Features

- High-performance single-rate finite impulse response (FIR), polyphase decimator, and interpolator implementations
- 1 through 1024 taps
- 2 through 32 bits input data precision
- 2 through 32 bits coefficient precision
- Up to 74-bit accumulator width
- Signed or unsigned input data
- Signed or unsigned filter coefficients
- Support for interpolation and decimation factors between 2 and 256 inclusive
- Support for 1 through 256 channels
- Support for 1 through 256 different coefficient sets
- Coefficient symmetry exploited for higher performance and compact implementation
- Capable of using multiple MAC engines to implement a filter
- To be used with v7.1i or later of the Xilinx CORE Generator™ system

Applications

- Upsampling and pulse shaping filter in a modulator (for example, QPSK, 16-QAM, and so forth)
- Image processing, for example, edge detection and general 2-D convolution
- Matched filter (with sample rate change) in a digital receiver
- Digital down and up converters
- Correlation processing

General Description

The Xilinx MAC FIR core implements a highly configurable, high-performance, and area efficient FIR filter. Single-rate polyphase decimators and interpolators are supported. Multiple data channel operation is supported for all filter types. Symmetry in the coefficient set is exploited for single MAC implementations to increase overall performance and minimize resource utilization. All internal data-paths provide full-precision arithmetic to avoid the possibility of overflow. The full-precision sum-of-products is presented on the Data Output port. A set of three handshake control signals provides an easy-to-use user interface. The conventional tapped delay line representation of a FIR filter is shown in Figure 1.

The MAC FIR core uses one or more time-shared multiply accumulate (MAC) functional units to service the N sum-of-product calculations in the filter. The core automatically determines the minimum number of MAC engines required to meet the user specified throughput.

Figure 2 is a block diagram of a single MAC engine. The figure shows storage for the filter coefficients, the filter regressor vector, and the control circuit that sequences the appropriate coefficients and data to the multiply-accumulator for the specified integration period.
Theory of Operation

The MAC FIR core performs a sum-of-products operation to compute the convolution sum. The convolution sum is defined by Equation 1. The core supports three filter functions: single-rate, polyphase decimation, and polyphase interpolation.

Single-Rate

In a single-rate filter, the output result rate is equal to the input sample rate. The filter output $y(k)$ is computed according to Equation 1 where $N$ is the number of filter coefficients $a(i)$ $i = 0,..., N - 1$ are the filter coefficients and $x(n)$ represents the input time-series.

$$y(k) = \sum_{n=0}^{N-1} a(n)x(k - n) \quad \text{Equation 1}$$

Polyphase Decimator

A polyphase decimation filter provides a single output result for every $M$ input sample, where $M$ is referred to as the decimation rate or rate change. Note that the output rate is $1/M$'th of the input rate. The filter implementation exploits the low output sample rate by not starting a computation until a new set of $M$ input samples is available.

Polyphase Interpolator

A polyphase interpolator filter provides $L$ output results for each new input sample, where $L$ is referred to as the interpolation rate or rate change. Note that the output rate is $L$ times the input rate. The filter is implemented as a computationally efficient polyphase architecture.
Filter Symmetry

The impulse response for many practical filters possess symmetry that can be exploited to minimize the number of arithmetic operations required to compute an output.

Figure 3 is the impulse response for a 9-tap symmetric FIR filter. Note that the coefficients are symmetric about the center tap. The number of multiplications can be reduced by first summing terms in the regressor vector that ultimately engage the same filter coefficient value. This reduces the computation workload by almost a factor of two.

![Figure 3: Symmetric FIR, Odd Number of Terms](image)

Figure 4 illustrates the structure. When the core is customized, the user has the option of taking advantage of both symmetric and negative symmetric coefficient data. If symmetry is exploited, the logic resources required by the core are usually slightly higher but the filter throughput is nearly doubled.

![Figure 4: Exploiting Coefficient Symmetry, Odd Number of Filter Taps](image)

Note that for single-rate and decimating filters, the symmetry can be exploited. For interpolating, filter symmetrical coefficients cannot be exploited. For multiple MAC engine filter implementations, symmetric coefficients are not exploited in this revision of the core. Future versions of the core will add this capability.
Multiple MAC Engine Filters

The MAC FIR v5.1 core automatically generates an implementation that meets the user defined throughput requirements based on the system clock rate, sample rate, number taps and channels, and rate change. The core inserts one or more multipliers to meet the overall throughput requirements.

