDIO.) <p>The invert register option (inv_register) does not support DDIO.</p>	none	Do not use double data I/O.	normal	Data is passed to the core on both the positive and negative clock edges	resync	Data is resynchronized to pass both data signals on the positive clock edge. <pin name>_hi is the positive edge and <pin name>_lo is the negative edge.
none	Do not use double data I/O.							
normal	Data is passed to the core on both the positive and negative clock edges							
resync	Data is resynchronized to pass both data signals on the positive clock edge. <pin name>_hi is the positive edge and <pin name>_lo is the negative edge.							
Drive Strength	Depends on I/O standard	<p>Choose the drive strength current in mA.</p> <p>The HVIO and HSIO have different drive strength options depending on the I/O standard you choose. If you change the I/O standard, the Interface Designer resets the drive strength setting if the value is out of range for the selected standard.</p>						
Enable Fast Slew Rate	On or off	Optionally enable slew rate.						
Static Delay Setting	0 - 15	<p>Choose the amount of static delay, each step adds approximately 60 ps of delay.</p> <p>You can only set the static delay for individual signals, not buses.</p>						

Inout Mode

Use **inout** mode for bidirectional signals. Inout mode has the same options for the input and output as the input and output modes.

Inout mode also has an output enable signal (optionally registered) to enable or disable the output buffer. The pin name you specify should be the same as the one you use in your RTL design. Setting the output enable signal to high ("1") in your RTL design enables the output buffer.



Learn more: For information on how to create a tri-state buffer, refer to "How do I create a Tri-State Buffer" in the [Support Center Knowledgebase](#).

Clock Output Mode

Use **clkout** mode for clock output signals. You do not need to name the pin, but you do need to specify the output clock **Pin Name**.

None

Use **none** for unused signals. Specify whether the unused signal should have a weak pullup (default) or pulldown.

Using the GPIO Bus Block

The GPIO bus block is an easy way to add a group of GPIO blocks and make settings for the signal group.

1. Click **Create New Bus**. The Add New Bus wizard opens.
2. Specify a bus name, the width, and the mode (input, output, or inout) and click **Next**.
3. The wizard displays options for input, output, or inout, depending on the mode you selected. Refer to [Using the GPIO Block](#) on page 54 for a description of these options. Make your selections and click **Next**.
4. Review the bus properties and click **Finish**. The software adds the new bus under GPIO.

After you create a bus, you can make additional settings for each signal.

1. Expand **GPIO > <bus name>**.
2. Make any block-specific settings in the Block Editor.
3. [Assign a resource for the signal using the Resource Assigner](#).
4. Save.

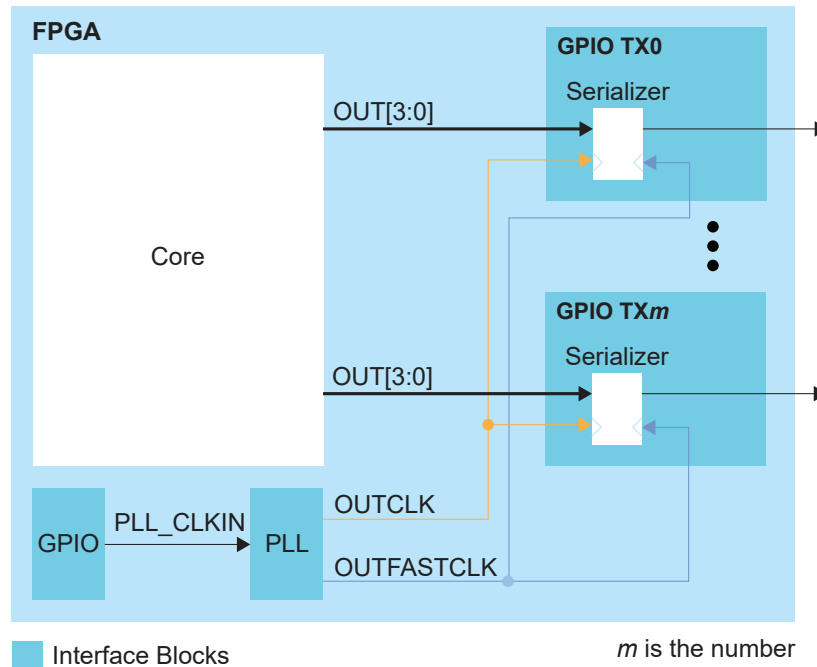


Note: Any changes that you make to individual bus members are over written if you later edit the bus.

Create a TX Serializer Interface

The following figure shows a completed TX serializer interface, the serialization width is always 4 and m is the number of TX lanes.

Figure 25: Complete TX Serializer Interface Block Diagram



Follow these steps to build this interface using the Efinity® Interface Designer.

1. Add a PLL block with the following settings:

Option	Description
Resource	You can use any PLL resource.
Reference Clock Mode	Any
Reference Clock Frequency	Any
Output Clock	Define the output clocks so that you have one for the fast clock (serial) and one for the slow clock (parallel). The fast clock (OUTFASTCLK) should be 4 times faster than the slow clock (OUTCLK). The serial clock phase shift should be between 45 and 135 degrees.

2. Add a GPIO block with these settings to provide the reference clock input to the PLL:

Option	Description
Mode	Input
Pin Name	Any
Connection Type	pll_clkln
GPIO Resource	Assign the dedicated PLL_CLKIN pin that corresponds to the PLL you chose.

3. Add a GPIO block with these settings:

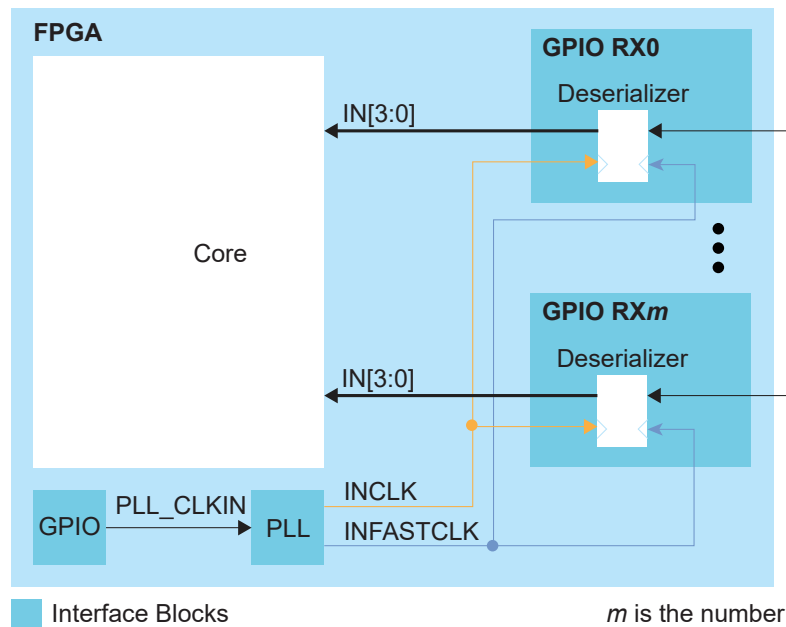
Option	Description
Mode	output
Register Option	register
Enable Serialization	Turn on
Clock Pin Name	Use the slow clock output name that corresponds to the PLL you chose.
Serial Clock Pin Name	Use the fast clock output name that corresponds to the PLL you chose.

- Repeat step 3 for each TX serializer you want to implement.

Create a RX Deserializer Interface

The following figure shows a completed RX deserializer interface, the deserialization width is 4 and m is the number of RX lanes.

Figure 26: Complete RX Deserializer Interface Block Diagram



Follow these steps to build this interface using the Efinity® Interface Designer.

- Add a PLL block with the following settings:

Option	Description
Resource	You can use any PLL resource.
Reference Clock Mode	Any
Reference Clock Frequency	Any
Output Clock	Define the output clocks so that you have one for the fast clock (serial) and one for the slow clock (parallel).

Option	Description
	The fast clock (INFASTCLK) should be 4 times faster than the slow clock (INCLK). . The serial clock phase shift should be between 45 and 135 degrees.

2. Add a GPIO block with these settings to provide the reference clock input to the PLL:

Option	Description
Mode	Input
Pin Name	Any
Connection Type	pll_clkin
GPIO Resource	Assign the dedicated PLL_CLKIN pin that corresponds to the PLL you chose.

3. Add a GPIO block with these settings:

Option	Description
Mode	input
Register Option	register
Enable Serialization	Turn on
Clock Pin Name	Use the slow clock output name that corresponds to the PLL you chose.
Serial Clock Pin Name	Use the fast clock output name that corresponds to the PLL you chose.

4. Repeat step 3 for each RX deserializer you want to implement.

Design Check: GPIO Messages

When you check your design, the Interface Designer applies design rules to your configuration settings. The following tables show some of the error and warning messages you may encounter and explains how to fix them.

bus_rule_members_consistent (error)

Message	Bus <name> has mismatch properties with its members Members of bus <name> has inconsistent shared pin/register settings with <member>: <inconsistent_members_name>
To fix	The properties you have set for the bus are inconsistent with the settings for the bus members. Review the settings and fix any mismatches.

bus_rule_name (error)

Message	Bus name is empty Valid characters are alphanumeric characters with dash and underscore only
To fix	Specify a valid bus name.

gpio_rule_input_mode (error)

Message	For input mode, input must be configured
To fix	You need to configure the input parameters.
Message	For input mode, input pin name must be configured
To fix	Specify a name for the input pin.
Message	Input pin name with square bracket does not match a bus name with index
To fix	You probably used a bracket [or] in the pin name. Rename the pin without brackets.

gpio_rule_input_register (error)

Message	Input clock pin name is empty Input clock pin name has illegal character
To fix	If you are using the register option, you need to specify a valid clock pin name.

gpio_rule_input_register (warning)

Message	Input clock pin name is empty Input clock pin name has illegal character
To fix	If you have a GPIO block in inout mode, you get this message if you do not specify an input pin name or if you use invalid characters in the name. You should specify a name to use the pin as bidirectional.
Message	Alternate input connection cannot be registered
To fix	When you are using an alternate connection type, you cannot use the register. Set Register Option to none .

gpio_rule_output_mode (error)

Message	For output mode, output must be configured
To fix	You need to configure the output parameters.
Message	For output mode without constant output, output pin name must be configured
To fix	For GPIO output blocks, you can drive them as 0 or a 1 with the Constant Output option. In that mode, you do not need to specify an output pin name. If you are using Constant Output set to none (as you would for a regular output pin), you need to specify a pin name.
Message	Output pin name with square bracket does not match a bus name with index
To fix	You probably used a bracket [or] in the pin name. Rename the pin without brackets.

gpio_rule_inout_mode (error)

Message	For inout mode, both output and output enable must be configured
To fix	When using a GPIO block in inout mode, you need to set the options for the output and output enable (as well as the input).
Message	For inout mode, both output pin name and output enable pin name must be configured
To fix	Specify a valid name for the output pin and output enable pin.
Message	Output enable pin name with square bracket does not match a bus name with index
To fix	You probably used a bracket [or] in the pin name. Rename the pin without brackets.

gpio_rule_clkout_mode (error)

Message	For clkout mode, output must be configured
To fix	When using a GPIO block in clkout mode, you need to set all of the options.
Message	For clkout mode, output clock pin name must be configured
To fix	When using a GPIO block in clkout mode, you need to specify a valid pin name for the output clock.

gpio_rule_output_clock (error)

Message	Output is registered but clock pin name is empty
To fix	If you choose register or inv_register as the Register Option , then you also need to specify an output clock pin name.
Message	Output enable is registered but clock pin name is empty
To fix	If you are using a GPIO in inout mode, you need to specify the output enable pin name if you set the output enable to register .
Message	Output is registered but clock pin name has illegal character Output enable is registered but clock pin name has illegal character
To fix	Use valid names for these pins.
Message	Output clock pin name is not the same as output enable clock pin name
To fix	For a GPIO block in inout mode, you need to use the same pin names for the output clock pin and the output enable clock pin.
Message	Output clock inversion is not the same as output enable clock inversion
To fix	For a GPIO block in inout mode, if you invert the output clock pin you also need to invert the output enable clock pin. Similarly, if you do not invert the output clock pin, you cannot invert the output enable clock pin.

gpio_rule_unused_mode (error)

Message	For unused (none) mode, its state needs to be configured
To fix	If you set a GPIO block mode to none , you need to set the Unused State . The default is input with weak pull-up. You can change this setting globally by clicking the GPIO category in the Design Explorer and setting the Unused State option.

gpio_rule_input_alt_conn (error)

Message	Connection type <type> is not supported by <resource>
To fix	If you want to use the alternate function of a GPIO block, you need to choose a resource that supports it. For example, global clock (GCLK) is only supported on the P pin. You can filter for resources by alternate function in the Resource Assigner.
Message	<resource> only supports normal connection type
To fix	You need to choose a different connection type or assign a different resource that supports the connection type you want to use. You can filter for resources by alternate function in the Resource Assigner.

gpio_rule_input_alt_conn (warning)

Message	Connection type <type> is not supported by <resource>
To fix	For a GPIO block in inout mode, you get a warning if you do not specify the input pin name. Specify the input pin name to use the block as bidirectional or leave it empty to use it as open-drain.
Message	<resource> only supports normal connection type
To fix	The software thinks you are creating an open-drain if you leave the input pin name empty, so this choice is valid, but it lets you know that you made this selection in case you really want to use it as bidirectional.

gpio_rule_ddio_resource (error)

Message	Double Data I/O must be assigned to resource that supports DDIO
To fix	To use the DDIO feature, you need to pick a resource that supports it. You can filter for resources by DDIO in the Resource Assigner Features column.

gpio_rule_ddio_resource (warning)

Message	Double Data I/O must be assigned to resource that supports DDIO
To fix	You get this error if you use a resource that does not support DDIO but have not yet specified an input pin name. Either turn off DDIO or choose a resource that supports it. You can filter for resources by DDIO in the Resource Assigner Features column.

gpio_rule_ddio_input (error)

Message	Input with DDIO requires register option to be set
To fix	If you are using DDIO, you must set the the Register Option to register .

gpio_rule_ddio_input (warning)

Message	Input with DDIO requires register option to be set
To fix	For a GPIO block in inout mode, if the Double-Data I/O Option is other than none , you need to choose register as the Register Option . You get this warning when you have set some options for the input pin but have not yet specified a pin name.

gpio_rule_ddio_output (error)

Message	Output with DDIO requires register option to be set
To fix	To use DDIO you must set the Double-Data I/O Option set to something other than none .

gpio_rule_ddio_pin_name (error)

Message	Double Data I/O must have both HI and LO input pin names defined
To fix	When using DDIO, you need to specify pin names for the Pin Name (HI) and Pin Name (LO) .

gpio_rule_ddio_pin_name (warning)

Message	Double Data I/O must have both HI and LO input pin names defined
To fix	For a GPIO block in inout mode, if the Double-Data I/O Option is other than none , you need to specify pin names for the Pin Name (HI) and Pin Name (LO) . You get this warning when you have set some options for the input pin but have not yet specified these pin names.

gpio_rule_alt_conn (warning)

Message	Connection type <type> must be used by valid PLL
To fix	The GPIO is connected to a PLL clock input but is the resource you assigned does not support the pll_clkln alternate function. Choose a different resource that supports it. You can filter resources by alternate function in the Resource Assigner.

Message	Connection type <type> cannot be used on an unbonded resource
To fix	You get this error if the resource you choose is not available in the FPGA/package combination you are using. Choose another resource.

Message	pll_clkln connection to PLL clock source not being used in <instance>
To fix	The GPIO block is set to be a PLL reference clock (pll_clkln connection type) but the PLL is not configured to use it. Make sure that the clock you are choosing in the PLL is associated with this GPIO's resource.

Message	pll_clkln connection to PLL clock source but none of the external clock source in PLL <instance> is selected
To fix	The GPIO block is set to be a PLL reference clock (pll_clkln connection type) but the PLL is not configured to use it. In the PLL block, choose external or dynamic as the Clock Source and make sure that the clock you are choosing is associated with this GPIO's resource.

Message	pll_clkln connection to PLL clock source but PLL Clock source on <instance> is set to core
To fix	The GPIO block is set to be a PLL reference clock (pll_clkln connection type) but the PLL is not configured to use it. In the PLL block, choose external or dynamic as the Clock Source and make sure that the clock you are choosing is associated with this GPIO's resource.

Message	pll_extfb connection to PLL external feedback pin on <instance> is not set to external
To fix	The GPIO block is set to be external feedback for the PLL (pll_extfb connection type) but the PLL is not configured to use it. In the PLL Clock Calculator, choose External as the Feedback Mode.

gpio_rule_sample_device (error)

Message	Unsupported features in ES device: <features>
To fix	You get this error when you try to use a feature that is not supported in an engineering sample (ES) version of the FPGA.

gpio_rule_resource (error)

Message	Resource name is empty Resource is not a valid GPIO device instance
To fix	You need to choose a valid resource.

gpio_rule_io_standard_bank (warning)

Message	Mismatch voltage in I/O standard assignment in bank (<voltage>) and instance (<io_std>)
To fix	You get this error when the voltage for the I/O bank does not match the I/O standard you chose for the GPIO. Either change the I/O bank voltage or choose a compatible I/O standard.

gpio_rule_io_standard_compatibility (error)

Message	I/O standard <value> is not supported in bank <bank>
To fix	You get this error when you choose an I/O standard that is not supported in the bank. You need to choose another I/O standard or pick a resource in another bank. See Titanium I/O Banks on page 24 for the voltages supported in each bank and Types of GPIO on page 44 for the I/O standards the GPIO support.

gpio_rule_drive_strength (error)

Message	Valid drive strength for <iostd> is: <list>
To fix	Choose the drive strength based on the recommendation in the message.
Message	Invalid drive strength <value> for <iostd>. Check for valid I/O standard
To fix	Confirm the drive strengths that are allowed for the I/O standard you want to use and then change the setting accordingly.

gpio_rule_ddio_serial (error)

Message	DDIO has to be none when serialization is enabled on both input and output DDIO on input has to be none when deserialization is enabled DDIO on output has to be none when serialization is enabled
To fix	You cannot use DDIO and serialization or deserialization at the same time. Turn one of them off.

gpio_rule_transmit_toggling (warning)

Message	Bank <name> has <int> GPIO in inout/output mode that exceed max limit of <int> which can result in LVTTTL simultaneous switching noise
To fix	If you use more than 6 HVIO pins as GPIO in output or inout mode in the same bank, it can cause switching noise. Instead, use resources from another bank.

gpio_rule_serial_input_clk (error)

Message	Input clock inversion is not allowed with deserialization enabled
To fix	If you use deserialization, you cannot also invert the clock. Turn off the Inverted option.

gpio_rule_serial_output_clk (error)

Message	Output clock inversion is not allowed with serialization enabled
To fix	If you use serialization, you cannot also invert the clock. Turn off the Inverted option.

gpio_rule_static_input_delay (error)

Message	Static delay, <int> is outside of limit (0-15) for non-Differential HSTL/SSTL I/O Standard Static delay, <int> is outside of limit (0-63) for Differential HSTL/SSTL I/O Standard
To fix	The static input delay you selected is not valid. Use a number in the range specified in the message.

gpio_rule_io_standard_valid (error)

Message	I/O standard <value> is not supported in bank <bank>
To fix	You get this error when you choose an I/O standard that is not supported in the bank. You need to choose another I/O standard or pick a resource in another bank. See Titanium I/O Banks on page 24 for the voltages supported in each bank and Types of GPIO on page 44 for the I/O standards the GPIO support.

gpio_rule_hsio_usage (error)

Message	HSIO resource <name> was assigned to GPIO, LVDS and MIPI LANE HSIO resource <name> was assigned to both GPIO and LVDS HSIO resource <name> was assigned to both GPIO and MIPI LANE
To fix	You get this error if you try to use the same resource for more than one block type. Remove blocks so that you only are only using the resource once.

gpio_rule_io_standard_stl (error)

Message	This resource is reserved as vref for bank <name>. Use a different resource to configure single ended HSTL/SSTL
To fix	Some resources can be used as the VREF for an I/O standard. If you are using an I/O standard that uses a VREF pin, you must use this resource as a VREF. Choose another resource for the GPIO function.

Message	GPIO <name> has to be configured as vref input mode to support I/O standard <iostd> GPIO <name> has to be configured as vref input connection type to support I/O standard <iostd> GPIO <name> has to be configured as vref input to support I/O standard <iostd>
To fix	If you are using an I/O standard that uses a VREF pin, you must use this resource as a VREF. Configure the GPIO as an input and choose vref as the Connection Type .

Message	I/O Standard <iostd> cannot be used due to unbonded vref resource on the same bank <name> I/O Standard <iostd> cannot be used due to vref resource not bonded out
To fix	If a VREF pin is not available in the I/O bank (e.g., it is not in the FPGA/package you chose), you cannot use an I/O standard that requires it. Instead choose a different I/O standard for the GPIO or a resource in a different I/O bank that has a VREF pin bonded out. .

gpio_rule_io_standard_stl (warning)

Message	Skip checking Vref requirement on a single-ended input configuration: input path is not used
To fix	You get this warning when the GPIO is in inout mode but you have not specified an input pin name.

gpio_rule_io_standard_differential (error)

Message	GPIO resource, <name> with differential I/O standard was configured for multiple use Differential I/O standard can only be assigned to pad P resource
To fix	You cannot use differential HSTL/SSTL for the N resource. Change the resource to a P one.
Message	Differential I/O standard is not valid on GPIO resource <name> due to the corresponding N resource not bonded
To fix	To use a differential I/O standard, both the N and P resources must be available. If the N resource is not bonded out, then you cannot use a differential standard. Choose another pair of N and P resources.

gpio_rule_dynamic_delay (error)

Message	Input dynamic delay is not supported in non-P resource, <resname>
To fix	The HSIO N resource does not support dynamic delay. Either change the delay to static or pick a P resource.
Message	Clock pin name in input dynamic delay is empty
To fix	When using the dynamic delay, you need to specify a clock input pin name.

gpio_rule_serialization (error)

Message	Register Option has to be set to register when using serialization
To fix	You cannot use serialization unless the Register Option is set to register . Either change the option or do not use serialization.
Message	Serialization cannot be used with DDIO Option
To fix	If you are using serialization, you cannot also use DDIO. Either disable serialization or set Double Data I/O Option to none .
Message	Serial and parallel clock names are required to be non-empty with serialization
To fix	You need to specify the parallel and serial clock names if you are using serialization.
Message	Serialization is not supported due to PLL is not available
To fix	When the GPIO is using the serializer, the serial and parallel clock signals must come from a PLL. You need to make two PLL clock outputs available to the GPIO.
Message	Serial and parallel clocks cannot be the same clock
To fix	Use different PLL output clocks for the signals.
Message	Serial and parallel clock names are not PLL output clocks Serial clock name is not a PLL output clock Parallel clock name is not a PLL output clock
To fix	When the GPIO is using the serializer, the serial and parallel clock signals must come from a PLL. You need to use two PLL clock outputs, one for serial and one for parallel.
Message	Serial and parallel clocks are not from the same PLL instance
To fix	Use PLL output clocks from the same PLL for the serial and parallel clocks.
Message	Serial clock phase shift <deg> is not one of the valid values: 45, 90, 135
To fix	Choose one of the values in the message.
Message	One of the clock frequencies is 0
To fix	Change the clock frequency to be something other than zero.
Message	Serial clock frequency has to be 4 times faster than parallel clock
To fix	Change the PLL output clock frequencies such that the serial clock is 4 times faster than the parallel clock.

gpio_rule_deserialization (error)

Message	Register Option has to be enabled when using deserialization
To fix	You cannot use deserialization unless the Register Option is set to register . Either change the option or do not use deserialization.
Message	Deserialization cannot be used with DDIO Option
To fix	If you are using serialization, you cannot also use DDIO. Either disable deserialization or set Double Data I/O Option to none .
Message	Serial and parallel clock names are required to be non-empty with deserialization
To fix	You need to specify the parallel and serial clock names if you are using deserialization.
Message	Deserialization is not supported due to PLL is not available
To fix	When the GPIO is using the deserializer, the serial and parallel clock signals must come from a PLL. You need to make two PLL clock outputs available to the GPIO.
Message	Serial and parallel clocks cannot be the same clock
To fix	Use different PLL output clocks for the signals.
Message	Serial and parallel clock names are not PLL output clocks Serial clock name is not a PLL output clock Parallel clock name is not a PLL output clock
To fix	When the GPIO is using the deserializer, the serial and parallel clock signals must come from a PLL. You need to use two PLL clock outputs, one for serial and one for parallel.
Message	Serial and parallel clocks are not from the same PLL instance
To fix	Use PLL output clocks from the same PLL for the serial and parallel clocks.
Message	Serial clock phase shift <deg> is not one of the valid values: 45, 90, 135
To fix	Choose one of the values in the message.
Message	One of the clock frequencies is 0
To fix	Change the clock frequency to be something other than zero.
Message	Serial clock frequency has to be 4 times faster than parallel clock
To fix	Change the PLL output clock frequencies such that the serial clock is 4 times faster than the parallel clock.

gpio_rule_bus_hold (error)

Message	Bus hold is only supported on 1.2/1.5/1.8 V LVCMOS I/O standard HSIO only
To fix	You cannot use the bus hold option for the I/O standard you chose. Either turn off Enable Bus Hold or use a different I/O standard.

gpio_rule_differential_stl_inout (error)

Message	For inout mode with differential STL, output enable (N) pin name must be configured
To fix	Specify the output enable pin name.

gpio_rule_cfg_io_standard_valid (error)

Message	Unsupported I/O standard
To fix	The I/O standard you chose is not supported. Choose another one. Refer to Types of GPIO on page 44.

gpio_rule_cfg_slew_rate (error)

Message	Resource <name> does not support Slew Rate feature. It will be ignored
To fix	The Titanium HVIO pins do not support slew rate. Either disable that option or choose another resource.

gpio_rule_cfg_dyn_delay (error)

Message	Resource <name> does not support Dynamic Delay feature.
To fix	The resource you used does not support dynamic delay. Refer to Features for HVIO and HSIO Configured as GPIO on page 46 for which GPIO support that feature. Either turn off dynamic delay or choose another resource that supports it.

gpio_rule_cfg_bus_hold (warning)

Message	Resource <name> does not support Bus Hold feature. It will be ignored
To fix	Not all resources support bus hold. Refer to Features for HVIO and HSIO Configured as GPIO on page 46 for which GPIO support that feature. Either turn off bus hold or choose another resource that supports it.

gpio_rule_cfg_dyn_pullup (warning)

Message	Resource <name> does not support Dynamic Pullup feature. It will be ignored
To fix	Not all resources support dynamic pullup. Refer to Features for HVIO and HSIO Configured as GPIO on page 46 for which GPIO support that feature. Change the Pull Option or choose another resource that supports it.

gpio_rule_cfg_serialization (error)

Message	Resource <name> does not support Serialization feature.
To fix	Not all resources support serialization. Refer to Features for HVIO and HSIO Configured as GPIO on page 46 for which GPIO support that feature. Turn off serialization or choose another resource that supports it.

gpio_rule_cfg_deserialization (error)

Message	Resource <name> does not support Deserialization feature.
To fix	Not all resources support deserialization. Refer to Features for HVIO and HSIO Configured as GPIO on page 46 for which GPIO support that feature. Turn off deserialization or choose another resource that supports it.

