PetaLinux Tools Documentation

Workflow Tutorial

UG1156 (v2017.3) October 4, 2017

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
10/04/2017	2017.3	Updated for PetaLinux Tools 2017.3 release

Table of Contents

Revision History . . . 2

Chapter 1: PetaLinux Workflow Tutorial

1. Introduction . . . 4
2. PetaLinux Reference BSP’s . . . 6
3. Test Drive a PetaLinux BSP Image . . . 9
4. Configuring Custom Hardware for Embedded Linux . . . 15
5. Working with a PetaLinux Project . . . 18
6. Software Testing with QEMU . . . 21
7. Building a Bootable System Image . . . 23

Appendix A: Internal Architecture of PetaLinux Projects

Working with the PetaLinux Menuconfig System . . . 24
Yocto Settings . . . 29
PetaLinux Project Structure . . . 30
Anatomy of a PetaLinux Project . . . 30
PetaLinux Project Directory Structure . . . 31

Appendix B: Additional Resources and Legal Notices

Xilinx Resources . . . 35
Solution Centers . . . 35
References . . . 35
Please Read: Important Legal Notices . . . 36

PetaLinux Workflow Tutorial

1. Introduction

The PetaLinux Workflow tutorial demonstrates how to successfully use the PetaLinux design workflow through a series of real world examples. This tutorial aligns with PetaLinux v2017.3 and some syntax or command-line options may not apply or work in the same manner for different versions of PetaLinux. This tutorial assumes that you have already installed and licensed both Vivado® and PetaLinux. For more details on installation and licensing, see the *PetaLinux Tools Documentation: Reference Guide* (UG1144) [Ref 1]

In general, the methodologies and steps presented here are universal to all PetaLinux designs. The specific examples presented here utilize the Zynq family device (including Zynq® and Zynq UltraScale+™ MPSoC). Where appropriate, differences or variations specific to MicroBlaze™-based FPGA designs are denoted in context.

For additional details on the specific PetaLinux tools or command-line options, see the *PetaLinux Command Line Reference* (UG1157) [Ref 4]. For more information on specific PetaLinux workflows, see the *PetaLinux Tools Documentation: Reference Guide* (UG1144) [Ref 1]

Design Flow Overview

In general, the PetaLinux tools follow a sequential workflow model. The table below provides an example design workflow, demonstrating the order in which the tasks should be completed and the corresponding tool or workflow for that task.

Table 1-1: **Design Flow Overview**

Design Flow Step	Tool / Workflow
Hardware Platform Creation	Vivado
Create PetaLinux Project	petalinux-create -t project
Initialize PetaLinux Project	petalinux-config --get-hw-description
Configure System-Level Options	petalinux-config
Create User Components	petalinux-create -t COMPONENT
Configure the Linux Kernel	petalinux-config -c kernel
Configure the Root Filesystem	petalinux-config -c rootfs
Build the System	petalinux-build
Deploy the System	petalinux-package
Test the System	petalinux-boot

2. PetaLinux Reference BSP's

PetaLinux Reference Board Support Package (BSP) files are reference designs that you may use to get up and going quickly. In addition, these designs can be used as a basis for creating your own projects. PetaLinux BSP's are provided in the form of installable BSP files and include all necessary design and configuration files required to get started. Pre-built hardware and software images included in the BSP package are ready for download to your board or for booting in the QEMU system simulation environment.

BSP reference designs are not included in the PetaLinux tools installer and need to be downloaded and installed separately. PetaLinux BSP packages are available on the Xilinx.com Download Center.

Follow the below steps to install a BSP:

1. Change to the directory under which you want PetaLinux projects to be created. For example, if you want to create projects under /home/user:

   ```
   $ cd/home/user
   ```

2. Run `petalinux-create` command on the command console:

   ```
   petalinux-create -t project -s <path-to-bsp>
   ```

   Based on the BSP you are installing, you will see the output, similar to the output below:

   ```
   INFO: Create project:
   INFO: Projects:
   INFO:    * xilinx-zcu102-zu9-es2-rev1.0-2017.3
   INFO: has been successfully installed to /home/user
   INFO: New project successfully created in /home/user
   ```

In the above example, when the command runs, it tells you the projects that are extracted and installed from the BSP. If the specified location is on the Network File System (NFS), it changes the TMPDIR to `/tmp/<projname_timestamp>`. If `/tmp/<projname_timestamp>` is also on NFS, then it throws an error. You can change TMPDIR anytime through petalinux-config -->Yocto-settings. Do not configure the same location as TMPDIR for two different PetaLinux projects, this can cause build errors. If you run `ls` from "/home/user", you will see the installed project(s). For more details on the structure of a PetaLinux project, see PetaLinux Project Structure in Appendix A

Rebuilding the Reference Design System

So far, you have installed a PetaLinux reference BSP and explored its contents. This section describes the steps to re-build the BSP image, you can test it in QEMU or on hardware.

The steps below outline how to re-build the BSP reference design system:

1. Run `petalinux-build` to compile the software images:

   ```
   $ petalinux-build
   ```

   This step generates a device tree DTB file, a first stage bootloader (if selected), U-Boot, the Linux kernel, and a root filesystem image. Finally, it generates the necessary boot images.

2. The compilation progress shows on the console. Wait until the compilation finishes.

TIP: *A detailed compilation log is in "`<plnx-proj-root>/build/build.log`" file.*

When the build finishes, the generated images will be within the `<plnx-proj-root>/images` and `/tftpboot` directories.

TIP: *The build process may report errors writing to the `/tftpboot` directory, if this directory does not exist or you cannot write to it. These error messages are informational only and do not affect the output images. You may mitigate these messages by disabling the "`Copy final images to tftpboot`" feature in the `Image Packing Configuration` menu of the system-level configuration.*

