5G Polar Intel® FPGA IP User Guide

Updated for Intel® Quartus® Prime Design Suite: **20.3**

IP Version: **1.0.0**

Subscribe

Send Feedback

UG-20308 | 2020.10.15

Latest document on the web: PDF | HTML

Contents

1. About the 5G Polar Intel® FPGA IP

The IP implements polar codes compliant with the 3rd Generation Partnership Project (3GPP) 5G specification for integration in your wireless design. Polar codes support the high throughput for 5G new radio (NR).

The IP comprises:

- Polar encoder and polar list decoder with a list size of 4 or 8
- Interleaver and deinterleaver
- CRC encoder and decoder

Polar codes represent a new emerging class of error-correcting codes with power to approach the capacity of a discrete memoryless channel based on the recent invention by Arikan. This new code family is based on a channel polarization concept transforming independent channels into synthesized or polarized channels with different reliabilities: the good and the bad channels. The IP recursively applies such polarization transformation over the resulting channels such that the channels are polarized. The polarized channel encoder transmits information bits (i.e., free bits) over the noiseless channels while assigning fixed bits (i.e., frozen bits) to the noisy ones.

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)

1.1. 5G Polar Intel® FPGA IP Features

- Complies with the 3GPP 5G Polar specification
- Run-time configurable code block length, code rate, frozen bit, and parity check bit locations with optional reconfiguration for each code block
- Code block length from 32, 64, 128, 256, 512 and 1024
- Optional CRC, including CRC6, 11, 16, 24a, 24b and 24c
- Optional DCI format with RNTI scrambling for CRC24c
- Optional interleaving and deinterleaving
- Successive cancellation list decoding scheme, compile-time configurable list size, from L=4 or L=8
- Decoder output buffering allows the downstream to receive result while the decoder processes the next data block

- C and MATLAB bit-accurate models for performance simulation and RTL test vector generation
- System Verilog HDL testbench
- Avalon® streaming input and output interfaces

1.2. 5G Polar 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 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 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. 5G Polar IP Device Family Support

Device Family	Support
Intel® Agilex™	Advance
Intel Stratix® 10	Final
Other device families	No support

Intel Agilex devices meet final support when all timing models go final.

1.3. Release Information for the 5G Polar Intel FPGA IP

IP versions are the same as the Intel Quartus® Prime Design Suite software versions up to v19.1. From Intel Quartus Prime Design Suite software version 19.2 or later, IP cores have a new IP versioning scheme.

The IP version (X.Y.Z) number may change from one Intel Quartus Prime software version to another. A change in:

- X indicates a major revision of the IP. If you update your 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 2. 5G Polar IP Release Information

Item	Description
Version	1.0.0
Release Date	September 2020
Ordering Code	IP-POLAR

1.4. 5G Polar IP Performance and Resource Utilization

Table 3. 5G Polar IP Performance and Resource Utilization

The table shows f_{MAX} frequency reduced by 15%. The actual average is f_{MAX} x 1.15

Device	Speed Grade	Component	f_{MAX} (MHz)	ALM	M20K
Intel Agilex AGFA014R24A2E2VR0	-2V with extended temperature range	Encoder	632	3.3k	2
		Decoder NUM_LIST=4	503	8.2k	33
		Decoder NUM_LIST=8	436	25k	56
Intel Stratix 10 1SG280LU3F50E2LG	-2L with extended temperature range	Encoder	441	3.5k	3
		Decoder NUM_LIST=4	398	8k	33
		Decoder NUM_LIST=8	381	24k	56

2. Getting Started with the 5G Polar Intel FPGA IP

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 1. IP Core Installation Path

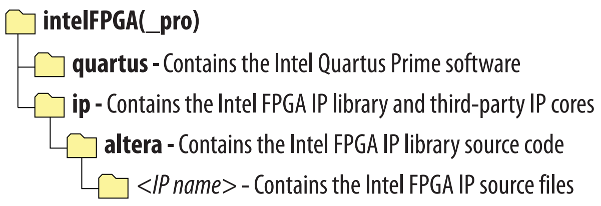

intelFPGA(_pro)
quartus - Contains the Intel Quartus Prime software
ip - Contains the Intel FPGA IP library and third-party IP cores
altera - Contains the Intel FPGA IP library source code
<IP name> - Contains the Intel FPGA IP source files

Table 4. 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.

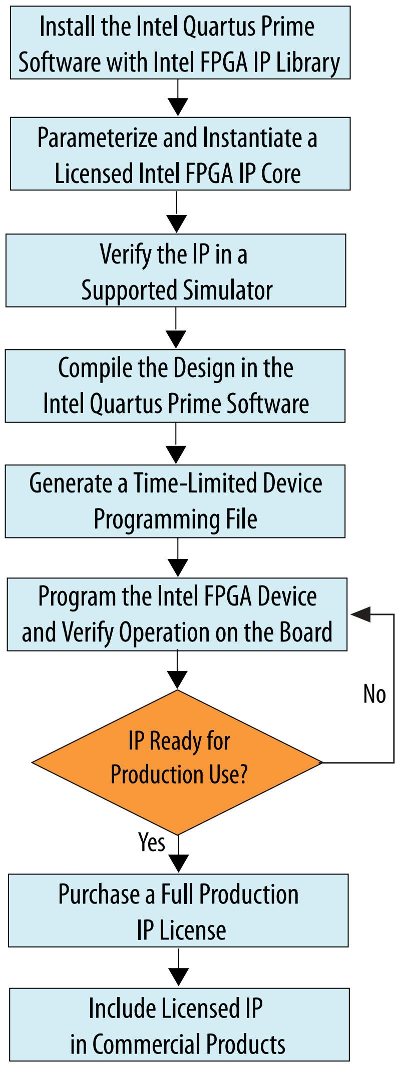
Intel FPGA IP Evaluation Mode supports the following operation modes:

- **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 2. Intel FPGA IP Evaluation Mode Flow


Install the Intel Quartus Prime Software with Intel FPGA IP Library
Parameterize and Instantiate a Licensed Intel FPGA IP Core
Verify the IP in a Supported Simulator
Compile the Design in the Intel Quartus Prime Software
Generate a Time-Limited Device Programming File
Program the Intel FPGA Device and Verify Operation on the Board
No
IP Ready for Production Use?
Yes
Purchase a Full Production IP License
Include Licensed IP in Commercial Products

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 Polar 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, `source_data` goes low.

Related Information

AN 320: OpenCore Plus Evaluation of Megafunctions

3. Designing with the 5G Polar Intel FPGA IP

3.1. 5G Polar IP Directory Structure

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

3.2. Generating a 5G Polar 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 IP, a C model, a MATLAB model, and simulation scripts.

1. Create a New Intel Quartus Prime project
2. Open IP Catalog.
3. Select **DSP ➤ FEC ➤ 5G Polar** and click **Add**
4. Enter a name for your IP variant and click **Create**.