LVDS Interface

Contents:

- [HSIO Configured as LVDS](#)
- [Using the LVDS Block](#)
- [Create an LVDS TX Interface](#)
- [Create an LVDS RX Interface](#)
- [Design Check: LVDS Errors and Warnings](#)

Each HSIO block can use a pair of I/O pins as an LVDS receiver (RX), transmitter (TX), or bidirectional (RX/TX) signal.

HSIO Configured as LVDS

You can configure each HSIO block in RX, TX, or bidirectional LVDS mode. As LVDS, the HSIO has these features:

- Programmable V_{OD} , depending on the I/O standard used.
- Programmable pre-emphasis.
- Up to 1.5 Gbps.
- Programmable $100\ \Omega$ termination to save power (you can enable or disable it at runtime).
- LVDS input enable to dynamically enable/disable the LVDS input.
- Support for full rate or half rate serialization.
- Up to 10-bit serialization to support protocols such as 8b10b encoding.
- Programmable delay chains.
- Optional 8-word FIFO for crossing from the parallel (slow) clock to the user's core clock to help close timing (RX only).
- Dynamic phase alignment (DPA) that automatically eliminates skew for clock to data channels and data to data channels by adjusting a delay chain setting so that data is sampled at the center of the bit period. The DPA supports full-rate serialization mode only.

Table 37: Full and Half Rate Serialization

Mode	Description	Example
Full rate clock	In full rate mode, the fast clock runs at the same frequency as the data and captures data on the positive clock edge.	Data rate: 800 Mbps Serialization/Deserialization factor: 8 Slow clock frequency: 100 Mhz (800 Mbps / 8) Fast clock frequency: 800 Mhz
Half rate clock	In half rate mode, the fast clock runs at half the speed of the data and captures data on both clock edges.	Data rate: 800 Mbps Serialization / Deserialization factor: 8 Slow clock frequency: 100 Mhz (800 Mbps / 8) Fast clock frequency: 400 Mhz (800 / 2)

You use a PLL to generate the serial (fast) and parallel (slow) clocks for the LVDS pins. The slow clock runs at the data rate divided by the serialization factor.

LVDS RX

You can configure an HSIO block as one LVDS RX signal.

Figure 27: LVDS RX Interface Block Diagram

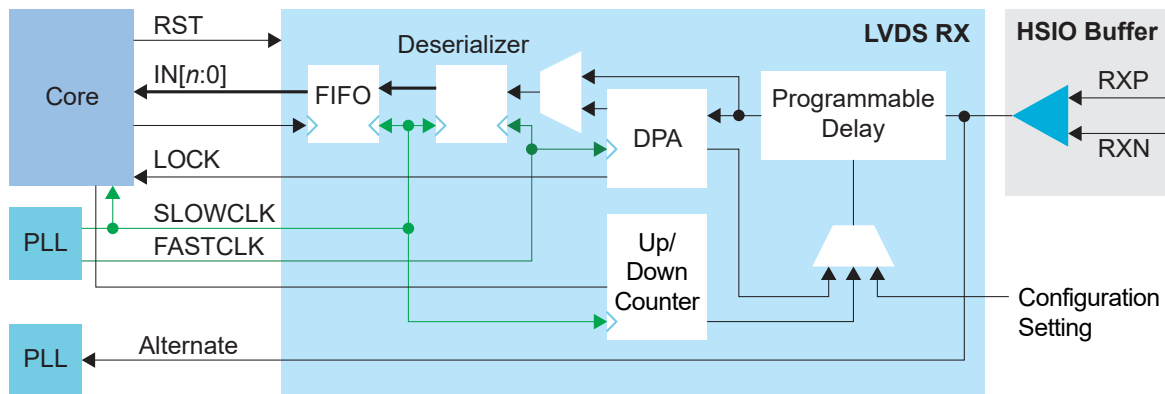
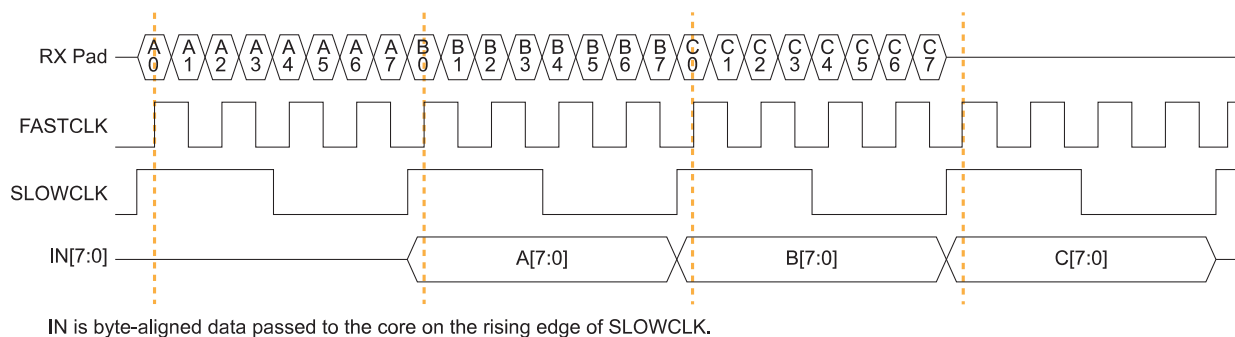


Table 38: LVDS RX Signals (Interface to FPGA Fabric)

Signal	Direction	Clock Domain	Description
IN[9:0]	Output	SLOWCLK	Parallel input data to the core. The width is programmable.
ALT	Output		Alternate input, only available for an LVDS RX resource in bypass mode (deserialization width is 1; alternate connection type). Alternate connections are PLL_CLKIN, PLL_EXTFB, GCLK, and RCLK.
LOCK	Output		(Optional) When DPA is enabled, this signal indicates that the DPA has achieved training lock and data can be passed.
FIFO_EMPTY	Output	FIFOCLK	(Optional) When FIFO is enabled, this signal indicates that the FIFO is empty.
SLOWCLK	Input	-	Parallel (slow) clock.
FASTCLK	Input	-	Serial (fast) clock.
FIFOCLK	Input	-	(Optional) Core clock to read from the FIFO.
FIFO_RD	Input	FIFOCLK	(Optional) Enables FIFO to read.
RST	Input	FIFOCLK SLOWCLK	(Optional) Asynchronous. Resets the FIFO and serializer. If the FIFO is enabled, it is relative to FIFOCLK; otherwise it is relative to SLOWCLK.
ENA	Input	-	Dynamically enable or disable the LVDS input buffer. Can save power when disabled. 1: Enabled 0: Disabled
TERM	Input	-	Enables or disables termination in dynamic termination mode. 1: Enabled 0: Disabled
DLY_ENA	Input	SLOWCLK	(Optional) Enable the dynamic delay control or the DPA circuit, depending on which one is enabled.
DLY_INC	Input	SLOWCLK	(Optional) Dynamic delay control. Cannot be used with DPA enabled. When DLY_ENA is 1: 1: Increments 0: Decrements
DLY_RST	Input	SLOWCLK	(Optional) Reset the delay counter or the DPA circuit, depending on which one is enabled.

The following waveform shows the relationship between the fast clock, slow clock, RX data coming in from the pad, and byte-aligned data to the core.

Figure 28: LVDS RX Timing Example Serialization Width of 8 (Half Rate)



LVDS TX

You can configure an HSIO block as one LVDS TX signal. LVDS TX can be used in the serial data output mode or reference clock output mode.

Figure 29: LVDS TX Interface Block Diagram

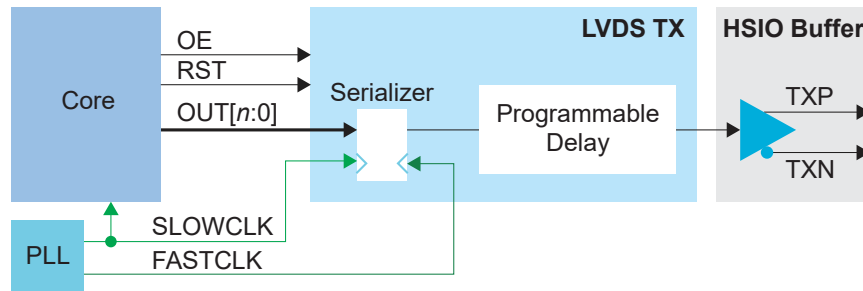
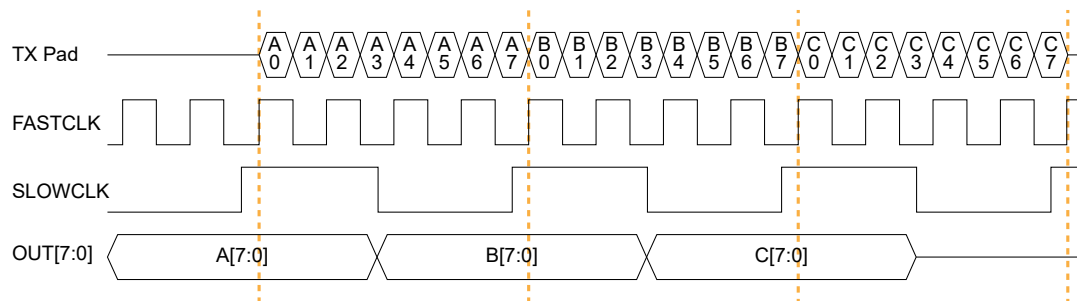


Table 39: LVDS TX Signals (Interface to FPGA Fabric)

Signal	Direction	Clock Domain	Description
OUT[9:0]	Input	SLOWCLK	Parallel output data from the core. The width is programmable.
SLOWCLK	Input	-	Parallel (slow) clock.
FASTCLK	Input	-	Serial (fast) clock.
RST	Input	SLOWCLK	(Optional) Resets the serializer.
OE	Input	-	(Optional) Output enable signal.

The following waveform shows the relationship between the fast clock, slow clock, TX data going to the pad, and byte-aligned data from the core.

Figure 30: LVDS Timing Example Serialization Width of 8 (Half Rate)



OUT is byte-aligned data passed from the core on the rising edge of SLOWCLK.

LVDS Bidirectional

You can configure an HSIO block as one LVDS bidirectional signal. You must use the same serialization for the RX and TX.

Figure 31: LVDS Bidirectional Interface Block Diagram

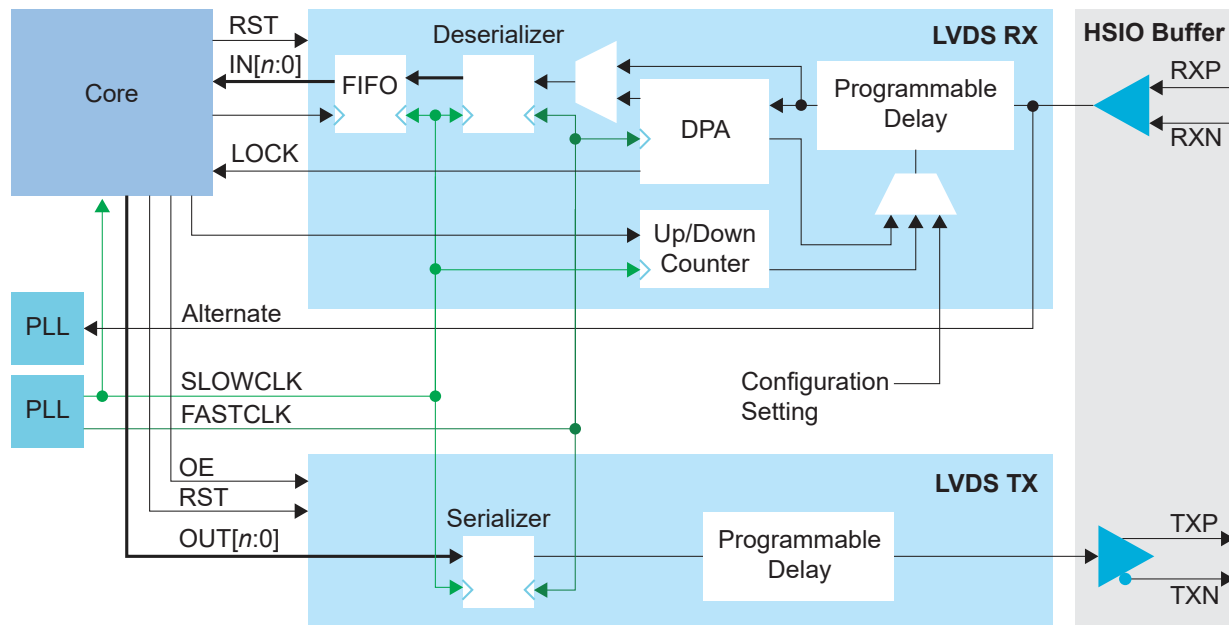


Table 40: LVDS Bidirectional Signals (Interface to FPGA Fabric)

Signal	Direction	Clock Domain	Description
IN[9:0]	Output	SLOWCLK	Parallel input data to the core. The width is programmable.
LOCK	Output	-	(Optional) When DPA is enabled, this signal indicates that the DPA has achieved training lock and data can be passed.
FIFO_EMPTY	Output	FIFOCLK	(Optional) When the FIFO is enabled, this signal indicates that the FIFO is empty.
INSLOWCLK	Input	-	Parallel (slow) clock for RX.
INFASTCLK	Input	-	Serial (fast) clock for RX.
FIFOCLK	Input	-	(Optional) Core clock to read from the FIFO.
FIFO_RD	Input	FIFOCLK	(Optional) Enables FIFO to read.
INRST	Input	FIFOCLK SLOWCLK	(Optional) Asynchronous. Resets the FIFO and RX serializer. If the FIFO is enabled, it is relative to FIFOCLK; otherwise it is relative to SLOWCLK.
ENA	Input	-	Dynamically enable or disable the LVDS input buffer. Can save power when disabled. 1: Enabled 0: Disabled
TERM	Input	-	Enables or disables termination in dynamic termination mode. 1: Enabled 0: Disabled

Signal	Direction	Clock Domain	Description
DLY_ENA	Input	SLOWCLK	(Optional) Enable the dynamic delay control or the DPA circuit, depending on which one is enabled.
DLY_INC	Input	SLOWCLK	(Optional) Dynamic delay control. Cannot be used with DPA enabled. When DLY_ENA is 1, 1: Increments 0: Decrements
DLY_RST	Input	SLOWCLK	(Optional) Reset the delay counter or the DPA circuit, depending on which one is enabled.
OUT[9:0]	Input	SLOWCLK	Parallel output data from the core. The width is programmable.
OUTSLOWCLK	Input	-	Parallel (slow) clock for TX.
OUTFASTCLK	Input	-	Serial (fast) clock for TX.
OUTRST	Input	SLOWCLK	(Optional) Resets the TX serializer.
OE	Input	-	Output enable signal.

LVDS Pads

Table 41: LVDS Pads

Signal	Direction	Description
P	Output	Differential pad P.
N	Output	Differential pad N.

Using the LVDS Block

The LVDS block defines the functionality of the LVDS pins. You can choose whether the block is a transmitter (TX), receiver (RX), or bidirectional.

LVDS TX

Table 42: LVDS TX Options

Option	Choices	Description
Instance Name	User defined	Enter a name.
LVDS Resource	Resource list	Choose a resource.
Output Differential Type	lvds	Use for LVDS, RSDS, and mini-LVDS.
	sublvds	Use for subLVDS.
	custom	Choose this option when you want to use a VREF pin to specify the differential. Set the GPIO input connection type to vref . Choose a GPIO pin with that supports VREF in the same bank as the LVDS TX resource.
Output Differential, VOD	Typical, large, small	The actual voltage depends on this setting and the differential type and is shown in the Block Summary Value field.
Output-Pre-Emphasis	high, medium-high, medium-low, low	Choose an output pre-emphasis setting.
Mode	serial data output	Use the transmitter as a simple output buffer or serialized output.
	reference clock output	Use the transmitter as a clock output. Specify the serial and parallel clocks. When choosing this mode, the serialization width should match the serialization for the rest of the LVDS bus.
Output Pin/Bus Name	User defined	Output pin or bus that feeds the LVDS transmitter parallel data. The width should match the serialization factor.
Output Enable Pin Name	User defined	Use with serial data output mode.
Enable Serialization	On	Use as a simple buffer.
	Off	Use as an LVDS serializer: <ul style="list-style-type: none"> Optionally enable half rate serialization. Choose a value of 2, 3, 4, 5, 6, 7, 8, or 10 (1 is a simple buffer). A value of 9 is not legal. Specify the serial clock and parallel clock. Specify reset pin name.
Static Mode Delay Setting	0 - 63	Choose the amount of static delay, each step adds approximately 25 ps of delay.

The maximum LVDS rate is 1.5 Gbps.

- *Half rate calculation*—serial clock frequency = parallel clock frequency * (serialization / 2)
- *Full rate calculation*—serial clock frequency = parallel clock * serialization

The serial clock must have a phase shift that is between 45 degrees and 135 degrees. Both clocks must come from the same PLL.

The serial clock (also known as the fast clock) outputs data to the pin, the parallel clock (also known as the slow clock) transfers it from the core. An equation defines the relationship between the clocks. For LVDS TX the parallel clock captures data from the core and the serial clock outputs it to the LVDS buffer.

In half-rate mode, new data is output on both edges of the serial clock, in full rate mode it is only on the rising (positive) edge.

LVDS RX

Table 43: LVDS RX Options

Option	Choices	Description
Instance Name	User defined	Enter a name.
LVDS Resource	Resource list	Choose a resource.
Connection Type	normal	LVDS RX function.
	pll_clkln	Alternate function. Use as PLL reference clock.
	pll_extfb	Alternate function. Use as PLL external feedback.
	gclk	Alternate function. Use as global clock.
	rclk	Alternate function. Use as regional clock.
Input Pin/Bus Name	User defined	Input pin or bus that feeds the LVDS transmitter parallel data. The width should match the deserialization factor.
Dynamic Enable Pin Name	User defined	Dynamically enables or disables the LVDS RX buffer. Disabling the buffer can reduce power consumption when the pin is not in use.
Enable Common Mode Driver	On or off	If you implement an AC coupled connection, turn on this option. For a typical DC coupled connection, leave this option off.
Termination	on, off, dynamic	For dynamic , specify the pin that controls the dynamic termination.
Enable Deserialization	Off	Use as a simple buffer.
	On	Use as an LVDS deserializer: <ul style="list-style-type: none"> Optionally enable half rate serialization. Choose a width of 2, 3, 4, 5, 6, 7, 8, or 10 (1 is a simple buffer). A width of 9 is not legal. Specify the serial clock and parallel clock. Optionally turn on Enable Clock Crossing FIFO, which uses a FIFO to cross between the slow clock and the user core clock. Specify the FIFO read, clock, and empty pin names.
Delay Mode	static	Integer from 0 - 63. Each step adds approximately 25 ps of delay.
	dynamic	Specify the pin names to control the dynamic delay.

The serial clock (also known as the fast clock) captures data from the pin, the parallel clock (also known as the slow clock) transfers it to the core. An equation defines the relationship between the clocks.

The maximum LVDS rate is 1.5 Gbps.

- *Half rate calculation*—serial clock frequency = parallel clock frequency * (deserialization / 2)
- *Full rate calculation*—serial clock frequency = parallel clock * deserialization

The serial clock must have a phase shift that is between 45 and 135 degrees. Both clocks must come from the same PLL.

LVDS Bidirectional

The LVDS bidirectional block has the same options and choices as the LVDS RX and TX blocks.



Important: You must use the same value for the serialization/deserialization.

PLL Requirements for Serial and Parallel Clocks

With Titanium FPGAs, you need to use the output clocks from specific PLLs as the LVDS serial and parallel clocks.

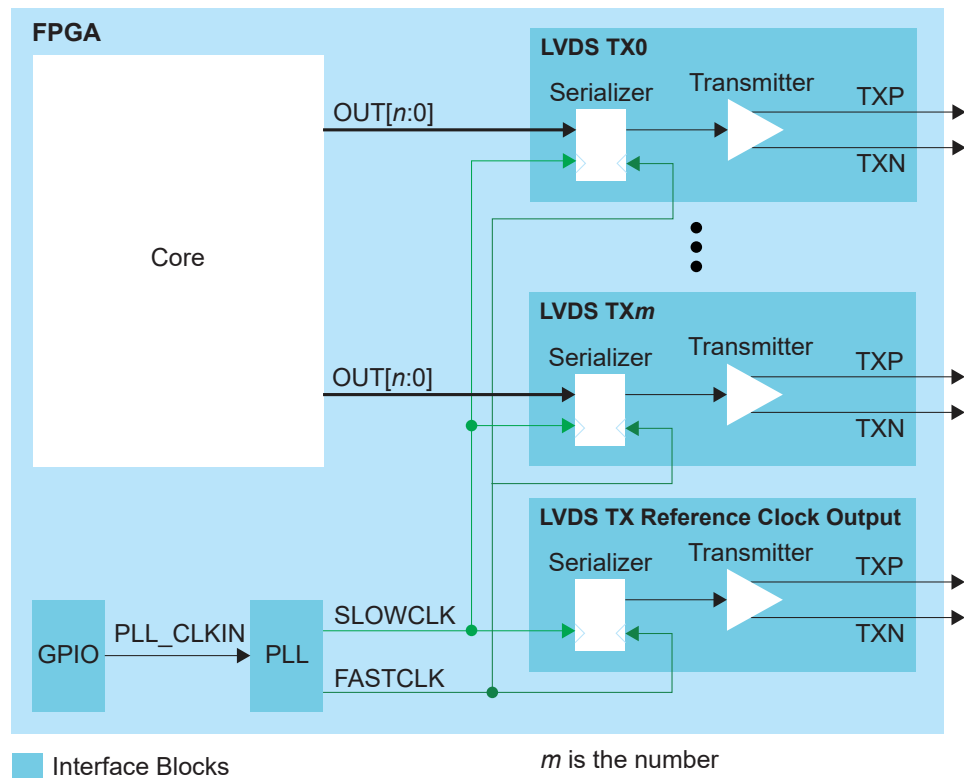
Table 44: PLL Requirements

FPGA	Side	PLL
Ti35, Ti60, Ti60ES	Left	BL_PLL, TL_PLL
	Right	BR_PLL, TR_PLL
	Top	TR_PLL, TL_PLL
	Bottom	BR_PLL, BL_PLL

Create an LVDS TX Interface

The following figure shows a completed LVDS TX interface, where n is the serialization width and m is the number of TX lanes.

