**Important Note:** This downloadable PDF of an Answer Record is provided to enhance its usability and readability. It is important to note that Answer Records are Web-based content that are frequently updated as new information becomes available. You are reminded to visit the Xilinx Technical Support Website and review [Xilinx Answer 53786](https://www.xilinx.com) for the latest version of this Answer.

---

**Introduction**

This document illustrates the things a user needs to know to use 7 Series Xilinx Integrated PCI Express Block core v1.8 in Vivado 2012.4. All the steps are illustrated with screenshots without minimal description. The provided screenshots and the captions are self-descriptive. This should help users to get quickly familiar with the tool flow while using 7 Series Xilinx Integrated PCI Express Block core v1.8 in their design.

Along with the core output products generation, simulation and debugging of the hardware using Chipscope have also been described. Users who are familiar with generating the core in Coregen will find this document helpful in quick migration from Coregen to Vivado platform.

**PCIe Core Output Products Generation (Generate Example Design)**

After creating a Vivado project and generating the core as described in PG054, the example design files have to be generated separately by clicking on ‘Generate Output Products’ as show in Figure 1.

![Figure 1 – Generate PCIe Output Products](image-url)
After the example design files have been generated, open the example design project as shown in Figure 3. This opens a separate Vivado project. The example design project location is shown in Figure 5.
PCIe Example Design Hierarchy

Figure 5 – PCIe Core Example Design Vivado Project

Figure 6 – PCIe Example Design Vivado Project GUI

Figure 7 – PCIe IP Example Design Hierarchy (Part 1)
Figure 8 - PCIe IP Example Design Hierarchy (Part-2)

Figure 9 – ‘Collapse All’ option for Project Hierarchy
Figure 10 - ‘Expand All’ option for Project Hierarchy

PCle Example Design Synthesis

Figure 11 – Vivado Synthesis Settings GUI

Figure 12 – Run Synthesis
PCIe Example Design Implementation

Figure 13 - After PCIe Example Design Synthesis

Figure 14 - Vivado Implementation Options GUI
Figure 15 - After PCIe Example Design Implementation

PCle Example Design Bitstream Generation

Figure 16 – After PCIe Example Design Bitstream Generation

Xilinx Answer 53786 - 7-Series Integrated Block for PCI Express in Vivado
PCIe Example Design Implementation Summary and Report

Figure 17 – PCIe Example Design Project Summary after Bitstream Generation

Figure 18 – PCIe Example Design Post-Synthesis Resource Utilization

Figure 19 – Post Implementation ‘Design Runs’ Report
PCIe Example Design Project Directory Structure

Project directory structure in Vivado IP core generation can be confusing. Figure 20 shows the content of the top level project directory after generating the example design files. `example_project` directory shown in Figure 20 is generated only after generating the example design files. Another thing to note is that the content of `project_1.srcs` is different between before and after the example design files generation. This is shown in Figure 22 and Figure 23.

![Figure 20 – PCIe Vivado Project Directory Content after Example Design Generation](image)

![Figure 21 - project_1.data Content](image)
Figure 22 – project_1.srcs Content before PCIe Example Design Generation

