

Zynq Troubleshooting Guide

FSBL

FSBL is responsible for all system initialization and configuration.

FSBL Checklist

1. Are relevant PS7 settings correct in Vivado IPI BD customization Dialog?
2. If using board part files, has the automation been executed on the PS7 block with apply presets enabled?
3. Has Vivado flow been executed successfully until bitgen?
4. Has the hardware been exported with include bitstream option?
5. Is SDK using the hardware description exported from Vivado?
6. Has bsp for FSBL been regenerated from sources since last export?
7. Is FSBL set to use correct bsp?
8. Has FSBL been re-compiled?
9. Is bootgen using the correct fsbl.elf file?
10. Is the sdcard formatted as ms-dos, not gpt?

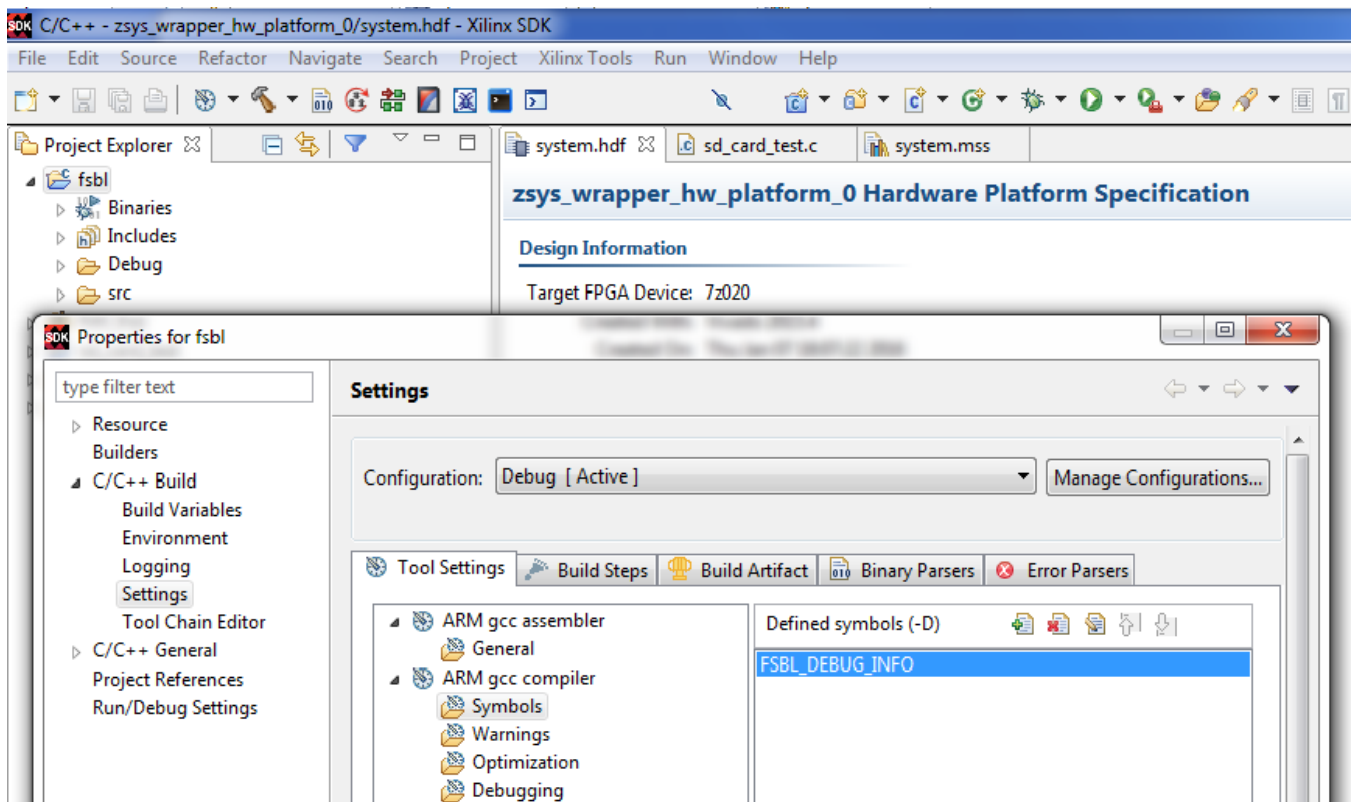
No Console Output

Xilinx standard FSBL when compiled with default settings is in "quiet" mode, with no console output if something goes wrong. If there is any doubt that there are problems with FSBL it is necessary to make FSBL more verbose. One possibility is to enable DEBUG logging in FSBL by defining compiler symbol.

