Document identifier: QCVS_SerDes_User_Guide
Rev. 4.x, 01/2021

QCVS SerDes Tool User Guide

Contents

Chapter 1
SerDes Configuration and Validation

This document describes the capabilities of the SerDes configuration and validation tool (SerDes stands for *Serializer/Deserializer*) for the QorIQ devices.

The SerDes tool is a part of the QorIQ Configuration and Validation Suite (QCVS) product.

This chapter contains the following sections:

- Introduction
- SerDes configuration
- SerDes validation
- How to use a SerDes configuration
- Licensing

1.1 Introduction

The SerDes tool allows you to configure the SerDes block and provides you a GUI application to validate the configuration.

You can start with a default and arbitrary configuration or connect to your board and read its current SerDes configuration. The validation features of the tool allows you to exercise the SerDes built-in test capabilities (for example, BIST, Jitter scope, and Tx pattern generation) and view the results. The validation features are licensed. For more details, see Licensing.

The SerDes tool supports numerous QorIQ devices. See QCVS Release Notes for details on which QorIQ SoCs are supported by the SerDes tool.

This section contains the following subsections:

- Acronyms
- SerDes hardware block challenges
- Creating a SerDes project
- Working with SerDes components

1.1.1 Acronyms

This section provides a list of all the acronyms used in the current document.

Table 1. Acronyms

Acronym	Meaning
BIST	Built-in self-test
GUI	Graphical user interface
IP	Internet protocol
PBL	Pre-boot loader
PCIe	Peripheral component interconnect express
PLL	Phase-locked loop
QCVS	QorIQ Configuration and Validation Suite

Table continues on the next page...

Table 1. Acronyms (continued)

Acronym	Meaning
RCW	Reset configuration word
Rx	Receive/receiver
SATA	Serial advanced technology attachment
SerDes	Serializer/Deserializer
SoC	System-on-chip
Tx	Transmit/transmitter
UI	User interface

1.1.2 SerDes hardware block challenges

SerDes is a hardware block that is used for high-speed data transmission.

The SerDes hardware block is present in most of the SoCs that need to enable a reliable and high throughput for different types of traffic coming from different peripherals, such as PCIe, SATA, and so on.

Having an optimally configured and validated SerDes block is critical for designing a board successfully. This involves not only planning what protocols and speeds can and will be used, but also setting that configuration via the RCW. It also involves configuring electrical behaviors of lanes via memory-mapped registers to get reliable and optimal performance. The latter requires driving traffic through the lanes and analyzing the result. Fortunately, the SerDes blocks on most QorIQ SoCs have characterization circuitry to support this, without expensive scope or other testing equipment.

An optimal SerDes configuration is usually obtained by tweaking a configuration, examining the effect on transmission, and repeating until the best configuration has been identified. The SerDes tool allows doing all this with a rich, user-friendly GUI.

1.1.3 Creating a SerDes project

The first step to configure or validate SerDes is to create a QCVS project with a SerDes component.

To create a SerDes project, perform these steps:

1. Launch QCVS.
2. Select **File > New > QorIQ Configuration Project**. The **New QorIQ Configuration Project** wizard starts, displaying the **Create a QorIQ Configuration Project** page.
3. Specify the project name in the **Project name** text box, and click **Next**.
4. On the **Devices** page, choose a device to work with. See QCVS Release Notes for details on which QorIQ SoCs are supported by the SerDes tool.
5. Now, choose a device revision. If only one revision is available, it is automatically chosen. Click **Next** to continue.
6. On the **Toolset selection** page, select **SERDES Configuration Tool**, and click **Next** to continue.
7. On the **SERDES Configuration** page, the following two options are available to create a SerDes component (as shown in the next figure):
   - **Create default configuration**: Creates a default SerDes configuration with predefined SerDes settings.
   - **Read from target**: The SerDes configuration will be read from the board you are connected to. If you select this option, you also need to perform the following steps:
     a. Specify the target connection parameters, including the probe type and connection string.
     b. Click the **Read from target** button, and see if a message appears in the adjacent text box indicating that the read operation was successful.

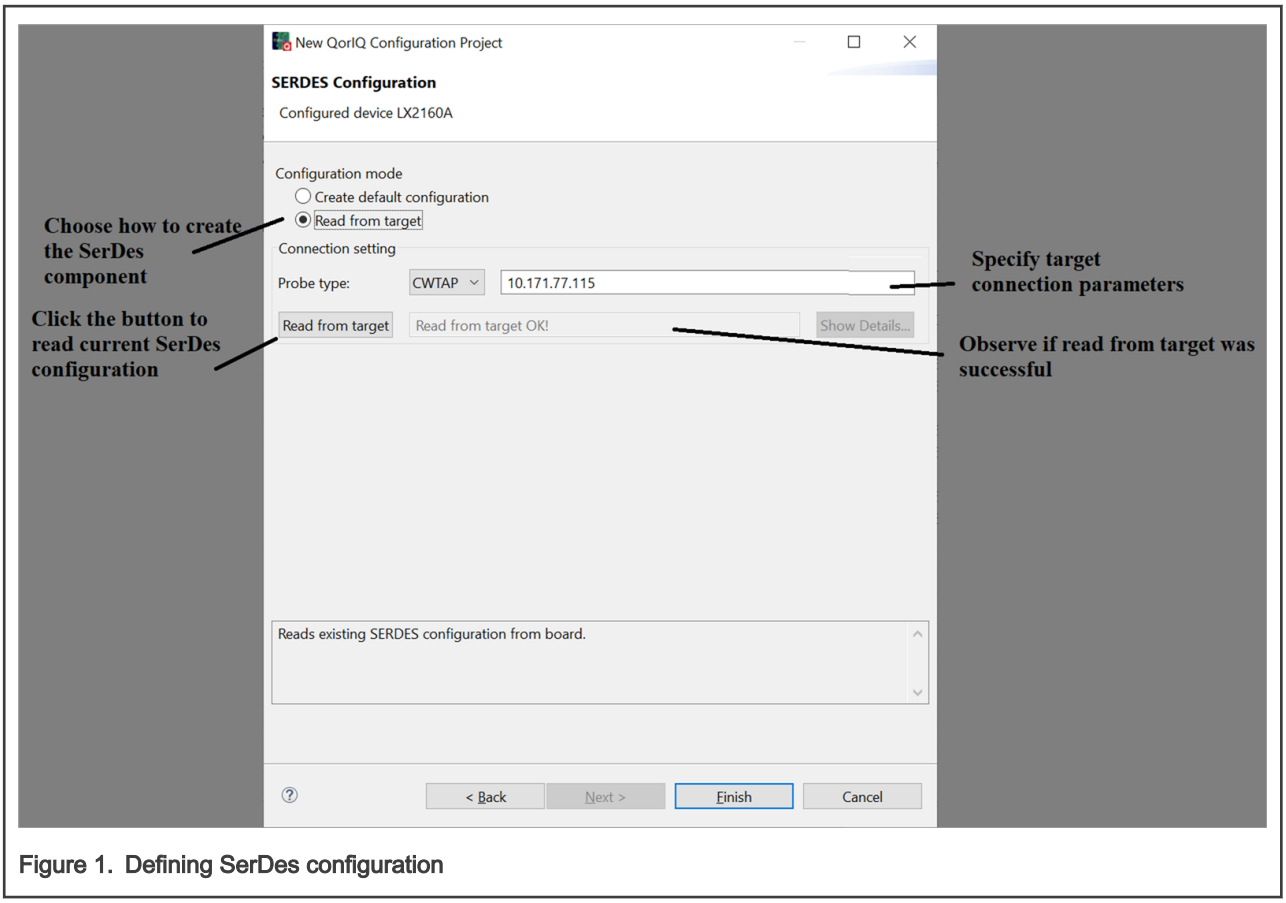


Figure 1. Defining SerDes configuration

8. Click **Finish** to create the project.

When a new project is created, each of the SerDes modules defined for the selected device is represented as a SerDes component in the project. The next section shows how to use a SerDes component to configure and validate the SerDes block.

1.1.4 Working with SerDes components

When a new project is created, the SerDes components are added in the **Components** view.

The figure below shows four SerDes components added in the **Components** view. Each component corresponds to one SerDes module. The LX2160A SoC has four SerDes modules. As shown in the figure, all four components are grouped under the parent SerDes block component.

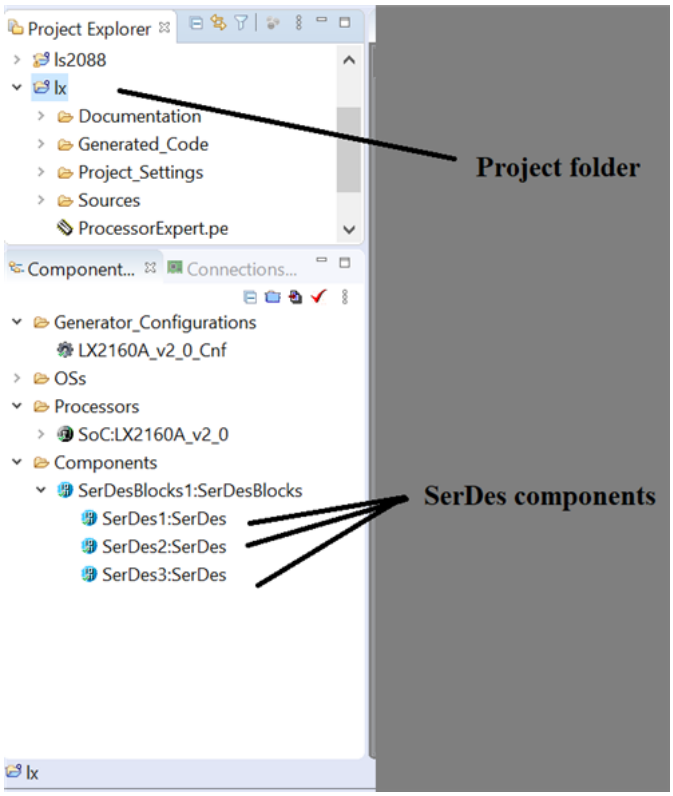


Figure 2. Components view

The configuration and validation of SerDes is done using the **Component Inspector** view. To show the **Component Inspector** view, either double-click a SerDes component, or right-click a SerDes component and choose **Inspector** from the shortcut menu. The figure below shows the areas of the **Component Inspector** view and how it fits into the overall SerDes GUI.

