

HB0562
Handbook
CoreCIC v2.1

01 2018



Microsemi makes no warranty, representation, or guarantee regarding the information contained herein or the suitability of its products and services for any particular purpose, nor does Microsemi assume any liability whatsoever arising out of the application or use of any product or circuit. The products sold hereunder and any other products sold by Microsemi have been subject to limited testing and should not be used in conjunction with mission-critical equipment or applications. Any performance specifications are believed to be reliable but are not verified, and Buyer must conduct and complete all performance and other testing of the products, alone and together with, or installed in, any end-products. Buyer shall not rely on any data and performance specifications or parameters provided by Microsemi. It is the Buyer's responsibility to independently determine suitability of any products and to test and verify the same. The information provided by Microsemi hereunder is provided "as is, where is" and with all faults, and the entire risk associated with such information is entirely with the Buyer. Microsemi does not grant, explicitly or implicitly, to any party any patent rights, licenses, or any other IP rights, whether with regard to such information itself or anything described by such information. Information provided in this document is proprietary to Microsemi, and Microsemi reserves the right to make any changes to the information in this document or to any products and services at any time without notice.



Microsemi Corporate Headquarters
One Enterprise, Aliso Viejo,
CA 92656 USA
Within the USA: +1 (800) 713-4113
Outside the USA: +1 (949) 380-6100
Sales: +1 (949) 380-6136
Fax: +1 (949) 215-4996
E-mail: sales.support@microsemi.com
www.microsemi.com

About Microsemi

Microsemi Corporation (Nasdaq: MSCC) offers a comprehensive portfolio of semiconductor and system solutions for aerospace & defense, communications, data center and industrial markets. Products include high-performance and radiation-hardened analog mixed-signal integrated circuits, FPGAs, SoCs and ASICs; power management products; timing and synchronization devices and precise time solutions, setting the world's standard for time; voice processing devices; RF solutions; discrete components; enterprise storage and communication solutions; security technologies and scalable anti-tamper products; Ethernet solutions; Power-over-Ethernet ICs and midspans; as well as custom design capabilities and services. Microsemi is headquartered in Aliso Viejo, Calif., and has approximately 4,800 employees globally. Learn more at www.microsemi.com.

©2018 Microsemi Corporation. All rights reserved. Microsemi and the Microsemi logo are registered trademarks of Microsemi Corporation. All other trademarks and service marks are the property of their respective owners.

Revision History

The revision history describes the changes that were implemented in the document. The changes are listed by revision, starting with the most current publication.

1.1 Release 2.0

Updated this document for CoreCIC v2.1

1.2 Release 1.0

Revision 1.0 was the first publication of this document. Created for CoreCIC v2.0

Contents

- Revision History..... 3
 - 1.1 Release 2.0..... 3
 - 1.2 Release 1.0..... 3
- 2 Introduction 8
 - 2.1 General Description 8
 - 2.2 Key Features..... 10
 - 2.3 Supported Families 10
 - 2.4 Core Version..... 10
 - 2.5 Utilization and Performance 11
- 3 Theory of Operations 16
 - 3.1 Moving Average 16
 - 3.1.1 CIC Filter Structures 17
 - 3.2 Multiple Channel Support..... 19
 - 3.2.1 Multiple Interfaces 19
 - 3.2.2 Comb and Integrator Time Share 20
 - 3.3 Bit Growth..... 21
- 4 Interface Description..... 22
 - 4.1 Parameters and Generics..... 22
 - 4.2 Ports..... 23
 - 4.2.1 Decimator Interface..... 27
 - 4.2.2 Interpolator Interface 28
- 5 Implementation Details 32
 - 5.1 Reset 32
 - 5.2 Latency..... 32
 - 5.3 Multiple Interface Connections 32
 - 5.4 Variable Rate..... 33
 - 5.4.1 RAM Block Use 34
 - 5.5 Decimation Filter Timing..... 34
 - 5.6 Interpolation Filter Timing 37
- 6 Tool Flow 40
 - 6.1 License 40
 - 6.1.1 RTL..... 40
 - 6.2 SmartDesign..... 40
 - 6.3 Simulation Flows..... 41
 - 6.4 Synthesis in Libero 41
 - 6.5 Place-and-Route in Libero..... 42

7	Testbench	43
7.1	User Testbench	43
8	References.....	44
9	Ordering Information	45
9.1	Ordering Codes	45

List of Figures

Figure 1 CIC Filter Application Examples	9
Figure 2 Moving Average Block Diagram	16
Figure 3 Recursive Moving Average	16
Figure 4 Non-optimized Boxcar Decimator Filter	17
Figure 5 Boxcar Interpolation Filter (Non-optimized)	17
Figure 6 One-Stage CIC Decimation Filter	17
Figure 7 N-Stage Decimation CIC filter	18
Figure 8 One-Stage CIC Interpolation Filter	18
Figure 9 N-Stage Interpolation CIC Filter	18
Figure 10 Three Interfaces, One Channel per Interface CIC Decimator	19
Figure 11 Four Interfaces, One Channel per Interface CIC Interpolator	20
Figure 12 Three Interfaces, Two Channels per Interface CIC Decimator	20
Figure 13 Four Interfaces, Two Channels per Interface CIC Interpolator	21
Figure 14 I/O Ports	23
Figure 15 Fixed Rate Single Channel CIC Filter	25
Figure 16 Variable Rate Single Channel CIC Filter	26
Figure 17 Fixed Rate Multiple Channel Mode	26
Figure 18 Using RFD_PILOT Signal	29
Figure 19 Example of Interpolator Input Signals at DIN_VALID Always Active	30
Figure 20 Example of Interpolator Input Signals at DIN_VALID Active Every Other Clock	30
Figure 21 I/O Channel Mapping for an Interpolator Example	31
Figure 22 Data Concatenation for Multiple Interface Decimator	33
Figure 23 Output Concatenation of a Multiple Interface Interpolator	33
Figure 24 Decimation CIC Timing - Single Channel, Data Permanently Valid	34
Figure 25 Decimation CIC Timing - Single Channel, Data Coming Every Other Clock	35
Figure 26 Decimation CIC Timing - Three Time Share Channels, Data Permanently Valid	35
Figure 27 Decimation CIC Timing - Two Interfaces, Data Permanently Valid	36
Figure 28 6-Channel Decimator Timing - Two Interfaces Three Time Share Channels Each, Data Always Valid	36
Figure 29 Input Data Shift Between Interfaces IF0 and IF1	37
Figure 30 Interpolation CIC Timing - Single Channel, Data Permanently Valid	37
Figure 31 Interpolation CIC Timing - Single Channel, Data Coming Every Other Clock	38
Figure 32 Interpolation CIC Timing - Three Time Share Channels, Data Permanently Valid	39
Figure 33 Interpolation CIC Timing - Two Interfaces, Data Permanently Valid	39
Figure 34 CoreCIC Full I/O View	40
Figure 35 SmartDesign CoreCIC Configuration Window	41
Figure 36 CoreCIC User Testbench Simplified Block Diagram	43

List of Tables

Table 1 CIC Decimator Resource Utilization and Performance	11
Table 2 CIC Interpolator Resource Utilization and Performance.....	12
Table 3 CIC Decimator Resource Utilization and Performance	13
Table 4 CIC Interpolator Resource Utilization and Performance.....	14
Table 5 CoreCIC Parameter and Generic Descriptions	22
Table 6 CIC Filter In or Out Signals.....	24
Table 7 Ordering Codes	45

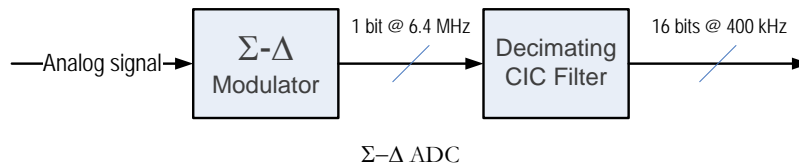
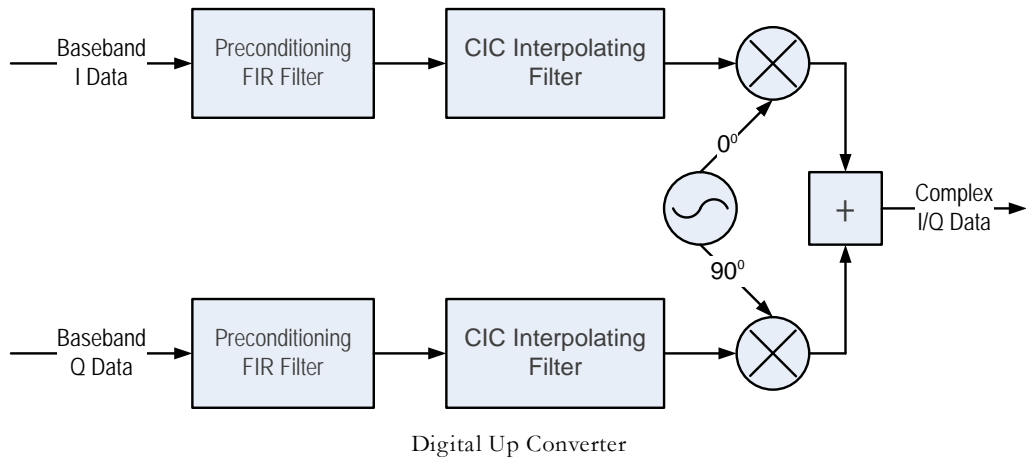
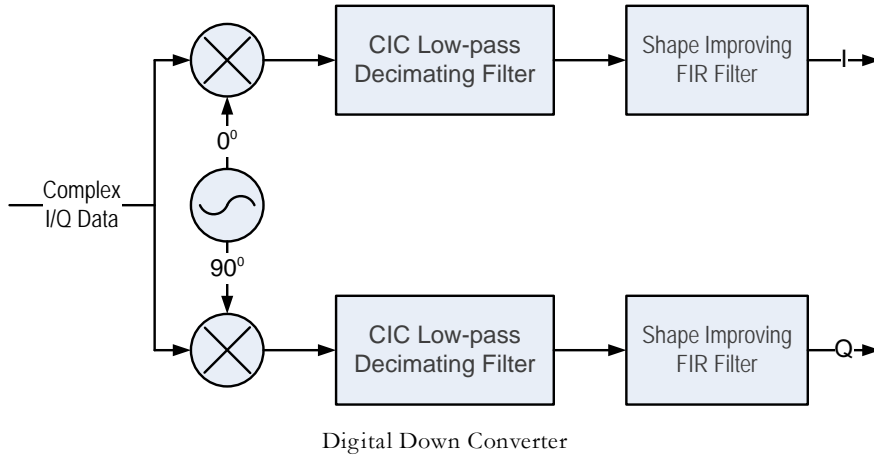
2 Introduction

2.1 General Description

Microsemi® CoreCIC IP is a highly configurable RTL generator for the decimation or interpolation cascaded integrator-comb (CIC) filters. The CIC filters are widely used in multi-rate signal processing, particularly in up-converters and down-converters, modulators and demodulators, sigma-delta analog to digital converters, and so on. These filters are popular in decimation and interpolation filters where substantial rate change factor is required. The CIC filters provide a linear phase response.

The following figure shows a few application examples of the CIC filter.

Figure 1 CIC Filter Application Examples



2.2 Key Features

CoreCIC supports decimation and interpolation filter types. Following are the key features of CoreCIC filter:

- Fixed or programmable rate change from 2 to 1024
- One to eight integrator-comb stages
- Comb differential delay of one or two
- Signed 2's complement input data
- Input data width from 1 to 32 bits
- Output data width up to 100 bits
- Choice of output data truncation and two rounding types
- Optional Hogenauer pruning
- Support for up to 64 channels

2.3 Supported Families

CoreCIC supports the following families:

- PolarFire®
- RTG4™
- IGLOO®2
- SmartFusion®2

2.4 Core Version

This handbook applies to CoreCIC v2.1.

2.5 Utilization and Performance

The resource utilization and core performance are shown on [Table 1](#) and [Table 2](#) for SmartFusion2 M2S050 device, speed grade -1.

