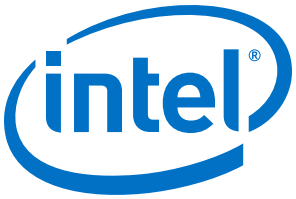


High-speed Reed-Solomon IP Core User Guide

UG-01166
2016.11.02


Subscribe


Send Feedback

Contents

1 About the High-speed Reed-Solomon IP Core

The Altera High-speed Reed-Solomon IP Core uses a highly parallel architecture for large applications that require throughput of 100 Gbps and greater. The IP core is suitable for 10G (such as optical transport networks (OTN)) or 100G Ethernet (IEEE 802.3bj/bm) applications.

Related Links

- Introduction to Intel FPGA IP Cores
  Provides general information about all Intel FPGA IP cores, including parameterizing, generating, upgrading, and simulating IP cores.
- Creating Version-Independent IP and Qsys Simulation Scripts
  Create simulation scripts that do not require manual updates for software or IP version upgrades.
- Project Management Best Practices
  Guidelines for efficient management and portability of your project and IP files.
- Reed-Solomon II IP Core User Guide
  The Reed-Solomon II IP core is highly parameterizable for low throughput applications.

1.1 High-speed Reed-Solomon IP Core Features

- High-performance greater than 100 Gbps encoder or decoder for error detection and correction:
- — Fully parameterizable:
  - — Number of bits per symbol
  - — Number of symbols per codeword
  - — Number of check symbols per codeword
  - — Field polynomial
- Multichannels and backpressure for decoders
- Fracturable decoder that supports 100 Gbps Ethernet (GbE), 2 x 50 GbE, and 4 x 25 GbE
- Avalon® Streaming (Avalon-ST) interfaces
- Testbenches to verify the IP core
- IP functional simulation models for use in Intel-supported VHDL and Verilog HDL simulators

1.2 DSP IP Core Device Family Support

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

- Advance support—the IP core 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—Intelverifies the IP core with final timing models for this device family. The IP core meets all functional and timing requirements for the device family. You can use it in production designs.

Table 1. DSP IP Core Device Family Support

Device Family	Support
Arria® II GX	Final
Arria II GZ	Final
Arria V	Final
Arria 10	Final
Cyclone® IV	Final
Cyclone V	Final
MAX® 10 FPGA	Final
Stratix® IV GT	Final
Stratix IV GX/E	Final
Stratix V	Final
Stratix 10	Advance
Other device families	No support

1.3 DSP IP Core Verification

Before releasing a version of an IP core, Intel runs comprehensive regression tests to verify its quality and correctness. Intel generates custom variations of the IP core to exercise the various parameter options and thoroughly simulates the resulting simulation models with the results verified against master simulation models.

1.4 High-speed Reed-Solomon IP Core Release Information

Use the release information when licensing the IP core.

Table 2. Release Information

Item	Description
Version	16.1
Release Date	November 2016
Ordering Code	IP-RSCODEC-HS (IPR-RSCODEC-HS)
Product ID	00FC (encoder and decoder) ???? (Fractuarable encoder and decoder)
Vendor ID	6AF7

Intel verifies that the current version of the Quartus Prime software compiles the previous version of each IP core. Intel does not verify that the Quartus Prime software compiles IP core versions older than the previous version. The *Intel FPGA IP Release Notes* lists any exceptions.

Related Links

- Intel FPGA IP Release Notes
- Errata for Reed-Solomon IP core in the Knowledge Base

1.5 High-speed Reed-Solomon IP Core Performance and Resource Utilization

Decoder

Table 3. Performance and Resource Utilization for Arria 10 Devices

Typical expected performance using the Quartus Prime software with with Arria 10 (10AX115R4F40I3SG) devices for RS(*n*,*k*) where *n* is the codeword length and *k* the number of information symbols. The M20K numbers in brackets indicate the M20K usage when you turn on **True-dual port ROM**.

Parameters				ALM	Memory M20K	fMAX (MHz)
RS Code	Parallelism (P)	Latency	Favor ROM			
(255,239)	15	91	0	5092	3	351
			1	4545	20 (12)	335
		132	0	5160	3	444
			1	4583	20 (12)	439
(528,514)	32	87	0	15,127	8	412
			1	8,418	44 (25)	361
		115	0	15,053	8	429
			1	8,467	44 (25)	431
(528,514)	16	99	0	8282	4	377
			1	4665	21 (13)	374
		127	0	8225	4	429
			1	4712	21 (13)	448

continued...