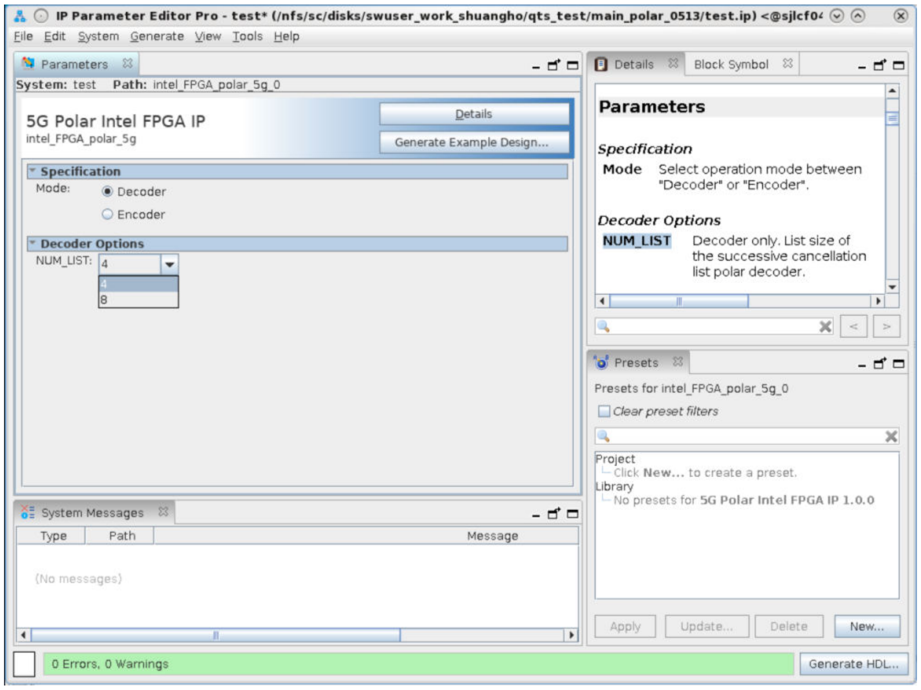
   The name is for both the top-level RTL module and the corresponding `.ip` file.

   The parameter editor for this IP appears.
5. Choose your parameters.

Table 5. 5G Polar Parameters

Parameter Name	Values	Description
Mode	Encoder Decoder	Select between encoder or decoder.
NUM_LIST	4 8	List size of the polar list decoder.

Figure 3. 5G Polar Parameter Editor



IP Parameter Editor Pro - test* (/nfs/sc/disks/swuser_work_shuangho/qts_test/main_polar_0513/test.ip) <@sjlcf04

File Edit System Generate View Tools Help

Parameters

System: test Path: intel_FPGA_polar_5g_0

5G Polar Intel FPGA IP
intel_FPGA_polar_5g

Details
Generate Example Design...

Specification
Mode: (●) Decoder ( ) Encoder

Decoder Options
NUM_LIST: 4
4
8

Details Block Symbol

Parameters

Specification
Mode Select operation mode between "Decoder" or "Encoder".

Decoder Options
NUM_LIST Decoder only. List size of the successive cancellation list polar decoder.

Presets

Presets for intel_FPGA_polar_5g_0
Clear preset filters

Project
Click New... to create a preset.
Library
No presets for 5G Polar Intel FPGA IP 1.0.0

Apply Update... Delete New...

System Messages

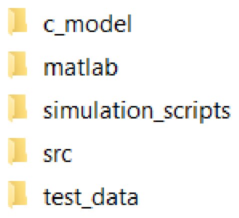
Type	Path	Message
(No messages)		

0 Errors, 0 Warnings

Generate HDL...

6. For an optional design example, click **Generate Example Design**
   The software creates a design example.

Figure 4. Design Example Directory Structure



c_model
matlab
simulation_scripts
src
test_data

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 Polar IP with the VCS Simulator

Verify that the RTL behaves the same as this models.

Before simulating, generate a 5G Polar design example.

