Contents

About this User Guide ................................................................. v
  Revision History ........................................................................ v
  How to Contact Altera ............................................................... v
  Typographic Conventions ....................................................... iv

Chapter 1. About this Megafunction
  Device Family Support .............................................................. 1–1
  Introduction ............................................................................... 1–1
  Features .................................................................................... 1–1
  General Description ................................................................. 1–2
    Power-Up Mode Calibration and On-Demand Calibration
      in Quartus II ........................................................................ 1–3
  Common Applications ............................................................. 1–4
  Resource Utilization & Performance ....................................... 1–4

Chapter 2. Getting Started
  Software and System Requirements .......................................... 2–1
  MegaWizard Customization ..................................................... 2–1
  MegaWizard Page Descriptions .............................................. 2–2
  Inferring Megafuctions from HDL Code ................................. 2–7
  Instantiating Megafunctions in HDL Code ................................ 2–8
  Identifying a Megafuction after Compilation ......................... 2–8
  Simulation ................................................................................ 2–8
    Quartus II Simulation ........................................................... 2–8
    EDA Simulation .................................................................... 2–9
  SignalTap II Embedded Logic Analyzer .................................... 2–9
  Design Examples: Series Calibration for Four Calibration Blocks Using Stratix III ...................................................... 2–10
  Design Files ............................................................................. 2–10
  Example .................................................................................. 2–10
    Generate the Dynamic On-Chip Termination Calibration Blocks ......................................................... 2–10
    Assignment Editor—Verify Termination/Calibration Assignments to the oct_test
    Design in Quartus II ............................................................. 2–16
    Implement the oct_test Design ............................................. 2–18
    Functional Results—Simulate the oct_test Design in Quartus II ............................................................ 2–19
    Functional Results—Simulate the oct_test Design in ModelSim-Altera .............................................. 2–23
About this User Guide

Revision History

The table below displays the revision history for the chapters in this User Guide.

<table>
<thead>
<tr>
<th>Date</th>
<th>Document Version</th>
<th>Changes Made</th>
</tr>
</thead>
<tbody>
<tr>
<td>December 2006</td>
<td>1.0</td>
<td>Initial release.</td>
</tr>
</tbody>
</table>

How to Contact Altera

For the most up-to-date information about Altera® products, go to the Altera world-wide web site at www.altera.com. For technical support on this product, go to www.altera.com/mysupport. For additional information about Altera products, consult the sources shown below.

<table>
<thead>
<tr>
<th>Information Type</th>
<th>USA &amp; Canada</th>
<th>All Other Locations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Technical support</td>
<td><a href="http://www.altera.com/mysupport/">www.altera.com/mysupport/</a></td>
<td>altera.com/mysupport/</td>
</tr>
<tr>
<td></td>
<td>(800) 800-EPLD (3753) (7:00 a.m. to 5:00 p.m. Pacific Time)</td>
<td>(408) 544-7000 (1) (7:00 a.m. to 5:00 p.m. Pacific Time)</td>
</tr>
<tr>
<td>Product literature</td>
<td><a href="http://www.altera.com">www.altera.com</a></td>
<td><a href="http://www.altera.com">www.altera.com</a></td>
</tr>
<tr>
<td>Altera literature services</td>
<td><a href="mailto:literature@altera.com">literature@altera.com</a> (1)</td>
<td><a href="mailto:literature@altera.com">literature@altera.com</a> (1)</td>
</tr>
<tr>
<td>Non-technical customer service</td>
<td>(800) 767-3753 (7:00 a.m. to 5:00 p.m. Pacific Time)</td>
<td>(408) 544-7000 (7:30 a.m. to 5:30 p.m. Pacific Time)</td>
</tr>
<tr>
<td>FTP site</td>
<td>ftp.altera.com</td>
<td>ftp.altera.com</td>
</tr>
</tbody>
</table>

Note to table:
(1) You can also contact your local Altera sales office or sales representative.
Typographic Conventions

This document uses the typographic conventions shown below.

