2013년 6월 전자공학회 논문지 제 50 권 제 6 호
Journal of The Institute of Electronics Engineers of Korea Vol. 50, NO. 6, June 2013 http://dx.doi.org/10.5573/ieek.2013.50.6.188

논문 2012-50-6-22

플래시 저장장치 컨트롤러 시스템을 위한 동적 낸드 오퍼레이션 스케줄링

(Dynamic NAND Operation Scheduling for Flash Storage Controller Systems)

정 재 형*, 송 용 호**

(Jaehyeong Jeong and Yong Ho Song©)

요 약

낸드 플래시 메모리 기반 저장장치는 성능을 높이기 위하여 내부에 다수의 플래시 메모리가 공유하는 데이터 버스들을 구성하고, 이 구조를 이용하여 다수의 플래시 메모리 오퍼레이션을 동시에 수행하는 병렬 기법을 사용한다. 저장장치의 성능은 개별 데이터 버스의 성능에 의하여 많은 영향을 받기 때문에, 저장장치 컨트롤러가 오퍼레이션을 효과적으로 스케줄링 함으로써 버스의 성능을 높이는 것이 중요하다. 그러나 오퍼레이션 별로 상이한 동작시간과 버스사용 특성으로 인하여 시시각각 변화하는 버스의 상황은 스케줄링을 어렵게 만든다. 또한 단순히 버스 사용효율을 높이기 위한 스케줄링 기법은 예상하지 못한 오퍼레이션의 지연과 저장장치의 자원 낭비를 초래할 수 있다. 본 논문에서는 데이터 버스의 성능과 저장장치의 자원 효율을 고려한 동적인 오퍼레이션 스케줄링 기법들을 제안한다. 제안하는 기법들은 오퍼레이션을 세 단계로 구분한 후 오퍼레이션의 특성과 데이터 버스의 상황에 따라 이들을 스케줄링 한다. 제안된 기법들을 컨트롤러에 적용하여 FPGA 플랫폼에서 검증한 결과, 제안된 기법을 적용한 컨트롤러는 정적인 스케줄링 기법을 사용하는 컨트롤러에 비하여 쓰기 오퍼레이션의 수가 1.9% 줄어들었으며 4-7% 높은 버스 사용효율과 4-19% 높은 처리량을 보였다.

Abstract

In order to increase its performance, NAND flash memory-based storage is composed of data buses that are shared by a number of flash memories and uses a parallel technique that can carry out multiple flash memory operations simultaneously. Since the storage performance is strongly influenced by the performance of each data bus, it is important to improve the utilization of the bus by ensuring effective scheduling of operations by the storage controller. However, this is difficult because of dynamic changes in buses due to the unique characteristics of each operation with different timing, cost, and usage by each bus. Furthermore, the scheduling technique for increasing bus utilization may cause unanticipated operation delay and wastage of storage resource. In this study, we suggest various dynamic operation scheduling techniques that consider data bus performance and storage resource efficiency. The proposed techniques divide each operation into three different stages and schedule each stage depending on the characteristics of the operation and the dynamic status of the data bus. We applied the suggested techniques to the controller and verified them on the FPGA platform, and found that program operation decreased by 1.9% in comparison to that achieved by a static scheduling technique, and bus utilization and throughput was approximately 4 - 7% and 4 - 19% higher, respectively.

Keywords : NAND Flash Memory, Storage, Controller, Operation Scheduling

* 학생회원, 한양대학교 전자컴퓨터통신공학과
(Dept. of Electronics and Computer Engineering, Hanyang University)
** 정회원, 한양대학교 융합전자공학부
(Dept. of Electronic Engineering, Hanyang University)
※ 본 논문(저서)은 산업통상자원부 기술혁신사업 (융복합혁신반도체기술개발, 10039204)으로 지원된 연구임.
© Corresponding Author(E-mail: yhsong@enc.hanyang.ac.kr)
접수일자 2013년4월29일, 수정완료일 2013년5월21일

Ⅰ. 서 론

낸드 플래시 메모리 기반 저장장치는 개별 디바이스의 성능이 낮은 플래시 메모리의 단점을 극복하기 위하여, 저장장치 내부에 다수의 데이터 버스와 이를 공유하는 디바이스들을 구성하고, 디바이스들이 동시에 작업을 수행하는 병렬 기법[1~2]을 사용한다. 병렬 기법은 호스트 요청을 플래시 메모리의 데이터 동작인 오퍼레이션(operation)으로 변환하는 FTL (Flash Translation Layer)과 플래시 메모리에게 오퍼레이션을 전달하고 오퍼레이션의 실행 과정을 제어하는 저장장치 컨트롤러에 의하여 이루어진다.

병렬 기법의 효과를 높이려면 FTL이 저장장치의 플래시 메모리 구조를 활용하여 오퍼레이션을 버스에 골고루 분배하고, 오퍼레이션을 전달받은 플래시 메모리들이 오퍼레이션을 동시에 병렬 수행할 수 있도록 컨트롤러가 오퍼레이션들을 스케줄링 하여야 한다.

서로 다른 데이터 버스에 속한 디바이스들은 오퍼레이션을 수행함에 있어서 상호 독립적으로 동작 가능하기 때문에 버스 사이에 오퍼레이션에 대한 스케줄링을 필요로 하지 않는다. 그러나 버스 내부의 디바이스들에 대한 오퍼레이션들은 컨트롤러가 버스 사용효율을 고려하여 오퍼레이션을 스케줄링 하는 것이 중요하다. 이들을 스케줄링 하는 컨트롤러의 기법에 의하여 버스의 사용 효율이 달라지기 때문이다. 더욱이 버스 사용효율이 낮다면 개별 버스의 성능이 떨어질 뿐만 아니라 저장장치의 전체 성능도 낮아지기 때문에 컨트롤러의 오퍼레이션 스케줄링 기법은 병렬 기법을 이용한 저장장치 성능향상에 있어서 매우 중요한 요소이다.

그러나 플래시 메모리는 읽기, 쓰기, 소거 오퍼레이션에 대한 시간적 비용이 서로 다르고, 그 비용을 예측하기 어렵기 때문에 이를 효과적으로 스케줄링 하는 것은 쉽지 않다. 더욱이 오퍼레이션들의 실행과정 중에 플래시 메모리들이 공유하는 데이터 버스를 사용하기 위하여 발생하는 지연과 오퍼레이션마다 서로 다른 데이터 버스 사용 특성은 스케줄링을 더욱 어렵게 만든다. 또한 기존의 저장장치 컨트롤러의 오퍼레이션 스케줄링 기법은 단순히 채널의 사용효율을 높이는데 초점을 맞추고 있으며, 동적인 데이터 버스의 상황, 개별 오퍼레이션의 응답성 그리고 스케줄링이 저장장치의 자원에 미치는 영향 등에 대하여 고려하지 않는다[1~2].

