PetaLinux SDK User Guide

Board Bringup Guide

UG980 (v2013.04) April 22, 2013

XILINX®

Revision History

Date	Version	Notes
2012-08-03	3.1	Updated for PetaLinux SDK 3.1 release
2012-09-03	12.9	Updated for PetaLinux SDK 12.9 release
2012-12-17	2012.12	Updated for PetaLinux SDK 2012.12 release
2013-04-22	2013.04	Updated for PetaLinux SDK 2013.04 release

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 SDK 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 basic understanding of the Xilinx Platform Studio tools, including the Base System Builder wizard. It also assumes that you are familiar with the basics of working with PetaLinux SDK.

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

Overview

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

1. Use the Xilinx Platform Studio (XPS) to create and/or configure an EDK hardware project ready for PetaLinux.
2. Create a new PetaLinux SDK software platform.
3. Propagate the hardware platform configuration settings into the new software platform and complete any further software platform configuration steps.

Hardware Platform

Hardware platforms can be created from a number of sources, such as an existing XPS project, or by using Xilinx Base System Builder wizard. Each of these is illustrated below.

Regardless of how the core hardware system is created, there are a small number of hardware IP and software platform configuration changes required to make the hardware system "PetaLinux-ready". These are also illustrated below.

Create a Hardware Project with Xilinx Base System Builder

Xilinx Base System Builder supports multiple architectures. Zynq and MicroBlaze (little-endian AXI) are supported in PetaLinux SDK.
The following sections illustrate how to create a PetaLinux-ready hardware project for each of these system architectures.

RECOMMENDED: *It is recommended that you create your custom hardware projects in the*

```
$PETALINUX/hardware/user-platforms
```

subdirectory. By using a standard location it will be easier to find your hardware projects, if you need to make backups, or copy them to a new PetaLinux installation. "$PETALINUX" refers to the directory path where you installed PetaLinux SDK. For example, if PetaLinux is installed in "/home/user/petalinux directory", the "$PETALINUX" refers to "/home/user/petalinux".

Zynq

1. Start Xilinx Platform Studio.
2. Click on **Create New Blank Project** to start a empty project.
3. Click **Browse** button to set the Project file to your custom hardware project location

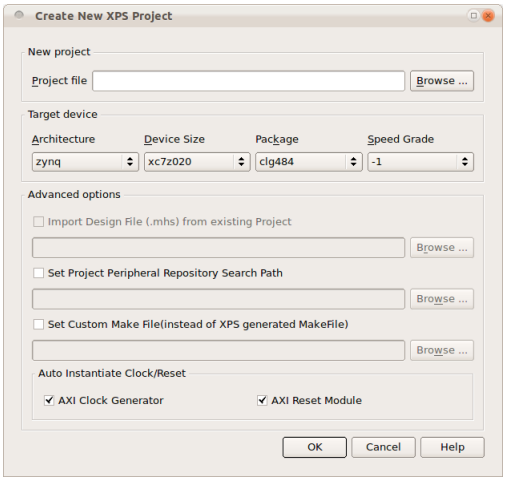

Figure 1: Create New XPS Project

4. Selection the target Zynq device, and click **Ok** to continue.
5. Click **Ok** if the How to Add IPs to Design window pops up
6. If you have a system configuration available import this configuration using the **Import** button located at the top of the System Assembly View tab. (e.g. for the ZC702 import `"xc7z020.xml"`) If you do not have a configuration you will need to manually configure your system.

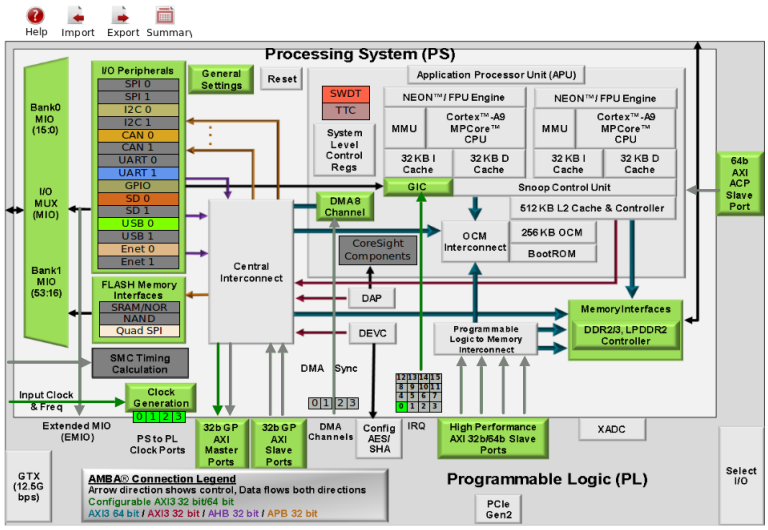

Figure 2: System Diagram

7. Recommended minimum configuration is shown below:
   - Memory (32 MB or more)
   - UART (Required)
   - Triple Timer Counter (Required)

- Non-volatile memory (optional) e.g. Parallel Flash (NAND/NOR), QSPI Flash, SD/MMC
- Ethernet (optional, required for network access)

8. Click **Yes** to set the MIO configuration in the Platform Studio window
9. The hardware project is opened in the XPS GUI.

