Debugging Transceiver Links 11

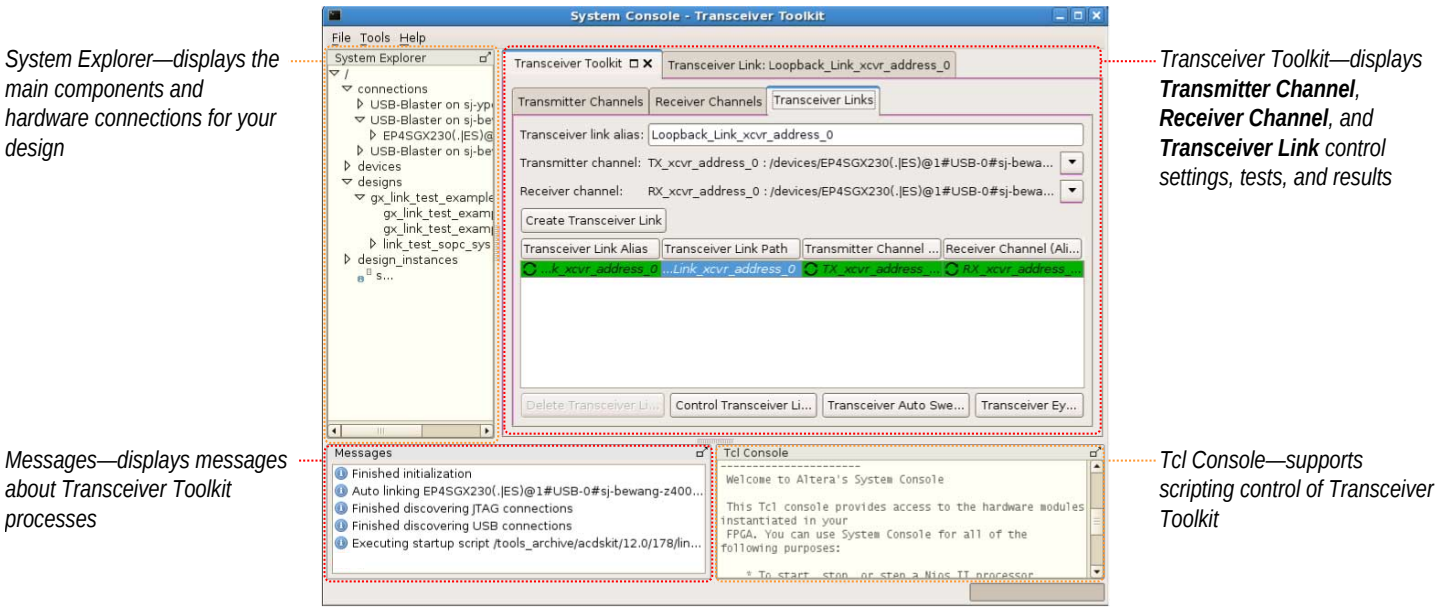
2013.11.04

QII53029 Subscribe Send Feedback

This chapter describes using the Transceiver Toolkit to optimize high-speed serial links in your board design. The Transceiver Toolkit provides real-time control, monitoring, and debugging of the transceiver links running on your board.

You can control the transmitter or receiver channels to optimize transceiver settings and hardware features. The toolkit tests bit-error rate (BER) while running multiple links at the target data rate. You can also run auto sweep tests to identify the best PMA settings for each link. An "EyeQ "graph displays the receiver horizontal and vertical eye margin during testing. The toolkit supports testing of multiple devices across one or more boards simultaneously.

Figure 11-1: Transceiver Toolkit GUI



Quick Start

Get started quickly by downloading Transceiver Toolkit design examples from the **On-Chip Debugging Design Examples** website. For an online demonstration of how to use the Transceiver Toolkit to run a high-speed link test with one of the design examples, refer to the **Transceiver Toolkit Online Demo** on the Altera website.

Transceiver Debugging Overview

Testing transceiver links involves configuring your system for debug, and then running various link tests. The following table details the steps in this flow.

Table 11-1: Transceiver Link Debugging Flow

	Flow Description
System Configuration Steps	1. Use one of the following methods to define a system that includes necessary transceiver debugging components: • Click **Tools** > **Qsys** and modify Altera design examples for use with your design. • Click **Tools** > **MegaWizard Plug-In Manager** to define and integrate debugging components into your own design. 2. Click **Assignments** > **Pin Planner** to assign device I/O pins to match your device and board. 3. Click **Processing** > **Start Compilation** to compile your design. 4. Connect your target device to Altera programming hardware. 5. Click **Tools** > **Programmer** to program your target device.
Link Debugging Steps	1. Click **File** > **Load Design**, and select the SRAM Object File (**.sof**) generated for your transceiver design. 2. (Optional) Create any additional links between transmitter channel and receiver. 3. Run any of the following tests: • Run BER with various combinations of PMA settings. • Run PRBS Signal eye tests • Run custom traffic tests • Run link optimization tests • Directly control PMA analog settings to experiment with settings while the link is running

Using the Transceiver Toolkit GUI

The Transceiver Toolkit GUI helps you to easily visualize and debug transceiver links in your design. To launch the GUI, click **Tools** > **System Console** > **Transceiver Toolkit**.

Related Information

Toolkit GUI Setting Reference on page 11-21

Controlling Transceiver Channels

You can directly control and monitor transmitters, receivers, and links running on the board in real time. You can transmit a data pattern across the transceiver link, and then report the signal quality of the received data in terms of bit error rate or eye margin with EyeQ. Click **Control Transmitter** (**Transmitter Channels** tab), **Control Receiver** (**Receiver Channels** tab), or **Control Link** (**Transceiver Links** tab) to adjust transmitter or receiver settings while the channels are running.

Auto Sweep Testing

Use the auto sweep feature to automatically sweep ranges for the best transceiver PMA settings. You can store a history of the test runs and keep a record of the best PMA settings. You can use the best found settings in your final design for improved signal integrity compared with the default settings.

Adaptive Equalization Control

Adaptive equalization (AEQ) automatically evaluates and selects the best combination of reconfiguration equalizer settings for the receiver. AEQ continuously evaluates and changes the settings for current conditions. You can use AEQ for multiple, independently controlled receiver channels.

Enable this feature by selecting **One-time adaptive** for the **Equalization control** receiver setting.

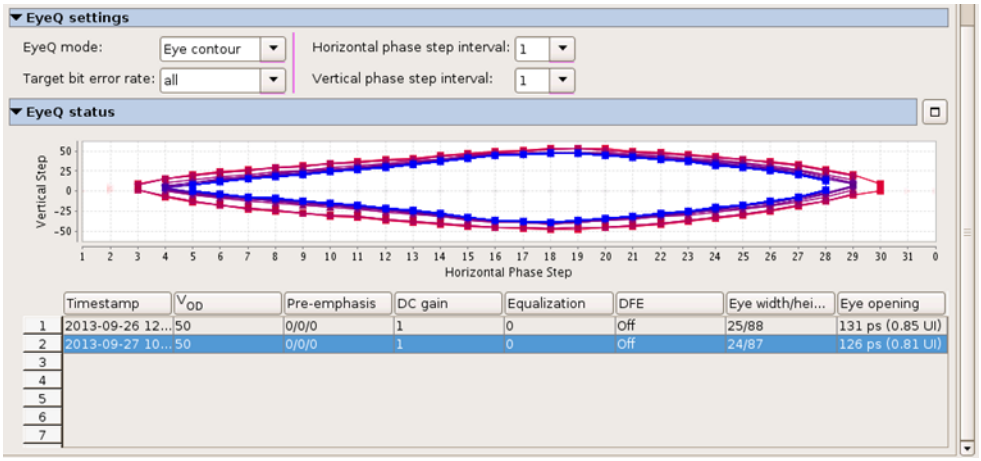
Signal Eye Margin Testing

Some Altera devices include EyeQ circuitry that allows visualization of the horizontal and vertical eye margin at the receiver. For supported devices, use signal eye tests to tune the PMA settings of your transceiver, which results in the best eye margin and BER at high data rates. This GUI is disabled for unsupported devices.

The EyeQ graph displays a bathtub curve or eye diagram representing eye margin. The run list displays the statistics of each EyeQ test. When PMA settings are suitable, the bathtub curve is wide, with sharp slopes near the edges. The curve is up to 30 units wide. If the bathtub is narrow, then the signal quality is poor. The wider the bathtub curve, the wider the eye. The smaller the bathtub curve, the smaller the eye. The eye contour shows the estimated horizontal and vertical eye opening at the receiver.

You can right-click any of the test runs in the list, and then click **Apply Settings to Device** to quickly apply that PMA setting to your device. You can also click **Export**, **Import**, or **Create Report**.

Figure 11-2: EyeQ Settings and Status Showing Results of Two Test Runs



Serial Bit Comparator Mode

Serial bit comparator mode allows you to run EyeQ diagnostic features with any PRBS patterns or user-design data, without disrupting the data path.

To enable this mode you must enable the following debugging component options when configuring the debugging system:

Table 11-2: Component Settings for Serial Bit Comparator Mode

Debugging Component	Setting for Serial Bit Mode
Transceiver Reconfiguration Controller	Turn on **Enable EyeQ block** and **Enable Bit Error Rate Block**
Data Pattern Generator	Turn on **Enable Bypass interface**

Serial bit comparator mode is less accurate than **Data pattern checker** mode for single bit error checking. Do not use **Serial bit comparator** mode if you require an exact error rate. Use the **Serial bit comparator** mode for checking a large window of error.

Serial bit comparator mode has the following hardware limitations:

- You can add only one BER counter per reconfiguration controller. You can monitor only one channel at a time if the reconfiguration controller is shared by multiple channels.
- Use of the reconfiguration controller is limited while BER is running. For example, you cannot switch logical channels, run decision feedback equalization one-time, run adaptive equalization one-time, or recalibrate because it corrupts the error count.
- This method may double count the number of errors.
- The bit error counter is not read in real-time because it is read through the memory mapped interface.

Scripting Support

You can alternatively use Tcl commands to access Transceiver Toolkit functions, rather than using the GUI. You can script various tasks, such as loading a project, creating design instances, linking device resources, and identifying high-speed serial links. You can save your project setup in a Tcl script for use in subsequent testing sessions. You can also build a custom test routine script.

After you set up and define links that describe the entire physical system, you can click Save Tcl Script to save the setup for future use. To run the scripts, double-click script names in the System Explorer scripts folder.

