JESD204B IP Core User Guide

Last updated for Altera Complete Design Suite: 14.1

UG-01142
2014.12.15

Subscribe

Send Feedback

101 Innovation Drive
San Jose, CA 95134
www.altera.com

Contents

JESD204B IP Core Quick Reference..... 1-1

About the JESD204B IP Core.....2-1
Datapath Modes..... 2-3
IP Core Variation..... 2-3
JESD204B IP Core Configuration..... 2-4
Run-Time Configuration..... 2-4
Channel Bonding..... 2-5
Performance and Resource Utilization..... 2-6

Getting Started..... 3-1
Introduction to Altera IP Cores..... 3-1
Installing and Licensing IP Cores..... 3-2
Upgrading IP Cores..... 3-2
IP Catalog and Parameter Editor..... 3-4
Design Walkthrough..... 3-5
Creating a New Quartus II Project..... 3-5
Parameterizing and Generating the IP Core..... 3-6
Generating and Simulating the IP Core Testbench..... 3-7
Compiling the JESD204B IP Core Design..... 3-9
Programming an FPGA Device..... 3-10
JESD204B IP Core Design Considerations..... 3-10
Integrating the JESD204B IP core in Qsys..... 3-10
Pin Assignments..... 3-11
Adding External Transceiver PLL..... 3-12
JESD204B IP Core Parameters..... 3-12
JESD204B IP Core Component Files..... 3-15
JESD204B IP Core Testbench..... 3-16
Testbench Simulation Flow..... 3-17

JESD204B IP Core Functional Description..... 4-1
Transmitter..... 4-4
TX Data Link Layer..... 4-5
TX PHY Layer..... 4-8
Receiver..... 4-8
RX Data Link Layer..... 4-9
RX PHY Layer..... 4-12
Operation..... 4-13
Operating Modes..... 4-13
Scrambler/Descrambler..... 4-14

SYNC_N Signal....4-14
Link Reinitialization....4-16
Link Startup Sequence....4-17
Error Reporting Through SYNC_N Signal....4-18
Clocking Scheme....4-18
Device Clock....4-20
Link Clock....4-21
Local Multi-Frame Clock....4-22
Clock Correlation....4-23
Reset Scheme....4-24
Signals....4-26
Transmitter....4-26
Receiver....4-35
Registers....4-42
Register Access Type Convention....4-42

JESD204B IP Core Design Guidelines....5-1
JESD204B IP Core Design Example....5-1
Design Example Components....5-3
System Parameters....5-37
System Interface Signals....5-41
Example Feature: Dynamic Reconfiguration....5-46
Generating and Simulating the Design Example....5-52
Generating the Design Example For Compilation....5-53
Compiling the JESD204B IP Core Design Example....5-54

JESD204B IP Core Debug Guidelines....6-1
Clocking Scheme....6-1
JESD204B Parameters....6-1
SPI Programming....6-2
Converter and FPGA Operating Conditions....6-2
Signal Polarity and FPGA Pin Assignment....6-2
Debugging JESD204B Link Using SignalTap II and System Console....6-3

Additional Information....7-1
JESD204B IP Core Document Revision History....7-1
How to Contact Altera....7-2

JESD204B IP Core Quick Reference

1

2014.12.15

UG-01142

Subscribe Send Feedback

The Altera® JESD204B MegaCore® function is a high-speed point-to-point serial interface intellectual property (IP).

The JESD204B MegaCore function is part of the MegaCore IP Library, which is distributed with the Quartus® II software and downloadable from the Altera website at **www.altera.com**.

Note: For system requirements and installation instructions, refer to *Altera Software Installation & Licensing*.

Table 1-1: Brief Information About the JESD204B MegaCore Function

Item		Description
Release Information	Version	14.1
	Release Date	December 2014
	Ordering Code	IP-JESD204B
	Product ID	0116
	Vendor ID	6AF7

IP Core Information	Protocol Features	• Joint Electron Device Engineering Council (JEDEC) JESD204B.01, 2012 standard release specification
		• Device subclass:
		  • Subclass 0—Backwards compatible to JESD204A.
		  • Subclass 1—Uses *SYSREF* signal to support deterministic latency.
		  • Subclass 2—Uses *SYNC_N* detection to support deterministic latency.

Item		Description
IP Core Information	Core Features	• Run-time configuration of parameters L,M, and F • Data rates up to 12.5 gigabits per second (Gbps) • Single or multiple lanes (up to 8 lanes per link) • Serial lane alignment and monitoring • Lane synchronization • Modular design that supports multidevice synchronization • MAC and PHY partitioning • Deterministic latency support • 8B/10B encoding • Scrambling/Descrambling • Avalon® Streaming (Avalon-ST) interface for transmit and receive datapaths • Avalon Memory-Mapped (Avalon-MM) interface for Configuration and Status registers (CSR) • Dynamic generation of simulation testbench
	Typical Application	• Wireless communication equipment • Broadcast equipment • Military equipment • Medical equipment • Test and measurement equipment
	Device Family Support	• Arria® V FPGA device families • Arria V GZ FPGA device families • Arria 10 FPGA device families • Stratix® V FPGA device families Refer to the device support table and *What's New in Altera IP* page of the Altera website for detailed information.
	Design Tools	• Qsys parameter editor in the Quartus II software for design creation and compilation • TimeQuest timing analyzer in the Quartus II software for timing analysis • ModelSim®-Altera, Aldec Riviera-Pro, VCS/VCS MX, and NCSim software for design simulation or synthesis

Related Information

- **Altera Software Installation and Licensing**
- **What's New in Altera IP**

About the JESD204B IP Core 2

2014.12.15

UG-01142

Subscribe Send Feedback

The Altera JESD204B IP core is a high-speed point-to-point serial interface for digital-to-analog (DAC) or analog-to-digital (ADC) converters to transfer data to FPGA devices. This unidirectional serial interface runs at a maximum data rate of 12.5 Gbps. This protocol offers higher bandwidth, low I/O count and supports scalability in both number of lanes and data rates. The JESD204B IP core addresses multi-device synchronization by introducing Subclass 1 and Subclass 2 to achieve deterministic latency.

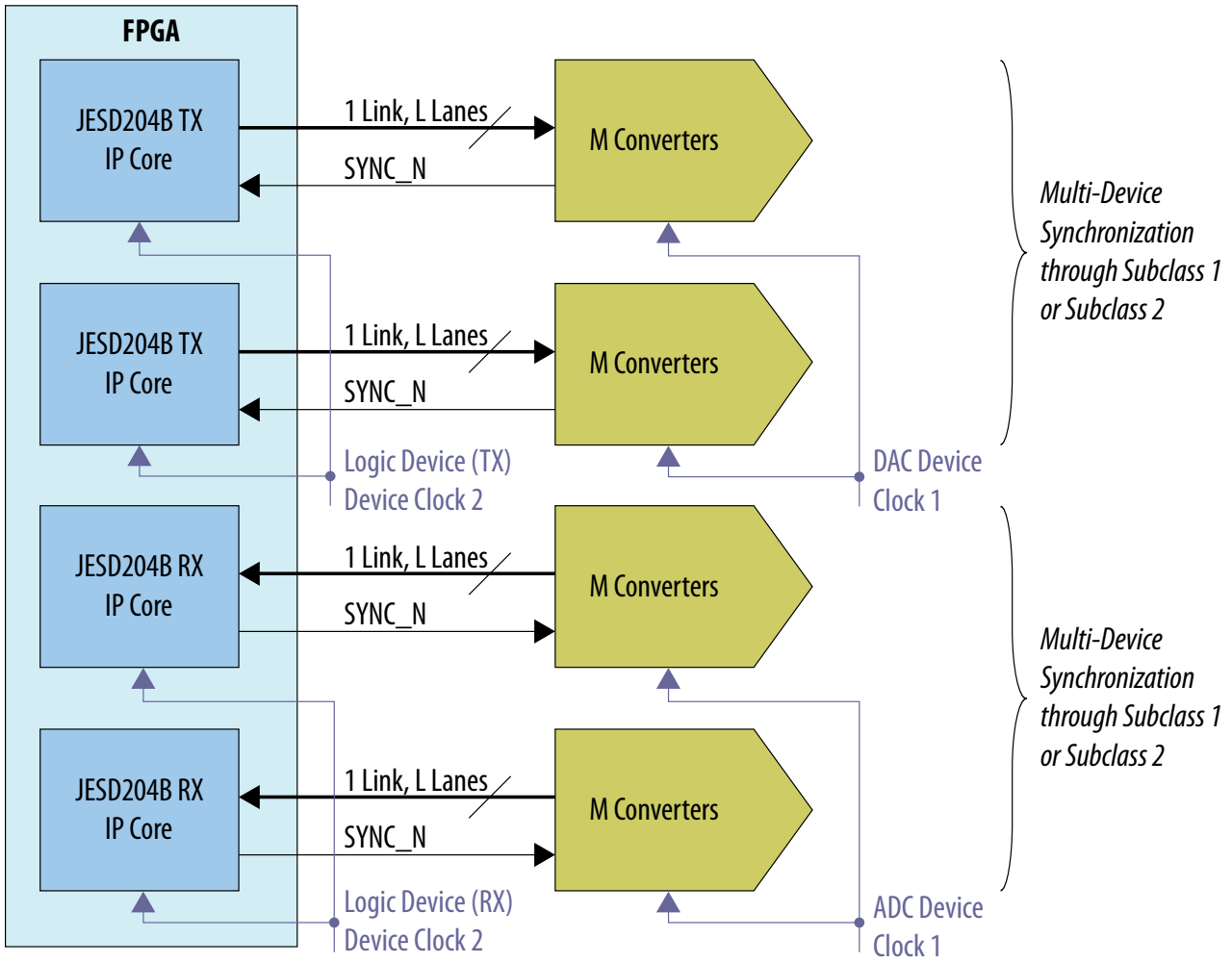
The JESD204B IP core incorporates:

- Media access control (MAC)—data link layer (DLL) block that controls the link states and character replacement.
- Physical layer (PHY)—physical coding sublayer (PCS) and physical media attachment (PMA) block.

The JESD204B IP core does not incorporate the Transport Layer (TL) that controls the frame assembly and disassembly. The TL and test components are provided as part of a design example component where you can customize the design for different converter devices.

Figure 2-1: Typical System Application for JESD204B IP Core

The JESD204B IP core utilizes the Avalon-ST source and sink interfaces, with unidirectional flow of data, to transmit and receive data on the FPGA fabric interface.



Key features of the JESD204B IP core:

- Data rate of up to 12.5 Gbps
- Run-time JESD204B parameter configuration (L, M, F, S, N, K, CS, CF)
- MAC and PHY partitioning for portability
- Subclass 0 mode for backward compatibility to JESD204A
- Subclass 1 mode for deterministic latency support (using *SYSREF*) between the ADC/DAC and logic device
- Subclass 2 mode for deterministic latency support (using *SYNC_N*) between the ADC/DAC and logic device
- Multi-device synchronization

Datapath Modes

The JESD204B IP core supports TX-only, RX-only, and Duplex (TX and RX) mode. The IP core is a unidirectional protocol where interfacing to ADC utilizes the transceiver RX path and interfacing to DAC utilizes the transceiver TX path.

The JESD204B IP core generates a single link with a single lane and up to a maximum of 8 lanes. If there are two ADC links that need to be synchronized, you have to generate two JESD204B IP cores and then manage the deterministic latency and synchronization signals, like *SYSREF* and *SYNC_N*, at your custom wrapper level.

The JESD204B IP core supports duplex mode only if the LMF configuration for ADC (RX) is the same as DAC (TX) and with the same data rate. This use case is mainly for prototyping with internal serial loopback mode. This is because typically as a unidirectional protocol, the LMF configuration of converter devices for both DAC and ADC are not identical.

IP Core Variation

The JESD204B IP core has three core variations:

- JESD204B MAC only
- JESD204B PHY only
- JESD204B MAC and PHY

In a subsystem where there are multiple ADC and DAC converters, you need to use the Quartus II software to merge the transceivers and group them into the transceiver architecture. For example, to create two instances of the JESD204B TX IP core with four lanes each and four instances of the JESD204 RX IP core with two lanes each, you can apply one of the following options:

- MAC and PHY option
  1. Generate JESD204B TX IP core with four lanes and JESD204B RX IP core with two lanes.
  2. Instantiate the desired components.
  3. Use the Quartus II software to merge the PHY lanes.
- MAC only and PHY only option—based on the configuration above, there are a total of eight lanes in duplex mode.
  1. Generate the JESD204B Duplex PHY with a total of eight lanes. (TX skew is reduced in this configuration as the channels are bonded).
  2. Generate the JESD204B TX MAC with four lanes and instantiate it two times.
  3. Generate the JESD204B RX MAC with two lanes and instantiate it four times.
  4. Create a wrapper to connect the JESD204B TX MAC and RX MAC with the JESD204B Duplex PHY.

Note: If the data rate for TX and RX is different, the transceiver does not allow duplex mode to generate a duplex PHY. In this case, you have to generate a RX-only PHY on the RX data rate and a TX-only PHY on the TX data rate.

JESD204B IP Core Configuration

Table 2-1: JESD204B IP Core Configuration

Symbol	Description	Value
L	Number of lanes per converter device	1-8
M	Number of converters per device	1-32
F	Number of octets per frame	1, 2, 4-256
S	Number of transmitted samples per converter per frame	1-32
N	Number of conversion bits per converter	1-32
N'	Number of transmitted bits per sample (JESD204 word size, which is in nibble group)	4-32
K	Number of frames per multiframe	$17/F \leq K \leq 32$; 1-32
CS	Number of control bits per conversion sample	0-3
CF	Number of control words per frame clock period per link	0-32
HD	High Density user data format	0 or 1
LMFC	Local multiframe clock	(F × K /4) link clock counts (1)

Run-Time Configuration

The JESD204B IP core allows run-time configuration of LMF parameters.

The most critical parameters that must be set correctly during IP generation are the L and F parameters. Parameter L denotes the maximum lanes supported while parameter F denotes the size of the deskew buffer needed for deterministic latency. The hardware generates during parameterization, which means that run-time programmability can only fall back from the parameterized and generated hardware, but not beyond the parameterized IP core.

You can use run-time configuration for prototyping or evaluating the performance of converter devices with various LMF configurations. However, in actual production,Altera recommends that you generate the JESD204B IP core with the intended LMF to get an optimized gate count.

For example, if a converter device supports LMF = 442 and LMF = 222, to check the performance for both configurations, you need to generate the JESD204B IP core with maximum F and L, which is L = 4 and F = 2. During operation, you can use the fall back configuration to disable the lanes that are not used in LMF = 222 mode. You must ensure that other JESD204B configurations like M, N, S, CS, CF, and HD do not violate the parameter F setting. You can access the Configuration and Status Register (CSR) space to modify other configurations such as:

- K (multi-frame)
- device and lane IDs
- enable or disable scrambler
- enable or disable character replacement

(1) The value of F x K must be divisible by 4.

F Parameter

This parameter indicates how many octets per frame per lane that the JESD204B link is operating in. You must set the F parameter according to the JESD204B IP Specification for a correct data mapping.

To support the High Density (HD) data format, the JESD204B IP core tracks the start of frame and end of frame because F can be either an odd or even number. The start of frame and start of multi-frame wrap around the 32-bits data width architecture. The RX IP core outputs the start of frame (`sof[3:0]`) and start of multiframe (`somf[3:0]`), which act as markers, using the Avalon-ST data stream. Based on these markers, the transport layer build the frames.

In a simpler system where the HD data format is set to 0, the F will always be 1, 2, 4, 6, 8, and so forth. This simplifies the transport layer design, so you do not need to use the `sof[3:0]` and `somf[3:0]` markers.

Channel Bonding

The JESD204B IP core supports channel bonding—bonded and non-bonded modes.

The channel bonding mode that you select may contribute to the transmitter channel-to-channel skew. A bonded transmitter datapath clocking provides low channel-to-channel skew as compared to non-bonded channel configurations.

Table 2-2: Maximum Number of Lanes (L) Supported in Bonded and Non-Bonded Mode

- In PHY-only mode, you can generate up to 32 channels, provided that the channels are on the same side. In MAC and PHY integrated mode, you can generate up to 8 channels.
- In bonded channel configuration, the lower transceiver clock skew and equal latency in the transmitter phase compensation FIFO for all channels result in a lower channel-to-channel skew. You must use adjacent channels when you select ×6 bonding. You must also place logical channel 0 in either physical channel 1 or 4. Physical channels 1 and 4 are indirect drivers of the ×6 clock network. The JESD204B IP core automatically selects between xN or feedback compensation (fb_compensation) bonding depending on the number of transceiver channels you set.
- When you select bonded channel and L<6, the IP core automatically selects xN/x6 bonding mode for the transceiver. When you select bonded channel and L≥6, the IP core automatically selects x6 PLL fb_compensation bonding mode for the transceiver
- In non-bonded channel configuration, the transceiver clock skew is higher and latency is unequal in the transmitter phase compensation FIFO for each channel. This may result in a higher channel-to-channel skew.

Device Family	Core Variation	Bonding Mode Configuration	Maximum Number of Lanes (L)
Arria V	PHY only	Bonded	32 (2)
		Non-bonded	32 (2)
	MAC and PHY	Bonded	6
		Non-bonded	8

(2) The maximum lanes listed here is for configuration simplicity. Refer to the *Altera Transceiver PHY User Guide* for the actual number of channels supported.

Device Family	Core Variation	Bonding Mode Configuration	Maximum Number of Lanes (L)
Arria V GZ Arria 10 Stratix V	PHY only	Bonded	32 (2)
		Non-bonded	32 (2)
	MAC and PHY	Bonded	8
		Non-bonded	8

Performance and Resource Utilization

Table 2-3: JESD204B IP Core FPGA Performance

Device Family	PMA Speed Grade	FPGA Fabric Speed Grade	Data Rate		Link Clock F_{MAX} (MHz)
			Enable Hard PCS (Gbps)	Enable Soft PCS (Gbps) (3)	
Arria V	*<Any supported speed grade>*	*<Any supported speed grade>*	1.0 to 6.55	— (4)	163.84
Arria V GZ	2	3	2.0 to 9.9	— (4)	247.50
	3	4	2.0 to 8.8	— (4)	220.00
Arria 10	2 or 3	1 or 2	2.0 to 12.0 (5)	2.0 to 12.5	312.50
	4	3	—	2.0 to 9.8	312.50
Stratix V	1	1 or 2	2.0 to 12.2	2.0 to 12.5	312.50
	2	1 or 2	2.0 to 12.2	2.0 to 12.5	
		3	2.0 to 9.8	2.0 to 12.5	
	3	2 or 3 or 4	2.0 to 8.5	2.0 to 8.5	

The following table lists the resources and expected performance of the JESD204B IP core. These results are obtained using the Quartus II software targeting the following Altera FPGA devices:

- Arria V : 5AGXFB3H4F35C5
- Arria V GZ : 5AGZME5K2F40C3
- Arria 10 : 10AX115H2F34I2SGES
- Stratix V : 5SGXEA7H3F35C3

(3) Select Enable Soft PCS to achieve maximum data rate. For the TX IP core, enabling soft PCS incurs an additional 3–8% increase in resource utilization. For the RX IP core, enabling soft PCS incurs an additional 10–20% increase in resource utilization.

(4) Enabling Soft PCS does not increase the data rate for the device family and speed grade. You are recommended to select the *Enable Hard PCS* option.

(5) Only -1 FPGA speed grade supports the data rate of up to 12.0 Gbps using Hard PCS.

All the variations for resource utilization are configured with the following parameter settings:

Table 2-4: Parameter Settings To Obtain the Resource Utilization Data

Parameter	Setting
JESD204B Wrapper	Base and PHY
JESD204B Subclass	1
Data Rate	5 Gbps
PCS Option	Enabled Hard PCS
PLL Type	• ATX (for 10 series devices) • CMU (for V series devices)
Bonding Mode	Non-bonded
Reference Clock Frequency	125.0 MHz
Octets per frame (F)	1
Enable Scrambler (SCR)	Off
Enable Error Code Correction (ECC_EN)	Off

Table 2-5: JESD204B IP Core Resource Utilization

The numbers of ALMs and logic registers in this table are rounded up to the nearest 10.

Note: The resource utilization data are extracted from a full design which includes the Altera Transceiver PHY Reset Controller IP Core. Thus, the actual resource utilization for the JESD204B IP core should be smaller by about 15 ALMs and 20 registers.

Device Family	Data Path	Number of Lanes	ALMs	ALUTs	Logic Registers	Memory Block (M10K/M20K) [6] [7]
Arria V	RX	1	1040	1530	1180	1
		2	1570	2310	1810	2
		4	2820	4030	3070	4
		8	5310	7520	5570	8
	TX	1	730	1140	930	0
		2	880	1410	1050	0
		4	1190	1930	1280	0
		8	1700	2830	1740	0

[6] M10K for Arria V device, M20K for Arria V GZ, Stratix V and Arria 10 devices.
[7] The Quartus II software may auto-fit to use MLAB when the memory size is too small. Conversion from MLAB to M20K or M10K was performed for the numbers listed above.

Device Family	Data Path	Number of Lanes	ALMs	ALUTs	Logic Registers	Memory Block (M10K/M20K) (6) (7)
Arria V GZ	RX	1	1050	1530	1200	1
		2	1610	2330	1840	2
		4	2930	4080	3130	4
		8	5550	7660	5700	8
	TX	1	710	1140	930	0
		2	940	1480	1070	0
		4	1340	2100	1340	0
		8	2130	3400	1880	0
Arria 10	RX	1	1030	1490	1180	1
		2	1560	2250	1800	2
		4	2820	3910	3050	4
		8	5310	7310	5530	8
	TX	1	690	1090	920	0
		2	880	1360	1030	0
		4	1230	1880	1270	0
		8	1890	2960	1740	0
Stratix V	RX	1	1060	1530	1200	1
		2	1610	2330	1840	2
		4	2920	4080	3130	4
		8	5540	7660	5700	8
	TX	1	720	1140	930	0
		2	930	1480	1070	0
		4	1340	2100	1340	0
		8	2130	3400	1880	0

Related Information

- **JESD204B IP Core Parameters** on page 3-12
- **Fitter Resources Reports in the Quartus II Help**
  Information about the Quartus II resource utilization reporting, including ALMs needed.

(6) M10K for Arria V device, M20K for Arria V GZ, Stratix V and Arria 10 devices.
(7) The Quartus II software may auto-fit to use MLAB when the memory size is too small. Conversion from MLAB to M20K or M10K was performed for the numbers listed above.

Getting Started 3

2014.12.15

UG-01142

Subscribe Send Feedback

The JESD204B IP core is part of the MegaCore IP Library distributed with the Quartus II software and downloadable from the Altera website at **www.altera.com**.

Related Information

Altera Software Installation & Licensing

Introduction to Altera IP Cores

Altera and strategic IP partners offer a broad portfolio of off-the-shelf, configurable IP cores optimized for Altera devices. The Altera Complete Design Suite (ACDS) installation includes the Altera IP library. The OpenCore and OpenCore Plus IP evaluation features enable fast acquisition, evaluation, and hardware testing of Altera IP cores.

You can integrate optimized and verified IP cores into your design to shorten design cycles and maximize performance. The Quartus II software also supports IP cores from other sources. Use the IP Catalog to efficiently parameterize and generate a custom IP variation for instantiation in your design.

The Altera IP library includes the following IP core types:

- Basic functions
- DSP functions
- Interface protocols
- Memory interfaces and controllers
- Processors and peripherals

Note: The IP Catalog (**Tools** > **IP Catalog**) and parameter editor replace the MegaWizard™ Plug-In Manager for IP selection and parameterization, beginning in Quartus II software version 14.0. Use the IP Catalog and parameter editor to locate and paramaterize Altera and other supported IP cores.

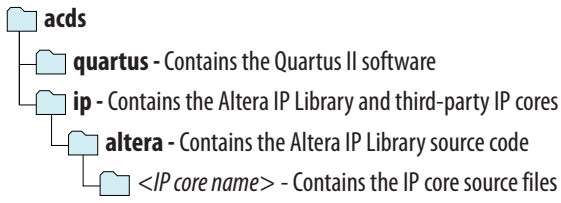
Related Information

- **IP User Guide Documentation**
- **Altera IP Release Notes**

Installing and Licensing IP Cores

The Altera IP Library provides many useful IP core functions for production use without purchasing an additional license. You can evaluate any Altera IP core in simulation and compilation in the Quartus II software using the OpenCore® evaluation feature. Some Altera IP cores, such as MegaCore functions, require that you purchase a separate license for production use. You can use the OpenCore Plus feature to evaluate IP that requires purchase of an additional license until you are satisfied with the functionality and performance. After you purchase a license, visit the Self Service Licensing Center to obtain a license number for any Altera product.

Figure 3-1: IP Core Installation Path



Note: The default IP installation directory on Windows is **<*drive*>:\altera**<*version number*>; on Linux it is <*home directory*>**/altera/** <*version number*>.

Related Information

- **Altera Licensing Site**
- **Altera Software Installation and Licensing Manual**

Upgrading IP Cores

IP core variants generated with a previous version of the Quartus II software may require upgrading before use in the current version of the Quartus II software. Click **Project** > **Upgrade IP Components** to identify and upgrade IP core variants.

The **Upgrade IP Components** dialog box provides instructions when IP upgrade is required, optional, or unsupported for specific IP cores in your design. You must upgrade IP cores that require it before you can compile the IP variation in the current version of the Quartus II software. Many Altera IP cores support automatic upgrade.

The upgrade process renames and preserves the existing variation file (**.v**, **.sv**, or **.vhd**) as <*my_variant*>_**BAK.v**, **.sv**, **.vhd** in the project directory.

Table 3-1: IP Core Upgrade Status

IP Core Status	Corrective Action
Required Upgrade IP Components	You must upgrade the IP variation before compiling in the current version of the Quartus II software.

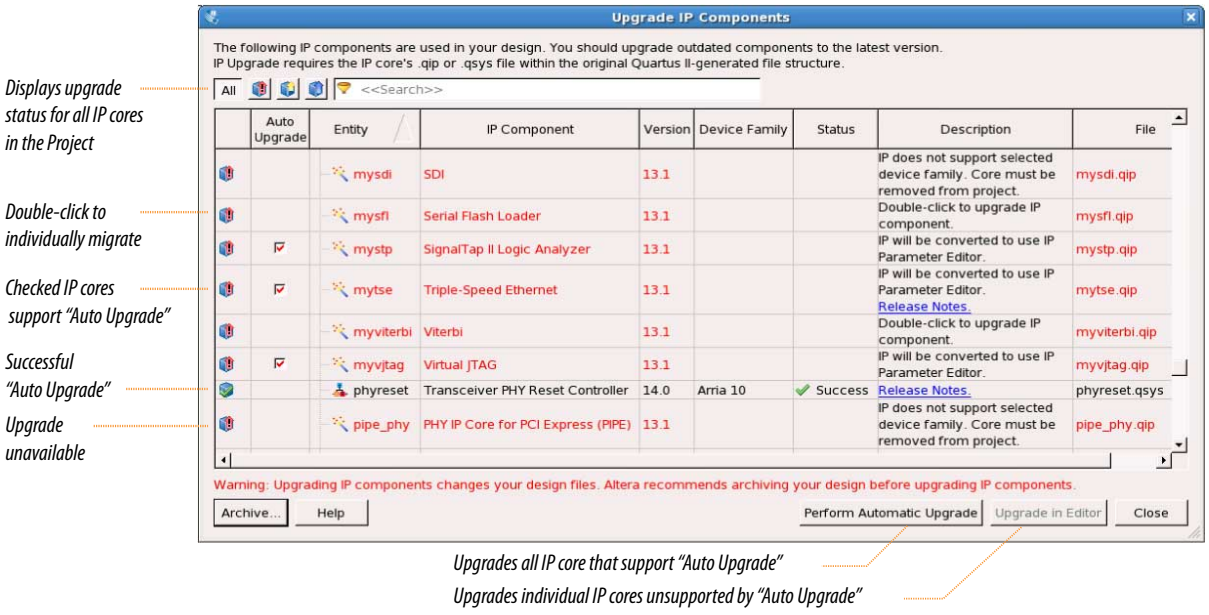
IP Core Status	Corrective Action
Optional Upgrade IP Components	Upgrade is optional for this IP variation in the current version of the Quartus II software. You can upgrade this IP variation to take advantage of the latest development of this IP core. Alternatively you can retain previous IP core characteristics by declining to upgrade.
Upgrade Unsupported	Upgrade of the IP variation is not supported in the current version of the Quartus II software due to IP core end of life or incompatibility with the current version of the Quartus II software. You are prompted to replace the obsolete IP core with a current equivalent IP core from the IP Catalog.

Before you begin

- Archive the Quartus II project containing outdated IP cores in the original version of the Quartus II software: Click **Project** > **Archive Project** to save the project in your previous version of the Quartus II software. This archive preserves your original design source and project files.
- Restore the archived project in the latest version of the Quartus II software: Click **Project** > **Restore Archived Project**. Click **OK** if prompted to change to a supported device or overwrite the project database. File paths in the archive must be relative to the project directory. File paths in the archive must reference the IP variation **.v** or **.vhd** file or **.qsys** file (not the **.qip** file).

1. In the latest version of the Quartus II software, open the Quartus II project containing an outdated IP core variation. The **Upgrade IP Components** dialog automatically displays the status of IP cores in your project, along with instructions for upgrading each core. Click **Project** > **Upgrade IP Components** to access this dialog box manually.
2. To simultaneously upgrade all IP cores that support automatic upgrade, click **Perform Automatic Upgrade**. The **Status** and **Version** columns update when upgrade is complete. Example designs provided with any Altera IP core regenerate automatically whenever you upgrade the IP core.

Figure 3-2: Upgrading IP Cores



Example 3-1: Upgrading IP Cores at the Command Line

You can upgrade IP cores that support auto upgrade at the command line. IP cores that do not support automatic upgrade do not support command line upgrade.

- To upgrade a single IP core that supports auto-upgrade, type the following command:

```
quartus_sh –ip_upgrade –variation_files <my_ip_filepath/my_ip>.<hdl>
<qii_project>

Example:
quartus_sh -ip_upgrade -variation_files mega/pll25.v hps_testx
```

- To simultaneously upgrade multiple IP cores that support auto-upgrade, type the following command:

```
quartus_sh –ip_upgrade –variation_files “<my_ip_filepath/my_ip1>.<hdl>;
<my_ip_filepath/my_ip2>.<hdl>”  <qii_project>

Example:
quartus_sh -ip_upgrade -variation_files "mega/pll_tx2.v;mega/pll3.v"
hps_testx
```

Note: IP cores older than Quartus II software version 12.0 do not support upgrade. Altera verifies that the current version of the Quartus II software compiles the previous version of each IP core. The *Altera IP Release Notes* reports any verification exceptions for Altera IP cores. Altera does not verify compilation for IP cores older than the previous two releases.

Related Information

Altera IP Release Notes

IP Catalog and Parameter Editor

The Quartus II IP Catalog (**Tools** > **IP Catalog**) and parameter editor help you easily customize and integrate IP cores into your project. You can use the IP Catalog and parameter editor to select, customize, and generate files representing your custom IP variation.

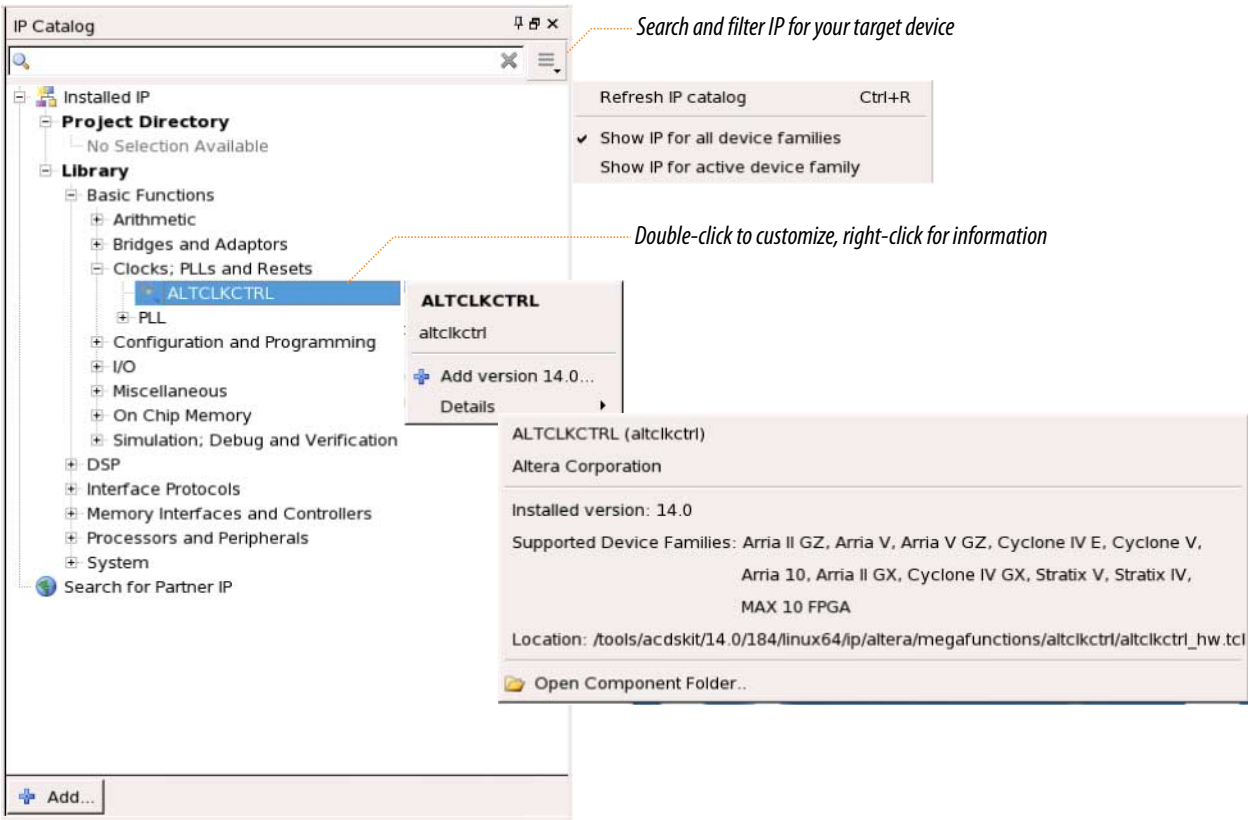
Note: The IP Catalog (**Tools** > **IP Catalog**) and parameter editor replace the MegaWizard™ Plug-In Manager for IP selection and parameterization, beginning in Quartus II software version 14.0. Use the IP Catalog and parameter editor to locate and paramaterize Altera IP cores.

The IP Catalog lists IP cores available for your design. Double-click any IP core to launch the parameter editor and generate files representing your IP variation. The parameter editor prompts you to specify an IP variation name, optional ports, and output file generation options. The parameter editor generates a top-level Qsys system file (**.qsys**) or Quartus II IP file (**.qip**) representing the IP core in your project. You can also parameterize an IP variation without an open project.

Use the following features to help you quickly locate and select an IP core:

- Filter IP Catalog to **Show IP for active device family** or **Show IP for all device families**.
- Search to locate any full or partial IP core name in IP Catalog. Click **Search for Partner IP**, to access partner IP information on the Altera website.
- Right-click an IP core name in IP Catalog to display details about supported devices, open the IP core's installation folder, andor view links to documentation.

Figure 3-3: Quartus II IP Catalog



Note: The IP Catalog is also available in Qsys (**View** > **IP Catalog**). The Qsys IP Catalog includes exclusive system interconnect, video and image processing, and other system-level IP that are not available in the Quartus II IP Catalog. For more information about using the Qsys IP Catalog, refer to *Creating a System with Qsys* in the *Quartus II Handbook*.

Design Walkthrough

This walkthrough explains how to create a JESD204B IP core design using Qsys in the Quartus II software. After you generate a custom variation of the JESD204B IP core, you can incorporate it into your overall project.

Creating a New Quartus II Project

You can create a new Quartus II project with the **New Project Wizard**. This process allows you to:

- specify the working directory for the project.
- assign the project name.
- designate the name of the top-level design entity.

