

Vivado Design Suite Tutorial

Design Analysis and Closure Techniques

UG938 (v2022.1) May 11, 2022

Xilinx is creating an environment where employees, customers, and partners feel welcome and included. To that end, we're removing non-inclusive language from our products and related collateral. We've launched an internal initiative to remove language that could exclude people or reinforce historical biases, including terms embedded in our software and IPs. You may still find examples of non-inclusive language in our older products as we work to make these changes and align with evolving industry standards. Follow this [link](#) for more information.



Table of Contents

Tutorial Overview.....	4
Navigating Content by Design Process.....	4
Introduction.....	4
Tutorial Description.....	4
Software Requirements.....	5
Locating Tutorial Design Files.....	5
Lab 1: Setting Waivers with the Vivado IDE.....	6
Introduction.....	6
Step 1: Starting the Vivado IDE.....	6
Step 2: Generating the CDC Report.....	7
Step 3: Waiving a Single CDC Violation.....	8
Step 4: Generating a Report for Waived Violations.....	12
Step 5: Generating a Text Report with Details for Waived Violations.....	13
Step 6: Waiving Multiple CDC Violations.....	14
Step 7: Exporting Waivers.....	17
Step 8: Using the create_waiver Command.....	18
Step 9: Waiving Multiple CDC Violations.....	19
Step 10: Waiving Multiple DRC Violations.....	21
Step 11: Generating a Summary Report for Waived Violations.....	26
Step 12: Using Waiver Commands.....	29
Summary.....	30
Lab 2: Using Report QoR Suggestions and Report QoR Assessment for Timing Closure.....	31
Introduction.....	31
Step 1: Understanding the Design.....	32
Step 2: Running Report QoR Assessment.....	34
Step 3: Running Report QoR Suggestions.....	38
Step 4: Understanding the Report.....	39
Step 5: Run with Suggestions.....	43
Step 6: Accumulating Suggestions.....	47

Step 7: Automatically Running QoR Suggestions.....	49
Summary.....	52
Lab 3: Running ML Strategies	53
Introduction.....	53
Step 1: Generating an ML Strategy RQS File.....	53
Step 2: Creating ML Strategy Runs.....	56
Summary.....	57
Lab 4: Intelligent Design Runs.....	58
Introduction.....	58
Step 1: Creating Intelligent Design Runs.....	58
Step 2: Navigating Intelligent Design Runs.....	61
Step 3: Analyzing the Reports and Log File.....	63
Step 4: Final Analysis.....	66
Summary.....	67
Appendix A: Additional Resources and Legal Notices.....	68
Revision History.....	68
Please Read: Important Legal Notices.....	68

Tutorial Overview

Navigating Content by Design Process

Xilinx® documentation is organized around a set of standard design processes to help you find relevant content for your current development task. All Versal® ACAP design process [Design Hubs](#) and the [Design Flow Assistant](#) materials can be found on the [Xilinx.com](#) website. This document covers the following design processes:

- **Hardware, IP, and Platform Development:** Creating the PL IP blocks for the hardware platform, creating PL kernels, functional simulation, and evaluating the Vivado® timing, resource use, and power closure. Also involves developing the hardware platform for system integration. Topics in this document that apply to this design process include:
 - [Lab 2: Using Report QoR Suggestions and Report QoR Assessment for Timing Closure](#)
 - [Lab 3: Running ML Strategies](#)
 - [Lab 4: Intelligent Design Runs](#)

Introduction

This tutorial uses the Vivado® design rules checker (`report_drc`), clock domain crossing checker (`report_cdc`), and quality of results enhancer (`report_qor_suggestions`) to analyze example designs for issues, and shows you how to take corrective actions. It also outlines how to run ML strategies and Intelligent Design Runs (IDRs).

Tutorial Description

Lab 1 walks you through creating waivers for CDC, methodology, and DRC violations.

Lab 2 is a guide to using the `report_qor_suggestions` (RQS) command.

Note: The designs used in this tutorial are intended to exhibit issues for demonstration purposes, and should not be used as a reference for designs outside this tutorial.

Software Requirements

This tutorial requires that the 2021.1 Vivado® Design Suite software release or later is installed.

For a complete list of system and software requirements, see the *Vivado Design Suite User Guide: Release Notes, Installation, and Licensing* ([UG973](#)).

Locating Tutorial Design Files

1. Download the [reference design files](#) from the Xilinx® website.
2. Extract the ZIP file contents into any write-accessible location.

This tutorial refers to the location of the extracted ZIP file contents as <Extract_Dir>.

Setting Waivers with the Vivado IDE

Introduction

In the Vivado® Design Suite, you can use the waiver mechanism to waive clock domain crossing (CDC), design rule check (DRC), or methodology check violations. After a violation is waived, it is no longer reported by the `report_cdc`, `report_drc`, or `report_methodology` commands. Waived checks are also filtered out from the mandatory DRCs run at the start of the implementation commands, such as `opt_design`, `place_design`, and `route_design`. For more information, see this [link](#) in the *Vivado Design Suite User Guide: Design Analysis and Closure Techniques* (UG906).



IMPORTANT! The content of the waiver is built with the objects that exist when the waiver is created.

However, if an instance referenced inside a waiver is replicated by Vivado®, the replicated instance is automatically added to the waiver and saved in subsequent checkpoints and XDC.

This lab shows how to set waivers with the Vivado integrated design environment (IDE) using both menu commands and the Tcl Console. The lab focuses on CDC waivers, but the methods for waiving DRC and methodology violations are similar.

Step 1: Starting the Vivado IDE

This lab uses a Vivado design checkpoint (`.dcp` file), which is a snapshot of a design. When you launch the Vivado IDE using a design checkpoint, a subset of the Vivado IDE functionality is available.



TIP: To launch the Vivado Tcl Shell on Windows, select **Start**→**All Programs**→**Xilinx Design Tools**→**Vivado <version>**→**Vivado <version> Tcl Shell**.

1. From the command line or the Vivado Tcl Shell, change to the directory where the lab materials are stored:

```
cd <Extract_Dir>/Lab1
```

2. To start the Vivado IDE with the design checkpoint loaded, enter the following:

```
vivado my_ip_example_design_placed.dcp
```

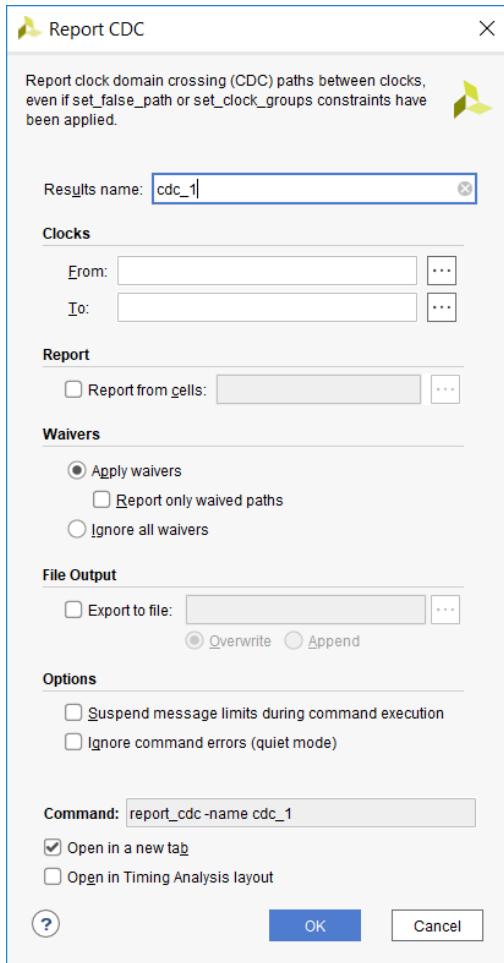


TIP: You can disregard the critical warnings about the unbounded GT locations.

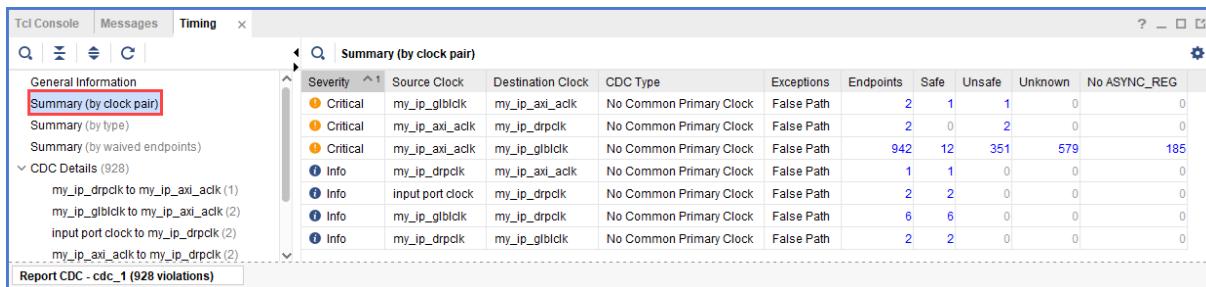
Step 2: Generating the CDC Report

In this step, you generate the CDC report to view the associated CDC violations.

1. Select **Reports**→**Timing**→**Report CDC**.
2. In the Report CDC dialog box, leave the default settings as-is, and click **OK**.

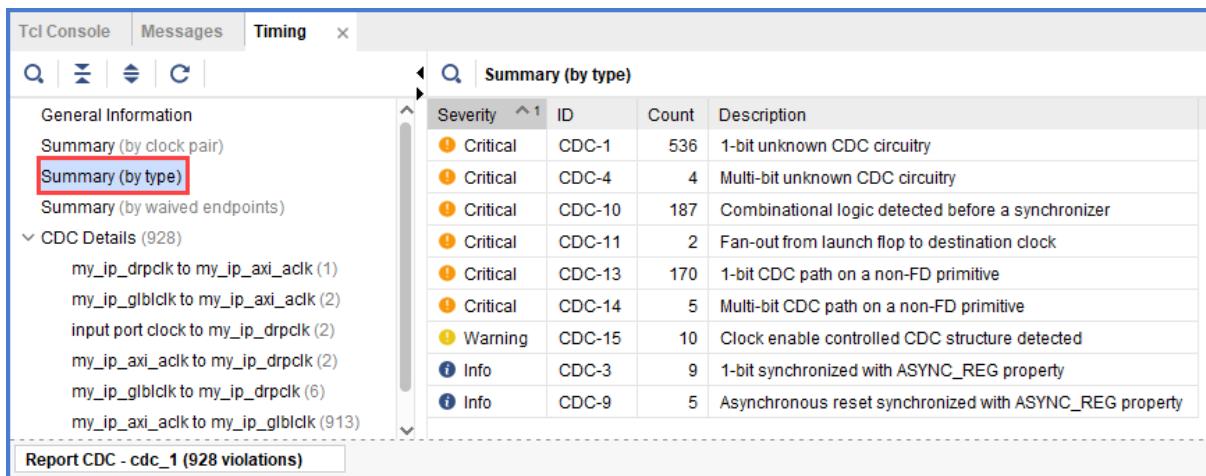


The Summary (by clock pair) section of the CDC Report appears as follows.



Severity	Source Clock	Destination Clock	CDC Type	Exceptions	Endpoints	Safe	Unsafe	Unknown	No ASYNC_REG
Critical	my_ip_glblclk	my_ip_axi_aclk	No Common Primary Clock	False Path	2	1	1	0	0
Critical	my_ip_axi_aclk	my_ip_drpclk	No Common Primary Clock	False Path	2	0	2	0	0
Critical	my_ip_axi_aclk	my_ip_glblclk	No Common Primary Clock	False Path	942	12	351	579	185
Info	my_ip_drpclk	my_ip_axi_aclk	No Common Primary Clock	False Path	1	1	0	0	0
Info	input port clock	my_ip_drpclk	No Common Primary Clock	False Path	2	2	0	0	0
Info	my_ip_glblclk	my_ip_drpclk	No Common Primary Clock	False Path	6	6	0	0	0
Info	my_ip_drpclk	my_ip_glblclk	No Common Primary Clock	False Path	2	2	0	0	0

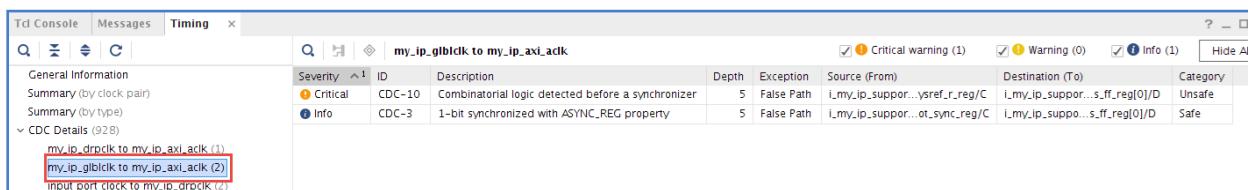
The Summary (by CDC type) section appears as follows.



Severity	ID	Count	Description
Critical	CDC-1	536	1-bit unknown CDC circuitry
Critical	CDC-4	4	Multi-bit unknown CDC circuitry
Critical	CDC-10	187	Combinational logic detected before a synchronizer
Critical	CDC-11	2	Fan-out from launch flop to destination clock
Critical	CDC-13	170	1-bit CDC path on a non-FD primitive
Critical	CDC-14	5	Multi-bit CDC path on a non-FD primitive
Warning	CDC-15	10	Clock enable controlled CDC structure detected
Info	CDC-3	9	1-bit synchronized with ASYNC_REG property
Info	CDC-9	5	Asynchronous reset synchronized with ASYNC_REG property

Step 3: Waiving a Single CDC Violation

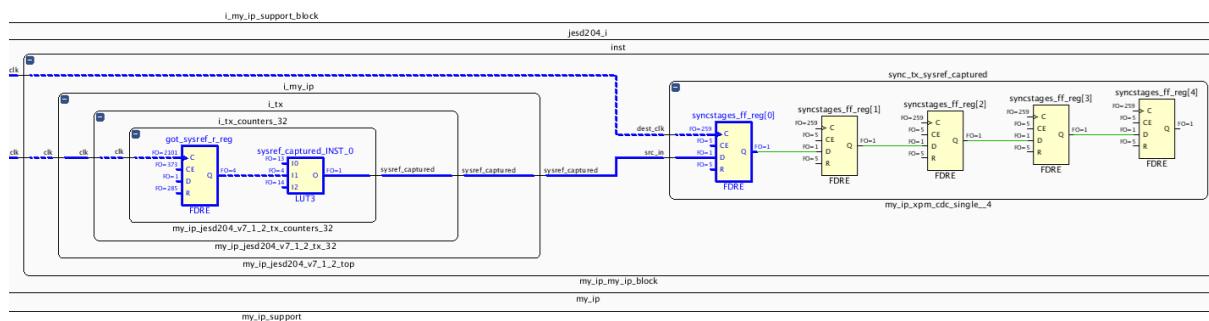
The `my_ip_glblclk to my_ip_axi_aclk` clock pair includes one Critical CDC-10 violation due to combinational logic on the CDC path. This step covers how to waive the CDC-10 violation.



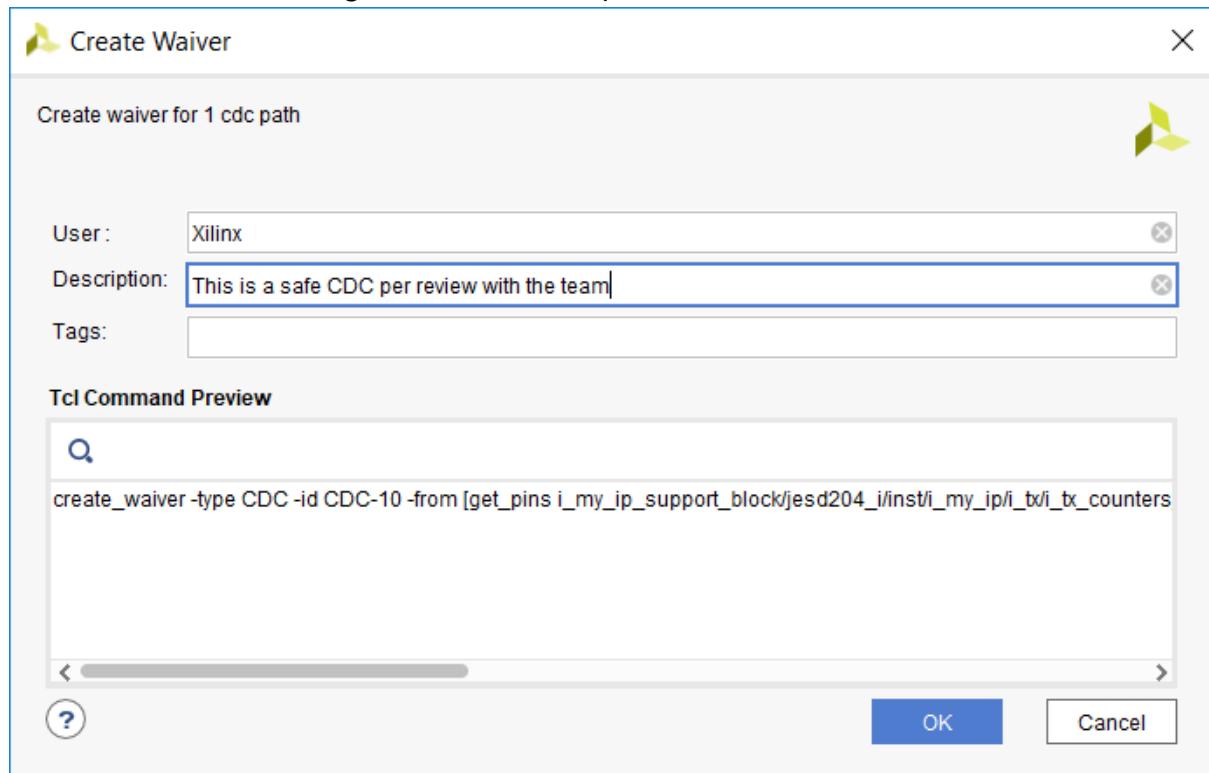
Severity	ID	Description	Depth	Exception	Source (From)	Destination (To)	Category
Critical	CDC-10	Combinatorial logic detected before a synchronizer	5	False Path	i_my_ip_support...ysref_r_reg/C	i_my_ip_support...s_ff_reg[0]/D	Unsafe
Info	CDC-3	1-bit synchronized with ASYNC_REG property	5	False Path	i_my_ip_support...ot_sync_rreg/C	i_my_ip_support...s_ff_reg[0]/D	Safe

1. To view a schematic of the violation, select the CDC-10 row in the CDC Report, and click the Schematic toolbar button .

Note: Alternatively, you can press **F4** to generate the schematic. However, using the toolbar button provides a more detailed schematic that includes all the levels of the downstream synchronizer.



2. To waive the violation, select the **CDC-10** row in the CDC Report, right-click, and select **Create Waiver**.
3. In the Create Waiver dialog box, enter a description, and click **OK**.



IMPORTANT! A waiver tracks the date the waiver was added, the user that added the waiver, and a description of why the violation was waived. The date is automatically added by the system. The Tags field is an optional description or list of keywords that can be used for documentation purposes.

4. After the waiver is created, check the CDC Report.

To indicate that a waiver was created, the CDC-10 row is gray and disabled.

Note: Rows are only disabled in the Report CDC result window from which the waivers were created.

my_ip_gblclk to my_ip_axi_aclk							Critical warning (1)		Warning (0)		Info (1)		Hide All	
Severity	ID	Description			Depth	Exception	Source (From)	Destination (To)	Category					
Critical	CDC-10	Combinational logic detected before a synchronizer	5	False Path	I_my_ip_support...sysref_r_reg/C	I_my_ip_support...es_ff_reg[0]D	Unsafe							
Info	CDC-3	1-bit synchronized with ASYNC_REG property	5	False Path	I_my_ip_support...ot_sync_reg/C	I_my_ip_support...es_ff_reg[0]D	Safe							
CDC Details (928)														
my_ip_drpclk to my_ip_axi_aclk (1) my_ip_gblclk to my_ip_axi_aclk (2) input port clock to my_ip_drpclk (2)														
Report CDC - cdc_1 (928 violations)														

- To see the impact of the CDC-10 waiver, select **Reports**→**Timing**→**Report CDC** to rerun Report CDC.

Note: When a waiver is created or deleted, you must rerun Report CDC, Report DRC, or Report Methodology to see the updated results.

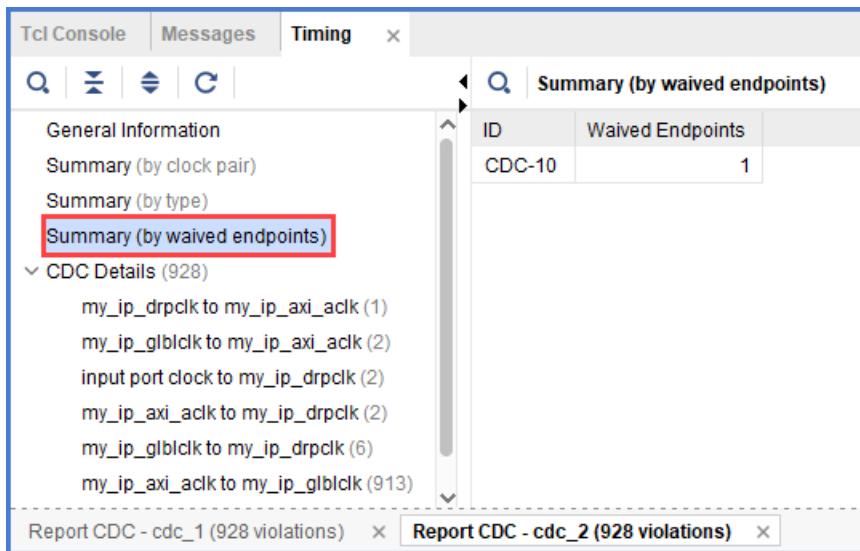
- See the CDC Report to view the updated information.

The differences from the previous Summary by clock pair and Summary by type sections are highlighted in red in the following figures.

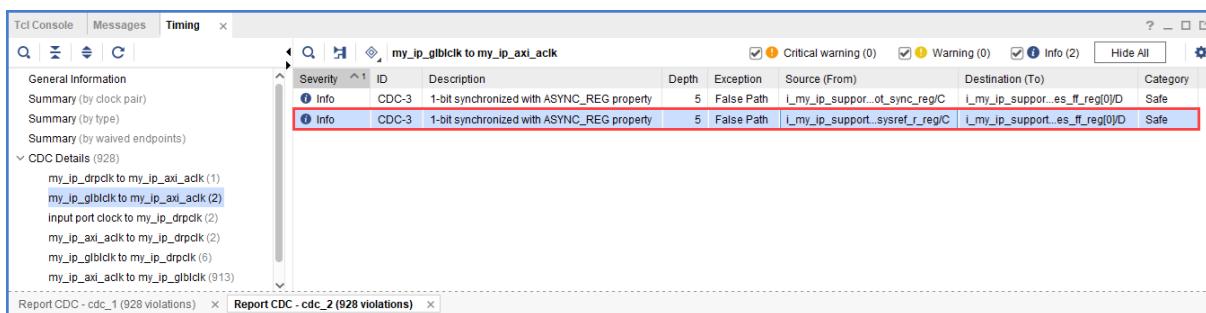
Summary (by clock pair)								
Severity	Source Clock	Destination Clock	CDC Type	Exceptions	Endpoints	Safe	Unsafe	Unknown
Critical	my_ip_axi_aclk	my_ip_drpclk	No Common Primary Clock	False Path	2	0	2	0
Critical	my_ip_axi_aclk	my_ip_gblclk	No Common Primary Clock	False Path	942	12	351	579
Info	my_ip_drpclk	my_ip_axi_aclk	No Common Primary Clock	False Path	1	1	0	0
Info	my_ip_gblclk	my_ip_axi_aclk	No Common Primary Clock	False Path	2	2	0	0
Info	input port clock	my_ip_drpclk	No Common Primary Clock	False Path	2	2	0	0
Info	my_ip_gblclk	my_ip_drpclk	No Common Primary Clock	False Path	6	6	0	0
Info	my_ip_drpclk	my_ip_gblclk	No Common Primary Clock	False Path	2	2	0	0
my_ip_gblclk to my_ip_drpclk (6) my_ip_axi_aclk to my_ip_drpclk (2)								
Report CDC - cdc_1 (928 violations) x Report CDC - cdc_2 (928 violations) x								

Summary (by type)						
Severity	ID	Count	Description			
Critical	CDC-1	536	1-bit unknown CDC circuitry			
Critical	CDC-4	4	Multi-bit unknown CDC circuitry			
Critical	CDC-10	186	Combinational logic detected before a synchronizer			
Critical	CDC-11	2	Fan-out from launch flop to destination clock			
Critical	CDC-13	170	1-bit CDC path on a non-FD primitive			
Critical	CDC-14	5	Multi-bit CDC path on a non-FD primitive			
Warning	CDC-15	10	Clock enable controlled CDC structure detected			
Info	CDC-3	10	1-bit synchronized with ASYNC_REG property			
Info	CDC-9	5	Asynchronous reset synchronized with ASYNC_REG property			
my_ip_drpclk to my_ip_axi_aclk (1) my_ip_gblclk to my_ip_axi_aclk (2) input port clock to my_ip_drpclk (2) my_ip_axi_aclk to my_ip_drpclk (2) my_ip_gblclk to my_ip_drpclk (6) my_ip_axi_aclk to my_ip_gblclk (913)						
Report CDC - cdc_1 (928 violations) x Report CDC - cdc_2 (928 violations) x						

You can also view a summary with the list of waived endpoints.

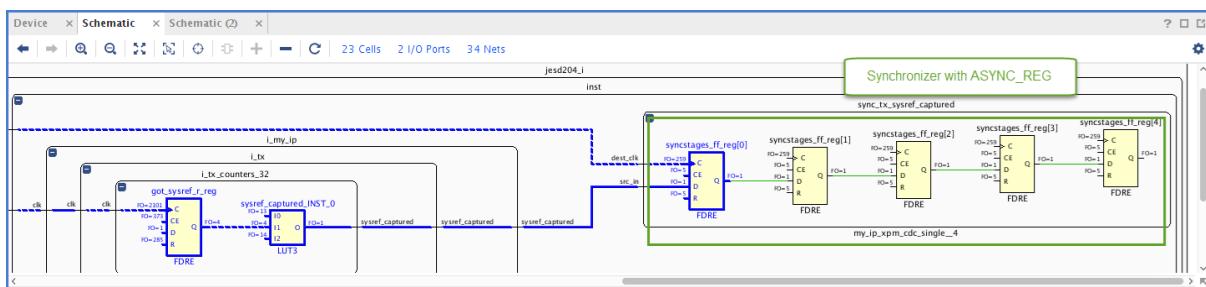


The detailed section for the `my_ip_glblclk to my_ip_axi_aclk` CDC shows that the Critical CDC-10 was replaced with an Info CDC-3.



