Build U-Boot

bootloader is a part of the Xilinx design flow described in Xilinx Open Source Linux. U-Boot depends upon an externally build device tree compiler (dtc) in order to build successfully. Please build the dtc tool before proceeding with the steps described below.

Table of Contents

Task Dependencies (Pre-requisites)

- Install Xilinx tools
- Fetch Sources
- Build Device Tree Compiler

Tools Required

- Xilinx SDK

Input Files Required

- config.mk (MicroBlaze only)
- xparameters.h (MicroBlaze only)

Output Files Produced

- u-boot
- mkimage

U-boot build steps

All commands have to be executed in your u-boot source directory.

- Clone u-boot git clone https://github.com/Xilinx/u-boot-xlnx.git
- cd to u-boot directory "cd u-boot-xlnx".
- Add tool chain to path and then set tool chain as below(tool chain name may vary based on toll chain

ZynqUS+:

export CROSS_COMPILE=aarch64-linux-gnu-

export ARCH=aarch64

Microblaze:

export CROSS_COMPILE=microblazeel-xilinx-linux-gnu-

export ARCH=microblazeel

- Configure the sources for the intended target. Xilinx u-boot supports the following targets

Platform	U-Boot defconfig
Zynq UltraScale+ MPSoC mini	xilinx_zynqmp_mini_defconfig
Zynq UltraScale+ MPSoC mini_emm0	xilinx_zynqmp_mini_emmc0_defconfig
Zynq UltraScale+ MPSoC mini_emm1	xilinx_zynqmp_mini_emmc1_defconfig
Zynq UltraScale+ MPSoC mini_nand	xilinx_zynqmp_mini_nand_defconfig
Zynq UltraScale+ MPSoC mini_qspi	xilinx_zynqmp_mini_qspi_defconfig
Zynq UltraScale+ MPSoC ZC1232	xilinx_zynqmp_zc1232_revA_defconfig
Zynq UltraScale+ MPSoC ZC1254	xilinx_zynqmp_zc1254_revA_defconfig
Zynq UltraScale+ MPSoC ZC1275	xilinx_zynqmp_zc1275_revB_defconfig
Zynq UltraScale+ MPSoC ZC1751_dc1	xilinx_zynqmp_zc1751_xm015_dc1_defconfig
Zynq UltraScale+ MPSoC ZC1751_dc2	xilinx_zynqmp_zc1751_xm016_dc2_defconfig
Zynq UltraScale+ MPSoC ZC1751_dc3	xilinx_zynqmp_zc1751_xm017_dc3_defconfig
Zynq UltraScale+ MPSoC ZC1751_dc4	xilinx_zynqmp_zc1751_xm018_dc4_defconfig
Zynq UltraScale+ MPSoC ZC1751_dc5	xilinx_zynqmp_zc1751_xm019_dc5_defconfig
Zynq UltraScale+ MPSoC ZCU100	xilinx_zynqmp_zcu100_revC_defconfig
Zynq UltraScale+ MPSoC ZCU102	xilinx_zynqmp_zcu102_rev1_0_defconfig
Zynq UltraScale+ MPSoC ZCU104	xilinx_zynqmp_zcu104_revC_defconfig
Zynq UltraScale+ MPSoC ZCU106	xilinx_zynqmp_zcu106_revA_defconfig
Zynq UltraScale+ MPSoC ZCU111	xilinx_zynqmp_zcu111_revA_defconfig
Zynq UltraScale+ MPSoC Ultra96	avnet_ultra96_rev1_defconfig
Zynq CC108	zynq_cc108_defconfig
Zynq cse_nand	zynq_cse_nand_defconfig
Zynq cse_nor	zynq_cse_nor_defconfig

Zynq cse_qspi	zynq_cse_qspi_defconfig
Zynq ZC702	zynq_zc702_defconfig

Zynq ZC770 DC2 with x8 NAND	zynq_zc770_xm011_defconfig
Zynq ZC770 DC2 with x16 NAND	zynq_zc770_xm011_x16_defconfig
Zynq ZC770 DC3	zynq_zc770_xm012_defconfig
Zynq ZC770 DC4	zynq_zc770_xm013_defconfig
Avnet Zynq Zed (Z7) Board	zynq_zed_defconfig
Avnet Mini-Zed (Z7) Board	zynq_minized_defconfig
Avnet Pico-Zed (Z7) Board	zynq_picozed_defconfig
Avnet MicroZed (ZU+) Board	zynq_microzed_defconfig
Digilent ZYBO (1st Edition)	zynq_zybo_defconfig
Digilent ZYBO-7Z (2nd Edition)	zynq_zybo_z7_defconfig
Microblaze Board	microblaze-generic_defconfig

As an example to build U-Boot for ZC702 execute:

```
make distclean
make zynq_zc702_defconfig
make
```

As an example to buildU-Boot for ZCU102 execute:

```
make distclean
make xilinx_zynqmp_zcu102_rev1_0_defconfig
make
```

After the build process completes the target u-boot elf-file is created in the top level source directory, named 'u-boot/u-boot.elf'. Additionally in the tools/ directory the 'mkimage' utility is created, which is used in other tasks to wrap images into u-boot format.
For 2020.1 and above releases common defconfig is being made.

