Introduction

Much of the signal processing performed in modern wireless communications systems takes place in the digital domain. Given the increasing processing demands, the parallel processing capabilities of Altera® programmable logic devices makes them an attractive technology for baseband/IF digital signal processing (DSP) applications.

The Altera Stratix™ device family—with its dedicated DSP blocks, TriMatrix™ memory, and on-chip PLLs—is particularly well-suited for the demands of communications signal processing functions. In addition, by combining the use of Altera parameterizable DSP cores and Stratix devices, complex high performance DSP designs can be implemented in a relatively short period of time. For digital modulator/demodulator applications, Altera provides a quadrature phase-shift keying (QPSK) modem reference design as an example design.

The QPSK modem reference design demonstrates key features of the DSP Development Kit, Stratix Edition and DSP Development Kit, Stratix Professional Edition. The QPSK modem reference design shows by example how to use the DSP Builder for system modeling, simulation, and hardware verification on the Stratix EP1S25 DSP development board and EP1S80 DSP development board. The QPSK modem reference design is implemented using a combination of Altera intellectual property DSP megafuntions, the Altera library of parameterizable modules (LPMs), and custom logic. The targeted device on the Stratix EP1S25 DSP development board, is the Altera EP1S25F780C5; the targeted device on the Stratix EP1S80 DSP development board, is the Altera EP1S180B956C6.

For more information on the Stratix EP1S25 DSP development board, refer to the Stratix EP1S25 DSP Development Board Data Sheet.

For more information on the Stratix EP1S80 DSP development board, refer to the Stratix EP1S80 DSP Development Board Data Sheet.
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.

The automatically generated HDL files are at the register transfer level (RTL). They are optimized for use in the Quartus II software for rapid prototyping. Using DSP Builder, you can also automatically generate from Simulink a testbench that you can use to simulate the HDL files. 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 programmable logic.

The QPSK modem reference design is a basic design that uses a simplified channel model. It implements an encoder, a channel, and a decoder, and uses modulation, filtering, and error correction. It does not implement any synchronization functions. The parameters used in the design are arbitrary, and do not conform to any particular standard.

The design uses the following items:

- Stratix device DSP blocks
- Altera® Viterbi Compiler MegaCore® function
- Altera Reed-Solomon (RS) Compiler MegaCore function
- Altera numerically controlled oscillator (NCO) Compiler MegaCore function
- Altera FIR Compiler MegaCore function
- DSP Builder with the SignalTap® II logic analyzer read-back feature
- ModelSim PE or SE simulator
- Quartus® II software version 3.0
- MATLAB version 6.5 and Simulink version 5.1, including the DSP and communications blockset
- Stratix EP1S25 DSP development board or the Stratix EP1S80 DSP development board

When you install the software from the DSP Development Kit, Stratix Edition CD-ROM, the design files are installed in the directory structure shown in Figure 1.
This application note comprises the following sections:

- **Functional Description**—This section describes the QPSK modem reference design in detail
- **Exercise 1**—In this exercise, you review the QPSK modem reference design using DSP Builder.
- **Exercise 2**—In this exercise, you use the MATLAB and DSP Builder software to analyze the Altera-generated models of the QPSK modem reference 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 QPSK modem reference design and use the SignalTap II read-back feature in the DSP Builder to capture data from internal Stratix device nodes. SignalTap II readback occurs while the system under test 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.

**Features**

The QPSK modem reference design provides the following features:

- Single data channel at 5 Mbps
- Concatenated RS/Viterbi forward error correction
- 110 dB SFDR 20 MHz carrier
- 80 MSPS transmitter output with 10 MHz bandwidth and 20 dB out-of-band rejection
Related Links
- Third Generation Partnership Project, www.3gpp.org

Functional Description

Figure 2 shows a simple block diagram for the QPSK modem reference design.

**Data Generation**

A linear feedback shift register (LFSR) generates the data in a single-bit data stream. The data rate is 5 Mbps. The LFSR implements a primitive polynomial, which produces a cyclic pattern of data. The primitive polynomial is $x^{14} + x + 1$.

