PetaLinux Tools Documentation

Workflow Tutorial

UG1156 (v2015.2) June 30, 2015

XILINX®

Notice of Disclaimer
The information disclosed to you hereunder (the "Materials") is provided solely for the selection and use of Xilinx products. To the maximum extent permitted by applicable law: (1) Materials are made available "AS IS" and with all faults, Xilinx hereby DISCLAIMS ALL WARRANTIES AND CONDITIONS, EXPRESS, IMPLIED, OR STATUTORY, INCLUDING BUT NOT LIMITED TO ARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, OR FITNESS FOR ANY PARTICULAR PURPOSE; and (2) Xilinx shall not be liable (whether in contract or tort, including negligence, or under any other theory of liability) for any loss or damage of any kind or nature related to, arising under, or in connection with, the Materials (including your use of the Materials), including for any direct, indirect, special, incidental, or consequential loss or damage (including loss of data, profits, goodwill, or any type of loss or damage suffered as a result of any action brought by a third party) even if such damage or loss was reasonably foreseeable or Xilinx had been advised of the possibility of the same. Xilinx assumes no obligation to correct any errors contained in the Materials or to notify you of updates to the Materials or to product specifications. You may not reproduce, modify, distribute, or publicly display the Materials without prior written consent. Certain products are subject to the terms and conditions of the Limited Warranties which can be viewed at http://www.xilinx.com/warranty.htm; IP cores may be subject to warranty and support terms contained in a license issued to you by Xilinx. Xilinx products are not designed or intended to be fail-safe or for use in any application requiring fail-safe performance; you assume sole risk and liability for use of Xilinx products in Critical Applications: http://www.xilinx.com/warranty.htm#critapps.

© Copyright 2015 Xilinx, Inc. Xilinx, the Xilinx logo, Artix, ISE, Kintex, Spartan, Virtex, Vivado, Zynq, and other designated brands included herein are trademarks of Xilinx in the United States and other countries. All other trademarks are the property of their respective owners.

Revision History

Date	Version	Notes
11/24/2014	2014.4	Initial public release for PetaLinux Tools 2014.4
06/30/2015	2015.2	Updated for PetaLinux Tools 2015.2 release

Online Updates

Please refer to the PetaLinux v2015.2 Master Answer Record (Xilinx Answer Record #55776) for the latest updates on PetaLinux Tools usage and documentation.

Table of Contents

Revision History 1

Online Updates 2

Table of Contents 3

Introduction 5
Design Flow Overview 5

Chapter 1: Working With PetaLinux Reference BSP's 6
Rebuilding the Reference Design Software Image 7
Compile PetaLinux Reference Design System 7

Chapter 2: Test Drive a PetaLinux BSP Image 10
Testing the Pre-Built PetaLinux Images 10
Testing the Pre-Built PetaLinux Image on Hardware 10
Boot Pre-built Images from SD Card (Zynq Only) 10
Boot Pre-built Images with JTAG 11
Troubleshooting 13
Test Pre-Built PetaLinux Image with QEMU 13
Testing the Re-Built PetaLinux Images 14
Test the Rebuilt Image on Hardware 14
Test the Rebuilt Image with QEMU 15

Chapter 3: Configuring Custom Hardware for Embedded Linux 16
Configuring a Hardware Platform for Linux 16
Zynq 16
MicroBlaze 17
Exporting the Hardware Platform for PetaLinux 18

Chapter 4: Working with a PetaLinux Project 19
Creating a New Project 19
Import Hardware Description 19
Configure Project Components 20

Chapter 5: Software Testing with QEMU 21
Exiting the QEMU Emulator 21
Boot the Default Linux Kernel Image 21
Boot a Specific Linux Image 22
Boot a Linux Image with a Specific DTB 23

Chapter 6: Building a Bootable System Image 24
Generate Boot Image for Zynq 24
Generate Downloadable Image for MicroBlaze 24

