

Vivado Design Suite Tutorial

Design Analysis and Closure Techniques

UG938 (v2021.1) July 14, 2021

Revision History

The following table shows the revision history for this document.

Section	Revision Summary
07/14/2021 Version 2021.1	
Lab 2: Increasing Design Performance Using Report QoR Suggestions	Added clarifications.
Lab 3: Running ML Strategies	Added clarifications.
Lab 4: Intelligent Design Runs	New section added.

Table of Contents

Revision History	2
Tutorial Overview	5
Introduction.....	5
Tutorial Description.....	5
Software Requirements.....	5
Locating Tutorial Design Files.....	5
Navigating Content by Design Process.....	6
Lab 1: Setting Waivers with the Vivado IDE	7
Introduction.....	7
Step 1: Starting the Vivado IDE.....	7
Step 2: Generating the CDC Report.....	8
Step 3: Waiving a Single CDC Violation.....	9
Step 4: Generating a Report for Waived Violations.....	13
Step 5: Generating a Text Report with Details for Waived Violations.....	14
Step 6: Waiving Multiple CDC Violations.....	15
Step 7: Exporting Waivers.....	18
Step 8: Using the create_waiver Command.....	19
Step 9: Waiving Multiple CDC Violations.....	20
Step 10: Waiving Multiple DRC Violations.....	22
Step 11: Generating a Summary Report for Waived Violations.....	27
Step 12: Using Waiver Commands.....	30
Summary.....	31
Lab 2: Increasing Design Performance Using Report QoR	
Suggestions	32
Introduction.....	32
Step 1: Understanding the Design.....	33
Step 2: Running Report QoR Suggestions.....	35
Step 3: Understanding the Report.....	36
Step 4: Run with Suggestions.....	42
Step 5: Accumulating Suggestions.....	46

Summary.....	47
Lab 3: Running ML Strategies	49
Introduction.....	49
Step 1: Generating an ML Strategy RQS File.....	49
Step 2: Creating ML Strategy Runs.....	51
Summary.....	53
Lab 4: Intelligent Design Runs.....	54
Introduction.....	54
Step 1: Creating Intelligent Design Runs.....	54
Step 2: Navigating Intelligent Design Runs.....	57
Step 3: Analyzing the Reports and Log File.....	59
Step 4: Final Analysis.....	62
Summary.....	64
Appendix A: Additional Resources and Legal Notices.....	65
Please Read: Important Legal Notices.....	65

Tutorial Overview

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>`.

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](#) 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: Increasing Design Performance Using Report QoR Suggestions](#)
 - [Lab 3: Running ML Strategies](#)
 - [Lab 4: Intelligent Design Runs](#)

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>/src/Lab1
```

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

```
vivado my_ip_example_design_placed.dcp
```

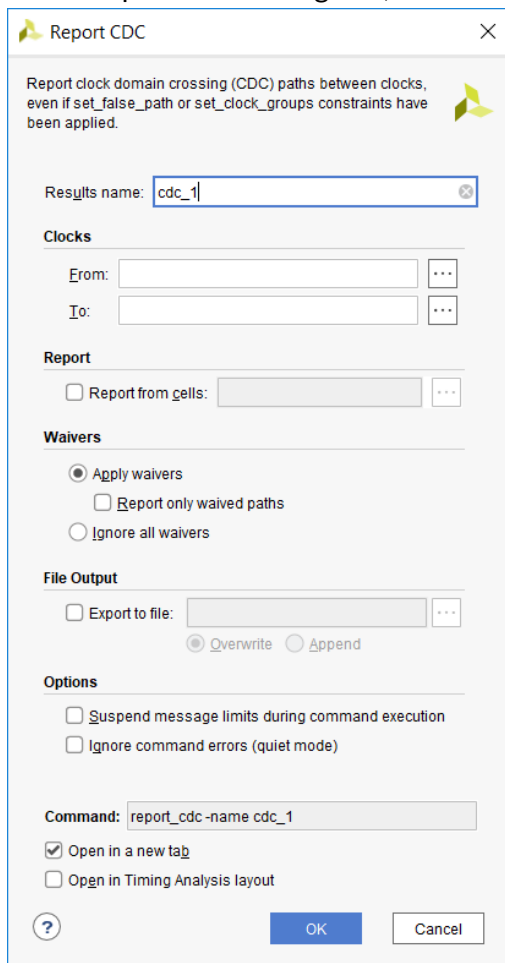


TIP: You can disregard the critical warnings about the unbounded GT locations.

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.

The screenshot shows the 'Summary (by clock pair)' section of the Timing console. The table lists various clock pairs with their severity, source and destination clocks, CDC types, exceptions, and counts for different CDC categories.

Severity	Source Clock	Destination Clock	CDC Type	Exceptions	Endpoints	Safe	Unsafe	Unknown	No ASYNC_REG
Critical	my_ip_glblck	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_glblck	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_glblck	my_ip_drpclk	No Common Primary Clock	False Path	6	6	0	0	0
Info	my_ip_drpclk	my_ip_glblck	No Common Primary Clock	False Path	2	2	0	0	0

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

The screenshot shows the 'Summary (by type)' section of the Timing console. The table lists CDC violations by their severity, ID, count, and description.

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

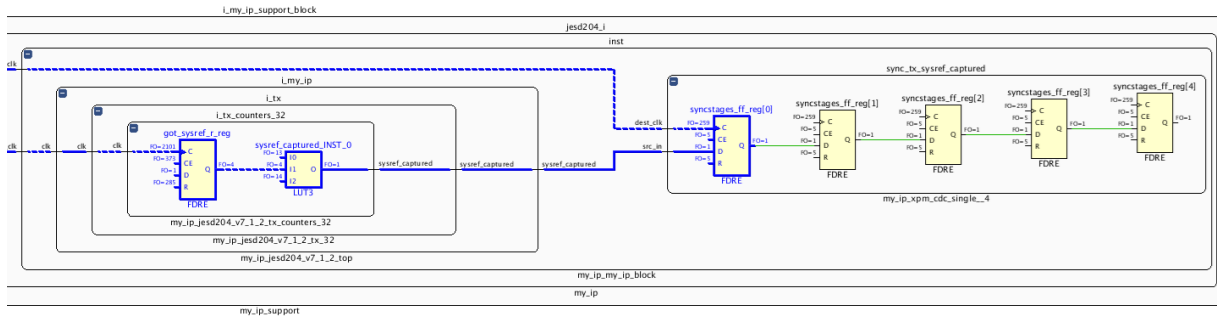
The `my_ip_glblck` to `my_ip_axi_aclk` clock pair includes one Critical CDC-10 violation due to combinational logic on the CDC path. This step covers how to waive the CDC-10 violation.

The screenshot shows the details for a CDC-10 violation. The table lists the severity, ID, description, depth, exception, source, destination, and category.

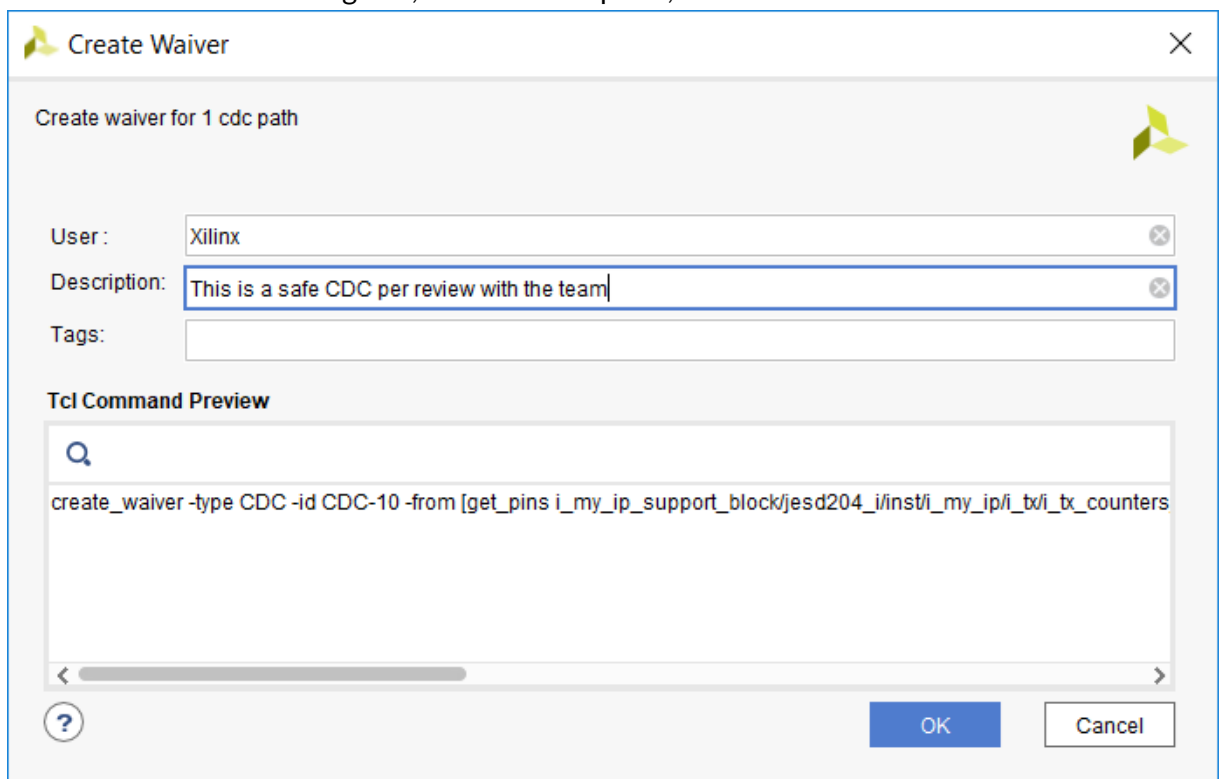
Severity	ID	Description	Depth	Exception	Source (From)	Destination (To)	Category
Critical	CDC-10	Combinational logic detected before a synchronizer	5	False Path	I_my_ip_supp...sref_r_reg/C	I_my_ip_supp...s_ff_reg[0]/D	Unsafe
Info	CDC-3	1-bit synchronized with ASYNC_REG property	5	False Path	I_my_ip_supp...ot_sync_reg/C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



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

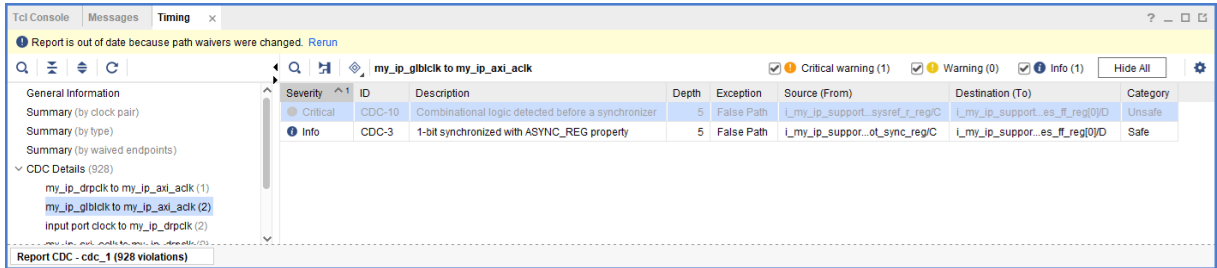


★ IMPORTANT! A waiver tracks the date the waiver was added, the user that added the waiver, and a description of why the violation was waived. The date is automatically added by the system. The Tags field is an optional description or list of keywords that can be used for documentation purposes.

- 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.

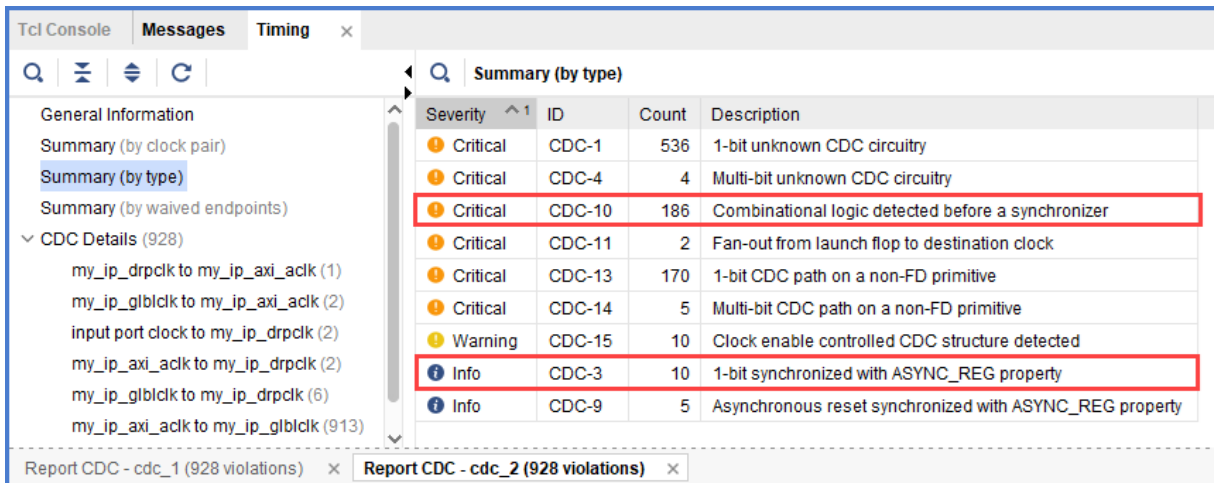
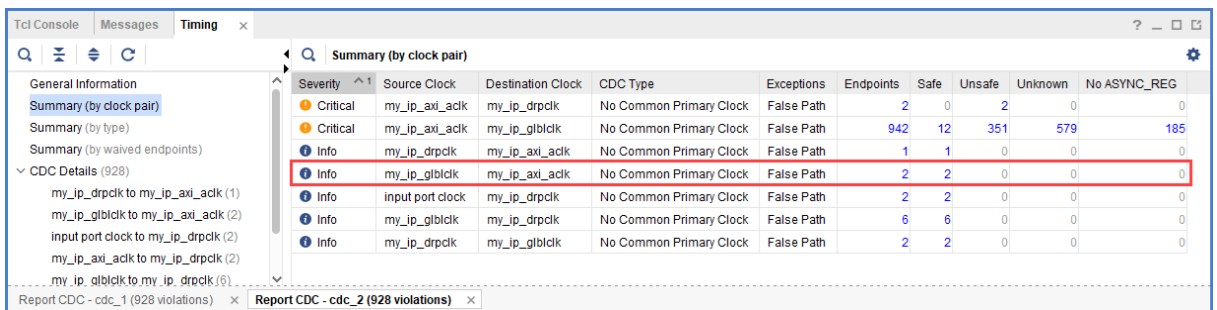


- To see the impact of the CDC-10 waiver, select **Reports** → **Timing** → **Report CDC** to rerun Report CDC.

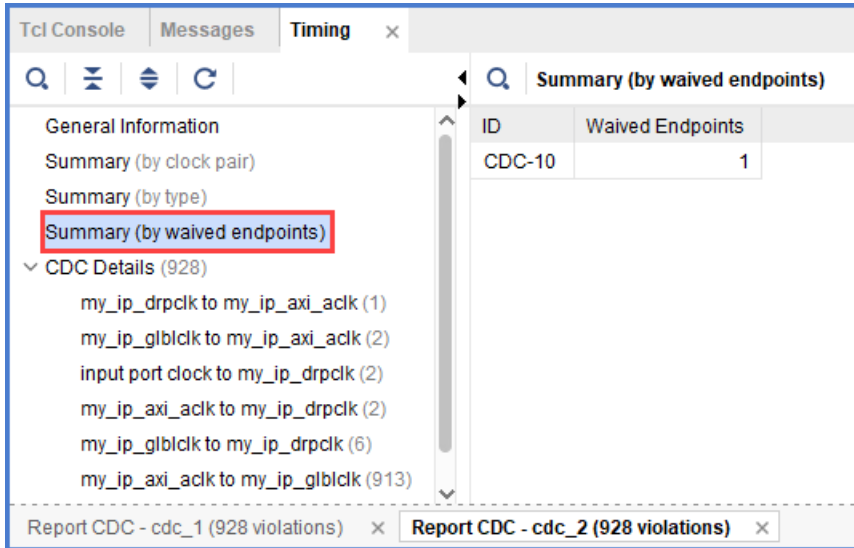
Note: When a waiver is created or deleted, you must rerun Report CDC, Report DRC, or Report Methodology to see the updated results.

- See the CDC Report to view the updated information.

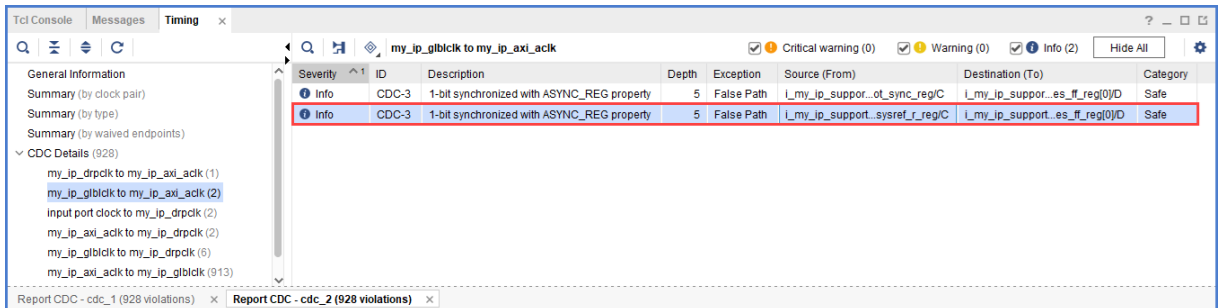
The differences from the previous Summary by clock pair and Summary by type sections are highlighted in red in the following figures.



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

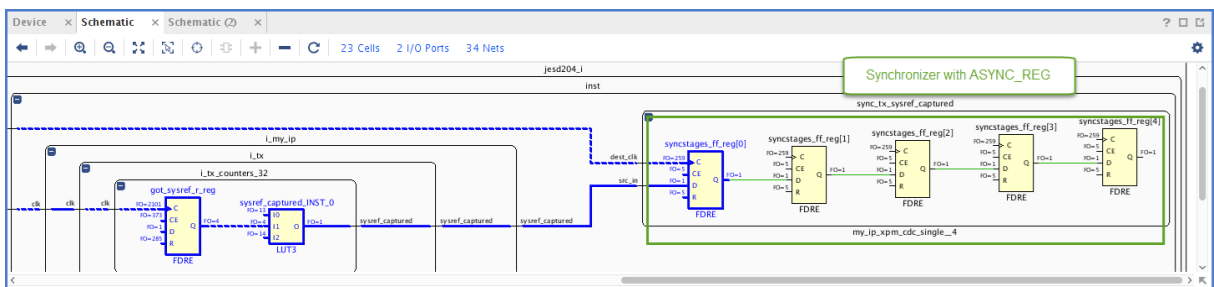



The detailed section for the my_ip_glblclk to my_ip_axi_aclk CDC shows that the Critical CDC-10 was replaced with an Info CDC-3.




7. Select the new CDC-3 row, and click the Schematic toolbar button . 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.

ID	Severity	Count	Description
CDC-10	Critical	1	Combinational logic detected before a synchronizer

ID	Waived Endpoints
CDC-10	1

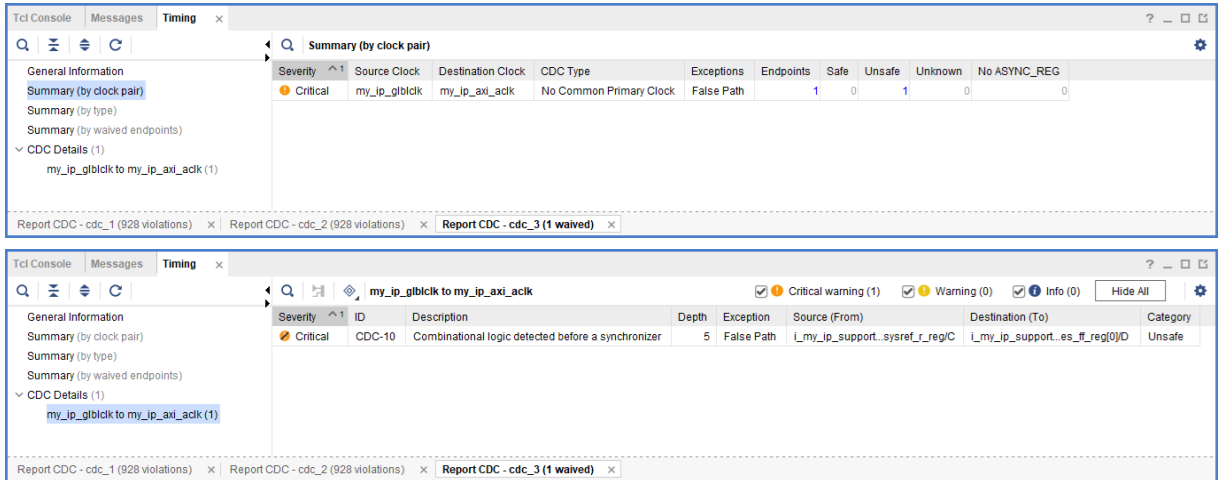
Source Clock: my_ip_glbclk
 Destination Clock: my_ip_axi_aclk
 CDC Type: No Common Primary Clock

Row	ID	Severity	Description	Depth	Level	Destination (To)
1	CDC-10	Critical	Combinational logic detected before a synchronizer	5	Fail	_reg/C 1_my_ip_support_block/jesd204_i/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 🛑.



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.

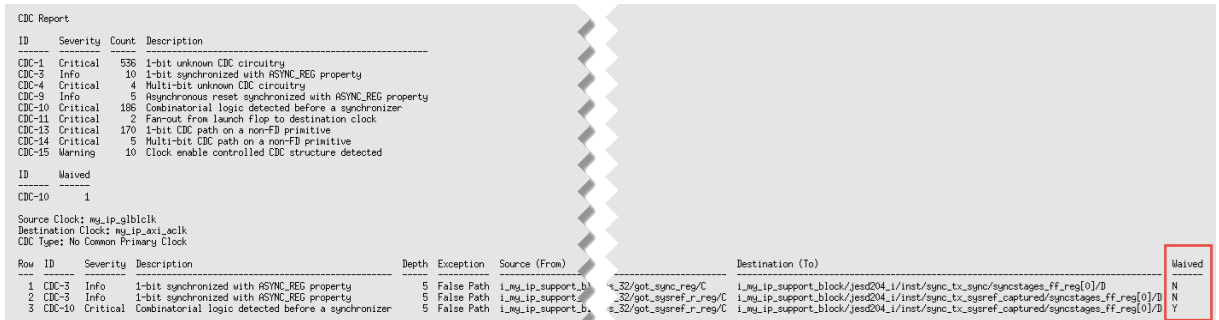
Generating a List of Rules with Waived Violations

1. In the Tcl Console, enter:

```
report_cdc -details -show_waiver
```

2. Verify that the my_ip_glbclk to my_ip_axi_ack 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.



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



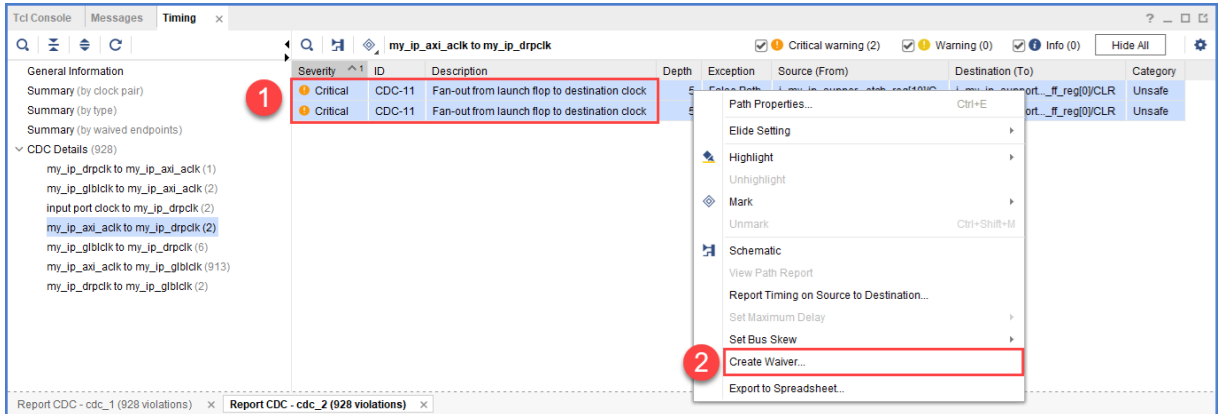
TIP: You can also generate a list of all violations regardless of the waivers from the Vivado IDE. Select **Reports** → **Timing** → **Report CDC**. In the Report CDC dialog box, enable **Ignore all waivers**, and click **OK**.

Step 6: Waiving Multiple CDC Violations

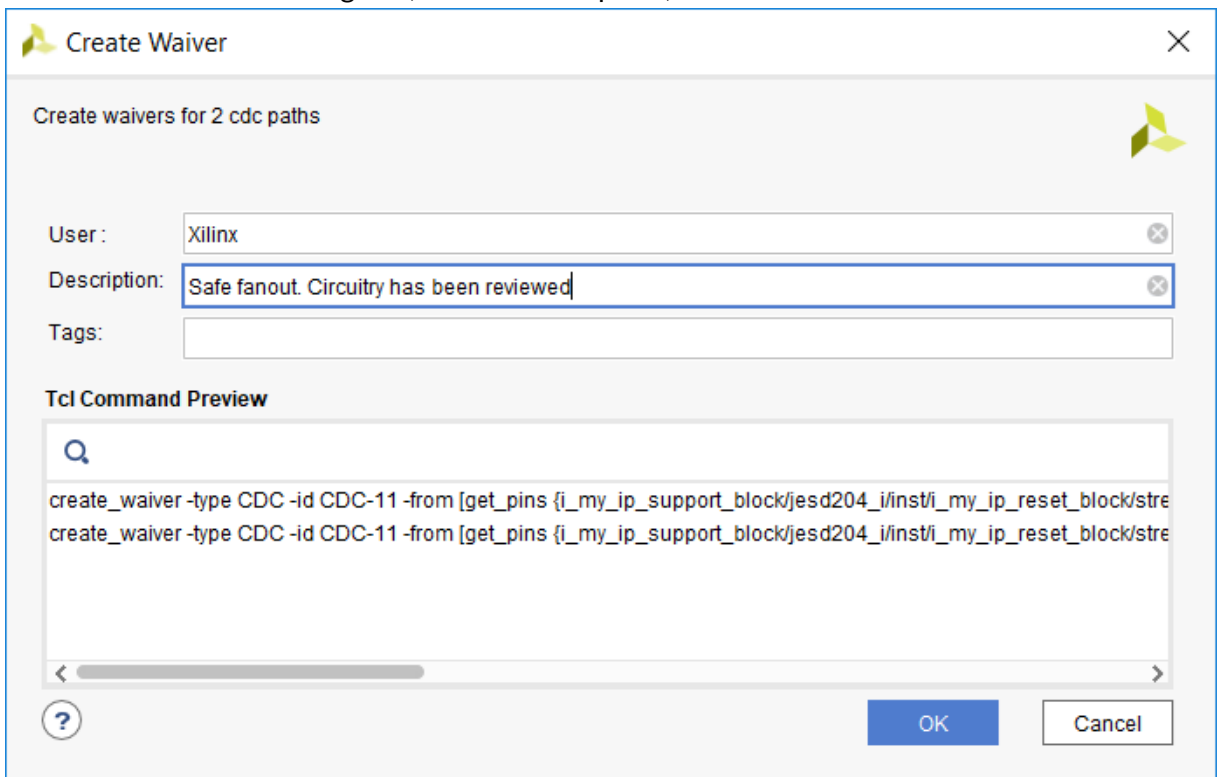
The my_ip_axi_aclk to my_ip_drpclk CDC includes two Critical CDC-11 violations. This step covers how to waive both CDC-11 violations simultaneously.

Severity	ID	Description	Depth	Exception	Source (From)	Destination (To)	Category
Critical	CDC-11	Fan-out from launch flop to destination clock	5	False Path	l_my_ip_support_etch_reg10jC	l_my_ip_support_etch_reg10jC	Unsafe
Critical	CDC-11	Fan-out from launch flop to destination clock	5	False Path	l_my_ip_support_etch_reg10jC	l_my_ip_support_etch_reg10jC	Unsafe

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**.



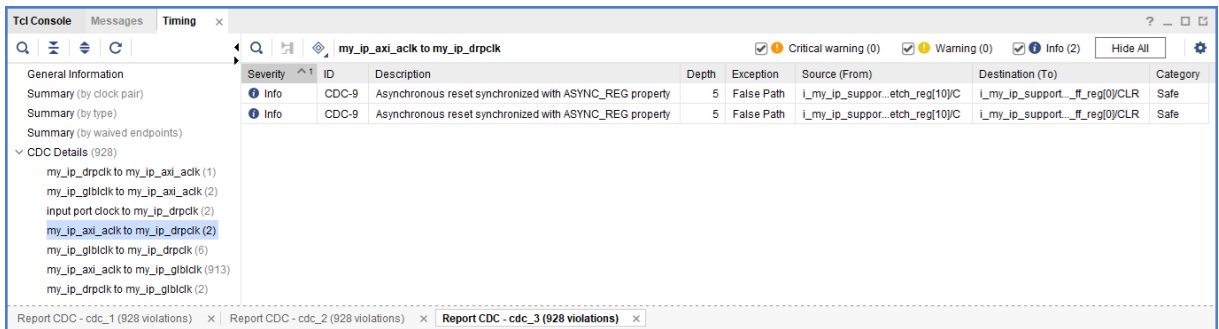
In the Timing Report, the two selected rows are disabled when the waivers are created.

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

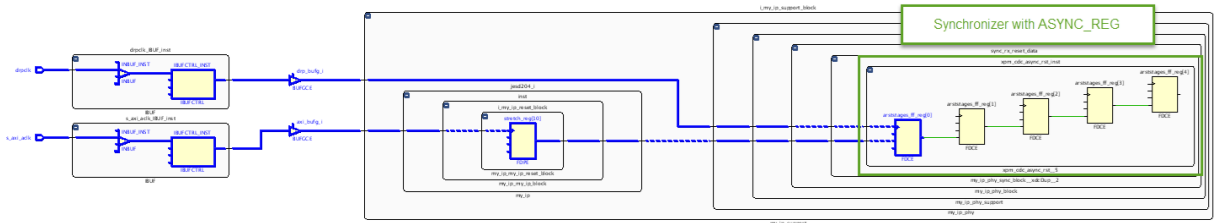
Severity	ID	Description	Depth	Exception	Source (From)	Destination (To)	Category
Critical	CDC-11	Fan-out from launch flop to destination clock	5	False Path	i_my_ip_supp...tch_reg[10]/C	i_my_ip_supp...ff_reg[0]/CLR	Unsafe
Critical	CDC-11	Fan-out from launch flop to destination clock	5	False Path	i_my_ip_supp...tch_reg[10]/C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

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

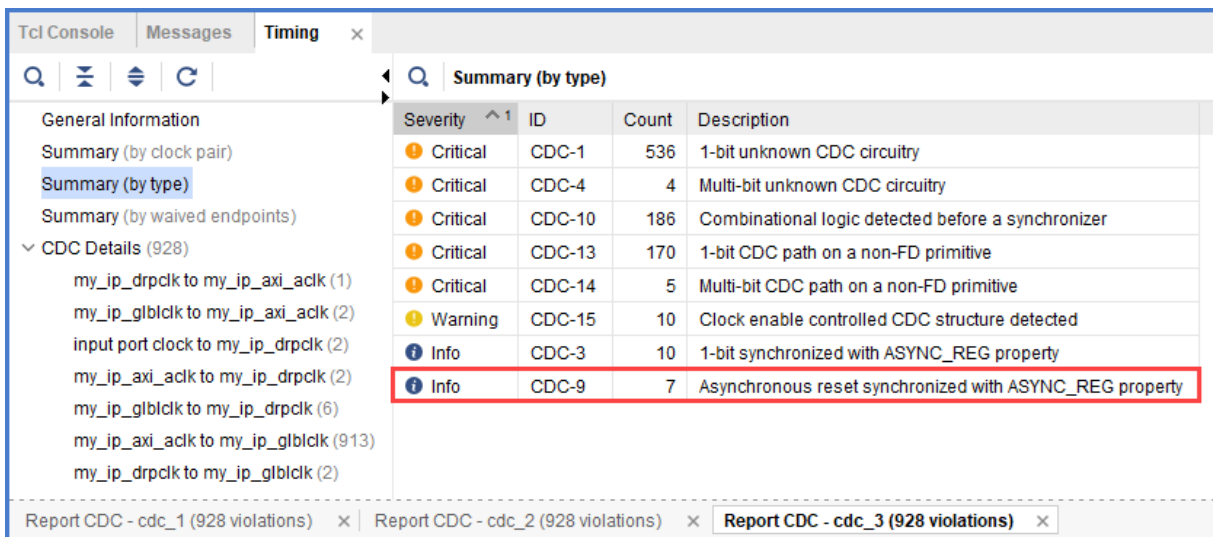


- To view a schematic of the violation, select the CDC-9 row in the CDC Report, and click the Schematic toolbar button
- Verify that there is a 5-level synchronizer on the destination clock domain.



- 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.



- 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.

The screenshot shows the 'Timing' window in Vivado IDE. On the left, there is a navigation pane with options: 'General Information', 'Summary (by clock pair)', 'Summary (by type)', 'Summary (by waived endpoints)' (which is highlighted), and 'CDC Details (928)'. On the right, the 'Summary (by waived endpoints)' table is displayed:

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.

The screenshot shows the output of the `report_cdc -details -waived` command. It includes a summary table and detailed violation information.

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

Row	ID	Severity	Description	Depth	Exception	Source (From)	Destination (To)
1	CDC-10	Critical	Combinatorial logic detected before a synchronizer	5	False Path	l_my_ip_support_block/reg32/get_sigref_r_reg/C	l_my_ip_support_block/jesd204_lv1/inst/sync_tx_sigref_captured/syncstages_ff_reg[0]/D

Row	ID	Severity	Description	Depth	Exception	Source (From)	Destination (To)
1	CDC-11	Critical	Fan-out from Launch Flop to destination clock	5	False Path	l_my_ip_support_block/reg10/C	l_my_ip_support_block/l_jesd204_phy/inst/jesd204_phy_block_l1/sync_rx_reset_data/spm_cdc_async_rst_inst/arststages_ff_reg[0]/CLR
2	CDC-11	Critical	Fan-out from Launch Flop to destination clock	5	False Path	l_my_ip_support_block/reg10/C	l_my_ip_support_block/l_jesd204_phy/inst/jesd204_phy_block_l1/sync_tx_reset_data/spm_cdc_async_rst_inst/arststages_ff_reg[0]/CLR

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_glblclk` 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	i_my_ip_support...modes_reg[1]/C	i_my_ip_supp...d_ris_r_reg[0]	Safe
Critical	CDC-14	Multi-bit CDC path on a non-FD primitive	0	False Path	i_my_ip_support...odes_reg[2:1]/C	i_my_ip_supp.../TXDATA[2:1]	Unknown
Critical	CDC-14	Multi-bit CDC path on a non-FD primitive	0	False Path	i_my_ip_support...odes_reg[2:1]/C	i_my_ip_supp.../TXDATA[2:1]	Unknown
Critical	CDC-14	Multi-bit CDC path on a non-FD primitive	0	False Path	i_my_ip_support...odes_reg[2:1]/C	i_my_ip_supp.../TXDATA[2:1]	Unknown
Critical	CDC-14	Multi-bit CDC path on a non-FD primitive	0	False Path	i_my_ip_support...odes_reg[2:1]/C	i_my_ip_supp.../TXDATA[2:1]	Unknown
Critical	CDC-13	1-bit CDC path on a non-FD primitive	0	False Path	i_my_ip_support...modes_reg[2]/C	i_my_ip_supp.../TXCTRL[2:0]	Unsafe
Critical	CDC-13	1-bit CDC path on a non-FD primitive	0	False Path	i_my_ip_support...modes_reg[2]/C	i_my_ip_supp.../TXCTRL[2:1]	Unsafe
Critical	CDC-13	1-bit CDC path on a non-FD primitive	0	False Path	i_my_ip_support...modes_reg[2]/C	i_my_ip_supp.../TXCTRL[2:3]	Unsafe
Critical	CDC-13	1-bit CDC path on a non-FD primitive	0	False Path	i_my_ip_support...modes_reg[1]/C	i_my_ip_supp.../TXDATA[0]	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.

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
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

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	Combinational 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	Maived
CDC-10	1
CDC-11	2
CDC-14	10

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)
1	CDC-10	Critical	Combinational logic detected before a synchronizer	5	False Path	i_my_ip_support_block/jesd204_i/inst/i_my_ip/tx/tx_counters_32/got

Source Clock: my_ip_axi_aclk
Destination Clock: my_ip_drpclk
CDC Type: No Common Primary Clock

Row	ID	Severity	Description	Depth	Exception	Source (From)
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
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]/C

Source Clock: my_ip_axi_aclk
Destination Clock: my_ip_glblclk
CDC Type: No Common Primary Clock

Row	ID	Severity	Description	Depth	Exception	Source (From)	Destination
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_suppr
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_suppr
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_suppr
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_suppr
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_suppr

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 cdc -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_i/inst/i_my_ip/i_tx/i_tx_counters_32/get_pins {i_my_ip_support_block/jesd204_i/inst/i_my_ip_reset_block/stretch_reg[10]}]}
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]}]}
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]}]}
create_waiver -type CDC -id {CDC-14} -user "Xilinx" -desc "No more CDC-14!" -from [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[1]/C}]]
current_instance -quiet
```

