Building a RIFFA 2.0 design with Vivado: Xilinx Virtex-7 VC707

Matt Jacobsen (mdjacobs@cs.ucsd.edu)

This is a step by step guide to building a RIFFA 2.0 reference design for a Xilinx Virtex-7 VC707 development board using Vivado. Though it is likely that this guide will work for other 7 Series based FPGA development boards.

RIFFA 2.0 provides a simple to use interface for communicating between a workstation and FPGA cores. It uses a Xilinx PCIe Endpoint IP core to drive the transceivers. The PCIe Endpoint core for 7 Series FPGAs is the *7 Series Integrated Block for PCI Express*. This core is licensed by the Xilinx End User License Agreement and is provided with the Xilinx Vivado Design suite with no additional charge. A prebuilt design is provided and ready for download to your Xilinx VC707 board. Building your own RIFFA 2.0 design requires generating the PCIe Endpoint core and then merging it with the RIFFA 2.0 source HDL.

To create a RIFFA 2.0 design with Vivado:
1. Create a project in Xilinx Vivado.
2. Use Vivado to generate the PCIe Endpoint core.
3. Add the RIFFA 2.0 HDL as design sources.
4. Synthesize and implement.

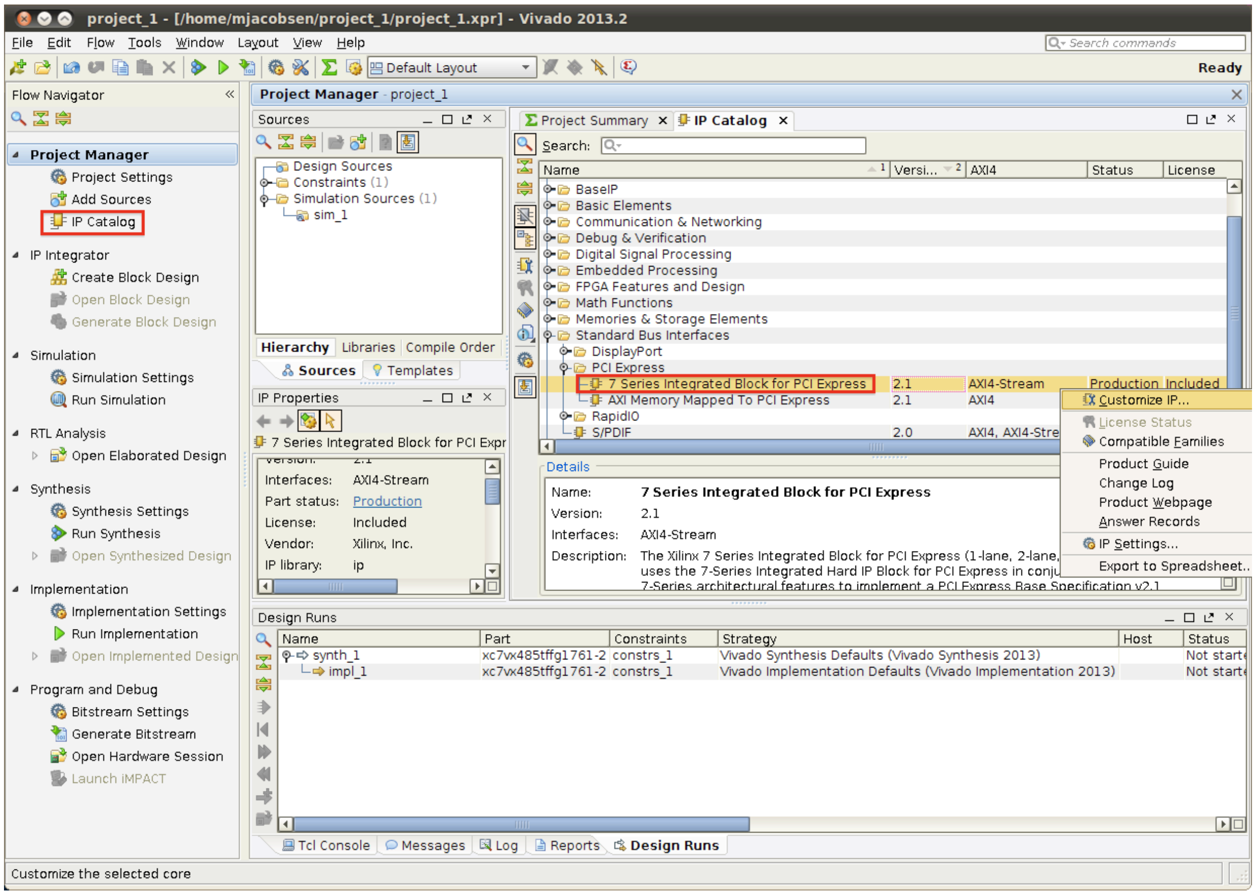
Detailed instructions on how to do each step follow.

1. Create a project in Xilinx Vivado.

I expect you know how to create a new RTL based project in Vivado for your development board. So I won't provide step by step instructions.

2. Use Vivado to generate the PCIe Endpoint core.

Select IP Catalog and double click on the *7 Series Integrated Block for PCI Express* core. Version 2.1 is the latest production version of the core at the time of this writing.



