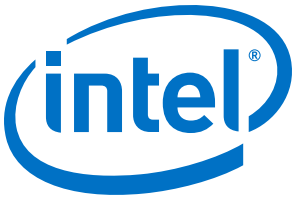


CIC IP Core

User Guide

Updated for Intel® Quartus® Prime Design Suite: **17.1**

Subscribe
Send Feedback

UG-CIC | 2017.11.06
Latest document on the web: PDF | HTML

Contents

1 About The CIC IP Core 3
1.1 Intel DSP IP Core Features 3
1.2 CIC IP Core Features 3
1.3 DSP IP Core Device Family Support 4
1.4 DSP IP Core Verification 5
1.5 CIC IP Core Release Information 5
1.6 CIC IP Core Performance and Resource Utilization 5

2 CIC IP Core Getting Started 8
2.1 Installing and Licensing Intel FPGA IP Cores 8
2.1.1 Intel FPGA IP Evaluation Mode 8
2.1.2 CIC IP Core Intel FPGA IP Evaluation Mode Timeout Behavior 11
2.2 IP Catalog and Parameter Editor 11
2.3 Generating IP Cores (Intel Quartus Prime Pro Edition) 13
2.3.1 IP Core Generation Output (Intel Quartus Prime Pro Edition) 14
2.4 Simulating Intel FPGA IP Cores 17
2.5 DSP Builder for Intel FPGAs Design Flow 17

3 CIC IP Core Functional Description 18
3.1 Variable Rate Change Factors 19
3.2 Multichannel Support 19
3.2.1 Multiple Input Single Output (MISO) 19
3.2.2 Single Input Multiple Output (SIMO) 20
3.3 Output Options 21
3.3.1 Output Data Width 21
3.3.2 Output Rounding 22
3.3.3 Hogenauer Pruning 22
3.4 FIR Filter Compensation Coefficients 23
3.5 CIC IP Core Parameters 24
3.6 CIC IP Core Interfaces and Signals 26
3.6.1 Avalon-ST Interfaces in DSP IP Cores 26
3.6.2 CIC IP Core Signals 27
3.6.3 Avalon-ST Interface Data Transfer Timing 28
3.6.4 Packet Data Transfers 28

A CIC IP Core User Guide Document Archives 30

5 Document Revision History 31

1 About The CIC IP Core

The Intel® CIC IP core implements a cascaded integrator-comb (CIC) filter with data ports that are compatible with the Avalon® Streaming (Avalon-ST) interface. CIC filters (also known as Hogenauer filters) are computationally efficient for extracting baseband signals from narrow-band sources using decimation. They also construct narrow-band signals from processed baseband signals using interpolation.

CIC filters use only adders and registers; they require no multipliers to handle large rate changes. Therefore, CIC is a suitable and economical filter architecture for hardware implementation, and is widely used in sample rate conversion designs such as digital down converters (DDC) and digital up converters (DUC).

Related Links

- Introduction to Altera 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.

1.1 Intel DSP IP Core Features

- Avalon Streaming (Avalon-ST) interfaces
- DSP Builder for Intel FPGAs ready
- Testbenches to verify the IP core
- IP functional simulation models for use in Intel-supported VHDL and Verilog HDL simulators

1.2 CIC IP Core Features

- Interpolation and decimation filters with variable rate change factors (2 to 32,000), a configurable number of stages (1 to 12), and two differential delay options (1 or 2).
- Single clock domain with selectable number of interfaces and a maximum of 1,024 channels.
- Selectable data storage options with an option to use pipelined integrators.
- Configurable input data width (1 to 32 bits) and output data width (1 to full resolution data width).

- Selectable output rounding modes (truncation, convergent rounding, rounding up, or saturation) and Hogenauer pruning support.
- Optimization for speed by specifying the number of pipeline stages used by each integrator.
- Compensation filter coefficients generation.

1.3 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—Intel verifies 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
Intel Arria 10	Final
Cyclone® IV	Final
Cyclone V	Final
Intel Cyclone 10	Final
Intel MAX® 10 FPGA	Final
Stratix® IV GT	Final
Stratix IV GX/E	Final
Stratix V	Final
Intel Stratix 10	Advance
Other device families	No support

1.4 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.

1.5 CIC IP Core Release Information

Table 2. CIC IP Core Release Information

Item	Description
Version	17.1
Release Date	November 2017
Ordering Code	IP-CIC

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 CIC IP core in the Knowledge Base

1.6 CIC IP Core Performance and Resource Utilization

The following parameters apply:

- **Number of stages**: 8
- **Rate change factor**: 8
- **Differential delay**: 1
- **Integrator data storage**: Memory (whenever possible)
- **Differentiator data storage**: Memory (whenever possible)
- **Input data width**: 16
- **Output data width**: Full precision
- **Output rounding**: No rounding

The target f_{MAX} is 1 GHz.

Table 3. CIC IP Core Performance

Typical performance using the Quartus II software with the Arria V (5AGXFB3H4F40C4), Cyclone V (5CGXFC7D6F31C6), and Stratix V (5SGSMD4H2F35C2) devices

Device	Filter Type	ALM	Memory		Registers		f_{MAX} (MHz)
			M10K	M20K	Primary	Secondary	
Arria V	Decimator	493	2	--	1,149	5	207.34
Arria V	Decimator 5 Channels	1,162	2	--	3,749	6	207

continued...

Device	Filter Type	ALM	Memory		Registers		f_{MAX} (MHz)
			M10K	M20K	Primary	Secondary	
Arria V	Decimator 5 Channels 3 Interfaces	911	37	--	1,722	6	255
Arria V	Decimator Hogenauer Pruning	352	1	--	785	12	304
Arria V	Decimator Trunction	463	2	--	1,055	5	198.69
Arria V	Decimator Variable Rate Change	919	37	--	1,730	7	256
Arria V	Interpolator	326	1	--	728	18	320
Arria V	Interpolator 5 Channels	762	1	--	2,369	27	288
Arria V	Interpolator 5 Channels 3 Interfaces	886	27	--	1,776	17	232.61
Arria V	Interpolator Convergent Rounding	352	1	--	785	12	304
Arria V	Interpolator Variable Rate Change	889	27	--	1,772	23	235
Cyclone V	Decimator	492	2	--	1,137	17	182
Cyclone V	Decimator 5 Channels	1,162	2	--	3,748	8	190.15
Cyclone V	Decimator 5 Channels 3 Interfaces	906	37	--	1,719	9	204
Cyclone V	Decimator Hogenauer Pruning	352	1	--	784	14	246
Cyclone V	Decimator Truncation	463	2	--	1,054	4	177
Cyclone V	Decimator Variable Rate Change	917	37	--	1,730	5	193.27
Cyclone V	Interpolator	324	1	--	709	37	264
Cyclone V	Interpolator 5 Channels	760	1	--	2,383	11	235
Cyclone V	Interpolator 5 Channels 3 Interfaces	890	27	--	1,747	48	168
Cyclone V	Interpolator Convergent Rounding	352	1	--	784	14	246.06
Cyclone V	Interpolator Variable Rate Change	894	27	--	1,725	70	165
Stratix V	Decimator	515	--	1	1,152	6	377
Stratix V	Decimator 5 Channels	1,176	--	1	3,750	8	413
Stratix V	Decimator 5 Channels 3 Interfaces	1,891	--	11	5,562	8	450.05
Stratix V	Decimator Hogenauer Pruning	361	--	0	790	13	450
Stratix V	Decimator Truncation	483	--	1	1,059	4	376
Stratix V	Decimator Variable Rate Change	1,900	--	11	5,574	3	450
Stratix V	Interpolator	335	--	0	737	14	450.05

