



Analyzing Designs Using Lattice Radiant Software Tools

Application Note

FPGA-AN-02080-1.0

February 2024

Disclaimers

Lattice makes no warranty, representation, or guarantee regarding the accuracy of information contained in this document or the suitability of its products for any particular purpose. All information herein is provided AS IS, with all faults, and all associated risk is the responsibility entirely of the Buyer. The information provided herein is for informational purposes only and may contain technical inaccuracies or omissions, and may be otherwise rendered inaccurate for many reasons, and Lattice assumes no obligation to update or otherwise correct or revise this information. Products sold by Lattice have been subject to limited testing and it is the Buyer's responsibility to independently determine the suitability of any products and to test and verify the same. LATTICE PRODUCTS AND SERVICES ARE NOT DESIGNED, MANUFACTURED, OR TESTED FOR USE IN LIFE OR SAFETY CRITICAL SYSTEMS, HAZARDOUS ENVIRONMENTS, OR ANY OTHER ENVIRONMENTS REQUIRING FAIL-SAFE PERFORMANCE, INCLUDING ANY APPLICATION IN WHICH THE FAILURE OF THE PRODUCT OR SERVICE COULD LEAD TO DEATH, PERSONAL INJURY, SEVERE PROPERTY DAMAGE OR ENVIRONMENTAL HARM (COLLECTIVELY, "HIGH-RISK USES"). FURTHER, BUYER MUST TAKE PRUDENT STEPS TO PROTECT AGAINST PRODUCT AND SERVICE FAILURES, INCLUDING PROVIDING APPROPRIATE REDUNDANCIES, FAIL-SAFE FEATURES, AND/OR SHUT-DOWN MECHANISMS. LATTICE EXPRESSLY DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY OF FITNESS OF THE PRODUCTS OR SERVICES FOR HIGH-RISK USES. The information provided in this document is proprietary to Lattice Semiconductor, and Lattice reserves the right to make any changes to the information in this document or to any products at any time without notice.

Inclusive Language

This document was created consistent with Lattice Semiconductor's inclusive language policy. In some cases, the language in underlying tools and other items may not yet have been updated. Please refer to Lattice's inclusive language [FAQ 6878](#) for a cross reference of terms. Note in some cases such as register names and state names it has been necessary to continue to utilize older terminology for compatibility.

Contents

Contents.....	3
Acronyms in This Document	6
1. Introduction	7
2. RTL Coding Best Practices and Debugging	8
2.1. RTL Coding Best Practices.....	8
2.2. Post-RTL Development Debugging.....	8
3. Software Tools for Debugging	9
3.1. Netlist Analyzer	9
3.1.1. Netlist Analyzer Menu Options	10
3.1.2. Selecting an Instance	11
3.1.3. Focusing on a Module of Interest.....	12
3.1.4. Signal Tracing Through Pins.....	15
3.1.5. Flattening and Unflattening a Design Hierarchy.....	16
3.1.6. Net Tracing	18
3.2. Physical Designer – Placement Mode	22
3.2.1. An Example of When to Use Floorplanning.....	23
3.2.2. An Example of When Not to Use Floorplanning.....	23
3.3. Physical Designer – Routing Mode	24
3.3.1. Analyzing Net Delay in the Physical View Routing mode	24
3.3.2. Examples of Using Routing Mode to Debug a Design.....	26
3.4. Physical Designer – Cross-Probing.....	29
References	31
Technical Support Assistance	32
Revision History	33

Figures

Figure 3.1. Example View of the Netlist Analyzer in the Radiant Software	9
Figure 3.2. Selected Instance from the Schematic View.....	11
Figure 3.3. Selected Instance from the Instance Window	11
Figure 3.4. Highlighted Instance from the RTL Code	11
Figure 3.5. Filtered Module	12
Figure 3.6. Click on Isolate Path of Interested Module.....	13
Figure 3.7. Filtered Instance with Signal Paths	13
Figure 3.8. Selected Module to be Trimmed	13
Figure 3.9. Click on Less to Trim the Selected Module	14
Figure 3.10. Schematic View after Selected Module is Trimmed	14
Figure 3.11. Restoring the Schematic View	14
Figure 3.12. Module of Interest	15
Figure 3.13. Logic Connected to the Selected Pin.....	16
Figure 3.14. Logic Connected to the <i>d</i> Pin	16
Figure 3.15. Flatten the Module Using Dissolve Instance.....	17
Figure 3.16. Using Dissolve Instance on Inner Module.....	18
Figure 3.17. Restoring to Original Schematic.....	18
Figure 3.18. Jump to HDL File Option	19
Figure 3.19. Jump to Technology View Option	19
Figure 3.20. Go to Driver Option.....	20
Figure 3.21. Go to Connected Instances Option	20
Figure 3.22. Highlight Option.....	21
Figure 3.23. Properties Option	21
Figure 3.24. Pin Pair Delay Option	25
Figure 3.25. Output Window Display.....	25
Figure 3.26. RTL Code of a Simple Inverter.....	26
Figure 3.27. Routing Mode Showing the Attribute is Not Honored	26
Figure 3.28. Modified RTL Code of a Simple Inverter	27
Figure 3.29. Routing Mode Showing the Attribute is Honored	27
Figure 3.30. Chosen Net of Interest.....	28
Figure 3.31. Routing Mode Menu Options	28
Figure 3.32. Route of Interest	28
Figure 3.33. Selected Component in the Placement Mode	29
Figure 3.34. Physical View of the Component	29
Figure 3.35. Selected Net of Interest	30
Figure 3.36. Placement Mode View.....	30

Tables

Table 2.1. Best Practices for RTL Coding.....	8
Table 2.2. Available Tools in the Radiant Software for Post-RTL Debugging.....	8
Table 3.1. Netlist Analyzer Menu Icons and their Definitions.....	10
Table 3.2. Advantages and Consideration of Flattening a Design Hierarchy.....	16
Table 3.3. Key Features of Placement Mode.....	22
Table 3.4. When to Do Floorplanning.....	22
Table 3.5. Preparation Before Floorplanning.....	23
Table 3.6. Considerations During Floorplanning.....	23
Table 3.7. Key Features of Routing Mode.....	24

Acronyms in This Document

A list of acronyms used in this document.

Acronym	Definition
FPGA	Field Programmable Gate Array
HDL	Hardware Description Language
LSE	Lattice Synthesis Engine
PAR	Place and Route
PLC	Programmable Logic Cell
RTL	Register-Transfer Level

1. Introduction

This document describes the various tools available in the Lattice Radiant™ software that can be used to debug and analyze your design at different development stages.

Debugging process needs to be performed when design functionality is not working correctly. Debugging can be performed in various stages, namely, Register-Transfer Level (RTL) coding, simulation, synthesis, mapping, place and route (PAR), and bitstream generation.

The Radiant software is a complete design environment for Lattice Semiconductor Field Programmable Gate Array (FPGA). The software includes a comprehensive set of tools for all design tasks, including project management, design entry, simulation, synthesis, place and route, in-system logic analysis, and more.

2. RTL Coding Best Practices and Debugging

2.1. RTL Coding Best Practices

It is important to apply RTL coding best practices to ensure a smooth design and effective debugging processes. [Table 2.1](#) describes the best practices for RTL coding.

Table 2.1. Best Practices for RTL Coding

Best Practices	Descriptions
Commenting for clarity	When working with RTL code, incorporating comments is a crucial practice. Comments are notes in the code that explain what each section does which makes it easier to understand the purpose and functionality of the code.
Immediate testing after changes	After making changes to the RTL code, it is essential to test those modifications immediately to ensure that the alterations have not introduced new issues and the code still functions as expected. Immediate testing helps catch and address any issues early in the development process.
Pre-debugging understanding	Before diving into the debugging process, take the time to thoroughly understand the RTL code. Knowing how each part of the code contributes to the overall functionality allows for more effective and targeted debugging process.

2.2. Post-RTL Development Debugging

There are different tools available in the Radiant software which can be used to debug and analyze your design at various stages of the RTL development. By effectively utilizing these tools, you can streamline the debugging process leading to a more robust and reliable design. [Table 2.2](#) lists the software tools available that can be used for debugging at different post-RTL development stages.

