

Using the IP Catalog and IP Integrator

Introduction

In this lab you will use the IP Catalog to generate a clock resource. You will instantiate the generated clock core in the provided waveform generator design. You will also use IP Integrator to generate a FIFO core and then use it in the HDL design.

Objectives

After completing this lab, you will be able to:

- Include an IP in the project during the project creation
- Use IP Catalog to generate a clocking core
- Instantiate the generated clock
- Create a block design using IP Integrator
- Instantiate the block design
- Generate bitstream and verify the functionality in hardware

Procedure

This lab is broken into steps that consist of general overview statements providing information on the detailed instructions that follow. Follow these detailed instructions to progress through the lab.

Design Description

The design used in this lab is a programmable waveform generator, also known as a signal generator.

The waveform generator in this design is intended to be a “standalone” device that is controlled via a PC (or other terminal device) using RS-232 serial communication. The design described here implements the RS-232 communication channel, the waveform generator and connection to the external DAC, and a simple parser to implement a small number of “commands” to control the waveform generation.

The wave generator implements a look-up table of 1024 samples of 16 bits each in a RAM. The wave generator also implements three variables:

- `nsamp`: The number of samples to use for the output waveform. Must be between 1 and 1024.
- `prescale`: The prescaler for the sample clock. Must be 32 or greater.
- `speed`: The speed (or rate) for the output samples in units of the prescaled clock.

The wave generator can be instructed to send the appropriate number of samples once, cycling from 0 to `nsamp-1` once and then stopping, or continuously, where it continuously loops the `nsamp` samples. When enabled, either once or continuously, the wave generator will send one sample to the DAC every (`prescale x speed`) `clk_tx` clock cycles. The contents of the RAM, as well as the three variables, can be changed via commands sent over the RS-232 link, as can the mode of the wave generator. The wave generator will generate responses for all commands.

There are three clock domains within this design: `clk_rx`, `clk_tx`, and `clk_samp`. The clock generator module instantiates all the clocking resources required for generating these three clocks. All three clocks are derived from a single clock input, coming in on `clk_pin`. The frequency of the clock input depends on the oscillator available on the target board; for the ZedBoard it is 100MHz and the Zybo it is 125 MHz.

The block diagram is as shown in **Figure 1**.

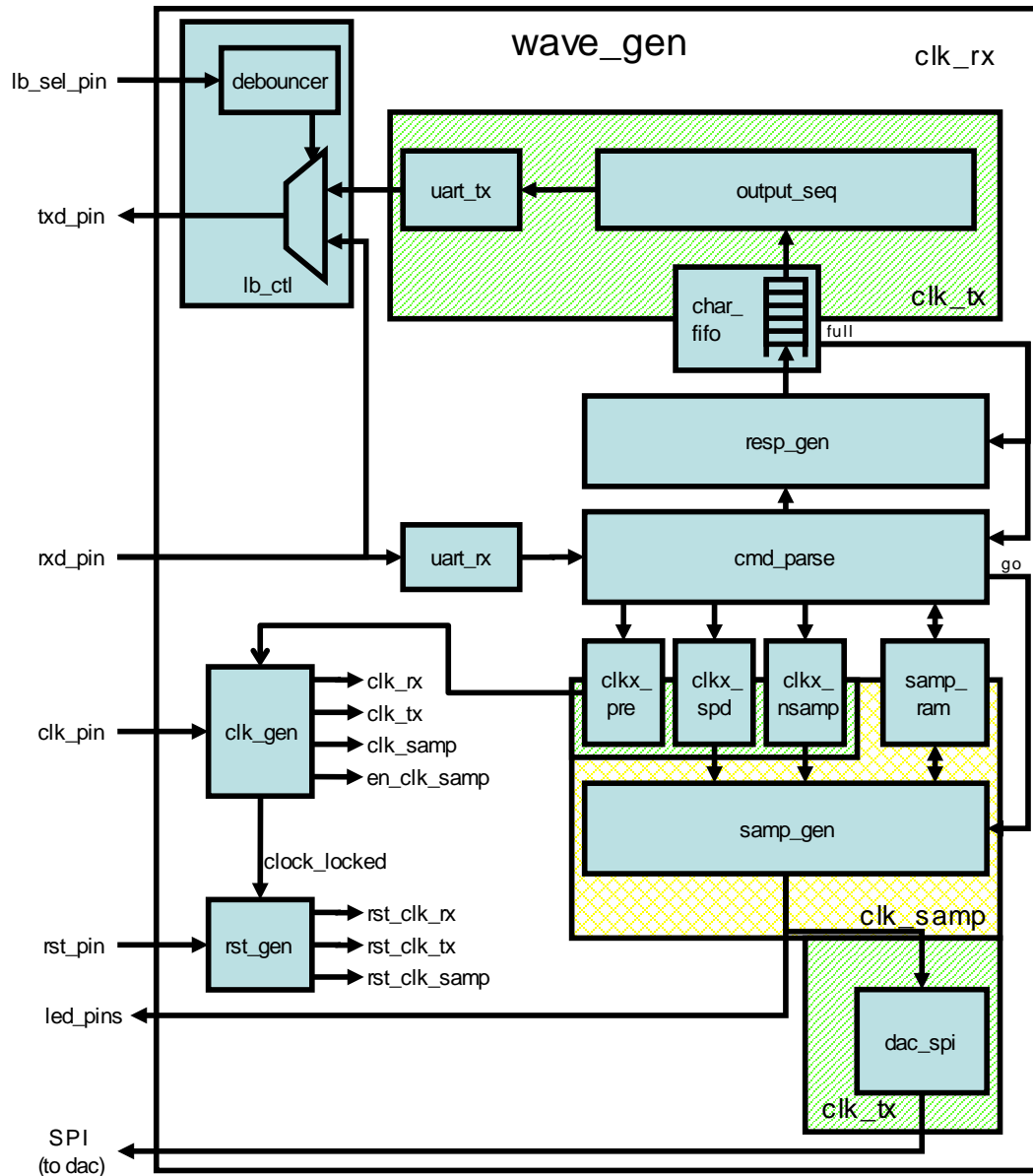
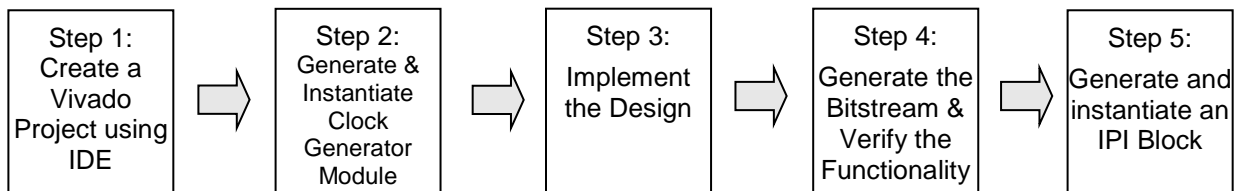


Figure 1. The design

General Flow



Create a Vivado Project using IDE

Step 1

- 1-1. **Launch Vivado and create a project targeting the XC7Z020CLG484-1 device for the ZedBoard or the XC7Z010CLG400-1 device for the Zybo and using the Verilog HDL. Use the provided Verilog source files, a device specific ip, and XDC files from the <2014_2_zynq_sources>\<board>\lab4 subdirectory.**

References to <2014_2_zynq_labs> is a placeholder for the c:\xup\fpga_flow\2014_2_zynq_labs directory and <2014_2_zynq_sources> is a placeholder for the c:\xup\fpga_flow\2014_2_zynq_sources directory.