7. Select the new CDC-3 row, and click the Schematic toolbar button . If F4 is used to open the schematic, double-click the Q pin of the output register to expand the schematic to match what is shown in the following figure.

The CDC path includes a 5-level synchronizer on the output of the selected destination register. This is the reason the CDC-10 was replaced with CDC-3 for this topology, as shown in the following figure.



IMPORTANT! By default, Report CDC only reports a single violation per endpoint and per clock pair. When multiple violations apply to the same endpoint, only the violation with the highest precedence is reported. Because CDC-10 has a higher precedence than CDC-3, only CDC-10 is reported when both CDC-10 and CDC-3 apply to the same endpoint. For more information on CDC rules precedence, see this [link](#) in the Vivado Design Suite User Guide: Design Analysis and Closure Techniques (UG906).

TIP: To report all of the CDC violations for each endpoint regardless of the precedence rules, use the command line option `-all_checks_per_endpoint`. However, unsafe rules are not reported on a register if at least one safe rule on the same register is detected.

Step 4: Generating a Report for Waived Violations

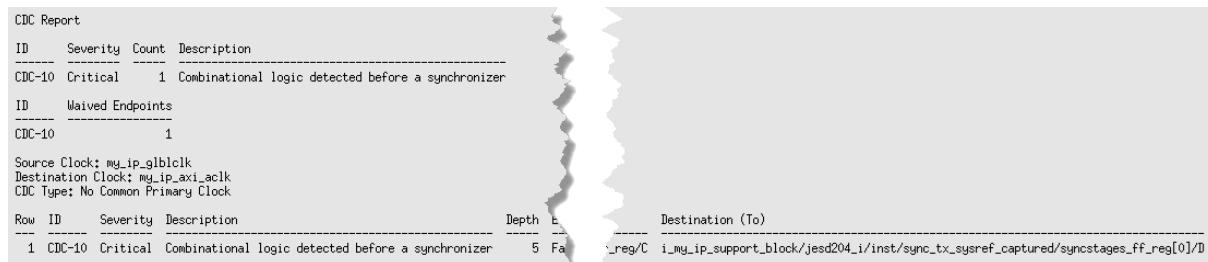
You can generate a report for the CDC, DRC, or methodology check violations that were waived. This step shows how to generate a report for waived CDC violations using the Tcl Console as well as the Vivado IDE menu commands.

Generating a Text Report for Waived Violations

1. In the Tcl Console, enter:

```
report_cdc -waived
```

2. In the CDC report, verify that a single CDC-10 violation is listed, because only one waiver was created.

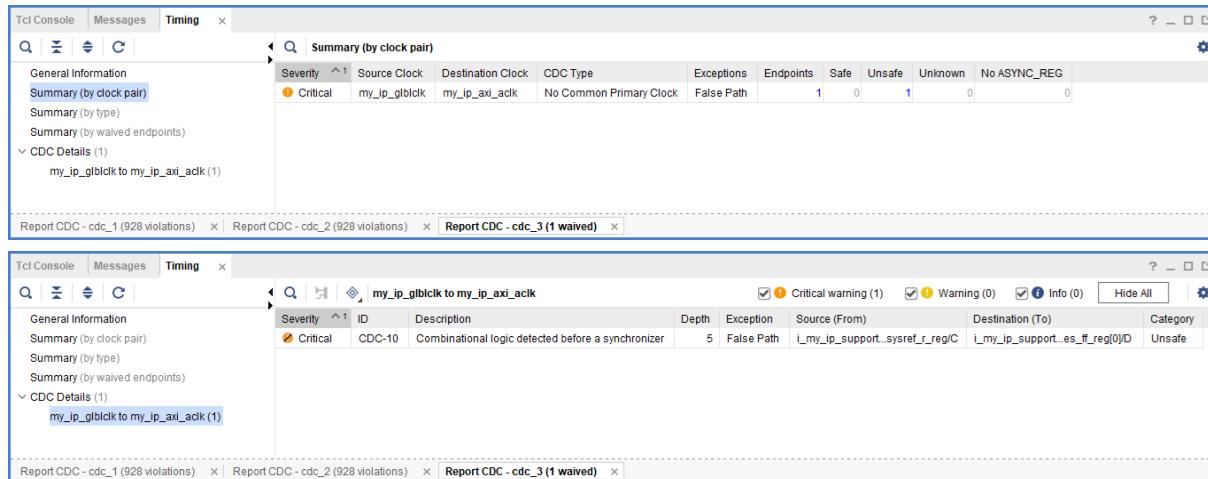


CDC Report								
ID	Severity	Count	Description					
CDC-10	Critical	1	Combinational logic detected before a synchronizer					
Waived Endpoints								
CDC-10		1						
Source Clock: my_ip_glbclk								
Destination Clock: my_ip_axi_aclk								
CDC Type: No Common Primary Clock								
Row	ID	Severity	Description		Depth			
1	CDC-10	Critical	Combinational logic detected before a synchronizer		5			
			Destination (To)					
			i_my_ip_support_block/jesd204_1/inst/sync_tx_sysref_captured/syncstages_ff_reg[0]/D					

Generating a Vivado IDE CDC Report for Waived Violations

1. Select **Reports**→**Timing**→**Report CDC**.
2. In the Report CDC dialog box, enable **Report only waived paths**, and click **OK**.
3. In the CDC Report, check the Summary (by clock pair) and CDC Details to verify that a single CDC-10 violation is listed.

Note: The icon next to the violation shows that the violation was waived 🎉.



The screenshot shows two windows of the Vivado IDE. The top window is titled 'Timing' and displays a 'Summary (by clock pair)' table. It shows one critical violation: 'my_ip_glblclk to my_ip_axi_aclk' with 'No Common Primary Clock'. The bottom window is also titled 'Timing' and shows a detailed view of the same violation, with a status bar indicating 'Report CDC - cdc_3 (1 waived)'.

Step 5: Generating a Text Report with Details for Waived Violations

In this step, you generate text reports with additional details, including a list of all of the rules and all of the violations regardless of the waivers.

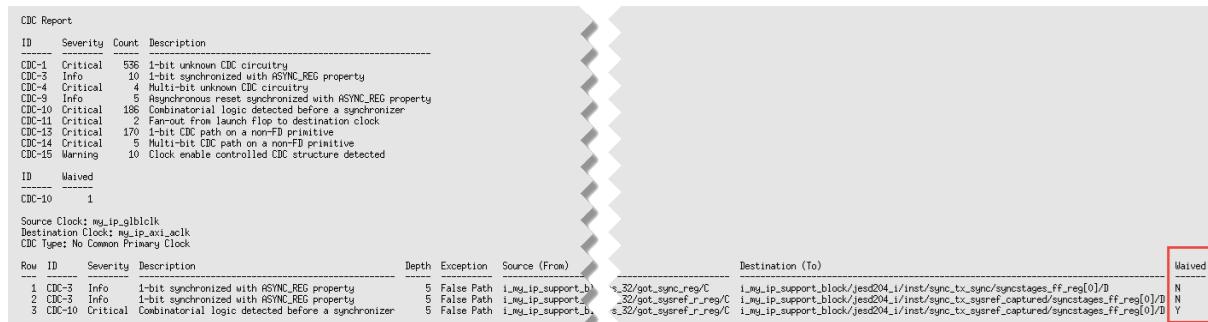
Generating a List of Rules with Waived Violations

1. In the Tcl Console, enter:

```
report_cdc -details -show_waiver
```

2. Verify that the `my_ip_glblclk to my_ip_axi_aclk` CDC-10 violation is waived and the two CDC-3 violations are not waived.

Note: In the text report, all of the rules are reported, whether they were waived or not. The Waived column indicates the status of the rule.



The screenshot shows a 'CDC Report' table with columns: ID, Severity, Count, Description, and Waived. It lists various CDC rules and their details. A red box highlights the 'Waived' column for the CDC-10 row, which corresponds to the violation in the screenshot above.

CDC Report								
ID	Severity	Count	Description	Waived				
CDC-1	Critical	536	1-bit unknown CDC circuitry					
CDC-3	Info	10	1-bit synchronized with ASYNC_REG property					
CDC-4	Critical	4	Multi-bit unknown CDC circuitry					
CDC-9	Info	5	Asynchronous reset synchronized with ASYNC_REG property					
CDC-10	Critical	182	Combinatorial logic detected before a synchronizer	1				
CDC-11	Critical	170	1-bit CDC path from launch flop to destination clock					
CDC-13	Critical	5	1-bit CDC path on a non-FD primitive					
CDC-14	Critical	5	Multi-bit CDC path on a non-FD primitive					
CDC-15	Warning	10	Clock enable controlled CDC structure detected					
ID	Waived							
CDC-10	1							
Source Clock: my_ip_glblclk Destination Clock: my_ip_axi_aclk CDC Type: No Common Primary Clock								
Row	ID	Severity	Description	Depth	Exception	Source (From)	Destination (To)	Waived
1	CDC-3	Info	1-bit synchronized with ASYNC_REG property	5	False Path	i_my_ip_support_b	s_32/got_sync_reg/C i_my_ip_support_block/jesd204_i/inst/sync_tx/syncstages_ff_reg[0]/D	N
2	CDC-3	Info	1-bit synchronized with ASYNC_REG property	5	False Path	i_my_ip_support_b	s_32/got_sysref_r_reg/C i_my_ip_support_block/jesd204_i/inst/sync_tx/syncstages_ff_reg[0]/D	N
3	CDC-10	Critical	Combinatorial logic detected before a synchronizer	5	False Path	i_my_ip_support_b	s_32/got_sysref_r_reg/C i_my_ip_support_block/jesd204_i/inst/sync_tx/syncstages_ff_reg[0]/D	Y

Generating a List of All Violations Regardless of the Waivers

1. In the Tcl Console, enter:

```
report_cdc -no_waiver
```

2. In the text report, verify that the table matches the original report from Report CDC before the CDC-10 waiver was created.

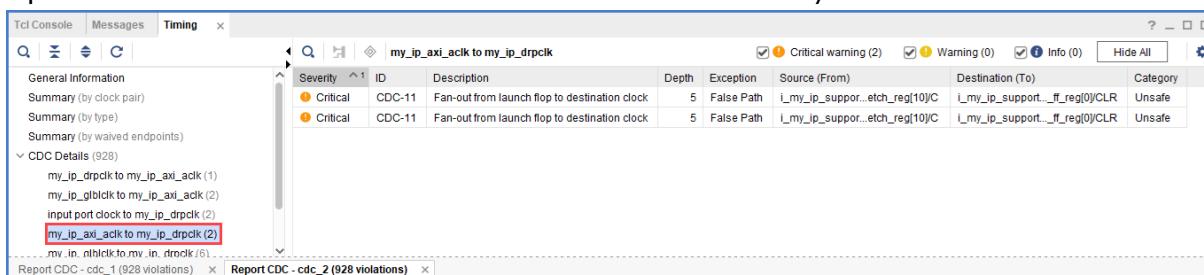
CDC Report										
Severity	Source Clock	Destination Clock	CDC Type	Exceptions	Endpoints	Safe	Unsafe	Unknown	No ASYNC_REG	
Critical	my_ip_glblclk	my_ip_axi_aclk	No Common Primary Clock	False Path	2	1	1	0	0	
Critical	my_ip_axi_aclk	my_ip_drpclk	No Common Primary Clock	False Path	2	0	2	0	0	
Critical	my_ip_axi_aclk	my_ip_glblclk	No Common Primary Clock	False Path	942	12	351	579	185	
Info	my_ip_drpclk	my_ip_axi_aclk	No Common Primary Clock	False Path	1	1	0	0	0	
Info	input port clock	my_ip_drpclk	No Common Primary Clock	False Path	2	2	0	0	0	
Info	my_ip_glblclk	my_ip_drpclk	No Common Primary Clock	False Path	6	6	0	0	0	
Info	my_ip_drpclk	my_ip_glblclk	No Common Primary Clock	False Path	2	2	0	0	0	



TIP: You can also generate a list of all violations regardless of the waivers from the Vivado IDE. Select **Reports**→**Timing**→**Report CDC**. In the Report CDC dialog box, enable **Ignore all waivers**, and click **OK**.

Step 6: Waiving Multiple CDC Violations

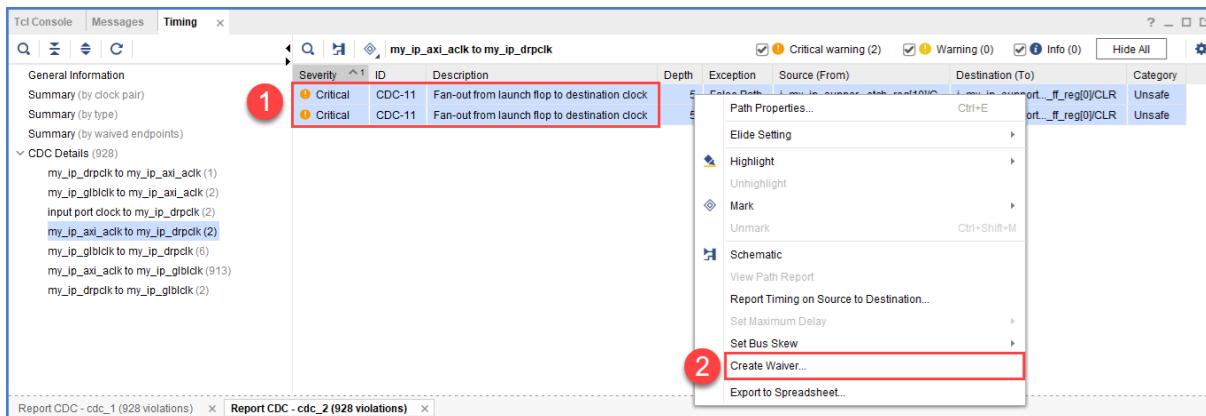
The `my_ip_axi_aclk` to `my_ip_drpclk` CDC includes two Critical CDC-11 violations. This step covers how to waive both CDC-11 violations simultaneously.



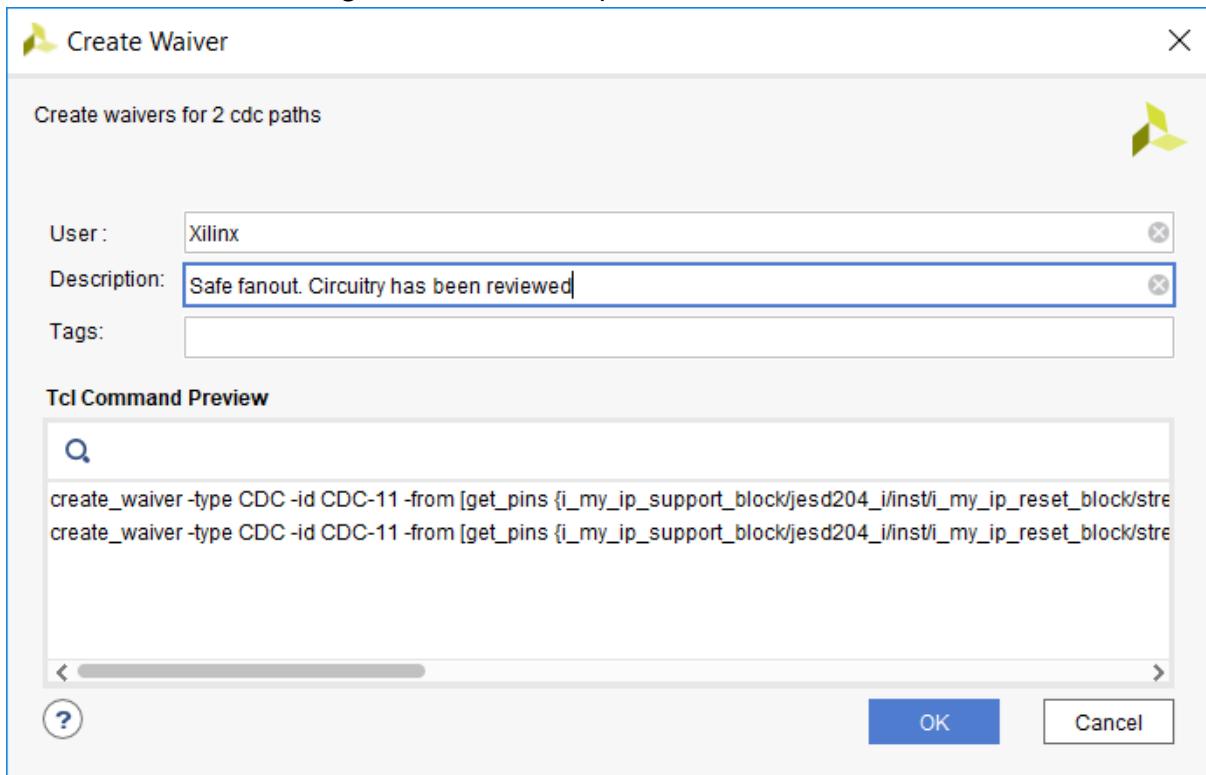
The screenshot shows the Vivado IDE's CDC Report table. The table lists violations for the `my_ip_axi_aclk` to `my_ip_drpclk` pair. There are two Critical CDC-11 violations, both of which are selected (highlighted with a red border). The violations are:

- `my_ip_drpclk to my_ip_axi_aclk (1)`
- `my_ip_glblclk to my_ip_axi_aclk (2)`
- `input port clock to my_ip_drpclk (2)`
- `my_ip_axi_aclk to my_ip_drpclk (2)`

1. To waive the violations, select the **CDC-11** rows in the CDC Report, right-click, and select **Create Waiver**.



2. In the Create Waiver dialog box, enter a description, and click **OK**.



In the Timing Report, the two selected rows are disabled when the waivers are created.

Note: One waiver is created for each selected row. In this example, two waivers are created.

Severity	ID	Description	Depth	Exception	Source (From)	Destination (To)	Category
Critical	CDC-11	Fan-out from launch flop to destination clock	5	False Path	i_my_ip_support...tch_reg[10]/C	i_my_ip_support...ff_reg[0]/CLR	Unsafe
Critical	CDC-11	Fan-out from launch flop to destination clock	5	False Path	i_my_ip_support...tch_reg[10]/C	i_my_ip_support...ff_reg[0]/CLR	Unsafe

3. Select **Reports** → **Timing** → **Report CDC** to rerun Report CDC. In the Report CDC dialog box, make sure that *Report only waived paths* is unchecked, and click **OK**.
4. In the CDC Report, look at the `my_ip_axi_aclk to my_ip_drpclk` CDC.

The two Critical CDC-11 violations were replaced with two Info CDC-9 violations. Based on the CDC precedence rules, waiving CDC-11 unmasks CDC-9 for this circuit.

Tcl Console Messages Timing ×

General Information

Summary (by clock pair)

Summary (by type)

Summary (by waived endpoints)

CDC Details (928)

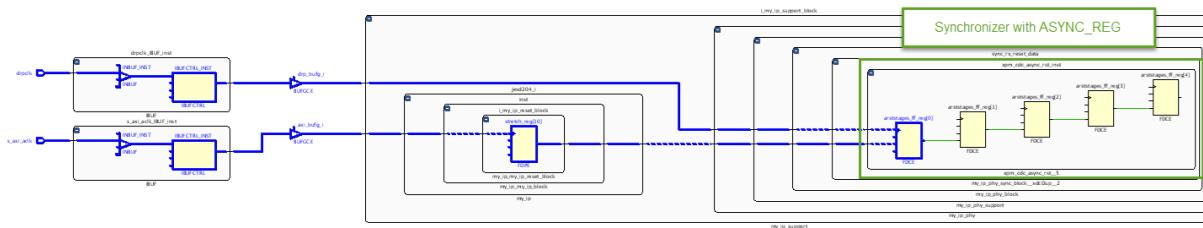
- my_ip_drpclk to my_ip_axi_... (1)
- my_ip_gblclk to my_ip_axi_... (2)
- input port clock to my_ip_drpclk (2)
- my_ip_axi_... to my_ip_drpclk (2)
- my_ip_gblclk to my_ip_drpclk (5)
- my_ip_axi_... to my_ip_gblclk (913)
- my_ip_drpclk to my_ip_gblclk (2)

my_ip_axi_... to my_ip_drpclk

Severity	ID	Description	Depth	Exception	Source (From)	Destination (To)	Category
Info	CDC-9	Asynchronous reset synchronized with ASYNC_REG property	5	False Path	i_my_ip_support_ff_reg[10]C	i_my_ip_support_ff_reg[0]CLR	Safe
Info	CDC-9	Asynchronous reset synchronized with ASYNC_REG property	5	False Path	i_my_ip_support_ff_reg[10]C	i_my_ip_support_ff_reg[0]CLR	Safe

Report CDC - cdc_1 (928 violations) × Report CDC - cdc_2 (928 violations) × Report CDC - cdc_3 (928 violations) ×

5. To view a schematic of the violation, select the CDC-9 row in the CDC Report, and click the Schematic toolbar button  .
6. Verify that there is a 5-level synchronizer on the destination clock domain.



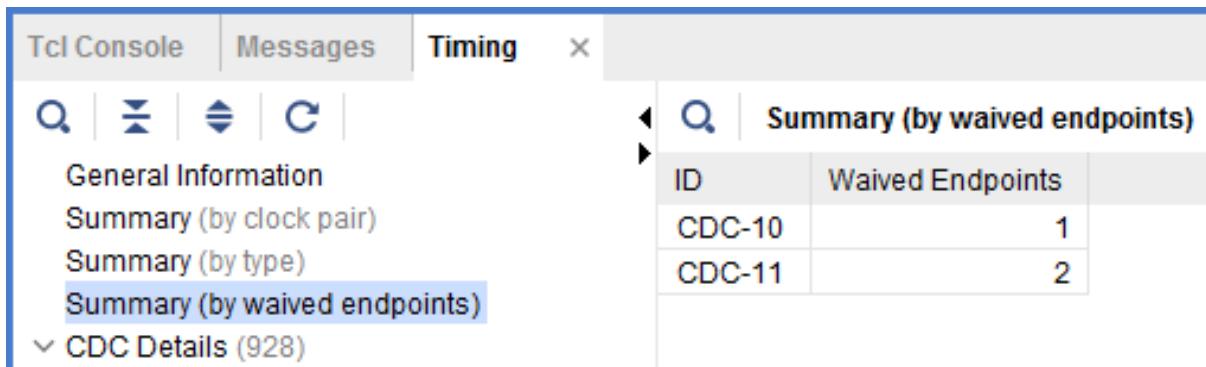
7. Compare the new Summary (by type) information with the information from the previous CDC Report.

In the updated CDC Report, the two CDC-11 violations are no longer listed. Instead, there are two new CDC-9 violations.

Timing				
Summary (by type)				
Severity	ID	Count	Description	
! Critical	CDC-1	536	1-bit unknown CDC circuitry	
! Critical	CDC-4	4	Multi-bit unknown CDC circuitry	
! Critical	CDC-10	186	Combinational logic detected before a synchronizer	
! Critical	CDC-13	170	1-bit CDC path on a non-FD primitive	
! Critical	CDC-14	5	Multi-bit CDC path on a non-FD primitive	
! Warning	CDC-15	10	Clock enable controlled CDC structure detected	
! Info	CDC-3	10	1-bit synchronized with ASYNC_REG property	
! Info	CDC-9	7	Asynchronous reset synchronized with ASYNC_REG property	

8. Look at the Summary (by waived endpoints) information.

In the updated CDC Report, there are three waived endpoints. This number is different from the number of waived violations (2), because CDC-11 is a multi-bit violation.



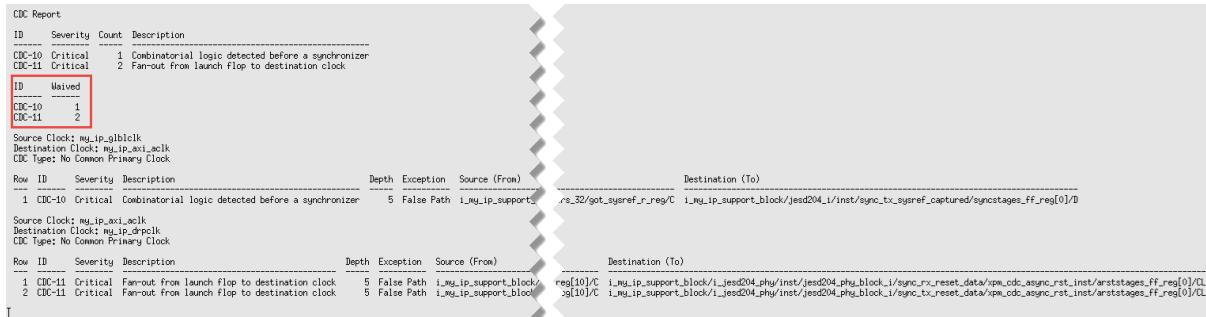
ID	Waived Endpoints
CDC-10	1
CDC-11	2

9. Generate different text reports and compare the results with previous reports.

For example, you can run the following Tcl commands:

```
report_cdc -details
report_cdc -details -waived
report_cdc -details -show_waiver
report_cdc -details -no_waiver
```

The following report was generated using the `report_cdc -details -waived` Tcl command and shows that three violations were waived.



ID	Severity	Count	Description
CDC-10	Critical	1	Combinatorial logic detected before a synchronizer
CDC-11	Critical	2	Fan-out from launch flop to destination clock

ID	Waived
CDC-10	1
CDC-11	2

Source Clock: `u_MIP_AXI_ACLK`
 Destination Clock: `u_MIP_AXI_ACLK`
 CDC Type: No Common Primary Clock

Row	ID	Severity	Description	Depth	Exception	Source (From)	Destination (To)
1	CDC-10	Critical	Combinatorial logic detected before a synchronizer	5	False Path	<code>l_MIP_SUPPORT_BLOCK/l_jesd204_pHY_block/l_sync_rx_reset_data/xpa_cdc_async_rst_inst/arststages_ff_reg[0]/D</code>	<code>rs_32/got_susref_r_reg/C</code>

Source Clock: `u_MIP_AXI_ACLK`
 Destination Clock: `u_MIP_AXI_ACLK`
 CDC Type: No Common Primary Clock

Row	ID	Severity	Description	Depth	Exception	Source (From)	Destination (To)
1	CDC-11	Critical	Fanout from launch flop to destination clock	5	False Path	<code>l_MIP_SUPPORT_BLOCK/l_jesd204_pHY_block/l_sync_tx_susref_captured/syncstages_ff_reg[0]/D</code>	<code>reg[10]/C</code>
2	CDC-11	Critical	Fanout from launch flop to destination clock	5	False Path	<code>l_MIP_SUPPORT_BLOCK/l_jesd204_pHY_block/l_sync_tx_susref_captured/syncstages_ff_reg[0]/D</code>	<code>reg[10]/C</code>

Step 7: Exporting Waivers

In this step, you export waivers with the `write_waivers` Tcl command.