본 논문에서는 낸드 플래시 메모리 기반 저장장치의 컨트롤러 시스템을 위한 동적인 오퍼레이션 스케줄링 기법들을 제안한다. 제안하는 기법들은 데이터 버스의 사용효율과 처리량을 높임으로서 저장장치의 성능을 향상시키고, 버퍼 공간을 점유하고 있는 오퍼레이션을 선행하도록 스케줄링 함으로써 저장장치의 자원 효율성을 높이기 위한 것이다. 이를 위하여 플래시 메모리 오퍼레이션을 동작 특성에 따라 세분화한 다음, 데이터 버스 사용 특성을 고려하여 세분화된 동작별로 버스 사용 시간과 권한에 차등을 두고 버스의 상황에 따라 이들을 동적으로 스케줄링을 한다. 제안된 기법을 컨트롤러에 적용하여 FPGA에서 검증한 결과, 정적인 컨트롤러에 비하여 4-7% 높은 버스 사용효율과 4-19% 높은 처리량을 보였으며, 버퍼 공간의 낭비를 줄여 쓰기 오퍼레이션을 1.9% 줄어든 것을 확인할 수 있었다.

본 논문의 구성은 다음과 같다. Ⅱ장에서는 배경지식에 대하여 언급하고 Ⅲ장에서는 본 논문에서 제안한 낸드 플래시 메모리 오퍼레이션의 동적 스케줄링 기법들을 제안한다. 또한 제안된 기법들을 적용한 컨트롤러 구조에 대하여 설명하고 Ⅳ장에서는 실험을 통하여 제안한 기법들의 성능을 평가하고, 마지막으로 Ⅴ장에서는 결론에 대하여 서술한다.

Ⅱ. 배경지식

1. 낸드 플래시 메모리와 오퍼레이션

낸드 플래시 메모리 디바이스는 하나 이상의 다이(die)을 포함하고, 다이는 하나 이상의 플래인(plane)과 이를 제어하기 위한 제어 유닛(control unit)으로 구성된다. 플래시 메모리 다이는 소거 동작의 단위인 블록(block)들의 집합이고, 블록은 읽기와 쓰기 동작의 단위인 페이지(page)들의 집합이다. 페이지는 사용자 데이터를 저장하는 데이터 영역(data area)과 오류 정정 부호(ECC, Error Correction Code)[4]와 같은 메타 정보를 저장하는 영역인 스페어 영역(spare area)으로 구분된다. 제어 유닛을 구성하는 요소로는 명령과 주소, 상태 등의 제어 레지스터들과 데이터 입출력 버퍼인 데이터 레지스터(data register) 등이 있다.

플래시 메모리의 데이터 읽기, 쓰기, 소거 동작은 각각을 위한 전용 오퍼레이션(operation)을 구비하고 있다. 각 오퍼레이션은 유사한 준비 과정을 갖는데, 이 과정 동안 디바이스에 오퍼레이션의 식별 명령, 타깃 페이지/블록의 주소, 확인 명령을 순차적으로 입력한다. 확인 명령을 받으면 플래시 메모리가 내부 동작을 시작한다. 내부 동작은 오퍼레이션 별로 차이가 있는데, 읽

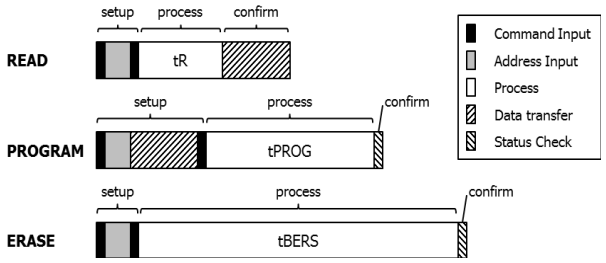


그림 1. 플래시 메모리의 읽기, 쓰기, 소거 오퍼레이션
Fig. 1. NAND flash memory operations: Read, Program and Erase.

기와 쓰기 오퍼레이션은 데이터 레지스터와 페이지 사이에서 데이터가 전송되고, 소거 오퍼레이션은 블록에 저장된 데이터가 소거된다. 쓰기와 소거 오퍼레이션은 내부 동작이 끝난 후 상태 레지스터를 읽어 들여 오퍼레이션이 정상적으로 수행되었는지 확인하는 과정이 필요하다.

이 논문에서는 그림 1과 같이 읽기, 쓰기, 소거 오퍼레이션을 준비(setup), 동작(process), 확인(confirm)의 세 단계로 구분한다. 준비단계는 식별 명령부터 확인 명령을 입력하는 단계이고, 동작단계는 플래시 메모리 내부의 데이터 전송이나 소거 동작이다. 쓰기 오퍼레이션의 준비단계는 플래시 메모리에 저장할 데이터를 데이터 레지스터에 입력하는 데이터 입력이 포함된다. 확인단계는 읽기 오퍼레이션의 데이터 출력과 쓰기, 소거 오퍼레이션의 상태 레지스터를 읽는 단계이다. 준비와 확인단계는 플래시 메모리 버스를 사용하는 동작인 반면, 동작단계는 버스를 사용하지 않는 플래시 메모리의 내부 동작이다.

2. 저장장치의 구조와 병렬 기법

이 논문에서 저장장치의 플래시 메모리 디바이스들이 공유하는 데이터 버스를 채널(channel), 채널을 공유하는 디바이스를 웨이(way)라고 하고, 다수의 채널/웨이로 이루어진 저장장치의 플래시 메모리 구조를 멀티-채널/웨이 구조라고 한다. 또한, 여러 채널이 동시에 다수의 오퍼레이션을 수행하는 병렬기법을 채널 인터리빙이라고 하고, 채널을 공유하는 웨이들이 채널의 유휴시간에 채널을 사용하여 오퍼레이션을 끼워 넣음으로서 채널 내부에서 다수의 오퍼레이션을 중첩하여 수행하는 병렬기법을 웨이 인터리빙이라고 한다[1~2, 4].

그림 2에서 보인 것과 같이 웨이 인터리빙 기법을 사용할 때 오퍼레이션을 스케줄링 하는 방법에 따라 채널의 사용효율이 크게 달라진다. 이 그림에서 *CH*는 채널,

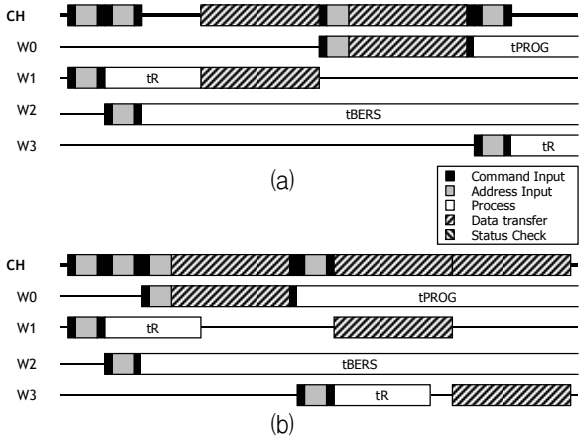


그림 2. 웨이 인터리빙:
(a) 일반적인 컨트롤러, (b) 적응적 컨트롤러
Fig. 2. Way interleaving:
(a) Typical controller, (b) Adaptive controller.

*Wx*는 동일 채널에 속한 웨이를 의미한다. 일반적인 컨트롤러는 채널의 오퍼레이션들을 효과적으로 스케줄링 하지 못하여 채널 사용효율이 낮은 모습을 보인다. 그러나 적응적 컨트롤러는 오퍼레이션을 효과적으로 스케줄링 하여 일반적인 컨트롤러에 비하여 높은 채널 사용효율을 보인다. 이 논문에서는 그림 2 (b)와 같은 수준의 웨이 인터리빙을 제공하는 적응적 컨트롤러를 베이스라인(baseline) 컨트롤러라고 한다.

