논문 2011-48SD-7-7

SSD 스토리지 시스템을 위한 효율적인 DRAM 버퍼 액세스 스케줄링 기법

(Efficient DRAM Buffer Access Scheduling Techniques for SSD Storage System)

박 준 수*, 황 용 중*, 한 태 희**

(Jun Su Park, Yong Joong Hwang, and Tae Hee Han)

요 약

최근 NAND 플래시 메모리를 이용한 새로운 저장매체인 SSD(Solid State Disk)가 모바일 기기를 중심으로 HDD(Hard Disk Drive)를 대체하면서 가격대비 성능을 향상시키려는 연구가 다양한 접근 방식을 통해 진행 중이다. 병렬처리를 통한 NAND 플래시 대역폭 향상을 위해 채널수를 확장하면서 호스트(PC)와 NAND 플래시 간의 버퍼 캐시의 역할을 하는 DRAM 버퍼가 SSD 성능 개선의 bottleneck으로 작용하게 되었다. 이 문제를 해소하기 위해 본 논문에서는 DRAM Multi-bank를 활용한 스케줄링 기법을 통해 DRAM 버퍼 대역폭을 개선함으로써 저비용으로 SSD의 성능을 향상시키는 효과적인 방안을 제안한다. 호스트와 NAND 플래시 다중 채널이 동시에 DRAM 버퍼의 접근을 요청하는 경우, 이들의 목적지를 확인하여 DRAM 특성을 고려한 스케줄링 기법을 적용함으로써 bank 활성화 시간과 row latency에 대한 overhead를 감소시키고 결과적으로 DRAM 버퍼 대역폭 활용을 최적화할 수 있다. 제안한 기법을 적용하여 실험한 결과, 무시할만한 수준의 하드웨어 변경 및 증가만으로 기존의 SSD 시스템과 비교하여 SSD의 읽기 성능은 최대 47.4%, 쓰기 성능은 최대 47.7% 향상됨을 확인하였다.

Abstract

Recently, new storage device SSD(Solid State Disk) based on NAND flash memory is gradually replacing HDD(Hard Disk Drive) in mobile device and thus a variety of research efforts are going on to find the cost-effective ways of performance improvement. By increasing the NAND flash channels in order to enhance the bandwidth through parallel processing, DRAM buffer which acts as a buffer cache between host(PC) and NAND flash has become the bottleneck point. To resolve this problem, this paper proposes an efficient low-cost scheme to increase SSD performance by improving DRAM buffer bandwidth through scheduling techniques which utilize DRAM multi-banks. When both host and NAND flash multi-channels request access to DRAM buffer concurrently, the proposed technique checks their destination and then schedules appropriately considering properties of DRAMs. It can reduce overheads of bank active time and row latency significantly and thus optimizes DRAM buffer bandwidth utilization. The result reveals that the proposed technique improves the SSD performance by 47.4% in read and 47.7% in write operation respectively compared to conventional methods with negligible changes and increases in the hardware.

Keywords: SSD, NAND Flash, DRAM buffer, Scheduling, DRAM bank

Ⅰ. 서 론

NAND 플래시 메모리는 기존의 하드 디스크에 비해 가격이 높은 단점에도 불구하고 빠른 처리 속도, 저전

* 학생회원, ** 평생회원, 성균관대학교 정보통신공학부
(School of Information Communication Engineering, Sungkyunkwan University)
※ 본 연구보고서는 지식경제부 출연금으로 수행한 ETRI 시스템반도체진흥센터 위탁연구의 결과입니다.
접수일자: 2011년3월23일, 수정완료일: 2011년6월15일

력, 강한 내구성, 무소음, 경박단소와 같은 장점으로 인해 디지털 카메라, MP3 플레이어, 스마트 폰과 같은 많은 모바일 기기의 주 저장 매체로 쓰이고 있다. 최근 용량 대비 가격이 HDD(Hard Disk Drive) 수준에 근접하면서 NAND 플래시 기반의 새로운 대용량 저장매체인 SSD(Solid State Disk) 시장이 확대되어 일부 서버급 컴퓨터 등에도 채용되는 추세이다.

HDD의 경우, 플래터가 헤드를 통해서 데이터를 검색하는 기계적 특성으로 인해 많은 지연 시간이 발생할 뿐만 아니라 동 시간에 다수의 플래터 중 1개의 플래터만이 헤드를 통해 사용되므로 병렬처리가 불가능하다. 이러한 특성 때문에 대역폭 향상에 제한이 있다.[1]

반면에 SSD는 기계적인 장치 구동에 따른 지연 시간이 없으며 병렬처리가 가능하다. SSD에 사용되는 NAND 플래시 메모리 한 채널의 최대 대역폭은 133 MByte/s이며 일반적으로 8개의 NAND 플래시 채널을 사용하는데, 이 경우 대역폭은 1064 MByte/s로 HDD에 비해 7배 이상의 데이터 처리 속도를 가진다. 또한 채널의 수가 증가할수록 대역폭은 비례적으로 향상이 가능하다. 이러한 이유로 넷북, 노트북, 태블릿과 같은 모바일 기기에서부터 고성능 서버용 컴퓨터까지 SSD가 장착되면서 시장 영향력을 넓혀가고 있다.[2]

