Lattice
Semiconductor Corporation

LatticeCORE™

JESD204A IP Core User's Guide

Table of Contents

Introduction

JEDEC Standard No. 204A (JESD204A) describes a serialized interface between data converters and logic devices. It contains the information necessary to allow designers to implement devices which can communicate with other devices that are compliant with the standard. Lattice's JESD204A IP Core offerings support both an Rx core (ADC to FPGA direction) and/or a Tx core (FPGA to DAC direction). The Rx and Tx cores can each be generated separately and with different parameters.

Quick Facts

Table 1-1 gives quick facts about the JESD204A IP Cor for LatticeECP3™ devices.

Table 1-1. Quick Facts

		JESD204A IP Configuration
		2-lane configuration (F=3, L=2, K=9)
Core Requirements	FPGA Families Supported	LatticeECP3
Resources Utilization	Target Device	LFE3-70E
	Data Path Width	16 bits per lane, 32 bits total for 2 lanes
	LUTs	Rx: 1012 / Tx: 483
	sysMEM™ EBRs	Rx: 0 / Tx: 0
	Registers	Rx: 761 / Tx: 342
Design Tool Support	Lattice Implementation	Lattice Diamond™ 1.1 or ispLEVER® 8.1SP01
	Synthesis	Synopsys® Synplify™ Pro for D-2010.03L-SP1
	Simulation	Aldec® Active-HDL™ 8.2 Lattice Edition
		Mentor Graphics ModelSim™ SE 6.3F

Features

• Compliant with JEDEC Standard No. 204A (JESD204A) April 2008

• Rx core performs lane alignment buffering / detection / monitoring and correction

• Rx core performs frame alignment detection / monitoring and octet reconstruction

• Rx core performs user-enabled descrambling

• Rx core recovers link configuration parameters during initial lane synchronization and compares them to user-selected parameters to generate a configuration mismatch error

• Tx core performs user-enabled scrambling

• Tx core generates initial lane alignment sequence

• Tx core performs alignment character generation

• Tx core sources link configuration data with user selected parameter values during initial lane synchronization sequence

What Is Not Supported

The core does not support the following features:

- Configuration of Rx core by link configuration parameters

- Octet to frame stream conversion in Rx core

- Frame to octet stream conversion in Tx core

- Verification of transport layer test samples within the Rx core

Conventions

The nomenclature used in this document is based on the Verilog language. This includes radix indications and logical operators.

Data Ordering and Data Types

- The JESD204A IP core operates on individual bytes only, so there is no byte ordering defined for the core. However, to minimize the FPGA fabric clock frequency, the data path within the IP cores is two bytes wide. The byte in the lower 8 bits of a 16-bit pair is the "low" byte, and the byte in the upper 8 bits is the "high" byte. On the interface to user's logic, each lane is a 16 bit interface. Within each 16-bit pair of bytes, the high byte is the first byte transmitted or received, and the low byte is the second byte transmitted or received. On the interface between the IP cores and the PCS/serdes, the low byte is the first byte of the pair, and the high byte is the second byte of the pair.

- The most significant bit within data bytes is bit 7.

Signal Names

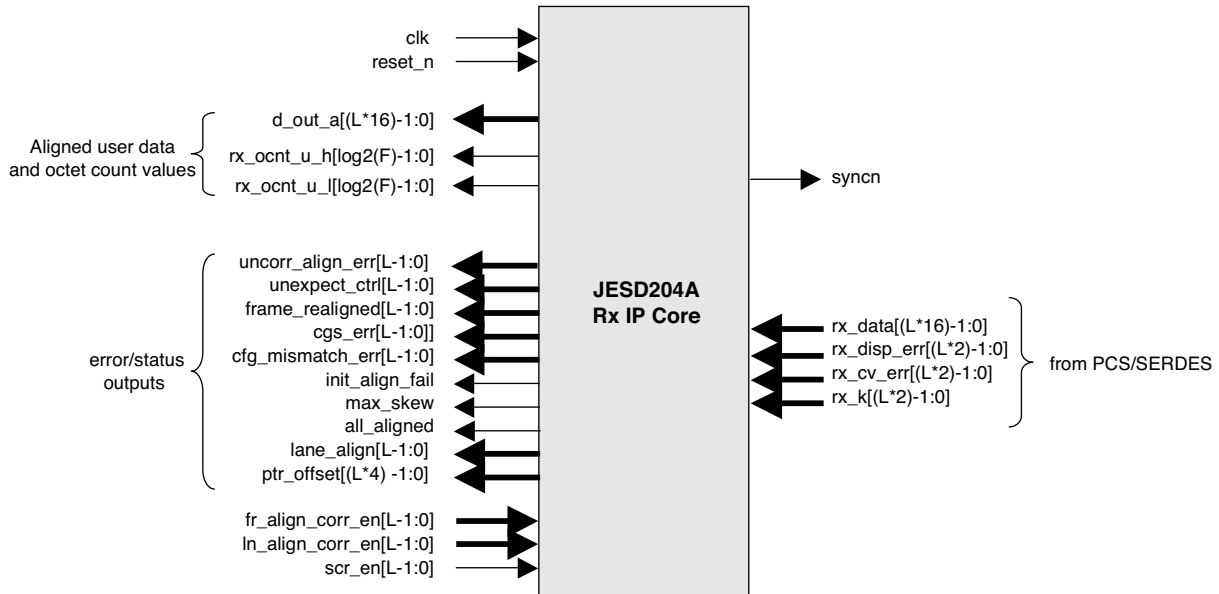- Signal names that end with "_n" are active low.

Functional Description

The JESD204A IP core implements the functionality described in the JEDEC standard. The following sections describe the Rx and Tx cores.

Rx Core

Figure 2-1 shows the ports on the Rx IP core.

Figure 2-1. JESD204A Rx IP Core I/O

Referring to Figure 2-2, in the Rx direction, serial I/F data is received by the logic device (FPGA) over one or more serial I/F bit-stream lanes. The number of lanes is equal to the parameter L. The Rx core finds the framing information for each lane and aligns all lanes to a common frame boundary. When the Rx core loses synchronization, it sets the SYNC signal active which tells the Tx Block to send a new initialization sequence. Once the individual lanes are aligned, the Rx core sources aligned sample stream data on its outputs to the user interface.

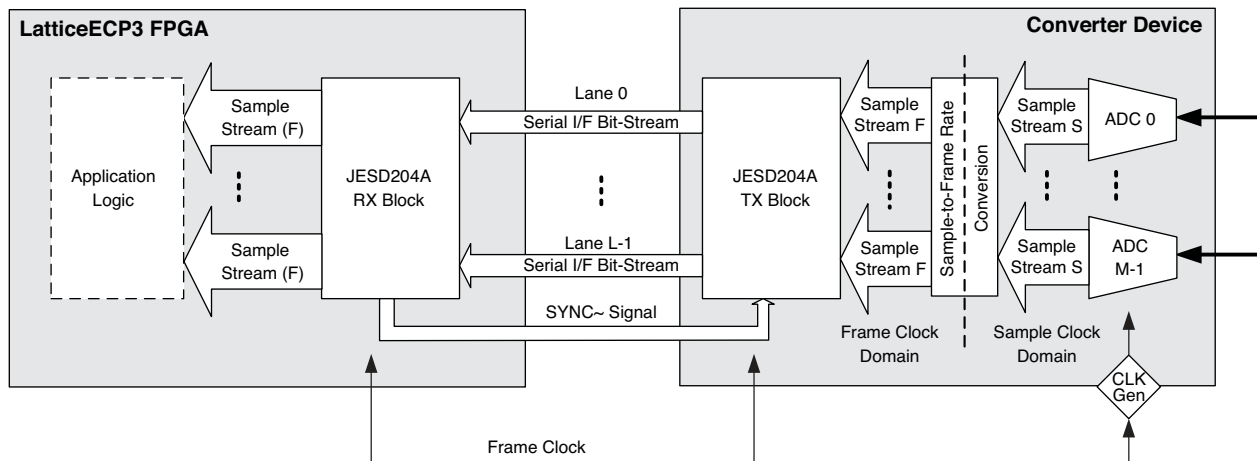*Figure 2-2. JESD204A Single Device Application*

Figure 2-3 shows the functions performed internal to the Rx core. The Serial Bit Streams of the JESD204A links go through the PCS/SERDES block which performs 8b/10b decoding. The data is then passed to the Rx IP core which performs the synchronization and alignment functions, followed by descrambling. The IP core then sources the aligned data to the FPGA fabric where additional logic can perform the octet to frame stream conversion.