An example of the compilation progress output is shown below:

```
[INFO] building project
[INFO] generating Kconfig for project
[INFO] oldconfig project
[INFO] sourcing bitbake
[INFO] generating plnxtool conf
[INFO] generating meta-plnx-generated layer
[INFO] generating machine configuration
[INFO] generating bbappends for project . This may take time !
[INFO] generating u-boot configuration files
[INFO] generating kernel configuration files
[INFO] generating kconfig for Rootfs
Generate rootfs kconfig
[INFO] oldconfig rootfs
[INFO] generating petalinux-user-image.bb
INFO: bitbake petalinux-user-image
Loading cache: 100%
|########################################################################| Time:
0:00:00
Loaded 3252 entries from dependency cache.
Parsing recipes: 100%
|######################################################################| Time:
0:00:01
```

```
Parsing of 2461 .bb files complete (2422 cached, 39 parsed). 3253 targets, 224
skipped, 0 masked, 0 errors.
NOTE: Resolving any missing task queue dependencies
Initialising tasks: 100%
|##################################################################| Time: 0:00:11
Checking sstate mirror object availability: 100%
|##########################################| Time: 0:00:13
NOTE: Executing SetScene Tasks
NOTE: Executing RunQueue Tasks
pmu-firmware-2017.3+gitAUTOINC+b3f9b55770-r0 do_compile: NOTE: pmu-firmware:
compiling from external source tree /opt/pkg/petalinux/tools/hsm/data/embeddedsw
fsbl-2017.3+gitAUTOINC+b3f9b55770-r0 do_compile: NOTE: fsbl: compiling from external
source tree /opt/pkg/petalinux/tools/hsm/data/embeddedsw
NOTE: Tasks Summary: Attempted 2459 tasks of which 1865 didn't need to be rerun and
all succeeded.
INFO: Copying Images from deploy to images
INFO: Creating images/linux directory
INFO: TFTP directory does not exist, creating directory:  /tftpboot
[Errno 13] Permission denied: '/tftpboot'
NOTE: Failed to copy built images to tftp dir:  /tftpboot
[INFO] successfully built project
```

The final image is `<plnx-proj-root>/images/linux/image.ub` which is a FIT image. The kernel image is "`Image`" for Zynq UltraScale+ MPSoC, "`zImage`" for Zynq-7000 or "`image.elf`" for MicroBlaze and is located in the "`<plnx-proj-root>/images/linux`" directory. Optionally, a copy is also placed in the "`/tftpboot`" directory if this option is enabled in the system-level configuration for the PetaLinux project.

3. Test Drive a PetaLinux BSP Image

In 2. PetaLinux Reference BSP's you have successfully installed one or more PetaLinux projects from a PetaLinux reference BSP and rebuilt the system image using `petalinux-build`.

Testing the Pre-Built PetaLinux Images

Now, you can try out one of the prebuilt reference designs shipped with your BSP package. This is achieved with the `petalinux-boot` tool. The `petalinux-boot` workflows boot the reference design on physical hardware (JTAG) or under software simulation (QEMU). Later, we will also boot the system image that you rebuilt in 2. PetaLinux Reference BSP's as well.

Testing the Pre-Built PetaLinux Image on Hardware

PetaLinux BSPs include pre-built FPGA bitstreams for each reference design, allowing you to quickly boot Linux on your hardware.

Boot Pre-built Images from SD Card (Zynq UltraScale+ MPSoC Only)

This section is for Zynq UltraScale+ MPSoC only as they allow to boot from SD card. The steps to boot the pre-built images from SD card are mentioned below:

1. Mount your SD card to your host machine.
2. Copy the following files from `<plnx-proj-root>/pre-built/linux/images/` into the root directory of the first FAT partition of your SD card:
    - `BOOT.BIN`
    - `image.ub`
3. Connect the serial port on the board to your workstation.
4. Open a console on your workstation and then start your preferred serial communication program (for example, PuTTY, Tera Term, kermit, etc.) with the baud rate set to 115200 on that console.
5. Power off the board.
6. Set the boot mode of the board to SD boot.
7. Plug the SD card into the board.
8. Power on the board.

9. Watch the console. Hit any key to stop automatic boot when you see the following message on the console:

```
Hit any key to stop autoboot:
```

10. At this point you can explore the U-Boot environment or simply use the command 'run sdboot' in the U-Boot console to boot Linux from SD card:

```
Hit any key to stop autoboot: 0
U-Boot> run sdboot
```

11. Boot messages which are similar to the following appears in the serial console:

```
[ 10.709243] Freeing unused kernel memory: 5568K (ffffffc000c20000 -
ffffffc001190000)
[ 13.448003] udevd[1666]: starting version 3.2
[ 13.458788] random: udevd: uninitialized urandom read (16 bytes read)
[ 13.556064] udevd[1667]: starting eudev-3.2
[ 14.045406] random: udevd: uninitialized urandom read (16 bytes read)
[ 37.446360] random: dd: uninitialized urandom read (512 bytes read)
[ 40.406936] IPv6: ADDRCONF(NETDEV_UP): eth0: link is not ready
[ 41.460975] macb ff0e0000.ethernet eth0: link up (100/Full)
[ 41.474152] IPv6: ADDRCONF(NETDEV_CHANGE): eth0: link becomes ready
[ 44.787172] random: dropbearkey: uninitialized urandom read (32 bytes read)
PetaLinux 2017.3 xilinx-zcu102-2017_3 /dev/ttyPS0
xilinx-zcu102-2017_3 login: root
Password:
root@xilinx-zcu102-2017_3:~#
root@xilinx-zcu102-2017_3:~#
```

12. At the login prompt, use the login `root` and password `root` to log into the Linux system.

Boot Pre-built Images with JTAG

Follow the below steps to download the prebuilt images onto the board with JTAG.

1. Power off the board.
2. Connect the JTAG port on the board with the JTAG cable to your workstation.
3. Connect the serial port on the board to your workstation.
4. Connect the Ethernet port on the board to the local network via a network switch.
5. For Zynq family devices boards, ensure the mode switches are set to JTAG mode. For more details, see the Xilinx Answer 43745.
6. Power on the board.
7. Open a console on your workstation and then start your preferred serial communication program (for example, kermit and minicom) with the baud rate set to 115200 on that console.
8. Run the `petalinux-boot` command as follows on your workstation:

```
$ petalinux-boot --jtag --prebuilt 3
```

The `--jtag` option tells `petalinux-boot` to boot on hardware via JTAG, and the `--prebuilt 3` boots the prebuilt linux kernel. This command takes some time to finish as it downloads the kernel over JTAG. Wait until you see the shell prompt again on the command console before continuing.