SSD는 제조사마다 다소 차이가 있지만, 일반적인 구조는 그림 1과 같다.[3] SRAM에 부트코드와 실행코드가 담겨 있고, 호스트 인터페이스를 통해 읽기 또는 쓰기에 대한 요청이 들어오면 프로세서는 DRAM과 일부 NAND 플래시에 있는 메타 데이터를 이용해서 데이터의 알맞은 위치를 파악하고 NAND 플래시 메모리에 데이터를 기록하거나 읽어온다. HDD와 마찬가지로 버퍼 캐시의 역할을 하는 DRAM 버퍼가 존재하고 캐시 hit

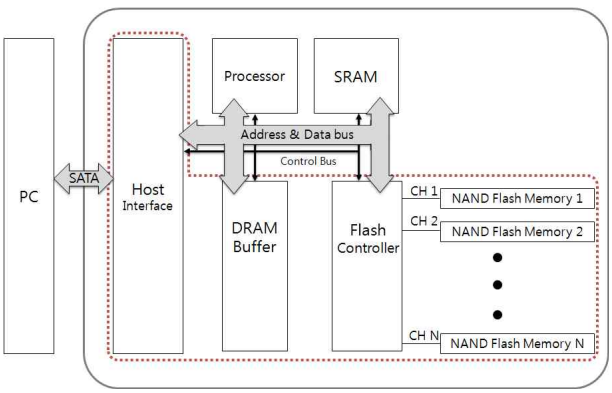


그림 1. 전형적인 SSD의 구조
Fig. 1. Typical Structure of SSD.

비율에 따라서 SSD의 성능에 상당한 영향을 미칠 수 있다.

HDD는 기계적인 특성으로 인한 지연과 병렬적인 처리를 못하는 제약 때문에 DRAM 버퍼의 대역폭 중 제한된 부분만을 사용한다. 반면에 SSD는 NAND 플래시의 대역폭을 200 MByte/s 혹은 400 MByte/s로 높이고, 병렬성을 향상시키기 위해 채널을 16개까지 확장하면, 대역폭이 3200 MByte/s에서 6400 MByte/s까지 증가하여 DRAM 버퍼의 대역폭이 부족해지는 상황을 발생시킬 수 있다.[4] 이 때문에 데이터 전송 시 병목 현상을 불러와 DRAM 버퍼가 전체 시스템의 성능을 저해하는 요인이 될 수 있다.[5]

이를 해소하기 위해 고속 DRAM 버퍼를 사용할 수 있지만 이러한 해결책은 그에 따른 비용이 상승한다는 단점이 있다. 그러므로 본 논문에서는 DRAM 버퍼의 대역폭을 효율적으로 사용하면서도 저비용으로 성능을 개선할 수 있는 DRAM 버퍼 스케줄링 기법에 초점을 맞추어 연구를 진행하였다.

본 논문에서는 SSD내의 DRAM 버퍼에 접근 하는 장치로부터 발생하는 읽기 혹은 쓰기 요청들을 DRAM의 Multi-bank 기능을 활용하여 효율적으로 스케줄링 하는 방안을 제안한다. 각 장치들이 요청을 하는 경우 접근하려는 DRAM 버퍼의 bank를 확인 후 bank 활성화 시간, RAS(Row Address Strobe)와 CAS(Column Address Strobe) 지연 시간에 대한 overhead를 줄일 수 있도록 DRAM 버퍼 액세스 요청 순서를 스케줄링 함으로써, DRAM 버퍼의 대역폭 사용을 최적화 할 수 있다.

본 논문은 다음과 같이 구성 되어 있다. Ⅱ장에서는 기존의 SSD의 성능 개선을 위한 연구 및 현재 사용되고 있는 스케줄링 기법에 대해 소개하고, Ⅲ장에서는 SSD 시스템의 구성도와 제안하는 스케줄링 기법에 대하여 설명한다. Ⅳ장에서는 이 제안한 방식을 기반으로 하는 시스템을 모델링한 뒤에, 시뮬레이션을 통한 결과와 분석에 대하여 설명한다. 마지막으로 Ⅴ장에서는 결론을 도출하였다.

Ⅱ. SSD 성능 개선 방법 및 DRAM 버퍼 스케줄링

1. 기존의 SSD 성능 개선 연구

SSD의 성능을 개선을 위한 연구는 다음과 같이 다양

한 접근 방식을 통해 진행되어왔다.

- 빠르고 내구성이 좋은 SLC(Single-Level Cell)와 저비용으로 대용량 저장 장치를 구성할 수 있는 MLC(Multi-Level Cell)의 장점을 혼합한 SSD 성능 개선.[6]

- 페이지 단위의 쓰기만 가능한 NAND 플래시는 크기가 작은 데이터를 갱신할 때 overhead가 발생하므로, 이를 줄이기 위해 바이트 단위의 쓰기가 가능한 FRAM(Ferroelectric Random Access Memory)과 같은 차세대 메모리를 일부 혼합하여 이용하는 방안[7]

- 소프트웨어적으로 매핑 정보의 적중률에 따라 매핑 정보의 크기를 동적으로 변경하여 기존의 매핑 기법에 비해 매핑 테이블의 크기를 감소시키고 성능 향상을 꾀하는 방법[8]

본 논문에서는 소프트웨어적인 개선 방법과 결합이 가능하며 기존의 연구에 비해 저비용으로 SSD의 성능을 개선하기 위해 DRAM 버퍼 대역폭을 효율적으로 활용할 수 있는 DRAM 버퍼 액세스 스케줄링 기법을 제안한다.