Parameters				ALM	Memory M20K	fMAX (MHz)
RS Code	Parallelism (P)	Latency	Favor ROM			
(528,514)	8	132	0	5192	2	352
			1	3313	11 (4)	375
		160	0	5174	2	473
			1	3273	11 (4)	461
(544,514)	136	134	0	96,606	34	321
			1	70257	185 (110)	317
		194	0	95,982	34	340
			1	70527	185 (110)	330
(544,514)	34	151	0	28335	9	345
			1	21058	45 (28)	330
		211	0	28044	9	394
			1	21018	45 (28)	380

Table 4. Performance and Resource Utilization for Arria 10 Devices (Fracturable IP Core)

Typical expected performance using the Quartus Prime software with with Arria 10 (10AX115R4F40I3SG) and (10AX115R2F40I1SG) devices for RS(*n*,*k*) where *n* is the codeword length and *k* the number of information symbols.

Parameters		ALM	Memory M20K	fMAX (MHz)	
RS Code	Latency			I1	I3
(528,514)	112	10,834	24	401	336
	128	10,910	24	460	384
	164	10,812	25	440	363
	196	11,029	25	451	396

Table 5. Performance and Resource Utilization for Stratix V Devices

Typical expected performance using the Quartus Prime software with Stratix V (5SGXEA7H3F35C3) devices for RS(*n*,*k*) where *n* is the codeword length and *k* the number of information symbols.. The M20K numbers in brackets indicate the M20K usage when you turn on **True-dual port ROM**.

Parameters				ALM	Memory M20K	fMAX (MHz)
RS Code	Parallelism (P)	Latency	Favor ROM			
(255,239)	15	91	0	4894	3	351
			1	4426	20	335
		123	0	5077	3	444
			1	4354	20	439
(528,514)	32	87	0	14,948	8	390
			1	8,418	44 (25)	361
		115	0	14,916	8	437

continued...

Parameters				ALM	Memory M20K	fMAX (MHz)
RS Code	Parallelism (P)	Latency	Favor ROM			
			1	7,735	44 (25)	394
(528,514)	16	99	0	8126	4	377
			1	4493	21 (13)	374
		127	0	8060	4	429
			1	4523	21 (13)	448
(528,514)	8	132	0	5174	2	352
			1	3303	11 (4)	375
		160	0	5191	2	473
			1	3244	11 (4)	461
(544,514)	32	146	0	27090	8	286
			1	19801	44	280
		206	0	26724	8	365
			1	19463	44	357

Encoder

Table 6. Performance and Resource Utilization for Arria 10 Devices

Typical expected performance using the Quartus Prime software with with Arria 10 (10AX115R4F40I3SG) devices.

Parameters		ALM	fMAX (MHz)
RS Code	Parallelism (P)		
(255,239)	15	1,500	541
(528,514)	132	12,346	327
	33	4,003	430
	16	2,666	484
	8	2,104	498
(544,514)	136	25,750	309
	32	9,824	381

Table 7. Performance and Resource Utilization for Stratix V Devices

Typical expected performance using the Quartus Prime software with Stratix V (5SGXEA7H3F35C3) devices.

Parameters		ALM	fMAX (MHz)
RS Code	Parallelism (P)		
(255,239)	15	2,095	546
(528,514)	132	11677	307
	33	3917	416
	16	2633	473

continued...

Parameters		ALM	fMAX (MHz)
RS Code	Parallelism (P)		
	8	2004	462
(544,514)	136	23819	306
	32	9343	369

2 High-speed Reed-Solomon IP Core Getting Started

2.1 Installing and Licensing IP Cores

The Quartus® Prime software installation includes the Intel FPGA IP library. This library provides useful IP core functions for your production use without the need for an additional license. Some MegaCore® IP functions in the library require that you purchase a separate license for production use. The OpenCore® feature allows evaluation of any Intel FPGA IP core in simulation and compilation in the Quartus Prime software. Upon satisfaction with functionality and performance, visit the Self Service Licensing Center to obtain a license number for any Intel FPGA product.

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

Figure 1. IP Core Installation Path

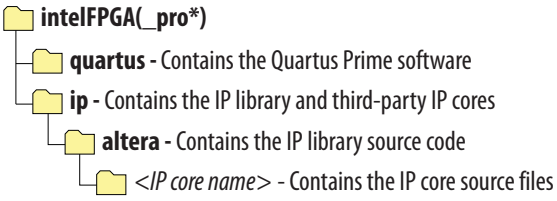


Table 8. IP Core Installation Locations

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

2.1.1 OpenCore Plus IP Evaluation

The free OpenCore Plus feature allows you to evaluate licensed MegaCore IP cores in simulation and hardware before purchase. Purchase a license for MegaCore IP cores if you decide to take your design to production. OpenCore Plus supports the following evaluations:

- Simulate the behavior of a licensed 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.

OpenCore Plus evaluation supports the following two operation modes:

- Untethered—run the design containing the licensed IP for a limited time.
- Tethered—run the design containing the licensed IP for a longer time or indefinitely. This operation requires a connection between your board and the host computer.

Note: All IP cores that use OpenCore Plus time out simultaneously when any IP core in the design times out.

