Elph Training Exercises

Exercises

All the required images and code for Elph training are uploaded at sharepoint https://www.dropbox.com/sh/0liva3mj09mld78/AABgahs-y1oxM-ZjI289bQyLa?dl=0

Session 1 Exercises

Download all the images.tgz from the sharepoint under Images directory. Untar the same under Elph directory:
$ mkdir Elph
$ cd Elph
$ tar -xvf images.tgz
This should get MLO, u-boot.img, rd-ext2.bin, am335x-boneblack.dtb, zImage, user-mmc.txt, uEnv.txt uEnv-mmc.txt, rootfs.tgz under Images directory

Exercise 1: Booting up the board with pre-built images

The idea over here is to boot up the board with available images from the sharepoint.

1. Plug in the uSD card into Beaglebone Black and power up the board with USB cable. Make sure that the minicom is configured and you are able to target console on minicom.
2. Mount the first partition of uSD Card and transfer the images
   $ mount /dev/mmcblk0p1 /mnt (On board)
   $ cd Elph/Images
   $ scp zImage uEnv.txt rd-ext2.bin am335x-boneblack.dtb MLO u-boot.img usr-mmc.txt root@192.168.7.2:/mnt/
   $ umount /mnt
   $ reboot
3. Once the system is rebooted, it should display “Welcome to Embitude”

Exercise 2: Stopping at uboot

The objective is to boot-up the board to uboot (second stage bootloader)

1. Mount the first parition of uSD card and rename uEnv.txt to uEnv-org.txt
   $ mount /dev/mmcblk0p1 /mnt
   $ cd /mnt
   $ mv uEnv.txt uEnv-org.txt
   $ umount /mnt
2. Reboot the board
   $ reboot
3. If it still comes up to user space, next thing is to do the similar change for eMMC:
   $ sudo mount /dev/mmcblk1p1 /mnt
   $ cd /mnt/boot/
   $ mv uEnv.txt uEnv-org.txt
   $ sudo umount /mnt
   $ reboot
4. With this, the target board should not proceed beyond uboot. To stop at the uboot, quickly press the “SPACE” once the board is rebooted
5. Next step is to help uboot locate the kernel image and boot up with the same:

$ fatload mmc 0:1 $loadaddr zImage
$ bootz $loadaddr

6 Next thing is to try with zImage and dtb:
$ fatload mmc 0:1 $loadaddr zImage
$ fatload mmc 0:1 $fdt_addr_r am335x-boneblack.dtb
$ bootz $loadaddr - $fdt_addr_r

7 Next thing to set the bootargs
$ setenv bootargs console=ttyO0,115200n8
Repeat the above steps
Observe the behaviour

Exercise 3: Booting up with Ramdisk

The objective here is to provide the rootfs.This is where the ramdisk would be used

1 Reboot the board and press 'SPACE' during the boot-up. This would provide the u-boot prompt.
2 Load the uEnv-org.txt, export it to the shell environment & boot up:
$ fatload mmc 0:1 $loadaddr uEnv-org.txt
$ run importbootenv
$ run uenvcmd
This should boot-up the board with ramdisk

Exercise 4: Preparing the uSD Card for Beaglebone Black boot up

The objective here is to prepare the uSD Card by populating the various stuffs such as bootloaders, rootfs and so on. Prior to this the uSD needs to be partitioned and formatted.

Let's first preserve the existing images in eMMC

$ mount /dev/mmcblk0p1 /mnt (Mount the uSD first partition)
$ mkdir tmp
$ mount /dev/mmcblk1p1 tmp (Mount the eMMC first partition)
$ cp /mnt/* /root/tmp/opt/backup/
$ umount /mnt
$ umount tmp

Partitioning the uSD

1 Execute fdisk on uSD & delete all the existing partitions:
$ fdisk /dev/mmcblk0
$ 'u' (This would change the unit to cylinders)
$ 'd' (Deletes the partition)
2 Next thing is to set the uSD geometry. Its recomended to have the 255 heads, 63 sectors/tracker
$ 'x' (Switches to expert mode)
$ 'h' (Set the number of heads to 255)
$'s' (Set the number of sectors to 63)
$ 'c' (This needs to be calculated as (size in bytes)/(255 * 63 * 512))
$ 'r' (Return back to normal mode)
3 Let's create the partitions
$ 'n' (New partition)
$ 'p' (Primary partition and give the partition number as 1 and first cylinder as 4 and last cylinder as +128M)

