

## Introduction

The filtering reference designs provided in the *DSP Development Kit, Stratix Edition*, and in the *DSP Development Kit, Stratix Professional Edition*, show you how to use the Altera® DSP Builder for system design, simulation, and board-level verification. DSP Builder is a digital signal processing (DSP) development tool that interfaces The MathWorks industry leading system-level DSP modeling tool Simulink with the Altera Quartus® II development software. DSP Builder provides a seamless design flow in which you can perform algorithmic design and system integration in the MATLAB and Simulink software and then port the design to hardware description language (HDL) files for use in the Quartus II software.

Using DSP Builder, you can automatically generate a register transfer level (RTL) design and an RTL testbench from Simulink. These files are pre-verified RTL output files optimized for use in the Altera Quartus II software for rapid prototyping. The built-in DSP Builder **SignalTap® II Analysis** block allows you to capture signal activity from internal Stratix device nodes, while the system under test runs at system speed in hardware. You can import SignalTap II data into the MATLAB workspace for further analysis. This development flow is easy and intuitive even if your experience designing with programmable logic design software is not extensive.

The Stratix Filtering Lab uses the following items:

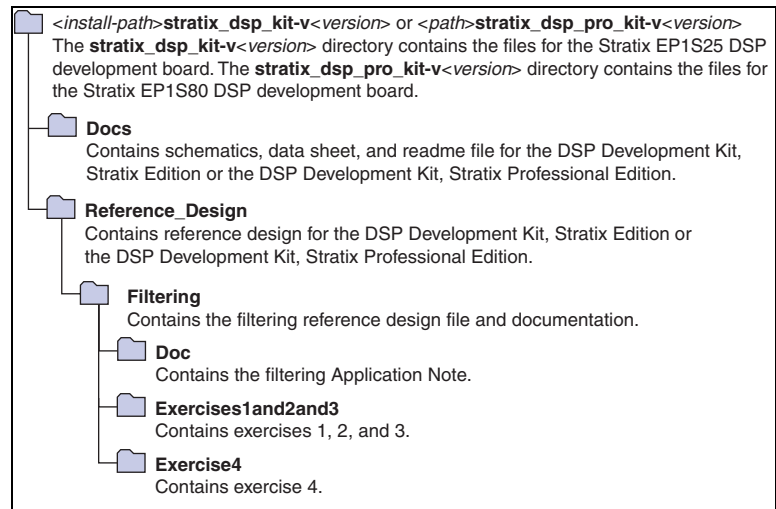
- Quartus II software
- DSP Builder with the SignalTap II logic analyzer read-back feature
- Altera finite impulse response (FIR) Compiler MegaCore® function
- Altera numerically controlled oscillators (NCO) Compiler MegaCore function
- The MathWorks MATLAB
- The MathWorks Simulink
- Mentor Graphics® ModelSim®-Altera, ModelSim PE, or ModelSim SE simulation software
- Stratix EP1S25 DSP development board or Stratix EP1S80 DSP development board

**Figure 2** shows the top-level schematic for the filtering reference design. Two NCOs generate a 1-MHz sinusoidal signal and a 10-MHz sinusoidal signal respectively. The signals are added together on-chip before they pass through a digital-to-analog (D/A) converter on the Stratix DSP

development board. The resulting analog signal is looped back to an analog-to-digital (A/D) converter on the board and then passed to an on-chip, low-pass filter with a cut-off frequency of 3 MHz. The low-pass filter removes the 10-MHz sinusoidal signal and allows the 1-MHz sinusoidal signal through to the `fir_result` output.

When you install the software from the *DSP Development Kit, Stratix & Stratix Professional Edition* CD-ROM, the design files are installed in the directory structure, as shown in [Figure 1](#).

**Figure 1. Filtering Reference Design Directory Structure**



This application note provides the following exercises:

- [“Exercise 1: Review the Filtering Design” on page 5](#)—Review the filtering design using DSP Builder.
- [“Exercise 2: Simulate the Model in Simulink” on page 15](#)—Analyze the DSP Builder-generated models and simulate the filtering design in Simulink.
- [“Exercise 3: Perform RTL Simulation” on page 19](#)—Perform RTL simulation using the ModelSim software simulation tool.
- [“Exercise 4: Analyze the Results in Hardware” on page 23](#)—Program the Stratix device with the filtering design and use the SignalTap II read-back feature in DSP Builder to capture data from internal Stratix device nodes while the design runs at system speed. You then compare the results from SignalTap II analysis with the simulation results from Exercise 2 to verify that the design is functioning correctly in hardware.

## Before You Begin

The instructions in this application note assume that you have already installed the software provided with the *DSP Development Kit, Stratix Edition*, or the *DSP Development Kit, Stratix Professional Edition* on your PC.



For more information on installation instructions, see the *DSP Development Kit, Stratix & Stratix Professional Edition Getting Started User Guide*.

You must have the following software installed on your PC:

- Quartus II software starting with version 4.2
- DSP Builder version 2.2.1
- FIR Compiler MegaCore function starting with version 3.2.0
- NCO Compiler MegaCore function starting with version 2.2.1
- The MathWorks MATLAB version 7.0.1
- The MathWorks Simulink version 6.1
- ModelSim-Altera software, ModelSim PE, or ModelSim SE software version 5.8d



This application note assumes that you have installed the software into the default locations.

You must run a DSP Builder setup script once, following the installation of the MegaCores. The script updates DSP Builder for other newly installed or upgraded MegaCores.



For more information see the *Using MegaCore Functions* chapter in the *DSP Builder User Guide*.

To run the setup script, follow these steps:

1. Run the MATLAB software.
2. In the **Current Directory** browser, browse to the directory where DSP Builder is installed:

```
<dsp_builder_install_dir>\DSPBuilder\AltLib
```

3. Run the script by typing `setup_dspbuilder` at the MATLAB prompt in the workspace.

## Exercise 1: Review the Filtering Design

To review the filtering design, follow these steps:

1. Run the MATLAB software.
2. In the **Current Directory** browser, browse to one of the following directories:
  - For the Stratix EP1S25 DSP development board:  

```
c:\altera\kits\stratix_dsp_kit-v<version>  
\Reference_Design\Filtering\Exercises1and2and3
```
  - For the Stratix EP1S80 DSP development board:  

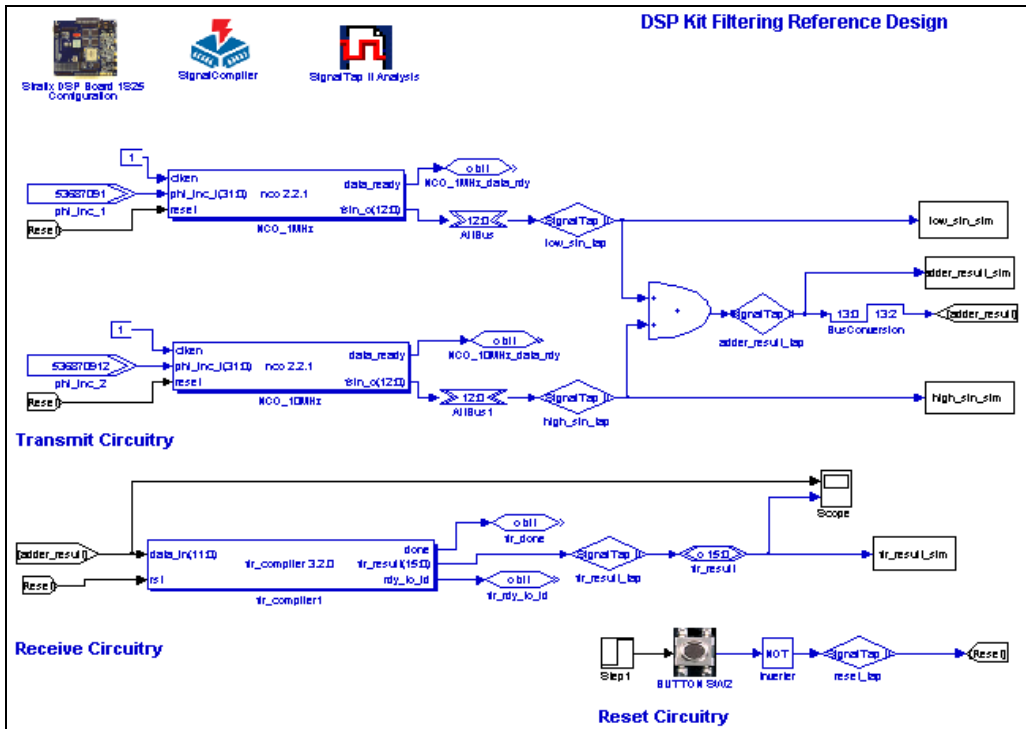
```
c:\altera\kits\stratix_dsp_pro_kit-v<version>  
\Reference_Design\Filtering\Exercises1and2and3
```
3. Choose **Open** (File menu) and select the file **filter\_design.mdl** (.mdl is a Simulink Model File).
4. Review the Simulink design (see [Figure 2](#)).

