Cascaded Integrator-Comb (CIC) Filter V3.0

March 14, 2002 Product Specification

Xilinx, Inc.
2100 Logic Drive
San Jose, CA 95124
Phone: +1 408-559-7778
FAX: +1 408-559-7114
URL: www.xilinx.com/ipcenter
Support: www.support.xilinx.com

Features

- Drop-in module for Virtex™, Virtex™-E, Virtex™-II, Virtex-II Pro™, Spartan™-II, and Spartan™-IIE FPGAs
- 1- to 32-bit input data precision
- Supports decimation and interpolation rate changes between 8 and 16,383
- Dual-channel (time-shared) support for decimator structures
- Number of CIC stages programmable between 1 and 8
- Multiplierless filter architecture is ideal for systems requiring compact filters operating at high sample rates
- Uses relationally placed macro (RPM) mapping and placement technology, for maximum and predictable performance
- Incorporates Xilinx Smart-IP™ technology for utmost parameterization and optimum implementation
- To be used with version 4.2i or later of the Xilinx CORE Generator System

General Description

Cascaded integrator-comb (CIC), or Hogenauer filters, are multirate filters used for realizing large sample rate changes in digital systems. Both decimation and interpolation structures are supported by the Core. CIC filters are multiplierless structures, consisting of only adders, subtractors and registers. They are typically employed in applications that have a large excess sample rate. That is, the system sample rate is much larger than the bandwidth occupied by the signal. CIC filters are frequently used in digital down-converters (DDCs) and digital up-converters (DUCs).

Applications

- Channelization functions in a digital radio or MODEM
- Part of the digital up-conversion signal processing chain in a transmitter
- Any filter structure that is required to efficiently effect a large sample rate change

Theory of Operation

The following description of the CIC decimator and interpolator is based closely on that provided in Reference 1 in the References section later in this document.

CIC Decimator

Figure 1 shows the basic structure for a CIC decimation filter. The integrator section consists of N ideal integrator stages operating at the high sampling rate f_S. Each stage is implemented as a one-pole filter with a unity feedback coefficient. The transfer functions for a single integrator is

$$H_I(z) = \frac{1}{1 - z^{-1}} \qquad (1)$$

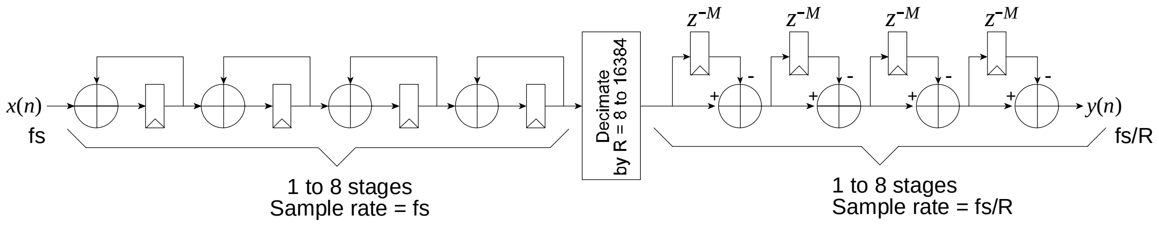


Figure 1: CIC Decimation Filter. The CIC decimator consists of a cascade of integrators followed by a resampling switch and a cascade of differentiators. The differential delay M in the differentiator chain may be defined by the user to be either 1 or 2.

The comb section operates at the low sampling rate f_S / R where R is the integer rate change factor. This section consists of comb stages with a differential delay of M samples per stage. The differential delay is a filter design parameter used to control the filter's frequency response. M is restricted to be either 1 or 2. The transfer function for a single comb stage, referenced to the high input sample rate is

$$H_C(z) = 1 - z^{-RM} \tag{2}$$

There is a rate change switch (indicated in the figure as the *decimation* function) between the two filter sections. The decimator subsamples the output of the last integrator stage, reducing the sample rate from f_S to f_S/R. The system transfer function for the composite CIC filter, referenced to the high sampling rate, is

$$\begin{aligned} H(z) &= H_I^N(z) H_C^N = \frac{(1-z^{-RM})^N}{(1-z^{-1})^N} \\ &= \left[\sum_{k=0}^{RM-1} z^{-k}\right]^N \end{aligned} \tag{3}$$

From the last form of the CIC transfer function, we observe that the CIC filter is equivalent to a cascade of N uniform FIR (finite impulse response) filter stages with unit coefficients; that is, the filter is equivalent to a cascade of N boxcar filters.

To ensure high system clock frequencies, the CIC decimator is actually implemented using the pipelined architecture in Figure 2. The pipeline registers P0, P1, P2 and P3 shorten the critical path through the differentiator cascade of the basic architecture shown in Figure 1. The pipelined implementation has only a single adder between registered nodes.

CIC Interpolator

Exchanging the integrator cascade with the differentiator cascade, as shown in Figure 3, produces a CIC interpolator. Data is presented to the filter at the rate f_S / R where it is processed by the differentiators. The rate expander in the figure causes a rate increase by a factor R by inserting R-1 zero valued samples between consecutive samples of the comb section output. The up-sampled and filtered data stream is presented to the output at the sample rate f_S. Just like the CIC decimator, the Core implementation uses a pipelined structure as shown in Figure 4.

Frequency Response

CIC filters have a lowpass frequency characteristic. The frequency response is obtained by evaluating Eq. (3) at

$$z = e^{\frac{j2\pi f}{R}} \tag{4}$$

where f is the frequency relative to the low sampling rate f_S / R. As part of the filter design process R, M, and N are chosen to provide acceptable passband characteristics over the frequency range from zero to a predetermined cutoff frequency f_S expressed relative to the low sampling rate. Evaluating Eq. (3) in the z-plane at the sample points defined by Eq. (4) gives the magnitude frequency response as

$$|H(f)| = \left[\frac{\sin \pi M f}{\sin \frac{\pi f}{R}}\right]^N \tag{5}$$

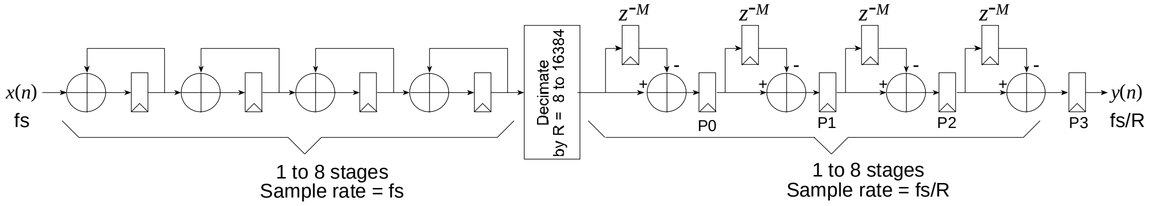


Figure 2: Pipelined CIC Decimator

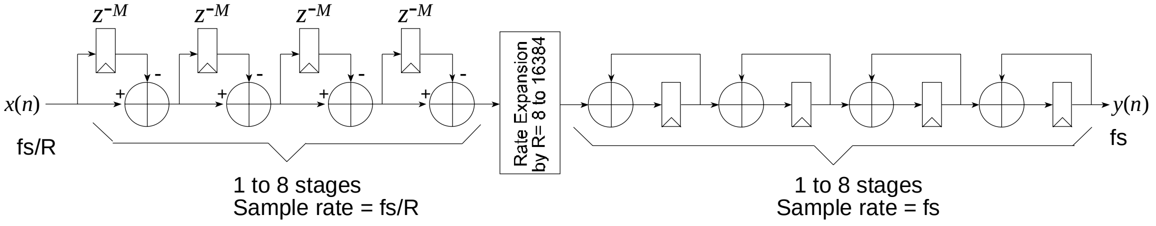


Figure 3: CIC Interpolation Filter

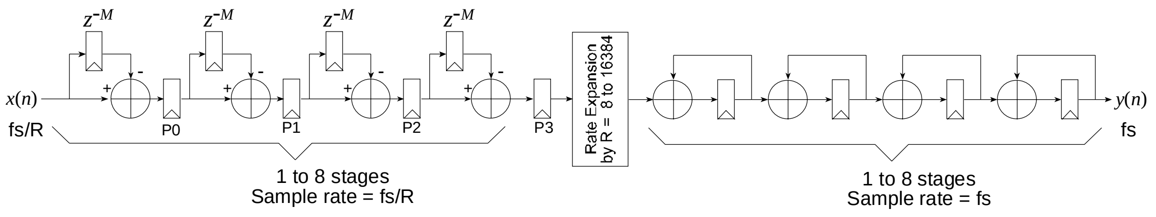


Figure 4: Pipelined CIC Interpolation Filter

Eq. (5) indicates that there a nulls (transfer functions zeros) at integer multiples of $f = 1/M$. Thus, the differential delay parameter, M, can be used as a design parameter to control the placement of the filters zeros. Figure 5 shows the frequency response of a 4-stage ($N = 4$) CIC filter with unity differential delay ($M = 1$) and a sample rate change $R = 7$. When employed as a decimator, the frequency bands in the interval

