5G LDPC-V Intel® FPGA IP User Guide

Updated for Intel® Quartus® Prime Design Suite: **21.2**

IP Version: **3.0.0**

Subscribe

Send Feedback

UG-20251 | 2021.07.19

Latest document on the web: **PDF** | **HTML**

Contents

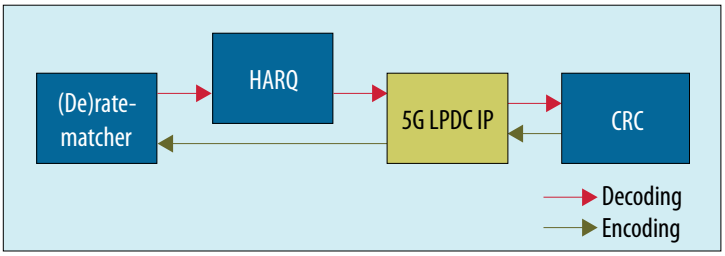
1. About the 5G LDPC-V Intel® FPGA IP

Low-density parity-check (LDPC) codes are linear error correcting codes that help you to transmit and receive messages over noisy channels. The 5G LDPC-V Intel® FPGA IP implements LDPC codes compliant with the 3rd Generation Partnership Project (3GPP) 5G specification for integration in your wireless design. LDPC codes offer better spectral efficiency than Turbo codes and support the high throughput for 5G new radio (NR).

The 5G LDPC-V IP is a complete channel coding IP that is optimized for virtual radio access networks (vRAN). The 5G LDPC-V IP is based on the 5G LDPC Intel FPGA IP and includes a 5G NR LDPC channel coder, which comprises:

- LDPC code block segmentation CRC module
- LDPC encoder and decoder
- LDPC rate matcher and derate matcher
- Hybrid automatic repeat request (HARQ) block (decoder only)

Figure 1. 5G LDPC-V IP



Related Information

- 3GPP New Radio Specification
  The final equivalents are Release 15, 3GPP Technical Specification Group RAN 1, NR:
  - (1) Multiplexing and channel coding, 3GPP TS 38.212 (v15.3.0)
  - (2) Physical layer procedures for data, 3GPP TS 38.214 (v15.3.0)
- 5G LPDC Intel FPGA IP User Guide

1.1. 5G LDPC-V Intel FPGA IP Features

- 3GPP 5G LDPC specification compliant
- For the transmitter:
  - CRC checker module (CRC24B without concatenation)
  - Rate matcher
  - Per-block modifiable code block length and code rate
- For the receiver:
  - Improved block-error rate (BLER) performance for high reliability signal-to-noise ratios (SNRs) for ultra reliable low-latency communications (URLLC)
  - 5 bits or 6 bits LLR input width
  - Derate matcher
  - Bypassable hybrid automatic repeat request (HARQ) block
  - Code block segmentation CRC module (CRC24B without concatenation)
  - Per-block modifiable code block length, code rate, and maximum number of iterations
  - Configurable input precision
  - Layered decoder scheduling architecture to double the speed of convergence compared to non-layered architecture
  - Early termination based on the syndrome check using four layers or full syndrome after each iteration.
- No external memory requirement
- Bit-accurate C models and MATLAB models for performance simulation
- Verilog HDL testbench option

Related Information

3GPP New Radio LDPC Specification

1.2. 5G LDPC-V Intel FPGA IP Device Family Support

The IP supports the following devices families:

- Intel Agilex™ devices
- Intel Arria® 10 devices
- Intel Stratix® 10 devices

For the device support levels for Intel FPGA IP, refer to *Device Support and Pin-Out Status* in the *Intel Quartus Prime Pro Edition: Version 21.2 Software and Device Support Release Notes*.

Related Information

Device Support and Pin-Out Status

1.3. Release Information for the 5G LDPC-V 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 1. 5G LDPC-V IP Release Information

Item	Description
Version	3.0.0
Release Date	June 2021

1.4. 5G LDPC-V Performance and Resource Utilization

Table 2. Performance and Resource Utilization

Shows the performance with Intel Quartus Prime Pro Edition v21.2.

Family	Speed Grade	Device	Component	Ave f_{MAX} (reduced 15% for margin) (MHz)	ALM	M20K	DSP Blocks
Intel Agilex	2	AGFB014R24B2I2V	Transmitter	606	14.7k	27	2
			Receiver (6-bit LLR)	497	79.5k	907	1
			Receiver (5-bit LLR)	504	71.9k	797	1
Intel Stratix 10	2	1SG280HU2F50E2VG	Transmitter	401	12.5k	27	2
			Receiver (6-bit LLR)	375	80.1k	907	1
			Receiver (5-bit LLR)	385	72.1k	797	1
Intel Stratix 10	2L	1SG280HU2F50E2LG	Transmitter	380	12.5k	27	2
			Receiver (6-bit LLR)	360	80.1k	907	1
			Receiver (5-bit LLR)	358	72.1k	797	1
Intel Arria 10	1	10AT115S1F45E1SG	Transmitter	363	12.2k	27	2
			Receiver (6-bit LLR)	286	67.4k	880	1
			Receiver (5-bit LLR)	299	70.0k	774	1

Related Information

Design Compilation
Understand how the Intel Quartus Prime software compiles and synthesizes your RTL design.

2. Getting Started with the 5G LDPC-V Intel FPGA IP

Related Information

- Introduction to Intel FPGA IP
- IP Catalog and Parameter Editor
  The IP Catalog displays the IP available for your project.
- Generating Intel FPGA IP
  Quickly configure Intel FPGA IP cores in the Intel Quartus Prime parameter editor. Double-click any component in the IP Catalog to launch the parameter editor. The parameter editor allows you to define a custom variation of the IP core. The parameter editor generates the IP variation synthesis and optional simulation files, and adds the `.ip` file representing the variation to your project automatically.

