

Implementing JESD204B IP Core System Reference Design with Nios II Processor As Control Unit

2015.05.04

AN-729



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 or from the FPGA devices.

The JESD204B IP core is part of the MegaCore IP Library, which is distributed with the Quartus® II software and downloadable from the Altera website.

This reference design demonstrates the JESD204B IP core operating as part of a system that includes:

- Altera JESD204B transport layer (assembler and deassembler)
- test pattern generator and checker
- core PLL
- SPI master
- reset sequencer
- various dynamic reconfiguration controllers
- Nios® II processor as the control unit

The key feature of this reference design is the software-based control flow that utilizes the Nios II processor control unit.

The reference design utilizes the Arria 10 FPGA Development Kit to interoperate with the Analog Devices (ADI) AD9680 ADC daughter card connected to the development board.

Related Information

- [Reference Design Files for AN 729](#)
- [JESD204B IP Core User Guide](#)
- [Quartus II Handbook Volume 1: Design and Synthesis](#)
For detailed description of the reset sequencer module.
- [Arria 10 Transceiver PHY User Guide](#)
- [Nios II Processor Reference Handbook](#)
- [Nios II Software Developer's Handbook](#)
- [Embedded Peripherals IP User Guide](#)
- [ADI AD9680 ADC Datasheet](#)
- [ADI AD9516 Clock Module Datasheet](#)

Intel Corporation. All rights reserved. Intel, the Intel logo, Altera, Arria, Cyclone, Enpirion, MAX, Nios, Quartus and Stratix words and logos are trademarks of Intel Corporation or its subsidiaries in the U.S. and/or other countries. Intel warrants performance of its FPGA and semiconductor products to current specifications in accordance with Intel's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Intel assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Intel. Intel customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

*Other names and brands may be claimed as the property of others.

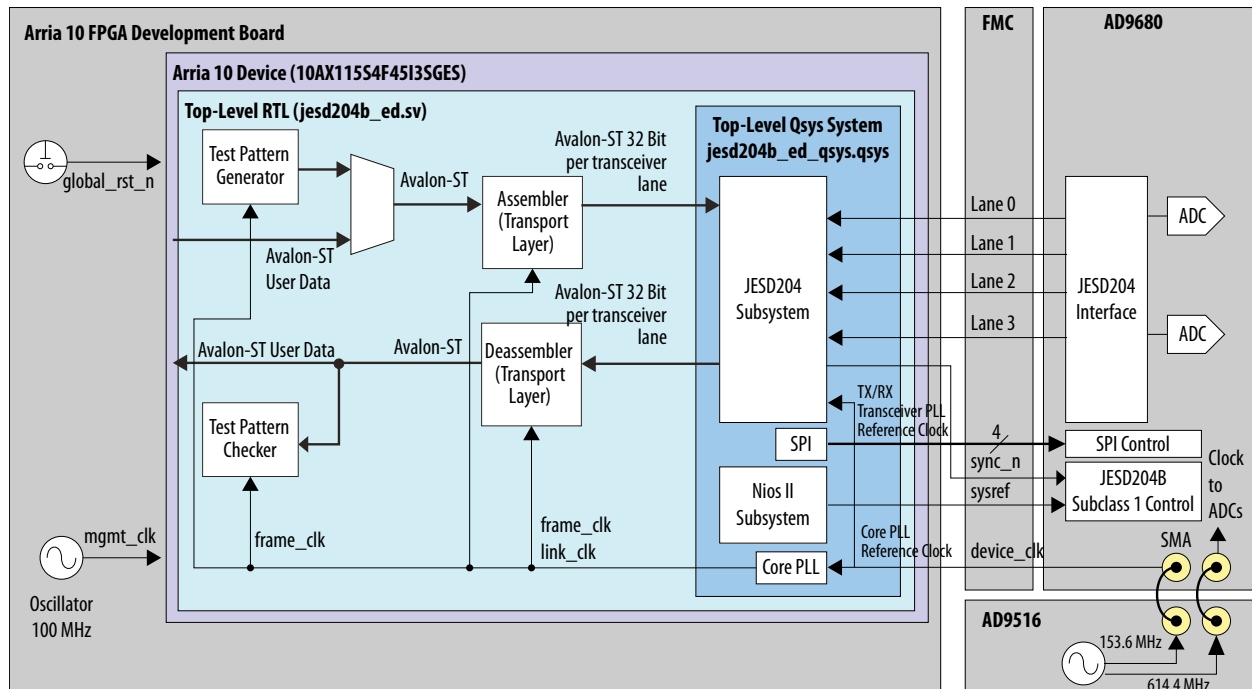
ISO
9001:2008
Registered

ALTERA
now part of Intel

Reference Design Overview

Figure below shows a system level block diagram of the JESD204B reference design with the Nios II processor control unit. This design is implemented on the Arria 10 FPGA development board interoperating with the ADI AD9680 ADC converter card.

Figure 1: Block Diagram of the JESD204B Reference Design with Nios II Processor



Reference design blocks:

- QSYS components:
 - JESD204B subsystem
 - Nios II subsystem
 - SPI master
 - Core PLL and core PLL reconfiguration controller
- HDL components:
 - Altera transport layer (assembler and deassembler)
 - Test pattern generator and checker
- Nios II subsystem generates *SYSREF* for the JESD204B IP core and the AD9680 module (for Subclass 1 mode).
- The `device_clk` (153.6Mhz) that is sent to the FPGA from the AD9516 external clock module is the reference clock for the on-chip core PLL, ATX PLL (which supplies the serial clock to the TX transceiver) and RX transceiver PLL.
- Core PLL module generates the link clock (`link_clk`) and frame clock (`frame_clk`).
- Oscillator on-board the Arria-10 FPGA development board supplies a 100 MHz management clock (`mgmt_clk`) to clock the control plane.

AD9516 external clock module

- Supplies a 614.4 MHz clock to the ADCs on the AD9680 module via an SMA connector.
- Supplies a 153.6 MHz reference clock to the FPGA via an SMA connector on the AD9680 module. The reference clock is passed through from the AD9680 module to the FPGA via the FMC connector.
- You can replace this module with any external clock module that supplies a 614.4 MHz and 153.6 MHz reference clock.

AD9680 module

- Configured to transmit on 4 high-speed transceiver lanes (L=4) to the FPGA.
- Each lane is configured to 6.144 Gbps data rate
- Derives power from the FMC connector on the Arria 10 FPGA development board.
- Passes the FPGA reference clock (`device_clk`) to the FPGA via the FMC connector.

Table 1: System Clocking

This table summarizes the system clocking of the reference design.

Clocks	Description	Modules Clocked
<code>device_clk</code>	Reference clock to the FPGA	Core PLL, ATX PLL, RX transceiver PLL.
<code>link_clk</code>	Link layer clock	JESD204B IP core link layer, transport layer link interface.
<code>frame_clk</code>	Frame layer clock	Transport layer, test pattern generator and checker, downstream modules.

Clocks	Description	Modules Clocked
mgmt_clk	Control plane clock	Nios II subsystem and any modules connected to Nios II via Avalon-MM bus interconnect.

Related Information**[JESD204B IP Core User Guide](#)**

For more information about the JESD204B system clocking.

Hardware and Software Requirements

This reference design uses the following hardware and software tools:

- Arria 10 FPGA development kit
- ADI AD9680 ADC converter card
- ADI AD9516 clock module
- Quartus II software version 15.0
- Nios II Embedded Design Suite (EDS)

Hardware Setup

1. Install the ADI AD9680 converter card module to the FMC port B (J2) on the Arria 10 FPGA development board.
2. Connect the mini-USB cable to the mini-USB connector (J3) on the development board.
3. Connect the power adapter shipped with the development board to the power supply jack (J13).
4. Connect the AD9516 power adapter to the power supply jack on the AD9516 clock module card.
5. Connect the USB cable from your workstation to the USB connector on the AD9516 card.
6. Connect the AD9516 clock module `OUT0` SMA connector (J0A) to the `CLKIN` SMA connector (J801) on the AD9680 card.
7. Connect the AD9516 clock module `OUT8` SMA connector (J8A) to the `Refclk to FPGA` SMA connector (J804) on the AD9680 card.
8. Turn on the power for the AD9516 clock module card.
9. Configure the clock settings of the AD9516 to output 614.4 MHz clock for the ADCs at `OUT0` SMA connector and 153.6 MHz for the FPGA at `OUT8` SMA connector. Refer to the AD9516 documentation for more information on configuring the AD9516 clock module.
10. Turn on the power for the Arria 10 FPGA development board.

The hardware system is now ready for programming. Refer to instructions in the Generating Programming File section on how to generate and download the programming file onto the FPGA.

Related Information

[Generating Programming File](#) on page 16

System Modules

The reference design consists of the following key modules and sub-modules:

- QSYS system
 - JESD204B subsystem
 - Nios II subsystem
 - Core PLL and PLL reconfiguration controller
 - SPI master
- Transport layer
 - Assembler (TX data path)
 - Deassembler (RX data path)
- Test pattern generator and checker

The transport layer (assembler and deassembler) and test pattern generator and checker modules are instantiated in the top level RTL (*jesd204b_ed.sv*).

Related Information

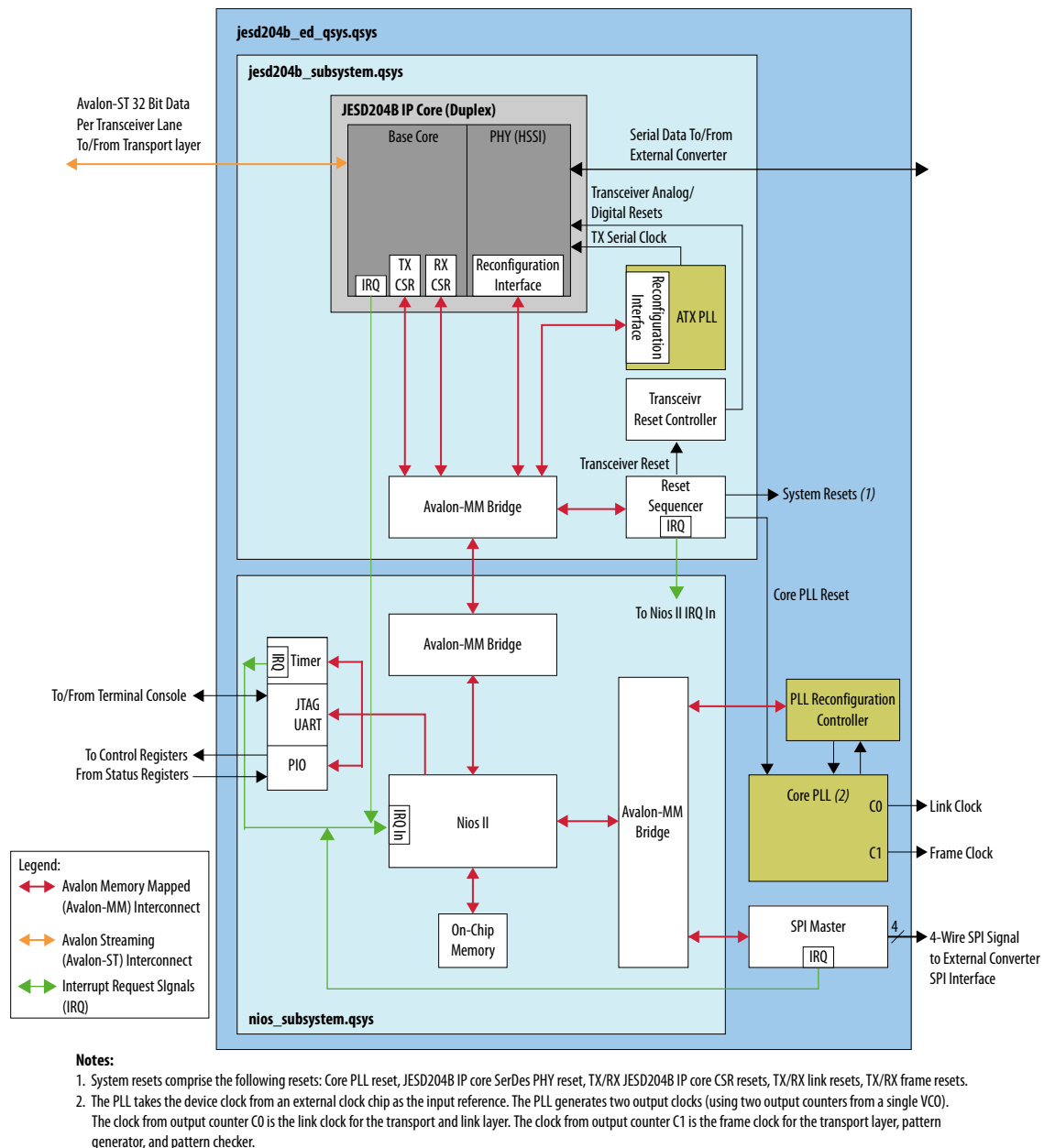
[JESD204B IP Core User Guide](#)

For more information about the features and functionality of the design example modules.