Change the partition type as Windows LBA and make it bootable
$ ‘t’ and enter c
$ ‘a’ & select 1 (This would make partition 1 as bootable. Enter ‘p’ & notice the ‘*’ under boot)

4 Create the second primary partition with start cylinder as 20 and for last cylinder, enter +2G
5 Create the third primary partition with start cylinder as 281 and end cylinder as +1G. The last cylinder can be default (to use the entire disk) if its 4GB card.
6 Finally, write the partition table with ‘w’
7 Next step is to create the filesystem:
$ mkfs.vfat -F 32 -n boot /dev/mmcblk0p1 (Create FAT32 filesystem on first partition)
$ mkfs.ext2 -L FristRootfs /dev/mmcblk0p2 (Create Ext2 filesystem on second partiton)
$ mkfs.ext2 -L SecondRootfs /dev/mmcblk0p3 (Create Ext2 filesystem on third partiton)
8 Next step is to populate the partitions. The first partition should ideally have MLO, u-boot.img, uEnv.txt and am335x-boneblack.dtb
$ mount /dev/mmcblk0p1 /mnt
$ cd Elph/Images
$ scp MLO u-boot.img am335x-boneblack.dtb zImage root@192.168.7.2:/mnt/
$ scp uEnv-mmc.txt root@192.168.7.2:/mnt/uEnv.txt
$ umount /mnt
9 Next step is to populate the Rootfs
$ mount /dev/mmcblk0p2 /mnt
$ scp rootfs.tgz root@192.168.7.2:/mnt/
$ cd /mnt
$ tar -xvf rootfs.tgz
$ cd
$ umount /mnt
10 This should boot up with second partition

Exercise 5: Building the uboot

The objective here is to clone the u-boot repo and switch to v2019.04 version and build the same for bbb

1 Navigate to Elph directory and create a directory by name ‘Bootloaders’
$ cd Elph
$ mkdir Bootloaders
$ cd Bootloaders
2 Clone the git repo and check v2019.04
$ git clone https://github.com/u-boot/u-boot
$ cd u-boot
$ git checkout v2019.04 -b tmp
3 Next step is to fetch & apply the patch:
$ mkdir ../Patches
$ wget -c -P ../Patches/ https://raw.githubusercontent.com/eewiki/u-boot-patches/master/v2019.04/0001-am335x_evm-uEnv.txt-bootz-n-fixes.patch
$ patch -p1 < ../Patches/0001-am335x_evm-uEnv.txt-bootz-n-fixes.patch
4 Configure & build the uboot

$ make am335x_evm_defconfig
$ make CROSS_COMPILE=arm-linux-gnueabihf-

5 This would generate MLO and u-boot.img, which can be used to boot up the board

Exercise 6: Booting with the third (back-up) partition

The objective here is to prepare the back up partition which can be used for recovering the board in case it fails to boot up with default partition

1 First thing is to populate the contents of the third partition
$ mount /dev/mmcblk0p3 /mnt (on the target board)
$ cd Elph/Images
$ scp rootfs.tgz root@192.168.7.2:/mnt/
$ cd /mnt
$ tar -xvf rootfs.tgz (This should untar the rootfs contents)
$ Create a directory by name third (This is just to distinguish between 2nd and 3rd partition)
$ cd (Come out of /mnt)
$ umount /mnt

2 Next step is to modify uEnv.txt file. For this, navigate to the Images directory under Elph and modify the uEnv-mmc.txt
$ cd Elph/Images
Create a copy of uEnv-mmc.txt
$ cp uEnv-mmc.txt uEnv-third.txt
Modify to load the kernel image & dtb from the third partition and set the root=/dev/mmcblk0p2

3 Transfer the uEnv-third.txt to the board
$ mount /dev/mmcblk0p1 /mnt (On target)
$ scp uEnv-third.txt root@192.168.7.2:/mnt/

4 Next preserve the uEnv.txt to uEnv-old.txt and replace uEnv.txt
$ cp uEnv.txt uEnv-old.txt
$ cp uEnv-third.txt uEnv.txt
$ cd
$ umount /mnt

5 Let's reboot and check if its booted from 3rd partition by verifying if directory 'third' is present in /

Exercise 7: Booting with the serial

The serial booting is very handy for unbricking the board. This would be useful in case uboot and MLO have been corrupted or have been removed.