Figure 2-3. JESD204A Receiver Functions
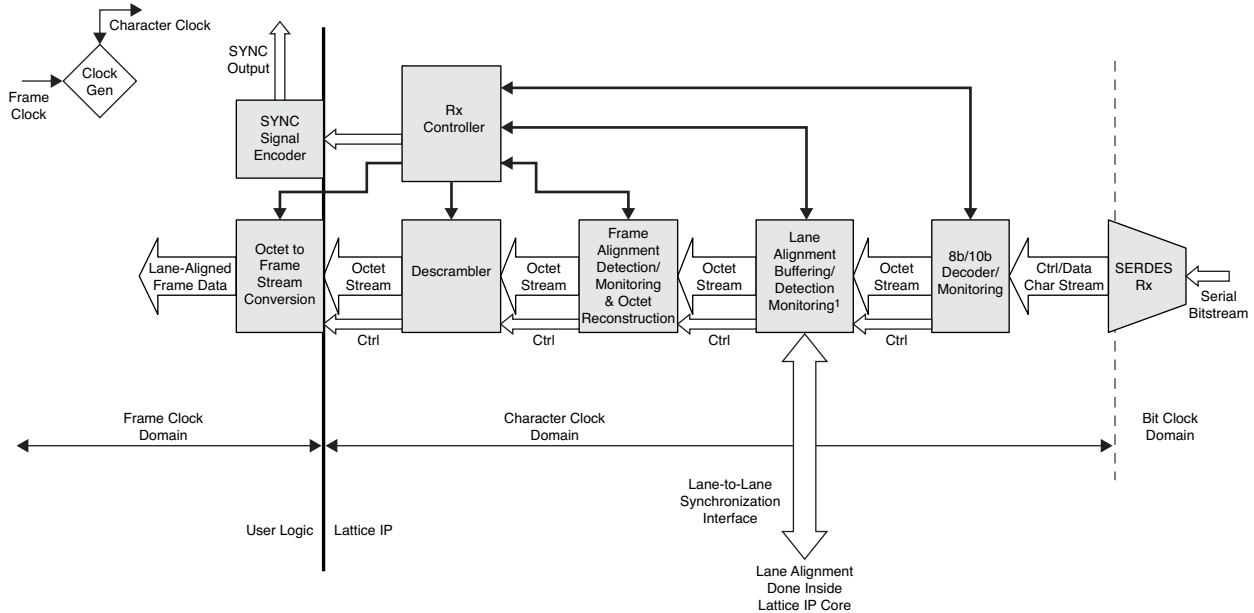
Figure 2-4 shows the functions performed by the Rx IP core for a 2-lane configuration, although the following discussion applies equally well to configurations with any number of lanes. As shown in the figure, two lanes of serial data are received by the PCS/SERDES which performs 8b/10b decoding and word alignment. The data is passed as two 18-bit buses to the Rx IP core which first performs code group synchronization as specified by Figure 24 of the JESD204A Standard. Each of the 18-bit buses per lane contains two 8-bit data words and two control signals. The byte in the lower bit positions of the 2-byte wide bus was received on the serial link first.

Figure 2-4. JESD204A Rx IP Core Block Diagram

- Code group synchronization state machine
- Initial frame sync. state machine
- Initial lane alignment state machine
- Lane alignment/correction monitoring
- Frame alignment/correction monitoring
- MCC – multi-frame character counter
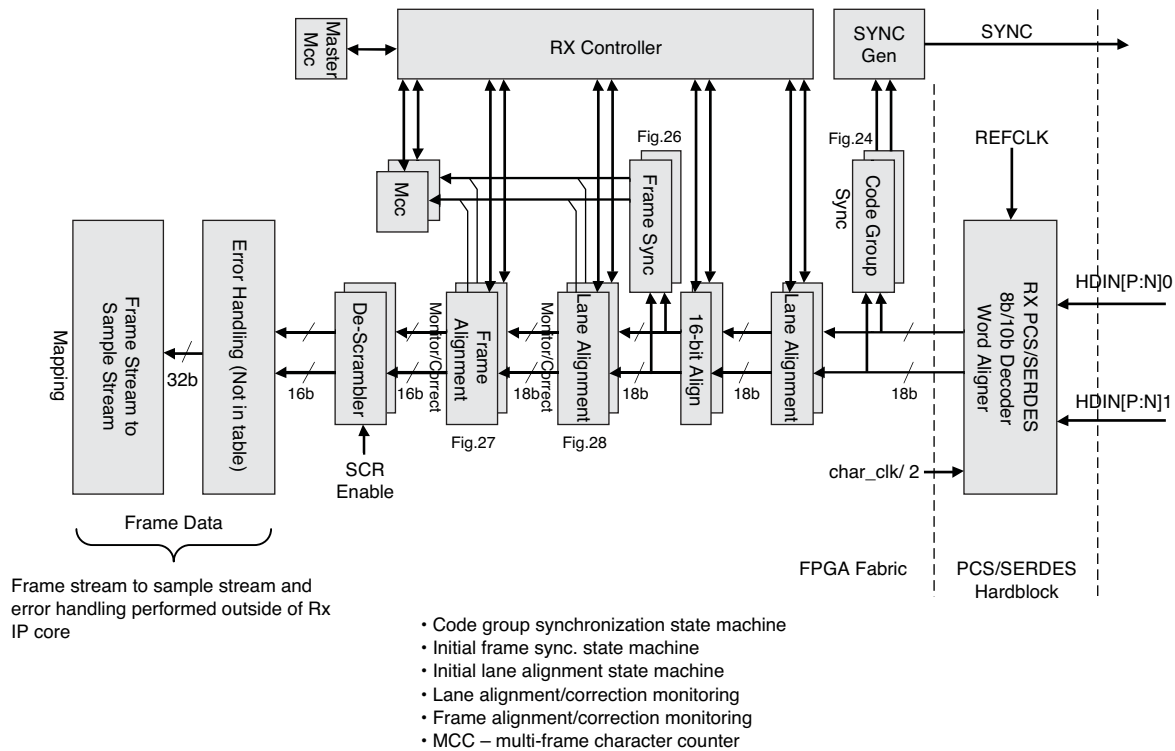
When a loss of code group synchronization is detected on any lane, a sync_request is passed to the SYNC_gen block which asserts the SYNC output to the transmitting DAC device(s).

Next, the lane alignment and 16-bit align modules perform lane alignment. During initialization, the start of the frame is aligned to the high-byte position. After initialization, these modules will realign the frames and multi-frames across all lanes to remove any data skew between lanes.

The IP core then performs frame synchronization as specified in Figure 26 of the Standard document. The alternative state machine of Figure 26 can be used instead of that shown in Figure 25 because the Rx IP core in the FPGA is assumed to be the only receiver device.

Next the lane alignment monitoring and correction function is performed as specified in Figure 28 of the Standard document, followed by frame alignment monitoring and correction as shown in Figure 27 of the Standard. These two modules replace the K28.3 and K28.7 control characters with the original user data.

Control signals from the Frame sync, Lane Alignment Monitoring/Correction, and Frame Alignment Monitoring/Correction blocks all feed into the Multi-frame character clock (Mcc) circuits which generate an octet count value for both the high and low octets for each lane. In addition, there is an Rx Controller module which monitors the states of the alignment circuits as well as the Mcc circuits for each lane. During initialization, the Rx Controller resets a Master Mcc circuit and centers all of the lane alignment buffers based on the skew detected across all of the lanes. If the skew across all lanes exceeds that for which the lane alignment buffers can compensate, then the output signal max_skew is asserted.

After initialization, the Rx Controller uses the Master Mcc to determine when an individual lane has lost alignment and must be corrected. This is done by the realign module which is part of the Rx controller. This module sends a signal to the lane which is out of alignment to perform either a positive or negative adjustment to move it back into alignment with the other lanes. Over time, random movements on individual lanes can eventually walk the pointers of the lane alignment buffers to the extreme edge of what they can compensate for. When this happens, further movement of the alignment between lanes cannot be compensated for and an uncorrectable alignment error

occurs. The signal uncorr_align_err is asserted to indicate this condition. At this point the system must force an initialization sequence. This can be done by forcing the SYNC signal to go active when a low-to-high transition is detected on uncorr_align_err.

The aligned data is passed through a descrambler which can be enabled or disabled by a control input (scr_en) to the Rx core. Following the descrambler, the data is output from the Rx core as a single bus having L*16 bits. Data for each lane occupies 16 bits of the overall bus. The upper 8 bits out of each group of 16 (high-byte) are the first byte of the sequence, the lower 8 bits of 16 (low-byte) are the next byte. The signals rx_ocnt_u_h and rx_ocnt_u_l are octet counters which count up from 0 to (F-1) and indicate which octet of the frame each byte represents. rx_ocnt_u_l is associated with the lower byte (for example, bits 7:0 for lane 0, and bits 23:16 for lane 1) and rx_ocnt_u_h is associated with the higher byte (bits 15:8 for lane 0 and 31:24 for lane 1) of each 16-bit group.

