Steps for integrating a Linux kernel to the hardware design

The following are the important steps to be carried out to make the hardware design work along with a Linux kernel

1. Building the UBOOT
2. Generating Bit stream
3. Creating FSBL
4. Generating BOOT.BIN
5. Configuring and building the kernel
6. Configuting the dts
7. Formatting the SD card
8. Setting up the SD card
9. Testing the sobel filter application

1. Building the U-BOOT

The UBOOT is responsible for starting up the kernel and loading it into the device memory.

1. Retrieve U-BOOR from Xilinx repository

   git clone git://git.xilinx.com/u-boot-xlnx.git

   cd u-boot-xlnx

   git checkout -b xilinx-v14.4 xilinx-v14.4

2. Configuring the U-BOOT, to boot from the EXT4 file system in SD card

   Add the following lines in the “sdboot” section

   “sdboot=echo Copying Linux kernel from SD to RAM...RFS in ext4;” ****

   “mmcinfo;” ****

   “fatload mmc 0 0x3000000 ${kernel_image};” ****

   “fatload mmc 0 0x2A00000 ${devicetree_image};” ****

   “bootm 0x3000000 - 0x2A00000\0” ****

   Save and exit the editor.

3. Building the u-boot

   make distclean

   make zynq_zed_config

   make

4. The u-boot execuitable file will be gereated as **“u-boot”.** Rename it to as **“u-boot.elf”**

2. Generating the bit stream

The bit stream contains the programming logic of the hardware design for FPGA. It is generated by Xilinx XPS tool.

1. Source Xilinx ISE tool

   source <Xilinx ISE directory> / settings64.sh

2. The reference design for ZED board from Analog devices. This reference design is modified to include the sobel filter.

   /nfs/TUEIEDAprojects/SystemDesign/work/ipxact/vivin-richards/fpgahdl_xilinx/cf_adv7511_zed

3. Open Xilinx XPS and the open the project system.xmp in cf_adv7511_zed

4. Click on export design to SDK, which generated the bit stream and exports the project to SDK for software development

5. The bit stream is generated as system.bit

3. Creating FSBL

The First Stage Boot Loader is responsible for configuring the ARM Cortex A9 for basi requirements such as UART, clock system, reset and interrupt system. The FSBL then hands over the control to U-Boot for loading the Linux Kernel

1. In the exported design in SDK tool, create new application project and choose ZYNQ FSBL from the template.

2. After the build process is completed, the **zynq_fsbl.elf** is created.

4. Generating BOOT.BIN

The three files which we have generated before has to be merged together to a single file called BOOT.BIN.

1. In SDK tool click Xilinx tools → Create Boot Image

2. Enter a BIF file name

3. Add the following file which we have generated before in the same order.

    1. zynq_fsbl.bit
    2. system.bit
    3. u-boot.elf

The pregenerated hardware design files are present in

/nfs/TUEIEDAprojects/SystemDesign/work/ipxact/vivin-richards/boot_image

4. click on create image which generated the BOOT image file as **output.bin**. Rename it as **BOOT.BIN**

5. Configuring and building the kernel

The latest Linux kernel v3.14 is used here. The linux kernel has to be configured before building it, to include the desired drivers to the kernel. The pre-build linux kernel v 3.14 is present in

/nfs/TUEIEDAprojects/SystemDesign/work/ipxact/vivin-richards/ubuntu

1. Download the Linux kernel.

    git clone git://github.com/analogdevicesinc/linux.git ubuntu

    cd ubuntu

    git checkout xcomm_zynq

2. The configuration file for the kernel in found in /arch/arm/configs/zynq_xcomm_adv7511_defconfig

The following are the important configurations for providing the kernel support for mouse, keyboard and web- camera interface

 # Input device support

CONFIG_INPUT=y

CONFIG_INPUT_FF_MEMLESS=y

CONFIG_INPUT_SPARSEKMAP=y

 # Mouse and keyboard interfaces

CONFIG_INPUT_MOUSEDEV=y

CONFIG_INPUT_MOUSEDEV_PSAUX=y

CONFIG_INPUT_MOUSEDEV_SCREEN_X=1024

CONFIG_INPUT_MOUSEDEV_SCREEN_Y=768

CONFIG_INPUT_EVDEV=y

CONFIG_INPUT_KEYBOARD=y

CONFIG_KEYBOARD_ATKBD=y

CONFIG_MOUSE_PS2=y

CONFIG_MOUSE_PS2_ALPS=y

CONFIG_MOUSE_PS2_LOGIPS2PP=y

CONFIG_MOUSE_PS2_SYNAPTICS=y

CONFIG_MOUSE_PS2_CYPRESS=y

CONFIG_MOUSE_PS2_TRACKPOINT=y

 # Video and camera support

CONFIG_MEDIA_CAMERA_SUPPORT=y

CONFIG_MEDIA_CONTROLLER=y

CONFIG_VIDEO_DEV=y

CONFIG_VIDEO_V4L2_SUBDEV_API=y

CONFIG_VIDEO_V4L2=y

CONFIG_VIDEOBUF2_CORE=y

CONFIG_VIDEOBUF2_MEMOPS=y

CONFIG_VIDEOBUF2_DMA_CONTIG=y

CONFIG_VIDEOBUF2_VMALLOC=y

CONFIG_MEDIA_USB_SUPPORT=y

CONFIG_USB_VIDEO_CLASS=y

CONFIG_USB_VIDEO_CLASS_INPUT_EVDEV=y

CONFIG_V4L_PLATFORM_DRIVERS=y

CONFIG_VIDEO_AXI_HDMI_RX=y

4. After editing the configuration file, build the kernel as follows. Make sure the xilinx tools are sourced before building the kernel, as the build process uses the arm cross compiler libraries