2. DRAM 버퍼 스케줄링 기법

SSD에 사용되는 DRAM 버퍼 스케줄링 기법으로는 Fixed-Priority 방식, Round-Robin 방식 그리고 TDM + Round-Robin 등이 있다.

가. Fixed-Priority

그림 2(a)에 나와 있는 Fixed Priority 방식은 각 마스터들에게 고정된 우선순위를 갖도록 하는 방식이다. 이 방식은 마스터의 중요도에 따라 우선순위를 높일 수 있는 장점이 있지만, 마스터의 개수가 많을 경우 심각한 starvation 현상을 발생 시킬 수 있다.

나. Round-Robin

그림 2(b)는 기존의 SSD에서 일반적으로 사용되는 Round-Robin 방식을 나타낸다. 이 방식은 Fixed Priority의 starvation 문제를 해결하기 위해 제안되었다. Round-Robin은 모든 마스터의 우선순위를 동일하게 유지하게 하여, 버스를 한 마스터씩 돌아가면서 사용함으로, 모든 마스터가 공평하게 버스를 사용할 수 있도록 해 주는 방식이다.

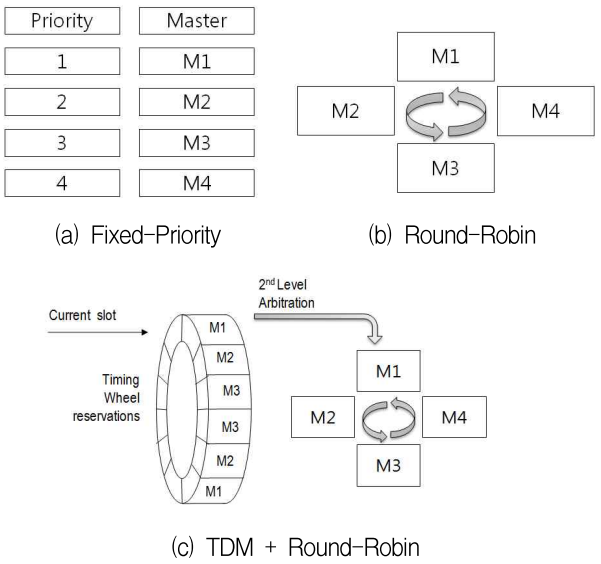


그림 2. 스케줄링 방식의 종류
Fig. 2. Types of Scheduling Methods.

이 방식은 starvation을 방지할 수는 있지만 중요한 마스터의 데이터 처리를 빠르게 수행할 수 없는 단점이 있다.

다. TDM + Round-Robin[9]

TDM(Time-Division Multiplexing) + Round-Robin 방식은 각 마스터에게 타이밍 휠에서 버스를 할당하는 슬롯의 수를 다르게 배분하고, 슬롯이 사용되지 않을 때에는 Round-Robin을 사용하는 방식으로 그림 2(c)에 나타내었다. 이것은 각 마스터의 starvation 현상을 방지하면서도 중요한 마스터에게 많은 슬롯 할당해 데이터 처리를 높이는 방식이지만 여전히 중요한 마스터의 대기시간이 길어질 수 있다. 그리고 현재 슬롯에 할당된 마스터의 요청이 없으면 Round-Robin으로 2단계 중재를 하면서 추가 지연 시간이 발생하는 단점이 있다.

위에서 설명한 기존의 스케줄링 기법의 경우에는 우선순위가 높은 마스터의 버스 점유율을 높여서 데이터의 처리를 빠르게 하거나 혹은 모든 마스터의 버스 점유율을 균등하게 배분하여 starvation을 없애는 역할을 할 뿐, 실질적으로 DRAM 버퍼의 대역폭을 효율적으로 사용하지 못하였다.

본 논문에서는 DRAM의 Multi-bank 기능을 활용하여 DRAM 버퍼의 대역폭을 효율적으로 활용할 수 있는 스케줄링 기법을 제안한다.