3. 버퍼관리기법

플래시 메모리 기반 저장장치는 데이터의 시간적/공간적 지역성을 이용하여 저장장치의 성능을 높이기 위하여 호스트 인터페이스와 컨트롤러 사이에 버퍼를 구성한다. 버퍼의 공간은 한정되어 있기 때문에 공간이 부족하게 되면 빅팀(victim) 데이터를 선정하여 버퍼에서 퇴거시켜 새로운 데이터를 저장할 공간을 확보해야 한다. 이때, 빅팀을 선정하는 정책에 따라 저장장치의 성능이 달라질 수 있기 때문에 저장장치의 특성에 맞는 버퍼 관리기법이 요구된다.

플래시 저장장치를 위한 대표적인 버퍼 관리 기법으로, 버퍼의 데이터 엔트리를 시간 순으로 정렬한 후 최근 사용빈도가 낮은 데이터를 빅팀으로 선정하는 LRU (Least Recently Used) 기법을 개량한, CFLRU (Clean-First LRU)[3]가 있다. 이 기법은 LRU 데이터 엔트리에서 빅팀을 선정할 때, 버퍼에서 업데이트 되지 않은 클린(clean)을 우선하여 선정하는 기법이다. 이는 클린은 버퍼에서 퇴거되어도 플래시 메모리에 아무런 동작을 유발하지 않는 반면, 버퍼에서 업데이트 된 더

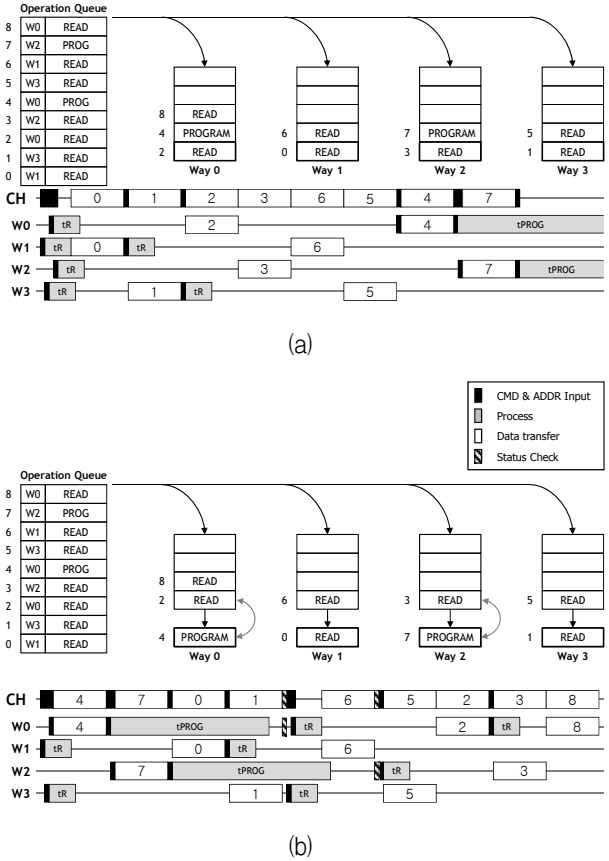
티(dirty)가 빅팀으로 선정되면 플래시 메모리에 저장하여야 하기 때문이다. 더욱이 쓰기 오퍼레이션은 오랜 시간이 걸리는 작업일 뿐만 아니라 플래시 메모리의 수명에도 영향을 미치기 때문에 쓰기 동작을 줄이는 것이 유리하다.

Ⅲ. 스케줄링 기법과 컨트롤러

이 장에서는 본 논문에서 제안하는 낸드 플래시메모리 오퍼레이션의 동적인 스케줄링 기법들과 제안된 기법을 적용한 컨트롤러에 대하여 설명한다.

1. 쓰기 오퍼레이션 선행

쓰기 오퍼레이션 선행(PP, Program-Precedence) 기법은 쓰기 오퍼레이션을 읽기 오퍼레이션보다 먼저 수행하도록 스케줄링하는 기법이다. 쓰기 오퍼레이션은 읽기 오퍼레이션에 비하여 오랜 시간이 걸리는 작업이기 때문에 읽기 오퍼레이션을 우선하도록 스케줄링하거나 순서를 변경하지 않는 것이 일반적이다. 그러나 쓰기 오퍼레이션은 읽기 오퍼레이션과 서로 상이한 채널 사용 특성이 있기 때문에 쓰기 오퍼레이션을 선행하여 읽기 오퍼레이션과 병행하여 수행되도록 스케줄링하면 성능 측면에서 이득을 얻을 수 있다.



(a)

(b)

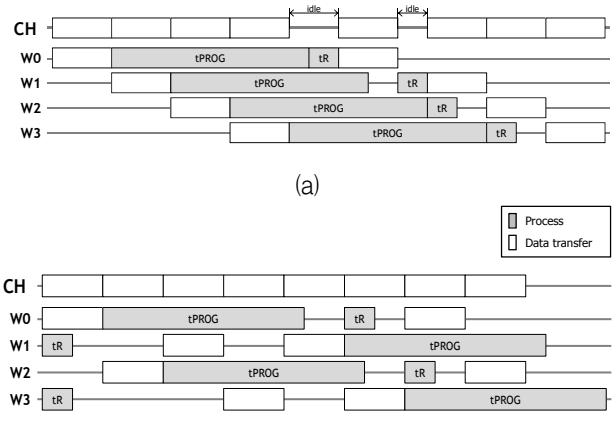
그림 3. 플래시 메모리 오퍼레이션의 타이밍 도:
(a) 베이스라인, (b) 쓰기 오퍼레이션 선행

Fig. 3. Timing diagrams for NAND flash memory operations: (a) Baseline, (b) Program-Precedence

PP를 통하여 얻을 수 있는 또 다른 이득은 버퍼 공간의 낭비를 줄일 수 있다는 것이다. 쓰기 오퍼레이션은 클린이 버퍼에서 퇴거되었음에도 불구하고 충분한 공간이 확보되지 않아 더티가 퇴거될 때 발생하는데, 이 오퍼레이션이 종료되기 전에는 더티가 점유하고 있는 버퍼 영역을 비울 수 없다. 만약 쓰기 오퍼레이션에서 에러가 발생하여 FTL이 새로운 페이지를 할당하여 다시 쓰기 오퍼레이션을 수행해야 한다면 지연으로 인하여 버퍼 낭비가 더욱 심각해질 것이다.

오퍼레이션 큐에 입력된 연속된 오퍼레이션들을 스케줄링 하는 예를 그림 3에 보인다. 베이스라인에서 쓰기 오퍼레이션은 읽기 오퍼레이션들이 종료된 후 실행되기 때문에 지연시간이 길고 다른 오퍼레이션과 중첩되지 않는다. 또한, W0와 W2에서는 이전 읽기 오퍼레이션이 끝난 후 아무런 동작을 하지 못하고 채널을 사용권한을 기다린다.

