Lab 6: Basic Hardware Design with the Vivado Design Suite and PetaLinux Tool

Introduction

The real power of Linux on an FPGA platform comes when the custom logic resources of the FPGA are combined with the usability and software infrastructure of the operating system. Xilinx provides a powerful tool to ease FPGA design. The purpose of this lab is to go through the Vivado® Design Suite and Vivado IP integrator (IPI) and learn how to use the PetaLinux tools to configure an embedded Linux system for a new platform.

In the previous labs, you learned how to configure and build embedded Linux systems. Embedded Linux always runs on a certain hardware platform. In this lab, you will create a Linux-capable FPGA platform from scratch.

Objectives

After completing this lab, you will be able to:

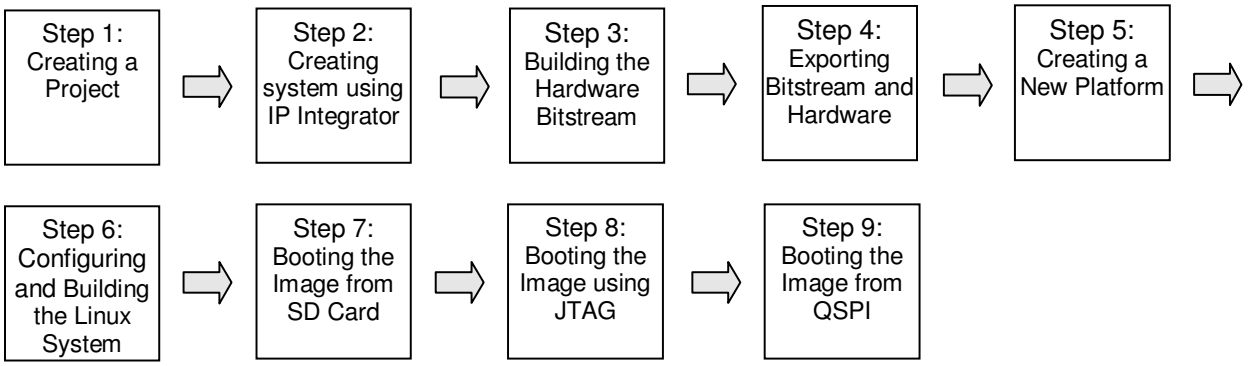
- Create a Vivado Design Suite project for a Zynq® All Programmable SoC system
- Use the PetaLinux tools to create a new embedded Linux target for the hardware platform
- Build and boot the new embedded Linux system

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



Creating a Vivado Design Suite Project — Step 1

The Vivado Design Suite provides a design flow to help you manage a complex system design. In this step, you will learn how to build a Vivado Design Suite project.

1.1. Start the Vivado Design Suite and set the project path.

1.1.1. Create and change to the lab directory by using the following commands:

```
[host] $ mkdir ~/emblnx/labs/lab6

[host] $ mkdir ~/emblnx/labs/lab6/hardware

[host] $ cd ~/emblnx/labs/lab6/hardware
```

1.1.2. Run the following command to launch the Vivado Design Suite:

```
[host] $ vivado
```

If the program does not start then source the program settings by executing the following command:

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

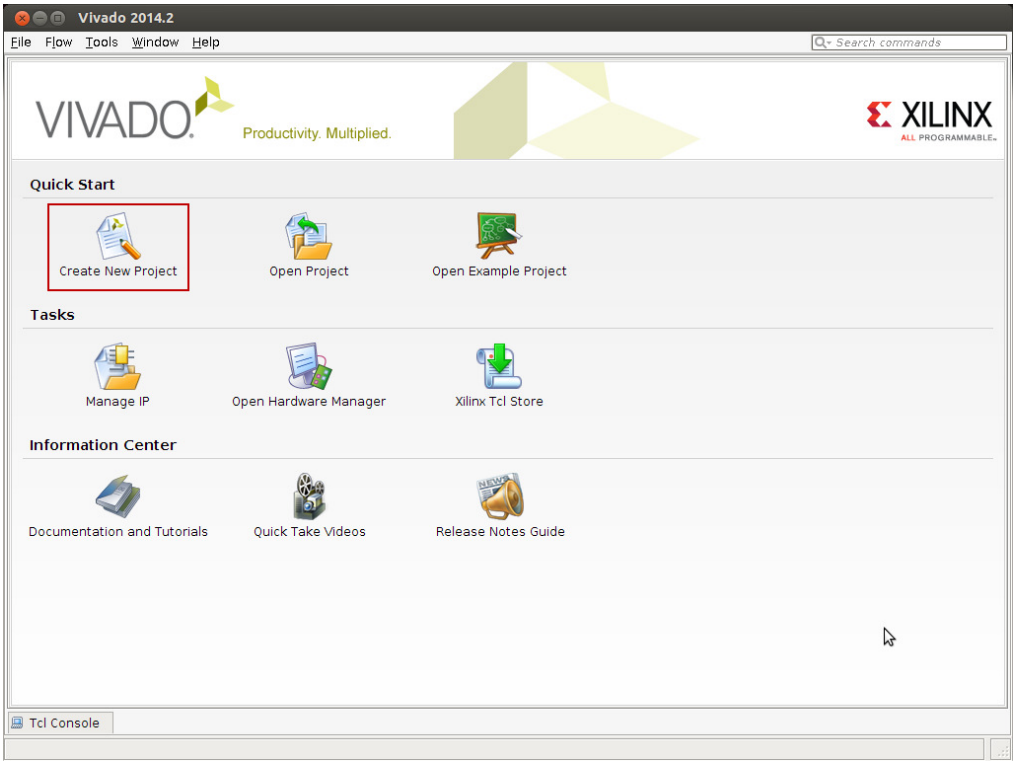


Figure 1. Vivado Design Suite's Getting Started screen

If a Xilinx License Error dialog box displays, click **OK** to continue. Click **Close** in the Xilinx License Configuration Manager dialog box if displayed. You will need a valid license to run the program. Obtain and install the valid license.

1.1.3. Click **Create New Project**.

This will launch the New Project Wizard.

1.1.4. Click **Next**.

1.2. You will now encounter a series of dialog boxes asking you to enter different pieces of information describing the project.

1.2.1. Enter **lab6** in the Project name field.

1.2.2. Verify that **/home/petalinux/emblnx/labs/lab6/hardware** is the Project location field. Alternatively, you can use the browse feature to navigate to where you want the project to reside.

1.2.3. De-select the **Create project subdirectory** option as leaving this checked will create an unnecessary level of hierarchy for the lab.

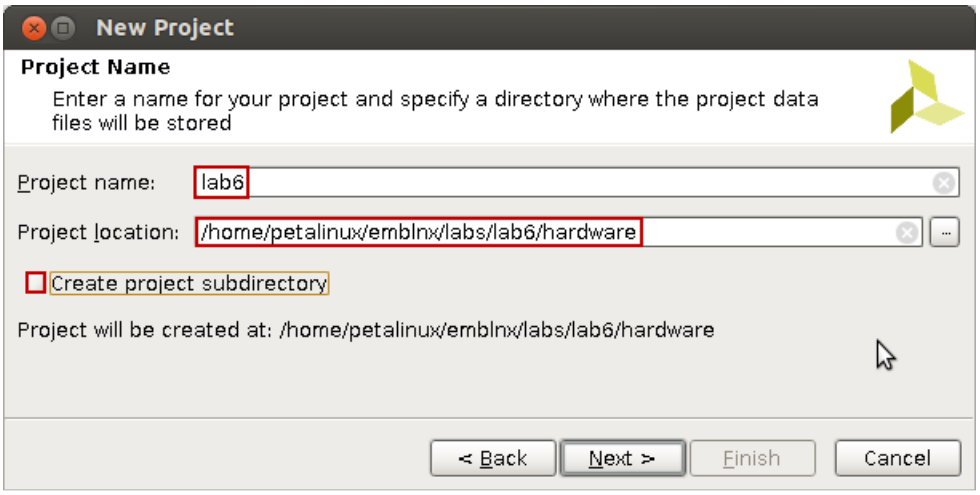


Figure 2. Entering the Project Name and Location

1.2.4. Click **Next**.

This next dialog box invites you to choose between an RTL project or a post-synthesis project. Selecting an RTL Project enables you to add or create new HDL files and synthesize them whereas the Post-synthesis Project requires pre-synthesized files. When creating an empty design, an RTL Project is used.

1.2.5. Select **RTL Project**.

1.2.6. Select **Do not specify sources at this time**, which makes the created project blank. Click **Next**.

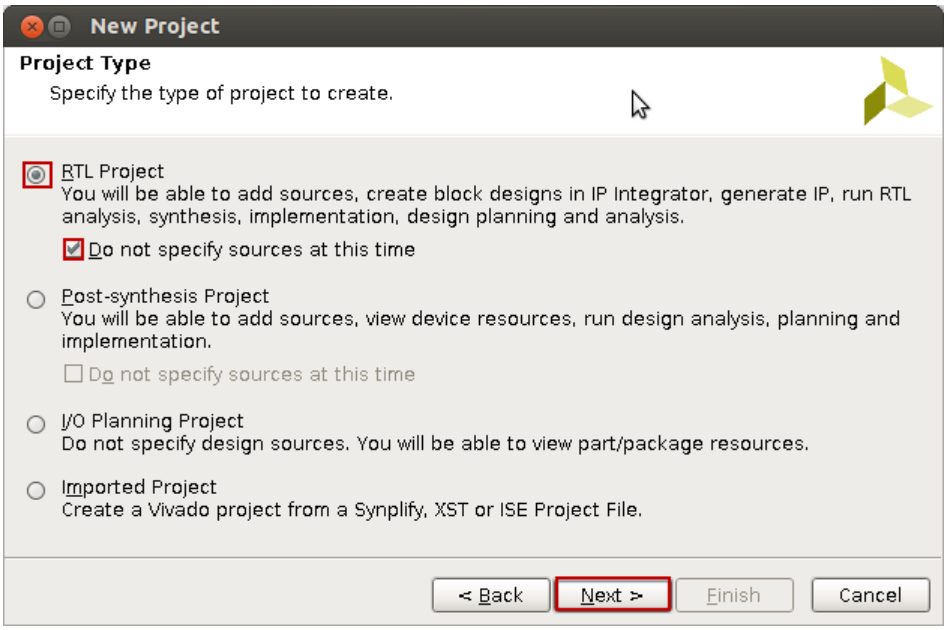


Figure 3. Selecting Project Type

1.3. You can now select the target board by first filtering by board and then selecting the board. If you are not using a supported board, you will need to filter by part.

