



Trion[®] Interfaces User Guide

UG-TINTF-v8.7
November 2024
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
Clocking Interface Blocks.....	10
Design Check: Clock Messages.....	12
Designing an Interface.....	13
Create or Delete a Block.....	13
Using the Resource Assigner.....	14
Resource View.....	15
Importing and Exporting Assignments.....	15
Viewing the Package Pinout.....	17
Selecting a Pin.....	19
Browsing for Pins.....	20
Interface Designer Output Files.....	20
Scripting an Interface Design.....	21
Chapter 2: Device Settings.....	22
Configuration Interface.....	22
Enable Internal Reconfiguration.....	22
Design Check: Configuration Messages.....	23
I/O Banks Interface.....	24
I/O Banks.....	24
Trion I/O Banks.....	24
Design Check: I/O Bank Messages.....	26
Chapter 3: DDR Interface.....	27
About the DDR DRAM Interface.....	27
DDR Interface Designer Settings.....	33
Using the DDR Block.....	37
Design Check: DDR Messages.....	38
Chapter 4: GPIO Interface.....	41
About the General-Purpose I/O Logic and Buffer.....	41
Simple I/O Buffer.....	44
Complex I/O Buffer.....	45
Double-Data I/O.....	46
Using the GPIO Block.....	48
Using LVDS as GPIO.....	51
Using the GPIO Bus Block.....	52
Design Check: GPIO Messages.....	52
Chapter 5: JTAG User TAP Interface.....	59
JTAG Mode.....	59
Using the JTAG User TAP Block.....	60
Design Check: JTAG User Tap Messages.....	60
Chapter 6: LVDS Interface.....	62
About the LVDS Interface.....	62
LVDS TX.....	63
LVDS RX.....	65
Using the LVDS Block.....	66
Create an LVDS TX or RX Interface.....	68
Create an LVDS TX Interface.....	68
Create an LVDS RX Interface.....	70
Design Check: LVDS Messages.....	71

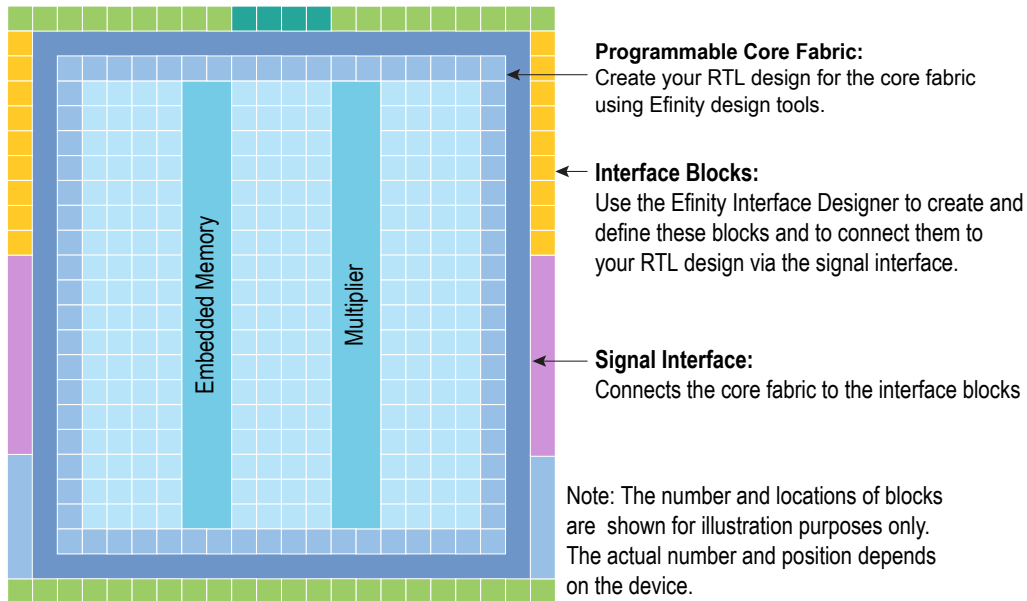
Chapter 7: MIPI CSI-2 Interface.....	77
About the MIPI Interface.....	77
MIPI TX.....	78
MIPI RX.....	85
D-PHY Timing Parameters.....	92
Understanding the RX and TX Pixel Clock.....	93
Power Up Sequence.....	94
Using the MIPI Block.....	95
Design Check: MIPI Messages.....	96
Chapter 8: PLL Interface.....	99
About the Simple PLL Interface.....	99
Using the PLL Block.....	100
Using the PLL Clock Calculator.....	101
Set up the PLL Manually.....	101
Design Check: Simple PLL Messages.....	101
Chapter 9: Advanced PLL Interface.....	104
About the Advanced PLL Interface.....	104
Using the PLL Block.....	108
Using the PLL Clock Calculator.....	108
Understanding PLL Phase Shifting.....	109
Configuring the PLL Manually.....	110
Output Clock Swapping.....	110
Design Check: Advanced PLL Messages.....	110
Chapter 10: Oscillator Interface.....	115
Oscillator.....	115
Using the Oscillator Block.....	115
Design Check: Oscillator Messages.....	115
Chapter 11: SPI Flash Interface.....	117
About the SPI Flash Memory.....	117
Using the SPI Flash Interface.....	118
Design Check: SPI Flash Messages.....	119
Chapter 12: Interface Floorplans.....	121
Icon Reference.....	131
Revision History.....	132

About the Interface Designer

Trion® 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

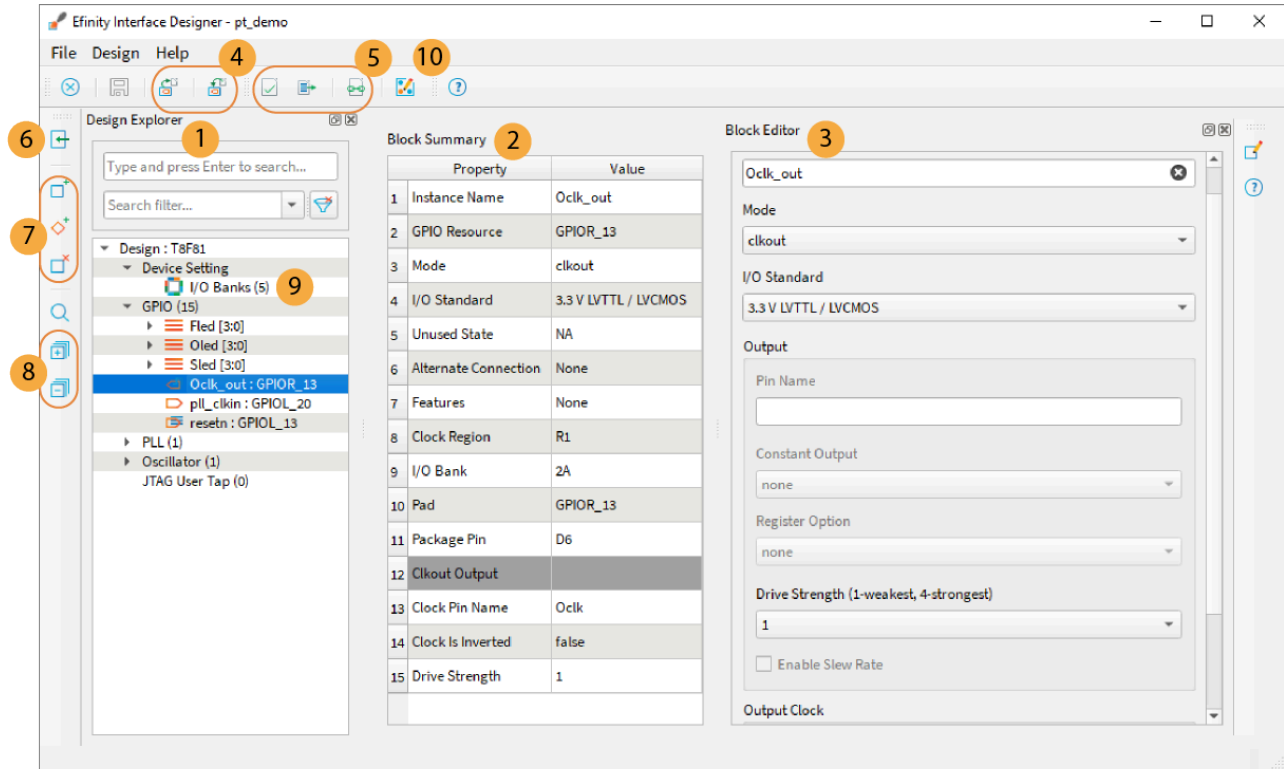
Contents:

- **Interface Blocks**
 - **Package/Interface Support Matrix**
 - **Interface Block Connectivity**
 - **Clocking Interface Blocks**
 - **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

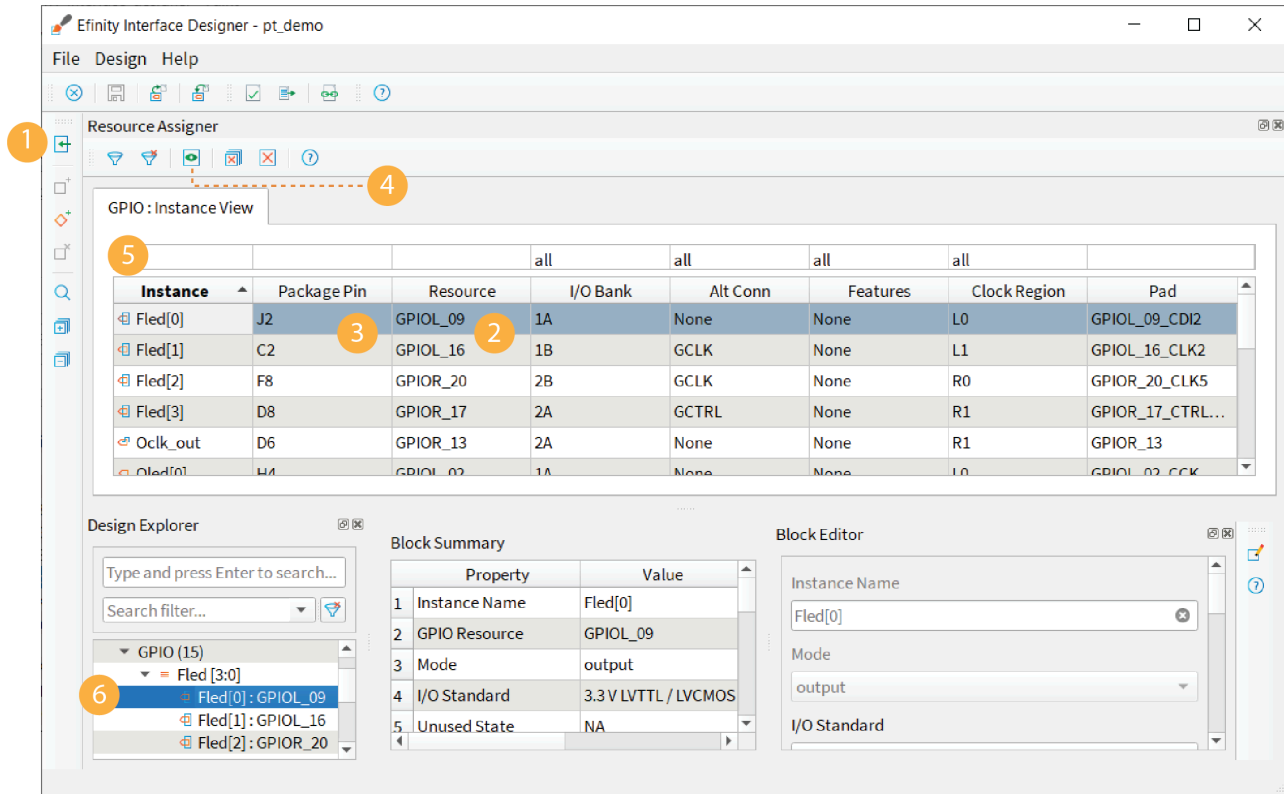


Notes:

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.
10. The Package Planner lets you see the pins and assignments graphically.

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

Trion® 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.



Note: New package support is often added in patches. Refer to the Efinity Release Notes in the [Support Center](#) for the latest patch support.

Table 1: Trion Interface Block Support by Package

Interface	T4	T8	T13	T20	T35	T55	T85	T120
GPIO	All	All	All	All	All	All	All	All
GPIO bus	All	All	All	All	All	All	All	All
I/O bank	All	All	All	All	All	All	All	All
JTAG User TAP ⁽¹⁾	F81	F81, Q144	All	All	All	All	All	All
LVDS	-	Q144	All	Q100, Q144, F169, F256, F324, F400	All	All	All	All
MIPI	-	-	F169	W80, F169, F324	F324	All	F324, F576	F324, F576
DDR	-	-	-	F324, F400	F324, F400	All	All	All
Simple PLL (V1)	All	F49, F81	-	-	-	-	-	-
Advanced PLL (V2)	-	Q144	All	All	All	All	All	All
Oscillator	All	F49, F81	-	-	-	-	-	-
SPI Flash	-	-	Q100	Q100	-	-	-	-

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.

⁽¹⁾ The T4 and T8 in the F49 package do not support the JTAG User TAP because this package does not have the dedicated JTAG pins (TDI, TDO, TCK, TMS).

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. Refer to the data sheet for package-dependent resources.





























 **Note:** New package support is often added in patches. Refer to the Efinity Release Notes in the **Support Center** for the latest patch support.

Table 2: Supported Trion Interface/Package Combinations

Package	T4	T8	T13	T20	T35	T55	T85	T120
F49								
W80								
F81								
Q100								
Q144								
F169								
F256								
F324								
F400								
F484								
F576								

Trion Family Legend:

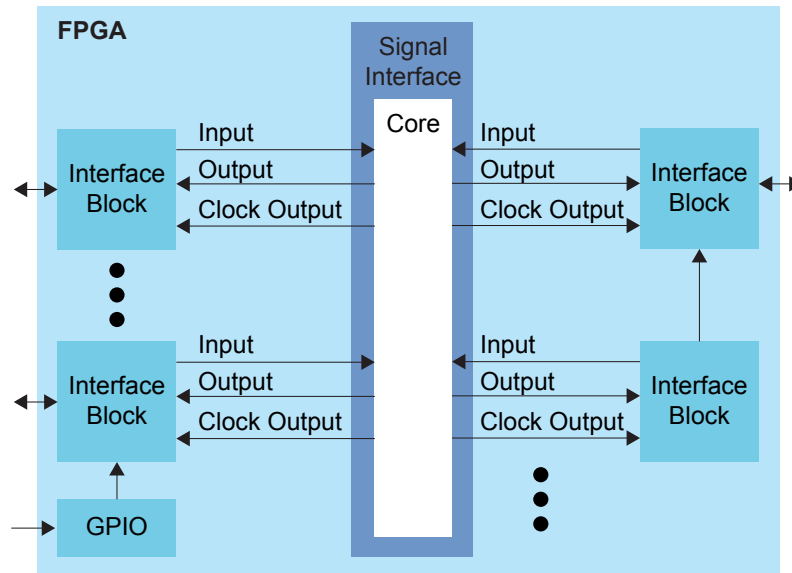
-  Oscillator
-  PLL
-  LVDS
-  MIPI CSI-2 Controller
-  DDR DRAM Controller
-  SPI Flash

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 Trion® 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. Signals and block diagrams are shown from the perspective of the interface, not the core.

Clocking Interface Blocks

The Interface Designer defines clock connections to the interfaces in the **<project name>.interface.csv** file, which allows the software to connect the clocks between the core

and periphery. To use a clock in the periphery, simply use the clock name in the relevant field in the interface block.

- Clocks used in interfaces do not have to be used in the RTL.
- Clocks feeding the interfaces should not drive outputs in the RTL.



Note: This section assumes you understand how to create and use the GPIO and PLL blocks.

(See [GPIO Interface](#) on page 41 and [PLL Interface](#)).

The following sections show some common clocking examples to illustrate the general concept. There are other Interface Designer blocks that use clocks, and other modes for the GPIO and PLL blocks. You use the same method for referencing clocks in other blocks and modes.

GPIO Clocking I/O Register

To use a GPIO to clock an I/O register:

1. Instantiate a GPIO in input mode with connection type **gelk** or **rclk**. Put the name in **Input Clock tab > Pin Name**, e.g., `gpio_inst12`. This name is the clock name.
2. Instantiate a GPIO in output mode with **Output tab > Register Option > register**.
3. In **Output Clock tab > Pin Name**, enter the clock name from step 1 (`gpio_inst12`).

If this clock is not used in your RTL design, you do not have to include it. It's simply a clock from one interface block to another.

PLL Output Clocking I/O Register

To use a PLL output clock to clock an I/O register:

1. Instantiate a PLL and define an output clock, `p11_out1`. This name is the clock name.
2. Instantiate a GPIO in output mode with **Output tab > Register Option > register**.
3. In **Output Clock tab > Pin Name**, enter the clock name from step 1 (`p11_out1`).

If this clock is not used in your RTL design, you do not have to include it. It is simply a clock from one interface block to another.

Internal Clock Clocking I/O Register

To use an internally generated clock to clock an I/O register:

1. You create an internally generated clock, `div_clock`. This name is the clock name.
2. Instantiate a GPIO in output mode with **Output tab > Register Option > register**.
3. In **Output Clock tab > Pin Name**, enter the clock name from step 1 (`div_clock`).

You do not need to connect the internally generate clock to an output in the RTL. The Interface Designer connects to the clock network automatically.

PLL Output Driving a Pin

To use a PLL output clock to clock a pin:

1. Instantiate a PLL and define an output clock, `p11_out1`. This name is the clock name.
2. Instantiate a GPIO in **clkout** mode.
3. In **Output Clock tab > Pin Name**, enter the clock name from step 1 (`p11_out1`).

If this clock is not used in your RTL design, you do not have to include it. It is simply a clock from one interface block to another.

You can use this output pin to connect to other components on your board.

Design Check: Clock 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.

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 are using more clocks than are available. You need to remove some clocks on the left/right and top/bottom clock regions.
Message	Cannot connect to more than <int> different clocks per region (40 rows) on left and right
To fix	You are using more clocks than are available. You need to remove some clocks on the left/right clock regions.
Message	Cannot connect to more than <int> different clocks on top and bottom
To fix	You are using more clocks than are available. You need to remove some clocks on the top/bottom clock regions.

clock_rule_max_count (error)

Message	Number of core clock used exceeds max limit of <int>
To fix	Your design has too many clocks (GPIO configured in clkout mode) coming from the core. You need to remove some clocks

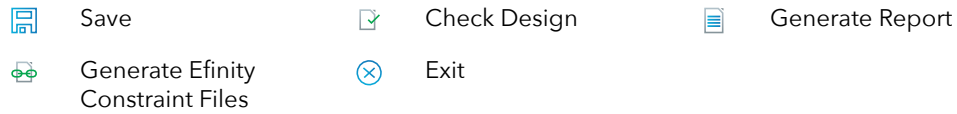
clock_rule_invalid_name (error)

Message	Invalid clock name
To fix	Enter a valid clock name.

clock_rule_undefined_name (info)

Message	Clock <clock name> not defined in the interface, assuming core generated.
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 info message indicates that you have not defined it as an interface block. If the clock is generated in the core you can ignore this message.

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 13 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

-  Resource Assigner
  Switch View
  Clear Selected Resource
  Clear All Resources
 Show/Hide Filter
  Reset Filter

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.



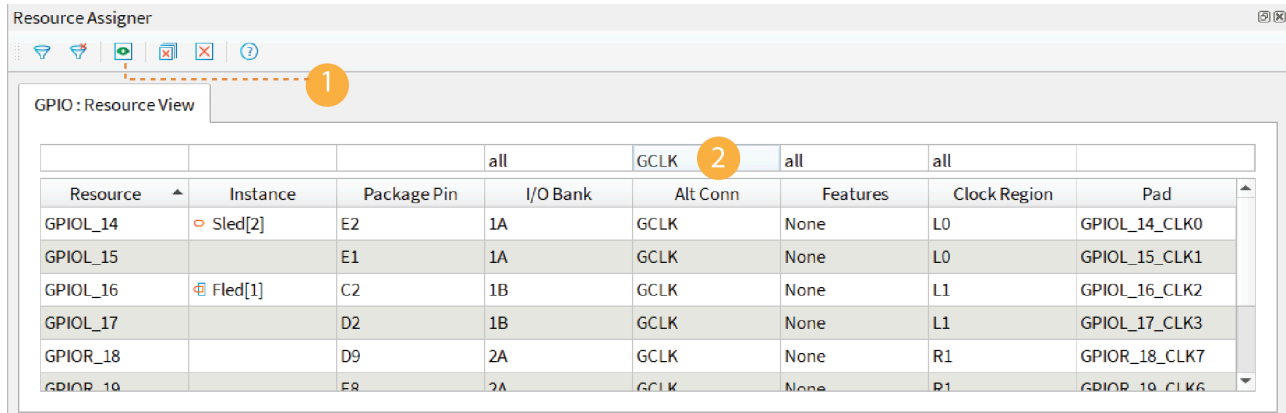
Note: When LVDS resources are used for both LVDS and GPIO within the same bank, they must be separated by 2 unused pairs of LVDS pins to avoid any unwanted interference. The Efinity software issues an error if you do not leave this separation. Refer to [Table 34: LVDS Resources Assignment Example with LVDS and GPIO Signals in Same Bank](#) on page 51.

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

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		
...		
(2)	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

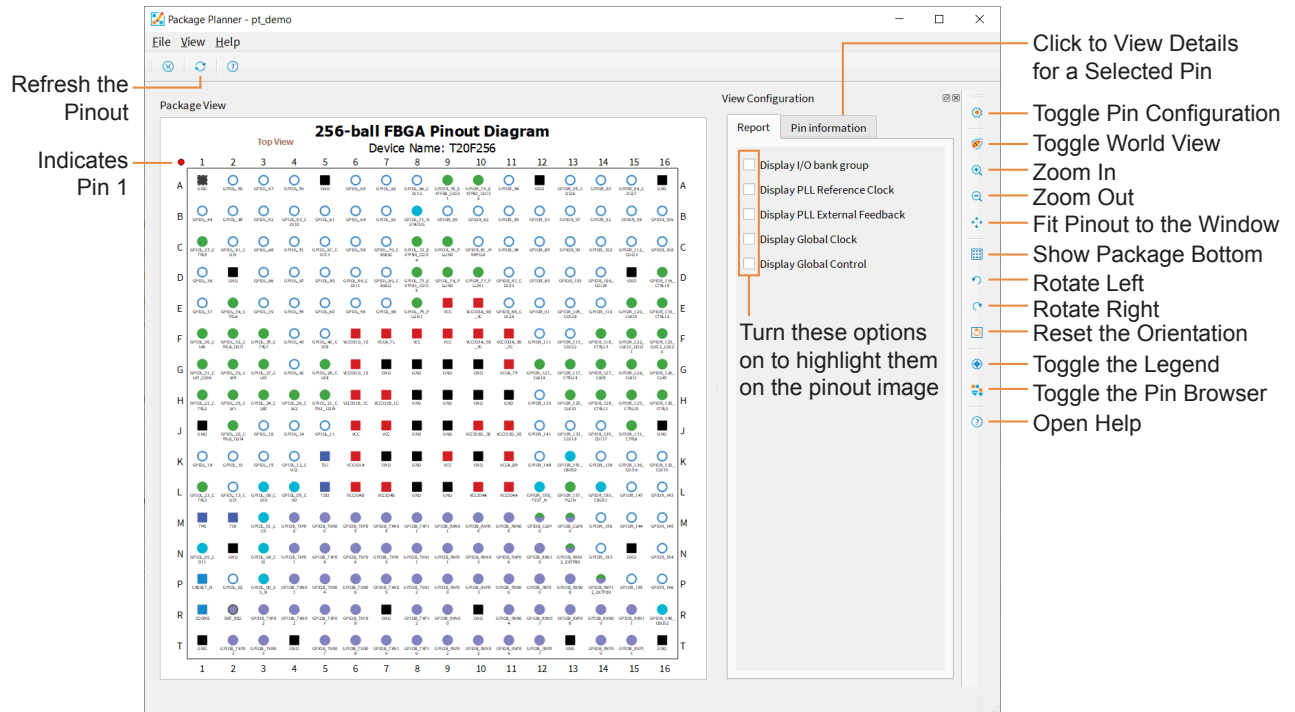
Viewing the Package Pinout

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

⁽²⁾ Unassigned instances have a blank field for the Package Pin-Pad Name column.

has the feature you need. This tool is helpful when planning how to map the signals in your design to package pins.

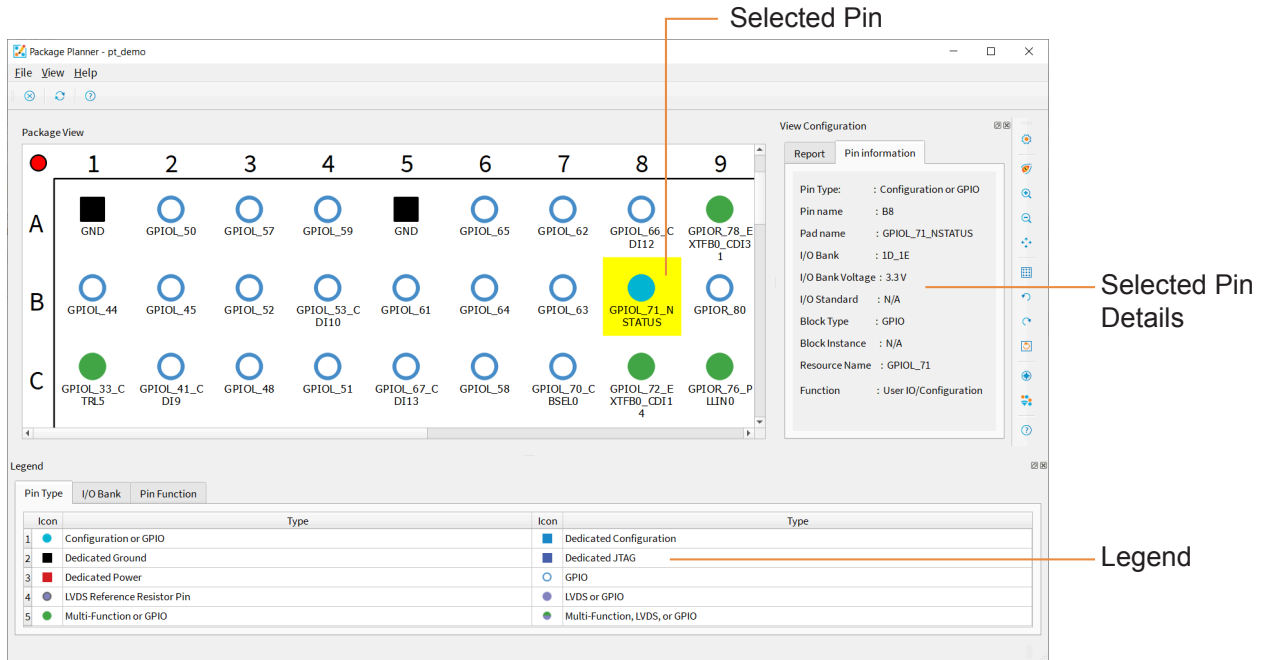
Figure 6: Package Planner



Selecting a Pin

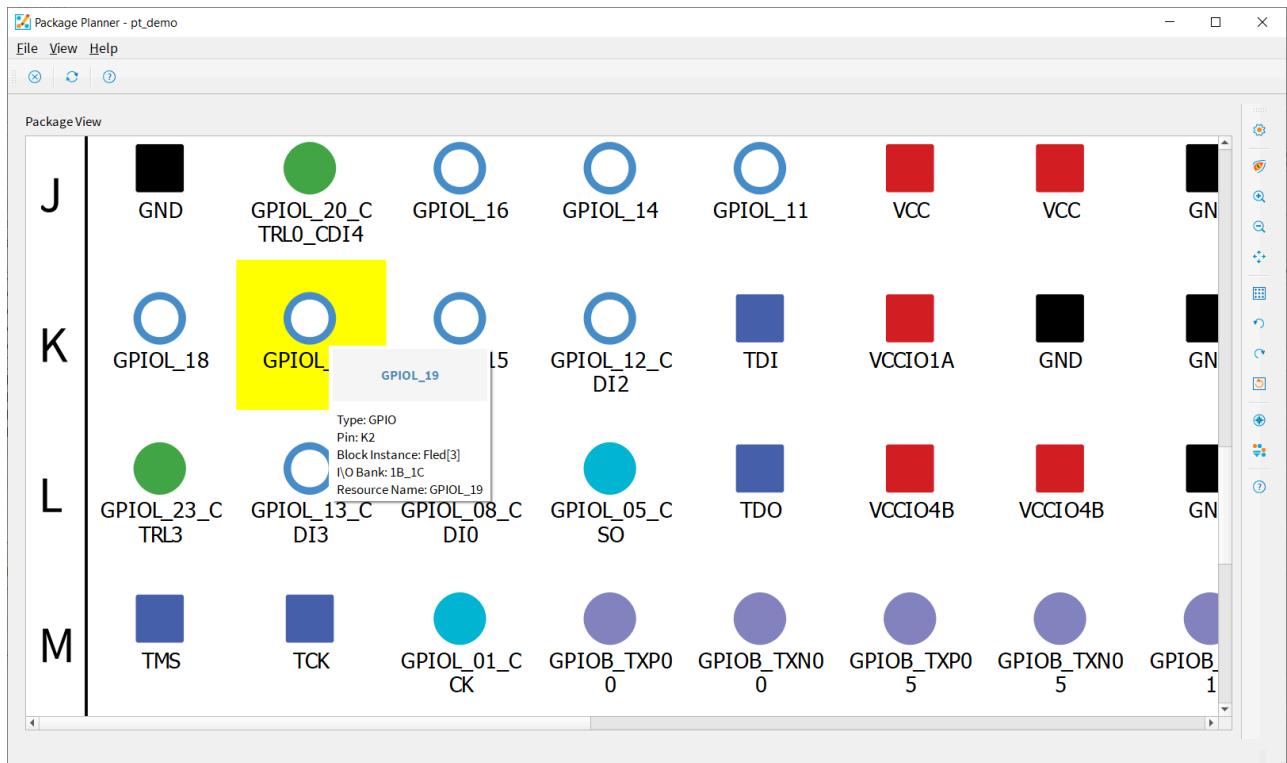
Click a pin in the pinout to highlight it. The **Pin Information** tab opens to show the details about the selected pin. Open the Legend to view the meaning of the pins' color coding.

