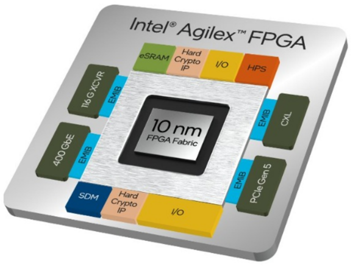
Manuals+ — User Manuals Simplified.

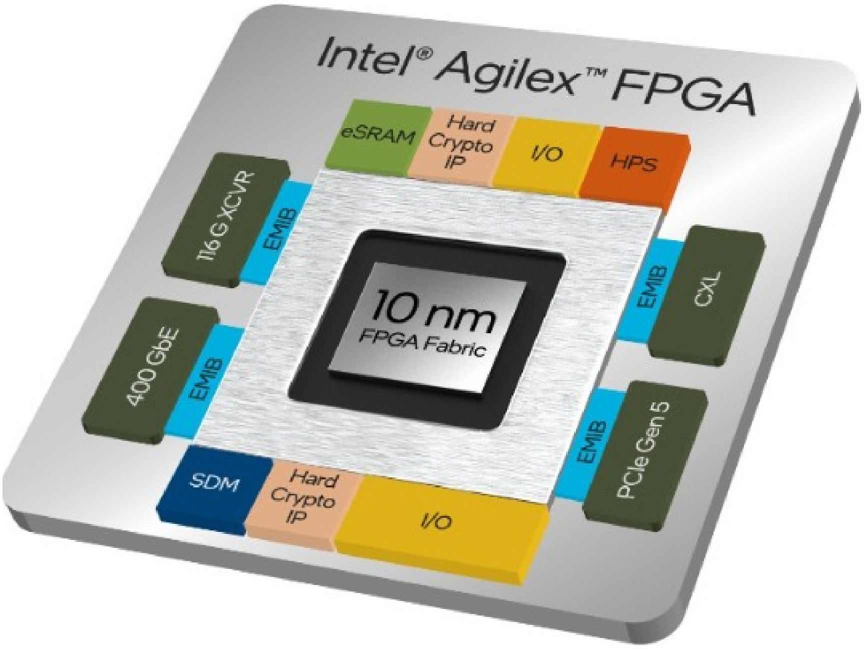


intel Fronthaul Compression FPGA IP User Guide

Home » Intel » intel Fronthaul Compression FPGA IP User Guide

intel®

**Fronthaul Compression FPGA IP
User Guide**



Contents

1 Fronthaul Compression FPGA IP
2 About the Fronthaul Compression Intel® FPGA IP
3 Getting Started with the Fronthaul Compression Intel FPGA IP
4 Fronthaul Compression IP Functional Description
5 Fronthaul Compression IP Registers
6 Fronthaul Compression Intel FPGA IPs User Guide Archive
7 Document Revision History for the Fronthaul Compression Intel FPGA IP User Guide
8 Documents / Resources
9 Related Posts

Fronthaul Compression FPGA IP

Fronthaul Compression Intel® FPGA IP User Guide
Updated for Intel® Quartus® Prime
Design Suite: 21.4 IP
Version: 1.0.1

About the Fronthaul Compression Intel® FPGA IP

The Fronthaul Compression IP consists of compression and decompression for U-plane IQ data. The compression engine computes µ-law or block floating-point compression based on user data compression header (udCompHdr). This IP uses an Avalon streaming interface for IQ data, conduit signals, and for metadata and sideband signals, and Avalon memory-mapped interface for control and status registers (CSRs).
The IP maps compressed IQs and the user data compression parameter (udCompParam) as per the section payload frame format specified in the O-RAN specification O-RAN Fronthaul Control, User and Synchronization Plane Version 3.0 April 2020 (O-RAN-WG4.CUS.0-v03.00). Avalon streaming sink and source interface data width are 128-bits for the application interface and 64 bits for the transport interface to support maximum compressoin ratio of 2:1.

Related Information

O-RAN website

1.1. Fronthaul Compression Intel® FPGA IP Features

- -law and block floating-point compression and decompression
- IQ width 8-bit to 16-bit
- Static and dynamic configuration of U-plane IQ format and compression header
- Multisections packet (if O-RAN Compliant is on)

1.2. Fronthaul Compression Intel® FPGA IP Device Family Support

Intel offers the following device support levels for Intel FPGA IP:

- Advance support–the IP is available for simulation and compilation for this device family. FPGA programming file (.pof) support is not available for Quartus Prime Pro Stratix 10 Edition Beta software and as such IP timing closure cannot be guaranteed. Timing models include initial engineering estimates of delays based on early post-layout information. The timing models are subject to change as silicon testing improves the correlation between the actual silicon and the timing models. You can use this IP core for system architecture and resource utilization studies, simulation, pinout, system latency assessments, basic timing assessments (pipeline budgeting), and I/O transfer strategy (data-path width, burst depth, I/O standards tradeoffs).
- Preliminary support–Intel verifies the IP core with preliminary timing models for this device family. The IP core

meets all functional requirements, but might still be undergoing timing analysis for the device family. You can use it in production designs with caution.

- Final support–Intel verifies the IP with final timing models for this device family. The IP meets all functional and timing requirements for the device family. You can use it in production designs.

Table 1. Fronthaul Compression IP Device Family Support

Device Family	Support
Intel® Agilex™ (E-tile)	Preliminary
Intel Agilex (F-tile)	Advance
Intel Arria® 10	Final
Intel Stratix® 10 (H-, and E-tile devices only)	Final
Other device families	No support

Table 2. Device Supported Speed Grades

Device Family	FPGA Fabric Speed Grade
Intel Agilex	3
Intel Arria 10	2
Intel Stratix 10	2

1.3. Release Information for the Fronthaul Compression Intel FPGA IP

Intel FPGA IP versions match the Intel Quartus® Prime Design Suite software versions until v19.1. Starting in Intel Quartus Prime Design Suite software version 19.2, Intel FPGA IP has a new versioning scheme.

The Intel FPGA IP version (X.Y.Z) number can change with each Intel Quartus Prime software version. A change in:

- X indicates a major revision of the IP. If you update the Intel Quartus Prime software, you must regenerate the IP.
- Y indicates the IP includes new features. Regenerate your IP to include these new features.
- Z indicates the IP includes minor changes. Regenerate your IP to include these changes.