1.3.1. Under the Specify area, select **Boards**. Select **your board** (ZedBoard). Click **Next.**

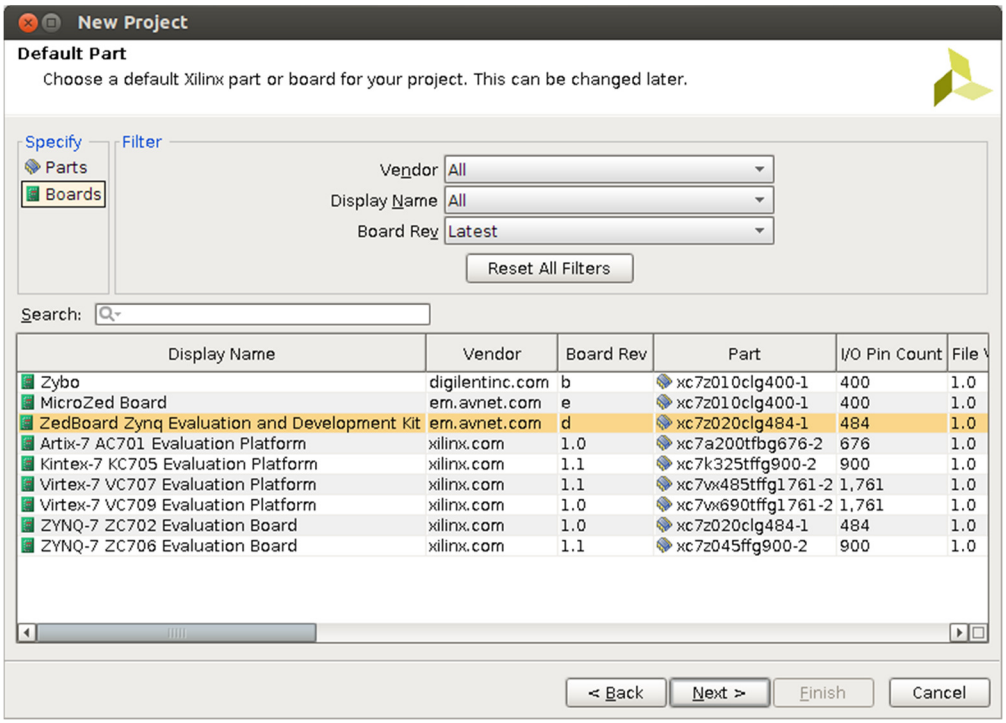


Figure 4. Selecting a part using the Boards filter

A summary of your project is displayed. If you want to change any of the information that you entered, you can do so now by clicking **Back** until you reach the correct dialog box and make the correction, or you can create the project now and edit the project properties, add or remove files, etc. later.

1.3.2. Click **Finish**.

Your project is created and the Vivado Design Suite GUI is launched.

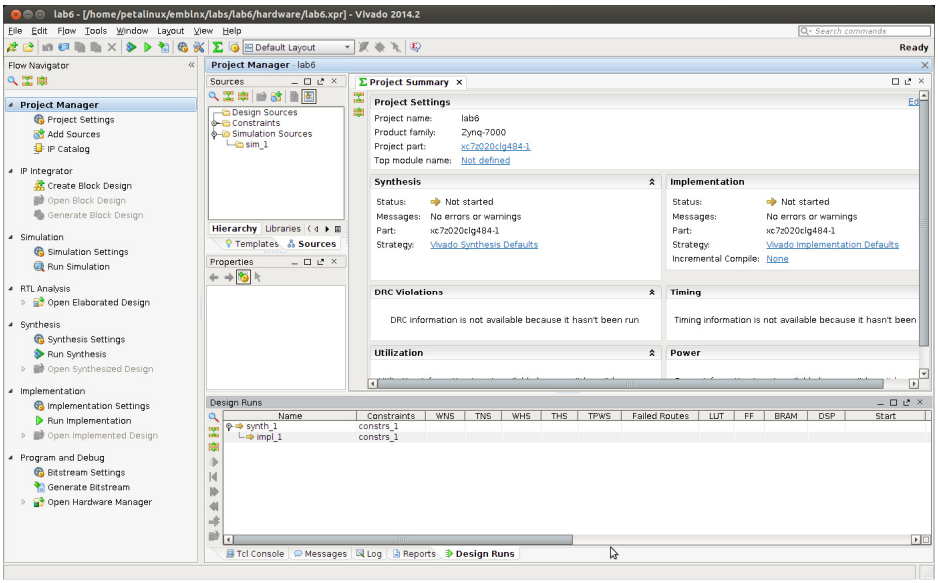


Figure 5. Vivado Design Suite GUI

Creating System Using the IP Integrator — Step 2

2-1. Use the Vivado IP integrator to create a new block design, and generate the ARM® Cortex™-A9 processor-based system for the targeted board.

2-1-1. In the Flow Navigator, click **Create Block Design** under IP Integrator.

2-1-2. Enter **system** for the design name.

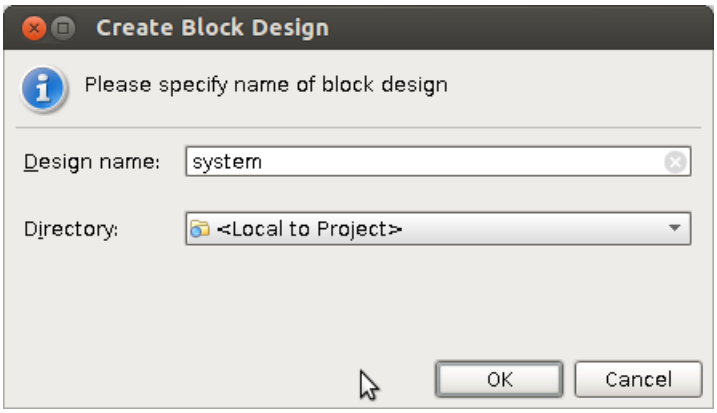


Figure 6. Creating a new block design

2-1-3. Click **OK**.

2-1-4. IP from the IP Catalog can be added in different ways. Click **Add IP** in the message at the top of the Diagram panel (1), or click the **Add IP** icon in the block diagram side bar (2), or press <**Ctrl+I**> or right-click anywhere in the diagram workspace and select **Add IP** (3)

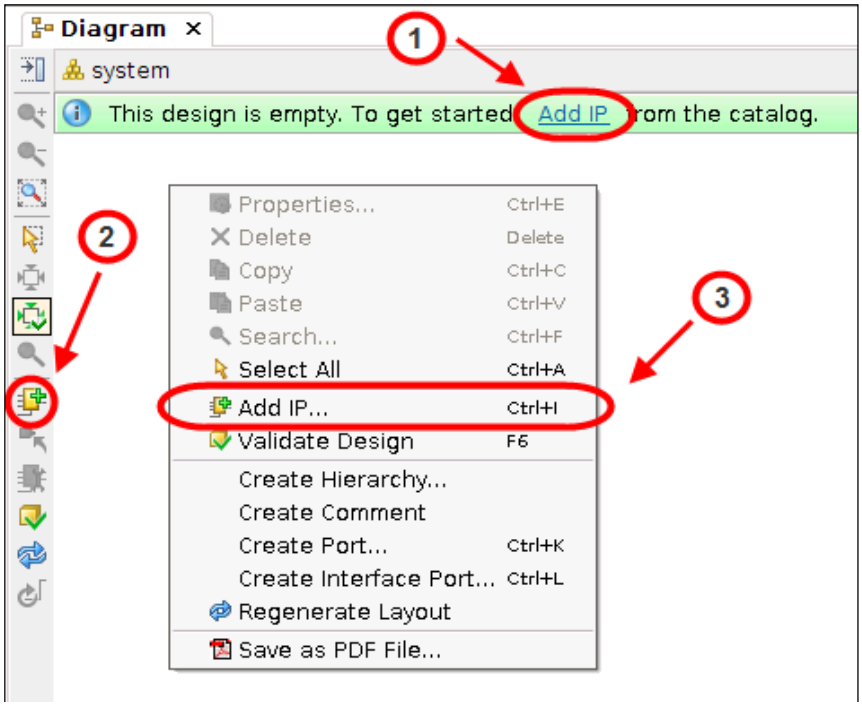


Figure 7. Adding IP to block diagram

Once the IP catalog opens, you can search for the Zynq All Programmable Soc processor block.

2-1-5. Enter part of the processor name into the Search field to narrow the search parameters (1).

For example, type **zynq** for the Zynq All Programmable SoC. Note that the Zynq PS IP will only appear when Zynq-7000 All Programmable SoCs or boards with these parts have been selected for the design.

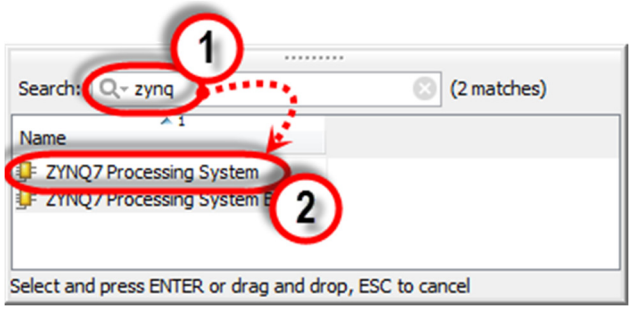


Figure 8. Narrowing search parameters (Zynq AP SoC)

2-1-6. Double-click the processor name entry to add its IP block to the design (2).

The ZYNQ block will be added with the instance name, *processing_system7_0*.

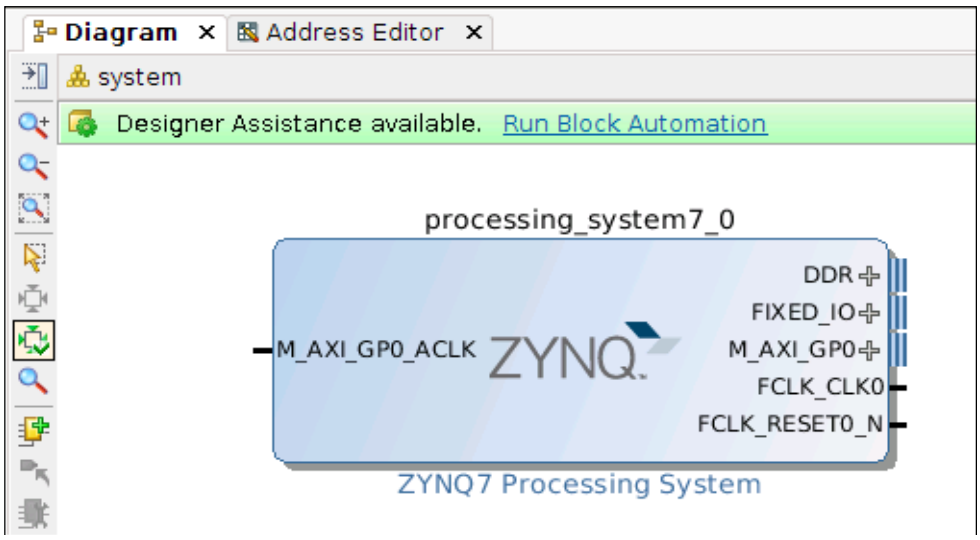


Figure 9. Zynq Processing system IP

Notice the message at the top of the Diagram window that *Designer Assistance* available.

