ECEN 749 LAB REPORT

Excercise #4: Linux boot-up on ZYBO board via SD Card

Tanmay Verma

10.06.2017

INTRODUCTION

The purpose of this lab is to boot Processing System of Zybo board with Linux. The multiplier IP developed as part of previous design is incorporated in the design and must also be included in “device tree blob” for the bootloader.

PROCEDURE

The following steps were carried out:

1. A new block design was created in the project.
2. In the diagram, ‘ZYNQ7 Processing System’ IP was added.
3. Provided xml file was used to re-configure the PS IP.
4. The block automation was run.
5. In re-customize IP following Peripheral I/O pins were enabled : SD 0, UART 1 and TTC 0.
6. The “multiply” IP designed in the previous lab was imported and added to the block design in the project.
7. The connection automation was run.
8. A HDL wrapper was created for the design and bitstream was generated.
9. Next, the U-Boot was configured and build for our target platform: Zybo-zynq using the cross-compile tools provided by vivado.
   > make CROSS_COMPILE=arm-xilinx-linux-gnueabi- zynq_zybo_config
   > make CROSS_COMPILE=arm-xilinx-linux-gnueabi-
10. The generated u-boot file was renamed with .elf extension.
11. Next the design was exported from vivado to SDK along with the bitstream.
12. A new Application with Zynq FSBL template was created in Vivado.
13. The whole project was compiled.
14. The Zynq boot image**(boot.BIN)** was created with the following partitions: FSBL(bootloader), bitstream(datafile) and u-boot(datafile).
15. Next, Linux was configured and compiled.
    make ARCH=arm CROSS_COMPILE=arm-xilinx-linux-gnueabi-
    xilinx_zynq_defconfig
    make ARCH=arm CROSS_COMPILE=arm-xilinx-linux-gnueabi-
16. The obtained zImage was unzipped using u-boot tools and **uImage** is created.

17. The dts file at arch/arm/boot/dts/zynq-zybo.dts was modified to include an entry for the multpily IP which was added at custom over Zynq PS.
    multiply {
    compatible = “ecen449,multiply”;
    reg = <0x43C00000 0x10000>;
    };
18. The modified .dts file was converted to **.dtb**
    ./scripts/dtc/dtc -I dts -O dtb -o ./devicetree.dtb arch/arm/boot/dts/zynq-zybo.dts
19. The ramdisk file **uramdisk.image.gz** was created using ramdisk file ad mkimage command(u-boot tools) to wrap the headers.
    ./u-boot/tools/mkimage -A arm -T ramdisk -c gzip -d ./ramdisk8M.image.gz uramdisk.image.gz
20. BOOT.bin, uImage, uramdisk.image.gz and devicetree.dtb files were copied to the SD card .
21. The board JP5 was configured to boot from the SD card. pico com was setup to monitor the activity and interact with the board.
    >picocom -b 115200 -r -l /dev/ttyUSB1
22. SD card was plugged in the board and PS-SRST was pressed to initiate the boot sequence.