continued...

Device	Filter Type	ALM	Memory		Registers		f_{MAX} (MHz)
			M10K	M20K	Primary	Secondary	
Stratix V	Interpolator 5 Channels	771	--	0	2,390	8	450
Stratix V	Interpolator 5 Channels 3 Interfaces	1,625	--	8	4,635	70	450
Stratix V	Interpolator Convergent Rounding	361	--	0	790	13	450
Arria 10	Interpolator Variable Rate Change	464	--	0	128	128	451

2 CIC IP Core Getting Started

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

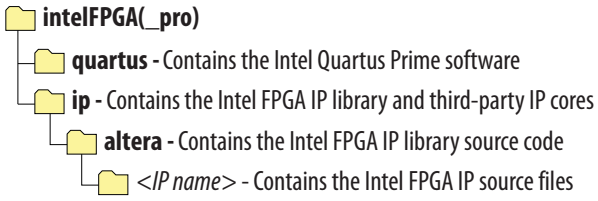


Table 4. IP Core Installation Locations

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.

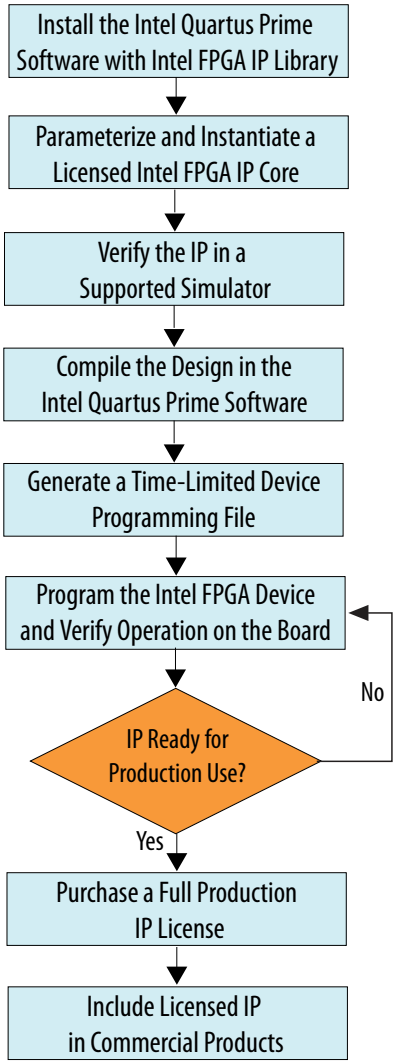
Intel FPGA IP Evaluation Mode supports the following operation modes:

- 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



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 or contact your local Intel FPGA representative.

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 Links

- Intel Quartus Prime Licensing Site
- Intel FPGA Software Installation and Licensing

2.1.2 CIC IP Core Intel FPGA IP Evaluation Mode Timeout Behavior

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, 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 II 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, the data output signal 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 Intel Quartus Prime IP file (`.ip`) for an IP variation in Intel Quartus Prime Pro Edition projects.

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

Figure 3. IP Parameter Editor (Intel Quartus Prime Pro Edition)

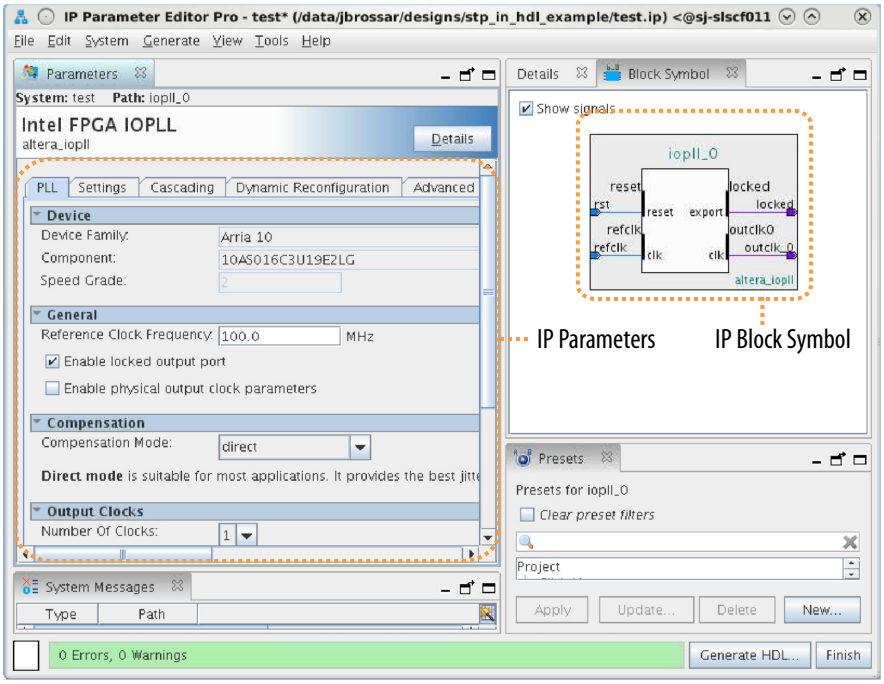
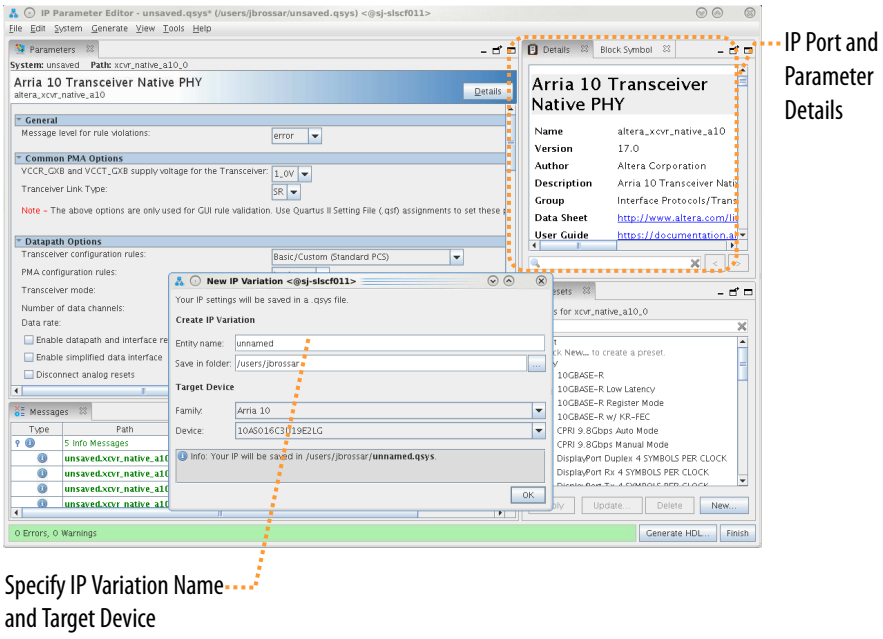