Ⅲ. 제안하는 SSD 성능 개선 방안

본 논문에서 제안하는 기법은 DRAM의 Multi-bank 기능을 활용해 bank 활성화, RAS 와 CAS 지연 시간을 줄이는 것이다. 제안하는 기법을 설명하기에 앞서 그림 3의 SDRAM의 상태도를 통해 SDRAM이 읽기/쓰기 상태에서 데이터 전송을 마친 후 다시 읽기나 쓰기 상태로 오기 위해 필요한 지연 시간을 설명한다. SDRAM은 읽기/쓰기 상태에서 데이터를 전송한 후에 bank내의 해당 row를 precharge하고 IDLE 상태로 가며 이 때 발생하는 지연 시간은 t_{RP}이다. 다음 읽기/쓰기 동작을 준비하기 위해 bank 활성화와 row 주소 활성화에 대한 지연 시간인 t_{RAS}가 필요하다. 마지막으로 Row Active 상태에서 읽기 동작으로 활성화되는 경우 DRAM에서 데이터를 전송하기 전까지 CAS Latency 만큼의 추가 지연이 발생하여 식 (1)과 같이 표현할 수 있다. 그러나 쓰기 동작은 추가 지연 시간이 없으므로 식 (1)에서 CAS Latency 항을 제거해 표현할 수 있다.

$$t_{READ} = t_{RAS} + t_{RP} + CAS\ Latency \quad (1)$$

DRAM의 내부는 여러 개의 bank로 구성이 되어 있으며 각각의 데이터에 접근하기 위해서는 계속해서 식 (1)과 같은 지연 시간이 필요하다. 그러나 특정 bank에 연속으로 접근 하지 않는 경우, bank 활성화와 주소로의 접근 과정을 병렬적으로 처리할 수 있기 때문에 이러한 지연 시간을 줄일 수 있게 된다. 본 논문에서는 이러한 DRAM의 Multi-bank 기능을 이용해 효율적으로 대역폭을 활용하는 스케줄링 기법을 SSD 내부에 있는 Arbiter를 통해 구현하려고 한다.

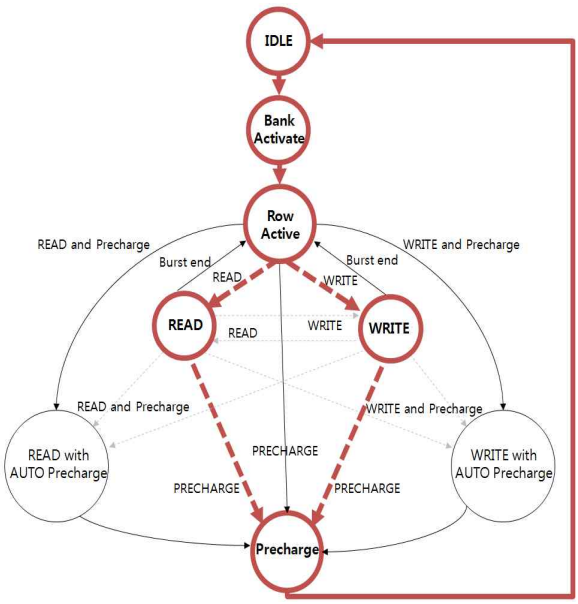


그림 3. SDRAM의 상태도
Fig. 3. Sate diagram of SDRAM.

그림 4는 그림 1의 점선으로 표시되어 있던 부분을 상세하게 표현한 것이다. SSD의 내부에는 그림 4의 블록도와 같이 각 마스터가 DRAM 버퍼에 접근 하는 것을 중재하기 위한 Arbiter가 존재한다. 마스터는 호스트 버퍼 컨트롤러와 각 채널별로 존재 하는 NAND 플래시 버퍼 컨트롤러이다. 두 개 이상의 마스터들이 동시에 읽기 또는 쓰기 요청을 하는 경우 Arbiter에서 이러한 마스터들의 요청을 중재하여 접근 요청 순서를 조정하게 된다. 본 논문에서 제안하는 DRAM 버퍼 스케줄링 기법은 연속으로 같은 bank에 접근하는 것을 방지하기

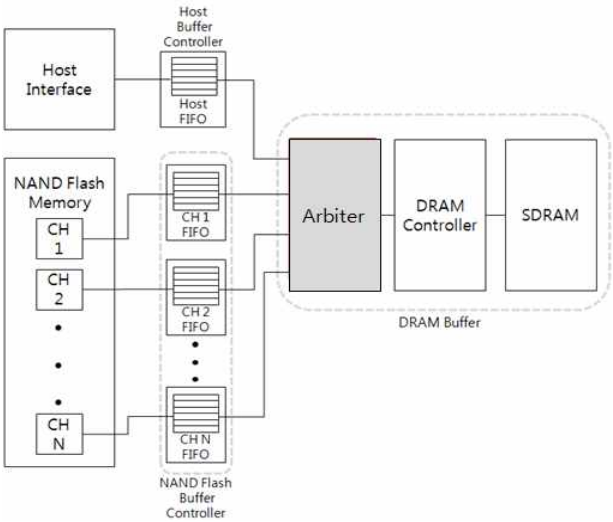


그림 4. SSD 상세 내부구조도
Fig. 4. Detailed Block Diagram of SSD.

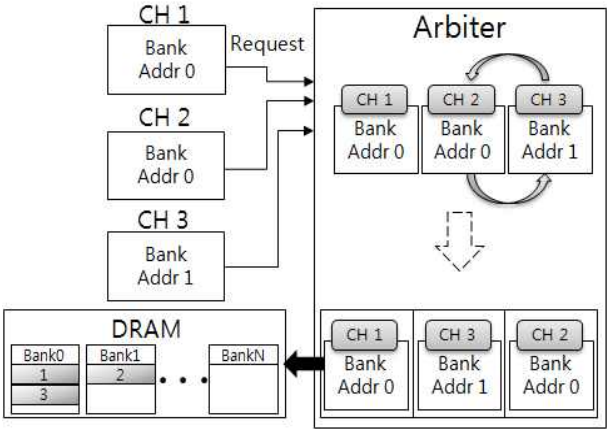


그림 5. 제안하는 스케줄링 기법
Fig. 5. Proposed scheduling techniques.