2-1-7. Click **Run Block Automation** and select /**processing_system7_0**.

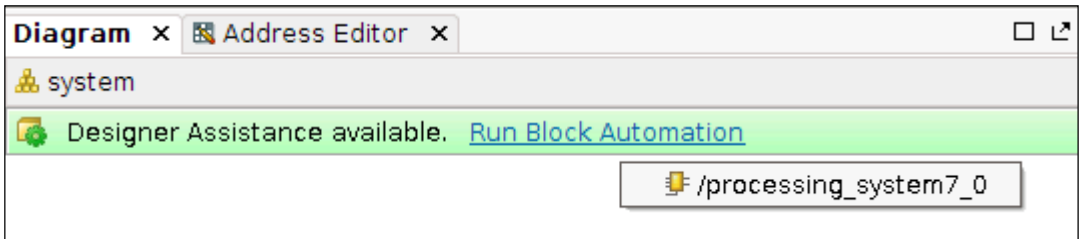


Figure 10. Designer Assistance message

2-1-8. Click **OK** when prompted to run automation.

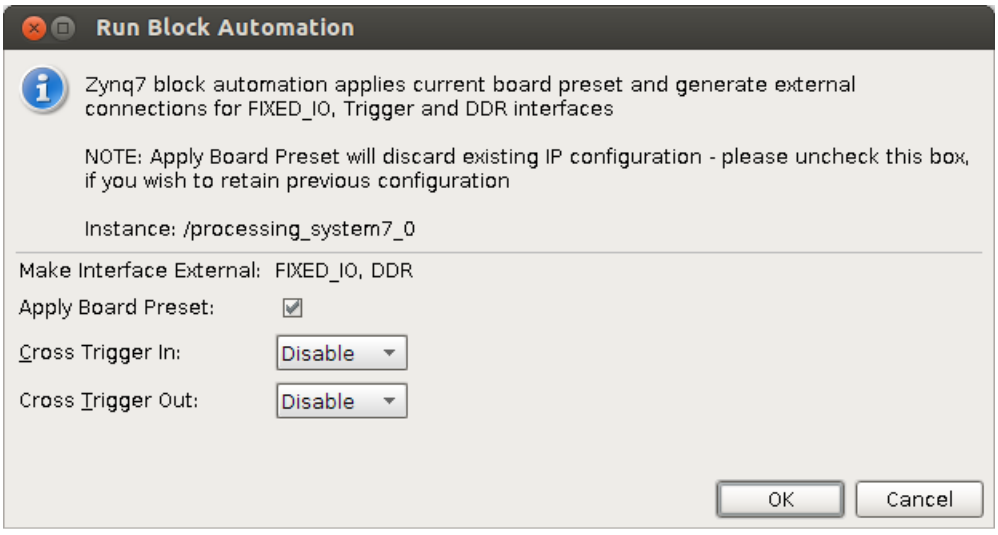


Figure 11. Run Block Automation

Once block automation has been completed, notice that ports have been automatically added for the DDR and fixed I/O, and some additional ports are now visible. A default configuration for the Zynq All Programmable SoC has been applied, which you will now modify.

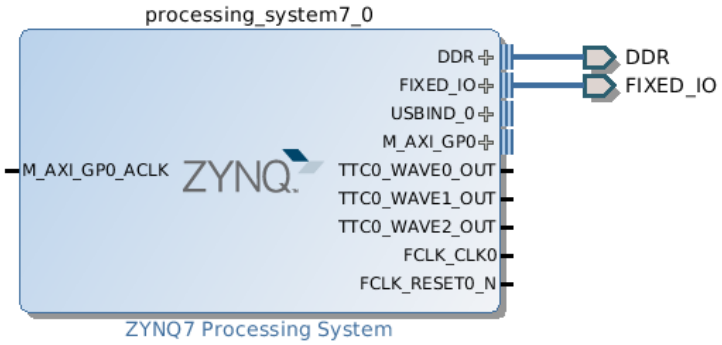


Figure 12. Zynq Block with DDR and Fixed I/O Ports

2-2. Customize the ZYNQ7 processing system (PS).

2-2-1. Double-click the **ZYNQ7 Processing System** block.

The Re-customize IP dialog box opens.

Take a moment to view the architecture of the PS. Click the blocks in green to view the available settings. There are dozens of registers that need to be configured. Some of these registers include DDR configuration information. Because this is an established board, it is much easier to import an existing configuration and modify it to your needs.

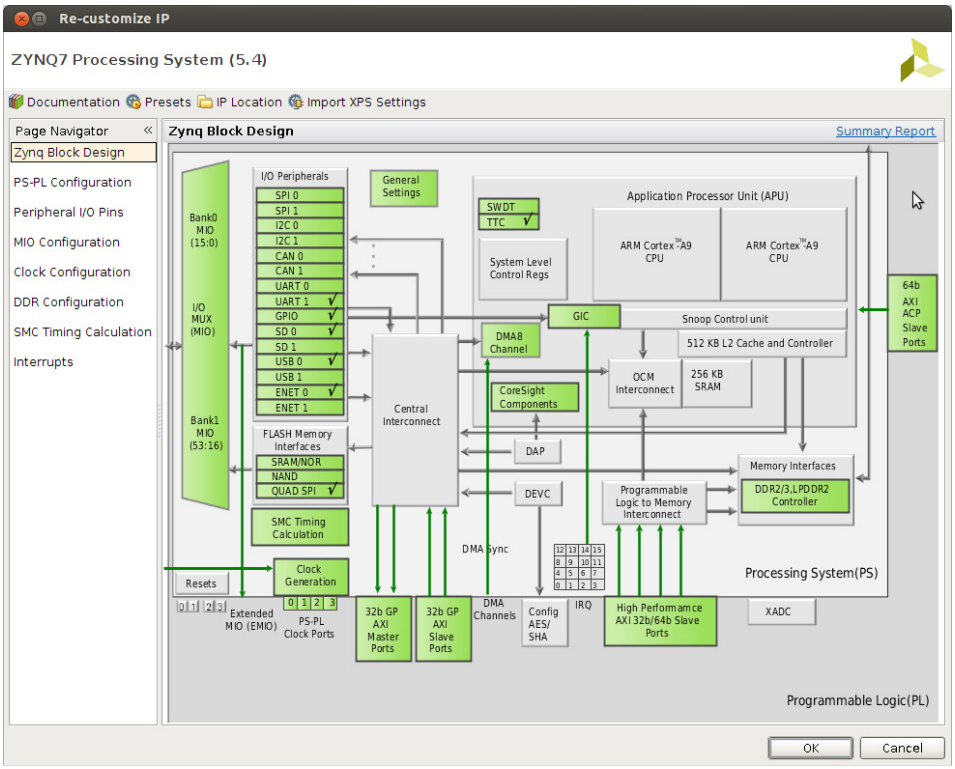


Figure 13. Zynq system configuration view

The following is a list of requirements for a Zynq All Programmable SoC hardware project to boot Linux:

- One triple timer counter (TTC):
  - If there are multiple TTCs that are enabled, the Zynq All Programmable SoC Linux kernel uses the first TTC block from the device tree.
  - Make sure that the TTC is not used by anything else.
- External memory controller (DDR controller)
- UART for serial console
- Non-volatile memory (QSPI Flash, SD, for example)
- Ethernet

2-2-2. Select **MIO Configuration** in the Page Navigator.

2-2-3. Expand **I/O Peripherals**.

2-2-4. De-select **USB0** and **GPIO MIO**.

2-2-5. Select **Clock Configuration** in the Page Navigator.

2-2-6. Expand **PL Fabric Clocks**.

2-2-7. Observe that **FCLK_CLK0** is enabled with **100** MHz frequency.

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

2-3. Add the GPIO instance that connects to the switches.

2-3-1. Click the **Add IP** icon  and search for **gpio**.

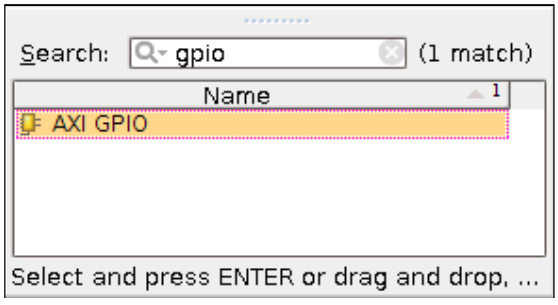


Figure 14. Searching for GPIO IP

2-3-2. Double-click the **AXI GPIO** to add the core to the design.

The core will be added to the design and the block diagram will be updated.

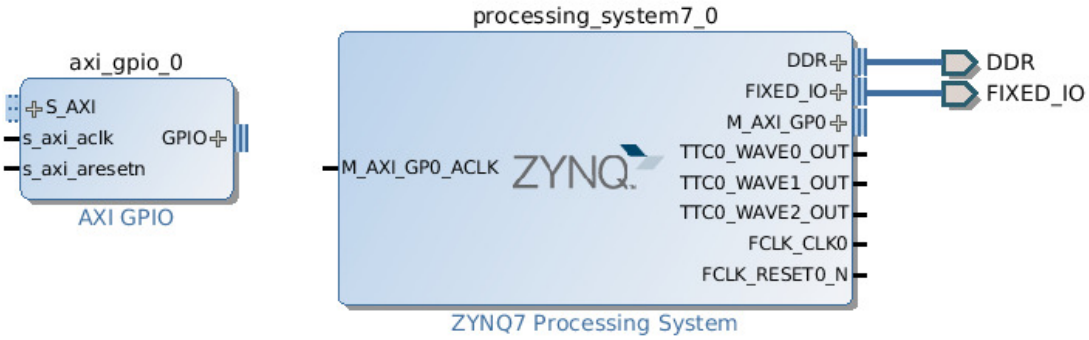


Figure 15. Zynq system with AXI GPIO added

2-3-3. Click the **AXI GPIO** block to select it.

2-3-4. In the *Block Properties* tab, change the name to **sw_8Bits**.

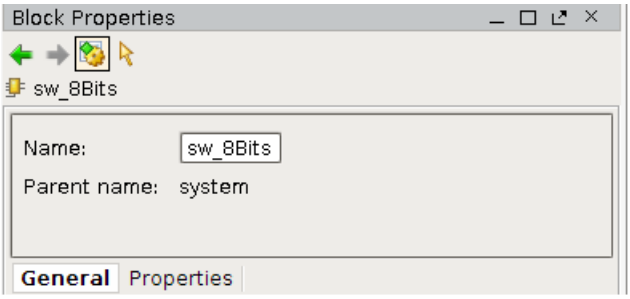


Figure 16. Changing the AXI GPIO default name to sw_8Bits

2-4. Customizing the sw_8Bits instance.