1. From the Windows Start menu, select **Programs > Altera > Quartus II** <version> to launch the Quartus II software. Alternatively, you can use the Quartus II Web Edition software.
2. On the **File** menu, click **New Project Wizard**.
3. In the **New Project Wizard: Directory, Name, Top-Level Entity** page, specify the working directory, project name, and top-level design entity name. Click **Next**.
4. In the **New Project Wizard: Add Files** page, select the existing design files (if any) you want to include in the project.[8] Click **Next**.
5. In the **New Project Wizard: Family & Device Settings** page, select the device family and specific device you want to target for compilation. Click **Next**.
6. In the **EDA Tool Settings** page, select the EDA tools you want to use with the Quartus II software to develop your project.
7. Review the summary of your chosen settings in the **New Project Wizard** window, then click **Finish** to complete the Quartus II project creation.

Parameterizing and Generating the IP Core

Before you begin

Refer to **Table 3-4** for the IP core parameter values and description.

1. In the IP Catalog (**Tools** > **IP Catalog**), locate and double-click the name of the IP core you want to customize.
2. Specify a top-level name for your custom IP variation. This name identifies the IP core variation files in your project. If prompted, also specify the target Altera device family and output file HDL preference. Click **OK**.
3. In the Main tab, select the the following options:
   - **Jesd204b wrapper**
   - **Data path**
   - **Jesd204b subclass**
   - **Data Rate**
   - **PCS Option**
   - **PLL Type**
   - **Bonding Mode**
   - **PLL/CDR Reference Clock Frequency**
   - **Enable Transceiver Dynamic Reconfiguration**
   - **Enable Altera Debug Master Endpoint**
   - **Enable Bit reversal and Byte reversal**
4. In the Jesd204b Configurations tab, select the following configurations:

[8] To include existing files, you must specify the directory path to where you installed the JESD204B IP core. You must also add the user libraries if you installed the MegaCore IP Library in a different directory from where you installed the Quartus II software.

- **Common configurations (L, M, F, N, N', S, K)**
- **Advanced configurations (SCR, CS, CF, HD, ECC_EN, PHADJ, ADJCNT, ADJDIR)**

5. In the Configurations and Status Registers tab, set the the following configurations:
   - **Device ID**
   - **Bank ID**
   - **Lane ID**
   - **Lane checksum**
6. After parameterizing the core, click **Generate Example Design** to create the simulation testbench. Skip to **step 8** if you do not want to generate the design example.
7. Set a name for your <*example_design_directory*> and click **OK** to generate supporting files and scripts. The testbench and scripts are located in the <*example_design_directory*>**/ip_sim** folder.

   The **Generate Example Design** option generates supporting files for the following entities:
   - IP core for simulation—refer to **Generating and Simulating the IP Core Testbench** on page 3-7
   - IP core design example for simulation—refer to **Generating and Simulating the Design Example** on page 5-52
   - IP core design example for synthesis—refer to **Compiling the JESD204B IP Core Design Example** on page 5-54
8. Click **Finish** or **Generate HDL** to generate synthesis and other optional files matching your IP variation specifications. The parameter editor generates the top-level **.qip** or **.qsys** IP variation file and HDL files for synthesis and simulation.

   The top-level IP variation is added to the current Quartus II project. Click **Project** > **Add/Remove Files in Project** to manually add a **.qip** or **.qsys** file to a project. Make appropriate pin assignments to connect ports.

Note: Some parameter options are grayed out if they are not supported in a selected configuration or it is a derived parameter.

Generating and Simulating the IP Core Testbench

You can simulate your JESD204B IP core variation by using the provided IP core demonstration testbench.

To use the JESD204B IP core testbench, follow these steps:

1. Generate the simulation model. Refer to **Generating the Testbench Simulation Model** on page 3-7.
2. Simulate the testbench using the simulator-specific scripts that you have generated. Refer to **Simulating the IP Core Testbench** on page 3-8.

Note: Some configurations are preset and are not programmable in the JESD204B IP core testbench. For more details, refer to **JESD204B IP Core Testbench** on page 3-16 or the **README.txt** file located in the <*example_design_directory*>**/ip_sim** folder.

Generating the Testbench Simulation Model

To generate the testbench simulation model, execute the generated script (*gen_sim_verilog.tcl* or *gen_sim_vhdl.tcl*) located in the <*example_design_directory*>**/ip_sim** folder.

To run the Tcl script using the Quartus II sofware, follow these steps:

1. Launch the Quartus II software.
2. On the View menu, click **Utility Windows** > **Tcl Console**.
3. In the **Tcl Console**, type `cd <example_design_directory>/ip_sim` to go to the specified directory.
4. Type `source gen_sim_verilog.tcl` (Verilog) or `source gen_sim_vhdl.tcl` (VHDL) to generate the simulation files.

To run the Tcl script using the command line, follow these steps:

1. Obtain the Quartus II software resource.
2. Type `cd <example_design_directory>/ip_sim` to go to the specified directory.
3. Type `quartus_sh -t gen_sim_verilog.tcl` (Verilog) or `quartus_sh -t gen_sim_vhdl.tcl` (VHDL) to generate the simulation files.

Simulating the IP Core Testbench

The JESD204B IP core simulation supports the following simulators:

- ModelSim-Altera SE/AE
- VCS
- VCS MX
- Cadence
- Aldec Riviera

Note: VHDL is not supported in ModelSim-Altera AE and VCS simulators.

Table 3-2: Simulation Setup Scripts

This table lists the simulation setup scripts and run scripts.

Simulator	File Directory	Script
ModelSim®-Altera SE/AE	*<example_design_directory>*/ip_sim/testbench/setup_scripts/mentor	msim_setup.tcl
VCS	*<example_design_directory>*/ip_sim/testbench/setup_scripts/synopsys/vcs	vcs_setup.sh
VCS MX	*<example_design_directory>*/ip_sim/testbench/setup_scripts/synopsys/vcsmx	vcsmx_setup.sh synopsys_sim.setup
Aldec Riviera	*<example_design_directory>*/ip_sim/testbench/setup_scripts/aldec	rivierapro_setup.tcl
Cadence	*<example_design_directory>*/ip_sim/testbench/setup_scripts/cadence	ncsim_setup.sh

Table 3-3: Simulation Run Scripts

Simulator	File Directory	Script
ModelSim-Altera SE/AE	*<example_design_directory>*/ip_sim/testbench/mentor	run_altera_jesd204_tb.tcl
VCS	*<example_design_directory>*/ip_sim/testbench/synopsys/vcs	run_altera_jesd204_tb.sh
VCS MX	*<example_design_directory>*/ip_sim/testbench/synopsys/vcsmx	run_altera_jesd204_tb.sh
Aldec Riviera	*<example_design_directory>*/ip_sim/testbench/aldec	run_altera_jesd204_tb.tcl
Cadence	*<example_design_directory>*/ip_sim/testbench/cadence	run_altera_jesd204_tb.sh

To simulate the testbench design using the ModelSim-Altera or Aldec Riviera-PRO simulator, follow these steps:

1. Launch the ModelSim-Altera or Aldec Riviera-PRO simulator.
2. On the File menu, click **Change Directory** > **Select** *<example_design_directory>*/**ip_sim/testbench/** *<simulator name>*.
3. On the File menu, click **Load** > **Macro file**. Select **run_altera_jes204_tb.tcl**. This file compiles the design and runs the simulation automatically, providing a pass/fail indication on completion.

To simulate the testbench design using the VCS, VCS MX (in Linux), or Cadence simulator, follow these steps:

1. Launch the VCS, VCS MX, or Cadence simulator.
2. On the File menu, click **Change Directory** > **Select** *<example_design_directory>*/**ip_sim/testbench/** *<simulator name>*.
3. Run the **run_altera_jes204_tb.sh** file. This file compiles the design and runs the simulation automatically, providing a pass/fail indication on completion.

Related Information

Simulating Altera Designs
More information about Altera simulation models.

Compiling the JESD204B IP Core Design

Before you begin

Refer to the **JESD204B IP Core Design Considerations** on page 3-10 before compiling the JESD204B IP core design.

To compile your design, click **Start Compilation** on the Processing menu in the Quartus II software. You can use the generated **.qip** file to include relevant files into your project.

Related Information

- **JESD204B IP Core Design Considerations** on page 3-10
- **Quartus II Help**
  More information about compilation in Quartus II software.

Programming an FPGA Device

After successfully compiling your design, program the targeted Altera device with the Quartus II Programmer and verify the design in hardware. For instructions on programming the FPGA device, refer to the *Device Programming* section in volume 3 of the Quartus II Handbook.

Related Information

Device Programming

JESD204B IP Core Design Considerations

You must be aware of the following conditions when integrating the JESD204B IP core in your design:

- Intergrating the IP core in Qsys
- Pin Assignments
- Adding External Transceiver PLL

Integrating the JESD204B IP core in Qsys

You can integrate the JESD204B IP core with other Qsys components within Qsys.

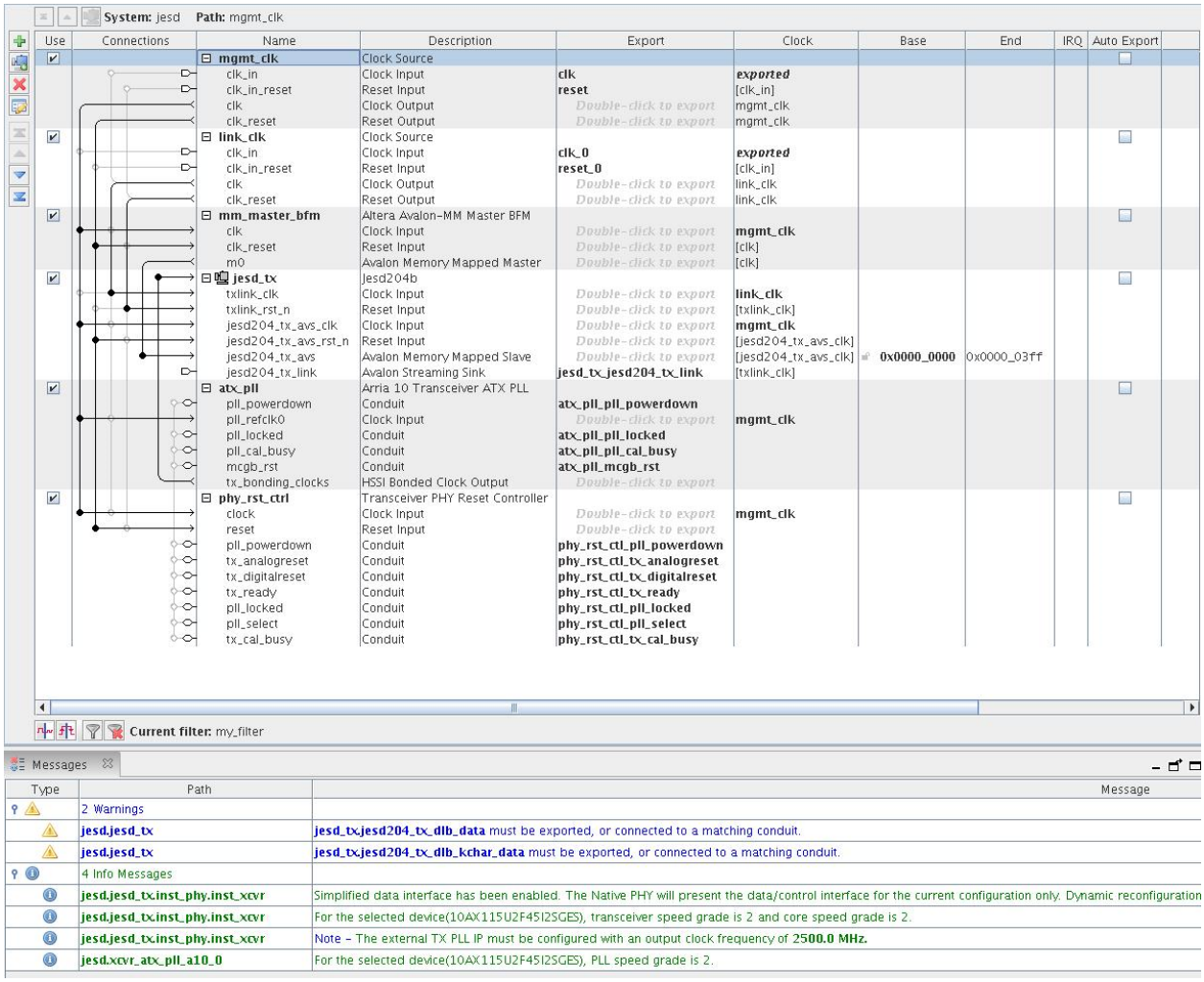
You can connect standard interfaces like clock, reset, Avalon-MM, Avalon-ST, HSSI bonded clock, HSSI serial clock, and interrupt interfaces within Qsys. However, for conduit interfaces, you are advised to export all those interfaces and handle them outside of Qsys. [(9)] This is because conduit interfaces are not part of the standard interfaces. Thus, there is no guarantee on compatibility between different conduit interfaces.

Note: The Transport Layer provided in this JESD204B IP core design example is not supported in Qsys. Therefore, you must export all interfaces that connect to the Transport Layer (for example, *jesd204_tx_link* interface) and connect them to a transport layer outside of Qsys.

(9) You can also connect conduit interfaces within Qsys but you must create adapter components to handle all the incompatibility issues like incompatible signal type and width.

Figure 3-4: Example of Connecting JESD204B IP Core with Other Qsys Components in Qsys

Figure shows an example of how you can connect the IP core with other Qsys components in Qsys.



Related Information

Transport Layer on page 5-8

Pin Assignments

Set the pin assignments before you compile to provide direction to the Quartus II software Fitter tool. You must also specify the signals that should be assigned to device I/O pins.

You can create virtual pins to avoid making specific pin assignments for top-level signals. This is useful when you want to perform compilation, but are not ready to map the design to hardware. Altera recommends that you create virtual pins for all unused top-level signals to improve timing closure.

Note: Do not create virtual pins for the clock or reset signals.

Adding External Transceiver PLL

The JESD204B IP core variations that target an Arria 10 FPGA device require external transceiver PLLs for compilation.

JESD204B IP core variations that target a V-series FPGA device contain transceiver PLLs. Therefore, no external PLLs are required for compilation.

You are recommend to use an ATX PLL or CMU PLL to get a better jitter performance.

Note: The PMA width is 20 bits for Hard PCS and 40 bits for Soft PCS.

Related Information

Arria 10 Transceiver PHY User Guide
More information about the Arria 10 transceiver PLLs and clock network.

JESD204B IP Core Parameters

Table 3-4: JESD204B IP Core Parameters

Parameter	Value	Description
Main Tab		
Device Family	• Arria V • Arria V GZ • Arria 10 • Stratix V	Select the targeted device family.
JESD204B Wrapper	• Base Only • PHY Only • Both Base and PHY	Select to generate base only (MAC), PHY only (PCS and PMA) , or both base and PHY layers integrated (MAC and PHY). The base only option is not supported for Duplex mode.
Data Path	• Receiver • Transmitter • Duplex	Select the operation modes. This selection enables or disables the receiver and transmitter supporting logic. • RX—instantiates the receiver to interface to the ADC • TX—instantiates the transmitter to interface to the DAC • Duplex—instantiates the receiver and transmitter to interface to both the ADC and DAC
JESD204B Subclass	• 0 • 1 • 2	Select the subclass modes. • 0—Set subclass 0 • 1—Set subclass 1 • 2—Set subclass 2

Parameter	Value	Description
Data Rate	1.0–12.5	Set the data rate for each lane. (10) • Arria V—1.0 Gbps to 6.55 Gbps • Arria V GZ—2.0 Gbps to 9.9 Gbps • Arria 10—2.0 Gbps to 12.5 Gbps • Stratix V—2.0 Gbps to 12.5 Gbps
PCS Option	• Enabled Hard PCS • Enabled Soft PCS	Select the PCS modes. • Enabled Hard PCS—utilize Hard PCS components. Select this option to minimize resource utilization with data rate that supports up to the Hard PCS's limitation. **Note:** For this setting, you will use 8G PCS mode with 20 bits PMA width and 32 bits PCS width. • Enabled Soft PCS—utilize Soft PCS components. Select this option to allow higher supported data rate but increase in resource utilization. **Note:** For this setting, you will use 10G PCS mode with 40 bits PMA width and 40 bits PCS width.
PLL Type	• CMU • ATX	Select the Phase-Locked Loop (PLL) types. Arria V device family only supports CMU PLL.
Bonding Mode	• Bonded • Non-bonded	Select the bonding modes. • Bonded—select this option to minimize inter-lanes skew for the transmitter datapath. • Non-bonded—select this option to disable inter-lanes skew control for the transmitter datapath. **Note:** The bonding type is automatically selected based on the device family and number of lanes that you set.
PLL/CDR Reference Clock Frequency	50.0–625.0	Set the transceiver reference clock frequency for PLL or CDR.
Enable Transceiver Dynamic Reconfiguration	On, Off	Turn on this option to enable dynamic data rate change. When you enable this option, you need to connect the reconfiguration interface to the transceiver reconfiguration controller. (11) For Arria 10 devices, turn on this option to enable the Transceiver Native PHY reconfiguration interface with "Share Reconfiguration Interface" enabled for multiple channels.

(10) The maximum data rate is limited by different device speed grade, transceiver PMA speed grade, and PCS options. Refer to the **Table 3-4** table for the maximum data rate support.

(11) To perform dynamic reconfiguration, you have to instantiate the Transceiver Reconfiguration Controller from the IP Catalog and connect it to the JESD204B IP core through the `reconfig_to_xcvr` and `reconfig_from_xcvr` interface.

Parameter	Value	Description
Enable Altera Debug Master Endpoint	On, Off	Turn on this option for the Transceiver Native PHY IP core to include an embedded Altera Debug Master Endpoint (ADME). This ADME connects internally to the Avalon-MM slave interface of the Transceiver Native PHY and can access the reconfiguration space of the transceiver. It can perform certain test and debug functions via JTAG using System Console. This parameter is valid only for Arria 10 devices and when you turn on the Enable Transceiver Dynamic Reconfiguration parameter. **Note:** This option requires you to include the JTAG Debug Link IP core in the system.
Enable Bit reversal and Byte reversal	On, Off	Turn on this option to set the data transmission order in MSB-first serialization. If this option is off, the data transmission order is in LSB-first serialization.
JESD204B Configurations Tab		
Lanes per converter device (L)	1–8	Set the number of lanes per converter device. (12)
Converters per device (M)	1–32	Set the number of converters per converter device.
Octets per frame (F)	1, 2, 4–256	The number of octets per frame derived from the formula of F= M*N'*S/(8*L).
Converter resolution (N)	1–32	Set the number of conversion bits per converter.
Transmitted bits per sample (N')	4–32	Set the number of transmitted bits per sample (JESD204B word size). **Note:** If parameter CF equals to 0 (no control word), parameter N' must be larger than or equal to sum of parameter N and parameter CS (N' ≥ N + CS). Otherwise, parameter N' must be larger than or equal to parameter N (N'≥N).
Samples per converter per frame (S)	1–32	Set the number of transmitted samples per converter per frame.
Frames per multiframe (K)	1–32	Set the number of frames per multiframe. This value is dependent on the value of F and is derived using the following constraints: • The value of K must fall within the range of 17/F <= K <= min(32, floor (1024/F)) • The value of F*K must be divisible by 4

(12) Refer to the **Table 3-6** table for the supported range for L.

Enable scramble (SCR)	On, Off	Turn on this option to scramble the transmitted data or descramble the receiving data.
Control Bits (CS)	0–3	Set the number of control bits per conversion sample.
Control Words (CF)	0–32	Set the number of control words per frame clock period per link.
High density user data format (HD)	On, Off	Turn on this option to set the data format. This parameter controls whether a sample may be divided over more lanes. • On: High Density format • Off: Data should not cross the lane boundary
Enable Error Code Correction (ECC_EN)	On, Off	Turn on this option to enable error code correction (ECC) for memory blocks.
Phase adjustment request (PHADJ)	On, Off	Turn on this option to specify the phase adjustment request to the DAC. • On: Request for phase adjustment • Off: No phase adjustment This parameter is valid for Subclass 2 mode only
Adjustment resolution step count (ADJCNT)	0–15	Set the adjustment resolution for the DAC LMFC. This parameter is valid for Subclass 2 mode only
Direction of adjustment (ADJDIR)	• Advance • Delay	Select to adjust the DAC LMFC direction. This parameter is valid for Subclass 2 mode only
Configurations and Status Registers Tab		
Device ID	0–255	Set the device ID number.
Bank ID	0–15	Set the device bank ID number.
Lane# ID	0–31	Set the lane ID number.
Lane# checksum	0–255	Set the checksum for each lane ID.

Related Information

Performance and Resource Utilization on page 2-6

JESD204B IP Core Component Files

The following table describes the generated files and other files that may be in your project directory. The names and types of generated files specified may vary depending on whether you create your design with VHDL or Verilog HDL.

Table 3-5: Generated Files

Extension	Description
<*variation name*>.**v** or **.vhd**	IP core variation file, which defines a VHDL or Verilog HDL description of the custom IP core. Instantiate the entity defined by this file inside of your design. Include this file when compiling your design in the Quartus II software.
<*variation name*>**.cmp**	A VHDL component declaration file for the IP core variation. Add the contents of this file to any VHDL architecture that instantiates the IP core.
<*variation name*>**.sdc**	Contains timing constraints for your IP core variation.
<*variation name*>**.qip**	Contains Quartus II project information for your IP core variation.
<*variation name*>**.tcl**	Tcl script file to run in Quartus II software.
<*variation name*>**.sip**	Contains IP core library mapping information required by the Quartus II software.The Quartus II software generates a **. sip** file during generation of some Altera IP cores. You must add any generated **.sip** file to your project for use by NativeLink simulation and the Quartus II Archiver.
<*variation name*>**.spd**	Contains a list of required simulation files for your IP core.

JESD204B IP Core Testbench

The JESD204B IP core includes a testbench to demonstrate a normal link-up sequence for the JESD204B IP core with a supported configuration. The testbench also provides an example of how to control the JESD204B IP core interfaces.

The testbench instantiates the JESD204B IP core in duplex mode and connects with the Altera Transceiver PHY Reset Controller IP core. Some configurations are preset and are not programmable in the JESD204B IP core testbench. For example, the JESD204B IP core always instantiates in duplex mode even if RX or TX mode is selected in the JESD204B parameter editor.

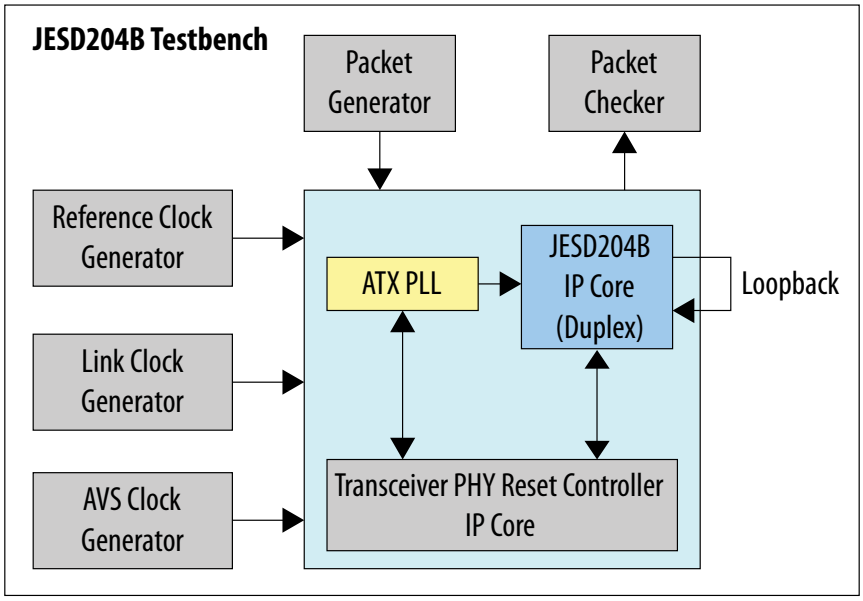
Note: Dynamic reconfiguration is not supported in this JESD204B IP core testbench.

Table 3-6: Preset Configurations for JESD204B IP Core Testbench

Configuration	Preset Value
JESD204B Wrapper	Base and PHY (MAC and PHY)
Data Path	Duplex
PLL/CDR Reference Clock Frequency	• data_rate/20 (if you turn on Enabled Hard PCS) • data_rate/40 (if you turn on Enabled Soft PCS)
Link Clock	Data rate/40
AVS Clock	125 MHz

Figure 3-5: JESD204B IP Core Testbench Block Diagram

The external ATX PLL is present only in the JESD204B IP core testbench targeting an Arria 10 FPGA device family.



Related Information

Generating and Simulating the IP Core Testbench on page 3-7

Testbench Simulation Flow

The JESD204B testbench simulation flow:

1. At the start, the system is under reset (all the components are in reset).
2. After 100 ns, the Transceiver Reset Controller IP core power up and wait for the `tx_ready` signal from the Transceiver Reset Controller IP to assert.
3. The reset signal of the JESD204B TX Avalon-MM interface is released (go HIGH) once the `tx_ready` signal is asserted. At the next positive edge of the `link_clk` signal, the JESD204B TX link powers up by releasing its reset signal.
4. The JESD204B TX link starts transmitting K28.5 characters and wait for the Transceiver Reset Controller IP core to assert the `rx_ready` signal.
5. The reset signal of the JESD204B RX Avalon-MM interface is released (go HIGH) once the `rx_ready` signal is asserted. At the next positive edge of the `link_clk` signal, the JESD204B RX link powers up by releasing its reset signal.
6. Once the link is out of reset, a *SYSREF* pulse is generated to reset the LMFC counter inside both the JESD204B TX and RX IP core.
7. When the `txlink_ready` signal is asserted, the packet generator starts sending packets to the TX datapath.
8. The packet checker starts comparing the packet sent from the TX datapath and received at the RX datapath after the `rxlink_valid` signal is asserted.
9. The testbench reports a pass or fail when all the packets are received and compared.

The testbench concludes by checking that all the packets have been received.

If no error is detected, the testbench issues a `TESTBENCH PASSED` message stating that the simulation was successful. If an error is detected, the testbench issues a `TESTBENCH FAILED` message to indicate that the testbench has failed.

JESD204B IP Core Functional Description

4

2014.12.15

UG-01142

The JESD204B IP core implements a transmitter (TX) and receiver (RX) block. Each block has two layers and consists of the following components:

- Media access control (MAC)—DLL block that consists of the link layer (link state machine and character replacement), CSR, Subclass 1 and 2 deterministic latency, scrambler or descrambler, and multiframe counter.
- Physical layer (PHY)—PCS and PMA block that consists of the 8B/10B encoder, word aligner, serializer, and deserializer.

You can specify the datapath and wrapper for your design and generate them separately.

The TX and RX blocks in the DLL utilizes the Avalon-ST interface to transmit or receive data and the Avalon-MM interface to access the CSRs. The TX and RX blocks operate on 32-bit data width per channel, where the frame assembly packs the data into four octets per channel. Multiple TX and RX blocks can share the clock and reset if the link rates are the same.

Figure 4-1: Overview of the JESD204B IP Core Block Diagram

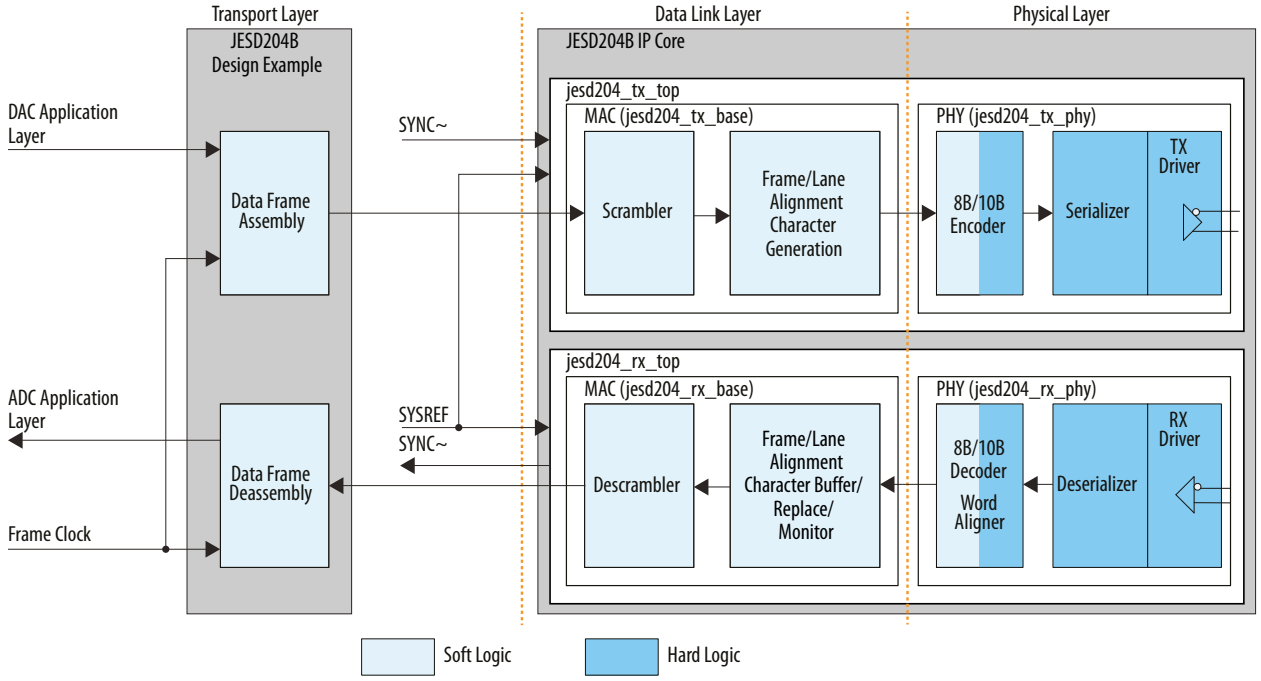
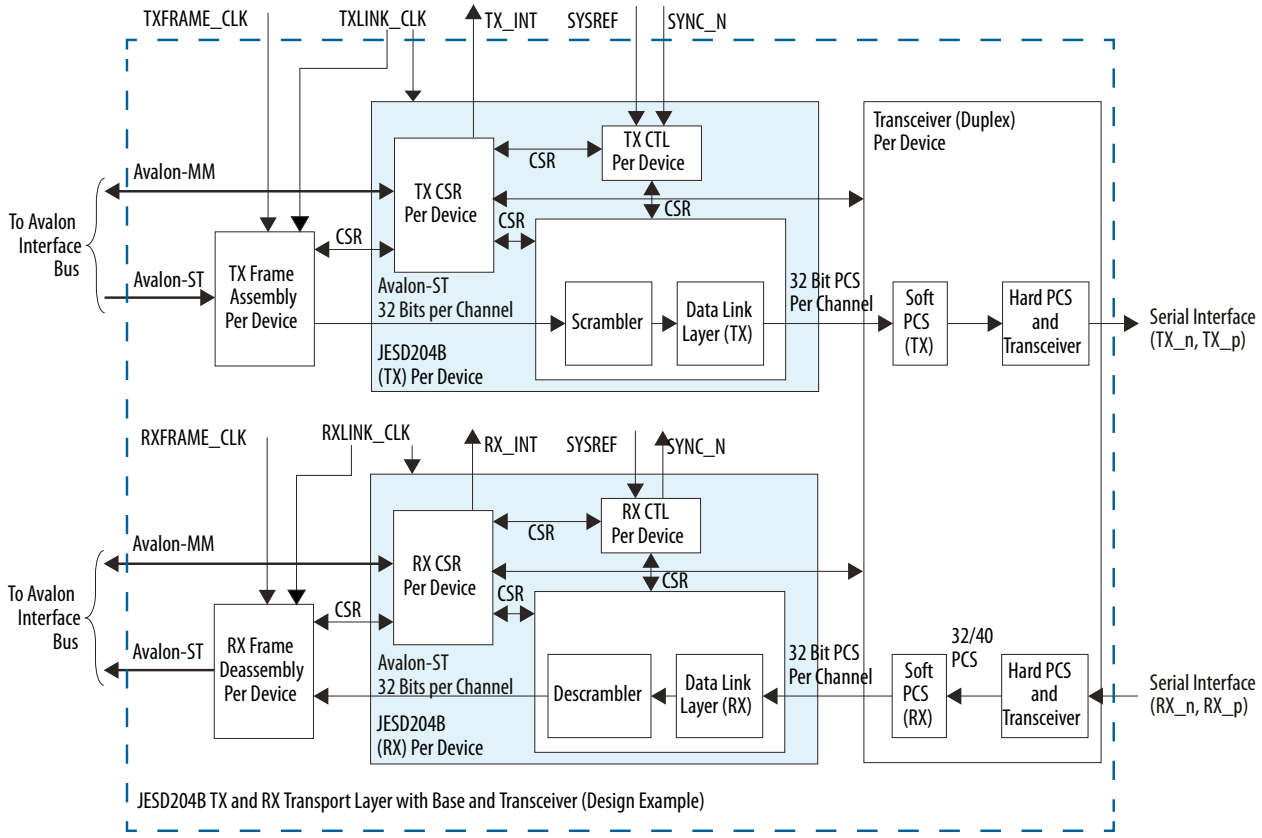


Figure 4-2: JESD204B IP Core TX and RX Datapath Block Diagram

The JESD204B IP core utilizes the Avalon-ST source and sink interfaces, with unidirectional flow of data, to transmit and receive data on the FPGA fabric interface.



32-Bits Architecture

The JESD204B IP core consist of 32-bit internal datapath per lane. This means that JESD204B IP Core expects the data samples to be assembled into 32-bit data (4 octets) per lane in the transport layer before sending the data to the Avalon-ST data bus. The JESD204 IP core operates in the link clock domain. The link clock runs at (data rate/40) because it is operating in 32-bit data bus after 8B/10B encoding.

As the internal datapath of the core is 32-bits, the (F × K) value must be in the order of 4 to align the multi-frame length on a 32-bit boundary. Apart from this, the deterministic latency counter values such as LMFC counter, RBD counter, and Subclass 2 adjustment counter will be in link clock count instead of frame clock count.

Avalon-ST Interface

The JESD204 IP core and transport layer in the design example use the Avalon-ST source and sink interfaces. There is no backpressure mechanism implemented in this core. The JESD204B IP core expects continuous stream of data samples from the upstream device.

Avalon-MM Interface

The Avalon-MM slave interface provides access to internal CSRs. The read and write data width is 32-bits (DWORD access). The Avalon-MM slave is asynchronous to the `txlink_clk`, `txframe_clk`, `rxlink_clk`, and `rxframe_clk` clock domains. You are recommended to release the reset for the CSR configuration space first. All run-time JESD204B configurations like L, F, M, N, N', CS, CF, and HD should be set before releasing the reset for link and frame clock domain.

Each write transfer has a *writeWaitTime* of 0 cycle while a read transfer has a *readWaitTime* of 1 cycle and *readLatency* of 1 cycle.

Related Information

Avalon Interface Specification
More information about the Avalon-ST and Avalon-MM interfaces, including timing diagrams.

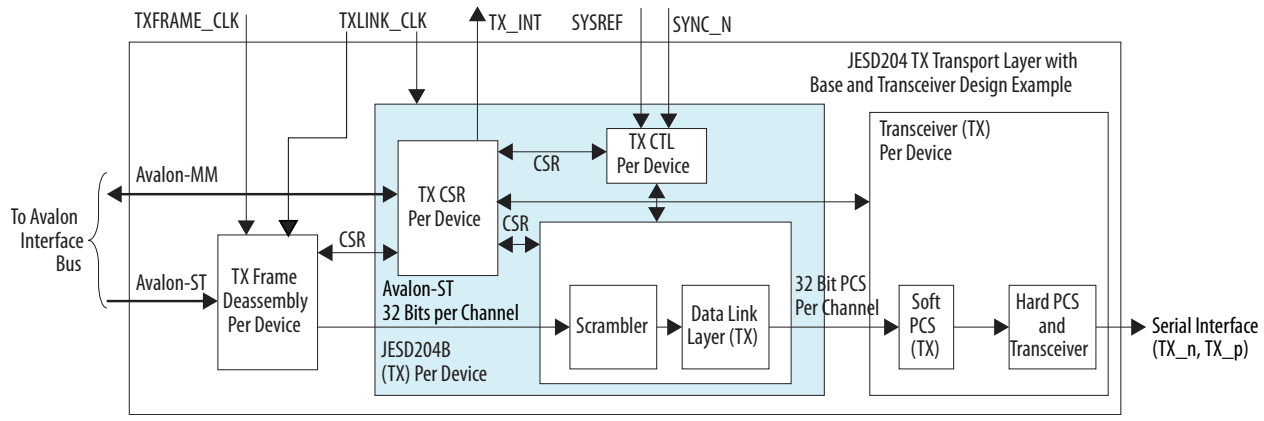
Transmitter

The transmitter block, which interfaces to DAC devices, takes one of more digital sample streams and converts them into one or more serial streams.