Qsys System

The Qsys system is the heart of the reference design, instantiating both the JESD204B IP core data path and the Nios II subsystem control path.

Figure 2: Qsys Subsystem



The top level Qsys system, *jcsd204b_ed_qsys.qsys*, instantiates the following modules:

- JESD204B subsystem
- Nios II subsystem
- Core PLL and PLL reconfiguration controller
- SPI master

The main data path flows through the JESD204B subsystem. On the RX data path, serial data flows from the external converters to the JESD204B IP core PHY module and out from the JESD204B IP core base

module to the transport layer via a 32-bit per transceiver lane Avalon Streaming (Avalon-ST) interface. In this reference design, since there are 4 transceiver lanes ($L=4$), the total bit width of the Avalon-ST interface is 128 bits.

The control path is centered on the Nios II processor in the Nios II subsystem and connects to various peripherals via the Avalon Memory-Mapped (Avalon-MM) interface. A secondary control path from the SPI master module links out to the SPI configuration interface of external converters via a 4-wire SPI interconnect. The configuration of the external converters is done by writing configuration data from the Nios II processor to the SPI master module. The SPI master module handles the serial transfer of data to the SPI interface on the converter end via the 4-wire SPI interconnect.

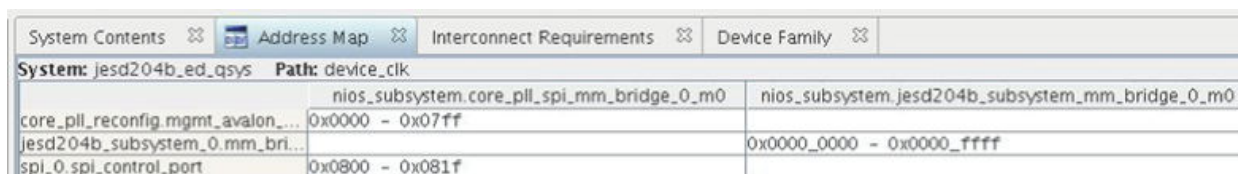
The core PLL generates the link clock and frame clock for the system. During a data rate dynamic reconfiguration process, the core PLL is dynamically reconfigurable at run time via the PLL reconfiguration controller. The data rate dynamic reconfiguration process is described in the Data Rate Reconfiguration section.

To view the top level Qsys system in Qsys, follow the steps below:

1. Launch the Quartus II software.
2. On the File menu, click Open.
3. Browse and select the `jesd204b_ed_qsys.qsys` file located in the project directory.
4. Click Open to view the QSYS system.

You can access the address mapping of the submodules in the top level Qsys project by clicking on the **Address Map** tab in the Qsys window.

Figure 3: Address Map View in Qsys



System: jesd204b_ed_qsys Path: device_clk	
core_pll_reconfig_mgmt_avalon...	0x0000 - 0x07ff
jesd204b_subsystem_0_mm_bri...	0x0000_0000 - 0x0000_ffff
spl_0_spi_control_port	0x0800 - 0x081f

The Qsys system supports multi-link scenarios (up to 16 links) using the existing address map. To add more links to the system, add more `jesd204b_subsystem.qsys` modules to the project, connect them to the `jesd204b_subsystem` Avalon-MM bridge, and adjust the address map accordingly. Bits 16-19 of the `jesd204b_subsystem` Avalon-MM bridge are reserved to support multi-links.

Related Information

[Data Rate Reconfiguration](#) on page 28

JESD204B Subsystem

The JESD204B subsystem Qsys project, `jesd204b_subsystem.qsys`, instantiates the following modules:

- JESD204B IP core configured in duplex, non-bonded mode (with TX and RX datapaths)
- Reset sequencer
- Transceiver PHY reset controller
- ATX PLL
- Avalon-MM bridge

Table 2: JESD204B IP Core Parameter Configuration

Parameter	Value	Description
Subclass	1	Subclass mode
L	4	Number of lanes per converter device
M	2	Number of converters per device
F	1	Number of octets per frame
S	1	Number of transmitted samples per converter per frame
N	14	Number of conversion bits per converter
N'	16	Number of transmitted bits per sample
K	32	Number of frames per multiframe
CS	0	Number of control bits per conversion sample
CF	0	Number of control words per frame clock period per link
HD	0	High Density user data format
SCR	Off	Enable scramble
FRAMECLK_DIV	4	The divider ratio of frame clock (used in transport layer assembler and deassembler modules)

Configuring the JESD204B IP core parameters in QSYS

To change the initial configuration parameters of the JESD204B IP core in Qsys, follow these steps:

1. Launch the Quartus II software.
2. On the File menu, click Open.
3. Browse and select the `jesd204b_ed_qsys.qsys` file located in the project directory.
4. Click Open to view the Qsys system.
5. Right-click on the `jesd204b_subsystem_0` module.
6. Select the **Drill into subsystem** option.
7. Double-click the `jesd204b` module. The parameter editor window appears.
8. After you are done entering the parameter values, click the **Move to the top of hierarchy** button on the top of the Qsys window to move back to the top level Qsys project hierarchy (`jesd204b_ed_qsys.qsys`).

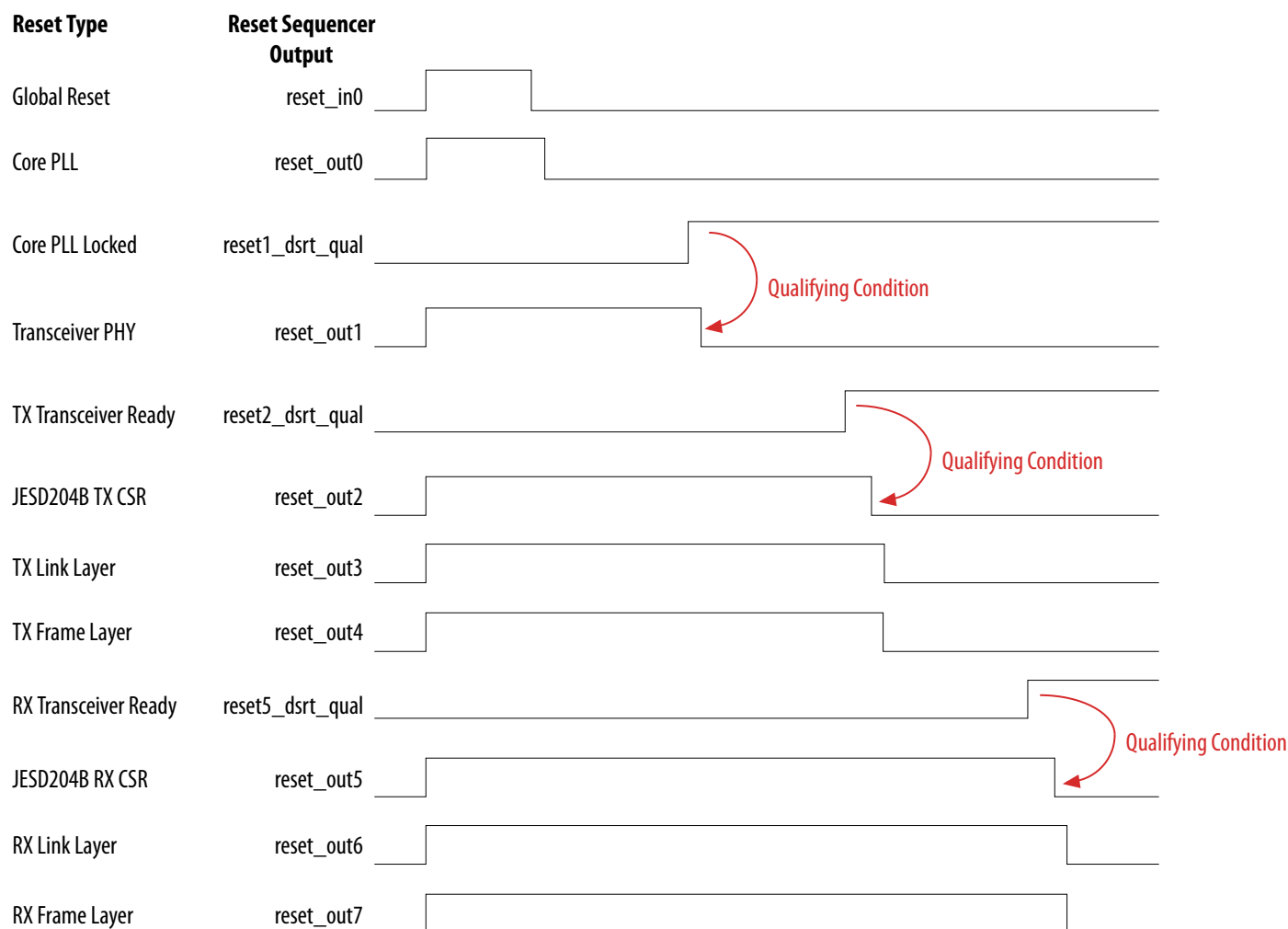
9. Click the **Generate HDL** button on the bottom right corner of the Qsys window.
10. After Qsys is done generating the HDL, click the **Finish** button on the bottom right corner of the Qsys window.
11. Recompile your design.

Reset Sequencer

The reset sequencer generates the following system resets to reset various modules in the system:

- Core PLL reset—resets the core PLL
- Transceiver reset—resets the JESD204B IP core PHY module
- TX/RX JESD204B IP core CSR reset—resets the TX/RX JESD204B IP core configuration status registers
- TX/RX link reset—resets the TX/RX JESD204B IP core base module and transport layer
- TX/RX frame reset—resets the TX/RX transport layer and downstream modules

The reset sequencer has hard and soft reset options. The hard reset port connects to the `PB0` push button on the Arria 10 FPGA development board. The soft reset option is executed by issuing the reset command via the terminal console in the Nios II SBT (Software Build Tools) for Eclipse tool. When you assert a hard reset, the reset sequencer cycles through all the various module resets based on a pre-set sequence. The figure below illustrates the sequence and also shows how the reset sequencer output ports correspond to the modules that are being reset.

Figure 4: Reset Sequence**Related Information**

- [Setting Up the Software Command Line Environment](#) on page 17
- [Quartus II Handbook Volume 1: Design and Synthesis](#)
For detailed description of the reset sequencer module.
- [Arria 10 Transceiver PHY User Guide](#)
For detailed description of the transceiver PHY reset controller and ATX PLL.

Transceiver PHY Reset Controller and ATX PLL

The transceiver PHY reset controller takes the transceiver PHY reset output from the reset sequencer and generates the proper analog and digital reset sequencing for the transceiver PHY module. The ATX PLL supplies a low-jitter serial clock to the transceiver PHY module.

Related Information

- [Quartus II Handbook Volume 1: Design and Synthesis](#)
For detailed description of the reset sequencer module.
- [Arria 10 Transceiver PHY User Guide](#)
For detailed description of the transceiver PHY reset controller and ATX PLL.

JESD204B Subsystem Address Map

You can access the address mapping of the submodules in the JESD204B subsystem by clicking on the **Address Map** tab in the QSYS window. The memory allocation address map is described in the table below.