Table 1 CIC Decimator Resource Utilization and Performance

Configuration						Resource Utilization				Maximum Clock Rate, MHz
Input Data Width	Output Data Width	Rate Change Factor	Number of Stages	Number of IF	Channels per IF	4LUT	DFF	RAM 64x18	RAM 1K18	
RAM blocks are not used; Differential Delay = 1										
16	16	4	3	1	1	751	730	0	0	385
16	16	4	3	4	1	1,905	1,902	0	0	345
16	16	4	3	1	5	1,452	1,706	0	0	314
16	16	40	3	1	1	1,063	1,022	0	0	341
16	16	40	3	4	1	2,713	2,702	0	0	331
16	16	40	3	1	5	2,030	2,388	0	0	313
16	16	4	2	1	1	511	495	0	0	386
16	16	4	2	4	1	1,241	1,224	0	0	366
16	16	4	2	1	5	956	1,023	0	0	358
RAM blocks are not used; Differential Delay = 2										
16	16	4	3	1	5	1,989	2,217	0	0	317
16	16	4	5	1	1	1,849	1,959	0	0	336
16	16	4	5	4	1	4,749	4,803	0	0	306
16	16	4	5	1	5	3,880	4,506	0	0	304
Use RAM blocks is On; Maximum MicroRAM Depth=64; Differential Delay = 1										
12	16	4	3	4	14	2,232	2,090	15	0	250
30	18	4	3	4	14	4,238	4,008	30	0	250
Use RAM blocks is On; Maximum MicroRAM Depth=0; Differential Delay = 1										
12	16	4	3	4	16	2,244	2,096	0	15	287
30	18	4	3	4	16	3,712	3,476	0	15	286
12	16	4	2	4	16	1,466	1,355	0	10	317
30	18	4	2	4	16	2,479	2,299	0	10	317
Use RAM blocks is On; Maximum MicroRAM Depth=0; Differential Delay = 2										
12	16	4	2	4	16	1,558	1,433	0	10	315
30	18	4	2	4	16	2,575	2,376	0	10	308

Table 2 CIC Interpolator Resource Utilization and Performance

Configuration						Resource Utilization				Maximum Clock Rate, MHz
Input Data Width	Output Data Width	Rate Change Factor	Number of Stages	Number of IF	Channels per IF	4LUT	DFF	RAM 64x18	RAM 1K18	
RAM blocks are not used; Differential Delay = 1										
16	16	4	3	1	1	639	604	0	0	383
16	16	4	3	4	1	1,756	1,756	0	0	345
16	16	4	3	1	5	1,335	1,554	0	0	330
16	16	40	3	1	1	716	667	0	0	333
16	16	40	3	4	1	1,905	1,886	0	0	327
16	16	40	3	1	5	1,447	1,703	0	0	320
16	16	4	2	1	1	441	410	0	0	375
16	16	4	2	4	1	1,179	1,168	0	0	329
16	16	4	2	1	5	891	1,002	0	0	324
RAM blocks are not used; Differential Delay = 2										
16	16	4	3	1	5	1,622	1,870	0	0	316
16	16	4	5	1	1	1,335	1,377	0	0	334
16	16	4	5	4	1	3,703	3,731	0	0	325
16	16	4	5	1	5	2,927	3,478	0	0	301
Use RAM blocks is On; Max MicroRAM Depth=64; Differential Delay = 1										
11	16	4	3	4	14	2,246	2,118	15	0	250
29	18	4	3	4	14	4,157	3,996	30	0	250
Use RAM blocks is On; Max MicroRAM Depth=0; Differential Delay = 1										
11	16	4	3	4	16	2,257	2,124	0	15	327
29	18	4	3	4	16	3,627	3,462	0	15	307
11	16	4	2	4	16	1,500	1,392	0	10	340
29	18	4	2	4	16	2,441	2,310	0	10	320
Use RAM blocks is On; Max MicroRAM Depth=0; Differential Delay = 2										
12	16	4	2	4	16	1,554	1,445	0	10	332
30	18	4	2	4	16	2,489	2,351	0	10	312

The results shown on [Table 1](#) and [Table 2](#) were achieved at the Operating Conditions COM. The following tools were used:

- Libero v11.4.
- SynplifyPro H-2013.03M-SP1-1.

The other core parameters were set as follows:

- Enable Variable Rate = No

- Apply Hogenauer Pruning = No
- Rounding Mode = Truncation

The utilization and core performance data are shown in [Table 3](#) and [Table 4](#) for PolarFire (MPF300T) Device family. The overall device utilization and performance of the core is system dependent.

Table 3 CIC Decimator Resource Utilization and Performance

Configuration						Resource Utilization				Maximum Clock Rate, MHz
Input Data Width	Output Data Width	Rate Change Factor	Number of Stages	Number of IF	Channels per IF	4LUT	DFF	RAM 64x12	RAM 1K20	
RAM blocks are not used; Differential Delay = 1										
16	16	4	3	1	1	690	668	0	0	303.8
16	16	4	3	4	1	1,814	1804	0	0	322.7
16	16	4	3	1	5	1,431	1661	0	0	310.6
16	16	40	3	1	1	996	960	0	0	232.5
16	16	40	3	4	1	2,623	2612	0	0	240.9
16	16	40	3	1	5	2,009	2343	0	0	240.9
16	16	4	2	1	1	450	434	0	0	307.7
16	16	4	2	4	1	1,153	1132	0	0	328.8
16	16	4	2	1	5	910	1048	0	0	306.6
RAM blocks are not used; Differential Delay = 2										
16	16	4	3	1	5	1,984	2225	0	0	215.6
16	16	4	5	1	1	1,805	1897	0	0	208.2
16	16	4	5	4	1	4,677	4706	0	0	205.1
16	16	4	5	1	5	3,918	4461	0	0	224.6
Use RAM blocks is On; Maximum MicroRAM Depth=64; Differential Delay = 1										
12	16	4	3	4	14	1955	1965	30	0	171.9
30	18	4	3	4	14	3507	3153	45	0	171.9
Use RAM blocks is On; Maximum MicroRAM Depth=0; Differential Delay = 1										
12	16	4	3	4	16	2,148	2019	0	15	178.6
30	18	4	3	4	16	3,519	3321	0	15	178.6
12	16	4	2	4	16	1,377	1278	0	10	179.3
30	18	4	2	4	16	2,316	2,148	0	10	178.6
Use RAM blocks is On; Maximum MicroRAM Depth=0; Differential Delay = 2										
12	16	4	2	4	16	1,463	1355	0	10	178.5
30	18	4	2	4	16	2,402	2225	0	10	178.0

Table 4 CIC Interpolator Resource Utilization and Performance

Configuration						Resource Utilization				Maximum Clock Rate, MHz
Input Data Width	Output Data Width	Rate Change Factor	Number of Stages	Number of IF	Channels per IF	4LUT	DFF	RAM 64x12	RAM 1K20	
RAM blocks are not used; Differential Delay = 1										
16	16	4	3	1	1	623	588	0	0	324.5
16	16	4	3	4	1	1685	1685	0	0	315.7
16	16	4	3	1	5	1308	1510	0	0	233.1
16	16	40	3	1	1	754	716	0	0	217.3
16	16	40	3	4	1	1873	1869	0	0	207.9
16	16	40	3	1	5	1495	1726	0	0	204.5
16	16	4	2	1	1	421	388	0	0	333.9
16	16	4	2	4	1	1,105	1094	0	0	316.5
16	16	4	2	1	5	848	960	0	0	233.1
RAM blocks are not used; Differential Delay = 2										
16	16	4	3	1	5	1608	1828	0	0	233.1
16	16	4	5	1	1	1,335	1375	0	0	302.9
16	16	4	5	4	1	3,647	3674	0	0	213.6
16	16	4	5	1	5	2,961	3435	0	0	233.1
Use RAM blocks is On; Max MicroRAM Depth=64; Differential Delay = 1										
11	16	4	3	4	14	1939	1906	29	0	174.8
29	18	4	3	4	14	3460	3536	45	0	174.8
Use RAM blocks is On; Max MicroRAM Depth=0; Differential Delay = 1										
11	16	4	3	4	16	2,137	2032	0	15	184.3
29	18	4	3	4	16	3,472	3344	0	15	181.7
11	16	4	2	4	16	1394	1,306	0	10	181.7
29	18	4	2	4	16	2300	2194	0	10	181.7
Use RAM blocks is On; Max MicroRAM Depth=0; Differential Delay = 2										
12	16	4	2	4	16	1,492	1403	0	10	181.6
30	18	4	2	4	16	2,389	2283	0	10	181.6

Note: The data in this table is achieved using typical synthesis and layout settings. Frequency (in MHz) was set to 100 and speed grade was -1.

The results shown in [Table 3](#) and [Table 4](#) were achieved at the operating Conditions COM. The following tools were used:

- Libero PolarFire v1.1
- Synplify Pro ME L-2016.09M-G5

The core parameter were set as follows:

- Enable Variable rate = No
- Apply Hogenauer Pruning = No
- Rounding Mode = Truncation.

3 Theory of Operations

3.1 Moving Average

The moving average is one of the most common filters in digital signal processing (DSP) and also called a boxcar filter. As the name implies, it averages a number of L input samples to generate each output sample:

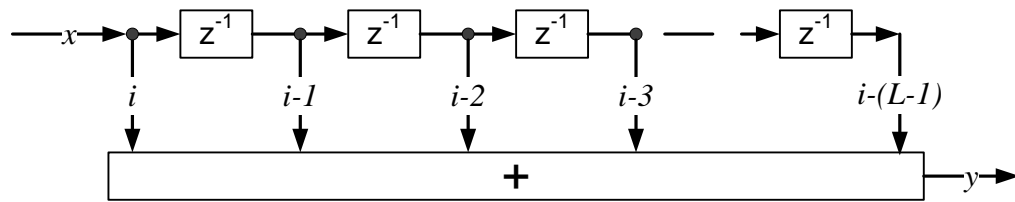
$$y(k) = \frac{1}{L} \sum_{j=0}^{L-1} X(k+j)$$

EQ1

The filter calculates a time domain convolution between an input signal and a boxcar function. A frequency domain counterpart for the boxcar is $\text{sinc}(X)/X$, which describes the frequency response of the filter. Hence, sinc filter is another name for moving average and CIC filters.

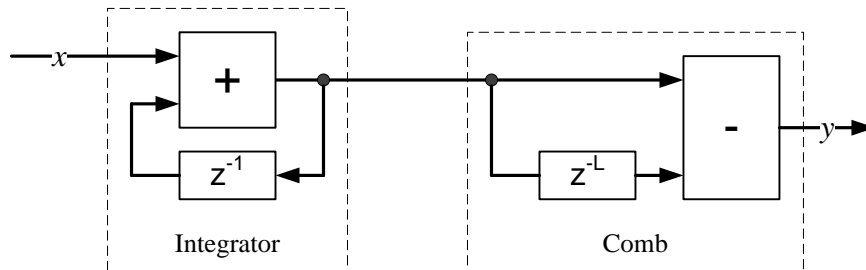
The following figure shows block diagram of the moving average filter except dividing the output by L.

Figure 2 Moving Average Block Diagram



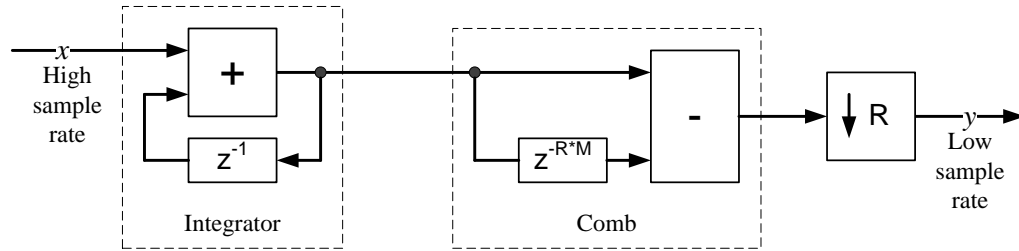
The same result can be obtained by using a recursive form of the boxcar filter as shown in the following figure. An integrator accumulates input samples, while a comb adds a new accumulated sum and subtracts a delayed by L version of the sum.

Figure 3 Recursive Moving Average



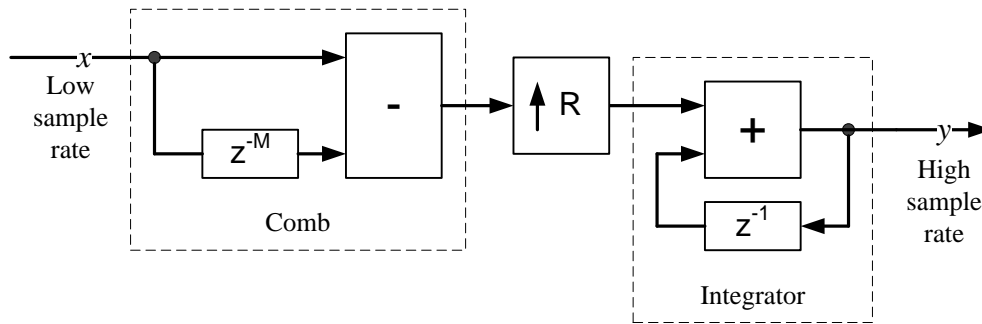
A recursive boxcar decimation filter adds a down-sampler by R where R is the decimation ratio. The following figure shows a non-optimized decimation filter assuming the delay $L = M * R$, where M is a constant coefficient called differential delay. $M = 1$ or 2 . The down-sampler by R $\downarrow R$ discards R-1 output samples from every R samples.