**Forward Error Correction (Transmitter)**

The transmitter forward error correction (FEC) block comprises a RS encoder and a convolutional encoder before the data is modulated. Figure 3 shows a simple block diagram of the FEC transmit block.

For a more detailed diagram, view the design in Simulink, see “Exercise 1: Review the QPSK Modem Design” on page 9.

The serial data at 5 Mbps is converted to parallel data and is encoded with the RS encoder.
For more information on the RS Compiler, refer to the Reed-Solomon Compiler User Guide.

Table 1 shows the convolutional encoder parameter values.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Constraint length</td>
<td>5</td>
</tr>
<tr>
<td>Polynomial</td>
<td>GA = 19; GB = 29</td>
</tr>
</tbody>
</table>

For more information on convolutional encoding, refer to the Viterbi Compiler User Guide.

Modulation & Filtering

The QPSK modem implements QPSK modulation. I and Q receive the first and second parity symbols from the convolutional encoder. These I and Q signals pass through a root-raised cosine (RRC) filter, which is implemented with a FIR function using the FIR Compiler. There is a matching RRC filter at the receiver end. The QPSK modem has an 80-MHz clock and the modulation is carried out with a sinusoid of 20 MHz implemented using the NCO. Figure 4 and Table 2 show the QPSK mapping.

Figure 4. QPSK Mapping

![QPSK Mapping Diagram]
Table 2. QPSK Mapping

<table>
<thead>
<tr>
<th>Logical Symbol</th>
<th>Channel Symbol</th>
<th>Note (1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>Y</td>
<td>I</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1 (01)</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0 (00)</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0 (00)</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>–1 (11)</td>
</tr>
</tbody>
</table>

*Note to Table 2:*
1. The number in parentheses is the two's complement binary representation of I or Q.

Figure 5 shows the QPSK modem modulation and filtering.

For a more detailed diagram, view the design in Simulink, see “Exercise 1: Review the QPSK Modem Design” on page 9.

The detailed diagram shows the control logic that controls the data input.

**Figure 5. Modulation & Filtering**

Channel

The reference design includes a null channel model that is simply a wire delay. There is also the option to physically insert errors into the channel, which is otherwise error-free, through a separate port. For simulation purposes, there is an additive white gaussian noise (AWGN) channel model from the Simulink communications blockset to verify the robustness of the modem.
For a more realistic channel, you can modify the design.

Demodulation

The demodulator multiplies the received signal with the same carrier frequency as in the transmitter. The carrier frequency is generated by an NCO. The QPSK modem filters the I and Q components with a matching RRC filter, which is implemented with a FIR function using the FIR Compiler. An integrator retrieves the transmitted symbols, which are then demapped by the QPSK demapper. The QPSK modem reference design does not implement synchronization, which is required in any real receiver—the modulator and the demodulator are artificially locked together. Figure 6 shows the demodulator.

For a more detailed diagram, view the design in Simulink, see “Exercise 1: Review the QPSK Modem Design” on page 9.

Figure 6. Demodulator

FEC (Receiver)

The receiver forward error correction (FEC) block comprises a parallel continuous Viterbi decoder and an RS decoder after the channel demodulator. Figure 7 shows a simple block diagram of the FEC receive block.

For a more detailed diagram, view the design in Simulink, see “Exercise 1: Review the QPSK Modem Design” on page 9.

The detailed diagram shows the testbench logic that compares the RS decoder output to the RS encoder data input.
The receive FEC block converts the data back to serial data for decoding.

For more information on the Viterbi Compiler, refer to the *Viterbi Compiler User Guide*.

The RS decoder parameter values are the same as the RS encoder parameter values (see Table 3).

For more information on the RS Compiler, refer to the *Reed-Solomon Compiler User Guide*.

### Before You Begin

These instructions assume that you have already installed the software provided with the DSP Development Kit, Stratix Edition onto your PC.

You must have the following software installed on your PC:

- Quartus II software version 2.2 (limited edition or a purchased version)
- DSP Builder version 2.1.3 or higher
- Viterbi Compiler MegaCore function version 3.2.2
- Reed-Solomon Compiler MegaCore function version 3.3.5
- NCO Compiler MegaCore function version 2.0.5
- FIR Compiler MegaCore function version 2.7
- MATLAB version 6.5 or higher
- Simulink version 5.0 or higher
  - DSP Blockset version 5.0 or higher
  - Communications Blockset version 2.5 or higher
- ModelSim PE or SE version 5.6 or higher

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

You must uninstall all other versions of the Viterbi Compiler MegaCore function and the RS MegaCore function, before you install the versions that are provided with the kit.

