Introduction

The filtering reference design lab provided in the DSP Development Kit, Stratix™ Edition and the DSP Development Kit, Stratix™ Professional Edition shows you how to use the Altera® DSP Builder for system design, simulation, and board-level verification. DSP Builder is a digital signal processing development tool that interfaces The MathWorks industry-leading system-level DSP 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 also generate an RTL design and an RTL testbench from Simulink automatically. These files are pre-verified RTL output files that are 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 speed in hardware. This development flow is easy and intuitive even if you do not have extensive experience designing with programmable logic design software.

The lab uses the following items:

- Altera NCO Compiler MegaCore® function
- Altera FIR Compiler MegaCore function
- DSP Builder with the SignalTap II logic analyzer read-back feature
- ModelSim-Altera, ModelSim PE, or ModelSim SE software
- Quartus® II software version 2.2
- Stratix EP1S25 or EP1S80 DSP development board

Figure 1 shows the top-level schematic for the filtering reference design. Two numerically-controlled oscillators generate a 1-MHz sinusoidal signal and a 10-MHz sinusoidal signal. The signals are added together on-chip before they pass through a digital-to-analog converter on the Stratix DSP board. The resulting analog signal is looped back to an analog-to-digital 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 Edition CD-ROM or DSP Development Kit, Stratix Professional Edition CD-ROM, the design files are installed in the directory structure shown in Figure 2.

Figure 2. Filtering Reference Design Directory Structure

- Labs: Contains all labs for the Stratix DSP Kit.
- Filtering: Contains the filtering reference design files and documentation.
  - Doc: Contains the Stratix DSP Kit filtering reference design documentation.
  - Exercise3: Contains exercise 3.
  - Exercise4: Contains exercise 4.
  - Exercises1and2: Contains exercises 1 and 2.
This application note comprises the following exercises:

- **Exercise 1**—In this exercise, you review the filtering design using DSP Builder.
- **Exercise 2**—In this exercise, you use the MATLAB and DSP Builder software to analyze the DSP Builder-generated models of the filtering design.
- **Exercise 3**—In this exercise, you perform RTL simulation using the ModelSim-Altera simulation tool.
- **Exercise 4**—In this exercise, you configure 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 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.

**Before You Begin**

These instructions assume that you have already installed the software provided with the development kit onto your PC.

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

You must have the following software installed on your PC:

- Quartus II software version 2.2
- DSP Builder version 2.1.2 or higher
- FIR Compiler MegaCore function version 2.6.2 or higher
- NCO Compiler MegaCore function version 2.0.3 or higher
- MATLAB version 6.1 or higher
- Simulink version 4.1 or higher
- ModelSim-Altera software version 5.5e or ModelSim PE or SE software version 5.6 or higher

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

**Exercise 1: Review the Filtering Design**

To review the filtering design, perform the following steps:

1. Run the MATLAB software.

2. In the **Current Directory** browser, browse to the directory `c:\MegaCore\stratix_dsp_kit-v<version>\labs\filtering\Exercises1and2` if you are using the Stratix EP1S25 DSP development board. Browse to the directory `c:\MegaCore\stratix_dsp_pro_kit-v<version>\labs\filtering\Exercises1and2` if you are using the Stratix EP1S80 DSP development board.
3. Choose Open (File menu) and select the file filter_design.mdl.

4. Review the Simulink design. Figure 3 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 design contains a combination of OpenCore Plus DSP MegaCore functions and DSP Builder blocks. The OpenCore feature lets you test-drive Altera MegaCore functions for free using the Quartus II software. 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. However, you cannot generate device programming files.

The OpenCore Plus feature set supplements the OpenCore evaluation flow by incorporating free hardware evaluation. The OpenCore Plus hardware evaluation feature allows you to generate time-limited programming files for designs that includes Altera MegaCore functions. You can use the OpenCore Plus hardware evaluation feature to perform board-level design verification before deciding to purchase licenses for the MegaCore functions. You only need to purchase a license when you are completely satisfied with a core’s functionality and performance, and would like to take your design to production.

If you are simulating a time-limited MegaCore function using the DSP Builder and Simulink, i.e., in software, the core operation does not time out and the done pin stays low.