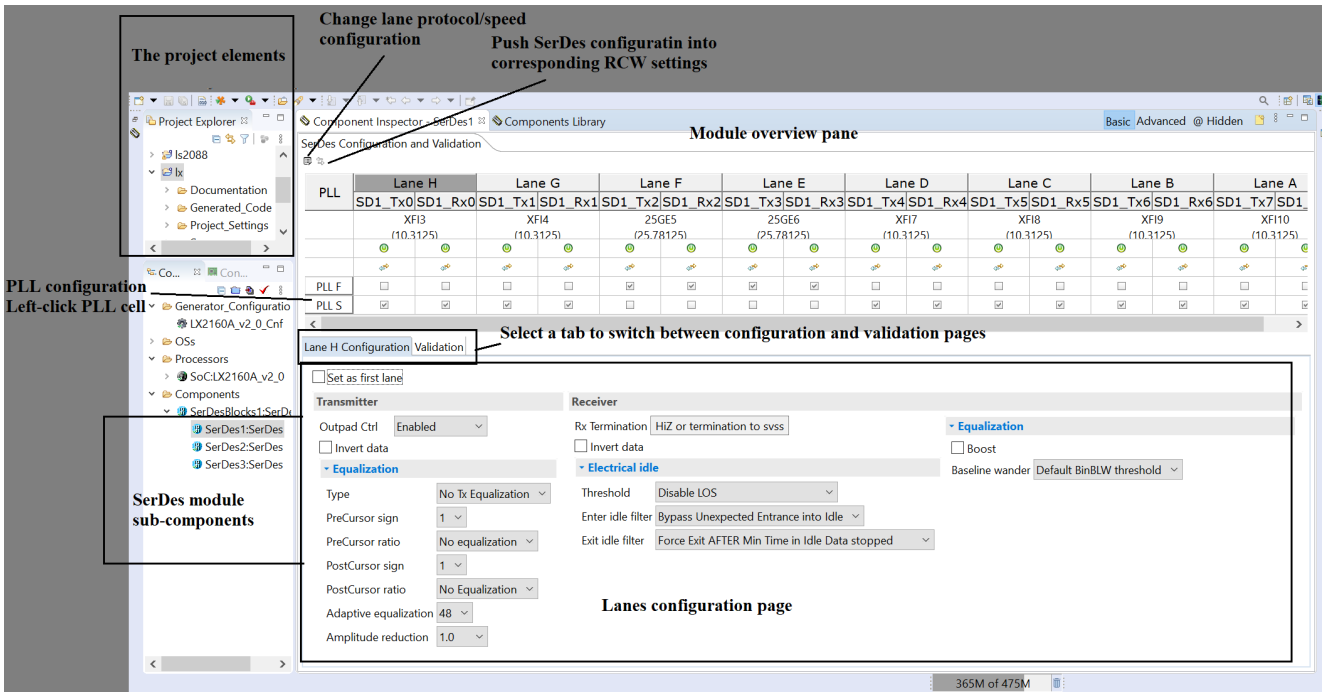


Figure 3. SerDes GUI

1.2 SerDes configuration

This section describes how to configure the SerDes block.

An SoC can have several SerDes modules. Each SerDes module is organized into entities called *lanes*. The number of lanes varies with the device type.

Configuring SerDes means effectively configuring its lanes. Configuring a lane involves specifying:

- Protocol and speed: Defines what kind of traffic and at which speed the lane can operate with
- PLL used by the lane
- Transmit and receive equalization and electrical parameters. These are more in-depth configuration parameters that can affect the quality of transmission on the lane.

There are many constraints that limit what protocols, speeds, and PLLs can be used on each lane. The constraints can be grouped into the following three categories:

- Per lane constraints
- Constraints across lanes
- Constraints across SerDes modules

All these constraints are known and applied by the SerDes tool, making configuration of the SerDes modules not only easier but also less error prone.

The SerDes configuration user interface allows you to change various SerDes parameters that map to the RCW bits and/or SerDes memory-mapped register values.

This section contains the following subsections:

- Module Overview pane
- Lane Configuration pane

- PLL Configuration pane
- Protocol/Speed Configuration window
- Configuration Registers view
- Errata support

1.2.1 Module Overview pane

The Module Overview pane provides an overview of all the lanes in the currently selected SerDes module.

This pane contains a table, called lanes overview table, which summarizes the protocol, speed, and PLL allocation for each lane. The Module Overview pane allows you to:

- Power up/down or reset a lane receiver and/or transmitter. If a lane is in the Power Up state, then it is displayed with a green icon; if it is in the Power Down state, then it is displayed with a red icon.
- Open the **Protocol/Speed Configuration** window (see Protocol/Speed Configuration window for details)
- Change advanced lane electrical behaviors that affect transmission quality
- Apply the protocol/speed/PLL configuration to the RCW in the PBL component of the project

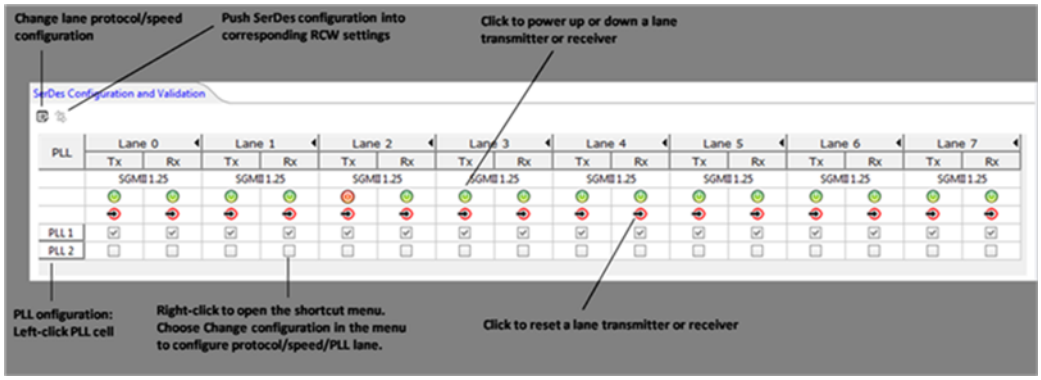


Figure 4. Module Overview pane

1.2.2 Lane Configuration pane

The Lane Configuration pane gives you access to various advanced SerDes configuration options, which are related to the transmitter and receiver electrical and equalization parameters.

See *SerDes Configuration and Validation Tool Companion Application Note* (AN5119) for more details on how these values should be changed.

The Lane Configuration pane shows the configuration of a single lane at a time. To configure a lane, click the lane in the Module Overview pane, and then configure it in the Lane Configuration pane (see figure below).

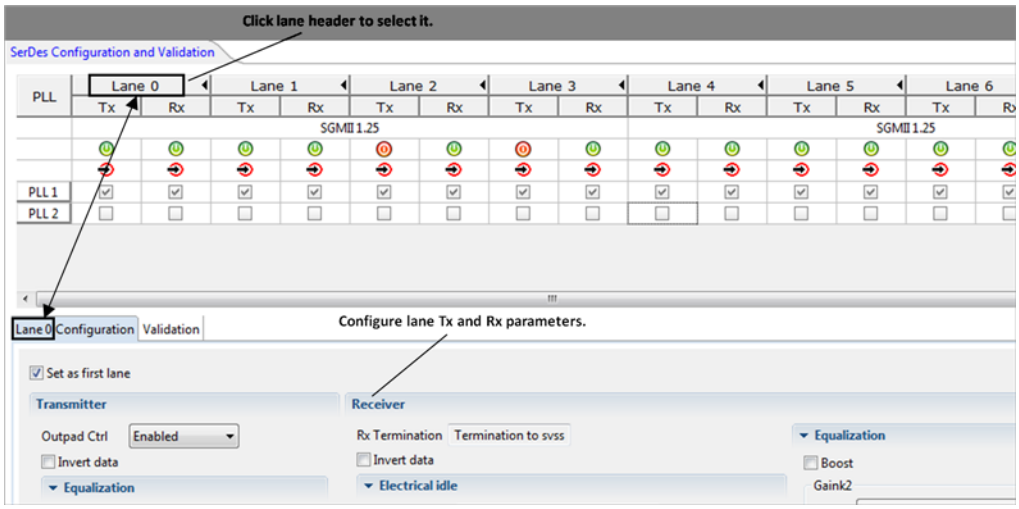


Figure 5. Lane Configuration pane

1.2.3 PLL Configuration pane

This pane allows you to view and modify PLL settings.

The Lane Configuration pane becomes the PLL Configuration pane when a PLL is selected in the Module Overview pane.

Each SerDes lane uses a PLL that is determined by the protocol and speed option described in Protocol/Speed Configuration window. The PLL used by each lane is displayed in the lanes overview table in the Read-Only mode, as shown in the figure below.

Note that for a lane, both Tx and Rx use the same PLL. You can configure a PLL by left-clicking it in the Module Overview pane. However, you should avoid altering most of these settings, as they are directly tied to what protocol and speed you chose for the module's lanes in the **Protocol/Speed Configuration** window. In other words, most PLL settings are automatically updated when you make your protocol and speed choices. Altering PLL settings in this pane can easily leave you with a configuration that makes no sense. Therefore, treat this pane as mostly informational.

The figure below shows the PLL allocation per lane, how to access the PLL Configuration pane, and the PLL Configuration pane.

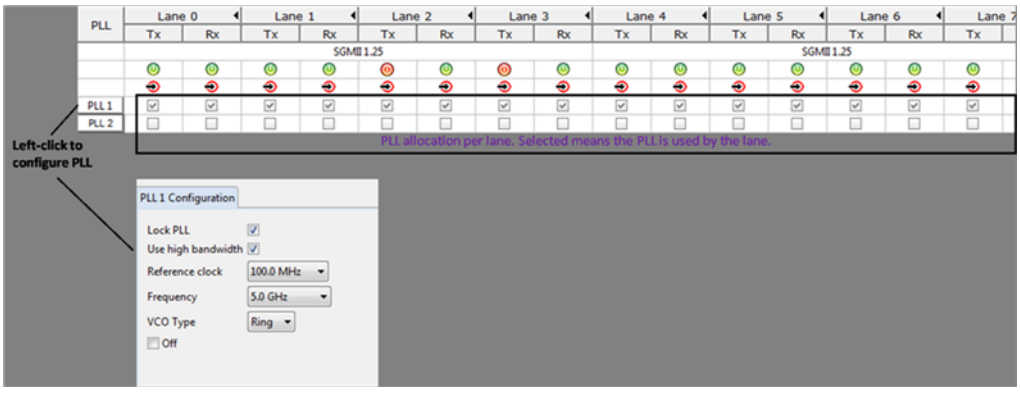


Figure 6. PLL Configuration pane

1.2.4 Protocol/Speed Configuration window

This section describes the functionality and usage of the **Protocol/Speed Configuration** window.

A QorIQ SoC is capable of simultaneously transmitting and receiving various types of serial traffic at different speeds. Each lane is an independent traffic path. There are protocol and speed lane constraints within a SerDes module and even across modules. Knowing and observing these constraints becomes difficult when relying strictly on the reference manual or errata document of the SoC. For example, answering the questions "Can my SoC support PCIe Gen 3, XFI, and SATA traffic simultaneously?" and "If so, how do I configure the SerDes blocks for that" becomes very difficult by looking at the SoC documentation. QCVS makes it easy. With the SerDes tool, the answers and configurations are a few mouse clicks away.

