Analysis of Bootloader and Transplantation of U-Boot Based on S5PC100 Processor

Xuchang Ding[1,2], Yongqing Liao[1,2], Jianguo Fu[1,3], Hongbin Huang[1,2], Weiping Liu[1,2]
1. Department of Electronics Engineering, Jinan University, Guangzhou, 510632
2. Institute of Industry Technology, Guangzhou & Chinese Academy of Sciences, R&D Center of Laser and Photoeletricity Technology, Guangzhou, 511458
3. Optoelectronic and Fiber Sensing Institute of Beijing Pioptics, Beijing, 100085
Email: sunnydxc@gmail.com, lyq2009186@126.com, fushiwangchao@163.com

***Abstract*—The ARM Cortex™-A8 processor is a high-performance, low-power, cached application processor that implements ARMv7-A architecture and provides full virtual memory capabilities. This paper describes the process of booting S5PC100 processor integrating an ARM Cortex™-A8, briefly introduces hardware environment of the target board cored with S5PC100 processor and an approach to build software developing environment for this embedded system, especially analyzes the process of U-Boot booting based on S5PC100 processor and the major steps of transplanting U-Boot to the target platform. U-Boot was transplanted successfully.**

Keywords-embedded system; S5PC100 processor; U-Boot; ARM Cortex™-A8; transplant

I. INTRODUCTION

With the development of 3G technology in the world, mobile internet and cloud computing have been the world's two major application technology, and they are developing rapidly. Meanwhile, the demand of mobile terminals is growing day by day, such as smart phones and tablet PCs. In order to meet higher data rate and better user experience, the performance of the microprocessors has been more and more important.

Nowadays, ARM9™ and ARM11™ processor families are still widely licensed around the world and provide cost-effective solutions for many of today's applications, but the ARM Cortex™-A8 processor is relatively small[1] [2]. Cortex™-A8 processor is baesd on the ARMv7 architecture and has the ability to scale in speed from 600MHz to greater than 1GHz[3]. And it has high-performance, superscalar microarchitecture and NEON™ technology for multi-media and SIMD processing. So the Cortex-A8 processor can meet the requirements for power-optimized mobile devices and performance-optimized consumer applications requiring higher Dhrystone MIPS.

Samsung S5PC100 integrated Cortex™-A8 processor is a 32-bit RISC cost-effective, low power, high performance micro-processor solution for mobile phones and general applications[4]. It includes many powerful hardware accelerators, integrated Multi Format Codec and numerous peripherals. S5PC100 is mainly used in consumer electronic products, such as high-end smart phones, tablet PCs, netbooks and STBs[4]. Bootloader is a small program to start up the main operating system, and U-Boot is a popular and powerful bootloader. The architecture of S5PC100 is different with S3C6410 and S5P6440, so the method of transplantation is not the same. Booting analysis of S5PC100 is a very important step to build embedded systems, so whether the transplantation is successful or not directly impacts on the performance of the system. Due to little study on S5PC100 at home and abroad, this paper focuses on the booting analysis of S5PC100 processor and U-Boot transplantation on UT-S5PC100 development-board. Besides, a few major steps mentioned in this paper will promote the application of the processor.

II. DEVELOPING ENVIRONMENT OF EMBEDDED SYSTEM

A. Hardware Environment

This article adopts UT-S5PC100 board as hardware development platform, and it is specially designed for electronic product, such as consumer electronics, industrial control, car navigation. This platform consists of a core-board and a base-board.The core of core-board is S5PC100 processor with stable and powerful performance, and it is also equipped with 256MB DDR2, 2MB NOR Flash and 256MB NAND Flash. The detail information of memory in the address space is shown as table 1. The base-board is equipped with many communication interfaces(10/100Mbps Ethernet, USB OTG 2.0, SD card, etc.), audio and video interfaces (HDMI, audio, TV Camera, VGA, etc.), man-machine interfaces(LCD, GPIO, keyboard, etc.).