For more information on OpenCore Plus hardware evaluation, see AN 176: OpenCore Plus Hardware Evaluation of MegaCore Functions.
5. Double-click the **Time Out Circuit** block to open the Time Out Circuit subsystem (see Figure 4). The OpenCore Plus timed out signals of the NCOs and the FIR Filter are fed into a NOR gate whose output is connected to the `system_timed_out` output. The `system_timed_out` signal is connected to an on-board LED that illuminates when one of the MegaCore functions time out.
6. Double-click the NCO_1MHz block to launch the NCO Compiler MegaWizard® Plug-In. The NCO_1MHz block generates a 1-MHz sinusoidal signal (see Figure 5).

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 the MegaWizard Plug-In.
7. Click **Cancel** when you are finished reviewing the parameter settings.

8. Perform the same action on the **NCO_10MHz** block. See Figure 6. Table 2 shows the parameters that you can set in the MegaWizard Plug-In.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Phase accumulator Width</td>
<td>32 bits</td>
</tr>
<tr>
<td>Sin/Cos Generator Width</td>
<td>12 bits</td>
</tr>
<tr>
<td>Output Width</td>
<td>13 bits</td>
</tr>
<tr>
<td>Generation Algorithm</td>
<td>Multiplier-based</td>
</tr>
<tr>
<td>Architecture</td>
<td>Use dedicated multipliers</td>
</tr>
<tr>
<td>Selected Outputs</td>
<td>Sin only</td>
</tr>
<tr>
<td>Implement Phase Dithering</td>
<td>Yes</td>
</tr>
<tr>
<td>Dither Level</td>
<td>5</td>
</tr>
<tr>
<td>Clock Rate</td>
<td>80 MHz</td>
</tr>
<tr>
<td>Desired Output Frequency</td>
<td>1 MHz</td>
</tr>
</tbody>
</table>

Table 2. NCO Compiler Parameters

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. The constant value fed into the phase increment input of the NCO block 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.
output frequency in the wizard. As shown in Figure 5, the calculated result for a 1-MHz sine wave is 53,687,091. The chosen clock frequency corresponds to the 80-MHz oscillator on the Stratix DSP development board. Similarly, as shown in Figure 6, the desired output frequency of 10 MHz yields a phase increment value of 536,870,912.

9. Click Cancel when you are finished reviewing the parameter settings.

Figure 6. Calculate the Phase Increment Value in the NCO Wizard

10. Double-click the FIR filter block to launch the FIR Compiler MegaWizard Plug-In (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. Table 3 shows the parameters that you can set in the MegaWizard Plug-In.
11. Click **Next** until you have finished reviewing the parameter settings.

### Table 3. FIR Compiler Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Filter Type</td>
<td>Low Pass</td>
</tr>
<tr>
<td>Window Type</td>
<td>Blackman</td>
</tr>
<tr>
<td>Sample Rate</td>
<td>8e7 Hz</td>
</tr>
<tr>
<td>Number of Coefficients</td>
<td>35</td>
</tr>
<tr>
<td>Cutoff Frequency</td>
<td>3e6 Hz</td>
</tr>
<tr>
<td>Multi-rate Filter Settings</td>
<td>Single Rate</td>
</tr>
<tr>
<td>Coefficient Width</td>
<td>14</td>
</tr>
<tr>
<td>Input Specification</td>
<td>12, Signed (A/D width)</td>
</tr>
<tr>
<td>Precision</td>
<td>Limited</td>
</tr>
<tr>
<td>Round</td>
<td>12 bits</td>
</tr>
<tr>
<td>Architecture</td>
<td>Parallel</td>
</tr>
<tr>
<td>Device Family</td>
<td>Stratix</td>
</tr>
<tr>
<td>Number of Clock Cycles to Compute Result</td>
<td>1</td>
</tr>
</tbody>
</table>
12. Click **Cancel** after you have finished reviewing the parameter settings.

**Exercise 2: Simulate the Model in Simulink**

To simulate the model in the Simulink software, perform the following steps:

1. Start the simulation by choosing **Start** (Simulation menu).

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

3. Click the binocular icon to auto-scale the waveforms. Figures 8 and 9 show the scaled waveforms in the time domain.

**Figure 8. Time Domain Plot of adder_result_sim - Unfiltered Data**

**Figure 9. Time Domain Plot of fir_result_sim - Filtered Data**

4. Switch to the MATLAB window.
5. 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:

```matlab
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 10).

b. To view the frequency response of the filtered data, type the following command:

```matlab
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 11).
Figure 10. FFT Response of adder_result_sim - Unfiltered Data

Figure 11. FFT Response of fir_result_sim - Filtered Data
Exercise 3: Perform RTL Simulation

To generate the simulation files for the filtering design example, perform the following steps:

1. Run the MATLAB software.

2. In the Current Directory browser, browse to the directory `c:\MegaCore\stratix_dsp_kit-v<version>\labs\filtering\Exercise3` if you are using the Stratix EP1S25 DSP development board. Browse to the directory `c:\MegaCore\stratix_dsp_pro_kit-v<version>\labs\filtering\Exercise3` if you are using the Stratix EP1S80 DSP development board.