The **Protocol/Speed Configuration** window allows you to choose the protocols and speeds for the lanes in all your SerDes modules. In an ideal world, you could choose any protocol and speed for each lane. In reality, only a few protocols and speeds are supported for each lane. Moreover, choosing a protocol/speed for one lane typically dictates or restricts the protocols and speeds on some or all of the other lanes in the SerDes module. In some cases, it can even restrict what you can use in other SerDes modules. The end result is that for each SerDes module, you have a limited number of combinations and permutations of protocols and speeds. A SerDes module is configured to use a particular combination/permutation out of reset via RCW[SRDS_PRTCL_Sn]. The values for these RCW fields (that is, the combinations and permutations available) are typically represented in tables in the SoC reference manual. The table below shows an example from LS2085A Reference Manual.

Table 2. SRDS_PRCTL choices for SerDes2 in LS2085A Reference Manual

SRDS_PRCTL_S2	A	B	C	D	E	F	G	H	PLL mapping
hex	SD2[0]	SD2[1]	SD2[2]	SD2[3]	SD2[4]	SD2[5]	SD2[6]	SD2[7]	A-H
07	SG9	SG10	SG11	SG12	SG13	SG14	SG15	SG16	22222222
09	*SG9*	*SG10*	*SG11*	*SG12*	*SG13*	*SG14*	*SG15*	*SG16*	11111111
0A	SG9	SG10	SG11	SG12	*SG13*	*SG14*	*SG15*	*SG16*	22221111
0C	*SG9*	*SG10*	*SG11*	*SG12*	SG13	SG14	SG15	SG16	11112222
0E	SG9	SG10	SG11	SG12	SG13	SG14	*SG15*	*SG16*	22222211
10	*SG9*	*SG10*	*SG11*	*SG12*	SG13	SG14	*SG15*	*SG16*	11112211
12	SG9	SG10	*SG11*	*SG12*	SG13	SG14	SG15	SG16	22112222
14	*SG9*	*SG10*	SG11	SG12	SG13	SG14	SG15	SG16	11222222
16	SG9	SG10	SG11	SG12	SG13	SG14	SG15	*SG16*	22222221
18	*SG9*	*SG10*	*SG11*	*SG12*	*SG13*	*SG14*	SG15	*SG16*	11111121
1A	SG9	SG10	SG11	SG12	SG13	*SG14*	SG15	SG16	22222122
1C	SG9	SG10	SG11	SG12	*SG13*	SG14	SG15	SG16	22221222
1E	SG9	SG10	SG11	*SG12*	SG13	SG14	SG15	SG16	22212222
20	SG9	SG10	*SG11*	SG12	SG13	SG14	SG15	SG16	22122222
22	SG9	*SG10*	SG11	SG12	SG13	SG14	SG15	SG16	21222222
24	*SG9*	SG10	SG11	SG12	SG13	SG14	SG15	SG16	12222222
3D	*PCIe3*								22222222
3E	*PCIe3*								22222222
3F	*PCIe3*				*PCIe4*				22222222

Table continues on the next page...

Table 2. SRDS_PRCTL choices for SerDes2 in LS2085A Reference Manual (continued)

SRDS_PRCTL_S2	A	B	C	D	E	F	G	H	PLL mapping
hex	SD2[0]	SD2[1]	SD2[2]	SD2[3]	SD2[4]	SD2[5]	SD2[6]	SD2[7]	A-H
40	*PCIe3*				*PCIe4*				22222222
41	PCIe3				PCIe4		SATA1	SATA2	11111122
42	PCIe3				PCIe4		SATA1	SATA2	11111122
43	*PCIe3*				X	X	SATA1	SATA2	11111122
44	*PCIe3*				X	X	SATA1	SATA2	11111122
45	SG9	>SG10	SG11	SG12	*PCIe4*				22222222
46	SG9	SG10	SG11	SG12	*PCIe4*				22222222
47	*PCIe3*	SG10	SG11	SG12	*PCIe4*	SG14	SG15	SG16	22222222
48	*PCIe3*	SG10	SG11	SG12	*PCIe4*	SG14	SG15	SG16	22222222
49	SG9	SG10	SG11	SG12	PCIe4		SATA1	SATA2	11111122
4A	SG9	SG10	SG11	SG12	PCIe4		SATA1	SATA2	11111122

Designing a board with a specific QorIQ SoC requires knowing what protocols and speeds are available, knowing how many lanes you can run these protocols and speeds on, and determining how to configure the board out of reset to support the choices you make. All this becomes a time consuming exercise if all you have at your disposal are tables in the SoC reference manual. The **Protocol/Speed Configuration** window makes it easy.

To open the **Protocol/Speed Configuration** window, use one of the following options:

- Right-click the lanes overview table and choose **Change configuration** from the shortcut menu.
- Click the **Change protocol and speed configuration** button

  ()

  on the toolbar of the **SerDes Configuration and Validation** page in the **Component Inspector** view.

The figure below presents an overview of the **Protocol/Speed Configuration** window, highlighting the main areas and basic operations.

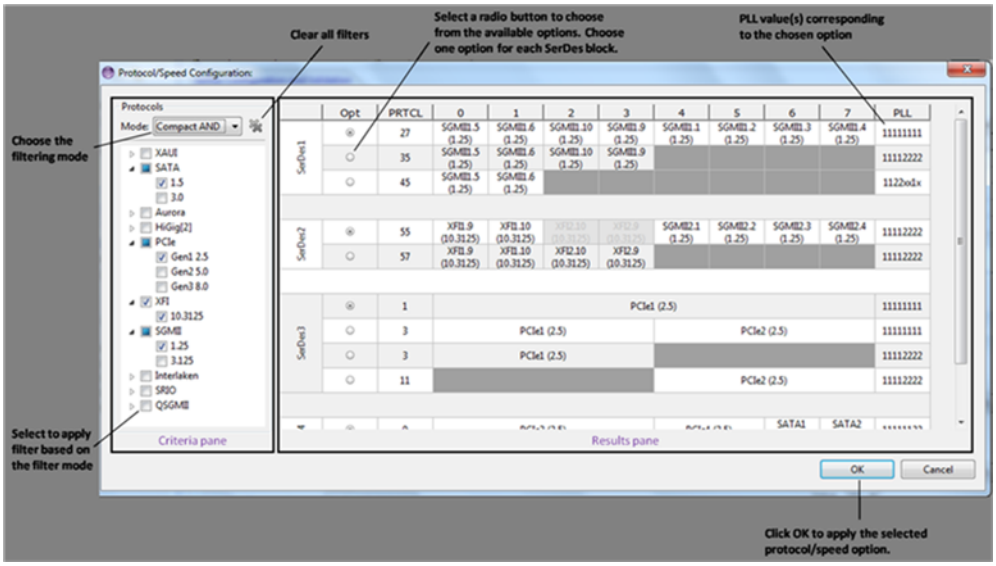
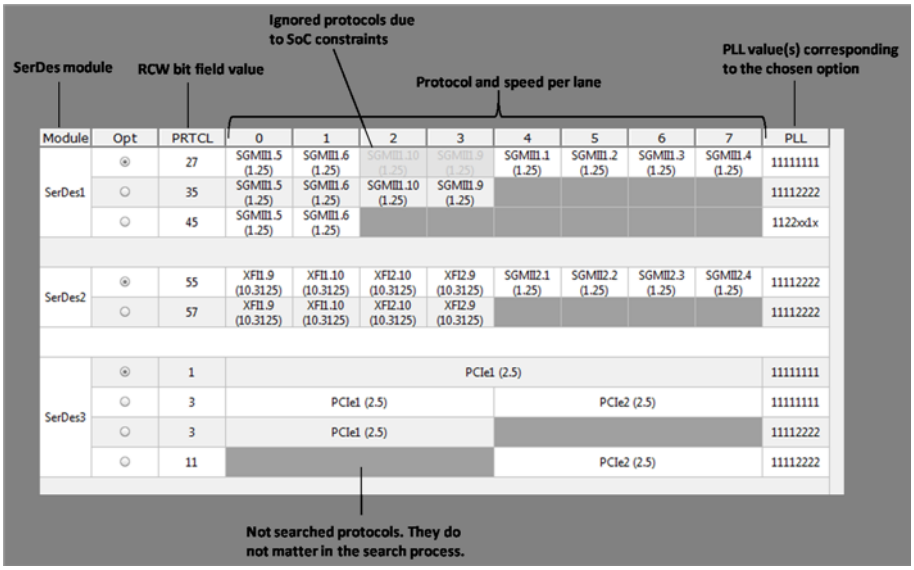


Figure 7. Protocol/Speed Configuration window (overview)

The **Protocol/Speed Configuration** window allows you to choose the protocols and speeds you want in your design, and shows you only the table rows (that is, the SRDS_PRTCL_Sn choices) that are relevant. It spares you the clutter of dozens of irrelevant table options. This window can also be used to quickly determine if a particular combination of protocols and speeds is supported.

The **Protocol/Speed Configuration** window can be divided into the following two panes:

- Criteria pane: The left portion of the window, the Criteria pane, is where you specify the protocols and speeds you are interested in and how you would like those selections used for filtering. This pane has a collapsable tree that contains every protocol/speed supported by all the SerDes modules on the SoC. Each protocol/speed has a checkbox. You can select the protocols/speeds you are interested in. Above the tree is a selector that controls how the selected items will be used to filter the SRDS_PRTCL_Sn options on the right:
  - Compact AND. In this filtering mode, only SRDS_PRTCL_Sn options that collectively and *simultaneously* support all the selected protocols/speeds are shown. If you have selected a combination of protocols/speeds that cannot simultaneously be supported by the SoC, a message indicating this will be displayed in place of SRDS_PRTCL_Sn tables. In addition, relevant SRDS_PRTCL_Sn choices that differ only in lanes containing protocols/speeds that are not selected will be combined, producing a more compact set of results. For example, if SRDS_PRTCL_Sn options 1 and 2 both have the desired PCIe Gen 2 protocol and speed in lanes 0-3, but one option has Aurora in lanes 4-7 and the other option has SATA in those lanes, this filtering mode will show only option 1 and display lanes 4-7 as unnamed dark-grey cells. The idea here is that because you expressed no interest in either Aurora or SATA, both options 1 and 2 equally meet your needs. This filtering mode therefore provides the most concise results.
  - Expanded AND. This filtering mode is similar to Compact AND except that it does not combine *equivalent* choices to reduce the result set. In the example above for Compact AND, both options 1 and 2 will be shown in the results. The non-relevant lanes (4-7) will be displayed in named dark-grey cells; therefore, allowing you to see the protocol/speed for these lanes.
  - Expanded OR. This filtering mode shows all the SRDS_PRTCL_Sn options that contain *any* of the selected protocols/speeds, whether or not the protocols/speeds you selected can all be supported simultaneously. For example, if you select every protocol/speed and choose Expanded OR, the results area will show every SRDS_PRTCL_Sn option the SoC makes available. However, of course, the SoC cannot support every protocol/speed simultaneously; the AND modes would show an empty result set in this situation. The OR mode is more suited for browsing than decision making.
- Results pane: The right portion of the window, the Results pane, shows you the SRDS_PRTCL_Sn options that meet the criteria on the left. This pane has the subset of the SRDS_PRTCL_Sn options that meet the criteria established on the left. These options are displayed in tables, similar to how they are presented in the SoC reference manual (see figure below).



