Lab 7: Integrating Custom Cores – AXI FIR and CODEC

Introduction

Xilinx provides commonly used bus interfaces for customized IP cores to talk to the system. Xilinx also provides a customized IP Packager tool to automatically interface to your IP core when you use it to create customized hardware.

In the previous lab, you learned how to create a hardware platform using the Vivado® IP integrator. The purpose of this lab is to show you how to add a custom IP core through the IP integrator after the project settings are made to point to the provided custom core

In this lab you are provided two AXI-Lite custom IPs – audio controller (zed_audio_ctrl) and FIR filter. The audio controller will communicate with the on-board CODEC chip whereas the FIR filter will process the input sample through a 58-tap FIR filter and output the sample. Since we are inputting 0x8000 (most negative) to the provided filter will show the first few coefficients as:
`0x17A,0x49, 0xFFE5, 0xFF56, 0xFED6, 0xFEA0, 0xFED2, 0XFF58 …`

So when an impulse input is given to the filter, you expect the output samples to produce impulse response corresponding to the coefficients.

Objectives

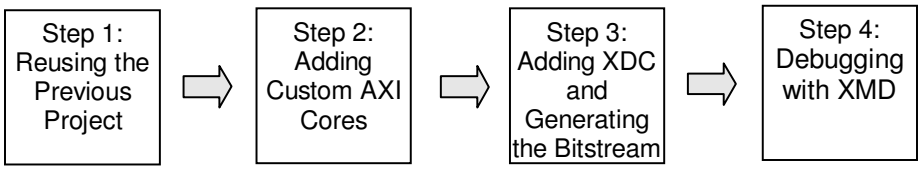
After completing this lab, you will be able to:
- Add custom IP to an existing project by using Vivado IP Integrator
- Use XMD to debug the hardware

Preparation

If this is the first lab that you are performing, then refer to the "Before You Start" section of Lab 1 for necessary preparatory information on how to set up the environment.

Please refer to the "Initializing the Workshop Environment" section of Lab 1 for detailed information.

General Flow for this Lab



Reusing the Previous Vivado Design Suite Project Step 1

You will use the design you created in Lab 6. You will save this previous project so that it can be used as the current lab project.

1-1. Create a `lab7` directory under the `labs` directory.

1-1-1. Create a `lab7` project directory in `~/emblnx/labs` by executing the following commands on the host machine:

```
[host] $ mkdir ~/emblnx/labs/lab7
[host] $ mkdir ~/emblnx/labs/lab7/hardware
[host] $ cd ~/emblnx/labs/lab7/hardware
```

1-2. Launch the Vivado Design Suite and open the Vivado Design Suite project of the previous lab (lab6) and save the project as lab7.

1-2-1. Launch the Vivado Design Suite, if it is not already open, by executing the following command:

```
[host] $ vivado
```

Note: If the command fails to start the Vivado Design Suite, then execute the following command first and then try the above command again:

```
host] $ source /opt/pkg/Xilinx/Vivado/2014.2/settings32.sh
```

1-2-2. Click the **Open Project** link in the Getting Started page.

1-2-3. Click **Browse Projects**, browse to the *lab6* directory in the *~/emblnx/labs/lab6/hardware* directory, and select the **lab6.xpr** entry.

1-2-4. Select **File** > **Save Project As**.

1-2-5. Click the **Browse** button next to the Project location field and browse to the */home/petalinux/emblnx/labs/lab7/hardware* directory.

1-2-6. Enter **lab7** in the Project name field.

1-2-7. Verify that the **Create project subdirectory** option is not selected.

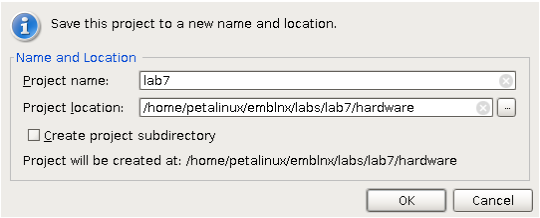


Figure 1. Saving the previously created project as a new project

1-2-8. Click **OK**.

