

Vivado Design Suite Tutorial

Implementation

UG986 (v2021.2) October 27, 2021

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





Revision History

10/27/2021: Released with Vivado® Design Suite 2021.2 without changes from 2021.1.

	Section	Revision Summary		
	08/24/2021 V	ersion 2021.1		
Re	lease updates.	Validated for Xilinx [®] release 2021.1.		
	02/17/2021 V	ersion 2020.2		
Re	lease updates.	Validated for Xilinx release 2020.2.		
	08/12/2020 V	ersion 2020.1		
•	Figure 35: Design Runs Window after Completion	Figure Updates		
•	Figure 37: Incremental Reuse Report			
•	Figure 45: Vivado [®] IDE Showing project_bft_core Project Details			



Table of Contents

Revision History	2
Navigating Content by Design Process	5
Implementation Tutorial	6
• Tutorial Design Description	7
Hardware and Software Requirements	7
Preparing the Tutorial Design Files	7
Chapter 1: Lab 1: Using Implementation Strategies	9
Step 1: Opening the Example Project	9
Step 2: Creating Additional Implementation Runs	15
Step 3: Analyzing Implementation Results	16
Step 4: Tightening Timing Requirements	18
Conclusion	19
Chapter 2: Lab 2: Using Incremental Implementation	20
Step 1: Opening the Example Project	
Step 2: Viewing the Incremental Column in the Design Runs Window	24
Step 3: Turning on Incremental Implementation	
Step 4: Compiling the Reference Design	27
Step 5: Making Incremental Changes	28
Step 6: Rerunning Synthesis and Implementation	
Conclusion	33
Chapter 2: Lah 2: Manual and Directed Routing	24
Stop 1: Opening the Example Breigst	34
Step 2: Performing Place and Poute on the Design	
Step 3: Apalyzing Output Bus Timing	
Step 4: Improving Bus Timing through Placement	40 ⊿5
Step 5: Using Manual Routing to Reduce Clock Skew	4
Step 6: Copying Routing to Other Nets	
Conclusion	



Chapter 4: Lab 4: Vivado ECO Flow	65
Step 1: Creating a Project Using the Vivado New Project Wizard	67
Step 2: Synthesizing, Implementing, and Generating the Bitstream	
Step 3: Validating the Design on the Board	69
Step 4: Making the ECO Modifications	
Step 5: Implementing the ECO Changes	
Step 6: Replacing Debug Probes	
Conclusion	99
Appendix A: Additional Resources and Legal Notices	100
Please Read: Important Legal Notices	100





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.





Implementation Tutorial

IMPORTANT! This tutorial requires the use of the Kintex[®]-7 and Kintex[®] UltraScale[™] family of devices. You will need to update your Vivado[®] Design Suite tools installation if you do not have these device families installed. Refer to the Vivado Design Suite User Guide: Release Notes, Installation, and Licensing (UG973) for more information on Adding Design Tools or Devices.

This tutorial includes four labs that demonstrate different features of the Xilinx[®] Vivado Design Suite implementation tool:

- Lab 1 demonstrates using implementation strategies to meet different design objectives.
- Lab 2 demonstrates the use of the incremental compile feature after making a small design change.
- Lab 3 demonstrates the use of manual placement and routing, and duplicated routing, to finetune the timing on the design.
- Lab 4 demonstrates the use of the Vivado ECO to make quick changes to your design post implementation.

Vivado implementation includes all steps necessary to place and route the netlist onto the FPGA device resources, while meeting the logical, physical, and timing constraints of a design.

VIDEO: You can also learn more about implementing the design by viewing the following Quick Take videos:

- Vivado Quick Take Video: Implementing the Design
- Vivado Quick Take Video: Using Incremental Implementation in Vivado

TRAINING: Xilinx provides training courses that can help you learn more about the concepts presented in this document. Use these links to explore related courses:

- Designing FPGAs Using the Vivado Design Suite 1
- Designing FPGAs Using the Vivado Design Suite 2
- Designing FPGAs Using the Vivado Design Suite 3
- Designing FPGAs Using the Vivado Design Suite 4





Tutorial Design Description

The design used for Lab 1 is the CPU Netlist example design, project_cpu_netlist_kintex7, provided with the Vivado Design Suite installation. This design uses a top-level EDIF netlist source file, and an XDC constraints file.

The design used for Lab 2 and Lab 3 is the BFT Core example design, project_bft_kintex7. This design includes both Verilog and VHDL RTL files, as well as an XDC constraints file.

The design used for Lab 4 is available as a Reference Design from the Xilinx website. See information in Locating Design Files for Lab 4.

The CPU Netlist and BFT Core designs target an XC7K70T device, and the design for Lab 4 targets an XCKU040 device. Running the tutorial with small designs allows for minimal hardware requirements and enables timely completion of the tutorial, as well as minimizing data size.

Hardware and Software Requirements

This tutorial requires that the 2020.2 Vivado Design Suite software release or later is installed.

Refer to the Vivado Design Suite User Guide: Release Notes, Installation, and Licensing (UG973) for a complete list and description of the system and software requirements.

Preparing the Tutorial Design Files

Locating Design Files for Labs 1-3

You can find the files for Labs 1-3 in this tutorial in the Vivado Design Suite examples directory at the following location:

<Vivado_install_area>/Vivado/<version>/examples/Vivado_Tutorial

You can also extract the provided zip file, at any time, to write the tutorial files to your local directory, or to restore the files to their starting condition.

Extract the zip file contents from the software installation into any write-accessible location.

<Vivado_install_area>/Vivado/<version>/examples/Vivado_Tutorial.zip

The extracted Vivado_Tutorial directory is referred to as <Extract_Dir> in this tutorial.



Note: You will modify the tutorial design data while working through this tutorial. You should use a new copy of the original <code>Vivado_Tutorial</code> directory each time you start this tutorial

Locating Design Files for Lab 4

To access the reference design for Lab 4, do the following:

- 1. In your C: drive, create a folder called /Vivado_Tutorial.
- 2. Download the reference design files from the Xilinx website.
- 3. Unzip the tutorial source file to the /Vivado_Tutorial folder.



Chapter 1

Lab 1: Using Implementation Strategies

In this lab, you will learn how to use implementation strategies with design runs by creating multiple implementation runs employing different strategies, and comparing the results. You will use the CPU Netlist example design that is included.

Step 1: Opening the Example Project

1. Open the Xilinx Vivado IDE.

On Linux:

1. Change to the directory where the lab materials are stored:

```
cd <Extract_Dir>/Vivado_Tutorial
```

2. Launch the Vivado IDE: vivado

On Windows:

1. To launch the Vivado IDE, select:

```
Start \rightarrow All Programs \rightarrow Xilinx Design Tools \rightarrow Vivado 2020.x \rightarrow Vivado 2020.x
```

Note: Your Vivado Design Suite installation might be called something other than Xilinx Design Tools on the Start menu.

Note: As an alternative, click the Vivado 2020.x Desktop icon to start the Vivado IDE.

2. From the Getting Started page, click Open Example Project.



Quick Start

Create Project >

Open Project >

Open Example Project >

3. In the Create an Example Project page, click Next.

🍌 Open Example Project		\times
HLX Editions	Create an Example Project This wizard will guide you through the creation of a new Vivado project from a predefined template. To create a Vivado project you will specify the type of template project you would like to create. Next, you will need to provide a name and a location for your project files and choose a default board.	
E XILINX.		
?	< <u>Back</u> <u>Next</u> > <u>Finish</u> Cancel	

4. In the Select Project Template page, choose the **CPU (Synthesized)** project and click **Next**.



Open Example Project			×
Select Project Template Select one of the below predefined te	mplates on which to base your new project		A
Templates	Description		
Base Zynq UltraScale+ MPSoC Base MicroBlaze Base Zynq BFT	CPU (Synthesized) Large synthesized netlist project	bone	Î
Configurable MicroBlaze Design Configurable Zynq UltraScale+ MF CPU (HDL) CPU (Synthesized)		Clock	
Wavegen (HDL)	Open RISC CPU	USB 0	
	FFT BFT Engine	USB 1	~
?	< second	<u>N</u> ext >	Finish Cancel

- 5. In the Project Name page, specify the following, and click **Next**:
 - Project name: project_cpu_netlist
 - **Project location:** <Project_Dir>



🝌 Open Example Pr	roject		×
Project Name			
Enter a name for y	our project and specify a directory w	vhere the project data files will be stored	A
<u>P</u> roject name:	project_cpu_netlist		\otimes
Project location:	C://ivado_Tutorial		⊗
🕑 Create proje	ct subdirectory		
Project will be cr	reated at: C://ivado_Tutorial/project	t_cpu_netlist	
?		< <u>B</u> ack <u>N</u> ext >	<u>Finish</u> Cancel

6. In the Default Part screen, select the **xc7k70tfbg676-2** part and click **Next**.



٨	Open Example Projec	t							\times
De Ch	efault Part oose a default Xilinx	part for your proj	iect.						4
	Part	I/O Pin Count	Available IOBs	LUT Elements	FlipFlops	Block RAMs	Ultra RAMs	DSPs	Gb Tra
	xc7k70tfbg676-2	676	300	41000	82000	135	0	240	8
									>
Ċ				< <u>B</u>	ack	<u>N</u> ext ≻	<u>F</u> inish		Cancel

7. In the New Project Summary page, review the project details, and click **Finish**.





🍌 Open Example Project		\times
Open Example Project	New Project Summary A new project named 'project_cpu_netlist will be created from the 'CPU (Synthesized)' template. The default part and product family for the new project: Default Part: xc7k70tfbg676-2 Product: Kintex-7 Family: Kintex-7 Package: fbg676 Speed Grade: -2	×
E XILINX.	To create the project, click Finish	
?	< <u>B</u> ack <u>N</u> ext > <u>F</u> inish Cancel	1

The Vivado IDE opens with the default view.

Pin gin fing log log log log log log log log log lo	File Edit Flow Iools Re	ports Window Layout View Help ↓ III ☆ ∑ ☆ // × PROJECT MANAGER - project_1	Q+ Quick Access				Ready
Control Control Control <td>► → ■ × Flow Navigator Ξ 0 ? ✓ DPO LECT MANAGER</td> <td>PROJECT MANAGER - project_1</td> <td></td> <td></td> <td></td> <td></td> <td></td>	► → ■ × Flow Navigator Ξ 0 ? ✓ DPO LECT MANAGER	PROJECT MANAGER - project_1					
Ipone rest Ipone rest Ipone rest Ipone rest Product Namedation Fragentianes Ipone rest Ipone rest Product Namedation Fragentianes Ipone rest Ipone rest Product Namedation Fragentianes Ipone rest Ipone rest Product Namedation Fragentianes Ipone rest Ipone rest Product Namedation Fragentianes Ipone rest Ipone rest Product Namedation Fragentianes Ipone rest Ipone rest Product Namedation Fragentianes Product Namedation Fragentianes Ipone rest Ipone rest Product Namedation Fragentianes Product Namedation Fragentianes <td>Flow Navigator</td> <td>PROJECT MANAGER - project_1</td> <td></td> <td></td> <td></td> <td></td> <td>III Default Layout V</td>	Flow Navigator	PROJECT MANAGER - project_1					III Default Layout V
PROCINAMAGE Settings Settings Project Seminary Or construction Or construction<	PRO JECT MANAGER						? ×
• Selegios Mid Sources Logoupy Finaldato • Selection code Sources (1) • Se Design Bources (1) •	* PROJECT MANAGER						
Ad Sauras Laguaga Tensidas Laguaga Tensidas Delagis Sauras (D) Sauras Delagis Sauras (D) <td>Settings</td> <td>Sources</td> <td>? _ U U X</td> <td>Project Summary</td> <td></td> <td></td> <td>7 L L X</td>	Settings	Sources	? _ U U X	Project Summary			7 L L X
Language Templates So Description So Descrip	Add Sources	Q ± 0 + 2*	•	Overview Dashboard			
 SARLATON Rus Binudaton Selful an object to see properties Open Symptements Operating Operating Oper	Language Templates	> Design Sources (1)		Sottings Edit			î
 SHUATON Rut Bismudion Set Bismudion Set Bismudion Set Bismudion Set Bismudion		 Constraints (2) Simulation-Only Sources (1) 		octongo con			
Restinution Product State (State State	✓ SIMULATION	> Utility Sources		Project name: proj Project location: C:I	ec_1 Isers/mocklas/lab_1/moject_1		
 NETUET VARVSB Open dynkesiske Design Open dynkesiske Design Open Hardsare Mangare Open Hardsare Mangare Open Hardsare Mangare Open Hardsare Mangare Steld an object bisse properties 	Run Simulation			Product family: Kint	ex-7		
 NETLERFARTANON Open Injensestado Bosjin Open Injensestado Bosjin				Project part: xc7	70tbg676-2		
 Joen Sphesed Design MR-EBENTATION Open Ingenerated Design Select an object to see properties Beind Ingenerated Design Beind Ingenerated Design Open Ingenerated Design Select an object to see properties Beind Ingenerated Design Terrore Range Ran Ingelementation Design Rank X Design Rank X <li< td=""><td> NETLIST ANALYSIS </td><td></td><td></td><td>Top module name: top</td><td></td><td></td><td></td></li<>	 NETLIST ANALYSIS 			Top module name: top			
 MPLEMINITION Net indigeneration Open Hackage Manager Manager Manager Open Hackage Manager Open Hackage Manager Open Hackage Manager Select un object to see proprietes Open Hackage Manager Manager Manager Manager Manager Manager Manager Manager Man	> Open Synthesized Design			Simulator language: VHL	AL		
Restingtionentation Open Implementation Decomposition Decomposite Decomposition Decomposition Decompositer Dec	V IMPLEMENTATION						
 > Open Hardware Manager > Select an object to see properties > Select an object to see properties	Run Implementation	Librarias Compile Order		Implementation			
Precifies Pre	Open Implemented Design	Comple order					
 Produkt AND DEBUG Bessages Log Reports Select in object to see properties Produkt Model Produkt Model<td></td><td>Properties</td><td>? _ 🗆 🖾 ×</td><td>Status: Messanes:</td><td>Not started</td><td></td><td></td>		Properties	? _ 🗆 🖾 ×	Status: Messanes:	Not started		
Image: Select arr object to see properties Image: Select arr object to see properties Select arr object to see properties Teal opject metal on object to see properties Teal opject metal on object to see properties Image: Select arr object to see properties Select arr object to see properties Teal opject metal on object to see properties Teal opject metal on object to see properties Teal opject metal on object to see properties Image: Select arr object to see properties Teal opject metal on object to see properties Image: Select arr object to see properties Teal opject metal on object to see properties Teal opject metal on object to see properties Teal opject metal on object to see properties Image: Constraints Statist do in the properties Statist do in the properties of the properties of the properties Teal opject to an object to an	PROGRAM AND DEBUG		\leftarrow \rightarrow \Diamond	Part	xc7k70tfbg676-2		
> Open Hardware Manager Select in object to see properties Reports Parlager Value for properties Select in object to see properties Reports Parlager Value for properties Tot Consele Reports Parlager Value for properties Tot Consele Reports Parlager Value for properties Tot Consele Reports Parlager Reports Parlager Tot Consele Report Parlager Report Parlager Tot Consele Report Parlager <t< td=""><td>👫 Generate Bitstream</td><td></td><td></td><td>Strategy:</td><td>Vivado Implementation Defaults</td><td></td><td></td></t<>	👫 Generate Bitstream			Strategy:	Vivado Implementation Defaults		
Belied an object to see properties Constraint Constraint Constraint Constraint Run Inglementation to see CRO results Yet Consult Messages Log Run Inglementation to see CRO results Run Inglementation to see CRO results Run Inglementation to see CRO results Yet Consult Messages Log Run Inglementation to see CRO results Run Inglementation to see CRO results Run Inglementation to see CRO results Not State Not State Not State Not State Not State Report State	> Open Hardware Manager			Report Strategy:	Vivado Implementation Default Reports		
Select an object to see properties DRC Volations Run Inglementation to see DRC results Run Inglementation to Run Ingl				Constraints:	constrs_2		
Tel Conside Tening Conside Not Stated To Conside Memory Consider To Conside Memory Consider To Conside Memory Consider Conside Memory Consider To Conside Memory Consider Conside Memory Consider Conside Memory Consider Memory Consider Memory Consider Constants State Consider Dimpt_1 constants State Constants Not Stated Not Stated				incremental imprementation.			
Run Implementation to see DRC results Run Implementation to see DRC results TGC Console Messages Log Reports Design Runs X Q X I I X I I X I X I X I X I X I X I X I X I X I X I X I X I X I X I X X I X I X X I X X I X		Select an object to see properties		DRC Violations		Timing	
Tel Console Messages Log Reports Design Runs Constraints Status THE THE THE THE Falled Routes LUT FF BeNAIs LRAIN DSP Status Design Runs Report Status Name Constraints Status WHS THS THS THS THS THS Def Runt DSP Status DSP Status DSP Report Status Windown Def Runt Def Runt Report Status D ingl_1 constra_2 Not status THS THS THS THS THS Def Runt Def Runt Def Runt Def Runt Def Runt Def Runt Report Status							
Terr Consulta Messages Log Regr Design Paras ×					Run Implementation to see DRC results	Run Implementation to see timin	g results
Tel Constraints Manage Log Reports Design Runs x *							~
Tel Constaint Messagers Los (Reports Design Rems x → 7 - 0 Q, ≚							
Q. Ž. IV. IV. IV. IV. IV. IV. Total Power Failed Routes LUT FF BRAMS DSP Start Bageed Run Strategy Report Strategy Interp1_1 constra_2 Not started Visado Implementation Defaults (Visado Implementation Defaults		Tcl Console Messages Log Repo	orts Design Runs ×				? _ 🗆 🖸
Name Constraints Status VMIN THS THS TPVIN Total Power Palled Routes LUT FF BRMIs URAM DSP Statu Expansion Report Strategy Report Strategy Report Strategy Not statudes Not statudes <td></td> <td>Q ± ≑ 14 ≪ ▶ ≫</td> <td>+ %</td> <td></td> <td></td> <td></td> <td></td>		Q ± ≑ 14 ≪ ▶ ≫	+ %				
v imp_1 const_2 not same		Name Constraints Status	WNS TNS WH	3 THS TPWS Total Power	Failed Routes LUT FF BRAMs URAM DSP Start Elapsed	Run Strategy Ri	aport Strategy
		impl_1 constrs_2 Not starts	ed			vivado Implementation Defaults (vivado Implementation 2018) Vi	vado Implementation Detault Reports (vivado I
		<					>



Step 2: Creating Additional Implementation Runs

The project contains previously defined implementation runs as seen in the Design Runs window of the Vivado IDE. You will create new implementation runs and change the implementation strategies used by these new runs.

- 1. From the main menu, select **Flow** \rightarrow **Create Runs**.
- 2. The Create New Runs wizard opens.
- 3. Click **Next** to open the Configure Implementation Runs screen.

The screen appears with a new implementation run defined. You can configure this run and add other runs as well.

- 4. In the Run Strategy drop-down menu, select **Performance_Explore** as the strategy for the run.
- 5. Click the Add 🕇 button twice to create two additional runs.
- 6. Select Flow_RunPhysOpt as the Run Strategy for the impl_3 run.
- 7. Select **Flow_RuntimeOptimized** as the Run Strategy for the impl_4 run.

The Configure Implementation Runs screen now displays three new implementations along with the strategy you selected for each, as shown in the following figure.

A	Create New	Runs					×
c	configure I reate and co Create Impl	imple nfigu	ementation Run: re one or more imple ntation Runs	s mentation runs usin	g various parts, constraints, flows and strategies		4
	+ - Name impl_2	P	Constraints Set □ constrs_2 ∨	Part ⊕ xc7k70tfb ∨	Run Strategy	Report Strategy	Make Ac
	impl_3	Ø	□ constrs_2 ∨	@ xc7k70tfb ✓	In Flow_RunPhysOpt (Vivado Implementation 20 ♥	a UltraFast Design Methodology Reports (Vivado Impl ↔	0
			Condid_e			Child be beorgn methodology reports (rifedo implimited)	to create: 3
(?					< <u>B</u> ack <u>N</u> ext > Einish	Cancel

- 8. Click **Next** to open the Launch Options screen.
- 9. Select **Do not launch now**, and click **Next** to view the Create New Runs Summary.



10. In the Create New Runs Summary page, click Finish.

Step 3: Analyzing Implementation Results

1. In the Design Runs window, select all the implementation runs.

Design Runs	ign Runs ? ב ם 🗙 🗙									
$\mathbf{Q} \mid \mathbf{\Xi} \mid \mathbf{\varphi} \mid \mathbf{I} \mathbf{A} \mid \mathbf{W} \mid \mathbf{W} \mid \mathbf{W} \mid \mathbf{H} \mid \mathbf{W} \mid$										
Name	Part	Constraints	Status	Run Strategy	Description					
impl_1 (active)	xc7k70tfbg676-2	constrs_2	Not started	Vivado Implementation Defaults (Vivado Implementation 2018)	Default settings for Implementation.					
impl_2	xc7k70tfbg676-2	constrs_2	Not started	Performance_Explore (Vivado Implementation 2018)	Uses multiple algorithms for optimization, placement, and routing to get potentially better results.					
▷ impl_3	xc7k70tfbg676-2	constrs_2	Not started	Flow_RunPhysOpt (Vivado Implementation 2018)	Similar to the Implementation Run Defaults, but enables the physical optimization step (phys_opt_designation)					
▷ impl_4	xc7k70tfbg676-2	constrs_2	Not started	Flow_RuntimeOptimized (Vivado Implementation 2018)	Each implementation step trades design performance for better runtime. Physical optimization (phys_o					
<					\$					
dia ana amin'ny fisiana amin'ny fisiana amin'ny fisiana amin'ny fisiana amin'ny fisiana amin'ny fisiana amin'ny										

- 2. Click the Launch Runs toolbar button ▶.
- 3. In the Launch Runs dialog box, select **Launch runs on local host** and Number of jobs: **4**, as shown in the following figure. The number of jobs is the maximum number of implementation runs for parallel execution. Up to 4 jobs may run in parallel depending on number of available processors with remaining jobs put into a queue.