The transmitter performs the following functions:

- Data scrambling
- Frame or lane alignment
- Character generation
- Serial lane monitoring
- 8B/10B encoding
- Data serializer

Figure 4-3: Transmitter Data Path Block Diagram



The transmitter block consists of the following modules:

- TX CSR—manages the configuration and status registers.
- TX_CTL—manages the SYNC_N signal, state machine that controls the data link layer states, LMFC, and also the deterministic latency throughout the link.
- TX Scrambler and Data Link Layer—takes in 32-bits of data that implements the Initial Lane Alignment Sequence (ILAS), performs scrambling, lane insertion and frame alignment of characters.

TX Data Link Layer

The JESD204B IP core TX data link layer includes three phases to establish a synchronized link—Code Group Synchronization (CGS), Initial Lane Synchronization (ILAS), and User Data phase.

TX CGS

The CGS phase is achieved through the following process:

- Upon reset, the converter device (RX) issues a synchronization request by driving SYNC_N low. The JESD204 TX IP core transmits a stream of /K/ = /K28.5/ symbols. The receiver synchronizes when it receives four consecutive /K/ symbols.
- For Subclass 0, the RX converter devices deassert SYNC_N signal at the frame boundary. After all receivers have deactivated their synchronization requests, the JESD204 TX IP core continues to emit /K/ symbols until the start of the next frame. The core proceeds to transmit ILAS data sequence or encoded user data if csr_lane_sync_en signal is disabled.
- For Subclass 1 and 2, the RX converter devices deassert SYNC_N signal at the LMFC boundary. After all receivers deactivate the SYNC_N signal, the JESD204 TX IP core continues to transmit /K/ symbols until the next LMFC boundary. At the next LMFC boundary, the JESD204B IP core transmits ILAS data sequence. (There is no programmability to use a later LMFC boundary.)

TX ILAS

When lane alignment sequence is enabled through the csr_lane_sync_en register, the ILAS sequence is transmitted after the CGS phase. The ILAS phase takes up four multi-frames. For Subclass 0 mode, you can program the CSR (csr_ilas_multiframe) to extend the ILAS phase to a maximum of 256 multi-frames before transitioning to the encoded user data phase. The ILAS data is not scrambled regardless of whether scrambling is enabled or disabled.

The multi-frame has the following structure:

- Each multi-frame starts with a /R/ character (K28.0) and ends with a /A/ character (K28.3)
- The second multi-frame transmits the ILAS configuration data. The multi-frame starts with /R/ character (K28.0), followed by /Q/ character (K28.4), and then followed by the link configuration data, which consists of 14 octets as illustrated in the table below. It is then padded with dummy data and ends with /A/ character (K28.3), marking the end of multi-frame.
- Dummy octets are an 8-bit counter and is always reset when it is not in ILAS phase.
- For a configuration of more than four multi-frames, the multi-frame follows the same rule above and is padded with dummy data in between /R/ character and /A/ character.

Table 4-1: Link Configuration Data Transmitted in ILAS Phase

Configuration Octet	Bits								Description
	MSB	6	5	4	3	2	1	LSB	
0	DID[7:0]								DID = Device ID
1	ADJCNT[3:0]				BID[3:0]				ADJCNT = Number of adjustment resolution steps (13) BID = Bank ID
2	0	ADJDIR	PHADJ	LID[4:0]					ADJDIR = Direction to adjust DAC LMFC (13) PHADJ = Phase adjustment request (13) LID = Lane ID
3	SCR	0	0	L[4:0]					SCR = Scrambling enabled/disabled L = Number of lanes per device (link)
4	F[7:0]								F = Number of octets per frame per lane
5	0	0	0	K[4:0]					K = Number of frames per multi-frame
6	M[7:0]								M = Number of converters per device
7	CS[1:0]		0	N[4:0]					CS = Number of control bits per sample N = Converter resolution
8	SUBCLASSV[2:0]			N_PRIME[4:0]					SUBCLASSV = Subclass version N_PRIME = Total bits per sample
9	JESDV[2:0]			S[4:0]					JESDV = JESD204 version S = Number of samples per converter per frame
10	HD	0	0	CF[4:0]					HD = High Density data format CF = Number of control words per frame clock per link

(13) Applies to Subclass 2 only.

Configuration Octet	Bits								Description
	MSB	6	5	4	3	2	1	LSB	
11	RES1[7:0]								RES1 = Reserved. Set to 8'h00
12	RES2[7:0]								RES2 = Reserved. Set to 8'h00
13	FCHK[7:0]; automatically calculated using run-time configuration.								FCHK is the modulus 256 of the sum of the 13 configuration octets above. If you change any of the octets during run-time, make sure to update the new FCHK value in the register.

The JESD204 TX IP core also supports debug feature to continuously stay in ILAS phase without exiting. You can enable this feature by setting the bit in `csr_ilas_loop` register. There are two modes of entry:

- RX asserts `SYNC_N` and deasserts it after CGS phase. This activity triggers the ILAS phase and the CSR will stay in ILAS phase indefinitely until this setting changes.
- Link reinitialization through CSR is initiated. The JESD204B IP core transmits /K/ character and causes the RX converter to enter CGS phase. After RX deasserts `SYNC_N`, the CSR enters ILAS phase and will stay in that phase indefinitely until this setting changes.

In ILAS loop, the multi-frame transmission is the same where /R/ character (K28.0) marks the start of multi-frame and /A/ character (K28.3) marks the end of multi-frame, with dummy data in between. The dummy data is an increment of Dx.y.

User Data Phase

During the user data phase, character replacement at the end of frame and end of multi-frame is opportunistically inserted so that there is no additional overhead for data bandwidth.

Character replacement for non-scrambled data

The character replacement for non-scrambled mode in the IP core follows these JESD204B specification rules:

- At end of frame (not coinciding with end of multi-frame), which equals the last octet in the previous frame, the transmitter replaces the octet with /F/ character (K28.7). However, if an alignment character was transmitted in the previous frame, the original octet will be encoded.
- At the end of a multi-frame, which equals to the last octet in the previous frame, the transmitter replaces the octet with /A/ character (K28.3), even if a control character was already transmitted in the previous frame.

For devices that do not support lane synchronization, only /F/ character replacement is done. At every end of frame, regardless of whether the end of multi-frame equals to the last octet in previous frame, the transmitter encodes the octet as /F/ character (K28.7) if it fits the rules above.

Character replacement for scrambled data

The character replacement for scrambled data in the IP core follows these JESD204B specification rules:

- At end of frame (not coinciding with end of multi-frame), which equals to 0xFC (D28.7), the transmitter encodes the octet as /F/ character (K28.7).
- At end of multi-frame, which equals to 0x7C, the transmitter replaces the current last octet as /A/ character (K28.3).

For devices that do not support lane synchronization, only /F/ character replacement is done. At every end of frame, regardless of whether the end of multi-frame equals to 0xFC (D28.7), the transmitter encodes the octet as /F/ character (K28.7) if it fits the rules above.

TX PHY Layer

The 8B/10B encoder encodes the data before transmitting them through the serial line. The 8B/10B encoding has sufficient bit transition density (3-8 transitions per 10-bit symbol) to allow clock recovery by the receiver. The control characters in this scheme allow the receiver to:

- synchronize to 10-bit boundary.
- insert special character to mark the start and end of frames and start and end of multi-frames.
- detect single bit errors.

The JESD204 IP core supports transmission order from MSB first as well as LSB first. For MSB first transmission, the serialization of the left-most bit of 8B/10B code group (bit "a") is transmitted first.

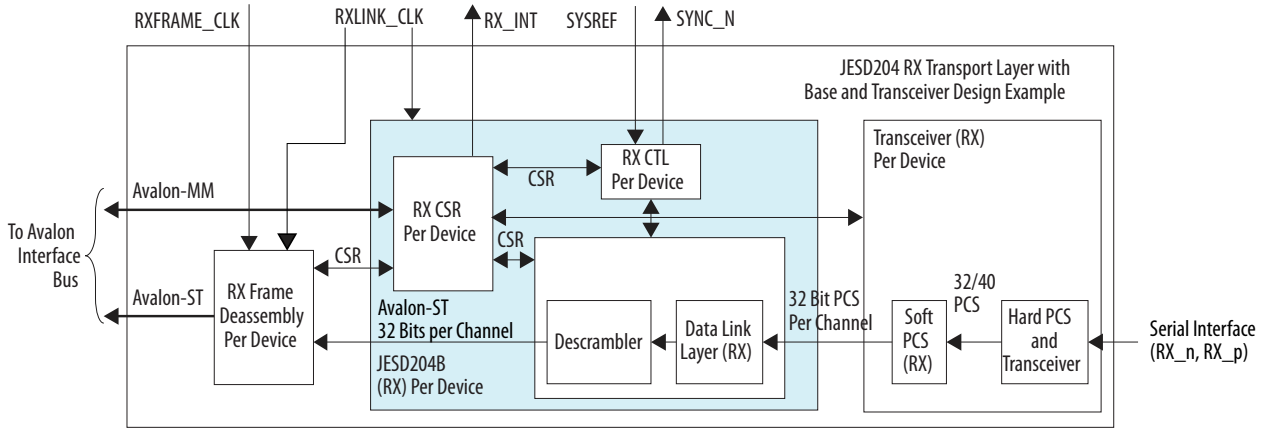
Receiver

The receiver block, which interfaces to ADC devices, receives the serial streams from one or more TX blocks and converts the streams into one or more sample streams.

The receiver performs the following functions:

- Data deserializer
- 8B/10B decoding
- Lane alignment
- Character replacement
- Data descrambling

Figure 4-4: Receiver Data Path Block Diagram



The receiver block includes the following modules:

- RX CSR—manages the configuration and status registers.
- RX_CTL—manages the SYNC_N signal, state machine that controls the data link layer states, LMFC, and also the buffer release, which is crucial for deterministic latency throughout the link.
- RX Scrambler and Data Link Layer—takes in 32-bits of data that decodes the ILAS, performs descrambling, character replacement as per the JESD204B specification, and error detection (code group error, frame and lane realignment error).

RX Data Link Layer

The JESD204B IP core RX data link layer buffers incoming user data on all lanes until the RX elastic buffers can be released. Special character substitution are done in the TX link so that the RX link can execute frame and lane alignment monitoring based on the JESD204B specification.

RX CGS

The CGS phase is the link up phase that monitors the detection of /K28.5/ character.

The CGS phase is achieved through the following process:

- Once the word boundary is aligned, the RX PHY layer detects the /K28.5/ 20-bit boundary and indicate that the character is valid.
- The receiver deasserts SYNC_N on the next frame boundary (for Subclass 0) or on the next LMFC boundary (for Subclass 1 and 2) after the reception of four successive /K/ characters.
- After correct reception of another four 8B/10B characters, the receiver assumes full code group synchronization. Error detected in this state machine is the code group error. Code group error always trigger link reinitialization through the assertion of SYNC_N signal and this cannot be disabled through the CSR. The CS state machine is defined as CS_INIT, CS_CHECK, and CS_DATA.
- The minimum duration for a synchronization request on the SYNC_N is five frames plus nine octets.

Frame Synchronization

After CGS phase, the receiver assumes that the first non-/K28.5/ character marks the start of frame and multi-frame. If the transmitter emits an initial lane alignment sequence, the first non-/K28.5/ character

will be /K28.0/. Similar to the JESD204 TX IP core, the csr_lane_sync_en is set to 1 by default, thus the RX core detects the /K/ character to /R/ character transition. If the csr_lane_sync_en is set to 0, the RX core detects the /K/ character to the first data transition. An ILAS error and unexpected /K/ character is flagged if either one of these conditions are violated.

When csr_lane_sync_en is set to 0, you have to disable data checking for the first 16 octets of data as the character replacement block takes 16 octets to recover the end-of-frame pointer for character replacement. When csr_lane_sync_en is set to 1 (default JESD204B setting), the number of octets to be discarded depends on the scrambler or descrambler block.

The receiver assumes that a new frame starts in every F octets. The octet counter is used for frame alignment and lane alignment.

Related Information

- **Scrambler/Descrambler** on page 4-14

Frame Alignment

The frame alignment is monitored through the alignment character /F/. The transmitter inserts this character at the end of frame. The /A/ character indicates the end of multi-frame. The character replacement algorithm depends on whether scrambling is enabled or disabled, regardless of the csr_lane_sync_en register setting.

The alignment detection process:

- If two successive valid alignment characters are detected in the same position other than the assumed end of frame—without receiving a valid or invalid alignment character at the expected position between two alignment characters—the receiver realigns its frame to the new position of the received alignment characters.
- If lane realignment can result in frame alignment error, the receiver issues an error.

In the JESD204 RX IP core, the same flexible buffer is used for frame and lane alignment. Lane realignment gives a correct frame alignment because lane alignment character doubles as a frame alignment character. A frame realignment can cause an incorrect lane alignment or link latency. The course of action is for the RX to request for reinitialization through SYNC_N. (14)

Lane Alignment

After the frame synchronization phase has entered FS_DATA, the lane alignment is monitored via /A/ character (/K28.3/) at the end of multi-frame. The first /A/ detection in the ILAS phase is important for the RX core to determine the minimum RX buffer release for inter-lane alignment. There are two types of error that is detected in lane alignment phase:

- Arrival of /A/ character from multiple lanes exceed one multi-frame.
- Misalignment detected during user data phase.

(14) Dynamic frame realignment and correction is not supported.

The realignment rules for lane alignment are similar to frame alignment:

- If two successive and valid /A/ characters are detected at the same position other than the assumed end of multi-frame—without receiving a valid/invalid /A/ character at the expected position between two /A/ characters—the receiver aligns the lane to the position of the newly received /A/ characters.
- If a recent frame alignment causes the loss of lane alignment, the receiver realigns the lane frame—which is already at the position of the first received /A/ character—at the unexpected position.

ILAS Data

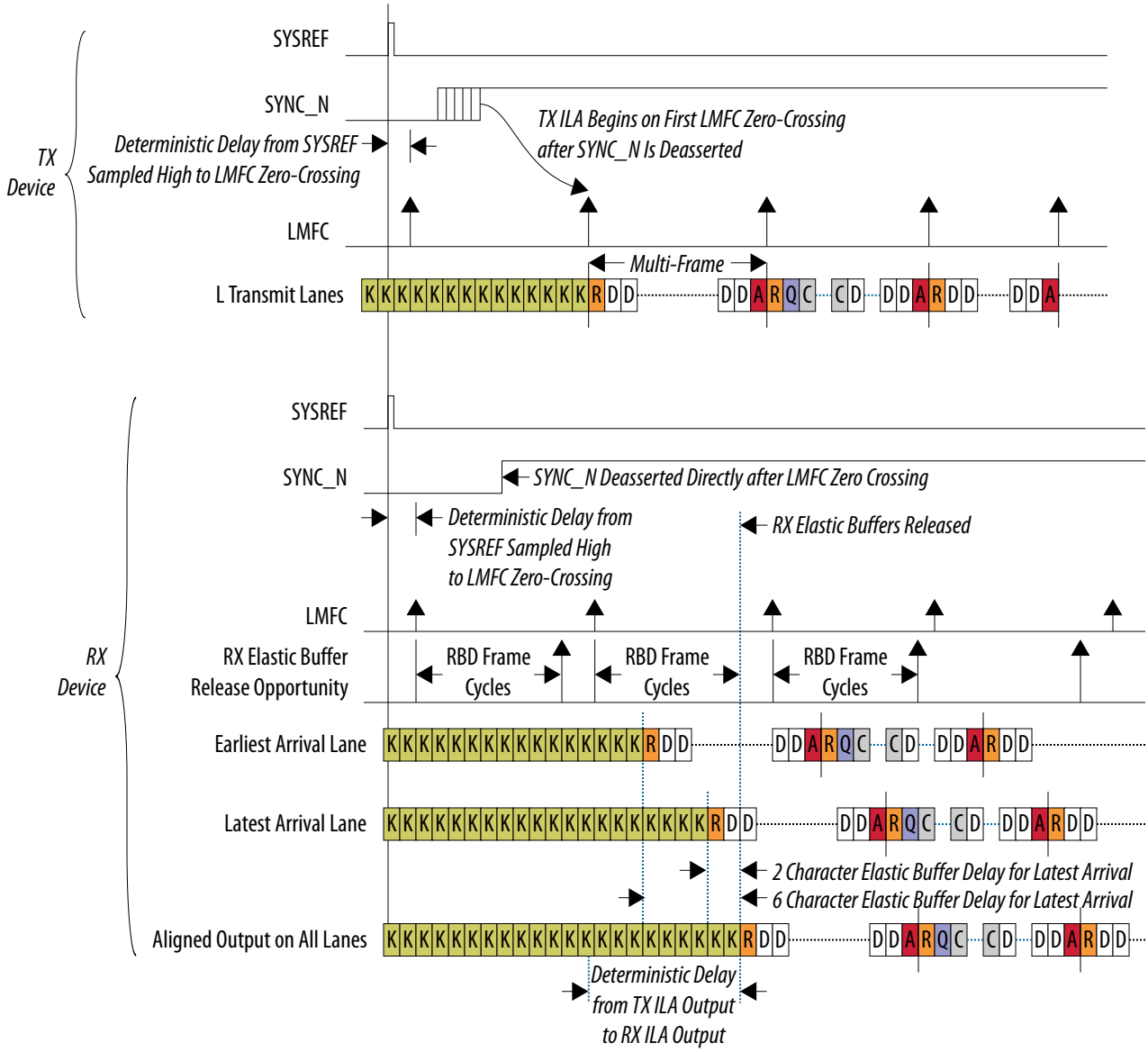
The JESD204 RX IP core captures 14 octets of link configuration data that are transmitted on the 2nd multi-frame of the ILAS phase. The receiver waits for the reception of /Q/ character that marks the start of link configuration data and then latch it into ILAS octets, which are per lane basis. You can read the 14 octets captured in the link configuration data through the CSR. You need to first set the `csr_ilas_data_sel` register to select which link configuration data lane it is trying to read from. Then, proceed to read from the `csr_ilas_octet` register.

Initial Lane Synchronization

The receivers in Subclass 1 and Subclass 2 modes store data in a memory buffer (Subclass 0 mode does not store data in the buffer but immediately releases them on the frame boundary as soon as the latest lane arrives.). The RX IP core detects the start of multi-frame of user data per lane and then wait for the latest lane data to arrive. The latest data is reported as RBD count (`csr_rbd_count`) value which you can read from the status register. This is the earliest release opportunity of the data from the deskew FIFO (referred to as RBD offset).

The JESD204 RX IP core supports RBD release at 0 offset and also provides programmable offset through RBD count. By default, the RBD release can be programmed through the `csr_rbd_offset` to release at the LMFC boundary. If you want to implement an early release mechanism, program it in the `csr_rbd_offset` register. The `csr_rbd_offset` and `csr_rbd_count` is a counter based on the link clock boundary (not frame clock boundary). Therefore, the RBD release opportunity is at every four octets.

Figure 4-5: Subclass 1 Deterministic Latency and Support for Programmable Release Opportunity



RX PHY Layer

The word aligner block identifies the MSB and LSB boundaries of the 10-bit character from the serial bit stream. Manual alignment is set because the /K/ character must be detected in either LSB first or MSB first mode. When the programmed word alignment pattern is detected in the current word boundary, the PCS indicates a valid pattern in the `rx_sync_status` (mapped as `pcs_valid` to the IP core). The code synchronization state is detected after the detection of the /K/ character boundary for all lanes.

In a normal operation, whenever synchronization is lost, JESD204 RX IP core always return back to the `CS_INIT` state where the word alignment is initiated. For debug purposes, you can bypass this alignment by setting the `csr_patternalign_en` register to 0.

The 8B/10B decoder decode the data after receiving the data through the serial line. The JESD204 IP core supports transmission order from MSB first as well as LSB first.

The PHY layer can detect 8B/10B not-in-table (NIT) error and also running disparity error.

Operation

Operating Modes

The JESD204B IP core supports Subclass 0, 1, and 2 operating modes.

Subclass 0

The JESD204 IP core maintains a LMFC counter that counts from 0 to (F × K/4)–1 and wraps around again. The LMFC counter starts counting at the deassertion of *SYNC_N* signal from multiple DACs after synchronization. This is to align the LMFC counter upon transmission and can only be done after all the converter devices have deasserted its synchronization request signal.

Subclass 1

The JESD204 IP core maintains a LMFC counter that counts from 0 to (F × K/4)–1 and wraps around again. The LMFC counter will reset within two link clock cycles after converter devices issue a common *SYSREF* frequency to all the transmitters and receivers. The *SYSREF* frequency must be the same for converter devices that are grouped and synchronized together.

Table 4-2: Example of SYSREF Frequency Calculation

In this example, you can choose to perform one of the following options:

- provide two *SYSREF* and device clock, where the ADC groups share both the device clock and *SYSREF* (18.75 MHz and 9.375 MHz)
- provide one *SYSREF* (running at 9.375 MHz) and device clock for all the ADC and DAC groups because the *SYSREF* period in the DAC is a multiplication of *n* integer.

Group	Configuration	SYSREF Frequency
ADC Group 1 (2 ADCs)	• LMF = 222 • K = 16 • Data rate = 6 Gbps	6 GHz / 40 / (2 x 16 / 4) = 18.75 MHz
ADC Group 2 (2 ADCs)	• LMF = 811 • K = 32 • Data rate = 6 Gbps	6 GHz / 40 / (1 x 32 / 4) = 18.75 MHz
DAC Group 3 (2 DACs)	• LMF = 222 • K = 16 • Data rate = 3 Gbps	3 GHz / 40 / (2 x 16 / 4) = 9.375 MHz

Subclass 2

The JESD204 IP core maintains a LMFC counter that counts from 0 to (F × K/4)–1 and wraps around again. The LMFC count starts upon reset and the logic device always acts as the timing master. The

converters adjust their own internal LMFC to match the master's counter. The alignment of LMFC within the system relies on the correct alignment of *SYNC_N* signal deassertion at the LMFC boundary.

The alignment of LMFC to RX logic is handled within the TX converter. The RX logic releases *SYNC_N* at the LMFC tick and the TX converter adjust its internal LMFC to match the RX LMFC.

For the alignment of LMFC to the TX logic, the JESD204 TX IP core samples *SYNC_N* from the DAC receiver and reports the relative phase difference between the DAC and TX logic device LMFC in the TX CSR (`dbg_phadj`, `dbg_adjdir`, and `dbg_adjcnt`). Based on the reported value, you can calculate the adjustment required. Then, to initiate the link reinitialization through the CSR, set the value in the TX CSR (`csr_phadj`, `csr_adjdir`, and `csr_adjcnt`). The values on the phase adjustment are embedded in bytes 1 and 2 of the ILAS sequence that is sent to the DAC during link initialization. On the reception of the ILAS, the DAC adjusts its LMFC phase by step count value and sends back an error report with the new LMFC phase information. This process may be repeated until the LMFC at the DAC and the logic device are aligned.

Scrambler/Descrambler

Both the scrambler and descrambler are designed in a 32-bit parallel implementation and the scrambling/descrambling order starts from first octet with MSB first.

The JESD204 TX and RX IP core support scrambling by implementing a 32-bit parallel scrambler in each lane. The scrambler and descrambler are located in the JESD204 IP MAC interfacing to the Avalon-ST interface. You can enable or disable scrambling and this option applies to all lanes. Mixed mode operation, where scrambling is enabled for some lanes, is not permitted.

The scrambling polynomial:

$1 + x^{14} + x^{15}$

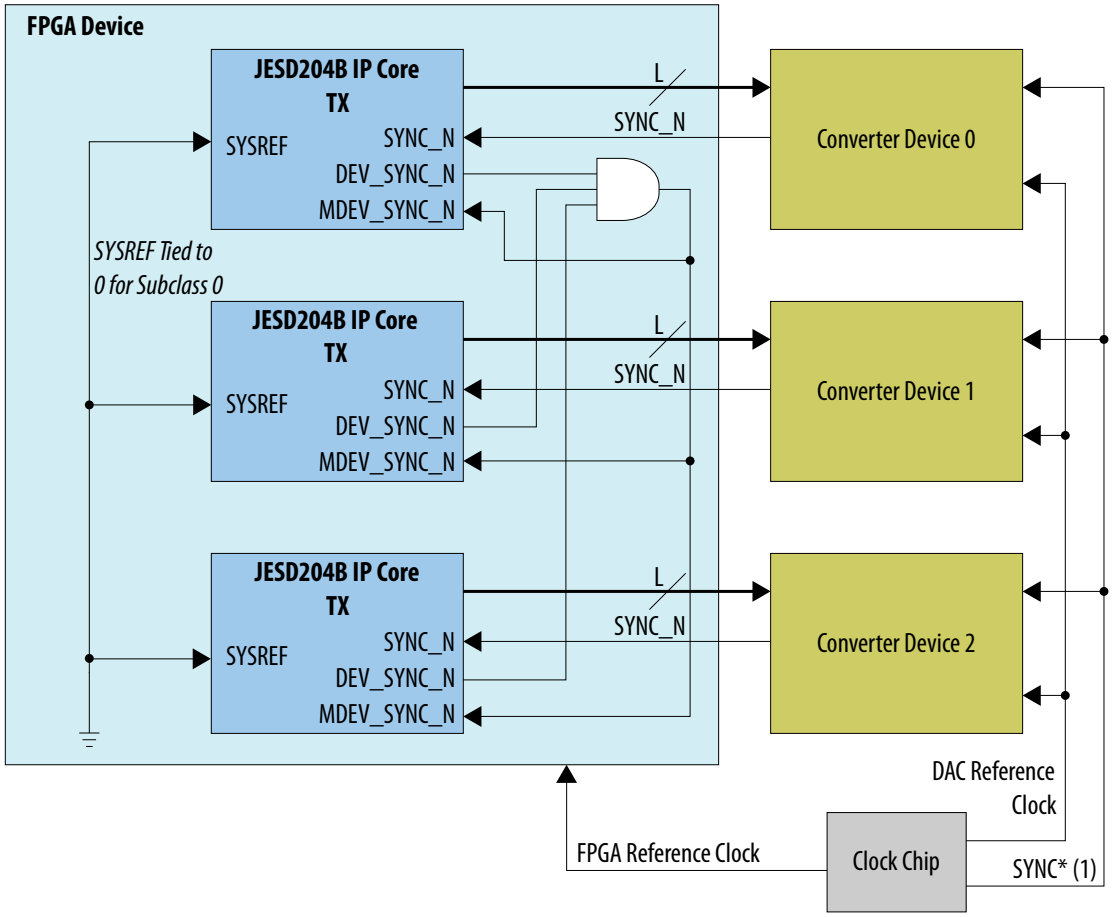
The descrambler can self-synchronize in eight octets. In a typical application where the reset value of the scrambler seed is different from the converter device to FPGA logic device, the correct user data is recovered in the receiver in two link clocks (due to the 32-bit architecture). The PRBS pattern checker on the transport layer should always disable checking of the first eight octets from the JESD204 RX IP core.

SYNC_N Signal

For Subclass 0 implementation, the `SYNC_N` signal from the DAC converters in the same group path must be combined.

In some applications, multiple converters are grouped together in the same group path to sample a signal (referred as multipoint link). The FPGA can only start the LMFC counter and its transition to ILAS after all the links deassert the synchronization request. The JESD204B TX IP core provides three signals to facilitate this application. The `SYNC_N` is the direct signal from the DAC converters. The error signaling from `SYNC_N` is filtered and sent out as `dev_sync_n` signal. For Subclass 0, you need to multiplex all the `dev_sync_n` signals in the same multipoint link and then input them to the IP core through `mdev_sync_n` signal.

Figure 4-6: Subclass 0 — Combining the SYNC_N Signal for JESD204B TX IP Core



Note:

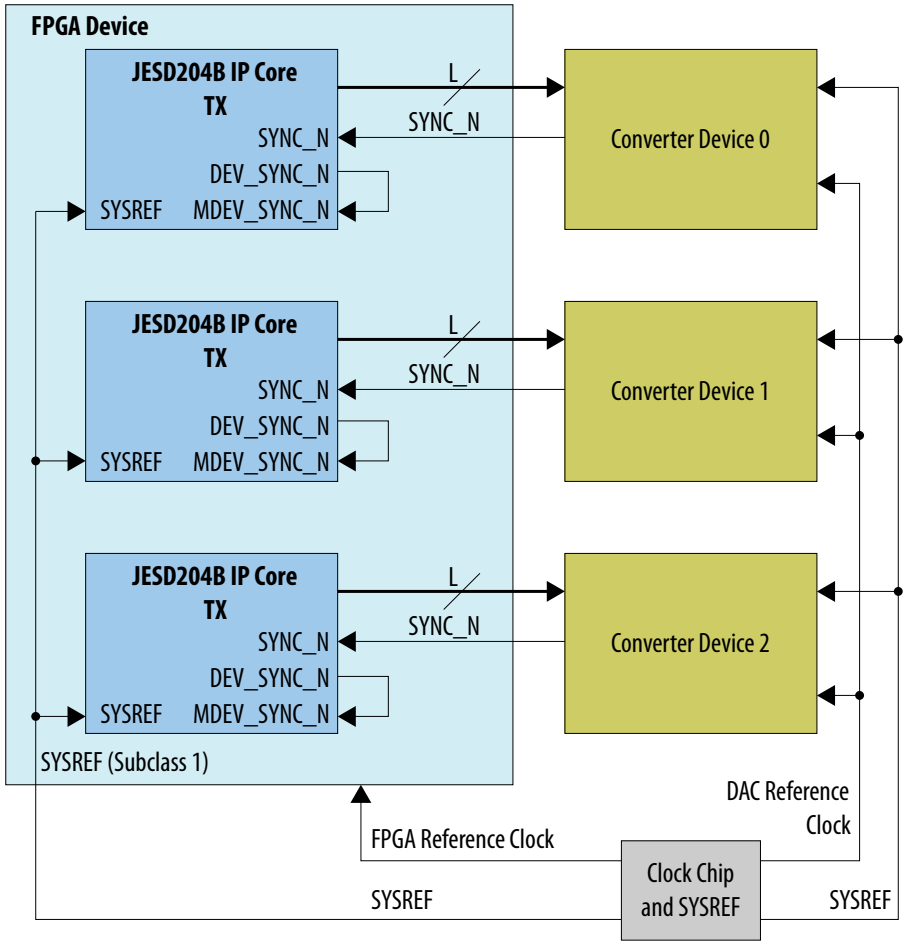
1. SYNC* is not associated to SYNC_N in the JESD204B specification. SYNC* refers to JESD204A (Subclass 0) converter devices that may support synchronization via additional SYNC signalling.

For Subclass 1 implementation, you may choose to combine or not to combine the SYNC_N signal from the converter device. If you implement two ADC converter devices as a multipoint link and one of the converter is unable to link up, the functional link will still operate. You must manage the trace length for the SYSREF signal and also the differential pair to minimize skew.

The SYNC_N is the direct signal from the DAC converters. The error signaling from SYNC_N is filtered and sent out as dev_sync_n output signal. The dev_sync_n signal from the JESD204B TX IP core must loopback into the mdev_sync_n signal of the same instance without combining the SYNC_N signal.

Apart from that, you must set the same RBD offset value (csr_rbd_offset) to all the JESD204B RX IP cores within the same multipoint link for the RBD release (the latest lane arrival for each of the links). The JESD204 RX IP core will deskew and output the data when the RBD offset value is met. The total latency is consistent in the system and is also the same across multiple resets. Setting a different RBD offset to each link or setting an early release does not guarantee deterministic latency and data alignment.

Figure 4-7: Subclass 1 — Combining the SYNC_N Signal for JESD204B TX IP Core



Link Reinitialization

The JESD204B TX and RX IP core support link reinitialization.

There are two modes of entry for link reinitialization:

- Hardware initiated link reinitialization:
  - For TX, the reception of SYNC_N for more than five frames and nine octets triggers link reinitialization.
  - For RX, the loss of code group synchronization, frame alignment and lane alignment errors cause the IP core to assert SYNC_N and request for link reinitialization.
- Software initiated link reinitialization—both the TX and RX IP core allow software to request for link reinitialization.
  - For TX, the IP core transmits /K/ character and wait for the receiver to assert SYNC_N to indicate that it has entered CS_INIT state.
  - For RX, the IP core asserts SYNC_N to request for link reinitialization.

Hardware initiated link reinitialization can be globally disabled through the csr_link_reinit_disable register for debug purposes.

Hardware initiated link reinitialization can be issued as interrupt depending on the error type and interrupt error enable. If lane misalignment has been detected as a result of a phase change in local timing reference, the software can rely on this interrupt trigger to initiates a LMFC realignment. The realignment process occurs by first resampling *SYSREF* and then issuing a link reinitialization request.

Link Startup Sequence

Set the run-time LMF configuration when the `txlink_rst_n` or `rxlink_rst_n` signals are asserted. Upon `txlink_rst_n` or `rxlink_rst_n` deassertion, the JESD204B IP core begins operation. The following sections describe the detailed operation for each subclass mode.

TX (Subclass 0)

Upon reset deassertion, the JESD204B TX IP core is in CGS phase. `SYNC_N` deassertion from the converter device enables the JESD204B TX IP core to exit CGS phase and enter ILAS phase (if `csr_lane_sync_en` = 1) or User Data phase (if `csr_lane_sync_en` = 0).

TX (Subclass 1)

Upon reset deassertion, the JESD204B TX IP core is in CGS phase. `SYNC_N` deassertion from the converter device enables the JESD204B TX IP core to exit CGS phase. The IP core ensures that at least one *SYSREF* rising edge is sampled before exiting CGS phase and entering ILAS phase. This is to prevent a race condition where the `SYNC_N` is deasserted before *SYSREF* is sampled. *SYSREF* sampling is crucial to ensure deterministic latency in the JESD204B Subclass 1 system.

TX (Subclass 2)

Similar to Subclass 1 mode, the JESD204B TX IP core is in CGS phase upon reset deassertion. The LMFC alignment between the converter and IP core starts after `SYNC_N` deassertion. The JESD204B TX IP core detects the deassertion of `SYNC_N` and compares the timing to its own LMFC. The required adjustment in the link clock domain is updated in the register map. You need to update the final phase adjustment value in the registers for it to transfer the value to the converter during the ILAS phase. The DAC adjusts the LMFC phase and acknowledge the phase change with an error report. This error report contains the new DAC LMFC phase information, which allows the loop to iterate until the phase between them is aligned.

RX (Subclass 0)

The JESD204B RX IP core drives and holds `SYNC_N` (`dev_sync_n` signal) low when it is in reset. Upon reset deassertion, the JESD204B RX IP core checks if there is sufficient /K/ character to move its state machine out of synchronization request. Once sufficient /K/ character is detected, the IP core deasserts `SYNC_N`.

RX (Subclass 1)

The JESD204B RX IP core drives and holds the `SYNC_N` (`dev_sync_n` signal) low when it is in reset. Upon reset deassertion, the JESD204B RX IP core checks if there is sufficient /K/ character to move its state machine out of synchronization request. The IP core also ensures that at least one *SYSREF* rising edge is sampled before deasserting `SYNC_N`. This is to prevent a race condition where the `SYNC_N` is deasserted based on internal free-running LMFC count instead of the updated LMFC count after *SYSREF* is sampled.

RX (Subclass 2)

The JESD204B RX IP core behaves the same as in Subclass 1 mode. In this mode, the logic device is always the master timing reference. Upon SYNC_N deassertion, the ADC adjusts the LMFC timing to match the IP core.

Error Reporting Through SYNC_N Signal

The JESD204 TX IP core can detect error reporting through SYNC_N when SYNC_N is asserted for two frame clock periods (if F >= 2) or four frame clock periods (if F = 1). When the downstream device reports an error through SYNC_N, the TX IP core issues an interrupt. The TX IP core samples the SYNC_N pulse width using the link clock.

