My Notes

Mohammad S. Sadri

Xilinx Zynq Boot Linux Over Network

Hi,

Here I briefly describe how you can load the linux kernel and the rootfs image over the network to your ZED, ZC-702 and ZC-706 boards.

Basically, you can load every thing to your board over JTAG cable using the XMD command console. When every thing got loaded into the board memory you can start the execution.

Here is an example sequence of commands with which you can load and run the linux kernel only using JTAG. inside XMD you type:

```
# first configure the PL , if you want!
fpga -f ../implementation/system.bit

# Then connect to the ARM debugging hardware
connect arm hw
rst -debug_sys

# Initialize the PS of ZYNQ. One easy way to do this is to create an XPS project for your board and
# then to Export your design to SDK. Through this procedure the ps7_init.tcl file will be created
# which you can use for initializing the PS7.
source ../SDK/SDK_Export/hw/ps7_init.tcl
ps7_init

# This is very important if you have custom IPs added to the system which talk to the PS
# Through the AXI interface. In this case you should enable AXI level shifters.
# unlock write protection
mwr 0xf8000008 0xdf0d
# make sure all axi level shifters are enabled
mwr 0xf8000900 0xf
# lock write protection
mwr 0xf8000004 0x767b

# Now Load u-boot into memory, run it, make sure it has executed and done the initial steps
dow u-boot
con

# stop u-boot, and load the rest of the system
exec sleep 1
stop

# Linux kernel
dow -data ../linux-xlnx/arch/arm/boot/zImage 0x8000
exec sleep 1

# Ramdisk – this is the rootfs image
dow -data ramdisk8M.image.gz 0x800000
exec sleep 1

# compiled device tree
dow -data ../linux-xlnx/arch/arm/boot/dts/zynq-zc702.dtb 0x1000000
exec sleep 1
con
exit

# the end!
```

Now, the point is running the above script each time may take a while and we need a faster method. Obviously one solution is to create the BOOT.ini file and put it on the flash (SDCard) and boot the system over the SD Card. However, what if we wanted to change some thing (e.g. the bitstream) frequently?

One solution is to boot the whole linux over network. Using dow, you load the uboot into the memory only. And then uboot loads the zImage and ramdisk over the network and start the linux kernel. This can be faster.

For this you will need a tftp server. tftpd-hpa is a good choice for ubuntu. When you install it, it is almost ready. In this case, the script that XMD executes is the following:

```
fpga -f ../bit/system.bit
connect arm hw
rst -debug_sys
source ../xps/SDK/SDK_Export/hw/ps7_init.tcl
ps7_init
mwr 0xf8000008 0xdf0d
mwr 0xf8000900 0xf
mwr 0xf8f02208 0xf
mwr 0xf8f02204 0x17
mwr 0xf8f02200 0x1
mwr 0xf8000004 0x767b
```

```
dow u-boot
con
```

As we see, after running u-boot, our work here is done and we don't need to do any further action using XMD. Instead we need a set of commands to be executed by u-boot. For this purpose, I create a u-boot script, as following:

```
setenv ipaddr BOARD_IP
tftp 0x3000000 SERVER_IP:uImage
tftp 0x2000000 SERVER_IP:uramdisk.image.gz
tftp 0x2A00000 SERVER_IP:zynq-zc706.dtb
bootm 0x3000000 0x2000000 0x2A00000
```

Now , I compile this u-boot script, and make a .img file, which u-boot understands and can run. mkimage is a utility created inside the tools folder inside your u-boot when you compiled u-boot. add it to you path.

```
mkimage -T script -C none -n 'Boot File' -d u_boot_script a.img
```

This commands reads the u_boot_script file (which contains the above mentioned lines) and generates a.img

Now, I need to copy all of these files to my /var/lib/tftpboot folder, where my tftp server looks to serve files to incoming requests.

copy these files to tftpboot folder : a.img , uImage , uramdisk.image.gz, zynq-zc702.dtb
change the ownership of these files to nogroup:nobody : inside the tftpboot folder execute:
chown nobody:nogroup *

(And the ownership of the tftpboot folder itself should be root:nogroup)

Now, on your local computer install tftp client and test your tftpserver and make sure it works.

Now, I want u-boot to automatically, load the a.img file from the server and execute it when it comes up. So, I need to make a modification in the source code of u-boot and add my favorie initial boot commands there. For the ZYNQ the file in u-boot which contains initial commands and default values for variables is here:

/u-boot-xlnx/include/configs/zynq_common.h

In this file change the following variables:

```
CONFIG_IPADDR  <put BOARD_IP>
CONFIG_SERVERIP <put SERVER_IP>CONFIG_BOOTCOMMAND "tftp 0x100000 a.img; source 0x100000"
```

Re-compile the u-boot. Now your u-boot has every thing needed to automatically load and execute a.img. And as we know inside a.img we load the linux kernel and ramdisk and device tree and start it.
That is it! You can now bring up your system much faster than before.

Finally, I should note that, there is another possibility:

You bring up the Linux on the PS , then, through the PS you re-configure the PL with the updated new bitstream when ever you like. This is another possibility which well personally I have never tried it "yet!".

Mo.

Appendix

Creating the .dtb file: There is a complete set of dts files inside the /arch/arm/boot/dts folder of Linux kernel source. For the board that you have, find the suitable .dts file. e.g. for ZC-706, we see zynq-zc706.dts

Now there is device tree compiler inside the linux-xlnx/scripts/dtc/ folder, use it and create the compiler dtb file. Before that, you can edit the dts file and add whatever customization that you want to it. e.g. one very useful one is to add in the bootargs line the "ip=111.222.333.444" statement (if your board has a static ip address).

in the dtc folder:
```
sudo ./dtc -I dts -O dtb -o /var/lib/tftpboot/zynq-zc706.dtb ../../arch/arm/boot/dts/zynq-zc706.dts
```

And now you have the updated dtb file.

This entry was posted in Uncategorized on November 14, 2013 [http://googoolia.com/mynotes/2013/11/14/xilinx-zynq-boot-linux-over-network/] .