

Laboratory Assignment 4 – Designing with IPs

Introduction

This Lab will guide you through the process of IP core creation, customization and integration into your design [1]. Vivado Design Suite offers IP Packager and IP Integrator tool to help you with the process of designing with IP. The Vivado Design Suite provides multiple ways to use IP in a design. The Vivado IDE provides an IP-Centric design flow that enables you to add IP modules to your project from various design sources. IP-Centric design flow helps you quickly turn design and algorithms into reusable IP. Figure 1 illustrates the IP-Centric design flow.

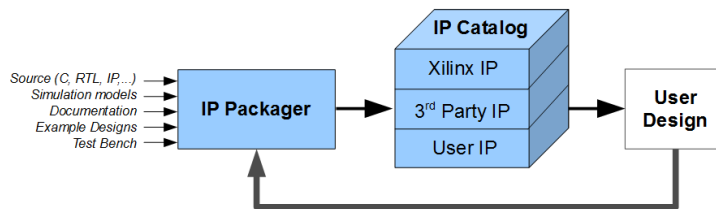


Figure 1 - Vivado IP-Centric Design Flow

As you can see from the illustration above, the IP developer uses the IP Packager to package HDL and other IP source files and create an archive (zip file). The packaged IP is then given to the user and added to the IP Catalog. When the IP is in the IP Catalog, a user can select the IP and create a customization for their design.

The Vivado IDE contains a Create and Package IP wizard that helps and guides you step-by-step through the IP creation and packaging steps. The Create and Package IP wizard offers the following functions:

- Create IP using source files and information from a project
- Create IP from a specified directory
- Create a template AXI4 peripheral that includes the HDL, drivers, a test application, a Bus Functional Model (BFM), and an example template

You can customize and add an IP into the project using the IP Catalog from the Vivado IDE. In the IP Catalog you can add the following:

- Modules from System Generator for DSP designs (MATLAB/Simulink algorithms) and Vivado High-Level Synthesis designs (C/C++ algorithms)
- Third party IP
- User designs packaged using IP Packager

The available methods to work with IP in a design are:

- Use the Managed IP Flow to customize IP and generate output products, including a Synthesized Design Checkpoint (DCP)
- Create and add IP within a Vivado Project. Access the IP Catalog in a project to create and add IP to design. Store the IP either inside the project or save it externally to the project, which is the recommended method for projects with small team sizes

[1] https://www.so-logic.net/documents/knowledge/tutorial/Basic_FPGA_Tutorial_VHDL/index.html

Objectives

After completing this lab, you will be able to:

- create and add user designs in the IP Catalog
- package designs using the IP Packager
- Instantiate your IP into the project using IP Catalog or IP Integrator tools.

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 block diagram shown in Figure 2 shows the structure of one possible system that can be used to generate a PWM signal, modulated using a sine wave with two different frequencies. The frequency chosen depends on the position of the two-state on-board switch.

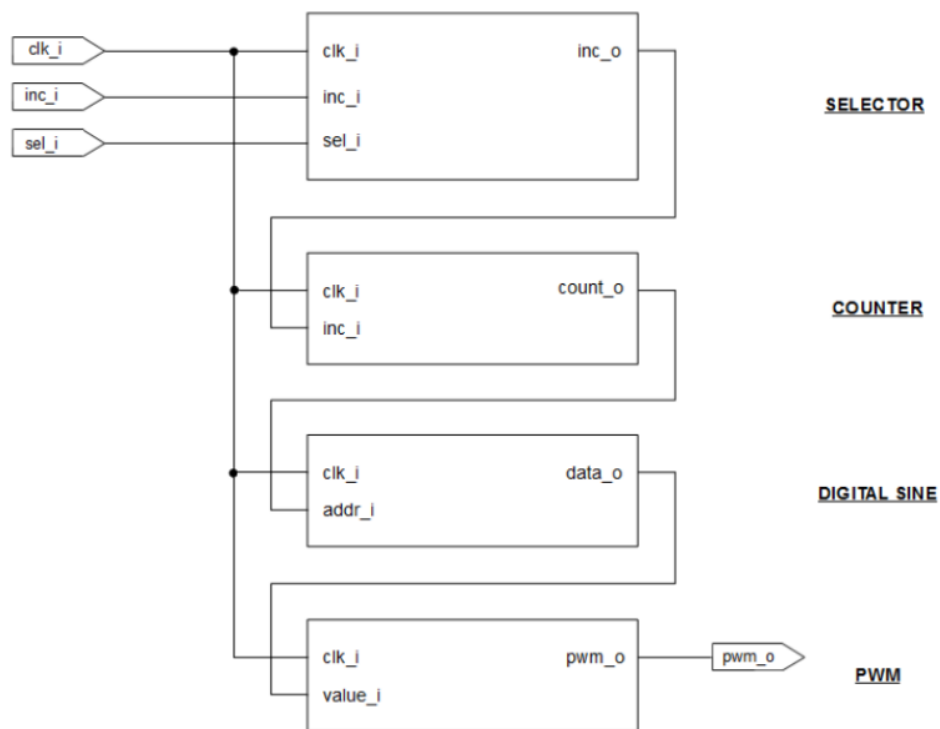
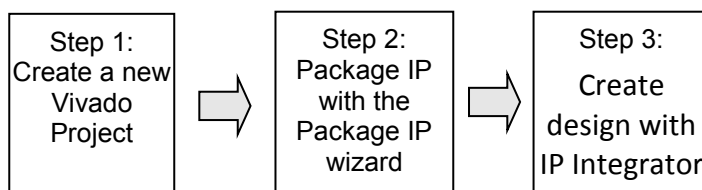


Figure 2. PWM generator Design

General Flow



In the instructions below;

{**sources**} refers to: C:\Users\MEE_CE\sources

{**labs**} refers to : C:\ Users\MEE_CE\labs

Create a new Vivado Project

Step 1

1-1. Launch Vivado and create a project targeting XC7A35TCPG236-1 (Basys3) and using the Verilog HDL.

- 1-1-1. Start downloading the source files provided (Moodle) and save them in {sources}/lab4
- 1-1-2. Open Vivado and 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 {labs}, and click **Select**.
- 1-1-4. Enter **frequency_trigger** 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. Verify that the **RTL Project** is selected and the **Do not specify sources at this time** option is **unchecked** and click **Next**.
- 1-1-6. In the **Add Sources** dialog box, click **Add Files** and select **frequency_trigger_rtl.vhd** source file and click **OK**.
- 1-1-7. Set the Target Language to **VHDL** to indicate the netlist language for synthesis and ensure that you select **Copy sources into project** option, because Xilinx strongly recommends the source files are present within the project. Click **Next**.
- 1-1-8. Click **Next** twice to bypass the *Add Existing IP* and *Add Constraints* dialog boxes.
- 1-1-9. In the *Default Part* form, using the **Parts** option and various drop-down fields of the **Filter** section, select **XC7A35TCPG236-1** (for the Basys3) part. Using the **Boards** option, you may select the **Basys3**.
- 1-1-10. Click **Next**.
- 1-1-11. Click **Finish** to create the Vivado project. After we finished with the new project creation, in a few seconds Vivado IDE will appear with the created **frequency_trigger** project (Figure 3).

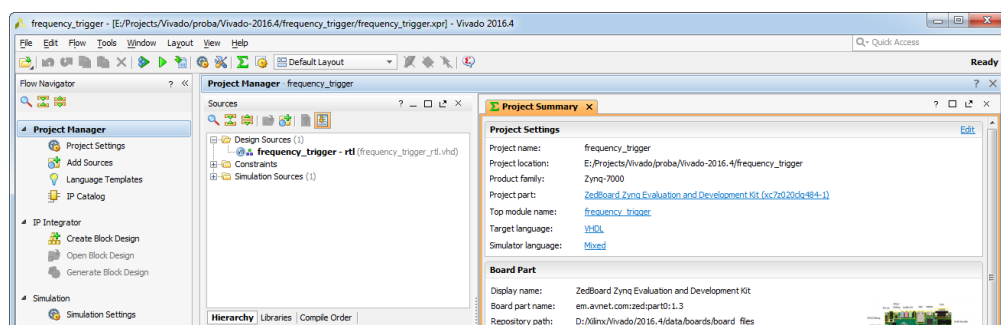


Figure 3. Created new frequency_trigger project

Use the Package IP wizard to package IP

Step 2

2-1. Set Packager Settings.

- 2-1-1. In the Vivado **Flow Navigator**, under the **Project Manager**, click **Project Settings** command and choose **IP** from the left pane, see Figure 4. Global IP project settings are available to help you be more productive when customizing IP.