<table>
<thead>
<tr>
<th>Visual Cue</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Bold Type with Initial Capital Letters</strong></td>
<td>Command names, dialog box titles, checkbox options, and dialog box options are shown in bold, initial capital letters. Example: <strong>Save As</strong> dialog box.</td>
</tr>
<tr>
<td><strong>bold type</strong></td>
<td>External timing parameters, directory names, project names, disk drive names, filenames, filename extensions, and software utility names are shown in bold type. Examples: ( f_{\text{MAX}} ), \texttt{qdesigns} directory, \texttt{d:} drive, \texttt{chiptrip.gdf} file.</td>
</tr>
<tr>
<td><strong>Italic Type with Initial Capital Letters</strong></td>
<td>Document titles are shown in italic type with initial capital letters. Example: <em>AN 75: High-Speed Board Design</em>.</td>
</tr>
<tr>
<td><strong>Italic type</strong></td>
<td>Internal timing parameters and variables are shown in italic type. Examples: ( t_{\text{PIA}} ), ( n + 1 ). Variable names are enclosed in angle brackets (&lt; &gt;) and shown in italic type. Example: (&lt;\text{file name}&gt;), \texttt{&lt;project name&gt;.pof} file.</td>
</tr>
<tr>
<td>Initial Capital Letters</td>
<td>Keyboard keys and menu names are shown with initial capital letters. Examples: Delete key, the Options menu.</td>
</tr>
<tr>
<td>”Subheading Title”</td>
<td>References to sections within a document and titles of on-line help topics are shown in quotation marks. Example: ”Typographic Conventions.”</td>
</tr>
<tr>
<td><strong>Courier type</strong></td>
<td>Signal and port names are shown in lowercase Courier type. Examples: \texttt{data1}, \texttt{tdi}, \texttt{input}. Active-low signals are denoted by suffix ( n ), e.g., \texttt{resetn}. Anything that must be typed exactly as it displays is shown in Courier. For example: \texttt{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 \texttt{SUBDESIGN}), as well as logic function names (e.g., \texttt{TRI}) are shown in Courier.</td>
</tr>
<tr>
<td>1., 2., 3., and a., b., c., etc.</td>
<td>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.</td>
</tr>
<tr>
<td>• • •</td>
<td>Bullets are used in a list of items when the sequence of the items is not important.</td>
</tr>
<tr>
<td>✔</td>
<td>The checkmark indicates a procedure that consists of one step only.</td>
</tr>
<tr>
<td>❗</td>
<td>The hand points to information that requires special attention.</td>
</tr>
<tr>
<td>!</td>
<td>A caution calls attention to a condition or possible situation that can damage or destroy the product or the user's work.</td>
</tr>
<tr>
<td>☢</td>
<td>A warning calls attention to a condition or possible situation that can cause injury to the user.</td>
</tr>
<tr>
<td>↘</td>
<td>The angled arrow indicates you should press the Enter key.</td>
</tr>
<tr>
<td>❖</td>
<td>The feet direct you to more information on a particular topic.</td>
</tr>
</tbody>
</table>
Chapter 1. About this Megafunction

Device Family Support

Megafunsions provide either full or preliminary support for target Altera® device families.

- **Full support** means the megafunction meets all functional and timing requirements for the device family and may be used in production designs.
- **Preliminary support** means the megafunction meets all functional requirements, but may still be undergoing timing analysis for the device family. It may be used in production designs with caution.

Table 1–1 shows the level of support offered by the ALT_OCT megafunction for each Altera device family.

<table>
<thead>
<tr>
<th>Device Family</th>
<th>Support</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stratix® III</td>
<td>Preliminary</td>
</tr>
<tr>
<td>Other device families</td>
<td>No support</td>
</tr>
</tbody>
</table>

Introduction

As design complexities increase, use of vendor-specific intellectual property (IP) blocks has become a common design methodology. Altera provides parameterizable megafunsions that are optimized for Altera device architectures. Using megafunsions instead of coding your own logic saves valuable design time. The Altera-provided megafunsions offer more efficient logic synthesis and device implementation. You can scale the megafunction’s size by setting parameters.

Features

Core features of this megafunction include:

- 8 to 10 on-chip termination (OCT) calibration blocks available for Stratix III devices.
- Support for calibrated on-chip series termination (RS) and calibrated on-chip parallel termination (RT) on all I/O pins.
- Calibrated termination values of 25 and 50 ohm.
General Description

The new OCT calibration architecture in Stratix III provides more flexibility and accuracy than the previous architecture. The SDATA, OCTUSRCLK, and ENASER signals are used to serially transfer calibrated codes from each OCT calibration block to any I/O.

To serially shift the 14-bit OCT RS calibration code and the 14-bit OCT RT calibration code into registers in the I/O buffer, 28 clock cycles using OCTUSRCLK are required. When calibration is complete, you must serially shift out the 28-bit OCT calibration code (14-bit OCT RS code and 14-bit OCT RT) from each OCT calibration block to the corresponding I/O buffer.

After calibrated codes are shifted in serially to each I/O bank, the calibrated codes must be converted from serial format to parallel format before being used in the I/O buffers. Use the S2PENA signals to complete serial-to-parallel shifting.

For more information, refer to the Stratix III Device I/O Features chapter in volume 1 of the Stratix III Device Handbook. In particular, refer to the Calibration section. This section provides details about the OCT calibration block location, OCT calibration block architecture, and OCT calibration block modes of operation.

Figure 1–1 shows the signals for shift-out codes from the OCT calibration block to I/O buffers.
Power-Up Mode Calibration and On-Demand Calibration in Quartus II

Stratix III has two termination related assignments: INPUT_TERMINATION and OUTPUT_TERMINATION. Termination can exist on input and output buffers, and sometimes simultaneously.

