Introduction

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**.

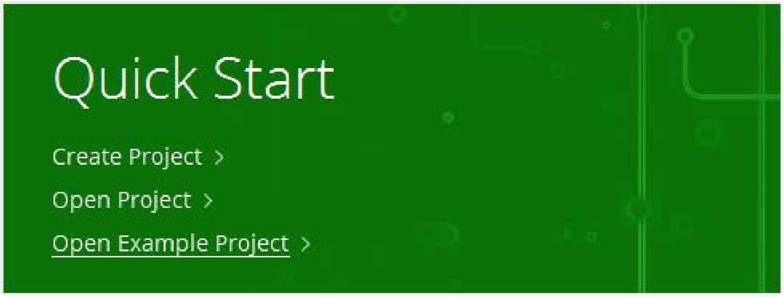
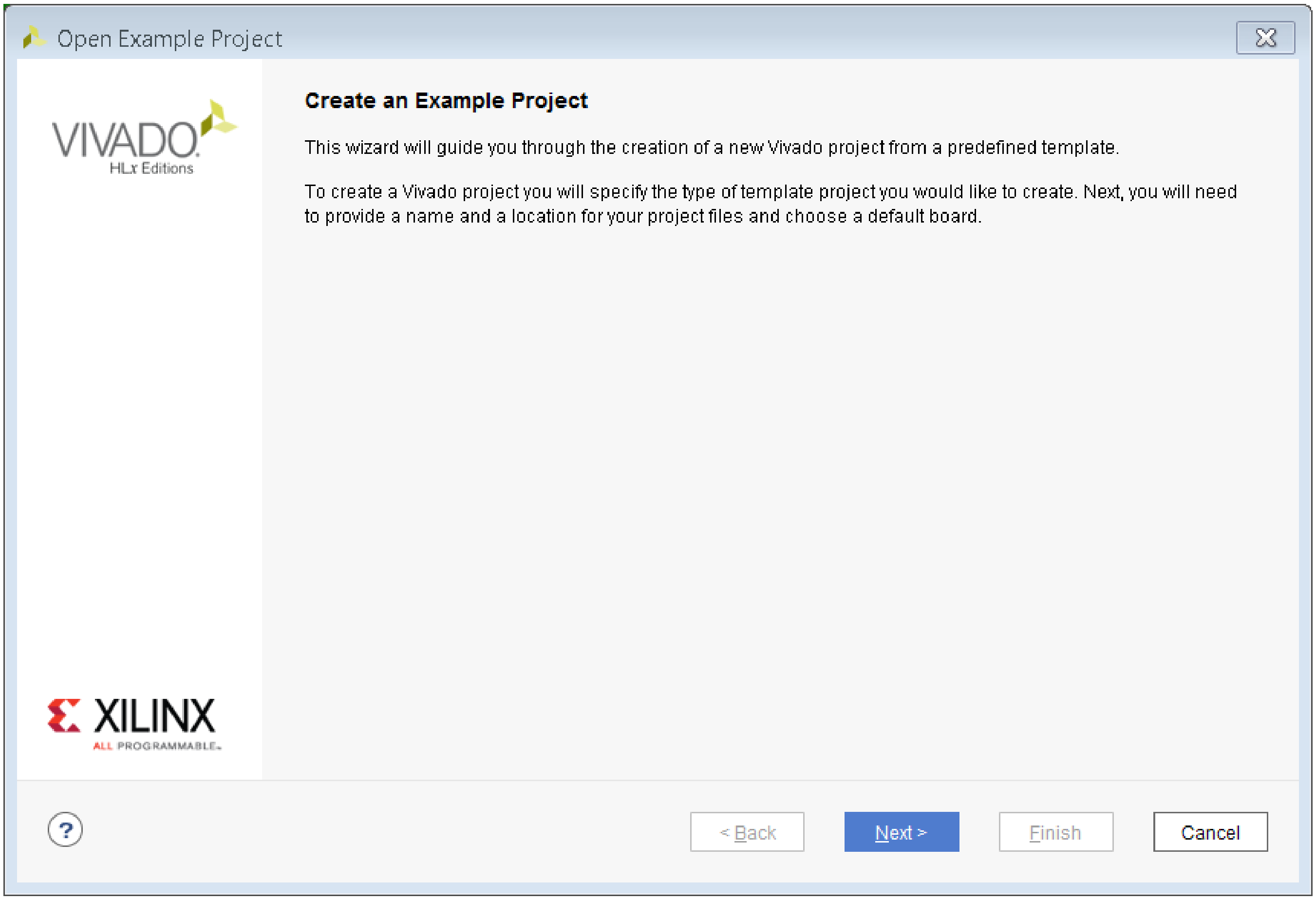


Figure 39: Open Example Project

3. In the Create an Example Project screen, click **Next**.



Open Example Project

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.

Figure 40: Open Example Project Wizard—Create an Example Project Screen

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

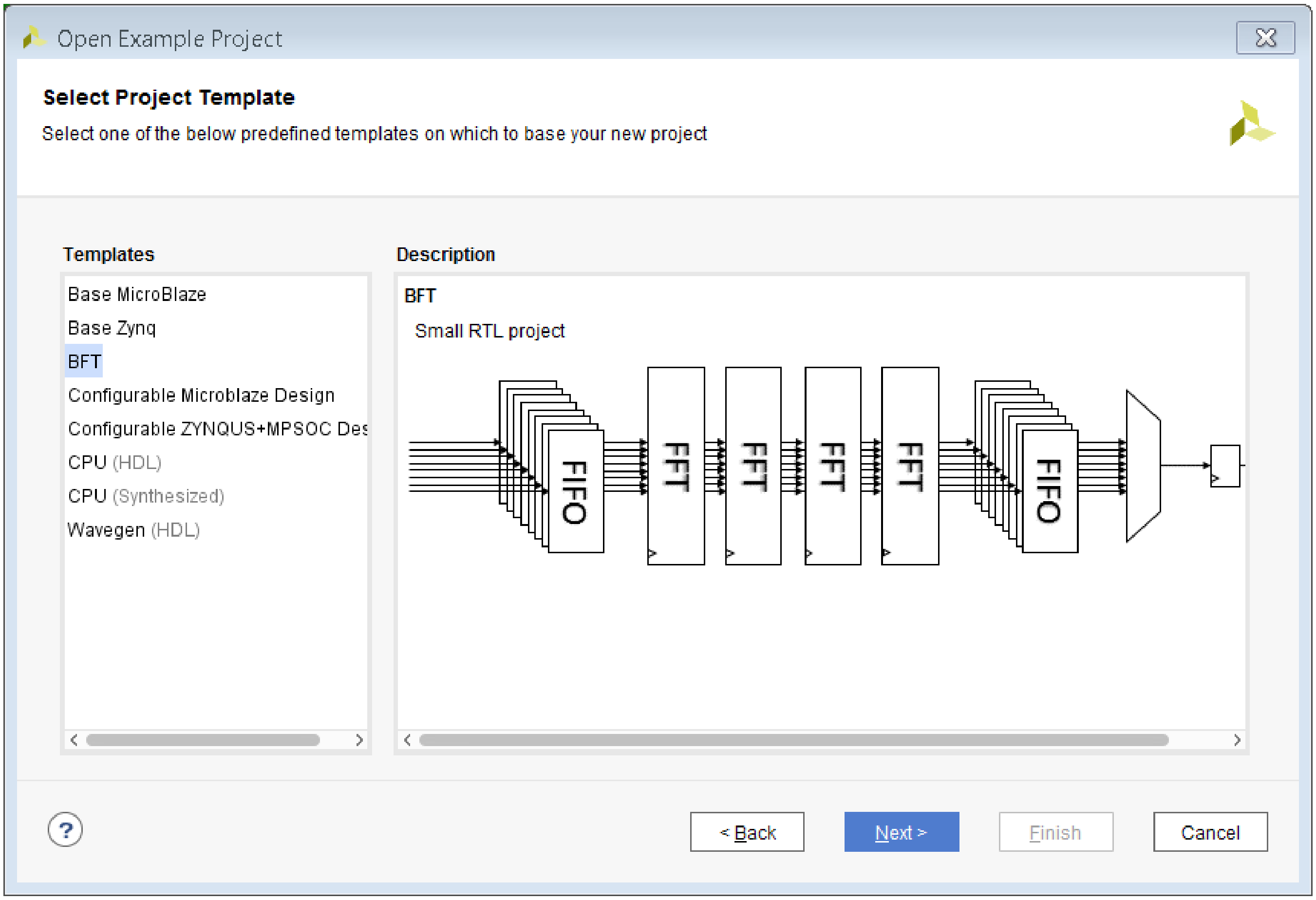
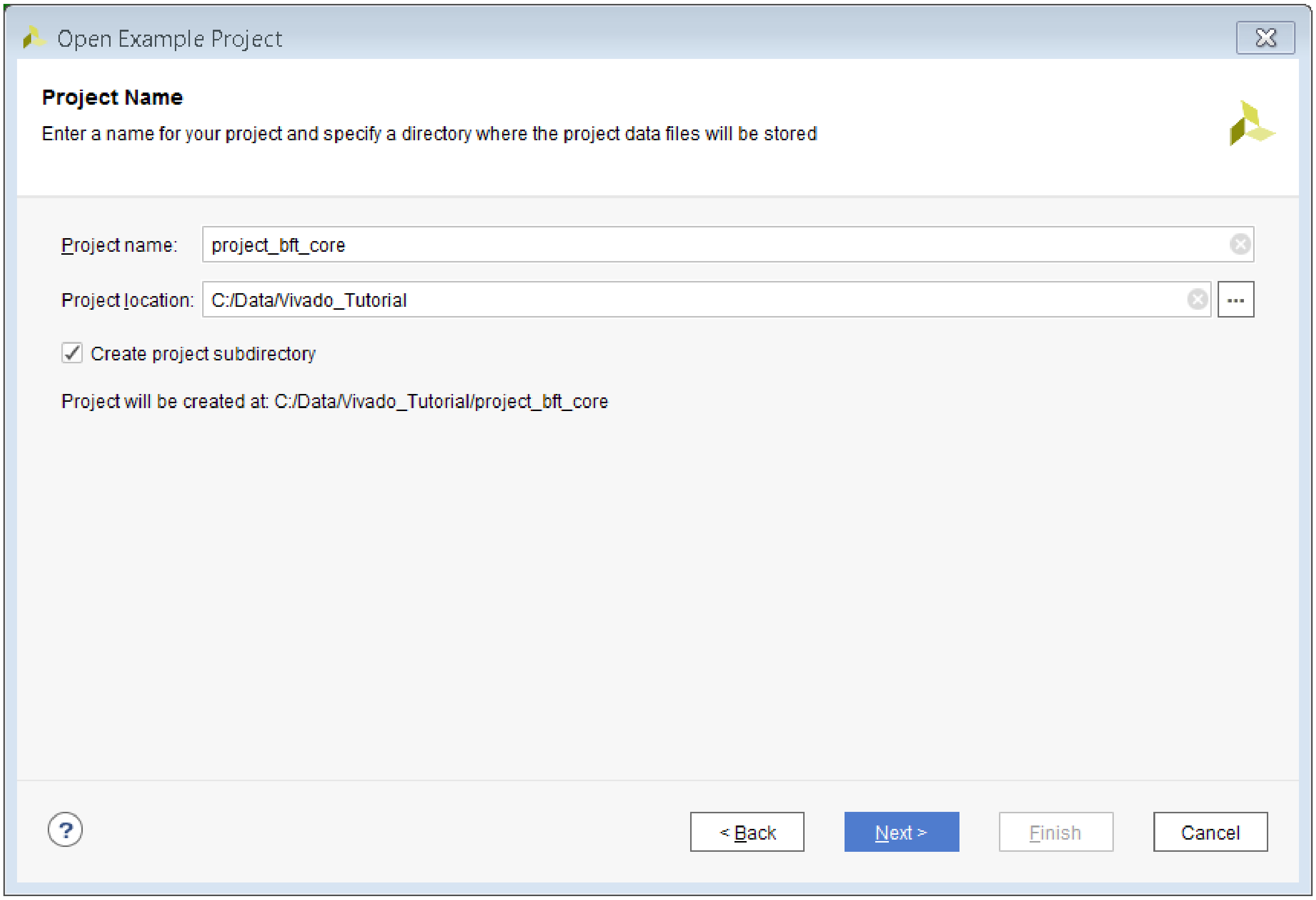


Figure 41: Open Example Project Wizard—Select Project Template Screen

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



Project Name

Enter a name for your project and specify a directory where the project data files will be stored