2-1-2. In the **IP** window, select **Packager** tab and fill the fields as it is shown on Figure 4 and click **OK**.

Packager sets default values for packaging new IP, including vendor, library and taxonomy. This category also allows you to set the default behavior when opening the IP Packager and allows you to specify file extension to be filtered automatically. If necessary, you can change the default values for packaging IP during the IP packaging process.

Note: Ensure that the **Create archive of IP** option is enabled in the **After Packaging** section to deliver an archive file.

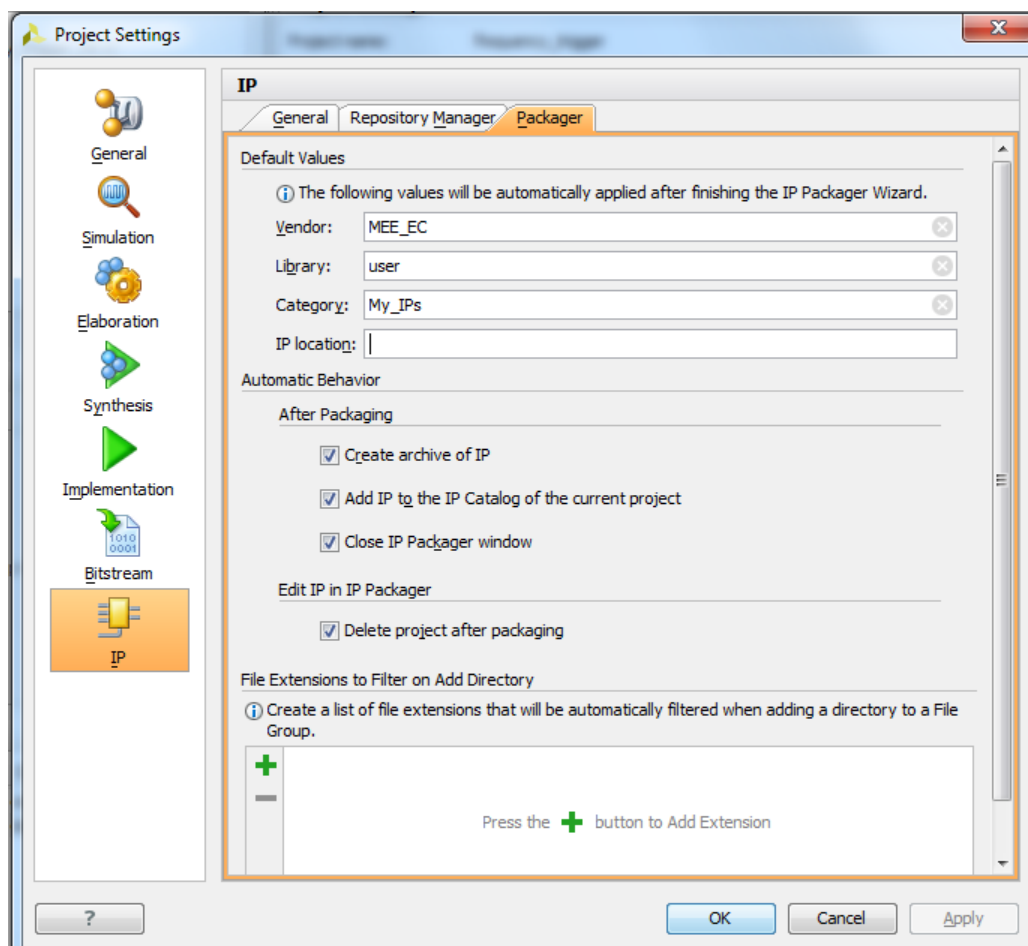


Figure 4. Packager window with configured settings that will be applied after packaging process

2-2. Package a Vivado project as IP.

2-2-1. In the main Vivado IDE menu, select **Tools -> Create and Package New IP...** option (Figure 5).

2-2-2. In the **Create and Package IP** dialog box, click **Next**, see Figure 6.

2-2-3. In the **Create Peripheral or Package IP** dialog box, choose **Package your current project** option and click **Next**, see Figure 7.

2-2-4. In the **Package Your Current Project** dialog box, choose **IP Location** and type of the **Packaging IP in the project** (Figure 8) and click **Next**.

- **IP Location:** The directory in which the tool creates the IP Definition. The default is the project sources directory.
- **Packaging IP in the project:**

- **Include .xci files:** If the project you are packaging includes subcores, package only the IP customization XCI file. By deciding to include the XCI files, the IP Packager packages only the XCI file of the IP customization. This creates a subcore reference to the parent IP and allows the packaged XCI file to be managed by the Vivado tool. The advantage is that the IP can easily be upgraded to the latest release by using the Vivado IP Upgrade methodology.
- **Include IP generated files:** Packages the generated RTL and XDC sources of the IP customization.

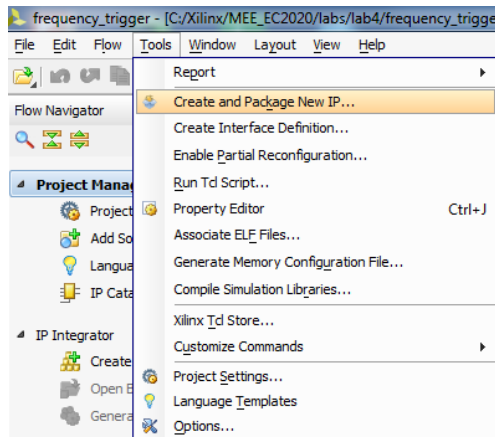


Figure 5. Create and Package IP option

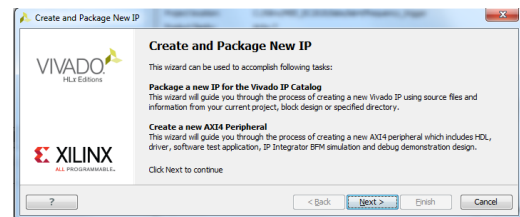


Figure 6. Create and Package IP dialog box

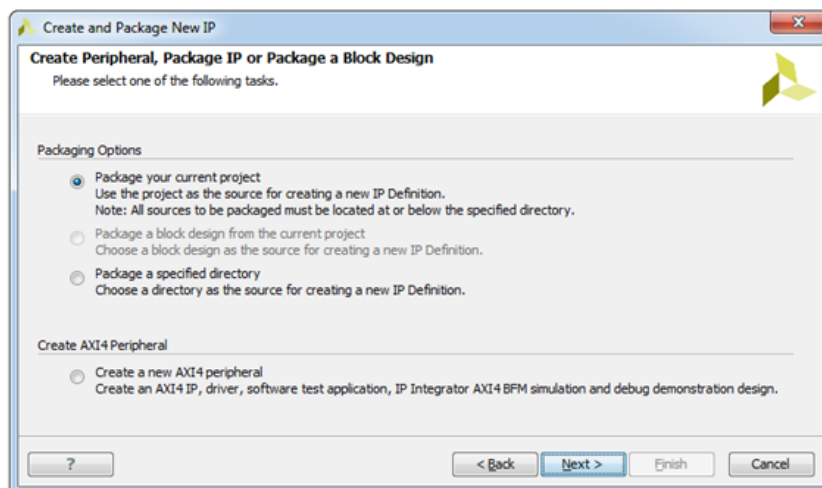


Figure 7. Choose Create Peripheral or Package IP dialog box

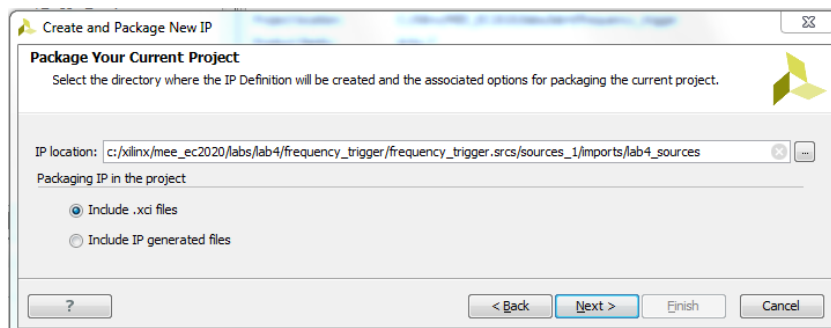


Figure 8. Package Your Current Project dialog box

2-2-5. In the **New IP Creation** dialog box, click **Finish**. If you have selected either **Package your current project** or **Package a specified directory** option, the **New IP Creation** dialog box opens automatically to summarize the information the wizard gathered about the project, and creates a basic IP package in a staging area (Figure 9).