Table 3. Fronthaul Compression IP Release Information

Item	Description
Version	1.0.1
Release date	February 2022
Ordering code	IP-FH-COMP

1.4. Fronthaul Compression Performance and Resource Usage

The resources of the IP targeting an Intel Agilex device, Intel Arria 10 device, and Intel Stratix 10 device

Table 4. Fronthaul Compression Performance and Resource Usage
All entries are for compression and decompression data direction IP

Device	IP	ALMs	Logic registers		M20K
			Primary	Secondary	
Intel Agilex	Block-floating point	14,969	25,689	6,093	0
	µ-law	22,704	39,078	7,896	0
	Block-floating point and µ-law	23,739	41,447	8,722	0
	Block-floating point, µ-law, and extended IQ width	23,928	41,438	8,633	0
Intel Arria 10	Block-floating point	12,403	16,156	5,228	0
	µ-law	18,606	23,617	5,886	0
	Block-floating point and µ-law	19,538	24,650	6,140	0
	Block-floating point, µ-law, and extended IQ width	19,675	24,668	6,141	0
Intel Stratix 10	Block-floating point	16,852	30,548	7,265	0
	µ-law	24,528	44,325	8,080	0
	Block-floating point and µ-law	25,690	47,357	8,858	0
	Block-floating point, µ-law, and extended IQ width	25,897	47,289	8,559	0

Getting Started with the Fronthaul Compression Intel FPGA IP

Describes installing, parameterizing, simulating, and initializing the Fronthaul Compression IP.

2.1. Obtaining, Installing, and Licensing the Fronthaul Compression IP

The Fronthaul Compression IP is an extended Intel FPGA IP that is not included with the Intel Quartus Prime release.

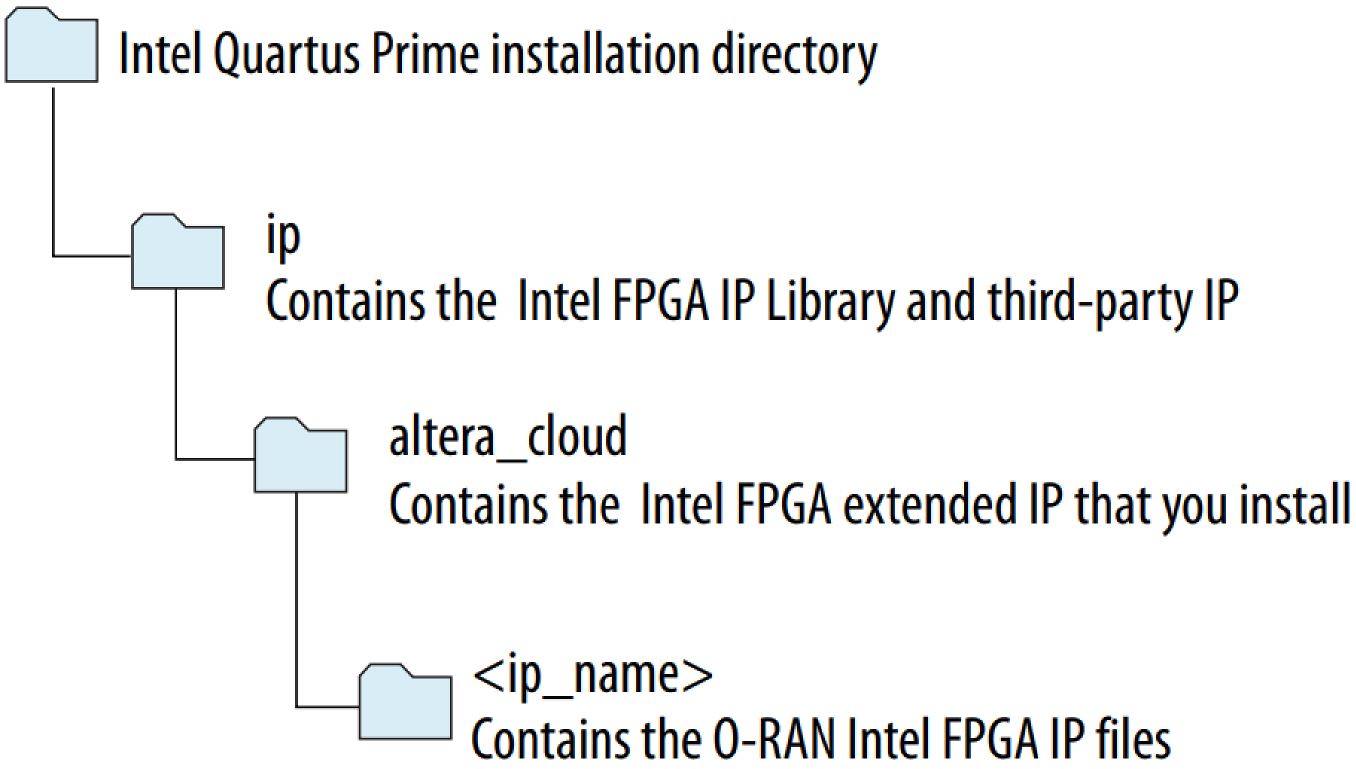
1. Create a My Intel account if you do not have one.
2. Log in to access the Self-Service Licensing Center (SSLC).

3. Purchase the Fronthaul Compression IP.
4. On the SSLC page, click Run for the IP. The SSLC provides an installation dialog box to guide your installation of the IP.
5. Install to the same location as Intel Quartus Prime folder.

Table 5. Fronthaul Compression Installation Locations

Location	Software	Platform
<drive>:\intelFPGA_pro\<version>\quartus\ip \altera_cloud	Intel Quartus Prime Pro Editi on	Windows*
<home directory>:/intelFPGA_pro/*<version>*/ quartus/ip/alt era_cloud	Intel Quartus Prime Pro Editi on	Linux*

Figure 1. Fronthaul Compression IP Installation Directory Structure Intel Quartus Prime installation directory


Intel Quartus Prime installation directory
ip
Contains the Intel FPGA IP Library and third-party IP
altera_cloud
Contains the Intel FPGA extended IP that you install
<ip_name>
Contains the O-RAN Intel FPGA IP files

The Fronthaul Compression Intel FPGA IP now appears in the IP Catalog.

Related Information

- Intel FPGA website
- Self-Service Licensing Center (SSLC)

2.2. Parameterizing the Fronthaul Compression IP

Quickly configure your custom IP variation in the IP Parameter Editor.

