Lattice Radiant Post-Synthesis Reveal Debug Flow Tutorial

December 20, 2024

Copyright

Trademarks

Disclaimers

Type Conventions Used in This Document

Convention	Meaning or Use
Bold	Items in the user interface that you select or click. Text that you type into the user interface.
<*Italic*>	Variables in commands, code syntax, and path names.
Ctrl+L	Press the two keys at the same time.
`Courier`	Code examples. Messages, reports, and prompts from the software.
`...`	Omitted material in a line of code.
`.` `.` `.`	Omitted lines in code and report examples.
`[ ]`	Optional items in syntax descriptions. In bus specifications, the brackets are required.
`( )`	Grouped items in syntax descriptions.
`{ }`	Repeatable items in syntax descriptions.
`|`	A choice between items in syntax descriptions.

Contents

Lattice Radiant Post-Synthesis Reveal Debug Flow Tutorial

This tutorial provides detailed instruction on how to run the post-synthesis debug flow in Reveal. It shows how you can isolate or monitor certain signals and focus changes or improvements in specific areas in your design. As your design becomes more complex, post-synthesis debugging can save you a significant amount of compile time.

About the Tutorial

After completing this tutorial, you should be able to perform the following:

- **Add the syn_rvl_debug attribute to signals that you want to monitor.**
  - Know the functions of the synthesis syn_rvl_debug attribute.
  - Be familiar with the attribute syntax.
  - Attach the syn_rvl_debug attribute to debug signals in your source file.
- **Debug a design in post-synthesis stage**
  - Select Post-Synthesis debug stage.
  - View marked debug signals on the Reveal interface after synthesis:
  - Identify post-synthesis process indicators.
  - Know the Tcl command for post-synthesis debug flow.
  - Verify the post-synthesis debug results.

Time to Complete

About 30 minutes.

Requirements

- To run this tutorial, it is assumed that you are already familiar with the basic Radiant process flow and, specifically, the standard RTL debug flow using Reveal Inserter and Reveal Analyzer.

  To learn more about debugging in Reveal, refer to the online Help and the Reveal User Guide.
- The Lattice Radiant software is required to complete the tutorial.

System Requirements You need:

- Radiant software, version 2024.1 or higher.
- CrossLink-NX Evaluation Board to download a bitstream and to do on-chip debugging. If you do not have the board, you can still do most of the tutorial.

Task 1: Create a New Project

Let us start by creating a new project in Radiant software.

To create a new project:

1. In the Radiant software, choose **File** > **New > Project**.
2. In the Add Source page, click **Add Source...**
3. Browse to: <*Radiant_install_path*>*/docs/tutorial/* ***post_synthesis_reveal_tutorial*** and select the **counter_top.v** file.

   Click **Next**.
4. In the **Select Device** page, choose the following options:
   - **Family**: LIFCL (CrossLink-NX)
   - **Device**: LIFCL-40
   - **Package**: CABGA400

   Click **Next**.
5. In the **Select Synthesis Tool** dialog box, choose **Lattice LSE**.
6. Click **Finish.**

Task 2: Attaching the syn_rvl_debug Attribute to Monitored Signals

One of the prerequisites to post-synthesis debugging is attaching the synthesis syn_rvl_debug attribute to signals that your want to monitor.

- This attribute highlights the signal so it can be easily identified in the user interface.
- In post-synthesis, this attribute tells the synthesis tool to preserve the signal without optimizing it. If it is a bus, the data width is also preserved. The synthesis tool passes the same attribute to the signal in the post-synthesis netlist (*.vm).

  If the signal is a port, the synthesis tool may add a suffix to the signal name because of the input buffer. The signal name, however, remains recognizable to the user. For more information on the expected changes in the signal name, refer to the online Help or the Reveal User Guide.

Here is the Verilog syntax:

```
/* synthesis syn_rvl_debug = 1 */;
```

Here is the VHDL syntax:

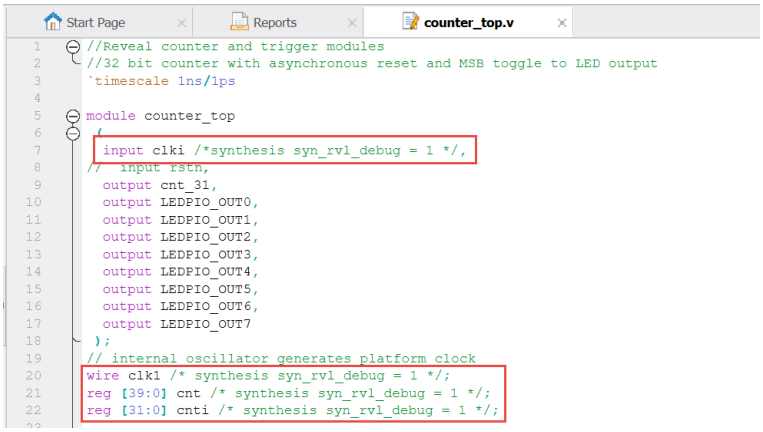
```
attribute syn_rvl_debug : boolean;
attribute syn_rvl_debug of sig1 : signal is true
```

To attach the syn_rvl_debug attribute to signals:

1. In the File List view, double-click the **counter_top.v** file to open the source code.
2. Add the synthesis syn_rvl_debug attribute to the debug signals **clki**, **clk1**, **cnt**, and **cnt1**.

   Note

   While you can add the syn_rvl_debug attribute to signals such as wire, reg, logic, port, input, and output, it is recommended that you use it mostly for internal wire, reg, and logic objects.



3. Save the **counter_top.v** file.
4. Run **Synthesize Design**.

Task 3: Inserting Reveal Debug Logic During Post-Synthesis Stage

The purpose of this task is to add the Reveal debug logic in the post-synthesis stage. Along the way, you will see how the marked debug signals are displayed on the Reveal Inserter interface and how the debug stage is indicated.

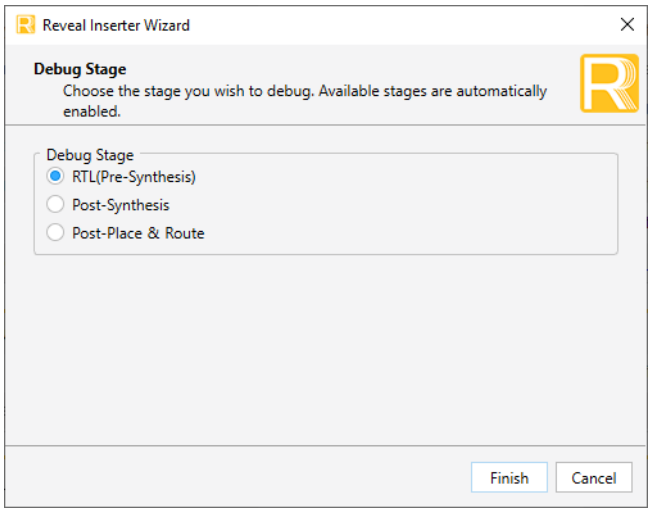
Note:

Before you add post-synthesis Reveal debug logic, make sure there is no active Reveal .rvl project after synthesis. If there is an active Reveal (.rvl) file in the Debug Files folder in File List view, set it as inactive by right-clicking the file and choosing Set as Inactive.

To insert Reveal debug logic in the Post-Synthesis stage:

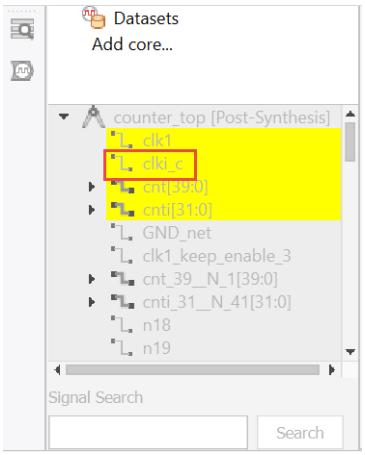
1. After running Synthesize Design, start Reveal Inserter.
2. The Reveal Inserter Wizard allows you to choose between three debug flows:
   - RTL (Pre-Synthesis) – This is the standard flow for debugging a design all at once. The debug logic is inserted before the design is synthesized the first time.
   - Post-Synthesis – This flow allows you to insert debug logic after the design is synthesized.
   - Post-Place & Route – This flow allows you to insert debug logic after running place and route.

Choose **Post-Synthesis** under Debug Stage and click **Finish**.



Reveal Inserter opens and shows the active Reveal project.

In the Datasets pane of the Reveal Inserter window, you will note that the debug signals previously attached with the syn_rvl_debug attribute are highlighted in yellow. This is applied up to the sub-module level.



Note

In the example, the clki signal is appended with a suffix and renamed as clki_c. For additional information regarding the renaming of marked debug signals, refer to the online Help and Reveal User Guide.

You can add multiple Logic Analyzer modules and one Controller module to your project.

