

Vivado Design Suite Tutorial

Using Constraints

Vivado Design Suite

UG945 (v2021.2) November 17, 2021

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





Revision History

The following table shows the revision history for this document.

Section	Revision Summary
11/17/2021 V	ersion 2021.2
Release updates	General 2021.2 release updates.
08/13/2021 V	ersion 2021.1
Release updates.	General 2021.1 release updates.





Table of Contents

Revision History2
Using Constraints Tutorial4
Tutorial Design Description5
Hardware and Software Requirements5
Preparing the Tutorial Design Files5
Navigating Content by Design Process6
Lab 1: Defining Timing Constraints and Exceptions7
Step 1: Opening the Example Project
Step 2: Defining Constraint Sets and Files
Step 3: Creating Timing Constraints14
Step 4: Using the Constraints Editor22
Step 5: Saving Constraints28
Step 6: Clock Interaction Report30
Step 7: Timing Summary Report
Conclusion34
Lab 2: Setting Physical Constraints
Step 1: Opening the Project
Step 2: Adding Placement Constraints
Step 3: Defining Additional Physical Constraints
Step 4: Defining Constraints with Object Properties40
Step 5: Saving Constraints44
Conclusion45
Appendix A: Additional Resources and Legal Notices
Xilinx Resources
Documentation Navigator and Design Hubs46
Please Read: Important Legal Notices 47





Using Constraints Tutorial

IMPORTANT! This tutorial requires the use of the Kintex[®]-7 family of devices. You will need to update your Vivado[®] tools installation if you do not have this device family 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 is comprised of two labs that demonstrate aspects of constraining a design in the Vivado[®] Design Suite. The constraints format supported by the Vivado[®] Design Suite is called Xilinx[®] Design Constraints (XDC), which is a combination of the industry standard Synopsys[®] Design Constraints and proprietary Xilinx[®] constraints. For more information on Timing Closure, see the *UltraFast Design Methodology Timing Closure Quick Reference Guide* (UG1292).

VIDEO: You can also learn more about defining constraints in the Vivado Design Suite by viewing the quick take video at Vivado Design Constraints Overview.

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:

- Essentials of FPGA Design
- Vivado Design Suite Static Timing Analysis and Xilinx Design Constraints

XDCs are not just simple strings; they are Tcl commands that the Vivado Tcl interpreter sequentially reads and parses. You can enter design constraints in several ways at different points in the design flow. You can store XDCs in one or more files that can be added to a constraint set in Vivado Project Mode, or read the same files directly into memory using the read_xdc command in Non-Project mode. For more information on Project and Non-Project modes, refer to the *Vivado Design Suite User Guide: Design Flows Overview* (UG892). With a design open in Vivado tools, you can also type constraints as commands directly in the Tcl Console when working in the Vivado IDE or at the Tcl command prompt when working outside of the IDE. This is particularly powerful for defining, validating, and debugging new constraints interactively in the design.

The Vivado Design Suite synthesis and implementation tools are timing driven. Having accurate and correct timing constraints is vital for meeting design goals and ensuring correct operation. Because the Vivado tools are timing driven, it is important to fully constrain a design, but not over-constrain, or under-constrain it. Over-constraining a design can lead to long compile times and sub-optimal results because the tool can struggle with unrealistic design objectives. Under-constraining a design can cause the Vivado tools to perform unnecessary optimizations, such as examining paths with multicycle delays or false paths, and prevent focus on the real critical paths.

This tutorial discusses different methods for defining and applying design constraints.



Tutorial Design Description

The sample design used throughout this tutorial consists of a small design called project_cpu_netlist. There is a top-level EDIF netlist source file, as well as an XDC constraints file.

The design targets an XC7K70T device. A small design is used to allow the tutorial to be run with minimal hardware requirements and to enable timely completion of the tutorial, as well as to minimize the data size.

Hardware and Software Requirements

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

See 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

You can find a ZIP file containing the files for this tutorial in the examples directory of the Vivado Design Suite software installation, at the following location:

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

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

The location of the extracted Vivado_Tutorial directory is referred to as the <Extract_Dir> in this Tutorial.

You can also extract the provided ZIP file at any time to restore the files to their starting condition.

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





Navigating Content by Design Process

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

- Hardware, IP, and Platform Development: Creating the PL IP blocks for the hardware platform, creating PL kernels, functional simulation, and evaluating the Vivado[®] timing, resource use, and power closure. Also involves developing the hardware platform for system integration. Topics in this document that apply to this design process include:
 - Lab 1: Defining Timing Constraints and Exceptions
 - Lab 2: Setting Physical Constraints





Lab 1

Defining Timing Constraints and Exceptions

In this lab, you will learn two methods of creating constraints for a design. You will use the Kintex[®]-7 CPU Netlist example design that is included in the Vivado[®] IDE.

Step 1: Opening the Example Project

- 1. Open Vivado IDE.
 - On Linux:
 - 1. Change the directory where the lab materials are stored.

cd <Extract_Dir>/Vivado_Tutorial

- 2. Launch the Vivado IDE: vivado
- On Windows:
 - 1. Launch the Vivado Design Suite IDE:

Start \rightarrow All Programs \rightarrow Xilinx Design Tools \rightarrow Vivado 2021.x \rightarrow Vivado 2021.x

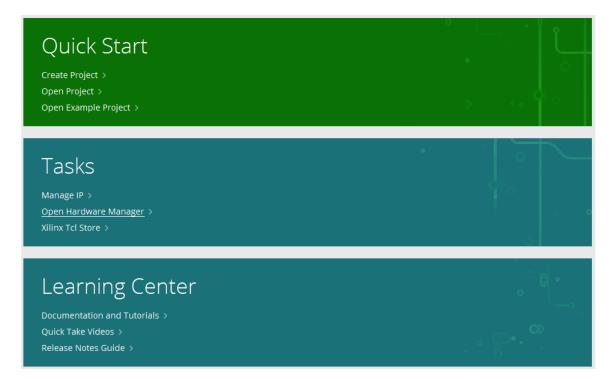
Note: Your Vivado Design Suite installation might be called something other than Xilinx Design Tools on the Start menu.

Note: As an alternative, click the Vivado 2021.x Desktop icon to start the Vivado IDE.

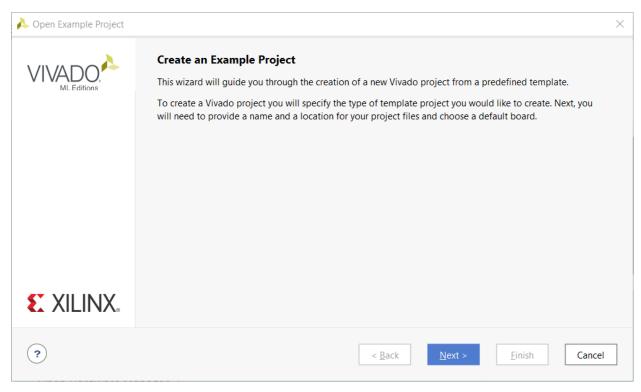
The Vivado IDE Getting Started page contains links to open or create projects and to view documentation.

2. From the Getting Started page, click Open Example Project, as shown in the following figure.





The Open Example Project wizard opens to the Create an Example Project page, as shown in the following figure.



3. Click Next.



4. In the Select Project Template page, select **CPU (Synthesized)**, as shown in the following figure.

	Open Example	e Project	\odot
lect Project Template ect one of the below predefined templ	ates on which to base your new projec	t	1
To fetch the latest available example emplates	e designs from git repository, click on 'F Description	Refresh' button. Dismiss	
Q, ★ ♦ ± ⊝ €	CPU (Synthesized) Large synthesized netlist project		
Xilinx AXI DMA BFT CIPS DDR PL Debug		nbone	
CPU (HDL) CPU (Synthesized) Versal Extensible Embedded Pla MicroBlaze Design Presets Zynq UltraScale+ MPSOC Design Multi-Rate GTY Versal ACAP CIPS PS VIP Versal MicroBlaze Design Preser	Open RISC CPU	Clock USB O	
 Wavegen (HDL) Zynq-7000 Design Presets 	FFT BFT	USB 1	

- 5. Click Next.
- 6. In the Project Name page, specify the project name and location, as shown in the following figure.



🝌 Open Examp	le Project				\times
Project Name Enter a name for y	our project and specify a directory where the	e project data files wil	l be stored		4
<u>P</u> roject name:	project_cpu_netlst				\otimes
Project location:	C:/Vivado_Tutorial				⊗
🗹 Create proje	ct subdirectory				
Project will be c	eated at: C:/Vivado_Tutorial/project_cpu_ne	etist			
?		< <u>B</u> ack	<u>N</u> ext ≻	<u>F</u> inish	Cancel

- Project name: project_cpu_netlist
- **Project location:** <Extract_Dir>
- 7. Click Next.
- 8. In the Default Part page, select the default part as shown in the following figure.

٨	Open Example P	Project							\times
	efault Part oose a default Xilinx	c part for your proj	ect.						4
	<u>S</u> earch: Q- Part xc7k70tfbg676-2	I/O Pin Count 676	Available IOBs 300	V LUT Elements 41000	FlipFlops 82000	Block RAMs	Ultra RAMs	DSPs 240	Gb 1 8
	<								>
(< <u>B</u> ac	k	<u>N</u> ext ≻	<u>F</u> inish	Ca	ancel

A New Project Summary appears, as shown in the following figure.



🍌 Open Example Project		\times
MLEditions	 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	
(?)	< Back <u>N</u> ext > <u>Finish</u> Cance	21

9. Click **Finish** to finish creating the project.

The Vivado IDE displays the default view of the opened project, as shown in the following figure.