Project name: project_bft_core

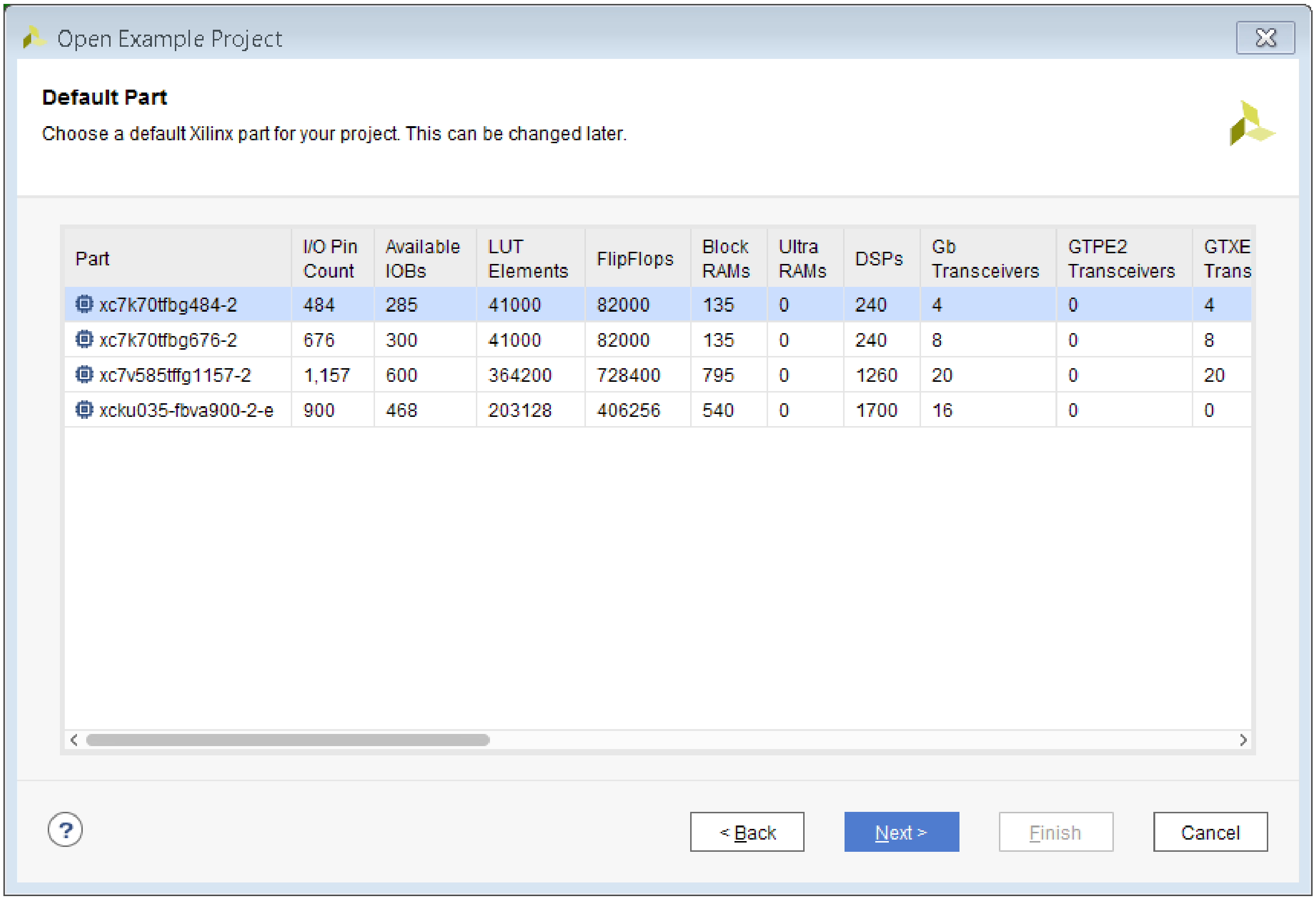
Project location: C:/Data/Vivado_Tutorial

☑ Create project subdirectory

Project will be created at: C:/Data/Vivado_Tutorial/project_bft_core

Figure 42: Open Example Project Wizard— Project Name Screen

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



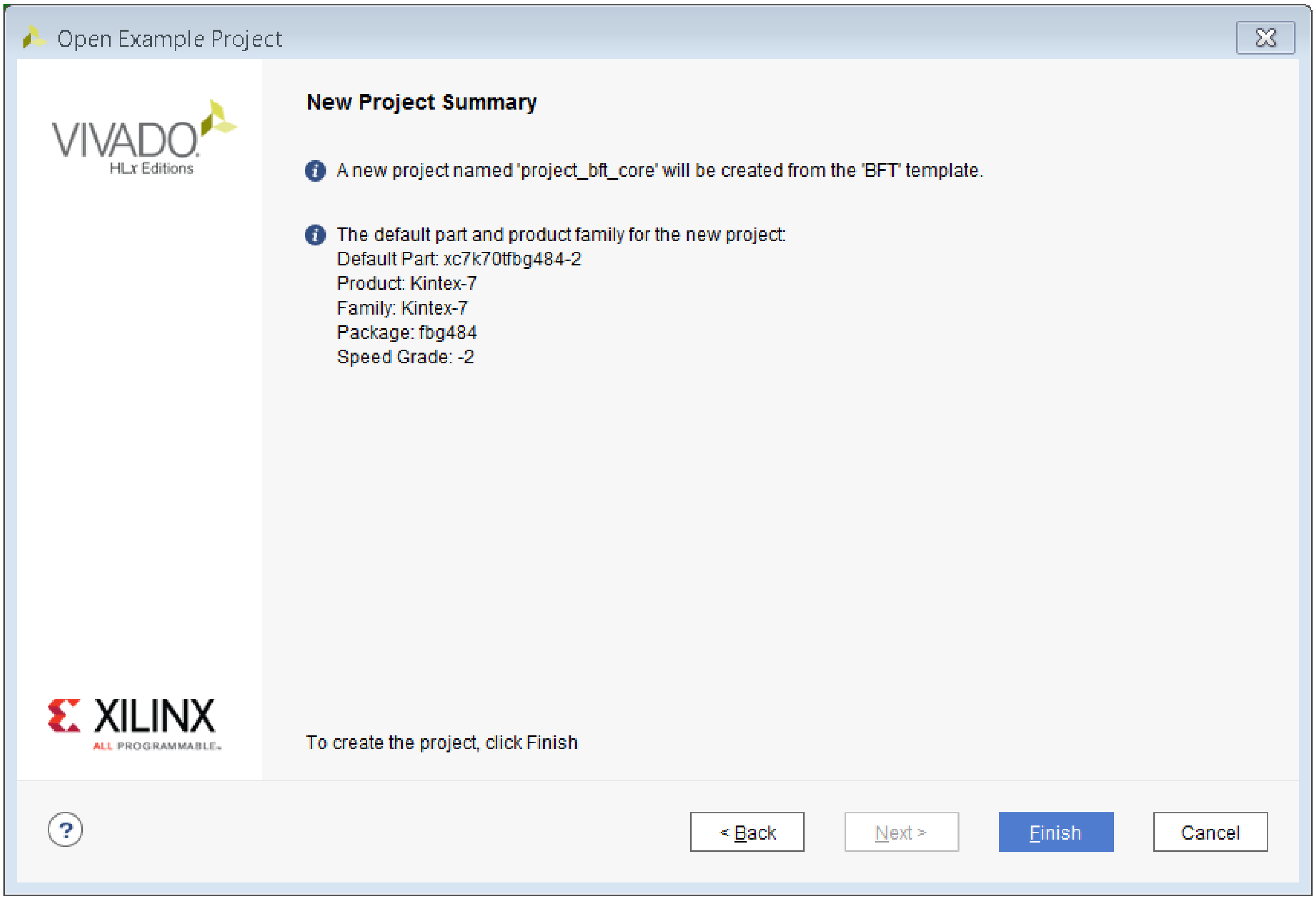
Default Part

Choose a default Xilinx part for your project. This can be changed later.

Part	I/O Pin Count	Available IOBs	LUT Elements	FlipFlops	Block RAMs	Ultra RAMs	DSPs	Gb Transceivers	GTPE2 Transceivers	GTXE Trans
xc7k70tfbg484-2	484	285	41000	82000	135	0	240	4	0	4
xc7k70tfbg676-2	676	300	41000	82000	135	0	240	8	0	8
xc7v585tffg1157-2	1,157	600	364200	728400	795	0	1260	20	0	20
xcku035-fbva900-2-e	900	468	203128	406256	540	0	1700	16	0	0

Figure 43: Open Example Project Wizard—Default Part Screen

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



New Project Summary

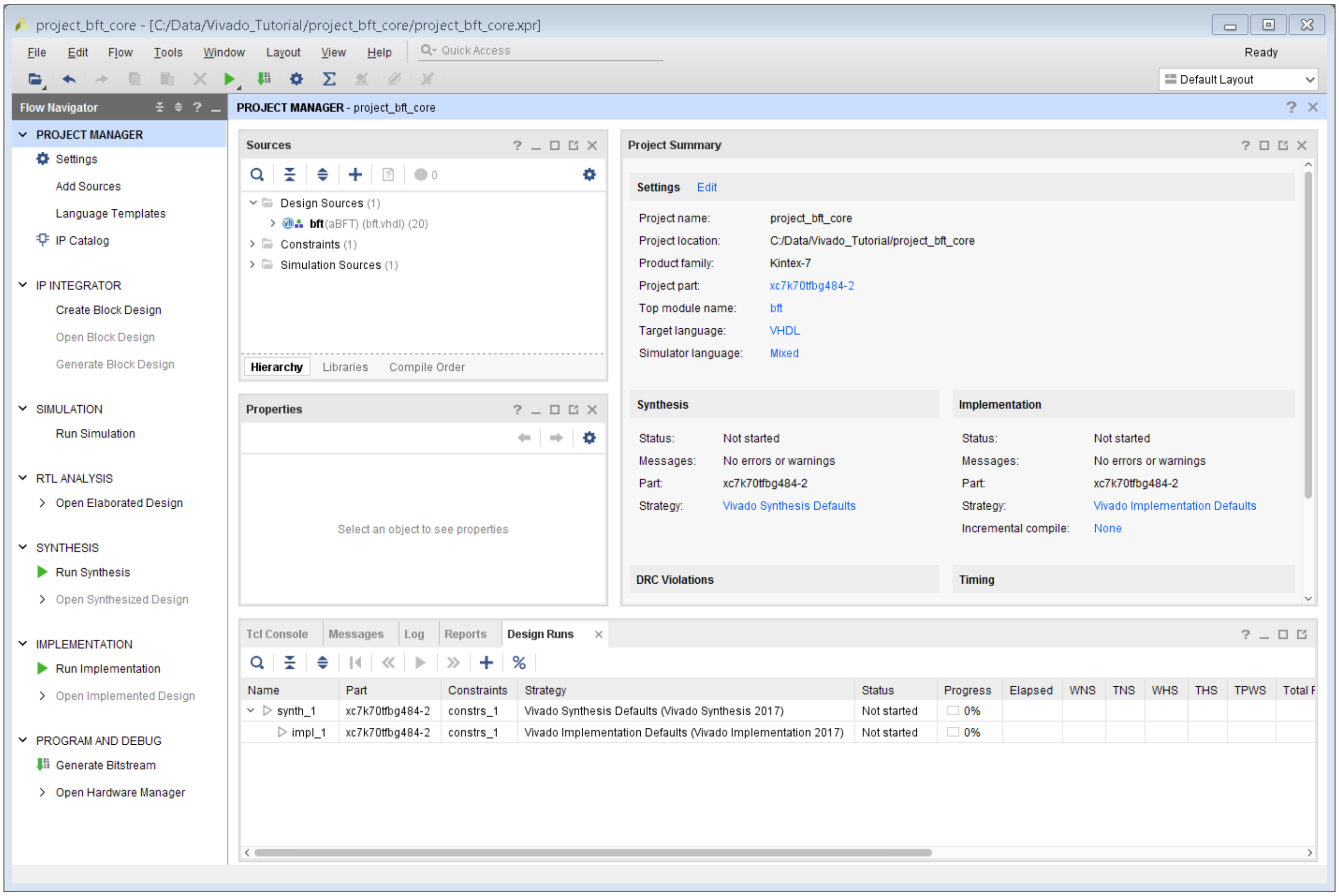
A new project named 'project_bft_core' will be created from the 'BFT' template.

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

To create the project, click Finish

Figure 44: Open Example Project Wizard—New Project Summary Screen

The Vivado IDE opens with the default view.



project_bft_core - [C:/Data/Vivado_Tutorial/project_bft_core/project_bft_core.xpr]

PROJECT MANAGER - project_bft_core

Sources

- Design Sources (1)
  - bft(aBFT) (bft.vhdl) (20)
- Constraints (1)
- Simulation Sources (1)

Properties

Select an object to see properties

Project Summary

Settings Edit

Project name:	project_bft_core
Project location:	C:/Data/Vivado_Tutorial/project_bft_core
Product family:	Kintex-7
Project part:	xc7k70tfbg484-2
Top module name:	bft
Target language:	VHDL
Simulator language:	Mixed

Synthesis

Status:	Not started
Messages:	No errors or warnings
Part:	xc7k70tfbg484-2
Strategy:	Vivado Synthesis Defaults

Implementation

Status:	Not started
Messages:	No errors or warnings
Part:	xc7k70tfbg484-2
Strategy:	Vivado Implementation Defaults
Incremental compile:	None

DRC Violations

Timing

Design Runs

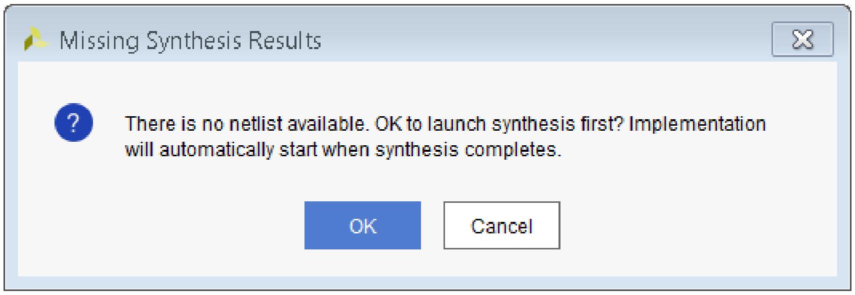
Name	Part	Constraints	Strategy	Status	Progress	Elapsed	WNS	TNS	WHS	THS	TPWS	Total F
synth_1	xc7k70tfbg484-2	constrs_1	Vivado Synthesis Defaults (Vivado Synthesis 2017)	Not started	0%							
impl_1	xc7k70tfbg484-2	constrs_1	Vivado Implementation Defaults (Vivado Implementation 2017)	Not started	0%							

Figure 45: Vivado IDE Showing project_bft_core Project Details

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.



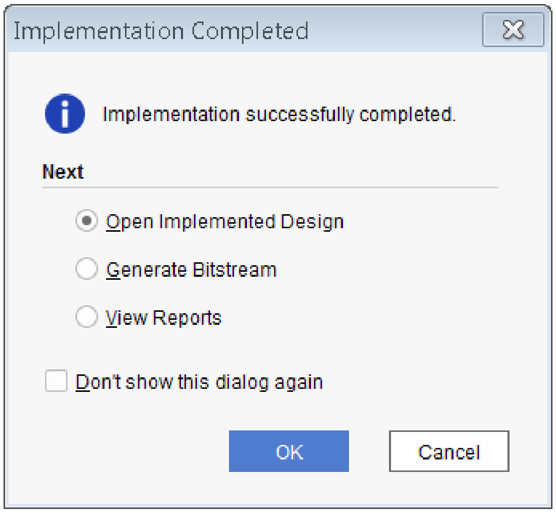
Missing Synthesis Results

There is no netlist available. OK to launch synthesis first? Implementation will automatically start when synthesis completes.