Appendix A: Internal Architecture of PetaLinux Projects **25**
Working with the PetaLinux Menuconfig System 25
Auto Config Settings 25
Subsystem AUTO Hardware Settings 26
System Processor 26
Memory Settings 26
Serial Settings 26
Ethernet Settings 26
Flash Settings 27
SD/SDIO Settings 27
Timer Settings 27
Reset GPIO Settings 27
Advanced bootable images storage Settings 27
Kernel Bootargs Sub-Menu 28
U-boot Configuration Sub-Menu 28
Image Packaging Configuration Sub-Menu 28
Firmware Version Configuration Sub-Menu 28
PetaLinux Project Structure 29
Anatomy of a PetaLinux Project 29
PetaLinux Project Directory Structure 30

Additional Resources **34**
References 34

Introduction

This tutorial demonstrates how to successfully use the PetaLinux design workflow through a series of real world examples. This tutorial aligns with PetaLinux v2015.2 and some syntax or command line options may not apply or work in the same manner for different versions of PetaLinux. The tutorial assumes that the user has already installed and licensed both Vivado and PetaLinux. Consult the PetaLinux v2015.2 Reference Guide for more details on installation and licensing.

In general, the methodologies and steps presented here are universal to all PetaLinux designs. The specific examples presented here utilize the Zynq AP SoC device family. Where appropriate, differences or nuances specific to MicroBlaze-based FPGA designs are denoted in context.

For additional details on the specific PetaLinux tools or command line options, please see the PetaLinux Reference Guide. For more information on specific PetaLinux workflows, please see UGxxx - PetaLinux Design Flows Reference Guide.

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 tasks should be completed and the corresponding tool or workflow for that task.

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
Test the System	petalinux-boot
Deploy the System	petalinux-package

Chapter 1: Working With PetaLinux Reference BSP's

PetaLinux Reference BSP's 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 (Board Support Package) 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 will need to be downloaded and installed separately. PetaLinux BSP packages are available on the Xilinx.com Download Center.

Below are the 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>
```

   You will see output similar to the following, according to which BSP you are installing:

```
INFO: Create project:

INFO: Projects:
INFO:   * Xilinx-ZC702-2015.2
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 which projects have been extracted from the BSP and installed. If you run `ls` from `"/home/user"`, you will see the installed project(s).
Please refer to the section *PetaLinux Project Structure* in Appendix A for more details on the structure of a PetaLinux project.

Rebuilding the Reference Design Software Image

So far, you have installed a PetaLinux reference BSP and explored its contents. The following section describes how to rebuild the BSP image so that you can test in QEMU or on hardware.

Compile PetaLinux Reference Design System

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

2. The compilation progress will show on the console. Wait until the compilation finishes.

> **TIP:**
>
> - *A detailed compilation log will be 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 the user 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 `Image Packing Configuration` menu of the system-level configuration.*

Here is an example of the compilation progress output:

```
INFO: Checking component...
INFO: Generating make files and build linux
INFO: Generating make files for the subcomponents of linux
INFO: Building linux
[INFO ] pre-build linux/rootfs/fwupgrade
[INFO ] pre-build linux/rootfs/peekpoke
[INFO ] pre-build linux/rootfs/uWeb
[INFO ] build system.dtb
[INFO ] build linux/kernel
[INFO ] update linux/u-boot source
[INFO ] generate linux/u-boot configuration files
[INFO ] build linux/u-boot
[INFO ] Setting up stage config
[INFO ] Setting up rootfs config
[INFO ] Updating for armv7a-vfp-neon
[INFO ] Updating package manager
[INFO ] Expanding stagefs
[INFO ] build linux/rootfs/fwupgrade
[INFO ] build linux/rootfs/peekpoke
[INFO ] build linux/rootfs/uWeb
[INFO ] build kernel in-tree modules
[INFO ] modules linux/kernel
[INFO ] post-build linux/rootfs/fwupgrade
[INFO ] post-build linux/rootfs/peekpoke
[INFO ] post-build linux/rootfs/uWeb
[INFO ] pre-install linux/rootfs/fwupgrade
[INFO ] pre-install linux/rootfs/peekpoke
[INFO ] pre-install linux/rootfs/uWeb
[INFO ] install linux/kernel
[INFO ] install linux/u-boot
[INFO ] Setting up rootfs config
[INFO ] Setting up stage config
[INFO ] Updating for armv7a-vfp-neon
[INFO ] Updating package manager
[INFO ] Expanding rootfs
[INFO ] install sys_init
[INFO ] install linux/rootfs/fwupgrade
[INFO ] install linux/rootfs/peekpoke
[INFO ] install linux/rootfs/uWeb
[INFO ] install kernel in-tree modules
[INFO ] modules_install linux/kernel
[INFO ] post-install linux/rootfs/fwupgrade
[INFO ] post-install linux/rootfs/peekpoke
[INFO ] post-install linux/rootfs/uWeb
[INFO ] package rootfs.cpio to /home/user/ZC702/images/linux
[INFO ] Update and install vmlinux image
[INFO ] vmlinux linux/kernel
[INFO ] install linux/kernel
[INFO ] package zImage
[INFO ] zImage linux/kernel
[INFO ] install linux/kernel
[INFO ] package FIT image
```

