15. In-System Modification of Memory and Constants

QII53012-12.0.0

This chapter explains how to use the Quartus®II In-System Memory Content Editor as part of your FPGA design and verification flow.

The In-System Memory Content Editor allows you to view and update memories and constants with the JTAG port connection.

The In-System Memory Content Editor allows access to dense and complex FPGA designs. When you program devices, you have read and write access to the memories and constants through the JTAG interface. You can then identify, test, and resolve issues with your design by testing changes to memory contents in the FPGA while your design is running.

Overview

This chapter contains the following sections:

When you use the In-System Memory Content Editor in conjunction with the SignalTap II Logic Analyzer, you can more easily view and debug your design in the hardware lab.

For more information about the SignalTap II Logic Analyzer, refer to the *Design Debugging Using the SignalTap II Logic Analyzer* chapter in volume 3 of the *Quartus II Handbook*.

The ability to read data from memories and constants allows you to quickly identify the source of problems. The write capability allows you to bypass functional issues by writing expected data. For example, if a parity bit in your memory is incorrect, you can use the In-System Memory Content Editor to write the correct parity bit values into your RAM, allowing your system to continue functioning. You can also intentionally write incorrect parity bit values into your RAM to check the error handling functionality of your design.

The Quartus II software offers a variety of on-chip debugging tools. For an overview and comparison of all tools available in the Quartus II software on-chip debugging tool suite, refer to *Section IV. System Debugging Tools* in volume 3 of the *Quartus II Handbook*.

For a list of the types of memories and constants currently supported by the Quartus II software, refer to *Megafunctions/LPM* in Quartus II Help.

Updating Memory and Constants in Your Design

To use the In-System Updating of Memory and Constants feature, perform the following steps:

1. Identify the memories and constants that you want to access.
2. Edit the memories and constants to be run-time modifiable.
3. Perform a full compilation.
4. Program your device.
5. Launch the In-System Memory Content Editor.

Creating In-System Modifiable Memories and Constants

When you specify that a memory or constant is run-time modifiable, the Quartus II software changes the default implementation. A single-port RAM is converted to a dual-port RAM, and a constant is implemented in registers instead of look-up tables (LUTs). These changes enable run-time modification without changing the functionality of your design.

To enable your memory or constant to be modifiable, refer to *Setting up the In-System Memory Content Editor* in Quartus II Help.

If you instantiate a memory or constant megafunction directly with ports and parameters in VHDL or Verilog HDL, add or modify the `lpm_hint` parameter as follows:

In VHDL code, add the following:

```
lpm_hint => "ENABLE_RUNTIME_MOD = YES,
  INSTANCE_NAME = <instantiation name>";
```

In Verilog HDL code, add the following:

```
defparam <megafunction instance name>.lpm_hint =
  "ENABLE_RUNTIME_MOD = YES,
  INSTANCE_NAME = <instantiation name>";
```

Running the In-System Memory Content Editor

The In-System Memory Content Editor has three separate panes: the **Instance Manager**, the **JTAG Chain Configuration**, and the **Hex Editor**.

The **Instance Manager** pane displays all available run-time modifiable memories and constants in your FPGA device. The **JTAG Chain Configuration** pane allows you to program your FPGA and select the Altera® device in the chain to update.

Using the In-System Memory Content Editor does not require that you open a project. The In-System Memory Content Editor retrieves all instances of run-time configurable memories and constants by scanning the JTAG chain and sending a query to the specific device selected in the **JTAG Chain Configuration** pane.

If you have more than one device with in-system configurable memories or constants in a JTAG chain, you can launch multiple In-System Memory Content Editors within the Quartus II software to access the memories and constants in each of the devices. Each In-System Memory Content Editor can access the in-system memories and constants in a single device.

Instance Manager

When you scan the JTAG chain to update the **Instance Manager** pane, you can view a list of all run-time modifiable memories and constants in the design. The **Instance Manager** pane displays the Index, Instance, Status, Width, Depth, Type, and Mode of each element in the list.

You can read and write to in-system memory with the **Instance Manager** pane. For more information refer to *Instance Manager Pane* in Quartus II Help.

In addition to the buttons available in the **Instance Manager** pane, you can read and write data by selecting commands from the Processing menu, or the right-click menu in the **Instance Manager** pane or **Hex Editor** pane.

The status of each instance is also displayed beside each entry in the **Instance Manager** pane. The status indicates if the instance is **Not running**, **Offloading data**, or **Updating data**. The health monitor provides information about the status of the editor.

The Quartus II software assigns a different index number to each in-system memory and constant to distinguish between multiple instances of the same memory or constant function. View the **In-System Memory Content Editor Settings** section of the Compilation Report to match an index number with the corresponding instance ID.