When calibrated termination uses only the Quartus® Settings File (QSF) assignments, the power-up mode of the calibration scheme is used, and on-demand calibration updates are unavailable.

To use on-demand calibration, the ALT_OCT megafuction must be instantiated into the design. If more than one group of pins needs to be calibrated on-demand, more than one calibration block must be instantiated.

There are two methods to associate pin groups with a calibration block:

- Instantiate the I/O buffer primitives at the top level and connect them to the appropriate calibration blocks.
Resource Utilization & Performance

- Use a QSF assignment to indicate which pin (bus) is associated with which calibration block.

Common Applications

A typical application area is in double-data rate (DDR) external memory interfaces. This megafunction is closely associated with the `altmemphy` megafunction.

For more information, refer to the *Stratix III Device I/O Features* chapter in volume 1 of the *Stratix III Device Handbook* and the *altmemphy Megafunction User Guide*.

Table 1–2 summarizes the resource usage of the ALT_OCT megafunction for the Stratix III device.

<table>
<thead>
<tr>
<th>Termination Type</th>
<th>Number of OCT Blocks</th>
<th>Resource Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Series 1</td>
<td>1</td>
<td>3 lpm_counter + 7 reg</td>
</tr>
<tr>
<td>Series 2</td>
<td>2</td>
<td>3 lpm_counter + 12 reg</td>
</tr>
<tr>
<td>Series 3</td>
<td>3</td>
<td>3 lpm_counter + 17 reg</td>
</tr>
<tr>
<td>Series 4</td>
<td>4</td>
<td>3 lpm_counter + 22 reg</td>
</tr>
<tr>
<td>Series 5</td>
<td>5</td>
<td>3 lpm_counter + 27 reg</td>
</tr>
<tr>
<td>Series 6</td>
<td>6</td>
<td>3 lpm_counter + 32 reg</td>
</tr>
<tr>
<td>Series 7</td>
<td>7</td>
<td>3 lpm_counter + 37 reg</td>
</tr>
<tr>
<td>Series 8</td>
<td>8</td>
<td>3 lpm_counter + 42 reg</td>
</tr>
<tr>
<td>Series 9</td>
<td>9</td>
<td>3 lpm_counter + 47 reg</td>
</tr>
<tr>
<td>Series 10</td>
<td>10</td>
<td>3 lpm_counter + 52 reg</td>
</tr>
<tr>
<td>Parallel 1</td>
<td>1</td>
<td>3 lpm_counter + 7 reg</td>
</tr>
<tr>
<td>Parallel 2</td>
<td>2</td>
<td>3 lpm_counter + 12 reg</td>
</tr>
<tr>
<td>Parallel 3</td>
<td>3</td>
<td>3 lpm_counter + 17 reg</td>
</tr>
<tr>
<td>Parallel 4</td>
<td>4</td>
<td>3 lpm_counter + 22 reg</td>
</tr>
<tr>
<td>Parallel 5</td>
<td>5</td>
<td>3 lpm_counter + 27 reg</td>
</tr>
<tr>
<td>Parallel 6</td>
<td>6</td>
<td>3 lpm_counter + 32 reg</td>
</tr>
<tr>
<td>Parallel 7</td>
<td>7</td>
<td>3 lpm_counter + 37 reg</td>
</tr>
<tr>
<td>Parallel 8</td>
<td>8</td>
<td>3 lpm_counter + 42 reg</td>
</tr>
<tr>
<td>Parallel 9</td>
<td>9</td>
<td>3 lpm_counter + 47 reg</td>
</tr>
<tr>
<td>Parallel 10</td>
<td>10</td>
<td>3 lpm_counter + 52 reg</td>
</tr>
</tbody>
</table>
About this Megafunction
Chapter 2. Getting Started

Software and System Requirements

The instructions in this section require the following software:

- For OS Support information, refer to:
  http://www.altera.com/support/software/os_support/oss-index.html
- The Quartus® II software version 6.1 or higher

MegaWizard Customization

The MegaWizard® Plug-In Manager creates or modifies design files that contain custom megafunction variations which can then be instantiated in a design file. The MegaWizard Plug-In Manager provides a wizard that allows you to specify options for the ALT_OCT megafunction. You can use the wizard to set the ALT_OCT megafunction features in the design.

Start the MegaWizard Plug-In Manager using one of the following methods:

- On the Tools menu, click MegaWizard Plug-In Manager.
- When working in the Block Editor, click MegaWizard Plug-In Manager in the Symbol window.
- Start the stand-alone version of the MegaWizard Plug-In Manager by typing the following command at the command prompt:
  qmegawiz
MegaWizard Page Descriptions

This section provides descriptions of the options available on the individual pages of the ALT_OCT MegaWizard Plug-In Manager. Use the MegaWizard Plug-In Manager to access this wizard in the following location:

libraries\megafun\IO\ALT_OCT

On page 1 of the ALT_OCT MegaWizard Plug-In Manager, select Create a new custom megafunction variation, Edit an existing custom megafunction variation, or Copy an existing custom megafunction variation (Figure 2–1).

