Getting Started With Embedded Linux – ZedBoard

1300 NE Henley Court, Suite 3
Pullman, WA 99163
(509) 334 6306 Voice | (509) 334 6300 Fax

Revision: January 13, 2013

Overview

Booting the Zynq-7000™ All Programmable SoC (Zynq AP SoC) from an SD card, or another form of compatible memory, requires that you first place four items onto your storage device. The four required items are the Linux file system (either Linaro or BusyBox), a Linux kernel image, a BOOT.BIN file, and a compiled device tree.

This guide provides instructions on how to generate these four items and on using them to boot the ZedBoard from an SD card. To complete these instructions, you must first ensure that you have a computer running a Linux distribution, a working knowledge of how to use the corresponding package manager to obtain software applications and libraries (e.g. yum for Fedora, or apt-get for Ubuntu), a 4GB or larger SD card, and a card reader.

Formatting the SD Card

Booting Linux on the ZedBoard from an SD card requires that you first set up the correct partitions on the SD card. You must format the first two partitions on the SD card to specific parameters. The first partition must have a FAT file system and be at least 1GB and the second partition must have an ext4 file system and be at least 3GB.

The second partition is only necessary when using the Linaro file system. However, Digilent Inc. recommends formatting your SD card with both partitions in case you decide to switch file systems in the future. Follow steps 1-4 on a Linux computer to properly format the SD card with both partitions.

Note: The shaded terminal display sections in this guide show operator input in bold characters.

1) Identify the SD card device node. Identify this node by making sure to remove the SD card from your Linux machine and then running `lsblk`.

```
[tinghui.wang@DIGILENT_LINUX ~]$ lsblk
NAME                          MAJ:MIN RM   SIZE RO TYPE MOUNTPOINT
sda                             8:0    0 465.8G  0 disk
├─sda1                          8:1    0   500M  0 part /boot
└─sda2                          8:2    0 465.3G  0 part
  ├─VolGroup-lv_root (dm-0) 253:0    0 455.5G  0 lvm  /
  └─VolGroup-lv_swap (dm-1) 253:1    0   9.8G  0 lvm  [SWAP]
sr0                            11:0    1   6.8G  0 rom
```

After first running `lsblk`, insert the SD card and run the command again.

```
[tinghui.wang@DIGILENT_LINUX ~]$ lsblk
NAME                          MAJ:MIN RM   SIZE RO TYPE MOUNTPOINT
sda                             8:0    0 465.8G  0 disk
├─sda1                          8:1    0   500M  0 part /boot
└─sda2                          8:2    0 465.3G  0 part
  ├─VolGroup-lv_root (dm-0)   253:0    0 455.5G  0 lvm  /
  └─VolGroup-lv_swap (dm-1)   253:1    0   9.8G  0 lvm  [SWAP]
sr0                            11:0    1   6.8G  0 rom
sdd                             8:48   1   7.5G  0 disk
└─sdd1                          8:49   1   3.7G  0 part /media/ZED_BOOT
```

A new line containing the SD card device node will appear the second time you run `lsblk`. In the example above, the SD card device node is `/dev/sdd`, highlighted in red.

2) Some distributions will automatically mount any partitions on an SD Card when you insert it. Input the `df` command to see if the SD card has any mounted partitions. If it does, ensure that you unmount these automatically mounted partitions before you repartition the disk.

```
[tinghui.wang@DIGILENT_LINUX ~]$ df
Filesystem           1K-blocks      Used Available Use% Mounted on
/dev/mapper/VolGroup-lv_root
                     470166952 316061992 130221800  71% /
tmpfs                  3988440       976   3987464   1% /dev/shm
/dev/sda1               495844     65557    404687  14% /boot
/dev/sdd1              3862528     10268   3852260   1% /media/ZED_BOOT
```

Call `umount` for each of the mounted partitions on your SD Card to remove them.

```
[tinghui.wang@DIGILENT_LINUX ~]$ sudo umount /media/ZED_BOOT/
[tinghui.wang@DIGILENT_LINUX ~]$ df
Filesystem           1K-blocks      Used Available Use% Mounted on
/dev/mapper/VolGroup-lv_root
                     470166952 316062016 130221776  71% /
tmpfs                  3988440       976   3987464   1% /dev/shm
/dev/sda1               495844     65557    404687  14% /boot
```