		-	U X				
			Ready				
		II Def					
Flow Navigator 🗄 🕴 ? 💶	PROJECT MANAGER - project_cpu_netist		? :				
	Sources ? _ D D X	Project Summary	7 🗆 🗆 X				
	Q ± 0 + 0 0	Overview Dashboard					
	> Gi Design Sources (1)						
Elle Edit Flow Iools Repo Commentation Edit Flow Iools Repo PROJECT MANAGER Settings Add Sources Language Templates SIMULATION Run Simulation NETLIST ANALYSIS > Open Synthesized Design IMPLEMENTATION PRUMEMENTATION PRUMEMENTATION PRUMEMENTATION PRUMEMENTATION PRUMEMENTATION PRUMEMENTATION PRUMEMENTATION	> 🛱 Constraints (2)	Settings Edit					
SIMULATION		Project name: project_cpu_netist					
Run Simulation		Product family: Kintex-7					
NETHOT AMAL VOID		Project part xc7k70tbg676-2					
Elle Edit Flow Tools Report	Libraries Compile Order Tarcel lancuage VHDL						
	Provide a second se	Simulator language: Mixed					
and the second		Implementation					
> Open Implemented Design		Status: Not started					
PROGRAM AND DEBUG	Select an object to see properties	Part. xc7k70tfbg676-2					
Generate Bitstream	ectant an extent to any high man	Strategy: Vivado Implementation Defaults					
> Open Hardware Manager		Project Summary Overview Dashboard Overview Dashboard Settings Edit Project Couton Project family: Project family: Kinds Edit Project family: Kinds Top module family: Kinds Project family: Kinds Status: Merentation Status: Not started Messages: No errors or warnings Part: xc7k70tbp676-2 Status: No errors or warnings Report Strategy: Vivado Implementation Defaults Report Strategy: Vivado Implementation Defaults	~				
Image: Settings Image: Settings Add Sources Language Templates Language Templates > © Design Sources (1) Simulation > © Design Sources (1) Simulation > © Interplates NETLIST ANALYSIS > © Den Synthesized Design Image: Numplementation > Open Synthesized Design PROGRAM AND DEBUG Image: Select Image: Complementation > Open Hardware Manager Open Hardware Manager Tel Console Messages Log Image: Interplates Image: Interplates Name Constraints		1					
	Tcl Console Messages Log Reports Design Runs ×		? _ 🗆 🖾				
	Q ≚ ≑ I4 ≪ ▶ ≫ + %						
	and the second	Total Power Failed Routes LUT FF BRAM URAM DSP Start E	and the second se				
	impl_1 constrs_2 Not started		vivado imple				
	<		3				
/iew							

Step 2: Defining Constraint Sets and Files

Start by creating a new constraint set and adding an empty XDC constraints file to it. The example design already contains two constraint sets, but you do not use them for this lab.

- 1. From the Flow Navigator, click Add Sources in the Project Manager section.
- 2. From the Add Sources dialog box, select Add or create constraints.
- 3. Click Next.
- 4. From the Add or Create Constraints dialog box, use the Specify Constraint Set drop-down menu to select **Create Constraint Set** as shown in the following figure.



5. In the Create Constraint Set dialog box, specify the constraint set name as lab1 and click OK.



- 6. Enable the **Make active** check box.
- 7. Click the Add button +, and select Create File to add a new XDC file to the project.

The Create Constraints File dialog box appears.

🝌 Create Co	nstraints File	×
Create a new co	onstraints file and add it to your project	4
<u>F</u> ile type:	XDC	~
F <u>i</u> le name:		
Fil <u>e</u> location:	<local project="" to=""></local>	*
?	ОК Са	ancel

- 8. Type timing as the file name, and leave the file location set to <Local to Project>.
- 9. Click OK.

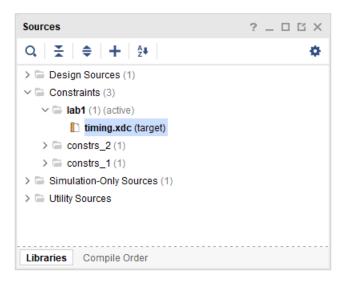
The timing.xdc file is added to the lab1 constraint set.

10. Click **Finish** to complete the creation of the new constraint set and XDC file.

You should see the new constraint set and XDC file in the Sources window, as shown in the following figure. The constraint set is made active, as you directed when you created it.

11. In the Sources window (shown in the following figure), right-click timing.xdc and select Set as Target Constraint File.

This sets the timing.xdc file as the target XDC file. All constraints added to the design are saved in the target XDC file.





Step 3: Creating Timing Constraints

In this step, you will open the synthesized design and use the Vivado[®] Timing Constraints wizard. The Timing Constraints wizard analyzes the gate level netlist and finds missing constraints. Use the Timing Constraints wizard to generate constraints for this design.

- 1. From the Flow Navigator, click **Open Synthesized Design**.
- 2. When the synthesized design opens, click **Constraints Wizard** under the Synthesized Design section.

The introduction page of the Timing Constraints wizard appears. This page describes the types of constraints that the wizard creates: Clocks, Input and Output Ports, and Clock Domain Crossings.

3. After reading the page, click **Next** to continue.

The Primary Clocks page of the Timing Constraints wizard displays all the clock sources with a missing clock definition. For this design, the wizard detected five missing clock constraints that are needed to time logical paths, and four missing clock constraints that are only needed to verify pulse width and minimum or maximum period requirements.

2	100 D II												
	Object	Name		Frequency (MHz) Per		Period ((ns) Rise A		(ns)	Fall At	(ns)	Jitter (ns)	
	M TXOUTCLK	mgtEngine/RC	_ROC	unde	fined	fined undefined							
	M TXOUTCLK	mgtEngine/ROCKETIO_WRAPPER_TILE_i/gt2_ROC			unde	fined	ined undefined						
	M TXOUTCLK	mgtEngine/ROCKETIO_WRAPPER_TILE_i/gt4_ROC			unde	fined	undefi	ined					
	III TXOUTCLK	mgtEngine/RC	_ROC	unde	fined	ned undefined							
_	ILL INCOULDER												
nst	I sysClk	sysClk			unde	fined	undefi	ined					
nst	M sysClk	sysClk			unde	fined	undefi	ined					
nst	SysClk	sysClk	Name	Frequ	unde ency (MHz)	fined			At (ns)	Fall At	(ns)	Jitter	(ns)
nst	In sysClk raints for Pulse Wi	sysClk idth Check Only		Frequ		Period			At (ns)	Fall At	(ns)	Jitter	(ns)
nst	I sysClk raints for Pulse Wi b / I J	sysClk idth Check Only LK_PAD_P_IN	Name	Frequ	ency (MHz)	Period	I (ns)		At (ns)	Fall At	(ns)	Jitter	(ns)
	In sysClk	sysClk idth Check Only LK_PAD_P_IN LK_PAD_P_IN	Name TILE0_REFCLK_PAD_P_IN	Frequ	ency (MHz) undefined	Period unde unde	l (ns) efined		At (ns)	Fall At	(ns)	Jitter	(ns)

You will fill in the periods for the five missing primary clocks in the design. The second category of clocks, shown in the Constraints for Pulse Width Check Only table, is optional. For this example, do not add these last constraints.

4. In the Recommended Constraints table, each row of the wizard is a missing constraint. To specify a constraint in the table, click the cell in the period column for the clock and type the value from the following table to fill in the periods of the five missing primary clocks in the design. When you set the period for a clock, the frequency is automatically populated.



Table 1: Values to Use on the Primary Clock Page of the Wizard

Primary Clock	Period (ns)
mgtEngine/ROCKETIO_WRAPPER_i/gt0_ROCKETIO_WRAPPER_TILE_i/GT0_TXOUTCLK_OUT	12.8
mgtEngine/ROCKETIO_WRAPPER_i/gt0_ROCKETIO_WRAPPER_TILE_i/GT2_TXOUTCLK_OUT	12.8
mgtEngine/ROCKETIO_WRAPPER_i/gt0_ROCKETIO_WRAPPER_TILE_i/GT4_TXOUTCLK_OUT	12.8
mgtEngine/ROCKETIO_WRAPPER_i/gt0_ROCKETIO_WRAPPER_TILE_i/GT6_TXOUTCLK_OUT	12.8
sysClk	10.0

If you do not want to enter a constraint for a specific clock, uncheck the box next to the clock.

For more information about how the wizard finds these missing constraints, click the Quick Help button ("?") in the lower left-hand corner of the wizard. The quick help pages are context specific and contain more information about the topologies the wizard is looking for and an explanation as to why the constraint is being suggested.

The completed page looks like the following figure.





Q,	122 🖉 🖽										
	Object	Name		Frequen	cy (MHz)	Period	(ns) Rise	At (ns)	Fall At (ns) Jitter (ns)	
	III TXOUTCLK	mgtEngine/ROCKETIO_WRAPPER_TILE_i/gt0_ROC			78.125	12	.800	0.000	6.40	0	
	M TXOUTCLK	mgtEngine/R0	OCKETIO_WRAPPER_TILE_i/gt2	_ROC	78.125	12	.800	0.000	6.40	0	
	TXOUTCLK	mgtEngine/R0	OCKETIO_WRAPPER_TILE_i/gt4	_ROC	78.125	12	.800	0.000	6.40	0	
	M TXOUTCLK	mgtEngine/R0	OCKETIO_WRAPPER_TILE_i/gt6	ROC	78.125	12	.800	0.000	6.40	0	
	M sysClk	sysClk			100.000	10	.000	0.000	5.00	0	
	M TILE0_REFC		TILE0_REFCLK_PAD_P_IN TILE1_REFCLK_PAD_P_IN	undefin undefin		defined defined					
	III TILE1_REFO	LK_PAD_P_IN	TILE1_REFCLK_PAD_P_IN	undefin	ed un	defined					
	I TILE2_REFC	LK_PAD_P_IN	TILE2_REFCLK_PAD_P_IN	undefin	ed un	defined					
	M TILE3_REFC	LK_PAD_P_IN	TILE3_REFCLK_PAD_P_IN	undefin	ed un	defined					
Tcl (Command Preview	(5) Existing	Create Clock Constraints (0)								

5. Click **Next** to continue.

The primary clock constraints have been added to the design. Next, the wizard looks for unconstrained generated clocks. Generated clocks are derived from primary clocks in the FPGA fabric. A good example would be a binary counter used to create a divided clock.

In this design, the wizard determined that there are no unconstrained generated clocks.

6. Click **Next** to continue.

Next, the wizard looks for forwarded clocks. A forwarded clock is a generated clock on a primary output port of the FPGA. These are commonly used for source synchronous buses when the capture clock travels with the data.

The wizard has determined that there are no unconstrained forwarded clocks in the design.

7. Click **Next** to continue.



Next, the wizard looks for external feedback delays. MMCM or PLL feedback delay outside the FPGA is used to compute the clock delay compensation in the timing reports.

The wizard did not find any unconstrained MMCM external feedback delays in the design.

