Timing Closure Techniques for Mid-Size FPGAs

Application Note

FPGA-AN-02074-1.1

April 2025

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 REDUDANCIES, 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

Figures

Tables

Abbreviations in This Document

A list of abbreviations used in this document.

Abbreviation	Definition
CDC	Clock-Domain Crossing
CPE	Constraint Propagation Engine
DRC	Design Rule Check
DSP	Digital Signal Processors
EBR	Embedded Block RAMs
IP	Intellectual Property
LSE	Lattice Synthesis Engine
MPW	Minimum Pulse Width
PAR	Place and Route
PFU	Programmable Functional Units
PIC	Programmable I/O cells
PLL	Phase-Locked Loop
PVT	Process, Voltage, Temperature
QoR	Quality of Result
RAM	Random Access Memory
STA	Static Timing Analysis
TCE	Timing Constraint Editor

1. Introduction

As more functionalities and features are offered in advanced technology nodes, timing closure has become a major bottleneck for logic designers. Any delay in meeting timing requirement will result in a much tighter time-to-market window. This document describes the methodologies in achieving optimal timing performance with Lattice FPGAs using best design practices alongside numerous tool settings provided by the Lattice Radiant™ software. While all the techniques described in this document can be applied to designs targeting various FPGA densities, these techniques are more impactful for mid-size to larger FPGAs as timing requirements are usually higher.

1.1. Timing Closure Consideration

When it comes to meeting timing requirement of complex designs, always plan ahead. A well-planned project will avoid last-minute surprises and greatly reduce the number of iterations to meet performance requirements. The following factors might impact the feasibility and difficulty in timing closure later in the design flow.

1.1.1. Device Selection

Every digital logic design has its own design specifications that will influence intellectual property (IP) selections, device variant and density, ease of migration, speed grade, and package offerings. The IP cores provided by the Radiant software can come in soft or hardened IP cores, each with its pros and cons. Ensure the device of choice has sufficient logic resources not only for functionality but also to meet timing requirement. Be aware of the locations of hard logic resources such as digital signal processors (DSPs), RAMs, and transceivers as placement and routing around these blocks are less flexible. A less congested design is always easier to achieve full timing closure.

1.1.2. Complexity of the Design

Be realistic when creating your designs. An FPGA with abundant of logic resources does not guarantee timing closure if the design has a great number of paths with long combinational logic. Also, if a timing-critical logic node has huge fanouts, the tool will have a hard time to place-and-route (PAR) the downstream logic optimally to meet timing. Add pipeline whenever possible as this will make the design less dependent on the outcome of PAR, making timing closure much easier.

1.1.3. Timing Guardband

When targeting a high-speed design, aim to achieve the highest possible Fmax. This is especially true when working independently on sub-module designs before they are integrated to the final project. It is common to encounter timing degradation when sub-modules are integrated into a fuller design because certain logic nodes might fight for the common placement and routing resources. It is less time-consuming and easier to optimize a smaller sub-module design when the designs are open to modifications.

1.1.4. Other considerations

Timing performance usually conflicts with other Quality of Result (QoR) metrics such as area and power. To achieve better timing performance, pipelining might be necessary. Also, the tool sometimes need to duplicate logic nodes across different parts of the device.

Adding debugging signals into the design such as the case of Reveal Inserter and Reveal Analyzer will also incur additional logic resources.

2. Constraining the Design

The purpose of Static Timing Analysis (STA) is to check for all possible paths in a design for timing violations, which consist of setup and hold timing checks. As the Lattice Radiant software's algorithm is timing driven, a properly constrained design not only ensures the design is fully functional in hardware but also provides guidance on the tools that can help with timing optimization and closure. An over-constrained or under-constrained design can make timing closure more challenging.

2.1. Fundamental of Timing Constraints

In general, the following three categories of timing constraints would be sufficient for any design to be fully constrained and timing-analyzed.

2.1.1. Clock Constraints

- Base clocks: Use **create_clock** to create base clocks in your designs. It is recommended to define clocks on the top-level ports
- Virtual clocks: Use **create_clock** without a target port to constrain virtual clock, which is used for I/O timing analysis
- Derived clocks: Use **create_generated_clock** to constrain clocks that are based off other clocks. This includes the outputs of phase-locked loops (PLLs), source synchronous outputs, clock multiplexers, and ripple clocks.

2.1.2. I/O Timing Constraints

Constrain all input and output delays with reference to virtual clocks. Use **set_input_delay** and **set_output_delay** to describe the circuit external to the FPGA:

- Input paths: Use **set_input_delay** to constrain input path delay on top-level input ports
- Output paths: Use **set_output_delay** to constrain input path delay on top-level input ports

On the input ports, the virtual clock will be the launch clock while on the output ports, it will be the latch clock.

2.1.3. Timing Exceptions

With the two types of constraints mentioned above, all the clocks in the design should be fully constrained. By default, all clocks are related and the final step is for you to set timing exceptions to relax the timing requirements. This will greatly help the tool and timing engine in the timing closure process. The following lists some examples of commonly used timing exceptions:

- **set_false_path**: This command tells the Radiant software not to analyze path(s) of interest. It could be applied between registers or between clocks. For the latter, all paths clocked by the targeted clocks will not be timing-analyzed.
- **set_clock_group**: This command is similar to set_false_path, where it instructs the Radiant software clocks that are asynchronous or not related. Timing analysis between the specified clock groups will be disabled. No paths are reported between the clock groups in both directions
- **set_max_delay/set_min_delay**: This command overrides the default setup and hold relationships, respectively. This constraint can be used to specify any arbitrary maximum or minimum delay for a given path without being associated with its clock period.
- **set_multicycle_path**: There are scenarios where the default setup and hold relationships can be modified. This command modifies the launch and/or latch clock edges. It can be applied between registers or clocks.

Figure 2.1 shows three types of timing constraints.

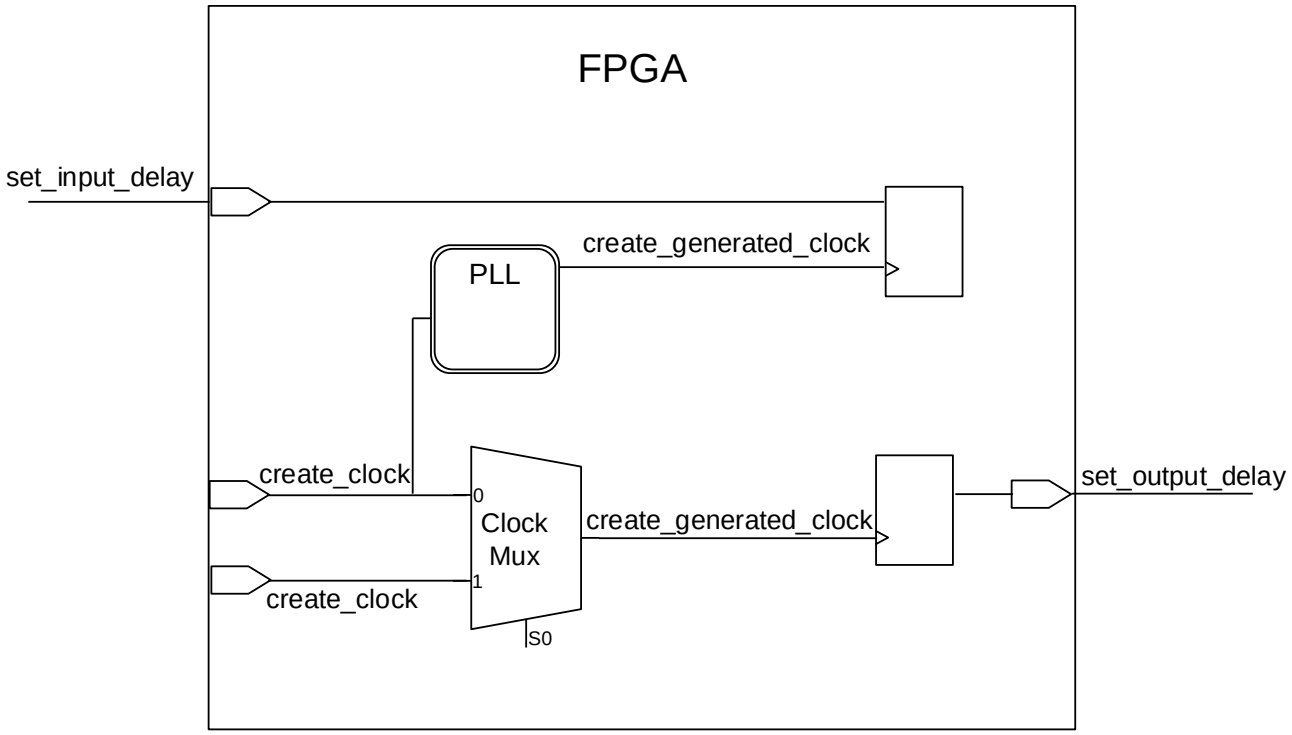


Figure 2.1. Three Types of Timing Constraints

For more information, refer to the *Lattice Radiant Timing Constraints Methodology Application Note (FPGA-AN-02059)*.

3. Timing Closure with the Lattice Radiant Software

The Lattice Radiant software provides a complete design environment to insert timing constraints and will work its way to meet the performance based on the provided timing constraints. The Lattice Radiant software is timing-driven, which means timing constraints you provide will be used in all implementation stages, from synthesis to place-and-route.

3.1. Timing Constraint Files

The Lattice Radiant software supports several design constraint files depending on the synthesis tool selected and whether it is for pre-synthesis or post-synthesis consumption.

Table 3.1. Design Constraint Files Supported in the Radiant Software

	Constraint File	Tool
Pre-Synthesis	Lattice Design Constraints (*.ldc*)	Lattice Synthesis Engine (LSE)
	FPGA Design Constraints (*.fdc*)	Synplify Pro®
	Synopsys Design Constraints (*.sdc*)	LSE and Synplify Pro
Post-Synthesis	Physical Design Constraints (*.pdc*)	Map design (MAP) and Place & Route (PAR) design, which includes timing and physical constraints

There are two types of constraints – **timing constraints** are timing requirements for the design while **physical constraints** affects the physical layout of the netlist.

The Lattice Radiant software provides a convenient way for you to insert timing constraints through the built-in Timing Constraint Editor (TCE):

- Pre-synthesis Timing Constraint Editor – timing constraints such as clocks, input/output timing constraints and timing exceptions, attributes, and macros. When invoked, the design will be compiled quickly and you can constrain the design objects such as pins, ports, registers, and nets conveniently.
- Post-synthesis Timing Constraint Editor – timing constraints for post-synthesis netlist and physical constraints. These constraints will be saved into a *.pdc* file where you can also edit it directly. This file will be consumed by the MAP and PAR processes.

For more information, refer to the *Lattice Radiant Timing Constraints Methodology Application Note (FPGA-AN-02059)*.

3.2. Constraint Propagation

As described in the Constraining the Design section, it is recommended to create clocks on top-level ports. However, constraints that are defined on Lattice IP might not contain the correct hierarchical names when integrated to a larger design. Therefore, constraints are not applied correctly during synthesis. To handle this problem, the Lattice Radiant software provides a Constraint Propagation Engine (CPE) that runs a design rule check (DRC) on all input constraints and translates them into a new constraint file in a way that all IP constraints are honored in top-level module.

For more information about constraint propagation, refer to *Constraint Propagation* in the Radiant Software Help.

Figure 3.1 summarizes how timing constraints are consumed by the Lattice Radiant software—from constraint creation to constraint propagation and the files that are generated in each stage of the software flow. Note the transition from the logical to physical domain.

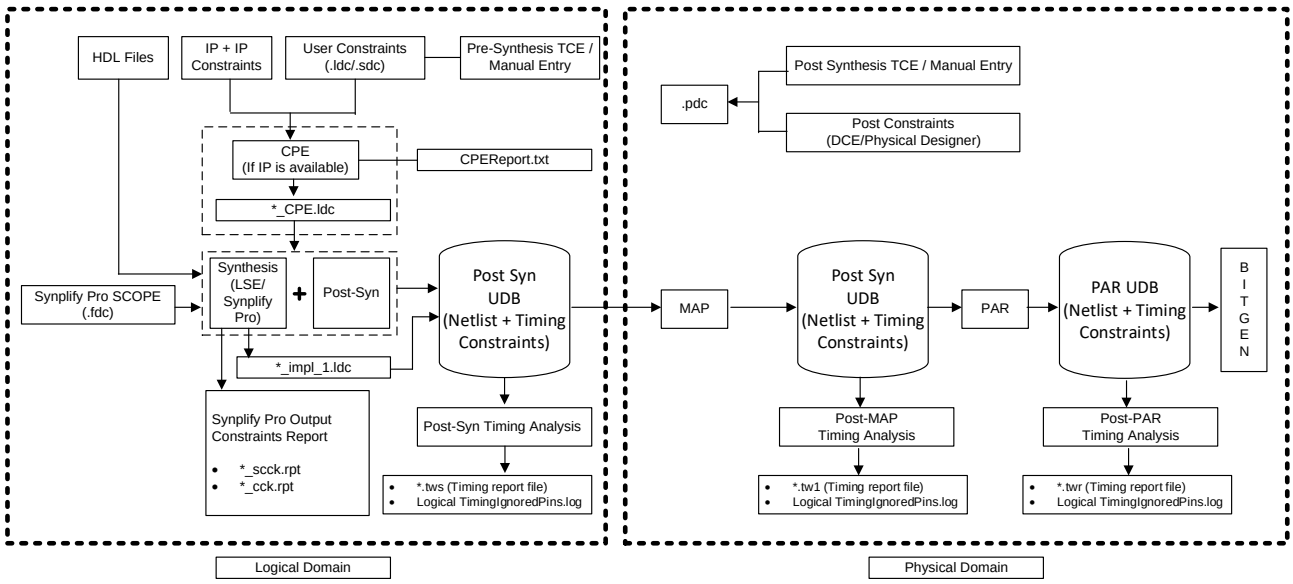


Figure 3.1. Lattice Radiant Software Timing Constraint Flow

3.3. Timing-Driven Flow with the Lattice Radiant Software

All the compilation flows in the Lattice Radiant software (For example, synthesis, MAP, and PAR) are timing-driven. In other words, proper timing constraints are necessary in driving the tool towards meeting the desired timing requirements.

