



Board Bring Up with PetaLinux SDK

v2.1 March 21, 2011



Table of Contents

Table of Contents	2
About This Guide	3
Related Documents	3
Overview	3
Create Hardware Project	4
Create a Hardware Project with Xilinx Base System Builder.....	4
<i>MicroBlaze AXI target (Xilinx 13.1 and later)</i>	4
<i>MicroBlaze PLB target</i>	9
<i>PowerPC 440 target</i>	14
Customise an Existing Hardware Platform for PetaLinux.....	18
<i>For MicroBlaze AXI target</i>	18
<i>For MicroBlaze PLB target</i>	18
<i>For PowerPC440 target</i>	19
Configuring Software Settings of the Hardware Project and FS-Boot	20
<i>For XSDK workflow (Xilinx 13.1 and later)</i>	20
<i>For XPS workflow (Xilinx 12.4 and earlier)</i>	22
Finalising	24
<i>XSDK workflow</i>	24
<i>XPS workflow</i>	25
Create new PetaLinux SDK Software Platform	25
Configure Software Platform	26
Build PetaLinux	27
Test PetaLinux on the Board	28
Direct kernel boot via JTAG.....	28
Indirect kernel boot via u-boot.....	28
Troubleshooting	30

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 MicroBlaze or PowerPC 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 the basics of working with PetaLinux SDK.

We strongly recommend 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.

As of version 1.3, PetaLinux SDK supports MicroBlaze and PowerPC440 system architectures. The PetaLinux development and boot flow for each architecture is identical.

Related Documents

The following other documents exist to help you to make the most of your PetaLinux experience:

- PetaLinux SDK Installation Guide
- Getting Started with PetaLinux SDK
- Guide to QEMU System Simulation in PetaLinux SDK
- Guide to Create and Debug User Applications
- PetaLinux Migration Guide

You can find PetaLinux SDK documentation online at

- http://www.petalogix.com/resources/documentation/petalinux_sdk

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

Create Hardware Project

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

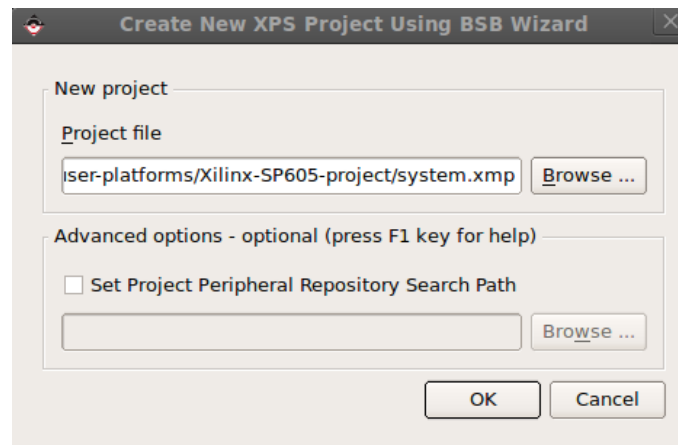
Regardless of how the core hardware system is created, there is a small number of hardware IP and software platform configuration changes required to make the hardware system “PetaLinux-ready”. These are also illustrated below.

Create a Hardware Project with Xilinx Base System Builder

Xilinx Base System Builder supports three different processor architectures, MicroBlaze AXI, MicroBlaze PLB and PPC. The following sections illustrate how to create a PetaLinux-ready hardware project for each of these system architectures.

MicroBlaze AXI target (Xilinx 13.1 and later)

1. Run the Xilinx BSB wizard to create a hardware project.



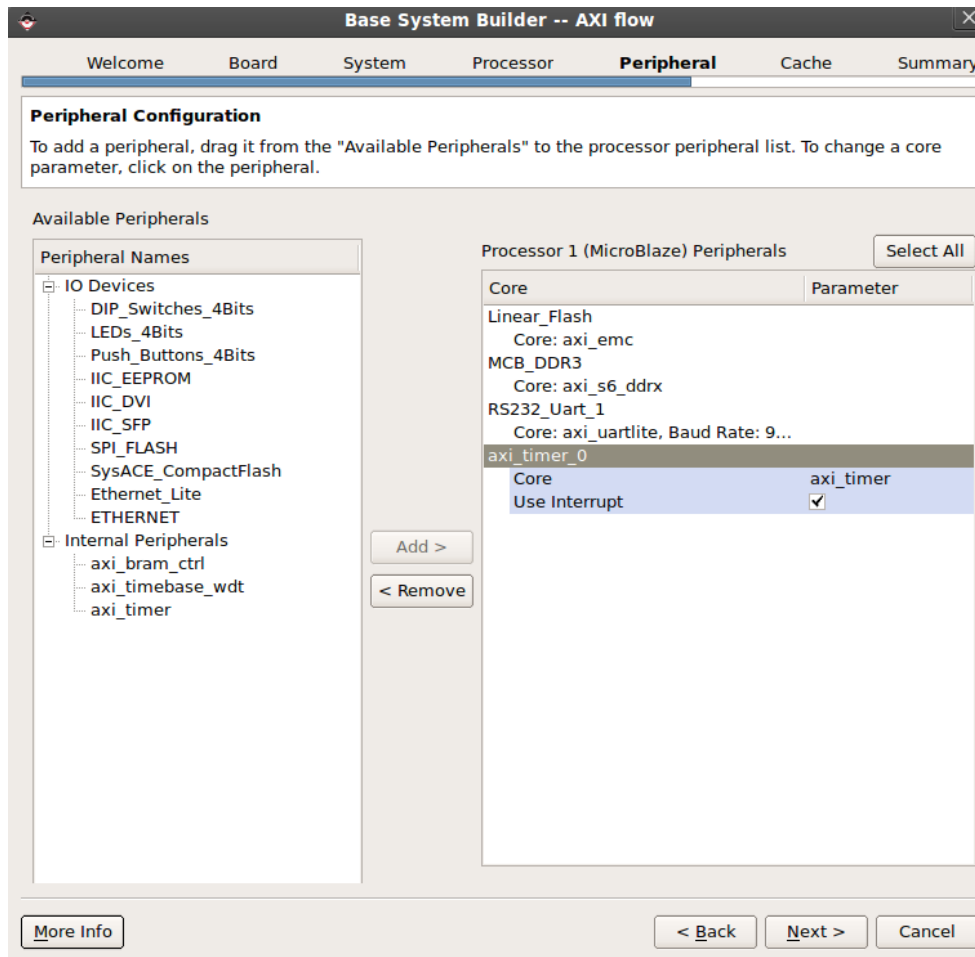
Note: We recommend creating your custom hardware projects in the

`$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.

2. Select “AXI system” and press “OK” button.
3. Move ahead with through BSB wizard to configure the hardware, until you get to the “Peripheral Configuration” dialog.
4. Add an “axi_timer” to the project and set the timer to use interrupt as shown below:



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.