Version:	2.1
Interfaces:	AXI4-Stream
Part status:	Production
License:	Included
Vendor:	Xilinx, Inc.
IP library:	ip

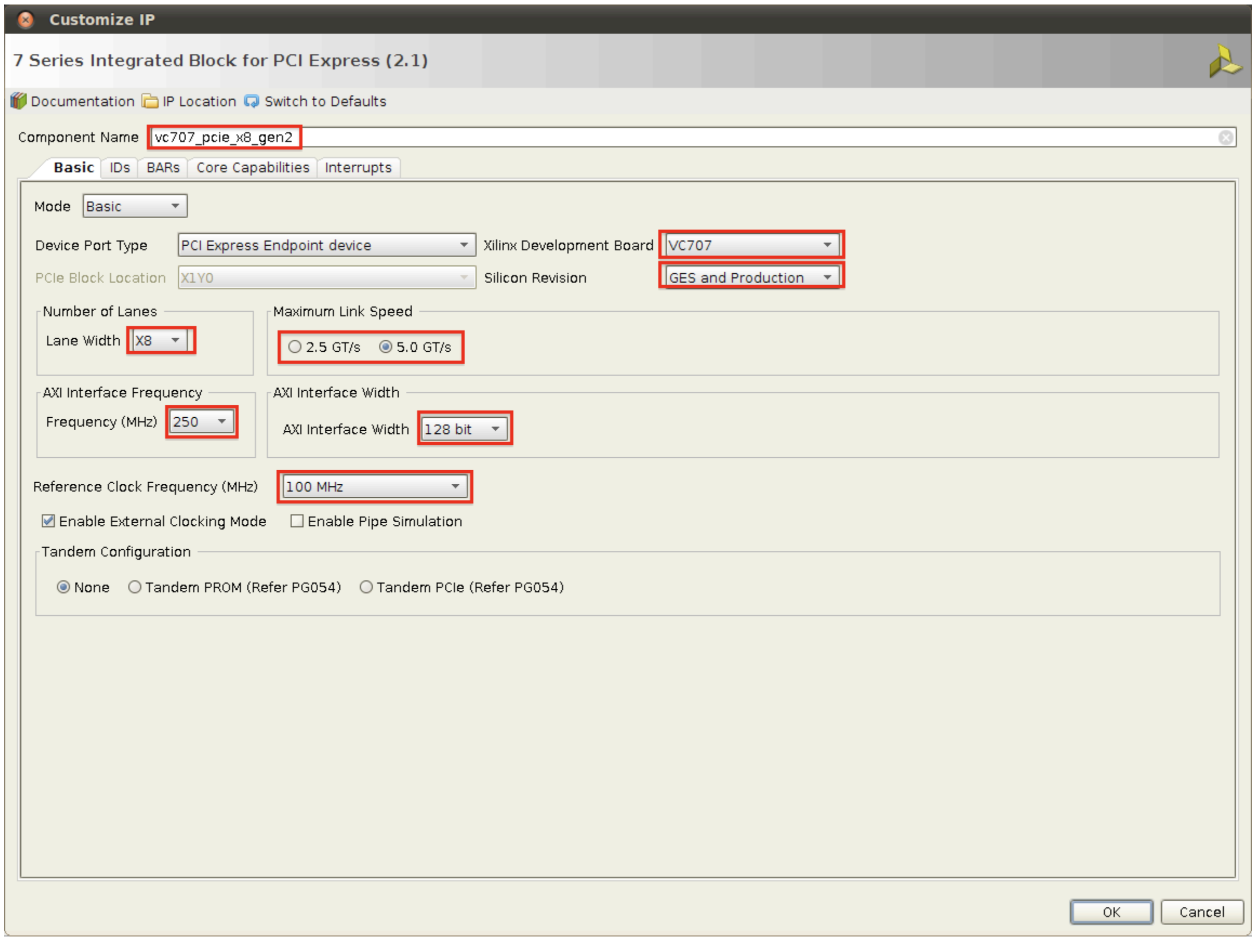
Name:	7 Series Integrated Block for PCI Express
Version:	2.1
Interfaces:	AXI4-Stream
Description:	The Xilinx 7 Series Integrated Block for PCI Express (1-lane, 2-lane, uses the 7-Series Integrated Hard IP Block for PCI Express in conju 7-Series architectural features to implement a PCI Express Base Specification v2.1

Name	Part	Constraints	Strategy	Host	Status
synth_1	xc7vx485tffg1761-2	constrs_1	Vivado Synthesis Defaults (Vivado Synthesis 2013)		Not starte
impl_1	xc7vx485tffg1761-2	constrs_1	Vivado Implementation Defaults (Vivado Implementation 2013)		Not starte

This will begin customization of the IP. Unless otherwise described, the default values on each wizard screen should be left as they are presented.

On the first screen, set the desired lane width and link speed. RIFFA 2.0 supports a 32, 64, and 128 bit interface. So any lane width and link speed selection you make will be supported. You can use any interface frequency the options allow. The reference design expects the component name specified below. Leave the reference clock frequency set to 100 MHz. Set the Xilinx Development Board to VC707 and set Silicon Revision to GES and Production. Below are maximum theoretical bandwidths for PCIe 1.0 and PCIe 2.0 (for reference):

Gen1 (2.5 GT/s):	Gen2 (5.0 GT/s):
x1 = 250 MB/s	x1 = 500 MB/s
x2 = 500 MB/s	x2 = 1000 MB/s
x4 = 1000 MB/s	x4 = 2000 MB/s
x8 = 2000 MB/s	x8 = 4000 MB/s



Component Name: vc707_pcie_x8_gen2

Mode: Basic

Device Port Type: PCI Express Endpoint device

Xilinx Development Board: VC707

PCIe Block Location: X1Y0

Silicon Revision: GES and Production

Number of Lanes — Lane Width: X8

Maximum Link Speed: 2.5 GT/s [✓ 5.0 GT/s]