Figure 46: Missing Synthesis Results

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.



Implementation Completed

Implementation successfully completed.

Next

● Open Implemented Design

○ Generate Bitstream

○ View Reports

□ Don't show this dialog again

Figure 47: Implementation Completed

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 to view the detailed routing resources in the Device window.

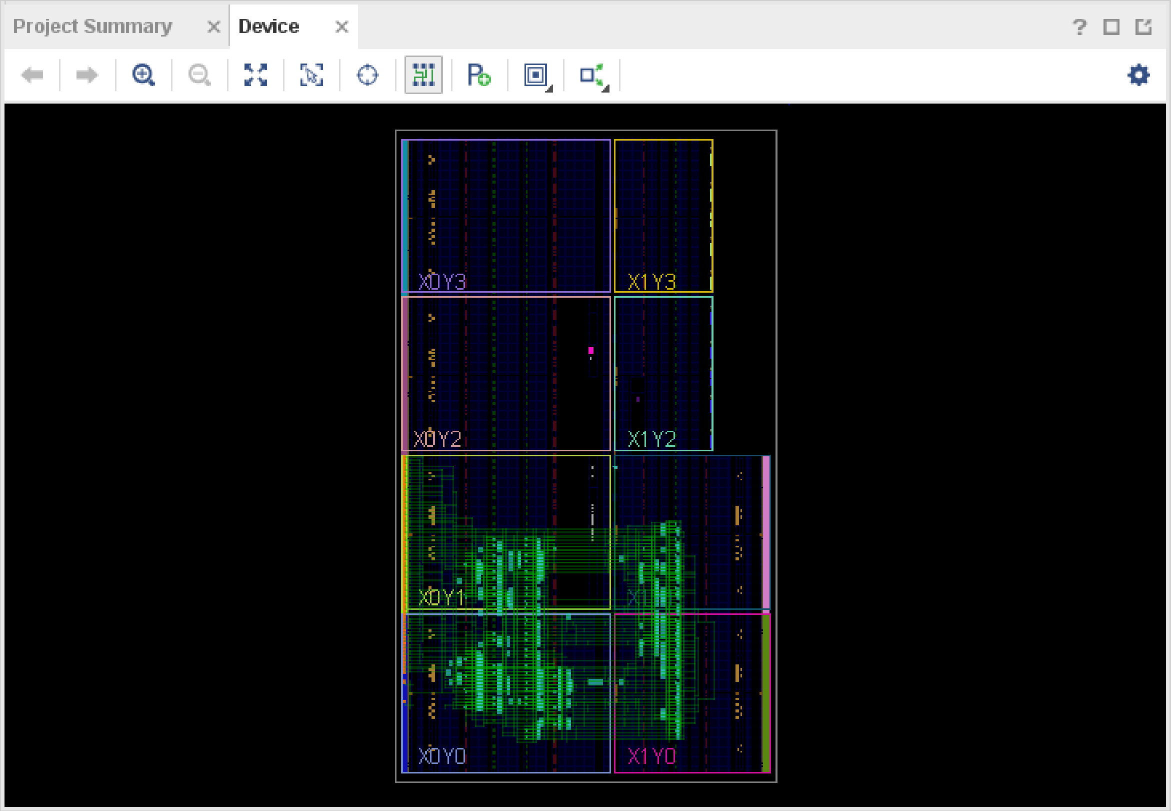


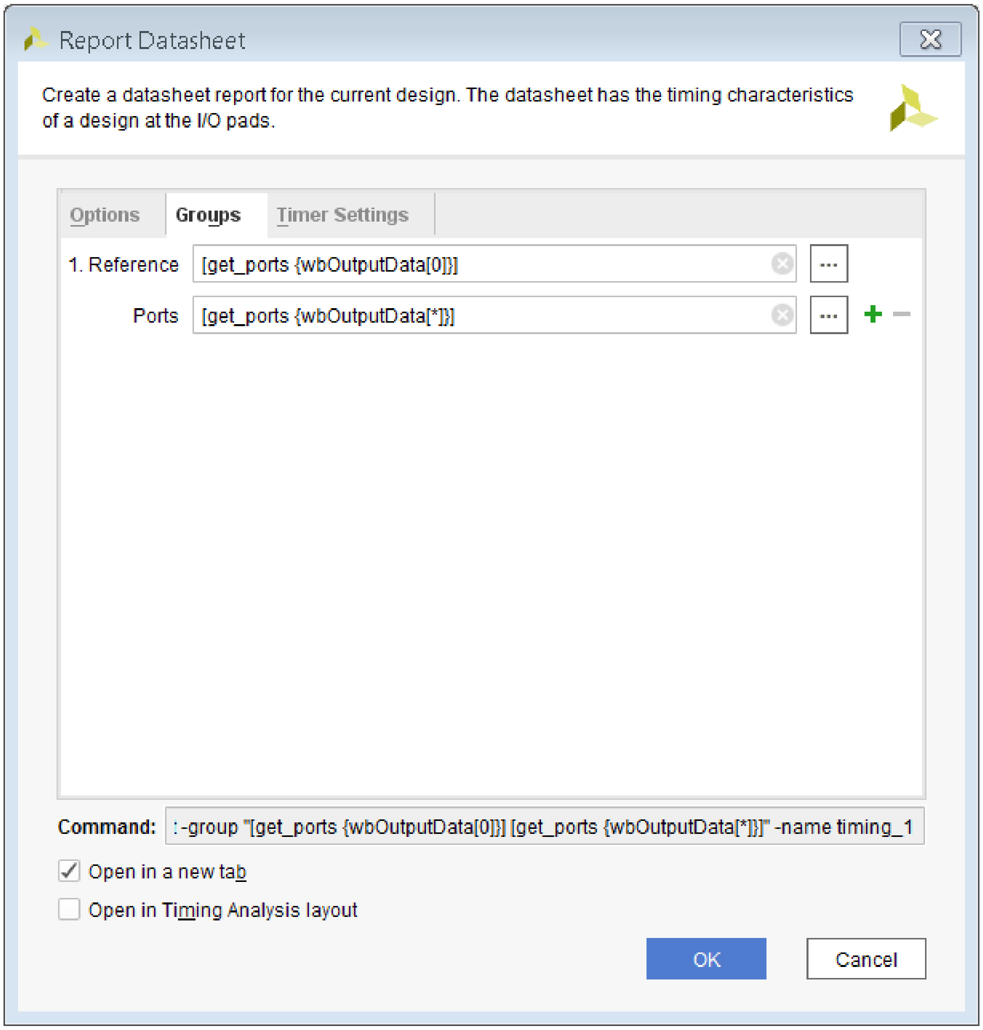
Figure 48: Device Window

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 **Tools > Timing > 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[*]}]`



Create a datasheet report for the current design. The datasheet has the timing characteristics of a design at the I/O pads.

1. Reference [get_ports {wbOutputData[0]}]

Ports [get_ports {wbOutputData[*]}]

Command: : -group "[get_ports {wbOutputData[0]}] [get_ports {wbOutputData[*]}]" -name timing_1

☑ Open in a new tab

☐ Open in Timing Analysis layout

Figure 49: Report Datasheet

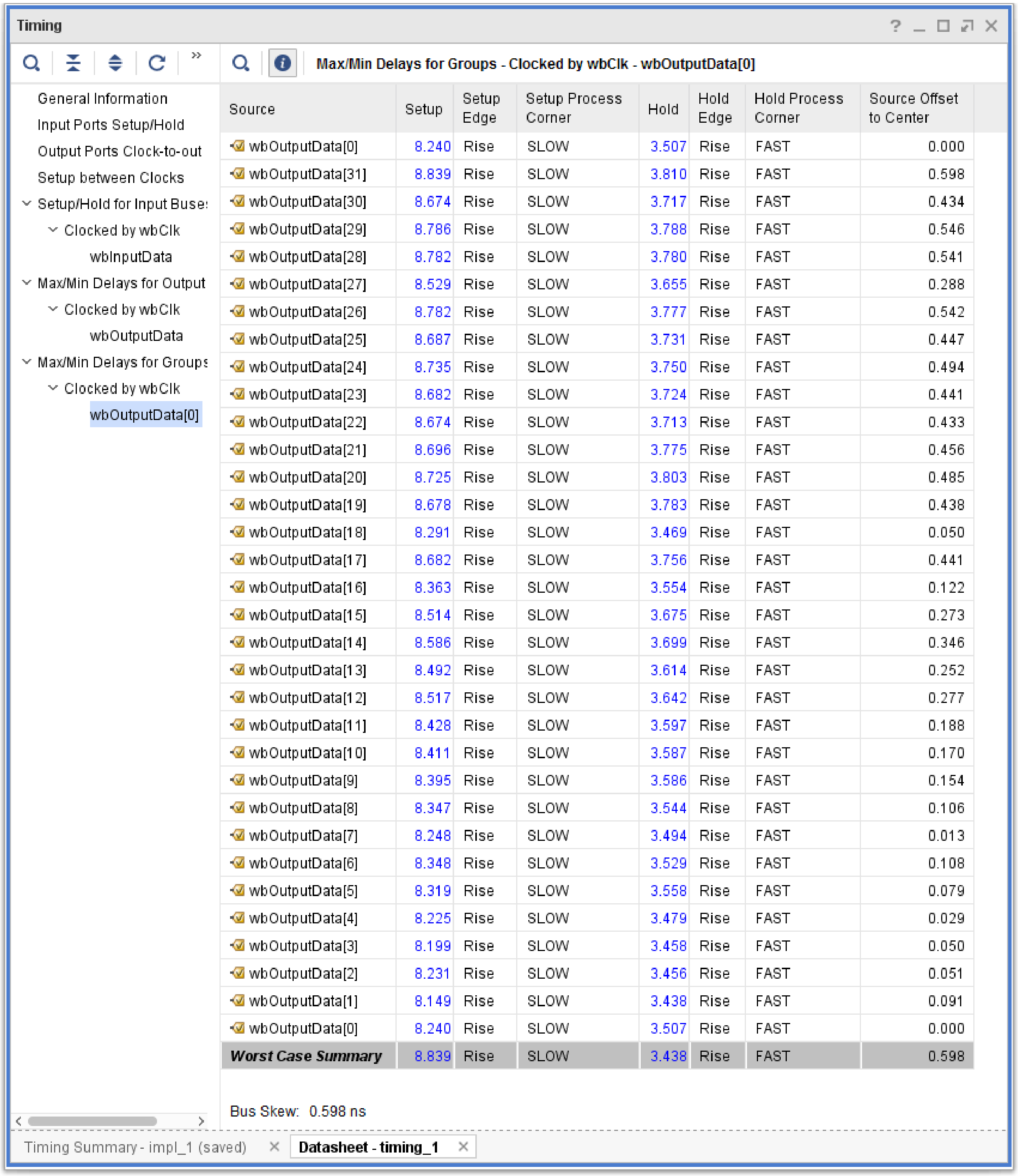
3. Click **OK**.

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

4. Click the **Maximize** 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 600 ps. The goal is to minimize the skew across the bus to less than 100 ps.



Timing

- General Information
- Input Ports Setup/Hold
- Output Ports Clock-to-out
- Setup between Clocks
- Setup/Hold for Input Buse:
  - Clocked by wbClk
    - wbInputData
- Max/Min Delays for Output
  - Clocked by wbClk
    - wbOutputData
- Max/Min Delays for Groups
  - Clocked by wbClk
    - wbOutputData[0]

Max/Min Delays for Groups - Clocked by wbClk - wbOutputData[0]

Source	Setup	Setup Edge	Setup Process Corner	Hold	Hold Edge	Hold Process Corner	Source Offset to Center
wbOutputData[0]	8.240	Rise	SLOW	3.507	Rise	FAST	0.000
wbOutputData[31]	8.839	Rise	SLOW	3.810	Rise	FAST	0.598
wbOutputData[30]	8.674	Rise	SLOW	3.717	Rise	FAST	0.434
wbOutputData[29]	8.786	Rise	SLOW	3.788	Rise	FAST	0.546
wbOutputData[28]	8.782	Rise	SLOW	3.780	Rise	FAST	0.541
wbOutputData[27]	8.529	Rise	SLOW	3.655	Rise	FAST	0.288
wbOutputData[26]	8.782	Rise	SLOW	3.777	Rise	FAST	0.542
wbOutputData[25]	8.687	Rise	SLOW	3.731	Rise	FAST	0.447
wbOutputData[24]	8.735	Rise	SLOW	3.750	Rise	FAST	0.494
wbOutputData[23]	8.682	Rise	SLOW	3.724	Rise	FAST	0.441
wbOutputData[22]	8.674	Rise	SLOW	3.713	Rise	FAST	0.433
wbOutputData[21]	8.696	Rise	SLOW	3.775	Rise	FAST	0.456
wbOutputData[20]	8.725	Rise	SLOW	3.803	Rise	FAST	0.485
wbOutputData[19]	8.678	Rise	SLOW	3.783	Rise	FAST	0.438
wbOutputData[18]	8.291	Rise	SLOW	3.469	Rise	FAST	0.050
wbOutputData[17]	8.682	Rise	SLOW	3.756	Rise	FAST	0.441
wbOutputData[16]	8.363	Rise	SLOW	3.554	Rise	FAST	0.122
wbOutputData[15]	8.514	Rise	SLOW	3.675	Rise	FAST	0.273
wbOutputData[14]	8.586	Rise	SLOW	3.699	Rise	FAST	0.346
wbOutputData[13]	8.492	Rise	SLOW	3.614	Rise	FAST	0.252
wbOutputData[12]	8.517	Rise	SLOW	3.642	Rise	FAST	0.277
wbOutputData[11]	8.428	Rise	SLOW	3.597	Rise	FAST	0.188
wbOutputData[10]	8.411	Rise	SLOW	3.587	Rise	FAST	0.170
wbOutputData[9]	8.395	Rise	SLOW	3.586	Rise	FAST	0.154
wbOutputData[8]	8.347	Rise	SLOW	3.544	Rise	FAST	0.106
wbOutputData[7]	8.248	Rise	SLOW	3.494	Rise	FAST	0.013
wbOutputData[6]	8.348	Rise	SLOW	3.529	Rise	FAST	0.108
wbOutputData[5]	8.319	Rise	SLOW	3.558	Rise	FAST	0.079
wbOutputData[4]	8.225	Rise	SLOW	3.479	Rise	FAST	0.029
wbOutputData[3]	8.199	Rise	SLOW	3.458	Rise	FAST	0.050
wbOutputData[2]	8.231	Rise	SLOW	3.456	Rise	FAST	0.051
wbOutputData[1]	8.149	Rise	SLOW	3.438	Rise	FAST	0.091
wbOutputData[0]	8.240	Rise	SLOW	3.507	Rise	FAST	0.000
Worst Case Summary	8.839	Rise	SLOW	3.438	Rise	FAST	0.598