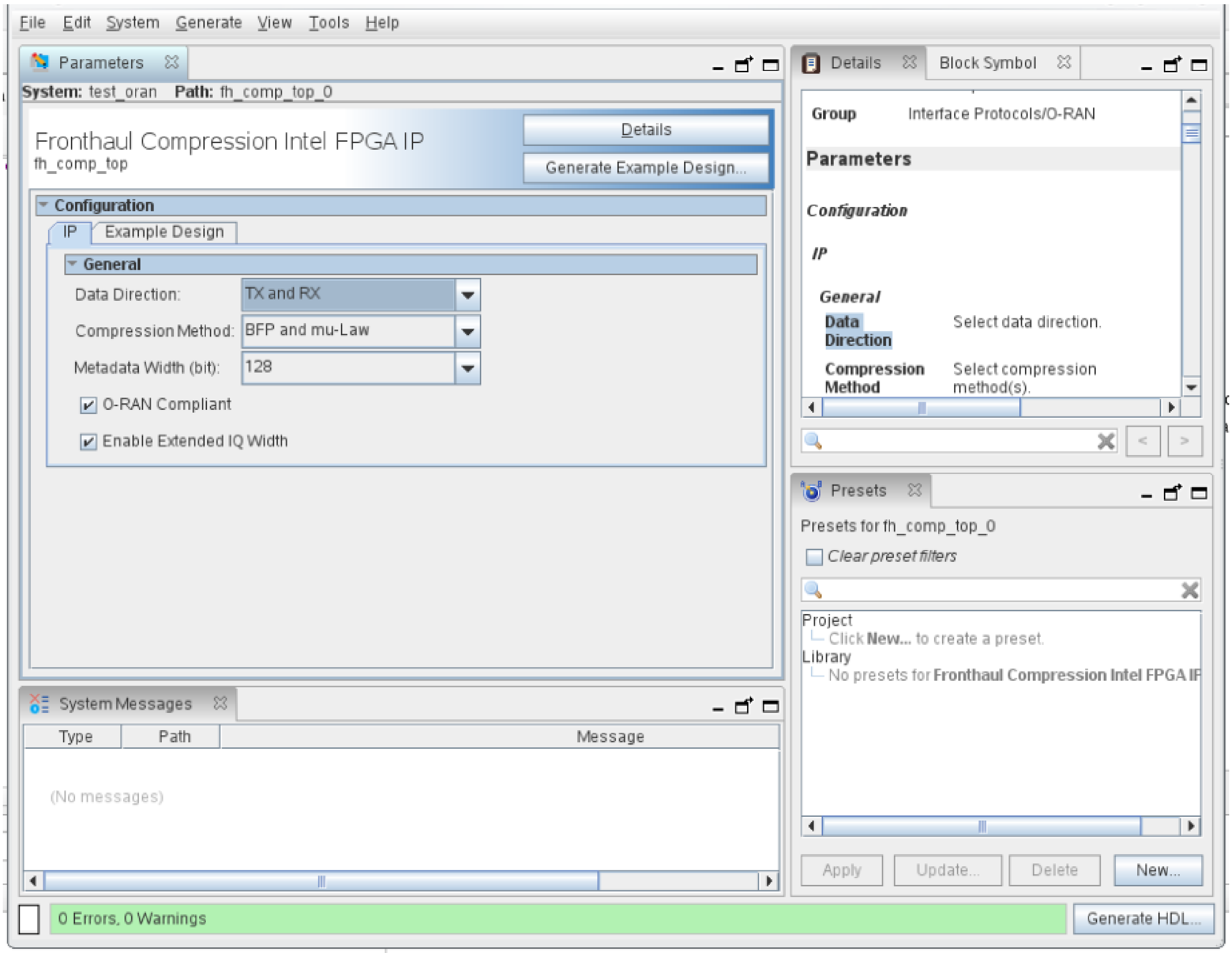
1. Create an Intel Quartus Prime Pro Edition project in which to integrate your IP core.
   a. In the Intel Quartus Prime Pro Edition, click File New Project Wizard to create a new Intel Quartus Prime project, or File Open Project to open an existing Quartus Prime project. The wizard prompts you to specify a

device.

b. Specify the device family that meets the speed grade requirements for the IP.

c. Click Finish.

2. In the IP Catalog, select Fronthaul Compression Intel FPGA IP. The New IP Variation window appears.
3. Specify a top-level name for your new custom IP variation. The parameter editor saves the IP variation settings in a file named <your_ip>.ip.
4. Click OK. The parameter editor appears.



System: test_oran Path: fh_comp_top_0

Fronthaul Compression Intel FPGA IP
fh_comp_top

Data Direction: TX and RX

Compression Method: BFP and mu-Law

Metadata Width (bit): 128

☑ O-RAN Compliant

☑ Enable Extended IQ Width

Group Interface Protocols/O-RAN

Parameters

Configuration

IP

General

Data Direction Select data direction.

Compression Method Select compression method(s).

Presets for fh_comp_top_0

□ Clear preset filters

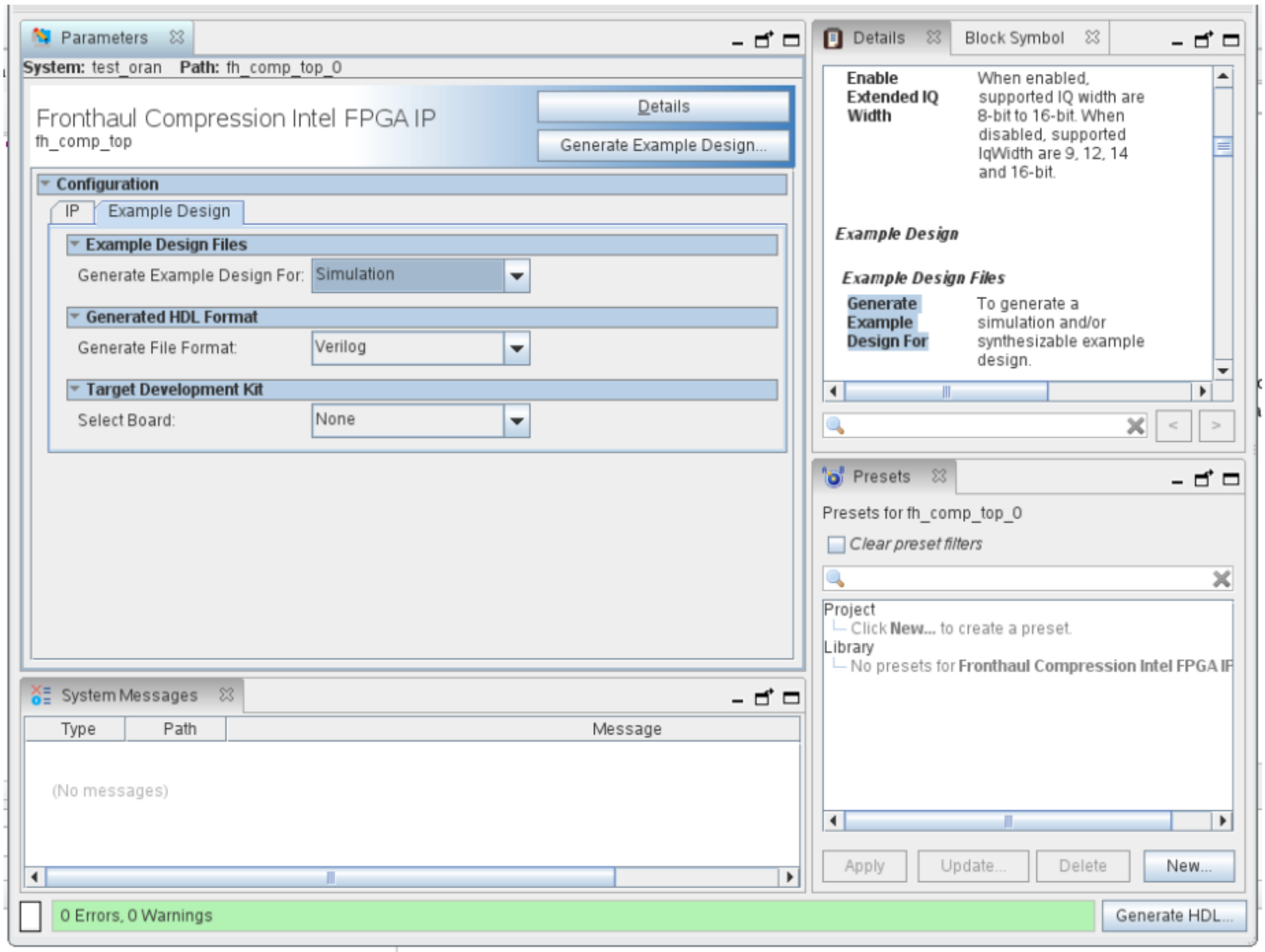
Project
Click New... to create a preset.
Library
No presets for Fronthaul Compression Intel FPGA IP