In the Radiant software, a *strategy* is a collection of settings that impact the different stages of the implementation process from synthesis to MAP and PAR. Strategies can influence whether the design is optimized for timing or area and PAR run time. The first time you create a Radiant software project, **Strategy1** appears as the default strategy. To switch to the area-focused strategy, go to **File List Panel → Area → Set as Active Strategy**, as shown in Figure 3.2.

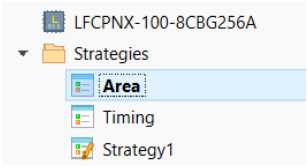


Figure 3.2. Switching to Area-Focused Strategy from the File List Panel

Changing between **Area** and **Timing** strategies will turn on/off certain Strategy options under the hood, which are discussed in the following sections.

3.3.1. Timing-Driven Synthesis

Both the LSE and Synplify Pro consume pre-synthesis timing constraints supplied by users to optimize circuit performance during synthesis. This is the default behavior in the Radiant software but can be changed to prioritize logic saving (area) if timing performance is not important.

3.3.1.1. Timing-Driven Synthesis with Synplify Pro

To ensure that the Synplify Pro is timing driven, you must select the correct strategy and its settings. For example, the **Area** setting is disabled (i.e *False*) when the **Timing** strategy is chosen, or you can customize your own settings to your preference, as shown in Figure 3.3.

Also, note how the **Frequency** option is set to 200 MHz. This is the default frequency used for Lattice Avant™ and Nexus™ devices if you do not provide any constraints to the associated clocks. If all the clock constraints are provided, the **Frequency** option is ignored. This setting is also ignored when **Area** is set to *True*.

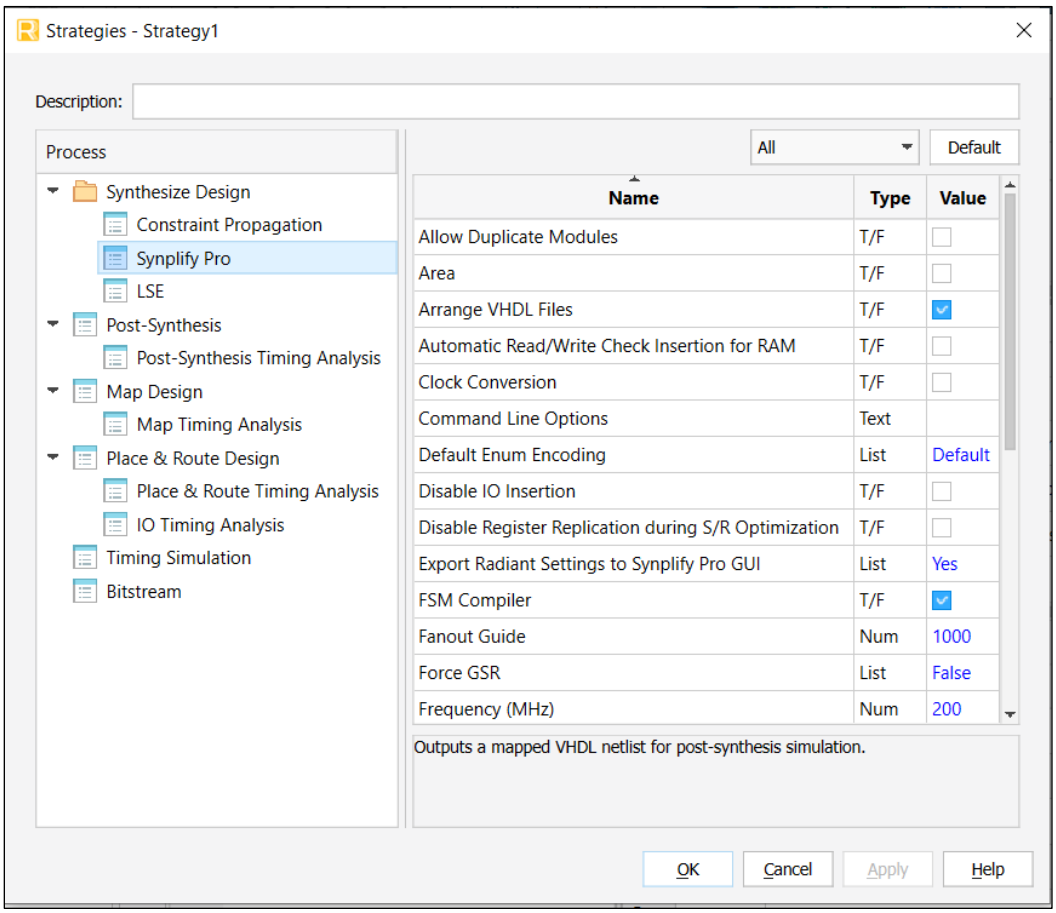


Figure 3.3. Synplify Pro Synthesis Settings

3.3.1.2. Timing-Driven Synthesis with LSE

Similar to Synplify Pro, when timing-driven strategy is chosen, the **Optimization Goal** setting will be set to *Timing,* as shown in Figure 3.4.

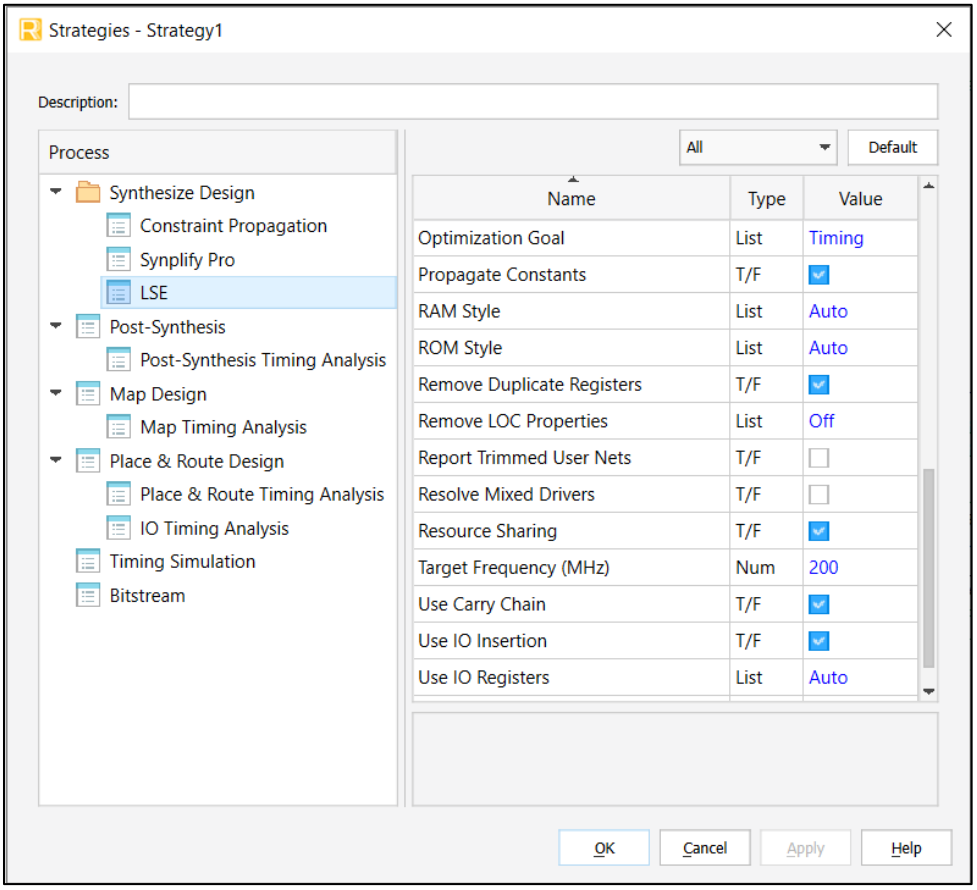


Figure 3.4. LSE Synthesis Settings

Other LSE synthesis settings may also be beneficial for timing performance but can be very design-dependent. Therefore, you might need to experiment on what works best for your designs. Table 3.2 lists the available settings that affect area or speed of a design.

Table 3.2. LSE Strategy Settings for Area and Timing

Settings	Area	Timing
FSM Encoding Style	Binary or Gray	One-Hot
Max Fanout Limit	<Maximum>	<Minimum>
Remove Duplicate Registers	True	False
Resource Sharing	True	False
Use I/O Registers	Auto or True	Auto or False

3.3.2. Timing-Driven MAP and PAR

Map design is the stage where your design that consists of device-independent components such as gates and flip-flops are converted into device-specific components such as programmable functional units (PFUs), embedded block RAMs (EBRs), and programmable I/O cells (PICs).

Place & Route (PAR) design consists of placement where device-specific components are assigned to specific locations on the device floorplan. Then, the route process will create physical connections through routing interconnects to join the placed components. The timing-driven PAR engine uses the provided constraints and settings to meet the design's timing requirement.

If the timing-based strategy is chosen, the *Disable Timing Driven* will be turned off (see Figure 3.5). If enabled, cost-based placement and routing are done instead. One scenario to disable the timing-driven option is when you would like to save run time to get a rough idea on the level of difficulty in placing and routing the design on the selected device.

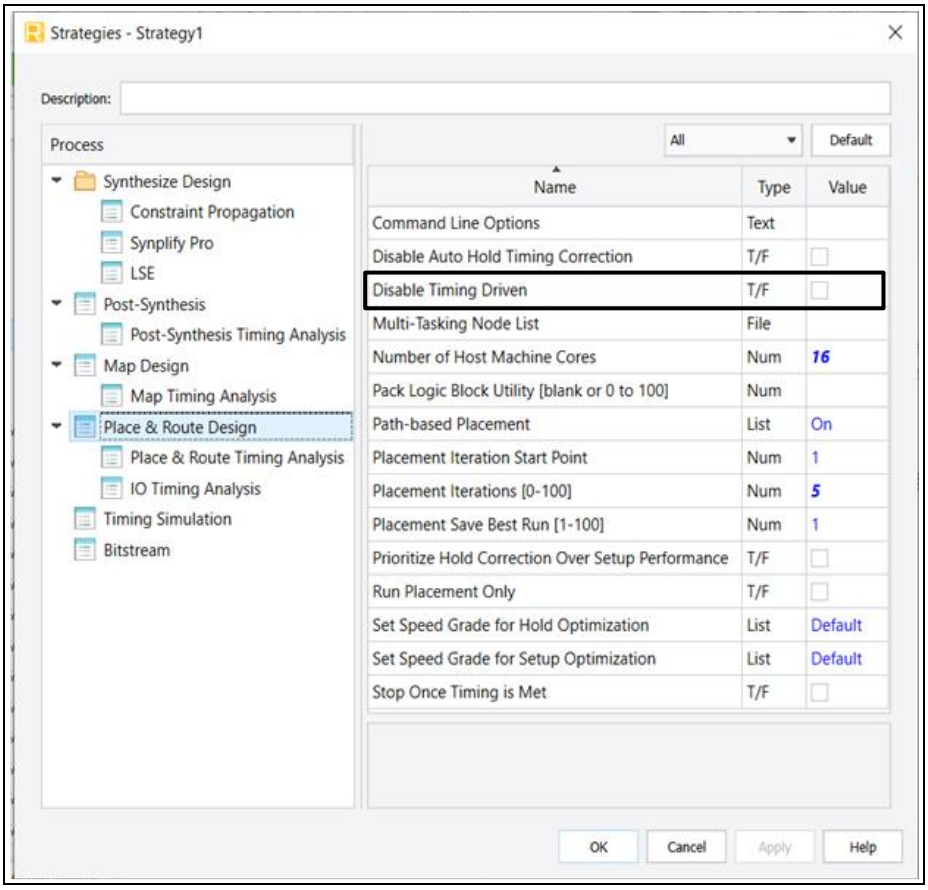


Figure 3.5. Place & Route Design Settings

Table 3.3 lists other options that would affect timing performance.

Table 3.3. Place & Route Design Settings

Options	Description
Disable Auto Hold Timing Correction	When turned on, the tool will not correct potential hold timing violations.
Pack Logic Block Utility (Not available in Avant devices)	Define from 0 to 100%, how slices in a device are packed. (0 = max density, 100 = min density). If a value is not specified, it defaults to a percentage based on the selected device. Choosing a value of 0 results in the densest packing could negatively impact design performance.
Path-Based Placement (Not available in Avant devices)	Enabling this setting yields better performance and predictability.
Prioritize Hold Correction Over Setup Performance	By default, hold violations are not prioritized to preserve setup performance. Enabling this option could negatively impact setup performance.
Run Placement Only	PAR will only run placement.
Placement Iteration Start Point	Also known as seed, having different values affect the placement of the logic nodes and hence will affect final timing result.
Placement Iteration [1-100]	Set the number of PAR iterations (0 = run until solved) to be run. If a start point value is specified, the iteration begins at that value.

3.4. Static Timing Analysis in the Radiant Software

Static timing analysis (STA) analyzes the timing of signals in a design and ensure they meet the specified timing requirements provided through timing constraints discussed earlier. These constraints include setup time, hold time, I/O timing, and clock frequency requirements. You can identify potential timing violations and make necessary adjustments to meet the desired performance goals.

The Radiant software computes the arrival times and the required times of signals at various points in the circuit. It checks if the arrival time at a particular point meets the required time for that point, considering the specified constraints. If any timing violations are detected, such as setup or hold violations, you can identify the problematic areas and make modifications to resolve the problems.

STA can be performed at any implementation stages—post-synthesis timing analysis, map timing analysis, and place & route timing analysis. This can be optionally enabled on the Radiant software's *Task Detailed View* window as shown in Figure 3.6. Note that starting in the Radiant software version 2023.2 onwards, PAR static timing analysis for Nexus and Avant devices is not optional and is part of the default flow.

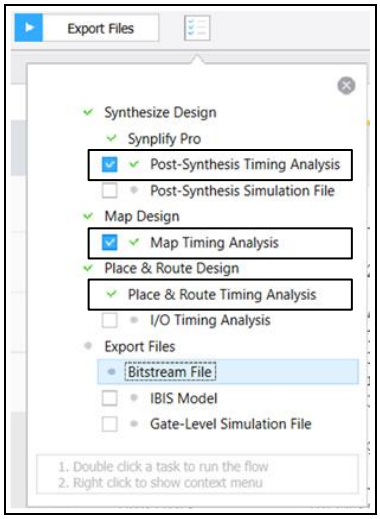


Figure 3.6. Enabling STA at Different Stages

Table 3.4 provides a summary of the different stages of STA.