Bus Skew: 0.598 ns

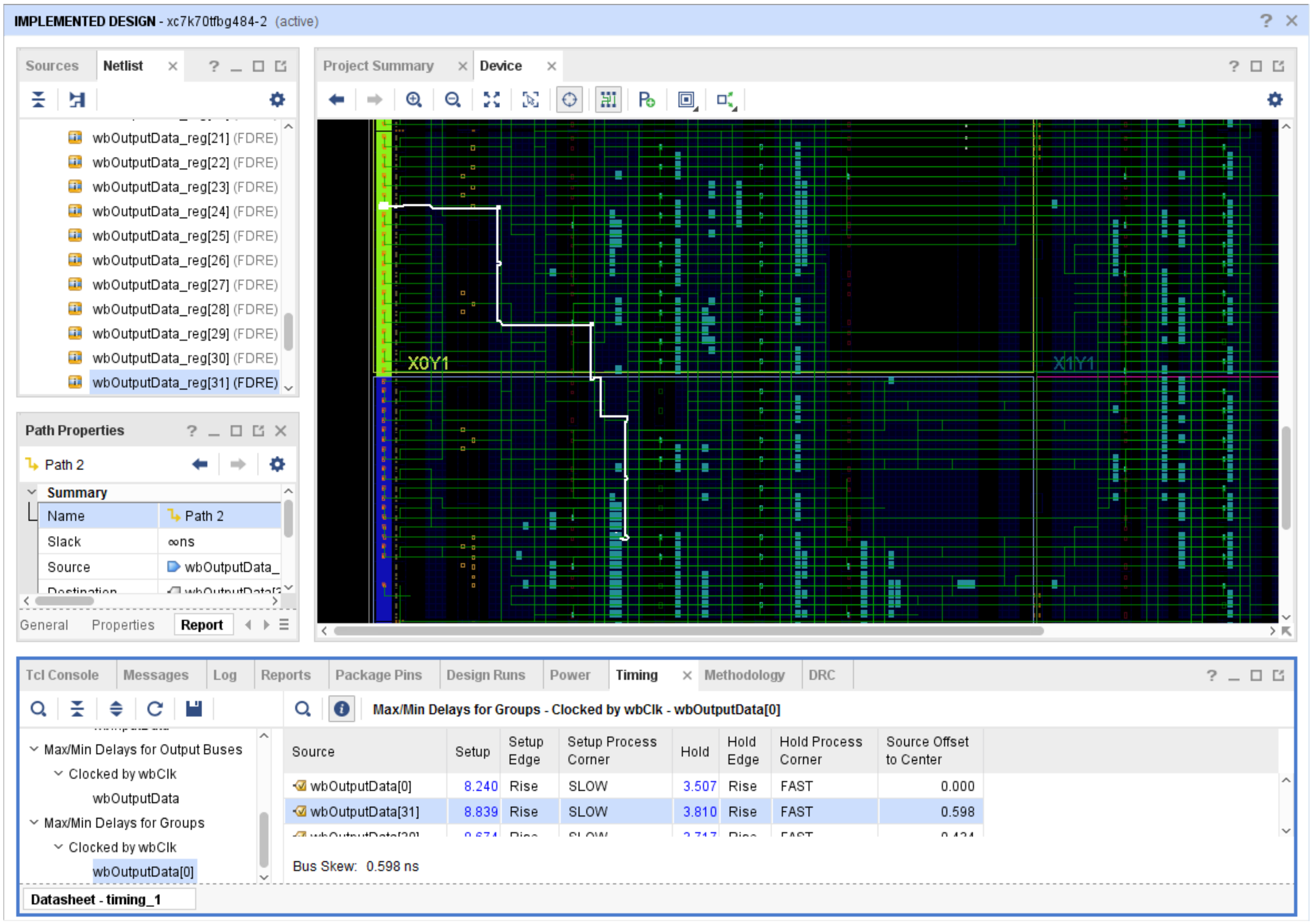
Figure 50: Report Datasheet Results

6. Click the **Restore** button 🗗 so you can simultaneously see the Device window and the Timing - Datasheet results.

7. Click the hyperlink for the Setup time of the source `wbOutputData[31]`.

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

   Note: *Make sure that the **Autofit Selection** ⊕ is highlighted in the Device window so you can see the entire path, as shown in the following figure.*



wbOutputData_reg[21] (FDRE)
wbOutputData_reg[22] (FDRE)
wbOutputData_reg[23] (FDRE)
wbOutputData_reg[24] (FDRE)
wbOutputData_reg[25] (FDRE)
wbOutputData_reg[26] (FDRE)
wbOutputData_reg[27] (FDRE)
wbOutputData_reg[28] (FDRE)
wbOutputData_reg[29] (FDRE)
wbOutputData_reg[30] (FDRE)
wbOutputData_reg[31] (FDRE)

Path Properties

Path 2

Summary

Name	Path 2
Slack	∞ns
Source	wbOutputData_
Destination	wbOutputData[3

- Max/Min Delays for Output Buses
  - Clocked by wbClk
    - wbOutputData
- Max/Min Delays for Groups
  - Clocked by wbClk
    - wbOutputData[0]

Max/Min Delays for Groups - Clocked by wbClk - wbOutputData[0]

Source	Setup	Setup Edge	Setup Process Corner	Hold	Hold Edge	Hold Process Corner	Source Offset to Center
wbOutputData[0]	8.240	Rise	SLOW	3.507	Rise	FAST	0.000
wbOutputData[31]	8.839	Rise	SLOW	3.810	Rise	FAST	0.598
wbOutputData[30]	8.674	Rise	SLOW	3.717	Rise	FAST	0.434

Bus Skew: 0.598 ns

Figure 51: Detailed Routing View

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, as shown in Figure 52. 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.

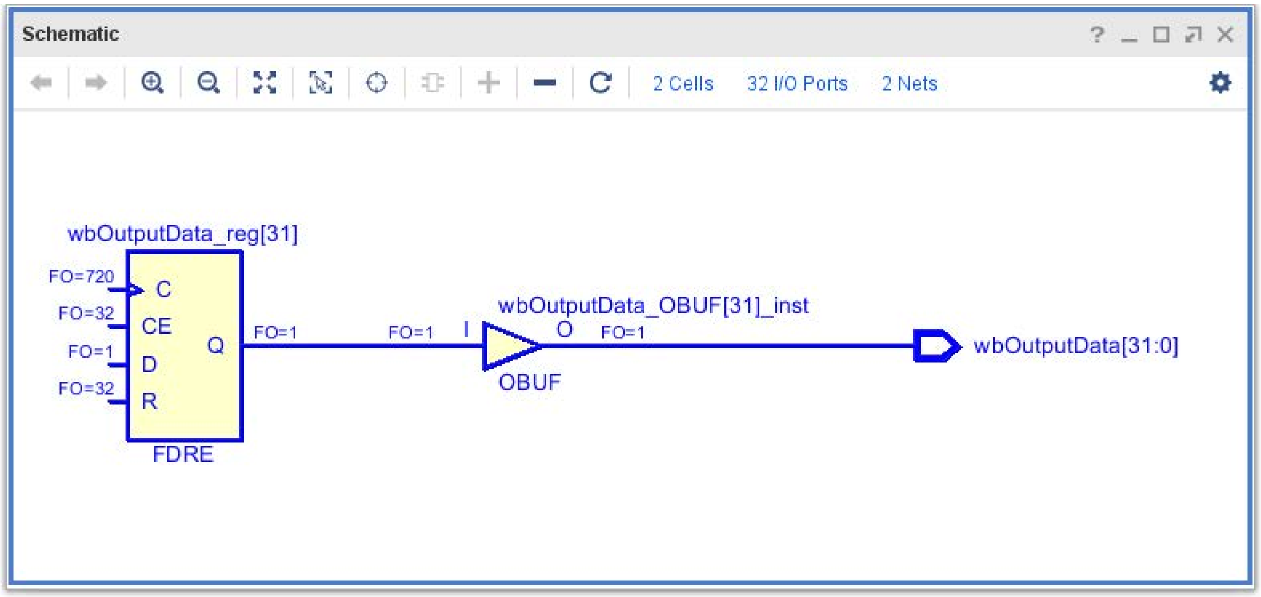


Figure 52: Schematic View of Output Path

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.

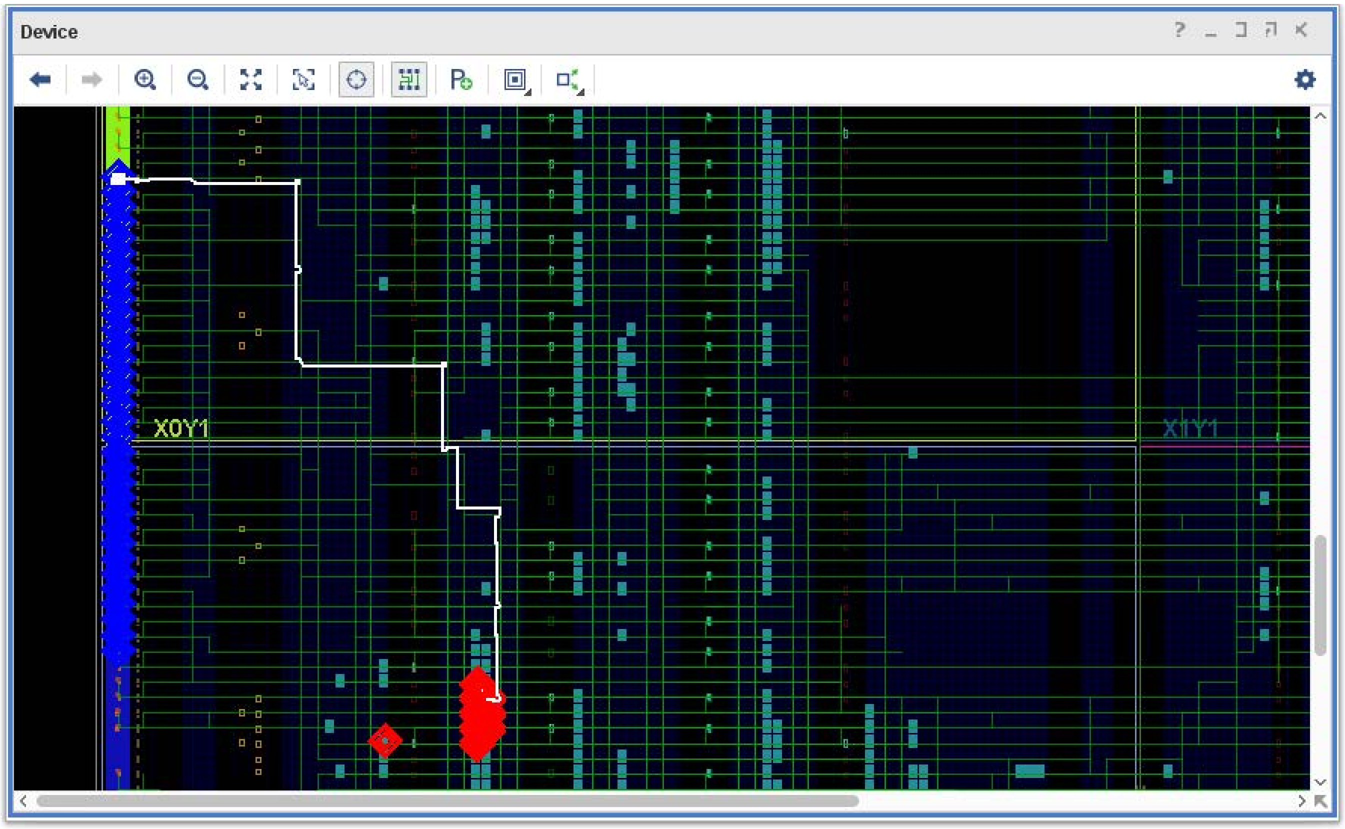


Figure 53: Marked Startpoints (red) and Endpoints (blue)

Zooming out on the Device window displays a picture similar to Figure 53.

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 as shown in Figure 54.

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.