Figure 4 Non-optimized Boxcar Decimator Filter



The following figure shows a non-optimized interpolation filter structure.

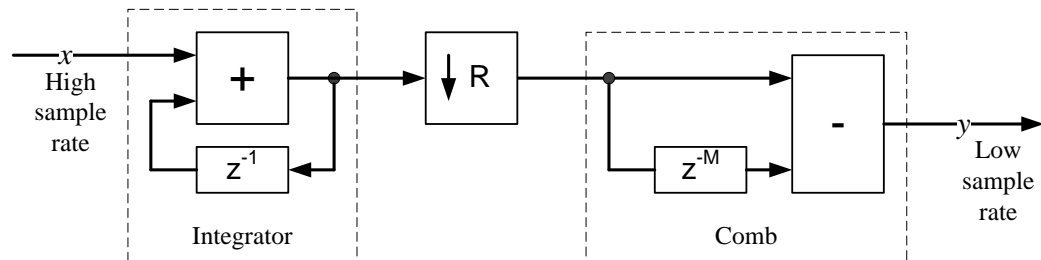
Figure 5 Boxcar Interpolation Filter (Non-optimized)



3.1.1 CIC Filter Structures

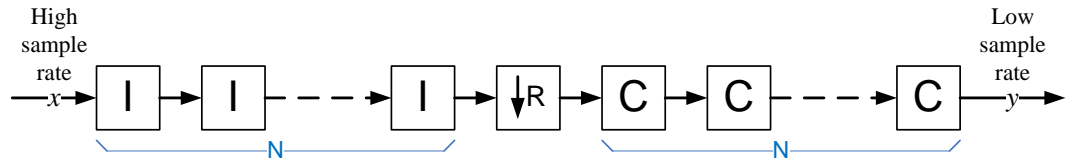
A CIC decimation filter is based on an optimized structure obtained from the non-optimized boxcar decimator block diagram as shown in Figure 4 (see References, 1 and 2). The following figure shows a one-stage CIC filter block diagram after placing the downsampler between the integrator and comb filter.

Figure 6 One-Stage CIC Decimation Filter



The following figure shows a cascaded N-stage decimation CIC filter where the integrators and combs are denoted as I and C, respectively. The filter contains N integrators and N combs. The frequency response is same as N cascaded boxcar filters.

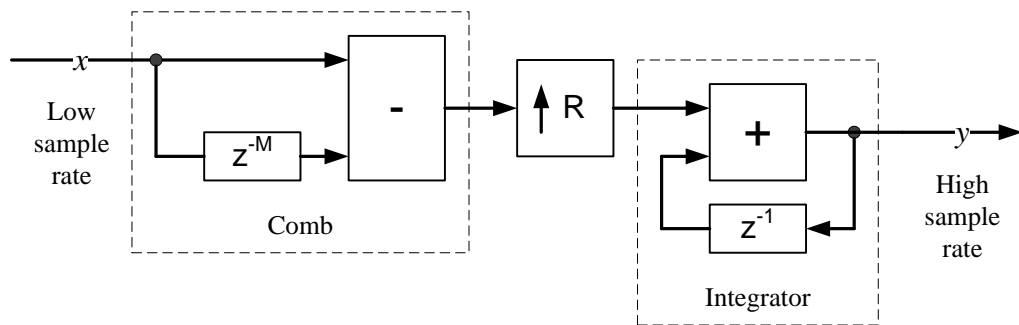
Figure 7 N-Stage Decimation CIC filter



An interpolation CIC filter optimizes the structure, refer to [Figure 5](#).

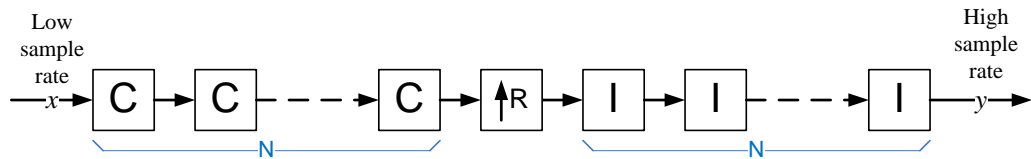
The following figure shows a one-stage interpolation CIC filter.

Figure 8 One-Stage CIC Interpolation Filter



The following figure shows an N-stage interpolation CIC filter.

Figure 9 N-Stage Interpolation CIC Filter



The CIC filter implementations are preferred over other rate changing filters because they only use adders and delays but not multipliers.

If the impulse response of a one-stage boxcar filter has a width of $M * R$, the N cascades of identical boxcar filters have the overall Impulse Response Width of $(M * R - 1) * N + 1$.

EQ3 describes the system response of the CIC filter (see Reference 3):

$$H(z) = \left[\sum_{k=0}^{M \cdot R - 1} z^{-k} \right]^N$$

EQ2

3.2 Multiple Channel Support

3.2.1 Multiple Interfaces

CIC filters of same configuration can share adders, subtractors, and other resources to process more channels, if the required data processing rate is relatively slow. This rate is always low for the CIC filter comb section where comb processing rate is R times lower than the processing rate of the integrator section, refer to [Figure 7](#) and [Figure 9](#). Thus a single comb can support up to R integrator sections. To fully utilize the comb section throughput, a CIC filter needs R integrator sections, each processing data at high sampling rate. In this handbook, such resource sharing is called comb sharing.

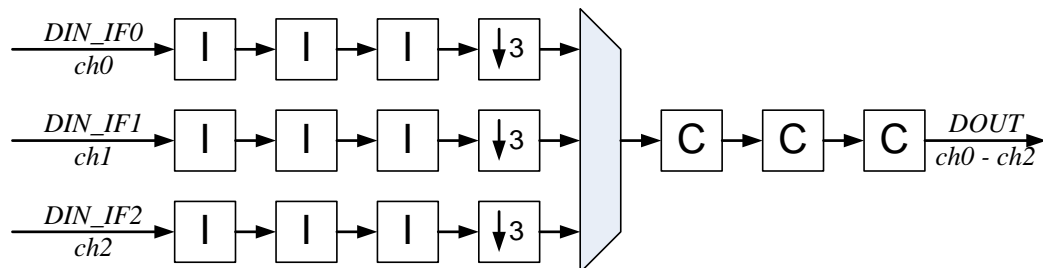
The following figure shows a 3-channel filter with a decimation factor, $R_RATE = 3$. The CIC filter has three integrator sections, and is capable of filtering three input samples per clock. The data samples from the channels are fed at each clock interval.

The downsampled data from integrator are multiplexed so that each channel data occupies a separate time slot of one clock cycle. The comb section utilizes the three clock intervals (obtained due to reduction in the rate by a factor of 3) to process the integrated data samples.

The following figure shows a structure that has three interfaces namely, DIN_IF0 , DIN_IF1 , DIN_IF2 and one output for the time-multiplexed output samples. For the core to generate the structure of three interfaces, one channel per interface CIC decimator, set the parameters as follows:

- $IF_NUM = 3$
- $CLK_PER_SAMPLE = 1$

Figure 10 Three Interfaces, One Channel per Interface CIC Decimator

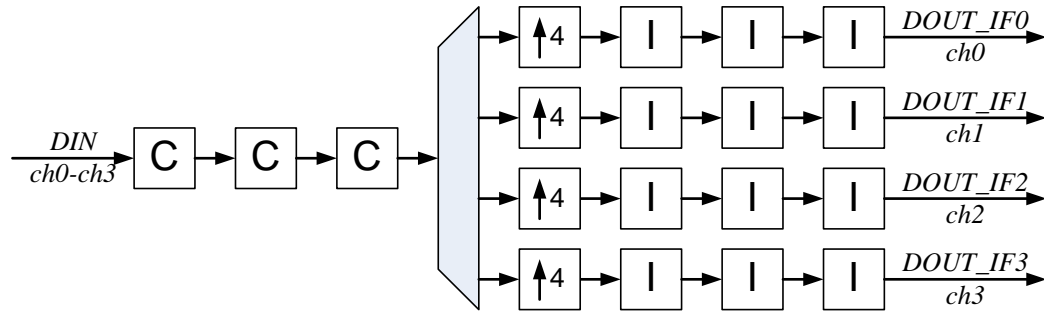


Similarly, the interpolation CIC filter can apply comb sharing to process more channels. The following figure shows an example of a 4-channel filter with interpolation rate factor of four. The low rate input channels are time-multiplexed, and the structure provides four output interfaces $DOUT_IF0$ to $DOUT_IF3$.

Set the parameter as:

- $CLK_PER_SAMPLE = 1$
- $IF_NUM = 4$

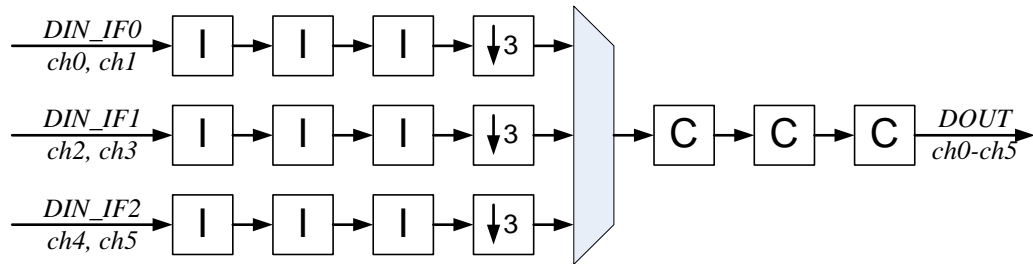
Figure 11 Four Interfaces, One Channel per Interface CIC Interpolator



3.2.2 Comb and Integrator Time Share

The input samples of each decimation channel can arrive at lower rate than the field programmable gate array (FPGA) clock rate, that is, there are idle clock intervals in between the samples of each input channel. Since they arrive at a lower rate, the integrator sections can be time shared as well. This handbook refers this as time sharing. The following figure shows a multi-channel CIC decimation example where input samples of every channel are separated by one idle clock cycle. If this instance is created when $CLK_PER_SAMPLE = 2$, each integrator can use two clock intervals, which is adequate to process two channels. The comb section has now six clock intervals, adequate to process all six channels. Every interface supports two channels totaling at $CLK_PER_SAMPLE * IF_NUM = 6$ channels.

Figure 12 Three Interfaces, Two Channels per Interface CIC Decimator

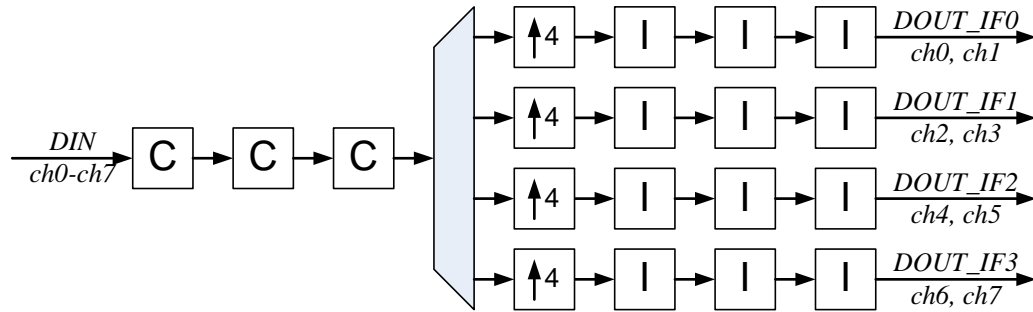


The time and comb sharing are independent of each other and can be combined as desired in decimation and interpolation CIC filter types. The following figure shows the CIC structure with four output interfaces and two channels per interface, totaling at $CLK_PER_SAMPLE * IF_NUM = 8$ channels.

The total number of channels processed by the CIC filter equals the product of the parameters IF_NUM and CLK_PER_SAMPLE . If the product is more than the actual number of channels to be processed, use dummy channels.

CoreCIC automatically identifies multiple channel filters, if the product of $IF_NUM * CLK_PER_SAMPLE > 1$ and implements time and/or comb sharing based on the CLK_PER_SAMPLE and IF_NUM parameter values.

Figure 13 Four Interfaces, Two Channels per Interface CIC Interpolator



3.3 Bit Growth

Data path width of the CIC filter needs to grow from input to output to support valid processing. The core automatically provides the required bit width for every component of the design. Internally, the processing results are calculated with full precision. You can limit the output bit width by entering a desired value in the Output Data Width field of the IP user interface and select truncation or rounding from **Rounding Mode** drop-down list. The core supports truncation, round away from 0 and convergent rounding (rounding to nearest even number). Either type of rounding or truncation applies to a full precision internal result.