The filtering design contains a combination of OpenCore® Plus DSP MegaCore functions and DSP Builder blocks. The OpenCore Plus feature lets you test-drive Altera MegaCore functions for free. You can verify the functionality of a MegaCore function quickly and easily, as well as evaluate its size and speed before making a purchase decision. The hardware evaluation feature allows you to generate time-limited programming files for designs that include Altera MegaCore function. You can perform board-level design verification before deciding to purchase licenses for each used MegaCore functions. You only need to purchase a license when you are completely satisfied with a MegaCore's functionality and performance, and would like to take your design to production.



For more information on the OpenCore Plus hardware evaluation, see *AN320: OpenCore Plus Evaluation of Megafunctions*.

Figure 2. Simulink Design for Exercises 1, 2, & 3 (filter\_design.mdl File)

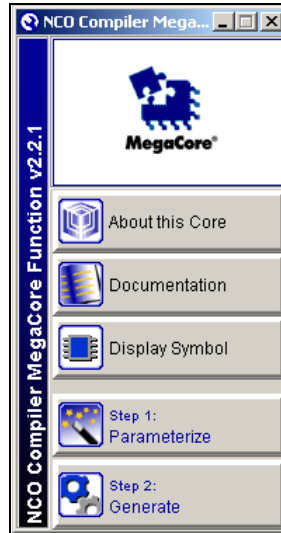


## Review the NCO\_1MHz MegaCore Function

To launch IP Toolbench for the NCO 1MHz Compiler MegaCore function, follow these steps:

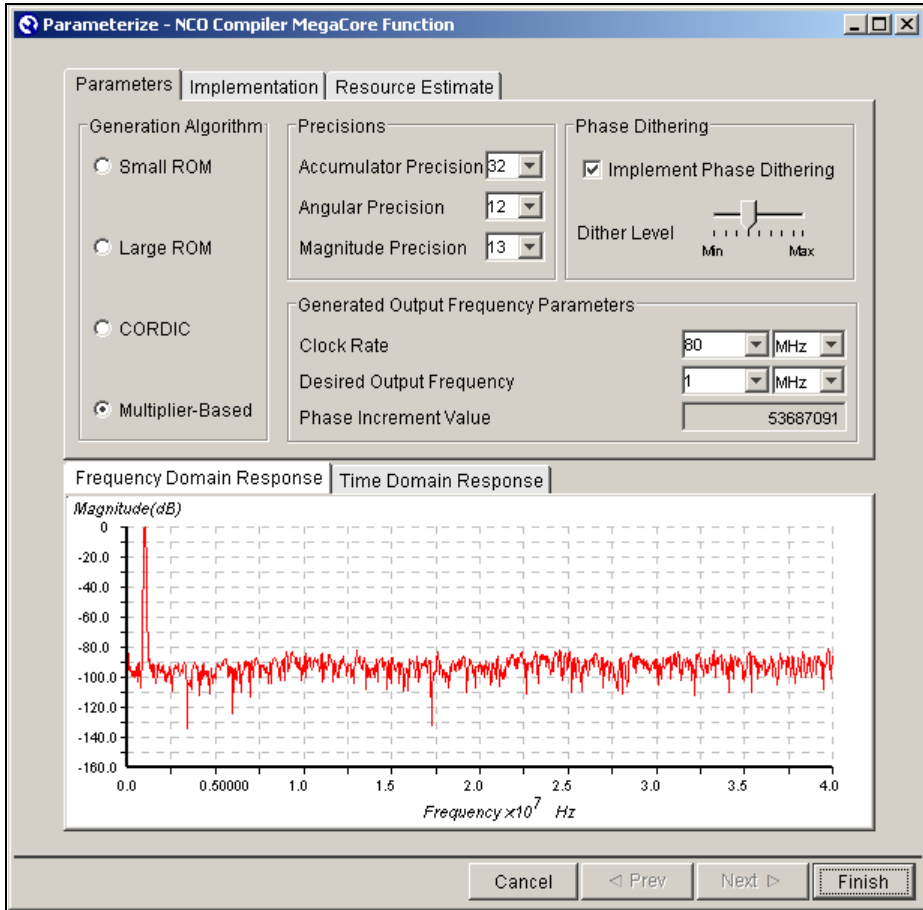
1. Double-click the NCO\_1MHz block to launch IP Toolbench for the NCO Compiler MegaCore function (see [Figure 3](#)).

**Figure 3. IP Toolbench for NCO Compiler MegaCore Function**



2. Click **Step 1: Parameterize** to review the parameters for the NCO\_1MHz block. The NCO\_1MHz block generates a 1-MHz sinusoidal signal (see [Figure 4](#)).

**Figure 4. 1-MHz Sinusoidal Signal**





The NCO block is implemented using the multiplier-based architecture, which reduces memory usage by using the hardware multipliers in the Stratix device. Table 1 shows the parameters that you can set in IP Toolbench **Parameters** tab.

<b>Table 1. NCO Compiler Parameters for NCO_1MHz</b>	
<b>Parameter</b>	<b>Value</b>
<b>In the Parameters Tab</b>	
Accumulator Precision	32 bits
Angular Precision	12 bits
Magnitude Precision	13 bits
Generation Algorithm	Multiplier-Based
Implement Phase Dithering	Yes
Dither Level	5
Clock Rate	80 MHz
Desired Output Frequency	1 MHz
<b>In the Implementation Tab</b>	
Architecture	Use Dedicated Multiplier(s)
Outputs	Single Output
Device Family	Stratix
Number of Channels	1

- Click **Cancel** to exit IP Toolbench when you are finished reviewing the parameter settings.

## Review the NCO\_10MHz MegaCore Function

To launch IP Toolbench for the NCO 10MHz Compiler MegaCore function, follow these steps:

1. Double-click the NCO\_10MHz block (see [Figure 2 on page 6](#)) to launch IP Toolbench for the NCO Compiler MegaCore function.
2. Click **Step 1: Parameterize** to review the parameters for the NCO\_10MHz block. The NCO\_10MHz block generates a 10-MHz sinusoidal signal, as shown in [Figure 5](#).