2.1. Installing and Licensing Intel FPGA IP Cores

The Intel Quartus Prime software installation includes the Intel FPGA IP library. This library provides many useful IP cores for your production use without the need for an additional license. Some Intel FPGA IP cores require purchase of a separate license for production use. The Intel FPGA IP Evaluation Mode allows you to evaluate these licensed Intel FPGA IP cores in simulation and hardware, before deciding to purchase a full production IP core license. You only need to purchase a full production license for licensed Intel IP cores after you complete hardware testing and are ready to use the IP in production.

The Intel Quartus Prime software installs IP cores in the following locations by default:

Figure 2. IP Core Installation Path

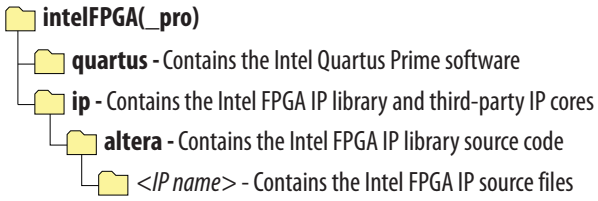


Table 3. IP Core Installation Locations

Location	Software	Platform
<*drive*>:\intelFPGA_pro\quartus\ip\altera	Intel Quartus Prime Pro Edition	Windows*
<*drive*>:\intelFPGA\quartus\ip\altera	Intel Quartus Prime Standard Edition	Windows
<*home directory*>:/intelFPGA_pro/quartus/ip/altera	Intel Quartus Prime Pro Edition	Linux*
<*home directory*>:/intelFPGA/quartus/ip/altera	Intel Quartus Prime Standard Edition	Linux

2.1.1. Intel FPGA IP Evaluation Mode

The free Intel FPGA IP Evaluation Mode allows you to evaluate licensed Intel FPGA IP cores in simulation and hardware before purchase. Intel FPGA IP Evaluation Mode supports the following evaluations without additional license:

- Simulate the behavior of a licensed Intel FPGA IP core in your system.
- Verify the functionality, size, and speed of the IP core quickly and easily.
- Generate time-limited device programming files for designs that include IP cores.
- Program a device with your IP core and verify your design in hardware.

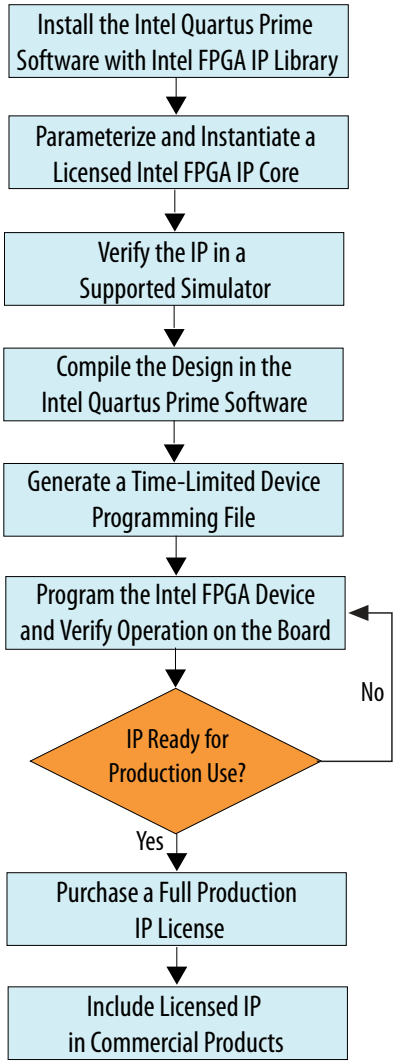
Intel FPGA IP Evaluation Mode supports the following operation modes:

- **Tethered**—Allows running the design containing the licensed Intel FPGA IP indefinitely with a connection between your board and the host computer. Tethered mode requires a serial joint test action group (JTAG) cable connected between the JTAG port on your board and the host computer, which is running the Intel Quartus Prime Programmer for the duration of the hardware evaluation period. The Programmer only requires a minimum installation of the Intel Quartus Prime software, and requires no Intel Quartus Prime license. The host computer controls the evaluation time by sending a periodic signal to the device via the JTAG port. If all licensed IP cores in the design support tethered mode, the evaluation time runs until any IP core evaluation expires. If all of the IP cores support unlimited evaluation time, the device does not time-out.
- **Untethered**—Allows running the design containing the licensed IP for a limited time. The IP core reverts to untethered mode if the device disconnects from the host computer running the Intel Quartus Prime software. The IP core also reverts to untethered mode if any other licensed IP core in the design does not support tethered mode.

When the evaluation time expires for any licensed Intel FPGA IP in the design, the design stops functioning. All IP cores that use the Intel FPGA IP Evaluation Mode time out simultaneously when any IP core in the design times out. When the evaluation time expires, you must reprogram the FPGA device before continuing hardware verification. To extend use of the IP core for production, purchase a full production license for the IP core.

You must purchase the license and generate a full production license key before you can generate an unrestricted device programming file. During Intel FPGA IP Evaluation Mode, the Compiler only generates a time-limited device programming file (<*project name*>_time_limited.sof) that expires at the time limit.

Figure 3. Intel FPGA IP Evaluation Mode Flow



Note: Refer to each IP core's user guide for parameterization steps and implementation details.

Intel licenses IP cores on a per-seat, perpetual basis. The license fee includes first-year maintenance and support. You must renew the maintenance contract to receive updates, bug fixes, and technical support beyond the first year. You must purchase a full production license for Intel FPGA IP cores that require a production license, before generating programming files that you may use for an unlimited time. During Intel FPGA IP Evaluation Mode, the Compiler only generates a time-limited device programming file (`<project name>_time_limited.sof`) that expires at the time limit. To obtain your production license keys, visit the Self-Service Licensing Center.

The Intel FPGA Software License Agreements govern the installation and use of licensed IP cores, the Intel Quartus Prime design software, and all unlicensed IP cores.

Related Information

- Intel FPGA Licensing Support Center
- Introduction to Intel FPGA Software Installation and Licensing

2.1.2. 5G LDPC-V IP Timeout Behavior

All IP in a device time out simultaneously when the most restrictive evaluation time is reached. If a design has more than one IP, the time-out behavior of the other IP may mask the time-out behavior of a specific IP .