Adding the Custom AXI Cores and Completing the Design Step 2

Before custom cores can be added, the project settings need to be updated with the IP repositories.

2-1. Set the project settings to add the provided AXI cores repository.

2-1-1. Click **Project Settings** under Project Manager in the Flow Navigator pane.

2-1-2. Select **IP** in the left pane of the Project Settings window.

2-1-3. Click **Add Repository** and browse to the */home/petalinux/emblnx/sources/lab7* directory.

2-1-4. Click **Select**.

Notice that the folder will be scanned and the **Fir** and **zed_audio_ctrl** IP entries will be displayed in the IP in Selected Repository section.

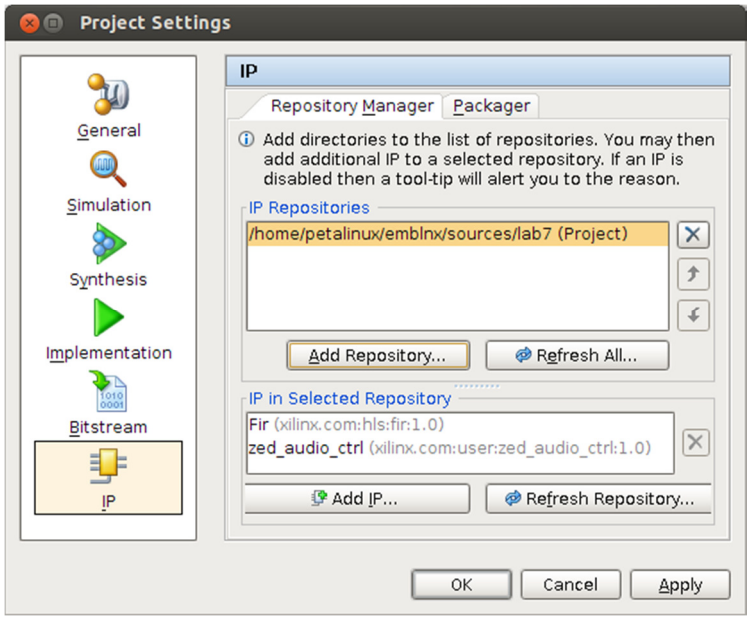


Figure 2. Adding IP Repository to the project

2-1-5. Click **OK** to close the Project Settings window.

2-2. Open the block design and delete all three GPIO IP instances (for switches, buttons, and LEDs) and their associated external ports.

2-2-1. Click **Open Block Design** under the IP Integrator sub-group in the Flow Navigator pane to open the block design.

2-2-2. Select each of the GPIO instances (switches, buttons, and LEDs) one at a time and press the <**Delete**> key to remove them from the design.

2-2-3. Right-click each of the external ports (of the deleted switches, buttons, and LEDs) and select **Delete** to delete them.

2-3. Add one instance of the zed_audio_ctrl and two instances of the FIR filter.

2-3-1. Add an instance of the *zed_audio_ctrl* IP.

2-3-2. Click on **Run Connection Automation**, select **/zed_audio_ctrl_0/S_AXI** and click **OK.**

Notice that the **S_AXI** interface of the added IP is connected to the **M00_AXI** interface of the *processing_system7_0_axi_periph* instance. Also, the S_AXI_ACLK and S_AXI_ARESETN ports are connected appropriately.

2-3-3. Add an instance of the **FIR** IP to the design.

2-3-4. Select the added instance in the diagram, and change its instance name to **fir_left** by typing it in the *Name* field of the *Block Properties* form in the left.

2-3-5. Similarly, add another instance of the FIR IP, and name it **fir_right**.

2-3-6. Click on **Run Connection Automation**, and select /**fir_left**/**S_AXI_FIR_IO** and click **OK.**

Notice that the **S_AXI** interface of the **fir_left** instance is connected to the **M01_AXI** interface of the *processing_system7_0_axi_periph* instance.