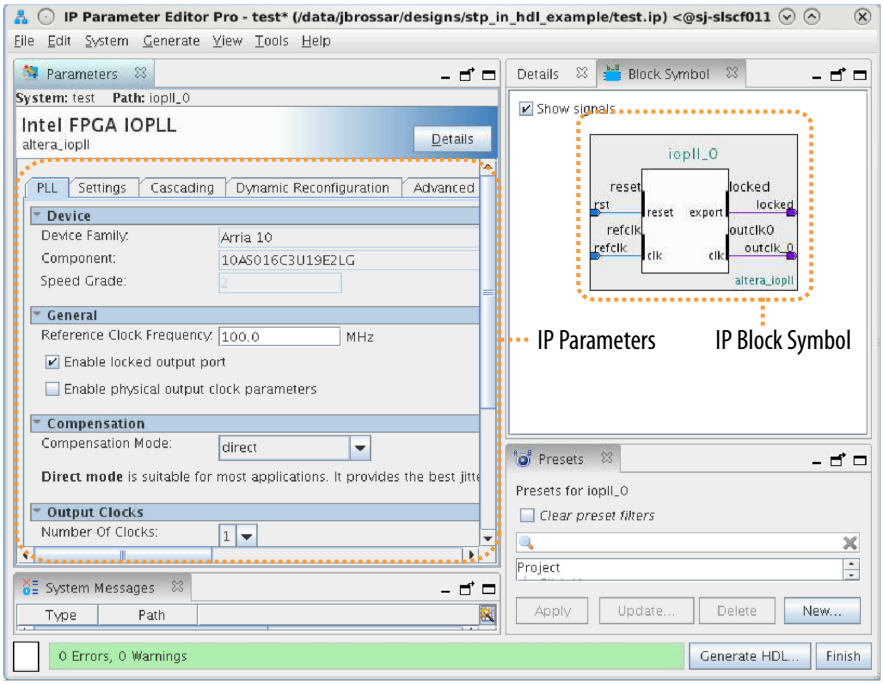
Figure 4. IP Parameter Editor (Intel Quartus Prime Standard Edition)



2.3 Generating IP Cores (Intel Quartus Prime Pro Edition)

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.

Figure 5. IP Parameter Editor (Intel Quartus Prime Pro Edition)



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

1. Create or open an Intel Quartus Prime project (`.qpf`) to contain the instantiated IP variation.
2. In the IP Catalog (**Tools ➤ IP Catalog**), locate and double-click the name of the IP core to customize. To locate a specific component, type some or all of the component's name in the IP Catalog search box. The New IP Variation window appears.
3. Specify a top-level name for your custom IP variation. Do not include spaces in IP variation names or paths. The parameter editor saves the IP variation settings in a file named <*your_ip*>`.ip`. Click **OK**. The parameter editor appears.
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.

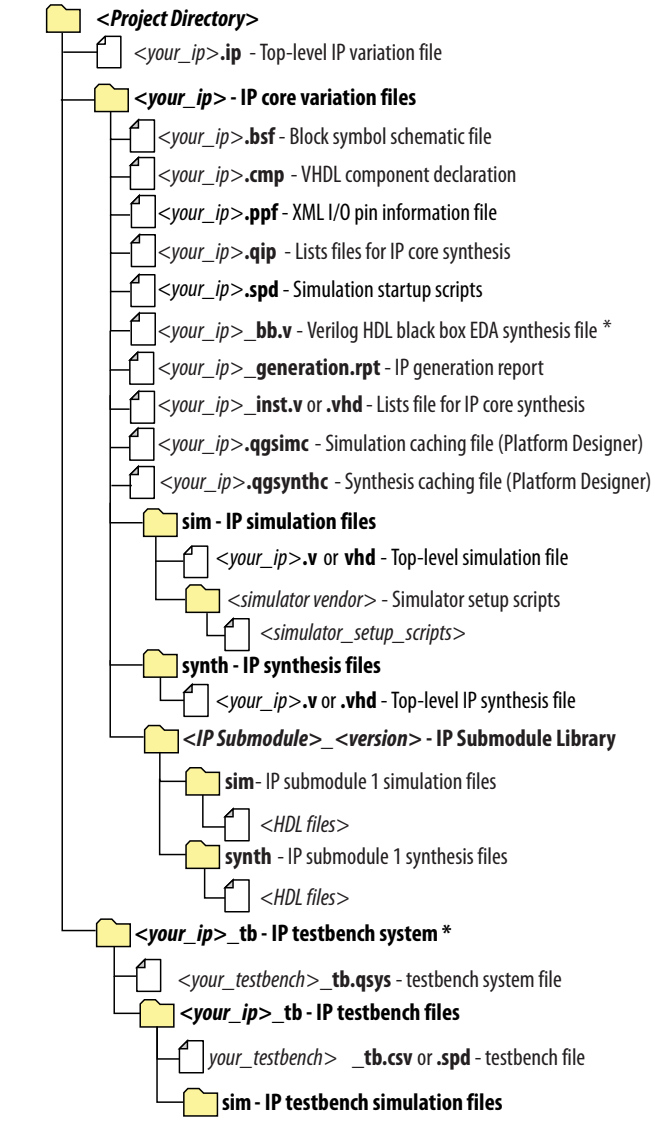
5. Click **Generate HDL**. The **Generation** dialog box appears.
6. Specify output file generation options, and then click **Generate**. The synthesis and simulation files generate according to your specifications.
7. To generate a simulation testbench, click **Generate ➤ Generate Testbench System**. 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*.