export CROSS_COMPILE=arm-xilinx-linux-gnueabi-

make ARCH=arm distclean

make ARCH=arm zynq_xcomm_adv7511_defconfig

make ARCH=arm

cd arch/arm/boot

gzip zImage

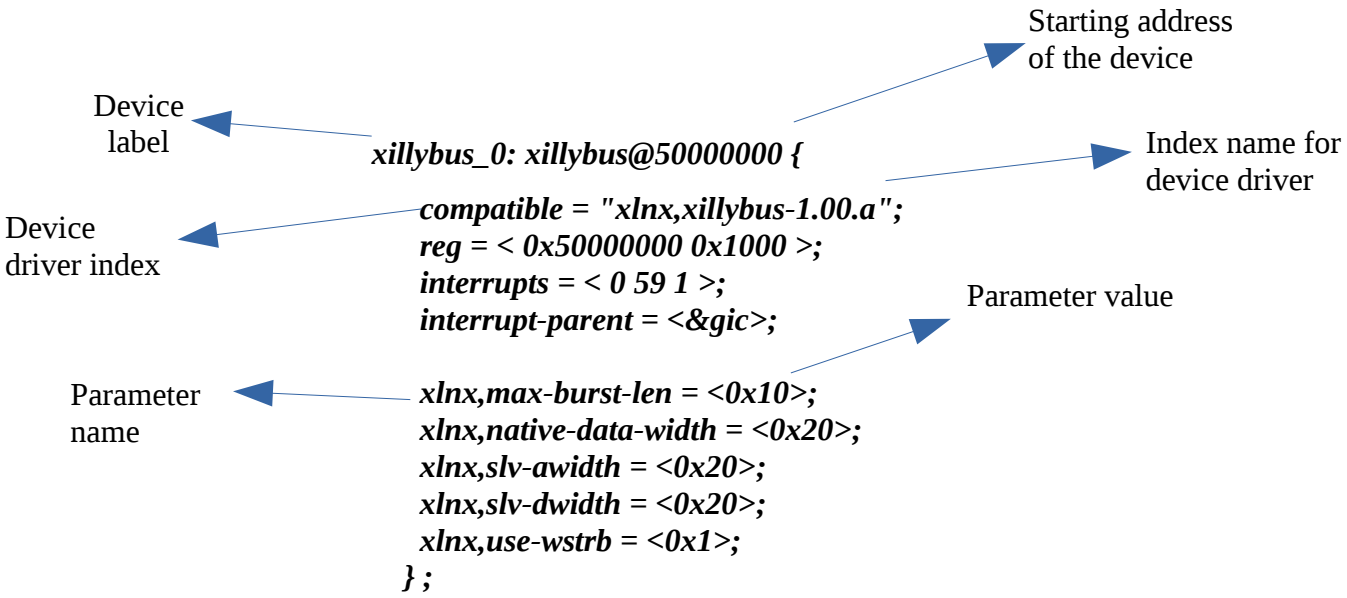
mkimage -A arm -a 0x8000 -e 0x8000 -n 'Linux kernel' -T kernel -d zImage.gz uImage

5. The kernel image file is found in **/arch/arm/boot/uImage**

6. Configuring the devicetree file

1. The kernel has to understand the hardware components present in the design. This information is provided by devicetree source files (.dts). The default devicetree file for zed board is found in **/arch/arm/boot/dts/zynq-zed-adv7511.dts**

2. The device tree file has to be modified according to our design. The following is the description of the syntax of a .dts file



3. The modified device tree is found here.

4. Build the device tree blob as follows

make ARCH=arm zynq-zed-adv7511.dtb

5. The .dtb file is generated in **arch/arm/boot/dts/zynq-zed-adv7511.dts**

7. Formatting the sd card

The SD card should be formatted to two partitions of ext4 and FAT32 format. EXT4 format holds the linaro file system and FAT32 holds the kernel image, boot file and device tree file.

1. Open gparted tool to format the SD card

2. Format the SD card for **40 MB** size of the format **FAT32**. Label this partition as **“boot”**

3. The remaining part of the SD card is formatted as **EXT4.** Label this partition as **“rootfs”**

8. Setting up the SD card

1. The “boot” partition contains the files necessary for booting up the linux kernel.

2. Copy the three files **BOOT.BIN, uImage, zynq-zed-adv7511.dtb** generated in steps 4,5 and 6 respectively. The pre-generated boot files are present in

   /nfs/TUEIEDAprojects/SystemDesign/work/ipxact/vivin-richards/sd_image_working

3. Rename zynq-zed-adv7511.dtb files as **devicetree.dtb**

4. Download the linaro file system

wget http://releases.linaro.org/11.12/ubuntu/oneiric- images/ubuntu-desktop/linaro-o-ubuntu-desktop-tar-20111219- 0.tar.gz

5. *Extract the linaro file system to “**rootfs”** partition of the SD card.*

sudo tar --strip-components=3 -C /media/rootfs -xzpf linaro-o-ubuntu-desktop-tar-20111219-0.tar.gz binary/boot/filesystem.dir

6. *Safely remove the SD card.*

9. Testing the sobel filter application

1. Copy the sobel filter application software from /nfs/TUEIEDAprojects/SystemDesign/work/ipxact/vivin-richards/camera_sw/test_app to the **rootfs** filesystem in SD card in **/home/linaro/** folder
2. Boot the Zed board from SD card.
3. Build the test application as follows

   cmake .

   Make clean

   make

   sudo ./camera

Note : Incase the camera executable file is already present in the workspace, it will cause error during the build process. So delete the camera exeecutable file first and then build it again.