The example of the messages displayed on the workstation console during successful `petalinux-boot`, is shown below:

```
INFO: Launching XSDB for file download and boot.
INFO: This may take a few minutes, depending on the size of your image.
INFO: Configuring the FPGA...
INFO: Downloading bitstream to the target.
INFO: Downloading ELF file to the target.
INFO: Downloading ELF file to the target.
INFO: Downloading ELF file to the target.
INFO: Downloading ELF file to the target.
```

The example of the messages displayed on the serial console during successful `petalinux-boot`, is shown below:

Console Prints:

```
[ 5.546354] clk: Not disabling unused clocks
[ 5.550616] ALSA device list:
[ 5.553528] #0: DisplayPort monitor
[ 5.576326] sd 1:0:0:0: [sda] 312581808 512-byte logical blocks: (160 GB/149 GiB)
[ 5.583894] sd 1:0:0:0: [sda] Write Protect is off
[ 5.588699] sd 1:0:0:0: [sda] Write cache: enabled, read cache: enabled, doesn't
support DPO or FUA
[ 5.630942] sda:
[ 5.633210] sd 1:0:0:0: [sda] Attached SCSI disk
[ 5.637897] Freeing unused kernel memory: 512K (ffffffc000c20000 -
ffffffc000ca0000)
INIT: version 2.88 booting
Starting udev
[ 5.746538] udevd[1772]: starting version 3.2
[ 5.754868] udevd[1773]: starting eudev-3.2
Populating dev cache
Starting internet superserver: inetd.
Running postinst /etc/rpm-postinsts/100-sysvinit-inittab...
Running postinst /etc/rpm-postinsts/libglib-2.0-0...
update-rc.d: /etc/init.d/run-postinsts exists during rc.d purge (continuing)
INIT: Entering runlevel: 5
Configuring network interfaces... [ 6.607236] IPv6: ADDRCONF(NETDEV_UP): eth0:
link is not ready
udhcpc (v1.24.1) started
Sending discover...
[ 7.628323] macb ff0e0000.ethernet eth0: link up (1000/Full)
[ 7.633980] IPv6: ADDRCONF(NETDEV_CHANGE): eth0: link becomes ready
Sending discover...
Sending select for 10.10.70.1...
Lease of 10.10.70.1 obtained, lease time 600
/etc/udhcpc.d/50default: Adding DNS 172.19.128.1
/etc/udhcpc.d/50default: Adding DNS 172.19.129.1
Done.
Starting Dropbear SSH server: Generating key, this may take a while...
```

```
Public key portion is:
ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAABAQCxGtijKDWcJgnDxRCGiUPJJIMapFc0tcsCkMGyjJEDs9LRugWzgaa
8XA+pGy4aTvZqHvGnFTvkMw4gZE/O+BBgO8mMK9dFei2BvENbljm8M4NotG5LXRCFDaw6bXBCtg4ekCKWNU
6lUQU+PPdpmj9X+JgnTHnHnNB3jP6MrymCuS5wfFbyHfKdrwWXwfLmCycZr7DjRumee7T/3SrBU3oRJoLcC
Vj2lf5Z7673+rOT1GM3QFzO2HWCCzyz/3IUcEh9mhKpjzgs4iNEKmxwyi29rl37x7PD7zRsQbaW8uUtheCa
in3M1mjKfPnnygopdVh6IFsAT3FFMK4PYJ1GPL+h root@xilinx-zcu102-zu9-es2-rev1_0-2017.3
Fingerprint: md5 f2:ce:1d:f2:50:e6:e2:55:5a:96:6f:bc:98:8f:82:99
dropbear.
Starting syslogd/klogd: done
Starting domain watchdog daemon: xenwatchdogd startup
PetaLinux 2017.3 xilinx-zcu102-zu9-es2-rev1_0-2017.3 /dev/ttyPS0
xilinx-zcu102-zu9-es2-rev1_0-2017.3 login: root
Password:
root@xilinx-zcu102-zu9-es2-rev1_0-2017:~#
```

By default, network settings for PetaLinux reference designs are configured using Dynamic Host Configuration Protocol (DHCP). The output you see may be slightly different from the above example, depending upon which PetaLinux reference design you test and your specific networking environment.

Use the login name `root` and password `root` on the serial console to log into the Linux system.

Troubleshooting

If your local network does not have a DHCP server, the system will fail to acquire an IP address. If so, see Appendix A, which describes how to manually specify the address using the system-level menuconfig.

If the `petalinux-boot` for JTAG command fails, it is typically from a JTAG connectivity failure. Ensure the board is powered on and your JTAG cable is properly connected. For more detailed troubleshooting, see the *Platform Cable USB II* (DS593) [Ref 5].

TIP: *Virtual machines require the ability to pass through USB connections in order to perform JTAG activity, and may run slower than a native machine. For best results, run JTAG operations on a dedicated Linux machine.*

Test Pre-Built PetaLinux Image with QEMU

PetaLinux provides QEMU support to enable testing of PetaLinux software image in a simulated environment without any hardware.

To test the PetaLinux reference design with QEMU, follow these steps:

1. Change to your project directory and boot the prebuilt linux kernel image:

   ```
   $ petalinux-boot --qemu --prebuilt 3
   ```

   The `--qemu` option tells `petalinux-boot` to boot QEMU, instead of real hardware via JTAG, and the `--prebuilt 3` boots the linux kernel.

- The `--prebuilt 1` performs a Level 1 (FPGA bitstream) boot. This option is not valid for QEMU.
- A Level 2 boot includes U-Boot.
- A Level 3 boot includes a pre-built Linux image.

The example of the messages displayed on the serial console during successful level 3 boot, is shown below:

```
[ 10.709243] Freeing unused kernel memory: 5568K (ffffffc000c20000 -
ffffffc001190000)
[ 13.448003] udevd[1666]: starting version 3.2
[ 13.458788] random: udevd: uninitialized urandom read (16 bytes read)
[ 13.556064] udevd[1667]: starting eudev-3.2
[ 14.045406] random: udevd: uninitialized urandom read (16 bytes read)
[ 37.446360] random: dd: uninitialized urandom read (512 bytes read)
[ 40.406936] IPv6: ADDRCONF(NETDEV_UP): eth0: link is not ready
[ 41.460975] macb ff0e0000.ethernet eth0: link up (100/Full)
[ 41.474152] IPv6: ADDRCONF(NETDEV_CHANGE): eth0: link becomes ready
[ 44.787172] random: dropbearkey: uninitialized urandom read (32 bytes read)
PetaLinux 2017.3 xilinx-zcu102-2017_3 /dev/ttyPS0
xilinx-zcu102-2017_3 login: root
Password:
root@xilinx-zcu102-2017_3:~#
root@xilinx-zcu102-2017_3:~#
```

2. Login to PetaLinux with the default user name `root` and password `root`.

TIP: *To exit QEMU, press Ctrl+A together, release and then press X.*

Testing the Re-Built PetaLinux Images

In the prior sections, you booted the pre-built image on real hardware via JTAG or in the QEMU software. Now you will boot the image that you rebuilt in 2. PetaLinux Reference BSP's using the same methods.