Your XPS project is now ready to run PetaLinux

MicroBlaze AXI

1. Start Xilinx Platform Studio.
2. Create a new hardware project using the BSB wizard.
3. Select **AXI system** and press **Ok** button.
4. Move ahead through the BSB wizard to configure the hardware, until you get to the Peripheral Configuration dialog.
5. Add an `axi_timer` to the project and set the timer to use interrupt as shown below:

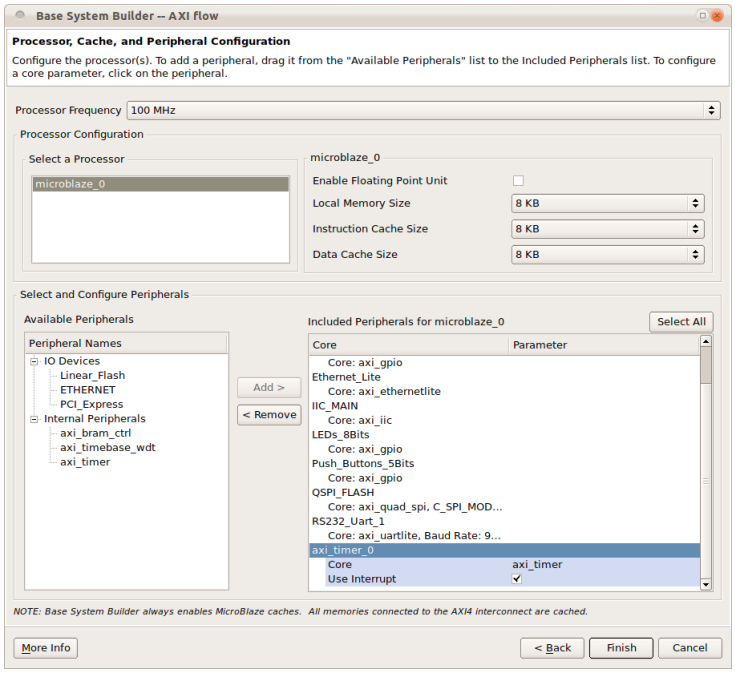

Figure 3: Peripheral Configuration

The recommended minimum set of IP is shown below:

- External memory controller with at least 8MB of memory (16MB or more recommended)
- Timer (required)
- Serial console (required)
- Non-volatile memory (optional), such as Linear Flash or SPI Flash
- Ethernet (essential for network access)

Any other IP cores are optional.

6. Choose your preference of `ethernetlite` or `axi_ethernet` as the ethernet controller.

IMPORTANT: *If using the `axi_ethernet`, DMA mode must be used. Regardless of your choice of ethernet controller, ensure that the **Use Interrupt** checkbox is set.*

7. Choose your preference of serial port console, either uartlite or UART16550.

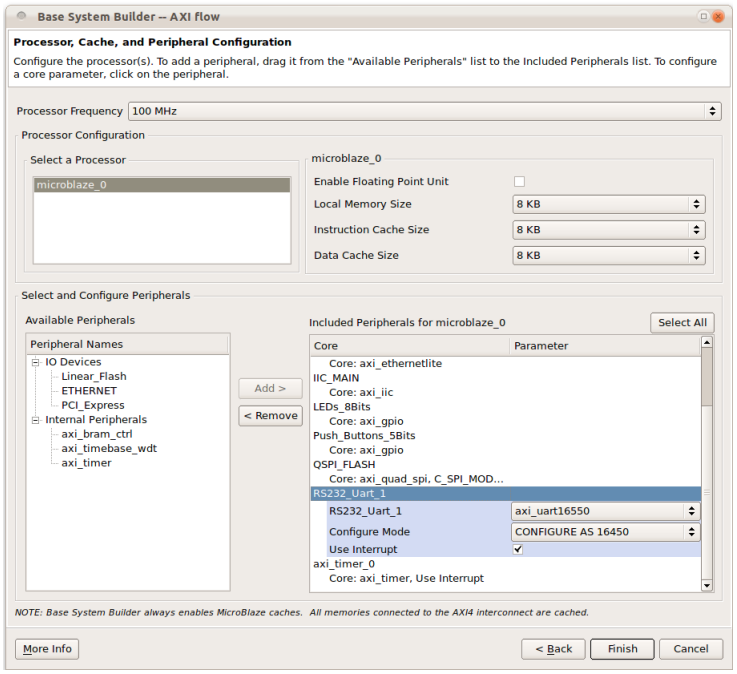

Figure 4: Peripheral Configuration

IMPORTANT: *If you use `axi_uartlite` as the UART IP Core, a baud rate of 115200 is recommended. Be sure to select the **Use Interrupt** check box.*