For a special case of F = 1, two frame clock periods are less than one link clock. Therefore, the error signaling from the receiver may be lost. You must program the converter device to extend the SYNC_N pulse to four frame clocks when F = 1.

The JESD204 RX IP core does not report an error through SYNC_N signaling. Instead, the RX IP core issues an interrupt when any error is detected.

You can check the csr_tx_err, csr_rx_err0, and csr_rx_err1 register status to determine the error types.

Clocking Scheme

This section describes the clocking scheme for the JESD204B IP core and transceiver.

Table 4-3: JESD204B IP Core Clocks

Clock Signal	Formula	Description
TX/RX Device Clock: pll_ref_clk	PLL selection during IP core generation	The PLL reference clock used by the TX Transceiver PLL or RX CDR. This is also the recommended reference clock to the Altera PLL IP Core (for Arria V or Stratix V devices) or Altera IOPLL (for Arria10 devices).
TX/RX Link Clock: txlink_clk rxlink_clk	Data rate/40	The timing reference for the JESD204B IP core.The link clock runs at data rate/40 because the IP core is operating in a 32-bit data bus architecture after 8B/10B encoding. The JESD204B transport layer in the design example requires both the link clock and frame clock to be synchronous.

Clock Signal	Formula	Description

TX/RX Frame Clock (in design example):
`txframe_clk`
`rxframe_clk`
	Data rate/(10 × F)	The frame clock as per the JESD204B specification. This clock is applicable to the JESD204B transport layer and other upstream devices that run in frame clock such as the PRBS generator/checker or any data processing blocks that run at the same rate as the frame clock.
		The JESD204B transport layer in the design example also supports running the frame clock in half rate or quarter rate by using the *FRAMECLK_DIV* parameter. The JESD204B transport layer requires both the link clock and frame clock to be synchronous. For more information, refer to the F1/F2_FRAMECLK_DIV parameter description and its relationship to the frame clock.

TX/RX Transceiver Serial Clock and Parallel Clock	Internally derived from the data rate during IP core generation	The serial clock is the bit clock to stream out serialized data. The transceiver PLL supplies this clock and is internal to the transceiver. The parallel clock is for the transmitter PMA and PCS within the PHY. This clock is internal to the transceiver and is not exposed in the JESD204B IP core. For Arria V and Stratix V devices, these clocks are internally generated as the transceiver PLL is encapsulated within the JESD204B IP core's PHY. For Arria 10 devices, you need to generate the transceiver PLL based on the data rate and connect the serial and parallel clock. These clocks are referred to as `*serial_clk` and `*bonding_clock` in Arria 10 devices. Refer to the *Arria10 Transceiver PHY IP Core User Guide* for more information.

TX/RX PHY Clock:
`txphy_clk`
`rxphy_clk`
	Data rate/40	The link clock generated from the transceiver serial or parallel clock for the TX path or the link clock generated from the CDR for the RX path. This clock has the same frequency as the TX/RX link clock and is an output from the JESD204B IP core.
		There is limited use for this clock. Only if the JESD204B configuration is F=4 and operating at Subclass 0 mode, this clock can be used as input for both the `txlink_clk` and `txframe_clk`, or `rxlink_clk` and `rxframe_clk`.

Clock Signal	Formula	Description

TX/RX AVS Clock:
`jesd204_tx_avs_clk`
`jesd204_rx_avs_clk`
	75–125 MHz	The configuration clock for the JESD204B IP core CSR through the Avalon-MM interface.

Transceiver Management Clock: `reconfig_clk`	100 MHz–125 MHz	The configuration clock for the transceiver CSR through the Avalon-MM interface. This clock is exported only when the transceiver dynamic reconfiguration option is enabled. This clock is only applicable for Arria 10 devices.

Device Clock

In a converter device, the sampling clock is typically the device clock.

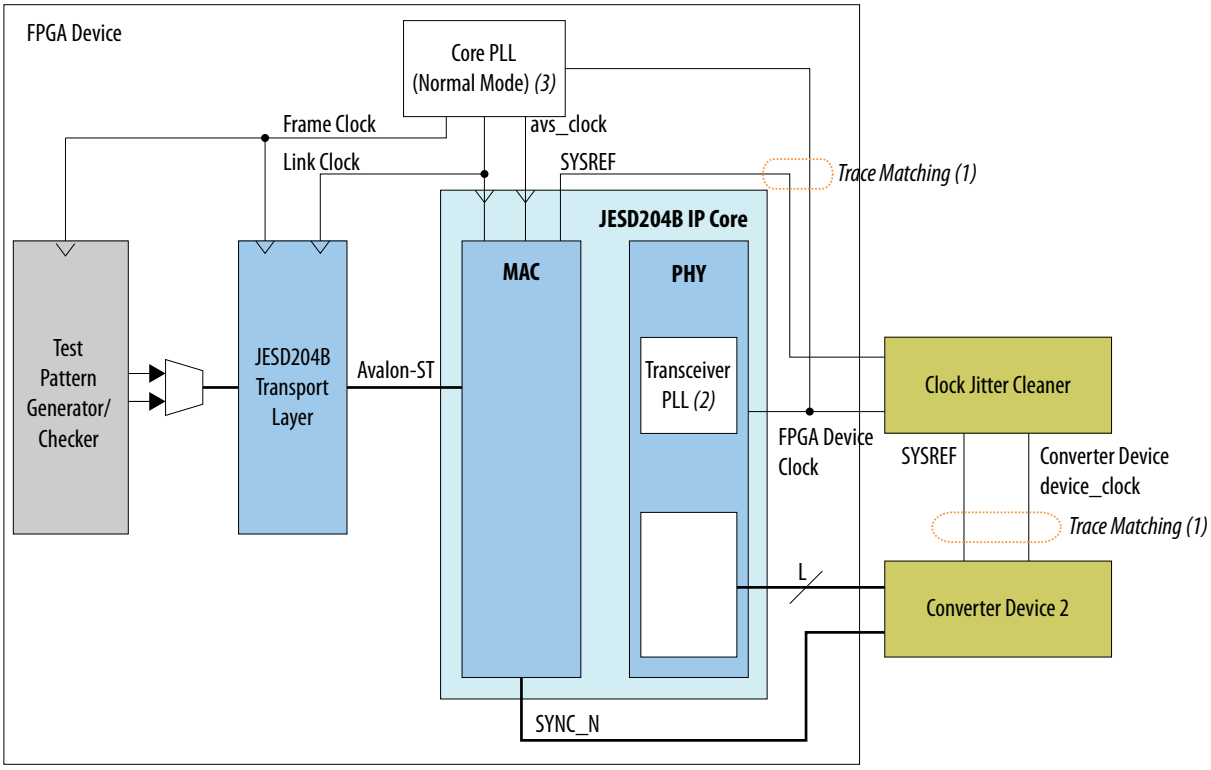
For the JESD204 IP core in an FPGA logic device, the device clock is used as the transceiver PLL reference clock and also the core PLL reference clock. The available frequency depends on the PLL type, bonding option, number of lanes, and device family. During IP core generation, the Quartus II software recommends the available device clock frequency for the transceiver PLL based on the user selection.

Note: You need to generate the Altera PLL IP core (in Arria V and Stratix V devices) or Altera IOPLL IP core (in Arria 10 devices) to generate the link clock and frame clock. The link clock is used in the JESD204 IP core (MAC) and the transport layer.

Based on the JESD204B specification, the device clock is the timing reference and is source synchronous with *SYSREF*. Due to the clock network architecture in the FPGA, you are recommended to use the device clock to generate the link clock and use the link clock as timing reference.

The JESD204B protocol does not support rate matching. Therefore, you must ensure that the TX or RX device clock (*pll_ref_clk*) and the PLL reference clock that generates link clock (*txlink_clk* or *rxlink_clk*) and frame clock (*txframe_clk* or *rxframe_clk*) have 0 ppm variation. Both PLL reference clocks should come from the same clock chip.

Figure 4-8: JESD204B Subsystem Clock Diagram (For Arria V and Stratix V Devices)



Notes:

1. The device clock to the Altera core PLL and SYSREF must be trace matched. The device clock to the converter device and SYSREF must be trace matched. The phase offset between the SYSREF to the FPGA and converter devices should be minimal.
2. For Arria 10 devices, the transceiver PLL is outside of the JESD204B IP core. For Arria V and Stratix V devices, the transceiver PLL is part of the JESD204B IP core.
3. The Altera core PLL provdes the link clock, frame clock, and AVS clock. The link clock and frame clock must be synchronous.

Related Information

Clock Correlation on page 4-23

Link Clock

The device clock is the timing reference for the JESD204B system.

Due to the clock network architecture in the FPGA, JESD204 IP core does not use the device clock to clock the *SYSREF* signal because the *GCLK* or *RCLK* is not fully compensated. You are recommended to use the Altera PLL IP core (in Arria V and Stratix V devices) or Altera IOPLL IP core (in Arria 10 devices) to generate both the link clock and frame clock. The Altera PLL IP core must operate in **normal mode** or **source synchronous mode** to achieve the following state:

- the *GCLK* and *RCLK* clock network latency is fully compensated.
- the link clock and frame clock at the registers are phase-aligned to the input of the clock pin.

To provide consistency across the design regardless of frame clock and sampling clock, the link clock is used as a timing reference.

The Altera PLL IP core should provide both the frame clock and link clock from the same PLL as these two clocks are treated as synchronous in the design.

For Subclass 0 mode, the device clock is not required to sample the *SYSREF* signal edge. The link clock does not need to be phase compensated to capture *SYSREF*. Therefore, you can generate both the link clock and frame clock using direct mode in the Altera PLL IP core. If F = 4, where link clock is the same as the frame clock, you can use the parallel clock output from the transceiver (`txphy_clk` or `rxphy_clk` signal).

Related Information

Clock Correlation on page 4-23

Local Multi-Frame Clock

The Local Multi-Frame Clock (LMFC) is a counter generated from the link clock and depends on the F and K parameter.

The K parameter must be set between 1 to 32 and meet the requirement of at least a minimum of 17 octets and a maximum of 1024 octets in a single multi-frame. In a 32-bit architecture, the K × F must also be in the order of four.

In a Subclass 1 deterministic latency system, the *SYSREF* frequency is distributed to the devices to align them in the system. The *SYSREF* resets the internal LMFC clock edge when the sampled *SYSREF* signal's rising edge transition from 0 to 1. Due to source synchronous signaling of *SYSREF* with respect to the device clock sampling (provided from the clock chip), the JESD204 IP core does not directly use the device clock to sample *SYSREF* but instead uses the link clock to sample *SYSREF*. Therefore, the Altera PLL IP core that provides the link clock must to be in **normal mode** to phase-compensate the link clock to the device clock.

Based on hardware testing, to get a fixed latency, at least 32 octets are recommended in an LMFC period so that there is a margin to tune the RBD release opportunity to compensate any lane-to-lane deskew across multiple resets. If F = 1, then K = 32 would be optimal as it provides enough margin for system latency variation. If F = 2, then K = 16 and above (18/20/22/24/26/28/30/32) is sufficient to compensate lane-to-lane deskew.

The JESD204B IP core implements the local multi-frame clock as a counter that increments in link clock counts. The local multi-frame clock counter is equal to (F × K/4) in link clock as units. The rising edge of *SYSREF* resets the local multi-frame clock counter to 0. There are two CSR bits that controls *SYSREF* sampling.

- `csr_sysref_singledet`—resets the local multi-frame clock counter once and automatically cleared after *SYSREF* is sampled. This register also prevents CGS exit to bypass *SYSREF* sampling.
- `csr_sysref_alwayson`—resets the local multi-frame clock counter at every rising edge of *SYSREF* that it detects. This register also enables the *SYSREF* period checker. If the provided *SYSREF* period violates the F and K parameter, an interrupt is triggered. However, this register does not prevent *CGS-SYSREF* race condition.

The following conditions occur if both CSR bits are set:

- resets the local multi-frame clock counter at every rising edge of *SYSREF*.
- prevents *CGS-SYSREF* race condition.
- checks *SYSREF* period.

Related Information

Clock Correlation on page 4-23

Clock Correlation

This section describes the clock correlation between the device clock, link clock, frame clock, and local multi-frame clock.

Example 1

Targeted device with LMF=222, K=16 and Data rate = 6.5 Gbps

Device Clock selected = 325 MHz (obtained during IP core generation)

Link Clock = 6.5 GHz/40 = 162.5 MHz Frame Clock = 6.5 GHz/(10x2) = 325 MHz

Local Multi-frame clock = 325 MHz / 16 = 20.3125 MHz

SYSREF Frequency = Local Multi-frame Clock / n; (n = integer; 1, 2, …)

Local multi-frame clock counter = (F × K/4) = (2×16/4) = 8 link clocks [15]

Example 2

Targeted device with LMF=244, K=16 and Data rate = 5.0 Gbps

Device Clock selected = 125 MHz (obtained during IP core generation)

Link Clock = 5 GHz/40 = 125 MHz [16]

Frame Clock = 5 GHz /(10×4) = 125 MHz [16]

Local Multi-frame clock = 125 MHz / 16 = 7.8125 MHz

SYSREF Frequency = Local Multi-frame Clock / n; (n = integer; 1, 2, …)

Local multi-frame clock counter = (F × K/4) = (4×8/4) = 8 link clocks [15]

Example 3

Targeted device with LMF=421, K=32 and Data rate = 10.0 Gbps

Device Clock selected = 250 MHz (obtained during IP core generation)

Link Clock = 10 GHz/40 = 250 MHz

Frame Clock = 10 GHz/(10×1) = 1 GHz [17]

Local Multi-frame clock = 1 GHz / 32 = 31.25 MHz

SYSREF Frequency = Local Multi-frame Clock / n; (n = integer; 1, 2, …)

Local multi-frame clock counter = (F × K/4) = (1×32/4) = 8 link clocks [15]

---

[15] Eight link clocks means that the local multi-frame clock counts from value 0 to 7 and then loopback to 0.

[16] The link clock and frame clock are running at the same frequency. You only need to generate one clock from the Altera PLL or Altera IO PLL IP core.

[17] For this example, the frame clock may not be able to run up to 1 GHz in the FPGA fabric. The JESD204B transport layer in the design example supports running the data stream of half rate (1 GHz/2 = 500 MHz), at two times the data bus width or of quarter rate (1GHz/4 = 250MHz), at four times the data bus width.

Related Information

- **Device Clock** on page 4-20
- **Link Clock** on page 4-21
- **Local Multi-Frame Clock** on page 4-22

Reset Scheme

All resets in the JESD204B IP core are synchronous reset signals and should be asserted and deasserted synchronously.

Note: Ensure that the resets are synchronized to the respective clocks for reset assertion and deassertion.

Table 4-4: JESD204B IP Core Resets

Reset Signal	Associated Clock	Description
`txlink_rst_n` `rxlink_rst_n`	TX/RX Link Clock	Active low reset controlled by the clock and reset unit. Altera recommends that you: • Assert the `txlink_rst_n`/`rxlink_rst_n` and `txframe_rst_n` /`rxframe_rst_n` signals when the transceiver is in reset. • Deassert the `txlink_rst_n` and `txframe_rst_n` signals after the Altera PLL IP core is locked and the `tx_ready[]` signal from the Transceiver Reset Controller is asserted. • Deassert the `rxlink_rst_n` and `rxframe_rst_n` signals after the Transceiver CDR `rx_islockedtodata[]` signal and `rx_ready[]` signal from the Transceiver Reset Controller are asserted. The `txlink_rst_n`/`rxlink_rst_n` and `txframe_rst_n` /`rxframe_rst_n` signals can be deasserted at the same time. These resets can only be deasserted after you configure the CSR registers.
`txframe_rst_n` `rxframe_rst_n`	TX/RX Frame Clock	Active low reset controlled by the clock and reset unit. If the TX/RX link clock and the TX/RX frame clock has the same frequency, both can share the same reset.

Reset Signal	Associated Clock	Description
tx_analogreset[L-1:0] rx_analogreset[L-1:0]	Transceiver Native PHY Analog Reset	Active high reset controlled by the transceiver reset controller. This signal resets the TX/RX PMA. The link clock, frame clock, and AVS clock reset signals (txlink_rst_n/rxlink_rst_n, txframe_rst_n/rxframe_rst_n and jesd204_tx_avs_rst_n/jesd204_rx_avs_rst_n) can only be deasserted after the transceiver comes out of reset. (18)
tx_digitalreset[L-1:0] rx_digitalreset[L-1:0]	Transceiver Native PHY Digital Reset	Active high reset controlled by the transceiver reset controller. This signal resets the TX/RX PCS. The link clock, frame clock, and AVS clock reset signals (txlink_rst_n/rxlink_rst_n, txframe_rst_n/rxframe_rst_n and jesd204_tx_avs_rst_n/jesd204_rx_avs_rst_n) can only be deasserted after the transceiver comes out of reset. (18)
jesd204_tx_avs_rst_n jesd204_rx_avs_rst_n	TX/RX AVS (CSR) Clock	Active low reset controlled by the clock and reset unit. Typically, both signals can be deasserted after the core PLL and transceiver PLL are locked and out of reset. If you want to dynamically modify the LMF at run-time, you can program the CSRs after AVS reset is deasserted. This phase is referred to as the configuration phase. After the configuration phase is complete, then only the txlink_rst_n/rxlink_rst_n and txframe_rst_n/rxframe_rst_n signals can be deasserted.

Related Information

Altera Transceiver PHY IP Core User Guide

(18) Refer to the *Altera Transceiver PHY IP Core User Guide* for the timing diagram of the tx_analogreset, rx_analogreset, tx_digitalreset, and rx_digitalreset signals.

Signals

The JESD204B IP core signals are listed by interface:

- Transmitter
- Receiver

Note: You should terminate any unused signals.

Transmitter

Table 4-5: Transmitter Signals

Signal	Width	Direction	Description
Clocks and Resets			
pll_ref_clk	1	Input	Transceiver reference clock signal. The reference clock selection depends on the FPGA device family and data rate. This signal is only applicable for V series FPGA variants.
txlink_clk	1	Input	TX link clock signal. This clock is equal to the TX data rate divided by 40. This clock must have the same frequency as the txphy_clk signal but can be differential in phase due to a different clock network. For Subclass 1, you cannot use the output of txphy_clk signal as txlink_clk signal . To sample *SYSREF* correctly, the core PLL must provide the txlink_clk signal and must be configured as normal operating mode.
txlink_rst_n_reset_n	1	Input	Reset for the TX link clock signal. This reset is an active low signal.
txphy_clk[]	L	Output	TX parallel clock output for the TX PCS. This clock must have the same frequency as txlink_clk signal. This clock is output as an optional port for user if the txlink_clk and txframe_clk signals are operating at the same frequency in Subclass 0 operating mode.

Signal	Width	Direction	Description
tx_digitalreset[] (19)	L	Input	Reset for the transceiver PCS block. This reset is an active high signal.
tx_analogreset[] (19)	L	Input	Reset for the transceiver PMA block. This reset is an active high signal.
pll_locked[] (19)	L	Output	PLL locked signal for the hard transceiver. This signal is asserted to indicate that the TX transceiver PLL is locked. This signal is an output signal for V series FPGA variants but an input signal for 10 series FPGA variants and above.
tx_cal_busy[] (19)	L	Output	TX calibration in progress signal. This signal is asserted to indicate that the TX transceiver calibration is in progress.
pll_powerdown[] (19)	• 1 if bonding mode = "xN" • L if bonding mode = feedback_ compensation	Input	TX transceiver PLL power down signal. This signal is only applicable for V series FPGA variants.
tx_bonding_clocks (Single Channel) tx_bonding_clocks_ ch<0..L-1> (Multiple Channels)	6	Input	The transceiver PLL bonding clocks. The transceiver PLL generation provides these clocks. This signal is only available if you select *Bonded* mode for Arria 10 FPGA variants.
tx_serial_clk0 (Single Channel) tx_serial_clk0_ ch<0..L-1> (Multiple Channels)	1	Input	The transceiver PLL serial clock. This is the serializer clock in the PMA. The transceiver PLL generation provides these clocks. This signal is only available if you select *Non-bonded* mode for Arria 10 FPGA variants.
Signal	**Width**	**Direction**	**Description**
Transceiver Interface			
tx_serial_data[]	L	Output	Differential high speed serial output data. The clock is embedded in the serial data stream.

(19) The Transceiver PHY Reset Controller IP Core controls this signal.

Signal	Width	Direction	Description
reconfig_to_xcvr[]	• (L+1)*70 if bonding mode = "xN" • L*140 if bonding mode = feedback compensation	Input	Reconfiguration signals from the Transceiver Reconfiguration Controller IP core to the PHY device. This signal is only applicable for V series FPGA variants. You must connect these signals to the Transceiver Reconfiguration Controller IP core regardless of whether run-time reconfiguration is enabled or disabled. The Transceiver Reconfiguration Controller IP core also supports various calibration function during transceiver power up.
reconfig_from_xcvr[]	• (L+1)*46 if bonding mode = "xN" • L*92 if bonding mode = feedback compensation	Output	Reconfiguration signals to the Transceiver Reconfiguration Controller IP core. This signal is only applicable for V series FPGA variants. You must connect these signals to the Transceiver Reconfiguration Controller IP core regardless of whether run-time reconfiguration is enabled or disabled. The Transceiver Reconfiguration Controller IP core also supports various calibration function during transceiver power up.
reconfig_clk	1	Input	The Avalon-MM clock input. The frequency range is 100–125 MHz. This signal is only available if you enable dynamic reconfiguration for Arria 10 FPGA variants.
reconfig_reset	1	Input	Reset signal for the Transceiver Reconfiguration Controller IP core. This signal is active high and level sensitive. This signal is only available if you enable dynamic reconfiguration for Arria 10 FPGA variants.
reconfig_avmm_address[]	$\log_2$L*1024	Input	The Avalon-MM address. This signal is only available if you enable dynamic reconfiguration for Arria 10 FPGA variants.

Signal	Width	Direction	Description
reconfig_avmm_writedata[]	32	Input	The input data. This signal is only available if you enable dynamic reconfiguration for Arria 10 FPGA variants.
reconfig_avmm_readdata[]	32	Output	The output data. This signal is only available if you enable dynamic reconfiguration for Arria 10 FPGA variants.
reconfig_avmm_write	1	Input	Write signal. This signal is active high. This signal is only available if you enable dynamic reconfiguration for Arria 10 FPGA variants.
reconfig_avmm_read	1	Input	Read signal. This signal is active high. This signal is only available if you enable dynamic reconfiguration for Arria 10 FPGA variants.
reconfig_avmm_waitrequest	1	Output	Wait request signal. This signal is only available if you enable dynamic reconfiguration for Arria 10 FPGA variants.
Signal	**Width**	**Direction**	**Description**
Avalon-ST Interface			
jesd204_tx_link_data[]	L*32	Input	Indicates a 32-bit user data at txlink_clk clock rate, where four octets are packed into a 32-bit data width per lane. The data format is big endian. The first octet is located at bit[31:24], followed by bit[23:16], bit[15:8], and the last octet is bit[7:0]. Lane 0 data is always located in the lower 32-bit data. If more than one lane is instantiated, lane 1 is located at bit[63:32], with the first octet position at bit[63:56].

Signal	Width	Direction	Description
jesd204_tx_link_valid	1	Input	Indicates whether the data from the transport layer is valid or invalid. The Avalon-ST sink interface in the TX core cannot be backpressured and assumes that data is always valid on every cycle when the jesd204_tx_link_ready signal is asserted. • 0—data is invalid • 1—data is valid
jesd204_tx_link_ready	1	Output	Indicates that the Avalon-ST sink interface in the TX core is ready to accept data. The Avalon-ST sink interface asserts this signal on the JESD204B link state of *USER_DATA* phase. The ready latency is 0.
jesd204_tx_frame_ready	1	Output	Indicates that the Avalon-ST sink interface in the transport layer is ready to accept data. The Avalon-ST sink interface asserts this signal on the JESD204B link state of ILAS 4th multiframe and also the *USER_DATA* phase. The ready latency is 0.
Signal	**Width**	**Direction**	**Description**
Avalon-MM Interface			
jesd204_tx_avs_clk	1	Input	The Avalon-MM interface clock signal. This clock is asynchronous to all the functional clocks in the JESD204B IP core. The JESD204B IP core can handle any cross clock ratio and therefore the clock frequency can range from 75 MHz to 125 MHz.
jesd204_tx_avs_rst_n	1	Input	This reset is associated with the jesd204_tx_avs_clk signal. This reset is an active low signal. You can assert this reset signal asynchronously but must deassert it synchronously to the jesd204_tx_avs_clk signal. After you deassert this signal, the CPU can configure the CSRs.

Signal	Width	Direction	Description
jesd204_tx_avs_chipselect	1	Input	When this signal is present, the slave port ignores all Avalon-MM signals unless this signal is asserted. This signal must be used in combination with read or write. If the Avalon-MM bus does not support chip select, you are recommended to tie this port to 1.
jesd204_tx_avs_address[]	8	Input	For Avalon-MM slave, the interconnect translates the byte address into a word address in the address space so that each slave access is for a word of data. For example, address = 0 selects the first word of the slave and address = 1 selects the second word of the slave.
jesd204_tx_avs_writedata[]	32	Input	32-bit data for write transfers. The width of this signal and the jesd204_tx_avs_readdata[31:0] signal must be the same if both signals are present
jesd204_tx_avs_read	1	Input	This signal is asserted to indicate a read transfer. This is an active high signal and requires the jesd204_tx_avs_readdata[31:0] signal to be in use.
jesd204_tx_avs_write	1	Input	This signal is asserted to indicate a write transfer. This is an active high signal and requires the jesd204_tx_avs_writedata[31:0] signal to be in use.
jesd204_tx_avs_readdata[]	32	Output	32-bit data driven from the Avalon-MM slave to master in response to a read transfer.
jesd204_tx_avs_waitrequest	1	Output	This signal is asserted by the Avalon-MM slave to indicate that it is unable to respond to a read or write request. The JESD204B IP core ties this signal to 0 to return the data in the access cycle.
Signal	**Width**	**Direction**	**Description**
JESD204 Interface			
sysref	1	Input	*SYSREF* signal for JESD204B Subclass 1 implementation. For Subclass 0 and Subclass 2 mode, tie-off this signal to 0.

Signal	Width	Direction	Description
sync_n	1	Input	Indicates SYNC_N from the converter device or receiver. This is an active low signal and is asserted 0 to indicate a synchronization request or error reporting from the converter device. To indicate a synchronization request, the converter device must assert this signal for at least five frames and nine octets. To indicate an error reporting, the converter device must ensure that the pulse is at least one cycle of the txlink_clk signal or two cycles of the txframe_clk signal (whichever period is longer).
dev_sync_n	1	Output	Indicates a clean synchronization request. This is an active low signal and is asserted 0 to indicate a synchronization request only. The sync_n signal error reporting is being masked out of this signal. This signal is also asserted during software-initiated synchronization.
mdev_sync_n	1	Input	Indicates a multidevice synchronization request. Synchronize signal combination should be done externally and then input to the JESD204B IP core through this signal. • For subclass 0—combine the dev_sync_n signal from all multipoint links before connecting to the mdev_sync_n signal. • For subclass 1—connect the dev_sync_n signal to the mdev_sync_n signal for each link respectively. In a single link instance where multidevice synchronization is not needed, tie the dev_sync_n signal to this signal.
Signal	**Width**	**Direction**	**Description**
CSR			
jesd204_tx_frame_error	1	Input	Optional signal to indicate an empty data stream due to invalid data. This signal is asserted high to indicate an error during data transfer from the transport layer to the TX core.

Signal	Width	Direction	Description
csr_l[]	5	Output	Indicates the number of active lanes for the link. The transport layer can use this signal as a run-time parameter.
csr_f[]	8	Output	Indicates the number of octets per frame. The transport layer can use this signal as a run-time parameter.
csr_k[]	5	Output	Indicates the number of frames per multiframe. The transport layer can use this signal as a run-time parameter.
csr_m[]	8	Output	Indicates the number of converters for the link. The transport layer can use this signal as a run-time parameter.
csr_cs[]	2	Output	Indicates the number of control bits per sample. The transport layer can use this signal as a run-time parameter.
csr_n[]	5	Output	Indicates the converter resolution. The transport layer can use this signal as a run-time parameter.
csr_np[]	5	Output	Indicates the total number of bits per sample. The transport layer can use this signal as a run-time parameter.
csr_s[]	5	Output	Indicates the number of samples per converter per frame cycle. The transport layer can use this signal as a run-time parameter.
csr_hd	1	Output	Indicates the high density data format. The transport layer can use this signal as a run-time parameter.
csr_cf[]	5	Output	Indicates the number of control words per frame clock period per link. The transport layer can use this signal as a run-time parameter.
csr_lane_powerdown[]	L	Output	Indicates which lane is powered down. You need to set this signal if you have configured the link and want to reduce the number of active lanes.

Signal	Width	Direction	Description
csr_tx_testmode[]	4	Output	Indicates the address space that is reserved for DLL testing within the JESD204B IP core. • 0—reserved for the IP core. • 1—program different tests in the transport layer. Refer to csr_tx_testmode register.
csr_tx_testpattern_a[]	32	Output	A 32-bit fixed data pattern for the test mode. (20)
csr_tx_testpattern_b[]	32	Output	A 32-bit fixed data pattern for the test mode. (20)
csr_tx_testpattern_c[]	32	Output	A 32-bit fixed data pattern for the test mode. (20)
csr_tx_testpattern_d[]	32	Output	A 32-bit fixed data pattern for the test mode. (20)
Signal	**Width**	**Direction**	**Description**
Out-of-band (OOB)			
jesd204_tx_int	1	Output	Interrupt pin for the JESD204B IP core. Interrupt is asserted when any error or synchronization request is detected. Configure the tx_err_enable register to set what type of errors that can trigger an interrupt.
Signal	**Width**	**Direction**	**Description**
Debug or Testing			
jesd204_tx_dlb_data[]	L*32	Output	Optional signal for parallel data from the DLL in TX to RX loopback testing. (21)
jesd204_tx_dlb_kchar_data[]	L*4	Output	Optional signal to indicate the K character value for each byte in TX to RX loopback testing. (21)

(20) You can use this signal in the transport layer to configure programmable test pattern.
(21) This signal is only for internal testing purposes. You can leave this signal disconnected.

Receiver

Table 4-6: Receiver Signals

Signal	Width	Direction	Description
Clocks and Resets			
pll_ref_clk	1	Input	Transceiver reference clock signal.
rxlink_clk	1	Input	RX link clock signal used by the Avalon-ST interface. This clock is equal to RX data rate divided by 40. For Subclass 1, you cannot use the output of rxphy_clk signal as rxlink_clk signal. To sample *SYSREF* correctly, the core PLL must provide the rxlink_clk signal and must be configured as normal operating mode.
rxlink_rst_n_reset_n	1	Input	Reset for the RX link clock signal. This reset is an active low signal.
rxphy_clk[]	L	Output	Recovered clock signal. This clock is derived from the clock data recovery (CDR) and the frequency depends on the JESD204B IP core data rate.
rx_digitalreset[] (22)	L	Input	Reset for the transceiver PCS block. This reset is an active high signal.
rx_analogreset[] (22)	L	Input	Reset for the CDR and transceiver PMA block. This reset is an active high signal.
rx_islockedtodata[] (22)	L	Output	This signal is asserted to indicate that the RX CDR PLL is locked to the RX data and the RX CDR has changed from LTR to LTD mode.
rx_cal_busy[] (22)	L	Output	RX calibration in progress signal. This signal is asserted to indicate that the RX transceiver calibration is in progress.
Signal	**Width**	**Direction**	**Description**
Transceiver Interface			
rx_serial_data[]	L	Input	Differential high speed serial input data. The clock is recovered from the serial data stream.

(22) The Transceiver PHY Reset Controller IP Core controls this signal.

Signal	Width	Direction	Description
reconfig_to_xcvr[]	L*70	Input	Dynamic reconfiguration input for the hard transceiver. This signal is only applicable for V series FPGA variants. You must connect these signals to the Transceiver Reconfiguration Controller IP core regardless of whether run-time reconfiguration is enabled or disabled. The Transceiver Reconfiguration Controller IP core also supports various calibration function during transceiver power up.
reconfig_from_xcvr[]	L*46	Output	Dynamic reconfiguration output for the hard transceiver. This signal is only applicable for V series FPGA variants. You must connect these signals to the Transceiver Reconfiguration Controller IP core regardless of whether run-time reconfiguration is enabled or disabled. The Transceiver Reconfiguration Controller IP core also supports various calibration function during transceiver power up.
reconfig_clk	1	Input	The Avalon-MM clock input. The frequency range is 100–125 MHz. This signal is only available if you enable dynamic reconfiguration for Arria 10 FPGA variants.
reconfig_reset	1	Input	Reset signal for the Transceiver Reconfiguration Controller IP core. This signal is active high and level sensitive. This signal is only available if you enable dynamic reconfiguration for Arria 10 FPGA variants.
reconfig_avmm_address[]	$\log_2$L*1024	Input	The Avalon-MM address. This signal is only available if you enable dynamic reconfiguration for Arria 10 FPGA variants.
reconfig_avmm_writedata[]	32	Input	The input data. This signal is only available if you enable dynamic reconfiguration for Arria 10 FPGA variants.

Signal	Width	Direction	Description
reconfig_avmm_readdata[]	32	Output	The output data. This signal is only available if you enable dynamic reconfiguration for Arria 10 FPGA variants.
reconfig_avmm_write	1	Input	Write signal. This signal is active high. This signal is only available if you enable dynamic reconfiguration for Arria 10 FPGA variants.
reconfig_avmm_read	1	Input	Read signal. This signal is active high. This signal is only available if you enable dynamic reconfiguration for Arria 10 FPGA variants.
reconfig_avmm_waitrequest	1	Output	Wait request signal. This signal is only available if you enable dynamic reconfiguration for Arria 10 FPGA variants.
Signal	**Width**	**Direction**	**Description**
Avalon-ST Interface			
jesd204_rx_link_data[]	L*32	Output	Indicates a 32-bit data from the DLL to the transport layer. The data format is big endian, where the earliest octet is placed in bit [31:24] and the latest octet is placed in bit [7:0].
jesd204_rx_link_valid	1	Output	Indicates whether the data to the transport layer is valid or invalid. The Avalon-ST source interface in the RX core cannot be backpressured and will transmit the data when the jesd204_rx_data_valid signal is asserted. • 0—data is invalid • 1—data is valid
jesd204_rx_link_ready	1	Input	Indicates that the Avalon-ST sink interface in the transport layer is ready to receive data.
jesd204_rx_frame_error	1	Input	Indicates an empty data stream due to invalid data. This signal is asserted high to indicate an error during data transfer from the RX core to the transport layer.
Signal	**Width**	**Direction**	**Description**
Avalon-MM Interface			

Signal	Width	Direction	Description
jesd204_rx_avs_clk	1	Input	The Avalon-MM interface clock signal. This clock is asynchronous to all the functional clocks in the JESD204B IP core. The JESD204B IP core can handle any cross clock ratio and therefore the clock frequency can range from 75 MHz to 125 MHz.
jesd204_rx_avs_rst_n	1	Input	This reset is associated with the jesd204_rx_avs_clk signal. This reset is an active low signal. You can assert this reset signal asynchronously but must deassert it synchronously to the jesd204_rx_avs_clk signal. After you deassert this signal, the CPU can configure the CSRs.
jesd204_rx_avs_chipselect	1	Input	When this signal is present, the slave port ignores all Avalon-MM signals unless this signal is asserted. This signal must be used in combination with read or write. If the Avalon-MM bus does not support chip select, you are recommended to tie this port to 1.
jesd204_rx_avs_address[]	8	Input	For Avalon-MM slave, the interconnect translates the byte address into a word address in the address space so that each slave access is for a word of data. For example, address = 0 selects the first word of the slave and address = 1 selects the second word of the slave.
jesd204_rx_avs_writedata[]	32	Input	32-bit data for write transfers. The width of this signal and the jesd204_rx_avs_readdata[31:0] signal must be the same if both signals are present.
jesd204_rx_avs_read	1	Input	This signal is asserted to indicate a read transfer. This is an active high signal and requires the jesd204_rx_avs_readdata[31:0] signal to be in use.
jesd204_rx_avs_write	1	Input	This signal is asserted to indicate a write transfer. This is an active high signal and requires the jesd204_rx_avs_writedata[31:0] signal to be in use.
jesd204_rx_avs_readdata[]	32	Output	32-bit data driven from the Avalon-MM slave to master in response to a read transfer.