3. Choose Open (File menu) and select the file `filter_design.mdl`. For simulation purposes, the filtering design uses the OpenCore versions of the cores, not the OpenCore Plus versions. The OpenCore feature is similar to the OpenCore Plus feature, except you cannot generate device programming files to test the cores in hardware until you purchase a license. For more information on the OpenCore and OpenCore Plus features, refer to step 4 of “Exercise 1: Review the Filtering Design” on page 3.

4. Review the schematic design. Figure 12 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 design used in this exercise is the same design that is used in Exercises 1 and 2 with the following exceptions:
   
   - The SignalTap II nodes are removed.
   - The OpenCore Plus versions of the FIR Filter and NCO MegaCore functions are replaced with the OpenCore versions.
   - The time out circuit is removed.
5. Double-click the SignalCompiler block in your model.

6. Click the **Testbench** tab.

7. Turn on the **Generate Stimuli for VHDL Testbench** option.

8. Click **Convert MDL to VHDL**. SignalCompiler generates a simulation script, `tb_filter_design.tcl`, and a VHDL testbench that imports the Simulink input stimuli, `tb_filter_design.vhd`.

9. Click **OK**.

10. Run the simulation in Simulink to generate the input stimulus files by choosing **Start** (Simulation menu).

11. Close the filtering design file when you are finished generating the input stimulus files.
To perform RTL simulation with the ModelSim software, perform the following steps:

1. Start the ModelSim software.
2. Choose Change Directory (File menu).
3. Browse to your working directory and click Open.
4. Choose Execute Macro (Macro menu).
5. Browse for the tb_filter_design.tcl script and click Open.
   
   For shorter simulation times, use the ModelSim PE or SE software version 5.6.
6. The simulation results are displayed in a waveform. The ModelSim waveform editor displays the signals in decimal notation, see Figure 13. Alternatively, you can view the signals as an analog waveform, see Figure 14.

**Figure 13. ModelSim Waveform Editor**

![ModelSim Waveform Editor](image)

**Figure 14. ModelSim Analog Waveform**

![ModelSim Analog Waveform](image)
Exercise 4: Analyze the Results in Hardware

Exercise 4 covers the following actions:

1. Set up the board for hardware analysis.
2. Review the changes made to the filtering design.
3. Configure the Stratix device on the Stratix EP1S25 or EP1S80 DSP development board with the filtering design.
4. 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. These cables are included with the kit. To connect the cables, perform the following 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 ByteBlaster II 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 15). The jumper settings connect the on-board 80-MHz oscillator to A/D1.

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

Figure 15. Jumper Connection for On-Board Oscillator
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 ByteBlaster II cable. These cables are included with the kit. To connect the cables, perform the following 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 ByteBlaster II 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 15 on page 16). The jumper settings connect the on-board 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 16).

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

Figure 16. Stratix EP1S80 DSP Development Board Jumper Connection
Review the Changes Made to the Filtering Reference Design

To review the changes made to the filtering reference design, perform the following steps:

1. Run the MATLAB software.

2. In the Current Directory browser, browse to the directory \c:\MegaCore\stratix_dsp_kit-v<version>\labs\filtering\Exercise4 if you are using the Stratix EP1S25 DSP development board. Browse to the directory \c:\MegaCore\stratix_dsp_pro_kit-v<version>\labs\filtering\Exercises4 if you are using the Stratix EP1S80 DSP development board.

3. Choose Open (File menu) and select the file filter_design.mdl.

4. Review the schematic design. Figure 17 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 design in Exercise 4 uses the time-limited OpenCore Plus versions of the MegaCore functions. The design is the same one used in Exercises 1 and 2 (see Figure 3), 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 digital to analog via the on-board 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 on-board A/D converters before re-entering the Stratix device.

   If the SMA cable is not securely connected between D/A 2 and A/D 1, you will 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 to add a DC offset of $2^{13}$. This conversion is needed because the on-board D/A converters assume the input samples are unsigned integers.
A register is placed after the bitwise XOR function to reduce the t\textsubscript{CO} (clock to output delay) of the transmit circuitry.

