Upgrading a FIR Compiler v3.1.x Design to v3.2.x

May 2005, ver. 1.0 Application Note 387

Introduction

This application note is intended for designers who have an FPGA design that uses the Altera® FIR Compiler MegaCore® function version 3.1.0 or earlier and want to upgrade their design to the FIR Compiler version 3.2.0 or later. The changes introduced in the FIR Compiler version 3.2.x affect the following cases:

- Throughput for interpolation filters
- Throughput for decimation filters
- Input data sequence for multichannel decimation filters

Design changes may be necessary in these cases.

This document discusses the following filters:

- Interpolation Filters
- Decimation Filters
- Multichannel Filters

The following discussion gives a brief description of each filter and then compares the differences between the FIR Compiler versions 3.1.x and 3.2.x.

For more information on the Altera FIR Compiler, refer to the *FIR Compiler MegaCore Function User Guide*.

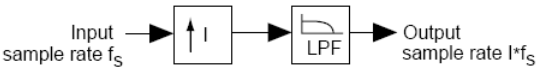
Multirate Filters

Multirate filters are required when changing the sampling rate of a data path in a system. Multirate filters include both interpolation and decimation filters. Interpolation increases the sample rate by inserting zero-valued samples between the original samples, while decimation discards samples to decrease the sample rate. The FIR Compiler automatically creates interpolation and decimation filters using polyphase decomposition. Polyphase filters simplify the overall system design and also reduce the number of computations per cycle required by the hardware.

Interpolation Filters

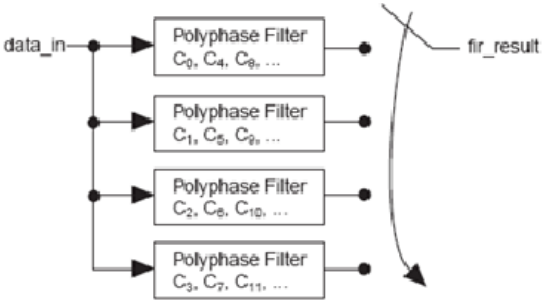
An interpolation filter increases the output sample rate by a factor of *I* through the insertion of *I-1* zeros between input samples, a process known as zero padding. Polyphase decomposition reduces the number of operations per clock cycle by ignoring the zeros that are padded in between the original input samples. Polyphase interpolation filters provide both speed and area optimization because each polyphase filter runs at the input data rate for maximum throughput (Figures 1 and 2).

Figure 1. Polyphase Interpolation Block Diagram



Note to Figure 1
(1) LPF = Low Pass Filter

Figure 2. Polyphase Decomposition for Interpolation Filters



In the FIR Compiler v3.1.0 and earlier, the IP implements interpolation filters using multiple engines in parallel to optimize throughput. In the FIR Compiler v3.2.0 and later, the IP implements interpolation filters using a single engine that is time-shared by the different phases to optimize area. This difference in implementation changes the overall throughput of the filter and the input sample rate. The throughput of the filter is the rate at which the filter generates the output (one output every *K* clock cycles). The input sample rate is the rate at which the filter processes input data samples (the input needs to be held for *L* clock cycles).

The values of *K* and *L* for the throughput and input sample rate of interpolation filters in the FIR Compiler v3.2.x depend on the filter architecture you select. Following are the definitions of *K* and *L* for the different filter architectures:

- **Fully Serial**

  K = N
  L = N * I

  where:

  N = Input bit width
  I = Interpolation factor

- **Multi-bit Serial**

  K = N/M
  L = N * I / M

  where:

  N = Input bit width
  M = Number of serial units
  I = Interpolation factor

  The structure of the Multi-bit Serial architecture requires the input bit width (*N*) to be an integer multiple of the number of serial units (*M*).

