12. Quick Design Debugging Using SignalProbe

QII53008-13.0.0

This chapter provides detailed instructions about how to use SignalProbe to quickly debug your design. The SignalProbe incremental routing feature helps reduce the hardware verification process and time-to-market for system-on-a-programmable-chip (SOPC) designs.

Easy access to internal device signals is important in the design or debugging process. The SignalProbe feature makes design verification more efficient by routing internal signals to I/O pins quickly without affecting the design. When you start with a fully routed design, you can select and route signals for debugging to either previously reserved or currently unused I/O pins.

The SignalProbe feature supports the Arria® series, Cyclone® series, MAX® II, and Stratix® series, device families.

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

Debugging Using the SignalProbe Feature

The SignalProbe feature allows you to reserve available pins and route internal signals to those reserved pins, while preserving the behavior of your design. SignalProbe is an effective debugging tool that provides visibility into your FPGA.

You can reserve pins for SignalProbe and assign I/O standards after a full compilation. Each SignalProbe-source to SignalProbe-pin connection is implemented as an engineering change order (ECO) that is applied to your netlist after a full compilation.

To route the internal signals to the device's reserved pins for SignalProbe, perform the following tasks:

1. Performing a Full Compilation, described on page 12–2.
2. Reserving SignalProbe Pins, described on page 12–2.
3. Assigning SignalProbe Sources, described on page 12–2.
4. Adding Registers Between Pipeline Paths and SignalProbe Pins, described on page 12–3.
5. Performing a SignalProbe Compilation, described on page 12–3.
6. Analyzing the Results of a SignalProbe Compilation, described on page 12–4.

© 2013 Altera Corporation. All rights reserved. ALTERA, ARRIA, CYCLONE, HARDCOPY, MAX, MEGACORE, NIOS, QUARTUS and STRATIX words and logos are trademarks of Altera Corporation and registered in the U.S. Patent and Trademark Office and in other countries. All other words and logos identified as trademarks or service marks are the property of their respective holders as described at www.altera.com/common/legal.html. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

ISO 9001:2008 Registered

Performing a Full Compilation

You must complete a full compilation to generate an internal netlist containing a list of internal nodes to probe.

To perform a full compilation, on the Processing menu, click **Start Compilation**.

Reserving SignalProbe Pins

SignalProbe pins can only be reserved after a full compilation. You can also probe any unused I/Os of the device. Assigning sources is a simple process after reserving SignalProbe pins. The sources for SignalProbe pins are the internal nodes and registers in the post-compilation netlist that you want to probe.

Although you can reserve SignalProbe pins using many features within the Quartus II software, including the Pin Planner and the Tcl interface, you should use the **SignalProbe Pins** dialog box to create and edit your SignalProbe pins.

For more information, refer to *About SignalProbe* in Quartus II Help.

Assigning SignalProbe Sources

A SignalProbe source can be any combinational node, register, or pin in your post-compilation netlist. To find a SignalProbe source, in the Node Finder, use the SignalProbe filter to remove all sources that cannot be probed. You might not be able to find a particular internal node because the node can be optimized away during synthesis, or the node cannot be routed to the SignalProbe pin. For example, you cannot probe nodes and registers within Gigabit transceivers in Stratix IV devices because there are no physical routes available to the pins.

To probe virtual I/O pins generated in low-level partitions in an incremental compilation flow, select the source of the logic that feeds the virtual pin as your SignalProbe source pin.

For more information, refer to *SignalProbe Pins Dialog Box* and *Add SignalProbe Pins Dialog Box* in Quartus II Help.

Because SignalProbe pins are implemented and routed as ECOs, turning the **SignalProbe enable** option on or off is the same as selecting **Apply Selected Change** or **Restore Selected Change** in the Change Manager window. (If the Change Manager window is not visible at the bottom of your screen, on the View menu, point to **Utility Windows** and click **Change Manager**.)

For more information about the Change Manager for the Chip Planner and Resource Property Editor, refer to the *Engineering Change Management with the Chip Planner* chapter in volume 2 of the *Quartus II Handbook*.