Editing Data Displayed in the Hex Editor Pane

You can edit data read from your in-system memories and constants displayed in the **Hex Editor** pane by typing values directly into the editor or by importing memory files.

For more information, refer to *Working with In-System Memory Content Editor Data* in Quartus II Help.

Importing and Exporting Memory Files

The In-System Memory Content Editor allows you to import and export data values for memories that have the In-System Updating feature enabled. Importing from a data file enables you to quickly load an entire memory image. Exporting to a data file enables you to save the contents of the memory for future use.

For more information, refer to *Working with In-System Memory Content Editor Data* in Quartus II Help.

Scripting Support

The In-System Memory Content Editor supports reading and writing of memory contents via a Tcl script or Tcl commands entered at a command prompt. For detailed information about scripting command options, refer to the Quartus II command-line and Tcl API Help browser.

To run the Help browser, type the following command at the command prompt:

```
quartus_sh --qhelp ↵
```

For more information about Tcl scripting, refer to the *Tcl Scripting* chapter in volume 2 of the *Quartus II Handbook* and *API Functions for Tcl* in Quartus II Help. For more information about command-line scripting, refer to the *Command-Line Scripting* chapter in volume 2 of the *Quartus II Handbook*.

The commonly used commands for the In-System Memory Content Editor are as follows:

- Reading from memory:

  ```
  read_content_from_memory
  [-content_in_hex]
  -instance_index <instance index>
  -start_address <starting address>
  -word_count <word count>
  ```

- Writing to memory:

  ```
  write_content_to_memory
  ```

- Save memory contents to file:

  ```
  save_content_from_memory_to_file
  ```

- Update memory contents from File:

  ```
  update_content_to_memory_from_file
  ```

For descriptions of the command options and scripting examples, refer to the Tcl API Help Browser and the *API Functions for Tcl* in Quartus II Help.

Programming the Device with the In-System Memory Content Editor

If you make changes to your design, you can program the device from within the In-System Memory Content Editor.

To program the device, refer to *Setting up the In-System Memory Content Editor* in Quartus II Help.

Example: Using the In-System Memory Content Editor with the SignalTap II Logic Analyzer

The following scenario describes how you can use the In-System Updating of Memory and Constants feature with the SignalTap II Logic Analyzer to efficiently debug your design in-system. You can use the In-System Memory Content Editor and the SignalTap II Logic Analyzer simultaneously with the JTAG interface.

Scenario: After completing your FPGA design, you find that the characteristics of your FIR filter design are not as expected.

1. To locate the source of the problem, change all your FIR filter coefficients to be in-system modifiable and instantiate the SignalTap II Logic Analyzer.
2. Using the SignalTap II Logic Analyzer to tap and trigger on internal design nodes, you find the FIR filter to be functioning outside of the expected cutoff frequency.
3. Using the **In-System Memory Content Editor**, you check the correctness of the FIR filter coefficients. Upon reading each coefficient, you discover that one of the coefficients is incorrect.
4. Because your coefficients are in-system modifiable, you update the coefficients with the correct data with the **In-System Memory Content Editor**.

In this scenario, you can quickly locate the source of the problem using both the In-System Memory Content Editor and the SignalTap II Logic Analyzer. You can also verify the functionality of your device by changing the coefficient values before modifying the design source files.

You can also modify the coefficients with the In-System Memory Content Editor to vary the characteristics of the FIR filter, for example, filter attenuation, transition bandwidth, cut-off frequency, and windowing function.

Conclusion

The In-System Updating of Memory and Constants feature provides access to a device for efficient debugging in a hardware lab. You can use the In-System Memory and Content Editor with the SignalTap II Logic Analyzer to maximize the visibility into an Altera FPGA. By maximizing visibility and access to internal logic of the device, you can identify and resolve problems with your design more easily.

Document Revision History

Table 15–1 shows the revision history of this chapter.

Table 15–1. Document Revision History

Date	Version	Changes
June 2012	12.0.0	Removed survey link.
November 2011	10.0.3	Template update.
December 2010	10.0.2	■ Changed to new document template. No change to content
August 2010	10.0.1	■ Corrected links
July 2010	10.0.0	■ Inserted links to Quartus II Help ■ Removed Reference Documents section
November 2009	9.1.0	■ Delete references to APEX devices ■ Style changes
March 2009	9.0.0	■ No change to content

Table 15–1. Document Revision History

November 2008	8.1.0	■ Changed to 8-1/2 x 11 page size. No change to content
May 2008	8.0.0	■ Added reference to Section V. In-System Debugging in volume 3 of the Quartus II Handbook on page 16-1 ■ Removed references to the Mercury device, as it is now considered to be a “Mature” device ■ Added links to referenced documents throughout document ■ Minor editorial updates

For previous versions of the *Quartus II Handbook*, refer to the Quartus II Handbook Archive.