5. Choose your preference of “ethernetlite” or “axi_ethernet” as the ethernet controller.

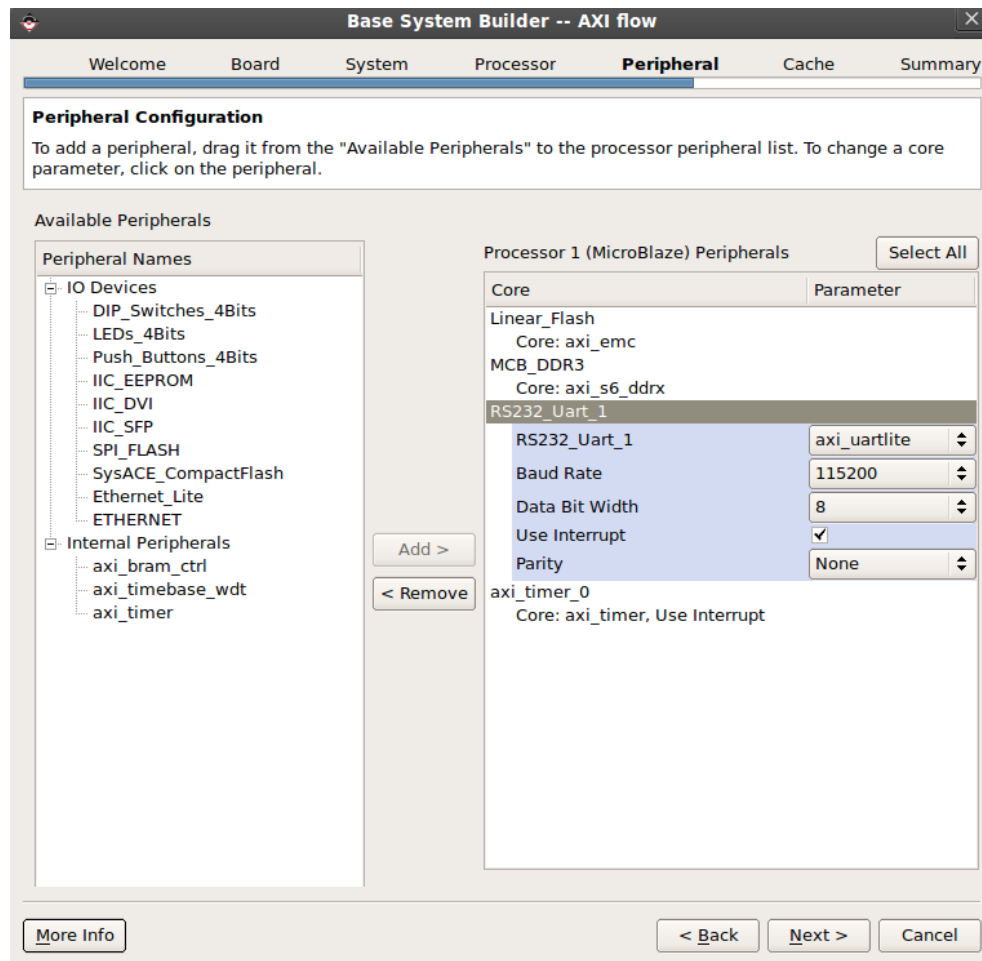
Notes

- 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.

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

Notes

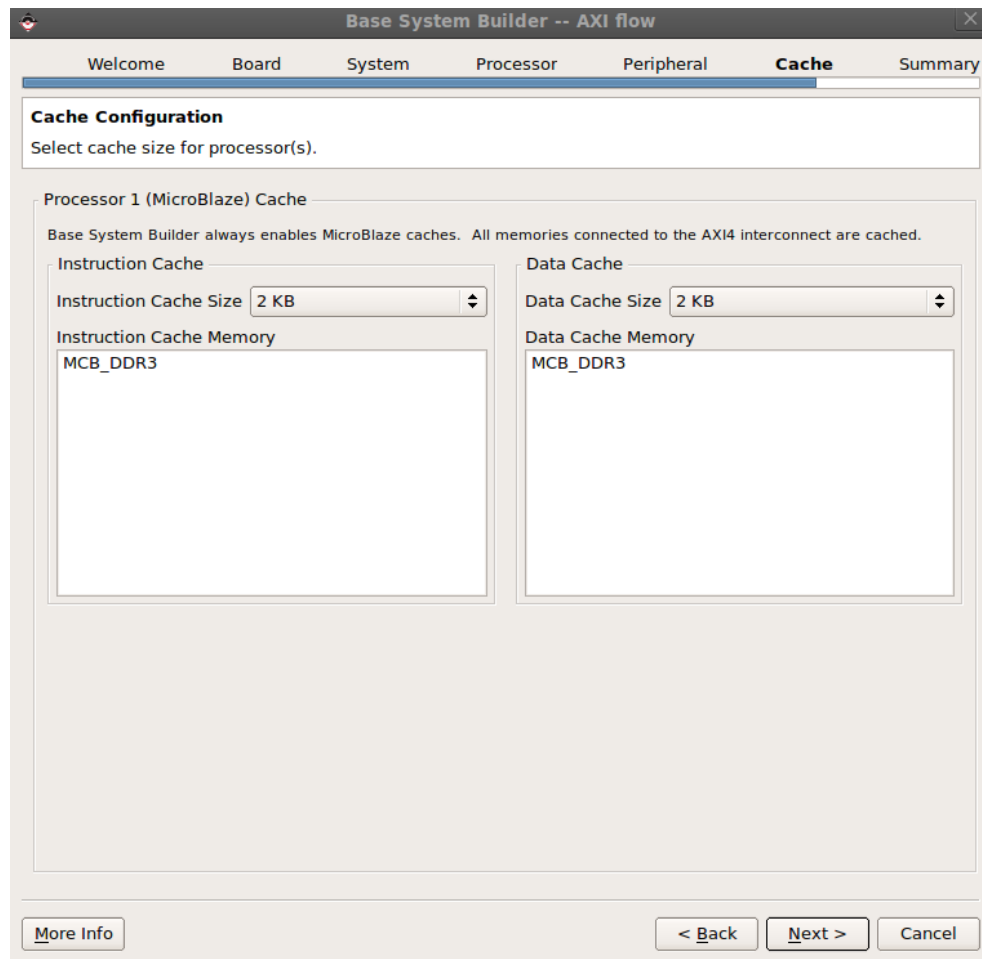
- If you use “axi_uartlite” as the UART IP Core, we recommend a baudrate of 115200:
- Be sure to select the “Use Interrupt” check box



7. After configuring the peripherals, the next step is to configure the CPU cache options, using the “Cache” window of the BSB wizard.

Note:

- PetaLogix recommends that instruction and data caches always be used, and configured to the full range of primary system memory (DDR/SDRAM) as shown below.



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

8. After the BSB wizard completes, the hardware project will be opened in the XPS GUI.
9. For MicroBlaze targets, PetaLinux requires that the CPU be configured with the MMU enabled. We also recommend to enable at least the barrel shifter feature, 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.

To configure MicroBlaze manually

For experienced XPS users, you can edit the `system.mhs` file to set the following MicroBlaze parameters 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.