AXI Interface Frequency — Frequency (MHz): 250

AXI Interface Width — AXI Interface Width: 128 bit

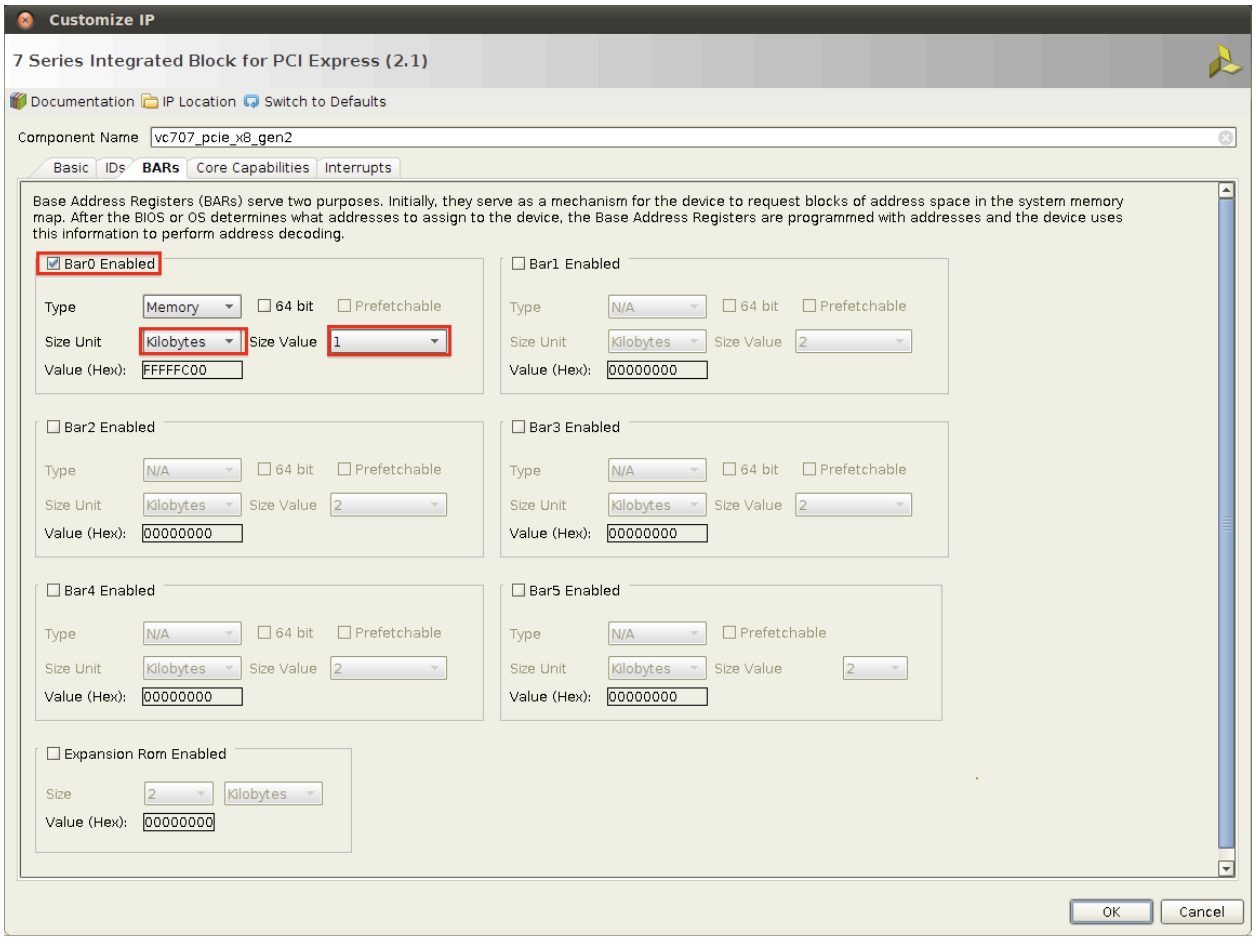
Reference Clock Frequency (MHz): 100 MHz

[✓] Enable External Clocking Mode

[ ] Enable Pipe Simulation

Tandem Configuration: [✓ None] Tandem PROM (Refer PG054) Tandem PCIe (Refer PG054)

On this screen, make sure only Bar0 is selected and is set to a size of 1 KB.


Customize IP

7 Series Integrated Block for PCI Express (2.1)

Documentation | IP Location | Switch to Defaults

Component Name: vc707_pcie_x8_gen2

Basic | IDs | BARs | Core Capabilities | Interrupts

Base Address Registers (BARs) serve two purposes. Initially, they serve as a mechanism for the device to request blocks of address space in the system memory map. After the BIOS or OS determines what addresses to assign to the device, the Base Address Registers are programmed with addresses and the device uses this information to perform address decoding.

☑ Bar0 Enabled
Type: Memory ☐ 64 bit ☐ Prefetchable
Size Unit: Kilobytes Size Value: 1
Value (Hex): FFFFFC00

☐ Bar1 Enabled
Type: N/A ☐ 64 bit ☐ Prefetchable
Size Unit: Kilobytes Size Value: 2
Value (Hex): 00000000

☐ Bar2 Enabled
Type: N/A ☐ 64 bit ☐ Prefetchable
Size Unit: Kilobytes Size Value: 2
Value (Hex): 00000000