2.3.1 IP Core Generation Output (Intel Quartus Prime Pro Edition)

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

Figure 6. Individual IP Core Generation Output (Intel Quartus Prime Pro Edition)



* If supported and enabled for your IP core variation.

Table 5. Output Files of Intel FPGA IP Generation

File Name	Description
<your_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 Platform Designer system, the parameter editor also generates a .qsys file.
<your_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.
<your_ip>_generation.rpt	IP or Platform Designer generation log file. Displays a summary of the messages during IP generation.

continued...

File Name	Description
<*your_ip*>.qgsimc (Platform Designer systems only)	Simulation caching file that compares the .qsys and .ip files with the current parameterization of the Platform Designer system and IP core. This comparison determines if Platform Designer can skip regeneration of the HDL.
<*your_ip*>.qgsynth (Platform Designer systems only)	Synthesis caching file that compares the .qsys and .ip files with the current parameterization of the Platform Designer system and IP core. This comparison determines if Platform Designer can skip regeneration of the HDL.
<*your_ip*>.qip	Contains all information to integrate and compile the IP component.
<*your_ip*>.csv	Contains information about the upgrade status of the IP component.
<*your_ip*>.bsf	A symbol representation of the IP variation for use in Block Diagram Files (.bdf).
<*your_ip*>.spd	Input file that ip-make-simscript requires to generate simulation scripts. The .spd file contains a list of files you generate for simulation, along with information about memories that you initialize.
<*your_ip*>.ppf	The Pin Planner File (.ppf) stores the port and node assignments for IP components you create for use with the Pin Planner.
<*your_ip*>_bb.v	Use the Verilog blackbox (_bb.v) file as an empty module declaration for use as a blackbox.
<*your_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.
<*your_ip*>.regmap	If the IP contains register information, the Intel 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.
<*your_ip*>.svd	Allows HPS System Debug tools to view the register maps of peripherals that connect to HPS within a Platform Designer system. During synthesis, the Intel 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 Platform Designer queries for register map information. For system slaves, Platform Designer accesses the registers by name.
<*your_ip*>.v <*your_ip*>.vhd	HDL files that instantiate each submodule or child IP core for synthesis or simulation.
mentor/	Contains a msim_setup.tcl script to set up and run a ModelSim 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*>/	Platform Designer generates /synth and /sim sub-directories for each IP submodule directory that Platform Designer generates.

2.4 Simulating Intel FPGA IP Cores

The Intel 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 Intel 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 scripts.
3. Compile simulation model libraries.
4. Run your simulator.

2.5 DSP Builder for Intel FPGAs Design Flow

DSP Builder for Intel FPGAs shortens digital signal processing (DSP) design cycles by helping you create the hardware representation of a DSP design in an algorithm-friendly development environment.

This IP core supports DSP Builder for Intel FPGAs. Use the DSP Builder for Intel FPGAs flow if you want to create a DSP Builder for Intel FPGAs model that includes an IP core variation; use IP Catalog if you want to create an IP core variation that you can instantiate manually in your design.

Related Links

Using MegaCore Functions chapter in the DSP Builder for Intel FPGAs Handbook.

3 CIC IP Core Functional Description

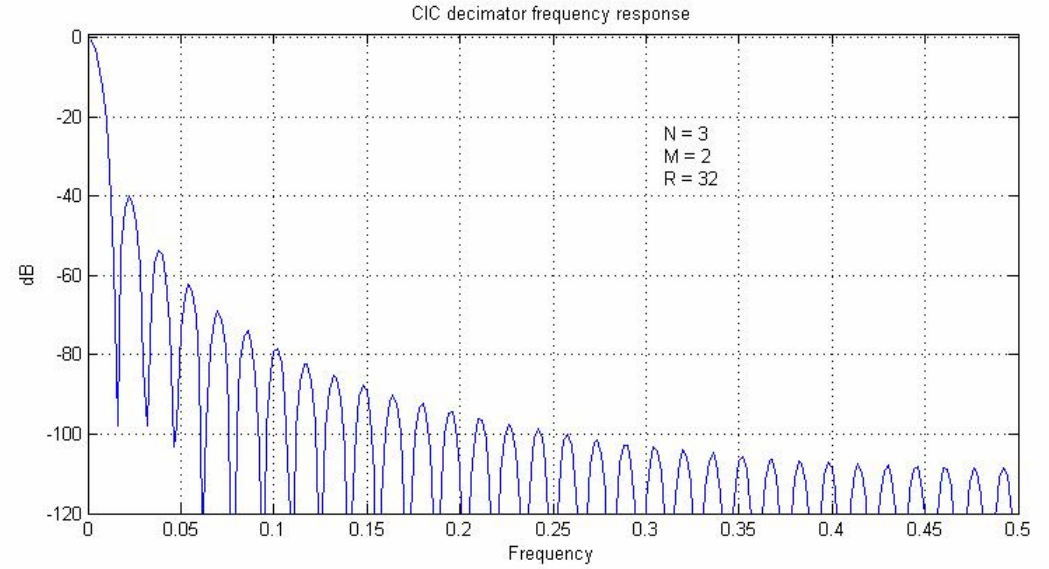
You can select either a decimation or interpolation CIC filter. A decimation CIC filter comprises a cascade of integrators (integrator), followed by a down sampling block (decimator) and a cascade of differentiators (called the differentiator or comb section). Similarly an interpolation CIC filter comprises a cascade of differentiators, followed by an up sampling block (interpolator) and a cascade of integrators.

In a CIC filter, both the integrator and comb sections have the same number of integrators and differentiators. Each pairing of integrator and differentiator is a stage. The number of stages (*N*) has a direct effect on the frequency response of a CIC filter. You determine the response of the filter by configuring:

- The number of stages *N*
- The rate change factor *R*
- The number of delays in the differentiators (differential delay) *M*. Generally, set the differential delay to 1 or 2.