MicroBlaze PLB target

1. Run the Xilinx BSB wizard to create a hardware project.

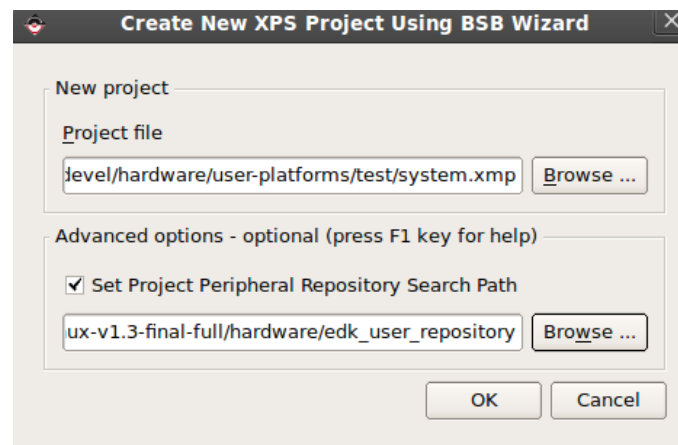
Note: We recommend creating your custom hardware projects in the

`$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.

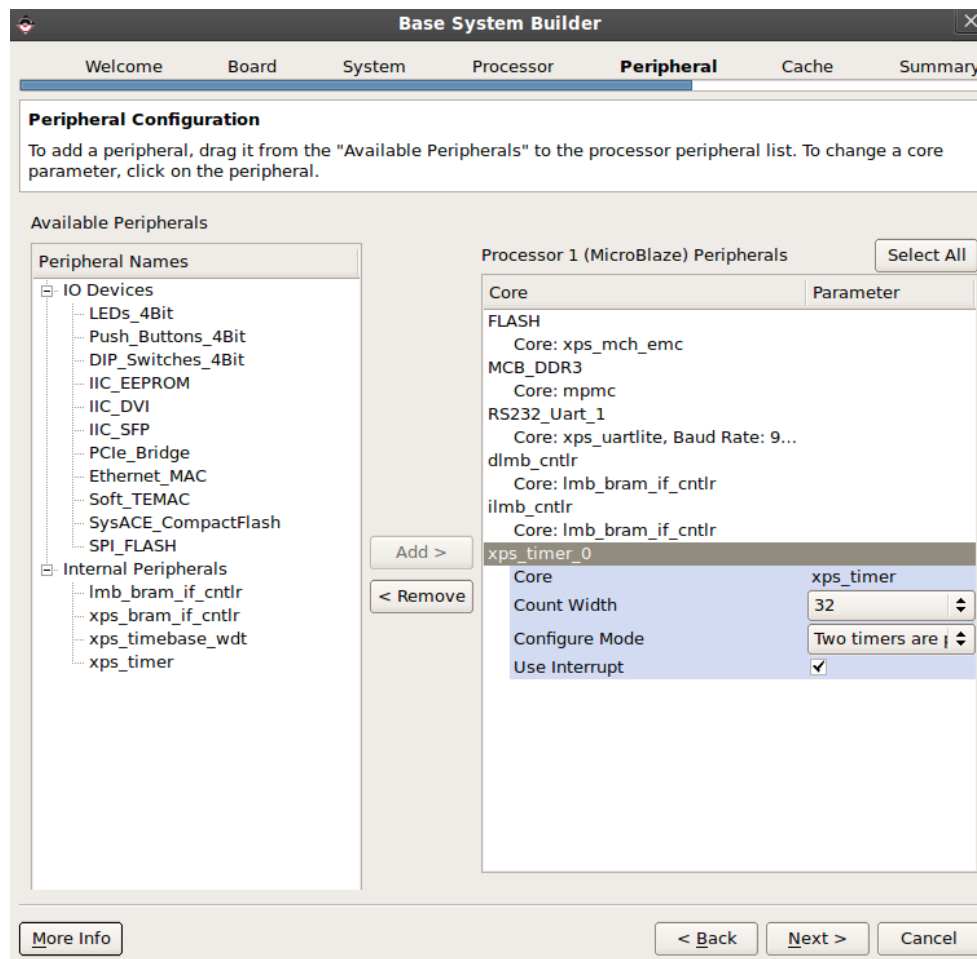
“\$PETALINUX” refers to the directory path where you installed PetaLinux SDK. For example, if PetaLinux is installed in `/home/user/petalinux` directory, the \$PETALINUX refers to `/home/user/petalinux`.

2. In the “Create New XPS project Using BSB Wizard” dialog, set the “Project Peripheral Repository Search Path” to the `$PETALINUX/hardware/edk_user_repository` (substitute the full path, or use the GUI to browse to the location): Please note this step is only required for Xilinx tool version 12.4 or earlier.



3. Select “PLB system”, then click “OK” button. (Xilinx tool version 12.3 or later)
4. Move ahead with through BSB wizard to configure the hardware, until you get to the “Peripheral Configuration” dialog.

5. Add an “xps_timer” to the project and set the timer to use interrupt, and dual-channel mode as shown below:



The bare 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)
- LMB BRAM interface controllers for MicroBlaze (required)
- Non-volatile memory (optional), such as Linear Flash or SPI Flash
- Ethernet (essential for network access)

Any other IP cores are optional.

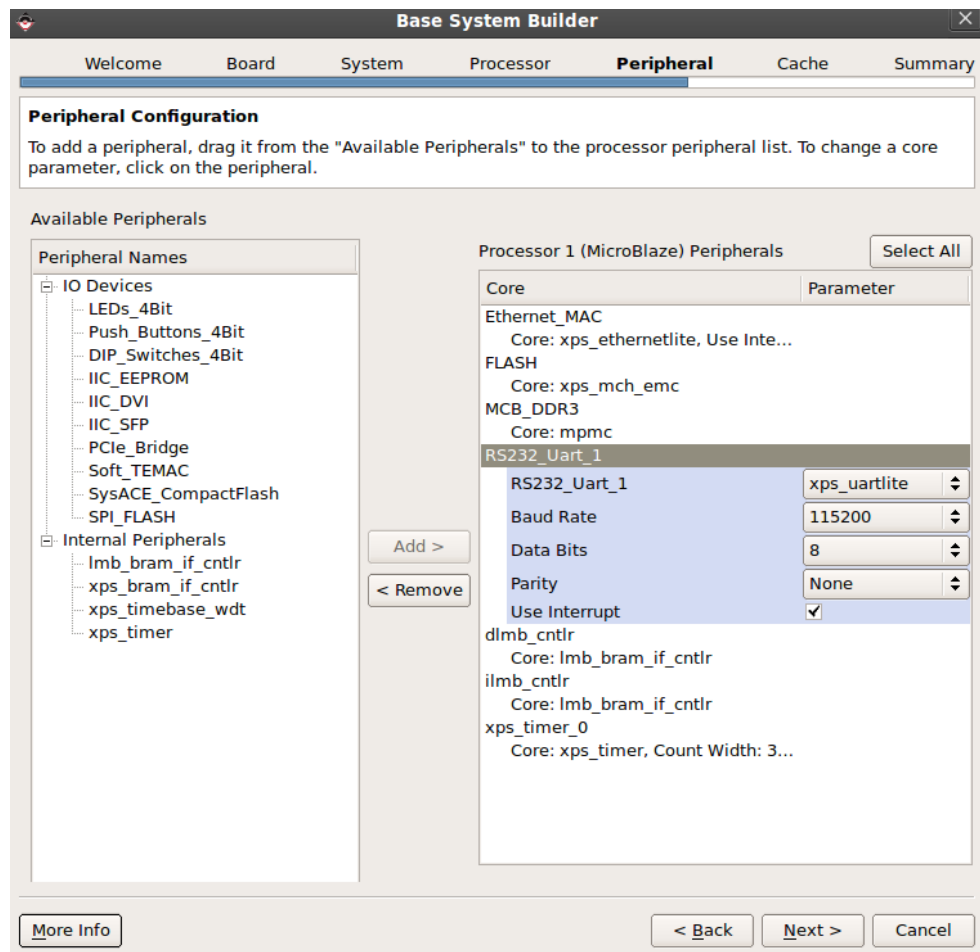
6. Choose your preference of **ethernetlite** or **ll_temac** as the ethernet controller.