8. After configuring the peripherals, the next step is to configure the CPU cache options.

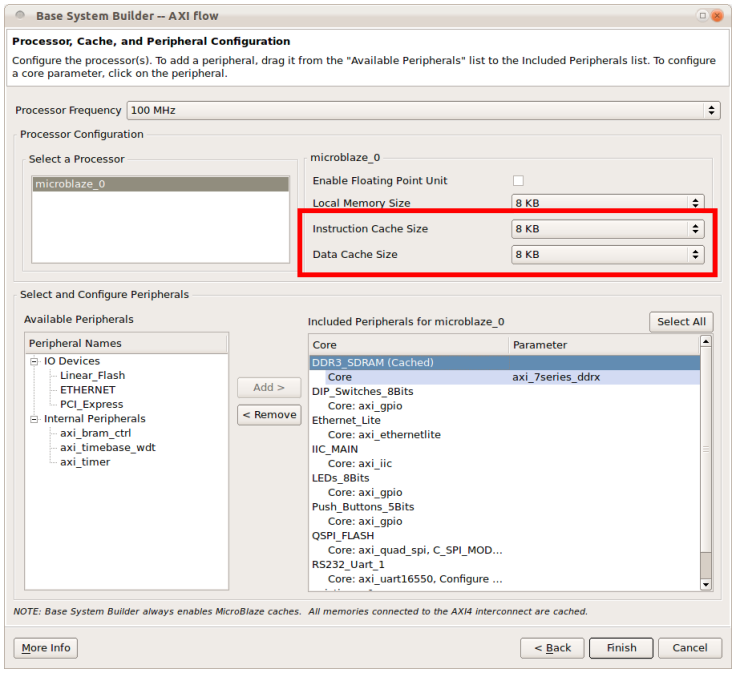

Figure 5: Cache Configuration

RECOMMENDED: *It is recommended that instruction and data caches always be used, and configured to the full range of primary system memory (DDR/SDRAM).*

The cache size and the cache memory may be different from the above image. It depends on your system requirement.

9. After the BSB wizard completes, the hardware project will be opened in the XPS GUI.
10. For MicroBlaze targets, PetaLinux requires that the CPU be configured with the MMU enabled. It is also recommended that at least the barrel shifter feature is enabled, for improved performance. Other CPU features may be enabled as required.
    (a) Double-click on the MicroBlaze instance in System Assembly View, to open the MicroBlaze Configuration Wizard.
    (b) Select the **Linux with MMU** or **Low-end Linux with MMU** configuration
    (c) If you wish to make further CPU customisations, either click through the wizard with the **Back** and **Next** buttons, or choose **Advanced** to get fine control of the CPU configuration.
    (d) Press **Ok** when you have finished.

TIP: *To configure MicroBlaze manually:*

For experienced XPS users, you can edit the "system.mhs" file to configure MicroBlaze to enable barrel shifter and support MMU:

```
PARAMETER C_USE_BARREL = 1
PARAMETER C_AREA_OPTIMIZED = 0
PARAMETER C_USE_MMU = 3
PARAMETER C_MMU_ZONES = 2
PARAMETER C_MMU_DTLB_SIZE = 4
PARAMETER C_MMU_ITLB_SIZE = 2
PARAMETER C_MMU_TLB_ACCESS = 3
```

Your XPS project is now ready to run PetaLinux.

Customise an Existing Hardware Platform for PetaLinux

If you have already had a hardware project, you may wish to copy the entire project directory into the "$PETALINUX/hardware/user-platforms" directory.

Zynq

Follow the Zynq Workflow section of Configuring Software Setting of the Hardware Project and FS-Boot section for detailed steps on how to create a PetaLinux BSP for the platform.

MicroBlaze AXI

Open the project in XPS, and complete the following checklist:

1. Ensure you have axi_timer configured in your system as follows:
   - Interrupt is enabled, and connected
   - Dual channel timer is selected
2. Ensure you have the UART core (axi_uartlite or axi_uart) configured to use interrupts, and that the interrupt signal is connected. If you are using axi_uartlite, please configure its baud rate to 115200.
3. If you have Ethernet core (either axi_ethernetlite or axi_ethernet) configured, please configure it to use interrupt and enable ipname for axi_ethernet.
4. It is recommended to configure the MicroBlaze to use both Instruction Cache, Data Cache and Barrel shifter to improve performance.. Ensure that the cacheable address range on the CPU matches the physical address range of the primary system memory.
5. Enable the MicroBlaze to use the MMU, with the correct configuration. See step 8 in Section Create a Hardware Project with Xilinx Base System Builder above for details.