Figure 7. Three-stage CIC Decimation Filter Frequency Response

CIC decimation filter with *N* = 3, *M* = 2 and *R* = 32



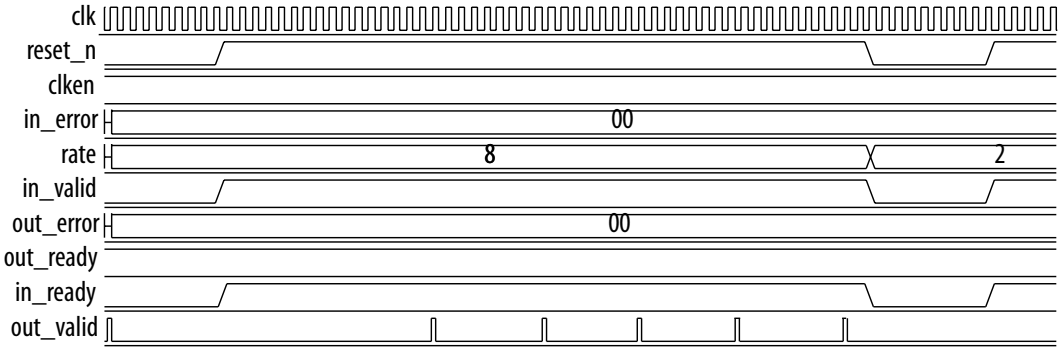
3.1 Variable Rate Change Factors

You can optionally set minimum and maximum values for the decimator or interpolator rate change factors and enable the rate change factors to be set at run time. With these options, the CIC provides an additional `rate` port that you can use to specify the rate change factor.

Note: With variable rate change factors, reset the IP core when you change the rate change factor, otherwise the CIC uses previous memory and register values. You cannot change the filter mode (interpolation or decimation) at run time.

Figure 8. Variable Rate Change Decimation CIC Filter Timing Diagram

The out_valid signal changes its period according to the variable rate change.



3.2 Multichannel Support

Often many channels of data in a digital signal processing (DSP) system require filtering by CIC filters with the same configuration. You can combine them into one filter, which shares the adders that exist in each stage and reduces the overall resource utilization.

Using a combined filter uses fewer resources than using many individual CIC filters. For example, a two-channel parallel filter requires two clock cycles to calculate two outputs. The resulting hardware needs to run at twice the data rate of an individual filter, which is especially useful for higher rate changes where adders grow particularly large.

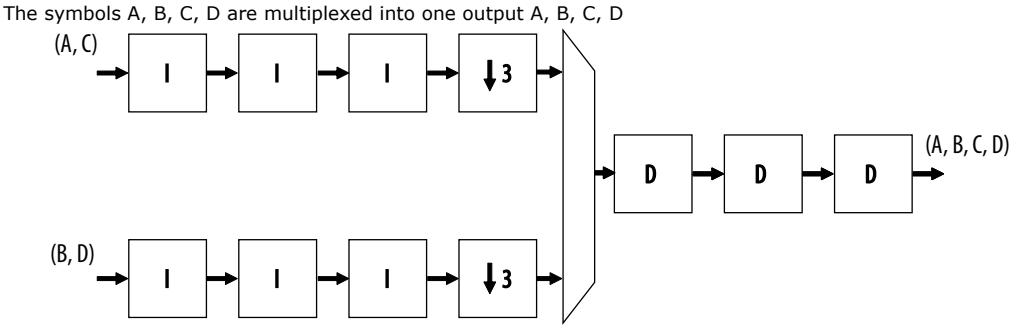
Note: To minimize the number of logic elements, use a multiple input single output (MISO) architecture for decimation filters, and a single input multiple output (SIMO) architecture for interpolation filters.

3.2.1 Multiple Input Single Output (MISO)

In many practical designs, channel signals come from different input interfaces. On each input interface, the same parameters including rate change factors apply to the channel data that the CIC filter is going to process. The CIC IP core allows multiple input single output (MISO) decimation filters, which allows the flexibility to exploit time sharing of the low-rate differentiator sections.

The CIC achieves time sharing by providing multiple input interfaces and processing chains for the high rate portions. It then combines all of the processing associated with the lower rate portions into a single processing chain. This strategy can lead to full utilization of the resources and represents the most efficient hardware implementation.

Figure 9. Multiple Input Single Output Architecture For Four Channels



The sampling frequency of the input data only allows time multiplexes of two channels per bus. Therefore, you must configure the CIC filter with two input interfaces. For two interfaces, the rate change factor must also be at least two to exploit this architecture. The CIC support up to 1,024 channels by using multiple input interfaces in this way.

Note: The CIC applies the MISO architecture when you select a decimation filter and the number of interfaces is greater than one.

3.2.2 Single Input Multiple Output (SIMO)

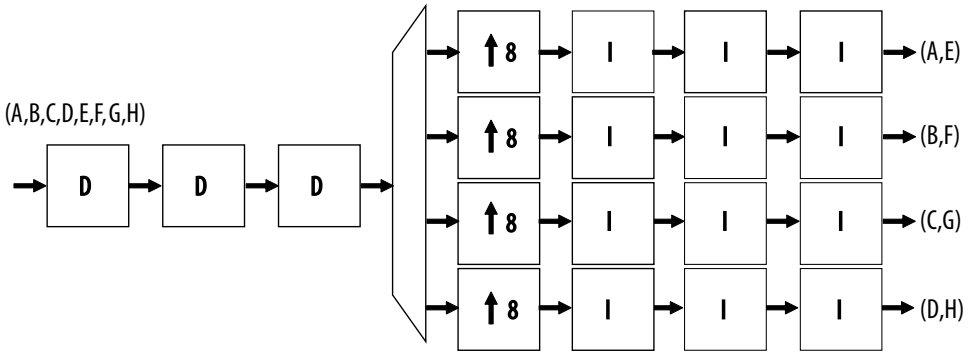
With single input multiple output (SIMO), all the channel signals presented for filtering come from a single input interface.

Like the MISO, you can share the low sampling rate differentiator section among more channels than the higher sampling frequency integrator sections. Therefore, this architecture features a single instance of the differentiator section and multiple parallel instances of the integrator sections.

After processing by the differentiator section, the CIC splits the channel signals into multiple parallel sections for processing in a high sampling frequency by the integrator sections.

Figure 10. Single Input Multiple Output Architecture with Eight Channels

The symbols A, B, C, D, E, F, G, H are demultiplexed into four outputs A, E; B, F; C, G; and D, H



The required sampling frequency of the output data only allows time multiplexes of two channels per bus. Therefore, you must configure the CIC filter with four output interfaces. The rate change factor must also be at least four to exploit this architecture, but this example shows a rate change of eight.

Note: The CIC applies a SIMO when you select an interpolation filter and the number of interfaces is greater than one.

The total number of input channels must be a multiple of the number of interfaces. To satisfy this requirement, you may need to either insert dummy channels or use more than one CIC IP core.

The CIC transfers data as packets using Avalon Avalon-ST interfaces.

Related Links

AN442: Tool Flow Design of Digital IF for Wireless Systems
An example design using multichannel MISO and SIMO architectures.

3.3 Output Options

You can select output options for the output data bit width and rounding options.