1 First thing is to erase the MLO from the eMMC, otherwise the board would always boot up with the one present in eMMC. The MLO is raw dumped at the offset 128K and uboot at offset 384KB. The back up for both is available at opt/backup/uboot
$ dd if=/dev/zero of=/dev/mmcblk1 bs=512 seek=256 count=256
$ dd if=/dev/zero of=/dev/mmcblk1 bs=512 seek=768 count=1024

2 Make sure that the uSD is removed as well. Next thing is to reset the board and we should get 'CCCC...'' on the serial console. The ROM code understand the Xmodem protocol which is supported by minicom as well. Press 'Ctrl + a', then release and then press 's' in minicom. This should give an options for 'Xmodem, Ymodem and Zmodem'. Let's select the xmodem. Next, it would prompt for file to be transferred. So, navigate to Elph/Bootloaders/u-boot/spl directory and select u-boot-spl.bin. This should start the transfer. If not, reset the board and it should start the transfer.

3 Once the X-loader is transferred, next thing is to transfer the u-boot. Press ‘Ctrl + a’ , then release and then press ‘s’. Select xmodem and select u-boot.bin from Bootloaders/u-boot/. This should transfer u-boot.bin.

4 Once the transfer is done, make sure to press the ‘space’ quickly to get the uboot prompt.

Exercise 8: Adding the command in u-boot

The idea over here is to add the custom command in the uboot

1 Get into the u-boot directory
$ cd Elph/Bootloaders/uboot

2 Make a copy of already existing command say led.c under cmd
$ cd cmd
$ cp led.c myprint.c

3 Modify the myprint.c as follows:
- Remove everything except do_led and U_BOOT_CMD
- Change the name of the function do_led to do_myprint
- Deleting everything in do_led, except the ‘return 0’
- Include the ‘printf’ statement above ‘return 0’

4 Modify the U_BOOT_CMD as below:

```
U_BOOT_CMD(
        myprint, 1, 1, do_myprint,
        "My printf",
        "My Test command"
);
```

5 Next thing is to modify the Kconfig to give out the menu option for our command. For this, modify the Kconfig to add the menu option as below:

```
config CMD_MYPRINT
        bool "MY Test Print"
        help
          Enable the Test printf
```

6 Next thing is to modify the Makefile to add the below line:
obj-$(CONFIG_CMD_MYPRINT) += myprint.o

7 Get into u-boot top level directory
$ make menuconfig
Search for MYPRINT and select this option

8 Rebuild the uboot
$ make CROSS_COMPILE=arm-linux-gnueabihf-

9 Check if myprint.o is generated under uboot/cmd/

10 Copy the u-boot.img to the first partition of the board:
$ mount /dev/mmcblk0p1 /mnt (on the board)
$ scp u-boot.img root@192.168.7.2:/mnt/ (On host)
$ sudo umount /mnt (on board)
Power-off the board
$ poweroff

11 Follow below steps to force a boot from uSD card:
- Power off the board & remove the usb power cable
- While keeping the switch S2 pressed, connect the usb power cable
- While booting up, press ‘Space’ key to get the uboot prompt.