같은 조건에서 PP를 사용한 경우를 그림 3(b)에 보인다. PP는 쓰기 오퍼레이션의 순서를 앞당김으로써 오퍼레이션의 응답시간을 효과적으로 감소시킨다. 또한, 쓰기 오퍼레이션의 진행단계에 다른 웨이의 읽기 오퍼레이션을 효과적으로 중첩함으로써 전체적인 오퍼레이



(a)

(b)

그림 4. 쓰기 오퍼레이션 선행의 타이밍 도:
(a) 단순한 쓰기 오퍼레이션 선행,
(b) 채널 균형을 고려한 쓰기 오퍼레이션 선행

Fig. 4. Timing diagrams for Program-Precedence:
(a) Simple Program-Precedence,
(b) Channel balanced Program-Precedence.

션 밀집도가 높다. 그러나 PP를 사용함에 따라 일부 읽기 오퍼레이션에 대한 응답시간에 변화가 발생할 수 있다. 그림 3에서 보이듯이 PP를 사용하였을 때 오퍼레이션 별로 차이는 있으나 대다수의 읽기 오퍼레이션이 지연된다. 그러나 모든 연산이 지연되는 것은 아니다. 이는 PP가 채널 안의 연산 밀집도를 높이는 효과가 있기 때문이다. 연산 밀집도는 연산의 종류에 따라 달라진다.

연산의 밀집도를 높이려면, PP를 사용할 때 채널 내부의 모든 웨이가 같은 오퍼레이션을 수행하는 상황을 가급적 피하도록 스케줄링 하는 것이 성능 측면에서 유리하다. 왜냐하면 모든 웨이가 같은 오퍼레이션을 수행하는 경우 데이터 전송으로 인하여 채널이 유휴상태가 되거나 포화상태가 되어 오퍼레이션 밀집도가 낮아지기 때문이다.

그림 4는 단순화하여 표현한 쓰기 오퍼레이션 선행의 타이밍도이다. PP에서 쓰기 준비단계가 끝난 후 채널이 유휴상태가 되었음에도 불구하고 쓰기 진행단계가 진행 중이기 때문에 읽기 오퍼레이션이 시작되지 못한다. 읽기와 쓰기 오퍼레이션의 비율을 고려하여 스케줄링 한 예를 (b)에서 보인다. 이렇게 함으로써 PP에 비하여 채널의 오퍼레이션 밀집도를 높이고 그로 인하여 전체 오퍼레이션의 실행시간이 감소되었다. 이와 같은 쓰기 오퍼레이션의 선행 기법을 채널 균형 PP (BPP, channel Balanced PP)이라고 한다.

앞서 설명한 것과 같이 PP, BPP는 쓰기 오퍼레이션의 지연을 줄일 수 있으나, 읽기 오퍼레이션을 지연시키는 문제를 갖고 있다. 더욱이 쓰기 오퍼레이션은 읽기 오퍼레이션에 비하여 시간적 비용이 높은 작업이기 때문에 읽기 오퍼레이션이 여러 번 순서를 양보할 경우 읽기 오퍼레이션의 응답성이 지나치게 저하되는 문제가 있다. 이 문제를 최소화하기 위하여 한번 순서를 양보한 읽기 오퍼레이션은 다른 오퍼레이션에게 순서를 양보하지 않도록 한다.

PP와 BPP 기법의 대상은 같은 웨이의 오퍼레이션들로 한정되고, 가비지 컬렉션을 위한 데이터 병합 동작과 같이 실행 순서에 대하여 종속성을 갖는 일련의 오퍼레이션들은 순서를 바꾸지 않는다[5].

2. 데이터 전송 일시중지-재개

데이터 전송 일시중지-재개(DPR, Data transfer Pause-Resume)는 오퍼레이션을 수행 중인 웨이가 데이터 전송을 하는 도중에 다른 웨이가 (데이터 전송을 제외한) 채널권한을 요청하면 데이터 전송을 일시적으로 중지하였다가 해당 요청을 처리한 후 재개하는 기법이다. 디바이스와 컨트롤러 사이의 데이터 전송은 채널을 사용하는 다른 작업에 비하여 오랜 시간이 걸리는 작업이기 때문에 이 작업이 채널에 미치는 영향에 대하여 각별한 주의가 요구된다. 데이터를 전송하는 동안 다른 웨이의 짧은 채널 사용 요청에 채널이 응답하지 못하여 오퍼레이션 지연이 일어날 수 있기 때문이다.

그림 5는 데이터 전송으로 인하여 오퍼레이션이 지연되는 예이다. 베이스라인의 경우, W0에서 쓰기 진행단계가 거의 끝날 무렵 시작된 W1의 데이터 전송이 채널을 점유하였기 때문에 쓰기 확인단계는 매우 짧은 시간이 걸리는 작업임에도 불구하고 데이터 전송이 끝날 때까지 기다려야 한다. 다른 예에서, 베이스라인은 W1에서 쓰기 준비단계가 채널을 사용하고 있어 읽기 오퍼레이션의 준비단계가 끼어들지 못하여 지연이 발생한다.

반면 DPR에서, W0이 확인단계를 수행하기 위하여 채널 사용권한을 요청하면 W1은 데이터 전송을 일시중지 한다. 그다음 W0이 확인단계를 수행한 후 W1에서 다시 데이터 전송을 재개한다. 그림 5에서 두 컨트롤러의 채널 사용효율은 비슷하지만, DPR에서 쓰기 오퍼레이션의 응답시간이 베이스라인과 비교하여 단축되었다. DPR에서, W1에서 데이터 전송이 시작되기 전 W0이 채널을 잠시 빌려 사용하여 읽기 오퍼레이션의 준비단계를 수행한다. 그런 다음 W1이 데이터 전송을 재개함으로써 데이터 전송과 읽기 진행단계가 중첩되어 베이스라인과 달리 오퍼레이션 지연이 나타나지 않는다.

데이터 전송으로 인한 오퍼레이션 지연 문제는 데이터 전송 시간이 길수록 더욱 심각하게 나타날 것이다. 동기식(synchronous) 혹은 토글(toggle)과 같은 고속 인터페이스를 가진 플래시 메모리 디바이스는 데이터 전

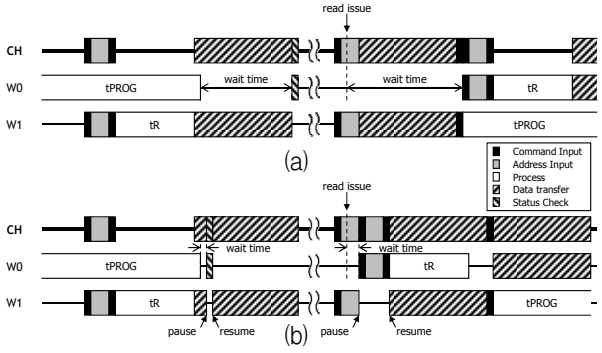


그림 5. 데이터 전송 타이밍 도: (a) 베이스라인, (b) 데이터 전송 일시중지-재개

Fig. 5. Timing diagrams for data transfer: (a) Baseline, (b) Data Transfer Pause-Resume.

송 시간이 짧기 때문에, 데이터 전송으로 인한 지연이 발생할 가능성이 낮고 그로 인한 오버헤드도 낮다. 그러나 단일 페이지의 크기가 점점 커지는 추세에 있고, 한 번의 오퍼레이션으로 두 페이지 데이터 전송이 이루어지는 멀티 플레인 오퍼레이션 등을 사용하는 경우 데이터 전송이 채널 미치는 영향은 무시할 수 없다.