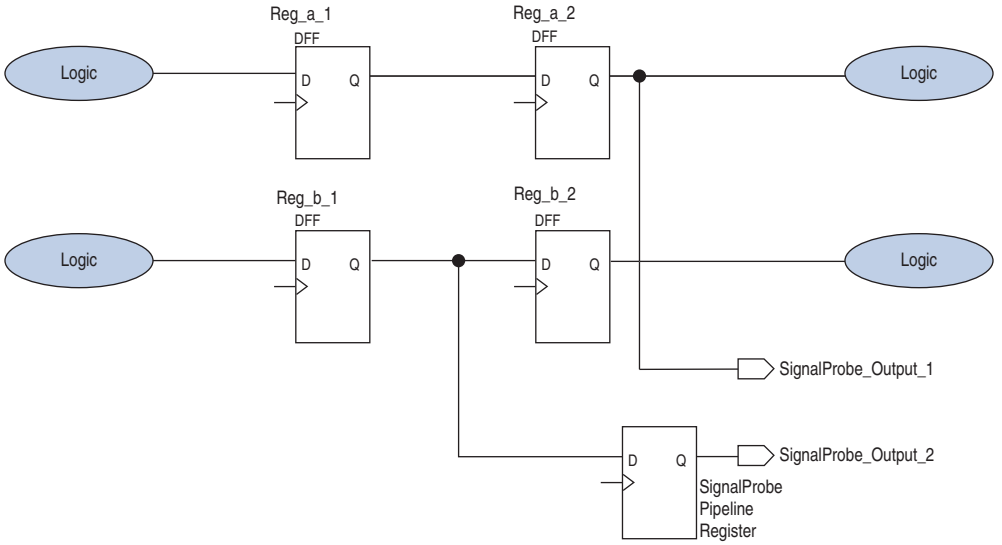
Adding Registers Between Pipeline Paths and SignalProbe Pins

You can specify the number of registers placed between a SignalProbe source and a SignalProbe pin. The registers synchronize data to a clock and control the latency of the SignalProbe outputs. The SignalProbe feature automatically inserts the number of registers specified into the SignalProbe path.

Figure 12–1 shows a single register between the SignalProbe source `Reg_b_1` and SignalProbe `SignalProbe_Output_2` output pin added to synchronize the data between the two SignalProbe output pins.

When you add a register to a SignalProbe pin, the SignalProbe compilation attempts to place the register to best meet timing requirements. You can place SignalProbe registers either near the SignalProbe source to meet f_{MAX} requirements, or near the I/O to meet t_{CO} requirements.

Figure 12–1. Synchronizing SignalProbe Outputs with a SignalProbe Register



To pipeline an existing SignalProbe connection, refer to *Add SignalProbe Pins Dialog Box* in Quartus II Help.

In addition to clock input for pipeline registers, you can also specify a reset signal pin for pipeline registers. To specify a reset pin for pipeline registers, use the Tcl command `make_sp`, as described in "Scripting Support" on page 12–6.

Performing a SignalProbe Compilation

Perform a SignalProbe compilation to route your SignalProbe pins. A SignalProbe compilation saves and checks all netlist changes without recompiling the other parts of the design. A SignalProbe compilation takes a fraction of the time of a full compilation to finish. The design's current placement and routing are preserved.

To perform a SignalProbe compilation, on the Processing menu, point to **Start** and click **Start SignalProbe Compilation**.

Analyzing the Results of a SignalProbe Compilation

After a SignalProbe compilation, the results are available in the compilation report file. Each SignalProbe pin is displayed in the **SignalProbe Fitting Result** page in the **Fitter** section of the Compilation Report. To view the status of each SignalProbe pin in the **SignalProbe Pins** dialog box, on the Tools menu, click **SignalProbe Pins**.

The status of each SignalProbe pin appears in the Change Manager window (Figure 12–2). (If the Change Manager window is not visible at the bottom of your GUI, from the View menu, point to **Utility Windows** and click **Change Manager**.)

Figure 12–2. Change Manager Window with SignalProbe Pins

For more information about how to use the Change Manager, refer to the *Engineering Change Management with the Chip Planner* chapter in volume 2 of the *Quartus II Handbook*.

To view the timing results of each successfully routed SignalProbe pin, on the Processing menu, point to **Start** and click **Start Timing Analysis**.

SignalProbe Compilation Functions

After a full compilation, you can start a SignalProbe compilation either manually or automatically. A SignalProbe compilation performs the following functions:

- Validates SignalProbe pins
- Validates your specified SignalProbe sources
- Adds registers into SignalProbe paths, if applicable
- Attempts to route from SignalProbe sources through registers to SignalProbe pins

To run the SignalProbe compilation immediately after a full compilation, on the Tools menu, click **SignalProbe Pins**. In the **SignalProbe Pins** dialog box, click **Start Check & Save All Netlist Changes**.