Figure 1: Compilation progress output

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

Chapter 2: Test Drive a PetaLinux BSP Image

In Chapter 1 you 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'll also boot the system image that you rebuilt in Chapter 1 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 Only)

This section is for Zynq only as Zynq allows to boot from SD card.

Here are the steps on how to boot pre-built images from SD card:

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 (e.g. `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-PetaLinux> run sdboot
```

11. Boot messages which similar to the following will appear in the serial console:

```
INIT: version 2.88 booting
random: nonblocking pool is initialized
Creating /dev/flash/* device nodes
starting Busybox inet Daemon: inetd... done.
Starting uWeb server:
NET: Registered protocol family 10
update-rc.d: /etc/init.d/run-postinsts exists during rc.d purge (continuing)
 Removing any system startup links for run-postinsts ...
  /etc/rcS.d/S99run-postinsts
INIT: Entering runlevel: 5
Configuring network interfaces... udhcpc (v1.22.1) started
Sending discover...
Sending select for 10.0.2.15...
Lease of 10.0.2.15 obtained, lease time 86400
xemacps e000b000.ethernet: Set clk to 124999999 Hz
xemacps e000b000.ethernet: link up (1000/FULL)
/etc/udhcpc.d/50default: Adding DNS 10.0.2.3
done.

Built with PetaLinux v2015.2 (Yocto 1.8) Xilinx-ZC702-2015_2 /dev/ttyPS0
Xilinx-ZC702-2015_2 login:
```

Figure 2: Serial console output of `Zynq SD boot`

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

Boot Pre-built Images with JTAG

You can also 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 boards, ensure the mode switches are set to JTAG mode. Refer to the board documentation for details.
6. Power on the board.
7. Open a console on your workstation and then start your preferred serial communication program (e.g. `kermit`, `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 linux kernel. This command will take 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 figures below are examples of the messages displayed on the workstation command console and on the serial console during this process.

```
$ petalinux-boot --jtag --prebuilt 3
INFO: The image provided is a zImage and no addition options were provided
INFO: Append dtb - /home/user/Xilinx-ZC702-2015.2/pre-built/linux/images/system.dtb
and other options to boot zImage
INFO: Configuring the FPGA...
INFO: FPGA configuration completed.
INFO: Downloading FSBL
INFO: FSBL download completed.
INFO: Launching XMD for file download and boot.
INFO: This may take a few minutes, depending on the size of your image.
```

Figure 3: Workstation console output for successful `petalinux-boot`

```
INIT: version 2.88 booting
random: nonblocking pool is initialized
Creating /dev/flash/* device nodes
starting Busybox inet Daemon: inetd... done.
Starting uWeb server:
NET: Registered protocol family 10
update-rc.d: /etc/init.d/run-postinsts exists during rc.d purge (continuing)
 Removing any system startup links for run-postinsts ...
  /etc/rcS.d/S99run-postinsts
INIT: Entering runlevel: 5
Configuring network interfaces... udhcpc (v1.22.1) started
Sending discover...
Sending select for 10.0.2.15...
Lease of 10.0.2.15 obtained, lease time 86400
xemacps e000b000.ethernet: Set clk to 124999999 Hz
xemacps e000b000.ethernet: link up (1000/FULL)
/etc/udhcpc.d/50default: Adding DNS 10.0.2.3
done.