Table 2.2. Available Tools in the Radiant Software for Post-RTL Debugging

Post-RTL Development Stages	Radiant Software Tools
Synthesized Netlist	To debug the synthesized netlist, leverage tools like Netlist Analyzer (LSE) or HDL Analyst (Synplify™ Pro) . These tools provide insights into the synthesized design which help to identify and rectify any issue related to the netlist.
Placement	Use the Floorplan View tool to debug the placement of components in your design. This tool allows you to visualize and optimize the physical arrangement of elements on the chip, ensuring efficient use of resources.
Routing	Use the Physical View tool to debug the routing of signals within the design. This tool enables you to inspect and optimize the paths that connect different components, ensuring a well-organized and efficient layout.
Timing Closure	To achieve timing closure, use the Timing Analysis View tool. This tool helps to analyze and optimize the timing characteristics of the design, ensuring that signals meet the required timing constraints.

3. Software Tools for Debugging

This section describes the tools available in the Radiant software that can be used for debugging process.

3.1. Netlist Analyzer

Netlist Analyzer provides various views that help you to debug your RTL code and synthesized netlists, namely, the RTL View and Technology View. The RTL View transforms your RTL code into a visually accessible schematic format for quick interpretation and analysis. The Technology View provides an insider's perspective which allows a detailed examination of the synthesized netlist's technical view.

Netlist Analyzer (LSE) works completely when LSE is selected as the synthesis tool. Only post-synthesis Technology View is available in the Netlist Analyzer when Synplify Pro is used for the synthesis. [Figure 3.1](#) shows an example view of the Netlist Analyzer in the Radiant software.

Netlist Analyzer can be used to:

- Understand the design hierarchy of your RTL code
- Monitor the design implementation post synthesis
- Cross probe from schematic view to corresponding RTL code

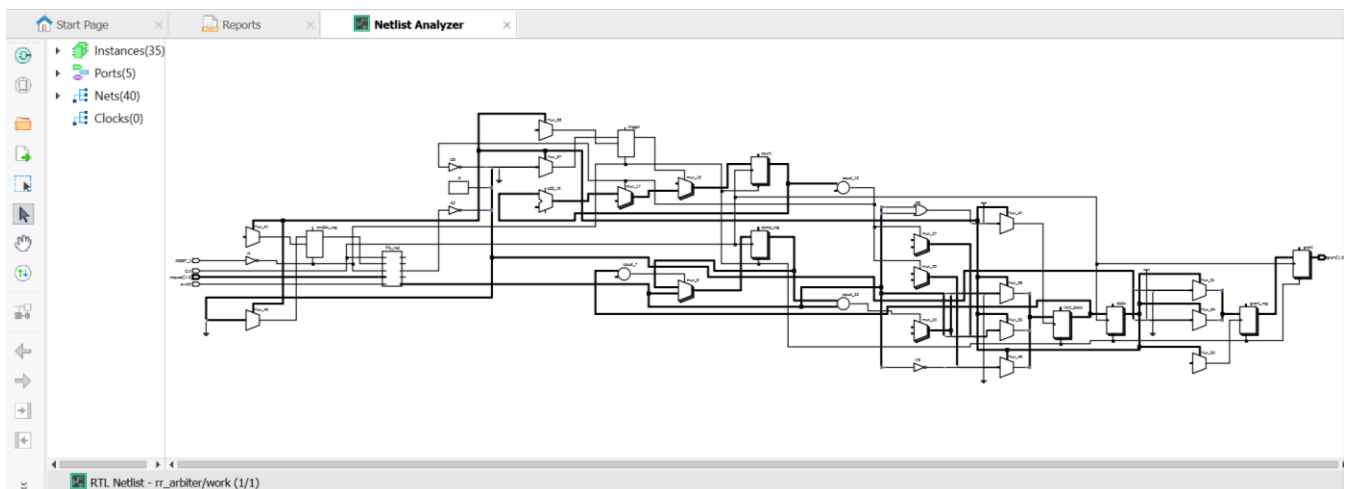


Figure 3.1. Example View of the Netlist Analyzer in the Radiant Software

3.1.1. Netlist Analyzer Menu Options

Table 3.1 lists the Netlist Analyzer menu icons and their definitions.

Table 3.1. Netlist Analyzer Menu Icons and their Definitions

Icons	Definitions
	RTL View
	Technology View
	Open External File – Opens an existing .vdb (Netlist database) file
	Export – Exports the current schematic to a pdf file
	Area Selection Mode
	Point Mode
	Pan Mode
	Push Down/Pop Up – Used to Push and Pop the hierarchy
	Filter Schematic
	Backward – Allows to go back to the previous view (similar to Undo)
	Forward – Allows to go forward to the next view (similar to Redo)
	Next Sheet
	Previous Sheet
	Go To Sheet
	Sheet Bookmark
	Show World View

3.1.2. Selecting an Instance

Netlist Analyzer can be used to select an instance from the schematic view and allows you to cross probe it to the instance declaration in your RTL code.

The following example shows how to select an instance:

1. Select an instance from the schematic view as shown in [Figure 3.2](#).

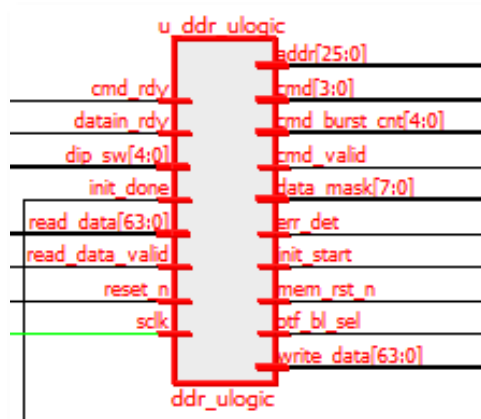


Figure 3.2. Selected Instance from the Schematic View

2. The corresponding instance is highlighted in the instance window as shown in [Figure 3.3](#).

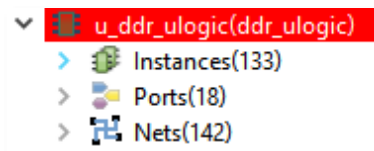


Figure 3.3. Selected Instance from the Instance Window

3. Double-click on this instance from the schematic window to open the instantiation of that instance in the source editor as shown in [Figure 3.4](#).

```

dds_uloic  u_ddr_uloic (
//-----inputs-----
  .sclk          (sclk),
  .reset_n      (reset_n),
  .init_done    (init_done),
  .cmd_rdy      (cmd_rdy),
  .datain_rdy   (datain_rdy),
  .read_data    (read_data),
  .read_data_valid (read_data_valid),
  .dip_sw       (dip_sw[4:0]),
  .cmd          (cmd),
  .cmd_valid    (cmd_valid),
  .addr         (addr),
  .cmd_burst_cnt (cmd_burst_cnt),
  .write_data   (write_data),
  .init_start   (init_start),
  .mem_rst_n    (mem_rst_n),
  .data_mask    (data_mask),
  .out_bl_sel   (ofly_burst_len),
  .err_det      (err_det)
) /*synthesis UGROUP="dds3_uloic" PBOX="25,20"

```

Figure 3.4. Highlighted Instance from the RTL Code

3.1.3. Focusing on a Module of Interest

While debugging, you may want to focus on one particular module and there are a few methods you can use.

Netlist Analyzer allows you to:


- Filter an Instance
- Filter with Signal Paths
- Trim the Schematic
- Push in the Hierarchy

The procedure for each method is detailed in the following section.

3.1.3.1. Filter an Instance

During the debugging process, it is essential to isolate and examine specific module within a larger system. Netlist Analyzer provides a feature that allows you to filter a particular instance, enabling focused scrutiny on the behavior and performance of that specific module. This is useful when dealing with complex systems where understanding the interactions between individual components is crucial for effective debugging.

The following example shows how to filter an instance:

1. Select a module of interest either from the Netlist browser or from the schematic view.
2. Click on the  (Filter Schematic) button. [Figure 3.5](#) shows the example of a filtered module.