- **Fully Parallel**

  K = 1
  L = I

  where:

  I = Interpolation factor

- **Multicycle**

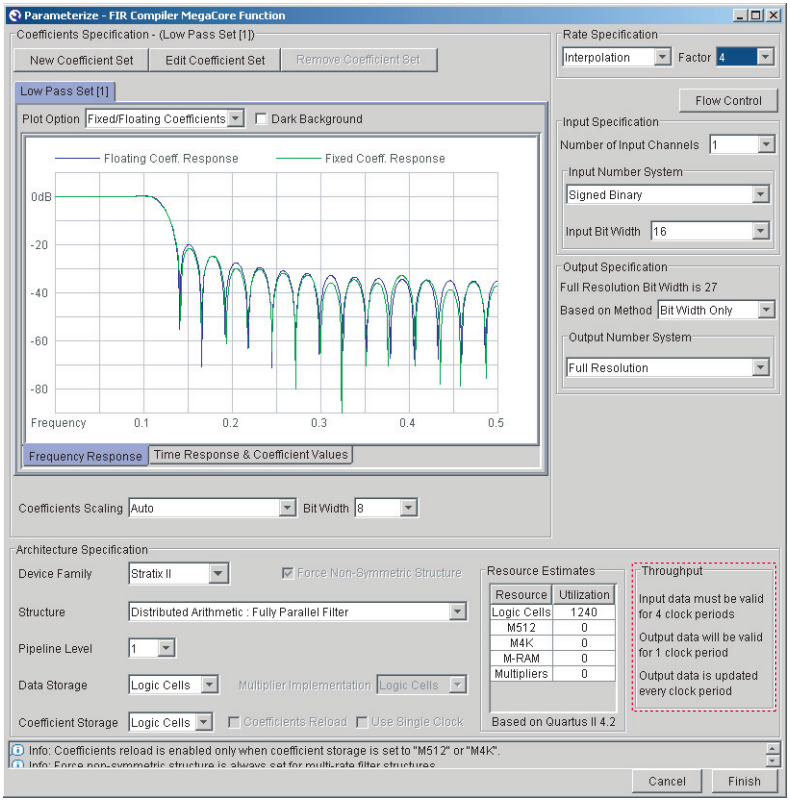
  K = C
  L = C * I

  where:

  C = Clocks per output data
  I = Interpolation factor

For systems that require higher throughput and input data rate, Altera recommends that you use parallel or multicycle variable structures. For the exact throughput and input data rate, refer to the FIR Compiler GUI using IP Toolbench (Figure 3).

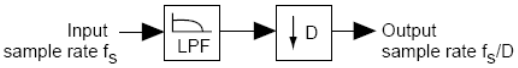
Figure 3. FIR Compiler GUI With Throughput Information for Interpolation Filters



Decimation Filters

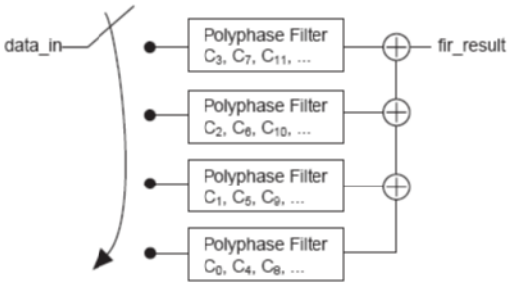
A decimation filter decreases the output sample rate by a factor of *D* by keeping only every *D*-th input sample. Polyphase decomposition reduces the number of computations per cycle by ignoring the input data samples that are discarded during down sampling. Polyphase decimation filters provide speed optimization because each polyphase filter runs at the output data rate (Figures 4 and 5).

Figure 4. Polyphase Decimation Block Diagram



Note to Figure 4:
(1) LPF = Low Pass Filter

Figure 5. Polyphase Decomposition for Decimation Filters



In the FIR Compiler v3.1.0 and earlier, the IP implements decimation filters using multiple engines in parallel to optimize throughput. In the FIR Compiler v3.2.0 and higher, the IP implements decimation filters using a single engine that is time-shared by the different phases to optimize area. This difference in implementation changes the overall throughput of the filter and the input sample rate. The throughput of the filter is the rate at which the filter generates the output (one output every *K* clock cycles). The input sample rate is the rate at which the filter processes input data samples (the input needs to be held for *L* clock cycles).