Note: The XDC output file can be imported using the `read_xdc` or `source` Tcl commands.

1. To export the CDC waivers, enter: `write_waivers -type cdc waivers.xdc`.



TIP: Alternatively, because there are no DRC or methodology waivers, you can enter:

```
write_waivers waivers.xdc or write_xdc -type waiver waivers.xdc
```

2. Open the `waivers.xdc` file to view the three waivers.

Note: The following example is reformatted to better show the different command line options.

```
create_waiver -type CDC -id {CDC-10} -user "Xilinx" \
  -desc "This is a safe CDC per review with the team" \
  -from [get_pins i_my_ip_support_block/jesd204_i/inst/i_my_ip/i_tx/
i_tx_counters_32/got_sysref_r_reg/C] \
  -to [get_pins {i_my_ip_support_block/jesd204_i/inst/
sync_tx_sysref_captured/syncstages_ff_reg[0]/D}] \
  -timestamp "<timestamp>" ;#1

create_waiver -type CDC -id {CDC-11} -user "Xilinx" \
  -desc "Safe fanout. Circuitry has been released" \
  -from [get_pins {i_my_ip_support_block/jesd204_i/inst/
i_my_ip_reset_block/stretch_reg[10]/C}] \
  -to [get_pins {i_my_ip_support_block/i_jesd204_phy/inst/
jesd204_phy_block_i(sync_rx_reset_data/xpm_cdc_async_rst_inst/
arststages_ff_reg[0]/CLR)}] \
  -timestamp "<timestamp>" ;#1

create_waiver -type CDC -id {CDC-11} -user "Xilinx" \
  -desc "Safe fanout. Circuitry has been released" \
  -from [get_pins {i_my_ip_support_block/jesd204_i/inst/
i_my_ip_reset_block/stretch_reg[10]/C}] \
  -to [get_pins {i_my_ip_support_block/i_jesd204_phy/inst/
jesd204_phy_block_i(sync_tx_reset_data/xpm_cdc_async_rst_inst/
arststages_ff_reg[0]/CLR)}] \
  -timestamp "<timestamp>" ;#2
```

Step 8: Using the `create_waiver` Command

Waivers added from the Report CDC dialog box are created using the `create_waiver` command. You can view these commands as follows.

Note: You can use the `create_waiver` command line command for CDC, DRC, and methodology waivers. The options differ slightly depending on whether you are creating a CDC, DRC, or methodology waiver. For more information, including information on the different options, see the [create_waiver](#) command in the Vivado Design Suite Tcl Command Reference Guide ([UG835](#)).

1. Open the Vivado journal file (`vivado.jou`) to see the three distinct `create_waiver` commands issued by the Vivado IDE.
2. Scroll through the history of the Tcl Console to see the same three `create_waiver` commands.



TIP: The `-from` and `-to` options are used to specify the startpoints and endpoints. When a waiver is set from the Report CDC dialog box, both `-from` and `-to` are specified to match the exact violation. However, you can specify a CDC waiver using only the `-from` option or only the `-to` option, but more paths might be waived than expected.

Step 9: Waiving Multiple CDC Violations

In this step, you waive multiple CDC violations simultaneously.

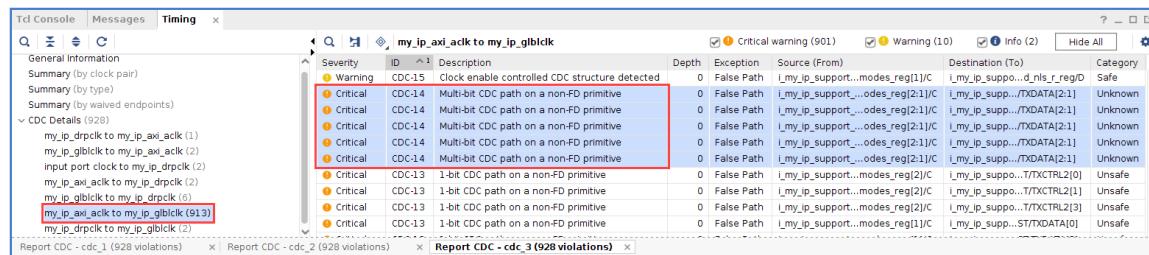
1. In the CDC Report, view the `my_ip_axi_aclk` to `my_ip_glbclk` CDC under CDC Details.

This crossing has five CDC-14 violations, which are multi-bit violations. The five CDC-14 violations all start from the same two register clock pins:

`i_my_ip_support_block/jesd204_i/inst/tx_cfg_test_modes_reg[2:1]/C`



TIP: You can sort the table by the column ID to more easily see the five CDC-14 violations.



Severity	ID	Description	Depth	Exception	Source (From)	Destination (To)	Category
Warning	CDC-15	Clock enable controlled CDC structure detected	0	False Path	<code>i_my_ip_support...modes_reg[1]/C</code>	<code>i_my_ip_support...d_nls_r_reg[0]</code>	Safe
Critical	CDC-14	Multi-bit CDC path on a non-FD primitive	0	False Path	<code>i_my_ip_support...odes_reg[2:1]/C</code>	<code>i_my_ip_support.../TxDATA[2:1]</code>	Unknown
Critical	CDC-14	Multi-bit CDC path on a non-FD primitive	0	False Path	<code>i_my_ip_support...odes_reg[2:1]/C</code>	<code>i_my_ip_support.../TxDATA[2:1]</code>	Unknown
Critical	CDC-14	Multi-bit CDC path on a non-FD primitive	0	False Path	<code>i_my_ip_support...odes_reg[2:1]/C</code>	<code>i_my_ip_support.../TxDATA[2:1]</code>	Unknown
Critical	CDC-14	Multi-bit CDC path on a non-FD primitive	0	False Path	<code>i_my_ip_support...odes_reg[2:1]/C</code>	<code>i_my_ip_support.../TxDATA[2:1]</code>	Unknown
Critical	CDC-14	Multi-bit CDC path on a non-FD primitive	0	False Path	<code>i_my_ip_support...odes_reg[2:1]/C</code>	<code>i_my_ip_support.../TxDATA[2:1]</code>	Unknown
Critical	CDC-13	1-bit CDC path on a non-FD primitive	0	False Path	<code>i_my_ip_support...modes_reg[0]/C</code>	<code>i_my_ip_support.../T/CTRL2[0]</code>	Unsafe
Critical	CDC-13	1-bit CDC path on a non-FD primitive	0	False Path	<code>i_my_ip_support...modes_reg[0]/C</code>	<code>i_my_ip_support.../T/CTRL2[1]</code>	Unsafe
Critical	CDC-13	1-bit CDC path on a non-FD primitive	0	False Path	<code>i_my_ip_support...modes_reg[0]/C</code>	<code>i_my_ip_support.../T/CTRL2[2]</code>	Unsafe
Critical	CDC-13	1-bit CDC path on a non-FD primitive	0	False Path	<code>i_my_ip_support...modes_reg[0]/C</code>	<code>i_my_ip_support.../T/CTRL2[3]</code>	Unsafe
Critical	CDC-13	1-bit CDC path on a non-FD primitive	0	False Path	<code>i_my_ip_support...modes_reg[0]/C</code>	<code>i_my_ip_support.../T/TxDATA[0]</code>	Unsafe

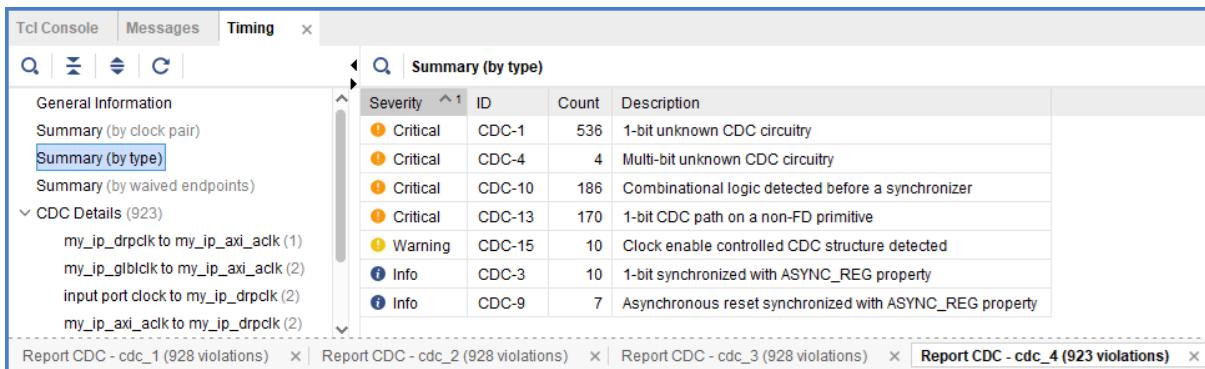
2. Because `i_my_ip_support_block/jesd204_i/inst/tx_cfg_test_modes_reg[*]/C` matches five pins and you only need to target two of those five pins, construct the list of startpoints as follows:

```
set startpoints [list \
    [get_pins i_my_ip_support_block/jesd204_i/inst/
    tx_cfg_test_modes_reg[1]/C] \
    [get_pins i_my_ip_support_block/jesd204_i/inst/
    tx_cfg_test_modes_reg[2]/C] \
]
```

3. To waive the five CDC-14 violations, use the `create_waiver` Tcl command with the `-from` option:

```
create_waiver -type {CDC} -id {CDC-14} -user {Xilinx} -desc {No more CDC 14!} -from $startpoints
```

4. From the Vivado IDE, select **Reports**→**Timing**→**Report CDC** to rerun Report CDC.
5. In the CDC Report, verify that the CDC-14 violations are no longer reported in the Summary section.



6. To report only the waived violations, enter:

```
report_cdc -details -waived
```

The following figure shows the waived CDC violations in two different tables. The first table shows the 5 CDC-14 violations waived as multi-bit violations. The second table shows the 10 single-bit violations, calculated by multiplying the 5 multi-bit violations by 2 bits per multi-bit violation.

ID	Severity	Count	Description
CDC-10	Critical	1	Combinatorial logic detected before a synchronizer
CDC-11	Critical	2	Fan-out from launch flop to destination clock
CDC-14	Critical	5	Multi-bit CDC path on a non-FD primitive

ID	Waived
CDC-10	1
CDC-11	2
CDC-14	10

Row	ID	Severity	Description	Depth	Exception	Source (From)
1	CDC-10	Critical	Combinatorial logic detected before a synchronizer	5	False Path	i_my_ip_support_block/jesd204_i/inst/i_my_ip/i_tx/i_tx_counters_32/go.

Row	ID	Severity	Description	Depth	Exception	Source (From)	Destination
1	CDC-11	Critical	Fan-out from launch flop to destination clock	5	False Path	i_my_ip_support_block/jesd204_i/inst/i_my_ip/reset_block/stretch_reg[10]/C	i_my_ip_suppo
2	CDC-11	Critical	Fan-out from launch flop to destination clock	5	False Path	i_my_ip_support_block/jesd204_i/inst/i_my_ip/reset_block/stretch_reg[10]/U	i_my_ip_suppo
1	CDC-14	Critical	Multi-bit CDC path on a non-FD primitive	0	False Path	i_my_ip_support_block/jesd204_i/inst/tx_cfg_test_modes_reg[2:1]/C	i_my_ip_suppo
2	CDC-14	Critical	Multi-bit CDC path on a non-FD primitive	0	False Path	i_my_ip_support_block/jesd204_i/inst/tx_cfg_test_modes_reg[2:1]/C	i_my_ip_suppo
3	CDC-14	Critical	Multi-bit CDC path on a non-FD primitive	0	False Path	i_my_ip_support_block/jesd204_i/inst/tx_cfg_test_modes_reg[2:1]/C	i_my_ip_suppo
4	CDC-14	Critical	Multi-bit CDC path on a non-FD primitive	0	False Path	i_my_ip_support_block/jesd204_i/inst/tx_cfg_test_modes_reg[2:1]/C	i_my_ip_suppo
5	CDC-14	Critical	Multi-bit CDC path on a non-FD primitive	0	False Path	i_my_ip_support_block/jesd204_i/inst/tx_cfg_test_modes_reg[2:1]/C	i_my_ip_suppo

7. To export all the waivers inside a script and verify that a total of four waivers were added, enter:

```
write_waivers -type cdc waivers.xdc -force
```

Note: Because the waivers.xdc file already exists, the -force option must be specified to override the file.



TIP: Alternatively, because there are no DRC or methodology waivers, you can enter:

```
write_waivers waivers.xdc -force
```

or

```
write_xdc -type waiver waivers.xdc -force
```

The list of waivers inside `waivers.xdc` appears as follows.

```
#  
# WRITE CDC Waivers  
# cmd: write_waivers -type odc -file waivers.xdc -force  
current_instance -quiet  
create_waiver -type CDC -id {CDC-10} -user "Xilinx" -desc "This is a safe CDC per review with the team" -from [get_pins i_my_ip.support_block/jesd204_1/inst/i_tx/i_tx_counters_32/got_0]  
create_waiver -type CDC -id {CDC-11} -user "Xilinx" -desc "Safe fanout. Circuitry has been released" -from [get_pins {i_my_ip.support_block/jesd204_1/inst/i_my_ip.reset_block/stretch_reg10}]  
create_waiver -type CDC -id {CDC-11} -user "Xilinx" -desc "Safe fanout. Circuitry has been released" -from [get_pins {i_my_ip.support_block/jesd204_1/inst/i_my_ip.reset_block/stretch_reg10}]  
create_waiver -type CDC -id {CDC-14} -user "Xilinx" -desc "No more CDC-14!" -from [list [get_pins {i_my_ip.support_block/jesd204_1/inst/bx_cfg_test_modes_reg[1]/C}] [get_pins {i_my_ip.support_
```

8. To import the `waivers.xdc` file, enter:

```
read_xdc waivers.xdc
```

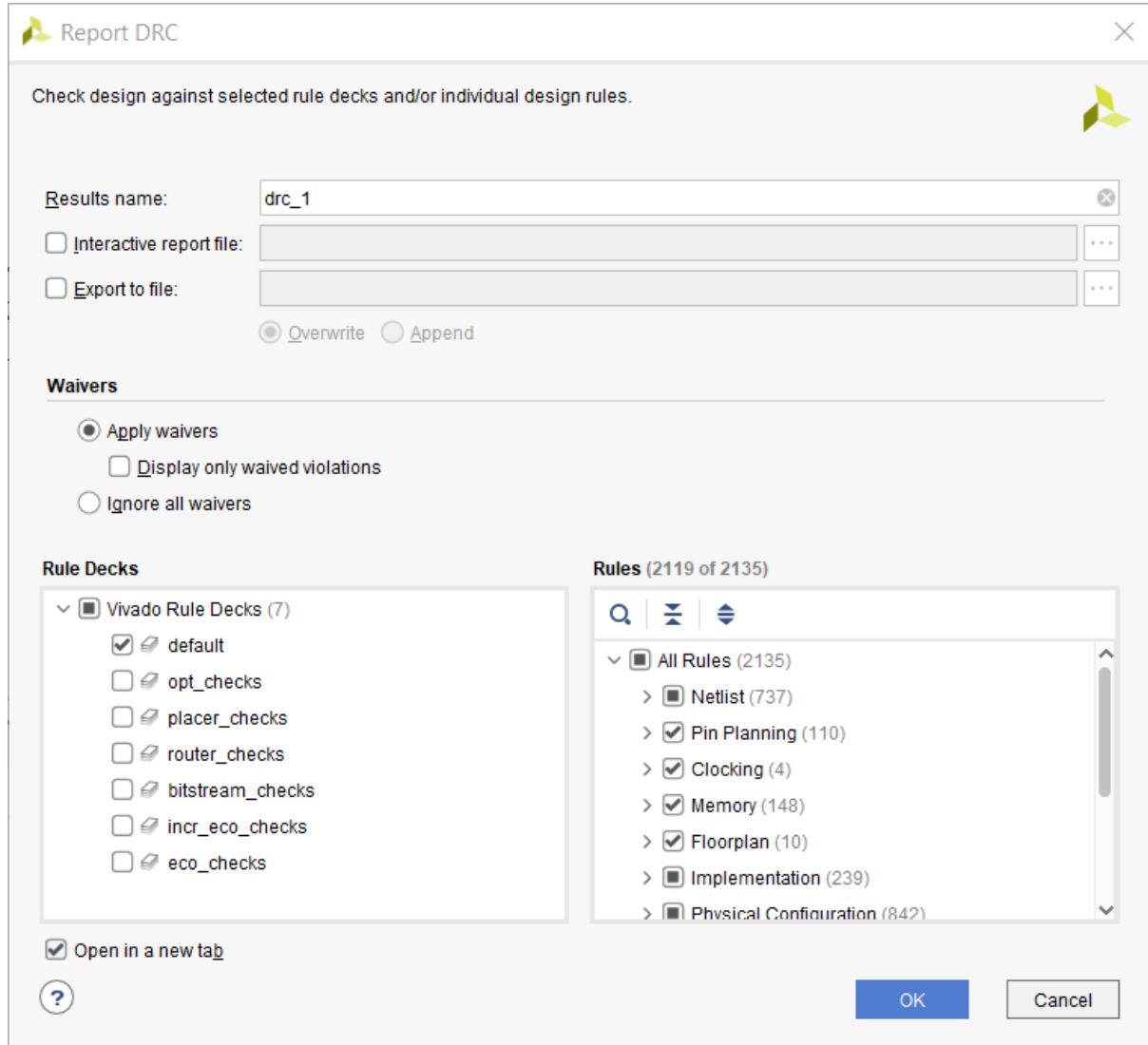
The following warnings show that duplicate waivers were not added to the existing waivers. Only waivers that are exact duplicates of existing waivers are rejected.

```
WARNING: [Vivado_Tcl 4-935] Waiver ID 'CDC-10' is a duplicate and will  
not be added again.  
WARNING: [Vivado_Tcl 4-935] Waiver ID 'CDC-11' is a duplicate and will  
not be added again.  
WARNING: [Vivado_Tcl 4-935] Waiver ID 'CDC-11' is a duplicate and will  
not be added again.  
WARNING: [Vivado_Tcl 4-935] Waiver ID 'CDC-14' is a duplicate and will  
not be added again.
```

Step 10: Waiving Multiple DRC Violations

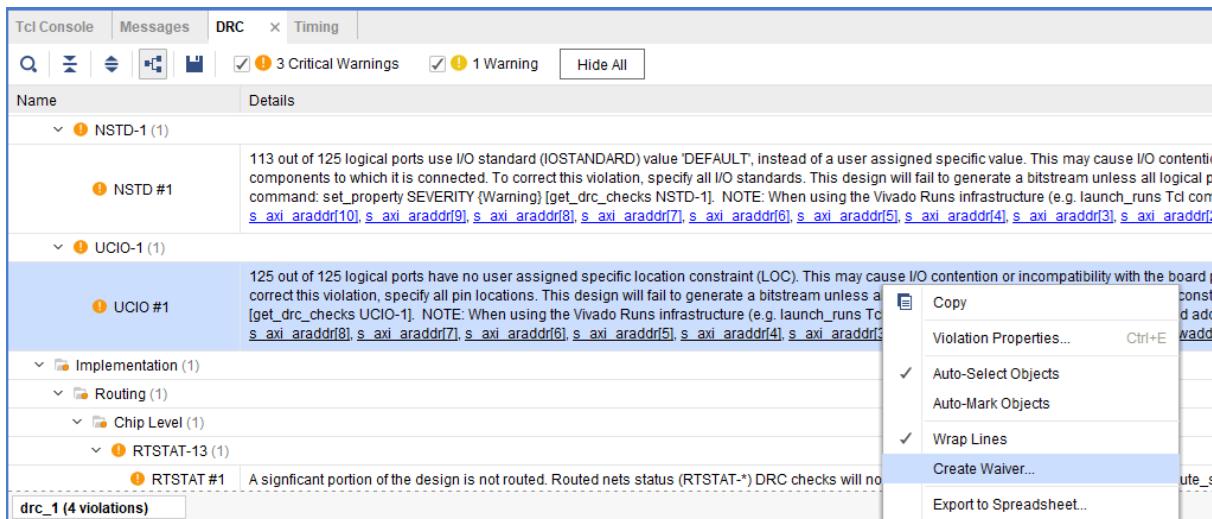
In this step, you waive multiple DRC violations simultaneously.

1. Select **Reports**→**Report DRC**.
2. In the Report DRC dialog box, leave all settings at their default, and click **OK**.

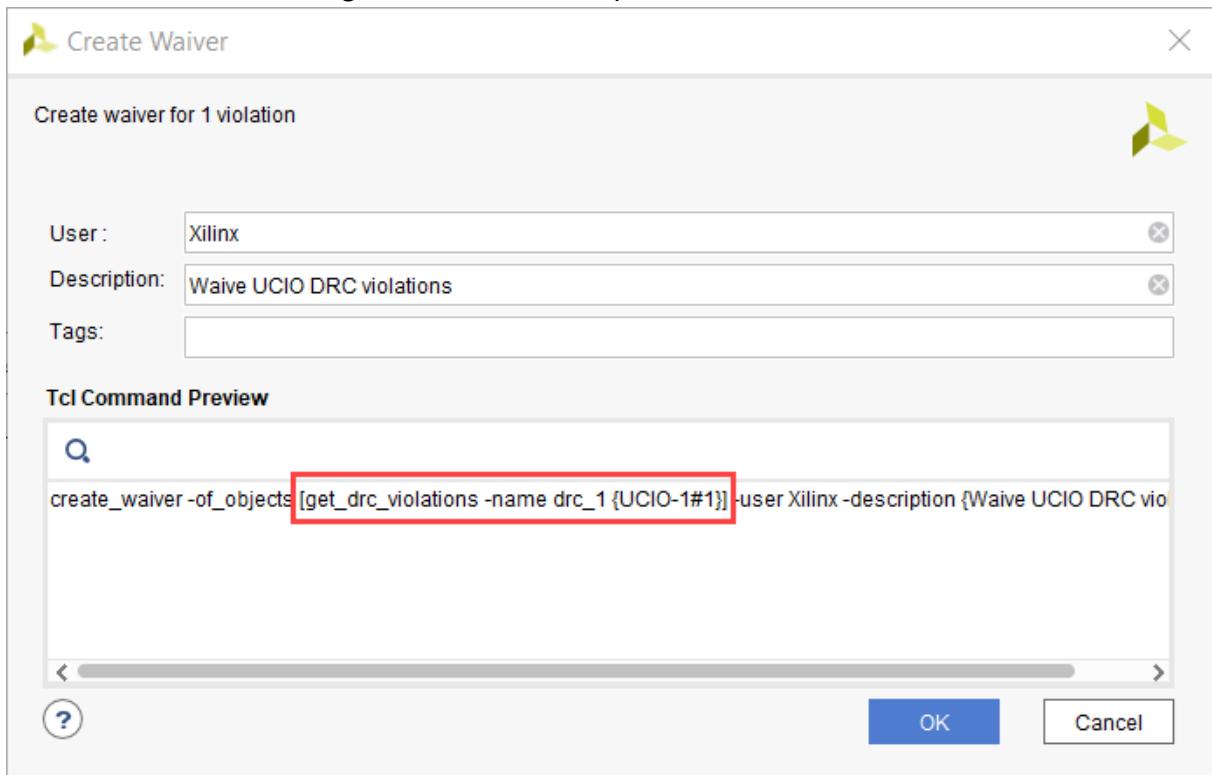


3. In the DRC Report, right-click **UCIO#1**, and select **Create Waiver** to create a waiver for the UCIO-1 violations.

Note: The UCIO#1 violation combines 125 individual violations into a single violation. Similarly, the NSTD#1 violation covers 113 ports.



4. In the Create Waiver dialog box, look at the output in Tcl Command Preview, and click **OK**.



5. To generate the `drc_waivers.xdc` file and verify that the waiver is waiving all 125 objects, enter:

```
write_waivers -type DRC drc_waivers.xdc
```

6. In the XDC file, look at the expanded port list, and notice that some of the strings from the violations message were converted to wildcards (*).

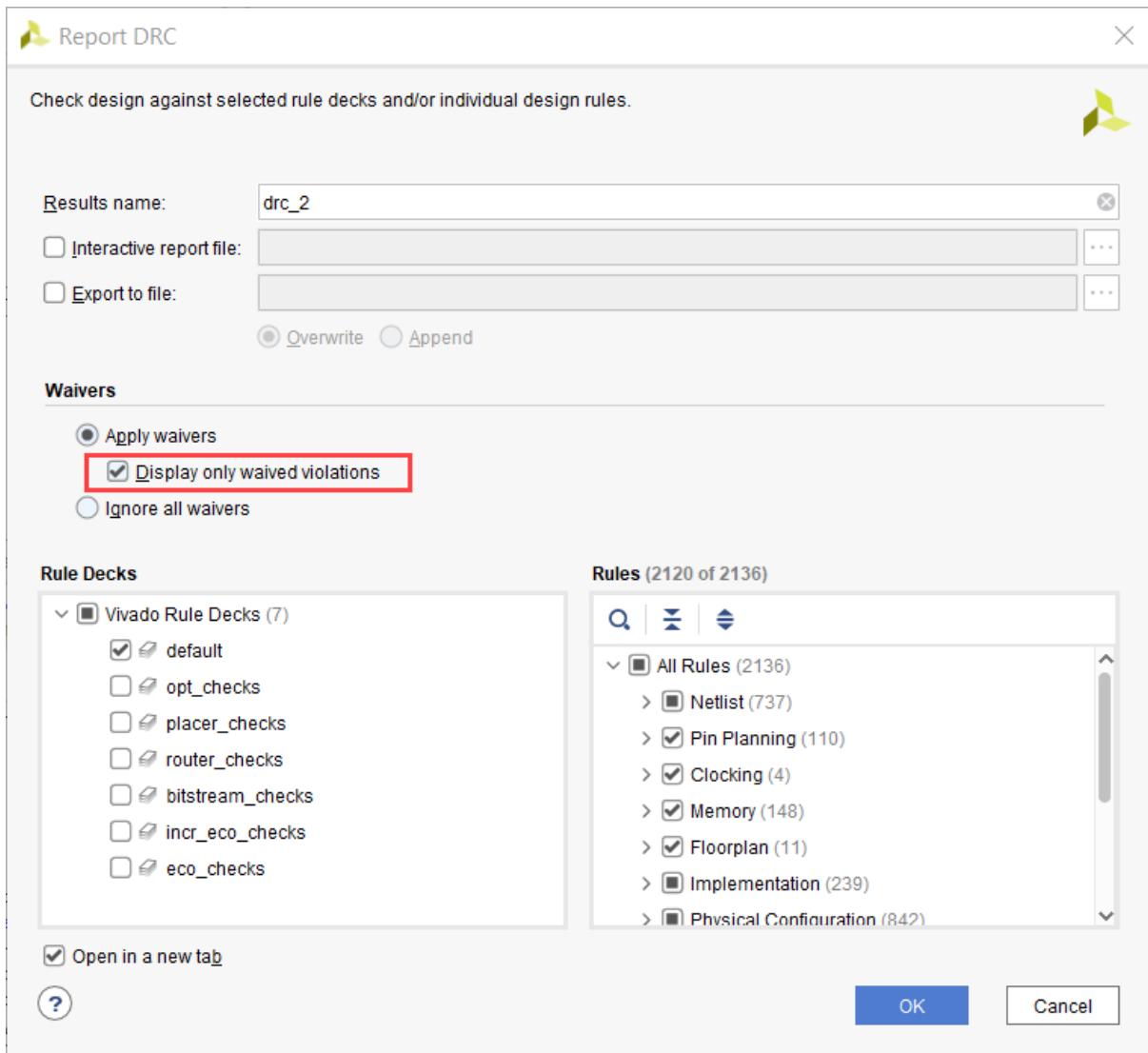
Strings are automatically converted to wildcards for UCIO-1, NSTD-1, TIMING-15, and TIMING-16 type violations. For UCIO-1, the numbers of objects in the violations are replaced with wildcards, because the numbers of elements are not meaningful.

