MegaCore®

# IIR Compiler

## MegaCore Function User Guide

ALTERA®

101 Innovation Drive
San Jose, CA 95134
(408) 544-7000
http://www.altera.com

| | |
|---|---|
| **Core Version:** | 1.3.2 |
| **Document Version:** | 1.3.2 rev. 1 |
| **Document Date:** | October 2002 |

I.S. EN ISO 9001

UG-IIRCOMPILER-1.4

# About this User Guide

This user guide provides comprehensive information about the Altera® Infinite Impulse Response (IIR) Compiler MegaCore® function.

Table 1 shows the user guide revision history.

Go to the following sources for more information:

- See "Features" on page 9 for a complete list of the core features, including new features in this release.
- Refer to the IIR Compiler readme file for late-breaking information that is not available in this user guide.

*Table 1. User Guide Revision History*

| Date | Description |
|---|---|
| October 2002, v1.3.2 rev.1 | Updated the user guide for version 1.3.2 of the core. Added information on support of Cyclone™ device family, and support for MATLAB 6.5 and Simulink 5.0. |
| July 2002, v1.3.1 rev. 1 | Updated the user guide for version 1.3.1 of the core. Added information on support for the OpenCore® Plus hardware evaluation feature. |
| June 2002, v1.3.0 rev.1 | Updated the user guide for version 1.3.0 of the core. Added information on Stratix and DSP Builder support. Updated the user guide structure, and added information on how to set up licensing. |
| February 12, 2001 | Updated the document for version 1.0.1 of the core, including updating the testing conditions information and adding waveforms to show the latency parameter. |
| January 5, 2001 | First released version of the document. |

## How to Find Information

- The Adobe Acrobat Find feature allows you to search the contents of a PDF file. Click the binoculars toolbar icon to open the Find dialog box.
- Bookmarks serve as an additional table of contents.
- Thumbnail icons, which provide miniature previews of each page, provide a link to the pages.
- Numerous links, shown in green text, allow you to jump to related information.

## How to Contact Altera

For the most up-to-date information about Altera products, go to the Altera world-wide web site at http://www.altera.com.

For technical support on this product, go to http://www.altera.com/mysupport. For additional information about Altera products, consult the sources shown in Table 2.

*Table 2. How to Contact Altera*

| Information Type | USA & Canada | All Other Locations |
|---|---|---|
| Technical support | http://www.altera.com/mysupport/ | http://www.altera.com/mysupport/ |
| | (800) 800-EPLD (3753) (7:00 a.m. to 5:00 p.m. Pacific Time) | (408) 544-7000 *(1)* (7:00 a.m. to 5:00 p.m. Pacific Time) |
| Product literature | http://www.altera.com | http://www.altera.com |
| Altera literature services | lit_req@altera.com *(1)* | lit_req@altera.com *(1)* |
| Non-technical customer service | (800) 767-3753 | (408) 544-7000 (7:30 a.m. to 5:30 p.m. Pacific Time) |
| FTP site | ftp.altera.com | ftp.altera.com |

***Note:***
(1) You can also contact your local Altera sales office or sales representative.

## Typographic Conventions

The *IIR Compiler MegaCore Function User Guide* uses the typographic conventions shown in Table 3.

*Table 3. Conventions*

| Visual Cue | Meaning |
|---|---|
| **Bold Type with Initial Capital Letters** | Command names, dialog box titles, checkbox options, and dialog box options are shown in bold, initial capital letters. Example: **Save As** dialog box. |
| **bold type** | External timing parameters, directory names, project names, disk drive names, filenames, filename extensions, and software utility names are shown in bold type. Examples: $\mathbf{f_{MAX}}$, **\qdesigns** directory, **d:** drive, **chiptrip.gdf** file. |
| *Italic Type with Initial Capital Letters* | Document titles are shown in italic type with initial capital letters. Example: *AN 75: High-Speed Board Design.* |
| *Italic type* | Internal timing parameters and variables are shown in italic type. Examples: $t_{PIA}$, $n + 1$. Variable names are enclosed in angle brackets (< >) and shown in italic type. Example: <*file name*>, <*project name*>**.pof** file. |
| Initial Capital Letters | Keyboard keys and menu names are shown with initial capital letters. Examples: Delete key, the Options menu. |
| “Subheading Title” | References to sections within a document and titles of on-line help topics are shown in quotation marks. Example: “Typographic Conventions.” |
| `Courier type` | Signal and port names are shown in lowercase Courier type. Examples: `data1`, `tdi`, `input.` Active-low signals are denoted by suffix n, e.g., `resetn`.<br><br>Anything that must be typed exactly as it appears is shown in Courier type. For example: `c:\qdesigns\tutorial\chiptrip.gdf`. Also, sections of an actual file, such as a Report File, references to parts of files (e.g., the AHDL keyword `SUBDESIGN`), as well as logic function names (e.g., `TRI`) are shown in Courier. |
| 1., 2., 3., and a., b., c.,... | Numbered steps are used in a list of items when the sequence of the items is important, such as the steps listed in a procedure. |
| ■ | Bullets are used in a list of items when the sequence of the items is not important. |
| ✓ | The checkmark indicates a procedure that consists of one step only. |
| ☞ | The hand points to information that requires special attention. |
| ↵ | The angled arrow indicates you should press the Enter key. |
| 👣 | The feet direct you to more information on a particular topic. |

## Abbreviations & Acronyms

| | |
|---|---|
| AGC | automatic gain control circuits |
| AHDL | Altera hardware description language |
| CIC | cascaded integrated comb |
| DSP | digital signal processing |
| EDA | electronic design automation |
| FIR | finite impulse response |
| IIR | infinite impulse response |
| IP | intellectual property |
| LE | logic element |
| LSB | least significant bit |
| MSB | most significant bit |

# Contents

# About this Core

## Release Information

Table 4 provides information about this release of the IIR Compiler MegaCore function.

*Table 4. IIR Compiler Release Information*

| Item | Description |
|---|---|
| Version | 1.3.2 |
| Release Date | October 20, 2002 |
| Ordering Code | IP-IIR |
| Product ID(s) | 0069 |
| Vendor ID(s) | 6AF8 |

## Introduction

The Altera IIR Compiler MegaCore function is used to implement filters for high data throughput applications. You can use the IIR Compiler wizard interface to implement cascaded and parallel structures and IIR orders from 1 to 15. The wizard includes a built-in zero / pole analyzer and frequency / time response simulator.

## New in Version 1.3.2

- Support for Cyclone™ devices
- Support for MATLAB 6.5
- Support for Simulink 5.0

## Features

- Support for Stratix™ devices
- Has the DSP Builder Ready certification
- Support for the OpenCore® Plus hardware evaluation feature
- System-level programmable logic solution for infinite impulse response (IIR) filters that dramatically shortens design cycles
- Automatic conversion to cascaded IIR structure
- Automatic conversion to parallel IIR structure
- Supports IIR orders from 1 to 15
- Supports data / coefficient bit widths from 1 to 32
- Built-in zero / pole analyzer
- Built-in frequency / time response simulator
- Optimized for Stratix, APEX™, FLEX®, and ACEX® devices
- Includes test vectors: provides a MATLAB / Simulink interface, including bit-accurate models and testbench
- OpenCore® feature allows designers to instantiate and simulate designs in the Quartus® II software prior to licensing

## General Description

Digital filters provide an important function in digital signal processing (DSP) design, and are used in a wide variety of applications such as signal separation, restoration, or shaping. The two classes of digital filters are IIR and finite impulse response (FIR). IIR filters are primarily used for high data throughput applications that require a sharp cut-off characteristic. IIR filters require less hardware than FIR filters and have a faster response. The IIR structure is well suited to automatic gain control circuits (AGC), Goertzel algorithm implementation, digital direct synthesis, or cascaded integrated comb (CIC) filters. Due to poles in their transfer function, IIR filters are known as feedback systems. The recursive nature of the IIR filter introduces additional design steps such as phase distortion analysis and finite word-length effects analysis. The IIR Compiler MegaCore® function speeds up IIR design cycles by combining built-in system level analysis tools with a parameterizable IIR MegaCore function.

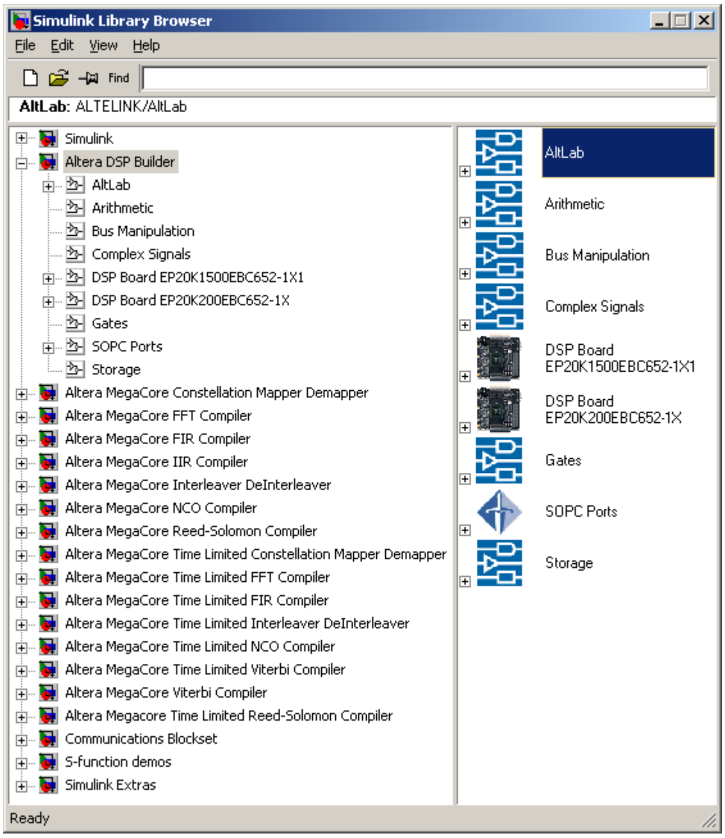
Altera has extensively tested the IIR Compiler function. The testing methodology consists of comparing the timing and functional simulation results of the design created by the IIR Compiler with the IIR Compiler MATLAB model and with the built-in C++ model. The parameters tested are the number of taps, structure, input bit width, coefficient bit width, rounding, and saturation.