Related Links

- Quartus Prime Licensing Site
- Quartus Prime Installation and Licensing

2.1.2 High-speed Reed-Solomon IP Core OpenCore Plus Timeout Behavior

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

All IP cores in a device time out simultaneously when the most restrictive evaluation time is reached. If there is more than one IP core in a design, a specific IP core's time-out behavior may be masked by the time-out behavior of the other IP cores. For IP cores, 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 OpenCore Plus Files (`.ocp`) in your project directory to identify your use of the OpenCore Plus evaluation program. After you activate the feature, do not delete these files..

When the evaluation time expires, `out_data` goes low .

Related Links

AN 320: OpenCore Plus Evaluation of Megafunctions

2.2 IP Catalog and Parameter Editor

The IP Catalog displays the IP cores available for your project. Use the following features of the IP Catalog to locate and customize an IP core:

- Filter IP Catalog to **Show IP for active device family** or **Show IP for all device families**. If you have no project open, select the **Device Family** in IP Catalog.
- Type in the Search field to locate any full or partial IP core name in IP Catalog.
- Right-click an IP core name in IP Catalog to display details about supported devices, to open the IP core's installation folder, and for links to IP documentation.
- Click **Search for Partner IP** to access partner IP information on the web.

The parameter editor prompts you to specify an IP variation name, optional ports, and output file generation options. The parameter editor generates a top-level Quartus Prime IP file (`.ip`) for an IP variation in Quartus Prime Pro Edition projects.

The parameter editor generates a top-level Quartus IP file (`.qip`) for an IP variation in Quartus Prime Standard Edition projects. These files represent the IP variation in the project, and store parameterization information.

Figure 2. IP Parameter Editor (Quartus Prime Pro Edition)

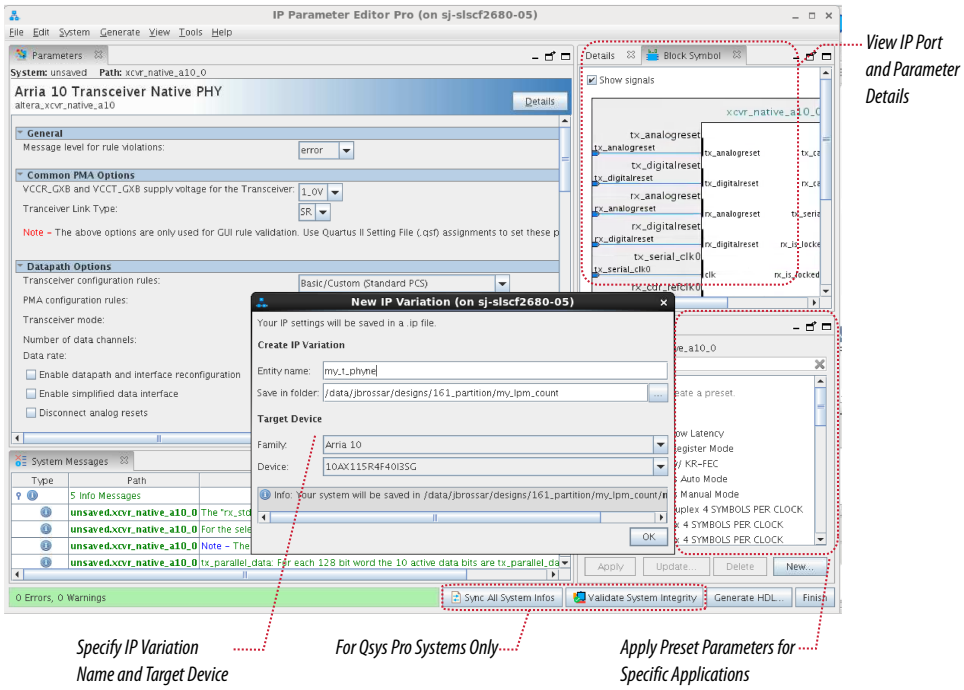
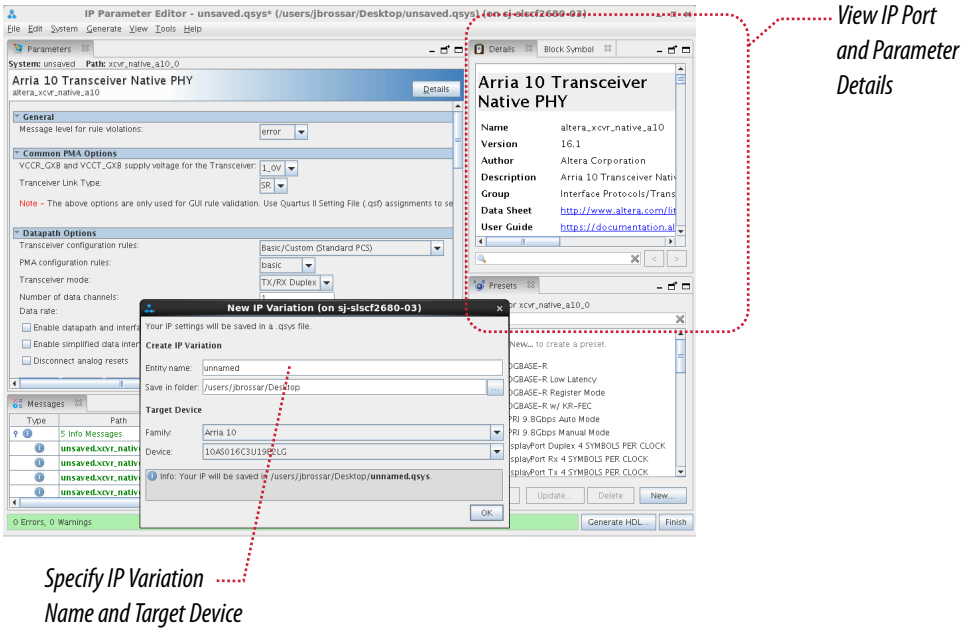