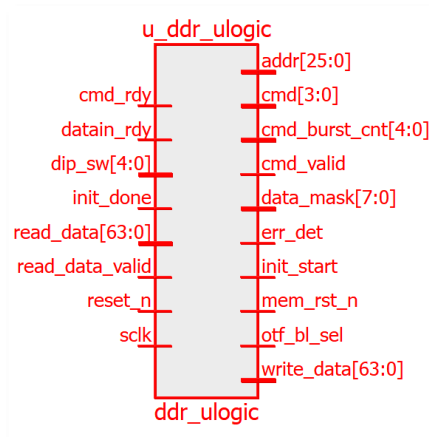


Figure 3.5. Filtered Module

3.1.3.2. Filter with Signal Paths

Netlist Analyzer offers the capability to filter based on signal paths which allows you to narrow down your analysis to specific signal flow within the module, making it easier to trace the propagation of signals and identify potential issues. This feature is useful in determining the exact areas where anomalies or errors may be occurring, streamlining the debugging process, and reducing the time required to identify and address the issues.

The following example shows how to filter with signal paths:

1. Click on the module(s) of interest.
2. In the schematic view, right-click and choose **Isolate Path** as shown in [Figure 3.6](#).

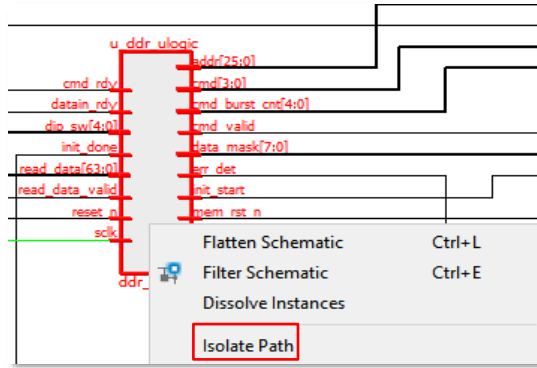


Figure 3.6. Click on Isolate Path of Interested Module

- Figure 3.7 shows the view after selecting **Isolate Path**. The dotted line indicates that the net has other connected instances or components. To see the connected components, double-click on the dotted nets.

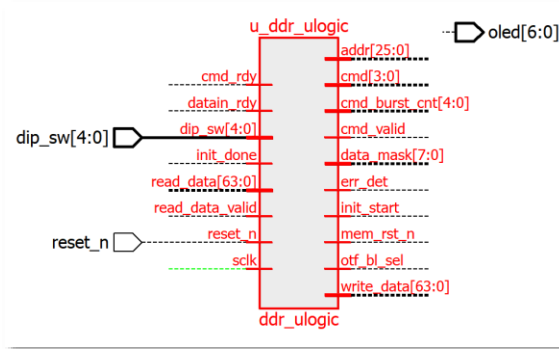


Figure 3.7. Filtered Instance with Signal Paths

3.1.3.3. Trim the Schematic

In debugging scenarios, a cluttered schematic can hinder the identification of issues. Netlist Analyzer addresses this by providing a *Trim the Schematic* feature. This functionality allows you to simplify the schematic representation, focusing only on the relevant details of the module under investigation. By decluttering the visual representation, you can enhance your ability to comprehend and troubleshoot the module’s functionality.

The following example shows how to trim the schematic:

- Select the object(s) that you do not want to focus on as shown in Figure 3.8.

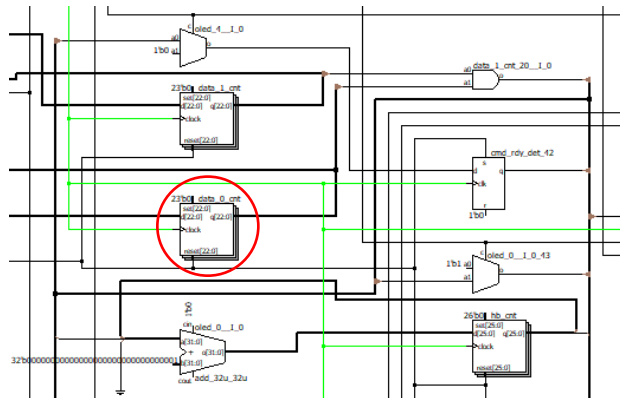


Figure 3.8. Selected Module to be Trimmed

- Right-click on the schematic view and select **Less** as shown in Figure 3.9.

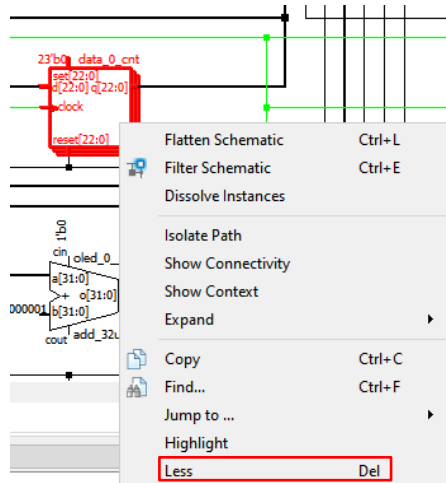


Figure 3.9. Click on Less to Trim the Selected Module

- Once the uninterested module is trimmed, the connection from that module becomes dotted as shown in Figure 3.10.

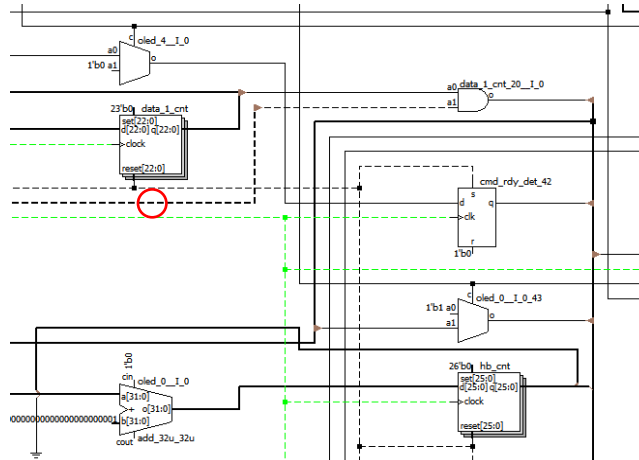


Figure 3.10. Schematic View after Selected Module is Trimmed

- If you want the schematic back, right-click and select **Restore Current Schematic** as shown in Figure 3.11.

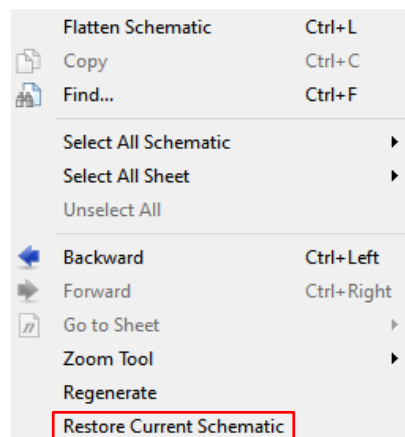



Figure 3.11. Restoring the Schematic View

3.1.3.4. Push in the Hierarchy

Understanding the internal structure of a module is crucial for effective debugging. Netlist Analyzer facilitates this by offering a *Push in the Hierarchy* feature. This functionality allows you to delve deeper into the internal components and connections of the module, providing a more granular view. This capability is particularly valuable when dealing with hierarchical designs, enabling engineers to inspect and debug at different levels of abstraction without losing context.

Perform the following procedure to use the *Push in the Hierarchy* feature:

1. Go inside the hierarchy to see the structure of the module.
2. Hover the mouse on the module. Click the  (Push Down/Pop Up) button.
 - The cursor changes to blue if push down is allowed. Click on the module to push down the hierarchy.
 - The cursor changes to green if pop up is allowed. Click on the module to pop up the hierarchy.
3. Right-click the mouse to stay at the current level.

3.1.4. Signal Tracing Through Pins

Netlist Analyzer provides the ability to track and analyze a specific signal as it travels through different pins and components within a circuit or module. This includes understanding the sequence of pins, components, and connections that the signal traverses.