8. Click **Next** to continue.

Next, the wizard looks at the input delays. The following figure shows the Input Delay page of the Timing Constraints wizard. There are three sections to the page.

affe	ct implementation quality of r	esults. More in	fo								
ecor	nmended Constraints								Delay Parameters		
~	Interface	Clock	Synchronous	^ 1	Alignment	^ 2	Data Rate and Edge	^ 3			
~	Dataln_pad_0_i[*]	III sysClk	System	~	Edge	~	Single Rise	v ^	Clock period:	10	ns
<	Dataln_pad_1_i[*]	L sysClk	System	~	Edge	~	Single Rise	~	tco_min:	undefined	ns
~	LineState_pad_0_i[*]	III sysClk	System	~	Edge	~	- Single Rise	~	tco_max:	undefined	ns
✓	LineState_pad_1_i[*]	II sysClk	System	~	Edge	~	Single Rise	~	trce_dly_min:	undefined	ns
✓	VStatus_pad_0_i[*]	I sysClk	System	~	Edge	~	Single Rise	~			
✓	VStatus_pad_1_i[*]	I sysClk	System	~	Edge	~	Single Rise	~	trce_dly_max:	undefined	ns
✓	GTPRESET_IN	🛄 mgtEngi	System	~	Edge	~	Single Rise	~			
✓	GTPRESET_IN	👖 mgtEngi	System	~	Edge	~	Single Rise	~		5	
✓	GTPRESET_IN	🖺 mgtEngi	System	~	Edge	~	Single Rise	~			
✓	GTPRESET_IN	🚺 mgtEngi	System	~	Edge	~	Single Rise	~	Rise Max = tco_max +	trce dly max	
✓	? RxActive_pad_0_i	SysCik	System	~	Edge	~	Single Rise	~	Rise Min = tco_min +		
✓	? RxActive_pad_1_i	SysClk	System	~	Edge	~	Single Rise	~			
✓	? RxError_pad_0_i	SysClk	System	~	Edge	~	Single Rise	~			
✓	? RxError_pad_1_i	SysCik	System	~	Edge	~	Single Rise	× .	A	pply	
	iommand Preview (49) E	Existing Set Inp	out Delay Constra	aints (0)	Waveform - S	iystem E	Edge Single Rise				_
	data			XXX			data			XX	_
				⊳ (tco_r	min + trce_dly_		trce_dly_max)				_

In section A, you can see all the input ports that are missing input delay constraints in the design. In this table, you select the timing template you would like to use to constrain the input.

In section B, you provide the delay values for the template. This section will change depending on the template chosen in section A.

In section C, there are three tabs:

• **Tcl Command Preview:** Previews the Tcl commands that will be used to constrain the design.



- Existing Set Input Delay Constraints: Shows input delay constraints that exist in the design.
- Waveform: Displays the waveform associated with the template.

Next, you will fill out the form based on the following table.

- 9. Click the **Clock** column header to sort the table alphabetically by clock name.
- 10. Select the template in section A, enter the values in section B, and observe the Tcl commands, existing input delays, and the template specific waveform in section C.
- 11. Skip entering the first four constraints shown in following table by unchecking the box to the left of the constraint. In this particular case, you false path these paths from the GTPRESET_IN port later, as it is an asynchronous reset signal synchronized inside the design.

The blocks of colored rows in the following table can all be entered at the same time in the Input Delays page of the Timing Constraints wizard, shown in the following figure, by selecting multiple rows in the wizard (use the **Shift** or **Ctrl** buttons and click to select multiple rows) and then entering the values once. Some inputs are constrained relative to virtual clocks, because they are captured by an internal generated clock with a waveform different than the board clock. In this case, the wizard creates a virtual clock with the same frequency and waveform as the internal clock, and recommends a constraint relative to the virtual clock.

The following table is shaded to indicate which groups of signals can be entered in this manner.

Interface	Clock	Synchronous	Alignment	Data Rate and Edge	tco_min (ns)	tco_max (ns)	trce_dly_min (ns)	trce_dly_max (ns)
GTPRESET_IN	mgtEngine/	System	Edge	Single Rise		Uncheck con	straint – will false pa	th later
GTPRESET_IN	mgtEngine/	System	Edge	Single Rise		Uncheck con	straint – will false pa	th later
GTPRESET_IN	mgtEngine/	System	Edge	Single Rise		Uncheck con:	straint – will false pa	th later
GTPRESET_IN	mgtEngine/	System	Edge	Single Rise		Uncheck con	straint – will false pa	th later
DataIn_pad_0_i[*]	sysClk	System	Edge	Single Rise	1	2	1	1
DataIn_pad_1_i[*]	sysClk	System	Edge	Single Rise	1	2	1	1
LineState_pad_ 0_i[*]	sysClk	System	Edge	Single Rise	1	2	1	1
LineState_pad_ 1_i[*]	sysClk	System	Edge	Single Rise	1	2	1	1
VStatus_pad_0_ i[*]	sysClk	System	Edge	Single Rise	1	2	1	1
VStatus_pad_1_ i[*]	sysClk	System	Edge	Single Rise	1	2	1	1
RxActive_pad_0 _i	sysClk	System	Edge	Single Rise	1	2	1	1
RxActive_pad_1 _i	sysClk	System	Edge	Single Rise	1	2	1	1
Rx_Error_pad_0 _i	sysClk	System	Edge	Single Rise	1	2	1	1
Rx_Error_pad_1 _i	sysClk	System	Edge	Single Rise	1	2	1	1

Table 2: Input Constrain Values



Interface	Clock	Synchronous	Alignment	Data Rate and Edge	tco_min (ns)	tco_max (ns)	trce_dly_min (ns)	trce_dly_max (ns)
Rx_Valid_pad_0_ i	sysClk	System	Edge	Single Rise	1	2	1	1
Rx_Valid_pad_1_ i	sysClk	System	Edge	Single Rise	1	2	1	1
TxReady_pad_0 _i	sysClk	System	Edge	Single Rise	1	2	1	1
TxReady_pad_1 _i	sysClk	System	Edge	Single Rise	1	2	1	1
usb_vbus_paf_0 _i	sysClk	System	Edge	Single Rise	1	2	1	1
usb_vbus_paf_1 _i	sysClk	System	Edge	Single Rise	1	2	1	1
or1200_clmode	VIRTUAL_cpuCl k_5	System	Edge	Single Rise	0.1	2.5	0.1	0.2
or1200_pic_ints	VIRTUAL_cpuCl k_5	System	Edge	Single Rise	0.1	2.5	0.1	0.2
reset	VIRTUAL_cpuCL K_5	System	Edge	Single Rise	0.1	2.5	0.1	0.2

Table 2: Input Constrain Values (cont'd)

The following figure shows the completed input delay page. Note the four constraints being skipped.





ut C	Delays											
	ays describe relative phase ct implementation quality of r			ally board	d clocks) and input	t signals	at the FPGA bound	dary. Inaccu	irate inpi	ut delay values can ma	ke timing fail	
ance	a implementation quality of	counto. more m										
ecor	nmended Constraints											
Q,	100 H II T								Del	lay Parameters		
	Interface	Clock ^1	Synchronous		Alignment		Data Rate and Ed	jae				
	GTPRESET_IN	∭ mgtEngi	-		Edge		Single Rise	-	^	Clock period:	10	ns
	GTPRESET_IN	∭ mgtEngi			Edge		Single Rise		1	tco_min:	1 😔	ns
	GTPRESET_IN	∭ mgtEngi			Edge		Single Rise		1	tco_max:	2 🛞	ns
	GTPRESET_IN	∭ mgtEngi	System		Edge		Single Rise			trce_dly_min:	1 🛛	ns
	DataIn_pad_0_i[*]	III sysClk	System	~	Edge	~	Single Rise	~				
<	DataIn_pad_1_i[*]	∭ sysClk	System	~	Edge	~	Single Rise	~	1	trce_dly_max:	1 🛇	ns
	LineState_pad_0_i[*]	III sysClk	System	~	Edge	~	Single Rise	~				
√	LineState_pad_1_i[*]	III sysClk	System	~	Edge	~	Single Rise	~				
	VStatus_pad_0_i[*]	SysCik	System	~	Edge	~	Single Rise	~	<			⇒
	VStatus_pad_1_i[*]	III sysClk	System	~	Edge	~	Single Rise	~	Ri	ise Max = tco_max + tro	e div max	
	RxActive_pad_0_i	SysClk	System	~	Edge	~	Single Rise	~		ise Min = tco_min + trc		
	RxActive_pad_1_i	∭ sysClk	System	~	Edge	~	Single Rise	~				
	RxError_pad_0_i	∭ sysClk	System	~	Edge	~	Single Rise	~				
	RxError_pad_1_i	M sysClk	System	~	Edge	~	Single Rise	~	~	Appl	у	
					1							
	Command Preview (41)	xisting Set Inp	ut Delay Constr	aints (0)	Waveform - Sy	ystem	Edge Single Rise					
Q,												
et_ir	nput_delay -clock [get_clocks	s {sysClk}] -min	-add_delay 2.0	[get_port	s {Datain_pad_0_	.i[*]}]						
_	nput_delay -clock [get_clocks											
_	nput_delay -clock [get_clocks											
_	nput_delay -clock [get_clocks nput_delay -clock [get_clocks											
_	nput_delay -clock [get_clocks											
_	nput_delay -clock [get_clocks											
et_ir	nput_delay -clock [get_clocks	s {sysClk}] -ma	c-add_delay 3.0	[get_por	ts {LineState_pad_	_1_i[*]}]						

12. When you have successfully entered all the input constraint values, click Next.

The Output Delays page of the wizard displays all the outputs that are unconstrained in the design. The page layout is very similar to the inputs page.

- 13. In the Output Delays page, click the **Clock** header to sort the table alphabetically by clock name.
- 14. Use the following table to constrain all the outputs as you did for the input constraint values. You can select multiple lines in the wizard at once and edit several entries as the same time.