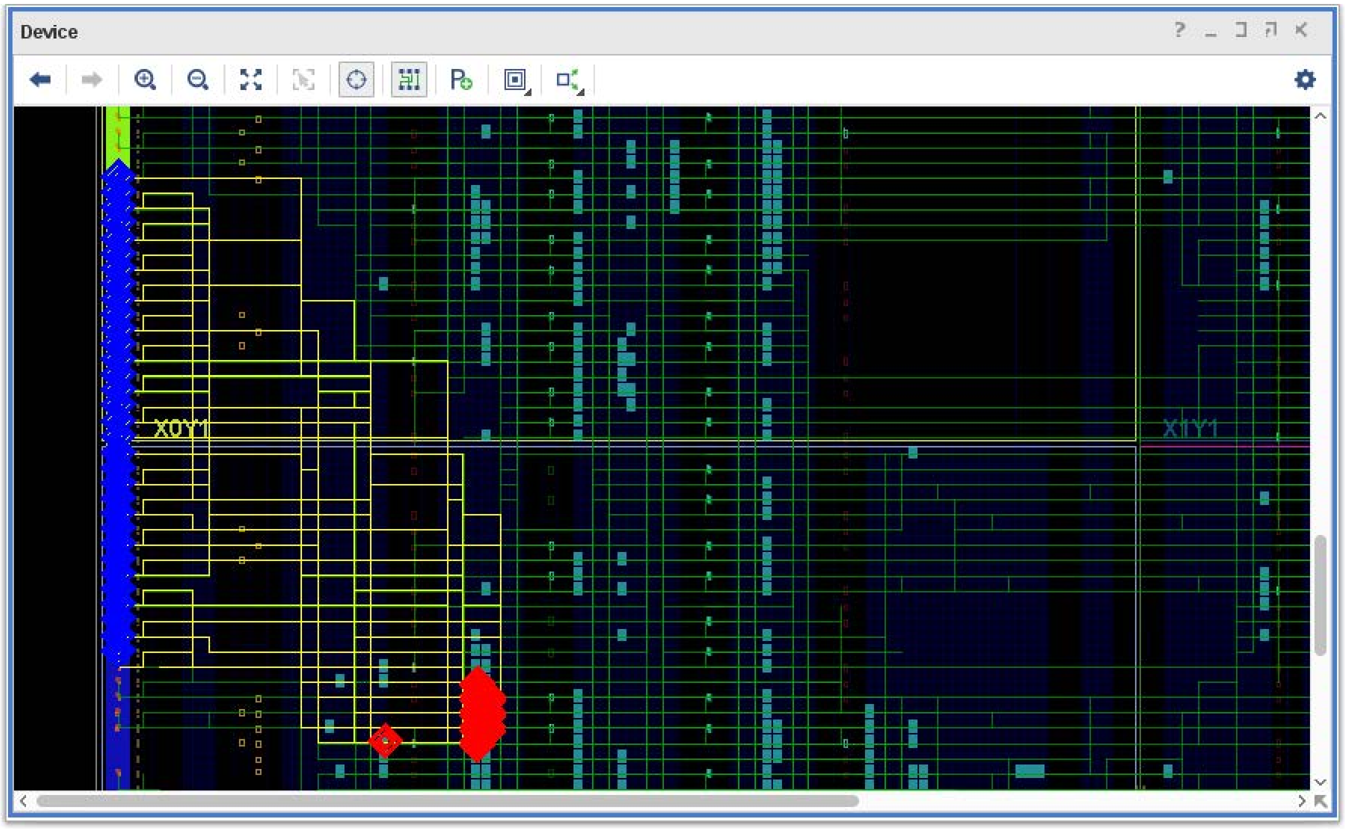


Figure 54: Highlighted Routes

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

1. In the Device window click to disable **Routing Resources** and make sure **AutoFit Selection** is still enabled on the sidebar menu.

   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. The view should be similar to Figure 55. By examining the device resources around the selected ports, you can identify a range of placement Sites for the output registers.

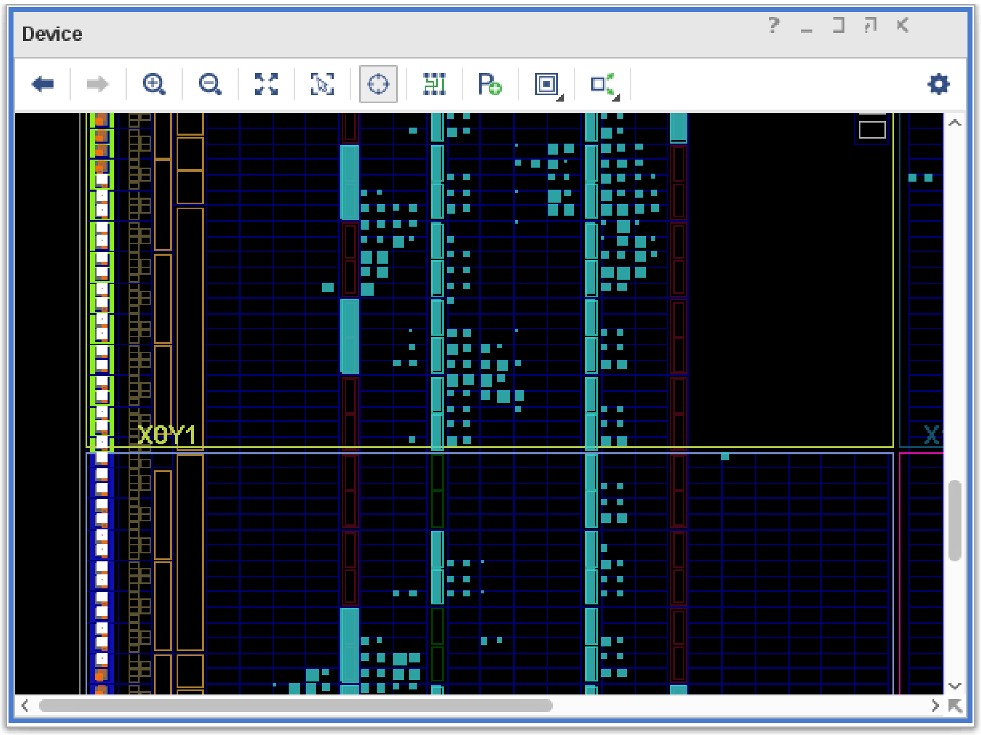
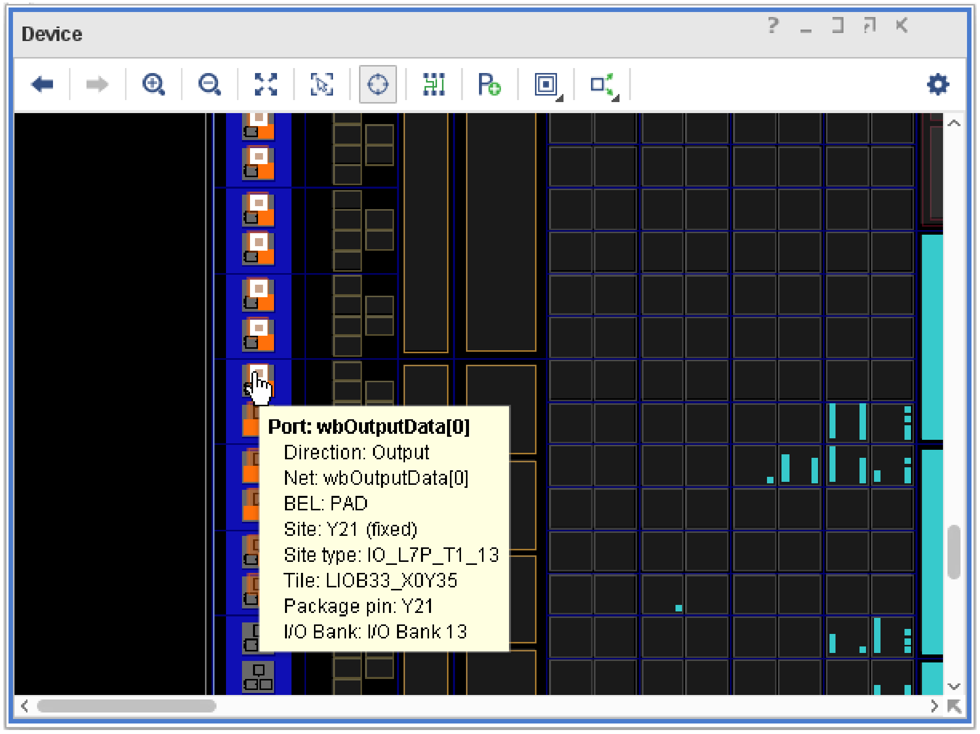


Figure 55: Selected wbOutputData Ports

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



Port: wbOutputData[0]
Direction: Output
Net: wbOutputData[0]
BEL: PAD
Site: Y21 (fixed)
Site type: IO_L7P_T1_13
Tile: LIOB33_X0Y35
Package pin: Y21
I/O Bank: I/O Bank 13

Figure 56: wbOutputData[0] Placement Details

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

As seen in Figure 56, 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 X0Y36. 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_X0Y[expr 36 + $i]]]
}
```

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
}
```

6. Now, 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. Then 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:

```
report_route_status
```

10. Click the **Routing Resources** button 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, as seen in Figure 57, 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.

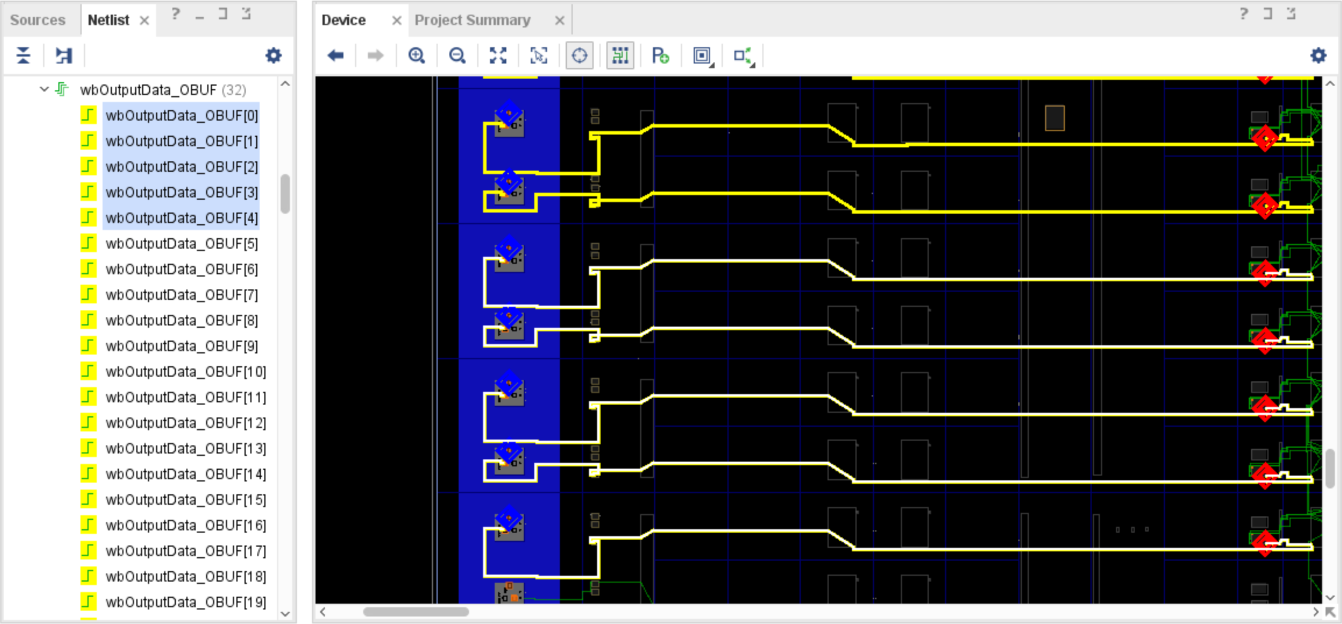


Figure 57: Improved Placement and Routing

14. Run the Tools > Timing > Report Datasheet command again.

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

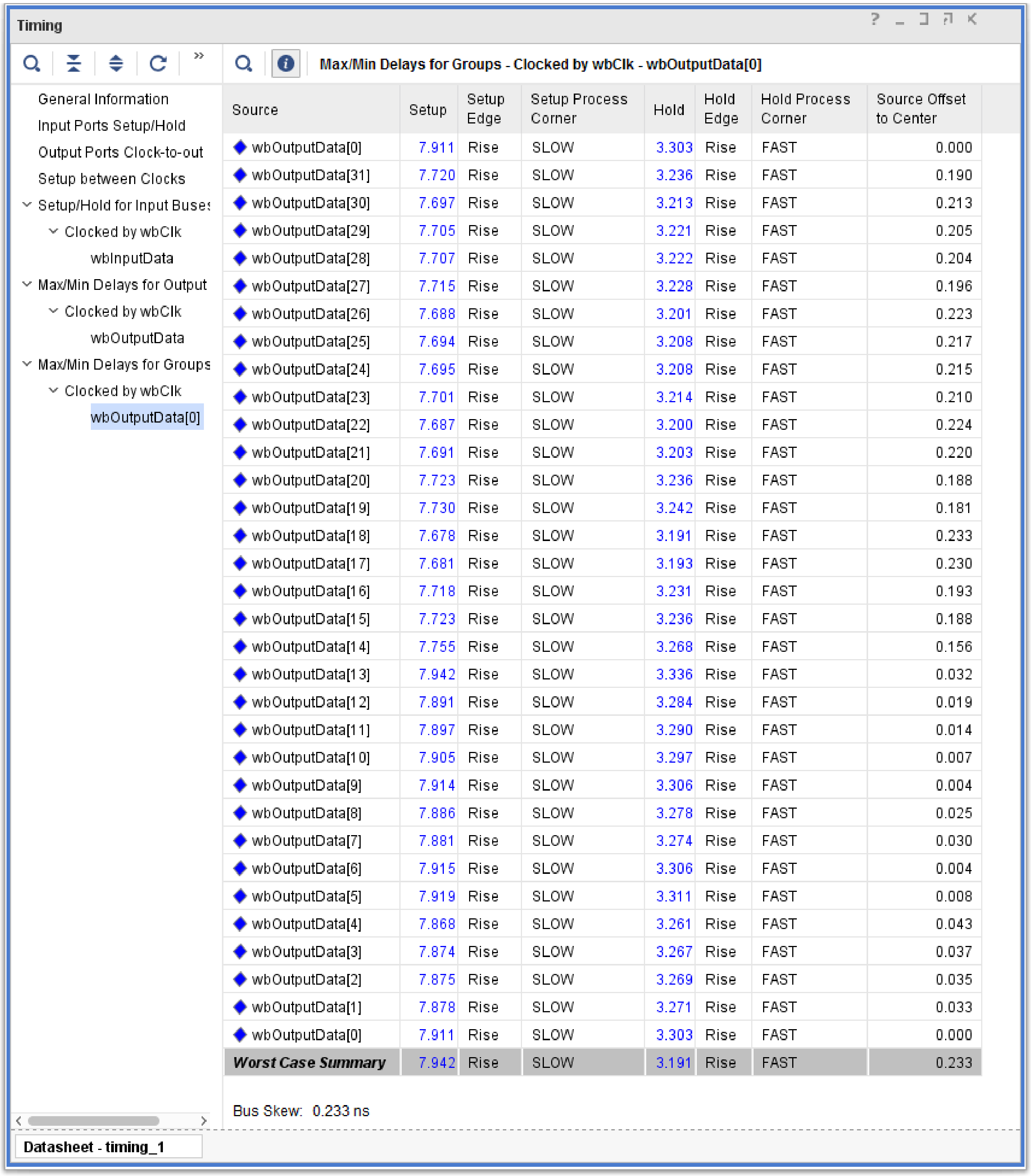
    - Reference: `[get_ports {wbOutputData[0]}]`
    - Ports: `[get_ports {wbOutputData[*]}]`

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

    Examining the results shown in Figure 58, 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, X0Y0 and X0Y1, which introduces clock network skew. Looking at the `wbOutputData` bus, notice that the Setup 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.