The number of multipliers required for a filter is determined by computing the number of clocks available to process each input sample (A) and then dividing the number of multiplies required to perform the computation (B) by A. The number of clocks available to process each input sample is found by:

\[
A = \text{floor}\left\{\frac{\text{System Clock Rate} \times \text{Decimating Rate Change}}{\text{Input Sample Rate} \times \text{Number Channels} \times \text{Interpolating Rate Change}}\right\}
\]

\[
B = \begin{cases} 
\text{Number Taps (for Single Rate & Decimating filters)} & \\
\text{Number Taps / Interpolating Rate Change (for Interpolating filters)} & 
\end{cases}
\]

\[
\text{Number of Multipliers} = \text{ceiling}\left(\frac{B}{A}\right)
\]

In all cases the floor function always rounds down to the next lower integer value if there is a non-zero remainder. Figure 5 illustrates a FIR implementation that requires four multipliers.

Multiple-channel Filters

The MAC FIR v5.1 core also provides support for processing up to 256 input sample streams using the same implementation. Each input stream is conditioned by the same filter coefficient set and rate-change values.

Total implementation throughput is divided evenly for each channel. For example, consider a single-rate 10-tap filter realized in an FPGA with a master clock frequency of 200 MHz, \(200/10 = 20\) mega-samples/second (MSPS). A two-channel core would evenly share this throughput or 10 MSPS per channel.

It is important to note that as the number of channels is increased, the throughput per channel is commensurately reduced, but that the FPGA logic resource requirements remain approximately the same as that of a single-channel filter. The memory requirements (for example, number of block memories) may increase as a consequence of processing multiple sample streams.

Multi-channel operation is very useful in systems that have two or more streams of data that are being filtered with identical filters. A common use of a two-channel filter is for a complex I/Q data stream. A common requirement is to then filter both the in-phase and quadrature components with identical

![Figure 5: Multiple MAC Engine Implementation](image)
matched filters. In this case, a two-channel filter can be used to process both the I and Q time series, using the same hardware, provided the total required throughput rate of the filter is not exceeded. In this example, approximately half the logic resources would be saved in comparison to an implementation that utilized separate filters to process the I and Q sample streams.

**MAC Engine Implementation**

The MAC FIR engine consists of 5 main modules: input buffer, data storage, coefficient storage, a multiply-accumulate unit, and control logic.

**Input Buffer**

When a multi-channel filter is defined, an input FIFO is automatically added to enable input samples to be supplied on sequential clocks, which is required for cascading multi-channel filters. An input FIFO is also added for polyphase decimation filters so that the filter can process incoming data in the most efficient manner.

**Data Storage**

Under user control, either distributed SelectRAM[1] (distributed memory) or block SelectRAM[1], when available, (block memory) can be selected to store the filters data sample. Block memory is the default storage type. Selection of distributed SelectRAM[1] memory may be advantageous for several cases:

- The device block memory resources are scheduled for an alternative function and are not available for use by the filter
- The number of data values is small and therefore the distributed RAM resources required would be minimal

**Coefficient Storage**

The core customization GUI provides the flexibility to allocate the coefficient storage ROM as either distributed SelectRAM[1] (distributed memory) or block SelectRAM[1] (block memory) if the FPGA target family supports this feature. When block memory is available, it is the default storage type. The user may select distributed SelectRAM[1] (RAM/ROM) memory for the following reasons:

- The device block memory resources are scheduled for an alternative function and are not available for use by the filter
- The number of coefficient values is small and therefore the distributed memory resources required would be minimal

**Multiply-Accumulate Unit**

The multiplier is automatically selected as either a LUT-based or dedicated embedded multiplier when available (for example, for Virtex-II Pro or Spartan-3 devices). With Virtex-4 device, the XtremeDSP slice is used to implement the MAC functionality. When no embedded multiplier is available, a LUT-based implementation is used. Full precision results from the multiplier are summed in the accumulator. The accumulator has sufficient precision to ensure that no overflow can occur during the sum-of-products operation. The final summation is presented on the DOUT output port and can be optionally registered so the output value remains static during successive computation periods.
Control Logic

The control logic manages the flow of data in and out of the input data buffer, the flow of data in and out of the data storage memory, reading of coefficients, and providing the control signals to the MAC core.