As an example, Table 2-1 shows the relationship of the aligned data to the octet counts which are output from a 2-lane IP core with F=3. The octet counter counts from 0 to 2, representing the first, second, and third bytes of a 3-byte frame. When the count value is 0, it represents the first byte of the frame, 1 is the second byte, and 2 is the third byte. Again, for any 2-byte pair, the high byte is the first byte received and the low byte is the next byte received.

Table 2-1. Octet Number Within Frame Versus Octet Data at Rx User Interface

rx_ocnt_u_l	1	0	2	1
rx_ocnt_u_h	0	2	1	0
d_out_a[7:0] - lane 0 (low byte)	byte 1	byte 0	byte 2	byte 1
d_out_a[15:8] - lane 0 (high byte)	byte 0	byte 2	byte 1	byte 0
d_out_a[23:16] - lane 1 (low byte)	byte 1	byte 0	byte 2	byte 1
d_out_a[31:24] - lane 1 (high byte)	byte 0	byte 2	byte 1	byte 0

Logic external to the IP core will be needed to assemble the octets of the aligned data from the Rx IP core into the samples and control bits according to the user's sample stream mapping. This is the reason it is necessary for the Rx IP core to indicate which octet of the frame appears on the d_out_a bus during each clock cycle.

Tx Core

Figure 2-5 shows the I/O ports on the Tx IP core.
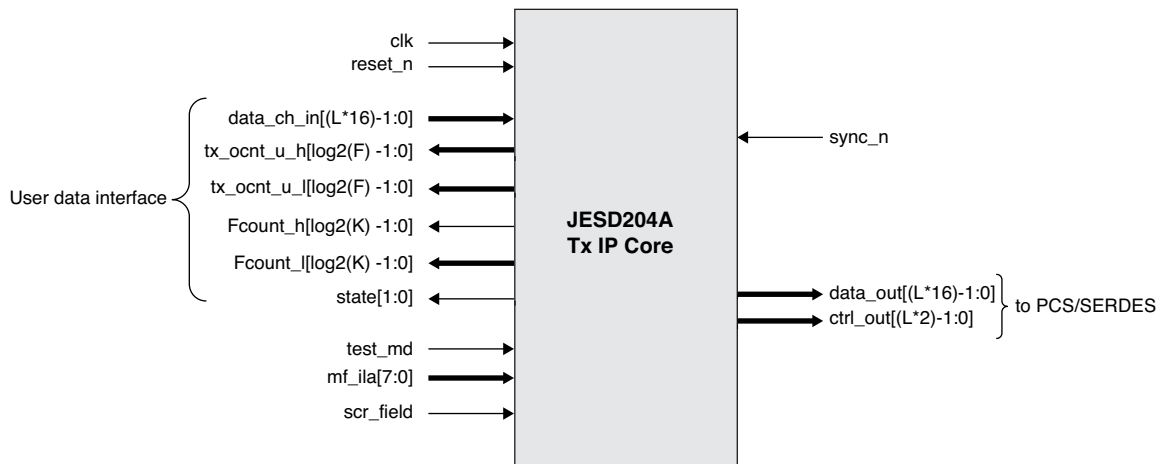
Figure 2-5. Tx IP Core I/O

Figure 2-6 shows a block diagram of the Tx IP core. Frame data from the user is input to the core at a rate of two octets per clock cycle per lane. The lane data then passes through the scrambler and alignment and framing characters are inserted into the data. A controller selects either frame data or initialization sequence data to send to the PCS/Serdes where 8b/10b encoding is performed.

Figure 2-6. JESD204A TX IP Core Block Diagram
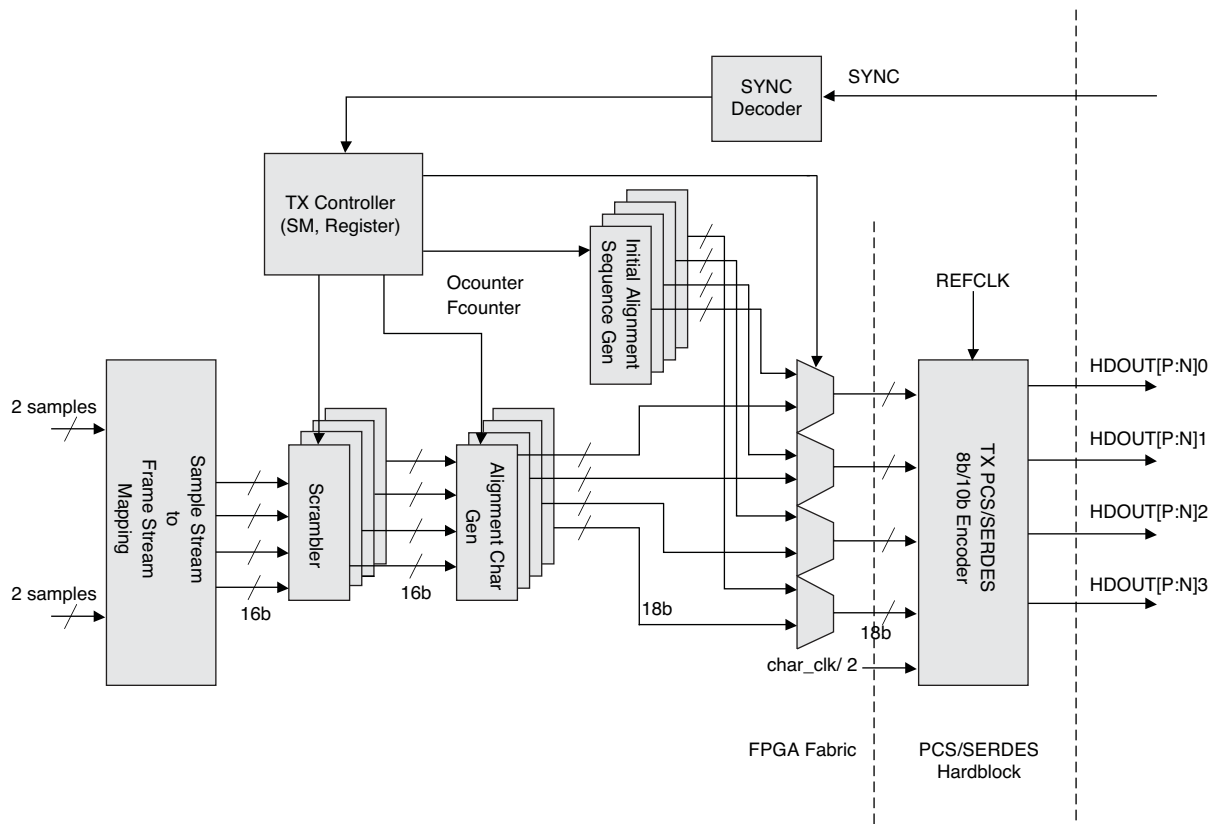
Frame data supplied by the user must be aligned to the octet counter outputs from the Tx IP core, similar to the way the Rx core outputs data aligned to the receive side octet counters. Table 2-2 shows how the transmit data input to the Tx IP core by the user must be aligned to the tx_ocnt_u_h and tx_ocnt_u_l counters. This example is for a 2-lane Tx IP core with F=3.

Table 2-2. Octet Count Values Versus Octet Data at Tx User Interface

tx_ocnt_u_l	1	0	2	1
tx_ocnt_u_h	0	2	1	0
data_ch_in[7:0] - lane 0 (low byte)	byte 1	byte 0	byte 2	byte 1
data_ch_in[15:8] - lane 0 (high byte)	byte 0	byte 2	byte 1	byte 0
data_ch_in[23:16] - lane 1 (low byte)	byte 1	byte 0	byte 2	byte 1
data_ch_in[31:24] - lane 1 (high byte)	byte 0	byte 2	byte 1	byte 0

Interface Descriptions

Table 2-3 shows the Rx core I/O signals. All signals are synchronous with respect to the character clock input (clk). All outputs are registered. All signals are active high unless otherwise noted.