Built with PetaLinux v2015.2 (Yocto 1.8) Xilinx-ZC702-2015_2 /dev/ttyPS0
Xilinx-ZC702-2015_2 login:
```

Figure 4: Serial console output of `petalinux-boot`

By default, network settings for PetaLinux reference designs are configured using 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.

9. Use the login name "`root`" and password "`root`" on the serial console to log into the Linux system.
10. Determine the IP address of the PetaLinux by running `ifconfig` on the system console.

Troubleshooting

If your local network does not have a DHCP server, the system will fail to acquire an IP address. If so, refer to 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. Please ensure the board is powered on and your JTAG cable is properly connected. Please refer to the Xilinx JTAG cable and tools documentation for more detailed troubleshooting.

TIP: *JTAG connections may not work as expected when run from within a virtual machine environment. 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.

   You should see the following kernel boot log on the console:

```
INIT: version 2.88 booting
random: nonblocking pool is initialized
Creating /dev/flash/* device nodes
starting Busybox inet Daemon: inetd... done.
Starting uWeb server:
NET: Registered protocol family 10
update-rc.d: /etc/init.d/run-postinsts exists during rc.d purge (continuing)
 Removing any system startup links for run-postinsts ...
  /etc/rcS.d/S99run-postinsts
INIT: Entering runlevel: 5
Configuring network interfaces... udhcpc (v1.22.1) started
Sending discover...
Sending select for 10.0.2.15...
Lease of 10.0.2.15 obtained, lease time 86400
xemacps e000b000.ethernet: Set clk to 124999999 Hz
xemacps e000b000.ethernet: link up (1000/FULL)
/etc/udhcpc.d/50default: Adding DNS 10.0.2.3
done.

Built with PetaLinux v2015.2 (Yocto 1.8) Xilinx-ZC702-2015_2 /dev/ttyPS0
Xilinx-ZC702-2015_2 login:
```

Figure 5: Serial console output of successful `petalinux-boot` prebuilt

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 prebuilt image on real hardware via JTAG or in the QEMU software. Now you will boot the image that you rebuilt in Chapter 1 using the same methods.

Test the Rebuilt Image on Hardware

Let's test the rebuilt software image on real hardware. First, follow the instructions from the *Testing the Pre-Built PetaLinux Image on Hardware* section to connect the board, serial and JTAG correctly.

1. (Zynq Only) Use `petalinux-boot` to boot the Zynq AP SoC with the FSBL and U-Boot

```
$ petalinux-boot --jtag --u-boot
```

2. Use `petalinux-boot` to program the FPGA with the reference design pre-built bitstream:

```
$ petalinux-boot --jtag --prebuilt 1
```

This command will take a few moments. Wait until you see the shell prompt again on the command console before proceeding.

3. Use `petalinux-boot` to download the built Linux image to the board and boot it:

```
$ petalinux-boot --jtag --kernel
```

This command will take a few minutes as it is downloading the entire kernel image over JTAG.

4. 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, you can refer to the section *Boot Pre-built Images from SD Card (Zynq Only)* to understand the basic steps. Instead of using the prebuilt `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 will be 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*

Chapter 3: Configuring Custom Hardware for Embedded Linux

In the prior chapters we explored a PetaLinux BSP reference design. Now we'll 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

Zynq

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

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

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

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

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

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

> **IMPORTANT:** *If soft IP is used, ensure 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 32MB 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. 4KByte is required for Parallel flash and 8KByte 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 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 (DTS), 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* .

Chapter 4: Working with a PetaLinux Project

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 <CPU_TYPE> --name <PROJECT_NAME>
```

The parameters are as follows:

- `--template` *TYPE* - The supported CPU types are `zynq` and `microblaze`
- `--name` *NAME* - The name of the project you are building.

This tool will create a new PetaLinux project folder from a default template. Later steps customise these settings to match the hardware project created previously.