Core Pinout

The MAC FIR symbol and ports are shown in Figure 6 and the port definitions are provided in Table 1.

![Core Schematic Symbol]

**Table 1: Core Pinout**

<table>
<thead>
<tr>
<th>Signal</th>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DIN</td>
<td>Input</td>
<td><strong>Data Input</strong>: Input data port</td>
</tr>
<tr>
<td>ND</td>
<td>Input</td>
<td><strong>New Data</strong> (active High): When this signal is asserted, the data sample presented on the DIN port is loaded into the MAC FIR.</td>
</tr>
<tr>
<td>RESET</td>
<td>Input</td>
<td><strong>Reset</strong> (active High): Asynchronous reset signal that initializes all flip-flops to the initial state. All memory pointers are also initialized, but the memory contents are not changed.</td>
</tr>
<tr>
<td>CLK</td>
<td>Input</td>
<td><strong>Clock</strong> (Rising Edge Active): Master processing clock</td>
</tr>
<tr>
<td>COEF_SEL</td>
<td>Input</td>
<td>Identifies which set of coefficients will be used to process the current data sample</td>
</tr>
<tr>
<td>DOUT</td>
<td>Output</td>
<td><strong>Data Output</strong>: Output data port for the MAC FIR. This is always a full-precision output port to avoid overflow.</td>
</tr>
<tr>
<td>RFD</td>
<td>Output</td>
<td><strong>Ready for Data</strong> (active High): Indicates when the MAC FIR can accept new data.</td>
</tr>
<tr>
<td>RDY</td>
<td>Output</td>
<td><strong>Ready</strong> (active High): Indicates that a new filter output sample is available on the DOUT port. This signal is valid for a single clock cycle when the DOUT results first become available.</td>
</tr>
<tr>
<td>SEL_I</td>
<td>Output</td>
<td>Identifies the channel to which new sample data is written (used for multi-channel filters only).</td>
</tr>
<tr>
<td>SEL_O</td>
<td>Output</td>
<td>Decodes the Data Output port and identifies the channel number for the current sample on the filter output port. (Used for multi-channel filters only).</td>
</tr>
</tbody>
</table>
**Interface and Control**

All of the filter classes employ 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 co-ordinate I/O operations. In addition, for multichannel filters, SEL_I and SEL_O are supplied to indicate the active input and output stream respectively. The COEF_SEL port indicates which set of coefficients will be accessed, similar to how the SEL_I port selects which set of data to use.

**Handshake Control Signals**

ND is an Active High input signal which, when asserted, indicates that the core has a valid input sample on the DIN port. ND is internally qualified with the active High output status signal RFD. When both RFD and ND are asserted, the DIN port is sampled on the rising clock edge. The active High output signal RDY indicates that a valid output sum-of-products is available on the DOUT port.

The handshake signals provide a simple and efficient interface to control the flow of sample data and results. Similar to a clock enable signal the ND signal is used to enable the input of samples into the filter. The difference between ND and a clock enable is that the ND signal starts the processing operation that continues to completion. By not asserting the ND signal further, processing is halted. A clock enable provides a start and stop operation of the processing. Given that the internal structure of the MAC is encapsulated and requires multiple clock cycles, a clock enable signal operation can be difficult to use. RFD provides a status signal for upstream data flow control and, when asserted, it indicates that the core can accept more input samples. The RDY signal is often used as a clock enable for the next stage of processing or as the ND signal when filters are cascaded.

**Reset**

RESET is an active High input port which, when asserted, forces the internal control logic to the initialized condition. No internal data is cleared from the memory during the reset process. Following a reset operation, the sum-of-product results will still be dependent on the prior input samples until the filter’s data memory is completely flushed.

**Input/Output Channel Decoding**

When configured for multiple-channel operation, two channel indicator status output ports are provided: SEL_I and SEL_O. The SEL_I port identifies the input channel number; SEL_O provides the mapping between the current sample on the filter output port DOUT and the sample stream number. These signals are often used as select controls for multiplexing input streams or de-multiplexing the time division multiplexed result bus. The SEL_O output is valid when RDY is asserted and changes after the falling edge of RDY.