Figure 2–1. MegaWizard Plug-In Manager [page 1]
Getting Started

On page 2a of the ALT_OCT MegaWizard Plug-In Manager, specify the family of device you want to use, the type of output file to create, and the name of the output file (Figure 2–2). You can choose AHDL (.tdf), VHDL (.vhd), or Verilog HDL (.v) as the output file type. A clear box netlist is always generated when using this megafunction.

Figure 2–2. MegaWizard Plug-In Manager [page 2a]

On page 3 of the ALT_OCT MegaWizard Plug-In Manager, select the type of termination, specify how many calibration blocks are to be used, specify optional inputs, and specify whether to enable automatic power-down when calibration is finished (Figure 2–3).
Table 2–1 shows the options available on page 3 of the ALT_OCT MegaWizard Plug-In Manager.

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Currently selected device family</td>
<td>Specify the device family you will use. The only option is Stratix III.</td>
</tr>
<tr>
<td>How many OCT blocks should be used?</td>
<td>Select from a minimum of 1 calibration block to a maximum of 10 calibration blocks.</td>
</tr>
<tr>
<td>Enable parallel termination</td>
<td>Specify the type of termination to use, either series serial termination (output) or parallel termination (input or bidirectional). If this option is disabled, series termination is enabled.</td>
</tr>
</tbody>
</table>
Page 4 of the ALT_OCT MegaWizard Plug-In Manager highlights the atom simulation library to be used during functional simulation, which in this case is Stratix III (Figure 2–4).

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create 'calibration_wait' input port to prevent calibration</td>
<td>When enabled, this option creates an input port, calibration_wait, which can be used to halt the calibration process. The calibration_wait option is for advanced users only. Typical users should not enable this option.</td>
</tr>
<tr>
<td>Create 'clken' input port</td>
<td>When enabled, this option creates an input port, clken, which is used as the clock enable signal for this block. The clken input port is for advanced users only. Typical users should not enable this option.</td>
</tr>
<tr>
<td>Create 'aclr' port</td>
<td>When enabled, this option creates an input port, aclr, which is used as the asynchronous clear signal for this block. The aclr signal is for advanced users only. Typical users should not enable this option.</td>
</tr>
</tbody>
</table>
On page 5 of the ALT_OCT MegaWizard Plug-In Manager, specify the types of files to be generated (Figure 2–5). Choose from the HDL wrapper file, \textit{<function name>.v}, \textit{<function name>.vhd}, \textit{<function name>.tdf}, \textit{<function name>.bsf}, \textit{<function name>_inst.v}, or \textit{<function name>_bb.v}.

The gray check marks indicate files that are always generated; the other files are optional and are generated only if selected (indicated by a red check mark). Turn on the boxes to select the files that you want generated.
Getting Started

**Inferring Megafunctions from HDL Code**

Synthesis tools, including Quartus II integrated synthesis, recognize certain types of HDL code and automatically infer the appropriate megafunction when a megafunction can provide optimal results. The Quartus II software uses the Altera® megafunction code when compiling your design, even though you did not specifically instantiate the megafunction. The Quartus II software infers megafunctions because they are optimized for Altera devices, so area usage and performance may be better than generic HDL code. Additionally, you must use megafunctions to access certain Altera architecture-specific features, including memory, DSP blocks, and shift registers. These features provide improved performance when compared to basic logic elements.

Refer to the Recommended HDL Coding Styles chapter in volume 1 of the Quartus II Handbook for specific information about your particular megafunction.
Instantiating Megafucntions in HDL Code

When you use the MegaWizard Plug-In Manager to set up and parameterize a megafunction, it creates either a VHDL or Verilog HDL wrapper file that instantiates the megafunction (a black box methodology). For some megafunctions, you can generate a fully synthesizable netlist for improved results with EDA synthesis tools, such as Synplify and Precision RTL Synthesis (a clear box methodology). Both clear box and black box methodologies are described in the third-party synthesis support chapters in the Synthesis section in volume 1 of the Quartus II Handbook.

Identifying a Megafunction after Compilation

During compilation with the Quartus II software, analysis and elaboration is performed to build the structure of your design. To locate your megafunction in the Project Navigator window, expand the compilation hierarchy and find the megafunction by its name. To search for node names within the megafunction (using the Node Finder), click Browse in the Look in box and select the megafunction in the Hierarchy box.

Simulation

The Quartus II Simulator provides an easy-to-use, integrated solution for performing simulations. The following sections describe the simulation options.