For IP, the untethered time-out is 1 hour; the tethered time-out value is indefinite. Your design stops working after the hardware evaluation time expires. The Quartus Prime software uses Intel FPGA IP Evaluation Mode Files (`.ocp`) in your project directory to identify your use of the Intel FPGA IP Evaluation Mode evaluation program. After you activate the feature, do not delete these files.

When the evaluation time expires, for the transmitter `o_source_data` goes low; for the receiver `o_source_data` and `o_ldpc_metrics` go low.

Related Information

AN 320: OpenCore Plus Evaluation of Megafunctions

3. Designing with the 5G LDPC-V Intel FPGA IP

3.1. 5G LDPC-V IP Directory Structure

The IP includes a `c_model`, `matlab`, `src`, `simulation_scripts`, and `test_data` directory.

Table 4. Files in the `c_model` Directory

Other .c and .cpp files in `c_model` are obfuscated models of different blocks.

File	Description
`ldpc5g_tx_chain.c`	Clear-text transmitter chain wrapper
`ldpc5g_tx_chain_test.c`	Clear-text transmitter chain testbench
`ldpc5g_rx_chain.cpp`	Clear-text receiver chain wrapper
`ldpc5g_rx_chain_test.cpp`	Clear-text receiver chain testbench
`ldpc5g_gen_tc.c`	Clear-text test cases for transmitter and receiver chain

Table 5. Directories in the `src` Directory

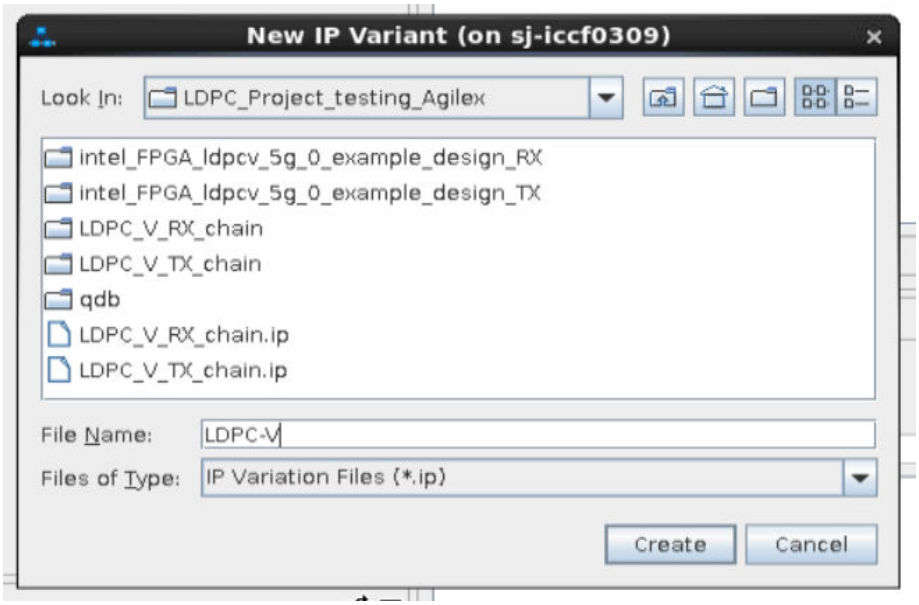
File	Description
`aldec`	Encrypted RTL files for Aldec
`cadence`	Encrypted RTL files for NCSim
`mentor`	Encrypted RTL files for ModelSim
`synopsys`	Encrypted RTL files for VCS

3.2. Generating a 5G LPDC-V IP

To include the IP in a design, generate the IP in the Intel Quartus Prime software. Or optionally, you can generate a design example that includes the generated 5G LDPC-V IP, a C model, a MATLAB model, simulation scripts, and test data. The software generates no hardware example in **Generate Example Design**.

1. Create a New Intel Quartus Prime project
2. Open IP Catalog.
3. Select **DSP ➤ Error Detection and Correction ➤ 5G LDPC-V** and click **Add**
4. Enter a name for your IP variant and click **Create**.

Figure 4. IP Variant File Name



The name is for both the top-level RTL module and the corresponding `.ip` file.

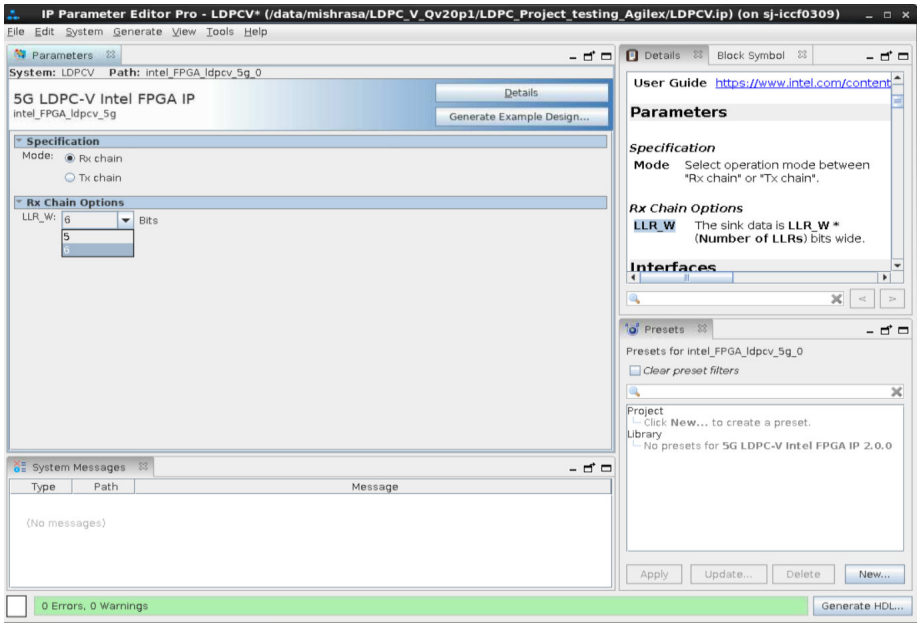
The parameter editor for this IP appears.