3.3.1 Output Data Width

If you select an output data width that is smaller than the full output resolution data width, apply the Hogenauer pruning technique to reduce the data widths across the filter stages and hence the overall resource utilization.

For a decimation filter, the gain at the output of the filter is:

$G = RM^N$

Therefore, the data width at the output stage for if full resolution is:

$B_{out} = B_{in} + N\log_2(RM)$

where Bin is the input data width.

Note: A data width of B_{out} is required for each integrator and differentiator for no data loss.

For an interpolation filter, the gain at each filter stage is:

$$G_i = \begin{cases} 2^i & i = 1, 2, \ldots, N \\ \frac{2^{2N-1}(RM)^{i-N}}{R} & i = N+1, \ldots, 2N \end{cases}$$

Hence the required data width at the ith stage is:

$W_i = [B_{in} + \log_2(G_i)]$

and the data width at the output stage is:

$B_{out} = [B_{in} + N\log_2(RM) - \log2(R)]$

where B_{in} is the input data width.

When the differential delay is one, the bit width at each integrator stage is increased by one to ensure stability.

For more information about these calculations, refer to Hogenauer, Eugene. *An Economical Class of Digital Filters For Decimation and Interpolation,* IEEE Transactions on Acoustics, Speech and Signal Processing, Vol. ASSP-29, pp. 155-162, April 1981.

3.3.2 Output Rounding

For high rate change factors, the maximum required data width for no data loss is large for many practical cases. To reduce the output data width to the input level, apply quantization at the end of the output stage. the CIC filter offers various rounding or saturation options. You can only apply these rounding options to the output stage of the filter. The data widths at the intermediate stages are not changed.

Table 6. Output Rounding Options

Option	Description
Truncation	The CIC drops the LSBs. (Equivalent to rounding to minus infinity.)
Convergent rounding	Also known as unbiased rounding. Rounds to the nearest even number. If the most significant deleted bit is one, and either the least significant of the remaining bits or at least one of the other deleted bits is one, then one is added to the remaining bits.
Round up	Also known as rounding to plus infinity. Adds the MSB of the discarded bits for positive and negative numbers via the carry in.
Saturation	Puts a limit value (upper limit in the case of overflow, or lower limit in the case of negative overflow) at the output when the input exceeds the allowed range. The upper limit is +2n-1 and lower limit is –2n

3.3.3 Hogenauer Pruning

Hogenauer pruning uses truncation in intermediate stages with the retained number of bits decreasing monotonically from stage to stage. The total error introduced is still no greater than the quantization error introduced by rounding the full precision output. This technique helps to reduce the number of logic cells used by the filter and gives better performance.

The existing algorithms for computing the Hogenauer bit width growth for large *N* and *R* values are computationally expensive.

For more information about these algorithms, refer to U. Meyer-Baese, *Digital Signal Processing with Field Programmable Gate Arrays*, 2nd Edition, Spinger, 2004.

The CIC IP core has precalculated Hogenauer pruning bit widths. The CIC does not have to calculate Hogenauer pruning bit widths if you enable Hogenauer pruning for a decimation filter.

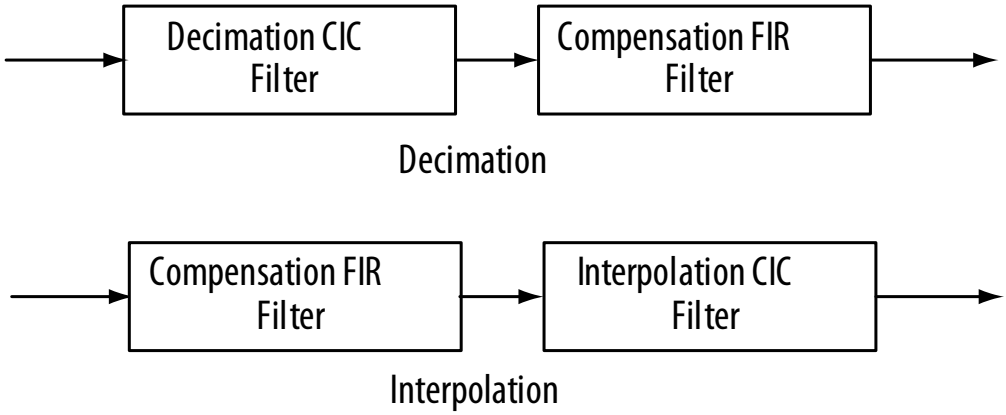
Note: Hogenauer pruning is only available to decimation filters when the selected output data width is smaller than the full output resolution data width.

3.4 FIR Filter Compensation Coefficients

CIC filters have a low-pass filter characteristic. Three parameters (the rate change factor *R*, the number of stages *N*, and the differential delay *M*) allow you to change the passband characteristics and aliasing or imaging rejection.

Typically, decimation or interpolation filtering applications require flat passband and narrow transition region filter performance. However, the CIC filter has drooping passband gains and wide transition regions. To overcome these problems connect the decimation or interpolation CIC filter to a compensation FIR filter, which narrows the output bandwidth and flattens the passband gain.

Figure 11. Using a CIC Compensation FIR Filter



You can use a frequency sampling method to determine the coefficients of a FIR filter that equalizes the undesirable passband droop of the CIC and construct an ideal frequency response.

Determine the ideal frequency response by sampling the normalized magnitude response of the CIC filter before inverting the response.

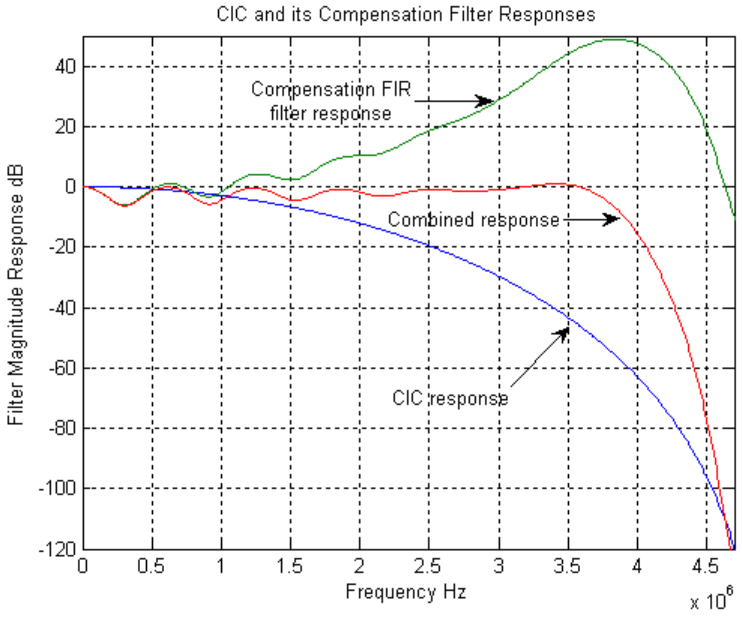
Generally, only equalize the response in the passband, but you can sample further than the passband to fine tune the cascaded response of the filter chain.