위해 동시에 Arbiter에 들어온 요청을 DRAM 버퍼의 다른 bank에 접근하여 처리하도록 순서를 조정한다.

예를 들어 그림 5에서 현재 Arbiter에 3개의 채널이 각각의 bank 주소를 가지고 요청을 한다고 가정한다. Arbiter는 들어온 3개의 요청들을 모니터링 하여 채널 1과 채널 2가 같은 bank 주소인 bank Addr 0이고, 채널 3에서는 다른 bank 주소인 bank Addr 1인 것을 확인한다. 초기에는 Round-Robin 방식에 따라서 CH 1, CH 2, CH 3로 접근 순서를 정한다. 그리고 연속으로 DRAM 버퍼의 같은 bank에 접근을 줄이기 위해 Arbiter는 CH 1(bank addr 0), CH 3(bank addr 1), CH 2(bank addr 0)로 재조정하여 각 요청이 요구하는 DRAM 버퍼로의 접근을 병렬적으로 처리하도록 한다.

제안하는 스케줄링 기법의 순서도는 그림 6과 같다. 병렬적인 처리를 위해서 각 채널에서 요청하는 bank 주소를 확인한 후, DRAM 버퍼의 특정 bank에 연속으로 접근을 하지 않도록 순서를 재조정한다. 제안하는 스케줄링 기법에서는 1 개의 채널에서 데이터가 전송되는 동안 Arbiter에서 다음 접근 권한의 순서를 조정하게 되므로, 중재로 인한 추가 지연 시간은 발생하지 않게 된다. 접근 권한을 받은 채널과 DRAM 버퍼 간의 데이터 전송을 할 때, Arbiter는 데이터 전송이 완료되기 전의 일정 시간($T_{activate}$) 동안 다음 접근 권한을 획득하는 채널의 주소와 Burstreq 신호를 DRAM 컨트롤러로 전송한다.

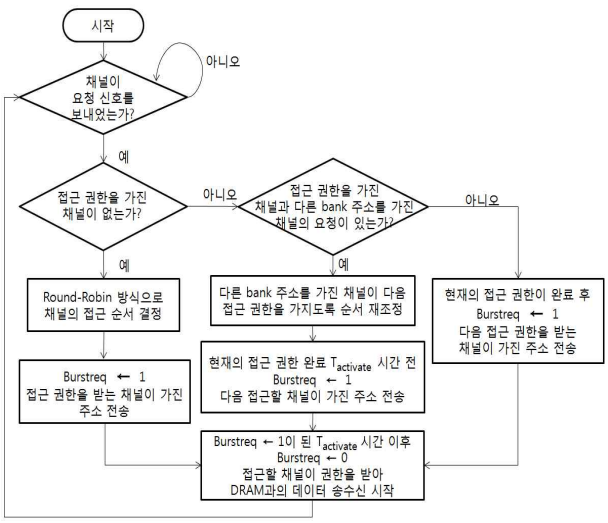


그림 6. 제안하는 DRAM 버퍼 액세스 스케줄링 기법의 순서도

Fig. 6. Flow chart of proposed DRAM buffer access scheduling techniques.

표 1. DDR2 SDRAM의 파라미터

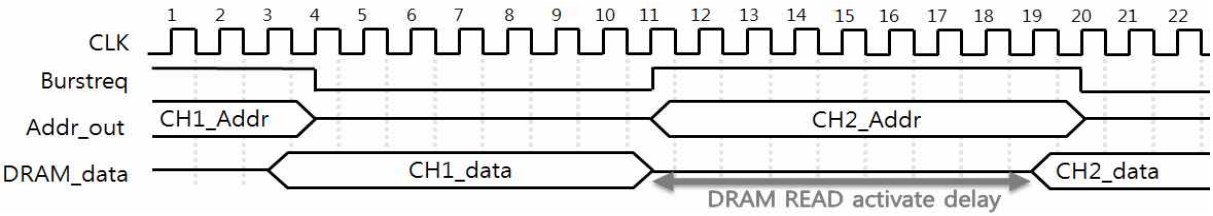
Table 1. Parameter of DDR2 SDRAM.