The following example shows how to trace a signal through pins:

1. Choose a module of interest (no need to select). [Figure 3.12](#) shows the example of module of interest used.

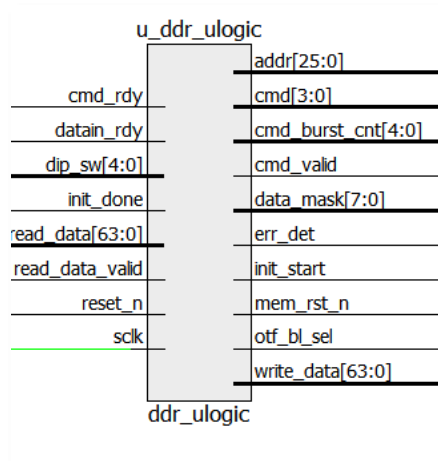


Figure 3.12. Module of Interest

2. Choose a pin inside the module to see the connected logic. In this example, *init_start* pin is selected.
3. Double click on the pin and the logic connected to that pin (the register connected) will be highlighted as shown in [Figure 3.13](#).

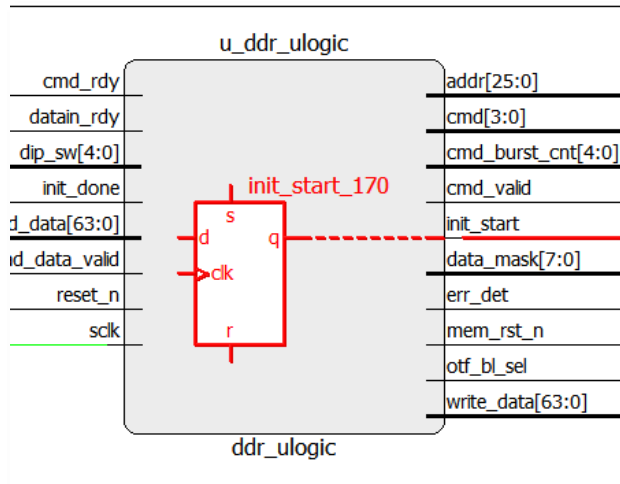


Figure 3.13. Logic Connected to the Selected Pin

- Now select the *d* pin and double-click. The connected logic to the *d* pin of the register gets highlighted as shown in Figure 3.14.

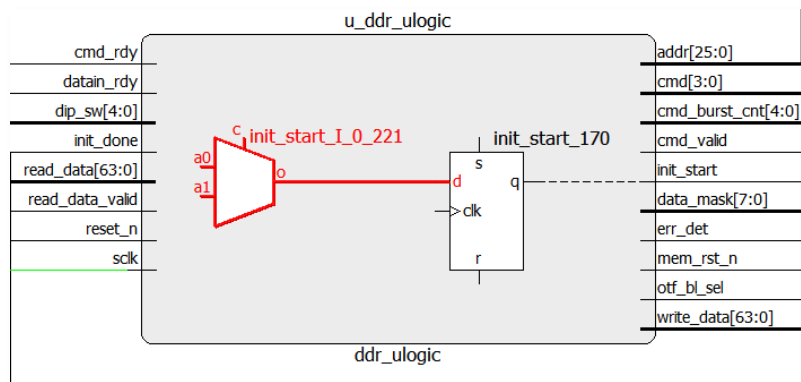


Figure 3.14. Logic Connected to the *d* Pin

3.1.5. Flattening and Unflattening a Design Hierarchy

Flattening the schematic refers to the process of simplifying the hierarchical structure of a design for better visualization and analysis. Flattening and unflattening are powerful functionalities that provide flexibility in navigating and analyzing schematic designs.

Understanding when to use these functionalities is crucial, you need to consider factors such as design size, analysis goals, and the trade-offs involved in flattening the hierarchical modules. Table 3.2 lists the advantages and consideration of flattening a schematic design.

Table 3.2. Advantages and Consideration of Flattening a Design Hierarchy

Advantages	Consideration
<p>Streamlined Visualization:</p> <ul style="list-style-type: none"> Flattening the design displays all the layers together, eliminating the need for hierarchical browsing. 	<p>Signal Tracing Challenges:</p> <ul style="list-style-type: none"> In large designs, signal tracing becomes more complex after flattening, potentially making it difficult to follow the signal paths.
<p>Focused Analysis:</p> <ul style="list-style-type: none"> Allows you to concentrate on the entire design or specific modules without navigating through hierarchical levels. 	

The following sections describe the steps, and example on how to flatten and unflatten a schematic design in Netlist Analyzer:

3.1.5.1. Flattening an Entire Schematic

1. To flatten the entire schematic, right-click on the schematic view.
2. Select the Flatten Schematic from the context menu that appears. This action consolidates all hierarchical layers, presenting the design as a single, flat representation.

3.1.5.2. Unflattening the Schematic

1. To revert to the hierarchical structure, right-click on the schematic and choose **Unflatten Hierarchy**. This action restores the original hierarchical organization, allowing you to navigate through different levels of the design.

3.1.5.3. Flattening a Hierarchical Instance

This feature is used to flatten a specific hierarchical instance (submodule) within the schematic.

1. Select the targeted hierarchical instance.
2. Right-click on the module and choose “Dissolve Instance.” This action breaks down the selected submodule, integrating its contents into the higher-level schematic.

Note: Multiple levels of instances can be dissolved to provide flexibility in the depth of flattening.

3.1.5.4. Unflattening a Hierarchical Instance

1. To reverse the process and restore the hierarchical instance, right-click and choose **Restore Current Schematic**. This action reconstructs the submodule within the hierarchy.

3.1.5.5. Example of Flattening a Module

1. If a module has multiple hierarchy, **dissolve instance** flattens the module once like shown in [Figure 3.15](#).

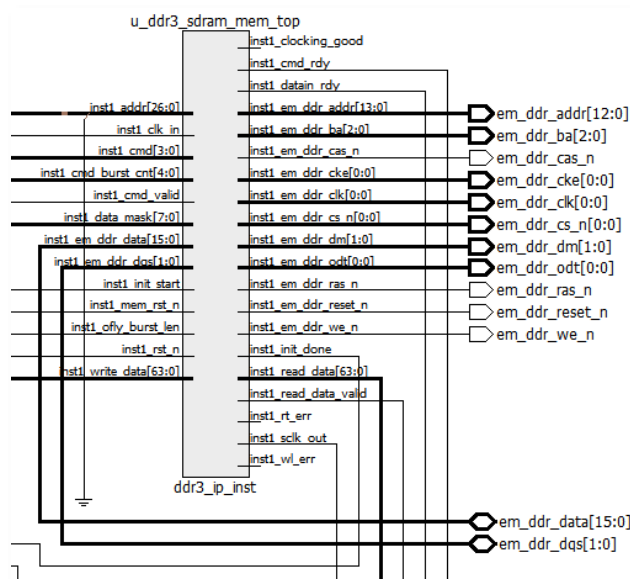


Figure 3.15. Flatten the Module Using Dissolve Instance

- If there is another level of hierarchy, select **dissolve instance** on the inner module to flatten the schematic as shown in [Figure 3.16](#).

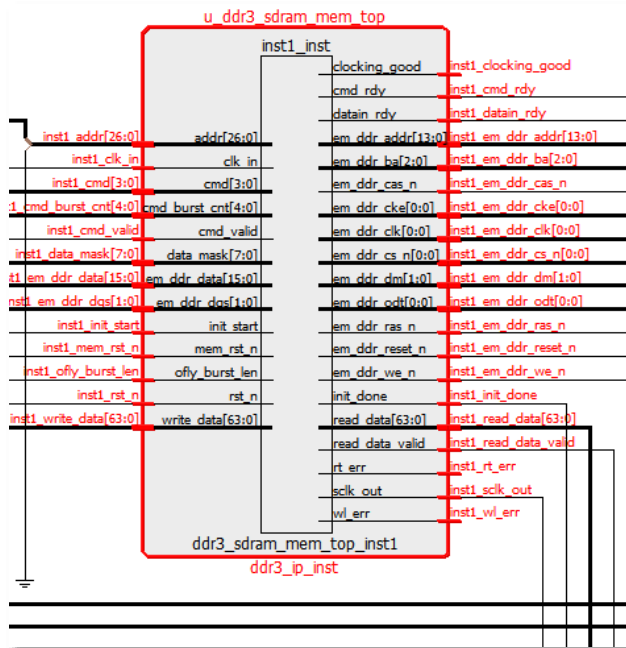


Figure 3.16. Using Dissolve Instance on Inner Module

- To get back to the original schematic, right-click anywhere in the schematic window and choose **Restore Current Schematic** as shown in [Figure 3.17](#).