🝌 Launch Runs	\times
Launch the selected synthesis or implementation runs.	4
Launch <u>d</u> irectory: Cefault Launch Directory>	*
Options	
● Launch runs on local host: Number of jobs: 4 ~	
O Generate scripts only	
_	
Don't show this dialog again	
OK Cano	el

4. Click OK.

Two runs launch simultaneously, with the remaining runs going into a queue. In this example, Vivado is running on a machine with only 2 available processors.



Design Runs	Design Runs ? _ ਹ ਨ X											
Q <u>≭</u> ≑ I•	Q ¥ \$\\$ H \$\langle H \$\langle \$\]											
Name 🗠 1	Part	Constraints	Strategy	Status	Progress	WNS	TNS	WHS	THS	TPWS	Total Power	F
 impl_1 (active) 	xc7k70tfbg676-2	constrs_2	Vivado Implementation Defaults (Vivado Implementation 2017)	Running place_design	50%							
impl_2	xc7k70tfbg676-2	constrs_2	Performance_Explore (Vivado Implementation 2017)	Running place_design	40%							
⊠ impl_3	xc7k70tfbg676-2	constrs_2	Flow_RunPhysOpt (Vivado Implementation 2017)	Queued	0%							
⊠ impl_4	xc7k70tfbg676-2	constrs_2	Flow_RuntimeOptimized (Vivado Implementation 2017)	Queued	0%							
<												>

When the active run, impl_1, completes, examine the Project Summary. The Project Summary reflects the status and the results of the active run. When the active run (impl_1) is complete, the Implementation Completed dialog box opens.

5. Click **Cancel** to close the dialog box.

Note: The implementation utilization results display as a bar graph at the bottom of the summary page (you might need to scroll down), as shown in the following figure.



When you open an implementation run, the report_power and the report_timing_summary results are automatically opened for the run in a new tab in the Results Window.

- 6. When all the implementation runs are complete, select the **Design Runs** window.
- 7. Right-click the impl_3 run in the Design Runs window, and select Make Active from the popup menu.

The Project Summary now displays the status and results of the new active run, impl_3.

Design Runs	Design Runs ? ב ם – א										
Q <u>∓</u> ≑ I•	• « > »	+ %									
Name	Part	Constraints	Status	Run Strategy	Elapsed	WNS	Description				
✓ impl_1	xc7k70tfbg676-2	constrs_2	route_design Complete!	Vivado Implementation Defaults (Vivado Implementation 2018)	00:07:01	0.105	Default settings for Implementation.				
✓ impl_2	xc7k70tfbg676-2	constrs_2	route_design Complete!	Performance_Explore (Vivado Implementation 2018)	00:07:10	0.105	Uses multiple algorithms for optimization, placement, and routin				
✓ impl_3 (active)	xc7k70tfbg676-2	constrs_2	route_design Complete!	Flow_RunPhysOpt (Vivado Implementation 2018)	00:07:36	0.105	Similar to the Implementation Run Defaults, but enables the ph				
✓ impl_4	xc7k70tfbg676-2	constrs_2	route_design Complete!	Flow_RuntimeOptimized (Vivado Implementation 2018)	00:06:54	0.799	Each implementation step trades design performance for better				
<							>				



- 8. Compare the results for the completed runs in the Design Runs window, as shown in the previous figure.
 - The Flow_RuntimeOptimized strategy in impl_4 completed in the least amount of time, as you can see in the Elapsed time column.
 - The WNS column shows that all runs met the timing requirements.

Step 4: Tightening Timing Requirements

To examine the impact of the Performance_Explore strategy on meeting timing, you will change the timing constraints to make timing closure more challenging.

1. In the Sources window, double-click the top_full.xdc file in the constrs_2 constraint set.

The constraints file opens in the Vivado IDE text editor.

Project Summary × top_full.xdc ×	?		
$\mathbf{Q} \mid \blacksquare \mid \bigstar \mid \nsim \mid \gg \mid \mathbf{X} \mid \blacksquare \mid \blacksquare \mid \mathbf{M} \mid \mathbf{M} \mid \blacksquare \mid \mathbf{Q} \mid$			
1 # Define the top level system clock of the design		-	•
2 create_clock -period 10 -name sysClk [get_ports sysClk]			
3			
4 ; # Define the clocks for the GTX blocks			
5 create_clock -name gt0_txusrclk_i -period 12.8 [get_pins mgtEngine/R0CKETI0]	_WRAPPE	SR_TI	
6 create_clock -name gt2_txusrclk_i -period 12.8 [get_pins mgtEngine/ROCKETIO	_WRAPPE	SR_TI	1
7 create_clock -name gt4_txusrclk_i -period 12.8 [get_pins mgtEngine/R0CKETI0]	_WRAPPE	SR_TI	
8 create_clock -name gt6_txusrclk_i -period 12.8 [get_pins mgtEngine/ROCKETIO]	_WRAPPE	ER_TI	
9			
10			

2. On line 2, change the period of the create_clock constraint from 10 ns to 7.35 ns.

The new constraint should read as follows:

create_clock -period 7.35 -name sysClk [get_ports sysClk]

3. Save the changes by clicking the Save File 💾 button in the toolbar of the text editor.

Note: Saving the constraints file changes the status of all runs using that constraints file from "Complete" to "Out-of-date," as seen in the Design Runs window.

Design Runs	sign Runs ? _ ច ភ ×									
Q <u>∓</u> ♦ •	$ \ll \gg \gg$	+ %								
Name	Part	Constraints	Status	Run Strategy	Elapsed	WNS	Description			
🎿 impl_1	xc7k70tfbg676-2	constrs_2	Implementation Out-of-date	Vivado Implementation Defaults (Vivado Implementation 2018)	00:07:01	0.105	Default settings for Implementation.			
🎺 impl_2	xc7k70tfbg676-2	constrs_2	Implementation Out-of-date	Performance_Explore (Vivado Implementation 2018)	00:07:10	0.105	Uses multiple algorithms for optimization, placement, and routin			
🔸 impl_3 (active)	xc7k70tfbg676-2	constrs_2	Implementation Out-of-date	Flow_RunPhysOpt (Vivado Implementation 2018)	00:07:36	0.105	Similar to the Implementation Run Defaults, but enables the ph			
✓ impl_4	xc7k70tfbg676-2	constrs_2	Implementation Out-of-date	Flow_RuntimeOptimized (Vivado Implementation 2018)	00:06:54	0.799	Each implementation step trades design performance for better			
<							>			

4. In the Design Runs window, select all runs and click the Reset Runs 🚺 button.



5. In the Reset Runs dialog box, click Reset.

This directs the Vivado Design Suite to remove all files associated with the selected runs from the project directory. The status of all runs changes from "Out-of-date" to "Not started."

6. With all runs selected in the Design Runs window, click the Launch Runs button.

The Launch Selected Runs window opens.

TIP: You can also launch runs without resetting them first. If the runs are out of date, the Reset Runs dialog box displays. In this dialog box, you can reset the runs before they are launched.

7. Select Launch runs on local host and Number of jobs: 2 and click OK.

When the active run (impl_3) completes, the Implementation Completed dialog box opens.

- 8. Click **Cancel** to close the dialog box.
- 9. Compare the Elapsed time for each run in the Design Runs window, as seen in the following figure.

Design Runs	Design Runs ? ਹ ਮ ×											
Q 素 ≑ I4	Q 素 ≑ K ≪ ▶ ≫ + %											
Name	Part	Constraints	Status	Run Strategy	Elapsed	WNS	TNS	WHS	THS			
✓ impl_1	xc7k70tfbg676-2	constrs_2	route_design Complete, Failed Timing!	Vivado Implementation Defaults (Vivado Implementation 2018)	00:10:47	-0.400	-3.509	0.046	0.000			
✓ impl_2 (active)	xc7k70tfbg676-2	constrs_2	route_design Complete, Failed Timing!	Performance_Explore (Vivado Implementation 2018)	00:15:26	-0.079	-0.084	0.046	0.000			
✓ impl_3	xc7k70tfbg676-2	constrs_2	route_design Complete, Failed Timing!	Flow_RunPhysOpt (Vivado Implementation 2018)	00:13:22	-0.462	-5.340	0.046	0.000			
✓ impl_4	xc7k70tfbg676-2	constrs_2	route_design Complete, Failed Timing!	Flow_RuntimeOptimized (Vivado Implementation 2018)	00:07:32	-0.526	-5.860	0.055	0.000			
<									>			

• Notice that the impl_2 run, using the Performance_Explore strategy is closest to meeting timing, but also took the most time to complete.

Note: Reserve the Performance_Explore strategy for designs that have challenging timing constraints and fail to meet timing with the Implementation Defaults strategy.

Conclusion

In this lab, you learned how to define multiple implementation runs to employ different strategies to resolve timing. You have seen how some strategies trade performance for results, and learned how to use those strategies in a more challenging design.

This concludes Lab 1. If you plan to continue directly to Lab 2, keep the Vivado IDE open and close the current project. If you do not plan to continue, you can exit the Vivado Design Suite.



Lab 2: Using Incremental Implementation

Incremental implementation can be used when a user wants to get faster implementation compile times and more consistent implementation results. Incremental implementation is a flow that achieves greater consistency of results and faster implementation compile times. It should be used when a design is relatively stable and only small changes are required.

After resynthesizing a design with minor changes, the incremental compile flow can speed up placement and routing by reusing results from a prior design iteration. This can help you preserve timing closure while allowing you to quickly implement incremental changes to the design.

In this lab, you use the BFT example design that is included in the Vivado[®] Design Suite, to learn how to use the Incremental Implementation. Refer to the Vivado Design Suite User Guide: *Implementation* (UG904) to learn more about Incremental Compile.

Step 1: Opening the Example Project

- 1. Start by loading Vivado IDE by doing one of the following:
 - Launch the Vivado IDE from the icon on the Windows desktop.
 - Type vivado from a command terminal.
- 2. From the Getting Started page, click **Open Example Project**.



3. In the Create an Example Project page, click Next.



🍌 Open Example Project		×
VIVADO HLx Editions	Create an Example Project This wizard will guide you through the creation of a new Vivado project from a predefined template. To create a Vivado project you will specify the type of template project you would like to create. Next, you will need to provide a name and a location for your project files and choose a default board.	
E XILINX	< <u>B</u> ack <u>Next></u> Finish Cancel	

4. In the Select Project Template page, select the BFT (Small RTL project) design, and click Next.





- 5. In the Project Name page, specify the following, and click Next:
 - Project name: project_bft_core_hdl
 - **Project location:** < Project_Dir>

🙏 Open Example Pr	oject			×
Project Name				
Enter a name for yo	ur project and specify a directory where the project da	ta files will be stored		A
<u>P</u> roject name:	project_bft_core_hdl			\otimes
Project location:	C:/Data/Vivado_Tutorial			⊗ …
🗹 Create projec	t subdirectory			
Project will be cr	eated at: C:/Data/Vivado_Tutorial/project_bft_core_hdl			
?		< <u>B</u> ack	<u>N</u> ext >	Einish Cancel

6. In the Default Part page, select the **xc7k70tfbg484-2** part, and click **Next**.



earch: Q-		~	1					
art	I/O Pin Count	Available IOBs	LUT Elements	FlipFlops	Block RAMs	Ultra RAMs	DSPs	Gb
c7k70tfbg484-2	484	285	41000	82000	135	0	240	4
c7k70tfbg676-2	676	300	41000	82000	135	0	240	8
c7v585tffg1157-2	1157	600	364200	728400	795	0	1260	20
cku035-fbva900-2-e	900	468	203128	406256	540	0	1700	16

7. In the New Project Summary page, review the project details, and click **Finish**.

🙏 Open Example Project	×
	New Project Summary A new project named 'project_bft_core_hdl' will be created from the 'BFT' template.
	The default part and product family for the new project: Default Part: xc7k70tfbg484-2 Product: Kintex-7 Family: Kintex-7 Package: fbg484 Speed Grade: -2
E XILINX.	To create the project, click Finish
(?)	< <u>B</u> ack <u>N</u> ext > <u>Finish</u> Cancel

The Vivado IDE opens with the default view.



🝌 project_bft_core_hdl - [C:/Data/Vi	ivado_Tutorial/project_bft_core_hdl/project_bft_core_	ndl.xpr] - Vivado 2018.1	1	- 0	×	
<u>File Edit Flow T</u> ools	Rep <u>o</u> rts <u>W</u> indow La <u>v</u> out <u>V</u> iew <u>H</u> elp	Q- Quick Access		R	ady	
	< 🕨 🕸 🗵 🖉 🖉			🗮 Default Layout	~	
Flow Navigator $\Xi \Rightarrow ?$	PROJECT MANAGER - project_bft_core_hdl				? ×	
✓ PROJECT MANAGER	Sources	? _ 🗆 🖒 ×	Project Summary	? 🗆	ЦX	
Settings Add Sources	Q ★ + 0 0 > Design Sources (1)	Updating 🕽 🌣	Settings Edit		î	
Language Templates	 bft(aBFT) (bft/vhdl) (20) Constraints (1) Simulation Sources (1) 		Project name: Project location: Product family:	project_bft_core_hdl C:/Data/Vivado_Tutorial/project_t Kintex-7	_bft_cori	
✓ IP INTEGRATOR Create Block Design	Hierarchy Libraries Compile Order		Project part	xc7k70tfbg484-2		
Open Block Design	Properties	? _ 🗆 🖒 X	Target language:	VHDL		
Generate Block Design		← → Q	Simulator language:	Mixed		
 SIMULATION Run Simulation 	Select an object to see propertie	s Synthesis			Impl	
RTL ANALYSIS Open Elaborated Design	Tcl Console Messages Log Reports	Design Runs 🛛 🗙	<	? _		
	Q	%				
 SYNTHESIS Run Synthesis Open Synthesized Design 	Name Constraints Status ✓ ▷ synth_1 constrs_1 Not started ▷ impl_1 constrs_1 Not started	Elapsed WNS T	INS WHS THS TPWS	S Total Power Failed Routes	LUT	
IMPLEMENTATION Run Implementation Open Implemented Design						
✓ PROGRAM AND DEBUG ~	<	-			>	

Step 2: Viewing the Incremental Column in the Design Runs Window

In the Design Runs window, right-click on any of the column headings and enable the Incremental column if it is not already enabled, as shown in the following figures:





Source File Prope	er ? _ D	ı c ×	40 46		errc);		Progress	
bft.vhdl	← →	a	47	¦ attri	bute fs		Incremental	
			49	49 attribute fs 🖌 WNS				is "one-hot"
General Properties						~	TNS	
· · ·				<		~	WHS	
Tcl Console M	Tcl Console Messages Log Reports Design Runs ×					~	THS	
Q, , , , , , , , , , , , , , , , , , ,	I € ≪	▶ ≫	+	%		~	WBSS	
Name	Constraints	Status		Elapsed	WNS .	~	TPWS	Total Power
✓ ▷ synth_1	constrs_1	Not starte	ed			~	Total Power	
⊳ impl_1	constrs_1	Not starte	ed			~	Failed Routes	
<						~	LUT	

Figure 1: Enabling Incremental Heading

Figure 2: Design Runs Window

Tcl Console M	essages Lo	g Reports	Design R	uns ×	
Q ★ ♦		▶ ≫ +	%		
Name	Constraints	Status	Elapsed	Incremental	^1 V
✓ ▷ synth_1	constrs_1	Not started		Off	
⊳ impl_1	constrs_1	Not started		Off	
<					

This column shows how the incremental flow was used. The information is also available in the messages in the Vivado log file.

Step 3: Turning on Incremental Implementation

1. Right-click on the impl_1 run and select Set Incremental Implementation.



General Propertie	es Opti ⊲	50 (a) end (entity bf	×	Implementation Run Properties	Ctrl+E Delete
Tcl Console Messa	ages Log Rej	ports Design R	uns ×		Make Active	
Q		+ %			Change Run Settings	
Name Co	onstraints Status	Elapsed	Increment		Set Incremental Implementation	
✓ ▷ synth_1 co	nstrs_1 Not sta	rted	Off		Save As Strategy	
▶ impl_1 co	nstrs_1 Not sta	rted	Off		Open Run	

2. In the Incremental Implementation dialog box, select **Automatically use the checkpoint from the previous run**.

ightarrow Incremental Implementation $ imes$					
Enable incremental implementation for this run. Allow the tool to automatically choose the design checkpoint from previous run or choose a specific checkpoint					
Automatically use the check	kpoint from the previous run				
Specify design checkpoint	No Existing Utility So 🗸				
Copy sources into projec	t				
O Disable incremental compile	e				
?	OK Cancel				

This dialog box can be opened from many places inside the Vivado IDE. These include the Implementation Run Options window and the Project Summary. The same functionality can also be built into scripts via Tcl commands.

3. To enable automatic checkpoint selection via Tcl, use the following command:

```
set_property AUTO_INCREMENTAL_CHECKPOINT 1 [get_runs impl_1]
```

After this is done, you should see the Incremental Column in the Design Runs window update to Auto(skipped). This setting indicates that Auto mode is enabled but that Incremental Implementation has not been run.



Tcl Console M	essages Lo	g Reports	Design R	uns ×	
Q, ¥ ♦		► » +	3 %		
Name	Constraints	Status	Elapsed	Incremental	^1
✓ ▷ synth_1	constrs_1	Not started	Off		
▷ impl_1	constrs_1	Not started		Auto(Skipped)	
<					

It is possible to select your own checkpoint, which is desirable when the checkpoint must not be updated or lower thresholds for reuse are OK to use. It is your responsibility to manage the suitability of the checkpoint in this case.

Step 4: Compiling the Reference Design

- 1. From the Flow Navigator, select **Run Implementation**.
- 2. In the Missing Synthesis Results dialog box that appears, click OK to launch synthesis first.

Synthesis runs, and implementation starts automatically when synthesis completes.

Note: The dialog box appears because you are running implementation without first running synthesis.



3. After implementation finishes, the Implementation Complete dialog box opens. Click **Cancel** to dismiss the dialog box.





Implementation Completed		
Implementation successfully completed.		
Open Implemented Design		
<u>G</u> enerate Bitstream		
◯ <u>V</u> iew Reports		
Don't show this dialog again		
OK Cancel		

In a project-based design, the Vivado Design Suite saves intermediate implementation results as design checkpoints in the implementation runs directory. You will use the final checkpoint as the reference to the incremental compile flow.

Step 5: Making Incremental Changes

In this step, you make minor changes to the RTL design sources. These changes necessitate resynthesizing the netlist and re-implementing the design.

1. In the Hierarchy tab of the Sources window, double-click the top-level VHDL file, core_transform.vhdlunder arnd1, to open the file in the Vivado IDE text editor, as shown in the following figure.





PROJECT MANAGER - project_bft_core_hdl		? ×
Sources ? _ 🗆 🖾 🗡	Project Summary × bft.vhdl * ×	? 🗆 🖸
Q ¥ ♦ + 2 ● 0 ◊		
 Intraff(aBFT) (bft/hdl) (20) Intraff(aBFT) (20) <	<pre>27</pre>	
Source File Properties ? _ □ □ □ × ⊗ bft.vhdl ← ⇒ ♦	<pre>46); 47 48 attribute fsm_encoding :string; 48 attribute fsm_encoding : fine terminate for the formula fo</pre>	
Ceneral Properties	<pre>s , actriouce ism_encoding of bit : entity is "one-not"; 50</pre>	~

2. Go to line 65 and make swap the inputs to uReg and xReg. The following code snippet shows the required changes:

From	То
<pre>begin process (clk) begin if rising_edge(clk) then xStepReg <= xStep; uReg <= u; xReg <= u; xReg <= x; end if; end process;</pre>	<pre>begin process (clk) begin if rising_edge(clk) then xStepReg <= xStep; uReg <= u; xReg <= x; uReg <= x; uReg <= u; end if; end process;</pre>

3. Save the changes by clicking the Save File 💾 button in the toolbar of the text editor.

As you can see in the following figure, changing the design source files also changes the run status for finished runs from Complete to Out-of-date.

Tcl Console N	lessages Lo	og Reports Desig	gn Runs	×					
Q ₹ \$		▶							
Name	Constraints	Status		Elapsed	Incremental ^ 1	WNS	TNS	WHS	THS
∨ 🔸 synth_1	constrs_1	Synthesis Out-of-date	e	00:01:23	Off				
🔸 impl_1	constrs_1	Implementation Out-	of-date	00:02:41	Auto(Skipped)	1.452	0	0.062	0
<									



Step 6: Rerunning Synthesis and Implementation

With changes to the RTL source now made, synthesis and implementation must be rerun. As Incremental Implementation has already been configured, all that must be done is to relaunch the tool flow as would be done in the default flow.

1. In the Flow Navigator, click on **Run Implementation**. At this point, all the runs are reset and relaunched.

In the Sources window, the Utility Sources is updated with the checkpoint from the previously routed $impl_l$ if the checkpoint has met certain criteria to ensure it is a good quality reference checkpoint for future runs.



Also updated is the Incremental column in the Design Runs window. This should now say Auto. If the checkpoint did not meet the criteria to be used as a suitable reference, it shows Auto(Skipped) as before.

After implementation is complete, the Design Runs window shows the completed run.

Design Runs										?.	_ D @ X
Q	∎€ ≪	▶ ≫ + %									
Name	Constraints	Status	Elapsed	Incremental	WNS	TNS	WHS	THS	WBSS	TPWS	Total Powe
∨ ✓ synth_1	constrs_1	synth_design Complete!	00:01:02	Off							
✓ impl_1	constrs_1	route_design Complete!	00:01:31	Auto	1.452	0.000	0.062	0.000		0.000	0.41
<							-				>

In the Design Runs window, it is possible to examine runtime and timing criteria. In this case:

• Runtime has reduced for implementation as seen in the Elapsed column.