**Figure 5. 10-MHz Sinusoidal Signal**

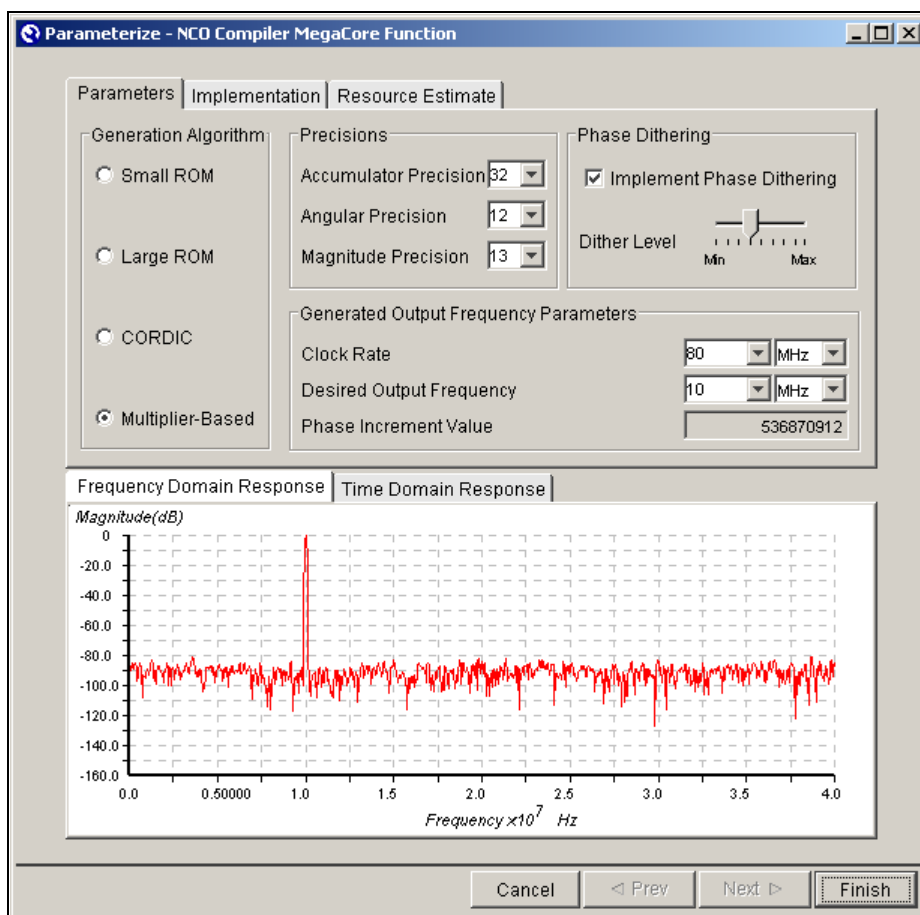


Table 2 shows the parameters that you can set in IP Toolbench Parameters tab.

<b>Table 2. NCO Compiler Parameters for NCO_10MHz</b>	
<b>Parameter</b>	<b>Value</b>
<b>In the Parameters Tab</b>	
Accumulator Precision	32 bits
Angular Precision	12 bits
Magnitude Precision	13 bits
Generation Algorithm	Multiplier-Based
Implement Phase Dithering	Yes
Dither Level	5
Clock Rate	80 MHz
Desired Output Frequency	10 MHz
<b>In the Implementation Tab</b>	
Architecture	Use Dedicated Multiplier(s)
Outputs	Single Output
Device Family	Stratix
Number of Channels	1

The **NCO\_10MHz** block contains the same parameter values as the **NCO\_1MHz** block, except for the constant value that is fed into the phase increment input and for the desired output frequency. This constant value determines the frequency of the NCO sinusoidal output. The NCO MegaWizard® Plug-In calculates the constant value when you enter the clock period and the desired output frequency in the wizard. [Figure 4 on page 8](#) shows the calculated result for a 1-MHz sine wave at 53,687,091. The chosen clock frequency corresponds to the 80-MHz oscillator on the Stratix EP1S25 DSP development board or the Stratix EP1S80 DSP development board. Similarly, the desired output frequency of 10 MHz yields a phase increment value of 536,870,912 (see [Figure 5](#)).

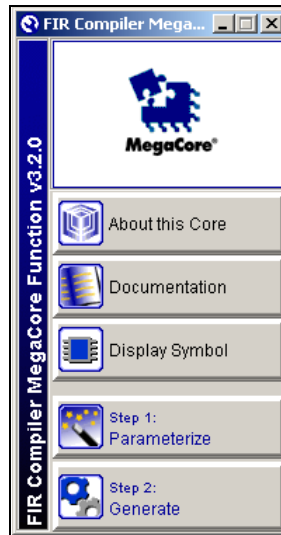
3. Click **Cancel** to exit IP Toolbench when you are finished reviewing the parameter settings.

## Review the fir\_compiler MegaCore Function

To launch IP Toolbench for the FIR Compiler MegaCore function, follow these steps:

1. Double-click the **fir\_compiler** block to launch IP Toolbench for the FIR Compiler MegaCore function (see [Figure 6](#)).

**Figure 6. IP Toolbench for FIR Compiler MegaCore Function**



2. Click **Step 1: Parameterize** to review the parameters for the **fir\_compiler** block (see [Figure 7](#)).

The **FIR filter** block is a 35-tap, low-pass filter with a cut-off frequency of 3 MHz. It is designed to filter out the 10-MHz sinusoidal signal.

Figure 7. FIR Filter Parameters

**Parameterize - FIR Compiler MegaCore Function**

Coefficients Specification - (Low Pass Set [1])

New Coefficient Set Edit Coefficient Set Remove Coefficient Set

Low Pass Set [1]

Plot Option Fixed/Floating Coefficients  Dark Background

Frequency Response Time Response & Coefficient Values

Coefficients Scaling Auto Bit Width 14

Architecture Specification

Device Family Stratix  Force Non-Symmetric Structure

Structure Distributed Arithmetic : Fully Parallel Filter

Pipeline Level 1

Data Storage Logic Cells Multiplier Implementation Logic Cells

Coefficient Storage Logic Cells  Coefficients Reload  Use Single Clock

Rate Specification

Single Rate Factor 2

Flow Control

Input Specification

Number of Input Channels 1

Input Number System Signed Binary

Input Bit Width 12

Output Specification

Full Resolution Bit Width is 29

Based on Method Actual Coefficients

Output Number System Custom Resolution

Bits to keep

0 16 13

MSB LSB

Truncate Round

Resource Estimates

Resource	Utilization
Logic Cells	1866
M512	0
M4K	0
M-RAM	0
Multipliers	0

Based on Quartus II 4.2

Throughput

Input data must be valid for 1 clock period

Output data will be valid for 1 clock period

Output data is updated every clock period

Info: Force non-symmetric structure is selectable only if coefficients reload is selected.

Info: Coefficients reload is enabled only when coefficient storage is set to "M512" or "M4K".

Cancel Finish

Table 3 shows the parameters that you can set in IP Toolbench **Parameters** tab.

<i>Table 3. FIR Compiler Parameters</i>	
<b>Parameter</b>	<b>Value</b>
Filter Type	Low Pass
Window Type	Blackman
Sample Rate	8e7 Hz
Number of Coefficients	35
Cutoff Frequency	3e6 Hz
Rate Specification (Multi-rate Filter Settings)	Single Rate
Bit Width (Coefficients)	14
Input Number System	Signed Decimal (A/D width)
Input Nit Width	12
Output Number System	Custom Resolution
Bits to Keep	16 bits
Least Significant Bit (LSB) (Round)	13 bits
Most Significant Bit (MSB) (Truncate)	0 bit
Structure	Distributed Arithmetic : Fully Parallel Filter
Device Family	Stratix
Pipeline Level	1
Data Storage	Logic Cells
Coefficient Storage	Logic Cells

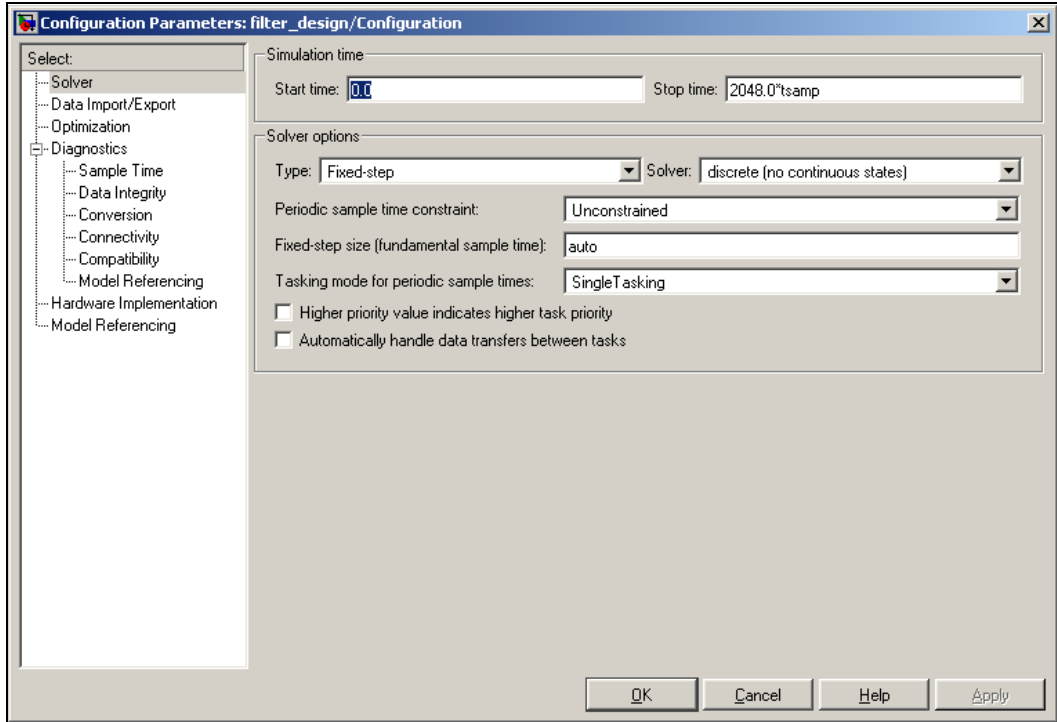
3. Click **Cancel** to exit IP Toolbench after you have finished reviewing the parameter settings.
4. Close the **filter\_design.mdl** file.