### DSP Builder Support

DSP system design in Altera programmable logic devices requires both high-level algorithms and HDL development tools. The Altera DSP Builder, which you can purchase as a separate product, integrates the algorithm development, simulation, and verification capabilities of The MathWorks MATLAB and Simulink system-level design tools with VHDL synthesis and simulation of Altera development tools.

DSP Builder allows system, algorithm, and hardware engineers to share a common development platform. The DSP Builder shortens DSP design cycles by helping you create the hardware representation of a DSP design in an algorithm-friendly development environment. You can combine existing MATLAB functions and Simulink blocks with Altera DSP Builder blocks to link system-level design and implementation with DSP algorithm development. The DSP Builder consists of libraries of blocks as shown in Figure 1.

*Figure 1. DSP Builder Blocks in Simulink Library Browser*



Simulink Library Browser

File Edit View Help

Find

AltLab: ALTELINK/AltLab

- Simulink
- Altera DSP Builder
  - AltLab
  - Arithmetic
  - Bus Manipulation
  - Complex Signals
  - DSP Board EP20K1500EBC652-1X1
  - DSP Board EP20K200EBC652-1X
  - Gates
  - SOPC Ports
  - Storage
- Altera MegaCore Constellation Mapper Demapper
- Altera MegaCore FFT Compiler
- Altera MegaCore FIR Compiler
- Altera MegaCore IIR Compiler
- Altera MegaCore Interleaver DeInterleaver
- Altera MegaCore NCO Compiler
- Altera MegaCore Reed-Solomon Compiler
- Altera MegaCore Time Limited Constellation Mapper Demapper
- Altera MegaCore Time Limited FFT Compiler
- Altera MegaCore Time Limited FIR Compiler
- Altera MegaCore Time Limited Interleaver DeInterleaver
- Altera MegaCore Time Limited NCO Compiler
- Altera MegaCore Time Limited Viterbi Compiler
- Altera MegaCore Viterbi Compiler
- Altera Megacore Time Limited Reed-Solomon Compiler
- Communications Blockset
- S-function demos
- Simulink Extras

AltLab
Arithmetic
Bus Manipulation
Complex Signals
DSP Board EP20K1500EBC652-1X1
DSP Board EP20K200EBC652-1X
Gates
SOPC Ports
Storage

Ready

DSP Builder version 2.0.0 and higher provides modular support for Altera DSP cores, including IIR Compiler. The MATLAB software automatically detects cores that support DSP Builder and the cores appear in the Simulink Library Browser.

## *Version Support for DSP Builder, MATLAB, and Simulink*

Version 1.3.2 of the IIR Compiler MegaCore supports MATLAB versions 6.1 and 6.5, Simulink versions 4.1 and 5.0, and DSP Builder 2.0.0.

For more information on using DSP Builder with IIR Compiler, see "DSP Builder Feature & Simulation Support" on page 44.

## Performance

Table 5 shows typical expected performance for the IIR Compiler for Stratix and APEX II devices. Results were generated using the Quartus II software version 2.1.

*Table 5. IIR Compiler Performance*

| Device | Filter | LEs Used | DSP Blocks | Data Rate (MSPS) |
|---|---|---|---|---|
| Stratix | IIR Order 2 (Butterworth, 16-bit coefficients, 8-bit data) | 44 | 2 | 90 |
| APEX II | | 620 | 0 | 81 |
| Stratix | IIR Order 4 (Chebychev bandpass, 18-bit coefficients, 12-bit data) Direct Form II | 124 | 3 | 80 |
| APEX II | | 1,129 | 0 | 60 |

# Getting Started

## Software Requirements

The instructions in this section require the following software:

- A PC running the Windows 98/NT/2000 operating system
- Quartus II version 2.0 or higher
- DSP Builder version 2.0 or higher (optional)

## Design Flow

Once you have purchased a license for IIR Compiler, the design flow involves the following steps:

☞ If you have not purchased a license, you can test-drive the core for free using the OpenCore feature.

1. Download and install the IIR Compiler function.
2. Set up licensing. This step is not required if you are test-driving the core using the OpenCore feature.
3. Create a custom variation of the IIR Compiler using the core's wizard.
4. Simulate the filter using the wizard-generated MATLAB models.
5. Implement the rest of your system using the Altera Hardware Description Language (AHDL), VHDL, Verilog HDL, or schematic entry.
6. Use the IIR Compiler wizard-generated simulation models to confirm your custom core's operation.
7. Compile your design and perform place-and-route.
8. Perform system verification.
9. Configure or program Altera devices with the design.

## Download & Install the Core

Before you can start using Altera MegaCore functions, you must obtain the MegaCore files and install them on your PC. The following instructions describe this process.

### Downloading the IIR Compiler MegaCore Function

If you have Internet access, you can download MegaCore functions from Altera's web site at http://www.altera.com. Follow the instructions below to obtain the IIR Compiler via the Internet. If you do not have Internet access, you can obtain the IIR Compiler from your local Altera representative.

1. Point your web browser to http://www.altera.com/ipmegastore.

2. Choose **Megafunctions** from the **Product Type** drop-down list box.

3. Choose **Signal Processing (DSP)** from the **Technology** drop-down list box.

4. Type `IIR Compiler` in the **Keyword Search** box.

5. Click **Go**.

6. Click the link for the Altera IIR Compiler MegaCore function in the search results table. The product description web page displays.

7. Click the Free Test Drive graphic on the top right of the product description web page.

8. Fill out the registration form, read the license agreement, and click the **I Agree** button at the bottom of the page.

9. Follow the instructions on the IIR Compiler download and installation page to download the function and save it to your hard disk.

### Installing the IIR Compiler Files

To install the IIR Compiler, perform the following steps:

1. Choose **Run** (Start menu).

2. Type <*path name*>`\`<*filename*>`.exe`, where <*path name*> is the location of the downloaded MegaCore function and <*filename*> is the filename of the function.

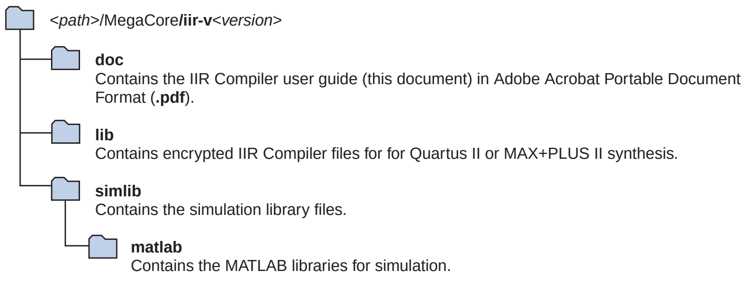
3. Click **OK**. The **IIR Compiler Installation** dialog box appears. Follow the on-screen instructions to finish installation.

4. After you have finished installing the MegaCore files, you may have to specify the core's library directory (typically <*path*>**\iir-v1.3.2\lib**) as a user library in the Quartus II software to access the core in the MegaWizard Plug-In Manager. Search for "User Libraries" in Quartus II Help for instructions on how to add these libraries.

## IIR Compiler Directory Structure

Figure 2 shows the directory structure for the IIR Compiler.

**Figure 2. IIR Compiler Directory Structure**


<path>/MegaCore/iir-v<version>
doc
Contains the IIR Compiler user guide (this document) in Adobe Acrobat Portable Document Format (.pdf).
lib
Contains encrypted IIR Compiler files for for Quartus II or MAX+PLUS II synthesis.
simlib
Contains the simulation library files.
matlab
Contains the MATLAB libraries for simulation.

## Set Up Licensing

You can use Altera's OpenCore evaluation software to compile and simulate the IIR Compiler MegaCore function in the Quartus II software, allowing you to evaluate it before purchasing a license. However, you must obtain a license from Altera before you can generate programming files or EDIF, VHDL, or Verilog HDL gate-level netlist files for simulation in third-party EDA tools.

After you purchase a license for IIR Compiler, you can request a license file from the Altera web site at http://www.altera.com/licensing and install it on your PC. When you request a license file, Altera e-mails you a **license.dat** file. If you do not have Internet access, contact your local Altera representative.

To install your license, you can either append the license to your **license.dat** file or you can specify the core's **license.dat** file in the Quartus II software.

Before you set up licensing for the IIR Compiler, you must already have the Quartus II software installed on your PC, with licensing set up.

## Append the License to Your license.dat File

To append the license, perform the following steps:

1. Close the following software if it is running on your PC:

- Quartus II
- MAX+PLUS II
- LeonardoSpectrum
- Synplify
- ModelSim

2. Open the IIR Compiler license file in a text editor. The file should contain one FEATURE line, spanning 2 lines.

3. Open your Quartus II **license.dat** file in a text editor.

4. Copy the FEATURE line from the IIR Compiler license file and paste it into a new line in the Quartus II license file.

   Do not delete any FEATURE lines from the Quartus II license file.