Notes

- If using the **ll_temac**, SDMA mode must be used.
 - Regardless of your choice of ethernet controller, ensure that the “**Use Interrupt**” checkbox is set.
6. Choose your preference of serial port console, either **uartlite** or **UART16550**.

Notes

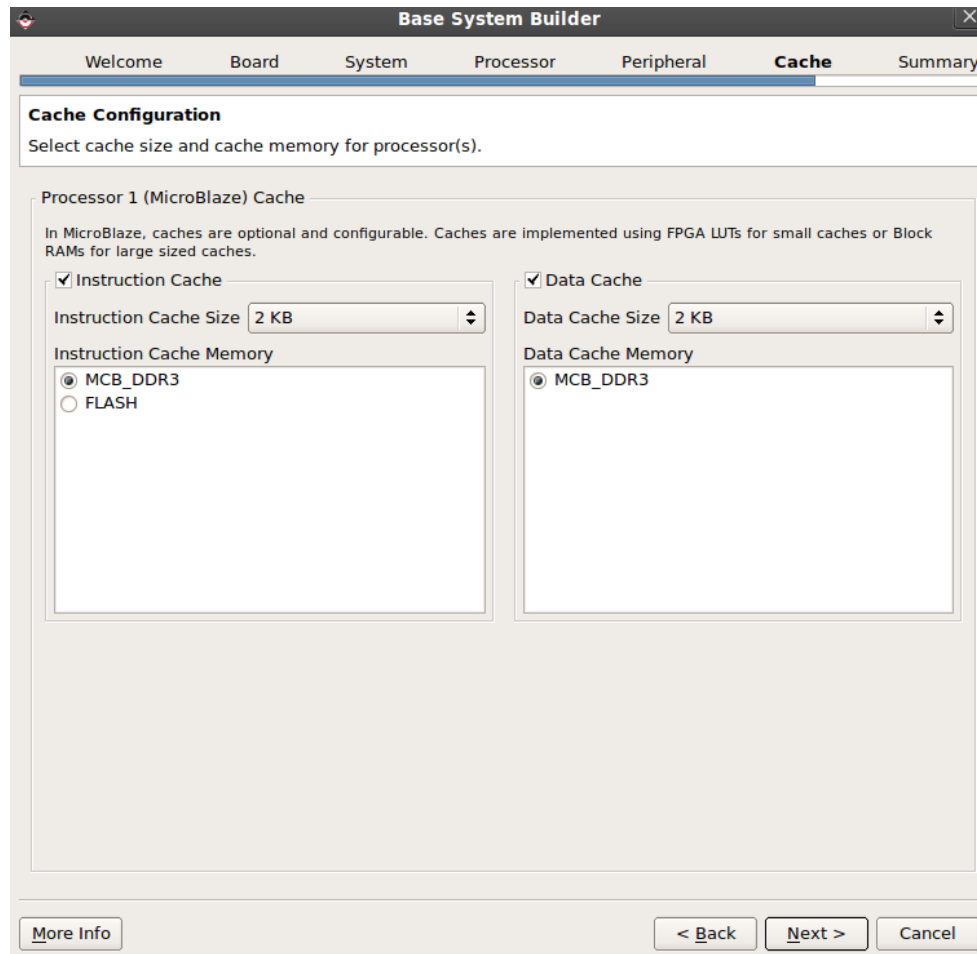
- Be sure to select the “**Use Interrupt**” check box
- If you use “**xps_uartlite**” as the UART IP Core, we recommend a baudrate of 115200:



- After configuring the peripherals, the next step is to configure the CPU cache options, using the “Cache” window of the BSB wizard.

Note:

- PetaLogix recommends that instruction and data caches always be used, and configured to the full range of primary system memory (DDR/SDRAM) as shown below.



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

- After the BSB wizard completes, the hardware project is opened in the XPS GUI.
- Check the `xps_timer` version, if it is version is “1.01.b”, please follow the instructions in <http://www.xilinx.com/support/answers/33880.htm> to fix the timer interrupt sensitivity issue.
- For MicroBlaze targets, PetaLinux requires that the CPU be configured with the MMU enabled. We also recommend to enable at least the barrel shifter feature, for improved performance. Other CPU features may be enabled as required.

To configure MicroBlaze with barrel shifter and MMU (EDK 11.5 and earlier)

- Select the MicroBlaze instance in the “System Assembly View”, right click and select “Configure IP” from the menu list.
- In the MicroBlaze instance “XPS Core Config” window:
 - In the “Instructions” tab, select “Enable Barrel Shifter”.
 - In the same “Instructions” tab, not select “Select implementation to optimize area” option.
 - In the “MMU” tab:
 - Set “Memory management” to “VIRTUAL”
 - Set “Data Shadow Translation Look-Aside Buffer Size” to “4”
 - Set “Instruction Shadow Translation Look-Aside Buffer Size” to “2”
 - Set “Enable Access to Memory Management Special Registers” to “FULL”
 - Set “Number of Memory Protection Zones” to “2”

To configure MicroBlaze for Linux (EDK 12.1 and later)

- 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.

To configure MicroBlaze manually

For experienced XPS users, you can edit the `system.mhs` file to set the following MicroBlaze parameters 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.

PowerPC 440 target

1. Run the Xilinx BSB wizard to create a hardware project.

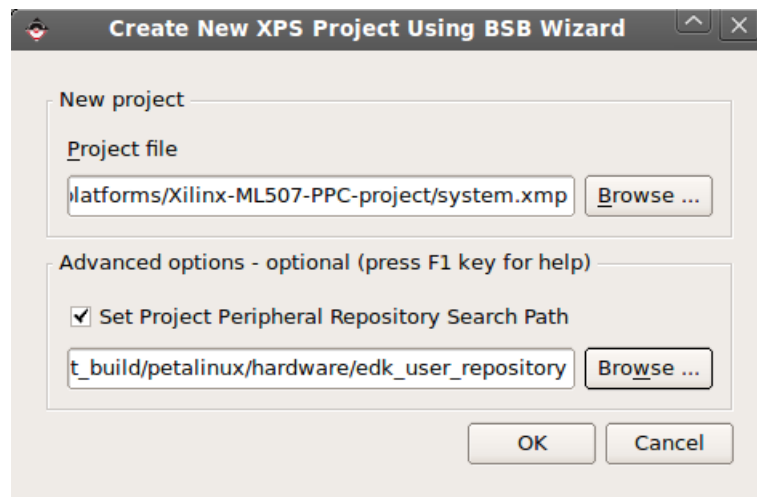
Note: We recommend creating your custom hardware projects in the

`$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.