3) Once you have unmounted all of the partitions, you can begin to repartition the SD card with the `fdisk` tool. Open the SD card device using `fdisk` and issue command `p` to print the current SD card partition table.

```
[tinghui.wang@DIGILENT_LINUX ~]$ sudo fdisk /dev/sdd
[sudo] password for tinghui.wang:

WARNING: DOS-compatible mode is deprecated. It's strongly recommended to
         switch off the mode (command 'c') and change display units to
         sectors (command 'u').

Command (m for help): p

Disk /dev/sdd: 3965 MB, 3965190144 bytes
228 heads, 2 sectors/track, 16983 cylinders
Units = cylinders of 456 * 512 = 233472 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x00047708

   Device Boot      Start         End      Blocks   Id  System
/dev/sdd1               1       16978     3870720    b  W95 FAT32
```

Input the `d` command to delete any existing partitions. If only one partition exists, it will be selected automatically.

```
Command (m for help): d
Selected partition 1

Command (m for help): p

Disk /dev/sdd: 3965 MB, 3965190144 bytes
228 heads, 2 sectors/track, 16983 cylinders
Units = cylinders of 456 * 512 = 233472 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x00047708

   Device Boot      Start         End      Blocks   Id  System
```

Once you have deleted the existing partitions, you can create the new partitions with the `n` command. Create two primary partitions with these properties.

- Partition Number 1: A primary partition starting from the first cylinder with a size of 1GB.
- Partition Number 2: A primary partition starting from the next available cylinder that ideally takes up the remainder of the available space on the SD Card.

Use the commands in the following terminal display to create these two partitions.

Note: The system will set any prompt you leave blank to the default value.

```
Command (m for help): n
Command action
   e   extended
   p   primary partition (1-4)
p
Partition number (1-4): 1
First cylinder (1-16983, default 1): 1
Last cylinder, +cylinders or +size{K,M,G} (1-16983, default 16983): +1G

Command (m for help): p

Disk /dev/sdd: 3965 MB, 3965190144 bytes
228 heads, 2 sectors/track, 16983 cylinders
Units = cylinders of 456 * 512 = 233472 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x00047708

   Device Boot      Start         End      Blocks   Id  System
/dev/sdd1               1        4600     1048799   83  Linux

Command (m for help): n
Command action
   e   extended
   p   primary partition (1-4)
p
Partition number (1-4): 2
First cylinder (4601-16983, default 4601):
Using default value 4601
Last cylinder, +cylinders or +size{K,M,G} (4601-16983, default 16983):
Using default value 16983

Command (m for help): p

Disk /dev/sdd: 3965 MB, 3965190144 bytes
228 heads, 2 sectors/track, 16983 cylinders
Units = cylinders of 456 * 512 = 233472 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x00047708

   Device Boot      Start         End      Blocks   Id  System
/dev/sdd1               1        4600     1048799   83  Linux
/dev/sdd2            4601       16983     2823324   83  Linux
```

Once you make the required changes, use command `w` to write them to the SD card's partition table. Issuing command `w` will cause `fdisk` to automatically exit.

```
Command (m for help): w
The partition table has been altered!

Calling ioctl() to re-read partition table.
Syncing disks.
```

4) The final step to partitioning your SD card is creating the file systems. Format Partition Number 1 to FAT with the label "ZED_BOOT" and Partition Number 2 to EXT4 with the label "ROOT_FS". Use the utility `mkfs` to format your partitions.

```
[tinghui.wang@DIGILENT_LINUX ~]$ sudo mkfs -t vfat -n ZED_BOOT /dev/sdd1
mkfs.vfat 3.0.9 (31 Jan 2010)
[tinghui.wang@DIGILENT_LINUX ~]$ sudo mkfs -t ext4 -L ROOT_FS /dev/sdd2
mke2fs 1.41.12 (17-May-2010)
Filesystem label=ROOT_FS
OS type: Linux
Block size=4096 (log=2)
Fragment size=4096 (log=2)
Stride=0 blocks, Stripe width=0 blocks
176704 inodes, 705831 blocks
35291 blocks (5.00%) reserved for the super user
First data block=0
Maximum filesystem blocks=725614592
22 block groups
32768 blocks per group, 32768 fragments per group
8032 inodes per group
Superblock backups stored on blocks:
        32768, 98304, 163840, 229376, 294912

Writing inode tables: done
Creating journal (16384 blocks): done
Writing superblocks and filesystem accounting information: done

This filesystem will be automatically checked every 37 mounts or
180 days, whichever comes first.  Use tune2fs -c or -i to override.
```

The SD card should be ready for the Linux file system once you have correctly formatted the partitions.