3. 채널 사용권한 연장

오퍼레이션의 준비, 확인단계는 채널을 이용하는 동작이기 때문에, 각 단계의 시작하기 전에 컨트롤러로부터 채널의 사용권한을 부여받은 후 사용이 끝나면 반납하는 과정이 필요하다. 만약, 오퍼레이션의 확인단계가 종료될 때 다음에 수행할 오퍼레이션이 읽기나 소거 오퍼레이션이라면, 채널 사용권한 연장(CAE, Channel Authority Extension)하여 다음 오퍼레이션의 준비단계를 바로 시작하는 것이 성능 측면에서 유리하다. 왜냐하면 이들 오퍼레이션의 준비단계는 채널 사용시간이 매우 짧고, 연이은 단계인 진행단계를 시작하기 위한 작업인데, 만약 선행과 후행 오퍼레이션 사이에 채널 사용권한을 반납하고 다시 승인 받는 사이에 다른 웨이가 채널을 선점한다면 후행 오퍼레이션이 지연될 수 있기 때문이다.

그림 6은 한 웨이의 연속된 오퍼레이션 사이에 다른 웨이의 오퍼레이션이 개입하는 경우에 대한 예이다. 베이스라인에서 W0의 선행 읽기 오퍼레이션이 종료된 후 컨트롤러가 후행 읽기 오퍼레이션을 해석하는 동안 W1이 채널 사용권한을 부여받고 쓰기 오퍼레이션을 시작한다. 쓰기 준비단계가 끝날 때까지 다른 웨이가 채널을 사용할 수 없으므로 W0의 두 번째 읽기 오퍼레이션이 시작되지 못한다.

반면 CAE에서, 선행 읽기 오퍼레이션이 종료된 후 채널을 반납하지 않고 후행 읽기 오퍼레이션의 준비단계를 이어서 실행한다. 이렇게 하면 컨트롤러가 채널 사용권한을 반납하고 다시 부여받는 과정 없이 후행 오퍼레이션을 신속하게 시작할 수 있고, 두 오퍼레이션 사이에 다른 오퍼레이션의 개입을 완전히 차단할 수 있다.

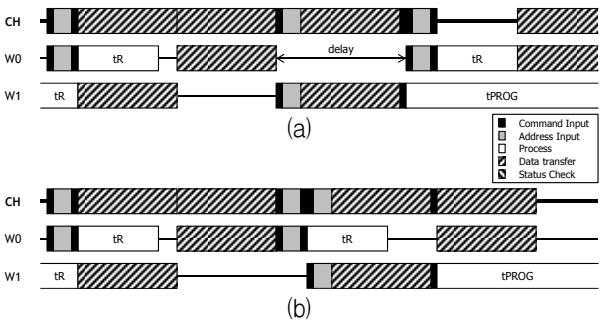


그림 6. 연속 오퍼레이션의 타이밍 도:
(a) 베이스라인, (b) 채널 사용권한 연장
Fig. 6. Timing diagrams for sequential operations:
(a) Baseline, (b) Channel authority extension.

4. 제안된 기법을 적용한 컨트롤러

이 논문에서 제안한 기법들을 적용한 플래시 메모리 기반 저장장치 컨트롤러의 구조를 그림 7에 보인다. 컨트롤러는 호스트 인터페이스, FTL 그리고 채널 컨트롤러로 구성된다. FTL은 호스트로부터 전달된 읽기 혹은 쓰기 요청을 플래시 메모리 오퍼레이션으로 변환하여 제어 레지스터(control register)를 통해 오퍼레이션 큐에 전달한다. 오퍼레이션은 플래시 메모리 디바이스를 제어하는 로우 레벨 컨트롤러 (low level controller)인 웨이 컨트롤러(way controller)에 의하여 플래시 메모리에 입력된다.

플래시 메모리 디바이스와 DRAM 버퍼 사이에 데이터 전송과 에러 검출 및 정정을 위하여 컨트롤러는 페이지 버퍼(page buffer)를 사용한다. 데이터를 전송할 때 페이지 버퍼의 크기만큼 데이터를 분할하여 전송하고, 전송 효율을 높이기 위하여 두 개 버퍼를 사용하여 교차 전송하는 더블 버퍼링(double buffering) 기법을 사용한다.

오퍼레이션 큐(operation queue)는 그림 8과 같이 오퍼레이션 분배장치(distributor)와 각 웨이를 위한 전용 오퍼레이션 큐들로 구성된다. 오퍼레이션 분배장치는 호스트로부터 전달된 오퍼레이션을 웨이 큐에 분배하고, 오퍼레이션 리오더(reorder)가 입력된 오퍼레이션 중에 다음에 실행할 오퍼레이션을 선정하여 오퍼레이션

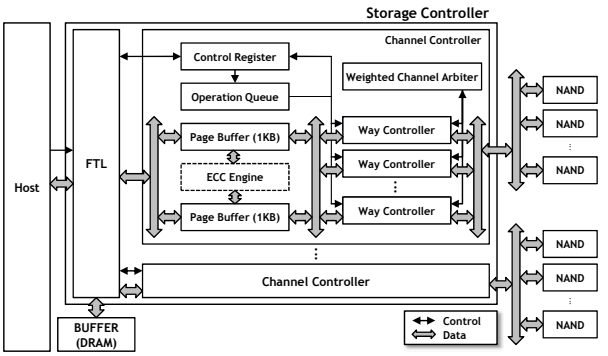


그림 7. 플래시메모리 저장장치 컨트롤러의 구조
Fig. 7. Architecture of NAND flash memory storage controller.

이슈 (issue) 레지스터에 저장한다. 그런 다음, 웨이 컨트롤러가 유휴상태가 되면 오퍼레이션 이슈 레지스터를 읽어 들여 플래시 메모리 디바이스를 구동한다.

CAE의 사용 여부는 채널 중재기의 개입 없이 전적으로 웨이 컨트롤러에 의하여 결정된다. 오퍼레이션 큐는 다음 오퍼레이션의 종류(next operation type)를 웨이 컨트롤러에게 전달하고, 웨이 컨트롤러가 이를 이용하여 CAE의 사용여부를 결정한다.

PP, BPP는 오퍼레이션 리오더에 의하여 실행되는데, 선행 가능한 읽기 오퍼레이션의 수는 큐의 깊이(depth), 큐에 있는 오퍼레이션 수와 종류 그리고 오퍼레이션 간의 데이터 의존성에 의하여 동적으로 변한다. 데이터 의존성은 오퍼레이션의 식별을 위한 ID, 페이지 혹은 블록과 주소, 데이터 레지스터와 DRAM 버퍼 사이에 데이터 전송을 위한 버퍼의 주소를 이용하여 어렵지 않게 판별할 수 있다.

BPP를 위하여 오퍼레이션 분배장치는 각 웨이의 리오더로부터 전달받은 다음 오퍼레이션 타입을 이용하여 채널의 쓰기 오퍼레이션 비율(prog ratio)을 각 웨이의 리오더에게 알리고, 리오더는 이를 이용하여 다음에 실행할 오퍼레이션을 확정한다. 어떤 웨이가 먼저 쓰기 오퍼레이션을 수행할지는 동적인 채널 상황에 따라 달라지는데 가장 먼저 유휴상태가 된 웨이가 우선권을 가진다. 만약 동시에 여러 웨이가 유휴 상태가 된다면 짝수 웨이가 우선권을 갖는다.