Platform	U-boot defconfig
All Zynq UltraScale+ (except for mini)	xilinx_zynqmp_virt_defconfig
All Zynq (except for mini)	xilinx_zynq_virt_defconfig

To build U-boot for ZC702 board, follow below steps.

```
make distclean
make xilinx_zynq_virt_defconfig
```

```
export DEVICE_TREE="zynq-zc702"
make
```



```
make distclean
make xilinx_zynqmp_virt_defconfig
export DEVICE_TREE="zynqmp-zcu102-rev1.0"
make
```

device tree can be found under arch/arm/dts/ with the name zynqmp-zcu102-rev1.0.dts.
Similarly one can find for all other zynqmp/zynq boards with matching name.
If u-boot is build without exporting DEVICE_TREE, the dtb needs to be loaded at 0x100000 in ddr(configurable with CONFIG_XILINX_OF_BOARD_DTB_ADDR).

device tree should be loaded before loading u-boot.elf.

The dtb load address(0x100000) is same for zynq/zynqmp by default.

To make mkimage available in other steps, it is recommended to add the tools directory to your $PATH.

```
cd tools
export PATH=`pwd`:$PATH
```

U-Boot for MicroBlaze

The Xilinx U-Boot project is based on the source code from git:git.denx.de

The devices that have been tested include UART lite, UART 16550, Linear flash, EMAC lite, LL TEMAC with PLB DMA, and AXI EMAC with AXI DMA. The timer counter and interrupt controller were also tested.
Tests were done on Spartan 605 (PLB and AXI) and Kintex 705 (AXI) evaluation platforms using XPS 14.2 and 14.3.

The hardware project

Typical
Make sure the following peripherals are included in the system:

Name	Value
MicroBlaze processor configured for (Low-end) Linux with MMU	microblaze cpu
serial	uartlite or uart16550
external memory controller	ddr or sdram
intc	interrupt controller
timer	timer with interrupts

Optional
U-Boot includes capabilities of writing to/reading from flash and also transferring files over a network if the available hardware is present:

flash	NOR flash

spi	spi/qspi flashes
i2c	i2c controller

Building U-Boot

There are two ways to compile u-boot for Microblaze board.
1. OSL flow to compile u-boot
The U-Boot source tree can be downloaded from the Xilinx Git server. The repository is **u-boot-xlnx** and the branch is **master**:

```
git clone https://github.com/Xilinx/u-boot-xlnx.git
```

The U-Boot compilation will use the definitions and flags defined in config.mk and xparameters.h . Import these two files generated by the petalinux project into the U-Boot directory for MicroBlaze

```
cp <plnx-proj-root>/components/plnx_workspace/fsboot/fsboot/mba_fs_boot_bsp/micro
cp <plnx-proj-root>/project-spec/meta-plnx-generated/recipes-bsp/u-boot/configs/c
```

Rename platform-auto.h to microblaze-generic.h

```
<plnx-proj-root>/project-spec/meta-plnx-generated/recipes-bsp/u-boot/configs/plat
```

After renaming, copy microblaze-generic.h to the u-boot-xlnx/include/configs

```
cp <plnx-proj-root>/project-spec/meta-plnx-generated/recipes-bsp/u-boot/configs/m
```

Replace the dts file of u-boot-xlnx via petalinux project

```
Convert the .dtb file into .dts file in petalinux project
dtc -I dtb -O dts -o <plnx-proj-root>/components/plnx_workspace/device-tree/devic
```

Renamed this generated dts file to microblaze-generic.dts and copy it to the u-boot-xlnx

```
<plnx-proj-root>/components/plnx_workspace/device-tree/device-tree-generation/mic
cp <plnx-proj-root>/components/plnx_workspace/device-tree/device-tree-generation/
```

Go into the u-boot-xilinx top directory and run the following commands in bash to compile the u-boot

```
make distclean
export CROSS_COMPILE=microblazeel-xilinx-linux-gnu-
export ARCH=microblazeel
make microblaze-generic_defconfig
```

```
scripts/kconfig/merge_config.sh -m .config  <plnx-proj-root >/project-spec/meta-p
make
```


Once the bitstream containing at least a minimally configured hardware system has been downloaded to the FPGA, XMD is used to download the cross-compiled U-Boot to the MicroBlaze soft processor:

```
XMD> connect mb mdm
XMD> dow <u-boot-xlnlx>/build/u-boot
XMD> run
```

U-Boot commands
The list of U-boot commands can be accessed while in the U-Boot prompt. Type "help" or "?" for a complete listing of available commands.