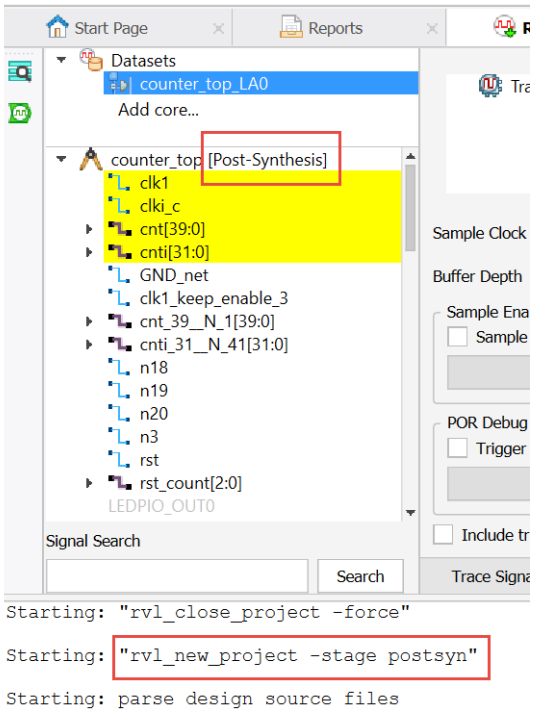
If you are using Reveal Controller for switches in post-synthesis, make sure the signals are not driven by other signals in the design. Multiple driver issues produce an error in Place & Route.

For this tutorial, we will simply add two Logic Analyzer modules.

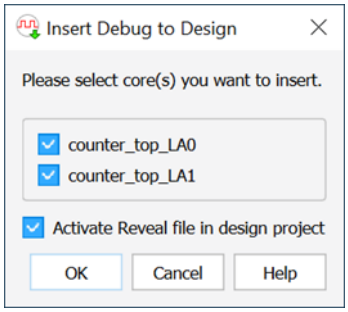
3. Click **Add Core > Add Logic Analyzer**.

   The top-level unit indicates that the design is being debugged in Post-Synthesis stage. In the Tcl Console, you will also find the command:

   ```
   rvl_new_project -stage postsyn
   ```



4. Set up the trace and trigger signals.
5. Add the second Logic Analyzer core and set up trace and trigger signals.
6. Click the **Design Rule Check** button.
7. Click the **Insert Debug** button.
8. In the Insert Debug to Design dialog box, select the module to insert. Select **Activate Reveal file in design project** to include it in synthesis.



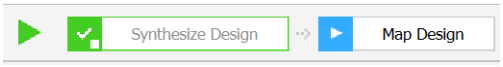
   Click **OK**.

9. Save the Reveal project as **post_syn1.rvl**.

The .rvl file is listed in the File List pane under Debug Files.

You will note that the Synthesize Design process bar now has a small white box.

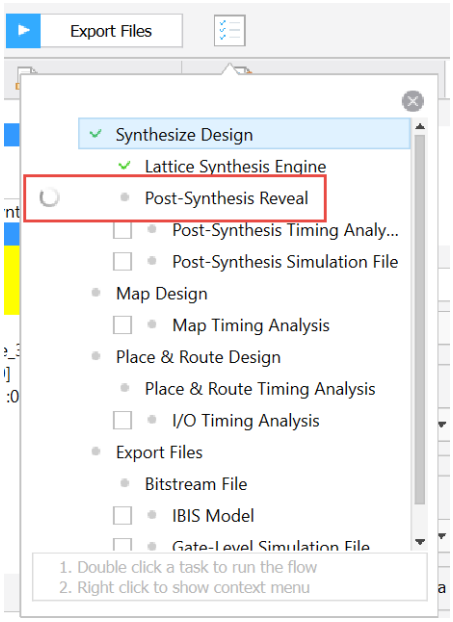
This indicates that entire design will not be synthesized but only the changes made in Reveal.



10. Run **Synthesize Design**.

    This time, only the changes in Reveal are processed.

    In Task Detail View, you will observe the added Post-Synthesis Reveal task in progress.

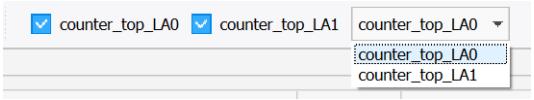


You can continue on with the standard Radiant process flow. Map, place, and route the design, and generate the bitstream data (.bit file).