TIP: *For more details about the PetaLinux directory structure, please refer to PetaLinux Project Directory Structure .*

Import Hardware Description

1. To start, change to the directory that contains the hardware description file generated from Vivado: E.g.

   ```
   $ cd <directory which contains hardware description file>
   ```

2. Import the hardware description with the `petalinux-config --get-hw-description` workflow as follows:

   ```
   $ petalinux-config --get-hw-description -p <plnx-proj-root>
   ```

   The `-p` option points to the PetaLinux project directory that will be initialized or updated to match the hardware platform configuration.

   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  --->
   Kernel Bootargs  --->
   u-boot Configuration  --->
   Image Packaging Configuration  --->
   Firmware Version Configuration  --->
   ```

   Please refer to *Auto Config Settings* for details on this menu.

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

```
--- Subsystem AUTO Hardware Settings
System Processor (ps7_cortexa9_0)  --->
Memory Settings  --->
Serial Settings  --->
Ethernet Settings  --->
Flash Settings  --->
SD/SDIO Settings  --->
[ ]   Advanced bootable images storage Settings  --->
```

Please refer to *Auto Config Settings* for the details of this menu.

The `"Subsystem AUTO Hardware Settings --->"` menu allows customising 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 will automatically enable its Linux kernel driver. In addition, it will also update the U-Boot configuration headers to use the selected Ethernet controller if the user elects 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
```

Chapter 5: Software Testing with QEMU

While prior chapters used QEMU to boot a system image for demonstration, this chapter provides additional details about the QEMU workflow and how to use it effectively. The `petalinux-boot` tool, using the `--qemu` option, is used to boot the system emulator.

Verbose usage information can be obtained with 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>`").*

Exiting the QEMU Emulator

Once QEMU is running, it can be exited gracefully by pressing the `Ctrl + A` keys, 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, this is `"<plnx-proj-root>/images/linux/image.elf"`. For Zynq, this is `"<plnx-proj-root>/images/linux/zImage"`.

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:

```
INIT: version 2.88 booting
random: nonblocking pool is initialized
Creating /dev/flash/* device nodes
starting Busybox inet Daemon: inetd... done.
Starting uWeb server:
NET: Registered protocol family 10
update-rc.d: /etc/init.d/run-postinsts exists during rc.d purge (continuing)
 Removing any system startup links for run-postinsts ...
  /etc/rcS.d/S99run-postinsts
INIT: Entering runlevel: 5
Configuring network interfaces... udhcpc (v1.22.1) started
Sending discover...
Sending select for 10.0.2.15...
Lease of 10.0.2.15 obtained, lease time 86400
xemacps e000b000.ethernet: Set clk to 124999999 Hz
xemacps e000b000.ethernet: link up (1000/FULL)
/etc/udhcpc.d/50default: Adding DNS 10.0.2.3
done.

Built with PetaLinux v2015.2 (Yocto 1.8) Xilinx-ZC702-2015_2 /dev/ttyPS0
Xilinx-ZC702-2015_2 login:
```

Figure 6: Boot PetaLinux Linux Image with QEMU

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're 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. Alternatively, you can use the `--dtb` option as follows:

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

Chapter 6: 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 straightforward using the `petalinux-package` tool.

Generate Boot Image for Zynq

This section is for Zynq devices only. Skip this section for MicroBlaze targets.

The `".BIN"` boot image can be put into Flash (using a PROM programmer) or copied directly to the first FAT partition of an SD card.

A Zynq 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.

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, please refer to the PetaLinux Command Reference Guide.

Generate Downloadable Image for MicroBlaze

This section is for MicroBlaze only. Skip this section for Zynq targets.

There are two options to generate an downloadable 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 initialised to BRAM. Consult the Vivado documentation for details on this workflow.