Interface	Clock	Synchronous	Alignment	Data Rate and Edge	tsu (ns)	thd (ns)	trce_dly_max (ns)	trce_dly_min (ns)
OpMode_pad_0_ o[*]	sysClk	System	Setup/Hold	Single Rise	1.0	0.1	0.1	0.1
OpMode_pad_1_ o[*]	sysClk	System	Setup/Hold	Single Rise	1.0	0.1	0.1	0.1
VControl_pad_0_ o[*]	sysClk	System	Setup/Hold	Single Rise	1.0	0.1	0.1	0.1
VControl_pad_0_ o[*]	sysClk	System	Setup/Hold	Single Rise	1.0	0.1	0.1	0.1

Table 3: Output Constraint Values



Interface	Clock	Synchronous	Alignment	Data Rate and Edge	tsu (ns)	thd (ns)	trce_dly_max (ns)	trce_dly_min (ns)
SuspendM_pad_0 _o	sysClk	System	Setup/Hold	Single Rise	1.0	0.1	0.1	0.1
SuspendM_pad_1 _0	sysClk	System	Setup/Hold	Single Rise	1.0	0.1	0.1	0.1
TermSel_pad_0_o	sysClk	System	Setup/Hold	Single Rise	1.0	0.1	0.1	0.1
TermSel_pad_1_o	sysClk	System	Setup/Hold	Single Rise	1.0	0.1	0.1	0.1
TxValid_pad_0_o	sysClk	System	Setup/Hold	Single Rise	1.0	0.1	0.1	0.1
TxValid_pad_1_o	sysClk	System	Setup/Hold	Single Rise	1.0	0.1	0.1	0.1
VControl_Load_p ad_0_o	sysClk	System	Setup/Hold	Single Rise	1.0	0.1	0.1	0.1
VControl_Load_p ad_1_o	sysClk	System	Setup/Hold	Single Rise	1.0	0.1	0.1	0.1
XcvSelect_pad_0_ o	sysClk	System	Setup/Hold	Single Rise	1.0	0.1	0.1	0.1
XcvSelect_pad_1_ o	sysClk	System	Setup/Hold	Single Rise	1.0	0.1	0.1	0.1
phy_rst_pad_0_o	sysClk	System	Setup/Hold	Single Rise	1.0	0.1	0.1	0.1
phy_rst_pad_1_o	sysClk	System	Setup/Hold	Single Rise	1.0	0.1	0.1	0.1
DataOut_pad_0_ o[*]	VIRTUAL_wbClk_4	System	Setup/Hold	Single Rise	2.1	0.6	0.1	0.0
DataOut_pad_1_ o[*]	VIRTUAL_wbClk_4	System	Setup/Hold	Single Rise	2.1	0.6	0.1	0.0
or1200_pm_out[*]	VIRTUAL_wbClk_4	System	Setup/Hold	Single Rise	2.1	0.6	0.1	0.0

Table 3: Output Constraint Values (cont'd)

15. Click **Next** to continue.

The wizard looks for any unconstrained combinational paths through the design. A combinational path is a path that traverses the FPGA without being captured by any sequential elements. This design does not contain any combinational paths.

16. Click **Next** to continue.

Physically exclusive clock groups are clocks that do not exist in the design at the same time. There are no unconstrained physically exclusive clock groups in this design.

17. Click **Next** to continue.

Logically exclusive clocks with no interaction are clocks that are active at the same time except on shared clock tree sections. Then these clocks do not have logical paths between each other and outside the shared sections, they are logically exclusive. There are no unconstrained logically exclusive clock groups with no interaction in the design.

18. Click **Next** to continue.

Logically exclusive clocks with interaction are clocks that are active at the same time except on shared clock tree sections. When these clocks have logical paths between each other, only the clocks limited to the shared clock tree sections are logically exclusive and are therefore constrained differently than the logically exclusive clock with no interaction. There are no unconstrained logically exclusive clock groups with interaction in the design.



19. Click **Next** to continue.

The Asynchronous Clock Domain Crossings page recommends constraints for safe clock domain crossings. This design does not contain any unconstrained clock domain crossings.

20. Click Next to continue.

The following figure shows the final page of the Timing Constraints wizard. All the constraints that were generated by the wizard can be viewed by clicking the links. If you would like to run any reports once the wizard is finished, you can select them using the check boxes in the wizard.

À Timing Constraints Wizard	×
ML Editions	On Finish
£ XILINX.	 View Timing Constraints Create Timing Summary report Create Check Timing report Create Methodology report To keep the new constraints and perform the selected actions, click Finish. The new constraints will automatically be saved to your target XDC file. To discard the constraints, click Cancel.
?	< Back

21. Click Finish to complete the Timing Constraints wizard.

Step 4: Using the Constraints Editor

 Click Edit Timing Constraints from the Flow Navigator under the Synthesized Design section. The Vivado IDE displays the Timing Constraints window.





Project Summar	ry × Device × Timing Constraints	×		? ð Ľ
₹ 	+,	🔸 🕂 📄 🧷 Create Clock		
V Clocks (7)	^	Clock Name	Period (ns)	Rise At (
Create Cl	ock (7)	mgtEngine/ROCKETIO_WRAPPER_TILE_i/gt0_ROCKETIO_WRAPPER_TILE_i/GT0_TXOUTCLK_OUT	12.800	0.
Create Ge	enerated Clock (0)	mgtEngine/ROCKETIO_WRAPPER_TILE_i/gt2_ROCKETIO_WRAPPER_TILE_i/GT2_TXOUTCLK_OUT	12.800	0
Rename /	Auto-Derived Clock (0)	mgtEngine/ROCKETIO_WRAPPER_TILE_i/gt4_ROCKETIO_WRAPPER_TILE_i/GT4_TXOUTCLK_OUT	12.800	0.
Set Clock	Latency (0)	mgtEngine/ROCKETIO_WRAPPER_TILE_i/gt6_ROCKETIO_WRAPPER_TILE_i/GT6_TXOUTCLK_OUT	12.800	0
Set Clock	Uncertainty (0)	sysClk	10.000	0.
Set Clock	Groups (0)	VIRTUAL_cpuClk_5	20.000	0
Set Clock	Sense (0)	VIRTUAL_wbClk_4	20.000	0
Set Input.	Jitter (0)	ick to create a Create Clock constraint		
Set Syster	m Jitter (0)			
Set Extern	al Delay (0)			
V Inputs (38)	Α			
Set Input I	Delay (38)			
 Outputs (38) Set Output 	t Delay (38)			
 Assertions (0) 				
· Assentions (0)	,			
All Constraints				
Q 素 ≑	F T Ø - Ö			
Position	Command			
✓ Constraints				-
✓ L timing.x	dc (C://ivado_Tutorial/project_cpu_netlst/project	t_cpu_netIst.srcs/lab1/new/timing.xdc)		
1	create_clock -period 12.800 -name mgtEngine	/ROCKETIO_WRAPPER_TILE_i/gt0_ROCKETIO_WRAPPER_TILE_i/GT0_TXOUTCLK_OUT -waveform {0.0	00 6.400} [get_	_pins m
2	create_clock -period 12.800 -name mgtEngine	ROCKETIO_WRAPPER_TILE_i/gt2_ROCKETIO_WRAPPER_TILE_i/GT2_TXOUTCLK_OUT -waveform {0.0	00 6.400} [get	_pins m
3	create_clock -period 12.800 -name mgtEngine	/ROCKETIO_WRAPPER_TILE_i/gt4_ROCKETIO_WRAPPER_TILE_i/GT4_TXOUTCLK_OUT -waveform {0.0	00 6.400} [get_	_pins m
4	create_clock -period 12.800 -name mgtEngine	ROCKETIO_WRAPPER_TILE_i/gt6_ROCKETIO_WRAPPER_TILE_i/GT6_TXOUTCLK_OUT -waveform {0.0	00 6.400} [get	pins m
5	create_clock -period 10.000 -name sysClk -wa	aveform {0.000 5.000} [get_ports sysClk]		
6	create_clock -period 20.000 -name VIRTUAL_	cpuClk_5 -waveform {0.000 10.000}		
7	set_input_delay -clock [get_clocks sysClk] -mi	n -add_delay 2.0 [get_ports {DataIn_pad_0_i[*]}]		~
<		Apply Cancel		>

There are three sections to the Timing Constraints window, as shown in the above figure:

- **Constraints tree view:** Labeled as section A. This section displays standard timing constraints, grouped by category. Double-clicking a constraint in this section opens a form to help you define the selected constraint.
- **Constraints Spreadsheet:** Labeled as section B. This section displays timing constraints of the type currently selected in the Constraints tree view. If you prefer, you can use this to directly define or edit constraints instead of using the Constraints wizard.
- All Constraints: Labeled as section C. This section displays all the timing constraints that currently exist in the design.

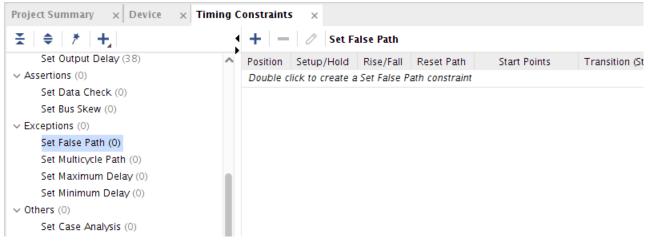
The Timing Constraints wizard identifies missing clocks, I/O delays, and clock domain crossings exceptions, but it does not handle general timing exceptions. You are going to use the timing constraints editor to create the exceptions that exist in this design.

First, you are going to set a false path on the GTPRESET_IN input. This is the input constraint that you skipped by unchecking it on the input delay page of the Timing Constraints wizard. The GTPRESET_IN signal is an asynchronous reset that is properly synchronized inside the design. For more information about false paths, see this link in the *Vivado Design Suite User Guide: Using Constraints* (UG903).

2. In the Constraints tree view, scroll down to the Exceptions category, shown in the following figure.



3. Double-click Set False Path.



4. To the right of the From text box, click the **Choose Start Points** button \square

The Specify Start Points dialog box appears.

- 5. From the Find names of type drop-down list, select **I/O Port** as shown in the following figure.
- 6. Under Options, select "Name", "contains" and in the search pattern text box, enter GTPRESET_IN.
- 7. Click the **Find** button.
- 8. Select **GTPRESET_IN** in the found results text box and press the right arrow to move it to the selected names text box.

TIP: You can also double-click GTPRESET_IN to move it from the Find results list to the Selected names list.

Notice that the Command field displayed at the bottom of the dialog box changes as you perform these different actions. The get_ports command changes to:

get_ports "*GTPRESET_IN*"



À Specify Start Points		×
Specify a list of path endpoints or clocks.		Å
Fin <u>d</u> names of type: VO Port 🗸		
NAME ~ contains	✓ GTPRESET_IN	⊗ +
Regular expression Ignore case Of Objects:	Eind	,
Found: 0	Selected: 1	Å₽
Use the buttons on the right to move items to this list.	GTPRESET_IN	↑ ↓ ↑ ↓
<u>Command:</u> get_ports "*GTPRESET_IN*"	Se <u>t</u> A	© Enumer <u>a</u> te

9. Click **Set** in the Specify Start Points dialog box.

The completed Set False Path dialog box is shown in the following figure. Notice the following command text in the Command field at the bottom of the dialog box:

set_false_path -from [get_ports "*GTPRESET_IN*"]

The Vivado IDE displays the Tcl command form of all constraints created via the dialogs for your review. This is useful for learning the Tcl command syntax, and for verifying the final constraint before adding it.