Figure 32: Complete LVDS TX Interface Block Diagram



Follow these steps to build an LVDS TX interface using the Efinity® Interface Designer.

1. Add a PLL block with the following settings:

Option	Description
Resource	You can use any PLL resource.
Reference Clock Mode	External
Reference Clock Frequency	Any
Output Clock	For LVDS serializer widths 2 - 8, define the output clocks so that you have one for the fast clock (serial) and one for the slow clock (parallel). Set the relationship between the clocks such that the serial clock frequency = parallel clock frequency * (serialization / 2). The serial clock must use the 90 degree phase shift.

2. Add a GPIO block with these settings to provide the reference clock input to the PLL:

Option	Description
Mode	Input
Pin Name	Any
Connection Type	pll_clkin
GPIO Resource	Assign the dedicated PLL_CLKIN pin that corresponds to the PLL you chose.

3. Add an LVDS TX block with these settings:

Option	Description
LVDS Type	Transmitter (TX)
LVDS Resource	Any channel
Mode	Serial data output
Enable Serialization	On
Serialization Width	<i>n</i>
Output Pin/ Bus Name	Any
Serial Clock Pin Name	Use the fast clock output name that corresponds to the PLL you chose.
Parallel Clock Pin Name	Use the slow clock output name that corresponds to the PLL you chose.

4. Repeat step 3 for each LVDS TX data lane you want to implement.
5. Add another LVDS block that will serve as the LVDS TX reference clock output:

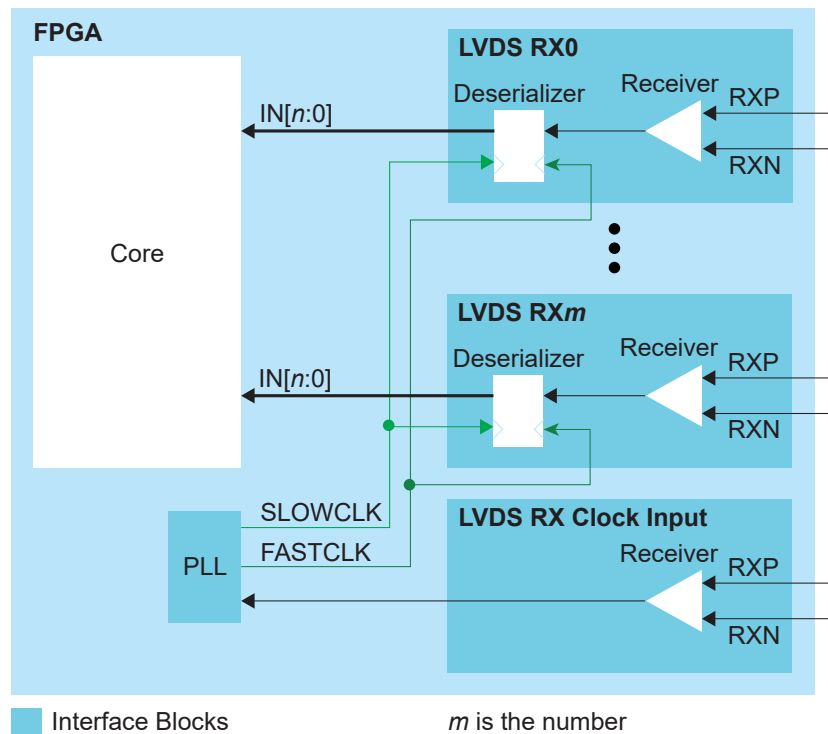
Option	Description
LVDS type	Transmitter (TX)
LVDS resource	Any channel
Mode	Reference clock output
Enable Serialization	On
Serialization width	<i>n</i>
Output pin/ bus name	Any

Option	Description
Parallel clock division	1: The output clock from the LVDS TX lane is parallel clock frequency. 2: The output clock from the TX lane is half of the parallel clock frequency.
Serial clock pin name	Specify the fast clock output name that corresponds to the PLL you chose.
Parallel clock pin name	Use the slow clock output name that corresponds to the PLL you chose.

Create an LVDS RX Interface

The following figure shows a completed LVDS RX interface, where n is the deserialization width and m is the number of RX lanes.

Figure 33: Complete LVDS RX Interface Block Diagram



Follow these steps to build an LVDS RX interface using the Efinity® Interface Designer.

1. Add an LVDS RX block to act as the PLL reference clock input:

Option	Description
LVDS Type	Receiver (RX)
LVDS Resource	Use any LVDS resource.
Connection Type	pll_clkin
Input Pin/Bus Name	Use the clock LVDS RX clock output name as the incoming clock.

2. Add a PLL block with the following settings:

Option	Description
Resource	Use any PLL resource.
Reference Clock Mode	External
Reference Clock Frequency	Set the reference clock frequency to match the clock coming from the LVDS RX reference clock you created in step 1.
Output Clock	For LVDS deserializer widths 2 - 8, define the output clocks so that you have one for the fast clock (serial) and one for the slow clock (parallel). Set the relationship between the clocks such that the serial clock frequency = parallel clock frequency * (serialization / 2). The serial clock must use the 90 degree phase shift.

3. Add an LVDS RX block with these settings:

Option	Description
LVDS Type	Receiver (RX)
LVDS Resource	Any channel
Enable Deserialization	On
Deserialization Width	<i>n</i>
Output Pin/ Bus Name	Any
Serial Clock Pin Name	Use the fast clock output name that corresponds to the PLL you chose.
Parallel Clock Pin Name	Use the slow clock output name that corresponds to the PLL you chose.

4. Repeat step 3 for each LVDS RX data lane you want to implement.

Design Check: LVDS Errors and Warnings

When you check your design, the Interface Designer applies design rules to your LVDS settings. The following tables show some of the error and warning messages you may encounter and explains how to fix them.

lvds_rule_bidir_tx (error)

Message	Output enable pin name must be configured in Bidirectional LVDS Tx
To fix	If you are using a bidirectional LVDS block, you need to specify the output enable pin name.

lvds_rule_clkout_mode (error)

Message	Serial clock name must be configured in clock output mode
To fix	When you are using the LVDS serializer (serialization width greater than 1), you need to specify the serial clock pin name.
Message	Parallel clock name must be configured in clock output mode
To fix	When you are using the LVDS serializer (serialization width greater than 1), you need to specify the parallel clock pin name.
Message	Clock output mode is not supported with serialization width 1
To fix	When you set the serialization width to 1, you are bypassing the serializer and using the block as a simple buffer. As a simple buffer, you cannot use the block as a reference clock output. Change the serialization width.
Message	Clock output mode is not supported with serialization disabled
To fix	If you turn off Enable Serialization , you are bypassing the serializer and using the block as a simple buffer. As a simple buffer, you cannot use the block as a reference clock output. Turn on Enable Serialization and set the serialization width.

lvds_rule_deserial_rate (error)

Message	Half rate deserialization only allowed with even deserialization width
To fix	When you turn on Enable Half Rate Serialization , you can only use an even number for the deserialization width. Change the width to an even number or turn the option off.

lvds_rule_output_mode (error)

Message	Instance of LVDS Tx has invalid configuration
To fix	The block settings are not correct. It might be best to remove the block and start over :)
Message	Output name must be configured in data output mode
To fix	Specify a valid pin name.
Message	Parallel clock name must be configured in data output mode
To fix	When you are using the LVDS serializer (serialization width greater than 1), you need to specify the parallel clock pin name.
Message	Serial clock name must be configured in data output mode
To fix	When you are using the LVDS serializer (serialization width greater than 1), you need to specify the serial clock pin name.

lvds_rule_resource (error)

Message	Resource name is empty Resource <string> is not a valid LVDS device instance
To fix	You need to choose a valid resource.

lvds_rule_usage (error)

Message	Resource <res name> was assigned multiple times
To fix	You cannot assign the same resource to more than one block type. Change the resource to a different one.

lvds_rule_rx_alt_conn (error)

Message	Connection type <type> is not supported by the resource
To fix	If you want to use the alternate function of an LVDS block, you need to choose a resource that supports it. You can filter for resources by alternate function in the Resource Assigner.

Message	The resource only supports normal connection type
To fix	You need to choose the normal connection type or assign a different resource that supports the connection type you want to use. You can filter for resources by alternate function in the Resource Assigner.

lvds_rule_alt_conn (warning)

Message	Connection type <type> must be used by valid PLL
To fix	The LVDS block is connected to a PLL clock input but the resource you assigned does not support the pll_clkln alternate function. Choose a different resource that supports it. You can filter resources by alternate function in the Resource Assigner.

Message	Connection type <type> cannot be used on an unbonded resource
To fix	You get this error if the resource you choose is not available in the FPGA/package combination you are using. Choose another resource.

Message	pll_clkln connection to PLL clock source not being used in <instance>
To fix	The LVDS block is set to be a PLL reference clock (pll_clkln connection type) but the PLL is not configured to use it. Make sure that the clock you are choosing in the PLL is associated with this GPIO's resource.

Message	pll_clkln connection to PLL clock source but none of the external clock source in PLL <instance> is selected
To fix	The LVDS block is set to be a PLL reference clock (pll_clkln connection type) but the PLL is not configured to use it. In the PLL block, choose external or dynamic as the Clock Source and make sure that the clock you are choosing is associated with this GPIO's resource.

Message	pll_clkln connection to PLL clock source but PLL Clock source on <instance> is set to core
To fix	The LVDS block is set to be a PLL reference clock (pll_clkln connection type) but the PLL is not configured to use it. In the PLL block, choose external or dynamic as the Clock Source and make sure that the clock you are choosing is associated with this GPIO's resource.

Message	pll_extfb connection to PLL external feedback pin but PLL feedback on <inst> is not set to default
To fix	The LVDS block is set to be external feedback for the PLL (pll_extfb connection type) but the PLL is not configured to use it. In the PLL Clock Calculator, choose External as the Feedback Mode.

lvds_rule_rx_clock (error)

Message	Serial and parallel clocks cannot be the same clock
To fix	You cannot use the same clock for both the serial (FASTCLK) and parallel (SLOWCLK) clocks.
Message	Serial clock name is not a PLL output clock
To fix	Use a PLL output clock as the serial (FASTCLK) clock.
Message	Parallel clock name is not a PLL output clock
To fix	Use a PLL output as the parallel (SLOWCLK) clock.
Message	Serial and parallel clocks are not from the same PLL instance
To fix	You need to use the same PLL to generate both clocks.
Message	Invalid phase shift difference: <phase diff> = Serial: <value> - Parallel: <value> (Min=45 degrees, Max=135 degrees)
To fix	The phase shift for the parallel and serial clocks must be between 45° and 135°.
Message	One of the clock frequencies is 0
To fix	The output clock frequency is invalid. Check that the PLL is configured correctly.
Message	Serial clock frequency has to be <float> times faster than parallel clock
To fix	Make sure that the PLL output clock frequencies are set correctly. <i>Half rate calculation</i> —serial clock frequency = parallel clock frequency * (serialization / 2) <i>Full rate calculation</i> —serial clock frequency = parallel clock * serialization

lvds_rule_rx_clock_region (error)

Message	Serial and Parallel clocks generated by PLL have to be driven to the same clock network. <Serial Parallel> clock <name> was generated by PLL output clock 4 that connects to regional clock network
To fix	In Ti35 and Ti60FPGAs, the PLL's output clock 4 can only drive the regional clock network. You should use the other clock outputs for the serial and parallel clocks.

lvds_rule_rx_config (error)

Message	Input name must be configured
To fix	Specify a valid pin name.
Message	Serial clock name must be configured
To fix	When you are using the LVDS deserializer (deserialization width greater than 1), you need to specify the serial clock pin name.
Message	Parallel clock name must be configured
To fix	When you are using the LVDS deserializer (deserialization width greater than 1), you need to specify the parallel clock pin name.

lvds_rule_rx_distance (error)

Message	These HSIO GPIO must be placed at least 1 pair away from LVDS <name> in order to avoid noise coupling from GPIO to LVDS: <violated list>
To fix	When using HSIO pins as GPIO, make sure to leave at least 1 pair of unassigned HSIO pins between any GPIO and HSIO used as LVDS RX in the same bank. This separation reduces noise.

lvds_rule_rx_dpa (error)

Message	Half-rate deserialization is not supported with DPA delay mode
To fix	You can only use full-rate serialization with DPA mode. Turn off the Enable Half Rate Deserialization option.

lvds_rule_rx_dpa_es_device (error)

Message	DPA delay mode is not supported in ES device
To fix	The ES FPGA does not support DPA.

lvds_rule_rx_dpa_serial (error)

Message	DPA delay mode is not supported with deserialization disabled
To fix	You cannot use dynamic phase alignment in bypass mode.

Message	DPA delay mode is not supported with deserialization width less than 3
To fix	You cannot use dynamic phase alignment with x1 or x2 modes.

lvds_rule_rx_empty_pins (error)

Message	Empty pin names found: <list>
To fix	You need to specify the pin names listed in the message.

lvds_rule_rx_fifo (error)

Message	Clock Crossing FIFO is not supported with deserialization width <1/2>
To fix	The Clock Crossing FIFO is only available for deserialization widths > 2. Disable the Clock Crossing FIFO or change the serialization value.

Message	Clock Crossing FIFO is only supported with deserialization enabled
To fix	The Clock Crossing FIFO is only available for deserialization widths > 2. Disable the Clock Crossing FIFO or change the serialization value.

lvds_rule_rx_param (error)

Message	Invalid parameters configuration: <list>
To fix	One of the parameters you set was incorrect. Review any other errors for details.

lvds_rule_rx_pll_refclk (error)

Message	Serial clock name is not a PLL output clock
To fix	Use a PLL output clock as the serial (FASTCLK) clock.
Message	Parallel clock name is not a PLL output clock
To fix	Use a PLL output as the parallel (SLOWCLK) clock.

lvds_rule_rx_pll_refclk (warning)

Message	Serial clock is expected to be from the following PLL instance: <resource>
To fix	Only a specific PLL instance can drive the LVDS RX clocks. Change the PLL to use that resource. See PLL Requirements for Serial and Parallel Clocks on page 79.
Message	PLL driving the serial clock should have its reference clock from an LVDS in pll_clkln connection type
To fix	The PLL's reference clock needs to be driven by a specific resource. Create an LVDS RX block and set the Connection Type to pll_clkln . Then use that block as the PLL reference clock.
Message	Parallel clock is expected to be from the following PLL instance: {}
To fix	Only a specific PLL instance can drive the LVDS RX clocks. Change the PLL to use that resource. See PLL Requirements for Serial and Parallel Clocks on page 79.
Message	PLL driving the parallel clock should have its reference clock from an LVDS in pll_clkln connection type
To fix	The PLL's reference clock needs to be driven by a specific resource. Create an LVDS RX block and set the Connection Type to pll_clkln . Then use that block as the PLL reference clock.

lvds_rule_rx_serial_width (error)

Message	Unsupported deserializaion width: 9
To fix	The LVDS block does not support a deserialization wiudth of 9. Choose another width.

lvds_rule_tx_width_1or2 (error)

Message	Parallel clock name is required with serialization width 2 Serialization width <1/2> only requires the parallel clock name to be specified
To fix	When you are using the LVDS serializer (serialization width greater than 1), you need to specify the parallel clock pin name.

lvds_rule_rx_width_1or2 (error)

Message	Parallel clock name is required with deserialization width 2 Deserialization width <1/2> only requires the parallel clock name to be specified
To fix	When you are using the LVDS deserializer (serialization width greater than 1), you need to specify the parallel clock pin name.

lvds_rule_serial_rate (error)

Message	Half rate serialization only allowed with even serialization width
To fix	When you turn on Enable Half Rate Serialization , you can only use an even number for the serialization width. Change the width to an even number or turn the option off.

lvds_rule_tx_clock (error)

Message	Serial and parallel clocks cannot be the same clock
To fix	You cannot use the same clock for both the serial (FASTCLK) and parallel (SLOWCLK) clocks.
Message	Serial clock name is not a PLL output clock
To fix	Use a PLL output clock as the serial (FASTCLK) clock.
Message	Parallel clock name is not a PLL output clock
To fix	Use a PLL output as the parallel (SLOWCLK) clock.
Message	Serial and parallel clocks are not from the same PLL instance
To fix	You need to use the same PLL to generate both clocks.
Message	Invalid phase shift difference: <phase diff> = Serial: <value> - Parallel: <value> (Min=45 degrees, Max=135 degrees)
To fix	The phase shift for the parallel and serial clocks must be between 45° and 135°.
Message	One of the clock frequencies is 0
To fix	The output clock frequency is invalid. Check that the PLL is configured correctly.
Message	Serial clock frequency has to be <float> times faster than parallel clock
To fix	Make sure that the PLL output clock frequencies are set correctly. <i>Half rate calculation</i> —serial clock frequency = parallel clock frequency * (serialization / 2) <i>Full rate calculation</i> —serial clock frequency = parallel clock * serialization

lvds_rule_tx_clock_region (error)

Message	Serial and Parallel clocks generated by PLL have to be driven to the same clock network. <Serial Parallel> clock <name> was generated by PLL output clock 4 that connects to regional clock network
To fix	In Ti35 and Ti60FPGAs, the PLL's output clock 4 can only drive the regional clock network. You should use the other clock outputs for the serial and parallel clocks.

lvds_rule_tx_distance (error)

Message	These HSIO GPIO must be placed at least 1 pair away from LVDS <name> in order to avoid noise coupling from GPIO to LVDS: <violated list>
To fix	When using HSIO pins as GPIO, make sure to leave at least 1 pair of unassigned HSIO pins between any GPIO and HSIO used as LVDS TX in the same bank. This separation reduces noise.

lvds_rule_tx_empty_pins (error)

Message	Empty pin names found: <list>
To fix	You need to specify the pin names listed in the message.

lvds_rule_tx_param (error)

Message	Invalid parameters configuration: <list>
To fix	One of the parameters you set was incorrect. Review any other errors for details.

lvds_rule_tx_serial_width (error)

Message	Unsupported serializaion width: 9
To fix	The LVDS block does not support a serialization wiudth of 9. Choose another width.

lvds_rule_tx_vref (error)

Message	This resource is reserved as vref for bank <name>. Us a different resource to configure LVDS Tx custom output differential type
To fix	Some resources can be used as the VREF for an I/O standard. If you are using an I/O standard that uses a VREF pin, you must use this resource as a VREF. Choose another resource for the LVDS function.
Message	GPIO <name> has to be configured as vref input mode to support LVDS Tx custom output differential type GPIO <name> has to be configured as vref input to support LVDS Tx custom output differential type
To fix	If you are using an I/O standard that uses a VREF pin, you must use this resource as a VREF. Configure the GPIO as an input and choose vref as the Connection Type .
Message	LVDS Tx custom output differential type cannot be used due to unbonded vref resource on the same bank <bank> LVDS Tx custom output differential type cannot be used due to vref resource not bonded out
To fix	If a VREF pin is not available in the I/O bank (e.g., it is not in the FPGA/package you chose), you cannot use an I/O standard that requires it. Instead choose a different I/O standard or a different resource.

HyperRAM Interface

Contents:

- [About the HyperRAM](#)
- [Using the HyperRAM Interface](#)

The Ti35 and Ti60 FPGAs in the F100S3F2 package include an integrated HyperRAM memory. You use the Interface Designer to connect this block to your user design. Only the Ti35 or Ti60 can communicate with the on-board HyperRAM.

About the HyperRAM

The Titanium FPGA in F100S3F2 package includes a HyperRAM. The HyperRAM has a density of 256 Mbits and a clock rate of up to 200 MHz. The HyperRAM supports double-data rates of up to 400 Mbps and supports a 16 bit data bus.

Figure 34: HyperRAM Block Diagram

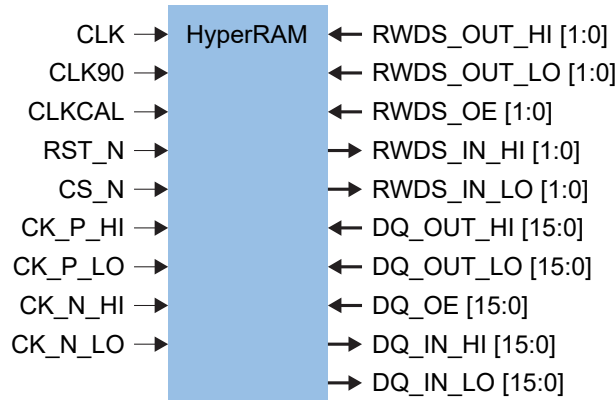


Table 45: HyperRAM Signals (Interface to FPGA Fabric)

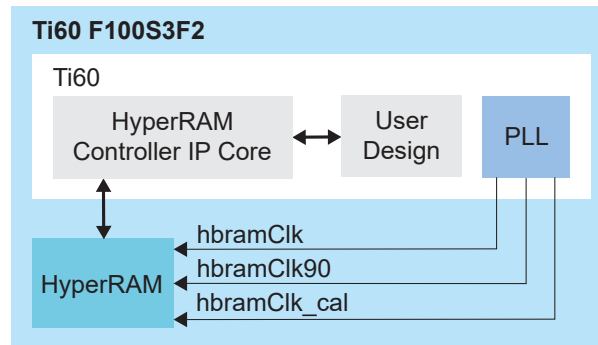
Signal	Direction	Description
CLK	Input	HyperRAM controller clock.
CLK90	Input	90 degree phase-shifted version of CLK.
CLKCAL	Input	Calibration clock for input data.
RST_N	Input	Active-low HyperRAM reset.
CS_N	Input	Active-low HyperRAM chip select signal.
CK_P_HI	Input	The clock provided to the HyperRAM. The clock is not required to be free-running. Registered in normal mode of DDIO.
CK_P_LO	Input	
CK_N_HI	Input	
CK_N_LO	Input	

Signal	Direction	Description
RWDS_OUT_HI [1:0]	Input	Read/write data strobe input ports for data mask during write operation. Registered in normal mode/resync mode of DDIO.
RWDS_OUT_LO [1:0]	Input	
RWDS_OE [1:0]	Input	Read/write data strobe output enable port.
RWDS_IN_HI [1:0]	Output	Read/write data strobe output ports for latency indication, also center-aligned reference strobe for read data. Registered in normal mode/resync mode of DDIO.
RWDS_IN_LO [1:0]	Output	
DQ_OUT_HI [15:0]	Input	DQ input ports for command, address and data. Registered in normal mode of DDIO.
DQ_OUT_LO [15:0]	Input	
DQ_OE [15:0]	Input	DQ output enable port.
DQ_IN_HI [15:0]	Output	DQ output ports for data.
DQ_IN_LO [15:0]	Output	

Using the HyperRAM Interface

To use the HyperRAM block, you add the block, choose the resource and specify the instance and pin names. Then, add a HyperRAM Controller IP core instance to your project to connect it to the HyperRAM. A PLL generates the control signals.

Figure 35: Example System with HyperRAM Block



Note: You can generate an example design with the IP Manager. Open the wizard for the HyperRAM Controller IP core and select **Deliverables > Example Design (Ti60F100_pll_cal)** and generate. Refer to the HyperRAM Controller IP Core User Guide for a description of this example.

