신기술 해설

내장형 운영체제의 파일 시스템 - Flash File System

박 성 호*, 정 기 동**

● 목 차 ●

1. 서 론

정보 통신 기술과 마이크로 프로세스 기술의 발전은 다양한 내장형 시스템의 발전을 가져왔다. 내장형 시스템은 운영체제, 실시간 시스템, 통신 시스템, 분산 시스템 기술이 모두 결합되어 설계되는 시스템으로, 향후 Post PC 시대를 맞이하게 되어, 그 중요성은 나날이 증대되고 있다. 그러나, 내장형 시스템은 크기가 제한되어 있고, 외부의 충격, 격심한 온도 변화 등 불안한 환경에서 동작하여야 하는 특성 때문에, 시스템 설계에 많은 어려움이 따르게 된다. 특히 저장 매체 및 저장 매체를 관리하는 파일 시스템은 내장형 시스템을 구성하는데 필수적인 기술이다.

최근 들어, 내장형 시스템의 저장매체로써 회전식 디스크 대신에 플래시메모리(Flash Memory)가 점차적으로 널리 사용되고 있다. 즉, 기존의 가전, 통신 기기, 휴대 기기, 가정용 셋탑 박스(Set-top box)나 인터넷 폰(Internet phone)에서 플래시메모리의 사용이 증대하고 있다. 이런 플래시메모리 사용의 증대는 플래시메모리가 기존의 회전식 자기 매체에 비하여 외부의 충격에 강하고, 저 전력으로 구동이 가능하며, 크기가 작은 장점을 가지고 있기 때문이다. 그러나 데이터의 내용을 수정할 때 많은 시간이 소요되고, 수명도 영구적이지 못하다는 단점을 지니고 있다. 그러므로 내장형 운영체제의 파일 시스템은 위에서 언급한 플래시메모리의 특성과 내장형 시스템의 특성을 고려하여 설계되어야 한다.

이후의 순서는 다음과 같다. 2장에서는 플래시메모리의 특성과 구조를 설명하고, 3장에서는 기존에 사용되고 있는 내장형 파일 시스템에 대하여 소개한다. 그리고 4장에서는 로그 파일 시스템을 소개하고 5장에서는 플래시 파일 시스템에 대하여 설명한다. 그리고 6장에서 마지막으로 결론을 내린다.

2. 플래시메모리의 특성 및 구조

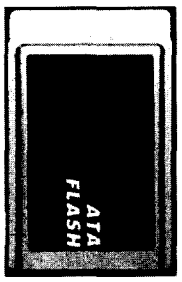
2.1 플래시메모리의 특성

현재 내장형 시스템의 저장 매체로 회전식 자기 매체를 이용한 디스크 대신하여 널리 사용되고 있는 플래시메모리(Flash Memory)는 다음과 같은 특성을 지닌다.

* 부산대학교 전자계산학과 대학원
** 부산대학교 전자계산학과 교수

- ㅇ 플래시메모리는 기존의 자기 디스크와 같은 회전식 자기 매체가 아니라 메모리이기 때문에 빠른 접근 속도를 가지므로 높은 성능을 제공한다.
- ㅇ 플래시메모리는 외부 충격에 강하고 비휘발성(Non-volatile)이다.
- ㅇ 저 전력 구동이 가능하고 크기가 작다.

위와 같은 장점 때문에 플래시메모리는 하드디스크에 비해서 메가바이트(Mega byte)당 5배에서 10배정도 비싸지만, IC-card와 같은 형태(그림 1)로 휴대용 컴퓨터(Portable computer)에서 보조 기억 장치로써 HDD(Hard Disk Driver)를 대체하고 있다.



(그림 1) Flash Memory

또한 ROM(Read Only Memory)와 달리 전기적으로 지우고 재 사용할 수 있는 플래시메모리는 <표 1>과 같은 일반적인 특정을 가진다[1]. 그리고 현재 1 기가 바이트(Giga byte) 플래시메모리가 개발되었고, 상용으로는 512M, 256M 등의 플래시메모리가 시판되고 있다.

<표 1> 플래시메모리 특성

Read Cycle	80 ~ 150 ns
Write Cycle	10 μs/byte
Erase Cycle	0.5 ~ 1 s/block
Erase Cycles limit	100,000 times
Erase Unit Size	64 ~ 256 kbytes
Power Consumption	- 30 ~ 50 mA in an active state - 20~100 μA in a standby state

저장 매체로써 플래시메모리를 사용하기 위해서는 다음의 특성을 고려하여야 한다.