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

```
read_xdc waivers.xdc
```

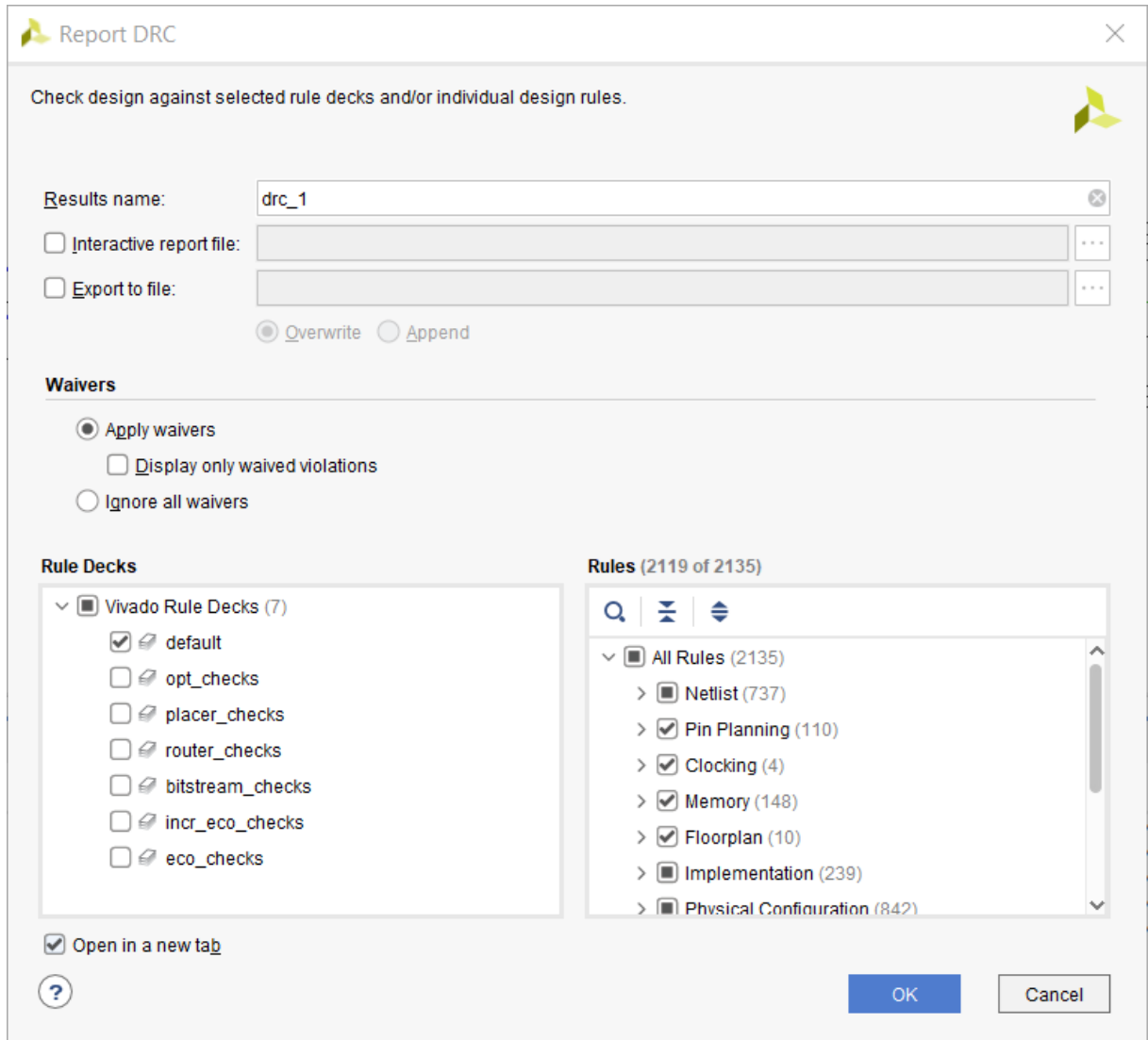
The following warnings show that duplicate waivers were not added to the existing waivers. Only waivers that are exact duplicates of existing waivers are rejected.

```
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

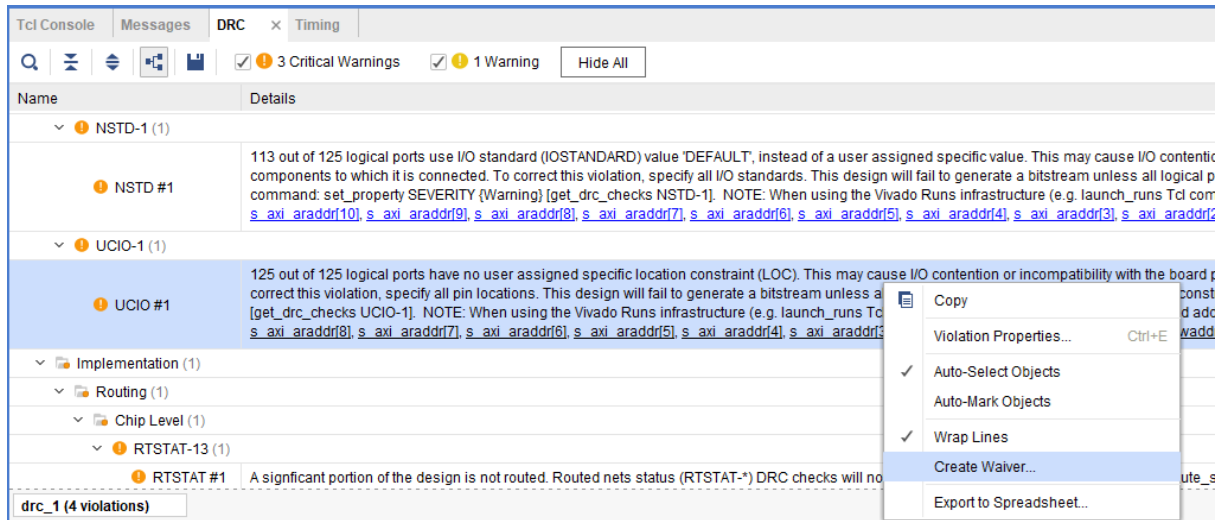
In this step, you waive multiple DRC violations simultaneously.

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

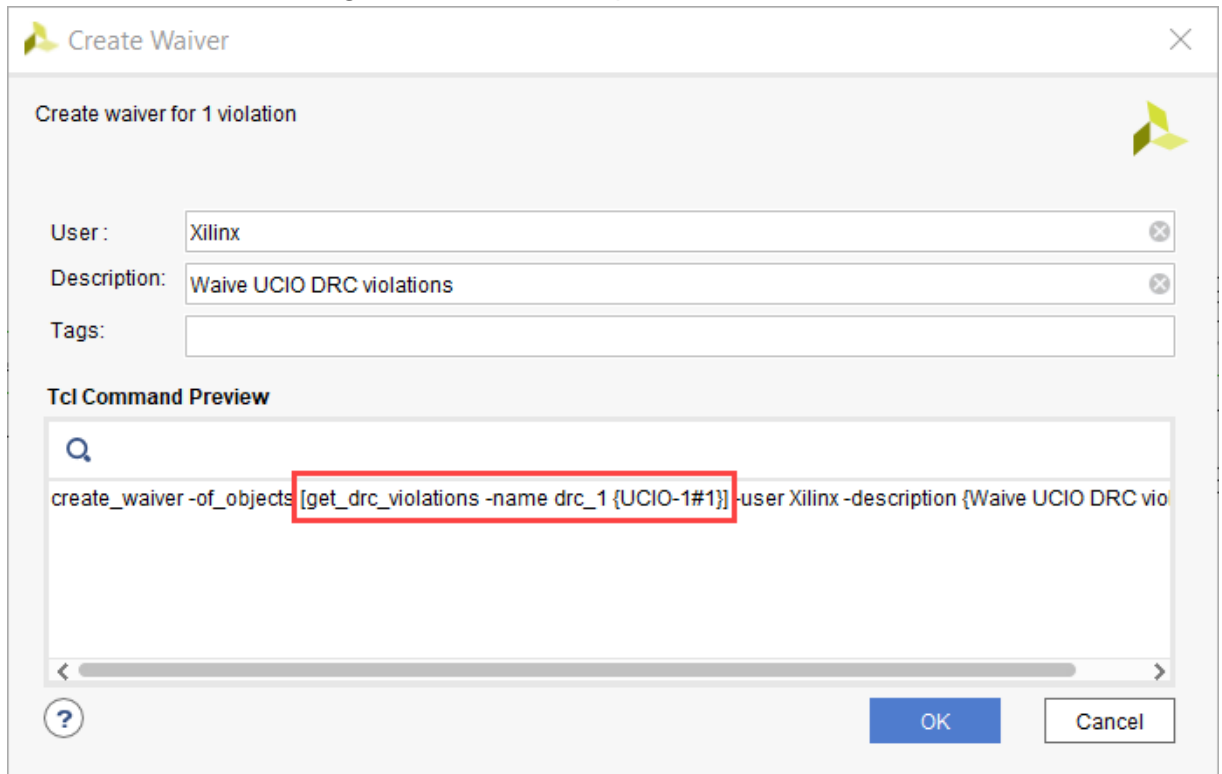


- In the DRC Report, right-click **UCIO#1**, and select **Create Waiver** to create a waiver for the UCIO-1 violations.

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.

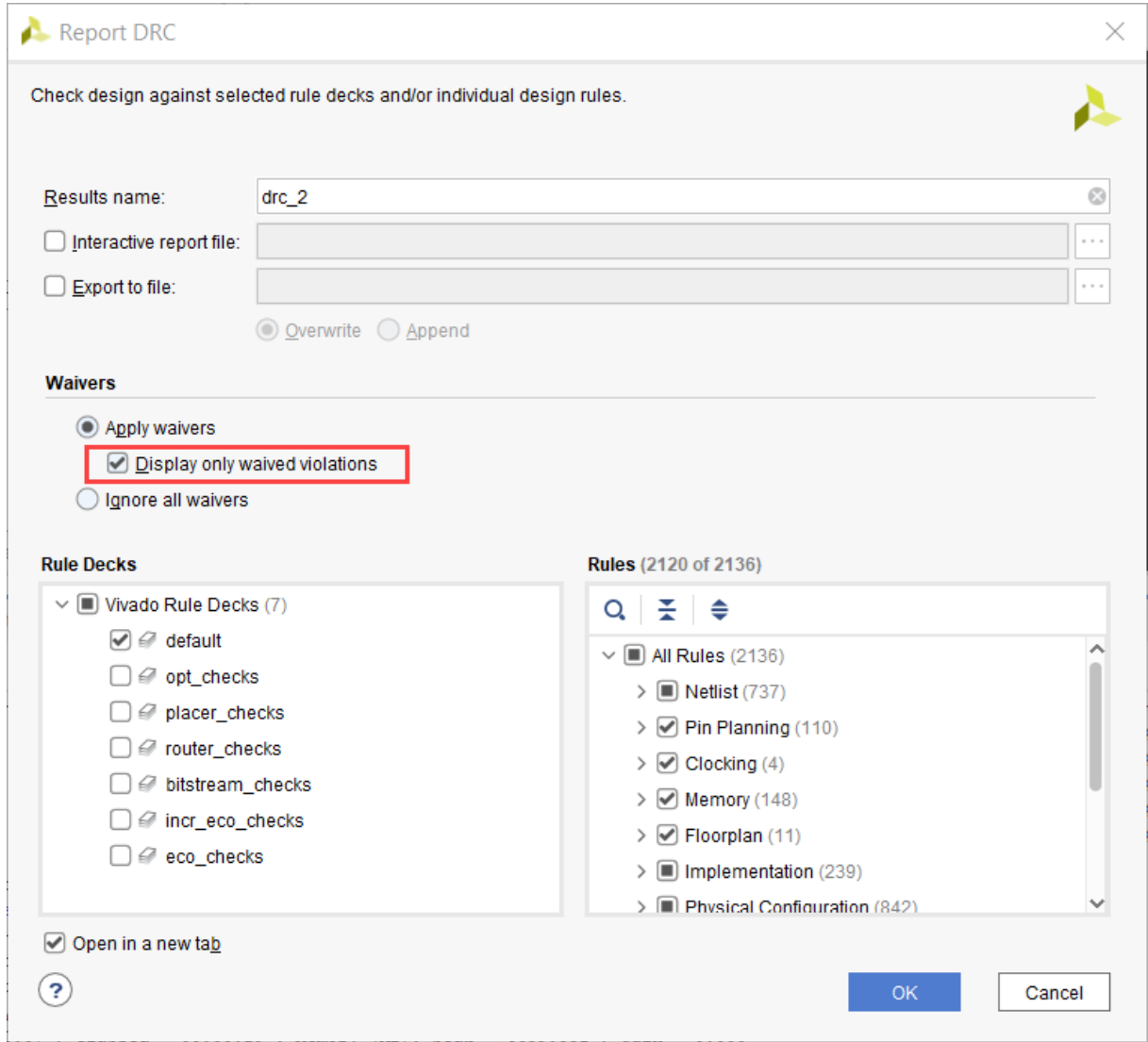
```
#
# WRITE DRC WAIVERS
# cmd: write_waivers -type DRC drc_waivers.xdc
current_instance -quiet
create_waiver -type DRC -id {UCIO-1} -user "Xilinx" -desc "Waive UCIO DRC violations" -objects [get_ports { refclk0p glibclkp refclk0n tx_start_of_frame[3] tx_start_of_multiframe[3] glibclkn
tx_reset drpclk tx_start_of_multiframe[2] txp[3] tx_start_of_multiframe[0] tx_start_of_frame[2] tx_start_of_frame[1] s_axi_rready tx_sync tx_sysref tx_aresetn s_axi_rdata[1] s_axi_rvalid
s_axi_rresp[0] s_axi_rresp[1] s_axi_rdata[0] s_axi_rdata[2] s_axi_rdata[3] s_axi_rdata[4] s_axi_rdata[9] s_axi_rdata[5] s_axi_rdata[10] s_axi_rdata[6] s_axi_rdata[7] s_axi_rdata[8]
s_axi_rdata[15] s_axi_rdata[11] s_axi_rdata[12] s_axi_rdata[13] s_axi_rdata[16] s_axi_rdata[17] s_axi_rdata[18] s_axi_rdata[14] s_axi_rdata[21] s_axi_rdata[22] s_axi_rdata[27]
s_axi_rdata[23] s_axi_rdata[19] s_axi_rdata[25] s_axi_rdata[26] s_axi_rready s_axi_rdata[28] s_axi_rdata[24] s_axi_rdata[30] s_axi_rdata[31] s_axi_araddr[12] s_axi_arvalid s_axi_rdata[29]
s_axi_araddr[7] s_axi_araddr[3] s_axi_araddr[4] s_axi_araddr[5] s_axi_araddr[6] s_axi_bvalid s_axi_araddr[8] s_axi_araddr[10] s_axi_bready s_axi_ustrb[0] s_axi_ustrb[1] s_axi_araddr[8]
s_axi_bresp[1] s_axi_rready s_axi_wvalid s_axi_wdata[0] s_axi_bresp[0] s_axi_ustrb[2] s_axi_araddr[11] s_axi_wdata[4] s_axi_wdata[1] s_axi_ustrb[3] s_axi_wdata[2] s_axi_wdata[3]
s_axi_wdata[9] s_axi_wdata[5] s_axi_wdata[6] s_axi_wdata[7] s_axi_wdata[8] s_axi_wdata[14] s_axi_wdata[10] s_axi_wdata[11] s_axi_wdata[12] s_axi_wdata[13] s_axi_wdata[19] s_axi_wdata[15]
s_axi_wdata[16] s_axi_wdata[17] s_axi_wdata[18] s_axi_wdata[24] s_axi_wdata[20] s_axi_wdata[28] s_axi_wdata[26] s_axi_wdata[27] s_axi_wdata[26] s_axi_wdata[26] s_axi_wdata[26] s_axi_wdata[26] s_axi_wdata[26]
tx_start_of_multiframe[1] txp[1] tx_start_of_frame[0] txp[2] txn[1] txn[3] txn[2] s_axi_awaddr[11] txn[0] txp[0] s_axi_araddr[1] s_axi_araddr[2] s_axi_araddr[3] s_axi_araddr[4] } ] -strings { "*" } -strings { "*" } -timestamp "Wed Mar 14 22:57:14 GMT 2018" #1
```

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

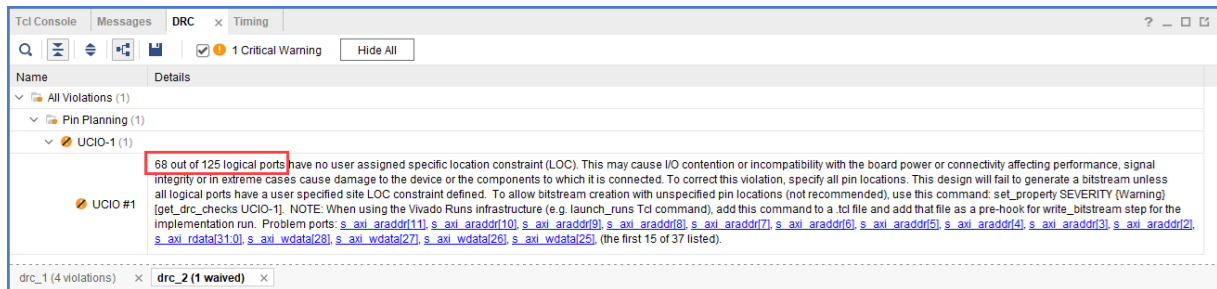
```
delete_waivers [get_waivers -type drc]
create_waiver -type DRC -id {UCIO-1} -user "Xilinx" -desc "Waive
selected UCIO violations" -objects [get_ports { s_axi_rdata[*]
s_axi_wdata[*] s_axi_araddr[*] } ] -strings { "*" } -strings { "*" }
```

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 (METHODOLOGY: no waivers)
4. REPORT DETAILS (CDC)

-----
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  Set Waivers
-----
UCIO-1*      Critical Warning  Unconstrained Logical Port  125         57              68           1             1

-----
4. REPORT DETAILS (CDC)
-----
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
2. REPORT DETAILS (DRC)
3. REPORT DETAILS (METHODOLOGY: no waivers)
4. REPORT DETAILS (CDC)

-----
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  Set Waivers
-----
UCIO-1*      Critical Warning  Unconstrained Logical Port                    125         57              68           1             1
AVAL-326    Critical Warning  Hard_block_must_have_LOC                      1           1              0            0             0
NSTD-1*     Critical Warning  Unspecified I/O Standard                      113        113             0            0             0
RTSTAT-13   Critical Warning  Insufficient Routing                           1           1              0            0             0

-----
4. REPORT DETAILS (CDC)
-----
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
CDC-1        Critical  1-bit unknown CDC circuitry                       536         536             0            0             0
CDC-3        Info     1-bit synchronized with ASYNC_REG property         9           9              0            0             0
CDC-4        Critical  Multi-bit unknown CDC circuitry                   28          28             0            0             0
CDC-9        Info     Asynchronous reset synchronized with ASYNC_REG property  5           5              0            0             0
CDC-13       Critical  1-bit CDC path on a non-FD primitive              170         170             0            0             0
CDC-15       Warning   Clock enable controlled CDC structure detected     10          10             0            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  Set Waivers
-----
UCIO-1*      Critical Warning  Unconstrained Logical Port                    125         57              68           1             1
AVAL-326    Critical Warning  Hard_block_must_have_LOC                      1           1              0            0             0
NSTD-1*     Critical Warning  Unspecified I/O Standard                      113        113             0            0             0
RTSTAT-13   Critical Warning  Insufficient Routing                           1           1              0            0             0

-----
3. REPORT DETAILS (METHODOLOGY)
-----
Rule          Severity  Description                                     Total Vios  Remaining Vios  Waived Vios  Used Waivers  Set Waivers
-----
AVAL-324     Critical Warning  Hard_block_needs_LOC                          1           1              0            0             0