2-3-7. Similarly, click on **Run Connection Automation** again, and select /**fir_right**/**S_AXI_FIR_IO** and click **OK.**

2-4. Add one instance of AXI GPIO with 2-bits output only on Channel 1 and 1-bit input only on Channel 2.

2-4-1. Add an instance of the *AXI GPIO* IP.

2-4-2. Click on **Run Connection Automation**, select the /**axi_gpio_0**/**S_AXI** and click **OK.**

2-4-3. Double-click on the *axi_gpio_0* instance and configure it to have **two** bits **output** only on Channel 1 and **one** bit **input** only on Channel 2.

2-5. Configure the processing system block to enable I2C 1. Enable FCLK_CLK1, the PL fabric clock and set its frequency to 8.000 MHz. Enable the PS-PL Interrupt ports > IRQ_F2P ports.

2-5-1. Double-click on the *processing_system7_0* instance to open the re-customization form.

2-5-2. Select the *MIO Configuration* tab on the left to open the configuration form and expand *I/O Peripheral* in the right pane.

2-5-3. Click on the check box of the *I2C 1*.

2-5-4. Select the *Clock Configuration* in the left pane, expand *the PL Fabric Clocks* entry in the right, and click the check-box of *FCLK_CLK1*.

2-5-5. Change the *Requested Frequency* value of *FCLK_CLK1* to **8.000** MHz.

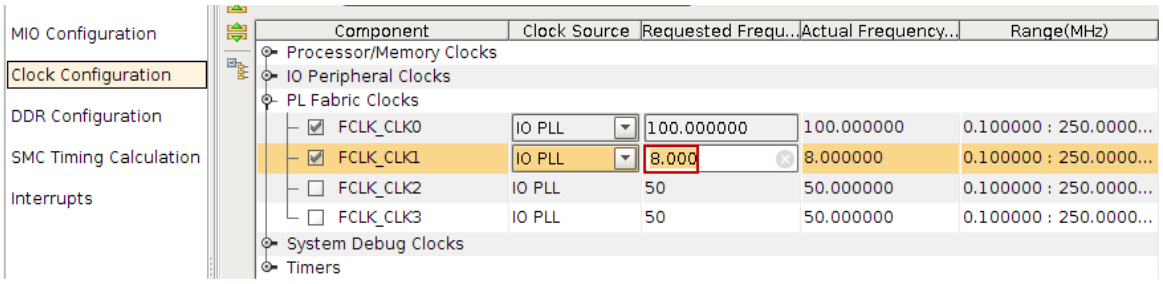


Figure 3. Enabling and setting the frequency of FCLK_CLK1

2-5-6. Select the *Interrupt* in the left pane, click on the *Fabric Interrupts* check box in the right.

2-5-7. Expand *the Fabric Interrupts > PL-PS Interrupt Ports > IRQ_F2P* entry in the right, and click the check-box of *IRQ_F2P[15:0]*.

2-5-8. Click **OK**.

2-6. Make IIC_1, GPIO, FCLK_CLK1, and zed_audio_ctrl ports external.

2-6-1. Right-click on the **GPIO** interface of the *axi_gpio_0* instance and select **Make External** to create an external port. This will create the external port named *GPIO*.

2-6-2. Similarly, make the **GPIO2** interface of the *axi_gpio_0* instance **External**.

2-6-3. Similarly, make the **IIC_1** interface and **FCLK_CLK1** port of the *processing_system7_0* instance external.

2-6-4. Similarly, selecting one port (input and output) at a time of the *zed_audio_ctrl_0* instance and make them external.

2-7. Add an instance of concat IP with two single-bit input ports. Connect input ports to the interrupt ports of the two FIR instances and the output port to the IRQ_F2P port of the processing_system7_0 instance.

2-7-1. Add an instance of the *concat* IP.

2-7-2. Connect the *interrupt* port of each of the FIR instances to the two input ports of the *xlconcat_0* instance.

2-7-3. Connect the output port of the *xlconcat_0* instance to the IRQ_F2P port of the *processing_system7_0* instance.