$$\frac{k}{M} \pm f_c \qquad k = 1,2,\ldots,\lfloor R/2 \rfloor \tag{6}$$

will alias back into the filter passband. Care must be taken to ensure that the integrated sidelobe levels do not impact the intended application.

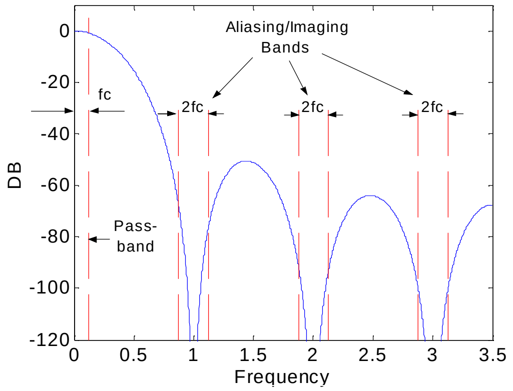


Figure 5: CIC filter frequency response for $N = 4$, $M = 1$, $R = 7$ and $f_c = 1/8$.

Figure 6 shows the frequency response for a filter with virtually the same parameters as that of Figure 5, but in this case the differential delay is $M = 2$. The transmission zeros are now located at integer multiples of 1/M = ½, as is clearly observed in the plot.

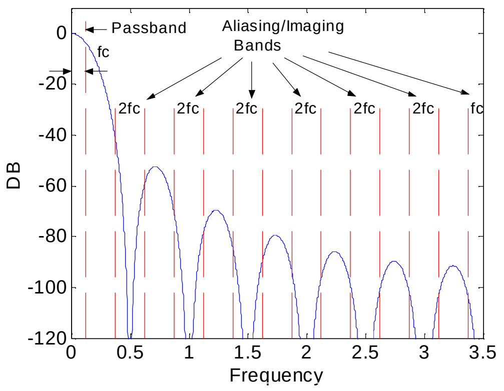


Figure 6: CIC filter frequency response for $N = 4$, $M = 2$, $R = 7$ and $f_c = 1/8$.

Figures 7 to 15 show the frequency response plots for various combinations of the design parameters N, R and M. Figures 9 to 13 provide insight into the filter behavior as the number of integrator and differentiator (N) stages are varied. In all these cases, the differential delay is held constant at $M = 1$ and the sample rate change is fixed at $R = 48$. For any CIC filter, there are always RM zeros in the transfer function. The zeros are equally spaced around the unit cirlce in the z-plane at integer multiples of $1/RM$ - there is of course no zero at $z = 0$. Increasing N has the effect of increasing the order of the zeros. This, in turn, increases the attenuation at frequencies in the locality of the zero. This effect is clearly illustrated in Figures 9 to 13 where we see increasing attenuation of the filter sidelobes as N is increased. Also note that as the order of the zeros increase, the passband droop also increases, thus narrowing the filter bandwidth. The increased droop may not be acceptable in some applications. The droop is frequently corrected using an additional (non-CIC-based) stage of filtering after the CIC decimator. In the case of a CIC interpolator, the signal may be precompensated to account for the impact in the passband as the signal is up-sampled by the CIC interpolator.

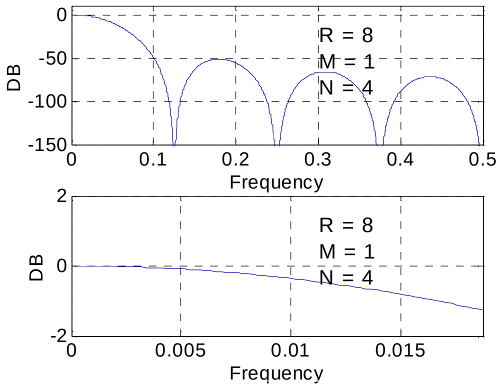


Figure 7: CIC filter frequency response for *N = 4, M = 1, R = 8.*

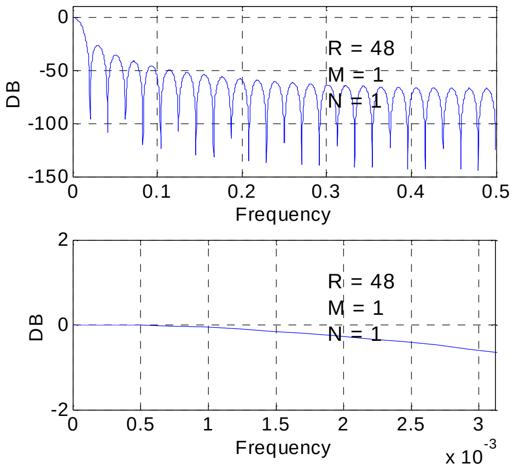


Figure 9: CIC filter frequency response for *N* = 1, *M* = 1, *R* = 48.

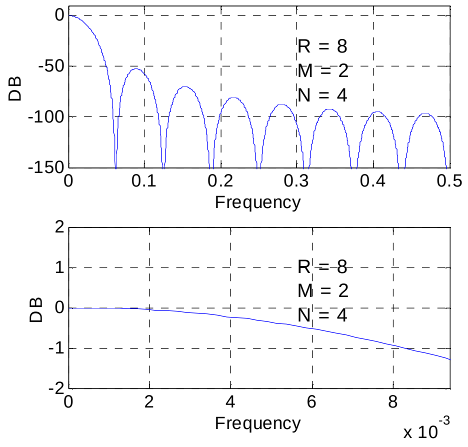


Figure 8: CIC Filter Frequency Response for *N* = 4, *M* = 2, *R* = 8.

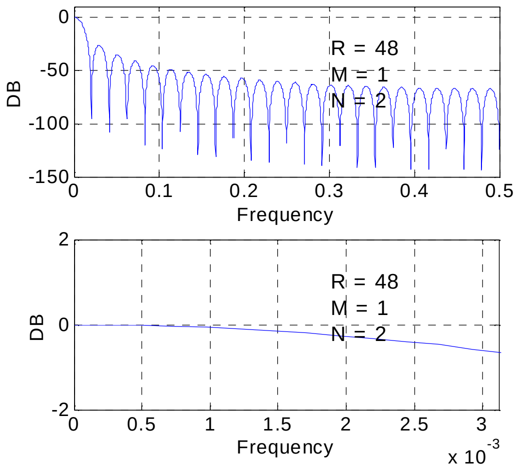


Figure 10: CIC filter frequency response for *N* = 2, *M* = 1, *R* = 48.

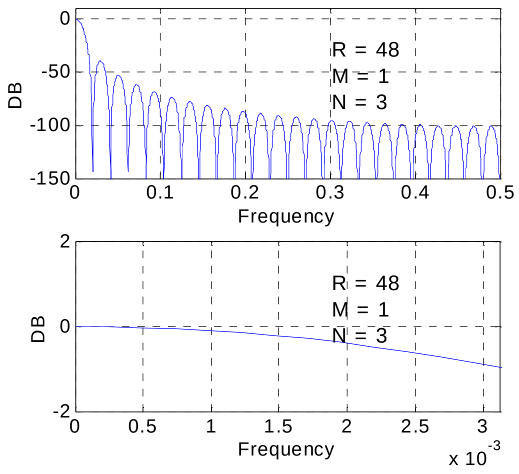


Figure 11: CIC filter frequency response for *N* = 3, *M* = 1, *R* = 48.