TABLE I. MEMORY INFORMATION ON UT-S5PC100 BOARD

Memory Information	Memory Type		
	DDR RAM	***NOR Flash***	***NAND Flash***
Chip Type	Samsung K4X1G163	AMD AM29LV160DB	Samsung K9F2G08
Chip Number	2	1	1
Totle Size	256MB	2MB	256MB
Chip Signal	Xm1CSn0	Xm0CSn0	Xm0CSn2 Xm0CSn3
Usage	DRAM	SMC Bank0	SMC Bank2 SMC Bank3
Start Address	0x2000_0000	0x8000_0000	0x9000_0000
Limit Address	0x3000_0000	0x8020_0000	0xA000_0000

B. Software Environment

This article adopts software developing environment with Vmware 7.0 and Ubuntu 10.04. First, installing Vmware 7.0

978-0-7695-4444-1/11 $26.00 © 2011 IEEE
DOI 10.1109/IHMSC.2011.22

virtual machine in Windows XP; then, installing a Linux distribution named Ubuntu 10.04 in the virtual machine; finally, making some modifies and configurations on Ubuntu. The cross-compiler tools are Cross-4.2.2-eabi.tar.bz2 and arm-none-linux-gnueabi-arm-2008q3-72-for-linux.tar.bz2, which are installed in the file /usr/local/arm.

III. Analysis of Booting S5PC100

Compared with S3C6410 and S5P6440 microprocessors, S5PC100 supports four booting devices, which are NAND Flash, OneNAND, MMC/SD Memory and USB. The booting device is determined by the OM and NFMOD pins. S5PC100 has ROM and SRAM inside the chip. At the system reset, the program execution starts at iROM[5].

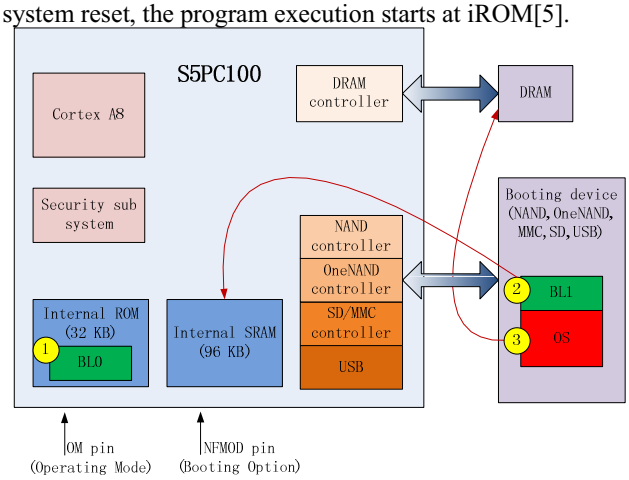

Figure 1. S5PC100 booting block diagram[5].

The boot loader of S5PC100 is divided into the BL0 and the BL1. BL0 which is placed in iROM, initializes the PLL & Clock setting with fixed value, the stack and heap region, and the Instruction Cache controller. Then BL0 loads BL1 from the booting device to iRAM. Besides, according to the secure boot key values, BL0 checks the integrity of BL1. If integrity check passes, then jump to 0x34010; if fails, then it stops. BL1 initializes the DRAM controller, then loads the OS image from the booting device to DRAM. And, according to the secure boot key values, BL1 checks the integrity of the OS image[5].

From above, it shows that the process of booting S5PC100 has its own features, such as ROM inside the chip, checking the integrity of image with secure boot key values. These features should be paid attention in applications.

IV. Analysis of Booting and Transplantation of U-Boot on S5PC100

U-Boot is an open-source bootloader. It performs powerfully, leads to a variety of operation systems, supports multiple architectures CPUs, and is very useful in embedded field. As a small program to start up the main operating system, the success of U-Boot transplantation is extremely important[6]. In the followings, firstly analyzing the U-Boot booting of S5PC100, then illustrating the keypoints of U-Boot transplantation and the method of generating executive files, finally describing a NAND Flash memory distribution for executive files on UT-S5PC100 platform.

A. Analysis of U-Boot

There are two stages in the progress of starting up U-Boot[7].

