PetaLinux Tools User Guide

Board Bringup Guide

UG980 (v2014.2) June 3, 2014

XILINX®

Revision History

Date	Version	Notes
2012-08-03	3.1	Updated for PetaLinux Tools 3.1 release
2012-09-03	12.9	Updated for PetaLinux Tools 12.9 release
2012-12-17	2012.12	Updated for PetaLinux Tools 2012.12 release
2013-04-29	2013.04	Updated for PetaLinux Tools 2013.04 release
2013-11-25	2013.10	Updated for PetaLinux Tools 2013.10 release
2014-06-03	2014.2	Updated for PetaLinux Tools 2014.2 release

Online Updates

Please refer to the PetaLinux v2014.2 Master Answer Record (Xilinx Answer Record #55776) for the latest updates on PetaLinux Tools usage and documentation.

XILINX

Table of Contents

About this Guide

One of the great strengths of an FPGA platform is the ability to completely customise your processor system architecture, either for an off-the-shelf evaluation board, or for your own custom designs.

PetaLinux Tools was created to embrace that flexibility, and includes a set of tools specifically designed to make it as easy as possible to boot a Zynq or MicroBlaze Linux platform on a new board or CPU subsystem design.

This document assumes a working knowledge of the Xilinx embedded development tools. It also assumes that you are familiar with the basics of working with PetaLinux Tools.

It is strongly recommended that you first become familiar with PetaLinux Tools by using the provided pre-built BSPs and reference designs, before attempting a custom board bringup.

Overview

Broadly, there are two stages to the board bringup process:

1. Create and/or configure a hardware platform ready for PetaLinux.
2. Export the hardware platform configuration settings into the new software platform and complete any further software platform configuration steps.

Hardware Platform

You can create a hardware platform with Vivado. Regardless of how the hardware platform is created and configured, there is a small number of hardware IP and software platform configuration changes required to make the hardware platform Linux ready. These are described below.

Configure a Hardware Platform for Linux

Zynq

The following is a list of 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 AXI

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.

Export Hardware

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

PetaLinux tools can generate device tree source file, u-boot config header files, and enable some Xilinx IP kernel drivers based on the hardware description file. This will be descibed in later sections.

Software Platform

Create New Project

The next step is to create a new PetaLinux Tools software platform, ready for building a Linux system customised to your new hardware platform. The `petalinux-create` command is used to achieve this:

```
$ petalinux-create --type project --template <CPU_TYPE> --name <PROJECT_NAME>
```

The parameters are as follows:

- `--template <CPU_TYPE>` - The supported CPU types are `zynq` and `microblaze`
- `--name <PROJECT_NAME>` - The name of the project you are building.

This command 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 detail of PetaLinux project structure, please refer to Appendix B in the PetaLinux Tools Getting Started Guide (UG977) .*

Import Hardwre Description

1. Go to the the directory which contains the hardware descriptiotn file generated from Vivado: E.g.

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

2. Import hardware description with `petalinux-config` command inside `<directory contains hardware description file>` as follows:

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

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

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

   ```
   linux Components Selection  --->
   Auto Config Settings  --->
   -*- Subsystem AUTO Hardware Settings  --->
   Kernel Bootargs  --->
   u-boot Configuration  --->
   Image Packaging Configuration  --->
   Firmware Version Configuration  --->
   ```

   Please refer to *Appendix A: Linux System Menu* for details on this menu.

   Make sure "`Subsystem AUTO Hardware Settings --->`" is selected, and go into the menu which is 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 *Appendix A: Linux System Menu* subsection *subsystem AUTO Hardware Settings* for the details of the menu.

"Subsystem AUTO Hardware Settings --->" menu allows customising system wide hardware settings.

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

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

Configure Project Components

If you want to do advanced PetaLinux project configurations such as enabling some kernel options, modifying the default flash partition table settings and so on, or you just want to view the current configuration, you can run petalinux-config to do so.

This section talks about how to use petalinux-config to configure your PetaLinux project or just view the configuration.