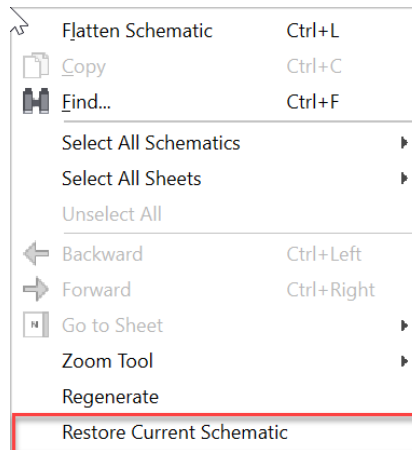


Figure 3.17. Restoring to Original Schematic

3.1.6. Net Tracing

Netlist Analyzer allows you to trace a net from the schematic. Follow below steps to perform net tracing in Netlist Analyzer:

- Click on the net of interest directly from the schematic view. The net will be highlighted in red, making it easily identifiable within the design.
- After selecting the net, right-click on it to reveal a context menu. There are few actions available for you to choose from which are listed below:

- Choosing Jump to HDL File option, shown in Figure 3.18, allows you to view the selected net in the Hardware Description Language (HDL) file, specifically in the RTL representation.

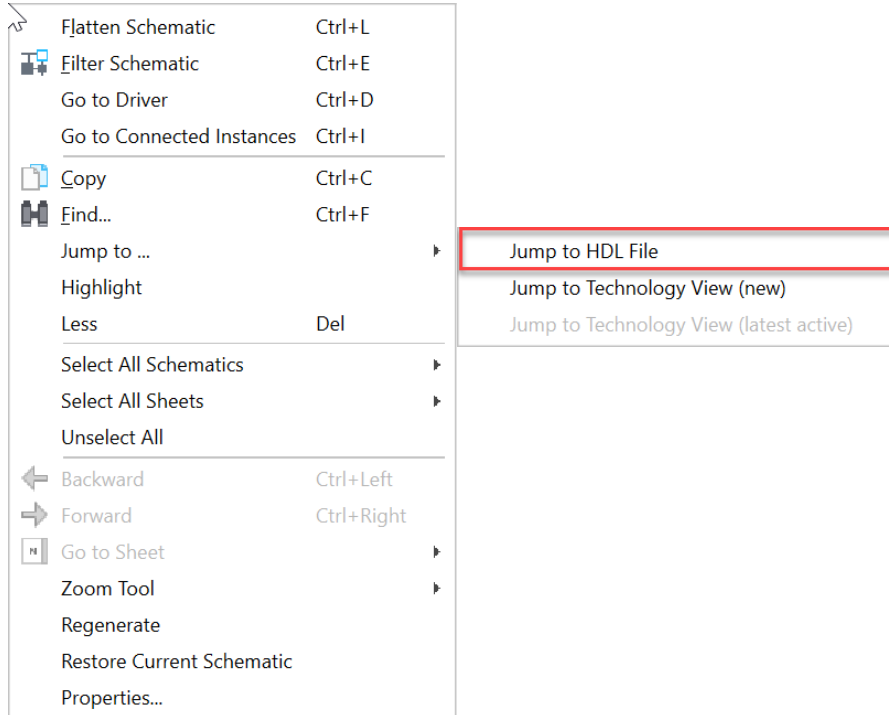


Figure 3.18. Jump to HDL File Option

- Choosing Jump to Technology View option, shown in Figure 3.19, allows you to inspect the net in the context of the technology view, providing insights into physical implementation details.

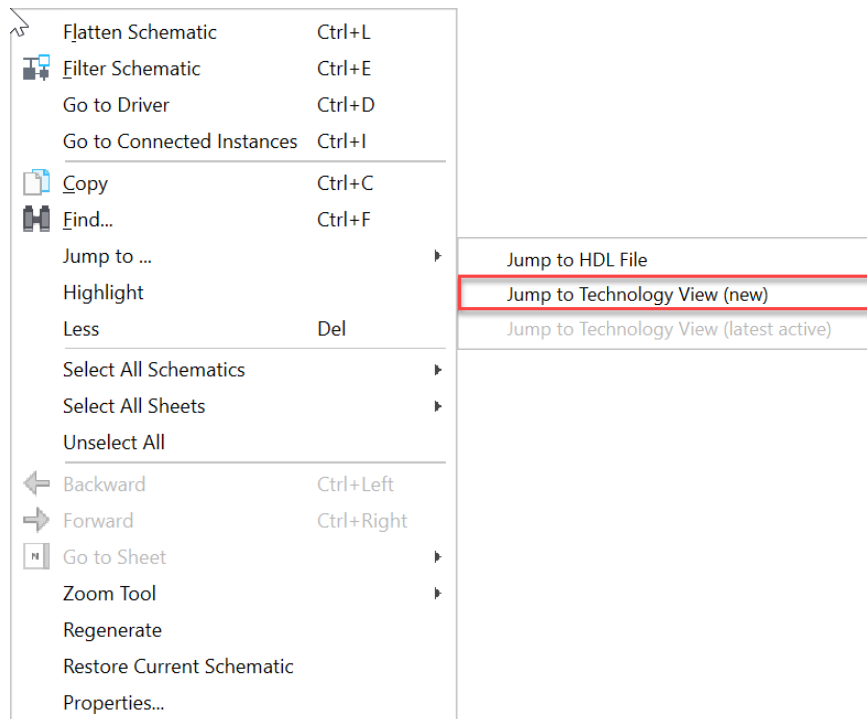


Figure 3.19. Jump to Technology View Option

- Selecting **Go to Driver** option, shown in [Figure 3.20](#), highlights the driver of the chosen net. This is useful to debug multi-driven nets, and to quickly identify and examine the source or driving element of the selected net.

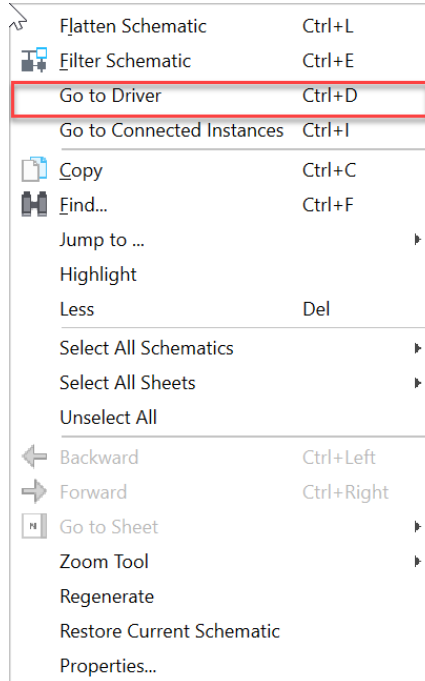


Figure 3.20. Go to Driver Option

- Choosing **Go to Connected Instances** option, shown in [Figure 3.21](#), highlights all the connected instances of the selected net. This aids in understanding the broader context of the net within the overall design and its interactions with other components.