5. Choose your parameters.

Table 6. 5G LDPC-V Parameters

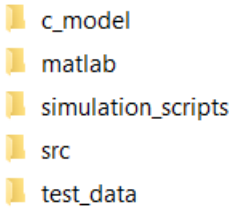
Parameter Name	Values	Description
Mode	Rx chain Tx chain	Select between receiver or transmitter.
LLR_W (receiver only)	5 6	Input LLR bitwidth. **LLR_W**=5, gives 3 integral bits and 2 fractional bits, i.e. a range of [-4.0, 3.75] **LLR_W**=6, gives 4 integral bits and 2 fractional bits, i.e. a range of [-8.0, 7.75]
NUM_DECODERS (receiver only)	1 2	The number of decoders that the IP builds.
MAX_LF_DECODER1 (receiver only)	96 128 192	The maximum lifting factor for the packet that the second decoder can process. Only valid when **NUM_DECODERS** is to 2

Figure 5. 5G LDPC Parameter Editor



6. For an optional design example, click **Generate Example Design**
   The software creates a design example of the transmitter or receiver.

Figure 6. Design Example Directory Structure



7. Click **Generate HDL**.

Intel Quartus Prime generates the RTL and the files necessary to instantiate the IP in your design and synthesize it.

Related Information

Generating IP Cores
Use this link for the Quartus Prime Pro Edition Software.

3.3. Simulating the 5G LDPC-V IP

Verify that the RTL behaves the same as these models.

Before simulating, generate a 5G LDPC-V design example.

1. Simulate the transmitter with the C-model:
   a. Go to the `c_model\` directory.

b. Compile the C code of the transmitter chain.

```
>> gcc -lm ldpc5g_tx_chain_test.c -o run_tx
```

c. Run the executable.

The C program takes three arguments, where the first is the starting test case index, second is the number of test cases, and the third is enabling or disenabling HARQ. For example, the command `run_tx 10 15 1` generates test cases from the 10th to the 25th with HARQ enabled.

The executable generates `tx_param.txt` `tx_in.txt`, and `tx_out.txt`.

2. Simulate the receiver with the C-model (always simulate the transmitter before you simulate the receiver and with the same arguments):

a. Go to the `c_model\` directory.

b. Compile the C code of the receiver chain.

```
>> g++ -lm ldpc5g_rx_chain_test.cpp -o run_rx
```

c. Run the executable.

The C program takes three arguments, where the first is the starting test case index, second is the number of test cases, and the third is enabling or disenabling HARQ. For example, the command `run_rx 10 15 1` generates test cases from the 10th to the 25th with HARQ enabled.

The executable takes `tx_out.txt` and generates `rx_in.txt`, `rx_param.txt`, and `rx_out.txt`.

3. Simulate with the VCS simulator:

a. Modify these files, if you want to see the waveform of the top level design:

- For the transmitter, in <*Example Design Directory*>`\src\ldpcv5g_tx_top_tb.sv` uncomment the following lines:
  - `// Dump waveform, may not work with your simulator`
  - `$vcdplusfile("./tx.vpd");`
  - `$vcdplusmemon();`
  - `$vcdpluson();`
- For the receiver, in <*Example Design Directory*>`\src\ldpcv5g_rx_top_tb.sv` uncomment the following lines:
  - `// Dump waveform, may not work with your simulator`
  - `$vcdplusfile("./rx.vpd");`
  - `$vcdplusmemon();`
  - `$vcdpluson();`
- In <*Example Design Directory*>`\simulation_scripts\synopsys\vcs\vcs_setup.sh`, modify `USER_DEFINED_ELAB_OPTIONS="-debug_access+r"`

b. Run `vcs_setup.sh` from <*Example Design Directory*>`\simulation_scripts\synopsys\vcs\`.

```
>> source vcs_setup.sh
>> simv
```

For other simulators (Aldec, Cadence, Mentor or Synopsys), run the script from the corresponding simulator directory in <*Example Design Directory*>\simulation_scripts\.

4. Simulate 5G LDPC-V IP with the MATLAB model:
   a. In MATLAB, run `make.m` from the `\matlab\` directory.

      ```
      >> make
      ```

      MATLAB generates MEX.
   b. Check if `p_mat/` exists in `matlab/`, if not, copy it from `../c_model/p_mat/` in the `/matlab/` directory.
   c. Run `ldpc5g_txrx_chain_test.m`, which is an example testbench to call the transmitter MATLAB function (`ldpc5g_tx_chain.m`) and receiver MATLAB function (`ldpc5g_rx_chain.m`).