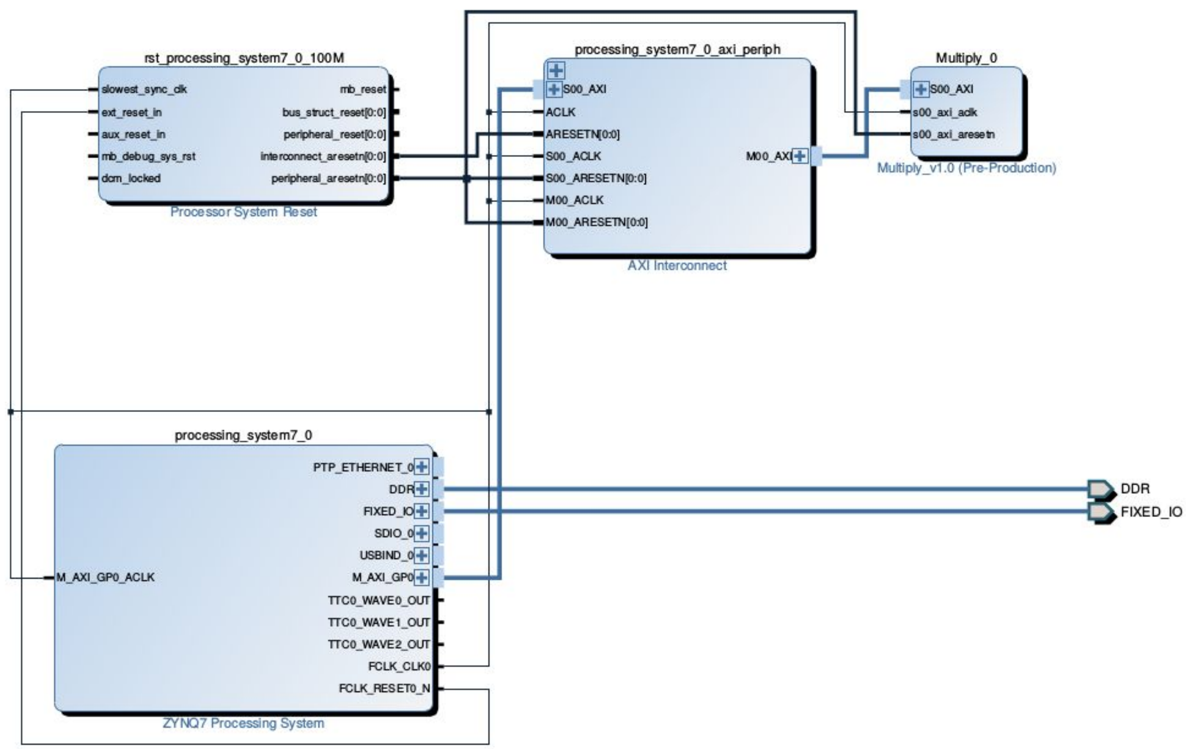
RESULTS

The following results were accomplished through this exercise. Using Vivado a Zynq(ARM Cortex A9) based microprocessor system suitable for running Linux was build, and the Linux kernel based on the specification of custom microprocessor system was configured and compiled. The bit stream with FSBL(First Stage Boot Loader) and u-boot(Universal Boot Loader) were used to create Zynq Boot Image. FSBL initializes the processing System(PS) with configuration data and initializes u-boot. U-boot is the boot loader that holds the instructions to boot the Linux Kernel. RAMDISK, a temporary file system that is mounted during Kernel boot, was configured as well. The device tree file was modified to let the bootloader and OS be aware of the custom IP added to the design. All these files were copied to the SD card to boot Linux on ZYBO board.

CONCLUSION

The exercise taught how to boot an Operating system on Zybo board Processing System along with an added custom IP..

Design(Same as Lab3)



Verilog Code for Custom IP(Same as Lab3)

Only the User Logic portion is shown as requested: all statement s writing to slv_reg3 were commented out.

```
// Add user logic here


   reg [0:C_S_AXI_DATA_WIDTH-1] tmp_reg;
   always @(posedge S_AXI_ACLK) begin
        if(S_AXI_ARESETN == 1'b0) begin        //Resets the output and temp register when reset is enabled.
        slv_reg2 <= 0;
        tmp_reg <= 0;
        end
        else begin
        tmp_reg <= slv_reg0 * slv_reg1;        //calculates the value of product.
        slv_reg2 <= tmp_reg;
        end
        end
   // User logic ends
```

DTS File Code(zynq-zybo.dts)

```
.
.
.
        } ;
        ps7_xadc: ps7-xadc@f8007100 {
                clocks = <&clkc 12>;
                compatible = "xlnx,zynq-xadc-1.00.a";
                interrupt-parent = <&ps7_scugic_0>;
                interrupts = <0 7 4>;
                reg = <0xf8007100 0x20>;
        } ;
        multiply {
                compatible = "ecen449,multiply";
                reg = <0x43C00000 0x10000>;
        } ;
        } ;
} ;
```

The code in bold is added one.