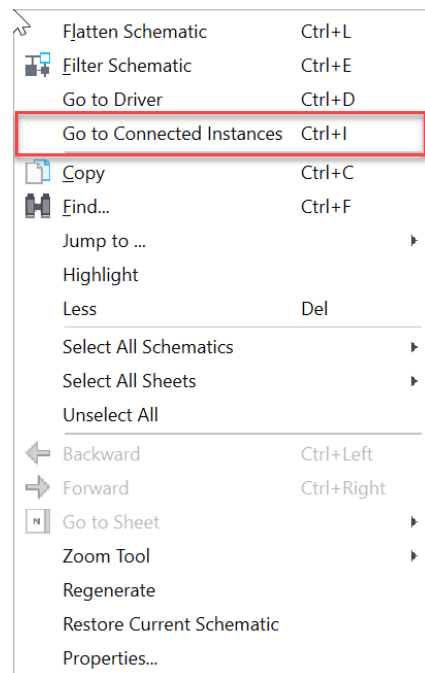


Figure 3.21. Go to Connected Instances Option

- Choosing the **Highlight** option, shown in [Figure 3.22](#), allows you to toggle the visibility of the net’s highlighting. This feature is useful for easy tracking and tracing especially in complex designs where multiple nets may be overlapping. You can revert the highlighted object by choosing **Unhighlight** option.

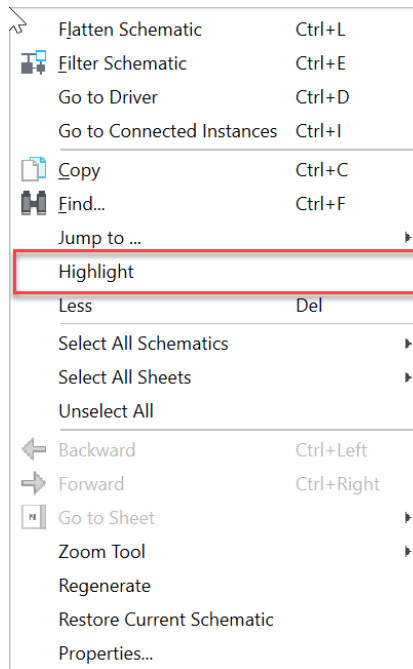


Figure 3.22. Highlight Option

- Selecting **Properties** option, shown in [Figure 3.23](#), provides detailed information about the selected net, including its Fan-out (the number of loads connected to the net), and the specific pins connected to the net. This information is valuable for comprehensive analysis and optimization.

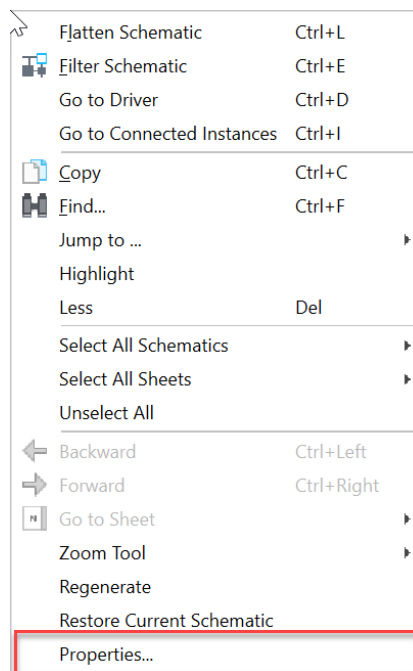


Figure 3.23. Properties Option

3.2. Physical Designer – Placement Mode

Placement Mode is a powerful asset in the floorplanning phase, enabling you to not only visualize the layout but also make informed decisions based on resource utilization and organization. This tool contributes to a more efficient and effective floorplanning process, especially when dealing with complex designs and diverse components.

In this mode, a large-component layout is presented, and all connections are visualized through fly-lines, offering a clear representation of the design’s interconnectivity. Placement Mode offers several features to enhance the floorplanning process and it is available after Synthesis. [Table 3.3](#) describes the key features of Placement Mode:

Table 3.3. Key Features of Placement Mode

Key Features	Descriptions
Creation of REGIONS and Bounding Boxes	You can create REGIONS, defining specific areas of interest on the floorplan. Additionally, bounding boxes for GROUPs can be established to encapsulate related components, facilitating a structured organization of the design.
Customization of Displayed Components and Connections	Placement Mode provides the flexibility to specify the types of components and connections to be displayed. This customization allows you to focus on relevant aspects of the design, streamlining the visualization process.
Interactive Tool Tips	As the mouse pointer traverses the floorplan layout, tool tips dynamically provide essential details. This includes information such as the number of resources allocated for each GROUP and REGION, the utilized slices for each PLC (Programmable Logic Cell) component, and precise details about components, ports, nets, and sites.
Resource Utilization Insights	Placement Mode offers insights into the resource utilization of different design elements. This includes details on the number of resources assigned to each GROUP and REGION, as well as the specific utilization of slices for PLC components. These insights empower you to make informed decisions during the floorplanning process.
Availability Upon Device Specification	Placement Mode becomes available as soon as the target device is specified. This ensures that you can seamlessly transition into floorplanning activities, leveraging the visualization and organizational features provided by Placement Mode.

Floorplanning is a critical step in the design optimization process, addressing issues related to performance, routing delays, and congestion. The success of floorplanning relies on a comprehensive understanding of both the target device architecture, and the intricacies of the design being implemented. Additionally, leveraging the capabilities of optimization tools in a sequential manner ensures efficient and effective floorplanning results. [Table 3.4](#) lists the situations when you should consider doing floorplanning and how floorplanning helps to address the issues:

Table 3.4. When to Do Floorplanning

Situations	Reasons
Desired performance cannot be met	Floorplanning becomes crucial when the initial design fails to achieve the desired performance goals. It allows for strategic placement of logic elements and resources to optimize the overall design.
Routing delays contribute significantly	When routing delays account for a substantial portion, specifically over 60 percent, of the critical path delays, floorplanning becomes a key strategy. Addressing routing challenges through effective floorplanning can significantly enhance the overall performance of the design.
Congestion issues with available resources	Floorplanning is beneficial in scenarios where congestion is causing issues, and there are ample resources available in the device. Strategic placement can alleviate congestion problems and improve the overall quality of the design.

To ensure efficient and effective floorplanning, it is recommended to do preparations that are listed in [Table 3.5](#):

Table 3.5. Preparation Before Floorplanning

Preparations	Reasons
Know Target device architecture before floorplanning	Understanding the architecture of the target device is crucial before initiating the floorplanning process. Different devices may have unique features and constraints, and aligning the floorplan with the device architecture is essential for optimal results.
Have detailed knowledge of the design	In-depth knowledge of the implemented design is a prerequisite for effective floorplanning. This includes understanding critical paths, signal paths, and dependencies within the design. A detailed understanding facilitates informed decisions during the floorplanning phase.

[Table 3.6](#) lists the items you can consider during your floorplanning:

Table 3.6. Considerations During Floorplanning

Considerations	Reasons
Tool optimization and constraint adherence	Emphasize that the tool is designed to produce optimized results while adhering to specified constraints. Trusting the tool to handle the initial placement is recommended, as it leverages algorithms to strategically place components for optimal performance.
Sequential approach	It is advisable to let the tool perform the initial placement before manually intervening. This sequential approach ensures that the tool's intelligence is utilized to its full potential before fine-tuning or making manual adjustments during the floorplanning process.

3.2.1. An Example of When to Use Floorplanning

Consider the following timing summary example. In this case, the routing delay is 80% and the logic delay is 20%. In this case, floorplanning might be useful.

```

Path Begin      : count_reg[8].ff_inst/Q (SLICE_R23C48A)
Path End       : comp_reg_reg[1].ff_inst/CE (SLICE_R30C42C)
Source Clock   : CLK ®
Destination Clock: CLK ®
Logic Level    : 6
Delay Ratio    : 80.3% (route), 19.7% (logic)
Clock Skew    : -0.633 ns
Setup Constraint : 10.000 ns
Common Path Skew : 0.606 ns
Path Slack     : -3.2 ns (Failed)
    
```

The register comp_reg_reg can be brought closer to count_reg[8] to reduce the routing delay for this path which may help in meeting the timing requirement. Refer to the Radiant software's help tool on *Physical Designer Placement Mode* for steps on how to do floorplanning.