Signal	Width	Direction	Description
jesd204_rx_avs_waitrequest	1	Output	This signal is asserted by the Avalon-MM slave to indicate that it is unable to respond to a read or write request. The JESD204B IP core ties this signal to 0 to return the data in the access cycle.
Signal	**Width**	**Direction**	**Description**
JESD204 Interface			
sysref	1	Input	SYSREF signal for JESD204B Subclass 1 implementation. For Subclass 0 and Subclass 2 mode, tie-off this signal to 0.
dev_sync_n	1	Output	Indicates a *SYNC~* from the receiver. This is an active low signal and is asserted 0 to indicate a synchronization request. Instead of reporting the link error through this signal, the JESD204B IP core uses the jesd204_rx_int signal to interrupt the CPU.
sof[]	4	Output	Indicates a start of frame. • [3]—start of frame for jesd204_rx_link_data[31:24] • [2]—start of frame for jesd204_rx_link_data[23:16] • [1]—start of frame for jesd204_rx_link_data[15:8] • [0]—start of frame for jesd204_rx_link_data[7:0]
somf[]	4	Output	Indicates a start of multiframe. • [3]—start of multiframe for jesd204_rx_link_data[31:24] • [2]—start of multiframe for jesd204_rx_link_data[23:16] • [1]—start of multiframe for jesd204_rx_link_data[15:8] • [0]—start of multiframe for jesd204_rx_link_data[7:0]
dev_lane_aligned	1	Output	Indicates that all lanes for this device are aligned.

Signal	Width	Direction	Description
alldev_lane_aligned	1	Input	Aligns all lanes for this device. For multidevice synchronization, multiplex all the dev_lane_aligned signals before connecting to this signal pin. For single device support, connect the dev_lane_aligned signal back to this signal.
Signal	**Width**	**Direction**	**Description**
CSR			
csr_l[]	5	Output	Indicates the number of active lanes for the link. The transport layer can use this signal as a run-time parameter.
csr_f[]	8	Output	Indicates the number of octets per frame. The transport layer can use this signal as a run-time parameter.
csr_k[]	5	Output	Indicates the number of frames per multiframe. The transport layer can use this signal as a run-time parameter.
csr_m[]	8	Output	Indicates the number of converters for the link. The transport layer can use this signal as a run-time parameter.
csr_cs[]	2	Output	Indicates the number of control bits per sample. The transport layer can use this signal as a run-time parameter.
csr_n[]	5	Output	Indicates the converter resolution. The transport layer can use this signal as a run-time parameter.
csr_np[]	5	Output	Indicates the total number of bits per sample. The transport layer can use this signal as a run-time parameter.
csr_s[]	5	Output	Indicates the number of samples per converter per frame cycle. The transport layer can use this signal as a run-time parameter.
csr_hd	1	Output	Indicates the high density data format. The transport layer can use this signal as a run-time parameter.

Signal	Width	Direction	Description
csr_cf[]	5	Output	Indicates the number of control words per frame clock period per link. The transport layer can use this signal as a run-time parameter.
csr_lane_powerdown[]	L	Output	Indicates which lane is powered down. You need to set this signal if you have configured the link and want to reduce the number of active lanes.
csr_rx_testmode[]	4	Output	Indicates the address space that is reserved for DLL testing within the JESD204B IP core. • 0—reserved for the IP core. • 1—program different tests in the transport layer. Refer to the csr_rx_testmoderegister.
Signal	**Width**	**Direction**	**Description**
Out-of-band (OOB)			
jesd204_rx_int	1	Output	Interrupt pin for the JESD204B IP core. Interrupt is asserted when any error is detected. Configure the rx_err_enable register to set what type of errors that can trigger an interrupt.
Signal	**Width**	**Direction**	**Description**
Debug or Testing			
jesd204_rx_dlb_data[]	L*32	Input	Optional signal for parallel data to the DLL in TX to RX loopback testing. (23)
jesd204_rx_dlb_data_valid[]	L	Input	Optional signal to indicate valid data for each byte in TX to RX loopback testing. (23)
jesd204_rx_dlb_kchar_data[]	L*4	Input	Optional signal to indicate the K character value for each byte in TX to RX loopback testing. (23)
jesd204_rx_dlb_errdetect[]	L*4	Input	Optional signal to indicate 8B/10B error. (23)
jesd204_rx_dlb_disperr[]	L*4	Input	Optional signal to indicate running disparity. (23)

(23) This signal is only for internal testing purposes. Tie this signal to low.

Registers

The JESD204B IP core supports a basic one clock cycle transaction bus. There is no support for burst mode and wait-state feature (the `avs_waitrequest` signal is tied to 0). The JESD204B IP core Avalon-MM slave interface has a data width of 32 bits and is implemented based on word addressing. The Avalon-MM slave interface does not support byte enable access.

Each write transfer has a *writeWaitTime* of 0 cycle while a read transfer has a *readWaitTime* of 1 cycle and *readLatency* of 1 cycle.

The following HTML files list the TX and RX core registers.

- **TX register map**
- **RX register map**

Register Access Type Convention

This table describes the register access type for Altera IP cores.

Table 4-7: Register Access Type and Definition

Access Type	Definition
RO	Software read only (no effect on write). The value is hard-tied internally to either '0' or '1' and does not vary.
RO/v	Software read only (no effect on write). The value may vary.
RC	• Software reads shall return the current bit value, then the bit is self-clear to 0. • Software reads also cause the bit value to be cleared to 0.
RW	• Software reads shall return the current bit value. • Software writes shall set the bit to the desired value.
RW1C	• Software reads shall return the current bit value. • Software writes 0 shall have no effect. • Software writes 1 shall clear the bit to 0, if the bit has been set to 1 by hardware. • Hardware sets the bit to 1. • Software clear has higher priority than hardware set.
RW1S	• Software reads shall return the current bit value. • Software writes 0 shall have no effect. • Software writes 1 shall set the bit to 1. • Hardware clears the bit to 0, if the bit has been set to 1 by software. • Software set has higher priority than hardware clear.

JESD204B IP Core Design Guidelines

5

2014.12.15

UG-01142

Subscribe Send Feedback

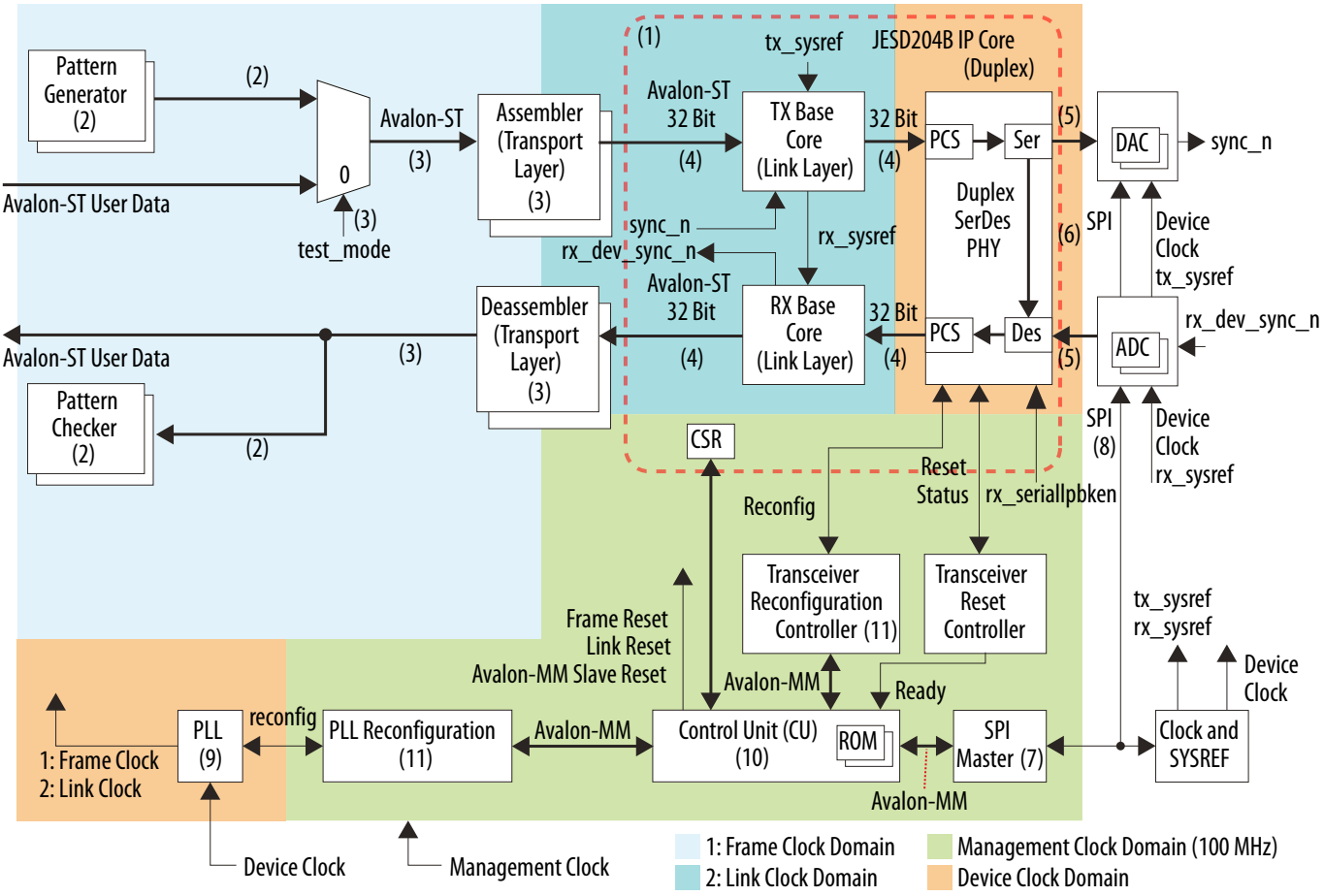
JESD204B IP Core Design Example

The design example entity consists of various components that interface with the JESD204B IP core to demonstrate the following features:

- single or multiple link configuration
- different LMF settings with scrambling and internal serial loopback enabled
- interoperability against diverse converter devices
- dynamic reconfiguration

You can use the synthesizable design example entity in both simulation and hardware environments. **Figure 5-1** illustrates the high level system architecture of the JESD204B IP core design example.

Figure 5-1: Design Example Block Diagram



The list below describes the mechanism of the design example architecture (with reference to the note numbers in the design example block diagram).

1. For multiple links, the JESD204B IP core is instantiated multiple times. For example, in 2x112 (LMF) configuration, two cores are instantiated, where each core is configured at LMF=112. (24)
2. The number of pattern generator or pattern checker instances is equivalent to the parameter value of LINK. The data bus width per instance is equivalent to the value of FRAMECLK_DIV*M*S*N. (24)
3. The number of transport layer instances is equivalent to the parameter value of LINK. The legal value of LINK is 1 and 2. The data bus width per instance is equivalent to the value of FRAMECLK_DIV*M*S*N.(24) The `test_mode` = 0 signal indicates a normal operation mode, where the assembler takes data from the Avalon-ST source. Otherwise, the assembler takes data from the pattern generator.
4. The Avalon-ST interface data bus is fixed at 32-bit. The number of 32-bit data bus is equal to the number of lanes (L).
5. The number of lanes per converter device (L).

(24) Refer to **Figure 5-17** and **Figure 5-18** for the illustration of a single and multiple JESD204B links.

6. You can enable internal serial loopback by setting the `rx_seriallpbken` input signal. You can dynamically toggle this input signal. When toggled to 1, the RX path takes the serial input from the TX path internally in the FPGA. When toggled to 0, the RX path takes the serial input from the external converter device. During internal serial loopback mode, the assembler takes input from the pattern generator.
7. A single serial port interface (SPI) master instance can control multiple SPI slaves. The SPI master is a 4-wire instance. If the SPI slave is a 3-wire instance, use a bidirectional I/O buffer in between the master and slave to interface the 4-wire master to 3-wire slave.
8. The SPI protocol interface. All slaves share the same data lines (MISO and MOSI, or DATAIO). Each slave has its own slave select or chip select line (`SS_n`).
9. The PLL takes the device clock from an external clock chip as the input reference. The PLL generates two output clocks (utilizing two output counters from a single VCO). Clock 1 is the frame clock for the transport layer, pattern generator, and pattern checker. Clock 2 is the link clock for the transport and link layer.
10. The control unit implements a memory initialization file (MIF) method for configuring the SPI. Each MIF corresponds to a separate external converter per device or clock chip. For example, in a system that interacts with both DAC and ADC, two MIFs are needed—one each for DAC and ADC.
11. The PLL reconfiguration and transceiver reconfiguration controller instances are only required for run time reconfiguration of the data rate.

Design Example Components

The design example for the JESD204B IP core consists of the following components:

- PLL
- PLL reconfiguration
- Transceiver reconfiguration controller
- Transceiver reset controller
- Pattern generator
- Pattern checker
- Assembler and deassembler (in the transport layer)
- SPI
- Control unit

The following sections describe in detail the function of each component.

PLL

The design example requires four different clock domains—device clock, management clock, frame clock, and link clock.

Typically, the device clock is generated from an external converter or a clock device while the management clock (AVS clock) is generated from an on-board 100 MHz oscillator.

For instance, if the JESD204B IP core is configured at data rate of 6.144 Gbps, transceiver reference clock frequency of 153.6 MHz, and number of octets per frame (F) = 2, the example below indicates the PLL clock frequencies:

- device clock = transceiver reference clock frequency = 153.6 MHz
- link clock = 6144 / 40 = 153.6 MHz
- frame clock = 153.6 x 32 / (8 x 2) = 307.2 MHz

Related Information

- **Clocking Scheme** on page 4-18
  More information about the JESD204B IP core clocks.

PLL Reconfiguration

The PLL reconfiguration utilizes the ALTERA_PLL_RECONFIG IP core to implement reconfiguration logic to facilitate dynamic real-time reconfiguration of PLLs in Altera devices. You can use this megafunction IP core to update the output clock frequency, PLL bandwidth, and phase shifts in real time, without reconfiguring the entire FPGA.

The design example uses the MIF approach to reconfigure the core PLL. The ALTERA_PLL_RECONFIG IP core has two parameter options—**Enable MIF Streaming** and **Path to MIF file**—for the MIF input. Turn on **Enable MIF Streaming** option and set the *core_pll.mif* as the value to **Path to MIF file** parameter.

The following PLL reconfiguration Avalon-MM operations occurs during data rate reconfiguration.

Table 5-1: PLL Reconfiguration Operation

Operation	Avalon-MM Interface Signal	Byte Address Offset (6bits)	Bit	Value
Arria V and Stratix V Devices				
Set MIF base address	pll_mgmt_*	0x01F	[8:0]	0x000 (maximum configuration) or 0x02E (downscale configuration)
Write to the START register to begin	pll_mgmt_*	0x02	[0:0]	0x01
Arria 10 Devices				
Start MIF streaming with MIF base address specified in data value	pll_mgmt_*	0x010	[31:0]	0x000 (maximum configuration) or 0x02E (downscale configuration) (25)

Related Information

AN 661: Implementing Fractional PLL Reconfiguration with Altera PLL and Altera PLL Reconfig Megafunctions
More information about the MIF streaming option.

Transceiver Reconfiguration Controller

The transceiver reconfiguration controller allows you to change the device transceiver settings at any time. Any portion of the transceiver can be selectively reconfigured. Each portion of the reconfiguration

(25) The MIF base address is 9 bits (LSB). The remaining bits are reserved.

requires a read-modify-write operation (read first, then write), in such a way that it modifies only the appropriate bits in a register and not changing other bits.

In the design example, MIF approach is used to reconfigure the ATX PLL and transceiver channel in the JESD204 IP core via the Transceiver Reconfiguration Controller. The number of reconfiguration interface is determined by number of lanes (L) + number of TX_PLL (different number of TX_PLL for bonded and non-bonded mode). Since the MIF approach reconfiguration for transceiver only supports non-bonded mode, the number of TX_PLL is equal to number of lanes. The number of reconfiguration interface = 2 x number of lanes (L).

The transceiver reconfiguration controller interfaces:

- MIF Reconfiguration Avalon-MM master interface—connects to the MIF ROM.
- Transceiver Reconfiguration interface—connects to the JESD204B IP core, which eventually connects to the native PHY.
- Reconfiguration Management Avalon-MM slave interface—connects to the control unit.

Note: The transceiver reconfiguration controller is only used in Arria V and Stratix V devices. For Arria 10 devices, the control unit directly communicates with the transceiver in the JESD204B IP core through the `reconfig_avmm_*` interface signals.

The following transceiver reconfiguration controller Avalon-MM operations are involved during data rate reconfiguration.

Table 5-2: Transceiver Reconfiguration Controller Operation for Arria V and Stratix V Devices

Operation	Avalon-MM Interface Signal	Byte Address Offset (6bits)	Bit	Value
Write logical channel number	reconfig_mgmt_*	0x38	[9:0]	0
Write MIF mode	reconfig_mgmt_*	0x3A	[3:2]	2'b00
Write 0 to streamer offset register	reconfig_mgmt_*	0x3B	[15:0]	0
Write MIF base address to streamer data register	reconfig_mgmt_*	0X3C	[31:0]	*32'h1000
Initiate a write of all the above data	reconfig_mgmt_*	0x3A	[0]	1'b1
Write 1 to streamer offset register	reconfig_mgmt_*	0x3B	[15:0]	1
Write to streamer data register to set up MIF streaming	reconfig_mgmt_*	0x3C	[31:0]	3
Initiate a write of all the above data to start streaming the MIF	reconfig_mgmt_*	0x3A	[0]	1'b1
Read the busy bit to determine when the write has completed	reconfig_mgmt_*	0x3A	[8]	1: Busy 0: Operation completed

Note: The above steps are repeated for the number of channels and followed by the number of TX_PLLs.

For Arria 10 devices, the only Avalon-MM operation is a direct write to the transceiver register through the `reconfig_avmm_*` interface at the JESD204B IP core. Every line in the MIF is `DPRIO_ADDR[25:16]+ BIT_MASK[15:8]+ DATA[7:0]`. The control unit maps the `DPRIO_ADDR` to `reconfig_avmm_address` and `BIT_MASK '&' DATA` to `reconfig_avmm_data`.

Related Information

Altera Transceiver PHY IP Core User Guide
More information about the transceiver reconfiguration controller.

Transceiver Reset Controller

The transceiver reset controller uses the Altera's Transceiver PHY Reset Controller IP Core to ensure a reliable initialization of the transceiver. The reset controller has separate reset controls per channel to handle synchronization of reset inputs, hysteresis of PLL locked status, and automatic or manual reset recovery mode.

In this design example, the reset controller targets both the TX and RX channels. The **TX PLL**, **TX Channel**, and **RX Channel** parameters are programmable to accommodate single and multiple (2) JESD204B links.

Related Information

- **Altera Transceiver PHY IP Core User Guide**
  More information about the Transceiver PHY Reset Controller IP Core.
- **Arria V Device Handbook, Volume 2: Transceivers**
  More information about the device usage mode.

Pattern Generator

The pattern generator instantiates any supported generators and has an output multiplexer to select which generated pattern to forward to the transport layer based on the test mode during run time. Additionally, the pattern generator also supports run-time reconfiguration (downscale) on the number of converters per device (M) & samples per converter per frame (S).

The pattern generator can be a parallel PRBS, alternate checkerboard, or ramp wave generator. The data output bus width of the pattern generator is equivalent to the value of FRAMECLK_DIV × M × S × N.

The pattern generator includes a *REVERSE_DATA* parameter to control data arrangement at the output. The default value of this parameter is 0.

- 0—no data rearrangement at the output of the generator.
- 1—data rearrangement at the output of the generator.

For example, when M=2, S=1, N=16, F1/F2_FRAMECLK_DIV=1, the input or output data width equals to [31:0], with the following data arrangement:

0: {m1s0[31:16], m0s0[15:0]}
1: {m0s0[31:16], m1s0[15:0]}

Parallel PRBS Generator

PRBS generator circuits often consists of simple shift registers with feedback that serve as test sources for serial data links. The output sequence is not truly random but repeats after 2^X–1 bits, where X denotes the

length of the shift register. Polynomial notation—which the polynomial order corresponds to the length of the shift register and the period of PRBS—provides a method of describing the sequence.

Alternate Checkerboard Generator

The alternate checkerboard generator circuit consists of simple flip registers that serve as test sources for serial data links.

The output sequence of subsequent N-bits sample is generated by inverting the previous N-bits (counting from LSB to MSB) of the same data pattern at that clock cycle. The first N-bits sample from LSB of the data pattern on next clock cycle is generated by inverting the last N-bits sample on the MSB of the data pattern on current clock cycle.

Ramp Wave Generator

The ramp wave generator circuit consists of a simple register and adders that serve as test sources for serial data links.

The output sequence of subsequent N-bits sample is an increment by one of the previous N-bits sample (counting from LSB to MSB) in the same data pattern at that clock cycle. The first N-bits sample from LSB of the data pattern on next clock cycle is generated by an increment by one of the last N-bits sample on the MSB of the data pattern on current clock cycle.

Pattern Checker

The pattern checker instantiates any supported checkers and support run time reconfiguration (downscale) of the number of converters per device (M) and samples per converter per frame (S).

The pattern checker can be either a parallel PRBS checker, alternate checkerboard checker, or ramp wave checker. The data input bus width of the pattern checker is equivalent to the value of FRAMECLK_DIV × M × S × N.

The pattern checker includes an *ERR_THRESHOLD* parameter to control the number of error tolerance allowed in the checker. The default value of this parameter is 1.

The pattern checker also includes a *REVERSE_DATA* parameter to control data arrangement at the input. The default value of this parameter is 0.

- 0—no data rearrangement at the input of the checker.
- 1—data rearrangement at the input of the checker.

Parallel PRBS Checker

The PRBS checker contains the same polynomial as in the PRBS generator. The polynomial is only updated when the enable signal is active, which indicates that the input data is valid. The feedback path is XOR'ed with the input data to do a comparison. The checker flags an error when it finds any single mismatch between polynomial data and input data.

Alternate Checkerboard Checker

The alternate checkerboard checker is implemented in the same way as in the alternate checkerboard generator. To do a comparison, an initial seed internally generates a set of expected data pattern result to XOR'ed with the input data. The seed is updated only when the enable signal is active, which indicates

that the input data is valid. The checker flags an error when it finds any single mismatch between the expected data and input data.

Ramp Wave Checker

The ramp wave checker is implemented in the same way as in the ramp wave generator. To do a comparison, an initial seed internally generates a set of expected data pattern result to XOR'ed with the input data. The seed is updated only when the enable signal is active, which indicates that the input data is valid. The checker flags an error when it finds any single mismatch between the expected data and input data.

Transport Layer

The transport layer in the JESD204B IP core consists of an assembler at the TX path and a deassembler at the RX path.

The transport layer provides the following services to the application layer (AL) and the DLL:

- The assembler at the TX path:
  - maps the conversion samples from the AL (through the Avalon-ST interface) to a specific format of non-scrambled octets, before streaming them to the DLL.
  - reports AL error to the DLL if it encounters a specific error condition on the Avalon-ST interface during TX data streaming.
- The deassembler at the RX path:
  - maps the descrambled octets from the DLL to a specific conversion sample format before streaming them to the AL (through the Avalon-ST interface).
  - reports AL error to the DLL if it encounters a specific error condition on the Avalon-ST interface during RX data streaming.

Supported System Configuration

The transport layer supports static configurations where before compilation, you can modify the configurations using the IP core's parameter editor in the Quartus II software. To change to another configuration, you have to recompile the design. The following list describes the supported configurations for the transport layer:

- Data rate (maximum) = 12.5 Gbps (`F1_FRAMECLK_DIV` = 4 and `F2_FRAMECLK_DIV` = 2)
- L = 1–8
- F = 1, 2, 4, 8
- N = 12, 13, 14, 15, 16
- N' = 16
- CS = 0–3
- CF = 0
- HD = 0

Dynamic Downscaling Of System Parameters (L, N, and F)

The Dynamic Downscaling of System Parameters (DDSP) feature enables you to dynamically downscale specific JESD204B system parameters through the CSR, without having to recompile the FPGA.

The transport layer supports dynamic downscaling of parameters L, F, and N only. The supported M and S parameters are determined by the L, F, and N' parameters. Some parameters (for example, CS and N')

do not have this capability in the transport layer. If you needs to change any of these parameters, you must recompile the system.

You are advised to connect the power down channels to higher indexes and connect used channel at lower lanes. Otherwise, you have to reroute the physical-used channels to lower lanes externally when connecting the IP core to the transport layer. For example, when L = 4 and csr_l = 8'd1 (which means two lanes out of four lanes are active), with lane 1 and lane 3 being powered down, connection from the MAC to the transport layer for lane 0 remains. However, lane 1 is powered down while lane 2 is not powered down. Thus, lane 2 output from the MAC should be rerouted to lane 1 data input of the transport layer. The data port for those power-down channels will be tied off within the transport layer.

The 16-bit N' data for F = 1 is formed through the data from 2 lanes. Thus, F = 1 is not supported for odd number of lanes, for example, when LMF = 128. In this case, you can only reconfigure from F = 8 to F = 4 and F = 2 but not F = 1.

Relationship Between Frame Clock and Link Clock

The frame clock and link clock are synchronous.

The ratio of `link_clk` period to `frame_clk` period is given by this formula:

32 x L / M x S x N'

Table 5-3: txframe_clk and rxframe_clk Frequency for Different F Parameter Settings

For a given f_{txlink} (txlink_clk frequency) and f_{rxlink} (rxlink_clk frequency), the $f_{txframe}$ (txframe_clk frequency) and $f_{rxframe}$ (rxframe_clk frequency) are derived from the formula listed in this table.

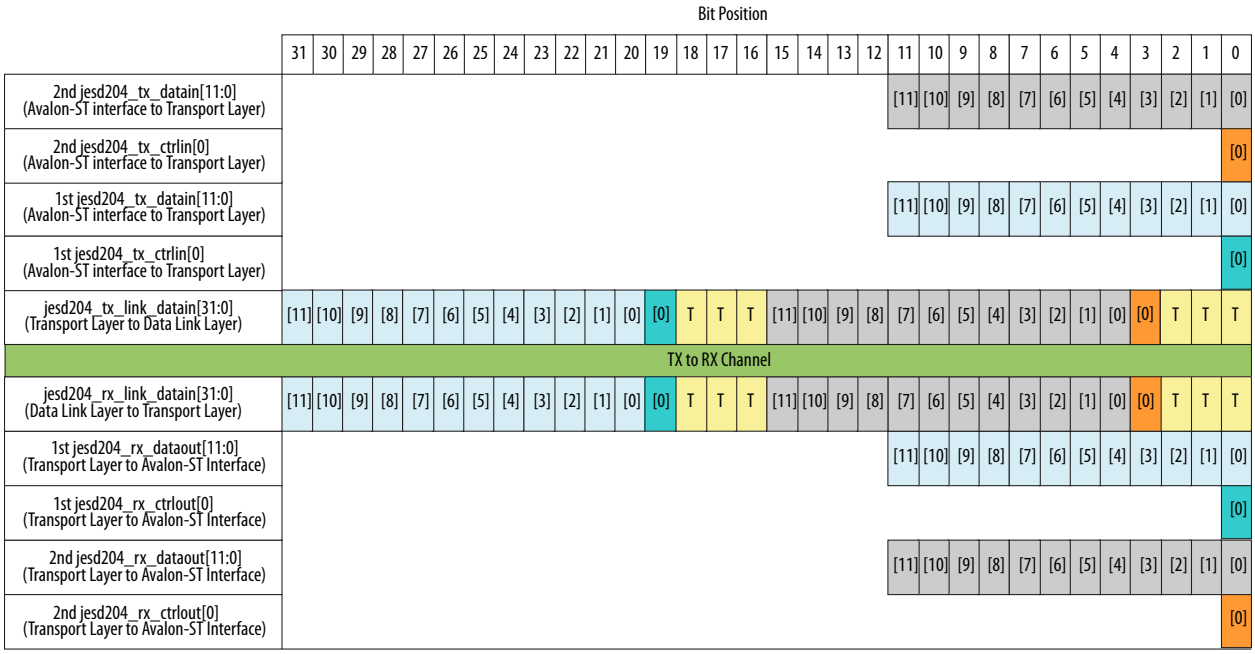
F Parameter	$f_{txframe}$ (txframe_clk frequency)	$f_{rxframe}$ (rxframe_clk frequency)
1	f_{txlink} x (4 / `F1_FRAMECLK_DIV`)	f_{rxlink} x (4 / `F1_FRAMECLK_DIV`)
2	f_{txlink} x (2 / `F2_FRAMECLK_DIV`)	f_{rxlink} x (2 / `F2_FRAMECLK_DIV`)
4	f_{txlink}	f_{rxlink}
8	f_{txlink} / 2	f_{rxlink} / 2

Data Bit and Content Mapping Scheme

One major function of the transport layer is to arrange the data bits in a specific way between the Avalon-ST interface and the DLL in the JESD204B IP core.

Figure 5-2 shows the mapping scheme in the transport layer across various TX to RX interfaces for a specific system configuration.

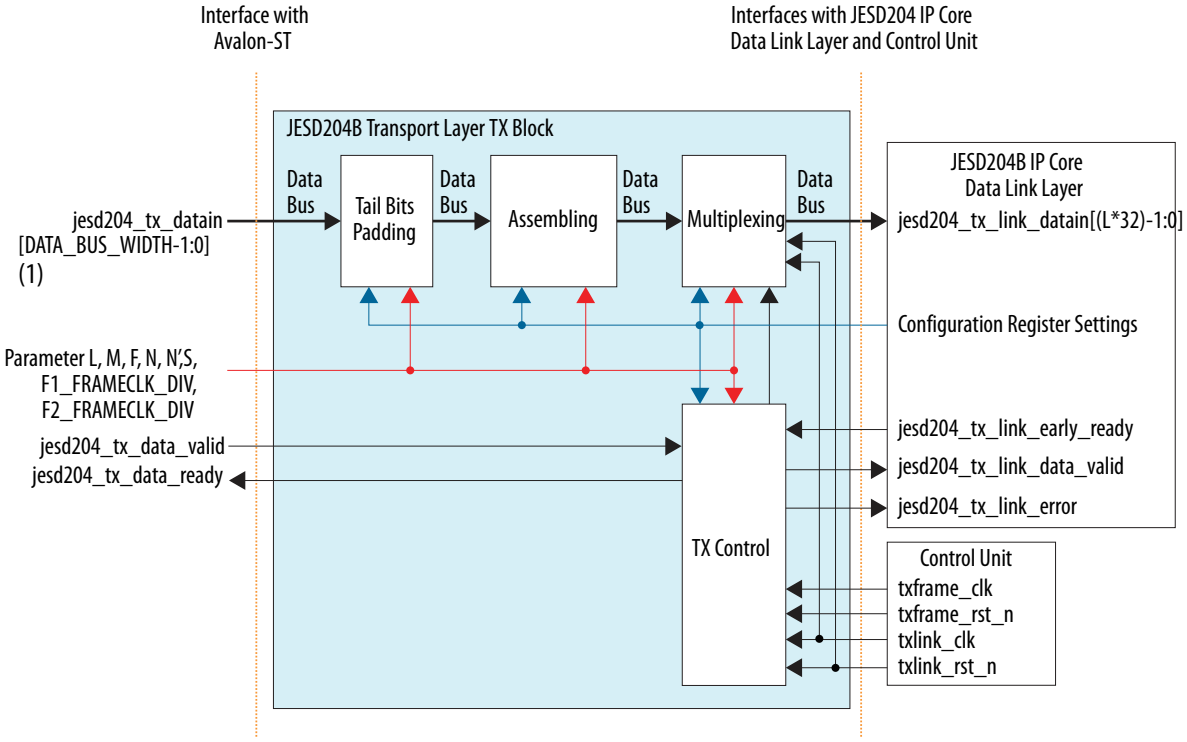
Figure 5-2: Mapping of Data Bit and Content Across Various Interfaces (LMF = 112, N = 12, N' = 16, S = 1, T represents the tail bits).



TX Path

The assembler in the TX path consists of the tail bits dropping, assembling, and multiplexing blocks.

Figure 5-3: TX Path Assembler Block Diagram



Note:
1. The DATA_BUS_WIDTH value is the data input bus width size, which depends on the F and L parameter.
bus_width=M*S*N
F=(M*S*N_PRIME)/(8*L)
M*S=(8*F*L)/N_PRIME
bus_width=(8*F*L*N)/N_PRIME

- Tail bits padding block—pads incoming data (`jesd204_tx_datain`) with "0" if **N** < 16, so that the padded data is 16 bits per sample.
- Assembling block—arranges the resulting data bits in a specific way according to the mapping scheme (refer to **Figure 5-2**).
- Multiplexing block—sends the multiplexed data to the DLL interface, determined by certain control signals from the TX control block.

Table 5-4: Assembler Parameter Settings

Parameter	Description	Value
L	Number of lanes per converter device.	1–8
F	Number of octets per frame.	1, 2, 4. 8
CS	Number of control bits or conversion sample.	0–3
N	Number of conversion bits per converter.	12-16
N'	Number of transmitted bits per sample in the user data format.	16

Parameter	Description	Value
F1_FRAMECLK_DIV	Only applies to cases where F=1. The divider ratio on the `frame_clk`. The assembler always uses the post-divided frame_clk (txframe_clk). (26)	1, 4
F2_FRAMECLK_DIV	Only applies to cases where F=2. The divider ratio on the `frame_clk`. The assembler always uses the post-divided frame_clk (txframe_clk). (26)	1, 2
RECONFIG_EN	Enable reconfiguration support in the transport layer. Only downscaling reconfiguration is supported. Disable the reconfiguration to reduce the logic.	0, 1
DATA_BUS_WIDTH	The data input bus width size that depends on the F and L. bus_width = M*S*N F = (M*S*N_PRIME)/(8*L) M*S = (8*F*L)/N_PRIME Therefore the data bus width = (8*F*L*N)/N_PRIME	(8*F*L*N)/N_PRIME
CONTROL_BUS_WIDTH	The control output bus width size. The width depends on the CS parameter as well as the M and S parameters. When CS=0, the control data is one bit wide (tie the signal to 0). If CS=0, bus width = 1 Else bus width = (OUTPUT_BUS_WIDTH/N*CS) while OUTPUT_BUS_WIDTH/N = M*S	OUTPUT_BUS_WIDTH/N*CS

Table 5-5: Assembler Signals

Signal	Clock Domain	Direction	Description
Control Unit			
`txlink_clk`	—	Input	TX link clock signal. This clock is equal to the TX data rate divided by 40. This clock is synchronous to the `txframe_clk` signal.
`txframe_clk`	—	Input	TX frame clock used by the transport layer. The frequency is a function of parameters *F*, *F1_FRAMECLK_DIV*, *F2_FRAMECLK_DIV* and *txlink_clk*. This clock is synchronous to the `txlink_clk` signal.

(26) Refer to the Table 5-7 to set the desired frame clock frequency with different **FRAMECLK_DIV** and **F** values.