1. Run `vcsmx_setup.sh` from *<Example Design Directory>* `\simulation_scripts\synopsys\vcsmx\`.

   ```
   >> source vcsmx_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\`.

3.3.1. Simulation Results for the 5G Polar IP

Results with the VCS Simulator

Figure 5. Encoder with old parameters

The input after a previous packet without new parameters

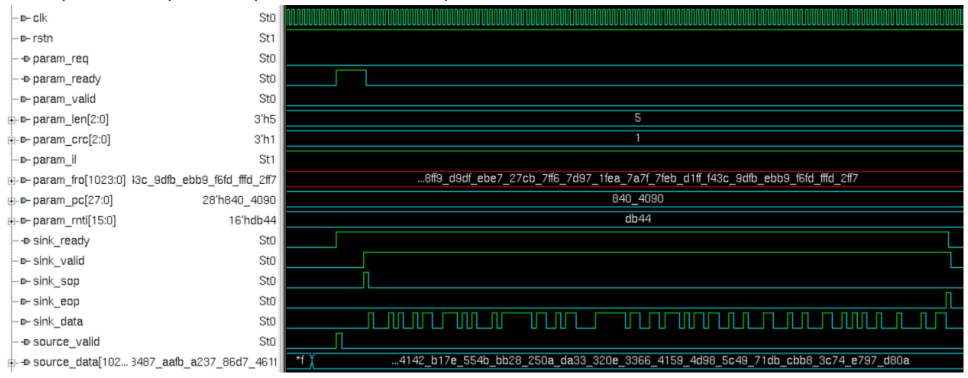

clk St0
rstn St1
param_req St0
param_ready St0
param_valid St0
param_len[2:0] 3'h5
param_crc[2:0] 3'h1
param_il St1
param_fro[1023:0] 43c_9dfb_ebb9_f6fd_fffd_2ff7
param_pc[27:0] 28'h840_4090
param_rnti[15:0] 16'hdb44
sink_ready St0
sink_valid St0
sink_sop St0
sink_eop St0
sink_data St0
source_valid St0
source_data[102... 3487_aafb_a237_86d7_461
5
1
...8ff9_d9df_ebe7_27cb_7ff6_7d97_1fea_7a7f_7feb_d1ff_f43c_9dfb_ebb9_f6fd_fffd_2ff7
840_4090
db44
*f
...4142_b17e_554b_bb28_250a_da33_320e_3366_4159_4d98_5c49_71db_cbb8_3c74_e797_d80a

Figure 6. Encoder with new parameters

The input after a previous packet with new parameters

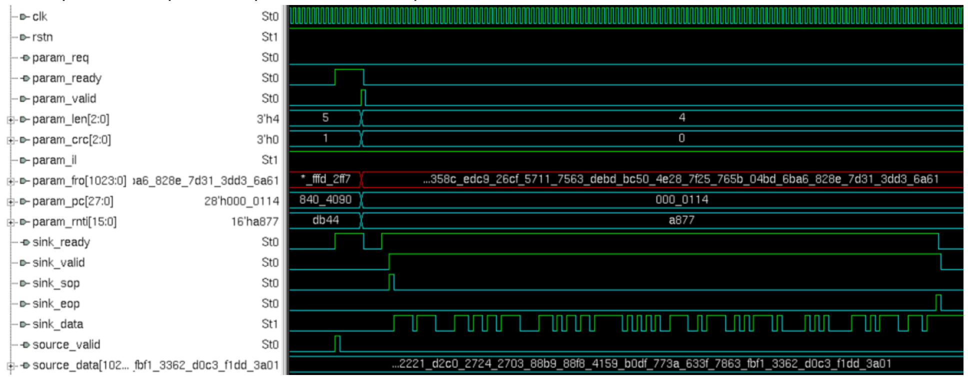

clk St0
rstn St1
param_req St0
param_ready St0
param_valid St0
param_len[2:0] 3'h4
param_crc[2:0] 3'h0
param_il St1
param_fro[1023:0] ba6_828e_7d31_3dd3_6a61
param_pc[27:0] 28'h000_0114
param_rnti[15:0] 16'ha877
sink_ready St0
sink_valid St0
sink_sop St0
sink_eop St0
sink_data St1
source_valid St0
source_data[102... fbf1_3362_d0c3_f1dd_3a01