Table 3: JESD204B Subsystem Address Map

Avalon-MM Peripheral	Address Map
JESD204B IP core transceiver reconfiguration interface	0x0000 – 0x3FFF
ATX PLL (up to 4 modules per link)	0x8000 – 0x8FFF (Module 0) 0x9000 – 0x9FFF (Module 1) 0xA000 – 0xAFFF (Module 2) 0xB000 – 0xBFFF (Module 3)
JESD204B IP core CSR – TX	0xC000 – 0xC3FF
JESD204B IP core CSR – RX	0xD000 – 0xD3FF
Reset sequencer	0xE000 – 0xE0FF

Nios II Subsystem

The Nios II subsystem Qsys project, *nios_subsystem.qsys*, instantiates the following peripherals:

- Nios II processor
- On-chip memory—provides both instruction and data memory space
- Timer—provides a general timer function for software
- JTAG UART—serves as the main communications portal between the user and the Nios II processor via the terminal console in NIOS II SBT for Eclipse tool
- Avalon-MM bridges
- PIO—provides general input/output (I/O) access from the Nios II processor to the HDL components in the FPGA via two sets of 32-bit registers:
 - `io_status` (status registers input from the HDL components to NIOS-II)
 - `io_control` (control registers output from NIOS-II to the HDL components)

The tables below describe the signal connectivity for the `io_status` and `io_control` registers.

Table 4: Signal Connectivity for `io_status` Registers

Bit	Signal
0	Core PLL locked
1	TX transceiver ready (Link 0)
2	RX transceiver ready (Link 0)
3	Test pattern checker data error (Link 0)
4–31	TX transceiver ready, RX transceiver ready, and test pattern checker data error signals for subsequent links, if present.

Table 5: Signal Connectivity for `io_control` Registers

Bit	Signal
0	RX serial loopback enable for lane 0 (Link 0)
1	RX serial loopback enable for lane 1 (Link 0)
2	RX serial loopback enable for lane 2 (Link 0)
3	RX serial loopback enable for lane 3 (Link 0)
4–30	RX serial loopback enable for subsequent links, if present.
31	Sysref

Nios II Subsystem Address Map

You can access the address mapping of the peripherals in the Nios II subsystem by clicking on the **Address Map** tab in the Qsys window.

Figure 5: Address Map View in Qsys

System Contents Address Map Interconnect Requirements			
System: jesd204b_ed_qsys.nios_subsystem			
		nios2.data_master	nios2.instruction_master
core_pll_spi_mm_bridge_0.s0		0x0010_0000 - 0x0010_0fff	
jesd204b_subsystem_mm_brid...		0x0000_0000 - 0x000f_ffff	
jtag_uart_avalon_jtag_slave		0x0040_1030 - 0x0040_1037	
lcd.control_slave		0x0040_1020 - 0x0040_102f	
memory.s1		0x0020_0000 - 0x0022_7fff	0x0020_0000 - 0x0022_7fff
nios2.debug_mem_slave		0x0040_0800 - 0x0040_0fff	0x0040_0800 - 0x0040_0fff
pio_control.s1		0x0040_1040 - 0x0040_104f	
pio_status.s1		0x0040_1050 - 0x0040_105f	
timer.s1		0x0040_1000 - 0x0040_101f	

Reference Design Files

The reference design comes with a ZIP file that includes a Quartus II archive file (.qar) that contains:

- Quartus II project files
- source files
- Qsys projects
- ancillary files
- system libraries

The ZIP file also contains the software C code that you can use to compile and download into the Nios II processor control unit. When restoring the archive file in the Quartus II software, create a destination folder and name it `jесd204b_ed` to store all the reference design files.

Table 6: Reference Design Files

This table lists the important folders and files that you may need to edit to customize the design.

File Type	File/Folder	Description
Quartus project files	jесd204b_ed.qpf	Quartus project file.
	jесd204b_ed.qsf	Quartus settings file.
Verilog HDL design files	jесd204b_ed.sv	Top level HDL.
	jесd204b_ed.sdc	Synopsys Design Constraints (SDC) file containing all timing/placement constraints.
	transport_layer	Transport layer folder containing assembler and de-assembler HDL.
	pattern	Pattern folder containing the test pattern generator and checker HDL. Contains HDL for loopback version and AD9680 test pattern generator and checker (files with <code>_AD9680</code> suffix).
QSYS Projects	jесd204b_ed_qsys.qsys	Top level QSYS system project.
	jесd204b_subsystem.qsys	JESD204B subsystem (refer to JESD204B Subsystem)
	nios_subsystem.qsys	Nios II subsystem (refer to Nios II Subsystem)
Software files	software	Folder containing all software-related files (detailed description in Table 7).

Table 7: Software File Directory

File Type	File	Description
Header files	altera_jesd204_qsys_regs.h	Offsets, masks, and bit position definitions for peripherals in QSYS system that do not have standard access libraries. This includes the following peripherals: <ul style="list-style-type: none"> • JESD204B TX and RX CSR • Reset sequencer • PIO control • PIO status • Core PLL reconfiguration
	altera_xcvr_atx_pll_a10_reconfig_parameters.h	RAM arrays that contain the dynamic reconfiguration address offsets and data values for the ATX PLL reconfiguration registers.
	altera_xcvr_native_a10_reconfig_parameters.h	RAM arrays that contain the dynamic reconfiguration address offsets and data values for the transceiver native PHY IP core reconfiguration registers.
	main.h	General user parameter definitions.
	functions.h	Contains function prototype definitions of sub-functions in main.c.
	rules.h	Contains function prototype definitions of rule functions in rules.c.
	macros.h	Contains function prototype definitions of macro functions in macros.c.
Source files	main.c	Main C program. Also contain sub functions.
	rules.c	Rule checking functions used by the dynamic reconfiguration functions.
	macros.c	JESD204B QSYS system device access macros.

Related Information

- [Sub Functions in main.c Source File](#) on page 32
- [Functions in rules.c Source File](#) on page 36
- [Custom Peripheral Access Macros in macros.c Source File](#) on page 38

FPGA Pin Assignments

The top level signals with its corresponding FPGA pin assignments on the Arria 10 FPGA development board are listed in the table below.

Table 8: Top Level Signals and the Corresponding FPGA Pins

Top Level Signal Name	FPGA Pin Number	I/O Standard	Direction
global_rst_n	T12	1.8 V	Input
device_clk	W8	LVDS	Input
device_clk (n)	W7	LVDS	Input
mgmt_clk	BD32	1.8 V	Input
sma_clkout	E24	1.8 V	Output
spi_MISO	A17	1.8 V	Input
spi_MOSI	F18	1.8 V	Output
spi_SCLK	G18	1.8 V	Output
spi_SS_n[0]	G21	1.8 V	Output
tx_serial_data[3]	Y1	High speed differential I/O	Output
tx_serial_data[3] (n)	Y2	High speed differential I/O	Output
tx_serial_data[2]	AB1	High speed differential I/O	Output
tx_serial_data[2] (n)	AB2	High speed differential I/O	Output
tx_serial_data[1]	V1	High speed differential I/O	Output
tx_serial_data[1] (n)	V2	High speed differential I/O	Output
tx_serial_data[0]	T1	High speed differential I/O	Output
tx_serial_data[0] (n)	T2	High speed differential I/O	Output
rx_serial_data[3]	W3	High speed differential I/O	Input
rx_serial_data[3] (n)	W4	High speed differential I/O	Input
rx_serial_data[2]	AA3	High speed differential I/O	Input
rx_serial_data[2] (n)	AA4	High speed differential I/O	Input
rx_serial_data[1]	Y5	High speed differential I/O	Input
rx_serial_data[1] (n)	Y6	High speed differential I/O	Input
rx_serial_data[0]	V5	High speed differential I/O	Input
rx_serial_data[0] (n)	V6	High speed differential I/O	Input
sysref_out	J19	LVDS	Output
sysref_out (n)	K19	LVDS	Output
sync_n_out	C15	LVDS	Output
sync_n_out (n)	B15	LVDS	Output

Generating Programming File

To compile the HDL and generate the programming file (.sof), follow these steps:

1. Extract the reference design from `jesd204b_nios_ref_design.zip`.
2. Launch the Quartus II software.
3. On the File menu, click **Open Project**.
4. Navigate to your project directory and select the Quartus project archive file (`jesd204b_ed.qar`). Click **Open**.
5. In the **Restore Archived Project** window, verify that the archive file name is `jesd204b_ed.qar` and set the destination folder to `<project directory>/jesd204b_nios_ref_design/jesd204b_ed/`. Click **OK**. The Quartus project is now open in the Quartus window.
6. In the top level HDL file, `<project directory>/jesd204b_nios_ref_design/jesd204b_ed/jesd204b_ed.sv`, set the project parameter settings according to your configuration. Ensure that the settings match the JESD204B parameter settings in QSYS. Refer to the Top Level HDL Parameters section for more details on the project parameters
7. To compile the HDL, navigate to the **Processing** menu and select **Start Compilation**.
8. After compilation is done, you are ready to program the FPGA device with the programming file. Navigate to the **Tools** menu and click **Programmer**.
9. In the **Programmer** window, click **Add File**.
10. In the **Select Programming File** window, navigate to `<project directory>/jesd204b_nios_ref_design/jesd204b_ed/output_files/jesd204b_ed.sof` and click **Open**.
11. Verify that all the hardware setup options are set correctly to your system configurations.
12. Click **Start** to download the file into the Arria 10 FPGA device on the development board.

Alternatively, if you want to use the pre-generated golden programming file, skip the Quartus compilation in step 6 and 7. In step 10, select `<project directory>/jesd204b_nios_ref_design/jesd204b_ed/output_files/jesd204b_ed_golden.sof` and proceed accordingly.

After programming the Arria 10 FPGA device on the development board, the system needs to be initialized via software before the link can be fully active. Follow the steps in the Setting Up the Software Command Line Environment section to complete the link initialization process.

Related Information

[Setting Up the Software Command Line Environment](#) on page 17

Top Level HDL Parameters

The top level HDL file (`jesd204b_ed.sv`) located in the `<project directory>/jesd204b_nios_ref_design/jesd204b_ed/` directory includes project parameters that define the configuration of the reference design as a whole. The parameters are described in the table below.

Table 9: Top Level Parameters

Parameter	Description
LINK	Number of JESD204B links. Set to match the configuration in QSYS.

Parameter	Description
L	Number of JESD204B lanes per converter device. Set to match the configuration in QSYS.
M	Number of JESD204B converters per device. Set to match the configuration in QSYS.
F	Number of JESD204B octets per frame. Set to match the configuration in QSYS.
N	Number of JESD204B conversion bits per converter device. Set to match the configuration in QSYS.
N_PRIME	Number of JESD204B transmitted bits per sample. Set to match the configuration in QSYS.
S	Number of JESD204B transmitted samples per converter device per frame. Set to match the configuration in QSYS.
CS	Number of JESD204B control bits per conversion sample. Set to match the configuration in QSYS.
F1_FRAMECLK_DIV	Divider ratio for <code>frame_clk</code> when F=1. Refer to the JESD204B IP Core User Guide for more details.
F2_FRAMECLK_DIV	Divider ratio for <code>frame_clk</code> when F=2. Refer to the JESD204B IP Core User Guide for more details.
POLYNOMIAL_LENGTH	Defines the polynomial length for the PRBS pattern generator and checker. Refer to the JESD204B IP Core User Guide for more details.
FEEDBACK_TAP	Defines the feedback tap for the PRBS pattern generator and checker. Refer to the JESD204B IP Core User Guide for more details.