Signal	Clock Domain	Direction	Description
txlink_rst_n	txlink_clk	Input	Reset for the TX link clock domain logic in the assembler. This reset is an active low signal and the deassertion is synchronous to the rising-edge of txlink_clk.
txframe_rst_n	txframe_clk	Input	Reset for the TX frame clock domain logic in the assembler. This reset is an active low signal and the deassertion is synchronous to the rising-edge of txframe_clk.
Signal	**Clock Domain**	**Direction**	**Description**
Between Avalon- ST and Transport Layer			
jesd204_tx_datain[(DATA_BUS_WIDTH)-1:0]	txframe_clk	Input	TX data from the Avalon-ST source interface. The source shall arrange the data in a specific order, as illustrated in the cases listed in **TX Path Data Remapping** section
jesd204_tx_controlin[(CONTROL_BUS_WIDTH)-1:0]	txframe_clk	Input	TX control data from the Avalon-ST source interface. The source shall arrange the data in a specific order, as illustrated in the cases listed in **TX Path Data Remapping** section
jesd204_tx_data_valid	txframe_clk	Input	Indicates whether the data from the Avalon-ST source interface to the transport layer is valid or invalid. • 0—data is invalid • 1—data is valid
jesd204_tx_data_ready	txlink_clk	Output	Indicates that the transport layer is ready to accept data from the Avalon-ST source interface. • 0—transport layer is not ready to receive data • 1—transport layer is ready to receive data
Signal	**Clock Domain**	**Direction**	**Description**
Between Transport Layer and DLL			

Signal	Clock Domain	Direction	Description
jesd204_tx_link_datain[(L*32)-1:0]	txlink_clk	Output	Indicates transmitted data from the transport layer to the DLL at txlink_clk clock rate, where four octets are packed into a 32-bit data width per lane. The data format is big endian. The table below illustrates the data mapping for L = 4: jesd204_tx_link_datain [x:y] / Lane [31:0] / 0 [63:32] / 1 [95:64] / 2 [127:96] / 3 Connect this signal to the TX DLL jesd204_tx_link_data[] input pin.
jesd204_tx_link_data_valid	txlink_clk	Output	Indicates whether the jesd204_tx_link_datain[] is valid or invalid. • 0—jesd204_tx_link_datain[] is invalid • 1—jesd204_tx_link_datain[] is valid Connect this signal to the TX DLL jesd204_tx_link_valid input pin.
jesd204_tx_link_early_ready (27)	txlink_clk	Input	Indicates that the DLL requires valid data at the subsequent implementation-specific duration. Connect this signal to the TX DLL jesd204_tx_frame_ready output pin.
jesd204_tx_link_error	txlink_clk	Output	Indicates an error at the Avalon-ST source interface. Specifically, this signal is asserted when jesd204_tx_data_valid = "0" while jesd204_tx_data_ready = "1". The DLL subsequently reports this error to the CSR block. Connect this signal to the TX DLL jesd204_tx_frame_error input pin.

Signal	Clock Domain	Direction	Description
CSR in DLL			

(27) If a JESD device of No Multiple-Converter Device Alignment, Single-Lane (NMCDA-SL) class is deployed, Altera recommends that you tie this input signal to "1".

Signal	Clock Domain	Direction	Description
csr_l[4:0] (28)	mgmt_clk	Input	Indicates the number of active lanes for the link. This 5-bit bus represents the **L** value in zero-based binary format. For example, if **L** = 1, the csr_l[4:0] = "00000". This design example supports the following values: • 00000 • 00001 • 00011 • 00111 Any programmed value beyond the supported range may result in undeterminable behavior in the transport layer. You must ensure that the csr_l[4:0] value always matches the system parameter **L** value when it is in static configuration. Runtime reconfiguration supports L fallback. For static configuration, set the maximum L and reconfigure csr_l[] to a smaller value during runtime. This transport layer only supports higher index channels to be powered down. To interleave the de-commision channels, you need to modify the interface connection from the DLL to transport layer. Connect this signal to the TX DLL csr_l[] output pin.
csr_f[7:0] (28)	mgmt_clk	Input	Indicates the number of octets per frame. This 8-bit bus represents the **F** value in zero-based binary format. For example, if **F** = 2, the csr_f[7:0] = "00000001". This design example supports the following values: • 00000000 • 00000001 • 00000011 • 00000111 Any programmed value beyond the supported range may result in undeterminable behavior in the transport layer. Ensure that the csr_f[7:0] value always matches the system parameter **F** value when it is in static configuration. Connect this signal to the TX DLL csr_f[] output pin.

(28) This signal should be static and valid before the deassertion of the link_rst_n and frame_rst_n signals.

Signal	Clock Domain	Direction	Description
csr_n[4:0] (28)	mgmt_clk	Input	Indicates the converter resolution. This 5-bit bus represents the **N** value in zero-based binary format. For example, if **N** = 16, the csr_n[4:0] = "01111". This design example supports the following values: • 01011 • 01100 • 01101 • 01110 • 01111 Any programmed value beyond the supported range may result in undeterminable behavior in the transport layer. You must ensure that the csr_n[4:0] value always match the system parameter **N** value. Connect this signal to the TX DLL csr_n[] output pin.

TX Path Operation

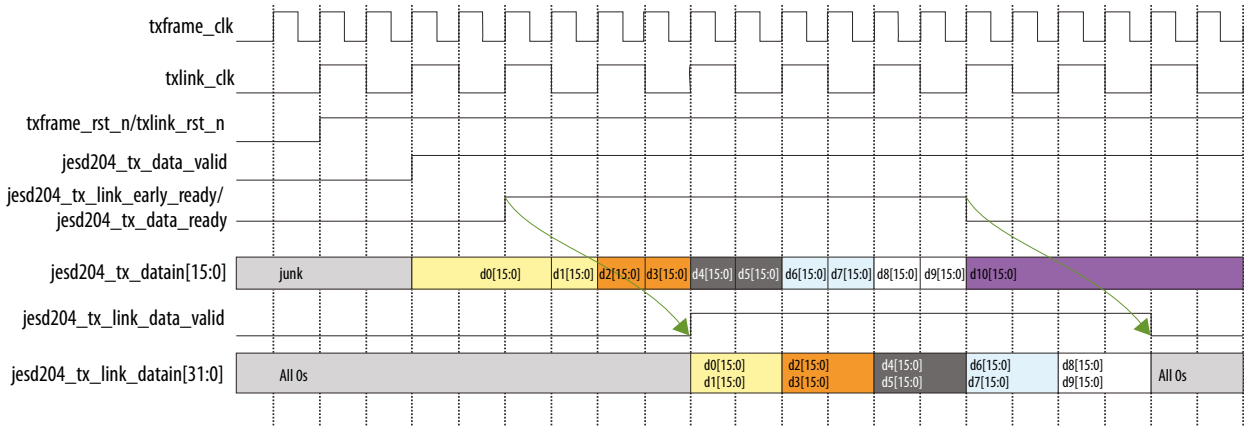
The data transfer protocol between the Avalon-ST interface and the TX path transport layer is data transfer with backpressure, where ready_latency = 0.

Figure 5-4: TX Operation Behavior

This figure shows the data transmission for a system configuration of **LMF** = 112, **N** = **N'** = 16, **S** = 1.

Operation:

- Upon the deassertion of the `txframe_rst_n` signal, the `jesd204_tx_link_early_ready` signal from the DLL to the transport layer is asserted some time later, which activates the transport layer to start sampling the `jesd204_tx_datain[15:0]` signal from the Avalon-ST interface.
- Each sampled 16-bit data is first written in a FIFO with a depth of four.
- Once the FIFO accumulates 32-bit data, the data is streamed to the DLL accordingly through the `jesd204_tx_link_datain[31:0]` signal.
- Finally, the `jesd204_tx_link_early_ready` and `jesd204_tx_data_ready` signals deassert because the DLL has entered code group synchronization state in this scenario.



TX Data Transmission

This section explains the data transmission behavior when there is a valid TX data out from the TL to DLL.

Upon the deassertion of `txframe_rst_n` signal, the link's `jesd204_tx_link_early_ready` signal equals to "1". This setting activates the TL to start sampling `jesd204_tx_datain` signal from the Avalon-ST interface and transmits sampled data (`jesd204_tx_link_datain`) to the TX link. The TX link only captures valid data from the TL when the `jesd204_tx_link_ready` signal equals to "1" (in user data phase). This means all the data transmitted from the TL before `jesd204_tx_link_ready` signal equals to "1" are ignored.

Figure 5-5: TX Data Transmission

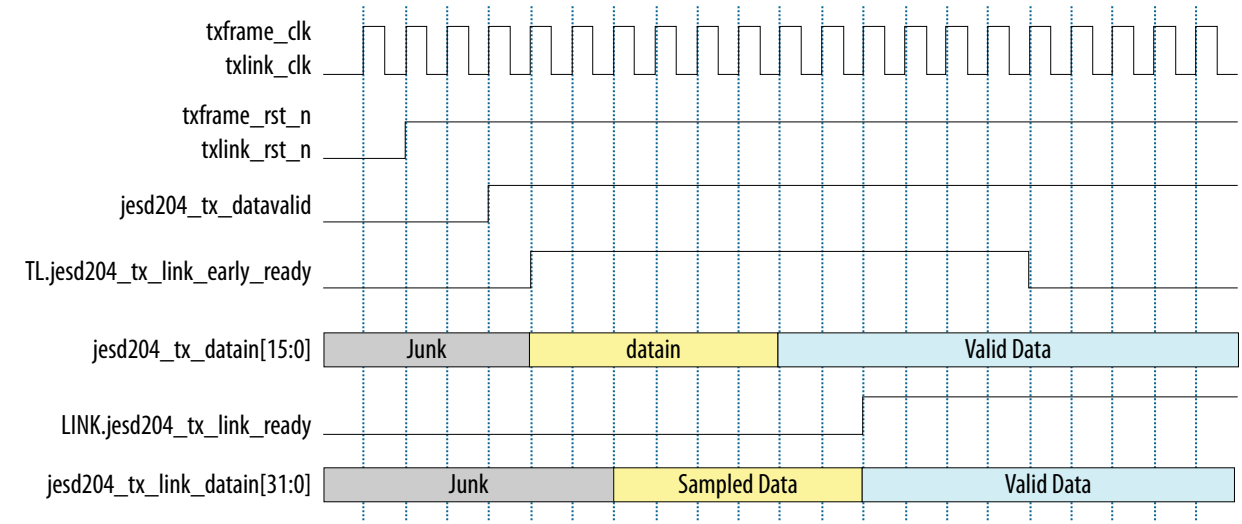
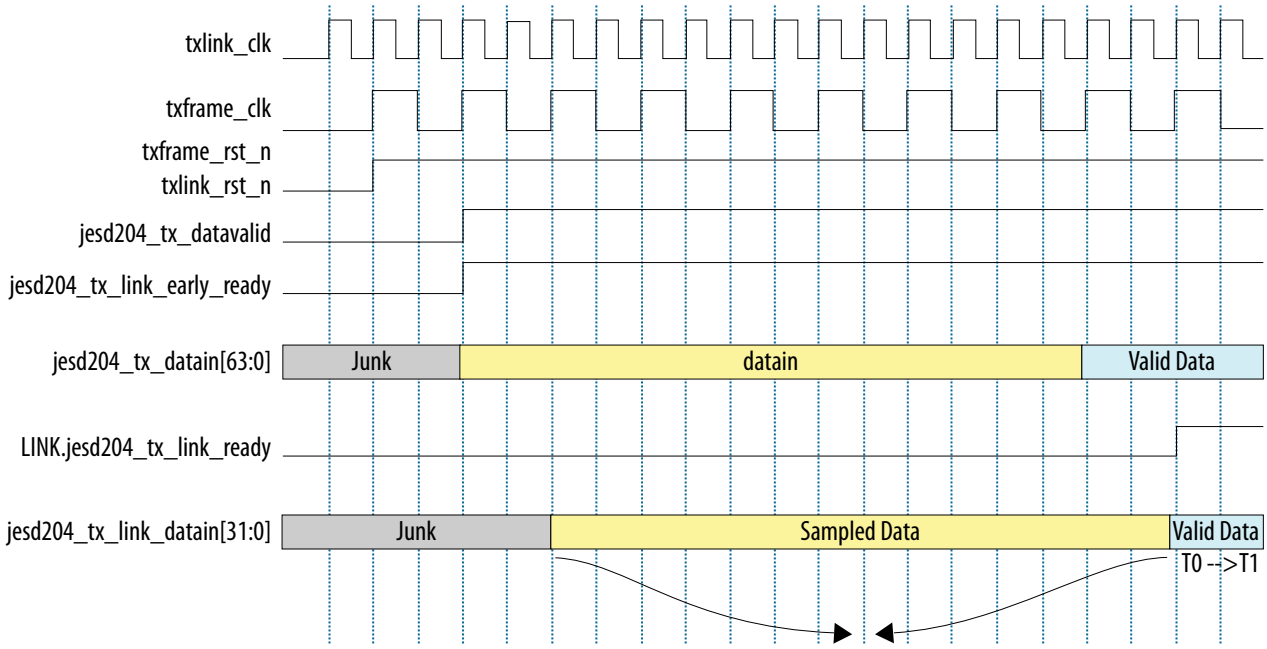


Figure 5-6: TX Data Transmission (For F = 8)



When F = 8, the data latency for jesd204_tx_link_datain should always be in an even latency link_clk count to ensure that the first valid data captured by the TX link is T0 data followed by T1 data.

TX Path Data Remapping

The JESD204B IP core implements the data transfer in big endian format.

The data is arranged so that the L0 is always on the right (LSB) in the data bus. In the big endian implementation, the oldest data (F0) is placed at the MSB in L0.

Table 5-6: Data Mapping for F=1, L=4

F = 1				
Supported M and S	M*S=2 for F=1, L=4 F=1 supports either (case1: M=1, S=2) or (case2: M=2, S=1)			
F1_FRAMCLK_DIV=1	{F8F12, F0F4} = jesd204_tx_datain[31:0]	Case1: M=1, S=2	M0S0=F0F4, M0S1=F8F12, at 1st frameclk	
		Case2: M=2, S=1	M0S0=F0F4, M1S0=F8F12, at 1st frameclk	
	{F9F13, F1F5} = jesd204_tx_datain[31:0]	Case1: M=1, S=2	M0S0=F1F5, M0S1=F9F13, at 2nd frameclk	
		Case2: M=2, S=1	M0S0=F1F5, M1S0=F9F13, at 2nd frameclk	
	{F10F114, F2F6} = jesd204_tx_datain[31:0]	Case1: M=1, S=2	M0S0=F2F6, M0S1=F10F14, at 3rd frameclk	
		Case2: M=2, S=1	M0S0=F2F6, M1S0=F10F14, at 3rd frameclk	
	{F11F15, F3F7} = jesd204_tx_datain[31:0]	Case1: M=1, S=2	M0S0=F3F7, M0S1=F11F15, at 4th frameclk	
		Case2: M=2, S=1	M0S0=F3F7, M1S0=F11F15, at 4th frameclk	
F1_FRAMCLK_DIV=4	{{F11F15, F3F7},{F10F114, F2F6},{F9F13, F1F5},{F8F12, F0F4}} = jesd204_tx_datain[127:0]			
Lane	L3	L2	L1	L0
Data Out	{F12, F13, F14, F15}	{F8, F9, F10, F11}	{F4, F5, F6, F7}	{F0, F1, F2, F3}

Table 5-7: Data Mapping for F=2, L=4

F = 2	
Supported M and S	M*S=4 for F=2, L=4 F=2 supports either (case1: M=1, S=4), (case2: M=2, S=2) or (case3: M=4, S=1)

F = 2				
F2_FRAMCLK_DIV=1	{F12F13, F8F9,F4F5, F0F1}= jesd204_tx_datain[63:0]	Case1: M=1, S=4	M0S0=F0F1, M0S1=F4F5, M0S2=F8F9, M0S3=F12F13 at 1st frameclk	
		Case2: M=2, S=2	M0S0=F0F1, M0S1=F4F5, M1S0=F8F9, M1S1=F12F13 at 1st frameclk	
		Case3: M=4, S=1	M0S0=F0F1, M1S0=F4F5, M2S0=F8F9, M3S0=F12F13 at 1st frameclk	
	{F14F15, F10F11,F6F7, F2F3} = jesd204_tx_datain[63:0]	Case1: M=1, S=4	M0S0=F2F3, M0S1=F6F7, M0S2=F10F11, M0S3=F14F15 at 2nd frameclk	
		Case2: M=2, S=2	M0S0=F2F3, M0S1=F6F7, M1S0=F10F11, M1S1=F14F15 at 2nd frameclk	
		Case3: M=4, S=1	M0S0=F2F3, M1S0=F6F7, M2S0=F10F11, M3S0=F14F15 at 2nd frameclk	
F2_FRAMCLK_DIV=2	{{F14F15, F10F11,F6F7, F2F3}, {F12F13, F8F9,F4F5, F0F1}} = jesd204_tx_datain[127:0]			
Lane	L3	L2	L1	L0
Data Out	{F12, F13, F14, F15}	{F8, F9, F10, F11}	{F4, F5, F6, F7}	{F0, F1, F2, F3}

Table 5-8: Data Mapping for F=4, L=4

F = 4				
Supported M and S	M*S=8 for F=4, L=4 F=4 supports either (case1: M=1, S=8), (case2: M=2, S=4), (case3: M=4, S=2) or (case4: M=8, S=1)			
F=4	{F14F15,F12F13, F10F11, F8F9,F6F7,F4F5, F2F3,F0F1} = jesd204_tx_datain[127:0]	Case1: M=1, S=8	{M0S7, M0S6, M0S5, M0S4, M0S3, M0S2, M0S1, M0S0}	
		Case2: M=2, S=4	{M1S3, M1S2, M1S1, M1S0, M0S3, M0S2, M0S1, M0S0}	
		Case3: M=4, S=2	{M3S1, M3S0, M2S1, M2S0, M1S1, M1S0, M0S1, M0S0}	
		Case4: M=8, S=1	{M7S0, M6S0, M5S0, M4S0, M3S0, M2S0, M1S0, M0S0}	
Lane	L3	L2	L1	L0
Data Out	{F12, F13, F14, F15}	{F8, F9, F10, F11}	{F4, F5, F6, F7}	{F0, F1, F2, F3}

Table 5-9: Data Mapping for F=8, L=4

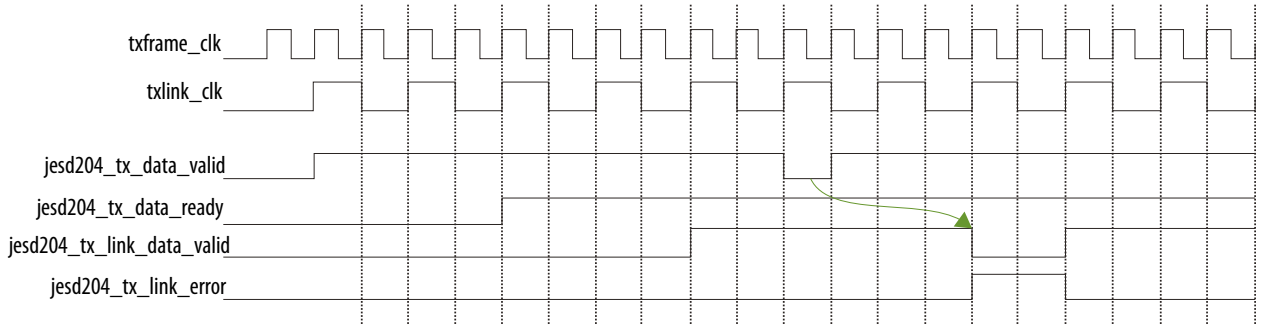
F = 8				
Supported M and S	M*S=16 for F=8, L=4 F=8 supports either (case1: M=1, S=16), (case2: M=2, S=8), (case3: M=4, S=4), (case4: M=8, S=2) or (case5: M=16, S=1)			
F=8	{{F3031, F28F29,F26F27, F24F25}, {F22F23, F20F21,F18F19, F16F17}, {F14F15, F12F13,F10F11, F8F9}, {F6F7,F4F5, F2F3,F0F1}} = jesd204_tx_ datain[255:0]	Case1: M=1, S=16	{M0S15, M0S14, M0S13, M0S12, M0S11, M0S10, M0S9, M0S8, M0S7, M0S6, M0S5, M0S4, M0S3, M0S2, M0S1, M0S0}	
Lane	L3	L2	L1	L0
Data Out at linkclk T0	{F24, F25, F26, F27}	{F16, F17, F18, F19}	{F8, F9, F10, F11}	{F0, F1, F2, F3}
Data Out at linkclk T1	{F28, F29, F30, F31}	{F20, F21, F22, F23}	{F12, F13, F14, F15}	{F4, F5, F6, F7}

TX Error Reporting

For TX path error reporting, the transport layer expects a valid stream of TX data from the Avalon-ST interface (indicated by `jesd204_tx_data_valid` signal = 1) as long as the `jesd204_tx_data_ready` remains asserted. If the `jesd204_tx_data_valid` signal unexpectedly deasserts during this stage, the transport layer reports an error to the DLL by asserting the `jesd204_tx_link_error` signal and deasserting the `jesd204_tx_link_data_valid` signal accordingly, as shown in the timing diagram below.

Figure 5-7: TX Error Reporting

The `jesd204_tx_data_valid` signal deasserts for one frame_clk and cannot be sampled by the link_clk.



TX Latency

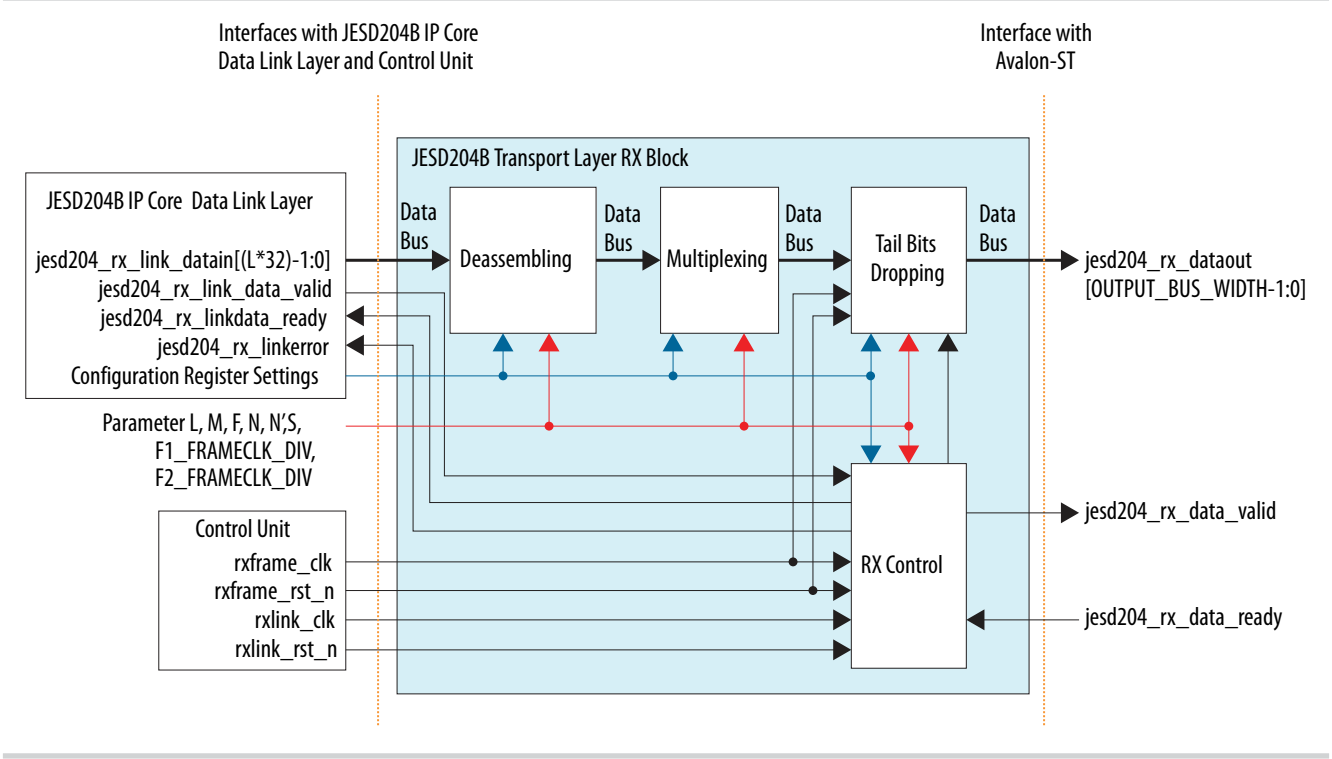
Table 5-10: TX Latency Associated with Different F and FRAMECLK_DIV Settings.

F	FRAMECLK_DIV	TX Latency
1	1	3 txframe_clk period. • Maximum 5 txframe_clk period for byte 3 • Minimum 2 txframe_clk period for byte 0
1	4	1 txframe_clk period
2	1	3 txframe_clk period. • Maximum 4 txframe_clk period for byte 2 and byte 3 • Minimum 3 txframe_clk period for byte 0 and byte 1
2	2	1 txframe_clk period
4	—	1 txframe_clk period
8	—	1 txframe_clk period

RX Path

The deassembler in the RX path consists of the tail bits dropping, deassembling, and multiplexing blocks.

Figure 5-8: RX Path Assembler Block Diagram



- Tail bit dropping block—drops padded tail bits in the incoming data (`jesd204_rx_link_datain`).
- Deassembling block—rearranges the resulting data bits in a specific way according to the mapping scheme (refer to **Figure 5-2**).
- Multiplexing block—sends the multiplexed data to the Avalon-ST interface, determined by certain control signals from the RX control block.

Table 5-11: Deassembler Parameter Settings

Parameter	Description	Value
L	Number of lanes per converter device.	1–8
F	Number of octets per frame.	1, 2, 4, 8
CS	Number of control bits or conversion sample.	0–3
N	Number of conversion bits per converter.	12-16
N'	Number of transmitted bits per sample in the user data format.	16
F1_FRAMECLK_DIV	Only applies to cases where F=1. The divider ratio on the `frame_clk`. The deassembler always uses the post-divided frame_clk (rxframe_clk). (29)	1, 4
F2_FRAMECLK_DIV	Only applies to cases where F=2. The divider ratio on the `frame_clk`. The deassembler always uses the post-divided frame_clk (rxframe_clk). (29)	1, 2
RECONFIG_EN	Enable reconfiguration support in the transport layer. Only downscaling reconfiguration is supported. Disable the reconfiguration to reduce the logic.	0, 1
OUTPUT_BUS_WIDTH	The data output bus width size that depends on the F and L. bus_width = M*S*N F = (M*S*N_PRIME)/(8*L) M*S = (8*F*L)/N_PRIME Therefore the output bus width = (8*F*L*N)/N_PRIME	(8*F*L*N)/N_PRIME
CONTROL_BUS_WIDTH	The control output bus width size. The width depends on the CS parameter as well as the M and S parameters. When CS=0, the control data is one bit wide (tie the signal to 0). If CS=0, bus width = 1 Else bus width = (OUTPUT_BUS_WIDTH/N*CS) while OUTPUT_BUS_WIDTH/N = M*S	OUTPUT_BUS_WIDTH/N*CS

(29) Refer to the **Table 5-7** to set the desired frame clock frequency with different **FRAMECLK_DIV** and **F** parameter values.

Table 5-12: Deassembler Signals

Signal	Clock Domain	Direction	Description
Control Unit			
rxlink_clk	—	Input	RX link clock signal. This clock is equal to the RX data rate divided by 40. This clock is synchronous to the rxframe_clk signal.
rxframe_clk	—	Input	RX frame clock used by the deassembler. The frequency is a function of parameters *F*, *F1_FRAMECLK_DIV*, *F2_FRAMECLK_DIV* and *rxlink_clk*. This clock is synchronous to the rxlink_clk signal.
rxlink_rst_n	rxlink_clk	Input	Reset for the RX link clock domain logic in the deassembler. This reset is an active low signal and the deassertion is synchronous to the rising-edge of rxlink_clk.
rxframe_rst_n	rxframe_clk	Input	Reset for the RX frame clock domain logic in the deassembler. This reset is an active low signal and the deassertion is synchronous to the rising-edge of rxframe_clk.
Signal	**Clock Domain**	**Direction**	**Description**
Between Avalon- ST and Transport Layer			
jesd204_rx_dataout[(OUTPUT_BUS_WIDTH)-1:0]	rxframe_clk	Output	RX data to the Avalon-ST source interface. The transport layer arranges the data in a specific order, as illustrated in the cases listed in **RX Path Data Remapping** section.
jesd204_rx_controlout[CONTROL_BUS_WIDTH -1:0]	rxframe_clk	Output	RX control data to the Avalon-ST source interface. The transport layer arranges the data in a specific order, as illustrated in the cases listed in **RX Path Data Remapping** section.
jesd204_rx_data_valid	rxframe_clk	Output	Indicates whether the data from the transport layer to the Avalon-ST sink interface is valid or invalid. • 0—data is invalid • 1—data is valid

Signal	Clock Domain	Direction	Description
jesd204_rx_data_ready	rxframe_clk	Input	Indicates that the Avalon-ST sink interface is ready to accept data from the transport layer. • 0—Avalon-ST sink interface is not ready to receive data • 1—Avalon-ST sink interface is ready to receive data
Signal	**Clock Domain**	**Direction**	**Description**
Between Transport Layer and DLL			
jesd204_rx_link_datain[(L*32)-1:0]	rxlink_clk	Input	Indicates received data from the DLL to the transport layer, where four octets are packed into a 32-bit data width per lane. The data format is big endian. The table below illustrates the data mapping for L = 4: jesd204_rx_link_datain [x:y] / Lane [31:0] / 0 [63:32] / 1 [95:64] / 2 [127:96] / 3 Connect this signal to the RX DLL jesd204_rx_link_data[] output pin.
jesd204_rx_link_data_valid	rxlink_clk	Input	Indicates whether the jesd204_rx_link_datain[] is valid or invalid. • 0—jesd204_rx_link_datain[] is invalid • 1—jesd204_rx_link_datain[] is valid Connect this signal to the RX DLL jesd204_rx_link_valid output pin.
jesd204_rx_link_data_ready	rxframe_clk	Output	Indicates that the transport layer is ready to sample jesd204_rx_link_datain[]. • 0—transport layer is not ready to sample jesd204_rx_link_datain[] • 1—transport layer starts sampling jesd204_rx_link_datain[] at the next clock cycle. Connect this signal to the RX DLL jesd204_rx_link_ready input pin.

Signal	Clock Domain	Direction	Description
jesd204_rx_link_error	rxlink_clk	Output	Indicates an empty data stream due to invalid data. This signal is asserted high to indicate an error at the Avalon-ST sink interface (for example, when jesd204_rx_data_valid = "1" while jesd204_rx_data_ready = "0"). The DLL subsequently reports this error to the CSR block. Connect this signal to the RX DLL jesd204_rx_frame_error input pin.
Signal	**Clock Domain**	**Direction**	**Description**
CSR in DLL			
csr_l[4:0] (30)	mgmt_clk	Input	Indicates the number of active lanes for the link. This 5-bit bus represents the **L** value in zero-based binary format. For example, if **L** = 1, the csr_l[4:0] = "00000". This design example supports the following values: • 00000 • 00001 • 00011 • 00111 Any programmed value beyond the supported range may result in undeterminable behavior in the transport layer. You must ensure that the csr_l[4:0] value always match the system parameter **L** value. Runtime reconfiguration supports L fallback. For static configuration, set the maximum L and reconfigure csr_l[] to a smaller value during runtime. This transport layer only supports higher index channels to be powered down. To interleave the de-commision channels, you need to modify the interface connection from the DLL to transport layer. Connect this signal to the RX DLLcsr_l[] output pin.

(30) This signal should be static and valid before the deassertion of the link_rst_n and frame_rst_n signals.

Signal	Clock Domain	Direction	Description
csr_f[7:0] (30)	mgmt_clk	Input	Indicates the number of octets per frame. This 8-bit bus represents the **F** value in zero-based binary format. For example, if **F** = 2, the csr_f[7:0] = "00000001". This design example supports the following values: • 00000000 • 00000001 • 00000011 • 00000111 Any programmed value beyond the supported range may result in undeterminable behavior in the transport layer. You must ensure that the csr_f[7:0] value always match the system parameter **F** value. Connect this signal to the RX DLL csr_f[] output pin.
csr_n[4:0] (30)	mgmt_clk	Input	Indicates the converter resolution. This 5-bit bus represents the **N** value in zero-based binary format. For example, if **N** = 16, the csr_n[4:0] = "01111". This design example supports the following values: • 01011 • 01100 • 01101 • 01110 • 01111 Any programmed value beyond the supported range may result in undeterminable behavior in the transport layer. You must ensure that the csr_n[4:0] value always match the system parameter **N** value. Connect this signal to the RX DLL csr_n[] output pin.

RX Path Operation

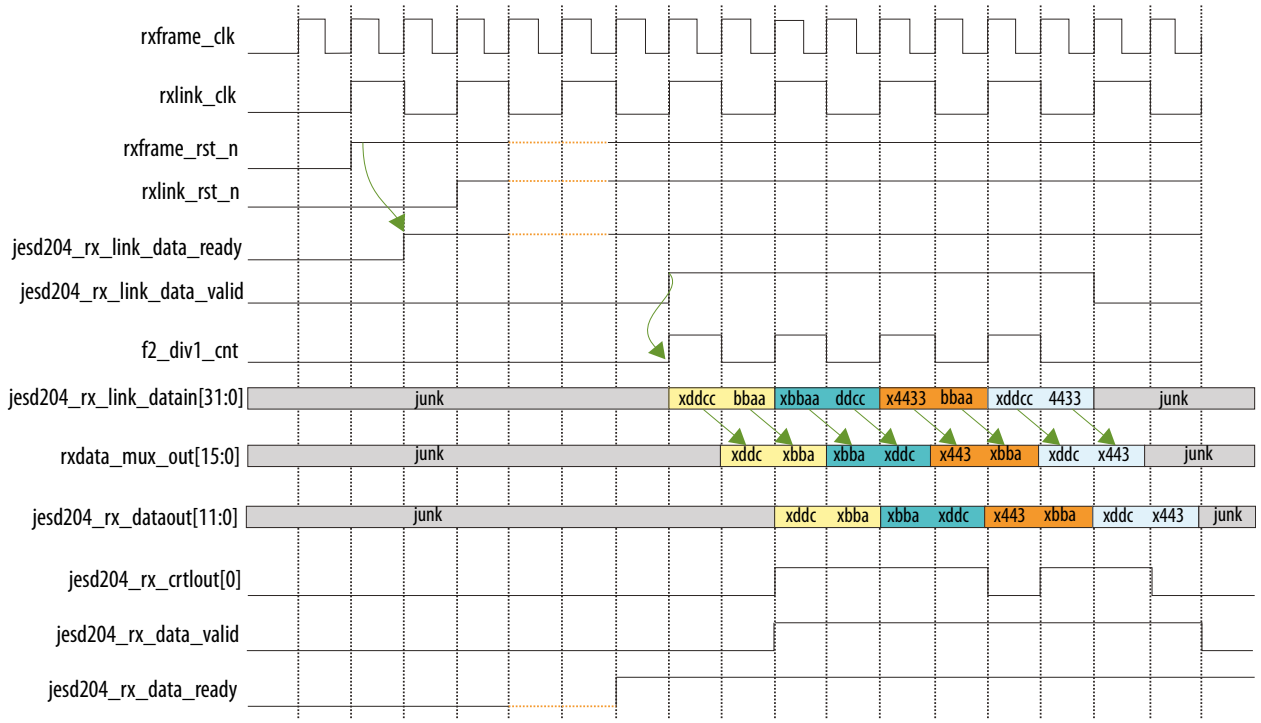
The data transfer protocol between the Avalon-ST interface and the RX path transport layer is data transfer without backpressure. Therefore, the sink shall always be ready to sample the incoming data whenever data at the source is valid.

Figure 5-9: RX Operation Behavior

This figure shows the data transmission for a system configuration of **LMF** = 112, **N** = 12, **N'** = 16, **S** =1.

Operation:

- Upon the deassertion of the `rxframe_rst_n` signal, the `jesd204_rx_link_data_ready` signal from the deassembler to the DLL is asserted at the next `rxframe_clk`.
- Subsequently, the DLL asserts the `jesd204_rx_link_data_valid` signal for the deassembler to activate the `f2_div1_cnt` signal logic and to start sampling the `jesd204_rx_link_datain[31:0]` signal. (31)
- At the following `rxframe_clk`, the `jesd204_rx_data_valid` is asserted along with the multiplexed `jesd204_rx_dataout[11:0]` signal to stream data to the Avalon-ST interface.
- Finally, the DLL deasserts the `jesd204_rx_link_data_valid` signal when there is no more valid data.
- The deassembler deactivates the `f2_div1_cnt` signal logic accordingly, and deasserts the `jesd204_rx_data_valid` at the next `rxframe_clk`.