Configuring Software Setting of the Hardware Project and FS-Boot

After designing the hardware system, it is necessary to configure a Linux BSP for that platform. This is required to automate the Linux board bringup.

Zynq

TIP: *For Zynq platforms, the Xilinx-provided FSBL is used instead to load the second stage boot-loader.*

1. Select **Export Hardware design to SDK** from the Project menu in the XPS GUI.
2. Tick **Include bitstream and BMM file** box, then click **Export & Launch SDK** button. (Please note, if **Include bitstream and BMM file** is ticked, XPS will generate the `"system.bit"` and `"system_bd.bmm"` bitstream files before launching SDK. This can take a while)
3. In the Workspace Launcher set the workspace path to the current hardware project directory, for example `"$PETALINUX/hardware/user-platforms/<hardware project>/workspace"`. Substitute the correct value of the `"$PETALINUX"` installation directory as appropriate, or use the GUI to browse to the correct location. Please create the workspace directory if it does not exist. Then click **Ok** to start XSDK.

RECOMMENDED: *It is recommended to use one Eclipse workspace per hardware project, since each hardware project will have it own BSP configuration. (You can still use global workspace for your software development. However, this is not a supported workflow for PetaLinux SDK.)*

4. Select **Switch workspace** in **File** menu in XSDK, then select **Other**. Ensure the workspace path is point to the `"workspace"` directory inside your hardware project directory.
5. Add the PetaLinux repository to the list of available BSPs in XSDK:
   (a) Click **Xilinx Tools** menu tab and select **Repositories**
   (b) Select **Repositories** on the left menu under Xilinx SDK
   (c) Click **New** (for local repositories box) on the right hand top corner.
   (d) Browse to PetaLinux `"edk_user_repository"` directory - `"$PETALINUX/hardware/edk_user_repository"` and click **Ok**
   (e) Click **Ok** to finish adding local repository.
6. Create Zynq FSBL project with XSDK standalone BSP:
   (a) Click **File** menu tab select **New** and select **Project**
   (b) Select **Xilinx Application Project** wizard to create a Zynq FSBL project with XSDK standalone BSP.
   (c) Click **Next**.
   (d) Ensure the "Zynq FSBL" template is selection and click **Finish**.

7. Create PetaLinux BSP and Configure PetaLinux BSP for U-boot and Linux operation system:
   (a) Click **File** menu tab select **New** and select **Project**
   (b) Select **Xilinx Board Support Package** wizard
   (c) Select **petalinux** as the **Board Support Package OS**.
   (d) Follow the wizard to complete creating the PetaLinux BSP project
   (e) When you click **Finish** in the **Xilinx Board Support Package** wizard, a Board Support Package Settings window will pop up. If it did not pop up, follows this
      i. Right click on the "petalinux_bsp_0" project in Project explorer
      ii. Select **Board Support Package Settings**.
   (f) Select **petalinux** in the Board Support Package Settings window.
      i. Set stdout and stdin to ps7_uart_1 or ps7_uart_0 (depending on your configuration)
      ii. Set main_memory to ps7_ddr_0

      Additionally configure if available:
      i. flash_memory
      ii. sdio
      iii. ethernet
   (g) Click **Ok** to accept the settings

MicroBlaze

1. Select **Export Hardware design to SDK** from the **Project** menu in the XPS GUI.
2. Tick **Include bitstream and BMM file** box, then click **Export & Launch SDK** button. (Please note, if **Include bitstream and BMM file** is ticked, XPS will generate the "system.bit" and "system_bd.bmm" bitstream files before launch SDK. This can take a while)
3. In the Workspace Launcher set the workspace path to the current hardware project directory, for example "$PETALINUX/hardware/user-platforms/<hardware project>/workspace". Substitute the correct value of the "$PETALINUX" installation directory as appropriate, or use the GUI to browse to the correct location. Please create the workspace directory if it does not exist. Then click **Ok** to start XSDK.

RECOMMENDED: *It is recommended to use one Eclipse workspace per hardware project, since each hardware project will have it own BSP configuration. (You can still use global workspace for your software development. However, this is not a supported workflow for PetaLinux SDK.)*

4. Select **Switch workspace** in **File** menu in XSDK, then select **Other**. Ensure the workspace path is point to the "workspace" directory inside your hardware project directory.
5. Add the PetaLinux repository to XSDK to the list of available BSPs:
   (a) Click **Xilinx Tools** menu tab select **Repository**
   (b) Select **Repositories** on the left menu
   (c) Click **New** (for local repositories box) on the right hand top corner.

