LinuxBIOS를 이용한 X86 기반 임베디드 시스템의 빠른 부팅기법

윤 희 철, 마 진 석, 김 선 자
한국전자통신연구원
내장형 소프트웨어 연구팀
(E-mail) {hcyun,jsma,sunjakim}@etri.re.kr
☎ 042-860-1673 (Fax) 042-860-6671

Booting Linux less than 10 second with LinuxBIOS on x86 based Embedded System.

Heechul Yun, JinSuk Ma, Sunja Kim
Embedded Software Team
ETRI

요 약

이 논문은 x86 기반의 내장형 시스템에서 빠른 부팅을 위한 BIOS 솔루션으로서 LinuxBIOS에 관해 다룬다. 본 논문에서는 특히 LinuxBIOS의 동작 메커니즘과 포팅을 위한 구체적인 접근 방법 및 문제점 등에 대해 상세히 기술한다. 우리는 Embedded Linux를 운영체제로 하는 x86기반 임베디드 시스템에 LinuxBIOS를 적용하여 부팅 시간을 30초에서 11초로 단축시켰다.

1. Introduction

홈 게이트웨이, PVR 같은 고성능의 CPU 성능을 요구하는 정보가전제품 늘어나는 추세이다. 이와 같은 시스템에서는 강력하면서도 가격이 상대적으로 저렴한 x86 기반의 시스템을 사용하려는 시도가 늘어나고 있다. X86 기반의 시스템에서는 시스템의 기본 초기화 및 부팅을 위한 firmware 를 BIOS 라고 하는데 BIOS 는 MS-DOS 와의 호환성의 유지를 위해 다른 아키텍처의 Firmware 와 달리 복잡하고 불필요한 기능이 많다. Linux 와 같은 현대의 운영체제는 MS-DOS 에서 BIOS 가 수행해야만 했던 기능들이 대부분 운영체제에 포함되어 있기 때문에 중복된 수행으로 인한 실행시간의 저하를 초래한다. 일반적인 상용 BIOS 들은 파워를 켜는 순간부터 운영체제 로딩을 시작하는 순간까지 걸리는 시간(이하 bios 시간) 이 무려 10~20 초 정도로 매우 긴 시간이 필요로 하다. 여기에 운영체제까지 로딩하는 시간을 더하면 실제 부팅 시간은 30 ~ 60 초 가량의 시간이 소요된다. PC 와는 달리 정보가전 제품에 있어서는 이처럼 긴 시간 동안 사용자를 기다리게 하는 것은 제품 경쟁력에 있어 매우 커다란 마이너스 요소이다. 이와 같은 긴 부팅시간을 획기적으로 단축시키면서 동시에 라이센스 비용이 필요 없는 공개 소프트웨어인 LinuxBIOS [1] 라는 대안이 있다. LinuxBIOS 를 firmware 로서의 최소한의 기능 즉 부팅을 위한 최소한의 기능만을 수행하고 나머지는 운영체제에 맡김으로서 수행시간을 최소화한다. 때문에 MS-DOS 구형 운영체제와의 호환성은 제공하지 못하지만 초기화에 소요되는 시간은 1 초 미만으로 기존의 BIOS 의 10~20 초와 비교하여 획기적인 부팅 속도의 향상을 기대할 수 있다. 또한 대부분의 코드가 C 언어로 작성되어 있기 때문에 수정이 용이하고 이해하기 쉽다.

본 논문의 구성은 다음과 같다. 2 장에서 LinuxBIOS 에 대한 소개를 하고 3 장에서는 ETRI 의 정보가전 연구부에서 개발중인 정보가전 시스템인 Hestia 에 LinuxBIOS 를 포팅한 구체적인 과정을 소개하고 4 장에서는 Hestia 에서 LinuxBIOS 와 일반 상용 BIOS 인 Phoenix BIOS 를 적용했을 때의 각각의 부팅 시간을 측정하고 비교한다. 5 장에서는 LinuxBIOS 를 적용하는데 있어 발생했던 여러가지 문제에 대해 다루며 마지막으로 6 장에서 결론을 맺는다.