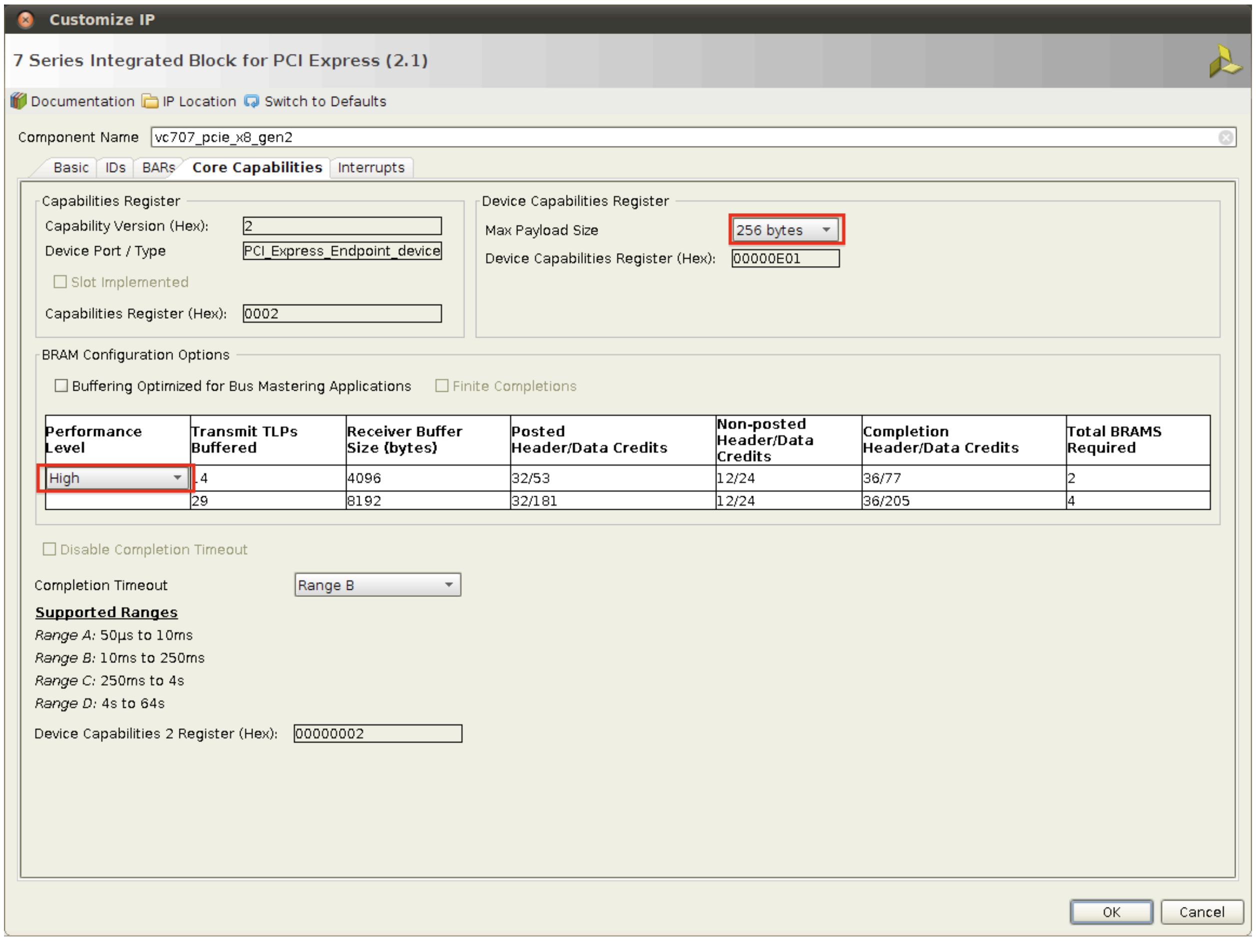
☐ Bar3 Enabled
Type: N/A ☐ 64 bit ☐ Prefetchable
Size Unit: Kilobytes Size Value: 2
Value (Hex): 00000000

☐ Bar4 Enabled
Type: N/A ☐ 64 bit ☐ Prefetchable
Size Unit: Kilobytes Size Value: 2
Value (Hex): 00000000

☐ Bar5 Enabled
Type: N/A ☐ Prefetchable
Size Unit: Kilobytes Size Value: 2
Value (Hex): 00000000

☐ Expansion Rom Enabled
Size: 2 Kilobytes
Value (Hex): 00000000

On this screen, set Performance Level to High. Additionally, set the Max Payload Size to the maximum value offered. These changes are not necessary for RIFFA 2.0 to function. They are required to achieve maximum performance.


Customize IP

7 Series Integrated Block for PCI Express (2.1)

Documentation | IP Location | Switch to Defaults

Component Name: vc707_pcie_x8_gen2

Tabs: Basic | IDs | BARs | Core Capabilities | Interrupts

Capabilities Register
- Capability Version (Hex): 2
- Device Port / Type: PCI_Express_Endpoint_device
- Slot Implemented: ☐
- Capabilities Register (Hex): 0002

Device Capabilities Register
- Max Payload Size: 256 bytes
- Device Capabilities Register (Hex): 00000E01

BRAM Configuration Options
- ☐ Buffering Optimized for Bus Mastering Applications
- ☐ Finite Completions

Performance Level	Transmit TLPs Buffered	Receiver Buffer Size (bytes)	Posted Header/Data Credits	Non-posted Header/Data Credits	Completion Header/Data Credits	Total BRAMS Required
High	14	4096	32/53	12/24	36/77	2
	29	8192	32/181	12/24	36/205	4

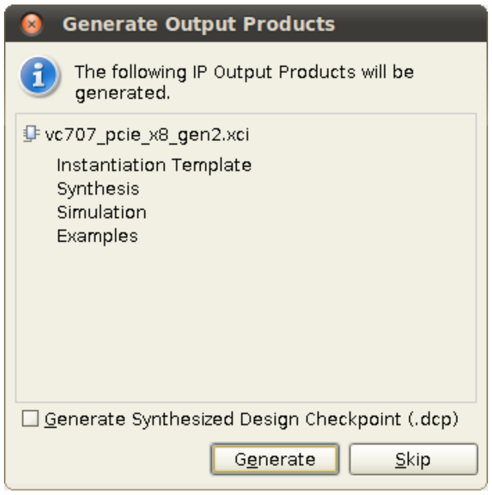
☐ Disable Completion Timeout

Completion Timeout: Range B

Supported Ranges
Range A: 50µs to 10ms
Range B: 10ms to 250ms
Range C: 250ms to 4s
Range D: 4s to 64s

Device Capabilities 2 Register (Hex): 00000002

Then complete the wizard and generate the core. When prompted with the following dialog, click Generate.