(d) Browse to PetaLinux "edk_user_repository" directory - "$PETALINUX/hardware/edk_user_repository" and click **Ok**

(e) Click **Ok** to finish adding local repository.

6. Create fs-boot project and PetaLinux BSP:

   (a) Click **File** menu tab select **New** and select **Project**

   (b) Select **Xilinx Application Project** wizard

   (c) Follow the wizard to create FS-boot project with PetaLinux BSP

      i. Specify **Project name**, e.g. **fs-boot**

      ii. Select **petalinux** as **OS Platform**

      iii. Select **Create New** Board Support Package

      iv. Select **FS-BOOT** from **Available Templates**

7. Configure PetaLinux BSP for U-boot and Linux operation system:

   (a) Right click on the PetaLinux BSP project created from the previous step project in Project explorer

   (b) Select **Board Support Package Settings**.

   (c) Select **petalinux** in the Board Support Package Settings window

   (d) These settings are automatically selected, please review to ensure they are correct.

   (e) Click **Ok** to accept the settings

WARNING: *At this point, there may be an error message about fs-boot compilation error due to memory overlapped for MicroBlaze system with 4Kbyte of BRAM. This is normal and it will be addressed in the next step.*

8. Configure fs-boot settings. The fs-boot bootloader is used on system boot to move the U-Boot image from the U-boot partition in Flash to main memory and then run U-Boot from main memory. To do this, the start address of U-boot partition in Flash memory must be set:

   (a) Right click on the **fs-boot** project in Project explorer and select **C/C++ Build settings**

   (b) Select the **C/C++ Build** configuration and select **Settings**

   (c) Select on the **Tool Settings** tab

   (d) Set configuration profile to **[All configurations]**

   (e) Select Symbols

      - For MicroBlaze target, click **Symbols** under MicroBlaze gcc compiler

   (f) In the Defined symbols (-D) box click the **Add** button (the small icon with a green plus symbol on it)

   (g) In the **Enter Value** window, enter CONFIG_FS_BOOT_OFFSET=<value>, replace <value> to appropriate offset. This must match the offset from the start of flash to the boot partition, as set in the PetaLinux system configuration menu, under Flash Partition Table.

TIP: *If the location of the boot offset is not yet determined, specify 0 and return to update once the location is determined.*

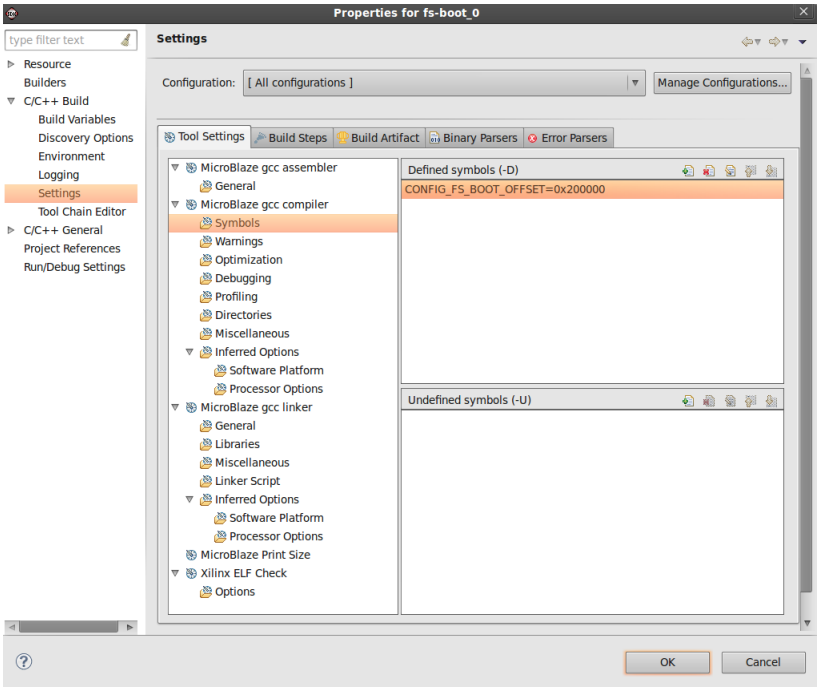

Figure 6: FS-Boot Configuration

(h) For MicroBlaze targets **only**. As fs-boot is a simple bootloader, the use of interrupts is not required and as such can be disabled. The following linker flag can be set to disable the use of a vector table:

- Click **Miscellaneous** under MicroBlaze gcc linker
- Append `-Zxl-mode-novectors` flag to **Linker Flags** field.