🝌 Set False	e Path	×
Define false p	paths in the design that are not considered during timing a	analysis.
Taracta	Ortions	
T <u>a</u> rgets	Options	
Start Poin	ts	
<u>F</u> rom:	[get_ports "*GTPRESET_IN*"] 💿 🗤 Tra	ansition rise/fall 🗸
Through P	Points	
Th <u>r</u> ou	gh: Transitio	on rise/fall 😽 🕂
End Points	S	
<u>T</u> o:	•••• Tra	ansition rise/fall 🗸
Command:	set_false_path -from [get_ports "*GTPRESET_IN*"]	
?	Reference Reset to Defaults	OK Cancel

10. Click **OK** to close the Timing Constraint Editor.

Vivado creates the false path exception.

Add a MultiCycle Path

Next, you will add a multicycle path using the constraints editor.

- 1. Double-click **Set Multicycle Path** under the Exceptions category of the tree.
- 2. In the Set Multicycle Path dialog box, set the path multiplier to 2.
- 3. In the Through entry box, type the following string (alternately, you can copy and paste it from here):

```
[get_pins cpuEngine/or1200_cpu/or1200_alu/*]
```



Notice that the Tcl command displays in the Command field.

4. Click OK.

A new multicycle path is added to the constraints editor in the <unsaved_constraints> section, as shown in the following figure.

Project Summar	ny × Device × Timing	Constraints	×							? 🗗 🖸
₹ \$ *	+,	+ -	🖉 Set Mul	ticycle Path						
✓ Assertions (0) Set Data C	heck (0)	Position 85 Double c	Path Multiplier 2 lick to create a S			Start/End	Reset Path	Start Points	Transition (Start)	Through Poir -through [get_pir
Set Bus Sk V Exceptions (2)										
Set False F										
	/cle Path (1)									
Set Maxim	um Delay (0)									
Set Minim	um Delay (0)									
V Others (0)										
Set Case A	Analysis (0)									
Group Pat										
Set Disabl	e Timing (0) 🗸 🗸	<								>
All Constraint	s									
Q ₹ ♦	f T/ 🖉 — Ö									
Position	Command									
75	set_output_delay -clock [get_cl	ocks sysClk]	-max -add_del	ay 1.1 [get_po	rts VControl	_Load_pad_1	1_0]			^
76	set_output_delay -clock [get_cl	ocks sysClk]	-min -add_dela	ay 0.0 [get_por	ts XcvSelect	_pad_0_0]				
77	set_output_delay -clock [get_cl	ocks sysClk]	-max -add_del	ay 1.1 [get_po	rts XcvSelec	t_pad_0_o]				
78	set_output_delay -clock [get_cl	ocks sysClk]	-min -add_dela	ay 0.0 [get_por	ts XcvSelect	_pad_1_0]				
79	set_output_delay -clock [get_cl	ocks sysClk]	-max -add_del	ay 1.1 [get_po	rts XcvSelec	t_pad_1_0]				
80	set_output_delay -clock [get_cl	ocks sysClk]	-min -add_dela	ay 0.0 [get_por	ts phy_rst_p	ad_0_0]				
81	set_output_delay -clock [get_cl	ocks sysClk]	-max -add_del	ay 1.1 [get_po	rts phy_rst_	pad_0_0]				
82	set_output_delay -clock [get_cl									
83	set_output_delay -clock [get_cl	ocks sysClk]	-max -add_del	ay 1.1 [get_po	rts phy_rst_	pad_1_0]				
✓ □ <unsaved< p=""></unsaved<>										
84	set_false_path -from [get_ports									
85	set_multicycle_path -through [g	jet_pins cpu	Engine/or1200_	cpu/or1200_a	lu/*] 2					~
<				Apply	Cance	1				>

Adding a multicycle path by default pushes the setup timing to the specified number of cycles (N), *but it also pushes the hold timing to* N - 1 *cycles*. This is usually not what is intended and could cause Vivado tools to spend a lot of time fixing large hold violations. In this case you want the setup path clock. To achieve this, you need to define another multicycle path on the hold edge to 1, such that N - 1 is zero. For more information about this situation, see this link in the Vivado Design Suite User Guide: Using Constraints (UG903).

5. Double-click **Set Multicycle Path** under the Exceptions category of the tree for a second time.

Note that all the fields you entered previously are still filled in.

- 6. In the Set Multicycle Path dialog box, change the Path Multiplier to **1**.
- 7. Select the **Options** tab.
- 8. Under Setup/Hold, select the check box that says **Use path multiplier**.
- 9. Use the pull-down to select hold (minimum delay).



A Set Multicycle Path	×
Define multicycle path. This command allows specifying the total number of clock cycle required for propagation of a signal from its origin to destination when that propagation is longer than a single clock cycle.	4
Specify path <u>m</u> ultiplier:	
Targets Options	
Setup/Hold	
✓ Use path multiplier for hold (minimum delay) ✓ calculation	
Rise/Fall	
Use path multiplier for endpoint rising v delays	
Start/End	
☐ Mu <u>I</u> ticycle information is relative to the startpoint ✓ clock	
Remove existing path exceptions before setting multicycle path	
Command: set_multicycle_path -hold -through [get_pins cpuEngine/or1200_cpu/or1200_alu/*] 1	
Reference Reset to Defaults OK Ca	incel

10. Click OK.

Now you have a fully constrained design in memory. To save the constraints to disk, proceed to Step 5: Saving Constraints.

Step 5: Saving Constraints

Constraint management is an important step of the design flow, and the Vivado Design Suite provides you the flexibility of adding new constraints into an existing constraint file, overwriting existing constraints, or creating a new constraints file to track design changes or complete missing constraints.

You have created a few timing exceptions for the design, but the exceptions exist only in memory and not on disk yet. You need to save the exceptions to the timing.xdc file.

1. From the Sources window in Vivado, double-click timing.xdc under Constraints \rightarrow lab1.



2. Scroll to the bottom of the file and notice that the set_false_path and set_multicycle_path constraints do not exist in the file. This is also reflected in the Timing Constraints Editor as <unsaved_constraints>, as shown in the following figure.

All Constrain	s		
Q 🕺	►		
Position	Command	Scoped Cell	
79	set_output_delay -clock [get_clocks sysClk] -max -add_delay 1.1 [get_ports XcvSelect_pad_1_o]		
80	set_output_delay -clock [get_clocks sysClk] -min -add_delay 0.0 [get_ports phy_rst_pad_0_o]		
81	set_output_delay -clock [get_clocks sysClk] -max -add_delay 1.1 [get_ports phy_rst_pad_0_o]		
82	set_output_delay -clock [get_clocks sysClk] -min -add_delay 0.0 [get_ports phy_rst_pad_1_o]		
83	set_output_delay -clock [get_clocks sysClk] -max -add_delay 1.1 [get_ports phy_rst_pad_1_o]		
🗸 🗁 <unsave< td=""><td>d constraints></td><td></td><td></td></unsave<>	d constraints>		
84	set_false_path -from [get_ports *GTPRESET_IN*]		
85	set_multicycle_path -through [get_pins cpuEngine/or1200_cpu/or1200_alu/*] 2		
86	set_multicycle_path -hold -through [get_pins cpuEngine/or1200_cpu/or1200_alu/*] 1		
	Apply Cancel		

- 3. Click the **Save Constraints** button [■] or use the **File** → **Constraints** → **Save As** command from the main menu.
- 4. Click the **Reload** link in the banner of the timing.xdc tab to reload the constraints file from disk. Notice that the false path and multi-cycle paths are now visible in the timing.xdc text file, as shown in the following figure.

Proj	ect Summary × Device × timing.xdc ×	? 🗆 🖒
C:/V	ïvado_Tutorial/project_cpu_netlst/project_cpu_netlst.srcs/lab1/new/timing.xdc	×
Q,		۰
64	set_output_delay -clock [get_clocks sysClk] -min -add_delay 0.000 [get_ports TermSel_pad_0_o]	^
65	set_output_delay -clock [get_clocks sysClk] -max -add_delay 1.100 [get_ports TermSel_pad_0_o]	
66	set_output_delay -clock [get_clocks sysClk] -min -add_delay 0.000 [get_ports TermSel_pad_1_0]	
67	set_output_delay -clock [get_clocks sysClk] -max -add_delay 1.100 [get_ports TermSel_pad_1_0]	
68	set_output_delay -clock [get_clocks sysClk] -min -add_delay 0.000 [get_ports TxValid_pad_0_0]	
69	set_output_delay -clock [get_clocks sysClk] -max -add_delay 1.100 [get_ports TxValid_pad_0_0]	
70	set_output_delay -clock [get_clocks sysClk] -min -add_delay 0.000 [get_ports TxValid_pad_1_0]	
71	set_output_delay -clock [get_clocks sysClk] -max -add_delay 1.100 [get_ports TxValid_pad_1_0]	
72	set_output_delay -clock [get_clocks sysClk] -min -add_delay 0.000 [get_ports VControl_Load_pad_0_o	1
73	set_output_delay -clock [get_clocks sysClk] -max -add_delay 1.100 [get_ports VControl_Load_pad_0_o	1
74	set_output_delay -clock [get_clocks sysClk] -min -add_delay 0.000 [get_ports VControl_Load_pad_1_o	1
75	set_output_delay -clock [get_clocks sysClk] -max -add_delay 1.100 [get_ports VControl_Load_pad_1_o	1
76	set_output_delay -clock [get_clocks sysClk] -min -add_delay 0.000 [get_ports XcvSelect_pad_0_0]	
77	set_output_delay -clock [get_clocks sysClk] -max -add_delay 1.100 [get_ports XcvSelect_pad_0_0]	
78	set_output_delay -clock [get_clocks sysClk] -min -add_delay 0.000 [get_ports XcvSelect_pad_1_0]	
79	set_output_delay -clock [get_clocks sysClk] -max -add_delay 1.100 [get_ports XcvSelect_pad_1_0]	
80	set_output_delay -clock [get_clocks sysClk] -min -add_delay 0.000 [get_ports phy_rst_pad_0_0]	
81	set_output_delay -clock [get_clocks sysClk] -max -add_delay 1.100 [get_ports phy_rst_pad_0_o]	
82	<pre>set_output_delay -clock [get_clocks sysClk] -min -add_delay 0.000 [get_ports phy_rst_pad_1_0]</pre>	
83	<pre>set_output_delay -clock [get_clocks sysClk] -max -add_delay 1.100 [get_ports phy_rst_pad_1_0]</pre>	
84	<pre>set_false_path -from [get_ports *GTPRESET*]</pre>	
85	<pre>set _xlnx_shared_i0 [get_pins cpuEngine/or1200_cpu/or1200_alu/*]</pre>	
86	<pre>set_multicycle_path -through \$_xlnx_shared_i0 2</pre>	
87	<pre>set_multicycle_path -hold -through \$_xlnx_shared_i0 1</pre>	
88		~
		>

