

This chapter provides detailed instructions about how to use SignalProbe to quickly debug your design.

## Introduction

Hardware verification can be a lengthy and expensive process. 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 is fully functional with Arria® GX, Stratix® series, Cyclone® series, and MAX® II, device families.

If you are using the SignalProbe feature to debug your Stratix series, Cyclone series, or MAX II device, refer to [“Debugging Using the SignalProbe Feature”](#).



The Quartus® II software provides a portfolio of on-chip debugging solutions. For an overview and comparison of all of the tools available in the Quartus II software on-chip debugging tool suite, refer to [Section V. In-System Design Debugging](#) 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.



This section describes the SignalProbe process for the Stratix series, Cyclone series, and MAX II device families.

You can reserve pins for SignalProbe and assign I/O standards before or after a full compilation. Each SignalProbe-source to SignalProbe-pin connection is implemented as an ECO change 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. [Reserve the SignalProbe Pins](#), described on [page 14–2](#).
2. [Perform a Full Compilation](#), described on [page 14–3](#).
3. [Assign a SignalProbe Source](#), described on [page 14–3](#).
4. [Add Registers to the Pipeline Path to SignalProbe Pin](#), described on [page 14–4](#).
5. [Perform a SignalProbe Compilation](#), described on [page 14–5](#).

- Analyze the Results of the SignalProbe Compilation, described on page 14-5.

## Reserve the SignalProbe Pins

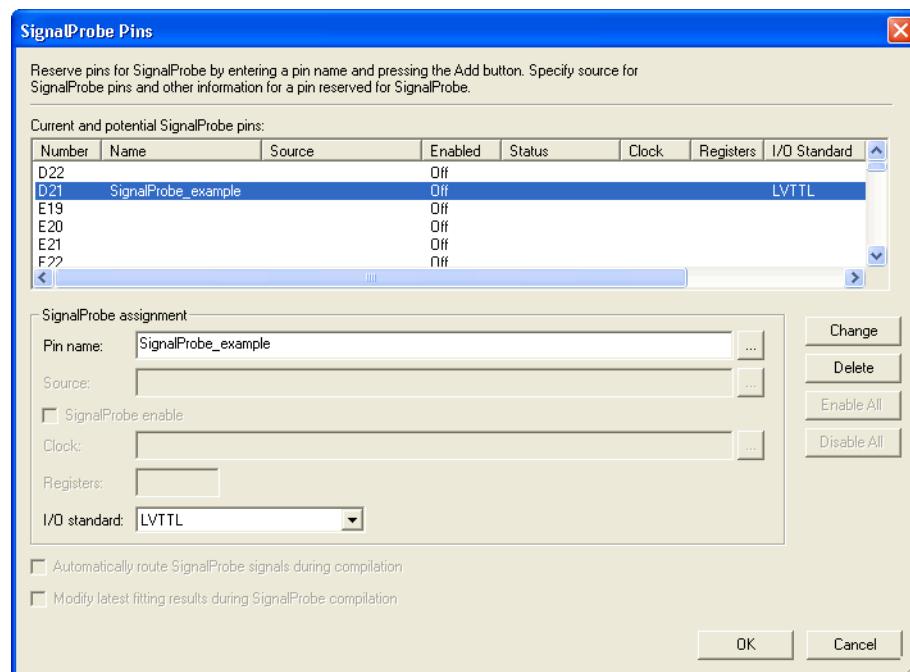
SignalProbe pins can be reserved before or after compiling your design. Reserving SignalProbe pins before a compilation is optional. You can also reserve any unused I/Os of the device for SignalProbe pins after compilation. Assigning sources is a simple process after reserving SignalProbe pins. The sources 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.

To reserve an available package pin as a SignalProbe pin using the **SignalProbe Pins** dialog box, perform the following steps:

- On the Tools menu, click **SignalProbe Pins**. The **SignalProbe Pins** dialog box appears (Figure 14-1). The Pin name and I/O Standard appear as the only fields that are editable if place-and-route or fitting have not been performed.

**Figure 14-1.** Reserving a SignalProbe Pin in the SignalProbe Pins Dialog Box



- In the **Current and potential SignalProbe pins** list, click a pin from the **Number** column and type your SignalProbe pin name in the **Pin name** box.
- Select an I/O standard from the **I/O standard** list.
- To add a new SignalProbe pin, click **Add**. To edit or change a previously reserved pin for SignalProbe, click **Change**. (Figure 14-1 shows how to use the the dialog box to edit a previously reserved pin; if you were adding a new SignalProbe pin, the **Add** button appears instead of the **Change** button.)