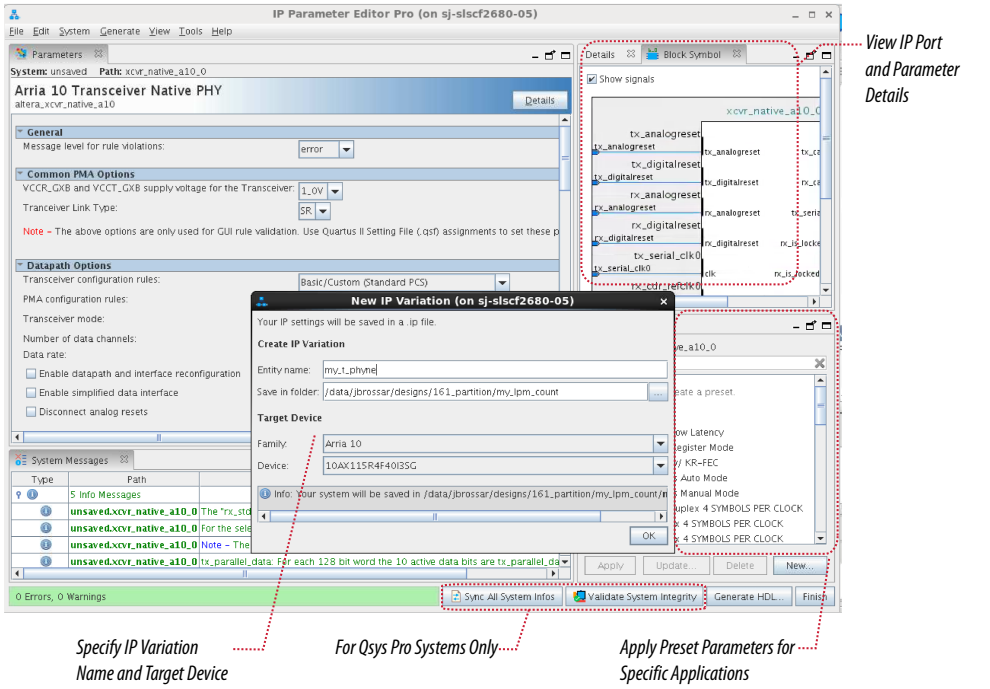
Figure 3. IP Parameter Editor (Quartus Prime Standard Edition)



2.3 Generating IP Cores (Quartus Prime Pro Edition)

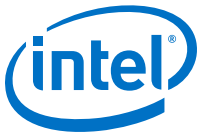
Configure a custom IP variation in the parameter editor.
Double-click any component in the IP Catalog to launch the parameter editor. The parameter editor allows you to define a custom variation of the selected IP core. The parameter editor generates the IP variation and adds the corresponding `.ip` file to your project automatically.

Figure 4. IP Parameter Editor (Quartus Prime Pro Edition)



Follow these steps to locate, instantiate, and customize an IP variation in the parameter editor:

1. Click **Tools ➤ IP Catalog**. To display details about device support, installation location, versions, and links to documentation, right-click any IP component name in the IP Catalog.
2. To locate a specific type of component, type some or all of the component's name in the IP Catalog search box. For example, type `memory` to locate memory IP components, or `axi` to locate IP components with AXI in the IP name. Apply filters to the IP Catalog display from the right-click menu.
3. To launch the parameter editor, double-click any component. Specify a top-level name for your custom IP variation. The parameter editor saves the IP variation settings in a file named <*your_ip*>`.ip`. Click **OK**. Do not include spaces in IP variation names or paths.
4. Set the parameter values in the parameter editor and view the block diagram for the component. The **Parameterization Messages** tab at the bottom displays any errors in IP parameters:



- Optionally select preset parameter values if provided for your IP core. Presets specify initial parameter values for specific applications.
- Specify parameters defining the IP core functionality, port configurations, and device-specific features.
- Specify options for processing the IP core files in other EDA tools.