RX Data Reception

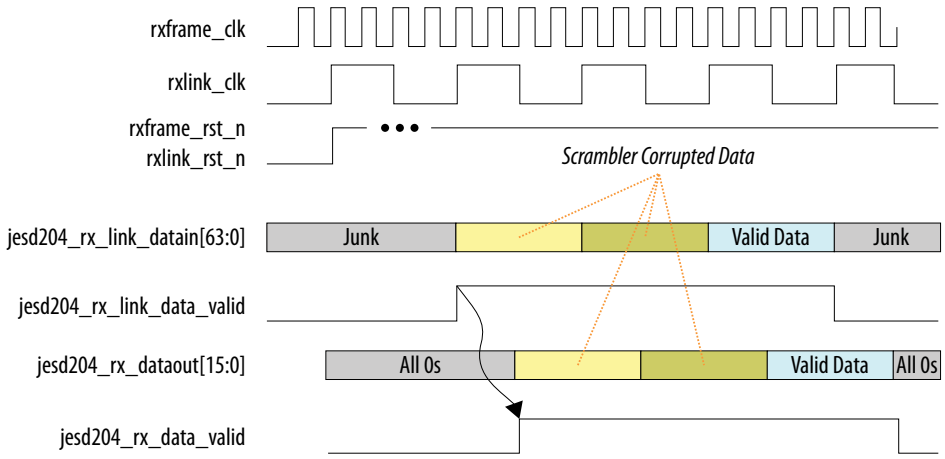
This section explains when there is a valid RX data out from the DLL to the TL to with scrambler enabled.

The MAC layer process the `jesd204_rx_dataout` signal once the TL asserts the `jesd204_rx_data_valid` signal. However, there are some data that should be discarded by the upper layer when the you enable the scrambler. This is because the initial unknown seed value within the scrambler can corrupt the very first eight octets, which is the data for the first two link clock cycles. The data can be translated to the frame

(31) The `f2_div1_cnt` signal is internally generated in the RX control block to correctly stream data to the Avalon-ST interface.

clock cycle depending on the *F* and *FRAMECLK_DIV* parameters selected based on the frame clock to link clock relationship.

Figure 5-10: RX Data Reception



Related Information

Relationship Between Frame Clock and Link Clock on page 5-9

RX Path Data Remapping

The JESD204B IP core implements the data transfer in big endian format.

The data is arranged so that the L0 is always on the right (LSB) in the data bus. In the big endian implementation, the oldest data (F0) is placed at the MSB in L0.

Table 5-13: Data Mapping for F=1, L=4

F = 1				
Lane	L3	L2	L1	L0
Data In	{F12, F13, F14, F15}	{F8, F9, F10, F11}	{F4, F5, F6, F7}	{F0, F1, F2, F3}
Supported M and S	M*S=2 for F=1, L=4 F=1 supports either (case1: M=1, S=2) or (case2: M=2, S=1) Assuming N=16, M0S0=jesd204_rx_dataout[15:0], M0S1/M1S0= jesd204_rx_dataout[31:16]			

F = 1			
F1_FRAMCLK_DIV=1	cnt=0 : jesd204_rx_dataout[31:0] = {F8F12, F0F4}	Case1: M=1, S=2	M0S0=F0F4, M0S1=F8F12, at 1st frameclk
		Case2: M=2, S=1	M0S0=F0F4, M1S0=F8F12, at 1st frameclk
	cnt=1: jesd204_rx_dataout[31:0] = {F9F13, F1F5}	Case1: M=1, S=2	M0S0=F1F5, M0S1=F9F13, at 2nd frameclk
		Case2: M=2, S=1	M0S0=F1F5, M1S0=F9F13, at 2nd frameclk
	cnt=2: jesd204_rx_dataout[31:0] = {F10F114, F2F6}	Case1: M=1, S=2	M0S0=F2F6, M0S1=F10F14, at 3rd frameclk
		Case2: M=2, S=1	M0S0=F2F6, M1S0=F10F14, at 3rd frameclk
	cnt=3: jesd204_rx_dataout[31:0] = {F11F15, F3F7}	Case1: M=1, S=2	M0S0=F3F7, M0S1=F11F15, at 4th frameclk
		Case2: M=2, S=1	M0S0=F3F7, M1S0=F11F15, at 4th frameclk
F1_FRAMCLK_DIV=4	jesd204_rx_dataout[127:0] = {{F11F15, F3F7},{F10F114, F2F6},{F9F13, F1F5},{F8F12, F0F4}}		

Table 5-14: Data Mapping for F=2, L=4

F = 2				
Lane	L3	L2	L1	L0
Data In	{F12, F13, F14, F15}	{F8, F9, F10, F11}	{F4, F5, F6, F7}	{F0, F1, F2, F3}
Supported M and S	M*S=4 for F=2, L=4 F=2 supports either (case1: M=1, S=4), (case2: M=2, S=2) or (case3: M=4, S=1)			

F = 2			
F2_FRAMCLK_DIV=1	cnt=0: jesd204_rx_dataout[63:0] = {F12F13, F8F9,F4F5, F0F1}	Case1: M=1, S=4	M0S0=F0F1, M0S1=F4F5, M0S2=F8F9, M0S3=F12F13 at 1st frameclk
		Case2: M=2, S=2	M0S0=F0F1, M0S1=F4F5, M1S0=F8F9, M1S1=F12F13 at 1st frameclk
		Case3: M=4, S=1	M0S0=F0F1, M1S0=F4F5, M2S0=F8F9, M3S0=F12F13 at 1st frameclk
	cnt=1: jesd204_rx_dataout[63:0] = {F14F15, F10F11,F6F7, F2F3}	Case1: M=1, S=4	M0S0=F2F3, M0S1=F6F7, M0S2=F10F11, M0S3=F14F15 at 2nd frameclk
		Case2: M=2, S=2	M0S0=F2F3, M0S1=F6F7, M1S0=F10F11, M1S1=F14F15 at 2nd frameclk
		Case3: M=4, S=1	M0S0=F2F3, M1S0=F6F7, M2S0=F10F11, M3S0=F14F15 at 2nd frameclk
F2_FRAMCLK_DIV=2	jesd204_rx_dataout[127:0] = {{F14F15, F10F11,F6F7, F2F3}, {F12F13, F8F9,F4F5, F0F1}}		

Table 5-15: Data Mapping for F=4, L=4

F = 4				
Lane	L3	L2	L1	L0
Data In	{F12, F13, F14, F15}	{F8, F9, F10, F11}	{F4, F5, F6, F7}	{F0, F1, F2, F3}
Supported M and S	M*S=8 for F=4, L=4 F=4 supports either (case1: M=1, S=8), (case2: M=2, S=4), (case3: M=4, S=2) or (case4: M=8, S=1)			
F=4	jesd204_rx_dataout[127:0] = {F14F15, F12F13,F10F11, F8F9,F6F7,F4F5, F2F3,F0F1}	Case1: M=1, S=8	{M0S7, M0S6, M0S5, M0S4, M0S3, M0S2, M0S1, M0S0}	
		Case2: M=2, S=4	{M1S3, M1S2, M1S1, M1S0, M0S3, M0S2, M0S1, M0S0}	
		Case3: M=4, S=2	{M3S1, M3S0, M2S1, M2S0, M1S1, M1S0, M0S1, M0S0}	
		Case4: M=8, S=1	{M7S0, M6S0, M5S0, M4S0, M3S0, M2S0, M1S0, M0S0}	

Table 5-16: Data Mapping for F=8, L=4

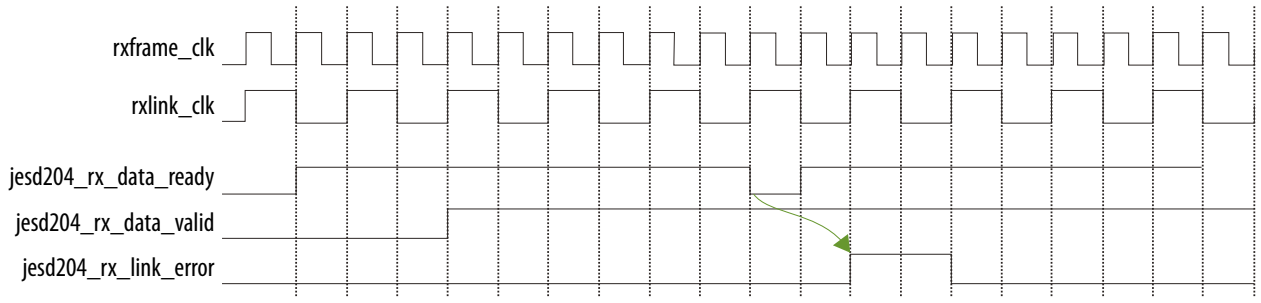
F = 8				
Lane	L3	L2	L1	L0

F = 8				
Data In linkclk T0	{F24, F25, F26, F27}	{F16, F17, F18, F19}	{F8, F9, F10, F11}	{F0, F1, F2, F3}
Data In linkclk T1	{F28, F29, F30, F31}	{F20, F21, F22, F23}	{F12, F13, F14, F15}	{F4, F5, F6, F7}
Supported M and S	M*S=16 for F=8, L=4 F=8 supports either (case1: M=1, S=16), (case2: M=2, S=8), (case3: M=4, S=4), (case4: M=8, S=2) or (case5: M=16, S=1)			
F=8	jesd204_rx_dataout[255:0] = {{F3031, F28F29,F26F27, F24F25},{F22F23, F20F21,F18F19, F16F17},{F14F15, F12F13, F10F11,F8F9}, {F6F7,F4F5, F2F3,F0F1}}	Case1: M=1, S=16	{M0S15, M0S14, M0S13, M0S12, M0S11, M0S10, M0S9, M0S8, M0S7, M0S6, M0S5, M0S4, M0S3, M0S2, M0S1, M0S0}	

RX Error Reporting

For RX path error reporting, the transport layer expects the AL to always be ready to sample the RX data (as indicated by the `jesd204_rx_data_ready` signal equal to "1") as long as the `jesd204_rx_data_valid` remains asserted. If the `jesd204_rx_data_ready` signal unexpectedly deasserts, the transport layer reports the error to the DLL by asserting the `jesd204_rx_link_error` signal, as shown in the timing diagram below.

Figure 5-11: RX Error Reporting



RX Latency

The RX latency is defined as the time needed to fully transfer a 32-bit data in a lane (`jesd204_rx_link_datain*`) to the Avalon-ST interface (`jesd204_rx_dataout*`) when the `jesd204_rx_link_data_valid` signal equals to "1".

Table 5-17: RX Latency Associated with Different F and FRAMECLK_DIV Settings.

F	FRAMECLK_DIV	RX Latency
1	1	• Maximum 5 rxframe_clk period for byte 3 • Minimum 2 rxframe_clk period for byte 0
1	4	2 rxframe_clk period
2	1	• Maximum 3 rxframe_clk period for byte 2 and byte 3 • Minimum 2 rxframe_clk period for byte 0 and byte 1
2	2	2 txframe_clk period
4	—	2 txframe_clk period
8	—	2 txframe_clk period

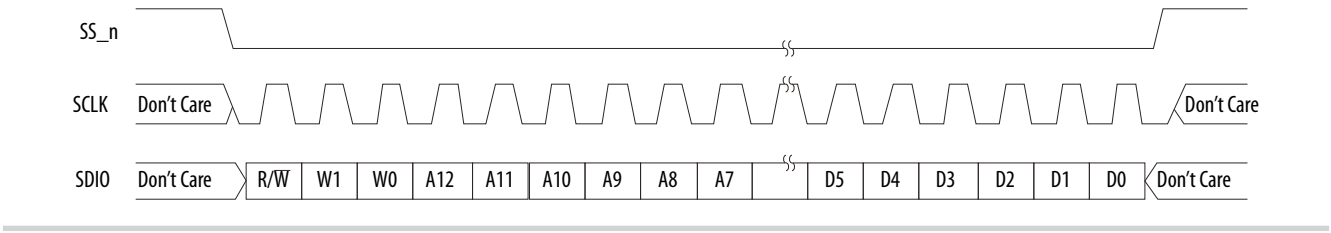
Serial Port Interface (SPI)

An external converter device with a SPI allows you to configure the converter for specific functions or operations through a structured register space provided inside the converter device. The SPI gives flexibility and customization, depending on the application. Addresses are accessed via the serial port and can be written to or read from the port. The memory is organized into bytes that can be further divided into fields.

The SPI communicates using two data lines, a control line, and a synchronization clock. A single SPI master can work with multiple slaves. The SPI core logic is synchronous to the clock input provided by the Avalon-MM interface. When configured as a master, the core divides the Avalon-MM clock to generate the SCLK output.

Figure 5-12: Serial Port Interface (24-bit) Timing Diagram

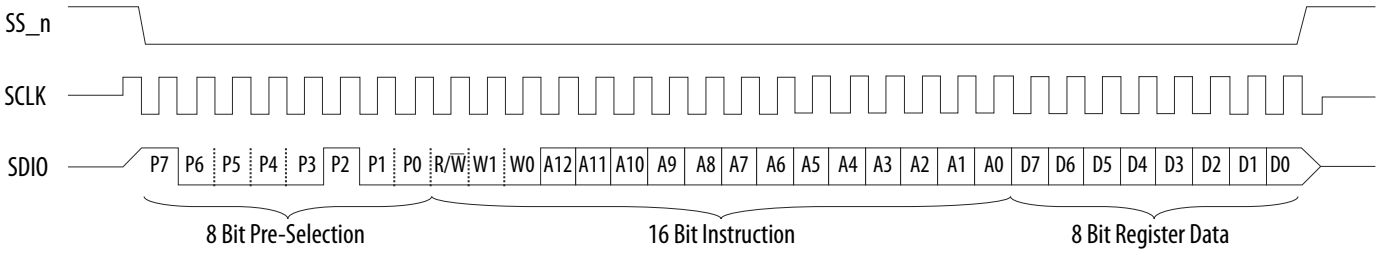
Figure shows the timing diagram of a 24-bit SPI transaction required by a typical external converter device.



The first 16 bits are instruction data. The first bit in the stream is the read or write indicator bit. This bit goes high to indicate a read request. W1 and W0 represent the number of data bytes to transfer for either a read or write process. For implementation simplicity, W1 and W0 are always set at 0 in this design example. The subsequent 13 bits represent the starting address of the data sent. The last 8 bits are register data.

For a 32-bit SPI transaction, each SPI programming cycle needs to be preceded with a preselection byte. The preselection byte is typically used to forward the SPI command to the right destination. **figure** shows the timing diagram of a 32-bit SPI transaction.

Figure 5-13: Serial Port Interface (32-bit) Timing Diagram



In this design example, the SPI core is configured as a 4-wire master protocol to control three independent SPI slaves—ADC, DAC, and clock devices. The width of the receive and transmit registers are configured at 32 bits. Data is sent in MSB-first mode in compliance with the converter device default power up mode. The SPI clock (`sclk`) rate is configured at a frequency of the SPI input clock rate divided by 5. If the SPI input clock rate is 100 MHz (in the `mgmt_clock` domain), the `sclk` rate is 20 MHz. If the external converter device's SPI interface is a 3-wire protocol without both MOSI (master output, slave input) and MISO (master input, slave output) lines but with a single DATAIO pin, you can use the ALTIOBUF Megafunction IP core (configured with bidirectional buffer) with the SPI master to convert the MOSI and MISO lines to a single DATAIO pin. The DATAIO pin can be dynamically reconfigured as MOSI by asserting the output enable (`oe`) signal or as MISO by deasserting the `oe` signal. For implementation simplicity, you can directly connect the master MOSI pin to the slave DATAIO pin if read transactions are not required.

Related Information

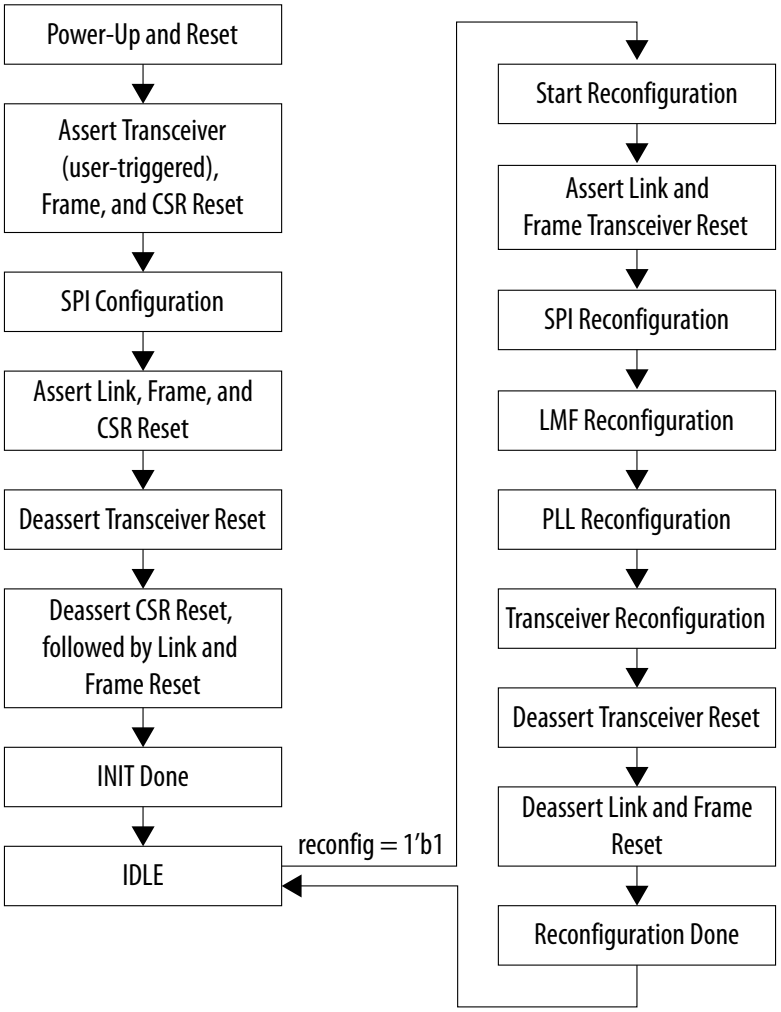
I/O Buffer (ALTIOBUF) Megafunction User Guide
More information about configuring the ALTIOBUF Megafunction IP Core.

Control Unit

The control unit has access to the CSR interface of the JESD204B IP core duplex base core, PLL reconfiguration, transceiver reconfiguration controller, and SPI master. The control unit also serves as a clock and reset unit (CRU) for the design example.

The control unit replaces the software-based Nios II processor to perform device configuration and initialization on the JESD204B duplex base core. This configuration and initialization process includes the transceivers, transport layer, pattern generator and checker, external converters (ADC/DAC), and clock devices over the SPI interface.

Figure 5-14: Control Unit Process Flow



Memory Block (ROM)

The control unit is a finite state machine (FSM) that works with multiple memory blocks (ROMs).

Each ROM holds the configuration data required to configure the external converter or clock devices for each SPI slave. A memory initialization file (MIF) contains the initial values for each address in the memory. Each memory block requires a separate file. The MIF can be created in the Quartus II software text editor tool.

Figure 5-15: Example of MIF Format and Content

```
-- MIF content for ADC

WIDTH=24;        -- the size of data in bits
DEPTH=8;         -- the size of memory in words
ADDRESS_RADIX=UNS;  -- the radix for address values
DATA_RADIX=BIN;    -- the radix for data values

CONTENT BEGIN
  0  :  000000000101111100010101;  -- write 0x15 to link control 1 register 0x5F to disable the lane
  1  :  000000000101111001000100;  -- write 0x44 to quick config register  0x5E for L=4, M=4
  2  :  000000000110010011000000;  -- write 0xC0 to DID register 0x64
  3  :  000000000110111000000011;  -- write 0x03 to parameter SCR/L register 0x6E to disable scrambler
  4  :  000000000111000000001111;  -- write 0x0F to parameter K register 0x70 for K=16 in base IP core
  5  :  000000000000110100000100;  -- write 0x04 to test mode register 0x0D for checkerboard test pattern
  6  :  000000000101111100010100;  -- write 0x14 to link control 1 register 0x5F to enable the lane
  7  :  111111111111111111111111;  -- indicates end of mif or end of programming sequence
END;
```

The initial values for each address and sequence is defined based on the requirement of the external converter and clock devices. The example above is based on 24-bit SPI write-only programming.

The last word must not be a valid data and must be set to all 1's to indicate the end of the MIF or programming sequence. This is because each converter device may have a different number of programmable registers and hence involves a different number of MIF words. In this design example, three ROMs are used by default for each external ADC, DAC, and clock devices. If either one of the device is not used, a single word MIF with all 1's can be created.

Note: The MIFs in this design example is an example for a particular converter device. You must define the MIF content based on the requirement of the external converter devices.

Finite State Machine (FSM)

The steps below describe the FSM flow:

1. Initialize the SPI:
   a. Perform a read transaction from the ROM on per word basis and write to the SPI master for SPI write transaction to the external SPI slave.
   b. Perform a read transaction from the next ROM and perform the same SPI write transaction to next SPI slave.
2. Initialize the JESD204B IP base core, transport layer, pattern generator, and pattern checker upon successful initialization of the transceiver.

System Parameters

Table 5-18: System Parameter Settings

This table lists the parameters exposed at the system level.

Parameter	Value [32]	Default	Description
LINK	1, 2	1	Number of JESD204B link. One link represent one JESD204B instance.
L	1, 2, 4, 8	2	Number of lanes per converter device.
M	1, 2, 4, 8	2	Number of converters per device.
F	1, 2, 4, 8	2	Number of octets per frame.
S	1, 2	1	Number of transmitted samples per converter per frame.
N	12–16	16	Number of conversion bits per converter.
N'	16	16	Number of transmitted bits per sample in the user data format.
F1_FRAMECLK_DIV	1, 4	4	The divider ratio on `frame_clk` when F = 1. The transport layer uses the post-divided `frame_clk`.
F2_FRAMECLK_DIV	1, 2	2	The divider ratio on `frame_clk` when F = 2. The transport layer uses the post-divided `frame_clk`.
POLYNOMIAL_LENGTH	7, 9, 15, 23, 31	7	Defines the polynomial length for the PRBS pattern generator and checker, which is also the equivalent number of stages for the shift register. • If PRBS-7 is required, set this parameter to 7. • If PRBS-9 is required, set this parameter to 9. • If PRBS-15 is required, set this parameter to 15. • If PRBS-23 is required, set this parameter to 23. • If PRBS-31 is required, set this parameter to 31. This parameter value must not be larger than N, which is the output data width of the PRBS pattern generator or converter resolution. If an N of 12-14 is required, PRBS-7 and PRBS-9 are the only feasible options. If an N of 15-16 is required, PRBS-7, PRBS-9, and PRBS-15 are the only feasible options.

[32] Values supported or demonstrated by this design example.

Parameter	Value [32]	Default	Description
FEEDBACK_TAP	6, 5, 14, 18, 28	6	Defines the feedback tap for the PRBS pattern generator and checker. This is an intermediate stage that is XOR-ed with the last stage to generate to next PRBS bit. • If PRBS-7 is required, set this parameter to 6. • If PRBS-9 is required, set this parameter to 5. • If PRBS-15 is required, set this parameter to 14. • If PRBS-23 is required, set this parameter to 18. • If PRBS-31 is required, set this parameter to 28.

Table below lists the configuration that this design example supports. However, the design example generated by the Qsys system is always fixed at a data rate of 6144 Mbps and a limited set of configuration as shown in the table below. If your setting in the Qsys parameter editor does not match one of the LMF and bonded mode parameter values in **Table**, the design example is generated with the default values of LMF = 124.

Table 5-19: Static and Dynamic Reconfiguration Parameter Values Supported

Mode	Link	L	M	F	Reference Clock	Frame Clock	Link Clock	F1_ FRAMECLK_ DIV	F2_ FRAMECLK_ DIV
Static									
Bonded/Non-bonded	2	1	1	2	153.6	153.6	153.6		2
Bonded/Non-bonded	1	1	1	4	153.6	153.6	153.6	1	
Bonded/Non-bonded	1	1	2	4	153.6	153.6	153.6	1	
Bonded/Non-bonded	1	1	4	8	153.6	76.8	153.6	1	
Bonded/Non-bonded	1	2	1	1	153.6	153.6	153.6	4	
Bonded/Non-bonded	1	2	1	2	153.6	153.6	153.6		2
Bonded/Non-bonded	1	2	1	4	153.6	153.6	153.6	1	
Bonded/Non-bonded	2	2	2	2	153.6	153.6	153.6		2
Bonded/Non-bonded	1	2	2	4	153.6	153.6	153.6	1	
Bonded/Non-bonded	1	2	4	4	153.6	153.6	153.6	1	
Bonded/Non-bonded	1	4	2	1	153.6	153.6	153.6	4	
Bonded/Non-bonded	1	4	2	2	153.6	153.6	153.6		2
Bonded/Non-bonded	1	4	4	2	153.6	153.6	153.6		2
Bonded/Non-bonded	1	4	4	4	153.6	153.6	153.6	1	
Bonded/Non-bonded	1	4	8	4	153.6	153.6	153.6	1	

[32] Values supported or demonstrated by this design example.

Mode	Link	L	M	F	Reference Clock	Frame Clock	Link Clock	F1_ FRAMECLK_ DIV	F2_ FRAMECLK_ DIV
Bonded/Non-bonded	1	8	1	1	307.2	153.6	153.6	4	
Bonded/Non-bonded	1	8	2	1	307.2	153.6	153.6	4	
Bonded/Non-bonded	1	8	4	1	307.2	153.6	153.6	4	
Bonded/Non-bonded	1	8	4	2	307.2	153.6	153.6		2
Dynamic Reconfiguration									
Non-bonded	2	2	2	2	153.6	153.6	153.6		2

The following figures show the datapath of single and multiple JESD204B links.

Figure 5-16: Datapath of A Single JESD204B Link

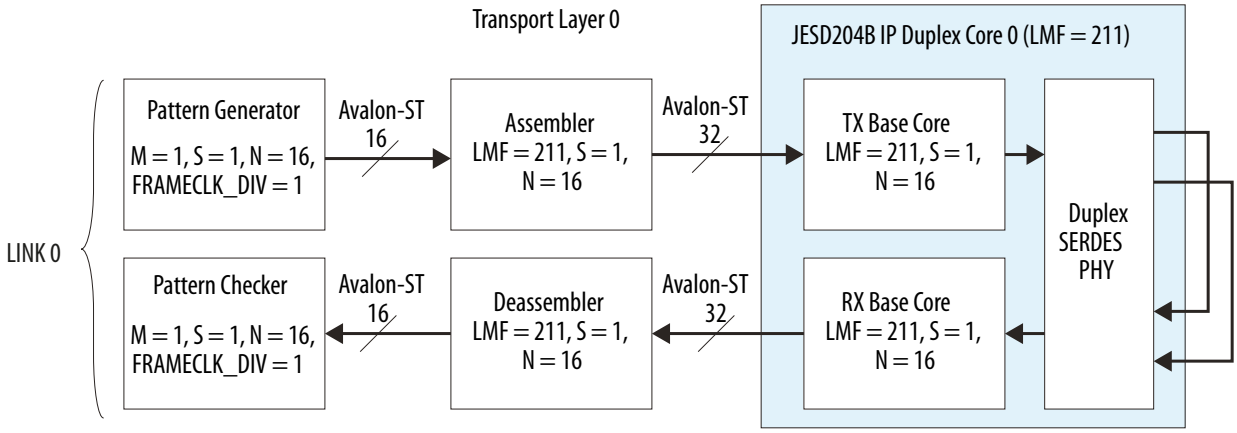
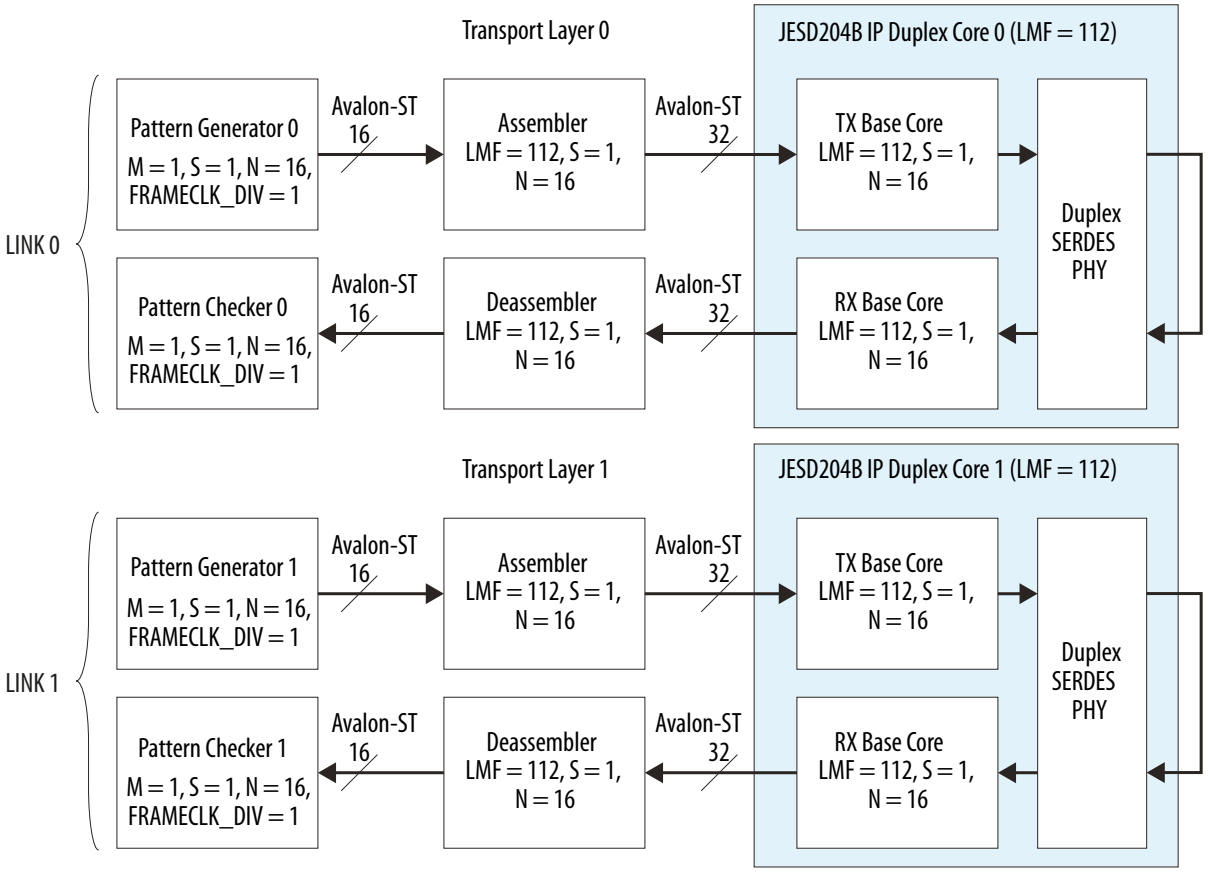


Figure 5-17: Datapath of Multiple JESD204B Links



Run-Time Reconfiguration

The JESD204B IP core supports run-time reconfiguration for the LMF and data rate settings. The design example only demonstrates the following set of configuration.

To generate the design example with run-time reconfiguration enabled, the LMF and bonding mode parameters must match the default value listed in the table below.

Table 5-20: Run-time Reconfiguration Demonstrated By The Design Example

Parameter	Default	Run-time Reconfiguration
LMF	222	112
FRAMECLK_DIV	2	2
Data Rate	6144 Mbps	3072 Mbps
Link Clock	153.6 MHz	76.8 MHz
Frame Clock	153.6 MHz	76.8 MHz
Bonding Mode	Non-bonded	Non-bonded

System Interface Signals

Table 5-21: Interface Signals

Signal	Clock Domain	Direction	Description
Clocks and Resets			
device_clk	—	Input	Device clock signal from the external converter or clock device.
mgmt_clk	—	Input	Management clock signal from the on-board 100 MHz oscillator.
frame_clk	—	Output	Internally generated clock. The Avalon-ST user data input must be synchronized to this clock domain for normal operation mode.
global_rst_n	mgmt_clk	Input	Global reset signal from the push button. This reset is an active low signal and the deassertion of this signal is synchronous to the rising-edge of mgmt_clk.
Signal	**Clock Domain**	**Direction**	**Description**
JESD204B			
tx_sysref[LINK-1:0]	link_clk	Input	TX *SYSREF* signal for JESD204B Subclass 1 implementation.
sync_n[LINK-1:0]	link_clk	Input	Indicates a TX SYNC_N from the receiver. This is an active low signal and is asserted 0 to indicate a synchronization request or error reporting.
mdev_sync_n[LINK-1:0]	link_clk	Input	Indicates a multidevice synchronization request at the TX path. Synchronize signal combination should be done externally and then input to the JESD204B IP core through this signal. In a single link instance where multidevice synchronization is not needed, you need to tie this signal to the dev_sync_n signal.
alldev_lane_aligned	link_clk	Input	Aligns all lanes for this device at the RX path. For multidevice synchronization, multiplex all the dev_lane_aligned signals before connecting to this signal pin. For single device support, connect the dev_lane_aligned signal back to this signal.

Signal	Clock Domain	Direction	Description
rx_sysref[LINK-1:0]	link_clk	Input	RX *SYSREF* signal for JESD204B Subclass 1 implementation.
tx_dev_sync_n[LINK-1:0]	link_clk	Output	Indicates a clean synchronization request at the TX path. This is an active low signal and is asserted 0 to indicate a synchronization request. The SYNC_N signal error reporting is masked out of this signal. This signal is also asserted during software-initiated synchronization.
dev_lane_aligned[LINK-1:0]	link_clk	Output	Indicates that all lanes for this device are aligned at the RX path.
rx_dev_sync_n[LINK-1:0]	link_clk	Output	Indicates a SYNC_N to the transmitter. This is an active low signal and is asserted 0 to indicate a synchronization request. Instead of reporting the link error through this signal, the JESD204B IP core uses the jesd204_rx_int signal to indicate an interrupt.
Signal	**Clock Domain**	**Direction**	**Description**
SPI			
miso	sclk	Input	Output data from a slave to the input of the master.
mosi	sclk	Output	Output data from the master to the inputs of the slaves.
sclk	mgmt_clk	Output	Clock driven by the master to slaves, to synchronize the data bits.
ss_n[2:0]	sclk	Output	Active low select signal driven by the master to individual slaves, to select the target slave. Defaults to 3 bits.
Signal	**Clock Domain**	**Direction**	**Description**
Serial Data and Control			
rx_serial_data[LINK*L-1:0]	—	Input	Differential high speed serial input data. The clock is recovered from the serial data stream.
tx_serial_data[LINK*L-1:0]	device_clk	Output	Differential high speed serial output data. The clock is embedded in the serial data stream.