The Linux File System

The ZedBoard currently supports two different Linux file systems, a BusyBox ramdisk and a Linaro Ubuntu distribution.

The BusyBox ramdisk is a very small file system that includes basic functionality and runs through RAM. BusyBox is non-persistent, which means it will not save any changes you make during your operating session after you power down the ZedBoard. (See Figure 1.)

```
[    1.070000] new mode: 720x480 858x525 27000
[    1.070000] new mode: 720x480 858x525 27000
[    1.070000] new mode: 1920x1080 2200x1125 74250
[    1.080000] mmcblk0: mmc0:e624 SU04G 3.69 GiB
[    1.080000] new mode: 1440x480 1716x525 27000
[    1.080000] new mode: 1440x480 1716x525 27000
[    1.090000] new mode: 640x480 800x525 25175
[    1.100000]  mmcblk0: p1
[    1.100000] drivers/gpu/drm/analog/analog_drm_fbdev.c:analog_drm_fbdev_probe[241]
[    1.120000] setting clock to: 74250
[    1.120000] raw_edid: d8ae1180 7
[    1.120000] Using YCbCr output
[    1.140000] Console: switching to colour frame buffer device 160x45
[    1.170000] fb0:  frame buffer device
[    1.170000] drm: registered panic notifier
[    1.170000] [drm] Initialized analog_drm 1.0.0 20110530 on minor 0
[    1.260000] EXT4-fs (ram0): couldn't mount as ext3 due to feature incompatibilities
[    1.310000] EXT4-fs (ram0): mounting ext2 file system using the ext4 subsystem
[    1.310000] EXT4-fs (ram0): warning: mounting unchecked fs, running e2fsck is recommended
[    1.320000] EXT4-fs (ram0): mounted filesystem without journal. Opts: (null)
[    1.320000] VFS: Mounted root (ext2 filesystem) on device 1:0.
[    1.330000] devtmpfs: mounted
[    1.330000] Freeing init memory: 152K
Starting rcS...
++ Mounting filesystem
++ Setting up mdev
++ Configure static IP 192.168.1.10
[    1.510000] GEM: lp->tx_bd ffdfa000 lp->tx_bd_dma 19bd7000 lp->tx_skb d8070480
[    1.520000] GEM: lp->rx_bd ffdfb000 lp->rx_bd_dma 19bd8000 lp->rx_skb d8070580
[    1.520000] GEM: MAC 0x00350a00, 0x00002201, 00:0a:35:00:01:22
[    1.530000] GEM: phydev d8b6f400, phydev->phy_id 0x1410dd1, phydev->addr 0x0
[    1.530000] eth0, phy_addr 0x0, phy_id 0x01410dd1
[    1.540000] eth0, attach [Marvell 88E1510] phy driver
++ Starting telnet daemon
++ Starting http daemon
++ Starting ftp daemon
++ Starting dropbear (ssh) daemon
++ Starting OLED Display
[    1.580000] pmodoled-gpio-spi [zed_oled] SPI Probing
++ Exporting LEDs & SWs
rcS Complete
zynq>
```

Figure 1. ***BusyBox boot output on a terminal connected to the ZedBoard UART port***

The Linaro file system is a complete Linux distribution based on Ubuntu. It includes a graphical desktop that displays via the onboard HDMI port. Linaro executes from a separate partition on the SD card, and all changes made are written to memory. The utility of Linaro is that it will save files even after you power down and reboot the ZedBoard.