Figure 23 - project_1.srcs Content after PCIe Example Design Generation
hierarchy.txt is generated with the generation of the example design files. It contains the entire hierarchy of the PCIe example design files. The content of this file is shown in Figure 24 and Figure 25.

```
xilinx_pcie_2_1_ep_7x
|-- pcie_7x_vl_8_0_pipe_clock (When External Clocking enabled)
  |-- pcie_7x_vl_8_0 (Core Top level module Generated by Vivado in synth directory)
    |-- pcie_7x_vl_8_0_top (Static Top level file)
    |   |-- pcie_7x_vl_8_core_top
    |       |-- pcie_7x_vl_8_0_pcie_top
    |           |-- pcie_7x_vl_8_0_axi_basic_top
    |               |-- pcie_7x_vl_8_0_axi_basic_rx
    |                   |-- pcie_7x_vl_8_0_axi_basic_rx_pipeline
    |                   |-- pcie_7x_vl_8_0_axi_basic_rx_null_gen
    |                   |-- pcie_7x_vl_8_0_axi_basic_tx
    |                   |-- pcie_7x_vl_8_0_axi_basic_tx_pipeline
    |                   |-- pcie_7x_vl_8_0_axi_basic_tx_thrtl_ctl
    |   |-- pcie_7x_vl_8_0_pcie_7x
    |   |   |-- pcie_7x_vl_8_0_pcie_bram_top_7x
    |   |   |   |-- pcie_7x_vl_8_0_pcie_brams_7x (an instance each for Rx & Tx)
    |   |   |   |   |-- pcie_7x_vl_8_0_pcie_bram_7x
    |   |   |   |-- PCIe 21 (Integrated Block Instance)
    |   |   |   |-- pcie_7x_vl_8_0_pcie_pipe_pipeline
    |   |   |       |-- pcie_7x_vl_8_0_pcie_pipe_misc
    |   |   |       |-- pcie_7x_vl_8_0_pcie_pipe_lane (per lane)
    |   |-- pcie_7x_vl_8_0_gt_top
    |       |-- pcie_7x_vl_8_0_pipe_wrapper
    |       |-- pcie_7x_vl_8_0_pipe_clock
    |       |-- pcie_7x_vl_8_0_pipe_reset
    |       |-- pcie_7x_vl_8_0_qpll_reset
    |       |-- pcie_7x_vl_8_0_pipe_user
    |       |-- pcie_7x_vl_8_0_pipe_rate
    |       |-- pcie_7x_vl_8_0_pipe_sync
    |       |-- pcie_7x_vl_8_0_pipe_drp
    |       |-- pcie_7x_vl_8_0_pipe_eq
    |       |   |-- pcie_7x_vl_8_0_rxeq_scan
    |       |   |-- pcie_7x_vl_8_0_qpll_drp
    |       |   |-- pcie_7x_vl_8_0_qpll_wrapper
    |       |   |-- pcie_7x_vl_8_0_gt_wrapper
    |       |   |-- GTXE2_CHANNEL
    |       |       |-- pcie_7x_vl_8_0_qpll_drp.v
    |       |       |-- pcie_7x_vl_8_0_qpll_wrapper.v
```

Figure 24 – PCIe Example Design Files Hierarchy (Part-1)
PCIe Example Design Vivado Simulation

Simulation of PCIe Example Design in Vivado can be done with Vivado Simulator and Modelsim. In this section, steps for simulating the PCIe example design are shown for Vivado Simulator.
Simulation with Vivado Simulator

Figure 27 – Vivado Simulation Panel

Figure 28 – Vivado Simulation Settings

Figure 29 – PCIe Example Design Behavioral Simulation in Vivado
In Figure 29, it shows all ‘simulation’ options are enabled. If the design has not been synthesized yet, only ‘Run Behavioral Simulation’ will be enabled.

After running the simulation, you could select the signals from the ‘objects’ window shown in Figure 32 and drag it to the waveform viewer. Figure 32 shows user-link-up signal in the waveform viewer. This signal indicates that the PCIe link between the Endpoint and the Root Port has come up and the enumeration from root port to the endpoint can be started.
Figure 32 – PCIe ‘user_lnk_up’ Assertion

If you do not save the signals that you selected for monitoring in the waveform viewer, all this will be lost if you re-run the simulation. In order that the same set of selected signals appear on the waveform viewer after re-running the simulation, save your waveform file as shown in Figure 33 and also select this waveform in the ‘view wave’ option as shown in Figure 34.

Figure 33 – Saving Vivado Simulation Waveform
Figure 34 - Open already Saved Vivado Simulation Waveform

Figure 35 shows the PCIe example design simulation in progress. In the working simulation, the `user_link_up` should be asserted and you should see the output on the console as shown in Figure 35.

Figure 35 – PCIe Example Design Simulation in Progress
You could also run the already ran simulation by opening *.wdb file as shown in Figure 36.

![Figure 36 – Open Already Completed Simulation](image)

Vivado has a number of windows. The Vivado GUI allows customizing windows layout by providing a certain layouts specific for Simulation, Floorplanning etc. Figure 37 shows the ‘Simulation Layout’

![Figure 37 - Vivado Simulation Layout](image)
Simulation in Modelsim

When generating the PCIe Example Design as illustrated in the previous section, it comes with an entire simulation setup along with a script to simulate the example design in Modelsim. The location of that script is shown in Figure 38.

```
- project_1
  - project_1.data
  - project_1.srcs
    - sources_1
      - ip
        - PCIe_7X_v1_8_0
          - doc
        - PCIe_7X_v1_8_0
          - example_design
        - simulation
          - dsport
        - functional
          - board.f
          - board.v
          - board_common.vh
          - pipe_interconnect.v
          - simulate_mti.do
          - simulate_ncsim.sh
          - simulate_vcs.sh
          - sys_clk_gen.v
          - sys_clk_gen_ds.v
          - waves_vcs.tcl
          - xilinx_lib_vcs.f
```

