유효 페이지 색인 테이블을 활용한 NAND Flash Translation Layer 설계

신정환, 이인환
한양대학교 공과대학 전자전파통신공학과

Design of NAND Flash Translation Layer Based on Valid Page Lookup Table

Junghwan Shin, Inhwan Lee
Department of Electrical and Computer Engineering
Hanyang University
E-mail : {jhshin, ihlee}@csl.hanyang.ac.kr

Abstract

Flash memory becomes more important for its fast access speed, low-power, shock resistance and nonvolatile storage. But its native restrictions that have limited lifetime, inability of update in place, different size unit of read/write and erase operations need to managed by FTL(Flash Translation Layer). FTL has to control the wear-leveling, address mapping, bad block management of flash memory. In this paper, we focuses on the fast access to address mapping table and proposed the way of faster valid page search in the flash memory using the VPLT(Valid Page Lookup Table). This method is expected to decrease the frequency of access of flash memory that have an significant effect on performance of read and block-transfer operations. For the validations, we implemented the FTL based on Windows CE platform and obtained an improved result.

I. 서론

플래시 메모리는 현재 내장형, 컴팩트 플래시, 스마트 미디어 카드 등의 다양한 형태로 디지털 카메라나 MP3 플레이어, 고성능 휴대폰 등의 시스템에서 프로그램이나 데이터를 저장하는 기억장치로써 사용되고 있으며, 기존의 마그네틱 디스크에 비해 빠른 처리 속도, 저전력 소비, 내구성등에서의 우수성과 메모리 용량의 증가에 따라 그 응용 범위는 점점 다양해지고 있다. 플래시 메모리의 특징으로는 우선, 읽기, 쓰기, 삭제 등의 기능에 따른 처리 시간이 각각 다르며, 각 동작 단위가 읽기와 쓰기는 페이지(섹터) 단위로 이루어지고, 삭제는 블럭 단위로 동작되는 차이가 있다. 플래시 메모리는 셀 구조에 따라 NOR 형 플래시와 NAND 형 플래시로 크게 나누어 질 수 있는데, NOR 형 플래시는 임의의 번지를 읽을 수 있으나, 가격이 비싼 단점이 있고, NAND 형 플래시는 기본적으로 페이지 단위의 접근을 통한 액세스를 하기 때문에 읽기 동작이 느린 반면 빠른 쓰기, 삭제 동작과 NOR 형 플래시에 비해 가격이 저렴한 장점을 갖고 있다. 현재 일반적으로 사용되고 있는 NOR 형 플래시와 NAND 형 플래시의 각 동작에 따른 액세스 타임과 가격은 다음과 같다.

플래시 타입	처리 시간			가격
	읽기	쓰기	삭제	
NOR 플래시 (Intel 28F128J3A-150)	14.4μs	3.53ms	1.2s (128B)	14.5$ (128Mb)
NAND 플래시 (삼성 K9F1208U0M)	26.4μs	200μs	2ms (128B)	4.5$ (128Mb)

각 동작 단위는 1 page (512 Byte) 기준

표. 1. 플래시 메모리 처리 시간
Table. 1. Flash Memory Access Time

플래시 메모리는 삭제된 영역에 대해서만 1 에서 0 으로 변환시키는 쓰기 동작을 수행할 수 있기 때문에, 주어진 위치에서 실시간의 데이터 수정이 불가능한 구조를 갖고 있다 (Erase-Before-Write). 그러나, 삭제 동작은 페이지가 아닌 블록 단위로 이루어지기 때문에, 페이지의 데이터 갱신을 위해서는 새로운 물리적 페이지를 사용하여야 한다. 따라서, 하나의 논리적 페이지에 대한 물리적 페이지의 주소를 매핑 시켜주는 주소 매핑 테이블 (address mapping table)이 필요하며, 페이지 갱신시마다 주소 재매핑(address remapping) 수행이 이루어져야 한다. 그리고 플래시 메모리는 각 셀이 쓰기, 삭제에 대하여 제한된 수명을 갖고 있기 때문에 각 페이지의 사용 빈도에 대한 분배 정책(wear-levelling) 이 필요하다. 따라서, 이러한 주소 매핑과 페이지 분배 동작을 위하여 플래시 디바이스 드라이버와 호스트의 운영 체제 사이에서 이를 효과적으로 운영할 수 있는 Flash Translation Layer(FTL)를 구현하여 적용한다.