Note: Refer to your IP core user guide for information about specific IP core parameters.

5. Click **Generate HDL**. The **Generation** dialog box appears.
6. Specify output file generation options, and then click **Generate**. The synthesis and/or simulation files generate according to your specifications.
7. To generate a simulation testbench, click **Generate ➤ Generate Testbench System**. Specify testbench generation options, and then click **Generate**.
8. To generate an HDL instantiation template that you can copy and paste into your text editor, click **Generate ➤ Show Instantiation Template**.
9. Click **Finish**. Click **Yes** if prompted to add files representing the IP variation to your project.
10. After generating and instantiating your IP variation, make appropriate pin assignments to connect ports.

Note: Some IP cores generate different HDL implementations according to the IP core parameters. The underlying RTL of these IP cores contains a unique hash code that prevents module name collisions between different variations of the IP core. This unique code remains consistent, given the same IP settings and software version during IP generation. This unique code can change if you edit the IP core's parameters or upgrade the IP core version. To avoid dependency on these unique codes in your simulation environment, refer to *Generating a Combined Simulator Setup Script*.

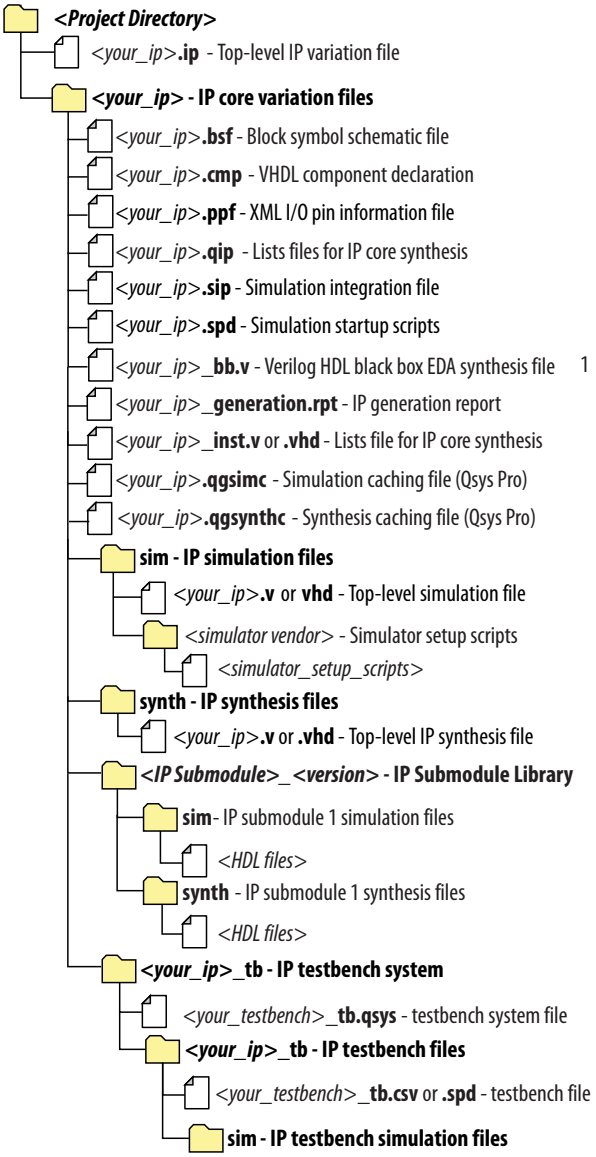
Related Links

- IP User Guide Documentation
- Intel FPGA IP Release Notes

2.3.1 IP Core Generation Output (Quartus Prime Pro Edition)

The Quartus Prime software generates the following output file structure for individual IP cores that are not part of a Qsys system.

Figure 5. Individual IP Core Generation Output (Quartus Prime Pro Edition)



1. If supported and enabled for your IP core variation.

Table 9. Files Generated for IP Cores

File Name	Description
<*my_ip*>.ip	Top-level IP variation file that contains the parameterization of an IP core in your project. If the IP variation is part of a Qsys Pro system, the parameter editor also generates a .qsys file.
<*my_ip*>.cmp	The VHDL Component Declaration (.cmp) file is a text file that contains local generic and port definitions that you use in VHDL design files.
<*my_ip*>_generation.rpt	IP or Qsys generation log file. A summary of the messages during IP generation.

continued...