Table 2-3. Rx IP Core I/O

Signal	Signal Direction	Description
clk	Input	Character clock input.
reset_n	Input	Active low core reset.
syncn	Output	Active low sync signal to JESD204A transmitter.
rx_data[(L*16)-1:0]	Input	Receive data from PCS/SERDES, 16 bits per lane.
rx_disp_err[(L*2)-1:0]	Input	Receive disparity error from PCS/SERDES, 2 bits per lane.
rx_cv_err[(L*2)-1:0]	Input	Receive coding violation error from PCS/SERDES, 2 bits per lane.
rx_k[(L*2)-1:0]	Input	Receive control character from PCS/SERDES, 2 bits per lane.
d_out_a[(L*16)-1:0]	Output	Aligned data output to user interface, 16 bits per lane.
ocnt_u_l[log2(F)-1:0]	Output	Octet counter for user interface (low-byte).
ocnt_u_h[log2(F)-1:0]	Output	Octet counter for user interface (high-byte).
fr_align_corr_en[L-1:0]	Input	Frame alignment correction enable, 1 enable bit per lane, allows automatic frame alignment corrections by Rx IP core.
ln_align_corr_en[L-1:0]	Input	Lane alignment correction enable, 1 enable bit per lane, allows automatic lane alignment corrections by Rx IP core.
uncorr_align_err[L-1:0]	Output	Uncorrectable alignment error, 1 bit per lane, an alignment error has occurred on a lane which cannot be compensated for, the sync signal to the transmitter must be forced active.
all_aligned	Output	All lanes aligned indicator, when high indicates that all lanes successfully aligned during initialization sequence.
lane_align[L-1:0]	Output	Active high indicates individual lanes successfully aligned during initialization sequence, reported 1 bit per lane.
ptr_offset[(L*4)-1:0]	Output	Read/write pointer offsets in alignment buffer, 4 bits per lane, indicates offset between read and write pointers in the 16 byte buffer.
unexpect_ctrl[L-1:0]	Output	Unexpected control character received, reported 1 bit per lane, active high indicates a control character was received in an unexpected byte of the frame.
frame_realigned[L-1:0]	Output	Frame realignment indicator, reported 1 bit per lane, active high indicates that a frame realignment has occurred.
cgs_err[L-1:0]	Output	Code group error detected, reported 1 bit per lane.
cfg_mismatch_err[L-1:0]	Output	Configuration mismatch error, reported 1 bit per lane, indicates mismatch between Rx parameters and configuration values received over link during initialization sequence.
init_align_fail	Output	Initialization sequence alignment error, indicates a failure to align all lanes during initialization sequence, this signal is only valid at the end of an initialization sequence.
max_skew	Output	Maximum skew reached in alignment buffer, indicates that an initialization sequence must be performed.
scr_en[L-1:0]	Input	Receive side descrambling enable, 1 enable bit per lane.

Table 2-4 shows the Tx core I/O signals. All signals are synchronous with respect to the character clock input (clk). All outputs are registered. All signals are active high unless otherwise noted.

Table 2-4. Tx IP Core I/O

Signal	Signal Direction	Description
clk	Input	Character clock input.
reset_n	Input	Active low core reset.
syncn	Input	Active low sync signal from JESD204A receiver.
data_out[(L*16)-1:0]	Output	Transmit data to PCS/serdes, 16 bits per lane.
ctrl_out[(L*2)-1:0]	Output	Transmit control charcter to PCS/serdes, 2 bits per lane.
data_ch_in[(L*16)-1:0]	Input	Frame data from user, 16 bits per lane.
ocnt_u_l[log2(F)-1:0]	Output	Octet counter (low byte) for user interface, used to synchronize frame data to octet data.
ocnt_u_h[log2(F)-1:0]	Output	Octet counter (high byte) for user interface, used to synchronize frame data to octet data.
Fcount_l[log2(K)-1:0]	Output	Frame counter (low byte), counts down from K-1 to 0, most implementations may not need to use this count value.
Fcount_h[log2(K)-1:0]	Output	Frame counter (high byte), counts down from K-1 to 0, most implementations may not need to use this count value.
state[1:0]	Output	State of transmitter state machine: 00: SYNC 01: INIT_LANE 10: DATA_ENC 11: illegal These values are provided as outputs only to monitor the state of the Tx core operation. They may not be needed in particular implementations.
test_md	Input	Test mode, not presently supported
mf_ila[7:0]	Input	Multi-frame initialization lane alignment value, sets the number of multi-frames in the initialization sequence from 4 to 256.
scr_field	Input	Scrambling enable for transmitter.

Parameter Settings

Parameters

Table 3-1 shows the user parameters used to generate the Rx or Tx IP core. Parameters F, K, and L are the only parameters which affect the core generation. For the Tx core, all parameter values selected by the user in the GUI during Tx core generation are sent in the link configuration data fields during the initialization sequence. For the Rx core, all parameters received in the link configuration data fields are compared to the user selected parameter values after the initialization sequence is complete.

Table 3-1. JESD204A IP Core Parameters

Parameter	Effects Core Generation	Compared to Configuration Values Received During Link Initialization	Description
F	Yes	Yes	Number of octets per frame
K	Yes	Yes	Number of frames per multiframe
L	Yes	Yes	Number of lanes on the JESD204A interface
CF		Yes	Number of control words per frame clock cycle per link
CS		Yes	Number of control bits per conversion sample
HD		Yes	High-density mode, samples are allowed to be divided across multiple lanes
N		Yes	Number of bits per sample
N'		Yes	Number of bits per sample after padding to nibble groups
M		Yes	Number of converter devices (ADC)
S		Yes	Number of samples per converter per frame

Rx Core Parameter Selection

When generating the Rx core using IPexpress, the Global tab of the GUI, shown in Figure 3-1, allows the F, K, and L parameters to be selected. Only combinations supported by the IP core are allowed to be entered.
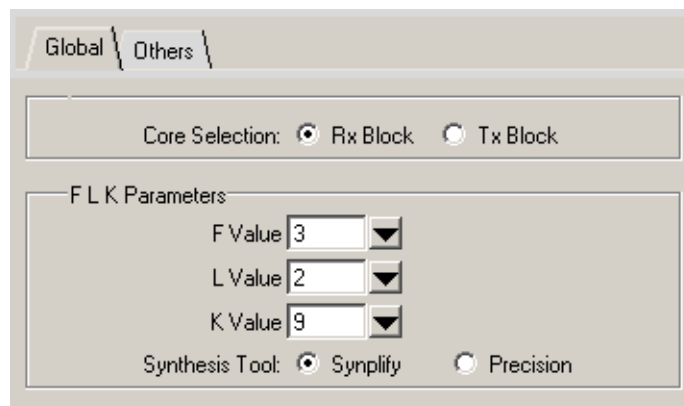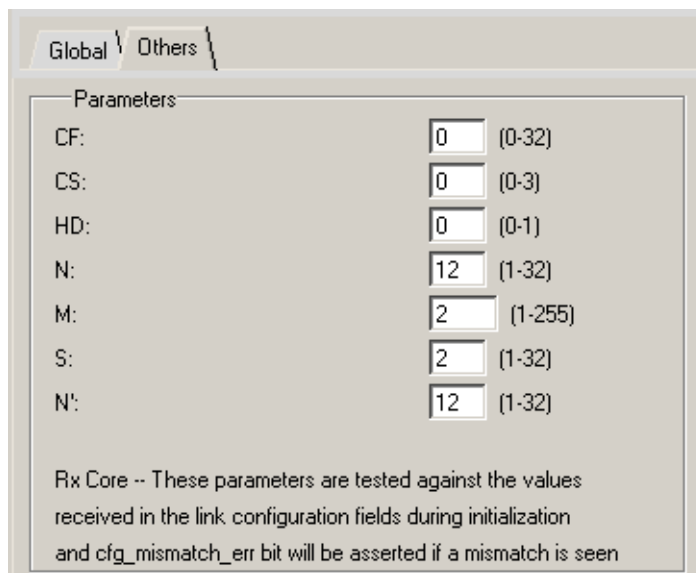
Figure 3-1. Rx Global Tab

Figure 3-2 shows the Others tab. These parameters set the expected values to be received during link configuration. Any mismatches will result in the cfg_mismatch_err being asserted following initialization.

