A Design Method of Linux Bootloader Based on S3C2440

Xianfan XU*, Kun ZHANG, Lei XU, Qiang LI
Key Lab of Intelligent Computing and Signal Processing of Ministry of Education, Anhui University
Hefei, China
e-mail:xxf@ahu.edu.cn

***Abstract*—Bootloader is an important part in the developing stage of embedded system. Most embedded Linux Bootloaders are developed based on development and application, which have the disadvantages of large code capacity and low booting speed. In this paper, by only operating necessary equipments closely related to booting Linux kernel and combining with dual-boot characteristic of S3C2440 and the staged booting method, a new fast booting Bootloader is designed, which can support dual-boot of NOR Flash and NAND Flash. The practical test results show that the Bootloader prompted in this paper has the advantages of low code capacity, fast booting speed and high implementation efficiency.**

Keywords-Bootloader; S3C2440; Linux kernel; Dual-boot

I. INTRODUCTION

Bootloader will have initialized hardware equipments and built memory space mapping before the running of operating system kernel, then the hardware and software environment of system will be brought into a state that is needed by booting operating system kernel, appropriate environment is finally prepared for calling operating system kernel[1]. The design of embedded Bootloader heavily relies on CPU architecture and hardware environment. Although using the same CPU, we also need the design matched with hardware environment respectively due to the difference of board-level equipments. So it is difficult to build a completely universal and standard Bootloader.

Embedded system developers need to design or transplant Bootloader by themselves according to CPU structure and hardware feature. The workload of transplantation is relatively small. However, most embedded Linux Bootloaders are tools that face to almost all hardware equipments, for example, U-Boot (Universal Bootloader), which is used commonly[2]. They can meet requirements of most hardware platforms, but they are developed based on development and application, which has huge code, complicate file structure and beyond understanding[3]. The existing self-designed Bootloader did not fully consider the booting speed. They also cannot support dual-boot of NOR Flash and NAND Flash [4][5].

Based on above analysis, the special target board, the core of that is S3C2440 micro-processor, is used as hardware test platform. We have designed a Bootloader which can run fast and support dual-boot combining the staged booting method and dual-boot characteristic of S3C2440. In order to further accelerate the booting speed in different booting modes, we try to omit operations which have nothing to do with the booting of Linux and take full advantage of the steppingstone function offered by S3C2440[6]. Finally, Linux kernel is made to boot fast in two booting modes by doing corresponding process.

II. INTRODUCTION OF SYSTEM HARDWARE PLATFORM

A. Composition of System Hardware

The hardware platform of this embedded system is based on the development board of Samsung's S3C2440A. Hardware system block diagram is showed in figure 1.

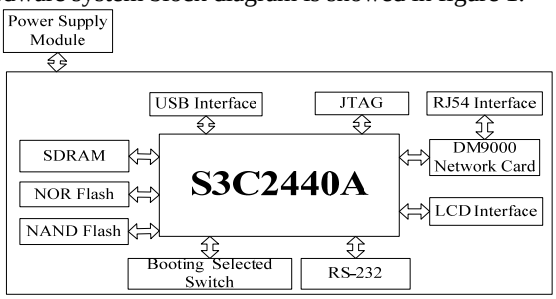

Figure 1. Hardware System Block Diagram.

Application of the NOR Flash is simple. Special gate circuit is needless. Interfaces of the NOR Flash are identical with RAM. All addresses are visible. The data of any address can be accessed randomly. It supports XIP (execute in place) which means that NOR Flash codes can be directly executed in the NOR Flash without copying codes to memory.

The NAND Flash interfaces are different from NOR Flash. It only contains several I/O pins which are only accessed in serial mode. The program code can not directly run in NAND Flash and have to be copied to memory. Reading NAND Flash one page one time is conducted. The timing sequence of reading and writing is complex. Reading and writing data in NAND Flash always needs support of Memory Technology Deriver (MTD). The NAND Flash used in this system is K9F2G08U0A of Samsung. The page size is 2KBytes. The bite wide is 8bit. The capacity is 256MBytes. It will need 5 cycles to realize access to address and data[7].

B. Booting Modes of S3C2440

S3C2440 can support two booting modes, NOR Flash and NAND Flash, which are configured by the operating mode pin OM1 and OM2. Whether S3C2440 booting from NOR Flash or NAND Flash, program will be executed from address 0x0000 0000, but the space mapping of address is

different. When S3C2440 boots from NOR Flash, the chip selection space of nGCS0 will be mapped to BANK0 (startup address 0x0000 0000). At this time, nGCS0 will connect NOR Flash so that booting program can be executed from NOR Flash directly.

The process of S3C2440 booting from NAND Flash is showed in figure 2. In order to support booting from NAND Flash, S3C2440 is equipped with a built-in 4KBytes SRAM (steppingstone). When S3C2440 boots from NAND Flash, the first 4KBytes data of NAND Flash memory will be loaded into the internal SRAM automatically by the internal integrated NAND Flash controller of S3C2440. At the same time, the 4KBytes SRAM is mapped to Bank0, then S3C2440 will execute the booting code which have been loaded into steppingstone[6].