LUTAR-1*     Warning     LUT Drives async reset alert                 40          40             0            0             0
TIMING-9     Warning     Unknown CDC Logic                             1           1              0            0             0
TIMING-10    Warning     Missing property on synchronizer              1           1              0            0             0
TIMING-18*   Warning     Missing input or output delay                 115         115            0            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

Increasing Design Performance Using Report QoR Suggestions

Introduction

The `report_qor_suggestions` (RQS) command enables the Vivado® Design Suite tools to analyze a design and provide automated solutions for enhancing QoR. Built into this command when using the Vivado IDE is the `report_qor_assessment` (RQA) command. The command can be run on an open design after synthesis or after any stage in the implementation flow.

The RQA command assesses the likelihood that a design will meet its timing requirements and provides a quick summary of design metrics. The RQS command evaluates the design in five key areas and suggests fixes or improvements in these areas. The five areas are utilization, clocking, constraints, congestion, and timing. Recommendations from RQS can take the following forms:

- RQS objects. These can add:
 - Switches to a given command.
 - Properties to a given design object.
 - Implementation strategies customized for the design using machine learning algorithms.
- Text recommendations that require user intervention.

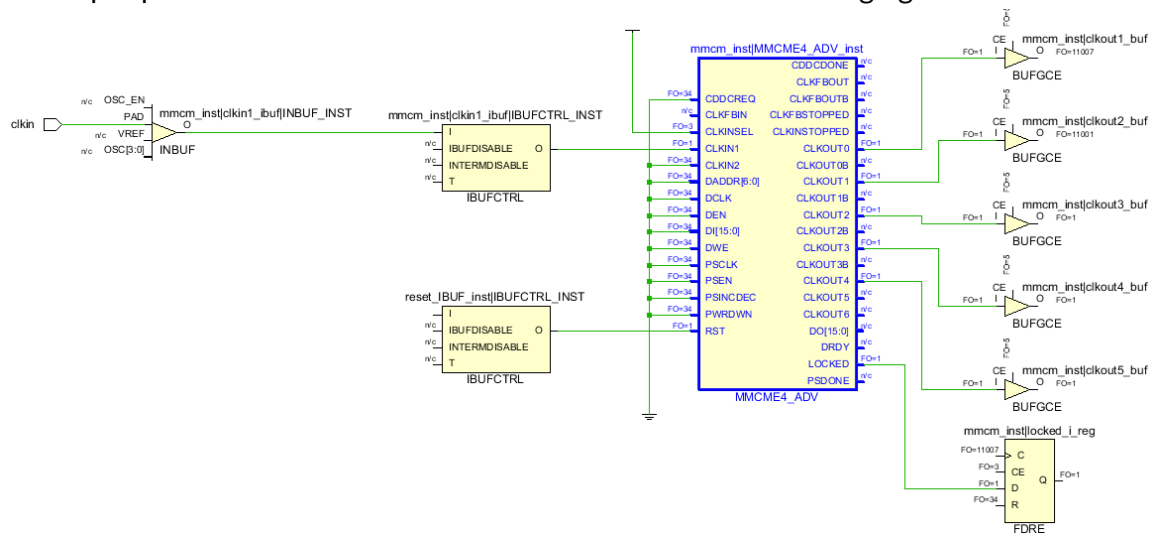
This lab covers how to:

- 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.

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:

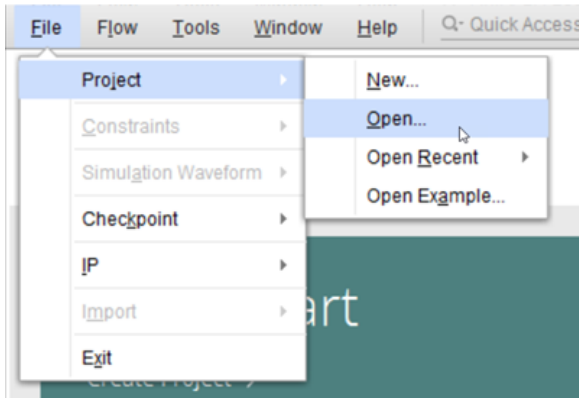
- Clocking Module:** The main clocking circuit for the design resides in `clocking_module.vhd`. For simplicity, RST is tied to GND. LOCKED is registered and tied to an output port. The structure of this block is shown in the following figure.



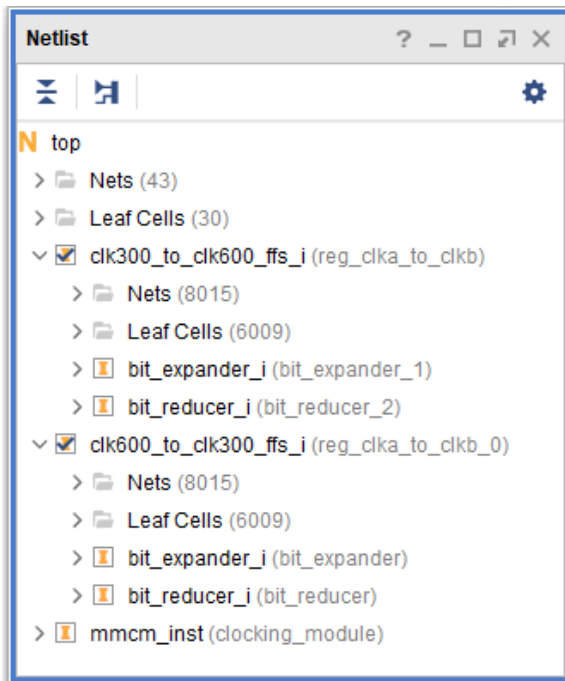
- Reg CLKA to CLKB Module:** This module contains a synchronous CDC for a large bus. It registers input data using CLKA and then passes it to a register on the CLKB domain to be passed to the output. Registering large buses on different related clock domains can impact hold slack (WHS/THS) and setup 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.

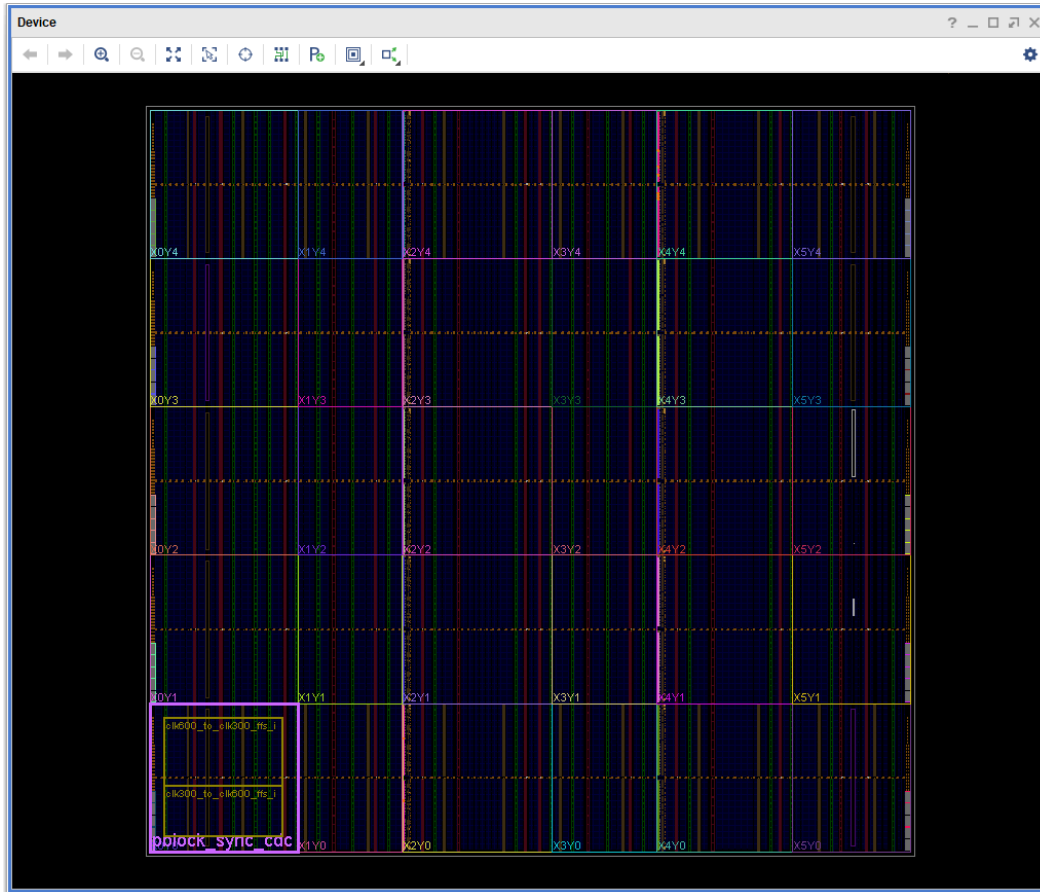
- In the Vivado Design Suite, go to **File** → **Project** → **Open** and select the project located in `<extract_Dir>/Lab2/project_2`.



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.



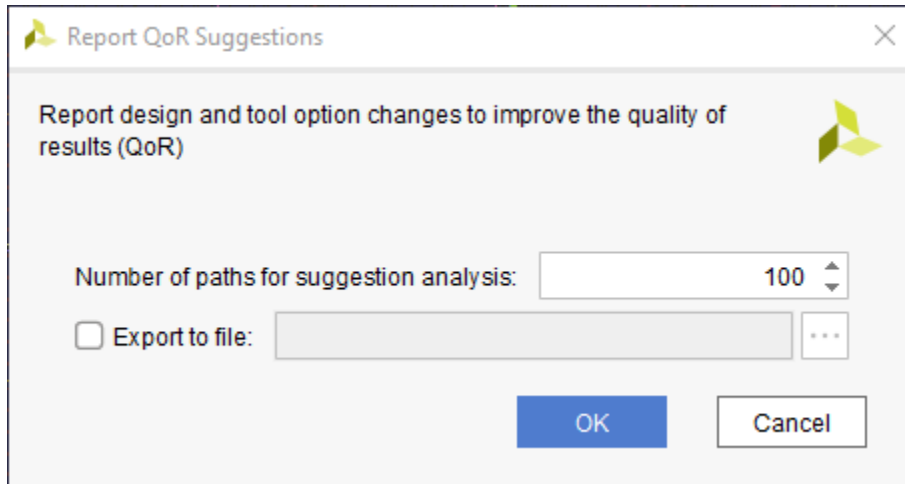
5. In Device view, look at the pblock. This has been added to control placement of the reg_clka_to_clkb modules and force a poor clock skew.



Step 2: Running Report QoR 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.

Built into the running of this command is the `report_qor_assessment` command when run from within the Vivado IDE. This command uses the design metrics and the same timing paths to make an assessment of how likely the design is to meet timing. In Tcl, this command is *not* called automatically, so must be called separately. The equivalent Tcl commands are as follows:

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

The command will:

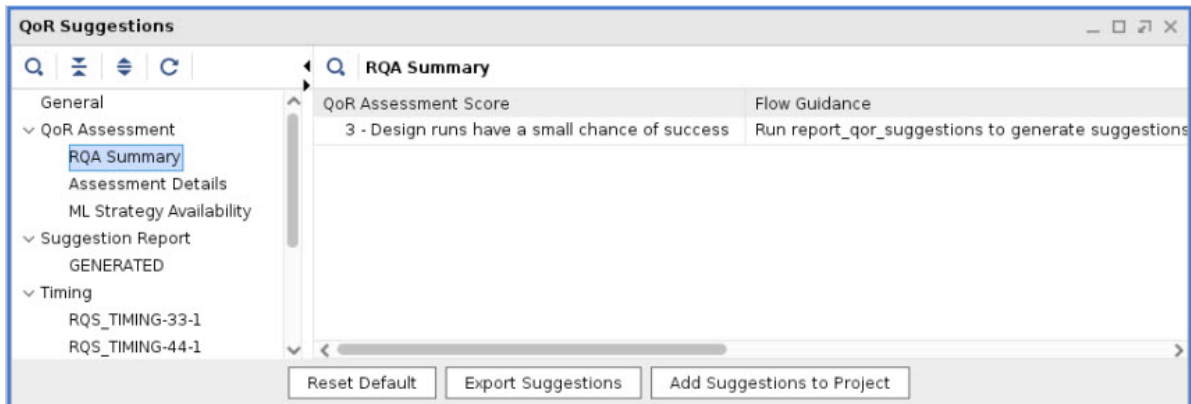
- Generate an assessment report.
- Examine the design and generate new suggestions.
- Generate a report on the suggestions.

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 3: 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.

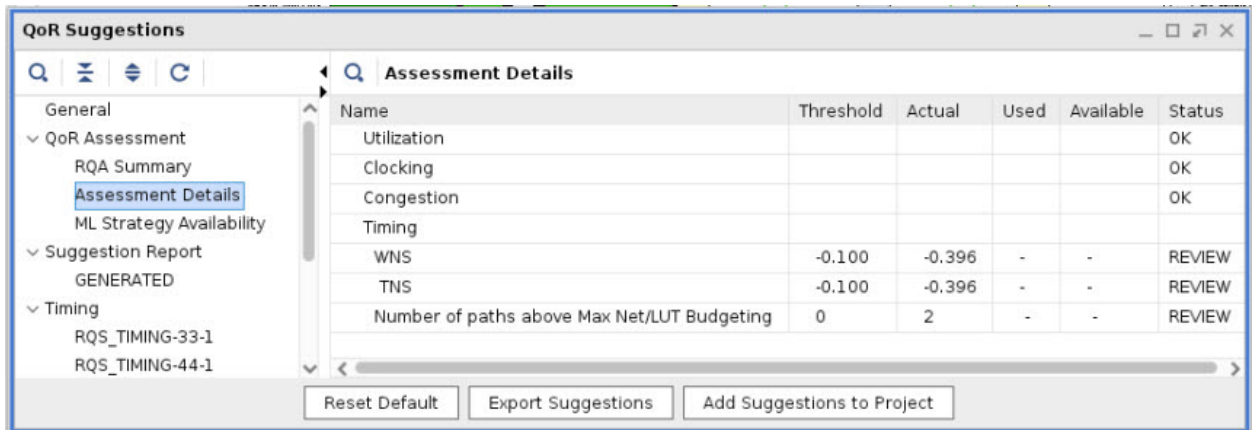
1. The report is broken down into two main sections, the QoR Assessment and the Suggestion Report. All items below the Suggestion Report are also related to suggestions. First click on the **RQA Summary** under the QoR Assessment section of the report. This shows the QoR Assessment Score and the Flow Guidance.



The QoR Assessment score is 3. This means that there is some chance of meeting timing but it is unlikely. The flow guidance is recommending to see if QoR suggestions can improve the design.

Note: The `report_qor_assessment` command runs using the latest data available to it. For example, clock skew is only accurate after logic is placed so less emphasis is given to the clock skew score at the pre-place stage in the implementation flow.

2. Click on the **Assessment Details** section of the report.

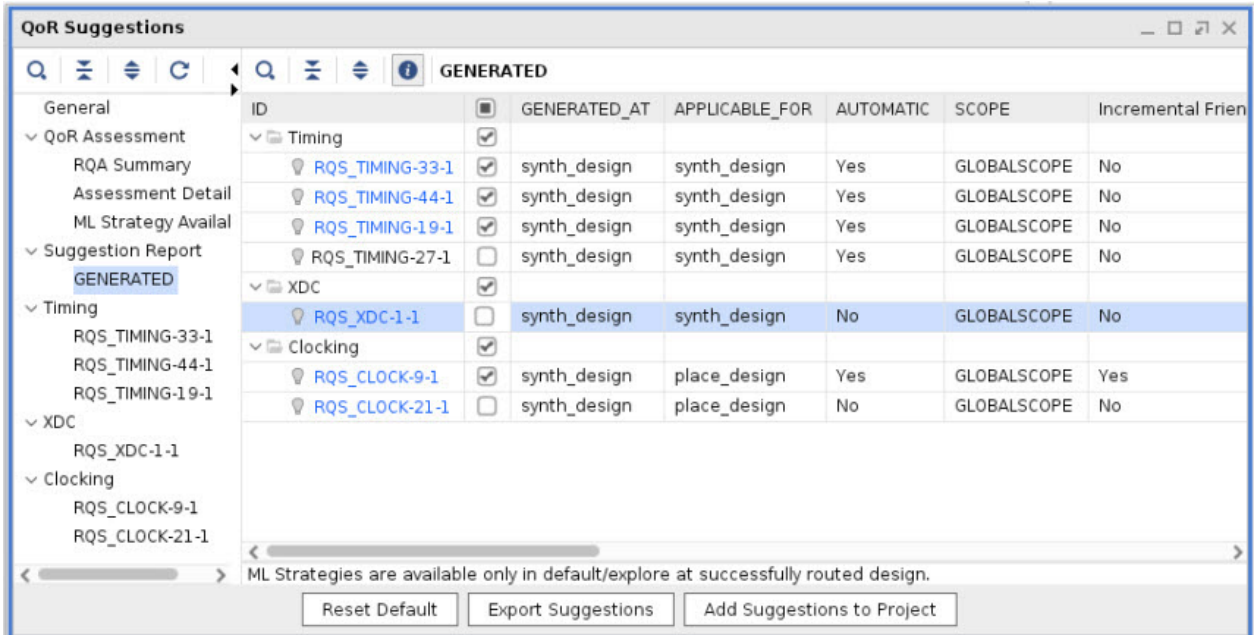


The detailed table shows the categories that have passed the assessment and individual failed metrics. The score is an aggregate of the failed metrics.

The metrics failing are timing based, and all the others are passing. Items marked as OK can be ignored, but items marked REVIEW should be compared against the threshold to see how severe the failure is.

Note: For paths failing Net and LUT budgeting, refer to the QoR suggestion RQS_XDC-1 for more details on the timing paths.

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



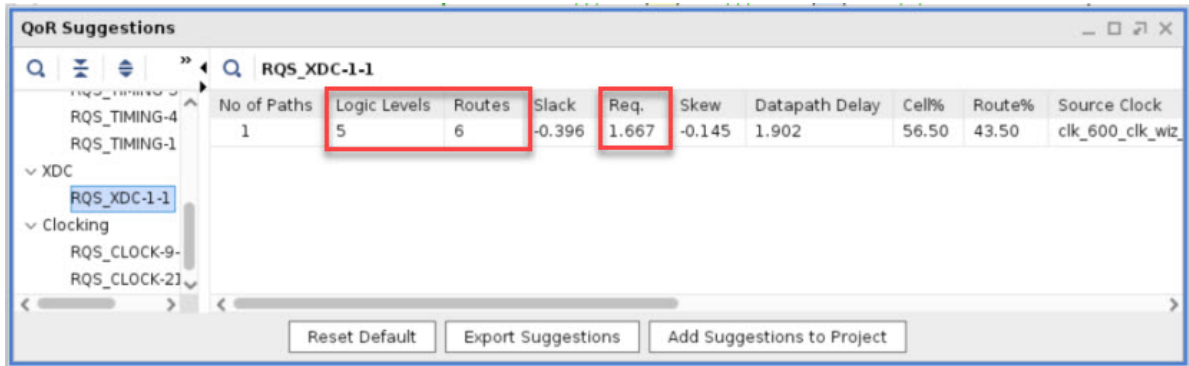
The GENERATED section provides a list of all the suggestions that have been generated at this stage of the current run. Each suggestion has a description that details the reason for the suggestion. Additionally, for each suggestion the following information is provided.

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. Additionally, 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.
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.

Under the other sections of the report there are usually details about the individual suggestions that have been generated.

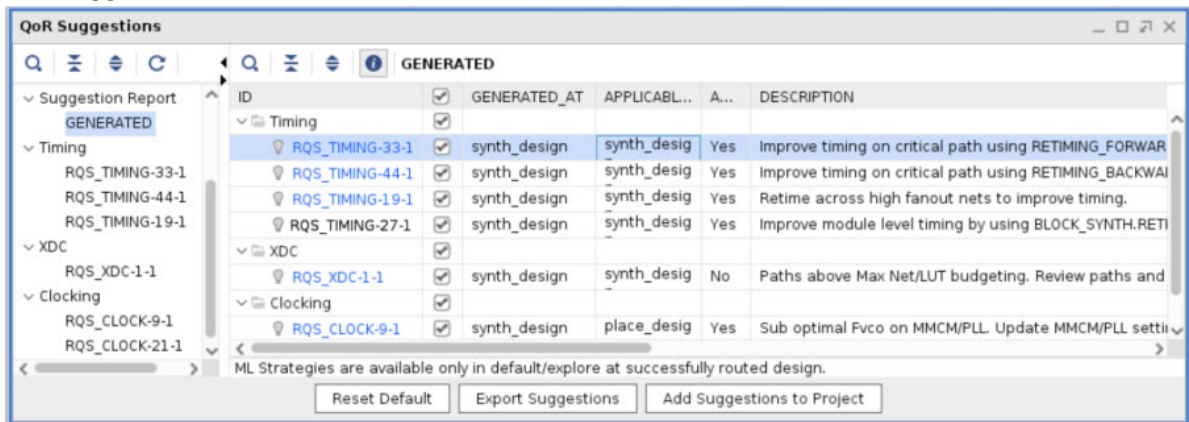
- Click on the **RQS_XDC_1_1** hyperlink. This will take you to the details section for this suggestion.



The suggestion description says that the timing constraint is too tight for the given path(s).