Symbol	Value	Unit
Clock	100	MHz
t_{RAS}	40	ns
CAS Latency	3	clock cycle
t_{RP}	15	ns

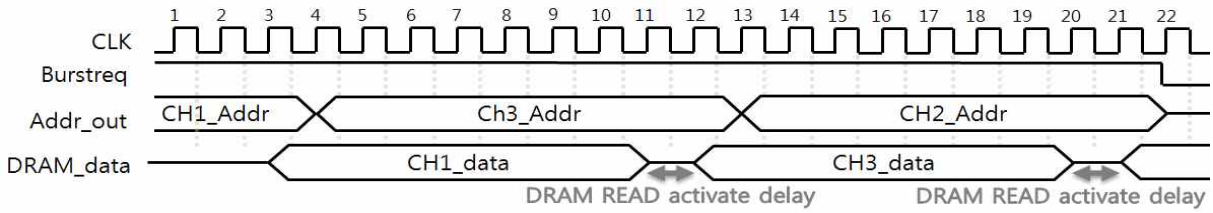
Burstreq 신호는 Arbiter에서 DRAM 컨트롤러로 요청하는 신호이며, Tactivate는 DRAM의 읽기/쓰기 동작이 활성화되어 데이터가 전송 혹은 수신되기 전까지 시간이다. 현재 접근 권한을 가진 채널과 DRAM 버퍼 간의 데이터 전송이 완료되면, 다음 접근 권한을 받는 채널은 줄어든 DRAM 활성화 지연 시간 이후에 데이터를 전송하거나 수신하게 된다.

본 논문에서 모델링에 사용된 삼성전자의 DDR2-400 SDRAM(K4T51083QG-HCCC)은 용량은 64 MByte이며, 대역폭은 3200 MByte/s로 지연 시간 계산에 필요한 파라미터 값은 표 1과 같다. 이를 기준으로 DRAM의 읽기 동작에서 기존의 스케줄링 기법과 제안하는 스케줄링 기법을 비교한 그림 7의 타이밍도에 대해 설명한다. 3 개의 NAND 플래시 채널이 DRAM 버퍼로 접근을 요청하는 예를 들었으며 채널 1과 채널 2는 같은 bank 주소를 갖고, 채널 3은 다른 bank 주소를 가지고 있다. Addr_out은 다음 접근 권한을 받는 채널의 주소를 Arbiter에서 DRAM 컨트롤러로 전송하는 신호로써, DRAM_data는 Arbiter와 DRAM 컨트롤러 간에 전송되는 데이터를 의미한다. DDR2 SDRAM의 경우 클럭당 16바이트 데이터가 전송되며, 버스트 사이즈가 128 바이트인 경우 전송 cycle은 8 cycle이 된다.

그림 7 (a)와 같이 기존의 스케줄링 기법에서는 Arbiter에서 bank 주소에 따라 순서를 재조정하지 않는다. 채널 1이 데이터 전송을 받은 후에, 같은 bank 주소를 가진 채널 2가 DRAM 버퍼에서 수신할 데이터의 주소를 전송한다. 그러므로 bank 활성화, RAS 그리고 CAS 지연 시간이 발생되며, 표 1의 파라미터에 의해 t_{READ} 시간을 계산하면 지연 시간은 9 cycle이 된다. 반면에 그림 7 (b)의 제안하는 스케줄링 기법은 Arbiter에서 bank 주소에 따라 순서를 재조정한다. DRAM 버퍼와 채널 1 간의 데이터가 전송되는 동안, 채널 1과 다른 bank 주소를 가진 채널 3이 DRAM 버퍼에서 수신할 데이터의 주소를 전송하므로 지연 시간을 감소시킬 수 있다. 즉, 데이터 전송이 이루어지는 동안 지연 시간을



(a) 기존의 스케줄링 방식



(b) 제안하는 스케줄링 방식

그림 7. 그림 5의 예에 대한 기존의 스케줄링 기법과 제안하는 스케줄링 기법의 타이밍도 비교
Fig. 7. Comparison of timing diagram of conventional DRAM scheduling techniques and proposed DRAM scheduling techniques about Fig 5.

줄이기 때문에 최대 8 cycle의 지연을 감소하여 단지 1 cycle의 지연 후에 다음 데이터 전송이 이루어 질 수 있다.

Ⅳ. 모델링 및 분석

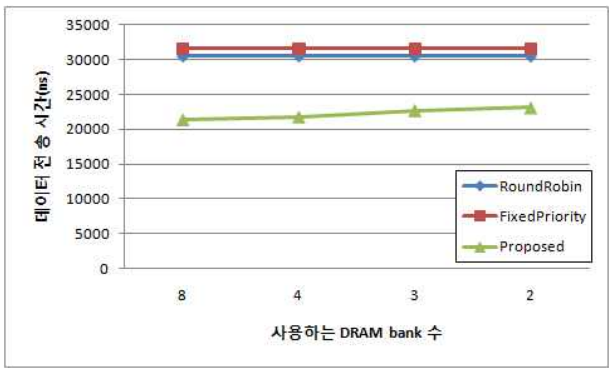
제안하는 스케줄링 기법을 사용하는 SSD 컨트롤러를 Verilog HDL로 모델링하고, Mentor Graphics사의 ModelSimXE III 6.3c - Custom Xilinx 버전으로 시뮬레이션을 하였다. 또한 제안하는 스케줄링 기법을 사용하는 Arbiter와 기존의 스케줄링 기법을 사용하는 Arbiter 면적을 비교하기 위해 동부하이텍 130nm 라이브러리로 합성을 해서 결과를 얻었다.