5
4
1
0
*_fffd_2ff7
...358c_edc9_26cf_5711_7563_debd_bc50_4e28_7f25_765b_04bd_6ba6_828e_7d31_3dd3_6a61
840_4090
000_0114
db44
a877
...2221_d2c0_2724_2703_88b9_88f8_4159_b0df_773a_633f_7863_fbf1_3362_d0c3_f1dd_3a01

Figure 7. Encoder after reset

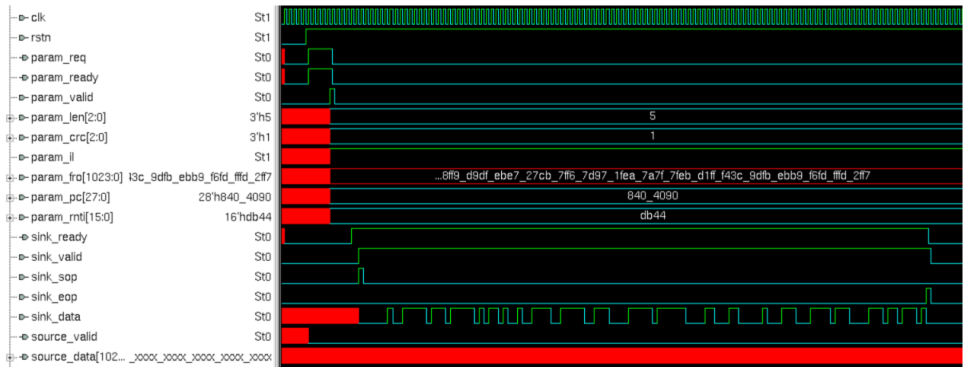


Figure 8. Decoder output

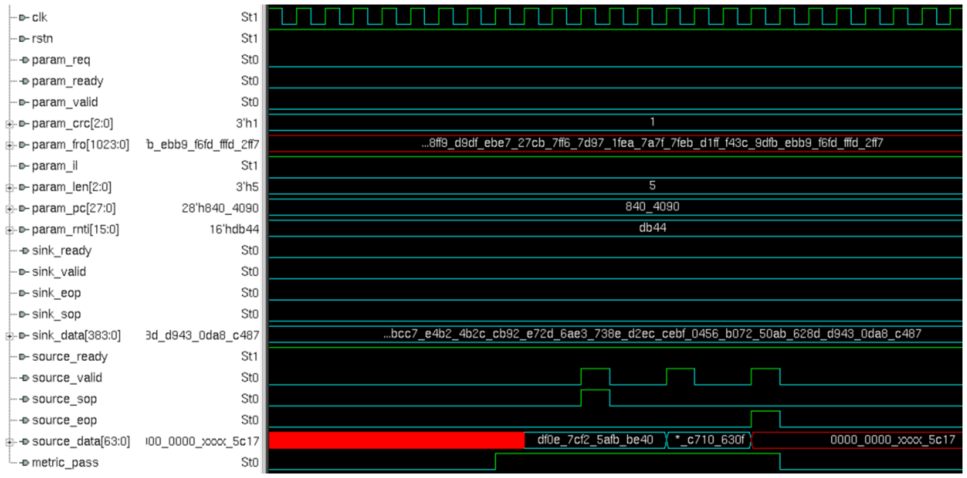


Figure 9. Decoder input with old parameters

The input after a previous packet without new parameters.

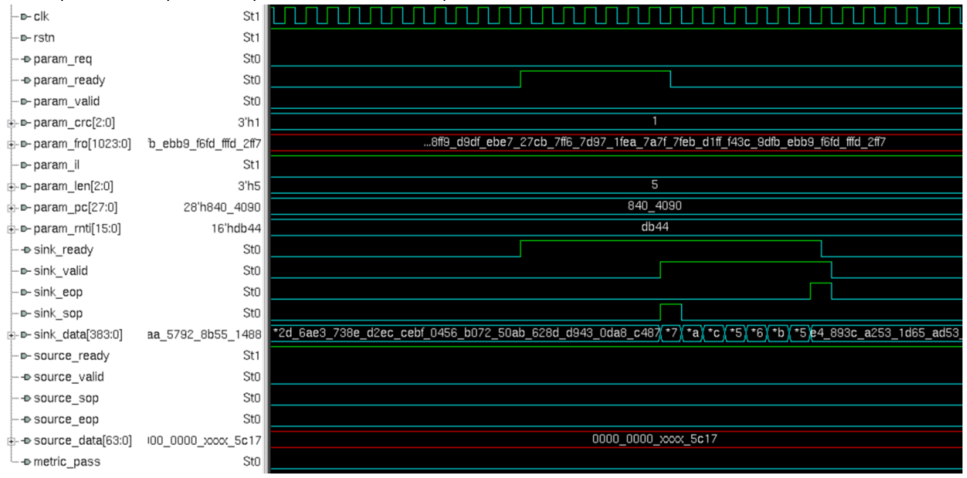


Figure 10. Decoder input with new parameters

The input after a previous packet with new parameters

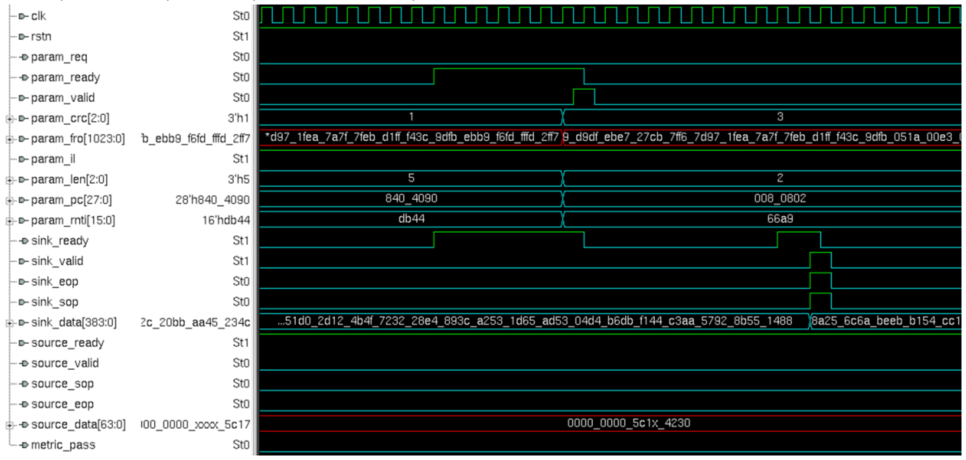
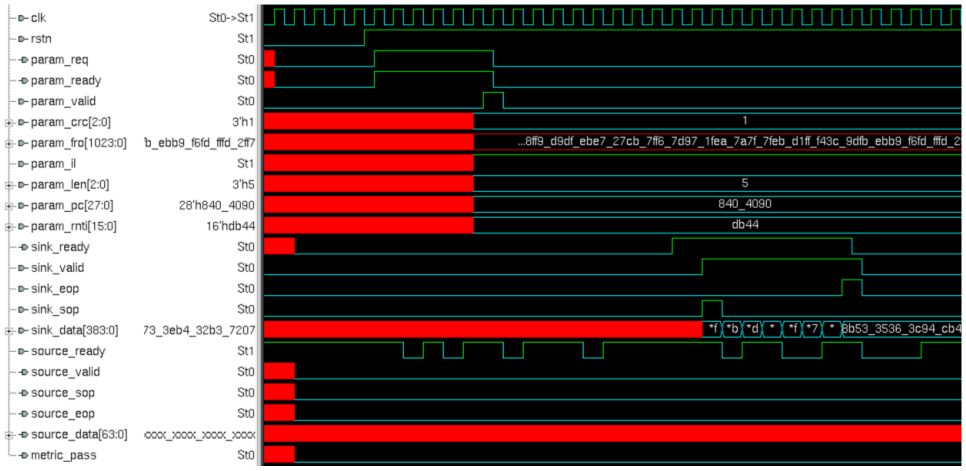


Figure 11. Decoder after reset



3.4. Simulating the 5G Polar IP with MATLAB

Verify that the RTL behaves the same as these models.

Before simulating, generate a 5G Polar design example.

1. In MATLAB, run `make.m` from the `\matlab\` directory.

   ```
   >> make
   ```

   MATLAB generates MEX.

2. Run the example test function `polar5g_codec_tb.m`.

   ```
   >>polar5g_codec_tb(<list_size>, <len_type>, <crc_type>,
   <il_on>);
   ```

where:

- `list_size` corresponds to the compile-time parameter NUM_LIST in the RTL
- `len_type` corresponds to the param_len input in the RTL
- `crc_type` corresponds to the param_crc input in the RTL
- `il_on` corresponds to the param_il input in the RTL

For example,