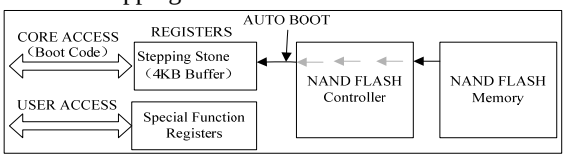

Figure 2. NAND Flash Controller Boot Loader Block Diagram.

III. OVERALL DESIGN OF BOOTLOADER PROGRAM

The booting type of Bootloader can be single-stage or multi-stage. In order to increase execution efficiency and system universality, Bootloader designed in this paper is divided into two stages. The first stage is written by assembly language, and the second stage is designed by C language. Completing the first stage depends on the necessary initialization of CPU architecture, such as turning off watchdog, setting CPU speed and clock frequency, initializing interrupt, initializing SDRAM, clearing BSS segment, setting stack, etc. The missions of the second stage are initializing hardware equipments which will be used in this stage, judging the mode of booting, setting booting parameters for kernel. In order to reduce the detections and initializations of board-level equipments, we just initialize necessary equipments and develop drivers of hardware devices which are related to the booting of Linux kernel. This method will minimize and optimize code and accelerate booting speed.

A. *The First Stage of Design Flow*

The first stage of design flow is showed in figure 3.

1) Building initialization environment.

Watchdog is enabled by default when system is powered on. In order to prevent system from rebooting constantly, it need to be turned off. The main mission of Bootloader is to boot kernel. Offering service for interrupt is the responsibility of operating system and applications. Therefore, Bootloader does not have to response any interrupt in implementation, without building exception vector table for interrupt.

2) Setting clock, enabling instruction Cache and accelerating booting speed.

System clock will be input clock when system is powered on. In order to accelerate booting speed, registers that are related to interior clock of S3C2440 are set. Basic frequency is set at 400M. At the same time, instruction Cache is enabled and programs are made to run fast.

3) Clearing BSS segment.

4) Initializing data width of SDRAM, access cycle of reading/writing and refresh cycle.

5) Initializing UART.

It will be convenient to debug through Hyper Terminal and observe booting information of Linux kernel.

6) Setting stack.

Setting stack is prepared for C language code. S3C2440 supports seven modes of operation. Although every operation mode of processor has it own physical stack register R13, CPU is in supervisor mode after initializing CPU (CPU must be in supervisor mode before booting kernel). We just need to initialize stack register in this mode[8].

7) Jumping to C language entrance of the second stage.

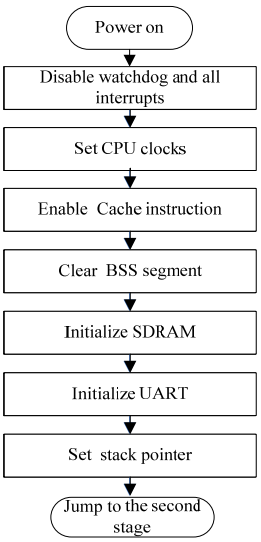

Figure 3. The First Stage of Booting Process.

B. *The Second Stage of Design Flow*

The second stage of design flow is showed in figure 4.

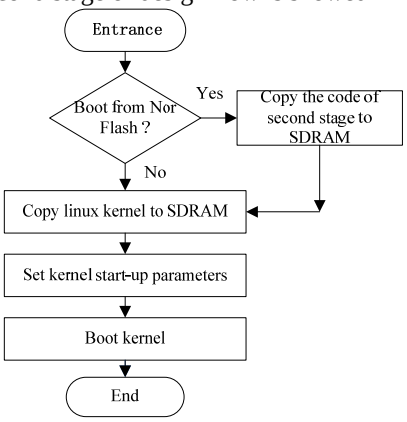

Figure 4. The Second Stage of Booting Process.

1) Judging the booting mode and realizing dual-boot.

The booting characteristic of S3C2440 shows that program will be run in the built-in SRAM (steppingstone) if S3C2440 boots from NAND Flash at the beginning. The data of address 0x0000 0000 can be read and written directly. When S3C2440 boots from the NOR Flash, program will be run in NOR Flash, data can not be written by directly assigning of C language.The data which is different from machine code at address 0x0000 0000 is written to address 0x0000 0000, then the data of address 0x0000 0000 is read and compared with the data written just now. If the result is identical, S3C2440 boots from NAND Flash, else it boots from NOR Flash[9].

2) Processing according to different booting modes and accelerating booting speed.

Bootloader will be executed in NOR Flash directly and the speed is far less than SDRAM if S3C2440 boots from the NOR Flash. In order to accelerate booting speed, the second stage program is loaded to SDRAM. If S3C2440 boots from the NAND Flash, it will copy the first 4KBytes content of NAND Flash to on-chip SRAM automatically. The size of Bootloader in this article is less than 3K. So there is no necessary to load codes of the second stage to SDRAM. At the same time, timing refreshing and control of reading and writing are needless. Then the booting speed can be further accelerated.

3) Copying Linux kernel to the specified location on SDRAM(0x30008000).