Related Information

[JESD204B IP Core User Guide](#)

Setting Up the Software Command Line Environment

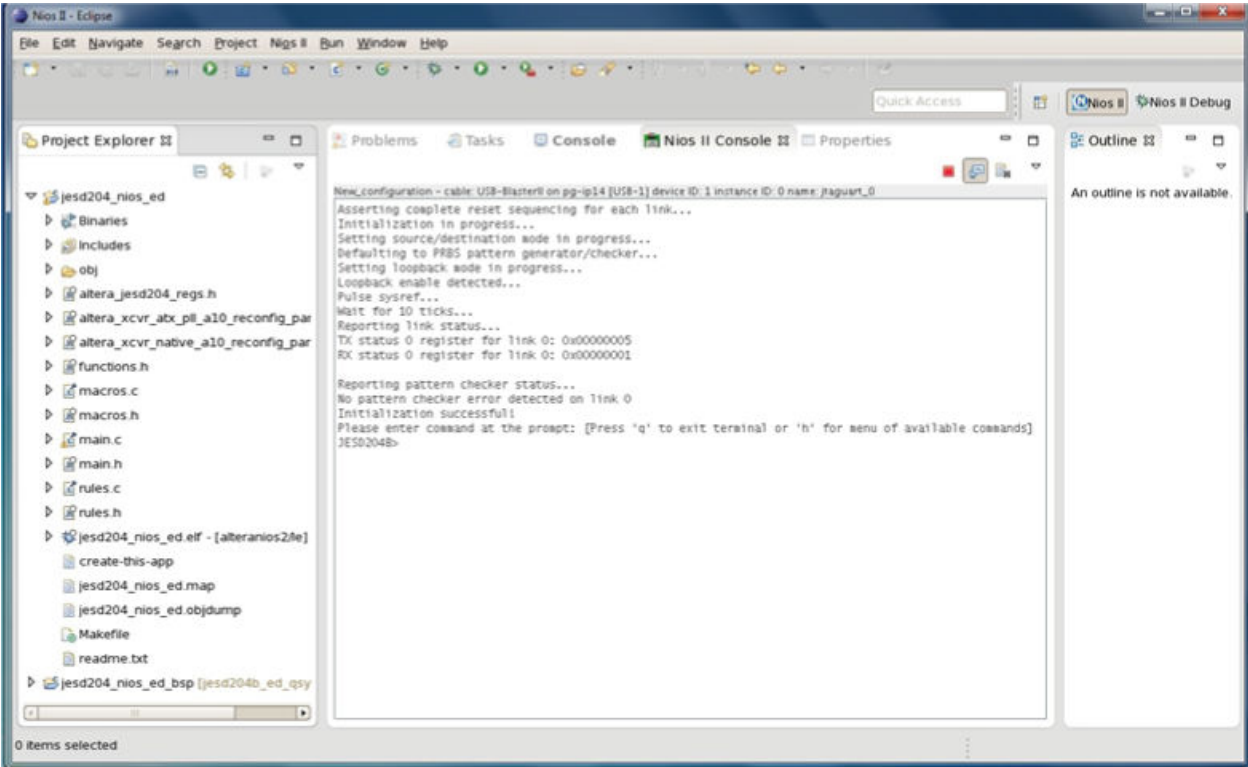
The primary user interface to the Nios II control unit is through a terminal console that transmit user commands to the Nios II processor via the JTAG UART peripheral. Altera provides the Nios II software build tools (SBT) via the Eclipse Integrated Development Environment (IDE). You can use this tool to view, edit, compile the source code, download the executable code to hardware, and generate a terminal console window for you to enter commands.

1. Launch the Quartus II software.
2. Compile and download the programming file onto the hardware platform as described in [Generating Programming File](#) section.
3. In the `<project directory>/jesd204b_nios_ref_design/jesd204b_ed/` directory, create a new folder named `software`.
4. On the **Tools** menu, select **Nios II Software Build Tools for Eclipse**.
5. In the **Select a workspace** dialog box, navigate to the software workspace, `<project directory>/jesd204b_nios_ref_design/jesd204b_ed/software` and click **OK**.

6. Create a new Nios II application and board support package (BSP) from the template. On the **File** menu, navigate to **New** and click on **Nios II Application and BSP From Template**.
7. In the **Nios II Application and BSP From Template** window, enter the following information:
 - SOPC Information File Name: `<project directory>/jesd204b_nios_ref_design/jesd204b_ed/jesd204b_ed_qsys.sopcinfo`
 - Project name: **jesd204_nios_ed**
 - Use default location: Checked
 - Templates: **Hello World**
8. Click **Next**. Verify that the default BSP name is **jesd204_nios_ed_bsp**, then click **Finish**. The Nios II application project (**jesd204_nios_ed**) and BSP (**jesd204_nios_ed_bsp**) appears in the **Project Explorer** window.
9. Whenever you modify and recompile the Quartus II project, the BSP must be regenerated. In the **Project Explorer** window, right-click on the **jesd204_nios_ed_bsp** project, navigate to **Nios II** and click **Generate BSP**. This regenerates the BSP based on your most current compiled Quartus II project settings.
10. Copy all the C source files (*.c) and header files (*.h) from `<project directory>/jesd204b_nios_ref_design/software` to `<project directory>/jesd204b_nios_ref_design/jesd204b_ed/software`. Delete the **hello_world.c** source file from the `<project directory>/jesd204b_nios_ref_design/jesd204b_ed/software` folder.
11. To compile the C code, navigate to the **Project** menu and select **Build All**. The compiler now compiles the C code into executable code.
12. To download the executable code to the hardware, navigate to the **Run** menu and select **Run Configurations**. In the **Run Configurations** window, double click **Nios II Hardware** on the left panel. Double check that all run configurations are correct then click **Run** on the bottom right corner.

The tool downloads the executable code onto the hardware and executes the code. The code initializes the link and opens up a terminal console with a prompt for you to enter commands to the Nios II processor.

Figure 6: Nios II Software Command Line Terminal



User Commands

The table below describes the commands that you can issue at the terminal console.

Table 10: User Commands

Type	Command	Description
Help	h	Display menu of available commands.

Type	Command	Description
Reset	r [link select #] [p x c l f h r]	<p>Selective/global soft system reset. The [link select #] option selects the link that the command will take effect on. If no [link select #] option is indicated, the command will take effect on all links identically. The [p x c l f] options indicate the specific submodule that the reset command will take effect on:</p> <p>[p] – Core PLL [x] – TX/RX Transceivers (JESD204B IP core PHY) [c] – TX/RX JESD204B IP core CSR [l] – TX/RX link reset [f] – TX/RX frame reset</p> <p>If none of the options above are indicated, all submodules are reset. You can indicate multiple options simultaneously to perform simultaneous submodule resets.</p> <p>The [h r] options indicate if the reset is asserted and held or released from a hold:</p> <p>[h] – Assert and hold reset [r] – Release reset</p> <p>If none of the options above are indicated, the resets are pulsed (asserted and released automatically).</p> <p>When the [r] option is indicated, the reset is released immediately without checking for any qualifying conditions (for example, the PLL locked or transceiver ready signals). You are responsible to qualify the signals before holding or releasing the resets.</p>

Type	Command	Description
Reset	<code>r [link select #] [p x c l f h r]</code>	<p>(continued)</p> <p>Depending on the <code>DATAPATH</code> setting in the <code>main.h</code> file, the command takes effect on either the TX only datapath (if <code>DATAPATH</code> is set to TX only), the RX only datapath (if <code>DATAPATH</code> is set to RX only) or both TX and RX datapaths (if <code>DATAPATH</code> is set to duplex).</p> <p>Command examples:</p> <ul style="list-style-type: none"> <code>> r</code> : This is a full auto-mode global reset that performs a series of initialization steps after global reset. The sequence of steps are: <ol style="list-style-type: none"> 1. Global system reset on all links according to the hardware reset sequence (refer to Figure 4 for timing diagram of hardware reset sequence). 2. Initialize link (set loopback and source/destination mode to default mode). 3. Trigger SYSREF pulse (for Subclass 1 configuration). 4. Wait 10 seconds. 5. Report link status. <code>> r 2 h</code> : Assert all submodule resets for link 2 and hold resets indefinitely. <code>> r r</code> : Release all submodules resets for all links immediately and concurrently (no pre-set hardware sequence) <p>Attention: Not recommended as no checking process for qualifying signals.</p> <ul style="list-style-type: none"> <code>> r x l f h</code> : Assert and hold transceiver, link, and frame resets for all links. <code>> r x r</code> : Release (deassert) transceiver resets for all links (but keep other submodule resets on hold if previously held) <code>> r 2 l f</code> : Pulse (assert and release) link and frame resets for link 2 (but keep other submodule resets on hold if previously held)
Load SPI	<code>ls [slave select #] <offset> <value></code>	Loads 8-bit value <value> (in C-style “0x” hexadecimal notation) into the register of external converter module connected to the SPI interface at offset <offset> (in C-style “0x” hexadecimal notation) indicated by [slave select #]. The maximum bit width of <offset> is 16 bits.
Get SPI	<code>gs [slave select #] <offset></code>	Gets the 8-bit value (in C-style “0x” hexadecimal notation) at the register of external converter module connected to SPI interface at offset <offset> (in C-style “0x” hexadecimal notation) indicated by [slave select #]. The maximum bit width of <offset> is 16 bits.
Config SPI	<code>cs</code>	Configure external converter modules via SPI interface.

Type	Command	Description
Initialize	i [link select #] [n]	<p>Initialize link indicated by [link select #]. The [link select #] option selects the link that the command will take effect on. If no [link select #] option is indicated, the command takes effect on all links identically. By default, the link is initialized to test mode (see 'Test' command). The optional [n] option initializes the link to user mode (see 'Test' command).</p> <p>Note: When setting to user mode, ensure that the user input data valid signal (<code>avst_usr_din_valid</code>) is properly connected in the top level HDL file (<code>jesd204b_ed.sv</code>). Failing to do so will result in continuous error interrupts and may cause the system to hang.</p> <p>The full sequence of steps executed by this command:</p> <ol style="list-style-type: none"> 1. Set test or user mode (set loopback and source/destination mode) as per options indicated. 2. Trigger SYSREF pulse (for Subclass 1 configuration). 3. Wait 10 seconds. 4. Report link status.
Status	s [link select #] [t r]	<p>Reports TX and/or RX link status of link indicated by [link select #]. Reads back <code>tx_status0</code> and/or <code>rx_status0</code> status registers from the JESD204B CSR identified by [link select #] and reports link status based on the values in those registers. For example, the command can report that the indicated link is not in sync or not in User Data Mode. In addition, the command reports the status of the pattern checker error signal. The [link select #] option selects the link that the command will take effect on. If no [link select #] option is indicated, the command takes effect on all links identically. The optional [t r] options indicates the datapath to be reported:</p> <p>[t] – Report TX path status only</p> <p>[r] – Report RX path status only</p> <p>[no option] – Report default path status according to DATAPATH setting</p> <p>Depending on the DATAPATH setting in the <code>main.h</code> file, the command will take effect on either the TX only datapath (if DATAPATH is set to TX only), the RX only datapath (if DATAPATH is set to RX only) or both TX and RX datapaths (if DATAPATH is set to duplex).</p>
Loopback	lb [link select #] [n]	<p>Puts the JESD204B IP core PHY indicated by [link select #] into transceiver serial loopback mode. This command is only applicable for links where the JESD204B IP core is configured in duplex mode (TX and RX datapaths present). The [link select #] option selects the link that the command will take effect on. If no [link select #] option is indicated, the command takes effect on all links identically. The optional [n] option sets the JESD204B IP core PHY indicated by [link select #] into non-serial loopback mode.</p>

Type	Command	Description
Source/ Destination	sd [link select #] [s d] [u a r p]	<p>Selects the source and destination datapath for the Avalon-ST interface of the link indicated by [link select #]. The [link select #] option selects the link that the command will take effect on. If no [link select #] option is indicated, the command takes effect on all links identically. The optional [s d] option indicates whether the source (TX) or destination (RX) datapath is being selected:</p> <p>[s] – Source datapath (TX)</p> <p>[d] – Destination datapath (RX)</p> <p>[no option] – Default datapath according to DATAPATH setting</p> <p>Depending on the DATAPATH setting in the <code>main.h</code> file, the command will take effect on either the TX only datapath (if DATAPATH is set to TX only), the RX only datapath (if DATAPATH is set to RX only) or both TX and RX datapaths (if DATAPATH is set to duplex).</p> <p>The optional [u a r p] option indicates the type of datapath to set to:</p> <p>Note: When setting to user mode, ensure that the user input data valid signal (<code>avst_usr_din_valid</code>) is properly connected in the top level HDL file (<code>jesd204b_ed.sv</code>). Failing to do so will result in continuous error interrupts and may cause the system to hang.</p> <p>[u] – User datapath (no test pattern generator and checker)</p> <p>[a] – Test pattern generator and checker set to alternate pattern</p> <p>[r] – Test pattern generator and checker set to ramp pattern</p> <p>[p] – Test pattern generator and checker set to PRBS pattern</p> <p>[no option] – Default to test pattern generator and checker set to PRBS pattern</p> <p>The test pattern generator and checker module has other configurable parameters that you can set. In particular, the pattern generator and checker derives its M and S values from the JESD204B IP core CSR. Also, there are other parameters such as <code>POLYNOMIAL_LENGTH</code> and <code>FEEDBACK_TAP</code> that are set during compile time. Refer to Chapter 5 of the JESD204B IP Core User Guide for more details.</p>