Quartus II Simulation

With the Quartus II Simulator, you can perform two types of simulations: functional and timing. A functional simulation enables you to verify the logical operation of your design without taking into consideration the timing delays in the FPGA. This simulation is performed using only your RTL code. When performing a functional simulation, add only signals that exist before synthesis. You can find these signals with the Registers: Pre-Synthesis, Design Entry, or Pin filters in the Node Finder. The top-level ports of megafunctions are found using these three filters.

In contrast, the timing simulation in the Quartus II software verifies the operation of your design with annotated timing information. This simulation is performed using the post place-and-route netlist. When performing a timing simulation, add only signals that exist after place-and-route. These signals are found with the post-compilation filter of the Node Finder. During synthesis and place-and-route, the names of RTL signals change. Therefore, it may be difficult to find signals from your megafunction instantiation in the post-compilation filter. To preserve the names of your signals during the synthesis and place-and-route stages, use the synthesis attributes keep or preserve. These are Verilog and VHDL synthesis attributes that direct analysis and
Getting Started

synthesis to keep a particular wire, register, or node intact. Use these synthesis attributes to keep a combinational logic node so you can observe the node during simulation.

For more information about these attributes, refer to the Integrated Synthesis chapter in volume 1 of the Quartus II Handbook.

EDA Simulation

Depending on the simulation tool you are using, refer to the appropriate chapter in the Simulation section in volume 3 of the Quartus II Handbook. The Quartus II Handbook chapters describe how to perform functional and gate-level timing simulations that include the megafunctions, with details about the files that are needed and the directories where the files are located.

SignalTap II Embedded Logic Analyzer

The SignalTap® II embedded logic analyzer provides a non-intrusive method of debugging the Altera megafunctions within your design. With the SignalTap II embedded logic analyzer, you can capture and analyze data samples for the top-level ports of Altera megafunctions while your system is running at full speed.

To monitor signals from Altera megafunctions, configure the SignalTap II embedded logic analyzer in the Quartus II software, and include the analyzer as part of your Quartus II project. The Quartus II software then embeds the analyzer in your design in the selected device seamlessly.

For more information about using the SignalTap II embedded logic analyzer, refer to the Design Debugging Using the SignalTap II Embedded Logic Analyzer chapter in volume 3 of the Quartus II Handbook.
This section presents a design example that uses the ALT_OCT megafunction to calibrate four calibration blocks using series termination for the Stratix III device. This example uses the MegaWizard Plug-In Manager in the Quartus II software. As you go through the wizard, each page is described in detail. When you are finished with this example, you can incorporate it into your overall project.

**Design Files**

The design files are available in the literature page of the Altera web site under user guides:

http://www.altera.com/literature/lit-ug.jsp

Select the Examples for ALT_OCT Megafunction User Guide link from the examples page to download the design files.

**Example**

In this example, you perform the following tasks:

- Generate dynamic on-chip termination calibration blocks using the ALT_OCT megafunction in the MegaWizard Plug-In Manager
- Make sure the right termination assignments are used in the design by using the Assignment Editor.
- Implement the oct_test project by assigning the EP3SE50F780C2 device to the project and compiling the project
- Simulate the oct_test design in Quartus II
- Simulate the oct_test design in ModelSim-Altera

**Generate the Dynamic On-Chip Termination Calibration Blocks**

To generate the dynamic on-chip termination calibration blocks, perform the following steps:

1. Unzip alt_oct_DesignExample.zip to any working directory on your PC.
2. Open the project file oct_test.qar.
3. Open the top-level file oct_test.bdf. This is an incomplete file that you will complete as part of this example.
4. Double-click on a blank area in the Block Design File (.bdf).
5. In the Symbol window, click MegaWizard Plug-In Manager.
Getting Started

6. On page 1, select Create a new custom megafunction variation.

7. Click Next. Page 2a appears (Figure 2–6).

**Figure 2–6. Wizard Plug-In Manager [page 2a]**

8. In the Which device family will you be using? list, select Stratix III.

9. Expand the I/O folder by clicking the “+” icon and select ALTOCT.

10. Under Which type of output file do you want to create?, select Verilog HDL.

11. Name the output file cal_out.

12. Turn on Generate clear box netlist file instead of a default wrapper file (for use with supported EDA synthesis tools only).

13. Click Next. Page 3 appears (Figure 2–7).
14. Under **How many OCT blocks should be used?**, select **4** from the drop-down menu.

15. Ensure that **Enable parallel termination** is turned off.

16. Turn off **Create 'calibration_wait' input port to prevent calibration**, **Create 'clken' input port** and **Create 'aclr' input port**.

17. Click **Next**. Page 4 appears (Figure 2–8).
18. No input is required on this page. Click **Next**. Page 5 appears (Figure 2–9).
The final page of the wizard shows the files that are generated for your custom megafunction variation. The gray check marks indicate files that are always generated; the other files are optional and are generated only if selected (indicated by a red check mark). Turn on the boxes to select the files that you want generated.