A counter circuit has been added to generate a pulse every 4,095 clock cycles after reset is asserted (see step 5).

---

**Figure 17. Simulink Design for Exercise 4**

5. Double-click the **Counter Circuit** Block to view the counter circuit subsystem (see Figure 18). When the clken input signal is high, the counter circuit generates a signal countReached that generates a pulse every 4,095 clock cycles. In “Run SignalTap II Analysis” on page 20, the falling edge of the signal countReached is set as a trigger in the SignalTap II Analyzer block. See “Run SignalTap II Analysis” on page 20 for more details on how the counter circuit is
used. The minimum 4,095 clock cycle delay ensures that the data is stable on the output of the on-board anti-aliasing filter, which is connected to the D/A converter, before the SignalTap II Analyzer begins to capture data.

**Figure 18. Counter Circuit**

Counter Circuit

After reset is asserted, the output count_reached generates a pulse every 4095 clock cycles.

---

**Configure the Stratix Device**

To configure the Stratix device, perform the following steps:

1. Double-click the **SignalCompiler** block.

2. Click **1- Convert MDL to VHDL**. SignalCompiler generates a Tcl script that you can use to run SignalTap II analysis in “Run SignalTap II Analysis” on page 20.

3. Optional. Click **2 - Synthesis** to synthesize the design. To save synthesis time, skip this step and go to step 5, configuring the device with the programming file provided with the development kit.

4. Optional. Click **3 - Quartus II Fitter** to compile the design. You can skip this step and go to step 5, configuring the device with the programming file provided with the development kit.

5. Click **4 - Program DSP Board**. SignalCompiler takes approximately one minute to configure the device.

**Run SignalTap II Analysis**

In **filter_design.mdl**, to specify falling edge as the trigger condition for **counter_reached_tap**, perform the following steps:

1. Double-click the **SignalTap II Analyzer** block. The SignalTap II Analyzer displays all of the nodes connected to SignalTap II blocks as signals to be analyzed.
2. Click on 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 as the radix (see Figure 19).

---

**Figure 19. Specify the Radix as Unsigned for adder_result_tap**

To run the analyzer and display the results in a MATLAB plot, perform the following steps:

1. If you are using the Stratix EP1S25 DSP development board, turn on SW3 pin 1 on the board to disable the counter circuit as shown in Figure 20.

---

**Figure 20. SW3 on the Stratix EP1S25 DSP Development Board**
If you are using the Stratix EP1S80 DSP development board, turn on SW3 pin 1 on the board to disable the counter circuit as shown in Figure 21.

**Figure 21. SW3 on the Stratix EP1S80 DSP Development Board**

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. Press SW2 on the Stratix DSP development board to generate a pulse on the reset input signal.

4. Turn off SW3 pin 1 on the board to assert clken and to enable the counter circuit. Setting the clken input signal high after generating a pulse on the reset input signal ensures that the trigger condition, the first falling edge of count_reached, occurs no sooner than 4,095 clock cycles after the design has been reset. This minimum delay requirement of 4,095 clock cycles allows the data at the output of the anti-aliasing filter sufficient time to stabilize before the SignalTap II logic analyzer begins acquiring data.

5. 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.
6. 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 22). The fir_result_tap signal is a scaled version of the 1-MHz sinusoid.

7. Return to the MATLAB window.

8. In the MATLAB window, type the following command, which runs a DSP Builder-generated script that reads the SignalTap data into the MATLAB workspace:

   filter_design_tap_variables
9. To view the FFT of the filtered and unfiltered signals, type the following command in the MATLAB Command Window:

```matlab
plot_fft(addzer_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)

A MATLAB plot displays the frequency response of the unfiltered data (see Figure 23).

![Figure 23. FFT Response of adder_result_tap—Unfiltered Data](image)

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

```matlab
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 24).

Figure 24. FFT Response of `fir_result_tap`—Filtered Data

11. Compare the plots generated in steps 9 and 10 with the plot generated in step 5 of “Exercise 2: Simulate the Model in Simulink” (page 11). The hardware results match the Simulink simulation results, with the exception of the impulse at frequency 0 in the plot of the unfiltered data. The impulse at frequency 0 occurs as a result of the DC offset added to the output of the adder (see the second bullet in step 4 of “Review the Changes Made to the Filtering Reference Design” on page 18 for more details).