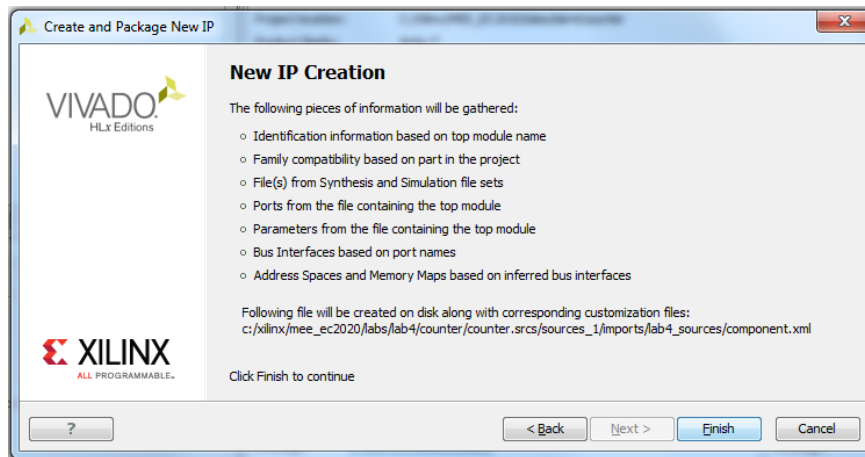


Figure 9. New IP Creation dialog box

2-2-6. The **Package IP - frequency_trigger** window will automatically appear on the right side of the Vivado IDE, see Figure 10.

Review the IP Packaging steps in the Package IP page:

- **Identification:** Information used to identify your IP
- **Compatibility:** Configure the parts and/or families of Xilinx devices that are compatible with your IP
- **File Groups:** Individual files for your IP are grouped into specific file groups
- **Customization Parameters:** Specify the parameters to customize your IP
- **Ports and Interfaces:** Top-level ports and interfaces for your IP
- **Addressing and Memory:** Specify the memory-maps or address spaces
- **Customization GUI:** Configure the parameters that appear on each page of the Customization GUI
- **Review and Package:** Summary of the IP and repackaging

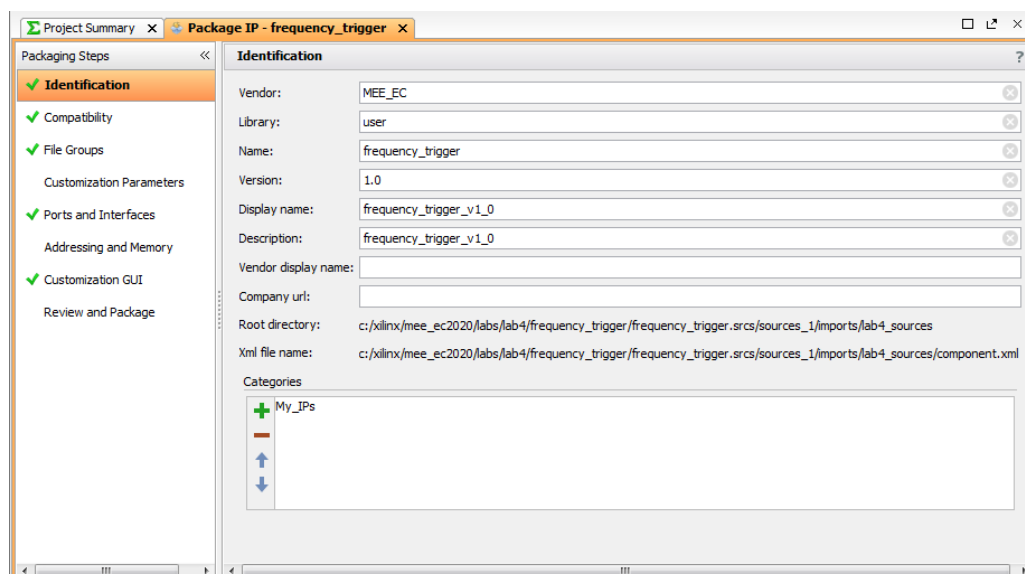


Figure 10. Identification window

2-2-7. In the **Package IP - frequency_trigger** window, in the Identification section, fill in fields as it is shown on Figure 10.

- 2-2-8.** After we finished with the IP **Identification**, select the **Review and Package** option in the **Package IP** window and check the specified project directory folder to make sure that the new archive file was added, see Figure 11.

The default naming convention for the archive is:

<vendor>_<library>_<name>_<version>.zip

In our case, the name of the zip file should be:

MEE_EC_user_frequency_trigger_1.0.zip

The user can change the default name and location of the archive by selecting the edit link next to the Create archive of IP name in the After Packaging selection, see Figure 11.

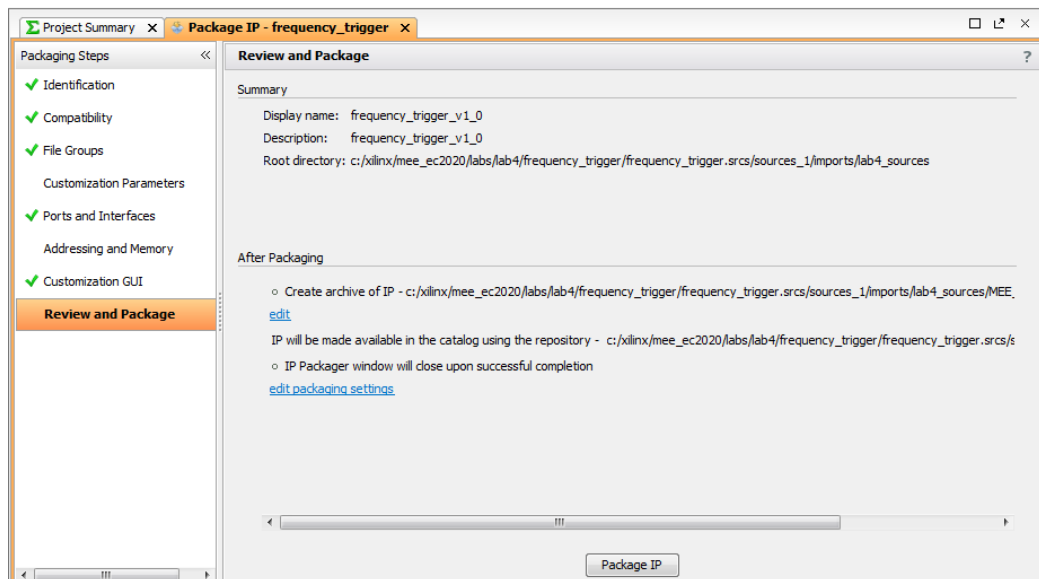


Figure 11. Review and Package window

- 2-2-9.** Click **edit** link next to the **Create archive of IP** name in the **After Packaging** selection to change the name and the location of the archive, see Figure 12.

- 2-2-10.** In the Package IP dialog box, change the Archive name to be:

MEE_modulator_frequency_trigger_1.0.zip.

- 2-2-11.** Before you change the **Archive location**, create a new folder, **ip_repository**, in the same folder where the **frequency_trigger** project was created. This new folder will be a place where we will keep all IPs (.zip files) that we will create.

- 2-2-12.** In the **Package IP** dialog box, change the Archive location to the new ip_repository folder, see Figure 12.

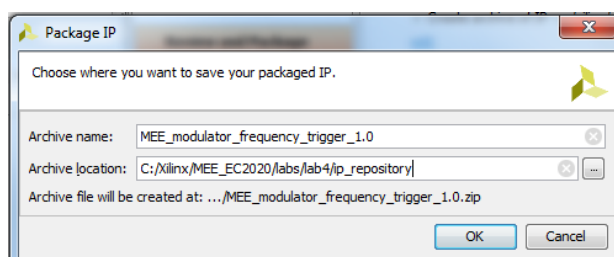


Figure 12. Package IP dialog box

- 2-2-13.** Click **OK** and you should see all the modifications that we made in the **After Packaging** sector of the **Review and Package** window, see Figure 13.