Figure 7: Selected Pin



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

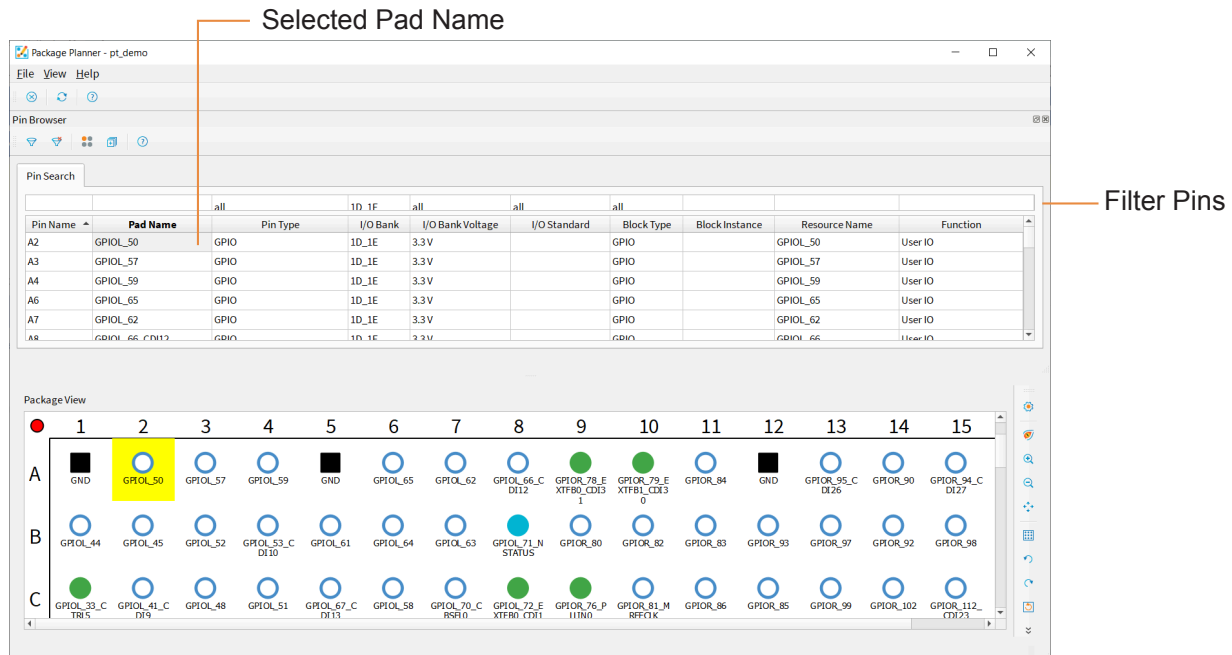
Figure 8: Pin Quick View



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 Trion® 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.⁽³⁾ 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](#).

⁽³⁾ Source: [What Is Python? Executive Summary](#)

Device Settings

Contents:

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

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



Note: Configuration device-wide settings are not available for T4 and T8 FPGAs.

Configuration Interface

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

Enable Internal Reconfiguration

Efinix® FPGAs (except the T4 and T8) 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, 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.

Table 4: Remote Update Tab Settings

Parameter	Choices	Notes
Enable Internal Reconfiguratio Interface	On, off	Default: off.
Clock Pin Name	User defined	Specify the clock pin name used to latch cfg_CBSEL when cfg_ENA is high.
Invert Clock	On, off	Default: off. Turn o to invert the clock pin.
Image Selector [1:0] Bus Name	User defined	Multi-image select signals to the internal reconfiguration interface (not package pins). Use these signals to choose which image to load from flash memory. Efinix recommends using the default name.

Parameter	Choices	Notes
Image Selector Capture Pin Name	User defined	When cfg_ENA is high, read the value of cfg_CBSEL. Efinix recommends using the default name.
Configuration Control Pin Name	User defined	Asynchronous control that initiates reconfiguration. Efinix recommends using the default name.
Error Status Pin Name	User defined	Status signal. Signal is set to 0 during power-up. Efinix recommends using the default name.
Remote Update Retries	0-7	Indicate how many times the FPGA should attempt to perform the remote update.

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 and warning 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/empty>
To fix	Enter a valid clock name.

configuration_rule_bus_name (error)

Message	Internal Reconfiguration Interface is enabled but bus name is <invalid/empty>
To fix	Enter a valid bus name.

configuration_rule_empty_pins (info)

Message	Empty pin names found: <Pin description names>
To fix	Specify valid pin names.

configuration_rule_invalid_pins (error)

Message	Invalid pin names found: <pin description names>
To fix	Enter a valid pin name.

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.

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.



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

Trion I/O Banks

Table 5: I/O Banks by FPGA and Package

Package	I/O Banks	Voltage (V)	DDIO Support	Merged Banks
T4				
F49, F81	1A - 1C, 2A, 2B	1.8, 2.5, 3.3	-	-
T8				
F49, F81	1A - 1C, 2A, 2B	1.8, 2.5, 3.3	-	-
Q144	1A - 1E, 3A - 3E	1.8, 2.5, 3.3	1B, 1C, 1D, 3B, 3C, 3D, 3E	1C_1D, 3B_3C
	4A, 4B	3.3	-	-
T13				
Q100F3	1A - 1E, 3A - 3E	1.8, 2.5, 3.3	1B, 1C, 1D, 3B, 3C, 3D, 3E	1A_1B_1C, 1D_1E, 3B_3C, 3D_3E
	4A, 4B	3.3	-	-
F169	1A - 1E, 3A - 3E	1.8, 2.5, 3.3	1B, 1C, 1D, 3B, 3C, 3D, 3E	1B_1C_1D, 3A_3B, 3C_3D_3E
	4A, 4B	3.3	-	-
F256	1A - 1E, 3A - 3E	1.8, 2.5, 3.3	1B, 1C, 1D, 3B, 3C, 3D, 3E	1B_1C, 1D_1E, 3A_3B_3C, 3D_3E

Package	I/O Banks	Voltage (V)	DDIO Support	Merged Banks
	4A, 4B	3.3	-	-
T20				
WP80	1A-1E, 3A-3E	1.8, 2.5, 3.3	1B, 1D, 3C, 3D, 3E	1B_1C_1D_1E, 3A_3B_3C, 3D_3E_4A_4B
Q100F3	1A - 1E, 3A - 3E	1.8, 2.5, 3.3	1B, 1C, 1D, 3B, 3C, 3D, 3E	1A_1B_1C, 1D_1E, 3B_3C, 3D_3E
	4A, 4B	3.3	-	-
Q144	1A - 1E, 3A - 3E	1.8, 2.5, 3.3	1B, 1C, 1D, 3B, 3C, 3D, 3E	1C_1D, 3B_3C
	4A, 4B	3.3	-	-
F169	1A - 1E, 3A - 3E	1.8, 2.5, 3.3	1B, 1C, 1D, 3B, 3C, 3D, 3E	1B_1C_1D, 3A_3B, 3C_3D_3E
	4A, 4B	3.3	-	-
F256	1A - 1E, 3A - 3E	1.8, 2.5, 3.3	1B, 1C, 1D, 3B, 3C, 3D, 3E	1B_1C, 1D_1E. 3A_3B_3C, 3D_3E
	4A, 4B	3.3	-	-
F324	1A - 1E, 2A - 2C, 3A - 3C, 4A, 4B, TR, BR	1.8, 2.5, 3.3	1A - 1E, 3C, TR, BR	1B_1C, 1D_1E, 3C_TR_BR
F400	1A - 1E, 2A - 2C, 3C, 4A, 4B, TR, BR	1.8, 2.5, 3.3	1A - 1E, 3C, TR, BR	3C_TR
T35				
F256	1A - 1E, 2A, 2B, 3C, 4A, 4B, TR, BR	1.8, 2.5, 3.3	1A - 1E, 3C, TR, BR	1A_1B, 2A_2B_2C, 3C_TR_BR
F324	1A - 1E, 2A - 2C, 3A - 3C, 4A, 4B, TR, BR	1.8, 2.5, 3.3	1A - 1E, 3C, TR, BR	1B_1C, 1D_1E, 3C_TR_BR
F400	1A - 1E, 2A - 2C, 3C, 4A, 4B, TR, BR	1.8, 2.5, 3.3	1A - 1E, 3C, TR, BR	3C_TR
T55, T85, T120				
F324	1A - 1G, 2D - 2F, 3D, TR, BR, 4E - 4F	1.8, 2.5, 3.3	Banks 1A-1G, 3D, TR, BR	1B_1C, 1D_1E_1F_1G, 3D_TR_BR
F484	1A - 1G, 2A - 2F, 3D, TR, BR, 4A - 4F	1.8, 2.5, 3.3	Banks 1A-1G, 3D, TR, BR	1B_1C, 1D_1E, 1F_1G, 3D_TR_BR
F576	1A - 1G, 2A - 2F, 3D, TR, BR, 4A - 4F	1.8, 2.5, 3.3	Banks 1A-1G, 3D, TR, BR	1B_1C, 1D_1E_1F_1G, 3D_TR_BR

Some I/O banks are merged at the package level by sharing VCCIO pins. Merged banks have underscores () between banks in the name (e.g., 1B_1C means 1B and 1C are connected).



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

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_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.

`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 if you are using pins in that bank for LVDS.

DDR Interface

Contents:

- [About the DDR DRAM Interface](#)
- [DDR Interface Designer Settings](#)
- [Using the DDR Block](#)
- [Design Check: DDR Messages](#)

Some Trion 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

The Trion DDR PHY interface supports DDR3, DDR3L, LPDDR3, LPDDR2 memories with x16 or x32 DQ widths (depending on the FPGA) and a memory controller hard IP block. The DDR PHY supports data rates up to 1066 Mbps per lane. The memory controller provides two AXI 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.

Table 6: DDR DRAM Performance

DDR DRAM Interface	Voltage (V)	Maximum Data Rate (Mbps) per Lane
DDR3	1.5	1066
DDR3L	1.35	1066
LPDDR3	1.2	1066
LPDDR2	1.2	1066

Figure 10: (T20, T35) DDR DRAM Block Diagram

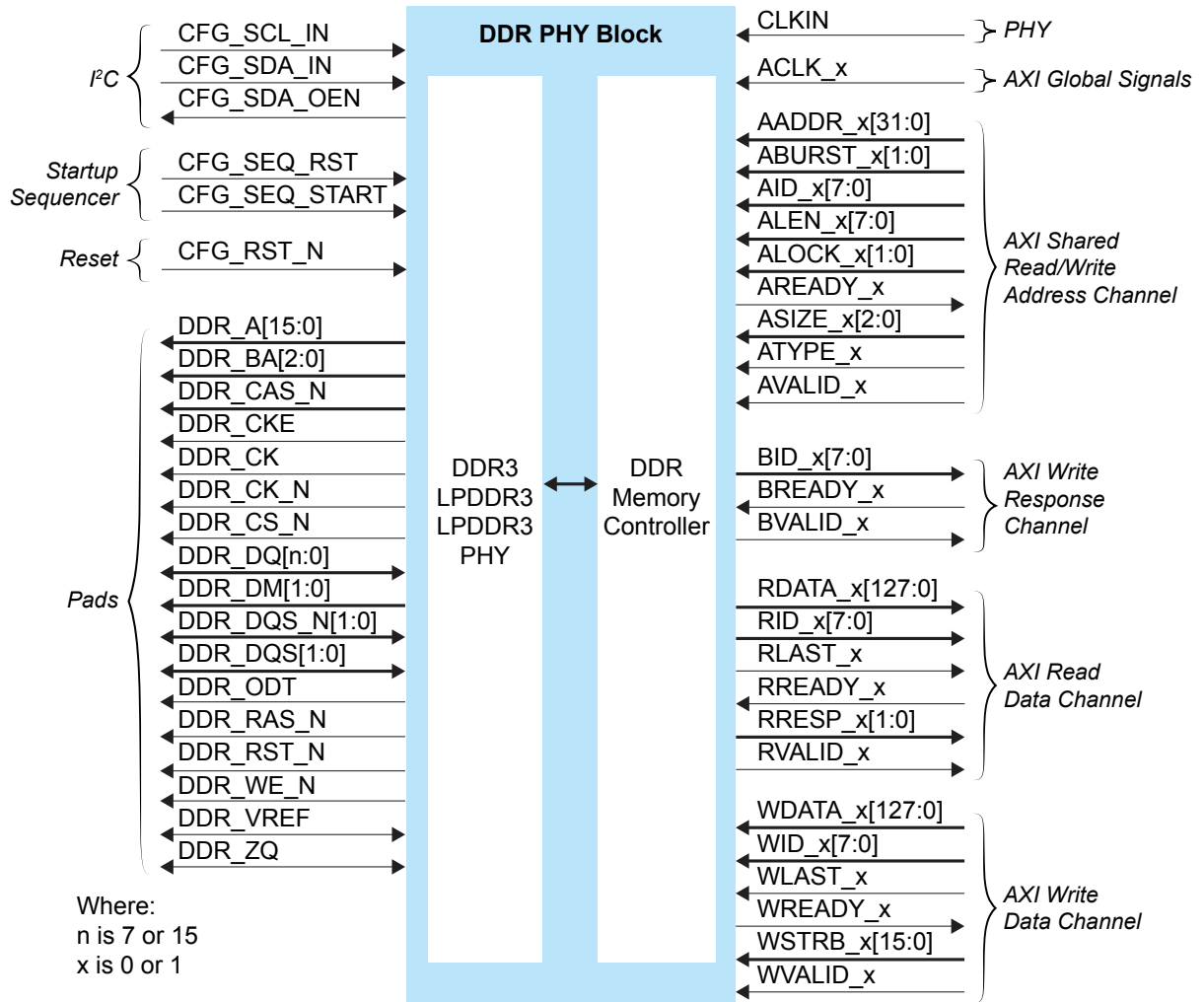
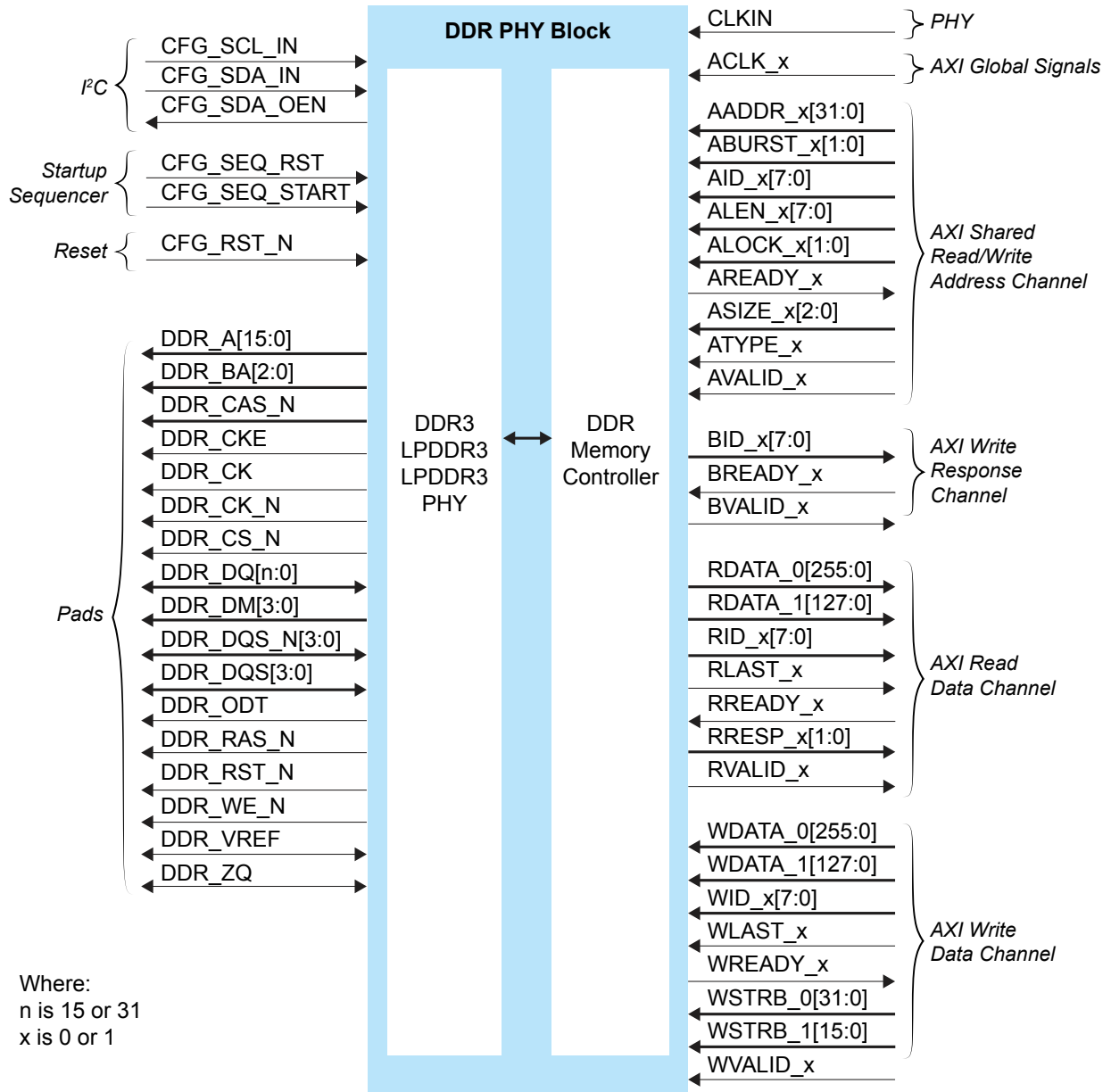


Figure 11: (T55, T85, T120) DDR DRAM Block Diagram



The DDR DRAM block supports an I²C calibration bus that can read/write the DDR configuration registers. You can use this bus to fine tune the DDR PHY for high performance.

Figure 12: DDR DRAM Interface Block Diagram

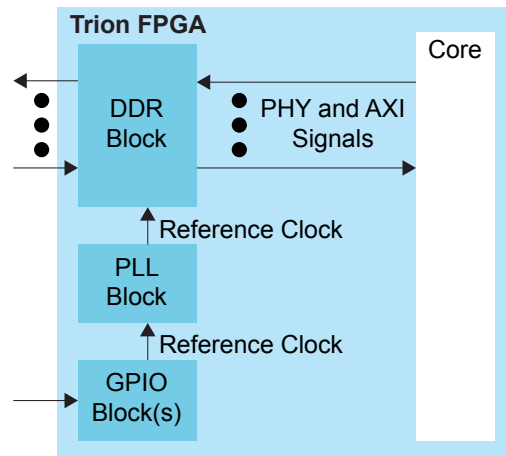


Table 7: PHY Signals (Interface to FPGA Fabric)

Signal	Direction	Clock Domain	Description
CLKIN	Input	N/A	High-speed clock to drive the DDR PHY. A PLL must generate this clock. The clock runs at half of the PHY data rate (for example, 800 Mbps requires a 400 MHz clock). The DDR DRAM block uses the PLL_BR0 CLKOUT0 resource as the PHY clock.

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 Advanced PLL Interface](#) on page 104 for more information about the PLL block.



Important: Efinix strongly recommends that you do not use any LVDS pins (either single-ended I/O or differential pair) as the primary clock to drive the PLL_BR0 or the DDR interface will not be initialized during the configuration phase. Make sure to incorporate a user reset and instantiate the DDR Hard Memory Controller-Reset IP to initialize the DDR interface in user mode. Contact Efinix support if you need LVDS pins as the primary clock for the PLL_BR0 DDR interface

Table 8: AXI Global Signals (Interface to FPGA Fabric)

Signal	Direction	Clock Domain	Description
ACLK_0, ACLK_1	Input	N/A	AXI clock inputs.

Table 9: AXI Shared Read/Write Signals (Interface to FPGA Fabric)

Signal x is 0 or 1	Direction	Clock Domain	Description
AADDR_x[31:0]	Input	ACLK_x	Address. ATYPE defines whether it is a read or write address. It gives the address of the first transfer in a burst transaction.
ABURST_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.
AID_x[7:0]	Input	ACLK_x	Address ID. This signal identifies the group of address signals. Depends on ATYPE, the ID can be for a read or write address group.
ALEN_x[7:0]	Input	ACLK_x	Burst length. This signal indicates the number of transfers in a burst.

Signal x is 0 or 1	Direction	Clock Domain	Description
ALOCK_x[1:0]	Input	ACLK_x	Lock type. This signal provides additional information about the atomic characteristics of the transfer.
AREADY_x	Output	ACLK_x	Address ready. This signal indicates that the slave is ready to accept an address and associated control signals.
ASIZE_x[2:0]	Input	ACLK_x	Burst size. This signal indicates the size of each transfer in the burst.
ATYPE_x	Input	ACLK_x	This signal distinguishes whether it is a read or write operation. 0 = read and 1 = write.
AVALID_x	Input	ACLK_x	Address valid. This signal indicates that the channel is signaling valid address and control information.

Table 10: AXI Write Response Channel Signals (Interface to FPGA Fabric)

Signal x is 0 or 1	Direction	Clock Domain	Description
BID_x[7:0]	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.
BVALID_x	Output	ACLK_x	Write response valid. This signal indicates that the channel is signaling a valid write response.

Table 11: AXI Read Data Channel Signals (Interface to FPGA Fabric)

Signal x is 0 or 1	Direction	Clock Domain	Description
RDATA_x[127:0]	Output	ACLK_x	(T20, T35): Read data.
RDATA_0[255:0]	Output	ACLK_0	(T55, T85, T120): AXI target 0 read data.
RDATA_1[127:0]	Output	ACLK_1	(T55, T85, T120): AXI target 1 read data.
RID_x[7:0]	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 12: AXI Write Data Channel Signals (Interface to FPGA Fabric)

Signal x is 0 or 1	Direction	Clock Domain	Description
WDATA_x[127:0]	Input	ACLK_x	(T20, T35): Write data.
WDATA_0[255:0]	Input	ACLK_0	(T55, T85, T120): AXI target 0 write data.
WDATA_1[127:0]	Input	ACLK_1	(T55, T85, T120): AXI target 1 write data.
WID_x[7:0]	Input	ACLK_x	Write ID tag. This signal is the ID tag of the write data transfer.

Signal x is 0 or 1	Direction	Clock Domain	Description
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[15:0]	Input	ACLK_x	(T20, T35): 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.
WSTRB_0[31:0] WSTRB_1[15:0]	Input	ACLK_x	(T55, T85, T120): 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 13: DDR DRAM I²C Interface Signals

Signal	Direction	Description
CFG_SCL_IN	Input	Clock input.
CFG_SDA_IN	Input	Data input.
CFG_SDA_OEN	Output	SDA output enable.

Table 14: DDR DRAM Startup Sequencer Signals

Signal	Direction	Description
CFG_SEQ_RST	Input	Active-high DDR configuration controller reset.
CFG_SEQ_START	Input	Start the DDR configuration controller.

Table 15: DDR DRAM Reset Signal

Signal	Direction	Description
CFG_RST_N	Input	Active-low master DDR DRAM reset. After you de-assert RST_N, you need to reconfigure and initialize before performing memory operations.

Table 16: DDR DRAM Pads

Signal	Direction	Description
DDR_A[15:0]	Output	Address signals to the memories.
DDR_BA[2:0]	Output	Bank signals to/from the memories.
DDR_CAS_N	Output	Active-low column address strobe signal to the memories.
DDR_CKE	Output	Active-high clock enable signals to the memories.
DDR_CK	Output	Active-high clock signals to/from the memories. The clock to the memories and to the memory controller must be the same clock frequency and phase.
DDR_CK_N	Output	Active-low clock signals to/from the memories. The clock to the memories and to the memory controller must be the same clock frequency and phase.
DDR_CS_N	Output	Active-low chip select signals to the memories.
DDR_DQ[n:0]	Bidirectional	Data bus to/from the memories. For writes, the pad drives these signals. For reads, the memory drives these signals. These signals are connected to the DQ inputs on the memories. n is 7, 15, or 31 depending on the FPGA and DQ width.
DDR_DM[n]	Output	Active-high data-mask signals to the memories. n is 1, 1:0, or 3:0 depending on the FPGA and DQ width.
DDR_DQS_N[n:0]	Bidirectional	Differential data strobes to/from the memories. For writes, the pad drives these signals. For reads, the memory drives these signals. These signals are connected to the DQS inputs on the memories. n is 1, 1:0, or 3:0 depending on the FPGA and DQ width.
DDR_DQS[n:0]	Bidirectional	
DDR_ODT	Output	ODT signal to the memories.
DDR_RAS_N	Output	Active-low row address strobe signal to the memories.
DDR_RST_N	Output	Active-low reset signals to the memories.
DDR_WE_N	Output	Active-low write enable strobe signal to the memories.
DDR_VREF	Bidirectional	Reference voltage.
DDR_ZQ	Bidirectional	ZQ calibration pin.

DDR Interface Designer Settings

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

Table 17: Base Tab

Parameter	Choices	Notes
DDR Resource	None, DDR_0	Only one resource available.
Instance Name	User defined	Indicate the DDR instance name. This name is the prefix for all DDR signals.
Memory Type	DDR3, LPDDR2, LPDDR3	Choose the memory type you want to use.

Table 18: Configuration Tab

Parameter	Choices	Notes
Select Preset		The Select Preset button opens a list of popular DDR memory configurations. Choose a preset to populate the configuration choices. If you do not want to use a preset, you can specify the memory configuration manually.
DQ Width	x8, x16, x32	DQ bus width. The width choices vary depending on the FPGA and package.
Type	DDR3, LPDDR2, LPDDR3	Memory type.
Enable Advanced Density Setting	On, Off	If you do not want to use the available density from the Density parameter, you can set the memory row and column width.
Row Width	0 - 99	Specify the memory row width.
Column Width	0 - 99	Specify the memory column width.
DDR3		
Speed Grade	1066E, 1066F, 1066G, 800D, 800E	Memory speed.
Width	x8, x16	Memory width.
Density	1G, 2G, 4G, 8G	Memory density in bits.
LPDDR2		
Speed Grade	400, 533, 667, 800, 1066	Memory speed.
Width	x16, x32	Memory width. The width choices vary depending on the FPGA and package.
Density	256M, 512M, 1G, 2G, 4G	Memory density in bits.
LPDDR3		
Speed Grade	800, 1066	Memory speed.
Width	x16, x32	Memory width. The width choices vary depending on the FPGA and package.
Density	4G, 8G	Memory density in bits.

Table 19: Advanced Options Tab - FPGA Setting Subtab

Parameter	Choices	Notes
FPGA Input Termination	Varies depending on the memory type	Specify the termination value for the FPGA input/output pins.
FPGA Output Termination		

Table 20: Advanced Options Tab - Memory Mode Register Settings Subtab

Parameter	Choices	Notes
DDR3		
Burst Length	8	Specify the burst length (only 8 is supported).
DLL Precharge Power Down	On, Off	Specify whether the DLL in the memory device is off or on during precharge power-down.