CoreCIC in Decimation mode also supports optional Hogenauer pruning, which limits the intermediate stage bit widths as described in the article by E.Hogenauer. The core does not calculate full precision results even internally but limits their bit width in accordance with the approach by Hogenauer. When the reduced precision internal result still exceeds the desirable output bit width, the truncation or rounding applies similar to the full precision results. The Hogenauer pruning may reduce resource utilization.

4 Interface Description

4.1 Parameters and Generics

The following table describes the CoreCIC parameters (Verilog) or generics (VHDL). All the parameters and generics are positive integer type numbers.

Table 5 CoreCIC Parameter and Generic Descriptions

Parameter Name	Valid Range	Default	Description
N_STAGES	1-8	3	Number of cascaded stages. There is always equal number of integrator and comb stages. Any selected number N_STAGES means the CIC filter has N_STAGES of integrators and N_STAGES of combs.
M_DLY	1, 2	1	Differential comb delay M.
CIC_TYPE	0, 1	0	0: Decimation CIC filter. 1: Interpolation CIC filter.
DIN_WIDTH	1-32	18	Input data bit width.
VAR_RATE	0, 1	0	0: Fixed rate change factor. 1: Variable programmable rate change factor.
R_RATE	2-1024	4	Rate change factor R. If variable rate change is disabled VAR_RATE = 0, the R_RATE defines fixed rate change factor. Otherwise it defines an initial value of the variable factor set upon core configuration.
VAR_R_MIN	2-1023	4	Minimal variable rate change factor. Available only if VAR_RATE is set.
VAR_R_MAX	3-1024	5	Maximal variable rate change factor. Available only if VAR_RATE is set.
PRUNE	0-1	0	1: Apply Hogenauer pruning across decimation filter stages. Available for decimation filter only, that is when CIC_TYPE = 0. 0: Do not apply pruning.
DOUT_WIDTH	2-100	18	Output data bit width. If the width set is less than full output bit width, the core truncates or rounds the filtered data to the DOUT_WIDTH.
QUANTIZATION	0-2	0	Output data quantization mode. Applies when the output data bit width selected is less than the automatic full precision processing bit width: 0: Truncation mode. 1: Round away from 0. 2: Convergent rounding.
IF_NUM	1-32	1	Number of interfaces. For decimation filter, it is a number of input interfaces; for interpolation filter it is a number of output interfaces. Each interface is capable of accepting or generating one or more data channels. In other words, IF_NUM indicates a number of physical integrator cascades implemented. IF_NUM cannot exceed the rate change factor R_RATE, that is $IF_NUM \leq R_RATE$. In the case of variable rate (VAR_RATE=1), the IF_NUM cannot exceed the value of VAR_R_MIN, that is $IF_NUM \leq VAR_R_MIN$. Since the channel number of a multichannel filter equals $IF_NUM * CLK_PER_SAMPLE$, and the maximum channel count is 64, the IF_NUM must not exceed the value of $64/CLK_PER_SAMPLE$.

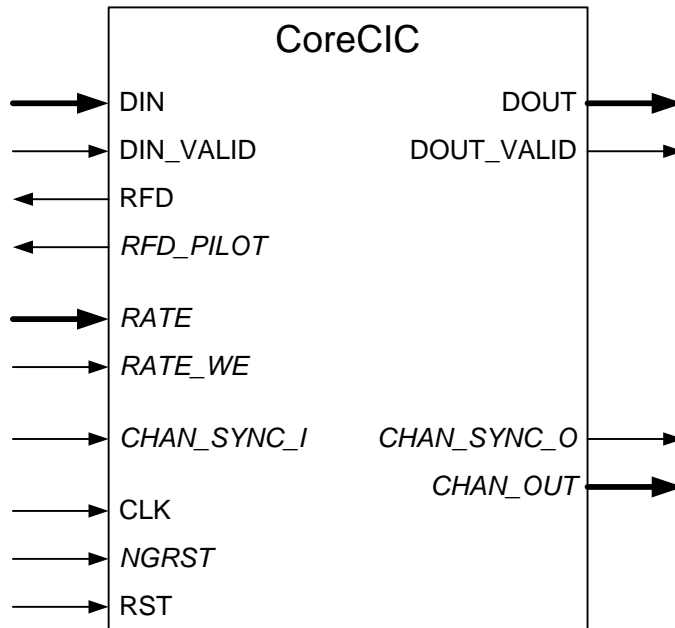
Table 5 CoreCIC Parameter and Generic Descriptions

Parameter Name	Valid Range	Default	Description
CLK_PER_SAMPLE	1-64	1	Number of FPGA clock cycles per sample of any channel. Indicates a number of time-multiplexed channels per interface. Since the channel number of a multichannel filter equals $IF_NUM * CLK_PER_SAMPLE$, and the maximum channel count is 64, the CLK_PER_SAMPLE must not exceed the value of $64/IF_NUM$.
USE_RAM	0, 1	0	Use RAM blocks to implement CIC registers. 0: Use only fabric-based register implementations. 1: Permission to use RAM blocks when appropriate.
URAM_MAX_DEPTH	0, 4, 8, 16, 32, 64, 128, 256, 512, 1024	0	Maximum depth of a RAM to be implemented using micro-RAM (uRAM) blocks. If USE_RAM is set, the core uses hard RAM blocks available on a selected FPGA device. When the uRAM blocks are available, the core uses them if the required memory depth does not exceed the URAM_DEPTH. Otherwise it builds the memory out of LSRAM blocks.

4.2 Ports

The following figure shows the CIC filter schematic representation where optional port names are in *Italics*.

Figure 14 I/O Ports



The pinout is a superset of all possible ports. The following table provides the port definitions for the core.

Table 6 CIC Filter In or Out Signals

Signal	In/Out	Port Width Bits	Description
DIN	In	DIN_WIDTH for Interpolation CIC; DIN_WIDTH*IF_NUM for decimation filter	Input data to be filtered. In case of interpolation filter or a single channel for any CIC type, the data width equals DIN_WIDTH. In case of decimation filter with multiple Interfaces, the input data width for each interface is DIN_WIDTH and the total input data width = DIN_WIDTH*IF_NUM.
DIN_VALID	In	1	Input data valid. Active High. When the signal is active, the input data sample is loaded into decimation CIC filter. If not used, the core assumes that every data sample is valid. For the interpolation CIC filter, the signal not only marks the valid input sample but also defines a rate for the interpolated output samples. For example, if DIN_VALID has a duty cycle of three, the output sample rate is three times less than the CLK rate. The input sample of the interpolator is valid, if it is accompanied by DIN_VALID and CHAN_SYNC_I signals. The signal must be one-clock wide.
RFD	Out	1	Ready for input data. In case of interpolation filter, the RFD notifies a data source that the CIC filter is ready for a new input sample. In case of decimation filter, the signal goes Low only when integrator reset process is underway. Input data samples are ignored when RFD is Low.
RFD_PILOT	Out	1	Optional pulse the CIC interpolator generates just before the RFD signal. The pulse width is one clock period. It can be used to let the data source more time to prepare another data sample.
DOUT	Out	DOUT_WIDTH for Decimation CIC; DOUT_WIDTH*IF_NUM for Interpolation filter	Output filtered data. In case of decimation filter or a single channel for any CIC type, the output data width equals DOUT_WIDTH. In case of interpolation filter with multiple Interfaces, the output data width for each interface is DOUT_WIDTH and the total result data width = DOUT_WIDTH*IF_NUM.
DOUT_VALID	Out	1 for Decimation CIC; IF_NUM for Interpolation filter	Filtered data valid indicates that a new output data sample is present at the DOUT port. For the decimation filter as well as a single interface interpolation filter, the signal is one bit wide. In case of interpolation filter with multiple interfaces, the output data width for each interface is one and the total result signal width = IF_NUM.
RATE	In	11	Variable rate value. The port is available when variable rate change mode is selected, VAR_RATE = 1.
RATE_WE	In	1	Register a new value of the variable RATE. The RATE_WE must be a one-clock wide pulse. The port is available when variable rate change mode is selected, VAR_RATE = 1. After the RATE value is stored in the core, the effective rate factor does not change yet. The new RATE value takes effect after the core receives the synchronous reset RST pulse.
CHAN_SYNC_I	In	1	Channel synchronization signal. For a decimation filter, it provides an advanced identification for the first data channel when CLK_PER_SAMPLE > 1. For an interpolation filter, the signal is expected to come even if the filter has only one

Table 6 CIC Filter In or Out Signals

Signal	In/Out	Port Width Bits	Description
			channel. Then the signal marks a time slot when the input sample is valid. For a multichannel interpolator, when CLK_PER_SAMPLE > 1 the signal is supposed to mark the valid input sample of the first time-share channel. The CHAN_SYNC_I must be a one-clock wide pulse.
CHAN_SYNC_O	Out	1	Output channel synchronization signal. This clock-wide pulse identifies the first channel data output sample.
CHAN_OUT	Out	6	Output channel numerical ID.
CLK	In	1	The core master clock.
NGRST	In	1	Optional asynchronous reset. Active Low. The signal is expected to follow the FPGA power-on. The signal initiates reset of all internal registers. If RAM blocks are used to implement integrator or comb registers, the actual reset can take several clock cycles. Then on the rear edge of the NGRST, the core automatically generates an internal reset wide enough to reset all the registers.
RST	In	1	Synchronous reset. Active High. The signal initiates reset of all internal registers. If RAM blocks are used to implement integrator or comb registers, the actual reset can take several clock cycles. Then the RST signal initiates an internal reset wide enough to reset all the registers.

Figure 15 and Figure 16 show examples of using the core in fixed and variable rate modes.

Figure 15 Fixed Rate Single Channel CIC Filter

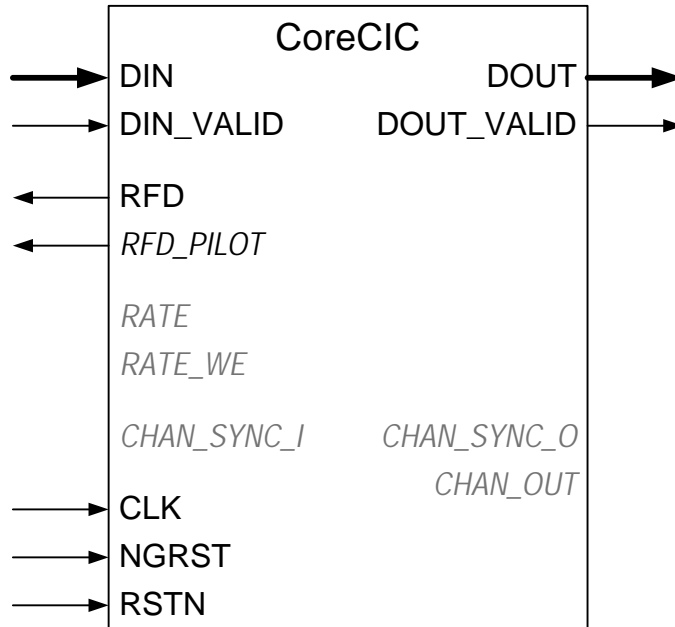
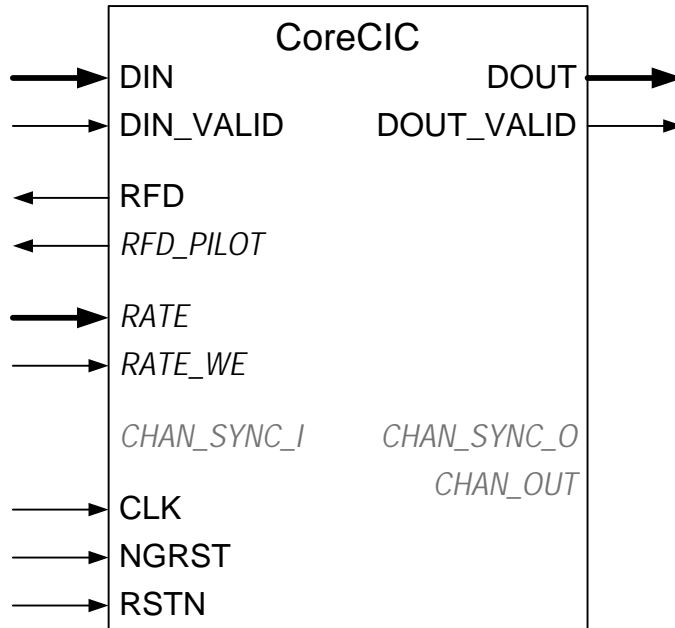
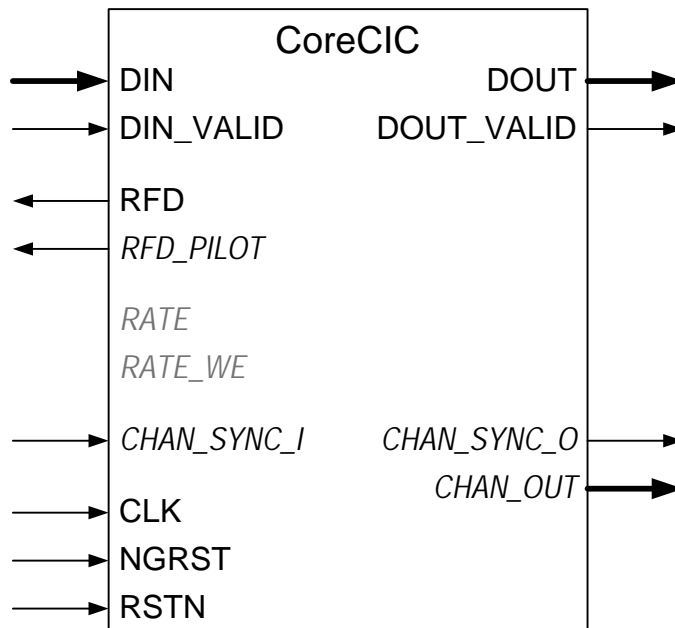


Figure 16 Variable Rate Single Channel CIC Filter



The following figure shows the filter connections in multiple channel mode.