5. Save the Quartus II license file as a text file.

   When using editors such as Microsoft Word or Notepad, ensure that the file does not have extra extensions appended to it after you save (e.g., **license.dat.txt** or **license.dat.doc**). Verify the filename at a command prompt.

## Specify the Core's License File in the Quartus II Software

To specify the core's license file, perform the following steps:

1. Create a text file with the FEATURE line and save it to your hard disk.

   Altera recommends that you give the file a unique name, e.g., <*core name*>**_license.dat**.

2. Run the Quartus II software.

3. Choose **License Setup** (Tools menu). The **Options** dialog box opens to the **License Setup** page.

4. In the **License file** box, add a semicolon to the end of the existing license path and filename.

5. Type the path and filename of the core license file after the semicolon.

   ☞ Do not include any spaces either around the semicolon or in the path/filename.

6. Click **OK** to save your changes.

## IIR Compiler Walkthrough

This walkthrough explains how to create IIR filters using the Altera IIR Compiler wizard and the Quartus II software. As you go through the wizard, each page is described in detail. The wizard creates an optimized netlist of the IIR filter as well as a bit-accurate MATLAB model. When you are finished generating an IIR filter, you can incorporate it into your overall project.

You can use Altera's OpenCore evaluation feature to compile and simulate the MegaCore functions, allowing you to evaluate the IIR Compiler before deciding to purchase a license. However, you must purchase a license before you can generate programming files or EDIF, VHDL, or Verilog HDL gate-level netlist files for simulation in third-party EDA tools.

This walkthrough consists of the following steps:

- "Create a New Quartus II Project" on page 18
- "Launch the MegaWizard Plug-In Manager" on page 19
- "Specify the IIR Order & Structure" on page 20
- "Specify the Coefficients" on page 21
- "Scale the Coefficients" on page 26
- "Simulate the Hardware Architecture" on page 30
- "Specify the Simulation Output Files" on page 36

## Create a New Quartus II Project

Before you begin, you must create a new Quartus II project. With the New Project wizard, you specify the working directory for the project, assign the project name, and designate the name of the top-level design entity. You will also specify the IIR Compiler user library. To create a new project, perform the following steps:

1. Choose **Altera > Quartus II** <*version*> (Windows Start menu) to run the Quartus II software. You can also use the Quartus II Web Edition software.
2. Choose **New Project Wizard** (File menu).
3. Click **Next** in the introduction (the introduction will not display if you turned it off previously).
4. Specify the working directory for your project.
5. Specify the name of the project.
6. Click **Next**.
7. Click **User Library Pathnames**.
8. Type <*path*>`\iir-v1.3.2\lib\` into the **Library name** box, where <*path*> is the directory in which you installed the IIR Compiler. The default installation directory is **c:\MegaCore**.
9. Click **Add**.
10. Click **OK**.
11. Click **Next**.
12. Click **Finish**.

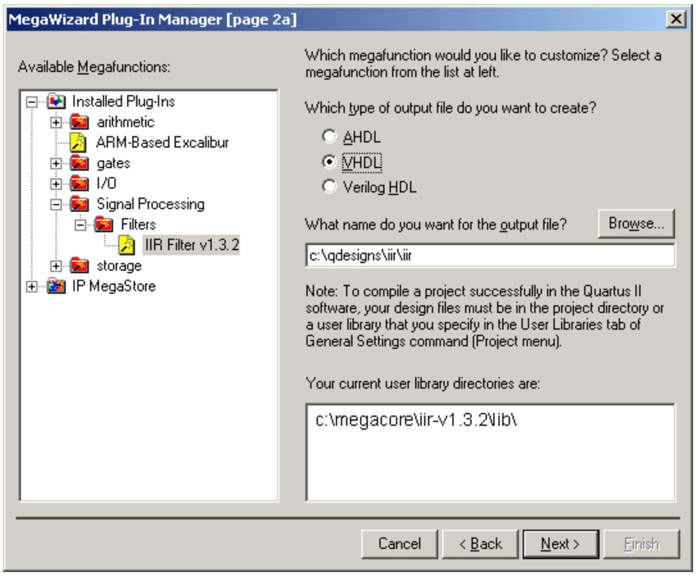
You have finished creating your new Quartus II project.

## Launch the MegaWizard Plug-In Manager

The MegaWizard Plug-In Manager allows you to run a wizard that helps you easily specify options for the IIR Compiler. Perform the following steps to launch the wizard and begin generating a filter.

1. Choose **MegaWizard Plug-In Manager** (Tools menu).
2. Select **Create a new custom megafunction variation** (default).
3. Click **Next**.
4. Expand the **Signal Processing** folder under **Installed Plug-Ins** by clicking the + next to the name.
5. Expand the **Filters** folder under **Signal Processing**.
6. Click **Interleaver v**<*version*>.
7. Choose the output file type for your design; the wizard supports AHDL, VHDL, and Verilog HDL.
8. Type the name of the output file. This walkthrough uses **iir_test**. Figure 3 on page 20 shows the wizard after you have made these settings.
9. Click **Next**.

*Figure 3. Choose IIR Compiler in the MegaWizard Plug-In Manager*



Which megafunction would you like to customize? Select a megafunction from the list at left.

Which type of output file do you want to create?

AHDL [✓ VHDL] Verilog HDL

What name do you want for the output file? c:\qdesigns\iir\iir

Note: To compile a project successfully in the Quartus II software, your design files must be in the project directory or a user library that you specify in the User Libraries tab of General Settings command (Project menu).

Your current user library directories are: c:\megacore\iir-v1.3.2\lib\

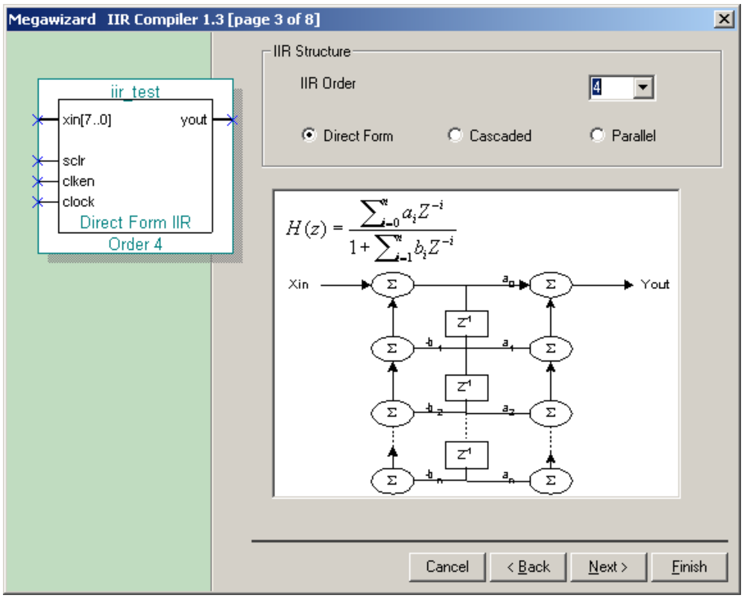
You are now ready to set the options for your custom IIR filter.

## Specify the IIR Order & Structure

Specify the IIR order and structure (direct form II, parallel, or cascaded). Parallel and cascaded architectures are available for IIR orders greater than 2. See Figure 4.

To switch architectures (direct form, cascaded, or parallel) after entering coefficients or zero-pole values for a given filter, you can return to the first of the wizard page and select another architecture. The IIR Compiler automatically computes the coefficient values of the new architecture.

*Figure 4. Specify the Filter Structure*



IIR Order: 4

[✓ Direct Form] Cascaded Parallel

$$H(z) = \frac{\sum_{i=0}^{n} a_i Z^{-i}}{1 + \sum_{i=1}^{n} b_i Z^{-i}}$$

## Specify the Coefficients

When you create a new variation, the wizard loads default coefficient values for a default filter type. To build a different type of filter you must enter the coefficients and zero-pole values. The wizard provides two ways to set the IIR filter characteristics: setting the coefficient values of the feed-forward and feedback IIR path or setting the zero-pole values of the transfer function.

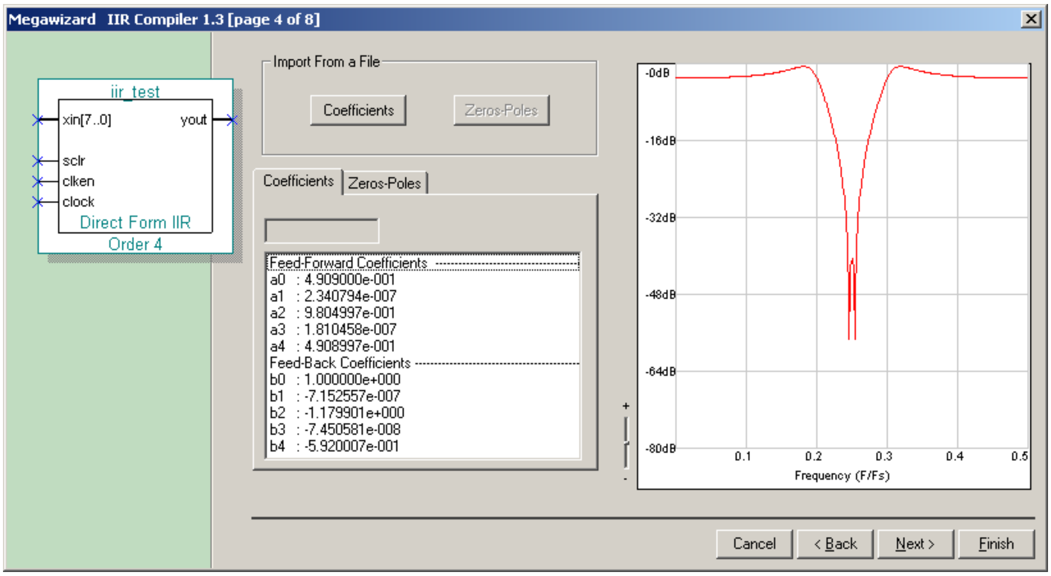
### *Enter Coefficient Values*

You can enter the coefficient values of the feed-forward IIR path ($a_i$) and the coefficient values of the feedback path ($b_i$). You can import the coefficients from a file or enter them manually. The coefficient values can be floating-point numbers or integers.