Parameter	Choices	Notes
Memory Auto Self-Refresh	Auto, Manual	Turn on or off auto-self refresh feature in memory device.
Memory CAS Latency (CL)	5 - 14	Specify the number of clock cycle between read command and the availability of output data at the memory device.
Memory Write CAS Latency (CWL)	5 - 12	Specify the number of clock cycle from the releasing of the internal write to the latching of the first data in at the memory device.
Memory Dynamic ODT (Rtt_WR)	Off, RZQ/2, RZQ/4	Specify the mode of dynamic ODT feature of memory device.
Memory Input Termination (Rtt_nom)	Off, RZQ/2, RZQ/4, RZQ/6, RZQ/8, RZQ/12	Specify the input termination value of the memory device.
Memory Output Termination	RZQ/6, RZQ/7	Specify the output termination value of the memory device.
Read Burst Type	Interleaved, Sequential	Specify whether accesses within a give burst are in sequential or interleaved order.
Sef-Refresh Temperature	Extended, Normal	Specify whether the self refresh temperature is normal or extended mode.
LPDDR2		
FPGA Input Termination (Ω)	120, Off	Specify the input termination of the FPGA device. Used in non-JEDEC standard only.  Note: Enabling this option leads to higher power consumption.
Burst Length	8	Specify the burst length (only 8 is supported).
Output Drive Strength	34.3, 40, 48, 60, 80, 120	Specify the output termination value of memory device.
Read Burst Type	Interleaved, Sequential	Specify whether accesses within a given burst are in sequential or interleaved order.
Read/Write Latency	RL=3/WL=1, RL=4/WL=2 RL=5/WL=2, RL=6/WL=3 RL=7/WL=4, RL=8/WL=4	Specify the read/write latency of the memory device.
LPDDR3		
DQ ODT	Disable, RZQ1, RZQ2, RZQ4	Specify the input termination value of memory device.
Output Drive Strength	34.3 34.3 pull-down/40 pull up 34.3 pull-down/48 pull up 40 40 pull down/48 pull up 48	Specify the output termination value of memory device.
Read/Write Latency	RL=3/WL=1, RL=6/WL=3 RL=8/WL=4, RL=9/WL=5	Specify the read/write latency of the memory device.

Table 21: Advanced Options Tab - Memory Timing Settings Subtab

Parameter	Choices	Notes
tFAW, Four Bank Active Window (ns)	User defined	Enter the timing parameters from the memory device's data sheet.
tRAS, Active to Precharge Command Period (ns)		
tRC, Active to Active or REF Command Period (ns)		
tRCD, Active to Read or Write Delay (ns)		
tREFI, Average Periodic Refresh Interval (ns)		
tRFC, Refresh to Active or Refresh to Refresh Delay (ns)		
tRP, Precharge Command Period (ns)		
tRRD, Active to Active Command Period (ns)		
tRTP, Internal Read to Precharge Delay (ns)		
tWTR, Internal Write to Read Command Delay (ns)		

Table 22: Advanced Options Tab - Controller Settings Subtab

Parameter	Choices	Notes
Controller to Memory Address Mapping	BANK-ROW-COL ROW-BANK-COL ROW-COL_HIGH-BANK-COL_LOW	Specify the mapping between the address of AXI interface and column, row, and bank address of memory device.
Enable Auto Power Down	Active, Off, Pre-Charge	Specify whether to allow automatic entry into power-down mode (pre-charge or active) after a specific amount of idle time.
Enable Self Refresh Controls	No, Yes	Specify whether to enable automatic entry into self-refresh mode after specific amount of idle period.

Table 23: Advanced Options Tab - Gate Delay Tuning Settings Subtab

Parameter	Choices	Notes
Enable Gate Delay Override	On or off	Turning this option on allows you to fine-tune the gate-delay values. This is an expert only setting.
Gate Coarse Delay Tuning	0 - 5	
Gate Fine Delay Tuning	0 - 255	

Table 24: Control Tab

Option	Notes
Disable Control	When selected, this option disables calibration and user reset.
Enable Calibration	Turn on to enable optional PHY calibration pins (master reset, SCL, and SDA pins). Efinix recommends that you use the default pin names. The names are prefixed with the instance name you specified in the Base tab. These pins connect to the DDR Hard Memory Controller - Calibration and Reset IP core.
Enable User Reset	Turn on to enable optional reset pins (master reset and sequencer start/reset). Efinix recommends that you use the default pin names. The names are prefixed with the instance name you specified in the Base tab. These pins connect to the DDR Hard Memory Controller - Reset IP core.
Enable Reset and Calibration	Turn on to enable the pins for calibration and user reset. These pins connect to the DDR Hard Memory Controller - Calibration and Reset IP core.

Table 25: 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 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.
Shared Read/Write Address Channel tab Write Response Channel tab Read Data Channel tab Write Data Channel tab	User defined	This tab defines the AXI signal names. Efinix recommends that you use the default names. The signals are prefixed with the instance name you specified in the Base tab.

Using the DDR Block

You can add one DDR interface block to your design ([see which packages support DDR](#)). Configuration settings are arranged in tabs.

- **Base**—Choose the resource and specify an instance name. This name becomes the prefix for all of the DDR interface signals. Also choose the **Memory Type** (DDR3, LPDDR2, or LPDDR3).
- **Configuration**—Specify the type of memory to which you want to connect. You can choose a preset configuration by clicking **Select Preset**, or you can manually specify the DQ width, speed, and density. T20 and T35 FPGAs support a x16 DQ width; T55, T85, and T120 FPGAs support x16 or X32 DQ widths.
- **Advanced Options**—Make the following settings:

Accordion Tab	Settings
FPGA Settings	Choose the input and output termination. The choices vary depending on the memory type you select in the Base tab.
Memory Mode Register Settings	Memory-specific settings. The choices vary depending on the memory type you select in the Base tab.
Memory Timing Settings	Specify the timing settings for the memory device you are using.
Controller Settings	Select how the memory address is mapped, and auto-power-down and self refresh behavior.
Gate Delay Tuning Settings	Optionally enable a gate delay override and specify coarse and fine delay tuning.

- **Control**—Optionally enable PHY calibration or soft reset. If you use this option, Efinix recommends that you keep the default pin names.
- **AXI 0**—Enable the AXI interface target 0 and specify the name of the AXI input clock pin. Efinix recommends that you keep the default names.
- **AXI 1**—Enable the AXI interface target 1 and specify the name of the AXI input clock pin. Efinix recommends that you keep the default names.

Design Check: DDR 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.

ddr_rule_adv_density (error)

Message	Row width / column width have to be greater than 0 when advanced density setting is enabled
To fix	Adjust the row width / column width for advanced density setting.

ddr_rule_config_settings (error)

Message	Invalid combination of configuration settings
To fix	Select only the available configuration settings under Configuration tab.

ddr_rule_controller_settings (error)

Message	Invalid selection in Controller Settings: <setting>
To fix	Select only the available configuration settings under Advance Setting > Controller tab.

ddr_rule_empty_pins (error)

Message	Empty pin names found
To fix	If you enable AXI target or the calibration, you must enter pin name for pins in that category.

ddr_rule_fpga_settings (error)

Message	Invalid selection in FPGA Settings: <setting>
To fix	Select only the available configuration settings under Advance Setting > FPGA tab.

ddr_rule_gate_delay_settings (error)

Message	Values invalid in Gate Delay Tuning Settings: <setting>
To fix	Select only the available configuration settings under Advance Setting > Gate Delay Tuning tab.

ddr_rule_invalid_pins (error)

Message	Invalid pin names found: <pin names>
To fix	Update the pin names. Valid characters are alphanumeric characters with dash and underscore only.

ddr_rule_lvds_ref_clock (warning)

Message	It is not recommended to use LVDS Rx as reference clock resource to drive the PHY Clock, as the DDR interface will not be initialized during configuration phase. We recommend that users reset the DDR interface after entering user mode to ensure the DDR memory was fully initialized.
To fix	Use another PLL reference clock resource other than LVDS RX to drive the DDR PHY Clock.

[ddr_rule_memory_settings \(error\)](#)

Message	Invalid selection in Memory Mode Register Settings: <setting>
To fix	Select only the available configuration settings under Advance Setting > Memory Mode Register tab.

[ddr_rule_memory_settings \(warning\)](#)

Message	For the <speed grade> speed grade, RL/WL values are below the minimum. (Suggested minimum value: RL=<>, WL=<>)
To fix	Change the Read/Write Latency value (Advanced Options tab > Memory Mode Register Settings subtab) to be equal to or greater than the minimum suggested value.

[ddr_rule_mem_timing_settings \(error\)](#)

Message	Values out of valid range in Memory Timing Settings: <setting>
To fix	Select only the available configuration settings under Advance Setting > Memory Timing tab.

[ddr_rule_minimal_usage \(error\)](#)

Message	None of the AXI targets are enabled
To fix	Configure any of the AXI target when the DDR block is instantiated.

[ddr_rule_phy_clock \(error\)](#)

Message	Output clock 0 in PLL resource <pll> for PHY Clock not configured
To fix	Configure the PLL instance that drives the PHY Clock Input, with only enabling the output clock 0.
Message	Output clock 0 in PLL <pll> not configured
To fix	Configure the output clock 0 of the PLL instance that drives the PHY Clock Input.

[ddr_rule_phy_clock \(warning\)](#)

Message	This configuration is not recommended as it can result in lower DDR performance and we recommend that users reset the DDR interface after entering user mode to ensure the DDR memory was fully initialized
To fix	Update the PLL Instance to use internal or local feedback mode for better DDR performance.

[ddr_rule_phy_clock_freq \(error\)](#)

Message	DDR PHY Clock frequency <freq>MHz exceeds maximum range <freq>MHz
To fix	The output clock frequency divided by possible clk divider value must be less or equal to 100 MHz.

[ddr_rule_pll_feedback \(error\)](#)

Message	Feedback mode for PLL <instance> can only be set to internal or local when DDR is configured
To fix	Change the feedback mode for the PLL instance.

[ddr_rule_ref_clock_freq \(warning\)](#)

Message	DDR may require auto-calibration for performance above 800 Mbps (Ref Clock Freq: {}MHz)
To fix	DDR may require auto-calibration for performance above 800 Mbps. See DDR Hard Memory Controller-Calibration and Reset Core User Guide for more information.

[ddr_rule_resource \(error\)](#)

Message	Resource name is empty
To fix	Enter a valid resource name.

Message	Resource <name> is not a valid DDR device instance
To fix	The resource you specified does not exist. Check whether you have a typo in the resource name.

GPIO Interface

Contents:

- [About the General-Purpose I/O Logic and Buffer](#)
- [Using the GPIO Block](#)
- [Using LVDS as GPIO](#)
- [Using the GPIO Bus Block](#)
- [Design Check: GPIO Messages](#)

Trion[®] 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.

Trion[®] 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
- LVDS as GPIO where the LVDS pin acts as GPIO instead of the LVDS function

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

About the General-Purpose I/O Logic and Buffer

The GPIO support the 3.3 V LVTTL and 1.8 V, 2.5 V, and 3.3 V LVCMOS I/O standards. The GPIOs are grouped into banks. Each bank has its own VCCIO that sets the bank voltage for the I/O standard.

Each GPIO 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.

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

Table 26: 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>Some FPGA/package combinations support DDIO. 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>Some FPGA/package combinations support DDIO. In DDIO mode, two registers sample the data on the positive and negative edges of the input clock, creating two data streams.</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 excluding LVDS as GPIO are configured in weak pull-up mode. The LVDS as GPIO pins are tri-stated without a pull-up or pull-down resistor.

By default, 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.

Table 27: Features for GPIO and LVDS as GPIO by FPGA and Package

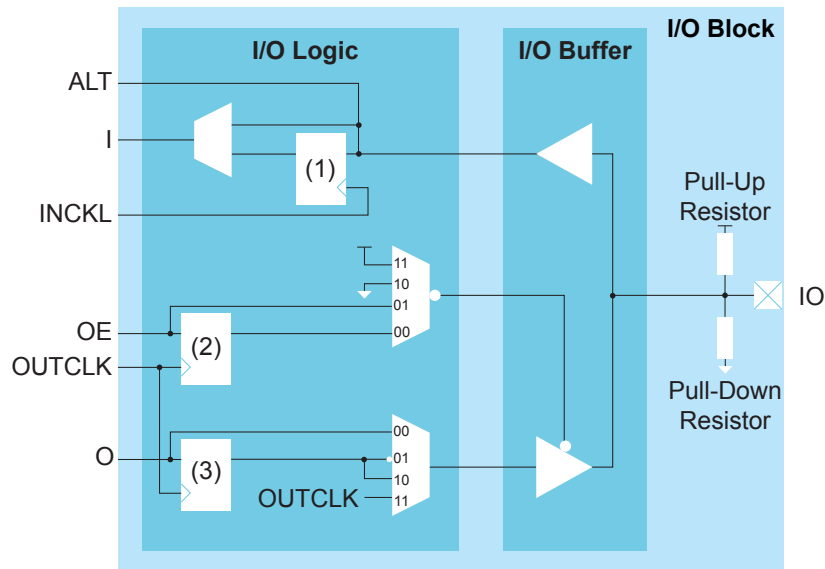
Package	Supported Features	
	GPIO	LVDS GPIO
<i>T4/T8</i>		
BGA49 BGA81	Schmitt Trigger Variable Drive Strength Pull-up Pull-down Slew Rate	-
<i>T8/T13/T20</i>		
WLCSP80 QFP100F3 QFP144 BGA169 BGA256	DDIO Schmitt Trigger Variable Drive Strength Pull-up Pull-down Slew Rate	Pull-up
<i>T20/T35/T55/T85/T120</i>		

Package	Supported Features	
	GPIO	LVDS GPIO
BGA324	DDIO	Variable Drive Strength
BGA400	Schmitt Trigger	Pull-up
BGA484	Variable Drive Strength	Slew Rate
BGA576	Pull-up	
	Pull-down	
	Slew Rate	

Simple I/O Buffer

T4/T8 FPGAs in F49 and F81 packages have simple I/O interface with logic and a buffer.

Figure 13: T8/T4 I/O Interface Block



Notes:

1. Input Register
2. Output Enable Register
3. Output Register

Table 28: GPIO Signals

Signal	Direction	Description
I	Output	Input data from the GPIO pad to the core fabric.
ALT	Output	Alternative input connection (in the Interface Designer, the input Register Option is none). Alternative connections are GCLK, GCTRL, and PLL_CLKIN.
O	Input	Output data to GPIO pad from the core fabric.
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 register. This clock is not visible in the user netlist.
INCLK	Input	Core clock that controls the input register. This clock is not visible in the user netlist.

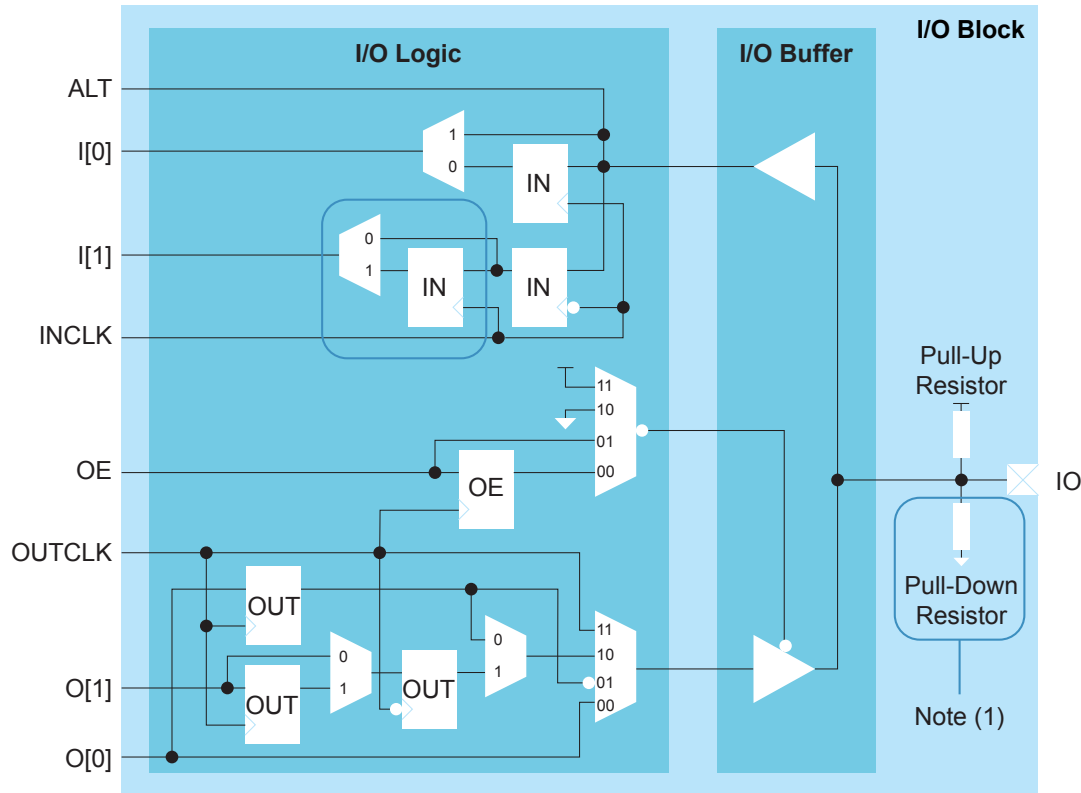
Table 29: GPIO Pads

Signal	Direction	Description
IO	Bidirectional	GPIO pad.

Complex I/O Buffer

T8 (Q144 only), T13, T20, T35, T55, T85, and T120 FPGAs have a complex I/O interface with logic and a buffer.

Figure 14: I/O Interface Block



1. GPIO pins using LVDS resources do not have a pull-down resistor.

Note: LVDS as GPIO do not have double data I/O (DDIO).

Table 30: GPIO Signals (Interface to FPGA Fabric)

Signal	Direction	Description
I[1:0]	Output	Input data from the GPIO pad to the core fabric. I[0] is the normal input to the core. In DDIO mode, I[0] is the data captured on the positive clock edge (HI pin name in the Interface Designer) and I[1] 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). Alternative connections are GCLK, GCTRL, PLL_CLKIN, and MIPI_CLKIN.
O[1:0]	Input	Output data to GPIO pad from the core fabric. O[0] is the normal output from the core. In DDIO mode, O[0] is the data output on the positive clock edge (HI pin name in the Interface Designer) and O[1] is the data output 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 unless you instantiate an EFX_GPIO_V2 primitive.
INCLK	Input	Core clock that controls the input registers. This clock is not visible in the user netlist unless you instantiate an EFX_GPIO_V2 primitive.

Table 31: GPIO Pads

Signal	Direction	Description
IO	Bidirectional	GPIO pad.

Double-Data I/O

Some Trion 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.

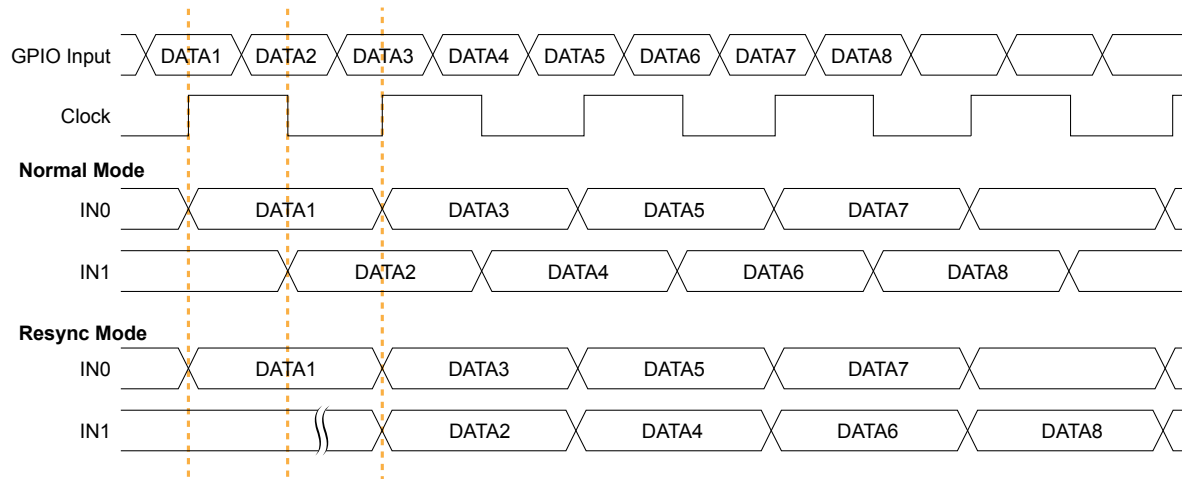
In normal mode, the interface receives or sends data directly to or from the core on the positive and negative clock edges. In resync mode, the interface resynchronizes the data to pass both signals on the positive clock edge only.

Not all GPIO support DDIO; additionally, LVDS as GPIO (that is, single ended I/O) do not support DDIO functionality.



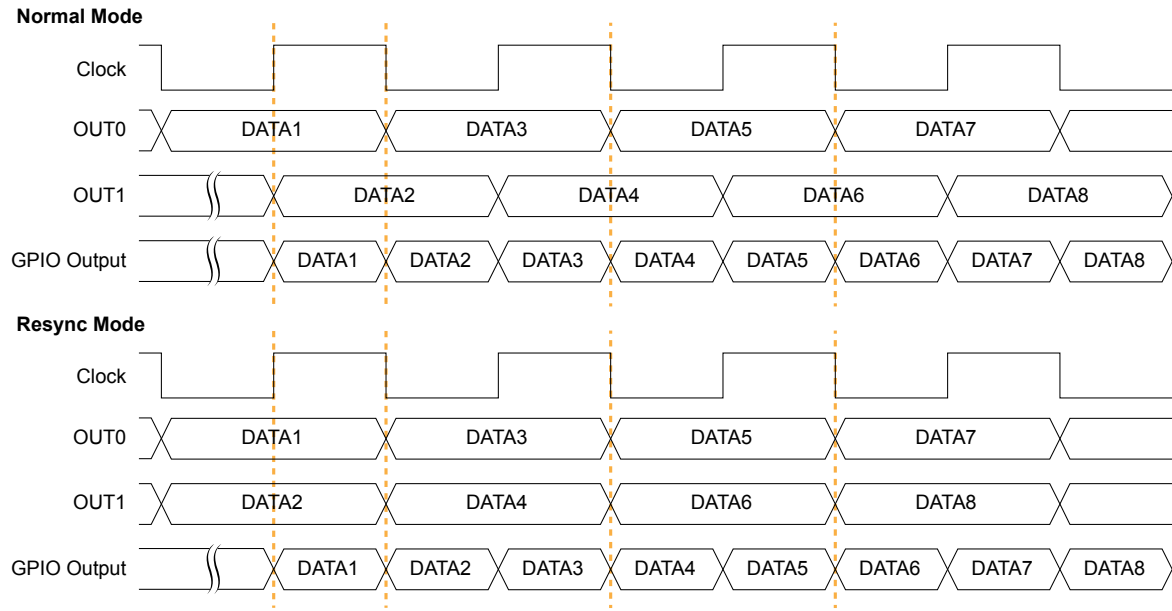
Note: The Resource Assigner in the Efinity® Interface Designer shows which GPIO support DDIO.

Figure 15: 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 16: DDIO Output Timing Waveform



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

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 dedicated GPIO resources or to LVDS resources. When you use LVDS resources as GPIO, some features are unavailable, depending on the FPGA. 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 LVDS pins as GPIO, make sure to leave at least 2 pairs of unassigned LVDS pins between any GPIO and LVDS 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 32: Input Mode Options

Option	Choices	Description						
Connection Type	normal, gclk, gctrl, pll_clkln, mipi_clkln	<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_clkln or mipi_clkln, 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>For FPGAs that have DDIO, 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="716 856 1422 1108"> <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. <i><pin name>_hi</i> is the positive edge and <i><pin name>_lo</i> is the negative edge.</td> </tr> </tbody> </table>	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. <i><pin name>_hi</i> is the positive edge and <i><pin name>_lo</i> 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. <i><pin name>_hi</i> is the positive edge and <i><pin name>_lo</i> is the negative edge.							
Pull Option	none, weak pullup, weak pulldown	Specify if you want a pull option.						
Enable Schmitt Trigger	On or off	Optionally enable a Schmitt trigger.						

Output Mode

Use output mode for output signals.

Table 33: 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.

Option	Choices	Description						
Register Option	none, register, inv_register	<p>Choose whether the output is registered or has an inverted register.</p> <p>For FPGAs that have DDIO, 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"> <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. <i><pin name>_hi</i> is the positive edge and <i><pin name>_lo</i> is the negative edge.</td> </tr> </tbody> </table> <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. <i><pin name>_hi</i> is the positive edge and <i><pin name>_lo</i> 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. <i><pin name>_hi</i> is the positive edge and <i><pin name>_lo</i> is the negative edge.							
Drive Strength	1, 2, 3, 4	Choose the drive strength level.						
Enable Fast Slew Rate	On or off	Optionally enable slew rate.						

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 LVDS as GPIO

You can use LVDS as GPIO by simply creating a GPIO block and assigning an LVDS resource to it. When you use LVDS resources as GPIO, some features are unavailable, depending on the FPGA as described in [Table 27: Features for GPIO and LVDS as GPIO by FPGA and Package](#) on page 42.

Assign a resource for the signal using the Resource Assigner.



Important: When LVDS resources are used for both LVDS and GPIO within the same bank, they must be separated by 2 unused pairs of LVDS pins to avoid any unwanted interference. The Efinity software issues an error if you do not leave this separation. Refer to [Table 34: LVDS Resources Assignment Example with LVDS and GPIO Signals in Same Bank](#) on page 51.

Table 34: LVDS Resources Assignment Example with LVDS and GPIO Signals in Same Bank

Bank Number	Pin Names	Assigned Signals
BANK4A	GPIOB_RXN06	LVDS
BANK4A	GPIOB_RXP06	LVDS
BANK4A	GPIOB_RXN07	X
BANK4A	GPIOB_RXP07	X
BANK4A	GPIOB_RXN08	X
BANK4A	GPIOB_RXP08	X
BANK4A	GPIOB_RXN09	GPIO
BANK4A	GPIOB_RXP09	GPIO
BANK4A	GPIOB_RXN10	X
BANK4A	GPIOB_RXP10	X
BANK4A	GPIOB_RXN12_EXTFB0	LVDS
BANK4A	GPIOB_RXP12_EXTFB0	LVDS
BANK4B	GPIOB_TXN00	X
BANK4B	GPIOB_TXP00	X
BANK4B	GPIOB_TXN01	X
BANK4B	GPIOB_TXP01	X
BANK4B	GPIOB_TXN02	GPIO
BANK4B	GPIOB_TXP02	GPIO
BANK4B	GPIOB_TXN03	X
BANK4B	GPIOB_TXP03	X
BANK4B	GPIOB_TXN05	LVDS
BANK4B	GPIOB_TXP05	LVDS

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 48 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.

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.

[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	Ensure the output enable pin name is in <BUS_NAME>[<IDX>] format. Example: my_bus[2]. Or remove the square bracket if the pin is not part of a bus.
Message	For input mode, weak_pulldown Pull Option is not supported
To fix	Disable weak pull down.

[gpio_rule_input_register \(error\)](#)

Message	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
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	Ensure the output enable pin name is in <BUS_NAME>[<IDX>] format. Example: my_bus[2]. Or remove the square bracket if the pin is not part of a bus.

[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.
Message	For input used in inout mode, weak_pulldown Pull Option is not supported
To fix	You get this message if you enable internal weak pull-down for LVDS as GPIO in inout mode. Disable weak pull down if you use inout mode.

[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 invalid
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 invalid
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 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_clkin 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_clkin connection to PLL clock source not being used in <instance>
To fix	The GPIO block is set to be a PLL reference clock (pll_clkin 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_clkin 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_clkin 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_clkin 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_clkin 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	Connection type <type> cannot be used on an unbonded MIPI resource
To fix	The MIPI TX Instance that associated to this GPIO is unbonded. Choose an available MIPI resource.
Message	A MIPI Tx must be configured when <type> connection type driving MIPI Tx is set
To fix	You receive this message when you set the connection type in GPIO but did not create a MIPI TX Instance that should be driving this GPIO. Check the MIPI TX Instance associated to this GPIO (Hardware / GPIO Resource dependent).

[gpio_rule_resource \(error\)](#)

Message	Resource name is empty
To fix	You need to choose a valid resource.
Message	Resource is not a valid GPIO device instance
To fix	You need to choose a valid resource.
Message	Resource is not a valid LVDS GPIO device instance
To fix	You need to choose a valid LVDS as GPIO 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_cfg_drive_strength \(error\)](#)

Message	Resource <resource> does not support Drive Strength feature. It will be ignored.
To fix	Confirm the drive strengths that are allowed for the I/O standard you want to use.

[gpio_rule_transmit_toggling \(warning\)](#)

Message	Bank <name> has <int> lvds gpio in inout/output mode that exceed max limit of <int> which can result in LVTTTL/LVCMOS simultaneous switching noise
To fix	Avoid using lvds gpio in output/inout mode more than the specified maximum count in the same bank.

[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.

[gpio_rule_cfg_slew_rate \(error\)](#)

Message	Resource <name> does not support Slew Rate feature. It will be ignored
To fix	LVDS as GPIO resources do not support slew rate. Either disable that option or choose another resource.

[gpio_rule_cck_pin \(error\)](#)

Message	The CCK pin cannot be used in user mode when LVDS Tx is configured
To fix	You cannot use the CCK pin in user mode when you have any LVDS TX configured in Q100F3 packages.

[gpio_rule_cfg_input_ddio \(warning\)](#)

Message	Resource <resource> does not support Double Data I/O feature. The LO pin name will be ignored
To fix	You get this message when set the LO pin name in resource that does not support it. Change to a resource that supports the double data I/O feature if you want to use the feature, else disable it.

[gpio_rule_cfg_input_ddio \(error\)](#)

Message	Resource <resource> does not support Double Data I/O feature. Please set the Double Data I/O Option to none or choose another resource.
To fix	You get this error when the GPIO resource does not support it. Change to a resource that supports the double data I/O feature if you want to use the feature.