Figure 17 Fixed Rate Multiple Channel Mode



4.2.1 Decimator Interface

This section describes basics of the CIC decimation functionality. For more information, refer to [Implementation Details](#) section.

A decimation filter receives R_RATE input samples to generate an output sample. The input data samples sequentially arrive at DIN port of a single channel decimator. Every input sample is accompanied by the DIN_VALID pulse of one clock period. The filter accepts arbitrary gaps between the input samples. The data source indicates the break by de-asserting the DIN_VALID signal. For maximum throughput, there must not be any gaps in between the input samples. If a data source can supply a new sample for every FPGA clock cycle, attach the DIN_VALID to VCC. Once the decimator gets R_RATE input samples, it posts an output filtered sample on the DOUT port after it finishes the sample processing. It also accompanies the valid output sample by the clock-wide DOUT_VALID signal. The CIC filter introduces processing latency that is described below.

A time share multichannel decimator, where CLK_PER_SAMPLE > 1 expects the data samples to come in the natural order. For example, at CLK_PER_SAMPLE = 3 the first channel sample comes first, followed by a sample of the second channel and then third channel. Input sample of each channel is accompanied by the DIN_VALID pulse. Similarly, to the single channel, if there are no gaps in between the input samples the DIN_VALID signal has to be permanently high. The data source identifies the first channel by the CHAN_SYNC_I pulse that accompanies the DIN sample of the first channel. In other words, the CHAN_SYNC_I pulse is a copy of the DIN_VALID pulse for the first data channel. The decimator assigns the channels numerical IDs, from 0 to 2. On receiving R_RATE input samples from all channels, the decimator outputs filtered DOUT data, one sample at a time for the channels 0 to 2. The output channels are accompanied by the DOUT_VALID pulses. The DOUT output for the channel 0 is accompanied by the CHAN_SYNC_O pulse. The CHAN_SYNC_O is a copy of the channel 0 DOUT_VALID signal. CHAN_OUT provides the number of channels currently posted on the DOUT output. In this example, CHAN_OUT sends the numbers 0, 1, 2 synchronized with valid output samples.

A multiple interface decimator, where IF_NUM > 1 expects input data samples to come to each interface, simultaneously. A single DIN_VALID pulse is used for all interfaces. It signifies another set of IF_NUM input samples is ready to be received by all interfaces. Similarly, to the single channel, if there are no gaps in between the input samples, the DIN_VALID signal must be permanently High. If multiple interface decimator does not utilize time share, that is the parameter CLK_PER_SAMPLE=1, the CHAN_SYNC_I signal must replicate the DIN_VALID pulse. Once the decimator receives R_RATE input samples on all its interfaces, it starts placing output samples on the DOUT port one channel at a time: an output sample for the channel coming to the interface 0 and the channel coming to the interface 1, and so on. Each output sample is accompanied by the DOUT_VALID pulse. CHAN_SYNC_O marks the channel of the interface 0. CHAN_OUT supplies the numerical channel IDs synchronized with the output samples.

A multichannel decimator can combine time share and comb share when IF_NUM > 1 and CLK_PER_SAMPLE > 1. The total number of channels processed by such filter equals IF_NUM*CLK_PER_SAMPLE. Consider an example of a decimator processing six channels: IF_NUM = 2 and CLK_PER_SAMPLE = 3.

The data samples are expected to come to the interface 0 in the following order:

- Sample 0 of the channel 0
- Sample 0 of the channel 1
- Sample 0 of the channel 2
- Sample 1 of the channel 0
- Sample 1 of the channel 1, and so on

Simultaneously the following data is coming to the interface 1:

- Sample 0 of the channel 3
- Sample 0 of the channel 4
- Sample 0 of the channel 5
- Sample 1 of the channel 3
- Sample 1 of the channel 4, and so on

Every sample of the channel 0 to channel 2 is accompanied by DIN_VALID pulses unless the data is coming without gaps. Samples of the channel 0 are accompanied by the CHAN_SYNC_I pulses. Since the data to interfaces 0 and 1 is coming simultaneously, the CHAN_SYNC_I pulse marks the channel 3 as well. The filter outputs samples of the channel 0 to 5 sequentially with every sample accompanied by the DOUT_VALID pulse. The core generates the CHAN_SYNC_O pulse when the channel 0 filtered sample shows up at the DOUT output. CHAN_OUT supplies channel IDs from 0 to 5 synchronized with valid output samples.

Upon reset signals, NGRST or RTS, the decimator enters the reset state. During the reset state, the decimator keeps the RFD signal Low.

Note: The DIN_VALID pulses are expected to be present at this time but the core ignores any input data while the RFD signal is Low. In the decimation mode, only during the reset state the RFD signal goes Low.

4.2.2 Interpolator Interface

This section describes basics of the CIC interpolation functionality. For more information, refer to [Implementation Details](#) section.

An interpolation filter generates R_RATE output samples for every input data sample. The input data samples arrive at DIN port of a single channel interpolator spaced by time intervals sufficient for the filter to output R_RATE filtered samples. To assist a data source in providing minimal sufficient time intervals, the interpolator generates the handshake signals request for data (RFD) and RFD_PILOT. The interpolator raises the RFD when it is ready to accept a new input sample. The RFD stays High until the data source provides a valid input sample.

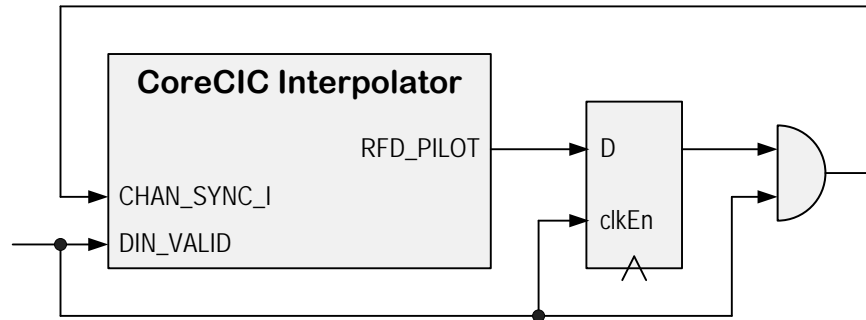
Note: The functionality of the DIN_VALID pulse differs from that of decimation CIC: the signal not only accompanies the valid input samples but also influences interpolated output sample rate. The DIN_VALID pulse can direct the interpolator to output filtered samples at a fraction of clock rate. For example, it is required that the CIC interpolator generates output samples at every third clock interval. Then the DIN_VALID duty cycle has to be 1/3, that is the DIN_VALID pulse comes at every third clock cycle. Since the input sample rate of the interpolation filter is R_RATE times lower, the valid input sample period of this example equals 3*R_RATE. It is accompanied by the CHAN_SYNC_I pulse. It means, the valid input sample is accompanied by CHAN_SYNC_I and DIN_VALID pulses.

The RFD_PILOT is a clock-wide pulse that precedes the RFD signal. The data source can use either or both handshake signals when generating another input data sample.

Note: The interpolator only accepts the data samples that are accompanied by the CHAN_SYNC_I signal while the RFD signal is High.

The filter tolerates gaps between the input samples that exceed the minimal time intervals. The filter raises the output RFD signal, and waits for the next valid input sample. For maximum throughput, the data source must supply a fresh input sample on the clock interval following the RFD_PILOT signal. Then the DIN_VALID signal must be connected to VCC. The RFD_PILOT pulse can be used to achieve the highest throughput at any given DIN_VALID rate, refer to the following figure.

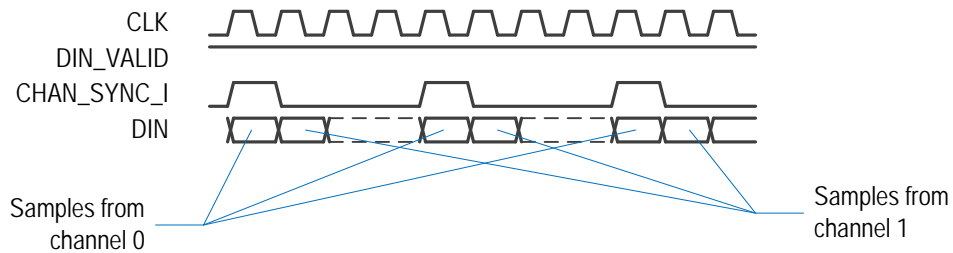
Figure 18 Using RFD_PILOT Signal



A time share multichannel interpolator where $CLK_PER_SAMPLE > 1$ expects the data samples to come in the natural order. For example, at $CLK_PER_SAMPLE = 3$, the first channel sample comes first followed by a sample of the second channel and the third channel. Samples must be spaced so that the CIC has sufficient time to output interpolated samples at the rate of DIN_VALID . It means, an interval between every two consecutive input samples must not be less than R_RATE of DIN_VALID time intervals. Every input sample is accompanied by the DIN_VALID pulse. The data source identifies the first time share channel by the $CHAN_SYNC_I$ pulse that accompanies the DIN sample of the first channel. It means, the $CHAN_SYNC_I$ is a copy of the DIN_VALID pulse for the first data channel. The interpolator assigns the channels numerical IDs from 0 to 2. On receiving an input sample from all three channels, the CIC outputs $3 * R_RATE$ interpolated $DOUT$ data, one sample at a time: a first interpolated sample for the channel 0, a first interpolated sample for the channel 1 and the first interpolated sample for the channel 2. Then it outputs the second interpolated sample for 0 to 2 channels, the third one and finally the R_RATE-1 interpolated sample for the channels 0 to 2. The output channels are accompanied by the $DOUT_VALID$ pulses. Their rate equals the rate of DIN_VALID pulses. If the DIN_VALID signal is permanently High, the $DOUT_VALID$ signal also stays High permanently. The $DOUT$ output for the channel 0 is accompanied by the $CHAN_SYNC_O$ pulse. The $CHAN_OUT$ provides the number of a channel currently posted on the $DOUT$ output. In this example, the $CHAN_OUT$ would output the numbers 0, 1, 2 synchronized with valid output samples.

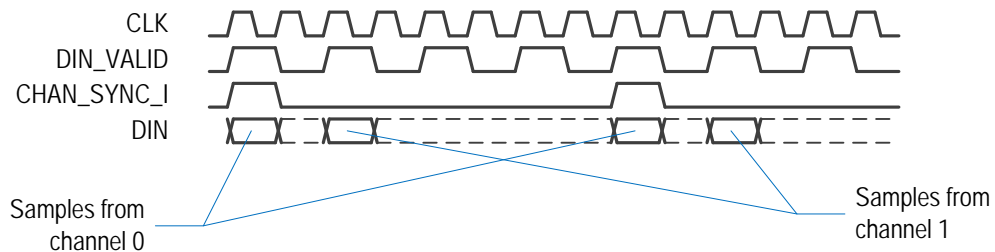
A multiple interface interpolator where $IF_NUM > 1$ expects all the input data samples to come to a single data input DIN sequentially: samples 0 of the channels 0 to IF_NUM-1 , then samples 1 of the channels 0 to IF_NUM-1 , etc. Every input sample is accompanied by the DIN_VALID pulse, and the samples of the channel 0 are additionally accompanied by the $CHAN_SYNC_I$ pulses. Only the samples from the same channel have to provide sufficient time interval for the filter to generate interpolated samples. It means, samples from channels 0 to IF_NUM-1 are supposed to come on every clock or spaced by DIN_VALID period. The following figure shows an example of interpolator input signals when $R_RATE = 4$, $IF_NUM = 2$, and input data are permanently valid, that is $DIN_VALID = 1$. The samples from channel 0 and 1 come without any interval but the DIN waits for two clock cycles so, the consecutive samples of the same channel are spaced by R_RATE-1 clock intervals.