Reference to <board> means either the **ZedBoard** or the **Zybo**.

- 1-1-1. Open Vivado by selecting **Start > All Programs > Xilinx Design Tools > Vivado 2014.2 > Vivado 2014.2**
- 1-1-2. Click **Create New Project** to start the wizard. You will see *Create A New Vivado Project* dialog box. Click **Next**.
- 1-1-3. Click the Browse button of the *Project location* field of the **New Project** form, browse to <2014_2_zynq_labs>, and click **Select**.
- 1-1-4. Enter **lab4** in the *Project name* field. Make sure that the *Create Project Subdirectory* box is checked. Click **Next**.
- 1-1-5. Select **RTL Project** option in the *Project Type* form, and click **Next**.
- 1-1-6. Select **Verilog** as the *Target Language* in the *Add Sources* form. Do the same for the Simulator Language option.
- 1-1-7. Click on the **Add Files...** button, browse to the <2014_2_zynq_sources>\<board>\lab4 directory, where <board> contains the set of source files specific to either the ZedBoard or the Zybo. Select all the Verilog files, click **OK**. Make sure all the sources are copied into the project and then click **Next**.
- 1-1-8. In the *Add Existing IP* form, click on the **Add Directories...** button, browse to the <2014_2_zynq_sources>\<board>\lab4 directory. Select the **ip** subdirectory. Click **Select**.
- 1-1-9. Make sure the sources are copied into the project and click **Next** to get to the *Add Constraints* form.
- 1-1-10. You will see timing and pinout XDC files have automatically been included. Make sure the sources are copied into the project and click **Next**.
- 1-1-11. In the *Default Part* form, using the **Parts** option and various drop-down fields of the **Filter** section, select the **XC7Z020CLG484-1** for the ZedBoard or the **XC7Z010CLG400-1** for the Zybo. Click **Next**.

1-1-12. Click **Finish** to create the Vivado project.

1-2. Correct the errors by adding a file.

1-2-1. You will notice eight *Critical Error Files* are being highlighted in the **Sources** pane.

If you check the messages Tab, you will see that these errors are due to missing files.

1-2-2. Click on **Add Sources** in the *Flow Navigator* pane.

1-2-3. Select *Add or Create Design Sources* and click **Next**.

1-2-4. Click on the **Add Files...** button and browse to `<2014_2_zynq_sources>\<board>\lab4`.

1-2-5. In the *File Type* field, select **All Files**, and then select `clomb2.txt` file.

1-2-6. Click **OK**. Make sure the source file is copied into the project and then click **Finish**.

The error messages should go away.

1-2-7. In the *Sources* pane, expand *Design Sources* and *wave_gen* (if necessary), and double-click on the `clk_gen_i0` entry.

Scroll down the file and notice that around line 79 there is an instruction to instantiate a clock core.

Generate and Instantiate Clock Generator Module

Step 2

2-1. Launch the clocking wizard from the Vivado IP Catalog and generate the clock core with input frequency of 100.00 MHz for the ZedBoard or 125.00 MHz for the Zybo and two output clocks of 100.000 MHz each.

2-1-1. Click on **IP Catalog** in the *Flow Navigator* pane.

The IP Catalog will open in the auxiliary pane.

2-1-2. Expand the **FPGA Features and Design > Clocking** sub-folders and double-click on the **Clocking Wizard** entry.

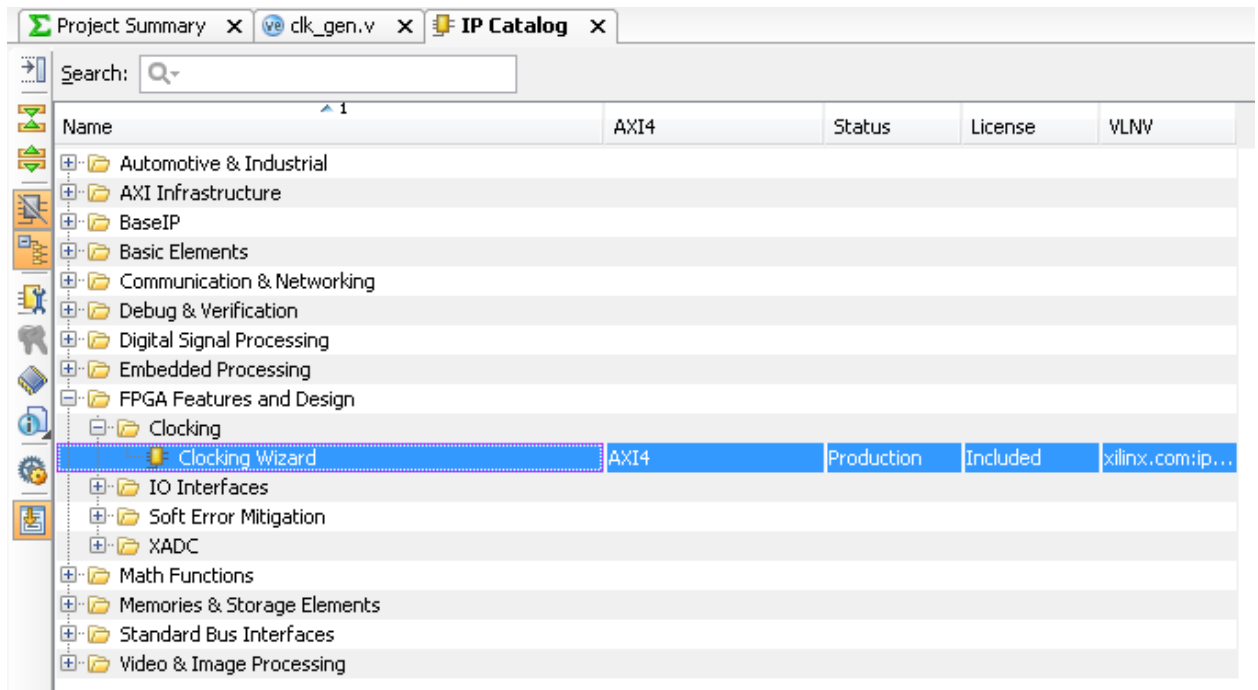


Figure 2. Accessing the Clocking Wizard

The clocking wizard will open.

- 2-1-3. Change the core name to **clk_core**. Refer to the figure below and Make sure that the *Primary* input clock frequency is **100.000** MHz for the ZedBoard or **125.00** MHz for the Zybo and the primitive used is the **MMCM**.