WNS > 0.000 has been maintained

Note: Synthesis has not been run in a different mode than before so should be similar to before.

The "Elapsed" column measurement includes improvements to the place_design (phys_opt_design is not run here) and route_design and also the extra commands that are required in the incremental flow such as read_checkpoint -incremental and the extra reporting. To see more significant runtime improvements, the flow should be used on larger designs with a good reference checkpoint.

Note: opt_design is not incremental and runtime for opt_design is unimpacted.

Designs that have a complex flow, requiring most effort in the reference run see the highest benefit. Ideally, reference checkpoints are timing-closed, with fewer than 5% of the leaf cells different than the updated design.

2. Select the **Reports** window and double-click the Incremental Reuse Report in the Route Design section, as shown in the following figure.

Tcl Console Messages Log Reports × Design Runs	
Q ≚ ≑ + − ∅ ►	
Report	Report Type
✓ Route Design (route_design)	
impl_1_route_report_drc_0	Report on error or violations against a set of design rule checks (report_drc)
impl_1_route_report_methodology_0	Report on error or violations against a set of methodology checks (report_methodology)
impl_1_route_report_power_0	Report power analysis details (report_power)
impl_1_route_report_route_status_0	Report on status of the routing. (report_route_status)
impl_1_route_report_timing_summary_0	Report timing summary (report_timing_summary)
impl_1_route_report_incremental_reuse_0	Report on achievable incremental reuse for the given design-checkpoint (report_incremental_reuse)
impl_1_route_report_clock_utilization_0	Report information about clock nets in design (report_clock_utilization)
impl_1_route_report_bus_skew_0	Report on calculated bus skew among the signals constrained by set_bus_skew (report_bus_skew)

So far, you have modified the generation of the readEgressFifo so that the signal is zero when the error signal is non-zero. This is a small change to the design so you would expect the reuse to be high. Now examine the Incremental Reuse Report to confirm this is the case.



Incremental Implementation Information

Table of Contents

Reuse Summary
 Reference Checkpoint Information

- 3. Comparison with Reference Run
- 4. Non Reuse Information

1. Reuse Summary

+ Туре +	/ Matched % (of Total)	Reuse % (of Total)	Fixed % (of Total)	Total
Cells	99.07	99.52	1.98	3578
Nets	99.71	98.97	0.00	5189
Pins		89.30	-	19495
Ports	100.00	100.00	100.00	71
+	+	-++		+

* Reused % can exceed matched % when unmatched cells take their placement directly from other cells that have been matched

2. Reference Checkpoint Information

+	
1	
L DCD Location:	C:/Wiwado Tutorial/project bft core/project bft core erce/utile l/importe/impl l/bft routed dop
DCF LOCACION.	<pre>/ c./vivado_idtoiiai/pioject_bit_core/pioject_bit_core.sics/dtits_i/imports/impi_i/bit_iodted.dcp</pre>
1	

+	Value
<pre>Vivado Version Vivado Version DCP State Recorded WNS Recorded WHS Reference Speed File Version Incremental Speed File Version</pre>	2019.1 POST_ROUTE 1.450 0.059 PRODUCTION 1.12 2017-02-17 PRODUCTION 1.12 2017-02-17

* Recorded WNS/WHS timing numbers are estimated timing numbers. They may vary slightly from sign-off timing numbers.

In the report you can confirm that a high percentage of cells, nets and ports are fully reused.

In the Reference Checkpoint Information section you can see information reported on the reference checkpoint. This is useful when the source of the checkpoint is unknown.



3. Comparison with Reference Run

+	-+-				+				+-			+
	į	WNS(ns)		i	Runtime(elapsed)(hh:mm)				Runtime(cpu)(hh:mm)			
Stage	i	Reference	i	Incremental	I	Reference	1	Incremental	ļ	Reference	i	Incremental
<pre>/ synth_design</pre>	i		ï		1	< 1 min	i	00:01	ī	< 1 min	1	00:01
opt_design	I.		I.		L.	00:01	Ľ	00:01	I.	00:01	Ľ	00:01
read_checkpoint	1		I.		L		L	< 1 min	I.		L	< 1 min
place_design	I.	2.209	I.	1.795	L.	< 1 min	Ľ	< 1 min	I.	00:01	Ľ	< 1 min
route_design	I.	1.794	I.	1.793	I.	00:01	L	00:01	I.	00:01	L	00:01

4. Non Reuse Information

+	++			
I Type	%			
+	++			
Non-Reused Cells	0.62			
New	0.62			
Partially reused nets	0.00			
Non-Reused nets	0.54			
Non-Reused Ports	0.00			
+	++			

In the Comparison with Reference Run section, you can see how the runtime and WNS at each stage of the flow compares. This is good for debugging purposes to understand where WNS and runtime diverge when there are issues. Note that these designs are not 100% the same so this information is a only guide.

When run with the RuntimeOptmized directive, the target WNS of this run is either 0.0 or the WNS from the reference run if it is < 0.0. To always target a timing closed design (Target WNS = 0.0 ns), the incremental directive must be set TimingClosure. This can be done by applying the following command:

```
set_property -name INCREMENTAL_CHECKPOINT.MORE_OPTIONS -value {-
incremental_directive TimingClosure} -objects [get_runs impl_1]
```

Conclusion

This concludes Lab 2. You can close the current project and exit the Vivado IDE.

In this lab, you learned how to run the Incremental Implementation portion of the Incremental Compile flow, using a checkpoint from a previously implemented design. You also examined the similarity between a reference design checkpoint and the current design by examining the Incremental Reuse Report.



Chapter 3

Lab 3: Manual and Directed Routing

In this lab, you learn how to use the Vivado[®] IDE to assign routing to nets for precisely controlling the timing of a critical portion of the design.

- You will use the BFT HDL example design that is included in the Vivado[®] Design Suite.
- To illustrate the manual routing feature, you will precisely control the skew within the output bus of the design, wbOutputData.

Step 1: Opening the Example Project

- 1. Start by loading Vivado IDE by doing one of the following:
 - Launch the Vivado IDE from the icon on the Windows desktop.
 - Type vivado from a command terminal.
- 2. From the Getting Started page, click Open Example Project.



3. In the Create an Example Project page, click Next.





🍌 Open Example Project		×
VIVADO HLut Editions	Create an Example Project This wizard will guide you through the creation of a new Vivado project from a predefined template. To create a Vivado project you will specify the type of template project you would like to create. Next, you will need to provide a name and a location for your project files and choose a default board.	
E XILINX	< Back Einish Cancel	

4. In the Select Project Template page, select the BFT (Small RTL project) design, and click Next.





- 5. In the Project Name page, specify the following, and click **Next**:
 - Project name: project_bft_core
 - **Project location:** <Project_Dir>

🝌 Open Example Pr	oject					×
Project Name Enter a name for yo	our project and specify a dire	ctory where the pro	oject data files will l	be stored		A
<u>P</u> roject name:	project_bft_core					\otimes
Project location:	C:/Vivado_Tutorial					
🗹 Create proje	ct subdirectory					
Project will be cr	eated at: C:/Vivado_Tutorial/	project_bft_core				
?			< <u>B</u> ack	<u>N</u> ext >	<u>F</u> inish	Cancel

6. In the Default Part page, select the **xc7k70tfbg484-2** as your Default Part, and click **Next**.


٨.	Open Example Project								×
De	fault Part								
Ch	oose a default Xilinx parl	t for your project.							
	Part	I/O Pin Count	Available IOBs	LUT Elements	FlipFlops	Block RAMs	Ultra RAMs	DSPs	GI
	xc7k70tfbg484-2	484	285	41000	82000	135	0	240	4
	xc7k70tfbg676-2	676	300	41000	82000	135	0	240	8
	xc7v585tffg1157-2	1157	600	364200	728400	795	0	1260	20
	xcku035-fbva900-2-e	900	468	203128	406256	540	0	1700	16
	4								
	·								-
	\mathbf{D}			< <u>B</u> ack	N	ext >	<u>F</u> inish	Can	cel

7. In the New Project Summary page, review the project details, and click Finish.





À Open Example Project	×
HLx Editions	 New Project Summary A new project named 'project_bft_core' will be created from the 'BFT' template. The default part and product family for the new project: Default Part: xc7k70tfbg484-2 Product: Kintex-7 Family: Kintex-7 Package: fbg484 Speed Grade: -2
E XILINX.	To create the project, click Finish
?	< <u>B</u> ack <u>N</u> ext > <u>Finish</u> Cancel

The Vivado IDE opens with the default view.

🔺 project.Mr.com / [Cu/Wado_Tutorial/project.Mr.com/project.Mr.co													
Eile Edit Flow Tools Report	s Window Layout View Help Q- Q	lick Access								Ready			
	t≊ φ Σ ± ∅ x									🖽 Default Layout 🗸 🗸			
Flow Navigator 🗧 🔍 🚊	PROJECT MANAGER - project_bft_core									? ×			
✓ PROJECT MANAGER	Sources	? _ D G X	Project Summary							? 🗆 🗆 X			
Settings		0	Overview Dashboard										
Add Sources	> Design Sources (1)	-	Dational										
Language Templates	> • • bft(aBFT) (bft.vhdl) (20)		Settings Edit							1			
IP Catalog	> Constraints (1)		Project name:	project_bft_core									
Y IPINTECRATOR	> G Utility Sources		Project location:	C://ivado_Tutorial/project	t_bft_core								
Create Block Design			Project part	xc7k70tfbg484-2									
Open Block Design			Top module name:	bft									
Generate Block Design			Target language:	VHDL									
			simulalor language.	Mixed									
 SIMULATION 	Hiararchy Librariae Compila Ordar		Synthesis					Implementation					
Run Simulation	nerarchy Ebranes Completorder												
Y RTI ANALYSIS	Properties	? _ 🗆 🗆 ×	Messages;	Not started No errors or warnings				Status: Messages;	Not started No errors or warnings				
Open Elaborated Design		\leftrightarrow \Rightarrow \Diamond	Part	xc7k70tbg484-2				Part.	xc7k70tfbg484-2				
			Strategy: Vivado Synthesis Defaults					Strategy:	Vivado Implementation Defaults				
✓ SYNTHESIS			Incremental synthesis:	Report Strategy: Vivado Synthesis Default Reports Report St Incremental synthesis: None Increment					Incremental implementation: None				
Run Synthesis													
 Open Synthesized Design 	Select an object to see propertie	,	DRC Violations					Timing					
				Run Implementation	to see DRC results				Run Implementation to see timi	na results			
 Run Implementation 													
> Open Implemented Design			Utilization					Power					
										~			
 PROGRAM AND DEBUG 	Tcl Console Messages Log Reports	Design Runs 🛛 🖂								? _ 🗆 🗳			
Generate Bitstream	Q ± ⊕ ≪ ▶ ≫ +	%											
> Open Hardware Manager	Name Constraints Status	WNS TNS WHS	THS TPWS Total P	ower Failed Routes LU	JT FF BRAMS	URAM DSP	Start Elaps	ed Run Strategy		Report Strategy			
	∨ ▷ synth_1 constrs_1 Not started							Vivado Synthesis Defaults (/ivado Synthesis 2019)	Vivado Synthesis Default Reports (Vivado Synthe			
	D Impi_1 consuls_1 Not stated							vivado imprementation Dera	uns (vivado implementation 2019)	vivado imprementarion Deraut Reports (vivado il			
	<									>			



Step 2: Performing Place and Route on the Design

1. In the Flow Navigator, click **Run Implementation**.

The Missing Synthesis Results dialog box opens to inform you that there is no synthesized netlist to implement, and prompts you to start synthesis first.



2. Click OK to launch synthesis first.

Implementation automatically starts after synthesis completes, and the Implementation Completed dialog box opens when complete, as shown in the following figure.

mplementation Completed	×									
Implementation successfully completed.										
Next										
Open Implemented Design										
◯ <u>G</u> enerate Bitstream										
◯ <u>V</u> iew Reports										
Don't show this dialog again										
OK Cancel										

3. In the Implementation Completed dialog box, select **Open Implemented Design** and click **OK**.

The Device window opens, displaying the placement results.

4. Click the Routing Resources button is to view the detailed routing resources in the Device window.





Step 3: Analyzing Output Bus Timing

IMPORTANT! The tutorial design has an output data bus, *wbOutputData*, which feeds external logic. Your objective is to precisely control timing skew by manually routing the nets of this bus.

You can use the Report Datasheet command to analyze the current timing of members of the output bus, wbOutputData. The Report Datasheet command lets you analyze the timing of a group of ports with respect to a specific reference port.

- 1. From the main menu, select **Reports** \rightarrow **Timing** \rightarrow **Report Datasheet**.
- 2. Select the **Groups** tab in the Report Datasheet dialog box, as seen in the following figure, and enter the following:
 - Reference: [get_ports {wbOutputData[0]}]
 - Ports: [get_ports {wbOutputData[*]}]



A	Report Data	sheet					\times		
C of	reate a datas a design at	sheet report the I/O pads	for the current design. The da	atasheet has the ti	ming characte	eristics			
	Options Groups Timer Settings								
	1. Reference	ce [get_po	ts {wbOutputData[0]}]		8	••••			
	Por	ts [get_po	ts {wbOutputData[*]}]		8	+ -			
	Command:	t -group "[و	et_ports {wbOutputData[0]}] [get_ports {wbOutp	utData[*]}]" -n	ame timing_	1		
	🕑 Open in	a new ta <u>b</u>							
	Open in	Ti <u>m</u> ing Anal	sis layout						
(?				ОК	Cancel			

3. Click OK.

In this case, you are examining the timing at the ports carrying the <code>wbOutputData</code> bus, relative to the first bit of the bus, <code>wbOutputData[0]</code>. This allows you to quickly determine the relative timing differences between the different bits of the bus.

- 4. Click the Maximize D button to maximize the Timing Datasheet window and expand the results.
- 5. Select the Max/Min Delays for Groups → Clocked by wbClk → wbOutputData[0] section, as seen in the following figure.

You can see from the report that the timing skew across the wbOutputData bus varies by almost 660 ps. The goal is to minimize the skew across the bus to less than 100 ps.