```
>> polar5g_codec_tb(4, 2, 4, 1);
```

This test case runs list size = 4, code block length = 64, CRC = CRC16, and the interleaver is on.

The test function generates `polar5g_codec_params.txt`, `polar5g_enc_in.txt`, `polar5g_enc_out.txt`, `polar5g_dec_in.txt`, and `polar5g_dec_out.txt`, which you may use in RTL simulation as inputs or as reference outputs.

The simulation runs both the encoder and the decoder functions, even if you generate only an encoder or a decoder.

3.5. Simulating the 5G Polar IP with the C-model

Verify that the RTL behaves the same as these models.

Before simulating, generate a 5G Polar design example.

1. Go to the `c_model\` directory.
2. Compile the C code.

   ```
   >> gcc -lm polar5g_codec_tb.c
   ```

3. Run the executable.

   ```
   >> ./a.out <list_size> <len_type> <crc_type> <il_on>
   ```

   where:

   - `list_size` corresponds to the compile-time parameter NUM_LIST in the RTL
   - `len_type` corresponds to the param_len input in the RTL
   - `crc_type` corresponds to the param_crc input in the RTL
   - `il_on` corresponds to the param_il input in the RTL

   For example,

   ```
   >> ./a out(4, 2, 4, 1);
   ```

   This test case runs list size = 4, code block length = 64, CRC = CRC16, and the interleaver is on.

   The test function generates `polar5g_codec_params.txt`, `polar5g_enc_in.txt`, `polar5g_enc_out.txt`, `polar5g_dec_in.txt`, and `polar5g_dec_out.txt`, which you may use in RTL simulation as inputs or as reference outputs.

   The simulation runs both the encoder and the decoder functions, even if you generate only an encoder or a decoder.

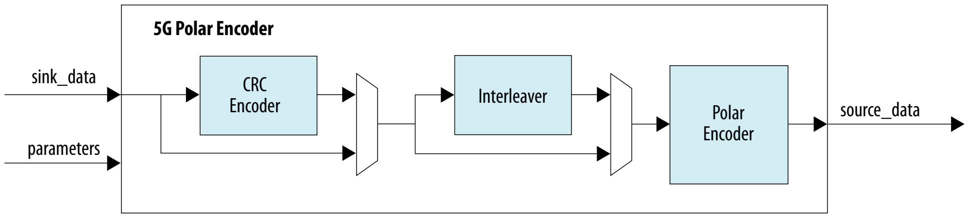
4. 5G Polar Intel FPGA IP Functional Description

The IP includes an encoder and decoder.

The IP complies with the following parts of *3GPP 5G NR specs TS38.212* and *TS38.214*:

- The input length to the encoder, the *number_of_message_bits*, agree with the code block length, CRC type, frozen bit locations, and parity bit locations. Where the number_of_message_bits = *number_of_information_bits* (non-frozen) – *number_of_parity_bits* – *number_of_CRC_bits*.
- *number_of_message_bits* is at least 12.
- The input length to the decoder agrees with the code block length.
- The first information bit (non-frozen bit) is not a parity check bit.
- If the interleaver is on, the *number_of_interleaved_bits* is no less than 31 and no more than 164. Where the *number_of_interleaved_bits* = *number_of_information_bits* (non-frozen) – *number_of_parity_bits*.
- When the code block length == 32,
  - You cannot turn on the interleaver
  - You cannot select CRC16, CRC24a, CRC24b, CRC24c (whether in DCI format or not)

Figure 12. 5G Polar Encoder



The encoder comprises a CRC encoder, an interleaver with a frozen bit and parity check bit insertion block, and a polar encoder. The encoder accepts the input message data and optionally appends the CRC bits based on the CRC type. Then the interleaver optionally shuffles the result and the encoder encodes the message.

Figure 13. Input to the Polar Encoder

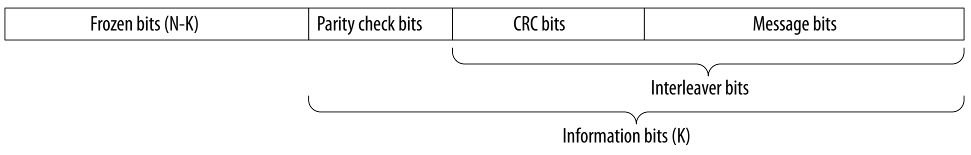
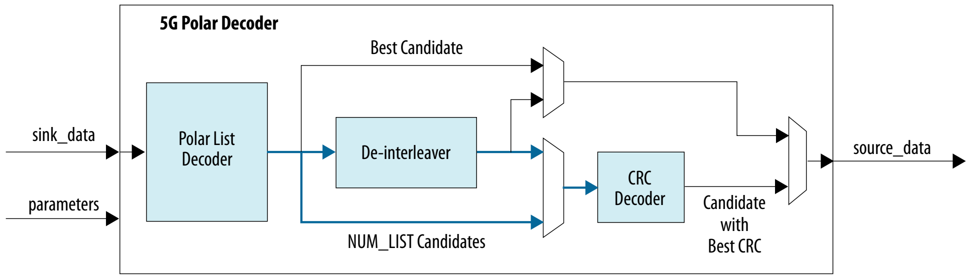


Figure 14. 5G Polar Decoder



The decoder comprises a polar list decoder (list size of 4 or 8), a deinterleaver, and a CRC decoder. The polar list decoder is based on the successive cancellation decoding algorithm. The compile-time parameter **NUM_LIST** sets the list size. The list decoder parallelizes **NUM_LIST** computations of each decoding step of the algorithm. It generates 2***NUM_LIST** temporary candidates. It performs a sorting to keep the better **NUM_LIST** candidates to proceed to the next decoding step, until it finishes all the decoding steps. The polar list decoder first decodes the received messages and provides the total number of **NUM_LIST** decoded candidates, and it provides the index of the best candidate. The deinterleaver optionally reverts the encoder interleaving for all candidates. If you turn off the CRC, the IP produces the best candidate as determined by the polar list decoder. If you turn on the CRC, all candidates go through the CRC decoder so that the IP produces the candidate with the best CRC decoding. The result contains CRC bits.