5. Click **OK**.

## Perform a Full Compilation

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

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

## Assign a SignalProbe Source

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, as it is untappable. For example, internal nodes and registers within the Gigabit transceivers cannot be probed because there are no physical routes to the pins available.



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.

To assign a SignalProbe source to your SignalProbe reserved pin, perform the following steps:

1. On the Tools menu, click **SignalProbe Pins**. The **SignalProbe Pins** dialog box appears (Figure 14-1 on page 14-2).
2. If a SignalProbe reserved pin is shown, in the **Current and potential SignalProbe pins** list, click the pin. Alternately, you can click an available pin number in the **Current and potential SignalProbe pins** list and type a new SignalProbe pin name in the **Pin name** box.
3. In the **Source** box, specify the source name. Click the browse button. The **Node Finder** dialog box appears.
4. When you open the **Node Finder** dialog box from the **SignalProbe Pins** dialog box, **SignalProbe** is selected by default in the **Filter** list. To show a set of nodes that can be probed in the **Nodes Found** list, click **List**.
5. In the **Nodes Found** list, select your source node and click the **>** button. The selected node appears in the **Selected Nodes** list.
6. Click **OK**.
7. After a source is selected, the **SignalProbe enable** option is turned on. Click **Change** or **Add** to accept the changes.



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*.

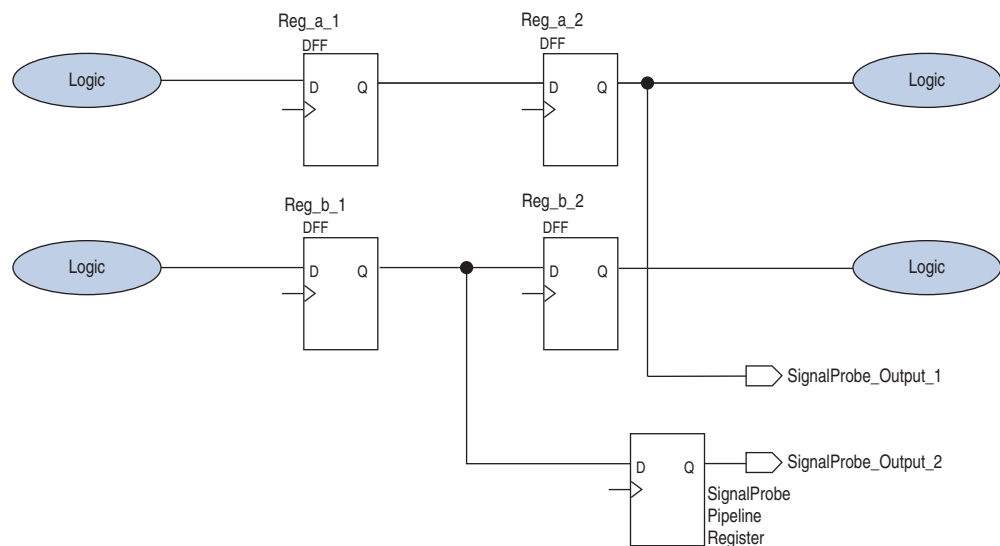
## Add Registers to the Pipeline Path to SignalProbe Pin

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

Figure 14-2 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 fit 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 14-2.** Synchronizing SignalProbe Outputs with a SignalProbe Register



To pipeline an existing SignalProbe connection, perform the following steps:

1. On the Tools menu, click **SignalProbe Pins**. The **SignalProbe Pins** dialog box appears.
2. Select a SignalProbe pin and in the **Clock** dialog box, type the clock name used to drive your registers, or click the browse button to use the Node Finder to select your clock source.
3. In the **Registers** dialog box, specify the number of registers you want to add in between the SignalProbe source and the SignalProbe output.
4. Click **Change**.
5. Click **OK**.

 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 14–11.

## Perform 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 and completes compilation in a fraction of the time of a full compilation. 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**.