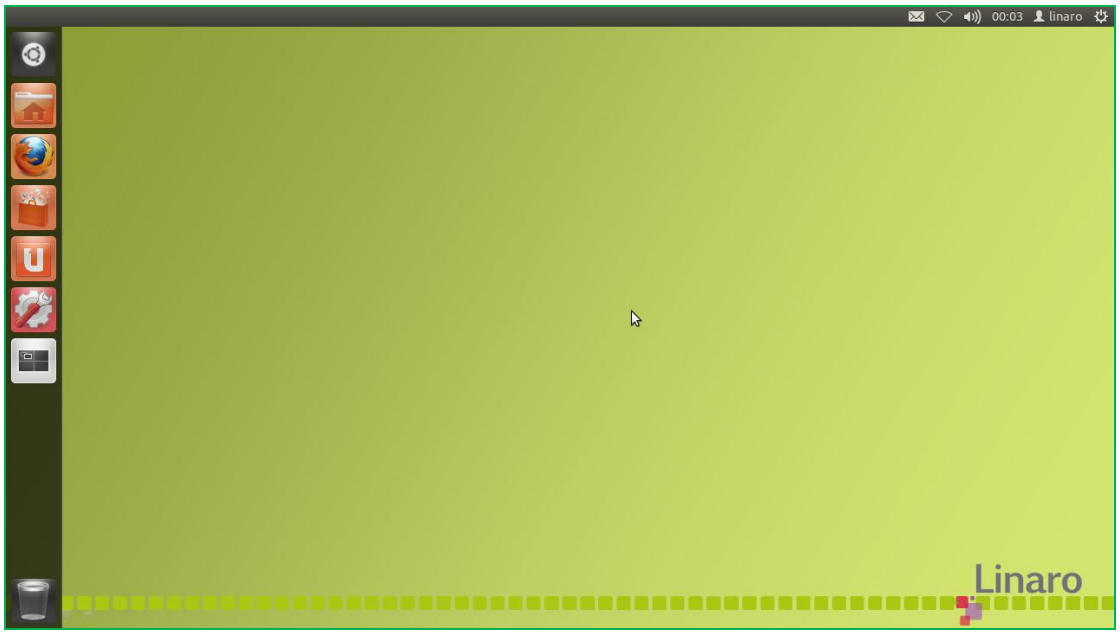

Figure 2. ***Linaro graphical desktop***

Before using your ZedBoard you will need to choose which of these two file systems would work best for your needs. Once you have selected a system, see the corresponding sub-headings below to prepare your removable storage device for booting Linux.

Using a BusyBox Ramdisk

You may find a prebuilt BusyBox ramdisk for the ZedBoard inside the Linux Hardware Design, available at www.digilentinc.com/zedboard. The ramdisk is found within the project at `ZedBoard_Linux_Design/sd_image/ramdisk8M.image.gz`. To use the prebuilt ramdisk, place the "ramdisk8M.image.gz" file on the FAT partition of the SD card. You are now ready to build the Linux kernel.

Digilent's prebuilt ramdisk uses source code that Xilinx provides online. See the Xilinx materials at: http://wiki.xilinx.com/zynq-rootfs for a detailed description of the ramdisk and how to create a custom system.

Using a Linaro File System

The first step in preparing the Linaro file system is to obtain the tarball of your preferred Linaro Ubuntu distribution. You can obtain these from Linaro at http://releases.linaro.org/ by clicking the desired version and then traversing to ubuntu/precise-images. Linaro provides several different Ubuntu builds, some of which are very lightweight and do not use a desktop. You can find a version that does contain a graphical desktop and that has been tested on the ZedBoard at: http://releases.linaro.org/12.09/ubuntu/precise-images/ubuntu-desktop/linaro-precise-ubuntu-desktop-20120923-436.tar.gz.

After downloading the tarball to your home directory, complete steps 1-6 to copy the file system to the ext4 partition on the SD Card.

1) Create a folder under `/tmp` named `linaro`, and copy the zipped Linaro image to it.

Note: We omitted the username from the command line in the terminal display to prevent word wrap.

```
[~]$ mkdir -p /tmp/linaro
[~]$ sudo cp linaro-precise-ubuntu-desktop-20120923-436.tar.gz /tmp/linaro/fs.tar.gz
[~]$ cd /tmp/linaro/
[linaro]$ ls
fs.tar.gz
```

2) Unpack the disk image using the `tar` command.

```
[tinghui.wang@DIGILENT_LINUX linaro]$ sudo tar zxf fs.tar.gz
[tinghui.wang@DIGILENT_LINUX linaro]$ ls
binary      fs.tar.gz
```

3) Insert the SD Card. Unmount any automatically mounted partitions by following the procedures this guide previously listed in step 2 of the "Formatting the SD Card" section.