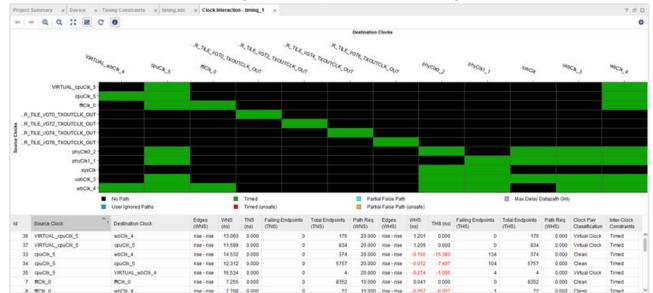
Note: The Tcl variable _xlnx_shared_i0 is automatically inferred by Vivado to share the same collection of objects between multiple timing constraints.



Step 6: Clock Interaction Report

After or during constraints creation, you must verify that the constraints are complete and safe. Vivado Design Suite times all clocks together by default unless you specify otherwise by defining clock groups or other timing exceptions. The set_clock_groups command specifies asynchronous or exclusive clock domains and disables timing analysis between them. You can also use the set_false_path exception between two clocks to disable timing on all paths between them, or use it on specific netlist objects to only disable some paths. The set_multicycle_path exception modifies the clock edges used during timing analysis instead of the default single cycle assumption. For more information on using these constraints, see the Vivado Design Suite User Guide: Using Constraints (UG903).

Vivado automatically infers timing path requirements for paths that cross between two different clock domains, called inter-clock paths, making assumptions regarding phase and offset. The Report Clock Interaction command reports inter-clock paths, to help identify potential problems such as unrealistic setup or hold requirements between two clocks, or unsafe timing between asynchronous clocks (no known phase relationship) which can lead to unstable hardware behavior. For more information on the Clock Interaction Report, see the "Details of the Clock Interaction Report" in *Vivado Design Suite User Guide: Design Analysis and Closure Techniques* (UG906).



1. From the Flow Navigator, select **Synthesized Design** → **Report Clock Interaction** and click **OK** in the Report Clock Interaction dialog box to accept the default settings.



The Vivado IDE generates a graphical matrix illustrating the relationship of the various clocks in the design, as shown in the previous figure. For this design, the primary clock (sysClk) connects to an MMCM, which generates six additional clocks. The clock interactions shown are between these generated clocks. In addition, the Timing Constraints wizard created additional generated clocks and virtual clocks to fully constrain the design.

In general, the Clock Interaction report shows clock pairs with no path between them (black), with paths safely timed (green and light blue), with paths not safely timed (red and orange) and with paths covered by Max Delay Datapath Only constraints. In this design, only black cells and green cells are displayed in the matrix.

IMPORTANT! Green in the matrix does not mean that timing is met, it simply means that the timing constraints and the clock tree topologies allows safe timing analysis and accurate slack computation.

In the Clock Interaction report, unsafe means there is no common primary clock (no known phase relationship), or no common node (uncommon scenario that results in unknown phase relationship), or no common clock period within the first 1000 clock cycles of the source and destination clocks. The Vivado timing engine selects edges on the launch and capture clocks based on the first 1000 cycles, but these edges might not reflect the most pessimistic analysis between the clocks.



TIP: The colors described here are the default colors. Your colors might be configured differently from those shown in the previous figure.

2. Close the Clock Interaction window by clicking the **Close** button \times in the window tab.

Step 7: Timing Summary Report

Timing paths start and end at clocked elements. Input and Output ports are not sequential elements, and by default Vivado timing analysis does not time paths to or from I/O ports in the design, unless input/output delay constraints are specified.

In this step you will generate and interpret timing reports in Vivado.

- 1. Select **Reports → Timing → Report Timing Summary**.
- 2. Click **OK** to generate the report, using the default options.

The Timing Summary tab opens, as shown in the following figure.

Tcl Console Messages Log	Reports	Design Runs Timing ×					? _ 🗆
Q 꽃 ♦ C 🕍 🤑	1	Design Timing Summary					
General Information Timer Settings	î	Setup		Hold		Pulse Width	
Design Timing Summary		Worst Negative Slack (WNS):	2.250 ns	Worst Hold Slack (WHS):	-0.274 ns	Worst Pulse Width Slack (WPWS):	3.000 ns
Clock Summary (14)		Total Negative Slack (TNS):	0.000 ns	Total Hold Slack (THS):	-372.506 ns	Total Pulse Width Negative Slack (TPWS):	0.000 ns
> 🐤 Check Timing (9)		Number of Failing Endpoints:	0	Number of Failing Endpoints:	6476	Number of Failing Endpoints:	0
> 🕞 Intra-Clock Paths		Total Number of Endpoints:	46377	Total Number of Endpoints:	46377	Total Number of Endpoints:	16037
> 🚡 Inter-Clock Paths > 🚍 Other Path Groups		Timing constraints are not met.					
> 🚍 User Ignored Paths	~						
Timing Summary - timing_1							



The design passes setup timing but fails hold analysis. Before implementing the design, timing analysis uses estimated net delays that represent ideal placement. Small hold violations are common at this point of the flow and will be fixed during the routing step. For now, review the content of the report.

3. Click the **Worst Negative Slack** link in the design timing summary section to see the worst timing path in the design, as shown in the following figure.

N ¥ ≑ C 🖬 9	1	Q - 5	1 🔬 M		inter-Clo	ck Paths - ust	CIK_3 to cpuCIK_5 - Setup					
> > phyClk0_2 to wbClk_4	^	Name	Slack ^1	Levels	Routes	High Fanout	From	То	Total Delay	Logic Delay	Net Delay	Requirement
> 🚍 phyClk1_1 to cpuClk_5		👍 Path 473	2.250	17	18	165	usbEngine1/u4/inta_reg/C	cpuEngine/or1u_err_o_reg/D	7.406	1.583	5.823	10.0
> 🚍 phyClk1_1 to sysClk		👍 Path 474	2.250	17	18	165	usbEngine1/u4/inta_reg/C	cpuEngine/or1g_o_reg[0]/D	7.406	1.583	5.823	10.0
> 📬 phyClk1_1 to usbClk_3		👍 Path 475	2.250	17	18	165	usbEngine1/u4/inta_reg/C	cpuEngine/or1g_o_reg[1]/D	7.406	1.583	5.823	10.0
> phyClk1_1 to wbClk_4		👍 Path 476	2.250	17	18	165	usbEngine1/u4/inta_reg/C	cpuEngine/or1g_o_reg[3]/D	7.406	1.583	5.823	10.0
> 🖴 sysClk to phyClk0_2		👍 Path 477	2.257	17	18	165	usbEngine1/u4/inta_reg/C	cpuEngine/or1g_o_reg[2]/D	7.399	1.583	5.816	10.0
> 🖴 sysClk to phyClk1_1		👍 Path 478	2.257	17	18	165	usbEngine1/u4/inta_reg/C	cpuEngine/or1cstb_o_reg/D	7.399	1.583	5.816	10.0
w lash clk_3 to cpuClk_5		1, Path 479	2.581	16	17	165	usbEngine1/u4/inta_reg/C	cpuEngine/or1r_o_reg[13]/D	7.075	1.540	5.535	10.0
Setup 2.250 ns (10)	~	<	:						-			>

4. When the worst path is selected, press the **F4** key to bring up its schematic. The following figure shows the worst setup path in the design.

Constraints x timing_xdc x Clock Interaction - timing_1 x Schematic x Ib Ib Ib Ib Ib Ib Ib Ib Ib Ib	? 🗆 🖸
	*

- 5. In the timing summary tree, select **Check Timing**.
 - There are nine issues flagged by Check Timing shown in the following figure.
 - Eight of these are pulse_width_clock checks, which were also flagged by the Timing Constraints wizard, but were not constrained. These violations have low severity because the corresponding missing clocks are not needed for timing logic paths.
 - The remaining issue flagged by Check Timing is a no_input_delay check, which is due to a missing input constraint on the reset signal that was set to false_path. This can also be ignored in this example.



Q 🛛 🛣 🛛 🜲 🛛 C 🛛 🔛 🕘 🛛	Q Check Timing						
Check Timing (9)	Timing Check	Count v1	Worst Severity				
no_clock (0)	pulse_width_clock	8	Low				
constant_clock (0)	no_input_delay	1	😣 Medium				
pulse_width_clock (8)	no_clock	0					
unconstrained_internal_endpoints (0)	constant_clock	0					
• no_input_delay (1)	unconstrained_internal_endpoints	0					
no_output_delay (0)	no_output_delay	0					
multiple_clock (0)	multiple_clock	0					
generated_clocks (0)	generated_clocks	0					
loops (0)	loops	0					
partial_input_delay (0)	partial_input_delay	0					
partial_output_delay (0)	partial_output_delay	0					
latch_loops (0)	latch_loops	0					
lintra-Clock Paths							

6. In the timing summary tree, select the Clock Summary, as shown in the following figure.

The clock summary section of the timing summary report lists all the clocks in the design and shows the resulting frequencies and waveforms of each clock. The hierarchy shows the relationship between the generated clocks and the primary clock (for example, cpuClk_5 vs. sysClk). For example, it shows that cpuClk_5 is generated from primary clock SysClk, and its period is twice that of sysClk.