Figure 19 Example of Interpolator Input Signals at DIN_VALID Always Active



The following figure shows an example of the same interpolator configuration. The input data is valid on every other clock interval.

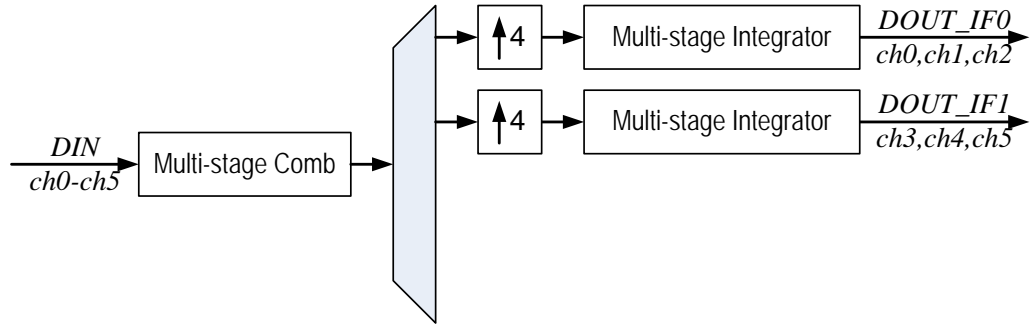
Figure 20 Example of Interpolator Input Signals at DIN_VALID Active Every Other Clock



In both examples, on receiving an input sample from channels 0 and 1, the CIC outputs $4 \cdot R_RATE$ interpolated DOUT data, one sample at a time: a first interpolated sample for the channel 0 and a first interpolated sample for the channel 1, refer to [Figure 19](#) and [Figure 20](#). Then it outputs the second interpolated sample for the channels 0 and 1, etc and finally the fourth interpolated sample for the channels 0 and 1. The output channels are accompanied by the DOUT_VALID pulses. Their rate equals the rate of DIN_VALID pulses. If the DIN_VALID signal is permanently High, the DOUT_VALID signal also stays High permanently. The DOUT output for the channel 0 is accompanied by the CHAN_SYNC_O pulse.

A multichannel interpolator can combine time share and comb share when $IF_NUM > 1$ and $CLK_PER_SAMPLE > 1$. The total number of channels processed by such filter equals $IF_NUM \cdot CLK_PER_SAMPLE$. For example, an interpolator processing six channels: $IF_NUM = 2$, $CLK_PER_SAMPLE = 3$, and rate changing factor $R_RATE = 4$. The following figure shows in/out channel mapping.

Figure 21 I/O Channel Mapping for an Interpolator Example



The data samples are expected to come to the interpolator DIN input in the following order: sample 0 of the channel 0, sample 0 of the channel 1, and sample 0 of the channel 5. The interpolated output samples appear on the interface 0 for the channels 0 to 2 and on the interface 1 for the channels 3 to 5.

5 Implementation Details

5.1 Reset

The CoreCIC filter must be reset to generate correct results. It gets reset automatically on powering ON an FPGA. The core initializes the reset based on positive edge of the NGRST signal that normally follows the power ON. The RST signal initiates the core reset. Depending on the core configuration, the reset state can take several clock cycles to properly initialize the core. Once initiated by NGRST or RST signals, the core ignores any data while the reset state takes place. When the reset state is over, the core raises the RFD signal and the core is ready for normal processing.

Note: The DIN_VALID pulses are expected to be present during the reset state but the core ignores any input data while the RFD signal is Low.

5.2 Latency

EQ3 describes the core latency expressed in clock CLK cycles when DIN_VALID is not High permanently:

$$\text{Latency} = N_STAGES * (3 * \text{DIN_VALID} + 4) + 2 \text{ Clock Cycles}$$

EQ3

When DIN_VALID is permanently active, the formula changes to:

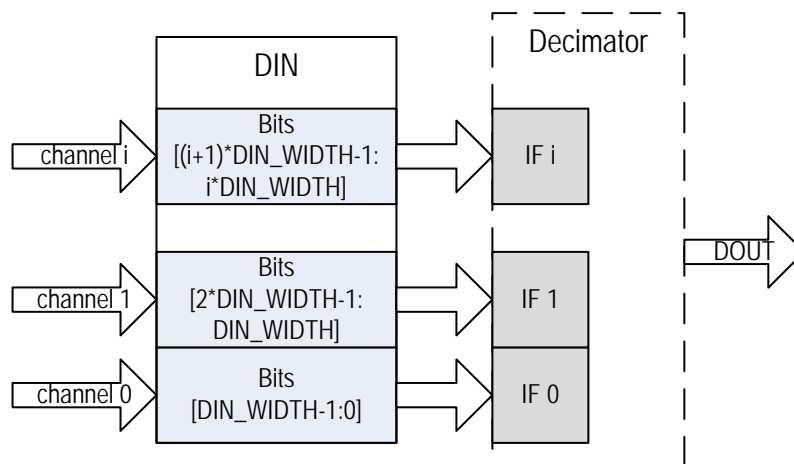
$$\text{Latency} = 7 * N_STAGES + 2 \text{ Clock Cycles}$$

EQ4

5.3 Multiple Interface Connections

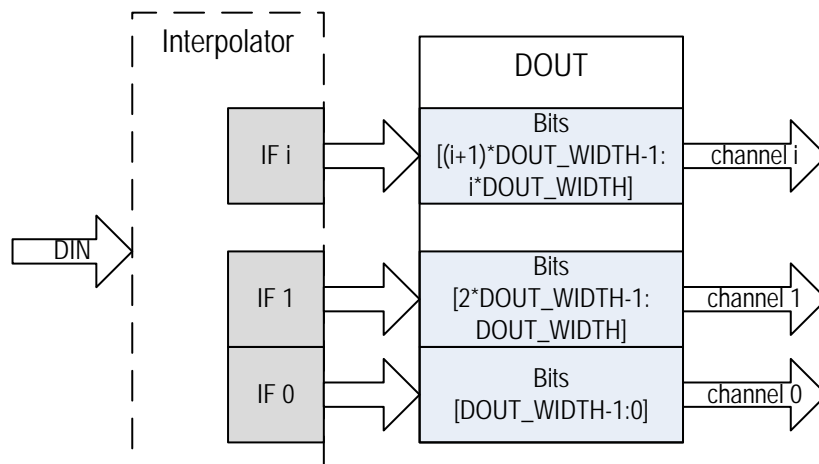
Both decimation and interpolation filters can have multiple interfaces ($IF_NUM > 1$) to provide connections to high speed data source or data sink. Multiple interface decimator receives data from multiple channels simultaneously, refer to [Figure 10](#) and [Figure 12](#). The data from different channels must be concatenated to form a single DIN word, refer to the following figure. The concatenated DIN word width = $IF_NUM * \text{DIN_WIDTH}$ bits.

Figure 22 Data Concatenation for Multiple Interface Decimator



Similarly, the multiple interface interpolator provides parallel data for multiple channels by generating a concatenated DOUT word, refer to [Figure 11](#) and [Figure 13](#). Bits arrangement for the DOUT interpolation word is depicted in the following figure. The concatenated DOUT word width = IF_NUM*DOUT_WIDTH.

Figure 23 Output Concatenation of a Multiple Interface Interpolator



5.4 Variable Rate

In addition, to a fixed upsampling or downsampling rate, the core supports programmable rate change factor. To enable the programmable rate mode, select **Enable Variable Rate** check box and enter the initial Rate Change Factor, Minimum, and Maximum programmable rate values. On power ON, the core automatically uses the Rate Change Factor value. To update the rate change value, you need to provide the desired rate change value on the RATE port of the core and issue a clock wide pulse on the RATE_WE pin. Then the desired RATE value gets stored inside the core. The core uses the previously entered rate change value or initial Rate Change Factor until the synchronous reset pulse on the RST port is provided. When the core completes its internal reset, the new rate change value is used.

The minimum and maximum programmable rate values are used at the IP generation time to create RTL that can handle the indicated values.

Note: The wider the variable (programmable) rate range, the more FPGA resources will be utilized and potentially lower max clock rate will be achieved.

5.4.1 RAM Block Use

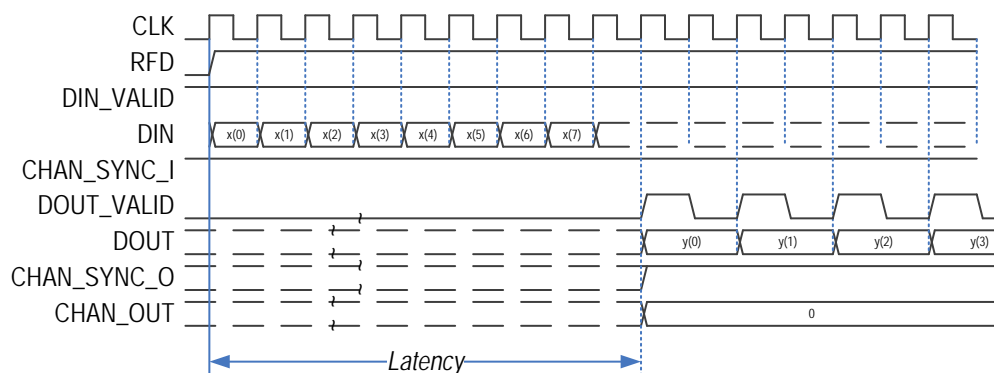
The CIC filter components such as, delay line of the comb section and multiple channel accumulators of the integrator section can be implemented using hard RAM blocks available on FPGA. Such implementation reduces fabric resource utilization and can improve processing speed. The core automatically infers the RAM blocks, if the **Use RAM Blocks** check box is selected. You can decide the RAM blocks to be used, LSRAM or uRAM by providing **Max MicroRAM Depth** value. The core utilizes uRAM blocks whenever the RAM depth required does not exceed the value entered. Otherwise it uses the LSRAM blocks.

5.5 Decimation Filter Timing

Figure 24 to Figure 27 show a few examples of decimator timing diagrams. The fixed downsampling rate $R_RATE = 2$ for all the examples, that is the decimator outputs one sample for every two input samples. The timing diagrams show time intervals immediately after the RFD goes High in response to the NGRST or RST signals issued earlier. This is done to illustrate the Latency time. The timing relations between the signals stay the same indefinitely after.

The following figure shows a timing diagram for a single channel decimator where data samples come at every clock cycle. It is assumed that the filter has been reset earlier so, that the RFD signal is active. Data samples $x(0)$ and $x(1)$ are coming to the DIN input at every clock cycle from when DIN_VALID goes High. CHAN_SYNC_I for the example should be indefinitely active, as there is only a single channel.

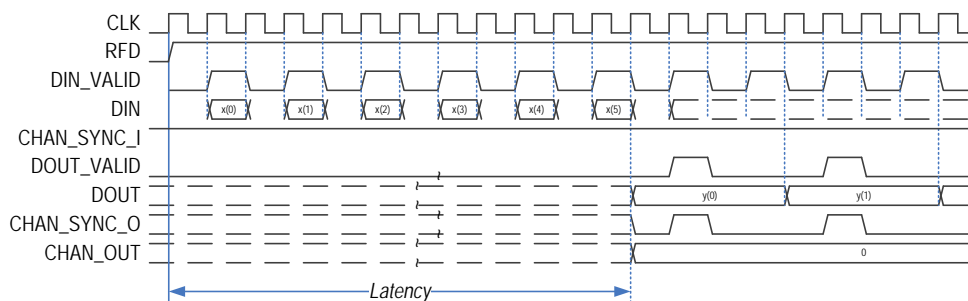
Figure 24 Decimation CIC Timing - Single Channel, Data Permanently Valid



After the latency time of the filter expires, the decimator raises the DOUT_VALID signal and outputs a decimated sample $y(0)$ on the DOUT port. The DOUT_VALID signal stays High for a single clock period and then goes Low for R_RATE clock periods, in this case for one clock period. The DOUT port keeps the output sample until the next DOUT_VALID pulse starts. Once the DOUT_VALID pulse gets asserted for the first time since filter reset happened, the CHAN_SYNC_O goes High and stays in the active state indefinitely. The CHAN_OUT outputs the channel number of 0.

The following figure shows a timing diagram for the single channel decimator with data coming every other clock. The input samples $x(0)$ and $x(1)$ are accompanied by the DIN_VALID pulses. $CHAN_SYNC_I$ can be a copy of the DIN_VALID pulses or just stay High indefinitely. After Latency, the core raises the $DOUT_VALID$ signal for one clock period and starts outputting the decimated sample $y(0)$, $y(1)$, etc. The $DOUT_VALID$ pulses are separated by four clock periods. The $CHAN_SYNC_O$ signal replicates the $DOUT_VALID$ signal, and the $CHAN_OUT$ equals 0.