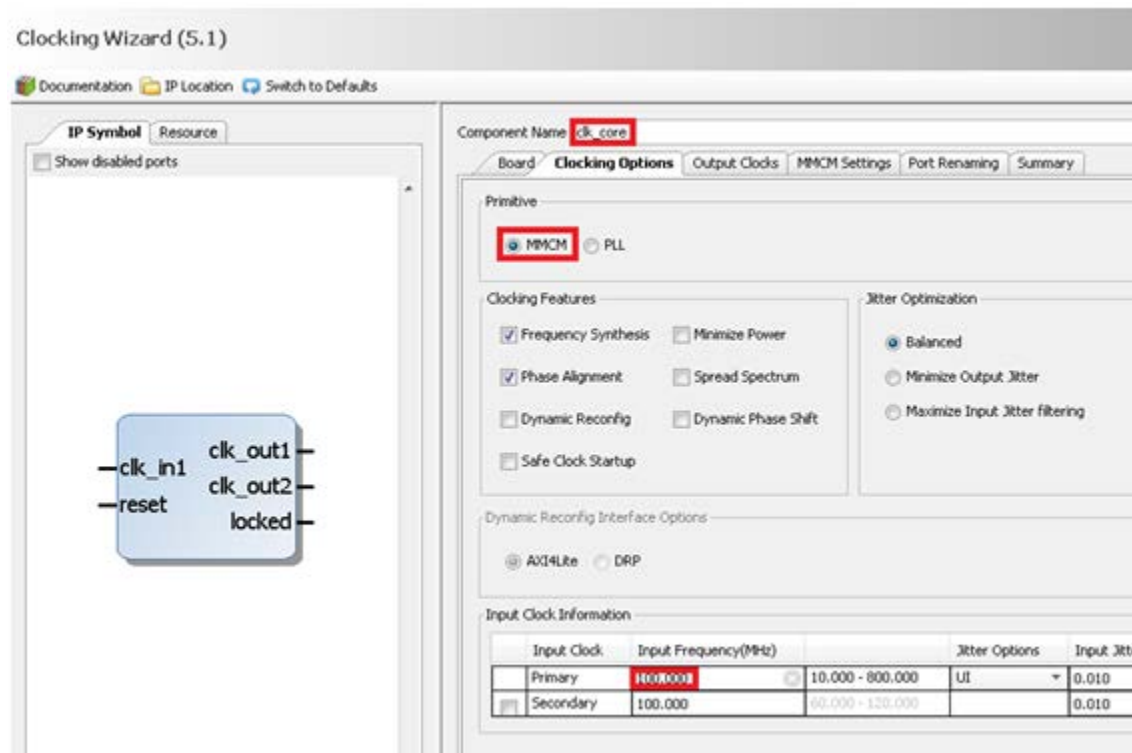


Figure 3. The Clocking Wizard for the ZedBoard

Clocking Wizard (5.1)

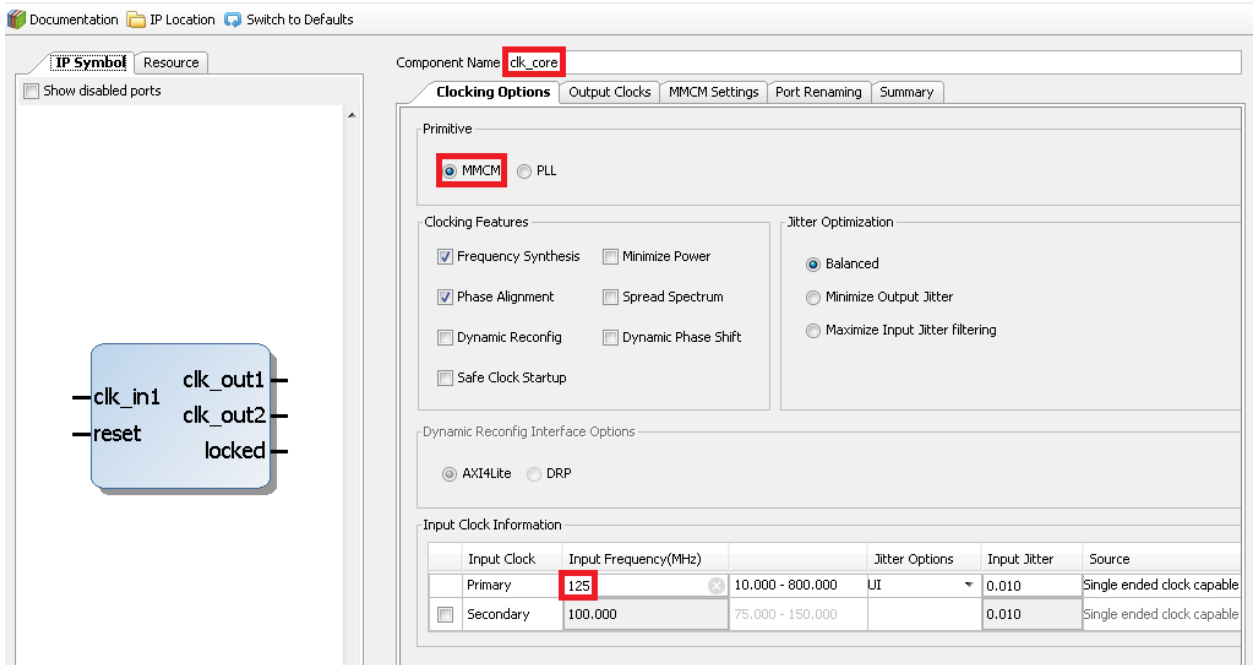


Figure 3. The Clocking Wizard for the Zybo

2-1-4. Select the Output Clocks tab. Click on the check box to enable the second clock output.

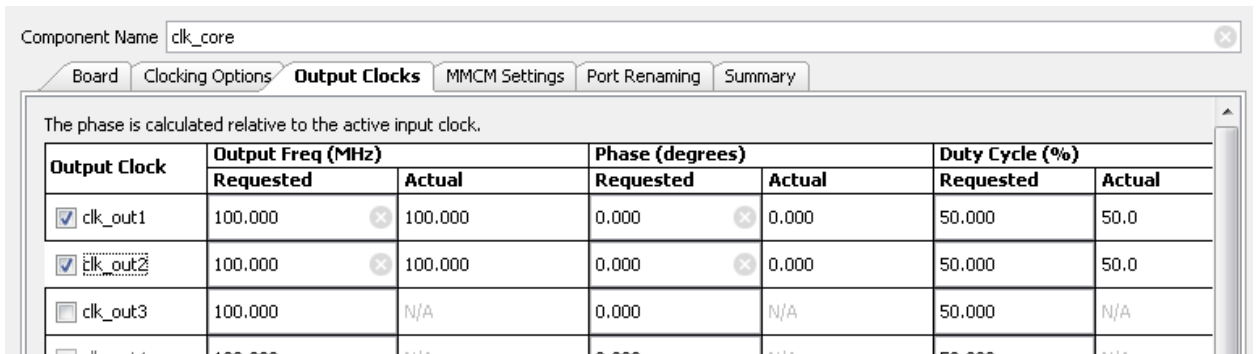


Figure 4. Setting output clocks

2-1-5. Click on the Summary tab and check the information.

Component Name: clk_core

Board | Clocking Options | Output Clocks | MMCM Settings | Port Renaming | **Summary**

Attribute	Value
Input Clock (MHz)	100.000
Phase Shift	None
Divide Counter	1
Mult Counter	10.000
CLKOUT0 Divider	10.000
CLKOUT1 Divider	10
CLKOUT2 Divider	OFF
CLKOUT3 Divider	OFF
CLKOUT4 Divider	OFF
CLKOUT5 Divider	OFF
CLKOUT6 Divider	OFF

Figure 5. Summary page of the clock core being generated for the ZedBoard

2-1-6. Click **OK** to see the *Generate Output Products* form.