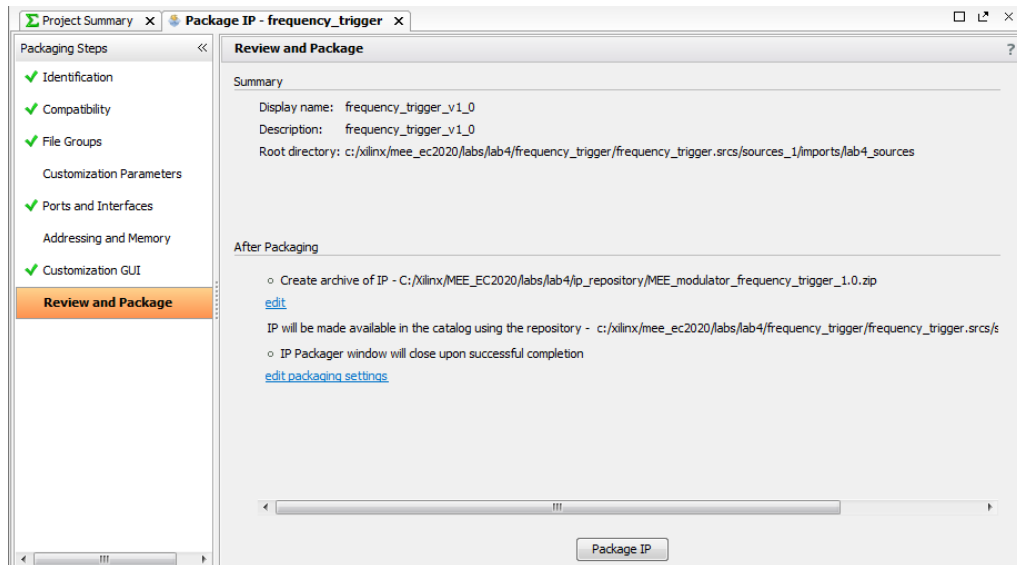


Figure 13. Package IP dialog box with selected new archive location

- 2-2-14.** If you are satisfied with the Package IP information, click the **Package IP** button at the bottom of the **Review and Package** window to finish with the **frequency_trigger** IP packaging process
- 2-2-15.** In the Flow Navigator, under the **Project Manager**, click **IP Catalog** command to verify the presence of our frequency_trigger IP in the **IP Catalog** (Figure 14)

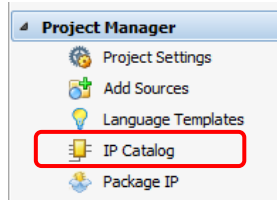


Figure 14. IP Catalog

- 2-2-16.** In the **IP Catalog**, search for the **frequency_trigger_v1_0** IP, see Figure 15. If you select the **frequency_trigger_v1_0** IP, all the data that we entered in the process of the IP creation should appear in the **Details** window.

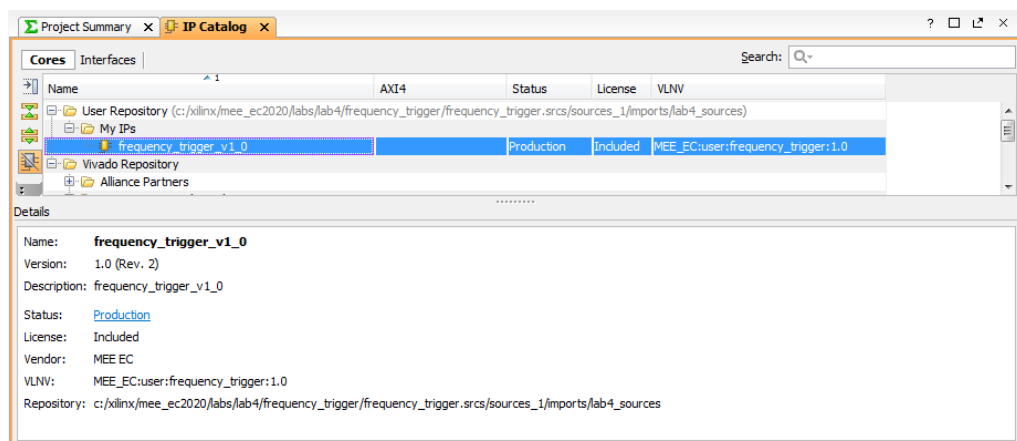


Figure 15. frequency_trigger IP in the IP Catalog

2-3. Adding Other Modules.

2-3-1. Now, when you know the procedure for IP creation, repeat previous steps to create the rest of the IPs (counter, digital_sine and pwm), necessary for the Modulator project. Note that you will have to create a **new project** for each IP, so close first the current one. They are the following:

Counter IP:

- Name the project "**counter**" when you start new project creation
- In the project creation process, in the **Add Source Files** dialog box, choose **counter_rtl.v** source file and click **OK**
- In the **Packager IP** wizard, in the **Review and Package** window, click **edit** link next to the **Create archive of IP** name in the **After Packaging** selection to change the name and the location of the archive:
 - Change the **Archive name** to be: **MEE_modulator_counter_1.0.zip**
 - Change the **Archive location** to the new **ip_repository** folder

Digital Sine IP:

- Name the project "**digital_sine**" when you start new project creation
- In the project creation process, in the **Add Source Files** dialog box, choose **sine_rtl.v** and **sine_values.dat** source files and click **OK**
- In the **Packager IP** wizard, in the **Review and Package** window, click **edit** link next to the **Create archive of IP** name in the **After Packaging** selection to change the name and the location of the archive:
 - Change the **Archive name** to be: **MEE_modulator_digital_sine_1.0.zip**
 - Change the **Archive location** to the new **ip_repository** folder

PWM IP:

- Name the project "**pwm**" when you start new project creation
- In the project creation process, in the **Add Source Files** dialog box, choose **pwm_rtl.v** and **frequency_trigger_rtl.v** source files and click **OK**
- In the **Packager IP** wizard, in the **Review and Package** window, click **edit** link next to the **Create archive of IP** name in the **After Packaging** selection to change the name and the location of the archive:
 - Change the **Archive name** to be: **MEE_modulator_pwm_1.0.zip**
 - Change the **Archive location** to the new **ip_repository** folder

Now that all IPs are created, it's time to create a new project where we will instantiate and connect these IPs. Close the current project.

2-4. Add packaged IP to the IP Catalog.

2-4-1. Create new Vivado project, **modulator_ip**, without adding any source file. The following steps will show you how to add packaged IP to the IP Catalog.

2-4-2. Using Windows Explorer, open **ip_repository** folder with packaged IPs (.zip files) and extract each IP separately to a directory with the same name.

2-4-3. Then, In the **Flow Navigator**, under the **Project Manager**, click **Project Settings** command and choose **IP** from the left pane

2-4-4. Select **Repository Manager** tab, see Figure 16. **Repository Manager** lets you add or remove user repositories and establish precedence between repositories.

2-4-5. In the **Repository Manager** window, click + icon to add the desired repository (Figure 16)

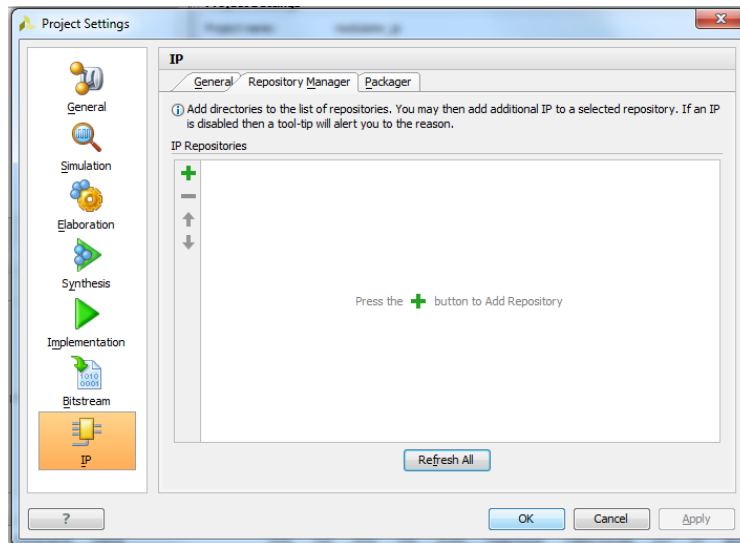


Figure 16. Repository Manager window

2-4-6. In the **IP Repositories** window, choose **ip_repository** folder and click **Select**

2-4-7. In the **Add Repository** dialog box, click **OK** to add the selected repository (*ip_repository* with 4 IPs) to the project, see Figure 17

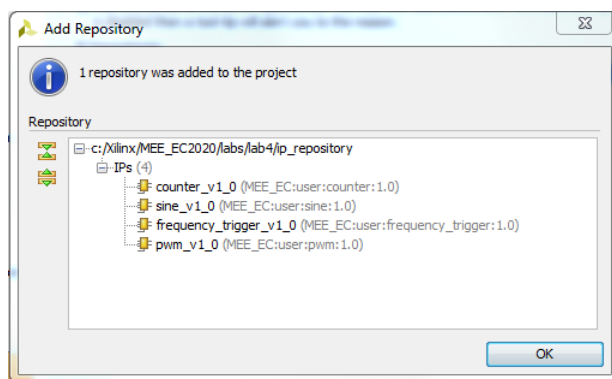


Figure 17. Add Repository dialog box

2-4-8. In the **Repository Manager** window, when **ip_repository** is added to the **IP Repositories** section, click **OK**, see Figure 18