Max/Min Delays for Groups - Clocked by wbClk - wbOutputData[0]

Source	Setup	Setup Edge	Setup Process Corner	Hold	Hold Edge	Hold Process Corner	Source Offset to Center
wbOutputData[0]	7.911	Rise	SLOW	3.303	Rise	FAST	0.000
wbOutputData[31]	7.720	Rise	SLOW	3.236	Rise	FAST	0.190
wbOutputData[30]	7.697	Rise	SLOW	3.213	Rise	FAST	0.213
wbOutputData[29]	7.705	Rise	SLOW	3.221	Rise	FAST	0.205
wbOutputData[28]	7.707	Rise	SLOW	3.222	Rise	FAST	0.204
wbOutputData[27]	7.715	Rise	SLOW	3.228	Rise	FAST	0.196
wbOutputData[26]	7.688	Rise	SLOW	3.201	Rise	FAST	0.223
wbOutputData[25]	7.694	Rise	SLOW	3.208	Rise	FAST	0.217
wbOutputData[24]	7.695	Rise	SLOW	3.208	Rise	FAST	0.215
wbOutputData[23]	7.701	Rise	SLOW	3.214	Rise	FAST	0.210
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

Bus Skew: 0.233 ns

Figure 58: Report Datasheet—Improved Skew

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]]]"
   }
   ```

   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:

   ```
   14  { CLBLL_LL_AQ CLBLL_LOGIC_OUTS4 WW2BEG0 IMUX_L34 IOI_OLOGIC0_D1 LIOI_OLOGIC0_OQ LIOI_O0 }
   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_OLOGIC0_OQ LIOI_O0 }
   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_OLOGIC0_OQ LIOI_O0 }
   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_OLOGIC0_OQ LIOI_O0 }
   21  { CLBLL_LL_AQ CLBLL_LOGIC_OUTS4 WW2BEG0 IMUX_L34 IOI_OLOGIC1_D1 LIOI_OLOGIC1_OQ LIOI_O1 }
   22  { CLBLL_LL_AQ CLBLL_LOGIC_OUTS4 WW2BEG0 IMUX_L34 IOI_OLOGIC0_D1 LIOI_OLOGIC0_OQ LIOI_O0 }
   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_OLOGIC0_OQ LIOI_O0 }
   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_OLOGIC0_OQ LIOI_O0 }
   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_OLOGIC0_OQ LIOI_O0 }
   29  { CLBLL_LL_AQ CLBLL_LOGIC_OUTS4 WW2BEG0 IMUX_L34 IOI_OLOGIC1_D1 LIOI_OLOGIC1_OQ LIOI_O1 }
   30  { CLBLL_LL_AQ CLBLL_LOGIC_OUTS4 WW2BEG0 IMUX_L34 IOI_OLOGIC0_D1 LIOI_OLOGIC0_OQ LIOI_O0 }
   31  { 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**.

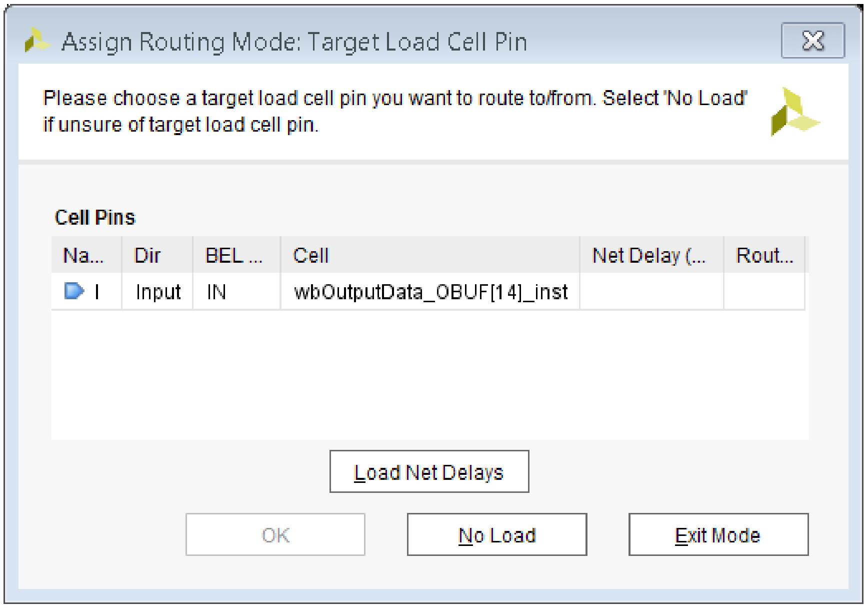
4. Click **Yes** in the Confirm Unroute dialog box.

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

5. Click the **Maximize** button □ 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 Figure 59, to let you select a load pin to route to or from. In this case, only one load pin populates: `wbOutputData_OBUF[14]_inst`.



Please choose a target load cell pin you want to route to/from. Select 'No Load' if unsure of target load cell pin.

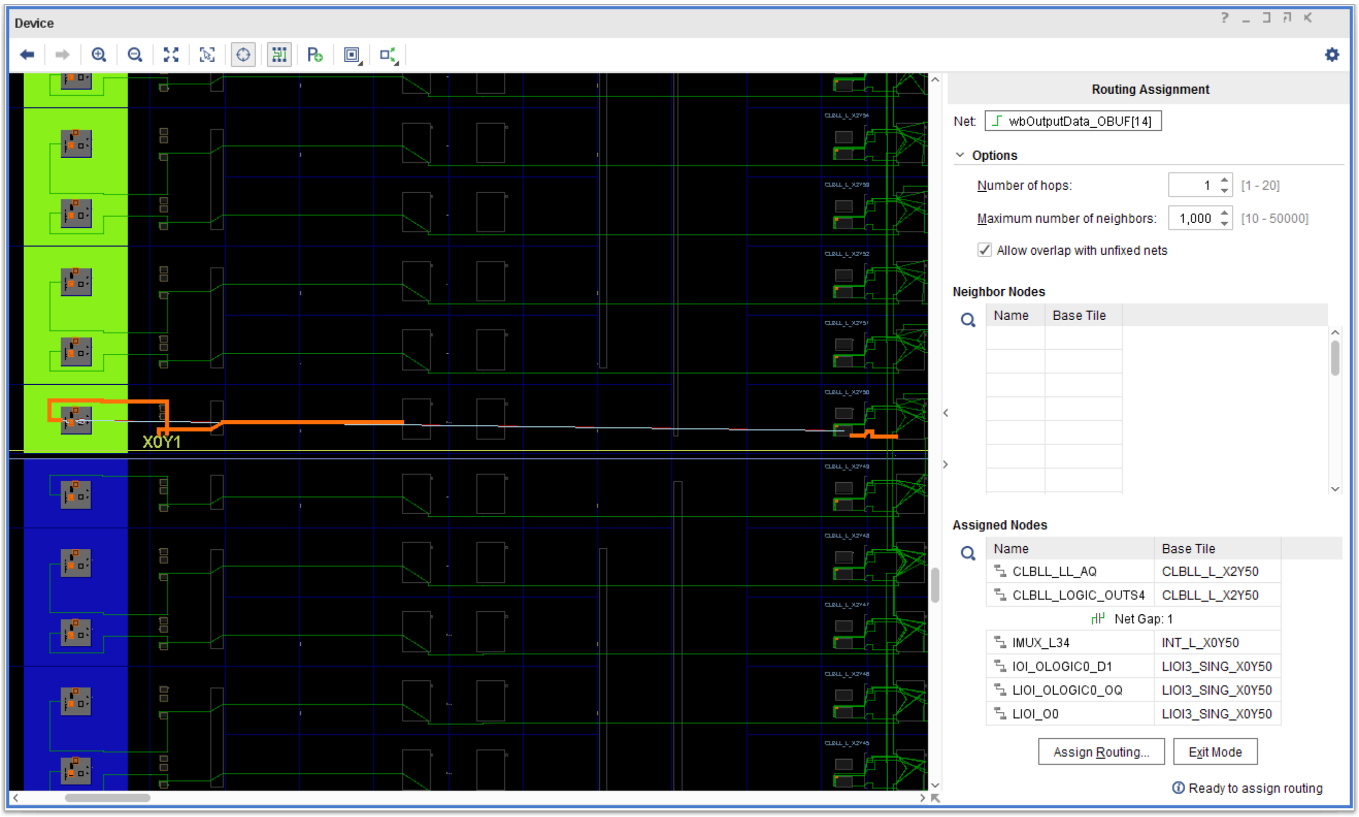
Cell Pins

Na...	Dir	BEL ...	Cell	Net Delay (...	Rout...
I	Input	IN	wbOutputData_OBUF[14]_inst		

Figure 59: Target Load Cell Pin Dialog Box

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.



Routing Assignment

Net: wbOutputData_OBUF[14]

Options

Number of hops: 1 [1 - 20]

Maximum number of neighbors: 1,000 [10 - 50000]

☑ Allow overlap with unfixed nets

Neighbor Nodes

Name	Base Tile

Assigned Nodes

Name	Base Tile
CLBLL_LL_AQ	CLBLL_L_X2Y50
CLBLL_LOGIC_OUTS4	CLBLL_L_X2Y50
Net Gap: 1	
IMUX_L34	INT_L_X0Y50
IOI_OLOGIC0_D1	LIOI3_SING_X0Y50
LIOI_OLOGIC0_OQ	LIOI3_SING_X0Y50
LIOI_O0	LIOI3_SING_X0Y50

ⓘ Ready to assign routing

Figure 60: Assign Routing Mode

The Routing Assignment window includes the following sections, as seen in Figure 60:

- **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.
  - **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.

As you can see in Figure 60, 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_OLOGIC0_D1`, `LIOI_OLOGIC0_OQ`, and `LIOI_O0` 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.

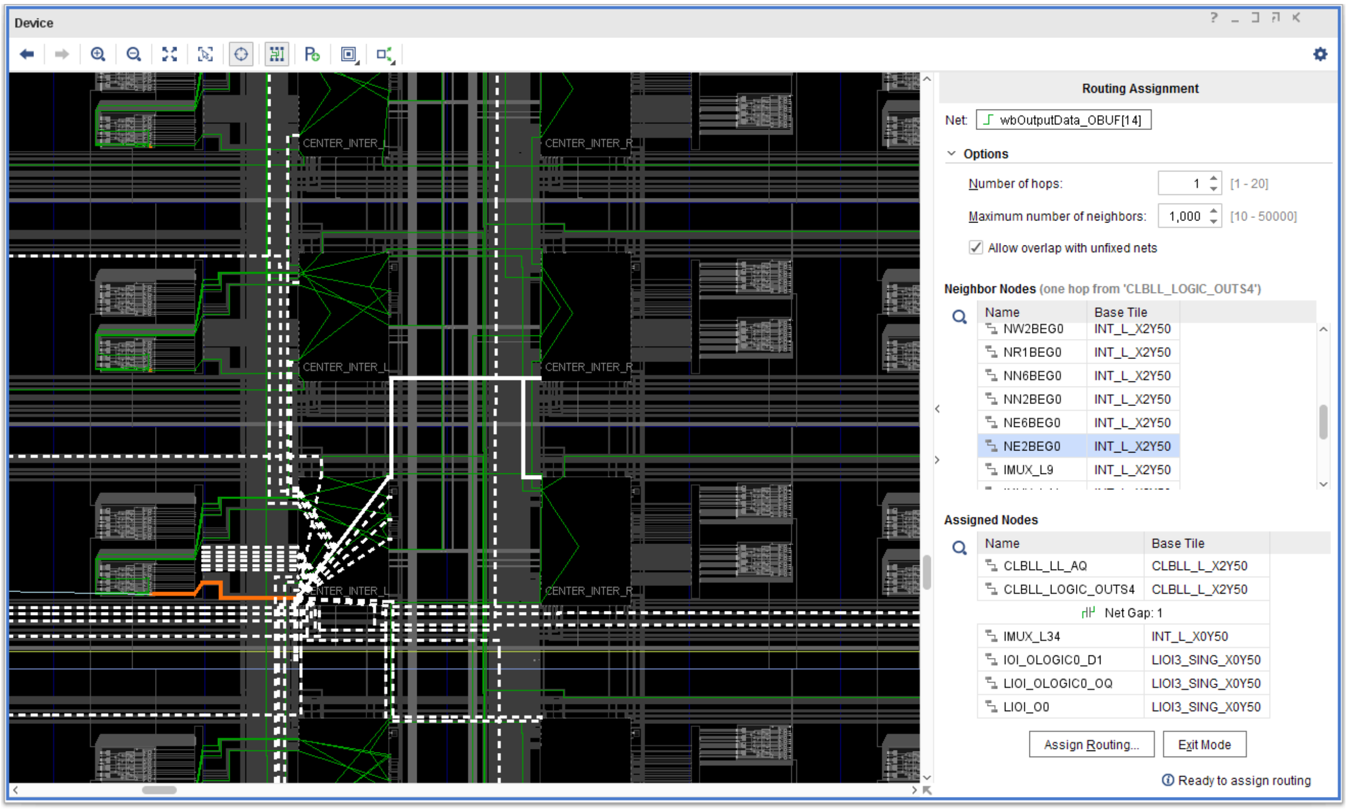
You can route the gap by selecting a node on either side of the gap and then choosing the neighbor node to assign the route to. Selecting the node displays possible neighbor nodes in the Neighbor Nodes section of the Routing Assignment window and appear as dashed white lines in the Device window.