2. LinuxBIOS

LinuxBIOS 는 X86 기반 시스템의 일반적인 BIOS (AMI, Phoenix) 등을 대체하기 위해 시작된 공개소스 프로젝트이다. 이 프로젝트는 Los Alamos National Laboratory (LANL)의 클러스터 컴퓨팅 연구그룹에 의해 클러스터링 시스템의 구축을 위한 연구중의 하나로 시작되었다. 본래의 목적은 복잡한 클러스터 시스템에 다양한 부팅 방법을 제공하기 위해 리눅스 커널을 강력한 부트로더로서 사용하기 위함이었으나 현재는 LinuxBIOS 자체에 다양한 부팅 메커니즘을 추가하는 방향으로 진행되고 있다. 또한 본래의 주 사용 영역이었던 클러스터 컴퓨터 이외에 LinuxBIOS 의 빠른 부팅 속도 및 로열티가 없다는 장점으로 인해 정보가

전기기와 같은 임베디드 시스템에 LinuxBIOS 를 적용하려는 개발자들이 늘어나고 있는 추세이다.

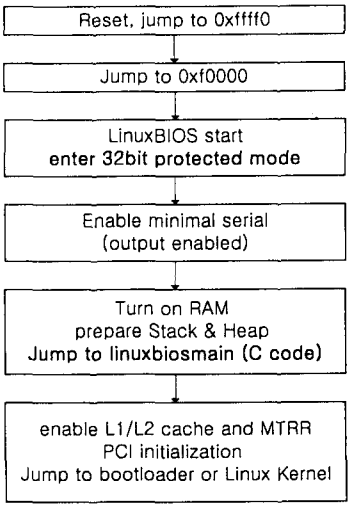

그림 1. Booting Sequence

그림 1 은 LinuxBIOS 의 부팅 시퀀스를 나타내고 있다. 그림에서 볼 수 있듯이 리눅스 바이오스는 시작과 함께 32bit protected 모드로 진입하는 것이 특징이다. 때문에 빠른 연산 수행과 32bit linear address 로 메모리 접근이 가능하다. 또한 램 초기화를 마친 이후에는 거의 모든 코드가 C 로 작성된다. 때문에 코드의 이해 및 수정이 매우 용이하다. 이에 반해 전통적인 BIOS 는 대부분 16bit real mode 에서 동작하며 모든 코드는 Assembler 로 짜여있다.

3. Porting Procedure

BIOS 는 그 특성상 칩셋 및 보드 디자인에 매우 의존적이다. 현재 LinuxBIOS 에서 지원되는 칩셋 및 보드의 수는 점차 늘어나고 있으나 모든 칩셋과 보드를 지원하지는 않으며 지원하더라도 제한적인 기능만 지원되는 경우가 많다. 우리가 사용한 하드웨어인 Hestia 의 주요 spec 은 다음과 같다.

CPU	P4 2GHz
Chipset	SIS645/SIS961
Memory	168pin SDRAM x 1
Integrated Peripheral	VGA, RTL8139 NIC x 2, IEEE1394, SIS7012 Audio, VoIP Controller, PCMCIA,

표 1. H/W (Hestia) Spec

Hestia 의 핵심 칩셋인 SIS645/961 칩셋은 현재 LinuxBIOS 에서 지원되지 않는 칩셋이다. 때문에 여기에서 기술하는 포팅 과정은 우리의 경험을 바탕으로 이처럼 지원되지 않는 칩셋의 코드를 직접 작성하는 경우를 포함하여 기술한다.

표 2 은 포팅을 위해 필요한 장비 및 문서들을 보여준다. 가장 중요한 것은 칩셋과 관련한 문서와 Universal Flash Bunner 이고 post-card 및 rom-emulator 가 있다면 더욱 편리한 개발환경을 구축할 수 있다. Post-card 는 Serial 포트를 초기화하기 이전의 디버깅을 위해 필요하며 rom-emulator 는 flash chip 대신 사용하여 매번 chip 을 뺐다 끼우는 번거로움을 없앤다.

[illegible]	universal bunner, post-card, rom-emulator
[illegible]	northbridget, southbridge chipset data sheet & programming guidesuper I/O chipset datasheet

표 2. Requirement

포팅 과정에서 가장 먼저 할 일은 serial 포트를 초기화하여 부팅 메시지 및 디버깅 메시지를 볼 수 있도록 하는 것이다. Serial 포트는 일반적으로 super I/O chipset 에서 제어되므로 super I/O datasheet 를 참조하여 serial 장치를 초기화 하는데 보통의 경우 단순히 특정 레지스터 값을 설정하는 것이며 각종 메시지 출력 루틴들은 LinuxBIOS 에서 지원되는 코드를 그대로 사용할 수 있다. Hestia 에서 사용한 winbond 의 w83697f chip [2] LinuxBIOS 에서 이미 지원되고 있던 w83627h 코드를 수정하여 사용하였다.