Q ¥ C W MaxMini Delays for Groups - Clocked by WCIK - wtoOutputData[U] General Information Input Ports SetupHold Output Ports SetupHold Output Ports (SetupHold Output Ports (SetupHold Setup Detwer Clocks Pad Max Max Max Pads Max	Timing									?	7
Ceneral Information Input Ports Setup-Hold Output Ports (Societ-bout Setup-between Clocks Pad Max Edge Max Edge Max Correre Man Edge	Q 素 ♦ C 📕	Q 🚯 Max/Min Del	ays for G	roups - C	Clocked by wbCl	k - wbOut	putData	[0]			
Output Ports Clock-boult Stubp between Clocks 9 woodputData[2] 9 221 Rise 8LOW 4 000 Rise FAST 0.659 > Setup/Hold for input Buses · Clocked by wClk · woodputData[2] 9 220 Rise SLOW 4 023 Rise FAST 0.664 · MaxNin Delgs for Output Data[2] 9 202 Rise SLOW 4 023 Rise FAST 0.568 · Clocked by wClk · woodputData[2] 9 202 Rise SLOW 3 997 Rise FAST 0.568 · Clocked by wClk · woodputData[2] 9 100 Rise SLOW 3 997 Rise FAST 0.568 · Clocked by wClk · woodputData[2] 9 100 Rise SLOW 3 997 Rise FAST 0.588 · Clocked by wClk · woodputData[2] 8 101 Rise SLOW 3 897 Rise FAST 0.582 · Clocked by wClk · woodputData[2] 8 281 Rise SLOW 3 801 Rise FAST 0.402 · Woodp	General Information	Pad	Max Delay	Max Edge	Max Process Corner	Min Delay	Min Edge	Min Process Corner	E ~ 1		
Satup between Clocks 9 w00dupulData[31] 9.249 Rise 9.000 4.054 Rise FAST 0.627 Satup/hold for input Buses 9 w00dupulData[27] 9.214 Rise S.UOW 4.029 Rise FAST 0.652 MaxMin Delays for ObupulData 9 w00dupulData[27] 9.214 Rise S.UOW 3.997 Rise FAST 0.562 V MoxMin Delays for ObupulData 9 w00dupulData[23] 9.079 Rise S.UOW 3.997 Rise FAST 0.450 V MoxMin Delays for ObupulData[23] 9.079 Rise S.UOW 3.997 Rise FAST 0.457 V MoxMin Delays for ObupulData[23] 9.071 Rise S.UOW 3.891 Rise FAST 0.392 V MoxMinData[21] 8.914 Rise S.UOW 3.871 Rise FAST 0.323 WoudputData[21] 8.914 Rise S.UOW 3.817 Rise FAST 0.323 WoudputData[18] 8.917 Rise S.UOW 3.818	Output Ports Clock-to-out	🕢 wbOutputData[28]	9.281	Rise	SLOW	4.060	Rise	FAST	0.659		
SetupHold for input Buses	Setup between Clocks	🕢 wbOutputData[31]	9.249	Rise	SLOW	4.054	Rise	FAST	0.627		
Clocked by wbCik I wbOuburDatai221 92.44 Rise SLOW 4.023 Rise FAST 0.592 MaXMin Delays for CouburDatai224 9.100 Rise SLOW 3.997 Rise FAST 0.568 V Clocked by wbCik I wbOuburDatai223 9.070 Rise SLOW 3.997 Rise FAST 0.568 MaXMin Delays for Coupu I wbOuburDatai223 9.077 Rise SLOW 3.947 Rise FAST 0.367 MaXMin Delays for Coupu I wbOuburDatai223 9.071 Rise SLOW 3.947 Rise FAST 0.362 MaXMin Delays for Coupu I wbOuburDatai221 8.948 Rise SLOW 3.874 Rise FAST 0.362 I wbOuburDatai221 8.948 Rise SLOW 3.874 Rise FAST 0.362 I wbOuburDatai212 8.948 Rise SLOW 3.872 Rise FAST 0.162 I wbOuburDatai121 8.821 Rise SLOW 3.822 Ri	✓ Setup/Hold for Input Buses	🕢 wbOutputData[30]	9.226	Rise	SLOW	4.029	Rise	FAST	0.604		
winipulData @ woodupulData[26] 9.202 Rise SLOW 3.997 Rise FAST 0.580 ^ MaxMiin Delays for OrupulData @ woodupulData[23] 9.190 Rise SLOW 3.949 Rise FAST 0.580 ^ MaxMiin Delays for Groups @ woodupulData[23] 9.079 Rise SLOW 3.949 Rise FAST 0.650 ^ WoodupulData[20] 9.011 Rise SLOW 3.849 Rise FAST 0.362 ^ WoodupulData[20] 8.911 Rise SLOW 3.847 Rise FAST 0.362 @ woodupulData[21] 8.926 Rise SLOW 3.877 Rise FAST 0.323 @ woodupulData[21] 8.926 Rise SLOW 3.874 Rise FAST 0.117 @ woodupulData[11] 8.818 Rise SLOW 3.874 Rise FAST 0.125 @ woodupulData[11] 8.716 Rise SLOW 3.817 Rise FAST 0.117	Clocked by wbClk	🕢 wbOutputData[27]	9.214	Rise	SLOW	4.023	Rise	FAST	0.592		
 ✓ MaxMin Delays for Output Blues ✓ Clocked by wbCik ✓ Moduput Data[23] Ø.130 Rise SLOW Ø.998 Rise FAST Ø.568 ✓ Moduput Data[23] Ø.079 Rise SLOW Ø.999 Rise FAST Ø.568 Ø.004put Data[23] Ø.079 Rise SLOW Ø.999 Rise FAST Ø.568 Ø.004put Data[23] Ø.011 Rise SLOW Ø.999 Rise FAST Ø.568 Ø.004put Data[23] Ø.011 Rise SLOW Ø.999 Rise FAST Ø.362 Ø.004put Data[22] Ø.942 Rise SLOW Ø.939 Rise FAST Ø.362 Ø.004put Data[22] Ø.942 Rise SLOW Ø.897 Rise FAST Ø.362 Ø.904put Data[22] Ø.942 Rise SLOW Ø.897 Rise FAST Ø.362 Ø.904put Data[21] Ø.822 Rise SLOW Ø.817 Rise FAST Ø.940 O.940 Rise FAST Ø.940 Ø.9404put Data[11] Ø.813 Rise SLOW Ø.810 Rise FAST Ø.940 Ø.9404put Data[11] Ø.747 Rise SLOW Ø.773 Rise FAST Ø.9404put Data[11] Ø.722 Rise SLOW Ø.773 Rise FAST	wbinputData	🕢 wbOutputData[26]	9.202	Rise	SLOW	3.997	Rise	FAST	0.580		
	\sim Max/Min Delays for Output Buses	wbOutputData[24]	9.190	Rise	SLOW	3.998	Rise	FAST	0.568		
wb0.utputData 9.079 Rise SLOW 3.949 Rise FAST 0.457 MaxMilin Delays for Groups wb0.utputData[22] 9.011 Rise SLOW 3.929 Rise FAST 0.389 wb0.utputData[25] 8.984 Rise SLOW 3.887 Rise FAST 0.340 wb0.utputData[21] 8.952 Rise SLOW 3.874 Rise FAST 0.340 @wb0.utputData[21] 8.952 Rise SLOW 3.874 Rise FAST 0.340 @wb0.utputData[21] 8.952 Rise SLOW 3.874 Rise FAST 0.340 @wb0.utputData[21] 8.928 Rise SLOW 3.822 Rise FAST 0.179 @wb0.utputData[18] 8.810 Rise SLOW 3.827 Rise FAST 0.162 @wb0.utputData[19] 8.708 Rise SLOW 3.774 Rise FAST 0.162 @wb0.utputData[11] 8.727 Rise SLOW 3.774 Rise FAST 0.101 @wb0.utputData[11] </td <td>Clocked by wbClk</td> <td>🕢 wbOutputData[29]</td> <td>9.130</td> <td>Rise</td> <td>SLOW</td> <td>3.971</td> <td>Rise</td> <td>FAST</td> <td>0.508</td> <td></td> <td></td>	Clocked by wbClk	🕢 wbOutputData[29]	9.130	Rise	SLOW	3.971	Rise	FAST	0.508		
✓ MaxMin Delays for Groups ✓ Clockad by wbCik wbOutputDatal(2) 8.944 Rise SLOW 3.807 Rise FAST 0.389 ✓ MoutputDatal(2) 8.944 Rise SLOW 3.807 Rise FAST 0.340 ④ wbOutputDatal(2)1 8.942 Rise SLOW 3.807 Rise FAST 0.205 ● wbOutputDatal(2)1 8.942 Rise SLOW 3.807 Rise FAST 0.206 ● wbOutputDatal(2)1 8.843 Rise SLOW 3.822 Rise FAST 0.191 ● wbOutputDatal(1)1 8.813 Rise SLOW 3.802 Rise FAST 0.162 ● wbOutputDatal(1)1 8.716 Rise SLOW 3.707 Rise FAST 0.149 ● wbOutputDatal(1)1 8.727 Rise SLOW 3.707 Rise FAST 0.149 ● wbOutputDatal(1)1 8.727 Rise SLOW 3.763 Rise FAST 0.100 ● wbOutputDatal(1)1 8.727 Rise	wbOutputData	🕢 wbOutputData[23]	9.079	Rise	SLOW	3.949	Rise	FAST	0.457		
Clocked by wbClk wbOudputData[21] 8.984 Rise SLOW 3.897 Rise FAST 0.362 @ wbOudputData[21] 8.962 Rise SLOW 3.874 Rise FAST 0.340 @ wbOudputData[21] 8.962 Rise SLOW 3.874 Rise FAST 0.323 @ wbOudputData[12] 8.828 Rise SLOW 3.827 Rise FAST 0.120 @ wbOudputData[12] 8.828 Rise SLOW 3.827 Rise FAST 0.191 @ wbOudputData[13] 8.714 Rise SLOW 3.827 Rise FAST 0.179 @ wbOudputData[13] 8.714 Rise SLOW 3.827 Rise FAST 0.119 @ wbOudputData[14] 8.747 Rise SLOW 3.774 Rise FAST 0.118 @ wbOudputData[16] 8.717 Rise SLOW 3.768 Rise FAST 0.100 @ wbOudputData[16] 8.714 Rise SL	✓ Max/Min Delays for Groups	🕑 wbOutputData[20]	9.011	Rise	SLOW	3.929	Rise	FAST	0.389		
wbOutputData[21] 8.962 Rise SLOW 3.874 Rise FAST 0.340 @ wbOutputData[22] 8.945 Rise SLOW 3.870 Rise FAST 0.323 @ wbOutputData[12] 8.628 Rise SLOW 3.821 Rise FAST 0.223 @ wbOutputData[18] 8.813 Rise SLOW 3.821 Rise FAST 0.101 @ wbOutputData[18] 8.813 Rise SLOW 3.822 Rise FAST 0.179 @ wbOutputData[19] 8.764 Rise SLOW 3.801 Rise FAST 0.1149 @ wbOutputData[13] 8.771 Rise SLOW 3.771 Rise FAST 0.125 @ wbOutputData[14] 8.747 Rise SLOW 3.772 Rise FAST 0.116 @ wbOutputData[14] 8.722 Rise SLOW 3.765 Rise FAST 0.100 @ wbOutputData[11] 8.522 Rise SLOW 3.765	Clocked by wbClk	🚽 wbOutputData[25]	8.984	Rise	SLOW	3.897	Rise	FAST	0.362		
• wbOutputData[22] 8.945 Rise SLOW 3.870 Rise FAST 0.323 • wbOutputData[12] 8.828 Rise SLOW 3.824 Rise FAST 0.205 • wbOutputData[18] 8.83 Rise SLOW 3.824 Rise FAST 0.191 • wbOutputData[6] 8.801 Rise SLOW 3.801 Rise FAST 0.162 • wbOutputData[13] 8.771 Rise SLOW 3.801 Rise FAST 0.149 • wbOutputData[14] 8.777 Rise SLOW 3.803 Rise FAST 0.149 • wbOutputData[14] 8.777 Rise SLOW 3.803 Rise FAST 0.149 • wbOutputData[14] 8.777 Rise SLOW 3.773 Rise FAST 0.116 • wbOutputData[11] 8.722 Rise SLOW 3.763 Rise FAST 0.100 • wbOutputData[11] 8.722 Rise SLOW 3.645 Rise FAST 0.001 • wbOutputData[16] 8.714 Ri	wbOutputData[0]	🕢 wbOutputData[21]	8.962	Rise	SLOW	3.874	Rise	FAST	0.340		
Image: series of the series		🕢 wbOutputData[22]	8.945	Rise	SLOW	3.870	Rise	FAST	0.323		
Image: second		wbOutputData[12]	8.828	Rise	SLOW	3.821	Rise	FAST	0.206		
Image: method matrix 8.801 Rise SLOW 3.822 Rise FAST 0.179 Image: method matrix 8.764 Rise SLOW 3.801 Rise FAST 0.162 Image: method matrix 8.771 Rise SLOW 3.808 Rise FAST 0.149 Image: method matrix 8.771 Rise SLOW 3.772 Rise FAST 0.138 Image: method matrix 8.771 Rise SLOW 3.772 Rise FAST 0.125 Image: method matrix 8.774 Rise SLOW 3.778 Rise FAST 0.110 Image: method matrix 8.722 Rise SLOW 3.758 Rise FAST 0.100 Image: method matrix 8.722 Rise SLOW 3.758 Rise FAST 0.100 Image: method matrix 8.722 Rise SLOW 3.758 Rise FAST 0.010 Image: method matrix 8.717 Rise SLOW 3.753 Rise FAST 0.021 Image: method matrix <td< td=""><td></td><td>wbOutputData[18]</td><td>8.813</td><td>Rise</td><td>SLOW</td><td>3.784</td><td>Rise</td><td>FAST</td><td>0.191</td><td></td><td></td></td<>		wbOutputData[18]	8.813	Rise	SLOW	3.784	Rise	FAST	0.191		
Image: wboutputData[9] 8.764 Rise SLOW 3.801 Rise FAST 0.162 Image: wboutputData[13] 8.771 Rise SLOW 3.808 Rise FAST 0.149 Image: wboutputData[14] 8.760 Rise SLOW 3.773 Rise FAST 0.138 Image: wboutputData[14] 8.747 Rise SLOW 3.774 Rise FAST 0.125 Image: wboutputData[11] 8.738 Rise SLOW 3.782 Rise FAST 0.116 Image: wboutputData[11] 8.732 Rise SLOW 3.768 Rise FAST 0.100 Image: wboutputData[11] 8.522 Rise SLOW 3.764 Rise FAST 0.100 Image: wboutputData[1] 8.522 Rise SLOW 3.753 Rise FAST 0.001 Image: wboutputData[1] 8.522 Rise SLOW 3.753 Rise FAST 0.0095 Image: wboutputData[16] 8.717 Rise SLOW 3.766 Rise FAST 0.008 Image:		🕢 wbOutputData[6]	8.801	Rise	SLOW	3.822	Rise	FAST	0.179		
Image: WbOutputData[13] 8.771 Rise SLOW 3.808 Rise FAST 0.149 Image: WbOutputData[14] 8.760 Rise SLOW 3.773 Rise FAST 0.138 Image: WbOutputData[14] 8.747 Rise SLOW 3.774 Rise FAST 0.125 Image: WbOutputData[14] 8.747 Rise SLOW 3.782 Rise FAST 0.116 Image: WbOutputData[11] 8.738 Rise SLOW 3.768 Rise FAST 0.100 Image: WbOutputData[15] 8.722 Rise SLOW 3.768 Rise FAST 0.100 Image: WbOutputData[1] 8.522 Rise SLOW 3.753 Rise FAST 0.009 Image: WbOutputData[1] 8.522 Rise SLOW 3.753 Rise FAST 0.009 Image: WbOutputData[16] 8.717 Rise SLOW 3.766 Rise FAST 0.008 Image: WbOutputData[16] 8.714 Rise SLOW 3.766 Rise FAST 0.0014 Image:		🕢 wbOutputData[9]	8.784	Rise	SLOW	3.801	Rise	FAST	0.162		
Image: WbOutputData[19] 8,760 Rise SLOW 3,773 Rise FAST 0,138 Image: WbOutputData[14] 8,747 Rise SLOW 3,774 Rise FAST 0,125 Image: WbOutputData[11] 8,738 Rise SLOW 3,782 Rise FAST 0,116 Image: WbOutputData[15] 8,722 Rise SLOW 3,782 Rise FAST 0,100 Image: WbOutputData[1] 8,522 Rise SLOW 3,645 Rise FAST 0,100 Image: WbOutputData[1] 8,522 Rise SLOW 3,645 Rise FAST 0,100 Image: WbOutputData[1] 8,522 Rise SLOW 3,753 Rise FAST 0,009 Image: WbOutputData[16] 8,717 Rise SLOW 3,753 Rise FAST 0,092 Image: WbOutputData[17] 8,657 Rise SLOW 3,766 Rise FAST 0,065 Image: WbOutputData[7] 8,676 Rise SLOW 3,736 Rise FAST 0,053 Image: Wb		🕢 wbOutputData[13]	8.771	Rise	SLOW	3.808	Rise	FAST	0.149		
Image: WbOutputData[14] 8.747 Rise SLOW 3.774 Rise FAST 0.125 Image: WbOutputData[11] 8.738 Rise SLOW 3.782 Rise FAST 0.116 Image: WbOutputData[15] 8.722 Rise SLOW 3.762 Rise FAST 0.100 Image: WbOutputData[15] 8.722 Rise SLOW 3.645 Rise FAST 0.100 Image: WbOutputData[11] 8.522 Rise SLOW 3.645 Rise FAST 0.100 Image: WbOutputData[11] 8.522 Rise SLOW 3.753 Rise FAST 0.095 Image: WbOutputData[2] 8.717 Rise SLOW 3.763 Rise FAST 0.092 Image: WbOutputData[16] 8.717 Rise SLOW 3.763 Rise FAST 0.081 Image: WbOutputData[10] 8.517 Rise SLOW 3.764 Rise FAST 0.053 Image: WbOutputData[7] 8.677 Rise SLOW 3.736 Rise FAST 0.052 Image:		🕢 wbOutputData[19]	8.760	Rise	SLOW	3.773	Rise	FAST	0.138		
Image: WbOutputData[11] 8.738 Rise SLOW 3.782 Rise FAST 0.116 Image: WbOutputData[15] 8.722 Rise SLOW 3.758 Rise FAST 0.100 Image: WbOutputData[11] 8.522 Rise SLOW 3.645 Rise FAST 0.100 Image: WbOutputData[1] 8.522 Rise SLOW 3.645 Rise FAST 0.100 Image: WbOutputData[16] 8.717 Rise SLOW 3.753 Rise FAST 0.092 Image: WbOutputData[3] 8.534 Rise SLOW 3.764 Rise FAST 0.092 Image: WbOutputData[3] 8.534 Rise SLOW 3.768 Rise FAST 0.088 Image: WbOutputData[10] 8.514 Rise SLOW 3.666 Rise FAST 0.065 Image: WbOutputData[7] 8.675 Rise SLOW 3.736 Rise FAST 0.053 Image: WbOutputData[7] 8.674 Rise SLOW 3.737 Rise FAST 0.052 Image: WbO		wbOutputData[14]	8.747	Rise	SLOW	3.774	Rise	FAST	0.125		
Image: WbOutputData[15] 8.722 Rise SLOW 3.758 Rise FAST 0.100 Image: WbOutputData[1] 8.522 Rise SLOW 3.645 Rise FAST 0.100 Image: WbOutputData[8] 8.717 Rise SLOW 3.753 Rise FAST 0.095 Image: WbOutputData[6] 8.717 Rise SLOW 3.753 Rise FAST 0.092 Image: WbOutputData[16] 8.714 Rise SLOW 3.763 Rise FAST 0.092 Image: WbOutputData[3] 8.534 Rise SLOW 3.666 Rise FAST 0.092 Image: WbOutputData[10] 8.514 Rise SLOW 3.667 Rise FAST 0.088 Image: WbOutputData[17] 8.677 Rise SLOW 3.736 Rise FAST 0.053 Image: WbOutputData[2] 8.674 Rise SLOW 3.737 Rise FAST 0.052 Image: WbOutputData[2] 8.674 Rise SLOW 3.736 Rise FAST 0.028 Image: WbOu		wbOutputData[11]	8.738	Rise	SLOW	3.782	Rise	FAST	0.116		
Image: WbOutputData[1] 8.522 Rise SLOW 3.645 Rise FAST 0.100 Image: WbOutputData[8] 8.717 Rise SLOW 3.753 Rise FAST 0.095 Image: WbOutputData[16] 8.714 Rise SLOW 3.753 Rise FAST 0.095 Image: WbOutputData[16] 8.714 Rise SLOW 3.753 Rise FAST 0.092 Image: WbOutputData[3] 8.534 Rise SLOW 3.646 Rise FAST 0.088 Image: WbOutputData[10] 8.541 Rise SLOW 3.646 Rise FAST 0.081 Image: WbOutputData[17] 8.667 Rise SLOW 3.736 Rise FAST 0.065 Image: WbOutputData[2] 8.674 Rise SLOW 3.736 Rise FAST 0.052 Image: WbOutputData[2] 8.674 Rise SLOW 3.736 Rise FAST 0.028 Image: WbOutputData[2] 8.642 Rise SLOW 3.736 Rise FAST 0.021 Image: WbOu		🕢 wbOutputData[15]	8.722	Rise	SLOW	3.758	Rise	FAST	0.100		
Image: WbOutputData[8] 8.717 Rise SLOW 3.753 Rise FAST 0.095 Image: WbOutputData[16] 8.714 Rise SLOW 3.753 Rise FAST 0.092 Image: WbOutputData[3] 8.534 Rise SLOW 3.646 Rise FAST 0.092 Image: WbOutputData[3] 8.534 Rise SLOW 3.646 Rise FAST 0.088 Image: WbOutputData[10] 8.514 Rise SLOW 3.667 Rise FAST 0.081 Image: WbOutputData[10] 8.541 Rise SLOW 3.708 Rise FAST 0.065 Image: WbOutputData[17] 8.687 Rise SLOW 3.736 Rise FAST 0.053 Image: WbOutputData[7] 8.674 Rise SLOW 3.735 Rise FAST 0.052 Image: WbOutputData[2] 8.674 Rise SLOW 3.736 Rise FAST 0.028 Image: WbOutputData[4] 8.649 Rise SLOW 3.736 Rise FAST 0.021 Image: WbOu		🕢 wbOutputData[1]	8.522	Rise	SLOW	3.645	Rise	FAST	0.100		
Image: WbOutputData[16] 8.714 Rise SLOW 3.753 Rise FAST 0.092 Image: WbOutputData[3] 8.534 Rise SLOW 3.646 Rise FAST 0.088 Image: WbOutputData[10] 8.541 Rise SLOW 3.667 Rise FAST 0.081 Image: WbOutputData[10] 8.541 Rise SLOW 3.667 Rise FAST 0.081 Image: WbOutputData[17] 8.687 Rise SLOW 3.708 Rise FAST 0.065 Image: WbOutputData[7] 8.675 Rise SLOW 3.736 Rise FAST 0.053 Image: WbOutputData[7] 8.674 Rise SLOW 3.735 Rise FAST 0.052 Image: WbOutputData[2] 8.674 Rise SLOW 3.730 Rise FAST 0.028 Image: WbOutputData[4] 8.649 Rise SLOW 3.736 Rise FAST 0.028 Image: WbOutputData[5] 8.642 Rise SLOW 3.746 Rise FAST 0.000 Image: WbOu		🕢 wbOutputData[8]	8.717	Rise	SLOW	3.753	Rise	FAST	0.095		
Image: WbOutputData[3] 8.534 Rise SLOW 3.646 Rise FAST 0.088 Image: WbOutputData[10] 8.541 Rise SLOW 3.667 Rise FAST 0.081 Image: WbOutputData[17] 8.687 Rise SLOW 3.708 Rise FAST 0.065 Image: WbOutputData[17] 8.687 Rise SLOW 3.708 Rise FAST 0.053 Image: WbOutputData[7] 8.675 Rise SLOW 3.735 Rise FAST 0.053 Image: WbOutputData[2] 8.674 Rise SLOW 3.735 Rise FAST 0.052 Image: WbOutputData[2] 8.674 Rise SLOW 3.730 Rise FAST 0.052 Image: WbOutputData[4] 8.649 Rise SLOW 3.736 Rise FAST 0.028 Image: WbOutputData[5] 8.642 Rise SLOW 3.746 Rise FAST 0.000 Image: WbOutputData[0] 8.622 Rise SLOW 3.645 Rise FAST 0.059 Image: WbOut		🕢 wbOutputData[16]	8.714	Rise	SLOW	3.753	Rise	FAST	0.092		
Image: WbOutputData[10] 8.541 Rise SLOW 3.667 Rise FAST 0.081 Image: WbOutputData[17] 8.687 Rise SLOW 3.708 Rise FAST 0.065 Image: WbOutputData[17] 8.675 Rise SLOW 3.708 Rise FAST 0.053 Image: WbOutputData[7] 8.674 Rise SLOW 3.735 Rise FAST 0.053 Image: WbOutputData[2] 8.674 Rise SLOW 3.735 Rise FAST 0.052 Image: WbOutputData[2] 8.674 Rise SLOW 3.709 Rise FAST 0.052 Image: WbOutputData[4] 8.649 Rise SLOW 3.736 Rise FAST 0.028 Image: WbOutputData[5] 8.642 Rise SLOW 3.746 Rise FAST 0.021 Image: WbOutputData[0] 8.622 Rise SLOW 3.645 Rise FAST 0.059 Image: WbOutputData[0] 8.622 Rise SLOW 3.645 Rise FAST 0.059		🕢 wbOutputData[3]	8.534	Rise	SLOW	3.646	Rise	FAST	0.088		
Image: WbOutputData[17] 8.687 Rise SLOW 3.708 Rise FAST 0.065 Image: WbOutputData[7] 8.675 Rise SLOW 3.736 Rise FAST 0.053 Image: WbOutputData[7] 8.674 Rise SLOW 3.735 Rise FAST 0.053 Image: WbOutputData[2] 8.674 Rise SLOW 3.735 Rise FAST 0.052 Image: WbOutputData[2] 8.649 Rise SLOW 3.709 Rise FAST 0.028 Image: WbOutputData[5] 8.642 Rise SLOW 3.736 Rise FAST 0.021 Image: WbOutputData[0] 8.622 Rise SLOW 3.746 Rise FAST 0.000 Image: WbOutputData[0] 8.622 Rise SLOW 3.645 Rise FAST 0.059 Image: WbOutputData[0] 8.622 Rise SLOW 3.645 Rise FAST 0.659		wbOutputData[10]	8.541	Rise	SLOW	3.667	Rise	FAST	0.081		
Image: WbOutputData[7] 8.675 Rise SLOW 3.736 Rise FAST 0.053 Image: WbOutputData[2] 8.674 Rise SLOW 3.735 Rise FAST 0.052 Image: WbOutputData[2] 8.674 Rise SLOW 3.735 Rise FAST 0.052 Image: WbOutputData[4] 8.649 Rise SLOW 3.709 Rise FAST 0.028 Image: WbOutputData[5] 8.642 Rise SLOW 3.736 Rise FAST 0.021 Image: WbOutputData[0] 8.622 Rise SLOW 3.746 Rise FAST 0.000 Image: WbOutputData[0] 8.622 Rise SLOW 3.645 Rise FAST 0.000 Image: WbOutputData[0] 8.622 Rise SLOW 3.645 Rise FAST 0.059		🕢 wbOutputData[17]	8.687	Rise	SLOW	3.708	Rise	FAST	0.065		
Image: wboutputData[2] 8.674 Rise SLOW 3.735 Rise FAST 0.052 Image: wboutputData[4] 8.649 Rise SLOW 3.709 Rise FAST 0.028 Image: wboutputData[5] 8.642 Rise SLOW 3.736 Rise FAST 0.021 Image: wboutputData[0] 8.622 Rise SLOW 3.716 Rise FAST 0.000 Image: wboutputData[0] 8.622 Rise SLOW 3.746 Rise FAST 0.000 Image: wboutputData[0] 9.281 Rise SLOW 3.645 Rise FAST 0.659		🕢 wbOutputData[7]	8.675	Rise	SLOW	3.736	Rise	FAST	0.053		
Image: WbOutputData[4] 8.649 Rise SLOW 3.709 Rise FAST 0.028 Image: WbOutputData[5] 8.642 Rise SLOW 3.736 Rise FAST 0.021 Image: WbOutputData[0] 8.622 Rise SLOW 3.716 Rise FAST 0.000 Image: WbOutputData[0] 8.622 Rise SLOW 3.645 Rise FAST 0.000 Image: WbOutputData[0] 9.281 Rise SLOW 3.645 Rise FAST 0.659		🕢 wbOutputData[2]	8.674	Rise	SLOW	3.735	Rise	FAST	0.052		
✓ wbOutputData[5] 8.642 Rise SLOW 3.736 Rise FAST 0.021 ✓ wbOutputData[0] 8.622 Rise SLOW 3.716 Rise FAST 0.000 Worst Case Summary 9.281 Rise SLOW 3.645 Rise FAST 0.659		🕢 wbOutputData[4]	8.649	Rise	SLOW	3.709	Rise	FAST	0.028		
Image: wboutputData[0] 8.622 Rise SLOW 3.716 Rise FAST 0.000 Worst Case Summary 9.281 Rise SLOW 3.645 Rise FAST 0.659		🚽 wbOutputData[5]	8.642	Rise	SLOW	3.736	Rise	FAST	0.021		
Worst Case Summary9.281RiseSLOW3.645RiseFAST0.659		🚽 wbOutputData[0]	8.622	Rise	SLOW	3.716	Rise	FAST	0.000		
		Worst Case Summary	9.281	Rise	SLOW	3.645	Rise	FAST	0.659		
Bus Skew: 0.659 ns Timing Summary-impl 1 (saved) × Datasheet timing 1 ×	Timing Summary - impl 1 (saved)	Bus Skew: 0.659 ns	×								

- 6. Click the Restore 🗗 button so you can simultaneously see the Device window and the Timing Datasheet results.
- 7. Click the hyperlink for the Max Delay of the source wbOutputData[28].