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

```
$ cd <project-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 rootfs configuration menu and configure it to meet your requirements:

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

Build System Image

1. Build the hardware bitstream with Xilinx Vivado tool, if you have not done so already.
2. Change into the directory of your PetaLinux project.

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

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

```
$ petalinux-build
```

The console shows the compilation progress. e.g.:

```
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
```

The full compilation log `"build.log"` is stored in the `build` subdirectory of your PetaLinux project.

The software images and the device tree are generated in the `images` subdirectory of your PetaLinux project.

IMPORTANT: *By default, besides the kernel, rootfs and u-boot, the PetaLinux project is configured to generate and build the first stage bootloader. Please refer to Appendix B: Generate First Stage Bootloader Within Project for more details on the auto generated first stage bootloader.*

Generate Boot Image for Zynq

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

The boot image can be put into Flash or SD card, when you power on the board, it can boot from the boot image. A boot image usually contains a first stage bootloader image, FPGA bitstream and u-boot.

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

```
$ petalinux-package --boot --fsbl <FSBL image> --fpga <FPGA bitstream> --uboot
```

For detailed usage, please refer to the --help option or Appendix C in the PetaLinux Tools Getting Started Guide (UG977) .

Generate Downloadable Bitstream for MicroBlaze

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

Use Vivado to generate the bitstream which has the `"fs-boot"` initialised to BRAM.

Boot System Image on Board

After the hardware bitstream and the software images have been built, you can now test your new PetaLinux platform.

MicroBlaze Kernel Boot via JTAG

IMPORTANT: *The section is for MicroBlaze only*

1. Change into the root directory of your project.

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

2. Program the FPGA using the Xilinx JTAG programming tools
3. Use `petalinux-boot --jtag` to download the image to the board and boot it:

```
$ petalinux-boot --jtag --image images/linux/image.elf
```

Please note direct kernel boot via JTAG can take a while, depending on the kernel image size. Please do not interrupt this process, otherwise JTAG cable lockup can occur.

Zynq Kernel Boot

IMPORTANT: *The section is for Zynq only*

There are many different way to boot a Zynq kernel, for example:

1. SD boot by copying the `"BOOT.BIN"` and `"image.ub"` files to an SD card.
2. JTAG boot with manual `xmd` operation.
3. JTAG boot via PetaLinux prebuilt capability.

This section describes the simplest method to JTAG boot the kernel, using the PetaLinux prebuilt capability. Follow the steps to package a prebuilt image and boot it.

1. Change into the root directory of your project.

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

2. Please ensure the board boot mode is set to JTAG.
3. Ensure you have run `petalinux-package --boot` to generate BOOT.BIN as described in section *Generate Boot Image for Zynq*
4. Use `petalinux-package --prebuilt` to package the prebuilt images:

```
$ petalinux-package --prebuilt --fpga <FPGA bitstream>
```

For detailed usage, please refer to the --help option or Appendix C in the PetaLinux Tools Getting Started Guide (UG977) .

5. Use `petalinux-boot --jtag` to download the images to the board and boot it:

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

For detailed usage, please refer to the --help option or Appendix C in the PetaLinux Tools Getting Started Guide (UG977) .

Troubleshooting

This section describes some common issues you may experience when performing board bring up with PetaLinux Tools, and the possible ways to solve them.

Problem/Error Message	Description and Solution
Cannot see any console output when trying to boot u-boot or kernel on hardware but boots correctly on QEMU.	***Problem Description:*** This problem is usually caused by one or more of the following: • The serial communication terminal application is set with the wrong baud rate. • Mismatch between hardware and software platforms. ***Solution:*** • Ensure your terminal application baud rate is correct and matches your hardware configuration. • Ensure the PetaLinux project with built against the right hardware platform. ◦ Rerun `petalinux-config --get-hw-description` again from the directory which contains the hardware description file. ◦ Check the `"Subsystem AUTO Hardware Settings --->"` submenu to make sure it matches the hardware platform ◦ Ensure `stdout`, `stdin` are set to the correct UART IP core. ◦ Rebuild software images

Appendix A: Linux System Menu

Linux Components Selection

In this release, the linux system components available in the submenu are shown as follows:

- `first stage bootloader`
- `u-boot`
- `kernel`
- `rootfs`

Auto Config Settings

If a component is selected to enable autoconfig, when `petailnux-config` is run, its config files will be auto updated based on the top system level settings.

component in the menu	Files impacted when autoconfig is enabled
Device tree	• `<project-root>/subsystems/linux/configs/device-tree/ps.dtsi` • `<project-root>/subsystems/linux/configs/device-tree/pl.dtsi` • `<project-root>/subsystems/linux/configs/device-tree/system-conf.dtsi`
kernel	`<project-root>/subsystems/linux/configs/kernel/config`
rootfs	`<project-root>/subsystems/linux/configs/rootfs/config`
u-boot	• `<project-root>/subsystems/linux/configs/u-boot/config.mk` • `<project-root>/subsystems/linux/configs/u-boot/platform-auto.h`

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

subsystem AUTO Hardware Settings

System Processor

The processor on which the system runs.

Memory Settings

This menu allow user 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 impat the memory settings in the device tree and u-boot PetaLinux auto config files.

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

Serial Settings

This submemu allows users to select which serial is the system's primary stdin/stdout. If `manual` is selected as the primary serial, the user is responsible for the serial settings for the system.

Ethernet Settings

This submenu 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 the ethernet settings for the system.

Flash Settings

This submenu 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 submemu is for Zynq only. It allows users to select which SD controller is the system primary SD.

Timer Settings

This submenu 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 submenu is for MicroBlaze only. It allows users to select which GPIO is the system reset GPIO.

TIP: *MicroBlaze system uses GPIO as soft reset input. If reset GPIO is selected, you can reboot from Linux.*

Advanced bootable images storage Settings

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

If this submenu is disabled, PetaLinux will use the flash partition table in the `"Flash Settings --->"` submenu to define the location of the bootable images.

bootable image / u-boot environment partition	**Default flash partition**	**Description**
boot image	boot	• BOOT.BIN for Zynq • Relocatable U-Boot BIN file u-boot-s.bin for MicroBlaze
u-boot env partition	bootenv	U-Boot environment variable partition. In this release, PetaLinux u-boot auto configuration supports the U-Boot env on Flash only.
kernel image	kernel	Kernel image image.ub which is a FIT image.
dtb image	dtb	If `"Advanced bootable images storage Settings"` is disabled and a `dtb` partition is found in the flash partition table settings, PetaLinux auto configured u-boot will get the DTB from the partition table, otherwise, it will assume a DTB is contained in the kernel image.

Kernel Bootargs Submenu

This submenu allows users to let PetaLinux auto generate the kernel boot commandline settings in DTS, or pass PetaLinux user defined kernel boot commandline settings.

U-boot Configuration Submenu

This submenu allows user to select to use PetaLinux u-boot auto configuration or use a u-boot board configuration target.

Image Packaging Configuration Submenu

This submenu allows user to set the following image packaging configurations:

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

TIP: *When using `petalinux-build` to build a project, it always generates a FIT image as the kernel image.*

Firmware Version Configuration Submenu

This submenu allows user 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

Appendix B: Generate First Stage Bootloader Within Project

This is *OPTIONAL.*
By default, the top level system settings are set to generate the first stage bootloader.
You can configure the project to build first stage bootloader as follows:

1. *Launch top level system settings configuration menu and configure:*

```
$ petalinux-config
```

   (a) *Click into "`linux Components Selection --->`" submenu.*
   (b) *Select "`First Stage Bootloader`" option*

```
[*] First Stage Bootloader
```

   (c) *Exit the menu and save the change.*

   This operation will generate the FSBL (First Stage Bootloader) source into `components/bootloader/` inside your PetaLinux project root directory if it doesn't already exist. For Zynq, it will be:

```
components/bootloader/zynq_fsbl
```

   For MicroBlaze, it will be:

```
components/bootloader/fs-boot
```

   We only support FSBL in the local project directory in this release.

2. Launch `petlainux-build` to build the FSBL:

   Build the FSBL when building the project:

```
$ petalinux-build
```

   Build the FSBL only:

```
$ petalinux-build -c bootloader
```

   The bootloader ELF file will be installed as `zynq_fsbl.elf` for Zynq and `fs-boot.elf` for MicroBlaze in `images/linux` inside the project root directory.

Additional Resources

References

- *PetaLinux Tools Application Development Guide (UG981)*
- *PetaLinux Tools Board Bringup Guide (UG980)*
- *PetaLinux Tools Firmware Upgrade Guide (UG983)*
- *PetaLinux Tools Getting Started Guide (UG977)*
- *PetaLinux Tools Installation Guide (UG976)*
- *PetaLinux Tools QEMU System Emulation Guide (UG982)*

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