Select the Zeros-Poles tab to generate the zero-pole values of the coefficients. The IIR Compiler calculates the roots of the numerator and denominator polynomial and the transfer function.

The wizard displays those values in the Z plane, providing information on the IIR stability for floating-point coefficients. See Figure 5.

*Figure 5. Enter the Coefficients*



```
Feed-Forward Coefficients
a0  : 4.909000e-001
a1  : 2.340794e-007
a2  : 9.804997e-001
a3  : 1.810458e-007
a4  : 4.908997e-001
Feed-Back Coefficients
b0  : 1.000000e+000
b1  : -7.152557e-007
b2  : -1.179901e+000
b3  : -7.450581e-008
b4  : -5.920007e-001
```

The frequency diagram is described below:

- The Y axis scale represents the dB attenuation of the filter based on the coefficients. A slider bar lets you to zoom the Y axis in and out.
- The X axis is an absolute frequency, and is the ratio of the filter frequency characteristic over the sample frequency.
- The diagram is updated when you change the coefficients.
- To enter or modify the coefficients manually:
- Select the coefficient.
- Enter the new value in the text box.
- Click the **New** button.
- To enter a coefficient set from a file, click on the **Coefficient** button and load the ASCII coefficient file that you have created in a third party system tool.

For direct form II filters, the expected format of the coefficient file is:

```
A0
A1
A2
..
Ai
..
An
B0
B1
B2
..
Bi
..
Bn
```

Where *Ai* are the floating-point coefficient values of the polynomial of the numerator of the transfer function and *Bi* are the floating-point coefficient values of the polynomial of the denominator of the transfer function. See Equation 1.

***Equation 1***

$$H(z) = \frac{\sum_{i=0}^{n} a_i Z^{-i}}{1 + \sum_{i=1}^{n} b_i Z^{-i}}$$

For cascaded and parallel architectures, the expected format of the coefficient file is:

```
A00 A01 A02 B00 B01 B02
...
Ai0 Ai1 Ai2 Bi0 Bi1 Bi2
...
An0 An1 An2 Bn0 Bn1 Bn2
K
```

Where *Aij* are the floating-point coefficient values of the feed-forward path of the biquad *i*, *Bij* are the floating-point coefficient values of the feedback data path of the biquad *i*, and K is the gain factor of the cascaded or parallel structure.

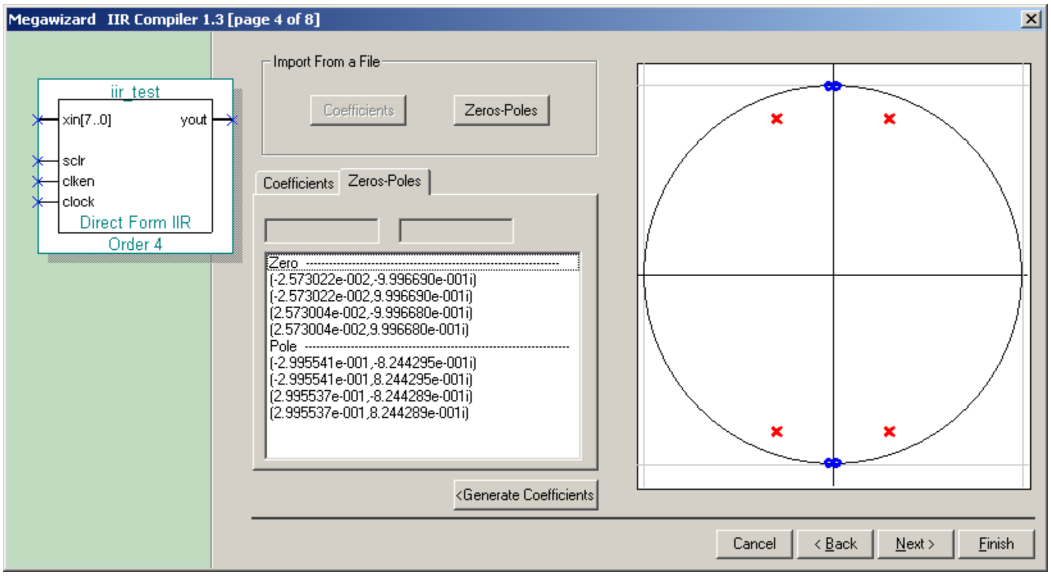
The values displayed in the wizard are the coefficient values of the hardware implementation (see Equation 2). The coefficients of the feedback path are equal to the inverse of the coefficients of the polynomial of the coefficient of the transfer function.

**Equation 2**

$$Y_n = \sum_{i=0}^{n} a_i X_{(n-i)} - \sum_{i=1}^{n} b_i Y_{(n-i)}$$

### Enter Zero-Pole Values

Enter the zero-pole values of the transfer function. The zero-pole values can be imported from an ASCII file or can be entered manually as shown in Figure 6.



```
Zero
(-2.573022e-002,-9.996690e-001i)
(-2.573022e-002,9.996690e-001i)
(2.573004e-002,-9.996680e-001i)
(2.573004e-002,9.996680e-001i)
Pole
(-2.995541e-001,-8.244295e-001i)
(-2.995541e-001,8.244295e-001i)
(2.995537e-001,-8.244289e-001i)
(2.995537e-001,8.244289e-001i)
```

**Figure 6. Enter the Zero-Poles**

The diagram is described below:

- The X axis represents the real part of the zero and the pole; the Y axis represents the imaginary part of the zero and the pole.
- The blue circle are the zeros; the red crosses are the poles.

- The Z plane representation is scaled automatically based on the zero-pole values.
- To enter or modify the zeros-pole values manually:
- Select the zero-pole value in the list.
- Enter the new value in the text box.
- Click **New**.
- To enter a zero-pole set from a file, click the **Zero-Poles** button and load the ASCI file that you have created from a third party system tool.

The expected format of the file is:

```
Real(Z0) Imag(Z0)
Real(Zi) Imag(Zi)
....
Real(Zn) Imag(Zn)
Real(P0) Imag(P0)
Real(Pi) Imag(Pi)
...
Real(Pn) Imag(Pn)
```

Where Real(*c*) is the real part of the complex number *c* and Imag(*c*) is the imaginary part of the complex number *c*.

### MATLAB Script Example

Altera provides an example MATLAB M File (**.m**) with the IIR Compiler. You can run the M File, **dformband.m**, at the MATLAB prompt (see Figure 7). This script generates a low-pass filter and yields three ASCII files:

- **Coef.txt**—Coefficients for the direct form II implementation
- **CoefSos.txt**—Coefficients for the cascaded representation
- **ZerosPoles.txt**—Zero-pole values

The IIR Compiler outputs the coefficient values in two ASCII text files, <*design name*>**_FloatCoef.txt** (the floating point values) and <*design name*>**_FixedCoef.txt** (fixed point values).

*Figure 7. dformband.m MATLAB Script*

```
FilterOrder = 4;
[FeedForward,FeedBack] = ellip(2,3,50,[250/500,300/500],'bandpass');
figure(1);
freqz(FeedForward,FeedBack);
figure(2);
zplane(FeedForward,FeedBack);
fid = fopen('coef.txt','w');
for i=1:FilterOrder+1
   fprintf(fid,'%e\n',FeedForward(i));
end
for i=1:FilterOrder+1
   fprintf(fid,'%e\n',FeedBack(i));
end
fclose(fid);

[z,p,k] = tf2zp(FeedForward,FeedBack);

fid = fopen('ZeroPole.txt','w');
for i=1:FilterOrder
   fprintf(fid,'%e %e\n',real(z(i)),imag(z(i)));
end
for i=1:FilterOrder
   fprintf(fid,'%e %e\n',real(p(i)),imag(p(i)));
end
fclose(fid);
```

## Scale the Coefficients

Coefficient values are often represented as floating-point numbers. To convert these numbers to a fixed-point system, the coefficients must be multiplied by a scaling factor and rounded. The IIR Compiler provides the following scaling options:

- *Scale to a specified number of precision bits*—Because the coefficients are represented by a certain number of bits, it is possible to apply whatever gain factor is required such that the maximum coefficient value equals the maximum possible value for a given number of bits. This approach produces coefficient values with a maximum signal-to-noise ratio.

- *Limit scaling factors to powers of 2*—With this approach, the IIR Compiler chooses the largest power of two scaling factor that can represent the largest number within a particular number of bits of resolution. Multiplying all of the coefficients by a particular gain factor is the same as adding a gain factor before the IIR filter. In this case, applying a power of two scaling factor makes it relatively easy to remove the gain factor by shifting a binary decimal point.