2-4-1. Double-click the **AXI GPIO** block to open the customization window.

As the target board was selected during project creation, and a board support package is available for the targeted board, the Vivado Design Suite has the knowledge of the available resources on the targeted board.

2-4-2. Select **Generate Board based IO Constraints** in the Board tab.

2-4-3. Under Board Interface for GPIO, click **Custom** to view the drop-down list options.

2-4-4. Select **sws 8bits**.

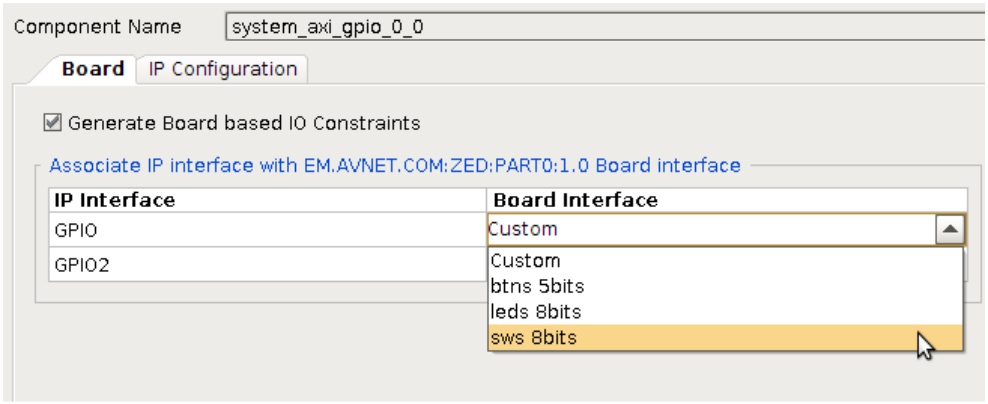


Figure 17. Configuring the GPIO (sws_8bits) instance

2-4-5. Select the **IP Configuration** tab.

Note that the GPIO width is set to 8. Also notice that the peripheral can be configured for two channels.

But because you want to use only one channel without an interrupt, leave GPIO Supports Interrupts and Enable Channel 2 unchecked.

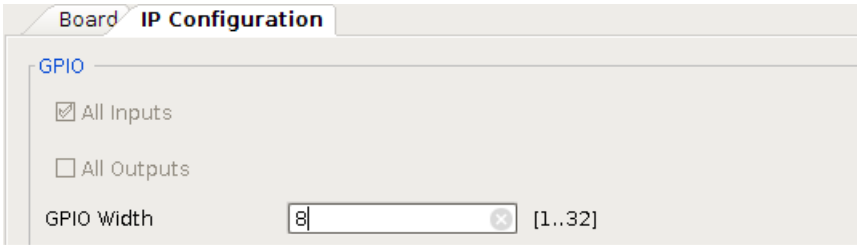


Figure 18. GPIO width set according to the board resource

2-4-6. Click **OK** to close the customization window.

2-5. Use Run Connection Automation to make the connections for the sw_8Bits instance.

2-5-1. Notice that **Designer Assistance** is available.

2-5-2. Click **Run Connection Automation** and select **/sw_8Bits/S_AXI**.

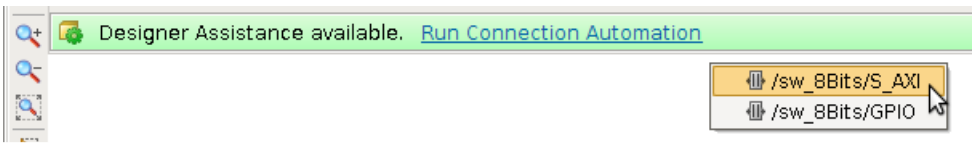


Figure 19. Selecting Run Connection Automation for sw_8Bits GPIO

2-5-3. Click **OK** when prompted to automatically connect the master and slave interfaces.

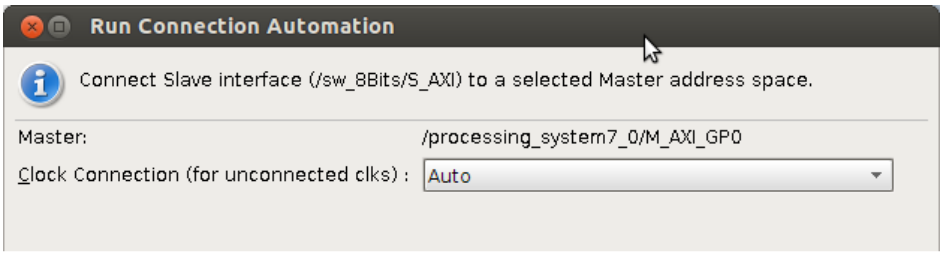


Figure 20. Running Connection Automation for sw_8Bits GPIO

Notice that two additional blocks (Proc Sys Reset and AXI Interconnect) have been automatically been added to the design.

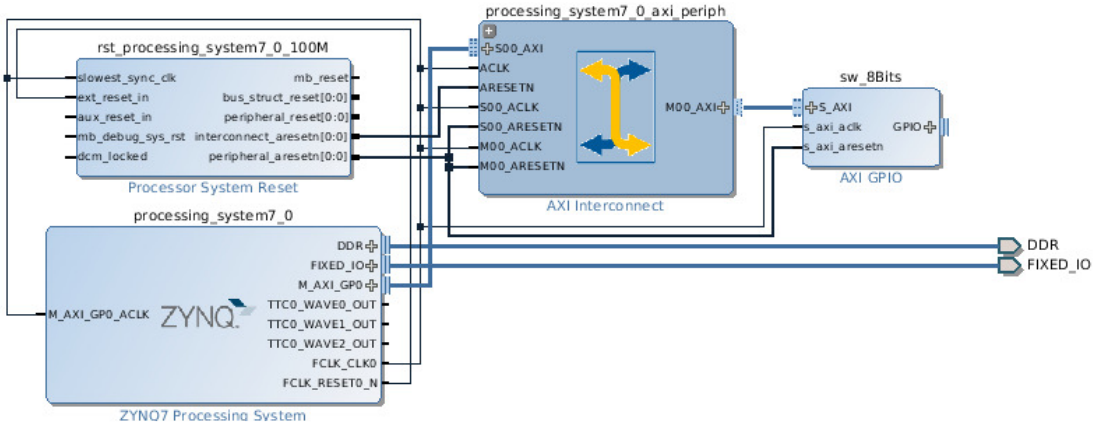


Figure 21. Design with sw_8bits automatically connected

2-6. Similarly, add two more instances of GPIO naming them as btn_4Bits and led_4Bits. Connect them using the Run Connection Automation wizard.

- **btn_5Bits (select the board interface as btns 5Bits)**
- **led_8Bits (select the board interface as leds 8Bits)**

2-6-1. Follow the previous steps to add two more GPIO instances.

2-6-2. Name the added instances as **btn_4Bits** and **led_4Bits**.

2-6-3. Click on the *Run Connection Automation* link and connect the interfaces for each added peripherals.

2-6-4. Configure the added instances to use the following board interfaces:

Peripheral Name	Board Interface
btn_5Bits	btns 5Bits
led_8Bits	leds 8Bits

2-7. Use Run Connection Automation to create a port for the switches, buttons and leds instances.

2-7-1. Click **Run Connection Automation** and select **/btn_5Bits/GPIO.**

2-7-2. In the Run Connection Automation dialog box, in the Select Board Interface drop-down list, select **btns_5bits.**

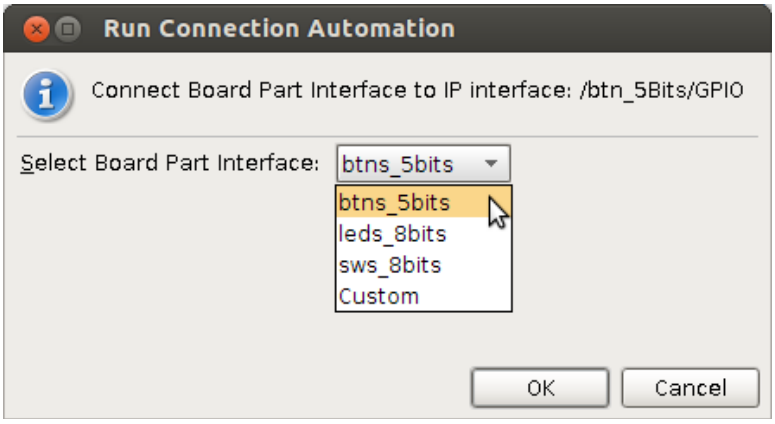


Figure 22. Selecting the Board Interface

2-7-3. Click **OK**.

2-7-4. Similarly, use Run Connection Automation to create an external port for the **sw_8Bits** and **led_8Bits** instances.

Your design should look similar to the figure below. You can click the **Regenerate** icon ( ) to redraw diagram.

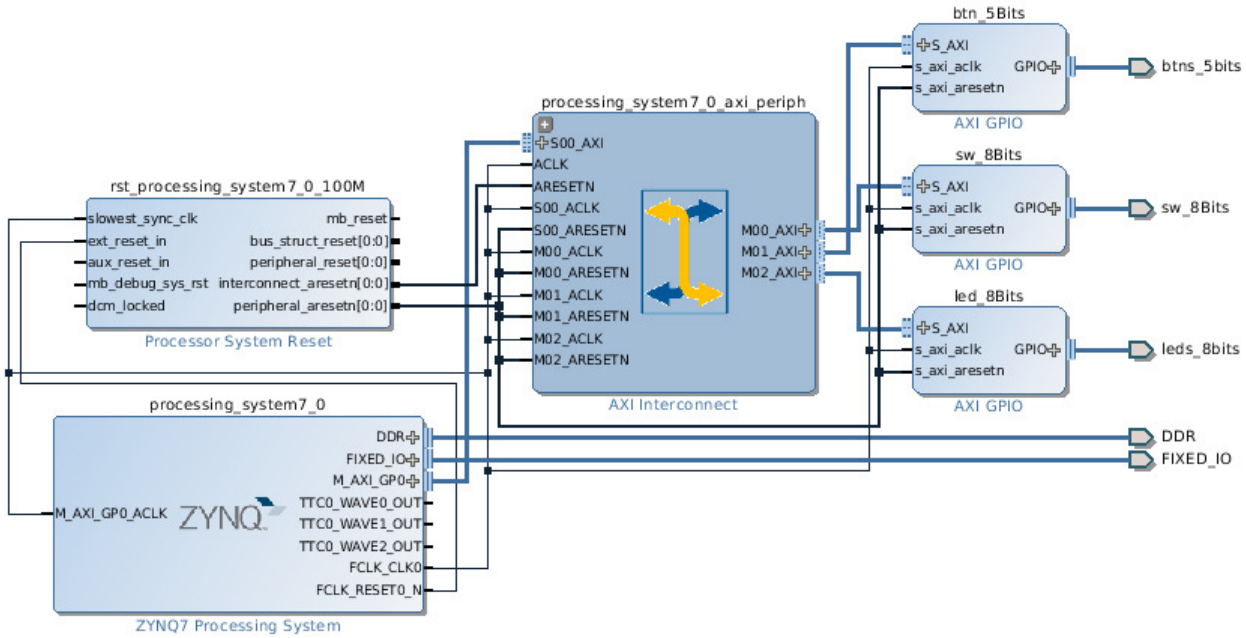


Figure 23. All peripherals added and signals connected

2-8. Assign addresses to the peripherals added to the design.