1) Stage 1 uses assemble code, and the following things are completed:

a) Initialize hardware device: set interrupt vector table, initialize hardware device depending on CPU architecture(set CPU mode as supervisor mode, use ARM instruction set, turn off FIQ and IRQ, turn off MMU and caches), read booting information,etc. The code is located in the directory cpu/s5pc1xx/start.S.

b) Prepare RAM space for loading stage 2: shut down watchdog, initialize SRAM controller, system clock, serial interface, NAND Flash, etc. The code is located in the directory board/ Samsung/Smdkc100/lowlevel_init.S.

c) Copy stage 2 to RAM space: according to the booting information, select booting device, and copy stage 2 to RAM.

d) Initialize heap region: turn on MMU, set sp pointer.

e) Jump to C entrance of stage 2: clear bss section before jumping, call function start_armboot in the directory lib_arm/Board.c.

2) Stage 2 uses C code to accomplish complex functions, and it has better readability and portability. The following things are completed:

a) Allocate RAM space for U-Boot: allocate space for stack, heap, environment variables, global parameters and board parameters.

b) Set SMC controller.

c) Test the system memory map: function dram_init() specifies the starting address and size of memory.

d) Copy kernel image and root file system image from Flash to RAM.

e) Set booting parameters for kernel: set parameters, such as tagged list and environment variables.

f) Enable interrupt, call for the kernel.

B. Transplantation of U-Boot

The project of U-Boot is so huge that we must modify a lot of source files. There are major steps of U-Boot transplantation as follows:

1) Add the header file smdkc100.h to the directory include/configs, we can modify this file to reduct U-Boot:

a) "#define MEMORY_BASE_ADDRESS 0x20000000": memory base address is 0x20000000.

*b) "#define CFG_UBOOT_SIZE (2*1024*1024)" :* the size of allocated memory for U-boot is 2MB.

c) "#define CONFIG_BOOTARGS root=/dev/mtdblock3 rootfstype=yaffs2 init=/init console=ttySAC0, 115200": loading root file system to the

fourth partition, the type of file system is yaffs2, and initialization process is init.

2) Creat new folder smdkc100 in the directory board/Samsung, add smdkc100.c、flash.c、lowlevel_init.S 、config.mk、Makefile、u-boot.lds to this folder[8]。

a) The code section of u-boot.lds file is:

```
.text:
        {
         cpu/s5pc1xx/start.o  (.text)
         cpu/s5pc1xx/s5pc100/cpu_init.o  (.text)
         board/samsung/smdkc100/lowlevel_init.o  (.text)
          cpu/s5pc1xx/onenand_cp.o  (.text)
          cpu/s5pc1xx/nand_cp.o  (.text)
          cpu/s5pc1xx/movi.o  (.text)
         *(.text)
        }
```

b) Makefile, change the OBJS and SOBJS parameters as follows:

```
OBJS        := smdkc100.o flash.o
SOBJS       := lowlevel_init.o
```

c) File config.mk is used to set the start address of program link:

```
ifndef TEXT_BASE
TEXT_BASE = 0xc7e00000
endif
```

3) Creat new folder s5pc1xx to the directory cpu/, which has many source files closely related with the architecture of S5PC100 processor, such as start.S which accomplishes the functions including set processor status, initialize interrupts, relocate U-Boot. The interrupt vector table is set as follow:

```
.globl _start
_start: b       reset
        ldr     pc, _undefined_instruction
        ldr     pc, _software_interrupt
        ldr     pc, _prefetch_abort
        ldr     pc, _data_abort
        ldr     pc, _not_used
        ldr     pc, _irq
        ldr     pc, _fiq
```

4) Modify the Makefile in root directory, add two lines in the corresponding position as follow:

```
smdkc100_config : unconfig
        @$(MKCONFIG) $(@:_config=) arm s5pc1xx smdkc100 samsung s5pc100
```

The parameters above represent the CPU architecture is arm, the type of CPU is s5pc1xx, the type of development board is smdkc100, the company is samsung, and the system on chip is s5pc100.

C. Generate U-Boot Executive Code

Login to the Ubuntu system, copy U-Boot source compressed package utc100_uboot.rar to the directory home/sunnydxc/utc100/, open *application->accessories->terminal*, enter commands as follow:

```
$ cd /home/sunnydxc/utc100
$ tar xvf utc100_uboot.tgz
$ cd uboot-samsung
$ make distclean
$ make smdkc100_config
$ make
```

Combining with above details, the command "make smdkc100_config" means the command "./mkconfig smdkc100 arm s5pc1xx smdkc100 samsung s5pc100". After compiling the U-Boot, we get u-boot.bin, u-boot, and u-boot.srec. The binary file u-boot.bin is the executive file we need to write to NAND Flash.