Table 46: HyperRAM Properties

Option	Values	Notes
Instance Name	User defined	
HyperRAM Resource	HYPER_RAM0	
HyperRAM Controller Clock Pin Name	User defined	Must come from PLL output
Calibration Clock Pin Name	User defined	Must come from PLL output
90 Degree Phase-Shifted Clock Pin Name	User defined	Must come from PLL output

Option	Values	Notes
Active-Low HyperRAM Reset Pin Name	User defined	
Active-Low HyperRAM Chip Select Pin Name	User defined	
Differential Clock Pin Name (P HI)	User defined	
Differential Clock Pin Name (P LO)	User defined	
Differential Clock Pin Name (N HI)	User defined	
Differential Clock Pin Name (N LO)	User defined	
Read/Write Data Strobe Output [1:0] Bus Name (HI)	User defined	
Read/Write Data Strobe Output [1:0] Bus Name (LO)	User defined	
Read/Write Data Strobe Output Enable [1:0] Bus Name	User defined	
Read/Write Data Strobe Input [1:0] Bus Name (HI)	User defined	
Read/Write Data Strobe Input [1:0] Bus Name (LO)	User defined	
DQ Output [15:0] Bus Name (HI)	User defined	
DQ Output [15:0] Bus Name (LO)	User defined	
DQ Output Enable [15:0] Bus Name	User defined	
DQ Input [15:0] Bus Name (HI)	User defined	
DQ Input [15:0] Bus Name (LO)	User defined	

JTAG User TAP Interface

Contents:

- [JTAG Mode](#)
- [Using the JTAG User TAP Block](#)
- [Design Check: JTAG User Tap Errors and Warnings](#)

Titanium FPGAs have dedicated JTAG pins to support configuration and boundary scan testing.

JTAG Mode

The JTAG serial configuration mode is popular for prototyping and board testing. The four-pin JTAG boundary-scan interface is commonly available on board testers and debugging hardware.

Efnix FPGAs support IEEE standard 1149.1 - 2001.



Learn more: Refer to the following web sites for more information about the JTAG interface:

<http://ieeexplore.ieee.org/document/6515989/>

<https://en.wikipedia.org/wiki/JTAG>

Table 47: Supported JTAG Instructions

Instruction	Binary Code [4:0]	Description
BYPASS	11111	Enables BYPASS.
DEVICE_STATUS	01100	Lets you read the device configuration status.
EFUSE_PREWRITE	11000	Loads user data for fuse operations.
EFUSE_USER_WRITE	11010	Blows fuses as defined in EFUSE_PREWRITE.
EFUSE_WRITE_STATUS	11011	Returns status of EFUSE_USER_WRITE operation.
ENTERUSER	00111	Changes the FPGA into user mode.
EXTEST	00000	Enables the boundary-scan EXTEST operation.
IDCODE	00011	Enables shifting out the IDCODE.
INTEST	00001	Enables the boundary-scan INTEST operation.
JTAG_USER1	01000	Connects the JTAG User TAP 1.
JTAG_USER2	01001	Connects the JTAG User TAP 2.
JTAG_USER3	01010	Connects the JTAG User TAP 3.
JTAG_USER4	01011	Connects the JTAG User TAP 4.
PROGRAM	00100	JTAG configuration.
SAMPLE/PRELOAD	00010	Enables the boundary-scan SAMPLE/PRELOAD operation.

Instruction	Binary Code [4:0]	Description
USERCODE	01101	Use this instruction to program a 32-bit signature into the FPGA during programming.



Learn more: Refer to the [AN 038: Programming with an MCU and the JTAG Interface](#) for more information about programming Efinix® FPGAs with a microcontroller using JTAG mode.

Connect the FPGA pins as shown in the following diagrams.

Figure 36: JTAG Programming (Ti35 and Ti60 FPGAs)

See [Resistors in Configuration Circuitry](#) for the resistor values.

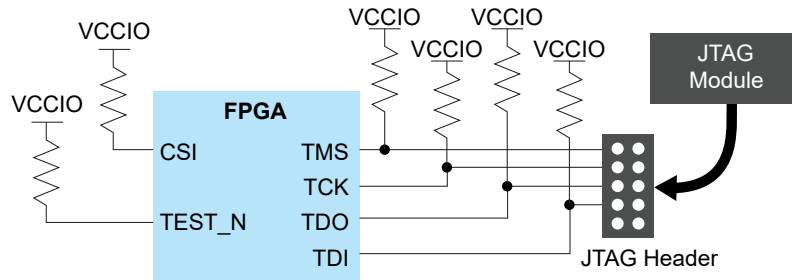
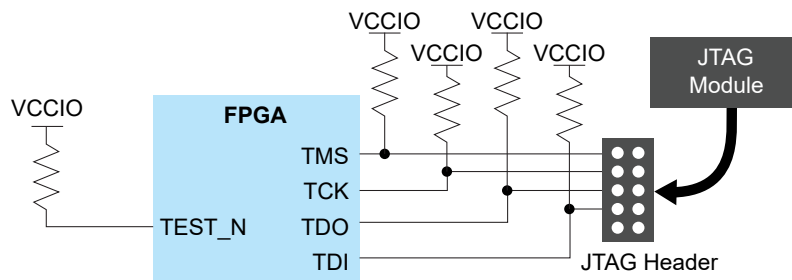


Figure 37: JTAG Programming (Ti90, Ti120, and Ti180 FPGAs)

See [Resistors in Configuration Circuitry](#) for the resistor values.

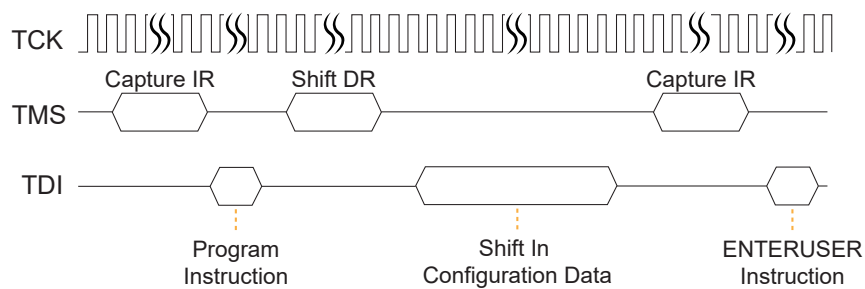


The $\overline{\text{CRESET_N}}$ signal needs to be deasserted before JTAG configuration begins. When configuration ends, the JTAG host issues the ENTERUSER instruction to the FPGA. After CDONE goes high and the FPGA receives the ENTERUSER instruction, the FPGA waits for t_{USER} to elapse, and then it goes into user mode.



Note: The FPGA may go into user mode before t_{USER} has elapsed. Therefore, you should keep the system interface with the FPGA in reset until t_{USER} has elapsed.

Figure 38: JTAG Programming Waveform

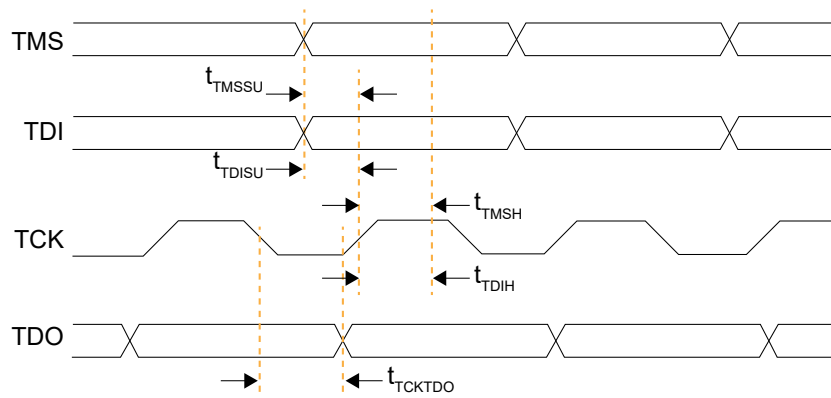


Design Considerations

- Because the TCK and TMS signals connect devices in the JTAG chain, they must have good signal quality.
- TCK should transition monotonically at the receiving devices and should be terminated correctly. Poor TCK quality can limit the maximum frequency you can use for configuration.
- Buffer TMS and TCK so they have sufficient drive strength at all receiving devices.
- Ensure that the logic high voltage is compatible with all devices in the JTAG chain.
- If your chain contains devices from different vendors, you might need to drive optional JTAG signals, such as TRST and enables.

Timing Parameters

Figure 39: Boundary-Scan Timing Waveform



Learn more: Refer to the FPGA data sheet for timing specifications.

Refer to the Virtual I/O Debug Core section in the [Efinix Software User Guide](#) for more information about JTAG User TAP interface.

Using the JTAG User TAP Block

Add the JTAG User TAP block to your interface if you want to use the FPGA JTAG pins to communicate with the design running in the core.

You specify the instruction to use with the **JTAG Resource** setting. Titanium FPGAs have four JTAG User TAP blocks. To use more than one, add JTAG User TAP blocks to your interface design, one for each resource.

Table 48: JTAG User TAP Signals

Signal	Direction	Description
<instance>_TDI	Input	JTAG test data in pin.
<instance>_TCK	Input	JTAG test clock pin.
<instance>_TMS	Input	JTAG mode select pin.
<instance>_SEL	Input	User instructive active pin.
<instance>_DRCK	Input	Gated test clock.
<instance>_RESET	Input	Reset.
<instance>_RUNTEST	Input	Run test pin.

Signal	Direction	Description
<instance>_CAPTURE	Input	Capture pin.
<instance>_SHIFT	Input	Shift pin.
<instance>_UPDATE	Input	Update pin.
<instance>_TDO	Output	JTAG test data out pin.

Design Check: JTAG User Tap Errors and Warnings

When you check your design, the Interface Designer applies design rules to your JTAG User Tap settings. The following tables show some of the error and warning messages you may encounter and explains how to fix them.

jtag_rule_clock (warning)

Message	Clock pin name is empty
To fix	Enter a valid clock pin name.

jtag_rule_resource (error)

Message	Resource name is empty Resource is not a valid JTAG device instance
To fix	Specify a valid JTAG resource.

MIPI RX/TX Lane Interface

Contents:

- [HSIO Configured as MIPI Lane](#)
- [MIPI Groups by Package](#)
- [Using the MIPI TX Lane or MIPI RX Lane Block](#)
- [Create a MIPI TX Interface](#)
- [Create a MIPI RX Interface](#)
- [Design Check: MIPI Lane Messages](#)

Each HSIO block can use a pair of I/O pins as a MIPI RX or TX data lane or clock lane.

HSIO Configured as MIPI Lane

You can configure the HSIO block as a MIPI RX or TX lane. The block supports bidirectional data lane, unidirectional data lane, and unidirectional clock lane which can run at speeds up to 1.5 Gbps. The MIPI lane operates in high-speed (HS) and low-power (LP) modes. In HS mode, the HSIO block transmits or receives data with x8 serializer/deserializer. In LP mode, it transmits or receives data without deserializer/serializer.

The MIPI lane block does not include the MIPI D-PHY core logic. A full MIPI D-PHY solution requires:

- Multiple MIPI RX or TX lanes (at least a clock lane and a data lane)
- Soft MIPI D-PHY IP core programmed into the FPGA fabric

The MIPI D-PHY standard is a point-to-point protocol with one endpoint (TX) responsible for initiating and controlling communication. Often, the standard is unidirectional, but when implementing the MIPI DSI protocol, you can use one TX data lane for LP bidirectional communication.

The protocol is source synchronous with one clock lane and 1, 2, 4, or 8 data lanes. The number of lanes available depends on which package you are using. A dedicated HSIO block is assigned on the RX interface as a clock lane while the clock lane for TX interface can use any of the HSIO block in the group.

MIPI RX Lane

In RX mode, the HS (fast) clock comes in on the MIPI clock lane and is divided down to generate the slow clock. The fast and slow clocks are then passed to neighboring HSIO blocks to be used for the MIPI data lanes.

The data lane fast and slow clocks must be driven by a clock lane in the same MIPI group (dedicated buses drive from the clock lane to the neighboring data lanes).

The MIPI RX function is defined as:

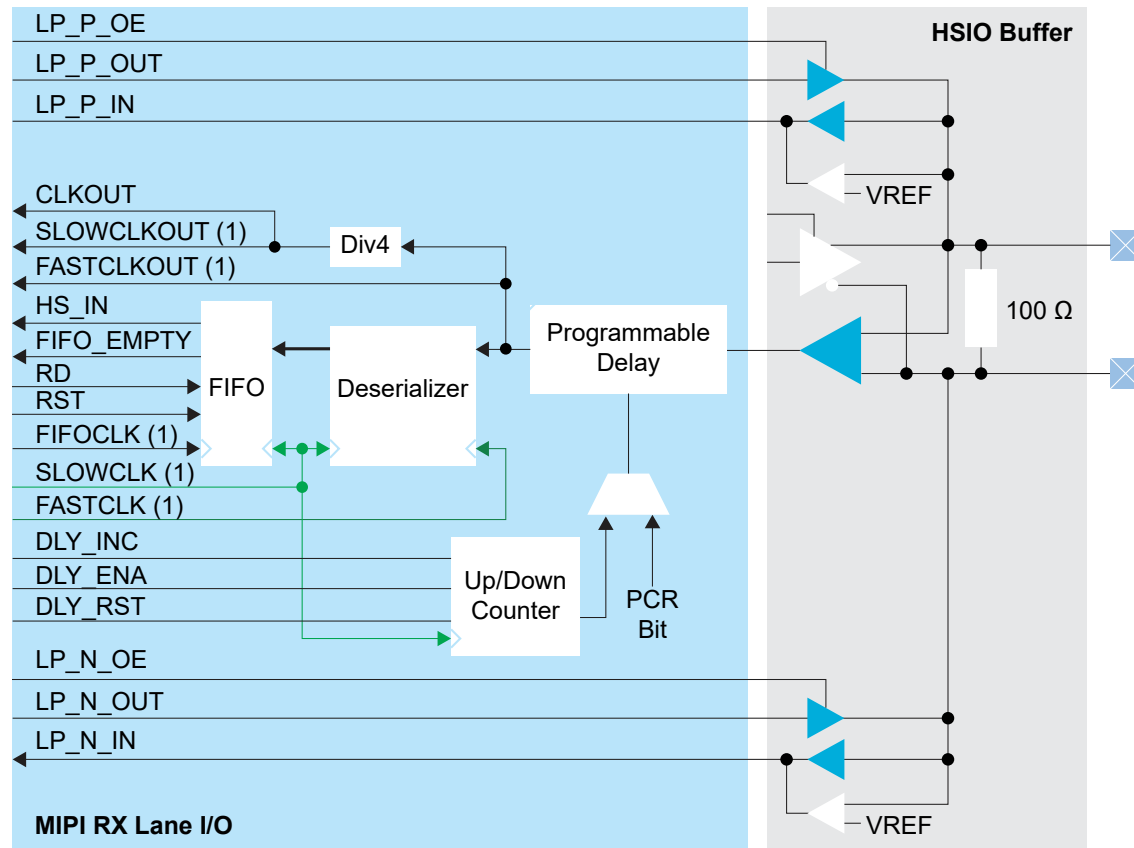
Table 49: MIPI RX Function

MIPI RX Function	Description
RX_DATA_xy_zz	MIPI RX Lane. You can use any data lanes within the same group to form multiple lanes of MIPI RX group. $x = P \text{ or } N$ $y = 0 \text{ to } 7$ data lanes (Up to 8 data lanes) Ti35 and Ti60:zz = I0 to I11 MIPI RX group (Up to 12 MIPI RX groups) Ti90, Ti120, and Ti180:zz = I0 to I17 MIPI RX group (Up to 18 MIPI RX groups)
RX_CLK_x_zz	MIPI RX Clock Lane. $x = P \text{ or } N$ Ti35 and Ti60:zz = I0 to I11 MIPI RX group (Up to 12 MIPI RX groups) Ti90, Ti120, and Ti180:zz = I0 to I17 MIPI RX group (Up to 18 MIPI RX groups)



Learn more: Refer to the pinout file for your FPGA for more information about the MIPI RX function for each HSIO and for which pins are in the same MIPI group.

Figure 40: MIPI RX Lane Block Diagram



1. These signals are in the primitive, but the software automatically connects them for you.

Table 50: MIPI RX Lane Signals

Interface to MIPI soft CSI/DSI controller with D-PHY in FPGA Fabric

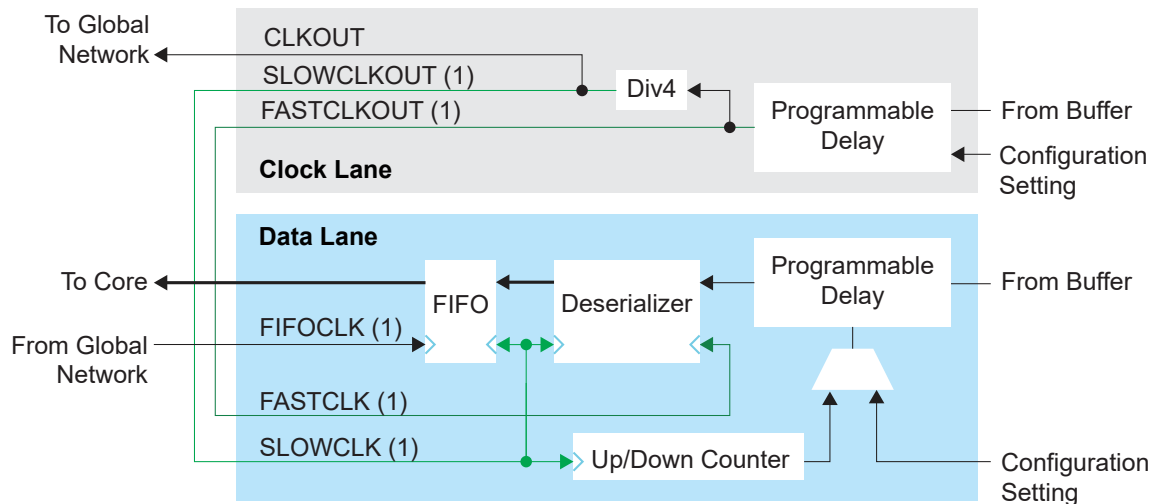
Signal	Direction	Clock Domain	Description
HS_IN[7:0]	Output	SLOWCLK	High-speed parallel data input.
LP_P_IN	Output	-	LP input data from the P pad.
LP_N_IN	Output	-	LP input data from the N pad.
LP_P_OUT	Input	-	(Optional) LP output data from the core for the P pad. Used if the data lane is reversible.
LP_N_OUT	Input	-	(Optional) LP output data from the core for the N pad. Used if the data lane is reversible.
FIFO_EMPTY	Output	FIFOCLK	(Optional) When the FIFO is enabled, this signal indicates that the FIFO is empty.
SLOWCLKOUT ⁽⁷⁾	Output	-	Divided down parallel (slow) clock from the pads. Can only drive RX DATA lanes.
FASTCLKOUT ⁽⁷⁾	Output	-	Serial (fast) clock from the pads. Can only drive RX DATA lanes.

⁽⁷⁾ These signals are in the primitive, but the software automatically connects them for you.

Signal	Direction	Clock Domain	Description
CLKOUT	Output	-	Divided down parallel (slow) clock from the pads that can drive the core clock tree. Used to drive the core logic implementing the rest of the D-PHY protocol. It should also connect to the FIFOCLK of the data lanes.
SLOWCLK ⁽⁷⁾	Input	-	Parallel (slow) clock.
FASTCLK ⁽⁷⁾	Input	-	Serial (fast) clock.
FIFOCLK ⁽⁷⁾	Input	-	(Optional) Core clock to read from the FIFO.
FIFO_RD	Input	FIFOCLK	(Optional) Enables FIFO to read.
RST	Input	FIFOCLK SLOWCLK	(Optional) Asynchronous. Resets the FIFO and serializer. If the FIFO is enabled, it is relative to FIFOCLK; otherwise it is relative to SLOWCLK.
HS_ENA	Input	-	Dynamically enable the differential input buffer when in high-speed mode.
HS_TERM	Input	-	Dynamically enables input termination high-speed mode.
DLY_ENA	Input	SLOWCLK	(Optional) Enable the dynamic delay control.
DLY_INC	Input	SLOWCLK	(Optional) Dynamic delay control. When DLY_ENA is 1, 1: Increments 0: Decrements
DLY_RST	Input	SLOWCLK	(Optional) Reset the delay counter.

The clock lane generates the fast clock and slow clock for the RX data lanes within the interface group. It also generates a clock that feeds the global network. The following figure shows the clock connections between the clock and data lanes.

Figure 41: Connections for Clock and RX Data Lane in the Same MIPI RX Group



1. The software automatically connects this signal for you.

MIPI TX Lane

In TX mode, a PLL generates the parallel and serial clocks and passes them to the clock and data lanes.

Figure 42: MIPI TX Lane Block Diagram

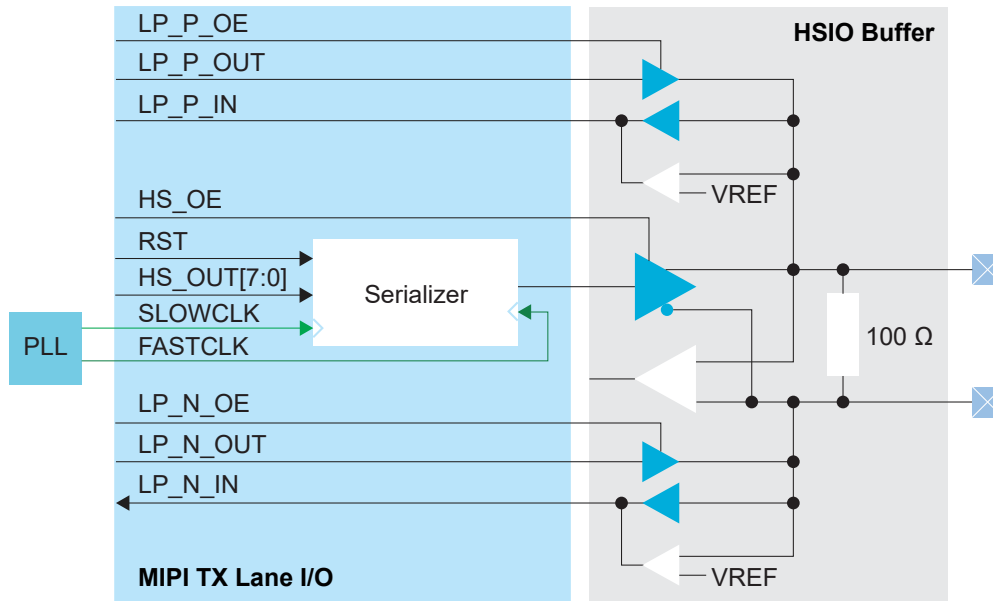


Table 51: MIPI TX Lane Signals

Interface to MIPI soft CSI/DSI controller with D-PHY in FPGA fabric

Signal	Direction	Clock Domain	Description
HS_OUT[7:0]	Input	SLOWCLK	High-speed output data from the core. Always 8-bits wide.
LP_P_OUT	Input	-	LP output data from the core for the P pad.
LP_N_OUT	Input	-	LP output data from the core for the N pad.
LP_P_IN	Output	-	(Optional) LP input data from the P pad. Used if data lane is reversible.
LP_N_IN	Output	-	(Optional) LP input data from the N pad. Used if data lane is reversible.
SLOWCLK	Input	-	Parallel (slow) clock.
FASTCLK	Input	-	Serial (fast) clock.
RST	Input	SLOWCLK	(Optional) Resets the serializer.
HS_OE	Input	-	High-speed output enable signal.
LP_P_OE	Input	-	LP output enable signal for P pad.
LP_N_OE	Input	-	LP output enable signal for N pad.

MIPI Lane Pads

Table 52: MIPI Lane Pads

Signal	Direction	Description
P	Output	Differential pad P.
N	Output	Differential pad N.

MIPI Groups by Package

You can use multiple HSIO as MIPI D-PHY lanes to build complete MIPI interfaces with one clock lane and up to 8 data lanes.

- For MIPI TX interfaces, you can use any lane anywhere on the FPGA.
- For MIPI RX interfaces, the number of data lanes is restricted by the number of lanes in the MIPI group. These groups vary depending on the package.

The following figures show the emulated MIPI RX groups for each package. The Resource Assigner also shows the group in the Block Summary's **Feature** field.