4) Setting booting parameters of kernel.

The interaction between Bootloader and kernel is unidirectional. Bootloader transfer the parameters to Linux kernel, while Bootloader and kernel can not be run at the same time. So firstly, Bootloader will storage the parameters into an appointed place（address 0x30000100）, then it will boot kernel, finally, kernel will obtain the parameters from this place. The kernels after Linux2.4 all expect to pass booting parameters by tag list. Tag is a structure. Tag list is to store several marks one by one. Tag list starts with ATAG_CORE and ends with ATAG_NONE[10]. Main codes are showed as follows:

```
//Setting booting tag (The start position is 0x30000100)
setup_start_tag();
//Setting system memory parameters
setup_memory_tag();
// Setting the command line parameters
setup_commandline_tag();
// Setting end tag
setup_end_tag();
```

5) Jumping to the entrance of kernel (0x30008000) and booting kernel.

The main codes are showed as follows:

```
void (*theKernel)(int zero, int arch, unsigned int params);
theKernel = (void (*)(int, int, unsigned int))0x30008000;
theKernel(0, MACH_TYPE_S3C2440, BOOT_PARAMS_ADDR);
```

The equivalent codes written by assembly language are showed as follows:

```
mov r0， #0
ldr r1， = MACH_TYPE_S3C2440
ldr r2，= BOOT_PARAMS_ADDR
mov pc， #0x30008000
```

Kernel will call bottom function according to MACH_TYPE_S3C2440 (machine ID). BOOT_PARAMS_AD DR (the value is 0x3000 0100 in this system) will tell kernel the start address of tag list.

With this, the task of Bootloader has been completed. System's control right will be handed over to the Linux kernel. Kernel will start to execute.

IV. THE EXPERIMENTAL RESULTS

In order to fully verify the feasibility of the Bootloader, the staged debug has been carried out in the design .LED is right lit in the first stage, which insured reading CPU registers and initializing CPU correctly. UART is initialized in the second stage. Hyper Terminal will correctly output debugging information. Bootloader is complied with cross-complier and the generated binary file is written to the NOR Flash and NAND Flash of development board via JTAG. Booting from NOR Flash or NAND Flash is based on booting selected switch. The booting information is showed in Figure 5. The experimental results have showed that the Bootloader can boot Linux kernel quickly and correctly. It can also support dual-boot of NOR Flash and NAND Flash.

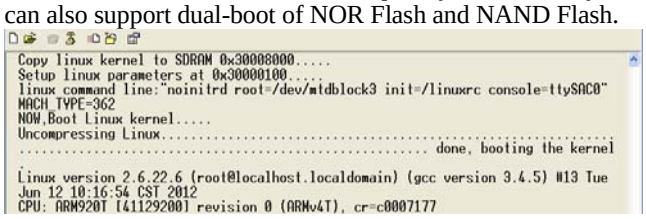

Figure 5. Bootloader Boot the Linux Kernel.

V. CONCLUSIONS

The special target board, the core CPU of that is S3C2440, is used to design a new Bootloader.The Bootloader not only has advantages of small code scale convenient maintenance, but also can support dual-boot of NOR Flash and NAND Flash. The initialization and the driver development of equipments that are related to booting Linux can accelerate the booting speed. We also take full consideration of the difference between NOR Flash and NAND Flash, and take full advantage of built-in 4Kbytes SRAM of S3C2440. The whole design flow and critical codes are provided, and every step of the design is described in detail too. The design method has strong practical value in the development and can offer a new good reference for other Bootloader's developments simultaneously.

REFERENCES

[1] Wang Ya-gang."Analysis and Transplant of Embedded Bootloader Mechanism", Computer Engineering,Vol.36,No.6,2010:267-269

[2] Wolfgang Denk. The DENX U-Boot and Linux Guide (DULG) for canyonlands [EB/OL]. http://www.den x.de/wiki/ publish/DULG/ DULG-canyonlands.pdf,2012

[3] Ye Lin,Fang Jian-jun."Design of the BootLoader Based on ARM9 Embedded System", Science &Technology Information, 2009(11):428-429.

[4] Tian Hui-feng,"Design and Realization of Bootloader Based on S3C2440",Computer Applications,Vol.29,No.7,2010:29-32

[5] Jiang Wei,"Analysis and Design of Embedded System's Bootloader Based on ARM S3C2410",Electronic Engineer,Vol.34,No.10, 2008:49-52

[6] Samsung Electronics.S3C2440A 32-Bit RISC micro controller user's manual [Z],2004.

[7] Samsung Electronics. K9F2G08UXA Flash Memory Data Sheet [Z], 2007.

[8] Guo Feng, Yuan Guo-liang, Wang Li-fang."Analysis and design of embedded Linux Bootloader", Information Technology,2011(11), 123-125

[9] Du Chun-lei. Configuration System and Programme in ARM[M]. Beijing: Tsinghua University Press,2003.

[10] Wei Dong shan. Embedded Linux Application Development Completely Manual [M] Beijing: People's Posts and Telecommunications Press, 2008.