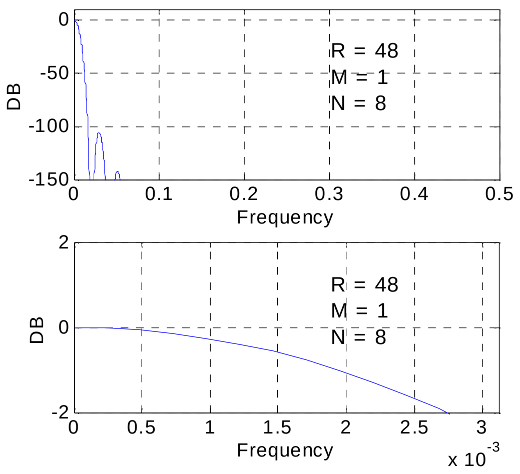


Figure 13: CIC filter frequency response for *N* = 8, *M* = 1, *R* = 48.

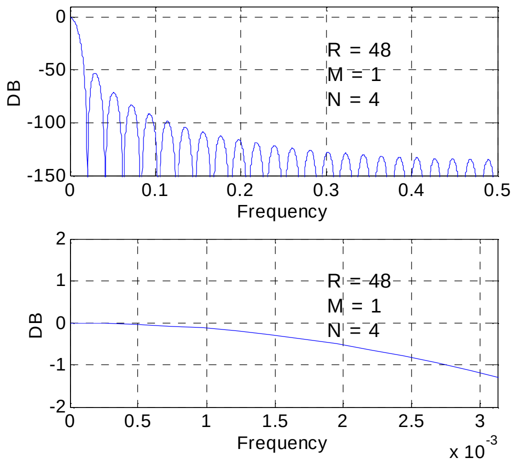


Figure 12: CIC filter frequency response for *N* = 4, *M* = 1, *R* = 48.

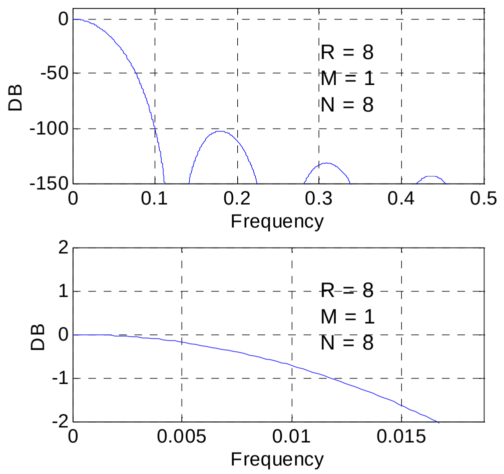


Figure 14: CIC filter frequency response for *N* = 8, *M* = 1, *R* = 8.

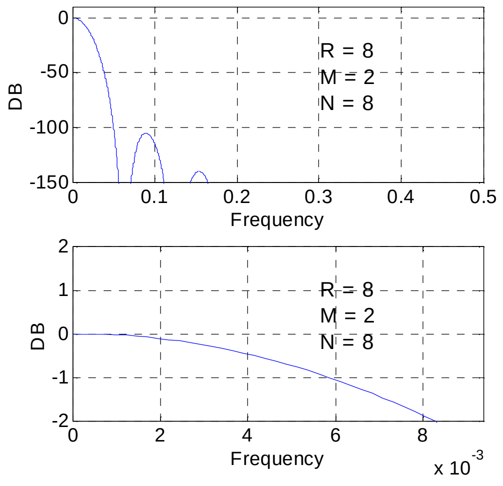


Figure 15: CIC filter frequency response for *N* = 8, *M* = 2, *R* = 8.

A compensation filter (not part of the CIC Core) can be used to flatten the passband frequency response. For a CIC decimator, the compensation filter operates at the decimated sample rate. The compensation filter provides $(x/\sin(x))^N$ shaping. An example of a 3rd order (*N* = 3) *R* = 64 compensated CIC system is shown in Figure 16. The plot shows the uncompensated CIC frequency response, the compensation filter frequency response and the compensated CIC. In this case, since the number of CIC stages is 3, the compensation filter has a cubic response of the form $(x/\sin(x))^3$.

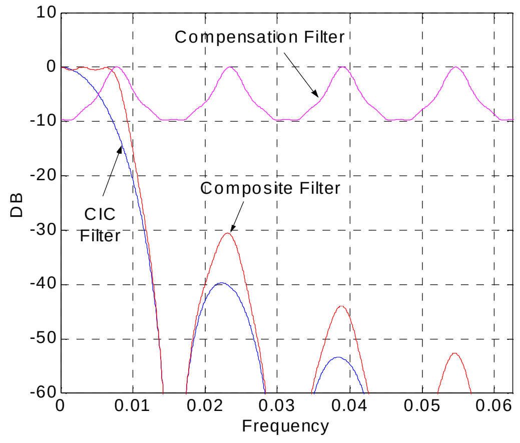


Figure 16: CIC compensation filter for *N* = 3, *R* = 64.

The compensation filter coefficients employed were

[-1, 4, 16, 32, –64, 136, –352, 1312, –352,

136, –64, 32, –16, 4, -1]

Figure 17 provides an exploded view of the compensated filter passband.

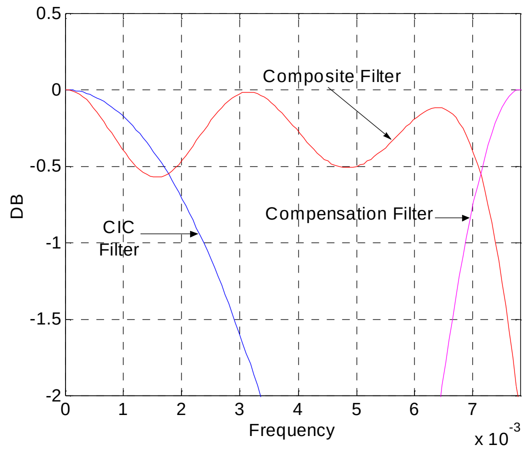


Figure 17: CIC compensation filter. *N* = *3, R* = *64*. Passband response.

Register Growth

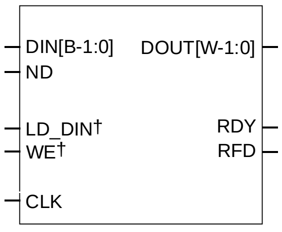
The CIC datapath undergoes internal register growth that is a function of all the design parameters: *N*, *M*, *R,* in addition to the input sample precision *B*. As shown in Reference 1, the most significant bit B_{MAX} at the filter output is defined by

$$B_{\text{MAX}} = \lceil N \log_2 RM + B - 1 \rceil \tag{7}$$

where $\lceil \cdot \rceil$ denotes the ceiling operator. The CIC core uses B_{MAX} bits internally for each of the integrator and differentiator stages, and so produces a full-precision result at the filter output port.

Core Symbol and Port Definitions

Figure 18 is a schematic representation of the CIC core. The port names are the same for both the decimation and interpolation mode of operation.



† Optional pin

Figure 18: CIC symbol. The symbol is the same for the decimator and interpolator.

The Core port definitions are provided in Table 1.