4.1. 5G Polar IP Signals

All signals are synchronous to `clk`.

Encoder

Table 6. Polar FEC 5G Encoder Signals

Signal	Width	Direction	Description
`Clk`	1	Input	Clock.
`rstn`	1	Input	Active-low synchronous reset.
`param_req`	1	Output	Request input parameters. The IP asserts this signal after reset. You should provide input parameters when the IP asserts this signal. The IP does not accept input data until you provide input parameters.
`param_ready`	1	Output	Indicates that the IP is ready to accept input parameters. Ready latency is 1 clock cycle. If this signal is asserted, the IP takes the valid signal (`param_valid`) in the next clock cycle. The IP asserts this signal before each code block. The IP asserts this signal when the IP asserts `param_req`. If the IP asserts this signal but not `param_req`, you can optionally provide input parameters. If you do not provide input parameters, the IP keeps the previous parameters.
`param_valid`	1	Input	Assert when incoming packet parameter signals are valid. Assert this signal for one clock cycle when `param_ready` is asserted. If asserted for more than one cycle, the IP takes only the parameters at the first cycle.

continued...

Signal	Width	Direction	Description
`param_len`	3	Input	1:N=32 2:N=64 3:N=128 4:N=256 5:N=512 6:N=1024
`param_crc`	3	Input	0:CRC OFF 1:CRC24a 2:CRC24b 3:CRC24c 4:CRC16 5:CRC11 6:CRC6 7:CRC24c in DCI format
`param_il`	1	Input	0:deinterleaving off 1:deinterleaving on
`param_fro`	1024	Input	Indicates frozen bit location. `param_fro[i]==1` indicates bit i is a frozen bit. `param_fro[1023:N]` are ignored when N<1024.
`param_pc`	28	Input	Indicates parity check bit location. `param_pc[i]==1` indicates the i-th information bit is a parity check bit. `param_pc[j]` is ignored if the number of information bits is less than `j`. `param_pc[0]` should always be zero. k-th information bit is never a parity check bit for all k>=28.
`param_rnti`	16	Input	Indicates the radio network temporary identifier (RNTI) bits. When you do not select CRC type as CRC24c in DCI format (3'd7), the IP ignores the RNTI.
`sink_ready`	1	Output	Indicates that the IP can take an input packet. Ready latency is 1 clock cycle. If the IP asserts this signal, the IP takes the valid signal (`sink_valid`) in the next clock cycle. When asserted, this signal keeps asserted until the end of the input packet.
`sink_valid`	1	Input	Assert when incoming packet data signal is valid. The IP accepts this signal only when the IP asserts `sink_ready` in the previous clock cycle. When this signal is low, the IP ignores `sink_sop, eop, and data`. Do not assert this signal at the next clock cycle of a valid EOP, as the IP ignores it. The IP does not accept SOP immediately after EOP. Do not assert this signal at the next clock cycle of `param_valid`, as the IP ignores it. The IP does not accept SOP immediatley after a new parameter setup. If the IP asserts both `param_ready` and `sink_ready` on the previous clock cycle and you assert both `param_valid` and `sink_valid` on the current clock cycle, the IP only accepts `param_valid` and ignores `sink_valid`.
`sink_sop`	1	Input	Indicates the start of an incoming packet. The IP starts to accept incoming packets when you assert this signal. Assert this signal with the first input data.
`sink_eop`	1	Input	Indicates the end of an incoming packet. Assert this signal with the last input data.

continued...

Signal	Width	Direction	Description
sink_data	1	Input	Frame data input.
source_valid	1	Output	The IP asserts source_valid when dumping an output packet, indicating the outputs are valid.
source_data	1024	Output	Encoded data output. The IP sends an N-bit codeword at source_data[N-1:0].

Figure 15. Encoder timing diagram of packet input after reset

Input length =100. Gray shading represents don't care.

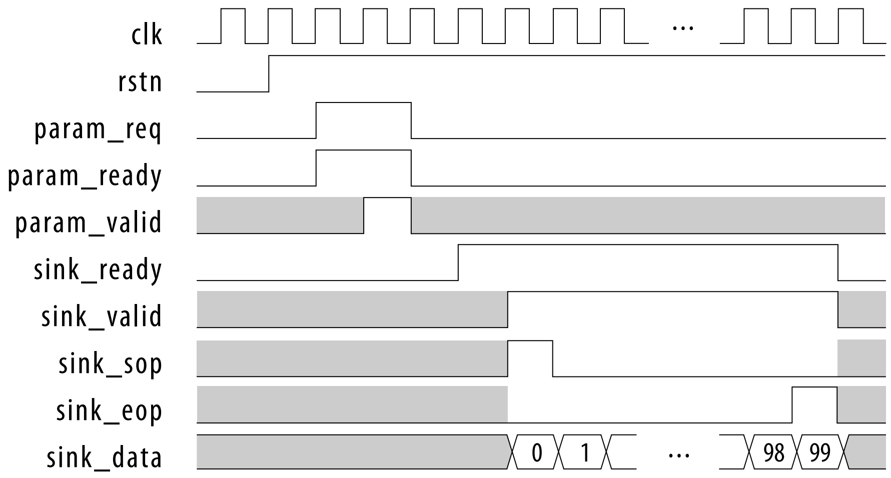
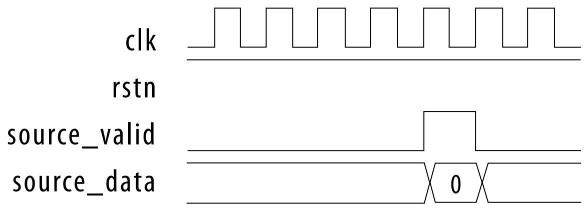