This highlights the path in the Device window that is currently open.

Note: Ensure the Autofit Selection \bigcirc is highlighted in the Device window so you can see the entire path, as shown in the following figure.



> project_bft_core - [C:/Vivado_Tutonal/]	/project_bit_core/project_bit_core.pprj - Wwado 2018.3	- U X
Eile Edit Flow Tools Rep	pgrts Window Layout View Help Q- Quick Access	Implementation Complete 🗸
	× ∞ ▶ 蒜 坊 1 10 倍 \$ Σ 羔 ∅ ¥	🗉 Default Layout 🗸 🗸
Flow Navigator ± 0 ? _	IMPLEMENTED DE SIGN - sc?r2/0tbp484-2 (active)	? ×
V PROJECT MANAGER		
Settings	Sources Netist X Y LL Project summary X Uevrice X	7 L L
Add Sources		0
Language Templates	wboutputData_reg(25) (FDRE)	^
C IP Catalog	wooduputda_reg28 (FORE)	
	woodpubsa_equ/(vole)	
IP INTEGRATOR	wb0ubpuData_reg(28) (FDRE)	
Create Block Design	wb0utputData_reg[30](FDRE)	
Open Block Design	woutputData_reg(31(FORE)	
Generate Block Design	WWWWEQUEUL_INS(IGOF)	
	> 1 and2 (cond_2)	
 SIMULATION 	> 101 and3 (round_3)	
Run Simulation	> [] and4 (round_4)	
) III egress.cop() egressm (n/neutren)	
 RTL ANALYSIS A Construction of Decision 		
> Open Elaborated Design	Path Properties ? _ D C X	
 SYNTHESIS 	L Pan 5 ← → ◊	
Run Synthesis	Summary A	
> Open Synthesized Design	L Name & Path 5	
	a sauce in wolds we down and a same and a	
IMPLEMENTATION	Destination 🕼 wbOutputData(28) (output port)	
Run Implementation	Path Group (none)	
 Open Implemented Design 	Path Tupe Max at Slow Process Corner Y	
Constraints Wizard	General Properties Report Cells Nets () → Ξ	ži do se
Edit Timing Constraints		
😇 Report Timing Summary	TclConsole Messages Log Reports Design Runs Power DRC Methodology Timing ×	? _ 🗆 🖸
Report Clock Networks	Q ± ⊕ C ₩ Q Maximin Delays for Groups - Clocked by wbClk - wbOutputData[0]	
Report Clock Interaction	General Information Pad Max Max Max Frocess Min Min Min Process E 1	
Report Methodology	Input Parts Setup/Hold Develop Experiment Develop Exper	^
Report DRC	Setup between Clocks 4 wooutputData[31] 9.249 Rise SLOW 4.054 Rise FAST 0.627	
Report Noise	✓ Setup/Hold for Input Buses ≪ wbChitruit/Data1300 9 226 Rise 8 ISIN 4 029 Rise FAST 0 604	~
Report Bilitation	Clocked by wbClk Bus Skew: 0.659 ns	
Report Oditzation	Timing Summary-Imp[_1(zaved) × Datasheet-Iming_1 ×	
Keport Power *	AT P	

8. In the Device window, right-click on the highlighted path and select Schematic from the popup menu.

This displays the schematic for the selected output data bus. From the schematic, you can see that the output port is directly driven from a register through an output buffer (OBUF).

If you can consistently control the placement of the register with respect to the output pins on the bus and control the routing between registers and the outputs, you can minimize skew between the members of the output bus.



9. Change to the Device window.

To better visualize the placement of the registers and outputs, you can use the $mark_objects$ command to mark them in the Device window.



10. From the Tcl Console, type the following commands:

```
mark_objects -color blue [get_ports wbOutputData[*]]
mark_objects -color red [get_cells wbOutputData_reg[*]]
```

Blue diamond markers show on the output ports, and red diamond markers show on the registers feeding the outputs, as seen in the following figure.



The outputs marked in blue are spread out along two banks on the left side starting with wbOutputData[0] (on the bottom) and ending with wbOutputData[31] (at the top), while the output registers marked in red are clustered close together on the right.

To look at all of the routing from the registers to the outputs, you can use the highlight_objects Tcl command to highlight the nets.

11. Type the following command at the Tcl prompt:

```
highlight_objects -color yellow [get_nets -of [get_pins -of [get_cells \
wbOutputData_reg[*]] -filter DIRECTION==OUT]]
```

This highlights all the nets connected to the output pins of the wbOutputData_reg[*] registers.



In the Device window, you can see that there are various routing distances between the clustered output registers and the distributed outputs pads of the bus. Consistently placing the output registers in the slices next to each output port eliminates a majority of the variability in the clock-to-out delay of the wbOutputData bus.



12. In the main toolbar, click the Unhighlight All 🏂 button and the Unmark All 繸 button.

Step 4: Improving Bus Timing through Placement

To improve the timing of the wbOutputData bus you will place the output registers closer to their respective output pads, then rerun timing to look for any improvement. To place the output registers, you will identify potential placement sites, and then use a sequence of Tcl commands, or a Tcl script, to place the cells and reroute the connections.

RECOMMENDED: Use a series of Tcl commands to place the output registers in the slices next to the *wbOutPutData* bus output pads.



In the Device window click to disable Routing Resources and make sure Autofit Selection
 is still enabled on the toolbar.

This lets you see placed objects more clearly in the Device window, without the added details of the routing.

2. Select the wbOutputData ports placed on the I/O blocks with the following Tcl command:

```
select_objects [get_ports wbOutputData*]
```

The Device window will show the selected ports highlighted in white, and zoom to fit the selection. By examining the device resources around the selected ports, you can identify a range of placement sites for the output registers.



3. Zoom into the Device window around the bottom selected output ports. The following figure shows the results.



Device ? _ J A K											
$\Leftarrow \Rightarrow @ @ X X \bigcirc H B \Box_j =_j'$	0										
Port: wbOutputData[0] Direction: Output Net: wbOutputData[0] BEL: PAD Site: Y21 (fixed) Site: Y21 (fixed)											

The bottom ports are the lowest bits of the output bus, starting with wbOutputData[0].

This port is placed on Package Pin Y21. Over to the right, where the Slice logic contains the device resources needed to place the output registers, the Slice coordinates are XOY36. You will use that location as the starting placement for the 32 output registers, wbOutputData_reg[31:0].

By scrolling or panning in the Device window, you can visually confirm that the highest output data port, wbOutputData[31], is placed on Package Pin K22, and the registers to the right are in Slice X0Y67.

Now that you have identified the placement resources needed for the output registers, you must make sure they are available for placing the cells. You will do this by quickly unplacing the Slices to clear any currently placed logic.

4. Unplace any cells currently assigned to the range of slices needed for the output registers, SLICE_X0Y36 to SLICE_X0Y67, with the following Tcl command:

```
for {set i 0} {$i<32} {incr i} {
    unplace_cell [get_cells -of [get_sites SLICE_XOY[expr 36 + $i]]]
}</pre>
```

This command uses a for loop with an index counter (i) and a Tcl expression (36 + \$i) to get and unplace any cells found in the specified range of Slices. For more information on for loops and other scripting suggestions, refer to the Vivado Design Suite User Guide: Using Tcl Scripting (UG894).



TIP: If there are no cells placed within the specified slices, you will see warning messages that nothing has been unplaced. You can safely ignore these messages.

With the slices cleared of any current logic cells, the needed resources are available for placing the output registers. After placing those, you will also need to replace any logic that was unplaced in the last step.

5. Place the output registers, wbOutputData_reg[31:0], in the specified slice range with the following command:

```
for {set i 0} {$i<32} {incr i} {
    place_cell wbOutputData_reg[$i] SLICE_X0Y[expr 36 + $i]/AFF
}</pre>
```

6. Place any remaining unplaced cells with the following command:

```
place_design
```

Note: The Vivado placer works incrementally on a partially placed design.

7. As a precaution, unroute any nets connected to the output register cells, wbOutputData_reg[31:0], using the following Tcl command:

```
route_design -unroute -nets [get_nets -of [get_cells \
wbOutputData_reg[*]]]
```

8. Route any currently unrouted nets in the design:

route_design

Note: The Vivado router works incrementally on a partially routed design.

9. Analyze the route status of the current design to ensure that there are no routing conflicts:

 ${\tt report_route_status}$

- 10. Click the Routing Resources button **He** to view the detailed routing resources in the Device window.
- 11. Mark the output ports and registers again, and re-highlight the routing between them using the following Tcl commands:

```
mark_objects -color blue [get_ports wbOutputData[*]]
mark_objects -color red [get_cells wbOutputData_reg[*]]
highlight_objects -color yellow [get_nets -of [get_pins -of [get_cells\
wbOutputData_reg[*]] -filter DIRECTION==OUT]]
```



TIP: Because you have entered these commands before, you can copy them from the journal file (*vivado.jou*) to avoid typing them again.

- 12. In the Device window, zoom into some of the marked output ports.
- 13. Select the nets connecting to them.



TIP: You can also select the nets in the Netlist window, and they will be cross-selected in the Device window.

In the Device window, you can see that all output registers are now placed equidistant from their associated outputs, and the routing path is very similar for all the nets from output register to output. This results in clock-to-out times that are closely matched between the outputs.



14. Run the **Reports** \rightarrow **Timing** \rightarrow **Report Datasheet** command again.

The Report Datasheet dialog box is populated with settings from the last time you ran it:

- Reference: [get_ports {wbOutputData[0]}]
- **Ports:** [get_ports {wbOutputData[*]}]

15. In the Report Datasheet results, select the Max/Min Delays for Groups → Clocked by wbClk → wbOutputData[0] section.

Examining the results, the timing skew is closely matched within both the lower bits, wbOutputData[0-13], and the upper bits, wbOutputData[14-31], of the output bus. While the overall skew is reduced, it is still over 200 ps between the upper and lower bits.

With the improved placement, the skew is now a result of the output ports and registers spanning two clock regions, XOYO and XOY1, which introduces clock network skew. Looking at the wbOutputData bus, notice that the Max delay is greater on the lower bits than it is on the upper bits. To reduce the skew, add delay to the upper bits.

You can eliminate some of the skew using a BUFMR/BUFR combination instead of a BUFG, to clock the output registers. However, for this tutorial, you will use manual routing to add delay from the output registers clocked by the BUFG to the output pins for the upper bits, wbOutputData[14-31], to further reduce the clock-to-out variability within the bus.



nformation	Pad	Max Delay	Max Edge	Max Process Corner	Min Delay	Min Edge	Min Process Corner	Edge Skew
orts Clock-to-out	wbOutputData[0]	7.911	Rise	SLOW	3.303	Rise	FAST	0.000
tween Clocks	wbOutputData[31]	7.720	Rise	SLOW	3.236	Rise	FAST	0.190
old for Input Buses	wbOutputData[30]	7.697	Rise	SLOW	3.213	Rise	FAST	0.213
ked by wbClk	wbOutputData[29]	7.705	Rise	SLOW	3.221	Rise	FAST	0.205
wbInputData	wbOutputData[28]	7.707	Rise	SLOW	3.222	Rise	FAST	0.204
Delays for Output I	 wbOutputData[27] 	7.715	Rise	SLOW	3.228	Rise	FAST	0.196
ed by wbClk	wbOutputData[26]	7.688	Rise	SLOW	3.201	Rise	FAST	0.223
bOutputData	wbOutputData[25]	7.694	Rise	SLOW	3.208	Rise	FAST	0.217
Delays for Groups	wbOutputData[24]	7.695	Rise	SLOW	3.208	Rise	FAST	0.215
ked by wbClk	wbOutputData[23]	7.701	Rise	SLOW	3.214	Rise	FAST	0.210
wbOutputData[0]	wbOutputData[22]	7.687	Rise	SLOW	3.200	Rise	FAST	0.224
	 wbOutputData[21] 	7.691	Rise	SLOW	3.203	Rise	FAST	0.220
	wbOutputData[20]	7.723	Rise	SLOW	3.236	Rise	FAST	0.188
	wbOutputData[19]	7.730	Rise	SLOW	3.242	Rise	FAST	0.181
	wbOutputData[18]	7.678	Rise	SLOW	3.191	Rise	FAST	0.233
	 wbOutputData[17] 	7.681	Rise	SLOW	3.193	Rise	FAST	0.230
	wbOutputData[16]	7.718	Rise	SLOW	3.231	Rise	FAST	0.193
	wbOutputData[15]	7.723	Rise	SLOW	3.236	Rise	FAST	0.188
	wbOutputData[14]	7.755	Rise	SLOW	3.268	Rise	FAST	0.156
	wbOutputData[13]	7.942	Rise	SLOW	3.336	Rise	FAST	0.032
	wbOutputData[12]	7.891	Rise	SLOW	3.284	Rise	FAST	0.019
	wbOutputData[11]	7.897	Rise	SLOW	3.290	Rise	FAST	0.014
	wbOutputData[10]	7.905	Rise	SLOW	3.297	Rise	FAST	0.007
	wbOutputData[9]	7.914	Rise	SLOW	3.306	Rise	FAST	0.004
	wbOutputData[8]	7.886	Rise	SLOW	3.278	Rise	FAST	0.025
	 wbOutputData[7] 	7.881	Rise	SLOW	3.274	Rise	FAST	0.030
	wbOutputData[6]	7.915	Rise	SLOW	3.306	Rise	FAST	0.004
	wbOutputData[5]	7.919	Rise	SLOW	3.311	Rise	FAST	0.008
	wbOutputData[4]	7.868	Rise	SLOW	3.261	Rise	FAST	0.043
	wbOutputData[3]	7.874	Rise	SLOW	3.267	Rise	FAST	0.037
	wbOutputData[2]	7.875	Rise	SLOW	3.269	Rise	FAST	0.035
	wbOutputData[1]	7.878	Rise	SLOW	3.271	Rise	FAST	0.033
	wbOutputData[0]	7.911	Rise	SLOW	3.303	Rise	FAST	0.000
	Worst Case Summary	7.942	Rise	SLOW	3.191	Rise	FAST	0.233

Step 5: Using Manual Routing to Reduce Clock Skew

To adjust the skew, begin by examining the current routing of the nets, wbOutputData_OBUF[14:31], to see where changes might be made to consistently add delay. You can use a Tcl for loop to report the existing routing on those nets, to let you examine them more closely.



1. In the Tcl Console, type the following command:

```
for {set i 14} {$i<32} {incr i} {
    puts "$i [get_property ROUTE [get_nets -of [get_pins -of \
       [get_cells wbOutputData_reg[$i]] -filter DIRECTION==OUT]]]"
}</pre>
```

This for loop initializes the index to 14 (set i 14), and gets the ROUTE property to return the details of the route on each selected net.

The Tcl Console returns the net index followed by relative route information for each net:

```
{ CLBLL_LL_AQ CLBLL_LOGIC_OUTS4 WW2BEG0 IMUX_L34 IOI_OLOGIC0_D1
14
LIOI_OLOGICO_OQ LIOI_OO }
15 { CLBLL_LL_AQ CLBLL_LOGIC_OUTS4 WW2BEG0 IMUX_L34 IOI_OLOGIC1_D1
LIOI_OLOGIC1_OQ LIOI_O1 }
16 { CLBLL_LL_AQ CLBLL_LOGIC_OUTS4 WW2BEG0 IMUX_L34 IOI_OLOGIC0_D1
LIOI_OLOGICO_OQ LIOI_OO }
17 { CLBLL_LL_AQ CLBLL_LOGIC_OUTS4 WW2BEG0 IMUX_L34 IOI_OLOGIC1_D1
LIOI_OLOGIC1_OQ LIOI_O1 }
18 { CLBLL_LL_AQ CLBLL_LOGIC_OUTS4 WW2BEG0 IMUX_L34 IOI_OLOGIC0_D1
LIOI_OLOGICO_OQ LIOI_OO }
19 { CLBLL_LL_AQ CLBLL_LOGIC_OUTS4 WW2BEG0 IMUX_L34 IOI_OLOGIC1_D1
LIOI_OLOGIC1_OQ LIOI_O1 }
20 { CLBLL_LL_AQ CLBLL_LOGIC_OUTS4 WW2BEG0 IMUX_L34 IOI_OLOGIC0_D1
LIOI_OLOGICO_OQ LIOI_OO }
21 { CLBLL_LL_AQ CLBLL_LOGIC_OUTS4 WW2BEG0 IMUX_L34 IOI_OLOGIC1_D1
LIOI_OLOGIC1_OQ LIOI_O1 }
   { CLBLL_LL_AQ CLBLL_LOGIC_OUTS4 WW2BEG0 IMUX_L34 IOI_OLOGIC0_D1
22
LIOI_OLOGICO_OQ LIOI_OO }
23 { CLBLL_LL_AQ CLBLL_LOGIC_OUTS4 WW2BEG0 IMUX_L34 IOI_OLOGIC1_D1
LIOI_OLOGIC1_OQ LIOI_O1 }
24 { CLBLL_LL_AQ CLBLL_LOGIC_OUTS4 WW2BEG0 IMUX_L34 IOI_OLOGIC0_D1
LIOI_OLOGICO_OQ LIOI_OO }
25 { CLBLL_LL_AQ CLBLL_LOGIC_OUTS4 WW2BEG0 IMUX_L34 IOI_OLOGIC1_D1
LIOI_OLOGIC1_OQ LIOI_O1 }
26 { CLBLL_LL_AQ CLBLL_LOGIC_OUTS4 WW2BEG0 IMUX_L34 IOI_OLOGIC0_D1
LIOI_OLOGICO_OQ LIOI_OO }
27 { CLBLL_LL_AQ CLBLL_LOGIC_OUTS4 WW2BEG0 IMUX_L34 IOI_OLOGIC1_D1
LIOI_OLOGIC1_OQ LIOI_O1 }
28 { CLBLL_LL_AQ CLBLL_LOGIC_OUTS4 WW2BEG0 IMUX_L34 IOI_OLOGIC0_D1
LIOI_OLOGICO_OQ LIOI_OO }
   { CLBLL_LL_AQ CLBLL_LOGIC_OUTS4 WW2BEG0 IMUX_L34 IOI_OLOGIC1_D1
29
LIOI_OLOGIC1_OQ LIOI_O1 }
30 { CLBLL_LL_AQ CLBLL_LOGIC_OUTS4 WW2BEG0 IMUX_L34 IOI_OLOGIC0_D1
LIOI_OLOGIC0_OQ LIOI_O0 }
   { CLBLL_LL_AQ CLBLL_LOGIC_OUTS4 WW2BEG0 IMUX_L34 IOI_OLOGIC1_D1
LIOI_OLOGIC1_OQ LIOI_O1 }
```

From the returned ROUTE properties, note that the nets are routed from the output registers using identical resources, up to node IMUX_L34. Beyond that, the Vivado router uses different nodes for odd and even index nets to complete the connection to the die pad.

By reusing routing paths, you can manually route one net with an even index, like wbOutputData_OBUF[14], and one net with an odd index, such as
wbOutputData_OBUF[15], and copy the routing to all other even and odd index nets in
the group.



2. In the Tcl Console, select the first net with the following command:

```
select_objects [get_nets -of [get_pins -of \
[get_cells wbOutputData_reg[14]] -filter DIRECTION==OUT]]
```

- 3. In the Device window, right-click to open the popup menu and select Unroute.
- 4. Click Yes in the Confirm Unroute dialog box.

The Device window displays the unrouted net as a fly-line between the register and the output pad.

- 5. Click the Maximize button \Box to maximize the Device window.
- 6. Right-click the net and select **Enter Assign Routing Mode**.

The Target Load Cell Pin dialog box opens, as seen in the following Figure, to let you select a load pin to route to or from. In this case, only one load pin populates: wbOutputData_OBUF[14]_inst.

A	À Assign Routing Mode: Target Load Cell Pin 🛛 🕹 🕹										
Please choose a target load cell pin you want to route to/from. Select 'No Load' if unsure of target load cell pin.											
	Cell Pin	s									
	^1	Dir	BEL	Cell	Net Delay (Rout.					
	$\square +$	Input	IN	wbOutputData_OBUF[14]_inst							
	Load Net Delays OK No Load Exit Mode										

7. Select the load cell pin wbOutputData_OBUF[14]_inst/I, and click OK.

The Vivado IDE enters into Assign Routing mode, displaying a new Routing Assignment window on the right side of the Device window, as shown in the following figure.



Device										? _]	ак
← ⇒ @,	Q 🔀	N 0 N	Ro 🛛 🗖 🗖	ç.							۰
1 a a .	6								Routing A	ssignment	
- 1							Net: [~ Op	_ wbOu tions	tputData_OBUF[14	1	
j i o:							<u>N</u>	umber of I aximum n	hops: umber of neighbor	1 ↓ [1 - 20] s: 1,000 ↓ [10 - 50	0000]
1							Neighl	Allow ov	erlap with unfixed r	nets	
j							Q	Name	Base rile		Î
	XOY1						<				
- <mark>ju n</mark> i							> Assin	ned Node	9		v
							Q	Name	L_LL_AQ	Base Tile CLBLL_L_X2Y50	
								CLBL	L_LOGIC_OUTS4	CLBLL_L_X2Y50 ap: 1	
								5 IOL 0	LJ4	LIOI3 SING X0Y50	
								트 LIOI_ 트 LIOI_	0L0GIC0_0Q	LIOI3_SING_X0Y50 LIOI3_SING_X0Y50	
								[Assign <u>R</u> outing	Exit Mode	routing

The Routing Assignment window includes the following sections:

- Net: Displays the current net being routed.
- Options: Are hidden by default, and can be displayed by clicking Options.
 - **Number of hops:** Defines how many programmable interconnect points, or PIPs, to look at when reporting the available neighbors. The default is 1.
 - Maximum number of neighbors: Limits the number of neighbors displayed for selection.
 - Allow overlap with unfixed nets: Enables or disables a loose style of routing which can create conflicts that must be later resolved. The default is ON.
- **Neighbor Nodes:** Lists the available neighbor PIPs/nodes to choose from when defining the path of the route.
- Assigned Nodes: Shows the currently assigned nodes in the route path of the selected net.
- Assign Routing: Assigns the currently defined path in the Routing Assignment window as the route path for the selected net.
- Exit Mode: Closes the Routing Assignment window.

