Reveal Troubleshooting for Lattice Radiant Software

This document describes the design restrictions for using on-chip debug.

HDL Language Restrictions

The following features are valid in the VHDL, Verilog and System Verilog languages but are not supported in Lattice Radiant™ Reveal Inserter when you use the RTL flow:

- Array types of two dimensions or more are not available for tracing and triggering.
- Undeclared wires attached to instantiated component instances are not shown in the hierarchical design tree. You must declare these wires explicitly if you want to trace or trigger with them.
- Variables used in generate statements are not available for tracing and triggering.
- Variables used in conditional statements like if-then-else statements are not available for tracing and triggering.
- Variables used in selection statements like the case statement are not available for tracing and triggering.
- If function calls are used in the array declaration, the actual size of the array is unknown to Reveal Inserter.
- Entity and architecture of the same design cannot be in different files.
- In Verilog, you must explicitly declare variables at the very beginning of a module body to avoid obtaining different results from various synthesis tools.
- In VHDL, you must declare synthesis attributes within an entity, not within an architecture, to avoid obtaining different results from various synthesis tools.

- In VHDL, always define the syn_keep and preserve_signal attributes as Boolean types when you declare them in your design. Synplify defines them as Boolean types, and Reveal Inserter will issue an error message if you define them as strings.
- Some signals in a System Verilog design appear in the signal hierarchy but are not available for triggering or tracing. These signals include:
  - Array types of two dimensions or more are not shown in the port or node section
  - Signals that are user defined enumerated types, integer type, byte/ shortint/int/longint type
  - Signals that belong to typedef, interface, struct and union

Reveal Inserter and Radiant Software Errors

This section discusses errors that can occur when you run Reveal Inserter from Radiant software.

Design Parsing Problems in Reveal Inserter

When you start Reveal Inserter from Radiant software, it parses and statically elaborates the design in order to build the hierarchy representation and signal list to make them available for debugging. If the design cannot be parsed and elaborated because of syntax errors, Reveal Inserter's graphical user interface will not open. Instead, a message box opens with an error message similar to that shown in Figure 1. All warnings and errors while reading the design are written to the reveal_error.log file, which is located in the implementation directory of the current Radiant software project. This file contains all the information, warning, and error messages issued by the compiler when it tries to read the design.

Radiant Software Flow Messages

After Reveal Inserter inserts the debug logic, it generates the debug logic cores and passes the information to the File List view for building the design. Several issues could potentially cause the implementation flow to fail because of the debug insertion. Two types of problems could occur:

- Problems with debug design generation
- Problems with design implementation

Debug Design Messages

The debug cores are generated in Reveal Inserter. However, the design must also be modified to allow the debug cores to be connected to the appropriate signals. The modified design is generated during the Synthesis step in the Radiant software design flow. The design is modified with the necessary connections for the debug cores, a temporary HDL file is generated, and the files are synthesized and converted to the Lattice Semiconductor netlist format. Errors generated during this stage are displayed in the automake.log file.

Design Implementation Messages

During the mapping process, errors can occur, such as running out of available resources or tracing or triggering on signals that are not available in the FPGA fabric. During debug insertion, Reveal Inserter checks to make sure that the debug logic is not using more resources than are available in the FPGA. But it does not check to see if the debug logic is using more resources than are available after the design is placed in the FPGA. Currently, resource use can only be accurately checked during the mapping process. Exceeding the available resources results in a mapping error, requiring the debug configuration in Reveal Inserter to be reduced in order to fit.

Signals Unavailable for Tracing and Triggering

Some signals in a VHDL design appear in the signal hierarchy but are not available for triggering or tracing. The following signals are currently unavailable:

- Signals used in “generate” statements are not available for tracing and triggering.
- If function calls are used in the array declaration, the actual size of the array is unknown to Reveal Inserter.
- Signals that are user-defined enumerated types, integer type, or Boolean type are not available for tracing or triggering.

Some signals in a Verilog design appear in the signal hierarchy but are not available for triggering or tracing. The following signals are currently unavailable:

- Array types of two dimensions or more are not shown in the port or node section.
- Undeclared wires attached to instantiated component instances are not shown in the hierarchical design tree. You must declare these wires explicitly if you want to trace or trigger with them.
- If function calls are used in the array declaration, the actual size of the array is unknown to Reveal Inserter.

Some signals that are used in a design but are implemented as hard routes in the FPGA instead of using the FPGA routing fabric are not available for tracing or triggering. Examples are connections to IB and OB components.

Many common hard routes are automatically shown as unavailable in Reveal Inserter, but some are not. If you select a signal for tracing or triggering that is implemented as a hard route, an error will occur during the synthesis, mapping, placement, or routing steps.