3.2.2. An Example of When Not to Use Floorplanning

Consider the following timing summary example. In this case, the routing delay is ~25% and the logic delay is ~75%. In this case, floorplanning is not effective.

```

Path Begin      : count_reg[4].ff_inst/Q (SLICE_R23C43A)
Path End       : comp_reg_reg[2].ff_inst/CE (SLICE_R23C54C)
Source Clock   : CLK ®
Destination Clock: CLK ®
Logic Level    : 6
Delay Ratio    : 24.4% (route), 74.6% (logic)
Clock Skew    : -0.633 ns
Setup Constraint : 10.000 ns
Common Path Skew : -1.606 ns (Failed)
    
```

You need to follow other timing closure techniques to close the timing on this path. Refer to [Timing Closure Techniques for Mid-Size FPGAs \(FPGA-AN-02074\)](#) document for more information on timing closure techniques.

3.3. Physical Designer – Routing Mode

Routing Mode offers a comprehensive, read-only view of the design, providing intricate details that include switch boxes and the physical wire connections within the layout. This mode presents a detailed and realistic representation of the routed connections, utilizing Manhattan-style lines for established connections and flylines for those yet to be routed.

[Table 3.7](#) describes the key Features of Routing Mode:

Table 3.7. Key Features of Routing Mode

Key Features	Descriptions
Detailed Connection Visualization	Routed connections are prominently displayed as Manhattan-style lines, offering a clear and detailed representation of the physical pathways of signals within the design. Unrouted connections are indicated by flylines, providing a visual cue for connections yet to be established.
Interactive Layout Exploration	Hovering over the mouse pointer slowly over the layout, tool tips dynamically reveal valuable information. Details such as the name and location of each REGION, group, component, port, net, and site are displayed. This interactive feature enhances the exploration and understanding of the design's layout by providing real-time information about the various elements.
Switch Box Representation	Routing Mode includes the visualization of switch boxes within the layout. This feature allows designers to identify and understand the placement and interconnection of switch boxes, critical components in managing signal routing and directing the flow of data within the design.
Enhanced Design Understanding	The detailed layout presented in Routing Mode contributes to an enhanced understanding of the physical structure of the design. Designers can easily comprehend the spatial relationships and connections between different design elements, fostering better decision-making during the design refinement process.
Facilitating Debugging and Optimization	Routing Mode is a valuable tool for debugging and optimization efforts. By providing a detailed, visual representation of routed connections, designers can quickly identify potential issues, assess the efficiency of the routing paths, and make informed decisions to optimize signal integrity and performance.

3.3.1. Analyzing Net Delay in the Physical View Routing mode

In the Physical View Routing Mode, you have the capability to conduct a detailed analysis of the net delay associated with a selected net, offering insights into both the driver and the delay to the pins. The following steps outline how to view the net delay effectively:

1. Navigate to the Physical View of the design to delve into the detailed characteristics of individual nets and their corresponding delays.
2. Identify the net of interest within the Physical View. This could be a crucial signal path that requires in-depth scrutiny.
3. Right-click on the identified net within the Physical View to unveil a contextual menu.

- From the contextual menu, choose the **Pin Pair Delay** option as shown in [Figure 3.24](#). This action initiates the analysis process, providing specific information related to the net's delay characteristics.

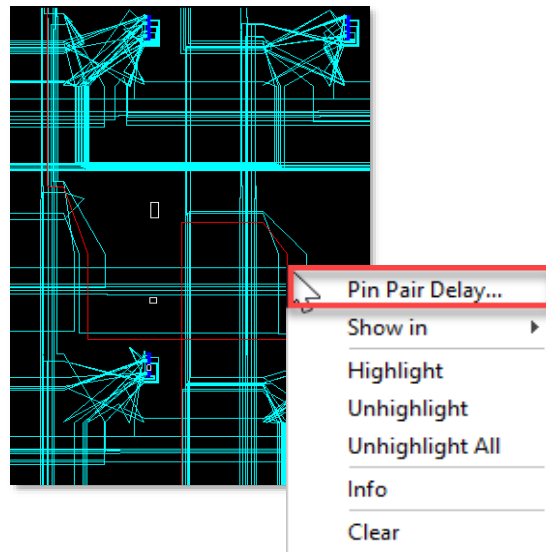


Figure 3.24. Pin Pair Delay Option

- Upon selecting **Pin Pair Delay** option, the output window is populated with valuable details pertaining to the net under consideration as shown in [Figure 3.25](#). The output window provides the following information:
 - Information about the driver of the selected net which includes details about the source of the signal, facilitating a clear understanding of the net's origin within the design.
 - Information regarding the delay of the net as it propagates towards its destination pins which includes insights into the time it takes for the signal to traverse the entire net. This is crucial for optimizing signal integrity and overall performance.

```
Net "u_ddr3_sdram_mem_top/inst1_inst/U1_inst1/U1_ddr3_sdram_phy/p_read3":  
  driver - Pin "R61C10D.Q1"  
  1.221ns - Pin "R72C11A.M0"  
  1.345ns - Pin "R72C11D.B1"
```

Output	Td Console	Error	Warning*	Info
--------	------------	-------	----------	------

Figure 3.25. Output Window Display

3.3.2. Examples of Using Routing Mode to Debug a Design

3.3.2.1. Simple Inverter Example

1. Consider the following RTL code example of a simple inverter. In this RTL code, synthesis attribute `/*Synthesis syn_useioff=1*/` is used on the output register `out` as shown in [Figure 3.26](#).

```
module top ( in, d, out);  
  input in;  
  input d;  
  output reg out /*synthesis syn_useioff = 1*/;  
  wire clk ;  
  wire wire_clk;  
  wire inv_clk/*synthesis syn_keep = 1, nomerge =1*/ ;  
  
  INV inverter (.A (in),  
               .Z (clk)  
               );  
  
  always @(posedge clk) begin  
    out <= d;  
  end  
endmodule
```

Figure 3.26. RTL Code of a Simple Inverter

2. Observe that post PAR, Physical Designer Routing Mode shows that the attribute is not honored by the tool as shown in [Figure 3.27](#). This gives a good perspective in understanding if there are any modification required in your RTL code.

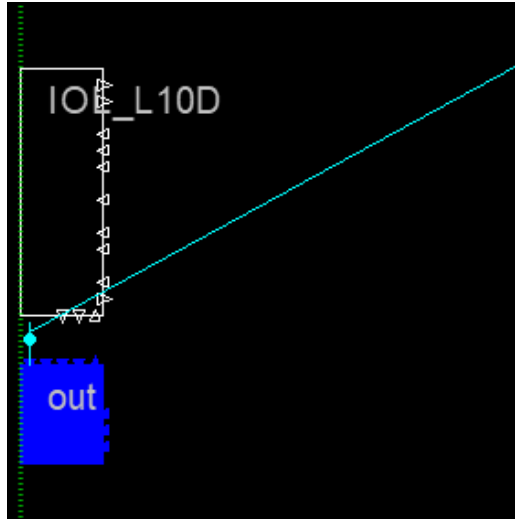


Figure 3.27. Routing Mode Showing the Attribute is Not Honored

3. Modify your RTL code to add a register to make the attribute honored as shown in [Figure 3.28](#).

```
module top ( clk, d, out, clk_out);
  input clk;
  input d;
  output reg out /*synthesis syn_useioff = 1*/;
  output clk_out;
  wire clk_n /*synthesis syn_keep =1, nomerge =1*/;

  reg dummy;
  wire d_in;

  assign d_in = d;

  INV inverter (.A (clk),
               .Z (clk_n)
               );

  always @(posedge clk) begin
    dummy <= d_in;
  end

  always @(posedge clk_n) begin
    out <= dummy;
  end
endmodule
```

Figure 3.28. Modified RTL Code of a Simple Inverter

4. With this implementation, post PAR shows that the tool has honored the attribute as shown in [Figure 3.29](#).

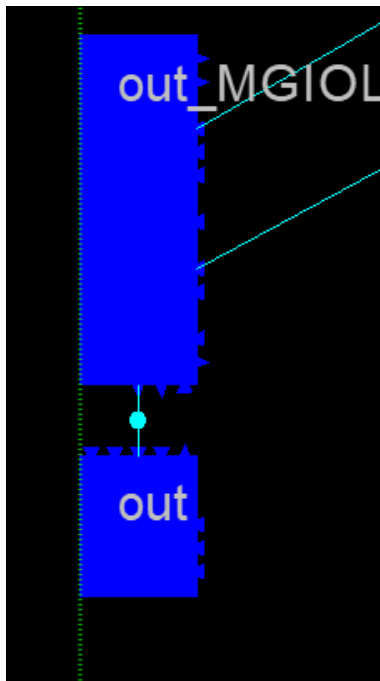


Figure 3.29. Routing Mode Showing the Attribute is Honored

3.3.2.2. One-path Analysis Example

1. Physical Designer Routing Mode also allows you to focus on one path for Analysis. Click on a particular net of interest as shown in [Figure 3.30](#).