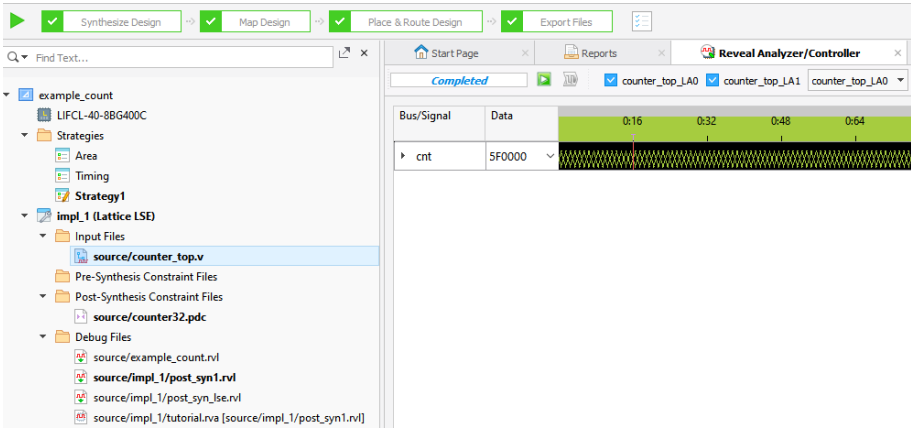
Task 4: Verifying the Results in Reveal Analyzer

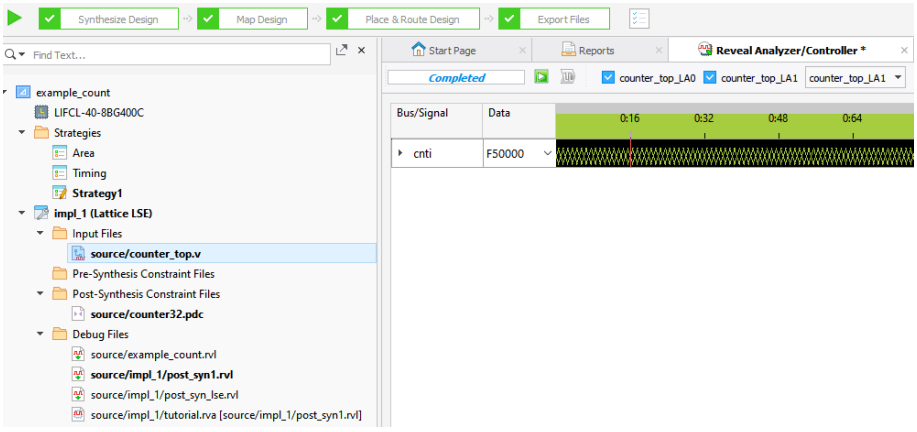
The purpose of this task to is verify the results of the post-synthesis debug process in Reveal Analyzer.

1. Set up a cable connection, and download the design onto the device by using Programmer.
2. Start Reveal Analyzer and create a new project.
3. In the startup wizard, select options and fill out the required fields. Indicate **post_syn1.rvl** in the RVL source field**.** Click **OK.**
4. In the Reveal Analyzer interface LA Trigger tab, select the active debug core that you want to analyze using the drop-down button.



5. Click the **Run** button to start the logic analysis of the active core.
6. Click the LA Waveform tab to view the resulting waveform.





Optionally, you can export the waveform data for each core in a value change dump (.vcd) file for use in third-party tools or in an ASCII-format text (.txt) file.

Summary of Accomplishments

You have completed the *Lattice Radiant Post-Synthesis Reveal Debug Flow Tutorial*. In this tutorial, you have learned how to:

- Add the syn_rvl_debug attribute to monitored signals.
- View marked debug signals on the Reveal interface:
- Identify post-synthesis debug stage through indicators in the interface.
- Identify post-synthesis process indicators.
- Verify the post-synthesis debug results in Reveal Analyzer.

Example Project

You can review this tutorial or familiarize yourself further with the Reveal post-synthesis debug flow using an example project, ***example_count.rdf***. It is available in the ***<Radiant_install_path>/docs/tutorial/post_synthesis_reveal_tutorial/example_post_synthesis_reveal_nexus*** directory.

Recommended References

You can find additional information on the subjects covered by this tutorial in these resources:

- Radiant Help: User Guides > Testing and Debugging On-Chip > Post-Synthesis Debugging
- Reveal User Guide for Radiant Software

Revision History

The following table gives the revision history for this document.

Date	Version	Description
12/20/2024	2024.2	Updated to reflect changes in Radiant 2024.2.
06/28/2024	2024.1	Initial release.