Table 1: CIC Core Ports and Definitions

Signal Name	Direction	Description
CLK	Input	**Clock Master Clock** (active rising edge)
DIN	Input	**Data Input**: B-bit wide filter input port.
ND	Input	**New Data** (active high): When this signal is asserted the data sample presented on the *DIN* port is loaded into the filter.
DOUT	Output	**Filter Output Sample**: W-bit wide filter output sample bus.
RDY	Output	**Filter Output Sample Ready** (active high): Indicates that a new filter output sample is available on the *DOUT* port.
RFD	Output	**Ready for Data** (active high): Indicates when the filter can accept a new input sample.
LD_DIN	Input	***LD_DIN* input bus**: used to supply the sample rate change value when the programmable rate change option for the CIC decimator is selected. *LD_DIN* is an optional port and is only available with decimation filters.
WE	Input	**Write Enable signal** (active high): This signal is associated with the *LD_DIN* port. The value on the *LD_DIN* bus is latched by the core on the rising edge of the clock qualified with *WE*=high*.* Just like *LD_DIN*, *WE* is an optional port and may only be selected for decimation filters.

Interface, Control, and Timing

The CIC filter employs a data-flow style interface for supplying input samples to the Core and for reading the filter output port. *ND (New Data), RFD (Ready For Data)* and *RDY (Ready)* are used to coordinate I/O operations. The Core output status signal *RFD* signals to the system that the filter is ready for data. *RFD* is active high. Asserting *ND* high indicates to the Core the availability of a new input sample on the *DIN* port. *ND* is very similar in functionality to the *clock enable* signal found on many VLSI (very large scale integrated-circuit) devices. The *RDY* output signal indicates that a new filter output sample is available on the *DOUT* port.

The interface signals would typically be used in the following manner. The user system would first wait for *RFD*=1. This signals that a new input sample can be written to the filter. The new input sample would be placed on the *DIN* port and *ND* would be placed in the active state (*ND*=1) for a single clock cycle. Asserting *ND* indicates to the Core that it should sample the *DIN* port. The filter will sample *DIN* on the rising edge of the clock (*CLK*) qualified with *ND*=1. A filter read operation can occur when the Core asserts *RDY*=1. *RDY* could be used as a clock enable signal for a down-stream processing block that is consuming the filter output samples.

The timing for a CIC *R*-fold decimator is shown in Figure 19. In this example a new input sample is applied on every clock edge, so *ND* must be continuously held high. Some number of clock cycles after the first input sample has been written to the filter *RDY* will be asserted by the filter to indicate that the first output sample is available. This time interval is a function of the down-sampling factor *R* and a fixed latency that is related to internal pipeline registers in the Core. The number of pipeline stages depends on the core customization parameters. After the very first output sample has been produced, subsequent outputs will be available every *R* clock cycles. It is strongly recommended that designers employ the *RDY* signal as a gating signal for any processes that consume the filter output samples.

Figure 20 shows the timing for an *R*-fold decimator where *ND* is being used to control the flow of input samples to the Core. The timing diagram shows new input samples being written to the Core every two clock cycles. New output samples will become available every 2*R* clock cycles. As with the timing diagram in Figure 20, the timing of the first output sample will include an additional temporal component that is the Core startup latency.

Figure 21 shows the timing for a CIC interpolator with an up-sampling factor *R*=8. A new input sample is written to the filter every 8th period of the master clock. After the initial start-up latency, *RDY* is asserted, and a new filter output is available on every subsequent clock edge: for every input delivered to the Core, 8 outputs (or *interpolants*) are generated.

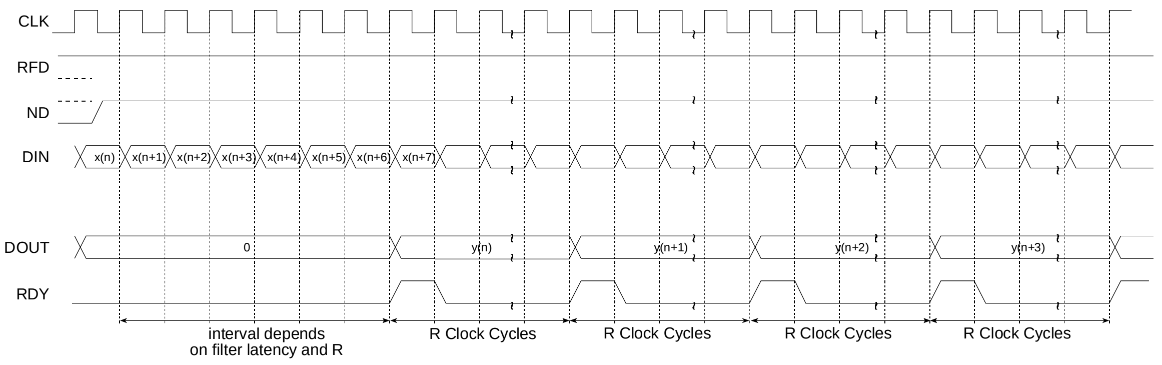


Figure 19: CIC decimator timing. The example shows a new input data sample supplied to the Core on every clock cycle.

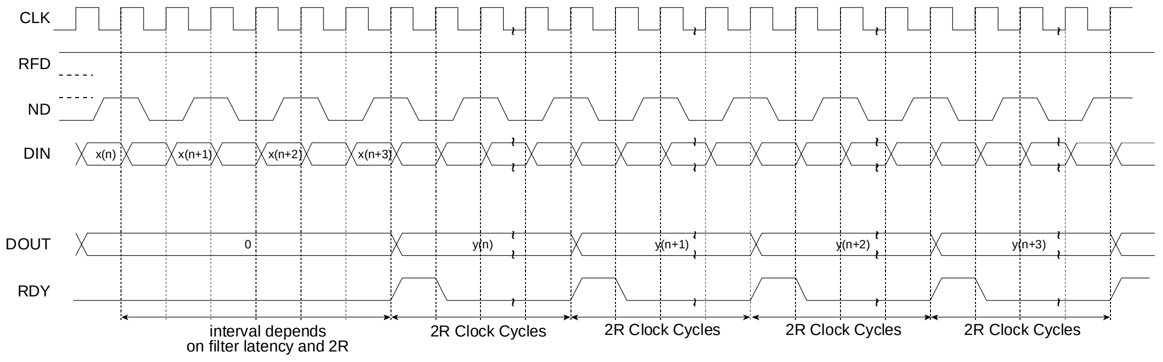


Figure 20: CIC decimator timing. The example shows a new input data sample supplied to the Core every two clock cycles. *ND* is used to control the flow of input samples to the filter. Filter results are available every 2*R* clock cycles.

Figure 22 shows the timing for the same filter demonstrated in Figure 21, but in this case new samples are written to filter every 9 rather than 8 clock cycles. Note that this is reflected in the behavior of the *RDY* signal. *RDY* pulses low for one clock cycle every 8 clock periods, indicating that a new output is *not* available on the associated clock edge. Also observe that the filter output maintains the previous output samples on the *DOUT* port.

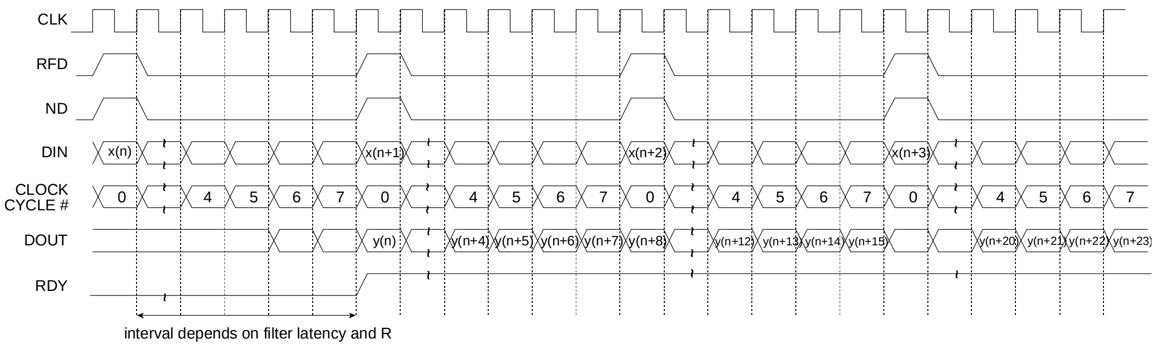


Figure 21: CIC interpolator timing. The example employs an up-sampling factor *R*=8. Input samples are supplied every 8 clock cycles.