Figure 3-2. Rx Others Tab

Tx Core Parameter Selection

The Global tab for generating the Tx core is shown in Figure 3-3. This dialog box allows the F, K, and L parameters to be set. Only combinations supported by the IP core are allowed to be entered.
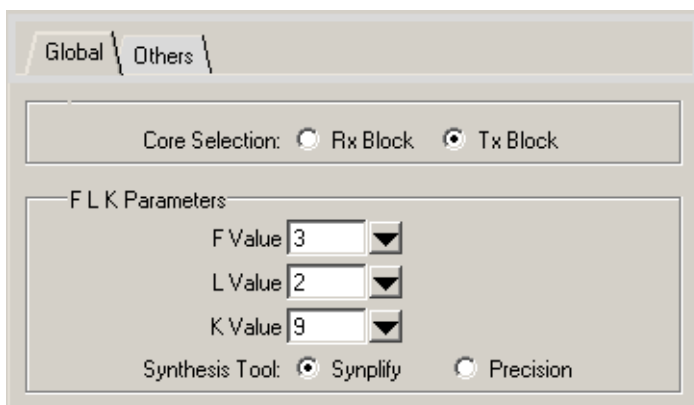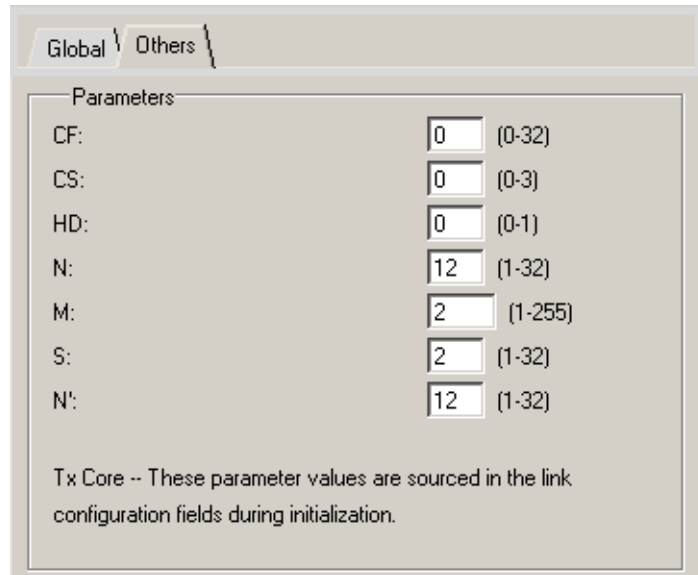
Figure 3-3. Tx Global Tab

Figure 3-4 shows the Others tab. These parameters set the values which will be transmitted in the link configuration data during the initialization sequence. These values have no other effect.

Figure 3-4. Tx Others Tab

IP Core Generation

This chapter provides information on how to generate the Lattice JESD204A IP core using the Diamond or ispLEVER software IPexpress tool, and how to include the core in a top-level design.

Licensing the IP Core

An IP core-specific license is required to enable full, unrestricted use of the JESD204A IP core in a complete, top-level design. Instructions on how to obtain licenses for Lattice IP cores are given at:

www.latticesemi.com/products/intellectualproperty/aboutip/isplevercoreonlinepurchas.cfm

Users may download and generate the JESD204A IP core and fully evaluate the core through functional simulation and implementation (synthesis, map, place and route) without an IP license. The JESD204A IP core also supports Lattice's IP hardware evaluation capability, which makes it possible to create versions of the IP core that operate in hardware for a limited time (approximately four hours) without requiring an IP license. See "Hardware Evaluation" on page 22. for further details. However, a license is required to enable timing simulation, to open the design in the ispLEVER or Diamond EPIC tool, and to generate bitstreams that do not include the hardware evaluation timeout limitation.

Getting Started

The JESD204A IP core is available for download from the Lattice IP Server using the IPexpress tool. The IP files are automatically installed using ispUPDATE technology in any customer-specified directory. After the IP core has been installed, the IP core will be available in the IPexpress GUI dialog box shown in Figure 4-1.

The IPexpress GUI dialog box for the JESD204A IP core is shown in Figure 4-1. To generate a specific IP core configuration the user specifies:

- **Project Path** – Path to the directory where the generated IP files will be located.

- **File Name** – "username" designation given to the generated IP core and corresponding folders and files.

- **(Diamond) Module Output** – Verilog or VHDL.

- **(ispLEVER) Design Entry Type** – Verilog HDL or VHDL.

- **Device Family** – Device family to which IP is to be targeted (e.g. LatticeECP3, etc.). Only families that support the particular IP core are listed.

- **Part Name** – Specific targeted part within the selected device family..

Figure 4-1. IPexpress Dialog Box (Diamond Version)
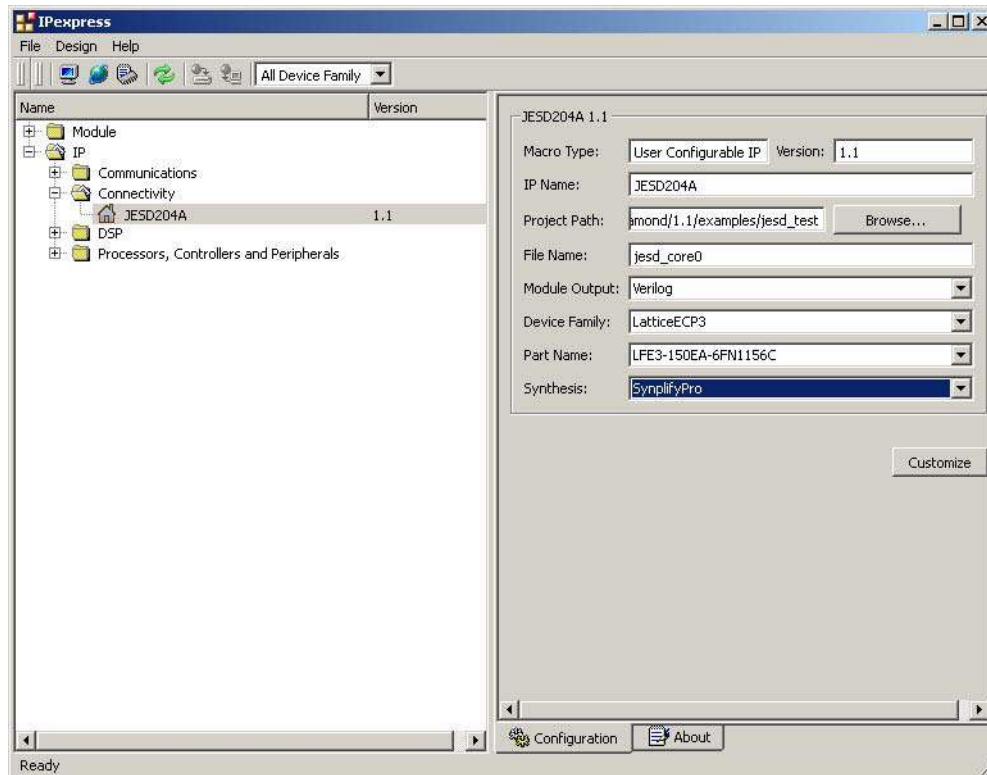
Note that if the IPexpress tool is called from within an existing project, Project Path, Module Output (Design Entry in ispLEVER), Device Family and Part Name default to the specified project parameters. Refer to the IPexpress tool online help for further information.

To create a custom configuration, the user clicks the **Customize** button in the IPexpress dialog box to display the JESD204A IP core Configuration GUI, as shown in Figure 4-2. From this dialog box the user can select the IP parameter options specific to their application. Refer to "Parameter Settings" on page 12 for more information on the JESD204A parameter settings.

Figure 4-2. Configuration GUI (Diamond Version)
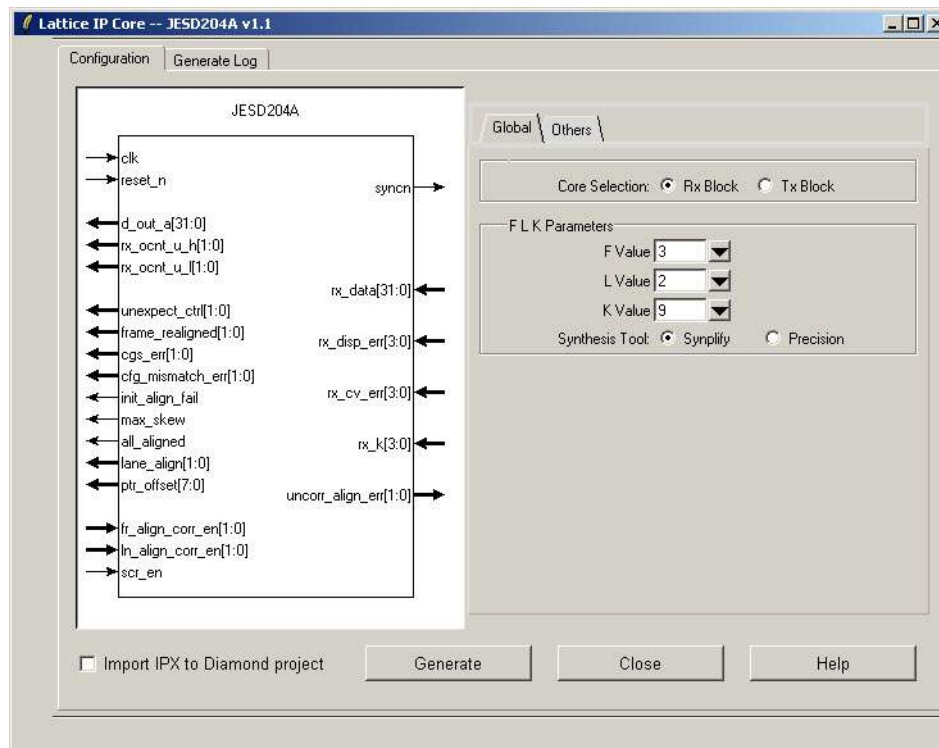
IPexpress-Created Files and Top Level Directory Structure

When the user clicks the **Generate** button in the IP Configuration dialog box, the IP core and supporting files are generated in the specified "Project Path" directory. The directory structure of the generated files is shown in Figure 4-3.