The values of *K* and *L* for the throughput and input sample rate of decimation filters in FIR Compiler v3.2.x depend on the filter architecture you select. Following are the definitions of *K* and *L* for the different filter architectures:

- **Fully Serial**

  K = N * D
  L = N

  where:

  N = Input bit width
  D = Decimation factor

- **Multi-bit Serial**

  K = N * D/M
  L = N/M

  where:

  N = Input bit width
  M = Number of serial units
  D = Decimation factor

  The structure of the Multi-bit Serial architecture requires the input bit width (*N*) to be an integer multiple of the number of serial units (*M*).

- **Fully Parallel**

  K = D
  L = 1

  where:

  D = Decimation factor

- **Multicycle**

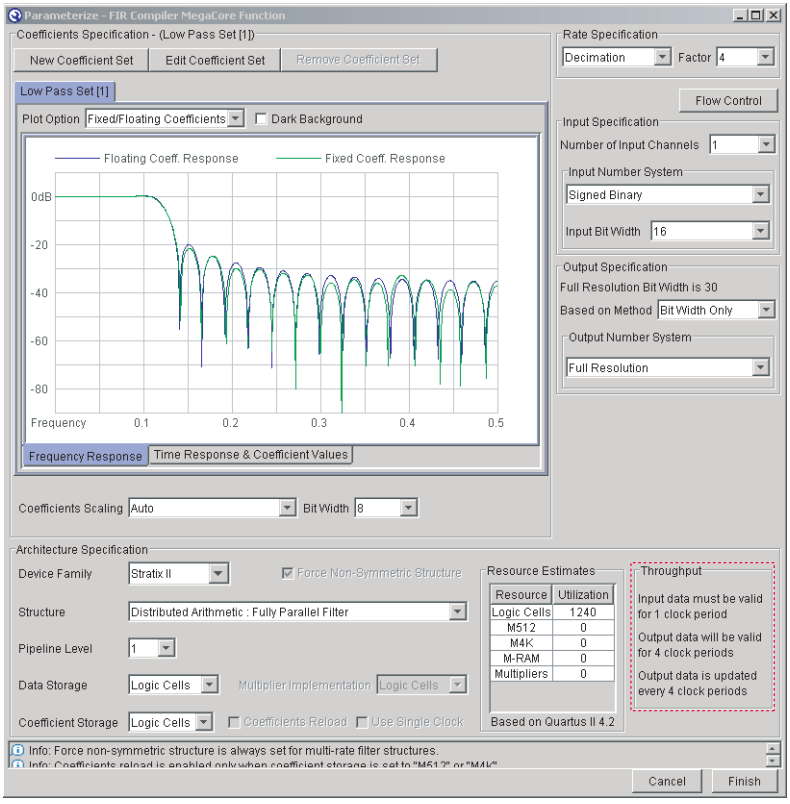
  K = C * D
  L = C

  where:

  C = Clocks per output data
  D = Decimation factor

For systems that require higher throughput and input data rate, Altera recommends that you use parallel or multicycle variable structures. For the exact throughput and input data rate, refer to the FIR Compiler GUI using IP Toolbench (Figure 6).