Testing the Re-built Image on Hardware

Let us test the re-built software image on real hardware. First, follow the instructions from the Testing the Pre-Built PetaLinux Image on Hardware to connect the board, serial and JTAG correctly.

1. Use `petalinux-boot` to boot U-Boot

```
$ petalinux-boot --jtag --u-boot --fpga --bitstream <BIT stream>
```

2. Use `petalinux-boot` to boot the kernel

```
$ petalinux-boot --jtag --kernel --fpga --bitstream <BIT stream>
```

This command takes a few moments as it is downloading the bistream and the entire kernel. Wait until you see the shell prompt again on the command console before proceeding.

3. In the serial console, you should see the Linux startup messages scroll by as the Linux kernel starts up.

You can now repeat the previous steps for connecting to the board via the serial console.

For Zynq family devices, see Boot Pre-built Images from SD Card (Zynq UltraScale+ MPSoC Only) to understand the basic steps. Instead of using the pre-built `image.ub` file from that example, use the one you just rebuilt from `<plnx-proj-root>/images/linux/image.ub`.

Test the Rebuilt Image with QEMU

1. Use `petalinux-boot --qemu` command to test the newly built software image:

   ```
   $ petalinux-boot --qemu --kernel
   ```

   The system boot messages are shown on the console where QEMU is running.

2. When you see the login prompt on the QEMU console, login as `root` with password `root`.

TIP: *To exit QEMU, press Ctrl+A together, release and then press X.*

---

4. Configuring Custom Hardware for Embedded Linux

Previously, we explored a PetaLinux BSP reference design. Now, we will build a Linux system with PetaLinux using a Vivado design of our own. When a Vivado hardware platform is defined, there are a small number of hardware and peripheral IP configuration changes required, to ensure that the hardware platform is Linux-ready. These changes are detailed below.

Configuring a Hardware Platform for Linux

The configuration for each of the hardwares is different. This section describes the hardware configuration steps for Zynq-7000, Zynq UltraScale+ MPSoC and MicroBlaze.

Zynq-7000

The following is a list of hardware requirements for a Zynq-7000 hardware project to boot Linux:

1. One Triple Timer Counter (TTC) (Required).

IMPORTANT: *If multiple TTCs are enabled, the Zynq-7000 Linux kernel uses the first TTC block from the device tree. Ensure the TTC is not used by others.*

2. External memory controller with at least 32 MB of memory (Required)
3. UART for serial console (Required)

IMPORTANT: *If soft IP is used, ensure that the interrupt signal is connected.*

4. Non-volatile memory (Optional). For example, QSPI Flash and SD/MMC
5. Ethernet (Optional, essential for network access)

IMPORTANT: *If soft IP is used, ensure that the interrupt signal is connected.*

Zynq UltraScale+ MPSoC

The following is a list of hardware requirements for a Zynq UltraScale+ MPSoC hardware project to boot Linux:

1. External memory controller with at least 64 MB of memory (Required)
2. UART for serial console (Required)

IMPORTANT: *If soft IP is used, ensure that the interrupt signal is connected*

3. Non-volatile memory (Optional). For example, QSPI Flash, SD/MMC
4. Ethernet (Optional, essential for network access)

IMPORTANT: *If soft IP is used, ensure that the interrupt signal is connected*

MicroBlaze

The following is a list of requirements for a MicroBlaze hardware project to boot Linux:

1. IP core check list:
   - External memory controller with at least 32 MB of memory (Required)
   - Dual channel timer with interrupt connected (Required)
   - UART with interrupt connected for serial console (Required)
   - Non-volatile memory such as Linear Flash or SPI Flash (Optional)
   - Ethernet with interrupt connected (Optional, but required for network access)
2. MicroBlaze CPU configuration:
   - MicroBlaze with MMU support by selecting either Linux with MMU or Low-end Linux with MMU configuration template in the MicroBlaze configuration wizard.

IMPORTANT: *Do not disable any instruction set related options that are enabled by the template, unless you understand the implications of such a change.*

   - The MicroBlaze initial bootloader, called FS-BOOT, has a minimum BRAM requirement. 4 KB is required for Parallel flash and 8K KB for SPI flash when the system boots from non-volatile memory

Exporting the Hardware Platform for PetaLinux

After you have finished configuring your hardware platform, implement the design and build a bitstream if necessary. PetaLinux requires a hardware description file (.hdf) in order to properly initialize your PetaLinux project. This hardware description file is generated by using the “Export Hardware” functionality within Vivado.

During project initialization (or update), PetaLinux generates a device tree source () file, U-Boot configuration header files, and enables Linux kernel drivers based on the hardware description file. These details are explored in Appendix A, Internal Architecture of PetaLinux Projects.

Notes for Zynq UltraScale+ MPSoC

If you want First Stage Boot Loader (FSBL) built for Cortex-R5 boot, you will also need to build it with XSDK since the FSBL built with PetaLinux tools is for A53 boot. For details on how to build the FSBL for Cortex-R5 with XSDK, see *MPSoC Software Development Guide* (UG1137) [Ref 2].

5. Working with a PetaLinux Project

This section contains details on creating a new project, importing a hardware description and configuring project components.

Creating a New Project

After exporting your hardware definition from Vivado, the next step is to create and initialize a new PetaLinux project. The `petalinux-create` tool is used to create the basic project directory:

```
$ petalinux-create --type project --template <TEMPLATE> --name <PROJECT>
```

The parameters are as follows:

- `--template` CPU_TYPE - The supported CPU types are zynqMP, zynq and microBlaze
- `--name` NAME - The name of the project you are building.

The `petalinux-create` tool is used to create a new PetaLinux project directory from a default template. Later steps customize these settings to match the hardware project created previously. If the specified location is on NFS, it changes the TMPDIR to `/tmp/<projname_timestamp>`. If `/tmp/<projname_timestamp>` is also on NFS, then it throws an error. You can change the location as TMPDIR through petalinux-config. Do not configure the same TMPDIR for two different PetaLinux projects. This can cause build errors

TIP: *For more details about the PetaLinux directory structure, see* *PetaLinux Project Directory Structure*

Import Hardware Description

1. To start, change to the directory of your PetaLinux project:

```
$ cd <plnx-proj-root>
```

2. Import the hardware description with the `petalinux-config` command, by giving the path of the directory containing the .hdf file.

```
$ petalinux-config
--get-hw-description=<path-to-directory-which-contains-hardware-description-file>
```

After the initialization, the tool displays the system-level menuconfig interface. This automatic launch of the system-level menuconfig interface only occurs after the first time PetaLinux initializes a project. To return to this menuconfig later, execute `petalinux-config` from within the PetaLinux project directory.

```
Linux Components Selection  --->
Auto Config Settings  --->
-*- Subsystem AUTO Hardware Settings  --->
```

```
DTG Settings --->
Kernel Bootargs  --->
ARM Trusted Firmware Compilation Configuration  --->
PMU FIRMWARE Configuration  --->
u-boot Configuration  --->
Image Packaging Configuration  --->
Firmware Version Configuration  --->
Yocto Settings  --->
```

For details on this menu, see Auto Config Settings.

