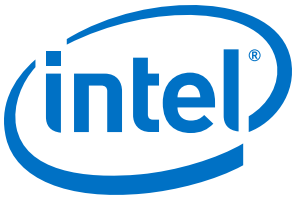


AN 799: Quick Intel® Arria® 10 Design Debugging Using Signal Probe and Rapid Recompile

Updated for Intel® Quartus® Prime Design Suite: **18.0**

Subscribe
Send Feedback

AN-799 | 2018.08.16
Latest document on the web: PDF | HTML

Contents

AN 799: Quick Intel® Arria® 10 Design Debugging Using Signal Probe and Rapid Recompile

This application note showcases a debugging technique that provides easy access to internal device signals without affecting the design.

Intel® Quartus® Prime Pro Edition Signal Probe feature allows you to route an internal node to a top-level I/O. 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.

During Rapid Recompile, the Compiler reuses previous synthesis and fitting results whenever possible, and does not reprocess unchanged design blocks. When you make small design changes, using Rapid Recompile reduces timing variations and the total recompilation time.

Related Information

System Debugging Tools Overview
In *Debug Tools User Guide: Intel Quartus Prime Pro Edition*

Setup Requirements

This demonstration requires Intel Quartus Prime Pro Edition software version 18.0.

Note: This document assumes familiarity with basic project compilation in Intel Quartus Prime software.

Debug Flow with Signal Probe and Rapid Recompile

To add verification capabilities to a design using Signal Probe routing feature:

Reserve Signal Probe Pins on page 3
Compile the Design on page 4
Assign Nodes to Signal Probe Pins on page 4
Recompile the Design on page 5
Check Connection Table in Fitter Report on page 5

Reserve Signal Probe Pins

You create and reserve a pin for Signal Probe with a Tcl command:

```
set_global_assignment –name CREATE_SIGNALPROBE_PIN <pin_name>
```

pin_name Specifies the name of the Signal Probe pin.

Optionally, you can assign locations for the Signal Probe pins. If you do not assign a location, the Fitter places the pins automatically.

Note: If from the onset of the debugging process you know which internal signals you want to route, you can reserve pins and assign nodes before compilation. This early assignment removes the recompilation step from the flow.

Example 1. Tcl Command to Reserve Signal Probe Pins

```
set_global_assignment -name CREATE_SIGNALPROBE_PIN wizard
set_global_assignment -name CREATE_SIGNALPROBE_PIN probey
```

Related Information

Constraining Designs with Tcl Scripts
In *Design Constraints User Guide: Intel Quartus Prime Pro Edition*

Compile the Design

Perform a full compilation of the design. You can use Intel Quartus Prime software, a command line executable, or a Tcl command.

Example 2. Tcl Command to Compile the Design

```
execute_flow -compile
```

At this point in the design flow you determine the nodes you want to debug.

Related Information

Design Compilation
In *Compiler User Guide: Intel Quartus Prime Pro Edition*

Assign Nodes to Signal Probe Pins

You can assign any node in the post-compilation netlist to a Signal Probe pin. In Intel Quartus Prime software, click **View ➤ Node Finder**, and filter by **Signal Tap: post-fitting** to view the nodes you can route.

You specify the node that connects to a Signal Probe pin with a Tcl command:

```
set_instance_assignment –name CONNECT_SIGNALPROBE_PIN <pin_name> –to
<node_name>
```

pin_name Specifies the name of the Signal Probe pin that connects to the node.

node_name Specifies the full hierarchy path of the node you want to route.

Example 3. Tcl Commands to Connect Pins to Internal Nodes

```
# Make assignments to connect nodes of interest to pins
set_instance_assignment -name CONNECT_SIGNALPROBE_PIN wizard -to sprobe_me1
set_instance_assignment -name CONNECT_SIGNALPROBE_PIN probey -to sprobe_me2
```

Recompile the Design

After assigning nodes to the Signal Probe pins, run Rapid Recompile. Rapid Recompile preserves timing and reduces compilation time by reusing previous results whenever possible.

You can run Rapid Recompile from the Intel Quartus Prime software, a command line executable, or a Tcl script.

Example 4. Tcl Command to Recompile the Design

```
# Run the fitter with --recompile to preserve timing
# and quickly connect the Signal Probe pins
execute_module -tool fit -args {--recompile}
```

After recompilation, you are ready to program the device and debug the design.