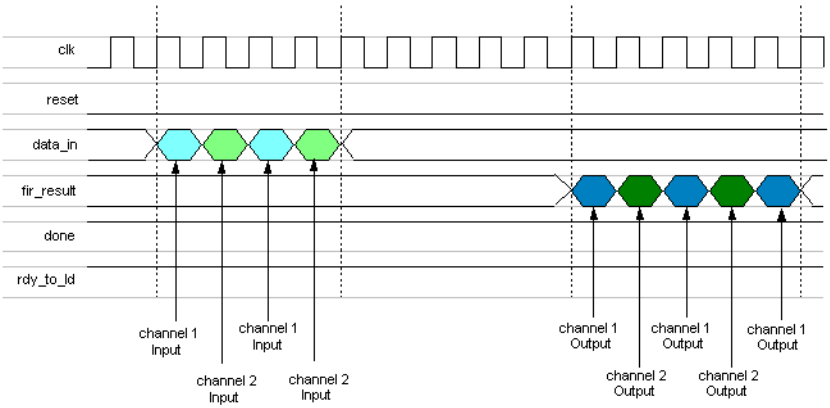
Figure 6. FIR Compiler GUI With Throughput Information for Decimation Filters



Multichannel Filters

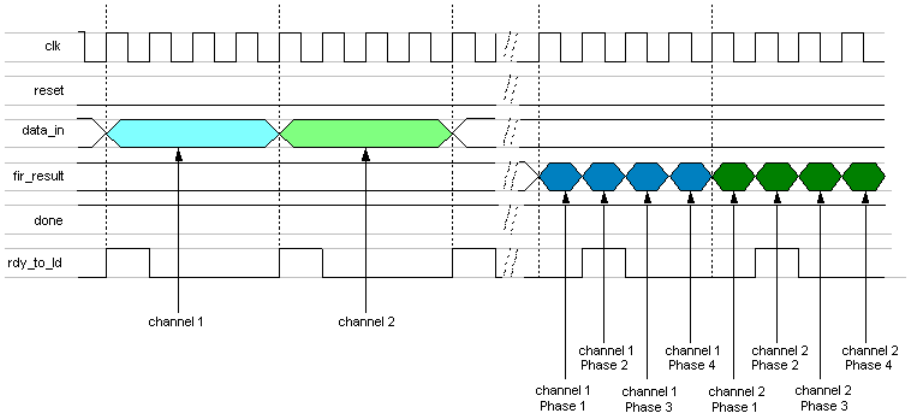
When designing a DSP system that requires multiple data streams to be filtered using the same filter coefficient set, you can time-share the same filter architecture across the different data streams to minimize the number of resources used. In the FIR Compiler, the multichannel filter architecture expects the input data stream to interleave the channels. The output data stream also is interleaved in the same manner. Figure 7 shows an example of a multichannel, single rate filter using the parallel structure.

Figure 7. Multichannel Timing Diagram—Single Rate Filter, Parallel Structure With Two Channels



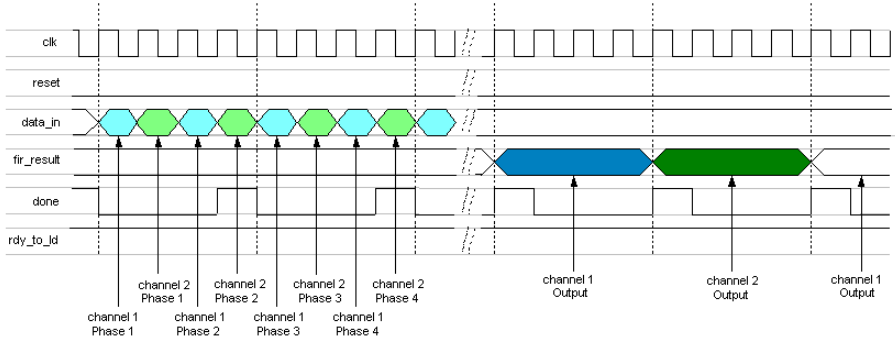
For a multichannel interpolation filter, the input for each channel is held until all phases of that channel are computed. The next channel input is then applied to the interpolation filter. Thus, for each channel in a multichannel scheme, an input is held constant until all phases are computed before the next channel's input can be applied. The expected input data stream for a multichannel interpolation filter is the same in both FIR Compiler v3.1.x and FIR Compiler v3.2.x (Figure 8).