Type	Command	Description
Test	t [link select #] [n]	<p>Sets the link indicated by [link select #] to test mode. The [link select #] option selects the link that the command will take effect on. If no [link select #] option is indicated, the command takes effect on all links identically. The test mode is defined by:</p> <ul style="list-style-type: none"> Source and destination datapath set to default (test pattern generator and checker set to PRBS pattern) Transceiver set to serial loopback mode <p>The optional [n] option negates the test mode (set to user mode). User mode is defined as:</p> <ul style="list-style-type: none"> Source and destination datapath set to user datapath Transceiver set to non-serial loopback mode <p>Note: When setting to user mode, ensure that the user input data valid signal (avst_usr_din_valid) is properly connected in the top level HDL file (jesd204b_ed.sv). Failing to do so will result in continuous error interrupts and may cause the system to hang.</p>
Reinitialization	ri [link select #] [t r]	<p>Trigger a link reinitialization on links indicated by [link select #]. The optional [link select #] option selects the link that the command will take effect on. If no [link select #] option is indicated, the command takes effect on all links identically. The optional [t r] option indicates the type of reinitialization operation:</p> <p>[t] – TX link reinitialization; TX link transmits K28.5 packets continuously until link is out of CGS (Code Group Synchronization) phase</p> <p>[r] – RX link reinitialization; SYNC_N signal is driven low until link is out of CGS (Code Group Synchronization)</p> <p>Note: Forcing RX link reinitialization will trigger a TX SYNCN error interrupt to the Nios II processor. The interrupt is automatically cleared by the software.</p> <p>[no option] – For DATAPATH setting set to TX only or duplex, default to TX link reinitialization. For DATAPATH setting set to RX only, default to RX link reinitialization</p> <p>For more details on the reinitialization operation, refer to the JESD204B IP Core User Guide.</p>
Sysref	sy	Pulse SYSREF signal one time (“one-shot”)

Type	Command	Description
Reconfiguration	rc [link select #] [l <value>] [m <value>] [f <value>] [s <value>] [n <value>] [np <value>] [cs <value>] [k <value>] [hd <value>] [scr <value>] [sub <value>] [dr <value>]	<p>Dynamically reconfigure link indicated by [link select #] according to parameter-value pair option indicated. The optional [link select #] option selects the link that the command will take effect on. If no [link select #] option is indicated, the command takes effect on all links identically. The parameters that are dynamically configurable are as follows:</p> <p>[l] – Lanes per converter device [m] – Converters per device [f] – Octets per frame</p> <p>Note: The Altera test pattern generator and checker do not support dynamic reconfiguration of F.</p> <p>[s] – Samples per converter per frame [n] – Converter resolution [np] – Transmitted bits per sample [cs] – Control bits [k] – Frames per multi-frame [hd] – High density user data format [scr] – Enable scrambler [sub] – Subclass select [dr] – Serial data rate</p> <p>The valid value ranges that can be entered for each parameter are governed by rules enforced by software. Refer to the Dynamic Reconfiguration section for more details.</p> <p>Command examples:</p> <ul style="list-style-type: none"> > rc l 2 m 2 f 2 : Reconfigure the JESD204B IP core to L=2, M=2, F=2. > rc np 16 hd 1 : Reconfigure the JESD204B IP core to N'=16, HD=1. > rc dr 2 : Reconfigure the link serial data rate to the initially configured data rate divide by 2.

Dynamic Reconfiguration

One of the key features enabled by the Nios II processor is dynamic reconfiguration of the JESD204B parameters. You can issue the reconfiguration command (“rc”) along with the parameters and values that you desire. The valid ranges for the values entered for each parameter are governed by certain rules. These rules are a function of the JESD204B IP core and transport layer valid parameter value ranges (as described in the JESD204B IP Core User Guide) and also the initially configured values for the JESD204B IP core. The rules for valid ranges are implemented as discrete functions in the `rules.c` file (see

Functions in rules.c Source File) and are described fully in the table below. If you enter an invalid value, the software flags the error to the screen and disallow the change.

The Altera transport layer has a more restricted value range for certain parameters compared to the JESD204B IP core. In those cases where the parameter value ranges are governed by both the JESD204B IP core and the Altera transport layer, the more restrictive value range will take precedence. In the table below, the transport layer rules are indicated by “TL:”. If you are not using the Altera transport layer, turn off the Altera transport layer rule checking by setting the ALTERA_TRANSPORT_LAYER parameter in the main.h header file to 0 (see the Customizing the C Code section).

Table 11: Dynamic Reconfiguration Command Options and Rules

Command Options	Parameters	Rule
l	L	The value for L must be an integer within the range of 1-8. Returns 0 if valid, 1 if invalid.
l	L	The value for L must not exceed the initially configured value. Returns 0 if valid, 1 if invalid.
l	L, F	TL: The value for L must be an even number if F = 1. Returns 0 if valid, 1 if invalid.
m	M	The value for M must be an integer within the range of 1-32. Returns 0 if valid, 1 if invalid.
f	F	The value for F must be an integer within the range of 1,2, 4-256 (any integer value between 1-256 except 3) TL: The value for F must be an integer of the values 1, 2, 4, 8 Returns 0 if valid, 1 if invalid. Note: The Altera test pattern generator and checker do not support dynamic reconfiguration of F.
f, m, s, np, l	F, M, S, N', L	The values for M, S, N' and L must be such that the current value of F conforms to the formula $F(\text{current}) = (M * S * N') / (8 * L)$. If a new value of F is indicated, then $F(\text{current}) = F(\text{new})$. If not, then $F(\text{current}) = F(\text{initially configured})$. Returns 0 if valid, 1 if invalid. Note: The Altera test pattern generator and checker do not support dynamic reconfiguration of F.

Command Options	Parameters	Rule
f, m, s, np, l	F, M, S, N', L	<p>The value for F must not exceed the initially configured value. By extension, since $F = (M * S * N') / (8 * L)$, the values for M, S, N' and L must be such that the new value for F not exceed the initial configured value.</p> <p>Returns 0 if valid, 1 if invalid.</p> <p>Note: The Altera test pattern generator and checker do not support dynamic reconfiguration of F.</p>
s	S	<p>The value for S must be an integer within the range of 1-32.</p> <p>Returns 0 if valid, 1 if invalid.</p>
n	N	<p>The value for N must be an integer within the range of 1-32</p> <p>TL: The value for N must be an integer within the range of 12-16.</p> <p>Returns 0 if valid, 1 if invalid.</p>
n, np	N	<p>The value for N must adhere to the following range: $N \leq N'$.</p> <p>Returns 0 if valid, 1 if invalid.</p>
np	N'	<p>The value for N' must be an integer within the range of 4-32.</p> <p>TL: Only N'=16 configuration is supported. Dynamic reconfiguration of N' parameter is not supported.</p> <p>Returns 0 if valid, 1 if invalid.</p>
cs	CS	<p>The value for CS must be an integer within the range of 0-3.</p> <p>Returns 0 if valid, 1 if invalid.</p>
k	K	<p>The value for K must be an integer within the range $17/F \leq K \leq \min(32, \text{floor}(1024/F))$.</p> <p>Returns 0 if valid, 1 if invalid.</p>
f, k	F, K	<p>The value of $F * K$ must be divisible by 4.</p> <p>Returns 0 if valid, 1 if invalid.</p>
hd	High Density (HD)	<p>The value for HD must be either 0 or 1.</p> <p>Returns 0 if valid, 1 if invalid.</p>
hd, n	HD, N	<p>TL: The value for HD can be 1 if and only if N=16.</p> <p>Returns 0 if valid, 1 if invalid.</p>

Command Options	Parameters	Rule
scr	Scrambler Enable	The value for SCR must be either 0 or 1. Returns 0 if valid, 1 if invalid.
sub	Subclass	The value for subclass must be 0, 1 or 2. Returns 0 if valid, 1 if invalid.
dr	Serial Data Rate	The value for DR must be 1, 2, 4. DR value is integer divisor of initially configured serial data rate. For example, DR=2 means the initially configured data rate divided by 2. Returns 0 if valid, 1 if invalid.
dr	Serial Data Rate	The value for DR must not fall below the minimum allowable range of the target device family. Minimum serial data rate specification: <ul style="list-style-type: none"> Arria V GZ, Arria 10, Stratix V: 2 Gbps Arria V: 1 Gbps Refer to the JESD204B IP Core User Guide for the latest information on the minimum serial data rate spec. The minimum data rate value is defined by the <code>DATA_RATE_MIN</code> parameter in the <code>main.h</code> header file. You are responsible to ensure that the minimum data rate value for the target device is set correctly. Returns 0 if valid, 1 if invalid.
dr, m, s, np, l	Frame clock, serial data rate, M, S, N', L	The values of serial data rate, M, S, N' and L must be such that the new frame clock value, $FC(new) = (Serial\ data\ rate * L * 8) / (M * S * N' * 10)$ not exceed the initially configured frame clock value, FC (initially configured). Furthermore, the ratio of FC (initially configured) to FC (new) must be 1, 2, 4 only. Returns 0 if valid, 1 if invalid.

Related Information

- [Functions in rules.c Source File](#) on page 36
- [Customize the C code](#) on page 30

Data Rate Reconfiguration

The existing reference design supports dynamic data rate reconfiguration for initially configured serial data rate of 6144 Mbps. At this data rate, you can dynamically reconfigure the data rate to half of the initially configured value (3072 Mbps). However, you are not able to reconfigure to a quarter of the initially configured value even though this is a valid option because the resulting data rate (1536 Mbps) falls below the minimum data rate allowed by the JESD204B IP core and is disallowed by the software.

Dynamically reconfiguring the transceiver data rate requires reading in pre-generated configuration files that are specifically generated for a certain transceiver configuration (including the data rate).

Customizing the Data Rate Reconfiguration Parameters

To customize the design to support a different transceiver configuration, follow the steps below:

1. Follow the steps in the [JESD204B Subsystem](#) section to open the *jesd204b_subsystem* module and enter your desired serial data rate in the **Data Rate** box.
2. Turn on the **Enable Transceiver Dynamic Reconfiguration** option. On the **File** menu, select **Save** to save the settings.
3. Click the **Generate HDL** button on the bottom right corner of the Qsys window.
In addition to generating the Qsys system HDL, Qsys also generates the transceiver configuration data stored as an array in a C header file.
4. After Qsys is done generating the HDL, click the **Finish** button on the bottom right corner of the Qsys window.
5. In your shell, navigate to the Qsys-generated C header file location: `<project directory>/jesd204b_nios_ref_design/jesd204b_ed/jesd204b_ed_qsys/altera_xcvr_native_a10_141/synth/reconfig/altera_xcvr_native_a10_reconfig_parameters.h`.
6. Open the *altera_xcvr_native_a10_reconfig_parameters.h* file and copy the array `'unsigned int altera_xcvr_native_a10_ram_array[120] = {...}'`
7. Navigate to the project transceiver reconfiguration parameters header file location: `<project directory>/jesd204b_nios_ref_design/jesd204b_ed/software/jesd204_nios_ed/altera_xcvr_native_a10_reconfig_parameters.h`.
8. Paste the array `'unsigned int altera_xcvr_native_a10_ram_array[120] = {...}'` and rename the array to `'unsigned int altera_xcvr_native_a10_ram_array_1[120] = {...}'`
9. Repeat steps 1-8 above, this time entering the original data rate divided by 2 in the **Data Rate** option of the JESD204B IP parameter editor window in Qsys. Be aware of the minimum data rate limits of the JESD204B IP core that is targeted to the device family of your choice. Refer to the JESD204B IP Core User Guide for the latest information on the minimum serial data rate specification. Qsys generates the configuration data array representing the new data rate. Copy and paste the data array `unsigned int altera_xcvr_native_a10_ram_array[120] = {...}` from the `<project directory>/jesd204b_nios_ref_design/jesd204b_ed/jesd204b_ed_qsys/altera_xcvr_native_a10_141/synth/reconfig/altera_xcvr_native_a10_reconfig_parameters.h` header file to the `<project directory>/jesd204b_nios_ref_design/jesd204b_ed/software/jesd204_nios_ed/altera_xcvr_native_a10_reconfig_parameters.h` header file as `'unsigned int altera_xcvr_native_a10_ram_array_2[120] = {...}'`
10. Repeat steps 1-8 above, this time entering the original data rate divided by 4 in the **Data Rate** option of the JESD204B IP parameter editor window in Qsys. Be aware of the minimum data rate limits of the JESD204B IP core that is targeted to the device family of your choice. Refer to the JESD204B IP Core User Guide for the latest information on the minimum serial data rate specification. Qsys generates the configuration data array representing the new data rate. Copy and paste the data array `unsigned int altera_xcvr_native_a10_ram_array[120] = {...}` from the `<project directory>/jesd204b_nios_ref_design/jesd204b_ed/jesd204b_ed_qsys/altera_xcvr_native_a10_141/synth/reconfig/altera_xcvr_native_a10_reconfig_parameters.h` header file to the `<project directory>/jesd204b_nios_ref_design/`

```
jesd204b_ed/software/jesd204_nios_ed/altera_xcvr_native_a10_reconfig_
parameters.h header file as 'unsigned int
altera_xcvr_native_a10_ram_array_4[120] = {...}'
```

11. Repeat steps 1-10 above for the ATX PLL module (*xcvr_atx_pll_a10_0*). Make sure to turn on the **Generate C header file** option under the **Dynamic Reconfiguration** tab of the ATX PLL parameter editor window in QSYS.