19. Select all available output files.

20. Click Finish. The cal_oct module is built.
Getting Started

21. In the Symbol window (Figure 2–10), click OK.

22. Move the mouse to place the cal_out symbol between the input/output ports in the oct_test.bdf file. Click to place the symbol. You have now completed the design file, as shown in Figure 2–11.

23. On the File menu, click Save.
Assignment Editor—Verify Termination/Calibration Assignments to the \textit{oct\_test} Design in Quartus II

This section describes the necessary assignments when on-chip calibration blocks are used in the \textit{oct\_test} design:

1. In the Quartus II software, on the Assignments menu, click Assignment Editor. The Assignment Editor appears. Figure 2–12 shows the assignments used in the \textit{oct\_test} design.
The first set of assignments are I/O Standard assignments which are used on the pins of the design. This selection depends on which I/O standards are valid for a particular application. In this design example, the functional output pins are `test_output[3..0]`, which are all assigned the SSTL-18 Class 1 standard.

The second set of assignments are Output Termination assignments which are assigned to pins that will have termination. You can specify non-calibrated termination or calibrated termination. In this design example, the functional output pins are `test_output[3..0]`, which are all assigned the Output Termination assignments with the value of series 50 ohm with calibration. In this assignment, these pins are functioning as outputs only for the FPGA chip.

The third set of assignments are Termination Control Block assignments which are used to indicate which calibration block will be calibrating a particular I/O pin/group. In this design example, the functional output pins are `test_output[3..0]`, which are all assigned the Termination Control Block assignments with the value of `cal_out:inst|cal_out_alt_oct_cio:cal_out_alt_oct_cio_component|sdia_x`, where `x` indicates the termination.
calibration block used. This is based on the megafunction, which in this case uses four calibration blocks. Since the output pins are four bits, each bit of the pin has been assigned to one calibration block.

2. Verify that all the assignments in the Assignment Editor match what is shown in Figure 2–12.

**Implement the oct_test Design**

This section describes how to implement the oct_test design, assign the EP3SE50F780C2 device to the project, and compile the project.

To implement the oct_test design, perform the following steps:

1. In the Quartus II software, on the Assignments menu, click **Settings**. The **Settings** dialog box appears (Figure 2–13).

   **Figure 2–13. Settings Dialog Box**

   ![Settings Dialog Box](image)

   2. In the **Category** list, select **Device**.
3. In the Family list, select Stratix III.


5. Click OK.

6. To compile the design, on the Processing menu, click Start Compilation, or click the Compilation button on the toolbar.

7. When the Full Compilation was successful message appears, click OK. (The message may indicate a number of warnings; this is unimportant to the design.)

Functional Results—Simulate the oct_test Design in Quartus II

This section describes how to verify the design example you created by simulating the design using the Quartus II Simulator. To set up the Quartus II Simulator, perform the following steps:

1. In the Quartus II software, with oct_test.qar open, on the Processing menu, click Generate Functional Simulation Netlist.

2. When the Functional Simulation Netlist Generation was successful message appears, click OK.

3. On the Assignments menu, click Settings. The Settings dialog box appears (Figure 2–14).
4. From the **Category** list, select **Simulator Settings**.

5. From the **Simulation mode** drop-down list, select **Functional**.

6. Type `oct_test.vwf` in the **Simulation input** box, or browse to select the file in the project folder.

7. Turn on **Run simulation until all vector stimuli are used**.

8. Turn on **Automatically add pins to simulation output waveforms** and **Simulation coverage reporting**.

9. Turn off **Overwrite simulation input file with simulation results**.

10. Click **OK**.
11. On the Processing menu, click **Start Simulation**.

12. When the **Simulator was successful** message appears, click **OK**.

13. The Simulation Report window appears. Verify the simulation waveform results (**Figure 2–15**).

**Figure 2–15. Simulation Waveforms**

For these functional results, the primary objective is to highlight the dynamic calibration process. This design uses four calibration blocks:

- calibration block [3]
- calibration block [2]
- calibration block [1]
- calibration block [0]

Each calibration block has its own set of control/data signals. For example, calibration block [3] has calibration_request[3], calibration_busy[3], and cal_shift_busy[3] signals.
To request calibration, \texttt{calibration\_request\[x\]} must be asserted for at least 1 clock cycle to initiate the calibration process for that particular calibration block. Multiple calibration block requests are allowed at a particular time.

The \texttt{calibration\_busy\[x\]} signal indicates the calibration process that is being performed in that particular calibration block. Calibration typically takes 200 clock cycles. Calibration of that particular block is complete when the \texttt{calibration\_busy\[x\]} signal is deasserted.