7. To delete the DRC waiver and rewrite the waiver using wildcards to target a subset of the ports objects, enter:

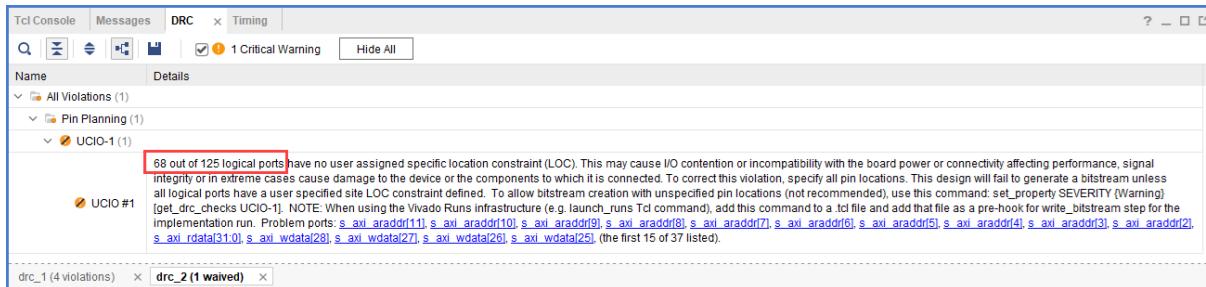
```
delete_waivers [get_waivers -type drc]
create_waiver -type DRC -id {UCIO-1} -user "Xilinx" -desc "Waive
selected UCIO violations" -objects [get_ports { s_axi_rdata[*]
s_axi_wdata[*] s_axi_araddr[*] } ] -strings { "*" } -strings { "*" }
```

Note: This command only covers a subset of the original 125 objects.

8. Select **Reports** → **Report DRC** to rerun Report DRC.
9. In the Report DRC dialog box, select **Display only waived violations** to report only waived violations, and click **OK**.



In the DRC Report, verify that only 68 violations are waived out of 125.



IMPORTANT! You cannot waive **READONLY** or **NODISABLE** violations. For example, if you enter:

```
create_waiver -type DRC -id RTSTAT-1 -description "Waive RTSTAT-1"
```

The Vivado tools issue the following error:

```
ERROR: [Vivado_Tcl 4-934] Waiver ID 'RTSTAT-1' is READONLY or  
NODISABLE and cannot be waived.  
These Factory designations specify that a check is required and may  
not be overridden by user action.
```

Step 11: Generating a Summary Report for Waived Violations

This step covers how to use the `report_waivers` Tcl command to generate a summary report for CDC, DRC, and methodology waivers.



IMPORTANT! Before running the `report_waivers` command, you must rerun Report CDC, Report DRC, or Report Methodology to ensure that added or removed waivers are included in the statistics reported by `report_waivers`.

1. To rerun Report CDC, enter:

```
report_cdc
```

2. To rerun Report DRC, enter:

```
report_drc
```

Note: You do not need to rerun Report Methodology, because no methodology waivers were set.

3. To create a summary report, enter:

```
report_waivers
```

By default, `report_waivers` reports only waived violations. The following figure shows the UCIO-1, CDC-10, CDC-11, and CDC-14 rules, which have defined waivers.

Table Of Contents																																						
1. REPORT SUMMARY	2. REPORT DETAILS (DRC)	3. REPORT DETAILS (METHODODOLOGY: no waivers)	4. REPORT DETAILS (CDC)																																			
1. REPORT SUMMARY																																						
<table border="1"> <thead> <tr> <th>Waiver Type</th> <th>Total Vios</th> <th>Remaining Vios</th> <th>Waived Vios</th> <th>Used Waivers</th> <th>Set Waivers</th> <th></th> </tr> </thead> <tbody> <tr> <td>DRC</td> <td>240</td> <td>172</td> <td>68</td> <td>1</td> <td>1</td> <td></td> </tr> <tr> <td>METHODODOLOGY</td> <td>0</td> <td>0</td> <td>0</td> <td>0</td> <td>0</td> <td></td> </tr> <tr> <td>CDC</td> <td>957</td> <td>944</td> <td>13</td> <td>4</td> <td>4</td> <td></td> </tr> </tbody> </table>							Waiver Type	Total Vios	Remaining Vios	Waived Vios	Used Waivers	Set Waivers		DRC	240	172	68	1	1		METHODODOLOGY	0	0	0	0	0		CDC	957	944	13	4	4					
Waiver Type	Total Vios	Remaining Vios	Waived Vios	Used Waivers	Set Waivers																																	
DRC	240	172	68	1	1																																	
METHODODOLOGY	0	0	0	0	0																																	
CDC	957	944	13	4	4																																	
Note: This report is based on the most recent report_drc/report_methodology/report_cdc runs.																																						
2. REPORT DETAILS (DRC)																																						
<table border="1"> <thead> <tr> <th>Rule</th> <th>Severity</th> <th>Description</th> <th>Total Vios</th> <th>Remaining Vios</th> <th>Waived Vios</th> <th>Used Waivers</th> <th>Set Waivers</th> </tr> </thead> <tbody> <tr> <td>UCIO-1*</td> <td>Critical Warning</td> <td>Unconstrained Logical Port</td> <td>125</td> <td>57</td> <td>68</td> <td>1</td> <td>1</td> </tr> </tbody> </table>							Rule	Severity	Description	Total Vios	Remaining Vios	Waived Vios	Used Waivers	Set Waivers	UCIO-1*	Critical Warning	Unconstrained Logical Port	125	57	68	1	1																
Rule	Severity	Description	Total Vios	Remaining Vios	Waived Vios	Used Waivers	Set Waivers																															
UCIO-1*	Critical Warning	Unconstrained Logical Port	125	57	68	1	1																															
4. REPORT DETAILS (CDC)																																						
<table border="1"> <thead> <tr> <th>Rule</th> <th>Severity</th> <th>Description</th> <th>Total Vios</th> <th>Remaining Vios</th> <th>Waived Vios</th> <th>Used Waivers</th> <th>Set Waivers</th> </tr> </thead> <tbody> <tr> <td>CDC-10</td> <td>Critical</td> <td>Combinational logic detected before a synchronizer</td> <td>187</td> <td>186</td> <td>1</td> <td>1</td> <td>1</td> </tr> <tr> <td>CDC-11</td> <td>Critical</td> <td>Fan-out from launch flop to destination clock</td> <td>2</td> <td>0</td> <td>2</td> <td>2</td> <td>2</td> </tr> <tr> <td>CDC-14</td> <td>Critical</td> <td>Multi-bit CDC path on a non-FD primitive</td> <td>10</td> <td>0</td> <td>10</td> <td>1</td> <td>1</td> </tr> </tbody> </table>							Rule	Severity	Description	Total Vios	Remaining Vios	Waived Vios	Used Waivers	Set Waivers	CDC-10	Critical	Combinational logic detected before a synchronizer	187	186	1	1	1	CDC-11	Critical	Fan-out from launch flop to destination clock	2	0	2	2	2	CDC-14	Critical	Multi-bit CDC path on a non-FD primitive	10	0	10	1	1
Rule	Severity	Description	Total Vios	Remaining Vios	Waived Vios	Used Waivers	Set Waivers																															
CDC-10	Critical	Combinational logic detected before a synchronizer	187	186	1	1	1																															
CDC-11	Critical	Fan-out from launch flop to destination clock	2	0	2	2	2																															
CDC-14	Critical	Multi-bit CDC path on a non-FD primitive	10	0	10	1	1																															
Note: Any 'Rule' which is flagged by '*' is an aggregating message and its counts are based on the number of objects represented, rather than the number of messages.																																						

Note the number of waived objects and total violations:

- The aggregating DRCs are reported as 1 violation per object inside the violation. Because there are 113 objects in NSTD-1, 125 objects in UCIO-1, 1 in RTDAT-13, and 1 in AVAL-326, a total number of 240 DRC violations are reported in the Summary table.
- The Report Summary table reports all of the violations.
- The Report Details tables only report the check IDs that have one or more waivers.

4. To generate detailed tables with all of the rules, including rules with no waivers, enter:

```
report_waivers -show_msgs_with_no_waivers
```

The following figure shows the report with all DRC and CDC rules reported in the Report Details.

Table Of Contents						
1. REPORT SUMMARY						
Waiver Type	Total Vios	Remaining Vios	Waived Vios	Used Waivers	Set Waivers	
DRC	240	172	68	1	1	
METHODOLOGY	0	0	0	0	0	
CDC	957	944	13	4	4	
Note: This report is based on the most recent report_drc/report_methodology/report_cdc runs.						
2. REPORT DETAILS (DRC)						
Rule	Severity	Description	Total Vios	Remaining Vios	Waived Vios	Used Waivers
UCIO-1*	Critical Warning	Unconstrained Logical Port	125	57	68	1
AVAL-326	Critical Warning	Hard_block_must_have_LOC	1	1	0	0
NSTD-1*	Critical Warning	Unspecified I/O Standard	113	113	0	0
RTSTAT-13	Critical Warning	Insufficient Routing	1	1	0	0
4. REPORT DETAILS (CDC)						
Rule	Severity	Description	Total Vios	Remaining Vios	Waived Vios	Used Waivers
CDC-10	Critical	Combinational logic detected before a synchronizer	187	186	1	1
CDC-11	Critical	Fan-out from launch flop to destination clock	2	0	2	2
CDC-14	Critical	Multi-bit CDC path on a non-FD primitive	10	0	10	1
CDC-1	Critical	1-bit unknown CDC circuitry	536	536	0	0
CDC-3	Info	1-bit synchronized with ASYNC_REG property	9	9	0	0
CDC-4	Critical	Multi-bit unknown CDC circuitry	28	28	0	0
CDC-9	Info	Asynchronous reset synchronized with ASYNC_REG property	5	5	0	0
CDC-13	Critical	1-bit CDC path on a non-FD primitive	170	170	0	0
CDC-15	Warning	Clock enable controlled CDC structure detected	10	10	0	0
Note: Any 'Rule' which is flagged by '*' is an aggregating message and its counts are based on the number of objects represented, rather than the number of messages.						

5. To run Report Methodology, enter:

```
report_methodology
```

6. To generate detailed tables with all of the rules, including rules with no waivers, enter:

```
report_waivers -show_msgs_with_no_waivers
```

The exact statistics are reported, as shown in the following figure.

Note: This figure does not include the Report Details (CDC) section.

1. REPORT SUMMARY						
Waiver Type	Total Vios	Remaining Vios	Waived Vios	Used Waivers	Set Waivers	
DRC	240	172	68	1	1	
METHODOLOGY	158	158	0	0	0	
CDC	957	944	13	4	4	
Note: This report is based on the most recent report_drc/report_methodology/report_cdc runs.						
2. REPORT DETAILS (DRC)						
Rule	Severity	Description	Total Vios	Remaining Vios	Waived Vios	Used Waivers
UCIO-1*	Critical Warning	Unconstrained Logical Port	125	57	68	1
AVAL-326	Critical Warning	Hard_block_must_have_LOC	1	1	0	0
NSTD-1*	Critical Warning	Unspecified I/O Standard	113	113	0	0
RTSTAT-13	Critical Warning	Insufficient Routing	1	1	0	0
3. REPORT DETAILS (METHODOLOGY)						
Rule	Severity	Description	Total Vios	Remaining Vios	Waived Vios	Used Waivers
AVAL-324	Critical Warning	Hard_block_needs_LOC	1	1	0	0
LUTAR-1*	Warning	LUT drives sync reset alert	40	40	0	0
TIMING-9	Warning	Unknown CDC Logic	1	1	0	0
TIMING-10	Warning	Missing property on synchronizer	1	1	0	0
TIMING-18*	Warning	Missing input or output delay	115	115	0	0

Step 12: Using Waiver Commands

In this step, you run additional commands related to the waivers.

1. To return a collection of CDC waiver objects, enter:

```
get_waivers -type cdc
```

The following CDC waivers are returned:

```
CDC-10#1 CDC-11#1 CDC-11#2 CDC-14#1
```

2. To filter the list of waivers to only return CDC-14 waivers, enter:

```
get_waivers -filter {ID == CDC-14}  
CDC-14#1
```

3. To report all of the properties on a CDC waiver object, enter:

```
report_property [lindex [get_waivers -type cdc] end]
```

The following properties are returned:

Property	Type	Read-only	Value
CLASS	string	true	cdc_waiver
DESCRIPTION	string	false	No more CDC-14!
ID	string	true	CDC-14
INDEX	string	true	1
IS_SCOPED	bool	true	0
NAME	string	true	CDC-14#1
OBJECT_COUNTS	string	true	pins:2
SCOPE	string	true	
TAGS	string	false	
TIME	string	true	<timestamp>
TYPE	string	true	CDC
USED_CNT	string	true	10
USER	string	true	Xilinx

Note: You cannot retrieve the design objects attached to a waiver object.

4. To delete all of the previously created CDC-14 waivers, enter:

```
delete_waivers [get_waivers -filter {ID == CDC-14}]
```

Note: After a waiver object is deleted, the waiver no longer applies and the violations that it waived are reported again.

5. To delete all of the remaining CDC waivers, enter:

```
delete_waivers [get_waivers -type cdc]
```

Summary

In this lab, you accomplished the following:

- Waived CDC and DRC violations
- Generated reports for waived violations
- Exported waivers
- Used waiver commands

Lab 2

Using Report QoR Suggestions and Report QoR Assessment for Timing Closure

Introduction

The `report_qor_suggestions` (RQS) command enables the Vivado® Design Suite tools to analyze a design and provide automated solutions for enhancing quality of results (QoR). The command can be run on an open design after synthesis or after any stage in the implementation flow. The RQS command evaluates the design and suggests fixes or improvements.

Recommendations from RQS can take the following forms:

- RQS objects. These objects can add switches to a given command, or properties to a given design object. They also allow you to add implementation strategies customized for the design using machine learning algorithms.
- Text recommendations that require user intervention.

Using the same analysis techniques, the `report_qor_assessment` (RQA) command assesses the likelihood that a design will meet its timing requirements and provides a quick summary of design metrics.

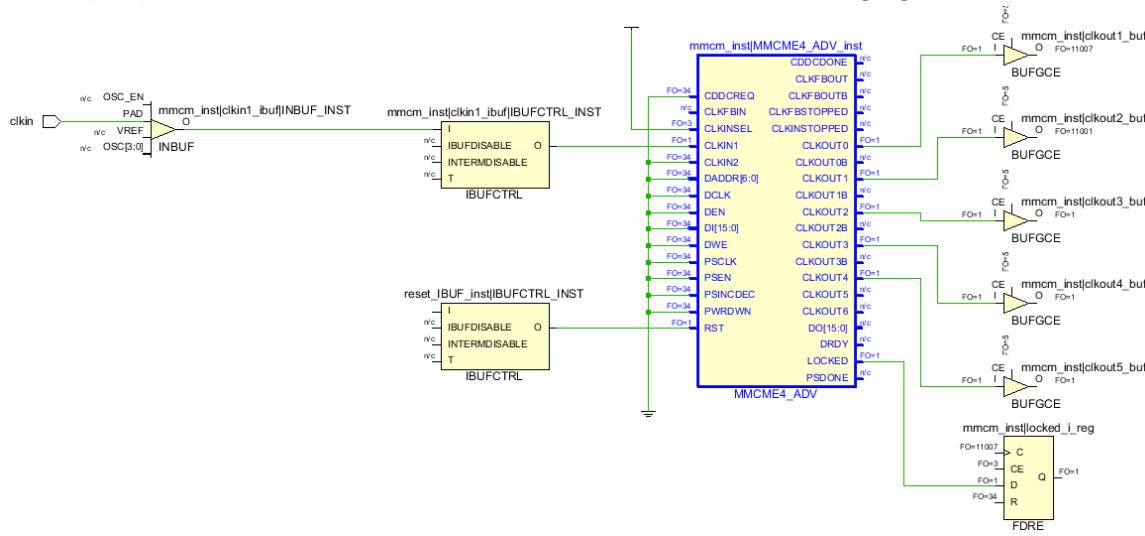
This lab covers how to do the following:

- Analyze the QoR assessment report.
- Analyze the QoR suggestion report.
- Export suggestions to an RQS file.
- Add an RQS file to synthesis and implementation runs.
- Accumulate suggestions in an RQS file by generating suggestions at different implementation stages or on different runs.
- Add the AUTO_RQS property to a project-based design run and automatically accrue suggestions.

Step 1: Understanding the Design

This lab uses an architected design to demonstrate some of the features of RQS. Suggestions are triggered by the design of the RTL and the placement of blocks using floorplanning. The design contains the following modules:

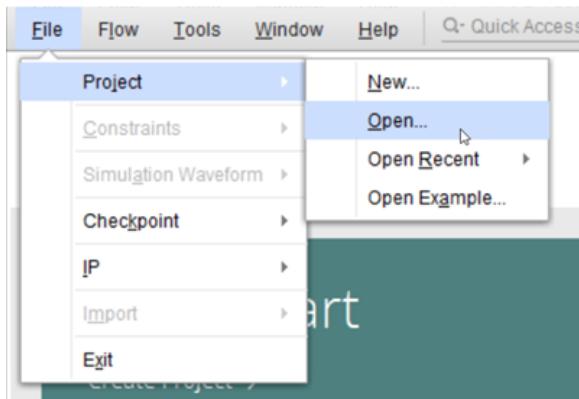
- **Clocking Module:** The main clocking circuit for the design resides in `clocking_module.vhd`. For simplicity, RST is tied to GND. LOCKED is registered and tied to an output port. The structure of this block is shown in the following figure.



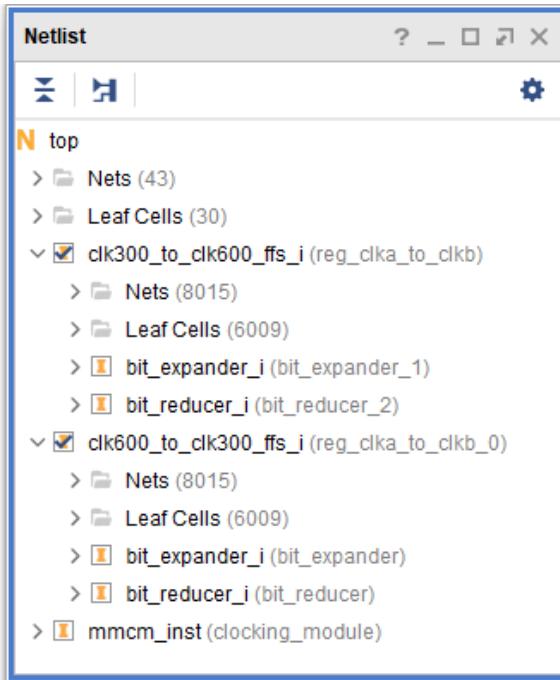
- **Reg CLKA to CLKB Module:** This module contains a synchronous CDC for a large bus. It registers input data using CLKA and then passes it to a register on the CLKB domain to be passed to the output. Registering large buses on different related clock domains can impact hold slack (WHS/THS) and setup slack (WNS/TNS).
- **Bit Expander and Bit Reducer Modules:** These modules enable the expansion and contraction of internal data widths so that the design does not run out of I/Os. The modules take an arbitrary data width and expand or contract it to or from a desired size. The contraction logic creates many logic levels.

The following steps cover opening the project and examining the placement of the floor-planned modules.

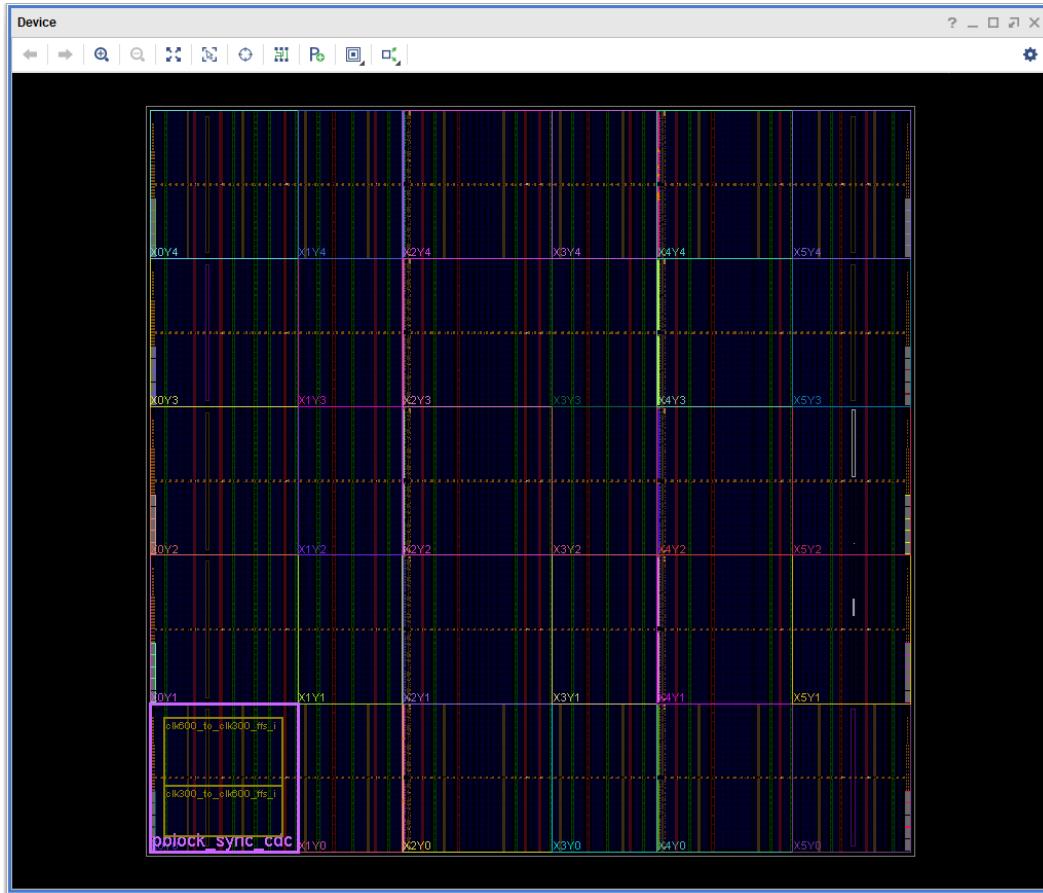
1. In the Vivado Design Suite, go to **File** → **Project** → **Open** and select the project located in `<extract_Dir>/Lab2/project_2`.



2. In the Flow Navigator, click **Run Synthesis** and wait for synthesis to complete.
3. In the Flow Navigator, click **Open Synthesized Design**.
4. In the Netlist view, look at the hierarchy.



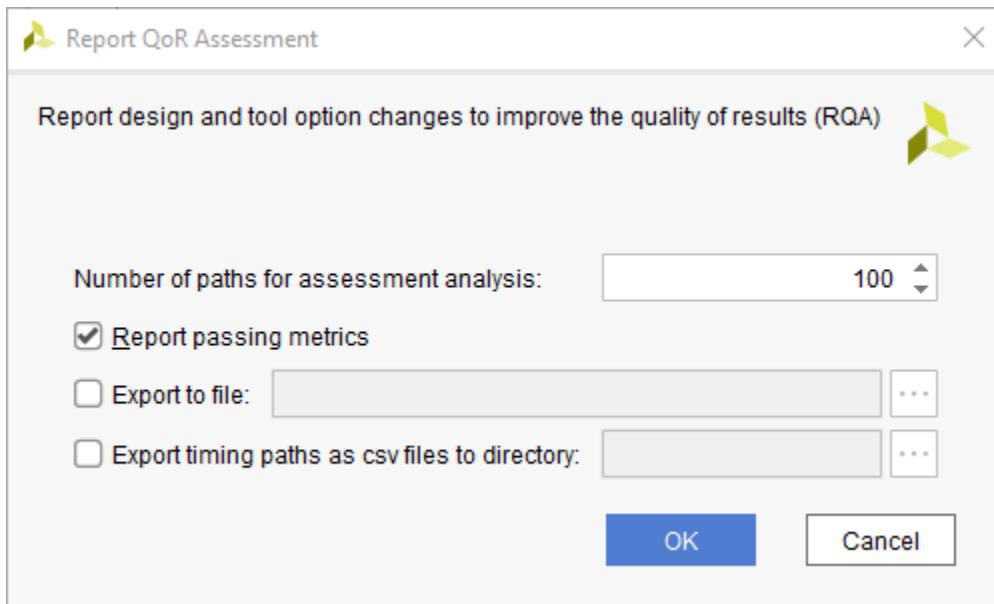
5. In Device view, look at the pblock. This has been added to control placement of the `reg_clka_to_clkb` modules and force a poor clock skew.



Step 2: Running Report QoR Assessment

In this step, you will run the `report_qor_assessment` command on the open design. This step can be run after any stage of the implementation process: `synth_design`, `opt_design`, `place_design`, `phys_opt_design`, and `route_design`. It provides an assessment score that details the likelihood of the design closing timing, as well as a quick first analysis returning items that need to be further investigated.

1. With the design open, click **Reports** → **Report QoR Assessment....**
2. In the Report QoR Assessment dialog box, select **Report Passing Metrics** and click **OK**.