When configured for multiple-coefficient operation an additional input port COEF_SEL is provided. The COEF_SEL port identifies which set of coefficients will be used to process the current set of data. This port is latched along with the input data DIN when ND is asserted. The value on this port is used to address the portion of the coefficient buffer containing the desired coefficients.
Filter Timing

Figure 7 illustrates the timing for a single-rate, single-channel, N-tap filter. ND is asserted while valid input is available on the DIN port. At the rising edge of the clock, the data is sampled and processing begins. RFD is deasserted to reflect that the MAC FIR core is processing the data and unable to accept further input samples for RFD latency clocks. The RFD latency and latency of the filter is a function of the number of taps, filter type, number of channels, and symmetry. After a number of clock cycles equal to the filter length and the in-built filter latency, RDY is asserted and the valid filter output is presented on the DOUT port. In this example, the DOUT value is held in the optional output register.

Note that in a multi-channel configuration the DIN port has an input buffer, which allows multiple input samples to be burst into the filter.

For multi-channel, polyphase decimation filters, some clock cycles are needed for storing the data; therefore, the maximum theoretical throughput rate cannot be supported. Because of latencies through the filter, five extra clocks are needed to process the data. For example, for a 30-tap, non-symmetric, polyphase decimating filter, 35 clocks are needed to process each set of M input samples, rather than the theoretical minimum of 30 clocks.

Figure 8 illustrates the timing for a dual-channel, single-rate filter. ND is asserted for each input sample, with each channel’s data sequentially presented on the DIN port. SEL_I identifies the input channel and SEL_O identifies the result data on the DOUT port. That is, SEL_I and SEL_O decode the input and output data ports of the filter, respectively.
Core Graphical User Interface

The CORE Generator system’s graphical user interface (GUI) provides an easy process for defining the configuration of the MAC FIR core. The MAC core GUI has a series of five screens. This section of the document describes each of the parameters shown in the GUI. Figures 9 through 13 show the MAC FIR customization panels.

- **Component Name**: Enter a string of characters to be used as the name for the implementation. This parameter defines the HDL component name and is used as the basis for all core-specific files generated by the Core Generator. The name must begin with a legal character, a to z, and can contain numbers and underscore after the first character. Do not use “/”, “*”, “&”, “\”, “(“, or “)”. All characters must be lowercase.

- **Filter Type**: Select one of the filter types: single-rate, polyphase decimator, or polyphase interpolator.

- **Filter Options**: If the user has selected a polyphase decimator or interpolator, this dialog box is made available to specify the rate change value. The interpolation and decimation rate values can vary from 2 through 256.

- **Channels**: Defines the number of individual identical filters that the core implements. Valid range is from 1 through 256.
- **Taps**: The user specifies the number of total taps in the filter. Valid range is from 1 through 1024.

- **Impulse Response**: Choice of three possibilities: Symmetric, Non-Symmetric, and Negative Symmetric. Filter coefficient symmetry can be exploited to increase the throughput capabilities. Select “Symmetric” if the filter has coefficients that are symmetric about the center tap. If the filter’s coefficients are symmetric with the pairings having opposite signs, “Negative Symmetric” should be selected. If either the filter contains no symmetry or the user chooses not to exploit the symmetry, the “Non Symmetric” option should be selected. For polyphase interpolation filters, symmetry cannot be exploited, so “Non Symmetric” is the only option.

- **Coefficient Width**: Number of bits needed to represent the largest magnitude coefficient. This parameter can range from 2 through 32 bits. With Virtex-4 devices the coefficient width is limited to 18 bits for signed coefficients and 17 bits for unsigned coefficients. This is because only single XtremeDSP slices can be used to provide the MAC functionality (there is no cascading of multiple XtremeDSP slices to support coefficient widths up to 32 bits).

- **Number of Coefficient Sets**: Number of different sets of coefficients. The valid range is from 1 through 256.

- **Coefficient Type**: The coefficients in the filter can be either signed two’s complement numbers or unsigned values. Selecting “Signed” interprets the coefficient values as two’s complement values. “Unsigned” selected has the coefficients interpreted as positive magnitude numbers.

- **Coefficient Buffer Type**: Xilinx devices support two types of memory: distributed SelectRAM[1] (distributed memory) and block SelectRAM[1] (block memory). The user can select Block or Distributed memory.
COE File: User coefficients are stored in an external file with a specific file format. This file format allows the user to define the radix of the coefficient values. Clicking the Load Coefficients button brings up a dialog box that allows the user to browse to where the coefficient file is located. Clicking the Show Coefficients button allows the user to display the coefficients that will be loaded into the MAC FIR.