## Exercise 2: Simulate the Model in Simulink

To simulate the model in the Simulink software, follow these steps:

1. Choose **Configuration Parameters** (Simulation menu). The settings for the Simulink simulation parameters should be the same as shown in [Figure 8](#). If not, change them to match [Figure 8](#).

**Figure 8. Simulink Simulation Parameters**



2. Click **OK**.
3. Start the simulation by choosing **Start** (Simulation menu).

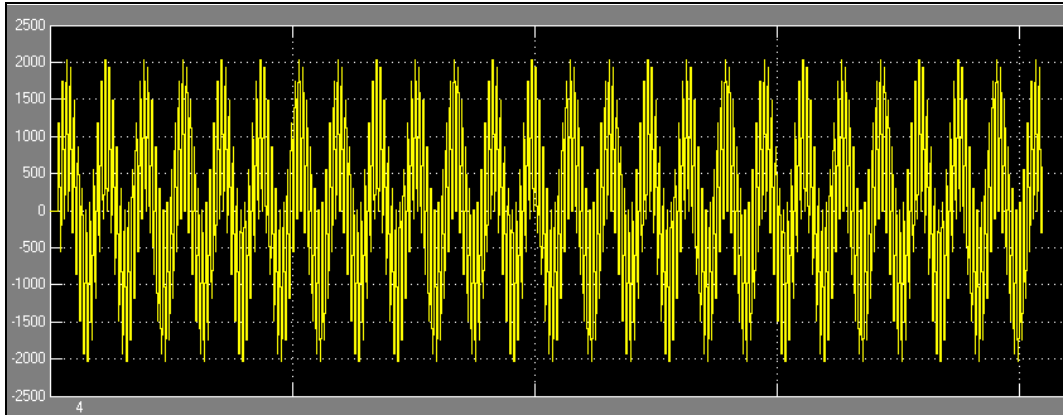


Ignore MATLAB warnings about unconnected input or output pins.

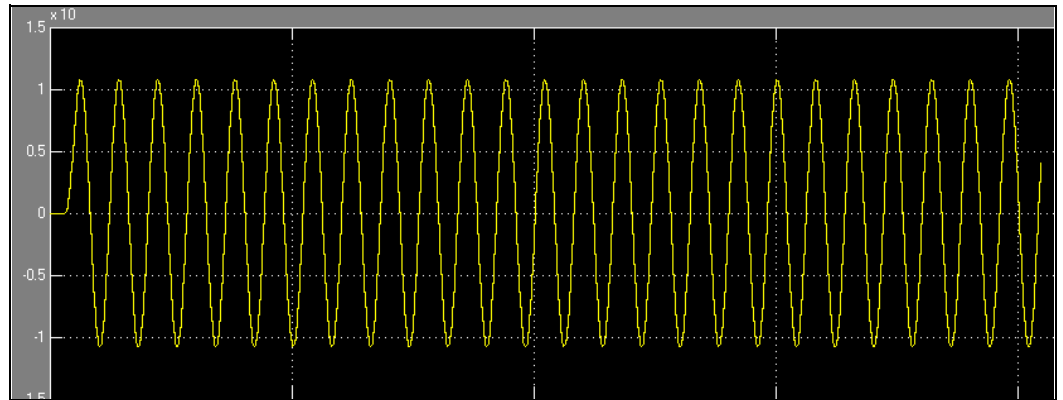
4. Double-click the **Scope** block to view the filtered and unfiltered signals in the time domain.

- Click the **binocular icon** to auto-scale the waveforms. **Figure 9** and **Figure 10** show the scaled waveforms in the time domain.

**Figure 9. Time Domain Plot of adder\_result\_sim—Unfiltered Data**



**Figure 10. Time Domain Plot of fir\_result\_sim—Filtered Data**



- Switch to the MATLAB window.



7. To view the frequency response of the filtered and unfiltered signals, use the `plot_fft.m` file, which is included with the lab.
  - a. To view the unfiltered data, type the following command in the MATLAB command window:

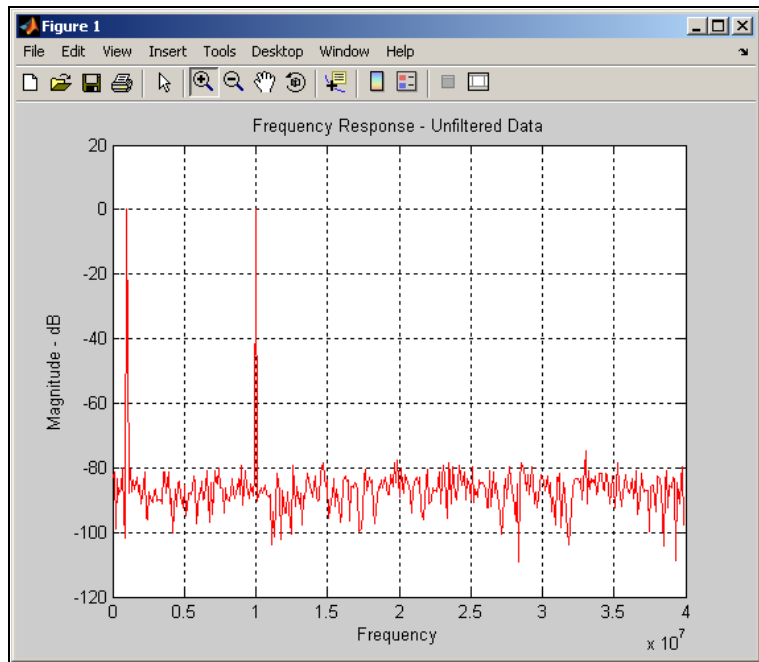
```
plot_fft(adder_result_sim, 'Frequency Response - Unfiltered Data', 8e7) ←
```

where:

- `adder_result_sim` is the name of the signal at the output of the adder
- `Frequency Response - Unfiltered Data` is the title of the plot
- `8e7` is the sampling frequency (80 MHz), which is well above the Nyquist frequency

A MATLAB plot displays the frequency response of the unfiltered data (see [Figure 11](#)).