File Name	Description
<*my_ip*>.qgsimc (Qsys Pro systems only)	Simulation caching file that compares the .qsys and .ip files with the current parameterization of the Qsys Pro system and IP core. This comparison determines if Qsys Pro can skip regeneration of the HDL.
<*my_ip*>.qgsynth (Qsys Pro systems only)	Synthesis caching file that compares the .qsys and .ip files with the current parameterization of the Qsys Pro system and IP core. This comparison determines if Qsys Pro can skip regeneration of the HDL.
<*my_ip*>.qip	Contains all information to integrate and compile the IP component.
<*my_ip*>.csv	Contains information about the upgrade status of the IP component.
<*my_ip*>.bsf	A symbol representation of the IP variation for use in Block Diagram Files (.bdf).
<*my_ip*>.spd	Required input file for ip-make-simscript to generate simulation scripts for supported simulators. The .spd file contains a list of files you generate for simulation, along with information about memories that you initialize.
<*my_ip*>.ppf	The Pin Planner File (.ppf) stores the port and node assignments for IP components you create for use with the Pin Planner.
<*my_ip*>_bb.v	Use the Verilog blackbox (_bb.v) file as an empty module declaration for use as a blackbox.
<*my_ip*>.sip	Contains information you require for NativeLink simulation of IP components. Add the .sip file to your Quartus Prime Standard Edition project to enable NativeLink for supported devices. The Quartus Prime Pro Edition software does not support NativeLink simulation.
<*my_ip*>_inst.v or _inst.vhd	HDL example instantiation template. Copy and paste the contents of this file into your HDL file to instantiate the IP variation.
<*my_ip*>.regmap	If the IP contains register information, the Quartus Prime software generates the .regmap file. The .regmap file describes the register map information of master and slave interfaces. This file complements the ..sopcinfo file by providing more detailed register information about the system. This file enables register display views and user customizable statistics in System Console.
<*my_ip*>.svd	Allows HPS System Debug tools to view the register maps of peripherals that connect to HPS within a Qsys Pro system. During synthesis, the Quartus Prime software stores the .svd files for slave interface visible to the System Console masters in the .sof file in the debug session. System Console reads this section, which Qsys Pro queries for register map information. For system slaves, Qsys Pro accesses the registers by name.
<*my_ip*>.v <*my_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.
aldec/	Contains a Riviera-PRO script rivierapro_setup.tcl to setup 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.
/submodules	Contains HDL files for the IP core submodule.
<*IP submodule*>/	For each generated IP submodule directory Qsys Pro generates /synth and /sim sub-directories.

2.4 Simulating Intel FPGA IP Cores

The Quartus Prime software supports IP core RTL simulation in specific EDA simulators. IP generation creates simulation files, including the functional simulation model, any testbench (or example design), and vendor-specific simulator setup scripts for each IP core. Use the functional simulation model and any testbench or example design for simulation. IP generation output may also include scripts to compile and run any testbench. The scripts list all models or libraries you require to simulate your IP core.

The Quartus Prime software provides integration with many simulators and supports multiple simulation flows, including your own scripted and custom simulation flows. Whichever flow you choose, IP core simulation involves the following steps:

1. Generate simulation model, testbench (or example design), and simulator setup script files.
2. Set up your simulator environment and any simulation script(s).
3. Compile simulation model libraries.
4. Run your simulator.

3 High-speed Reed-Solomon IP Core Functional Description

This topic describes the IP core's architecture, interfaces, and signals.

3.1 High-Speed Reed-Solomon Architecture

The encoder receives data packets and generates the check symbols; the decoder detects and corrects errors.

The High-speed Reed-Solomon IP core has a parallelized architecture to achieve very high throughout. The inputs and outputs contain multiple data symbols.

The fracturable decoder has preset parameters to support 4 x 25 GbE, 2 x 50 GbE and 1 x 100 GbE with parallelism *p* of 8, 16, and 32, respectively.

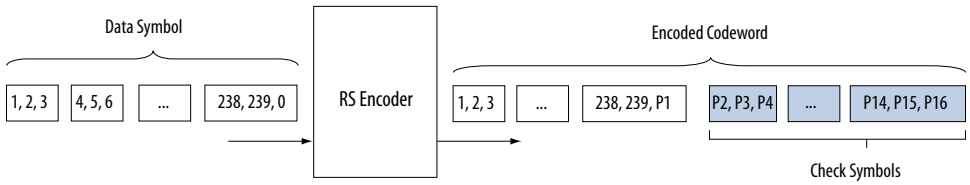
3.1.1 High-Speed Reed-Solomon Encoder

When the encoder receives data symbols, it generates check symbols for a given codeword and sends the input codeword together with the check symbols to the output interface. .

The encoder may uses backpressure on the upstream component when it generates the check symbols and the parallelism is smaller than the number of check symbols

Figure 6. High-speed Reed-Solomon Encoding

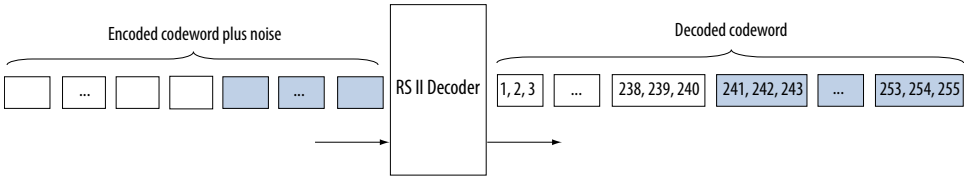
Shows how a codeword is encoded



3.1.2 High-Speed Reed-Solomon Decoder

When the decoder receives the encoded codeword, it uses the check symbols to detect errors and correct them. The decoder is a streaming decoder that allows continuous input data with no backpressure on the upstream component.