The CIC IP core generates a MATLAB script `<variation_name>_fir_comp_coeff.m` in the project directory. You can run this script in MATLAB to generate FIR coefficients that provide appropriate passband equalization. The generated coefficients are saved in a text file, for use by the FIR IP.

The MATLAB script requires the following parameters for the compensation FIR filter:

- *L*: FIR filter length, which is same as the number of taps or the number of coefficients
- F_S: FIR filter sample rate in Hz before decimation/interpolation
- F_C: FIR filter cutoff frequency in Hz
- *B*: Coefficient bit width if coefficients are written in fixed-point numbers

Figure 12. CIC and Compensation Filter Responses



Related Links

AN455: Understanding CIC Compensation Filters

3.5 CIC IP Core Parameters

Table 7. CIC IP Core Parameters

Parameter	Value	Description
Filter Specification		
Filter type	Decimator, Interpolator	Selects a decimator or interpolator.
Number of stages	1 to 12	Specifies the required number of stages.
Differential delay	1, 2	Specifies the differential delay in cycles.
Enable variable rate change factor	On or Off	Turn on to enable a variable rate change factor that you can change at runtime. When this option is on, the **Rate change factor** parameter is not available but you can specify minimum and maximum values.
Rate change factor	2 to 32000	Specifies the rate change factor.

continued...

Parameter	Value	Description
Number of interfaces	1 to 128	Specifies the number of MISO inputs or SIMO outputs The product of the **Number of interfaces** and the **Number of channels per interface** must be no more than 1024.
Number of channels per interface	1 to 1024	Specifies the number of channels per interface. The product of the **Number of interfaces** and the **Number of channels per interface** must be no more than 1024
Interface Specification		
Input data width	1 to 32	Specifies the input data width in bits.
Output Rounding Options	None, Truncation, Convergent rounding, Rounding up, Saturation, Hogenauer pruning	Selects the required rounding output mode. Select **None** for full output resolution. The saturation limit is the maximum value for overflow or the minimum value for negative overflow. Hogenauer pruning is available only when a **Decimator** filter type is selected in the **Architecture** page.
Output data width	1 to calculated maximum data width	Specifies the output data width in bits.
implementation Options		
Integrator data storage	Logic Element, Memory	Selects whether to implement the integrator data storage as logic elements or memory. The **Memory** option is available for integrator data storage when the **Number of channels per interface** is greater than 4.
RAM type of integrator data storage	AUTO, M9K, M10K, M20K, M144K, MLAB	When you select **Memory**, you can select the RAM type for integrator data storage. The **Memory** option is available for integrator data storage when the **Number of channels per interface** is greater than 4.
Differentiator data storage	Logic Element, Memory	Selects whether to implement the differentiator data storage as logic elements or memory. The **Memory** option is available for differentiator data storage when the product of the **Differential delay**, **Number of channels per interface** and **Number of interfaces** is greater than 4.
RAM type of differentiator data storage	AUTO, M9K, M10K, M20K, M144K, MLAB	When you select **Memory**, you can select the RAM type for differentiator data storage. The options available depend on the target device family. When AUTO is selected, the Quartus II software automatically selects the optimum RAM type for the currently selected device family.
Pipeline stages per integrator	—	Enter the pipeline stages per integrator. This option is available when the **Number of channels per interface** is greater than or equal to 2 (or greater than or equal to 6, when you select the **Memory** option for integrator data storage). Use this option for multichannel designs that have large input bit width and require high f_{MAX}, but not for designs targeting Cyclone devices.
Pipeline stages per integrator	1 to 4	Specifies the number of pipeline stages used by each integrator. Adding additional integrators can improve f_{MAX} but increases the resource utilization. The maximum number of pipeline stages depends on the number of channels and whether you select **Memory** or **Logic Cells** for integrator data storage. For **Memory**, the maximum number of pipeline stages equals the number of channels minus 5. For **Logic Cells**, the maximum number of pipeline stages equals the number of channels.

3.6 CIC IP Core Interfaces and Signals

Table 8. Avalon-ST Interface Parameters

All parameters not explicitly listed have undefined values.

Parameter Name	Value
READY_LATENCY	0
BITS_PER_SYMBOL	Data width
SYMBOLS_PER_BEAT	Single input, single output architectures, have one symbol per beat at the source and the sink. MISO architectures have *<number of interfaces>* symbols per beat at the sink, and a single symbol per beat at the source. SIMO architectures have *<number of interfaces>* symbols per beat at the source, and a single symbol per beat at the sink.
SYMBOL_TYPE	Signed
ERROR_DESCRIPTION	• 00: No error • 01: Missing `startofpacket` (SOP) • 10: Missing `endofpacket` (EOP) • 11: Unexpected EOP or any other error

Related Links

Avalon Interface Specifications
For more information about the Avalon-ST interface

3.6.1 Avalon-ST Interfaces in DSP IP Cores

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

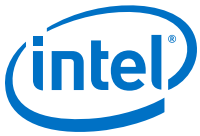
The input interface is an Avalon-ST sink and the output interface is an Avalon-ST source. The Avalon-ST interface supports packet transfers with packets interleaved across multiple channels.

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 can also support more complex protocols for burst and packet transfers with packets interleaved across multiple channels. The Avalon-ST interface inherently synchronizes multichannel designs, which allows you to achieve efficient, time-multiplexed implementations without having to implement complex control logic.