Figure 4-3. JESD204A Core Directory Structure
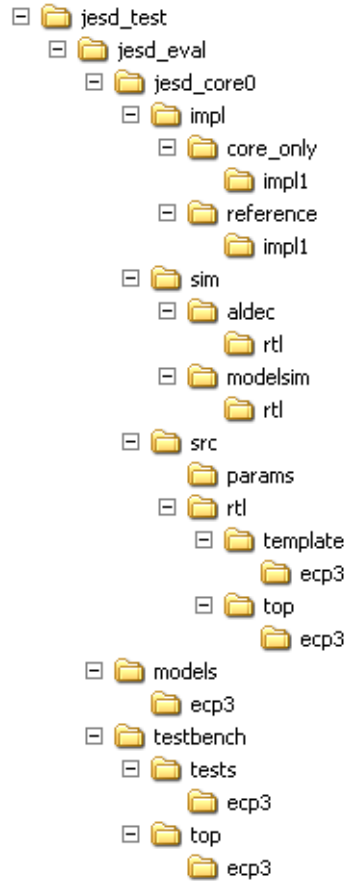
The design flow for IP created with IPexpress uses a post-synthesized module (NGO) for synthesis and a protected model for simulation. The post-synthesized module is customized and created during IPexpress generation. The protected simulation model is not customized during IPexpress and instead uses a fixed set of parameters (F=3, K=9, L=2).

Table 4-1 provides a list of key files and directories created by IPexpress and how they are used. IPexpress creates several files that are used throughout the design cycle. The names of most of the created files created are customized to the user's module name specified in IPexpress.

Table 4-1. Files Generated by IPexpress

File	Description
*<username>*_inst.v	This file provides a Verilog instance template for the IP.
*<username>*_inst.vhd	This file provides a VHDL instance template for the IP.
<username>.v	This file provides the JESD204A IP core wrapper for simulation. It instantiates and configures <username>_beh.v below.
*<username>*_beh.v	This file provides the simulation model for the JESD204A IP core.
*<username>*_bb.v	This file provides the synthesis black box for the user's synthesis.
<username>.ngo	This file provides the GUI configured synthesized IP core.
<username>.lpc	This file contains the IPexpress options used to recreate or modify the core in IPexpress.
<username>.ipx	The IPX file holds references to all of the elements of an IP or Module after it is generated from the IPexpress tool (Diamond version only). The file is used to bring in the appropriate files during the design implementation and analysis. It is also used to re-load parameter settings into the IP/Module generation GUI when an IP/Module is being re-generated.

Table 4-1. Files Generated by IPexpress (Continued)

File	Description
*<username>*_eval	This directory contains a sample reference design that utilizes the IP core. This design supports simulation and implementation using Diamond or ispLEVER.
eval_support	This directory contains simulation models and .ngo support for the reference design that utilizes the IP core.

Most of the files required to use the JESD204A IP core in a user's design reside in the root directory `\<username>` created by IPexpress. This includes the simulation model supporting simulation and a synthesis black box and .ngo file for implementing the design in Diamond or ispLEVER. The `\<username>` folder (`jesd` in this example) contains files/folders with content specific to the *<username>* configuration. This directory is created by IPexpress each time the core is generated and regenerated each time the core with the same file name is regenerated. A separate \<username> directory is generated for cores with different names, e.g. `\<my_core_0>`, `\<my_core_1>`, etc.

The `\<username>_eval` (jesd_eval in this example) and subtending directories provide files supporting JESD204A IP core evaluation that includes both simulation and implementation via Diamond or ispLEVER. The \<username>_eval directory contains files/folders with content that is constant for all configurations of the JESD204A IP core. The `\jesd_eval` directory is created by IPexpress the first time the core is generated and updated each time the core is regenerated.

The implementation part of user evaluation supports both core only and reference design options using Synplify (Precision RTL synthesis will be supported in the next release). Separate directories located at `\<username>\jesd_eval\<username>\impl\[core_only` or `reference]` are provided for implementation (synthesis, map, place and route) using Diamond or ispLEVER and contain specific pre-built project files. Implementations performed in the core_only directory contain only the core and can therefore be used to show the size of the core. The reference directory implementation option reveals a much larger design because it contains additional example user logic functions. The models directory provides library elements such as the PCS/SERDES.

The simulation part of user evaluation provides project files and test cases supporting RTL simulation for both the Active-HDL and ModelSim simulators. Separate directories located at `\<username>\jesd_eval\<username>\sim\[Aldec` or `Modelsim]\rtl` are provided and contain specific pre-built simulation project files. See section for more details.

Directory `\<username>\eval_support` provides simulation model and .ngo build support files for the reference design example user logic functions described above.

Instantiating the Core

The generated JESD204A IP core package includes a black-box (*<username>*_bb.v) and an instance (*<username>*_inst.v/vhd) template that can be used to instantiate the core in a top-level design. Three example RTL top-level source files (jesd_tx_core_only_top.v, jesd_rx_core_only_top.v, and jesd_reference_top.v) are provided and can also be used for instantiation template information of the IP core. These files can be found at `\<username>\jesd_eval\<username>\src\rtl\top\<device_family>`. Users may also use the top-level reference version as the starting point for their top-level design.

Running Functional Simulation

Simulation support for the JESD204A IP core is provided for Active-HDL and ModelSim simulators. The JESD204A IP core simulation model is generated from IPexpress with the name *<username>*.v. This file calls <username>_beh.v which contains the obfuscated simulation model. An obfuscated simulation model is Lattice's unique IP protection technique which scrambles the Verilog HDL while maintaining logical equivalence. VHDL users will use the same Verilog model for simulation.

The functional simulation includes a fixed configuration behavioral model of the JESD204A IP Core that is instanti-ated in an FPGA top level source file along with simulation support modules (see section "Simulation Strategies" on page 20 for details).

The top-level file supporting ModelSim eval simulation is provided in
`\<project_dir>\jesd_eval\<username>\sim\modelim\rtl\<device_family>\reference_top.v.`

This FPGA top is instantiated in an evaluation test bench provided in
`\<project_dir>\jesd_eval\testbench\top\<device_family>\tb_ecp3.v`. This module includes a simple pattern generator which drives the Tx user data inputs (data_ch_in) with an incrementing count pattern.

Users may run the eval simulation by doing the following:

1. Open ModelSim.

2. Under the **File** tab, select **Change Directory** and choose folder
   `\<project_dir>\jesd_eval\<username>\sim\modelsim\rtl.`

3. Under the **Tools** tab, select **Execute Macro** and execute one of the ModelSim "do" scripts shown.

The top-level file supporting Aldec Active-HDL simulation is provided in
`\<project_dir>\jesd_eval\<username>\sim\aldec\rtl.` This FPGA top is instantiated in an eval test bench provided in `\<project_dir>\jesd_eval\testbench\top\<device_family>` that drives the Tx data with a simple incrementing count pattern.

Users may run the eval simulation by doing the following:

1. Open Active-HDL.

2. Under the **Console** tab change the directory to:
   `\<project_dir>\jesd_eval\<username>\sim\aldec\rtl.`

3. Execute the Active-HDL "do" scripts shown.

The simulation waveform results will be displayed in the Aldec Wave window.

Simulation Strategies

This section describes the simulation environment which demonstrates basic JESD204A functionality. Figure 4-4 shows a block diagram of the simulation environment.