Type	Path	Message
(No messages)		

0 Errors, 0 Warnings

Figure 2. Fronthaul Compression IP Parameter Editor

5. Specify the parameters for your IP variation. Refer to Parameters for information about specific IP parameters.
6. Click the Design Example tab and specify the parameters for your design example.



System: test_oran Path: fh_comp_top_0

Fronthaul Compression Intel FPGA IP
fh_comp_top

Configuration

Example Design Files

Generate Example Design For: Simulation

Generated HDL Format

Generate File Format: Verilog

Target Development Kit

Select Board: None

Enable Extended IQ Width	When enabled, supported IQ width are 8-bit to 16-bit. When disabled, supported IqWidth are 9, 12, 14 and 16-bit.

Example Design

Example Design Files

Generate Example Design For	To generate a simulation and/or synthesizable example design.

Presets for fh_comp_top_0

Project
Click **New...** to create a preset.
Library
No presets for **Fronthaul Compression Intel FPGA IP**

(No messages)

0 Errors, 0 Warnings

Figure 3. Design Example Parameter Editor

7. Click Generate HDL. The Generation dialog box appears.
8. Specify output file generation options, and then click Generate. The IP variation files generate according to your specifications.
9. Click Finish. The parameter editor adds the top-level .ip file to the current project automatically. If you are prompted to manually add the .ip file to the project, click Project Add/Remove Files in Project to add the file.
10. After generating and instantiating your IP variation, make appropriate pin assignments to connect ports and set any appropriate per-instance RTL parameters.

2.2.1. Fronthaul Compression IP Parameters

Table 6. Fronthaul Compression IP Parameters

Name	Valid Values	Description
Data direction	TX and RX, TX only, RX only	Select TX for compression; RX for decompression.
Compression method	BFP, mu-Law, or BFP and mu-Law	Select block floating-point, µ-law, or both.

Metadata width	0 (Disable Metadata Ports), 32, 64, 96, 128 (bit)	Specify the bit width of the metadata bus (uncompressed data).
Enable extended IQ width	On or off	Turn on for supported IqWidth of 8-bit to 16-bit. Turn off for supported IqWidth of 9, 12, 14 and 16-bits.
O-RAN compliant	On or off	Turn on to follow ORAN IP mapping for metadata port and assert metadata valid signal for each section header. The IP supports 128-bit width metadata only. The IP supports single section and multiple sections per packet. Metadata is valid at each section with metadata valid assertion. Turn off so the IP uses metadata as passthrough conduit signals with no mapping requirement (e.g.: U-plane numPrb is assumed 0). The IP supports metadata widths of 0 (Disable Metadata Ports), 32, 64, 96, 128 bits. The IP supports single section per packet. Metadata is valid only once at the metadata valid assertion for each packet.

2.3. Generated IP File Structure

The Intel Quartus Prime Pro Edition software generates the following IP core output file structure.