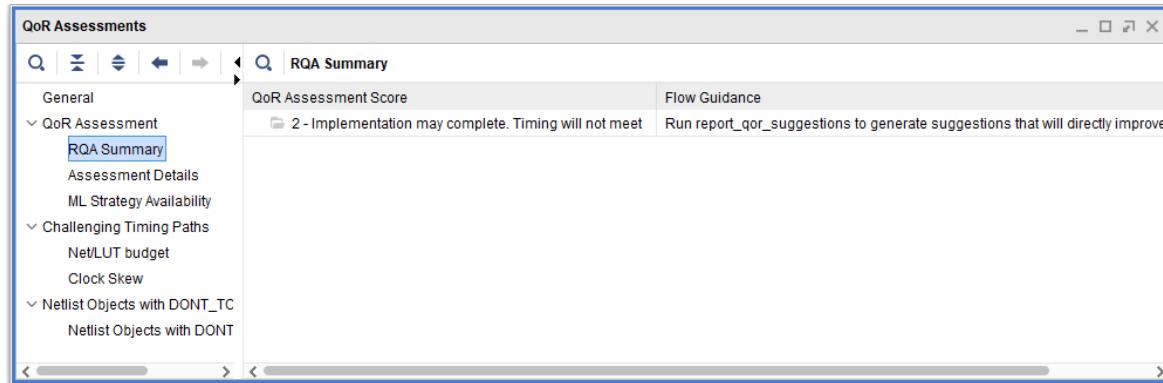
The equivalent Tcl command is as follows:

```
report_qor_assessment -exclude_methodology_checks -name qor_assessments -  
max_paths 100 -full_assessment_details
```

This opens up the assessment report, with three main sections providing the following:

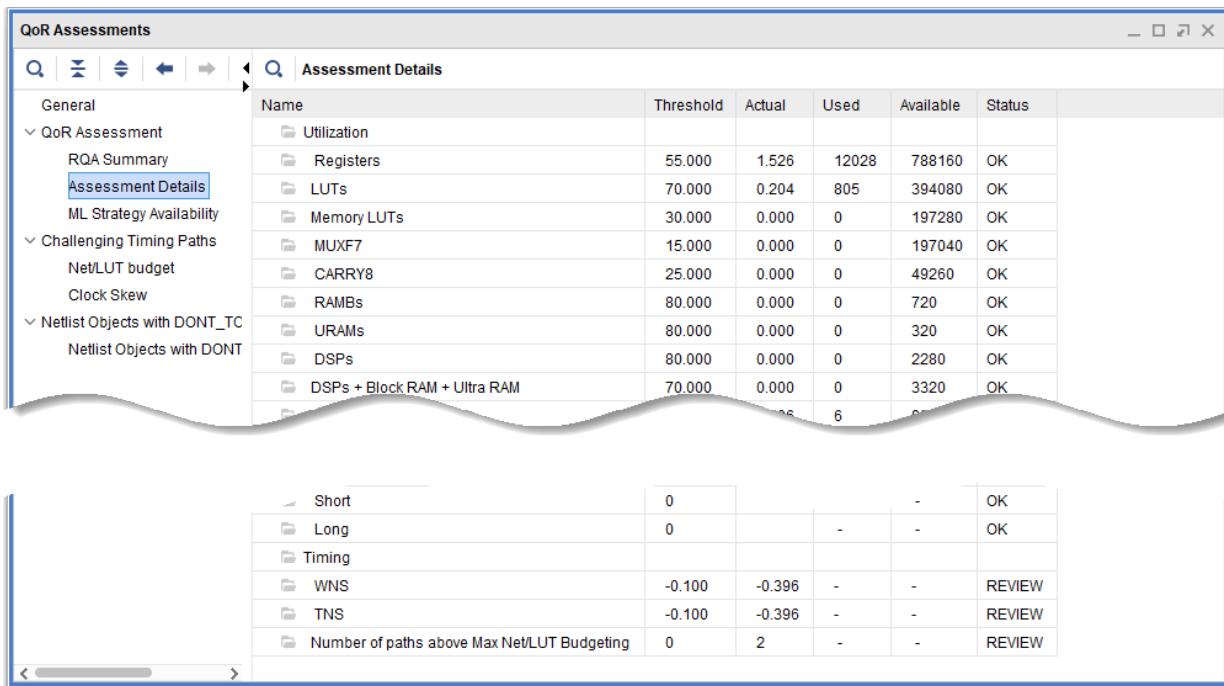
- Assessment summary
- Difficult timing paths requiring investigation
- Objects with DONT_TOUCH properties that prevent tool optimizations

3. In the first section of the report, select **RQA Summary**.



This section has the score marked as "2 - Implementation may complete. Timing will not meet". It also recommends running `report_qorSuggestions`. The RQA command understands when suggestions are available and can offer the guidance to check them.

4. Select **Assessment Details**.



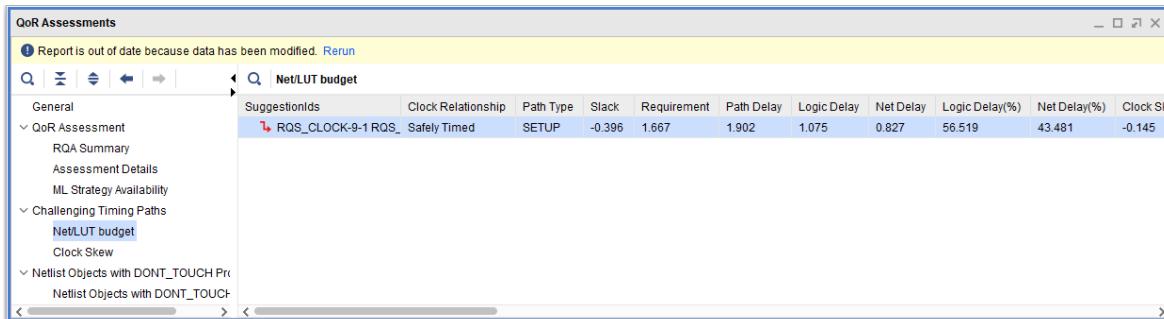
Name	Threshold	Actual	Used	Available	Status
Utilization					
Registers	55.000	1.526	12028	788160	OK
LUTs	70.000	0.204	805	394080	OK
Memory LUTs	30.000	0.000	0	197280	OK
MUXF7	15.000	0.000	0	197040	OK
CARRY8	25.000	0.000	0	49260	OK
RAMBs	80.000	0.000	0	720	OK
URAMs	80.000	0.000	0	320	OK
DSPs	80.000	0.000	0	2280	OK
DSPs + Block RAM + Ultra RAM	70.000	0.000	0	3320	OK

Path Type	Value	Requirement	Path Delay	Logic Delay	Net Delay	Logic Delay(%)	Net Delay(%)	Clock Skew
Short	0							OK
Long	0							OK
Timing								
WNS		-0.100	-0.396	-	-			REVIEW
TNS		-0.100	-0.396	-	-			REVIEW
Number of paths above Max Net/LUT Budgeting	0	2	-	-	-			REVIEW

This section lists the items that have led to the assessment score of 2. Because the option to **Report passing metrics** was selected, items marked as OK are shown. Typically, only items marked REVIEW are shown. In this example, you can see WNS and TNS metrics marked for review, as well as paths above their net/LUT budget.

Timing paths usually offer the values that are achievable in the best case scenarios. Not all paths can achieve these values. Net and LUT budget checks substitute net or LUT values with more typical values, and also penalize paths that are more challenging due to a netlist profile or a resource being less sparse. The result is two checks performed under this item. Look at all of these paths before continuing.

5. Select **Net/LUT budget** under Challenging Timing Paths.



SuggestionIds	Clock Relationship	Path Type	Slack	Requirement	Path Delay	Logic Delay	Net Delay	Logic Delay(%)	Net Delay(%)	Clock Skew
RQS_CLOCK-9-1 RQS_	Safely Timed	SETUP	-0.396	1.667	1.902	1.075	0.827	56.519	43.481	-0.145

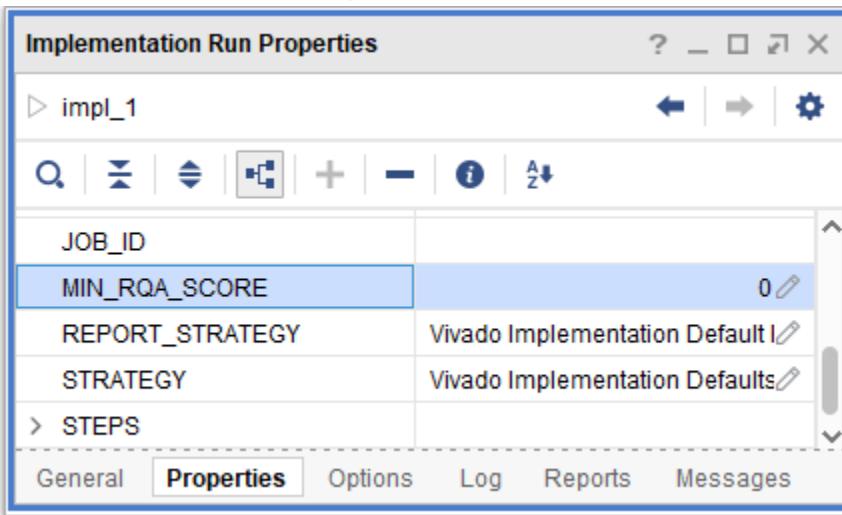
Scroll across the screen to see all the path characteristics reported. The following items are of particular interest:

- **SuggestionIds:** These IDs correspond to suggestions that, if triggered, impact this path.

- **Net Check Slack:** This is the slack when the path is substituted with higher net delay values.
- **LUT Check Slack:** This is the slack when LUTs are substituted with higher LUT delay values.

6. Double-click on the path to bring up the Timing Path report. You can also press **F4** to show a schematic. All of these items use standard Vivado cross probing. At this stage, you can also explore other items in the report.

7. In the Design Runs window, select **impl_1**, expand the **Implementation Run Properties** window, and select the **Properties** tab. Locate the **MIN_RQA_SCORE** property.



The **MIN_RQA_SCORE** property can terminate the project implementation run when it is greater than the RQA score. This property defaults to zero, and does not impact unless it is set to another value that is ≥ 2 . The RQA report must also be run. This can be done by performing the following steps:

- Select **Timing Closure Report Strategy**.
- Add **report_qor_assessment** to custom report strategies.
- Add **report_qor_assessment** to a Tcl hook. This must execute the command in the run directory to be effective.

8. Edit the **MIN_RQA_SCORE** value to 3. Select the **Reports** tab and update the **Report Strategy** to **Timing Closure Reports**.

9. Launch the **impl_1** run.

After `opt_design` is complete, the `report_qor_assessment` command is run and the run should terminate. You should see the following in the Design Runs status column:

Name	Constraints	Status	Elapsed
✓ synth_1	OriginalConstraints	synth_design Complete!	00:02:01
● impl_1	OriginalConstraints	Flow terminated due to RQA score tolerance not met (step: opt_design)	00:01:20

The log file should have the following message:

```
INFO: [runtcl 7-1] RQA Score (opt_design): 2 (RQA score tolerance: 3)
ERROR: [runtcl 8-1] Flow terminated - RQA score threshold not met after
'opt_design' step
```

The following message appears when the flow successfully passes the check:

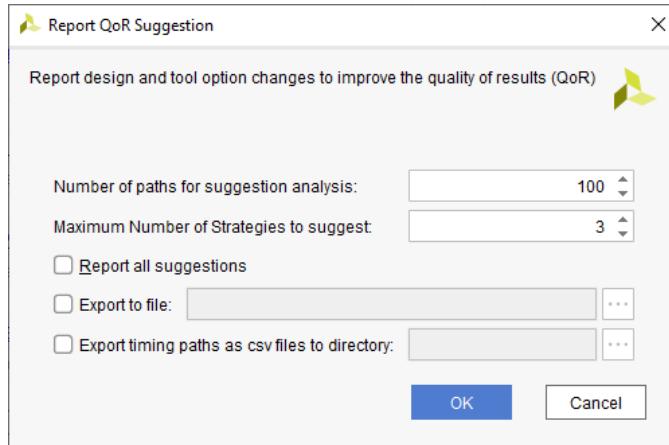
```
INFO: [runtcl 7-1] RQA Score (opt_design): 2 (RQA score tolerance: 2)
INFO: [runtcl 9-1] Flow continues - RQA score threshold met after
'opt_design' step
```

10. Update the **MIN_RQA_VALUE** to **1**. Revert the reports back to **Vivado Implementation Default Reports** and reset the run.

Step 3: Running Report QoR Suggestions

This step covers running the `report_qor_suggestions` command to generate a report. The command can be run on an open design at any stage of the implementation flow after synthesis. In project mode, this is typically after synthesis or implementation. In non-project mode, this can be after `synth_design`, `link_design`, `opt_design`, `place_design`, `phys_opt_design`, or `route_design`.

1. In the Vivado IDE, from the pull-down menus, click **Reports**→**Report QoR Suggestions...** to bring up the dialog box shown in the following figure.



2. Click **OK** to run the command. The report opens automatically in the integrated design environment (IDE). Due to the interactive nature of the report, only one instance of the report can be open at any time. The equivalent Tcl command is as follows:

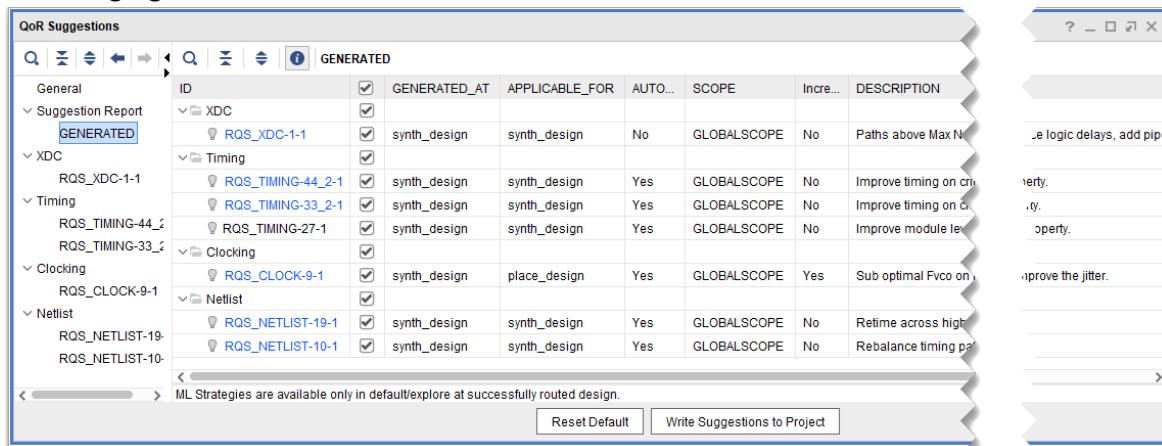
```
report_qor_suggestions -max_paths 100 -file rqs.rpt
```

Note: By default, the RQS command reports on the 100 worst failing paths per clock group. You can change the number of paths that RQS uses for the analysis of timing-critical paths by modifying the `-max_paths` switch. Increasing this number generates more suggestions, but on paths that are reducing in criticality.

Step 4: Understanding the Report

This step explains the different sections of the generated QoR Suggestions report. On the left of the report window, you can navigate to the different sections of the report; on the right, more information is provided.

1. In the Suggestion Report, select **GENERATED**. This brings up the report section shown in the following figure.



ID	GENERATED_AT	APPLICABLE_FOR	AUTOMATIC	SCOPE	INCREMENTAL	DESCRIPTION
RQS_XDC-1-1	synth_design	synth_design	No	GLOBALSCOPE	No	Paths above Max N...
RQS_TIMING-44-2-1	synth_design	synth_design	Yes	GLOBALSCOPE	No	Improve timing on cri...
RQS_TIMING-33-2-1	synth_design	synth_design	Yes	GLOBALSCOPE	No	Improve timing on cri...
RQS_TIMING-27-1	synth_design	synth_design	Yes	GLOBALSCOPE	No	Improve module level...
RQS_CLOCK-9-1	synth_design	place_design	Yes	GLOBALSCOPE	Yes	Sub optimal Fvco on...
RQS_NETLIST-19-1	synth_design	synth_design	Yes	GLOBALSCOPE	No	Retime across high...
RQS_NETLIST-10-1	synth_design	synth_design	Yes	GLOBALSCOPE	No	Rebalance timing pa...

The **GENERATED** section provides a list of all the suggestions that have been generated at this stage of the current run. Each suggestion has a description that details the reason for the suggestion. For each suggestion, the following information is also provided.

Table 1: RQS Summary Column Description

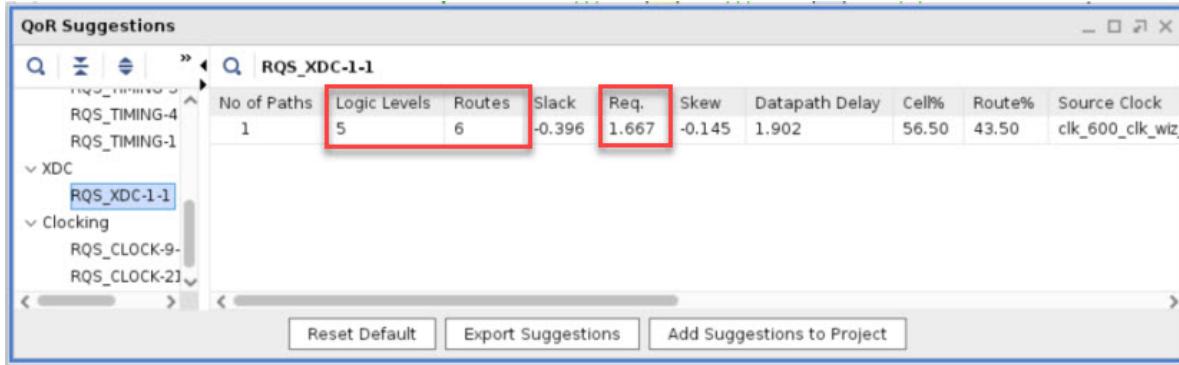
Item	Description	Comment
GENERATED_AT	This shows what stage of the design the suggestion was generated at. Typical values are <code>place_design</code> or <code>route_design</code> .	As you progress through the design stages, the decisions that the tool makes are based on the information available at the time. Information accuracy increases after placement and again after routing.
APPLICABLE_FOR	This stage must be rerun for the suggestion to take effect.	Most suggestions are executed at either <code>synth_design</code> or <code>place_design</code> .
AUTOMATIC	Indicates if the suggestion is executed automatically or if manual intervention is required.	Automatic suggestions either recommend a switch to the tool or a property to be added to a cell or net.
INCREMENTAL FRIENDLY	Indicates if the suggestion is optimized for the incremental flow.	Non-incremental friendly suggestions must be already present in the reference checkpoint. If you want to add non-incremental friendly suggestions, an updated reference checkpoint must be used.

Table 1: RQS Summary Column Description (cont'd)

Item	Description	Comment
SCOPE	Indicates the target synthesis run level. GLOBALSCOPE is top level. Otherwise, a sub-module is targeted.	Allows a single RQS file to be used on top-level and sub-module out of context (OOC) synthesis runs. Only applicable to synthesis suggestions.

The other sections of the report usually provide details about the individual suggestions that have been generated.

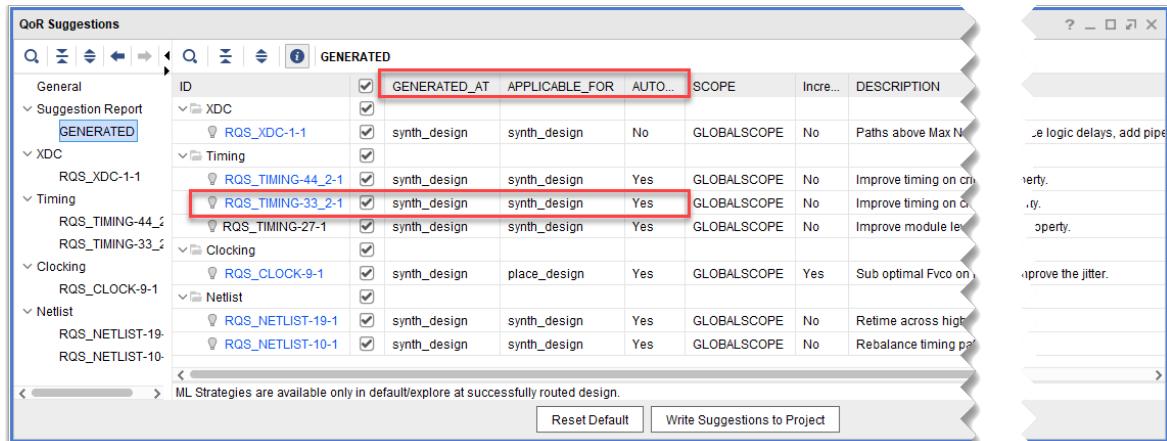
2. Click the **RQS_XDC-1-1** hyperlink. This takes you to the details section for this suggestion.



No of Paths	Logic Levels	Routes	Slack	Req.	Skew	Datapath Delay	Cell%	Route%	Source Clock
1	5	6	-0.396	1.667	-0.145	1.902	56.50	43.50	clk_600_clk_wiz

The suggestion description says that the timing constraint is too tight for the given path(s). The path has a large negative slack, which stands out in a timing report. Timing paths use net delays that are optimal. This gives the tools the correct order to place and route them. Close analysis shows this is a 600 MHz path with high logic levels. This path needs to be fixed.

3. Click the back arrow button (←) to go back to the GENERATED view.
4. In the GENERATED view, click the **RQS_TIMING-33_2-1** row. You can see this is an AUTOMATIC suggestion that is applicable for `synth_design` (see the APPLICABLE_FOR column). This indicates that you must rerun the `synth_design` command to make use of this suggestion.

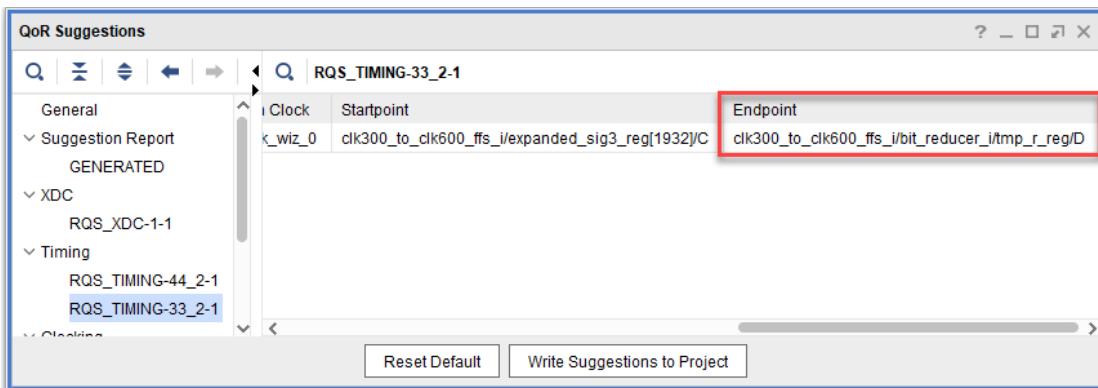


ID	GENERATED_AT	APPLICABLE_FOR	AUTO...	SCOPE	Incre...	DESCRIPTION	
RQS_XDC-1-1	✓	synth_design	synth_design	No	GLOBALSCOPE	No	Paths above Max N...
RQS_TIMING-44_2-1	✓	synth_design	synth_design	Yes	GLOBALSCOPE	No	Improve timing on cri...
RQS_TIMING-33_2-1	✓	synth_design	synth_design	Yes	GLOBALSCOPE	No	Improve timing on cri...
RQS_TIMING-27-1	✓	synth_design	synth_design	Yes	GLOBALSCOPE	No	Improve module level...
RQS_CLOCK-9-1	✓	synth_design	place_design	Yes	GLOBALSCOPE	Yes	Sub optimal Fvco on...
RQS_NETLIST-19-1	✓	synth_design	synth_design	Yes	GLOBALSCOPE	No	Retime across high...
RQS_NETLIST-10-1	✓	synth_design	synth_design	Yes	GLOBALSCOPE	No	Rebalance timing pat...

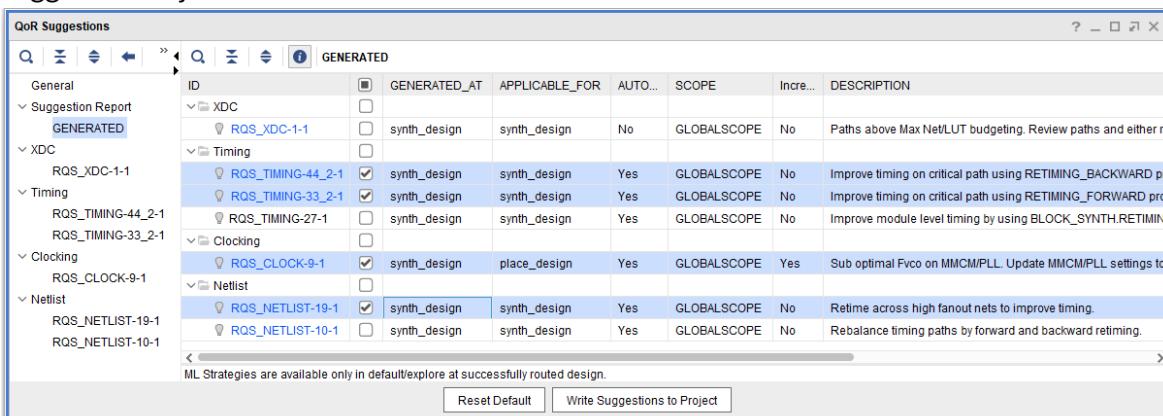
When you select the row in the GENERATED view, the suggestion object is selected and the properties can be seen in the QoR Suggestion Properties window. If you examine the command property, you can confirm that it generates a retiming forward property for synth_design.

QoR Suggestion Properties	
COMMAND	set_property retiming_forward 1 [get_cells {{clk300_tc}}
CLASS	qor_suggestion
CATEGORY	Timing
ID	RQS_TIMING-33_2
NAME	RQS_TIMING-33_2-1
INTERNAL_ID	RQS_TIMING-33_2-1636704388605679
DESCRIPTION	Improve timing on critical path using RETIMING_FORWARD
SUMMARY	Retiming of Flops in Critical paths
GENERATED_AT	synth_design
APPLICABLE_FOR	synth_design
ENABLED	<input type="checkbox"/>
APPLIED	<input checked="" type="checkbox"/>
AUTO	<input checked="" type="checkbox"/>
INCR_FRIENDLY	
RQA_SCORE	3
SWITCHES	None
TYPE	TCL
IS_USER_SUGGESTION	
FLOW_SUPPORT	Default
DISABLE_DONT_TOUCH_REQUIRED	
SUGGESTION_SOURCE	Current Run
FAILED_TO_APPLY	
TIMESTAMP	Fri Nov 12 12:06:28 2021
General Properties	

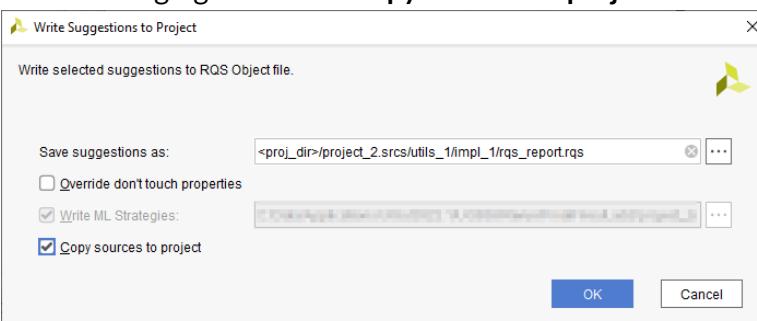
5. In the GENERATED view, click the **RQS_TIMING-33_2-1** ID to go to the details table for that suggestion. Careful examination of the Endpoint column confirms that this is the same path that was mentioned for RQS_XDC-1-1.