첫째, 한 번 데이터를 쓴 영역에 바로 다시 쓸 수 없다. 즉, 새로운 데이터를 쓰기 전에는 반듯이 플래시메모리를 지워야 한다.

둘째, 플래시메모리를 지우는 작업은 EU(Erase Unit)단위로 이루어지고, <표 1>에서 나타나듯이 하나의 EU를 지우는데 소요되는 시간은 0.5~1 초로 많이 시간이 요구된다. 그러므로 단지 몇 바이트를 수정하기 위하여 한 EU를 지우고 다시 쓰는 것은 시스템의 큰 성능 저하를 가져올 것이다.

셋째, 플래시메모리를 지우는 회수는 제안되어 있다. 즉, 플래시메모리를 영구적으로 사용할 수 없기 때문에 지우는 작업을 최대한 억제하여 수명을 늘려야 한다.

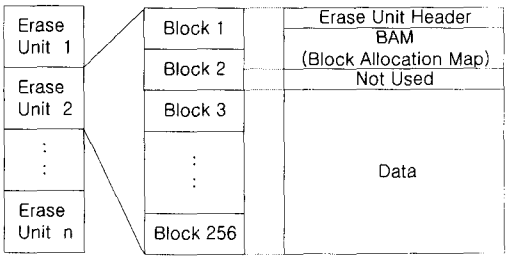
위와 같은 플래시메모리의 특성 때문에 기존에 개발된 플래시메모리를 이용한 파일 시스템에서는 로그 파일 시스템(LFS - Log-Structured File System)을 많이 사용한다. 즉, 수정된 데이터를 데이터가 저장된 블록에 기록하는 것이 아니라, 다른 빈 블록에 수정된 데이터를 기록하고 기존의 데이터가 저장된 블록은 플래그를 이용하여 사용하지 않는 방법을 채택하고 있다.

2.2 플래시메모리의 구조

플래시메모리의 데이터 구조는 대부분 유사한 구조를 가진다. 본 절에서는 인텔사의 28F320S3 [10] 플래시메모리를 예를 들어 플래시메모리의 데이터 구조를 설명한다.

28F320S3은 한 EU당 128 KBytes, 256개 블록으로 구성된다. 한 블록은 512 바이트이다. (그림 2)는 플래시메모리에 데이터 구조의 예를 나타낸다. 각 EU마다 앞부분에 EU에 대한 정보와 각 블록에 대한 정보가 저장된다.

EU의 앞부분에 각 EU의 정보를 기억하는 EHU (Erase Unit Header)가 저장된다. EHU는 Validation



(그림 2) 플래시메모리의 데이터 구조

code, 현재 EU의 논리적 번호, 한 블록의 크기, 첫 번째 EU의 번호, EU의 개수, BAM(Block Allocation Map)의 오프셋(Offset), EU의 상태 정보, EU의 지워지 횟수 등을 기록하고 있다. EUH 뒤에는 블록의 정보를 나타내는 BAM(Block Allocation Map)이 저장된다. 각 블록이 free인지, invalid인지, valid인지 등의 상태 정보와 논리적인 주소가 무엇인지를 나타낸다. 논리적인 주소는 상위 파일 시스템에서 플래시메모리 관리자에게 요청하는 주소인데, 플래시메모리 관리자가 논리적인 주소를 실제의 물리적인 플래시메모리 주소로 바꾸어야 한다. 그리고 BAM이 끝나는 다음 블록부터 실제 데이터가 저장된다. 시스템의 초기화 시, 플래시메모리 관리자는 각 EU의 EUH를 살피고, 전체 EU의 개수, 블록의 크기 등을 초기화한다. 그리고 오류가 발생한 EU가 있는지를 조사하여 오류를 복구한다.

3. 내장형 파일 시스템

내장형 시스템이 사용하는 자원(Resource)을 극도로 제한시키려 한다면 파일시스템을 사용하지 않을 수도 있다. 즉, 커널 내부에 각종 애플리케이션들을 포함시켜 커널과 하나의 바이너리 이미지로 만든 후 커널 스레드의 하나로 이들 애플리케이션들을 실행할 수 있다. 그러나 이러한 방법은 애플리케이션을 커널 스레드로 변환해야 하는 어려움이 있고 디버깅도 커널 디버깅 레벨에서 수행하여야 함으로 새로운 애플리케이션을 추가하는데 많은 비용이 소요된다. 반면에 파일 시스템을 가지게 되면 내장형 시스템 자체가 일반적인 PC 시스템과 그 구성이 비슷해지고 애플리케이션의 개발에 있어서 다른 아키텍처에서 시뮬레이션이 가능해지며 수많은 오픈 소스 애플리케이션들이 기반으로 하는 시스템과 유사성을 가지므로 시스템 구성이 매우 용이해진다.