## Analyze the Results of the 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 14–3). (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 14–3.** Change Manager Window with SignalProbe Pins



Index	Node Name	Change Type	Old Value	Target Value	Current Value	Disk Value
1	signalprobe_1	SignalProbe	Disconnected	/f/itref/state_m_inst1/itref/idle	/f/itref/state_m_inst1/itref/idle	/f/itref/state_m_inst1/itref/idle
2	signalprobe_2	SignalProbe	Disconnected	/f/itref/state_m_inst1/itref/tap1	/f/itref/state_m_inst1/itref/tap1	/f/itref/state_m_inst1/itref/tap1
3	signalprobe_3	SignalProbe	Disconnected	/f/itref/state_m_inst1/itref/tap2	/f/itref/state_m_inst1/itref/tap2	/f/itref/state_m_inst1/itref/tap2
4	signalprobe_4	SignalProbe	Disconnected	/f/itref/state_m_inst1/itref/tap3	/f/itref/state_m_inst1/itref/tap3	/f/itref/state_m_inst1/itref/tap3
5	signalprobe_5	SignalProbe	Disconnected	/f/itref/state_m_inst1/itref/tap4	/f/itref/state_m_inst1/itref/tap4	/f/itref/state_m_inst1/itref/tap4

 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 ECO Flows

SignalProbe pins are implemented using the same flow as other post-compilation changes made as ECOs. The following section describes SignalProbe ECO flows with and without the Quartus II incremental compilation feature.

## SignalProbe ECO Flow with Quartus II Incremental Compilation

The incremental compilation feature is turned on by default. The top-level design is automatically set to a design partition when the incremental compilation feature is on. A design partition during incremental compilation can have different netlist types. (Netlist types can be set to source HDL, post synthesis, or post-fit.) The netlist type indicates whether that partition should be resynthesized or refit during Quartus II incremental compilation. Incremental compilation saves you time and preserves the placement of unchanged partitions in your design if small changes must be made to some partitions late in the design cycle.



For more information about the Quartus II incremental compilation feature, refer to the *Quartus II Incremental Compilation Feature for Hierarchical and Team-Based Design* chapter in volume 1 of the *Quartus II Handbook*.

The behavior of SignalProbe pins during an incremental compilation depends on the Netlist Type setting. When the top-level partition netlist type is set to **post-fit**, SignalProbe ECOs are retained if the partition being probed is preserved when you recompile the design.

SignalProbe connections always link the partition being probed with the top-level partition. As such, a SignalProbe connection might change the preservation attributes in a lower-level partition. This is known as *partition linking*. When partition linking occurs, all partitions that become linked share the attribute for the preservation level that is the strictest among all of the affected partitions. As a result, when you tap any partitions that are not post-fit and the top level is set to a netlist type of post-fit, your SignalProbe connection is preserved.

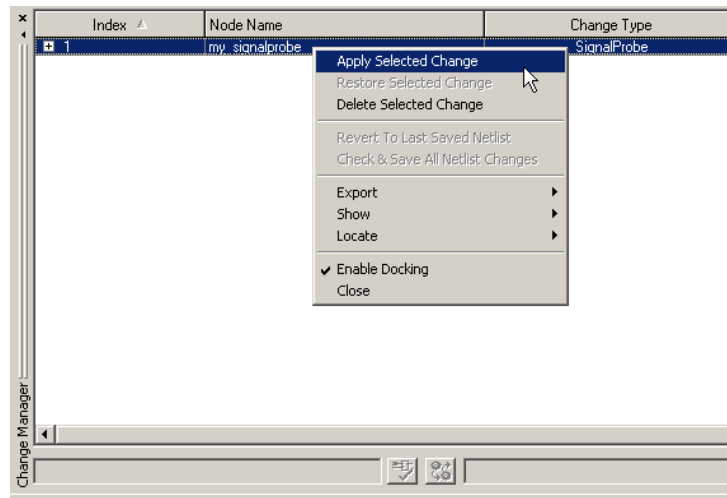
The behavior is different in the case that your top-level partition netlist type is set to **post-synthesis** and you have no other lower-level partitions defined. In this case, the partition with the strictest preservation type is set to **post-synthesis**. If you create SignalProbe ECOs and recompile the design, your SignalProbe ECOs are not retained and a warning message appears in the Messages window. The warning indicates that ECO modifications are discarded; however, all of the ECO information is retained in the Change Manager. In this case, apply SignalProbe ECOs from the Change Manager and perform the **Check and Save All Netlist Changes** step, as described in *“SignalProbe ECO Flow Without Quartus Incremental Compilation”* on page 14-6.