**Figure 11. FFT Response of `adder_result_sim` - Unfiltered Data**



- b. To view the frequency response of the filtered data, type the following command in the MATLAB command window:

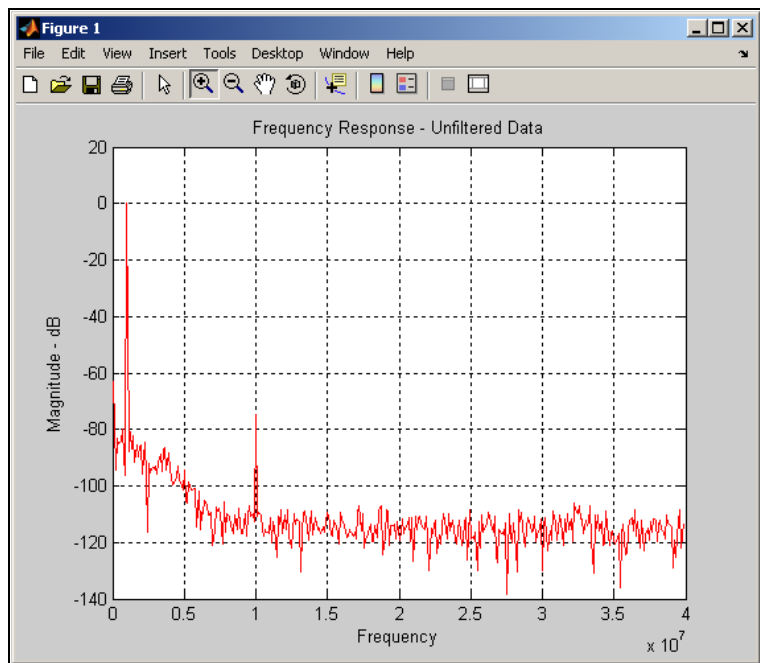
```
plot_fft(fir_result_sim, 'Frequency Response - Filtered Data', 8e7) ←
```

where:

- `fir_result_sim` is the name of the signal at the output of the FIR filter
- `Frequency Response - Filtered Data` is the title of the plot
- `8e7` is the sampling frequency (80 MHz), which is well above the Nyquist frequency

A MATLAB plot displays the frequency response of the filtered data (see [Figure 12](#)).

**Figure 12. FFT Response of `fir_result_sim` - Filtered Data**

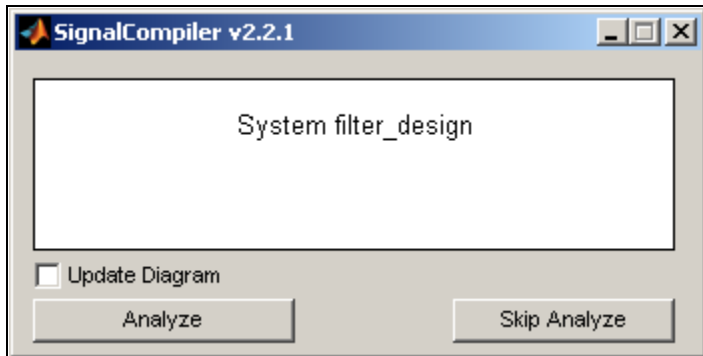


## Exercise 3: Perform RTL Simulation

To generate the simulation files for the filtering design example, follow these steps:

1. Double-click the **SignalCompiler** block in your model to display the SignalCompiler Analyze feature (see [Figure 13](#)).

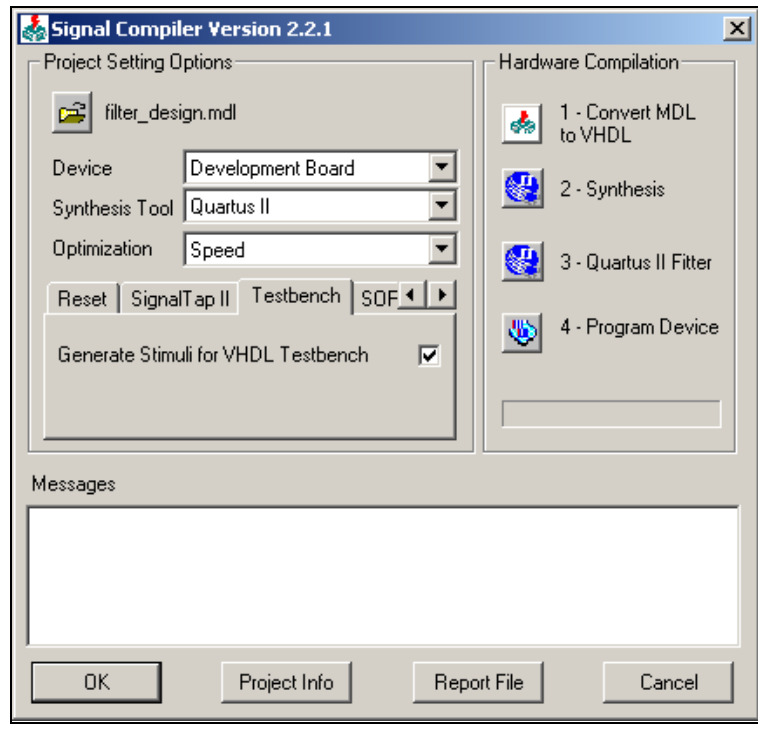
**Figure 13. SignalCompiler Block, Analyze Feature**



2. Click **Analyze**.

3. Click the **Testbench** tab as shown in [Figure 14](#).

**Figure 14. Signal Compiler Block, Hardware Compilation Feature**



4. Turn on the **Generate Stimuli for VHDL Testbench** option.
5. Under the **Hardware Compilation** section of [Figure 14](#), click **1-Convert MDL to VHDL**. The Signal Compiler generates a simulation script, **tb\_filter\_design.tcl** (.tcl is a tool command language file), and a VHDL testbench that imports the Simulink input stimuli, **tb\_filter\_design.vhd**.
6. Click **OK**.
7. Run the simulation in Simulink to generate the input stimulus files by choosing **Start** (Simulation menu).
8. Close the **filtering design** file when you are finished generating the input stimulus files.

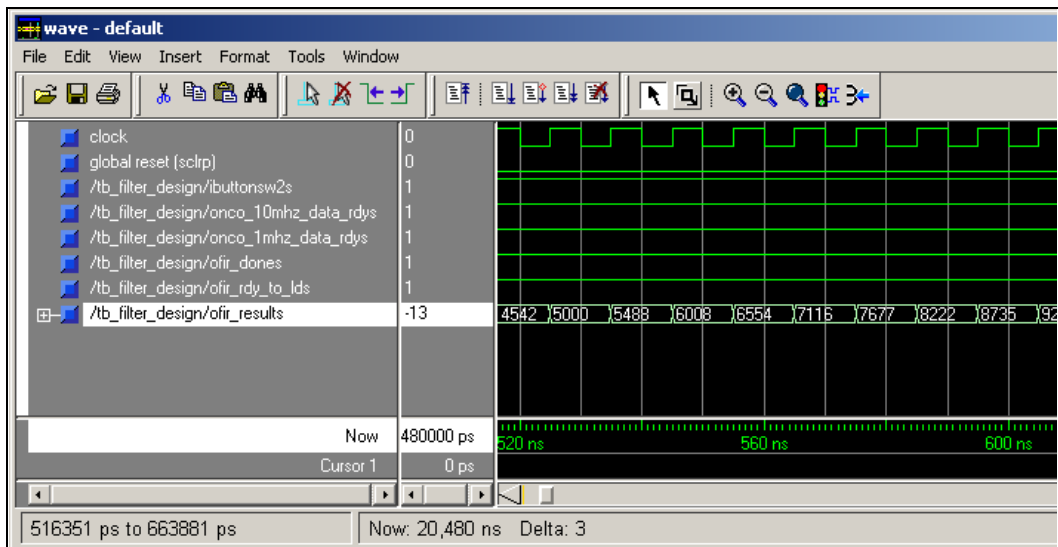
To perform RTL simulation with the ModelSim software, follow these steps:



For shorter simulation times, use the ModelSim PE or SE software.