Figure 38 – PCIe Example Design Modelsim Simulation Script

Debugging with Chipscope

```
Synchronized Design - xckt325dflp300i-2 (active)
```

Figure 39 – PCIe Example Design Vivado Project GUI after opening the Synthesized Design
The details on how to debug a design using chipscope in Vivado is provided in UG936[1]. This section illustrates how to grab signals for debugging in PCIe example design. For more information, please refer to UG936.

Vivado allows selecting signals for debugging, same as in Chipscope inserter. There is an additional feature where you could search for specific nets, using wild cards, in the whole design. This is shown in Figure 40. To start grabbing signals for chipscope, you should first open the synthesized design as shown in Figure 39.

![Figure 40 – Search ‘nets’ for Probing in Chipscope](image)

![Figure 41 – PCIe Example Design user_link_up Signal](image)
Figure 42 – ‘Mark Debug’ for Probing user_link_up in Chipscope

OK to debug user_link_up net?

This will create MARK_DEBUG constraints, which will be added to the target XDC constraint file when you save the design, causing synthesis to go out of date. To avoid having to rerun synthesis you can CLICK Force-up-to-date.
Figure 43 – user_link_up Net Properties after enabling ‘MARK_DEBUG’

Figure 44 – Saving ‘Mark Debug’ Constraints to the existing XDC file.
Generating Debug Cores (Set up Debug)

Figure 45 – MARK_DEBUG Constraint in PCIe Example Design XDC File

Figure 46 – Generating Debug Cores
Set up Debug

This wizard will guide you through the process of choosing nets and connecting them to debug cores.

The wizard is automatically populated with any selected nets and with nets from the Unassigned Debug Nets folder.

To continue, click Next

Figure 47 – Setup Debug GUI

Figure 48 – Selected Nets for Probing in Chipscope
Figure 49 – Add/Remove nets in Setup Debug

Figure 50 - Set up Debug Summary

Set up Debug Summary

- 0 debug cores will be removed:
- 2 debug cores will be created
- Found 2 clocks

To apply the above changes, click Finish
Figure 51 – Chipscope Debug Cores in Netlist Window

Figure 52 – PCIe Example Design Selected Debug Signals in ‘Debug’ Window
Debug Cores Schematic

Figure 53 - PCIe Example Design Schematic with Debug Cores

Figure 54 – ILA_0 Core

Figure 55 – ILA_1 Core
Adding More Nets for Chipscope Debugging

Figure 56 – Add `cfg_bus_number` for probing in Chipscope

Figure 57 – `MARK_DEBUG` constraints in the XDC file for `cfg_bus_number`
Figure 58 – Setup Debug Options for the added signals

Figure 59 - Setup Debug Options for New Nets
Vivado Design Implementation Strategies

Vivado provides different implementation strategies as shown in Figure 60. A user could try by playing with these implementation strategies if the timing is not met. If the timing is still not met after trying out all the implementation strategies shown, the user might need to implement the design by altering different implementation options.

The 'Design Runs' window shows the result for different implementation strategies as shown in Figure 61.

**PCle Example Design with Debug Cores - Timing Analysis**

This section illustrates techniques and tools for timing analysis in Vivado for the PCIe example design with the debug cores. In this specific test example, there is a hold time violation in one of the paths. The screenshots provided show how to dig in detail information on that particular path.
Figure 62 - Reports Tab

Figure 63 - Invoking Timing Summary Report after Implementation

Figure 64 – Timing Summary Report
Table 1 - Timing Error Example

<table>
<thead>
<tr>
<th>Name</th>
<th>Slack (ns)</th>
<th>Type</th>
<th>Worst Slack (ns)</th>
<th>Total Violation (ns)</th>
<th>Failing Endpoints</th>
<th>Total Endpoints</th>
</tr>
</thead>
<tbody>
<tr>
<td>Path 113</td>
<td>-0.023</td>
<td>Setup</td>
<td>4.278</td>
<td>0.000</td>
<td>0</td>
<td>57</td>
</tr>
<tr>
<td></td>
<td>-0.023</td>
<td>Hold</td>
<td></td>
<td>-0.046</td>
<td>2</td>
<td>57</td>
</tr>
</tbody>
</table>