Module	Opt	PRTCL	0	1	2	3	4	5	6	7	PLL
SerDes1	◉	27	SGMII1.5 (1.25)	SGMII1.6 (1.25)	SGMII1.10 (1.25)	SGMII1.9 (1.25)	SGMII1.1 (1.25)	SGMII1.2 (1.25)	SGMII1.3 (1.25)	SGMII1.4 (1.25)	11111111
	○	35	SGMII1.5 (1.25)	SGMII1.6 (1.25)	SGMII1.10 (1.25)	SGMII1.9 (1.25)					11112222
	○	45	SGMII1.5 (1.25)	SGMII1.6 (1.25)							1122xx1x
SerDes2	◉	55	XFI1.9 (10.3125)	XFI1.10 (10.3125)	XFI2.10 (10.3125)	XFI2.9 (10.3125)	SGMII2.1 (1.25)	SGMII2.2 (1.25)	SGMII2.3 (1.25)	SGMII2.4 (1.25)	11112222
	○	57	XFI1.9 (10.3125)	XFI1.10 (10.3125)	XFI2.10 (10.3125)	XFI2.9 (10.3125)					11112222
SerDes3	◉	1	PCIe1 (2.5)								11111111
	○	3	PCIe1 (2.5)				PCIe2 (2.5)				11111111
	○	3	PCIe1 (2.5)								11112222
	○	11					PCIe2 (2.5)				11112222

Figure 8. Protocol/Speed Configuration window (detailed)

If you click **OK**, all the following elements will be changed for the modules that are displayed after filtering is applied:

- Protocol used for each lane and the 20-bit interface field corresponding to that protocol
- Speed for each lane (both Tx and Rx speed will be set to the value used in the GUI)
- PLL number associated with each lane
- PLL frequency and reference clock to what the selected option permits. The tool takes into consideration the valid SerDes reference clock for each protocol and speed, as defined in the reference manual for the SoC being used.
- PLL off option. For example, if one of the PLLs used in the new selected option is set as off, then it will be set as on.

Note that this selection does not change any equalization setting. Therefore, before starting the validation, it is recommended to change the equalization settings to the optimal ones. To determine the optimal values that should be used for each protocol and speed, see the reference manual for the SoC being used.

1.2.5 Configuration Registers view

With the **Configuration Registers** view, you can see how the changes in SerDes configuration reflect in the values of memory-mapped registers, at any moment.

Note that only the registers used in the SerDes tool are displayed with their changed values in this view. To open the **Configuration Registers** view, right-click a SerDes component (module) in the **Components** view and choose **Configuration Registers** from the shortcut menu.

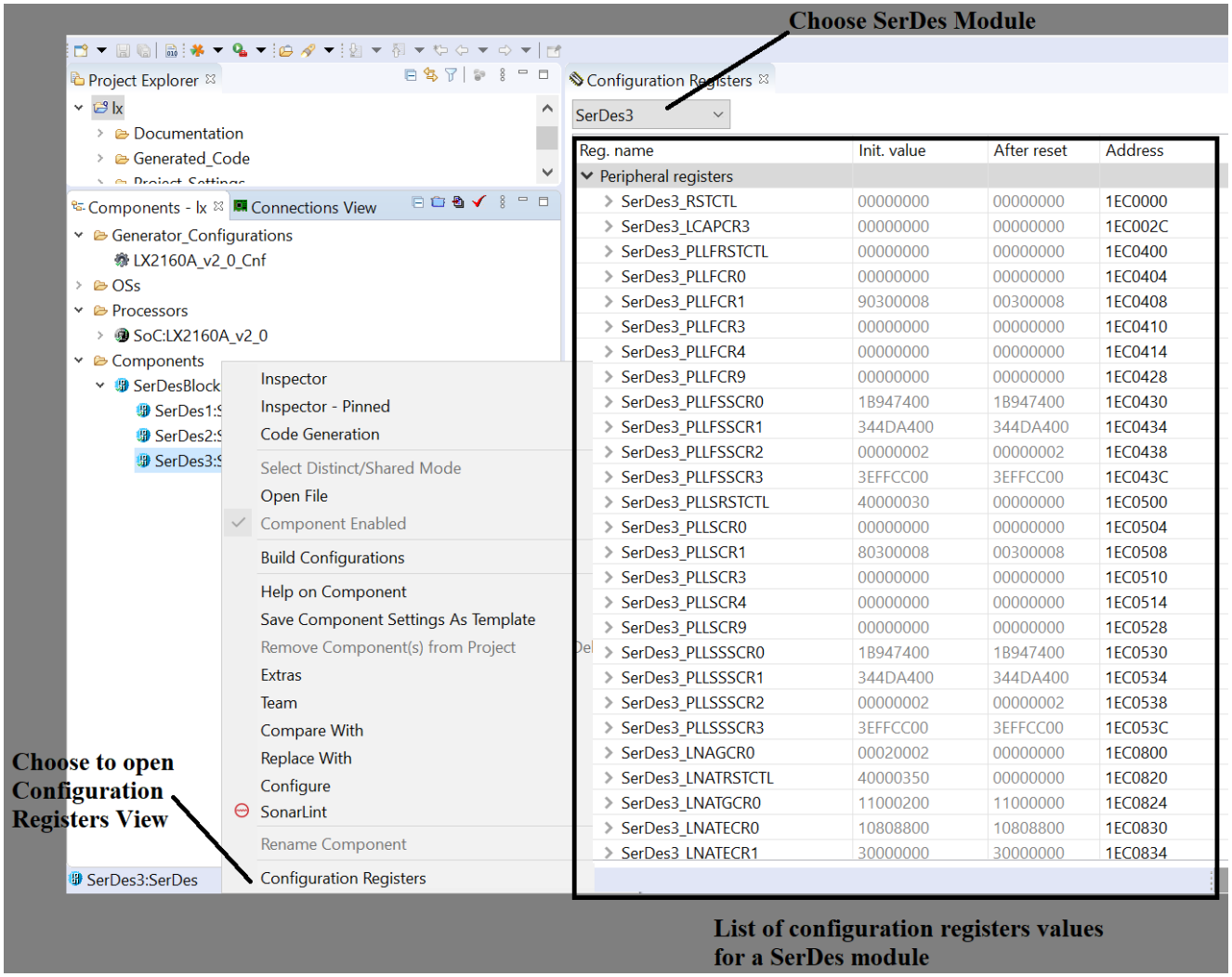


Figure 9. Configuration Registers view

1.2.6 Errata support

The SerDes tool provides support for the errata that impact the SerDes IP block. By default, the SerDes tool supports the latest approved and published errata at the time of the QCVS release.

The errata depend on the conditions that can be read from the target (such as, device type, selected protocol) and/or conditions that are external to the device or SerDes IP block (such as, outside temperature, settings in other IP blocks).

The SerDes tool addresses the errata in the following ways:

- If an erratum becomes applicable or may become applicable, then a warning message is displayed in the **Problems** view that contains the description of the erratum as described in the errata document. The **Problems** view in the figure below shows descriptions of two such errata.

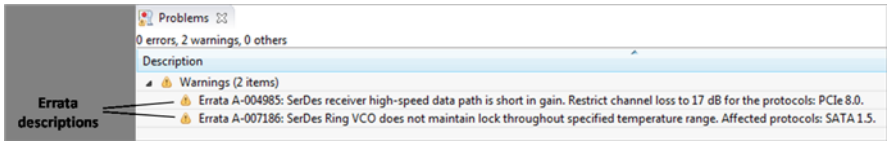


Figure 10. Errata displayed in Problems view

- Any erratum that becomes active due to a change made in the **Protocol/Speed Configuration** window is displayed as a message at the bottom of the window, besides displaying in the **Problems** view. The **Protocol/Speed Configuration** window in the figure below shows descriptions of two such errata.

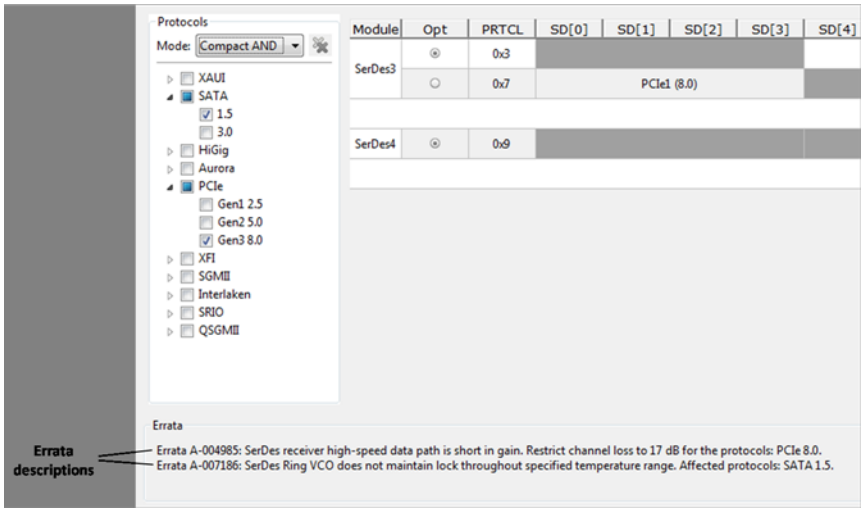


Figure 11. Errata displayed in Protocol/Speed Configuration window

- Any errata that depend on the conditions that cannot be determined by the SerDes tool are always displayed. This makes the user aware that such errata may become active, and the user can take appropriate actions (for example, implement errata workarounds), if needed.

NOTE

See the errata document of the SoC for additional details on the displayed errata warning.

Note that:

- The user is not prevented from running the SerDes validation scenarios even if there are active errata. The user is responsible to implement the necessary errata workaround.
- Automatic errata workaround implementation and switching errata on and off are features considered for future QCVS releases

1.3 SerDes validation

This section describes how to validate the configuration of the SerDes block.

After a SerDes configuration has been created in terms of protocol/speed, lane receive and transmit, and PLL allocation, the question is whether that configuration will perform reliably under the intended type of traffic. For that, the SerDes validation capability comes into the picture.

The validation relies on SerDes hardware block built-in, programmable test capabilities that are used to verify the block under different traffic conditions. The validation programs the built-in testing capabilities, and then runs a series of tests. The results of the tests are displayed in a format that enables you to make a decision on whether or not the SerDes configuration is reliable. If the answer is no, then you need to adjust the SerDes configuration and run the validation again, until the reliable SerDes configuration is determined.

The added value of validation feature is:

- Easy programmability of SerDes built-in test capabilities
- One click run of the validation test, which includes applying SerDes configuration to the target, programming the test block, and collecting and aggregating the results

- A quick synthesis of the test results in industry standard interpretations, such as data eye diagram or recovered data stream diagram

The validation feature requires a working target connection and a valid license. For license details, see Licensing.