Table 7. Generated IP Files

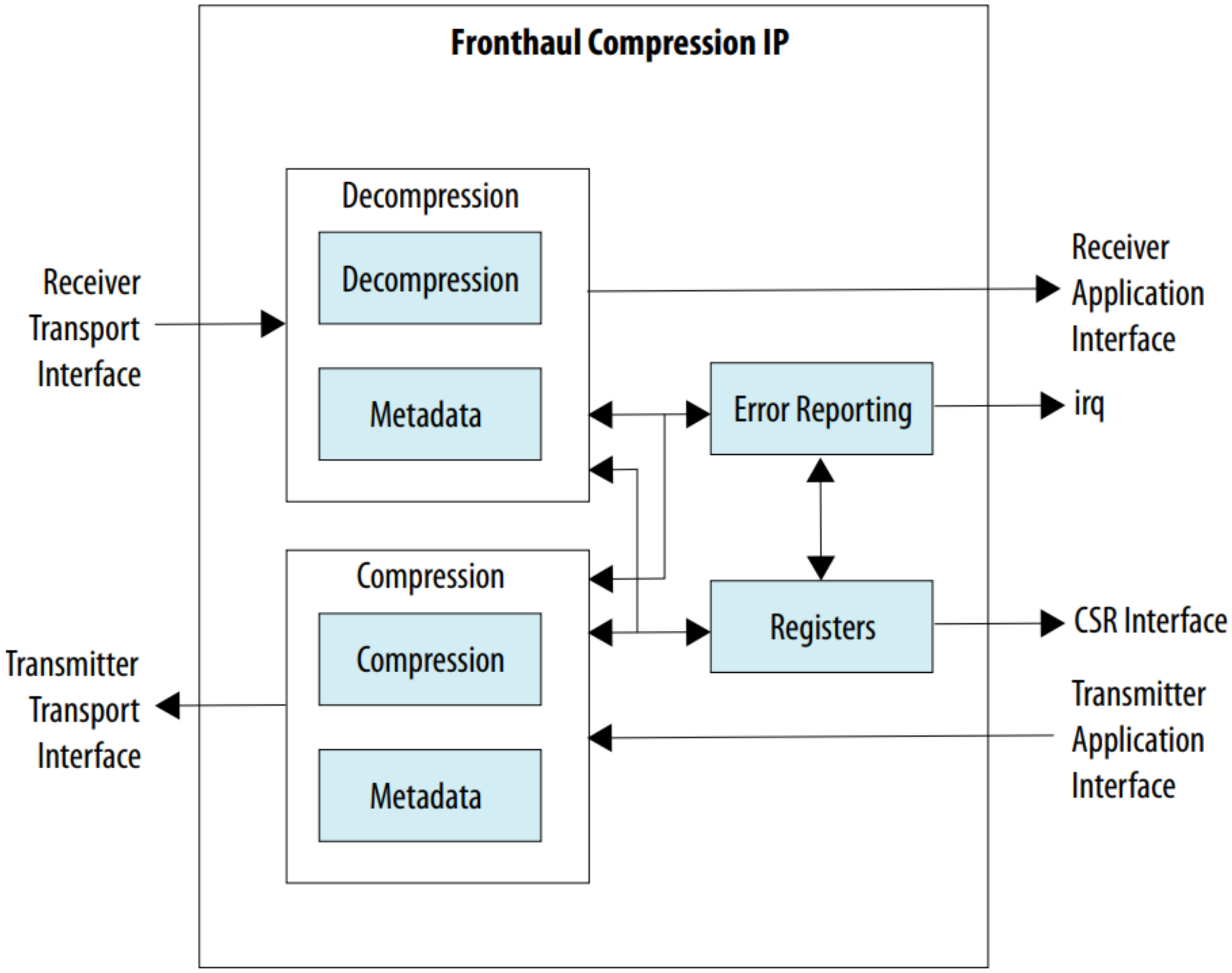
File Name	Description
<*your_ip*>.ip	The Platform Designer system or top-level IP variation file. <*your_ip*> is the name that you give your IP variation.
<*your_ip*>.cmp	The VHDL Component Declaration (.cmp) file is a text file that contains local generic and port definitions that you can use in VHDL design files.
<*your_ip*>.html	A report that contains connection information, a memory map showing the address of each slave with respect to each master to which it is connected, and parameter assignments.
<*your_ip*>_generation.rpt	IP or Platform Designer generation log file. A summary of the messages during IP generation.
<*your_ip*>.qgsimc	Lists simulation parameters to support incremental regeneration.

<*your_ip*>.qgsynthc	Lists synthesis parameters to support incremental regeneration.
<*your_ip*>.qip	Contains all the required information about the IP component to integrate and compile the IP component in the Intel Quartus Prime software.
<*your_ip*>.sopcinfo	Describes the connections and IP component parameterizations in your Platform Designer system. You can parse its contents to get requirements when you develop software drivers for IP components. Downstream tools such as the Nios® II tool chain use this file. The .sopcinfo file and the system.h file generated for the Nios II tool chain include address map information for each slave relative to each master that accesses the slave. Different masters may have a different address map to access a particular slave component.
<*your_ip*>.csv	Contains information about the upgrade status of the IP component.
<*your_ip*>.bsf	A Block Symbol File (.bsf) representation of the IP variation for use in Intel Quartus Prime Block Diagram Files (.bdf).
<*your_ip*>.spd	Required input file for ip-make-simscript to generate simulation scripts for supported simulators. The .spd file contains a list of files generated for simulation, along with information about memories that you can initialize.

<*your_ip*>.ppf	The Pin Planner File (.ppf) stores the port and node assignments for IP components created for use with the Pin Planner.
<*your_ip*>_bb.v	You can use the Verilog black-box (_bb.v) file as an empty module declaration for use as a black box.
<*your_ip*>_inst.v or _inst.vhd	HDL example instantiation template. You can copy and paste the contents of this file into your HDL file to instantiate the IP variation.
<*your_ip*>.v or <*your_ip*>.vhd	HDL files that instantiate each submodule or child IP core for synthesis or simulation.
mentor/	Contains a ModelSim* script msim_setup.tcl to set up and run a simulation.
synopsys/vcs/ synopsys/vcsmx/	Contains a shell script vcs_setup.sh to set up and run a VCS* simulation . Contains a shell script vcsmx_setup.sh and synopsys_ sim.setup file to set up and run a VCS MX* simulation.
cadence/	Contains a shell script ncsim_setup.sh and other setup files to set up and run an NCSIM* simulation.
aldec/	Contains a shell script rivierapro_setup.sh to setup and run an Aldec* simulation.
xcelium/	Contains a shell script xcelium_setup.sh and other setup files to set up and run an Xcelium* simulation.
submodules/	Contains HDL files for the IP core submodules.
<child IP cores>/	For each generated child IP core directory, Platform Designer generates synth/ and sim/ sub-directories.

Fronthaul Compression IP Functional Description

Figure 4. The Fronthaul Compression IP comprises compression and decompression. Fronthaul Compression IP Block Diagram



Compression and Decompression

A preprocessing block-based bit shift block generates the optimum bit-shifts for a resource block of 12 resource elements (REs). The block reduces the quantization noise, especially for low-amplitude samples. Hence, it reduces the error vector magnitude (EVM) that compression introduces. The compression algorithm is almost independent of the power value. Assuming the complex input samples is x = x1 + jxQ, the maximum absolute value of the real and imaginary components for the resource block is:

$$maxI_n = max\left\{ \left|xI_{12(n-1)+1}\right| , \left|xI_{12(n-1)+2}\right|, \ldots, \left|xI_{12n}\right| \right\}$$

$$maxQ_n = max\left\{ \left|xQ_{12(n-1)+1}\right| , \left|xQ_{12(n-1)+2}\right|, \ldots, \left|xQ_{12n}\right| \right\}$$

The maximum value of the resource block n is:

$$maxVal_n = max\{ maxI_n , maxQ_n \}$$

Having the maximum absolute value for the resource block, the following equation determines the left shift value assigned to that resource block:

$$lshift_n = \begin{cases} bitWidth - \lceil \log_2(maxVal_n) \rceil - 1 \\ 0 \end{cases}$$

$$if \quad maxVal_n < 2^{bitWidth - 1}$$

$$else$$

Where bitWidth is the input bit width.

The IP supports compression ratios of 8, 9, 10, 11, 12, 13, 14, 15, 16.

Mu-Law Compression and Decompression

The algorithm uses Mu-law companding technique, which speech compression widely uses. This technique passes the input uncompressed signal, x, through a compressor with function, f(x), before rounding and bit-truncation. The technique sends compressed data, y, over the interface. The received data passes through an expanding function (which is the inverse of the compressor, F-1(y). The technique reproduces the uncompressed data with minimal quantization error.