For more installation information, refer to the *DSP Development Kit Stratix and Stratix Professional Edition Getting Started User Guide*. 
Exercise 1: Review the QPSK Modem Design

To review the QPSK modem design, perform the following steps:

1. Run the MATLAB software.

2. If you are targeting the Stratix EP1S25 DSP development board, browse to the directory \`\`c:\MegaCore\Stratix_DSP_kit-<version>\ref_designs\qpsk_modem_ref\projects\target_stratix_ocp\`\` in the Current Directory browser.

   If you are targeting the Stratix EP1S80 DSP development board, browse to the directory \`\`c:\MegaCore\Stratix_DSP_Pro_kit-<version>\ref_designs\qpsk_modem_ref\projects\target_stratix_ocp\`\` in the Current Directory browser.

3. Choose Open (File menu), and select the file `modem_ref_demo_ocp.mdl`.

4. Review the schematic design (see Figure 8). The QPSK modem design file 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 `fec_tx` block.

6. Double-click the `rs_enc_tx_ocp` block to launch the RS Encoder MegaWizard Plug-In (see Figure 9).
7. Click Next to see the RS Encoder parameters (see Figure 10).

Figure 9. RS Encoder MegaWizard Plug-In

Figure 10. RS Encoder Parameters
Table 3 shows the RS encoder parameter values, which are widely used in DVB standards.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of bits per symbol</td>
<td>8</td>
</tr>
<tr>
<td>Number of symbols per codeword</td>
<td>204</td>
</tr>
<tr>
<td>Number of check symbols per codeword</td>
<td>16</td>
</tr>
<tr>
<td>Field polynomial</td>
<td>285</td>
</tr>
<tr>
<td>First root of generator polynomial</td>
<td>0</td>
</tr>
<tr>
<td>Root spacing in generator polynomial</td>
<td>1</td>
</tr>
</tbody>
</table>

8. Click **Cancel** after you have finished reviewing the parameter settings.

9. On the original Simulink schematic, **modem_ref**, double-click the **modem_tx** block. To review the parameter settings for the blocks, double-click the **mod_tx** block.

   For more information on the modulator block, see “Modulation & Filtering” on page 5.

10. On the **modem_tx/mod_tx** schematic, double-click the **rrc_I_tx_ocp** block to launch the FIR Compiler MegaWizard Plug-In (see **Figure 11**). The FIR filter block is a 60-tap, RRC filter with cut-off frequency of 5 MHz, assuming an 80-MSPS sample rate.
Table 4 shows the RRC filter parameter values.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Filter type</td>
<td>Root Raised Cosine</td>
</tr>
<tr>
<td>Window type</td>
<td>Hanning</td>
</tr>
<tr>
<td>Sample rate</td>
<td>8e+007 Hz</td>
</tr>
<tr>
<td># of coef</td>
<td>60</td>
</tr>
<tr>
<td>Cutoff freq (1)</td>
<td>5e+006 Hz</td>
</tr>
<tr>
<td>Coefficient bit width</td>
<td>11</td>
</tr>
<tr>
<td>Signed (two’s complement) binary data bit width</td>
<td>2</td>
</tr>
<tr>
<td>Number of input channels</td>
<td>1</td>
</tr>
<tr>
<td>Architecture options</td>
<td>fixed coefficient fully parallel filter</td>
</tr>
<tr>
<td>Pipeline level</td>
<td>1</td>
</tr>
</tbody>
</table>
11. Click **Next** until you have finished reviewing the parameter settings.

12. Click **Cancel** after you have finished reviewing the parameter settings.

13. On `modem_tx/mod_tx` schematic, double-click the `nco_ocp` block to launch the NCO Compiler MegaWizard Plug-In (see Figure 12). The `nco_ocp` block generates a 20-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.