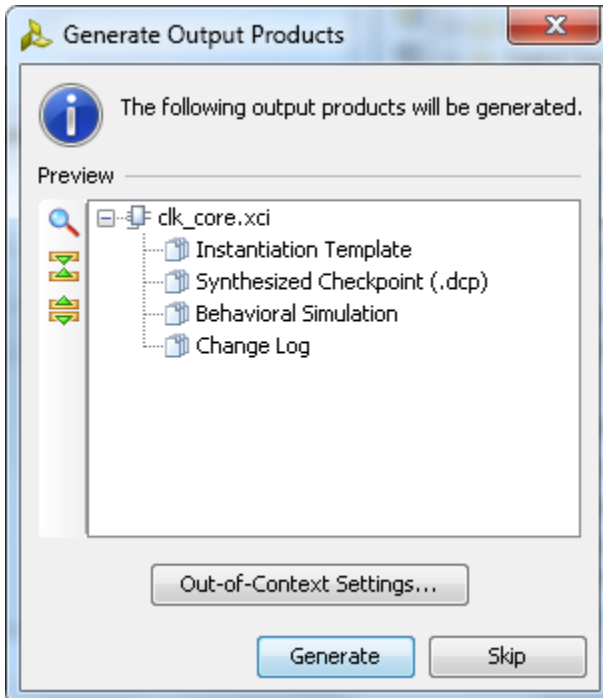


Figure 6. Generate Output Products form

2-1-7. Click on **Generate** to generate the output products including the instantiation template.

2-2. Instantiate the generated clock core.

2-2-1. Select the **IP Sources** tab in the *Sources* pane.

2-2-2. Click the "+" sign next to IP (2). Notice the two IP entries. The char_fifo IP is the core that was included while creating project. The second core clk_core is the one that you have generated.

- 2-2-3.** Expand **clk_core > Instantiation Template** and double-click on **clk_core.veo** to see the instantiation template.
- 2-2-4.** Copy lines 71 through 80 and paste them at or around line 79 of the **clk_gen.v** file.
- 2-2-5.** Change the instance name and net names to as shown in the figure below to match the names of existing signals in the design.

```
// Instantiate clk_core - generated by the Clocking Wizard
clk_core clk_core_i0
  // Clock in ports
  .clk_in1(clk_pin), // input clk_in1
  // Clock out ports
  .clk_out1(clk_rx), // output clk_out1
  .clk_out2(clk_tx), // output clk_out2
  // Status and control signals
  .reset(rst_i), // input reset
  .locked(clock_locked); // output locked
```

Figure 7. Assigning instance name and net connections

- 2-2-6.** Select **File > Save File** to save the Verilog file.
- 2-2-7.** Select the Hierarchy tab and expand the **wave_gen > clk_gen_i0** hierarchy and verify that **clk_core.xci** is in the hierarchy.

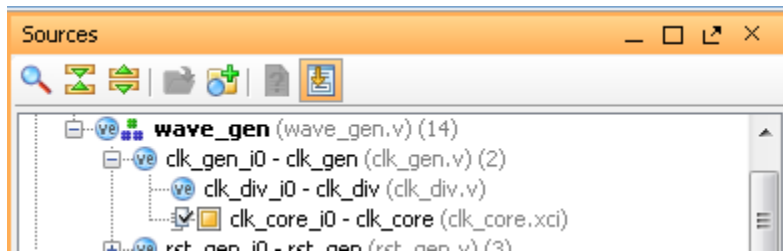


Figure 8. The clk_core instantiated and shown in the hierarchy

Implement the Design

Step 3

3-1. Implement the design.

- 3-1-1.** Click on the **Run Implementation** in the *Flow Navigator* pane.
- 3-1-2.** Click **OK** and run the synthesis first before running the implementation process.
When the implementation is completed, a dialog box will appear with three options.
- 3-1-3.** Select the *Open Implemented Design* option and click **OK**.

3-2. View the amount of FPGA resources consumed by the design using Report Utilization.

- 3-2-1.** In the *Flow Navigator* pane, select **Implemented Design > Report Utilization**.

The Report Utilization dialog box opens.

3-2-2. Click **OK**.

3-2-3. Verify that the design is using the clock resource.

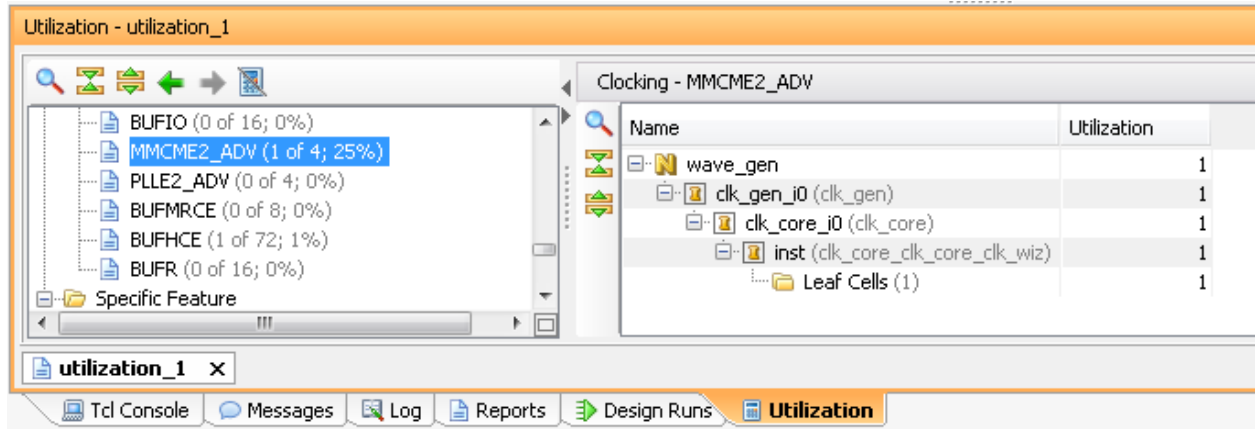


Figure 9. Clock resource utilization for the ZedBoard

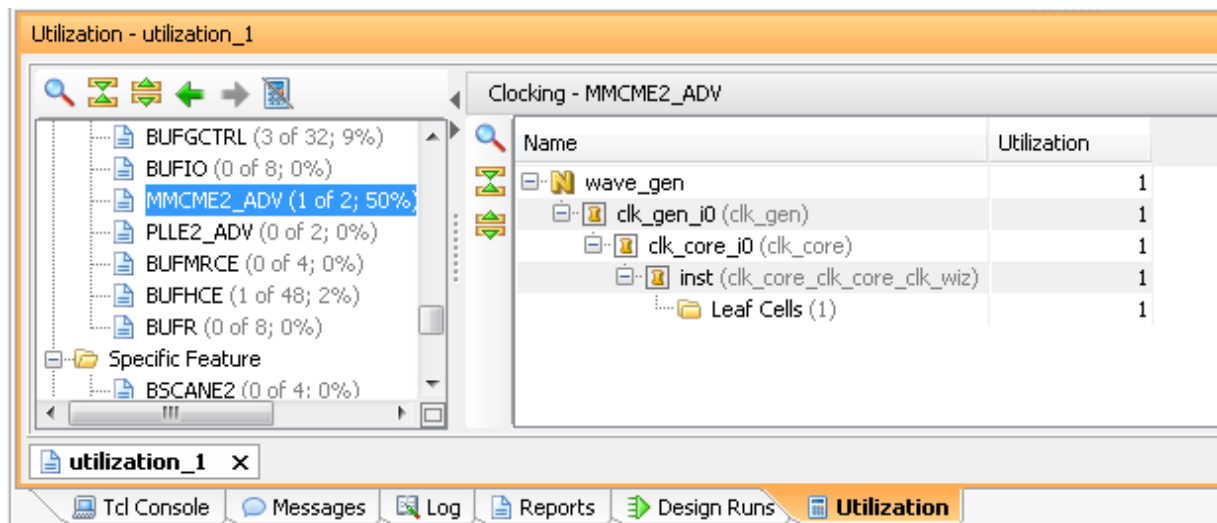


Figure 9. Clock resource utilization for the Zybo

Generate the Bitstream and Verify the Functionality

Step 4

4-1. Generate the bitstream.