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.



Routing Assignment

Net: wbOutputData_OBUF[14]

Options

Number of hops: 1 [1 - 20]

Maximum number of neighbors: 1,000 [10 - 50000]

☑ Allow overlap with unfixed nets

Neighbor Nodes (one hop from 'CLBLL_LOGIC_OUTS4')

Name	Base Tile
NW2BEG0	INT_L_X2Y50
NR1BEG0	INT_L_X2Y50
NN6BEG0	INT_L_X2Y50
NN2BEG0	INT_L_X2Y50
NE6BEG0	INT_L_X2Y50
NE2BEG0	INT_L_X2Y50
IMUX_L9	INT_L_X2Y50

Assigned Nodes

Name	Base Tile
CLBLL_LL_AQ	CLBLL_L_X2Y50
CLBLL_LOGIC_OUTS4	CLBLL_L_X2Y50
Net Gap: 1	
IMUX_L34	INT_L_X0Y50
IOI_OLOGIC0_D1	LIOI3_SING_X0Y50
LIOI_OLOGIC0_OQ	LIOI3_SING_X0Y50
LIOI_O0	LIOI3_SING_X0Y50

Ready to assign routing

Figure 61: Routing the Gap from CLBLL_LOGIC_OUTS4

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.

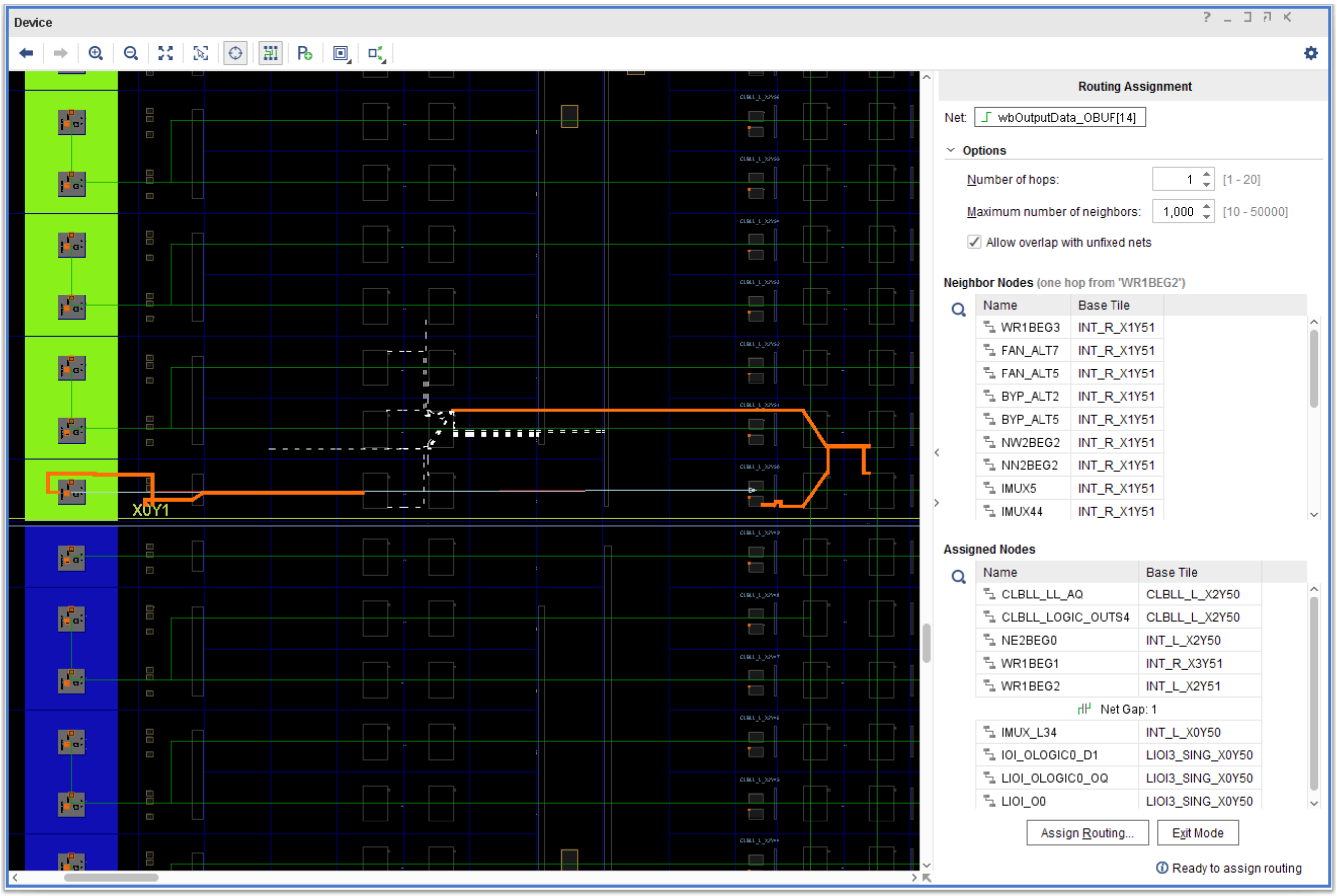
10. In **Neighbor Nodes**, assign node `NE2BEG0` by double-clicking it.

    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, right click, and select* ***Remove*** *on the context menu.*
>
> *You can turn off the Auto Fit Selection ⌖ 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.



Routing Assignment

Net: wbOutputData_OBUF[14]

Options

Number of hops: 1 [1 - 20]

Maximum number of neighbors: 1,000 [10 - 50000]

☑ Allow overlap with unfixed nets

Neighbor Nodes (one hop from 'WR1BEG2')

Name	Base Tile
WR1BEG3	INT_R_X1Y51
FAN_ALT7	INT_R_X1Y51
FAN_ALT5	INT_R_X1Y51
BYP_ALT2	INT_R_X1Y51
BYP_ALT5	INT_R_X1Y51
NW2BEG2	INT_R_X1Y51
NN2BEG2	INT_R_X1Y51
IMUX5	INT_R_X1Y51
IMUX44	INT_R_X1Y51

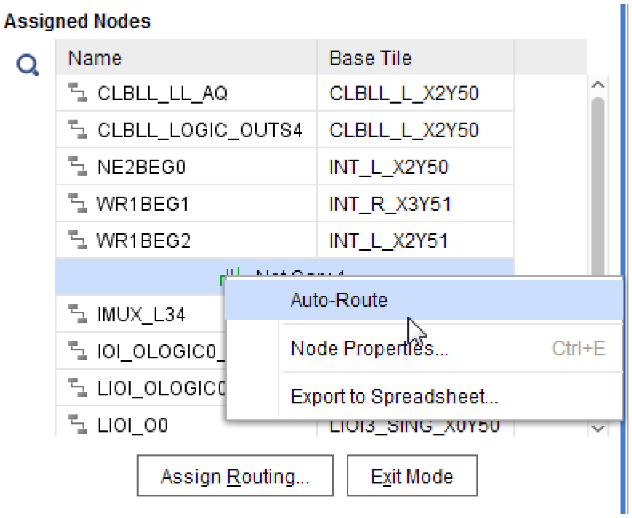
Assigned Nodes

Name	Base Tile
CLBLL_LL_AQ	CLBLL_L_X2Y50
CLBLL_LOGIC_OUTS4	CLBLL_L_X2Y50
NE2BEG0	INT_L_X2Y50
WR1BEG1	INT_R_X3Y51
WR1BEG2	INT_L_X2Y51
Net Gap: 1	
IMUX_L34	INT_L_X0Y50
IOI_OLOGIC0_D1	LIOI3_SING_X0Y50
LIOI_OLOGIC0_OQ	LIOI3_SING_X0Y50
LIOI_O0	LIOI3_SING_X0Y50

Ready to assign routing

Figure 62: Closing the 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.



Assigned Nodes

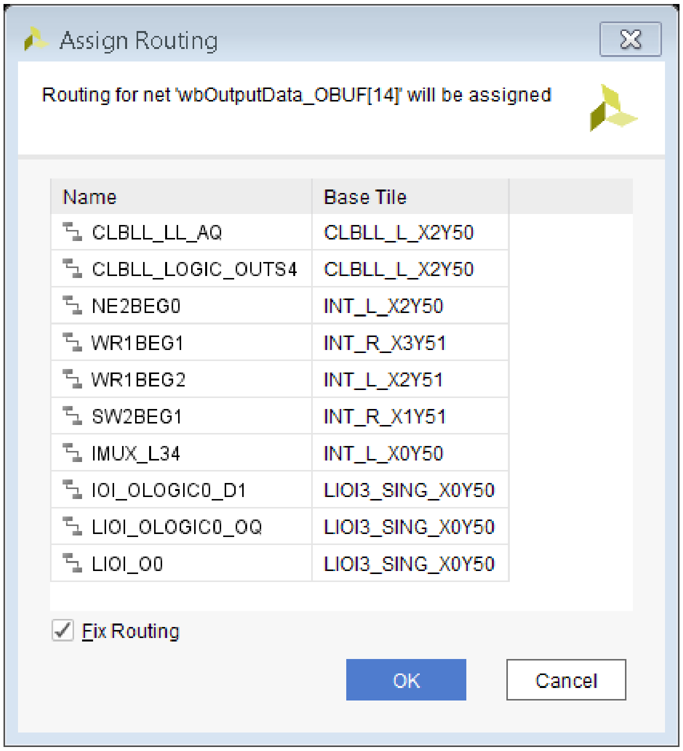
Name	Base Tile
CLBLL_LL_AQ	CLBLL_L_X2Y50
CLBLL_LOGIC_OUTS4	CLBLL_L_X2Y50
NE2BEG0	INT_L_X2Y50
WR1BEG1	INT_R_X3Y51
WR1BEG2	INT_L_X2Y51
Net Gap: 1	
IMUX_L34	
IOI_OLOGIC0_	
LIOI_OLOGIC0	
LIOI_O0	LIOI3_SING_X0Y50

Figure 63: Auto-Route the Gap

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.



Assign Routing

Routing for net 'wbOutputData_OBUF[14]' will be assigned

Name	Base Tile
CLBLL_LL_AQ	CLBLL_L_X2Y50
CLBLL_LOGIC_OUTS4	CLBLL_L_X2Y50
NE2BEG0	INT_L_X2Y50
WR1BEG1	INT_R_X3Y51
WR1BEG2	INT_L_X2Y51
SW2BEG1	INT_R_X1Y51
IMUX_L34	INT_L_X0Y50
IOI_OLOGIC0_D1	LIOI3_SING_X0Y50
LIOI_OLOGIC0_OQ	LIOI3_SING_X0Y50
LIOI_O0	LIOI3_SING_X0Y50

☑ Fix Routing

Figure 64: Assign Routing—Even Nets

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.

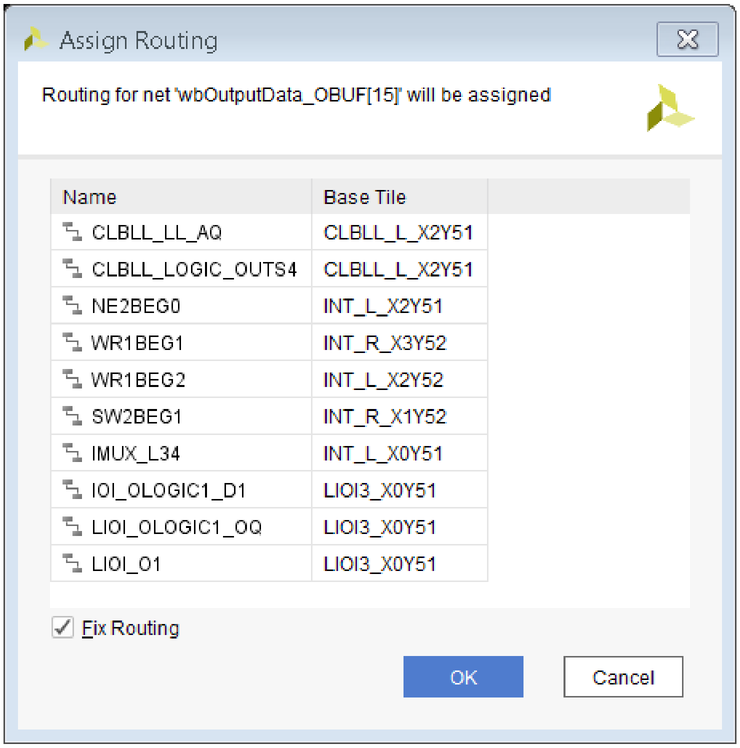
    c. Select the Load Cell Pin.

    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.