PICOCOM OUTPUT(the boot sequence)

```
 U-Boot 2014.01 (Sep 29 2017 - 16:22:39)

I2C:   ready
Memory: ECC disabled
DRAM:  512 MiB
MMC:   zynq_sdhci: 0
spi_setup_slave: No QSPI device detected based on MIO settings
SF: Failed to set up slave
*** Warning - spi_flash_probe() failed, using default environment

In:       serial
Out:   serial
Err:   serial
Net:   Gem.e000b000
Hit any key to stop autoboot:  0
Device: zynq_sdhci
Manufacturer ID: 3
OEM: 5344
Name: SL08G
Tran Speed: 50000000
Rd Block Len: 512
SD version 3.0
High Capacity: Yes
Capacity: 7.4 GiB
Bus Width: 4-bit
reading uEnv.txt
** Unable to read file uEnv.txt **
Copying Linux from SD to RAM...
reading uImage
3447864 bytes read in 302 ms (10.9 MiB/s)
reading devicetree.dtb
7446 bytes read in 14 ms (518.6 KiB/s)
reading uramdisk.image.gz
3693174 bytes read in 323 ms (10.9 MiB/s)
## Booting kernel from Legacy Image at 03000000 ...
   Image Name:   Linux-3.18.0-xilinx
   Image Type:   ARM Linux Kernel Image (uncompressed)
   Data Size:    3447800 Bytes = 3.3 MiB
   Load Address: 00008000
```

```
   Entry Point:  00008000
   Verifying Checksum ... OK
## Loading init Ramdisk from Legacy Image at 02000000 ...
   Image Name:
   Image Type:   ARM Linux RAMDisk Image (gzip compressed)
   Data Size:    3693110 Bytes = 3.5 MiB
   Load Address: 00000000
   Entry Point:  00000000
   Verifying Checksum ... OK
## Flattened Device Tree blob at 02a00000
   Booting using the fdt blob at 0x2a00000
   Loading Kernel Image ... OK
   Loading Ramdisk to 1f7aa000, end 1fb2fa36 ... OK
   Loading Device Tree to 1f7a5000, end 1f7a9d15 ... OK

Starting kernel ...

Booting Linux on physical CPU 0x0
Linux version 3.18.0-xilinx (tanmay2592@lin03-424cvlb.ece.tamu.edu) (gcc version 4.9.1
(Sourcery CodeBench Lite 2014.11-30) ) #1 SMP PREEMPT Fri Sep 29 17:10:41 CDT 2017
CPU: ARMv7 Processor [413fc090] revision 0 (ARMv7), cr=18c5387d
CPU: PIPT / VIPT nonaliasing data cache, VIPT aliasing instruction cache
Machine model: Xilinx Zynq
cma: Reserved 16 MiB at 0x1e400000
Memory policy: Data cache writealloc
PERCPU: Embedded 10 pages/cpu @5fbd3000 s8768 r8192 d24000 u40960
Built 1 zonelists in Zone order, mobility grouping on.  Total pages: 130048
Kernel command line: console=ttyPS0,115200 root=/dev/ram rw earlyprintk
PID hash table entries: 2048 (order: 1, 8192 bytes)
Dentry cache hash table entries: 65536 (order: 6, 262144 bytes)
Inode-cache hash table entries: 32768 (order: 5, 131072 bytes)
Memory: 492632K/524288K available (4650K kernel code, 258K rwdata, 1616K rodata, 212K
init, 219K bss, 31656K reserved, 0K highmem)
Virtual kernel memory layout:
        vector  : 0xffff0000 - 0xffff1000   (   4 kB)
        fixmap  : 0xffc00000 - 0xffe00000   (2048 kB)
        vmalloc : 0x60800000 - 0xff000000   (2536 MB)
        lowmem  : 0x40000000 - 0x60000000   ( 512 MB)
        pkmap   : 0x3fe00000 - 0x40000000   (   2 MB)
        modules : 0x3f000000 - 0x3fe00000   (  14 MB)
        .text : 0x40008000 - 0x40626b1c   (6267 kB)
        .init : 0x40627000 - 0x4065c000   ( 212 kB)
        .data : 0x4065c000 - 0x4069cb60   ( 259 kB)
```

```
        .bss : 0x4069cb60 - 0x406d3a78   ( 220 kB)
Preemptible hierarchical RCU implementation.
   Dump stacks of tasks blocking RCU-preempt GP.
   RCU restricting CPUs from NR_CPUS=4 to nr_cpu_ids=2.
RCU: Adjusting geometry for rcu_fanout_leaf=16, nr_cpu_ids=2
NR_IRQS:16 nr_irqs:16 16
L2C-310 erratum 769419 enabled
L2C-310 enabling early BRESP for Cortex-A9
L2C-310 full line of zeros enabled for Cortex-A9
L2C-310 ID prefetch enabled, offset 1 lines
L2C-310 dynamic clock gating enabled, standby mode enabled
L2C-310 cache controller enabled, 8 ways, 512 kB
L2C-310: CACHE_ID 0x410000c8, AUX_CTRL 0x76360001
ps7-slcr mapped to 60804000
zynq_clock_init: clkc starts at 60804100
Zynq clock init
sched_clock: 64 bits at 325MHz, resolution 3ns, wraps every 3383112499200ns
ps7-ttc #0 at 60806000, irq=43
Console: colour dummy device 80x30
Calibrating delay loop... 1292.69 BogoMIPS (lpj=6463488)
pid_max: default: 32768 minimum: 301
Mount-cache hash table entries: 1024 (order: 0, 4096 bytes)
Mountpoint-cache hash table entries: 1024 (order: 0, 4096 bytes)
CPU: Testing write buffer coherency: ok
CPU0: thread -1, cpu 0, socket 0, mpidr 80000000
Setting up static identity map for 0x467598 - 0x4675f0
CPU1: Booted secondary processor
CPU1: thread -1, cpu 1, socket 0, mpidr 80000001
Brought up 2 CPUs
SMP: Total of 2 processors activated.
CPU: All CPU(s) started in SVC mode.
devtmpfs: initialized
VFP support v0.3: implementor 41 architecture 3 part 30 variant 9 rev 4
regulator-dummy: no parameters
NET: Registered protocol family 16
DMA: preallocated 256 KiB pool for atomic coherent allocations
cpuidle: using governor ladder
cpuidle: using governor menu
hw-breakpoint: found 5 (+1 reserved) breakpoint and 1 watchpoint registers.
hw-breakpoint: maximum watchpoint size is 4 bytes.
zynq-ocm f800c000.ps7-ocmc: ZYNQ OCM pool: 256 KiB @ 0x60880000
vgaarb: loaded
SCSI subsystem initialized
```