Note that when validation is run, the following changes are made:

- All the protocol control registers are cleared
- SerDes is put into the Test mode, and the current configuration is applied (changes are made only to the SerDes memory-mapped registers)
- All idle filters are cleared
- The lane, on which validation is run, is set as the first lane (this is required by the SerDes Test mode)
- Tx output pad control signal for common mode is set to 0 (External and External Loopback UI scenario settings)

This section contains the following subsections:

- SerDes Validation pane
- Data generation modes
- SerDes validation scenarios
- SerDes validation best practices
- Connections View

1.3.1 SerDes Validation pane

The SerDes Validation pane allows you to work with the validation feature of SerDes.

The SerDes validation feature is accessible for each lane, meaning that each lane can be tested individually.

To show the SerDes Validation pane, click the **Validation** tab next to the **Configuration** tab in the SerDes GUI (see Working with SerDes components). The figure below shows the different areas you can work with and different operations you can perform in the SerDes Validation pane.

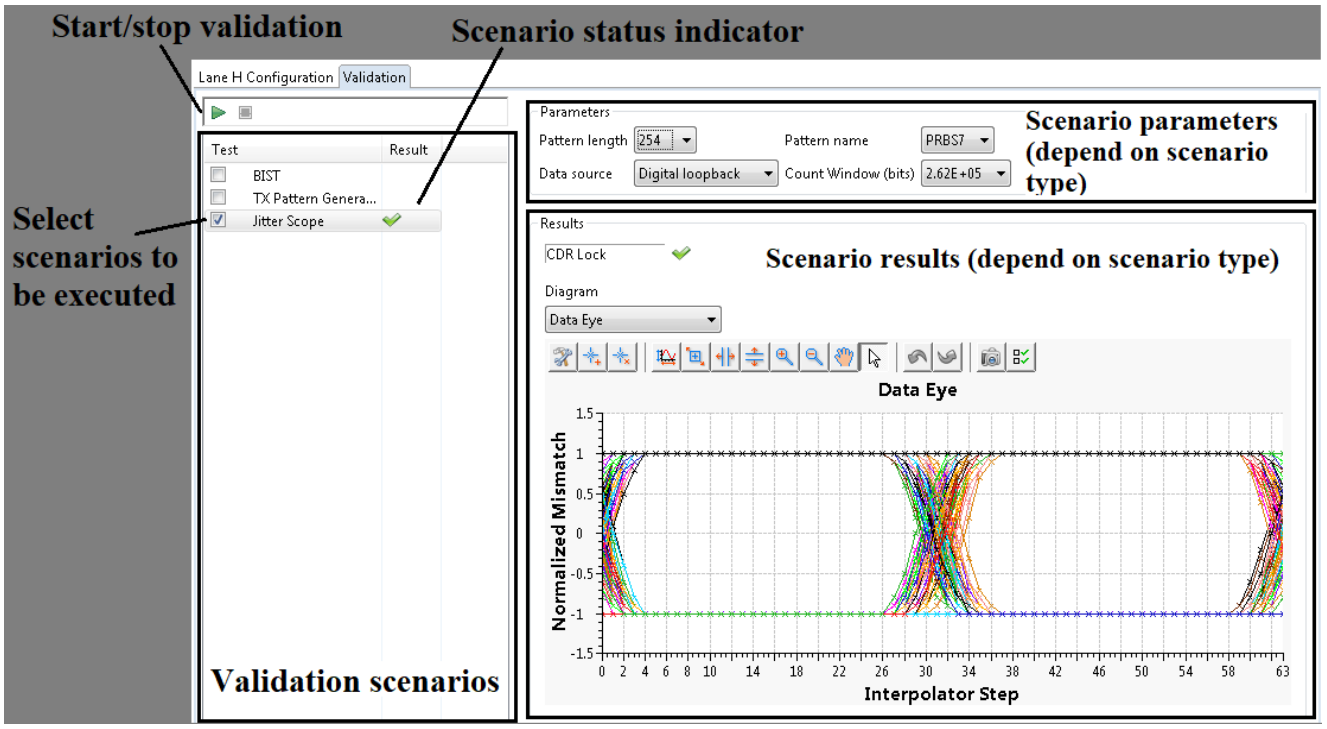


Figure 12. SerDes Validation pane

1.3.2 Data generation modes

This section explains the data generation modes supported by the SerDes tool.

The SerDes tool allows data to be generated using one of the following modes:

- Digital Loopback mode

  In the Digital Loopback mode, transmit data is internally generated on the transmit side using a built-in pattern generator, and is internally looped back to the receive path, as shown in the figure below. This mode is also known as internal loopback mode.

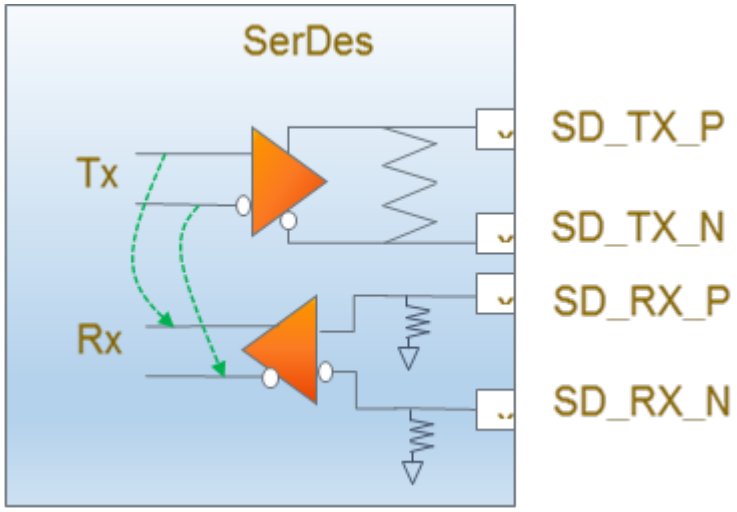


Figure 13. Digital Loopback mode

- External Loopback mode

  In the External Loopback mode, transmit data is internally generated on the transmit side using a built-in pattern generator, and is externally looped back to the receive path, as shown in the figure below. The user needs to provide an external connection between the SD_TX_P/SD_TX_N and SD_RX_P/SD_RX_N pins of the SerDes lane being validated.

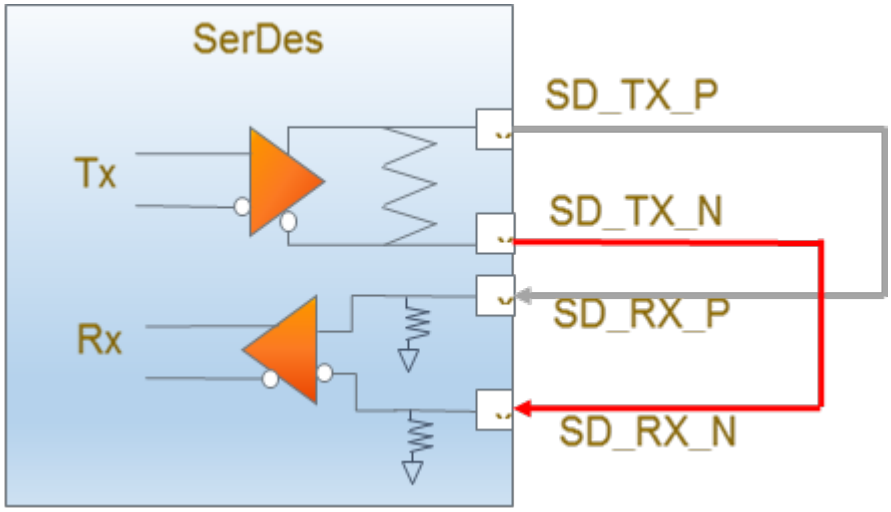


Figure 14. External Loopback mode

- External mode

  In the External mode, an external source, such as a link partner or an external test equipment, generates data that is input to the receive path of the SerDes lane being validated. BIST and Jitter scope scenarios can be run in the External mode. When running BIST in the External mode, the selected test pattern must match the test pattern generated by the external source. When running the Jitter scope scenario in the External mode, the selected pattern length must match the externally sourced pattern.

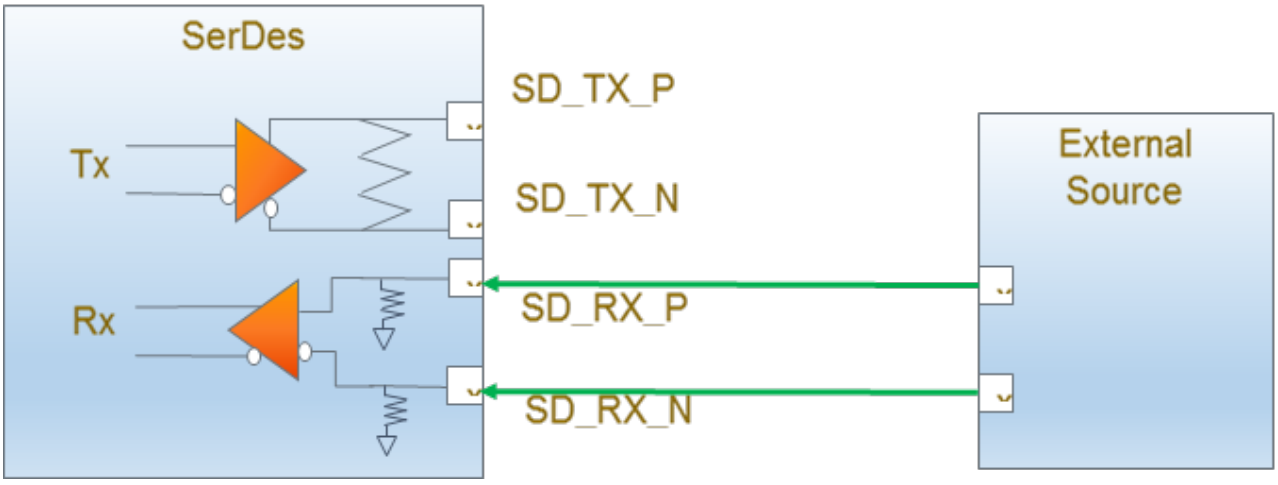


Figure 15. External mode

1.3.3 SerDes validation scenarios

The SerDes validation scenarios are mapped with the SerDes built-in test module, which can be programmed to perform different sets of tests.

Running a SerDes validation scenario involves the following steps:

1. Program the SerDes built-in test capability for a specific test. The scenario can be customized using the SerDes Validation pane (see SerDes Validation pane).
2. Set the mode in which the traffic is generated (Digital Loopback, External Loopback, or External (that is, traffic is generated from an SoC outside the source))
3. Set the SerDes self-test module into the Run mode, and process the traffic according to the user settings
4. Collect the results as the traffic is processed
5. After scenario completion, compile the results into a summary that can easily be interpreted by the user. Depending on the complexity of the scenario and collected results, the summary can be in text format (BIST scenario) or as a diagram (Jitter scope scenario).

The SerDes tool provides you the following three validation scenarios to validate SerDes configuration:

- BIST scenario
- Tx pattern generation scenario
- Jitter scope scenario
- Pattern-independent jitter scope scenario

1.3.3.1 BIST scenario

Built-in self-test (BIST) refers to a quick self-evaluation of a hardware block.