The path has a large negative slack which would stand 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 is a path that will need to be fixed.

- Click on the back arrow button to go back to the GENERATED view.
- In the GENERATED view, click on the **RQS_TIMING-33_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.



When you select the row in the RQS Summary 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 generates a retiming forward property for `synth_design`.

QoR Suggestion Properties

RQS_TIMING-33-1

APPLICABLE_FOR: synth_design

APPLIED:

AUTO:

CATEGORY: Timing

CLASS: qor_suggestion

COMMAND: set_property retiming_forward 1 [get_cells {{clk300_to_clk600_ffs_i/...

DESCRIPTION: Improve timing on critical path using RETIMING_FORWARD property.

ENABLED:

FAILED_TO_APPLY:

FLOW_SUPPORT: Default

General **Properties**

- In the GENERATED view, click on the **RQS_TIMING-33_1** ID to go to the details table for that suggestion. Careful examination of the Endpoint column will confirm that this is the same path that was mentioned for RQS_XDC-1.

QoR Suggestions

RQS_TIMING-33-1

Route	Datapath Delay	Cell%	Route%	Source Clock	Destination Clock	Startpoint	Endpoint
	1.902	56.50	43.50	clk_600_clk_wiz_0	clk_600_clk_wiz_0	clk300_to_clk600_ffs_i/expanded_sig3_reg[1932]C	clk300_to_clk600_ffs_i/bit_reducer_i/tmp_r_regD

Reset Default Export Suggestions Add Suggestions to Project

- In the GENERATED view, you can see the remaining suggestions. The **RQS_CLOCK-9** suggestion is applicable for `place_design`. This is shown in the following figure:

QoR Suggestions

GENERATED

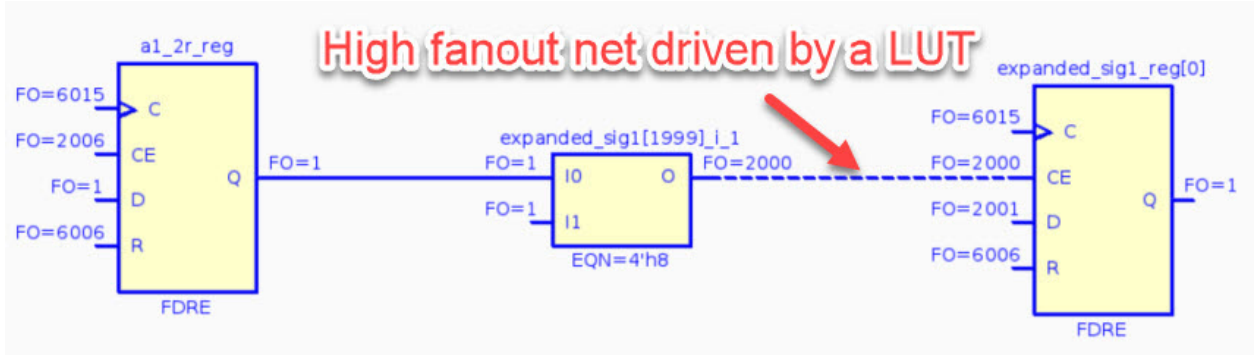
ID	GENERATED_AT	APPLICABLE_FOR	AUTOMATIC	Incre...	DESCRIPTION
RQS_TIMING-33-1	<input checked="" type="checkbox"/>	synth_design	Yes	No	Improve timing on critical path using RETIMING_FORWARD property.
RQS_TIMING-44-1	<input checked="" type="checkbox"/>	synth_design	Yes	No	Improve timing on critical path using RETIMING_BACKWARD property.
RQS_TIMING-27-1	<input checked="" type="checkbox"/>	synth_design	Yes	No	Improve module level timing by using BLOCK_SYNTH.RETIMING property.
RQS_XDC-1-1	<input checked="" type="checkbox"/>	synth_design	No	No	Paths above Max NetLUT budgeting. Review paths and either reduce logic...
RQS_CLOCK-9-1	<input checked="" type="checkbox"/>	synth_design	Yes	Yes	Sub optimal Fvco on MMCM/PLL. Update MMCM/PLL settings to improve the j...

ML Strategies are available only in default/Explore at successfully routed design.

Reset Default Export Suggestions Add Suggestions to Project

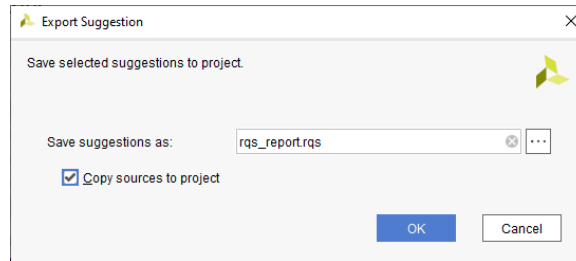
- Click **RQS_TIMING-19-1** and select the timing path. Careful examination of these paths show there is no timing failure on these paths.

10. Press **F4** to load the schematic. You can see this is a high fanout net driven by a LUT.

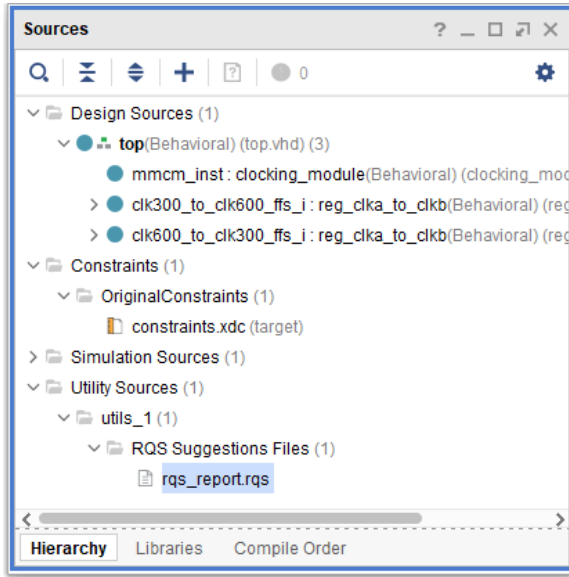


This suggestion is slightly different to the other retiming suggestions. When RQS identifies paths with a difficult profile that it can improve, it attempts to do so regardless of whether there is a timing error.

11. Note RQS_TIMING-27. This 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**. This suggestion applies to the entire module and has overlap with the other suggestions. For clarity, it will not be applied.
12. With the remaining boxes checked, click **Add Suggestions to Project**. When the Export Suggestion dialog box opens, change the file name to **rqs_report.rqs** and select **Copy sources to project** as shown in the following figure.



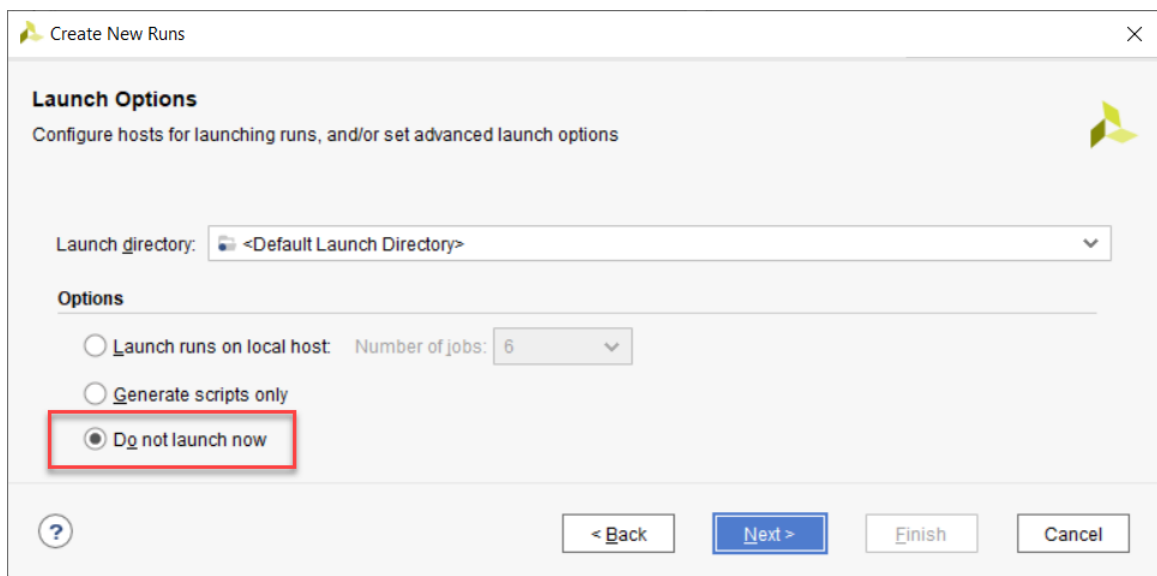
13. Examine the Sources window. You will see that the RQS file has been added to the `utils_1` fileset. This ensures that the file is captured using the `get_files` command, project archives and recognized in the next step when we add the file to a run.



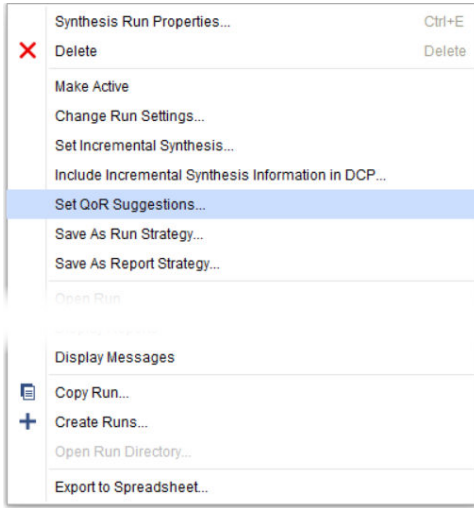
Step 4: Run with Suggestions

You will now add a suggestion to a run and examine what happens when a suggestion is applied and how it is reported.

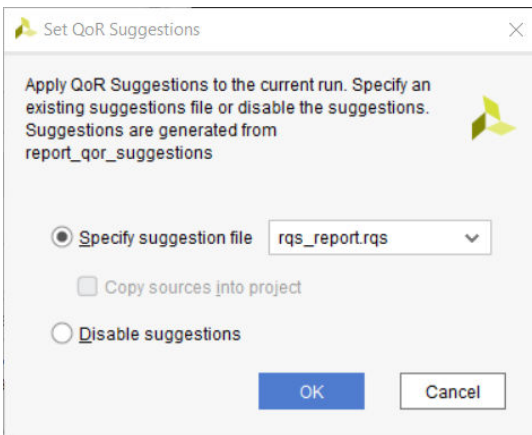
1. In the Design Runs window, click the + icon and generate both a new Synthesis and Implementation run. Ensure both runs have the Default run strategy and in the final step, select **Do not launch now**.



2. In the Design Runs window, right-click on the new **synth_2** run and select **Set QoR Suggestions**

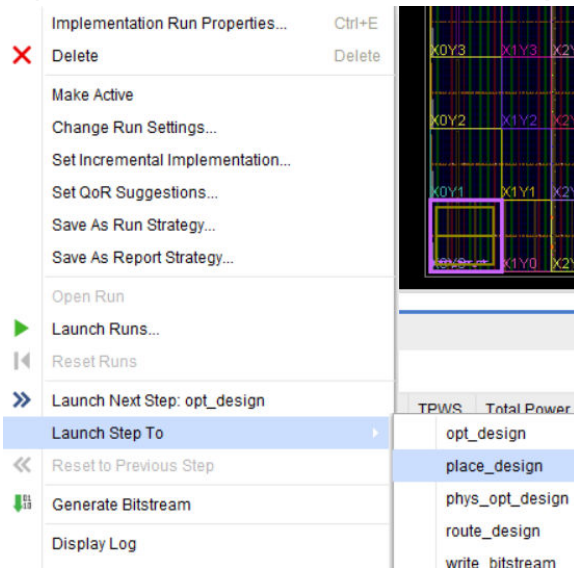


3. Specify the suggestion file as the RQS file added to the project from the previous step and click **OK**.

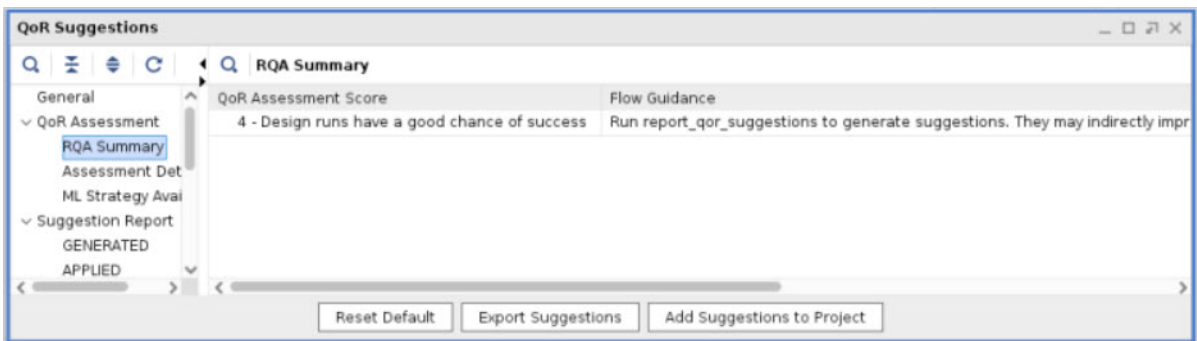


4. Repeat steps 2 and 3 for the implementation run. Specify the same RQS file for each run.
5. In the Design Runs window, right-click on synth_2 and select **Make Active**.
6. In the Flow Navigator, click **Run Synthesis**.

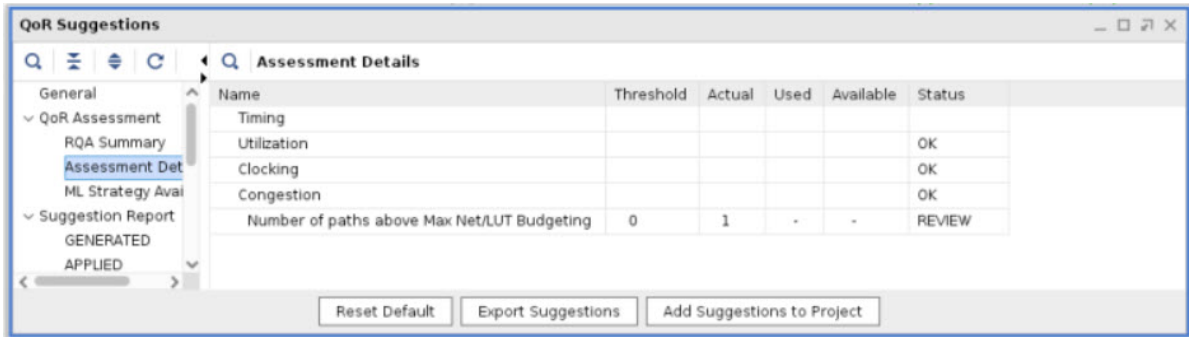
- As this design takes a long time to route, we will only run to place_design and analyze at this stage. When synthesis is complete, in the Design Runs window, right-click on the new implementation run and select **Launch Step To → place_design**.



- With the implementation running, select the Design Runs window and **right click → Open Run** on the synth_2 synthesis run with QoR Suggestions. When opened select **Reports → QoR Suggestions...** and select **Ok** in the dialog box to run the command.
- Click on the **RQA Summary**. You will see that the score has improved from 3 to 4.



- Next click on the **Assessment Details**. You will see that the Net and LUT budget score has been reduced but not eliminated. This is a consequence of the high frequency we are forcing paths to run at in this design:



11. Close the synthesized design.
12. When `place_design` is finished, first go and examine the very top of the implementation log file for the new implementation run. You can see a table summary of the suggestions that have been read in. This summary helps confirm that what is read in is what you expect.