2-8-1. Select the **Address Editor** tab.

2-8-2. From the vertical toolbar, click **Expand All** ( ).

You may find the unmapped peripherals.

2-8-3. From the vertical toolbar, click **Auto Assign Address** ( ).

The addresses are automatically assigned to the peripherals.

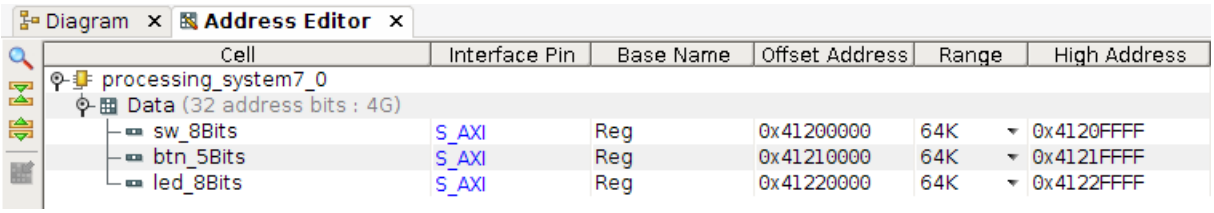


Figure 24. Addresses assigned to the peripherals

2-8-4. Select **File** > **Save Block Design** to save the design.

2-9. Validate the design to catch any connection and address map errors.

2-9-1. Select **Tools** > **Validate Design**.

Any issues are displayed in the Console window.

2-10. Generate the output products.

2-10-1. Select the **Sources** > **Hierarchy** tab.

2-10-2. Under Design Sources, right-click **system (system.bd)** and select **Generate Output Products…**

2-10-3. Click **Generate**.

Building the Hardware Bitstream Step 3

The embedded portion of the design is now specified. The next step in the design process is running the Xilinx implementation tools to generate a bitstream then exporting the hardware design including the bitstream

3-1. The embedded processing system is a component in the overall design. It is now necessary to create a top-level wrapper. Many designs will require other components to be added to the project by instantiating them within the top-level wrapper, while others are sufficient with only the embedded system in the top-level wrapper. For this design, only the embedded system is required. The template that you create will be in the language specified in the Project Settings.

3-1-1. From the Flow Navigator, click **Project Settings** under Project Manager.

3-1-2. Verify that **Verilog** is the target language in the **General** settings.

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

3-2. Create the HDL wrapper for the embedded system.

3-2-1. In the *Sources* window, select the **Hierarchy** tab.

3-2-2. Under *Design Sources*, right-click **system (system.bd)** and select **Create HDL Wrapper**.

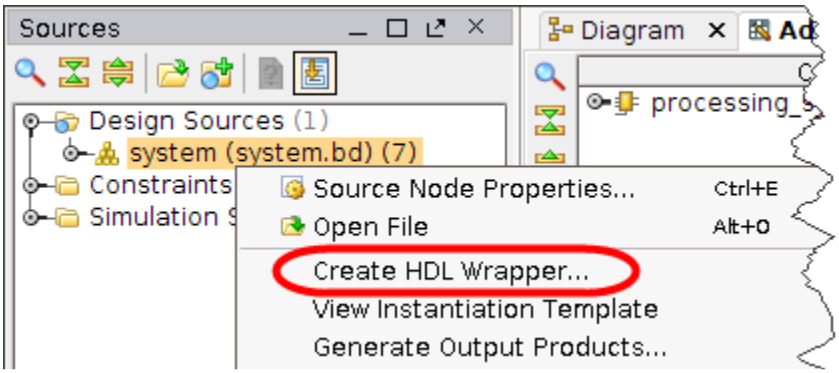


Figure 25. Generating a top-level wrapper for the embedded system

3-2-3. Use the default **Let Vivado manage wrapper and auto-update** option and click **OK**.

This feature builds an editable wrapper for the embedded system in the language you select (in this case, Verilog by default).

Optionally, you can double-click **system_wrapper** to see what was created.

3-3. Now you will run the synthesis.

3-3-1. In the Flow Navigator, under Synthesis, click **Run Synthesis**.

Once the synthesis completes, the Synthesis Completed dialog box opens.

3-3-2. In the Synthesis Completed dialog box, select **Open Synthesized Design**.

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

3-4. Verify the I/O pin constraints of the peripheral instance.

3-4-1. Select **I/O Planning** from the Layout drop-down menu.

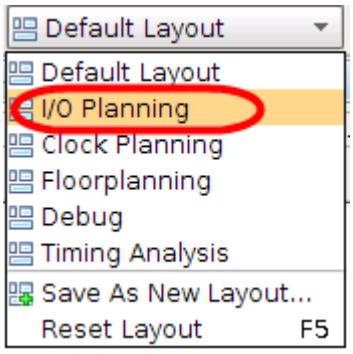


Figure 26. Selecting the IO Planning layout

3-4-2. In the I/O ports tab, expand **GPIO_59088 > btns_5Bits_tri_i, GPIO_34779 > leds_8Bits_tri_o, and GPIO_22244 > sws_8Bits_tri_i**.

Notice that pins have already been assigned to this peripheral. The pin information was included in the board support package and was automatically assigned when the IP was automatically connected to the port.

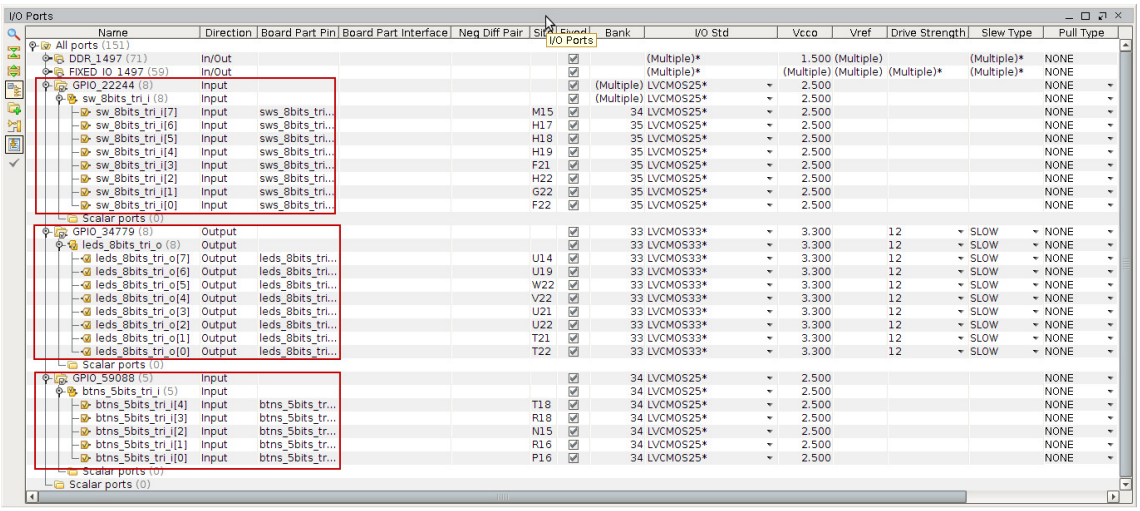


Figure 27. IO Ports assigned according to the board interface details

3-5. Generate the bitstream.

3-5-1. In the Flow Navigator, click **Generate Bitstream**.

3-5-2. Click **Yes** if the No implementation Results Available dialog box opens.

The Bitstream Generation Completed dialog opens.

3-5-3. Select **Open Implemented Design**.

3-5-4. Click **Yes** to close the Synthesized Design before opening Implemented Design dialog box.

3-5-5. Click **OK** to continue.

Exporting Bitstream and Hardware Step 4

After you have configure your hardware project build bitstream if it is necessary. PetaLinux project requires hardware description file. You can pass the hardware description file by running "Export Hardware" from Vivado.

Petalinux tools will generate device tree source file, u-boot config header files, and enable some Xilinx IP kernel drivers based on the hardware description file.

4-1. While still in the Vivado IDE, with the block design and implemented design open, export the embedded system's hardware.

4-1-1. Select **File** > **Export** > **Export Hardware**.

4-1-2. Select **Export Hardware**.

4-1-3. Make sure that the **Include bitstream** box is checked.

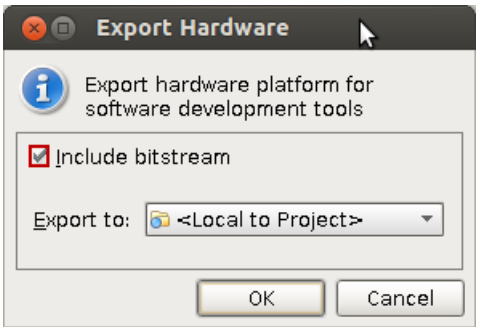


Figure 28. Exporting the processor hardware

4-1-4. Click **OK**.

4-1-5. In the Vivado Design Suite, select **File** > **Exit**.

4-1-6. Click **OK** to exit.

Creating a New Platform with PetaLinux Configuration Options Step 5

The next step is to create a new PetaLinux software platform, ready for building a Linux system customized to your new hardware platform.

5-1. Verify the PetaLinux working environment or set it by running the PetaLinux setup script.

5-1-1. Run the following command on the host machine to see if the PetaLinux tools are installed and sourced properly

```
[host] $ echo $PETALINUX
```

It should display

```
/opt/pkg/petalinux-v2014.2-final
```

5-1-2. If it doesn't then run the following command on the host machine to source the set up script:

```
[host] $ source /opt/pkg/petalinux-v2014.2-final/settings.sh
```

5-2. Use the `petalinux-create` command to create a new embedded Linux platform in the project directory.

5-2-1. Change to the project directory. Enter the following command:

```
cd ~/emblnx/labs/lab6
```

5-2-2. Run the following command to create a new Petalinux project:

```
[host] $ petalinux-create -t project --name software
```

```
petalinux@ubuntu:~/emblnx/labs/lab6$ petalinux-create -t project --name software
INFO: Create project: software
INFO: New project successfully created in /home/petalinux/emblnx/labs/lab6/softw
are
```

Figure 29. Creating a New Petalinux Project

Configuring and Building the Linux System Step 6

The final step is to customize the software platform template to precisely match your unique hardware system. This is done by copying and merging the platform configuration files generated during the hardware build phase into the newly created software platform.