Related Information

Scripting API on page 11-26

Configuring Systems for Debug

To debug transceivers, you must first configure a system that includes the appropriate Altera IP core(s) that supports each debugging operation. You can create such a system by either modifying an Altera design example, or by integrating debugging components into your own design.

You can quickly parameterize the debugging components by clicking **Tools** > **MegaWizard Plug-In Manager**. Refer to the following configuration for your debugging operation.

Table 11-3: Transceiver Toolkit IP Core Configuration

Component	Debugging Functions	Parameterization Notes
Transceiver Native PHY	Supports all debugging functions	• If **Enable 10G PCS** is enabled, **10G PCS protocol mode** must be set to **basic** on the **10G PCS** tab.
Custom PHY	Test all possible transceiver parallel data widths	• Set lanes, group size, serialization factor, data rate, and input clock frequency to match your application. • Turn on **Avalon data interfaces**. • Disable **8B/10B**. • Set **Word alignment mode** to **manual**. • Disable **rate match FIFO**. • Disable **byte ordering block**.
Low Latency PHY	Test at more than 8.5 Gbps in GT devices or use of PMA direct mode (such as when using six channels in one quad)	• Set **Phase compensation FIFO mode** to **EMBEDDED** above certain data rates and set to **NONE** for PMA direct mode (Stratix IV designs only). • Turn on **Avalon data interfaces**. • Set serial loopback mode to enable serial loopback controls in the toolkit.
Avalon-ST Data Pattern Generator	Generates standard data test patterns at Avalon-ST source ports	• Select PRBS7, PRBS15, PRBS23, PRBS31, high frequency, or low frequency patterns
Avalon-ST Data Pattern Checker	Validates incoming data stream against test patterns accepted on Avalon streaming sink ports	• Specify a value for `ST_DATA_W` that matches the FPGA-fabric interface width.
Reconfiguration Controller	Supports PMA control and other transceiver settings	• Connect the reconfiguration controller to all PHYs that you want controlled by the toolkit. • Connect `reconfig_from_xcvr` to `reconfig_to_xcvr`. • Enable Analog controls. • Enable EyeQ block (Stratix V devices only). • Enable AEQ block (Stratix V devices only). • Enable DFE block (Stratix V devices only).
JTAG to Avalon Master Bridge	Accepts encoded streams of bytes with transaction data and initiates Avalon-MM transactions	N/A

Related Information

- **Adapting Altera Design Examples**
- **Integrating Debug Components In Your Design** on page 11-8

Adapting Altera Design Examples

Altera provides design examples to help you quickly test your own design. You can experiment with these designs and modify them for your own application. Refer to the **readme.txt** of each design example for more information. Download the Transceiver Toolkit design examples from the On-Chip Debugging Design Examples page of the Altera website.

You can use the design examples as a starting point to work with a particular signal integrity development board. The design examples provide the components to quickly test the functionality of the receiver and transmitter channels in your design. You can easily change the transceiver settings in the design examples to see how they affect your transceiver link performance. You can isolate and verify the high-speed serial links without debugging other logic in your design. You can modify and customize the design examples to match your intended transceiver design.

Once you have downloaded the design examples, open the Quartus II software and restore the design example project archive. If you have access to the same development board with the same device as mentioned in the **readme.txt** file of the example, you can directly program the device with the provided programming file in that example. If you want to recompile the design, you must make your modifications to the configuration in Qsys, regenerate in Qsys, and recompile the design in the Quartus II software to generate a new programming file.

If you have the same board as mentioned in the **readme.txt** file, but a different device on your board, you must choose the appropriate device and recompile the design. For example, some early development boards are shipped with engineering sample devices.

You can make changes to the design examples so that you can use a different development board or different device. You make pin assignment changes and then recompile the design. If you have a different board, you must edit the necessary pin assignments and recompile the design examples.

Related Information

- **On-Chip Debugging Design Examples**

Modifying Design Examples

You can adapt an Altera design example to experiment with various configurations that match your own design. For example, you can change data rate, number of lanes, PCS-PMA width, FPGA-fabric interface width, or input reference clock frequency. To modify the design examples, you modify the IP core parameters and regenerate the system in Qsys. Next, you modify the top-level design file, and reassign device I/O pins as necessary.

Before you begin

To modify a design example PHY block to match your design, follow these steps:

1. Determine the number of channels required by your design.
2. Open the <*project name*>**.qpf** for the design example in the Quartus II software.
3. Click **Tools** > **Qsys**.
4. On the **System Contents** tab, right-click the PHY block and click **Edit**. On the **General** tab, specify options for the PHY block to match your design requirement for number of lanes, data rate, PCS-PMA width, FPGA-fabric interface width, and input reference clock frequency.
5. Right-click the PHY block and click **Edit**. Click **Additional Options** and specify a multiple of the FPGA-fabric interface data width for **Avalon Data Symbol Size**. The available values are 8 or 10. Click **Finish**.

6. Delete any timing adapter from the design. The timing adaptors are not required.
7. Right-click **data pattern generator** and click **Edit**. Specify a value for ST_DATA_W that matches the FPGA-fabric interface width.
8. Right-click **data pattern checker** and click **Edit**. Specify a value for ST_DATA_W that matches the FPGA-fabric interface width.
9. Right-click **transceiver configuration controller** and click **Edit**. Specify 2* number of lanes for the number of reconfigurations interfaces. Click **finish**.
10. Add one **data pattern generator** and **data pattern checker** for each transmitter and receiver lane. From the **Component Library**, instantiate the **data pattern generator** and **data pattern checker** components. The components are under **Debug and Performance** under **Peripherals**.
11. Create connections for the data pattern generator and data pattern checker components. Right-click the net name in the **System Contents** tab and specify the following connections.

From		To	
Block Name	**Net Name**	**Block Name**	**Net Name**
clk_100	clk	data_pattern_generator	csr_clk
clk_100	clk_reset	data_pattern_generator	csr_clk_reset
master_0	master	data_pattern_generator	csr_slave
xcvr_*_phy_0	tx_clk_out0	data_pattern_generator	pattern_out_clk
xcvr_*_phy_0	tx_parallel_data0	data_pattern_generator	pattern_out
clk_100	clk	data_pattern_checker	csr_clk
clk_100	clk_reset	data_pattern_checker	csr_clk_reset
master_0	master	data_pattern_checker	csr_slave
xcvr_*_phy_0	rx_clk_out0	data_pattern_checker	pattern_in_clk
xcvr_*_phy_0	rx_parallel_data0	data_pattern_checker	pattern_in

12. Click **System** > **Assign Base Addresses**.
13. Connect the reset port of timing adapters to `clk_reset` of `clk_100`.
14. To implement the changes to the system, click **Generate**.
15. If you modify the number of lanes in the PHY, you must update the top-level file accordingly. The following example shows Verilog HDL code for a two-channel design that declares input and output ports in the top-level design. The example design includes the low latency PHY IP core. If you modify the PHY parameters, you must modify the top-level design with the correct port names. Qsys displays an example of the PHY in the design on the **HDL example** tab.

```
module low_latency_10g_1ch DUT (
 input  wire GXB_RXL11,
 input  wire GXB_RXL12,
 output wire GXB_TXL11,
```

```
 output wire GXB_TX12
);
.....

low_latency_10g_1ch DUT (
 .....
 .xcvr_low_latency_phy_0_tx_serial_data_export ({GXB_TXL11,
GXB_TXL12}),
 .xcvr_low_latency_phy_0_rx_serial_data_export ({GXB_RXL11,
GXB_TXL12}),
 .....
 );
```

16. Click **Assignments** > **Pin Planner** and update pin assignments to match your board.
17. Edit the design's Synopsys Design Constraints (**.sdc**) to reflect the reference clock change. Ignore the reset warning messages.
18. Recompile the design.

Modifying Design Example Pin Assignments

Click **Assignments** > **Pin Planner** to modify pin assignments in your project to match development kit or your own board.

Related Information

- **Managing I/O Pins**

Integrating Debug Components In Your Design

This section describes integrating debugging IP components into your own design. You can integrate transceiver debugging components into your design, rather than modifying the Altera Debugging Design Examples.

Configuring BER Tests

To integrate components with your design for BER testing, follow these steps.

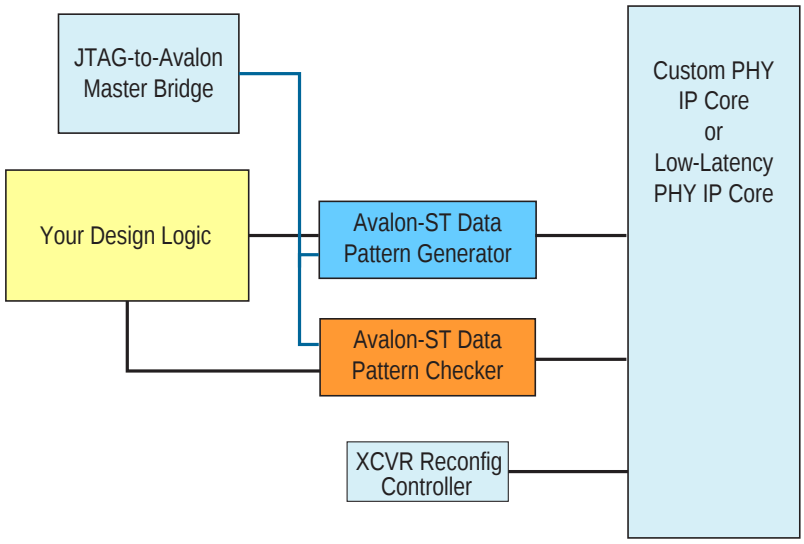
1. To add and connect debugging components to your system, click **Tools** > **Qsys**.
2. Define and instantiate the following from the Qsys component library:
   - Click **Peripherals** > **Debug and Performance** > **Altera Avalon Data Pattern Generator**. Turn on **Enable Bypass interface** for connection to design logic.
   - Click **Peripherals** > **Debug and Performance** > **Altera Avalon Data Pattern Checker**. Turn on **Enable Bypass interface** for connection to design logic.
   - Click **Bidges** > **Memory-Mapped** > **JTAG to Avalon Master Bridge**.
3. Make the following connections between components in your system:

From	To
Your Design Logic	Data Pattern Generator bypass port
Data Pattern Generator	PHY input port
JTAG to Avalon Master Bridge	Altera Avalon Data Pattern Generator

From	To
JTAG to Avalon Master Bridge	Altera Avalon Data Pattern Checker
JTAG to Avalon Master Bridge	PHY input port
Data Pattern Checker	PHY output port
Transceiver Reconfiguration Controller	PHY input port

4. To generate the system, click **Generate** > **Generate**.
5. To compile the design and generate configuration files, click **Processing** > **Start Compilation**.
6. Click **Tools** > **Programmer** and configure the target device with your debugging design.
   You are now prepared to run BER tests.