Understanding errors reported because of hard routes can be difficult. Here is an example error from the synthesis log file, <*cktname*>.log:

```
@E:"f:\cws\bugs\cr37986\reveal_workspace\tmprveal\rx_ddr_rvl.vh
d":648:8:648:12|Port 'serin' on Chip 'RX_DDR' drives 1 PAD
loads and 1 non PAD loads
```

In this example error message, the serin signal is a hard route, and serin is not the name of the original signal that was traced. The hierarchical path shown is for the debug core that was generated. It is not part of the original design and is not information displayed during the debug insertion. The error message does not specify which user-selected signal used as a trace or trigger is causing the problem. To manually determine which signal is causing this error, you can use two approaches.

- Remove signals one by one in Reveal Inserter to see which caused the error. If you have only a few signals, this would be the best approach.
- Manually look through the design to determine the problem. If you have many signals, this approach would be the best.

  However, the error message refers to the temporary HDL design that is generated during debug logic insertion. Normally this HDL source is deleted after the database is built. To save this temporary HDL source in the case of errors in mapping, you must set an environment variable.

  a. In the System control panel, click on the Advanced tab, then click on the environment variable button at the bottom of the window.

  b. Create a new environment variable named KEEP_REVEAL_TEMP. The value can be anything, but it is normally set to TRUE.

  c. Once this variable is set, exit Radiant software.

  d. Open Radiant software and rebuild the database.

  You can now open the generated HDL to determine which signal caused the error. You can open this file with a text editor, or use HDL Explorer to open and explore the design. The top-level generated file is located at <*project_directory*>/reveal_workspace/tmpreveal/<*project_name*>_rvl.<v or vhd>.

Following is another example of an error generated during mapping. This one is caused by forcing a register whose input is being traced to be implemented as an input flip-flop because of a constraint, USE DIN.

```
ERROR - map: IO register/latch FF_inst cannot be implemented in
PIC.
```

In this case, allowing the register to be implemented as an internal flip-flop by removing the constraint resolves the issue.

NOTE

The current version of the Radiant software support for UltraPlus devices doest not support multi-core debug for Reveal.

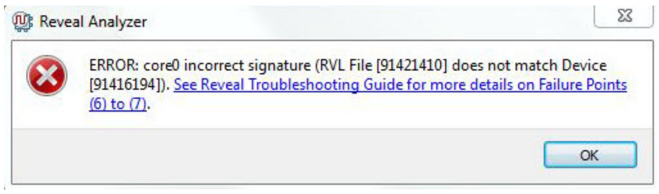
Reveal now supports IEEE-P1735 encryption. If this encryption is applied to a design, it will no longer error out as it did in previous versions. The design tree will allow only the visible ports and signals that are not encrypted to be inserted by the Reveal Inserter for triggering purposes.

Incorrect Signature and Sample Clock

Reveal software uses a signature mechanism to insure that the design loaded in the software and the design programmed in the FPGA match. This prevents wasted time caused by trying to debug one design configuration while a different one is actually loaded. When Reveal Inserter writes out the debug information into its file (.rvl file) a signature is added based on the timestamp. This signature is implemented into the debug core which is programmed into the FPGA. When Reveal Analyzer creates a new file (.rva file), it reads the Reveal Inserter file and also reads the signature from the debug core. If they do not match, this causes the incorrect signature error message. There are three main causes for this error.

The first cause is that a different design is programmed into the FPGA than is represented by the Reveal Inserter file (.rvl file). This can be caused by programming an old bit file or by changing the Reveal Inserter file after programming the FPGA. Opening Reveal Inserter and then saving the file after the design has been programmed will cause this error. In this situation the error message will look similar to the message below where the mismatch is between two valid numbers.

Figure 1: Reveal Analyzer Invalid Design Error Message

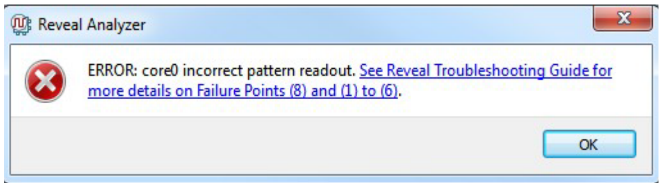


ERROR: core0 incorrect signature (RVL File [91421410] does not match Device [91416194]). See Reveal Troubleshooting Guide for more details on Failure Points (6) to (7).

The second cause is from a sample clock problem.

The sample clock is used by Reveal debug logic to clock data into the trace buffer and in the triggering logic. The sample clock is also needed when Reveal Analyzer communicates with the debug logic through JTAG. If the sample clock is not running or is running too slow, Reveal Analyzer cannot detect that the Reveal debug logic is available. This information is especially important when you create a new Reveal Analyzer project. Reveal Analyzer checks the debug logic for a signature to make sure that the bitstream matches the design. If Reveal Analyzer cannot communicate with the debug logic because the sample clock is not running, the project creation or the Reveal Analyzer run command will fail with an error, and Reveal Analyzer issues an error message similar to that shown in Figure 2. For these reasons, the sample clock should be a signal with a reasonably regular frequency rather than a signal with intermittent pulses. The frequency of the sample clock should also be faster than the speed of the JTAG clock that is used.