4-1-1. In the Flow Navigator, under Program and Debug, click **Generate Bitstream**.

4-1-2. The `write_bitstream` command will be executed (you can verify it by looking in the Tcl console).

4-1-3. Click **Cancel** when the bitstream generation is completed. In some cases, a dialog box may not pop up. Do not worry, just move to the next step.

4-2. Plug-in the PmodUSB UART module into the top-row of the JA PMOD connector for the ZedBoard or the top-row if the JE PMOD connector for the Zybo. Connect the module to the host machine using a micro-USB cable. Connect the board and power it ON. Open a Hardware Manager, and program the FPGA.

4-2-1. For the ZedBoard, plug-in the PmodUSB UART module into the top row of the JA PMOD connector. For the Zybo, plug the PmodUSB UART module into the top row of the JE PMOD connector.

4-2-2. Connect the micro-USB cable between the PmodUSB UART module and the host USB port.

4-2-3. Make sure that the Micro-USB cable is connected to the JTAG PROG connector (next to the power supply connector). Connect the power jack and turn ON the power.

4-2-4. Select the *Open Hardware Manager* option.

The Hardware Manager window will open indicating an “unconnected” status.

4-2-5. Click on the **Open New Hardware Target** link.

You can also click on the *Open Recent Hardware Target* link if the board was already targeted before. In this case skip to step 4-2-10.

4-2-6. Click **Next** to see the Vivado Hardware Server Settings form.

4-2-7. Click **Next** with the Hardware Target selected.

The JTAG cable should be detected and identified as a hardware target. It will also show the hardware devices detected in the chain.

4-2-8. Click **Next** and then click **Finish**.

4-2-9. The Hardware Manager status changes from Unconnected to the server name and the device is highlighted. Also notice that the Status indicates that it is not programmed.

4-2-10. Select the device and verify that the **wave_gen.bit** is selected as the programming file in the General tab.

4-3. Start a terminal emulator program such as TeraTerm or HyperTerminal. Select an appropriate COM port (you can find the correct COM number using the Control Panel). Set the COM port for 115200 baud rate communication. Program the FPGA and verify the functionality.

4-3-1. Start a terminal emulator program such as TeraTerm or HyperTerminal.

4-3-2. Select the appropriate COM port (you can find the correct COM number using the Control Panel).

4-3-3. Set the *COM* port for **115200** baud rate communication.

4-3-4. Right-click on the FPGA entry in the Hardware window and select **Program Device...**

4-3-5. Click on the **Program** button.

The programming bit file be downloaded and the DONE light will be turned ON indicating the FPGA has been programmed.

4-3-6. Slide *Switch 0* to the **ON** position and type in some characters in the terminal window and see the character is echoed back. Setting Switch 0 to the ON position makes the design function as a loopback. After verifying, slide Switch 0 back to the **OFF** position.

4-3-7. Select **File > Send File ...** in the Tera Term window.

4-3-8. Browse to `<2014_2_zynq_sources>\<board>\lab4`, select `testpattern.txt` file, and click **Open**.

The file content will be send to the design. You will see the last line sent to the device. The file content is as follows:

```
*PFFFF          < -- specifies the pre-scaling
*S0fff          < -- specifies the speed value
*N000f          < -- specifies the number of samples to play
*W00000000     < -- write first sample of value 0 at location 0000
*W00011111     < -- write second sample of value 0x1111 at location 0001
*W00022222
*W00033333
*W00044444
*W00055555
*W00066666
*W00077777
*W00088888
*W00099999
*W000AAAAA
*W000BBBBB
*W000CCCCC
*W000DDDDD
*W000EEEEEE
*W000FFFFFF
```

4-3-9. The design understands various commands as listed in figure below. All values are in hexadecimal. All values and addresses are in hexadecimal.

Cmd	Input	Response	Description
*W	aaaavvvv	-OK or -ERR	03ff ≥ aaaa ≥ 0000. Value "vvvv" is written into RAM at location "aaaa" and "-OK" is return.
*R	aaaa	-hhhh dddd or -ERR	03ff ≥ aaaa ≥ 0000. If in range, then the value at "aaaa" is returned in hex and decimal.
*N	vvvv	-OK or -ERR	0400 ≥ vvvv ≥ 0001. Specifies the number of samples before recycling.
*P	vvvv	-OK or -ERR	ffff ≥ vvvv ≥ 0020. Specifies prescaling value to divide <i>clk_tx</i> by to produce <i>clk_samp</i> .
*S	vvvv	-OK or -ERR	ffff ≥ vvvv ≥ 0001. Specifies "speed" value to divide <i>clk_samp</i> by to produce the rate of read from RAM.
*n/*p/*s		-hhhh dddd	Returns current value of nsamp, prescale, and speed.
*G		-OK	Triggers a single pass through nsamp memory locations.
*C		-OK	Starts continuous triggering.
*H		-OK	Halts continuous loop at end of current cycle.

Figure 10. Commands

4-3-10. Next type *G in the terminal window and observe the LED pattern changing slowly as written by the above file.

4-3-11. You can also type *H to halt the play.

```
*PFFFF-OKf
*S0ffff-OK
*N000f-OK
*W00000000-OK
*W00011111-OK
*W00022222-OK
*W00033333-OK
*W00044444-OK
*W00055555-OK
*W00066666-OK
*W00077777-OK
*W00088888-OK
*W00099999-OK
*W000AAAAA-OK
*W000BBBBB-OK
*W000CCCCC-OK
*W000DDDDD-OK
*W000EEEEEE-OK
*W000FFFFFF-OK
*C-OK
*p-fff 65535
*n-000f 00015
*s-0fff 04095
*C-OK
*H-OK
```

Figure 11. Terminal window display

4-3-12. When satisfied, close the terminal emulator program and power OFF the board.

4-3-13. Select **File > Close Hardware Manager**. Click **OK** to close it.

Generate and Instantiate an IPI Block

Step 5

5-1. Save the project as lab4_ipi. Remove the char_fifo IP from the design.

5-1-1. Select **File > Save Project As...** and save it as **lab4_ipi** in the <2014_2_zynq_labs> directory making sure that the *Create Project Subdirectory* option is checked. Click **OK** to save the project.

5-1-2. Select the **IP Sources** tab in the *Sources* pane.

5-1-3. Expand IP(2) and right-click on *char_fifo*, and select **Remove File from Project...**

5-1-4. Click on the check-box of *Also delete the project local file/directory from disk*, and click **OK**.

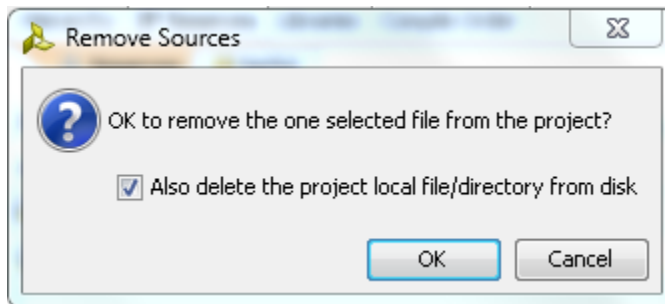


Figure 12. Removing an existing IP from the project

5-1-5. Select **Hierarchy** tab in the *Sources* pane and observe that the *char_fifo* instance has a ? mark indicating that it is missing the source file.