내장형 시스템에서 파일 시스템을 구성하려면 PC의 하드디스크와 같은 저장 장치가 필요하다. 이런 목적으로 쓰일 수 있는 것들로는 RAM, ROM, 플래시메모리 등이 있으며 내장형 운영 체제에서는 각각의 장비를 디스크처럼 사용할 수 있도록 하는 다양한 종류의 디바이스 드라이버가 제공되고 있다.

다음과 같은 파일 시스템이 내장형 운영체제에서 많이 사용되고 있다.

3.1 Network File System (NFS)

NFS는 실제로 파일 시스템을 내장하기 위해서라기보다는 시스템 구성 디버깅을 위해서 많이 쓰인다. NFS 루트 마운트 기능을 개발 중인 내장형 시스템에 올리고 루트 파일시스템을 NFS로 마운트해 내장형 시스템의 재부팅이 없이 시스템을 구성해 나아갈 수 있다. 이 방법은 개발 시간을 단축시킬 수 있는 장점이 있다.

3.2 Initial Ramdisk

이니셜 램디스크(Initial Ramdisk)는 일반적인 운영체제에서 지원하는 램디스크를 이용해 루트 파일시스템을 구성하는 방법이다. 램디스크는 물리적으로 램의 일부분을 디스크로 사용할 수 있는 방법이며 이때 생성된 디스크는 일반적인 운영체제의 VFS 인터페이스를 통해 마운트되므로 일반적인 하드디스크와 동일하게 사용할 수 있다. 램디스크는 전원 공급이 끊기면 내용을 보존할 수 없는 비영구적인 저장 공간이므로 처음 부팅 시에 영구적 저장

공간에서 그 내용을 읽어오고 시스템을 멈출 때 저장이 필요한 내용은 따로 저장해야 된다는 오버헤드가 따른다. 하지만 램을 디스크로 사용하므로 속도 면에서 가장 빠르다는 장점을 가지고 있다.

3.3 ROM File System (ROMFS)

ROM 파일시스템은 EEPROM과 같은 ROM을 위해서 만들어진 파일시스템이며 ROM을 디스크로서 사용한다. 이 파일시스템은 ROM의 특성상 읽기 전용(Read-Only)으로 쓰일 수밖에 없다. ROM 파일시스템은 크기가 매우 작다. 즉, 일반적인 파일시스템 중 가장 크기가 작은 것 중 하나인 미닉스 파일시스템의 경우, 그 드라이버 크기가 20KB 이상이지만 ROMFS의 드라이버 크기는 4KB 정도다.

3.4 RAM File System (RAMFS)

RAM 파일시스템은 램디스크에 적합한 파일시스템으로 설계되었으며 성능도 최적화되어 있다.

3.5 Compressed RAM File System (CRAMFS)

압축된 RAM 파일시스템은 RAMFS와 마찬가지로 램디스크에 최적화된 파일시스템으로서 파일시스템 자체가 압축된 상태라는 것이 특징이다. 즉, 파일 접근 시 런타임에 파일시스템 드라이버 레벨에서 압축이 풀려 VFS 인터페이스로 전해지는 구조로 이루어져 있으며 따라서 저장 공간이 넓어지는 효과를 볼 수 있다. 하지만 읽기 전용 파일시스템이라는 점과 RAMFS보다 속도가 떨어지는 단점이 있다.

3.6 Journaling Flash File System (JFFS)

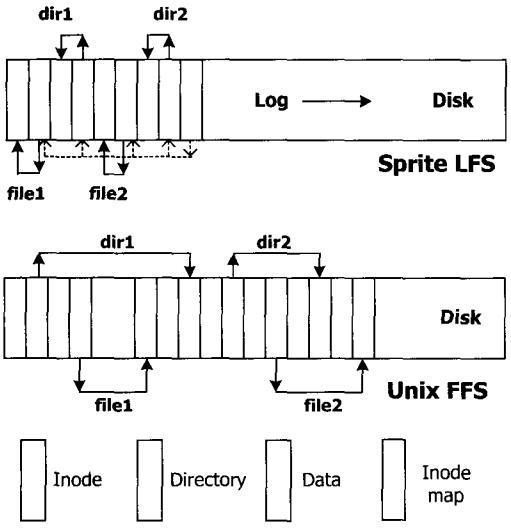
저널링 플래시 파일시스템은 비교적 대용량의 기억 장치인 플래시메모리에 적합한 파일 시스템을 목표로 개발되었다. 저널링 플래시 파일시스템은 로그 파일 시스템 구조를 채택하고 있기 때문에 시스템의 예기치 않게 에러에 대하여 파일시스템의 내용이 파괴되지 않는 장점을 가지고 있다.