Figure 11-3: BER Test Configuration



Related Information

Running BER Tests on page 11-18

Configuring PRBS Signal Eye Tests

To integrate components with your design for testing PRBS signal eye, follow these steps.

1. To add and connect debugging components to your system, click **Tools** > **Qsys**.
2. Define and instantiate the following from the Qsys component library:

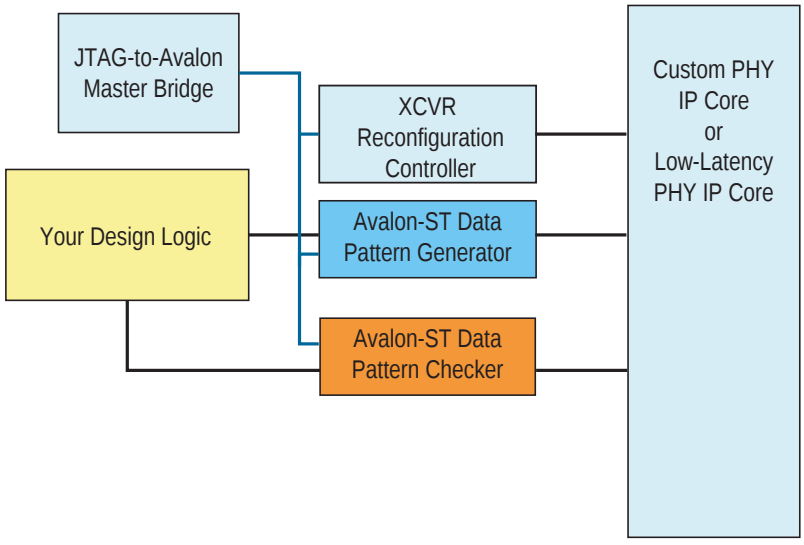
- Click **Peripherals** > **Debug and Performance** > **Altera Avalon Data Pattern Generator**. Turn on **Enable Bypass interface** for connection to design logic.
- Click **Peripherals** > **Debug and Performance** > **Altera Avalon Data Pattern Checker**. Turn on **Enable Bypass interface** for connection to design logic.
- Click **Bridges** > **Memory-Mapped** > **JTAG to Avalon Master Bridge**.
- Click **Interface Protocols** > **Transceiver PHY** > **Transceiver Reconfiguration Controller**. Turn on **Enable EyeQ block** to enable signal eye analysis.

3. Make the following connections between components in your system:

From	To
Your Design Logic	Data Pattern Generator bypass port
Data Pattern Generator	PHY input port
JTAG to Avalon Master Bridge	Altera Avalon Data Pattern Generator
JTAG to Avalon Master Bridge	Altera Avalon Data Pattern Checker
Data Pattern Checker	PHY output port
JTAG to Avalon Master Bridge	Transceiver Reconfiguration Controller
JTAG to Avalon Master Bridge	PHY input port
Transceiver Reconfiguration Controller	PHY input port

4. To generate the system, click **Generate** > **Generate**.
5. To compile the design and generate configuration files, click **Processing** > **Start Compilation**.
6. Click **Tools** > **Programmer** and configure the target device with your debugging design.
You are now prepared to run PRBS signal eye tests.

Figure 11-4: PRBS Signal Eye Test Configuration



Related Information

Running PRBS Signal Eye Tests on page 11-19

Configuring Custom Traffic Signal Eye Tests

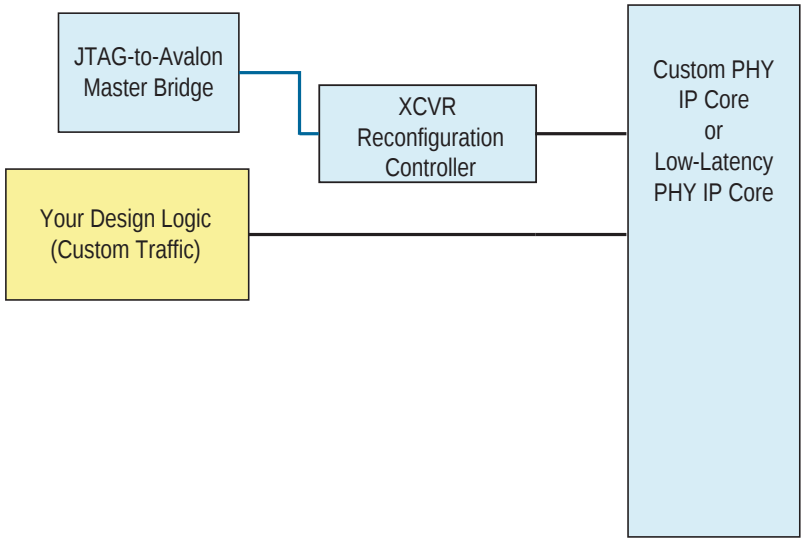
To integrate components with your design for testing custom traffic signal eye, follow these steps.

1. To add and connect debugging components to your system, click **Tools** > **Qsys**.
2. Define and instantiate the following from the Qsys component library:
   - Click **Bridges** > **Memory-Mapped** > **JTAG to Avalon Master Bridge**.
   - Click **Interface Protocols** > **Transceiver PHY** > **Transceiver Reconfiguration Controller**. Turn on **Enable EyeQ block** to enable signal eye analysis. Turn on **Enable Bit Error Rate Block** to perform BER testing.
3. Make the following connections between components in your system:

From	To
Your design logic with custom traffic	PHY input port
JTAG to Avalon Master Bridge	Transceiver Reconfiguration Controller
JTAG to Avalon Master Bridge	PHY input port
Transceiver Reconfiguration Controller	PHY input port

4. To generate the system, click **Generate** > **Generate**.
5. To compile the design and generate configuration files, click **Processing** > **Start Compilation**.
6. Click **Tools** > **Programmer** and configure the target device with your debugging design.
   You are now prepared to run custom traffic signal eye tests.

Figure 11-5: Custom Traffic Signal Eye Test Configuration



Related Information

Running Custom Traffic Tests on page 11-19

Configuring Link Optimization Tests

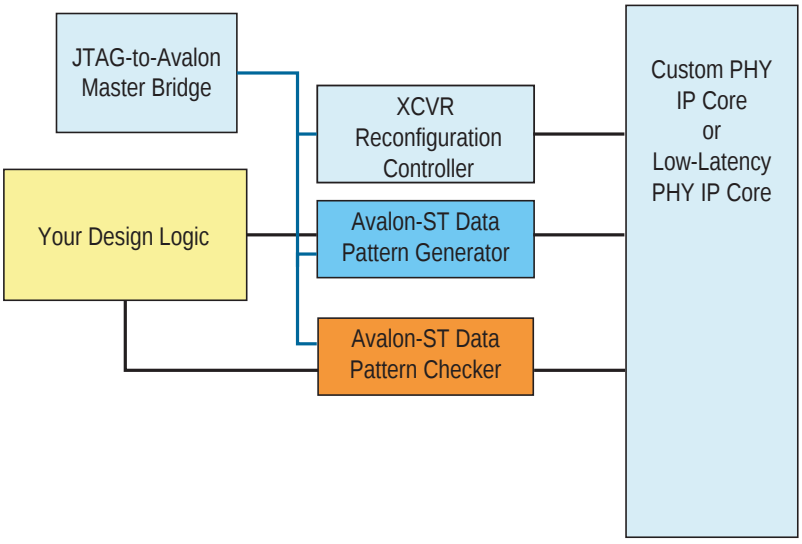
To integrate components with your design for link optimization tests, follow these steps.

1. To add and connect debugging components to your system, click **Tools** > **Qsys**.
2. Define and instantiate the following from the Qsys component library:
   - Click **Peripherals** > **Debug and Performance** > **Altera Avalon Data Pattern Generator**. Turn on **Enable Bypass interface** for connection to design logic.
   - Click **Peripherals** > **Debug and Performance** > **Altera Avalon Data Pattern Checker**. Turn on **Enable Bypass interface** for connection to design logic.
   - Click **Bridges** > **Memory-Mapped** > **JTAG to Avalon Master Bridge**.
   - Click **Interface Protocols** > **Transceiver PHY** > **Transceiver Reconfiguration Controller**. Turn on **Enable EyeQ block**, **Enable Analog controls**, **Enable decision feedback equilizer (DFE) block**, and **Enable adaptive equalization (AEQ) block** to enable all types of link analysis.
3. Make the following connections between components in your system:

From	To
Your Design Logic	Data Pattern Generator bypass port
Data Pattern Generator	PHY input port
JTAG to Avalon Master Bridge	Altera Avalon Data Pattern Generator
JTAG to Avalon Master Bridge	Altera Avalon Data Pattern Checker
Data Pattern Checker	PHY output port
JTAG to Avalon Master Bridge	Transceiver Reconfiguration Controller
JTAG to Avalon Master Bridge	PHY input port
Transceiver Reconfiguration Controller	PHY input port

4. To generate the system, click **Generate** > **Generate**.
5. To compile the design and generate configuration files, click **Processing** > **Start Compilation**.
6. Click **Tools** > **Programmer** and configure the target device with your debugging design.
   You are now prepared to run BER tests.

Figure 11-6: Link Optimization Test Configuration



Related Information

Running Link Optimization Tests on page 11-20

Configuring PMA Analog Setting Control

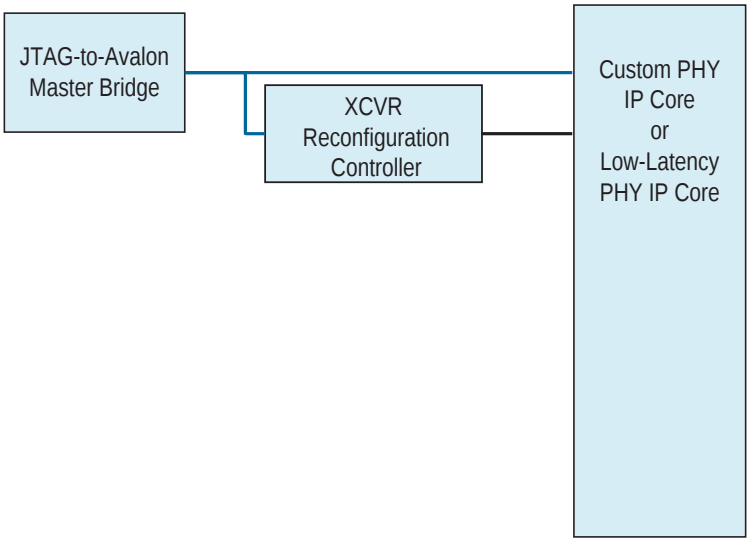
To integrate components for PMA analog setting control, follow these steps.