Running a BIST scenario involves the following steps:

1. Choose a pattern to be generated and how the traffic will flow through the lane (that is, data source mode: Digital Loopback, External Loopback, or External).
2. Choose a count window that represents how much traffic will be generated and analyzed by the SerDes lane. It is set in bits and corresponds to a time value. The count window can widely vary from seconds to days, having direct impact on when the BIST results will be available.
3. The traffic formed out of the chosen pattern flows through the SerDes lane and results are collected. Optionally, as BIST runs with the largest count window, the user can insert some check errors. The BIST output should contain the same number of generated errors as the number of errors inserted. That is a quick, on-the-fly method to detect if some bits were lost.
4. After BIST scenario completion, a text summary is displayed, as shown in the figure below.

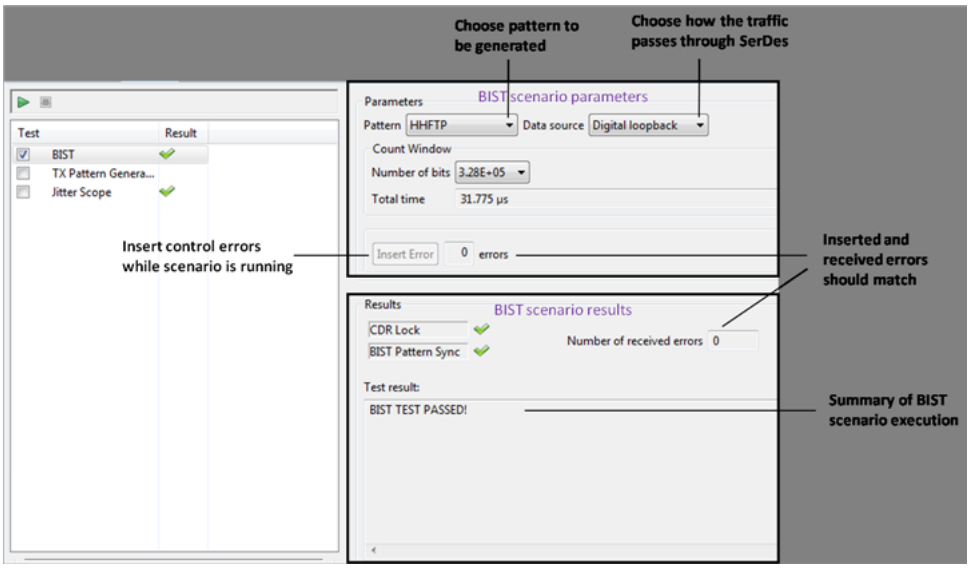


Figure 16. BIST validation scenario

1.3.3.2 Tx pattern generation scenario

Running the Tx pattern generation scenario drives the SerDes module to generate a specific pattern (set up by the user) on the Tx side of the lane that is being verified.

The Tx pattern generation scenario is not an actual validation scenario because no results are collected when it runs.

The figure below shows how the Tx pattern generation scenario works.

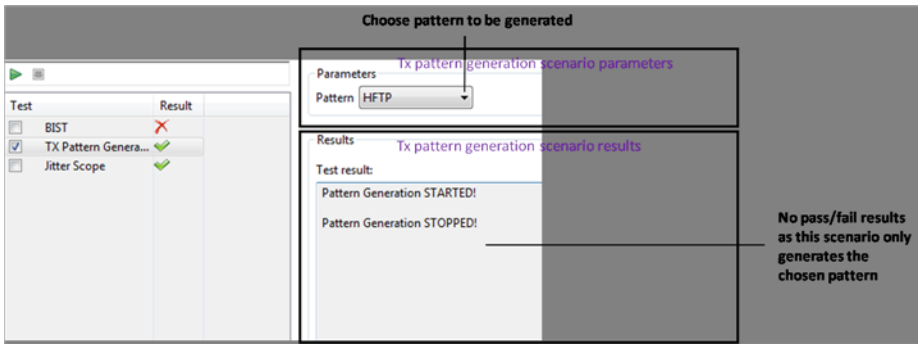


Figure 17. Tx pattern generation scenario

The Tx pattern generation scenario can be used in an environment with two devices connected such that:

- One device starts transmitting data with the Tx pattern generation scenario
- The other device verifies the received data with BIST or Jitter scope scenario in the External mode
- The Tx pattern generation scenario continues to run until the stop button is pressed

The figure below depicts the Tx pattern generation scenario use case described above.

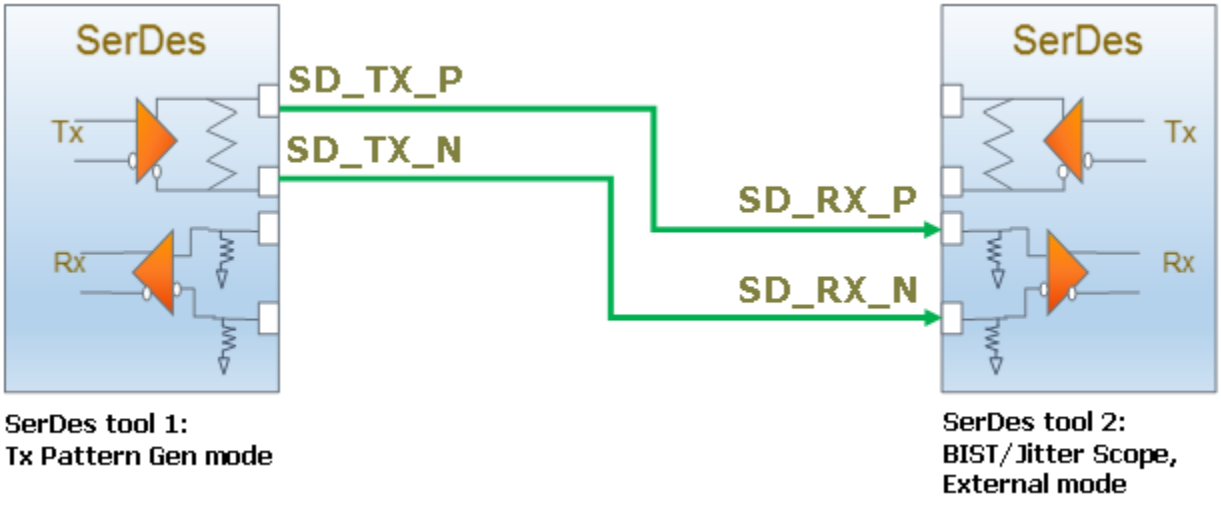


Figure 18. Tx pattern generation scenario use case

1.3.3.3 Jitter scope scenario

The Jitter scope scenario displays the results of the SerDes configuration validation as a diagram (data eye diagram or recovered data stream diagram).

With Jitter scope scenario, a pattern is repeatedly input into the receive path and a sampling point is moved across the incoming data stream, which gets compared to either 1 or 0. The bit is either 1 or 0 until it gets to the transition area. Within the transition area, the number of mismatches will be proportional to the bit error rate. The data eye diagram provides a measurement of eye opening in the receive path. The recovered data stream diagram shows the received pattern.

In the Digital Loopback and External Loopback modes, the pattern is generated by the SerDes block. In case of the External mode, you can send any pattern from an external source (such as a test equipment), as long as it matches the selected length.

The figure below shows how the Jitter scope scenario works.

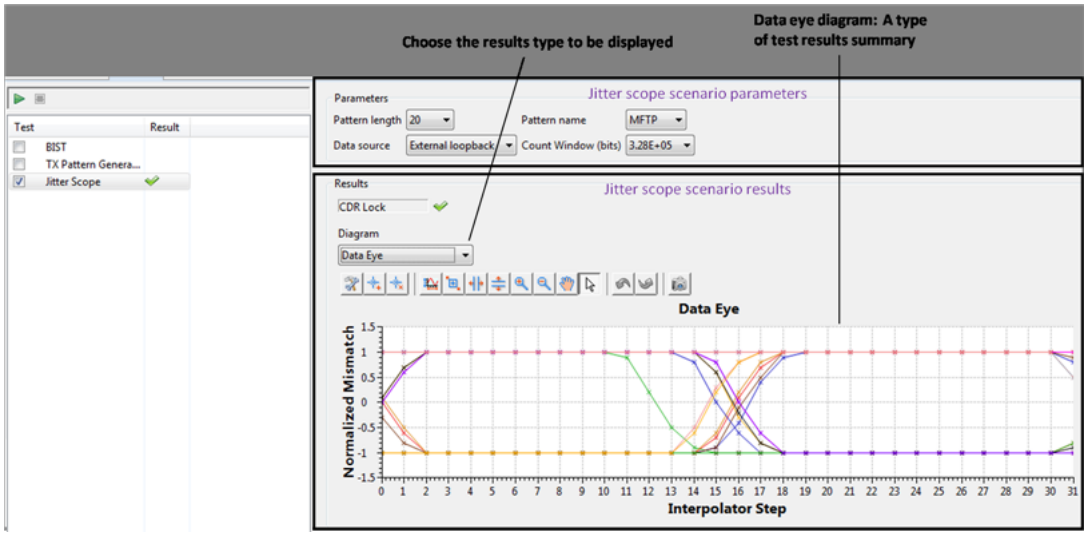


Figure 19. Jitter scope scenario

The figure below describes the data eye and recovered data stream diagrams.

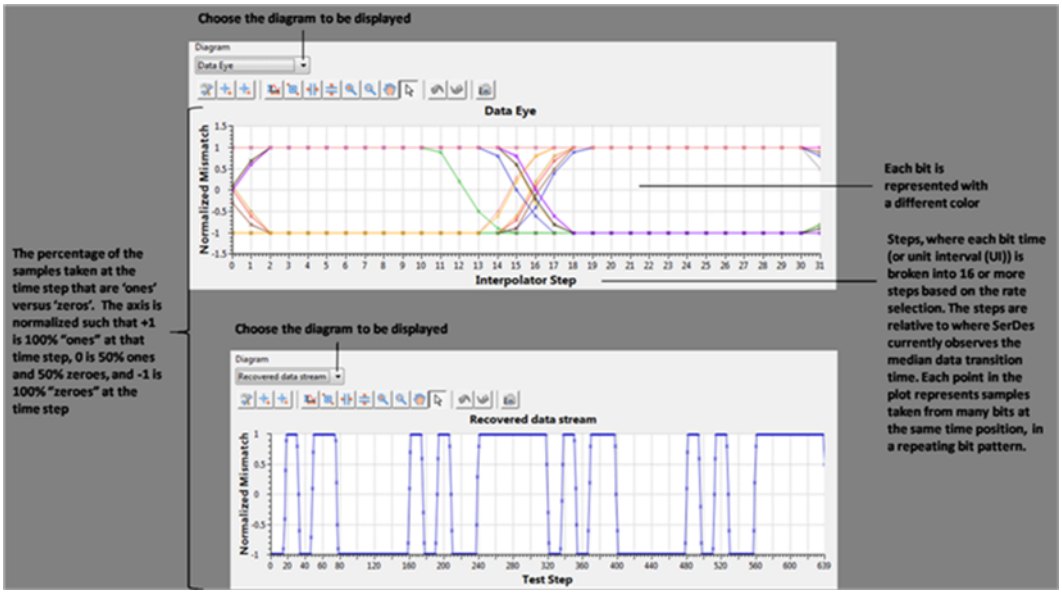


Figure 20. Data eye and recovered data stream diagrams

The raw data based on which the Jitter scope diagrams are created can be retrieved from `/<workspace_folder>/<project_name>/.SerDes_QorIQ_<id>_<device_name>/SerDes<module_index>_LN<lane_index>_JS_results.txt`.