The \texttt{cal\_shift\_busy\[x\]} signal is asserted at the same time as the \texttt{calibration\_busy\[x\]} signal. This signal actually indicates both the duration of the calibration and the serial shifting of termination codes from OCT calibration blocks to the particular I/O buffers. Serial shifting of series termination codes typically takes 14 clock cycles.

After calibrated codes are shifted in serially to the particular I/O bank, the calibrated codes must be converted from serial format to parallel format before being used in the I/O buffers. The \texttt{s2pload\[x\]} signal can be asserted at any time for 1 clock cycle after the shifting process to achieve this.

In the simulation waveform shown in Figure 2-15, calibration block [3], calibration block [2], calibration block [1], and calibration block [0] have their respective \texttt{calibration\_request} signals asserted at a particular time. Initially, requests for calibration are done simultaneously for calibration block [0] and calibration block [3]. This can be observed in the behavior of the \texttt{calibration\_request\[3\]} and \texttt{calibration\_request\[0\]} signals. Both are asserted and deasserted at the same time. This is because multiple OCT calibration blocks can be calibrated at the same time. When these signals get deasserted, calibration is complete.

Observe the behavior of the \texttt{calibration\_busy\[3\]} and \texttt{calibration\_busy\[0\]} signals. Notice that both are also asserted and deasserted at the same time. This is because multiple OCT calibration blocks can be calibrated at the same time. When these signals get deasserted, calibration is complete.

Next, observe the \texttt{cal\_shift\_busy\[3\]} and \texttt{cal\_shift\_busy\[0\]} signals. They are both asserted at the same time as the \texttt{calibration\_busy\[3\]} and \texttt{calibration\_busy\[0\]} signals, and both get deasserted after a number of clock cycles from their respective \texttt{calibration\_busy\[3\]} and \texttt{calibration\_busy\[0\]} signals. This is because these signals indicate both the calibration and serial shifting of the termination codes.
Notice that the `cal_shift_busy[3]` and `cal_shift_busy[0]` signals get deasserted at different time—`cal_shift_busy[0]` gets deasserted first and then only `cal_shift_busy[3]`. This is because the shifting process is serial, and only one calibration block can be active at any one time shifting out the termination codes. Priority for which block should shift out codes in the event of multiple calibration block requests is based on \([0, 1, 2, 3, 4, 5, 6, 7, 8, 9]\), where 0 (calibration block 0) shifts codes first and is then followed by the other blocks (1,2,3,4..9).

`cal_shift_busy[0]` gets deasserted first, and then only `cal_shift_busy[3]`, because calibration block [0] shifts out the termination codes before calibration block [3].

Finally, the calibrated codes must be converted from serial format to parallel format before being used in the I/O buffers. This is done by asserting and deasserting the `s2pload[0]` and `s2pload[3]` signals for 1 clock cycle only after `cal_shift_busy[3]` and `cal_shift_busy[0]` signals have been deasserted. This is also shown in Figure 2–15.

The remaining portion of the waveform is similar to what is described above, but applies to calibration block [2] and calibration block [1].

**Functional Results—Simulate the oct_test Design in ModelSim-Altera**

Simulate the design in the ModelSim-Altera tool to compare the results of both simulators.

This User Guide assumes that you are familiar with using the ModelSim-Altera tool before trying out the design example. If you are unfamiliar with ModelSim-Altera, refer to www.altera.com/support/software/products/modelsim/mod-modelsim.html, a support page for ModelSim-Altera. There are links to topics such as installation, usage, and troubleshooting.

For this megafunction, the new Stratix III pre-compiled library is required, and only ModelSim-Altera 6.1 supports this.

Set up the ModelSim-Altera simulator by performing the following steps:

1. Unzip the `alt_oct_msim.zip` file to any working directory on your PC.
2. Browse to the folder in which you unzipped the files and open the `alt_oct_msim.do` file in a text editor.
3. In line 1 of the `alt_oct_msim.do` file, replace
   `<insert_directory_path_here>` with the directory path of the
   appropriate library files. For example,

   C:/altera/61/modelsim_ae/altera/verilog/stratixiii

   Then, in line 2 of the `alt_oct_msim.do` file, replace
   `<insert_directory_path_here>` with the directory path of the
   appropriate supporting library files. For example,

   C:/altera/61/modelsim_ae/altera/verilog/altera

4. On the File menu, click Save.

5. Start ModelSim-Altera.


7. Select the folder in which you unzipped the files and click OK.

8. On the Tools menu, click Execute Macro.

9. Select the `alt_oct_msim.do` file and click Open. This is a script file
   for ModelSim that automates all the necessary settings for the
   simulation.

10. Verify the results by looking at the Waveform Viewer window.
    You may need to rearrange signals, remove redundant signals, and
    change the radix to suit the results in the Quartus II Simulator.
Figure 2–16 shows the expected simulation results in ModelSim.