At this stage the design should look like shown below (you may have to click the regenerate [ ] button).At this stage the design should look like shown below (you may have to click the regenerate [ ] button).

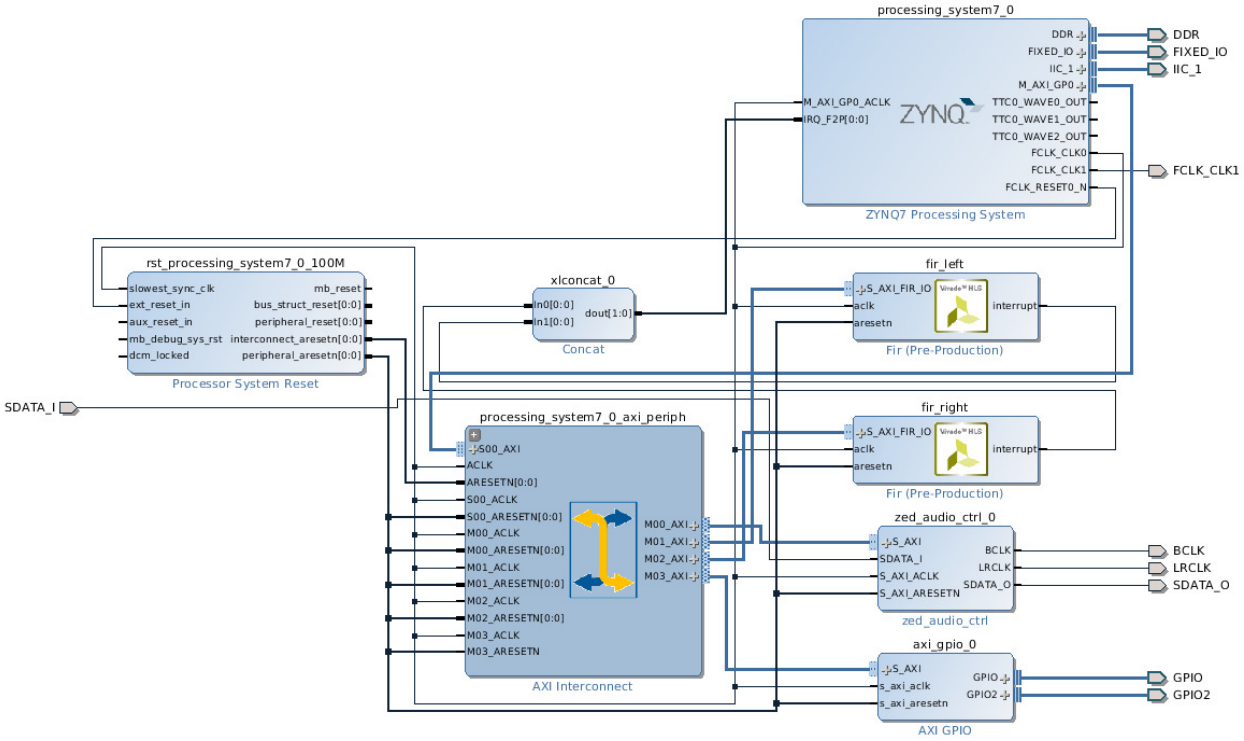
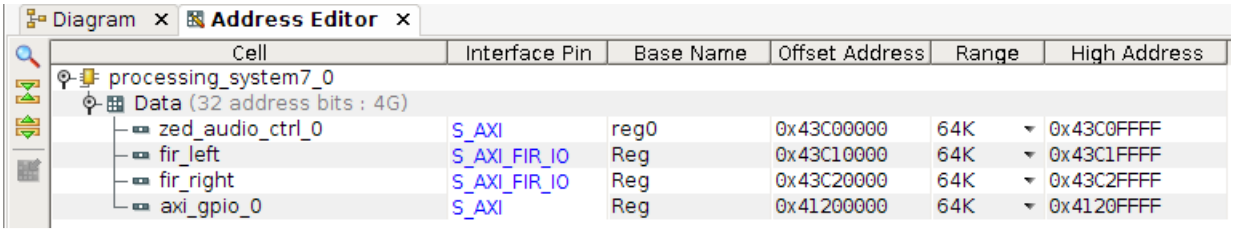


Figure 4. The block design

2-8. Verify the addresses and validate the design.