Tcl Console Messages Log Reports	Design Runs Timing ×			?	_ 0
Q, X ♦ C 💾 🟮	Q				
General Information	Name	Waveform	Period (ns)	Frequency (MHz)	
Timer Settings	VIRTUAL_cpuClk_5	{0.000 10.000}	20.000	50.000	
Design Timing Summary	VIRTUAL_wbClk_4	{0.000 10.000}	20.000	50.000	
Clock Summary (14)	mgtEngine/ROCKETIO_WRAPPER_TILE_I/gt0_ROCKETIO_WRAPPER_TILE_I/GT0_TXOUTCLK_OUT	{0.000 6.400}	12.800	78.125	
Check Timing (9)	mgtEngine/ROCKETIO_WRAPPER_TILE_I/gt2_ROCKETIO_WRAPPER_TILE_I/GT2_TXOUTCLK_OUT	{0.000 6.400}	12.800	78.125	
Intra-Clock Paths	mgtEngine/ROCKETIO_WRAPPER_TILE_i/gt4_ROCKETIO_WRAPPER_TILE_i/GT4_TXOUTCLK_OUT	{0.000 6.400}	12.800	78.125	
Inter-Clock Paths	mgtEngine/ROCKETIO_WRAPPER_TILE_i/gt6_ROCKETIO_WRAPPER_TILE_i/GT6_TXOUTCLK_OUT	{0.000 6.400}	12.800	78.125	
> Coucik_5 to VIRTUAL_wbClk_4	✓ sysClk	{0.000 5.000}	10.000	100.000	
wbClk_4 to VIRTUAL_wbClk_4	cikfbout	{0.000 5.000}	10.000	100.000	
phyClk0_2 to sysClk	cpuClk_5	{0.000 10.000}	20.000	50.000	
Description of the sysClk is a second sec	fficik_0	{0.000 5.000}	10.000	100.000	
> a usbClk_3 to sysClk	phyClk0_2	{0.000 5.000}	10.000	100.000	
> VIRTUAL_cpuClk_5 to cpuClk_5	phyClk1_1	{0.000 5.000}	10.000	100.000	
> Image: First of the second secon	usbClk_3	{0.000 5.000}	10.000	100.000	
Timing Summary - timing_1 ×	Y				

The remaining sections of the timing summary report group paths by their type. Each section lists the top ten paths (specified when the report was generated) in that group. These include inter-clock paths, intra-clock paths, other path groups, user ignored paths, and unconstrained paths. Clicking the roots will show a summary of the paths beneath. Expanding the tree further will ultimately display the top timing paths for each group.



Conclusion

At this point you can either continue to Lab #2: Setting Physical Constraints, or exit the Vivado Design Suite and continue later. Lab #2 uses the project created during this lab, so be sure not to delete your work.

In this lab, you learned:

- How to Create a constraint set and set a target constraint file
- How to add timing constraints to a design using the Timing Constraints Wizard
- How to add timing exceptions using the Timing Constraints Editor
- The importance of saving constraints to disk versus in-memory constraints
- How to generate the clock interaction report and properly interpret the resulting matrix
- How to generate the timing summary report and properly interpret the results





Lab 2

Setting Physical Constraints

In this lab, you will create physical constraints for the CPU Netlist design, observing how actions in the GUI translate into Tcl commands. Using Tcl commands, complex operations are easily scripted for repeated use, at various stages of the flow.

Note: If you are continuing from Lab 1, and your design is open, skip ahead to Step 2: Adding Placement Constraints.

Step 1: Opening the Project

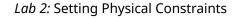
This lab continues from the end of Lab #1 in this tutorial. You must complete Lab #1 prior to beginning Lab #2. If you closed the tool, or closed the tutorial project at the end of Lab #1, you will need to open them again.

- 1. Start by loading the Vivado[®] Integrated Design Environment (IDE) by doing one of the following
 - Launch Vivado IDE from the icon on the Windows desktop.
 - Select Windows \rightarrow Programs \rightarrow Xilinx Design Tools \rightarrow Vivado 2021.x \rightarrow Vivado 2021.x.
 - Type vivado from a command terminal.

From the Getting Started screen you can open recent projects.

2. Under Recent Projects, click project_cpu_netlist as shown in the following figure.







	Recent Projects
uick Start	project_cpu_netlist C:/Vivado_Tutorial/project_cpu_netlist
ate Project >	
en Project >	
en Example Project >	
asks	
nage IP >	
en Hardware Manager >	
ado Store >	

Step 2: Adding Placement Constraints

Explore some of the design hierarchy, and begin placing logic elements to create physical constraints.

1. From the Flow Navigator, select **Open Synthesized Design**. The synthesized design might already be open if you came directly to this step.

The synthesized netlist opens with the Device window displayed.

- 2. Select the **Netlist** window and expand the **clkgen** hierarchy.
- 3. Expand the Leaf Cells folder and select the instance mmcm_adv_inst (MMCME2_ADV).





Sources	Netlist ×	? _ 0 6
¥		0
🕅 top		^
> 🗎 Nets	(4432)	
> 🗎 Leaf	Cells (223)	
~ 🔳 clkge	n (clock_generator)	
> 🖹 N	ets (18)	
~ 🗎 L	eaf Cells (10)	
	clkf_buf (BUFG)	
	clkout1_buf (BUFG)	
	clkout2_buf (BUFG)	
	clkout3_buf (BUFG)	
	clkout4_buf (BUFG)	
	clkout5_buf (BUFG)	
	clkout6_buf (BUFG)	
•		
	VCC (VCC)	
	ngine (or1200_top)	
> 🔋 fftEng	ine (fftTop)	
> 🔳 matE	naine (matTop)	~

- 4. Look in the Cell Properties view, under the Properties tab, and notice that the STATUS is UNPLACED, and there are no IS_LOC_FIXED or IS_BEL_FIXED properties shown.
- 5. Check this in the Tcl Console by typing:

get_property IS_LOC_FIXED [get_cells clkgen/mmcm_adv_inst]

This returns a zero, indicating the object is not fixed to a location.

6. Zoom into the bottom right of the Device view, to display the lower half of Clock Region X1YO, to prepare for placing the selected object. See the following figure.

TIP: It is easier to place logic in the Device window if Routing Resources are not displayed. If they are displayed, select the **Routing Resources** toolbar button **to** disable.



Sources Netlist ×	? _ 🗆 🖸	Project Sun	nmary >	Device	×						? [
X H	0	← →	@ , 6	20	E () H	Po	ie.				
clkout1_buf	(BUFG)											
clkout2_buf	(BUFG)							X1Y1		الككك		
clkout3_buf	(BUFG)											
clkout4_buf	(BUFG)											
clkout5_buf	(BUFG)											
clkout6_buf												
GND (GND)												
mmcm_adv	_inst (MMCME2_ADV)									الالككم		
VCC (VCC)										الالككم		
cpuEngine (or1200_	_top)											
fftEngine (fftTop)										الككك		
mgtEngine (mgtTop										الككك ا		
ushEngine0 (ushf t	ion) 🗸											
Cell Properties	? _ 🗆 🖒 ×									الككك		
mmcm_adv_inst	$\leftarrow \Rightarrow \diamond$											
Q 🗶 🌲 🖷	+ - 0 2+											
FILE_NAME	C:/Vivado_Tutorial/project_cpu_netlist/pro ^											
IS_BLACKBOX												
IS_DEBUGGABLE	×											
IS_ORIG_CELL												
IS_PRIMITIVE	~							X4X0				
								X1Y0				

- 7. In the Netlist window, click on the **mmcm_adv_inst** and drag it into the Device window to place it into the bottom right MMCME2_ADV.
- 8. Look in the Tcl Console. You should see something like these three commands:

```
startgroup
place_cell clkgen/mmcm_adv_inst MMCME2_ADV_X1Y0/MMCME2_ADV
endgroup
```

The startgroup and endgroup Tcl commands bracket sequences of commands to support the undo function in the Vivado tools. If you make a mistake, you can use the undo command in the Tcl Console, or the **Edit** \rightarrow **Undo** command. This will undo the placement and allow you to redo it. For more information on startgroup, endgroup, and undo, refer to the Vivado Design Suite Tcl Command Reference Guide (UG835).

9. Look at the Properties tab of the Cell Properties window for the MMCM cell you placed.

Notice that the IS_BEL_FIXED and IS_LOC_FIXED properties are now checked, reflecting that the object has been placed, as shown in the following figure. The STATUS property is set to FIXED as well.

Note: The Cell Drag and Drop mode in the Device window determines whether only IS LOC FIXED is set, or IS BEL FIXED is also set, when placing objects. Refer to this link in the Vivado Design Suite User Guide: Using the Vivado IDE (UG893) for more information on the Device window.



Cell Properties	? _ 🗆 🖒 X
mmcm_adv_inst	⇔ ⇒
Q ★ ♦ •€ + -	A Z
DIVCLK_DIVIDE	10 ^
FILE_NAME	C://ivado_Tutorial/project_cpu_ne
IS_BEL_FIXED	\checkmark
IS_BLACKBOX	
IS_BOUNDARY_INST	
IS_DEBUGGABLE	× 1
IS_LOC_FIXED	
IS_MATCHED	~
IS_ORIG_CELL	~
IS_PRIMITIVE	· ·
General Properties Nets	Cell Pins

The IS_BEL_FIXED and IS_LOC_FIXED properties on the object are physical constraints reflecting the placement of the object. These constraint are used by Vivado implementation, and will not be changed by the tool. However, if the properties are invalid, they will cause errors downstream in the design flow.

Notice that when you place mmcm_adv_inst in the Device window, the Save Constraints

button is enabled. The physical constraints are added to the Vivado tool in-memory design, but are not yet saved to the target constraint file.

Step 3: Defining Additional Physical Constraints

In this step, you will define additional physical constraints to the design, such as the PACKAGE_PIN, and PROHIBIT constraints.

1. Select Layout → I/O Planning to open the I/O Planning view layout from the Layout Selector in the tool bar menu.

The I/O Planning view layout displays the Package window, as well as the I/O Ports and Package Pins windows, to facilitate planning the I/O port assignment for the design.

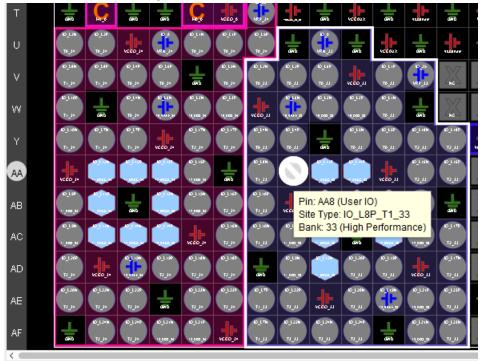
For the purposes of this tutorial, assume the PCB layout has been completed, and therefore certain pins are not accessible on the FPGA package. You can prohibit the Vivado tool from using these pins during placement and routing (assuming you have not already specified all of your I/O assignments).