If there is however problem with DDR memory there may not come any log messages, also when DEBUG is enabled. In such cases adding

```
xil_printf("FSBL before DDR");
```

just at the begin of main() in FSBL would help to see if at least the console UART is working.



Enabling Debug info logging in FSBL.

zynq_flash

Is a command-line tool from Xilinx to write nonvolatile memory connected to Zynq PS. With current versions of tools there is minor problem with file extensions: on windows platforms only lowercase .bin is accepted as extension. Boot bin generated with petalinux would be however generated as BOOT.BIN with uppercase extension. It is not possible to use it with Xilinx SDK Flash Programmer GUI or the commandline zynq_flash. GUI Flasher will not allow to select a file with upper case extension, and zynq_flash will also refuse to flash from it if the file is given with uppercase extension. As a workaround the image to be burned has to be renamed with lower case extension (.bin). The name of the image can be anything, just the extension has to be bin in lowercase. There is no need to convert bin to mcs for the zynq_flash, it makes little sense.

bootgen

File order is important when creating Zynq Boot image with Xilinx bootgen, BITSTREAM if present must be specified as second file, before application and second stage bootloader (normally u-boot). If not then Xilinx standard FSBL will give an error.

Note: Xilinx how-to "Prepare Boot Image" is incomplete and out-dated as well (rev 23 from November 2014). Xilinx how-to does not show a bitstream at all, and lists files to be included that are not needed with latest petalinux standard kernels.

SDK

Create Zynq Boot Image

X

Create Zynq Boot Image

Creates Zynq Boot Image in .bin and .mcs formats from given FSBL elf and partition files in specified output folder.

Create new BIF file

Import from existing BIF file

Import BIF file path:

B:\micromodules\TE0715\current\hdl\TE0715_ref\TE0715_ref.sdk\linux\bootimage\linux.bif

Browse

Output BIF file path:

B:\micromodules\TE0715\current\hdl\TE0715_ref\TE0715_ref.sdk\linux\bootimage\linux.bif

Browse

Use Authentication

Authentication keys

PPK:

Browse

PSK:

Browse

SPK:

Browse

SSK:

Browse

SPK signature:

Browse

Use encryption

Encryption key:

Key file:

Browse

Key store:

BRAM

EFUSE

Part name:

Boot image partitions

File path	Encrypted	Authenticated
(bootloader) B:\micromodules\TE0715\current\hdl\TE0715_ref\TE0715_ref.sdk\FSBL\Debug\FSBL.elf	none	none
B:\micromodules\TE0715\current\hdl\TE0715_ref\TE0715_ref.sdk\design_1_wrapper_hw_platform_0\design_1_wrapper.bit	none	none
B:\micromodules\TE0715\current\hdl\TE0715_ref\TE0715_ref.sdk\linux\u-boot.elf	none	none

Add

Delete

Edit

Up

Down

Output path:

B:\micromodules\TE0715\current\hdl\TE0715_ref\TE0715_ref.sdk\linux\bootimage\BOOT.bin