Data Width: Number of bits used for the input sample data. Valid range is from 2 through 32 bits and defines the width of the DIN port. With Virtex-4 devices the data width is limited to 18 bits for signed data and 17 bits for unsigned data. This is because only single XtremeDSP slices can be used to provide the MAC functionality (there is no cascading of multiple XtremeDSP slices to support data widths up to 32 bits). With symmetric filters implemented in Virtex-4 devices, the maximum data width is reduced to 17 bits for signed data and 16 bits for unsigned data.

Data Type: The data in the filter can be either signed two’s complement numbers or unsigned magnitudes. Selecting “Signed” interprets the data values as two’s complement values. “Unsigned” selected has the data interpreted as positive magnitude numbers.

Data Buffer Type: Two types of memory is supported: distributed SelectRAM[1] (distributed memory) and block SelectRAM[1] (block memory). The user can select Block or Distributed RAM for input data storage.
• **Performance Optimization**: The user can select from one of four optimization methods: Auto, Minimal Area/High Speed, Minimum Area, and Maximum Speed.
  
  - **Auto**: Default setting providing an optimal solution. Utilizes the System Clock Rate parameter to add pipeline stages as needed to meet the specified clock frequency.

  **Note**: Virtex-4 devices use Auto for maximum pipelining to maximize performance without requiring CLB resources.

  - **Minimal Area-High-Speed**: Adds pipeline stages in front of the multiplier and within the multiplier itself if the embedded multiplier is being used.
  
  - **Minimum Area**: Removes pipeline stages wherever possible, reducing resources at the expense of lower performance.

  - **Maximum Speed**: Adds pipeline registers at every possible point in the implementation.

  - **System Clock Rate**: This option determines how much pipelining is used through the MAC when “Auto” is chosen for Performance Optimization. This is also used in conjunction with the Input Sample Rate setting to determine how many MAC Engines are needed to implement the filter.

  - **Input Sample Rate**: There are several parameters that interact to determine if a valid set of conditions exists for the filter when determining how many MAC Engines are required to implement the filter. The number of clock cycles between each output is determined by the following equation:

    \[ \text{# Clock Cycles} = \frac{\text{System Clock Rate}}{\text{Input Sample Rate} \times \text{Channels} \times \text{Interpolate Factor}} \geq 1 \quad \text{Equation 2} \]
If this equation is not met then the input sample rate is too high for the filter to process. If the result is an integer value than the output will be periodic. Otherwise, it will be pseudo-periodic (see Design Example 2 for a description of this situation).

The System Clock Rate does not need to be evenly divisible by the Input Sample Rate in order for the MAC FIR to generate a core. However, if the System Clock Rate is not a multiple of the Input Sample Rate than the user must provide a buffer external to the core to handle the different clock rates. This is because the MAC FIR only has a single-clock port.

Assuming the parameters are valid, the number of MAC Engines is then found by dividing the number of multiplies per computation by the result on the left-hand side of the above equation and rounding the result up to the nearest integer value. When multiple MAC Engines are being implemented no optimization is done for symmetry in which the number of multiply operations can be reduced in half.

See Design Example 2 to make sure a valid set of parameters is chosen.

- **Registered Output**: The output of the MAC FIR can be stored so that the DOUT port holds the data until the next RDY pulse indicates that new data is available.
- **Create RPM**: This allows the core to try and optimize the placement of the MAC Engine. This feature is not available for Spartan-3 devices. When enabled, this feature could increase the number of slices required to implement the design. However, this can also increase the maximum clock speed if the multiply-accumulator section of the filter is the bottleneck in the design.

The final GUI screen shown in Figure 13 is a summary of the filter parameters. This screen verifies that the core is configured correctly.

![Figure 13: MAC FIR GUI Screen 5](image-url)
Design Example 1

An example use of a MAC FIR filter would be in a Digital Down Converter (DDC). Figure 14 illustrates a simple block diagram for a DDC configured for GSM base station application. The input sample rate is 53 MSPS. The CIC filter performs decimation by 48, resulting in an input sample rate of 1.08 MSPS into the CFIR filter. CFIR filter is 21-tap, 2 to 1 decimating FIR filter; with a resulting output rate of 0.54 MSPS. The PFIR is a 63-tap FIR, with 2 to 1 decimation.