## SignalProbe ECO Flow Without Quartus Incremental Compilation

If you do not use the Quartus II incremental compilation feature and you implement SignalProbe pins after the initial compilation of your design, SignalProbe ECOs are not retained during recompilation. However, all of the SignalProbe ECOs remain in the Change Manager.

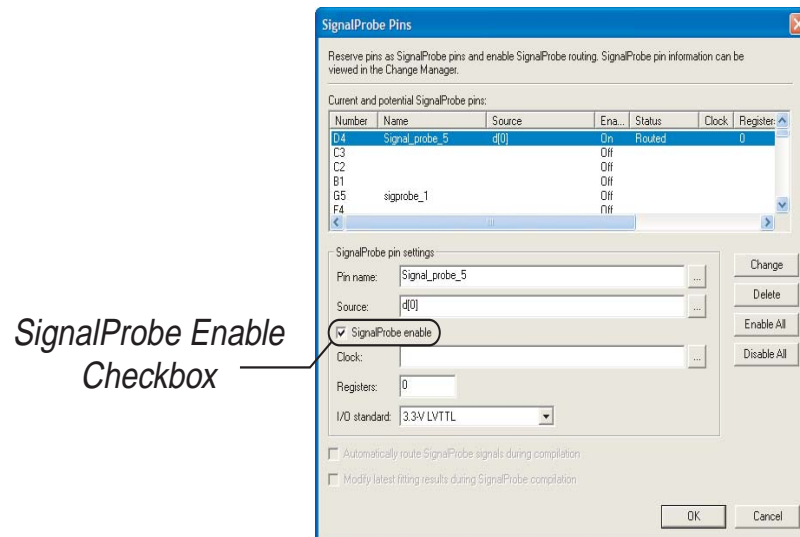
To apply a SignalProbe ECO, right-click in the Change Manager and select **Apply Selected Change** (Figure 14-4). (If the Change Manager window is not visible at the bottom of your screen, from the View menu, point to **Utility Windows** and click **Change Manager**.)

**Figure 14-4.** Applying SignalProbe ECOs



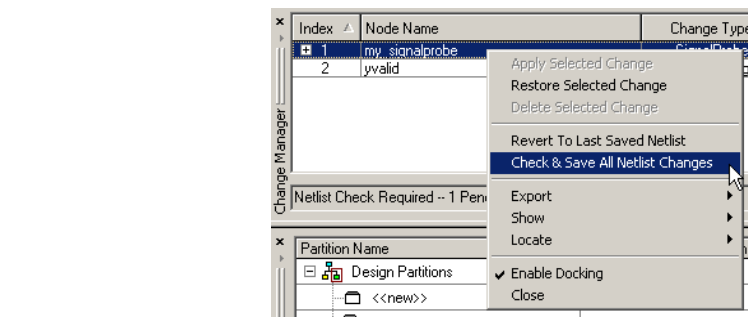
Alternately, you can use the **SignalProbe Pins** dialog box to enable the ECOs, as shown in [Figure 14-5](#). This has the same effect as applying the SignalProbe ECOs within the Change Manager.

**Figure 14-5.** Enabling ECOs in the SignalProbe Pins Dialog Box



After applying the selected SignalProbe ECO, either right-click anywhere in the Change Manager and select **Check and Save All Netlist Changes** ([Figure 14-6](#)), or, on the Processing menu, point to **Start** and click **Start Check and Save All Netlist Changes** to perform the ECO compilation.

Figure 14-6. Check and Save All Netlist Changes



## Common Questions About the SignalProbe Feature

The following are answers to common questions about the SignalProbe feature.

### Why Did I Get the Following Error Message, “Error: There are No Enabled SignalProbes to Process”?

This error message is generated when a SignalProbe compilation was attempted with either no SignalProbe pins to route, or with all SignalProbe pins disabled.

This might occur if you perform a SignalProbe compilation after a full compilation. For example, when a full compilation is performed, all SignalProbe pins are disabled. You can create or re-enable your SignalProbe pins in the **SignalProbe Pins** dialog box.

### How Can I Retain My SignalProbe ECOs During Re-Compilation of My Design?

To retain your existing ECOs during recompilation of your design, you must use Quartus II incremental compilation. To learn more about the flow, refer to [“SignalProbe ECO Flow with Quartus II Incremental Compilation”](#) on page 14-6.