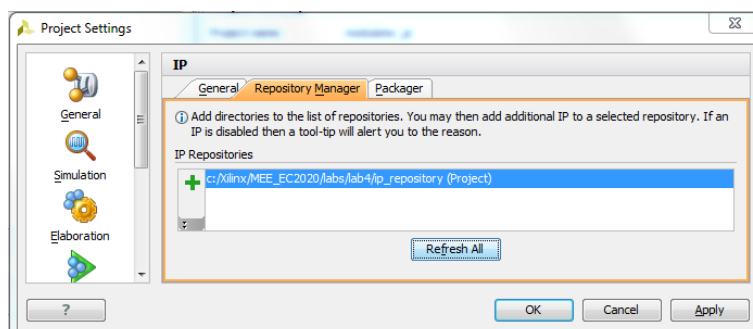


Figure 18. Repository Manager with selected ip_repository

- 2-4-9.** In the **Flow Navigator**, under the **Project Manager**, click **IP Catalog** command to verify the presence of the previously created IPs in the IP Catalog

In the next step you will instantiate a few IPs in the IP Integrator tool and then stitch them up to create an IP sub-system design. You will be introduced to the IP Integrator GUI, run design rule checks (DRC) on your design, and then integrate the design in a top-level design in the Vivado Design Suite. Finally, you will run synthesis and implementation process, generate bitstream file and run your design on the Basys3 development board.

IP Integrator

Step 3

3-1. Create a Block Design.

- 3-1-1.** In the **Flow Navigator**, expand **IP Integrator** and select **Create Block Design** command, see Figure 19.

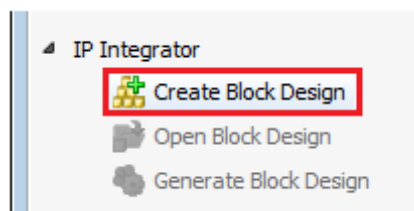


Figure 19. Create Block Design option

- 3-1-2.** In the **Create Block Design** dialog box, specify **modulator_ipi** name of the block design in the **Design name** field and click **OK**, see Figure 20.

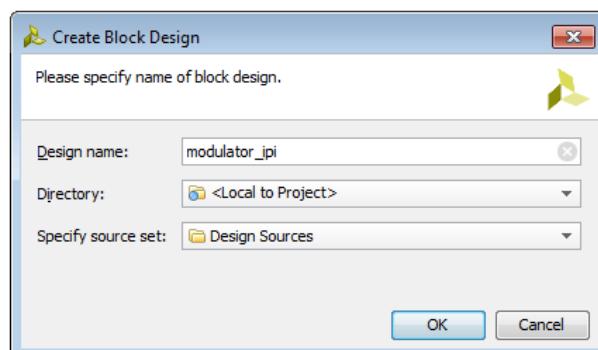


Figure 20. Create Block Design dialog box

The Vivado IDE will display a blank design canvas. You can quickly create complex subsystem by integrating IP cores in it, see Figure 21.

- 3-1-3.** The **modulator_ipi design** is empty. To get started, add IPs from the IP Catalog. You can do that on three ways (Figure 22):
- In the design canvas, right-click and choose **Add IP...** option, or
 - Use the **Add IP** link in the IP Integrator canvas, or
 - Click on the **Add IP** button in the IP Integrator sidebar menu.

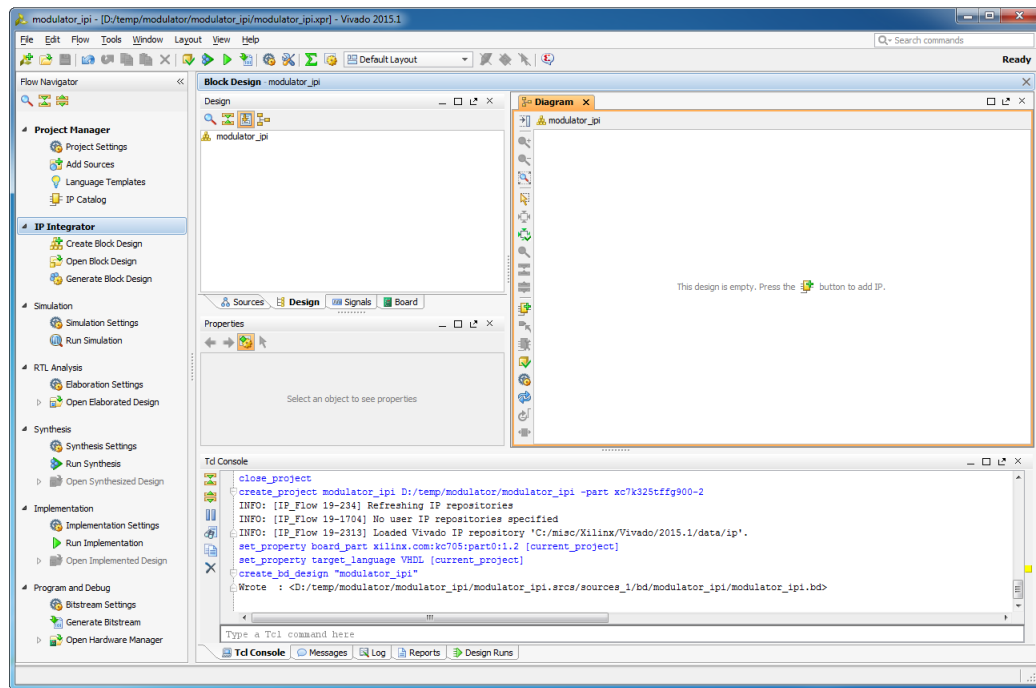


Figure 21. Vivado IDE with a blank design canvas

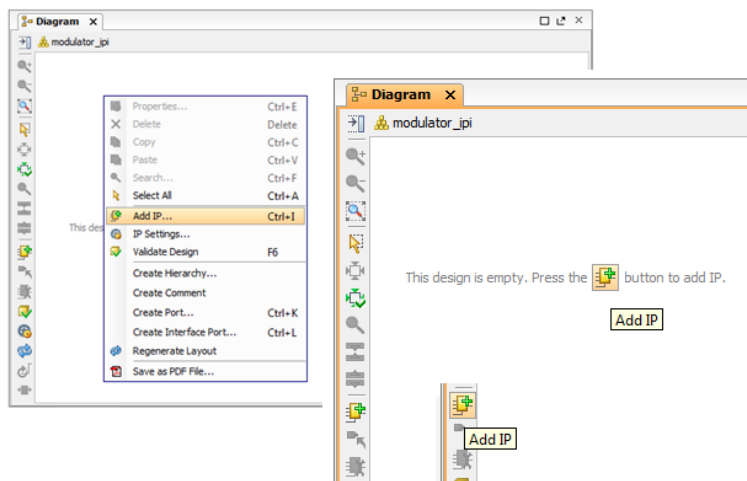


Figure 22. Add IP options

3-1-4. In the IP Catalog, search for the **frequency_trigger_v1_0** core, see Figure 23.

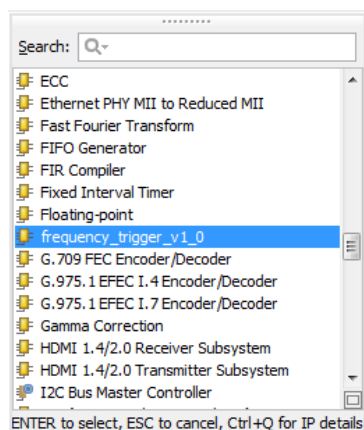


Figure 23. frequency_trigger_v1_0 core in the IP Catalog

- 3-1-5.** When you find it, press enter on the keyboard or simply double-click on the **frequency_trigger_v1_0** core in the IP Catalog and the selected core will be automatically instantiated into the IP Integrator design canvas, see Figure 24.

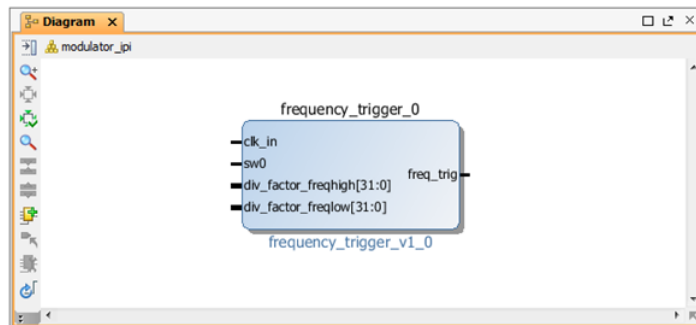



Figure 24. Automatically instantiated frequency_trigger_v1_0 core in the IP Integrator

- 3-1-6.** Right-click in the IP integrator canvas and select the **Add IP...** option to add the rest of the necessary IPs (**counter_v1_0**, **sine_v1_0** and **pwm_v1_0**). At this point, the IP Integrator canvas should look like as it is shown on the Figure 25.

