Reveal Troubleshooting Guide

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

HDL Language Restrictions

The following features are valid in the VHDL and Verilog languages but are not supported in 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.

EDIF Support

The EDIF flow is fully supported in Reveal. However, you must be aware of the following:

- Reveal Inserter must be started from a Diamond project. In order to use the EDIF flow with Reveal Inserter, you must start Reveal Inserter from a Diamond project containing either EDIF source or mixed VHDL & Verilog source files. Projects with only VHDL or Verilog will run Reveal Inserter with an HDL source flow.
- In the EDIF flow, the representation in Reveal Inserter is of the EDIF hierarchy and signal names. Buses appear as individual signals instead of buses, as in the RTL flow.

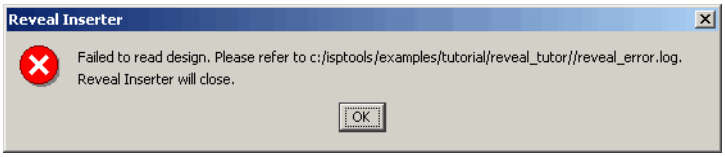
Reveal Inserter and Diamond Errors

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

Design Parsing Problems in Reveal Inserter

When you start Reveal Inserter from Diamond, 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 Diamond project. This file contains all the information, warning, and error messages issued by the compiler when it tries to read the design.

Figure 1: Reveal Inserter Design Parsing Error Message



Diamond 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. Three types of problems could occur:

- Problems with debug design generation
- Problems with connecting JTAG functionality
- 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 Diamond 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.

Another potential failure occurs when you trigger or trace signals that are implemented as hard routes in the FPGA and that are not available in the FPGA routing fabric. An example would be the output of a SERDES block. This output is directly routed within the FPGA and is not available in the FPGA routing fabric. Design signals that are implemented on these hard routes are therefore unavailable for debugging since debugging requires access on the routing fabric. Many hard routes are detected and blocked from use as trigger or trace signals, but it is still possible to select some signals that cannot be connected to the debug logic.

Preferences Not Recognized Due to Reveal Being Present

When Reveal Inserter adds debugging information to a design, it must connect to the signals being used for triggers, traces, and the sample clock. If any of these signals have preferences on them, there is a chance that the preference may no longer be recognized. This only occurs on signals that do not go to the top level of the design when Reveal is not present.

This occurs because when Reveal Inserter adds debugging cores to the design, the necessary signals are connected to the debugging cores. Signals are traversed when a design is loaded and the top-level name of the signal is used for assigning preferences. If the signal now connects to a top-level debug core, the name on the net segment attached to the debugging core is now used instead of the original name.

This problem occurs most often on the sample clock. Being a clock signal, it is likely there is a preference already assigned to this signal. If the sample clock was not originally present in the top level of the design, the original preference is not recognized. This results in one or more warnings of the following form when Design Planner is opened:

WARNING - baspe: Semantic Error: clocksig_200 matches no nets in the design. Occurred in "USE PRIMARY PURE NET "clocksig_200" ; ". Disabled this preference.

One solution is to copy the preference for the original signal and make a new preference for the net name that is now being used. Reveal Inserter always generates a name in the format of reveal_ist_<*number*>.

In the case of the above warning, the Period/Frequency sheet of Spreadsheet View in Design Planner shows a clknet named reveal_ist_67 instead of clocksig_200. In the preference file (<*projectname*>.lpf), this can be corrected by copying the existing preference line ‘USE PRIMARY PURE NET “clocksig_200” ;’ and adding the line ‘USE PRIMARY PURE NET “reveal_ist_67” ;’.

In ispLEVER 8.0 and later releases and in Diamond 1.0, Reveal Inserter attempts to automatically prevent errors by creating an alias in the constraints file (.lpf) for any clock signals that are used. In the constraints file, a preference of the form 'rvl_alias "reveal_clock_signal" "original_clock_signal";' is automatically created for signals that are recognized as clocks by Reveal Inserter.

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 Diamond.

d. Open Diamond 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 (which is part of ispLEVER) 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 preference, 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 preference resolves the issue.

JTAG Restrictions

Reveal requires a JTAG connection for configuring the debug logic and transferring captured data to the PC. If other modules or components that also use JTAG are present, an error will occur during the Translate step in the Process view, indicating that there are multiple JTAG modules present. The same error occurs if an IP or module contains a JTAG module. An instantiated JTAG component primitive causes the same error.

Reveal supports debugging a device in a JTAG chain with multiple devices on the chain. Reveal also supports debugging a device with multiple cores. However, when you debug a design with multiple devices on a JTAG chain, all the cores must be present in a single device. Multiple cores distributed among multiple devices are not currently supported.

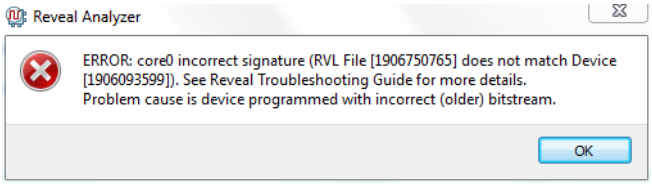
If you are debugging an FPGA in a JTAG chain, you must supply an .xcf file in Reveal Analyzer project. The .xcf file must reside in the project directory; it cannot reside in any other location.

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.