To run a SignalProbe compilation manually after a full compilation, on the Processing menu, point to **Start** and click **Start SignalProbe Compilation**.

You must run the Fitter before a SignalProbe compilation. The Fitter generates a list of all internal nodes that can serve as SignalProbe sources.

Turn the **SignalProbe enable** option on or off in the **SignalProbe Pins** dialog box to enable or disable each SignalProbe pin.

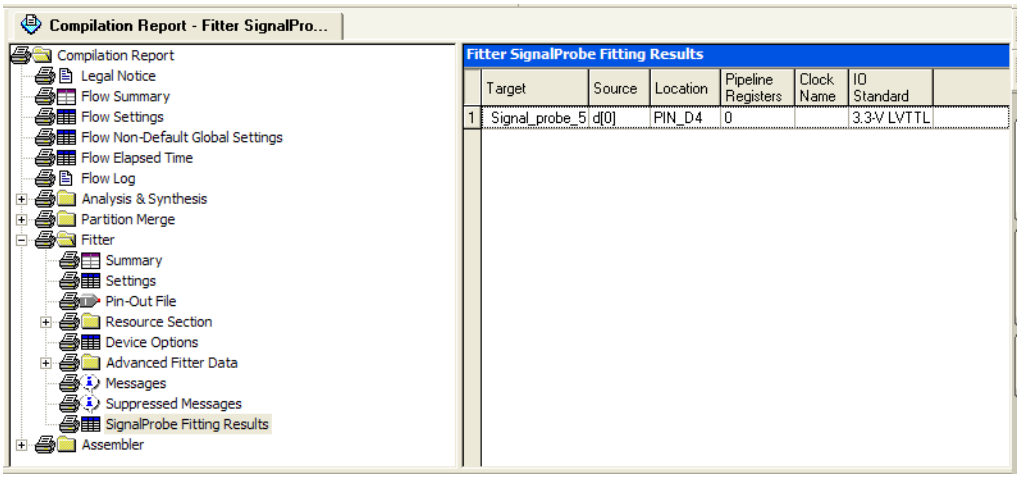
Understanding the Results of a SignalProbe Compilation

After a SignalProbe compilation, the results appear in two sections of the compilation report file. The fitting results and status (Table 12–1) of each SignalProbe pin appears in the **SignalProbe Fitting Result** screen in the Fitter section of the Compilation Report (Figure 12–3).

Table 12–1. Status Values

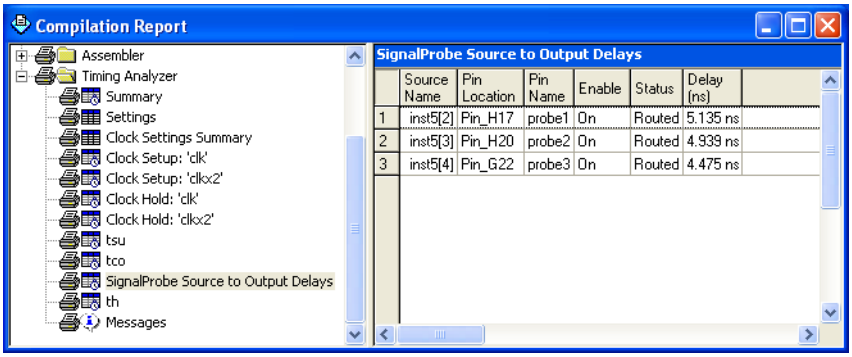
Status	Description
Routed	Connected and routed successfully
Not Routed	Not enabled
Failed to Route	Failed routing during last SignalProbe compilation
Need to Compile	Assignment changed since last SignalProbe compilation

Figure 12–3. SignalProbe Fitting Results Page in the Compilation Report Window



The **SignalProbe source to output delays** screen in the Timing Analysis section of the Compilation Report displays the timing results of each successfully routed SignalProbe pin (Figure 12–4).

Figure 12–4. SignalProbe Source to Output Delays Page in the Compilation Report Window



After a SignalProbe compilation, the processing screen of the Messages window also provides the results for each SignalProbe pin and displays slack information for each successfully routed SignalProbe pin.

Analyzing SignalProbe Routing Failures

A SignalProbe compilation can fail for any of the following reasons:

- **Route unavailable**—the SignalProbe compilation failed to find a route from the SignalProbe source to the SignalProbe pin because of routing congestion.
- **Invalid or nonexistent SignalProbe source**—you entered a SignalProbe source that does not exist or is invalid.
- **Unusable output pin**—the output pin selected is found to be unusable.