Table 3.4. Different Stages of Static Timing Analysis

	Post-Synthesis STA	Map STA	PAR STA
Filename extension	.tws	.tw1	.twr
Netlist used	Estimated delays (without place-and-route information)	Actual types of components and estimated routing delays	Based on actual place and route netlist
Accuracy	Least accurate	Moderately accurate	Most accurate
Usage	Identifying constraint issues (for example, syntax error)	Assessing preliminary timing performance (for example, long logic level)	Detailed timing analysis and signoff

3.4.1. Understanding Timing Reports in the Radiant Software

After timing analysis is selected and completed in the Radiant software, a timing report is produced. The timing report format from post-synthesis, MAP, and PAR is almost identical. Because PAR timing analysis must be used as the final signoff, you must refer to the Place & Route Timing Analysis report.

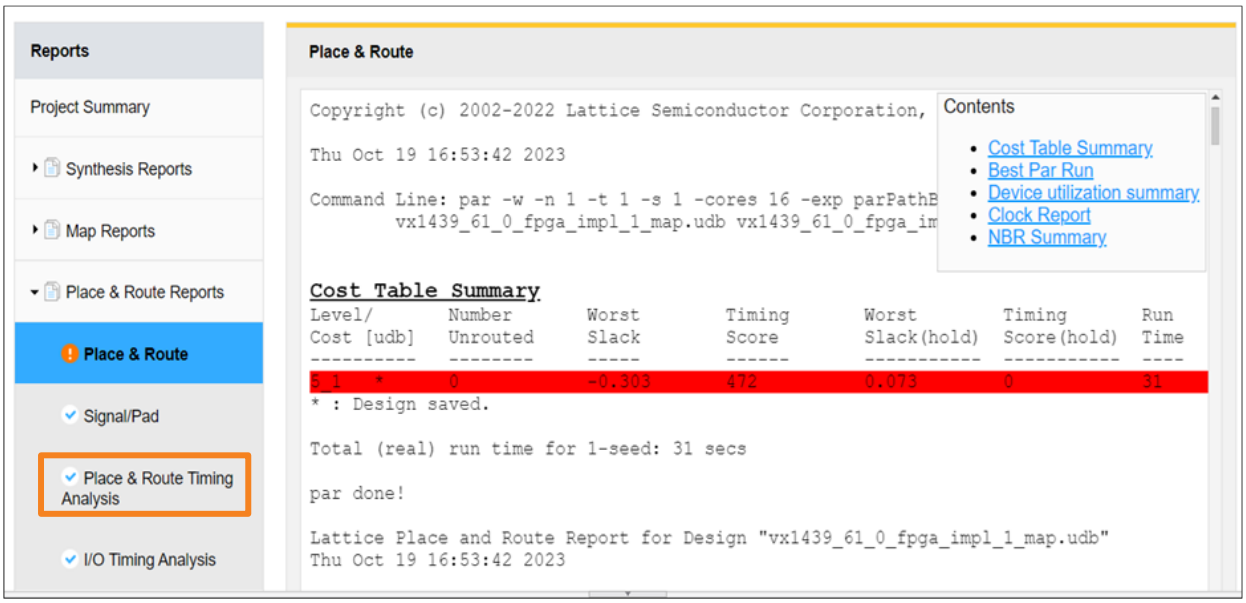


Figure 3.7. Place & Route Report Indicating a Timing Failure

Note that starting in the Radiant software version 2023.2, the timing slacks in the *Cost Table Summary* of the Place & Route report and Place & Route Timing Analysis report in Nexus and Avant devices will not be identical as the timing in former report is just an estimate, as shown in Figure 3.7. You are required to verify the timing in STA for actual sign-off.

For more information, refer to the Timing Analysis in the Radiant Software Version 2023.2 section.

3.4.1.1. Place & Route Timing Analysis Report

The *Place & Route Timing Analysis* report provides extensive details about the timing information of your design. It is broken down into four sections, as shown in Figure 3.8.

```
1  DESIGN CHECKING
        1.1  SDC Constraints
        1.2  Constraint Coverage
        1.3  Overall Summary
        1.4  Unconstrained Report
        1.5  Combinational Loop
    2 Setup at User Specified Speed Grade Corner at User Specified Degrees
        2.1  Clock Summary
        2.2  Endpoint slacks
        2.3  Detailed Report
    3 Setup at User Specified Speed Grade Corner at Minimum Degrees
        3.1  Clock Summary
        3.2  Endpoint slacks
        3.3  Detailed Report
    4 Hold at User Specified Speed Grade Corner at Minimum Degrees
        4.1  Endpoint slacks
        4.2  Detailed Report
```

Figure 3.8. The Four Sections in Timing Analysis Report

Section 1—Design Checking should be where you first look at to determine if your supplied constraints are effective. Remember, a well-constrained design helps the tool achieve a better performance. Table 3.5 lists the description of the categories in Section 1.

Table 3.5. Section 1 (Design Checking) of Timing Analysis Report

Category	Description
SDC Constraints	Lists all the timing constraints that are applied, including user supplied and tool-generated constraints. Ensure all intended constraints are listed.
Constraint Coverage	Displays the total percentage of timing constraints applied in the design. Make sure the coverage is as close to 100% as possible.
Overall Summary	Displays the overall summary of design's timing status, including Setup and Hold timing.
Unconstrained Report	Reports unconstrained paths with start and end point of each path. Suggested timing constraints are provided to constrain the given path. Make sure all paths are constrained to achieve a higher constraint coverage.
Combinational Loop	Lists all combinational or asynchronous loops in the circuit for which timing paths cannot be analyzed. Verify if this is intended, else make changes to eliminate loops.
Error/Warning Messages	Shows the timing constraint violations.

Constraint Coverage and Unconstrained Paths

The *Constraint Coverage* section tells you if your designs are well-constrained, where a 100% constrained design means it is fully constrained. In the event if a design achieves less than 100% coverage, refer to the *Unconstrained Report* section, as shown in Figure 3.9 and Figure 3.10.

Unconstrained Start and End Points

Figure 3.9 shows that the start points are clocked but end points are not constrained. Check the end point to see if any clock or false path constraints are missing.

```
Clocked but unconstrained timing start points
-------------------------------------------------------------------
        Listing 10 Start Points          |           Type
-------------------------------------------------------------------
rx3_sda_pad_cZ_IOL/TOUT                  |          No required time
rx3_scl_pad_cZ_IOL/TOUT                  |          No required time
rx2_sda_pad_cZ_IOL/TOUT                  |          No required time
rx2_scl_pad_cZ_IOL/TOUT                  |          No required time
rx1_sda_pad_cZ_IOL/TOUT                  |          No required time
rx1_scl_pad_cZ_IOL/TOUT                  |          No required time
rx0_sda_pad_cZ_IOL/TOUT                  |          No required time
rx0_scl_pad_cZ_IOL/TOUT                  |          No required time
bd_o_pad[7].bb_inst_IOL/DOUT             |          No required time
bd_o_pad[6].bb_inst_IOL/DOUT             |          No required time
-------------------------------------------------------------------
                                         |
Number of unconstrained timing start po  |
ints                                     |                        36
                                         |
-------------------------------------------------------------------
```

Figure 3.9. Unconstrained Timing Start Points

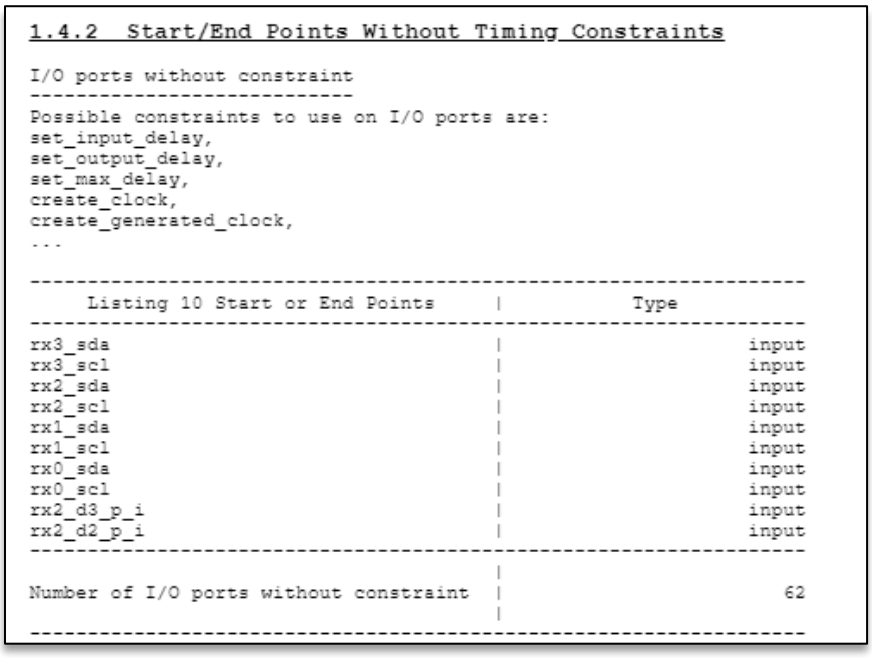
Figure 3.10 shows that the end points are clocked but start points are not constrained. Check the end point to see if any clock or false path constraints are missing.

```
Clocked but unconstrained timing end points
-------------------------------------------------------------------
          Listing 10 End Points          |           Type
-------------------------------------------------------------------
r_cnt3_reg[26].ff_inst/DF                |    No arrival or required
r_cnt3_reg[25].ff_inst/DF                |    No arrival or required
{r_cnt3_reg[25].ff_inst/LSR   r_cnt3_reg[26].ff_inst/LSR}
                                         |          No required time
r_cnt3_reg[24].ff_inst/DF                |    No arrival or required
r_cnt3_reg[23].ff_inst/DF                |    No arrival or required
{r_cnt3_reg[23].ff_inst/LSR   r_cnt3_reg[24].ff_inst/LSR}
                                         |          No required time
r_cnt3_reg[22].ff_inst/DF                |    No arrival or required
r_cnt3_reg[21].ff_inst/DF                |    No arrival or required
{r_cnt3_reg[21].ff_inst/LSR   r_cnt3_reg[22].ff_inst/LSR}
                                         |          No required time
r_cnt3_reg[20].ff_inst/DF                |    No arrival or required
-------------------------------------------------------------------
                                         |
Number of unconstrained timing end poin  |
ts                                       |                      7001
                                         |
-------------------------------------------------------------------
```

Figure 3.10. Unconstrained Timing End Points

Unconstrained I/O Ports

Unconstrained input and output ports. Use set_input_delay for input ports and set_output_delay for output ports.



```
1.4.2  Start/End Points Without Timing Constraints

I/O ports without constraint
----------------------------
Possible constraints to use on I/O ports are:
set_input_delay,
set_output_delay,
set_max_delay,
create_clock,
create_generated_clock,
...

------------------------------------------------------------------
     Listing 10 Start or End Points     |           Type
------------------------------------------------------------------
rx3_sda                                 |                    input
rx3_scl                                 |                    input
rx2_sda                                 |                    input
rx2_scl                                 |                    input
rx1_sda                                 |                    input
rx1_scl                                 |                    input
rx0_sda                                 |                    input
rx0_scl                                 |                    input
rx2_d3_p_i                              |                    input
rx2_d2_p_i                              |                    input
------------------------------------------------------------------
                                        |
Number of I/O ports without constraint  |                      62
                                        |
------------------------------------------------------------------
```

Figure 3.11. Unconstrained Input and Output Ports

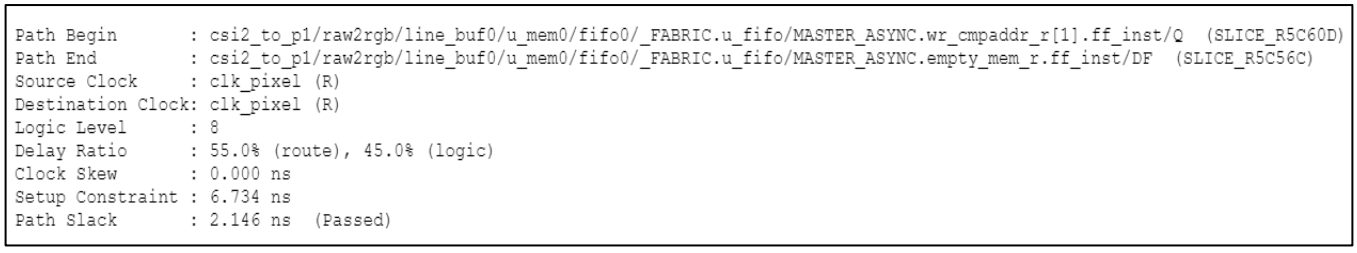
Setup and Hold Timing Reports

Sections 2, 3, and 4 provide the detailed setup and hold slacks of the design. Note that there are two sections for setup timing check and one section for hold timing check. This is because the Radiant software supports multi-corner timing analysis for certain devices (Avant and Nexus devices). Setup is calculated at user speed grade but at two different temperatures (low and high temperature), and hold analysis is performed at 'm' (minimum) speed grade at 0 degree, which represents faster silicon than the fastest performance grade of the device.

All the constrained clocks are shown in the *Clock Summary* section, including the exact timing constraints that are applied (For example, create_clock and create_generated_clock). It also lists out the target Fmax, the actual achievable Fmax that is based on path slacks and Fmax that is limited by the minimum pulse width (MPW). Clock-domain crossing (CDC) transfer paths are also shown in the report.

The *Endpoint slacks* section provides the endpoint slacks that are arranged based on worst slack first. You can quickly look at this report to get firsthand knowledge of the magnitude of timing violations in the design.