Equation 1. Compressor and decompressor functions

$$F(x) = sgn(x)\frac{ln(1+\mu|x|)}{ln(1+\mu)} \qquad -1 \le x \le 1$$

$$F^{-1}(y) = sgn(y)\frac{(1+\mu)^{|y|} - 1}{\mu} \qquad -1 \le y \le 1$$

The Mu-law IQ compression algorithm follows the O-RAN specification.

Related Information

O-RAN website

3.1. Fronthaul Compression IP Signals

Connect and control the IP.

Clock and Reset Interface Signals=

Table 8. Clock and Reset Interface Signals

Signal Name	Bitwidth	Direction	Description
tx_clk	1	Input	Transmitter clock. Clock frequency is 390.625 MHz for 25 Gbps and 156.25MHz for 10 Gbps . All transmitter interface signals are synchronous to this clock.
rx_clk	1	Input	Receiver clock. Clock frequency is 390.625 MHz for 25 Gbps and 156.25MHz for 10 Gbps . All receiver interface signals are synchronous to this clock.
csr_clk	1	Input	Clock for CSR interface. Clock frequency is 100 MHz.

tx_rst_n	1	Input	Active low reset for transmitter interface synchronous to tx_clk.
rx_rst_n	1	Input	Active low reset for receiver interface synchronous to rx_clk.
csr_rst_n	1	Input	Active low reset for CSR interface synchronous to csr_clk.

Transmit Transport Interface Signals

Table 9. Transmit Transport Interface Signals

All signal types are unsigned integer.

Signal Name	Bitwidth	Direction	Description
tx_avst_source_valid	1	Output	When asserted, indicates valid data is available on avst_source_data.
tx_avst_source_data	64	Output	PRB fields including udCompParam, iSample and qSample. Next section PRB fields are concatenated to previous section PRB field.
tx_avst_source_startofpacket	1	Output	Indicates first byte of a frame.
tx_avst_source_endofpacket	1	Output	Indicates last byte of a frame.
tx_avst_source_ready	1	Input	When asserted, indicates the transport layer is ready to accept data. readyLatency = 0 for this interface.
tx_avst_source_empty	3	Output	Specifies the number of empty bytes on avst_source_data when avst_source_endofpacket is asserted.

tx_udcomphdr_o	8	Output	User data compression header field. Synchronous with tx_avst_source_valid. Defines the compression method and IQ bit width for the user data in a data section. • [7:4] : udIqWidth • 16 for udIqWidth=0, otherwise equals udIqWidth e,g,: — 0000b means I and Q are each 16 bits wide; — 0001b means I and Q are each 1 bit wide; — 1111b means I and Q are each 15 bits wide • [3:0] : udCompMeth — 0000b – no compression — 0001b – block-floating point — 0011b – µ-law — others – reserved for future methods.
tx_metadata_o	METADATA_WIDTH	Output	Conduit signals passthrough and are not compressed. Synchronous with tx_avst_source_valid. Configurable bitwidth METADATA_WIDTH. When you turn on **O-RAN compliant**, refer to Table 13 on page 17 .When you turn off **O-RAN compliant**, this signal is only valid when tx_avst_source_startofpacket is 1. tx_metadata_o does not have valid signal and uses tx_avst_source_valid to indicate valid cycle. Not available when you select **0 Disable Metadata Ports** for **Metadata width**.

Receive Transport Interface Signals

Table 10. Receive Transport Interface Signals

No backpressure at this interface. Avalon streaming empty signal is not necessary in this interface because it is always zero.

Signal Name	Bitwidth	Direction	Description
rx_avst_sink_valid	1	Input	When asserted, indicates valid data is available on avst_sink_data. No avst_sink_ready signal at this interface.

rx_avst_sink_data	64	Input	PRB fields including udCompParam, iSample and qSample. Next section PRB fields are concatenated to previous section PRB field.
rx_avst_sink_startofpacket	1	Input	Indicates first byte of a frame.
rx_avst_sink_endofpacket	1	Input	Indicates last byte of a frame.
rx_avst_sink_error	1	Input	When asserted in the same cycle as avst_sink_endofpacket, indicates th e current packet is an error packet
rx_udcomphdr_i	8	Input	User data compression header field. Synchronous with rx_metadata_vali d_i. Defines the compression method and IQ bit width for the user data in a d ata section. • [7:4] : udIqWidth • 16 for udIqWidth=0, otherwise equals udIqWidth. e.g. — 0000b means I and Q are each 16 bits wide; — 0001b means I and Q are each 1 bit wide; — 1111b means I and Q are each 15 bits wide • [3:0] : udCompMeth — 0000b – no compression — 0001b – block floating point — 0011b – µ-law — others – reserved for future methods.
rx_metadata_i	METADATA_WIDTH	Input	Uncompressed conduit signals passthrough. rx_metadata_i signals are valid when rx_metadata_valid_i is asserted, sy nchronous with rx_avst_sink_valid. Configurable bitwidth METADATA_WIDTH. When you turn on **O-RAN compliant**, refer to Table 15 on page 18. When you turn off **O-RAN compliant**, this rx_metadata_i signal is only v alid when both rx_metadata_valid_i and rx_avst_sink_startofpacket equal to 1. Not available when you select **0 Disable Metadata Ports** for **Metadata width**.

rx_metadata_valid_i	1	Input	Indicates that the headers (rx_udcomphdr_i and rx_metadata_i) are valid. Synchronous with rx_avst_sink_valid. Compulsory signal. For O-RAN backward compatibility, assert rx_metadata_valid_i if the IP has valid common header IEs and repeated section IEs. On providing new section physical resource block (PRB) fields in rx_avst_sink_data, provide new section IEs in rx_metadata_i input together with rx_metadata_valid_i.