[gpio_rule_cfg_output_ddio \(warning\)](#)

Message	Resource <resource> does not support Double Data I/O feature. The LO pin name will be ignored
To fix	You get this message when set the LO pin name in resource that does not support it. Change to a resource that supports the double data I/O feature if you want to use the feature, else disable it.

[gpio_rule_cfg_output_ddio \(error\)](#)

Message	Resource <resource> does not support Double Data I/O feature. Please set the Double Data I/O Option to none or choose another resource.
To fix	You get this error when the GPIO resource does not support it. Change to a resource that supports the double data I/O feature if you want to use the feature.

[gpio_rule_cfg_pull_option \(warning\)](#)

Message	Resoure <name> does not support weak pull down for Pull Option feature. It will be ignored
To fix	You get this error when enable weak pull down option in a resource that does not support it. Change to a resource that supports the weak pull-down feature if you want to use the feature, else disable it.

[gpio_rule_cfg_schmitt_trigger \(warning\)](#)

Message	Resource <name> does not support Schmitt Trigger feature. It will be ignored
To fix	You get this error when enable Schmitt Trigger option in a resource that does not support it. Change to a resource that supports the Schmitt Trigger feature if you want to use the feature, else disable it.

[gpio_rule_inst_name \(error\)](#)

Message	GPIO name <instance name> is used
To fix	You need to define a unique instance name.
Message	Instance name is empty
To fix	You need to choose a valid resource.

JTAG User TAP Interface

Contents:

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

Trion® FPGAs have dedicated JTAG pins to for 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.

Table 35: Supported JTAG Instructions

Instruction	Binary Code [3:0]	Description
SAMPLE/PRELOAD	0010	Enables the boundary-scan SAMPLE/PRELOAD operation
EXTEST	0000	Enables the boundary-scan EXTEST operation
BYPASS	1111	Enables BYPASS
IDCODE	0011	Enables shifting out the IDCODE
PROGRAM	0100	JTAG configuration
ENTERUSER	0111	Changes the FPGA into user mode.
JTAG_USER1	1000	Connects the JTAG User TAP 1.
JTAG_USER2	1001	Connects the JTAG User TAP 2.
JTAG_USER3	1010	Connects the JTAG User TAP 3.
JTAG_USER4	1011	Connects the JTAG User TAP 4.



Note: For detailed information about using JTAG for configuration, refer to [AN 006: Configuring Trion FPGAs](#).

For detailed information about using JTAG for boundary-scan testing, refer to [AN 021: Performing Boundary-Scan Testing on Trion FPGAs](#).

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. Trion FPGAs have two JTAG User TAP blocks. To use both USER1 and USER2, add 2 blocks to your interface design, one for each resource.

Table 36: 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.
<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 Messages

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_inst_name \(error\)](#)

Message	Instance name is empty
To fix	Enter a valid clock pin name.
Message	Valid characters are alphanumeric characters with dash and underscore only
To fix	Update the pin names. Valid characters are alphanumeric characters with dash and underscore only.

[jtag_rule_clock \(error\)](#)

Message	Test Clock name is empty
To fix	Enter a valid Test Clock (TCK) pin name.
Message	Test Clock name is invalid
To fix	Update the pin names. Valid characters are alphanumeric characters with dash and underscore only.

[jtag_rule_resource \(error\)](#)

Message	Resource name is empty
To fix	Enter a valid JTAG User Tap resource name.

Message	Resource is not a valid JTAG device instance.
To fix	Assign instance to a resource that exists in the device.

[jtag_rule_invalid_pins \(error\)](#)

Message	Invalid pin names found: <Pin description names>
To fix	Update the pin names. Valid characters are alphanumeric characters with dash and underscore only.

LVDS Interface

Contents:

- [About the LVDS Interface](#)
- [Using the LVDS Block](#)
- [Create an LVDS TX or RX Interface](#)
- [Design Check: LVDS Messages](#)

Some Trion[®] FPGAs support low-voltage differential signaling (LVDS) on their pins. LVDS offers the advantage of running at high speeds with low power. Refer to the [Package/Interface Support Matrix](#) on page 9 to find out if your FPGA supports LVDS. The following sections describe the LVDS pins and how to use them in your Efinity[®] RTL design.

About the LVDS Interface

The LVDS hard IP transmitters and receivers operate independently.

- LVDS TX consists of LVDS transmitter and serializer logic.
- LVDS RX consists of LVDS receiver, on-die termination, and de-serializer logic.

Trion[®] FPGAs have one or more PLLs for use with the LVDS receiver, depending on which FPGA you use.

You can use the LVDS TX and LVDS RX channels as single-ended GPIO pins, see [About the General-Purpose I/O Logic and Buffer](#) on page 41. The voltage supported depends on the FPGA.



Note: When LVDS resources are used for both LVDS and GPIO within the same bank, they must be separated by 2 unused pairs of LVDS pins to avoid any unwanted interference. The Efinity software issues an error if you do not leave this separation. Refer to [Table 34: LVDS Resources Assignment Example with LVDS and GPIO Signals in Same Bank](#) on page 51.

The LVDS hard IP has these features:

- Dedicated LVDS TX and RX channels (the number of channels depends on the FPGA and package)
- Up to 600 or 800 Mbps for LVDS data transmit or receive (depending on the FPGA and package)
- Supports serialization and deserialization factors: 8:1, 7:1, 6:1, 5:1, 4:1, 3:1, and 2:1
- Ability to disable serialization and deserialization
- Source synchronous clock output edge-aligned with data for LVDS transmitter and receiver
- 100 Ω on-die termination resistor for the LVDS receiver

LVDS TX

Figure 17: LVDS TX Interface Block Diagram

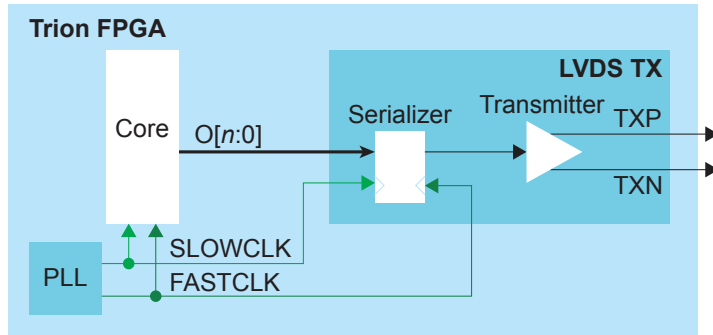


Table 37: LVDS TX Signals (Interface to FPGA Fabric)

Signal	Direction	Notes
$O[n-1:0]$	Input	Parallel output data where n is the serialization factor. A width of 1 bypasses the serializer.
FASTCLK	Input	Fast clock to serialize the data to the LVDS pads.
SLOWCLK	Input	Slow clock to latch the incoming data from the core.

Table 38: LVDS TX Pads

Pad	Direction	Description
TXP	Output	Differential P pad.
TXN	Output	Differential N pad.



Important: For QFP100F3 packages, do not toggle the CCK pin when any LVDS TX is used.

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 18: LVDS Timing Example Serialization Width of 8

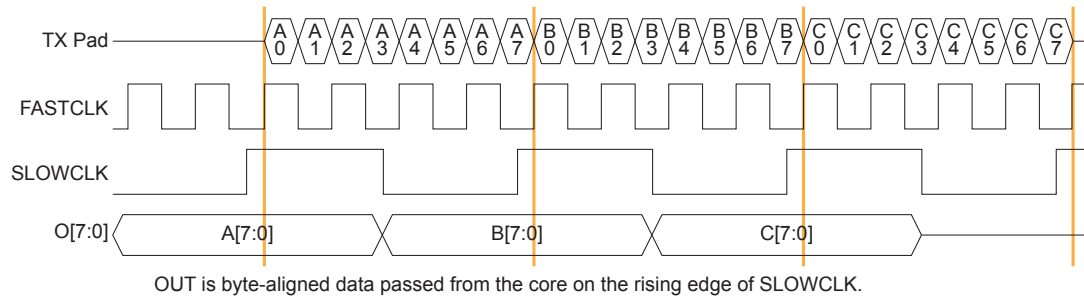


Figure 19: LVDS Timing Data and Clock Relationship Width of 8 (Parallel Clock Division=1)

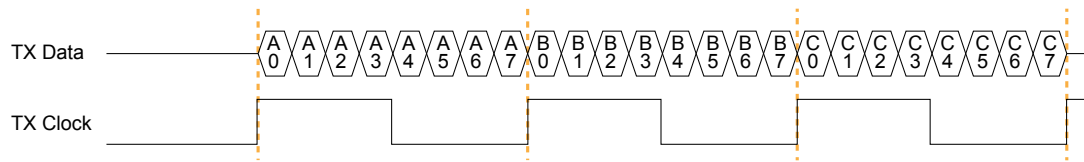


Figure 20: LVDS Timing Data and Clock Relationship Width of 7 (Parallel Clock Division=1)

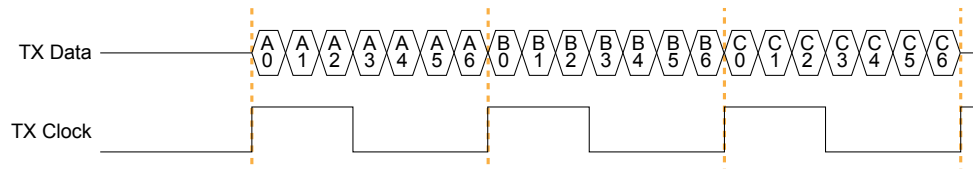


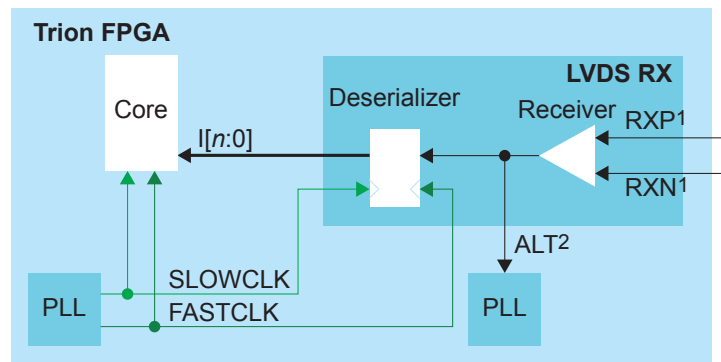
Table 39: LVDS TX Settings in Efinity® Interface Designer

Parameters	Choices	Notes
Instance Name	User defined	
LVDS Resource	Resource list	Choose a resource.
Mode	serial data output or reference clock output	serial data output —Simple output buffer or serialized output. reference clock output —Use the transmitter as a clock output. When choosing this mode, the Serialization Width you choose should match the serialization for the rest of the LVDS bus.
Parallel Clock Division	1, 2	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.
Enable Serialization	On or off	When off, the serializer is bypassed and the LVDS buffer is used as a normal output.
Serialization Width	2, 3, 4, 5, 6, 7, or 8	Supports 8:1, 7:1, 6:1, 5:1, 4:1, 3:1, and 2:1. Specify the serial clock and parallel clock.
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. Only available when serialization is disabled.
Reduce VOD Swing	On or off	When true, enables reduced output swing (similar to slow slew rate).

Parameters	Choices	Notes
Output Load	QFP144: 5, 7, or 10 All others: 3, 5, 7, or 10	Output load in pF. Use an output load of 7 pF or higher to achieve the maximum supported toggle rate. Refer to the data sheet for the maximum toggle rate.

LVDS RX

Figure 21: LVDS RX Interface Block Diagram



1. There is a ~30k Ω internal weak pull-up to VCCIO (3.3V).
2. Only available for an LVDS RX resource in bypass mode (deserialization width is 1).

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

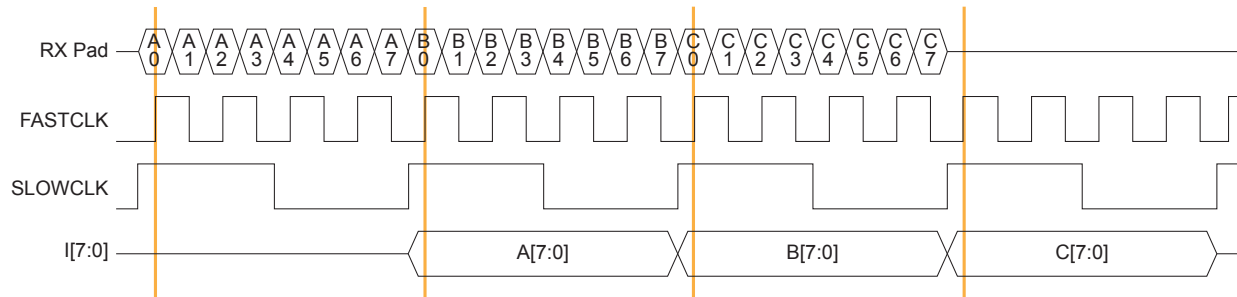
Signal	Direction	Notes
I[n-1:0]	Output	Parallel input data where n is the de-serialization factor. A width of 1 bypasses the deserializer.
ALT	Output	Alternative input, only available for an LVDS RX resource in bypass mode (deserialization width is 1; alternate connection type). Alternative connections are PLL_CLKIN.
FASTCLK	Input	Fast clock to de-serialize the data from the LVDS pads.
SLOWCLK	Input	Slow clock to latch the incoming data to the core.

Table 41: LVDS RX Pads

Pad	Direction	Description
RXP	Input	Differential P pad.
RXN	Input	Differential N pad.

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 22: LVDS RX Timing Example Serialization Width of 8



I is byte-aligned data passed to the core on the rising edge of SLOWCLK.

Table 42: LVDS RX Settings in Efinity® Interface Designer

Parameter	Choices	Notes
Instance Name	User defined	
LVDS Resource	Resource list	Choose a resource.
Connection Type	normal, pll_clkln	normal —Regular RX function. pll_clkln —Use the PLL CLKIN alternate function of the LVDS RX resource.
Input Pin/Bus Name	User defined	Input pin or bus that feeds the LVDS transmitter parallel data. The width should match the deserialization factor.
Enable Deserialization	On or off	When off, the de-serializer is bypassed and the LVDS buffer is used as a normal input. Specify the serial clock and parallel clock.
Deserialization Width	2, 3, 4, 5, 6, 7, or 8	Supports 8:1, 7:1, 6:1, 5:1, 4:1, 3:1, and 2:1.
Enable On-Die Termination	On or off	When on, enables an on-die 100-ohm resistor.
Static Mode Delay Setting	0 - 63	Choose the amount of static delay, each step adds approximately 25 ps of delay.

Using the LVDS Block

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

LVDS TX

The maximum LVDS rate is 800 Mbps. The serial clock frequency = parallel clock frequency * (serialization / 2).

- For a serialization of 3, the fast clock must be phase shifted by 45°.
- For all other serializations, the fast clock must be phase shifted by 90°.

Both clocks must come from the same PLL. The software issues an error if you do not follow these guidelines.



Note: Efinix® recommends that you select values of 2 or higher for the PLL post divider and output divider. These settings provide a more stable clock signal for faster speeds.

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.

New data is output on both edges of the serial clock.

LVDS RX

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 800 Mbps. The serial clock frequency = parallel clock frequency * (serialization / 2). The serial clock should use the 90 degree phase shift and both clocks must come from the same PLL. The software issues a warning if you do not use these guidelines.



Note: Efinix® recommends that you select 2 or higher for the PLL post divider and output divider. These settings provide a more stable clock signal for faster speeds. If the LVDS receiver speed is 600 Mbps or higher, the Efinity® software issues a warning if you select 1 as the PLL post divider and output divider values.

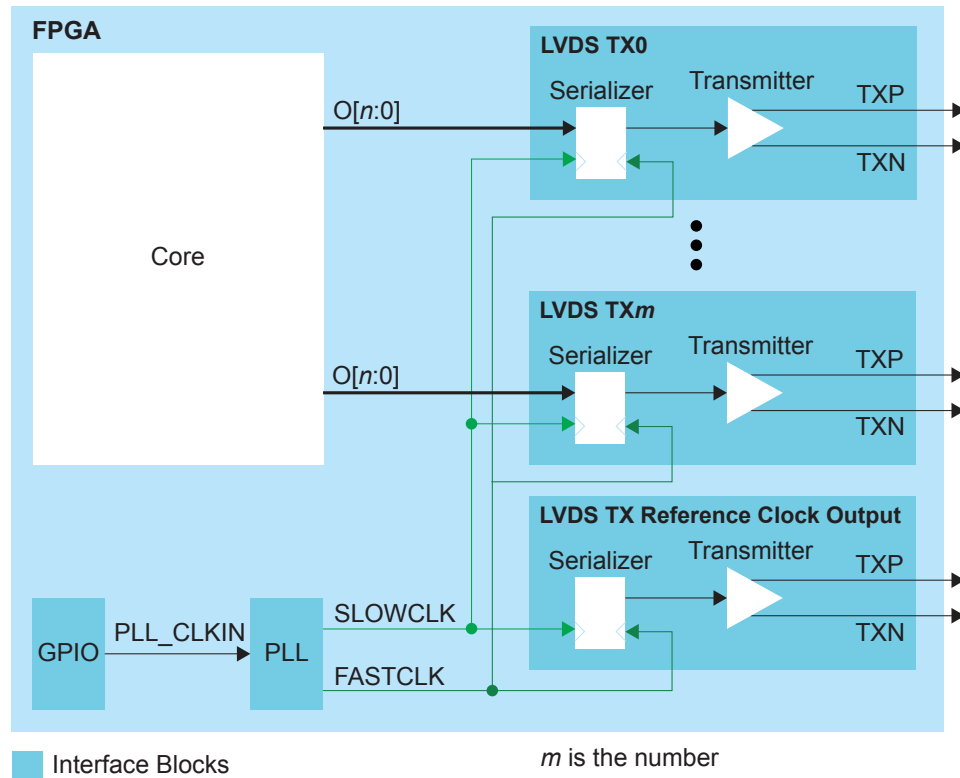
Create an LVDS TX or RX Interface

You build a complete interface using the Efinity® Interface Designer and LVDS, PLL, and GPIO blocks.

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 23: Complete LVDS TX Interface Block Diagram



Note: Use LVDS TX blocks from the same side of the FPGA to minimize skew between data lanes and TX reference clock output in an LVDS TX interface.

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	Always reserve PLL_BR0 for the RX interface when designing with T8 (Q144), T13, and T20 (Q100, Q144, F169, F256) FPGAs. Otherwise, you can use any PLL resource.
Reference Clock Mode	External
Reference Clock Frequency	Any

Option	Description
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_clkln
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:

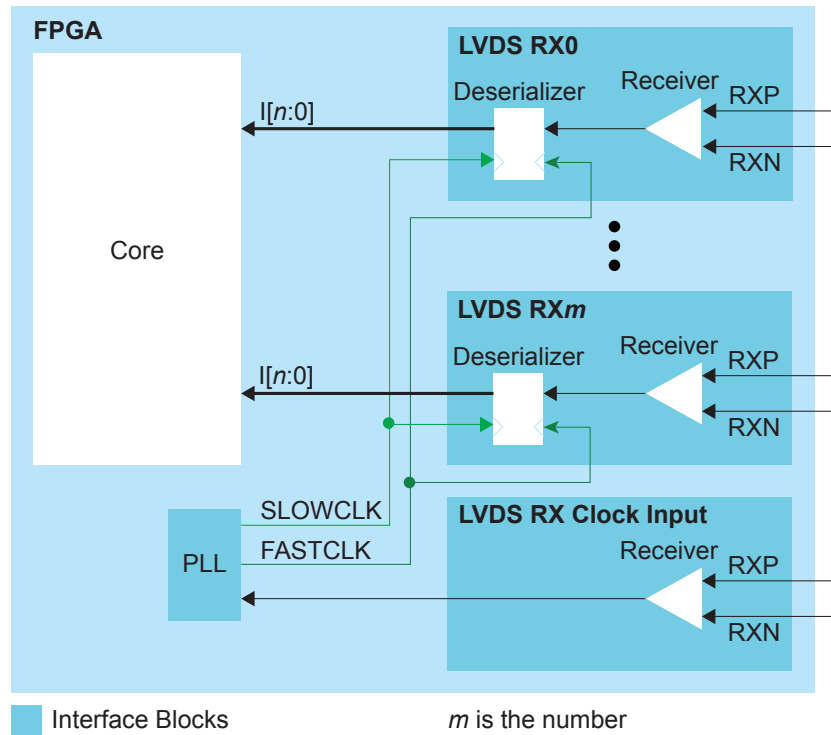
Option	Description
LVDS type	Transmitter (TX)
LVDS resource	Any channel
Mode	Reference clock output
Enable Serialization	On
Serialization width	<i>n</i>
Output pin/ bus name	Any
Parallel clock division	1: The output clock from the LVDS TX lane is parallel clock frequency. 2: The output clock from the 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.

Option	Description
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 24: Complete LVDS RX Interface Block Diagram



Note: Use LVDS RX blocks from the same side of the FPGA to minimize skew between data lanes and RX clock input in an LVDS RX interface.

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 GPIOB_CLK0 when designing with T8 (Q144), T13, and T20 (Q100, Q144, F169, F256) FPGAs. Otherwise, use any 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	Always use PLL_BR0 for the RX interface when designing with T8 (Q144), T13, and T20 (Q100, Q144, F169, F256) FPGAs. Otherwise, use any 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 Messages

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_clkout_mode \(error\)](#)

Message	Serial clock name must be configured in clock output mode
To fix	When you are using the clkout mode, 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 clkout mode, you need to specify the parallel clock pin name.
Message	Parallel clock division <value> is out of range <min>-<max>
To fix	Parallel clock division factor has to be within specified range (min=1, max=2).
Message	Parallel clock division <value> is not an integer
To fix	Parallel clock division factor has to be an integer.

[lvds_rule_output_mode \(error\)](#)

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 non-bypass mode, 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 non-bypass mode, you need to specify the serial clock pin name.

[lvds_rule_resource \(error\)](#)

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

[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_clkin 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_clkin connection to PLL clock source not being used in <instance>
To fix	The LVDS block is set to be a PLL reference clock (pll_clkin 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 LVDS RX's resource.
Message	pll_clkin 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_clkin 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 LVDS RX's resource.
Message	pll_clkin 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_clkin 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 LVDS RX's resource.

[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	One of the clock frequencies is 0
To fix	Set a valid output clock frequency.
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. serial clock frequency = parallel clock frequency * (serialization / 2)

[lvds_rule_rx_clock \(warning\)](#)

Message	Serial clock <string> phase shift has to be 90 degrees
To fix	Adjust the phase shift for the serial clock to be 90 degrees.

[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.
Message	Serialization must be configured
To fix	When you are using the LVDS deserializer, you need to set the serialization width,

[lvds_rule_rx_distance \(error\)](#)

Message	These Rx LVTTTL must be placed at least 2 pairs away from LVDS <name> in order to avoid noise coupling from LVTTTL to LVDS: <violated list>
To fix	When using LVDS pins as GPIO, make sure to leave at least 2 pair of unassigned LVDS pins between any LVDS and LVDS used as LVDS RX in the same bank. This separation reduces noise.

[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.
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: <resource>
To fix	Only a specific PLL instance can drive the LVDS RX clocks. Change the PLL to use that resource.
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_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	Parallel clock <name> phase shift has to be 0 degree
To fix	Set the Parallel clock <name>'s phase shift to 0 degree
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	One of the clock frequencies is 0
To fix	Set a valid output clock frequency.
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. serial clock frequency = parallel clock frequency * (serialization / 2)
Message	Serial clock <string> phase shift has to be 45 degrees
To fix	Adjust the phase shift for the serial clock to be 45 degrees. When serialization width is 3, serial clock phase shift has to be 45 degree.
Message	Allowed serial clock (<name>) phase shift values: 45, 90, 135 degrees
To fix	Set to the allowed phase shift values when serialization width is not 3.

[lvds_rule_tx_distance \(error\)](#)

Message	These Tx LVTTTL must be placed at least 2 pairs away from LVDS <name> in order to avoid noise coupling from LVTTTL to LVDS: <violated list>
To fix	When using LVDS pins as GPIO, make sure to leave at least 2 pair of unassigned LVDS pins between any LVDS and LVDS used as LVDS TX in the same bank. This separation reduces noise.

[lvds_rule_clkout_ser_disabled \(error\)](#)

Message	Output clock name must be configured in clock output mode with serialization disabled
To fix	Specify the output clock name.

[lvds_rule_rx_pll_feedback \(warning\)](#)

Message	PLL <pll_slow_inst_name> driving the LVDS Rx clock sources should have its feedback mode set to core for optimized performance
To fix	Set feedback mode of the PLL to core for better performance.

[lvds_rule_rx_parallelclk_to_refclk_freq \(error\)](#)

Message	Parallel clock frequency (<value>MHz) to PLL reference clock frequency (<value>MHz) ratio is expected to be an integer
To fix	Specify an integer value.

[lvds_rule_alt_config \(error\)](#)

Message	<resource> can only be used as an LVDS PLL reference clock
To fix	If LVDS Rx does not have pout (data) pins, then only can be used as a reference clock (i.e. GPIOB_RX_CLK).

MIPI CSI-2 Interface

Contents:

- [About the MIPI Interface](#)
- [Using the MIPI Block](#)
- [Design Check: MIPI Messages](#)

Some Trion FPGAs have a hardened Mobile Industry Processor Interface (MIPI) block to communicate with cameras and sensors. Refer to the [Package/Interface Support Matrix](#) on page 9 to find out if your FPGA supports MIPI.



About the MIPI Interface

The MIPI CSI-2 interface is the most widely used camera interface for mobile.⁽⁴⁾ You can use this interface to build single- or multi-camera designs for a variety of applications.

Trion FPGAs can include hardened MIPI D-PHY blocks (4 data lanes and 1 clock lane) with MIPI CSI-2 IP blocks. The MIPI RX and MIPI TX can operate independently with dedicated I/O banks.



Note: The MIPI D-PHY and CSI-2 controller are hard blocks; users cannot bypass the CSI-2 controller to access the D-PHY directly for non-CSI-2 applications.

The MIPI TX/RX interface supports the MIPI CSI-2 specification v1.3 and the MIPI D-PHY specification v1.1. It has the following features:

- Programmable data lane configuration supporting 1, 2, or 4 lanes
- High-speed mode supports up to 1.5 Gbps data rates per lane
- Operates in continuous and non-continuous clock modes
- 64 bit pixel interface for cameras
- Supports Ultra-Low Power State (ULPS)

Table 43: MIPI Supported Data Types

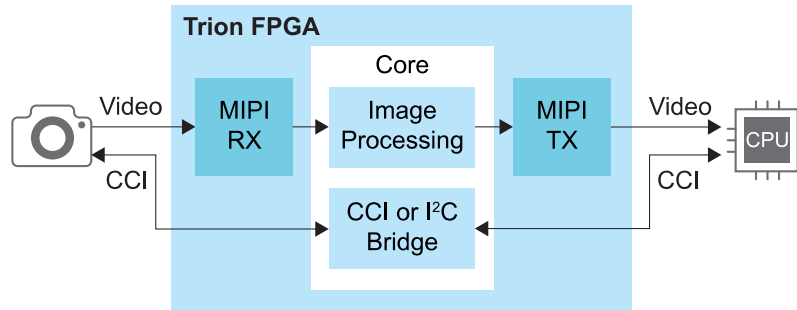
Supported Data Type	Format
RAW	RAW6, RAW7, RAW8, RAW10, RAW12, RAW14
YUV	YUV420 8-bit (legacy), YUV420 8-bit, YUV420 10-bit, YUV420 8-bit (CSPS), YUV420 10-bit (CSPS), YUV422 8-bit, YUV422 10-bit

⁽⁴⁾ Source: MIPI Alliance <https://www.mipi.org/specifications/csi-2>

Supported Data Type	Format
RGB	RGB444, RGB555, RGB565, RGB666, RGB888
User Defined	8 bit format

With more than one MIPI TX and RX blocks, Trion® FPGAs support a variety of video applications.