Browse

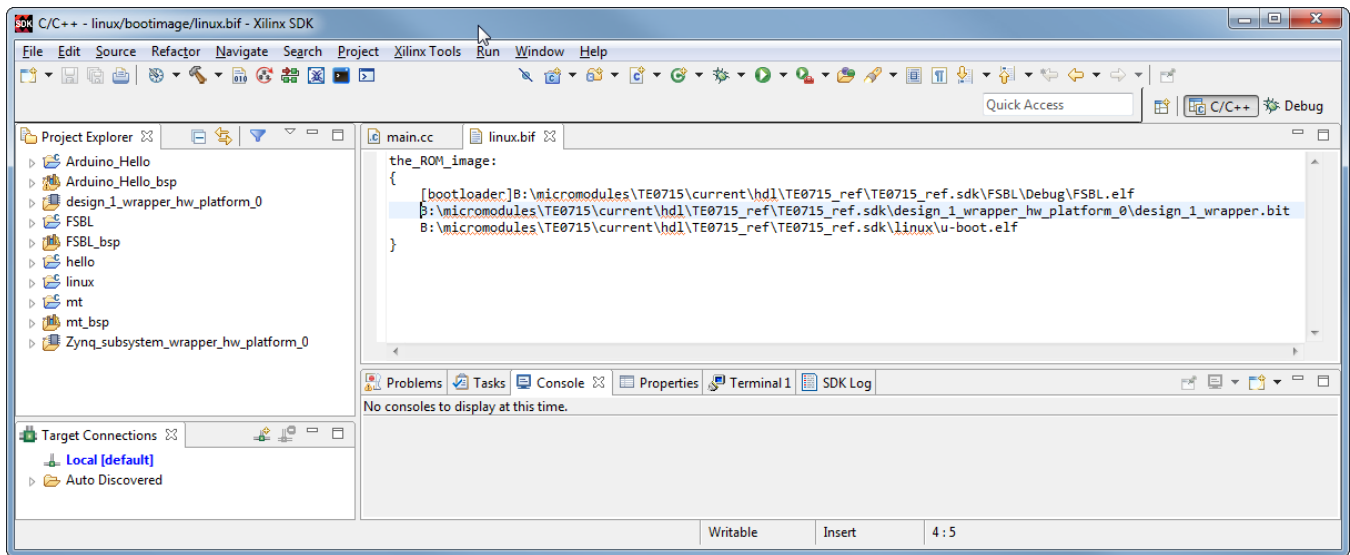
?