1. To add and connect debugging components to your system, click **Tools** > **Qsys**.
2. Define and instantiate the following from the Qsys component library:
   - Click **Bridges** > **Memory-Mapped** > **JTAG to Avalon Master Bridge**.
   - Click **Interface Protocols** > **Transceiver PHY** > **Transceiver Reconfiguration Controller**. Turn on **Enable Analog controls**. You can optionally turn on **Enable EyeQ block**, **Enable decision feedback equilizer (DFE) block**, and **Enable adaptive equalization (AEQ) block** to enable these types of link analysis.
3. Make the following connections between components in your system:

From	To
JTAG to Avalon Master Bridge	Transceiver Reconfiguration Controller
JTAG to Avalon Master Bridge	PHY input port
Transceiver Reconfiguration Controller	PHY input port

4. To generate the system, click **Generate** > **Generate**.
5. To compile the design and generate configuration files, click **Processing** > **Start Compilation**.
6. Click **Tools** > **Programmer** and configure the target device with your debugging design.
   You are now prepared to control PMA settings.

Figure 11-7: PMA Analog Setting Control Configuration



Related Information

Controlling PMA Analog Settings on page 11-20

Debugging Transceiver Links

The Transceiver Toolkit allows you to control and monitor the performance of high-speed serial links running on your board in real-time. You can identify the transceiver links in your design, transmit a data pattern across the transceiver link, and report the signal quality of the received data in terms of bit error rate, bathtub curve, or EyeQ graph (for supported families).

The toolkit automatically identifies the transceiver links in your design, or you can manually create transceiver links. You can then run auto sweep to help you quickly identify the best physical media attachment (PMA) settings for each link. You can directly control the transmitter/receiver channels to experiment with various settings suggested by auto sweep. The EyeQ graph allows you to visualize the estimated horizontal and vertical eye opening at the receiver.

The Transceiver Toolkit supports various transceiver link testing configurations. You can identify and test the transceiver link between two Altera devices, or you can transmit a test pattern with a third-party device and monitor the data on an Altera device receiver channel. If a third-party chip includes self-test capability, then you can send the test pattern from the Altera device and monitor the signal integrity at the third-party device receiver channel. If the third-party device supports reverse serial loopback, you can run the test entirely within the Transceiver Toolkit.

Before you can monitor transceiver channels, you must configure a system with debugging components, and program the design into an FPGA. Once those steps are complete, use the following flow to test the channels:

1. Load the design in Transceiver Toolkit
2. Link hardware resources
3. Verify hardware connections
4. Identify transceiver channels
5. Run link tests or control PMA analog settings
6. View results

Step 1: Load Your Design

The Transceiver Toolkit automatically loads the last compiled design upon opening. To load any design into the toolkit, click **File** > **Load Design** and select the **.sof** programming file generated for your transceiver design. Loading the **.sof** automatically links the design to the target hardware in the toolkit. The toolkit automatically discovers links between transmitter and receiver of the same channel. The System Explorer displays information about the loaded design.

Step 2: Link Hardware Resources

The toolkit automatically discovers connected hardware and designs. You can also manually link a design to connected hardware resources in the System Explorer.

You can identify and test the transceiver link between two Altera devices, or you can transmit a test pattern with a third-party device and monitor the data on an Altera device receiver channel. If a third-party device includes self-test capability, you can send the test pattern from the Altera device and monitor the signal integrity on the third-party receiver channel. If the third-party device supports reverse serial loopback, you can run the test entirely within the Transceiver Toolkit.

If you are using more than one Altera board, you can set up a test with multiple devices linked to the same design. This setup is useful when you want to perform a link test between a transmitter and receiver on two separate devices. You can also load multiple Quartus II projects and make links between different systems. You can perform tests on completely separate and unrelated systems in a single tool instance.

Note: Prior to the Transceiver Toolkit version 11.1, you must manually load and link your design to hardware. In version 11.1 and later, the Transceiver Toolkit automatically links any device programmed with a project.

Figure 11-8: One Channel Loopback Mode

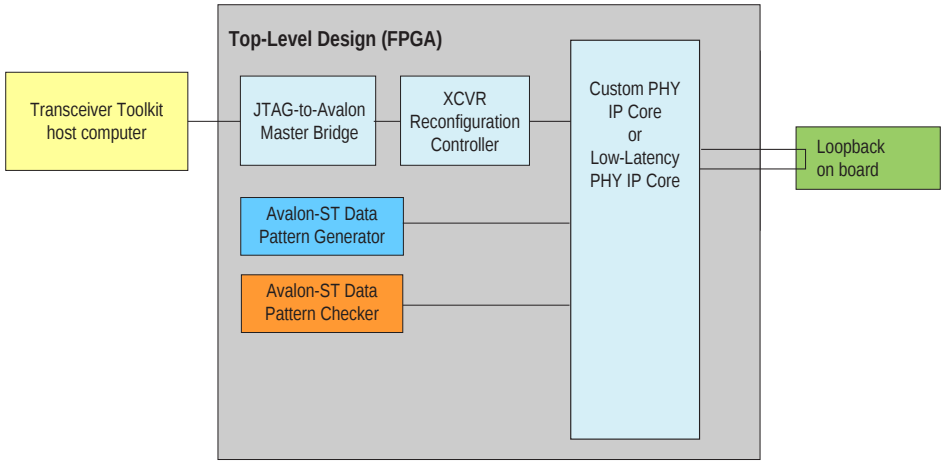
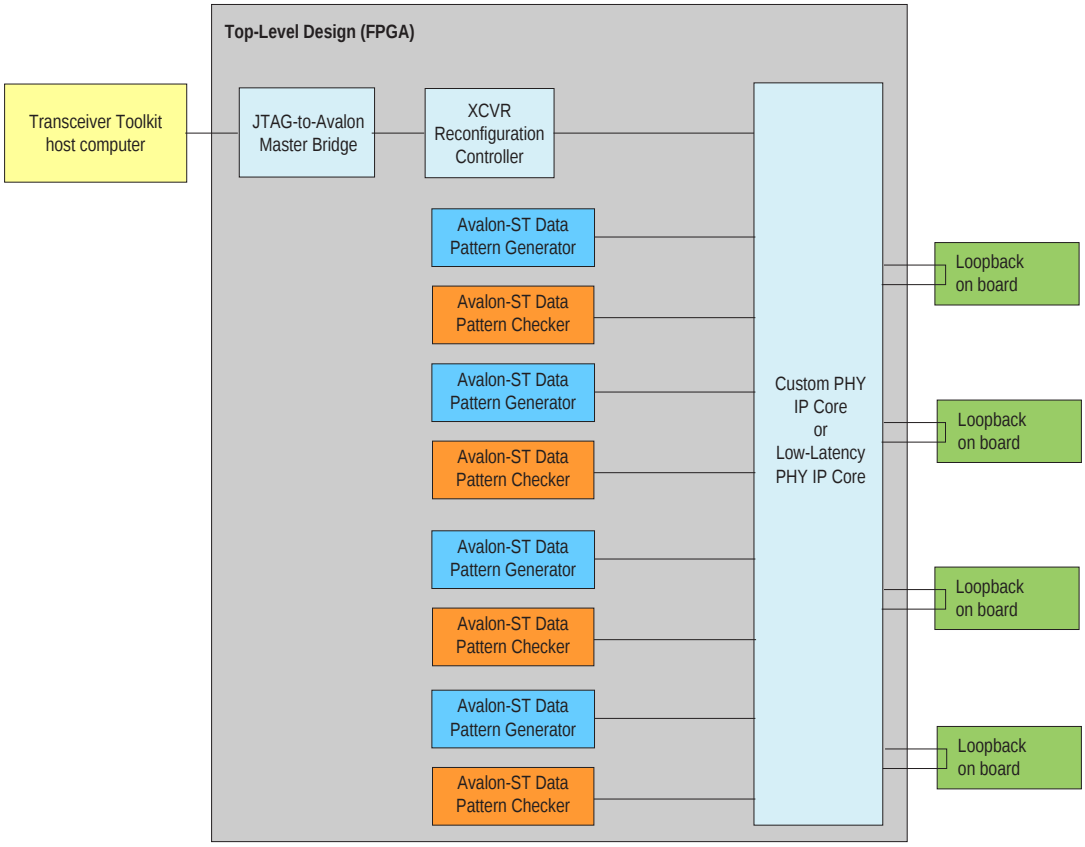


Figure 11-9: Four Channel Loopback Mode



Linking One Design to One Device

To link one design to one device by one USB-Blaster download cable, follow these steps:

1. Load the design for your Quartus II project.

2. Link each device to an appropriate design if the design has not auto-linked.
3. Create the link between channels on the device to test.

Linking Two Designs to Two Devices

To link two designs to two separate devices on the same board, connected by one USB-Blaster download cable, follow these steps:

1. Load the design for all the Quartus II project files you might need.
2. Link each device to an appropriate design if the design has not auto-linked.
3. Open the project for the second device.
4. Link the second device on the JTAG chain to the second design (unless the design auto-links).
5. Create a link between the channels on the devices you want to test.

Linking Designs and Devices on Separate Boards

To link two designs to two separate devices on separate boards, connected to separate USB-Blaster download cables, follow these steps:

1. Load the design for all the Quartus II project files you might need.
2. Link each device to an appropriate design if the design has not auto-linked.
3. Create the link between channels on the device to test.
4. Link the device you connected to the second USB-Blaster download cable to the second design.
5. Create a link between the channels on the devices you want to test.

Linking One Design on Two Devices

To link the same design on two separate devices, follow these steps:

1. In the Transceiver Toolkit, open the **.sof** you are using on both devices.
2. Link the first device to this design instance.
3. Link the second device to the design.
4. Create a link between the channels on the devices you want to test.