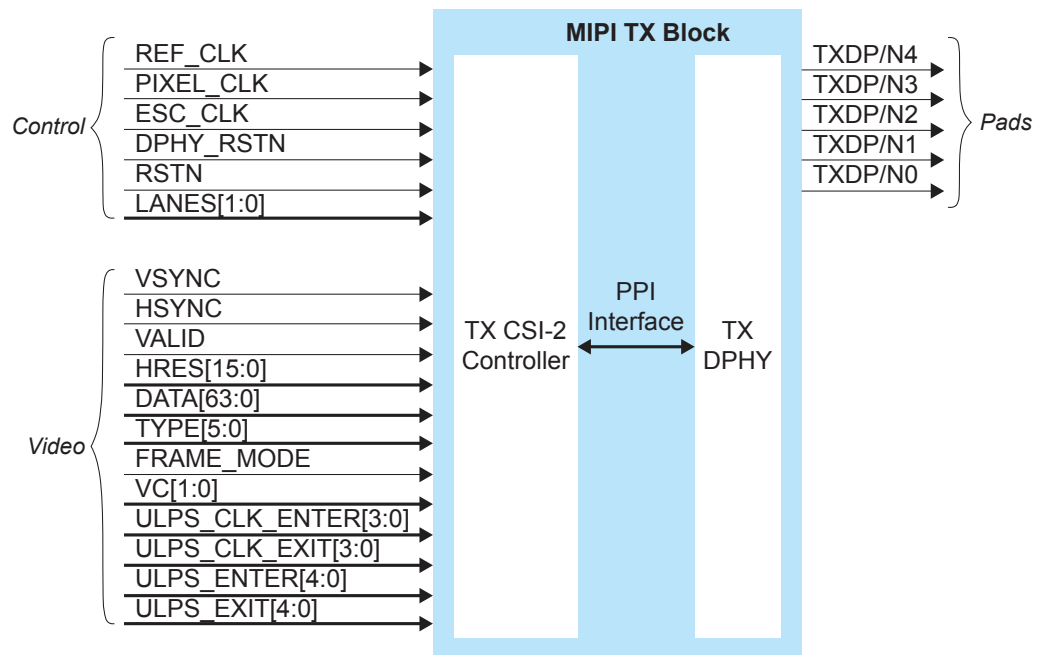
Figure 25: MIPI Example System



MIPI TX

The MIPI TX is a transmitter interface that translates video data from the Trion® core into packetized data sent over the HSSI interface to the board. Five high-speed differential pin pairs (four data, one clock), each of which represent a lane, connect to the board. Control and video signals connect from the MIPI interface to the core.

Figure 26: MIPI TX x4 Block Diagram



The control signals determine the clocking and how many transceiver lanes are used. All control signals are required except the two reset signals. The reset signals are optional, however, you must use both signals or neither.

The MIPI block requires an escape clock (ESC_CLK) for use when the MIPI interface is in escape (low-power) mode, which runs between 11 and 20 MHz.



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

The video signals receive the video data from the core. The MIPI interface block encodes it and sends it out through the MIPI D-PHY lanes.

Figure 27: MIPI TX Interface Block Diagram

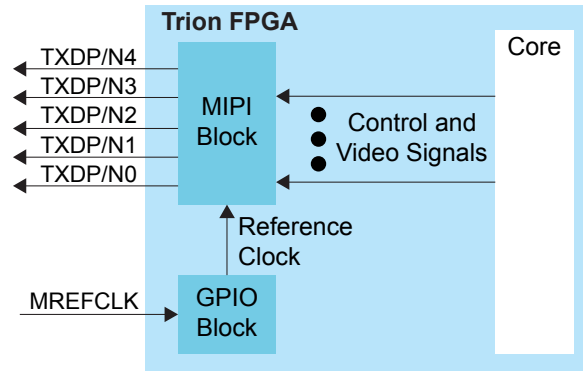


Table 44: MIPI TX Control Signals (Interface to FPGA Fabric)

Signal	Direction	Clock Domain	Description
REF_CLK	Input	N/A	Reference clock for the internal MIPI TX PLL used to generate the transmitted data. The FPGA has a dedicated GPIO resource (MREFCLK) that you must configure to provide the reference clock. All of the MIPI TX blocks share this resource. The frequency is set using Interface Designer configuration options.
PIXEL_CLK	Input	N/A	Clock used for transferring data from the core to the MIPI TX block. The frequency is based on the number of lanes and video format. Refer to Understanding the RX and TX Pixel Clock on page 93.
ESC_CLK	Input	N/A	Slow clock for escape mode (11 - 20 MHz).
DPHY_RSTN	Input	N/A	(Optional) Reset for the D-PHY logic, active low. Reset with the controller. See MIPI Reset Timing .
RSTN	Input	N/A	(Optional) Reset for the CSI-2 controller logic, active low. Typically, you reset the controller with the PHY. (See MIPI Reset Timing .) However, when dynamically changing the horizontal resolution, you only need to trigger RSTN. (See TX Requirements for Dynamically Changing the Horizontal Resolution).
LANES[1:0]	Input	PIXEL_CLK	Determines the number of lanes enabled. Can only be changed during reset. 00: lane 0 01: lanes 0 and 1 11: all lanes

Table 45: MIPI TX Video Signals (Interface to FPGA Fabric)

Signal	Direction	Clock Domain	Description
VSYNC	Input	PIXEL_CLK	Vertical sync.
HSYNC	Input	PIXEL_CLK	Horizontal sync.
VALID	Input	PIXEL_CLK	Valid signal.
HRES[15:0]	Input	PIXEL_CLK	Horizontal resolution. Can only be changed when VSYNC is low, and should be stable for at least one TX pixel clock cycle before VSYNC goes high.
DATA[63:0]	Input	PIXEL_CLK	Video data; the format depends on the data type. New data arrives on every pixel clock.
TYPE[5:0]	Input	PIXEL_CLK	Video data type. Can only be changed when HSYNC is low, and should be stable for at least one TX pixel clock cycle before HSYNC goes high.
FRAME_MODE	Input	PIXEL_CLK	Selects frame format. ⁽⁵⁾ 0: general frame 1: accurate frame Can only be changed during reset.
VC[1:0]	Input	PIXEL_CLK	Virtual channel (VC). Can only be changed when VSYNC is low, and should be stable at least one TX pixel clock cycle before VSYNC goes high.
ULPS_CLK_ENTER	Input	PIXEL_CLK	Place the clock lane into ULPS mode. Should not be active at the same time as ULPS_CLK_EXIT. Each high pulse should be at least 5 μ s.
ULPS_CLK_EXIT	Input	PIXEL_CLK	Remove clock lane from ULPS mode. Should not be active at the same time as ULPS_CLK_ENTER. Each high pulse should be at least 5 μ s.
ULPS_ENTER[3:0]	Input	PIXEL_CLK	Place the data lane into ULPS mode. Should not be active at the same time as ULPS_EXIT[3:0]. Each high pulse should be at least 5 μ s.
ULPS_EXIT[3:0]	Input	PIXEL_CLK	Remove the data lane from ULPS mode. Should not be active at the same time as ULPS_ENTER[3:0]. Each high pulse should be at least 5 μ s.

Table 46: MIPI TX Pads

Pad	Direction	Description
TXDP[4:0]	Output	MIPI transceiver P pads.
TXDN[4:0]	Output	MIPI transceiver N pads.

⁽⁵⁾ Refer to the MIPI Camera Serial Interface 2 (MIPI CSI-2) for more information about frame formats.

Table 47: MIPI TX Settings in Efinity® Interface Designer

Tab	Parameter	Choices	Notes
Base	PHY Bandwidth (Mbps)	80.00 - 1500.00	Choose one of the possible PHY bandwidth values.
	Frequency (reference clock)	6, 12, 19.2, 25, 26, 27, 38.4, or 52 MHz	Reference clock frequency.
	Enable Continuous PHY Clocking	On or Off	Turns continuous clock mode on or off.
Control	Escape Clock Pin Name	User defined	
	Invert Escape Clock	On or Off	
	Pixel Clock Pin Name	User defined	
	Invert Pixel Clock	On or Off	
Lane Mapping	TXD0, TXD1, TXD2, TXD3, TXD4	clk, data0, data1, data2, or data3	Map the physical lane to a clock or data lane.
Clock Timer			
Timing	T _{CLK-POST} T _{CLK-TRAIL} T _{CLK-PREPARE} T _{CLK-ZERO}	Varies depending on the PHY frequency	Changes the MIPI transmitter timing parameters per the DPHY specification. Refer to D-PHY Timing Parameters on page 92.
	Escape Clock Frequency (MHz)	User defined	Specify a number between 11 and 20 MHz.
	T _{CLK-PRE}	Varies depending on the escape clock frequency	Changes the MIPI transmitter timing parameters per the DPHY specification. Refer to D-PHY Timing Parameters on page 92.
	Data Timer		
	T _{HS-PREPARE} T _{HS-ZERO} T _{HS-PTRAIL}	Varies depending on the PHY frequency	Changes the MIPI transmitter timing parameters per the DPHY specification. Refer to D-PHY Timing Parameters on page 92.

MIPI TX Video Data TYPE[5:0] Settings

The video data type can only be changed when HSYNC is low.

Table 48: MIPI TX TYPE[5:0]

TYPE[5:0]	Data Type	Pixel Data Bits per Pixel Clock	Pixels per Clock	Bits per Pixel	Maximum Data Pixels per Line
0x20	RGB444	48	4	12	2,880
0x21	RGB555	60	4	15	2,880
0x22	RGB565	64	4	16	2,880
0x23	RGB666	54	3	18	2,556
0x24	RGB888	48	2	24	1,920
0x28	RAW6	60	10	6	7,680
0x29	RAW7	56	8	7	6,576
0x2A	RAW8	64	8	8	5,760
0x2B	RAW10	60	6	10	4,608

TYPE[5:0]	Data Type	Pixel Data Bits per Pixel Clock	Pixels per Clock	Bits per Pixel	Maximum Data Pixels per Line
0x2C	RAW12	60	5	12	3,840
0x2D	RAW14	56	4	14	3,288
0x18	YUV420 8 bit	Odd line: 64 Even line: 64	Odd line: 8 Even line: 4	Odd line: 8 Even line: 8, 24	2,880
0x19	YUV420 10 bit	Odd line: 60 Even line: 40	Odd line: 6 Even line: 2	Odd line: 10 Even line: 10, 30	2,304
0x1A	Legacy YUV420 8 bit	48	4	8, 16	3,840
0x1C	YUV420 8 bit (CSPS)	Odd line: 64 Even line: 64	Odd line: 8 Even line: 4	Odd line: 8 Even line: 8, 24	2,880
0x1D	YUV420 10 bit (CSPS)	Odd line: 60 Even line: 40	Odd line: 6 Even line: 2	Odd line: 10 Even line: 10, 30	2,304
0x1E	YUV422 8 bit	64	4	8, 24	2,880
0x1F	YUV422 10 bit	40	2	10, 30	2,304
0x30 - 37	User defined 8 bit	64	8	8	5,760

MIPI TX Video Data DATA[63:0] Formats

The format depends on the data type. New data arrives on every pixel clock.

Table 49: RAW6 (10 Pixels per Clock)

63	6059	5453	4847	4241	3635	3029	2423	1817	1211	6 5	0
0	Pixel 10	Pixel 9	Pixel 8	Pixel 6	Pixel 6	Pixel 5	Pixel 4	Pixel 3	Pixel 2	Pixel 1	

Table 50: RAW7 (8 Pixels per Clock)

63	5655	4948	4241	3534	2827	2120	1413	7 6	0
0	Pixel 8	Pixel 7	Pixel 6	Pixel 5	Pixel 4	Pixel 3	Pixel 2	Pixel 1	

Table 51: RAW8 and User Defined (8 Pixels per Clock)

63	5453	4847	4039	3231	2423	1615	8 7	0
Pixel 8	Pixel 7	Pixel 6	Pixel 5	Pixel 4	Pixel 3	Pixel 2	Pixel 1	

Table 52: RAW10 (6 Pixels per Clock)

63	6059	5049	4039	3029	2019	109	0
0	Pixel 6	Pixel 5	Pixel 4	Pixel 3	Pixel 2	Pixel 1	

Table 53: RAW12 (5 Pixels per Clock)

63	6059	4847	3635	2423	1211	0
0	Pixel 5	Pixel 4	Pixel 3	Pixel 2	Pixel 1	

Table 54: RAW14 (4 Pixels per Clock)

63	5655	4241	2827	1413	0
0	Pixel 4	Pixel 3	Pixel 2	Pixel 1	

Table 55: RGB444 (4 Pixels per Clock)

63	4847	3635	2423	1211	0
0	Pixel 4	Pixel 3	Pixel 2	Pixel 1	

Table 56: RGB555 (4 Pixels per Clock)

63	6059	4544	3029	1514	0
0	Pixel 4	Pixel 3	Pixel 2	Pixel 1	

Table 57: RGB565 (4 Pixels per Clock)

63	4847	3231	1615	0
Pixel 4	Pixel 3	Pixel 2	Pixel 1	

Table 58: RGB666 (3 Pixels per Clock)

63	5453	3635	1817	0
0	Pixel 3	Pixel 2	Pixel 1	

Table 59: RGB888 (2 Pixels per Clock)

63	4847		2423		0
0	Pixel 2		Pixel 1		

Table 60: YUV420 8 bit Odd Line (8 Pixels per Clock), Even Line (4 Pixels per Clock)

63	5655	4847	4039	3231	2423	1615	8 7	0
Odd Lines								
Pixel 8 Y8	Pixel 7 Y7	Pixel 6 Y6	Pixel 5 Y5	Pixel 4 Y4	Pixel 3 Y3	Pixel 2 Y2	Pixel 1 Y1	
Even Lines								
Pixel 4	Pixel 3			Pixel 2	Pixel 1			
Y4	V3	Y3	U3	Y2	V1	Y1	U1	

Table 61: Legacy YUV420 8 bit (4 Pixels per Clock)

63	4847		4039	3231	2423	1615	8 7	0
0	Pixel 4	Pixel 3		Pixel 2	Pixel 1			
Odd Lines	Y4	Y3	U3	Y2	Y1	U1		
Even Lines	Y4	Y3	V3	Y2	Y1	V1		

Table 62: YUV420 10 bit Odd Line (6 Pixels per Clock) Even Line (2 Pixels per Clock)

63	6059	5049	4039	3029	2019	109	0
Odd Lines							
0	Pixel 6 Y6	Pixel 5 Y5	Pixel 4 Y4	Pixel 3 Y3	Pixel 2 Y2	Pixel 1 Y1	
Even Lines							
0			Pixel 2	Pixel 1			
			Y2	V1	Y1	U1	

Table 63: YUV422 8 bit (4 Pixels per Clock)

63	5655	4847	4039	3231	2423	1615	8 7	0
Pixel 4	Pixel 3			Pixel 2	Pixel 1			
Y4	V3	Y3	U3	Y2	V1	Y1	U1	

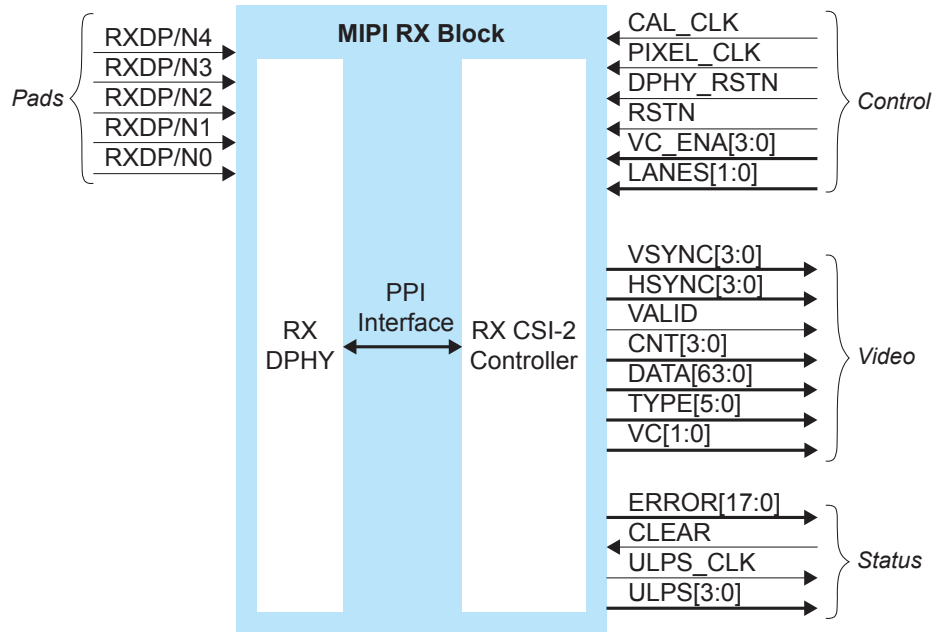
Table 64: YUV422 10 bit (2 Pixels per Clock)

63	4039		3029	2019	109	0
0		Pixel 2	Pixel 1			
		Y2	V1	Y1	U1	

MIPI RX

The MIPI RX is a receiver interface that translates HSSI signals from the board to video data in the Trion® core. Five high-speed differential pin pairs (one clock, four data), each of which represent a lane, connect to the board. Control, video, and status signals connect from the MIPI interface to the core.

Figure 28: MIPI RX x4 Block Diagram



The control signals determine the clocking, how many transceiver lanes are used, and how many virtual channels are enabled. All control signals are required except the two reset signals. The reset signals are optional, however, you must use both signals or neither.

The video signals send the decoded video data to the core. All video signals must fully support the MIPI standard.

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

Figure 29: MIPI RX Interface Block Diagram

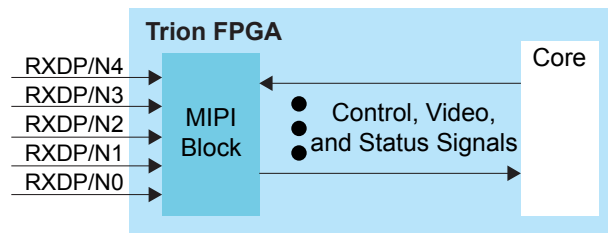


Table 65: MIPI RX Control Signals (Interface to FPGA Fabric)

Signal	Direction	Clock Domain	Notes
CAL_CLK	Input	N/A	Used for D-PHY calibration; must be between 80 and 120 MHz.
PIXEL_CLK	Input	N/A	Clock used for transferring data to the core from the MIPI RX block. The frequency based on the number of lanes and video format. Refer to Understanding the RX and TX Pixel Clock on page 93.
DPHY_RSTN	Input	N/A	(Optional) Reset for the D-PHY logic, active low. Must be used if RSTN is used. See MIPI Reset Timing .
RSTN	Input	N/A	(Optional) Reset for the CSI-2 controller logic, active low. Must be used if DPHY_RSTN is used. See MIPI Reset Timing .
VC_ENA[3:0]	Input	PIXEL_CLK	Enables different VC channels by setting their index high.
LANES[1:0]	Input	PIXEL_CLK	Determines the number of lanes enabled: 00: lane 0 01: lanes 0 and 1 11: all lanes Can only be set during reset.

Table 66: MIPI RX Video Signals (Interface to FPGA Fabric)

Signal	Direction	Clock Domain	Notes
VSYNC[3:0]	Output	PIXEL_CLK	Vsync bus. High if vsync is active for this VC.
HSYNC[3:0]	Output	PIXEL_CLK	Hsync bus. High if hsync is active for this VC
VALID	Output	PIXEL_CLK	Valid signal.
CNT[3:0]	Output	PIXEL_CLK	Number of valid pixels contained in the pixel data.
DATA[63:0]	Output	PIXEL_CLK	Video data, format depends on data type. New data every pixel clock.
TYPE[5:0]	Output	PIXEL_CLK	Video data type.
VC[1:0]	Output	PIXEL_CLK	Virtual channel (VC).

Table 67: MIPI RX Status Signals (Interface to FPGA Fabric)

Signal	Direction	Signal Interface	Clock Domain	Notes
ERROR[17:0]	Output	IN	PIXEL_CLK	Error bus register. Refer to Table 68: MIPI RX Error Signals (ERROR[17:0]) on page 87 for details.
CLEAR	Input	OUT	PIXEL_CLK	Reset the error registers.
ULPS_CLK	Output	IN	PIXEL_CLK	High when the clock lane is in the Ultra-Low-Power State (ULPS).
ULPS[3:0]	Output	IN	PIXEL_CLK	High when the lane is in the ULPS mode.

Table 68: MIPI RX Error Signals (ERROR[17:0])

Bit	Name	Description
0	ERR_ESC	Escape Entry Error. Asserted when an unrecognized escape entry command is received.
1	CRC_ERROR_VC0	CRC Error VC0. Set to 1 when a checksum error occurs.
2	CRC_ERROR_VC1	CRC Error VC1. Set to 1 when a checksum error occurs.
3	CRC_ERROR_VC2	CRC Error VC2. Set to 1 when a checksum error occurs.
4	CRC_ERROR_VC3	CRC Error VC3. Set to 1 when a checksum error occurs.
5	HS_RX_TIMEOUT_ERR	HS RX Timeout Error. The protocol should time out when no EoT is received within a certain period in HS RX mode.
6	ECC_1BIT_ERROR	ECC Single Bit Error. Set to 1 when there is a single bit error.
7	ECC_2BIT_ERROR	ECC 2 Bit Error. Set to 1 if there is a 2 bit error in the packet.
8	ECCBIT_ERROR	ECC Error. Asserted when an error exists in the ECC.
9	ECC_NO_ERROR	ECC No Error. Asserted when an ECC is computed with a result zero. This bit is high when the receiver is receiving data correctly.
10	FRAME_SYNC_ERROR	Frame Sync Error. Asserted when a frame end is not paired with a frame start on the same virtual channel.
11	INVLD_PKT_LEN	Invalid Packet Length. Set to 1 if there is an invalid packet length.
12	INVLD_VC	Invalid VC ID. Set to 1 if there is an invalid CSI VC ID.
13	INVALID_DATA_TYPE	Invalid Data Type. Set to 1 if the received data is invalid.
14	ERR_FRAME	Error In Frame. Asserted when VSYNC END received when CRC error is present in the data packet.
15	CONTROL_ERR	Control Error. Asserted when an incorrect line state sequence is detected.
16	SOT_ERR	Start-of-Transmission (SoT) Error. Corrupted high-speed SoT leader sequence while proper synchronization can still be achieved.
17	SOT_SYNC_ERR	SoT Synchronization Error. Corrupted high-speed SoT leader sequence while proper synchronization cannot be expected.



Note: If error report is all logic low, there is an EOT or a contention error. Check the physical connection of MIPI lanes or adjust the EXIT and TRAIL parameters according to the MIPI Utility.

Table 69: MIPI RX Pads

Pad	Direction	Description
RXDP[4:0]	Input	MIPI transceiver P pads.
RXDN[4:0]	Input	MIPI transceiver N pads.

Table 70: MIPI RX Settings in Efinity® Interface Designer

Tab	Parameter	Choices	Notes
Control	DPHY Calibration Clock Pin Name	User defined	
	Invert DPHY Calibration Clock	On or Off	
	Pixel Clock Pin Name	User defined	
	Invert Pixel Clock	On or Off	
Status	Enable Status	On or Off	Indicate whether you want to use the status pins.
Lane Mapping	RXD0, RXD1, RXD2, RXD3, RXD4	clk, data0, data1, data2, or data3	Map the physical lane to a clock or data lane.
	Swap P&N Pin	On or Off	Reverse the P and N pins for the physical lane.
Timing	Calibration Clock Freq (MHz)	User defined	Specify a number between 80 and 120 MHz.
	Clock Timer (T _{CLK-SETTLE})	40 - 2,590 ns	Changes the MIPI receiver timing parameters per the DPHY specification. Refer to D-PHY Timing Parameters on page 92.
	Data Timer (T _{HS-SETTLE})	40 - 2,590 ns	Changes the MIPI receiver timing parameters per the DPHY specification. Refer to D-PHY Timing Parameters on page 92.

MIPI RX Video Data TYPE[5:0] Settings

The video data type can only be changed when HSYNC is low.

Table 71: MIPI RX TYPE[5:0]

TYPE[5:0]	Data Type	Pixel Data Bits per Pixel Clock	Pixels per Clock	Bits per Pixel	Maximum Data Pixels per Line
0x20	RGB444	48	4	12	2,880
0x21	RGB555	60	4	15	2,880
0x22	RGB565	64	4	16	2,880
0x23	RGB666	54	3	18	2,556
0x24	RGB888	48	2	24	1,920
0x28	RAW6	48	8	6	7,680
0x29	RAW7	56	8	7	6,576
0x2A	RAW8	64	8	8	5,760
0x2B	RAW10	40	4	10	4,608
0x2C	RAW12	48	4	12	3,840
0x2D	RAW14	56	4	14	3,288

TYPE[5:0]	Data Type	Pixel Data Bits per Pixel Clock	Pixels per Clock	Bits per Pixel	Maximum Data Pixels per Line
0x18	YUV420 8 bit	Odd line: 64 Even line: 64	Odd line: 8 Even line: 4	Odd line: 8 Even line: 8, 24	2,880
0x19	YUV420 10 bit	Odd line: 40 Even line: 40	Odd line: 4 Even line: 2	Odd line: 10 Even line: 10, 30	2,304
0x1A	Legacy YUV420 8 bit	48	4	8, 16	3,840
0x1C	YUV420 8 bit (CSPS)	Odd line: 64 Even line: 64	Odd line: 8 Even line: 4	Odd line: 8 Even line: 8, 24	2,880
0x1D	YUV420 10 bit (CSPS)	Odd line: 40 Even line: 40	Odd line: 4 Even line: 2	Odd line: 10 Even line: 10, 30	2,304
0x1E	YUV422 8 bit	64	4	8, 24	2,880
0x1F	YUV422 10 bit	40	2	10, 30	2,304
0x30 - 37	User defined 8 bit	64	8	8	5,760

MIPI RX Video Data DATA[63:0] Formats

The format depends on the data type. New data arrives on every pixel clock.

Table 72: RAW6 (8 Pixels per Clock)

63	4847		4241	3635	3029	2423	1817	1211	6 5	0
0		Pixel 8	Pixel 7	Pixel 6	Pixel 5	Pixel 4	Pixel 3	Pixel 2	Pixel 1	

Table 73: RAW7 (8 Pixels per Clock)

63	5655	4948	4241	3534	2827	2120	1413	7 6	0
0	Pixel 8	Pixel 7	Pixel 6	Pixel 5	Pixel 4	Pixel 3	Pixel 2	Pixel 1	

Table 74: RAW8 (8 Pixels per Clock)

63	5655	4847	4039	3231	2423	1615	8 7	0
Pixel 8	Pixel 7	Pixel 6	Pixel 5	Pixel 4	Pixel 3	Pixel 2	Pixel 1	

Table 75: RAW10 (4 Pixels per Clock)

63	4039		3029	2019	109	0
0		Pixel 4	Pixel 3	Pixel 2	Pixel 1	

Table 76: RAW12 (4 Pixels per Clock)

63	4847		3635	2423	1211	0
0	Pixel 4	Pixel 3	Pixel 2	Pixel 1		

Table 77: RAW14 (4 Pixels per Clock)

63	5655	4241	2827	1413	0
0	Pixel 4	Pixel 3	Pixel 2	Pixel 1	

Table 78: RGB444 (4 Pixels per Clock)

63	4847		3635	2423	1211	0
0	Pixel 4	Pixel 3	Pixel 2	Pixel 1		

Table 79: RGB555 (4 Pixels per Clock)

63	6059	4544	3029	1514	0
Pixel 4	Pixel 3	Pixel 2	Pixel 1		

Table 80: RGB565 (4 Pixels per Clock)

63	4847		3231	1615	0
Pixel 4	Pixel 3	Pixel 2	Pixel 1		

Table 81: RGB666 (3 Pixels per Clock)

63	5453	3635	1817	0
0	Pixel 3	Pixel 2	Pixel 1	

Table 82: RGB888 (2 Pixels per Clock)

63	4847		2423		0
0	Pixel 2		Pixel 1		

Table 83: YUV420 8 bit Odd Line (8 Pixels per Clock), Even Line (4 Pixels per Clock)

63	5655	4847	4039	3231	2423	1615	8 7	0
Odd Lines								
Pixel 8 Y8	Pixel 7 Y7	Pixel 6 Y6	Pixel 5 Y5	Pixel 4 Y4	Pixel 3 Y3	Pixel 2 Y2	Pixel 1 Y1	
Even Lines								
Pixel 4	Pixel 3			Pixel 2	Pixel 1			
Y4	U3	Y3	V3	Y2	U1	Y1	V1	

Table 84: Legacy YUV420 8 bit (4 Pixels per Clock)

63	4847		4039	3231	2423	1615	8 7	0
0	Pixel 4	Pixel 3		Pixel 2	Pixel 1			
Odd Lines	Y4	U3	Y3	Y2	U1	Y1		
Even Lines	Y4	V3	Y3	Y2	V1	Y1		

Table 85: YUV420 10 bit Odd Line (4 Pixels per Clock), Even Line (2 Pixels per Clock)

63	4039		3029	2019	109	0
Odd Lines						
0	Pixel 4 Y4	Pixel 3 Y3	Pixel 2 Y2	Pixel 1 Y1		
Even Lines						
0	Pixel 1 V1	Pixel 2 Y2	Pixel 1 U1		Y1	

Table 86: YUV422 8 bit (4 Pixels per Clock)

63	5655	4847	4039	3231	2423	1615	8 7	0
Pixel 4	Pixel 3			Pixel 2	Pixel 1			
Y4	V3	Y3	U3	Y2	V1	Y1	U1	

Table 87: YUV422 10 bit (2 Pixels per Clock)

63	4039		3029	2019	109	0
0	Pixel 1 V1	Pixel 2 Y2	Pixel 1 U1		Y1	

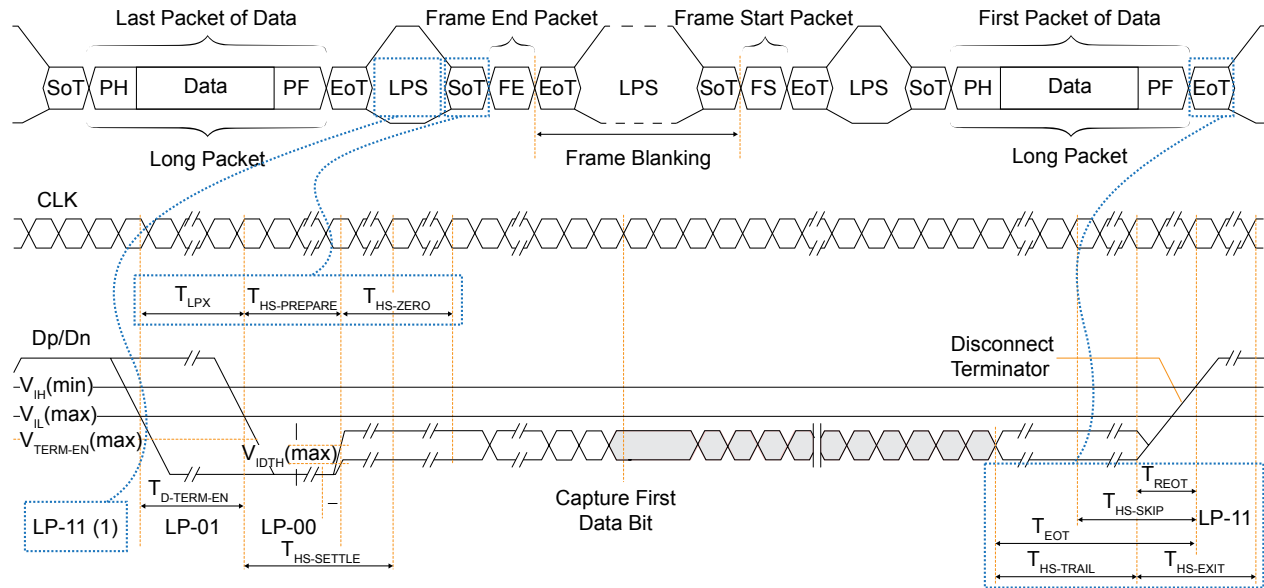
D-PHY Timing Parameters

During CSI-2 data transmission, the MIPI D-PHY alternates between low power mode and high-speed mode. The D-PHY specification defines timing parameters to facilitate the correct hand-shaking between the MIPI TX and MIPI RX during mode transitions.