```

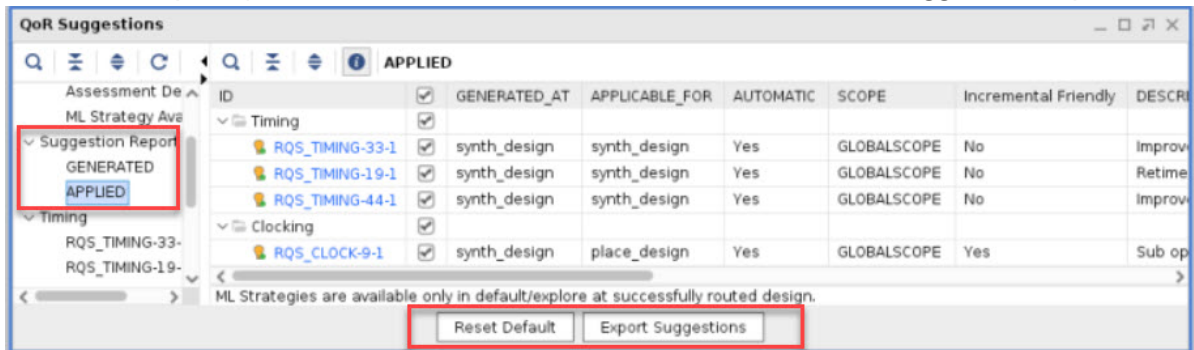
1. Read QOR Suggestions Summary
-----

Read QOR Suggestions Summary
+-----+-----+-----+
|          Suggestion Summary          | Incr Friendly | Total |
+-----+-----+-----+
| Total Number of Suggestions         |               | 5 |
|   Automatic                         |               | 4 |
|   Manual                             |               | 1 |
|   APPLICABLE_AT                     |               |   |
|     synth_design                     |               | 4 |
|     opt_design                       |               | 0 |
|       That overlap with synthesis suggestions |               | 0 |
|     place_design                     |               | 1 |
|     postplace_phys_opt_design        |               | 0 |
|     route_design                     |               | 0 |
|     postroute_phys_opt_design        |               | 0 |
+-----+-----+-----+

```

13. Right-click on the implementation run and select **Open Run Directory**. Next open the checkpoint file by double-clicking on `top_placed.dcp`. This step is necessary as we are examining an intermediate run step in the interests of saving time.
14. In the new instance of Vivado tools, select **Reports → Report QoR Suggestions ...** and click **OK**.

15. In the new report, you will notice that there are more sections under Suggestion Report.



- **GENERATED** - This is where new suggestions are listed
- **EXISTING** - These are suggestions that existed previously but are not applied (Not Shown)
- **APPLIED** - These are where suggestions that have been applied are listed
- **FAILED TO APPLY** - Suggestions where the design objects no longer exist and consequently were not applied (Not shown)

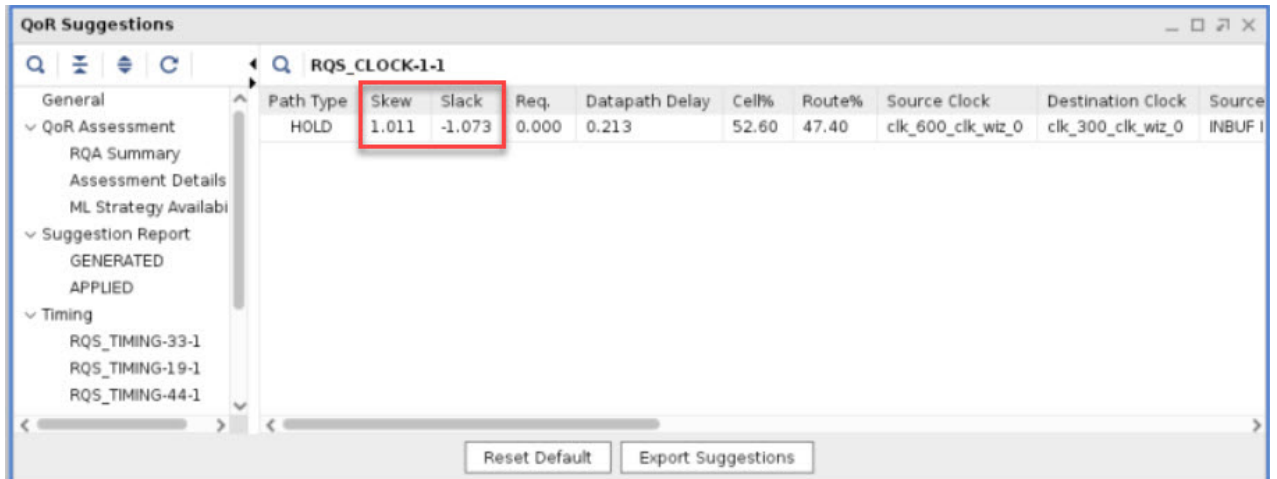
Also note that the option to add to the project is not available when a checkpoint is opened. You can still export a file but it is not added to a project.

16. 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 as some items may have changed.

Step 5: 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 the 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	1.011	-1.073	0.000	0.213	52.60	47.40	clk_600_clk_wiz_0	clk_300_clk_wiz_0	INBUF I

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_qor_suggestions`. 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 **Export Suggestions**. When suggestions are exported, the APPLIED status is reset. All the previous suggestions and the new RQS_CLOCK-1-1 are combined into one file. You can overwrite the previous file and reuse the runs, or create a new file and 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_qor_suggestions` on your own design.

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

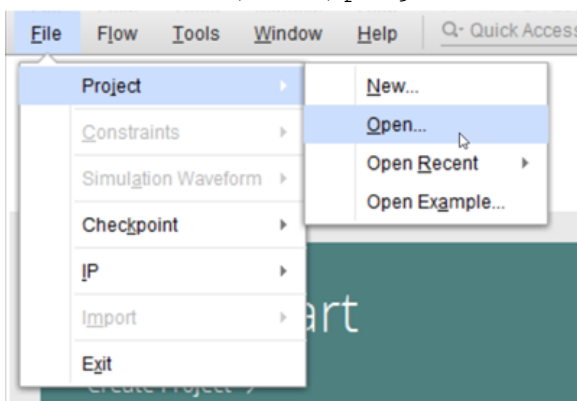
The `report_qor_suggestions` command can generate an implementation design strategy that is predicted to be optimal for the design using machine learning algorithms. In this tutorial, you will look at:

- 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

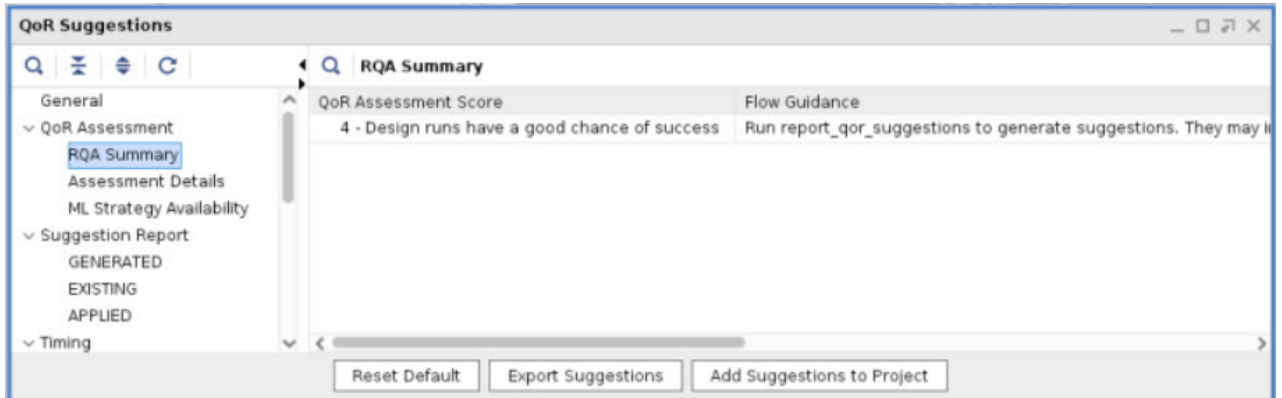
This step shows the process of opening a routed design with QoR Suggestions and generating a new RQS file with strategies. For details on the design, refer to [Step 1: Understanding the Design](#).

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

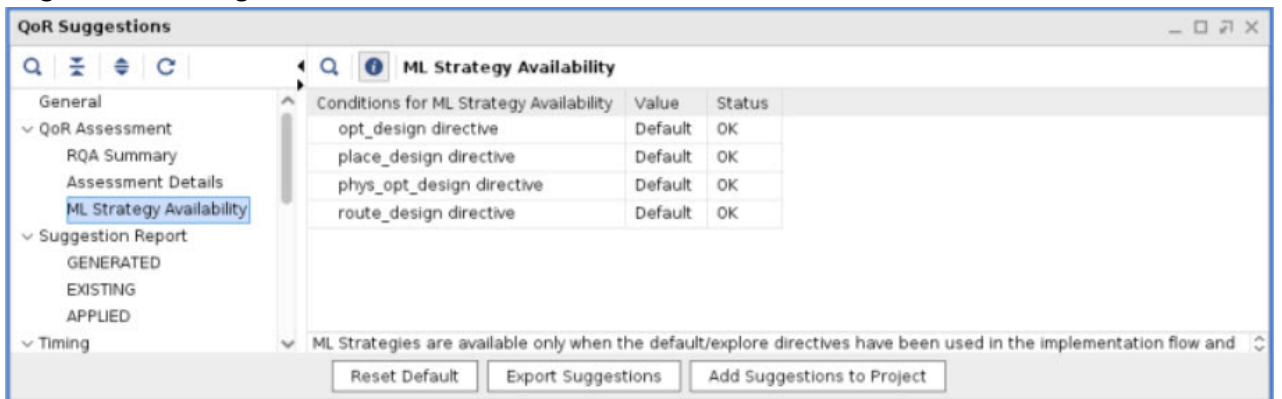


2. In the Flow Navigator, click **Open Implemented Design**.
3. From the pull-down menus, select **Reports** → **Report QoR Suggestions ...**, and click **OK**.

- 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.

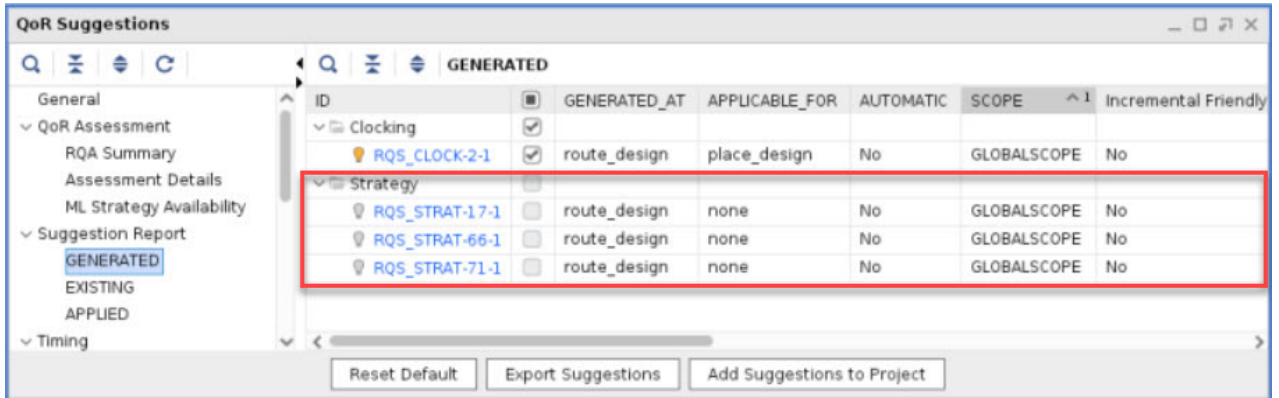


- Click **ML Strategy Availability**. This table details the required directives for the reference run to generate strategies.

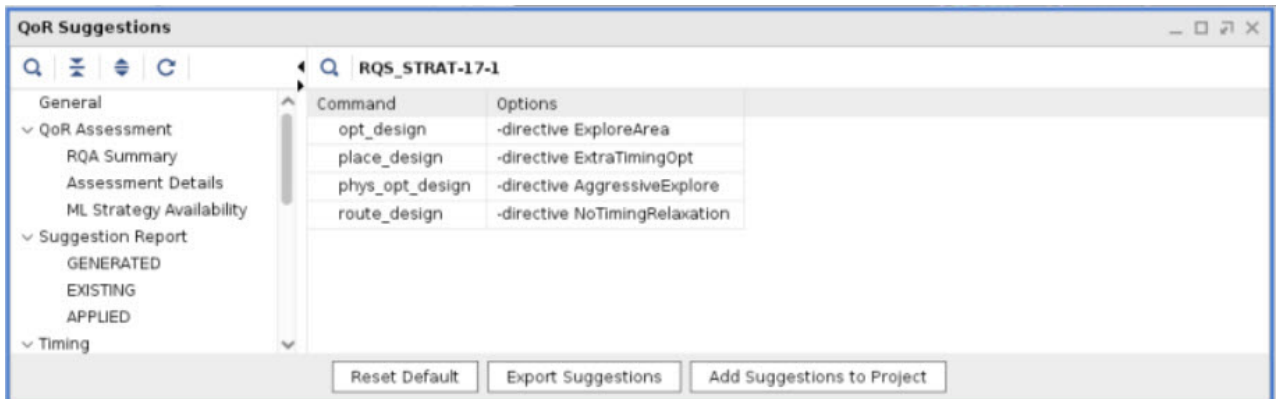


The status for all directives must be listed as OK to generate strategies. The requirements are as follows:

- 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.
- 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.
 - In the QoR suggestion report, select **GENERATED**. Three new strategies have been generated but they are not selectable because strategy generation is currently only available using Tcl.



8. Select **RQS_STRAT-17-1**. 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.



9. At the Tcl console, ensure you are in a suitable writable directory. This can be at the same level as the project. Issue the `write_qor_suggestions` command to write out the suggestions as shown in the following example.