Avalon-ST 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 Links

Avalon Interface Specifications



3.6.2 CIC IP Core Signals

Table 9. CIC IP Core Signals

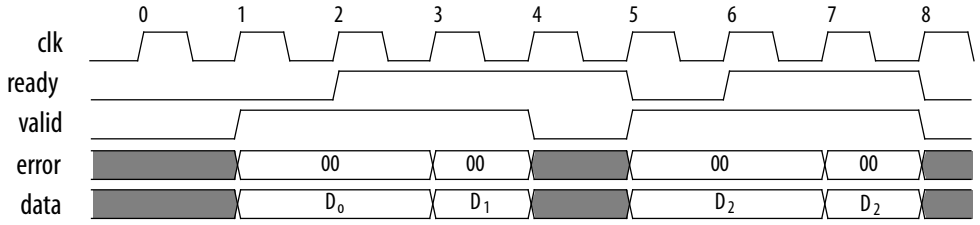
Signal	Direction	Description
av_st_in_data	Input	In Qsys systems, this Avalon-ST-compliant data bus includes all the Avalon-ST input data signals. For multi-interface designs Interface 0 is in the MSB; Interface *N* is the LSB.
clk	Input	Clock signal for all internal registers.
clken	Input	Optional top-level clock enable.
reset_n	Input	Active low reset signal. You must always reset the CIC MegaCore function before receiving data. If not, the CIC filter may produce unexpected results because of feedback signals.
in_data	Input	Sample input. For multiple input cases, the input data ports are in0_data, in1_data, and so on.
in_endofpacket	Input	Marks the end of the incoming sample group. For *N* channels, the end of packet signal must be high when the sample belonging to the last channel, channel *N*-1, is presented at in_data.
in_error	Input	Error signal indicating Avalon-ST protocol violations on input side: • 00: No error • 01: Missing start of packet • 10: Missing end of packet • 11: Unexpected end of packet Other types of error are also marked as 11.
in_ready	Output	Indicates when the IP core can accept data.
in_startofpacket	Input	Marks the start of the incoming sample group. The start of packet is interpreted as a sample from channel 0.
in_valid	Input	Asserted when data at in_data is valid. When in_valid is not asserted, processing is stopped until valid is re-asserted. If clken is 0, in_valid is not be asserted.
av_st_out_data	Output	In Qsys systems, this Avalon-ST-compliant data bus includes all the Avalon-ST output data signals. For multi-interface designs Interface 0 is in the MSB; Interface *N* is the LSB.
out_channel	Output	Specifies the channel whose result is presented at out_data.
out_data	Output	Filter output. The data width depends on the parameter settings. For multiple output cases, the output data ports are named as out0_data, out1_data, and so on.
out_endofpacket	Output	Marks the end of the outgoing result group. If '1', a result corresponding to channel N-1 is output, where N is the number of channels.
out_error	Output	Error signal indicating Avalon-ST protocol violations on source side: • 00: No error • 01: Missing start of packet • 10: Missing end of packet • 11: Unexpected end of packet Other types of errors may also be marked as 11.
out_ready	Input	Asserted by the downstream module if it is able to accept data.

continued...

Signal	Direction	Description
out_startofpacket	Output	Marks the start of the outgoing result group. If '1', a result corresponding to channel 0 is output.
out_valid	Output	Asserted by the IP core when there is valid data to output.
rate	Input	This signal is available when the variable rate change factor option is enabled. You can use it to change the decimation or interpolation rate during run time. It has the size Ceil(log_2(maximum rate)).

3.6.3 Avalon-ST Interface Data Transfer Timing

Figure 13. Avalon-ST Interface Timing with READY_LATENCY=0



The source provides data and asserts `valid` on cycle 1, even though the sink is not ready. The source waits until cycle 2, when the sink does assert `ready`, before moving onto the next data cycle. In cycle 3, the source drives data on the same cycle and because the sink is ready to receive it, the transfer occurs immediately. In cycle 4, the sink asserts `ready`, but the source does not drive valid data.

3.6.4 Packet Data Transfers

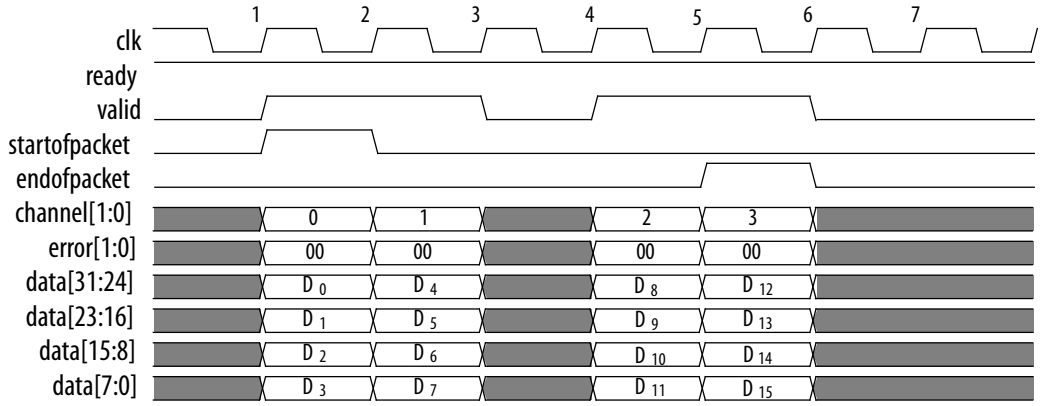
A beat is the transfer of one unit of data between a source and sink interface. This unit of data may consist of one or more symbols and makes it is possible to support modules that convey more than one piece of information about each valid cycle.

Packet data transfers are used for multichannel transfers. Two additional signals (`startofpacket` and `endofpacket`) are defined to implement the packet transfer.

The multiple symbols per beat scenario applies to both the sink interface on MISO CIC filters and the source interface of SIMO CIC filters. All other interfaces operate with a single symbol per beat, but the interfaces also support multiple channels using packets.

Figure 14. Packet Data Transfer

Four symbols are transferred on each beat. The data transfer occurs on cycles 1, 2, 4, and 5, when both ready and valid are asserted.



During cycle 1, the CIC IP core asserts `startofpacket`, and transfers the first four bytes of packet. During cycle 5, the CIC IP core asserts `endofpacket` indicating that this is the end of the packet. The `channel` signal indicates the channel index associated with the data. For example, on cycle 1, the data D_0, D_1, D_2, and D_3 associated with channel 0 are available.

A CIC IP Core User Guide Document Archives

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

IP Core Version	User Guide
14.1	CIC IP Core User Guide v14.1

5 Document Revision History

CIC IP Core User Guide revision history.

Table 10.

Date	Version	Changes
November 2017	2017.11.06	• Removed product ID and vendor ID. • Added support for Intel Cyclone 10 and Intel Stratix 10 devices • Corrected `av_st_in_data` signal direction to input.
2014.12.15	14.1	• Added final support for Arria 10 devices • Reordered parameters tables to match wizard
August 2014	14.0 Arria 10 Edition	• Added support for Arria 10 devices. • Added new av_st_in_data and av_st_out_data bus descriptions. • Added Arria 10 generated files description. • Removed table with generated file descriptions.
June 2014	14.0	• Removed support for Cyclone III and Stratix III devices • Added instructions for using IP Catalog
November 2013	13.1	• Removed support for the following devices: — Arria — Cyclone II — HardCopy II, HardCopy III, and HardCopy IV — Stratix, Stratix II, Stratix GX, and Stratix II GX • Added full support for the following devices: — Arria V — Stratix V
November 2012	12.1	Added support for Arria V GZ devices.