D. Memory Distribution of System NAND Flash

Put U-Boot image u-boot.bin, Linux kernel image zImage, Android root file system image utc100_root.img burn to NAND Flash, the memory allocation for these three images is shown in Figure 2.

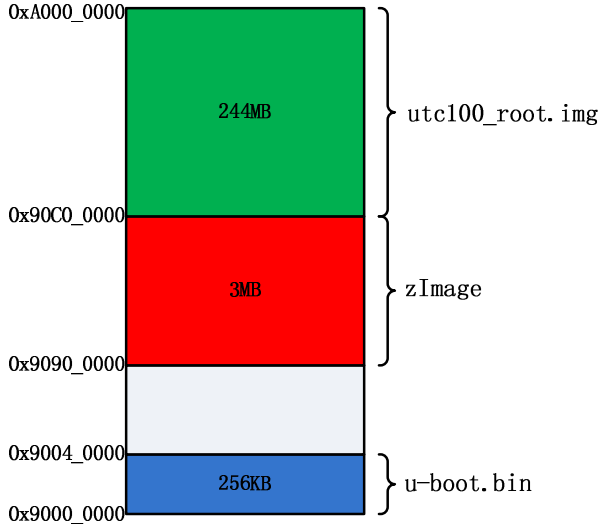

Figure 2. NAND Flash memory allocation diagram.

After system is powered on, execute BL0 in iROM, then load u-boot.bin from NAND Flash to iRAM and run it. At last, according to the booting command "nand read 20008000 900000 300000; bootm 20008000", copy Linux kernel image to DRAM addressed by 0x20008000 and jump here to run.

V. CONCLUSION

With the advantages of low power consumption and high performance, ARM Cortex™-A8 processor is an important member of Cortex™-A series ones. Samsung S5PC100 based on Cortex™-A8 processor provides high performance microprocessor solution for mobile phones and general applications. Compared with other ARM series microprocessor, the process of booting S5PC100 has its own features. This article describes the process of booting S5PC100 processor, then analyzes the process of starting up U-Boot. And we transplant U-Boot to the UT-S5PC100 board based on S5PC100 processor, run it on this board successfully, and discuss the key technology and steps of the

transplantation. Besides, we describes a memory distribution of Android embedded system.

ACKNOWLEDGMENT

This work is supported in part by *University-Industry-Science Partnership Project of Guangdong Province and National Education Ministry(2009B090200023) and National 973 Plan Project (2010CB327806).*

REFERENCES

[1] Yanpeng Sun, Peng Peng, Yuan Zhang, "Linux Transplantation Based on The Processor S3C2440", The Ninth International Conference on Electronic Measurement & Instruments, 2009, pp.2-306–2-309.

[2] Wenyan Ci, Xueli Chen, Suhua Cai, Chunmei Xia, "Methods and skills on transplanting Linux to ARM S3C2410", 2nd International Conference on Computer Engineering and Technology, 2010, pp.V4-8–V4-11.

[3] ARMV7-A architecture reference manual. Available on Request from ARM

[4] ARM Cortex-A8 technical reference manual. [Online]. Available: http://infocenter.arm.com/help/topic/com.arm.doc.ddi0344k/DDI0344K_cortex_a8_r3p2_trm.pdf

[5] S5PC100 USER'S MANUAL. Available on Request from SAMSUNG

[6] Jun Li, Huachun Zhang, "Porting method of U-Boot and Linux 2.6 to platform based on S3C2440A", Chinese Journal Of Electron Devices, 2008, (10):1667–1670.

[7] Dongshan Wei, Embedded Linux application development completely manual, Beijing: People's Posts & Telecom press, 2008, pp.240–292.

[8] Tianze Sun, Wenju Yuan, Haifeng Zhang, The Guideline of Embedded Linux Design and Linux Driver Development, 3rd Edition, Beijing: Publishing House of Electronics Industry, 2009, pp.121–171