6-1. Configure the kernel to enable the AXI GPIO driver.

6-1-1. Change the directory to the `<XSDK workspace directory>` directory:

```
[host] $ cd ~/emblnx/labs/lab6/hardware/lab6.sdk
```

6-1-2. Use the `petalinux-config` command to import the hardware configuration:

```
[host] $ petalinux-config --get-hw-description -p
~/emblnx/labs/lab6/software
```

It launches the top system configuration menu when petalinux-config --get-hw-description runs for the first time for the PetaLinux project or the tool detects there is a change in the system primary hardware candidates.

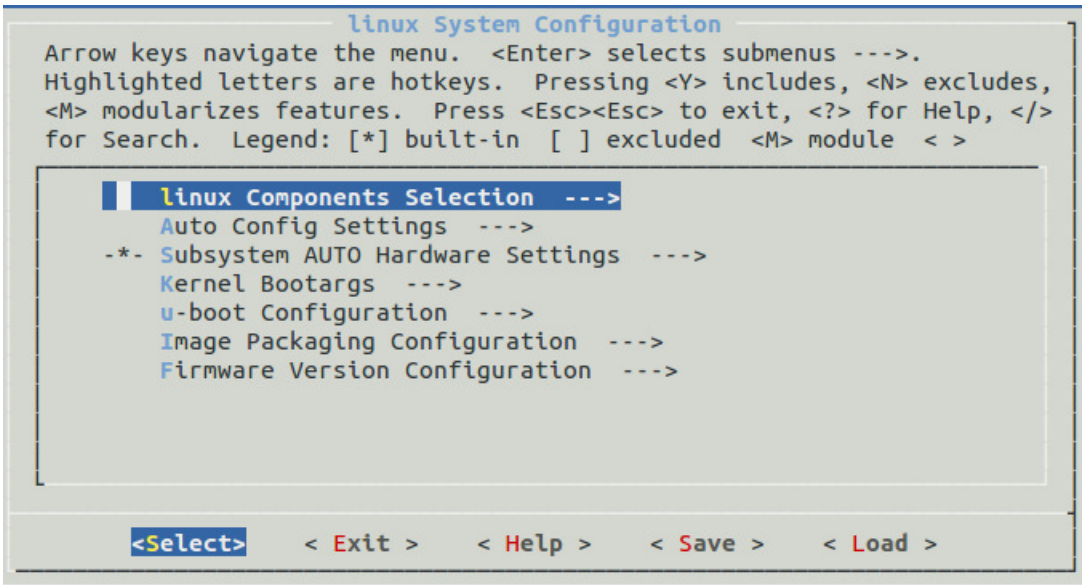


Figure 30. Configuring the software platform

6-1-3. Select the **Subsystem AUTO Hardware Settings** (using down arrow key and pressing *Enter*) to go into the sub-menu.

6-1-4. Move down to *Advanced bootable images storage Settings* option and press **Y** or space bar to enable the setting.

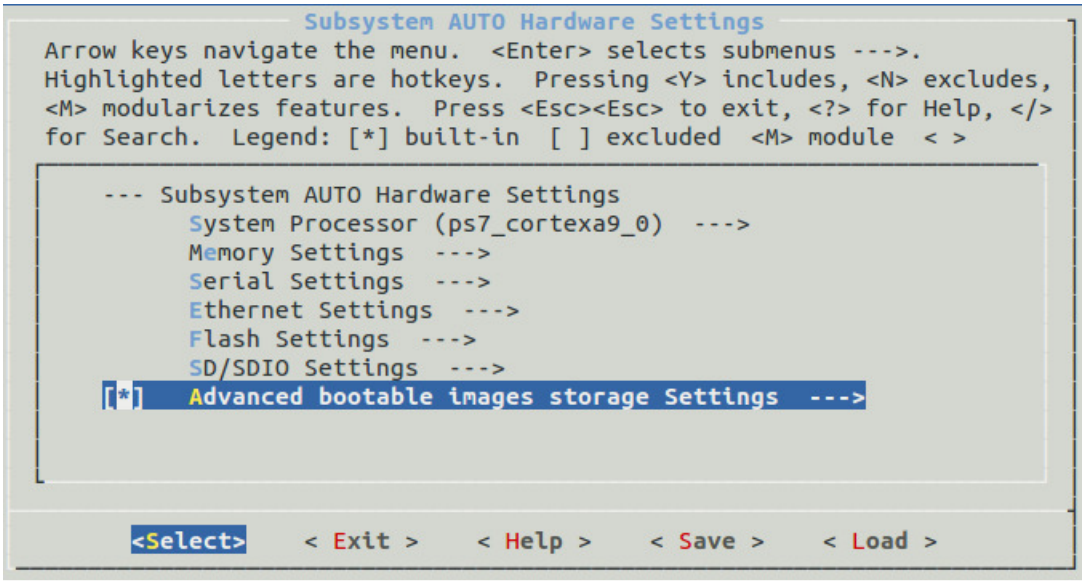


Figure 31. Enabling the Advanced bootable images storage settings

This sub-menu allows users to specify where bootable images are. The settings of this menu are used by PetaLinux auto configured u-boot.

6-1-5. Press **Enter** to go into the **Advanced bootable images storage Settings** sub-menu.

6-1-6. Go into the **boot image storage > image storage media (primary flash)** and select the *primary sd* option.

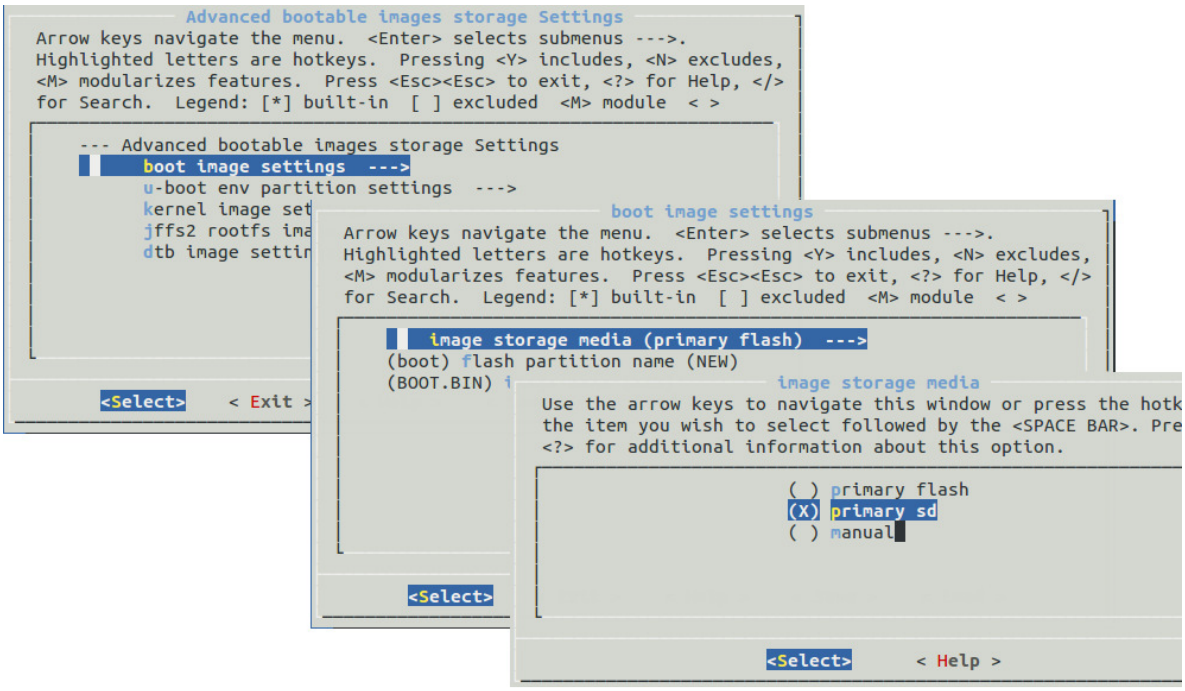


Figure 32. Selecting boot device

6-1-7. Press **Enter.**

6-1-8. Select **<Exit>** to go one level up.

6-1-9. Similarly, select the **kernel image settings** and select the *image storage media* as **primary sd**.

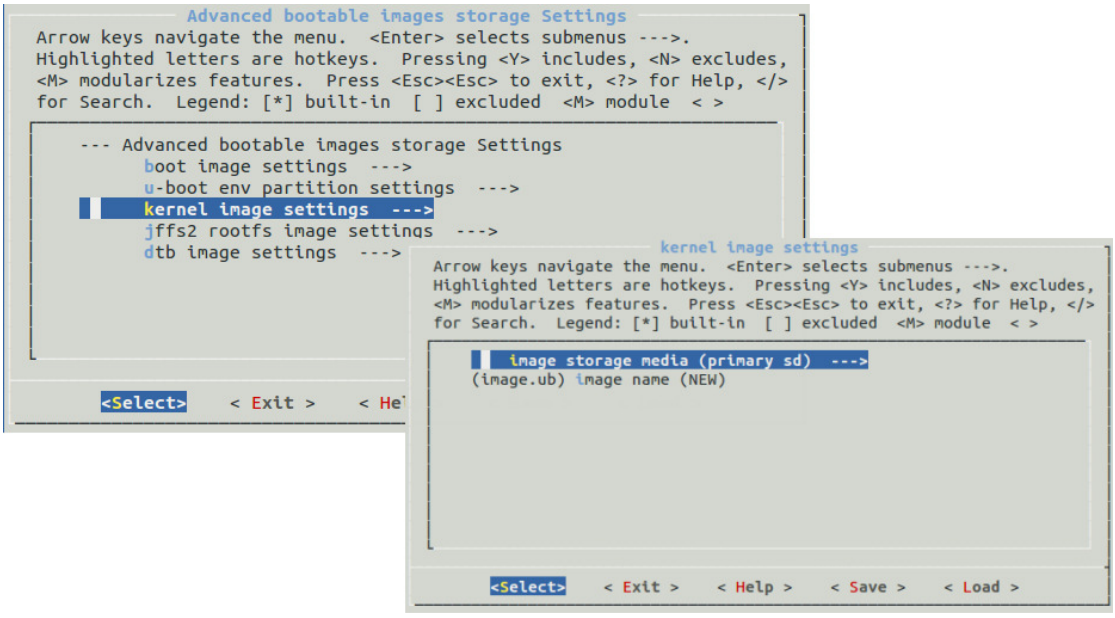


Figure 33. Selecting the kernel image storage media as SD

6-1-10. Press **Enter,** and select **exit** to go one level up to **Subsystem AUTO Hardware Settings.**

6-2. Modify the Ethernet settings to use the static IP address.