The Qsys-generated C header file is found in location: <project_directory>/jesd204b_nios_ref_design/jesd204b_ed/jesd204b_ed_qsys/altera_xcvr_atx_pll_a10_141/synth/reconfig/altera_xcvr_atx_pll_a10_reconfig_parameters.h. The ATX PLL reconfiguration data array is named 'unsigned int altera_xcvr_atx_pll_a10_ram_array[11] = {...}'. The project ATX PLL reconfiguration parameters header file location is: <project_directory>/jesd204b_nios_ref_design/jesd204b_ed/software/jesd204_nios_ed/altera_xcvr_atx_pll_a10_reconfig_parameters.h. Rename the data arrays appropriately when copying between the two header file locations.

12. Repeat step 11 for the number of ATX PLL modules that are present in your design.

The steps above illustrate how you can customize your code to support the existing data rate reconfiguration function (for example, reconfigure to data rate divided by 2 and by 4) using an initially configured data rate of your choice. Modify the C code to further customize your code to support arbitrary data rate changes and even different transceiver configurations (refer to the Customize the C code section).

Customize the C code

You can customize the behavior of the C code via a set of user parameters found in the `main.h` header file.

Table 12: User Parameters in `main.h`

Parameter	Description
DEBUG_MODE	Set to 1 to print debug messages, else set to 0.
PRINT_INTERRUPT_MESSAGES	Set to 1 to print JESD204B error interrupt messages, else set to 0.
CONFIG_SPI	Set to 1 to configure external converters via SPI interface at start of <code>main.c</code> execution, else set to 0.
PATCHK_EN	Set to 1 when test pattern checker is included in the initial design configuration, else set to 0.
ALTERA_TRANSPORT_LAYER	Set to 1 when using Altera transport layer, else set to 0.
BONDED	Set to 1 when transceivers configured in bonded mode, set to 0 when transceivers configured in unbonded mode. Note: Serial data rate reconfiguration is only supported in unbonded mode (BONDED=0).

Parameter	Description
DATAPATH	Set to indicate JESD204B IP configuration: 1 – TX datapath only 2 – RX datapath only 3 – Duplex datapath (TX and RX datapath)
DATA_RATE_LINK_n	Set to indicate the initially configured serial data rate of link n in Mbps (for example, to set link 0 to 6144 Mbps, DATA_RATE_LINK_0=6144). For multi-link scenarios, add additional DATA_RATE_LINK_n parameters to the DR_init[] array in main.c.
DATA_RATE_MIN	Set to indicate the minimum serial data rate (in Mbps) supported by the JESD204B IP core for the following device families: <ul style="list-style-type: none"> • Arria V GZ, Arria 10, Stratix V: 2000 • Arria V: 1000
MAX_LINKS	Set to indicate the number of links in the design (for example, for dual link, set MAX_LINKS=2) Note: When using the design as-is, the maximum value of MAX_LINKS is 16. To increase the limit, redesign the address map in QSYS.
LINE_BUFFER	Sets the maximum number of characters that user can enter on command line.
MAX_NUM_OPTIONS	Sets the maximum number of options per command.
MAX_OPTIONS_CHAR	Sets the maximum number of characters per command option
XCVR_CFG_FILES_PER_LINK	Sets the maximum number of transceiver configuration files per link.
XCVR_PLL_PER_LINK	Sets the maximum number of transceiver TX PLLs (ATX PLL) per link. Current address map supports up to four PLLs. Refer to the Arria 10 Transceiver PHY User Guide for more details on the maximum number of TX transceiver PLLs required. Note: When using the design as-is, the maximum value of XCVR_PLL_PER_LINK is 4. To increase the limit, redesign the address map in QSYS.
F1_FRAMECLK_DIV	Set to the F1_FRAMECLK_DIV parameter as defined in the top level HDL file (jesd204b_ed.sv).
F2_FRAMECLK_DIV	Set to the F2_FRAMECLK_DIV parameter as defined in the top level HDL file (jesd204b_ed.sv).

You can further customize the software behavior by directly modifying the C code. A full description of the functions and macros used in the design is given at the end of this document.

Related Information

- [Sub Functions in main.c Source File](#) on page 32
- [Functions in rules.c Source File](#) on page 36
- [Custom Peripheral Access Macros in macros.c Source File](#) on page 38
- [Arria 10 Transceiver PHY User Guide](#)

Sub Functions in main.c Source File

The function prototypes of the sub functions listed in the table below can be found in the `functions.h` header file located in the `software` file directory.

Table 13: Sub Functions in main.c

Function Prototype	Description
<code>void chomp (char * <i>string</i>)</code>	Chomps trailing '\n' character from <i>string</i> .
<code>int StringIsNumeric (char * <i>string</i>)</code>	Tests whether <i>string</i> is numeric. Returns 1 if true, 0 if false.
<code>int StringIsHex (char * <i>string</i>)</code>	Tests whether <i>string</i> is a hexadecimal number. Returns 1 if true, 0 if false.
<code>void DelayCounter (alt_u32 <i>count</i>)</code>	Delay counter. Counts up to <i>count</i> ticks, each tick is roughly 1 second (not accurate).
<code>void Get_L_Init (int * <i>L_init</i>)</code>	Read initially configured value of L from each JESD204B IP core CSR in the design and store the value in <i>L_init</i> array.
<code>void Get_F_Init (int * <i>F_init</i>)</code>	Read initially configured value of F from each JESD204B IP core CSR in the design and store the value in <i>F_init</i> array.
<code>void Get_FC_Init (int * <i>FC_init</i> , int * <i>DR_init</i>)</code>	Calculate initially configured value of the frame rate by reading the relevant parameters from each JESD204B IP core CSR in the design and the serial data rate values stored in <i>DR_init</i> . Stores calculated value in <i>FC_init</i> array.
<code>int Initialize (char * <i>options</i> [], int * <i>held_resets</i>)</code>	Executes initialize command according to <i>options</i> . This function performs the following actions: <ul style="list-style-type: none"> • Set link to test mode (source or destination set to PRBS test pattern generator or checker, transceiver set to serial loopback mode) or negate • Pulse sysref • Wait 10 seconds • Report link status Returns 0 if success, 1 if fail, 2 if sync errors found, 4 if pattern checker errors found, 6 if both sync errors and pattern checker errors found.
<code>void Help (void)</code>	Prints menu of available commands.

Function Prototype	Description
int Reset (char * <i>options</i> [], int * <i>held_resets</i>)	<p>Executes reset command according to <i>options</i>. This function calls two other subfunctions depending on <i>options</i>:</p> <ul style="list-style-type: none"> Reset sequence (initiate full hardware reset sequence) Reset force (force individual resets high, low, or pulse depending on <i>options</i>) <p>Returns 0 if success, 1 if fail.</p>
int LoadSPI (char * <i>options</i> [])	<p>Executes the load SPI command according to <i>options</i>.</p> <p>Returns 0 if success, 1 if fail.</p>
int GetSPI (char * <i>options</i> [])	<p>Executes get SPI command according to <i>options</i>.</p> <p>Returns 0 if success, 1 if fail.</p>
int ConfigSPI (int * <i>held_resets</i> , int <i>dnr</i>)	<p>Executes the configure SPI command according to <i>options</i>.</p> <p>Returns 0 if success, 1 if fail.</p>
int Status (char * <i>options</i> [])	<p>Executes the report link status command according to <i>options</i>.</p> <p>Returns 0 if success, 1 if fail, 2 if sync errors found, 4 if pattern checker errors found, 6 if both sync errors and pattern checker errors found.</p>
int Loopback (char * <i>options</i> [], int * <i>held_resets</i> , int <i>dnr</i>)	<p>Executes the loopback command according to <i>options</i>.</p> <p>Returns 0 if success, 1 if fail.</p>
int SourceDest (char * <i>options</i> [], int * <i>held_resets</i> , int <i>dnr</i>)	<p>Executes the source or destination datapath selection command according to <i>options</i>.</p> <p>Returns 0 if success, 1 if fail.</p>
int Test (char * <i>options</i> [], int * <i>held_resets</i>)	<p>Executes the test mode command according to <i>options</i>. Test mode:</p> <ul style="list-style-type: none"> Set source or destination datapath selection to PRBS test pattern generator or checker. Set transceiver to serial loopback mode. <p>Returns 0 if success, 1 if fail.</p>
int Reinit (char * <i>options</i> [], int * <i>held_resets</i>)	<p>Executes the reinit command according to <i>options</i>. This function performs the following actions:</p> <ul style="list-style-type: none"> Write reinit values to the appropriate registers in the JESD204B IP core CSR to trigger reinit operation Pulse sysref Wait 10 seconds Report link status <p>Returns 0 if success, 1 if fail, 2 if sync errors found, 4 if pattern checker errors found, 6 if both sync errors and pattern checker errors found.</p>

Function Prototype	Description
void Sysref (void)	Pulse SYSREF signal one time (that is in "one-shot").
int Reconfig (char * options [], int * L_init, int * F_init, int * DR_init, int * FC_init, int * current_dr_div, unsigned int link_clk_pll_counter_val_init, unsigned int frame_clk_pll_counter_val_init, unsigned int * xcvr_native_array_ptr [], unsigned int * xcvr_pll_array_ptr [], int * held_resets)	<p>Executes dynamic reconfiguration command. This function performs three major tasks:</p> <ul style="list-style-type: none"> • Parses user options to identify the parameters and values to be dynamically reconfigured by user • Performs rule-checking to ensure all values entered conform to valid ranges • Performs read-modify-writes (RMW) to relevant CSR registers based on valid options entered by user <p>There are four categories of the CSR register RMWs performed by the software:</p> <ul style="list-style-type: none"> • Write new parameter values to the TX/RX JESD204B CSR <code>ilas_data1</code> and <code>ilas_data2</code> registers. • Write to TX/RX JESD204B CSR <code>lane_control_n</code> registers to power-down or power-up lanes in response to changes in L parameter. <p>Note: This feature is not fully implemented in the hardware.</p> <ul style="list-style-type: none"> • Write to the core PLL reconfiguration module in response to changes in link or frame clock data rate. • Write to the transceiver reconfiguration interface and/or ATX PLL reconfiguration interface in response to changes in transceiver serial data rate. <p>For more details on dynamic reconfiguration features, refer to the Dynamic Reconfiguration section.</p> <p>Returns 0 if success, 1 if fail.</p>
void ResetHard (void)	Triggers full hardware reset sequence via PIO control registers.
int ResetSeq (int link, int * held)	<p>Performs full hardware reset sequence on the <i>indicated link</i> via software interface .</p> <p>Returns 0 if success, 1 if fail.</p>
int ResetForce (int link, int reset_val, int hold_release, int * held_resets)	<p>Forces reset assertion or deassertion on submodule resets for the <i>link</i> indicated by <i>reset_val</i>. The function also decides whether to assert and hold (<i>hold_release</i> =2), deassert (<i>hold_release</i> =1) or pulse (<i>hold_release</i> =0) indicated resets. The function has mechanisms (using the global <i>held_resets</i> flag) to ensure that held resets that are not the target of the reset force function are not affected by it.</p> <p>Returns 0 if success, 1 if fail.</p>
int Reset_PLL_Release (int link, int * held_resets)	<p>Deassert the core PLL reset signal. Wait until the core PLL locked signal assert before returning.</p> <p>Returns 0 if success, 1 if fail.</p>