- *Scale manually*—The IIR Compiler lets you manually scale the coefficient values by a specified gain factor.
- *No scaling*—The IIR Compiler can read in pre-scaled integer values for the coefficients and not apply scaling factors.

The new set of scaled fixed point coefficient defines a new transfer function $H_{fixed}(z)$ as shown in Equation 3.

***Equation 3***

$$H_{fixed}(z) = \frac{\sum_{i=0}^{n} a_{ifixed} Z^{-i}}{1 + \sum_{i=1}^{n} b_{ifixed} Z^{-i}}$$

The MegaWizard Plug In Manager plots $H_{fixed}(z)$ in the Z plane. If a pole is outside the unit circle, the filter will be unstable. The wizard graphically plots the poles and you can adjust the scaling if any poles appear outside the unit circle.

The IIR latency is the number of clock cycles that the IIR filter requires to compute `yout`. See Figure 8.

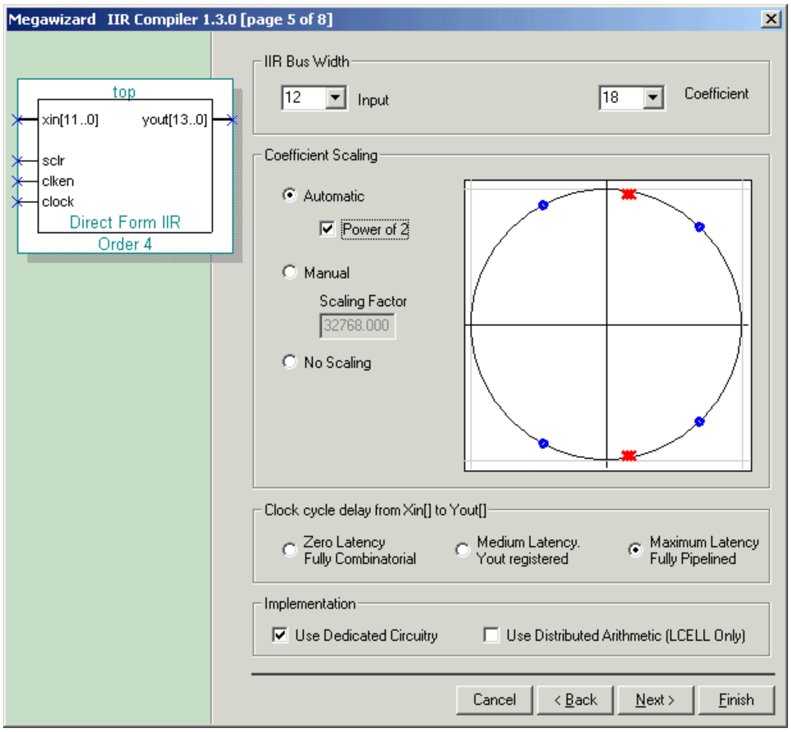
- If you choose zero latency, `yout` is combined and is calculated at the same time that `xin` appears. This mode is useful for building feedback control systems such as automatic gain control circuitry (AGC) or digital clock and data recovery.
- If you choose the medium latency option, `yout` is registered and is calculated one clock cycle after `xin` appears.
- If you choose the maximum latency option, the feed-forward path is fully pipelined; the delay is based on the number of taps. The timing critical path is the feedback path only.

You can also specify two implementation settings:

- *Use Dedicated Circuitry*—Use this option when targeting Stratix devices. If this option is turned on, the IIR transfer function uses Stratix DSP blocks.

- *Use Distributed Arithmetic*—Use this option when targeting any Altera device family. When this option is turned on, the wizard implements the IIR transfer function using LEs, based on distributed arithmetic mapping. When this option is turned off, the wizard implemnts the IIR transfer function using LEs, based on the `LPM_MULT` and `LPM_ADD` functions.

**Figure 8. Scale the Coefficients**



IIR Bus Width: Input 12; Coefficient 18

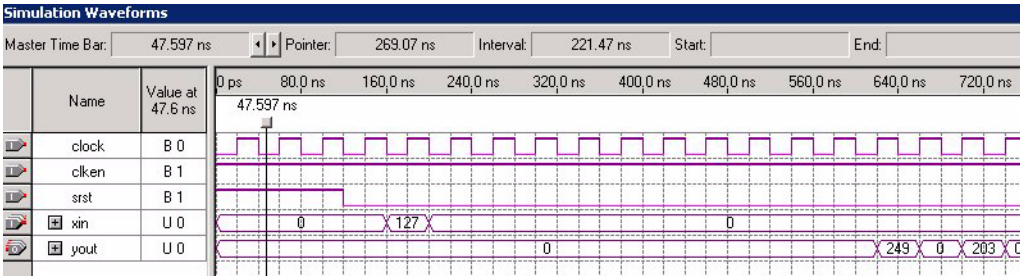
Coefficient Scaling: [✓ Automatic] ☑ Power of 2; Manual, Scaling Factor 32768.000; No Scaling

Clock cycle delay from Xin[] to Yout[]: Zero Latency Fully Combinatorial; Medium Latency Yout registered; [✓ Maximum Latency Fully Pipelined]

Implementation: ☑ Use Dedicated Circuitry; ☐ Use Distributed Arithmetic (LCELL Only)

Figure 9 shows the simulation waveform for full latency.

*Figure 9. Full Latency*

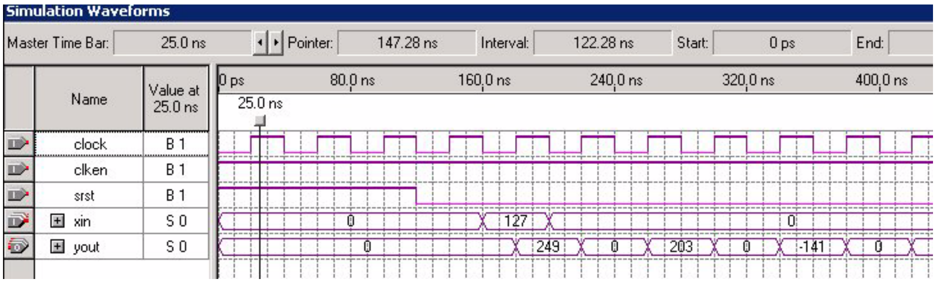

Simulation Waveforms

Master Time Bar: 47.597 ns Pointer: 269.07 ns Interval: 221.47 ns Start: End:

| Name | Value at 47.6 ns |
|---|---|
| clock | B 0 |
| clken | B 1 |
| srst | B 1 |
| xin | U 0 |
| yout | U 0 |

Figure 10 shows the simulation waveform for medium latency.

*Figure 10. Medium Latency*

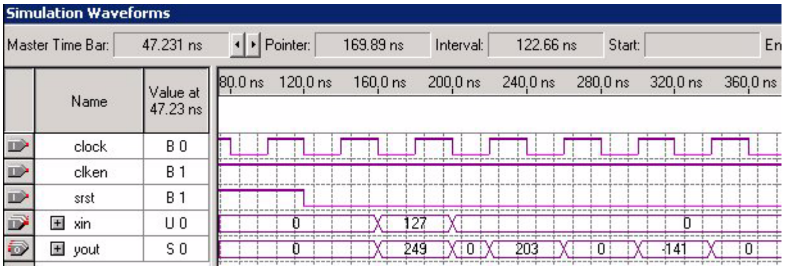

Simulation Waveforms

Master Time Bar: 25.0 ns Pointer: 147.28 ns Interval: 122.28 ns Start: 0 ps End:

| Name | Value at 25.0 ns |
|---|---|
| clock | B 1 |
| clken | B 1 |
| srst | B 1 |
| xin | S 0 |
| yout | S 0 |

Figure 11 shows the simulation waveform for no latency.

*Figure 11. No Latency*


Simulation Waveforms

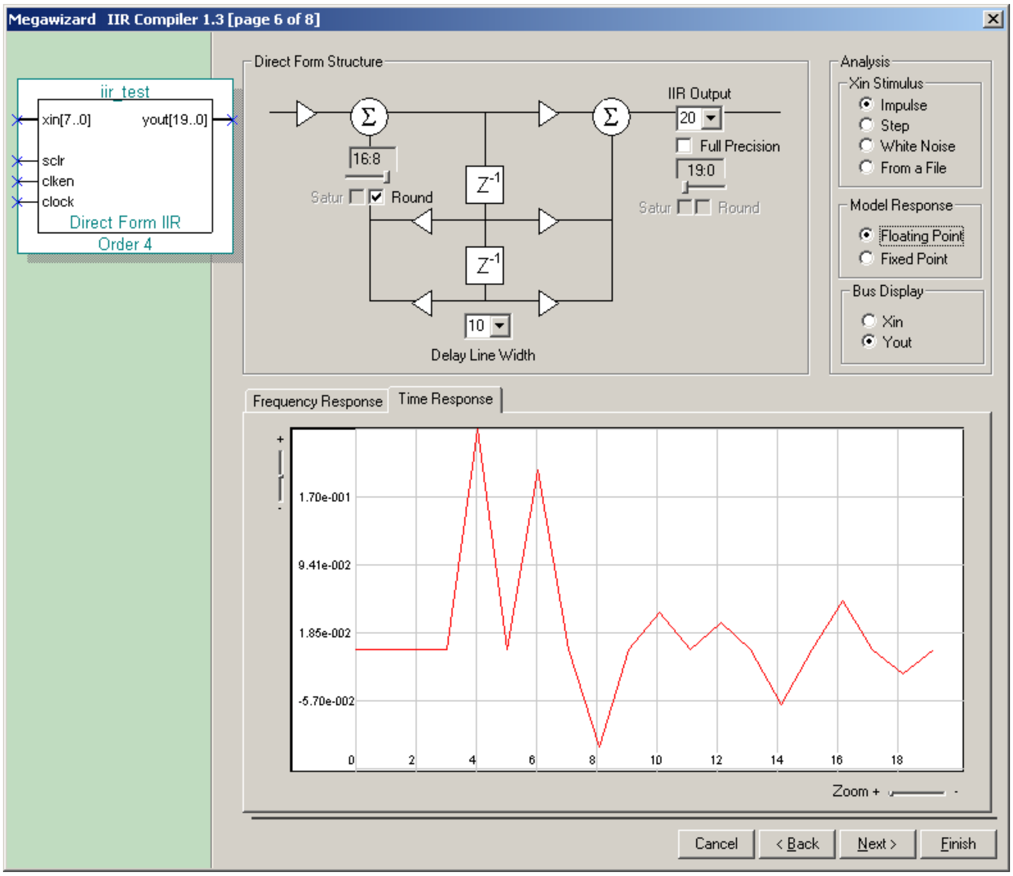
Master Time Bar: 47.231 ns Pointer: 169.89 ns Interval: 122.66 ns Start: En