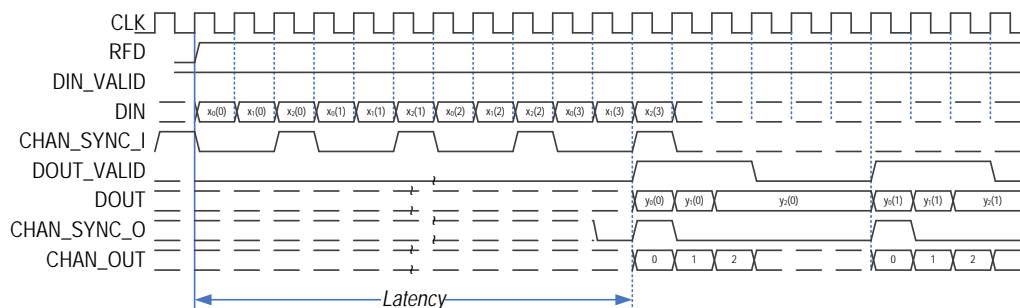
Figure 25 Decimation CIC Timing - Single Channel, Data Coming Every Other Clock



The following figure shows an example of a multi-channel decimator timing diagram. In this example, a data source is capable of providing a fresh input sample at every clock interval thus, the DIN_VALID signal is High permanently. The data comes from three multiplexed time share channels x_0 , x_1 and x_2 on the same DIN bus. Multiplexing the three channels into a single bus is possible, as for this example the parameter $CLK_PER_SAMPLE = 3$. $CHAN_SYNC_I$ marks every sample of the first channel data x_0 .

The $CHAN_SYNC_I$ is supposed to provide advanced warning when the first channel data is coming, that is it must come immediately before the first channel data x_0 . Since the channels normally are cyclically multiplexed, it is recommended to raise the $CHAN_SYNC_I$ signal when the last channel data is present on the core DIN input. The following figure shows the $CHAN_SYNC_I$ signal active while the channel x_2 data enter the filter.

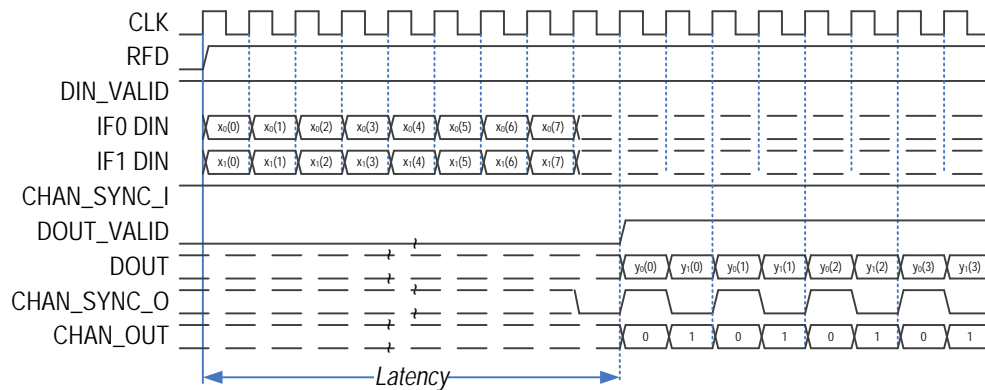
Figure 26 Decimation CIC Timing - Three Time Share Channels, Data Permanently Valid



Once the decimator is ready to output the filtered samples, it raises the $DOUT_VALID$ signal that lasts for three clock intervals. Then the $DOUT_VALID$ signal goes inactive for $(R_RATE - 1) * CLK_PER_SAMPLE$ clock periods, that is for three periods in this example. $CHAN_SYNC_O$ marks every output sample of the channel y_0 , and the $CHAN_OUT$ counts channels 0 to 2 while the $DOUT_VALID$ signal is active.

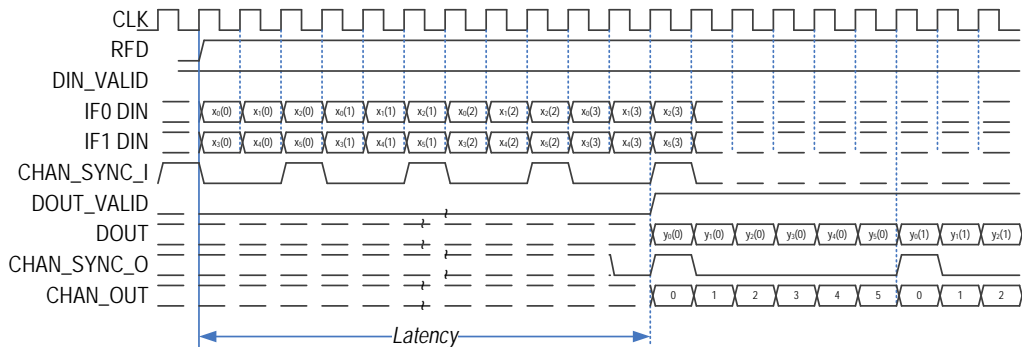
The following figure shows a timing diagram for the two-channel decimator that has two interfaces, IF_NUM = 2. Data from the channel x0 comes to the lower bits of the DIN input (see section 8.3). It is called IF0 DIN, refer to the following figure. Data from the channel x1 comes to the upper bits of the DIN input named IF1 DIN. For this example, CHAN_SYNC_I stays High, as each interface does not utilize time share (CLK_PER_SAMPLE = 1). After Latency interval, the core raises DOUT_VALID and starts outputting the decimated samples: y0(0) is calculated based on the two samples x0(0) and x0(1), y0(1) is based on the samples x0(2) and x0(3), etc. Similarly, the decimated samples of the interface IF1 are calculated. The data from both interfaces appear on the port DOUT. The CHAN_SYNC_O signal marks the output samples originated at the interface IF0. The CHAN_OUT signal indicates the channel that is currently being present at the DOUT port.

Figure 27 Decimation CIC Timing - Two Interfaces, Data Permanently Valid



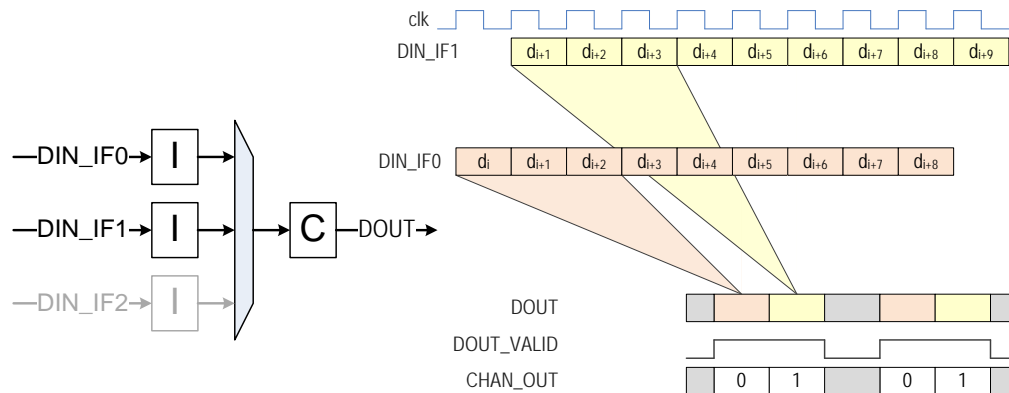
The following figure shows a six-channel CIC decimator timing diagram. The decimator has two interfaces IF_NUM = 2, with each interface processing three multiplexed channels at CLK_PER_SAMPLE = 3. CHAN_SYNC_I provides advance warning the first channel for each interface is about to come. The CHAN_SYNC_I signal coincides with the last time share channel, that is the channel x2 on the interface IF0 and the channel x5 of the interface IF1. Since the example implements the maximum number of interfaces IF_NUM = 2 allowed at R_RATE = 2, the comb section of the filter is busy all the time and the valid decimated samples are generated without breaks. This is signified by the DOUT_VALID High permanently after initial Latency.

Figure 28 6-Channel Decimator Timing - Two Interfaces Three Time Share Channels Each, Data Always Valid



There is a subtle difference in processing parallel streams of data coming to multiple CIC interfaces. The following figure shows an example of a CIC decimator with IF_NUM=2, R_RATE=3, and CLK_PER_SAMPLE=1. The processing of the DIN_IF1 data is shifted by one sample with regard to the input data of the DIN_IF0.

Figure 29 Input Data Shift Between Interfaces IF0 and IF1

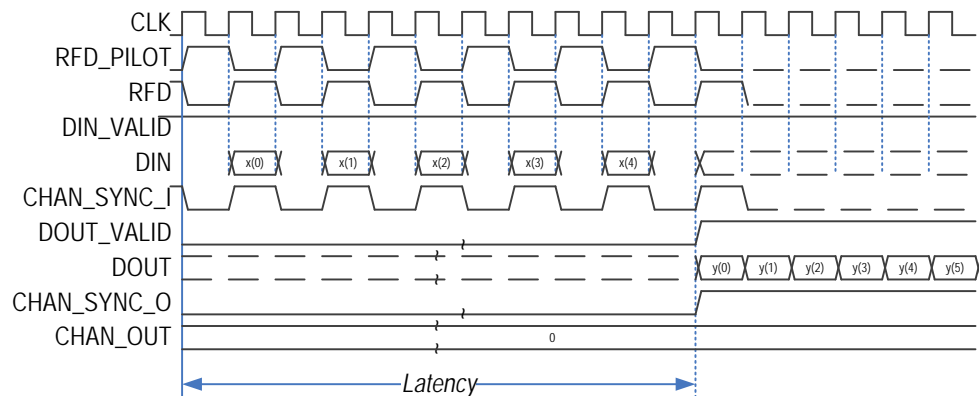


5.6 Interpolation Filter Timing

Figure 30 to Figure 33 shows a few examples of interpolator timing diagrams. The fixed upsampling rate R_RATE = 2 for all the examples, that is the interpolator generates two samples for every input sample. The timing diagrams show time intervals immediately after resetting the filter by the NGRST or RST signals issued earlier. This is done to illustrate the Latency time. The timing relations between the signals stay the same indefinitely after. Figure 18 shows the examples use CHAN_SYNC_I generation technique.

The following figure shows the case of a single channel interpolator where input data are always available (DIN_VALID is High). The interpolator generates RFD_PILOT pulses on every other clock interval. The RFD follows the RFD_PILOT signal. A data source responds with a fresh data sample x(0), x(1), etc every time when the RFD signal is active. Every input sample is accompanied with the CHAN_SYNC_I pulse.

Figure 30 Interpolation CIC Timing - Single Channel, Data Permanently Valid

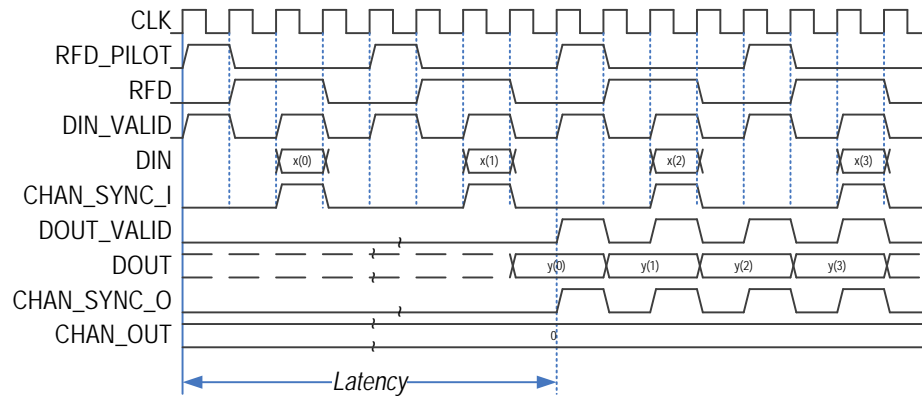


After Latency, the core sets the DOUT_VALID signal, which stays active indefinitely and starts generating interpolated samples $y(0)$, $y(1)$, and so on. The CHAN_SYNC_O signal is a copy of the DOUT_VALID signal for this example.

The following figure shows a timing diagram for the single channel interpolator with data coming every other clock. The core generates the RFD signal that stays active until the data source supplies another valid sample marked by the CHAN_SYNC_I pulse.