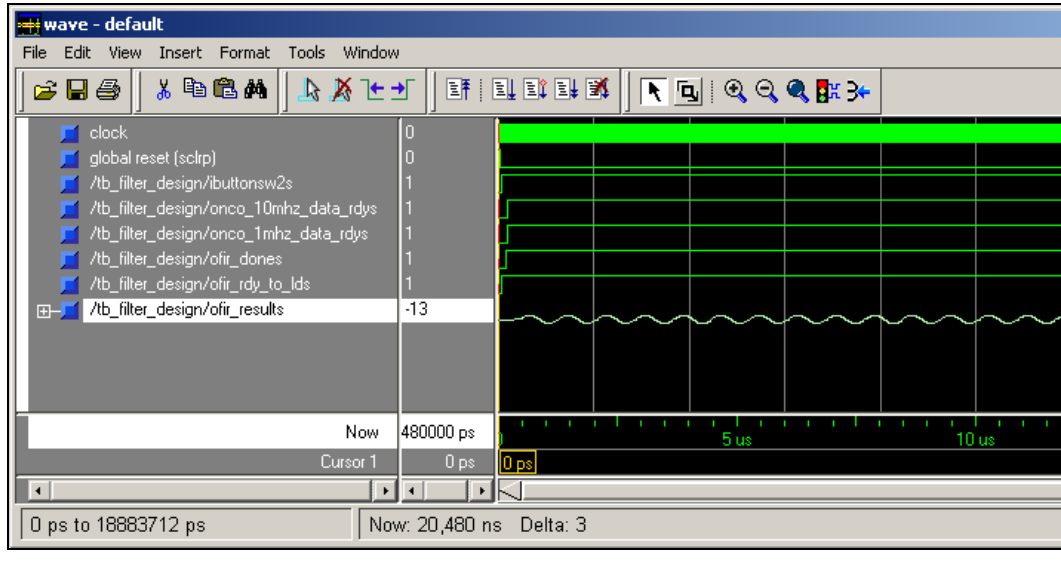
1. Start the ModelSim software.
2. Choose **Change Directory** (File menu).
3. Browse to your working directory and click **Open**.
4. Choose **Execute Macro** (Tools menu).
5. Browse for the **tb\_filter\_design.tcl** script and click **Open**.
6. The simulation results are displayed in a waveform. The ModelSim waveform editor displays the signals in decimal notation (see [Figure 15](#)) or as an analog waveform (see [Figure 16](#)).

**Figure 15. ModelSim SE Waveform Editor**



To display an analog waveform, right-click on the signal (shown in Figure 15) and select **Format > Analog**. This opens the **Wave Analog** window. Turn on **Analog Step** and click **OK**.

Figure 16. ModelSim SE Analog Waveform



## Exercise 4: Analyze the Results in Hardware

Exercise 4 includes the following actions:

- Set up the Stratix EP1S25 DSP development board or the Stratix EP1S80 DSP development board for hardware analysis.
- Review the changes made to the filtering reference design.
- Program the Stratix EP1S25 device or the Stratix EP1S80 device with the filtering reference design. You must select the correct device for the development board you set up.
- Run SignalTap II analysis in the DSP Builder to examine the filtered and unfiltered data.

### Set Up the Stratix EP1S25 DSP Development Board for Hardware Analysis

Before performing hardware analysis, you must connect two cables to the DSP board: the SMA cable and the ByteBlasterMV™ cable. The kit includes both cables. To connect the cables, follow these steps:

1. Connect the SLP-50 anti-aliasing filter to D/A2 on the board.
2. Connect the SMA cable to the SLP-50 anti-aliasing filter and A/D1 on the board.
3. Connect the ByteBlasterMV cable to your PC and to the board's 10-pin Joint Test Action Group (JTAG) header for Stratix configuration.



Align the ByteBlasterMV connector so that the red stripe is oriented towards the Altera logo on the board.

After you connect the cables, connect a jumper across jumper pins 1 and 2 on JP23 on the board (see [Figure 17](#)). The jumper settings connect the onboard 80-MHz oscillator to A/D1.

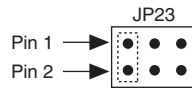


For detailed instructions on how to connect the cables to the board, see the *Stratix EP1S25 DSP Development Board Data Sheet*.



For details on installing the ByteBlasterMV driver on a PC (Windows NT, 2000, or XP) see the *ByteBlasterMV Download Cable User Guide*.

**Figure 17. Stratix EP1S25 DSP Development Board Jumper Connections for JP23**



## Set Up the Stratix EP1S80 DSP Development Board for Hardware Analysis

Before performing hardware analysis, you must connect two cables to the DSP board: the SMA cable and the ByteBlasterMV cable. The kit includes both cables. To connect the cables, follow these steps:

1. Connect the SLP-50 anti-aliasing filter to D/A2 on the board.
2. Connect the SMA cable to the SLP-50 anti-aliasing filter and A/D1 on the board.
3. Connect the ByteBlasterMV cable to your PC and to the board's 10-pin JTAG header for Stratix configuration.

After you connect the cables, connect a jumper across jumper pins 1 and 2 on JP23 on the board (see [Figure 18](#)). The jumper settings connect the onboard 80-MHz oscillator to A/D1.

Connect a jumper across pins 4 and 6 on JP26 to connect the PLL-generated clock from the Stratix EP1S80 device to D/A2 (see [Figure 18](#)).

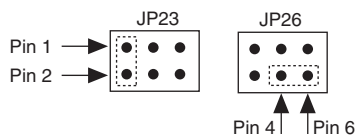


For detailed instructions on how to connect the cables to the board, see the *Stratix EP1S80 DSP Development Board Data Sheet*.



For details on installing the ByteBlasterMV driver on a PC (Windows NT, 2000, or XP) see the *ByteBlasterMV Download Cable User Guide*.

**Figure 18. Stratix EP1S80 DSP Development Board Jumper Connections for JP23 and JP26**





## Review the Changes Made to the Filtering Reference Design

To review the changes made to the filtering reference design, follow these steps:

1. Run the MATLAB software.
2. In the **Current Directory** browser, browse to one of the following directories:
  - For the Stratix EP1S25 DSP development board:
 

```
c:\altera\kits\stratix_dsp_kit-v<version>\Reference_Design\Filtering\Exercise4
```
  - For the Stratix EP1S80 DSP development board:
 

```
c:\altera\kits\stratix_dsp_pro_kit-v<version>\Reference_Design\Filtering\Exercise4
```
3. Choose **Open** (File menu) and select the file `filter_design.mdl`.
4. Review the schematic design (see [Figure 19 on page 26](#)). The figure shows the filtering reference design for the Stratix EP1S25 DSP development board. The design for the Stratix EP1S80 DSP development board is the same, except the Stratix DSP Board 1S25 Configuration block is replaced with the Stratix DSP Board 1S80 Configuration block.

The filtering reference design in Exercise 4 is the same one used in Exercises 1, 2, and 3 (see [Figure 2 on page 6](#)), except:

- The output of the adder is not directly connected to the input of the filter. The adder output is connected to a D/A converter and the filter input is connected to an A/D converter. The combined NCO-generated sinusoids are converted from D/A via the onboard D/A converters. The signal exits the board via the D/A SMA connector, loops back into the board through the A/D SMA connector, and is converted to digital by the onboard A/D converters before re-entering the Stratix device.



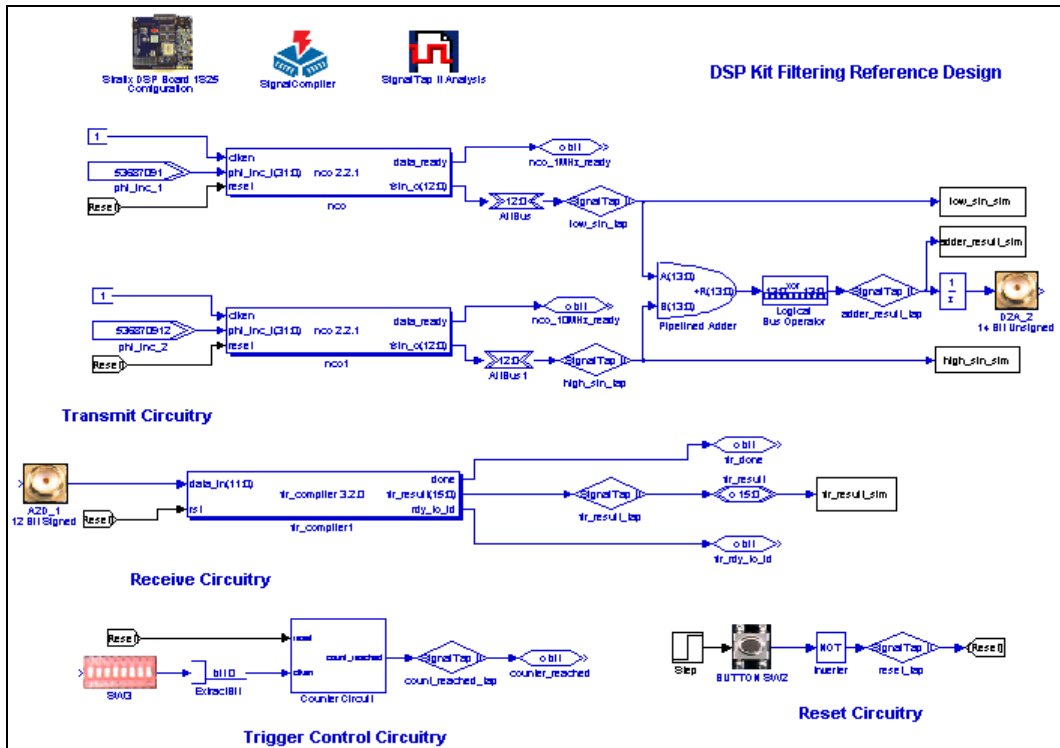
If the SMA cable is not securely connected between DAC and ADC, you do not see a signal at the output of the FIR filter during SignalTap II analysis.