4) Mount the SD Card to `/tmp/sd_ext4`. Make sure to replace the device node, highlighted red in the following terminal display, with the device node of the ext4 partition on your SD Card.

```
[tinghui.wang@DIGILENT_LINUX linaro]$ mkdir -p /tmp/sd_ext4
[tinghui.wang@DIGILENT_LINUX linaro]$ sudo mount /dev/sdd2 /tmp/sd_ext4
```

5) Use the command `rsync` to copy the root file system onto the SD card. This command will preserve those attributes that should remain unchanged.

```
[tinghui.wang@DIGILENT_LINUX linaro]$ cd binary/boot/filesystem.dir/
[tinghui.wang@DIGILENT_LINUX filesystem.dir]$ pwd
/tmp/linaro/binary/boot/filesystem.dir
[tinghui.wang@DIGILENT_LINUX filesystem.dir]$ sudo rsync –a ./ /tmp/sd_ext4
```

6) Unmount before removing the SD card to make sure all the files have been synchronized to it. Unmounting `/tmp/sd_ext4` may take several minutes, but you must wait to see that **umount** returns before removing the SD card.

```
[tinghui.wang@DIGILENT_LINUX filesystem.dir]$ sudo umount /tmp/sd_ext4
[tinghui.wang@DIGILENT_LINUX filesystem.dir]$
```

The Linaro file system should now be on the SD card and you are ready to build the Linux kernel.

Building the Linux Kernel

Digilent maintains a Linux source tree targeted to run on Digilent system boards. This repository contains custom drivers for onboard peripherals and attachable Pmods. Before building the kernel users must first download and install the ARM GNU tools from Xilinx. The installer and instructions for these Xilinx tools are available at: http://wiki.xilinx.com/zynq-tools.

1. After you install the ARM GNU tool chain, we recommend that you open the .bashrc file in your home folder and add the lines in the terminal display below.

   Note: You may need to change the path line, in red below, to identify where you installed the ARM GNU tools on your system.

```
PATH=~/CodeSourcery/Sourcery_CodeBench_Lite_for_Xilinx_GNU_Linux/bin:$PATH
export CROSS_COMPILE=arm-xilinx-linux-gnueabi-
```

   The addition of these lines causes the tool chain environment variables to be set each time the system opens a bash terminal.

2. Download the Linux kernel source code using `git`. You can obtain and install `git` using the package manager in your Linux distribution. After installing `git`, open a terminal and change to the directory where you would like to place the source. Then run the command in the terminal display below.

```
git clone https://github.com/Digilent/linux-digilent.git
```

3. After the download has completed, change to the `linux-digilent` directory. Run the command below to configure the kernel for the ZedBoard.

```
make ARCH=arm digilent_zed_defconfig
```

The command, `make ARCH=arm digilent_zed_defconfig`, will configure the kernel to work properly with the hardware on your board. To view or alter this default configuration run the following command.

```
make ARCH=arm menuconfig
```

The `make ARCH=arm menuconfig` command will open up a graphical interface for modifying the kernel. You must install the library "ncurses" on your system to successfully use menuconfig. Many operators commonly use this interface for selecting drivers built into the kernel and those built as loadable modules. Do not modify any of these settings right now.