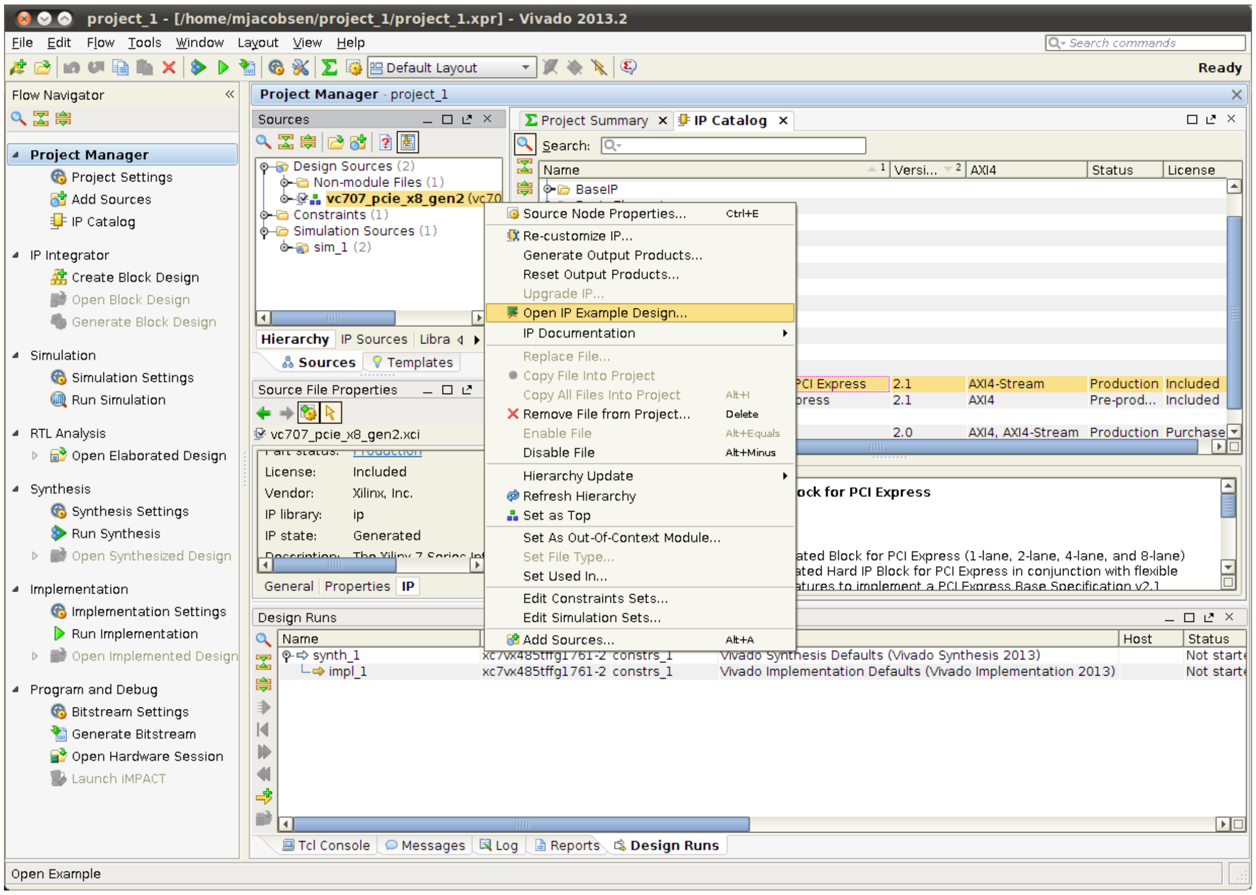

Generate Output Products

The following IP Output Products will be generated.

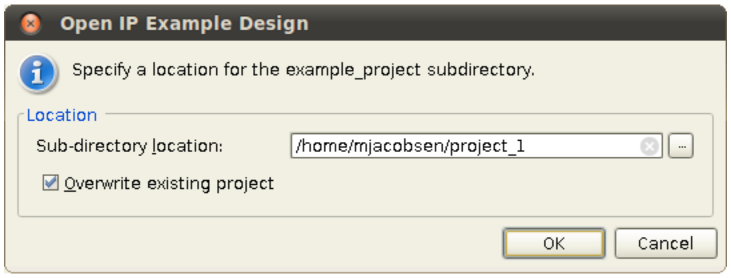
vc707_pcie_x8_gen2.xci
- Instantiation Template
- Synthesis
- Simulation
- Examples

☐ Generate Synthesized Design Checkpoint (.dcp)

In Vivado you must open the IP Example Design before you can use the PCIe Endpoint. This will create a new Vivado project with the PCIe Endpoint Example Design and open another Vivado instance with this new project.


project_1 - [/home/mjacobsen/project_1/project_1.xpr] - Vivado 2013.2

When prompted, specify a location of your choosing for the new Vivado project.


Open IP Example Design

Specify a location for the example_project subdirectory.

Location

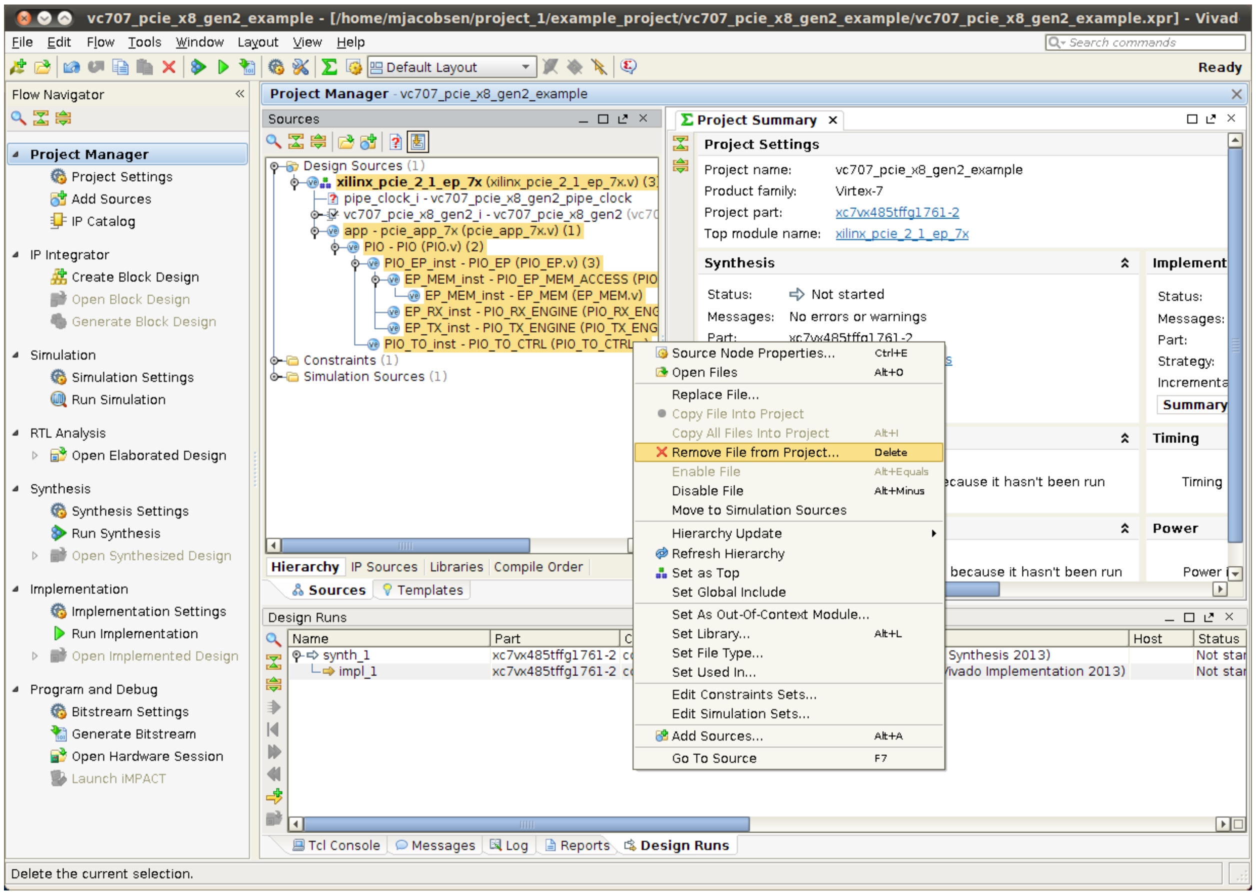
Sub-directory location: /home/mjacobsen/project_1