DDR2를 기준으로 한 SDRAM은 클럭당 4 바이트 크기의 데이터 전송이 4번 이루어지며 NAND 플래시 채널은 데이터 인터페이스가 1바이트로 되어 있다. 호스트 버퍼 컨트롤러의 FIFO와 NAND 플래시 버퍼 컨트롤러의 FIFO 사이즈는 256 바이트이다. 버퍼 컨트롤러는 DRAM 버퍼에서 데이터를 읽는 경우, FIFO의 공간이 128 바이트 이상이면 DRAM 버퍼 접근을 요청한다. 반대로 데이터를 쓰는 경우, FIFO에 128 바이트 이상의 데이터를 채우면 버퍼 컨트롤러는 DRAM 버퍼 접근을 요청한다. SSD에서 호스트와 연결되어 있는 SATA 인터페이스는 300MByte/s로 동작을 하며. DDR2 SDRAM(K4T51083QG-HCCC)의 동작 클럭 주파수는 100MHz이다. NAND 플래시는 현재 표준 스펙인 133MByte/s에서 200MByte/s로 대역폭을 증가시키고 NAND 플래시의 채널은 16개로 시뮬레이션을 진행하였다. 실제 SSD에서는 DDR2 SDRAM에서 대역폭 133MByte/s인 NAND 플래시 8 채널을 사용하기 때문에 대역폭이 부족한 상황이 발생하지 않는다. 그러므로 NAND 플래시 성능의 개선이나 플래시 채널의 확장으로 인한 DRAM 버퍼의 대역폭이 부족한 상황에서 실험하기 위해 NAND 플래시의 대역폭은 높이고 채널의 수를 증가 시켜 실험을 하였다.

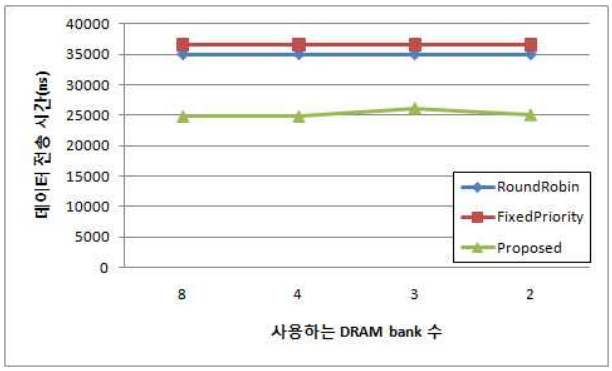
각 스케줄링 기법 사이의 정확한 성능 비교를 위해 동일한 조건을 정하였다. 전송하는 데이터의 양은 2 섹터인 1024 바이트로 하였고, 전송 시 버스트 사이즈는 128 바이트로 고정 하였다. 스케줄링 방식에 상관없이 호스트 인터페이스는 가장 높은 우선순위를 가지며 DRAM 버퍼에 요청하는 경우에 항상 hit한다고 가정한다. 성능 측정은 DRAM 버퍼에 접근을 요청하는 모든 채널이 읽기/쓰기 명령을 보내는 시점부터 요청한 모든 데이터의 전송이 완료 할 때 까지 걸리는 시간을 데이터 전송 시간이라 정의하고 측정을 하였다. 기존의 스케줄링 기법인 Fixed Priority와 Round-Robin과의 데이터 전송 시간을 비교 분석하였다. TDM+Round-Robin의 경우, 본 논문 2장에서 설명한대로 2단계 중재로 인한 추가 지연 시간을 소모해 다른 방식에 비해 성능이 떨어져 실험에서 제외하였다.

그림 8은 16개의 NAND 플래시 채널이 8개의 DRAM bank 중 사용하는 수에 따른 결과를 보여준다. 각 플래시 채널과 버퍼 컨트롤러 내부에 있는 FIFO와의 데이터 전송이 FIFO와 DRAM 버퍼 간의 데이터 전송보다 느리다. 그러므로 한 개의 채널에서 데이터 전송이 완료되는 시점까지 요청하는 시간보다 요청을 하지 않는 시간이 길게 되는 것이다. Round-Robin 방식은 FIFO에 데이터가 채워진 뒤, 플래시 채널에서 데이터를 가져가는 동안에 다음 채널이 접근 권한을 받고 FIFO에 데이터를 채우기 때문에 모든 채널이 비슷한 시점에서 데이터 전송이 끝나게 된다. 반면에 Fixed Priority 방식은 우선순위가 높은 마스터 간에서만 접근 권한을 넘기면서 데이터의 전송이 완료되고, 결국 우선순위가 낮은 채널만 남아서 요청을 하므로 데이터 전송을 완료하는 시간이 길어진다. 본 논문에서 제안하는 DRAM 액세스 스케줄링 방식은 접근 권한을 가진 채널의 bank 주소와 다른 bank 주소를 가진 채널이 다음 접근 권한을 가지도록 순서를 정리한 뒤에, 그 채널이 가진 주소를 미리 DRAM에 전달을 해서 bank 활성화, RAS와 CAS의 지연을 감소시켜 더 좋은 성능을 보여주게 된다. 즉 기존의 스케줄링 기법에서 DRAM 버퍼가 데이터를 전송 시 지연 시간이 9 cycle이고, 역으로 DRAM에서 데이터를 수신하는 경우는 6 cycle이다. 제안하는 스케줄링 기법은 이러한 지연시간을 줄임으로써 성능 개선의 효과를 얻게 된다. 제안하는 스케줄링 방식은 16개의 채널에서 8개의 DRAM bank를 모두 사용할 때, 가장 좋은 성능을 보여준다. 반면에 플래시 채널에서 사용하는 bank 수가 적을수록 데이터 전송 시 간이 느리다. 이는 지속적으로 접근 권한을 받지 못한 채널만이 남아서 Arbiter에 요청을 하기 때문에 요청하는 시간 간격이 길어지고 다른 bank 주소를 가진 채널은 데이터 전송이 완료되어 DRAM 활성화에 필요한 지연시간을 줄일 수 없기 때문이다.