```
file mkdir <extract_Dir>/Lab3/project_2/ML_STRAT
cd <extract_Dir>/Lab3/project_2/ML_STRAT
write_qor_suggestions -strategy_dir ./
```

This writes one RQS file per strategy. Each RQS file also contains all of the other suggestions that are not strategy suggestions. This ensures that you can use all the other QoR suggestions and that there is no confusion as to which strategy suggestion the tools should select.

Step 2: Creating ML Strategy Runs

In this step, you will use the files generated to create ML strategy project-based runs.

1. Examine the contents of the ML_STRAT directory.

Name	Date modified	Type	Size
impl_3Project_MLStrategyCreateRun1.tcl	21/06/2021 14:19	TCL File	2 KB
impl_3Project_MLStrategyCreateRun2.tcl	21/06/2021 14:19	TCL File	2 KB
impl_3Project_MLStrategyCreateRun3.tcl	21/06/2021 14:19	TCL File	2 KB
impl_3SuggestionFile1.rqs	21/06/2021 14:19	RQS File	8 KB
impl_3SuggestionFile2.rqs	21/06/2021 14:19	RQS File	8 KB
impl_3SuggestionFile3.rqs	21/06/2021 14:19	RQS File	8 KB
NonProject_MLStrategyCreateRun1.tcl	21/06/2021 14:19	TCL File	1 KB
NonProject_MLStrategyCreateRun2.tcl	21/06/2021 14:19	TCL File	1 KB
NonProject_MLStrategyCreateRun3.tcl	21/06/2021 14:19	TCL File	1 KB

You can see the following contents:

- 3 x RQS files
- 3 x project-based Tcl scripts
- 3 x 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. The project-based scripts can be sourced. Each of the three scripts references one of the RQS files. All three should be sourced.

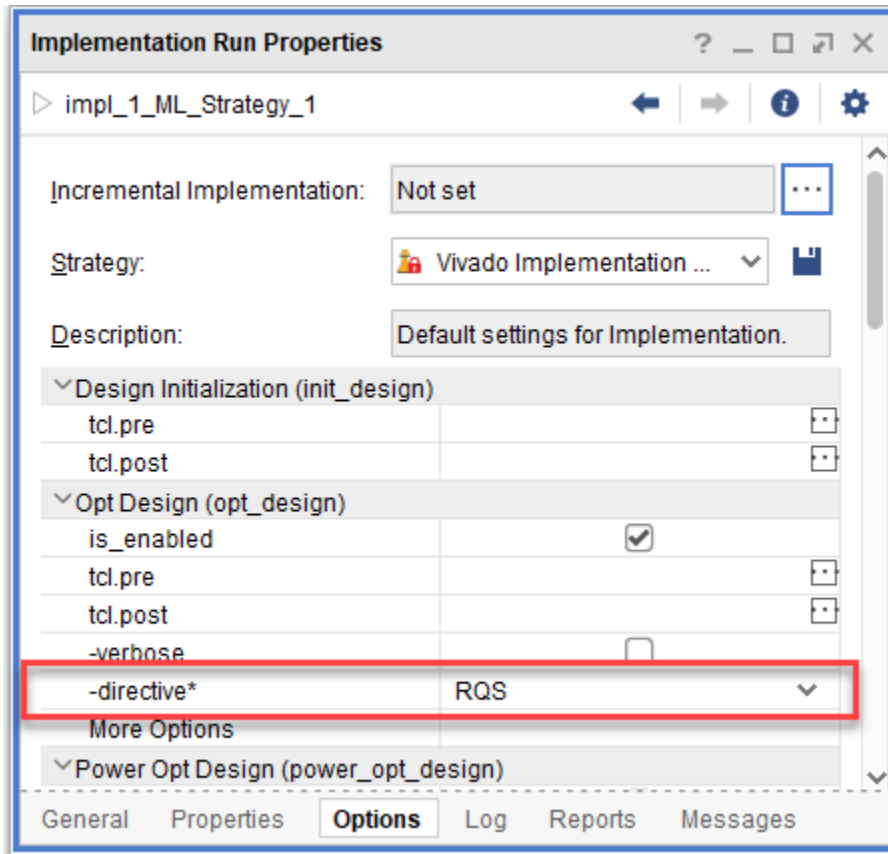
2. Source each of the project-based Tcl files. Each one creates a run in the Design Runs window, sets up the RQS file, and sets the directives to RQS. The run options are copied from the reference run.