To analyze the details of these individual worst paths, refer to the *Detailed Report* section. Its report header shows the source and destination nodes, the clocks for these nodes, number of logic level, clock skew, setup constraint, and path slack as shown in Figure 3.12. This is useful to know if the timing is failing timing due to huge logic level or large clock skew.



```
Path Begin       : csi2_to_p1/raw2rgb/line_buf0/u_mem0/fifo0/_FABRIC.u_fifo/MASTER_ASYNC.wr_cmpaddr_r[1].ff_inst/Q  (SLICE_R5C60D)
Path End         : csi2_to_p1/raw2rgb/line_buf0/u_mem0/fifo0/_FABRIC.u_fifo/MASTER_ASYNC.empty_mem_r.ff_inst/DF  (SLICE_R5C56C)
Source Clock     : clk_pixel (R)
Destination Clock: clk_pixel (R)
Logic Level      : 8
Delay Ratio      : 55.0% (route), 45.0% (logic)
Clock Skew       : 0.000 ns
Setup Constraint : 6.734 ns
Path Slack       : 2.146 ns  (Passed)
```

Figure 3.12. Detailed Report Header

The remaining report provides a breakdown of the source and destination clock delay as well as data path delay. This is where all the delay components are calculated for final timing slack. Figure 3.13 shows a data path delay of one of the timing paths. Here, you can ascertain which sub-components of the path is the bottleneck that contribute to the timing violations. Large routing delay usually indicates far placement and will need special attention to tackle the violations. Same goes to the number of fanouts because a node with huge fanouts might create a 'tension' effect and makes it hard for the tool to achieve an optimal placement.

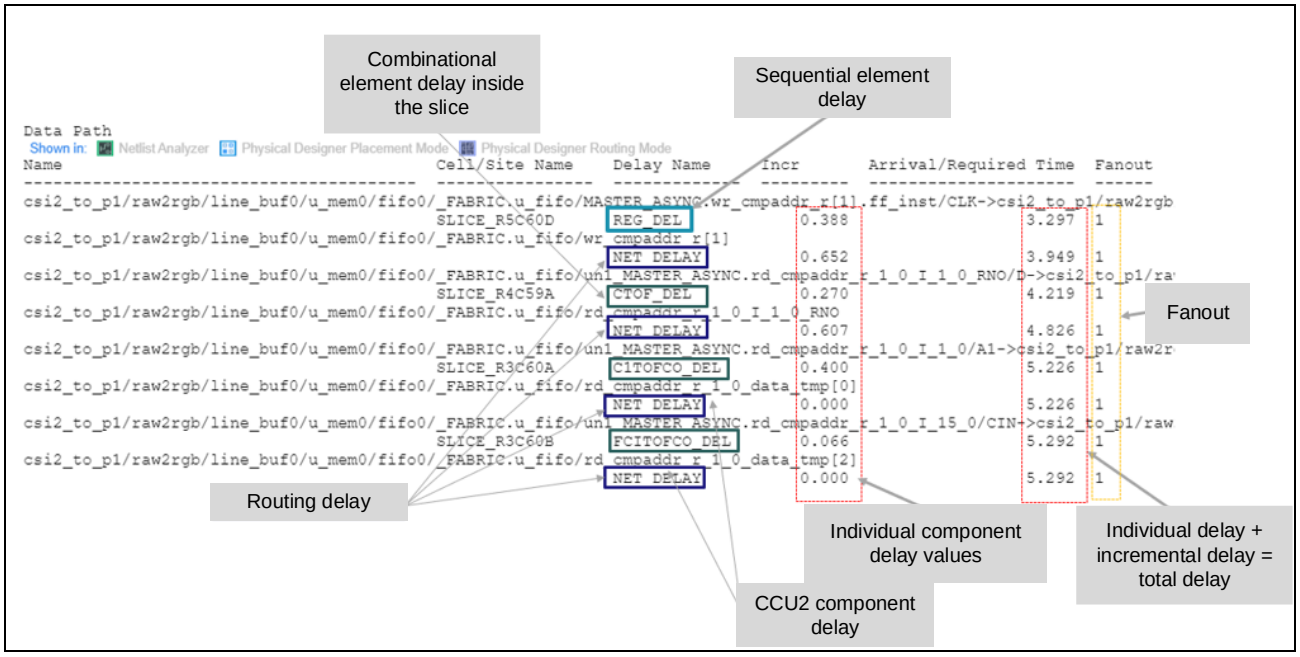


Figure 3.13. Timing Report of a Data Path

3.4.1.2. Timing Analyzer GUI

Besides the *Place & Route Timing Analysis* report, the Timing Analyzer GUI also provides you with the Place-and-Route timing analysis. The results are presented in multiple-spreadsheet tabs with a Query tab for you to easily search through the paths. The information in the Timing Analyzer is very similar to that in the *Place & Route Timing Analysis* report but is presented differently.

You can also customize how timing analysis data are presented in the Timing Analyzer GUI through its setting menu, including the run mode (setup and/or hold), performance grade speed, report format, and number of paths to be reported.

Once run, it has five tabs arranged along the bottom of the view. The Query tab allows you to explicitly search for data paths to be reported out. Select a source clock, destination clock, start point, or end point. More than one items can be selected. Click **Search** to find associated data paths.

3.4.1.3. Standalone Timing Analyzer

You can also run what-if scenarios using the standalone Timing Analyzer tool. Unlike the Timing Analyzer GUI, the Standalone Timing Analyzer can run on designs using post-synthesis, post-map, and post-PAR Unified Design Database (*.udb*). Insert new timing constraints through the **Edit → Timing Constraint Editor** to apply new constraints and observe the new updated timing slacks while maintaining the desired compiled netlist. Alternatively, you can load a PDC file through **File → Load Timing Constraint File**.

In addition, you can evaluate what the slacks would be under different operating conditions through the **Edit → Operating Condition Setting** menu, where the Junction Temperature and Voltage can be modified.

3.5. Cross-Probing Capabilities in the Radiant Software

In certain cases, looking at the timing report alone might not be sufficient to identify timing performance bottleneck of a timing path. The Radiant software provides cross-probing capabilities that lets you examine any timing paths in the Netlist Analyzer, Physical Designer (Placement Mode), or Physical Designer (Routing Mode), as shown in Figure 3.14 and Figure 3.15.

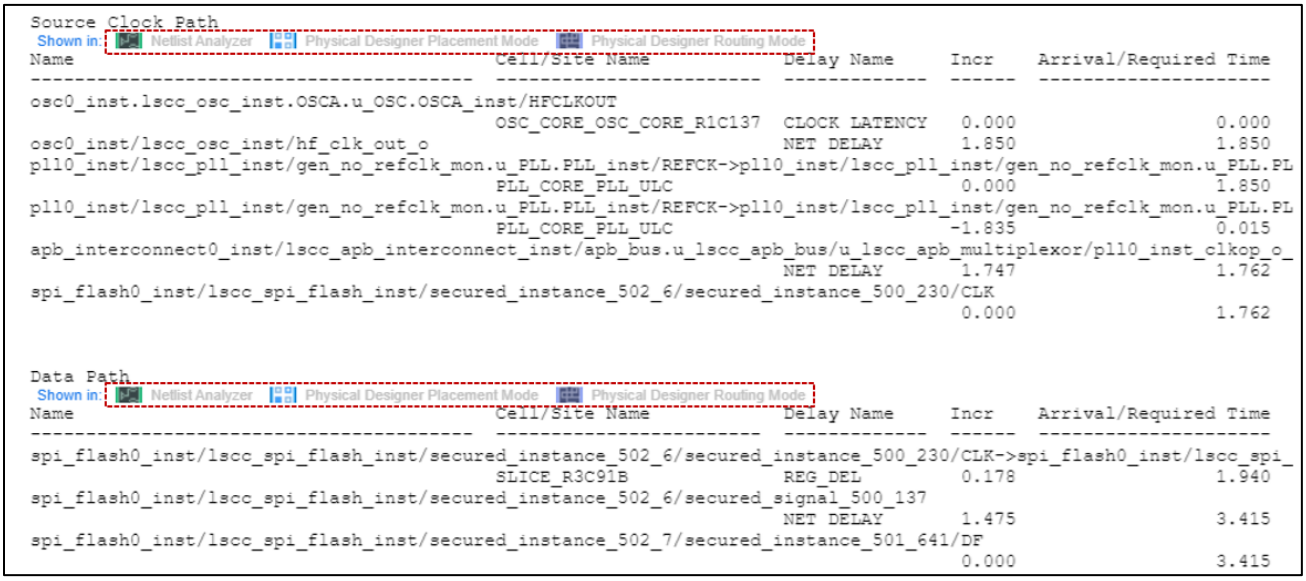


Figure 3.14. Cross-Probing Capabilities from the Place & Route Timing Analysis Report

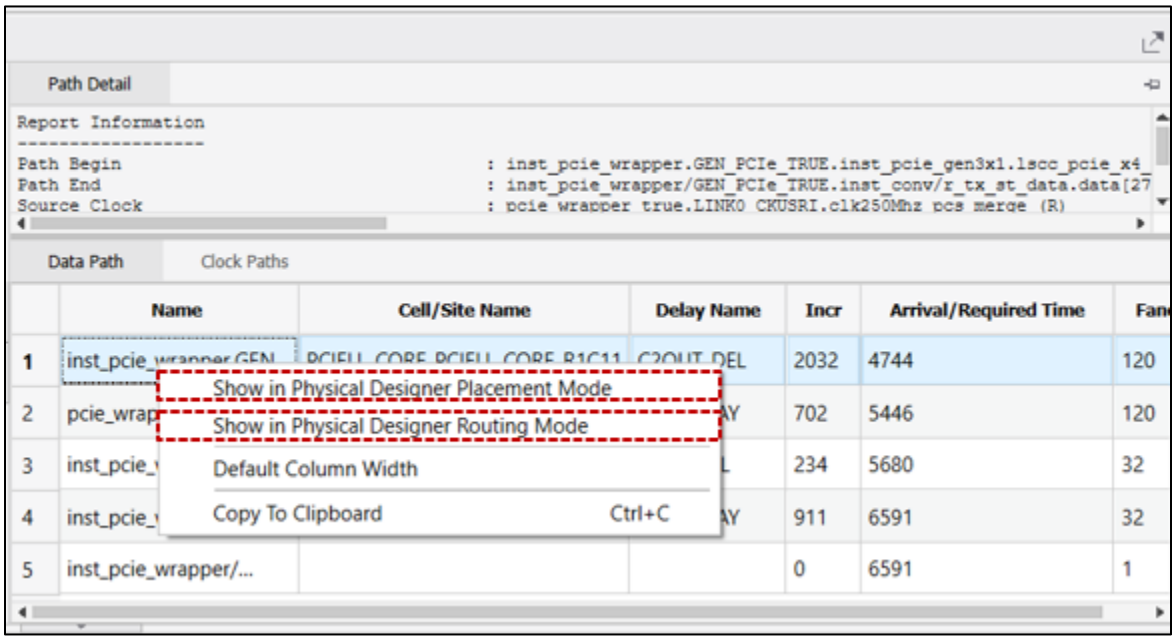


Figure 3.15. Cross-Probing Capabilities from the Timing Analyzer GUI

Some examples of how the cross-probing capability is useful in identifying timing limitations in a design would be provided in subsequent chapters in this document.

4. Optimizing Timing Performance for Mid-Size FPGAs

This section focuses on what to do if there are timing violations in your design.

4.1. Analyzing Timing Result

The first step in dealing with timing violations is understanding and correlating the provided timing constraints to the final timing results. Knowing how your constraints are being reflected when analyzing a timing path is crucial as it will influence the next course of action in rectifying the timing violations.

In the simplest form, static timing analysis is formed based on the constrained clocks, which are created and applied to the design. These clocks typically have interactions within their domains and with other clock domains. These forms the 'Setup Constraint' and 'Hold Constraint' in the Radiant software, which will be used by the tool to meet the requirements and as final timing sign-off.

Figure 4.1 shows a simplified data transfer from source register A to destination register B. If both registers are clocked by the same clock source, it is known as common clock transfer but if they are from different clock sources, this is known as a clock-domain crossing (CDC) transfer.

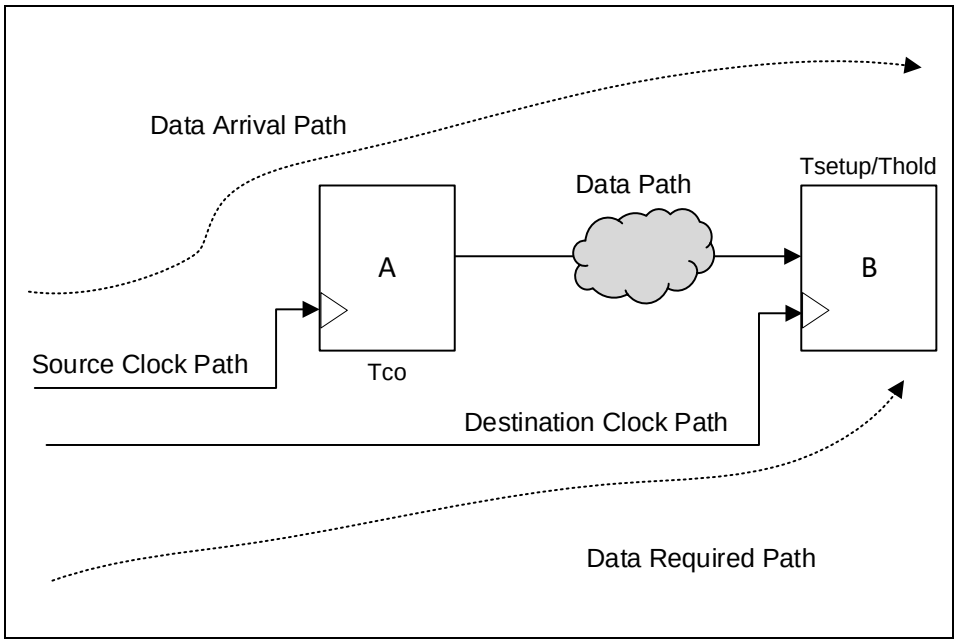


Figure 4.1. A Simple Register-to-Register Transfer

From Figure 4.1, the following equations can be deduced:

- Data Arrival Path = Source Clock Edge + Source Clock Path + Source Register Tco + Data Path + (Tsetup or Thold)
- Data Required Path = Destination Clock Edge + Destination Clock Path

Setup Slack Calculation = Data Required Path – Data Arrival Path

= (Destination Clock Edge – Source Clock Edge) + (Destination Clock Path – Source Clock Path) - Source Register Tco – Data Path - Tsetup

= **For non-common clock transfer**

(Destination Clock Edge – Source Clock Edge) - Clock Skew – Data Path – Source Register Tco – Tsetup

= **For common clock transfer**

Clock Period* - Clock Skew – Data Path – Source Register Tco – Tsetup

Hold Slack Calculation = Data Arrival Path – Data Required Path

= (Source Clock Edge – Destination Clock Edge) + (Source Clock Path - Destination Clock Path) + Source Register Tco + Data Path + Thold

= **For common clock transfer**

(-Clock Skew) + Source Register Tco + Data Path + Thold

= **For non-common clock transfer**

(Source Clock Edge – Destination Clock Edge) + (-Clock Skew) + Source Register Tco + Data Path + Thold

*where Clock Period = Destination Clock Edge - Source Clock Edge and Clock Skew = Destination Clock Path - Source Clock Path.