      ```
      >> ldpc5g_txrx_chain_test
      ```

3.4. 5G LDPC-V Simulation Results

Figure 7. Transmitter Top-level Simulation Results

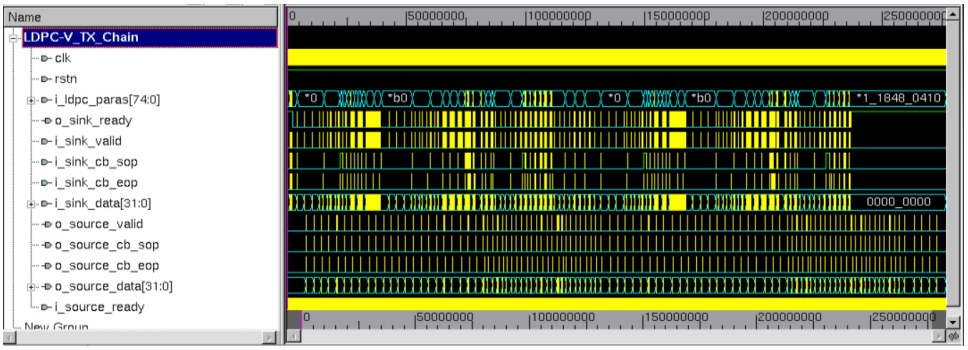


Figure 8. Transmitter Top-level Simulation Results: Zoomed View

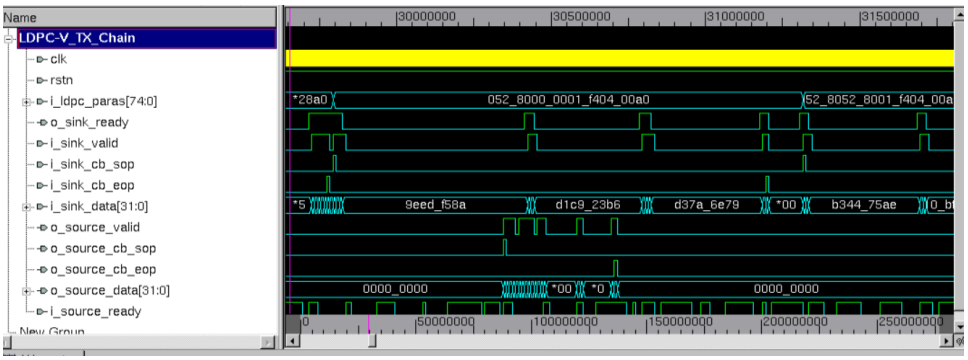


Figure 9. Receiver Top-level Simulation Results

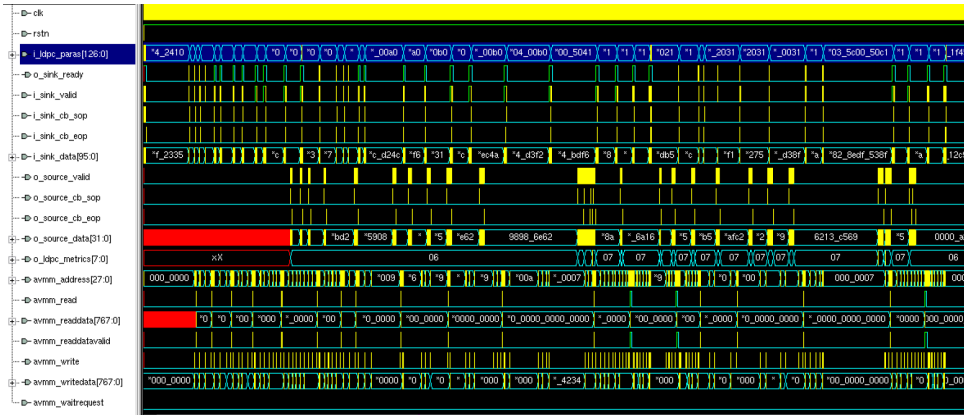
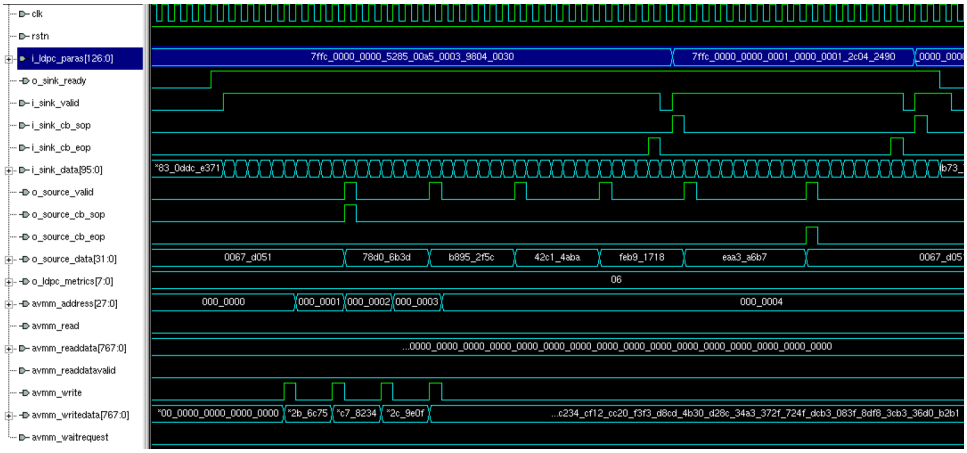


Figure 10. Receiver Top-level Simulation Results: Zoomed View



4. 5G LDPC-V Intel FPGA IP Functional Description

The 5G LDPC-V Intel FPGA IP comprises an encoder and a decoder.

4.1. 5G LDPC-V Transmitter Functional Description

The LDPC code block segmentation CRC module attaches the CRC for each code block and inserts null bit. For more information, refer to 3GPP TS 38.212. Also, the LDPC code block segmentation CRC module controls its input pace.

The LDPC encoder takes code block data from LDPC code block segmentation CRC module and produces the encoded code block LDPC for the rate matcher.

The LDPC rate matcher implements the rate matching processing (refer to *3GPP TS 38.212*) and concatenates the rate matched code block for its output.

All the submodules' interfaces are based on Avalon streaming interface specification.

Related Information

Avalon Interface Specifications

4.1.1. 5G LDPC-V Transmitter Signals

All signals are synchronous to `clk`.