가중치 기반 채널 중재기는 웨이 컨트롤러의 요청(request)에 따라 채널 사용권한을 승인(grant)하는 장치로, 어떤 요청에 권한을 인가할 것인지 여부는 사전에 부여한 가중치(weight)에 따른다. 가중치는 순위가 가장 높은 읽기 및 소거 오퍼레이션의 준비단계에서 부

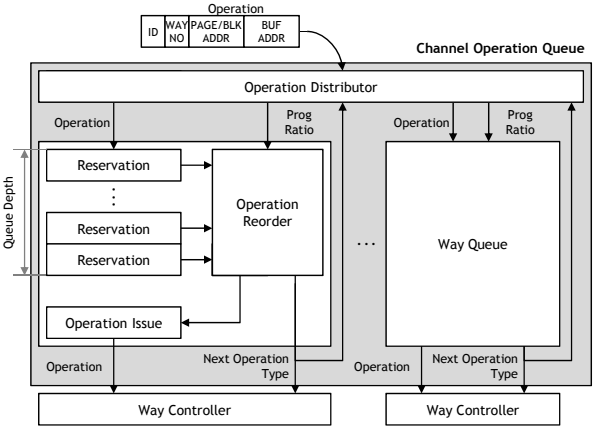


그림 8. 오퍼레이션 큐
Fig. 8. Operation queue.

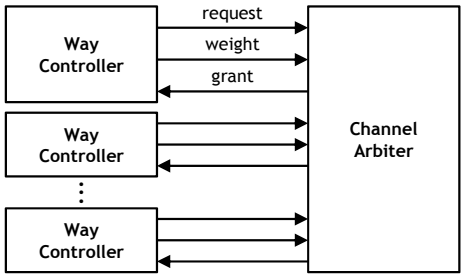


그림 9. 가중치 기반 채널 중재기
Fig. 9. Weighted Channel Arbiter.

터 쓰기 및 소거 오퍼레이션의 확인단계, 쓰기 오퍼레이션의 준비, 읽기 오퍼레이션의 확인단계 순으로 부여된다. 모든 오퍼레이션의 동작 단계는 채널을 사용하지 않으므로 가중치를 부여하지 않는다. 만약 동일한 가중치의 요청이 동시에 들어오면 순환 순서(round robin) 정책으로 권한을 부여한다.

DPR은 웨이 컨트롤러와 채널 중재기에 의하여 이루어지는데 제어 복잡도를 낮추기 위해 일시중지와 재개는 데이터 분할 전송 사이에 수행한다. (앞서 설명한 것과 같이 컨트롤러와 플래시 메모리 사이에서 데이터는 페이지 버퍼 크기 단위로 분할되어 전송된다.) DPR을 사용할 때 일시중지된 데이터 전송이 다른 웨이로부터 요청된 새로운 데이터 전송에게 채널 권한을 빼앗기는 것을 방지하기 위하여 별도의 가중치를 설정한다. 채널 중재기는 별도로 설정된 가중치를 이용하여 데이터 전송을 재개하여야 하는 웨이 컨트롤러를 판별할 수 있다.

Ⅳ. 실 험

1. 실험환경

제안된 기법을 적용한 컨트롤러의 프로토타입을 구현하여 FPGA 플랫폼 보드에서 실험하였다. 실험에 사용된 플랫폼은 비휘발성 메모리에 관한 연구를 위하여 자체 제작한 개발 보드로, 빌트인 프로세서인 PPC 440 2개와 PCI Express MAC/PHY 등이 내장된 FPGA를 비롯하여 SRAM, DRAM 비휘발성 메모리와 NAND 플래시 메모리 등의 컴포넌트 등을 구비하고 있다. 상세한 사양과 구성을 표 1과 그림 10에 각각 보인다.

플래시 메모리 디바이스는 두 개의 다이로 구성되어 있으며, 페이지의 데이터 영역 크기는 8KB, 스페어 영역의 크기는 448바이트이다. 플래시 메모리 디바이스들은 16채널/4웨이로 구성되고 채널 폭은 8비트이다. 실

표 1. 테스트 플랫폼 보드의 사양
Table 1. Specification of test platform board.

구분	품명	사양
FPGA	Virtex 5 (XI5VAX 130T IF1738-1)	1ea
MCU	PC440® (Frequency: 400MHz, I-$: 32KB, D-$: 32KB, FPU, Timer)	2ea
SRAM	K6R4016V1D-UC10 (512KB×2EU)	1MB
DDR	H5MS1G22mP (128MB×32EU)	512MB
NAND	H27UDP8V5AT (16GB×32EU)	512GB
Host I/F	PCI Express 1.1 (4 lane)	1GB/s

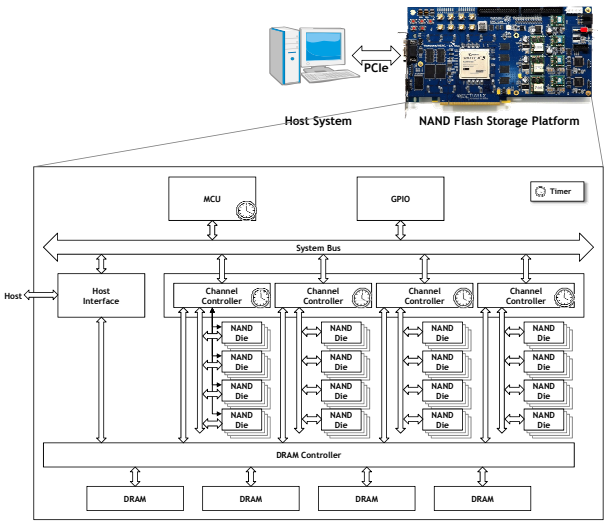


그림 10. 실험 셋업
Fig. 10. Experimental Setup.

험을 위하여 16채널을 4개 채널 단위 클러스터로 구성하여 논리적으로 한 채널 폭이 32비트인 4채널/4웨이 구조를 갖도록 하였다. 클러스터 논리 페이지의 데이터 영역은 32KB, 스페어 영역은 1792B 이다.

윈도우 7 기반의 호스트와 플랫폼 보드는 PCIe 1.1를 통하여 연결하고, 플래시 메모리 영역은 파일시스템은 NTFS (New Technology File System)를 사용하였다. FPGA에 내장된 프로세서인 PPC 440 중 하나를 MCU로 이용하고, 본 논문의 기법이 적용된 플래시 저장장치 컨트롤러와 PCIe 호스트인터페이스, 저장장치의 성능 및 플래시 메모리의 동작을 모니터링 하는 타이머 모듈 등이 포함되어 있다. 각 웨이 컨트롤러의 플래시 메모리 오퍼레이션 큐 깊이는 32개이다. DRAM 가운데 매핑 테이블(mapping table)을 위하여 256MB, 버퍼 영역으로 128MB를 사용한다. 주소변환 기법은 페이지 매핑(page mapping)을 사용하였으며, 버퍼 관리 기법은 CFLRU를 이용하였다. 저장장치용 벤치마크 프로그램인 Iometer[6]를 사용하여 저장장치와 컨트롤러의 성능을 평가하였다.