4. 저널링 파일 시스템

4.1 로그 파일 시스템(LFS)

지난 수년간 CPU 성능, 메인 메모리 및 하드디스크의 용량은 급격하게 발전한 반면 하드디스크 접근 시간은 상대적으로 느리게 발전하여 왔다. 이러한 환경에서 로그 파일 시스템(Log-structured File System)은 기존의 파일 시스템의 다음과 같은 문제점을 극복하기 위하여 개발되었다.

- 각 파일에 대한 정보가 하드디스크 전반에 걸쳐 저장된다. 이러한 파일 시스템의 구조는 각 파일에 대한 정보를 읽어들이기 위해서 매우 많은 하드디스크에 대한 접근이 발생하므로 전체 시스템의 성능의 저하를 발생시킨다(그림 3).
- 파일의 내용에 변경이 발생되면 수정이 발생한 이후의 블록은 모두 다시 쓰기가 발생한다. 즉, 수정되지 않은 블록들도 수정된 블록의 영



(그림 3) LFS와 Unix FS의 비교

향을 받아 다시 쓰기를 수행하여야 한다. 이러한 특성은 파일의 수정시 시스템의 성능을 저하시키는 주요한 요인이 된다.

그러므로 로그 파일 시스템은 (그림 3)와 같이 단일 순차적 쓰기 기법을 통해 쓰기 작업의 성능을 향상시켰다. 즉, 수정시 기존의 파일을 직접 수정하는 것이 아니라 수정된 내용을 로그로 저장한다. 그리고 파일을 하드디스크에서 메모리로 읽어들일 때 저장된 로그를 이용하여 파일을 수정한다.

4.2 저널링 파일 시스템(JFS)

저널링 파일 시스템은 일정부분을 로그를 위해 남겨두어 백업(Backup) 및 복구 기능을 수행할 수 있는 파일 시스템이다. 즉, 로그 파일 시스템과 일반적인 파일 시스템을 혼용하여 사용한다. 저널링 파일 시스템은 사용자가 임의의 파일의 내용을 입력 또는 수정하면 그 내용을 바로 하드디스크에 기록하기 전에 관련 내용을 로그(log)에 기록하므로써 다양한 내용을 빈번하게 기록하고 수정하는 서버에서 수많은 사용자들이 입력하고 수정하는 내용이 예기치 못한 사고로 인하여 시스템이 다운(Down)되더라도 다시 복구할 수 있는 확률을 기존 Unix FS 시스템보다 상당히 향상시켰다. 그러나 저널링 파일 시스템은 로깅 기능에 따라 파일 시스템이 데이터가 수정될 때마다 로깅에 따른 수많은 자원을 필요로 하는 단점을 지닌다. 이러한 문제점은 무조건 저널링 파일 시스템을 사용하는 것보다는 필요한 부분에 한해서만 사용하는 것이 전체 시스템을 효율을 높일 수 있다.

저널링 파일 시스템에는 IBM사의 독자적인 파일 시스템인 'JFS(Journaling File System)', SGI의 'XFS (eXtended File System)', 독인의 한스 라이저(Hans Reiser)가 개발한 '라이저 파일 시스템(Reiser File System : 줄여서 reiserfs)', 리눅스 공동체에서 개발 중인 'ext3(Extension3)' 등이 있다. 이들 파일 시스템은 <표 2>와 같은 특성을 지닌다.