Figure 43: 64-Ball WLCSP MIPI RX Groups

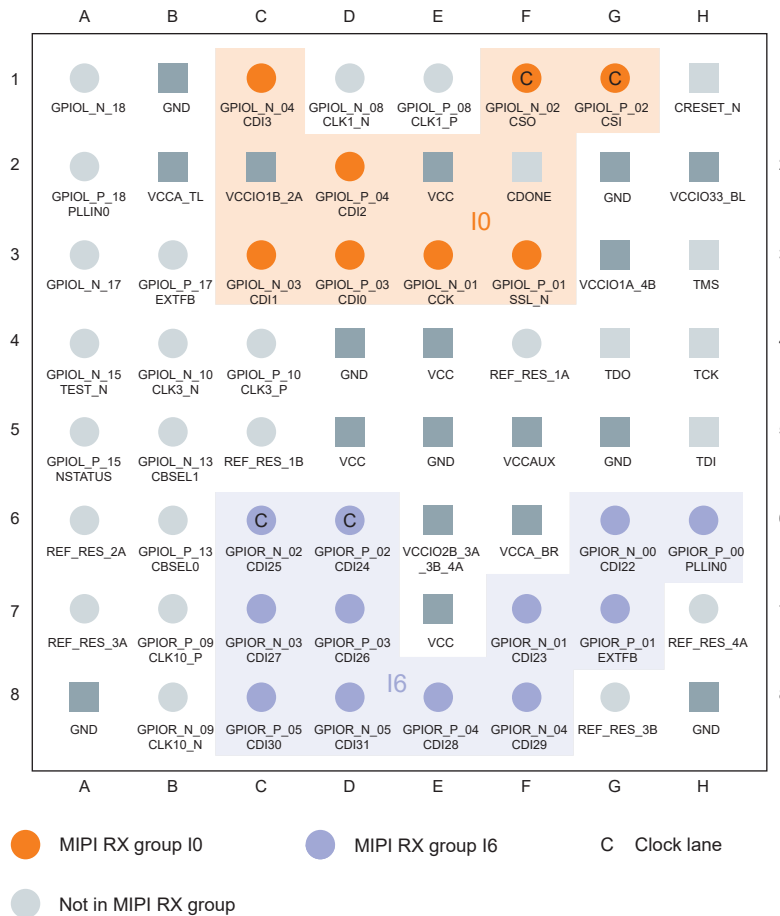


Figure 44: 100-Ball FBGA MIPI RX Groups

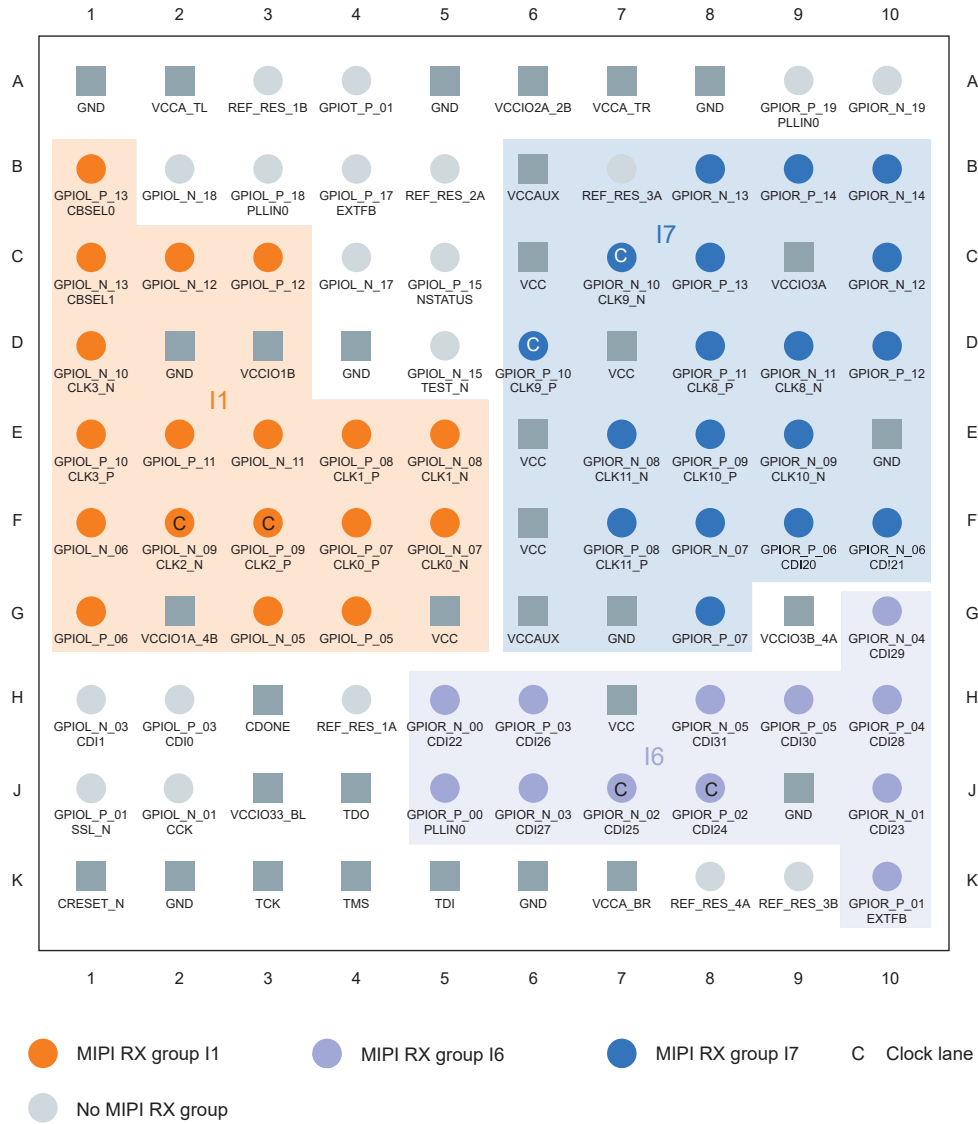
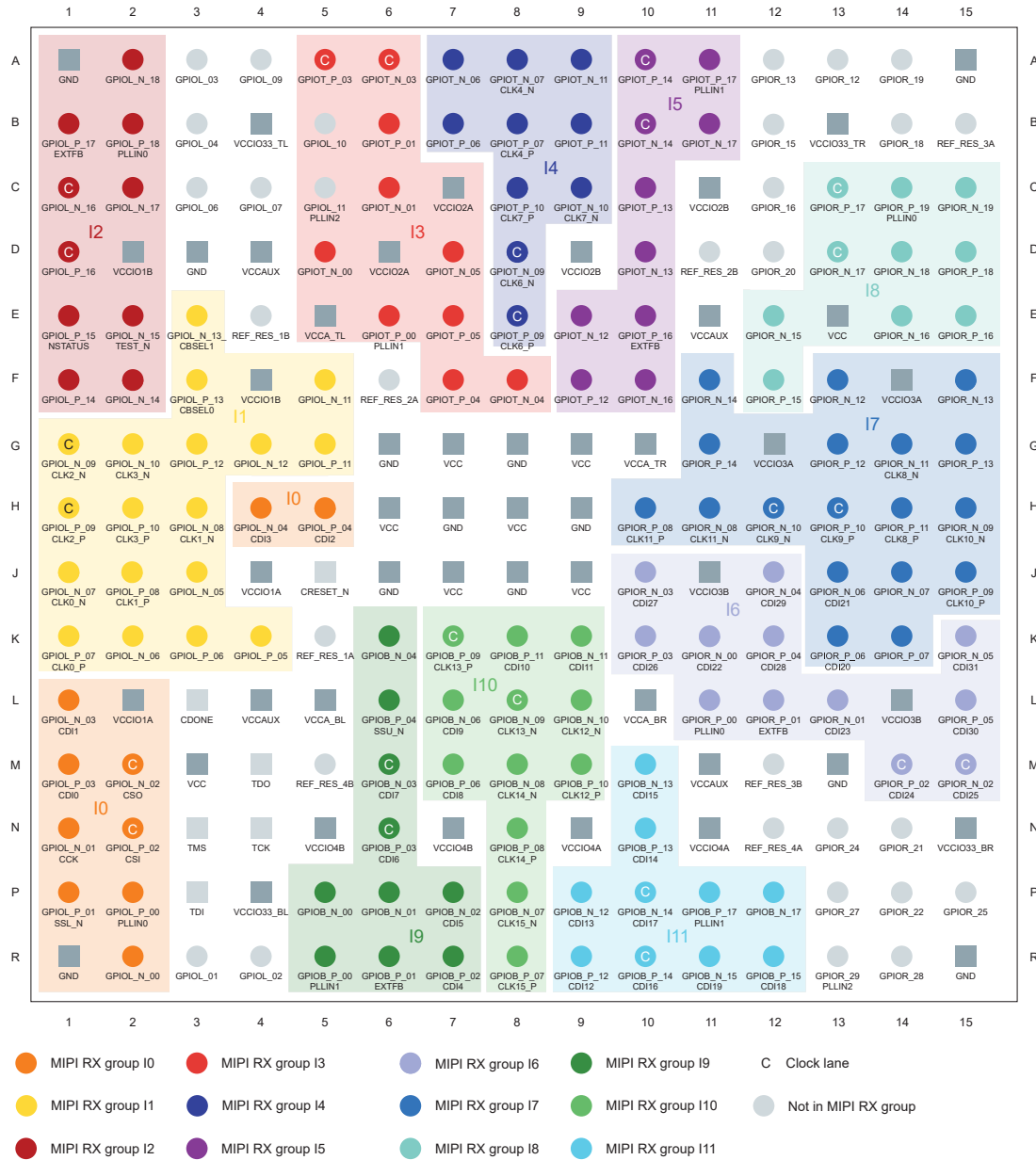


Figure 45: 225-Ball FBGA MIPI RX Groups



Using the MIPI TX Lane or MIPI RX Lane Block

The following tables show how to implement a MIPI TX Lane or MIPI RX Lane block. Later sections explain how to build a complete interface.

MIPI TX Lane Block

Table 53: MIPI TX Lane Block

Option	Choices	Description
Instance Name	User defined	Type the instance name and press enter.
MIPI Lane Resource	Resource list	Choose a resource
Mode	data lane, clock lane	Choose whether the block is a clock lane or data lane.
Enable LP Reverse Communication	On or off	When on, specify the low-power N and P pins.
Pin names (various)	User defined	Specify the interface bus and pin names.
Serial Clock Pin Name	User defined	Name you are using for FASTCLK_D or FASTCLK_C.
Parallel Clock Pin Name	User defined	Name you are using for SLOWCLK.
Static Delay Mode Setting	0 - 63	Choose the amount of static delay, each step adds approximately 25 ps of delay.

MIPI RX Lane Block

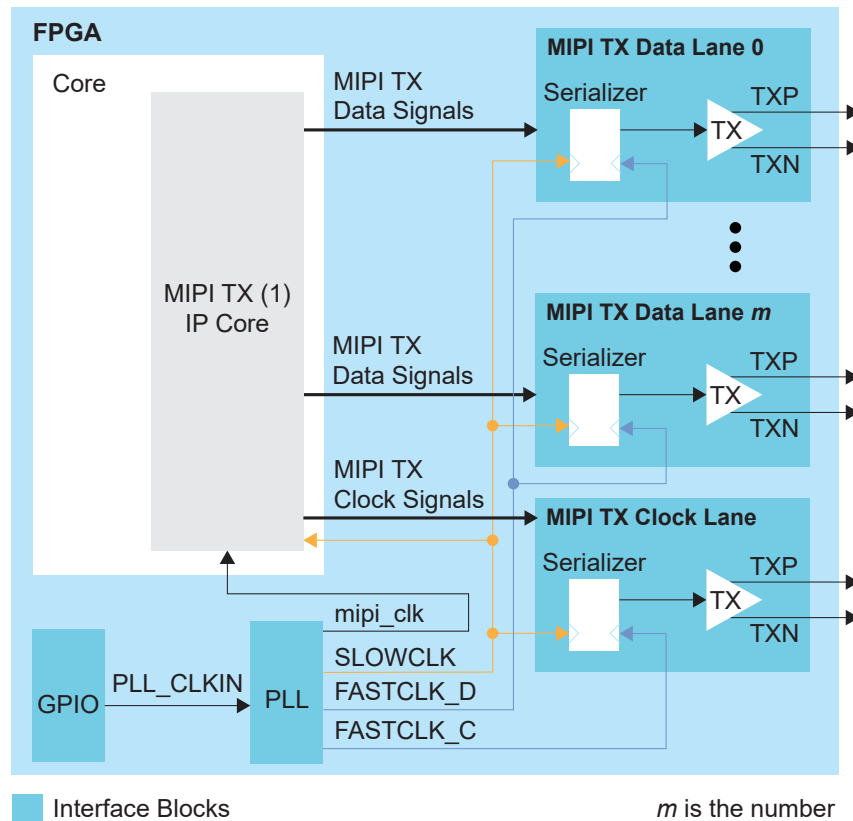
Table 54: MIPI RX Lane Block

Option	Choices	Description
Instance Name	User defined	Type the instance name and press enter.
MIPI Lane Resource	Resource list	Choose a resource.
Mode	data lane, clock lane	Choose whether the block is a clock lane or data lane.
Connection Type	gclk, rclk	In clock lane mode, choose global clock (gclk) or regional clock (rclk).
Enable LP Reverse Communication	On or off	When on, specify the low-power N and P pins.
Enable Clock Crossing FIFO	On or off	When on, specify the FIFO read and empty pins.
Pin names (various)	User defined	Specify the interface bus and pin names.
Delay Mode	static	Integer from 0 - 63. Each step adds approximately 25 ps of delay.
	dynamic	Specify the pin names to control the dynamic delay.

Create a MIPI TX Interface

To build a complete MIPI TX interface you need to have at least one data lane and one clock lane. Unlike MIPI RX, they can be in *any* MIPI group. The following figure shows the blocks used for a complete MIPI TX interface.

Figure 48: MIPI TX Interface



1. Refer to the [Efinix® Software User Guide](#) for a listing of available MIPI-related IP cores.



Important: You need to use specific phase shifts for the SLOWCLK, FASTCLK_C, and FASTCLK_D output clocks from the PLL as shown in step 1 below.

1. Add a PLL block with the following settings:

Option	Description
Resource	You can use any PLL resource.
Reference Clock Mode	External
Reference Clock Frequency	User defined.
Output Clocks	mipi_clk—Frequency defined in MIPI IP core, phase shift 0° ⁽⁸⁾ SLOWCLK—Frequency is 1/8 the PHY speed, phase shift 0°, enable feedback.

⁽⁸⁾ This PLL also generates the mipi_clk, which is used in the MIPI IP core. Refer to the user guide for the IP core for details.

Option	Description
	FASTCLK_D–Frequency is the speed you are running the PHY, phase shift 45.00° FASTCLK_C–Frequency is the speed you are running the PHY, phase shift 135.00° For example, if the PHY is running at 1,000 Mbps, FASTCLK_D and FASTCLK_C will run at half that 500 MHz (because it transfers data on both clock edges), and SLOWCLK will run at 125 MHz.
Locked Pin	Turn on

2. Add a GPIO block with these settings to provide the reference clock input to the PLL:

Option	Description
Mode	Input
Pin Name	Any
Connection Type	pll_clkin
GPIO Resource	Assign the dedicated PLL_CLKIN pin that corresponds to the PLL you chose.

3. Add MIPI TX Lane block with these settings:

Option	Description
Mode	data lane
Parallel Clock Pin Name	Name you are using for SLOWCLK.
Serial Clock Pin Name	Name you are using for FASTCLK_D.

4. Repeat step 3 for each MIPI TX data lane you want to implement.
5. Add another MIPI TX Lane block for the clock lane:

Option	Description
Mode	clock lane
Parallel Clock Pin Name	Name you are using for SLOWCLK.
Serial Clock Pin Name	Name you are using for FASTCLK_C.

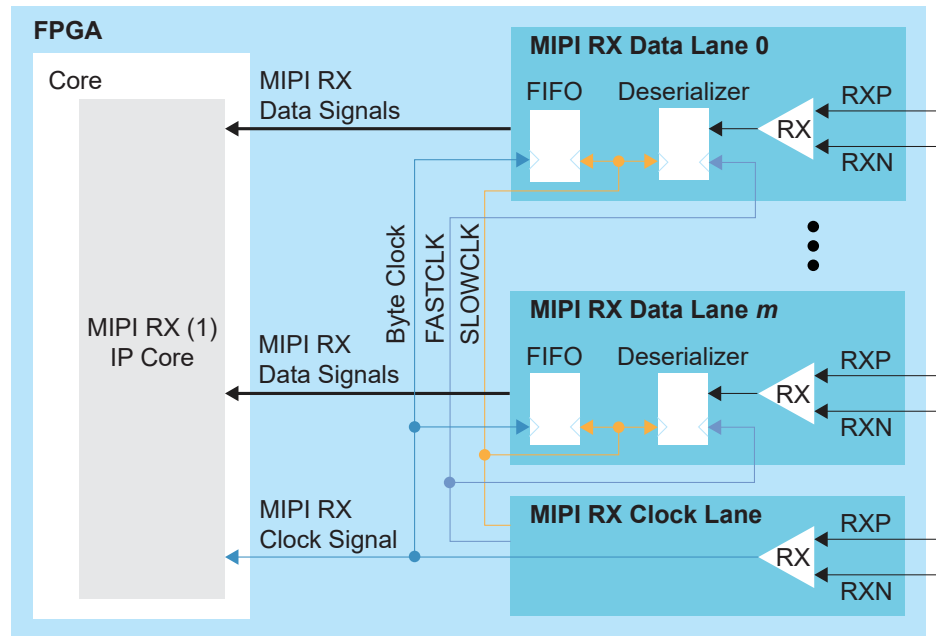
6. Implement the rest of the MIPI TX interface in RTL using a MIPI TX IP core (CSI-2, D-PHY, or DSI). Refer to the user guide for the IP core for instructions.

Create a MIPI RX Interface

To build a complete MIPI RX interface you need to have at least one data lane and one clock lane in the *same* MIPI group. The following figure shows the blocks used for a complete MIPI RX interface.

Tip: The Interface Designer Block Summary shows the MIPI group name in the **Features** property. You can also refer to [MIPI Groups by Package](#) on page 103.

Figure 49: MIPI RX Interface



Interface Blocks

m is the number

1. Refer to the [Efinix® Software User Guide](#) for a listing of available MIPI-related IP cores.

1. Add MIPI RX Lane block for the clock lane:

Option	Description
MIPI Lane Resource	Choose a resource in any MIPI group (all lanes should be in the same MIPI group).
Mode	data lane

2. Repeat step 1 for each MIPI RX data lane you want to implement.

3. Add another MIPI RX Lane block for the clock lane:

Option	Description
MIPI Lane Resource	Choose a resource in the same MIPI group as the data lane(s).
Mode	clock lane
Byte Clock (core) Pin Name	The name for the byte clock that feeds the core and automatically is used to clock the FIFO from the data lanes in this MIPI group.

4. Implement the rest of the MIPI RX interface in RTL using a MIPI RX IP core (CSI-2, D-PHY, or DSI). Refer to the user guide for the IP core for instructions.

Design Check: MIPI Lane Messages

When you check your design, the Interface Designer applies design rules to your configuration settings. The following tables show some of the error and warning messages you may encounter and explains how to fix them.

mipi_ln_rule_resource (error)

Message	Resource name is empty Resource <name> is not a valid MIPI LANE <Rx/Tx> device instance
To fix	Choose a valid resource.

mipi_ln_rule_group_clock (error)

Message	No clock lane was configured in the MIPI LANE group <group> No clock lane was configured in the same MIPI LANE group
To fix	The minimum requirement to create a MIPI interface is one clock lane and one data lane. For RX, they must also be in the same MIPI group. Add a clock lane.

mipi_ln_rule_group_data (error)

Message	Instance is not a resource associated to a MIPI LANE Rx group Instance is not a valid MIPI LANE data lane resource
To fix	In some packages there are not enough MIPI resources in a group to have both a clock and a data lane for RX interfaces. In those cases, you cannot use that pin as a MIPI lane. Choose another resource. See MIPI Groups by Package on page 103.

mipi_ln_rule_rx_clk_conn (error)

Message	Connection type <type> is not supported by the resource The resource does not support clock lane mode
To fix	Not all resources support both GCLK and RCLK connection types. Use the Resource Assigner to pick a different resource that supports GCLK and RCLK.

lvds_rule_rx_distance (error)

Message	These HSIO GPIO must be placed at least 1 pair away from MIPI LANE <name> in order to avoid noise coupling from GPIO to MIPI LANE: <violated list>
To fix	When using HSIO pins as GPIO, make sure to leave at least 1 pair of unassigned HSIO pins between any GPIO and HSIO used as MIPI RX lanes in the same bank. This separation reduces noise.

mipi_ln_rule_rx_empty_pins (error)

Message	Empty pin names found: <list>
To fix	Specify the missing pin names in the list.

mipi_ln_rule_rx_param (error)

Message	Invalid parameters configuration: <features>
To fix	One of the parameters you set was incorrect. Review any other errors for details.

mipi_In_rule_tx_clock (error)

Message	Serial and parallel clocks cannot be the same clock
To fix	You cannot use the same clock for both the serial (FASTCLK_C or FASTCLK_D) and parallel (SLOWCLK) clocks.
Message	Serial clock name is not a PLL output clock
To fix	Use a PLL output clock as the serial (FASTCLK_C or FASTCLK_D) clock.
Message	Parallel clock name is not a PLL output clock
To fix	Use a PLL output as the parallel (SLOWCLK) clock.
Message	Serial and parallel clocks are not from the same PLL instance
To fix	You need to use the same PLL to generate both clocks.
Message	Expected clocks phase shift in <data/clock> mode: Serial: <int> degree Parallel: <int> degree Expected clocks phase shift in <data/clock> mode: Serial: <int> degree Expected clocks phase shift in <data/clock> mode: Parallel: <int> degree
To fix	You need to use specific phase shifts for the clocks. Use the phase shift given in the message.
Message	One of the clock frequencies is 0
To fix	The output clock frequency is invalid. FASTCLK_D and FASTCLK_C should be the same frequency as the PHY. SLOWCLK should be 1/8 the PHY frequency. For example, if the PHY is running at 800 MHz, FASTCLK_D and FASTCLK_C should be 800 MHz and SLOWCLK should be 100 MHz.
Message	Serial clock frequency has to be 8 times faster than parallel clock
To fix	FASTCLK_D and FASTCLK_C should be the same frequency as the PHY. SLOWCLK should be 1/8 the PHY frequency. For example, if the PHY is running at 800 MHz, FASTCLK_D and FASTCLK_C should be 800 MHz and SLOWCLK should be 100 MHz.

mipi_In_rule_tx_clock_region (error)

Message	Serial and Parallel clocks generated by PLL have to be driven to the same clock network. <Serial Parallel> clock <name> was generated by PLL output clock 4 that connects to regional clock network
To fix	In Ti35 and Ti60FPGAs, the PLL's output clock 4 can only drive the regional clock network. You should use the other clock outputs for the serial and parallel clocks.

lvds_rule_tx_distance (error)

Message	These HSIO GPIO must be placed at least 1 pair away from MIPI LANE <name> in order to avoid noise coupling from GPIO to MIPI LANE: <violated list>
To fix	When using HSIO pins as GPIO, make sure to leave at least 1 pair of unassigned HSIO pins between any GPIO and HSIO used as MIPI TX lanes in the same bank. This separation reduces noise.

mipi_In_rule_tx_empty_pins (error)

Message	Empty pin names found: <list>
To fix	Specify the missing pin names in the list.

mipi_ln_rule_tx_param (error)

Message	Invalid parameters configuration: <features>
To fix	One of the parameters you set was incorrect. Review any other errors for details.

mipi_ln_rule_usage (error)

Message	Resource <res name> was assigned multiple times
To fix	You get this error if you choose the same resource for more than one block type (LVDS, MIPI DPHY, or GPIO).

MIPI D-PHY Interface

Contents:

- [MIPI RX D-PHY](#)
- [MIPI TX D-PHY](#)
- [MIPI DPHY TX Interface Designer Settings](#)
- [MIPI DPHY RX Interface Designer Settings](#)



Important: All information is preliminary and pending definition.

In addition to the HSIO, which you can configure as MIPI RX or TX lanes, Titanium FPGAs have hardened MIPI D-PHY blocks, each with 4 data lanes and 1 clock lane. The MIPI D-PHY RX and MIPI D-PHY TX can operate independently with dedicated I/O banks.

You can use the hardened MIPI D-PHY blocks along with the HSIO configured as MIPI D-PHY lanes to create systems that aggregate data from many cameras or sensors.