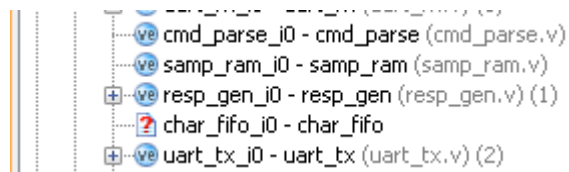


Figure 13. Removed source file

5-1-6. Double-click on the **wave_gen.v** to open it in the editor window.

5-1-7. Remove the instantiation of the *char_fifo* from the file around line 336. You can choose to comment out the lines.

5-1-8. Select **File > Save File**.

5-2. Create a block design naming it as char_fifo and add an instance of an FIFO Generator IP.

5-2-1. Click on **Create Block Design** in the Flow Navigator block.

5-2-2. Enter **char_fifo** as the block design name.

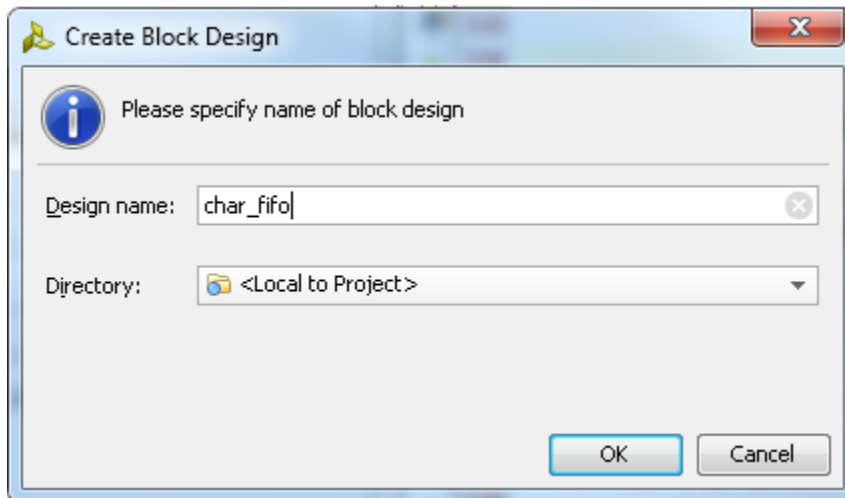


Figure 14. Naming the new block design

5-2-3. Click **OK**.

The IP Integrator workspace opens and, in the information area, invites you to begin adding IP.

5-2-4. In the Tcl Console, type the following command:

```
set_param bd.skipSupportedIPCheck true
```

This will enable the FIFO Generator to appear in the IP Integrator IP Catalog. This is required because by default the IP Catalog in the IP Block design will include just the processor based system IPs.

5-2-5. Right-click in the IP Integrator design canvas and select **Add IP**.

The IP Integrator IP Catalog opens, displaying a list of IP available in the IP Integrator.

5-2-6. Type **FIFO** in the search box at the top of the IP Integrator Catalog to see FIFO related available IPs.

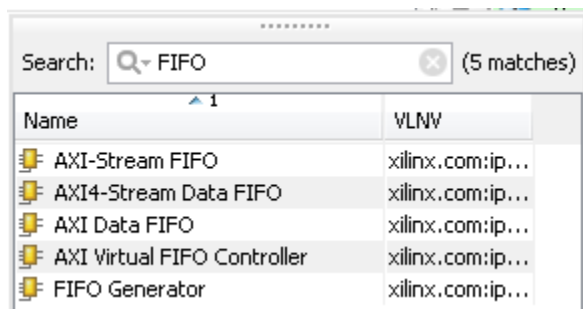


Figure 15. Searching for an IP in the IP Catalog

5-2-7. Double-click **FIFO Generator**.

The FIFO is added to the IP Integrator design canvas.

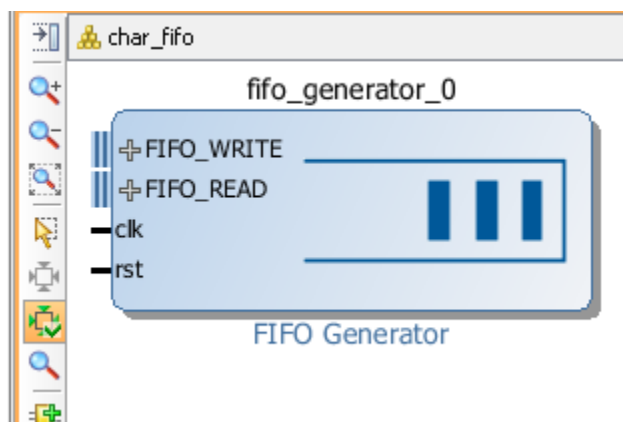


Figure 16. FIFO Generator instantiated

5-3. Customize the FIFO Generator IP instance.

5-3-1. Double-click the FIFO Generator IP.

The FIFO Generator displays in the Re-customize IP dialog box.

5-3-2. Make sure that the default **Native** option is selected for the interface type.

5-3-3. Select **Independent Clocks Block RAM** from the Fifo Implementation drop-down list.