For common-clock hold slack calculation, the difference between Destination Clock Edge and Source Clock Edge is 0.

From the equations, setup and/or hold slack is influenced by multiple elements. Because Tsetup/Thold cannot be controlled by you, the final slack could be altered by influencing the values of data path delay, source clock path delay, destination clock path delay, or any combinations of the above.

The Timing Analyzer GUI provides the delay elements of a timing path in the Path Detail information window, as shown in Figure 4.2 for Lattice Standard and Figure 4.3 for Diamond Style report format.

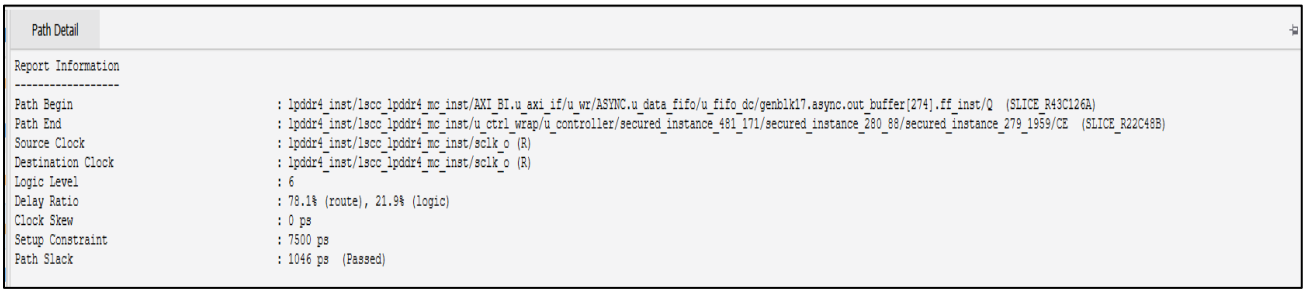


Figure 4.2. Timing Analyzer Path Detail Window (Lattice Standard)

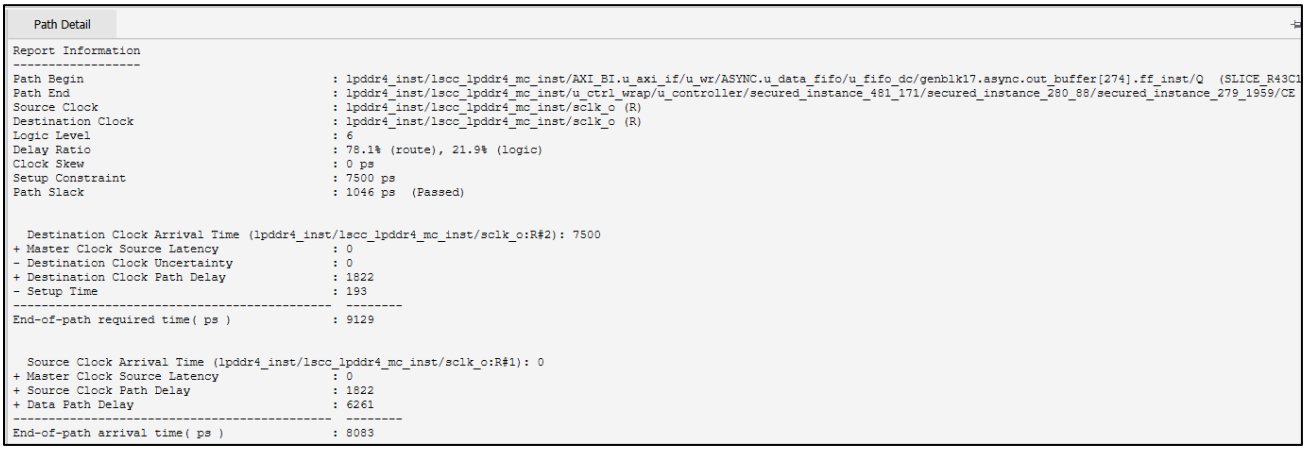


Figure 4.3. Timing Analyzer Path Detail Window (Diamond Style)

4.2. Tackling Timing Violations

When a design fails to meet its timing requirement, the first question that you would typically ask is whether the software is doing good enough or the way the RTL code is written is efficient enough to take advantage of the device. Let us begin by looking at some of the actions that require less effort.

4.2.1. Review Timing Constraints and Warning Messages

Ensure the design achieves high timing constraint coverage, ideally 100%. In many cases, timing violations are due to improper or dropped constraint resulting in false alarms. Check if clock-crossing paths are properly constrained through timing exceptions such as set_clock_groups or set_max_delay. By eliminating these paths, the Radiant software can focus on the real critical paths in subsequent compilations.

Also, look for critical or warning messages in any of the compilation reports. For example, a heavily congested design or when trying to meet hold time requirement, the Radiant software will produce a critical warning message indicating hold timing correction is disabled to reduce runtime. It is also useful to check if any of the sub-modules or resources have been optimized away at this stage.

4.2.2. Long Logic Level

As discussed in the Analyzing Timing Result section, setup slack is hugely influenced by data path delay, considering it is a common-clock transfer where clock skew is negligible. A large data path delay could be contributed by logic cell delay (sequential or combinational logic delay) or net delay (routing delay), or a combination of both. Since each logic has its own delay, a transfer path with large logic level will have an accumulation of cell delay and the net delay required to connect these cells will also increase accordingly. This can be confirmed by examining the 'Logic Level' count in the Detailed Report for timing paths as shown in Figure 4.4. The delay ratio which is the ratio between routing interconnect and logic cell is also listed. In this example, we can see close to 62% of the total delay is contributed from the logic delay itself.

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                    Detailed Report for timing paths
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

++++Path 1  ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

Path Begin        : top_module/sub_module/child_module/source_node
Path End          : top_module/sub_module/child_module/destination_node
Source Clock      : pclk (R)
Destination Clock: pclk (R)
Logic Level       : 7
Delay Ratio       : 38.1% (route), 61.9% (logic)
Clock Skew        : -0.084 ns
Setup Constraint  : 6.399 ns
Path Slack        : -0.682 ns  (Failed)
```

Figure 4.4. Path Detail Report Showing the Logic Level Count and Delay Ratio

To better illustrate the data path, you can cross-probe it to the Netlist Analyzer, as shown in Figure 4.5. The white rectangular boxes represent sequential logic (for example, source nodes and destination nodes) while the gray rectangular boxes represent combinational logic.

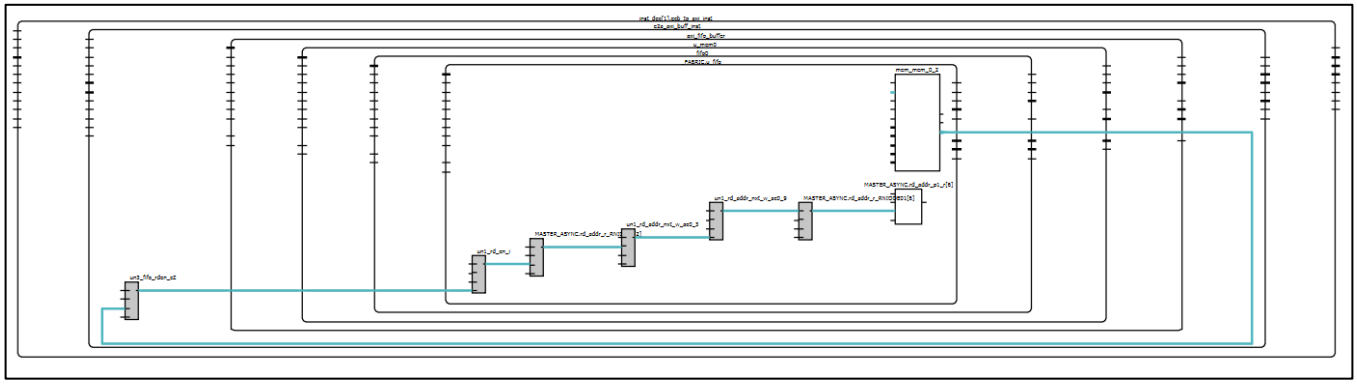


Figure 4.5. A Critical Path Cross-Probed to the Netlist Analyzer

In this scenario, consider adding pipeline registers to break up the long logic level between the source and destination nodes. Adding registers will result in additional latency in the design so it is easier if you plan ahead to accommodate for this. Changing latency typically requires modifying associated control logic or other parts of the system to work in tandem with the pipelined data path. These additional latencies may also affect simulation testbenches.

4.2.3. Sub-optimal Logic Placement

There are also cases where a data path fails setup requirement despite having low logic level between the source and destination nodes. In such a scenario, it could be due to far placement between the nodes and this results in long routing delay. Figure 4.6 is an example of such path where it fails timing despite having only one logic level. To confirm this, the path is cross-probed to as shown in the Physical Designer (Figure 4.7).

As a solution, you can use floorplanning method to direct the tool to place the logic nodes at the desired location. This method can be applied to solve both setup and hold time violations as discussed in the next sub-sections.

```
Path Begin        :  top/source
Path End          :  top/destination
Source Clock      :  clk (R)
Destination Clock:  clk (R)
Logic Level       :  1
Delay Ratio       :  26.8% (route), 73.2% (logic)
Clock Skew        :  -0.093 ns
Setup Constraint  :  4.000 ns
Path Slack        :  -0.474 ns  (Failed)
```

Figure 4.6. A Path Failing Setup Requirement Despite Low Logic Level

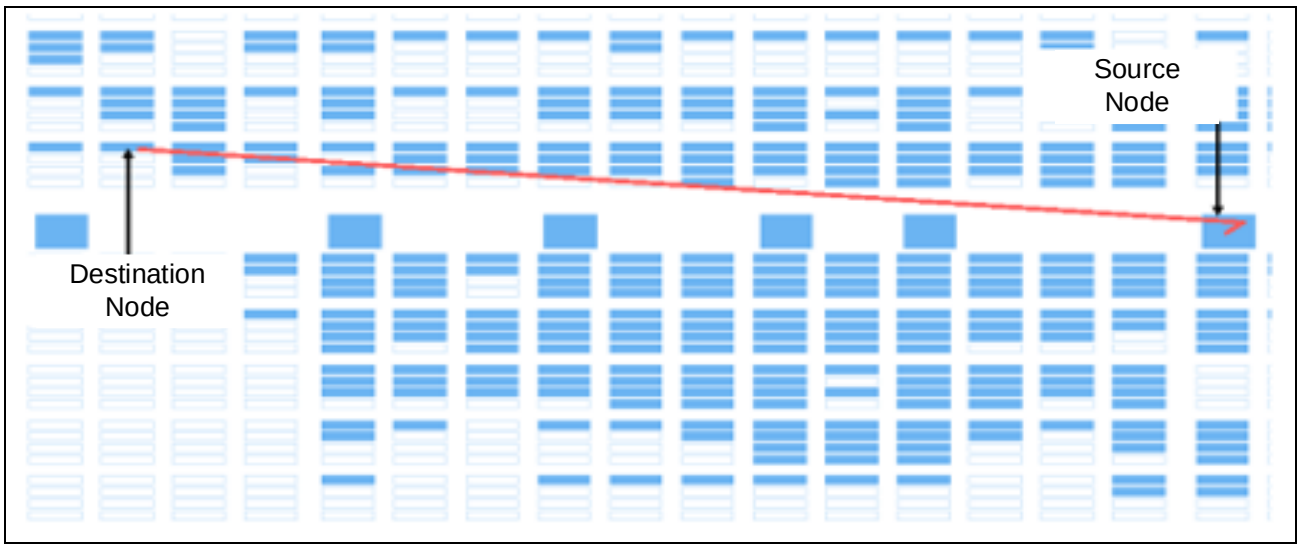


Figure 4.7. Physical Designer Showing Far Placement

4.2.3.1. Using Floor-Planning to Resolve Setup Violations

From Figure 4.7, it is apparent that the timing violation is from the output of an EBR block (source node) to a core register (destination node). The far placement is causing the long routing delay. To overcome this problem, use the floor-planning method in the Radiant software via “Bounding Boxes” (Bboxes), which instructs the tool to place the nodes of interest in your desired location.

Figure 4.8 shows you how to specify the locations of the EBR blocks and the core register using Bboxes. Compared to the data path in Figure 4.7, the data path is now much shorter and it can now meet its timing requirement.

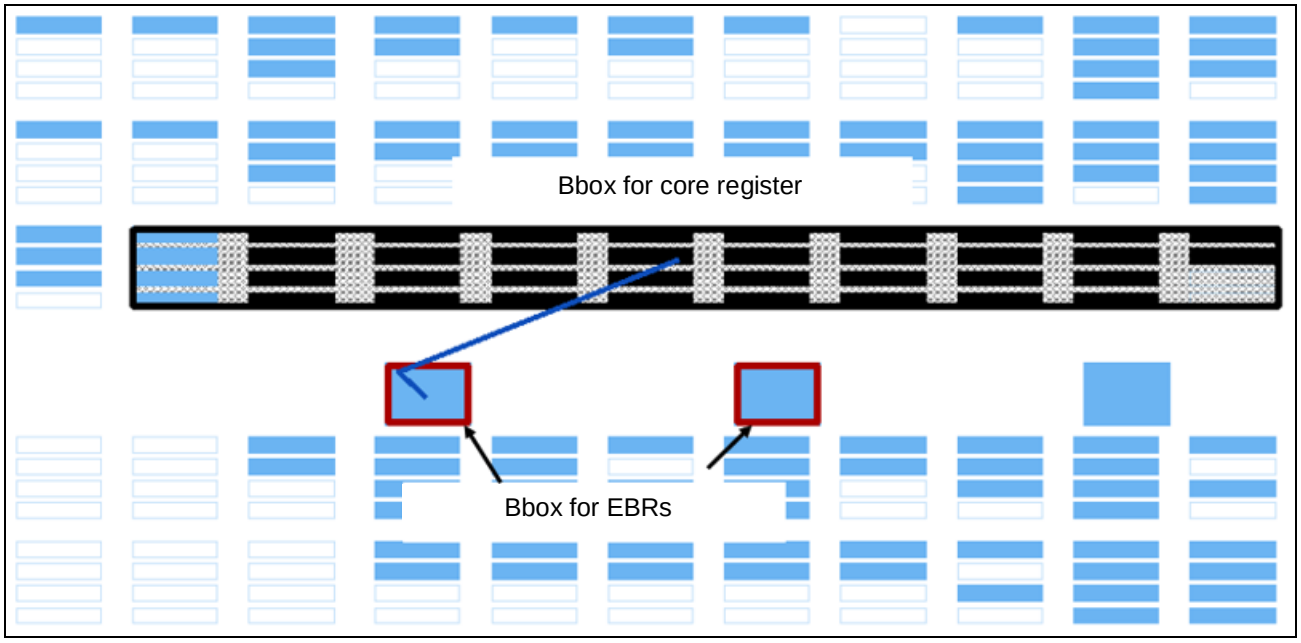


Figure 4.8. Using Bboxes to Physically Constrain Nodes for Setup Violations

4.2.3.2. Using Floor-Planning to Resolve Hold Violations

Unlike setup violations, hold time requirements are usually resolved by the tool either through placing the source and destination nodes far apart or explicitly adding more routing interconnect or a combination of both. Hold violations are less typical in common clock transfers because the clock skew, which is the difference between the source clock and destination clock delay is usually very small.

In the Radiant software's Place & Route Design settings, you can selectively enable **Prioritize Hold Correction Over Setup Performance** to address hold violations but at the expense of setup time. However, this might not be sufficient in situations where hold violations are due to huge clock skew and floor-planning using Bounding Boxes (Bboxes) could be used to resolve such problems. Figure 4.9 shows a path failing hold time requirement with a slack of -2.815 ns, where the clock skew is close to 3 ns.

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                    Detailed Report for timing paths
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

++++Path 1  ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

Path Begin        :  source node direct
Path End          :  destination node clock divider
Source Clock      :  direct_pll_out (R)
Destination Clock:  pll_thru_clkdiv (R)
Logic Level       : 2
Delay Ratio       : 27.9% (route), 72.1% (logic)
Clock Skew        : 2.988 ns
Hold Constraint   : 0.000 ns
Common Path Skew  : -0.046 ns
Path Slack        : -2.815 ns   (Failed)
```