![Figure 14: Digital Down Converter Block Diagram](image)

The MAC FIR core is an ideal implementation choice for the CFIR and PFIR. The MAC FIR v5.1 core also supports multiple channels, which can be used to implement two or more of the same filters; CFIR and PFIR are ideal candidates. Figure 15 illustrates the GSM DDC with the MAC FIR v5.1 cores utilized for the CFIR and PFIR.

![Figure 15: GSM DDC Utilizing the MAC FIR v5.1](image)

Design Example 2

This example highlights how various parameters interact to determine how many MAC engines are used within the MAC FIR. The user must be careful to insure that the MAC FIR will produce a new output every N clock cycles (where N is an integer) if the user needs a periodic output from the MAC FIR.

For instance, if the user is building an interpolate-by-2 rate change filter with the following parameters:

- System Clock Rate = 76.8 MHz
- Input Sample Rate = 3.84 MHz
- Number of Taps = 47
- Number of Channels = 4
- Impulse Response = Non-symmetric (by default)

By using the formula in Equation 2, the user can determine how often a new output will be generated.

If the result is not an integer value (as in this example) then the MAC FIR will output data in a pseudo-periodic manner. In this example, the MAC FIR will want to generate a new output every 2.5
clock cycles. After an input sample is received, the MAC FIR will generate the first output after \( L \) clock cycles (where \( L \) is the latency through the filter) and will then generate the second output two clock cycles after the first output has been produced. The first output from the next input sample will then be produced three clock cycles after the second output from the first input sample. This pattern will continue so that the MAC FIR will produce 2 outputs every 5 clock cycles.

For the MAC FIR to generate a periodic output under these set of conditions the System Clock Rate must be doubled so that the MAC FIR is able to produce a new output every 5 clock cycles. Under either set of conditions the MAC FIR will produce data at the correct output rate of 7.68 MHz (for each channel).

**XCO File Parameters**

During core generation, a file containing all of the user parameters is created. This file has the same name as the component with an .xco file extension. This file has a dual purpose: as documentation, or to regenerate a new core implementation. The .xco file can be a very convenient method of making small changes to a core’s parameters and generating a new implementation. Shown below is an example .xco file for the MAC FIR v5.1. Table 2 defines the .xco parameters and their valid ranges.

```
SELECT MAC_FIR_Filter virtex2 Xilinx,_Inc. 3.0
CSET component_name = mac_fir_test
CSET coefficient_type = unsigned
CSET create rpm = true
CSET interpolation_factor = 1
CSET filter_type = single_rate
CSET channels = 1
CSET impulse_response = non_symmetric
CSET select_input_port_direction = output
CSET data_buffer_type = block_ram
CSET taps = 11
CSET performance_optimization = auto_performance
CSET coefficient_buffer_type = block_memory
CSET data_width = 15
CSET number_of_coefficient_sets = 1
CSET zero_packing_factor = 1
CSET decimation_factor = 1
CSET system_clock_rate = 300.0
CSET data_type = unsigned
CSET input_sample_rate = 0.1
CSET coefficient_width = 12
CSET coefficient_file = a.coe
CSET registered_output = true
GENERATE
```