Figure 11. Transmitter Signals

This figure does not show the Avalon streaming interface signals

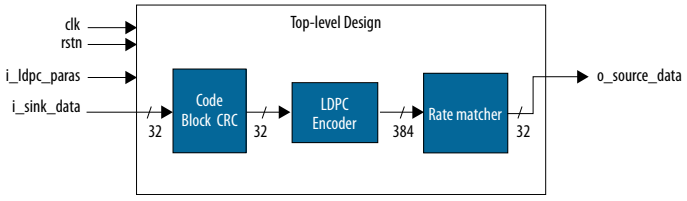


Table 7. Transmitter Top-Level Signals

Name	Direction	Description
clk	Input	Clock. All signals are synchronous to clk.
rstn	Input	Reset, active-low. Assert for at least for 10 clock cycles.
i_ldpc_paras	Input	Aligns with i_sink_cb_sop [0] is base graph (BG) (1 bit), where: 0:BG1, 1:BG2 [6:1] is Zc_idx (6 bits) BG is the index of the lifting factor Zc. Choose Zc from *Table 5.3.2-1 of TS 38.212*. Look up the index of Zc in *Lifting Factor Index* table. [7] is use_crc (1 bit), where 0: not use code block CRC; 1: use code block CRC (CRC24B). [17:8] is the number of null bits (10 bits). Plug in the value of K-K', where K=22Zc(BG1) or 10Zc(BG2). Check the definition of K and K' in 5.2.2 in *TS 38.212*. [20:18] is the code rate index (3 bits). The *Code Rate* table shows the code rate choices supported by the LDPC encoder and decoder IP. The code rate is not target code rate. [23:21] is Qm_idx (3 bits): • 0:BPSK • 1:QPSK • 2:16QAM • 3: 64QAM • 4:256QAM [44:24] is E (21 bits), output length of the rate matcher, or equivalently, input length of the derate matcher. [59:45] is k0 position (15 bits). Calculate k0 based on *Table 5.4.2.1-2 of TS 38.212*. [74:60] is limited circular buffer size.
i_sink_data	Input	32 message bits. • [0]: msg seq# 0 • [1]: msg seq# 1 • … • [31]: msg seq# 31 Total number of input bits is K' if CRC is not used; K'-24 if CRC is used
i_sink_valid	Input	Qualifies the i_sink_data signal When i_sink_valid is not asserted, the IP stops processing input until i_sink_valid signal is reasserted. Assert when o_sink_ready is asserted.
i_sink_cb_sop	Input	Indicates the start of an incoming packet You cannot have two valid SOPs in any five consecutive clock cycles
i_sink_cb_eop	Input	Indicates the end of an incoming packet
i_source_ready	Input	Assert this signal to inform the transmitter that the downstream is ready to take transmitter outputs. The downstream should have readyAllowance of up to 30 clock cycles. The transmitter can produce up to 30 cycles of valid data after i_source_ready is deasserted.
o_sink_ready	Output	Indicates that the receiver is ready to receive data in the next clock cycle. Ignore when rst is asserted. The IP can backpressure incoming data by deasserting this signal. When o_sink_ready==0 is observed, deassert i_sink_valid in the next clock cycle.
o_source_data	Output	32 output bits from rate matcher • data[0] -> bit0 • data[1] -> bit1 • … • data[31] -> bit31 Total number of output bits is E.

continued...

Name	Direction	Description
o_source_valid	Output	The transmitter asserts this signal when o_source_data holds valid data.
o_source_cb_sop	Output	The transmitter asserts this signal to mark the start of a packet.
o_source_cb_eop	Output	The transmitter asserts this signal to mark the end of a packet

Related Information

5G LDPC-V Lifting Factor and Code Rate Indexes on page 18

4.1.2. 5G LDPC-V Lifting Factor and Code Rate Indexes

Use these values for the i_ldpc_paras parameter.

Table 8. Lifting Factor Index

Zc	Zc_idx	Zc	Zc_idx
4	2	52	27
5	3	56	28
6	4	60	29
7	5	64	30
8	6	72	31
9	7	80	32
10	8	88	33
11	9	96	34
12	10	104	35
13	11	112	36
14	12	120	37
15	13	128	38
16	14	144	39
18	15	160	40
20	16	176	41
22	17	192	42
24	18	208	43
26	19	224	44
28	20	240	45
30	21	256	46
32	22	288	47
36	23	320	48
40	24	352	49
44	25	384	50
48	26	-	-

Table 9. Code Rate Index

Code Rate Index	Code Rate	Base Graph 1		Base Graph 2	
		Number of Rows in Parity Check Matrix	Number of Columns in Parity Check Matrix	Number of Rows in Parity Check Matrix	Number of Columns in Parity Check Matrix
000	1/5	NA	NA	42	52
001	1/3	46	68	22	32
010	2/5	35	57	17	27
011	1/2	24	46	12	22
100	2/3	13	35	7	17
101	22/30 (~3/4)	10	32	NA	NA
110	22/27 (~5/6)	7	29	NA	NA
111	22/25 (~8/9)	5	27	NA	NA

4.2. 5G LDPC-V Receiver Functional Description

The receiver comprises: a LDPC derate matcher, HARQ block, LDPC decoder, and LDPC code block segmentation CRC module.

LDPC derate matcher implements the rate recovery process, which is to reverse rate matching process. Refer to *3GPP Specification 38.212*.

The HARQ block stores and combines derate matcher outputs from previous and current transmissions for LDPC decoder.

LDPC decoder implements the LDPC decode process, which is to reverse the LDPC encode process. Refer to *3GPP Specification 38.212*.

The LDPC code block segmentation CRC module checks the 24-bit CRC embedded in the LDPC decoded bits. The LDPC code block segmentation CRC module. Refer to *3GPP Specification 38.212*.

4.2.1. 5G LDPC-V Receiver Signals

All signals are synchronous to `clk`.

Figure 12. Receiver Top-level Block Diagram

This figure does not show the Avalon streaming interface signals

.