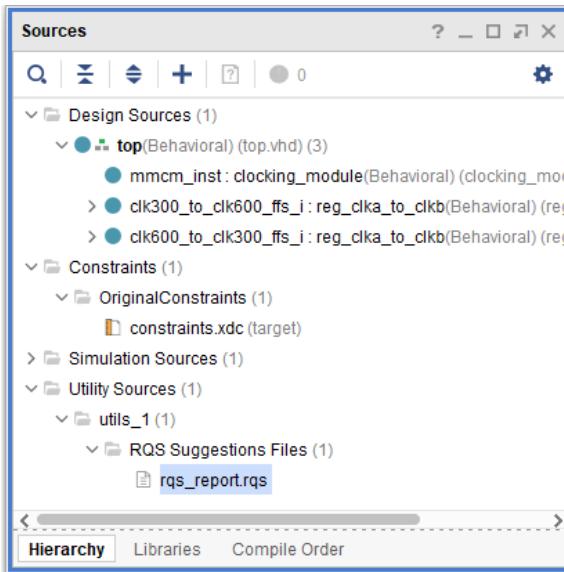
- In the GENERATED view, you can see the remaining suggestions. The RQS_CLOCK-9 suggestion is applicable for `place_design`. The other RQS_TIMING-44 suggestion is similar to the existing one.
- The RQS_TIMING-27-1 suggestion uses BLOCK_SYNTH properties at synthesis and applies to the entire module. You can determine the target cell by looking at the properties of the object as described in step 4. Uncheck the box for **RQS_TIMING-27-1**. This suggestion applies to the entire module and overlaps with the other suggestions. For clarity, it will not be applied.
- Only select the four suggestions highlighted in the following image to write to the RQS suggestion object file:



- Click **Write Suggestions to Project**. When the Write Suggestions to Project dialog box opens, ensure it is set to write to the `rqs_report.rqs` file in the `utils_1` directory, as shown in the following figure. Select **Copy sources to project**.



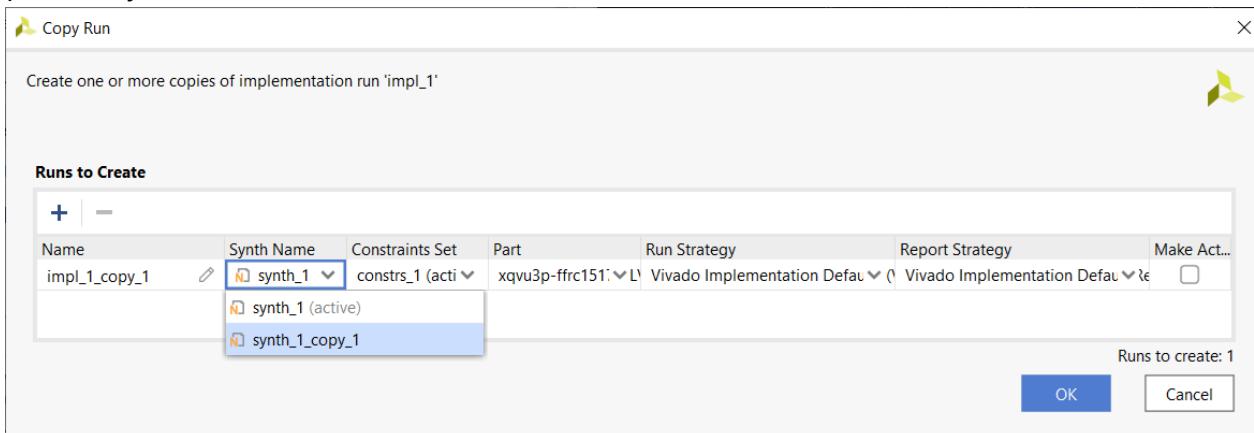
10. Examine the Sources window. The RQS file has been added to the `utils_1` fileset. This ensures that the file is captured using the `get_files` command and recognized in the next step when you add the file to a run.



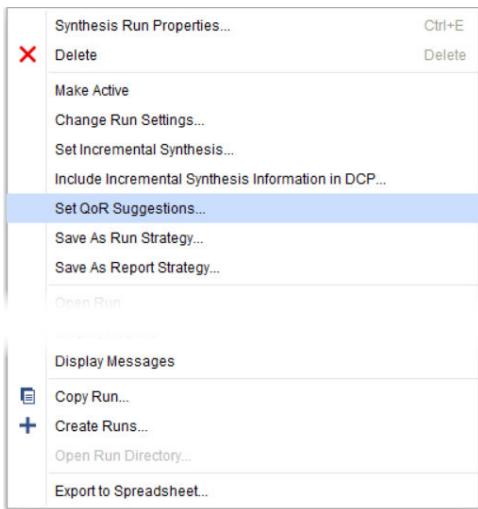
Step 5: Run with Suggestions

In this step, you will add a suggestion to a run and examine what happens when a suggestion is applied and how it is reported.

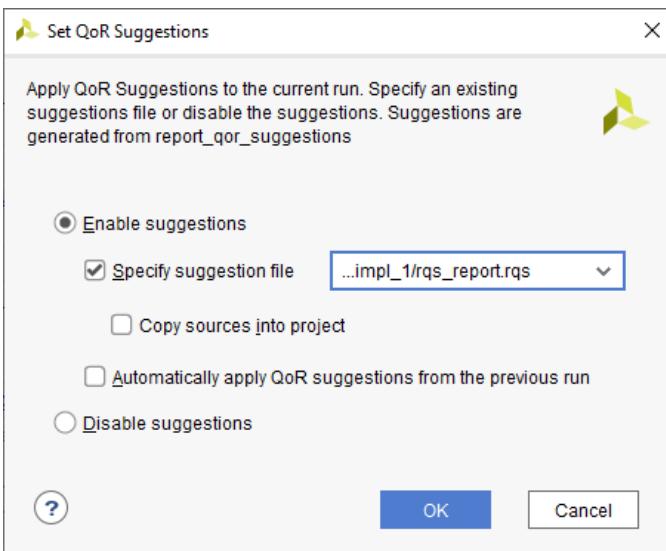
1. In the Design Runs window, right-click on the synthesis run, select **Copy Run**, and click **OK**. Do the same for the implementation run, but update the synthesis run name to the new one you have just created, and click **OK**.



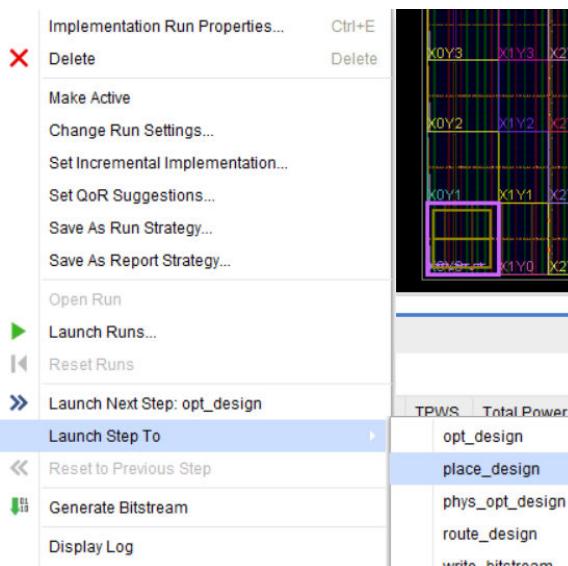
2. In the Design Runs window, right-click the new `synth_1_copy_1` run and select **Set QoR Suggestions**.



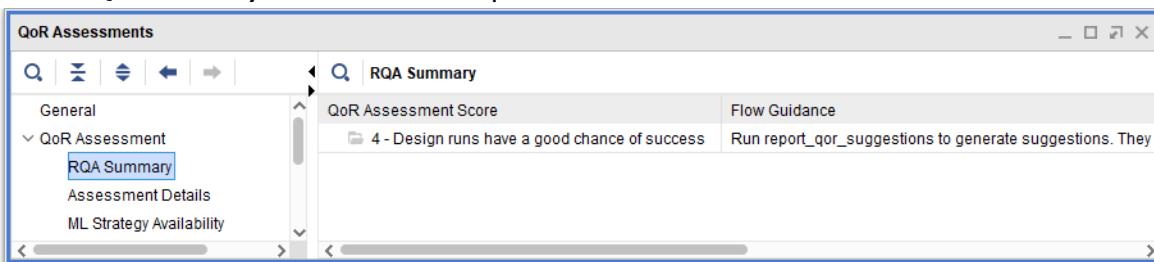
3. Specify the suggestion file as the RQS file added to the project from the previous step and click **OK**.



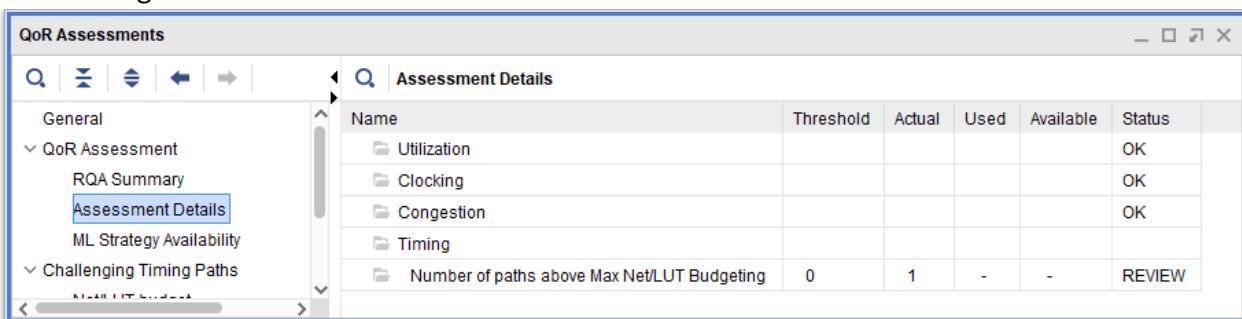
4. Repeat steps 2 and 3 for the implementation run, making sure to specify the new synthesis run as the parent run. Specify the same RQS file for each run.
5. In the Design Runs window, right-click **synth_1_copy** and select **Make Active**.
6. In the **Flow Navigator**, click **Run Synthesis**.
7. Because this design takes a long time to route, you will only run to `place_design` and `analyze` at this stage. When synthesis is complete, in the Design Runs window, right-click on the new implementation run and select **Launch Step To → place_design**.



- With the implementation running, select the Design Runs window. Right-click the **synth_1_copy** synthesis run and click **Open Run**.
- When the run has opened, select **Reports → QoR Assessment...** and click **OK**.
- Click **RQA Summary**. The score has improved from 2 to 4.



- Click **Assessment Details**. The Net and LUT budget score has been reduced but not eliminated. This is a consequence of the high frequency that paths are being forced to run at in this design.



- Close the synthesized design.

13. When `place_design` is finished, examine the very top of the implementation log file for the new implementation run. It provides a table summary of the suggestions that have been read in. This summary helps you confirm that what has been read in is what you expect.

```
1. Read QoR Suggestions Summary
```

```
-----
```

```
Read QoR Suggestions Summary
```

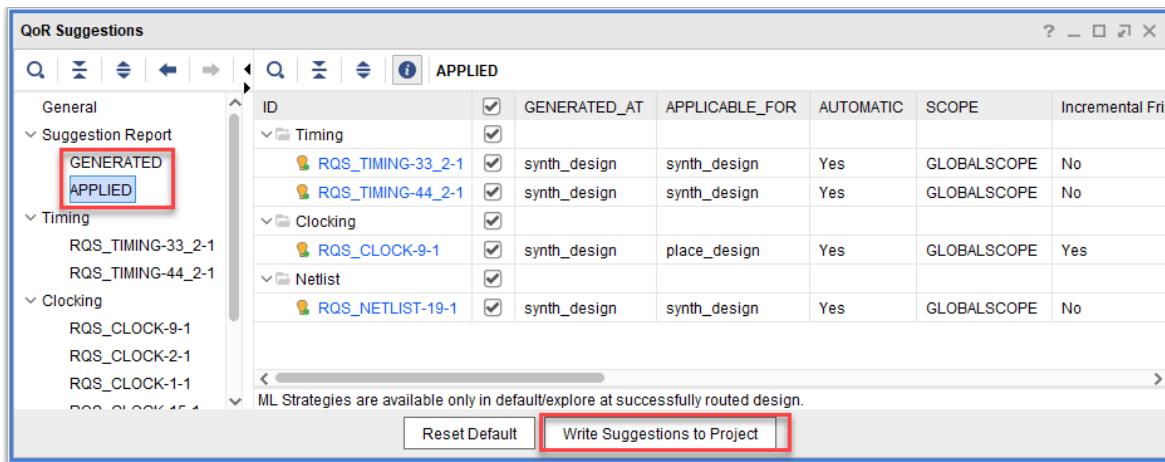
Suggestion Summary	Incr	Friendly	Total
Total Number of Enabled Suggestions	1	1	4
Automatic	1	1	4
Manual	0	0	0
APPLICABLE_FOR			
synth_design	0	0	3
opt_design	0	0	0
That overlap with synthesis suggestions	0	0	0
place_design	1	1	1
postplace_phys_opt_design	0	0	0
route_design	0	0	0
postroute_phys_opt_design	0	0	0
ML Strategy	0	0	0
Total Number of Disabled Suggestions	0	0	0

14. Right-click on the implementation run and select **Open Run Directory**. Open the checkpoint file by double-clicking `top_placed.dcp`. This step is necessary because you are examining an intermediate run step in the interests of saving time.

15. In the new instance of the Vivado tools, select **Reports** → **Report QoR Suggestions ...** and click **OK**.

16. In the new report, there are more sections under Suggestion Report:

- **GENERATED:** New suggestions are listed in this section.
- **EXISTING:** Suggestions that existed previously but have not been applied are listed in this section (not shown).
- **APPLIED:** Suggestions that have been applied are listed in this section.
- **FAILED TO APPLY:** Suggestions that apply to design objects that no longer exist are listed in this section (not shown).



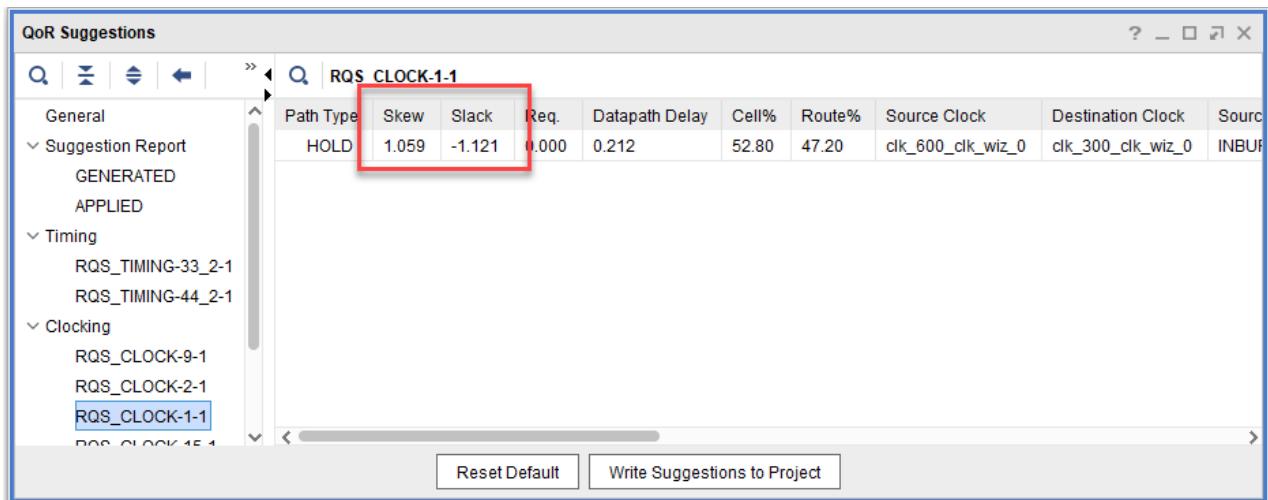
The suggestion file can be written by using the Write Suggestions to Project button.

17. Click **APPLIED** and select the details table for one of the items. For APPLIED suggestions, the timing path summary is still available but it is not possible to cross probe to other views in Vivado because some items might have changed.

Step 6: Accumulating Suggestions

You can now review the newly generated suggestions and add them to the RQS file.

1. Click **GENERATED**. The RQS_CLOCK-15 message reports high THS paths but does not provide an automatic suggestion.
2. Examine RQS_CLOCK-2-1. This suggestion recommends changing the clock buffers to BUFGCE_DIV to improve the timing path uncertainty. It is highly recommended to implement this. Because this suggestion is not automated, however, it requires an RTL edit. If you wish, you can make the recommendation and see the improvement, but this step can be skipped. The next steps focus on the automated suggestions.
3. Click on **RQS_CLOCK-1-1** to view the detailed report. This suggestion applies CLOCK_DELAY_GROUP to related clocks. In this report, you can see that there is a high clock skew and failing slack.



Path Type	Skew	Slack	Req.	Datapath Delay	Cell%	Route%	Source Clock	Destination Clock	Source
HOLD	1.059	-1.121	0.000	0.212	52.80	47.20	clk_600_clk_wiz_0	clk_300_clk_wiz_0	INBUF

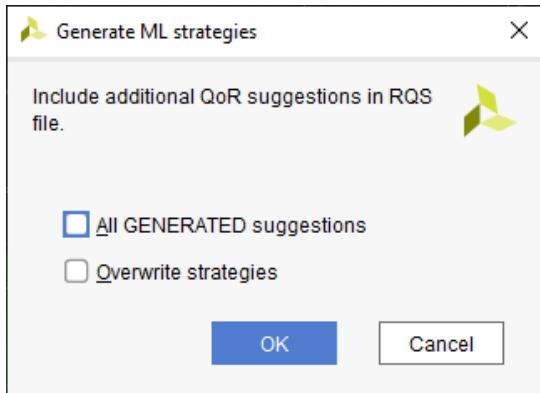
Clock skew is difficult to identify before `place_design` because the skew estimate depends heavily on placement. As a consequence, RQS does not offer this suggestion unless a design is placed. Whenever there is a change in information level, it might be advisable to run `report_qorSuggestions`. The following summarizes the changes as you progress through the tool flow:

- Clocking estimates are accurate after `place_design`.
- Congestion is available after placement and improves further after routing.
- Timing estimates improve throughout the flow and are impacted by the number of paths analyzed.

4. Click **Write Suggestions To Project**. When suggestions are written, the APPLIED status is reset. All the previous suggestions and the new RQS_CLOCK-1-1 suggestion are combined into one file. You can overwrite the previous file and reuse the runs, or create a new file and new runs.
5. Select the file location to overwrite the existing file. You can find out the location of this by selecting it in the sources window. Alternatively, it should be at the following location if you have followed the steps carefully: `<extract_dir>/Lab2/project_2/project_2.srcs/utils_1/imports/project_2`.

You are now at the point where you know the fundamentals in handling RQS files and accumulating suggestions. If you have time, rerun implementation through to `route_design` and examine the impact of the latest suggestion. Alternatively, generate alternative suggestions by running `report_qorSuggestions` on your own design.

6. Close the run.
7. In the Design Runs window, right-click the implementation run `impl_1_copy_1` and select **Launch to → Route Design**. When routing is complete, right-click and select **Generate ML Strategies**. Doing this generates three RQS files that each contain one ML strategy suggestion, the APPLIED suggestions, and optionally any GENERATED suggestions.

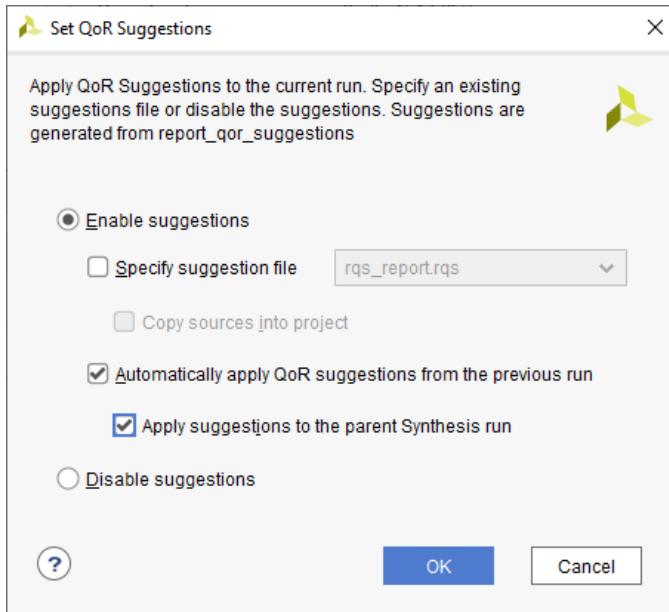


8. After the generation is complete, in the Design Runs window, right-click the implementation run **impl_1_copy_1** and select **Create ML Strategy Runs**. This creates three implementation runs, each targeting a different ML-based implementation strategy.

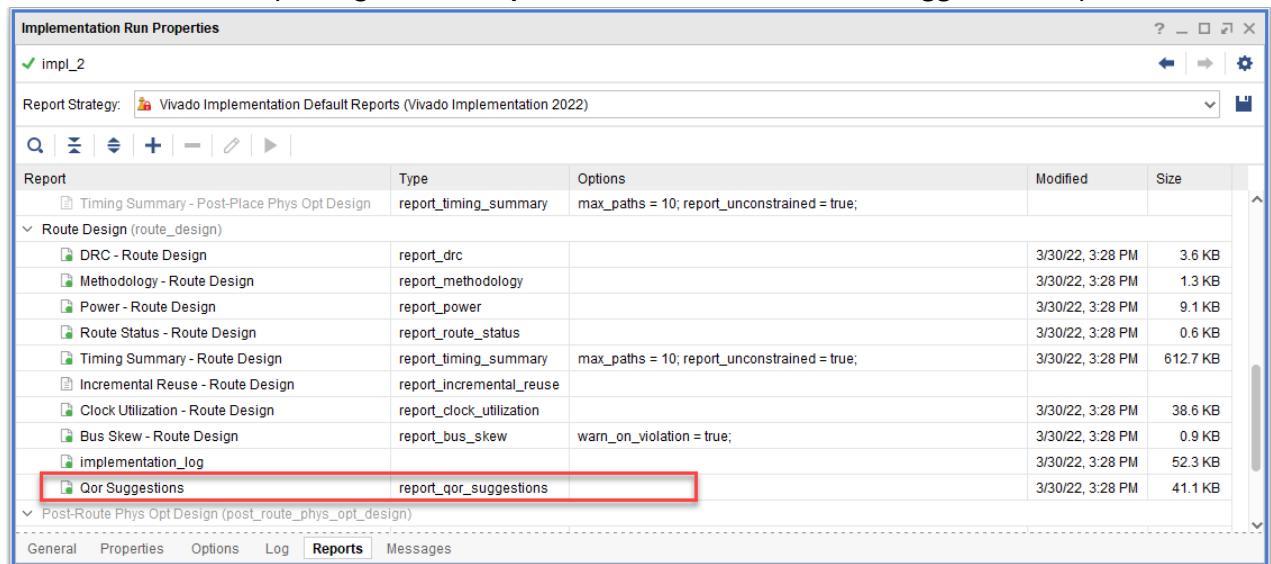
Step 7: Automatically Running QoR Suggestions

In this section you will run the AUTO_RQS flow, where suggestions are generated at the end of an implementation run and automatically added to the run when it is reset.

1. In the Design Runs window, copy the original synthesis and implementation run again using the right-click menu.
2. In the Design Runs window, right-click **impl_2** → **Set QoR Suggestions**.
3. Select **Automatically apply QoR suggestions from the previous run** and **Apply suggestions to the parent Synthesis run**.



4. You can now launch the runs. Suggestions are only generated at the end of `route_design` using this method. The picture the flow sees can be different to the previous steps, so you can expect different results. In addition, all AUTO suggestions generated or regenerated at `route_design` will be applied.
5. Launch the runs. This step takes some time because it launches the full implementation flow.
6. When the run is complete, go to the **Reports** tab and select the **QoR Suggestions** report.



This report is generated automatically when the AUTO RQS flow is enabled. There is also automatic writing of QoR suggestions. The suggestions written to the RQS file have one of the following properties:

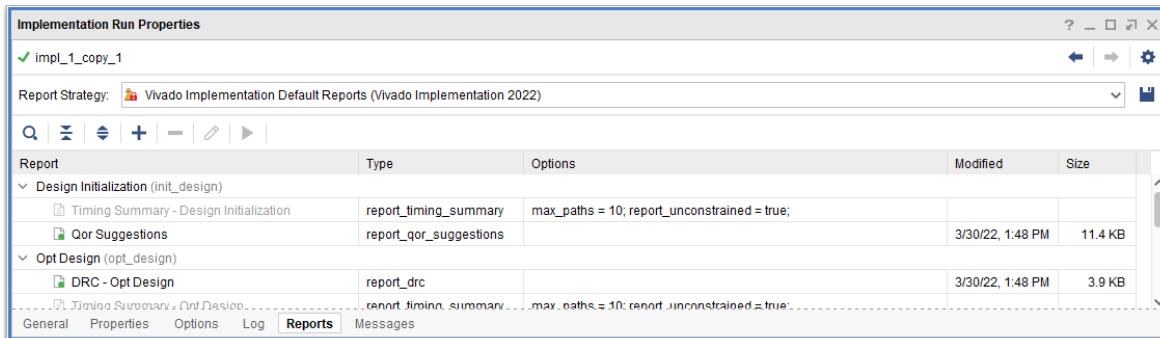
- **GENERATED_AT** or **REGENERATED_AT** equal to the final flow step, which can be `route_design` or `postroute_phys_opt`

- AUTOMATIC
- APPLIED

7. Reset the implementation run. This makes the synthesis run out of date at the same time. Because the option to apply suggestions to the parent synthesis run is selected, the RQS file will also be read in this run.
8. Right-click **impl_1_copy_2** and launch the implementation run, also selecting the option to restart the synthesis run.
9. With the run launched, select **impl_1_copy_2** in the Design Runs window and examine the RQS_FILES property on the Implementation Run Properties window, which will have been updated during the reset run process. At the same time, an RQS file is copied to the **utils_1** fileset. RQS files in the implementation run directory will be deleted when the run is reset.

If the RQS_FILES property previously pointed to a different file, all the APPLIED suggestions from this file are copied to the new RQS file and subsequently, this RQS file is no longer required and is dropped.