The general format for setting parameters is to use a “CSET” at the beginning of the line, followed by the parameter “=” parameter assignment.
### Table 2: Core XCO Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Definition</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>component_name</td>
<td>Module name</td>
<td>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.</td>
</tr>
<tr>
<td>taps</td>
<td>Number of taps in the filter</td>
<td>[1,…,1024]</td>
</tr>
<tr>
<td>filter_type</td>
<td>Specifies the filter's operation</td>
<td>(single_rate</td>
</tr>
<tr>
<td>data_width(^{1})</td>
<td>Input sample width in bits</td>
<td>[2,…, 32]</td>
</tr>
<tr>
<td>data_type</td>
<td>Select either two's complement or magnitude only for the data</td>
<td>{signed</td>
</tr>
<tr>
<td>data_buffer_type</td>
<td>Type of memory used to store data samples</td>
<td>{block_ram</td>
</tr>
<tr>
<td>coefficient_type</td>
<td>Select either two's complement or magnitude only for the coefficients</td>
<td>{signed</td>
</tr>
<tr>
<td>coefficient_file</td>
<td>Name of file that contains the coefficients (COE file)</td>
<td>Any valid file name for the user's operating system consisting of the letters a…z, 0…9 and “.” The file should have a .coe extension.</td>
</tr>
<tr>
<td>coefficient_width</td>
<td>Coefficient sample width in bits</td>
<td>[2,…, 32]</td>
</tr>
<tr>
<td>coefficient_buffer_type</td>
<td>Type of memory used to store coefficients</td>
<td>{block_memory</td>
</tr>
<tr>
<td>interpolation_factor</td>
<td>For interpolation filters, the rate change value</td>
<td>[2,…, 256]</td>
</tr>
<tr>
<td>decimation_factor</td>
<td>For decimation filters, the rate change value</td>
<td>[2,…, 256]</td>
</tr>
<tr>
<td>impulse_response</td>
<td>Indicates if the coefficients are symmetric, non-symmetric, or negative symmetric</td>
<td>{Symmetric</td>
</tr>
<tr>
<td>registered_output</td>
<td>Capture the filter's results in a hold register</td>
<td>{true</td>
</tr>
<tr>
<td>channels</td>
<td>Number of channels the filter will process</td>
<td>[1,…, 256]</td>
</tr>
<tr>
<td>system_clock_rate</td>
<td>The clock that drives the filter</td>
<td>[0.01,…, 300.0]</td>
</tr>
<tr>
<td>input_sample_rate</td>
<td>The rate that data is being input to the filter (MHz)</td>
<td>[0.01,…, 300.0]</td>
</tr>
<tr>
<td>performance_optimization</td>
<td>Controls the amount of pipelining in the MAC</td>
<td>{auto_performance</td>
</tr>
<tr>
<td>create_rpm</td>
<td>Enables placement directives to be used for the MAC</td>
<td>{true</td>
</tr>
</tbody>
</table>
### Coefficient File Format

The filter coefficients are supplied to the core using a coefficient file with a .coe extension. This is an ASCII text file with a single-line header that defines the radix (base-2, base-10, or base-16) of the number representation used for the coefficient data, followed by the coefficient values themselves. This is shown in Figure 16 for an 8-tap filter.

1. Maximum width is reduced for Virtex-4 FPGA devices.

#### Table 2: Core XCO Parameters (Continued)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Definition</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select_input_port_direction</td>
<td>XILINX RESERVED</td>
<td>XILINX RESERVED</td>
</tr>
<tr>
<td>number of coefficient sets</td>
<td>Number of coefficient sets stored in coefficient buffer</td>
<td>[1,...,256]</td>
</tr>
<tr>
<td>zero_packing_factor</td>
<td>XILINX RESERVED</td>
<td>[1,...,256]</td>
</tr>
</tbody>
</table>

```plaintext
radix=10;
coefdata=20, -256, 200, 255, 39, 117, -235, 47;
```

**Figure 16:** Filter Coefficient File Format, 8-tap Filter with Base-10 Coefficient Values

The coefficient values may also be placed on separate lines as shown in Figure 17.

```plaintext
radix=16;
coefdata=20, F01, F9E, 12C, 4D, 57, FB2, C1;
```

**Figure 17:** Filter Coefficient File Format with Coefficients on Separate Lines

Even when the coefficient values are symmetrical, the coefficient file should contain the complete set of coefficient values. The filter coefficient file for a symmetric 8-tap filter is shown in Figure 18.

```plaintext
radix=10;
coefdata=20, -25, 200, 255, 255, 200, -25, 20;
```

**Figure 18:** Filter Coefficient File Format, 8-tap Symmetric Filter with Base-10 Coefficient Values
For multiple coefficient filters, a single COE file is used to specify the coefficient sets. Each coefficient set should be appended to the previous set of coefficients.

For example, if a 2-coefficient set, 10-tap symmetric filter was being designed
and coefficient set #0 was coefdata = -1, -2, -3, 4, 5, 5, 4, -3, -2, -1;
and coefficient set #1 was coefdata = -9, -10, -11, 12, 13, 13, 12, -11, -10, -9;
then the COE file for the entire filter would be: radix = 10;
coefdata = -1, -2, -3, 4, 5, 5, 4, -3, -2, -1, -9, -10, -11, 12, 13, 13, 12, -11, -10, -9;

**Resource Utilization**

Table 3 provides characterization data for several filter implementations in a Virtex-II Pro FPGA. The examples use block SelectRAM[1] for both data and coefficient storage and make use of the embedded multiplier fabric in this device family.