---

**Figure 12. NCO Parameters**

![NCO Parameters](image-url)
14. Click **Cancel** after you have finished reviewing the parameter settings.

15. On the original Simulink schematic, **modem_ref_demo.ocp**, double-click the **fec_rx** block.

16. Double-click the **vit_dec_rx_ocp** block to launch the Viterbi Decoder MegaWizard Plug-In (see **Figure 13**).

---

**Figure 13. Viterbi Decoder Parameters**

![Viterbi Decoder Parameters](image_url)
Table 5 shows the parallel continuous Viterbi decoder parameter values.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Architecture</td>
<td>Parallel continuous</td>
</tr>
<tr>
<td>Constraint length</td>
<td>5</td>
</tr>
<tr>
<td>Softbits</td>
<td>4</td>
</tr>
<tr>
<td>Traceback</td>
<td>30</td>
</tr>
<tr>
<td>Number of coded bits</td>
<td>2</td>
</tr>
<tr>
<td>Polynomials</td>
<td>GA = 19; GB = 29</td>
</tr>
<tr>
<td>Puncturing</td>
<td>No puncturing</td>
</tr>
<tr>
<td>Coefficient bit width</td>
<td>11</td>
</tr>
<tr>
<td>Signed (two’s complement)</td>
<td>2</td>
</tr>
</tbody>
</table>

17. Click **Next** until you have finished reviewing the parameter settings.

18. Click **Finish** after you have finished reviewing the parameter settings.

Unlike the other cores in this exercise you must click **Finish**, rather than **Cancel**, after reviewing the Viterbi decoder.

19. Double-click the **rs_dec_rx_ocp** block, to launch the RS Decoder MegaWizard Plug-In (see **Figure 14**).
20. Click **Next** until you have finished reviewing the parameter settings.

21. Click **Cancel** after you have finished reviewing the parameter settings.

22. Close all of the MegaWizard Plug-Ins and all of the Simulink diagrams, except `modem_ref_demo_ocp.mdl`.

Exercise 2: Simulate the Model in Simulink

Before you simulate the model in Simulink, observe the setting of the three manual switches.

1. There is a manual switch feeding the channel block through the `noise_source[ ]` port.

   - Turn the switch on to set the AWGN block as the channel model.
   - Turn the switch off to set the user selectable error rate channel, whereby the error bits are injected manually through the `error_sw1` push-button switch. For the purpose of the Simulink simulation, the step function is used as a stimulus.

2. There is a manual switch feeding the `switch_plot` block through the `plot_signal` port. Turn the switch on to bring up the plot of the input and output signal of the modem in both time and frequency domain at the end of the simulation.

3. There is a manual switch feeding the `switch_plot` block through the `plot_mod_frequency` port. Turn the switch on to bring up the plot of the signal before and after the `mod_tx` block in the transmitter block.
To review or change the parameters of the AWGN block, perform the following steps:

1. On the original Simulink schematic, **modem_ref_demo_ocp**, double-click the **channel** block.

2. Double-click the **awgn_bb** block.

   The AWGN block is used for Simulink simulation purposes only. It is treated as a black-box during the VHDL conversion. In the VHDL domain, it is substituted with a simple wire design **awgn_bb.vhd**.

3. On the **channel/awgn_bb** schematic, double-click on the **AWGN Channel** block to bring up the block parameter window (see **Figure 15**).

4. Click **Cancel** after you have finished reviewing the parameter settings. If you want to change and save the new parameters settings, click **OK**.

To simulate the model in the Simulink software, perform the following steps:
1. Double-click the Signal Compiler block.

2. Check the Update Diagram box.

3. Click Analyze.

4. In the Signal Compiler window, under Project Settings Options, click the folder icon and browse to the relevant \ref_designs\qpsk_modem_ref\projects\target_stratix_ocp directory and select the modem_ref_demo_ocp.mdl file.

5. Click OK.

6. Choose Simulation Parameters (Simulation menu).

7. Type 150000 in the Stop Time box to display 150,000 samples.

8. Start the simulation by choosing Start (Simulation menu).

During simulation, an XY plot displays (see Figure 16) the results of the constellation demapper in the modem_rx block. The results depend on the channel mode that you choose. There are also BER calculation blocks in the top-level block. The BER results counter (top-level) that compares the RS encoder and Decoder starts after 2,635 samples due to latency in the signal generation. The BER results counter (top-level) that compares the QPSK modem input and output starts after 80,971 samples due to latency in the block decoder.