Figure 7. Codeword Decoding



The received encoded codeword may differ from the original codeword due to the noise in the channel. The decoder detects errors using several polynomials to locate the error location and the error value. When the decoder obtains the error location and value, the decoder corrects the errors in a codeword, and sends the codeword to the output. As the number of errors increases, the decoder gets to a stage where it can no longer correct but only detect errors, at which point the decoder asserts the `out_error` signal.

3.2 High-Speed Reed-Solomon IP Core Parameters

Table 10. Parameters

Parameter	Legal Values	Default Value	Description
Reed-Solomon Core	Fractuarable 100G Ethernet or custom	Fracturable 100G Ethernet	-
Reed-Solomon module	Encoder or Decoder	Decoder	An encoder or a decoder.
Custom IP Core			
Number of channels	1 to 10	1	Decoder only.
Number of bits per symbol	3 to 12	8	The number of bits per symbol (*M*).
Number of symbols per codeword	1 to 2*M* – 1	255	The total number of symbols per codeword (*N*).
Number of data symbols per codeword	2 to *N*– 2	239	The number of data symbols per codeword (*K* = *N* - *R*). Where *R* is the number of check symbols.
Field polynomial	Any valid polynomial	285	The primitive polynomial defining the Galois field. The parameter editor allows you to select only legal values. If you cannot find your intended field polynomial, contact Intel MySupport.
Parallelism	*P* < *N*/2	-	The number of parallel inputs and outputs. The last output fills up with zeros.
Bypass Mode	On or off	Off	Turn on to produce the received codeword without error correction but with only error detection. This message is output after few clock cycles.
Fractuarable IP core Parameters			
100G Ethernet	On	On	A single 100G channels with parallelism 32.

continued...

Parameter	Legal Values	Default Value	Description
2 x 50G Ethernet	On or off	Off	Two independent 50 GbE channels with parallelism 16.
4 x 25G ethernet	On or off	Off	Four independent 25 GbE channels with parallelism 8.
Custom and Fracturable IP Core Parameters			
Latency	*N*/*P*+ (*BM speed* x *R*) +10; with *BMspeed* is 4 or 6.	*BMspeed* is 4	The latency as a function of the Berlekamp–Massey (BM) speed (decoder only).
Favor ROM	On or off	On	Uses M20K memory to reduce the ALMs. Savings are significant with large parallelism.
Use true dual-port ROM	No or yes	No	-
Refresh ROM content	No or yes	No	The IP core continuously rewrites the ROM contents.
Use backpressure	No or yes	No	Decoder only. When you select **Yes**, you can use the `out_ready` signal to assert when the source is ready, but this option might limit f_{MAX}.
Decoder Output Signal Options			
Output decoding failure	On or off	On	Turns on `out_decfail` signal
Output error symbol count	On of off	On	Turns on `out_errors_out` signal.
Output error symbol values	On or off	On	Turns on `out_errorvalues_out` signal.
Output start and end of packet	On or off.	On	Use Avalon-ST SOP and EOP signal to indicate start and end of packet

3.3 High-speed Reed-Solomon IP Core Interfaces and Signals

The ready latency on the Avalon-ST input interface is 0; the number of symbols per beat is fixed to 1. The latency of the decoder is *N*/*P*+ *(BM speed* x *R*) + 10; the latency of the encoder is 6.

The clock and reset interfaces drive or receive the clock and reset signal to synchronize the Avalon-ST interfaces and provide reset connectivity. The status interface is a conduit interface that consists of three error status signals for a codeword. The decoder obtains the error symbol value, number of error symbols, and number of error bits in a codeword from the status signals.

Avalon-ST Interfaces

Avalon-ST 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-ST sink and the output interface is an Avalon-ST source.

Avalon-ST 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-ST interfaces support backpressure, which is a flow control mechanism where a sink can indicate 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.

3.3.1 High-speed Reed-Solomon IP Core Signals

Table 11. Clock and Reset Signals

Name	Avalon-ST Type	Direction	Description
clk_clk	clk	Input	The main system clock. The whole IP core operates on the rising edge of clk_clk .
reset_reset_n	reset_n	Input	An active low signal that resets the entire system when asserted. You can assert this signal asynchronously. However, you must deassert it synchronous to the clk_clk signal. When the IP core recovers from reset, ensure that the data it receives is a complete packet.