2-8-1. Select the **Address Editor** tab, and expand **processing_system7_0** > **Data** if necessary.



Cell	Interface Pin	Base Name	Offset Address	Range	High Address
processing_system7_0					
Data (32 address bits : 4G)					
zed_audio_ctrl_0	S_AXI	reg0	0x43C00000	64K	0x43C0FFFF
fir_left	S_AXI_FIR_IO	Reg	0x43C10000	64K	0x43C1FFFF
fir_right	S_AXI_FIR_IO	Reg	0x43C20000	64K	0x43C2FFFF
axi_gpio_0	S_AXI	Reg	0x41200000	64K	0x4120FFFF

Figure 5. Generated address map

2-8-2. Select **File** > **Save Block Design**.

2-8-3. Select the **Diagram** tab.

2-8-4. Select **Tools** > **Validate Design** to run design validation and verify that there are no errors.

2-9. Generate the output products which will also update the system_wrapper file.

2-9-1. In the **Sources > Hierarchy** window, expand the *system_wrapper* hierarchy, right-click on *system_i* and select **Reset Output Products…**

2-9-2. Click **Reset**.

2-9-3. In the Flow Navigator, click on the **Generate Block Design** to generate the output products.

2-9-4. Click **Generate**.

This will also update the system_wrapper file so the added ports will be brought out to the top level.

Adding the XDC File and Generating the Bitstream Step 3

Once the system wrapper file is generated, the user ports are exposed at the top level. The user ports must be constrained with the constraints file before the bitstream is generated. Here you will add the provided XDC (Xilinx Design Constraints) file with the I/O ports assignments and then generate the bitstream.

3-1. Add the provided XDC file and generate the bitstream.

3-1-1. Click **Add Sources** under Project Manager in the Flow Navigator pane and select **Add or Create Constraints**.

3-1-2. Click **Next**.

3-1-3. Click **Add Files** button.

3-1-4. Browse to the */home/petalinux/emblnx/sources/lab7* directory and select **lab7.xdc**.

3-1-5. Click **OK**.

3-1-6. Click **Finish** to add the file.

3-1-7. Click **Generate Bitstream** in the Flow Navigator to run the synthesis, implementation, and bitstream generation processes.

3-1-8. Click **Yes** to re-run the synthesis process.

3-1-9. When the bitstream generation process completes, click **OK** to open the implemented design.

3-2. Export the hardware, launch SDK, and exit SDK.

Launching SDK will create the system_wrapper_hw_platform_0 directory under which ps7_init.tcl file will be generated. This file will be used in the next step to debug using xmd.

3-2-1. Select **File > Export > Export Hardware**.

3-2-2. Make sure that Include Bitstream check box is checked. Click **OK**.

3-2-3. Select **File > Launch SDK**.

This will invoke SDK and create the *system_wrapper_hw_platform_0* directory under which **ps7_init.tcl** file will be generated. This file will be used in the next step.

3-2-4. Exit SDK by selecting **File > Exit**.

Debugging the AXI FIR IP Core with XMD — Step 4

Low-level hardware debugging is useful when testing a new hardware IP core; otherwise, if you combine the hardware test, driver test, and application test and there is a problem, determining whether it is hardware or software problem could be difficult.

The Xilinx Microprocessor Debugger (XMD) allows you to directly write to AXI registers. In this step, you will export the design, invoke SDK, program the FPGA from SDK, and use XMD to debug the FIR AXI IP core.

4-1. Make sure that the board is configured in the JTAG boot mode. Connect the board and power it ON.

4-1-1. Connect the board's JTAG port to the host machine using the provided micro-USB cable.

4-1-2. Set the board jumpers to JTAG boot.

4-1-3. Power ON the board.

4-2. Program the FPGA using Open Hardware Manager.