Step 3: Verify Hardware Connections

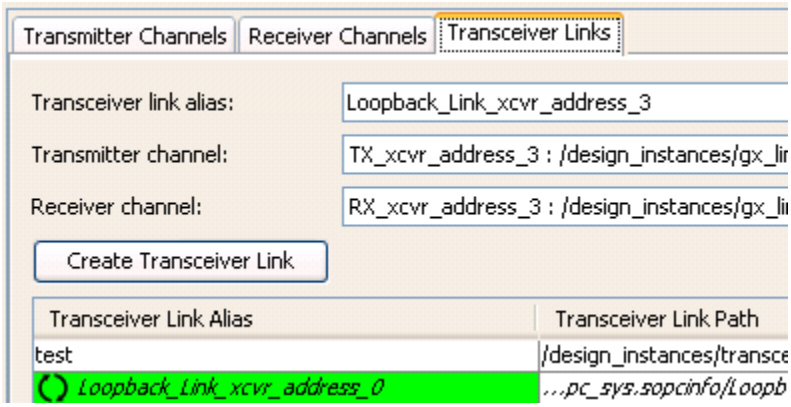
After you load your design and link your hardware, verify that the channels are connected correctly and looped back properly on the hardware. Use the toolkit to send data patterns and receive them correctly. Verifying your link and correct channel before you perform Auto Sweep or EyeQ tests can save time in the work flow.

After you have verified that the transmitter and receiver are communicating with each other, you can create a link between the two transceivers so that you can perform Auto Sweep and EyeQ tests with this pair.

Step 4: Identify Transceiver Channels

The Transceiver Toolkit automatically displays recognized transmitter and receiver channels. The toolkit identifies a channel automatically whenever a receiver and transmitter share a transceiver channel. You can also manually identify the transmitter and receiver in a transceiver channel and create a link between the two for testing.

Figure 11-10: Identified Transceiver Links and Status



When you run link tests, channel color highlights indicate the test status:

Table 11-4: Transceiver Toolkit IP Core Configuration

Color	Transmitter Channel	Receiver Channel
Red	Channel is closed or generator clock is not running	Channel is closed or checker clock is not running
Green	Generator is sending a pattern	Checker is checking and data pattern is locked
Neutral	Channel is open, generator clock is running, and generator is not sending a pattern, or generator is not sending a pattern	Channel is open, checker clock is running, and checker is not checking, or checker is not checking
Yellow	N/A	Checker is checking and data pattern is not locked

Step 5: Run Link Tests

Once you identify the transceiver channels for debugging, you can run various link tests in the toolkit.

Use the **Transceiver Links** tab to control link test. For example, use the Auto Sweep feature to sweep transceiver settings to determine the parameters that support the best BER value. Open the **Transmitter**, **Receiver** and **Link Control** panels to control the PMA settings and run tests.

Running BER Tests

You can run BER tests across your transceiver link. After programming the FPGA with your debugging design, loading the design in the toolkit, and linking hardware, follow these steps to run BER tests:

1. Click **Tools** > **System Console** > **Transceiver Toolkit**.
2. Click the **Transmitter Channels** tab, select the generator you want to control, and click **Create Transmitter Channel**.
3. Click the **Receiver Channels** tab, select the checker you want to control, and click **Create Receiver Channel**.

4. Click the **Transceiver Links** tab, select the transmitter and receiver pair you want to control, and click **Create Transceiver Link**.
5. Click **Control Transceiver Link**, and specify a PRBS **Test pattern** and **Checker mode** for the **Data pattern checker**. If you select **Bypass** for the **Test pattern**, the toolkit bypasses the PRBS generator and runs your design through the link. The bypass option is only available after you turn on **Enable Bypass interface** in the Reconfiguration Controller component.
6. Experiment with **Reconfiguration**, **Generator**, or **Checker** settings.
7. Click **Start** to run the pattern with your settings. You can then click **Inject Error** to inject error bits, **Reset** the counter, or **Stop** the test.

Related Information

Configuring BER Tests on page 11-8

Running PRBS Signal Eye Tests

You can run PRBS signal eye tests to visualize the estimated horizontal and vertical eye opening at the receiver. After programming the FPGA with your debugging design, loading the design in the toolkit, and linking hardware, follow these steps to run PRBS signal eye tests:

1. Click **Tools** > **System Console** > **Transceiver Toolkit**.
2. Click the **Transmitter Channels** tab, select the generator you want to control, and click **Create Transmitter Channel**.
3. Click the **Receiver Channels** tab, select the checker you want to control, and click **Create Receiver Channel**.
4. Click the **Transceiver Links** tab, select the transmitter and receivers you want to control, and click **Create Transceiver Link** .
5. Click **Transceiver EyeQ**, and select **EyeQ** as the **Test mode**. The **EyeQ** mode displays test results as a bathtub curve or eye contour representing bit error and phase offset data.
6. Specify the PRBS **Test pattern** and the **Checker mode**. Use **Serial bit comparator** checker mode only for checking a large window of error with custom traffic.
7. Specify **Run length** and **EyeQ settings** to control the test coverage and type of EyeQ results displayed, respectively.
8. Click **Start** to run the pattern with your settings. EyeQ uses the current channel settings to start a phase sweep of the channel. The phase sweep runs 32 iterations. As the run progresses, view the status under **EyeQ status**. Use this diagram to compare PMA settings for the same channel and to choose the best combination of PMA settings for a particular channel.
9. When the run completes the chart is displayed and the characteristics of each run are listed in the run list. You can click **Stop** to halt the test, change the PMA settings, and re-start the test. Click **Create Report** to export data to a table format for further viewing.

Related Information

Configuring PRBS Signal Eye Tests on page 11-9

Running Custom Traffic Tests

After programming the FPGA with your debugging design, loading the design in the toolkit, and linking hardware, follow these steps to run custom traffic tests:

1. Click **Tools** > **System Console** > **Transceiver Toolkit**.
2. Click the **Receiver Channels** tab, select the associated reconfiguration controller, and click **Create Receiver Channel**.

3. Click the **Receiver EyeQ**, and select **EyeQ** as the **Test mode**. The **EyeQ** mode displays test results as a bathtub curve or eye contour representing bit error and phase offset data.
4. Specify the PRBS **Test pattern**
5. For **Checker mode**, select **Serial bit comparator**. This option is only available after you turn on **Enable EyeQ block** and **Enable Bit Error Rate Block** for the Reconfiguration Controller component.
6. Specify **Run length** and **EyeQ settings** to control the test coverage and type of EyeQ results displayed, respectively.
7. Click **Start** to run the pattern with your settings. EyeQ uses the current channel settings to start a phase sweep of the channel. The phase sweep runs 32 iterations. As the run progresses, view the status under **EyeQ status**.
8. When the run completes the chart is displayed and the characteristics of each run are listed in the run list. You can click **Stop** to halt the test, change the PMA settings, and re-start the test. Click **Create Report** to export data to a table format for further viewing.

Related Information

Configuring Custom Traffic Signal Eye Tests on page 11-11

Running Link Optimization Tests

After programming the FPGA with your debugging design, loading the design in the toolkit, and linking hardware, follow these steps to run link optimization tests:

1. Click **Tools** > **System Console** > **Transceiver Toolkit**.
2. Click the **Transceiver Links** tab, and select the channel you want to control.
3. Click **Transceiver Autosweep**. The **Advanced** tab appears with **Auto sweep** as **Test mode**.
4. Specify the PRBS **Test pattern**.
5. Specify **Run length** and experiment with the **Transmitter settings**, **Reciever settings** to control the test coverage and PMA settings, respectively.
6. Click **Start** to run all combinations of tests meeting the PMA parameter limits.
7. When the run completes the chart is displayed and the characteristics of each run are listed in the run list. You can click **Stop** to halt the test, change the PMA settings, and re-start the test. Click **Create Report** to export data to a table format for further viewing.
8. To use decision feedback equalization (DFE) to determine the best tap settings, follow these steps:
   a. Use the Auto sweep to find optimal PMA settings while leaving the **DFE mode** set to **Off**.
   b. If BER = 0, use the best PMA settings achieved.
   c. If BER > 0, use this PMA setting, and set the minimum and maximum values obtained from Auto Sweep to match this setting. Set the maximum DFE range to limits for each of the three DFE settings.
   d. Run **Create Report** to view the results and determine which DFE setting has the best BER. Use these settings in conjunction with the PMA settings for the best results.

Related Information

Configuring Link Optimization Tests on page 11-12

Controlling PMA Analog Settings

You can directly control PMA analog settings to experiment with settings while the link is running. To control PMA analog settings, follow these steps:

1. Click **Tools** > **System Console** > **Transceiver Toolkit**.

2. Click the **Transmitter Channels** tab, define a transmitter without a generator, and click **Create Transmitter Channel**.
3. Click the **Receiver Channels** tab, define a receiver without a generator, and click **Create Receiver Channel**.
4. Click the **Transceiver Links** tab, select the transmitter and receivers you want to control, and click **Create Transceiver Link** .
5. Click **Control Receiver Channel**, **Control Transmitter Channel**, or **Control Transceiver Link** to directly control the PMA settings while running.

Related Information

Configuring PMA Analog Setting Control on page 11-13

Toolkit GUI Setting Reference

The following settings are available on the **Receiver Channels**, **Transmitter Channels**, and **Transceiver Links** tabs for interaction with the transmitter or receiver channels or transceiver links in the Transceiver Toolkit GUI.