Appendix A: 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`
- `u-boot`
- `kernel`
- `rootfs`

Auto Config Settings

If a component is selected to enable automatic configuration (autoconfig) in the system-level menuconfig, its configuration files will be automatically updated when `petailnux-config` is run.

component in the menu	Files impacted when autoconfig is enabled
Device tree	• `<plnx-proj-root>/subsystems/linux/configs/device-tree/skeleton.dtsi (Zynq only)` • `<plnx-proj-root>/subsystems/linux/configs/device-tree/zynq-7000.dtsi (Zynq only)` • `<plnx-proj-root>/subsystems/linux/configs/device-tree/pcw.dtsi (Zynq only)` • `<plnx-proj-root>/subsystems/linux/configs/device-tree/pl.dtsi` • `<plnx-proj-root>/subsystems/linux/configs/device-tree/system-conf.dtsi`
kernel	`<plnx-proj-root>/subsystems/linux/configs/kernel/config`
rootfs	`<plnx-proj-root>/subsystems/linux/configs/rootfs/config`
u-boot	• `<plnx-proj-root>/subsystems/linux/configs/u-boot/config.mk` • `<plnx-proj-root>/subsystems/linux/configs/u-boot/platform-auto.h`

If device tree autoconfig is enabled, the kernel configuration file `<plnx-proj-root>/subsystems/linux/configs/kernel/config` will be automatically updated with the top system level settings when `petalinux-config` runs.

Subsystem AUTO Hardware Settings

The menu allows the user to customize how the Linux system interacts with the underlying hardware platform.

System Processor

This menu specifies the CPU processor on which the system runs.

Memory Settings

This menu allows users 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 kernel base address for Zynq only
- set the u-boot text base address offset to a memory high address

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

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

Serial Settings

This sub-menu allows users to select which serial device is the system's primary STDIN/STDOUT interface. If `manual` is selected as the primary serial, the user is responsible for ensuring proper serial interface settings for the system.

Ethernet Settings

This sub-menu allows users to:

- select which Ethernet is the system's primary Ethernet
- set the MAC address of the primary Ethernet
- set whether to use DHCP or static IP on the primary Ethernet.

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

Flash Settings

This sub-menu allows users to:

- select which flash is the system's primary flash
- set the flash partition table

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

SD/SDIO Settings

This sub-memu is for Zynq devices only. It allows users to select which SD controller is the system's primary SD card interface.

Timer Settings

This sub-menu is for MicroBlaze only. It allows users to select which timer is the primary timer.

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

Reset GPIO Settings

This sub-menu is for MicroBlaze only. It allows users 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.*

Advanced bootable images storage Settings

This sub-menu allows users 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 will use the flash partition table specified in the `"Flash Settings --->"` sub-menu to define the location of the bootable images.

Bootable Image / U-Boot Environment Partition	Default Partition Name	Description
Boot Image	boot	• BOOT.BIN for Zynq • Relocatable U-Boot BIN file (`u-boot-s.bin`) for MicroBlaze

U-Boot Environment Partition	bootenv	U-Boot environment variable partition. In this release, PetaLinux U-Boot configuration supports the U-Boot `env` in flash only.
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 will assume a DTB is contained in the kernel image.

Kernel Bootargs Sub-Menu

This sub-menu allows users to let PetaLinux automatically generate the kernel boot command line settings in DTS, or pass PetaLinux user defined kernel boot command line settings.

U-boot Configuration Sub-Menu

This sub-menu allows users to elect to use U-Boot automatic configuration (autoconfig) by PetaLinux or use a U-Boot board configuration target.

Image Packaging Configuration Sub-Menu

This sub-menu allows users 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 Sub-Menu

This sub-menu allows users to set the firmware version information:

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

PetaLinux Project Structure

A PetaLinux project includes many directories and configuration files. 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)
- U-Boot (optional)
- Linux kernel
- root filesystem. The root filesystem is composed of the following sub-components:
  - prebuilt packages
  - Linux user applications (optional)
  - Linux user libraries (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>
    |-.petalinux/
    |-hw-description/
    |-config.project
    |-subsystems/
    |    |-linux/
    |    |    |-config
    |    |    |-hw-description/
    |    |    |-configs/
    |    |    |    |-device-tree/
    |    |    |    |-kernel/
    |    |    |    |-u-boot/
    |    |    |    |-rootfs/
    |-components/
    |    |-apps/
    |    |    |-myapp/
    |-build/
    |    |-build.log
    |    |-linux/
    |    |    |-rootfs/
    |    |    |    |-targetroot/
    |    |    |    |-stage/
    |    |    |    |-apps/
    |    |    |    |    |-myapp/
    |    |    |-kernel/
    |    |    |-u-boot/
    |    |    |-device-tree/
    |    |    |-bootloader/
    |-images/
    |    |-linux/
```

WARNING: *"`<plnx-proj-root>/build/`" is automatically generated. Don't manually edit. 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`.

Project Path	Description
"<plnx-proj-root>/.petalinux/"	Directory to hold tools usage and WebTalk data
"<plnx-proj-root>/hw-description/"	Project level hardware description. NOT USED for this release.
"<plnx-proj-root>/config.project"	Project configuration file. It defines the external components search path and the subsystem in the project.
"<plnx-proj-root>/subsystems/"	Subsystems of the project
"<plnx-proj-root>/subsystems/linux/"	Linux subsystem. This is the only subsystem supported.
"<plnx-proj-root>/subsystems/linux/config"	Linux subsystem configuration file
"<plnx-proj-root>/subsystems/linux/hw-description/"	Subsystem hardware description imported from Vivado
"<plnx-proj-root>/subsystems/linux/configs/"	Configuration files of the components of the subsystem
"<plnx-proj-root>/subsystems/linux/configs/kernel/config"	Configuration file used to build the Linux kernel
"<plnx-proj-root>/subsystems/linux/configs/rootfs/config"	Configuration file used to build the root filesystem
"<plnx-proj-root>/subsystems/linux/configs/device-tree"	Device tree files used to build device tree The following files are auto generated by petalinux-config: • skeleton.dtsi (Zynq only) • zynq-7000.dtsi (Zynq only) • pcw.dtsi (Zynq only) • pl.dtsi • system-conf.dtsi system-top.dts is not modified by any PetaLinux tools and is under full control by user. This file is safe to use with revision control systems. In addition, users can add their own DTSI files to this directory.

"<plnx-proj-root>/subsystems/linux/configs/u-boot"	U-Boot PetaLinux configuration files. The following files are auto generated by petalinux-config: • config.mk • platform-auto.h platform-top.h will not be modified by any PetaLinux tools and is under full control by user. When U-Boot builds, these files are copied into U-Boot build directory build/linux/u-boot/src/<U_BOOT_SRC>/ as follows: • config.mk is copied to board/xilinx/zynq/ for Zynq and board/xilinx/microblaze-generic/ for MicroBlaze. • platform-auto.h and platform-top.h is copied to include/configs/ directory.
"<plnx-proj-root>/components/"	Directory for local components such as applications and libraries. If you don't have local components, this directory is not required. Components created by petalinux-create will be placed into this directory. You can also manually copy components into this directory. Here is the rule to place a local component: "<plnx-proj-root>/components/<COMPONENT_TYPE>/<COMPONENT>"

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

Build Directory Output	Description
"<plnx-proj-root>/build/build.log"	Log file of the build
"<plnx-proj-root>/build/linux/"	Directory to hold files related to the linux subsystem build
"<plnx-proj-root>/build/linux/rootfs/"	Directory to hold files related to the rootfs build
"<plnx-proj-root>/build/linux/rootfs/targetroot/"	Deployment-ready root filesystem. These files are safe to deploy via a live filesystem on disk.
"<plnx-proj-root>/build/linux/rootfs/stage/"	Stage directory to hold the libs and header files required to build user apps/libs
"<plnx-proj-root>/build/linux/kernel/"	Directory to hold files related to the Linux kernel build
"<plnx-proj-root>/build/linux/u-boot/"	Directory to hold files related to the U-Boot build
"<plnx-proj-root>/build/linux/device-tree/"	Directory to hold files related to the device-tree build
"<plnx-proj-root>/build/linux/bootloader/"	Directory to hold files related to the bootloader build

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-proj-root>" excluding the following:*

- *"<plnx-proj-root>/.petalinux"*
- *"<plnx-proj-root>/build"*
- *"<plnx-proj-root>/images"*

Additional Resources

References

PetaLinux Tools Documentation is available at http://www.xilinx.com/petalinux.