시리얼 포트를 초기화하였으면 다음으로는 가장 중요한 램 초기화 루틴을 작성해야 한다. 일반적으로 사용되는 SDR, DDR, RDRAM 등은 모두 특별한 초기화 과정을 거치게 되는데 이 초기화 과정은 chipset 마다 다르며 보드에 따라 buffer strength 등의 각종 파라미터 값들을 조정할 필요가 있다. 우리는 Hestia 의 northbride 인 SIS645 칩의 datasheet [3]및 programming guide [4] 를 참고하여 코드를 작성하였다. 여기까지의 과정은 모두 assembler 로 작성되어야 하는데 그 이유는 램이 초기화 되어야만 C 코드가 수행되기 위해서 꼭 필요한 스택을 사용할 수 있기 때문이다. 램 초기화가 끝난 이후의 코드는 모두 C 로 작성이 가능하다. 램의 초기화가 정상적으로 되었는지 확인하기 위해서 우리는 ram/ramtest.inc 와 memtest86 [6] 테스트슈트를 사용하였다.

[illegible]	Files to be written (or modified)
[illegible]	Superio/<vendor>/<chip>/setup_serial.inc
[illegible]	North[south]bridge/<vendor>/<chipset>/raminit.inc north[south]bridge/<vendor>/<chipset>/northbridge.c
[illegible]	southbridge/<vendor>/<chipset>/southbridge.c
[illegible]	mainboard/<vendor>/<board>/mainboard.c mainboard/<vendor>/<board>/irq_tables.c
Configuration	'Config' file in each directory

표 3. 포팅을 위한 핵심 코드

램초기화를 위한 northbridge 설정 작업이 끝난 후에는 IDE, keyboard 등의 peripheral 들과 관련된

southbridge 설정 작업을 수행한다. 관련 내용은 SIS 961 Data Sheet 를 참조하였다 [5]. Southbridge 와 관련된 또 다른 중요한 것은 PCI 장치들의 Interrupt Routing 정보의 작성이다. Southbridge chip 에는 Programmable Interrupt Router (PIC) 라고 하는 것이 있는데 이것은 시스템의 모든 PCI 장치들에서 들어오는 interrupt 를 적절하게 routing 하는 역할을 하며 Linux 는 PIC 를 통해 어느 장치에서 들어온 인터럽트인지를 판별하여 적절한 device driver 에게 전달한다. PIC 의 routing 정보는 보드 설계시에 결정되므로 board schematic 을 참고하여 작성해 주어야 한다 [7]. LinuxBIOS 에서는 PIC 를 프로그램하지는 않으며 단지 routing 정보만을 Linux 에게 넘겨주고 후에 Linux 커널이 LinuxBIOS 의 routing 정보를 이용하여 PIC 를 프로그래밍 한다. 우리는 이 과정에서 몇몇 PCI 장치들의 routing 이 제대로 수행되지 않는 문제를 발견하였는데 이는 Linux 커널이 SIS961 chip 의 router 지원에 문제점이 있었기 때문이었다. 때문에 커널의 router 관련 코드를 일부분 수정하였다.

이제 Linux 를 올릴 수 있는 모든 준비가 되었다. LinuxBIOS 는 커널을 부팅하기 위한 다양한 옵션이 가능하다. 커널을 표준 Flash EEPROM 에 같이 포함시킬 수 도 있으며, DOC [5]를 사용하거나, 디스크로부터 로딩하는 것도 가능하다. 또 etherboot [8] 와 같은 외부 부트로더를 통해 커널을 부팅하는 것이 가능하다. 이때 주의할 것은 BIOS 인터럽트 서비스 루틴을 이용하는 LILO 와 같은 일반적인 부트로더를 사용할 수 없다는 점이다. 바이오스 인터럽트를 사용하지 않는 가장 안정된 부트로더는 etherboot 이다. Etherboot 는 본래 ethernet 을 통해 부팅하기 위한 것이지만 local disk 및 floppy 로부터 커널을 로딩하는 것도 가능하다. 우리는 Etherboot 의 ide disk 드라이버를 이용하여 disk 로부터 리눅스를 부팅하였다.