Note: You can select **Regenerate Layout** in the right click menu or using the side button 

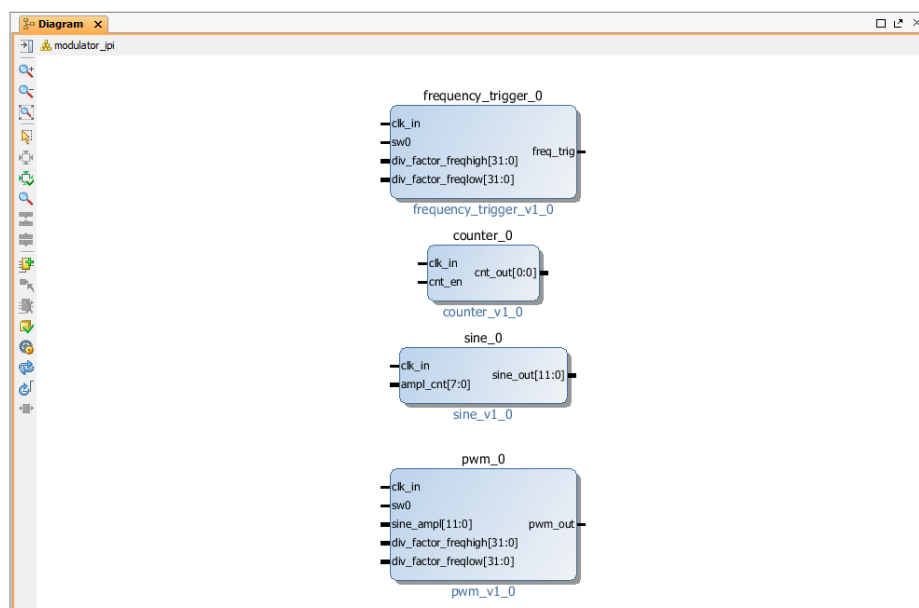


Figure 25. IP Integrator design canvas with all four instantiated IPs

- 3-1-7.** Double-click on the each of the IP cores to re-customize them according to what is defined in Figure 26. This configures each of the IPs in your block design.

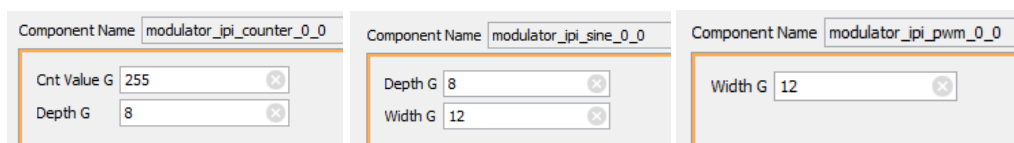


Figure 26. IP Customization

- 3-1-8.** After we re-customize all four IPs, the IP Integrator canvas should look like as it is shown on the Figure 27.

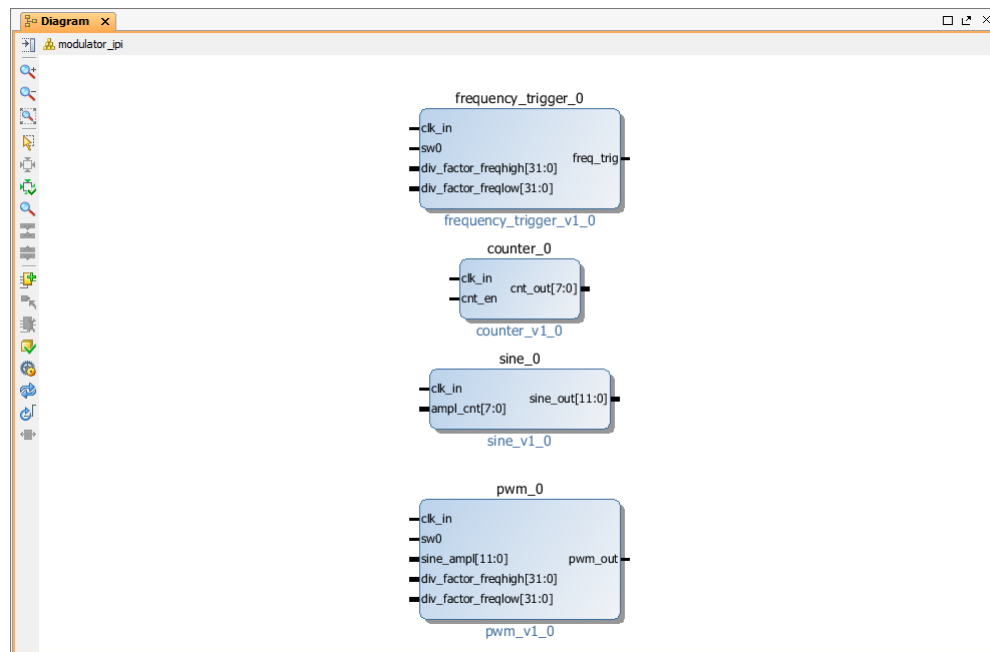


Figure 27. IP Integrator design canvas with all four re-customized IPs

3-2. Adding Constants.

- 3-2-1.** The last IP necessary for our design is the **Constant** IP. Add **Constant** IP four times into the block design.

Two **Constant** IP instances will be connected to the **div_factor_freqhigh(31:0)** and **div_factor_freqlow(31:0)** ports of the **frequency_trigger_v1_0** module and remaining two instances to the **div_factor_freqhigh(31:0)** and **div_factor_freqlow(31:0)** ports of the **pwm_v1_0** module, see Figure 28.

- 3-2-2.** Double-click on the first **Constant (xlconstant_0)** block and set the **Const Width** value to **32** and **Const Value** value to **110592**, see Figure 29.

- **Const Width** to **32** - because **div_factor_freqhigh** port that we would like to connect to is 32-bit wide
- **Const Value** to **110592** - because 110592 is the number that divides frequency of the input clock signal (100 MHz) to the required frequency.

- 3-2-3.** Do the same procedure with the second **Constant (xlconstant_1)** IP block. Set the **Const Width** value to **32** and **Const Value** value to **389120**.

- 3-2-4.** In the third **Constant (xlconstant_2)** IP block, set the **Const Width** value to **32** and **Const Value** value to **27** ($110592/4096=27$)

Note: Const Value to **27** ($110592/4096=27$), because PWM module must operate at 2^{width} ($2^{12}=4096$) higher frequency then the Sine module. This is required in order to generate correct pwm signal.

- 3-2-5.** In the forth Constant (xlconstant_3) IP block, set the Const Width value to **32** and Const Value value to **95** ($389120/4096=95$)

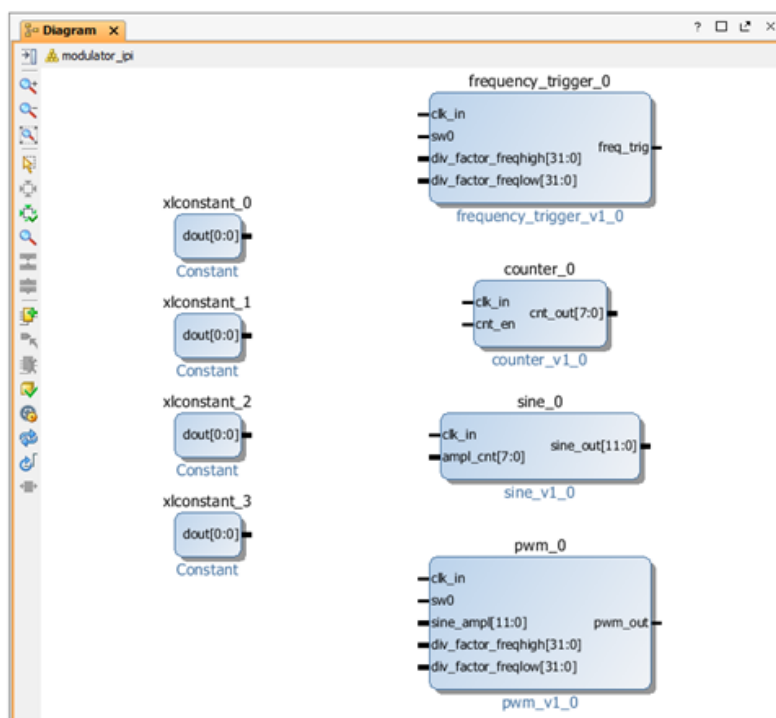


Figure 28. IP Integrator design canvas with instantiated Constant IPs

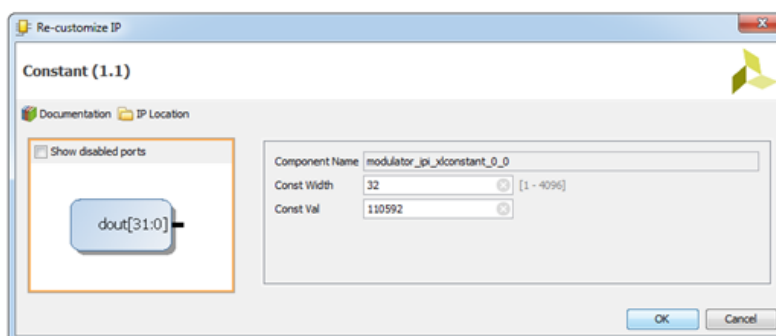


Figure 29. Constant block re-customization dialog box

3-3. Connect IPs between themselves.