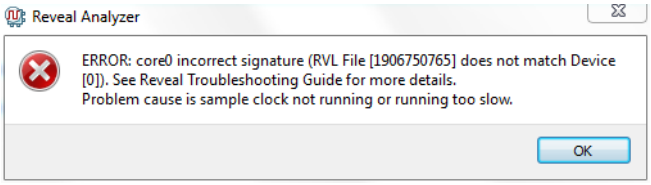
Figure 2: Reveal Analyzer Invalid Design Error Message



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 3. 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 3: Reveal Analyzer Sample Clock Error Message



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. Contact Lattice Technical Support if encountering this message.

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 open Reveal Inserter for an EDIF project, it must read and load the entire design in order to build the design hierarchy and signal list. For very large designs, loading can take several minutes. On these large designs, there is a delay until Reveal Inserter starts and then a further delay while it builds the hierarchy and signal list. Although load times may be long, the delay is approximately the same as that of other tools loading similar-sized designs.

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 time taken to configure the debug logic depends on how many trigger units (TUs) and trigger expressions (TEs) are used. For a few TUs and one or two TEs, the time is usually a few seconds. For a maximum configuration of 16 TUs and 16 TEs on multiple debug logic cores, the time to configure could take several minutes. The debug logic does not start looking for the triggers set until the `Configuring ...` message is gone and the `Running Device ...` message appears.

Each debug logic core offers an optional trigger-out signal. 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.

Creating a Reveal Analyzer Project

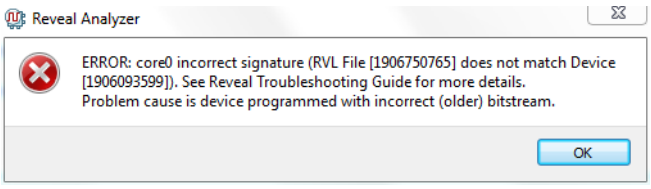
When you create a new Reveal Analyzer project, some additional setup is required that Reveal Inserter does not require. The target board is expected to be programmed and turned on. When you first start Reveal Analyzer, the cable settings should be set up. Even if the cable has already been set up in ispVM System, you must specify it in Reveal Analyzer. If you are using multiple USB cables on the PC, Reveal Analyzer only recognizes the cable at the USB 0 slot. Other cables are not available. It may be necessary to switch USB cables to enable Reveal Analyzer to see the target device in this situation.

If you are debugging an FPGA in a JTAG chain, you must supply an .xcf file in the Reveal Analyzer project. The .xcf file must reside in the project directory; it cannot reside in any other location.

As noted in “Incorrect Signature and Sample Clock” on page 6, the sample clock should be running when you create a new project or open Reveal Analyzer on an existing project. If the sample clock is not running, Reveal Analyzer cannot detect that the Reveal debug logic is available. When you create or open a project, Reveal Analyzer checks the debug logic for a signature to make sure the bitstream matches the design. If Reveal Analyzer cannot communicate with the debug logic because the sample clock is not running, the project creation will fail with an error.

Reveal checks both the debug project files and the bitstream in the FPGA to make sure that they match. Reveal Inserter generates a unique signature code that is stored in the settings file and is implemented as hardware in the debug logic core. It prevents you from looking at the settings read from the design files while trying to debug a different setup that was downloaded by mistake to the FPGA. However, if you used Reveal Inserter to change the debug settings, even though the design has not been re-implemented and programmed into the FPGA, a signature error similar to that shown in Figure 4 will occur. Most of the debug settings files, which contain the automatically generated signature, are shared between Reveal Inserter and Reveal Analyzer. You can change the settings in Reveal Analyzer without any signature mismatches occurring. Since Reveal Inserter can change settings that affect the debug implementation, any changes in Reveal Inserter cause the signature to be regenerated. Opening Reveal Inserter without changing any settings and saving the project also causes the signature to be regenerated.

Figure 4: Reveal Analyzer Mismatch Error Message



If you run Reveal Analyzer with a 7.0 SP2 version of the .rvl file or earlier, you will see a “Core version number mismatch” error

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 beomg 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 Diamond Signal/Pad report.

4. The JTAG communicates with the user design using the Debugger inserted 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 Trace 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 Projects to allow you to verify that Reveal is working correctly on your computer.

- counter_reveal_ECP2
- counter_reveal_ECP3
- counter_reveal_ECP5
- counter_reveal_LIFMD
- counter_reveal_MACHXO2
- counter_reveal_XO
- counter_reveal_XP2

The Reveal Debug Projects are located in a folder in the examples directory:

<diamond_install_path>\examples\reveal_debugger\

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

The reference design is a 32-bit counter which includes one input reset and 4 outputs LEDPIO_OUT0, LEDPIO_OUT1, LEDPIO_OUT2 and LEDPIO_OUT3.

Choose the appropriate device package and locate the output ports to see if the counter is toggling with the LED.

The clock signal is driven by internal oscillator and the LEDPIO_OUT3 to LEDPIO_OUT0 are connecting to the most significant bit cnt[31:28]. This reference design has a Reveal module inserted and has trace signal TU1 that monitoring cnt[5:0] equal to 6b'100000 as trigger condition.

To start the Reveal example project:

1. Choose **File > Open > Project**.

   The Open Example dialog box opens.

2. Browse to the desired Reveal Debug Project folder, and choose the apppropriate **.ldf** file.
3. Click **Open**.
4. Ensure that the Diamond project settings match the device on your board.
5. Follow the steps outlined in the “Performing Logic Analysis” section of the Diamond 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.

Trademarks