### Why Did My SignalProbe Source Disappear in the Change Manager?

The SignalProbe source information for each SignalProbe connection is stored in the project database (**db** directory). SignalProbe pins are post-compilation changes to your netlist and are interpreted as ECOs. These changes are stored in the project **db** and if the project database is removed, the SignalProbe source information is lost and does not appear in the **SignalProbe Pins** dialog box. To restore your SignalProbe pins after the design compilation step, source the `signalprobe_qlsf.tcl` script located in your project directory.

To restore your SignalProbe source information after compilation, type the following command from a command-line prompt:


```
quartus_cdb -t signalprobe_qlsf.tcl ↵
```



Before typing this command, you must close your design project. When the command finishes, you can open your design project again. The Change Manager shows the sources for SignalProbe pins.

### What is an ECO and Where Can I Find More Information about ECOs?

ECOs are late design cycle changes made to your design that do not alter functionality and timing.

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

### How Do I Migrate My Previous SignalProbe Assignments in the Quartus II Software Version 5.1 and Earlier to Version 6.0 and Later?

In earlier versions of the Quartus II software, SignalProbe pins were stored in the Quartus II Settings File (.qsf). These assignments are automatically converted into ECO changes when you open the **SignalProbe** dialog box or when you start a SignalProbe compilation in the Quartus II software versions 6.0 and higher.

For example, the SignalProbe source assignment from a .qsf file is removed and added to the Change Manager as an ECO after the **SignalProbe** dialog box is opened, or when you perform a SignalProbe compilation. [Example 14-1](#) shows SignalProbe assignments in the .qsf file. [Example 14-2](#) shows the same assignments after opening the **SignalProbe Pins** dialog box.

#### Example 14-1. SignalProbe Assignments in the Quartus II Settings File

---

```
set_location_assignment PIN_C22 -to my_signalprobe_pin
set_instance_assignment -name RESERVE_PIN "AS SIGNALPROBE OUTPUT" -to my_signalprobe_pin
set_instance_assignment -name IO_STANDARD LVTTTL -to my_signalprobe_pin
set_instance_assignment -name SIGNALPROBE_ENABLE ON -to my_signalprobe_pin
set_instance_assignment -name SIGNALPROBE_SOURCE inst5[0] -to my_signalprobe_pin
```

---

#### Example 14-2. SignalProbe Assignments in the Quartus II Settings File after Opening the SignalProbe Pins Dialog Box


---

```
set_location_assignment PIN_C22 -to my_signalprobe_pin
set_instance_assignment -name RESERVE_PIN "AS SIGNALPROBE OUTPUT" -to my_signalprobe_pin
set_instance_assignment -name IO_STANDARD LVTTTL -to my_signalprobe_pin
set_instance_assignment -name SIGNALPROBE_ENABLE ON -to my_signalprobe_pin
```

---

### What are all the Changes for the SignalProbe Feature between the Quartus II Software Version 5.1 and Earlier, and Version 6.0 and Later?

The following list highlights the changes that affect users of the SignalProbe feature in the Quartus II software versions 5.1 and earlier. This applies to Stratix series, Cyclone series, and MAX II device families.

 For more information about the changes that pertain to each release of the Quartus II software, refer to the [Release Notes](#) on the Altera website ([www.altera.com](http://www.altera.com)).

- In Quartus II software versions 5.1 and earlier, the **SignalProbe Pins** dialog box was accessed on the Assignments menu. To access it with the Quartus II software version 6.0 and later, on the Tools menu, click **SignalProbe Pins**.
- A full compilation is required before making SignalProbe connections. However, you can still reserve pins before compilation for later use by SignalProbe. You can reserve pins by creating a SignalProbe in the **SignalProbe** dialog box without specifying a source. This is the same behavior as in the Quartus II software version 5.1.