전체 포팅 과정에 걸린 시간은 약 2 달로 대부분의 시간은 LinuxBIOS 에서 지원하지 않았던 SIS645 칩셋의 초기화 코드의 작성에 소요되었다. 특히 가장 핵심적인 메모리 초기화 코드의 작성에 있어 어려움을 겪었다. 메모리 초기화는 LinuxBIOS 에서 수행하는 가장 중요한 일이지만 칩셋마다 그 방법이 다르기 때문에 LinuxBIOS 를 포팅하는 데 있어 가장 어려운 부분이다.

4. Time Comparison

다음의 표는 Hestia 플랫폼에서 우리가 포팅한 LinuxBIOS 를 사용하였을 때와 주문 제작한 Phoenix BIOS 를 사용하였을 때의 부팅 시간을 비교한 것이다. 표에서 BIOS Time 은 BIOS 에서 소요된 시간 즉 파워를 켠 순간부터 BIOS 가 운영체제로 제어를 넘기는 시점까지의 시간을 뜻하며 Linux Time 은 Hestia 에서 사용한 Embedded Linux 운영체제인 Qplus 가 동작하는데 소요된 시간으로 Login 프롬프트가 나타나는데 까지 소요된 시간을 말한다. 전체 부팅 시간은 BIOS Time 과 Linux Time 을 더한 값이다.

표에서 알 수 있듯이 LinuxBIOS 는 1 초의 시간이 소요되었으나 Phoenix BIOS 는 20 초의 시간이 소요되었다. 운영체제의 실행 속도는 두 경우 모두 동일하였다.

	LinuxBIOS	Phoenix BIOS
BIOS Time	1	20
Linux Time	10	10
Total Time	11	30

표 4. LinuxBIOS 와 Phoenix BIOS R4.0 의 부팅 시간 비교.

5. Problems of LinuxBIOS

LinuxBIOS 는 훌륭한 성능과 공개소스라는 장점에도 불구하고 몇 가지 문제점이 있다. 가장 커다란 문제는 전통적인 BIOS 서비스 루틴을 제공하지 않기 때문에 BIOS 서비스를 많이 사용하는 LILO 와 같은 프로그램이나 DOS, Windows 의 사용이 기본적으로 불가능하다는 것이다. 또한 비디오 카드에서 제공되는 VGABIOS 또한 BIOS 서비스를 사용하기 때문에 VGABIOS 도 사용할 수 없다. 때문에 VGA 가 꼭 필요한 경우에는 BIOS 에 의존적이지 않는 커널 수준의 framebuffer device driver 가 필요하다. 현재로서는 Matrox 및 SIS graphic chipset 이 이러한 driver 를 제공한다. 이 문제는 video chipset 업체의 도움이 있다면 쉽게 해결 가능한 문제이다. 또 한가지 대안은 LinuxBIOS 에 BIOS 서비스 루틴을 제공하는 것인데 현재 이러한 시도가 활발하게 진행되고 있으며 Windows 의 부팅에 성공한 사례도 있다. 이것이 안정적으로 제공된다면 기존의 BIOS 를 완전히 대치하는 것이 가능하다.

6. Conclusion

LinuxBIOS 는 빠르고 로열티가 없으며 소스가 공개되어 있기 때문에 x86 기반의 정보가전 기기에 매우 효과적인 BIOS 솔루션이다. 기존의 BIOS 와 비교하여 약 20 초의 시간 단축을 기대할 수 있으며 코드가 공개되어 있고 그 대부분이 C 로 작성되어 있어 수정이 용이하다. 우리는 ETRI 정보가전 연구부의 홈서버 시스템인 Hestia 플랫폼에 LinuxBIOS 를 적용하여 20 초의 부팅 시간 단축을 얻었다.

7. Reference

[1] The LinuxBIOS Homepage, http://www.linuxbios.org
[2] W83697f Datasheet, Winbond
[3] SiS645 Host/Memory Controller With DDR333 and AGP4X North Bridge for Pentium. 4 400MHz FSB PC Systems, SIS
[4] SiS645 / SiS650 DDR Pentium 4 Chipset Programming Guide *Preliminary Rev. 0.97 Jan. 8, 2002*, SIS
[5] SiS961 MuTIOL Media I/O *Preliminary Rev. 2.90 Jan. 21, 2001*, SIS
[6] www.memtest86.com
[7] PCI System Architecture 4th edithion Tom Shanley, Don Aderson. MindShare Inc
[8] The Etherboot Homepage, www.etherboot.org