10. Select **Implementation Run Properties** and then the **Reports** tab. After the design initialization step is run, a report is made available that contains the QoR suggestions that have been read into the run from the RQS file. Select this report.



11. In the report, you can see that ML strategies have been generated. In the case of the AUTO_RQS flow, the required RQS files are generated for you automatically. These files are combined with all the suggestions that are currently APPLIED, as well as the suggestions that meet the criteria outlined above to be added when the run is reset.

2. ML Strategies				
#	Id	Command	Options	
1	RQS_STRAT-14	opt_design	-directive ExploreArea	
		place_design	-directive EarlyBlockPlacement	
		phys_opt_design	-directive AggressiveExplore	
		route_design	-directive NoTimingRelaxation	
2	RQS_STRAT-68	opt_design	-merge_equivalent_drivers	
		opt_design	-directive Explore	
		place_design	-directive AltSpreadLogic_high	
		phys_opt_design	-directive AggressiveExplore	
		route_design	-directive NoTimingRelaxation	
3	RQS_STRAT-26	opt_design	-directive ExploreSequentialArea	
		place_design	-directive EarlyBlockPlacement	
		phys_opt_design	-directive AggressiveExplore	
		route_design	-directive NoTimingRelaxation	

12. In the Design Runs window, right-click **impl_1_copy_2** → **Open Run Directory**. Locate the **MLStrategy** directory and confirm that there are three RQS files inside it. Close the folder.
13. In the Design Runs window, right-click **impl_1_copy_2** → **Create ML Strategy Runs**. Doing this creates three runs and automatically sets the RQS file and directives for you. ML strategies are covered in more detail in the following lab.
14. When the implementation run completes, new suggestions are generated. Using only **impl_1_copy_2**, repeat steps **6** to **10** for a clearer understanding of how the RQS file is updated with the new suggestions.

Summary

In this lab, you achieved the following:

- Using RQA, you conducted an assessment before and after improvements were implemented, examining an improvement in the score.
- Using RQS, you conducted a complex analysis of a demonstration design. You first examined the reports that showed RQS recommendations to solve implementation problems, then you generated an RQS file and added it to a project implementation run. The Vivado implementation tools executed the suggestions automatically for you. You subsequently performed further analysis and generated further suggestions, accumulating them in the RQS file.

Running ML Strategies

Introduction

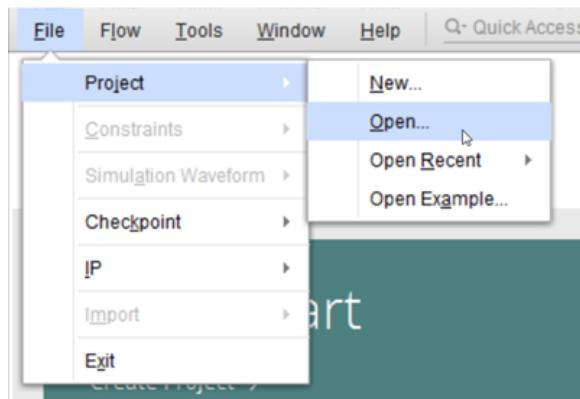
The `report_qorSuggestions` command can generate an implementation design strategy that is predicted to be optimal for the design using machine learning algorithms. In this tutorial, you will look at:

- How to generate ML strategy suggestions.
- How to setup the implementation run to use ML strategy suggestions.
- Reporting specifics related to ML strategies.

Step 1: Generating an ML Strategy RQS File

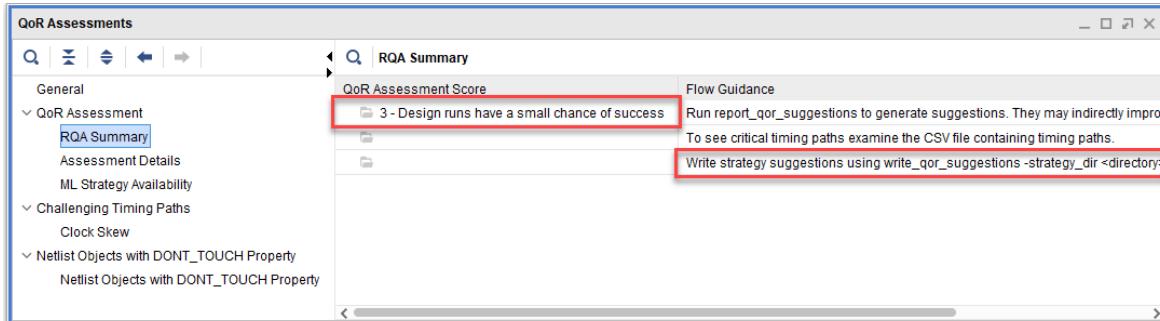
This step shows the process of opening a routed design with QoR suggestions and generating a new RQS file with strategies. For details on the design, refer to [Step 1: Understanding the Design](#).

1. In the Vivado Design Suite, go to **File** → **Project** → **Open** and select the project located in `<extract_Dir>/Lab3/project_2`.

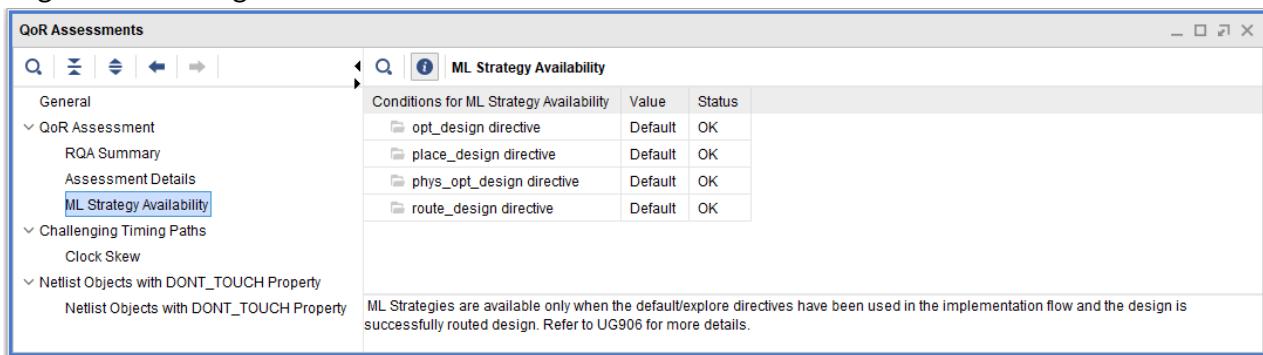


2. In the Flow Navigator, click **Open Implemented Design**.
3. From the pull-down menus, select **Reports** → **Report QoR Assessment**, and click **OK**.

4. In the RQA Summary table, you will see the QoR Assessment Score and Flow Guidance. This table helps identify good candidate designs on which to use ML strategy suggestions. QoR assessment scores of 3 and above have a chance to meet timing. Designs with an RQA score of less than 3 are not prevented from generating ML strategies.



5. Click **ML Strategy Availability**. This table details the required directives for the reference run to generate strategies.



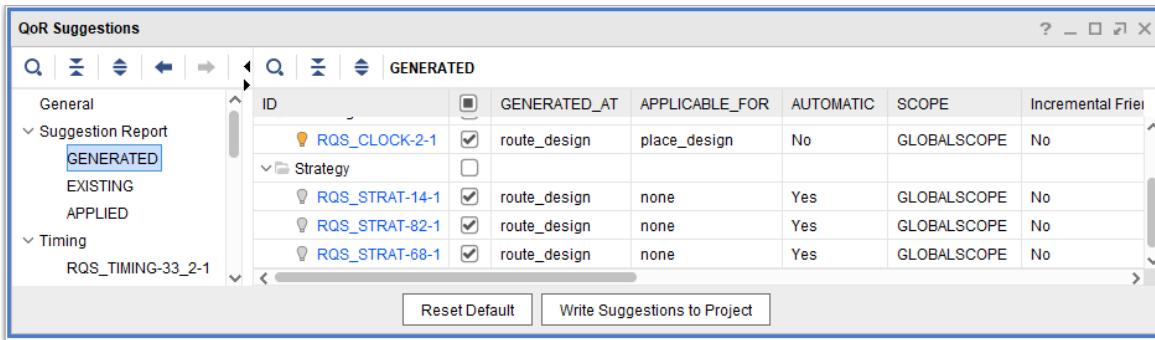
The status for all directives must be listed as OK to generate strategies. The requirements are as follows:

- The `opt_design` directive value must be either Default or Explore.
- The `place_design`, `phys_opt_design`, and `route_design` conditions must be the same as each other and must be set to either Default or Explore.

6. In the Design Runs window, confirm the strategy is Vivado Implementation Defaults. This requirement is met when a design has been run with either the Vivado implementation defaults or the `performance_explore` strategy.

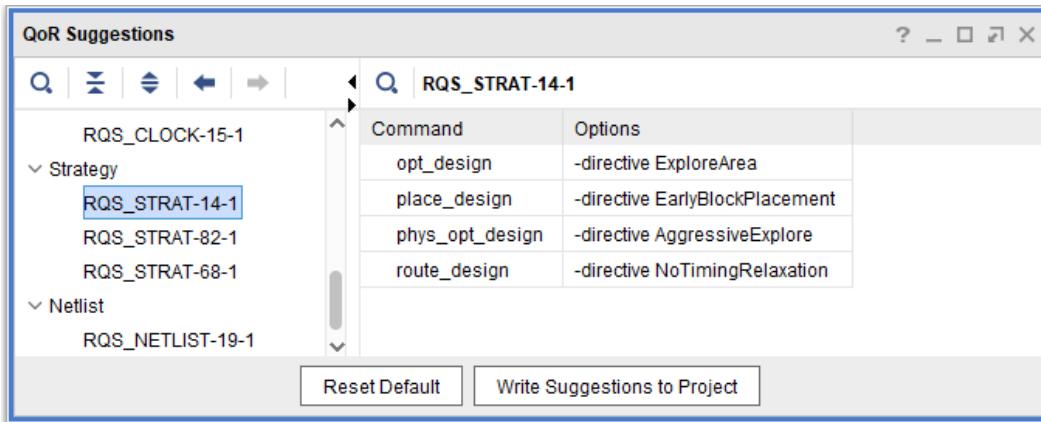
7. From the pull-down menus, select **Reports** → **Report QoR Suggestions**, and click **OK**.

8. In the QoR suggestion report, select **GENERATED**. Three new strategies have been generated.



ID	GENERATED_AT	APPLICABLE_FOR	AUTOMATIC	SCOPE	Incremental Friend
RQS_CLOCK-2-1	route_design	place_design	No	GLOBALSCOPE	No
RQS_STRAT-14-1	route_design	none	Yes	GLOBALSCOPE	No
RQS_STRAT-82-1	route_design	none	Yes	GLOBALSCOPE	No
RQS_STRAT-68-1	route_design	none	Yes	GLOBALSCOPE	No

9. In the Strategy section, select the topmost strategy. Here, you can see the details of the strategy being suggested. It is possible to set these up manually, but to automate the process more easily, the recommended flow is to read an RQS file containing strategies and set the directive to RQS on the implementation commands.



Command	Options
opt_design	-directive ExploreArea
place_design	-directive EarlyBlockPlacement
phys_opt_design	-directive AggressiveExplore
route_design	-directive NoTimingRelaxation

10. Select **Write Suggestions to Project** to write the following files:

- A top-level RQS file that does not contain strategies (this file can be ignored)
- An RQS file for each strategy as well as any other QoR suggestions (written to the `MLStrategy` directory).

Generating the strategy RQS files is the first part of a two-step process. This way of generating the suggestions gives complete control over what other suggestions are in the RQS file. Other ways to generate these files are as follows:

- Automatic generation using `AUTO_RQS` flow (as seen in [Lab 2: Using Report QoR Suggestions and Report QoR Assessment for Timing Closure](#))
- Clicking **Generate ML Strategies** from the right-click menu in the Design Runs window
- Automatic generation at the end of Stage 1: Design Optimization of intelligent design runs
- A post-route Tcl hook script

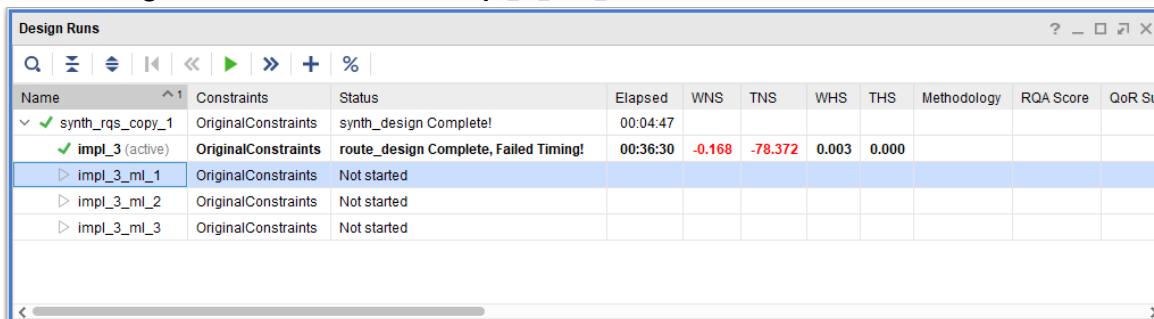
Step 2: Creating ML Strategy Runs

In this step, you will create the ML strategy implementation runs using the files created in [Step 1: Generating an ML Strategy RQS File](#).

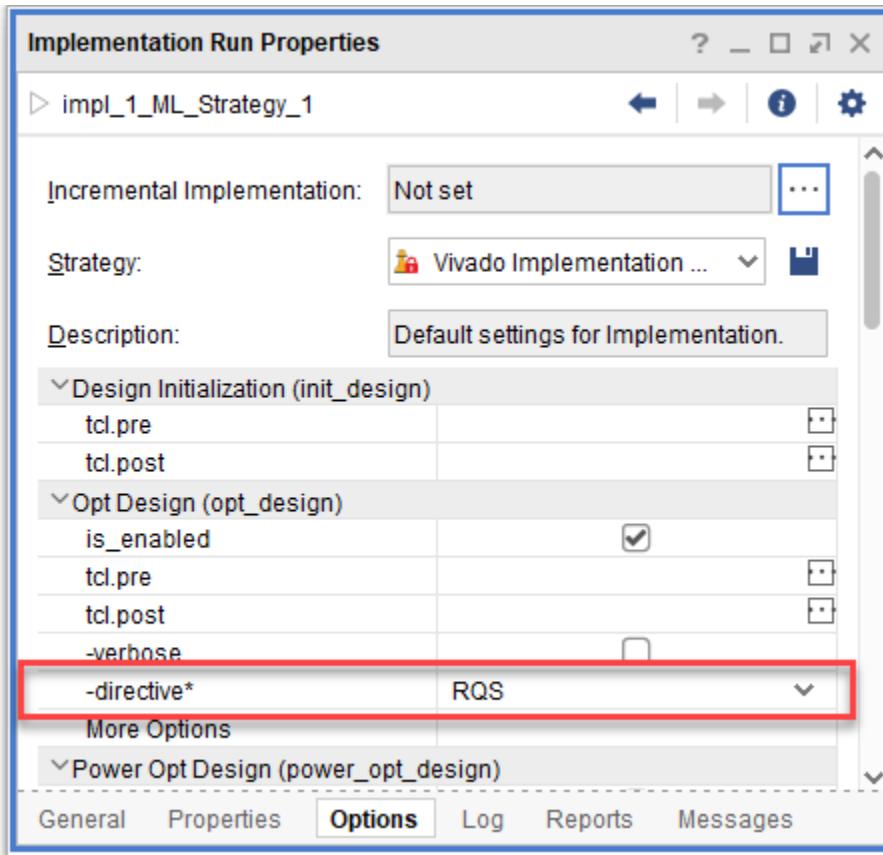
1. In the Design Runs window, select **impl_3**, then right-click and select **Open Run Directory**.
2. Search for the **MLStrategy** directory and examine the contents. You will see three RQS files and three non-project-based Tcl scripts. The RQS files are common for both project and non-project flows. The non-project scripts are examples of how to use the RQS file.

Name	Date modified	Type	Size
impl_3SuggestionFile1.rqs	31/03/2022 16:41	RQS File	11 KB
impl_3SuggestionFile2.rqs	31/03/2022 16:41	RQS File	11 KB
impl_3SuggestionFile3.rqs	31/03/2022 16:41	RQS File	11 KB
NonProject_MLStrategyCreateRun1.tcl	31/03/2022 16:41	TCL File	1 KB
NonProject_MLStrategyCreateRun2.tcl	31/03/2022 16:41	TCL File	1 KB
NonProject_MLStrategyCreateRun3.tcl	31/03/2022 16:41	TCL File	1 KB

3. The next step is to generate the ML strategy runs. As long as the files reside in the `<run_directory>/MLStrategy` directory, this process can be automated. In the Design Runs window, right-click the implementation run and select **Create ML Strategy runs**. Doing this creates three runs, one for each ML strategy file. Configure the runs to use the RQS directive and the RQS file to be read into each of them.
4. In the Design Runs window, select **impl_3_ML_1**.



5. In the Implementation Run Properties window, select the **Properties** tab and confirm that **RQS_FILES** is set.
6. In the Implementation Run Properties window, select the **Options** and confirm the directive is set to RQS for the `opt_design`, `place_design`, `phys_opt_design`, and `route_design` commands.



You are now set up to run with ML strategies. By the time you have an ML strategy file, you cannot generate new strategies after design changes, but you can add other suggestions.

7. You are now ready to launch the runs. Select all the ML strategy runs, right-click, and select **Launch Runs....** The runs will now proceed in parallel, and complete like a standard run.

Summary

In this lab, you achieved the following:

- Used `report_qorSuggestions` to generate ML strategies.
- Created the RQS and Tcl files using `write_qorSuggestions`.
- Sourced the Tcl to set up the ML strategy runs and confirmed the key aspects of setting up an ML strategy run.

Intelligent Design Runs

Introduction

In this tutorial, you will work with intelligent design runs (IDRs). An IDR is a simple-to-use implementation run that extends the performance of your design by automating QoR suggestions and ML strategies. These features are applied in a way that maximizes the improvement in performance.

This lab covers how to:

- Create an IDR.
- Add Tcl hooks to the IDR.
- Examine the log file for an IDR.
- Examine reports for an IDR.
- Create a single pass run from an IDR.

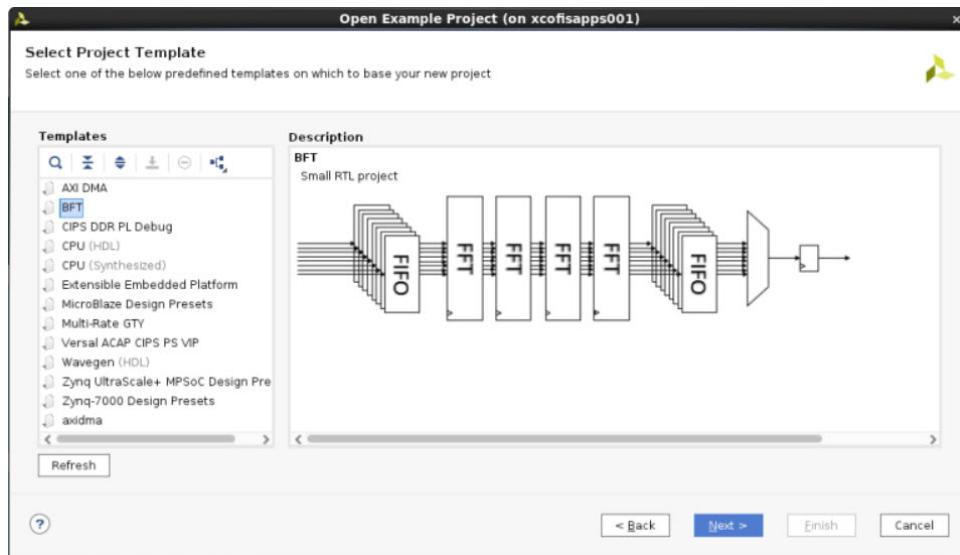
Step 1: Creating Intelligent Design Runs

In this section, you will create an example design and an Intelligent Design Run (IDR).

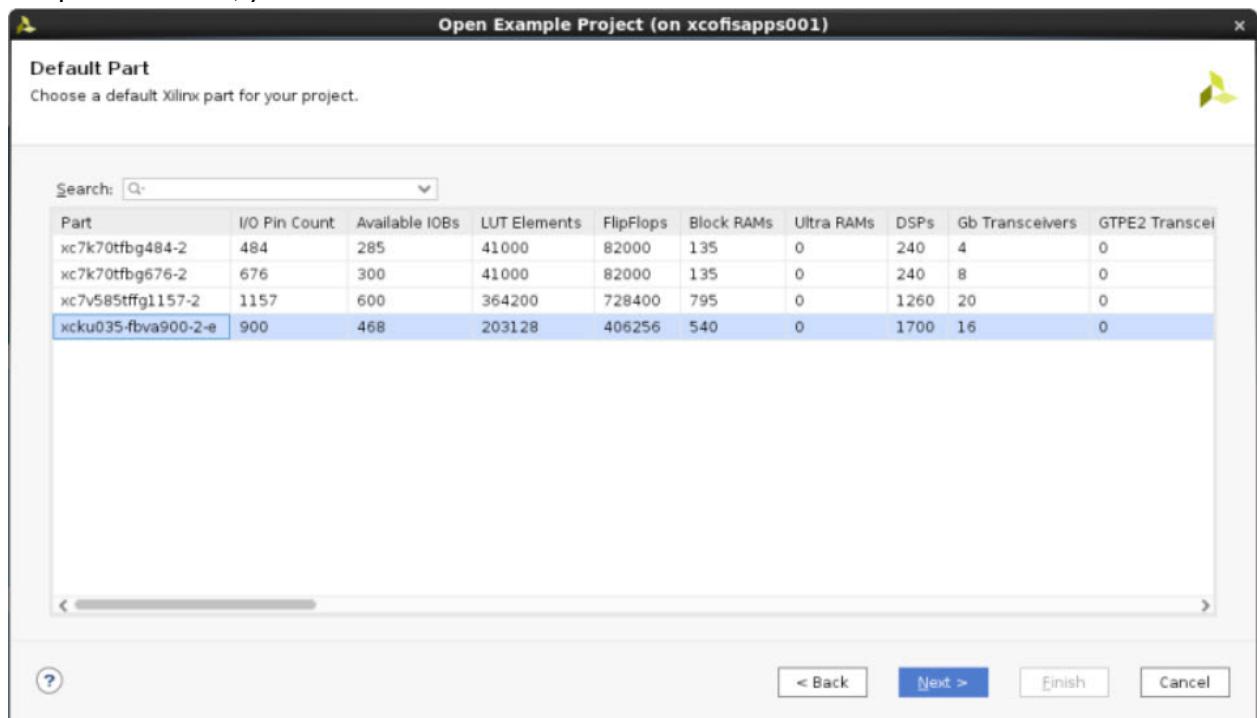
1. First, you need a project. Select **Open Example Project**.



2. Click **Next→ BFT** and then select **Next** again.



3. In the next screen, ensure you are working on in a suitable writable directory, and select **Next**.
4. For part selection, you *must* select **xcku035-fbva900-2-e** and then select **Next → Finish**.

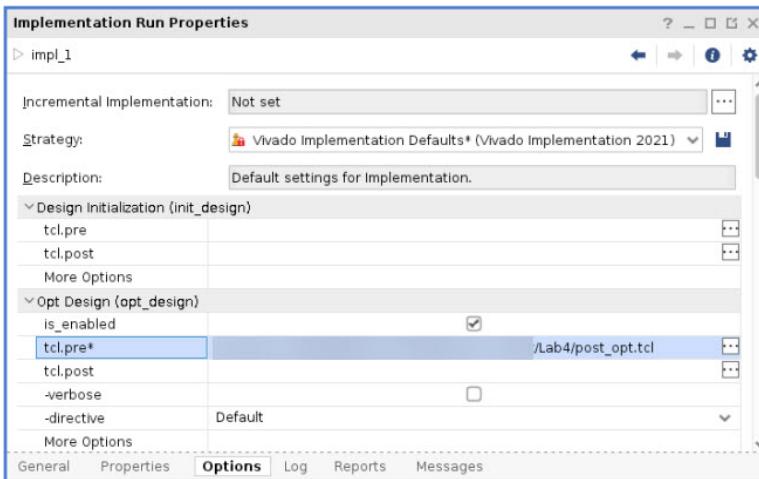


You should now have an open project that is waiting to be synthesized.

Note: IDRs are not supported for 7 series devices. Selecting the incorrect family requires you to regenerate the project.

5. This lab aims to demonstrate how to add a Tcl file to an IDR. To do this, add a file to **impl_1**. In the Design Runs window, select **impl_1**.

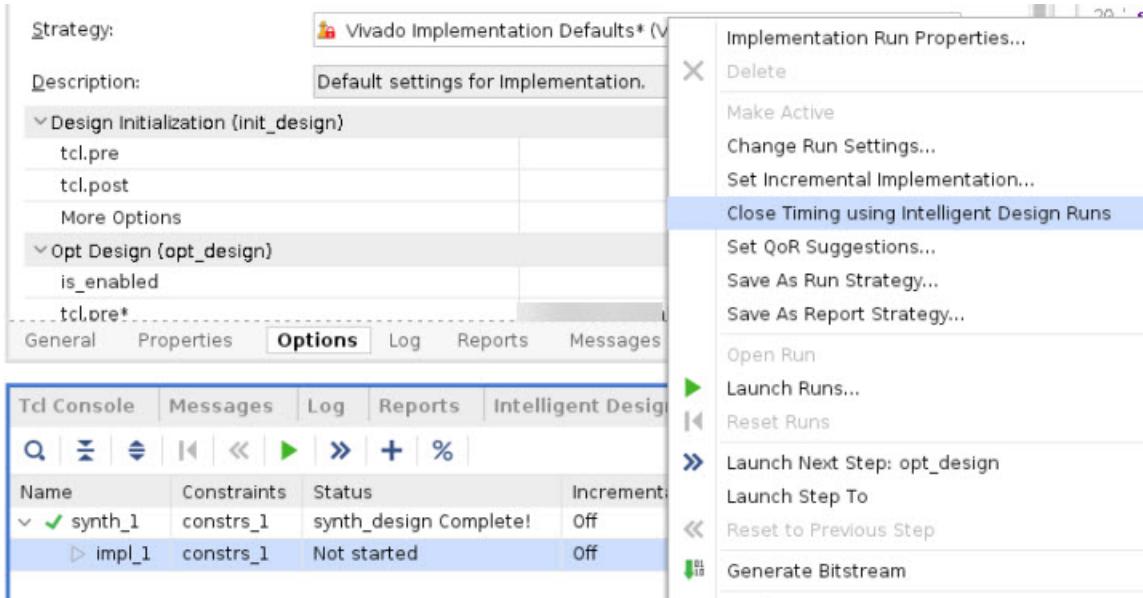
6. In the Implementation Run Properties window, select the **Options** tab.
7. Scroll to the `opt_design tcl.pre` option and add the Tcl script `<extract_dir>/Lab4/pre_opt.tcl` to the option.