Figure 65 - Timing Error Example

Figure 66 - Timing Error Quick Summary

Figure 67 - Path Properties for the Failing Path
**Figure 68 – Detailed Timing Report for the Failing Path**

<table>
<thead>
<tr>
<th>Path Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Source</strong></td>
</tr>
<tr>
<td><strong>Path Group</strong></td>
</tr>
<tr>
<td><strong>Path Type</strong></td>
</tr>
<tr>
<td><strong>Data Path Delay</strong></td>
</tr>
<tr>
<td><strong>Clock Path Skew</strong></td>
</tr>
<tr>
<td><strong>Clock Uncertainty</strong></td>
</tr>
</tbody>
</table>

### Summary

<table>
<thead>
<tr>
<th>Delay</th>
<th>Cumulative</th>
<th>Location</th>
<th>Logical Resource</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.000</td>
<td>0.000</td>
<td>Site: GTKEX2_CHANNEL_X017</td>
<td>pipe.7x7v1.0.0 UPLN/ps/psa/psa.pipe_pipe_wrap</td>
</tr>
<tr>
<td>0.344</td>
<td>0.344</td>
<td>Site: BURFCTRL_X016</td>
<td>est_clk.pipe_clock.pipe_TXOUTCLK.OUT</td>
</tr>
<tr>
<td>0.026</td>
<td>0.370</td>
<td>Site: BURFCTRL_X016</td>
<td>est_clk.pipe_clock.pipe_TXOUTCLK.OUT</td>
</tr>
<tr>
<td>0.605</td>
<td>0.975</td>
<td>Site: BURFCTRL_X016</td>
<td>est_clk.pipe_clock.pipe_TXOUTCLK.OUT</td>
</tr>
<tr>
<td>0.574</td>
<td>1.549</td>
<td>Site: BURFCTRL_X016</td>
<td>est_clk.pipe_clock.pipe_TXOUTCLK.OUT</td>
</tr>
<tr>
<td>0.026</td>
<td>1.625</td>
<td>Site: BURFCTRL_X016</td>
<td>est_clk.pipe_clock.pipe_TXOUTCLK.OUT</td>
</tr>
<tr>
<td>0.580</td>
<td>2.205</td>
<td>Site: BURFCTRL_X016</td>
<td>est_clk.pipe_clock.pipe_TXOUTCLK.OUT</td>
</tr>
<tr>
<td>-0.039</td>
<td>2.166</td>
<td>Site: BURFCTRL_X016</td>
<td>est_clk.pipe_clock.pipe_TXOUTCLK.OUT</td>
</tr>
<tr>
<td>0.000</td>
<td>0.000</td>
<td>Site: GTKEX2_CHANNEL_X017</td>
<td>pipe.7x7v1.0.0 UPLN/ps/psa/psa.pipe_pipe_wrap</td>
</tr>
<tr>
<td>0.000</td>
<td>0.000</td>
<td>Site: GTKEX2_CHANNEL_X017</td>
<td>pipe.7x7v1.0.0 UPLN/ps/psa/psa.pipe_pipe_wrap</td>
</tr>
<tr>
<td>0.080</td>
<td>0.160</td>
<td>Site: BURFCTRL_X016</td>
<td>est_clk.pipe_clock.pipe_TXOUTCLK.OUT</td>
</tr>
<tr>
<td>0.318</td>
<td>1.257</td>
<td>Site: BURFCTRL_X016</td>
<td>est_clk.pipe_clock.pipe_TXOUTCLK.OUT</td>
</tr>
<tr>
<td>0.026</td>
<td>1.282</td>
<td>Site: BURFCTRL_X016</td>
<td>est_clk.pipe_clock.pipe_TXOUTCLK.OUT</td>
</tr>
<tr>
<td>0.621</td>
<td>2.847</td>
<td>Site: BURFCTRL_X016</td>
<td>est_clk.pipe_clock.pipe_TXOUTCLK.OUT</td>
</tr>
<tr>
<td>0.030</td>
<td>1.294</td>
<td>Site: BURFCTRL_X016</td>
<td>est_clk.pipe_clock.pipe_TXOUTCLK.OUT</td>
</tr>
<tr>
<td>0.779</td>
<td>2.765</td>
<td>Site: BURFCTRL_X016</td>
<td>est_clk.pipe_clock.pipe_TXOUTCLK.OUT</td>
</tr>
<tr>
<td>-0.254</td>
<td>2.511</td>
<td>Site: BURFCTRL_X016</td>
<td>est_clk.pipe_clock.pipe_TXOUTCLK.OUT</td>
</tr>
<tr>
<td>0.100</td>
<td>2.611</td>
<td>Site: BURFCTRL_X016</td>
<td>est_clk.pipe_clock.pipe_TXOUTCLK.OUT</td>
</tr>
</tbody>
</table>

