에너지 사용량 측정이 가능한 플래시 메모리 시뮬레이터

최원철, 송준영, 이기혁, 한형진, 손기락
한국외국어대학교 컴퓨터공학과
e-mail : wc.choi@hufs.ac.kr

Energy Consumption Checkable Simulator for Flash Memory

WonChul Choi, JunYoung Song, KiHyuk Lee, HyungJin Han, Kirack Sohn
Dept of Computer and Information Communication Engineering, HanKuk University of Foreign Studies

요 약

플래시 메모리는 센서 네트워크가 가진 제약 조건을 만족하는 저장 매체이다. 그러나 센서 네트워크에 필요한 프로그램의 개발 환경은 불편하고, 가장 중요한 이슈인 에너지 소모에 대한 편리하고 정확한 측정 수단이 없는 것이 현실이다. 그에 따라 본 논문은 에너지 소모량의 측정을 지원하고 여러 플래시 메모리 환경을 조절하여 실험을 할 수 있는 플래시 메모리 소프트웨어 시뮬레이터를 설계 및 구현하였다.

1. 서론

최근 기술발전이 유비쿼터스 기반의 컴퓨팅으로 변환됨에 따라 휴대폰, PDA, 스마트카드, 디지털 카메라와 같은 이동 컴퓨팅 장치의 사용이 급증하면서 플래시 메모리에 대한 관심이 증가하고 있다.

플래시 메모리는 내부 소자들의 구성 형태에 따라 다양한 종류가 있는데, 그 중 가장 많이 사용되고 있는 것은 NOR형 플래시 메모리와 NAND형 플래시 메모리이다. NAND 플래시는 NOR 플래시에 비하여 집적도가 높아 같은 크기로 대용량을 제작하는 것이 가능하고, 동일 용량을 제작하는 가격이 상대적으로 저렴하여 NAND 플래시에 대하여 많은 연구가 진행되고 있다[1].

센서 네트워크를 구축할 때, 가장 많은 전력을 소모하는 부분은 통신 부분이다. 통신이 소모하는 에너지양은 저장에 필요한 그것의 100배이다. 그래서 통신을 줄이고 데이터를 센서 Mote의 저장소에 저장하는 방식이 에너지 효율적인 센서 네트워크 구축에 주효하다.[2] 센서 네트워크를 구축하는 Mote의 작은 크기와 제한된 에너지, 그리고 각 Mote의 가격을 줄이는 것이 중요하다는 제한조건을 고려할 때 플래시 메모리, 그 중에서도 NAND형 플래시 메모리가 센서 네트워크에 사용될 Mote의 저장소로 적절하다.

그러나 센서 네트워크에 사용할 프로그램을 Mote에서 개발하는 환경은 매우 불편하며, 소프트웨어를 개발한 경우에도 플래시 메모리는 제조업체에 따라 특성이 상이하여 모든 환경을 적용하기 어렵다. 특히 에너지 소모량, wear-level을 측정할 경우 직접 센서를 구동하여 측정하는 경우에 비하여 소프트웨어 시뮬레이터를 사용하는 것이 편리하고 정확한 값을 측정할 수 있다[3]. 현재, 센서 노드용으로 에너지 소모에 대한 측정 기능을 지원하는 플래시 메모리 소프트웨어 시뮬레이터는 존재하지 않는다.

본 논문에서는 위의 요구사항을 만족시키기 위하여 플래시 메모리 시뮬레이터(ECFMSIM)를 설계를 할 때 다음의 사항을 목표로 하였다.

본 연구는 21 세기 프론티어 연구개발사업의 일환으로 추진되고 있는 정보통신부의 유비쿼터스 컴퓨팅 및 네트워크 원천기반 기술 개발사업의 지원을 받았음

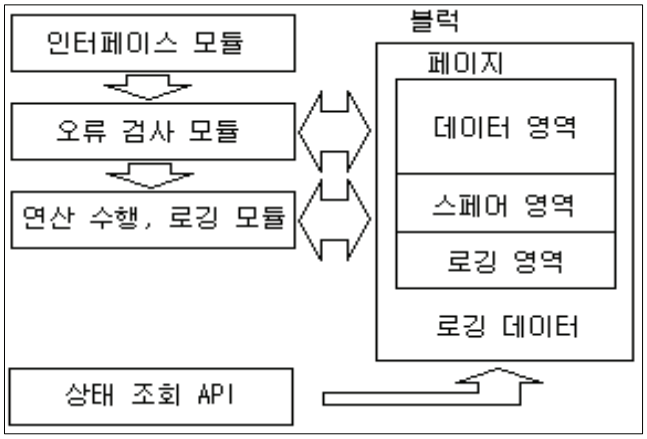
• 에너지 소모량의 측정 : 플래시 메모리에서 이루어지는 읽기, 쓰기, 지우기의 연산으로 인한 페이지, 블록 및 전체 디바이스의 에너지 소모량을 측정할 수 있다.

• 장치 상태 확인 : 특정 위치의 쓰기 상태나 에너지 소모 등의 직접적인 값이나 통계를 조회할 수 있다.