Figure 4.9. Timing Report for a Hold Violated Path due to Huge Clock Skew

Both the source and destination clocks come from the same source, which is an output of a PLL but the latter must go through an additional clock divider resulting in additional delay, as shown in Figure 4.10.

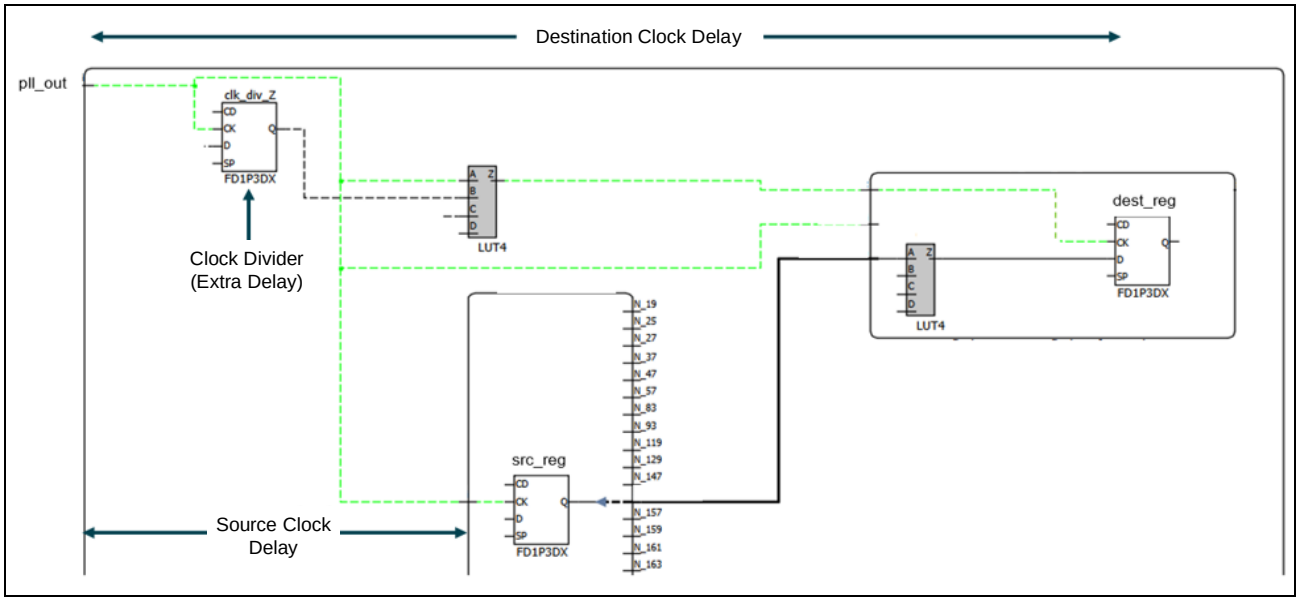


Figure 4.10. Netlist Analyzer Showing How the Destination Clock Goes Through Clock Divider

Furthermore, the clock divider must be placed close to the fixed clock resources on the chip, contributing to extra clock routing delay. To compensate for the huge clock skew, Bboxes can be used to ensure source and destination nodes are placed far from each other, as shown in Figure 4.11.

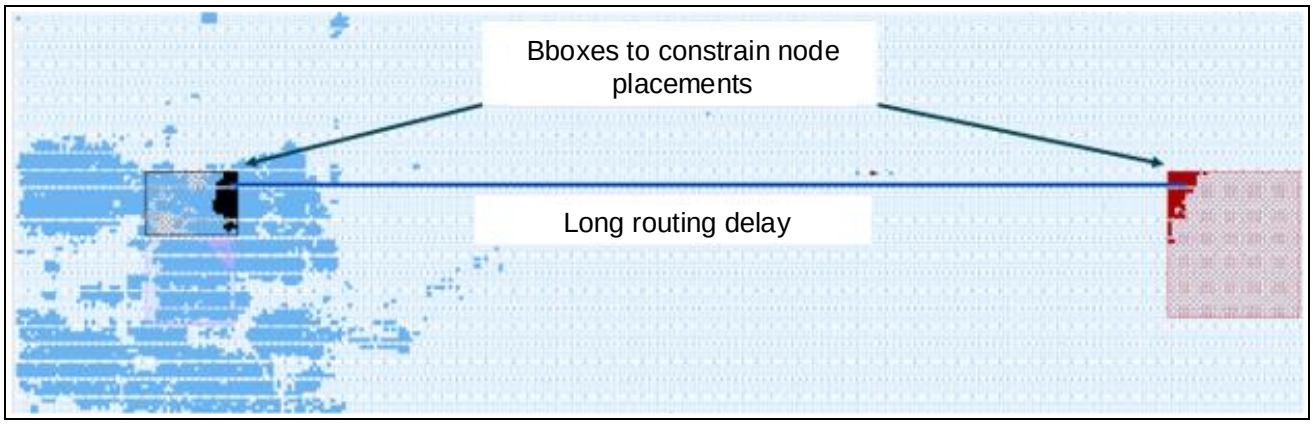


Figure 4.11. Using Bboxes to Place Source and Destination Nodes Far Apart

To know more about creating bounding boxes, refer to the *Lattice Radiant Software Help* and search for the section under *Applying Design Constraints -> Applying Radiant Software Physical Constraints -> Setting Port, Net, and Cell Attributes -> Creating GROUPs*.

Figure 4.12 shows the final result. Despite the huge clock skew, the path can now meet its hold requirement as the routing delay is long enough to compensate for that. Note how the delay ratio is dominated by the routing delay.

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                    Detailed Report for timing paths
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