본 논문에서는 FTL 의 주소 매핑과 관련하여 보다 빠른 읽기 동작을 수행할 수 있는 FTL 을 구현하고자 한다. 이를 위하여 2 절에서는 FTL 의 주소 매핑과 관련된 기존 기술들을 조사해보고, 3 절에서 유효 페이지 색인 이라는 기법을 제시하여, 4 절에서는 이를 구현한 후 개선된 실험적인 결과를 얻어 보았다.

II. Related Work

주소 매핑 테이블은 운영체제의 파일 시스템에서 주어진 논리적 주소를 플래시 메모리의 물리적인 페이지의 주소로 변환시켜 주는 역할을 한다. FTL 은 주소 매핑 테이블 설계와 관련하여 주소 매핑 수준과 페이지 재 매핑 전략 등의 사항을 고려하여야 한다. 우선, 주소 매핑 수준의 관점에서 주소 매핑은 페이지 단위[2] 혹은 블록 단위[3] 수준으로 구현될 수 있는데, 페이지 단위 수준의 경우 플래시 메모리 크기에 따라 매핑 테이블의 용량을 과도하게 필요로 하는 단점이 있으며, 블록 단위 수준 매핑의 경우에는 작은 영역에 대한 빈번한 갱신 동작 시에 비효율적인 메모리 낭비가 발생한다.

이러한 블록 매핑을 바탕으로 주소 매핑 테이블의 설계와 운용에 관한 주요 기술로는 Mitsubishi 사[3]와 M-System 사[4]에서 제시한 기술들이 있다. Mitsubishi 기술은 각 블록의 페이지를 논리적 주소의 오프셋에 매핑시키고, 유효한 데이터를 저장하고 있는 페이지의 내용 갱신시 새로운 페이지를 할당하는 용도로 블록 내에 예비 페이지 영역을 설정하는 기술이다[3].

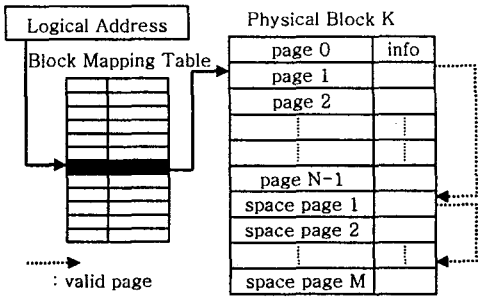

그림 1 Mitsubishi FTL 주소 매핑 개념도

Fig. 1 Mitsubishi FTL Address Mapping Diagram

위의 그림 1 에서 나타난 것처럼 예비 페이지 영역은 하나의 블록 내에서 M 개수의 페이지들이 비어 있는 상태로 존재하며, 유효한 데이터를 포함하고 있지 않는 상태에서는 논리적 주소에 매핑 되지 않는 영역이다. 유효한 데이터를 포함하고 있는 페이지는 해당 페이지의 논리적 주소와 페이지의 데이터 저장 여부 및 데이터의 유효성 플래그 등의 상태정보(페이지 ID)를 페이지 영역 내에 기록하게 된다. 페이지가 비어있지 않은 영역에 페이지 갱신을 하는 경우에 예비 페이지 영역에서 새로운 페이지를 할당 받고 현재 페이지와 이전 페이지의 페이지 ID 영역을 갱신한 후, 이전 페이지의 페이지 ID 에는 데이터가 유효하지 않음을 기록한다. 이러한 새로운 페이지 할당은 예비 페이지 영역을 모두 소진할 때까지 순차적으로 이루어 질 수 있다.