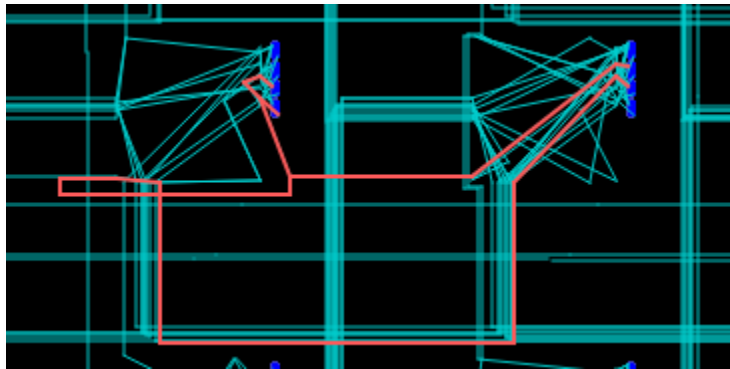


Figure 3.30. Chosen Net of Interest

2. From the menu options, click on the **Routes** icon as shown in [Figure 3.31](#).

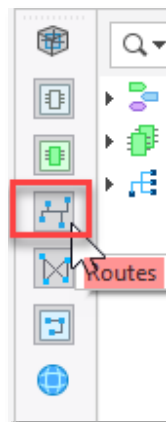


Figure 3.31. Routing Mode Menu Options

3. This removes all the other routes from the view allowing you to focus on the route of interest as shown in [Figure 3.32](#).

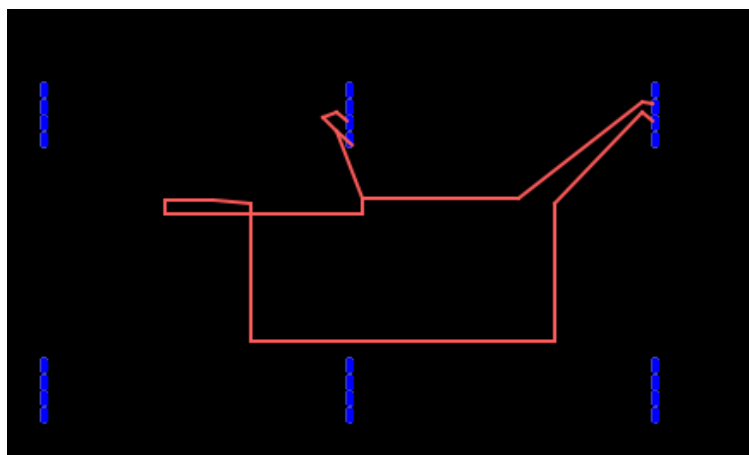


Figure 3.32. Route of Interest

3.4. Physical Designer – Cross-Probing

Another important use of Physical Designer is the ability to cross probe between Placement Mode and Routing Mode. You can do cross-probing by following the steps below:

1. Click on a component in the Placement Mode as shown in [Figure 3.33](#).

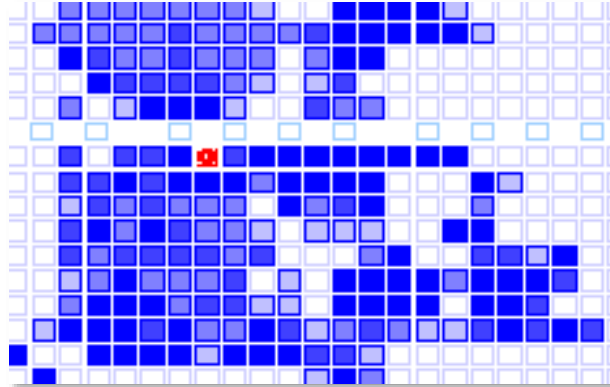


Figure 3.33. Selected Component in the Placement Mode

2. Change the view to Routing Mode to view the component in Physical View as shown in [Figure 3.34](#).

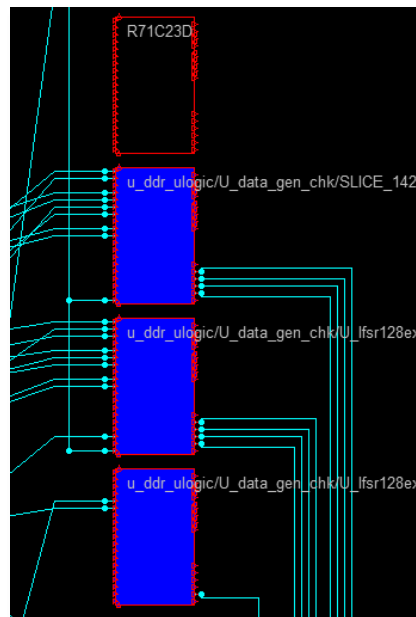


Figure 3.34. Physical View of the Component

Cross-Probing also works for nets. Follow the steps below to do cross-probing on nets:

1. Click on a net of interest in one mode as shown in [Figure 3.35](#).

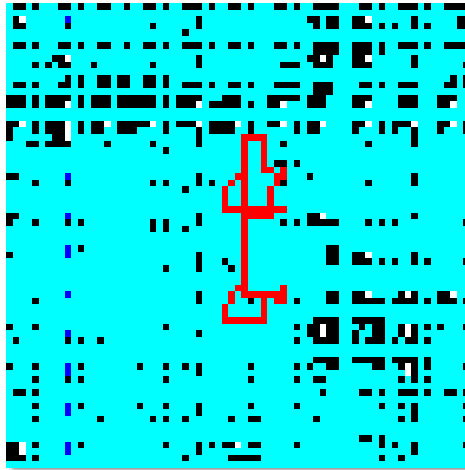


Figure 3.35. Selected Net of Interest

2. Change the mode to another mode to see the net in a different mode as shown in [Figure 3.36](#).

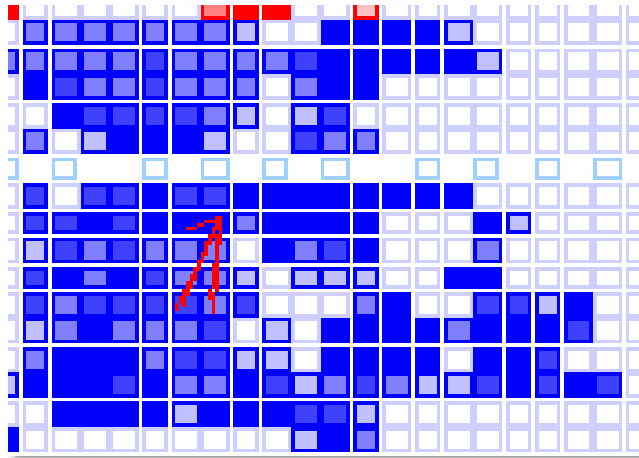


Figure 3.36. Placement Mode View

References

- [Lattice Radiant Timing Constraints Methodology \(FPGA-AN-02059\)](#)
- [Timing Closure Techniques for Mid-Size FPGAs \(FPGA-AN-02074\)](#)
- [Lattice Radiant Software](#) web page
- [Lattice Insights](#) web page for Lattice Semiconductor training courses and learning plans

Technical Support Assistance

Submit a technical support case through www.latticesemi.com/techsupport.

For frequently asked questions, refer to the Lattice Answer Database at www.latticesemi.com/en/Support/AnswerDatabase.

Revision History

Revision 1.0, February 2024

Section	Change Summary
All	Initial release.



www.latticesemi.com