The Assigned Nodes section displays six currently assigned nodes. The Vivado router automatically assigns a node if it is the only neighbor of a selected node and there are no alternatives to the assigned nodes for the route. In the Device window, the assigned nodes appear as a partial route in orange.



In the currently selected net, wbOutputData_OBUF[14], nodes CLBLL_LL_AQ and CLBLL_LOGIC_OUTS4 are already assigned because they are the only neighbor nodes available to the output register, wbOutputData_reg[14]. The nodes IMUX_L34, IOI_OLOGICO_D1, LIOI_OLOGICO_OQ, and LIOI_OO are also already assigned because they are the only neighbor nodes available to the destination, the output buffer (OBUF).

A gap exists between the two routed portions of the path where there are multiple neighbors to choose from when defining a route. This gap is where you will use manual routing to complete the path and add the needed delay to balance the clock skew.

You can route the gap by selecting a node on either side of the gap and then choosing the neighbor node to assign the route to. Selecting the node displays possible neighbor nodes in the Neighbor Nodes section of the Routing Assignment window and appear as dashed white lines in the Device window.

TIP: The number of reachable neighbor nodes displayed depends on the number of hops defined in the Options.

8. Under the Assigned Nodes section, select the CLBLL_LOGIC_OUTS4 node before the gap.

The available neighbors appear as shown in the following figure.

To add delay to compensate for the clock skew, select a neighbor node that provides a slight detour over the more direct route previously chosen by the router.



9. Under Neighbor Nodes, select node NE2BEG0.



This node provides a routing detour to add delay, as compared to some other nodes such as WW2BEG0, which provide a more direct route toward the output buffer. Clicking a neighbor node once selects it so you can explore routing alternatives. Double-clicking the node temporarily assigns it to the net, so that you can then select the next neighbor from that node.

10. In Neighbor Nodes, assign node NE2BEG0 by double-clicking it.

This adds the node to the Assigned Nodes section of the Routing Assignment window, which updates the Neighbor Nodes.

11. In Neighbor Nodes, select and assign nodes WR1BEG1, and then WR1BEG2.

TIP: In case you assigned the wrong node, you can select the node from the Assigned Nodes list, rightclick, and select Remove on the context menu.

You can turn off the Auto Fit Selection \bigcirc in the Device window if you would like to stay at the same zoom level.

The following figure shows the partially routed path using the selected nodes shown in orange. You can use the automatic routing feature to fill the remaining gap.



12. Under the Assigned Nodes section of the Routing Assignment window, right-click the **Net Gap**, and select **Auto-Route**, as shown in the following figure.



Assig	ned Nodes					
Q	Name	Base Tile				
	™ CLBLL_LL_AQ	CLBLL_L_X2Y50	^			
	E CLBLL_LOGIC_OUTS	CLBLL_L_X2Y50	- 11			
	™ NE2BEG0	INT_L_X2Y50	- 11			
	🖫 WR1BEG1	INT_R_X3Y51	- 11			
	S WR1BEG2	INT_L_X2Y51	- 11			
	(¹¹¹ × 1 - ×	d	_			
	🔁 IMUX_L34	uto-Route				
	5 IOI_OLOGICO_	lode Properties	Ctrl+E			
	LIOI_OLOGICC	xport to Spreadsheet				
	5≟ LIOI_O0	LIUI3_SING_X0Y50	V			
	E <u>x</u> it Mode					

The Vivado router fills in the last small bit of the gap. With the route path fully defined, you can assign the routing to commit the changes to the design.

13. Click Assign Routing at the bottom of the Routing Assignment window.

The Assign Routing dialog box opens, as seen in the following figure. This displays the list of currently assigned nodes that define the route path. You can select any of the listed nodes, highlighting it in the Device window. This lets you quickly review the route path prior to committing it to the design.





Ą	Assign Routing	×	
F a	Routing for net 'wbOutputData_ ssigned	OBUF[14]' will be	
	Name	Base Tile	
	LUL_AQ	CLBLL_L_X2Y50	
	LOGIC_OUTS4	CLBLL_L_X2Y50	
	NE2BEG0	INT_L_X2Y50	
	WR1BEG1	INT_R_X3Y51	
	WR1BEG2	INT_L_X2Y51	
	SW2BEG1	INT_R_X1Y51	
	LIMUX_L34	INT_L_X0Y50	
	LIOI_OLOGIC0_D1	LIOI3_SING_X0Y50	
	LIOI_OLOGIC0_OQ	LIOI3_SING_X0Y50	
	LIOI_00	LIOI3_SING_X0Y50	
	✓ <u>Fix Routing</u>		
	0	K Cancel	

14. Make sure Fix Routing is checked, and click OK.

The Fix Routing checkbox marks the defined route as fixed to prevent the Vivado router from ripping it up or modifying it during subsequent routing steps. This is important in this case, because you are routing the net manually to add delay to match clock skew.

15. Examine the Tcl commands in the Tcl Console.

The Tcl Console reports any Tcl commands that assigned the routing for the current net. Those commands are:

```
set_property is_bel_fixed 1 [get_cells {wbOutputData_reg[14]
wbOutputData_OBUF[14]_inst }]
set_property is_loc_fixed 1 [get_cells {wbOutputData_reg[14]
wbOutputData_OBUF[14]_inst }]
set_property fixed_route { { CLBLL_LL_AQ CLBLL_LOGIC_OUTS4 NE2BEG0
WR1BEG1 WR1BEG2 SW2BEG1 IMUX_L34 IOI_OLOGIC0_D1 LIOI_OLOGIC0_OQ
LIOI_O0 } } [get_nets {wbOutputData_OBUF[14]}]
```

IMPORTANT! The FIXED_ROUTE property assigned to the net, *wbOutputData_OBUF[14]*, uses a directed routing string with a relative format, based on the placement of the net driver. This lets you reuse defined routing by copying the FIXED_ROUTE property onto other nets that use the same relative route.



After defining the manual route for the even index nets, the next step is to define the route path for the odd index net, wbOutputData_OBUF[15], applying the same steps you just completed.

16. In the Tcl Console type the following to select the net:

select_objects [get_nets wbOutputData_OBUF[15]]

- 17. With the net selected:
 - a. Unroute the net.
 - b. Enter Routing Assignment mode.
 - c. Select the load cell pin.
 - d. Route the net using the specified neighbor nodes (NE2BEG0, WR1BEG1, and WR1BEG2).
 - e. Auto-Route the gap.
 - f. Assign the routing.

The Assign Routing dialog box, shown in the following figure, shows the nodes selected to complete the route path for the odd index nets.

٨	Assign Routing		×
F	Routing for net 'wbOutputData_	_OBUF[15]' will be as	signed
	Name	Base Tile	
	그 CLBLL_LL_AQ 코 CLBLL_LOGIC_OUTS4	CLBLL_L_X2Y51 CLBLL_L_X2Y51	
	칰 NE2BEG0 칰 WR1BEG1	INT_L_X2Y51 INT_R_X3Y52	
	립 WR1BEG2 립 SW2BEG1	INT_L_X2Y52 INT_R_X1Y52	
	툴 IMUX_L34 토 IOL OLOGIC1 D1	INT_L_X0Y51 LIOI3 X0Y51	
	립니어_OLOGIC1_OQ	LIOI3_X0Y51	
	✓ Fix Routing	1013_70131	
	mr. vouing	ОК	Cancel

You routed the wbOutputData_OBUF[14] and wbOutputData_OBUF[15] nets with the detour to add the needed delay. You can now run the Report Datasheet command again to examine the timing for these nets with respect to the lower order bits of the bus.



- 18. Switch to the Timing Datasheet report window. Notice the information message in the banner of the window indicating that the report is out of date because the design was modified.
- 19. In the Timing Datasheet report, click Rerun to update the report with the latest timing information.
- 20. Select Max/Min Delays for Groups \rightarrow Clocked by wbClk \rightarrow wbOutputData[0] to display the timing info for the wbOutputData bus, as seen in the following figure.

Timing								? _	ПЫX
Q 🚡 🖨 C 🎽 Q 🚺 Max/Min Delays for Groups - Clocked by wbClk - wbOutputData[0]									
General Information	Pad 1	Max Delay	Max Edge	Max Process Corner	Min Delay	Min Edge	Min Process Corner	Edge Skew	
Output Ports Clock-to-out	 wbOutputData[31] 	7.720	Rise	SLOW	3.236	Rise	FAST	0.190	
Setup between Clocks	 wbOutputData[30] 	7.697	Rise	SLOW	3.213	Rise	FAST	0.213	
✓ Setup/Hold for Input Buse	 wbOutputData[29] 	7.705	Rise	SLOW	3.221	Rise	FAST	0.205	
Clocked by wbClk	 wbOutputData[28] 	7.707	Rise	SLOW	3.222	Rise	FAST	0.204	
wbInputData	 wbOutputData[27] 	7.715	Rise	SLOW	3.228	Rise	FAST	0.196	
∼ Max/Min Delays for Outpເ	wbOutputData[26]	7.688	Rise	SLOW	3.201	Rise	FAST	0.223	
Clocked by wbClk	wbOutputData[25]	7.694	Rise	SLOW	3.208	Rise	FAST	0.217	
wbOutputData	wbOutputData[24]	7.695	Rise	SLOW	3.208	Rise	FAST	0.215	
✓ Max/Min Delays for Group	 wbOutputData[23] 	7.701	Rise	SLOW	3.214	Rise	FAST	0.210	
Clocked by wbClk	wbOutputData[22]	7.687	Rise	SLOW	3.200	Rise	FAST	0.224	
wbOutputData[0]	wbOutputData[21]	7.691	Rise	SLOW	3.203	Rise	FAST	0.220	
	 wbOutputData[20] 	7.723	Rise	SLOW	3.236	Rise	FAST	0.188	
	wbOutputData[19]	7.730	Rise	SLOW	3.242	Rise	FAST	0.181	
	wbOutputData[18]	7.678	Rise	SLOW	3.191	Rise	FAST	0.233	
	wbOutputData[17]	7.681	Rise	SLOW	3.193	Rise	FAST	0.230	
	wbOutputData[16]	7.718	Rise	SLOW	3.231	Rise	FAST	0.193	
	wbOutputData[15]	7.987	Rise	SLOW	3.366	Rise	FAST	0.076	
	wbOutputData[14]	7.949	Rise	SLOW	3.328	Rise	FAST	0.038	
	 wbOutputData[13] 	7.942	Rise	SLOW	3.336	Rise	FAST	0.032	
	 wbOutputData[12] 	7.891	Rise	SLOW	3.284	Rise	FAST	0.019	
	wbOutputData[11]	7.897	Rise	SLOW	3.290	Rise	FAST	0.014	
	wbOutputData[10]	7.905	Rise	SLOW	3.297	Rise	FAST	0.007	
	wbOutputData[9]	7.914	Rise	SLOW	3.306	Rise	FAST	0.004	
	wbOutputData[8]	7.886	Rise	SLOW	3.278	Rise	FAST	0.025	
	 wbOutputData[7] 	7.881	Rise	SLOW	3.274	Rise	FAST	0.030	
	🔶 wbOutputData[6]	7.915	Rise	SLOW	3.306	Rise	FAST	0.004	
	🔶 wbOutputData[5]	7.919	Rise	SLOW	3.311	Rise	FAST	0.008	
	🔶 wbOutputData[4]	7.868	Rise	SLOW	3.261	Rise	FAST	0.043	
	 wbOutputData[3] 	7.874	Rise	SLOW	3.267	Rise	FAST	0.037	
	 wbOutputData[2] 	7.875	Rise	SLOW	3.269	Rise	FAST	0.035	
	wbOutputData[1]	7.878	Rise	SLOW	3.271	Rise	FAST	0.033	
	wbOutputData[0]	7.911	Rise	SLOW	3.303	Rise	FAST	0.000	
	🔶 wbOutputData[0]	7.911	Rise	SLOW	3.303	Rise	FAST	0.000	
	Worst Case Summary	7.987	Rise	SLOW	3.191	Rise	FAST	0.233	
	Bus Skew: 0.233 ns								
Datasheet - timing 2	L								
Datasheet - tilling_2									



You can see from the report that the skew within the rerouted nets, wbOutputData[14] and wbOutputData[15], more closely matches the timing of the lower bits of the output bus, wbOutputData[13:0]. The skew is within the target of 100 ps of the reference pin wbOutputData[0].

In Step 6, you copy the same route path to the remaining nets, wbOutputData_OBUF[31:16], to tighten the timing of the whole wbOutputData bus.

Step 6: Copying Routing to Other Nets

To apply the same fixed route used for net wbOutputData_OBUF[14] to the even index nets, and the fixed route for wbOutputData_OBUF[15] to the odd index nets, you can use Tcl For loops as described in the following steps.

- 1. Select the **Tcl Console** tab.
- 2. Set a Tcl variable to store the route path for the even nets and the odd nets:

```
set even [get_property FIXED_ROUTE [get_nets wbOutputData_OBUF[14]]]
set odd [get_property FIXED_ROUTE [get_nets wbOutputData_OBUF[15]]]
```

3. Set a Tcl variable to store the list of nets to be routed, containing all high bit nets of the output data bus, wbOutputData_OBUF[16:31]:

```
for {set i 16} {$i<32} {incr i} {
    lappend routeNets [get_nets wbOutputData_OBUF[$i]]
}</pre>
```

4. Unroute the specified nets:

```
route_design -unroute -nets $routeNets
```

5. Apply the FIXED_ROUTE property of net wbOutputData_OBUF[14] to the even nets:

```
for {set i 16} {$i<32} {incr i 2} {
    set_property FIXED_ROUTE $even [get_nets wbOutputData_OBUF[$i]]
}</pre>
```

6. Apply the FIXED_ROUTE property of net wbOutputData_OBUF[15] to the odd nets:

```
for {set i 17} {$i<32} {incr i 2} {
    set_property FIXED_ROUTE $odd [get_nets wbOutputData_OBUF[$i]]
}</pre>
```

The even and odd nets of the output data bus, as needed, now have the same routing paths, adding delay to the high order bits. Run the route status report and the datasheet report to validate that the design is as expected.

7. In the Tcl Console, type the following command:

report_route_status



- **TIP:** Some routing errors might be reported if the routed design included nets that use some of the nodes you have assigned to the FIXED_ROUTE properties of the manually routed nets. Remember you enabled Allow Overlap with Unfixed Nets in the Routing Assignment window.
- 8. If any routing errors are reported, type the <code>route_design</code> command in the Tcl Console.

The nets with the FIXED_ROUTE property takes precedence over the auto-routed nets.

- 9. After route_design, repeat the report_route_status command to see the clean report.
- 10. Examine the output data bus in the Device window, as seen in the following figure:
 - All nets from the output registers to the output pins for the upper bits 14-31 of the output bus wbOutputData have identical fixed routing sections (shown as dashed lines).
 - You do not need to fix the LOC and the BEL for the output registers. It was done by the place_cell command in an earlier step.



Having routed all the upper bit nets, wbOutputData_OBUF[31:14], with the detour needed for added delay, you can now re-examine the timing of output bus.

11. Select the **Timing** window.

Notice the information message in the banner of the window indicating that the report is out of date because timing data has been modified.

12. Click **rerun** to update the report with the latest timing information.



13. Select the Max/Min Delays for Groups → Clocked by wbClk → wbOutputData[0] section to display the timing info for the wbOutputData bus.

The clock-to-out timing within all bits of output bus wbOutputData is now closely matched to within 83 ps.

- 14. Save the constraints to write them to the target XDC, so that they apply every time you compile the design.
- 15. Select File → Constraints → Save to save the placement constraints to the target constraint file, bft_full.xdc, in the active constraint set, constrs_1.

The synthesis and implementation will go out-of-date since constraints were updated. You can force the design to update by clicking on **Details** in tool bar, since new constraints are already applied.



Timing									? _ D @ X
Q 🗶 🚔 C 👋 Q 🚺 Max/Min Delays for Groups - Clocked by wbClk - wbOutputData[0]									
General Information	Pad	Max Delay	Max Edge	Max Process Corner	Min Delay	Min Edge	Min Process Corner	Edge Skew	
Output Ports Clock-to-out	wbOutputData[0]	7.911	Rise	SLOW	3.303	Rise	FAST	0.000	
Setup between Clocks	wbOutputData[31]	7.984	Rise	SLOW	3.366	Rise	FAST	0.073	
 Setup/Hold for Input Buse 	wbOutputData[30]	7.961	Rise	SLOW	3.343	Rise	FAST	0.050	
 Clocked by wbClk 	wbOutputData[29]	7.969	Rise	SLOW	3.351	Rise	FAST	0.058	
wblnputData	wbOutputData[28]	7.970	Rise	SLOW	3.351	Rise	FAST	0.059	
✓ Max/Min Delays for Outpu	wbOutputData[27]	7.978	Rise	SLOW	3.357	Rise	FAST	0.067	
Clocked by wbClk	wbOutputData[26]	7.951	Rise	SLOW	3.331	Rise	FAST	0.041	
wbOutputData	wbOutputData[25]	7.958	Rise	SLOW	3.338	Rise	FAST	0.047	
✓ Max/Min Delays for Group	wbOutputData[24]	7.959	Rise	SLOW	3.338	Rise	FAST	0.048	
Clocked by wbClk	wbOutputData[23]	7.965	Rise	SLOW	3.344	Rise	FAST	0.054	
wbOutputData[0]	wbOutputData[22]	7.951	Rise	SLOW	3.330	Rise	FAST	0.040	
	wbOutputData[21]	7.955	Rise	SLOW	3.333	Rise	FAST	0.044	
	wbOutputData[20]	7.986	Rise	SLOW	3.365	Rise	FAST	0.076	
	wbOutputData[19]	7.994	Rise	SLOW	3.371	Rise	FAST	0.083	
	wbOutputData[18]	7.942	Rise	SLOW	3.320	Rise	FAST	0.031	
	wbOutputData[17]	7.945	Rise	SLOW	3.323	Rise	FAST	0.034	
	wbOutputData[16]	7.981	Rise	SLOW	3.360	Rise	FAST	0.070	
	wbOutputData[15]	7.987	Rise	SLOW	3.366	Rise	FAST	0.076	
	wbOutputData[14]	7.949	Rise	SLOW	3.328	Rise	FAST	0.038	
	wbOutputData[13]	7.942	Rise	SLOW	3.336	Rise	FAST	0.032	
	wbOutputData[12]	7.891	Rise	SLOW	3.284	Rise	FAST	0.019	
	wbOutputData[11]	7.897	Rise	SLOW	3.290	Rise	FAST	0.014	
	wbOutputData[10]	7.905	Rise	SLOW	3.297	Rise	FAST	0.007	
	wbOutputData[9]	7.914	Rise	SLOW	3.306	Rise	FAST	0.004	
	wbOutputData[8]	7.886	Rise	SLOW	3.278	Rise	FAST	0.025	
	wbOutputData[7]	7.881	Rise	SLOW	3.274	Rise	FAST	0.030	
	wbOutputData[6]	7.915	Rise	SLOW	3.306	Rise	FAST	0.004	
	wbOutputData[5]	7.919	Rise	SLOW	3.311	Rise	FAST	0.008	
	wbOutputData[4]	7.868	Rise	SLOW	3.261	Rise	FAST	0.043	
	wbOutputData[3]	7.874	Rise	SLOW	3.267	Rise	FAST	0.037	
	wbOutputData[2]	7.875	Rise	SLOW	3.269	Rise	FAST	0.035	
	wbOutputData[1]	7.878	Rise	SLOW	3.271	Rise	FAST	0.033	
	wbOutputData[0]	7.911	Rise	SLOW	3.303	Rise	FAST	0.000	
	Worst Case Summary	7.994	Rise	SLOW	3.261	Rise	FAST	0.083	
<>	Bus Skew: 0.083 ns								
Datasheet - timing_2									

Conclusion

In this lab, you did the following:

• Analyzed the clock skew on the output data bus using the Report Datasheet command.



- Used manual placement techniques to improve the timing of selected nets.
- Used the Assign Manual Routing Mode in the Vivado IDE to precisely control the routing of a net.
- Used the FIXED_ROUTE property to copy the relative fixed routing among similar nets to control the routing of the critical portion of the nets.





Chapter 4

Lab 4: Vivado ECO Flow

In this lab, you will learn how to use the Vivado[®] Engineering Change Order (ECO) flow to modify your design post implementation, implement the changes, run reports on the changed netlist, and generate programming files.

For this lab, you will use the design file that is included with this guide and is targeted at the Kintex[®] UltraScale[™] KCU105 Evaluation Platform. For instructions on locating the design files, see Locating Design Files for Lab 4.

A block diagram of the design is shown in the following figure.



Figure 3: Block Diagram of the Design

In this design, a mixed-mode clock manager (MMCM) is used to synthesize a 100 MHz clock from the 300 MHz clock provided by the board.

A 29-bit counter is used to divide the clock down further. The four most significant bits of the counter form the count < 3: 0 > signal that is 0-extended to 8 bits and drives the 8 on-board LEDs through an 8-bit 2-1 mux.

The count<3:0> signal is also squared using a multiplier, and the product drives the other eight inputs of the mux. A Toggle signal controls the mux select and either drives the LEDs (shown in the following figure) with the counter value or the multiplier output.