Transmit Application Interface Signals
Table 11. Transmit Application Interface Signals

Signal Name	Bitwidth	Direction	Description
tx_avst_sink_valid	1	Input	When asserted, indicates valid PRB fields are available in this interface. When operating in streaming mode, ensure no valid signal deassertion between start ofpacket and end of packet The only exception is when the ready signal deasserted.
tx_avst_sink_data	128	Input	Data from application layer in network byte order.
tx_avst_sink_startofpacket	1	Input	Indicate the first PRB byte of a packet
tx_avst_sink_endofpacket	1	Input	Indicate the last PRB byte of a packet
tx_avst_sink_ready	1	Output	When asserted, indicates the O-RAN IP is ready to accept data from application interface. readyLatency = 0 for this interface

tx_udcomphdr_i	*8*	Input	User data compression header field. Synchronous with tx_avst_sink_valid. Defines the compression method and IQ bit width for the user data in a data section. • [7:4] : udIqWidth • 16 for udIqWidth=0, otherwise equals udIqWidth. e.g. — 0000b means I and Q are each 16 bits wide; — 0001b means I and Q are each 1 bit wide; — 1111b means I and Q are each 15 bits wide • [3:0] : udCompMeth — 0000b – no compression — 0001b – block-floating point — 0011b – µ-law — others – reserved for future methods.
tx_metadata_i	METADATA_WIDTH	Input	Conduit signals passthrough and are not compressed. Synchronous with tx_avst_sink_valid. Configurable bitwidth METADATA_WIDTH. When you turn on **O-RAN compliant**, refer to Table 13 on page 17. When you turn off **O-RAN compliant**, this signal only valid when tx_avst_sink_startofpacket equals to 1. tx_metadata_i does not have valid signal and uses tx_avst_sink_valid to indicate valid cycle. Not available when you select **0 Disable Metadata Ports** for **Metadata width**.

Receive Application Interface Signals

Table 12. Receive Application Interface Signals

Signal Name	Bitwidth	Direction	Description
rx_avst_source_valid	1	Output	When asserted, indicates valid PRB fields are available in this interface. No avst_source_ready signal at this interface.
rx_avst_source_data	128	Output	Data to application layer in network byte order.
rx_avst_source_startofpacket	1	Output	Indicates the first PRB byte of a packet

rx_avst_source_endofpacket	1	Output	Indicates the last PRB byte of a packet
rx_avst_source_error	1	Output	Indicates the packets contains error
rx_udcomphdr_o	8	Output	User data compression header field. Synchronous with rx_avst_source_valid. Defines the compression method and IQ bit width for the user data in a data section. • [7:4] : udIqWidth • 16 for udIqWidth=0, otherwise equals udIqWidth. e.g. — 0000b means I and Q are each 16 bits wide; — 0001b means I and Q are each 1 bit wide; — 1111b means I and Q are each 15 bits wide • [3:0] : udCompMeth — 0000b – no compression — 0001b – block floating point (BFP) — 0011b – µ-law — others – reserved for future methods.
rx_metadata_o	METADATA_WIDTH	Output	Uncompressed conduit signals passthrough. rx_metadata_o signals are valid when rx_metadata_valid_o is asserted, synchronous with rx_avst_source_valid. Configurable bitwidth METADATA_WIDTH. When you turn on **O-RAN compliant**, refer to Table 14 on page 18. When you turn off **O-RAN compliant**, rx_metadata_o is only valid when rx_metadata_valid_o equals 1. Not available when you select **0 Disable Metadata Ports** for **Metadata width**.
rx_metadata_valid_o	1	Output	Indicates that the headers (rx_udcomphdr_o and rx_metadata_o) are valid. rx_metadata_valid_o is asserted when rx_metadata_o is valid, synchronous with rx_avst_source_valid.

Metadata Mapping for O-RAN Backward Compatibility

Table 13. tx_metadata_i 128-bit input

Signal Name	Bitwidth	Direction	Description	Metadata Mapping
Reserved	16	Input	Reserved.	tx_metadata_i[127:112]
tx_u_size	16	Input	U-plane packet size in bytes for streaming mode.	tx_metadata_i[111:96]
tx_u_seq_id	16	Input	SeqID of the packet, which is extracted from eCPRI transport header.	tx_metadata_i[95:80]
tx_u_pc_id	16	Input	PCID for eCPRI transport and RoEflowId for radio over ethernet (RoE) transport.	tx_metadata_i[79:64]
Reserved	4	Input	Reserved.	tx_metadata_i[63:60]
tx_u_dataDirection	1	Input	gNB data direction. Value range: {0b=Rx (i.e. upload), 1b= Tx (i.e. download)}	tx_metadata_i[59]
tx_u_filterIndex	4	Input	Defines an index to the channel filter to be used between IQ data and air interface. Value range: {0000b-1111b}	tx_metadata_i[58:55]
tx_u_frameId	8	Input	A counter for 10 ms frames (wrapping period 2.56 seconds), specifically frameId= frame number modulo 256. Value range: {0000 0000b-1111 1111b}	tx_metadata_i[54:47]
tx_u_subframeId	4	Input	A counter for 1 ms subframes within 10 ms frame. Value range: {0000b-1111b}	tx_metadata_i[46:43]

tx_u_slotID	6	Input	This parameter is the slot number within a 1 ms subframe. All slots in one subframe are counted by this parameter. Value range: {00 0000b-00 1111b=slotID, 01 0000b-11 1111b=Reserved}	tx_metadata_i[42:37]
tx_u_symbolid	6	Input	Identifies a symbol number within a slot. Value range: {00 0000b-11 1111b}	tx_metadata_i[36:31]
tx_u_sectionId	12	Input	The sectionID maps U-plane data sections to the corresponding C-plane message (and Section Type) associated with the data. Value range: {0000 0000 0000b-11111 111 1111b}	tx_metadata_i[30:19]
tx_u_rb	1	Input	Resource block indicator. Indicate if every resource block is used or every other resource block is used. Value range: {0b=every resource block used; 1b=every other resource block used}	tx_metadata_i[18]
tx_u_startPrb	10	Input	The starting PRB of a user plane data section. Value range: {00 0000 0000b-11 1111 1111b}	tx_metadata_i[17:8]
tx_u_numPrb	8	Input	Define the PRBs where the user plane data section is valid. Value range: {0000 0001b-1111 1111b, 0000 0000b = all PRBs in the specified subcarrier spacing (SCS) and carrier bandwidth }	tx_metadata_i[7:0]
tx_u_udCompHdr	8	Input	Define the compression method and IQ bit width of the user data in a data section. Value range: {0000 0000b-1111 1111b}	N/A (tx_udcomphdr_i)

Table 14. rx_metadata_valid_i/o

Signal Name	Bitwidth	Direction	Description	Metadata Mapping
rx_sec_hdr_valid	1	Output	When rx_sec_hdr_valid is 1, the U-plane section data fields are valid. Common header IEs are valid when rx_sec_hdr_valid is asserted, synchronous with avst_sink_u_startofpacket and avst_sink_u_valid. Repeated section IEs are valid when rx_sec_hdr_valid is asserted, synchronous with avst_sink_u_valid. On providing new section PRB fields in avst_sink_u_data, provide new section IEs with rx_sec_hdr_valid asserted.	rx_metadata_valid_o