TIP: *Each option mentioned in the menu-config has its description in the help of the corresponding config option.*

In the menu, move the cursor to "`Subsystem AUTO Hardware Settings --->`" and press `<ENTER>` and go into the menu. The options available are similar to the following:

```
--- Subsystem AUTO Hardware Settings
        System Processor (psu_cortexa53_0) --->
        Memory Settings --->
        Serial Settings --->
        Ethernet Settings --->
        Flash Settings --->
        SD/SDIO Settings --->
        RTC Settings --->
[*]     Advanced bootable images storage Settings --->
```

For more details on this menu, see Auto Config Settings.

The "`Subsystem AUTO Hardware Settings --->`" menu allows customizing system-level hardware and software settings.

When exiting the system-level menuconfig interface the tool may take a few minutes to complete. The tool is parsing the hardware description file to update the device tree, U-Boot configuration, and the Linux kernel configuration files based on your settings. In addition, the tool is using the settings you specified in the "`Auto Config Settings --->`" and "`Subsystem AUTO Hardware Settings --->`" menus to ensure that your system is configured as you intend.

For example, if you select `ps7_ethernet_0` as the Primary Ethernet interface, the tool automatically enables its Linux kernel driver. In addition, it also updates the U-Boot configuration headers to use the selected Ethernet controller, if you select to automatically update U-Boot's configuration.

Configure Project Components

If you want to perform advanced PetaLinux project configuration such as enabling Linux kernel options or modifying flash partitions, use the `petalinux-config` tool with the appropriate `-c COMPONENT` option.

IMPORTANT: *Only Xilinx-specific drivers or optimizations in the Linux kernel configuration are supported by Xilinx technical support.*

The examples below demonstrate how to use `petalinux-config` to review or modify your PetaLinux project configuration.

1. Change into the root directory of your PetaLinux project.

```
$ cd <plnx-proj-root>
```

2. Launch the top level system configuration menu and configure it to meet your requirements:

```
$ petalinux-config
```

3. Launch the Linux kernel configuration menu and configure it to meet your requirements:

```
$ petalinux-config -c kernel
```

4. Launch the root filesystem configuration menu and configure it to meet your requirements:

```
$ petalinux-config -c rootfs
```

TIP: *Set U-BOOT TARGET in petalinux-config menuconfig as required, for your custom board.*
$petalinux-config
Set MACHINE_NAME as required. Values possible are ac701-full, kc705-lite, zc1751-dc1, zc706, zcu102-revb, zedboard, ac701-lite, kcu105, zc1751-dc2, zcu102-rev1.0, kc705-full, zc702, zcu102-reva, and zcu106-reva.

Note: Please make sure board and user specific dtsi entries are added to `project-spec/meta-user/recipes-bsp/device-tree/files/system-user.dtsi`.

Using template flow, for zcu102,zcu106 boards, you need to follow the below steps:

1. Add the following line to `<plnx-proj-root>/project-spec/meta-user/recipes-bsp/fsbl/fsbl_%.bbappend` for fsbl initializations.

```
YAML_COMPILER_FLAGS_append = " -DXPS_BOARD_ZCU102" #for zcu102
YAML_COMPILER_FLAGS_append = " -DXPS_BOARD_ZCU106" # for zcu106
```

PetaLinux automated the system currently, it does not add these macros.

6. Software Testing with QEMU

While prior sections used QEMU to boot a system image for demonstration, this section provides additional details about the QEMU workflow and how to use it effectively. Use the `petalinux-boot` tool with the `--qemu` option to boot the system emulator.

Verbose usage information can be obtained at the command line using `petalinux-boot --qemu -- help`.

IMPORTANT: *The petalinux-boot tool must be run from within a project directory ("<plnx-proj-root>").*

TIP: *Once QEMU is running, it can be exited gracefully by pressing Ctrl + A, and then X.*

Boot the Default Linux Kernel Image

The `--kernel` option is used to boot the project's most recently built Linux image. For Microblaze, it is "`<plnx-proj-root>/images/linux/image.elf`". For Zynq, it is "`<plnx-proj-root>/images/ linux/zImage`". For Zynq UltraScale+ MPSoC, this is "`<plnx-proj-root>/images/linux/Image`".

1. Build the system image using `petalinux-build`
2. After the image has been built, change into the "`<plnx-proj-root>`" directory if not already and run:

   ```
   $ petalinux-boot --qemu --kernel
   ```

3. During start up, you will see the normal Linux boot process, ending with a login prompt as shown below:

   ```
   [ 10.709243] Freeing unused kernel memory: 5568K (ffffffc000c20000 -
   ffffffc001190000)
   [ 13.448003] udevd[1666]: starting version 3.2
   [ 13.458788] random: udevd: uninitialized urandom read (16 bytes read)
   [ 13.556064] udevd[1667]: starting eudev-3.2
   [ 14.045406] random: udevd: uninitialized urandom read (16 bytes read)
   [ 37.446360] random: dd: uninitialized urandom read (512 bytes read)
   [ 40.406936] IPv6: ADDRCONF(NETDEV_UP): eth0: link is not ready
   [ 41.460975] macb ff0e0000.ethernet eth0: link up (100/Full)
   [ 41.474152] IPv6: ADDRCONF(NETDEV_CHANGE): eth0: link becomes ready
   [ 44.787172] random: dropbearkey: uninitialized urandom read (32 bytes read)
   PetaLinux 2017.3 xilinx-zcu102-2017_3 /dev/ttyPS0
   xilinx-zcu102-2017_3 login: root
   Password:
   root@xilinx-zcu102-2017_3:~#
   root@xilinx-zcu102-2017_3:~#
   ```

   You may see slightly different output from the above example, depending on the Linux image you test and its configuration.

4. Login to the virtual system when you see the login prompt on the emulator console with the login `root` and password `root`.
5. Try some Linux commands such as `ls`, `ifconfig`, `cat/proc/cpuinfo` and so on. They behave the same as on real hardware.
6. To exit the emulator when you are finished, press Ctrl + A, release and then X.