- The output of the adder is fed to a bitwise XOR function. The XOR function converts the output from two's complement format to unsigned integer format by inverting the most significant bit

(MSB) to add a DC offset of  $2^{13}$ . This conversion is needed because the onboard D/A converters assume the input samples are unsigned integers.

- A register is placed after the bitwise XOR function to reduce the  $t_{CO}$  (clock to output delay) of the transmit circuitry.
- A counter circuit has been added to generate a pulse every 4,096 clock cycles after reset is asserted.

Figure 19. Simulink Design for Exercise 4 (filter\_design.mdl File)



## Program the Stratix EP1S25 or EP1S80 Device

To program the Stratix EP1S25 or EP1S80 device, follow these steps:

1. Double-click the **SignalCompiler** block as shown in [Figure 19](#).
2. Click **Analyze** as shown in [Figure 13 on page 19](#). The Signal Compiler window opens as shown in [Figure 14 on page 20](#).
3. Under the **Hardware Compilation** section of Signal Compiler window, Click **1 - Convert MDL to VHDL**. The Signal Compiler generates a Tcl script that you can use to run SignalTap II analysis in ["Perform SignalTap II Analysis"](#). The design has been precompiled. If you use the design, you can skip the synthesis and fitting steps. Go to [step 5](#).
4. If you choose to recompile the design, you must:
  - a. Turn on the **SignalTap II** option in the Signal Compiler **Project Setting Options** section of the Signal Compiler window.

Perform **Hardware Compilation**:

- b. Click **1 - Convert MDL to VHDL**
  - c. Click **2 - Synthesis**
  - d. Click **3 - Quartus II Filter**
5. Click **4 - Program Device**.
  6. Click **OK**.

## Perform SignalTap II Analysis

In `filter_design.mdl`, to specify the falling edge as the trigger condition for `count_reached_tap`, follow these steps:

1. Double-click the **SignalTap II Analysis** block. The SignalTap II Analyzer displays all of the nodes connected to SignalTap II blocks as signals to be analyzed.
2. Click `count_reached_tap` under **Signal Name**.
3. Choose **Falling Edge** in the **Trigger Condition** list.
4. Click **Change**. The condition is updated.

5. Right click on `adder_result_tap` and select **Unsigned Decimal** as the radix (see Figure 20).

**Figure 20. Specify the Radix as Unsigned for `adder_result_tap`**

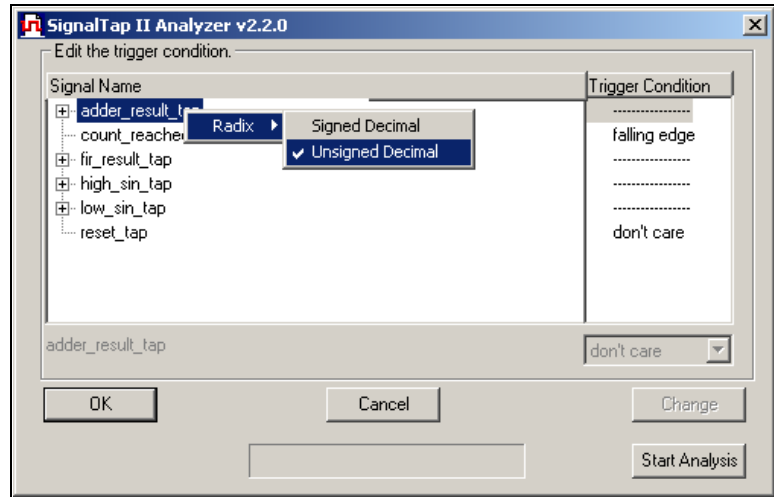


Figure 21 shows the switch locations on the EP1S25 DSP development board.

**Figure 21. SW3 on the Stratix EP1S25 DSP Development Board**

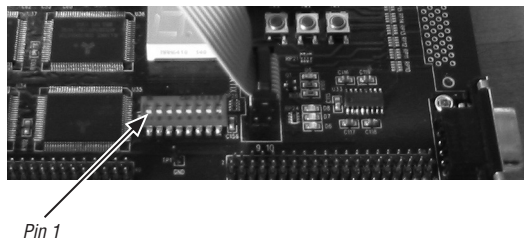
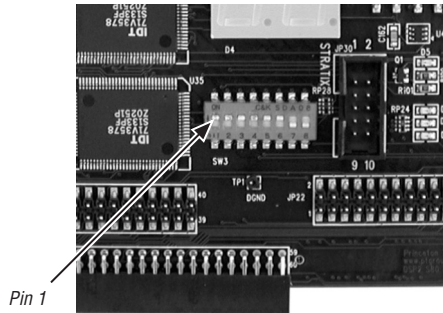


Figure 22 shows the switch locations on the EP1S80 DSP development board.

**Figure 22. SW3 on the Stratix EP1S80 DSP Development Board**



To run the analyzer and display the results in a MATLAB plot, follow these steps:

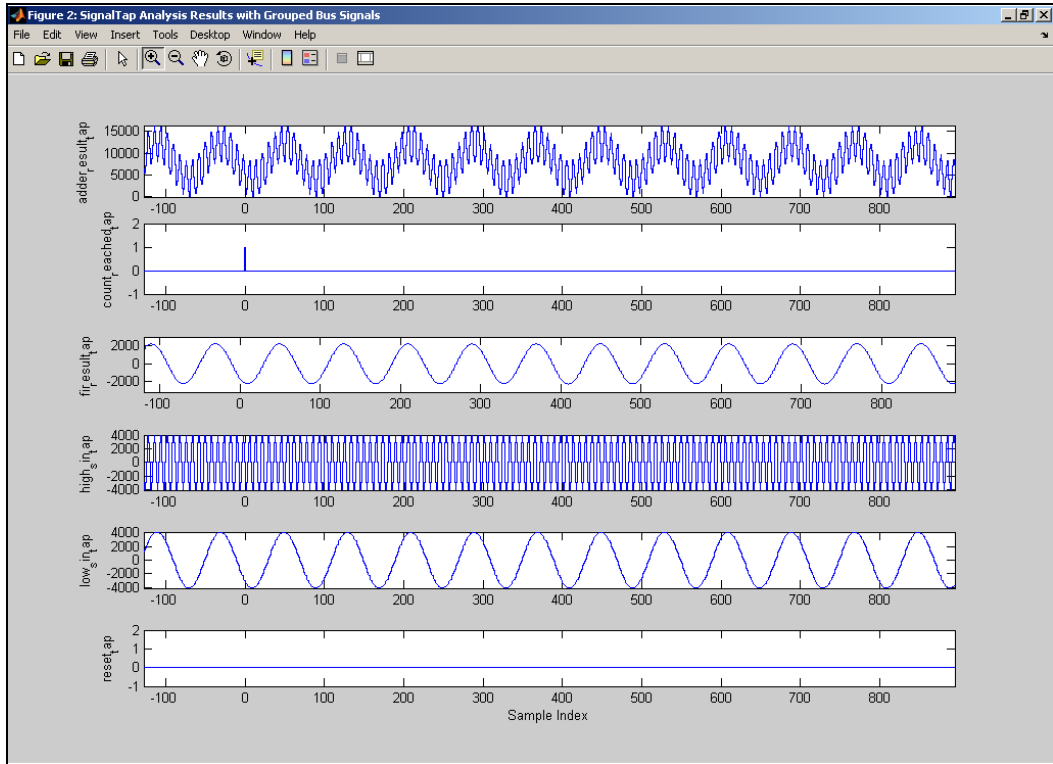
1. If you are using the Stratix EP1S25 DSP development board, turn on SW3 pin 1 on the board to enable the counter circuit as shown in Figure 21. The counter circuit `count_reached` falling edge is used as the trigger condition each 4,096 clock cycles.