**Figure 16. Demapper XY Plot**
1. In the lower-level schematics, double-click any scope block to view the signals in the time domain.

   ![You can add scope blocks to the top-level schematic.]

2. Click the binocular icon to auto-scale the waveforms. Figure 17 shows the data from scope 8 (modem_tx block) before and after QPSK mapping. Figure 18 shows the data from scope 6 (modem_rx block) before and after QPSK demapping.

---

**Figure 17. Before and After QPSK Mapping—Scope 1**

![Scope 1](image)
3. Switch to the MATLAB window.

The following list describes the QPSK modem design scope outputs:

- Scope 1 (fec_rx block) monitors the `numerr` and `decfail` outputs from the RS decoder
- Scope 3 (modem_rx/demod_rx block) shows the filtered output for I and Q at the demodulator before non-coherent integration of the symbols
Scope 6 (modem_rx block) shows the channel symbols at the receiver before QPSK de-mapping and after de-mapping

Scope 1 (top-level) shows data at the input of the RS decoder and compares it to the data at the output of the RS encoder. It also indicates if both data are equal or not

Scope 2 (top-level) shows data at the input and output of the modem. It also indicates if both data are equal or not

Scope 1 (modem_tx block) shows the logical bits for I and Q channels before QPSK mapping and the channel symbols after mapping

This exercise and Scope 1 (top-level) allow you to compare the input of the RS decoder with the output of the RS encoder. To compare the QPSK modem input data with the output data, you need to simulate for at least 85,000 samples.

The following list describes the QPSK modem design simulation plots (see Figure 19):

- Figure 1 shows the input and output of the mapper in the transmitter block
- Figure 2 shows the input and output of the demapper in the receiver block
- Figure 3 shows the input data (transmitter block) in the time and frequency domain
- Figure 4 shows the output data (receiver block) in the time and frequency domain
- Figure 5 shows the combined input data into the transmitter block and output data from the receiver block
- Figure 6 shows the input data (transmitter block) before and after modulation in frequency domain

Figure 19 shows the QPSK modem simulation results of using an AWGN channel with 60 E_s/No dB.
Exercise 3: Perform RTL Simulation

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

1. Run the MATLAB software.
2. If you are targeting the Stratix EP1S25 DSP development board, browse to the directory c:\MegaCore\Stratix_DSP_kit-<version>\ref_designs\qpsk_modem_ref\projects\target_stratix in the Current Directory browser.

If you are targeting the Stratix EP1S80 DSP development board, browse to the directory c:\MegaCore\Stratix_DSP_Pro_kit-<version>\ref_designs\qpsk_modem_ref\projects\target_stratix in the Current Directory browser.

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

4. Review the schematic design. The modem design used in this exercise is the same design that is used in Exercises 1 and 2 except for the following differences:

– The OpenCore Plus versions of the MegaCore functions are replaced with the OpenCore versions
– The SignalTap probe points are not present

The MegaCore functions only generate ModelSim simulation models if you use OpenCore versions of the core.

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

For shorter simulation times, use the ModelSim PE or SE software version 5.6.

6. Double-click the SignalCompiler block in your model. Click Analyze.

7. Click the Testbench tab.

8. Turn on the Generate Stimuli for VHDL Testbench option.

9. Click Convert MDL to VHDL. SignalCompiler generates a simulation script, tb_modem-ref.tcl, and a VHDL testbench that imports the Simulink input stimuli, tb_modem_ref.vhd.