usbcore: registered new interface driver usbfs
usbcore: registered new interface driver hub
usbcore: registered new device driver usb
media: Linux media interface: v0.10
Linux video capture interface: v2.00
pps_core: LinuxPPS API ver. 1 registered
pps_core: Software ver. 5.3.6 - Copyright 2005-2007 Rodolfo Giometti <giometti@linux.it>
PTP clock support registered
EDAC MC: Ver: 3.0.0
Advanced Linux Sound Architecture Driver Initialized.
Switched to clocksource arm_global_timer
NET: Registered protocol family 2
TCP established hash table entries: 4096 (order: 2, 16384 bytes)
TCP bind hash table entries: 4096 (order: 3, 32768 bytes)
TCP: Hash tables configured (established 4096 bind 4096)
TCP: reno registered
UDP hash table entries: 256 (order: 1, 8192 bytes)
UDP-Lite hash table entries: 256 (order: 1, 8192 bytes)
NET: Registered protocol family 1
RPC: Registered named UNIX socket transport module.
RPC: Registered udp transport module.
RPC: Registered tcp transport module.
RPC: Registered tcp NFSv4.1 backchannel transport module.
Trying to unpack rootfs image as initramfs...
rootfs image is not initramfs (no cpio magic); looks like an initrd
Freeing initrd memory: 3608K (5f7aa000 - 5fb30000)
hw perfevents: enabled with armv7_cortex_a9 PMU driver, 7 counters available
futex hash table entries: 512 (order: 3, 32768 bytes)
jffs2: version 2.2. (NAND) (SUMMARY) © 2001-2006 Red Hat, Inc.
msgmni has been set to 1001
io scheduler noop registered
io scheduler deadline registered
io scheduler cfq registered (default)
dma-pl330 f8003000.ps7-dma: Loaded driver for PL330 DMAC-241330
dma-pl330 f8003000.ps7-dma: DBUFF-128x8bytes Num_Chans-8 Num_Peri-4 Num_Events-16
xuartps e0001000.serial: ttyPS0 at MMIO 0xe0001000 (irq = 82, base_baud = 3125000) is a xuartps
console [ttyPS0] enabled
xdevcfg f8007000.ps7-dev-cfg: ioremap 0xf8007000 to 6086c000
[drm] Initialized drm 1.1.0 20060810
brd: module loaded
loop: module loaded

```
CAN device driver interface
e1000e: Intel(R) PRO/1000 Network Driver - 2.3.2-k
e1000e: Copyright(c) 1999 - 2014 Intel Corporation.
libphy: XEMACPS mii bus: probed
xemacps e000b000.ps7-ethernet: invalid address, use random
xemacps e000b000.ps7-ethernet: MAC updated fe:22:fd:af:42:b2
xemacps e000b000.ps7-ethernet: pdev->id -1, baseaddr 0xe000b000, irq 54
ehci_hcd: USB 2.0 'Enhanced' Host Controller (EHCI) Driver
ehci-pci: EHCI PCI platform driver
zynq-dr e0002000.ps7-usb: Unable to init USB phy, missing?
usbcore: registered new interface driver usb-storage
mousedev: PS/2 mouse device common for all mice
i2c /dev entries driver
Xilinx Zynq CpuIdle Driver started
sdhci: Secure Digital Host Controller Interface driver
sdhci: Copyright(c) Pierre Ossman
sdhci-pltfm: SDHCI platform and OF driver helper
sdhci-arasan e0100000.ps7-sdio: No vmmc regulator found
sdhci-arasan e0100000.ps7-sdio: No vqmmc regulator found
mmc0: SDHCI controller on e0100000.ps7-sdio [e0100000.ps7-sdio] using ADMA
ledtrig-cpu: registered to indicate activity on CPUs
usbcore: registered new interface driver usbhid
usbhid: USB HID core driver
TCP: cubic registered
NET: Registered protocol family 17
can: controller area network core (rev 20120528 abi 9)
NET: Registered protocol family 29
can: raw protocol (rev 20120528)
can: broadcast manager protocol (rev 20120528 t)
can: netlink gateway (rev 20130117) max_hops=1
zynq_pm_ioremap: no compatible node found for 'xlnx,zynq-ddrc-a05'
zynq_pm_late_init: Unable to map DDRC IO memory.
Registering SWP/SWPB emulation handler
drivers/rtc/hctosys.c: unable to open rtc device (rtc0)
ALSA device list:
  No soundcards found.
RAMDISK: gzip image found at block 0
EXT2-fs (ram0): warning: mounting unchecked fs, running e2fsck is recommended
VFS: Mounted root (ext2 filesystem) on device 1:0.
devtmpfs: mounted
Freeing unused kernel memory: 212K (40627000 - 4065c000)
Starting rcS...
++ Mounting filesystem
```