Table 12. Custom IP Core Avalon-ST Interface Signals

Name	Avalon-ST Type	Direction	Description
in_ready	ready	Output	Data transfer ready signal to indicate that the sink is ready to accept data. The sink interface drives the in_ready signal to control the flow of data across the interface. The sink interface captures the data interface signals on the current clk rising edge.
in_valid	valid	Input	Data valid signal to indicate the validity of the data signals. When you assert the in_valid signal, the Avalon-ST data interface signals are valid. When you deassert the in_valid signal, the Avalon-ST data interface signals are invalid and must be disregarded. You can assert the in_valid signal whenever data is available. However, the sink only captures the data from the source when the IP core asserts the in_ready signal.
in_data[]	data	Input	Data input for each codeword, symbol by symbol. Valid only when you assert the in_valid signal. Width is $P \times M$ bits. For the encoder, the number of information symbols (N - CHECK) is not necessarily a multiple of P. It means that the last input symbol may have to be filled with zeros.
out_data	data	Output	Encoder output. In Qsys systems for the decoder, this Avalon-ST-compliant data bus includes all the Avalon-ST output data signals (out_error_out, out_decfail, out_symol_out),) with length $\log_2(R+1) + 1$.
out_decfail	data	Output	Decoding failure.
out_errors_out	error	Output	Number of error symbol that the IP core decides. Size is $\log_2(R+1)$
out_errorvalues_out	error	Output	Error values.
out_ready	ready	Input	Data transfer ready signal to indicate that the downstream module is ready to accept data. The source provides new data (if available) when you assert the out_ready signal and stops providing new data when you deassert the

continued...

Name	Avalon-ST Type	Direction	Description
			out_ready signal. If the source is unable to provide new data, it deasserts out_valid for one or more clock cycles until it is prepared to drive valid data interface signals.
out_symbols_out	data	Output	Contains decoded output when the IP core asserts the out_valid signal. The corrected symbols are in the same order that they are entered.
out_valid	valid	Output	Data valid signal. The IP core asserts the out_valid signal high, whenever a valid output is on out_data ; the IP core deasserts the signal when there is no valid output on out_data .

Table 13. Fractuarable IP Core Avalon-ST Interface Signals

Name	Avalon-ST Type	Direction	Width	Description
in-valid	Valid	Input	1	Master valid signal. If in_valid is low it sets all valid_ch_in to low.
in-data	Data	Input	320 symbols_in + 4 valid_ch_in + 2 mode_in + sync_in	Data input.
valid_ch_in	Part of in_data	Input	4	Input valid signal for each channel.
symbols_in	Part of in_data	Input	32	Input symbols. • 100 GbE one channel • 50 GbE two channels • 25 GbE four channels
mode_in	Part of in_data	Input	2	• 0: 1x100 GbE • 1: 2x50 GbE or 4x25GbE • 2: 4x25 GbE
sync_in	Part of in_data	Input	1	Synchronize the output channels.
out_valid	Valid	Output	1	Master valid signal. out_valid is valid if any valid_ch_out is valid, i.e. if valid_ch0 or valid_ch1 etc are valid.
out_data	Data	Output	320 decoded symbols + 4 valid_out + 2 mode_out + 4 sop_out + 4 eop_out +4 decfail+ 12 errors_out	Output data.
errors_out	Part of out_data	Output	12	Number of error symbols that the IP core decides.
decfail	Part of out_data	Output	4	(Optional) decfail of each output channel

continued...

Name	Avalon-ST Type	Direction	Width	Description
eop_out	Part of out_data	Output	4	(Optional) eop of each output channel
sop_out	Part of out_data	Output	4	(Optional) sop of each channels
mode_out	Part of out_data	Output	2	Output mode.
valid_ch_out	Part of out_data	Output	4	Valid signal for each channel
symbols_out	Part of out_data	Output	320	Output symbols: • 100 GbE one channel • 50 GbE two channels • 25 GbE four channels

4 Document Revision History

High-speed Reed-Solomon IP Core User Guide revision history.

Date	Version	Changes
2016.11.02	16.1	• Added fracturable IP core parameters. • Added **Output error symbol values** parameter • Added **Bypass mode** parameter
2016.05.02	16.0	• Added new parameters: — Number of channels — Use backpressure — Use true dual-port ROM — Refresh ROM content — Output decoding failure — Output error symbol count — Output start and end of packet • Added `out_decfail` signal • Added **True dual-port ROM** performance data. • Changed description of parallelism parameter. • Changed `out_error_out` signal description
2015.11.01	15.1	Corrected encoder and decoder diagrams
2015.05.01	15.0	First release

A High-Speed Reed-Solomon IP Core Document Archive

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

IP Core Version	User Guide
15.1	High-speed Reed-Solomon IP Core User Guide

Related Links

About the High-speed Reed-Solomon IP Core on page 3
Provides a list of user guides for previous versions of the High-speed Reed-Solomon IP core.