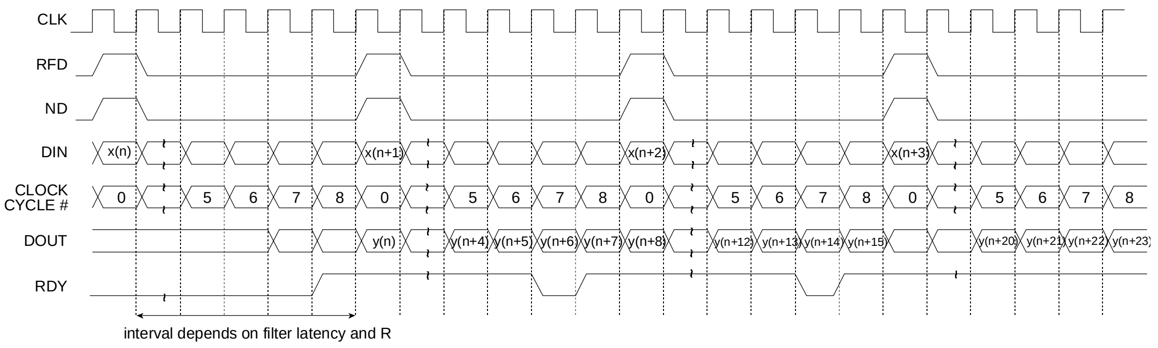
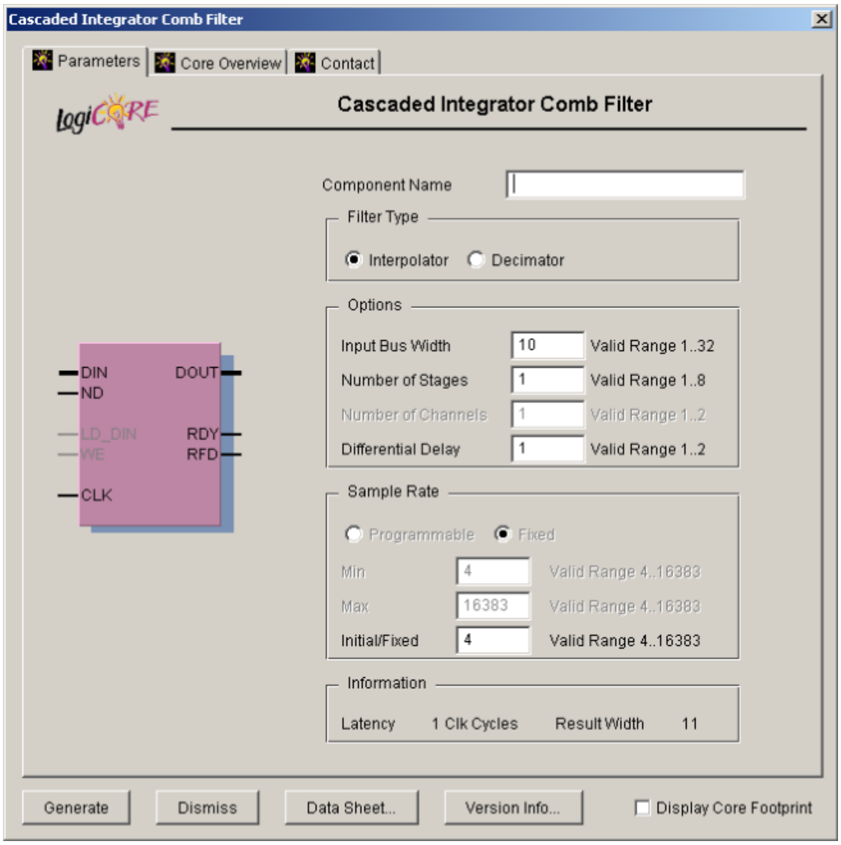


Figure 22: CIC interpolator timing. The example employs an up-sampling factor *R*=8. Input samples are supplied every 9 clock cycles.



Cascaded Integrator Comb Filter

Component Name

Filter Type

[✓ Interpolator] Decimator

Options

Input Bus Width	10	Valid Range 1..32
Number of Stages	1	Valid Range 1..8
Number of Channels	1	Valid Range 1..2
Differential Delay	1	Valid Range 1..2

Sample Rate

Programmable [✓ Fixed]

Min	4	Valid Range 4..16383
Max	16383	Valid Range 4..16383
Initial/Fixed	4	Valid Range 4..16383

Information

Latency 1 Clk Cycles Result Width 11

Display Core Footprint

Figure 23: CIC Parameterization Screen

Multi-Channel Decimator

The CIC decimation filter provides a hardware efficient implementation of a 2 channel filter. When this configuration is selected most of the hardware in the filter is time-division multiplexed to service two channels. Thus, 2 input streams can be processed with a dual-channel filter using less FPGA logic resources than a similar design that employs two separate filters. Note that due to the time shared nature of the hardware, the sample rate of each input stream will be reduced by a factor of two for the dual channel case. For example, if the master clock frequency to the CIC decimator is 100 MHz, a single channel decimator can support one sample stream with a sample rate of 100 MHz. In the dual-channel mode of operation, two 50 MHz input streams can be processed. Dual-channel capability is not supported by the CIC interpolator.

Figure 24 shows the timing for a dual-channel decimator. The two input streams *x* and *v* are supplied to the filter in an interleaved manner on the *DIN* port. *ND* must be high in order for the input samples to be accepted by the filter. In this example both the *x* and *v* streams are supplied in a continuous manner and so *ND* is held permanently high.

After the filter startup latency, the decimated output streams, *y* and *z,* corresponding to the inputs *x* and *v* respectively, will be presented on the *DOUT* port. As described earlier, the *RDY* status signal indicates the availability of a new output sample. In this case *RDY* is active (high) for two clock cycles and brackets the two new output samples as shown in Figure 24. From this point on, a new pair of output samples, one for each of the two input streams, become available every 2*R* clock cycles, where *R* is the decimation factor.

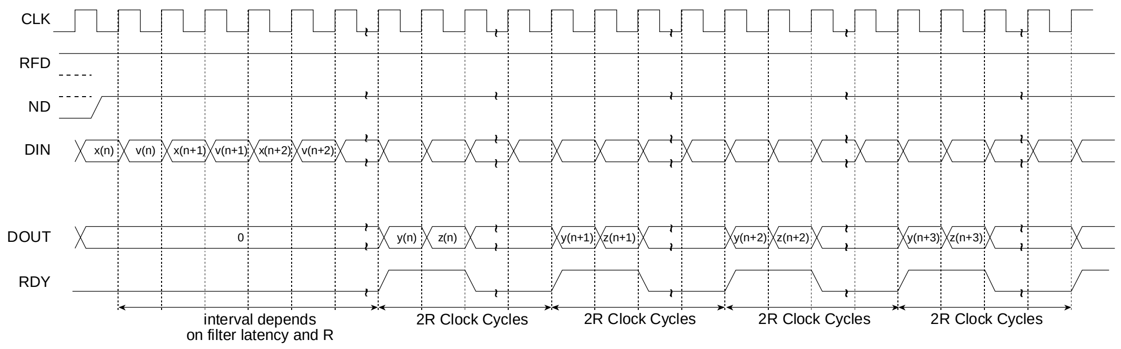


Figure 24: Dual-Channel CIC Decimator Timing

Parameters

The CIC parameterization screen is shown in Figure 23. The customization parameter definitions are:

- **Component Name:** The user defined CIC filter component name.
- **Input Bus Width:** Filter input data port.
- **Number of Stages:** Number of integrator and comb sections. If *N* stages are specified there will be *N* integrators and *N* differentiators in the filter.
- **Sample Rate Change:** Sample rate change *R.* For an interpolation filter this value specifies the up-sampling factor. For a decimator it is the down-sampling factor.
- **Differential Delay:** The number of unit delays employed in each differentiator section of either a decimator or interpolator.
- **Filter Type:** The CIC Core supports both interpolator and decimator architectures. When the filter type is selected as *decimator* the input sample stream is down-sampled by the factor *R.* When an *interpolator* is selected the input sample is up-sampled by *R.*
- **number_of_channels:** The CIC decimation filters support an optional dual channel mode of operation. When the number of channels is defined as 2, certain hardware modules in the CIC filter structure are time division multiplexed. This permits a more hardware efficient FPGA implementation in comparison to a system that would use two separate filters to implement the same function as a single dual-channel CIC. Multi-channel CIC interpolators are not supported by the core.