• 장치 환경 변경 가능성 : 페이지 크기, 블록 당 페이지 수, 장치의 크기 등에 대한 여러 가지 환경 값을 변경하여 다양한 장치 환경에 대해서 실험을 진행할 수 있다.

2. 설계

위에서 언급한 기능을 가진 시뮬레이터를 ECFMSIM(Energy Consumption Checkable Flash Memory Simulator)이라 한다. ECFMSIM의 각 모듈과 데이터가 갖는 관계에 대한 구조도는 (그림1)과 같다.



(그림 1) ECFMSIM 구조도

2.1 인터페이스 모듈

상위 계층에서 접근할 수 있는 아래와 같은 인터페이스를 제공한다.

```
read(unsigned long from, int len, unsigned char *buf)
write(unsigned long to, int len, unsigned char *buf)
read_oob(unsigned long off_set, int len, unsigned char *buf)
write_oob(unsigned long off_set, int len, unsigned char *buf)
read_ecc(unsigned long from, int len, unsigned char *buf,
        unsigned char *eccbuf)
write_ecc(unsigned long to, int len, unsigned char *buf,
        unsigned char *eccbuf)
erase(unsigned long off_set, int len)
```

read()는 플래시 메모리의 페이지 번호 'from'에서 'len' byte만큼의 데이터를 'buf'에 읽어오며, write()는 페이지 번호 'to'에 'len' byte만큼 'buf'의 내용을 기록한다. read_oob(), write_oob() 함수는 'off_set'가 가리키는 스페어 영역에 그 연산을 수행하는 점이 read(), write()와 다르다. read_ecc(), write_ecc() 함수는 데이터와 스페어 영역을 동시에 접근하며 데이터 영역은 'buf', 스페어 영역은 'eccbuf'로 접근한다. erase() 함수는 'off_set'부터 'len' byte를 삭제한다.

2.2 오류 검사 모듈

오류 검사 모듈은 주소의 적합성, nop(부분 프로그램 횟수 제한) 및 wear-level에 의한 마모도 검사를 한다.

2.3 연산 수행, 로깅 모듈

연산 수행 모듈은 플래시 메모리 연산을 수행한다. 로깅 모듈은 블록이나 페이지에 그 횟수나 에너지 소모량을 기록하고 상태 조회 API로 열람한다.

2.4 상태 조회 API

상태 조회 API는 특정 블록이나 페이지의 사용 상태 또는 에너지 소모량을 지정하여 확인할 수 있도록 지원한다. 조회할 수 있는 항목은 구현에 대한 부분에서 설명한다.

3. 구현

3.1 물리 메모리(physical memory) 관리

ECFMSIM은 다음과 같은 구조의 메모리 공간을 확보한다. 데이터 및 페이지 공간의 크기는 설정 파일의 특성 값을 따른다. 각 페이지마다 read, write, program 횟수를 저장하는 공간을 준비하며, 블록은 erase의 횟수, 삭제 여부 및 손상 여부를 기록하는 공간을 갖는다.

```
struct FM_page{
  char data[FM_data_size + FM_spare_size];
  unsigned long int FM_write_count;
  unsigned long int FM_read_count;
  unsigned short int FM_page_count;
  unsigned long int FM_EC_page;
}
struct FM_block{
  FM_page FM_pages[FM_page_per_block];
```

```
  unsigned long int FM_erase_count;
  bool FM_erased;
  bool FM_damaged;
}
struct FM_device{
  FM_block FM_blocks[FM_block_per_device];
}
```

3.2 ECFMSIM의 내부 모듈의 구현

• 인터페이스 모듈

인터페이스 모듈은 상위 계층에서 호출하게 되면, 제공받은 인자를 ECFMSIM에서 관리하는 메모리 주소 방식으로 변환하여 오류 검사 모듈을 호출한 뒤, 오류가 없을 경우에 연산 수행, 로깅 모듈을 차례로 호출한다. 그리고 그 결과를 상위 계층에 돌려준다. 이로 인하여 인터페이스 모듈은 실질적인 수행 처리 모듈의 의미를 갖는다.

• 오류 검사 모듈

오류 검사 모듈에서 검사하는 작업들은 오류가 발생하는 경우 오류를 알리기 위하여 원인에 해당하는 값을 되돌린다. 오류 검사 모듈은 상태 조회가 가능하도록 로깅 영역에 기록한다. 수행 과정을 간략히 설명하면 다음과 같다.

① 모든 연산이 주소가 영역을 벗어나는지 검사.
② write 연산이 수행될 페이지가 부분 쓰기 횟수를 초과했는지 검사.
③ write, erase 연산의 wear-level 값 초과 검사.
※ 모든 검사는 오류 시 그 원인 값을 return.

• 연산 수행, 로깅 모듈

연산 수행 모듈은 오류 검사를 마친 인자를 사용한 연산을 수행하며 각 연산의 수행 횟수 및 에너지 소모량을 로깅 영역에 기록한다. write_oob()의 수행 과정의 예는 다음과 같다.