Preview BIF Changes

Create Image

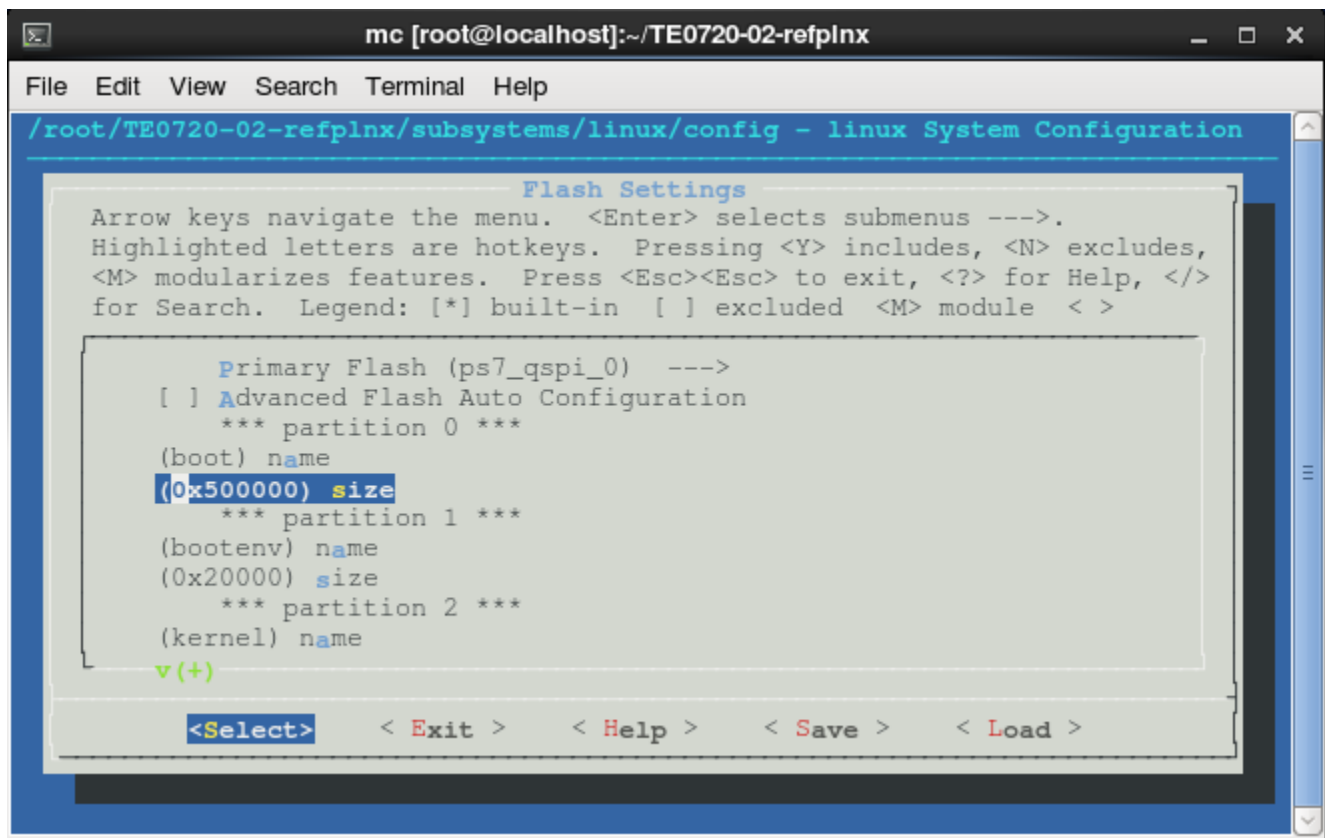
Cancel

Defaults by SDK, dummy "hello.elf" replaced with u-boot.elf to create BOOT.bin for Linux boot from SD Card.



Example BIF file for BOOT.bin, FPGA bitstream is clearly visible as second file. This is how SDK wizard does it by default, this order should not be changed. As 3rd file is normally the second stage bootloader (u-boot). For boot file for SD Card, this is all that goes into BOOT.bin, the remaining files will be copied to SD card. For SPI Flash images linux image should be added as well as last file, with correct offset (it must match the offset u-boot is expecting it).

Note: petalinux-package (2014.4) seems to have trouble with offset in BIF, so it would fail to create boot images for SPI Flash (SD Card images are OK). As workaround the boot images should be generated with bootgen using manually created bif files.



```
mc [root@localhost]:~/TE0720-02-refplnx
File Edit View Search Terminal Help
/root/TE0720-02-refplnx/subsystems/linux/config - linux System Configuration

Flash Settings
Arrow keys navigate the menu. <Enter> selects submenus --->.
Highlighted letters are hotkeys. Pressing <Y> includes, <N> excludes,
<M> modularizes features. Press <Esc><Esc> to exit, <?> for Help, </>
for Search. Legend: [*] built-in [ ] excluded <M> module < >

Primary Flash (ps7_qspi_0) --->
[ ] Advanced Flash Auto Configuration
*** partition 0 ***
(boot) name
(0x500000) size
*** partition 1 ***
(bootenv) name
(0x20000) size
*** partition 2 ***
(kernel) name
v(+)

<Select> <Exit> <Help> <Save> <Load>
```

Petalinux default size for boot partition is 0x50_0000 and 0x2_0000 for Flash environment, hence the offset for image.ub in the BOOT.bin for SPI Flash should be set to 0x520000.

Boot from SD Card

If there is doubt that booting from SD card is failing, then as first step a special BOOT.bin should be used to verify that Zynq Bootrom can load the boot code from SD Card at all. This special image does not use or initialize any PS peripherals including external DDR memory, it also does not attempt to configure the PL portion. The image simply Blinks a LED connected to some MIO pin. The same image can be used on any Zynq device, if the SD boot process succeeds at all then code in the image will be executing.



Xilinx Zynq bootrom does support SD Spec ver1 Cards, but Xilinx FSBL does not. So very old SD Cards would not boot at all when using Xilinx supplied FSBL.

SDIO MIO Pullups

If SDIO interface is connected to SD card via special purpose level shifter IC, then it maybe desirable or even necessary to disable MIO internal pullups for SDIO pins. Some SD Cards may fail when the SDIO MIO internal pullups are enabled.

References

1. [Xilinx AR63149](#)
2. [Xilinx AR59476](#)