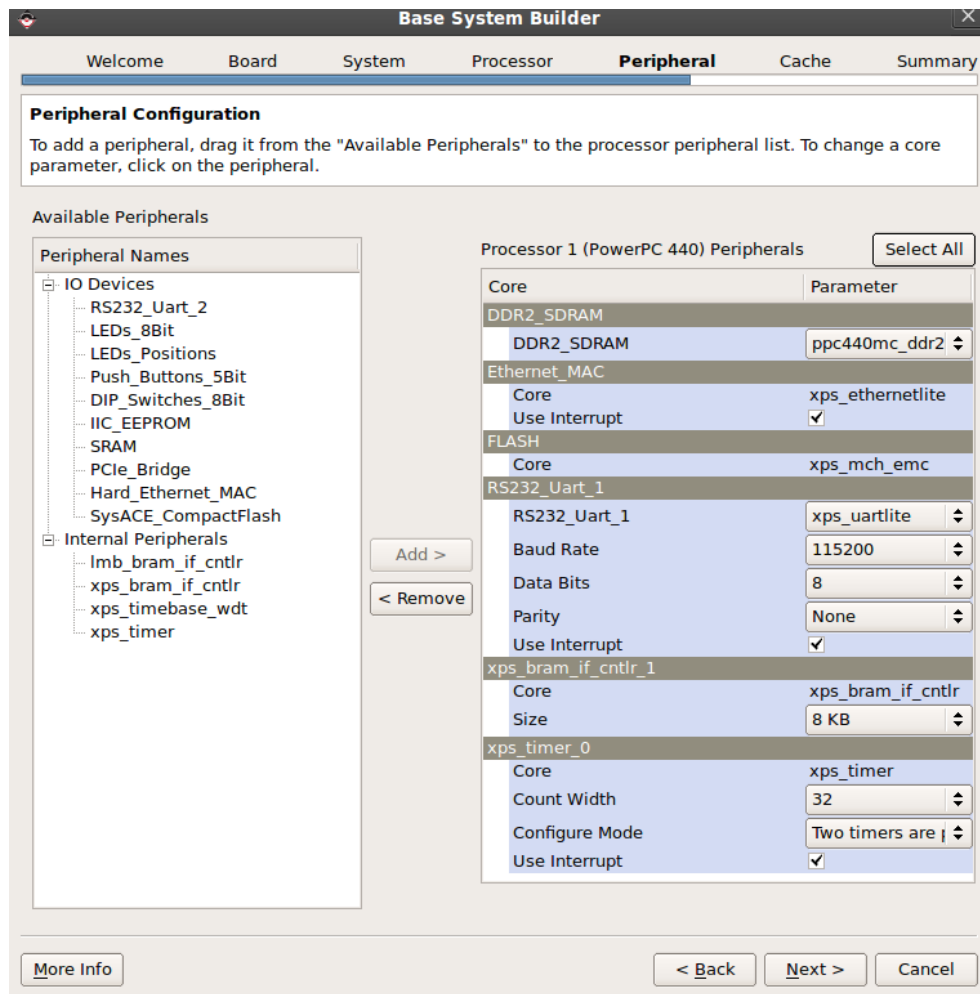
“\$PETALINUX” refers to the directory path where you installed PetaLinux SDK. For example, if PetaLinux is installed in `/home/user/petalinux` directory, the `$PETALINUX` refers to `/home/user/petalinux`.

2. In the “Create New XPS project Using BSB Wizard” dialog, set the “Project Peripheral Repository Search Path” to the `$PETALINUX/hardware/edk_user_repository` (substitute the full path, or use the GUI to browse to the location):



3. Select “PLB system”, then click “OK” button. (Xilinx tool version 12.3 or later)
4. Move ahead with through BSB wizard to configure the hardware, until you get to the “Peripheral Configuration” dialog.

5. Add an “xps_timer” to the project and set the timer to use interrupt, and dual-channel mode as shown below:



The bare 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)
- XPS BRAM Interface Controller (required for fs-boot)
- 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 **ll_temac** as the ethernet controller.

Notes

- If using the **ll_temac**, SDMA mode must be used.
- Regardless of your choice of ethernet controller, ensure that the “**Use Interrupt**” checkbox is set.

7. Choose your preference of ethernetlite or ll_temac as the ethernet controller.

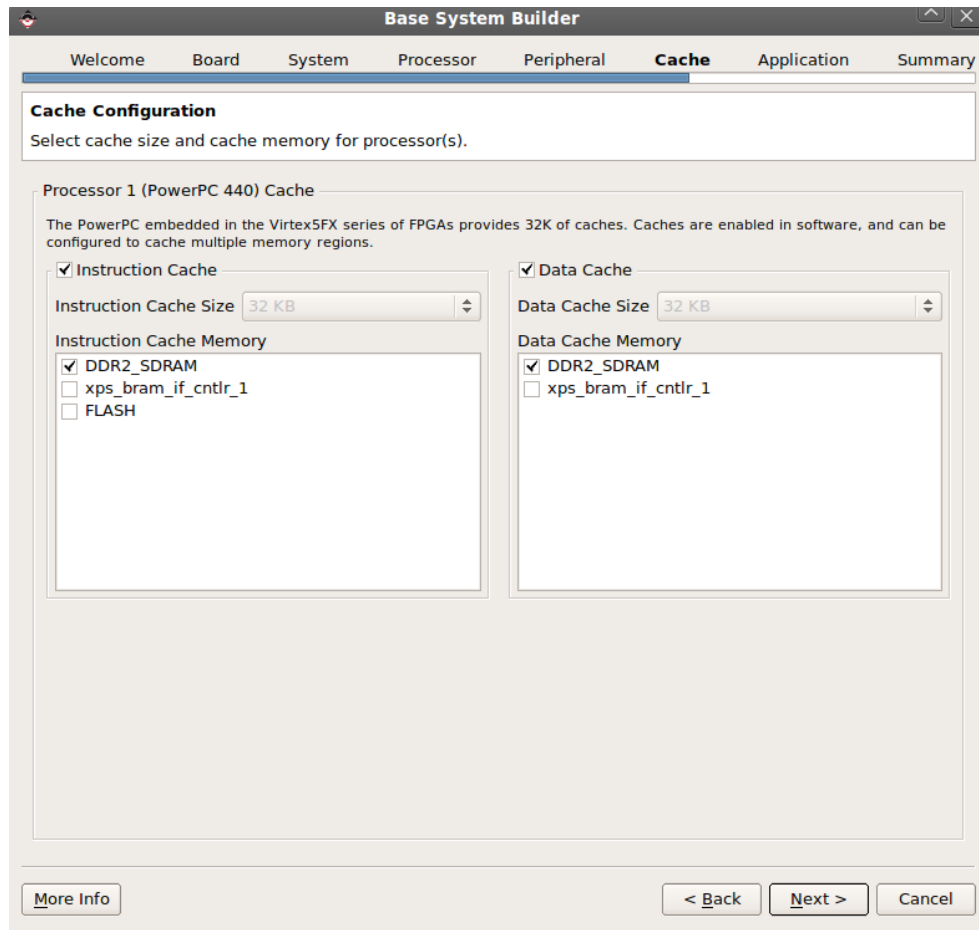
Notes

- Be sure to select the “Use Interrupt” check box
- If you use “xps_uartlite” as the UART IP Core, we recommend a baudrate of 115200.

- After configuring the peripherals, the next step is to configure the CPU cache options, using the “Cache” window of the BSB wizard.

Note:

- PetaLogix recommends that instruction and data caches always be used, and configured to the full range of primary system memory (DDR2/SDRAM) as shown below.