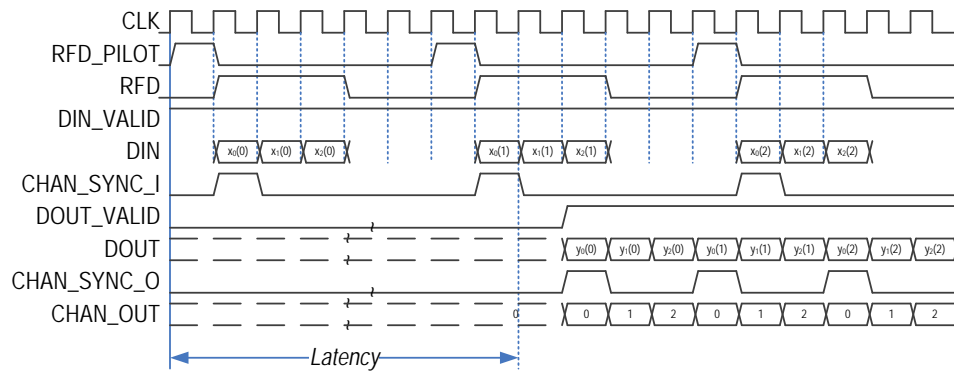
Note: CHAN_SYNC_I pulses are a subset of DIN_VALID pulses. The DIN_VALID pulse on the CIC interpolation filter actually define the rate of the interpolated output samples. The DOUT_VALID signal once it becomes active for the first time after a reset, follows the DIN_VALID rate. The interpolated samples appear on the DOUT port: the samples $y(0)$ and $y(1)$ are calculated based on the input sample $x(0)$, the samples $y(2)$ and $y(3)$ are based on the sample $x(1)$, etc. The CHAN_SYNC_O signal replicates the DOUT_VALID signal for this example. The CHAN_OUT signal keeps the value of 0 all the time.

Figure 31 Interpolation CIC Timing - Single Channel, Data Coming Every Other Clock



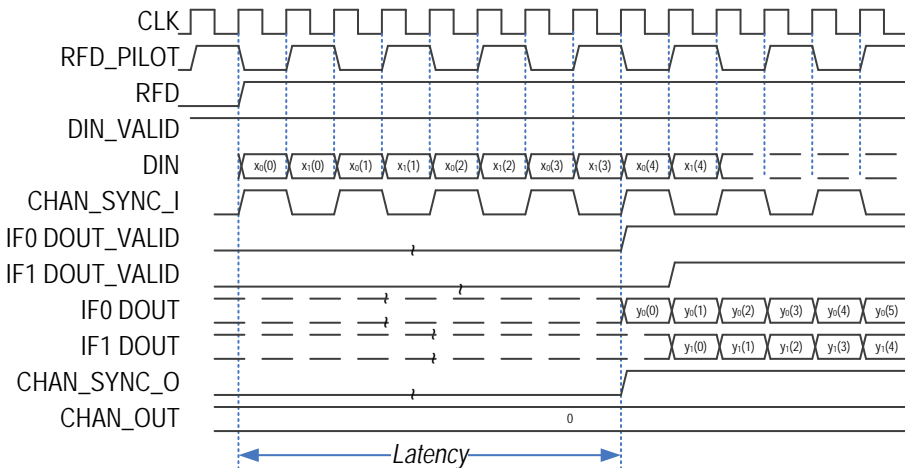
The following figure shows an example of a multi-channel interpolator timing diagram. In this example, a data source is capable of providing a fresh input sample at every clock interval, thus the DIN_VALID pulse is High permanently. The data comes from three multiplexed time share channels x_0 , x_1 and x_2 on the same DIN bus. Multiplexing the three channels into a single bus is possible, as for this example the parameter CLK_PER_SAMPLE = 3. The RFD signal stays active for three consecutive clock periods to accept an input sample from each of the channels. CHAN_SYNC_I marks every sample of the first channel data x_0 . The example of assumes that the core reset recently happened to show the Latency interval, refer to the following figure. Once the interpolator is ready to output the filtered samples after Latency, it raises the DOUT_VALID signal. The interpolated samples appear on the DOUT port: the samples $y_0(0)$ and $y_0(1)$ are calculated based on the input sample $x_0(0)$, the samples $y_0(2)$ and $y_0(3)$ are calculated based on the sample $x_0(1)$, and so on. CHAN_SYNC_O marks every output sample of the channel y_0 , and the CHAN_OUT counts channels 0 to 2 while the DOUT_VALID is active.

Figure 32 Interpolation CIC Timing - Three Time Share Channels, Data Permanently Valid



The following figure shows a timing diagram for the two-channel interpolator that has two interfaces, IF_NUM = 2. Data from the channels x0 and x1 come to the DIN port sequentially, x0(0), x1(0), x0(1), x1(1), etc. Once RFD gets active after initial reset, stays active indefinitely, as throughput of a comb section is fully utilized due to the number of interfaces being equal the upsampling rate, IF_NUM = R_RATE. CHAN_SYNC_I marks the data of the channel x0 intended for the interface IF0.

Figure 33 Interpolation CIC Timing - Two Interfaces, Data Permanently Valid



After Latency the core sets the bit 0 of the DOUT_VALID signal, which is a two bits word. The bit is referred to as IF0 DOUT_VALID, refer to above figure. The IF0 DOUT_VALID signal flags the valid interpolated samples y0 on the lower bits of the DOUT port (see section 8.3 for the bit arrangement of the concatenated interpolator output port DOUT). These bits are referred as IF0 DOUT, refer to above figure. On the next clock, the core raises the IF1 DOUT_VALID signal and starts outputting the channel y1 samples. CHAN_SYNC_O here replicates the IF0 DOUT_VALID signal. In this example, the CHAN_OUT signal keeps generating the value of 0 as there is only a single channel for each output interface.

6 Tool Flow

6.1 License

CoreCIC requires a RTL license to be used and instantiated.

6.1.1 RTL

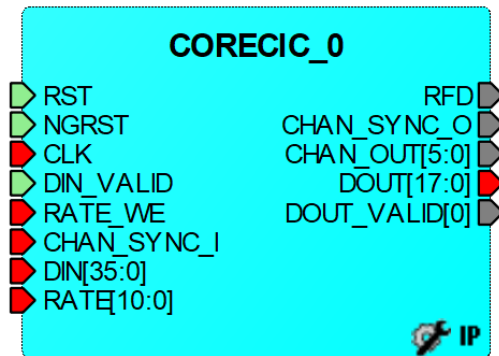
Complete RTL source code is provided for the core.

6.2 SmartDesign

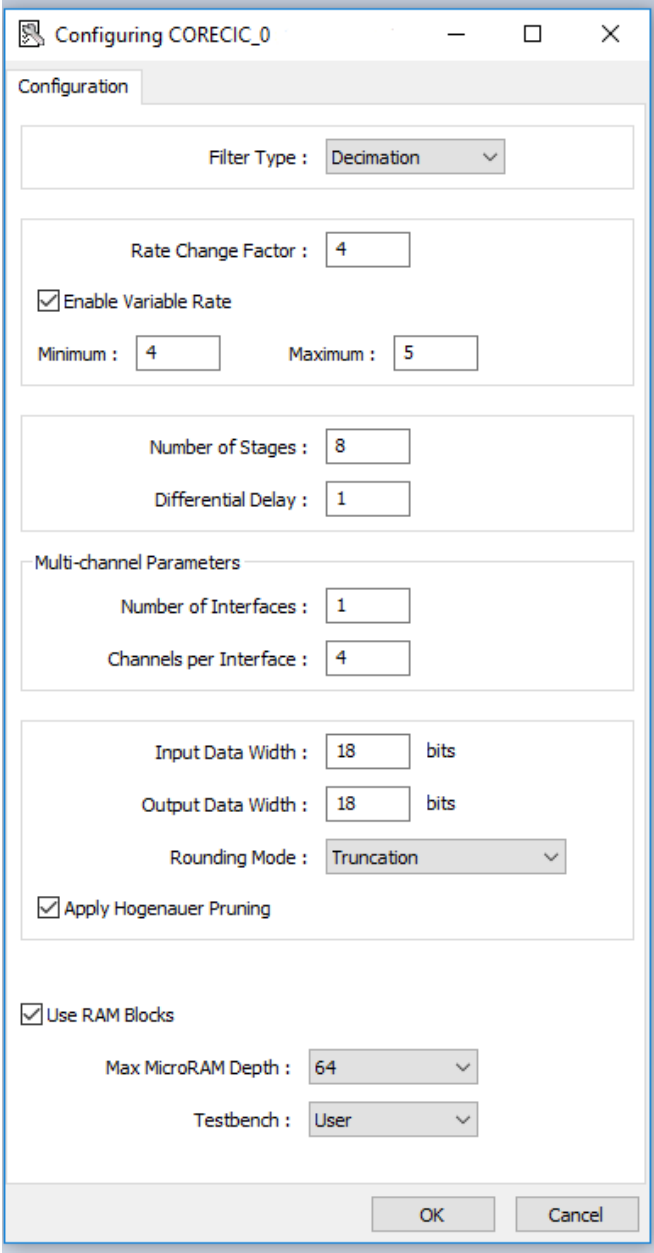
CoreCIC is preinstalled in the SmartDesign IP Deployment design environment.

The core should be configured using the configuration GUI within SmartDesign, as shown in the following figure. For information on using the SmartDesign to instantiate and generate cores, see [Libero SoC online help](#).

Figure 34 CoreCIC Full I/O View



The core is configured using the configuration GUI within SmartDesign, as shown in the following figure.

Figure 35 SmartDesign CoreCIC Configuration Window


Configuring CORECIC_0

Configuration

Filter Type : Decimation

Rate Change Factor : 4

Enable Variable Rate

Minimum : 4 Maximum : 5

Number of Stages : 8

Differential Delay : 1

Multi-channel Parameters

Number of Interfaces : 1

Channels per Interface : 4

Input Data Width : 18 bits

Output Data Width : 18 bits

Rounding Mode : Truncation

Apply Hogenauer Pruning

Use RAM Blocks

Max MicroRAM Depth : 64

Testbench : User

OK Cancel

6.3 Simulation Flows

To run simulations, select the user testbench in the core configuration window. After generating the CoreCIC, the pre-synthesis test-bench hardware description language (HDL) files are installed in Libero.

6.4 Synthesis in Libero

To run synthesis on the CoreCIC, set the design root to the IP component instance and run the synthesis tool from the Libero design flow pane.

6.5 Place-and-Route in Libero

After the design is synthesized, run the compilation and then place-and-route the tools.

CoreCIC requires no special place-and-route settings.

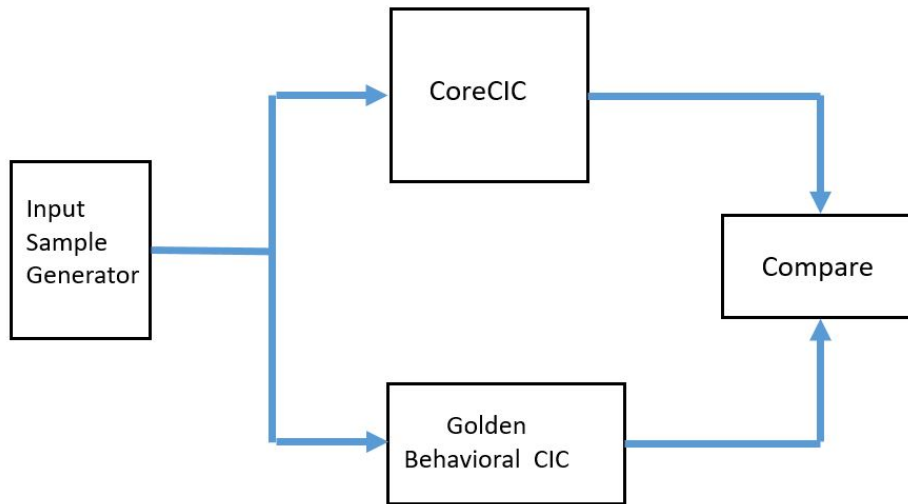
7 Testbench

7.1 User Testbench

A unified test-bench is used to verify and test CoreCIC called as user test-bench.

A simplified block diagram of the user testbench is as shown in the following figure. The testbench compares the output signals of the Golden module and the actual CoreCIC.

Figure 36 CoreCIC User Testbench Simplified Block Diagram



8 References

The following list of references are used in this document:

- Richard Lyons, Understanding Cascaded Integrator-Comb Filter, Embedded.com, March 31 2005 <http://www.embedded.com/design/configurable-systems/4006446/Understanding-cascaded-integrator-comb-filters>
- Fredric J.Harris Multirate Signal Processing for Communication Systems, Prentice Hall PTR, 2004
- Eugene B.Hogenauer, An Economical Class of Digital Filters for Decimation and Interpolation, IEEE Transactions of Acoustics, Speech, and Signal Processing, Vol. ASSP-29 No 2, April 1981

9 Ordering Information

9.1 Ordering Codes

Order CoreCIC through your local Microsemi sales representative. Use the following number convention when ordering: CoreCIC -XX. XX is listed in the following table.

Table 7 Ordering Codes

XX	Description
RM	RTL for RTL source — multiple-use license