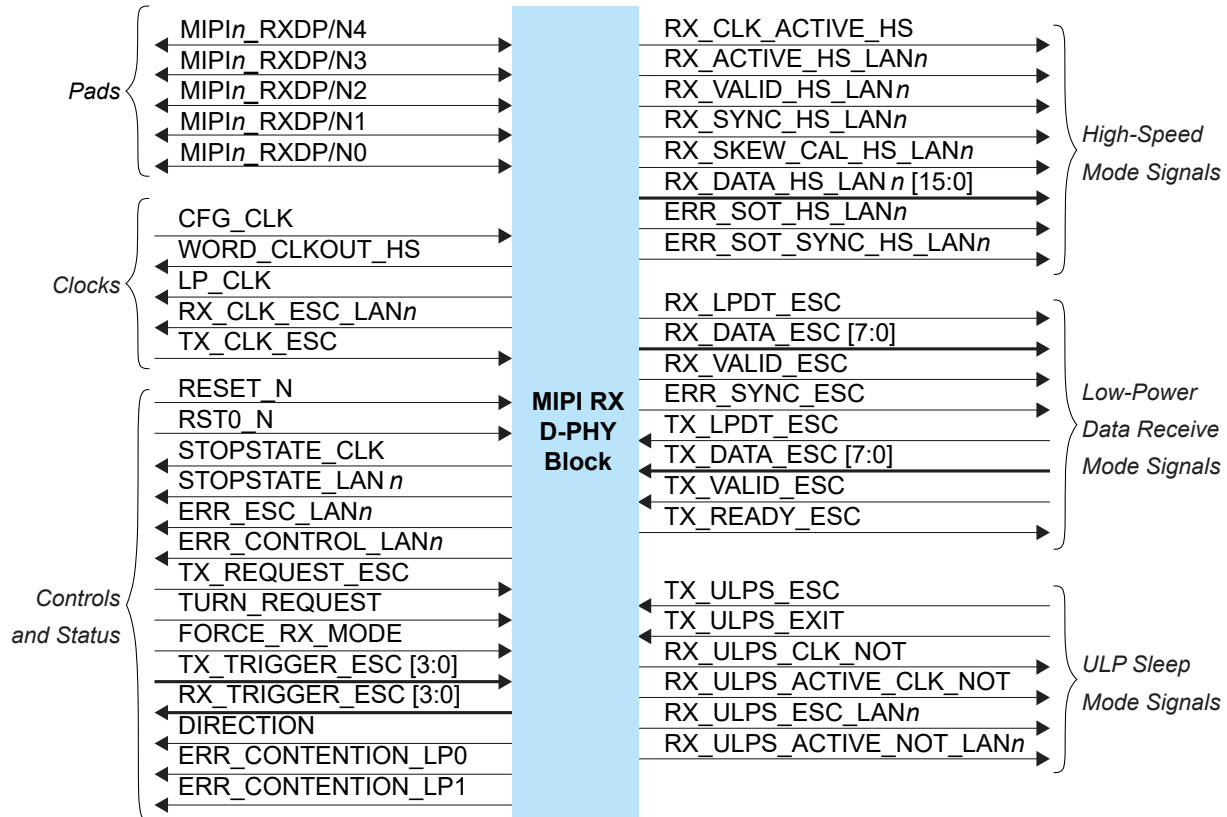
The MIPI TX/RX interface supports the MIPI D-PHY specification v1.2. It has the following features:

- Programmable data lane configuration supporting up to 4 lanes
- High-speed mode supports up to 2.5 Gbps data rates per lane
- Operates in continuous and non-continuous clock modes
- Supports Ultra-Low Power State (ULPS)

MIPI RX D-PHY

The MIPI RX D-PHY is a receiver interface designed to receive data and the control information of MIPI CSI, DSI, or other associated protocols. The MIPI RX D-PHY comprises of one clock lane and up to four data lanes for a single-channel configuration. The MIPI RX D-PHY also interfaces with MIPI-associated protocol controllers via a standard MIPI D-PHY PHY Protocol Interface (PPI) that supports the 8- or 16-bit high-speed receiving data bus.

Figure 50: MIPI RX D-PHY x4 Block Diagram



The status signals provide optional status and error information about the MIPI RX D-PHY interface operation.

Figure 51: MIPI RX D-PHY Interface Block Diagram

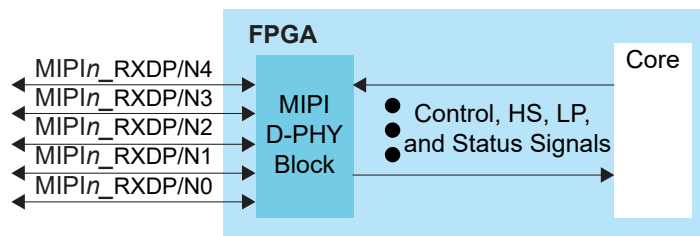


Table 55: MIPI RX D-PHY Clocks Signals (Interface to FPGA Fabric)

Signal	Direction	Clock Domain	Notes
CFG_CLK	Input	N/A	Configuration Clock (used for time counter and EQ calibration). The clock must be between 80 MHz to 120 MHz.
WORD_CLKOUT_HS	Output	N/A	HS Receive Byte/Word clock.
LP_CLK	Output	N/A	Low Power State clock.
RX_CLK_ESC_LANn	Output	N/A	Escape Mode Receive clock.
TX_CLK_ESC	Input	N/A	Escape Mode Transmit clock. The clock must be lower than 20 MHz.

Table 56: MIPI RX D-PHY Control and Status Signals (Interface to FPGA Fabric)

Signal	Direction	Clock Domain	Notes
RESET_N	Input	N/A	Reset. Disables PHY and reset the digital logic.
RST0_N	Input	N/A	Asynchronous FIFO reset and synchronous out of reset.
STOPSTATE_CLK	Output	N/A	Lane in Stop State.
STOPSTATE_LAN _n	Output	N/A	Data Lane in Stop State (Lane N).
ERR_ESC_LAN _n	Output	N/A	Lane <i>n</i> Escape Command Error.
ERR_CONTROL_LAN _n	Output	N/A	Lane <i>n</i> Has Line State Error.
TX_REQUEST_ESC	Input	TX_CLK_ESC	Lane 0 Request TX Escape Mode.
TURN_REQUEST	Input	TX_CLK_ESC	Lane 0 Request Turnaround.
FORCE_RX_MODE	Input	N/A	Lane 0 Force Lane into Receive Mode/Wait for Stop State.
TX_TRIGGER_ESC [3:0]	Input	TX_CLK_ESC	Lane 0 Send a Trigger Event.
RX_TRIGGER_ESC [3:0]	Output	RX_CLK_ESC_LAN0	Lane 0 Received a Trigger Event.
DIRECTION	Output	N/A	Lane 0 Transmit/Receive Direction (0 = TX, 1 = RX).
ERR_CONTENTION_LP0	Output	N/A	Lane 0 Contention Error when driving 0.
ERR_CONTENTION_LP1	Output	N/A	Lane 0 Contention Error when driving 1.

Table 57: MIPI RX D-PHY High Speed Mode Signals (Interface to FPGA Fabric)

Signal	Direction	Clock Domain	Notes
RX_CLK_ACTIVE_HS	Output	N/A	HS Clock Lane Active.
RX_ACTIVE_HS_LAN _n	Output	WORD_CLKOUT_HS	HS Reception Active.
RX_VALID_HS_LAN _n	Output	WORD_CLKOUT_HS	HS Data Receive Valid .
RX_SYNC_HS_LAN _n	Output	WORD_CLKOUT_HS	HS Receiver Sync. Observed.
RX_SKEW_CAL_HS_LAN _n	Output	WORD_CLKOUT_HS	HS Receiver DeSkew Burst Received.
RX_DATA_HS_LAN _n [15:0]	Output	WORD_CLKOUT_HS	HS Receive Data.
ERR_SOT_HS_LAN _n	Output	WORD_CLKOUT_HS	State-of-Transmission (SOT) Error.
ERR_SOT_SYNC_HS_LAN _n	Output	WORD_CLKOUT_HS	SOT Sync. Error.

Table 58: MIPI RX D-PHY Low-Power Data Receive Mode Signals (Interface to FPGA Fabric)

Signal	Direction	Clock Domain	Notes
RX_LPDT_ESC	Output	RX_CLK_ESC_LAN0	Lane 0 enter LPDT RX Mode.
RX_DATA_ESC [7:0]	Output	RX_CLK_ESC_LAN0	Lane 0 LPDT RX Data.
RX_VALID_ESC	Output	RX_CLK_ESC_LAN0	Lane 0 LPDT RX Data Valid.
ERR_SYNC_ESC	Output	N/A	Lane 0 LPDT RX Data Sync. Error.
TX_LPDT_ESC	Input	TX_CLK_ESC	Lane 0 Enter LPDT TX Mode.
TX_DATA_ESC [7:0]	Input	TX_CLK_ESC	Lane 0 LPDT TX Data.
TX_VALID_ESC	Input	TX_CLK_ESC	Lane 0 LPDT TX Data Valid.
TX_READY_ESC	Output	TX_CLK_ESC	Lane 0 LDPT TX Data Ready.

Table 59: MIPI RX D-PHY ULP Sleep Mode Signals (Interface to FPGA Fabric)

Signal	Direction	Clock Domain	Notes
TX_ULPS_ESC	Input	TX_CLK_ESC	Lane 0 Enter ULPS Mode.
TX_ULPS_EXIT	Input	TX_CLK_ESC	Lane 0 Exit ULPS Mode.
RX_ULPS_CLK_NOT	Output	N/A	CLK0 Enter ULPS Mode.
RX_ULPS_ACTIVE_CLK_NOT	Output	N/A	CLK0 is in ULPS (Active Low).
RX_ULPS_ESC_LAN n	Output	RX_CLK_ESC_LAN n	Lane n Enter ULPS Mode.
RX_ULPS_ACTIVE_NOT_LAN n	Output	N/A	Lane n is in ULPS (Active Low).

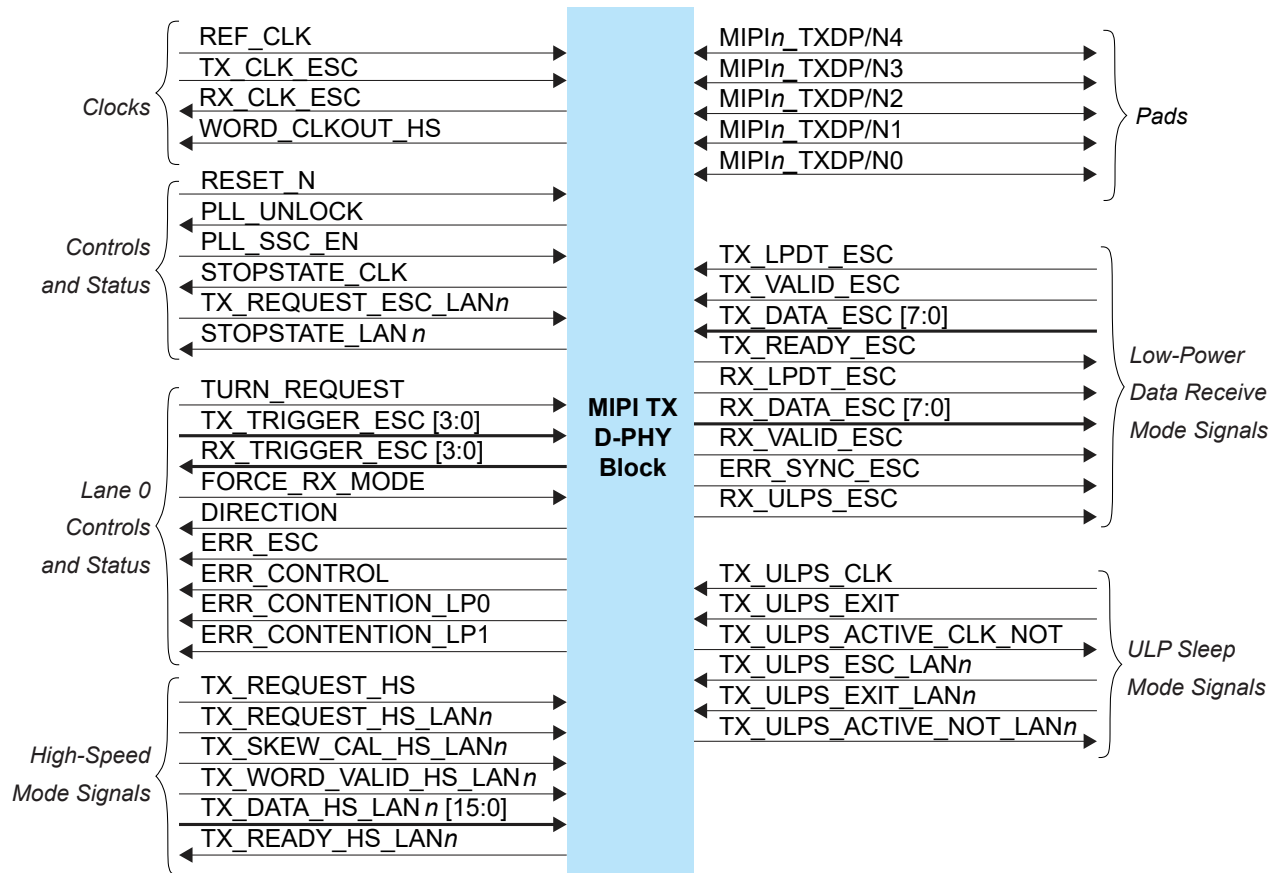
Table 60: MIPI RX D-PHY Pads

Pad	Direction	Description
MIPI n _RXDP[4:0]	Bidirectional	MIPI transceiver P pads.
MIPI n _RXDN[4:0]	Bidirectional	MIPI transceiver N pads.

MIPI TX D-PHY

The MIPI TX D-PHY is a transmitter interface designed to transmit data and the control information of MIPI CSI, DSI, or other associated protocols. The MIPI TX D-PHY comprises of one clock lane and up to four data lanes for a single-channel configuration. The MIPI TX D-PHY also interfaces with MIPI-associated protocol controllers via a standard MIPI D-PHY PPI that supports the 8- or 16-bit high-speed receiving data bus.

Figure 52: MIPI TX D-PHY x4 Block Diagram

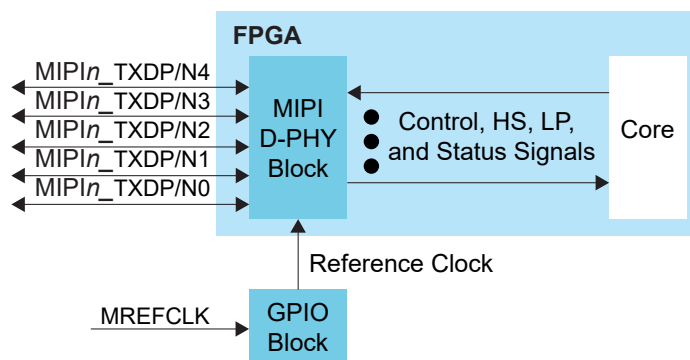


The MIPI TX D-PHY block requires an escape clock (TX_CLK_ESC) for use when the MIPI interface is in escape (low-power) mode, which runs up to 20 MHz.



Note: Efinix recommends that you set the escape clock frequency as close to 20 MHz as possible.

Figure 53: MIPI TX D-PHY Interface Block Diagram



Note: GPIO block is the default reference clock source. However, the PLL and core clock out can also be set as the reference clock source.

Table 61: MIPI TX D-PHY Clocks Signals (Interface to FPGA Fabric)

Signal	Direction	Clock Domain	Notes
REF_CLK	Input	N/A	Reference Clock. The clock must be between 12 MHz to 52 MHz.
TX_CLK_ESC	Input	N/A	Escape Mode Transmit Clock, used to generate escape sequence. The clock must be less than 20 MHz.
RX_CLK_ESC	Output	N/A	Escape Mode Receive Clock (lane 0 only)
WORD_CLKOUT_HS	Output	N/A	HS Transmit Byte/Word Clock. This signal must be 1/8 of the bit-rate in normal 8-bit HS-PPI D-PHY mode, or 1/16 of the bit-rate in 16-bit PHY mode.

Table 62: MIPI TX D-PHY Control and Status Signals (Interface to FPGA Fabric)

Signal	Direction	Clock Domain	Notes
RESET_N	Input	N/A	Reset. Disables PHY and reset the digital logic.
PLL_UNLOCK	Output	N/A	PLL is in unlock state.
PLL_SSC_EN	Input	N/A	(Optional) PLL SSC Enable: Enable: 1 Always disabled: 0 Driven by active signal for dynamic enable
STOPSTATE_CLK	Output	N/A	Clock Lane in Stop State (Clk 0).
TX_REQUEST_ESC_LAN _n	Input	TX_CLK_ESC	Escape Mode Transmit Request (Lane N).
STOPSTATE_LAN _n	Output	N/A	Data Lane in Stop State (Lane N).

Table 63: MIPI TX D-PHY Lane 0 Control and Status Signals (Interface to FPGA Fabric)

Signal	Direction	Clock Domain	Notes
TURN_REQUEST	Input	TX_CLK_ESC	Lane 0 Turnaround Request.
TX_TRIGGER_ESC [3:0]	Input	TX_CLK_ESC	Lane 0 Send an Escape Mode Trigger Event.
RX_TRIGGER_ESC [3:0]	Output	RX_CLK_ESC	Lane 0 Received an Escape Mode Trigger Event.
FORCE_RX_MODE	Input	N/A	Lane 0 Force into Receive Mode/Wait for Stop.
DIRECTION	Output	N/A	Lane 0 Transmit/Receive Direction: 0: TX, 1: RX
ERR_ESC	Output	N/A	Lane 0 Escape Command Error.
ERR_CONTROL	Output	N/A	Lane 0 Line State Error.
ERR_CONTENTION_LP0	Output	N/A	Lane 0 Line Contention Detected (when driving 0).
ERR_CONTENTION_LP1	Output	N/A	Lane 0 Line Contention Detected (when driving 1).

Table 64: MIPI TX D-PHY High Speed Mode Signals (Interface to FPGA Fabric)

Signal	Direction	Clock Domain	Notes
TX_REQUEST_HS	Input	WORD_CLKOUT_HS	HS Clock Request (Clk 0).
TX_REQUEST_HS_LAN _n	Input	WORD_CLKOUT_HS	HS Transmit Request and Data Valid (Lane 0-3).
TX_SKEW_CAL_HS_LAN _n	Input	WORD_CLKOUT_HS	HS Skew Calibration (Lane N).
TX_WORD_VALID_HS_LAN _n	Input	WORD_CLKOUT_HS	HS High Byte Valid (Lane N) for 16-bit mode.
TX_DATA_HS_LAN _n [15:0]	Input	WORD_CLKOUT_HS	HS Transmit Data (Lane N).
TX_READY_HS_LAN _n	Output	WORD_CLKOUT_HS	HS Transmit Ready (Lane N).

Table 65: MIPI TX D-PHY Low-Power Data Receive Mode Signals (Interface to FPGA Fabric)

Signal	Direction	Clock Domain	Notes
TX_LPDT_ESC	Input	TX_CLK_ESC	Lane 0 Enter LPDT Mode.
TX_VALID_ESC	Input	TX_CLK_ESC	Lane 0 LPDT Data Valid .
TX_DATA_ESC [7:0]	Input	TX_CLK_ESC	Lane 0 LPDT Data Bus.
TX_READY_ESC	Output	TX_CLK_ESC	Lane 0 LPDT Data Ready.
RX_LDPT_ESC	Output	RX_CLK_ESC	Escape LP Data Receive Mode.
RX_DATA_ESC[7:0]	Output	RX_CLK_ESC	Escape Mode Receive Data.
RX_VALID_ESC	Output	RX_CLK_ESC	Escape Mode Receive Data Valid.
ERR_SYNC_ESC	Output	N/A	LPDT Data Sync Error.
RX_ULPS_ESC	Output	RX_CLK_ESC	Lane 0 entered ULPS mode.

Table 66: MIPI TX D-PHY ULP Sleep Mode (Interface to FPGA Fabric)

Signal	Direction	Clock Domain	Notes
TX_ULPS_CLK	Input	TX_CLK_ESC	CLK0 to enter Ultra-Low Power State.
TX_ULPS_EXIT	Input	TX_CLK_ESC	CLK0 to exit Ultra-Low Power State.
TX_ULPS_ACTIVE_CLK_NOT	Output	N/A	Clock Lane in ULP State - Active Low (Clk 0).
TX_ULPS_ESC_LAN n	Input	TX_CLK_ESC	Lane n to enter Ultra-Low Power State.
TX_ULPS_EXIT_LAN n	Input	TX_CLK_ESC	Lane n to exit Ultra-Low Power State.
TX_ULPS_ACTIVE_NOT_LAN n	Output	N/A	Data Lane in ULP State - Active Low (Lane N).

Table 67: MIPI TX D-PHY Pads

Pad	Direction	Description
MIPI n _TXDP[4:0]	Bidirectional	MIPI transceiver P pads.
MIPI n _TXDN[4:0]	Bidirectional	MIPI transceiver N pads.

MIPI DPHY TX Interface Designer Settings

The following tables describe the settings for the Titanium MIPI DPHY TX blocks in the Interface Designer.

Table 68: Base Tab

Parameter	Choices	Notes
PHY bandwidth in Mbps	Integer up to 2500	Specify the bandwidth. Default: 2500
Instance Name	User defined	
MIPI TX Resource	None, MIPI_TX0, MIPI_TX1, MIPI_TX2, MIPI_TX3	Choose the resource.
Reference Clock Frequency	12.0, 19.2, 25.0, 26.0, 27.0, 38.4, 52.0	Choose the frequency for the reference clock.
Reference Clock Source Type	core, gpio, pll	Choose which resource generates the reference clock. For gpio and pll , the Block Editor shows you which resource to connect as the reference clock. For core , you specify the clock name.

Table 69: Control/Status Tab

Option	Choices	Notes
Enable Spread Spectrum Clock (SSC)	On or off	Turn on to enable SSC.
SSC Frequency for MIPI Internal PLL (kHz)	0 - 200	Spread-spectrum clock frequency setting. 20 - 200 kHz. Default: 0
SSC Initial Amplitude for MIPI Internal PLL (PPM)	0 - 50000	Spread-spectrum clock initial spread down amount. 2500 - 50000 ppm. Default: 0
SSC Amplitude for MIPI Internal PLL (PPM)	0 - 50000	Spread-spectrum clock amount. 2500 to 50000 ppm. Default: 0
<description> Pin Name	User defined	Control and status pin names. Efinix recommends that you use the defaults.
HS Transmit Byte/Word Clock Connection Type	gclk, rclk	Choose whether to connect to a global clock (gclk) or regional clock (rclk). Default: gclk
Invert Escape Mode Transmit Clock Pin	On or off	Turn on to invert the clock.

Table 70: Clock Lane Tab

Option	Choices	Notes
Escape Mode Receive Clock Pin Name	User defined	Specify the clock name.
Escape Mode Receive Clock Connection Type	normal, rclk	normal: Default. The clock signal is an input signal to the core. rclk: The clock signal is feeding the regional clock network.
<description> Pin Name	User defined	Clock lane pin names. Efinix recommends that you use the defaults.

Table 71: Data Lane Tab

Option	Choices	Notes
Enable Turn-around Feature in Data Lane 0	On or off	Lane 0 can operate as a bi-directional data lane when this option is on. Default: on
Number of data lanes	1, 2, 4	Choose the number of lanes. Default: 4
Width of the data bus	8, 16	Specify the width. Default: 8
<description> Pin Name	User defined	Data lane pin names. Efinix recommends that you use the defaults.

Table 72: Lane Mapping Tab

Parameter	Choices	Notes
Phy Lane <i>n</i>	clk, data0, data1, data2, data3, unused	The MIPI TX block supports 4 data lanes and 1 clock lane. Choose which lane to associate with the MIPI pad. The lane mapping must be unique, which the software enforces.
Swap P&N Pin	On or off	Turn on to change which pin is P or N. This setting can be helpful when laying out your board.

Table 73: Timing Tab

Parameter	Choices	Notes
T _{CYCLE_SEL} (ns)	0 - 7	
T _{PLL_FBK_FRA} (ns)	0 - 16777215	
T _{PLL_FBK_INT} (ns)	0 - 511	
T _{PLL_PRE_DIV} (ns)	0 - 3	
T _{CLANE_HS_CLK_POST_TIME} (ns)	0 - 255	
T _{CLANE_HS_CLK_PRE_TIME} (ns)	0 - 255	
T _{CLANE_HS_PRE_TIME} (ns)	0 - 255	
T _{CLANE_HS_TRAIL_TIME} (ns)	0 - 255	
T _{CLANE_HS_ZERO_TIME} (ns)	0 - 255	
T _{DLANE_HS_PRE_TIME} (ns)	0 - 255	
T _{DLANE_HS_TRAIL_TIME} (ns)	0 - 255	
T _{DLANE_HS_ZERO_TIME} (ns)	0 - 255	

MIPI DPHY RX Interface Designer Settings

The following tables describe the settings for the Titanium MIPI DPHY RX blocks in the Interface Designer.