Documentation on U-Boot may be found at http://www.denx.de/wiki/DULG/Manual .

Running Linux using U-Boot

Preparing the Linux images for U-Boot
Specific details on building the kernel for MicroBlaze are contained in the Build Kernel page.
Instructions to obtain a root filesystem can be found at the Build and Modify a Root File System page.

The kernel and the root filesystem must be wrapped with a U-Boot header in order for U-Boot to accept these files. The mkimage utility is used for this purpose, however, it is not part of the MicroBlaze GNU tools. It is part of U-Boot itself and if U-Boot has been compiled as specified on this page, it will be found under *<u-boot-xlnx>/build/tools/* .

When building linux.bin, as explained in the Build Kernel page, the kernel build process automatically creates an additional image, with the ".ub" suffix. *linux.bin.ub* is the Linux binary image wrapped with a U-Boot header. The Linux compilation process will automatically invoke mkimage, therefore, it is important to include the path to the U-Boot tools in the PATH environment variable.

The root filesystem also needs to be wrapped with a U-Boot header as follows:

```
mkimage -A microblaze -O linux -T ramdisk -C gzip -d ramdisk.image.gz uramdisk.im
```

Booting Linux
XMD or TFTP may be used to download Linux to the FPGA (memory). U-Boot can then use these files to boot Linux. Once the kernel, root filesystem, and device tree images are present in memory, the command to boot Linux is:

```
U-Boot> bootm <addr of kernel> <addr of rootfs> <addr of device tree blob (dtb)>
```

Note: Make sure the kernel and root filesystem images are wrapped by with the U-Boot header. The device tree blob does not need to be wrapped with the U-Boot header.

Creating the BSP required by U-Boot (Unverified)
This is old BSP and doesn't tested with latest u-boot.
To build U-Boot correctly, some information about the system will need to be provided to the U-Boot build tree. This information is generated by the U-Boot BSP.

uboot_bsp.tar.gz

First, download the U-Boot BSP source Once it has been installed and the project has been exported to SDK, the U-Boot BSP can be created for the system.

In order for the U-Boot BSP to generate the necessary settings for U-Boot, it needs to know which specific peripherals should be used for some corresponding tasks. These selections are made in the fields/drop-down menus located in the 'Settings' of the newly created U-Boot BSP. For example:

Name	Value
stdout	rs232_uart_1
stdin	rs232_uart_1
main_memory	ddr2_sdram
flash_memory	flash
ethernet	hard_ethernet_mac
timer	xps_timer_0

Once the U-Boot BSP has been compiled in SDK, the generated files, config.mk and xparameters.h, can be found at <sdk_workspace>/uboot_bsp_0/microblaze_0/libsrc/uboot_v*/ .

Downloading the source tree
The U-Boot source tree can be downloaded from the Xilinx Git server. The repository is **u-boot-xlnx** and the branch is **master**:

```
linux-host> git clone git://github.com/Xilinx/u-boot-xlnx.git
linux-host> cd u-boot-xlnx
```

Compiling U-Boot
The U-Boot compilation will use the definitions and flags defined in config.mk and xparameters.h . Import these two files generated by the U-Boot BSP in SDK into the U-Boot directory for MicroBlaze located at <u-boot-xlnx>/board/xilinx/microblaze-generic/ .

```
[[code]]
Xilinx's MicroBlaze GNU toolchain will be used to cross-compile U-Boot for MicroB
```

```
Now that U-Boot and the toolchain are configured correctly, we can cross-compile
```

```
linux-host> cd <u-boot-xlnx>
linux-host> export BUILD_DIR=$PWD/build
linux-host> make microblaze-generic_defconfig
linux-host> make
```


- Fetch Sources
- Build FSBL
- Build Device Tree Compiler (DTC)
- Build PMU Firmware
- Build Arm Trusted Firmware (ATF)
- (You are here) Build U-Boot
- Build and Modify a Root File System
- Build Device Tree Blob
- Build Linux Kernel
- Prepare Boot Image
- Prepare Boot Medium
- Setup a Serial Console
- Additional Information: Build Qt and Qwt Libraries

Related Links

- Zynq U-boot
- PowerPC U-boot
- Boot Zynq UltraScale+ MPSoC via JTAG

1