You set the timing parameters to correspond to the specifications of your hardware in the Efinity® Interface Designer.

- *RX parameters*— $T_{CLK-SETTLE}$, $T_{HS-SETTLE}$ (see [Table 65: MIPI RX Control Signals \(Interface to FPGA Fabric\)](#) on page 86)
- *TX parameters*— $T_{CLK-POST}$, $T_{CLK-TRAIL}$, $T_{CLK-PREPARE}$, $T_{CLK-ZERO}$, $T_{CLK-PRE}$, $T_{HS-PREPARE}$, $T_{HS-ZERO}$, $T_{HS-TRAIL}$ (see [Table 47: MIPI TX Settings in Efinity Interface Designer](#) on page 81)

Figure 30: High-Speed Data Transmission in Bursts Waveform



Note:

1. To enter high-speed mode, the D-PHY goes through states LP-11, LP-01, and LP-00. The D-PHY generates LP-11 to exit high-speed mode.

Figure 31: Switching the Clock Lane between Clock Transmission and Low Power Mode Waveform

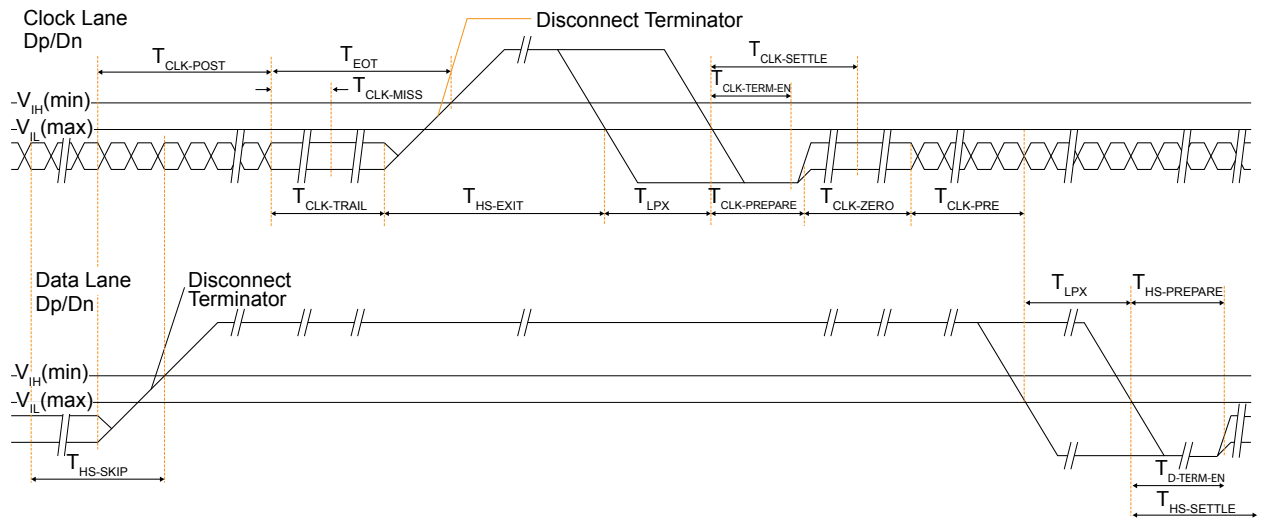


Table 88: D-PHY Timing Specifications

Parameter	Description	Min	Typ	Max	Unit
$T_{CLK-POST}$	Time that the transmitter continues to send HS clock after the last associated Data Lane has transitioned to LP Mode. Interval is defined as the period from the end of $T_{HS-TRAIL}$ to the beginning of $T_{CLK-TRAIL}$.	$60\text{ ns} + 52*UI$	-	-	ns
$T_{CLK-PRE}$	Time that the HS clock shall be driven by the transmitter prior to any associated Data Lane beginning the transition from LP to HS mode.	8	-	-	UI
$T_{CLK-PREPARE}$	Time that the transmitter drives the Clock Lane LP-00 Line state immediately before the HS-0 Line state starting the HS transmission.	38	-	95	ns
$T_{CLK-SETTLE}$	Time interval during which the HS receiver should ignore any Clock Lane HS transitions, starting from the beginning of $T_{CLK-PREPARE}$.	95	-	300	ns
$T_{CLK-TRAIL}$	Time that the transmitter drives the HS-0 state after the last payload clock bit of a HS transmission burst.	60	-	-	ns
$T_{CLK-PREPARE} + T_{CLK-ZERO}$	$T_{CLK-PREPARE}$ + time that the transmitter drives the HS-0 state prior to starting the Clock.	300	-	-	ns
$T_{HS-PREPARE}$	Time that the transmitter drives the Data Lane LP-00 Line state immediately before the HS-0 Line state starting the HS transmission	$40\text{ ns} + 4*UI$	-	$85\text{ ns} + 6*UI$	ns
$T_{HS-SETTLE}$	Time interval during which the HS receiver shall ignore any Data Lane HS transitions, starting from the beginning of $T_{HS-PREPARE}$. The HS receiver shall ignore any Data Lane transitions before the minimum value, and the HS receiver shall respond to any Data Lane transitions after the maximum value.	$85\text{ ns} + 6*UI$	-	$145\text{ ns} + 10*UI$	ns
$T_{HS-TRAIL}$	Time that the transmitter drives the flipped differential state after last payload data bit of a HS transmission burst.	$\max(n*8*UI, 60\text{ ns} + n*4*UI)^{(6)}$	-	-	ns
T_{LPX}	Transmitted length of any Low-Power state period	50	-	-	ns
$T_{HS-PREPARE} + T_{HS-ZERO}$	$T_{HS-PREPARE}$ + time that the transmitter drives the HS-0 state prior to transmitting the Sync sequence.	$145\text{ ns} + 10*UI$	-	-	ns

Understanding the RX and TX Pixel Clock

In a MIPI system, the pixel clock is the clock used to transfer the video data to or from the MIPI controller. This document calls it the *system pixel clock*. The system pixel clock is related to the video resolution. If you are using a standard monitor, you can simply look up the

⁽⁶⁾ Where $n = 1$ in Forward-direction HS mode and $n = 4$ for Reverse-direction HS mode.

clock specification in the SMPTE/CEA or VESA standards. Alternatively, you can calculate the clock you need.

Generally speaking, you calculate the system pixel clock frequency using the following equation:

$$\text{Pixel Clock Frequency} = \text{Total Horizontal Samples} \times \text{Total Vertical Lines} \times \text{Refresh Rate}^{(7)}$$

where the *Total Horizontal Samples* and *Total Vertical Lines* include the blanking period.

In the Interface Designer, the MIPI TX and RX interfaces also have RX and TX pixel clocks. These clocks are not the same as the system pixel clock, however, they are related. These pixel clocks take into account both the system pixel clock and the video data type.

The RX and TX pixel clocks must be equal to or faster than the system pixel clock divided by the number of pixels processed by the MIPI interface each clock cycle. The number of pixels processed per clock depends on the video data type.

For example, if the system pixel clock is running at 150 MHz and using the RGB444 RX data type, which processes 4 pixels per clock, the RX pixel clock must be at least 37.5 MHz.



Note: Efinix provides a MIPI utility that you can use to calculate the TX and RX pixel clock frequencies that work with the video data type. Refer to [Using the MIPI Utility](#).

Video Data Type

The video data type includes the color mode (RAW, RGB, and YUV) and the data format, which together determine the amount of video transmitted every pixel clock (that is, the bandwidth). The overall system bandwidth is simply the system pixel clock times the number of bits of video data transferred each clock cycle.

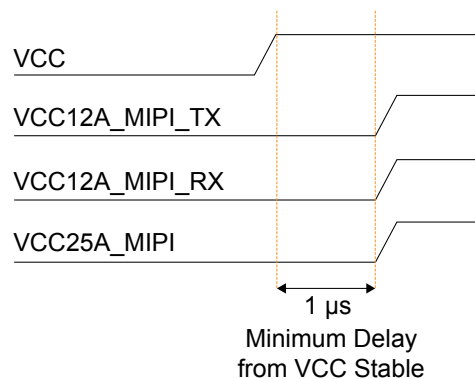
Power Up Sequence

The MIPI block has four power supplies:

- VCC—Digital supply voltage
- VCC25A_MIPI—2.5 V analog supply voltage
- VCC12A_MIPI_TX—1.2 V analog voltage supply to the MIPI TX
- VCC12A_MIPI_RX—1.2 V analog voltage supplies to the MIPI RX

When powering up the FPGA, VCC should power up and stabilize before the MIPI analog supplies power up.

Figure 32: MIPI Power Up Sequence



⁽⁷⁾ The *Refresh Rate* is also called the frame rate or vertical frequency.

Using the MIPI Block

You use the MIPI TX and RX blocks to configure the hard MIPI interface. You can add up to two MIPI TX and up to two MIPI RX blocks.



Note: The Efinity® software v2019.1 and later supports the MIPI interface block.

MIPI TX

The MIPI TX Block Editor organizes the settings into tabs. Most settings are simply naming the MIPI signals for your design and specifying timing parameters.

Table 89: Base Tab Settings

The Base tab is where you set the instance name, choose a resource, and make clock settings.

Setting	Choices	Notes
Instance Name	User defined	Specify an instance name.
MIPI TX Resource	MIPI_TX0 or MIPI_TX1	Choose which resource to instantiate.
PHY Bandwidth (Mbps)	80 to 1500 Mbps	Choose the speed at which to run the D-PHY.
Reference Clock Frequency (MHz)	6.00, 12.00, 19.00, 25.00, 26.00, 27.00, 38.00, 52.00	The software automatically assigns a GPIO resource for the reference clock. You must add a GPIO block in clock output mode and assign it this resource. Both MIPI resources share the same reference clock. Therefore, you must use the same frequency setting for both instances.
Enable Continuous PHY Clocking	On or Off	

- **Control Tab**—Specify names for control signals. The DPHY and CSI-2 resets are optional. Leave **DPHY Reset Pin Name** and **CSI-2 Reset Pin Name** blank if you do not want to use the resets. You must use both resets or neither.
- **Video Tab**—Specify names for video signals.
- **Video - ULPS Mode Tab**—The MIPI TX block supports the Ultra-Low Power State (ULPS) for the data and clock lanes. In this mode, the lane goes to sleep and does not transmit data. The MIPI block consumes almost no power in ULPS mode. If you want to use ULPS mode, specify the names of the ULPS enter and exit signals for the clock and data lanes.



Note: The MIPI CSI-2 controller automatically uses escape mode and low-power data transmission for low-power operation. You do not need to enable these modes.

- **Lane Mapping Tab**—The MIPI TX block supports 4 data lanes and 1 clock lane. In this tab you choose which lane to associate with the MIPI pad. Select a name from the drop-down list. The lane mapping must be unique, which the software enforces.
- **Timing Tab**—In this tab you specify the timing parameters for the clock and data timers.

MIPI RX

The MIPI TX Block Editor organizes the settings into tabs. Most settings are simply naming the MIPI signals for your design and specifying timing parameters.

- **Base Tab**—Specify an instance name and choose a MIPI RX resource. Choose **MIPI_RX0** or **MIPI_RX1**.
- **Control Tab**—Specify names for control signals.
- **Video Tab**—Specify names for video signals.
- **Status Tab**—You can choose to enable status signals. Turn on **Enable Status** and specify the signal names.
- **Lane Mapping Tab**—The MIPI TR block supports 4 data lanes and 1 clock lane. In this tab you choose which lane to associate with the MIPI pad. Select a name from the drop-down list. The lane mapping must be unique, which the software enforces.
- **Timing Tab**—In this tab you specify the timing parameters for the clock and data timers, as well as the calibration clock frequency.

Design Check: MIPI 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_rule_resource \(error\)](#)

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

[mipi_rule_inst_name \(error\)](#)

Message	Instance name is empty
To fix	Specify the the instance name.

[mipi_rule_empty_pins \(error\)](#)

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

[mipi_rule_reset_pins \(error\)](#)

Message	Either none or all reset pins non-empty
To fix	There are multiple reset pins defined. Either use all specified reset pin names or don't use them at all.

[mipi_rule_refclk \(error\)](#)

Message	The gpio resource <name> for REF_CLK is not configured
To fix	Add a GPIO block in input mode, set the Connection Type to mipi_clkin, and assign it to the correct resource.
Message	Reference clock resource <name> is not configured as mipi_clkin connection
To fix	Set the GPIO resource to Input mode with mipi_clkin connection type.
Message	Reference clock resource <name> is not configured as input
To fix	Set the GPIO resource to Input mode with mipi_clkin connection type.
Message	Reference clock resource <name> input configuration invalid
To fix	Set the GPIO resource to Input mode with alternate connection type.

[mipi_rule_phy_freq \(error\)](#)

Message	PHY frequency <freq>MHz is out of range. Min=<freq> Max=<freq>
To fix	Set the frequency within the valid range.

[mipi_rule_refclk_freq \(error\)](#)

Message	Reference clock frequency <freq>MHz is invalid
To fix	The selected frequency has to be from the valid list.

[mipi_rule_lane_swap \(error\)](#)

Message	Found <num> unused lane
To fix	All lanes must be assigned.
Message	Found <num> occurrence of duplicated logical_lane: <duplicated lanes>
To fix	Assign unique lanes.

[mipi_rule_refclk_freq_consistent \(error\)](#)

Message	Mismatch reference clock frequency: <freq> vs <freq> of <instance>
To fix	All instances of MIPI TX must have the same reference clock frequency settings if they are sourced from the same reference clock resource.

[mipi_rule_calib_clk_freq \(error\)](#)

Message	Calibration clock frequency <freq>MHz is out of range, Min=<freq> Max=<freq>
To fix	Set the frequency within the valid range.

[mipi_rule_esc_clk_freq \(error\)](#)

Message	Escape clock frequency <freq>MHz is out of range, Min=<freq> Max=<freq>
To fix	Set the frequency within the valid range.

[mipi_rule_invalid_pins \(error\)](#)

Message	Invalid pin names found: <pin description names>
To fix	Update the pin names such that it only contains alphanumeric characters with dash and underscore.

PLL Interface

Contents:

- **About the Simple PLL Interface**
- **Using the PLL Block**
- **Design Check: Simple PLL Messages**

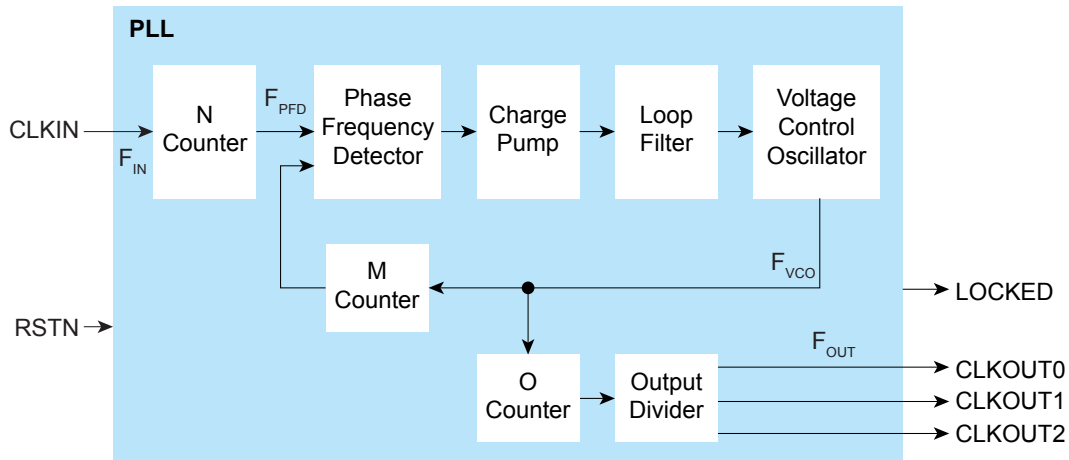
The following sections describe the simple PLL and how to use it. Refer to **Interface Blocks** on page 8 to find out if your FPGA supports the simple PLL.

Note: At startup, Efinix recommends that you hold the PLL in reset until the PLL's reference clock source is stable.

About the Simple PLL Interface

T4 and T8 FPGAs in F49 and F81 packages have 1 PLL to synthesize clock frequencies. The PLL's reference clock input comes from a dedicated GPIO's alternate input pin. The PLL consists of a pre-divider counter (N counter), a feedback multiplier counter (M counter), post-divider counter (O counter), and an output divider per clock output.

Figure 33: Trion PLL Block Diagram



The counter settings define the PLL output frequency:	where:
$F_{PFD} = F_{IN} / N$	F_{VCO} is the voltage control oscillator frequency
$F_{VCO} = F_{PFD} \times M$	F_{OUT} is the output clock frequency
$F_{OUT} = F_{VCO} / (O \times \text{Output divider})$	F_{IN} is the reference clock frequency
	F_{PFD} is the phase frequency detector input frequency

Note: The reference clock must be between 10 and 50 MHz.
 The PFD input must be between 10 and 50 MHz.
 The VCO frequency must be between 500 and 1,200 MHz.

Unlike other Trion® FPGAs, the Trion PLL output locks on the *negative* clock edge (not the positive edge). When you are using two or more clock outputs, they are aligned on the falling edge. If the core register receiving the clock is positive edge triggered, Efinix recommends inverting the clock outputs so they are correctly edge aligned.

Figure 34: PLL Output Aligned with Negative Edge

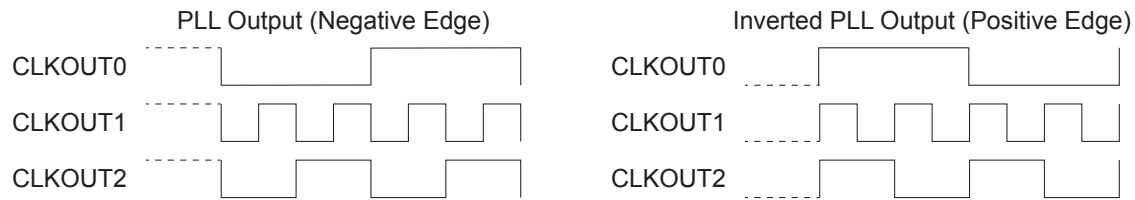


Table 90: PLL Pins

Port	Direction	Description
CLKIN	Input	Reference clock. This port is also a GPIO pin; the GPIO pins' alternate function is configured as a reference clock.
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.
CLKOUT0 CLKOUT1 CLKOUT2	Output	PLL output. The designer can route these signals as input clocks to the core's GCLK network.
LOCKED ⁽⁸⁾	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. This signal is analog asynchronous.

Table 91: PLL Settings

Configure these settings in the Efinity® Interface Designer.

Setting	Allowed Values	Notes
N counter	1 - 15 (integer)	Pre-divider
M counter	1 - 255 (integer)	Multiplier
O counter	1, 2, 4, 8	Post-divider
Output divider	2, 4, 8, 16, 32, 64, 128, 256	Output divider per output

Using the PLL Block

The Trion T4 and T8 FPGAs have a simple PLL. This PLL is referenced as PLL_V1 in the Python API.



Note: In this mode, a specific GPIO block with an alternate connection type must generate the reference clock(s). The PLL Block Summary shows the resource name to use.

1. Add a GPIO block.

⁽⁸⁾ The circuitry that generates the lock signal relies on a reference clock edge to transition the lock signal. A sudden removal of the reference clock will result in there being no positive clock edge with which to change the lock state from 1 back to 0. Therefore, the lock signal will remain on 1.

2. Enter the instance name.
3. Choose **input** as the mode.
4. Choose **pll_clkln** as the connection type.
5. In the Resource Assigner, assign it to the resource shown in the summary.

You can set up the PLL using the PLL Clock Calculator or manually using the Block Editor.

- 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.
- Click the **Manual Configuration** tab to configure the PLL manually.

Using the PLL Clock Calculator

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

- Turn signals on or off by clicking the icons (gray x or green arrow) next to the signal.
- Specify the signal names.
- Choose the clock phase.

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.

Set up the PLL Manually

Make these settings in the **Manual Configuration** tab.

- Specify general settings such as the PLL resource, reset pin name, and locked pin name.
- Under VCO Frequency, specify the reference clock frequency, multiplier, and pre-divider. The software calculates and displays the resulting VCO frequency. If the VCO is outside of the allowed range, it displays in red.
- Under PLL Frequency, choose the post divider. The software calculates and displays the PLL frequency.
- The simple PLL has three output clocks. Enable the output clocks you want to use, and specify the clock name and output divider. The software calculates and displays the resulting output frequency.

Design Check: Simple PLL 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.

[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_resource \(error\)](#)

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

[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_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.

[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_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_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_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_output_divider \(error\)](#)

Message	Output divider for <clock name> is invalid. Valid values are between 1-256
To fix	Choose a value between 1 and 256.

[pll_rule_output_number \(error\)](#)

Message	Output number for <clk name> is invalid. It must be between 0 to <int>
To fix	Make sure that the number is within specified range.

[pll_rule_clkin_driver \(error\)](#)

Message	ClockIn resource <gpio resource> is not configured as input
To fix	Set the GPIO resource that drives the reference input clock to input mode.
Message	The GPIO resource <gpio resource> for CLKIN is not configured
To fix	Configure the GPIO resource that drives the reference input clock.
Message	ClockIn resource <gpio resource> is not configured as pll_clkin connection
To fix	Set the GPIO resource that drives the reference input clock to input mode with pll_clkin connection type.

[pll_rule_instance_count \(error\)](#)

Message	There can only be one PLL instance
To fix	Only create the instance based on the number of available resources in the device.

[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_invalid_pins \(error\)](#)

Message	Invalid pin names found: <Pin description names>
To fix	Update the pin names. Valid characters are alphanumeric characters with dash and underscore only.

[pll_rule_oc_cascade \(warning\)](#)

Message	(O * C == 1), PLL cascading is not supported with possible PLL clkout: <clock names>
To fix	Update the output clock divider and post-VCO divider of the PLL.

Advanced PLL Interface

Contents:

- [About the Advanced PLL Interface](#)
- [Using the PLL Block](#)
- [Design Check: Advanced PLL Messages](#)

The following sections describe the advanced PLL and how to use it. Refer to the [Package/Interface Support Matrix](#) on page 9 to find out if your FPGA supports the advanced PLL.



Note: At startup, Efinix recommends that you hold the PLL in reset until the PLL's reference clock source is stable.

Similarly, Efinix recommends resetting all cascaded PLLs at startup. Hold the first PLL in reset until the PLL's reference clock source is stable. Hold the cascaded PLLs in reset until the previous PLL is locked.

Cascaded PLLs do not need a 50% duty cycle on the reference clock. However, the clock needs to meet the PLL minimum pulse width as specified in the data sheet.

About the Advanced PLL Interface

Trion FPGAs have PLLs to synthesize clock frequencies. The number of PLLs depends on the FPGA and package.



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

You can use the PLL to compensate for clock skew/delay via external or internal feedback to meet timing requirements in advanced application. 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.)

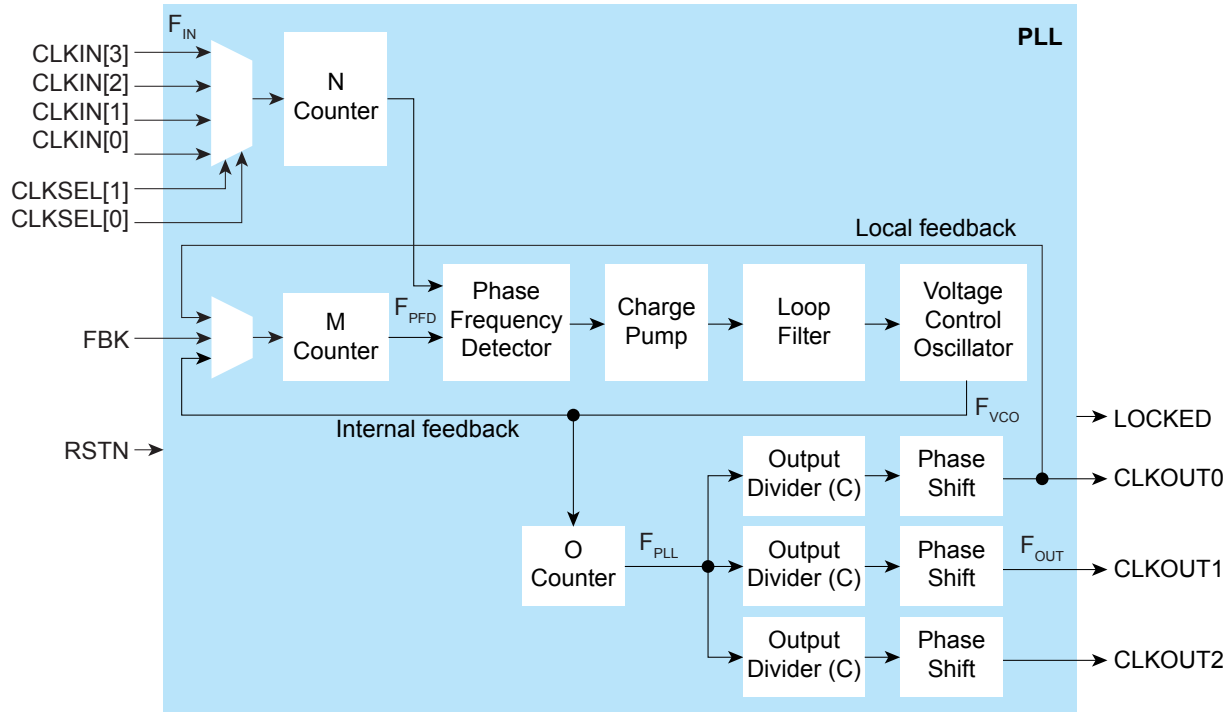
Depending on the package, one or more of the PLLs can use an LVDS RX buffer to input its reference clock.

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



Note: Refer to [Interface Floorplans](#) on page 121 for the location of the PLLs on the die.

Figure 35: Advanced PLL Block Diagram



The counter settings define the PLL output frequency:

Internal Feedback Mode	Local and Core Feedback Mode	Where:
$F_{PFD} = F_{IN} / N$ $F_{VCO} = F_{PFD} \times M$ $F_{OUT} = (F_{IN} \times M) / (N \times O \times C)$ $F_{PLL} = F_{VCO} / O$	$F_{PFD} = F_{IN} / N$ $F_{VCO} = (F_{PFD} \times M \times O \times C_{FBK})^{(9)}$ $F_{OUT} = (F_{IN} \times M \times C_{FBK}) / (N \times C)$ $F_{PLL} = F_{VCO} / O$	<p>F_{VCO} is the voltage control oscillator frequency</p> <p>F_{OUT} is the output clock frequency</p> <p>F_{IN} is the reference clock frequency</p> <p>F_{PFD} is the phase frequency detector input frequency</p> <p>F_{PLL} is the post-divider PLL frequency</p> <p>C is the output divider</p> <p>O is the post-divider</p> <p>M is the multiplier</p> <p>N is the pre-divider</p> <p>C_{FBK} is the output divider for CLKOUT0</p>



Note: The reference clock input must be within the values stated in the PLL Timing and AC Characteristic section of your Trion® FPGA Data Sheet.