Figure 16. Encoder timing diagram of packet output



Decoder

Table 7. Polar FEC 5G Decoder Signals

Signal	Width	Direction	Description
Clk	1	Input	Clock.
rstn	1	Input	Active-low synchronous reset.
param_req	1	Output	Request input parameters. The IP asserts this signal after reset. You should provide input parameters when the IP asserts this signal. The IP does not accept input data until you provide input parameters.
param_ready	1	Output	The IP can accept input parameters. Ready latency is 1 clock cycle. If the IP asserts this signal, the IP accepts the valid signal (param_valid) in the next clock cycle. The IP asserts this signal before each code block.

continued...

Signal	Width	Direction	Description
			The IP asserts this signal when it asserts param_req. If the IP asserts this signal but not param_req, you can optionally provide input parameters. If you provide no input parameters, the IP keeps the previous parameters.
param_valid	1	Input	Assert when incoming packet parameter signals are valid. Assert this signal for one clock cycle when the IP asserts param_ready. If asserted for more than one cycle, the IP accepts only the parameters at the first cycle.
param_len	3	Input	1:N=32 2:N=64 3:N=128 4:N=256 5:N=512 6:N=1024
param_crc	3	Input	0:CRC OFF 1:CRC24a 2:CRC24b 3:CRC24c 4:CRC16 5:CRC11 6:CRC6 7:CRC24c in DCI format
param_il	1	Input	0:deinterleaving off 1:deinterleaving on
param_fro	1024	Input	Indicate frozen bit location. param_fro[i]==1 indicates bit i is a frozen bit. param_fro[1023:N] are ignored when N<1024.
param_pc	28	Input	Indicate parity check bit location. param_pc[i]==1 indicates the i-th information bit is a parity check bit. param_pc[j] is ignored if the number of information bits is less than j. param_pc[0] should always be set to zero. k-th information bit is never a parity check bit for all k>=28.
param_rnti	16	Input	Indicates the RNTI bits. When you do not select CRC type as CRC24c in DCI format (3'd7), the IP ignores the RNTI.
sink_ready	1	Output	Indicates that the IP can accept an input packet. The ready latency is 1 clock cycle. If the IP asserts this signal, the IP accepts the valid signal (sink_valid) in the next clock cycle. When asserted, this signal keeps asserted until the end of the input packet.
sink_valid	1	Input	Assert when incoming packet data signal is valid. The IP accepts this signal only when the IP asserts sink_ready in the previous clock cycle. When this signal is low, the IP ignores sink_sop/eop/data. Do not assert this signal at the next clock cycle of a valid EOP, as the IP ignores it. The IP does not accept SOP immediately after EOP. Do not assert this signal at the next clock cycle of param_valid, as the IP ignores it. The IP does not accept SOP immediatley after a new parameter setup. If the IP asserts both param_ready and sink_ready on the previous clock cycle and you assert both param_valid are sink_valid on the current clock cycle, the IP only acepts param_valid and ignores sink_valid.

continued...

Signal	Width	Direction	Description
sink_sop	1	Input	Indicates the start of an incoming packet. The IP starts to accept incoming packet when you assert this signal. Assert this signal for one cycle only.
sink_eop	1	Input	Indicates the end of an incoming packet. Assert this signal with the last input data. Assert this signal for one cycle only.
sink_data	64*LLR_BIT (where LLR_BIT = 6)	Input	Frame data input. Each LLR is LLR_BIT-bit long. You should send 64 LLRs in each cycle. Each packet contains N LLRs of a frame, taking N/64 cycles. Send LLR[63]-LLR[0] in the first cycle; LLR[127]-LLR[64] in the second cycle, etc. When N=32, send LLR[31]-LLR[0] in sink_data[32*LLR_BIT-1:0]. 2’s compliment format, but exclude the extreme negative values. LLR ranges from $-(2^{LLR\_BIT-1}-1)$ to $+(2^{LLR\_BIT-1}-1)$. Saturate the value -2^{LLR_BIT-1} to $-(2^{LLR\_BIT-1}-1)$ before providing to the IP. Refer to Table 8 on page 22
source_ready	1	Input	Assert when you can accept an outgoing packet. The ready latency is 1 clock cycle. When you assert this signal, the IP provides the valid signal (source_valid) in the next clock cycle, if output data is available from the IP.