따라서, 페이지 읽기를 위해서 FTL 은 논리적 주소에서 계산된 오프셋에 해당하는 물리적 페이지부터 시작하여, 예비 페이지 영역의 모든 페이지 ID 들을 순차적으로 검색하고 이 과정에서 유효한 페이지를 만나는 경우에 데이터를 읽는다. 이러한 데이터 읽기 과정은 예비 페이지 영역이 모두 소진되어 블록 이동을 하여야 할 경우에도 마찬가지로 적용된다.

M-System 사에서 제시한 새로운 페이지 할당 기술(FMAX)은 예비 페이지 영역을 다른 블록 내에 설정하는 방법이다[4]. 이러한 블록 설정을 위하여 FTL 은 각 논리적 블록에 대하여 물리적 블록을 1:2 의 집합 연관 매핑을 하여 설정하며, 이는 하나의 논리적 페이지에 2 개의 물리적 페이지가 매핑되는 기법이다.

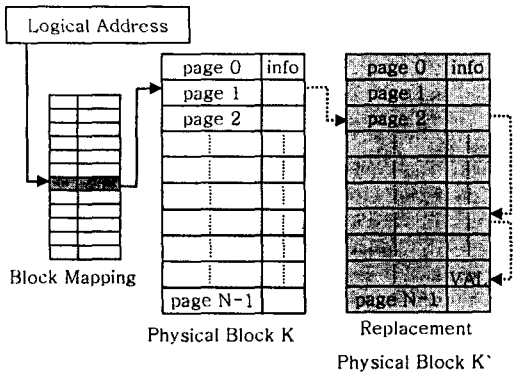

그림 2 M-System FTL 주소 매핑 개념도

Fig. 2 M-System FTL Address Mapping Diagram

위에서 제시된 기술들에서 기본적으로 읽기와 쓰기 동작 및 새로운 블록으로의 페이지 이동 시에 '유효 페이지 검색' 과정을 반드시 필요로 한다. 본 논문에서는 블록 단위 수준의 매핑을 바탕으로 유효 페이지 정보를 효과적으로 관리하여 대용량의 NAND 형 플래시 메모리에서 성능향상을 기대할 수 있는 방법을 제시하고자 한다.

III.유효 페이지 색인 테이블

FTL 에서 플래시 메모리의 읽기와 쓰기 동작은 유효한 데이터를 갖고 있는 페이지에 대한 접근을 포함하기 때문에, FTL 에서 각 블록의 유효 페이지 검색에 소요되는 시간이 플래시 메모리 동작 성능을 결정짓는 중요한 요소가 될 수 있다. 위 그림 1 에서 나타난 것처럼 기존의 기술은 처음 매핑된 논리적 오프셋 페이지부터 시작하여, 예비 페이지 영역들의 페이지 ID 를 순차적으로 읽어가면서 논리적 주소에 해당하는 페이지를 검색한다. 이러한 페이지 스캔을 바탕으로 한 유효 페이지 검색 방법은 갱신이 반복적으로 자주 발생하는 페이지에 대해서 심각한 오버헤드를 야기할 수 있다. 즉, 하나의 페이지를 읽거나 쓰기 위해 유효 페이지 검색을 하는 과정에서 최대 M+1 번의 페이지 읽기를 수행하여야 하는 문제점이 발생한다.