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

- After the BSB wizard completes, the hardware project is opened in the XPS GUI.
- Check the `xps_timer` version, if it is version is “1.01.b”, please follow the instructions in <http://www.xilinx.com/support/answers/33880.htm> to fix the timer interrupt sensitivity issue.

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.

For MicroBlaze AXI target

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

1. Ensure you have `axi_timer` configured in your system and make sure the timer interrupt is enabled, and connected
2. Ensure you have UART core configured and your Xilinx UART core is configured to use interrupts, and that the interrupt signal is connected. If your UART is a `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 DMA for `axi_ethernet`.
4. PetaLogix recommends to configure the MicroBlaze to use both Instruction Cache, Data Cache and Barrel shifter to make the system have better 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.

For MicroBlaze PLB target

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

1. Ensure you have `xps_timer` configured in your system and the timer is configured as follows:
 - interrupt is enabled, and connected
 - dual channel timer must be selected

Note:

- If your `xps_timer` is of version “1.01.b”, please follow the instructions in <http://www.xilinx.com/support/answers/33880.htm> to fix the timer interrupt sensitivity issue.
2. Ensure you have UART core configured and your Xilinx UART core is configured to use interrupts, and that the interrupt signal is connected. If your UART is a `xps_uartlite`, please configure its baud rate to `115200`.
 3. If you have Ethernet CORE – either `xps_ethernetlite` or `xps_ll_temac` –

configured, please configure it to use interrupt. Only SDMA mode is supported for the `xps_ll_temac` IP.

4. PetaLogix recommends to configure the MicroBlaze to use both Instruction Cache, Data Cache and Barrel shifter to make the system have better 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 10 in Section 'Create a Hardware Project with Xilinx Base System Builder' above for details.

For PowerPC440 target

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

1. Ensure you have `xps_timer` configured in your system and the timer is configured as follows:
 - interrupt is enabled, and connected
 - dual channel timer must be selected

Note:

- If your `xps_timer` is of version “1.01.b”, please follow the instructions in <http://www.xilinx.com/support/answers/33880.htm> to fix the timer interrupt sensitivity issue.
2. Ensure you have UART core configured and your Xilinx UART core is configured to use interrupts, and that the interrupt signal is connected. If your UART is a `xps_uartlite`, please configure its baud rate to `115200`.
 3. If you have Ethernet CORE – either `xps_ethernetlite` or `xps_ll_temac` – configured, please configure it to use interrupt. Only SDMA mode is supported for the `xps_ll_temac` IP.

Configuring Software Settings of the Hardware Project and FS-Boot

After designing the hardware system, it is then necessary to configure a Linux BSP for that project. This is required to automate the Linux board bringup. Since Xilinx EDK 13.1 and newer, the XPS BSP workflow is no longer available. Instead the Xilinx SDK (XDSK) workflow must be used to configure the Linux BSP. The steps required for each workflow is described in the following sections.

For XSDK workflow (Xilinx 13.1 and later)

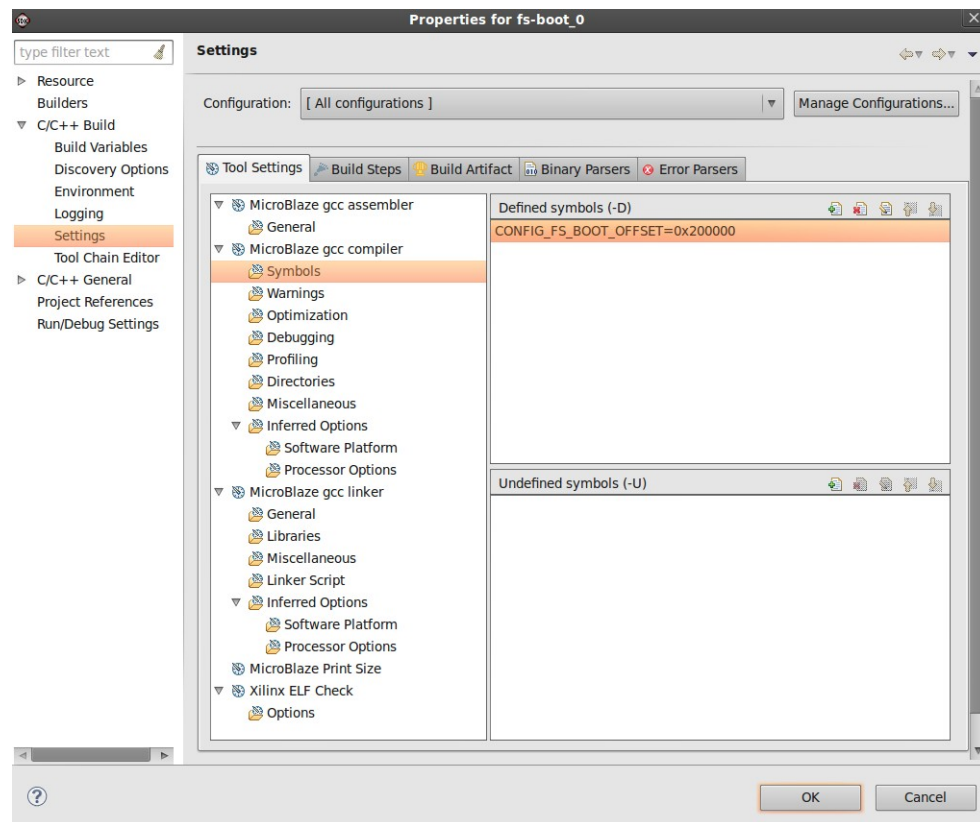
1. Select “Export Hardware design to SDK ” from the “Project” menu in the XPS GUI.
2. Tick “Include bistream and BMM file” box, then click “Export & Launch SDK” button. (Please note, if “Include bistream and BMM file” is ticked, XPS will generate the `system.bit` and `system_bd.bmm` bitstream files before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

Warning: 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.