☑ Overwrite existing project

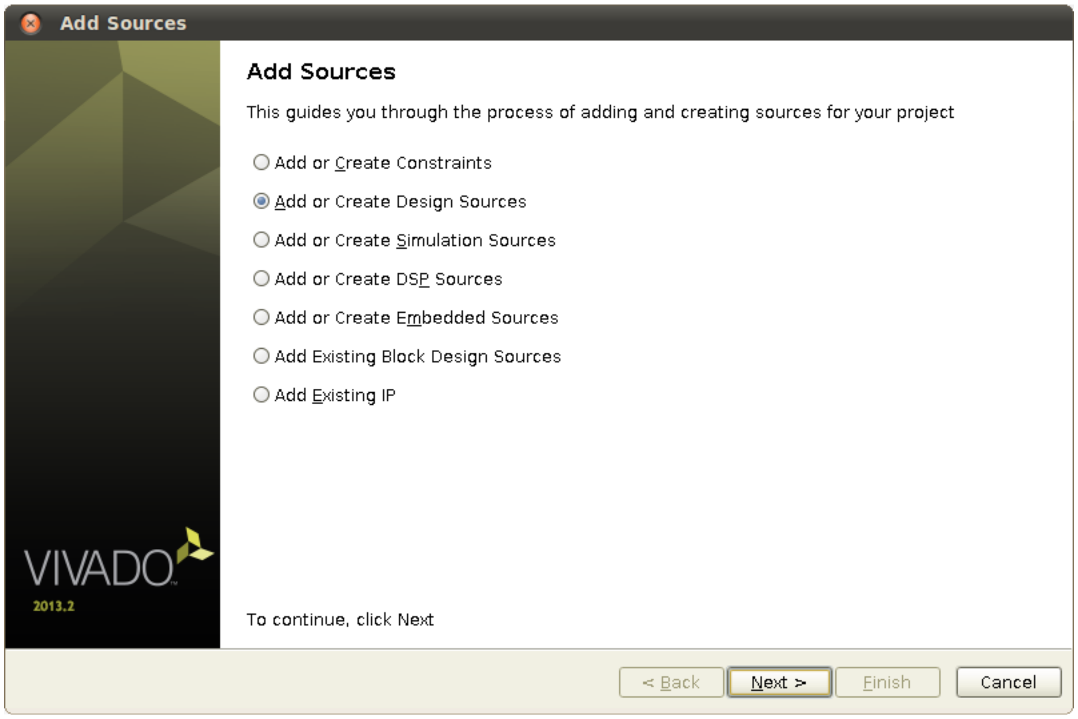
3. Add the RIFFA 2.0 HDL as design sources.

The Example Application Vivado project will be your design project. Before adding the RIFFA 2.0 HDL files, you need to remove the existing example application HDL files. Select the following files and remove them from the project:

```
xilinx_pcie_2_1_ep_7x.v
pcie_app_7x.v
PIO.v
PIO_EP.v
PIO_EP_MEM_ACCESS.v
EP_MEM.v
PIO_RX_ENGINE.v
PIO_TO_CTRL.v
PIO_TX_ENGINE.v
```


vc707_pcie_x8_gen2_example - [/home/mjacobsen/project_1/example_project/vc707_pcie_x8_gen2_example/vc707_pcie_x8_gen2_example.xpr] - Vivad
File Edit Flow Tools Window Layout View Help
Default Layout
Ready
Flow Navigator
Project Manager
Project Settings
Add Sources
IP Catalog
IP Integrator
Create Block Design
Open Block Design
Generate Block Design
Simulation
Simulation Settings
Run Simulation
RTL Analysis
Open Elaborated Design
Synthesis
Synthesis Settings
Run Synthesis
Open Synthesized Design
Implementation
Implementation Settings
Run Implementation
Open Implemented Design
Program and Debug
Bitstream Settings
Generate Bitstream
Open Hardware Session
Launch iMPACT
Project Manager - vc707_pcie_x8_gen2_example
Sources
Design Sources (1)
xilinx_pcie_2_1_ep_7x (xilinx_pcie_2_1_ep_7x.v) (3)
pipe_clock_i - vc707_pcie_x8_gen2_pipe_clock
vc707_pcie_x8_gen2_i - vc707_pcie_x8_gen2 (vc7
app - pcie_app_7x (pcie_app_7x.v) (1)
PIO - PIO (PIO.v) (2)
PIO_EP_inst - PIO_EP (PIO_EP.v) (3)
EP_MEM_inst - PIO_EP_MEM_ACCESS (PIO
EP_MEM_inst - EP_MEM (EP_MEM.v)
EP_RX_inst - PIO_RX_ENGINE (PIO_RX_ENG
EP_TX_inst - PIO_TX_ENGINE (PIO_TX_ENG
PIO_TO_inst - PIO_TO_CTRL (PIO_TO_CTRL
Constraints (1)
Simulation Sources (1)
Hierarchy IP Sources Libraries Compile Order
Sources Templates
Project Summary
Project Settings
Project name: vc707_pcie_x8_gen2_example
Product family: Virtex-7
Project part: xc7vx485tffg1761-2
Top module name: xilinx_pcie_2_1_ep_7x
Synthesis
Status: Not started
Messages: No errors or warnings
Part: xc7vx485tffg1761-2
Implement
Status:
Messages:
Part:
Strategy:
Incrementa
Summary
Timing
ecause it hasn't been run
Timing
Power
because it hasn't been run
Power
Source Node Properties... Ctrl+E
Open Files Alt+O
Replace File...
Copy File Into Project Alt+I
Copy All Files Into Project Alt+I
Remove File from Project... Delete
Enable File Alt+Equals
Disable File Alt+Minus
Move to Simulation Sources
Hierarchy Update
Refresh Hierarchy
Set as Top
Set Global Include
Set As Out-Of-Context Module...
Set Library... Alt+L
Set File Type...
Set Used In...
Edit Constraints Sets...
Edit Simulation Sets...
Add Sources... Alt+A
Go To Source F7
Design Runs
Name Part C Host Status
synth_1 xc7vx485tffg1761-2 Synthesis 2013) Not star
impl_1 xc7vx485tffg1761-2 Vivado Implementation 2013) Not star
Tcl Console Messages Log Reports Design Runs
Delete the current selection.