(i) Click **Apply** button to apply the settings

(j) Click **Ok** button to finish Compiler flag configuration

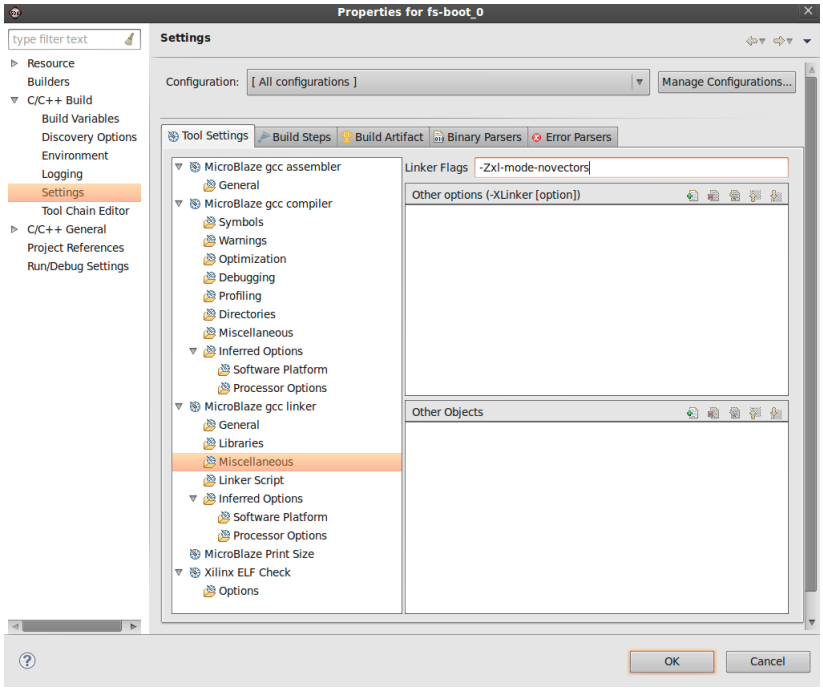

Figure 7: FS-Boot Configuration

9. Set to fs-boot to **Release** profile

   (a) Right click on the **fs-boot_0** project in Project explorer

   (b) Select **Build Configurations**

   (c) Select **Set Active** and select **Release**

10. Build the fs-boot project. XSDK also runs PetaLinux BSP generation tools. These tools automatically generate the software platform configuration files required in the subsequent board bringup stages. These files are described in more detail later in this document.

    If the Xilinx Eclipse/SDK did not automatically build the `fs-boot_0` software project, right click on the **fs-boot_0** project in Project explorer and select **Build Project** to build the project.

Finalising the FPGA Bitstream

At this stage, the hardware platform is ready and properly configured and the "fs-boot" executable binary is ready, but not initialised to BRAM. This section documents how to initialise the "fs-boot" to BRAM and create the bitstream.

TIP: *This step is not required for Zynq.*

1. In the **Project** tab of Project information area in XPS, expand the **ELF files** option.
2. Double click on the **microblaze_0** item
3. A **Select ELF file** window will pop up and allows you to select which ELF file to initialize to BRAM.
4. Browse to the "fs-boot" executable created by XSDK. You should able to find the "fs-boot_0.elf" in "<hardware project>/workspace/fs-boot_0/Release/" folder. Then click **Open**.
5. Click **Ok** to close Select Elf file window.
6. Select **Device Configuration** menu and select **Update Bitstream** or **Download Bitstream** to create the "download.bit" bitstream which has "fs-boot" initialised to BRAM.

For advanced users, you may also commence the BRAM initialisation using the normal XPS commands from the command line. The FPGA bitstream is now ready for download.

Software Platform

Create New Platform

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

```
$ petalinux-new-platform -c <cpu-arch> -v <VENDOR> -p <PLATFORM>
```

The parameters are as follows:

- `-c <cpu type>` - As of PetaLinux v2013.04, supported CPU types are arm and microblaze
- `-v <VENDOR>` - This is the vendor name - often you will use your company's name
- `-p <PLATFORM>` - The name of the platform or product you are building.

This command will create a new PetaLinux software platform from a default template. In later steps we will customise these settings to match the hardware project created previously.

The configuration files of the software platform are created in the directory `"$PETALINUX/software/petalinux-dist/vendors/<VENDOR>/<PLATFORM>"`.

Here is a list and description of the configuration files:

Configuration File	Description
`"config.arch, config.device"`	PetaLinux architecture configuration files. Do not modify.
`"config.linux-2.6.x"`	Default Linux kernel configuration for this platform.
`"config.vendor"`	User application and system configuration settings.
`"<VENDOR>-<PLATFORM>.dts"`	DTS (Device Tree Source) file describing the hardware platform. Used to configure the Linux kernel at boot up.
`"config.mk, xparameters.h"`	U-boot platform configuration files.

The new platform is automatically selected when running the `petalinux-new-platform` command, so there is no need to launch the top level configuration menu, and select the new platform.