⁽⁹⁾ (M x O x C_{FBK}) must be ≤ 255.

Figure 36: Advanced PLL Interface Block Diagram

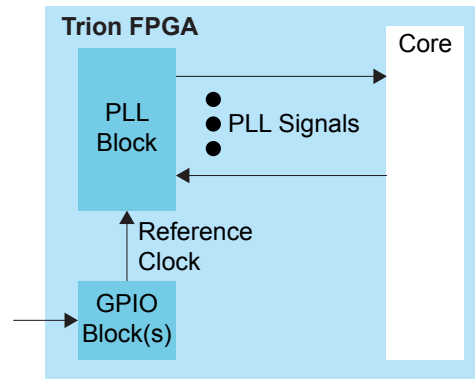


Table 92: Advanced PLL Signals (Interface to FPGA Fabric)

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. De-assert only when the CLKIN signal is stable. 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 feedback mode is not internal .
CLKOUT0 CLKOUT1 CLKOUT2	Output	PLL output. The designer can route these signals as input clocks to the core's GCLK network.
LOCKED ⁽¹⁰⁾	Output	Goes high when PLL achieves lock; goes low when a loss of lock is detected; remains at previous state if the CLKIN goes discontinuous. Connect this signal in your design to monitor the lock status. This signal is not synchronized to any clock and the minimum high or low pulse width of the lock signal may be smaller than the CLKOUT's period.

⁽¹⁰⁾ The circuitry that generates the lock signal relies on a reference clock edge to transition the lock signal. A sudden removal of the reference clock will result in there being no positive clock edge with which to change the lock state from 1 back to 0. Therefore, the lock signal will remain on 1.

Table 93: Advanced PLL Interface Designer Settings - Properties Tab

Parameter	Choices	Notes
Instance Name	User defined	
PLL Resource		The resource listing depends on the FPGA you choose.
Clock Source	External	PLL reference clock comes from external source through the REFCLK pin. Select the available external clock.
	Dynamic	PLL reference clock comes from up to four possible sources (external and core), and are controlled by the clock select bus. Specify the clock selector and core clock names.
	Core	PLL reference clock comes from the core. Specify the core clock pin name.
Automated Clock Calculation		Pressing this button launches the PLL Clock Calculation window. The calculator helps you define PLL settings in an easy-to-use graphical interface.

Table 94: Advanced PLL Interface Designer Settings - Manual Configuration Tab

Parameter	Choices	Notes
Reset Pin Name	User defined	
Locked Pin Name	User defined	
Feedback Mode	Internal	PLL feedback is internal to the PLL resulting in no known phase relationship between clock in and clock out.
	Local	PLL feedback is local to the PLL. Aligns the clock out phase with clock in.
	Core	PLL feedback is from the core. The feedback clock is defined by the COREFBK connection, and must be one of the three PLL output clocks. Aligns the clock out phase with clock in and removes the core clock delay.
Reference clock Frequency (MHz)	User defined	
Multiplier (M)	1 - 255 (integer)	M counter.
Pre Divider (N)	1 - 15 (integer)	N counter.
Post Divider (O)	1, 2, 4, 8	O counter. The value must be 2 or higher if you enable more than 1 PLL output.
Clock 0, Clock 1, Clock 2	On, off	Use these checkboxes to enable or disable clock 0, 1, and 2.
Pin Name	User defined	Specify the pin name for clock 0, 1, or 2.
Divider (C)	1 to 256	Output divider.
Phase Shift (Degree)	0, 45, 90, 135, 180, or 270	Phase shift CLKOUT by 45°, 90°, 135°, 180°, or 270°. The phase shifts are supported with the following C divider settings: C divider = 2 : 90°, 180°, and 270° C divider = 4 : 45°, 90°, and 135° C divider = 6 : 90° To phase shift 225°, select 45° and invert the clock at the destination. To phase shift 315°, select 135° and invert the clock at the destination.
Use as Feedback	On, off	



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

Using the PLL Block

Trion FPGAs (except the T4 and T8 in BGA49 and BGA81 packages) have an advanced PLL. This PLL is referenced as PLL_V2 in the Python API. This block lets you configure the reference clock, feedback options, frequency, and output clocks for the PLL. You can set up the PLL using the PLL Clock Calculator or manually using the Block Editor.

- 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.
- Click the **Manual Configuration** tab to configure the PLL manually.



Note: For FPGAs with DDR, PLL_BR0 is the clock resource for the DDR block. If you are using the DDR block with PLL_BR0, the PLL's CLKOUT0 can only drive the DDR PHY. You can use the PLL's CLKOUT1 and CLKOUT2 while the DDR is using CLKOUT0.

Reference Clock Settings

The PLL has four possible reference clocks. Two of the clocks can come from the FPGA core, and two 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 and clock 1. 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 or LVDS RX block with a **pll_clkln** connection type must generate the reference clock(s). The software displays which resource you need to 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_clkln** 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 (internal, core or local).
- Turn signals on (gray x) or off (green arrow) by clicking the icons next to the signal.
- Specify the signal names.
- Choose the clock phase.
- Choose which clock has feedback (for core feedback mode).

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.

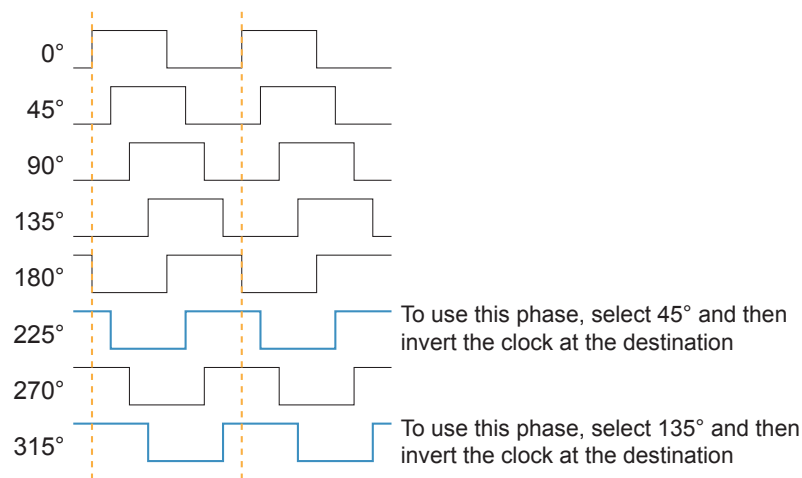
When you are finished using the calculator, click **Finish** to save your settings and close the GUI.

Understanding PLL Phase Shifting

The PLL supports clock phases from 0 to 315 degrees.

- You can select phases 0, 45, 90, 135, 180, and 270 in the Interface Designer directly.
- For phase 225, select 45 in the Interface Designer and then invert the clock at the destination.
- For phase 315, select 135 in the Interface Designer and then invert the clock at the destination.

Figure 37: PLL Clock Phases



Invert the Clock in the Interface Designer

If you connect the PLL clock output to a GPIO and want to invert it at the GPIO, use the Interface Designer GPIO Block Editor to do the inversion:

1. Add the GPIO block.
2. Choose **clkout** as the **Mode**.
3. Turn on the **Inverted** option.

Invert the Clock in Verilog HDL

This Verilog HDL example shows how to invert the clock `clk_45`:

```
always @ (negedge clk_45) begin // the negative edge inverts the clock
    <your code>
end
```

Invert the Clock in VHDL

This VHDL example shows how to invert the clock `clk_45`:

```
process (clk_45)
begin
  -- process
  if falling_edge(clk_45) then // the falling edge inverts the clock
    <your code>
  end if;
end process;
```

Configuring the PLL Manually

If you do not want to use the PLL clock calculator, you can manually configure the PLL using the **Manual Configuration** tab.

Specify the reset and locked pin names. If you do not want to use them, leave the boxes empty.

Choose the feedback mode. The PLL supports these modes:

- **core**—The PLL feedback comes from the FPGA core. The feedback clock must be one of the three PLL output clocks. The output clock and reference clock phases are aligned, and there is no core clock delay. Turn on the **Use as feedback** option for the clock you want to use for feedback.
- **internal**—The PLL feedback is internal to the PLL. The reference clock(s) and output clock(s) have no phase relationship.
- **local**—The PLL uses clock 0. The feedback is local to the PLL and the output clock is aligned with the reference clock.



Note: In local and core feedback modes, the post-divider and output divider of the clock used for feedback affect the VCO frequency.

Specify the reference clock frequency, multiplier, and pre-divider. The software calculates and displays the resulting VCO frequency. If the VCO is outside of the allowed range, it displays in red.

Choose the post divider. The software calculates and displays the PLL frequency.

The advanced PLL has three output clocks. Enable the output clocks you want to use, and specify the pin name, phase shift, and output divider and whether to use the clock as feedback (core mode only). The software calculates and displays the resulting output frequency.

Output Clock Swapping

When you perform a design check or generate constraints, the software tries to find a legal routing for the PLL output clock (`clkout0`, `clkout1`, or `clkout2`). To create a legal routing, it may swap the clock output setting (for example, `clkout0` to `clkout1` or vice versa). When this swap happens, the software updates the PLL block to reflect the change. The original naming is preserved and the result is functionally equivalent.

Design Check: Advanced PLL 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.

[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_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_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_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_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_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_number \(error\)](#)

Message	Output number for <clk name> is invalid. It must be between 0 to <int>
To fix	Make sure that the number is within specified range.

[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_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_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_output_divider (error)

Message	Output divider for <clock name> is invalid.
To fix	Choose a value from the specified list.

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 or LVDS RX 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	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 or LVDS RX 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	Reference clock at <resource> connected to external clock pin {0 1} has not been defined
To fix	The PLL expects to find the resource for the external clock. Add a GPIO or LVDS RX 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} 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	Clockln resource {} is not configured as pll_clkln connection
To fix	The clock set as reference clock for PLL should have its connection type set to pll_clkln.

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.

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	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.

[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	Total non-internal feedback division factor <int> is out of range. Max=255
To fix	The division factor (multiplier * post divider * output divider) must be within the specified range.
Message	Non-internal feedback multiplier <int> is out of range. Max=128
To fix	The multiplier setting must be within the specified range when in non-internal feedback mode.
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_clock \(warning\)](#)

Message	The phase of feedback clock is not 0-degree.
To fix	Set the feedback clock phase to 0 degrees in the PLL Clock Calculator. Efinix recommends a 0 degree phase for feedback clocks.

[pll_rule_phase_shift \(error\)](#)

Message	Output divider of clock <string> has to be 2 when phase shift is any of this values: 180,270
To fix	Set output clock divider to 2 when phase shift is 180 or 270.
Message	Output divider of clock <string> has to be 4 when phase shift is 45 or 135
To fix	Set output clk divider to 4 when phase shift is 45 or 135.
Message	Output divider of clock <string> has to be 2, 4, or 6 when phase shift is 90
To fix	Set output clk divider to 2, 4, or 6 when phase shift is 90.

[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.

[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.
Message	Output frequency <float>MHz driving DDR is out of range. Min=<float>MHz Max=<float>MHz
To fix	The clock driving the DDR input clock needs to be within the range specified. Adjust the parameters to meet that requirement. This is for a clock driving the DDR input clock with the limit depends on the DDR speedgrade configured by user.

[pll_rule_phase_shift_post_divider \(warning\)](#)

Message	Post-divider should be greater than 1 when there is an output clock with non-zero phase shift
To fix	Set the post-divider to more than 1.

[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 out of range. Change the PLL clock calculator settings so that it is in range.

[pll_rule_non_internal_vco_lower_bound \(warning\)](#)

Message	VCO frequency should be greater than 1.6GHz in local/core feedback mode
To fix	Set the parameters in the instance configuration result in a VCO Frequency to greater than 1.6GHz when feedback mode is set to local or core.

[pll_rule_post_div_multi_out_clk \(error\)](#)

Message	Post-divider should be greater than 1 when there are multiple output clocks.
To fix	Set the post-divider to more than 1.

[pll_rule_clkssel_pin \(error\)](#)

Message	Valid characters are alphanumeric characters with dash and underscore only.
To fix	Update the pin name for the clock selector pin when reference clock mode is dynamic

[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.

Oscillator Interface

Contents:

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

Oscillator

T4 and T8 FPGAs have 1 low-frequency oscillator tailored for low-power operation. The oscillator runs at nominal frequency of 10 kHz. Designers can use the oscillator to perform always-on functions with the lowest power possible. Its output clock is available to the GCLK network.

Using the Oscillator Block

To use the oscillator, specify the instance name. Choose the resource and the clock pin name. This oscillator block is available in Trion FPGAs in F49 and F81 packages.



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

Design Check: Oscillator 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.

[osc_rule_inst_name \(error\)](#)

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

[osc_rule_clock_name \(error\)](#)

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

[osc_rule_resource \(error\)](#)

Message	Resource name is empty
To fix	Choose the resource when you create an oscillator block.

Message	Resource is not a valid oscillator device instance
To fix	Choose the correct resource when you create an oscillator block.

[osc_rule_instance_count \(error\)](#)

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

SPI Flash Interface

Contents:

- [About the SPI Flash Memory](#)
- [Using the SPI Flash Interface](#)
- [Design Check: SPI Flash Messages](#)

Trion T13 and T20 FPGAs in the QFP100F3 package have an internal SPI flash memory.

About the SPI Flash Memory

The QFP100F3 packages include a Trion FPGA and a SPI flash memory. The SPI flash memory has a density of 16 Mbits and a clock rate of up to 20 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 QFP100F3 SPI flash.

When a configuration bitstream is stored in the SPI flash and the SPI active hardware connection is properly established, the SPI active configuration can automatically start after the power-up. In QFP100F3 packages you are only required to connect the `SS_N` pin to the `SPI_CS_N` pin for the SPI active configuration to start automatically. This is additional to other required configuration pin settings depending on the configuration mode you select.



Learn more: Refer to the [AN 006: Configuring Trion FPGAs](#) for detailed configuration requirements.

You can also use the SPI flash to store user data during user mode. To read or write the SPI flash during user mode, you must create the SPI flash interface block in the Efinity Interface Designer.



Important: You can only use the internal SPI flash in user mode if no LVDS TX is used.

Figure 38: SPI Flash Memory Block Diagram

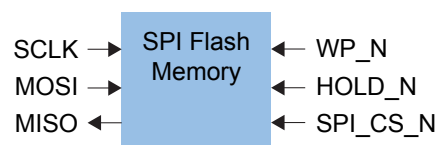


Table 95: SPI Flash Memory Signals (Interface to FPGA Fabric)

SPI Name	Signal	Direction	Description
SCLK	SCLK_OUT	Input	Clock output from FPGA CCK pin to SPI flash memory.
	SCLK_OE	Input	Output enable. Required for multiple controller.
MOSI	MOSI_IN	Output	Required for x2 or x4 data width.
	MOSI_OUT	Input	Data output from FPGA CDI0 to SPI flash memory.
	MOSI_OE	Input	Output enable. Required for x2 data width, x4 data width, or multiple controller.

SPI Name	Signal	Direction	Description
MISO	MISO_IN	Output	Data input to FPGA CDI1 from SPI flash memory.
	MISO_OUT	Input	Required for x2 or x4 data width.
	MISO_OE	Input	Output enable. Required for x2 or x4 data width.
WP_N	WP_N_IN	Output	Required for x4 data width.
	WP_N_OUT	Input	Data output from FPGA CDI2 pin to SPI flash memory.
	WP_N_OE	Input	Output enable. Required for x4 data width or multiple controller.
HOLD_N	HOLD_N_IN	Output	Required for x4 data width.
	HOLD_N_OUT	Input	Data output from FPGA CDI3 pin to SPI flash memory
	HOLD_N_OE	Input	Output enable. Required for x4 data width or multiple controller.
CS_N	CS_N_OUT	Input	Chip select output from FPGA SS_N pin to SPI flash memory.
	CS_N_OE	Input	Output enable. Required for multiple controller.
CLK	CLK	Input	Required for register interface.

Table 96: SPI Flash Interface Designer Settings

Option	Choices	Notes
Instance Name	User defined	
SPI Flash Resource	SPI_FLASH0	Only one resource available.
Enable Register Interface	0, 1	Default: 0 (Disable)
Read/Write Width	x1, x2, x4	Default: x1
Enable Multiple Controller	0, 1	Default: 0 (Disable)
Pin names (various)	User defined	Specify the interface pin names.

Using the SPI Flash Interface

The internal SPI flash is 16 Mbits and can store up to 2 bitstreams and user data.

If you want to use the internal SPI 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. Use the SPI flash interface block to communicate with the SPI flash in user mode.



Important: You do not need to use the SPI flash interface block if you are only using the internal SPI flash for storing bitstreams.

The following table lists the SPI flash interface and the internal SPI flash memory signals with the default resource assignments.



Important: Do not toggle the CCK pin when any LVDS TX is used.

Table 97: SPI Flash Resource Assignments

SPI Flash Interface Signal	SPI Flash Signal	Q100F3 Package Resource Assignment
SCLK	SCK	GPIOL_01_CCK

SPI Flash Interface Signal	SPI Flash Signal	Q100F3 Package Resource Assignment
MOSI	SI	GPIOL_08_CDI0
MISO	SO	GPIOL_09_CDI1
WP_N	WP#	GPIOL_12_CDI2
HOLD_N	HOLD#	GPIOL_13_CDI3
CS_N	-	GPIOL_00_SS_N
-	CS#	SPI_CS_N

Design Check: SPI Flash 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.

[spi_flash_rule_resource \(error\)](#)

Message	Resource name is empty
To fix	Choose a valid resource.
Message	Resource <name> is not a valid SPI Flash device instance
To fix	Choose a valid resource.

[spi_flash_rule_empty_pins \(error\)](#)

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

[spi_flash_rule_invalid_pins \(error\)](#)

Message	Invalid pin names found: <pin description names>
To fix	Update the pin names such that it only contains alphanumeric characters with dash and underscore only.

[spi_flash_rule_cck_pin \(error\)](#)

Message	The CCK pin cannot be used in user mode when LVDS Tx is configured
To fix	You cannot use the CCK pin in user mode when you enable LVDS Tx in Q100F3 packages. You need to choose either one.

[spi_flash_rule_reg_clk_pin \(error\)](#)

Message	Clock Pin Name is required when Register Interface is enabled
To fix	Assign the Clock Pin name if you want to use the Register Interface.

[spi_flash_rule_instance_count \(error\)](#)

Message	There can only be one SPI Flash instance.
To fix	The Trion® FPGAs has only one internal SPI flash. So, only one SPI flash interface can be instantiated.

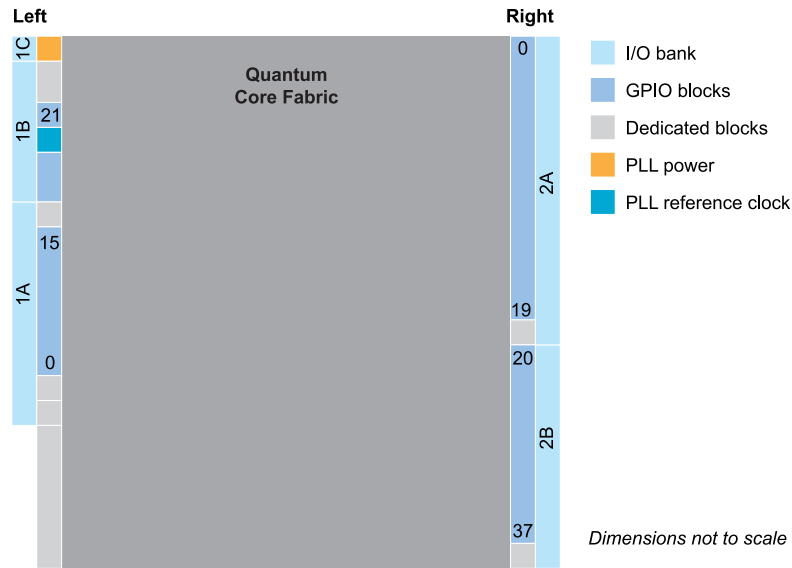
spi_flash_rule_usage (error)

Message	Use a different resource for the following instance as it conflicts with SPI Flash <SPI Flash instance name> resource usage: <Other instance name that has conflict with the SPI Flash resource>
To fix	You need to use a different resource as some of the resources are use for the SPI flash instance.

Interface Floorplans

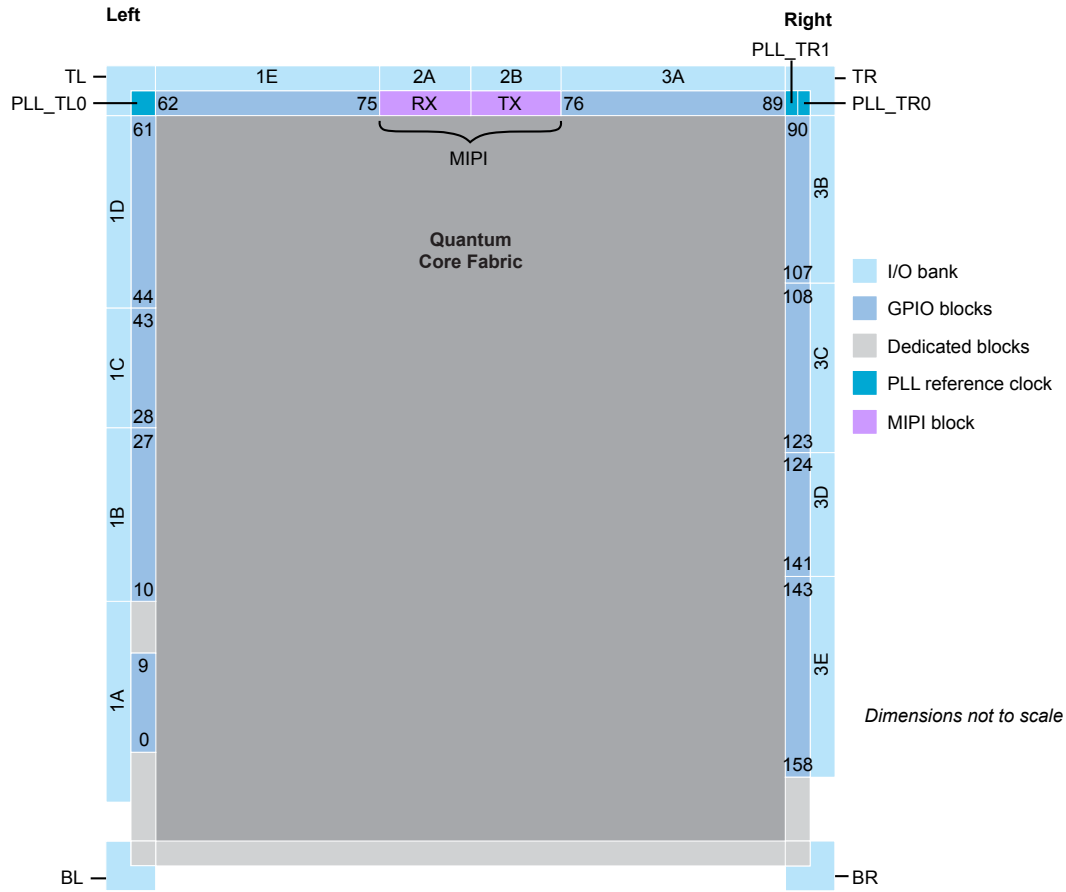
Floorplan Diagram for FPGAs in BGA49 and BGA81 Packages

Figure 39: T4 and T8 FPGAs



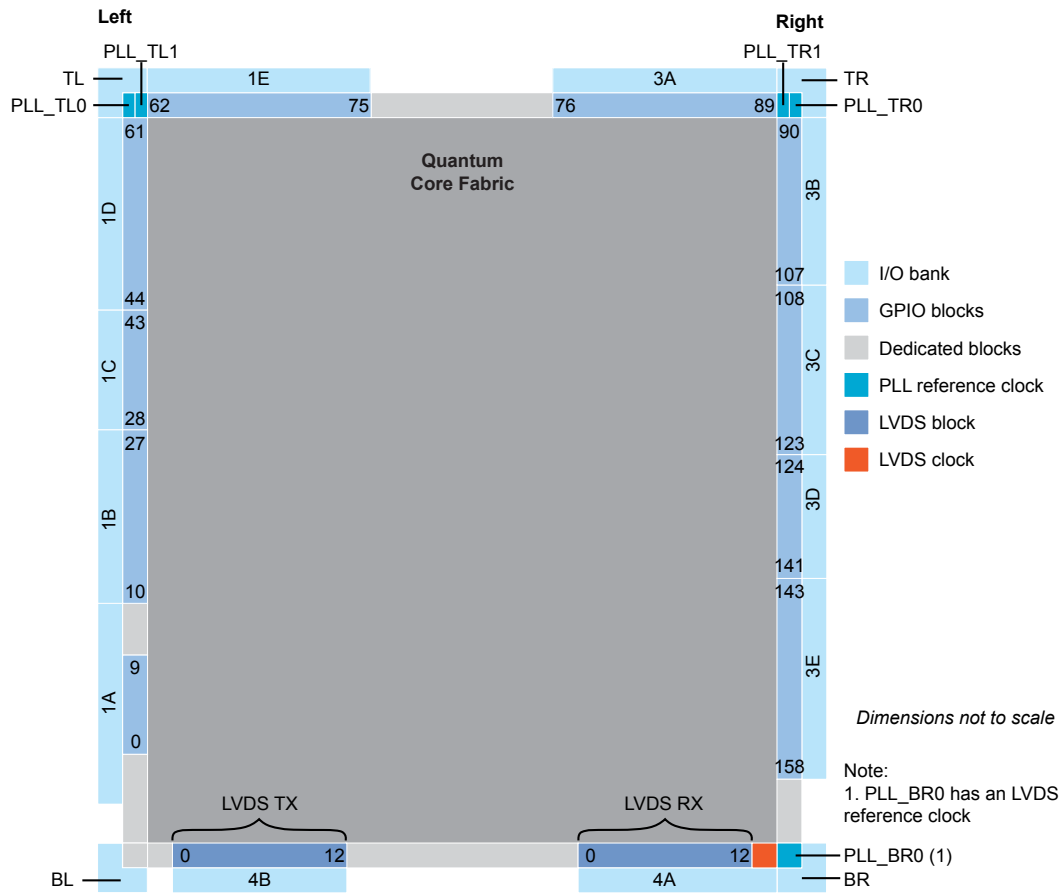
Floorplan Diagram for FPGAs in WLCSP80 Packages

Figure 40: T20 FPGAs



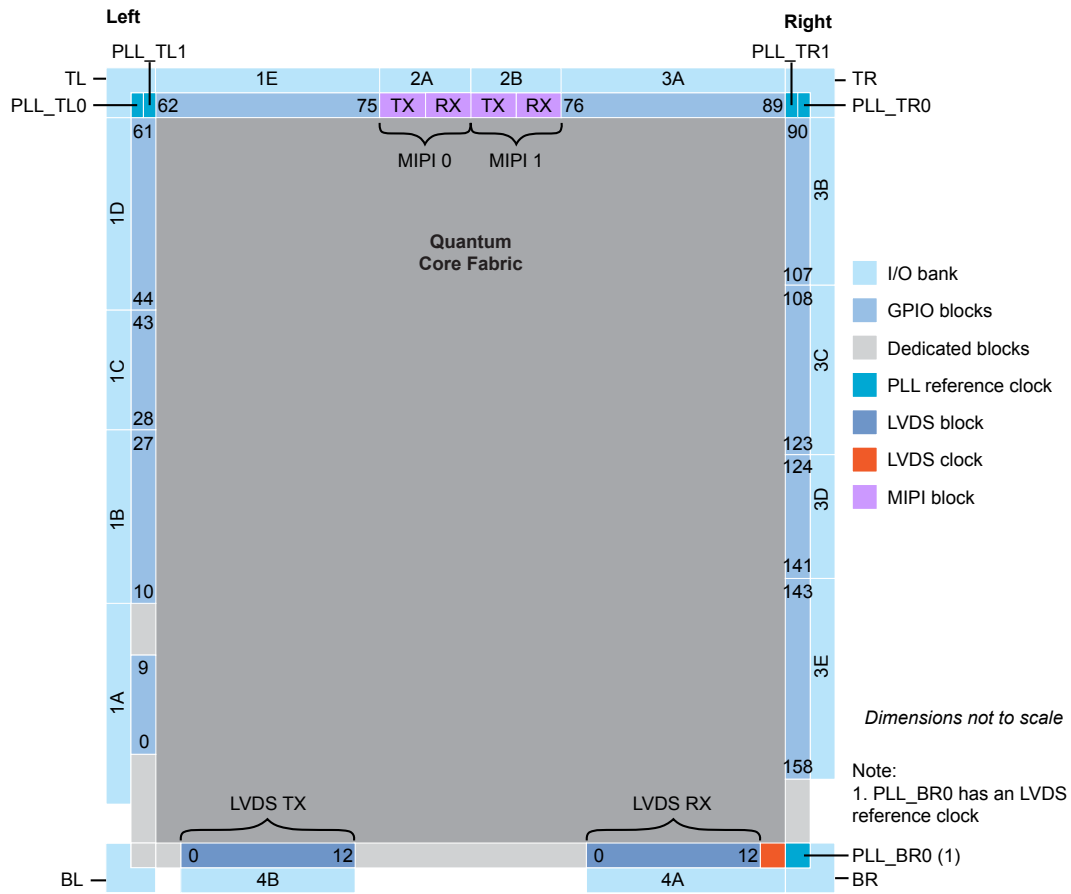
Floorplan Diagram for FPGAs in QFP100F3 and QFP144 Packages