Then add the RIFFA 2.0 HDL files to your project.



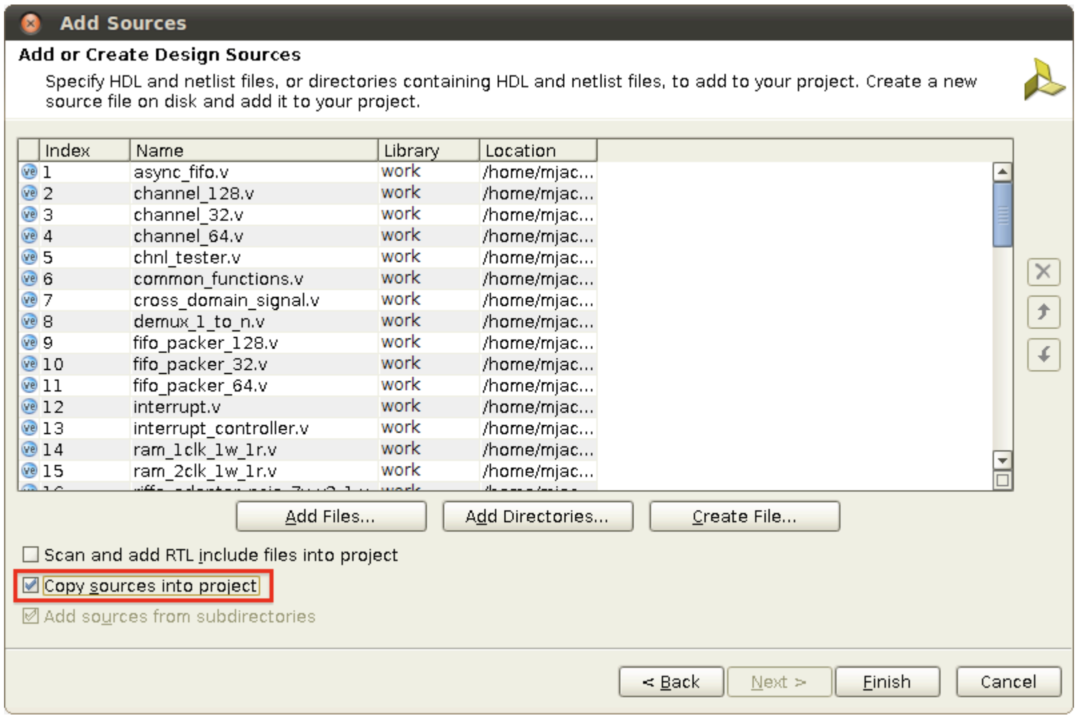
Add Sources

This guides you through the process of adding and creating sources for your project

- ○ Add or Create Constraints
- ● Add or Create Design Sources
- ○ Add or Create Simulation Sources
- ○ Add or Create DSP Sources
- ○ Add or Create Embedded Sources
- ○ Add Existing Block Design Sources
- ○ Add Existing IP

To continue, click Next

Be sure to add the `riffa_top_pcie_7x_v2_1.v` and `riffa_adapter_pcie_7x_v2_1.v` files from the board directory as well. Once all the files are added, be sure Copy sources into project is checked and click Finish.



Add or Create Design Sources

Specify HDL and netlist files, or directories containing HDL and netlist files, to add to your project. Create a new source file on disk and add it to your project.