By default, the Vivado simulator adds signals to the waveform configuration using a short name with the hierarchy reference removed. For some signals, it is important to know to which module they belong.

3-3-1. First step will be to create new ports.

- Select **clk_in** pin, right-click on it and select **Create port...** option, see Figure 30
- In the **Create Port** dialog box, make sure the "Connect to 'clk_in'seleted pin" is checked, leave all other parameters unchanged and click **OK**, see Figure 31
- Repeat the same procedure for **sw0** and **pwm_out** pins. After these modifications, the IP Integrator design canvas should look like as it is shown on Figure 32.

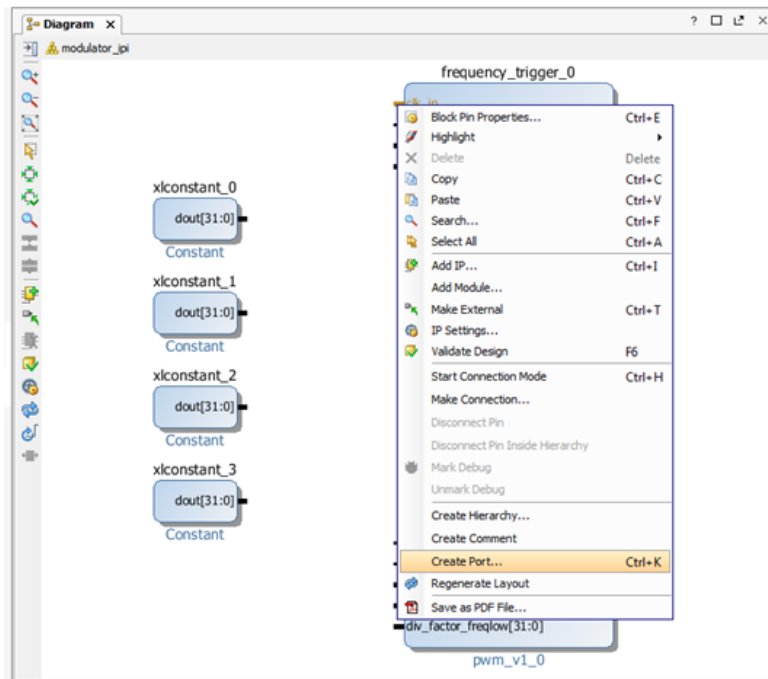


Figure 30. Create Port option Create Port option

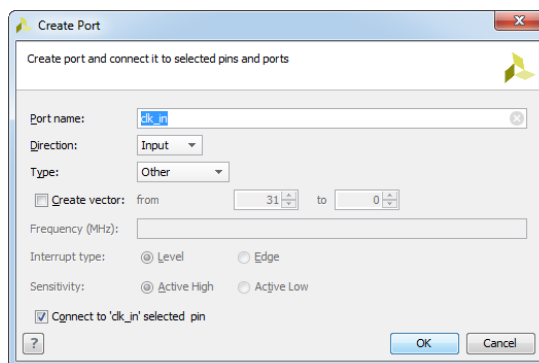


Figure 31. Create Port dialog box

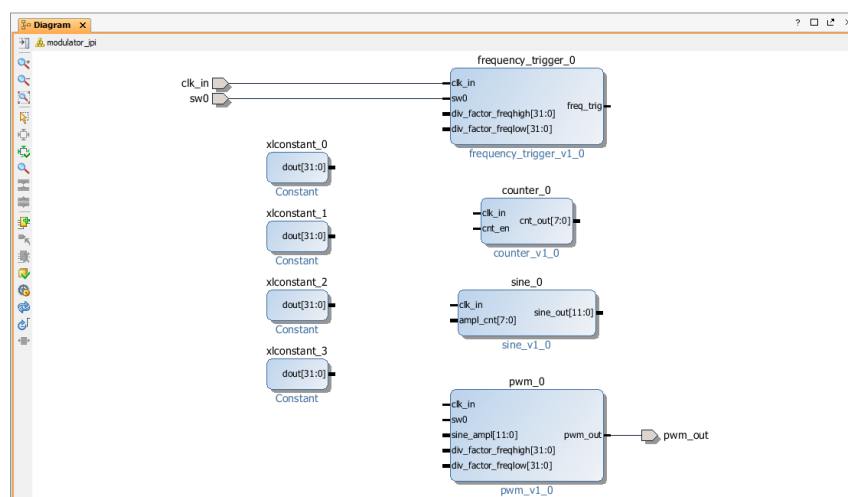


Figure 32. IP Integrator design canvas with new ports

3-3-2. Next step will be to connect the IPs:

Place the cursor on top of the desired pin and you can notice that the cursor changes into a pencil indicating that a connection can be made from that pin. Clicking the left mouse button a

connection starts. Click and drag the cursor from one pin to another. You must press and hold down the left mouse button while dragging the connection from one pin to another. As you drag the connection wire, a green checkmark appears on the port indicating that a valid connection can be made between these points.

The Vivado IP Integrator highlights all possible connections points in the subsystem design as you interactively wire the pins and ports. Release the left mouse button and Vivado IP integrator makes connection between desired ports. Repeat this procedure until all the pins become associated, click **Regenerate Layout** (Figure 33)

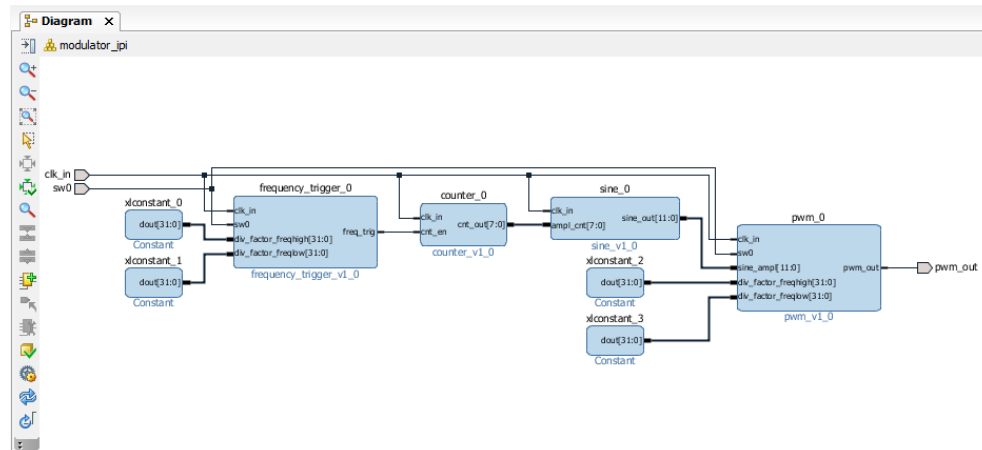


Figure 33. IP Integrator design canvas with connected IPs

- 3-3-3.** From the sidebar menu of the design canvas, run the IP subsystem design rule checks by clicking the **Validate Design** button. In the **Validate Design** dialog box, click **OK**.

Alternatively, you can do the same by selecting **Tools -> Validate Design** from the main menu, or by clicking the design canvas and selecting **Validate Design** button from the main toolbar menu, see Figure 34.