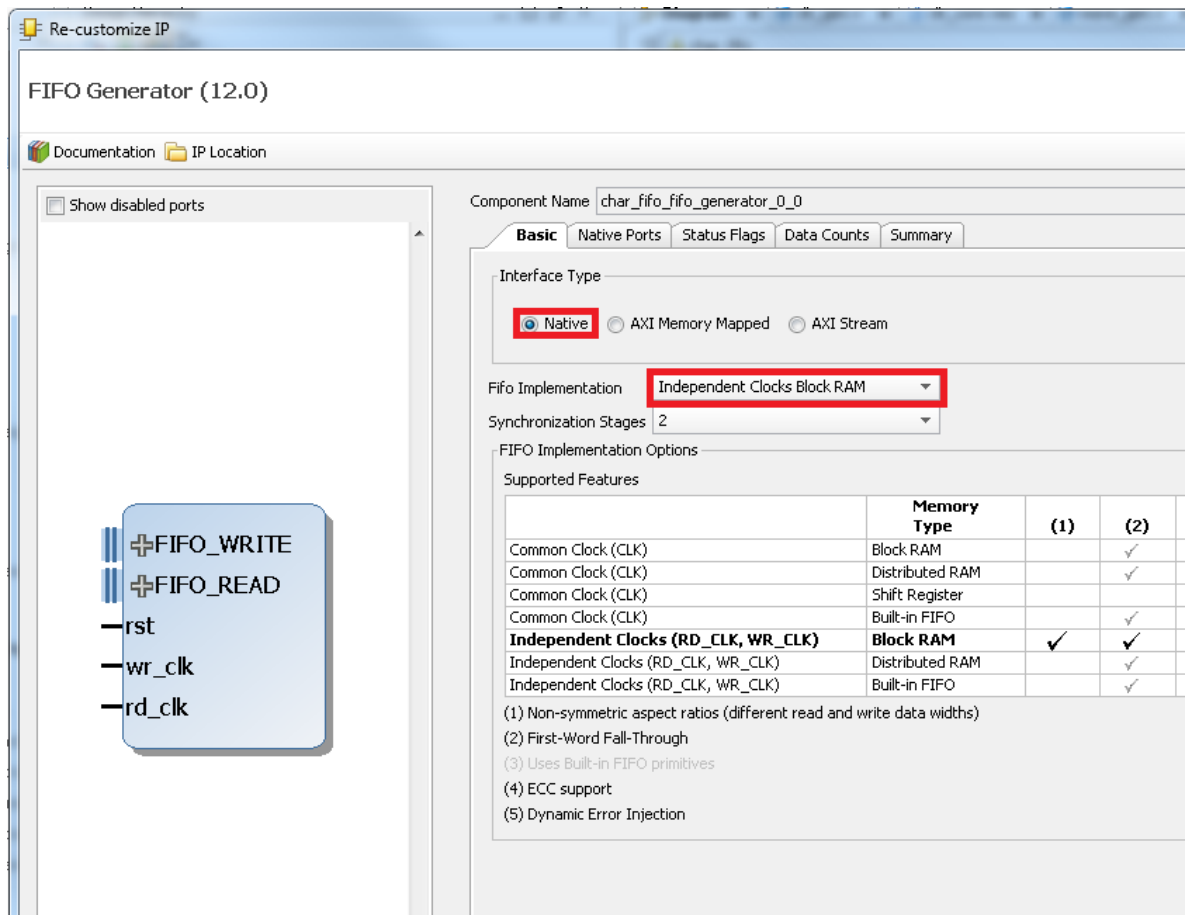


Figure 17. Configuring BRAM for separate read and write clocks

5-3-4. Select the **Native Ports** tab.

From the *Native Ports* tab you can configure the read mode, built-in FIFO options, data port parameters, and implementation options.

5-3-5. Select **First Word Fall Through** as the read mode.

5-3-6. Set the write width to be **8** bits.

5-3-7. Click in the *Read Width* field to change it to match the write width.

5-3-8. Leave everything else at their default settings.

Component Name: char_fifo_fifo_generator_0_0

Basic Native Ports Status Flags Data Counts Summary

Read Mode

Standard FIFO First Word Fall Through

Data Port Parameters

Write Width: 8 (1,2,3,..1024)

Write Depth: 1024 Actual Write Depth: 1025

Read Width: 8

Read Depth: 1024 Actual Read Depth: 1025

ECC And Output Register Options

ECC Single Bit Error Injection Double Bit Error Injection

Embedded Registers in BRAM or FIFO (when possible)

Initialization

Reset Pin Enable Reset Synchronization

Reset Type: Asynchronous Reset

Full Flags Reset Value: 1

Dout Reset Value: 0 (Hex)

Read Latency : 0

Figure 18. Configuring port width and read mode

5-3-9. Browse through the settings of the **Status Flags** and **Data Counts** tabs.

These tabs configure other options for the FIFO Generator. For this design, leave everything at their default settings

5-3-10. Select the **Summary** tab.

This tab displays a summary of all the selected configuration options, as well as listing resources used for this configuration.

Basic Native Ports Status Flags Data Counts Summary	
WARNING : Behavioral models do not model synchronization delays. Use post-par simulation models for accurate behaviour	
Block RAM resource(s) (18K BRAMs): 1	
Block RAM resource(s) (36K BRAMs): 0	
Clocking Scheme	Independent Clocks
Memory Type	Block RAM
Model Generated	Behavioral Model
Write Width	8
Write Depth	1025
Read Width	8
Read Depth	1025
Almost Full/Empty Flags	Not Selected/Not Selected
Programmable Full/Empty Flags	Not Selected/Not Selected
Data Count Outputs	Not Selected
Handshaking	Not Selected
Read Mode / Reset	First-word Fall-through / Asynchronous
Read Latency (From Rising Edge of Read Clock)	

Figure 19. Summary page

5-3-11. Verify that the information is correct. For this configuration you are using one 18K block RAM. Click **OK**.

5-4. Make the ports external and rename them.

5-4-1. Select **wr_clk** and then press and hold the **Ctrl** key and select the **rd_clk** ports of the FIFO.

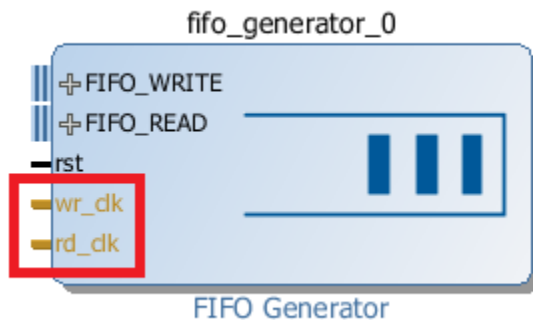


Figure 20. Selecting multiple ports

5-4-2. With the ports highlighted, right-click and select **Make External**.

Two external connections are created for the selected FIFO ports. Notice that the external connections have the same name as the IP module port that they connect to. You can rename these connections by selecting them and changing the name in the External Port Properties window.

5-4-3. Select the external connection port named **wr_clk**.

- 5-4-4.** In the External Port Properties window, in the Name field of the General tab, type the name **clk_rx** and press **Enter**. Similarly, select the external connection port named **rd_clk** and change its name to **clk_tx**.

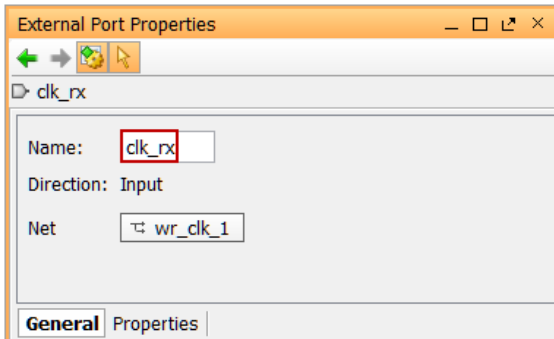


Figure 21. Changing the external port name

- 5-4-5.** You will need to expand FIFO_WRITE and FIFO_READ to see the signal names. While pressing the **Ctrl** key, click all of the remaining FIFO input and output ports and make them external.

- 5-4-6.** Change their names as listed below:

- o din = char_fifo_din
- o dout = char_fifo_dout
- o empty = char_fifo_empty
- o full = char_fifo_full
- o rd_en = char_fifo_rd_en
- o wr_en = char_fifo_wr_en
- o rst = rst_i

When you have finished, your subsystem design should look like the figure below.