The text file contains input for the data eye and recovered data stream diagrams as comma-separated values.

Each line from the raw data contains three values:

- Step number: Represents how many times the bits in a count window are compared against an expected value. The number of steps ensures that each bit is verified at least once.
- Number of mismatches: Indicates the sum of all the bits from a count window that do not match with a compare bit
- Normalized number of mismatches, that is, the number of mismatches scaled to the interval [-1, 1], where 1 indicates 100% 1s and -1 indicates 100% 0s

You should expect a delay of certain amount of time before Jitter scope results (data eye and recovered data stream diagrams) are displayed. The delay depends on the selected pattern length and speed. The table below shows the approximate expected delays.

Table 3. Approximate expected delays

Pattern Length	1.25G	2.5G	3.125G 5G	6.25G 10.3125G
10-bits	6 min	3 min	1.5 min	N/A
20-bits	14 min	7 min	3 min	1.5 min
40-bits	26 min	13 min	7 min	4 min
127-bits	84 min	42 min	21 min	N/A
240-bits	168 min	84 min	42 min	21 min
254-bits	N/A	N/A	N/A	21 min
511-bits	336 min	168 min	84 min	N/A
1022-bits	N/A	N/A	N/A	84 min

1.3.3.4 Pattern-independent jitter scope scenario

This feature is only available for the LX devices, which have 28G Lynx support. 28G Lynx provides the capability of constructing the recovered data eye diagram and *bathtub* curve of bit error density (bit error rate versus sampler phase offset curve), independent of the presence of a BIST pattern. This scenario shares the same Jitter scope scenario concept that data samples are compared to count the miscompares for a given early phase step, but it compares all data samples regardless of their patterns and considers the nearest data bit in the comparison. In the Digital Loopback and External Loopback modes, the pattern is generated by the SerDes block. In case of the External mode, you can send any pattern from an external source (such as a test equipment), as long as it matches the selected length.

The figure below shows how the Pattern-independent jitter scope scenario works.

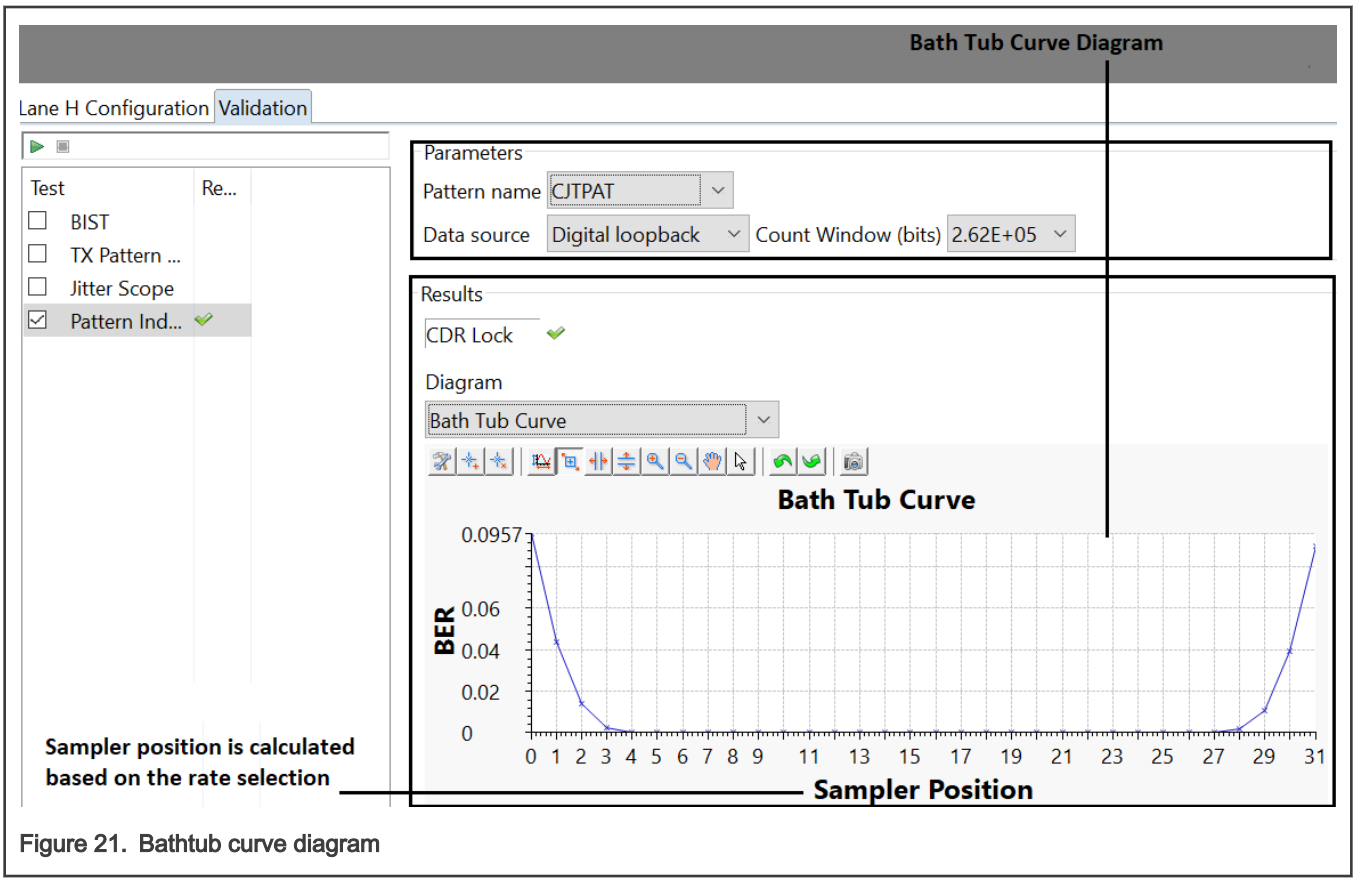


Figure 21. Bathtub curve diagram

The raw data based on which the Pattern-independent jitter scope scenario diagrams are created can be retrieved from the /<workspace_folder>/<project_name>/.SerDes_QorIQ_<id>_<device_name>/SerDes<module_index>_LN<lane_index>_PIJS_results.txt file. This file contains input for the bathtub curve BER diagram as comma-separated values.

You should expect some delay before Pattern-independent jitter scope scenario results are displayed.

1.3.4 SerDes validation best practices

The purpose of a SerDes validation is to gradually refine a SerDes configuration, by checking it against the provided set of scenarios.

Following are the best practice steps for running a SerDes validation:

1. Create a SerDes configuration by reading the configuration set on the target.
   - For each lane, choose the desired set of protocols using the **Protocol/Speed Configuration** window.
   - Use the protocol filter capability rather than seeking through the multitude of options.

- Change equalization settings for the desired lane. See the reference manual for the SoC being used, to know the optimal values that should be used for each protocol and speed.

2. Check if the lane you want to verify is powered up; otherwise, power it up.
3. Ensure that the input reference clock from the target (for the module and PLL you are using) matches the one that SerDes expects to receive (the one specified in the PLL configuration view).
4. Run the BIST scenario in the Digital Loopback mode with a small (few seconds) count window.
5. Run the BIST scenario again in the Digital Loopback mode with a larger count window. Use the largest count window to insert errors and verify if they are received correctly.
6. Run the BIST scenario in the External Loopback mode with a sufficient count (for example, in minutes) window. Set up the External Loopback mode according to the protocol you chose for that lane and the design of the target you are using. For example, if you chose PCIe as the protocol, ensure that a PCIe loopback card is installed on the target, and it is connected to the corresponding module and lane you are trying to validate.
7. Continue with running the Jitter scope scenario. In the Jitter scope scenario, start with the default count window and check if the data eye diagram forms a regular eye, rather than a "deformed" eye. See in the figure below the difference between a good and a bad SerDes configuration indicator diagrams (data eye diagrams or recovered data stream diagrams).
8. Make a setup with two devices connected such that:
   - One device starts transmitting data with the Tx pattern generation scenario, and
   - The other device verifies the received data with BIST or Jitter scope scenario in the External mode

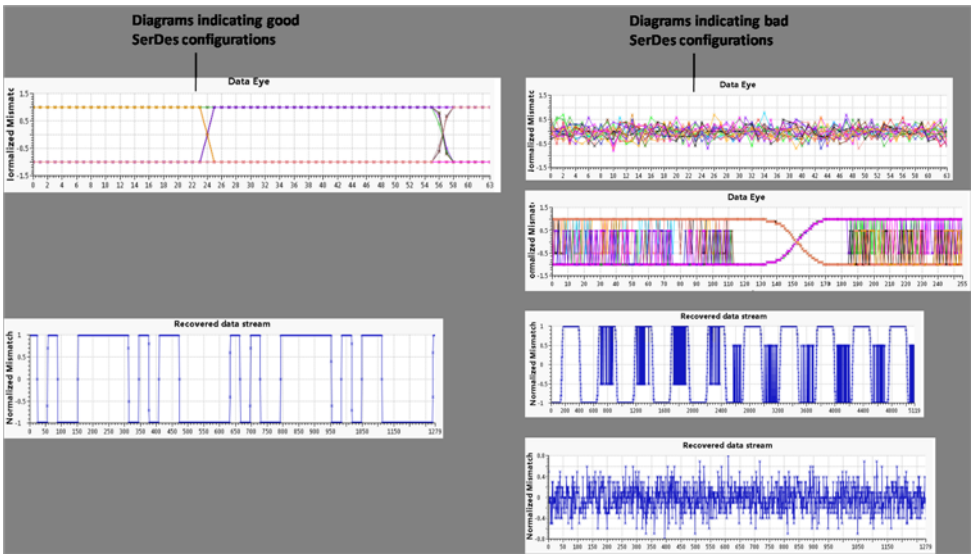


Figure 22. Good versus bad SerDes configuration indicator diagrams

1.3.5 Connections View

The **Connections View** allows you to configure settings for your target connection.

You can open the **Connections View** by choosing **Show View** > **Connections View** from the **Window** menu bar.

The SerDes validation requires a working connection to a target. Using the **Connections View**, you can add, delete, or edit a target connection. The current connection is highlighted in the **Connections View**, as shown in the figure below.