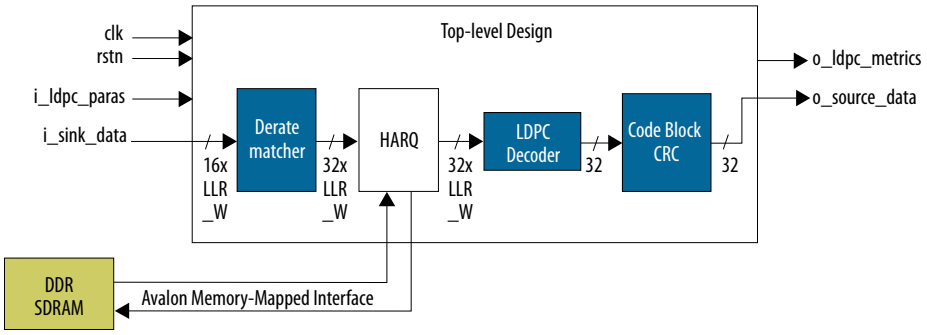


Table 10. Receiver Top-level Signals

Name	Direction	Description
clk	Input	Clock. All signals are synchronous to clk.
rstn	Input	Reset, active-low. Assert for at least for 10 clock cycles. Do not send data or parameters when reset is asserted
i_ldpc_paras	Input	Align with i_sink_cb_sop. [0] is base graph (BG) (1 bit): • 0:BG1 • 1:BG2 [6:1] is Zc_idx (6 bits), the index of lifting factor Zc. Choose Zc from *Table 5.3.2-1 of TS38.212*. Look up the index of Zc in the *Lifting Factor* table.. [7] is use_crc (1 bit) • 0: not use code block CRC • 1: use code block CRC (CRC24B) [17:8] is the number of null bits (10 bits). Plug in the value of K-K', where K=22Zc(BG1) or 10Zc(BG2). Check the definition of K and K' in *section 5.2.2 in TS 38.212* [20:18] is the code rate index (3 bits). The *Code Rate Index* table shows the code rate choices supported by the LDPC encoder and decoder IP. The code rate is not the target code rate. [23:21] is Qm_idx (3 bits): • 0:BPSK • 1:QPSK • 2:16QAM • 3: 64QAM • 4:256QAM [44:24] is E (21 bits), the output length of the rate matcher, or equivalently, input length of the de-rate matcher [59:45] is k0 position (15 bits).Calculate k0 based on *Table 5.4.2.1-2 of TS 38.212* [65:60] is max_iter (6 bits), the maximum number of iterations of LDPC decoding. [66] is use_harq (1 bit) • 0: not use HARQ • 1: use HARQ [81:67] is cb_old_len (15 bit), the length of the previously combined code block already stored in DRR. cb_old_len should be less than or equal to codeword_length of the current frame and the max codeword_length of all the previous frames (first frame is when use_harq = 0).

continued...

Name	Direction	Description
		[109:82] is `cb_ddr_addr` (28 bit), the base address to DDR for the combined code block [110] is `et_dis` (1 bit), the decoder early termination disable. • 0: enable early termination • 1: disable early termination [111] is `red_synd` (1 bit), the decoder reduce syndrome checks • 0: full syndrome check • 1: reduced syndrome check [124:112] is the limited circular buffer size.
`i_sink_data`	Input	16 LLRs x LLR_W bits per LLR [LLR_W-1:0]: LLR seq# 0, [LLR_W*2-1:LLR_W]: LLR seq #1,...
`i_sink_valid`	Input	Qualifies the `i_sink_data` signal When i_sink_valid is not asserted, the IP stops processing input until i_sink_valid signal is reasserted. Asserted when o_sink_ready is asserted.
`i_sink_cb_sop`	Input	Indicates the start of an incoming packet You cannot have two valid SOPs in any four consecutive clock cycles
`i_sink_cb_eop`	Input	Indicates the end of an incoming packet
`o_sink_ready`	Output	Indicates that the receiver is ready to receive data in the next clock cycle. Ignore when `rstn` is asserted. The IP can backpressure incoming data by deasserting this signal: when you see o_sink_ready==0, deassert i_sink_valid in the next clock cycle
`o_source_data`	Output	LDPC decoded hard bits, including code block CRC bit, not including NULL padding (K-K'). data[0] -> bit0, data[1] -> bit1,…,data[31] -> bit31
`o_source_valid`	Output	The receiver asserts this signal when `o_source_data` holds valid data
`o_source_cb_sop`	Output	The receiver asserts this signal to indicate the start of a packet
`o_source_cb_eop`	Output	The receiver asserts this signal to indicate the end of a packet
`o_ldpc_metrics`	Output	[0] is source_crc_pass (1 bit) • 1:pass • 0:fail or not checked, align with EOP [1] is source_et_pass (1 bit): refer to *5G LDPC Intel FPGA IP User Guide*, align with SOP [7:2] is source_iter (6 bits): refer to *5G LDPC Intel FPGA IP User Guide*, align with SOP.
`avmm_address`	Output	DDR SDRAM 28 bits address (Avalon memory-mapped master)
`avmm_read`	Output	DDR read request (Avalon memory-mapped master)
`avmm_readdata`	Input	DDR read data with 128*LLR_W bit width (Avalon memory-mapped master)
`avmm_readdatavalid`	Input	DDR read data valid (Avalon memory-mapped master)
`avmm_write`	Output	DDR write request (Avalon memory-mapped master)
`avmm_writedata`	Output	DDR write data with 128*LLR_W bit width (Avalon memory-mapped master)
`avmm_waitrequest`	Input	DDR wait request (Avalon memory-mapped master)

Related Information

5G LDPC-V Lifting Factor and Code Rate Indexes on page 18

4.3. Avalon Streaming Interfaces in DSP Intel FPGA IP

Avalon streaming interfaces define a standard, flexible, and modular protocol for data transfers from a source interface to a sink interface.

The input interface is an Avalon streaming sink and the output interface is an Avalon streaming source. The Avalon streaming interface supports packet transfers with packets interleaved across multiple channels.

Avalon streaming interface signals can describe traditional streaming interfaces supporting a single stream of data without knowledge of channels or packet boundaries. Such interfaces typically contain data, ready, and valid signals. Avalon streaming interfaces can also support more complex protocols for burst and packet transfers with packets interleaved across multiple channels. The Avalon streaming interface inherently synchronizes multichannel designs, which allows you to achieve efficient, time-multiplexed implementations without having to implement complex control logic.

Avalon streaming interfaces support backpressure, which is a flow control mechanism where a sink can signal to a source to stop sending data. The sink typically uses backpressure to stop the flow of data when its FIFO buffers are full or when it has congestion on its output.

Related Information

Avalon Interface Specifications

4.4. 5G LDPC-V Throughput and Latency

Throughput and latency scales linearly with the clock frequency.

Table 11. 5G LDPC-V Transmitter Throughput and Latency

Encoding chain clock frequency =268 MHz (arbitrary); Qm = 2; k0 = 0. Throughput is the number of user bits divided by the time difference between two consecutive sink SOPs when the design is running at full capacity. Latency is the time difference between the sink SOP and source SOP when the design is ready to process the input immediately.