Figure 2: Reveal Analyzer Sample Clock Error Message


Reveal Analyzer

ERROR: core0 incorrect pattern readout. See Reveal Troubleshooting Guide for more details on Failure Points (8) and (1) to (6).

The third cause for the incorrect signature error message is when the sample clock is not correctly connected to the debug logic. This can occur if a problem happens in the implementation flow. The signature read from the device will be all ones in this situation. To resolve this, the post-map netlist needs to be viewed directly to determine the root cause. Error messages and debug information will be saved in the reveal_debug.log file located under the implementation directory, please include this file when contacting Lattice Technical Support.

Unexpected Reveal Analyzer Results

Using a trigger signal that is the output of a very large logic cone may produce confusing results in Reveal Analyzer. A glitch on an asynchronous trigger signal in rare cases may cause the trigger logic to become active prematurely. If you encounter this situation, register your trigger signal with the sample clock and use that as the trigger.

Performance

When you open Reveal Inserter for an RTL project, it must first parse the entire design in order to build the design hierarchy and signal list. Normally this occurs within a few seconds. Very large designs may take significantly longer.

When you change trigger settings in Reveal Analyzer, the settings must be downloaded to the debug logic on the FPGA when you press the Run button. While the debug logic settings are being downloaded, the `Configuring ...` message appears in the upper left corner of the window.

The Reveal triggering logic, which is composed of trigger units and trigger expressions, offers unique capabilities and flexibility. However, there is a latency of five sample clocks to the output of the final trigger condition. Reveal Analyzer software automatically handles this latency delay so that the trigger point lines up with the correct data when waveforms are displayed in the Waveform view. The trigger-out signal also has this five-clock latency delay. When you use the trigger-out signal as an input to another core or as an external trigger-out signal, the five-sample-clocks delay from the actual trigger event must be taken into account. Otherwise, the captured data will not line up with the desired event.

Failure Points of Analyzer Function

The following are typical failure points of Reveal Analyzer, along with proposed solutions.

1. Reveal software client tool which communicates with the cable server.Some of the functions are providing the register settings, trigger points, downloading trace data, etc. The problem can include not being able to communicate with the cable server, which may not be responding, or receiving wrong data from the cable server which may be running in corrupted state.

   Solution: Terminate existing cable server process.

2. The cable server communicates with the device on the board through a cable. The cable should be the correct cable with the correct port and should be selected in the GUI. The type should match with the actual physical cable.

   Solution: Select the correct cable port.

3. The JTAG ports (TDI, TCK, TDO, TMS) should be properly located and connected on the board so that the cable has the correct connection from the Reveal Analyzer client to the device. The JTAG IO pins must not be shared with any other wires on the board.

   Solution: Check the JTAG pins located in Radiant software Signal/Pad report.

4. The JTAG communicates with the user design using the Debuggerinserted pre-synthesis. The trace trigger and data are in sample clock domain. The sample clock must be clean and continuous and not intermittent. The sample clock frequency also must be more-than or equal-to the JTAG clock frequency.

   Solution: Run PAR Timing Analysis to check for timing violations.

5. The board must be properly powered-up.

   Solution: Connect the board to a power supply with the correct voltage. If there is on/off switch on the board, make sure it is turned to ON.

6. The right config file should be used for configuration. Sometimes users use wrong or old config file by mistake.

   Solution: Select config file by correct name and correct type from rbt, bit, jed.

7. The Reveal project files .rvl, .rva files need to be under design directory.

   Solution: Select correct rvl to create new rva in Startup Wizard.

8. When SOFT-JTAG is used in some devices, the scan function for device id should not be used and correct port is selected and located.

   Solution: select correct cable port for debug different from programming port.

9. When multiple devices are chained then the correct chain information needs to be in the .xcf file.

   Solution: Select correct .xcf to create new rva in Startup Wizard.

Using the Reveal Debug Projects

If you are having trouble running Reveal with your design, Lattice provides the following pre-verified Reveal Debug Project to allow you to verify that Reveal is working correctly on your computer.

- counter_reveal

The Reveal Debug Project is located in a folder in the examples directory: <radiant_install_path>\examples\

Your computer must be connected to a board with the appropriate Lattice device.

The sample design is a 24-bit counter which includes one input reset and 1 output a1 to drive an instantiated RGB primitive.

The counter will be toggling with the LED once the design is programmed.

The clock signal is driven by internal oscillator and the RGB primitive are connecting to a1. This reference design has a Reveal module inserted and has trace signal TU1 thats monitoring count[3:0] equal to 4b'1001 as trigger condition.

To start the Reveal example project:

1. Choose **File > Open > Design Example**. The Open Example dialog box opens.
2. Click counter_reveal, browse to select the location.
3. Click **OK**.
4. Ensure that the Radiant software project settings match the device on your board.
5. Follow the steps outlined in the “Performing Logic Analysis” section of the Radiant software online help.
   - If you are able to run the Reveal Debug Project successfully in Reveal, then the problem may be with your design or in the clock source.

- If you are unable to run the Reveal Debug Project successfully in Reveal, contact Lattice Technical Support and send the project with the log files.