Signal	Clock Domain	Direction	Description
rx_seriallpbken[LINK*L-1:0]	—	Input	Assert this signal to enable internal serial loopback in the duplex transceiver.
Signal	**Clock Domain**	**Direction**	**Description**
User Request Control			
reconfig	mgmt_clk	Input	Active high reconfiguration request. Set this signal to static 0 during compile time if run time reconfiguration is not required.
runtime_lmf	mgmt_clk	Input	Reconfigure the LMF value at run-time. This value must be stable prior to assertion of the reconfig signal. • 0—Downscale to the LMF value stored in MIF file. • 1— Upscale back to maximum LMF value. Assuming at compile time, the LMF configuration is 222, set this signal to 0 to scale down the LMF configuration to 112. Set this signal to 1 to scale up the LMF configuration back to 222.
runtime_datarate	mgmt_clk	Input	Reconfigure the data rate at run-time. This value must be stable prior to assertion of reconfig signal. • 0— Downscale to data rate setting stored in PLL, PHY, and clock MIF. • 1— Upscale back to maximum data rate setting stored in PLL, PHY, and clock MIF. Assuming the compile time data rate is 3.072 Gbps, set this signal to 0 to scale down the data rate to 1.536 Gbps. Set this signal to 1 to scale up the data rate back to 3.072 Gbps.
cu_busy	mgmt_clk	Output	Assert high to indicate that the control unit is busy. All reconfiguration input will be ignored when this signal is high.
Signal	**Clock Domain**	**Direction**	**Description**
Avalon- ST User Data			

Signal	Clock Domain	Direction	Description
avst_usr_din[(FRAMECLK_DIV*LINK*M*S*N)-1:0]	frame_clk	Input	TX data from the Avalon-ST source interface. The source arranges the data in a specific order, as illustrated in the cases below: Case 1: If F1/F2_FRAMECLK_DIV =1, LINK = 1, M = 1, S =1, N = 16: • avst_usr_din[15:0] Case 2: If F1/F2_FRAMECLK_DIV =1, LINK = 1, M = 2 (denoted by m0 and m1), S =1, N = 16: • avst_usr_din[15:0] = m0[15:0] • avst_usr_din[31:16] = m1[15:0] Case 3: If F1/F2_FRAMECLK_DIV =1, LINK = 2 (denoted by link0 and link1), M = 1, S =1, N = 16: • avst_usr_din[15:0] = link0 • avst_usr_din[31:16] = link1 Case 4: If F1/F2_FRAMECLK_DIV =1, LINK = 2 (denoted by link0 and link1), M = 2 (denoted by m0 and m1), S =1, N = 16: • avst_usr_din[15:0] = link0, m0[15:0] • avst_usr_din[31:16] = link0, m1[15:0] • avst_usr_din[47:32] = link1, m0[15:0] • avst_usr_din[63:48] = link1, m1[15:0]
avst_usr_din_valid	frame_clk	Input	Indicates whether the data from the Avalon-ST source interface to the transport layer is valid or invalid. • 0—data is invalid • 1—data is valid
avst_usr_din_ready	frame_clk	Output	Indicates that the transport layer is ready to accept data from the Avalon-ST source interface. • 0—transport layer is not ready to receive data • 1—transport layer is ready to receive data

Signal	Clock Domain	Direction	Description
avst_usr_dout[(FRAMECLK_DIV*LINK*M*S*N)-1:0]	frame_clk	Output	RX data to the Avalon-ST sink interface. The transport layer arranges the data in a specific order, as illustrated in the cases below: Case 1: If F1/F2_FRAMECLK_DIV =1, LINK = 1, M = 1, S =1, N = 16: • avst_usr_dout[15:0] Case 2: If F1/F2_FRAMECLK_DIV =1, LINK = 1, M = 2 (denoted by m0 and m1), S =1, N = 16: • avst_usr_dout[15:0] = m0[15:0] • avst_usr_dout[31:16] = m1[15:0] Case 3: If F1/F2_FRAMECLK_DIV =1, LINK = 2 (denoted by link0 and link1), M = 1, S =1, N = 16: • avst_usr_dout[15:0] = link0 • avst_usr_dout[31:16] = link1 Case 4: If F1/F2_FRAMECLK_DIV =1, LINK = 2 (denoted by link0 and link1), M = 2 (denoted by m0 and m1), S =1, N = 16: • avst_usr_dout[15:0] = link0, m0[15:0] • avst_usr_dout[31:16] = link0, m1[15:0] • avst_usr_dout[47:32] = link1, m0[15:0] • avst_usr_dout[63:48] = link1, m1[15:0]
avst_usr_dout_valid	frame_clk	Output	Indicates whether the data from the transport layer to the Avalon-ST sink interface is valid or invalid. • 0—data is invalid • 1—data is valid
avst_usr_dout_ready	frame_clk	Input	Indicates that the Avalon-ST sink interface is ready to accept data from the transport layer. • 0—Avalon-ST sink interface is not ready to receive data • 1—Avalon-ST sink interface is ready to receive data

Signal	Clock Domain	Direction	Description
test_mode[3:0]	frame_clk	Input	Specifies the operation mode. • 0000—Normal mode. The design example takes data from the Avalon-ST source. • 1000—Test mode. The design example generates alternate checkerboard data pattern. • 1001—Test mode. The design example generates ramp wave data pattern. • 1010—Test mode. The design example generates the PRBS data pattern. • Others—Reserved
Signal	**Clock Domain**	**Direction**	**Description**
Status			
rx_is_lockedtodata [LINK*L-1:0]	device_clk	Output	Asserted to indicate that the RX CDR PLL is locked to the RX data and the RX CDR has changed from LTR to LTD mode.
data_error [LINK-1:0]	frame_clk	Output	Asserted to indicate that the pattern checker has found a mismatch in the received data and the expected data. One error signal per pattern checker.
jesd204_tx_int[LINK-1:0]	link_clk	Output	Interrupt pin for the JESD204B IP core (TX). The interrupt signal is asserted when an error condition or synchronization request is detected.
jesd204_rx_int[LINK-1:0]	link_clk	Output	Interrupt pin for the JESD204B IP core (RX). The interrupt signal is asserted when an error condition or synchronization request is detected.

Example Feature: Dynamic Reconfiguration

The JESD204B IP core design example demonstrates dynamic (run-time) reconfiguration of either the LMF or data rate, at any one time.

Dynamic Reconfiguration Operation

The dynamic reconfiguration feature implements various reconfiguration controller modules such as PLL reconfiguration, Transceiver Reconfiguration Controller, SPI master, and JESD204B IP core Avalon-MM slave. These modules connect to the control unit through the Avalon-MM interface. You can control the reconfiguration using the `reconfig`, `runtime_lmf`, and `runtime_datarate` input ports exposed at control unit interface.

Figure 5-18: Dynamic Reconfiguration Block Diagram (For 28 nm Device Families—Stratix V and Arria V)

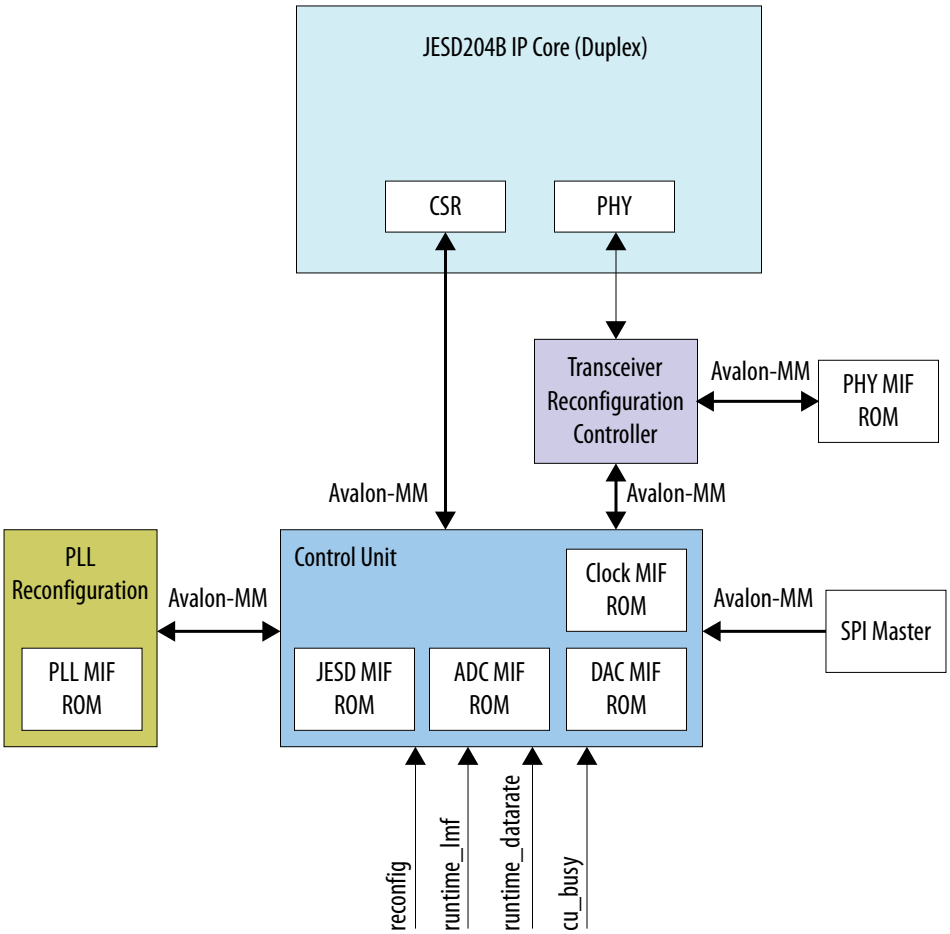
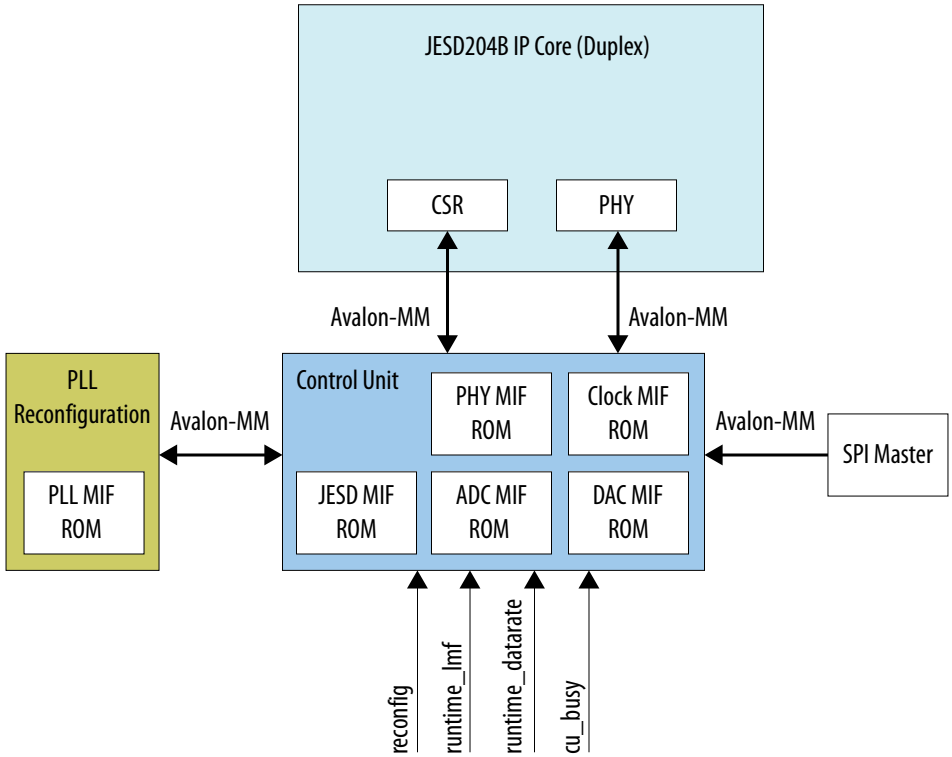


Figure 5-19: Dynamic Reconfiguration Block Diagram (For 20 nm Device Families—Arria 10)



The MIF ROM content for maximum and downscale configuration:

- PLL MIF ROM—contains the PLL counter, charge pump, and bandwidth setting.
- JESD MIF ROM—contains the LMF information.
- PHY MIF ROM—contains the transceiver channel and PLL setting.
- ADC MIF ROM—contains the ADC converter setting.
- DAC MIF ROM—contains the DAC converter setting.
- CLK MIF ROM—contains the device clock setting.

MIF ROM

You need to generate two MIF files for each reconfigurable IP core as shown in **Figure 5-19** or **Figure 5-20**, and merge them into a single MIF file for each IP core. The following section shows the MIF file format.

Core PLL

The MIF format is fixed by the PLL. You need to generate two PLLs with maximum and downscale setting to get these two MIF files. Then, merge the files into one (*core_pll.mif*). Only the PLL with maximum configuration is used in final compilation.

```
Maximum Configuration MIF
WIDTH=32;
DEPTH=92;

ADDRESS_RADIX=UNS;
```

```
DATA_RADIX=BIN;

CONTENT BEGIN
    0   :   00000000000000000000000000111110; -- START OF MIF
    1   :   00000000000000000000000000000100;
    2   :   00000000000000000000000100000001;
    3   :   00000000000000000000000000000011;
.
.
.
 42  :   00000000000000000000000000000010;
    43  :   00000000000000000000000000001000;
    44  :   00000000000000000000000001000000;
    45  :   00000000000000000000000000111111; -- END OF MIF



Downscale Configuration MIF

 46  :   00000000000000000000000000111110; -- START OF MIF
    47  :   00000000000000000000000000000100;
    48  :   00000000000000000000000100000001;
    49  :   00000000000000000000000000000011;
.
.
.
 88  :   00000000000000000000000000000010;
    89  :   00000000000000000000000000001000;
    90  :   00000000000000000000000001000000;
    91  :   00000000000000000000000000111111; -- END OF MIF
END;
```

PHY (Stratix V and Arria V)

The MIF format is fixed by the PHY. You need to generate two JESD204B IP cores with maximum and downscale setting. Then, compile each of the setting to get a total of four MIF files (two for TX PLL and two for channel MIF). Then, merge the files into one (*phy.mif*). Only the JESD204B IP cores with maximum configuration is used in final compilation.

```
Maximum TX PLL Configuration MIF
WIDTH=16;
DEPTH=186;

ADDRESS_RADIX=UNS;
DATA_RADIX=BIN;

CONTENT BEGIN
    0    :   0000000000100001; -- Start of MIF opcode (TX_PLL, 6144Mbps)
    1    :   0000000000100010;
.
.
.
 10   :   0011000000000000;
    11   :   0000000000011111; -- End of MIF opcode



Maximum Channel Configuration MIF

 12   :   0000000000100001; -- Start of MIF opcode (Channel, 6144Mbps)
    13   :   0000000000000010;
.
.
.
```

```
 [88..91]  :   0000000000000000;
    92   :   0000000000011111; -- End of MIF opcode


Downscale TX PLL Configuration MIF

 93   :   0000000000100001; -- Start of MIF opcode (TX_PLL, 3072Mbps)
    94   :   0000000000100010;
.
.
.
 103  :   0011000000000000;
    104  :   0000000000011111; -- End of MIF opcode


Downscale Channel Configuration MIF

 105  :   0000000000100001; -- Start of MIF opcode (Channel, 3072Mbps)
    106  :   0000000000000010;
.
.
.
 [181..184]  :   0000000000000000;
    185  :   0000000000011111; -- End of MIF opcode
END;
```

PHY (Arria 10)

The MIF format is fixed by the PHY. You need to generate two JESD204B IP cores with maximum and downscale setting. Then, compile each of the setting to get a total of four MIF files (two for TX PLL and two for channel MIF). Then, merge the files into two (*xcvr_atx_pll_combined.mif* and *xcvr_cdr_combined.mif*). Only the JESD204B IP cores with maximum configuration is used in final compilation.

xcvr_atx_pll_combined.mif

```
Maximum Configuration MIF

CONTENT BEGIN
    00    :   102FF71; -- Start of MIF
    01    :   103BF01;
    02    :   1047F04;
    03    :   1054700;
.
.
.
 10   :   11AFF00;
    11   :   11CE020;
    12   :   11DE020;
    13   :   3FFFFFF; -- End of MIF


Downscale Channel Configuration MIF

    14    :   102FF71; -- Start of MIF
    15    :   103BF01;
    16    :   1047F04;
    17    :   1054700;
.
.
.
 24   :   11AFF00;
    25   :   11CE020;
    26   :   11DE020;
```

```
    27   :   3FFFFFF; -- End of MIF
END;
```

xcvr_cdr_combined.mif

```
Maximum Configuration MIF

CONTENT BEGIN
    00    :   006DF02; -- Start of MIF
    01    :   007FF09;
    02    :   008FF04;
    03    :   00AFF01;
.
.
.
 76   :   173FF31;
    77   :   1741F0C;
    78   :   1753F13;
    79   :   3FFFFFF; -- End of MIF


Downscale Channel Configuration MIF

    7A    :   006DF02; -- Start of MIF
    7B    :   007FF09;
    7C    :   008FF04;
    7D    :   00AFF01;
.
.
.
 F0   :   173FF31;
    F1   :   1741F0C;
    F2   :   1753F13;
    F3   :   3FFFFFF; -- End of MIF
END;
```

JESD

The current JESD MIF contains only the LMF information. You need to manually code the MIF content in the following format.

```
Maximum Configuration MIF
WIDTH=16;
DEPTH=16;

ADDRESS_RADIX=UNS;
DATA_RADIX=BIN;

CONTENT BEGIN
    0    :   0000000000000001; -- L            (maximum config)
    1    :   0000000000000001; -- M
    2    :   0000000000000001; -- F
.
.
.
 3    :   1111111111111111; -- End of MIF
 [4..7]  :   0000000000000000;


Downscale Configuration MIF

 8    :   0000000000000000; -- L            (downscale config)
 9    :   0000000000000000; -- M
 10   :   0000000000000001; -- F
.
```

```
.
.
 11   :   1111111111111111; -- End of MIF
 [12..15]  :   0000000000000000;
END;
```

ADC/DAC/CLK

The content for ADC/DAC/CLK MIF is vendor-specific. The general format for the MIF is as shown below, with each section terminated by all 1's.

```
Maximum Configuration MIF
WIDTH=32;
DEPTH=128;

ADDRESS_RADIX=UNS;
DATA_RADIX=BIN;

CONTENT BEGIN
    0   :   10000100000000000001000001111100;  -- (Maximum Config)
    1   :   10000100000000000001010000000101;
    2   :   10000100000000000001011000000101;
    3   :   10000100000000000001110000000010;
.
.
.
 28  :   10000001000000001111111100000001;
    29  :   10000001000000000101111100010100;
    30  :   11111111111111111111111111111111;  -- End of MIF
    [31..63] :   00000000000000000000000000000000;


Downscale Configuration MIF

 64  :   10000100000000000001000001111100;   -- (downscale config)
 65  :   10000100000000000001010000000101;
 66  :   10000100000000000001011000000101;
 67  :   10000100000000000001110000000010;
.
.
.
 92  :   10000001000000001111111100000001;
 93  :   10000001000000000101111100010100;
 94  :   11111111111111111111111111111111;  -- End of MIF
 95..127]  :   00000000000000000000000000000000;
END;
```

Generating and Simulating the Design Example

To use the JESD204B IP core design example testbench, follow these steps:

1. Generate the design example simulation testbench. Refer to **Generating the Design Example Simulation Model** on page 5-52
2. Simulate the design example using simulator-specific scripts. Refer to **Simulating the JESD204B IP Core Design Example** on page 5-53

Generating the Design Example Simulation Model

After generating the IP core, generate the design example simulation testbench using the script (*gen_ed_sim_verilog.tcl* or *gen_ed_sim_vhdl*) located in the <*example_design_directory*>**/ed_sim** directory.

Note: For more information about the JESD204B design example testbench, refer to the **README_DESIGN_EXAMPLE.txt** file located in the <*example_design_directory*>**/ed_sim** folder.

To run the Tcl script using the Quartus II sofware, follow these steps:

1. Launch the Quartus II software.
2. On the View menu, click **Utility Windows** and select **Tcl Console**.
3. In the **Tcl Console**, type `cd <example_design_directory>/ed_sim` to go to the specified directory.
4. Type `source gen_ed_sim_verilog.tcl` (Verilog) or `source gen_ed_sim_vhdl.tcl` (VHDL) to generate the simulation files.

To run the Tcl script using the command line, follow these steps:

1. Obtain the Quartus II software resource.
2. Type `cd <example_design_directory>/ed_sim` to go to the specified directory.
3. Type `quartus_sh -t gen_ed_sim_verilog.tcl` (Verilog) or `quartus_sh -t gen_ed_sim_vhdl.tcl` (VHDL) to generate the simulation files.

Simulating the JESD204B IP Core Design Example

By default, the Quartus II software generates simulator-specific scripts containing commands to compile, elaborate, and simulate Altera IP models and simulation model library files. You can copy the commands into your simulation testbench script, or edit these files to add commands for compiling, elaborating, and simulating your design and testbench.

To simulate the design using the ModelSim-Altera SE/AE simulator, follow these steps:

1. Start the ModelSim-Altera simulator.
2. On the File menu, click **Change Directory** > **Select** <*example_design_directory*>**/ed_sim/testbench/mentor**.
3. On the File menu, click **Load** > **Macro file**. Select **run_tb_top.tcl**. This file compiles the design and runs the simulation automatically, providing a pass/fail indication on completion.

To simulate the design using the VCS MX simulator (in Linux), follow these steps:

1. Start the VCS MX simulator.
2. On the File menu, click **Change Directory** > **Select** <*example_design_directory*>**/ed_sim/testbench/synopsys/vcsmx**.
3. Run **run_tb_top.sh**. This file compiles the design and runs the simulation automatically, providing a pass/fail indication on completion.

To simulate the design using the Aldec Riviera-PRO simulator, follow these steps:

1. Start the Aldec Riviera-PRO simulator.
2. On the File menu, click **Change Directory** > **Select** <*example_design_directory*>**/ed_sim/testbench/aldec**.
3. On the Tools menu, click **Execute Macro**. Select **run_tb_top.tcl**. This file compiles the design and runs the simulation automatically, providing a pass/fail indication on completion.

Generating the Design Example For Compilation

Use the *gen_quartus_synth.tcl* script to generate the JESD204B design example for compilation.

Note: If you use the Quartus II Tcl console to generate the *gen_quartus_synth.tcl* script, close all Quartus II project before you start generating.

To run the Tcl script using the Quartus II sofware, follow these steps:

1. Launch the Quartus II software.
2. On the View menu, click **Utility Windows** and select **Tcl Console**.
3. In the **Tcl Console**, type `cd <example_design_directory>/ed_synth` to go to the specified directory.
4. Type `source gen_quartus_synth.tcl` to generate the JESD204B design example for compilation.

To run the Tcl script using the command line, follow these steps:

1. Obtain the Quartus II software resource.
2. Type `cd <example_design_directory>/ed_synth` to go to the specified directory.
3. Type `quartus_sh -t gen_ed_quartus_synth.tcl` to generate the JESD204B design example for compilation.

Compiling the JESD204B IP Core Design Example

You can use the generated **.qip** file to include relevant files into your project. Generate the Quartus II synthesis compilation files by running the script (*gen_quartus_synth.tcl*) located in the <*example_design_directory*>**/ed_synth/** directory.

Note: If you use the Quartus II Tcl console to generate the *gen_quartus_synth.tcl* script, close all Quartus II project before you start generating.

To compile your design using the Quartus II software , follow these steps:

1. Launch the Quartus II software.
2. On the File menu, click **Open Project** > **Select** <*example_design_directory*>**/ed_synth/example_design/**.
3. Select **jesd204b_ed.qpf**. (33)
4. On the Processing menu, click **Start Compilation**.

   At the end of the compilation, the Quartus II software provides a pass/fail indication.

(33) This is the default quartus project file that the Quartus II software automatically generates. You can edit this file and the **.qsf** file according to your design preference.

JESD204B IP Core Debug Guidelines

6

2014.12.15

UG-01142

This section lists some guidelines to assist you in debugging JESD204B link issues. Apart from applying general board level hardware troubleshooting technique like checking the power supply, external clock source, physical damage on components, a fundamental understanding of the JESD204B subsystem operation is important.

Related Information

- **Clocking Scheme** on page 6-1
- **JESD204B Parameters** on page 6-1
- **SPI Programming** on page 6-2
- **Converter and FPGA Operating Conditions** on page 6-2
- **Signal Polarity and FPGA Pin Assignment** on page 6-2
- **Debugging JESD204B Link Using SignalTap II and System Console** on page 6-3

Clocking Scheme

To verifying the clocking scheme, follow these steps:

1. Check that the frame and link clock frequency settings are correct in the Altera PLL IP core. For the design example, the frame clock is assigned to *outclk0* and link clock is assigned to *outclk1*.
2. Check the device clock frequency at the FPGA and converter.
3. For Subclass 1, check the *SYSREF* pulse frequency.
4. Check the clock frequency management. For the design example, using Stratix V and Arria V devices, this frequency is 100 MHz.

JESD204B Parameters

The parameters in both the FPGA and ADC should be set to the same values. For example, when you set K = 32 on the FPGA, set the converter's K value to 32 as well. Scrambling does not affect the link initialization in the CGS and ILAS phases but in the user data phase. When scrambling is enabled on the ADC, the FPGA descrambling option has to be turned on using the "Enable scramble (SCR)" option in the JESD204B IP core Qsys parameter editor. When scrambling is enabled on the FPGA, the DAC descrambling has to be turned on too.

Check these items:

- Turn off the scrambler and descrambler options as needed.
- Use single lane configuration and K = 32 value to isolate multiple lane alignment issue.
- Use Subclass 0 mode to isolate *SYSREF* related issues like setup or hold time and frequency of *SYSREF* pulse.

SPI Programming

The SPI interface configures the converter. Hence, it is important to check the SPI programming sequence and register bit settings for the converter. If you use the MIF to store the SPI register settings of the converter, mistakes may occur when modifying the MIF, for example, setting a certain bit to "1" instead of "0", missing or extra bits in a MIF content row.

Check these items:

- For example, in the ADI AD9250 converter, Altera recommends that you first perform register bit setting for the scramble (SCR) or lane (L) register at address 0x6E before setting the quick configuration register at address 0x5E.
- Determine that each row of the MIF has the same number of bits as the data width of the ROM that stores the MIF.

Converter and FPGA Operating Conditions

The transceiver channels at the converter and FPGA are bounded by minimum and maximum data rate requirements. Always check the most updated device data sheet for this info. For example, the Arria V GT device has a minimum data rate of 611 Mbps.

Ensure that the sampling rate of the converter is within the minimum and maximum requirements. For example, the ADC AD9250 has a minimum sampling rate of 40 Msps. For L = 2, M = 1 configuration, the minimum data rate of this ADC is calculated this way:

$$Minimum\ AD9250\ data\ rate = \frac{40 * 1 * 16 * \frac{10}{8}}{2} = 400\ Mbps$$

The minimum data rate for the JESD204B link is effectively 611 Mbps.

Check these items:

- Reduce the data rate or sampling clock frequency if your targeted operating requirement does not work.
- Verify the minimum and maximum data rate requirements in the device manufacturer's data sheet.

Signal Polarity and FPGA Pin Assignment

Verify that the transceiver channel pin assignments—`SYNC_N` and *SYSREF* (for Subclass 1 only)—device clock, and SPI interface are correct. Also verify the signal polarity of the differential pairs like `SYNC_N` and transceiver channels are correct.

Check these items:

- Review the schematic and board layout file to determine the polarity of the physical pin connection.
- Use assignment editor and pin planner to check the pin assignment and I/O standard for each pin.
- Use RTL viewer in the Quartus II software to verify that the top level port are connected to the lower level module that you instantiate.

Debugging JESD204B Link Using SignalTap II and System Console

The SignalTap II provides dynamic view of signals while the system console provides access to the JESD204B IP core register sets through the Avalon-MM interfaces.

The SignalTap II and system console are very useful tools in debugging the JESD204B link related issues. To use the system console, your design must contain a Qsys subsystem with the JTAG-to-Avalon-MM Master bridge component and the Merlin slave translator ports that connect to the JESD204B IP core Avalon-MM interface.

PHY Layer

Verify the RX PHY status through these signals in the <*ip_variant_name*>**.v**:

- rx_is_lockedtodata
- rx_analogreset
- rx_digitalreset
- rx_cal_busy

Verify the TX PHY status through these signals in the <*ip_variant_name*>**.v**:

- pll_locked
- pll_powerdown
- tx_analogreset
- tx_digitalreset
- tx_cal_busy

Verify the RX_TX PHY status through these signals in the <*ip_variant_name*>**.v**:

- rx_is_lockedtodata
- rx_analogreset
- rx_digitalreset
- rx_cal_busy
- rx_seriallpbken
- pll_locked
- pll_powerdown
- tx_analogreset
- tx_digitalreset
- tx_cal_busy

Use the `rxphy_clk[0]` or `txphy_clk[0]` signal as sampling clock for the SignalTap II.

For a normal operation of the JESD204B RX path, the `rx_is_lockedtodata` bit for each lane should be "1" while the `rx_cal_busy`, `rx_analogreset`, and `rx_digitalreset` bit for each lane should be "0".

For a normal operation of the JESD204B TX path, the `pll_locked` bit for each lane should be "1" while the `tx_cal_busy`, `pll_powerdown`, `tx_analogreset`, and `tx_digitalreset` bit for each lane should be "0".

Measure the `rxphy_clk` or `txphy_clk` frequency by connecting the clock to the *CLKOUT* pin on the FPGA. The frequency should be the same as link clock frequency.

Link Layer

Verify the RX PHY-link layer interface operation through these signals in the <*ip_variant_name*>**_inst_phy.v**:

- jesd204_rx_pcs_data
- jesd204_rx_pcs_data_valid
- jesd204_rx_pcs_kchar_data
- jesd204_rx_pcs_errdetect
- jesd204_rx_pcs_disperr

Verify the RX link layer operation through these signals in the <*ip_variant_name*>**.v**:

- jesd204_rx_avs_rst_n
- rxlink_rst_n_reset_n
- rx_sysref (for Subclass 1 only)
- rx_dev_sync_n
- jesd204_rx_int
- alldev_lane_aligned
- dev_lane_aligned
- rx_somf

Use the `rxlink_clk` signal as the sampling clock.

Verify the TX PHY-link layer interface operation through these signals in the <*ip_variant_name*>**_inst_phy.v**:

- jesd204_tx_pcs_data
- jesd204_rx_pcs_kchar_data

Verify the TX link layer operation through these signals in the <*ip_variant_name*>**.v**:

- jesd204_tx_avs_rst_n
- txlink_rst_n_reset_n
- tx_sysref (for Subclass 1 only)
- sync_n
- tx_dev_sync_n
- mdev_sync_n
- jesd204_tx_int

Altera recommends that you verify the JESD204B functionality by accessing the DAC SPI registers or any debug feature provided by the DAC manufacturer.

Figure 6-1: JESD204B Link Initialization

This is a SignalTap II image during the JESD204B link initialization. The JESD204B link has two transceiver channels (L = 2).

Description of the timing diagram:

a. The JESD204B link is out of reset.
b. The RX CDR is locked and PCS outputs valid characters to link layer.
c. No running disparity error and 8b/10b block within PCS successfully decodes the incoming characters.
d. The ADC transmits /K/ character or BC hexadecimal number to the FPGA, which starts the CGS phase.
e. Upon receiving 4 consecutive /K/ characters, the link layer deasserts the `rx_dev_sync_n` signal.
f. The JESD204B link transition from CGS to ILAS phase when ADC transmit /R/ or 1C hexadecimal after /K/ character.
g. Start of 2nd multi-frame in ILAS phase. 2nd multi-frame contains the JESD204B link configuration data.
h. Start of 3rd multi-frame.
i. Start of 4th multi-frame.
j. Device lanes alignment is achieved. In this example, there is only one device, the `dev_lane_aligned` connects to `alldev_lane_aligned` and both signals are asserted together.
k. Start of user data phase where user data is streamed through the JESD204B link

Transport Layer

Verify the RX transport layer operation using these signals in the **altera_jesd204_transport_rx_top.sv**:

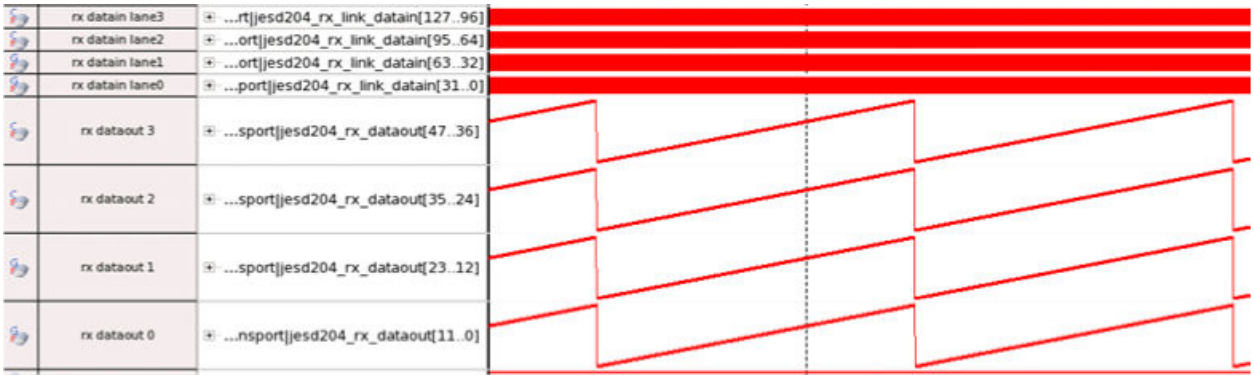
- jesd204_rx_dataout
- jesd204_rx_data_valid
- jesd204_rx_data_ready
- jesd204_rx_link_data_ready
- jesd204_rx_link_error
- rxframe_rst_n

Use the `rxframe_clk` signal as the sampling clock.

For normal operation, the `jesd204_rx_data_valid`, `jesd204_rx_data_ready`, and `jesd204_rx_link_data_ready` signals should be asserted while the `jesd204_rx_link_error` should be deasserted. You can view the ramp or sine wave test pattern on the `jesd204_rx_dataout` bus.

Figure 6-2: Ramp Pattern on the `jesd204_rx_dataout` Bus

This is a SignalTap II image during the JESD204B user data phase with ramp pattern transmitted from the ADC.



Verify the TX transport layer operation using these signals in the **altera_jesd204_transport_tx_top.sv**:

- txframe_rst_n
- jesd204_tx_datain
- jesd204_tx_data_valid
- jesd204_tx_data_ready
- jesd204_tx_link_early_ready
- jesd204_tx_link_data_valid
- jesd204_tx_link_error

Use the `txframe_clk` signal as the sampling clock.

For normal operation, the `jesd204_tx_data_valid`, `jesd204_tx_data_ready`, `jesd204_tx_link_early_ready`, and `jesd204_tx_link_data_valid` signals should be asserted while the `jesd204_tx_link_error` should be deasserted. You can verify the user data arrangement (shown in the data mapping tables in the **TX Path Data Remapping** on page 5-18) by referring to the `jesd204_tx_datain` bus.

Related Information

- **AN 696: Using the JESD204B MegaCore Function in Arria V Devices**
  More information about the performance and interoperability of the JESD204B IP core.
- **Altera Transceiver PHY IP Core User Guide**
  More information about the transceiver PHY signals.

Additional Information 7

2014.12.15

UG-01142 Subscribe Send Feedback

Additional information about the document and Altera.

JESD204B IP Core Document Revision History

Date	Version	Changes

December 2014	2014.12.15	• Updated the JESD204B IP Core FPGA Performance table with the data rate range.
		• Updated the JESD204B IP Core FPGA Resource Utilization table.
		• Updated the JESD204B IP Core Parameters table with the following changes:
		  • Revised the parameter name of **Enable PLL/CDR Dynamic Reconfiguration** to **Enable Transceiver Dynamic Reconfiguration**.
		  • Added information for a new parameter—**Enable Altera Debug Master Endpoint**.
		  • Added details about the rule check for parameter N' value.
		• Added a new topic—**Integrating the JESD204B IP core in Qsys** on page 3-10.
		• Updated **Figure 7-1**, **Figure 7-3**, and **Figure 7-4**.
		• Added a new table—Register Access Type Convention—to describe the access type for the IP core registers.
		• Added new signals description for `jesd204_tx_controlout` and `jesd204_rx_controlout`.
		• Added **CONTROL_BUS_WIDTH** parameter and description for the assembler and deassembler.
		• Added information on how to run the Tcl script using the Quartus II sofware before compiling the design example.
		• Updated the section on **Debugging JESD204B Link Using SignalTap II and System Console** on page 6-3 with verification information for TX PHY-link layer interface, TX link layer, and TX transport layer operations.

Date	Version	Changes

June 2014	2014.06.30	• Updated **Figure 2-1** to show a typical system application.
		• Updated the list of core key features.
		• Updated the Performance and Resource utilization values.
		• Updated the Getting Started chapter to reflect the new IP Catalog and parameter editor.
		• Added the following new sections to further describe the JESD204B IP core features:
		  • **Channel Bonding**
		  • **Datapath Modes**
		  • **IP Core Variation**
		  • **JESD204B IP Core Testbench**
		  • **JESD204B IP Core Design Considerations**
		  • **TX Data Link Layer**
		  • **TX PHY Layer**
		  • **RX Data Link Layer**
		  • **RX PHY Layer**
		  • **Operation**
		  • **Example Feature: Dynamic Reconfiguration**
		  • **JESD204B IP Core Debug Guidelines**
		• Updated the Clocking scheme section.
		• Added new transceiver signals that is supported in Arria 10 devices.
		• Updated the Transport Layer section.
		• Added run-time reconfiguration parameter values in the **System Parameters** section.
		• Updated the file directory names.

November 2013	2013.11.04	Initial release.

How to Contact Altera

Table 7-1: Altera Contact Information

Contact[34]	Contact Method	Address
Technical support	Website	www.altera.com/support
Technical training	Website	www.altera.com/training
	Email	custrain@altera.com
Product literature	Website	www.altera.com/literature

[34] You can also contact your local Altera sales office or sales representative.

Contact[34]		Contact Method	Address
Nontechnical support	General	Email	nacomp@altera.com
	Software licensing	Email	authorization@altera.com

Related Information

- **www.altera.com/support**
- **www.altera.com/training**
- **www.altera.com/literature**

[34] You can also contact your local Altera sales office or sales representative.