A Pause signal allows you to stop the counter, and a Reset signal allows you to reset the design. The Toggle, Pause, and Reset signals can either be controlled from on-board buttons shown in the following figure or a VIO in the Hardware Manager as shown in the subsequent figure. The VIO also allows you to observe the status of the LEDs. The following figures show the location of the push-buttons and the LEDs on the KCU105 board and a Hardware Manager dashboard. These allow you to control the push button and observe the LEDs through the VIO.



Figure 4: KCU105 On-Board Push Buttons and LEDs

hw_	_vios				? _ 🗆 🖓
	hw_vio_1				? _ □ ×
ions	Q 素 ≑ + −				
Opt	Name	Value	Acti	Directi	VIO
oard	✓ ≒ count_out_OBUF[7:0]	[U] 100		Input	hw_vio_1
shb	」 count_out_OBUF	•		Input	hw_vio_1
Ö	」 count_out_OBUF	٢		Input	hw_vio_1
	」 count_out_OBUF	۲		Input	hw_vio_1
\square	_ count_out_OBUF	•		Input	hw_vio_1
	_ count_out_OBUF	•		Input	hw_vio_1
	」 count_out_OBUF	٢		Input	hw_vio_1
	」 count_out_OBUF	•		Input	hw_vio_1
	_ count_out_OBUF	•		Input	hw_vio_1
	∿= pause_vio_out	0		Output	hw_vio_1
	∿a reset_vio_out	0		Output	hw_vio_1
	∿a toggle_vio_out	0		Output	hw_vio_1
	∿= vio_select	1		Output	hw_vio_1



Related Information

Locating Design Files for Lab 4

Step 1: Creating a Project Using the Vivado New Project Wizard

To create a project, use the New Project wizard to name the project, to add RTL source files and constraints, and to specify the target device.

- 1. Open the Vivado Design Suite integrated design environment (IDE).
- 2. In the Getting Started page, click **Create Project** to open the New Project wizard.
- 3. Click Next.
- 4. In the Project Name page, do the following:
 - a. Name the new project project_ECO_lab.
 - b. Provide the project location C:/Vivado_Tutorial.
 - c. Ensure that Create project subdirectory is selected.
 - d. Click Next.
- 5. In the Project Type page, do the following:
 - a. Specify the type of project to create as **RTL Project**.
 - b. Leave the Do not specify sources at this time check box unchecked.
 - c. Click Next.
- 6. In the Add Sources page, do the following:
 - a. Set the Target Language to Verilog.
 - b. Click Add Files.
 - c. In the Add Source Files dialog box, navigate to the /src/lab4 directory.
 - d. Select all Verilog source files.
 - e. Click OK.
 - f. Verify that the files are added.
 - g. Click Add Files.
 - h. In the Add Source Files dialog box, navigate to the /src/lab4/IP directory.
 - i. Select all of the XCI source files and click **OK**.
 - j. Verify that the files are added and **Copy sources into project** is selected.

Send Feedback



- k. Click Next.
- 7. In the Add Constraints dialog box, do the following:
 - a. Click the Add button +, and then select Add Files.
 - b. Navigate to the /src/lab4 directory and select ECO_kcu105.xdc.
 - c. Click Next.
- 8. In the Default Part page, do the following:
 - a. Select Boards and then select Kintex-UltraScale KCU105 Evaluation Platform.
 - b. Click Next.
- 9. Review the New Project Summary page. Verify that the data appears as expected, per the steps above.
- 10. Click Finish.

Note: It might take a moment for the project to initialize.

11. In the Sources window in the Vivado IDE, expand top to see the source files for this lab.



Step 2: Synthesizing, Implementing, and Generating the Bitstream

1. In the Flow Navigator, under Program and Debug, click Generate Bitstream.

This synthesizes, implements, and generates a bitstream for the design.

The No Implementation Results Available dialog box appears.

2. Click Yes.



After bitstream generation completes, the Bitstream Generation Completed dialog box appears. Open Implemented Design is selected by default.

- 3. Click OK.
- 4. Inspect the Timing Summary report and make sure that all timing constraints have been met.

				? _ D @ X
Design Timing Summary				
Setup	Hold		Pulse Width	
Worst Negative Slack (WNS): 6.410 ns	Worst Hold Slack (WHS):	0.004 ns	Worst Pulse Width Slack (WPWS):	0.666 ns
Total Negative Slack (TNS): 0.000 ns	Total Hold Slack (THS):	0.000 ns	Total Pulse Width Negative Slack (TPWS):	0.000 ns
Number of Failing Endpoints: 0	Number of Failing Endpoints:	0	Number of Failing Endpoints:	0
Total Number of Endpoints: 4842	Total Number of Endpoints:	4842	Total Number of Endpoints:	2749
All user specified timing constraints are me	L.			
	Design Timing Summary Setup Worst Negative Slack (WNS): 6.410 ns Total Negative Slack (TNS): 0.000 ns Number of Failing Endpoints: 0 Total Number of Endpoints: 4842 All user specified timing constraints are me	Design Timing Summary Setup Hold Worst Negative Slack (WNS): 6.410 ns Total Negative Slack (TNS): 0.000 ns Total Negative Slack (TNS): 0.000 ns Number of Failing Endpoints: 0 Total Number of Failing Endpoints: 4842 All user specified timing constraints are met. Total Number of Endpoints:	Design Timing Summary Hold Setup Hold Worst Negative Slack (WNS): 6.410 ns Total Negative Slack (TNS): 0.000 ns Number of Failing Endpoints: 0 Total Number of Failing Endpoints: 0 Total Number of Endpoints: 4842 All user specified timing constraints are met. Vorst Number of Endpoints:	Design Timing Summary Hold Pulse Width Setup Hold Worst Negative Slack (WNS): 6.410 ns Worst Hold Slack (WHS): 0.004 ns Worst Pulse Width Slack (WPWS): Total Negative Slack (TNS): 0.000 ns Total Hold Slack (THS): 0.000 ns Total Hold Slack (THS): 0.000 ns Total Pulse Width Negative Slack (TPWS): Number of Failing Endpoints: 0 Number of Failing Endpoints: 4842 Total Number of Endpoints: 4842 Total Number of Endpoints: 4842 All user specified timing constraints are met. Hold Slack (THS): 0.000 ns Total Number of Endpoints: 4842

You can use the generated bitstream programming file to download your design into the target FPGA device using the Hardware Manager. For more information, see the *Vivado Design Suite User Guide: Programming and Debugging* (UG908).

Step 3: Validating the Design on the Board

This step is optional, but will help you understand the ECO modifications that you will make in Step 4: Making the ECO Modifications.

1. From the main menu, select $Flow \rightarrow Open Hardware Manager$.

The Hardware Manager window opens.





project_ECO_lab - [C:/Data/Vivado_T	utorial/project_ECO_lab/project_ECO_lab.xpr]	
<u>File Edit Flow Tools W</u> indow	Layout View Help Q- Quick Access write_1	oitstream Complete 🛛 🗸
	🖩 🏚 ∑ 🗶 🖉 💥 Dashboard	Default Layout 🛛 🗸 🗸
Flow Navigator 😤 ≑ ? 💶	HARDWARE MANAGER - unconnected	? ×
✓ PROJECT MANAGER	No hardware target is open. Open target	
🍄 Settings	Hardware 2 D D X	
Add Sources		
Language Templates		
-₽ IP Catalog		
Create Block Design	No content	
Open Block Design		
Generate Block Design		
Contracts Brook Boorgin		
✓ SIMULATION		
Run Simulation	Properties ? _ 🗆 🖾 ×	
Y RTI ANALYSIS	$\leftarrow \rightarrow \diamond$	
Open Elaborated Design		
✓ SYNTHESIS		
Run Synthesis	Select an object to see properties	
> Open Synthesized Design		
Run Implementation		
> Open Implemented Design	Tel Consolo y Maceanae Sarial I/A Linke Sarial I/A Scane	2 0 7
		: _ 0 0
Y PROGRAM AND DEBUG	Second and secon	
👫 Generate Bitstream	INFO: [IP_Flow 19-234] Refreshing IP repositories	
✓ Open Hardware Manager	INFU: [1F_FIOW 19-1704] No user IP repositories specified [INFO: [IF_FIOW 19-2313] Loaded Vivado IP repository 'C:/Xilinx/Vivado/2017.1/data/ip'.	
Open Target		
Program Device	open hw	, v
Add Configuration Memory Device	Type a Tcl command here	

2. Connect to a hardware target using hw_server.

TIP: For more information about different ways to connect to a hardware target, refer to the Vivado Design Suite User Guide: Programming and Debugging (UG908).

 \bigcirc





HARDWARE MANAGER - localhost/xilinx_tcf/Digilent/21025	45167 ? ×
Hardware ? _ D 🗹	hw_ila_1 × hw_vios × ? □ Ľ
Q X ♦ Ø Ø Name Status	Dashboard Options
	Image: Setup 5 Se
Hardware Dovice Properties ? □<	Settings - hw_lia_1 Status - hw_lia_1 × ?
Tcl Console × Messages Serial HO Links St Q X + HI E Image: Console of the series of the s	allO Scams ? _ C C

3. In the Vivado Flow Navigator, under Program and Debug, click Program Device.

The Program Device dialog box opens.

🍌 Program Device		×					
Select a bitstream programming file and download it to your hardware device. You can optionally select a debug probes file that corresponds to the debug cores contained in the bitstream programming file.							
Bitstre <u>a</u> m file: Debug probes file: ✓ <u>E</u> nable end of st	Bitstream file: 'ivado_Tutorial/project_ECO_lab/project_ECO_lab.runs/impl_1/top.bit Debug probes file:						
•	<u>P</u> rogram Car	icel					

- 4. Navigate to the Bitstream file and Debug Probes file.
- 5. Click Program.

Now that the FPGA is configured, you can use the on-board buttons and the on-board LEDs to control and observe the hardware. Press the **Pause** button to pause the counter. Press the **Toggle** button to select between the count and the multiplier result. Press the Reset button to reset the counter.





Alternatively, you can use the VIO to control and observe the hardware.

If the following warning message appears, select one of the alternatives suggested in the message.

WARNING: [Labtools 27-1952] VIO hw_probe OUTPUT_VALUE properties for hw_vio(s) [hw_vio_1] differ from output values in the VIO core(s). Resolution: To synchronize the hw_probes properties and the VIO core outputs choose one of the following alternatives: 1) Execute the command 'Commit Output Values to VIO Core', to write down the hw_probe values to the core. 2) Execute the command 'Refresh Input and Output Values from VIO Core', to update the hw_probe properties with the core values. 3) First restore initial values in the core with the command 'Reset VIO Core Outputs', and then execute the command 'Refresh Input and Output Values from VIO Core'.

6. Select the **hw_vios** tab in the dashboard and click the **Add** button + to add probes.

The Add Probes dialog box opens.




Add Probes	×
★ ♦	•
Search: Q,*	
Probes for hw_vio_1 (5)	~
 W_vio_1 count_out_OBUF[7:0] pause_vio_out reset_vio_out toggle_vio_out vio_select 	
ОК С	ancel

- 7. Select all of the probes for hw_vio_1 and click OK.
- 8. In the hw_vios dashboard, select count_out_OBUF[7:0], then right-click and select Radix
 → Unsigned Decimal.



9. In the hw_vios dashboard, select count_out_OBUF[7:0], then right-click and select LED. The Select LED Colors dialog box opens.



🔥 Select LED Cold	ors 🛛 🔀
Low Value Color:	🥥 Red 🗸 🗸
<u>H</u> igh Value Color:	Green 🗸
ОК	Cancel

- 10. Select **Red** for the Low Value Color and **Green** for the High Value Color.
- 11. Click **OK**.
- 12. In the hw_vios dashboard, select **pause_vio_out**, **reset_vio_out**, and **toggle_vio_out**, then right-click and select **Active-High Button**.

	hw vio 1						
2	Q ≚ ≑ + -						
2	Name	Value		Acti	Directi	VIO	
5	> 🍓 count_out_OBUF[7:0]	[U] 1			Input	hw_vio_1	
	🛥 pause_vio_out	(B) 0	Ŧ		Output	hw_vio_1	
	∿a reset_vio_out	(B) 0	Ŧ		OI	Debug Probe Properties	Ctrl+E
	🍋 toggle_vio_out	(B) 0	Ŧ		0	Text	
J	∿■ vio_select	[B] 0	*		01	Active-High Button	
						Active-Low Button	
						Toggle Button	
						Radix	
						Rename	
						Name	+
						Remove	Delete
						Export to Spreadsheet	

13. In the hw_vios dashboard, right-click vio_select and select Toggle Button.



hw.	_ila_1 × hw_vi	os	×						
	hw_vio_1								
ions	Q ¥ ♦	+	-						
Opti	Name			Value	Acti	Directi	VIO		
oard	> 🍓 count_out_	OBUF[7:0]	[U] 121		Input	hw_vio_1		
ashb	🛥 pause_vio_	out		0		Output	hw_vio_1		
ő	∿= reset_vio_o	ut		0		Output	hw_vio_1		
	∿= toggle_vio_	out		0		Output	hw_vio_1		
	a vio_select		Det	bug Probe Pro	operties	Ctr	I+E		
			Acti	ve-High Butto ve-Low Butto	n n				
			Тод	g Button					
			Rad	dix			+		
			Rer Nar	name me			•		
			Rer	move		De	lete		
			Ехр	ort to Spread	sheet				

14. Expand count_out_OBUF[7:0] to view the VIO LEDs.

Now that the VIO is set up, you are ready to analyze the design.

- 15. Toggle the **vio_select** button to control the hardware from the VIO.
- 16. Press the **pause_vio_out** button to pause the counter.
- 17. Press the **toggle_vio_out** button to select between the count and the multiplier result.
- 18. Press the **reset_vio_out** button to reset the counter.



hw_vio_1 ? _ 🗆							
Q = = + =							
Name	Value	Acti	Directi	VIO			
v te count_out_OBUF[7:0]	[U] 100		Input	hw_vio_1			
	٠		Input	hw_vio_1			
	•		Input	hw_vio_1			
	۲		Input	hw_vio_1			
	•		Input	hw_vio_			
	•		Input	hw_vio_			
	۲		Input	hw_vio_1			
	•		Input	hw_vio_			
	•		Input	hw_vio_1			
🛰 pause_vio_out	0]	Output	hw_vio_1			
🛰 reset_vio_out	0]	Output	hw_vio_1			
🛏 toggle_vio_out	0]	Output	hw_vio_1			
∿a vio_select	1	1	Output	hw_vio_			

Step 4: Making the ECO Modifications

- 1. In the Flow Navigator, select the Project Manager.
- 2. In the Design Runs window, right-click on impl_1 and select Open Run Directory.
- 3. The run directory opens in a file browser, as seen in the following figure. The run directory contains the routed checkpoint (top_routed.dcp) to be used for the ECO flow.

TIP: In a project-based design, the Vivado Design Suite saves intermediate implementation results as design checkpoints in the implementation runs directory. When you re-run implementation, the previous results are deleted. Save the router checkpoint to a new directory to preserve the modified checkpoint.





📙 🖓 📙 🖛 impl_1							- 0	×
File Home Share View								~ 🔞
Image: Pin to Quick access Copy Paste Image: Copy path Paste Image: Paste shortcut	Move Copy to Copy	New item •	Properties	Select all	ne			
Clipboard	Organize	New	Open	Select				
← → × ↑ 📙 > This PC > OSDisk (C	:) > Vivado_Tutorial > project_EC	O_lab > project_ECO_lab.	runs → impl_1		~ Ō	Search impl_1		م
↓ Quick access		Name	Da	te modified	Туре	Size		^
T T : D		A top_placed	10	31/2018 11:00	Vivado Checkpoin	. 2,427 KB		
Ihis PC		top_power_routed	10	31/2018 11:02	RPT File	11 KB		
💣 Network		top_power_routed.rpx	10	31/2018 11:02	RPX File	2,299 KB		
		top_power_summary_r	outed.pb 10	31/2018 11:02	PB File	1 KB		
		top_route_status.pb	10	31/2018 11:02	PB File	1 KB		
	_	dop_route_status	10	31/2018 11:02	RPT File	1 KB		
		1 / top_routed	10	31/2018 11:01	Vivado Checkpoin	. 2,768 KB		
		top_timing_summary_r	outed.pb 10	31/2018 11:02	PB File	1 KB		
		fop_timing_summary_r	outed 10	31/2018 11:02	RPT File	625 KB		
		top_timing_summary_r	outed.rpx 10	31/2018 11:02	RPX File	549 KB		
		top_utilization_placed.p	ob 10	31/2018 11:00	PB File	1 KB		
		top_utilization_placed	10	31/2018 11:00	RPT File	11 KB		
		usage_statistics_webtal	k 10	31/2018 11:03	Firefox HTML Doc	62 KB		
		usage_statistics_webtal	k 10	31/2018 11:03	XML Document	97 KB		
		vivado.jou	10	31/2018 10:44	JOU File	1 KB		
		vivado.pb	10	31/2018 11:03	PB File	1 KB		
		write_bitstream.pb	10	31/2018 11:03	PB File	7 KB		~
65 items 1 item selected 2.70 MB								

- 4. Create a new directory named ECO in the original C:/Vivado_Tutorial/ project_ECO_lab project directory, and copy the top_routed.dcp file from the implementation runs directory to that newly created directory.
- 5. From the main menu, select **File** \rightarrow **Checkpoint** \rightarrow **Open**.

The Open Checkpoint dialog box opens.

6. Navigate to C:/Vivado_Tutorial/project_ECO_lab/ECO and select the top_routed.dcp checkpoint.

A dialog box opens, asking whether to close the current project.

- 7. Click Yes.
- 8. From the main menu, select Layout \rightarrow ECO.

The ECO Layout is selected. The ECO Navigator is displayed on the left of the layout (highlighted in red in the following figure). It provides access to netlist commands, run steps, report and analysis tools, and commands to save changes and generate programming files.

The Scratch Pad in the center of the layout (highlighted in red in the following figure) tracks netlist changes, as well as place and route status for cells, pins, ports, and nets.

Note: ECOs only work on design checkpoints. The ECO layout is only available after you have opened a design checkpoint in the Vivado IDE.





To illustrate the capabilities of the ECO flow, you next change the functionality of the multiplier from a square of count[3:0] to a multiply by two.

9. From the Tcl Console, type the following command:

```
mark_objects -color blue [get_cells my_mult_0]
```

TIP: To make it easier to locate objects that are included in the ECO modifications, it helps to mark or highlight the objects with different colors.





10. Zoom into the multiplier in the schematic window and select the in2[3:0] pins.

Alternatively, you can type the following command in the Tcl Console:

select_objects [get_pins my_mult_0/in2[*]]

11. Click the **Disconnect Net** button in the Edit section of the Vivado ECO Navigator. The net is disconnected from the pins in the schematic.



The Tcl Console reproduces the disconnect_net command that you just executed in the ECO Navigator. This is useful if you want to replay your ECO changes later by opening the original checkpoint and sourcing a Tcl script with the ECO commands.



Tcl Console	? _ D @ X
Q X ≑ II E II i	
<pre>mark_objects -color blue [get_cells my_mult_0] startgroup disconnect_net -objects [list (my_mult_0/in2[3])] disconnect_net -objects [list (my_mult_0/in2[1])] disconnect_net -objects [list (my_mult_0/in2[0])] endgroup </pre>	
Type a Tcl command here	

The Scratch Pad is populated with the four nets $divClk_reg[28:25]$ that you disconnected and the multiplier input pins $my_mult_0/in2[3:0]$. Note the following in the Scratch Pad:

- The Scratch Pad connectivity column (Con) shows a check mark next to the divClk_reg[28:25] nets, indicating that they are still connected to the other multiplier inputs.
- The my_mult_0/in2[3:0] pins do not show a check mark next to them because they no longer have nets connected.
- The Place and Route (PnR) column is unchecked for everything, indicating that the changes have not yet been implemented on the device.

Scra	tch Pad		_ D @ X
Q,	¥	≑ •€ + -	¢
Con	PnR	Object Name	
		~ 😼 my_mult_0/in2	
		my_mult_0/in2[3]	
		my_mult_0/in2[2]	
		my_mult_0/in2[1]	
		my_mult_0/in2[0]	
4		~ 手 divClk_reg	
1		_ divClk_reg[25]	
4		_ divClk_reg[26]	
1		_ divClk_reg[27]	
4		⊥ divClk_reg[28]	

- 12. In the Scratch Pad, select the my_mult_0/in2[3], my_mult_0/in2[2], and my_mult_0/in2[0] pins.
- 13. In the Edit section of the Vivado ECO Navigator, click Connect Net.

The Connect Net dialog box opens.



ECO Navigator	Scratch Pad × Cell Pin Propertie	es Netlist 🔄 🗆 🖸
Edit	Q ≍ ≑ -: + -	•
Create Net	Con PnR Object Name	
of out of the t	✓ S my_mult_0/in2	
Create Cell	my_mult_0/in:	2[3]
Create Port	my_mult_0/in:	2[2]
Create Pin	my_mult_0/in:	2[1]
	my_mult_0/in:	2[0]
Connect Net	✓ ✓ 与 divClk_reg	
Disconnect Net	✓ _ divClk_reg[25]	
Poplace Dobug Probas	✓ _ divClk_reg[26]	
Replace Debug Flobes	✓ divClk_reg[27]	
Place Cell	✓ _ divClk_reg[28]	
Unplace Cell		
Run		
Check ECO		
Optimize Logical Design		
Place Design		
Optimize Physical Design		
Route Design		

14. In the Connect Net dialog box, select **<const0>** from the GROUND section.



🍌 Connect Net	×
Connect selected pins or ports with a net.	
Hierarchy:	v
Net or Bus:	
Search: Q-	
GLOBAL_CLOCK	<u>^</u>
GROUND	
_ <const0></const0>	
GND_2	
De POWER	
_ <const1></const1>	
SIGNAL	
√ count_out	
_l ¥ count_out_pre	
_ f divClk_reg	
_ſ ¥ mul_out_pre	
_ſ 兼 mult_out_pre_pre	
_[p_1_in0	
-∬ sl_iport0	
_[sl_iport1_o_1	
-∬ sI_oport0	
Fr -1	~
ок	Cancel

15. Click **OK**.

<const0> is added to the Scratch Pad.