Boot a Specific Linux Image

The `petalinux-boot` tool can also boot a specific Linux image, using the image option (`-i` or `--image`):

```
$ petalinux-boot --qemu --image <path-to-Linux-image-file>
```

For example:

```
$ petalinux-boot --qemu --image ./images/linux/zImage
```

Boot a Linux Image with a Specific DTB

Device Trees (DTS / DTB files) are used to describe the hardware architecture and address map to the Linux kernel. The PetaLinux system emulator also uses DTB files to dynamically configure the emulation environment to match your hardware platform.

If no DTB file option is provided, `petalinux-boot` extracts the DTB file from the given `image.elf` for MicroBlaze and from `<plnx-proj-root>/images/linux/system.dtb` for Zynq family device (Zynq and Zynq UltraScale+ MPSoC). Alternatively, you can use the `--dtb` option as follows:

```
$ petalinux-boot --qemu --image ./images/linux/zImage --dtb
./images/linux/system.dtb
```

7. Building a Bootable System Image

Once a Linux system has been built and tested with the PetaLinux tools, the next step is to generate a boot image which can be deployed in the field. This process is straight-forward using the petalinux-package tool.

Generate Boot Image for Zynq Family Devices

This section is for Zynq family devices (Zynq-7000 and Zynq UltraScale+ MPSoC) only.

The ".BIN" boot image can be put into Flash or copied directly to the first FAT partition of an SD card.

A Zynq family device boot image usually contains a first stage bootloader image, (optionally) an FPGA bitstream, and the U-Boot.elf. Additionally, it may also contain the image.ub FIT image. For zynqMP, it additionally contains PMUFW and ATF.

Follow the steps below to generate the boot image in ".BIN" format.

```
$ petalinux-package --boot --format BIN --fsbl <FSBL image> --fpga <FPGA bitstream>
--u-boot
```

For detailed usage, see *PetaLinux Command Line Reference Guide* (UG1157) [Ref 4].

Generate Downloadable Image for MicroBlaze

This section is for MicroBlaze only. There are two options to generate a download-able image for MicroBlaze-based designs. In the first workflow, the petalinux-package command is used to build an MCS programming file.

```
$ petalinux-package --boot --format MCS --flash-size <SIZE> \--flash-intf
<INTERFACE> --fsbl <FSBL image> --fpga <FPGA bitstream> --u-boot
```

In the second workflow, Vivado may be used to generate a bitstream which has the fs-boot.elf initialized to BRAM. For more details on this workflow, see *Vivado Design Suite User Guide* (UG908) [Ref 3].

Internal Architecture of PetaLinux Projects

Working with the PetaLinux Menuconfig System

In this release, the Linux system components available in the sub-menu are shown as follows:

- First stage bootloader
- PMU Firmware, for Zynq UltraScale+ MPSoC only
- U-Boot
- Kernel
- ATF, for Zynq UltraScale+ MPSoC only

For ATF, U-Boot and kernel there are 3 options available

1. Default

   The default component is shipped through PetaLinux tool.

2. External source

   When you have a component downloaded at any specified location, you can feed your component instead of the default one through this config option.

   Note: The external source folder is required to be unique to a project and its user, but the content can be modified.

3. Remote

   If you want to build a component which was on a custom git repo, this config option has to be used.

Auto Config Settings

When a component is selected to enable automatic configuration (autoconfig) in the system-level menuconfig, its configuration files are automatically updated when the `petalinux-config` is run.

Table A-1: **Components and their Configuration Files**

Component in the Menu	Files Impacted when the Autoconfig is enabled
Device tree	The following files are in `<plnx-proj-root>/components/plnx_workspace/device-tree/device-tree-generation/` • `skeleton.dtsi` (Zynq-7000 only) • `zynq-7000.dtsi` (Zynq-7000 only) • `zynqmp.dtsi` (Zynq UltraScale+ MPSoC only) • `zynqmp-clk-ccf.dtsi` (Zynq UltraScale+ MPSoC only) • `pcw.dtsi` (Zynq-7000 and Zynq UltraScale+ MPSoC) • `pl.dtsi` • `system-conf.dtsi` • `system-top.dts` • `<board>.dtsi`
kernel	The following files are in `<plnx-proj-root>/project-spec/meta-plnx-generated/recipes-kernel/linux/configs/` • `plnx_kernel.cfg` • `bsp.cfg`
U-Boot	The following files are in `<plnx-proj-root>/project-spec/meta-plnx-generated/recipes-bsp/u-boot/configs/` • `config.cfg` • `config.mk` (MicroBlaze only) • `platform-auto.h`

Subsystem AUTO Hardware Settings

The Subsystem AUTO Hardware Settings menu allows you to customize how the Linux system interacts with the underlying hardware platform.

System Processor

The System Processor menu specifies the CPU processor on which the system runs.

Memory Settings

The Memory Settings menu allows you to:

- Select which memory IP is the primary system memory
- Set the system memory base address
- Set the size of the system memory
- Set the U-Boot text base address offset to a memory high address

The configuration in this menu impacts the memory settings in the device tree and U-Boot automatic configuration (autoconfig) files.

If `manual` is selected as the primary memory, you are responsible for ensuring proper memory settings for the system.

Serial Settings

The Serial Settings sub-menu allows you to select which serial device is the system's primary STDIN/STDOUT interface. If `manual` is selected as the primary serial, you are responsible for ensuring proper serial interface settings for the system.

Ethernet Settings

The Ethernet Settings sub-menu allows you to:

- Select which Ethernet is the systems' primary Ethernet
- Select to randomize MAC address
- Set the MAC address of the primary Ethernet

  Note: If MAC address is programmed into EEPROM, keep this empty here. Refer U-Boot documentation for commands to program EEPROM and to configure for the same.
- Set whether to use DHCP or static IP on the primary Ethernet

If `manual` is selected as the primary Ethernet, you are responsible for ensuring proper Ethernet settings for the system.

Flash Settings

The Flash Settings sub-menu allows you to:

- Select which flash is the system's primary flash
- Set the flash partition table

If `manual` is selected as the primary flash, you are responsible for the flash settings for the system.

SD/SDIO Settings

The SD/SDIO Settings sub-menu is for Zynq family devices (Zynq-7000 and Zynq UltraScale+ MPSoC) only. It allows you to select which SD controller is the system's primary SD card interface.

If `manual` is selected as the primary flash, you are responsible for the flash settings for the system.

Timer Settings

The Timer Settings sub-menu is for MicroBlaze and Zynq UltraScale+ MPSoC. It allows you to select which timer is the primary timer.