**Figure 2–16. Functional Simulation Results in ModelSim - Altera**

![Simulation Results](image)

### Conclusion

The Quartus II software provides parameterizable megafunctions ranging from simple arithmetic units, such as adders and counters, to advanced phase-locked loop (PLL) blocks, divisions, and memory structures. These megafunctions are performance-optimized for Altera devices and therefore, provide more efficient logic synthesis and device implementation because they automate the coding process and save valuable design time. Altera recommends using these functions during design implementation so you can consistently meet your design goals.
Chapter 3. Specifications

Ports & Parameters

The parameter details in this chapter are only relevant for users who bypass the MegaWizard® Plug-In Manager interface and use the megafunction as a directly parameterized instantiation in their design. The details of these parameters are hidden from MegaWizard Plug-In Manager interface users. The options listed in this section describe all of the ports and parameters that are available for each device to customize the ALT_OCT megafunction according to your application.

Refer to the latest version of the Quartus® II software Help for the most current information on the ports and parameters for this megafunction.

Figure 3–1 shows the ports for the ALT_OCT megafunction.

![Figure 3–1. ALT_OCT Megafunction Ports](image)
Table 3–1 shows the input ports, Table 3–2 shows the output ports, and Table 3–3 shows the parameters for the ALT_OCT megafunction.

### Table 3–1. ALT_OCT Megafunction Input Ports

<table>
<thead>
<tr>
<th>Port Name</th>
<th>Required</th>
<th>Description</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>aclr</td>
<td>No</td>
<td>Asynchronous clear input.</td>
<td>If omitted, value is GND.</td>
</tr>
<tr>
<td>calibration_request</td>
<td>Yes</td>
<td>User request for calibration.</td>
<td></td>
</tr>
<tr>
<td>calibration_wait</td>
<td>No</td>
<td>Specifies the counter type.</td>
<td></td>
</tr>
<tr>
<td>clock</td>
<td>Yes</td>
<td>System clock.</td>
<td></td>
</tr>
<tr>
<td>rdn</td>
<td>No</td>
<td>Pull-down reference resistor.</td>
<td></td>
</tr>
<tr>
<td>rup</td>
<td>No</td>
<td>Pull-up reference resistor.</td>
<td></td>
</tr>
<tr>
<td>s2pload</td>
<td>Yes</td>
<td>Signal used to enable serial to parallel</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>shifting of calibrated codes to I/O buffers.</td>
<td></td>
</tr>
<tr>
<td>clken</td>
<td>No</td>
<td>Clock enable signal.</td>
<td>If omitted, value is VCC.</td>
</tr>
</tbody>
</table>

### Table 3–2. ALT_OCT Megafunction Output Ports (Part 1 of 2)

<table>
<thead>
<tr>
<th>Port Name</th>
<th>Required</th>
<th>Description</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>cal_shift_busy</td>
<td>Yes</td>
<td>This output signal is asserted until the calibration operation and shifting</td>
<td>Output port [OCT_BLOCK_NUMBER - 1..0] wide.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>of termination codes to the corresponding I/O buffer for that particular OCT block is finished.</td>
<td></td>
</tr>
<tr>
<td>calibration_busy</td>
<td>Yes</td>
<td>This output signal is asserted until the calibration operation for that particular OCT block is finished.</td>
<td>Output port [OCT_BLOCK_NUMBER - 1..0] wide.</td>
</tr>
</tbody>
</table>
Specifications

Table 3–2. ALT_OCT Megafunction Output Ports (Part 2 of 2)

<table>
<thead>
<tr>
<th>Port Name</th>
<th>Required</th>
<th>Description</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>parallelterminationcontrol</td>
<td>Yes</td>
<td>Specifies parallel termination.</td>
<td>Receives the current state of the pull-up and pull-down transmitter control buses from a termination logic block. Dedicated route in the hardware, and does not connect to core routing. Output port ([13..0]) wide. Available for Stratix III devices only.</td>
</tr>
<tr>
<td>serieterminationcontrol</td>
<td>Yes</td>
<td>Specifies series termination.</td>
<td>Receives the current state of the pull-up and pull-down receiver control buses from a termination logic block. Dedicated route in the hardware, and does not connect to core routing. Output port ([13..0]) wide. Available for Stratix III devices only.</td>
</tr>
</tbody>
</table>

Table 3–3. ALT_OCT Megafunction Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>ENABLE_PARALLEL_TERMINATION</td>
<td>String</td>
<td>Yes</td>
<td>Enables the parallel termination (RT) State Machine. Values are “TRUE” and “FALSE”. If omitted, value is FALSE.</td>
</tr>
<tr>
<td>OCT_BLOCK_NUMBER</td>
<td>Integer</td>
<td>Yes</td>
<td>The number of on-chip termination (OCT) blocks in the design. Value must be an integer greater than “0”.</td>
</tr>
</tbody>
</table>