<table>
<thead>
<tr>
<th>Filter Configuration</th>
<th>Filter Length</th>
<th>Channels</th>
<th>Data/Coeff Width</th>
<th>MPY</th>
<th>Block Memories</th>
<th>Slice Count</th>
<th>Max Clock Frequency (MHz)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single-Rate</td>
<td>16</td>
<td>1</td>
<td>16/16</td>
<td>1</td>
<td>1</td>
<td>96 (63)</td>
<td>284 (205)</td>
</tr>
<tr>
<td>Decimate-by-2</td>
<td>16</td>
<td>1</td>
<td>16/16</td>
<td>1</td>
<td>1</td>
<td>117 (84)</td>
<td>285 (204)</td>
</tr>
<tr>
<td>Interpolate-by-2</td>
<td>16</td>
<td>1</td>
<td>16/16</td>
<td>1</td>
<td>1</td>
<td>106 (73)</td>
<td>286 (201)</td>
</tr>
<tr>
<td>Single-Rate</td>
<td>16</td>
<td>4</td>
<td>16/16</td>
<td>1</td>
<td>1</td>
<td>120 (88)</td>
<td>285 (203)</td>
</tr>
<tr>
<td>Decimate-by-2</td>
<td>16</td>
<td>4</td>
<td>16/16</td>
<td>1</td>
<td>2</td>
<td>140 (107)</td>
<td>247 (199)</td>
</tr>
<tr>
<td>Interpolate-Rate</td>
<td>16</td>
<td>4</td>
<td>16/16</td>
<td>1</td>
<td>1</td>
<td>133 (99)</td>
<td>274 (209)</td>
</tr>
<tr>
<td>Single-Rate</td>
<td>64</td>
<td>1</td>
<td>16/16</td>
<td>1</td>
<td>1</td>
<td>115 (73)</td>
<td>285 (203)</td>
</tr>
<tr>
<td>Decimate-by-8</td>
<td>64</td>
<td>1</td>
<td>16/16</td>
<td>1</td>
<td>1</td>
<td>139 (97)</td>
<td>282 (203)</td>
</tr>
<tr>
<td>Interpolate-by-8</td>
<td>64</td>
<td>1</td>
<td>16/16</td>
<td>1</td>
<td>1</td>
<td>127 (85)</td>
<td>286 (211)</td>
</tr>
<tr>
<td>Single-rate</td>
<td>64</td>
<td>4</td>
<td>16/16</td>
<td>1</td>
<td>1</td>
<td>133 (97)</td>
<td>286 (203)</td>
</tr>
<tr>
<td>Decimate-by-8</td>
<td>64</td>
<td>4</td>
<td>16/16</td>
<td>1</td>
<td>3</td>
<td>165 (132)</td>
<td>251 (190)</td>
</tr>
<tr>
<td>Interpolate-by-8</td>
<td>64</td>
<td>4</td>
<td>16/16</td>
<td>1</td>
<td>1</td>
<td>146 (112)</td>
<td>286 (201)</td>
</tr>
</tbody>
</table>

**Note:**
1. Clock frequency performance data is for a Virtex-II Pro XC2VP20-7 FPGA. This data was generated using the Xilinx ISE v6.3i tools, speed file revision ADVANCED 1.69. All filters reflect a non-symmetric impulse response. Slice Count and Clock Frequency entries: the numbers in parenthesis correspond to a non-pipelined filter implementation; other values are for a pipelined architecture.
References


Ordering Information

This core may be downloaded from the Xilinx IP Center for use with the Xilinx CORE Generator system v7.1i and later. The Xilinx CORE Generator system is bundled with all ISE Foundation software 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.

Revision History

The table below shows the revision history of this document.

<table>
<thead>
<tr>
<th>Date</th>
<th>Version</th>
<th>Revision</th>
</tr>
</thead>
<tbody>
<tr>
<td>3/28/03</td>
<td>1.0</td>
<td>Revision history initiated for the document.</td>
</tr>
<tr>
<td>5/21/04</td>
<td>1.1</td>
<td>Document updated to support MAC FIR core v5.0.</td>
</tr>
<tr>
<td>11/11/04</td>
<td>1.2</td>
<td>Document updated to support MAC FIR core v5.1 which reflects reduction in slice count for filters that require more than one multiplier.</td>
</tr>
<tr>
<td>04/28/05</td>
<td>1.3</td>
<td>Updated document to indicate support for Xilinx ISE software v7.1i and Spartan-3E.</td>
</tr>
</tbody>
</table>