Related Information

Running Rapid Recompile
In *Compiler User Guide: Intel Quartus Prime Pro Edition*

Check Connection Table in Fitter Report

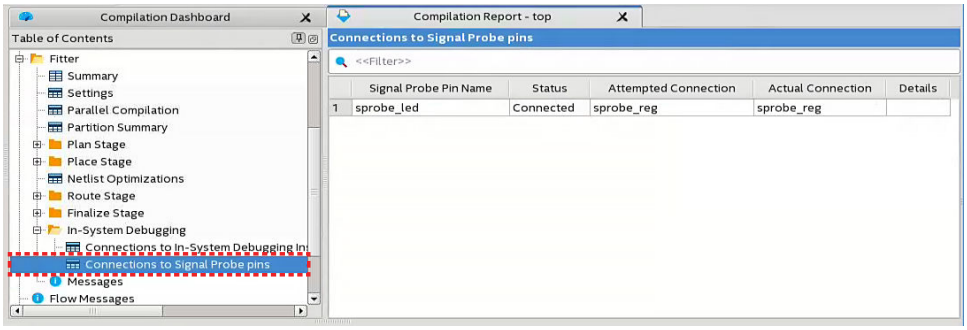
When you compile a design with Signal Probe pins, Intel Quartus Prime software generates a connection report table. To see this report, click **Processing ➤ Compilation Report**, open the **Fitter ➤ In-System Debugging** folder, and click **Connections to Signal Probe pins**.

The **Status** column informs whether or not the routing attempt from the nodes to the Signal Probe pins succeeded.

Table 1. Status of Signal Probe Connection

Status	Description
Connected	Routing succeeded.
Unconnected	Routing did not succeed. Possible reasons are: • Node belongs to an IO cell or another hard IP, thus cannot be routed. • Node hierarchy path does not exist in the design. • Node is not **Signal Tap: post-fitting**.

Example 5. Connections to Signal Probe Pins in the Compilation Report



Alternatively, you can find the Signal Probe connection information in the Fitter report file (<*project_name*>.fit.rpt).

Example 6. Connections to Signal Probe Pins in top.fit.rpt

```
+-------------------------------------------------------------------------------
---+
; Connections to Signal Probe
pins                                                    ;
+-------------------------------------------------------------------------------
---+
Signal Probe Pin Name : probey
Status                : Connected
Attempted Connection  : sprobe_me2
Actual Connection     : sprobe_me2
Details               :

Signal Probe Pin Name : wizard
Status                : Connected
Attempted Connection  : sprobe_me1
Actual Connection     : sprobe_me1
Details               :
+-------------------------------------------------------------------------------
---+
```

Related Information

- Untappable Signals
  In *Debug Tools User Guide: Intel Quartus Prime Pro Edition*
- Text-Based Report Files
  In *Scripting User Guide: Intel Quartus Prime Pro Edition*

Example of Using Signal Probe Routing Feature

Use this example to test the Signal Probe flow in a simple design. You can then modify this example to debug your own projects.

1. Copy and paste `top.tcl` and `top.v` to the working directory.
2. In the Intel Quartus Prime software, click **Tools ➤ Tcl Scripts...**.
3. Select `top.tcl`, and click **Run**.
4. When the script finishes, verify the connection success in the **Connections to Signal Probe Pins** report.

Alternatively, run the example from the command line:

```
quartus_sh -t top.tcl
```

The Tcl script executes with no errors. The Connections to Signal Probe Pins table appears on the Fitter report file, `top.fit.rpt`.

Related Information

Check Connection Table in Fitter Report on page 5

Example Files

Table 2. Example Files

File Name	Description	Code

top.tcl	Contains design setup and assignments	`# Load Quartus Prime Tcl Project package`
		`package require ::quartus::project`
		`package require ::quartus::flow`
		
		`set need_to_close_project 0`
		`set make_assignments 1`
		
		`# Check that the right project is open`
		`if {[is_project_open]} {`
		`  if {[string compare $quartus(project) "top"]} {`
		`  puts "Project top is not open"`
		`  set make_assignments 0`
		`  }`
		`} else {`
		`  # Only open if not already open`
		`  if {[project_exists top]} {`
		`  project_open -revision top top`
		`  } else {`
		`  project_new -revision top top`
		`  }`
		`  set need_to_close_project 1`
		`}`
		