XCO File Parameters

The parameters supplied via the filter GUI are captured and logged to the *.xco* file. The full name of this file is simply the *Component Name* with an *.xco* file extension. Table 2 defines the .xco file parameter names and range specifications. Figure 25 is an example .xco file.

Table 2: XCO file parameter names, definitions, and range specifications

Parameter Name	Definition	Range
BusFormat	Controls the notation employed for identifying buses in the output edif netlist file.	{BusFormatAngleBracket | BusFormatParen}
SimulationOutputProducts	Core HDL simulation selection – either VHDL or Verilog.	{VHDL | VERILOG}
XilinxFamily	The FPGA target device family.	{Virtex | Virtex2 | Virtex2P | Spartan2 | Spartan2}
DesignFlow	HDL flow specifier.	{VHDL | VERILOG}
FlowVendor	Design flow vendor information.	{Other | Synplicity | Exemplar | Synopsis | Foundation | Innoveda | ISE}
component_name	Component name as defined by the user.	Any valid file name for the user's operating system consisting of the letters a…z, 0…9 and '_'. The component name may be a maximum of 32 characters.
sample_rate_change	Sample rate change for decimator or interpolator. For the decimator this value defines the down-sampling factor. For the interpolator it defines the up-sampling factor.	[4,…,16383]
number_of_stages	Number of integrator and differentiator stages in the CIC filter.	[1,…,8]
number_of_channels	The CIC decimator supports single or dual channel modes of operation. In the dual channel mode of operation the one piece of hardware is time division multiplexed across the two input sample streams.	{1 | 2}
filter_type	Defines if a CIC decimator or interpolator is generated.	{decimator | interpolator}
differential_delay	Integer delay value in the differentiator stage of either a decimator or interpolator.	[1,2]
input_bus_width	Bit precision of the filter data input port.	[1,…,32]
create_rpm	Controls if the Core is generated with embedded placement information.	{true | false}

```
# Xilinx CORE Generator 4.2i; Cores Update # 2
# Username = chrisd
# COREGenPath = C:\Xilinx\coregen
# ProjectPath = C:\ip_portfolio\CIC\datasheet\eip2\xilinx
# ExpandedProjectPath = C:\ip_portfolio\CIC\datasheet\eip2\xilinx
# OverwriteFiles = True
# Core name: cic
# Number of Primitives in design: 864
# Number of CLBs used in design cannot be determined when there is no RPMed logic
# Number of Slices used in design cannot be determined when there is no RPMed logic
# Number of LUT sites used in design: 192
# Number of LUTs used in design: 192
# Number of REG used in design: 305
# Number of SRL16s used in design: 0
# Number of Distributed RAM primitives used in design: 0
# Number of Block Memories used in design: 0
# Number of Dedicated Multipliers used in design: 0
# Number of HU_SETs used: 0
#
SET BusFormat = BusFormatParen
SET SimulationOutputProducts = VHDL
SET XilinxFamily = Virtex2
SET OutputOption = DesignFlow
SET DesignFlow = VHDL
SET FlowVendor = Synplicity
SET FormalVerification = None
SELECT Cascaded_Integrator_Comb_Filter Virtex2 Xilinx,_Inc. 3.0
CSET input_bus_width = 10
CSET differential_delay = 1
CSET number_of_stages = 4
CSET number_of_channels = 1
CSET component_name = cic
CSET sample_rate_change = 10
CSET sample_rate_change_min = 4
CSET filter_type = Decimator
CSET sample_rate_change_type = Fixed
CSET sample_rate_change_max = 16383
GENERATE
```

Figure 25: Example .xco file

Area and Performance

Table 3 documents the FPGA logic requirements in terms of slices as a function of the number of CIC stages and the input sample precision *B*. This table was generated for a CIC decimator with a sample rate change *R* = 48.

The area data in Table 3 is presented graphically in Figure 26 for *N* = 1, 2, 3 and 4, while Figure 27 is for *N* = 5, 6, 7 and 8. The knee in the curves for *N* = 6, 7 and 8 is associated with additional pipelining of long carry chains in the adders and subtractors when the internal datapath precision is reasonably large.

Table 4 presents CIC logic area data as a function of the number of CIC stages and the input sample precision *B* for a sample rate change *R* = 4096.

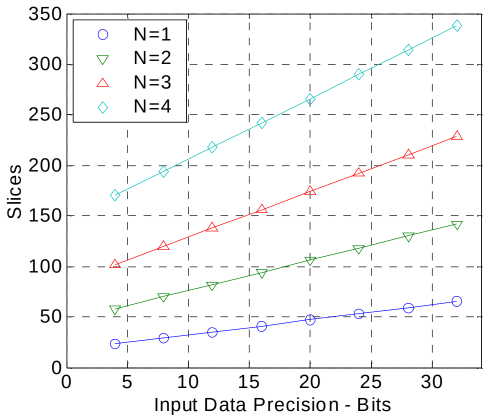


Figure 26: CIC slice count versus input sample precision with the number of CIC stages *N* as a parameter. N = 1, 2, 3, 4. *R* = 48.

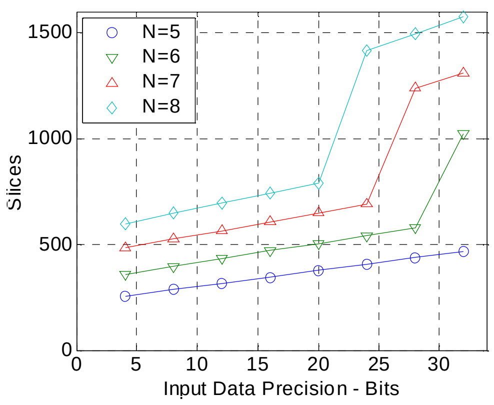


Figure 27: CIC slice count versus input sample precision with the number of CIC stages *N* as a parameter. N = 5, 6, 7, 8. *R* = 48.

Table 3: CIC Slice Count as a Function of the Number of CIC Stages *N* and the Input Sample Precision *B*. *R* = 48.

N	Slice Count							
	B = 4	B = 8	B = 12	B = 16	B = 20	B = 24	B = 28	B = 32
1	23	29	35	41	47	53	59	65
2	58	70	82	94	106	118	130	142
3	102	120	138	156	174	192	210	228
4	170	194	218	242	266	290	314	338
5	256	286	316	346	376	406	436	466
6	360	396	432	468	504	540	576	1020
7	482	524	566	608	650	692	1238	1308
8	598	646	694	742	790	1414	1494	1574

Table 4: CIC Slice Count as a Function of the Number of CIC Stages *N* and the Input Sample Precision *B*. *R* = 4096.

N	Slice Count							
	B = 4	B = 8	B = 12	B = 16	B = 20	B = 24	B = 28	B = 32
1	36	42	48	54	60	66	72	78
2	98	110	122	134	146	158	170	182
3	196	214	232	250	268	286	304	538
4	330	354	378	402	714	754	794	834
5	500	890	940	990	1040	1090	1140	1190
6	1186	1246	1306	1366	1426	1486	1546	1606
7	1592	1662	1732	1802	1872	1942	2012	2082
8	2058	2138	2218	2298	2378	2458	2538	2618

Table 5 presents CIC logic area data as a function of the number of CIC stages and the input sample precision *B* for a sample rate change *R* = 16.

Table 5: CIC Slice Count as a Function of the Number of CIC Stages *N* and the Input Sample Precision *B*. *R* = 16.