### Destination Clock Path

<table>
<thead>
<tr>
<th>Delay</th>
<th>Cumulative</th>
<th>Location</th>
<th>Logical Resource</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.000</td>
<td>0.000</td>
<td>Site: GTKEX2_CHANNEL_X017</td>
<td>pipe.7x7v1.0.0 UPLN/ps/psa/psa.pipe_pipe_wrap</td>
</tr>
<tr>
<td>0.000</td>
<td>0.000</td>
<td>Site: GTKEX2_CHANNEL_X017</td>
<td>pipe.7x7v1.0.0 UPLN/ps/psa/psa.pipe_pipe_wrap</td>
</tr>
<tr>
<td>0.000</td>
<td>0.000</td>
<td>Site: GTKEX2_CHANNEL_X017</td>
<td>pipe.7x7v1.0.0 UPLN/ps/psa/psa.pipe_pipe_wrap</td>
</tr>
<tr>
<td>0.080</td>
<td>0.160</td>
<td>Site: BURFCTRL_X016</td>
<td>est_clk.pipe_clock.pipe_TXOUTCLK.OUT</td>
</tr>
<tr>
<td>0.318</td>
<td>1.257</td>
<td>Site: BURFCTRL_X016</td>
<td>est_clk.pipe_clock.pipe_TXOUTCLK.OUT</td>
</tr>
<tr>
<td>0.026</td>
<td>1.282</td>
<td>Site: BURFCTRL_X016</td>
<td>est_clk.pipe_clock.pipe_TXOUTCLK.OUT</td>
</tr>
<tr>
<td>0.621</td>
<td>2.847</td>
<td>Site: BURFCTRL_X016</td>
<td>est_clk.pipe_clock.pipe_TXOUTCLK.OUT</td>
</tr>
<tr>
<td>0.030</td>
<td>1.294</td>
<td>Site: BURFCTRL_X016</td>
<td>est_clk.pipe_clock.pipe_TXOUTCLK.OUT</td>
</tr>
<tr>
<td>0.779</td>
<td>2.765</td>
<td>Site: BURFCTRL_X016</td>
<td>est_clk.pipe_clock.pipe_TXOUTCLK.OUT</td>
</tr>
<tr>
<td>-0.254</td>
<td>2.511</td>
<td>Site: BURFCTRL_X016</td>
<td>est_clk.pipe_clock.pipe_TXOUTCLK.OUT</td>
</tr>
<tr>
<td>0.100</td>
<td>2.611</td>
<td>Site: BURFCTRL_X016</td>
<td>est_clk.pipe_clock.pipe_TXOUTCLK.OUT</td>
</tr>
</tbody>
</table>

---

Xilinx Answer 53786 - 7-Series Integrated Block for PCI Express in Vivado 32
Xilinx Answer 53786 - 7-Series Integrated Block for PCI Express in Vivado

**Figure 69 - Timing Summary for the Failing Path**

**Figure 70 - Schematic Generation for the Failing Path**
PCI Example Design Clock Network Analysis

This section provides features in Vivado that could be used for PCIe example design clock network analysis. The screenshots provided show the clock networks in the design, list of clock nets and the clocking resources usage such as BUFG, BUFR etc.

**Implementation**
- Implementation Settings
- Run Implementation
- Implemented Design
  - Edit Timing Constraints
  - Report Timing Summary
  - Report Clock Networks
  - Report Clock Interaction
  - Report DRC
  - Report Noise
  - Report Utilization
  - Report Power

**Program and Debug**
- Bitstream Settings

**Figure 72 – Creating Clock Networks Report**
Figure 73 – Clock Networks in PCIe Example Design

Figure 74 – Generate Clock PCIe Example Design Clock Utilization Report

Figure 75 – PCIe Example Design Clock Resource Usage and Clock Nets
References

[1]. UG936, Vivado Design Suite Tutorial, Programming and Debugging
[2]. PG054, 7 Series Integrated PCI Express Block core
[3]. UG939, Vivado Design Suite Tutorial, Designing with IP