본 논문에서는 이를 개선하기 위한 방법으로 다음과 같은 유효 페이지 정보를 저장(lookup)하는 별도의 유효 페이지 색인 테이블(Valid Page Lookup Table)을 제안한다. 이 유효 페이지 색인 테이블은 블록 수준의 매핑을 기본으로 Mitsubishi 의 예비 페이지 영역 기법을 사용하며, 각 물리적 블록이 논리적 블록에 할당될 때마다 동적으로 하나씩 생성되어 사용된다. 유효 페이지 색인 테이블의 구조와 검색 방법은 아래 그림 4 와 같이 표현할 수 있다

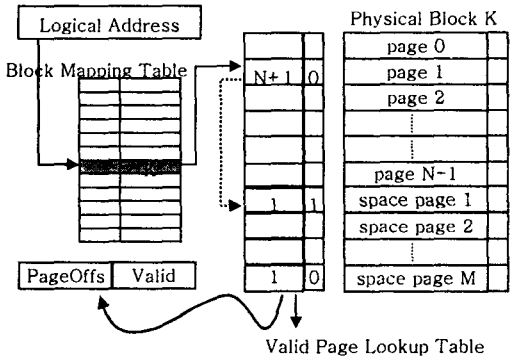

그림 3 유효 페이지 색인 테이블 상태도

Fig. 3 Valid Page Lookup Table Diagram

유효 페이지 색인 테이블에는 각 비트별로 각 물리적 페이지에 대한 유효성을 표시하는 부분(ValidFlag)과, 각 페이지별로 물리적 주소의 오프셋값을 저장하는 바이트 단위의 오프셋 배열(PageOffset[])로 구성된다. 유효 페이지 색인 테이블은 논리적 블록 사이즈에 상관없이 실제 물리적 블록 사이즈 전체를 순차적으로 관리하는데, 위 그림에서 물리적 블록 사이즈는 N+M 이다. 이 테이블에서 페이지가 유효함을 표현하기 위해서는 ValidFlag 가 1 이고 PageOffset 값이 논리적 페이지 오프셋 번호이어야 한다. 따라서, 위의 그림에서처럼 쓰기 동작시에 새로운 페이지(N+1 번)를 할당 받아 사용하는 경우, 그 이전 페이지(N+M)에 해당하는 ValidFlag 는 0 으로 클리어 시켜주고, 원래의 물리적 오프셋 페이지(1)의 PageOffset 에는 새로운 페이지 (N+1)을 기록한다.

그러므로, 읽기 동작시에는 논리적 페이지 오프셋 1 번의 PageOffset 을 읽고 물리적 오프셋 N+1 번 페이지의 데이터를 한번에 읽어 올 수 있게 된다. 즉 유효 페이지 검색을 위한 스캔작업이 필요없게 된다.

본 논문에서 제시한 이 유효 페이지 색인 테이블은 RAM 영역에 존재하게 되는데, 전원이 새로 켜진 경우에는 각 페이지의 여유 공간에 기록 되어있는 페이지 ID 를 읽어서 각 페이지에 대한 정보들을 셋업 시켜야하는 단점이 존재한다. 그러나, 유효 페이지 색인 테이블을 활용한 읽기 동작이 일반적인 페이지 읽기 뿐만 아니라, 블록 이동시에도 적용될 수 있기 때문에 전체적인 성능 향상을 기대할 수 있다.

IV. 구현 및 성능평가

4.1 구현

본 논문에서는 기존의 유효 페이지 검색 방법을 활용한 FTL과 본 논문에서 제시한 유효 페이지 매핑 테이블을 적용한 FTL을 각각 구현하여 그 성능을 비교 하였다. 구현 환경은 하드웨어 플랫폼으로 삼성 SMDK2410 평가 보드를 사용하였는데, 여기에는 ARM920T 코어를 적용한 S3C2410 32비트 RISC CPU와 삼성 K9F1208U0M NAND 형 플래시를 채용한 SMC 카드를 포함하고 있다. 운영체제로 마이크로소프트사의 Windows CE .NET 4.2를 사용하여 Windows CE 의 Block Device Driver와 인터페이스가 가능한 FTL을 플래시 디바이스 드라이버와 함께 포팅하였으며, FAT 파일 시스템을 적용하였다.