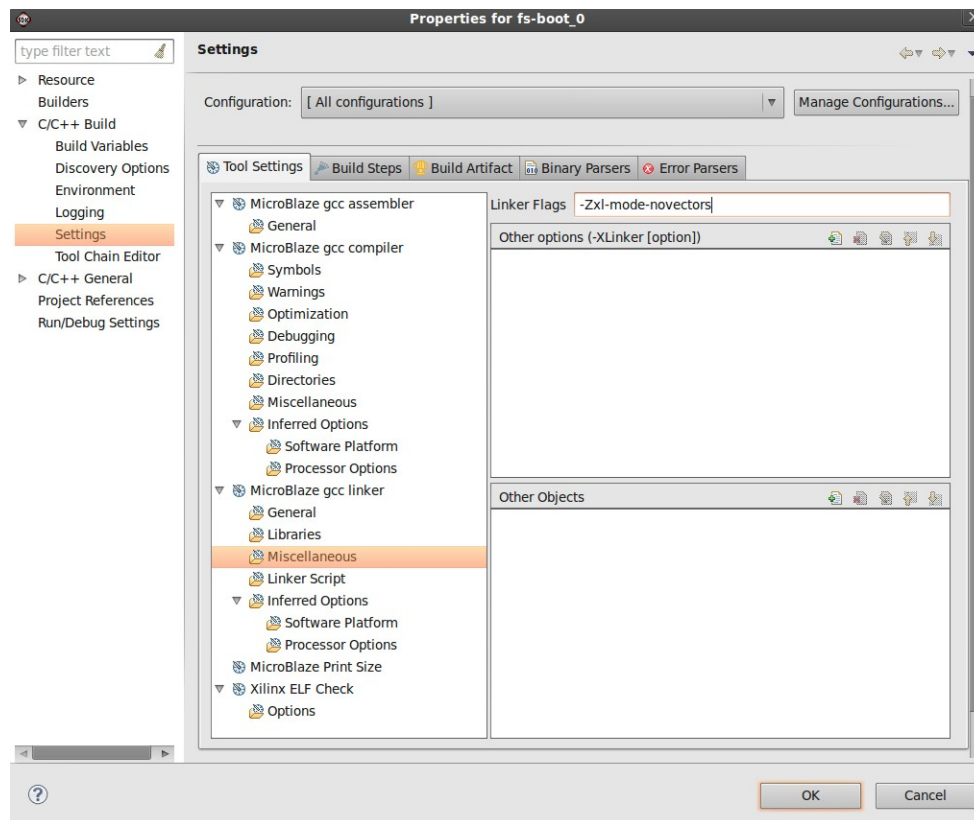
4. Select “Switch workspace” in “File” menu in XSDK, then select “Other”. Ensure the workspace path is point to the “workspace” directory inside your hardware project directory.
5. Add the PetaLinux repository to XSDK to the list of available BSPs:
 - Click “Xilinx Tools” menu tab and select “Repository”
 - Select “Repositories ” on the left menu
 - Click “New” (for local repositories box) on the right hand top corner.
 - Browse to PetaLinux `edk_user_repository` directory- “`$PETALINUX/hardware/edk_user_repository`” and click “OK”
 - Click “OK” to finish adding local repository.
6. Add fs-boot and PetaLinux BSP:
 - Click “File” menu tab and select “New” and select “Xilinx C Project”
 - Select “FS-BOOT” then click “Next”
 - Click “Finish”. By default it creates PetaLinux BSP “fs-boot-bsp_0” and fs-boot “fs-boot_0” application.
7. Configure PetaLinux BSP for U-boot and Linux operation system:
 - Right click on the “fs_boot_bsp_0” project in Project explorer
 - Select “Board Support Package Settings”.
 - Select “petalinux” in the Board Support Package Settings window
 - These settings are automatically selected, please review to ensure they are correct.
 - Click “OK” to accept the settings

Note: At this point, there may be some 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 address in Step 8.

8. Configure fs-boot settings. fs-boot bootloader is used to fetch U-Boot image from the U-boot partition in Flash to main memory and runs U-Boot from main memory when system boot up . To do this, the start address of U-boot partition in Flash memory must be give:
 - Right click on the “fs-boot_0” project in Project explorer and select “C/C++ Build settings”
 - Expand the “C/C++ Build” configuration and select “Settings”
 - Select on the “Tool Settings” tab
 - Set configuration profile to “[All configurations]”
 - Select Symbols
 - For MicroBlaze target, click “Symbols” under “MicroBlaze gcc compiler”
 - For PowerPC440 target, click “Symbols” under “PowerPC gcc compiler”
 - In the “Defined symbols (-D)” box click the “Add” button (the small icon with a green plus symbol on it)
 - In the “Enter Value” windows, 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”.



- **For MicroBlaze target ONLY.** As fs-boot is a simple bootloader, it does not require setting up vector tables. This can be done in following steps:
 - Click “Miscellaneous” under “MicroBlaze gcc linker”
 - Append “-Zx1-mode-novectors” flag to “Linker Flags” field.
- Click “Apply” button to apply the settings
- Click “OK” button to finish Compiler flag configuration



9. Set to fs-boot to “Release” profile
 - Right click on the “fs - boot_0” project in Project explorer
 - Select “Build Configurations”
 - 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.
 - Right click on the “fs - boot_0” project in Project explorer and select “Build Project” to build the project if the Xilinx Eclipse/SDK did not automatically build the “fs - boot_0” software project.

For XPS workflow (Xilinx 12.4 and earlier)

1. Select “Project Options” from the “Project” menu in the XPS GUI.
2. Modify the “Project Peripheral Repository Search Path” to point to the directory “\$PETALINUX/hardware/edk_user_repository”. Substitute the correct value of the \$PETALINUX installation directory as appropriate, or use the GUI to browse to the correct location.

3. Select “Software Platform Settings” from the “Software” menu in XPS GUI.
4. In the “Software Platform” tab of the “Software Platform Settings” window, select “petalinux” as the “OS”
5. In the “OS and Libs Configuration” tab of the “Software Platform Settings” window, choose the appropriate IP instances for the following system features:
 - “stdout” - the serial controller IP for the main system console (UARTLITE or UART16550)
 - “stdin” - same as for stdout
 - “main_memory” - the memory controller instance to which the main system memory is attached – usually the MPMC.
 - “flash_memory” - the parallel or serial (SPI) flash memory controller to which the main system flash memory is attached. Usually an XPS_EMC for parallel flash, or XPS_SPI for serial (SPI) flash. Leave blank if the system contains no flash memory.
 - “lmb_memory” - select the LMB BRAM controller instance that is mapped to the zero-based address space of MicroBlaze-based systems.
 - **NOTE: This parameter should not be set for PPC440 systems.**
 - “ethernet” - select the instance of the ethernet controller IP in the system. If your system contains multiple ethernet controllers, choose the primary one.
 - “timer” - select the XPS_TIMER instance that is to be used as the primary system timer. Its interrupt line must be connected, and it must be configured as a dual-channel timer.

Advanced users may wish to modify the `system.mss` directly. An example is shown below – you will need to adjust the parameter values to match the name and type of IP instances in your system.

```
BEGIN OS
PARAMETER OS_NAME = petalinux
PARAMETER OS_VER = 2.00.a
PARAMETER PROC_INSTANCE = <cpuname>
# do not set lmb_memory parameter for PowerPC systems
PARAMETER lmb_memory = dlmb_cntlr
PARAMETER main_memory = DDR2_SDRAM
PARAMETER main_memory_bank = 0
PARAMETER flash_memory = FLASH
PARAMETER flash_memory_bank = 0
PARAMETER stdin = RS232_Uart_1
PARAMETER stdout = RS232_Uart_1
PARAMETER timer = system_timer
```

```
PARAMETER ethernet = Hard_Ethernet_MAC
END
```

6. Add fs-boot software project

- Click “Project” tab of Project information area
- Double-click on the “Add Software Application Project..”
- Enter “fs-boot_0” in the name field of “Add Software Application Project” window

7. Configure fs-boot settings

- Right click on “Project: fs-boot_0” project and select “Set Compiler Options”
- Select “Environment” tab in the “Compiler Option” window
- Tick “Use Default Linker Script”
- Set “Stack size” to “0x400”
- Set “Heap size” to “0x0”
- Select “Path and Options” tab in the “Compiler Option” window
- Append compiler option for target
 - For MicroBlaze target: In the “Other Compiler Options to Append” box, enter “-DCONFIG_FS_BOOT_OFFSET=<value> -Zxl-mode-novectors”.
 - For PowerPC440 target: In the “Other Compiler Options to Append” box, enter “-DCONFIG_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”.