N	Slice Count							
	B = 4	B = 8	B = 12	B = 16	B = 20	B = 24	B = 28	B = 32
1	18	24	30	36	42	48	54	60
2	44	56	68	80	92	104	116	128
3	82	100	118	136	154	172	190	208
4	132	156	180	204	228	252	276	300
5	194	224	254	284	314	344	374	404
6	268	304	340	376	412	448	484	520
7	354	396	438	480	522	564	606	648
8	452	500	548	596	644	692	740	788

Table 6 presents CIC logic area data as a function of the number of CIC stages and the input sample precision *B* for a sample rate change *R* = 32.

Table 6: CIC Slice Count as a Function of the Number of CIC Stages *N* and the Input Sample Precision *B*. *R* = 32.

N	Slice Count							
	B = 4	B = 8	B = 12	B = 16	B = 20	B = 24	B = 28	B = 32
1	20	26	32	38	44	50	56	62
2	52	64	76	88	100	112	124	136
3	93	111	129	147	165	183	201	219
4	158	182	206	230	254	278	302	326
5	226	256	286	316	346	376	406	436
6	324	360	396	432	468	504	540	576
7	419	461	503	545	587	629	671	1224
8	550	598	646	694	742	790	1414	1494

Table 7 presents performance information, in terms of maximum Core clock frequency for Virtex-E and Virtex-II devices.

Table 7: CIC Performance Data for Virtex-E and Virtex-II. *N* = 4, *R* = 48.

Device	Performance (MHz)			
	B = 8	B = 12	B = 16	B = 20
Virtex-E -08 speed grade	162	162	162	162
Virtex-II -06 speed grade	208	203	191	191

CIC Design Example

CIC filters are frequently employed in digital down and up converters. The following design example shows how the Xilinx CIC core can be used to implement channelization functions in a digital receiver.

Virtually all digital receivers perform channel access using a digital down-converter (DDC). Modern basestation transceivers will often require a large number of DDCs to support multicarrier environments or for coherently down-converting and combining a number of narrow-band channels into one wide-band digital signal. The DDC is typically located at the front-end of the signal processing conditioning chain, close to the A/D, and is usually required to support high-sample rate processing in the region of 100 to 200 mega-samples-per-second. The high data rate, coupled with the large arithmetic workload, are not well suited for DSP microprocessor implementation. Application specific standard products (ASSP) are a common solution. A more flexible, and typically higher-performance alternative, is to implement the DDC using programmable logic. Since DDC functions only require a modest amount of FPGA silicon resources, many other receiver functions can be implemented in the same device. Consider the implementation of a single channel of the GC4016 quad digital receiver 2. A simplified block diagram is shown in Figure 29.

The desired channel is translated to baseband using the digital mixer comprising the multipliers *M1, M2* and a direct digital synthesizer (DDS). The sample rate of the signal is then adjusted to match the channel bandwidth. This is performed using a multi-stage multi-rate filter consisting of the filters $C(z)$,$G(z)$ and $H(z)$. The functions performed in the system are waveform synthesis (DDS), complex multiplication and multirate filtering. We will implement an FPGA version of the DDC that is suitable for GSM wireless applications. The spectral mask requirement for GSM is shown in Figure 28. The input sample rate is chosen to be 52 MHz. The GSM channel can be supported with an output sample rate of 270.8333 kHz. This corresponds to a sample rate change of 192.

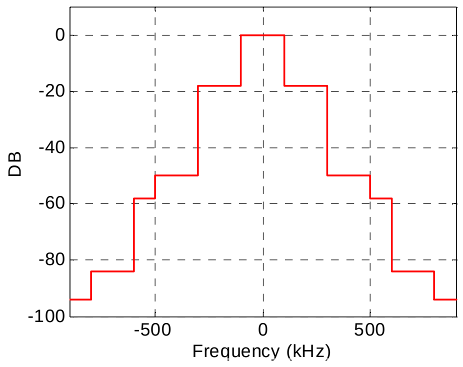
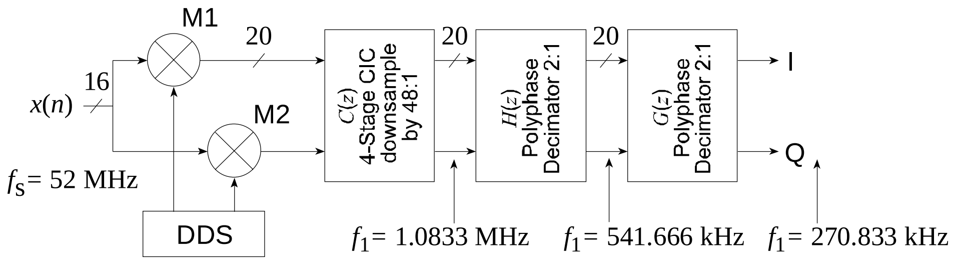
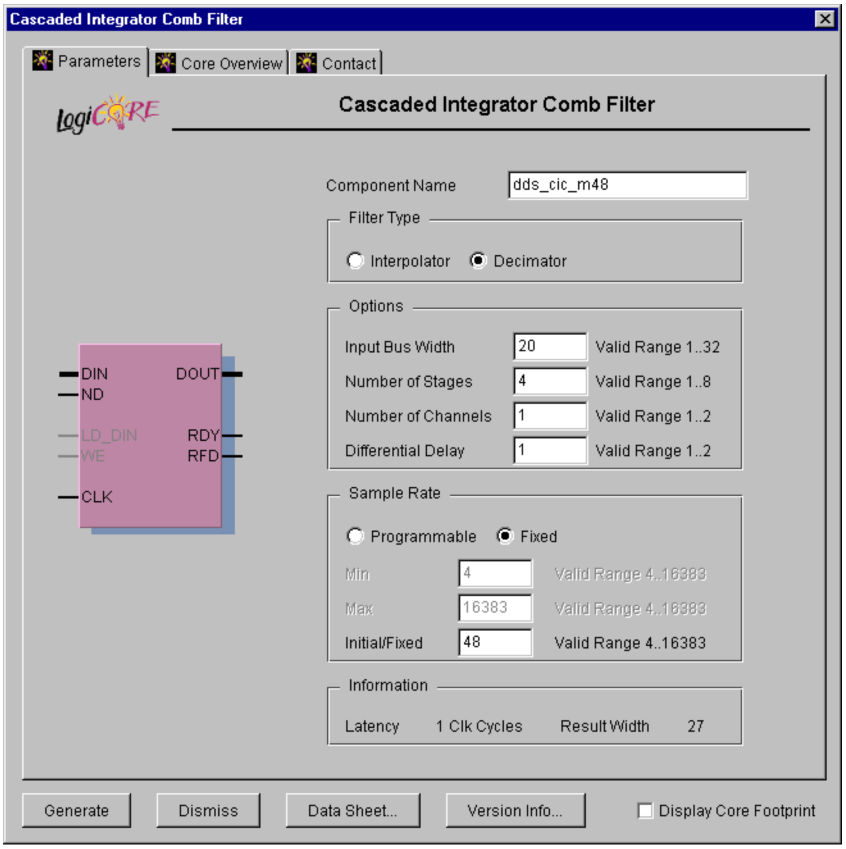


Figure 28: GSM spectral mask.



M1
M2
20
20
20
16
$x(n)$
f_S= 52 MHz
DDS
$C(z)$ 4-Stage CIC downsample by 48:1
$H(z)$ Polyphase Decimator 2:1
$G(z)$ Polyphase Decimator 2:1
I
Q
f_1= 1.0833 MHz
f_1= 541.666 kHz
f_1= 270.833 kHz

Figure 29: Digital down converter for GSM applications. A CIC decimator is employed to adjust the system sample rate to match the GSM channel bandwidth.