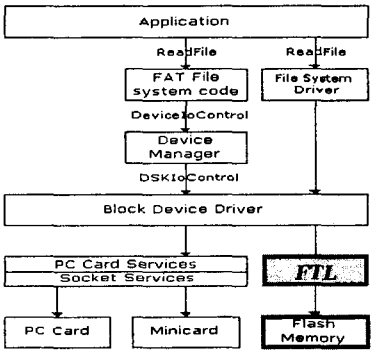

그림 4 Windows CE.NET 블록 디바이스 드라이버 구조
Fig. 4 Windows CE.NET Block D-Driver Architecture

4.2 성능평가

Windows CE .NET 4.2 상에서 새로 구현된 Flash Driver에 대하여 기존의 스캔 검색방법과 본 논문에서 제시한 유효 페이지 색인 검색방법에 대한 다음의 테스트를 진행하여 각각의 FTL에 대한 성능 결과를 얻었다.

	스캔 검색			유효 페이지 캐싱 검색		
	Read	Write	Erase	Read	Write	Erase
Test A	1937	497	5	96	508	5
Test B	2155	539	6	128	556	6

Test A에서는 Flash 메모리에 1MB크기의 16개의 파일에 대하여 32회의 읽기 쓰기 동작을 반복하였고, Test B에서는 크기가 다른 파일과 디렉토리를 각각 16개씩 생성 및 삭제 하는 방법으로 테스트를 진행하여 각각의 FTL에 대해 읽기, 쓰기 및 삭제에 대한 Flash 접근 동작 횟수를 기록하여 비교하였다.결과표에서 보여지듯이 유효 페이지 캐싱 검색 기법이 기존의 스캔 검색 기법에 비해 플래시 메모리를 직접 액세스 하는 읽기 동작 회수가 82% 정도 감소하여 전체적인 성능 향상에 기여 하였음을 알 수 있다.

V. 결론 및 추후 연구

휴대용 시스템에서 저장 매체로써 점차 중요성과 활용이 증가하고 있는 플래시 메모리에 대하여 플래시 메모리의 특징을 반영한 Flash Translation Layer 를 Windows CE .NET 상에서 플래시 디바이스 드라이버와 함께 구현하고, FTL 에서의 유효 페이지 검색을 개선하는 방법을 제안 및 적용해 보았다. 특히 빈번한 데이터 수정이 일어나는 환경에서 유효 페이지에 대한 정보를 색인하고 활용함으로써 기존의 방법에 비해 플래시 메모리에 대한 액세스 횟수를 감소시키는 성능 개선을 얻을 수 있었다. 향후에는 테스트 방법을 개선하여 Windows CE .NET 에서 제공하는 'VOBench' 프로그램을 활용하여 임의 크기 데이터의 읽기, 쓰기, 삭제 동작 등의 수행시간에 대한 보다 객관적인 결과값을 얻고자 한다.

참고문헌

[1] Intel Co., "Understanding the Flash Translation Layer (FTL) Specification", AP-684.

[2] A.Ban, "Flash file system", United States Patent, no.5,404,485.

[3] T.Shinogara, "Flash memory card with block memory address arrangement", United States Patent, no. 5,905,993.

[4] A.Ban, "Flash file system optimized for page-mode flash technologies", United States Patent, no.5,937,425.

[5] Samsung Electronics Co., "S3C2410X01 Risc Microprocessor", http://www.samsung.com/Products/Semiconductor/SystemLSI/MobileSolutions/MobileASSP/MobileComputing/S3C2410X/S3C2410X.htm.

[6] Samsung Electronics Co., "64M x 8 Bit NAND Flash Memory".http://www.samsung.com/Products/Semiconductor/Flash/NAND/512Mbit/K9F1208U0M/K9F1208U0M.htm.

[7] Microsoft Co., "Microsoft Windows CE .NET 4.2 Block Drivers", http://msdn.microsoft.com/library/en-us/wceddk40/html/cxconBlockDrivers.asp.