| Name | Value at 47.23 ns |
|---|---|
| clock | B 0 |
| clken | B 1 |
| srst | B 1 |
| xin | U 0 |
| yout | S 0 |

## Simulate the Hardware Architecture

On this page you can simulate the hardware architecture using a bit-accurate system model. After setting the value of the scaled coefficients, the wizard lets you adjust the internal hardware architecture nodes for both the feedback and feed-forward paths. Use the slider bar to adjust the settings for both paths (you can also use the **Satur** and **Round** options to perform those functions). See Figure 12.

*Figure 12. Simulate the Hardware Architecture*



Megawizard IIR Compiler 1.3 [page 6 of 8]

Direct Form Structure: 16:8, Satur [ ], Round [✓]; Delay Line Width: 10; IIR Output: 20, Full Precision [ ], 19:0, Satur [ ], Round [ ]

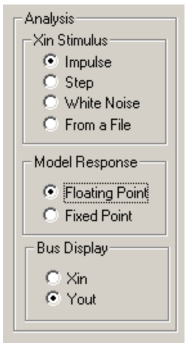
Analysis — Xin Stimulus: [✓ Impulse] Step White Noise From a File; Model Response: [✓ Floating Point] Fixed Point; Bus Display: Xin [✓ Yout]

The finite word length of the feedback and feed-forward paths may introduce non-linearity in the $H_{fixed}(z)$ transfer function. Therefore, the wizard lets you perform additional analysis and dynamically displays plots according to the settings you specify. You can view the plots in frequency or time, depending on which tab you select.

## *Simulation Setting*

If you choose the **Fixed Point** model response option the wizard plots (in blue) the bit-accurate simulation result of the IIR filter that will be implemented in the device. If you choose the **Floating Point** option the wizard plots (in red) the simulation result of the floating-point IIR filter that was specified on the coefficients wizard page. See Figure 13.

***Figure 13. Simulation Settings***



Analysis

Xin Stimulus: [✓ Impulse] Step White Noise From a File

Model Response: [✓ Floating Point] Fixed Point

Bus Display: Xin [✓ Yout]

You can toggle the view between frequency and time using tabs. Four input stimuli can be applied:

- Impulse
- Step
- White Noise—A new input white noise stimuli is generated every time you click on the white noise radio button.
- From a File—You can import `xin` values from an ASCII text file; the floating-point values can be generated from the MATLAB software or any other third-party tool. The result of the simulation is
  - In the time domain (up to 512 samples)
  - In the frequency domain
  - In the ASCII file **MYout.txt** (up to the number of samples present in the initial `xin` file).

The `xin` values are scaled and used for the MATLAB testbench or for the Quartus II Vector File stimulus.

The `xin` and `yout` options switch the plot from input data of the IIR filter or output data of the IIR filter, respectively.

### *Plotting*

The Time Response tab displays the time response of the filter. The X axis represents the sample number in time and the horizontal slider bar controls the simulation time length. The maximum simulation time length is 512 samples. For extended simulation time, Altera recommends you use either the bit-accurate MATLAB model generated by the wizard or the Quartus II VHDL or Verilog HDL output file.

### *Frequency Response*

The Y axis represents the amplitude of the filter response. The vertical slider bar zooms the response view in and out. The Y axis scale represents the dB attenuation of the filter based on the coefficients. A slider bar allows you to zoom the Y axis in and out. The X axis is an absolute frequency, which is the ratio of the filter frequency characteristic over the sample frequency. See Figures 14 and 15.

☞ To use the Up, Down, Left, and Right arrows on your keyboard to move the slider bars, click on the slider bar to select it.

***Figure 14. Time Response***

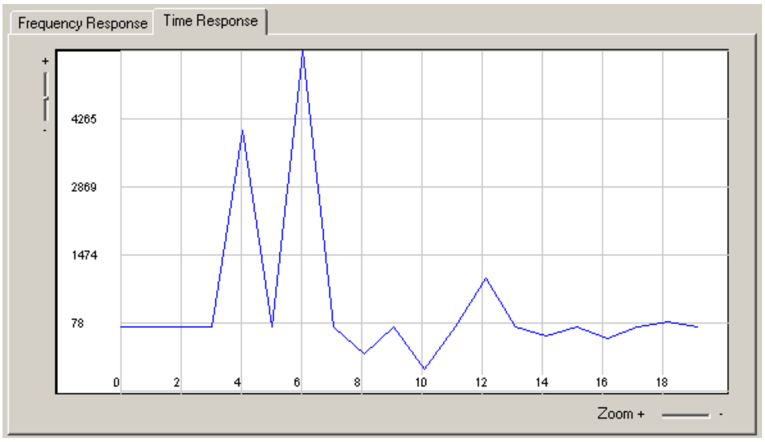
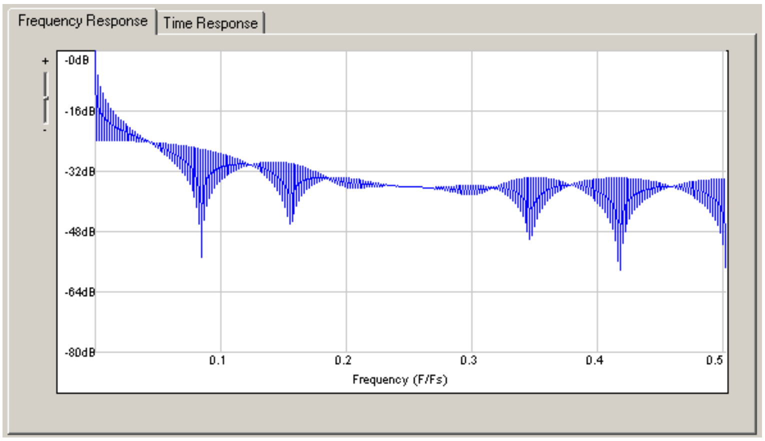


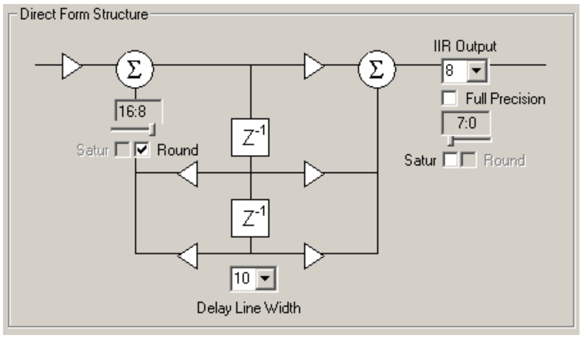
*Figure 15. Frequency Response (Step Stimulus)*



## Structure Setting

This section controls of the bit width of the feed-forward and feedback data of the IIR filter. See Figure 16.

*Figure 16. Structure Settings*



The **Delay Line Width** box sets the width of the delay element of the IIR filter. This global parameter applies to the cascaded and parallel structures; the tap delay line of each biquad has the same delay line width.

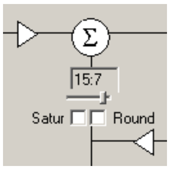
### Feedback Bit Width Control

The output feedback data path (*fb*) is defined by Equation 4 and shown in Figure 17.

*Equation 4*

$$fb_n = \sum_{i=1}^{n} -b_{ifixed} yout^{(n-i)}$$

*Figure 17. Feedback Settings*



$b_{ifixed}$ are the scaled coefficients of the feedback data path, *fb* is added to the incoming input data (`xin`). This adder width is equal to the delay line width. Therefore, the number of bits *fb* uses for this addition is smaller than the number of bits required to compute Equation 4 (full precision). A slider bar lets you select the slice of the *fb* bus that is added to `xin`. When the least significant bits (LSBs) are dropped you can choose to round or truncate using the **Satur** or **Round** options, respectively. When the most significant bits (MSBs) are dropped and possible overflows may occur, you can choose to saturate or truncate by turning the **Satur** option on and off, respectively. Saturation or rounding can reduce the bit width needed for the tap delay line at the cost of increasing the feedback data path delay. The effect of modifying any of these parameters is displayed in the simulation area.