If you are using the Stratix EP1S80 DSP development board, turn on SW3 pin 1 on the board to enable the counter circuit as shown in Figure 22. The counter circuit `count_reached` falling edge is used as the trigger condition each 4,096 clock cycles.

2. Click **Start Analysis**. DSP Builder runs a Tcl script to instruct the SignalTap II embedded logic analyzer to begin analyzing the data and wait for the trigger conditions to occur.
3. Click **OK** in the **SignalTap II Analysis** block when the SignalTap II logic analyzer finishes acquiring data. The **SignalTap II Analysis** block indicates that it has finished acquiring data by displaying the message "SignalTap II Analysis is complete." Two MATLAB plots display the captured data: in binary format, and in the radix you specified. The MATLAB plots display the captured data in the time domain.

4. Close the MATLAB plot of the data displayed in binary format. Examine the MATLAB plot of the data displayed in the radix you specified. Zoom in on the `fir_result_tap` signal (see Figure 23). The `fir_result_tap` signal is a scaled version of the 1-MHz sinusoid.

**Figure 23. SignalTap II Signals in the Time Domain**



5. Return to the MATLAB window.
6. In the MATLAB window, type the following command:

```
filter_design_tap_variables ←
```

This command runs a DSP Builder-generated script that reads the SignalTap II data into the MATLAB workspace.

7. To view the FFT of the filtered and unfiltered signals, type the following command in the MATLAB command window:

```
nplot_fft(adder_result_tap,'Frequency Response - Unfiltered Data',8e7) ←
```

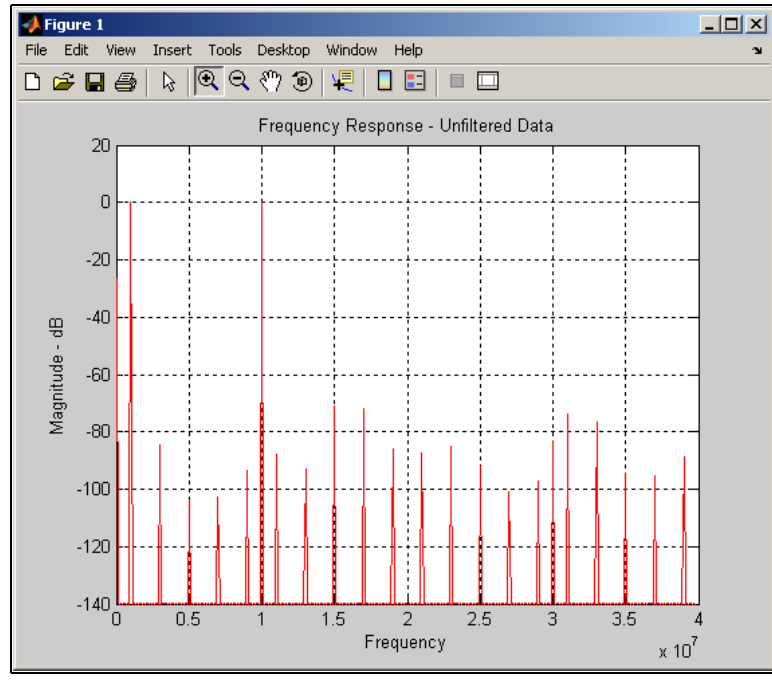
where:

- `adder_result_tap` is the name of the signal represented by the **adder\_result\_tap** SignalTap II block in the Simulink model
- `Frequency Response - Unfiltered Data` is the title of the plot
- `8e7` is the sampling frequency (80 MHz)

As previously described, the onboard D/A converters assume unsigned integer inputs, so the output of the adder is converted from signed integer to unsigned integer (see [Figure 19 on page 26](#)). The conversion adds a DC component to the FFT plot that is removed in the `nplot_fft` script (normalized `plot_fft` script). The difference in the data type format (signed integer and unsigned integer) and simulation and emulation tools (Simulink and SignalTap II) result in different background noise representation between [Figure 11 on page 17](#) and [Figure 24 on page 32](#), but the basics are similar—two peaks at 1 MHz and 10 MHz and background noise at about -80 dB or lower.

A MATLAB plot displays the frequency response of the unfiltered data (see Figure 24).

**Figure 24. FFT Response of adder\_result\_tap—Unfiltered Data**



8. To view the frequency response of the filtered data, type the following command in the MATLAB command window:

```
plot_fft(fir_result_tap, 'Frequency Response - Filtered Data', 8e7) ←
```

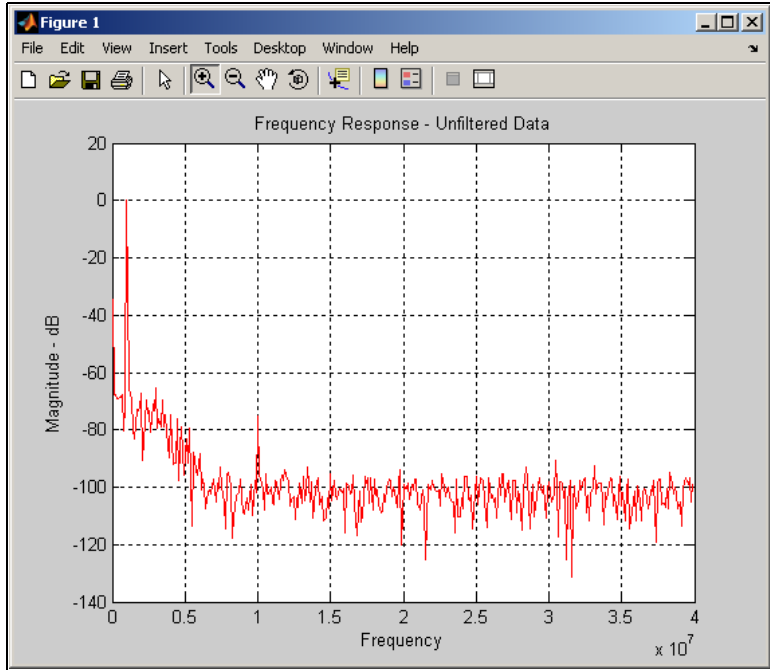
where:

- `fir_result_tap` is the name of the signal represented by the **fir\_result\_tap** SignalTap II block in the Simulink model
- `Filtered Response - Filtered Data` is the title of the plot
- `8e7` is the sampling frequency (80 MHz)



A MATLAB plot displays the frequency response of the filtered data (see [Figure 25](#)).

**Figure 25. FFT Response of fir\_result\_tap—Filtered Data**



9. Compare the filtered data plots of SignalTap II board results ([Figure 25](#)) with the Simulink simulation results ([Figure 12 on page 18](#)). Both results show a 1 MHz sine wave pass and a 10 MHz filtered sine wave.

## Troubleshooting

This section provides troubleshooting information.

### Why Do I Get Errors When I Load the Simulink filter\_design.mdl Design?

In order to load the **filter\_design.mdl** successfully, you must have the correct versions of the DSP Builder, MATLAB/Simulink, and IP cores. See [“Before You Begin” on page 4](#) for details.

### Why is My SignalTap II Filtered Signal Different From the One Figure 23 shows?

If the SMA cable is not securely connected between D/A2 and A/D1, you do not see a signal at the output of the FIR filter during SignalTap II analysis. See [Figure 23 on page 30](#).

## Conclusion

The Stratix and the Stratix Professional filtering reference designs provide a basic design example using the onboard A/D converters and D/A converters. It demonstrates SignalTap II as a real-time FPGA signal acquisition feature in the DSP Builder environment of Simulink.



101 Innovation Drive  
San Jose, CA 95134  
(408) 544-7000  
[www.altera.com](http://www.altera.com)  
Applications Hotline:  
(800) 800-EPLD  
Literature Services:  
[lit\\_req@altera.com](mailto:lit_req@altera.com)

Copyright © 2004 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.