Function Prototype	Description
<code>int Reset_X_L_F_Release (int <i>link</i>, int * <i>held_resets</i>)</code>	Deassert the transceiver, link, and frame resets. This function deasserts the TX transceiver reset first, waits until the TX transceiver ready signal asserts, then deasserts the TX link and TX frame resets. The function then repeats the above actions with the RX data path. Returns 0 if success, 1 if fail.
<code>void spiWrite (alt_u16 <i>offset</i>, alt_u8 <i>m_data</i>, alt_u8 <i>slave</i>)</code>	Performs SPI write operation of <i>m_data</i> to SPI slave indicated by <i>slave</i> number at address offset <i>offset</i> . The maximum bit width of <i>m_data</i> is 8 bits while the maximum bit width of <i>offset</i> is 16 bits.
<code>alt_u8 spiRead (alt_u16 <i>offset</i>, alt_u8 <i>slave</i>)</code>	Performs SPI read operation of SPI slave indicated by <i>slave</i> number at address offset <i>offset</i> . Returns 8-bit read value. The maximum bit width of <i>offset</i> is 16 bits.
<code>void spiVerify (alt_u16 <i>offset</i> , alt_u8 <i>slave</i>, alt_u8 <i>data</i>)</code>	Performs SPI read operation of SPI slave indicated by <i>slave</i> number at address offset <i>offset</i> , compares the read data to <i>data</i> , then reports whether the read data matches the user-given <i>data</i> . The maximum bit width of <i>data</i> is 8 bits while the maximum bit width of <i>offset</i> is 16 bits.
<code>void Config_AD9680 (alt_u8 <i>slave</i>)</code>	Executes a series of <i>spiWrites</i> to the AD9680 slave to configure it.
<code>void InitISR (void)</code>	Initialize the interrupt controllers for the following peripherals: <ul style="list-style-type: none"> • JESD204B IP core TX CSR • JESD204B IP core RX CSR • SPI master The timer and JTAG UART interrupt controllers are disabled. Modify the function to enable it. ⁽¹⁾
<code>static void ISR_JESD_RX (void * <i>context</i>)</code>	The JESD204B IP core RX interrupt service routine (ISR). Upon an interrupt event (IRQ assert), reads the RX JESD204B CSR <i>rx_err0</i> and <i>rx_err1</i> registers and reports the error code to screen. After that, ISR clears all the valid and active status bits in the <i>rx_err0</i> and <i>rx_err1</i> registers. ⁽¹⁾
<code>static void ISR_JESD_TX (void * <i>context</i>)</code>	The JESD204B IP core TX ISR. Upon an interrupt event (IRQ assert), read the TX JESD204B CSR <i>tx_err</i> register and reports the error code to the screen. After that, the ISR clears all the valid and active status registers in the <i>tx_err</i> register. ⁽¹⁾
<code>static void ISR_SPI (void * <i>context</i>)</code>	The SPI master ISR. Upon an interrupt event (IRQ assert), clear the IRQ flag and return. ⁽¹⁾

Related Information[Nios II Software Developer's Handbook](#)

⁽¹⁾ Refer to the Nios II Software Developer's Handbook for more details on writing the interrupt service routine (ISR).

Functions in rules.c Source File

The rules enforced by the dynamic reconfiguration function to check validity of the reconfiguration values for each JESD204B parameter are coded as discrete functions in rules.c. In the table below, the transport layer rules are indicated by “TL:”. The function prototypes of the functions listed in the table below can be found in the `rules.h` header file located in the `software` file directory

Table 14: rules.c

Function Prototype	Parameters	Rule/Function Description
<code>int Min (int val1 , int val2)</code>	—	Returns the minimum of <i>val1</i> and <i>val2</i> .
<code>int L_Range (int val)</code>	L	The value for L must be an integer within the range of 1-8. Returns 0 if valid, 1 if invalid.
<code>int L_Max (int val, int init)</code>	L	The value for L must not exceed the initially configured value. Returns 0 if valid, 1 if invalid.
<code>int L_Even (int L_val, int F_val)</code>	L, F	The value for L must be an even number if F = 1. Returns 0 if valid, 1 if invalid.
<code>int M_Range (int val)</code>	M	The value for M must be an integer within the range of 1-32. Returns 0 if valid, 1 if invalid.
<code>int F_Range (int val)</code>	F	The value for F must be an integer within the range of 1,2, 4-256 (any integer value between 1-256 except 3) TL: The value for F must be an integer of the values 1, 2, 4, 8. Returns 0 if valid, 1 if invalid.
<code>int F_Max (int val, int init)</code>	F, M ,S, N', L	The value for F must not exceed the initially configured value. By extension, since F is defined by the formula $F = (M * S * N') / (8 * L)$, the values for M, S, N' and L must be such that the new value for F not exceed the initial configured value. Returns 0 if valid, 1 if invalid.
<code>int F_Equal (int M_val, int S_val, int NP_val, int L_val, int F_val)</code>	F, M, S, N', L	The values for M, S, N' and L must be such that the current value of F conforms to the formula $F(\text{current}) = (M * S * N') / (8 * L)$. If a new value of F is indicated, then $F(\text{current}) = F(\text{new})$. If not, then $F(\text{current}) = F(\text{initially configured})$. Returns 0 if valid, 1 if invalid.
<code>int S_Range (int val)</code>	S	The value for S must be an integer within the range of 1-32. Returns 0 if valid, 1 if invalid.

Function Prototype	Parameters	Rule/Function Description
int N_Range (int <i>val</i>)	N	The value for N must be an integer within the range of 1-32. TL: The value for N must be an integer within the range of 12-16. Returns 0 if valid, 1 if invalid.
int N_Max (int <i>val</i> , int <i>max</i>)	N	The value for N must adhere to the range of $N \leq N'$. Returns 0 if valid, 1 if invalid.
int NP_Range (int <i>val</i>)	N'	The value for N' must be an integer within the range of 4-32. Returns 0 if valid, 1 if invalid.
int CS_Range (int <i>val</i>)	CS	The value for CS must be an integer within the range of 0-3. Returns 0 if valid, 1 if invalid.
int K_Range (int <i>K_val</i> , int <i>F_val</i>)	K	The value for K must be an integer within the range $17/F \leq K \leq \min(32, \text{floor}(1024/F))$. Returns 0 if valid, 1 if invalid.
int FxK_Div_4 (int <i>FxK_val</i>)	F, K	The value of $F * K$ must be divisible by 4. Returns 0 if valid, 1 if invalid.
int Calc_FxK (int <i>F_val</i> , int <i>K_val</i>)	—	Calculates $F * K$ value and checks the FxK_Div_4 rule. If $F * K$ is divisible by 4, return $F * K$ value, else return 0. Note: Error return value differs from the usual value.
int HD_Range (int <i>val</i>)	High Density (HD)	The value for HD must be either 0 or 1. Returns 0 if valid, 1 if invalid.
int HD_Transport (int <i>HD_val</i> , int <i>N_val</i>)	HD, N	TL: The value for HD can be 1 if and only if $N=16$. Returns 0 if valid, 1 if invalid.
int SCR_Range (int <i>val</i>)	Scrambler Enable	The value for SCR must be either 0 or 1. Returns 0 if valid, 1 if invalid.
int SUB_Range (int <i>val</i>)	Subclass	The value for subclass must be 0, 1 or 2. Returns 0 if valid, 1 if invalid.
int DR_Range (int <i>val</i>)	Serial Data Rate	The value for DR must be 1, 2, 4. DR value is integer divisor of initially configured serial data rate. For example, $DR=2$ means the initially configured data rate divided by 2. Returns 0 if valid, 1 if invalid.

Function Prototype	Parameters	Rule/Function Description
int DR_Min (int <i>dr_init</i> , int <i>val</i>)	Serial Data Rate	<p>The value for DR must not fall below the minimum allowable range of the target device family. Minimum serial data rate spec:</p> <ul style="list-style-type: none"> Arria V GZ, Arria 10, Stratix V: 2 Gbps Arria V: 1 Gbps <p>Refer to the JESD204B IP Core User Guide for the latest information on the minimum serial data rate spec. The minimum data rate value is defined by the DATA_RATE_MIN parameter in the <code>main.h</code> header file. You are responsible to ensure that the minimum data rate value for the target device is correctly set.</p> <p>Returns 0 if valid, 1 if invalid.</p>
int FC_Range (int <i>val</i>)	Frame clock, serial data rate, M, S, N', L	<p>The values of serial data rate, M, S, N' and L must be such that the new frame clock value, FC(new) does not exceed the initially configured frame clock value, FC(initially configured). Furthermore, the ratio of FC(initially configured) to FC(new) must be 1, 2, 4 only.</p> <p>Returns 0 if valid, 1 if invalid.</p>
int Calc_FC (int <i>M_val</i> , int <i>S_val</i> , int <i>NP_val</i> , int <i>L_val</i> , int <i>DR_div_val</i> , int <i>DR_init_val</i> , int <i>FC_init_val</i>)	—	<p>Calculates the new frame clock value, FC(new) according to the following formula: $FC(new) = (Serial\ data\ rate * L * 8) / (M * S * N' * 10)$. Calculates the ratio of initially configured frame clock value, FC(initially configured) to FC(new) and checks FC_Range rule (i.e 1, 2, 4). If ratio is within range, return ratio, else return 0.</p> <p>Note: Error return value differs from the usual value.</p>

Custom Peripheral Access Macros in macros.c Source File

A set of peripheral access macros are provided to allow you to access specific information in the CSR of the following peripherals:

- Reset sequencer
- JESD204B TX
- JESD204B RX
- PIO control
- PIO status
- Transceiver Native PHY IP core
- ATX PLL
- Core PLL Reconfiguration

The function prototypes of the macros listed in the table below can be found in the `macros.h` header file located in the `software` file directory