2. Select the **AA8** pin in the Package window.



TIP: Use the X and Y-axis values on the edge of the Package window to help you locate this pin on the package. You may need to zoom in or enlarge the Package window to make the values visible.

3. With the pin selected, right-click and select Set Prohibit.



When you unselect the pin, you will notice the site now has a red circle with a diagonal line through it, indicating that it is unusable.

4. Look at the Tcl Console and review the Tcl command produced by the Vivado IDE.

```
startgroup
set_property prohibit 1 [get_bels IOB_X1Y34/PAD]
set_property prohibit 1 [get_sites AA8]
endgroup
```

Step 4: Defining Constraints with Object Properties

You can create timing and placement constraints as you have seen in this tutorial. You can also change the properties of cells to control how they are handled by Vivado implementation. Many physical constraints are defined as properties on a cell object.

For example, if you discover a timing issue with a RAM in the design, to avoid resynthesis, you can change a property of the RAM cell to add in pipeline registers. After confirming with the designer and validation teams that this is an acceptable approach, you can change the design.





Setting Cell Properties

Because it can be too time consuming and costly to go back to the RTL after synthesis, you can make changes in the netlist as follows.

- 1. Select $Edit \rightarrow Find$ to open the Find dialog box, as shown in the following figure.
 - a. Select Cells from the Find drop-down list.
 - b. Under Properties, set PRIMITIVE_TYPE to BMEM.BRAM.
 - c. Make sure that Search Hierarchy is selected, as shown in the following figure.
 - d. Click OK.

nd objects by	filtering Tel properties and objects	
nd objects by	filtering Tcl properties and objects.	
R <u>e</u> sult name:	find_1 😒	
<u>F</u> ind:	Cells V	
Properties		
PRIMITIVE	E_TYPE V is V BMEM.BRAM	~ +
Regular ex	varession 🦳 Janare case 🔽 Search hierarchically	
<u>R</u> egular ex	xpression 🗌 Ignore case 🗹 Search hierarchically	
_	xpression 🗌 Ignore case 🗹 Search hierarchically	•
Of o <u>b</u> jects:		
Of o <u>bj</u> ects:	xpression Ignore case Search hierarchically show_objects -name find_1 [get_cells -hierarchical -filter { PRIMITIVE_TYPE =~ BMEM.t	
Of o <u>b</u> jects: C ommand: s	show_objects -name find_1 [get_cells -hierarchical -filter { PRIMITIVE_TYPE =~ BMEM.t	
Of o <u>b</u> jects:	show_objects -name find_1 [get_cells -hierarchical -filter { PRIMITIVE_TYPE =~ BMEM.t	

The Find Results window opens.

- 2. Select the **Show Search** button \bigcirc on the toolbar menu of the Find Results window.
- 3. Search for ingressLoop, and select the following cell: fftEngine/fftInst/ ingressLoop[7].ingressFifo/

In the Properties tab of the Cell Properties window, you can see the DOA_REG and DOB_REG are set to zero, indicating that the output registers are disabled.



4. Generate a custom timing report from this cell directly from the Tcl Console. The Tcl command to enter is:

```
report_timing -from [get_cells fftEngine/fftInst/
ingressLoop[7].ingressFifo/
buffer_fifo/infer_fifo.block_ram_performance.fifo_ram_reg]
```

TIP: You can copy and paste the cell name from the General tab of the Cell Properties window into the Tcl Console.

- 5. In the upper-right corner of the Tcl Console, click the **Maximize** button \Box to maximize the window and better view the timing report.
- 6. In the data path section of the report, 1.800 ns is added by this RAMB.

```
RAMB36E1 (Prop_ramb36e1_CLKBWRCLK_D0BD0[16])
1.800 -0.526 r fftEngine/fftInst/ingressLoop[7].ingressFifo/buffer_fifo/infer_fifo.block_ram_performance.
net (fo-2, unplaced) 0.466 -0.061 fftInst/toBft[15]_55[0]
DSP48E1 r transformLoop[7].ct/x0utReg_reg/A[0]
```

- 7. Restore the Tcl Console to its normal size.
- 8. In the Properties tab of the Cell Properties window, select the DOA_REG and DOB_REG properties for this cell and change their values for "0" to "1."

You can see two set_property commands run in the Tcl Console.

```
set_property DOA_REG {1} [get_cells {fftEngine/fftInst/
ingressLoop[7].ingressFifo/
buffer_fifo/infer_fifo.block_ram_performance.fifo_ram_reg}]
set_property DOB_REG {1} [get_cells {fftEngine/fftInst/
ingressLoop[7].ingressFifo
/buffer_fifo/infer_fifo.block_ram_performance.fifo_ram_reg}]
```

9. Run the timing report from the selected cell. The Tcl command to enter is:

report_timing -from [get_cells fftEngine/fftInst/ ingressLoop[7].ingressFifo /buffer_fifo/infer_fifo.block_ram_performance.fifo_ram_reg]

10. Notice that the data path delay for the RAM is now 0.622 ns.

Setting Design Properties

Next, set the configuration mode on the design. This is another property that results in a physical constraint, in this case a property of the design rather than of a cell. To begin, list all of the properties of the current design.

1. List the properties of the design in the Tcl Console:

```
list_property [current_design]
```





This command returns the list of all defined properties on the current design. To make the list more readable, you can use the standard Tcl join command to combine the properties output with "\n" newline character, resulting in each property displaying on a separate line.

```
join [list_property [current_design]] \n
```

2. The specific property of interest is CONFIG_MODE. To see what values this particular property can accept, use the <code>list_property_value</code> Tcl command:

```
join [list_property_value CONFIG_MODE [current_design]] \n
```

3. Use the **Tools** → **Edit Device Properties** command to set the CONFIG_MODE property for the current design.

The Edit Device Properties dialog box opens.

- 4. Click **Configuration Modes** to open the Configuration Modes panel.
- 5. Scroll down on the page to find the Master Serial configuration mode option, as shown in the following figure.

Edit Device Properties Use this dialog to edit the prog	ramming and configuration properties for your current design; default values are set automatically.
Q-	Configuration Modes
General Configuration Configuration Modes	Select up to two device configuration modes. JTAG is always selected. Selecting a check box assigns a mode and clicking an image displays details.
Startup Encryption	\Box Prohibit usage of the configuration pins as user I/O and persist after configuration C
Readback	Reset configuration mode C M[2:0] DOUT DIN INIT_B PUDC_B PUDC_B PROGRAM_B DONE CCLK Master Serial
Help	Reset All OK Cancel

6. Select the Master Serial configuration mode as shown, and click **OK** to close the dialog box.



The Tcl Console shows the set_property command that sets the CONFIG_MODE:

set_property CONFIG_MODE M_SERIAL [current_design]

The configuration mode has now been set.

7. Use the <property command to confirm that the CONFIG_MODE property was correctly set:</pre>

get_property CONFIG_MODE [current_design]

The property value M_SERIAL is returned by the Vivado tool.

Step 5: Saving Constraints

Notice that the Save Constraints icon is enabled because there are new design constraints. The cell and design properties you modified in Lab #2 have been added to the Vivado tool in-memory design, but are not yet saved to the target constraint file.

1. Click the Save Constraints button

The physical constraints you defined in Lab #2 are saved to the target constraint file.

2. Select the target XDC (in this case, timing.xdc) from the active constraint set in the Sources window to open the file in the Vivado IDE text editor.

Note: You might need to click the **Reload** link to update the file if it is already opened in the GUI.

Notice that the seven set_property commands you used in Lab #2 are saved to the constraint file. Only design constraints are written to the XDC file, not the object queries or reporting commands that you also used in this lab.

3. Look for the following constraints in the open constraint file:

```
set_property BEL MMCME2_ADV [get_cells clkgen/mmcm_adv_inst]
set_property LOC MMCME2_ADV_X1Y0 [get_cells clkgen/mmcm_adv_inst]
set_property PROHIBIT true [get_bels IOB_X1Y34/PAD]
set_property PROHIBIT true [get_sites AA8]
set_property DOA_REG 1 [get_cells \
{fftEngine/fftInst/ingressLoop[7].ingressFifo/buffer_fifo/
infer_fifo.block_ram_\
performance.fifo_ram_reg}]
set_property DOB_REG 1 [get_cells \
{fftEngine/fftInst/ingressLoop[7].ingressFifo/buffer_fifo/
infer_fifo.block_ram_\
performance.fifo_ram_reg}]
set_property COB_REG 1 [get_cells \
fftEngine/fftInst/ingressLoop[7].ingressFifo/buffer_fifo/
infer_fifo.block_ram_\
performance.fifo_ram_reg}]
set_property CONFIG_MODE M_SERIAL [current_design]
```

4. Exit the Vivado IDE.



Conclusion

In this lab, you learned how to use both the Vivado IDE and the Tcl Console to create and verify physical constraints. Most actions performed in the IDE result in Tcl commands being run in the Tcl Console. The Vivado IDE provides powerful interactive capabilities for developing physical and timing constraints, which can then be saved to constraint files and reused as needed.





Appendix A

Additional Resources and Legal Notices

Xilinx Resources

For support resources such as Answers, Documentation, Downloads, and Forums, see Xilinx Support.

Documentation Navigator and Design Hubs

Xilinx[®] Documentation Navigator (DocNav) provides access to Xilinx documents, videos, and support resources, which you can filter and search to find information. To open DocNav:

- From the Vivado[®] IDE, select Help → Documentation and Tutorials.
- On Windows, select Start → All Programs → Xilinx Design Tools → DocNav.
- At the Linux command prompt, enter docnav.

Xilinx Design Hubs provide links to documentation organized by design tasks and other topics, which you can use to learn key concepts and address frequently asked questions. To access the Design Hubs:

- In DocNav, click the **Design Hubs View** tab.
- On the Xilinx website, see the Design Hubs page.

Note: For more information on DocNav, see the Documentation Navigator page on the Xilinx website.





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, Kria, Spartan, Versal, Vitis, Virtex, Vivado, Zynq, and other designated brands included herein are trademarks of Xilinx in the United States and other countries. All other trademarks are the property of their respective owners.