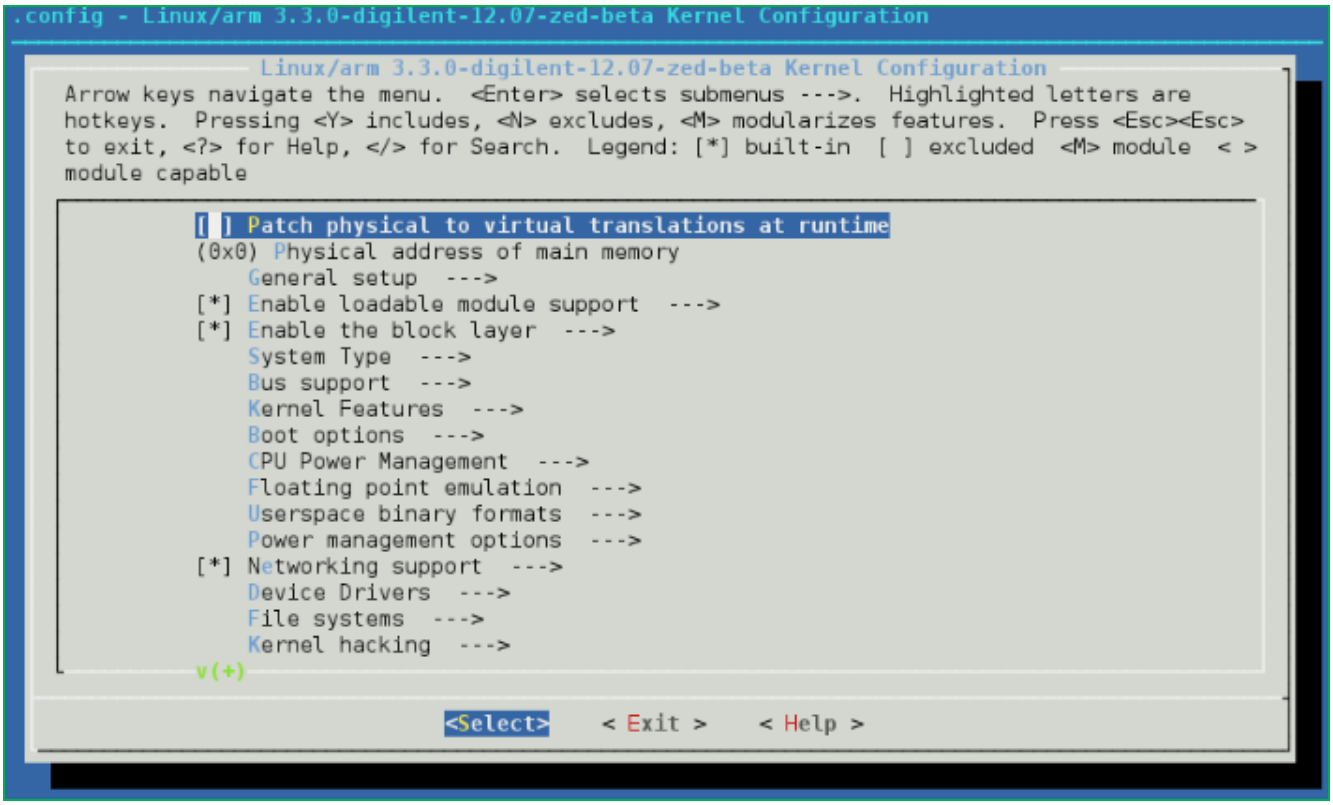

Figure 3. *`menuconfig` Interface for Configuring the Kernel*

4. After following steps 1-3 run the following command to build the kernel.

```
make ARCH=arm
```

If the build completes without errors, you will find the built kernel image (a single binary file named “zImage”) at `linux-digilent/arch/arm/boot/zImage`. Copy this file to the FAT partition of the SD Card.

Obtaining the BOOT.BIN File

The BOOT.BIN file is a container file for the various Xilinx specific files that initially configure the two sections of the Zynq AP SoC, the programmable logic and processing systems. This container also holds u-boot, a second-stage bootloader that is responsible for loading Linux.

Digilent distributes the source code for a Xilinx Embedded Design Kit (EDK) project that will configure the Zynq part on the ZedBoard in a manner that allows Linux to communicate properly with the onboard hardware. This project is called the “ZedBoard Linux Hardware Design” and can be obtained at: www.digilentinc.com/zedboard. Those interested in making changes to this design and building their own custom BOOT.BIN should view the included project guide in the doc folder. For the purposes of this getting started guide, copy the prebuilt BOOT.BIN from the sd_image folder to the FAT partition of the SD Card.

Compiling the Device Tree

The device tree is a data structure that describes the hardware present in your system to the Linux kernel. The tree lists devices as “nodes” that contain information needed for the corresponding driver to operate properly. The kernel parses through these nodes and initializes a driver for each of them during the boot up process.

The Digilent Linux repository contains a default device tree for the ZedBoard that corresponds with the Linux Hardware Design. You may find this default device tree at `linux-digilent/arch/arm/boot/digilent-zed.dts`.

Users must update the device tree to reflect any changes made to the Linux hardware design in Xilinx Platform Studio. Common changes that require an update to the device tree include, but are not limited to, changing the physical address of an IP core, changing the priority of an interrupt used by a device driver, and adding or removing an IP core.

In addition to describing the hardware, the device tree is also home to the boot arguments that configure the kernel at boot time. Boot arguments can be specified that instruct the kernel to do many different things. What should most concern you, is that this is also where the kernel is told what file system to load. This means that you will need to modify the digilent-zed.dts file to indicate which file system you are using.