IMPORTANT: *A Primary timer is required for a MicroBlaze system.*

Reset GPIO Settings

The Reset GPIO Settings sub-menu is for MicroBlaze only. It allows you to select which GPIO is the system reset GPIO.

TIP: *MicroBlaze systems use GPIO as a reset input. If a reset GPIO is selected, you can reboot the system from Linux.*

RTC Settings

Select an RTC instance that is used as a primary timer for the Linux kernel. If your preferred RTC is not on the list, select **manual**. In this case, you will be responsible to enable the property kernel driver for your RTC.

Advanced bootable images storage Settings

The advanced bootable images storage settings sub-menu allows you to specify where the bootable images are located. The settings in this sub-menu are used by PetaLinux to configure U-Boot.

If this sub-menu is disabled, PetaLinux uses the flash partition table specified in the "`Flash Settings --- >`" sub-menu to define the location of the bootable images.

Table A-2: **Flash Partition Table**

Bootable Image / U-Boot Environment Partition	Default Partition Name	Description
Boot Image	boot	• `BOOT.BIN` for Zynq family devices (Zynq and Zynq UltraScale+ MPSoC) • Relocatable U-Boot BIN file (`u-boot-s.bin`) for MicroBlaze
U-Boot Environment Partition	bootenv	U-Boot environment variable partition. When **primary sd** is selected, U-Boot Environment is stored in the first partition.When **primary flash** is selected, U-Boot Environment is stored in the partition mentioned in flash partition name option.
Kernel Image	kernel	Kernel image `image.ub` (FIT format)
DTB Image	dtb	If "Advanced bootable images storage Settings" is disabled and a dtb partition is found in the flash partition table settings, PetaLinux configures U-Boot to load the DTB from the partition table. Else, it assumes a DTB is contained in the kernel image.

Kernel Bootargs

The Kernel Bootargs sub-menu allows you to let PetaLinux automatically generate the kernel boot command-line settings in DTS, or pass PetaLinux user defined kernel boot command-line settings. The following are the default bootargs.

```
Microblaze-full -- console=ttyS0,115200 earlyprintk
Microblaze-lite -- console=ttyUL0,115200 earlyprintk
zynq            -- console=ttyPS0,115200 earlyprintk
zynqmp          -- earlycon clk_ignore_unused root=/dev/ram rw
```

For more information, see kernel documentation.

ATF Compilation Configuration

The ATF Compilation Configuration appears only for the ZynqMP platform. This sub-menu allows you to set:

- Extra ATF compilation settings
- Change the base address of bl31 binary
- Change the size of bl31 binary

PMU Firmware Configuration

The PMU Firmware Configuration option allows PetaLinux to add power related kernel configs.

U-Boot Configuration

The U-Boot Configuration sub-menu allows you to select to use U-Boot automatic configuration (autoconfig) by PetaLinux or use a U-Boot board configuration target.

Image Packaging Configuration

The Image Packaging Configuration sub-menu allows you to set the following image packaging configurations:

- Root filesystem type
- File name of the generated bootable kernel image
- Linux kernel image hash function
- DTB padding size
- Whether to copy the bootable images to host TFTP server directory.

TIP: *The `petalinux-build` tool always generates a FIT image as the kernel image.*

Firmware Version Configuration

The Firmware Version Configuration sub-menu allows you to set the firmware version information:

Table A-3: **Firmware Version Options**

Firmware Version Option	File in the Target RootFS
Host name	`/etc/hostname`
Product name	`/etc/petalinux/product`
Firmware Version	`/etc/petalinux/version`

TIP: *The host name does not get updated. Please see AR for more details.*

Yocto Settings

Yocto settings allows you to configure various yocto features available in a project.

Table A-4: **Yocto Settings**

Parameter	Description
TMPDIR Location	This directory is used by bitbake to store logs and build artifacts
Parallel thread execution	To limit the number of threads of bitbake instances

Table A-4: ***(Cont'd)*Yocto Settings**

Parameter	Description
Add pre-mirror url	Adds mirror sites for downloading source code of components
Local sstate feeds settings	To use local sstate cache at a specific location
Enable Debug Tweaks	Login into target without pass word
Enable Network sstate feeds	Enabled NW sstate feeds
User layers	Adds user layers into projects

PetaLinux Project Structure

A PetaLinux project includes many directories and configuration files. The PetaLinux project can be on the Network File System (NFS), but the TMPDIR has to point to any local storage. This section provides an overview of how a PetaLinux project is structured, including which files and/or directories are automatically managed by the PetaLinux tools. In addition, information is included about which files are safe to use with revision control systems such as Git.

Anatomy of a PetaLinux Project

A PetaLinux project is composed of the following components:

- Device tree (DTS / DTB)
- First stage bootloader (optional)
- PMUFW (optional), for Zynq UltraScale+ MPSoC only
- ATF (optional), for Zynq UltraScale+ MPSoC only
- U-Boot (optional)
- Linux kernel
- Root filesystem. The root filesystem is composed of the following sub-components:
    - Prebuilt packages
    - Linux user applications (optional)
    - User modules (optional)

A PetaLinux project directory contains configuration files of the project as well as the components of the system. During build, the `petalinux-build` tool builds the project based on the settings found in these configuration files. The `petlainux-config` tool is used to modify these configurations.

PetaLinux Project Directory Structure

A PetaLinux project has the following basic directory structure:

```
<plnx-proj-root>
     -build
        -bitbake.lock
        -build.log
        - cache
        -conf
        -downloads
        -misc
        -pyshtables.py
        -sstate-cache
        - tmp
     - components
        - plnx_workspace
     - config.project
     - .gitignore
     - hardware
        - xilinx-zc702-2017.3
     - images
        - linux
     - .petalinux
        -metadata
        - usage_statistics
        - usage_statistics_token
     - pre-built
        - linux
     - project-spec
        - attributes
        - configs
        - hw-description
        - meta-plnx-generated
        - meta-user
        - yocto-layer.log
     - README
```

CAUTION! *"`<plnx-proj-root>/build/`" is automatically generated. Do not manually edit it. Contents in this directory will get updated when you run `petalinux-config` or `petalinux-build`. "`<plnx-proj-root>/images/`" is automatically generated and managed. Files in this directory will get updated when you run `petalinux-build`.*

Table A-5: **Project Organization**

Project Path	Description
`".petalinux/"`	Directory to hold tools usage and WebTalk data.
`"<plnx-proj-root>/config.project/"`	Project configuration file.
`"<plnx-proj-root>/project-spec"`	Project specification of the project.
`"<plnx-proj-root>/project-spec/hw-description"`	Hardware description imported from Vivado.