Exercise 9: Adding the command to detect the switch S2 press

The idea over here is to add the custom command to get the switch status

1. Get into the u-boot directory
   $ cd Elph/Bootloaders/uboot
2. Modify board/ti/am335x/board.c to add the command for userbutton. The corresponding code (board.c) can be found under Code folder under Elph directory
3. Rebuild the uboot
   $ make CROSS_COMPILE=arm-linux-gnueabihf-
4. Copy the u-boot.img to the first partition of the board, reboot & stop at the uboot prompt:
   $ mount /dev/mmcblk0p1 /mnt (on the board)
   $ scp u-boot.img root@192.168.7.2:/mnt/ (On host)
   $ sudo umount /mnt (on board)
   $ reboot
   Press 'Space' to stop at the uboot prompt
5. $ userbutton (Should print the value of the switch)

Exercise 10: Changing the boot flow as per the switch

The objective of this exercise is to boot up with the third partition or ramdisk when the switch is pressed

Part (i) Create user.txt

1. Mount first partition and make a copy of uEnv.txt
   $ mount /dev/mmcblk0p1 /mnt
   $ cd /mnt
   $ cp uEnv-third.txt user.txt

Part (ii) U-boot changes

1. Get into the u-boot directory
   $ cd Elph/Bootloaders/uboot
2. Update BOOTENV_DEV_LEGACY_MMC in include/configs/am335x_evm.h to include the following (after gpio set 53 preferably):

```
"if userbutton; then " \
        "echo Setting bootenv to user.txt;" \
        "setenv bootenvfile user.txt;" \
        "setenv bootenv user.txt;" \
 "fi; " \
```

3. Rebuild the uboot
   $ make CROSS_COMPILE=arm-linux-gnueabihf-
4. Copy the u-boot.img to the first partition of the board, reboot & stop at the uboot prompt:
   $ mount /dev/mmcblk0p1 /mnt (on the board)
   $ scp u-boot.img root@192.168.7.2:/mnt/ (On host)
   $ sudo umount /mnt (on board)
   $ reboot
   Press 'S2" switch during the boot up and verify if the instructions in user.txt are executed

Exercise 11: Saving the environment in MMC

The idea over here is to save the environment at the beginning of the uSD. Ealier during the partitioning, 4 cylinders were left out for this purpose.

1. Get into the u-boot directory

$ cd Elph/Bootloaders/uboot

2. Modify the include/configs/am335x_evm.h to add the following:
   #define CONFIG_ENV_IS_IN_MMC   1
   #define CONFIG_SYS_MMC_ENV_DEV 0
   #define CONFIG_ENV_OFFSET   0x1000
3. Build the uboot
   $ make CROSS_COMPILE=arm-linux-gnueabihf-
4. Test it on the board by using the serial booting or by transferring it to the board using scp

Exercise 12: Building the Kernel

Building the Kernel for BBB

1. Get into the Elph Driver and create the folder by name OS
   $ cd Elph
   $ mkdir OS
2. Download the Kernel Source
   $ cd OS
   $ wget https://mirrors.edge.kernel.org/pub/linux/kernel/v4.x/linux-4.19.103.tar.gz
   $ tar -xvf linux-4.19.103.tar.gz
   $ cd linux-4.19.103/
3. Configure the Kernel. Download the configuration file (config-4.19.103.elph) from Images folder at Sharepoint
   $ cp config-4.19.103.elph.default .config
4. Add following in Makefile (Preferably at line no. 323, under ARCH ?= $(SUBARCH)
   - CROSS_COMPILE=arm-linux-gnueabihf-
   - ARCH=arm
5. Compile the Kernel and dtb
   $ make zImage dtbs
   This would generate the dtb & zImage
6. Transfer the kernel & dtb to the board
   $ mount /dev/mmcblk0p1 /mnt (On board)
   $ scp arch/arm/boot/zImage arch/arm/boot/dts/am335x-boneblack.dtb root@192.168.7.2:/mnt/
   $ umount /mnt
   $ reboot (On board)
   Verify the build time with ‘uname -a’

Exercise 13: Booting up with initramfs

Initramfs is the image with Rootfs & Kernel clubbed into the single image. In this exercise, the ramdisk image contents would be clubbed with kernel image and the board would be booted up with the same.