Table 11-5: Transceiver Toolkit Control Panel Settings

Setting	Description	Control Panel
1-D EyeQ mode	Ignores the vertical step settings for EyeQ. This feature should be used when **DFE** is on and the **EyeQ mode** is set to **Bathtub curve**.	Receiver Transceiver Link
Alias	Name you choose for the channel.	Transmitter Receiver Transceiver Link
Auto sweep status	Reports the current and best tested bits, errors, bit error rate, and case count for the current auto sweep test.	Receiver Transceiver Link
Bit error rate (BER)	Specifies errors divided by bits tested since the last reset of the checker.	Receiver Transceiver Link
Channel address	Logical address number of the transceiver channel.	Transmitter Receiver Transceiver Link
Checker mode	Specify **Data pattern checker** or **Serial bit comparator** for BER tests. If you enable **Serial bit comparator** the Data Pattern Generator sends the PRBS pattern, but the pattern is checked by the serial bit comparator. In **Bypass mode**, clicking **Start** begins counting on the Serial bit comparator.	Receiver Transceiver Link

Setting	Description	Control Panel
Data rate	Data rate of the channel as read from the project file or data rate as measured by the frequency detector. To use the frequency detector, turn on **Enable Frequency Counter** in the Data Pattern Checker IP core and/or Data Pattern Generator IP core, regenerate the IP cores, and recompile the design. The measured data rate depends on the Avalon management clock frequency as read from the project file. Click the refresh button next to the measured **Data rate** if you make changes to your settings and want to sample the data rate again.	Transmitter Receiver Transceiver Link
DC gain	Circuitry that provides an equal boost to the incoming signal across the frequency spectrum.	Receiver Transceiver Link
DFE mode and tap values 1-5	Decision feedback equalization (DFE) for improving signal quality. One-time mode DFE determines the best tap settings and stops searching. There is also a one-time adaptive mode button that automatically turns on one-time mode and immediately populates converged values into the manual settings lists. Adaptive mode DFE automatically tries to find the best tap values.	Receiver Transceiver Link
Enable word aligner	Forces the transceiver channel to align to the word you specify.	Receiver Transceiver Link
Equalization control	Boosts the high-frequency gain of the incoming signal, thereby compensating for the low-pass filter effects of the physical medium. AEQ one-time adaptation is supported in Auto Sweep. When used with DFE, you need to use DFE triggered mode or DFE continuous.	Receiver Transceiver Link
Equalization mode	Adaptive equalization (AEQ) automatically evaluates and selects the best combination of equalizer settings. The setting applies only to Stratix V devices. When turned on, it automatically turns off Equalization Control. The one-time selection determines the best setting and stops searching. You can use AEQ for multiple, independently controlled receiver channels.	Receiver Transceiver Link

Setting	Description	Control Panel
Error rate limit	Turns on or off error rate limits. **Start checking after** waits until the set number of bits are satisfied until it starts looking at the bit error rate (BER) for the next two checks. **Bit error rate achieves below** sets upper bit error rate limits. If the error rate is better than the set error rate, the test ends. **Bit error rate exceeds** Sets lower bit error rate limits. If the error rate is worse than the set error rate, the test ends.	Receiver Transceiver Link
EyeQ mode	Allows you to specify Eye contour or Bathtub curve as the type of EyeQ graph generated by the test.	Transmitter Receiver Transceiver Link
EyeQ phase step	Sets the phase step for sampling the data from an offset of the CDR (clock data recovery) data path; set to Off to use the regular clock data recovery (CDR) data path.	Receiver Transceiver Link
EyeQ status	Displays a graphical representation of signal integrity as an eye contour or bathtub curve plot.	Transmitter Receiver Transceiver Link
EyeQ vertical step	Sets the voltage threshold of the sampler to report the height of the eye. Negative numbers are allowed for vertical steps to capture asymmetric eye.	Receiver Transceiver Link
Horizontal phase step interval	Specify the number of horizontal steps to increment when performing a sweep. Increasing the value increases the speed of the test but at a lower resolution. This option only applies to eye contour.	Transmitter Receiver Transceiver Link
Increase test range	Right-click in the **Advanced** panel to use the span capabilities of Auto Sweep to automatically increase the span of tests by one unit down for the minimum and one unit up for the maximum, for the selected set of controls. You can span either PMA Analog controls (non-DFE controls), or the DFE controls. You can quickly set up a test to check if any PMA setting combinations near your current best could yield better results.	Receiver Transceiver Link
Inject Error	Flips one bit to the output of the data pattern generator to introduce an artificial error.	Transmitter Transceiver Link

Setting	Description	Control Panel
Maximum tested bits	Sets the maximum number of tested bits for each test iteration.	Receiver Transceiver Link
Number of bits tested	Specifies the number of bits tested since the last reset of the checker.	Receiver Transceiver Link
Number of error bits	Specifies the number of error bits encountered since the last reset of the checker.	Receiver Transceiver Link
Number of preamble beats	The number of clock cycles to which the preamble word is sent before the test pattern begins.	Transmitter Transceiver Link
PLL refclk freq	Channel reference clock frequency as read from the project file or measured reference clock frequency as calculated from the measured data rate.	Transmitter Receiver Transceiver Link
Populate with	Right-click in the **Advanced** panel to load current values on the device as a starting point, or initially load the best settings determined through auto sweep. The Quartus II software automatically applies the values you specify in the drop-down lists for the Transmitter settings and Receiver settings.	Receiver Transceiver Link
Preamble word	Word to send out if the preamble mode is used.	Transmitter Transceiver Link
Pre-emphasis	The programmable pre-emphasis module in each transmit buffer boosts high frequencies in the transmit data signal, which may be attenuated in the transmission media. Using pre-emphasis can maximize the data eye opening at the far-end receiver.	Transmitter Transceiver Link
Receiver channel	Specifies the name of the selected receiver channel.	Receiver Transceiver Link
Reset	Resets the current test.	Receiver Transceiver Link

Setting	Description	Control Panel
Rules Based Configuration (RBC) validity checking	Displays invalid combination of settings in red in each list under **Transmitter settings** and **Receiver settings**, based on previous settings. If selected, the settings remain in red to indicate the currently selected combination is invalid. This feature helps you to avoid manually testing invalid settings that cannot be compiled into your design. Also, this feature helps to prevent you from setting the device into an invalid mode for extended periods of time and potentially damaging the circuits.	Receiver Transceiver Link
Run length	Sets coverage parameters for test runs.	Transmitter Receiver Transceiver Link
Run table	Lists the statistics of each EyeQ test run. The run table is sortable. You can right-click any of the tests in the table and then click **Apply Settings to Device** to quickly apply the chosen PMA settings to your device. You can click **Import** to load reports from previously generated EyeQ runs into the run table. You can click **Export** to export single or multiple runs from the run table to a report.	Transmitter Receiver Transceiver Link
RX CDR PLL status	Shows the receiver in lock-to-reference (LTR) mode. When in auto-mode, if data cannot be locked, this signal alternates in LTD mode if the CDR is locked to data.	Receiver Transceiver Link
RX CDR data status	Shows the receiver in lock-to-data (LTD) mode. When in auto-mode, if data cannot be locked, the signal stays high when locked to data and never toggles.	Receiver Transceiver Link
Serial loopback enabled	Inserts a serial loopback before the buffers, allowing you to form a link on a transmitter and receiver pair on the same physical channel of the device.	Transmitter Receiver Transceiver Link
Start	Starts the pattern generator or checker on the channel to verify incoming data.	Transmitter Receiver Transceiver Link
Stop	Stops generating patterns and testing the channel.	Transmitter Receiver Transceiver Link

Setting	Description	Control Panel
Target bit error rate	Finds the contour edge of the bit error rate that you select. This option only applies to eye contour mode.	Transmitter Receiver Transceiver Link
Test mode	Allows you to specify the **Auto sweep**, **EyeQ**, or **Auto sweep and EyeQ** test mode.	Receiver Transceiver Link
Test pattern	Test pattern sent by the transmitter channel. Options include **PRBS7, PRBS15, PRBS23, PRBS31, LowFrequency,** and **HighFrequency** and **Bypass mode**. The Data Pattern Checker self-aligns both high and low frequency patterns. Use **Bypass mode** to send user-design data.	Transmitter Receiver Transceiver Link
Time limit	Specifies the time limit unit and value to have a maximum bounds time limit for each test iteration	Receiver Transceiver Link
Transmitter channel	Specifies the name of the selected transmitter channel.	Transmitter Transceiver Link
TX/CMU PLL status	Provides status of whether the transmitter channel PLL is locked to the reference clock.	Transmitter Transceiver Link
Use preamble upon start	If turned on, sends the preamble word before starting of the test pattern. If turned off, starts sending the test pattern immediately.	Transmitter Transceiver Link
Vertical phase step interval	Specify the number of vertical steps to increment when performing a sweep. Increasing the value increases the speed of the test but at a lower resolution. This option only applies to eye contour.	Transmitter Receiver Transceiver Link
V_{OD} control	Programmable transmitter differential output voltage.	Transmitter Transceiver Link

Scripting API

You can alternatively use Tcl commands to access Transceiver Toolkit functions, rather than using the GUI. You can script various tasks, such as loading a project, creating design instances, linking device resources, and identifying high-speed serial links. You can save your project setup in a Tcl script for use in subsequent testing sessions. You can also build a custom test routine script.

After you set up and define links that describe the entire physical system, you can click Save Tcl Script to save the setup for future use. To run the scripts, double-click script names in the System Explorer scripts folder.

View a list of the available Tcl commands in the Tcl Console window. Select Tcl commands in the list to view descriptions, including example usage.

To view Tcl command descriptions from the Tcl Console window:

1. Type `help help`. The Console displays all Transceiver Toolkit Tcl commands.
2. Type `help` <*command name*>. The Console displays the command description.

The following Tcl command example uses the following commands to locate and open an instance of the master service.

- `set`—Tcl command that sets the value of a variable.
- `m_path`—name of a variable.
- `lindex`—Tcl command that retrieves an element from a list.
- `get_service_paths`—returns a list of service instances by type.
- `master`—service type that accesses memory-mapped slaves.
- `n`—number of the master service.
- `open_service`—initiates a service instance.

```
set m_path [lindex [get_service_paths master] n]
open_service master $m_path
```

Transceiver Toolkit Commands

The following tables list the available Transceiver Toolkit scripting commands.

Table 11-6: Transceiver Toolkit Channel_rx Commands

Command	Arguments	Function
`transceiver_channel_rx_get_data`	<*service-path*>	Returns a list of the current checker data. The results are in the order of number of bits, number of errors, and bit error rate.
`transceiver_channel_rx_get_dcgain`	<*service-path*>	Gets the DC gain value on the receiver channel.
`transceiver_channel_rx_get_dfe_tap_value`	<*service-path*> <*tap position*>	Gets the current tap value of the specified channel at the specified tap position.
`transceiver_channel_rx_get_eqctrl`	<*service-path*>	Gets the equalization control value on the receiver channel.
`transceiver_channel_rx_get_pattern`	<*service-path*>	Returns the current data checker pattern by name.
`transceiver_channel_rx_has_dfe`	<*service-path*>	Gets whether this channel has the DFE feature available.
`transceiver_channel_rx_has_eyeq`	<*service-path*>	Gets whether the EyeQ feature is available for the specified channel.