If you are using a Linaro file system to boot the ZedBoard, open digilent-zed.dts in a text editor and replace the line containing the bootargs definition with the following code.

```
bootargs = "console=ttyPS0,115200 root=/dev/mmcblk0p2 rw earlyprintk rootfstype=ext4
rootwait devtmpfs.mount=0";
```

If you are using a ramdisk to boot the ZedBoard, replace bootargs with the following code.

```
bootargs = "console=ttyPS0,115200 root=/dev/ram rw initrd=0x800000,8M earlyprintk";
```

After you have correctly set the boot arguments, you can build the device tree with a tool called `dtc`. The kernel source includes the `dtc` tool, which you can find at `linux-digilent/scripts/dtc/dtc` after you build the kernel. Change to the `linux-digilent` directory and run the following command to build the device tree. You will need to change the input file, highlighted red in the terminal display below, to the modified digilent-zed.dts file location.

```
./scripts/dtc/dtc -I dts -O dtb -o ./devicetree.dtb ./digilent-zed.dts
```

The compiled device tree should now be at linux-digilent/devicetree.dtb. Copy this file to the FAT partition of the SD Card.

Booting the SD Card

Once you complete these guide instructions, the SD card will have everything it needs to boot Linux on the ZedBoard. Complete the procedures in steps 1-8 to test your SD card with the ZedBoard.

1. Insert the SD card into the ZedBoard.

2. Set the jumpers on the ZedBoard as follows:

    - MIO 6: set to GND
    - MIO 5: set to 3V3
    - MIO 4: set to 3V3
    - MIO 3: set to GND
    - MIO 2: set to GND
    - VADJ Select: Set to 1V8
    - JP6: Shorted
    - JP2: Shorted
    - All other jumpers should be left unshorted

3. Attach a computer running a terminal emulator to the UART port with a Micro-USB cable. Configure the terminal emulator with the following settings:

    - Baud: 115200
    - 8 data bits
    - 1 stop bit
    - no parity

4. Connect any peripherals you would like to use in Linux. If using a Linaro file system, we recommend that you connect a monitor to the HDMI port and a USB hub to the USB OTG port. You can then attach a mouse and keyboard to the USB hub.

5. Attach a 12V power supply to the ZedBoard and power it on.

6. Connect to the appropriate port in the terminal emulator. You should begin to see feedback from the boot process within a few seconds, depending on the speed of the SD card.

7. Wait for the boot process to complete. If using a BusyBox file system, you will know boot-up has completed when pressing return at the terminal presents you with a red "zynq>" prompt. (See Figure 1.) If you are operating with the Linaro file system, the attached monitor will display the Linaro desktop once the system boots up. (See Figure 2.)

8. You now have a complete Linux system running on the ZedBoard.

Additional Resources

Consult the following documents for additional information on designing embedded Linux systems for Digilent system boards.

- *Embedded Linux Development Guide*
  This document describes the differences between conventional Linux Development and Linux Development for Digilent system boards. It should be read by anyone who plans on tweaking the kernel or adding device drivers. The Embedded Linux Development Guide can be obtained from the embedded Linux product page on the Digilent website.

- *Embedded Linux Hands-on Tutorial – ZedBoard*
  This document walks the reader through the process of modifying the ZedBoard Linux Hardware Design to include additional hardware, making this hardware accessible to Linux by modifying the device tree, and finally designing a custom driver that brings the hardware's functionality up to the Linux user. It can be obtained from the ZedBoard product page on the Digilent website.

- *ZedBoard Linux Hardware Design Project Guide*
  This document describes the ZedBoard Linux Hardware Design, and walks the reader through the process of building all the sources required to generate the BOOT.BIN file. It is packaged along with the ZedBoard Linux Hardware Design, which can be obtained from the ZedBoard product page.

- *Linux Developer's Wiki*
  This web page contains an up to date list of hardware that is supported by the Digilent Linux repository and an FAQ section that addresses some issues you may run into while using the current version of the kernel. It also contains information on submitting patches for those who are interested in contributing code. You can find the Linux Developer's Wiki at: www.github.com/Digilent/linux-digilent/wiki.

Mouser Electronics

Authorized Distributor

Click to View Pricing, Inventory, Delivery & Lifecycle Information:

Digilent:
410-248P-KIT