Figure 41: T8, T13, and T20 FPGAs



Floorplan Diagram for FPGAs in BGA169 Packages (with MIPI)

Figure 42: T13 and T20 FPGAs



Floorplan Diagram for FPGAs in BGA256 Packages

Figure 43: T13 and T20 FPGAs

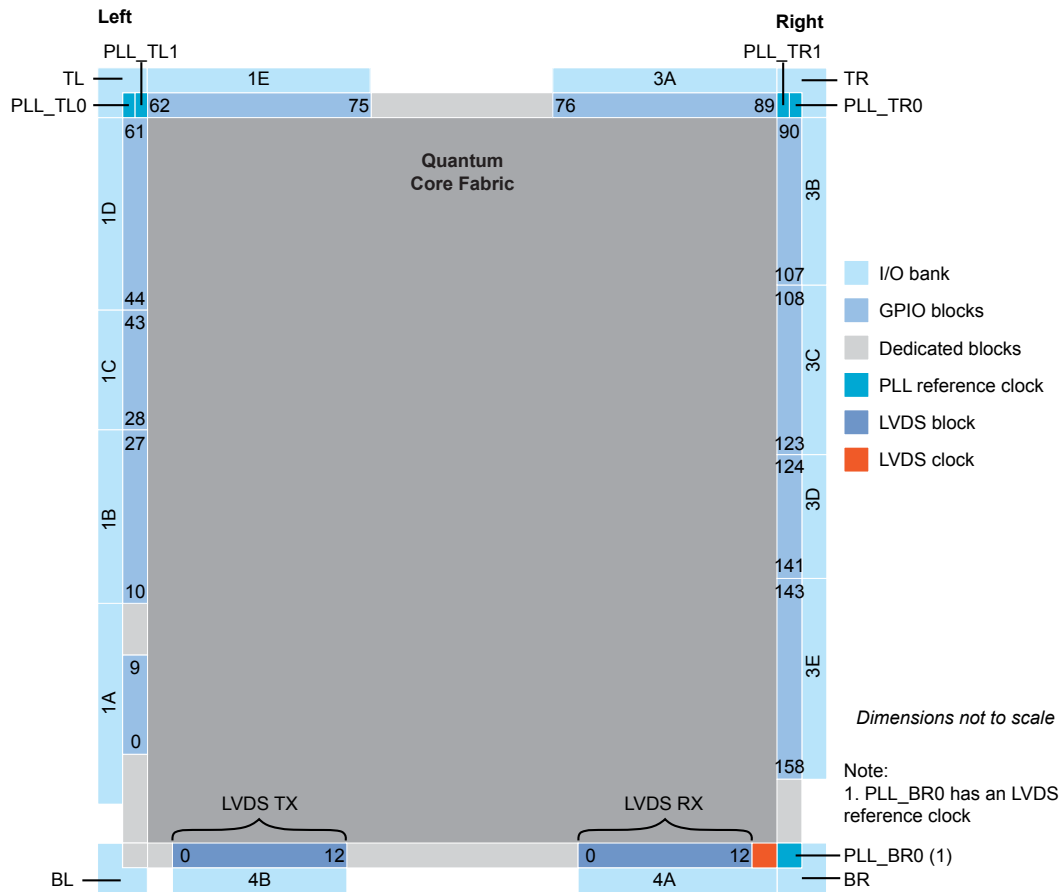
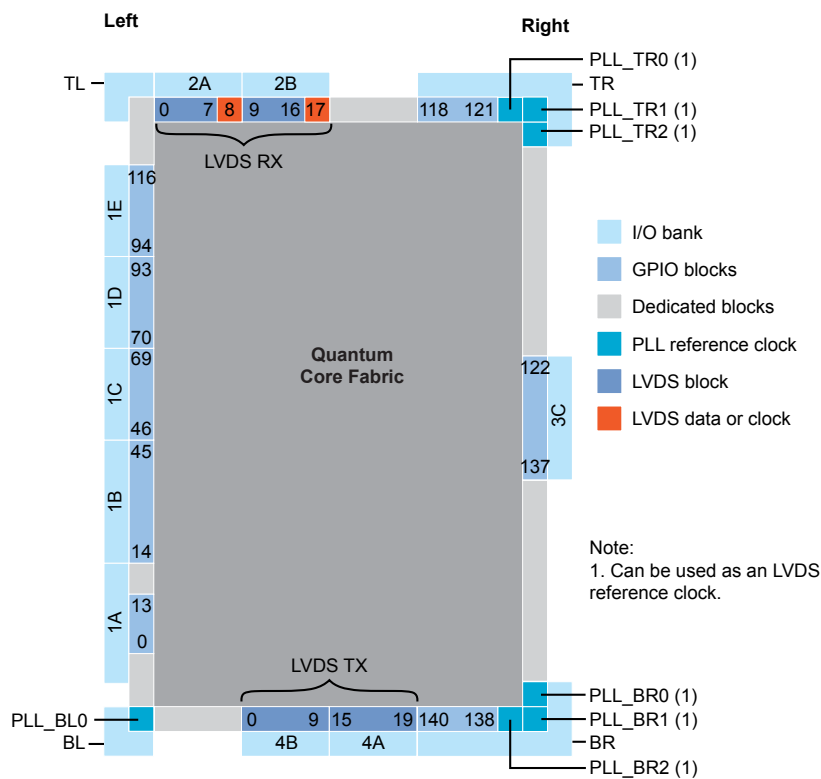


Figure 44: T35 FPGAs



Dimensions not to scale

Floorplan Diagrams for FPGAs in BGA324 Packages (with DDR and MIPI)

Figure 45: T20 and T35 FPGAs

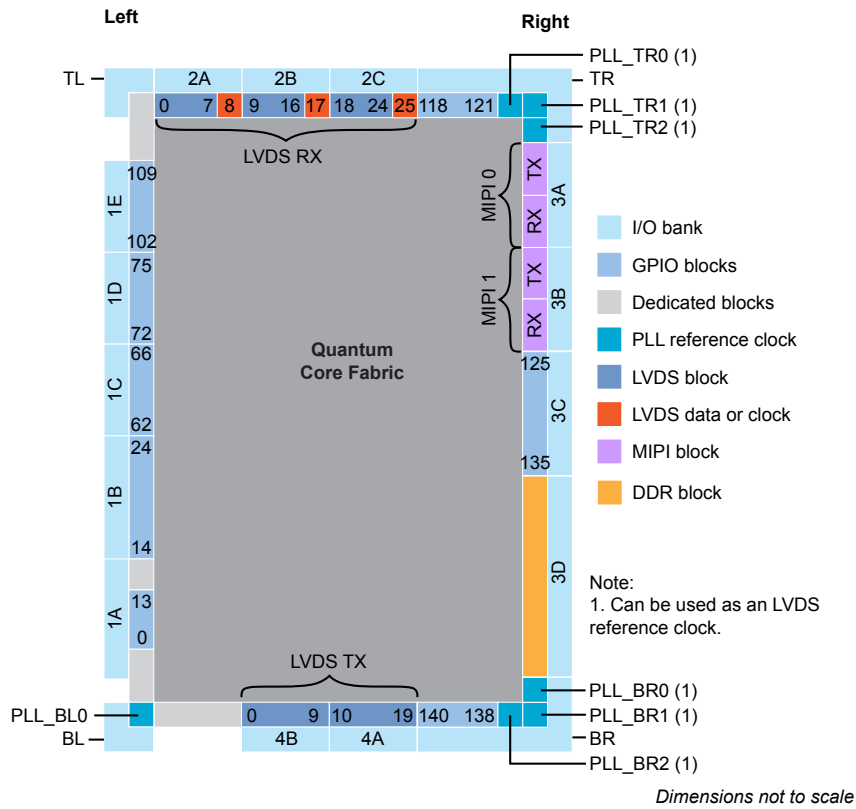
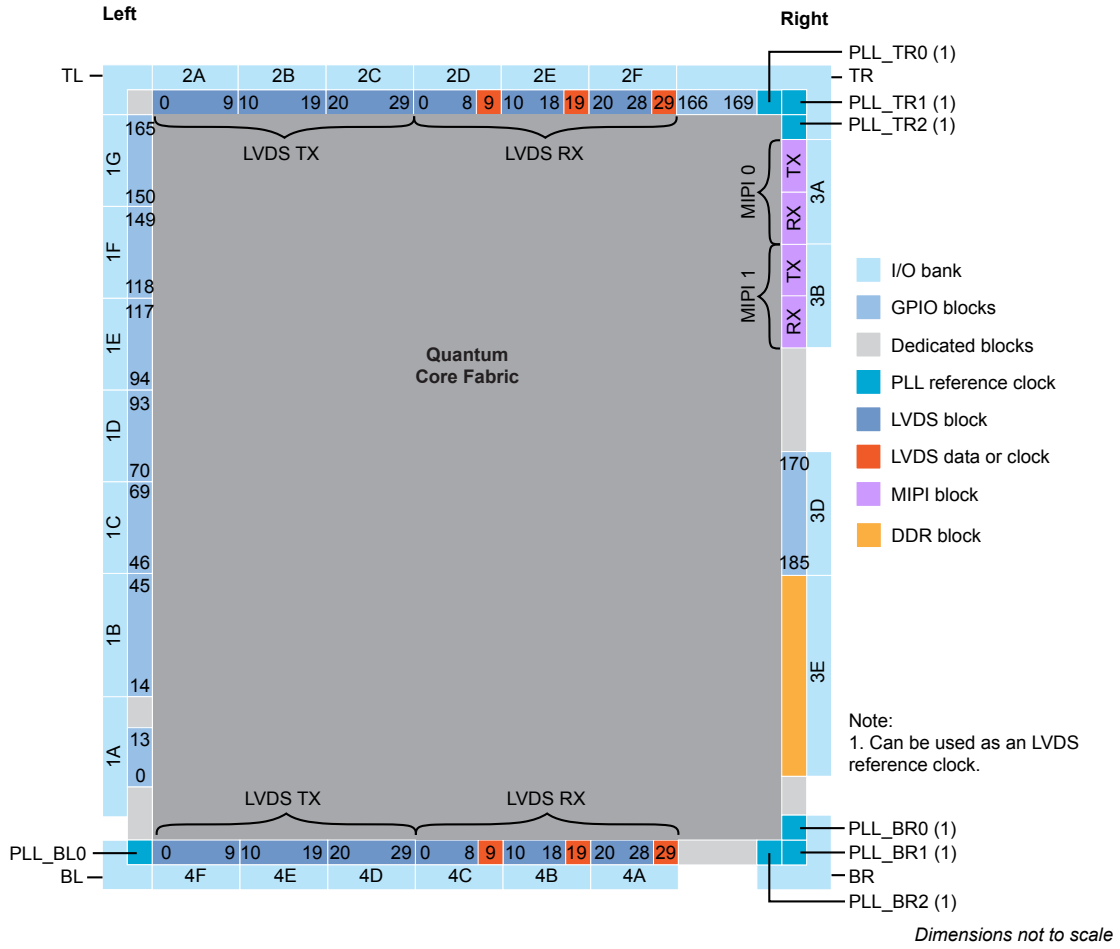
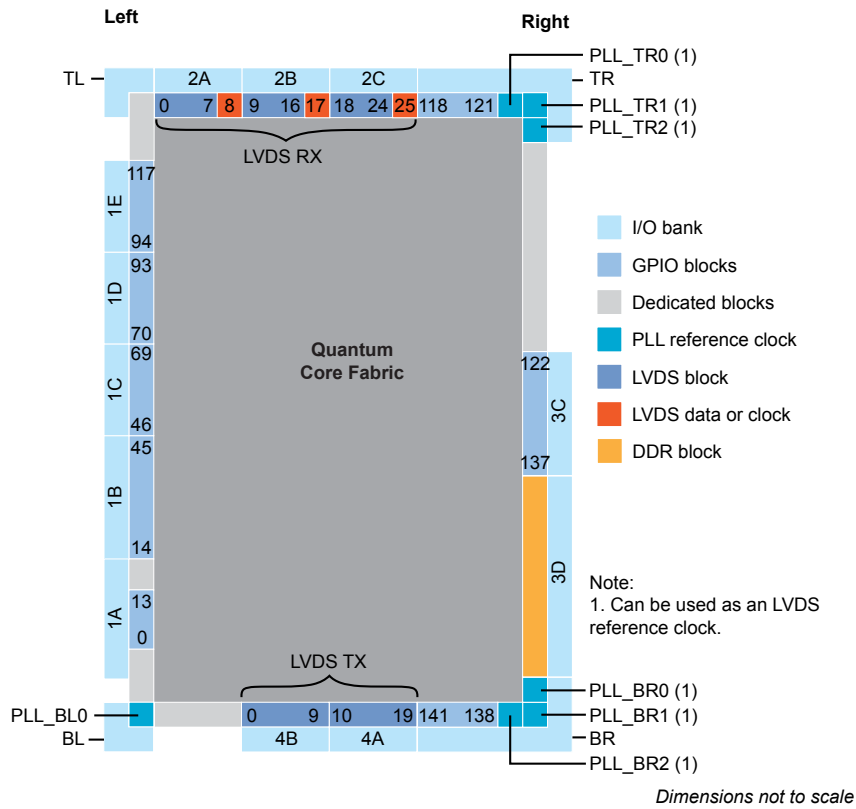


Figure 46: T55, T85, T120



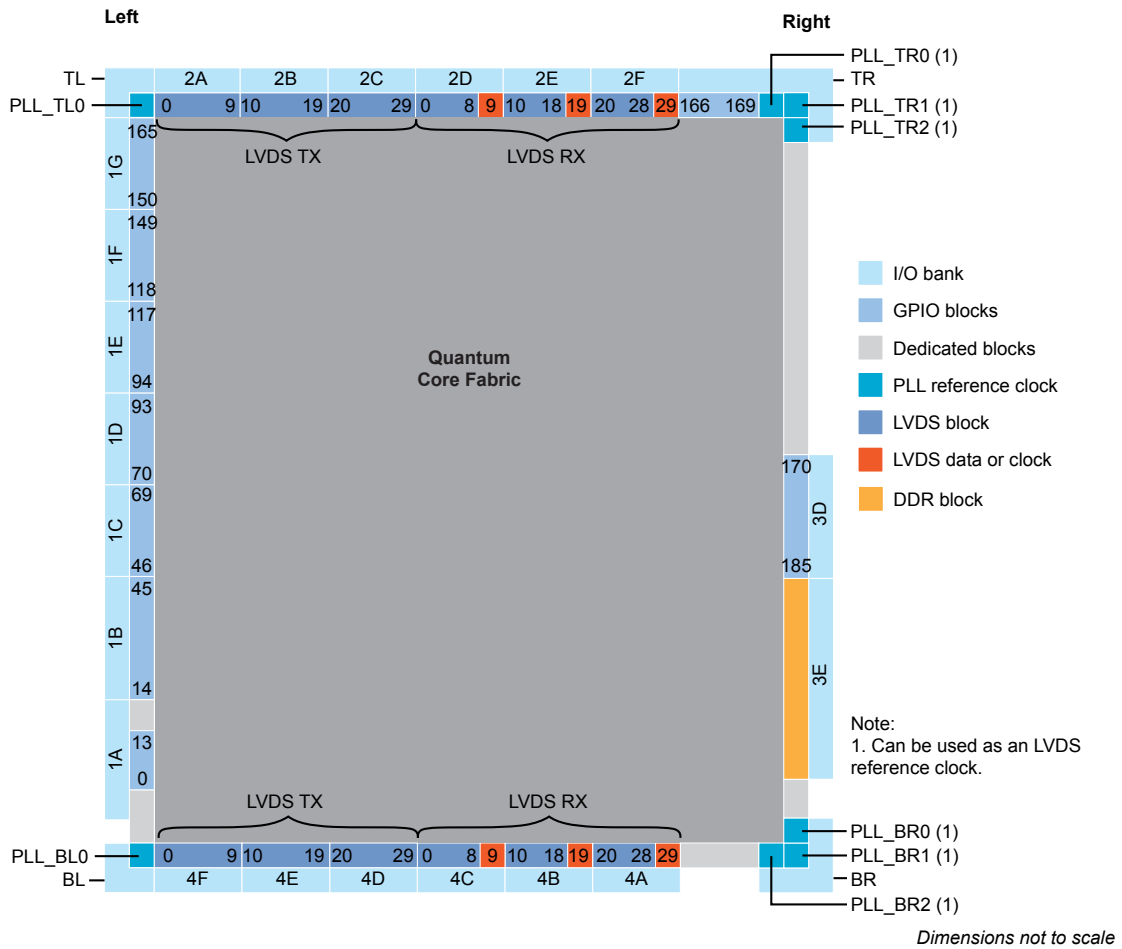
Floorplan Diagram for FPGAs in BGA400 Packages (with DDR)

Figure 47: T20 and T35 FPGAs



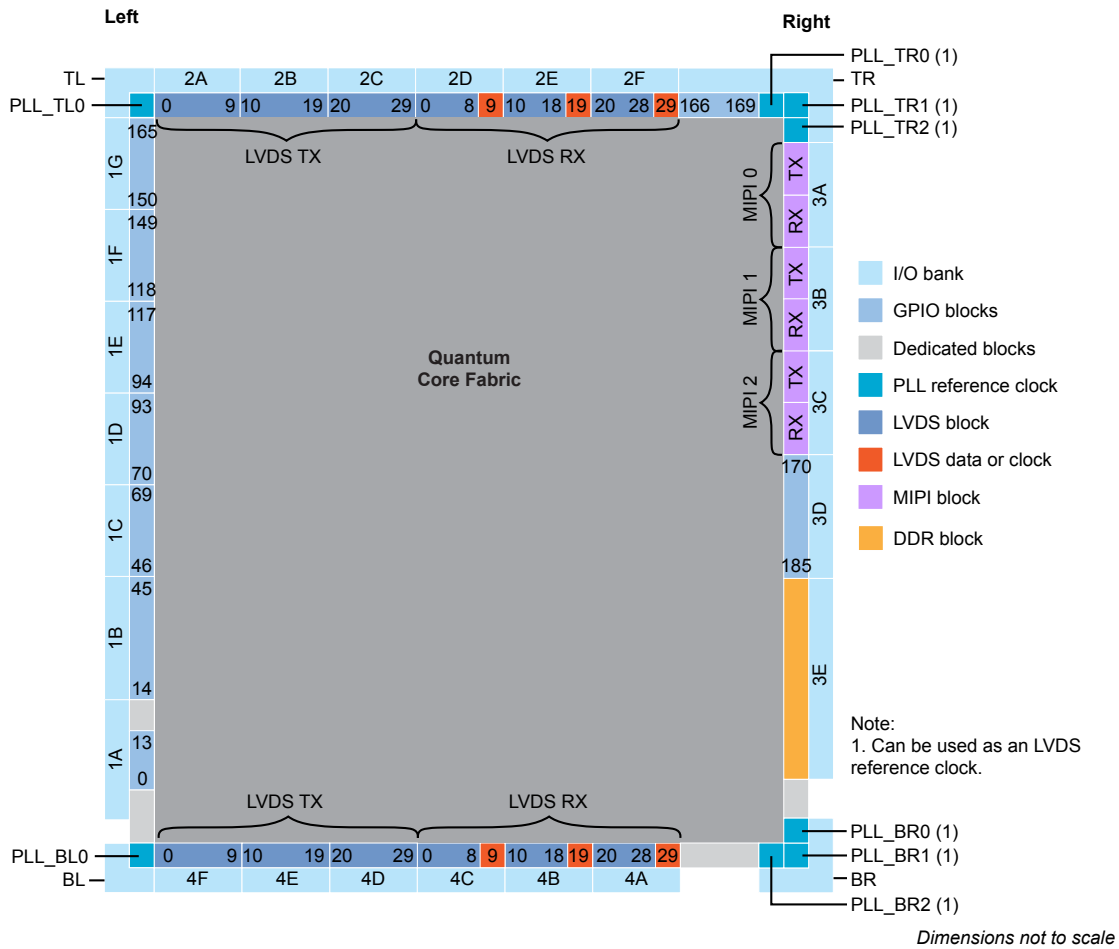
Floorplan Diagram for FPGAs in BGA484 Packages (with DDR)

Figure 48: T55, T85, and T120 FPGAs
















Floorplan Diagram for FPGAs in BGA576 Packages (with DDR and MIPI)

Figure 49: T55, T85, and T120








Icon Reference

Interface Designer Icons

	Interface Designer		Export GPIO Assignments
	Add Block		Import GPIO Assignments
	Create a GPIO bus		Clear Design
	Delete Block		Check Design for Errors
	Show or Hide Block Editor		Export Settings
	Resource Assigner		Generate Constraints File
			Package Planner

Resource Assigner

	Toggle Instance View and Resource View
	Clear Resource
	Clear All Resources
	Show/Hide Filter
	Clear Filter

Revision History

Table 98: Revision History

Date	Version	Description
November 2024	8.7	<p>Added clock messages. (DOC-2071)</p> <p>Added warning message for pll_rule_feedback_clock about 0 degree phases for feedback clocks. (DOC-2036)</p> <p>Added warning message for ddr_rule_memory_settings about the Read/Write Latency setting. (DOC-2049)</p> <p>Updated GPIO interface pin names (IN to I and OUT to O). (DOC-2086)</p>
June 2024	8.6	<p>Added BGA256 package to Table 2: Supported Trion Interface/Package Combinations on page 9. (DOC-1786)</p> <p>Updated the MIPI D-PHY Block Editor parameter name for the D-PHY speed (bandwidth instead of frequency). (DOC-1706)</p> <p>Removed Dynamic Enable Pin Name from Table 42: LVDS RX Settings in Efinity Interface Designer on page 66. This feature is not available in Trion FPGAs. (PT-2355)</p> <p>Correct link to table that describes which package supports the simple PLL. (DOC-1847)</p> <p>Added reset recommendations for PLLs and cascaded PLLs. (DOC-1900)</p>
December 2023	8.5	<p>Added design check topics. (DOC-1493)</p> <p>Added topic on clocking interface blocks. (DOC-1412)</p>
October 2023	8.4	<p>Added LVDS TX Static Mode Delay Setting. (DOC-1473)</p> <p>Updated SPI flash block topic. (DOC-1503)</p>
September 2023	8.3	<p>Added notes about not toggling CCK pin in QPF100F3 packages when LVDS TX is used in LVDS and SPI flash interfaces topics. (DOC-1448)</p> <p>Enhanced LVDS TX block Output Load parameter notes.</p>
June 2023	8.2	<p>Added support for QFP100F3 packages and SPI Flash Interface block. (DOC-1296)</p>
February 2023	8.1	<p>Added note to use LVDS blocks from the same side to minimize skew. (DOC-1150)</p> <p>Updated Advanced PLL Settings table descriptions. (DOC-945)</p>
December 2022	8.0	<p>Added Enable Advanced Density Setting options for the DDR block. (DOC-1008)</p>
November 2022	7.9	<p>Updated PLL Interface Designer Settings - Manual Configuration Tab notes. (DOC-1019)</p>
September 2022	7.8	<p>Updated the Interface Designer options for the DDR block to reflect new/updated DDR IP cores. (DOC-788)</p> <p>Removed PLL_EXTFB. (DOC-849)</p> <p>Updated important note for LVDS as GPIO and add LVDS resources assignment table. (DOC-886)</p> <p>Added topics on Package Planner.</p> <p>Updated Advanced PLL LOCKED signal description. (DOC-763)</p>

Date	Version	Description
June 2022	7.7	Updated Table 1: Trion Interface Block Support by Package on page 8 (DOC-791). Updated Package/Interface Support Matrix on page 9 (DOC-791). Clarified settings for creating LVDS TX and RX interfaces (DOC-791).
April 2022	7.6	Added note about not using LVDS RX as a reference clock resource to drive the PLL BR0. (DOC-768) Added T20 QFP144 to the Interface/Package Combinations table; W80 does not have LVDS. (DOC-791)
February 2022	7.5	Updated JTAG mode connection diagram. (DOC-546) When using a serialization of 3, the LVDS TX requires a 45° phase shift. (DOC-688)
June 2021	7.4	Updated recommendation for PLL settings for the LVDS clock signal. (DOC-467) Renamed simple PLL as V1 and renamed advanced PLL as V2.
February 2021	7.3	Removed TX and RX timing example for serialization width of 7 and added LVDS TX data and clock relationship waveform for width 8 and 7. Added Parallel Clock Division parameter to the LVDS TX Interface Designer settings.
February 2021	7.2	Added note in Oscillator Interface stating that the oscillator is disabled if not instantiated in Interface Designer. (DOC-370) Updated Density parameter description and added 256Mb to choice to LPDDR2 in DDR Interface Designer Settings. (DOC-377) Added LVDS TX and RX timing example for serialization width of 7. (DOC-359)
December 2020	7.1	Removed RST from LVDS RX and TX interface diagrams as they are not supported in software. (DOC-362)
December 2020	7.0	Update MIPI TX and RX Interface Block Diagram to include signal names. Updated REF_CLK description for clarity. Added notes to Output Load parameter in LVDS TX Settings in Efinity® Interface Designer table. Changed the name of the GPIO connection type from none to normal. Some alternate connection types are available as inputs to the core. Described how to use the PLL calculator for the simple PLL. Updated the notes for Output Load parameter in LVDS TX Settings in Efinity Interface Designer. (DOC-309) Updated PLL reference clock input note by asking reader to refer to PLL Timing and AC Characteristics. (DOC-336) Removed OE and RST from LVDS block as they are not supported in software. (DOC-328) Added floorplan diagram for T20 FPGAs in WLCSP80 package. Added WLCSP80 package to the Package/Interface matrix.

Date	Version	Description
July 2020	6.0	<p>Added supported features for GPIO and LVDS as GPIO.</p> <p>Added a topic on using LVDS as GPIO.</p> <p>Added note to LVDS RX interface block diagram.</p> <p>Added note about using output divider value of 4 when the LVDS receiver speed is higher 600 Mbps.</p> <p>Updated the LVDS RX and TX serilization and alternate function option descriptions.</p> <p>Updated the maximum F_{VCO} for advanced PLL to 1,600 MHz.</p> <p>You can use the PLL's CLKOUT1 and CLKOUT2 while the DDR is using CLKOUT0.</p> <p>The DDR PLL reference clock must be driven by I/O pads.</p> <p>Updated the DDR DRAM reset signal from RST_N to CFG_RST_N.</p> <p>Corrected DDR DRAM block diagram by adding DDR_CK signal.</p> <p>In MIPI RX and RTX interface description, updated maximum data pixels for RAW10 data type.</p> <p>Removed all instances of DDR3U.</p> <p>Added note to refer to AN 021 for boundary-scan testing information.</p> <p>Removed Efinity Interface Designer JTAG User TAP Interface subsection and added note and link to Efinity® Software User Guide for more information about JTAG User TAP interface.</p> <p>Added BGA400 package to interface matrix.</p> <p>Added BGA400 interface diagram.</p> <p>Added BGA400 I/O bank information.</p>
July 2020	6.0	<p>Updated PLLCLK pin name to PLL_CLKIN.</p> <p>Added PLL_EXTFB and MIPI_CLKIN as an alternative input in GPIO signals table for complex I/O buffer.</p> <p>Updated Memory CAS Latency (CL) choices in Advanced Options tab - Memory Mode register settings subtab.</p> <p>Updated Output Drive Strength choices for LPDDR2 in Advanced Options tab - Memory Mode register settings subtab.</p> <p>Corrected Enable Target 1 parameter notes in AXI 0 and AXI 1 tabs.</p> <p>Removed restriction on CLKOUT1 and CLKOUT2 when CLKIN is used to drive the DDR on CLKOUT0 in DDR DRAM PHY signals table.</p> <p>Updated description of how to select double data I/O for GPIO blocks.</p>
December 2019	5.0	<p>Enhanced the DDR interface description.</p> <p>Added a note about restrictions when using PLL_BR0 with the DDR block.</p> <p>Explained how to change the state of unused GPIO (pull up or down).</p>
August 2019	4.0	<p>Enhanced the MIPI interface description.</p> <p>Described the enhanced Resource Assigner.</p> <p>Added new FPGA and package support.</p> <p>Restructured the I/O bank information into a table.</p> <p>Clarified voltage support for DDR I/O banks.</p>
April 2019	3.0	<p>Added information for T55, T85, and T120 FPGAs.</p> <p>Updated the MIPI interface description.</p> <p>Added the DDR interface description.</p>
January 2019	2.0	<p>Added JTAG User TAP interface description.</p> <p>Added DDIO information to GPIO section.</p> <p>Published content in PDF as well as HTML Help.</p> <p>Minor changes throughout.</p>

Date	Version	Description
October 2018	1.0	Initial release.