Table 15: macros.c

Function Prototype	Description
int CALC_BASE_ADDRESS_LINK (int <i>base</i> , int <i>link</i>)	Calculates and returns the base address based on the <i>link</i> provided. In the QSYS system (<code>jesd204b_ed_qsys.qsys</code>) address map, bits 16-19 are reserved for multi-link addressing. The address map allocation allows for up to a maximum of 16 links to be supported using the existing address map. The number of multi-links in the design is defined by the <code>MAX_LINKS</code> parameter in the <code>main.h</code> header file. You are responsible to set the parameter correctly to reflect the system configuration.
int CALC_BASE_ADDRESS_XCVR_PLL (int <i>base</i> , int <i>instance</i>)	Calculates and returns the base address of the TX transceiver PLL (ATX PLL) based on the <i>instance</i> number. In the JESD204B subsystem (<code>jesd204b_subsystem.qsys</code>) address map, bits 12-13 are reserved for multi ATX PLL addressing. The address map allocation allows for up to a maximum of four ATX PLLs per link to be supported using the existing address map. The number of ATX PLLs per link in the design is defined by the <code>XCVR_PLL_PER_LINK</code> parameter in the <code>main.h</code> header file. You are responsible to set the parameter correctly to reflect the system configuration.
int IORD_RESET_SEQUENCER_STATUS_REG (int <i>link</i>)	Read reset sequencer status register at <i>link</i> and return the value.
int IORD_RESET_SEQUENCER_RESET_ACTIVE (int <i>link</i>)	Read reset sequencer status register at <i>link</i> and return 1 if the reset active signal is asserted, else return 0.
void IOWR_RESET_SEQUENCER_INIT_RESET_SEQ (int <i>link</i>)	Write reset sequencer at <i>link</i> to trigger full hardware reset sequence.
void IOWR_RESET_SEQUENCER_FORCE_RESET (int <i>link</i> , int <i>val</i>)	Write reset sequencer at <i>link</i> to force assert or deassert resets based on the <i>val</i> value.
int IORD_JESD204_TX_STATUS0_REG (int <i>link</i>)	Read the JESD204B TX CSR <code>tx_status0</code> register at <i>link</i> and return the value.
int IORD_JESD204_TX_SYNCN_SYSREF_CTRL_REG (int <i>link</i>)	Read the JESD204B TX CSR <code>syncn_sysref_ctrl</code> register at <i>link</i> and return the value.
void IOWR_JESD204_TX_SYNCN_SYSREF_CTRL_REG (int <i>link</i> , int <i>val</i>)	Write <i>val</i> value into the JESD204B TX CSR <code>syncn_sysref_ctrl</code> register at <i>link</i> .
int IORD_JESD204_RX_STATUS0_REG (int <i>link</i>)	Read JESD204B RX CSR <code>rx_status0</code> register at <i>link</i> and return value.
int IORD_JESD204_RX_SYNCN_SYSREF_CTRL_REG (int <i>link</i>)	Read JESD204B RX CSR <code>syncn_sysref_ctrl</code> register at <i>link</i> and return value.
void IOWR_JESD204_RX_SYNCN_SYSREF_CTRL_REG (int <i>link</i> , int <i>val</i>)	Write <i>val</i> value into the JESD204B RX CSR <code>syncn_sysref_ctrl</code> register at <i>link</i> .
int IORD_JESD204_TX_ILAS_DATA1_REG (int <i>link</i>)	Read the JESD204B TX CSR <code>ilas_data1</code> register at <i>link</i> and return the value.

Function Prototype	Description
int IORD_JESD204_RX_ILAS_DATA1_REG (int <i>link</i>)	Read the JESD204B RX CSR <i>ilas_data1</i> register at <i>link</i> and return the value.
void IOWR_JESD204_TX_ILAS_DATA1_REG (int <i>link</i> , int <i>val</i>)	Write <i>val</i> value into the JESD204B TX CSR <i>ilas_data1</i> register at <i>link</i> .
void IOWR_JESD204_RX_ILAS_DATA1_REG (int <i>link</i> , int <i>val</i>)	Write <i>val</i> value into the JESD204B RX CSR <i>ilas_data1</i> register at <i>link</i> .
int IORD_JESD204_TX_ILAS_DATA2_REG (int <i>link</i>)	Read the JESD204B TX CSR <i>ilas_data2</i> register at <i>link</i> and return the value.
int IORD_JESD204_RX_ILAS_DATA2_REG (int <i>link</i>)	Read the JESD204B RX CSR <i>ilas_data2</i> register at <i>link</i> and return the value.
void IOWR_JESD204_TX_ILAS_DATA2_REG (int <i>link</i> , int <i>val</i>)	Write <i>val</i> value into the JESD204B TX CSR <i>ilas_data2</i> register at <i>link</i> .
void IOWR_JESD204_RX_ILAS_DATA2_REG (int <i>link</i> , int <i>val</i>)	Write <i>val</i> value into the JESD204B RX CSR <i>ilas_data2</i> register at <i>link</i> .
int IORD_JESD204_TX_ILAS_DATA12_REG (int <i>link</i>)	Read the JESD204B TX CSR <i>ilas_data12</i> register at <i>link</i> and return the value.
int IORD_JESD204_RX_ILAS_DATA12_REG (int <i>link</i>)	Read the JESD204B RX CSR <i>ilas_data12</i> register at <i>link</i> and return the value.
void IOWR_JESD204_TX_ILAS_DATA12_REG (int <i>link</i> , int <i>val</i>)	Write <i>val</i> value into the JESD204B TX CSR <i>ilas_data12</i> register at <i>link</i> .
void IOWR_JESD204_RX_ILAS_DATA12_REG (int <i>link</i> , int <i>val</i>)	Write <i>val</i> value into the JESD204B RX CSR <i>ilas_data12</i> register at <i>link</i> .
int IORD_JESD204_TX_GET_L_VAL (int <i>link</i>)	Read the JESD204B TX CSR <i>ilas_data1</i> register at <i>link</i> and return the L value.
int IORD_JESD204_RX_GET_L_VAL (int <i>link</i>)	Read the JESD204B RX CSR <i>ilas_data1</i> register at <i>link</i> and return the L value.
int IORD_JESD204_TX_GET_F_VAL (int <i>link</i>)	Read the JESD204B TX CSR <i>ilas_data1</i> register at <i>link</i> and return the F value.
int IORD_JESD204_RX_GET_F_VAL (int <i>link</i>)	Read the JESD204B RX CSR <i>ilas_data1</i> register at <i>link</i> and return the F value.
int IORD_JESD204_TX_GET_K_VAL (int <i>link</i>)	Read the JESD204B TX CSR <i>ilas_data1</i> register at <i>link</i> and return the K value.
int IORD_JESD204_RX_GET_K_VAL (int <i>link</i>)	Read JESD204B RX CSR <i>ilas_data1</i> register at <i>link</i> and return K value.
int IORD_JESD204_TX_GET_M_VAL (int <i>link</i>)	Read the JESD204B TX CSR <i>ilas_data1</i> register at <i>link</i> and return the M value.
int IORD_JESD204_RX_GET_M_VAL (int <i>link</i>)	Read the JESD204B RX CSR <i>ilas_data1</i> register at <i>link</i> and return the M value.
int IORD_JESD204_TX_GET_N_VAL (int <i>link</i>)	Read the JESD204B TX CSR <i>ilas_data1</i> register at <i>link</i> and return the N value.

Function Prototype	Description
int IORD_JESD204_RX_GET_N_VAL (int <i>link</i>)	Read the JESD204B RX CSR <i>ilas_data1</i> register at <i>link</i> and return the N value.
int IORD_JESD204_TX_GET_NP_VAL (int <i>link</i>)	Read the JESD204B TX CSR <i>ilas_data1</i> register at <i>link</i> and return the NP value.
int IORD_JESD204_RX_GET_NP_VAL (int <i>link</i>)	Read the JESD204B RX CSR <i>ilas_data1</i> register at <i>link</i> and return the NP value.
int IORD_JESD204_TX_GET_S_VAL (int <i>link</i>)	Read the JESD204B TX CSR <i>ilas_data1</i> register at <i>link</i> and return the S value.
int IORD_JESD204_RX_GET_S_VAL (int <i>link</i>)	Read the JESD204B RX CSR <i>ilas_data1</i> register at <i>link</i> and return the S value.
int IORD_JESD204_TX_GET_HD_VAL (int <i>link</i>)	Read the JESD204B TX CSR <i>ilas_data1</i> register at <i>link</i> and return the HD value.
int IORD_JESD204_RX_GET_HD_VAL (int <i>link</i>)	Read the JESD204B RX CSR <i>ilas_data1</i> register at <i>link</i> and return the HD value.
int IORD_JESD204_TX_LANE_CTRL_REG (int <i>link</i> , int <i>offset</i>)	Read the JESD204B TX CSR <i>lane_ctrl_*</i> register at <i>link</i> and return the value.
int IORD_JESD204_RX_LANE_CTRL_REG (int <i>link</i> , int <i>offset</i>)	Read the JESD204B RX CSR <i>lane_ctrl_*</i> register at <i>link</i> and return the value.
void IOWR_JESD204_TX_LANE_CTRL_REG (int <i>link</i> , int <i>offset</i> , int <i>val</i>)	Write <i>val</i> value into the JESD204B TX CSR <i>lane_ctrl_*</i> register at <i>link</i> .
void IOWR_JESD204_RX_LANE_CTRL_REG (int <i>link</i> , int <i>offset</i> , int <i>val</i>)	Write <i>val</i> value into the JESD204B RX CSR <i>lane_ctrl_*</i> register at <i>link</i> .
int IORD_PIO_CONTROL_REG (void)	Read the PIO control register and return the value.
void IOWR_PIO_CONTROL_REG (int <i>val</i>)	Write <i>val</i> value into the PIO control register.
int IORD_PIO_STATUS_REG (void)	Read the PIO status register and return the value.
int IORD_JESD204_TX_TEST_MODE_REG (int <i>link</i>)	Read the JESD204B TX CSR <i>tx_test</i> register at <i>link</i> and return the value.
int IORD_JESD204_RX_TEST_MODE_REG (int <i>link</i>)	Read the JESD204B RX CSR <i>rx_test</i> register at <i>link</i> and return the value.
void IOWR_JESD204_TX_TEST_MODE_REG (int <i>link</i> , int <i>val</i>)	Write <i>val</i> value into the JESD204B TX CSR <i>tx_test</i> register at <i>link</i> .
void IOWR_JESD204_RX_TEST_MODE_REG (int <i>link</i> , int <i>val</i>)	Write <i>val</i> value into the JESD204B RX CSR <i>rx_test</i> register at <i>link</i> .
int IORD_JESD204_RX_ERR0_REG (int <i>link</i>)	Read the JESD204B RX CSR <i>rx_err0</i> register at <i>link</i> and return the value.
void IOWR_JESD204_RX_ERR0_REG (int <i>link</i> , int <i>val</i>)	Write <i>val</i> value into the JESD204B RX CSR <i>rx_err0</i> register at <i>link</i> .
int IORD_JESD204_RX_ERR1_REG (int <i>link</i>)	Read the JESD204B RX CSR <i>rx_err1</i> register at <i>link</i> and return the value.

Function Prototype	Description
void IOWR_JESD204_RX_ERR1_REG (int <i>link</i> , int <i>val</i>)	Write <i>val</i> value into the JESD204B RX CSR <i>rx_err1</i> register at <i>link</i> .
int IORD_JESD204_TX_ERR_REG (int <i>link</i>)	Read the JESD204B TX CSR <i>tx_err</i> register at <i>link</i> and return the value.
void IOWR_JESD204_TX_ERR_REG (int <i>link</i> , int <i>val</i>)	Write <i>val</i> value into the JESD204B TX CSR <i>tx_err</i> register at <i>link</i> .
int IORD_XCVR_NATIVE_A10_REG (int <i>link</i> , int <i>offset</i>)	Read the transceiver reconfiguration register at <i>link</i> and address offset at <i>offset</i> and return the value.
void IOWR_XCVR_NATIVE_A10_REG (int <i>link</i> , int <i>offset</i> , int <i>val</i>)	Write <i>val</i> value into the transceiver reconfiguration register at <i>link</i> and address offset at <i>offset</i> .
int IORD_XCVR_ATX_PLL_A10_REG (int <i>link</i> , int <i>instance</i> , int <i>offset</i>)	Read the ATX PLL reconfiguration register indicated by the instance number <i>instance</i> at <i>link</i> and address offset at <i>offset</i> and return the value.
void IOWR_XCVR_ATX_PLL_A10_REG (int <i>link</i> , int <i>instance</i> , int <i>offset</i> , int <i>val</i>)	Write <i>val</i> value into the ATX PLL reconfiguration register indicated by instance number <i>instance</i> at <i>link</i> and address offset at <i>offset</i> .
int IORD_CORE_PLL_RECONFIG_C0_COUNTER_REG (void)	Read the core PLL reconfiguration c0 counter register and return the value.
int IORD_CORE_PLL_RECONFIG_C1_COUNTER_REG (void)	Read the core PLL reconfiguration c1 counter register and return the value.
void IOWR_CORE_PLL_RECONFIG_C0_COUNTER_REG (int <i>val</i>)	Write <i>val</i> value into the core PLL reconfiguration c0 counter register.
void IOWR_CORE_PLL_RECONFIG_C1_COUNTER_REG (int <i>val</i>)	Write <i>val</i> value into the core PLL reconfiguration c1 counter register.

AN 729 Document Revision History

Table 16: Document Revision History

Date	Version	Changes
May 2015	2015.05.04	Updated the software requirement to Quartus II software version 15.0.
February 2015	2015.02.13	Initial release.