6-2-1. Use the down arrow key to select the Ethernet Settings sub-menu.

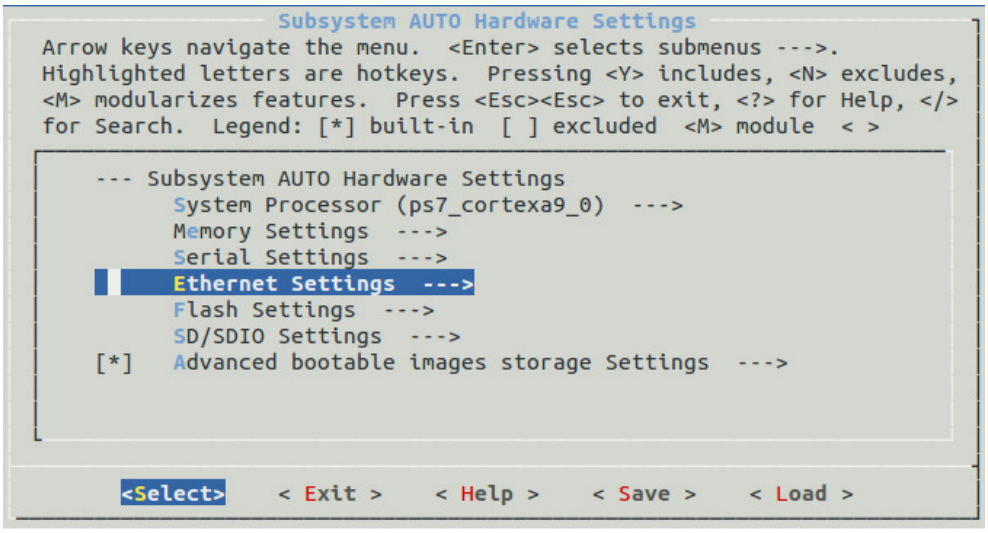


Figure 34. Selecting Ethernet Settings sub-menu

6-2-2. Use the down arrow key to go to the Obtain IP address automatically, and press space bar to unselect it.

Notice that the *Static IP address*, *Static IP netmask*, and *static IP gateway* menus show up.

6-2-3. Change the *Static IP address* to **192.168.1.2** as we will use that address for the target board.

6-2-4. Leave the *Static IP netmask* to **255.255.255.0** as we use the same netmask for the host machine.

6-2-5. Change the *Static IP gateway* to **192.168.1.1** as we will use that address for the target board gateway.

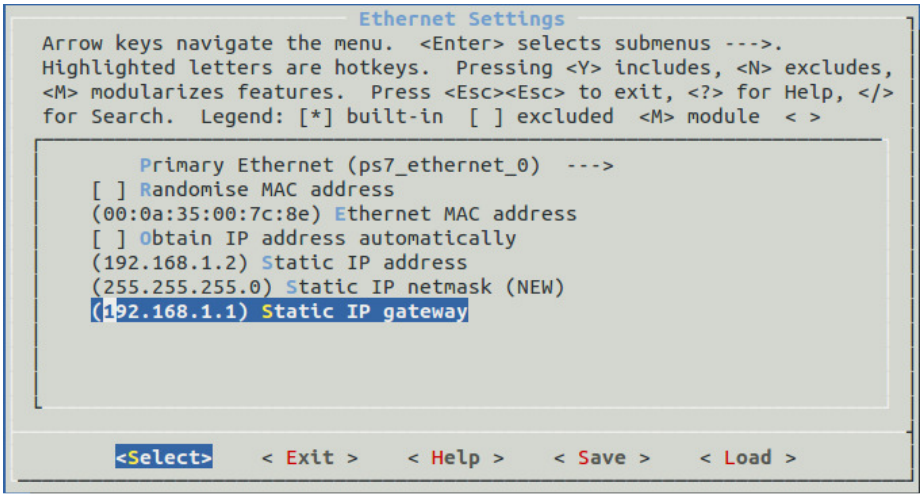


Figure 35. Setting static IP address

6-2-6. Exit the configuration and save the configuration.

This step may take a few minutes to complete. This is due to the tool will parse the hardware description file to get the hardware information to update the device tree, PetaLinux u-boot configuration files and the kernel config files based on the "Auto Config Settings" and "Subsystem AUTO Hardware Settings".

For example, If you select ps7_ethernet_0 as the Primary Ethernet, the tool will automatically enable its kernel driver if user selects to auto update kernel config, it will also update the u-boot configuration headers for u-boot to use the selected ethernet controller if user selects to auto update u-boot config.

The PetaLinux tools generate the hardware configuration files, if required, and copies the configuration files to the correct location in your project directory.

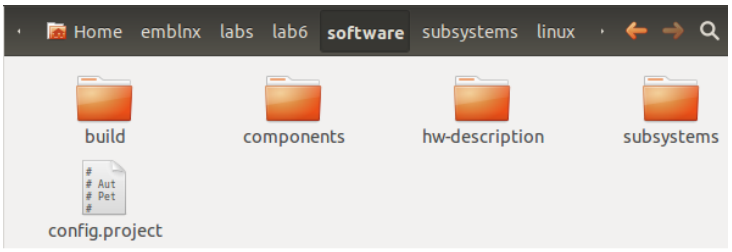


Figure 36. Directories created under the project folder software

6-2-7. Change into the root directory of your PetaLinux project:

```
[host] $ cd ~/emblnx/labs/lab6/software
```

6-3. Launch `petalinux-config` and change the host name and product name.

6-3-1. Launch the top-level system settings configuration menu:

```
[host] $ petalinux-config
```

6-3-2. Select the **Firmware Version Configuration** option and hit *Enter*.

6-3-3. Select **(software) Host name** and hit *Enter*.

6-3-4. Clear the default name by using the Backspace key and enter the new host name as **petalinux**.

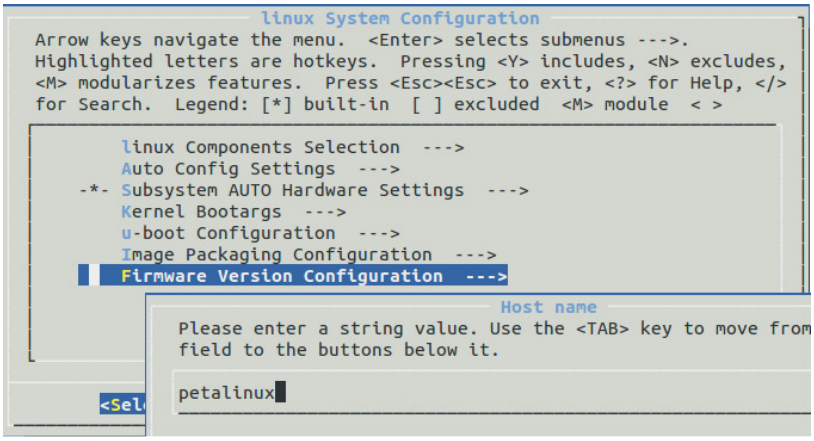


Figure 37. Changing the Host Name

6-3-5. Click **OK**.

6-3-6. Similarly, change the product name to **lab6**.

6-3-7. Click **Exit** twice and select **Yes** to save the new configuration.

6-4. Launch the Linux kernel configuration menu and configure it to meet your requirements.

6-4-1. Run the following command in the terminal:

```
[host] $ petalinux-config -c kernel
```

6-4-2. Review the menus.

For example, review the **Device Drivers > GPIO Support** menu.
No changes are required at this time.

6-4-3. Select **Exit**.

6-5. Launch the rootfs configuration menu and configure it to meet your requirements:

6-5-1. Run the following command in the terminal:

```
[host] $ petalinux-config -c rootfs
```

6-5-2. Select **Apps**.

6-5-3. Enable **gpio-demo** by clicking <**Y**>.

6-5-4. Select **Exit** and save the configuration.

6-6. Build the system image.

6-6-1. Copy the provided **system-top.dts** file from the `~/emblnx/sources/lab6` and place it in the `~/emblnx/labs/lab6/software/subsystems/linux/configs/device-tree` directory replacing the existing file. This is done to define the board specific Ethernet hardware.

6-6-2. Run the `petalinux-build` command again to build the system image:

```
[host] $ petalinux-build
```

Booting the PetaLinux Image with SD Card Step 7

Having configured to use the SD card as the primary boot device, you will create a boot image file that contains the Zynq All Programmable SoC FSBL, the BIT file for the programmable logic (PL) configuration, u-boot, and the Linux image for the SD card boot.

7-1. Create a BOOT.BIN file to boot from the SD card.

7-1-1. Change the directory to where linux image is generated by executing the following command.

```
[host] cd ~/emblnx/labs/lab6/software/images/linux
```

7-1-2. Enter the following command in the console:

```
petalinux-package --boot --fsbl zynq_fsbl.elf --fpga
~/emblnx/labs/lab6/hardware/lab6.runs/impl_1/system_wrapper.bit --uboot
```

The `BOOT.BIN` file will be generated and the system_wrapper.bit file will be copied in the linux image directory; i.e., `~/emblnx/labs/lab6/software/images/linux`.

7-1-3. Copy the BOOT.BIN and image.ub files from the `~/emblnx/labs/lab6/software/images/linux` directory to the SD card.

7-2. Set the jumper settings for booting from SD card. Connect the board and power it ON and set the serial port terminal.

7-2-1. Make sure that the jumper settings for booting from SD card are set as shown below.

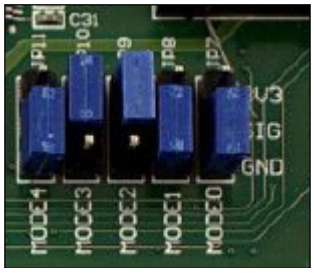


Figure 38. Jumper setting for SD card boot

7-2-2. Power ON the board.

7-2-3. Make sure that `/dev/ttyACM0` is set to read/write access:

```
[host] $ sudo chmod 666 /dev/ttyACM0
```

7-2-4. In the dashboard, in the Search field, enter the serial port.

7-2-5. Select the **Serial port terminal** application and select the appropriate port and 115200 baud.

7-3. Explore the built Linux system.

7-3-1. In the terminal, press the <**Enter**> key.

7-3-2. Use **root** as the login and password.

7-4. Verify the LEDs by using the `devmem` command.

7-4-1. In the console, enter the following command:

```
~#devmem 0x41220000 32 0xff
```

All the LEDS on the board will be ON.

7-5. Now use the `gpio-demo` application.

7-5-1. In the console, enter the following command:

```
~#gpio-demo -g 243 -o 0
```

All the LEDS on the board will be OFF.