Assign Routing

Routing for net 'wbOutputData_OBUF[15]' will be assigned

Name	Base Tile
CLBLL_LL_AQ	CLBLL_L_X2Y51
CLBLL_LOGIC_OUTS4	CLBLL_L_X2Y51
NE2BEG0	INT_L_X2Y51
WR1BEG1	INT_R_X3Y52
WR1BEG2	INT_L_X2Y52
SW2BEG1	INT_R_X1Y52
IMUX_L34	INT_L_X0Y51
IOI_OLOGIC1_D1	LIOI3_X0Y51
LIOI_OLOGIC1_OQ	LIOI3_X0Y51
LIOI_O1	LIOI3_X0Y51

☑ Fix Routing

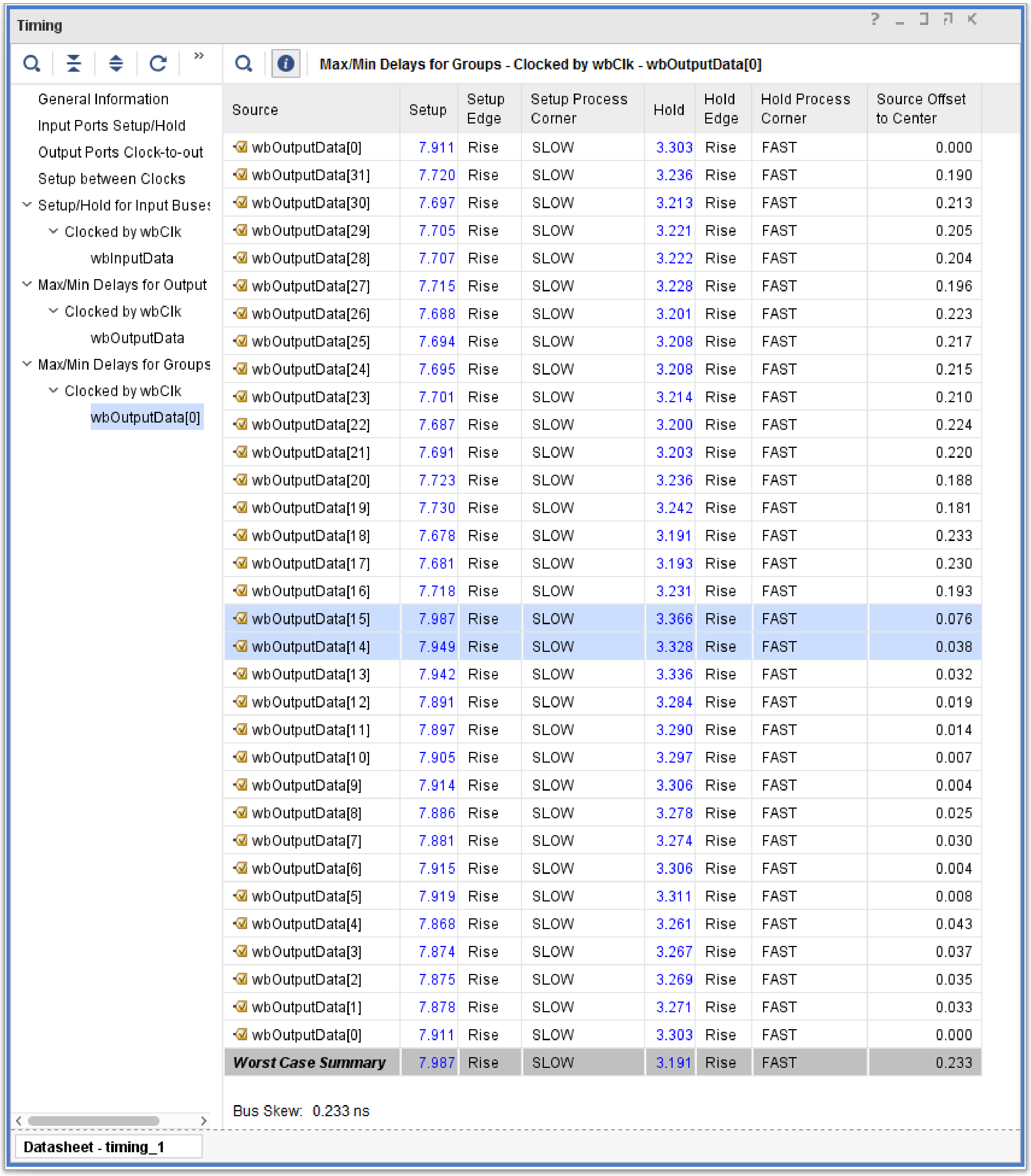
Figure 65: Assign Routing—Odd Nets

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 > Clocked by wbClk > wbOutputData[0]** to display the timing info for the `wbOutputData` bus, as seen in Figure 66.



Timing

Max/Min Delays for Groups - Clocked by wbClk - wbOutputData[0]

General Information
Input Ports Setup/Hold
Output Ports Clock-to-out
Setup between Clocks
Setup/Hold for Input Buses
  Clocked by wbClk
    wbInputData
Max/Min Delays for Output
  Clocked by wbClk
    wbOutputData
Max/Min Delays for Groups
  Clocked by wbClk
    wbOutputData[0]

Source	Setup	Setup Edge	Setup Process Corner	Hold	Hold Edge	Hold Process Corner	Source Offset to Center
wbOutputData[0]	7.911	Rise	SLOW	3.303	Rise	FAST	0.000
wbOutputData[31]	7.720	Rise	SLOW	3.236	Rise	FAST	0.190
wbOutputData[30]	7.697	Rise	SLOW	3.213	Rise	FAST	0.213
wbOutputData[29]	7.705	Rise	SLOW	3.221	Rise	FAST	0.205
wbOutputData[28]	7.707	Rise	SLOW	3.222	Rise	FAST	0.204
wbOutputData[27]	7.715	Rise	SLOW	3.228	Rise	FAST	0.196
wbOutputData[26]	7.688	Rise	SLOW	3.201	Rise	FAST	0.223
wbOutputData[25]	7.694	Rise	SLOW	3.208	Rise	FAST	0.217
wbOutputData[24]	7.695	Rise	SLOW	3.208	Rise	FAST	0.215
wbOutputData[23]	7.701	Rise	SLOW	3.214	Rise	FAST	0.210
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]	7.987	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	7.987	Rise	SLOW	3.191	Rise	FAST	0.233

Bus Skew: 0.233 ns

Datasheet - timing_1

Figure 66: Report Datasheet—Improved Routing

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]]
}
```

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]]
}
```

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]]
}
```

   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 `route_design` command from 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.

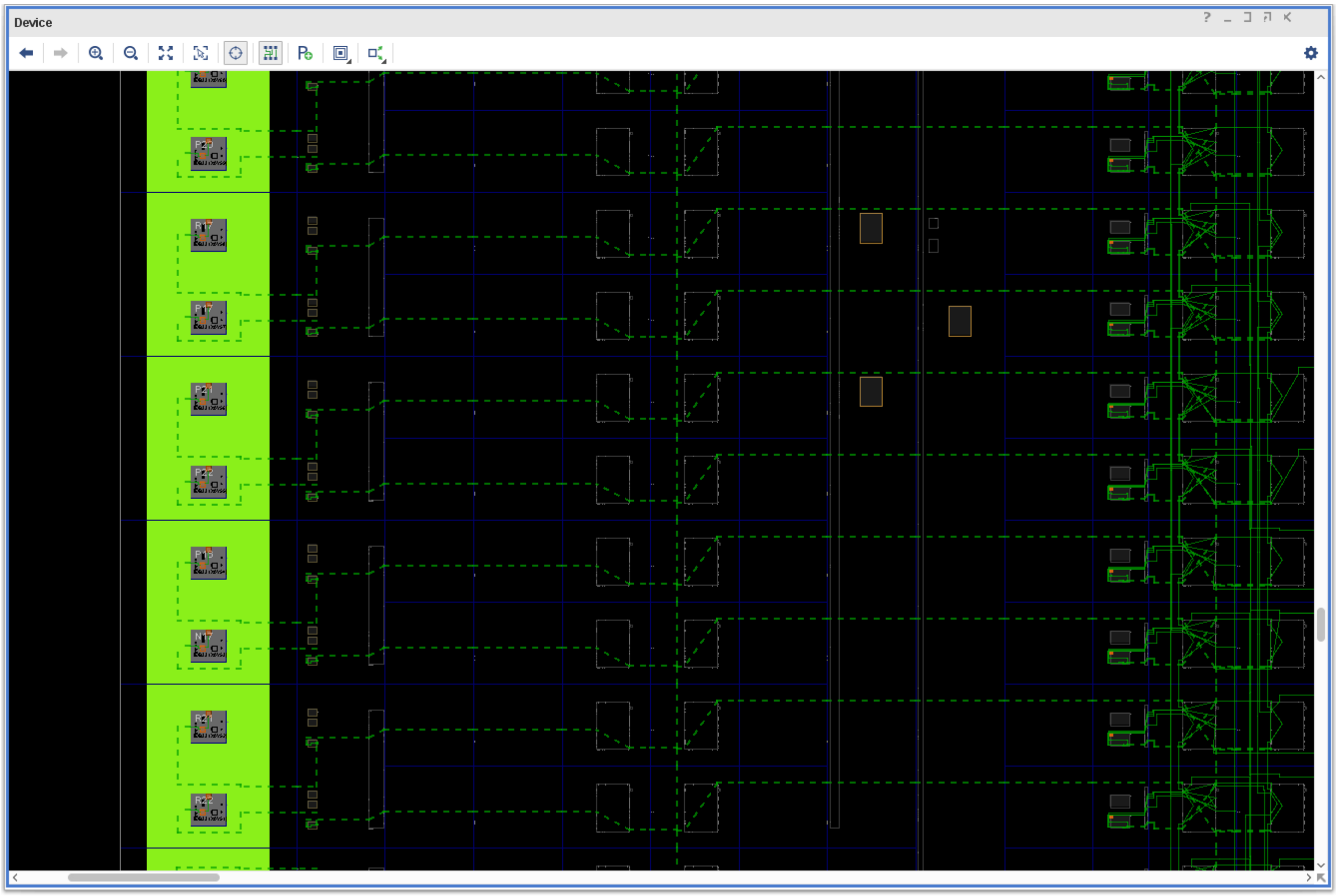


Figure 67: Final Routed Design

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 tab in the Results window area.

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

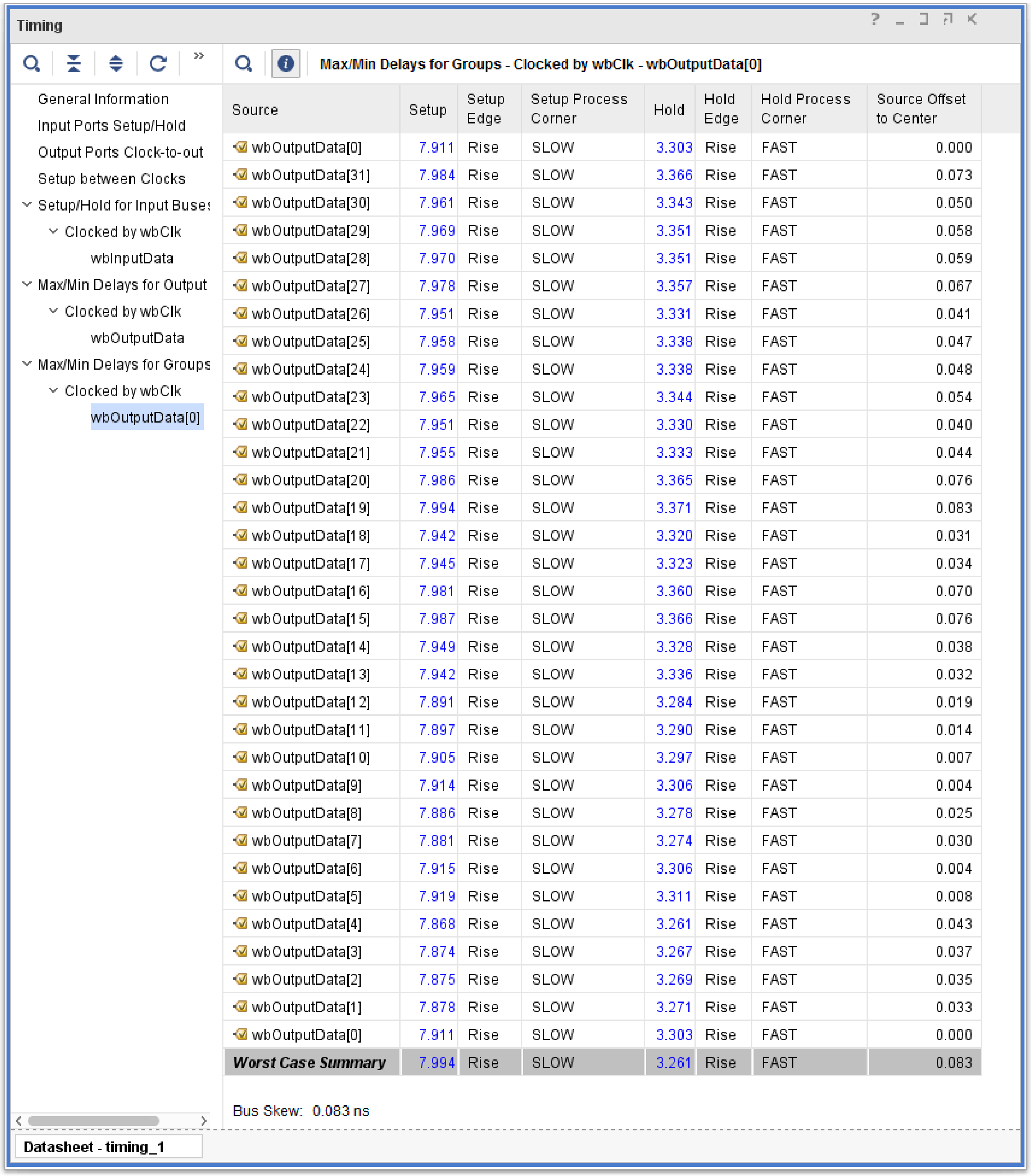
12. Click **rerun** to update the report with the latest timing information.

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

    As shown in Figure 68, 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 > Save Constraints** to save the placement constraints to the target constraint file, `bft_full.xdc`, in the active constraint set, `constrs_2`.



Max/Min Delays for Groups - Clocked by wbClk - wbOutputData[0]

Source	Setup	Setup Edge	Setup Process Corner	Hold	Hold Edge	Hold Process Corner	Source Offset to Center
wbOutputData[0]	7.911	Rise	SLOW	3.303	Rise	FAST	0.000
wbOutputData[31]	7.984	Rise	SLOW	3.366	Rise	FAST	0.073
wbOutputData[30]	7.961	Rise	SLOW	3.343	Rise	FAST	0.050
wbOutputData[29]	7.969	Rise	SLOW	3.351	Rise	FAST	0.058
wbOutputData[28]	7.970	Rise	SLOW	3.351	Rise	FAST	0.059
wbOutputData[27]	7.978	Rise	SLOW	3.357	Rise	FAST	0.067
wbOutputData[26]	7.951	Rise	SLOW	3.331	Rise	FAST	0.041
wbOutputData[25]	7.958	Rise	SLOW	3.338	Rise	FAST	0.047
wbOutputData[24]	7.959	Rise	SLOW	3.338	Rise	FAST	0.048
wbOutputData[23]	7.965	Rise	SLOW	3.344	Rise	FAST	0.054
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

Figure 68: Report Datasheet—Final

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.