		`if {$make_assignments} {`
		`  # Make general project assignments`
		`  set_global_assignment -name FAMILY "Arria 10"`
		`  set_global_assignment -name DEVICE 10AX115R3F40I2SGE2`
		`  set_global_assignment -name ORIGINAL_QUARTUS_VERSION 17.0.0`
		`  set_location_assignment PIN_N13 -to clk`
		`  set_instance_assignment -name IO_STANDARD LVDS -to clk`
		`  set_location_assignment PIN_N12 -to "clk(n)"`
		
		`  # Make initial assignments to create Signal Probe pins`
		`  set_global_assignment -name CREATE_SIGNALPROBE_PIN wizard`
		`  set_global_assignment -name CREATE_SIGNALPROBE_PIN probey`
		`  set_location_assignment PIN_B25 -to wizard`
		`  set_location_assignment PIN_G25 -to probey`
		
		`  # If you want to compile the design manually,`
		`  # you must save assignment changes to the QSF file`
		`  # by calling "export_assignment"`
		`  #export_assignments`
		
		`  # Full compile`
		`  execute_flow -compile`
		
		`  # Perform debugging operations`
		
		`  # Make assignments to connect nodes of interest to pins`
		`  set_instance_assignment -name CONNECT_SIGNALPROBE_PIN wizard -to sprobe_me1`
		`  set_instance_assignment -name CONNECT_SIGNALPROBE_PIN probey -to sprobe_me2`
		`  export_assignments`
		
		`  # Run the fitter with  --recompile to preserve timing`
		`  # and quickly connect the Signal Probe pins`
		`  execute_module -tool fit -args {--recompile}`
		
		`  # Close project`
		`  if {$need_to_close_project} {`
		`  project_close`
		`  }`
		`}`

top.v	Contains logic description	`//top.v`
		`module top(`
		`  input clk,`
		`  input reset,`
		`  output reg [7:0] bar,`
		`  output wire aux_out`
		`);`
		
		`  reg sprobe_me1;`
		`  reg sprobe_me2;`
		
		`  always @(posedge clk) begin`
		`  if(reset) begin`
		`  sprobe_me1 <= 1'b0;`
		`  sprobe_me2 <= 1'b0;`

continued...

File Name	Description	Code

		`end else begin`
		`sprobe_me1 <= bar[1] & bar[2];`
		`sprobe_me2 <= bar[7] ^ bar[5];`
		`end`
		`end`
		
		`always @(posedge clk) begin`
		`if(reset) begin`
		`bar <= 8'b00000000;`
		`end else begin`
		`bar <= bar + 1'b1;`
		`end`
		`end`
		
		`wire sub_aux;`
		`sub`
		`sub_inst(.clk(clk), .reset(reset), .i1(bar[4]), .i2(bar[3]), .aux_out(sub_aux));`
		
		`assign aux_out = (sprobe_me1 | sprobe_me2) ^ (!sub_aux);`
		
		`endmodule`
		
		`module sub(`
		`input clk,`
		`input reset,`
		`input i1, i2,`
		`output wire aux_out,`
		`output wire sp1,`
		`output wire sp2`
		`);`
		`reg sprobe_me1;`
		`reg sprobe_me2;`
		
		`always @(posedge clk) begin`
		`if(reset) begin`
		`sprobe_me1 <= 1'b1;`
		`sprobe_me2 <= 1'b0;`
		`end else begin`
		`sprobe_me1 <= i1 | i2;`
		`sprobe_me2 <= i1 ^ i2;`
		`end`
		`end`
		
		`assign aux_out = sprobe_me1 & sprobe_me2;`
		`assign sp1 = sprobe_me1;`
		`assign sp2 = sprobe_me2;`
		
		`endmodule`

Document Revision History for AN 799: Quick Intel Arria® 10 Design Debugging Using Signal Probe and Rapid Recompile

Document Version	Intel Quartus Prime Version	Changes
2018.08.16	18.0.0	Fixed link to *Running Rapid Recompile* in *Compiler User Guide: Intel Quartus Prime Pro Edition*
2018.05.07	18.0.0	Updated Tcl syntax to run recompile flow.
2017.05.08	17.0.0	Initial release of the document.