Table 74: Base Tab

Parameter	Choices	Notes
Instance Name	User defined	
MIPI RX Resource	None, MIPI_RX0, MIPI_RX1, MIPI_RX2, MIPI_RX3	Choose the resource.

Table 75: Control/Status Tab

Option	Choices	Notes
Calibration Clock Frequency (MHz)	80 - 120	Specify the frequency. Default: 0
<description> Pin Name	User defined	Control and status pin names. Efinix recommends that you use the defaults.
HS Transmit Byte/Word Clock Connection Type	gclk, rclk	Choose whether to connect to a global clock (gclk) or regional clock (rclk). Default: gclk
Invert Escape Mode Transmit Clock Pin	On or off	Turn on to invert the clock.

Table 76: Clock Lane Tab

Option	Choices	Notes
<description> Pin Name	User defined	Clock lane pin names. Efinix recommends that you use the defaults.

Table 77: Data Lane Tab

Option	Choices	Notes
Enable Turn-around Feature in Data Lane 0	On or off	Lane 0 can operate as a bi-directional data lane when this option is on. Default: on
Number of data lanes	1, 2, 4	Choose the number of lanes. Default: 4
Width of the data bus	8, 16	Specify the width. Default: 8
Lane <i>n</i> : Escape Mode Receive Clock Pin Name	User defined	Specify the clock.
Lane 0: Escape Mode Receive Clock Connection Type	normal, rclk	normal: Default. The clock signal is an input signal to the core. rclk: The clock signal is feeding the regional clock network.
Lane <i>n</i> : <description> Pin Name	User defined	Data lane pin names. Efinix recommends that you use the defaults.

Table 78: Lane Mapping Tab

Parameter	Choices	Notes
Phy Lane <i>n</i>	clk, data0, data1, data2, data3, unused	The MIPI TX block supports 4 data lanes and 1 clock lane. Choose which lane to associate with the MIPI pad. The lane mapping must be unique, which the software enforces.
Swap P&N Pin	On or off	Turn on to change which pin is P or N. This setting can be helpful when laying out your board.

Chapter 11

PLL

Contents:

- [About the PLL Interface](#)
- [Using the PLL V3 Block](#)
- [Implementing a Zero-Delay Buffer](#)
- [Design Check: PLL Errors](#)

About the PLL Interface

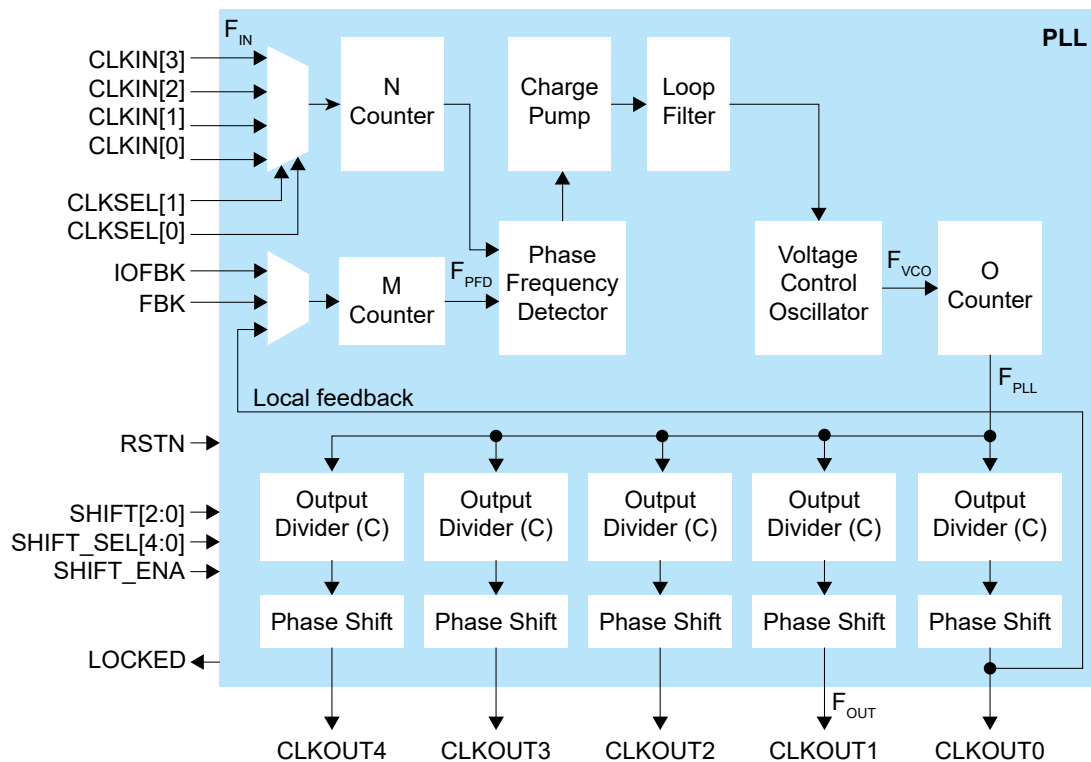
Titanium FPGAs have PLLs to synthesize clock frequencies. The PLLs are located in the corners of the FPGA. You can use the PLL to compensate for clock skew/delay via external or internal feedback to meet timing requirements in advanced applications. The PLL reference clock has up to four sources. You can dynamically select the PLL reference clock with the CLKSEL port. (Hold the PLL in reset when dynamically selecting the reference clock source.)

The PLL consists of a pre-divider counter (N counter), a feedback multiplier counter (M counter), a post-divider counter (O counter), and output dividers (C).



Note: You can cascade the PLLs in Titanium FPGAs. To avoid the PLL losing lock, Efinix recommends that you do not cascade more than two PLLs.

Figure 54: PLL Block Diagram



The counter settings define the PLL output frequency:

Local and Core Feedback Mode	Where:
$F_{PPFD} = F_{IN} / N$ $F_{VCO} = (F_{PPFD} \times M \times O \times C_{FBK})^{(9)}$ $F_{PLL} = F_{VCO} / O$ $F_{OUT} = (F_{IN} \times M \times C_{FBK}) / (N \times C)$	F_{VCO} is the voltage control oscillator frequency F_{PLL} is the post-divider PLL VCO frequency F_{OUT} is the output clock frequency F_{IN} is the reference clock frequency F_{PPFD} is the phase frequency detector input frequency O is the post-divider counter C is the output divider



Note: Refer to the [PLL Timing and AC Characteristics](#) for F_{VCO} , F_{OUT} , F_{IN} , F_{PLL} , and F_{PPFD} values.

Figure 55: PLL Interface Block Diagram

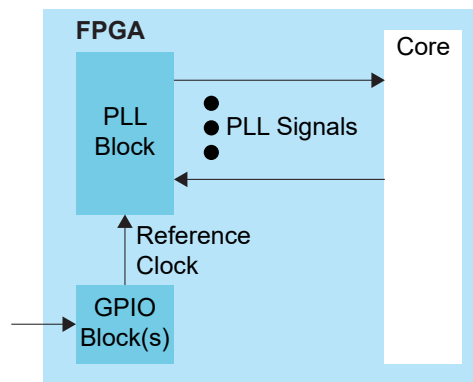


Table 79: PLL Signals (Interface to FPGA Fabric)

Signal	Direction	Description
CLKIN[3:0]	Input	Reference clocks driven by I/O pads or core clock tree.
CLKSEL[1:0]	Input	You can dynamically select the reference clock from one of the clock in pins.
RSTN	Input	Active-low PLL reset signal. When asserted, this signal resets the PLL; when de-asserted, it enables the PLL. Connect this signal in your design to power up or reset the PLL. Assert the RSTN pin for a minimum pulse of 10 ns to reset the PLL. Assert RSTN when dynamically changing the selected PLL reference clock.
FBK	Input	Connect to a clock out interface pin when the PLL is in core feedback mode.
IOFBK	Input	Connect to a clock out interface pin when the PLL is in external I/O feedback mode.

⁽⁹⁾ $(M \times O \times C_{FBK})$ must be ≤ 255 .

Signal	Direction	Description
CLKOUT0 CLKOUT1 CLKOUT2 CLKOUT3 CLKOUT4	Output	PLL output. You can route these signals as input clocks to the core's GCLK network. Ti35, Ti60: CLKOUT4 can only feed the top or bottom regional clocks. Ti35, Ti60: All PLL outputs lock on the negative clock edge. The Interface Designer inverts the clock polarity on the leaf cells by default (Invert Output Clock option unchecked). You can use CLKOUT0 only for clocks with a maximum frequency of 4x (integer) of the reference clock. If all your system clocks do not fall within this range, you should dedicate one unused clock for CLKOUT0.
LOCKED	Output	Goes high when PLL achieves lock; goes low when a loss of lock is detected. Connect this signal in your design to monitor the lock status.
SHIFT[2:0]	Input	(Optional) Dynamically change the phase shift of the output selected to the value set with this signal. Possible values from 000 (no phase shift) to 111 (3.5 F _{PLL} cycle delay). Each increment adds 0.5 cycle delay.
SHIFT_SEL[4:0]	Input	(Optional) Choose the output(s) affected by the dynamic phase shift.
SHIFT_ENA	Input	(Optional) When high, changes the phase shift of the selected PLL(s) to the new value.



Learn more: Refer to the device data sheet for the list of PLL reference clock assignments.

Using the PLL V3 Block

Titanium FPGA's PLL block lets you configure the reference clock, feedback options, frequency, and output clocks for the PLL. This PLL is referenced as V3. You set up the PLL using the PLL Clock Calculator, which provides an easy-to-use graphical way to specify the frequencies and other settings.

- In the PLL's **Properties** tab, you specify general settings such as the instance name, PLL resource, clock source, and external clock.
- Click the **Automated Clock Calculation** button to open the PLL Clock Calculator.

Reference Clock Settings

The PLL has four possible reference clocks. Depending on the PLL, one or two of the clocks can come from the FPGA core, and two or three can come from off chip. You select the clocks using the **Clock Source** drop-down box:

- **core**—The PLL reference clock comes from the FPGA core.
- **external**—Enables clock 0, 1, or 2. The PLL reference clock comes from an external pin. The GUI displays the resource(s) that can be the reference clock.



Note: In this mode, a GPIO block with a **pll_clkin** connection type must generate the reference clock(s). The software displays which resource(s) you can use (and the instance name if you have created it).

1. Add a GPIO block.
2. Enter the instance name.
3. Choose **input** as the mode.
4. Choose **pll_clkin** as the connection type.
5. In the Resource Assigner, assign it to the resource shown in the PLL's Properties tab.

- **dynamic**—Enables all four clocks; requires a clock selector bus to choose the clock dynamically. The GUI displays the resource(s) that can be the reference clock.

Using the PLL Clock Calculator

The PLL Clock Calculator provides a graphical way for you to set up the advanced PLL block. When you open the calculator, the GUI appears in automatic mode, which provides basic settings. You can:

- Choose the feedback mode (**Local**, **Core**, or **External**).
- Turn signals on (gray x) or off (green arrow) by clicking the icons next to the signal.
- Specify the signal names.
- Specify the phase shift.
- Choose which clock has feedback (for core feedback mode).

The Titanium PLL supports dynamic phase shifting. To enable it, click the **Dynamic** button for the clock output. The calculator adds three additional pins that you use to control the dynamic shifting. You need to specify the pin names.

As you make selections, the calculator determines the values for the pre-divider, multiplier, post divider, and clock dividers that meet your settings. The GUI prompts you if you make selections that are impossible to solve.

In manual mode, the interface displays the PLL's internal block diagram, and provides boxes for you to set the values for the pre-divider, multiplier, post divider, and clock dividers. As you adjust the values, the calculator prompts you if you make settings that result in F_{VCO} values that are out of range or are impossible to solve. When you turn manual mode off, the calculator adjusts the output clock frequencies to match the manual settings. If you have incorrect settings for the pre-divider, multiplier, post divider, and clock dividers, when you turn manual mode off, the calculator adjusts the values to ones that allow a valid solution.

When you are finished using the calculator, click **Finish** to save your settings and close the GUI.

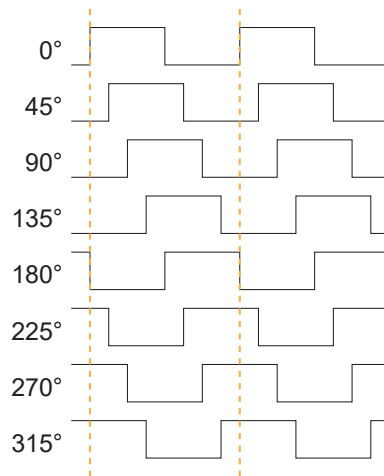
Understanding PLL Phase Shifting

The PLL supports clock phases from 0 to 315 degrees.

- You can set a phase shift for a clock output in the PLL Clock Calculator. The calculator tries to get as close to that number as it can.
- If you want to set the phase shift manually, where you use the options to set the M, N, O, divider, and shift values, the phase shift results you get are 0, 45, 90, 135, 180, 225, 270, and 315.

You can also control the phase shifting dynamically. To turn on dynamic phase shifting, click the Dynamic button next to the clock frequency in the PLL Clock Calculator. The GUI adds three pins to control the shift. the shift select, and the shift enable. You can dynamically change the phases simultaneously.

Figure 56: Example PLL Clock Phases



Manually Configuring the PLL

If you want more control over the PLL, you can manually configure it in the PLL Clock Calculator using manual mode. To enable this mode, click the **Manual Mode** slider to turn it on.

Table 80: PLL Manual Mode Options

Option	Choices	Description
Pre-Divider (N)	1, 2, 4	The pre-divider value.
Multiplier (M)	1, 2, 4	The multiplier value.
Post-Divider (O)	1, 2, 4, 8, 16, 32, 64, 128	Post-divider value.
CLK Divider n	1 - 128	Clock divider for each output.

Option	Choices	Description
Phase Shift Value n	0 - 7	Post-divider VCO cycle delay to phase shift, in degrees: 0: No phase shift 1: 0.5 2: 1 3: 1.5 4: 2 5: 2.5 6: 3 7: 3.5

The post-divider VCO cycle delay relates to the phase shift as shown in the following equation:

$$\text{Phase shift (in degrees)} = ((\text{Post-divider VCO cycle delay} \times O \times F_{\text{CLKOUT}}) / F_{\text{VCO}}) \times 360$$

where:

- O is the post-VCO division ration
- F_{CLKOUT} is the output frequency
- F_{VCO} is the VCO frequency

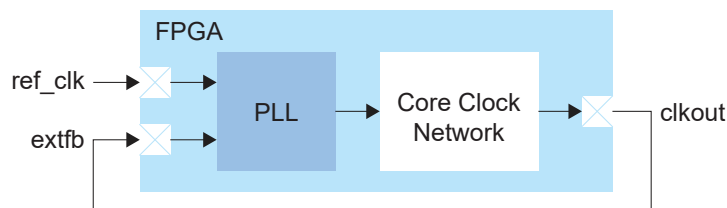
Implementing a Zero-Delay Buffer

Titanium PLLs can implement a zero-delay buffer circuit. In this mode, the PLL removes all of the clock-insertion delay from the input I/O buffer and core clock tree. You may want to use this buffer when you have a single clock signal that fans out to more than one destination with low skew.

To implement a zero-delay buffer:

- In the PLL Clock Calculator, use **Feedback Mode > External**.
- Add a GPIO for the clock output.
- Add a second GPIO for the external feedback.
- Add a third GPIO for the PLL reference clock.
- On your board, connect the clock output pin to the external feedback pin.

Figure 57: Zero-Delay Buffer Block Diagram



Choose the location of the `clkout` and `extfb` pins to minimize any board delay.

The following code example shows an `.isf` that implements a zero-delay buffer for the Ti60 F225.

```

# Efinity Interface Configuration
# Version: 2021.1.165.2.19
# Date: 2021-09-23 15:23
#
# Copyright (C) 2017 - 2021 Efinix Inc. All rights reserved.
#

```

```

# Device: Ti60F225
# Package: 225-ball FBGA (preliminary)
# Project: r4000
# Configuration mode: active (x1)
# Timing Model: C4 (preliminary)

# Create instance
design.create_clockout_gpio("clkout")
design.create_pll_ext_fb_gpio("fbk_clk")
design.create_pll_input_clock_gpio("ref_clk")
design.create_block("pll_inst1", "PLL")

# Set property, non-defaults
design.set_property("clkout", "OUT_CLK_PIN", "clk")
design.set_property("pll_inst1", "CLKOUT0_EN", "1", "PLL")
design.set_property("pll_inst1", "CLKOUT1_EN", "0", "PLL")
design.set_property("pll_inst1", "CLKOUT2_EN", "0", "PLL")
design.set_property("pll_inst1", "CLKOUT3_EN", "0", "PLL")
design.set_property("pll_inst1", "CLKOUT4_EN", "0", "PLL")
design.set_property("pll_inst1", "REFCLK_SOURCE", "EXTERNAL", "PLL")
design.set_property("pll_inst1", "CLKOUT0_DIV", "82", "PLL")
design.set_property("pll_inst1", "CLKOUT0_DYNPHASE_EN", "0", "PLL")
design.set_property("pll_inst1", "CLKOUT0_PHASE_SETTING", "0", "PLL")
design.set_property("pll_inst1", "CLKOUT0_PIN", "clk", "PLL")
design.set_property("pll_inst1", "EXT_CLK", "EXT_CLK0", "PLL")
design.set_property("pll_inst1", "LOCKED_PIN", "", "PLL")
design.set_property("pll_inst1", "M", "1", "PLL")
3design.set_property("pll_inst1", "N", "1", "PLL")
4design.set_property("pll_inst1", "O", "2", "PLL")
design.set_property("pll_inst1", "OUTPUT_CLOCKS_INVERTED", "0", "PLL")
design.set_property("pll_inst1", "PHASE_SHIFT_ENA_PIN", "", "PLL")
design.set_property("pll_inst1", "PHASE_SHIFT_PIN", "", "PLL")
design.set_property("pll_inst1", "PHASE_SHIFT_SEL_PIN", "", "PLL")
design.set_property("pll_inst1", "REFCLK_FREQ", "33.33", "PLL")
design.set_property("pll_inst1", "RSTN_PIN", "", "PLL")
design.set_property("pll_inst1", "FEEDBACK_MODE", "EXTERNAL", "PLL")
design.set_property("pll_inst1", "FEEDBACK_CLK", "CLK0", "PLL")

# Set resource assignment
design.assign_pkg_pin("clkout", "M7")
8design.assign_pkg_pin("ref_clk", "P2")
9design.assign_pkg_pin("fbk_clk", "R6")
10design.assign_resource("pll_inst1", "PLL_BL0", "PLL")

```

Design Check: PLL Errors

When you check your design, the Interface Designer applies design rules to your configuration settings. The following tables show some of the error messages you may encounter and explains how to fix them.

pll_rule_dynamic_shift_feedback (error)

Message	Output clock <name> used as feedback cannot be set with dynamic phase shift
To fix	If you are using a PLL output clock for feedback, you cannot use dynamic phase shifting. Instead, specify a phase.

pll_rule_dynamic_shift_pin (error)

Message	Dynamic phase shift is enabled but missing pin names: <list>
To fix	For a PLL output, if you are turn on Dynamic for the phase shift, you also need to specify names for the SHIFT, SHIFT_SELECT, and SHIFT_ENA pins.

pll_rule_extfb_io (error)

Message	External feedback and reference clock have to be of the same instance type and IO standard
To fix	Use the same I/O standard for the external feedback clock and reference clock GPIO blocks.

pll_rule_extfb_resource (error)

Message	There can only be one configured resource for external IO feedback
To fix	In external feedback mode, you can only specify one clock out pin for feedback.
Message	External IO feedback resource <name> is not configured as pll_extfb connection
To fix	In the GPIO block that is your feedback resource, set the Connection Type to pll_extfb .
Message	The resource for external IO feedback is not configured
To fix	Add a GPIO block in input mode, set the Connection Type to pll_extfb , and assign it to a resource that supports the pll_extb connection type.

pll_rule_fb_freq (error)

Message	Feedback frequency <#>MHz is out of range. Min=<>MHz Max=<>MHz
To fix	The feedback frequency needs to be within the range specified. Adjust the parameters to meet that requirement.

pll_rule_feedback_clock (error)

Message	Feedback clock name is required with non-internal feedback
To fix	You need to specify a clock pin name when you are not using internal feedback mode.
Message	Feedback clock name <string> is not from the same PLL
To fix	You need to use one of the output clocks from the PLL you are configuring as the feedback clock. You cannot use an output clock from a different PLL.
Message	Feedback clock in local mode has to connect to output clock 0
To fix	When Feedback Mode is Local , you can only use output clock 0 for feedback.
Message	Feedback clock <string> is not connected to pll clkout
To fix	The feedback clock you are using needs to be one of the output clocks from the PLL.

pll_rule_feedback_mode (error)

Message	Internal feedback mode is not supported
To fix	You may receive this error when configuring an interface with the API. Do not use internal feedback mode as it's not supported.

pll_rule_input_freq (error)

Message	Input Frequency <float> MHz (after pre-divider) is out of range. Min=<float>MHz Max=<float>MHz
To fix	Assign the reference clock frequency to a value within the specified range.

pll_rule_input_freq_limit (error)

Message	Input Frequency <float> MHz is out of range. Min=<float>MHz Max=<float>MHz
To fix	Assign the right reference clock frequency.

pll_rule_inst_name (error)

Message	Instance name is empty Valid characters are alphanumeric characters with dash and underscore only
To fix	Specify a valid instance name.

pll_rule_mipi_tx_clock (error)

Message	PLL output clock <name> is not allowed to connect to MIPI TX Lane Serial and Parallel clocks at the same time
To fix	You cannot use the slow clock for one MIPI TX lane as the fast clock for a different MIPI TX lane.

pll_rule_multiplier (error)

Message	Multiplier is out of range. Min=<int> Max=<int>
To fix	The multiplier frequency needs to be within the range specified. Adjust the parameters to meet that requirement.

pll_rule_output_clock (error)

Message	At least one output clock must be configured
To fix	Configure at least one PLL output clock and specify the output clock pin name.
Message	Output clock count is out of range. Min=<int>Max=<int>
To fix	You have specified the wrong number of output clocks (too many or none).

pll_rule_output_divider (error)

Message	Output divider for <clock name> is invalid. Valid values are between 1-128
To fix	Choose a value between 1 and 128.

pll_rule_output_freq (error)

Message	Output frequency <float>MHz is out of range. Min=<float>MHz Max=<float>MHz
To fix	The output frequency needs to be within the range specified. Adjust the parameters to meet that requirement.

pll_rule_output_name (error)

Message	PLL output clock names have to be unique. Duplicates found: <list of string>
To fix	You get this error when you use duplicate clock names. Rename them.

pll_rule_output_number (error)

Message	Output number for <clk name> is invalid. It must be between 0 to <int>
To fix	The output clocks are numbered (e.g., CLKOUT3). Make sure that the number is within specified range.

pll_rule_param (error)

Message	Invalid parameters configuration: <feature>
To fix	Performs a general check for invalid parameters. Review the other error messages.

pll_rule_pll_freq (error)

Message	PLL Frequency is out of range, Freq= < value > Min= < min > MHz Max= < max > MHz
To fix	The maximum post-divided VCO clock fmax is 4,000 Mhz. Change the PLL clock calculator settings so that it is in range.

pll_rule_post_divider (error)

Message	Post-divider is invalid. Valid values are <list of int>
To fix	Choose a post divider value from the list shown.

pll_rule_pre_divider (error)

Message	Pre-divider is out of range. Min=<int> Max=<int>
To fix	The pre-divider frequency needs to be within the range specified. Adjust the parameters to meet that requirement.

pll_rule_refclk (error)

Message	Bonded external reference clock pin has to be specified in dynamic mode
To fix	When using dynamic as the Clock Source , the PLL expects to find the resource for the external clock(s). Add a GPIO block in input mode, set the Connection Type to pll_clkln , and assign it to the resource shown in the PLL Properties tab under Dynamic Clock .
Message	Both core refclk pins have to be specified in dynamic mode
To fix	When using dynamic as the Clock Source , you need to specify the names for the core clocks 0 and 1. (Some PLLs use 2 core clocks in dynamic mode.)
Message	Core refclk pin has to be specified in core mode
To fix	When using core as the Clock Source , you need to specify the pin name.
Message	Core refclk pin has to be specified in dynamic mode
To fix	When using dynamic as the Clock Source , you need to specify the names for core clock 0. (Some PLLs use only 1 core clock in dynamic mode.)
Message	External refclk pin has to be set in external mode
To fix	When using external as the Clock Source , the PLL expects to find the resource for the external clock. Add a GPIO block in input mode, set the Connection Type to pll_clkln , and assign it to the resource shown in the PLL Properties tab under External Clock .
Message	External reference clock resource {} is not configured as pll_clkln connection
To fix	You use a GPIO block configured in alternate connection mode to be the reference clock for the PLL. Change the GPIO Connection Type to pll_clkln .
Message	Reference clock at <resource> connected to external clock pin {0 1 2} has not been defined
To fix	The PLL expects to find the resource for the external clock. Add a GPIO block in input mode, set the Connection Type to pll_clkln , and assign it to the resource shown in the PLL Properties tab under External Clock .
Message	Invalid external clock {0 1 2} resource selected: Resource Unbonded
To fix	In the FPGA/package combination you are using, you cannot use the external clock resource specified because it is not available in the package.
Message	The resource for CLKIN[<index>] is not configured
To fix	The PLL expects to find the resource for the PLL clockin. Add a GPIO block in input mode, set the Connection Type to pll_clkln , and assign it to the correct resource.

pll_rule_resource (error)

Message	Resource name is empty Resource is not a valid PLL device instance
To fix	Choose a valid PLL resource.

pll_rule_vco_freq (error)

Message	VCO frequency is out of range. Freq=<float> Min=<float>MHz Max=<float>MHz
To fix	The VCO frequency needs to be within the range specified. Adjust the parameters to meet that requirement.