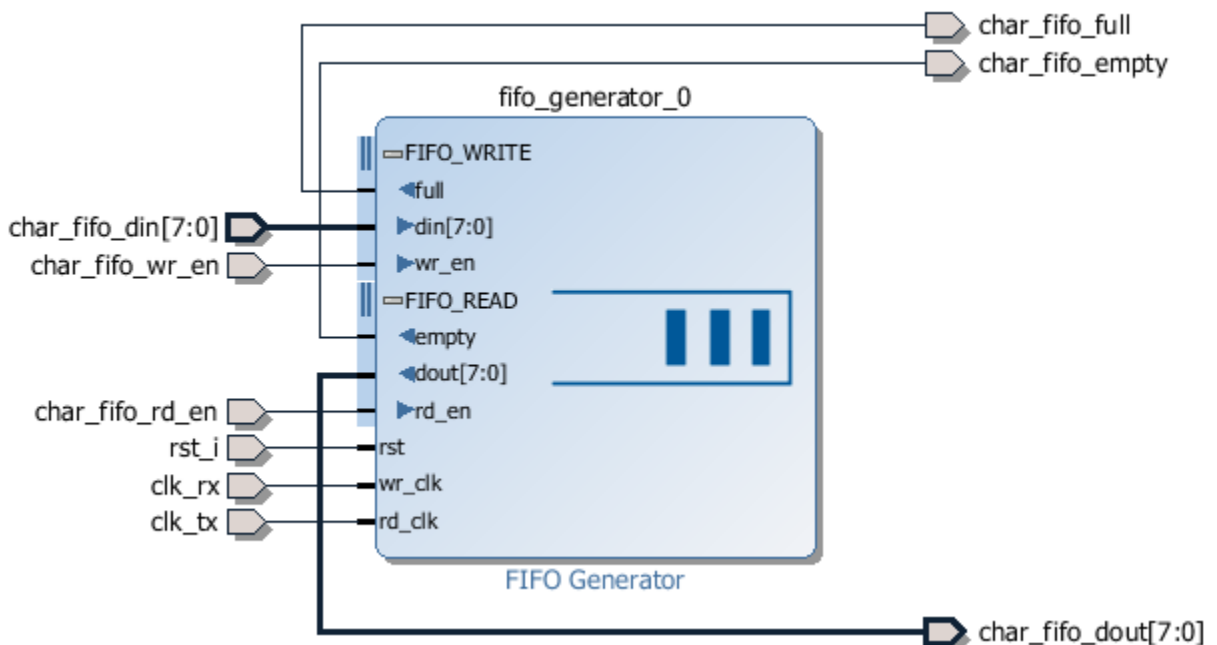



Figure 22. Renamed external ports

5-4-7. Click on Regenerate () icon from the vertical toolbar to see the above diagram.

5-4-8. Select **Tools > Validate Design**.

You should see a message that validation was successful, click **OK**.

5-5. Generate the output product.

5-5-1. In the IP Sources tab of the Sources window, select the **char_fifo** under the Block Designs branch.

5-5-2. Right-click and select **Generate Output Products**.

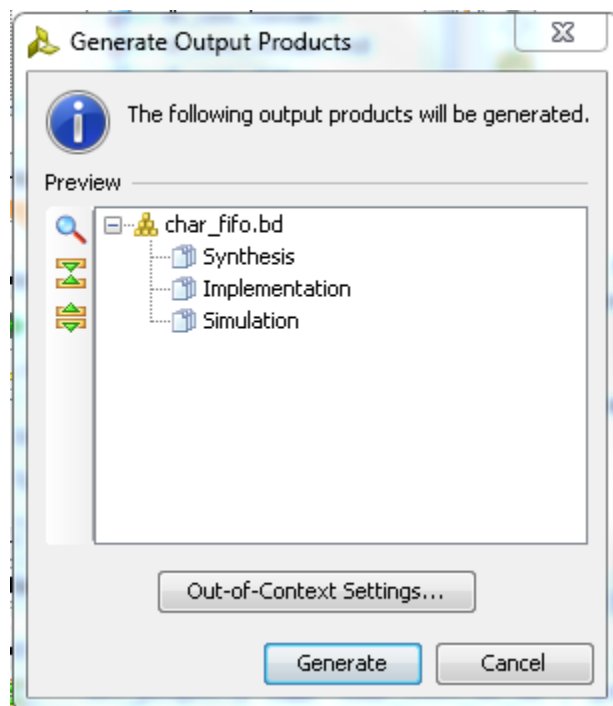


Figure 23. Generating the output products so the IP can be instantiated in the design

5-5-3. Click **Generate**.

You should see the various IP output products displayed in the IP Sources tab of the Sources window.

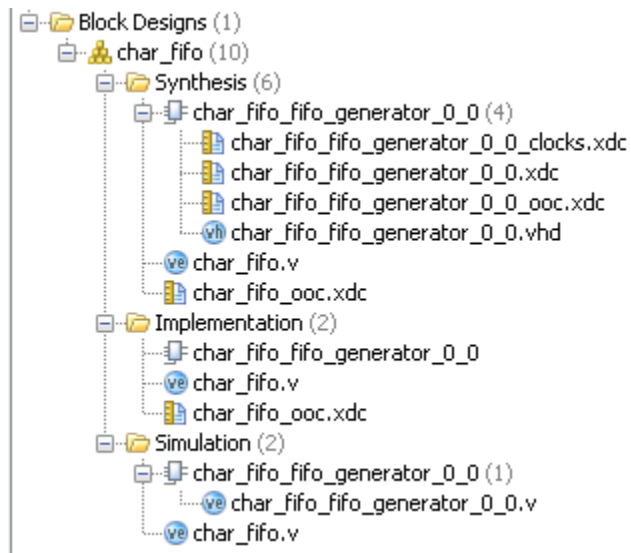


Figure 24. Generated output products

5-6. Instantiate the char_fifo IP in the project.

5-6-1. From the *IP Sources* tab of the *Sources* window, select the **char_fifo** module.

5-6-2. Right-click and select **View Instantiation Template**.

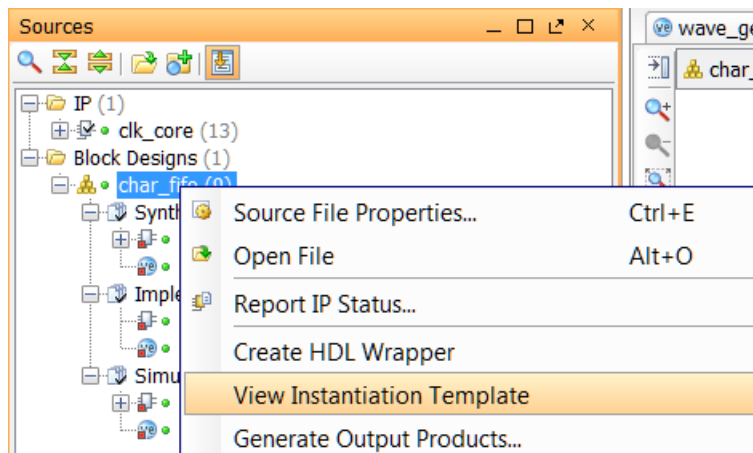


Figure 25. Generating an instantiation template

The char_fifo_wrapper.v instantiation template is opened in the text editor in the Vivado IDE.

```
42 char_fifo char_fifo_i
43     (.char_fifo_din(char_fifo_din),
44     .char_fifo_dout(char_fifo_dout),
45     .char_fifo_empty(char_fifo_empty),
46     .char_fifo_full(char_fifo_full),
47     .char_fifo_rd_en(char_fifo_rd_en),
48     .char_fifo_wr_en(char_fifo_wr_en),
49     .clk_rx(clk_rx),
50     .clk_tx(clk_tx),
51     .rst_i(rst_i));
52 endmodule
```

Figure 26. Part of the instantiation template

5-6-3. Copy lines 42 through line 51, and paste them at or around line 334 in the **wave_gen.v** file.

5-6-4. Save the Verilog file.

5-7. Generate the bitstream and verify the functionality in hardware.

5-7-1. Click on the **Run Implementation** in the *Flow Navigator* pane.

Click **Yes** when prompted to relaunch synthesis first. If prompted, click **Yes** to Save the project before proceeding.

5-7-2. When completed, generate the resource utilization report and verify that one Block RAM Tile is being used.

5-7-3. In the Flow Navigator, under Program and Debug, click **Generate Bitstream**.

5-7-4. Open the Hardware Manager and verify the functionality of the design in the hardware.

5-7-5. When done, close the **Vivado** program by selecting **File > Exit** and click **OK**.

Conclusion

In this lab, you learned how to add an existing IP during the project creation. You also learned how to use IP Catalog and generate a core. You then instantiated the core in the design, implemented the design, and verified the design in hardware. You also used the IP Integrator capability of the tool to generate a FIFO and then use it in the HDL design.