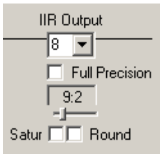
### Feed-Forward Bit Width Control

The output feedforward data path (*ff*) is defined by Equation 5 and shown in Figure 18.

*Equation 5*

$$ff_n = \sum_{i=1}^{n} a_{ifixed} xin^{(n-i)}$$

*Figure 18. Feedforward Settings*



$a_{ifixed}$ are the scaled coefficients of the feed-forward data path. If the **Full Precision** option is turned on the wizard computes the number of bits required for Equation 5, which is the full output resolution bit width. When the option is turned off you can choose an output bit width greater or smaller than the full precision.

For a smaller number of bits, the output slider bar allows you to select the slice of *ff* to keep. When LSBs are dropped you can choose to round or truncate using the **Satur** or **Round** options, respectively. When MSBs are dropped and possible overflows may occur, you can choose to saturate or truncate by turning the **Satur** option on and off, respectively. Saturation or rounding may reduce the bit width needed for the tap delay line at the cost of increasing the feedback data path delay. The effect of modifying any of these parameters is instantly displayed in the simulation area.

### Guidelines

Setting the most optimum bit width parameters consists of finding the values of the slide bars / combo box / check box width so that the simulation response of the fixed-point model is as identical as possible to the initial floating-point model. Use the following guidelines when you set the bit width parameters of the fixed-point IIR filter.

- First, use full precision for the output feed-forward data path. Choose the **Impulse** stimulus and the time domain plot.
- Compare the fixed-point and floating-point views, adjusting the width parameter of the feedback path.

  a. Choose the **Step** stimulus in the time domain plot. Tune the parameters.

  b. Choose the white noise stimulus in the frequency domain, adjusting the feedback parameters as needed.

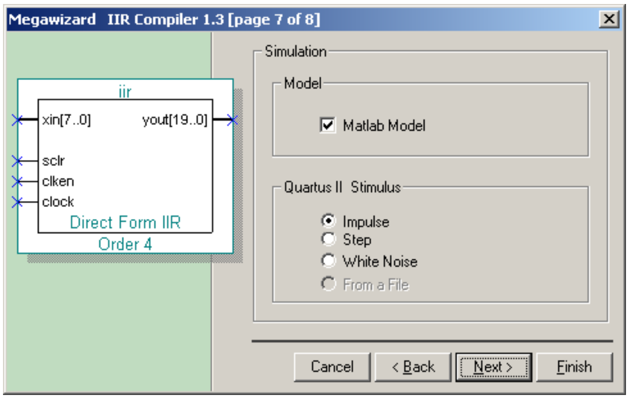
  c. Reduce the output precision of the feed-forward path to find the match.

If, after following the steps above, you cannot find any matching fixed-point filters, you can increase the coefficient bit width and the delay line bit width or change architectures (cascade or parallel).

## Specify the Simulation Output Files

Choose the simulate output files. The wizard can generate MATLAB, VHDL, and Verilog HDL simulation models as well as Quartus II stimulus. See Figure 19.

*Figure 19. Simulation Output File Settings*


Megawizard IIR Compiler 1.3 [page 7 of 8]

Simulation

Model

[✓] Matlab Model

Quartus II Stimulus

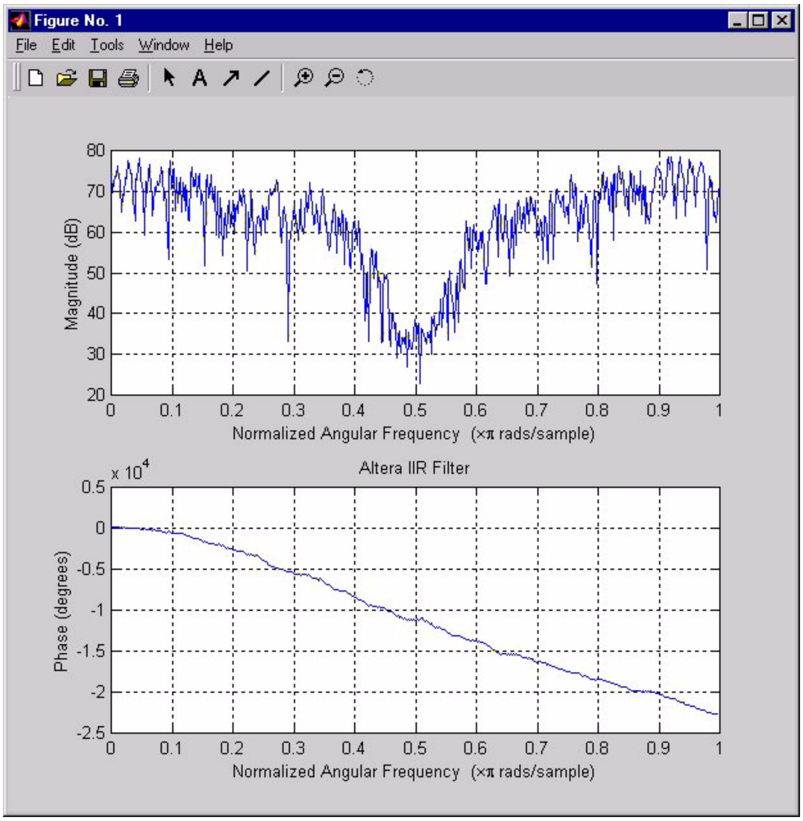
[● Impulse] Step White Noise From a File

## Using the MATLAB Model

You can use the wizard-generated files to simulate the filter in the MATLAB software. The generated MATLAB model requires a library file, **iirtrim.dll**, which is saved into the **\iir-v1.3.2\simlib\matlab** directory when you install the IIR Compiler. This file must be located in the MATLAB path or copied into your MATLAB working directory.

Execute <*your filter name*> at the MATLAB prompt. This action runs the MATLAB simulation of the bit-accurate model, plots the frequency and phase response, and shows the three-plane representation of the scaled coefficients. Figure 20 shows an example of analyzing the IIR phase of the fixed-point MATLAB filter generated by the wizard.

*Figure 20. MATLAB Analysis*



## Using the Core with Simulink & DSP Builder

You can use Simulink blocks, Altera DSP Builder blocks, and the IIR Compiler to build a model of your design in the Simulink software. Refer to "DSP Builder Support" on page 10, "DSP Builder Feature & Simulation Support" on page 44, and the *DSP Builder User Guide* for more information.

## Compile & Place & Route the Design

You can use the Quartus II software to compile and place-and-route your design. Refer to Quartus II Help for instructions on performing compilation. After you have verified that your design is functionally correct, you are ready to perform system verification.

## Perform Synthesis Compilation & Post-Routing Simulation

As a standard feature, Altera's Quartus II software works seamlessly with tools from all EDA vendors, including Cadence, Exemplar Logic, Mentor Graphics, Synopsys, Synplicity, and Viewlogic.

☞ After you have licensed the MegaCore function, you can use the NativeLink™ feature to integrate the Quartus II software with other EDA tools easily. See Quartus II Help for details.

The following sections describe the design flow to compile and simulate your custom MegaCore design with the Quartus II software and a third-party EDA tool. To synthesize your design in a third-party EDA tool and perform post-route simulation, perform the following steps:

1. Create your IIR Compiler instance using the IIR Compiler MegaWizard Plug-In.

2. Create your custom design instantiating an IIR function.

3. Select **Compile mode** (Processing menu).

4. Specify the Compiler settings in the **Compiler Settings** dialog box (Processing menu) or use the Compiler Settings wizard.

5. Specify the user libraries for the project and the order in which the Compiler searches the libraries.

6. Specify the input settings for the project. Choose **EDA Tool Settings** (Project menu). Select **Custom EDIF** in the Design entry/synthesis tool list. Click **Settings**. In the **EDA Tool Input Settings** dialog box, make sure that the relevant tool name or option is selected in the **Design Entry/Synthesis Tool** list.

7. Depending on the type of output file you want, specify Verilog HDL output settings or VHDL output settings in the **General Settings** dialog box (Project menu). Use the **1993 VHDL language** option.

8. Compile your design. The Quartus II Compiler synthesizes and performs place-and-route on your design, and generates output and programming files.

9. Import your Quartus II-generated output files (**.edo**, **.vho**, **.vo**, or **.sdo**) into your third-party EDA tool for post-route, device-level, and system-level simulation.

## Configuring a Device

After you have compiled and analyzed your design, you are ready to configure your targeted Altera device. If you are evaluating the MegaCore function with the OpenCore feature, you must license the function before you can generate configuration files.

*Notes:*

## Functional Description

IIR filters of order *n* are characterized by Equation 6.

**Equation 6**

$$Y_n = \sum_{i=0}^{n} a_i X_{(n-i)} - \sum_{i=1}^{n} b_i Y_i$$

Where:

- $X_i$ is the discrete input data
- $Y_n$ the resulted filtered output data
- The vector $a_i$ contains the coefficient value of the feed-forward path
- The vector $b_i$ contain the coefficient value of the feedback path

The coefficients $a_i$, $b_i$ set the characteristic of the filter. The values that are typically determined at the system level are the numerator and denominator, respectively, of the floating point value of the system transfer function in the Z-plane representation defined in Equation 7.