<표 2> JFS의 특성 비교

종류	MAX file system size	Block Size	Max file size
XFS	18 thousand petabytes	512 bytes ~ 64KB	18 thousand petabytes
JFS	512 bytes blocks / 4 petabytes	512, 1024, 2048, 4096 bytes	512 TB with 512 bytes block
	4 KB blocks / 32 petabytes		4 petabytes with 4 KB bytes block
ReiserFS	4KB blocks / 16TB	Up to 64 KB, Currently fixed 4KB	4GB
Ext3FS	4TB	1KB ~ 4KB	2GB

5. 플래시 파일 시스템

플래시메모리는 기존의 회전식 자기 매체에 비해서 속도가 빠르고, 충격에 강한 장점을 지니므로 기존의 가전, 통신 기기, 휴대 기기 등에서 저장 매체로써의 사용이 증대하고 있다. 또한 플래시메모리 기술이 발전함에 의해 플래시메모리의 용량이 증가하고 내장형 시스템에서 처리하여야 할 애플리케이션의 종류도 다양해짐에 의해 플래시메모리 관리 시스템은 일반적인 운영체제의 파일 시스템을 필요로 하고 있다.

저장 매체로써 플래시메모리는 앞에서 언급한 것과 같이 세 가지의 문제점을 가지고 있다. 즉, 데이터를 덧쓰기 위해서는 데이터를 저장하기 전에 플래시메모리를 EU 단위로 지워야 한다. 그러나 하나의 EU를 지우는 시간은 0.5초에서 1초로 많은 시간이 소요되며, 하나의 EU 내에는 여러 개의 블록이 존재하므로 변경되지 않은 블록까지 관리해야하는 오버헤더(Overhead)가 발생한다. 따라서 플래시메모리에 저장된 데이터를 수정할 때에는 많은 시간이 소요된다. 이러한 문제점을 해결하기 위

하여 많은 플래시메모리 파일 시스템은 기존의 로그 파일 시스템 방식으로 데이터를 저장하여 위와 같은 문제를 해결하고 있다. 또한 플래시메모리의 수명이 제한되어 있으므로 EU를 지우는 회수를 최적화 시켜야한다. 이런 문제를 해결하기 위하여 많은 논문에서 Cleaning Policy를 제안하고 있다.

스웨덴의 Axis Communications에서 개발된 JFFS v1(Journalling Flash File System version 1)은 플래시메모리 기술이 가지고 있는 제약을 고려하여 설계된 로그 파일 시스템이다. 이는 플래시메모리에 쓰기를 직접 수행함으로써 내장형 시스템이 비정상적인 종료가 발생하더라도 시스템의 데이터를 복구할 수 있는 기능을 가지고 있다.

Linux 개발사인 레드햇(Red Hat)에서 개발된 JFFS v2는 Linx 커널 2.4를 기반으로 하여 개발되었으며, JFFS v1의 이식성을 높이고 압축 저장 기능을 추가하였다. 그리고 Garba- ge collection 기능을 향상으로써 내장형 운영체제에서 플래시메모리 관리를 더욱 효과적으로 수행하고 있다.

또한 한국과학기술원에서 VFS(Virture File System)과 소용량 저장 매체에 적합한 FAT (File Allocation Table)를 이용한 플래시메모리 파일 시스템을 발표하였다[2].

이러한 플래시 파일 시스템은 플래시메모리의 쓰기 횟수의 제한으로 Cleaning policy를 채택하고 있다. 플래시 파일 시스템은 Cleaning policy에 따라서 언제 어떤 EU를 Cleaning할 지를 결정한다. Cleaning이 발생하는 경우는 일반적인 경우와 특수한 경우로 나눌 수 있다. 특수한 경우란 당연히 cleaning을 해야할 경우인데, 어떤 EU가 invalid 블록으로만 구성되어 있는 경우나 EUH의 validation code가 잘못 설정되어 있는 경우를 뜻한다. Invalid 블록만으로 구성된 EU나 EUH의 validation code가 잘못 설정되어 있는 EU가 있다면, 해당 EU를 cleaning한다. Cleaning policy에 사용되는 각 종 변수는 시스템 초기화 시에 정의된다. 플래시 파일 시스템의 성능 면에서는 cleaning은 최대한 연장되는 것이 좋다. 늦어지면 늦어질수록 invalid 블록이 많아지게 되고 결과적으로 valid 블록이 줄어들기 때문에 복사 비용이 낮아지게 된다. 반대로 플래시메모리 파일 시스템은 언제나 일정한 개수의 free EU를 가지고 있어야 한다. 그러므로 EU내에 있는 모든 블록이 invalid하지 EU를 삭제하여야 할 일반적인 경우가 발생한다. 모든 블록이 invalid하지 EU를 삭제하기 위해서는 백업(Back-up) EU를 이용하여 삭제될 EU내에 있는 valid한 블록을 복사한 후 선택되어진 EU를 삭제하여야 한다. 이때 백업 EU는 free EU 중에서 erase 횟수가 가장 적은 EU를 선택한다. 따라서 한 EU만 계속해서 사용되는 hot-spot 문제를 막을 수 있다. 즉, 한 EU만 계속해서 사용하여 erase 횟수가 증가하면, 전체 플래시메모리의 수명도 줄어들게 된다. 이와 같은 점을 고려하여 플래시 파일 시스템은 cleaning policy를 결정하여야 한다.