4-2-1. Select the *Open Hardware Manager* in the Flow Navigator under the *Program and Debug* group and click **OK**.

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

4-2-2. Click on the **Open a new hardware target** link.

You can also click on the **Open recent target** link if the board was already targeted before.

Figure 7. Opening new hardware target

4-2-3. Click **Next** to see the Vivado CSE Server Name form.

4-2-4. Click **Next** with the localhost port selected.

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

4-2-5. Click **Next** and **Finish**.

4-2-6. The Hardware Session status changes from Unconnected to the server name and the device is highlighted. Also notice that the Status indicates that it is not programmed.

4-2-7. Select *xc7z020_1* and verify that the **system_wrapper.bit** is selected as the programming file in the *General* tab of the *Hardware Device Properties*.

4-2-8. Click **Program device > xc7z020_1** and then click **Program** to download the bitstream (system_wrapper.bit) and program the PL section.

The file will be downloaded and the FPGA will be programmed. The DONE LED will be lit indicating that the PL section is programmed.

4-3. Launch XMD from the terminal window and connect XMD to the target. Stop the running application on the ARM® Cortex™-A9 processor.

4-3-1. Open a new Terminal window and change to the `~/emblnx/labs/lab7/hardware/lab7.sdk/system_wrapper_hw_platform_0` directory.

4-3-2. Start the XMD session by typing the following command.

```
[host] xmd
```

The `XMD%` prompt will be displayed.

4-3-3. In the XMD connect to the hardware target by typing the following command:

```
XMD% connect arm hw
```

4-3-4. Stop the processor by issuing the following command:

```
XMD% stop
```

4-4. Test your AXI FIR core by writing the following commands into and reading from the core.

You will write to 0x43c1001c, and 0x43c10000, and read from 0x43c10014 repeatedly to get the impulse response from the left channel FIR filter. You could do the same for the right channel by using 0x43c200xx.

4-4-1. Execute the following set of commands to source the ps7_init.tcl file, and execute the ps7_init, and ps7_post_config commands to enable level shifters so you can talk to the AXI peripherals.

```
XMD% source ps7_init.tcl
XMD% ps7_init
XMD% ps7_post_config
```

4-4-2. Execute the following set of commands to issue 0x8000 as the first input of an impulse input to the filter, send the convert pulse (toggling to 1 and then back to 0) and read the output sample.

```
XMD% mwr 0x43c1001c 0x8000
XMD% mwr 0x43c10000 0x1
XMD% mwr 0x43c10000 0x0
XMD% mrd 0x43c10014
```

```
It should display 0x17A as the first coefficient output
```

4-4-3. Execute the following set of commands to issue 0x00 as the next input of an impulse input to the filter, send the convert pulse (toggling to 1 and then back to 0) and read the output sample.

```
XMD% mwr 0x43c1001c 0x0
XMD% mwr 0x43c10000 0x1
XMD% mwr 0x43c10000 0x0
XMD% mrd 0x43c10014

It should display 0x49 as the next coefficient output
```

You can execute these set of commands (of step 4-4-3) as many times you want to read the next set of coefficients.

4-4-4. When satisfied, disconnect the xmd connection and exit out of the program by executing the following commands:

```
XMD% disconnect 64
XMD% exit
```

4-4-5. Select **File** > **Close Hardware Manager** in Vivado to close the hardware manager.

4-4-6. Click **OK**.

4-4-7. Power OFF the board.

4-4-8. Select **File** > **Exit** in the Vivado Design Suite and click **OK** to close the program.

Conclusion

In this lab, you have learned how to:

- Set the project settings so that a custom IP repository can be accessed
- Add the custom IP to the design
- Utilize the common AXI bus interface
- Use XMD to access software-accessible AXI registers and debug the design

Completed Solution

Set the board to boot using JTAG. Power ON the board. Open `Vivado > Open Hardware Manager`, establish the connection with the board, assign the **system_wrapper.bit** file located under the `labsolution\lab7\lab7.sdk` directory, and program the device. Follow 4-3 and 4-4 steps listed above to test the hardware.