- To route the SignalProbe pins, you must perform a SignalProbe compilation after a full compilation. The **Automatically route SignalProbe signals during compilations** and **Modify latest fitting results during SignalProbe compilation** options are no longer supported.
- After subsequent compiles, full or incremental, existing SignalProbe pins are disabled and are not present in the post-compilation netlist. To add them back, enable the SignalProbe pins and perform a SignalProbe compilation.
- SignalProbe pins are not controlled via assignments in the .qsf file because they are now ECOs. Existing .qsf files automatically convert to ECOs when a SignalProbe compilation is performed or when the **SignalProbe** dialog box is opened.
- The Tcl interface for creating SignalProbe pins has improved and is a part of the Chip Planner package `::quartus::chip_editor`. Refer to [“Scripting Support” on page 14-11](#).
- Previously, the `quartus_fit --signalprobe` command was used to perform a SignalProbe compilation. This is not supported in the Quartus II software version 6.0 and later, and is replaced by the improved Tcl interface and the `check_netlist_and_save` Tcl command.
- The SignalProbe timing report generated after a successful SignalProbe compilation is not available in the Quartus II software version 6.0 and later. You can view the timing results of your SignalProbe pins in the SignalProbe Fitting Results, under the Fitter report, or in the  $t_{CO}$  results page of the Timing report.
- You cannot make SignalProbe pins in the Assignment Editor. Use the **SignalProbe Pins** dialog box to make and edit your SignalProbe pins.

### Why Can't I Reserve 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 compile 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 family does not support the SignalProbe feature.

## Scripting Support

Running procedures and make settings using a Tcl script are described in this chapter. 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 *Scripting Reference Manual* includes the same information in PDF format.



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*.

### Make 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> ←
```

### Delete a SignalProbe Pin

To delete a SignalProbe pin, type the following command:

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

### Enable a SignalProbe Pin

To enable a SignalProbe pin, type the following command:

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

### Disable a SignalProbe Pin

To disable a SignalProbe pin, type the following command:

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

### Perform a SignalProbe Compilation

To perform a SignalProbe compilation, type the following command:

```
check_netlist_and_save ←
```

### Migrate Previous SignalProbe Pins to the Quartus II Software Versions 6.0 and Later

To migrate previous SignalProbe pins to the Quartus II software versions 6.0 and later, type the following command:

```
convert_signal_probes ←
```

### Script Example

**Example 14-3** 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 14-3.** Creating a SignalProbe Pin Called sp1

```

Package require ::quartus::chip_editor
Project_open project
Read_netlist
Make_sp -pin_name sp1 -src_name reg1
Check_netlist_and_save
Project_close

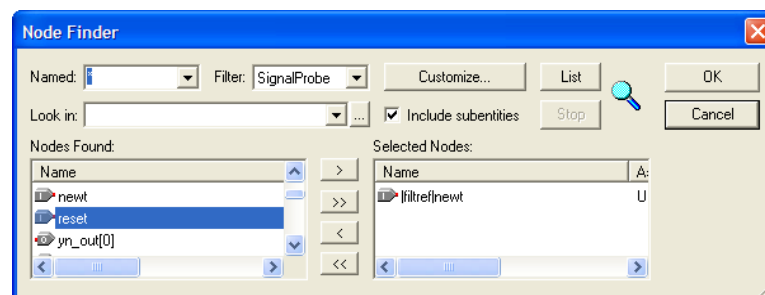
```

**Adding SignalProbe Sources**

A SignalProbe source is a signal in the post-compilation design database with a possible route to an output pin. To assign a SignalProbe source to a SignalProbe pin or an unused output pin, perform the following steps:

1. On the Tools menu, click **SignalProbe Pins**. The **SignalProbe Pins** dialog box appears.
2. In the **Current and potential SignalProbe pins** list, select the SignalProbe pin to which you want to add a SignalProbe source.
3. Click Browse and select a SignalProbe source.
4. Click **OK**.
5. In the **Assign SignalProbe Pins** dialog box, if a source has not been assigned to the SignalProbe pin, click **Add**. If a SignalProbe pin has been already assigned, click **Change**.
6. Click **OK**.

The **Node Finder** dialog box appears with the SignalProbe filter selected (Figure 14-7). Click **List** to view all of the available SignalProbe sources. If you cannot find a specific node with the SignalProbe filter, the node has either been removed by the Quartus II software during optimization or placed in the device where there are no possible routes to a pin.

**Figure 14-7.** Available SignalProbe Sources in the Node Finder

When the source of the SignalProbe pin is added or changed, the SignalProbe pin is automatically enabled. To disable a SignalProbe pin, turn off SignalProbe **enable**.

## Performing a SignalProbe Compilation

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
- If applicable, adds registers into SignalProbe paths
- Attempts to route from SignalProbe sources through registers to SignalProbe pins

To run the SignalProbe compilation automatically after a full compilation, on the Tools menu, click **SignalProbe Pins**. In the **SignalProbe Pins** dialog box, turn on **Automatically route SignalProbe signals during compilation**.

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 be used as SignalProbe sources.

To enable or disable each SignalProbe pin, in the **SignalProbe Pins** dialog box, turn the **SignalProbe enable** option on or off.

### Running SignalProbe with Smart Compilation

Running a smart compilation reduces compilation time by running only necessary modules during compilation. However, a full compilation is required if any design files, Analysis and Synthesis settings, or Fitter settings have changed.

To turn on smart compilation, on the Assignments menu, click **Settings**. In the **Category** list, select **Compilation Process Settings** and turn on **Use Smart compilation**.

If you run a SignalProbe compilation with smart compilation turned on, and there are changes to a design file or settings related to the Analysis and Synthesis or Fitter modules, the following message is displayed:

```
Error: Can't perform SignalProbe compilation because design requires a full compilation.
```



You should turn smart compilation on, which allows you to work with the latest settings and design files.

## Understanding the Results of a SignalProbe Compilation

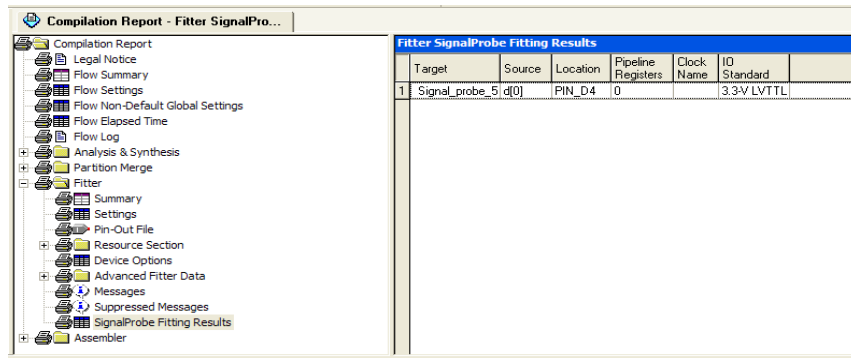
After a SignalProbe compilation, the results appear in two sections of the compilation report file. The fitting results and status (Table 14-1) of each SignalProbe pin is displayed in the **SignalProbe Fitting Result** screen in the Fitter section of the Compilation Report (Figure 14-8).

**Table 14-1.** Status Values (Part 1 of 2)

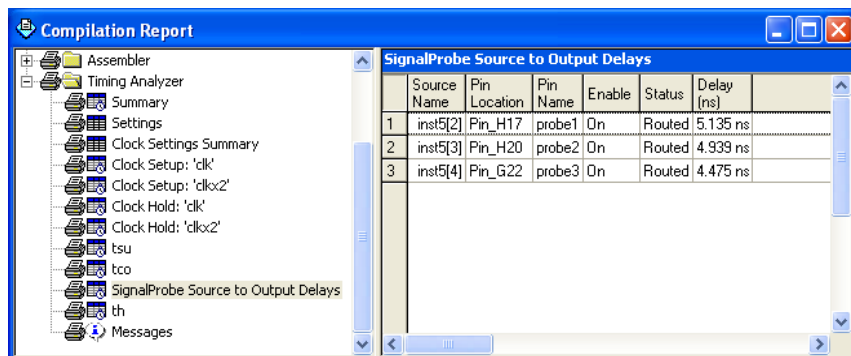
Status	Description
Routed	Connected and routed successfully
Not Routed	Not enabled

**Table 14-1.** Status Values (Part 2 of 2)

Status	Description
Failed to Route	Failed routing during last SignalProbe compilation
Need to Compile	Assignment changed since last SignalProbe compilation

**Figure 14-8.** SignalProbe Fitting Results Page in the Compilation Report Window

The timing results of each successfully routed SignalProbe pin is displayed in the **SignalProbe source to output delays** screen in the Timing Analysis section of the Compilation Report (Figure 14-9).

**Figure 14-9.** 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 of each SignalProbe pin and displays slack information for each successfully routed SignalProbe pin.

### Analyzing SignalProbe Routing Failures


The SignalProbe can begin compilation; however, one of the following reasons can prevent complete compilation:

- **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.

## SignalProbe Scripting Support

Running procedures and making settings using a Tcl script are described in this chapter. 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 *Scripting Reference Manual* includes the same information in PDF format.

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

## Reserving SignalProbe Pins

To reserve a SignalProbe pin, type the commands shown in [Example 14-4](#).

### Example 14-4. 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 [“Reserve the SignalProbe Pins” on page 14-2](#).

## Adding SignalProbe Sources

Use the following Tcl commands to add SignalProbe sources. For more information about adding SignalProbe sources, refer to [“Adding SignalProbe Sources” on page 14-12](#).

To assign the node name to a SignalProbe pin, type the following 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> ←
```