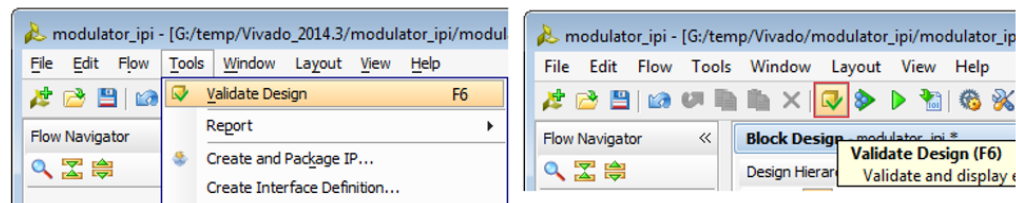


Figure 34. Validate Design option from the main menu

- 3-3-4.** At this point, you should save the IP integrator design. Use the **File -> Save Block Design** command from the main menu to save the design.
- 3-3-5.** In the **Sources** window, select **modulator_ipi**, right-click on it and choose **Create HDL Wrapper...** option, see Figure 35.

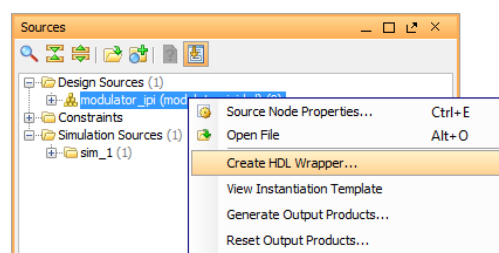


Figure 35. Create HDL Wrapper option

- 3-3-6.** In the **Create HDL Wrapper** dialog box, select **Let Vivado manage wrapper and auto-update** option and click **OK**, see Figure 36.
- 3-3-7.** After the HDL wrapper is generated, you should see it in the **Sources** window, see Figure 37.

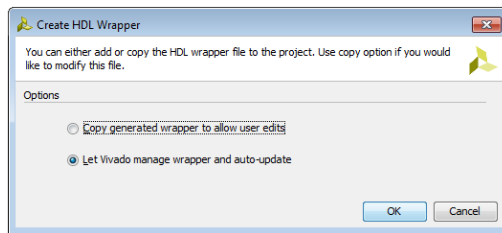


Figure 36. Create HDL Wrapper dialog box

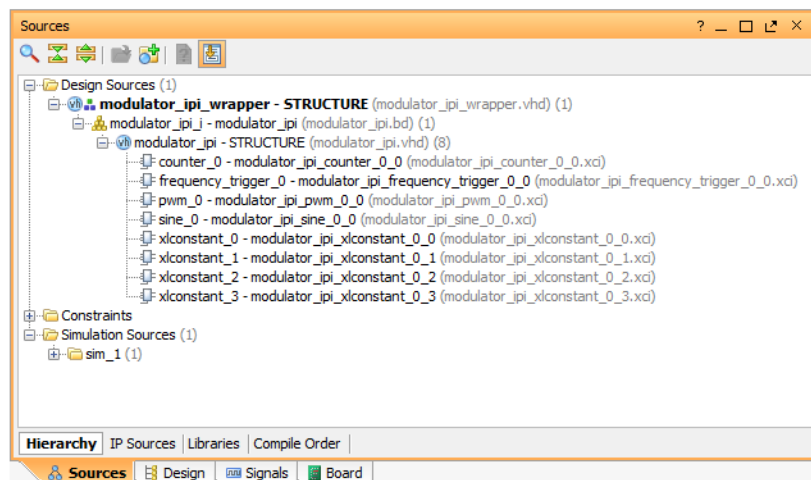


Figure 37. Sources window with generated modulator_ipi HDL wrapper

- 3-3-8.** Under **IP Integrator**, click **Generate Block Design** and select **Generate** (Figure 38). The option “out of context per IP” can significantly reduce synthesis run times because the IP cache can be used with this option to prevent Vivado synthesis from regenerating output products for specific IP if they do not change.

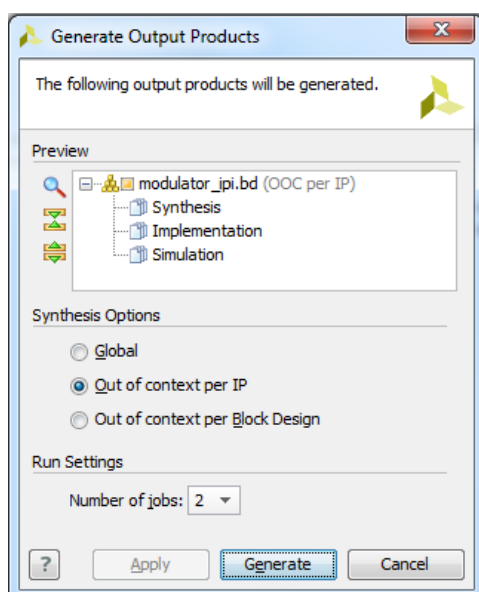


Figure 38. Generate Block Design window

- 3-3-9.** This mode creates an out-of-context (OOC) synthesis run and design checkpoint file (DCP) for every IP that is instantiated in the design. Notice that each IP in the BD is also marked with a filled square that indicates the IP is marked as OOC. The Design Runs window lists synthesis runs for each IP used in the Block Design, as shown in Figure 39.

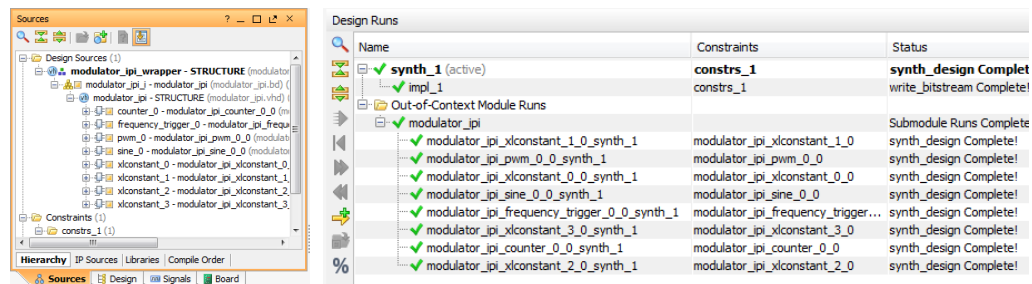


Figure 39. Design Runs Window for Out-of-Context per IP Synthesis

Generation of the individual output products in OOC per IP mode takes longer than a single global synthesis run; however, runtime improvements are realized in subsequent runs because only the updated blocks or IP are re-synthesized instead of the whole top-level design. In addition, with the IP Cache enabled (you can change the IP cache settings from the Settings > IP dialog box), Vivado synthesis can provide even greater runtime improvements because the only IP to re-synthesize have been re-customized or were impacted from parameter propagation.

- 3-3-10.** The last step in our design will be to create and add **modulator_ipi_rtl.xdc** constraints file. The content of the **modulator_ipi_rtl.xdc** constraints file is shown in the text below.

```
# CLK source 100 MHz
set_property -dict { PACKAGE_PIN W5  IOSTANDARD LVCMOS33 } [get_ports { clk_in }];
# SW0
set_property -dict { PACKAGE_PIN V17  IOSTANDARD LVCMOS33 } [get_ports { sw0 }];
# PWM out to Led
set_property -dict { PACKAGE_PIN U16  IOSTANDARD LVCMOS33 } [get_ports { pwm_out }];

# Timing
create_clock -period 10.000 -name clk_in -waveform {0.000 5.000} [get_ports clk_in]
```

- 3-3-11.** Synthesize your design with **Run Synthesis** option from the **Flow Navigator**. Your synthesized design should now include input and output buffers (Figure 40).

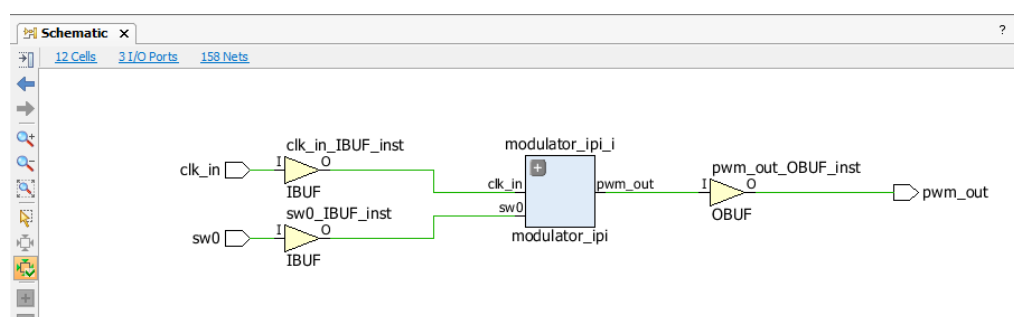


Figure 40. Synthesized schematic

- 3-3-12.** Implement your design with **Run Implementation** option from the **Flow Navigator**.
- 3-3-13.** Generate bitstream file with **Generate Bitstream** option from the **Flow Navigator**.
- 3-3-14.** Program your **Basys3** device and observe the **pwm** signal in LED0, with different frequencies, depending on SW0.