① 지정된 페이지에서 len만큼 ②를 반복.
② buf의 0에 해당하는 값을 0으로 변경.
③ 스페어 영역에서 ④를 반복.
④ eccbuf의 0에 해당하는 값을 0으로 변경.
⑤ FM_write_count, FM_page_count 값을 증가.

• 상태 조회 API

상태 조회 API는 ECFMSIM에서 수행 처리와는 그 역할이 구분된다. ECFMSIM이 작업을 하며 로그 데이터를 남기면, 상태 조회 API는 그것을 통하여 조회에 대한 결과를 제공한다. 조회할 수 있는 값은 다음과 같다.

- wear-level에 따른 전체 사용량의 표준 편차
- 페이지, 블록, 전체 단위의 에너지 소모량
- 전체 디바이스에서 최대 wear-level 값
- read, write, erase 연산의 각각의 수행 횟수
- 각 연산에 의한 에너지 소모량

다음은 Toshiba의 TC58DVG02A1FT00 1Gb (128MB) NAND 플래시 메모리[4]의 연산에 대한 에너지 소모량을 구하는 식이다.

$$W(d) = 24.54 + d \cdot 0.0962\mu J$$
$$R(d) = 4.07 + d \cdot 0.105\mu J$$

24.54와 4.07은 연산이 수행될 때 소모되는 기본 값이다. d는 연산을 수행한 byte 수이고, 0.0962와 0.105는 byte에 비례하여 소모되는 에너지량이다. erase 연산의 경우 그 과정을 살펴보면 write 연산이 블록 단위로 이루어지므로 위 식을 통해서 소모량을 계산할 수 있다[2].

• Configuration

ECFMSIM은 설정 파일의 값의 변경으로 플래시 메모리의 특성 값을 변경할 수 있다. 특성 값은 다양한 항목의 값이 존재하는데, 일부 플래시 메모리에 대한 값은 기본 설정 번호를 선택하여 그 특성 값을 이용할 수 있다.

아래의 예는 주어지는 특성 값을 사용하지 않고 (=FM_conf_table 0) 직접 각 특성 값을 입력하여 사용하는 예이다.

```
FM_conf_table 0        // 사용할 설정 테이블 번호
FM_read_FEC 4070   // read 연산 고정 소모량(nJ)
FM_write_FEC 24540// write 연산 고정 소모량(nJ)
FM_read_EC 96.2     // read 연산 비례 소모량(nJ)
FM_write_EC 105    // write 연산 비례 소모량(nJ)
FM_Nop 4               // 부분 프로그램 제한 횟수
FM_wear_level 100000          // 쓰기 한계 횟수
```

```
FM_data_size 512          // 페이지의 크기(Byte)
FM_spare_size 16          // 스페어 영역의 크기(Byte)
FM_page_per_block 32      // 한 블록의 페이지의 수
FM_total_block 2000       // 총 블록 수
```

4. 결론 및 향후 과제

ECFMSIM은 다양한 설정 값의 사용으로 실제 메모리에 가까운 소프트웨어의 동작 실험 환경을 제공한다. 또한 소프트웨어 동작에 의한 영역의 사용에 따른 에너지 소모에 대한 통계 및 물리적, 논리적 에러에 대한 검출 등 다양한 성능 분석 환경을 제공한다.

현재, ECFMSIM은 다음과 같은 한계성을 가진다. 확실한 데이터로 산출된 손상된 블록의 수치가 없고 그 이유도 정확히 알지 못한다. 그래서 손상된 블록에 대한 대처 방안이 정확치 못하다. 그리고 ECC부분에 있어서도 각 제조 회사들 사이의 정확한 규격 명세가 없어서 특정 회사의 제품만 가능하다. 그래서 다른 회사의 제품을 사용 시에는 프로그램의 특정 부분을 수정하여야 한다.

그래서 이러한 부분을 보완하기 위해서는 각 제조 회사들의 제품을 각각을 분석하여 제품의 타입을 정의해야 하고 더욱 정확한 손상된 블록의 산출 방식과 효율적인 대체 방안이 필요할 것이다.

참고문헌

[1] Soo Kwan Lee, Sang Lyul Min and Yoo Kun Cho, “플래시 메모리 관련 최근 기술 동향”, 한국정보과학회, 정보과학회지 제24권 제12호, pp. 99~106, 2006, 12.

[2] Gaurav Mathur, Peter Desnoyers, Deepak Ganesan and Prashant Shenoy, “Capsule: An Energy-Optimized Object Storage System for Memory-Constrained Sensor Devices”, Proceedings of the Fourth ACM Conference on Embedded Networked Sensor Systems (SenSys), Boulder CO, November 1-3, 2006.

[3] 정재용, 노삼혁, 민상렬, 조유근, “플래시 메모리 시뮬레이터의 설계 및 구현”, 한국정보과학회, 정보과학회논문지:컴퓨팅의 실제 제8권 제1호, pp. 36~45, 2002, 2.

[4] Toshiba America Electronic Components, Inc. (TAEC), www.toshiba.com/taec . Datasheet : TC58DVG02A1FT00, Jan 2003.