Part (i) Creating the Rootfs for Initramfs

1. Navigate to the Images directory under Elph & untar the contents of ramdisk
   $ cd Elph/Images
   $ mkdir RootfsInit
   $ sudo mount rd-ext2.bin /mnt
   Change the permissions of root directory & etc/shadow. This is needed to avoid using the ‘sudo’ for copying the contents.
   $ sudo chmod -R 0666 /mnt/root
   $ sudo chmod 0666 /etc/shadow

$ tar -C /mnt -cf - . | tar -C RootfsInit/ -xf – (This would copy the contents of /mnt into RootfsInit

Part (ii) Configure the Kernel for Initramfs

1. Configure the kernel with following
   $ make menuconfig
   Under the General setup->Initial RAM Filesystem and RAM disk, select Initramfs Source file(s)
   This would prompt for the Rootfs path, enter the complete path for RootfInit
2. Build the Kernel & transfer it to the board
   $ make zImage
   $ mount /dev/mmcblk0p1 /mnt (On the target)
   $ scp arch/arm/boot/zImage root@192.168.7.2:/mnt/
   $ umount /mnt

Part (iii) Update u-boot command file

Transfer uEnv-Initram.txt from the Images to the board:
$ mount /dev/mmcblk0p1 /mnt
$ scp uEnv-Initram.txt root@192.168.7.2:/mnt/user.txt
$ umount /mnt

Next, reboot the board and press switch S2 to boot up with the initramfs

Exercise 14: Booting up with Nfs

Part(i) Update the Kernel for NFS

1. Make sure that below options are added to support networking over USB device:
   CONFIG_USB_GADGET=y
   CONFIG_USB_MUSB_HDRC=y Driver for the USB OTG controller
   CONFIG_USB_MUSB_GADGET=y Use the USB OTG controller in device (gadget) mode
   CONFIG_USB_MUSB_DSPS=y
   CONFIG_AM335X_PHY_USB
   Find the ”USB Gadget precomposed configurations” menu and set it to static instead of module so that CONFIG_USB_ETH=y
   CONFIG_ROOT_NFS
2. Compile the kernel & Transfer it to the board
   $ make zImage
   $ mount /dev/mmcblk0p1 /mnt
   $ cd /mnt
   $ mv zImage zImage-org
   $ cd (To get out of /mnt)
   $ scp arch/arm/boot/zImage root@192.168.7.2:/mnt (on host)
   $ umount /mnt

Part (ii) Host set up for NFS

1. Install the nfs-server package package
   $ sudo apt install nfs-kernel-server
2. Untar the rootfs and export the rootfs over nfs for mounting
   $ cd Elph/Images
   $ mkdir Rootfs
   $ cd Rootfs
   $ tar -xvf ../rootfs.tgz (This would populate the Rootf)
   Add the following in /etc/exports (Open it with root privilges):
   <path to rootfs>/Rootfs
   192.168.7.2(rw,no_root_squash,no_subtree_check)
   Restart the nfs server

$ sudo /etc/init.d/nfs-kernel-server restart

Part (iii) Setting up the bootargs for nfs

Download uEnv-nfs.txt from the sharepoint (https://1drv.ms/u/s!Ao1TKHqoiM35hI94ftJpGh6ycCW_Ew?e=8Tb7zS). The file is uploaded under Images folder.

1. Update the nfsboot in uEnv-nfs.txt to include the path for the rootfs
2. Transfer the modified uEnv-nfs.txt to the board as user.txt. This would allow us to boot up with Nfs, when switch is pressed
   $ mount /dev/mmcblk0p1 /mnt (On target)
   $ scp uEnv-nfs.txt root@192.168.7.2:/mnt/user.txt
   $ umount /mnt (On target)
   $ reboot and make sure to press the Switch S2 during the uboot to boot up with NFS

Exercise 15: Swapping the LED functionality with DTB

The objective here is to make a small modification in DTB to swap the LEDs functionality

1. Navigate to the the kernel directory & modify arch/arm/boot/dts/am335x-bone-common.dtsi to swap the gpio pins (gpios property) for led2 & led3 nodes.
   $ cd Elph/OS/linux-4.19.103
   $ vi arch/arm/boot/dts/am335x-bone-common.dtsi
2. Compile the dtb & transfer it to the board
   $ make dtbs
   $ mount /dev/mmcblk0p1 /mnt (On target)
   $ scp arm/arm/boot/dts/am335x-boneblack.dtb root@192.168.7.2:/mnt/
   $ umount /mnt
   $ reboot
   Make sure to press the S2 switch during the boot up to select the user.txt