Oscillator

Contents:

- [Using the Oscillator Block](#)
- [Design Check: Oscillator Errors](#)

The Titanium FPGA has 1 low-frequency oscillator tailored for low-power operation. The oscillator runs at a nominal frequency of 10, 20, 40, or 80 MHz. You can use the oscillator to perform always-on functions with the lowest power possible. Its output clock is available to the core. You can enable or disable the oscillator to allow power savings when not in use.

Using the Oscillator Block

To use the oscillator block in your design:

1. Add the oscillator block.
2. Select the resource (**OSC_0**).
3. Choose the clock frequency (**10, 20, 40, or 80**).
4. Specify the instance name and clock pin.



Note: You can disable the internal oscillator in Titanium FPGAs. The internal oscillator is disabled if it is not instantiated in the Efinity® Interface Designer.

Design Check: Oscillator Errors

When you check your design, the Interface Designer applies design rules to your configuration settings. The following tables show some of the error messages you may encounter and explains how to fix them.

osc_rule_clock_name (error)

Message	Clock pin name is not specified Valid characters are alphanumeric characters with dash and underscore only
To fix	Specify a valid clock name.

osc_rule_frequency (error)

Message	Invalid frequency <value> Unrecognized frequency <value>
To fix	The oscillator only supports 10, 20, 40, or 80 MHz frequencies, so choose one of those.

osc_rule_instance_count (error)

Message	There can only be one oscillator instance
To fix	Titanium FPGAs only have one oscillator, so you can only use one oscillator block.

osc_rule_inst_name (error)

Message	Instance name is empty Valid characters are alphanumeric characters with dash and underscore only
To fix	Specify a valid instance name.

osc_rule_resource (error)

Message	Resource name is empty Resource is not a valid oscillator device instance
To fix	Although Titanium FPGAs only have one oscillator, you still need to choose the resource when you create an oscillator block.

SPI Flash Interface

Contents:

- [About the SPI Flash Memory](#)
- [Using the SPI Flash Interface](#)

Titanium Ti35 and Ti60 FPGAs in the F100S3F2 package have an integrated SPI flash memory.

About the SPI Flash Memory

Titanium FPGAs in the F100S3F2 package include a SPI flash memory. The SPI flash memory has a density of 16 Mbits and a clock rate of up to 85 MHz. In active configuration mode, the FPGA is configured using the configuration bitstream in the SPI flash memory. Typically you can fit two compressed bitstream images into the F100S3F2 SPI flash.



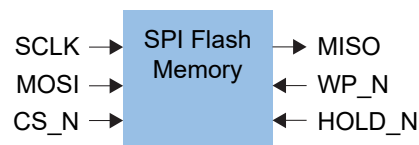
Important: You cannot enable the Titanium FPGA security features when using compressed bitstreams.

To use active programming mode for the Titanium F100S3F2 with the SPI flash memory, you must use the SPI Active using JTAG Bridge configuration mode to configure the SPI flash memory.



Learn more: Refer to the [AN 033: Configuring Titanium FPGAs](#) for information on programming the SPI flash memory.

Figure 58: SPI Flash Memory Block Diagram



Important: The SPI flash memory's VCC is connected to VCCIO1A_4B. If you are using the SPI flash memory, drive the VCCIO1A_4B with a 1.8 V supply.

Table 81: SPI Flash Memory Signals (Interface to FPGA Fabric)

Signal	Direction	Description
SCLK	Input	Clock output to SPI flash memory.
MOSI	Input	Data output to SPI flash memory.
CS_N	Input	Active-low SPI flash memory chip select.
WP_N	Input	Active-low write protect signal.
HOLD_N	Input	Active-low hold signal.
MISO	Output	Data input from SPI flash memory.

Using the SPI Flash Interface

The on-board flash is 16 Mbits and can hold:

- 1 uncompressed bitstream or
- 2 compressed bitstreams (typical designs) or
- 1 compressed bitstream and user data



Note: The maximum bitstream size for Ti35 and Ti60 FPGAs is about 13.7 Mbits; compression typically reduces the size by about 50%. So you have about half of the flash left over for user data if you only store one compressed bitstream.

If you want to use the on-board flash to store user data, you need to add the SPI flash block to your interface design. Simply add the block, choose the resource, and specify the instance and pin names. Then, connect the pins to your user design. Only use the SPI flash block to communicate with the on-board flash in user mode; you do not use this block for external flash devices.



Important: You **do not** need to use the SPI flash block if you are **only** using the on-board flash for storing bitstreams.

Table 82: SPI Flash Properties

Option	Values	SPI Name
Instance Name	User defined	
SPI Flash Resource	SPI_FLASH0	
Clock Output to Flash Pin Name	User defined	SCLK
Data Input from Flash Pin Name	User defined	MISO
Data Output to Flash Pin Name	User defined	MOSI
Flash Chip Select (Active-Low) Pin Name	User defined	CS_N
Hold (Active-Low) Pin Name	User defined	HOLD_N
Write Protect (Active-Low) Pin Name	User defined	WP_N

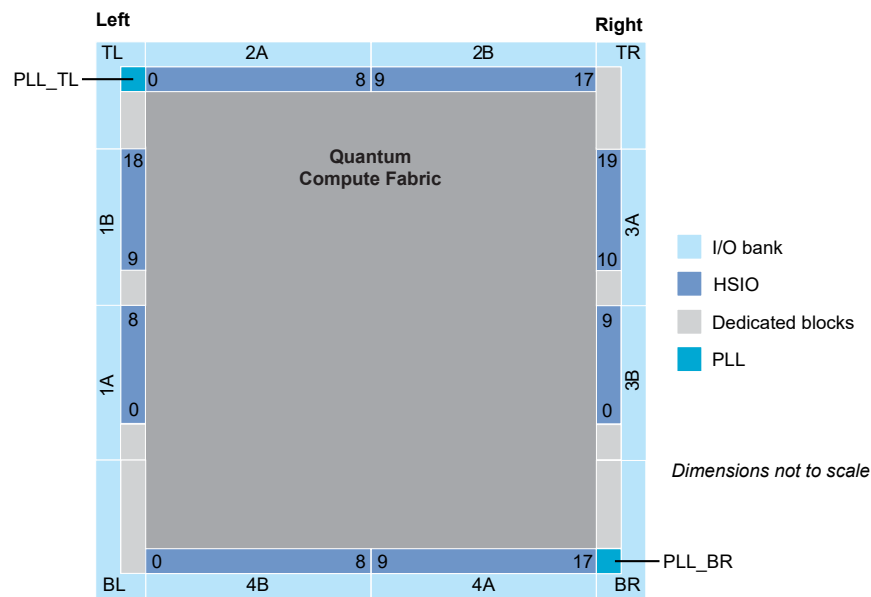
Interface Floorplans



Note: The numbers in the floorplan figures indicate the HVIO and HSIO number ranges. Some packages may not have all HVIO or HSIO pins in the range bonded out. Refer to the pinout for information on which pins are available in each package.

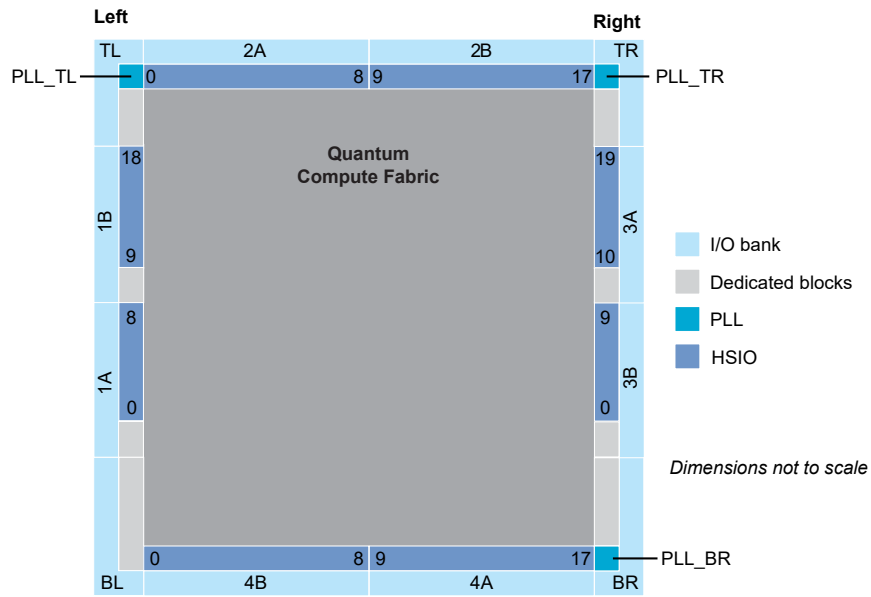
Floorplan Diagram for FPGAs in W64 Packages

Figure 59: Ti60 FPGAs



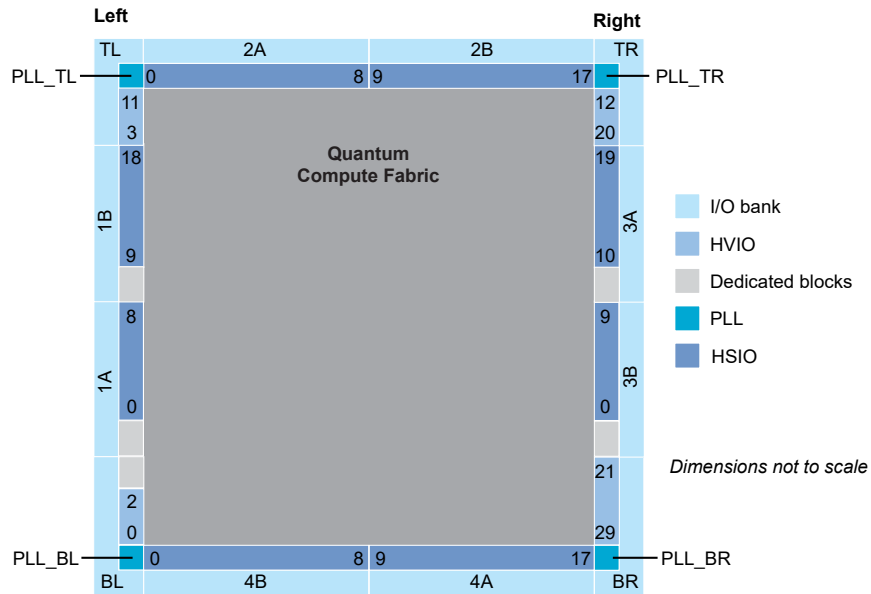
Floorplan Diagram for FPGAs in F100S3F2 Packages

Figure 60: Ti35 and Ti60 FPGAs



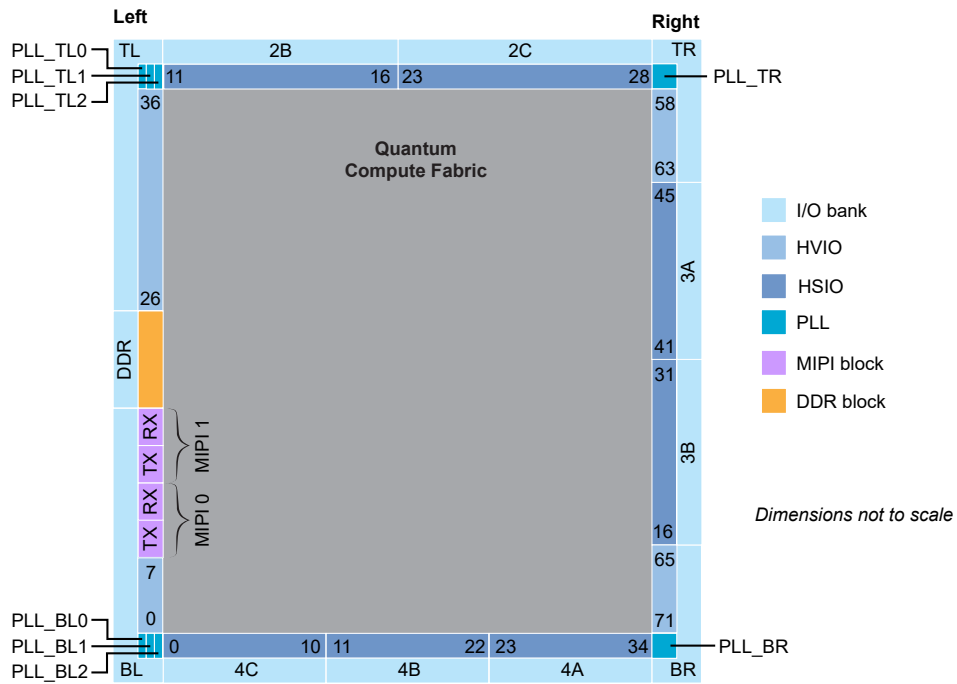
Floorplan Diagram for FPGAs in F225 Packages

Figure 61: Ti35 and Ti60 FPGAs



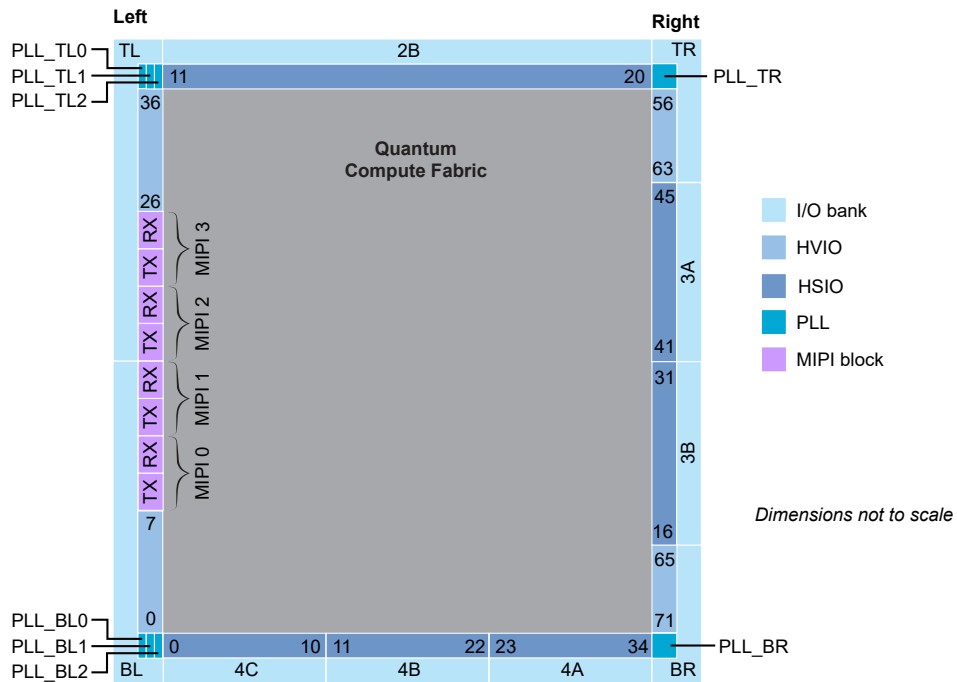
Floorplan Diagram for FPGAs in M361 and J361 Packages

Figure 62: Ti90, Ti120, and Ti180 FPGAs



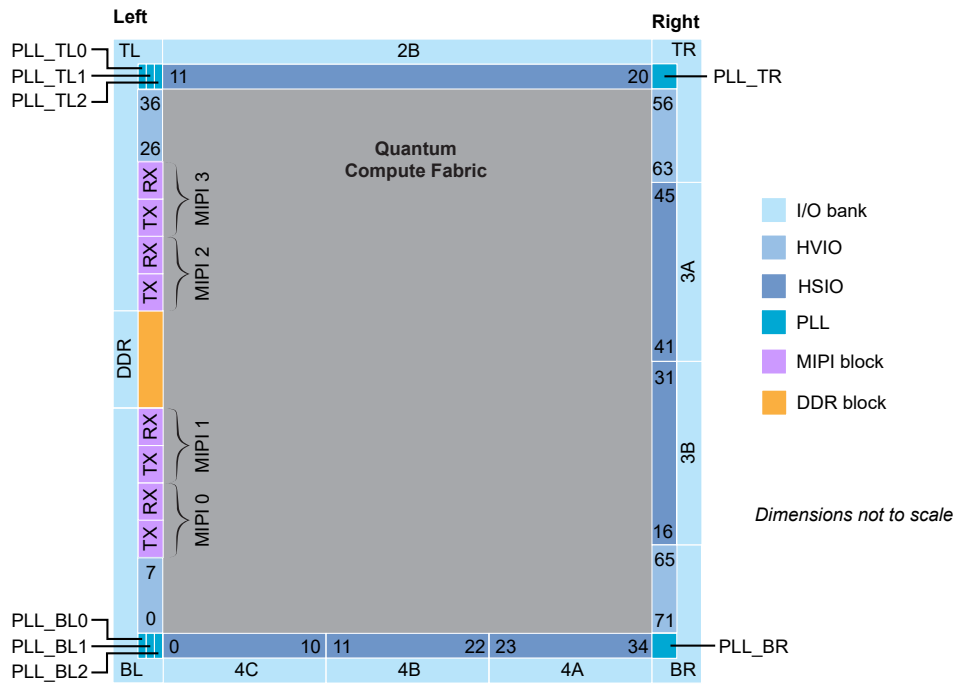
Floorplan Diagram for FPGAs in L484 Packages

Figure 63: Ti90, Ti120, and Ti180 FPGAs



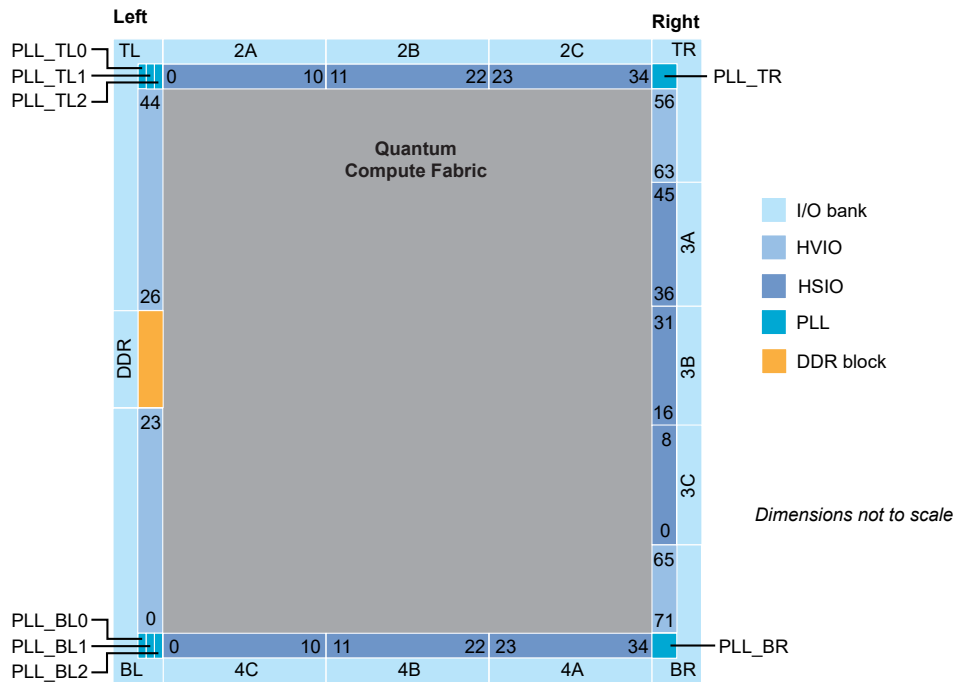
Floorplan Diagram for FPGAs in J484 and M484 Packages

Figure 64: Ti90, Ti120, and Ti180 FPGAs




















Floorplan Diagram for FPGAs in F529 and G529 Packages

Figure 65: Ti90, Ti120, and Ti180 FPGAs



Icon Reference

Interface Designer Icons

	Interface Designer		Export GPIO Assignments		Resource Assigner
	Add Block		Import GPIO Assignments		Toggle Instance View and Resource View
	Create a GPIO bus		Check Design for Errors		Clear Resource
	Delete Block		Export Settings		Clear All Resources
	Show or Hide Block Editor		Generate Constraints File		Show/Hide Filter
			View Report		Clear Filter

Revision History

Table 83: Revision History

Date	Version	Description
February 2023	2.7	Corrected PLL_SSC_EN MIPI TX D-PHY signal notes. (DOC-1101)
December 2022	2.6	Updated support for J361, J484, and G529 packages. Updated MIPI DPHY TX and DDR Interface Designer Settings.
October 2022	2.5	Updated DDR DRAM interface signals. Updated DDR DRAM Interface Designer Settings.
September 2022	2.4	Updated PLL clock for DDR DRAM block. (DOC-881) Corrected MIPI RX Lane Block Diagram. (DOC-878) Removed GCTRL and RCTRL. (DOC-895) Added topics on Package Planner. Corrected AWID_x, AWREADY_x, ARADDR_x, and AWADDR_x DDR signals directions and widths. (DOC-907) Removed PLL_EXTFB from alternative input. (DOC-849)
July 2022	2.3	Corrected floorplan diagrams.
July 2022	2.2	Corrected the LVDS maximum speed. (DOC-807) Removed reference to T13 and T20. (DOC-807) Updated MIPI D-PHY port names. (DOC-782) Added I/O banks by package information for Ti90, Ti120, and Ti180. (DOC-821) Updated DDR pad names. Added M361, L484, M484, and F529 floorplans.
April 2022	2.1	Corrected RD and RST signal directions in MIPI RX Lane Block Diagram. Corrected description for differential TX static programmable delay. (DOC-786) Updated HyperRAM clock rate and double data rate specs. (DOC-793) Corrected missing link and added pointer for list of clock sources in Global Buffer Configuration table.
February 2022	2.0	Added Titanium DDR block interface description. Added Titanium MIPI DPHY interface block description. HVIO I/O banks support dynamic voltage shifting. (DOC-444) Added MIPI RX clock groups for F484 package. Added interface floorplan for F484 package. New design rules: io_bank_rule_mode_sel, io_bank_rule_dyn_voltage. Updated label for Ti60 W64 pin A7. (DOC-651)

Date	Version	Description
November 2021	1.1	<p>Updated PLL Block Diagram to indicate F_{PLL}.</p> <p>Updated JTAG mode connection diagram. (DOC-546)</p> <p>Updated PLL phase-shift descriptions. (DOC-570)</p> <p>PLL outputs lock on the negative clock edge. (DOC-552)</p> <p>Added example PLL zero-delay buffer implementation. (DOC-551)</p> <p>Added an example for the PLL outputs for the Create a MIPI TX Interface topic. (DOC-580)</p> <p>New design rule: <code>clock_rule_lvds_rx_clock_source</code>. This rule is effective with Efinity patch v2021.1.4.10. (DOC-589)</p> <p>New design rules: <code>clock_rule_pll_ref_clock_lvds_rx</code>, <code>pll_rule_pll_freq</code>, <code>lvds_rule_tx_clock_region</code>, <code>lvds_rule_rx_clock_region</code>, <code>lvds_rule_rx_dpa_serial</code>, and <code>mipi_ln_rule_tx_clock_region</code>.</p>
June 2021	1.0	Initial release for Efinity software v2021.1.