```
source ./impl_3Project_MLStrategyCreateRun1.tcl
source ./impl_3Project_MLStrategyCreateRun2.tcl
source ./impl_3Project_MLStrategyCreateRun3.tcl
```

3. In the Design Runs window, select **impl_2_ML_Strategy_1**.

Name	Constraints	Status	Incremental	WNS	TNS	WHS	THS	WBSS	TPWS	Total P
synth_1_rqs	OriginalConstraints	synth_design Complete!	Off							
impl_3 (active)	OriginalConstraints	route_design Complete, Failed Timing!	Off	-0.168	-78.372	0.003	0.000		0.000	
impl_3_ML_Strategy_1	OriginalConstraints	Not started	Off							
impl_3_ML_Strategy_2	OriginalConstraints	Not started	Off							
impl_3_ML_Strategy_3	OriginalConstraints	Not started	Off							

4. In the Implementation Run Properties window, select the **Properties** tab and confirm that **RQS_FILES** is set.
5. 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.

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

Lab 4

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.

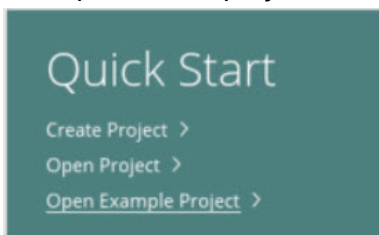
This lab covers how to:

- 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.

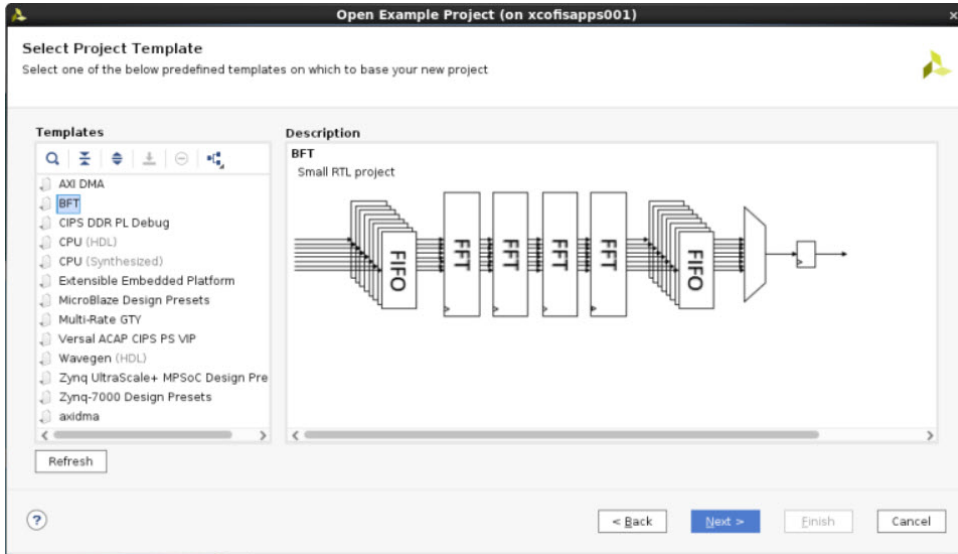
Step 1: Creating Intelligent Design Runs

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

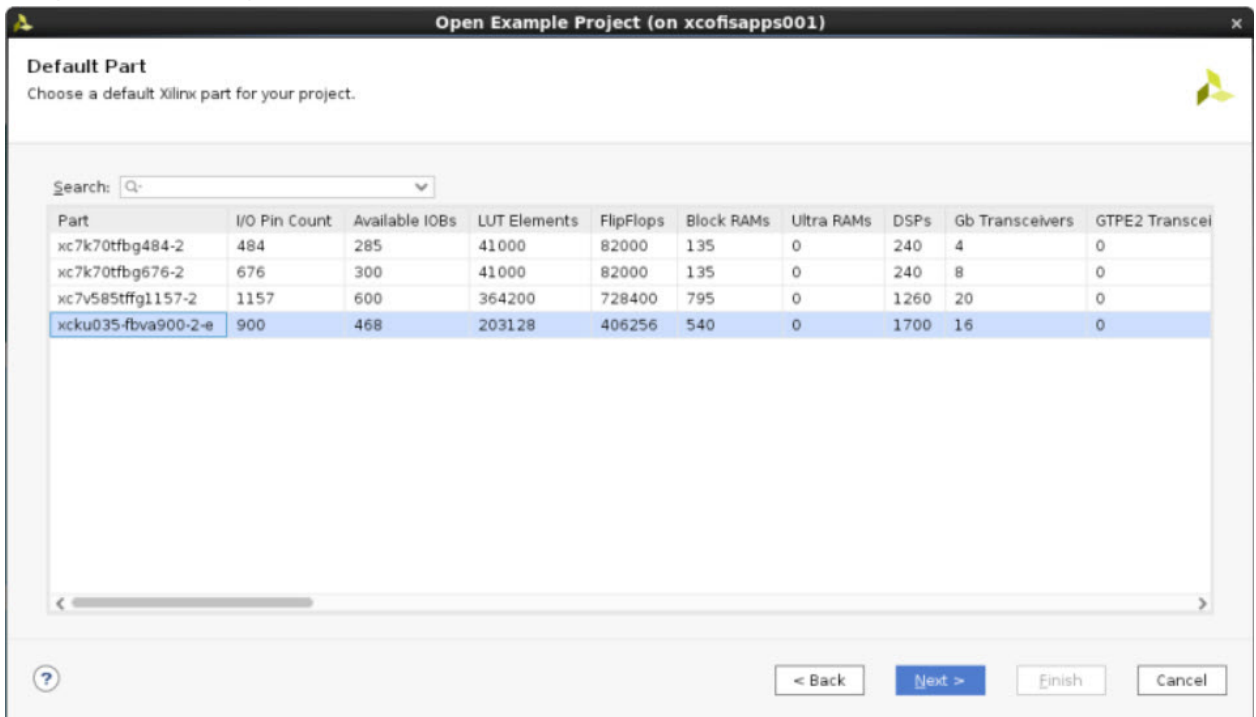
1. First, you need a project. Select **Open Example Project**.



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**.

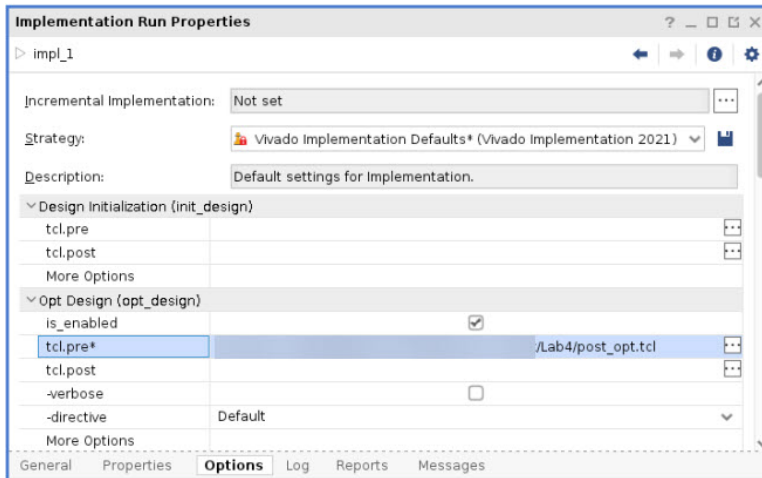


You should now have an open project that is waiting to be synthesized.

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**.

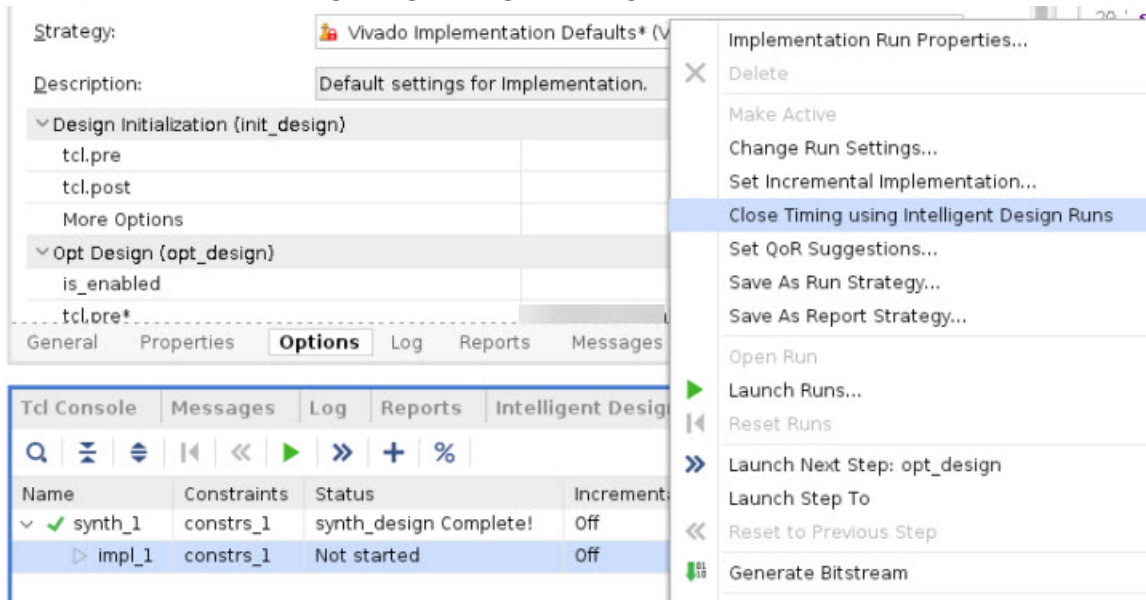
6. In the Implementation Run Properties window, select the **Options** tab.
7. Scroll to the `opt_design tcl.pre` option and add the Tcl script `<extract_dir>/Lab4/pre_opt.tcl` to the option.



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**.



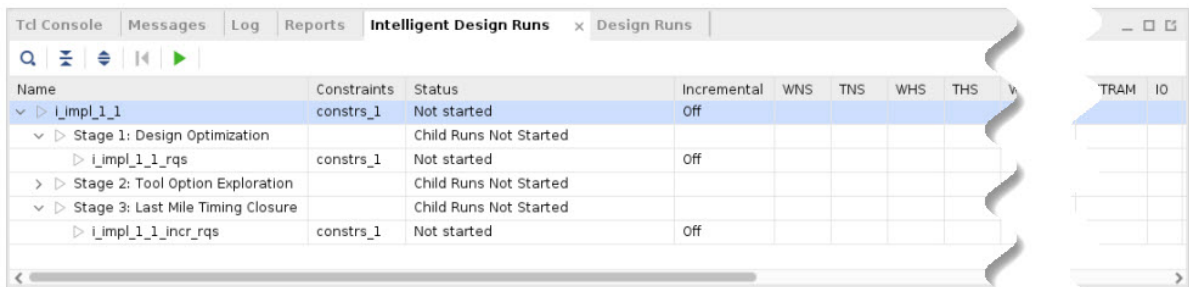
If you try this step again to create an additional run, you will find that the option is disabled. This is to prevent the creation of runs that would produce the same results. Runs created from the same netlist with different directives produce the same results because the IDR controls the directive,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

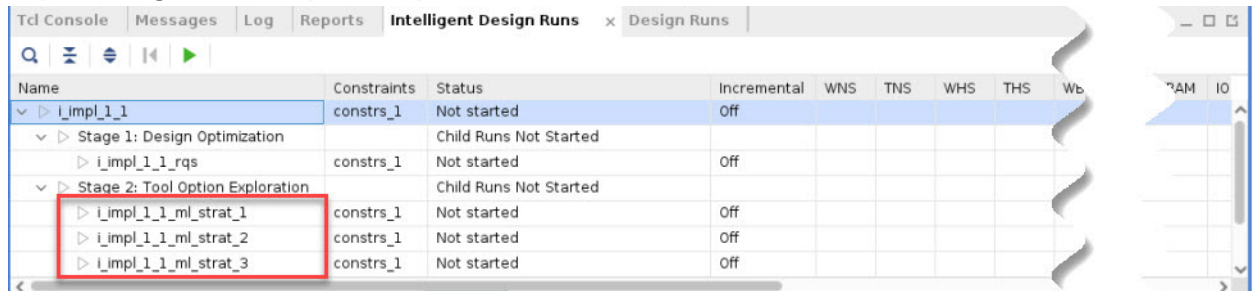


This window is opened when an IDR is created. If closed, it can be reopened by selecting **Window → Intelligent Design Runs**. From this window you can see that there is a hierarchical nature to Intelligent Design Runs. The top-level run is split into three stages:

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

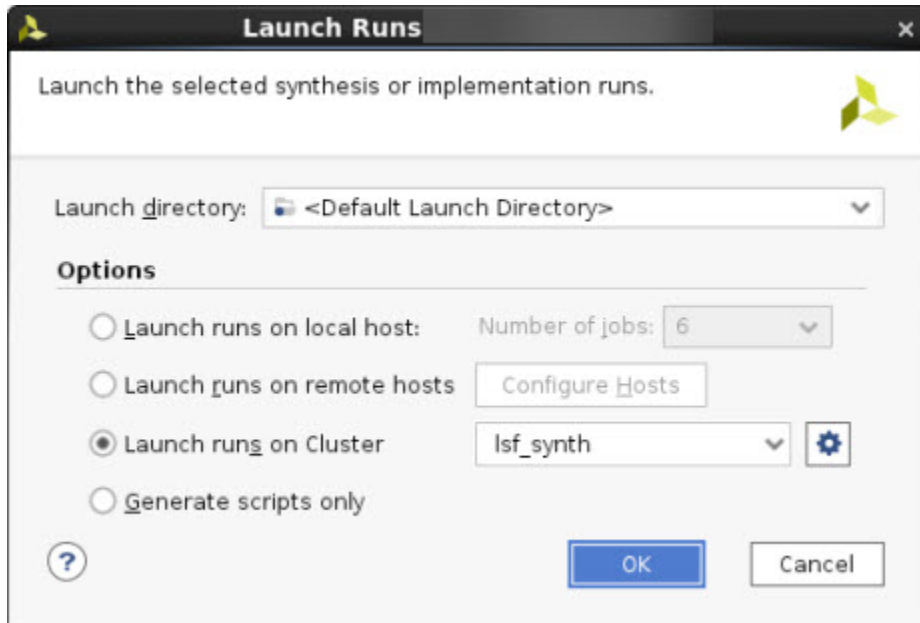
In this lab, only Stage 1 is demonstrated. This is the most complex stage, but can be completed quickly because you are working with a simple design.

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 the right time:

- 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.

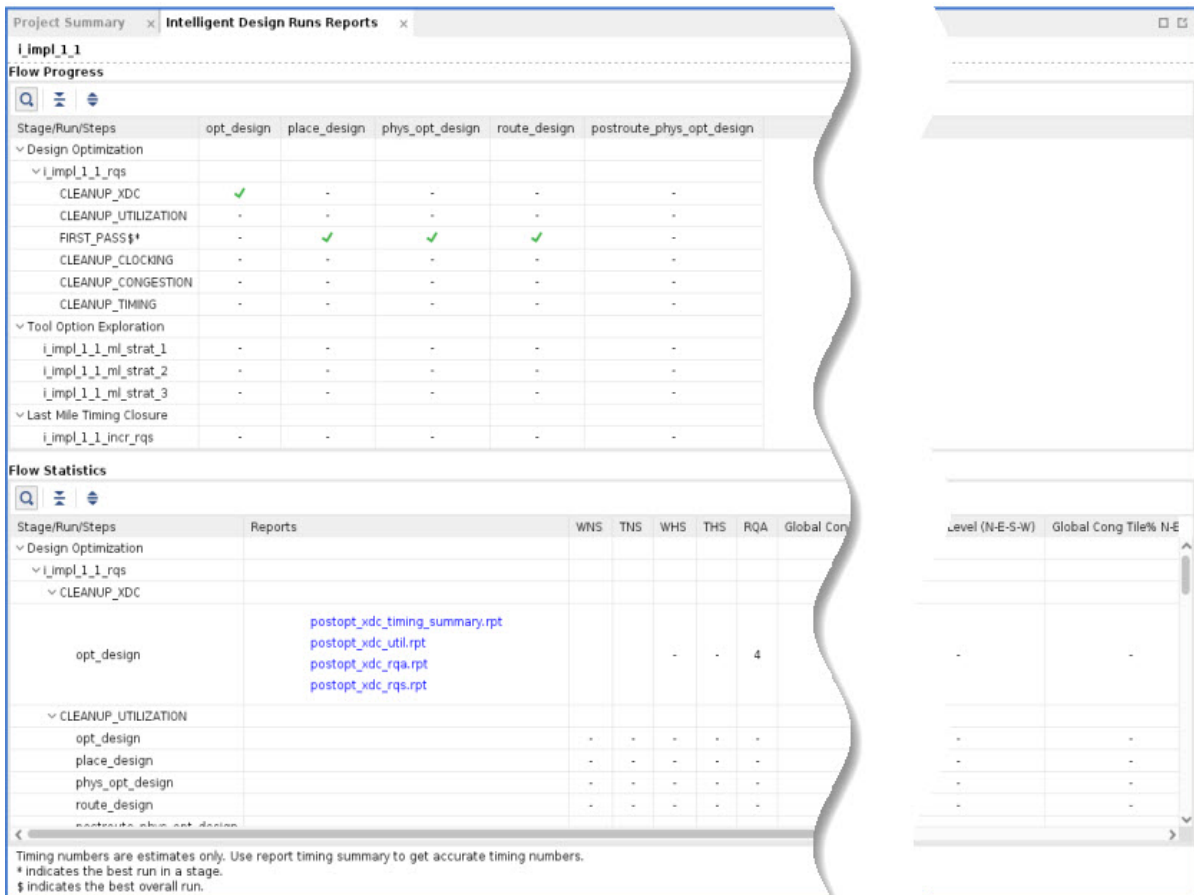


When the lower-level run is selected, you can see that there is a reduced set of options. The Open Run Directory and Open Run options are scoped to the lower-level run. If Open Run is selected, the best completed routed run is opened.

Step 3: Analyzing the Reports and Log File

In this task, you will see where to find reports and analyse 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 screenshot shows the 'Intelligent Design Run Reports' window for project 'i_impl_1_1'. It is divided into two main sections: 'Flow Progress' and 'Flow Statistics'.

Flow Progress Table:

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	-	-	-	-	-
Tool Option Exploration					
i_impl_1_1_mi_strat_1	-	-	-	-	-
i_impl_1_1_mi_strat_2	-	-	-	-	-
i_impl_1_1_mi_strat_3	-	-	-	-	-
Last Mile Timing Closure					
i_impl_1_1_incr_rqs	-	-	-	-	-

Flow Statistics Table:

Stage/Run/Steps	Reports	WNS	TNS	WHS	THS	RQA	Global Cong
Design Optimization							
i_impl_1_1_rqs							
CLEANUP_XDC							
opt_design	postopt_xdc_timing_summary.rpt postopt_xdc_util.rpt postopt_xdc_rqa.rpt postopt_xdc_rqs.rpt					4	
CLEANUP_UTILIZATION							
opt_design		-	-	-	-	-	-
place_design		-	-	-	-	-	-
phys_opt_design		-	-	-	-	-	-
route_design		-	-	-	-	-	-
postroute_phys_opt_design		-	-	-	-	-	-

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.

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.

Hover between sections and pull up the bottom section of the report

Collapse these sections

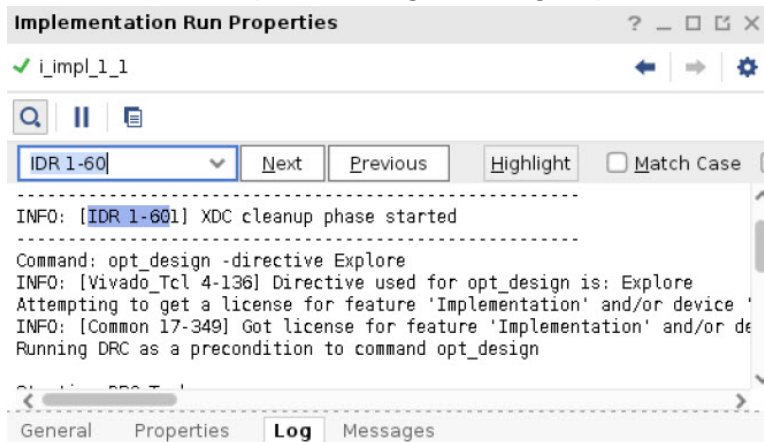
Stage/Run/Steps	Report	INS	TNS	WHS	THS	RQA	Global Cong Level (N-E)
opt_design		-	-	-	-	-	-
place_design	postplace_first_pass_timing_summary.rpt postplace_first_pass_util.rpt postplace_first_pass_rqa.rpt postplace_first_pass_rqs.rpt	2.051	0.000	-	-	3	-
phys_opt_design	postplace_physopt_first_pass_timing_summary.rpt postplace_physopt_first_pass_util.rpt postplace_physopt_first_pass_rqa.rpt postplace_physopt_first_pass_rqs.rpt	2.051	0.000	-	-	3	-
route_design	postroute_first_pass_timing_summary.rpt postroute_first_pass_util.rpt postroute_first_pass_rqa.rpt postroute_first_pass_rqs.rpt	1.407	0.000	0.000	0.000	5	-
postroute_phys_opt_design		-	-	-	-	-	-
> CLEANUP_CLOCKING							
> CLEANUP_CONGESTION							
> CLEANUP_TIMING							
> Tool Path Closure							
> Last Mile Timing Closure							

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.

- 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. You will be in 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 will 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.



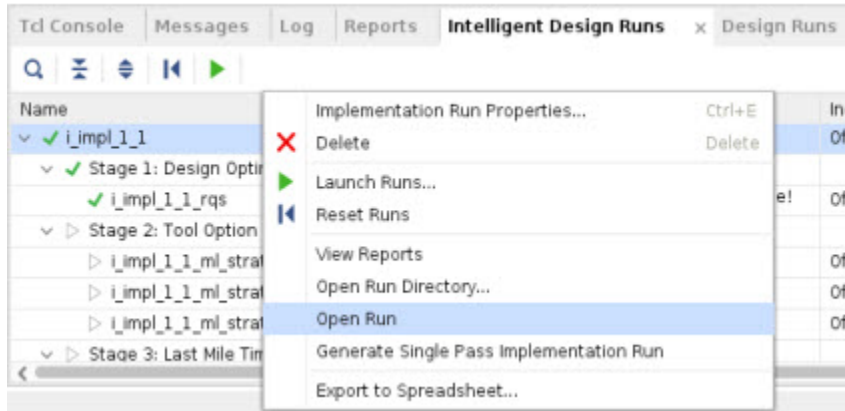
As the implementation process goes back and forth, it can be difficult to identify which step you are in. These messages will help clarify this.

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.

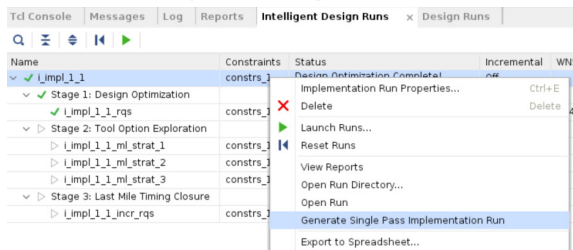
1. In the Intelligent Design Runs window, right-click on the top-level run, **impl_1_1**, and then click **Open Run**. Doing this opens the best overall run from the three stages. This is also the best run from stage 1: stage 2 and stage 3 were not run with this design, and we have seen the best run and best stage run 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.

Note: It is not possible to make an IDR the active run.

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, `impl11_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 an easy-to-meet-timing design, 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

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 2012-2021 Xilinx, Inc. Xilinx, the Xilinx logo, Alveo, Artix, Kintex, Spartan, Versal, 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.