6. 결 론

플래시메모리의 견고성과 저 전력 사용, 빠른 접근속도, 작은 크기 등의 특성은 비싼 가격에도 불구하고 내장형 시스템의 저장매체로써 회전식 디스크 대신에 플래시메모리(Flash Memory)의 사용이 점차적으로 확대되고 있다. 그러나 데이터의 내용을 수정할 때 많은 시간이 소요되고, 수명도 영구적이지 못하다는 단점을 지니고 있으므로 내장형 운영체제의 파일 시스템은 위에서 언급한 플래시메모리의 특성과 내장형 시스템의 특성을 고려하여 설계되어야 한다.

본 고에서는 플래시메모리의 특성과 구조를 살펴보고 기존에 사용되고 있는 내장형 파일 시스템에 대하여 설명하였다. 플래시 파일 시스템으로 많이 채택되고 있는 로그 파일 시스템을 소개하였다. 또한 현재 내장형 운영체제에서 주요 관심사가 되

고 있는 플래시 파일 시스템의 예를 설명하고 플래시 파일 시스템 설계 시 고려해야될 항목에 대하여 알아보았다.

참고 문헌

[1] Atsuo Kawaguchi, Shingo Nishioka and Hiroshi Motoda, "A Flash-Memory Based File System," Proceedings of 1995 USENIX Techinical Conference, 1995

[2] 박상호, 안우현, 박대현, 김정기, 박승민, "플래시 메모리를 위한 파일 시스템의 구현," 정보과학회논문지:컴퓨팅의 실제 제7권 제5호 pp402~415, 2001.

[3] Mendel Rosenblum and John K. Ousterhout, "The Design and Implementation of a Log-Structured File System," ACM Transactions on Computer Systems, Vol. 10, pp.26~52, 1992.

[4] David Woodhouse, "JFFS : The Journalling Flash File System," http:// sources.redhat.com/jffs2, 2001.

[5] Juan I. Santos Florido, "Journal File System," http://www.linuxgazette.com/ issue55/florido.html.

[6] Steve Best and Dave Kleikamp, "JFS overview and layout white paper," http:// www-106.ibm.com/developerworks/library/jfs.html, http://www-106.ibm.com/develo perworks/library/jfslayout/index.html

[7] Adam Sweeney, Doug Doucette, Wei Hu, Curtis Anderson, Mike Nishimoto, and Geoff Peck, "Scalability in the XFS File System," USENIX conference, 1996. http://oss.sgi.com/projects/xfs/publications.html

[8] Philip Trautman and Jim Mostek, "Scalability and Performance in Modern File Systems," http://oss.sgi.com/projects/ xfs/papers/xfs_white/xfs_white_paper.html

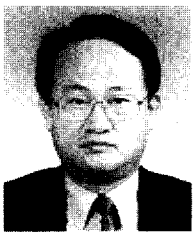
[9] Intel Corporation, "AP-684 Uuder- standing the Flash Translation Layer (FTL) Specification, 1998. http://www. intel.com/design/flcomp/applnots/ 297816.htm

[10] "3 volt FlashFile Memory 28F160S3 and 28F320S3, "Intel corporation, 1998.

저자약력

박 성 호

1996년 부산대학교 전자계산학과 졸업(학사)
1998년 부산대학교 전자계산학과 대학원 졸업(석사)
2002년 부산대학교 전자계산학과 대학원 졸업(박사)
관심분야 : 내장형 파일 시스템, VOD 시스템, 멀티미디어 통신, 인터넷 캐싱.
e-mail : shpark@melon.cs.pusan.ac.kr



정 기 동

1973년 서울대학교 졸업(학사)
1975년 서울대학교 대학원 졸업(석사)
1986년 서울대학교 대학원 계산통계학과 졸업(이학박사)
1990년-1991년 MIT, South Carolina 대학 교환 교수
1995년-1997년 부산대학교 전자계산소 소장
1978년-현재 부산대학교 전자계산학과 교수
관심분야 : 병렬처리, 멀티미디어
e-mail : kdchung@melon.cs.pusan.ac.kr