## 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 to the I/O Standards general description in the Quartus II Help.

## Adding Registers for Pipelining

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

```
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> ←
```

## Run SignalProbe Automatically

To run SignalProbe automatically after a full compile, type the following Tcl command:

```
set_global_assignment -name SIGNALPROBE_DURING_NORMAL_COMPILATION ON ←
```

For more information about running SignalProbe automatically, refer to [“Performing a SignalProbe Compilation” on page 14-13](#).

## Run SignalProbe Manually

To run SignalProbe manually with a Tcl command or the `quartus_fit` command, type the following at a command prompt.

```
execute_flow -signalprobe ←
```

The `execute_flow` command is in the flow package. At a command prompt, type the following command:

```
quartus_fit <project name> --signalprobe ←
```

For more information about running SignalProbe manually, refer to [“Performing a SignalProbe Compilation” on page 14-13](#).

## Enable or Disable All SignalProbe Routing

Use the Tcl command in [Example 14-5](#) 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 14-5. 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 14-13](#).

## Running SignalProbe with Smart Compilation

To turn on **Smart Compilation**, type the following Tcl command:

```
set_global_assignment -name SMART_RECOMPILE ON ←
```

For more information, refer to [“Running SignalProbe with Smart Compilation”](#) on [page 14-13](#).

## Allow 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 14-14](#).

## Conclusion

Using the SignalProbe feature can significantly reduce the time required compared to a full recompilation. Use the SignalProbe feature for quick access to internal design signals to perform system-level debugging.

## Referenced Documents

This chapter references the following documents:

- [Command-Line Scripting](#) chapter in volume 2 of the *Quartus II Handbook*
- [Engineering Change Management with the Chip Planner](#) chapter in volume 2 of the *Quartus II Handbook*
- [Release Notes](#) on the Altera website ([www.altera.com](http://www.altera.com))
- [Section V. In-System Design Debugging](#) in volume 3 of the *Quartus II Handbook*
- [Quartus II Incremental Compilation for Hierarchical and Team-Based Design](#) chapter in volume 1 of the *Quartus II Handbook*
- [Quartus II Settings File Reference Manual](#)

- *Tcl Scripting* chapter in volume 2 of the *Quartus II Handbook*

## Document Revision History

Table 14-2 shows the revision history for this chapter.

**Table 14-2.** Document Revision History

Date and Document Version	Changes Made	Summary of Changes
November 2009 v.9.1.0	<ul style="list-style-type: none"> <li>■ Removed all references and procedures for APEX devices.</li> <li>■ Style changes.</li> </ul>	Updated for the Quartus II software version 9.1 release.
March 2009 v9.0.0	<ul style="list-style-type: none"> <li>■ Removed the “Generate the Programming File” section</li> <li>■ Removed unnecessary screenshots</li> <li>■ Minor editorial updates</li> </ul>	Updated for the Quartus II software version 9.0 release.
November 2008 v8.1.0	<ul style="list-style-type: none"> <li>■ Modified description for preserving SignalProbe connections when using Incremental Compilation</li> <li>■ Added plausible scenarios where SignalProbe connections are not reserved in the design</li> </ul>	Updated for the Quartus II software version 8.1 release.
May 2008 v8.0.0	<ul style="list-style-type: none"> <li>■ Added “Arria GX” to the list of supported devices</li> <li>■ Removed the “On-Chip Debugging Tool Comparison” and replaced with a reference to the Section V Overview on page 13-1</li> <li>■ Added hyperlinks to referenced documents throughout the chapter</li> <li>■ Minor editorial updates</li> </ul>	Organizational changes for the Quartus II software version 8.0 release.



For previous versions of the *Quartus II Handbook*, refer to the [Quartus II Handbook Archive](#).