++ Setting up mdev
++ Starting telnet daemon
++ Starting http daemon
++ Starting ftp daemon
++ Starting dropbear (ssh) daemon
random: dropbear urandom read with 0 bits of entropy available
rcS Complete
#

QUESTIONS

4. [4 points.] Answers to the following questions:

(a) Compared to lab 3, the lab 4 microprocessor system shown in Figure 1 has 512 MB of SDRAM. However, our system still includes a small amount of local memory. What is the function of the local memory? Does this 'local memory" exist on a standard motherboard? If so, where?

The small amount of local memory referred to in the question functions like cache for faster access. This boost is possible due to temporal and spatial locality in the data accessed by the processor. The block of memory accessed is brought into the local memory making next accesses to it relatively faster.

On a standard board too, there is a provision for such local memory. There is a memory hierarchy implemented in the design with different levels of memory between SDRAM and processor register. Few or all of these levels can be fabricated to the processor chip itself.

(b) After your Linux system boots, navigate through the various directories. Determine which of these directories are writable. (Note that the man page for 'ls' may be helpful). Test the permissions by typing 'touch <filename>' in each of the directories. If the file, <filename>, is created, that directory is writable. Suppose you are able to create a file in one of these directories. What happens to this file when you restart the ZYBO board? Why?

Only the **proc** and **sys** directories are not writeable. When the system is restarted after writing a file in a directory, the written data is lost as the file system RAMDISK used is temporary and volatile.

(c) If you were to add another peripheral to your system after compiling the kernel, which of the above steps would you have to repeat? Why?

We need to regenerate the bitstream with added peripheral. The u-boot can be reused as it is already configured and compiled for the zybo-zynq processing system. The zynq boot image should be created again with the updated bitstream file. There is no need to re-compile linux as well. But the .dts file should be modified to include the address space of our added peripheral. This modified .dts file is to be converted to .dtb file again. Hence, we have to regenerate BOOT.bin and devicetree.dtb for our modified design.