Routing failures can occur if the SignalProbe pin's I/O standard conflicts with other I/O standards in the same I/O bank.

If routing congestion prevents a successful SignalProbe compilation, you can allow the compiler to modify routing to the specified SignalProbe source. On the Tools menu, click **SignalProbe Pins** and turn on **Modify latest fitting results during SignalProbe compilation**. This setting allows the Fitter to modify existing routing channels used by your design.

Turning on **Modify latest fitting results during SignalProbe compilation** can change the performance of your design.

Scripting Support

You can also run some procedures 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 ↵
```

The Tcl commands in this section are part of the `::quartus::chip_planner` Quartus II Tcl API. Source or include the `::quartus::chip_planner` Tcl package in your scripts to make these commands available.

For more information about Tcl scripting, refer to the *Tcl Scripting* chapter in volume 2 of the *Quartus II Handbook*. For more information about all settings and constraints in the Quartus II software, refer to the *Quartus II Settings File Reference Manual*. For more information about command-line scripting, refer to the *Command-Line Scripting* chapter in volume 2 of the *Quartus II Handbook*.

Making a SignalProbe Pin

To make a SignalProbe pin, type the following command:

```
make_sp [-h | -help] [-long_help] [-clk <clk>] [-io_std <io_std>] \
-loc <loc> -pin_name <pin name> [-regs <regs>] [-reset <reset>] \
-src_name <source name>
```

Deleting a SignalProbe Pin

To delete a SignalProbe pin, type the following Tcl command:

```
delete_sp [-h | -help] [-long_help] -pin_name <pin name>
```

Enabling a SignalProbe Pin

To enable a SignalProbe pin, type the following Tcl command:

```
enable_sp [-h | -help] [-long_help] -pin_name <pin name>
```

Disabling a SignalProbe Pin

To disable a SignalProbe pin, type the following Tcl command:

```
disable_sp [-h | -help] [-long_help] -pin_name <pin name>
```

Performing a SignalProbe Compilation

To perform a SignalProbe compilation, type the following command:

```
quartus_sh --flow signalprobe <project name>
```

Script Example

Example 12–1 shows a script that creates a SignalProbe pin called `sp1` and connects the `sp1` pin to source node `reg1` in a project that was already compiled.

Example 12–1. Creating a SignalProbe Pin Called sp1

```
package require ::quartus::chip_planner
project_open project
read_netlist
make_sp -pin_name sp1 -src_name reg1
check_netlist_and_save
project_close
```

Reserving SignalProbe Pins

To reserve a SignalProbe pin, add the commands shown in Example 12–2 to the Quartus II Settings File (**.qsf**) for your project.

Example 12–2. Reserving a SignalProbe Pin

```
set_location_assignment <location> -to <SignalProbe pin name>
set_instance_assignment -name RESERVE_PIN \
"AS SIGNALPROBE OUTPUT" -to <SignalProbe pin name>
```

Valid locations are pin location names, such as `Pin_A3`.

For more information about reserving SignalProbe pins, refer to "Reserving SignalProbe Pins" on page 12–2.

Common Problems When Reserving a SignalProbe Pin

If you cannot reserve a SignalProbe pin in the Quartus II software, it is likely that one of the following is true:

- You have selected multiple pins.
- A compilation is running in the background. Wait until the compilation is complete before reserving the pin.
- You have the Quartus II Web Edition software, in which the SignalProbe feature is not enabled by default. You must turn on TalkBack to enable the SignalProbe feature in the Quartus II Web Edition software.
- You have not set the pin reserve type to **As Signal Probe Output**. To reserve a pin, on the Assignments menu, in the **Assign Pins** dialog box, select **As SignalProbe Output**.
- The pin is reserved from a previous compilation. During a compilation, the Quartus II software reserves each pin on the targeted device. If you end the Quartus II process during a compilation, for example, with the **Windows Task Manager End Process** command or the UNIX `kill` command, perform a full recompilation before reserving pins as SignalProbe outputs.
- The pin does not support the SignalProbe feature. Select another pin.
- The current device family does not support the SignalProbe feature.

Adding SignalProbe Sources

Use the following Tcl commands to add SignalProbe sources.

To assign the node name to a SignalProbe pin, type the following Tcl command:

```
set_instance_assignment -name SIGNALPROBE_SOURCE <node name> -to \
<SignalProbe pin name>
```

The next command turns on SignalProbe routing. To turn off individual SignalProbe pins, specify `OFF` instead of `ON` with the following command:

```
set_instance_assignment -name SIGNALPROBE_ENABLE ON -to \
<SignalProbe pin name>
```

For more information about adding SignalProbe sources, refer to *SignalProbe Pins Dialog Box* and *Add SignalProbe Pins Dialog Box* in Quartus II Help.

Assigning I/O Standards

To assign an I/O standard to a pin, type the following Tcl command:

```
set_instance_assignment -name IO_STANDARD <I/O standard> -to \
<SignalProbe pin name>
```

For a list of valid I/O standards, refer *I/O Standards* to the in the Quartus II Help.

Adding Registers for Pipelining

To add registers for pipelining, type the following Tcl command:

```
set_instance_assignment -name SIGNALPROBE_CLOCK <clock name> -to \
<SignalProbe pin name>

set_instance_assignment \
-name SIGNALPROBE_NUM_REGISTERS <number of registers> -to \
<SignalProbe pin name>
```

Running SignalProbe Immediately After a Full Compilation

To run SignalProbe immediately after a full compilation, type the following Tcl command:

```
set_global_assignment -name SIGNALPROBE_DURING_NORMAL_COMPILATION ON
```

For more information about running SignalProbe automatically, refer to "SignalProbe Compilation Functions" on page 12–4.

Running SignalProbe Manually

To run SignalProbe as part of a scripted flow using Tcl, use the following in your script:

```
execute_flow -signalprobe
```

To perform a Signal Probe compilation interactively at a command prompt, type the following command:

```
quartus_sh_fit --flow signalprobe <project name>
```

For more information about running SignalProbe manually, refer to "SignalProbe Compilation Functions" on page 12–4.

Enabling or Disabling All SignalProbe Routing

Use the Tcl command in Example 12–3 to turn on or turn off SignalProbe routing. When using this command, to turn SignalProbe routing on, specify `ON`. To turn SignalProbe routing off, specify `OFF`.

Example 12–3. Turning SignalProbe On or Off with Tcl Commands

```
set spe [get_all_assignments -name SIGNALPROBE_ENABLE] \
foreach_in_collection asgn $spe {
   set signalprobe_pin_name [lindex $asgn 2]
   set_instance_assignment -name SIGNALPROBE_ENABLE -to \
$signalprobe_pin_name <ON|OFF> }
```

For more information about enabling or disabling SignalProbe routing, refer to page 12–4.

Allowing SignalProbe to Modify Fitting Results

To turn on **Modify latest fitting results**, type the following Tcl command:

```
set_global_assignment -name SIGNALPROBE_ALLOW_OVERUSE ON
```

For more information, refer to "Analyzing SignalProbe Routing Failures" on page 12–6.

Document Revision History

Table 12–2 shows the revision history for this chapter.

Table 12–2. Document Revision History

Date	Version	Changes
May 2013	13.0.0	Changed sequence of flow to clarify that you need to perform a full compilation before reserving SignalProbe pins. Affected sections are "Debugging Using the SignalProbe Feature" on page 12–1 and "Reserving SignalProbe Pins" on page 12–2. Moved "Performing a Full Compilation" on page 12–2 before "Reserving SignalProbe Pins" on page 12–2.
June 2012	12.0.0	Removed survey link.
November 2011	10.0.2	Template update.
December 2010	10.0.1	Changed to new document template.
July 2010	10.0.0	■ Revised for new UI. ■ Removed section SignalProbe ECO flows ■ Removed support for SignalProbe pin preservation when recompiling with incremental compilation turned on. ■ Removed outdated FAQ section. ■ Added links to Quartus II Help for procedural content.
November 2009	9.1.0	■ Removed all references and procedures for APEX devices. ■ Style changes.
March 2009	9.0.0	■ Removed the "Generate the Programming File" section ■ Removed unnecessary screenshots ■ Minor editorial updates
November 2008	8.1.0	■ Modified description for preserving SignalProbe connections when using Incremental Compilation ■ Added plausible scenarios where SignalProbe connections are not reserved in the design
May 2008	8.0.0	■ Added "Arria GX" to the list of supported devices ■ Removed the "On-Chip Debugging Tool Comparison" and replaced with a reference to the Section V Overview on page 13–1 ■ Added hyperlinks to referenced documents throughout the chapter ■ Minor editorial updates

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