BG	Z	E	Throughput (Gbps)	Latency (µs)
0	12	316	0.220	2.526
0	12	474	0.220	2.593
0	12	632	0.220	2.664
0	12	790	0.220	2.731
0	12	948	0.220	2.731
0	12	1106	0.220	2.731
0	12	1264	0.220	2.731
0	12	1422	0.220	2.731
0	12	1580	0.220	2.731
0	192	5068	3.359	3.787
0	192	7602	3.359	4.377
0	192	10136	3.127	4.966

continued...

0	192	12670	2.573	5.556
0	192	15204	2.310	5.556
0	192	17738	1.990	5.556
0	192	20272	1.747	5.556
0	192	22806	1.557	5.556
0	192	25340	1.405	5.556
0	384	10136	6.254	4.743
0	384	15204	4.371	5.929
0	384	20272	3.359	7.108
0	384	25340	2.728	8.287
0	384	30408	2.346	8.287
0	384	35476	2.016	8.287
0	384	40544	1.767	8.287
0	384	45612	1.572	8.287
0	384	50680	1.417	8.287
1	12	240	0.155	1.519
1	12	360	0.155	1.571
1	12	480	0.155	1.616
1	12	600	0.155	1.672
1	12	720	0.155	1.672
1	12	840	0.155	1.672
1	12	960	0.155	1.672
1	12	1080	0.155	1.672
1	12	1200	0.155	1.672
1	192	3840	2.486	2.396
1	192	5760	2.297	2.843
1	192	7680	1.812	3.291
1	192	9600	1.496	3.739
1	192	11520	1.376	3.739
1	192	13440	1.186	3.739
1	192	15360	1.042	3.739
1	192	17280	0.929	3.739
1	192	19200	0.838	3.739
1	384	7680	3.702	3.201
1	384	11520	2.586	4.097
1	384	15360	1.987	4.993
1	384	19200	1.613	5.888

continued...

1	384	23040	1.402	5.888
1	384	26880	1.205	5.888
1	384	30720	1.057	5.888
1	384	34560	0.941	5.888
1	384	38400	0.848	5.888

Table 12. 5G LDPC-V Receiver Throughput and Latency

Decoding chain clock frequency = 268 MHz (arbitrary), Qm = 2, k0 = 0, Decoder Iter.= 8. Throughput is the number of decoded bits divided by the time difference between two consecutive sink SOPs when the design is running at full capacity. Latency is the time difference between the sink SOP and source SOP when the design is ready to process the input immediately.

BG	Z	E	Throughput (Gbps)	Latency (µs)
0	12	316	0.086	17.668
0	12	474	0.040	39.985
0	12	632	0.032	49.951
0	12	790	0.027	58.530
0	12	948	0.027	58.567
0	12	1106	0.027	58.604
0	12	1264	0.027	58.937
0	12	1422	0.027	57.955
0	12	1580	0.027	58.063
0	192	5068	1.374	16.280
0	192	7602	0.644	38.806
0	192	10136	0.512	47.541
0	192	12670	0.430	55.481
0	192	15204	0.431	51.545
0	192	17738	0.430	50.907
0	192	20272	0.430	51.235
0	192	22806	0.429	50.276
0	192	25340	0.430	48.806
0	384	10136	2.748	15.507
0	384	15204	1.288	36.190
0	384	20272	1.022	43.489
0	384	25340	0.859	50.769
0	384	30408	0.861	47.310
0	384	35476	0.859	49.108
0	384	40544	0.859	48.836
0	384	45612	0.771	48.104
0	384	50680	0.696	49.993

continued...

1	12	240	0.050	17.317
1	12	360	0.032	25.922
1	12	480	0.020	39.414
1	12	600	0.020	39.500
1	12	720	0.020	39.459
1	12	840	0.020	39.448
1	12	960	0.020	39.388
1	12	1080	0.020	39.470
1	12	1200	0.020	39.433
1	192	3840	0.792	17.448
1	192	5760	0.506	26.604
1	192	7680	0.315	40.071
1	192	9600	0.314	39.698
1	192	11520	0.314	39.448
1	192	13440	0.313	38.840
1	192	15360	0.312	39.004
1	192	17280	0.312	38.619
1	192	19200	0.311	38.287
1	384	7680	1.571	17.522
1	384	11520	1.004	26.922
1	384	15360	0.625	40.108
1	384	19200	0.627	39.903
1	384	23040	0.622	39.231
1	384	26880	0.583	38.899
1	384	30720	0.513	39.940
1	384	34560	0.458	40.437
1	384	38400	0.414	40.799

5. 5G LPDC-V IP User Guide Archive

If an IP core version is not listed, the user guide for the previous IP core version applies.

IP Core Version	intel Quartus Prime Version	User Guide
0.1.0	19.2	5G LDPC-V IP User Guide
2.0.0	20.1	5G LDPC-V IP User Guide
2.0.0	20.2	5G LDPC-V IP User Guide

6. Document Revision History for the 5G LDPC-V Intel FPGA IP User Guide

Date	IP Version	Intel Quartus Prime Software Version	Changes
2021.07.19	3.0.0	21.2	• Deleted *LDPC-V Requirements* and *LDPC-V Limitaitons* • Updated performance table • Added two new parameters • Updated signal descriptions: — `i_ldpc_paras` — `i_source_ready` • Updated various receiver signal descriptions • Removed some entries in *Throughput and Latency* table
2020.08.19	2.0.0	20.2	• Corrected version to 20.2 • Corrected performance table
2020.06.30	2.0.0	20.1	• Corrected descriptions for: — `avmm_address` — `avmm_readdata` — `avmm_writedata` • Added new device to *Performance and Resource Utilization*
2020.06.02	2.0.0	20.1	• Removed derate matcher, HARQ, and decoder signal tables. • Removed code block CRC, encoder, and rate matcher signal tables, • Corrected and added signals to *Receiver Top-Level Signals* table • Added *Throughput and Latency*. • Added *Performance and Resource Utilization* • Added support for Intel Agilex devices • Updated simulation timing diagrams • Added *IP Requirements* and *IP Limitations*. • Removed second decoder option parameter.
2019.09.02	0.1.0	19.2	Corrected 5G LDPC-V block diagram.
2019.08.30	0.1.0	19.2	Initial release.