Figure 4-4. Simulation Environment Block Diagram
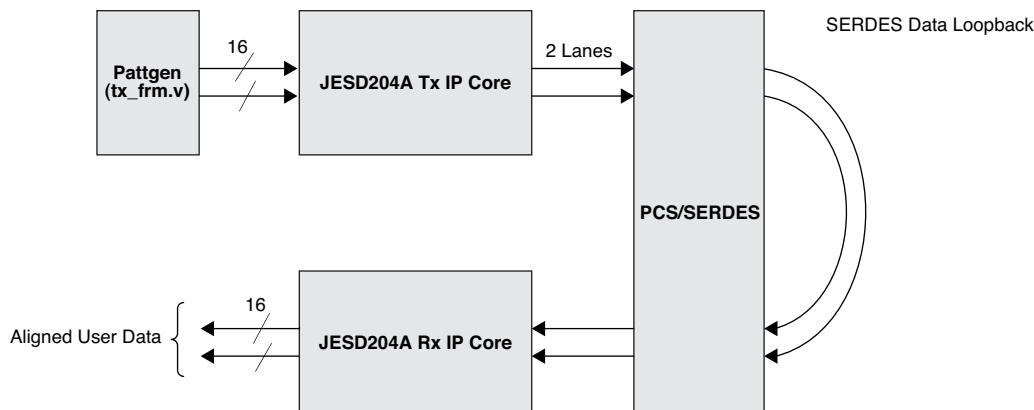
Simulation Environment

The simulation environment is made up of a Tx core and and Rx core connected to each other by looping back the PCS/SERDES outputs to inputs. Both the Tx and Rx IP cores used in this simulation environment have a fixed set of paramers (F=3, K=9, L=2). Data is driven into the Tx core's user inputs from a simple pattern generator which creates an incrementing upcount. Periodically the upcount is interrupted and the same value is repeated so that the Tx core can insert Frame and Alignment characters into the JESD204A link data stream. User data output from the Rx IP core is viewable in the simulator's wave window.

To verify that the circuit is operating correctly, after the Rx core reports that all lanes are aligned (output signal all_aligned = 1), the user can verify that the data on both lanes is identical which means that the lanes are aligned, and that the upcount pattern is seen. Figure 4-5 shows the results from a simulation. The data patterns high-lighted show the incrementing patterns on the Rx outputs (0xda, 0xdb, 0xdc, …). Also shown is that periodically the pattern generator will repeat the same value twice. If this were not done, the Tx IP core would never have the opportunity to insert /A/ alignment and /F/ framing characters into the JESD204A link datastream.

Figure 4-5. Data Patterns Seen on User Interface in Simulation
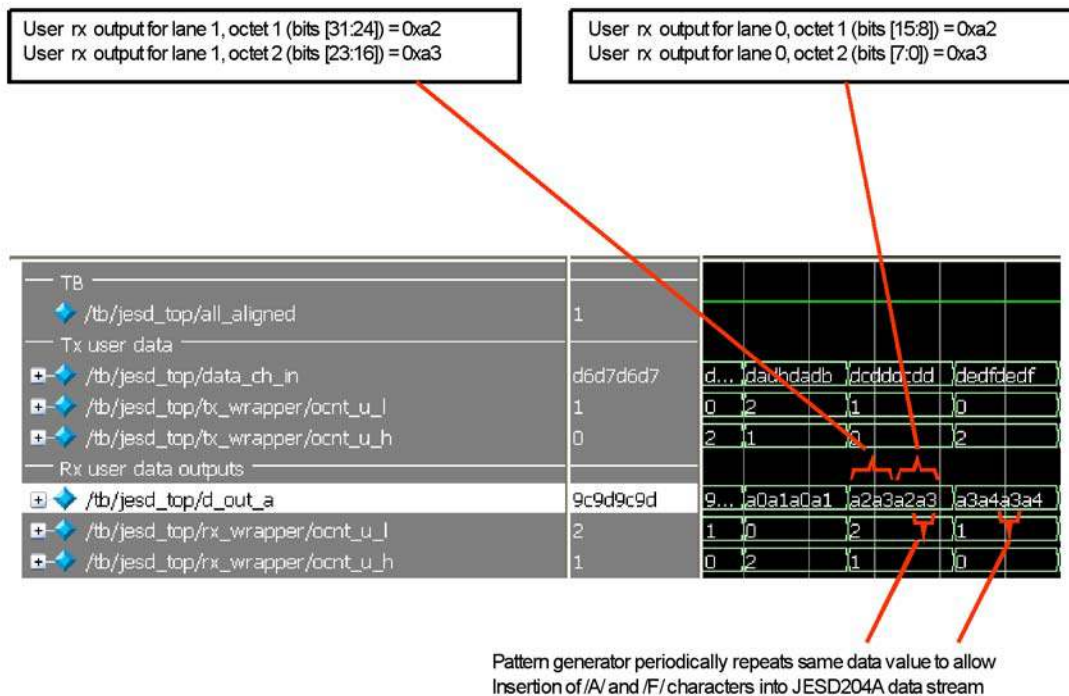
Synthesizing and Implementing the Core in a Top-Level Design

The JESD204A IP core itself is synthesized and provided in NGO format when the core is generated through IPex-press. Users may use the core in their own top-level design by instantiating the core in their top-level as described previously and then synthesizing the entire design with either Synplify or Precision RTL Synthesis.

The top-level file jesd_core_only_top.v provided in
`\<project_dir>\jesd_eval\<username>\src\rtl\top\<device_family>` supports the ability to implement the JESD204A IP Express core in isolation. Push-button implementation of this top-level design with either Synplify or Precision (future release) RTL Synthesis is supported via the Diamond or ispLEVER project files jesd_core_only_eval.syn located in the
`\<project_dir>\jesd_eval\<username>\impl\core_only\` directory.

To use this project file in Diamond:

1. Choose **File > Open > Project**.

2. Browse to
   `\<`*`project_dir`*`>\jesd_eval\<`*`username`*`>\impl\synplify (or precision)` in the Open Project dialog box.

3. Select and open *<username>*.ldf. At this point, all of the files needed to support top-level synthesis and implementation will be imported to the project.

4. Select the **Process** tab in the left-hand GUI window.

5. Implement the complete design via the standard Diamond GUI flow.

To use this project file in ispLEVER:

1. Choose **File > Open Project.**

2. Browse to
   `\<`*`project_dir`*`>\jesd_eval\<`*`username`*`>\impl\synplify (or precision)` in the Open Project dialog box.

3. Select and open *<username>*.syn. At this point, all of the files needed to support top-level synthesis and implementation will be imported to the project.

4. Select the device top-level entry in the left-hand GUI window.

5. Implement the complete design via the standard ispLEVER GUI flow.

Hardware Evaluation

The JESD204A IP core supports supports Lattice's IP hardware evaluation capability, which makes it possible to create versions of the IP core that operate in hardware for a limited period of time (approximately four hours) without requiring the purchase of an IP license. It may also be used to evaluate the core in hardware in user-defined designs.

Enabling Hardware Evaluation in Diamond

Choose **Project > Active Strategy > Translate Design Settings**. The hardware evaluation capability may be enabled/disabled in the Strategy dialog box. It is enabled by default.

Enabling Hardware Evaluation in ispLEVER

In the Processes for Current Source pane, right-click the **Build Database** process and choose **Properties** from the dropdown menu. The hardware evaluation capability may be enabled/disabled in the Properties dialog box. It is enabled by default.

Updating/Regenerating the IP Core

By regenerating an IP core with the IPexpress tool, you can modify any of its settings including device type, design entry method, and any of the options specific to the IP core. Regenerating can be done to modify an existing IP core or to create a new but similar one.

Regenerating an IP Core in Diamond

To regenerate an IP core in Diamond:

1. In IPexpress, click the **Regenerate** button.

2. In the Regenerate view of IPexpress, choose the IPX source file of the module or IP you wish to regenerate.

3. IPexpress shows the current settings for the module or IP in the Source box. Make your new settings in the T**ar-get** box.

4. If you want to generate a new set of files in a new location, set the new location in the **IPX Target File** box. The base of the file name will be the base of all the new file names. The IPX Target File must end with an .ipx extension.

5. Click **Regenerate.** The module's dialog box opens showing the current option settings.

6. In the dialog box, choose the desired options. To get information about the options, click **Help**. Also, check the About tab in IPexpress for links to technical notes and user guides. IP may come with additional information. As the options change, the schematic diagram of the module changes to show the I/O and the device resources the module will need.

7. To import the module into your project, if it's not already there, select **Import IPX to Diamond Project** (not available in stand-alone mode).

8. Click **Generate**.

9. Check the Generate Log tab to check for warnings and error messages.

10. Click **Close**.

The IPexpress package file (.ipx) supported by Diamond holds references to all of the elements of the generated IP core required to support simulation, synthesis and implementation. The IP core may be included in a user's design by importing the .ipx file to the associated Diamond project. To change the option settings of a module or IP that is already in a design project, double-click the module's .ipx file in the File List view. This opens IPexpress and the module's dialog box showing the current option settings. Then go to step 6 above.