10. Click OK.

11. Close the QPSK modem design file when you are finished generating the input stimulus files.
Because the AWGN channel block is a non-Altera blockset, it is treated as a black-box during the VHDL conversion. In the VHDL domain, it is subistuted with a simple wire design, *awgn_bb.vhd*. To add this file in the ModelSim Tcl script generated by Signal Compiler, perform the following steps:

1. Open the *tb_modem_ref_demo.tcl* Tcl script using any text editor.

2. Add the following line to the Tcl script just before the load simulation section:

   ```tcl
   Vcom -93 -explicit -work work "$workdir/awgn_bb.vhd"
   ``

3. Save the file.

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_modem_ref.tcl* script and click **Open**.

6. The simulation results are displayed in a waveform (see **Figure 20**).
7. A Simulink simulation stores a number of port values in the MATLAB workspace through the matrix variable \( y_{OUT} \). The ModelSim testbench stores the port values in text files every clock cycle. To compare the \( y_{OUT} \) with the text ports, type the following command at the MATLAB prompt:

\[
\text{compare\_all}
\]

Exercise 4: Analyze the Results in Hardware

Exercise 4 covers the following actions:

1. Set up the board for hardware analysis.

2. Configure the Stratix device on the Stratix DSP development board with the QPSK modem design.

3. Run SignalTap II analysis in the DSP Builder to examine the output and compare it with the Simulink plot.

Set Up the Board for Hardware Analysis

Before performing the exercises, you must connect the ByteBlaster™ II cable to the Stratix DSP board. To connect the cable, connect the ByteBlaster II cable to your PC and to the board’s 10-pin JTAG header for Stratix configuration.

For detailed instructions on how to connect cables to the board, refer to the Stratix DSP Development Kit Getting Started User Guide.
Configure the Stratix Device

To configure the Stratix device, perform the following steps:

1. Run the MATLAB software.

2. If you are targeting the Stratix EP1S25 DSP development board, browse to the directory `c:\MegaCore\Stratix_DSP_kit-<version>\ref_designs\qpsk_modem_ref\projects\target_stratix_ocp` in the Current Directory browser.

   If you are targeting the Stratix EP1S80 DSP development board, browse to the directory `c:\MegaCore\Stratix_DSP_Pro_kit-<version>\ref_designs\qpsk_modem_ref\projects\target_stratix_ocp` in the Current Directory browser.

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


5. Click Convert MDL to VHDL. SignalCompiler generates the files that you need to run SignalTap II analysis in “Run SignalTap II Analysis” on page 28.

6. Optional. Click Quartus II Compilation to synthesize and compile the design. To save synthesis time, skip this step and configure the device with the programming file provided with the DSP Development Kit, Stratix Edition.

   You must review the Viterbi decoder and click Finish in the MegaWizard Plug-In (see step 18 on page 16), before compiling in the Quartus II software.

7. Click Program, to configure the device (see Figure 21).
Run SignalTap II Analysis

In `modem_ref_demo_ocp.mdl`, to specify rising edge as the trigger condition for the output `bit_out_s`, perform the following steps:

1. Double-click the **SignalTap Analyzer** block. The SignalTap Analyzer displays all of the nodes connected to SignalTap II blocks as signals to be analyzed.

2. Click on `bit_out_s` under **Signal Name**.

3. Choose **Rising Edge** in the **Trigger Condition** list.

4. Click **Change**. The condition is updated (see **Figure 22**).
To run the analyzer and display the results in a MATLAB plot, perform the following steps:

1. Reset the board (see Figure 23):
   a. Turn off SW3 pin 2.
   b. Press SW0.
   c. Turn on SW3 pin 2.

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 assert the enable signal.

4. Click **OK** in the SignalTap Analysis block when you are finished. Two MATLAB plots display the captured data. The MATLAB plots display the captured data in the time domain.

5. Close the MATLAB plot of the data displayed in binary format. Examine the MATLAB plot of the data with the grouped bus signals (see Figure 24).

**Figure 24. SignalTap II Signals in the Time Domain**

6. Compare the MATLAB waveform `bits_out` with the one generated in the ModelSim simulator (see Figure 20). The hardware results match the ModelSim simulation results.

You also have the option to inject errors into the channel using the push-button SW1, while displaying the normalized error count onto the seven-segment displays. To use this option, perform the following steps:

1. Turn on SW3 pin 8.
2. Press SW1 to insert errors into the transmission channel.
3. Once the seven segment display reaches 99, it will roll-over and LED0 turns on.

The resource utilization is 9224 LEs, 5 DSP blocks (26, 9 × 9-bit multipliers), and 53 Kbits RAM.