Figure 8. Multichannel Interpolation Filter Timing Diagram—Parallel Structure With Two Channels, Interpolation Factor = 4



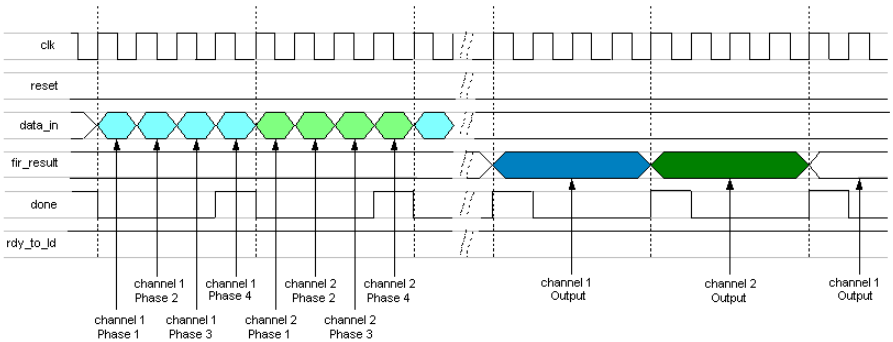
For multichannel decimation filters, the FIR Compiler v3.1.0 and earlier versions expect the input data stream, which feeds into the individual polyphase decomposed filters, to be interleaved with the different channels. The polyphase inputs for each channel are spread evenly across *X* clock cycles. The number of clock cycles, *X*, is determined by the number of channels and the decimation factor. The output of the filter result is interleaved between the channels (Figure 9).

Figure 9. Multichannel Decimation Filter Timing Diagram Using FIR Compiler v3.1.x—Parallel Structure With Two Channels, Decimation Factor = 4



In the FIR Compiler v3.2.x, the output data stream is interleaved differently than in earlier versions. The individual phases for each channel are generated in consecutive order, instead of being spread out across different clock cycles. Inputs from the same channel must be applied until the FIR Compiler computes the results for a single channel. Inputs from the next channel can then be applied. This reduces the overall latency because the input phases required to calculate each output per channel are presented in consecutive order. The input phases for each channel are no longer interleaved and spread out across the separate channels. Figure 10 shows an example of a multichannel decimation filter using the parallel structure.

Figure 10. Multichannel Decimation Filter Timing Diagram Using FIR Compiler v3.2.x—Parallel Structure With Two Channels, Decimation Factor = 4



If your design uses the multichannel decimation filter, and you are upgrading from FIR Compiler v3.1.x to FIR Compiler v3.2.x, you must change the control logic that manages the input data streams from the different channels feeding into the decimation filter.

Conclusion

The implementation of multirate filters in FIR Compiler v3.2.x optimizes for area and affects the throughput and input sample rate of both interpolation and decimation filters. You may have to choose a different filter architecture to retain the same throughput as FIR Compiler v3.1.x or earlier. For further optimization, the FIR Compiler v3.2.x implements the polyphase decomposed filters using a multichannel, multi-coefficient set structure with an appropriate wrapper. This alters the input data sequence for multichannel decimation filters. You must rearrange the polyphase inputs to be sequential for each channel in FIR Compiler v3.2.x, rather than interleaving them between the channels as in FIR Compiler v3.1.x.

101 Innovation Drive
San Jose, CA 95134
(408) 544-7000
www.altera.com
Applications Hotline:
(800) 800-EPLD
Literature Services:
literature@altera.com

Copyright © 2005 Altera Corporation. All rights reserved. Altera, The Programmable Solutions Company, the stylized Altera logo, specific device designations, and all other words and logos that are identified as trademarks and/or service marks are, unless noted otherwise, the trademarks and service marks of Altera Corporation in the U.S. and other countries. All other product or service names are the property of their respective holders. Altera products are protected under numerous U.S. and foreign patents and pending applications, maskwork rights, and copyrights. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera Corporation. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

I.S. EN ISO 9001