2. 실험결과

표 2는 저장장치 컨트롤러의 FPGA 자원소모량을 나타낸 것이다. BASELINE은 가중치 기반 채널 중재기를 포함한 컨트롤러이고, PROPOSED는 제안된 기법이 적용된 가중치 기반 채널 중재기와 오퍼레이션 큐가 포함된 컨트롤러이다. PROPOSED는 BASELINE에 비하여 항목에 따라 2-5% 정도 FPGA 자원을 더 사용한다.

PP가 버퍼에 미치는 영향을 알기 위하여 BASELINE과 PP 기법을 적용한 컨트롤러에서 워크로드를 실행한 후 각각에서 발생한 쓰기 오퍼레이션 수를 비교한 결과를 그림 11에 나타낸다. 실험에 사용한 워크로드의 읽기와 쓰기 비율은 5:5, 순차(sequential)와 임의(random) 비율은 2:8이다. 이는 버퍼에 클린이나 더티만 존재하거나 어느 한쪽으로 치우치는 상황을 회피하고, 버퍼의 데이터 엔트리가 다양한 크기를 갖도록 하기 위함이다. 또한 순차와 임의 비율은 다르지만 개별 요청에 대한 데이터 크기가 다르기 때문에 데이터량 측면에서 보면 비율은 5:5 정도이다.

그림 11에서 편의상 BASELINE을 기준으로 실행 중 10만회 쓰기 오퍼레이션이 발생할 때까지 결과를 보였다. Runtime은 워크로드 실행 중 발생한 쓰기 오퍼레이션이고, Flush는 워크로드를 실행한 후 버퍼의 데이터 엔트리 중 더티의 수이다.

표 2. FPGA 자원 사용량
Table 2. FPGA resource utilization.

Resource	BASELINE	PROPOSED
Slice Registers	26,837(32%)	28,476(34%)
Slice LUTs	31,275(38%)	35,148(42%)
Occupied Slice	15,146(73%)	16,048(78%)
BlockRAM	298(40%)	298(40%)
Bounded IOBs	716(85%)	716(85%)

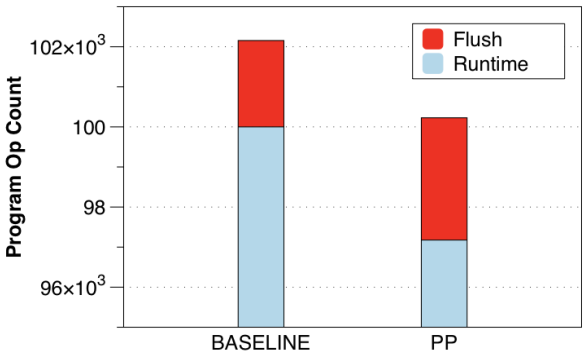


그림 11. 쓰기 오퍼레이션 선행기법이 버퍼에 미치는 효과
Fig. 11. Effects of PP on buffer.

PP가 BASELINE에 비하여 워크로드 실행 중에 2,824개 적은 쓰기 오퍼레이션이 발생되고, 실행 후에 버퍼에 남아있는 더티 데이터는 PP가 895개 많은 것으로 나타났다. 따라서 PP가 BASELINE 보다 총 1,929개의 쓰기 오퍼레이션이 적게 발생하였다. 이는 PP를 사용하였을 때 버퍼의 공간 확보가 빨라진 만큼 퇴거 횟수가 감소하여, 퇴거 되지 않은 데이터들에 대한 반복된 버퍼 적중이 일어났기 때문이다.

순차 요청은 매우 단조로운 채널 접근 특성을 가진 일련의 순차 오퍼레이션으로 변환되기 때문에 이를 위한 오퍼레이션 스케줄링 복잡도가 낮다. 이와 반대로 임의 요청은 복잡한 접근 특성을 보이지만 작은 단위의 요청이 반복되어 스케줄링 복잡도는 높지만 채널의 대역폭을 충분히 활용할 수 없다. 이러한 이유로 이후의 실험에서 호스트 요청의 순차와 임의 비율을 5:5로 고정하였다.

PP와 BPP로 인한 읽기 오퍼레이션의 지연이 성능에 미치는 영향을 평준화한 읽기 오퍼레이션 처리량(throughput)으로 그림 12에 나타내었다. PP는 쓰기 오퍼레이션의 비중이 크고 큐 깊이가 깊을수록 읽기 오퍼레이션의 처리량이 눈에 띄게 낮아지지만, BPP의 경우는 처리량 감소폭이 비교적 낮다. 이는 BPP가 PP에 비하여 읽기 오퍼레이션 지연이 비교적 적기 때문이다.

쓰기 오퍼레이션이 읽기 오퍼레이션에 비하여 수배의 시간이 걸리는 작업임을 고려하면 PP와 BPP로 인한 읽기 오퍼레이션의 처리량 감소량은 그다지 크지 않음을 알 수 있다. 이는 읽기 오퍼레이션이 항상 큐 깊이만큼 순서를 양보하는 것이 아니라 실행 중에 동적으로 양보할 순서가 결정되기 때문이다. 또 다른 이유는 그림 3에서 보인 것과 같이 모든 읽기 오퍼레이션이 지연되는 것이 아니고, 지연이 되더라도 그 정도가 각각 다르기 때문이다. 이 실험에서 읽기와 쓰기 비율이 2:8 일 때, 다른 경우와 비교하여 BPP의 처리량이 낮게 나타난 것은 읽기 오퍼레이션의 수가 적기 때문에 개별 오퍼레이션의 지연이 전체 성능에 비교적 크게 반영되었기 때문이다.

제안한 기법들의 성능을 채널 사용효율로 표현하여 그림 13에 나타낸다. 이 그림에서 BL은 BASELINE, ALL은 BPP, CAE, DPR 기법이 모두 적용된 컨트롤러이다. 본래 채널 사용효율은 데이터 영역의 채널 사용 비중만을 계상하여야 한다. 왜냐하면 오퍼레이션의 명령과 주소, 스페어 데이터 전송 등과 같은 동작은 저장장치의 성능과 관계없는 부가 동작이기 때문이다. 하지만 이 실험의 목적이 컨트롤러가 채널을 얼마나 효율적으로 사용하는가에 있기 때문에 채널을 사용하는 모든 동작을 대상으로 채널 사용효율을 측정하였다. 또한 읽기와 쓰기 요청의 비율에 대한 채널의 사용효율 변화를 알아보고자 호스트에서 읽기와 쓰기 요청의 비율을 변경하면서 채널 사용효율을 측정하였다.

전반적으로 제안된 기법을 사용하였을 때 채널 사용효율이 높아진 것을 볼 수 있다. 그러나 읽기와 쓰기 비율이 10:0과 0:10의 경우는 PP, BPP와 BL의 사용효율이 동일하게 나타난다. 이는 읽기나 쓰기만 있는 경우는 PP와 BPP를 사용할 수 없기 때문이다. CAE는 읽기 오퍼레이션의 비중이 높을 때, DPR은 쓰기 오퍼레이션의 비중이 높을 때 좋은 성능을 보이는 것을 관찰할 수 있다.