++++Path 1  ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

Path Begin        : source node direct
Path End          : destination node clock divider
Source Clock      : direct_pll_out (R)
Destination Clock: pll_thru_clkdiv (R)
Logic Level       : 2
Delay Ratio       : 95.6% (route), 4.4% (logic)
Clock Skew        : 3.396 ns
Hold Constraint   : 0.000 ns
Common Path Skew : -0.046 ns
Path Slack        : 0.018 ns  (Passed)
```

Figure 4.12. Timing Report After Using Bboxes to Resolve Hold Time Violations

4.2.4. Understanding the Hardware

You must be familiar with the device selected, including its resource types and locations. If the architecture and the characteristics are not well understood, its full performance capability would not be fully utilized. Be aware of the locations of hard blocks such as EBRs, DSPs, and other hard IP cores where the locations are less flexible due to their fixed locations.

4.2.4.1. Memory Modules

Make full use of the available registers in the memory modules for maximum performance. Refer to the user guide of the selected device to understand its architecture. For example, the input registers in EBR blocks in Avant devices are always registered but the output data of the memory is optionally registered. If possible, always enable the output registers of the EBR blocks. Otherwise, you will have a very large value of C2OUT_DEL, which is detrimental to the overall timing performance.

Also, for memory sizes larger than that of a single module, the multiple modules are cascaded (either in depth or width) to create a larger module. This will also affect the timing performance of the EBR blocks and subsequently the overall design. Table 4.1 summarizes the effect of different module parameters on the C2OUT_DEL on Avant devices.

As seen, enabling the output registers of the EBR blocks can effectively reduce the C2OUT_DEL delay by around 1 ns.

Table 4.1. RAM Modes and Resulting Implementations in Avant Devices

RAM Mode	Registered Output	Number of RAMs	C2OUT_DEL
4,096 x 9	Yes	1	335 ps
4,096 x 10	Yes	2	335 ps
4,096 x 9	No	1	1,333 ps
4,096 x 10	No	2	1,333 ps
8,192 x 9	Yes	2 (Cascaded)	335 ps
8,192 x 9	No	2 (Cascaded)	1,333 ps

4.2.4.2. DSP Blocks

Similar to memory modules, make full use of the internal registers of the DSP blocks for optimal performance. For example, the DSP blocks in Avant devices can be configured to implement different modes—from 8 × 8 multiplier to 18 × 18 multiplier where input, output, and pipeline registers in the DSP block can be optionally bypassed, as shown in Figure 4.13.

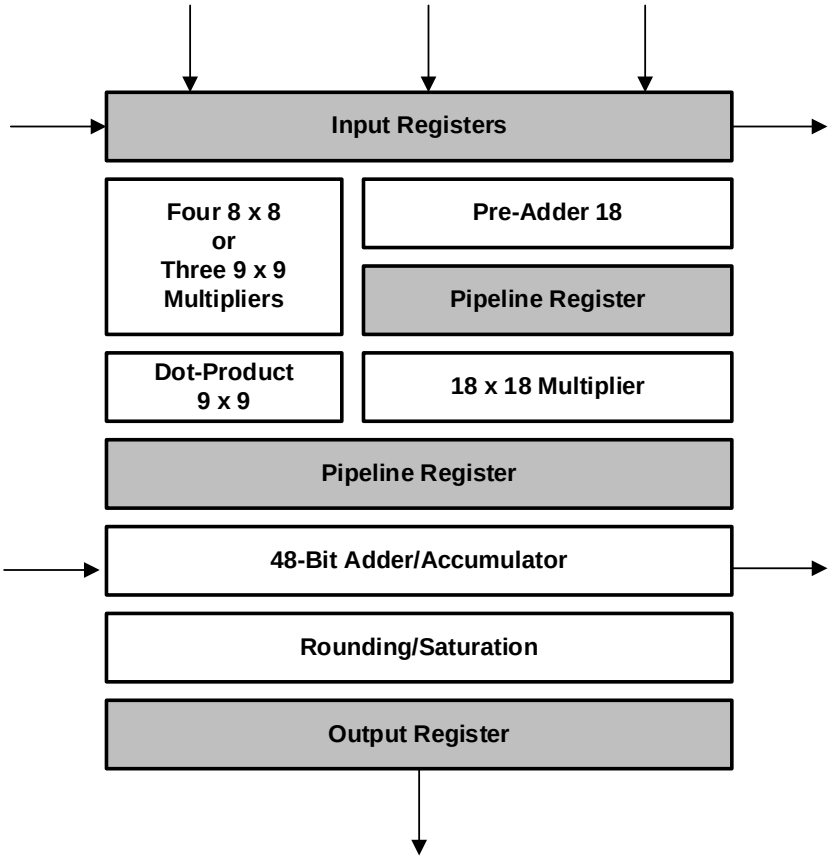


Figure 4.13. Overview of DSP Block in Avant Devices

Depending on how the DSP blocks are configured, the final implementation will have different impacts on the timing performance, as shown in Table 4.2.

Note that even a slight difference such as number of bits and signed or unsigned integer may result in difference in timing performance.

Table 4.2. DSP Modes and Resulting Implementations in Avant Devices

DSP Mode	Signed/Unsigned	Registered Output	Number of DSPs	C2OUT_DEL
18x18	Signed	Yes	1	210 ps
18x18	Signed	No	1	1,457 ps
19x19	Signed	Yes	4	2,669 ps
19x19	Signed	No	4	2,669 ps
18x18	Unsigned	Yes	3	2,669 ps
18x18	Unsigned	No	3	2,669 ps

4.2.5. Fanout Control

When a net has high fanout, it inherently becomes more challenging for the tool to find a routing with a low delay for all destinations of the net, particularly when the destinations are scattered at various locations on the chip. This causes a 'stretching effect' on the net and any destination nodes that are placed further away would easily fail timing. The more destinations there are, the harder it becomes to optimize the timing so that all nets have a low delay.

Figure 4.14 shows the timing report of a path that fails timing because the source net has destination nodes scattered all around the chip. While a fanout value of 33 does not seem large, the 'stretching effect' mentioned earlier is enough to cause a negative impact on its timing performance, as shown in Figure 4.15.

Data Path | Clock Paths

	Name	Cell/Site Name	Delay Name	Incr	Arrival/Required Time	Fanout
1	inst_mi/...	SLICE_R33C29D	REG_DEL	514	4189	33
2	inst_mi/...		NET DELAY	693	4882	33
3	inst_mi/...	SLICE_R32C35A	CTOF_DEL	355	5237	1

Figure 4.14. Timing Path of a Source Node with Multiple Fanouts

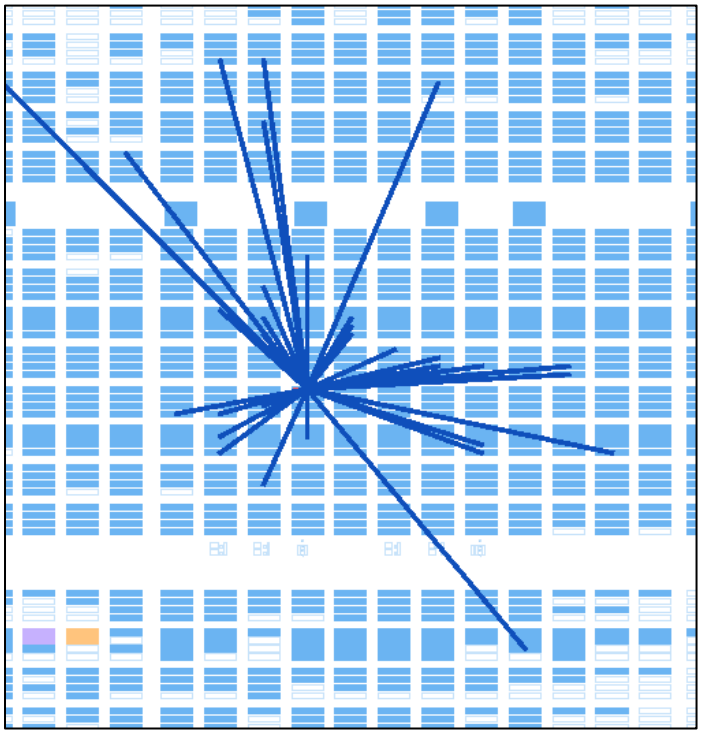


Figure 4.15. Destination Nodes Scattered in Different Directions

There are two solutions to tackle timing closure problems in high fanout nodes, which is discussed in the following sections.

4.2.5.1. Limiting Maximum Fanout in the Radiant Software Setting

You can set a maximum fanout assignment in the Radiant software through the **Synthesis Strategy** settings:

- LSE: Max Fanout Limit (Default value is 1,000)
- Synplify Pro: Fanout Guide (Default value is 1,000)

In this method, the Radiant software duplicates the source register when it has more than 1,000 destination nodes. For example, a register with 3,500 fanouts will be duplicated three times, resulting in a total of four registers with each of them having 1,000 fanouts except for the last one which has 500.

However, this method has a disadvantage because the tool might not have perfect knowledge in the fanout distribution of each of these duplicates. Besides reducing the number of fanouts, it is even more critical to ensure that each duplicate is physically distributed close to its destinations. Otherwise, the 'stretching effect' will still exist due to physical distance. To solve this problem more efficiently, perform the following method—Manual Duplication in RTL.

4.2.5.2. Manual Duplication in RTL

This method requires more effort but has significant advantage over the maximum fanout assignment method because you have precise control on how each duplicate should be connected. Not only can you control what registers get duplicated, but also exactly what destination each duplicate fanouts to. This is effective when a source register has distinct fanouts that are placed in different locations and this method allows the duplicates to be placed near them.

Note that typically when the tool recognizes more than one registers performing the same logical function, it will automatically merge them into one register, which defeats the purpose of manually duplicating the source register. In other words, if a register is manually replicated in the RTL code, the tool will replace all the duplicates with one register. To avoid this merging operation, it is necessary to disable this feature explicitly. A common way to accomplish this is through synthesis attributes such as syn_preserve.

Figure 4.16 summarizes the methods discussed to solve high fanout problems.

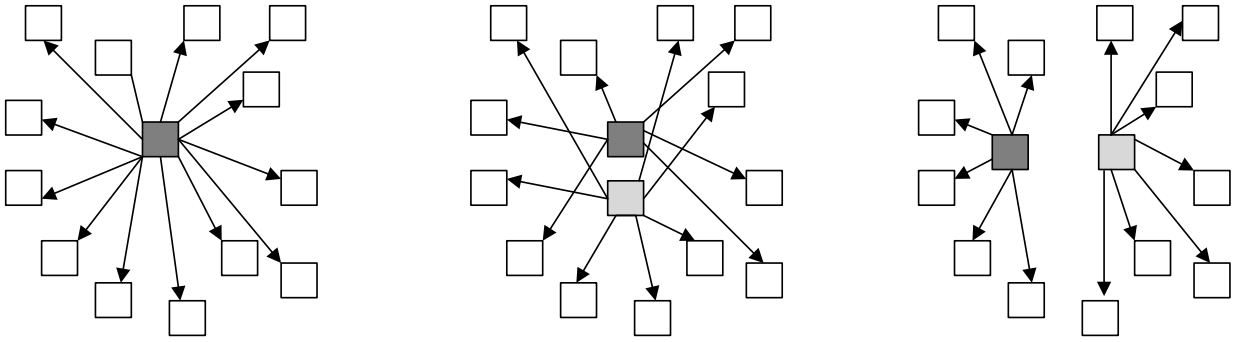


Figure 4.16. How Max Fanout Assignment and Manual Duplication Help in High Fanout Situation

4.3. Other Timing Closure Technique Tips

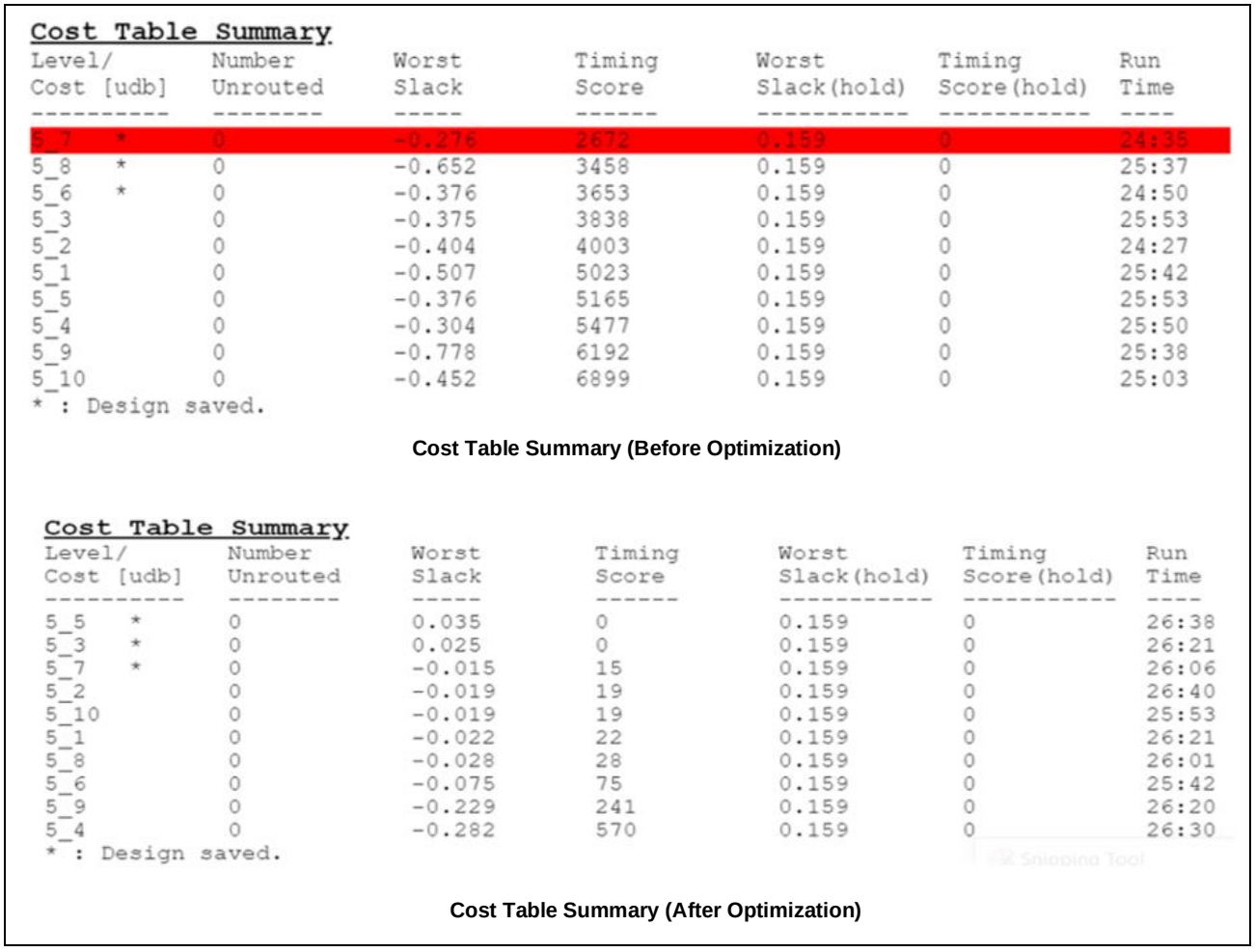
This section describes some of the best-known methods and techniques that can be used during the process of timing closure using the Lattice Radiant software.

4.3.1. Seed Iterations

As discussed in the Timing-Driven MAP and PAR section, the **Placement Iteration Start Point** and **Placement Iterations** settings allow you to influence the placement initial point. This means that different iteration start points (i.e seeds) will yield different final timing results for a given set of tool settings, input constraints, and RTL source files.

Any design optimization, either through RTL edits or tool setting tweaks can bring changes to timing performance, either positively, negatively, or stay the same. As such, running multiple iterations provides you with a method to assess timing performance improvement by giving an overview on what impact your design changes have on the final outcome.

Figure 4.17 shows two cost table summaries. The top cost table shows the original timing performance before optimization and the bottom cost table shows the timing performance after optimization. In this scenario, a total of 10 placement iterations are run and the results are arranged based on *Timing Score*.



Cost Table Summary

Level/ Cost [udb]	Number Unrouted	Worst Slack	Timing Score	Worst Slack(hold)	Timing Score(hold)	Run Time
5_7 *	0	-0.276	2672	0.159	0	24:35
5_8 *	0	-0.652	3458	0.159	0	25:37
5_6 *	0	-0.376	3653	0.159	0	24:50
5_3	0	-0.375	3838	0.159	0	25:53
5_2	0	-0.404	4003	0.159	0	24:27
5_1	0	-0.507	5023	0.159	0	25:42
5_5	0	-0.376	5165	0.159	0	25:53
5_4	0	-0.304	5477	0.159	0	25:50
5_9	0	-0.778	6192	0.159	0	25:38
5_10	0	-0.452	6899	0.159	0	25:03

* : Design saved.

Cost Table Summary (Before Optimization)

Cost Table Summary

Level/ Cost [udb]	Number Unrouted	Worst Slack	Timing Score	Worst Slack(hold)	Timing Score(hold)	Run Time
5_5 *	0	0.035	0	0.159	0	26:38
5_3 *	0	0.025	0	0.159	0	26:21
5_7 *	0	-0.015	15	0.159	0	26:06
5_2	0	-0.019	19	0.159	0	26:40
5_10	0	-0.019	19	0.159	0	25:53
5_1	0	-0.022	22	0.159	0	26:21
5_8	0	-0.028	28	0.159	0	26:01
5_6	0	-0.075	75	0.159	0	25:42
5_9	0	-0.229	241	0.159	0	26:20
5_4	0	-0.282	570	0.159	0	26:30

* : Design saved.

Cost Table Summary (After Optimization)

Figure 4.17. Timing Results Before and After Optimization Based on a 10-Iteration Runs

The results show that even with the exact same RTL code and tool settings, there are variances when multiple iterations are run. Therefore, an average or geo-metric value of all the seeds must be calculated so that any timing improvement or

degradation will not be masked by pure seed noise. Taking the example in Figure 4.17, the overall timing result after optimization has improved but if you only run one seed for each version; seed 7 before optimization (worst slack = -0.276 ns) against seed 4 after optimization (worst slack = -0.282 ns), then the judgement will be misguided.