16. Collapse the <const0> signal.

The three pins that you connected now show check marks in the Connectivity column of the Scratch Pad.



Scra	tch Pad		_ D @ X
Q	¥	≑ •ª + -	٥
Con	PnR	Object Name	
1		✓ - If divClk_reg	
1		∫ divClk_reg[25]	
1		∫ divClk_reg[26]	
1		∫ divClk_reg[27]	
1		∫ divClk_reg[28]	
		~ 😼 my_mult_0/in2	
1	-	my_mult_0/in2[3]	
1	-	my_mult_0/in2[2]	
		my_mult_0/in2[1]	
4	-	my_mult_0/in2[0]	
4		> _「 <const0></const0>	

- 17. In the Scratch Pad, select the my_mult_0/in2[1] pin.
- 18. Click Connect Net.

The Connect Net dialog box opens.

19. In the Connect Net dialog box, select **<const1>** from the POWER section.



🍌 Connect Net	×
Connect selected pins or ports with a net.	
Hierarchy:	×
Net or Bus:	
Search: Q-]
	^
GROUND	
_ <const0></const0>	
_ GND_2	
POWER	
_ <const1></const1>	
SIGNAL	
√ count_out	
- f * count_out_OBUF	
.∬ ≆ mul_out_pre	
.∬ ≭ mult_out_pre_pre	
. [p_1_in0	
.∬ sl_iport0	
.∬ sl_iport1_o_1	
sl_oport0	~
[P -1	
OK	Cancel

20. Click **OK**.

<const1> is added to the Scratch Pad.

21. Collapse the <const1> signal.

The pin that you connected now shows check marks in the Connectivity column of the Scratch Pad.



Scra	tch Pad		_ D @ X
Q,	¥	≑ +: + -	۰
Con	PnR	Object Name	
1		~ 「」 divClk_reg	
1		⊥ divClk_reg[25]	
1		⊥ divClk_reg[26]	
1		∫ divClk_reg[27]	
1		∫ divClk_reg[28]	
1		> _ <const0></const0>	
1			
1		my_mult_0/in2[3]	
1		my_mult_0/in2[2]	
1		my_mult_0/in2[1]	
1		my_mult_0/in2[0]	
1		> 」 <const1></const1>	

22. Select the **my_mult_0/in2** pin in the Scratch Pad.

This command highlights the pins in the currently open Schematic window, and shows the updated connections.

Note: Make sure that the Autofit Selection \bigcirc toggle button is highlighted in the Schematic window so you can see the entire path, as shown in the following figure.



When you observe the count signal on the LEDs, you only use four bits. The upper four bits are padded with zeroes.

Now, you will use the ECO flow to observe counter bit 24 on LED 7. The first step is to analyze the logic that drives $count_out_reg[3]$.



23. From the Tcl Console, type the following command:

select_objects [get_cells count_out[3]_i_1]

This lets you quickly identify the LUT3 that drives the $count_out_reg[3]$ register, which drives LED 3. The inputs are:

- mul_out_pre_reg[3] for pin IO
- count_out_pre_reg[3] for pin l1
- tog_state_reg for pin 12
- 24. Click the **Cell Properties** tab to view the cell properties and select the **Truth Table** tab.
- 25. Click Edit LUT Equation to view the equation for the LUT3. Note the LUT equation:

O= I1 & !I2 + I0 & I2

26. Click **Cancel** to close the window.



27. From the Tcl Console, type the following command:

select_objects [get_cells count_out[7]_i_1]



Scratch Pad Cell Properties × Netlist	? _ 🗆 🖆	Device × Package × Schematic ×	? 🗆 🖸
<pre>count_out[7]_i_1</pre>	← → Φ	← → Q, Q, X, X, Q D + − C 92 Cells 13 I/0 Ports 244 Nets	۰
I1 I0 O=I0 & I1 0 0 0 0 1 0 1 0 0 1 1 1		mul_out_pre_reg[6] FDRE F0-41 FD F0-41 CE F0-41 CE F0-41 F0-41 F0-41 F0-41 F0-53 R FDRE F0-41 FDRE F0-41	b 6
Properties Power Nets Cell Pins Iru	nth lable ∢ ▶ =		5 10

This command selects the LUT2 that drives the $count_out_reg[7]$ register, which drives LED 7 on the KCU105 board. The only inputs are tog_state_reg for pin IO and $mul_out_pre_reg[7]$ for pin I1. You need to replace the LUT2 with a 3-input LUT and connect the output of counter register $divClk_reg[24]$ to the additional input pin.

28. In the Vivado ECO Navigator, under Edit, click Create Cell.

The Create Cell dialog box opens.

- a. In the Cell name field, enter ECO_LUT3.
- b. In the Search field, enter LUT3.
- c. Select LUT3 as the cell type and copy the LUT equation O=I1 & !I2 + I0 & I2 from cell count_out[3]_i_1.
- d. Click OK.

ECO_LUT3 is added to the Scratch Pad and the schematic.

e. Right-click the newly added ECO_LUT3 cell in the Scratch Pad, then select Mark and the color red.

Note: Marking the ECO_LUT3 cell makes it easier to locate.



🝌 Create Cell		\times					
Create a cell or an array of cells at a specified level of hierarchy.							
Hierarchy:		× ···					
Cell <u>n</u> ame:	ECO_LUT3	\otimes					
Create array:	From 0 🗘 To 31 🗘						
Cell type							
Search: Q- LUT3	(1 match)						
LUT3							
LUT Equation: O=	1 & !!2 + 10 & 12	⊗ ···					
Location:		<u>F</u> ixed					
Create hierarchi	ical cell						
Cell reference nam	e:						
	ОК	Cancel					

Because you copied the LUT equation from cell $count_out[3]_i_1$, the nets must be hooked up in the same order, with the following connections:

- Net mul_out_pre[7] connected to pin I0
- Net divClk_reg_n_0_[24] connected to pin I1
- Net tog_state connected to pin I2 of ECO_LUT3
- 29. Locate the tog_state net driven by the tog_state_reg register in the schematic and select it. Alternatively you can select the net from the Tcl Console by running the following command:

```
select_objects [get_nets tog_state]
```

- **30.** Connect the I2 pin of the newly added ECO_LUT3 cell by doing the following:
 - a. Hold down the **Ctrl** key and select pin **12** in the Scratch Pad. This selects pin I2 in addition to the already selected tog_state net.
 - b. Click Connect Net.



Scratch	Pad	× Cell Pin Properties Netlist	_ 🗆 🖸	Device × Package × Schematic ×	? 🗆 1	Ľ
Q 3	E	≑ K + -	•	← → @ Q X ∑ [] + − C 93 Cells 13 I/O Ports 244 Nets		\$
Con Pri	nR	Object Name	Â	mul_out_pre_reg[5] FO-3 H FO-1 F0-41 CE FO-3 II FO-4 5 % FO-53 R FO-3 II FO-3 FD-41 CE D R FO-3 II FO-3 FD-55 R FD-73 II III O FO-1 FDRE FDRE FD-3 III O FO-1 FD-70-3 III O FO-1 FO-1 FD-8 FD-9 III O FO-1	₩ F0-53 R F0-224 F0-227 C F0-27 C F0-27 C F0-27 C F0-1 C F0-1 C	
3 3 4 3 3 4	333	 my_mult_0/in2[3] my_mult_0/in2[2] my_mult_0/in2[1] my_mult_0/in2[0] 		FO-3 1 FO-1 1	F0=2274	Ē
		 J <constl></constl> ECO_LUT3 0 10 11 12 		FDRE mul_out_pre_reg[7] FO=2274 C FO=41 C FO=41 C FO=3 10 FO=3 10 FO=3 10 FO=41 C FO=3 10 FO=4	F0-57 F0-1 B F0-53 F0-63 F0-63 F0-63 F0-63 F0-63	

31. Locate the mul_out_pre[7] net in the schematic and select it.

Alternatively, you can select the net from the Tcl Console by executing the following command:

select_objects [get_nets mul_out_pre[7]]

- **32.** Connect the IO pin of the newly added ECO_LUT3 cell by doing the following:
 - a. Hold down the **CTRL** key and select pin **IO** in the Scratch Pad. This selects pin IO in addition to the already selected mul_out_pre[7] net.
 - b. Click Connect Net.
- **33.** Locate the divClk_reg_n_0_[24] net in the schematic and select it.

Alternatively, you can select the net from the Tcl Console by executing the following command:

select_objects [get_nets divClk_reg_n_0_[24]]

- 34. Connect the I1 pin of the newly added ECO_LUT3 cell by doing the following:
 - a. Hold down the CTRL key and select pin 11 from the Scratch Pad. This selects pin I1 in addition to the already selected $divClk_reg_{n_0}[24]$ net.
 - b. Click Connect Net.

Next, you need to connect the updated logic function implemented in the newly created LUT3 to the D input of count_out_reg[7]. The first step is to delete the LUT2 that was previously connected to the D input.

35. Select the LUT2 count_out[7]_i_1 in the schematic window.

Alternately, you can select it by executing the following command in the Tcl Console:

select_objects [get_cell count_out[7]_i_1]

36. In the main toolbar, click the Delete button \times to delete the selected cell.



37. Select the net connected to the D input of the count_out_reg[7] register in the schematic window.

Alternatively you can select the net from the Tcl Console by executing the following command:

select_objects [get_nets count_out[7]_i_1_n_0]

- 38. Connect the O pin of the newly added ECO_LUT3 cell by doing the following:
 - a. Hold down the CTRL key and select pin O from the Scratch Pad.
 - b. Click Connect Net.



The ECO modifications are complete.

Step 5: Implementing the ECO Changes

Before you place and route the updates, you need to check for any illegal logical connections or other logical issues introduced during the ECO that would prevent a successful implementation of your changes.

1. In the Vivado ECO Navigator, under Run, click Check ECO.

The following figure shows the messages generated by the ECO DRC.

- The two Critical Warnings are due to the partially routed signals that are a result of the ECO and will be cleaned up during incremental place and route.
- The Warning message is due to nets in the debug hub instance that do not drive any loads. This Warning can be ignored.
- No other warnings were issued and you are ready to implement the changes.



DRC		? _ ם ×
Q 素 ♦ ■	🖌 🌗 2 Critical War	nings 📝 🕕 1 Warning 🛛 Hide All
Name	Severity ^ 1	Details
✓ ➡ All Violations (3)		
✓ ➡ Implementation (3)		
Y 🐱 Routing (3)		
Y 🐱 Chip Level (3)		
V 9 RTSTAT-5 (1)		
RTSTAT #1	Critical Warning	6 net(s) have a partial antenna. The problem bus(es) and/or net(s) are divClk reg, mul out pre[7], tog state.
V 9 RTSTAT-6 (1)		
RTSTAT #1	Critical Warning	6 net(s) have a partial conflict. The problem bus(es) and/or net(s) are divClk reg, GLOBAL_LOGIC0, GLOBAL_LOGIC1.
RTSTAT-10 (1)		
RTSTAT #1	Warning	27 net(s) have no routable loads. The problem bus(es) and/or net(s) are aempty fwft i, ctl reg, ctl reg, en 2[1], ctl reg, en
< cdc_1 (3 violations)		>

Because you added additional logic, you need to place the logic using the incremental place, and then route the updated net connections using incremental route.

2. In the Vivado ECO Navigator, under Run, click Place Design.

The Place Design dialog box opens, allowing you to specify additional options for the place_design command. For this exercise, do not specify additional options.

3. Click OK.

i Place Design
Run design placement with the specified options.
Options:
OK Cancel

4. Vivado runs the incremental placer.

At the end of the $place_design$ step, the incremental Placement Summary is displayed in the Tcl Console.



+				
Incremental Placement Summary				
туре	I	Count	I	Percentage
<pre>+ I Total instances</pre>	I	4880	I	100.00
Reused instances	1	4878	T	99.96
Non-reused instances	1	2	T	0.04
New	1	1	T	0.02
Discarded illegal placement due to netlist changes	I	1	I	0.02
+ Incremental Placement Runtime Summary				
<pre>+ Initialization time(elapsed secs)</pre>				4.08
Incremental Placer time(elapsed secs)				7.29

The incremental placement summary shows that the following two cells did not have their previous placement reused:

- The new ECO_LUT3 cell, which had to be placed from scratch
- The count_out_reg[7] cell, which had to get updated placement due to the placement of the ECO_LUT3 driving it
- 5. In the Vivado ECO Navigator, under Run, click Route Design.

The Route Design dialog box opens.



Depending on your selection, you have four options to route the ECO changes:

- Incremental Route: This is the default option.
- Route selected pin: This option limits the route operation to the selected pin.



- Route selected non-Power nets: This option routes only the selected signal nets.
- Route selected Power nets: This option routes only the selected VCC/GND nets.

In this case, the best choice is to route the changes you made incrementally.

- 6. Select Incremental Route.
- 7. Click OK.

At the end of the $route_design$ step, the incremental Routing Reuse Summary displays in the Tcl Console.

Incremental Routing Reus	e Summan	ry			I
Гуре	Count	t	Percentage		I
Fully reused nets Partially reused nets Non-reused nets	 	3761 3 1		99.89 0.08 0.03	1

Most of the nets did not require any routing and have been fully reused.

TIP: It is a good idea to run *report_route_status* after the route operation to make sure all the nets have been routed and none have any routing issues. This is especially true if you only routed selected pins or selected nets and want to make sure you have not missed any routes.

8. In the Tcl Console, run the report_route_status command.

The Design Route Status looks similar to the following status.

Design Route Status

	:	<pre># nets</pre>	5
	:		:
<pre># of logical nets</pre>	:	5160	:
<pre># of nets not needing routing</pre>	:	1363	:
<pre># of internally routed nets</pre>	:	1306	:
<pre># of nets with no loads</pre>	:	57	:
<pre># of routable nets</pre>	:	3797	:
<pre># of fully routed nets</pre>	:	3797	:
<pre># of nets with routing errors</pre>	:	0	:
	:		:

Before you generate a bitstream, run the ECO DRCs on the design.

9. In the ECO Navigator, click Check ECO. Make sure no Critical Warnings are generated.





10. In the Vivado ECO Navigator, under Program, click Save Checkpoint As.

The Save Checkpoint As dialog box opens and you can specify a name for the checkpoint file to write to disk.

11. Click **OK** to save a checkpoint file with your changes.

🝌 Save Checkpoint	As	×
Create a checkpoin database.	t file that contains the netlist, XDC constraints, and the physical	
Checkpoint file:	C:/Vivado_Tutorial/project_ECO_lab/ECO/checkpoint_1.dcp 🚳 🚥	
	OK Cancel	

12. In the Vivado ECO Navigator, under Program, click Generate Bitstream.

The Generate Bitstream dialog box opens.

You can specify a name for a Bit file and select the desired options for the write_bitstream operation.

13. Click **OK** to generate a bitstream with your changes.





🝌 Generate	Bitstream		×				
Create a programming file from the current design							
Bit File	Bit File :/Vivado_Tutorial/project_ECO_lab/ECO/project_top_routed.bit 😒						
Options	i -						
-ra	w_bitfile		<u>^</u>				
-m	ask_file						
-no	_binary_bitfile						
-bi	n_file						
-re	adback_file						
-lo	gic_location_file						
-ve	rbose		~				
Select	an option above to see a descr	iption of it OK Car	ncel				

14. In the Vivado ECO Navigator, under Program, click Write Debug Probes.

The Write Debug Probes dialog box opens.

À Write Debug I	Probes	×
Write debug pro	obes to a file.	4
<u>F</u> ile Name: <u>O</u> verwrite	vado_Tutorial/project_ECO_lab/ECO/probes_1.II®	
	OK Car	icel

You can specify a name for a $.\,{\tt ltx}$ file for your debug probes.



15. Click OK to generate debug probes file (LTX).

This command allows you to generate a new .ltx file for your debug probes. If you made changes to your debug probes using the Replace Debug Probes command, you need to save the updated information to a new debug probes file to reflect the changes in the Vivado Hardware Manager.

16. Follow the instructions in Step 3: Validating the Design on the Board to download the generated bitstream programming file and debug probes file into the target FPGA device using the Hardware Manager to check your ECO modifications.

Step 6: Replacing Debug Probes

Another powerful feature of the Vivado ECO flow is the ability to replace debug probes on a previously inserted Debug Hub. After the debug probes have been replaced, a new LTX file can be generated that contains the updated debug probe information.

To replace a debug probe in your previously modified design, do the following:

1. From the main menu, select File \rightarrow Checkpoint \rightarrow Open.

🝌 Open Check	point			\times
Look <u>i</u> n:	ECO	~	t û 🖵 ± 🍐 🖻 🗙 C	
🝌 checkpoin	_1.dcp	Recent Directories		
A top_routed	ldcp	C:/Vivado_Tutorial/project_E	ECO_lab/ECO	~
		File Preview		
		File: checkpoint_1.dcp Directory: C/Nkado_Tutorialy Created: Today at 12:05 FM Accesset: Today at 12:05 FM Modified: Today at 11:57 AM Size: 2:7 MB Type: Checkpoint design Owmer: XLNXigpocklas	rojed_ECO_lab/ECO	
File <u>n</u> ame:	checkpoint_1.dcp			
Files of type:	Vivado Checkpoint Files (.dcp)			~
			ок	Cancel

The Open Checkpoint dialog box opens.

- 2. Browse to the C:/Data/Vivado_Tutorial/project_ECO_lab/ECO directory and select the previously saved checkpoint_1.dcp file.
- 3. Close any previously open checkpoints.
- 4. From the main menu, select **Layout** \rightarrow **ECO**.
- 5. In the Vivado ECO Navigator, under Edit, click Replace Debug Probes.

The Replace Debug Probes dialog box opens.



In this example, you will replace the net $reset_vio$ that is connected to probe4 of u_ila_0 with the net $toggle_vio$.

- 6. Scroll to the bottom of the probes for u_{ila_0} in the Replace Debug Probes dialog and click the **reset_vio** net name in the Probe column to select it.
- 7. Click the Edit Probes button \checkmark .

🔥 Replace Debug Probes 🛛 🕹					
Use the Edit Probes button to replace one or more debug probes. To reflect these changes in the Vivado Hardware Manager, regenerate the debug probes file (LTX).					
王 ≑ ⊘ C ↑ ↓					
Search: Q-					
Name	Probe				
• Ch 0	_ ≸ count_out_pre[0] / ∧				
Och 1	_ f ∗ count_out_pre[1] Ø				
Och 2	_ f ∗ count_out_pre[2] Ø				
Och 3	_ ★ count_out_pre[3] 🖉				
✓ ₱ probe2 (8)					
Ch 0	∫ * mul_out_pre[0] ⊘				
Och 1	∫ * mul_out_pre[1] ⊘				
Ch 2	∫ * mul_out_pre[2]				
Ch 3	∫ * mul_out_pre[3]				
Ch 4	∫ * mul_out_pre[4]				
Ch 5	∫ * mul_out_pre[5]				
Ch 6	∫ * mul_out_pre[6]				
Ch 7	_ ≤ mul_out_pre[7]				
✓ ➡ probe3 (1)					
Ch 0	∫ * pause_vio 🖉				
✓ ➡ probe4 (1)					
Ch 0	∫ ≉ reset_vio 🖉				
✓ ₱ probe5 (1)					
Ch 0	🖌 🛎 state 🖉 🗸				
Probes changed: 0					
	OK Cancel				

The Choose Nets dialog box opens.

8. Choose a new net to connect to the debug probe probe4 by doing the following:



- a. Type toggle_vio in the search field of the Choose Nets dialog box.
- b. Click Find.
- c. Select the $toggle_vio$ net, and move it to the Selected names section.
- d. Click OK.

À Choose Nets					×
Choose nets to replace existing prob	es.				4
Properties					
NAME 🗸	contains 🗸 🗸	toggle_vi	0	⊗ +	
🗌 <u>R</u> egular expression <i> S</i> earch	hierarchically 🕑 Displa	ay unique n	ets		
Of o <u>bj</u> ects:					
		<u>F</u> ind	1		
Found: 2			Selected: 1 of 1		A Z ♣ Z A
_ toggle_vio _			_ toggle_vio		
					\times
		→			Ť
		⇒			1
					+
					<u>×</u>
				ОК	Cancel

- 9. In the Replace Debug Probes dialog box, click **OK**.
- 10. Repeat steps 5 through 14 of Step 5: Implementing the ECO Changes to generate an updated design checkpoint, bitstream file, and probes file (LTX).

The updated debug probes file has the $reset_vio$ net for probe4 replaced with net $toggle_vio$, which you can verify when you program the device with the updated bit file and debug probes file.



hw_ila	_1		? _ D 7 ×
nboard Options Settings - hw_ila_1	Waveform - hw_ila_1		? _ 🗆 X
	Q + - & >	» 📕 🗗 🔍	Q X • ± ± + + +
	ILA Status: Idle		
	Name	Value	o
ash	> National count_out_OBUF_1[7:0]	06	06
	> Variable Count_out_pre[3:0]	6	6
a 🗄	> 😻 mul_out_pre[7:0]	24	24
	🌆 pause_vio	0	
5	14 state	0	
itatu	15 toggle_vio	0	
			Updated at: 2017-Apr-03 12:53:54 🗸
		<	
	Trigger Setup - hw_ila_1 Ca		

Related Information

Step 5: Implementing the ECO Changes

Conclusion

In this lab you learned the following:

- Made changes to the previously implemented design using the Vivado ECO flow.
- Implemented the changes using incremental place and route.
- Generated a bitstream and probes file with your changes to configure the FPGA.
- Used the Replace Debug Probes command to switch the sources for debug probes in the design.





Appendix A

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.