Cascaded Integrator Comb Filter

Component Name: dds_cic_m48

Filter Type: ( ) Interpolator [✓ Decimator]

Options

Input Bus Width	20	Valid Range 1..32
Number of Stages	4	Valid Range 1..8
Number of Channels	1	Valid Range 1..2
Differential Delay	1	Valid Range 1..2

Sample Rate: ( ) Programmable [✓ Fixed]

Min	4	Valid Range 4..16383
Max	16383	Valid Range 4..16383
Initial/Fixed	48	Valid Range 4..16383

Information

Latency 1 Clk Cycles Result Width 27

☐ Display Core Footprint

Figure 30: CIC User Interface. The module generator is parameterized for input sample precision, sample rate change, and supports up-sampling and down-sampling operations.

The baseband channel is highly oversampled, so a simple cascade of boxcar filters, implemented as a cascaded integrator comb (CIC) [See Reference 1] will be employed to initially reduce the sample rate by a factor of 48. The CIC filter $C(z)$ is multiplierless, consisting only of integrator and differentiator sections. For this application, a cascade of 4 integrators followed by 4 differentiators, with an embedded 48:1 rate change, will be employed. A desirable characteristic of the CIC filter in the contest VLSI design is of course that it is multiplier free. However, as described in Reference 1, the precision required in the integrator and differentiator sections can be nontrivial, growing to 40 bits or more. This can potentially lead to operating speed issues in some technologies because of the serial dependency in the adder/subtractor carry chains. Modern FPGA devices like Virtex-II provide extremely high performance carry chains, and even the long carry-chains that can be required in some CIC filters can be supported at very high speed.

Figure 30 is a screen shot of the CIC user interface with the parameter fields defined for this design. The complex filter $C(z)$ is automatically generated using this approach and consumes 532 logic in total for the I and Q data streams.

The CIC filter is followed by a cascade of two 2:1 polyphase decimators to produce the required input-to-output sample rate change of 192:1. A 21-tap filter is used for the polyphase decimator $G(z)$ while a 63-tap filter is employed for $H(z)$. Figure 31 (a), (b) and (c) illustrate the spectral responses for the filters $C(z)$, $G(z)$, and $H(z)$ respectively. Figure 31 (d) is the aggregate response, obtained by convolving the impulse response of the three filters with each other. We observe that the combined response satisfies the GSM spectral mask requirements.

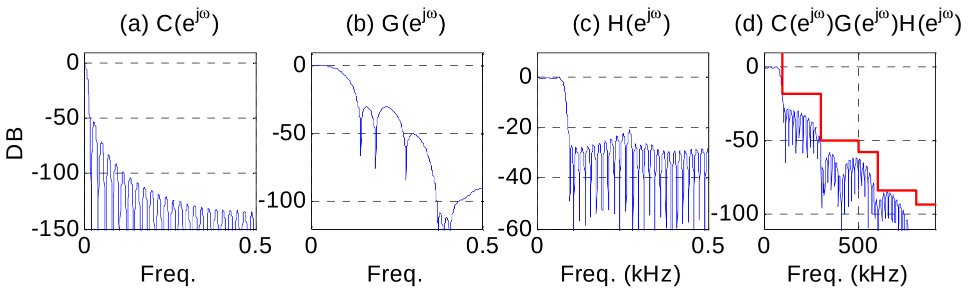


Figure 31: Frequency characterizations. (a) C(z), (b) G(z), (c) H(z), (d) Aggregate frequency response and GSM spectral mask overlay.

The multipliers *M1* and *M2* may be implemented using the *Multiplier* Core that is supplied with the Xilinx Core Generator System. For Virtex and Virtex-E devices the multipliers would be implemented in the logic fabric. A Virtex-II [See Reference 3] implementation would use the embedded hardware multipliers that is part of this FPGA family's architectural definition. Using the optional pipelined mode of operation the embedded Virtex-II multipliers can support samples rates in excess of 200 MHz. In this design, with an input sample rate of 52 MHz, a single multiplier could be time-shared to implement the input heterodyne. The DDS employed is a look-up table-based synthesizer. The FPGA block memory is used to store one quarter of a cycle of a sinusoid. The dual-port memory enables both the in-phase and quadrature components of the local oscillator to be generated simultaneously using a single block RAM. A single Virtex-II block RAM implementation can generate a 4096-sample full-wave 16-bit precision complex sinusoid. (SFDR). The DDS can be realized using the DDS Core (version 2.0 or later) supplied with the Core Generator System.

One very successful technique for implementing filters, including multirate structures, in FPGAs is distributed arithmetic (DA) [See References 4 and 5]. This approach is an option for realizing $G(z)$ and $H(z)$. The functional requirements of a DA filter, large shift registers, lookup tables and accumulation, closely match the silicon resources provided in an FPGA technology like Virtex, Virtex-E and Virtex-II. The two polyphase structures can be realized using parameterizable IP. The DA FIR filter (version 4.0 or later) supplied with the Xilinx Core Generator System can be used to quickly and efficiently implement the required filters. The complex filters $G(z)$ and $H(z)$ occupy 372 and 652 logic slices respectively. To put these figures in context, the Virtex-II family of FPGAs consists of devices with slice counts ranging from 40 to 61,400.

The coefficient files (*coe files*) that specify the filters $G(z)$ and $H(z)$ are supplied in this document's appendix.

References

[1] E. B. Hogenauer, "An Economical Class of Digital Filters for Decimation and Interpolation", *IEEE. Trans. Acoust., Speech Signal Processing*, Vol. 29, No. 2, pp. 155-162, April 1981.

[2] GC4016 Multi-Standard QUAD DDC Chip, Datasheet, Graychip, Pal Alto CA, USA, Aug., 2000.

[3] Xilinx Inc., *Virtex-II Platform FPGA Handbook*, 2000.

[4] Peled and B. Liu, "A New Hardware Realization of Digital Filters", *IEEE Trans. on Acoust., Speech, Signal Processing*, vol. 22, pp. 456-462, Dec. 1974.

[5] S. A. White, "Applications of Distributed Arithmetic to Digital Signal Processing*", IEEE ASSP Magazine*, Vol. 6(3), pp. 4-19, July 1989.

Ordering Information

This core may be downloaded from the Xilinx IP Center for use with the Xilinx CORE Generator System V4.2i and later. The Xilinx CORE Generator System tool is bundled with all Alliance and Foundation Series Software packages, at no additional charge.

To order Xilinx software, please visit the Xilinx Silicon Xpresso Cafe or contact your local Xilinx sales representative.

Information on additional Xilinx LogiCORE modules is available on the Xilinx IP Center.

Appendix

Coefficients for G(z) in GSM DDC Design Example

Figure 32 shows the coefficient file (*coe file*) that specifes the filter $G(z)$ in the GSM DDC design example. This file is compatible with version 4.0 or later of the DA FIR Core supplied with the Xilinx Core Generator System.

```
radix=10;
coefdata=
-98,
-679,
-2016,
-3234,
-2537,
850,
6053,
12060,
18230,
23239,
25212,
23239,
18230,
12060,
6053,
850,
-2537,
-3234,
-2016,
-679,
-98;
```

Figure 32: Coe file for the polyphase 2:1 decimator $G(z)$ in the GSM DDC design example.

Coefficients for H(z) in GSM DDC Design Example

Figure 33 shows the coefficient file (*coe file*) that specifes the filter in the GSM DDC design example. This file is compatible with version 4.0 or later of the DA FIR Core supplied with the Xilinx Core Generator System.

```
radix=10;
coefdata=
1007,
-1853,
79,
1807,
1633,
423,
265,
175,
-527,
-1331,
-1454,
-1087,
-721,
-149,
1008,
1985,
2164,
2005,
1483,
61,
-1756,
-3134,
-3953,
-4016,
-2714,
134,
4003,
8361,
12934,
17009,
19565,
20353,
19565,
17009,
12934,
8361,
4003,
134,
-2714,
-4016,
-3953,
-3134,
-1756,
61,
1483,
2005,
2164,
1985,
1008,
-149,
-721,
-1087,
-1454,
-1331,
-527,
175,
265,
423,
1633,
1807,
79,
-1853,
1007;
```

Figure 33: Coe file for the polyphase 2:1 decimator *H(z)* in the GSM DDC design example.