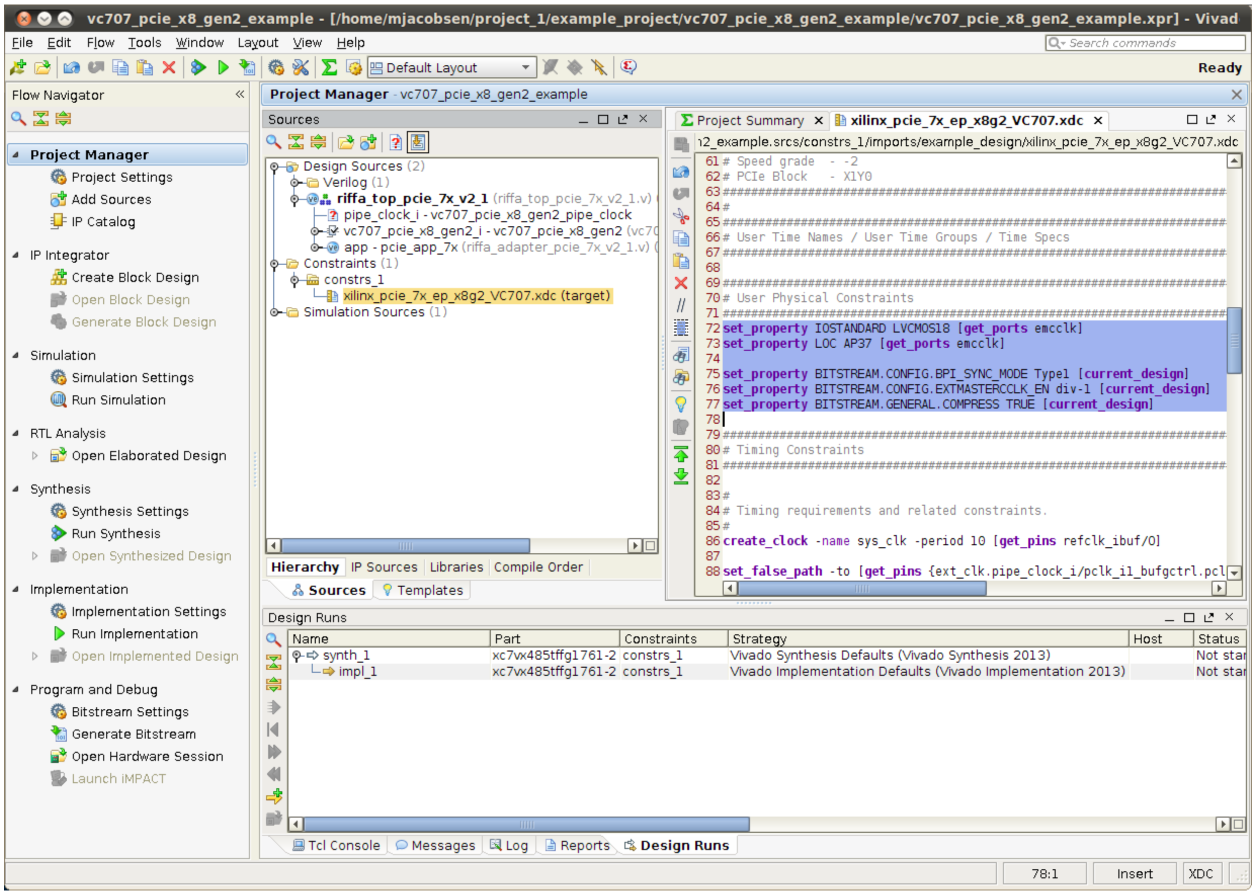
Index	Name	Library	Location
1	async_fifo.v	work	/home/mjac...
2	channel_128.v	work	/home/mjac...
3	channel_32.v	work	/home/mjac...
4	channel_64.v	work	/home/mjac...
5	chnl_tester.v	work	/home/mjac...
6	common_functions.v	work	/home/mjac...
7	cross_domain_signal.v	work	/home/mjac...
8	demux_1_to_n.v	work	/home/mjac...
9	fifo_packer_128.v	work	/home/mjac...
10	fifo_packer_32.v	work	/home/mjac...
11	fifo_packer_64.v	work	/home/mjac...
12	interrupt.v	work	/home/mjac...
13	interrupt_controller.v	work	/home/mjac...
14	ram_1clk_1w_1r.v	work	/home/mjac...
15	ram_2clk_1w_1r.v	work	/home/mjac...

- ☐ Scan and add RTL include files into project
- ☑ Copy sources into project
- ☑ Add sources from subdirectories

4. Synthesize and implement.

Before you can synthesize, you need to make modifications to the `.xdc` constraints file as per XTP207. Add the following constraints:

```
  set_property IOSTANDARD LVCMOS18 [get_ports emcclk]
  set_property LOC AP37 [get_ports emcclk]
  set_property BITSTREAM.CONFIG.BPI_SYNC_MODE Type1 [current_design]
  set_property BITSTREAM.CONFIG.EXTMASTERCCLK_EN div-1 [current_design]
  set_property BITSTREAM.GENERAL.COMPRESS TRUE [current_design]
```



```
# Speed grade   - -2
# PCIe Block    - X1Y0
##############################################################################
#
##############################################################################
# User Time Names / User Time Groups / Time Specs
##############################################################################

##############################################################################
# User Physical Constraints
##############################################################################
set_property IOSTANDARD LVCMOS18 [get_ports emcclk]
set_property LOC AP37 [get_ports emcclk]

set_property BITSTREAM.CONFIG.BPI_SYNC_MODE Type1 [current_design]
set_property BITSTREAM.CONFIG.EXTMASTERCCLK_EN div-1 [current_design]
set_property BITSTREAM.GENERAL.COMPRESS TRUE [current_design]

##############################################################################
# Timing Constraints
##############################################################################

#
# Timing requirements and related constraints.
#
create_clock -name sys_clk -period 10 [get_pins refclk_ibuf/O]

set_false_path -to [get_pins {ext_clk.pipe_clock_i/pclk_i1_bufgctrl.pcl
```

Design Runs

Name	Part	Constraints	Strategy	Host	Status
synth_1	xc7vx485tffg1761-2	constrs_1	Vivado Synthesis Defaults (Vivado Synthesis 2013)		Not star
impl_1	xc7vx485tffg1761-2	constrs_1	Vivado Implementation Defaults (Vivado Implementation 2013)		Not star

At this point, if you attempt to synthesize you'll encounter an error: `C_NUM_CHNL` is not defined. This is intentional. It is done to get you to open the RIFFA 2.0 adapter module file and edit it as needed. There are instructions in that file. However, all you need to do is uncomment the line that defines the `C_NUM_CHNL` parameter, set it to the number of channels you need (1-12), and you should be able to implement the design completely.

The adapter module instantiates a `chnl_tester` module for each channel. Sample user application software in the RIFFA 2.0 distribution can be used to send and receive data to/from the `chnl_tester` modules. The `chnl_tester` is meant to be an example. Your design will need to replace the `chnl_tester` modules with your own modules. You may also need to modify the `.xdc`, top level, and RIFFA adapter modules to bring in additional signals as dictated by your design.

That's it for the HDL design. See the RIFFA website to setup the driver and get started with software programming.