Table A-5: **Project Organization *(Cont'd)***

Project Path	Description
"<plnx-projroot>/project-spec/configs"	Configuration files of top level config and rootfs config
"<plnx-proj-root>/project-spec/configs/config"	Configuration file used store user setiings
"<plnx-proj-root>/project-spec/configs/rootfs_config"	Configuration file used for root filesystem.
"<plnx-proj-root>/components/plnx_workspace/device-tree/device-tree-generation/"	Device tree files used to build device tree. The following files are auto generated by petalinux-config: • `skeleton.dtsi` (Zynq-7000 only) • `zynq-7000.dtsi` (Zynq-7000 only) • `zynqmp.dtsi` (Zynq UltraScale+ MPSoC only) • `pcw.dtsi` (Zynq-7000 and Zynq UltraScale+MPSoC only) • `pl.dtsi` • `system-conf.dtsi` • `system-top.dts` • `zynqmp-clk-ccf.dtsi` (Zynq UltraScale+MPSoC only) • `<board.dtsi>` (Optionally created when the machine name is specified) It is not recommended to edit these files, as these files are regenerated by the tools.
"<plnx-proj-root>/project-spec/meta-user/recipes-bsp/device-tree/files/"	`system-user.dtsi` is not modified by any PetaLinux tools. This file is safe to use with revision control systems. In addition, you can add your own DTSI files to this directory. You have to edit the `project-spec/meta-user/recipes-bsp/device-tree/device-tree-generation_%.bbappend` by adding your dtsi file.
"<plnx-projroot>/project-spec/meta-plnx-generated/recipes-bsp/u-boot/configs"	U-Boot PetaLinux configuration files. The following files are auto generated by petalinux-config: • `config.mk` for MicroBlaze only • `platform-auto.h` • `config.cfg` `platform-top.h` is not modified by any PetaLinux tools. When U-Boot builds, these files are copied into U-Boot build directory `build/linux` as follows: • config is the U-Boot kconfig file. • *config.mk* is copied to board/xilinx/microblaze-generic/ for MicroBlaze.
<plnx-projroot>/project-spec/meta-user/recipes-bsp/u-boot/files/platform-top.h	• `platform-auto.h and platform-top.h` is copied to include/configs/ directory.

Table A-5: **Project Organization *(Cont'd)***

Project Path	Description
`"<plnx-proj-root>/components/"`	Directory for embeddedSW workspace and place to hold external sources while packing BSP. You can also manually copy components into this directory. Here is the rule to place a external component: "<plnx-proj-root>/components/ext_source/<COMPONENT>"(1)

Notes:

1. If the ext_source directory is not present, create a new directory 'ext_source'.
2. It is not mandatory to have external sources inside the PetaLinux project. If any external location is specified, take care of the path and source content while migrating the PetaLinux project

When the project is built, two directories will be auto generated:

- `"<plnx-proj-root>/build"` for the files generated for build
- `"<plnx-proj-root>/images"` for the bootable images

Table A-6: **Build Directory Output and Description**

Build Directory Output	Description
`"<plnx-proj-root>/build/build.log"`	Log file of the build
`"<plnx-proj-root>/build/misc/"`	Directory to hold files related to the project and rootfs configuration
`"<plnx-proj-root>/build/misc/rootfs/"`	Directory to hold files related to the rootfs build
`"<TMPDIR>/work/{MACHINE}-poky-linux-gnueabi/petalinux-ser-image/1.0-r0/rootfs`	Deployment-ready root filesystem. These files are safe to deploy via a live filesystem on disk
`"<TMPDIR>/sysroots/{MACHINE}`	Stage directory to hold the libs and header files required to build user apps
`"<TMPDIR>/work/plnx_aarch64-xilinx-linux/linux-xlnx/"`	Directory to hold files related to the Linux kernel build(1)
`"<TMPDIR>/work/aarch64-xilinx-linux/arm-trusted-firmware/"`	Directory to hold files related to the ATF build
`"<TMPDIR>/work/plnx_aarch64-xilinx-linux/u-boot-xlnx/v2017.03-xilinx-v2017.3+gitAUTOINC+92e3dd638b-r0/"`	Directory to hold files related to the U-Boot build(1)
`"<plnx-proj-root>/components/plnx_workspace/device-tree-generation"`	Directory to hold files related to the device-tree build
`"<plnx-proj-root>/components/plnx_workspace/fsbl/"`	Directory to hold files related to the bootloader build

Table A-6: **Build Directory Output and Description *(Cont'd)***

Build Directory Output	Description
`"<plnx-proj-root>/components/plnx_workspace/pmufw/pmu-firmware/"`	Directory to hold files related to the PMU Firmware build

Notes:
1. By default, the source code is removed after you build, to preserve space. You have to remove INHERIT += "rm_work" line from `local.conf`.
2. By default, after a successful build, the work directory will be deleted. To preserve the build artifacts, delete line INHERIT +="rm_work" in `build/conf/local.conf` before doing the build. This increases the size of TMPDIR.

Note: All components are built out of the tree module and any changes to the source files in the build directory will not be reflected. The machine variable is based on architecture. It is **plnx_aarch64** (for Zynq UltraScale+ MPSoC), **plnx_arm** (for Zynq) and **plnx_microblaze** (for MicroBlaze).

Table A-7: **Image Directory and Description**

Image Directory in a PetaLinux Project	Description
`"<plnx-proj-root>/images/linux/"`	Directory to hold the bootable images for Linux

TIP: *Version control software can be used with the entire PetaLinux project directory* `"<plnx-projroot>"` *excluding* `"<plnx-proj-root>/.petalinux"`, `"<plnx-proj-root>/build"`, `"<plnx-proj-root>/images"` *and* `<plnx-proj-root>/project-spec/meta-plnx-generated/`.

Additional Resources and Legal Notices

Xilinx Resources

For support resources such as Answers, Documentation, Downloads, and Forums, see Xilinx Support.

Solution Centers

See the Xilinx Solution Centers for support on devices, software tools, and intellectual property at all stages of the design cycle. Topics include design assistance, advisories, and troubleshooting tips.

References

1. *PetaLinux Tools Documentation: Reference Guide* (UG1144).
2. *Zynq UltraScale+ MPSoC Software Developer Guide* (UG1137)
3. *Vivado Design Suite User Guide: Programming and Debugging* (UG908)
4. *PetaLinux Command Line Reference* (UG1157)
5. *Platform Cable USB II* (DS593)

Please Read: Important Legal Notices