Configure Software Platform

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

1. Navigate to appropriate directory before using `"petalinux-copy-autoconfig"` command

- For the PetaLinux BSP directory is for example: "<hardware project directory>/workspace/petalinux_bsp_0/".

2. Use the petalinux-copy-autoconfig command:

```
$ petalinux-copy-autoconfig
INFO: Using MSS file ./system.mss and XML file ./../hw_platform_0/system.xml
INFO: Attempting vendor/platform auto-detect
INFO: Auto-detected platform "Xilinx/Xilinx-SP605-AXI-full-14.5"
INFO: Merging platform settings into kernel configuration
```

This could take a while. This is because it merges the existing platform configuration to Kernel configuration and enables the appropriated drivers in the Kernel configuration.

This tool generates the hardware configuration files, if required and copies the configuration files to the right location in PetaLinux.

- The DTS file will be placed in

```
$PETALINUX/software/petalinux-dist/linux-2.6.x/arch/
    <arm|microblaze>/boot/dts/<VENDOR>-<PLATFORM>.dts
```

- The "xparameters.h" and "config.mk" files are placed in

```
$PETALINUX/software/petalinux-dist/u-boot/petalogix/
    <arm|microblaze>-auto
```

NOTE: *<...> represents configuration specific fields, replace with the correct value for your platform.*

---

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

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

4. Launch the user applications and system settings configuration menu and configure it to meet your requirements:

```
$ petalinux-config-apps
```

TIP: *For Zynq:*

- *The boot device is QSPI by default. If you are using a different boot device, you can select different System boot device from System Settings submenu.*
- *If you are not using any boot device, you will need to select other as System boot device from System Settings submenu.*

---

5. Once you are satisfied with the configurations and they have been tested, it is recommended to update your default configurations, to avoid losing your configuration when switching from one software platform to another.

```
$ petalinux-platform-config --update
```

Customise U-boot Configuration

Usually, you don't need to configure u-boot yourself since PetaLinux configures u-boot to cover most common cases during u-boot compilation. However, if the default PetaLinux u-boot configuration doesn't fit your requirement, you can add your own u-boot configuration to:

```
$PETALINUX/software/petalinux-dist/u-boot/include/configs/petalinux/<vendor>-<platform>.h.template
```

You can use this file to define your u-boot configuration macros. PetaLinux has predefined some u-boot configuration macros in:

```
$PETALINUX/software/petalinux-dist/u-boot/include/configs/petalinux-auto-board.h.template
```

During u-boot compilation, it will include the PetaLinux generic u-boot configuration macros first, and then the customer one. And thus, if you want to redefine some macro which is already defined in the "petalinux-auto-board.h.template", you can undefine it in your "<vendor>-<platform>.h.template" and then redefine it.

"petalinux-platform-config --update" will save your customised "<vendor>-<platform>.h.template" to your default software platform configuration

```
$PETALINUX/software/petalinux-dist/vendors/<vendor>/<platform>/
```

Build PetaLinux

Finally, it is time to build your new platform.

1. Build the hardware bitstream with Xilinx EDK tool, if you have not done so already.
2. Run make in the "petalinux-dist" directory to build the PetaLinux system image:

   ```
   $ cd $PETALINUX/software/petalinux-dist
   $ make
   ```

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

   ```
   [INFO ] Building ucfront tool
   [INFO ] Building kernel
   [00:00] /
   ```

   And the compilation log stores in "build.log" file in the "petalinux-dist/" directory.

Please refer to the primary PetaLinux SDK documentation for details on the PetaLinux SDK platform configuration and build procedures.

Generate BOOT image for Zynq

This section is for Zynq only. MicroBlaze targets can skip this section.

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

```
$ petalinux-gen-zynq-boot -b <FSBL image> -f <FPGA bitstream> -u <uboot image>
    -o <output file name>
```

The parameters are as follows:

- `-b <FSBL image>` - Path to FSBL image location
- `-f <FPGA image>` - Path to FPGA bitstream location
- `-u <uboot image>` - Path to uboot image (default is `"images/u-boot.elf"`)
- `-o <output file>` - Output image name (default is `"images/BOOT.BIN"`)

Test PetaLinux on the Board

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

Direct Kernel Boot via JTAG

1. Program the FPGA using the Xilinx JTAG programming tools

IMPORTANT: *For Zynq, after programing the FPGA with the bitstream, you will need launch XMD and run XMD command `init_user` to enable level register:*

```
XMD% init_user
```

2. This step is for Zynq only, download the u-boot to the board and boot it:

```
$ cd $PETALINUX/software/petalinux-dist
$ petalinux-jtag-boot -u
```

It will download `"images/u-boot.elf"` to the board and boot it. This is required for Zynq because u-boot remaps OCM to higher address so that the lower 1MB of DDR can be accessed.