**Equation 7**

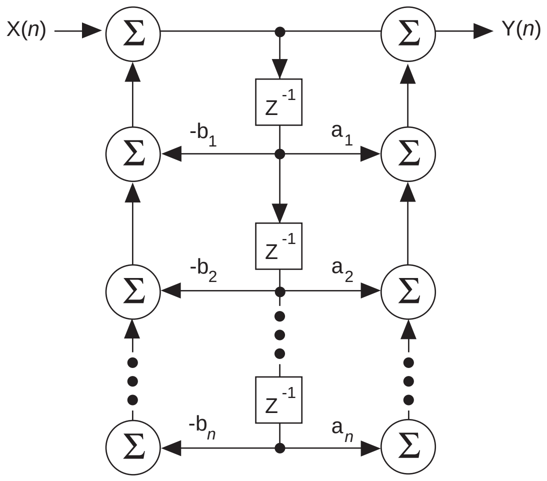
$$H(z) = \frac{\sum_{i=0}^{n} a_i Z^{-i}}{1 + \sum_{i=1}^{n} b_i Z^{-i}}$$

The roots of the polynomial equation of the numerator and the roots of the polynomial equation of the denominator are called the zeros and the poles of the IIR filter, respectively. In the Z plane, the filter is stable when the poles are entirely inside the unit circle or are coincident with zeros on the unit circle.

## Direct Form Structure

The hardware realization of Equation 7 consists of adding weighted past input values to weighted past output values as shown in Figure 21.

*Figure 21. Addition of Weighted Past Input & Output Values*



This structure is called direct form II because it is directly drawn from Equation 6. For high orders, the direct form structure may be sensitive to finite word-length effects. That is, a small change in a filter coefficient introduced by quantization affects the pole-zero location and therefore affects the filter stability and characteristics. One option to address this issue is to increase the internal precision of the delay line or the coefficient width. The IIR Compiler supports up to 32 bits of internal precision for the tap delay line as well as the coefficients.

Another option is to rearrange the coefficients for different hardware structures. $H(z)$ is rearranged in sections as second order and first order IIR filters that are connected serially (cascaded implementation) or in parallel. Each second-order filter is called a biquad.

Rearranging the coefficients of the direct form into cascaded or parallel forms can be performed in third-party system tools such as MATLAB. You can then import the coefficients tailored for cascaded or parallel forms into the compiler wizard. You can also let the IIR Compiler wizard perform this task by specifying the form on the first page of the wizard.

## Cascaded Structure

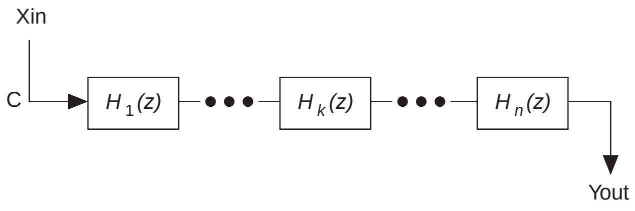
In a cascaded implementation, the system transfer function of the order *n* IIR is rearranged by factorizing the second and first order filter. An example is shown in Equation 8.

### Equation 8

$$H(z) = C \prod_{k=1}^{(N+1)/2} \frac{a_{0k} + a_{1k}z^{-1} + a_{2k}z^{-2}}{1 + b_{1k}z^{-1} + b_{2k}z^{-2}} = C \prod_{k=1}^{(N+1)/2} H_k(z)$$

Each product term of Equation 8 is a second or first order IIR filter. If *n* is an odd number, the last product term is a first-order IIR filter. Figure 22 shows the cascaded structure.

### Figure 22. Cascaded Structure



## Parallel Structure

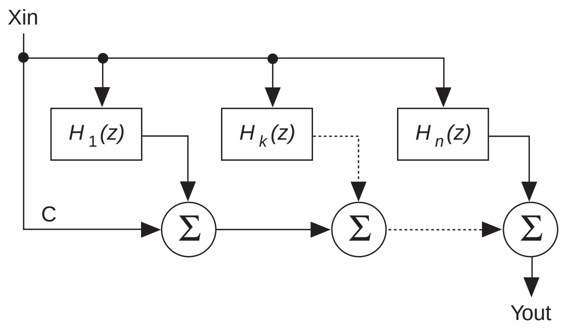
In a parallel implementation, the system transfer function of the *n* IIR filter is rearranged in a sum of *n*/2 second-order and first-order filter. The IIR system is obtained by performing a partial-fraction expansion of *H*(*z*) as shown in Equation 9 and rearranging the residue term into a biquad.

### Equation 9

$$H(z) = C + \sum_{k-1}^{(n+1)/2} \frac{a_{0k} + a_{1k}z^{-1}}{1 + b_{1k}z^{-1} + b_{2k}z^{-2}} = C + \sum_{k-1}^{(n+1)/2} H_k(z)$$

Each sum term of Equation 9 is a second order IIR filter. If N is an odd number, the last product term is a first order IIR filter. Figure 23 shows the parallel structure.

*Figure 23. Parallel Structure*



Parallel structures are generally larger than cascaded structures; however, they generally run at a faster data rate.

## DSP Builder Feature & Simulation Support

You can create Simulink Model Files (**.mdl**) using IIR Compiler and DSP Builder blocks. DSP Builder supports the following IIR Compiler options:

- Direct Form II structures
- Cascaded biquad structures

DSP Builder does not support the following IIR Compiler options:

- Maximum latency

After you create your model, you can perform simulation. DSP Builder supports the simulation types shown in Table 6 for IIR Compiler.

*Table 6. IIR Compiler Simulation File Support in DSP Builder*

| Simulation Type | Simulation Flow |
|---|---|
| Precompiled ModelSim model for RTL functional simulation | Not Supported |
| VHDL Output File (**.vho**) models for timing simulation | You can generate a **.vho** after you have purchased a license for your MegaCore function. Refer to the “VHDL Output File (.vho)“ topic in Quartus II Help for more information. |
| Visual IP Models | Not Supported |
| Quartus II simulation | The DSP Builder SignalCompiler block generates a Quartus II simulation vector file on-the-fly. |

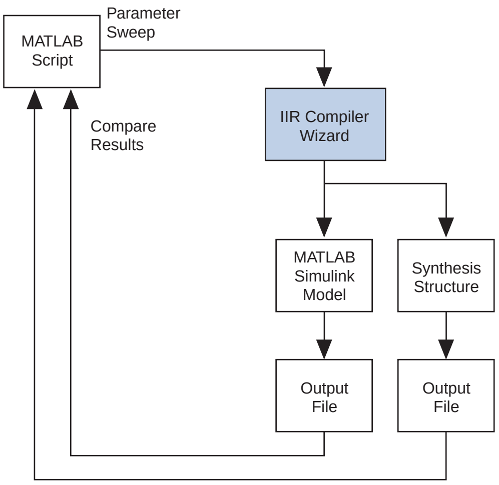
For more information on DSP Builder, see "DSP Builder Support" on page 10.

## Core Verification

Before releasing a version of the IIR Compiler, Altera runs a comprehensive regression test that executes the wizard to create the instance files. Next, Quartus II Vector Files are created and the results are compared to the MATLAB/Simulink software using the Quartus II simulator.

The regression suite covers various parameters such as architecture options, number of taps, coefficient and data bit widths. Figure 24 shows the regression flow.

*Figure 24. Regression Flow*



## Signals

The IIR Compiler function has the signals shown in Table 7.

*Table 7. IIR Compiler Signals*

| Signal | Direction | Description |
|---|---|---|
| `xin[]` | Input | Input data to be filtered. The input bit width is set in the IIR Compiler wizard. |
| `clock` | Input | Clock. |
| `srst` | Input | Synchronous reset (optional), active high. |
| `clken` | Input | Clock enable (optional), active high. |
| `yout[]` | Output | Filtered output data. The output bit width is set in the IIR Compiler wizard. |

## MegaWizard Plug-In Manager

The IIR Compiler function has an interactive wizard-driven interface that allows you to create custom functions easily. You can launch the MegaWizard Plug-In Manager from within the Quartus II software, or you can run it from the command line. The wizard allows you to input your choice of parameters, verifies that all choices are valid, and generates a custom MegaCore function in VHDL, AHDL, or Verilog HDL, which you can integrate into your system design. When you finish going through the wizard it outputs some or all of the following files, depending on parameters selected in the wizard.

- Text Design File (**.tdf**) used to instantiate an instance of the IIR filter in your design
- Simulation output files
  - A Vector File (**.vec**) (<*variation name*>**.vec**) for simulation in the Quartus II software
  - MATLAB and Simulink files used for simulation in MATLAB Simulink
  - VHDL and Verilog HDL models used for simulation in other EDA tools
- Symbol File (**.sym**) or Block Design File (**.bdf**) used to instantiate the filter into a schematic design

## References

Arthur Bernard Williams, Fred J. Taylor. ***Electronic Filter Design Handbook***, 2nd ed. McGraw-Hill Companies, November 1988.

Madisetti, Vijay. ***VLSI Digital Signal Processors: An Introduction to Rapid Prototyping and Design Synthesis***. Butterworth-Heinemann, November 1994.

Vaidyanathan, P. P. ***Multirate Systems and Filter Banks***. Simon & Schuster Trade, October 1992.

Bingham, John A. C. ***The Theory and Practice of Modem Design***. Wiley, John & Sons, Incorporated, November 1990.

William H. Press , Saul A. Teukolsky , William T. Vetterling, Brian P. Flannery. ***Numerical Recipes in C : The Art of Scientific Computing***. The Press Syndicate of the University of Cambridge, October 1992.

Luc Semeria Abhijit Ghosh. *Methodology for Hardware/Software Co-verification in C/C++*.

The MathWorks, Inc. ***Signal Processing Toolbox User's Guide***. Natick, MA. September 2000.

*Notes:*