Regenerating an IP Core in ispLEVER

To regenerate an IP core in ispLEVER:

1. In the IPexpress tool, choose **Tools > Regenerate IP/Module**.

2. In the Select a Parameter File dialog box, choose the Lattice Parameter Configuration (.lpc) file of the IP core you wish to regenerate, and click **Open**.

3. The Select Target Core Version, Design Entry, and Device dialog box shows the current settings for the IP core in the Source Value box. Make your new settings in the Target Value box.

4. If you want to generate a new set of files in a new location, set the location in the LPC Target File box. The base of the .lpc file name will be the base of all the new file names. The LPC Target File must end with an .lpc extension.

5. Click **Next**. The IP core's dialog box opens showing the current option settings.

6. In the dialog box, choose desired options. To get information about the options, click **Help**. Also, check the About tab in the IPexpress tool for links to technical notes and user guides. The IP core might come with additional information. As the options change, the schematic diagram of the IP core changes to show the I/O and the device resources the IP core will need.

7. Click **Generate**.

8. Click the **Generate Log** tab to check for warnings and error messages.

# Lattice Semiconductor Corporation

The LatticeECP3 I/O Protocol Board can be used for hardware evaluation of the JESD204A IP core. Other Lattice LatticeECP3 evaluation boards, such as the LatticeECP3 Serial Protocol Board, may not work due to insufficient SMA connectors on the board to support both the JESD204A link and the syncn signal.

The JESD204A IP core was initially tested by looping the PCS/SERDES outputs from the Tx IP core back to the PCS/SERDES inputs on the Rx IP core. This testing was done using LatticeECP2M™ and LatticeECP3 evaluation boards. This loop-back configuration verified that the Tx and Rx cores could successfully communicate with each other. For this testing, the Tx side pattern generator (tx_frm.v), included in the reference configuration, was used to generate a test pattern for the Tx side inputs. On the receive side, the Reveal Logic Analyzer was connected to the outputs of the Rx user interface. It was verified that the test pattern was received successfully, and that multiple lanes remained aligned. An RTL module was used to insert skew across lanes on the transmit side between the Tx IP core and the Tx PCS/SERDES inputs. The amount of skew was varied over time to verify that the Rx IP core could compensate for the skew and keep all lanes aligned (within the tolerance of the Rx buffers).

Next, interoperability testing was performed between a LatticeECP3 device on a Lattice evaluation board connected to an NXP ADC1413D on an NXP evaluation board. This tested the Lattice JESD204A Rx IP core. The parameters used for this testing were F=1, K=18, L=4, CF=0, CS=0, HD=1, M=2, N=14, N'=16, and S=1. The character clock was 125 MHz. The line rate was 2.5 Gb.

After establishing the connection and resetting and configuring the NXP board, the status outputs of the Lattice Rx IP core were checked to verify that the core could frame on the JESD204A link data from the NXP ADC device. Next, the NXP board sourced various test patterns to the Rx IP Core. The Reveal Logic Analyzer was connected to the Rx IP core user outputs (d_out_a) to verify that the data patterns sourced from the NXP board were successfully received by the Rx IP core, and that all lanes remained in alignment. Figure 6-1 shows the test setup.

Figure 6-1. LatticeECP3 Interoperability Testing with NXP ADC

Support Resources
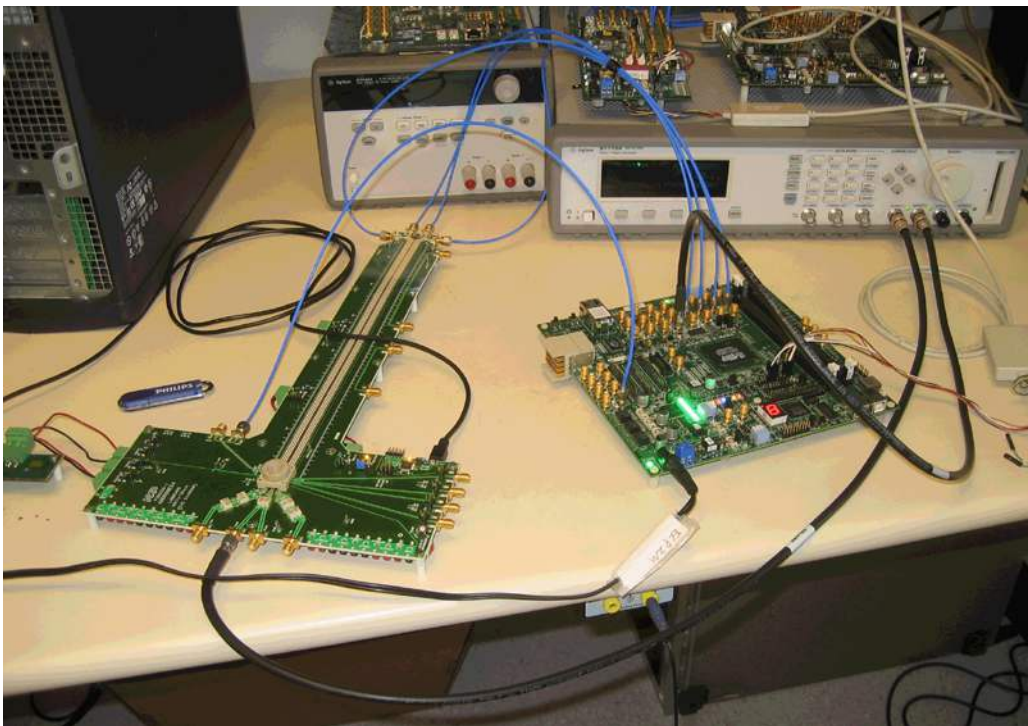
This chapter contains information about Lattice Technical Support, additional references, and document revision history.

Lattice Technical Support

There are a number of ways to receive technical support.

Online Forums

The first place to look is Lattice Forums (http://www.latticesemi.com/support/forums.cfm). Lattice Forums contain a wealth of knowledge and are actively monitored by Lattice Applications Engineers.

Telephone Support Hotline

Receive direct technical support for all Lattice products by calling Lattice Applications from 5:30 a.m. to 6 p.m. Pacific Time.

- For USA & Canada: 1-800-LATTICE (528-8423)

- For other locations: +1 503 268 8001

In Asia, call Lattice Applications from 8:30 a.m. to 5:30 p.m. Beijing Time (CST), +0800 UTC. Chinese and English language only.

- For Asia: +86 21 52989090

E-mail Support

- techsupport@latticesemi.com

- techsupport-asia@latticesemi.com

Local Support

Contact your nearest Lattice Sales Office.

Internet

www.latticesemi.com

References

- DS1021, LatticeECP3 Family Data Sheet

- TN1176, LatticeECP3 SERDES/PCS Usage Guide

- JEDEC Standard, JESD204A, April 2008, www.jedec.org

Revision History

Date	Document Version	IP Core Version	Change Summary
April 2010	01.0	1.0	Initial release.
May 2010	01.1	1.0	Minor corrections to the following diagrams: - JESD204A Receiver Functions - JESD204A Rx IP Core Block Diagram - JESD204A Tx IP Core Block Diagram
May 2010	01.2	1.0	Added "Licensing the IP Core" text section.
			Added "Hardware Evaluation" text section.
			Updated first footnote in Appendix A, LatticeECP3-70 Utilization table.
December 2010	01.3	1.1	Added support for Diamond software throughout.

Resage Utilization

This appendix gives resource utilization information for Lattice FPGAs using the JESD204A IP core.

IPexpress is the Lattice IP configuration utility, and is included as a standard feature of the Diamond and ispLEVER design tools. Details regarding the usage of IPexpress can be found in the IPexpress and Diamond or ispLEVER help system. For more information on the Diamond or ispLEVER design tools, visit the Lattice web site at: www.latticesemi.com/software.

LatticeECP3-70 Utilization

Table A-1 lists resource utilization for LatticeECP3-70 FPGAs using the JESD204A IP core.

Table A-1. Resource Utilization[1]

IPexpress User-Configurable Mode	Slices	LUTs	Registers	sysMEM EBRs	f_{MAX} (MHz)
Config 1 (Rx)	780	1012	761	0	125[2]
Config 1 (Tx)	337	483	342	0	125[2]

1. Performance and utilization data target an LFE3-70EA-6FN672C device using Lattice Diamond 1.1 and Synplify Pro for Lattice D-2010.03L-SP1 software. Performance may vary when using a different software version or targeting a different device density or speed grade within the LatticeECP3 family.
2. Fmax shown is for 2-lane configuration operating at 2.5 Gbaud using a -6 speed grade device. Higher line rates may require faster speed grades.

Ordering Part Number

The Ordering Part Number (OPN) for the JESD204A IP core targeting LatticeECP3-17/35/70/95/150 devices is JESD-204A-E3-U.