source_valid	1	Output	Asserted when dumping an output packet, indicating the outputs are valid. The IP can only assert this signal if it receives source_ready in the previous clock cycle.
source_sop	1	Output	Indicates the start of an output packet. After decoding finishes and when the downstream is ready, the IP asserts this signal when it provides the first cycle of the output data.
source_eop	1	Output	Indicates the end of an output packet. The IP asserts this signal when it provides the last cycle of the output data.
source_data	64	Output	Decoded data output. The IP sends out *M* bits out, where *M* = number of interleaved bits, *M* = *K* - number of parity check bits, and *M* = number of message bits + number of CRC bits. For more information, refer to the *Inputs to the Encoder* table in *5G Polar Functional Description*. The IP sends CRC bits and does not send parity check bits. The IP sends out[63:0] in the same cycle as source_sop. The IP sends out[127:64] in the cycle after. If *M* does not divide 64, the IP sends out[*M*-1: *M*-(*M* mod 64)] in the same cycle as source_eop, located at source_data[(M mod 64)-1:0]. An output packet takes ceil(M/64) cycles. When *M*<64, out[M-1:0] are located at source_data[M-1:0].
metric_pass	1	Output	Indicates CRC check result. 0: CRC is OFF or CRC check failed 1: CRC check passed. This value is valid from source_sop to source_eop and the IP does not change it during this period

Table 8. LLR Meanings

Final hard decision of an LLR is the sign bit (MSB) of the LLR. The hard decision of 0b000000 is 0. Do not feed -32 as an LLR, saturate it to -31.

Binary	Decimal	Meaning	Binary	Decimal	Meaning
011111	+31	Strongest 0	100001	-31	Strongest 1
000001	+1	Weakest 0	111111	-1	Weakest 1
000000	0	Unknown	100000	-32	Invalid

Figure 17. Decoder timing diagram of packet input after reset

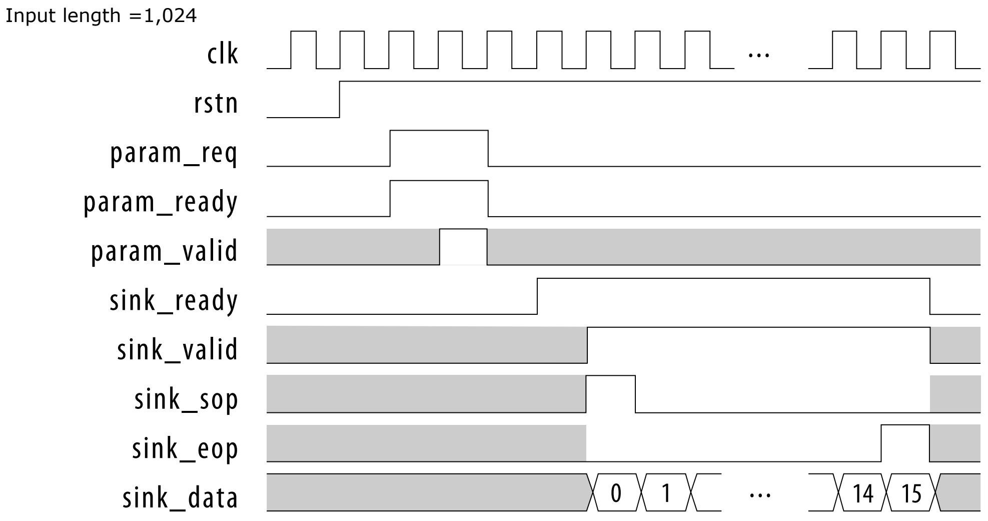
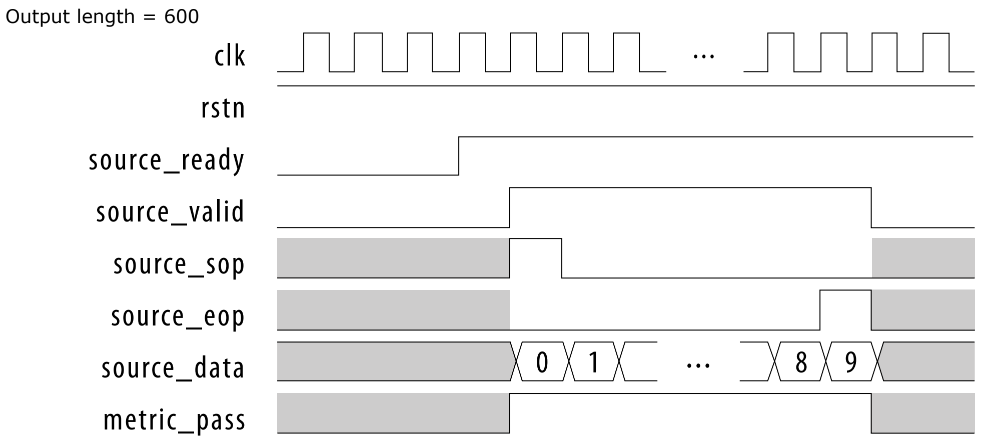


Figure 18. Decoder timing diagram of packet output



Encoder and Decoder Timing Diagrams

Figure 19. Timing diagram of packet input after a previous packet

With a new set of input parameters for the next packet

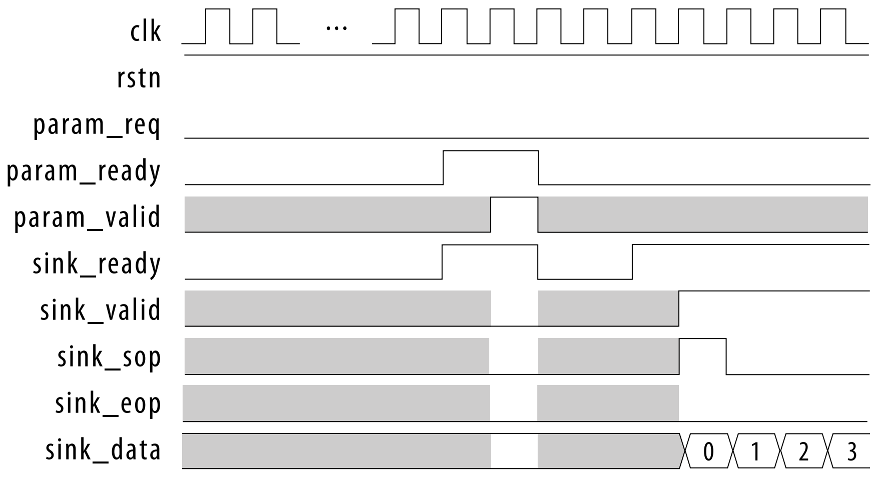
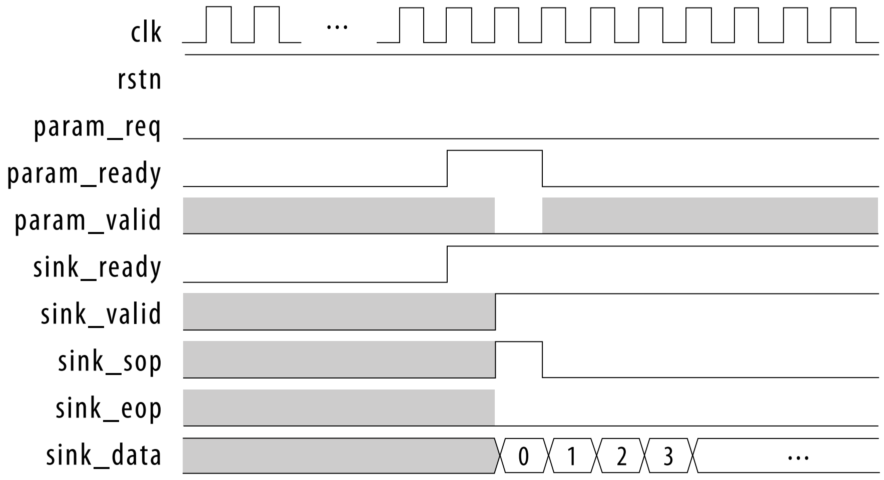


Figure 20. Timing diagram of packet input after a previous packet

Without a new set of input parameters for the next packet



Related Information

5G Polar Intel FPGA IP Functional Description on page 16

4.2. 5G Polar IP Throughput and Latency

Throughput and latency can vary for different input frames, different input LLR values, different locations of frozen bits and parity check bits, and the number of frozen bits that determines code rate. The latency improves by about 5% when you turn off CRC and interleaving, but the throughput remains the same. The **NUM_LIST** = 8 decoder has approximately 10% longer latency and lower throughput compared to the **NUM_LIST** = 4 decoder.

Figure 21. Encoder latency and throughput

Clock frequency = 370MHz. CRC and interleaving off.

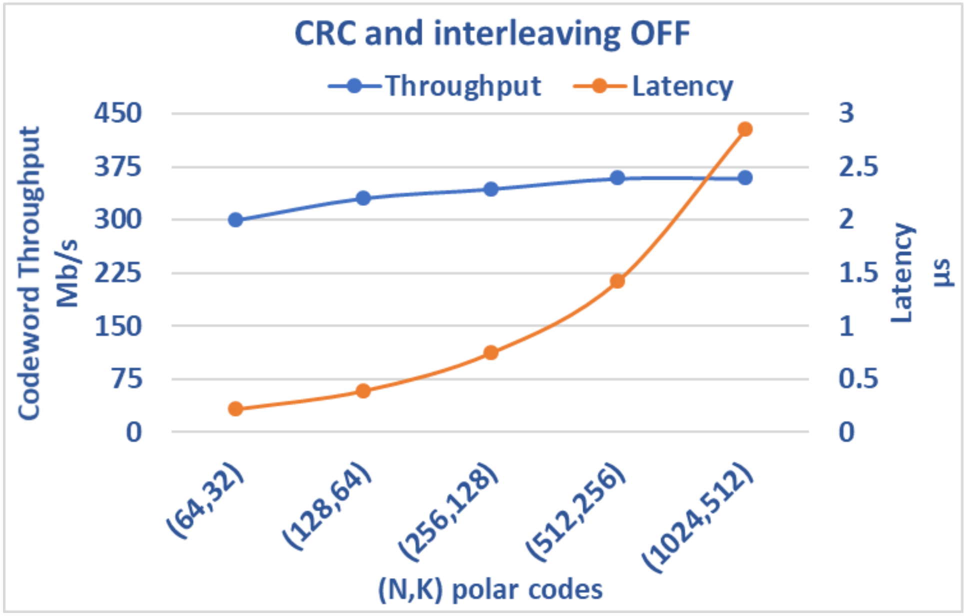
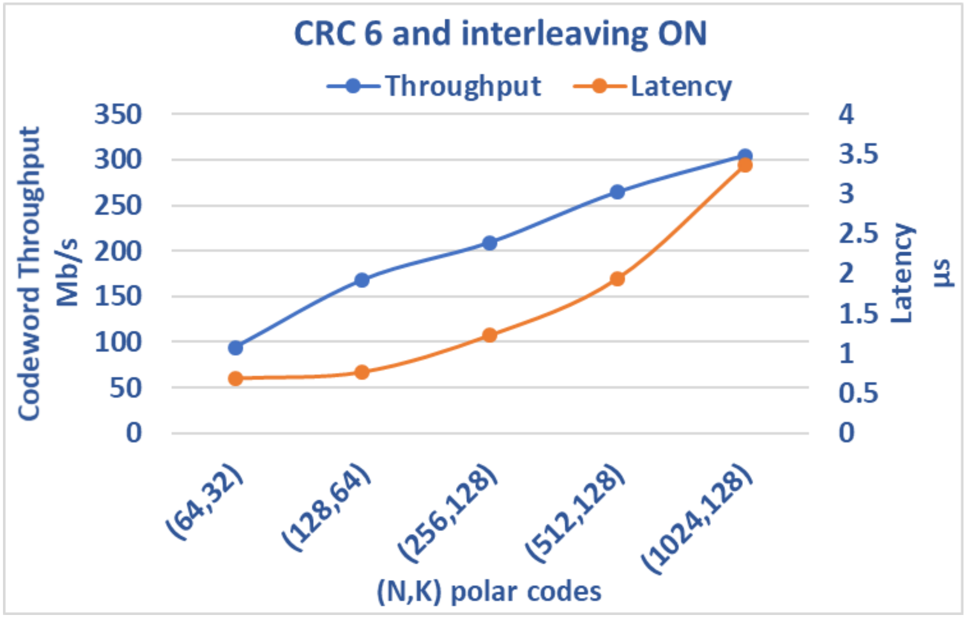


Figure 22. Encoder latency and throughput

CRC = 6 and interleaving on



For the decoder figures, Rc is the code rate, which equals to number_of_information_bits (K) / code_block_length (N).

Figure 23. Decoder latency

Clock frequency = 370MHz. **NUM_LIST** = 4

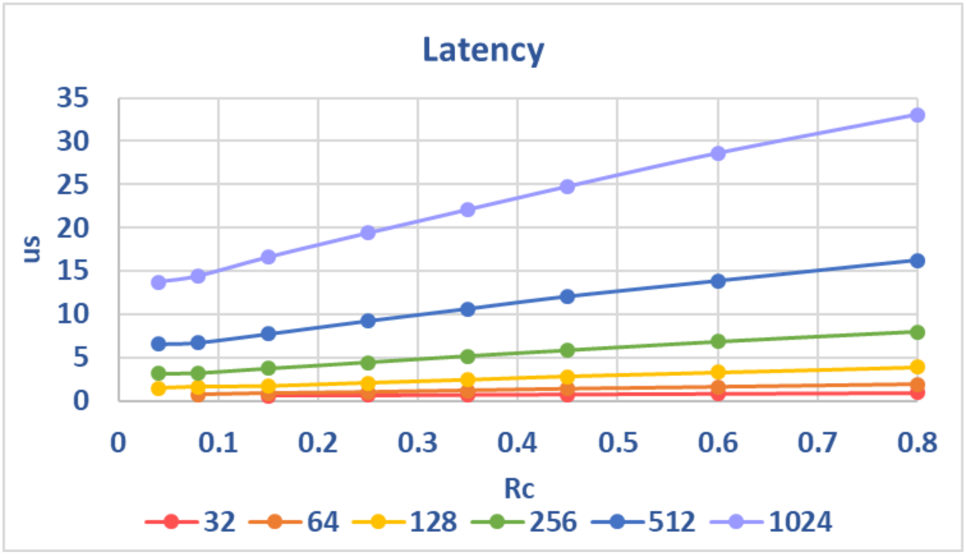


Figure 24. Decoder throughput (codeword bit)

Clock frequency = 370MHz. **NUM_LIST** = 4

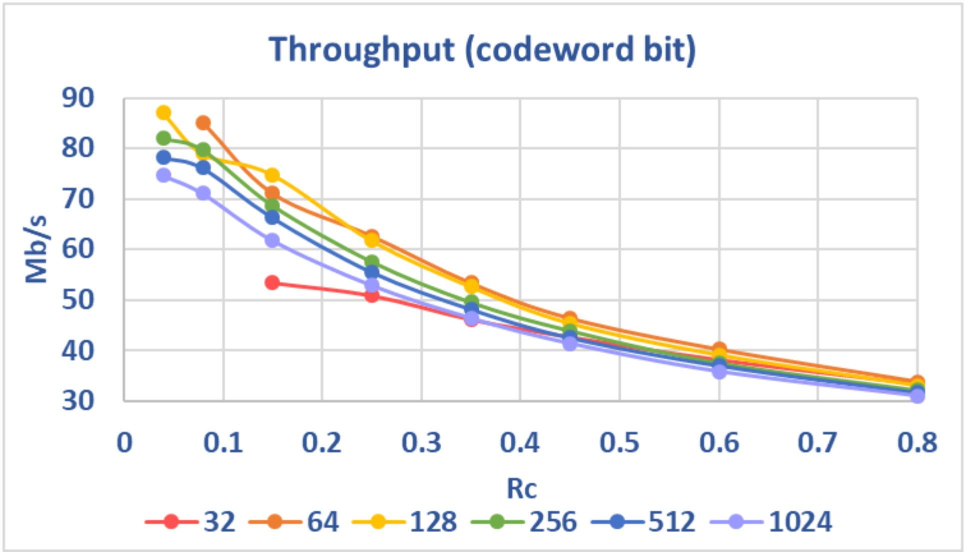
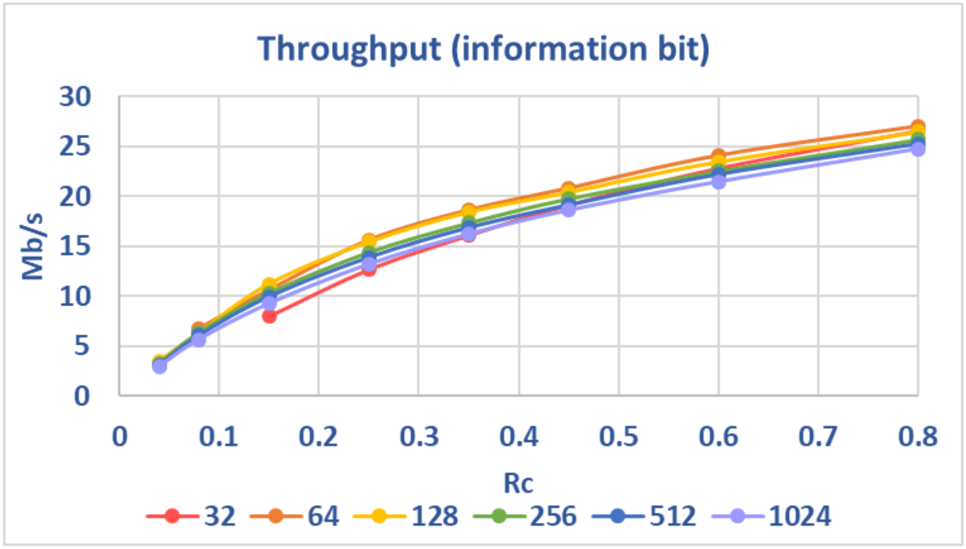


Figure 25. Decoder throughput (information bit)

Clock frequency = 370MHz. **NUM_LIST** = 4



5. 5G Polar IP User Guide Archive

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

IP Core Version	User Guide
1.0.0	5G Polar IP User Guide

6. Document Revision History for the 5G Polar Intel FPGA IP User Guide

Date	IP Version	Intel Quartus Prime Software Version	Changes
2019.10.15	1.0.0	20.3	Initial release.