8. Add Sources and Headers to fs-boot

- Right click on “Sources” of fs-boot_0 project and select “Add Existing Files...”
- Browse to “\$PETALINUX/hardware/fs-boot” directory and select all the *.c files then click “OK” to add them
- Right click on “Headers” of “Project: fs-boot_0” project and select “Add Existing Files...”
- Browse to “\$PETALINUX/hardware/fs-boot” directory and select all the *.h files then click “OK” to add them

9. Right click on “Project: fs-boot_0” project and select “Build Project” to build fs-boot

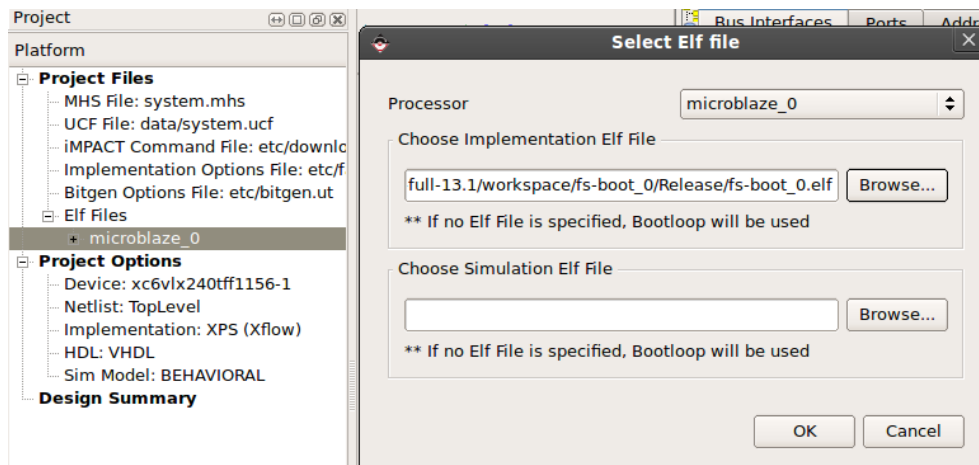
XPS tools will also run the PetaLinux BSP generation tools. These tools automatically generate the software platform configuration files required in the subsequent board bringup stages. These files are described in more detail later in this document.

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 initialise to BRAM yet. This following section documents how to initialise the fs-boot to BRAM and create the bitstream

XSDK workflow

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 be 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 init BRAM using the normal XPS commands from the command line.

The FPGA bistream is now ready for download.

XPS workflow (Xilinx 12.4 and earlier)

1. Click "Project" tab of Project information area
2. Right click on "Project: fs-boot_0" project and select "Mark to initialize to BRAMs"
3. Select "Device Configuration" menu and select "Update Bitstream" or "Download Bitstream" to create the bitstream, `download.bit`, which fs-boot is initialised to BRAM.

You may commence the hardware project build using the normal XPS commands, either through the GUI or from the command line for advanced users. This step will take a while to complete.

The FPGA bistream is now ready for download.

Create New PetaLinux SDK Software 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>
```

NOTE: Prior to PetaLinux SDK v1.3, the `-c` option was not required. From PetaLinux 1.3, this option exists to choose between MicroBlaze and PowerPC440 target architectures.

The parameters are as follows:

- `-c <cpu type>` As of PetaLinux 2.1, supported CPU types are 'microblaze' and 'ppc440'.
- `-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 bootup.
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 toplevel 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 appropriated directory before use “`petalinux-copy-autoconfig`” command
 - For XSDK workflow, navigate to the PetaLinux BSP directory “`<hardware project directory>/workspace/fs-boot_bsp_0/`”.
 - For XPS workflow, navigate to the hardware project directory
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-13.1"
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 generate the hardware configuration files if they not generated yet and copy the configuration files to the right place in PetaLinux.

- The DTS file will be placed in
\$PETALINUX/software/petalinux-dist/linux-2.6.x \\
/arch/{microblaze|powerpc}/boot/dts/<VENDOR>-<PLATFORM>.dts.
- The xparameters.h and config.mk files are placed in
\$PETALINUX/software/petalinux-dist/u-boot\
/petalogix/{microblaze|ppc440}-auto

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
```

5. Once you are satisfy with the configurations and proper tested. It is recommended to update your default configurations. So you do not lost your configuration when switch from one software platform to another.

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

Build PetaLinux

Finally, it's 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 ucf front 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.

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

```
$ cd $PETALINUX/software/petalinux-dist
$ petalinux-jtag-boot -i images/u-boot.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 bring up your board with PetaLinux SDK, and ways to solve them.

Problem	Description and Solution
<p>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.</p>	<p><i>Problem Description:</i></p> <p>This problem is usually because the type of the IP instance is not a standard Xilinx IP core but a customized IP Core.</p> <p><i>Solution:</i></p> <ul style="list-style-type: none"> ● Set that configuration item to “none”. ● Manually edit the DTS file: <code>petalinux-dist/linux-2.6.x/arch/{microblaze powerpc}/boot/dts/<VENDOR>-<PLATFORM>.dts</code> to configure the node of the IP CORE as you required before building PetaLinux. You can find the DTS introduction file here: <code>petalinux-dist/linux-2.6.x/Documentation/powerpc/booting-without-of.txt</code> for Linux kernel. ● Manually edit the <code>petalinux-dist/u-boot/petalogix/{microblaze ppc440}-auto/xparameters.h</code> 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. We recommend <code>of-platform device</code> framework for the IP CORE Linux driver to use the device tree for hardware configuration inquiry.

Problem	Description and Solution
Cannot see any output when try to boot u-boot with UART16550.	<p><u>Problem Description:</u></p> <p>This problem is usually because the serial communication terminal application is set with the wrong baudrate or the clock frequency of UART16550 is missing in the xparameters.h in u-boot.</p> <p><u>Solution:</u></p> <ul style="list-style-type: none">● Check whether your terminal application baudrate is correct.● Check whether “#define XILINX_UART16550_CLOCK_HZ <FREQ>” is in petalinux-dist/u-boot/petalogix/{microblaze ppc440}-auto/xparameters.h. If no, run petalinux-copy-autoconfig again from the hardware directory.● Check petalinux-dist/u-boot/petalogix/{microblaze ppc440}-auto/xparameters.h again. If it is still not there, it should be because of the Xilinx tool version is too old. It is recommended to have Xilinx 11.3 or the above. 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 PLB it connects to which is usually the system clock frequency.

Problem	Description and Solution
<p>Cannot see any output when try to boot u-boot with UART16550.</p>	<p><u>Problem Description:</u> This problem is usually because the serial communication terminal application is set with the wrong baudrate or the clock frequency of UART16550 is missing in the <code>xparameters.h</code> in dts.</p> <p><u>Solution:</u></p> <ul style="list-style-type: none"> ● Check whether your terminal application baudrate is correct. ● Check whether “<code>clock-frequency = <FREQ>;</code>” of the UART 16550 node is in <code>petalinux-dist/linux-2.6.x/arch/{microblaze powerpc}/boot/dts/<VENDOR>-<PLATFORM>.dts</code>. If no, run <code>petalinux-copy-autoconfig</code> again from the hardware project directory. ● Check the DTS file again. If it is still not there, it should be because of the Xilinx tool version is too old. It is recommended to have Xilinx 11.3 or the above. 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 PLB it connects to which is usually the system clock frequency.