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

```
$ cd $PETALINUX/software/petalinux-dist
$ petalinux-jtag-boot -i images/image.elf
```

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

Indirect Kernel Boot via U-Boot

If you have configured ethernet in your hardware and connected the board to the network, you can use u-boot to boot PetaLinux:

1. First, bootstrap the system by downloading u-boot via JTAG:

```
$ cd $PETALINUX/software/petalinux-dist
$ petalinux-jtag-boot -i images/u-boot.elf
```

2. After u-boot boots, in the u-boot console, check whether the TFTP server IP is set to the IP of the host which builds the PetaLinux:

```
u-boot> print serverip
```

   If it is not, set the server IP to the host IP:

```
u-boot> set serverip <HOST IP>
```

3. Confirm the IP address of the board, and set if required:

```
u-boot> print ipaddr
u-boot> set ipaddr <w.x.y.z>
```

4. Run netboot to download the PetaLinux image with TFTP and boot it:

```
u-boot> run netboot
```

Troubleshooting

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

Problem/Error Message	Description and Solution

Cannot see an IP instance name listed in the **OS and Lib configuration** selection list. e.g. Cannot see the IP instance `my_flash` in the `flash_memory` selection list.	***Problem Description:***
	This problem is usually because the type of the IP instance is not a standard Xilinx IP core but a customized IP Core.
	
	Solution:
	• Set that configuration item to `none`.
	• Manually edit the DTS file:
	`"petalinux-dist/linux-2.6.x/arch/<arm|microblaze>/boot/dts/<VENDOR>-<PLATFORM>.dts"`
	to configure the node of the IP CORE as you require before building PetaLinux.
	You can find the DTS introduction file here:
	`"petalinux-dist/linux-2.6.x/Documentation/powerpc/booting-without-of.txt"`
	for Linux kernel.
	• Manually edit the
	`"petalinux-dist/u-boot/petalogix/<arm|microblaze>-auto/xparameters.h"`
	for u-boot.
	• Usually since it is a customised IP CORE, you may also need to write the Linux and u-boot drivers for it. The of-platform device framework is recommended for the IP CORE Linux driver to use the device tree for hardware configuration inquiry.
	Note: `<...>` represents configuration specific fields, replace with the correct value for your platform.

Problem/Error Message	Description and Solution
Cannot see any output when trying to boot u-boot with UART16550.	***Problem Description:*** This problem is usually because the serial communication terminal application is set with the wrong baud rate or the clock frequency of UART16550 is missing in the `"xparameters.h"` in u-boot. ***Solution:*** • Check whether your terminal application baud rate is correct. • Check whether `#define XILINX_UART16550_CLOCK_HZ <FREQ>` is in `"petalinux-dist/u-boot/petalogix/<arm|microblaze>-auto/xparameters.h"`. If not, run `petalinux-copy-autoconfig` again from the hardware directory. • Check again for: `"petalinux-dist/u-boot/petalogix/<arm|microblaze>-auto/xparameters.h"` If it is still not there, it could be because of the Xilinx tool version is too old. The workaround to this problem is to add the UART 16550 clock frequency macro to the u-boot `"xparameters.h"`. The UART 16550 clock frequency is usually the clock frequency of the bus it connects to which is usually the system clock frequency. **Note:** `<...>` represents configuration specific fields, replace with the correct value for your platform.

Problem/Error Message	Description and Solution
Cannot see any output when trying to boot Linux with UART16550.	***Problem Description:*** This problem is usually because the serial communication terminal application is set with the wrong baud rate or the clock frequency of UART16550 is missing in the `"xparameters.h"` in DTS. ***Solution:*** • Check whether your terminal application baud rate is correct. • Check whether `clock-frequency = <FREQ>;` of the UART 16550 node is in: `"petalinux-dist/linux-2.6.x/arch/<arm|microblaze>/boot/dts/<VENDOR>-<PLATFORM>.dts"` If not, run `petalinux-copy-autoconfig` again from the hardware project directory. • Check the DTS file again. If it is still not there, it could be because of the Xilinx tool version is too old. The workaround to this problem is to add the UART 16550 clock frequency macro to the DTS file. The UART 16550 clock frequency is usually the clock frequency of the bus it connects to which is usually the system clock frequency. **Note:** `<...>` represents configuration specific fields, replace with the correct value for your platform.

Additional Resources

References

- PetaLinux SDK Application Development Guide (UG981)
- PetaLinux SDK Board Bringup Guide (UG980)
- PetaLinux SDK Eclipse Plugin Guide (UG979)
- PetaLinux SDK Firmware Upgrade Guide (UG983)
- PetaLinux SDK Getting Started Guide (UG977)
- PetaLinux SDK Installation Guide (UG976)
- PetaLinux SDK QEMU System Simulation Guide (UG982)