Table 15. rx_metadata_o 128-bit output

Signal Name	Bitwidth	Direction	Description	Metadata Mapping
Reserved	32	Output	Reserved.	rx_metadata_o[127:96]
rx_u_seq_id	16	Output	SeqID of the packet, which is extracted from eCPRI transport header.	rx_metadata_o[95:80]
rx_u_pc_id	16	Output	PCID for eCPRI transport and RoEflowId for RoE transport	rx_metadata_o[79:64]
reserved	4	Output	Reserved.	rx_metadata_o[63:60]
rx_u_dataDirection	1	Output	gNB data direction. Value range: {0b=Rx (i.e. upload), 1b=Tx (i.e. download)}	rx_metadata_o[59]

rx_u_filterIndex	4	Output	Defines an index to the channel filter to use between IQ data and air interface. Value range: {0000b-1111b}	rx_metadata_o[58:55]
rx_u_frameId	8	Output	A counter for 10 ms frames (wrapping period 2.56 seconds), specifically frameId= frame number modulo 256. Value range: {0000 0000b-1111 1111b}	rx_metadata_o[54:47]
rx_u_subframeId	4	Output	A counter for 1ms subframes within 10 ms frame. Value range: {0000b-1111b}	rx_metadata_o[46:43]
rx_u_slotID	6	Output	The slot number within a 1ms subframe. All slots in one subframe are counted by this parameter. Value range: {00 0000b-00 1111b=slotID, 01 0000b-111111b=Reserved}	rx_metadata_o[42:37]
rx_u_symbolid	6	Output	Identifies a symbol number within a slot. Value range: {00 0000b-11 1111b}	rx_metadata_o[36:31]

rx_u_sectionId	12	Output	The sectionID maps U-plane data sections to the corresponding C-plane message (and Section Type) associated with the data. Value range: {0000 0000 0000b-1111 1111 1111b}	rx_metadata_o[30:19]
rx_u_rb	1	Output	Resource block indicator. Indicates if every resource block is used or every other resource is used. Value range: {0b=every resource block used; 1b=every other resource block used}	rx_metadata_o[18]
rx_u_startPrb	10	Output	The starting PRB of a user plane data section. Value range: {00 0000 0000b-11 1111 1111b}	rx_metadata_o[17:8]
rx_u_numPrb	8	Output	Defines the PRBs where the user plane data section is valid. Value range: {0000 0001b-1111 1111b, 0000 0000b = all PRBs in the specified SCS and carrier bandwidth }	rx_metadata_o[7:0]
rx_u_udCompHdr	8	Output	Defines the compression method and IQ bit width of the user data in a data section. Value range: {0000 0000b-1111 1111b}	N/A (rx_udcomphdr_o)

CSR Interface Signals

Table 16. CSR Interface Signals

Signal Name	Bit Width	Direction	Description
csr_address	16	Input	Configuration register address.
csr_write	1	Input	Configuration register write enable.
csr_writedata	32	Input	Configuration register write data.
csr_readdata	32	Output	Configuration register read data.
csr_read	1	Input	Configuration register read enable.
csr_readdatavalid	1	Output	Configuration register read data valid.
csr_waitrequest	1	Output	Configuration register wait request.

Fronthaul Compression IP Registers

Control and monitor fronthaul compression functionality through the control and status interface.

Table 17. Register Map

CSR_ADDRESS (Word Offset)	Register Name
0x0	compression_mode
0x1	tx_error
0x2	rx_error

Table 18. compression_mode Register

Bit Width	Description	Access	HW Reset Value
31:9	Reserved	RO	0x0
8:8	Functional mode: • 1’b0 is static compression mode • 1’b1 is dynamic compression mode	RW	0x0
7:0	Static user data compression header: • 7:4 is udIqWidth — 4’b0000 is 16 bits — 4’b1111 is 15 bits — : — 4’b0001 is 1 bit • 3:0 is udCompMeth — 4’b0000 is no compression — 4’b0001 is block floating point — 4’b0011 is µ-law • Others are reserved	RW	0x0

Table 19. tx Error Register

Bit Width	Description	Access	HW Reset Value
31:2	Reserved	RO	0x0
1:1	Invalid IqWidth. The IP sets Iqwidth to 0 (16-bit Iqwidth) if it detects invalid or unsupported Iqwidth.	RW1C	0x0
0:0	Invalid compression method. The IP drops the packet.	RW1C	0x0

Table 20. rx Error Register

Bit Width	Description	Access	HW Reset Value
31:8	Reserved	RO	0x0
1:1	Invalid IqWidth. The IP drops the packet.	RW1C	0x0
0:0	Invalid compression method. The IP sets the compression method to the following default supported compression method: • Enabled block-floating point only: default to block-floating point. • Enabled μ-law only: default to μ-law. • Enabled both block-floating point and μ-law: default to block-floating point.	RW1C	0x0

Fronthaul Compression Intel FPGA IPs User Guide Archive

For the latest and previous versions of this document, refer to: Fronthaul Compression Intel FPGA IP User Guide. If an IP or software version is not listed, the user guide for the previous IP or software version applies.

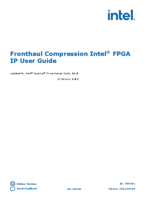
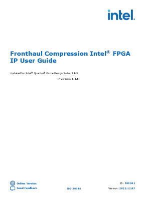
Document Revision History for the Fronthaul Compression Intel FPGA IP User Guide

Document Version	Intel Quartus Prime Version	IP Version	Changes
2022.08.08	21.4	1.0.1	Corrected metadata width 0 to 0 (Disable Metadata Ports).
2022.03.22	21.4	1.0.1	• Swapped signal descriptions: — tx_avst_sink_data and tx_avst_source_data — rx_avst_sink_data and rx_avst_source_data • Added *Device Supported Speed Grades* table • Added *Performance and Resource Usage*
2021.12.07	21.3	1.0.0	Updated ordering code.
2021.11.23	21.3	1.0.0	Initial release.

Online Version

Send Feedback
ID: 709301
UG-20346
Version: 2022.08.08
ISO 9001:2015 Registered

Documents / Resources

Fronthaul Compression Intel® FPGA IP User Guide Updated for Intel® Quartus® Prime Design Suite: 21.4 IP Version: 1.0.1 Online Version Send Feedback UG-20346 ID: 709301 Version: 2022.08.08	intel Fronthaul Compression FPGA IP [pdf] User Guide Fronthaul Compression FPGA IP, Fronthaul, Compression FPGA IP, FPGA IP
Fronthaul Compression Intel® FPGA IP User Guide Updated for Intel® Quartus® Prime Design Suite: 21.3 IP Version: 1.0.0 Online Version Send Feedback UG-20346 ID: 709301 Version: 2021.12.07	intel Fronthaul Compression FPGA IP [pdf] User Guide UG-20346, 709301, Fronthaul Compression FPGA IP, Fronthaul FPGA IP, Compression FPGA IP, FPGA IP