대부분의 경우에서 ALL이 BL에 비하여 높은 사용효율을 보인다. 그러나 읽기와 쓰기 비율이 0:10인 경우는 모든 컨트롤러의 채널 사용효율이 동일하다. 이 경우에서는 쓰기 오퍼레이션의 긴 진행단계로 인하여 그림

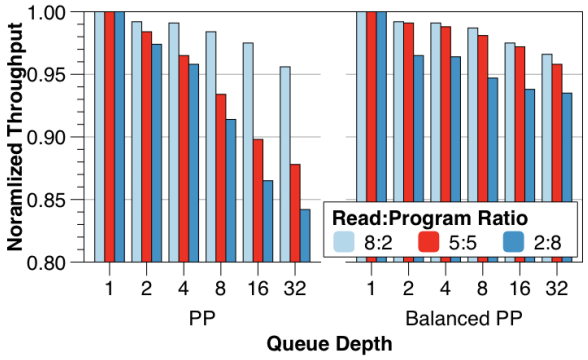


그림 12. 큐 깊이가 쓰기 오퍼레이션 선행기법에 미치는 효과

Fig. 12. Effects of queue depth on PP and Balanced PP.

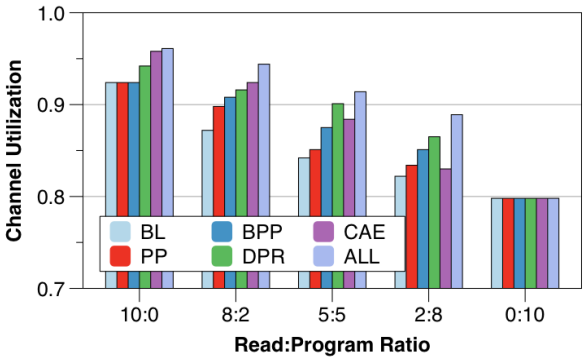


그림 13. 각 기법들의 채널 사용효율

Fig. 13. Channel Utilizations of each techniques.

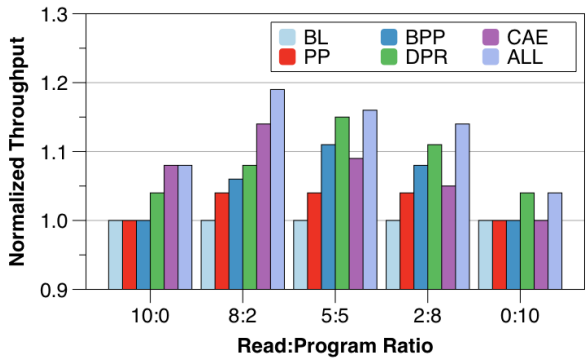


그림 14. 각 기법들의 평준화된 처리량
Fig. 14. Normalized throughputs of each techniques.

4(a)에서 보인 것과 유사한 채널의 유휴시간이 발생하였는데, 이러한 현상이 나타나면 오퍼레이션 스케줄링을 통하여 채널 사용효율을 높일 여지가 없기 때문이다. 이 경우를 제외하고, 나머지 모든 경우에서 ALL이 BL에 비하여 4-7% 높은 채널 사용효율을 보인다.

그림 14는 각 기법들의 성능을 BL을 기준으로 평준화한 처리량으로 나타낸다. 그림 13과 비교하여 보면 대부분의 경우 채널 사용효율과 처리량이 비례하는 경향을 보인다. 그러나 읽기와 쓰기 비율이 0:10의 경우 DPR과 ALL의 경우 채널 사용효율은 같으나 처리량이 다르다. 또한 나머지 다른 경우에서 채널 사용효율에 비하여 처리량이 더 큰 차이를 보인다. 이는 DPR 기법을 사용하여 쓰기 오퍼레이션의 확인단계를 보다 빨리 처리함으로써 오퍼레이션의 응답성이 향상되었기 때문이다. (그림 5에서 보인 것과 같이 채널의 사용효율과 처리량이 반드시 비례하는 것은 아니다.) 또한, 큐 깊이가 32임에도 불구하고 PP와 BPP를 사용하였을 때 처리량 향상이 된 것을 알 수 있다. 이러한 결과가 나타난 이유는 PP, BPP를 사용하여 얻은 향상된 쓰기 오퍼레이션의 처리량 이 읽기 오퍼레이션 지연으로 인한 처리량 손실보다 크기 때문이다. 제안된 기법을 모두 컨트롤러에 적용한 결과 읽기와 쓰기 오퍼레이션의 비율에 따라 4-19%의 처리량 향상이 있었음을 알 수 있다.

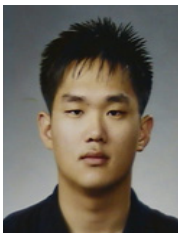
V. 결 론

본 논문에서는 낸드 플래시 메모리 저장장치 컨트롤러를 위한 동적인 오퍼레이션 스케줄링 기법들을 제안하였다. 제안된 기법들을 컨트롤러에 적용하여 FPGA 기반 플랫폼 보드에서 성능을 평가한 결과, 구현을 위한 추가 자원 소모량이 2-5% 불과하지만 정적인 오퍼레이션 스케줄링 기법을 사용하는 컨트롤러에 비하여 1.9% 쓰기 오퍼레이션이 감소되고, 채널의 사용효율이 4-7% 향상되었으며 저장장치의 처리량이 4-19% 향상되었음을 확인할 수 있었다.

REFERENCES

[1] Kang, Jeong-Uk, et al. “A multi-channel architecture for high-performance NAND flash-based storage system.” Journal of Systems Architecture 53.9 (2007): 644-658
[2] Nam, Eyee Hyun, et al. “Ozone (O3): An out-of-order flash memory controller architecture.” Computers, IEEE Transactions on 60.5 (2011): 653-666
[3] Park, Seon-yeong, et al. “CFLRU: a replacement algorithm for flash memory.” Proceedings of the 2006 international conference on Compilers, architecture and synthesis for embedded systems. ACM, 2006
[4] 공준진, 손홍락, and 설창규. “플래시 메모리 컨트롤러 (Flash Memory Controller) 기술.” 전자공학회지, 제39권 제9호 2012.9, page(s): 39-46
[5] 정상혁, 송용호. “낸드 플래시 메모리의 응답 지연 시간 최소화를 위한 선점방식 가비지 컬렉션 기법.” 정보과학회논문지: 시스템 및 이론 39.6 (2012): 398-404
[6] Sievert, Jerry. “Iometer: The I/O performance analysis tool for servers.” (2004).

저 자 소 개



정 재 형(학생회원)

2005년 배재대학교 정보통신공학 졸업(학사).
2007년 한양대학교 전자컴퓨터 통신공학과 졸업(석사).
2007년～현재 한양대학교 전자컴퓨터공학과 박사과정.

<주관심분야 : 컴퓨터구조, 임베디드 시스템, 비휘발성 메모리 등>

송 용 호(정회원)

1989년 서울대학교 컴퓨터공학과 졸업(학사)
1991년 서울대학교 컴퓨터공학과 졸업(석사).
2002년 University of Southern California Electrical Engineering 졸업 (박사).
1991년～1996년 삼성전자 전임연구원.
1996년～2002년 University of Southern California 연구조교.
2002년～2003년 (주)조선 IT 연구소장.
2003년～현재 한양대학교 융합전자공학부 교수.
IEEE International Parallel and Distributed Processing Symposium 프로그램 구성위원.

<주관심분야 : 시스템 구조, SoC, NoC, 멀티미디어, 비휘발성 메모리 등의 임베디드 시스템 구조 등>