Alternatively, use the following Tcl syntax:

```
add_files -fileset utils_1 -norecurse <extract_dir>/Lab4/post_opt.tcl
set_property STEPS.OPT_DESIGN.TCL.PRE [ get_files <extract_dir>/Lab4/
post_opt.tcl -of [get_fileset utils_1] ] [get_runs impl_1]
```

8. In the Flow Navigator, click **Run Synthesis**.
9. The next step is to create the Intelligent Design Run. In the **Design Runs** window, right-click on `impl_1` → **Close Timing using Intelligent Design Runs**.



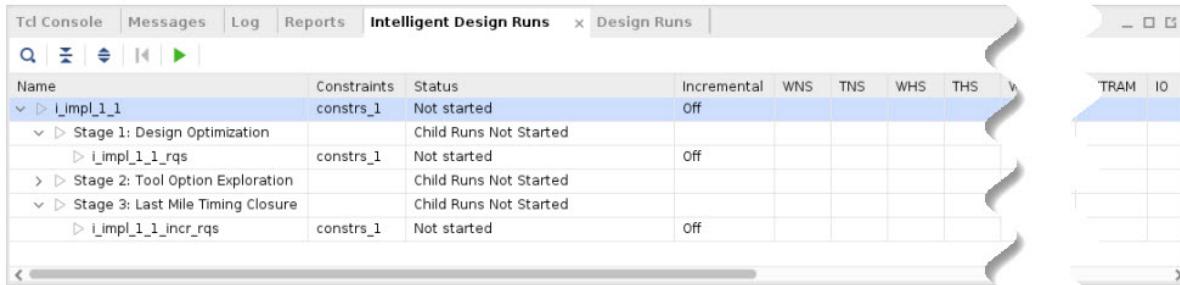
If you try this step again to create an additional run, you will find that the option is disabled. This is to prevent the creation of runs that would produce the same results. Runs created from the same netlist with different directives produce the same results because the IDR controls the directive, meaning that they are effectively the same.

If you have a different floorplan or speed grade, create different implementation runs. You can create one IDR from each implementation run.

Step 2: Navigating Intelligent Design Runs

In this step, you will learn how to control intelligent design runs and navigate options and reports.

1. Select the **Intelligent Design Runs** tab at the bottom of the Vivado® IDE.

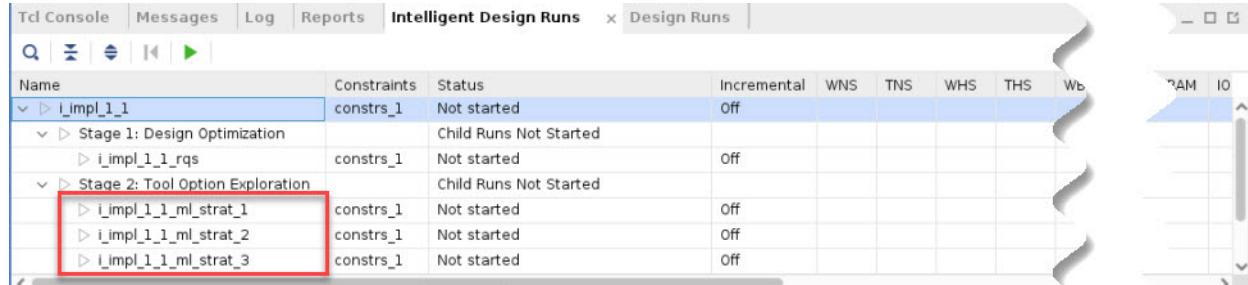


This window is opened when an IDR is created. If closed, it can be reopened by selecting **Window**→**Intelligent Design Runs**. From this window you can see that there is a hierarchical nature to intelligent design runs. The top-level run is split into three stages:

- Stage 1: Design Optimization
- Stage 2: Tool Option Exploration
- Stage 3: Last Mile Timing Closure

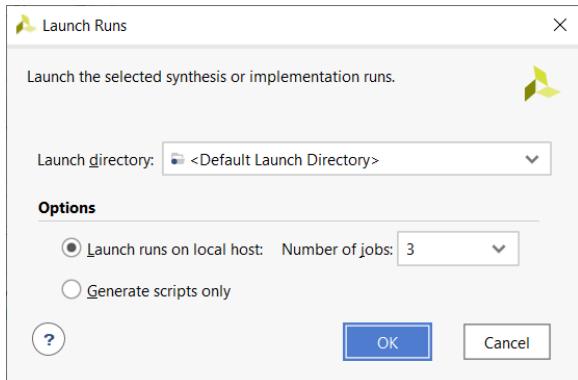
In this lab, only stage 1 is demonstrated. This is the most complex stage, but can be completed quickly because you are working with a simple design.

2. Expand **Stage 2: Tool Option Exploration**.

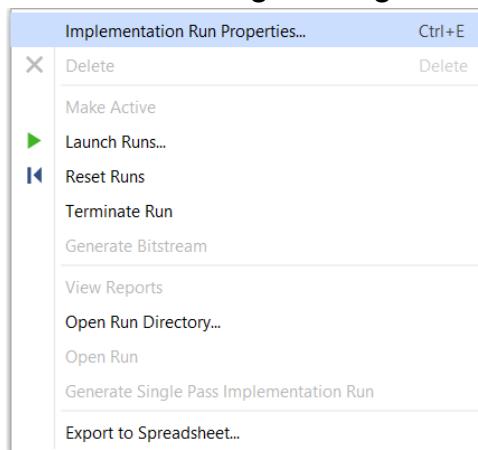


There are three runs underneath this stage. These runs can be run in parallel if your compute resources allow.

3. Right-click **i_impl_1_1**→**Launch Runs** and select how you normally run jobs. Click **OK**.



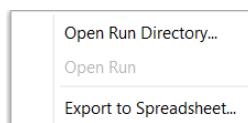
4. Select **impl_1_1** and view the Properties in the Implementation Run Properties window. A **PARENT** property is set to **synth_1**. As is the case with a normal implementation run, there is a dependency on the synthesis run to complete when this is set. If the RTL code is modified, the run will go out of date.
5. Click the **Design Runs** tab and confirm that the **synth_1** run is running or finished.
6. Click back to **Intelligent Design Runs** and right-click on the **impl_1_1** top-level run.



The right-click menu from the top-level run is the main way to control the intelligent design runs. Some options only become available at specific times:

- **Reset Runs** is only available after a run is launched.
- **View Reports** is only available after reports are generated.
- **Open Run** is only available after a run is successfully completed.
- **Generate Single Pass Run** is only available after a run is successfully completed.

7. Select the lower-level run, **i_impl_1_1_rqs**, and right-click on it.

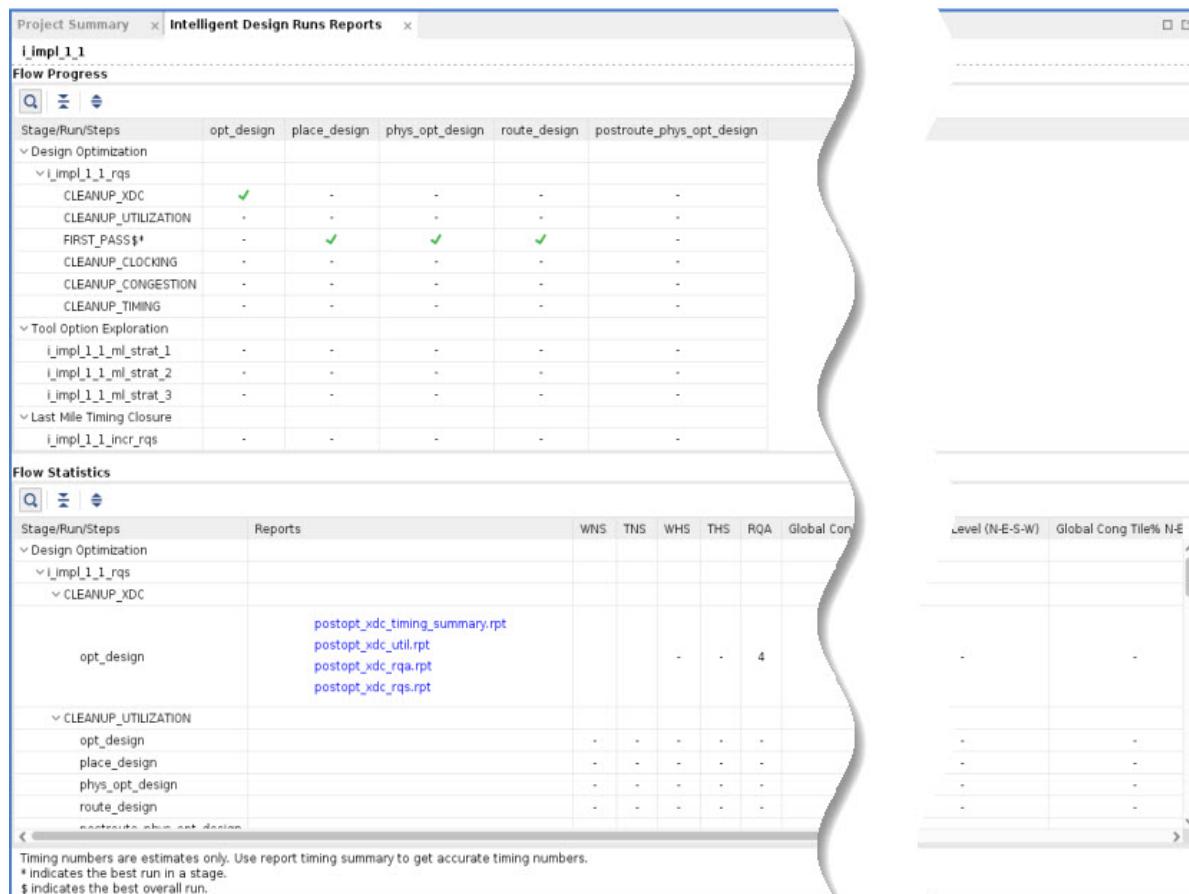


When the lower-level run is selected, you can see that there is a reduced set of options. The Open Run Directory and Open Run options are scoped to the lower-level run. If Open Run is selected, the best completed routed run is opened.

Step 3: Analyzing the Reports and Log File

In this task, you will see where to find reports and analyze runs and intermediate checkpoints from an intelligent design run.

1. If the window is not already open, select **Reports** → **Intelligent Design Run Reports**. This window captures all the metrics that are captured throughout the IDR and provides links to any generated reports.



The report is broken into two windows. The Flow Progress section is in the top half. This details which steps have been run and which steps are running currently. This is more important for an IDR as opposed to a standard implementation run for the following reasons:

- The IDR is a dynamic run, and not all stages are run for every design.
- The steps might be run repeatedly as the run is reset to apply QoR suggestions.

The Flow Statistics section is in the bottom half, and provides the following:

- Hyperlinks to any available reports
- Timing metrics such as WNS/TNS/WHS/THS
- RQA score
- Congestion level and tile % metrics for post-place and post-route designs

2. Focus in on the **Flow Progress** section of the report.

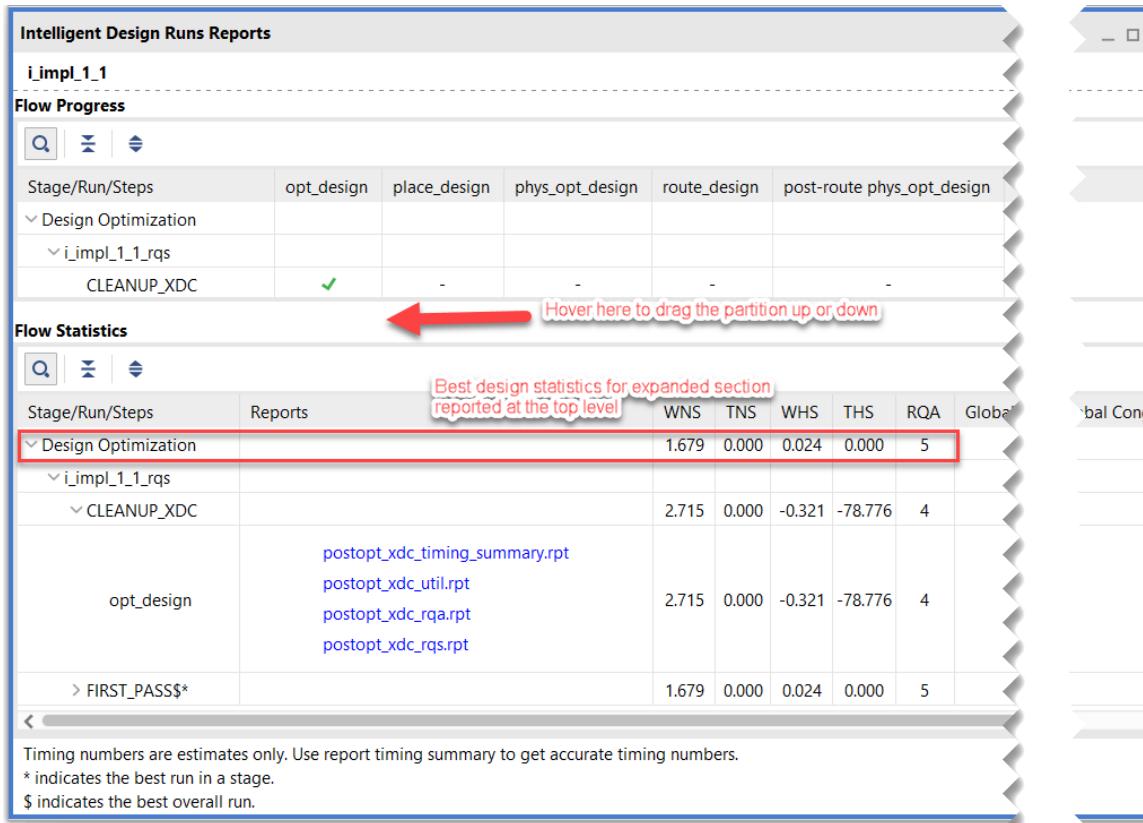
Flow Progress					
Stage/Run/Steps	opt_design	place_design	phys_opt_design	route_design	postroute_phys_opt_design
Design Optimization					
i_impl_1_1_rqs					
CLEANUP_XDC	✓	-	-	-	-
CLEANUP_UTILIZATION	-	-	-	-	-
FIRST_PASS\$*	-	✓	✓	✓	-
CLEANUP_CLOCKING	-	-	-	-	-
CLEANUP_CONGESTION	-	-	-	-	-
CLEANUP_TIMING	-	-	-	-	-

- Green ticks indicate a completed stage.
- Hyphens indicate the stage is not run.
- Circles indicate the stage is running.

Because there are no circle icons, you can infer that the run has completed successfully. Note also the FIRST_PASS step, which is generated for reporting purposes. This special step occurs when there are no utilization or clock suggestions. Only designs without these suggestions have this step.

At the bottom, there is a table footer that describes the meaning of the the \$ and * notations. These notations help you identify which runs you are opening when you select **Open Run**.

3. Look at the **Flow Statistics** section of the report. This report can be larger, and you can make more space by reducing the divider between the section above and clicking on the arrows to reduce sections of the report.



The screenshot shows the 'Intelligent Design Runs Reports' window for run **i_impl_1_1**. The window is divided into three main sections: **Flow Progress**, **Flow Statistics**, and **Congestion Metrics** (partially visible on the right).

Flow Progress: Shows the status of various stages: **opt_design**, **place_design**, **phys_opt_design**, **route_design**, and **post-route phys_opt_design**. The **CLEANUP_XDC** stage is marked with a green checkmark.

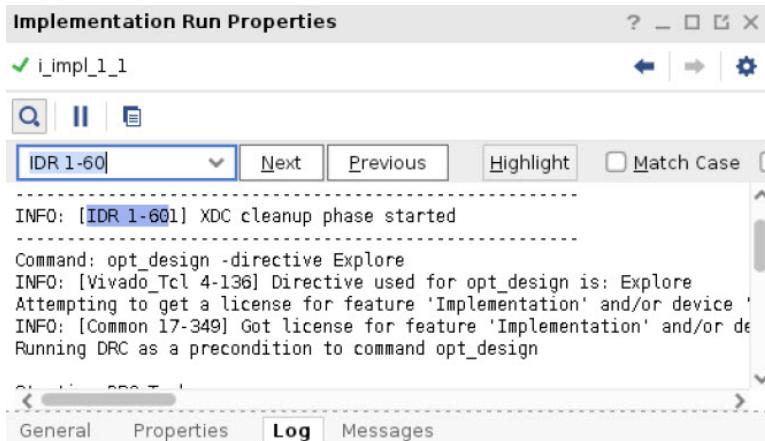
Flow Statistics: Displays timing and RQA statistics for the **Design Optimization** stage. A red arrow points to a horizontal separator line with the text **Hover here to drag the partition up or down**. The table includes columns for Stage/Run/Steps, Reports, WNS, TNS, WHS, THS, RQA, and Global. The **CLEANUP_XDC** row is expanded, showing reports like **postopt_xdc_timing_summary.rpt**, **postopt_xdc_util.rpt**, **postopt_xdc_rqa.rpt**, and **postopt_xdc_rqs.rpt**. The **opt_design** row also lists these reports. The **FIRST_PASS\$*** row shows overall statistics: 1.679, 0.000, 0.024, 0.000, and 5.

Congestion Metrics: Shows metrics for **Design Optimization**, **i_impl_1_1_rqs**, and **CLEANUP_XDC**. The table includes columns for Stage/Run/Steps, Reports, WNS, TNS, WHS, THS, RQA, and Global. The **CLEANUP_XDC** row is expanded, showing reports like **postopt_xdc_timing_summary.rpt**, **postopt_xdc_util.rpt**, **postopt_xdc_rqa.rpt**, and **postopt_xdc_rqs.rpt**. The **opt_design** row also lists these reports. The **FIRST_PASS\$*** row shows overall statistics: 1.679, 0.000, 0.024, 0.000, and 5.

Timing numbers are estimates only. Use report timing summary to get accurate timing numbers.
 * indicates the best run in a stage.
 \$ indicates the best overall run.

On the left of this window are the reports, in the middle are the timing and RQA statistics, and on the right are the congestion metrics. Statistics are only available if the flow stage has been run, or if the flow stage generates the statistics. For example, during place, hold statistics are not generated.

4. Expand **FIRST_PASS** and click **postplace_physopt_first_pass_util.rpt**. Pay attention to the name of this report: **postplace_phys_opt** indicates the implementation step and **first_pass** indicates the flow step. When you are ready, close the report.
5. In the Intelligent Design Runs window, right-click **impl_1_1** and select **Open Run Directory**. In this directory, open the **idrTextReport.txt** file. This is the text equivalent to the IDE report. It contains the same information except the links to the report. When you are ready, close the file.
6. In the directory explorer, go up one level to the **project_1.runs** folder. There is a directory for the main IDR and each of the sub-runs. Click into **impl_1_1_rqs**. Here, you can see all the reports and intermediate checkpoints. When you are ready, close the directory explorer.
7. In the Intelligent Design Runs window, select the top-level run **impl_1_1**.
8. In the Implementation Run Properties window, select the **Log** tab and maximize the window.
9. Select the search icon and enter "IDR 1-60". Click **Next** multiple times to navigate through the log file. This ID highlights the start and end banners that have been added when you enter and exit a step within Stage 1: Design Optimization.



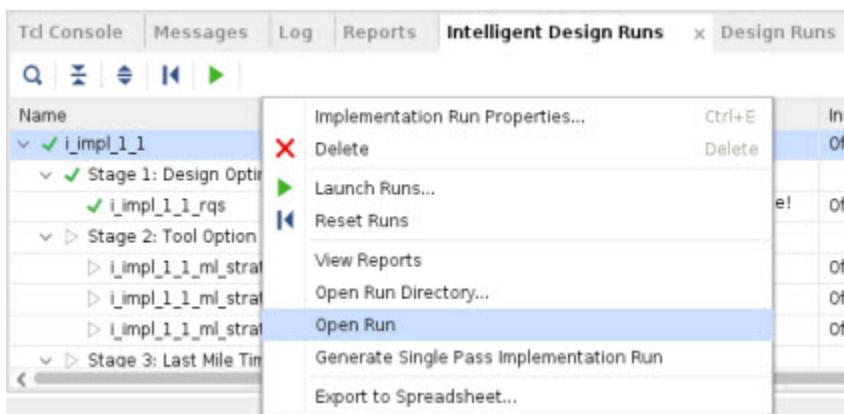
As the implementation process goes back and forth, it can be difficult to identify which step you are in. These messages can help clarify this.

10. Search for "INFO-TUTORIAL4". This has been inserted by the Tcl script you added in Step 1. This Tcl script is executed every time `opt_design` is called. In this case, `opt_design` is called only once, so it is similar to a normal flow.

Step 4: Final Analysis

In this task, you will see how to analyze the design and create a single pass run.

1. In the Intelligent Design Runs window, right-click on the top-level run, `i_impl_1_1`, and then click **Open Run**. Doing this opens the best overall run from the three stages. This is also the best run from stage 1. Stage 2 and stage 3 were not run with this design. The best run and best stage run are indicated by the \$ and * notations.

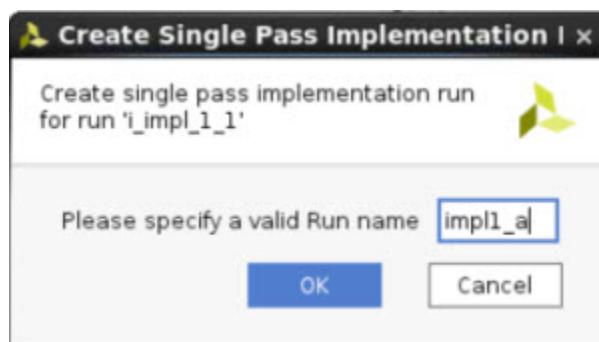


A device view opens where you can generate reports and analyze the design as you would with a normal implementation run. When you are ready, close the design.

2. In the Intelligent Design Runs window, right-click on the top-level **impl_1_1_rqs** and click **Open Run Directory**. This directory has all the intermediate checkpoints from the stage. You can open a new instance of Vivado and open the checkpoints for further analysis. If the DCP is associated with Vivado, you can open the checkpoints by double-clicking on them.
3. When you are finished with analyzing the design, because an IDR can take the equivalent of many implementation runs to complete, it is recommended to run the required commands and suggestions in a single implementation run. In the Intelligent Design Runs window, right-click on the top-level **impl_1_1** and select **Generate Single Pass Implementation Run**.



4. In the Create Single Pass Implementation dialog box, enter a run name, **impl1_a**, and click **OK**.



5. Switch to the **Design Runs** window. You can see that the new run has been made. Right-click on the run and select **Launch Runs**. Confirm that the results are the same as the IDR.

Note: For more difficult runs, you can examine the RQS_FILES property and directives, but because this is a design that meets timing easily, these are not set.

Summary

In this tutorial, you learned how to:

- Create and launch intelligent design runs.
- Analyze reports and checkpoints generated by an IDR.
- Create a single pass flow.

Additional Resources and Legal Notices

Revision History

The following table shows the revision history for this document.

Section	Revision Summary
05/11/2022 Version 2022.1	
Lab 1: Setting Waivers with the Vivado IDE	Minor updates.
Step 1: Generating an ML Strategy RQS File	Section updated.
Step 2: Creating ML Strategy Runs	Section updated.
Lab 2: Using Report QoR Suggestions and Report QoR Assessment for Timing Closure	General updates.
Step 7: Automatically Running QoR Suggestions	New section.

Please Read: Important Legal Notices

The information disclosed to you hereunder (the "Materials") is provided solely for the selection and use of Xilinx products. To the maximum extent permitted by applicable law: (1) Materials are made available "AS IS" and with all faults, Xilinx hereby DISCLAIMS ALL WARRANTIES AND CONDITIONS, EXPRESS, IMPLIED, OR STATUTORY, INCLUDING BUT NOT LIMITED TO WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, OR FITNESS FOR ANY PARTICULAR PURPOSE; and (2) Xilinx shall not be liable (whether in contract or tort, including negligence, or under any other theory of liability) for any loss or damage of any kind or nature related to, arising under, or in connection with, the Materials (including your use of the Materials), including for any direct, indirect, special, incidental, or consequential loss or damage (including loss of data, profits, goodwill, or any type of loss or damage suffered as a result of any action brought by a third party) even if such damage or loss was reasonably foreseeable or Xilinx had been advised of the possibility of the same. Xilinx assumes no obligation to correct any errors contained in the Materials or to notify you of updates to the Materials or to product

specifications. You may not reproduce, modify, distribute, or publicly display the Materials without prior written consent. Certain products are subject to the terms and conditions of Xilinx's limited warranty, please refer to Xilinx's Terms of Sale which can be viewed at <https://www.xilinx.com/legal.htm#tos>; IP cores may be subject to warranty and support terms contained in a license issued to you by Xilinx. Xilinx products are not designed or intended to be fail-safe or for use in any application requiring fail-safe performance; you assume sole risk and liability for use of Xilinx products in such critical applications, please refer to Xilinx's Terms of Sale which can be viewed at <https://www.xilinx.com/legal.htm#tos>.

AUTOMOTIVE APPLICATIONS DISCLAIMER

AUTOMOTIVE PRODUCTS (IDENTIFIED AS "XA" IN THE PART NUMBER) ARE NOT WARRANTED FOR USE IN THE DEPLOYMENT OF AIRBAGS OR FOR USE IN APPLICATIONS THAT AFFECT CONTROL OF A VEHICLE ("SAFETY APPLICATION") UNLESS THERE IS A SAFETY CONCEPT OR REDUNDANCY FEATURE CONSISTENT WITH THE ISO 26262 AUTOMOTIVE SAFETY STANDARD ("SAFETY DESIGN"). CUSTOMER SHALL, PRIOR TO USING OR DISTRIBUTING ANY SYSTEMS THAT INCORPORATE PRODUCTS, THOROUGHLY TEST SUCH SYSTEMS FOR SAFETY PURPOSES. USE OF PRODUCTS IN A SAFETY APPLICATION WITHOUT A SAFETY DESIGN IS FULLY AT THE RISK OF CUSTOMER, SUBJECT ONLY TO APPLICABLE LAWS AND REGULATIONS GOVERNING LIMITATIONS ON PRODUCT LIABILITY.

Copyright

© Copyright 2012-2022 Xilinx, Inc. Xilinx, the Xilinx logo, Alveo, Artix, Kintex, Kria, Spartan, Versal, Vitis, Virtex, Vivado, Zynq, and other designated brands included herein are trademarks of Xilinx in the United States and other countries. All other trademarks are the property of their respective owners.