Command	Arguments	Function
`transceiver_channel_rx_is_checking`	*<service-path>*	Returns non-zero if the checker is running.
`transceiver_channel_rx_is_dfe_enabled`	*<service-path>*	Gets whether the DFE feature is enabled on the specified channel.
`transceiver_channel_rx_is_locked`	*<service-path>*	Returns non-zero if the checker is locked onto the incoming data.
`transceiver_channel_rx_reset_counters`	*<service-path>*	Resets the bit and error counters inside the checker.
`transceiver_channel_rx_reset`	*<service-path>*	Resets the specified channel.
`transceiver_channel_rx_set_dcgain`	*<service-path> <value>*	Sets the DC gain value on the receiver channel.
`transceiver_channel_rx_set_dfe_enabled`	*<service-path> <disable(0)/enable(1)>*	Enables or disables the DFE feature on the specified channel.
`transceiver_channel_rx_set_dfe_tap_value`	*<service-path> <tap position> <tap value>*	Sets the current tap value of the specified channel at the specified tap position to the specified value.
`transceiver_channel_rx_get_dfe_adaptive`	*<service-path>*	Gets the mode of DFE adaptation. 0=off, 1=adaptive, 2= one-time adaptive
`transceiver_channel_rx_set_dfe_adaptive`	*<service-path>*	Sets the mode of DFE adaptation. 0=off, 1=adaptive, 2= one-time adaptive
`transceiver_channel_rx_set_eqctrl`	*<service-path> <value>*	Sets the equalization control value on the receiver channel.
`transceiver_channel_rx_start_checking`	*<service-path>*	Starts the checker.
`transceiver_channel_rx_stop_checking`	*<service-path>*	Stops the checker.
`transceiver_channel_rx_get_eyeq_phase_step`	*<service-path>*	Gets the current phase step of the specified channel.
`transceiver_channel_rx_set_pattern`	*<service-path> <pattern-name>*	Sets the expected pattern to the one specified by the pattern name.
`transceiver_channel_rx_is_eyeq_enabled`	*<service-path>*	Gets whether the EyeQ feature is enabled on the specified channel.

Command	Arguments	Function
`transceiver_channel_rx_set_eyeq_enabled`	*<service-path> <disable(0)/enable(1)>*	Enables or disables the EyeQ feature on the specified channel.
`transceiver_channel_rx_set_eyeq_phase_step`	*<service-path> <phase step>*	Sets the phase step of the specified channel.
`transceiver_channel_rx_set_word_aligner_enabled`	*<service-path> <disable(0)/enable(1)>*	Enables or disables the word aligner of the specified channel.
`transceiver_channel_rx_is_word_aligner_enabled`	*<service-path> <disable(0)/enable(1)>*	Gets whether the word aligner feature is enabled on the specified channel.
`transceiver_channel_rx_is_locked`	*<service-path>*	Returns non-zero if the checker is locked onto the incoming signal.
`transceiver_channel_rx_is_rx_locked_to_data`	*<service-path>*	Returns `1` if transceiver is in lock to data (LTD) mode. Otherwise `0`.
`transceiver_channel_rx_is_rx_locked_to_ref`	*<service-path>*	Returns `1` if transceiver is in lock to reference (LTR) mode. Otherwise `0`.
`transceiver_channel_rx_has_eyeq_1d`	*<service-path>*	Detects whether the eye viewer pointed to by *<service-path>* supports 1D-EyeQ mode.
`transceiver_channel_rx_set_1deye_mode`	*<service-path> <disable(0)/enable(1)>*	Enables or disables 1D-EyeQ mode.
`transceiver_channel_rx_get_1deye_mode`	*<service-path>*	Returns the current on or off status of 1D-EyeQ mode.

Table 11-7: Transceiver Toolkit Channel _tx Commands

Command	Arguments	Function
`transceiver_channel_tx_disable_preamble`	*<service-path>*	Disables the preamble mode at the beginning of generation.
`transceiver_channel_tx_enable_preamble`	*<service-path>*	Enables the preamble mode at the beginning of generation.
`transceiver_channel_tx_get_number_of_preamble_beats`	*<service-path>*	Returns the currently set number of beats to send out the preamble word.
`transceiver_channel_tx_get_pattern`	*<service-path>*	Returns the currently set pattern.

Command	Arguments	Function
`transceiver_channel_tx_get_preamble_word`	*<service-path>*	Returns the currently set preamble word.
`transceiver_channel_tx_get_preemph0t`	*<service-path>*	Gets the pre-emphasis pre-tap value on the transmitter channel.
`transceiver_channel_tx_get_preemph1t`	*<service-path>*	Gets the pre-emphasis first post-tap value on the transmitter channel.
`transceiver_channel_tx_get_preemph2t`	*<service-path>*	Gets the pre-emphasis second post-tap value on the transmitter channel.
`transceiver_channel_tx_get_vodctrl`	*<service-path>*	Gets the V_{OD} control value on the transmitter channel.
`transceiver_channel_tx_inject_error`	*<service-path>*	Injects a 1-bit error into the generator's output.
`transceiver_channel_tx_is_generating`	*<service-path>*	Returns non-zero if the generator is running.
`transceiver_channel_tx_is_preamble_enabled`	*<service-path>*	Returns non-zero if preamble mode is enabled.
`transceiver_channel_tx_set_number_of_preamble_beats`	*<service-path> <number-of-preamble-beats>*	Sets the number of beats to send out the preamble word.
`transceiver_channel_tx_set_pattern`	*<service-path> <pattern-name>*	Sets the output pattern to the one specified by the pattern name.
`transceiver_channel_tx_set_preamble_word`	*<service-path> <preamble-word>*	Sets the preamble word to be sent out.
`transceiver_channel_tx_set_preemph0t`	*<service-path>* *<preemph0t value>*	Sets the pre-emphasis pre-tap value on the transmitter channel.
`transceiver_channel_tx_set_preemph1t`	*<service-path>* *<preemph1t value>*	Sets the pre-emphasis first post-tap value on the transmitter channel.
`transceiver_channel_tx_set_preemph2t`	*<service-path>* *<preemph2t value>*	Sets the pre-emphasis second post-tap value on the transmitter channel.
`transceiver_channel_tx_set_vodctrl`	*<service-path>* *< vodctrl value>*	Sets the V_{OD} control value on the transmitter channel.

Command	Arguments	Function
`transceiver_channel_tx_start_generation`	<*service-path*>	Starts the generator.
`transceiver_channel_tx_stop_generation`	<*service-path*>	Stops the generator.

Table 11-8: Transceiver Toolkit Transceiver Toolkit Debug_Link Commands

Command	Arguments	Function
`transceiver_debug_link_get_pattern`	<*service-path*>	Gets the currently set pattern the link uses to run the test.
`transceiver_debug_link_is_running`	<*service-path*>	Returns non-zero if the test is running on the link.
`transceiver_debug_link_set_pattern`	<*service-path*> <*data pattern*>	Sets the pattern the link uses to run the test.
`transceiver_debug_link_start_running`	<*service-path*>	Starts running a test with the currently selected test pattern.
`transceiver_debug_link_stop_running`	<*service-path*>	Stops running the test.

Table 11-9: Transceiver Toolkit Reconfig_Analog Commands

Command	Arguments	Function
`transceiver_reconfig_analog_get_logical_channel_address`	<*service-path*>	Gets the transceiver logical channel address currently set.
`transceiver_reconfig_analog_get_rx_dcgain`	<*service-path*>	Gets the DC gain value on the receiver channel specified by the current logical channel address.
`transceiver_reconfig_analog_get_rx_eqctrl`	<*service-path*>	Gets the equalization control value on the receiver channel specified by the current logical channel address.
`transceiver_reconfig_analog_get_tx_preemph0t`	<*service-path*>	Gets the pre-emphasis pre-tap value on the transmitter channel specified by the current logical channel address.

Command	Arguments	Function
`transceiver_reconfig_analog_get_tx_preemph1t`	*<service-path>*	Gets the pre-emphasis first post-tap value on the transmitter channel specified by the current logical channel address.
`transceiver_reconfig_analog_get_tx_preemph2t`	*<service-path>*	Gets the pre-emphasis second post-tap value on the transmitter channel specified by the current logical channel address.
`transceiver_reconfig_analog_get_tx_vodctrl`	*<service-path>*	Gets the V_{OD} control value on the transmitter channel specified by the current logical channel address.
`transceiver_reconfig_analog_set_logical_channel_address`	*<service-path>* *<logical channel address>*	Sets the transceiver logical channel address.
`transceiver_reconfig_analog_set_rx_dcgain`	*<service-path>* *< dc_gain value>*	Sets the DC gain value on the receiver channel specified by the current logical channel address
`transceiver_reconfig_analog_set_rx_eqctrl`	*<service-path>* *< eqctrl value>*	Sets the equalization control value on the receiver channel specified by the current logical channel address.
`transceiver_reconfig_analog_set_tx_preemph0t`	*<service-path>* *<preemph0t value>*	Sets the pre-emphasis pre-tap value on the transmitter channel specified by the current logical channel address.
`transceiver_reconfig_analog_set_tx_preemph1t`	*<service-path>* *<preemph1t value>*	Sets the pre-emphasis first post-tap value on the transmitter channel specified by the current logical channel address.
`transceiver_reconfig_analog_set_tx_preemph2t`	*<service-path>* *<preemph2t value>*	Sets the pre-emphasis second post-tap value on the transmitter channel specified by the current logical channel address.
`transceiver_reconfig_analog_set_tx_vodctrl`	*<service-path>* *< vodctrl value>*	Sets the V_{OD} control value on the transmitter channel specified by the current logical channel address.

.

Table 11-10: Transceiver Toolkit Decision Feedback Equalization (DFE) Commands

Command	Arguments	Function
`alt_xcvr_reconfig_dfe_get_logical_channel_address`	*<service-path>*	Gets the logical channel address that other `alt_xcvr_reconfig_dfe` commands use to apply.
`alt_xcvr_reconfig_dfe_is_enabled`	*<service-path>*	Gets whether the DFE feature is enabled on the previously specified channel.
`alt_xcvr_reconfig_dfe_set_enabled`	*<service-path> <disable(0)/enable(1)>*	Enables or disables the DFE feature on the previously specified channel.
`alt_xcvr_reconfig_dfe_set_logical_channel_address`	*<service-path> <logical channel address>*	Sets the logical channel address that other `alt_xcvr_reconfig_eye_viewer` commands use.
`alt_xcvr_reconfig_dfe_set_tap_value`	*<service-path> <tap position> <tap value>*	Sets the tap value at the previously specified channel at specified tap position and value.

Table 11-11: Transceiver Toolkit Eye Monitor Commands

Command	Arguments	Function
`alt_xcvr_custom_is_word_aligner_enabled`	*<service-path> <disable(0)/enable(1)>*	Gets whether the word aligner feature is enabled on the previously specified channel.
`alt_xcvr_custom_set_word_aligner_enabled`	*<service-path> <disable(0)/enable(1)>*	Enables or disables the word aligner of the previously specified channel.
`alt_xcvr_custom_is_rx_locked_to_data`	*<service-path>*	Returns whether the receiver CDR is locked to data.
`alt_xcvr_custom_is_rx_locked_to_ref`	*<service-path>*	Returns whether the receiver CDR PLL is locked to the reference clock.
`alt_xcvr_custom_is_serial_loopback_enabled`	*<service-path>*	Returns whether the serial loopback mode of the previously specified channel is enabled.