However, running multiple placement (seed) iterations should not be considered as a systematic method to close timing as you have no control on how the place and route takes place in your design. Instead, you must put more effort in other methods, particularly during the early cycle in the design phase. Seed-sweeping should be one of the last resorts for meeting timing when the negative slacks are small and when other methods have been exhausted.

4.3.2. Changing the Number of Reported Paths in the Timing Analyzer

In the Lattice Radiant software, the default number of reported paths in the Timing Analyzer are as follow:

- Critical endpoints path number limit: 10
- Number of paths per constraint: 10
- End point number limit: 1

While this might be sufficient for small designs without serious timing violations, it is recommended to increase the number of reported paths to a much larger number especially when the design has many violated paths. This method allows you to see a bigger picture of the violated modules and analyze the critical paths easily.

5. What's New in the Lattice Radiant Software 2023.2?

5.1. Timing Model and Multi-Corner Timing Analysis

Timing models are essential in static timing analysis, which involves analyzing the circuit's timing paths and verifying that the setup and hold time requirements are met. They define the relationship between the input and output delays of a component and factors such as gate delays, interconnect delays, and other circuit parameters. Hence, the accuracy of the timing models directly affects the accuracy of the analysis and the ability to identify and fix any timing violations in the design.

Timing corners refer to a specific set of process, voltage, and temperature (PVT) conditions used to analyze and characterize the timing behavior of a digital circuit. They are important to account for the variations that can occur in the performance of a circuit due to different operating conditions.

The timing corners include:

- **Process Corner**
  This represents variations in the fabrication process such as manufacturing tolerances and process variations factors like doping levels and oxide thickness that can impact the performance of the circuit.

- **Voltage Corner**:
  This represents variations in the supply voltage level. Analyzing the circuit's timing behavior at different voltage levels allows you to determine the impact of voltage scaling on the circuit's performance.

- **Temperature Corner**:
  This represents variations in operating temperature. Temperature can affect the performance of the circuit by influencing the speed of transistors and the behavior of interconnects. Timing analysis at different temperature corners helps ensure that the circuit meets timing requirements under varying thermal conditions.

Figure 5.1 shows how the delay values could be affected by changes in the PVT variations. Note that timing must be met in all timing corners during the sign-off stage to ensure functionality robustness.

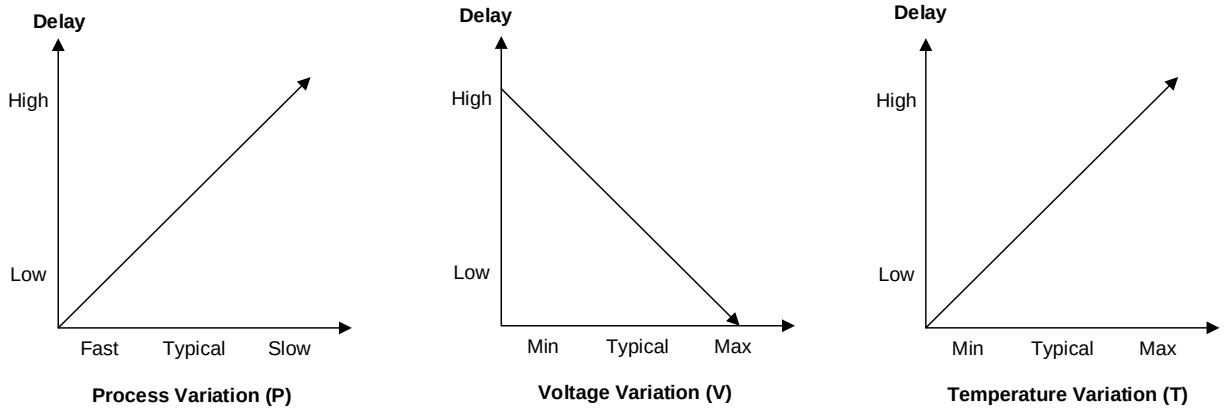


Figure 5.1. Effect of PVT Variations on Delay Value

From Figure 5.1, the delay through a digital circuit increases with temperature, leading to slower performance. This is the typical temperature behavior. However, in some specific scenarios such as different types of circuit technologies or technology nodes, temperature inversion can occur. This is where the delay through the circuit decreases as the temperature increases.

5.1.1. Timing Analysis in the Radiant Software Version 2023.2

In the Radiant software version 2023.1 and prior, multi-corner timing analysis is a manual process for Nexus platform. To cater for timing analysis across multiple timing corners, the Radiant software version 2023.2 automates the reporting of best-case and worst-case Static Timing Analysis for the Nexus platform. Note that multi-corner timing analysis has always been available for Avant devices by default.

This means that starting in the Radiant software version 2023.2, timing analysis for Nexus devices is switched to use a more accurate timing engine, similar to that of Avant devices. This has impact on the timing report of both PAR and STA signoff as summarized in Table 5.1.

Table 5.1. Timing Report Changes in the Radiant Software Version 2023.2 and Beyond

	Radiant Software Version 2023.1 and Prior	Radiant Software Version 2023.2 and Beyond
PAR Timing Report	Timing at default corner is used in timing-driven place & route. Timing reported by PAR typically matches with the STA report at default corner.	Timing model considering worst-case at various corners is used in timing-driven place & route. Timing reported by PAR is an estimate, which is typically worse than the actual timing. You are required to run STA to get the actual timing.
STA Timing Report	Setup time analysis is done at high temperature only.	Setup time analysis is done at both low temperature and high temperature.

5.2. Clock Skew

Clock skew refers to the difference between the clock delay to the launching register and the clock delay to the latching register. A positive clock skew happens when the clock delay to the latching register is larger than the clock delay to the launching register and is beneficial for setup time. On the contrary, a negative clock skew means that the clock delay to the launching register is smaller than the clock delay to the latching register and is beneficial for hold time.

5.2.1. Inclusion of Clock Skew in the Radiant Software Version 2023.2

In the Radiant software version 2023.1 and prior, zero clock skew method is applied for simplicity. Starting in the Radiant software version 2023.2, clock tree skew is considered during timing analysis for Nexus devices, and accurate timing calculation is adopted as well, as shown in Figure 5.3 and Figure 5.5. Note that the inclusion of clock skew during timing analysis has always been available for Avant devices by default.

5.2.1.1. Setup Analysis in Nexus Devices at User Speed Grade

To meet setup time, the following conditions must be fulfilled:

$$\text{TC2Q} + \text{Tdatapath\_max} + \text{Tsetup} <= \text{Clock Period} + \text{Tskew},$$

where TC2Q is the Source Register TCO and Tskew is the clock skew.

In the Radiant software version 2023.1 and prior, the clock tree was assumed to be a balanced clock tree architecture. Hence, the clock tree skew is 0ps, as shown in Figure 5.2. In this scenario, the Tskew is derived from the following calculation:

$$\text{Tskew} = \text{Tclk2\_max} - \text{Tclk1\_max}$$

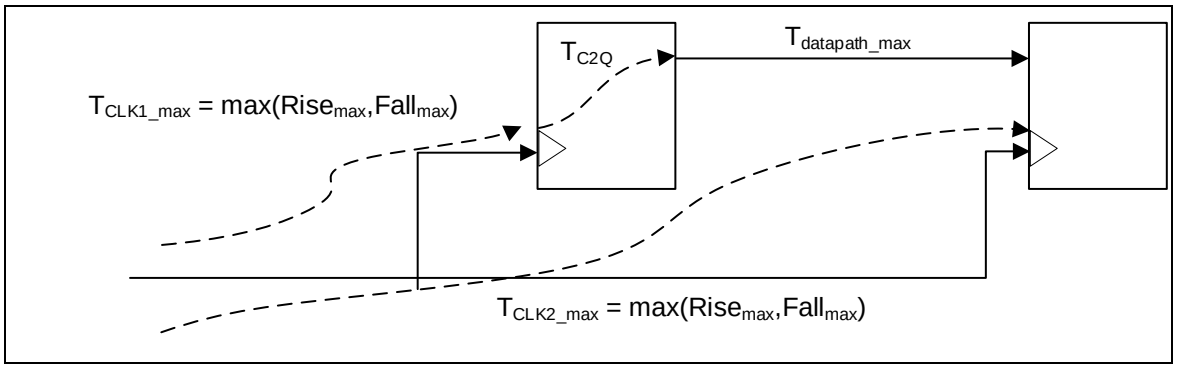


Figure 5.2. Clock Skew Analysis for Nexus Platform in the Radiant Software Version 2023.1 and Prior for Setup

Starting in the Radiant software version 2023.2**,** timing analysis for Nexus devices now accurately identifies common clock path, including its common path skew, as shown in Figure 5.3:

Tskew = Tclk2_min – Tclk1_max – (TcomPath_min – TcomPath_max)

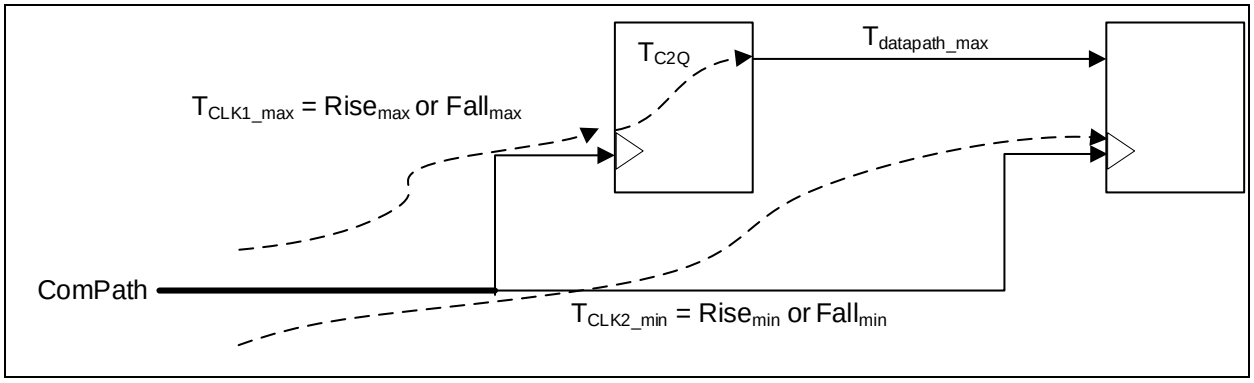


Figure 5.3. Clock Skew Analysis for Nexus Platform in the Radiant Software Version 2023.2 and Onwards for Setup

5.2.1.2. Hold Analysis in Nexus Devices at M Speed Grade

Similarly, to meet hold time, the following conditions must be fulfilled:

TC2Q + Tdatapath_min >= Tskew + Thold,

where TC2Q is the Source Register TCO and Tskew is the clock skew.

In the Radiant software version 2023.1 and prior, the clock tree was assumed to be a balanced clock tree architecture. Hence, the clock tree skew is 0ps, as shown in Figure 5.4. In this scenario, the Tskew is derived from the following calculation:

Tskew = Tclk2_min - Tclk1_min

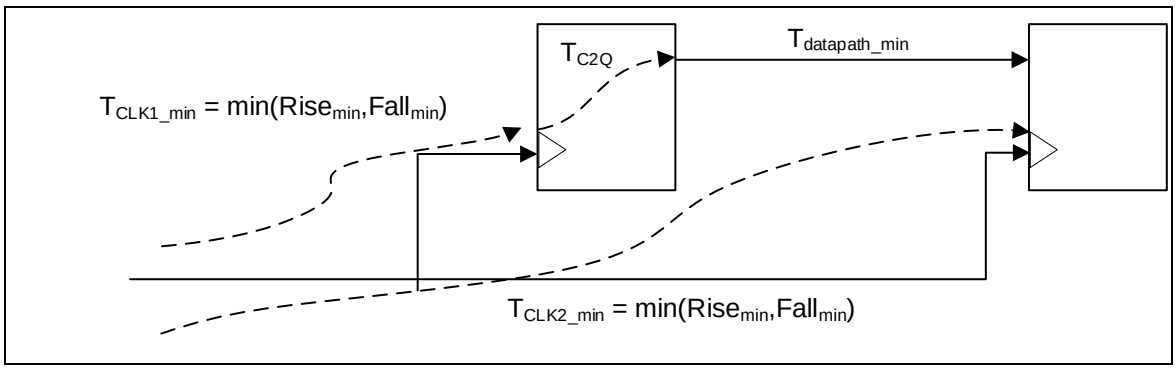


Figure 5.4. Clock Skew Analysis for Nexus Platform in the Radiant Software Version 2023.1 and Prior for Hold

Starting in the Radiant software version 2023.2, timing analysis for Nexus devices now accurately identifies common clock path, including its common path skew, as shown in Figure 5.5:

Tskew = Tclk2_max – Tclk1_min – (TcomPath_max – TcomPath_min)

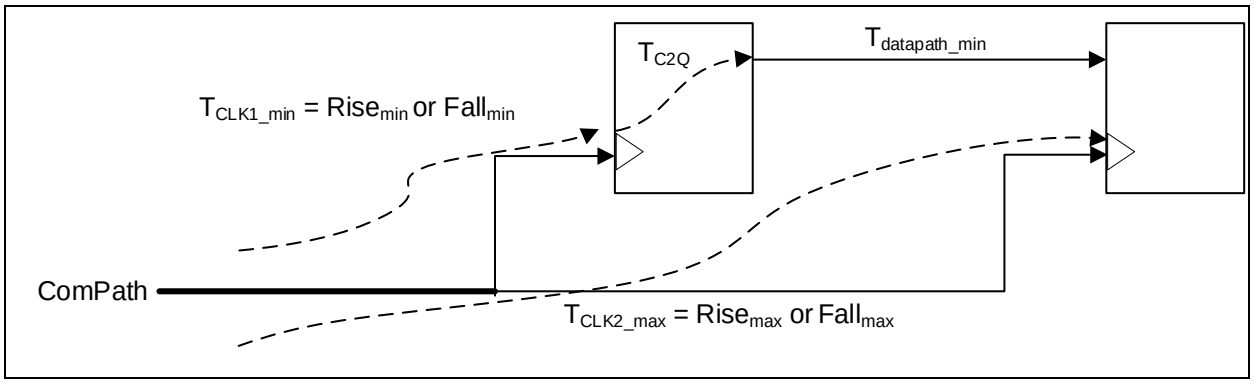


Figure 5.5. Clock Skew Analysis for Nexus Platform in the Radiant Software Version 2023.2 and Onwards for Hold

5.2.2. Derivation of Data Path Delay

There is also a change in how the data path delay from source to destination registers is calculated.

In the Radiant software version 2023.1 and prior, timing is slightly more pessimistic because it uses the worst-case delay for data path delay, regardless whether it is of rise or fall, as shown in the following equation:

Tdatapath_max = max (MaxDelay_Rise, MaxDelay_Fall)

Starting in the Radiant software version 2023.2, the derivation of data path delay now take into account the timing sense:

- Tdatapath_max = MaxDelay_Rise, if Rise
- Tdatapath_max = MaxDelay_Fall, if Fall
- Tdatapath_max = max (MaxDelay_Rise, MaxDelay_Fall), if the timing sense cannot be identified

5.2.3. Timing Report in the Radiant Software Version 2023.2

The timing report in the Radiant software version 2023.2 and onwards now include *Common Path Skew*, which was not available in earlier software versions. See Figure 5.6.

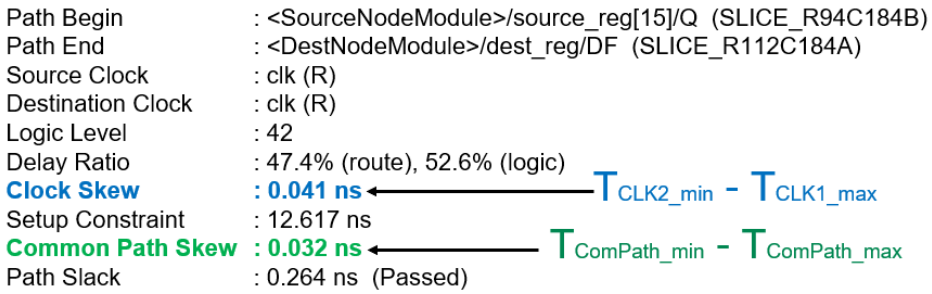


Figure 5.6. Timing Report Example in the Radiant Software Version 2023.2 for Nexus Devices

References

For more information, refer to the following resources:

- Lattice Radiant Software User Guide
- Lattice Radiant Software Help
- Lattice Radiant Timing Constraints Methodology Application Note (FPGA-AN-02059)
- Lattice Radiant Software web page
- Lattice Synthesis Engine web page
- Avant-E web page
- Avant-G web page
- Avant-X web page
- Lattice Nexus Platform 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 1.1, April 2025

Section	Change Summary
Abbreviations in This Document	Updated the section title and replaced acronyms with abbreviations.
Timing Closure with the Lattice Radiant Software	Updated the Understanding Timing Reports in the Radiant Software section.

Revision 1.0, January 2024

Section	Change Summary
All	Initial release.

www.latticesemi.com