Booting the PetaLinux Image using JTAG Step 8

You can also program the hardware and boot the Linux image using JTAG mode.

8-1. Set the jumper settings for the JTAG mode booting. Connect the board with two micro-usb cables and power it ON and set the serial port terminal.

8-1-1. Make sure that the jumper settings for the JTAG mode booting are set as shown below.

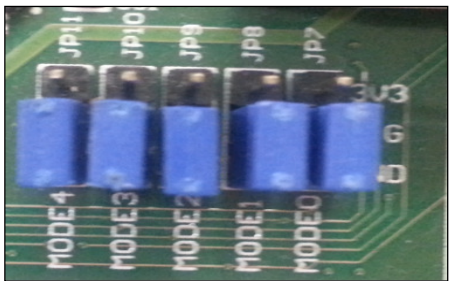


Figure 39. Jumper setting for the JTAG boot

8-1-2. Connect one micro-usb cable to the JTAG port and another to the UART port.

8-1-3. Power ON the board.

8-1-4. Make sure that `/dev/ttyACM0` is set to read/write access:

8-1-5. In the dashboard, in the Search field, enter the serial port.

8-1-6. Select the **Serial port terminal** application and select the appropriate port and 115200 baud.

8-2. Program the FPGA with the bitstream and download the Linux kernel.

8-2-1. Change the directory to where linux image is generated by executing the following command.

```
[host] cd ~/emblnx/labs/lab6/software/images/linux
```

8-2-2. Update the prebuilt image for the built design and images using the following command.

```
[host] petalinux-package --prebuilt --force --fpga system_wrapper.bit
```

This will create or update the **pre-built** directory under `~/emblnx/labs/lab6/software`

8-2-3. Go to `~/emblnx/labs/lab6/software/pre-built/linux/implementation` directory and rename the system_wrapper.bit file to **download.bit** file.

8-2-4. Download the image (including bitstream) using the following command.

```
[host] petalinux-boot --jtag --prebuilt 3
```

Wait for the DONE LED to lit and then wait for few minutes to download the image and execute.

8-3. Explore the built Linux system.

8-3-1. In the terminal, press the <**Enter**> key.

8-3-2. Use **root** as the login and password.

8-4. Verify the LEDs by using the `devmem` command.

8-4-1. In the console, enter the following command:

```
~#devmem 0x41220000 32 0xff
```

All the LEDS on the board will be ON.

8-5. Now use the `gpio-demo` application.

8-5-1. In the console, enter the following command:

```
~#gpio-demo -g 243 -o 0
```

All the LEDS on the board will be OFF.

Booting the PetaLinux Image from QSPI Step 9

You will need to configure the software plateform to use the QSPI flash as the primary boot device. You will create a boot image file that contains the Zynq All Programmable SoC FSBL, the BIT file for programmable logic (PL) configuration, u-boot, and the Linux image for the QSPI boot.

9-1. Use the `petalinux-config` command from the software project directory to select the primary boot device as QSPI.

9-1-1. Change to the project directory. Enter the following command:

```
cd ~/emblnx/labs/lab6/software
```

9-1-2. Use the `petalinux-config` command to open the software configuration:

```
[host] $ petalinux-config
```

9-1-3. Select the **Subsystem AUTO Hardware Settings** (using down arrow key and pressing *Enter*) to go into the sub-menu.

9-1-4. Move down to *Advanced bootable images storage Settings* option and press **Enter** to go into the **Advanced bootable images storage Settings** sub-menu.

9-1-5. Go into the **boot image storage > image storage media (primary flash)** and select the *primary flash* option.

9-1-6. Click **Enter**.

9-2. Change the image name to BOOT.mcs and kernel image media to QSPI.

9-2-1. Use the down arrow key to select the **image name**, and change the image to **BOOT.mcs**

9-2-2. Click **OK**.

9-2-3. Select **<Exit>** to go one level up.

9-2-4. Similarly, select the **kernel image settings** and select the *image storage media* as **primary flash**.

9-2-5. Use Exit twice to come back to **Subsystem AUTO Hardware Settings** sub-menu.

9-2-6. Use down-arrow key to select **Flash Settings** and hit Enter.

Notice the settings. The partition 0 size is 0x500000 whereas partition 1 size is 0x20000. So the partition 2 where the kernel image will reside is at 0x520000.

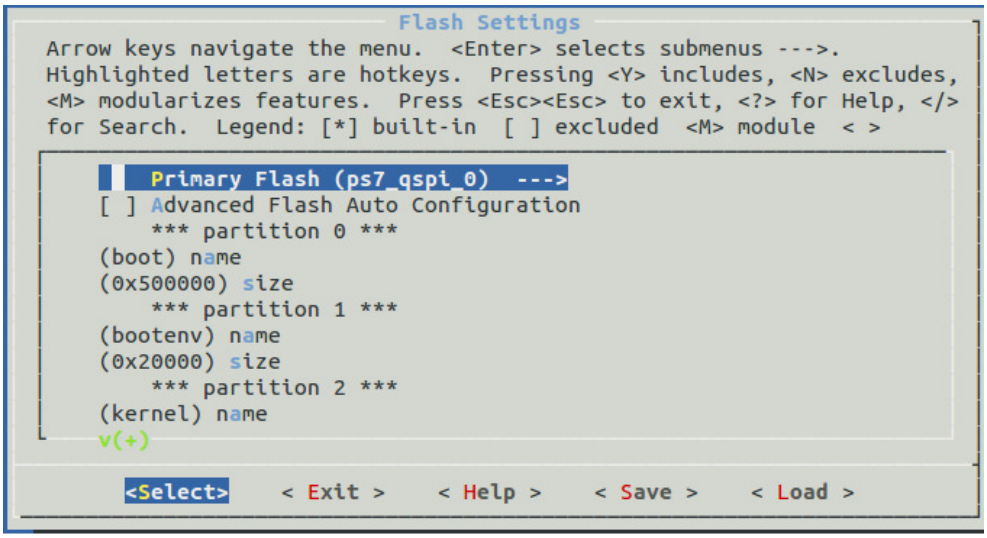


Figure 40. Flash settings

9-2-7. Exit the configuration and save the configuration.

The PetaLinux tools will update the image files in the `~/emblnx/labs/lab6/software/images/linux` directory.

9-2-8. Change into the root directory of your PetaLinux project:

```
[host] $ cd ~/emblnx/labs/lab6/software
```

9-3. Build the system image.

9-3-1. Run the `petalinux-build` command to build the system image:

```
[host] $ petalinux-build
```

9-4. Create a flash boot image using the provided `output.bif` file, and the `bootgen` and `zynq_flash` commands.

9-4-1. Copy the provided **output.bif** file from the `~/emblnx/sources/lab6` and place it in the `~/emblnx/labs/lab6/` directory.

This file lists where various files (zynq_fsbl.elf, system_wrapper.bit, u-boot.elf, and image.ub) are located. Make necessary changes if the directory paths to these files are different.

9-4-2. Change to the `lab6` home directory.

9-4-3. Make sure that the board is in the JTAG boot mode, connected, and is powered ON.

9-4-4. Execute the following bootgen command to generate the BOOT.mcs file.

```
[host] $ bootgen -image output.bif -o i
/home/petalinux/emblnx/labs/lab6/BOOT.mcs
```

9-4-5. Execute the following command to program the device.

```
[host] $ zynq_flash -f /home/petalinux/emblnx/labs/lab6/BOOT.mcs -
offset 0 -flash_type qspi_single -cable type xilinx_tcf
```

This process will take about four minutes to complete.

9-4-6. Now that you have programmed the flash, power OFF the board because you need to change the jumper settings for the flash boot.

9-5. Verify the jumper settings for booting from flash.

9-5-1. Make sure that the jumper settings for booting from flash are set as shown below.

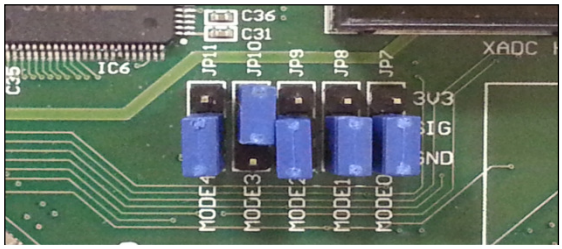


Figure 41. Jumper Setting for QSPI Boot

9-6. Power on the board and set the serial port terminal.

9-6-1. Power ON the board.

9-6-2. Make sure that `/dev/ttyACM0` is set to read/write access:

9-6-3. In the dashboard, in the Search field, enter the serial port.

9-6-4. Select the **Serial port terminal** application.

9-7. Explore the built Linux system.

9-7-1. In the terminal, press the <**Enter**> key.

9-7-2. Use **root** as the login and password.

9-8. Verify the LEDs by using the `devmem` command.

9-8-1. In the console, enter the following command:

```
~#devmem 0x41220000 32 0xff
```

All the LEDS on the board will be ON.

9-9. Now use the `gpio-demo` application.

9-9-1. In the console, enter the following command:

```
~#gpio-demo -g 243 -o 0
```

All the LEDS on the board will be OFF.

Conclusion

In this lab, you have learned how to:

- Use the Vivado Design Suite to create a Zynq AP SoC project
- Build a bitstream
- Create a new embedded Linux platform with the PetaLinux tools support
- Build an embedded Linux system for a hardware platform
- Boot the system using SD card, JTAG, and QSPI flash modes

Completed Solution

(i) SDCard Boot: If you want to run the solution then copy **`BOOT.bin`** and **`image.ub`** from the *`labsolution\lab6\SDCard`* directory onto a SD card. Place the SD card in the ZedBoard. Set ZedBoard in the SD Card boot mode. Connect the ZedBoard to the host machine using Ethernet cable.

Run the following command to start DHCP server on the host:

```
[host] $ sudo service isc-dhcp-server restart
```

Power ON the board. Set the terminal session.

Login into the system and test the lab.

(ii) FlashBoot: If you want to run the solution using FLASH as the boot device, then place the board in the JTAG boot mode and power ON the board. Execute the following command to program the flash, adjusting the path to BOOT.mcs file:

```
[host] $ zynq_flash -f /home/petalinux/emblnx/labs/lab6/BOOT.mcs -
offset 0 -flash_type qspi_single -cable type xilinx_tcf
```

Once the program is completed, turn OFF the board, set the board to boot in the QSPI mode, and power ON the board. Set the terminal session. Login into the system and test the lab.

(iii) JTAG Boot: If you want to run the solution using JTAG as the boot device, then place the board in the JTAG boot mode and power ON the board. Set the terminal session.

Download the image (including bitstream) using the following command from the directory from the `jtag-boot/software` directory.

```
[host] petalinux-boot --jtag --prebuilt 3
```

Wait for the DONE LED to lit and then wait for few minutes to download the image and execute.

Login into the system and test the lab