Command	Arguments	Function
`alt_xcvr_custom_set_serial_loopback_enabled`	<*service-path*> <*disable(0)/enable(1)*>	Enables or disables the serial loopback mode of the previously specified channel.
`alt_xcvr_custom_is_tx_pll_locked`	<*service-path*>	Returns whether the transmitter PLL is locked to the reference clock.
`alt_xcvr_reconfig_eye_viewer_get_logical_channel_address`	<*service-path*>	Gets the logical channel address on which other `alt_reconfig_eye_viewer` commands will use to apply.
`alt_xcvr_reconfig_eye_viewer_get_phase_step`	<*service-path*>	Gets the current phase step of the previously specified channel.
`alt_xcvr_reconfig_eye_viewer_is_enabled`	<*service-path*>	Gets whether the EyeQ feature is enabled on the previously specified channel.
`alt_xcvr_reconfig_eye_viewer_set_enabled`	<*service-path*> <*disable(0)/enable(1)*>	Enables or disables the EyeQ feature on the previously specified channel. Setting a value of 2 enables both EyeQ and the Serial Bit Comparator.
`alt_xcvr_reconfig_eye_viewer_set_logical_channel_address`	<*service-path*> <*logical channel address*>	Sets the logical channel address on which other `alt_reconfig_eye_viewer` commands will use to apply.
`alt_xcvr_reconfig_eye_viewer_set_phase_step`	<*service-path*> <*phase step*>	Sets the phase step of the previously specified channel.
`alt_xcvr_reconfig_eye_viewer_has_ber_checker`	<*service-path*>	Detects whether the eye viewer pointed to by <*service-path*> supports the Serial Bit Comparator.
`alt_xcvr_reconfig_eye_viewer_ber_checker_is_enabled`	<*service-path*>	Detects whether the Serial Bit Comparator is enabled.
`alt_xcvr_reconfig_eye_viewer_ber_checker_start`	<*service-path*>	Starts the Serial Bit Comparator counters.

Command	Arguments	Function
`alt_xcvr_reconfig_eye_viewer_ber_checker_stop`	<*service-path*>	Stops the Serial Bit Comparator counters.
`alt_xcvr_reconfig_eye_viewer_ber_checker_reset_counters`	<*service-path*>	Resets the Serial Bit Comparator counters.
`alt_xcvr_reconfig_eye_viewer_ber_checker_is_running`	<*service-path*>	Gets whether the Serial Bit Comparator counters are currently running or not.
`alt_xcvr_reconfig_eye_viewer_ber_checker_get_data`	<*service-path*>	Gets the current total bit, error bit, and exception counts for the Serial Bit Comparator.
`alt_xcvr_reconfig_eye_viewer_has_1deye`	<*service-path*>	Detects whether the eye viewer pointed to by <*service-path*> supports 1D-EyeQ mode.
`alt_xcvr_reconfig_eye_viewer_set_1deye_mode`	<*service-path*> <*disable(0)/enable(1)*	Enables or disables 1D-EyeQ mode.
`alt_xcvr_reconfig_eye_viewer_get_1deye_mode`	<*service-path*>	Gets the enable or disabled state of 1D-EyeQ mode.

Table 11-12: Channel Type Commands

Command	Arguments	Function
`get_channel_type`	<*service-path*> <*logical-channel- num* >	Reports the detected type (GX/GT) of channel <*logical-channel-num* > for the reconfiguration block located at <*service-path*>.
`set_channel_type`	<*service-path*> <*logical-channel- num* > <*channel-type*>	Overrides the detected channel type of channel <*logical-channel-num* > for the reconfiguration block located at <*service-path*> to the type specified (0:GX, 1:GT).

Table 11-13: Loopback Commands

Command	Arguments	Function
`loopback_get`	*<service-path>*	Returns the value of a setting or result on the loopback channel. Available results include: • Status—running or stopped. • Bytes—number of bytes sent through the loopback channel. • Errors—number of errors reported by the loopback channel. • Seconds—number of seconds since the loopback channel was started.
`loopback_set`	*<service-path>*	Sets the value of a setting controlling the loopback channel. Some settings are only supported by particular channel types. Available settings include: • Timer—number of seconds for the test run. • Size—size of the test data. • Mode—mode of the test.
`loopback_start`	*<service-path>*	Starts sending data through the loopback channel.
`loopback_stop`	*<service-path>*	Stops sending data through the loopback channel.

Data Pattern Generator Commands

You can use Data Pattern Generator commands to control data patterns for debugging transceiver channels. You must instantiate the Data Pattern Generator component to support these commands.

Table 11-14: Data Pattern Generator Commands

Command	Arguments	Function
`data_pattern_generator_start`	*<service-path>*	Starts the data pattern generator.
`data_pattern_generator_stop`	*<service-path>*	Stops the data pattern generator.

Command	Arguments	Function
`data_pattern_generator_is_generating`	<*service-path*>	Returns non-zero if the generator is running.
`data_pattern_generator_inject_error`	<*service-path*>	Injects a 1-bit error into the generator output.
`data_pattern_generator_set_pattern`	<*service-path*> <*pattern-name*>	Sets the output pattern specified by the <*pattern-name*>. In all, 6 patterns are available, 4 are pseudo-random binary sequences (PRBS), 1 is high frequency and 1 is low frequency. The PRBS7, PRBS15, PRBS23, PRBS31, HF (outputs high frequency, constant pattern of alternating 0s and 1s), and LF (outputs low frequency, constant pattern of 10b'1111100000 for 10-bit symbols and 8b'11110000 for 8-bit symbols) pattern names are defined. PRBS files are clear text and you can modify the PRBS files.
`data_pattern_generator_get_pattern`	<*service-path*>	Returns currently selected output pattern.
`data_pattern_generator_get_available_patterns`	<*service-path*>	Returns a list of available data patterns by name.
`data_pattern_generator_enable_preamble`	<*service-path*>	Enables the preamble mode at the beginning of generation.
`data_pattern_generator_disable_preamble`	<*service-path*>	Disables the preamble mode at the beginning of generation.
`data_pattern_generator_is_preamble_enabled`	<*service-path*>	Returns a non-zero value if preamble mode is enabled.
`data_pattern_generator_set_preamble_word`	<*service-path*> <*preamble-word*>	Sets the preamble word (could be 32-bit or 40-bit).
`data_pattern_generator_get_preamble_word`	<*service-path*>	Gets the preamble word.
`data_pattern_generator_set_preamble_beats`	<*service-path*> <*number-of-preamble- beats*>	Sets the number of beats to send out in the preamble word.

Command	Arguments	Function
`data_pattern_generator_get_preamble_beats`	<*service-path*>	Returns the currently set number of beats to send out in the preamble word.
`data_pattern_generator_fcnter_start`	<*service-path*> <*max-cycles*>	Sets the max cycle count and starts the frequency counter.
`data_pattern_generator_check_status`	<*service-path*>	Queries the data pattern generator for current status. Returns a bitmap indicating the status, with bits defined as follows: [0]-enabled, [1]-bypass enabled, [2]-avalon, [3]-sink ready, [4]-source valid, and [5]-frequency counter enabled.
`data_pattern_generator_fcnter_report`	<*service-path*> <*force-stop*>	Reports the current measured clock ratio, stopping the counting first depending on <*force-stop*>.

Data Pattern Checker Commands

You can use Data Pattern Checker commands to verify your generated data patterns. You must instantiate the Data Pattern Checker component to support these commands.

Table 11-15: Data Pattern Checker Commands

Command	Arguments	Function
`data_pattern_checker_start`	<*service-path*>	Starts the data pattern checker.
`data_pattern_checker_stop`	<*service-path*>	Stops the data pattern checker.
`data_pattern_checker_is_checking`	<*service-path*>	Returns a non-zero value if the checker is running.
`data_pattern_checker_is_locked`	<*service-path*>	Returns non-zero if the checker is locked onto the incoming data.
`data_pattern_checker_set_pattern`	<*service-path*> <*pattern-name*>	Sets the expected pattern to the one specified by the <*pattern-name*>.
`data_pattern_checker_get_pattern`	<*service-path*>	Returns the currently selected expected pattern by name.

Command	Arguments	Function
`data_pattern_checker_get_available_patterns`	<*service-path*>	Returns a list of available data patterns by name.
`data_pattern_checker_get_data`	<*service-path*>	Returns a list of the current checker data. The results are in the following order: number of bits, number of errors, and bit error rate.
`data_pattern_checker_reset_counters`	<*service-path*>	Resets the bit and error counters inside the checker.
`data_pattern_checker_fcnter_start`	<*service-path*> <*max-cycles*>	Sets the max cycle count and starts the frequency counter.
`data_pattern_checker_check_status`	<*service-path*>	Queries the data pattern checker for current status. Returns a bitmap indicating status, with bits defined as follows: [0]-enabled, [1]-locked, [2]-bypass enabled, [3]-avalon, [4]-sink ready, [5]-source valid, and [6]-frequency counter enabled.
`data_patte rn_checker_fcnter_report`	<*service-path*> <*force-stop*>	Reports the current measured clock ratio, stopping the counting first depending on <*force-stop*>.

Revision History

The following table shows the revision history for this chapter.

Table 11-16: Document Revision History

Date	Version	Changes
November, 2013	13.1.0	• Reorganization and conversion to DITA.
May, 2013	13.0.0	• Added Conduit Mode Support, Serial Bit Comparator, Required Files and Tcl command tables.
November, 2012	12.1.0	• Minor editorial updates. Added Tcl help information and removed Tcl command tables. Added 28-Gbps Transceiver support section.
August, 2012	12.0.1	• General reorganization and revised steps in modifying Altera example designs.
June, 2012	12.0.0	• Maintenance release for update of Transceiver Toolkit features.

Date	Version	Changes
November, 2011	11.1.0	• Maintenance release for update of Transceiver Toolkit features.
May, 2011	11.0.0	• Added new Tcl scenario.
December, 2010	10.1.0	• Changed to new document template. Added new 10.1 release features.
August, 2010	10.0.1	• Corrected links.
July 2010	10.0.0	• Initial release.

Related Information

Quartus II Handbook Archive