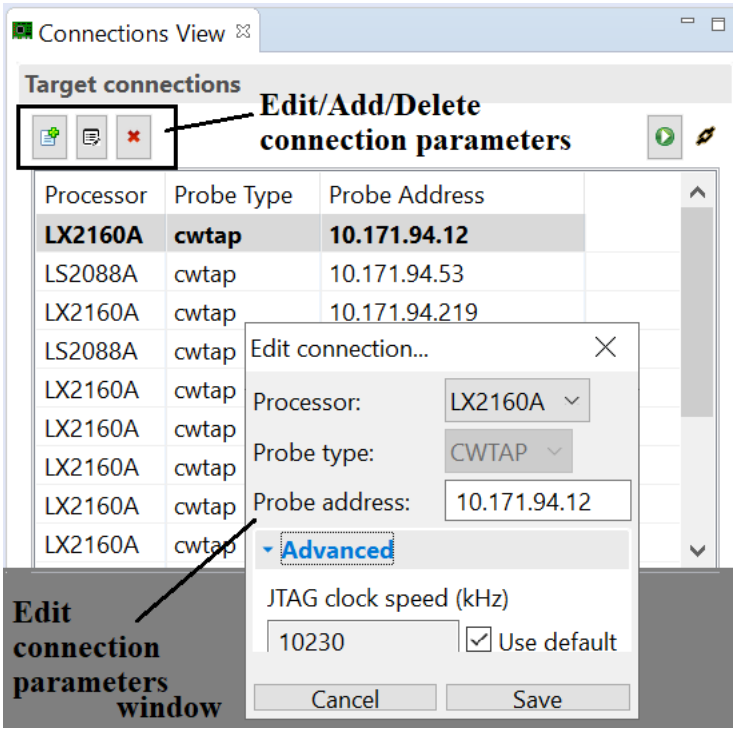


Figure 23. Connections View

If the target connection is lost during validation, then the chain icon showing in the **Target connections** group will change to a broken chain icon with an error icon. In that case, you would need to re-establish the connection by clicking the **Play** button located next to the chain icon.

1.4 How to use a SerDes configuration

This section describes how to use a SerDes configuration to program the target.

Once an optimized SerDes configuration is determined, it can be used to program the target. The SerDes configuration consists of:

- Values of SerDes memory-mapped registers. For details, see Code generation.
- RCW settings for the lanes protocols/speeds. For details, see Synchronization between SerDes and PBL.

Therefore, using a SerDes configuration means flashing a new RCW and then writing into memory-mapped registers.

This section contains the following subsections:

- RCW settings
- Code generation
- Synchronization between SerDes and PBL

1.4.1 RCW settings

The RCW settings determine how a platform is configured, before any software runs on it.

The protocols/speeds allocated to each SerDes lane, the reference clock SerDes expects to receive for each PLL, and the Power-Down state for each PLL are usually set in RCW. See Synchronization between SerDes and PBL for details on how to push all these changes into the corresponding RCW bits from PBL.

1.4.2 Code generation

The SerDes memory-mapped register values can be obtained by clicking the **Generate Processor Expert Code** button on the toolbar of the **Components** view.

The register values are generated in various code formats. Currently, the SerDes tool only supports U-Boot commands and C code as code formats. The figure below shows how to generate code.

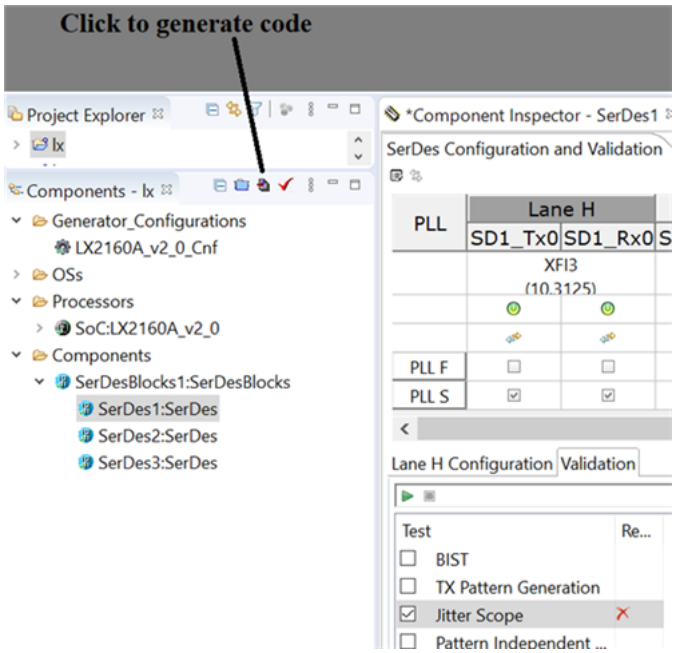


Figure 24. Generating code

1.4.3 Synchronization between SerDes and PBL

A pre-boot loader (PBL) is a binary image that is used to configure a hardware platform before any other software (for example, U-Boot) can run on it.

The RCW used to initially set up the platform contains several bits related to SerDes configuration. A SerDes configuration can be reflected into updated RCW bits with just one button click.

Before performing synchronization, you need to create a PBL component. After creating the PBL component, perform synchronization between SerDes and PBL by clicking the **Apply the configuration to PBL component** button ( ) on the toolbar of the **SerDes Configuration and Validation** page in the **Component Inspector** view, as shown in the figure below. When no PBL component exists in the project, the button is grayed out with tooltip text as "You must have a PBL component in your project in order to apply the serdes configuration also into RCW."

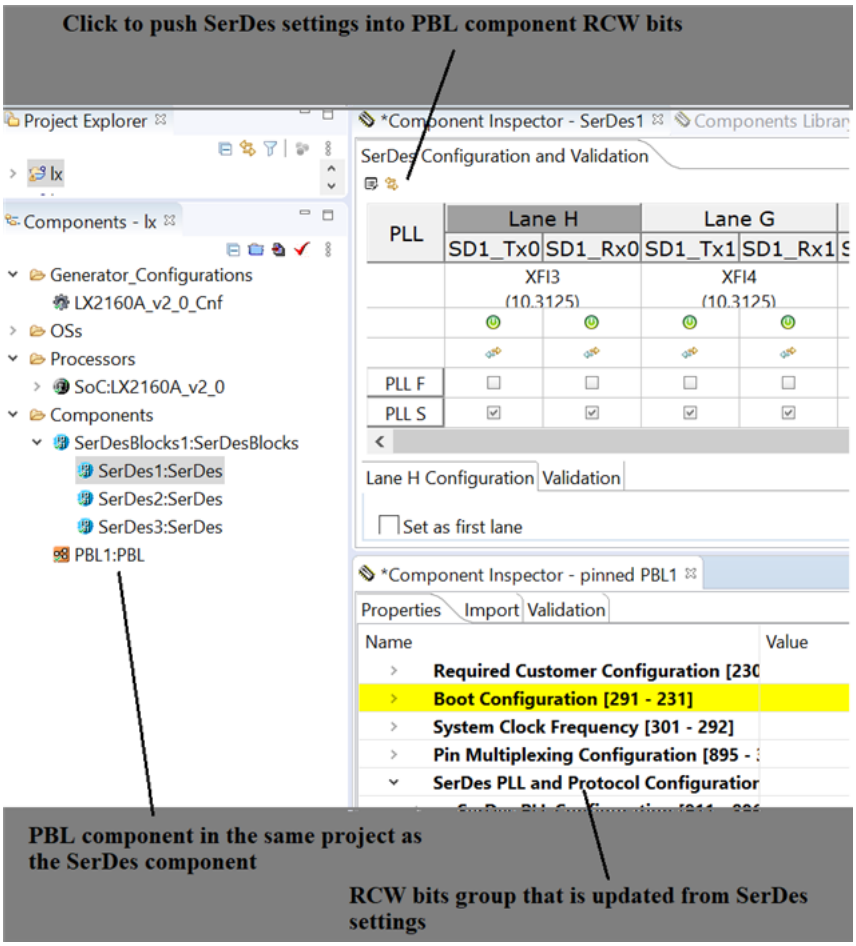


Figure 25. Synchronizing PBL with SerDes

1.5 Licensing

This section provides licensing details related to the SerDes tool.

The validation component of the SerDes tool is licensed. A valid license file should be located in the `<INSTALL_DIR>\eclipse\ProcessorExpert\` folder in QCVS product layout. If you do not have a valid license, you will not be able to perform certain validation operations. At the same time, you will be informed about how to obtain a new license. The figure below shows what to expect if no valid license was found.

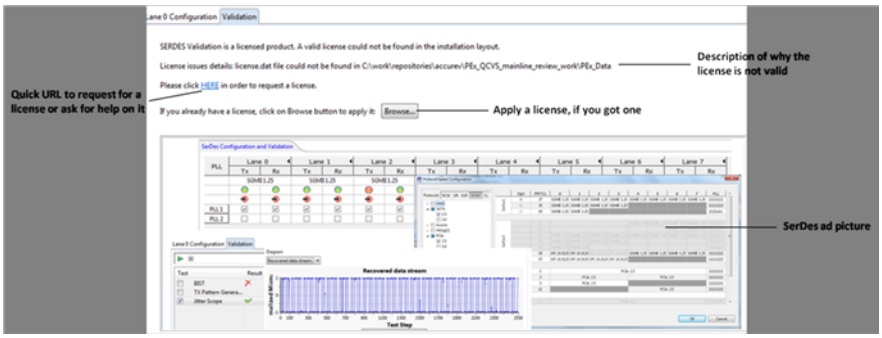


Figure 26. Invalid license error

If you click the **Here** link displayed on the **Validation** page, then you will be redirected to the www.nxp.com/cw4net web page. On this web page, you can:

- Click the **Download Eval** button to download the evaluation version of the SerDes validation tool. A temporary license will be generated when you install the tool.
- Click the **Buy** button to purchase the licensed version of the SerDes validation tool

Index

How To Reach Us

Home Page:

nxp.com

Web Support:

nxp.com/support

Limited warranty and liability — Information in this document is provided solely to enable system and software implementers to use NXP products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits based on the information in this document. NXP reserves the right to make changes without further notice to any products herein.

NXP makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does NXP assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. “Typical” parameters that may be provided in NXP data sheets and/or specifications can and do vary in different applications, and actual performance may vary over time. All operating parameters, including “typicals,” must be validated for each customer application by customer's technical experts. NXP does not convey any license under its patent rights nor the rights of others. NXP sells products pursuant to standard terms and conditions of sale, which can be found at the following address: nxp.com/SalesTermsandConditions.

Right to make changes — NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Security — Customer understands that all NXP products may be subject to unidentified or documented vulnerabilities. Customer is responsible for the design and operation of its applications and products throughout their lifecycles to reduce the effect of these vulnerabilities on customer’s applications and products. Customer’s responsibility also extends to other open and/or proprietary technologies supported by NXP products for use in customer’s applications. NXP accepts no liability for any vulnerability. Customer should regularly check security updates from NXP and follow up appropriately. Customer shall select products with security features that best meet rules, regulations, and standards of the intended application and make the ultimate design decisions regarding its products and is solely responsible for compliance with all legal, regulatory, and security related requirements concerning its products, regardless of any information or support that may be provided by NXP. NXP has a Product Security Incident Response Team (PSIRT) (reachable at PSIRT@nxp.com) that manages the investigation, reporting, and solution release to security vulnerabilities of NXP products.

For more information, please visit: http://www.nxp.com
For sales office addresses, please send an email to: salesaddresses@nxp.com

Date of release: 01/2021
Document identifier: QCVS_SerDes_User_Guide