그림 9는 8개의 DRAM bank를 할당받은 16개의 NAND 플래시 채널 중 동시에 요청하는 NAND플래시 채널수에 따른 데이터 전송 시간을 측정한 결과이다.

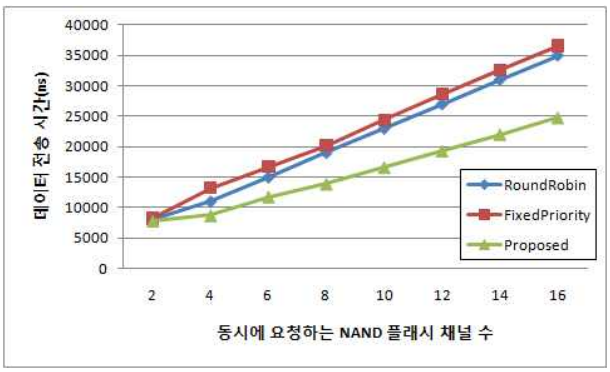


(a) SSD 읽기 동작

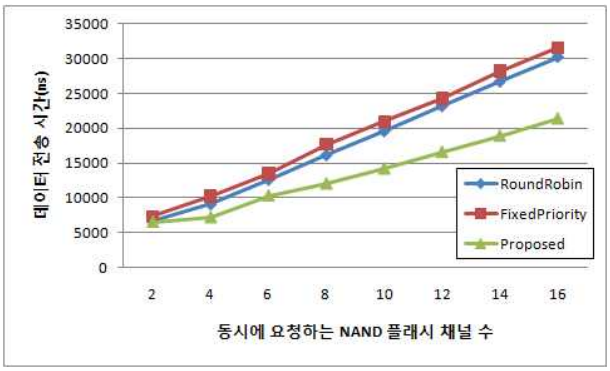


(b) SSD 쓰기 동작

그림 8. 사용하는 DRAM bank 수에 따른 성능 비교
Fig. 8. Performance comparison according to using DRAM banks.



(a) SSD 읽기 동작



(b) SSD 쓰기 동작

그림 9. 동시에 요청하는 NAND 플래시 채널수에 따른 성능 비교
Fig. 9. Performance comparison according to request NAND Flash channels at the same time.

표 2. 스케줄링 방식에 따른 Arbiter의 면적
Table 2. Arbiter size in accordance with scheduling method.

	Fixed Priority	Round Robin	Proposed
Area (um^2)	20934	21772	22399
로직게이트(gates)	4187	4355	4480

소수의 NAND 플래시 채널이 동시에 요청할 때에는 기존의 스케줄링 방식에 비해 성능향상 면에서 별 차이가 없다. 그러나 기존의 스케줄링 기법에 비해 제안하는 스케줄링 기법은 데이터 전송 완료 후 다음 데이터 전송 시작까지 더 적은 지연 시간이 소모되기 때문에 동시에 요청하는 NAND 플래시 채널수의 증가에 따라 더 높은 성능 향상을 보인다. 동시에 요청하는 NAND 플래시 채널의 수가 16개인 경우, 제안하는 방식은 Fixed Priority와 Round Robin 방식에 비해 읽기 성능 면에서는 각각 47.4%와 41.2%의 성능 향상을 보이며, 쓰기 성능 면에서는 47.7%와 41.3%의 성능 개선을 확인할 수 있었다.

표 2는 동부하이텍 130nm 공정을 통해서 합성을 한 후에 기존의 스케줄링 방식을 가진 Arbiter와 제안한 스케줄링 방식을 가지는 Arbiter의 사이즈를 측정한 값이다. Fixed Priority 방식에 비해서는 7%가, Round-Robin 방식에 비해서는 3%가 커지는 것으로 나타났다. 그러나 SSD 컨트롤러의 로직 게이트 수가 약 500만개 이상인 것을 감안하면 칩 내에서 제안하는 DRAM 버퍼 액세스 스케줄링 방식을 가진 Arbiter에 의한 면적의 증가는 0.01%로 무시할만한 수준이다.

V. 결 론

본 논문에서는 비용 효율적인 방법을 통해 SSD 성능을 향상 시키는 연구의 일환으로 SSD내의 DRAM 버퍼의 대역폭을 효율적으로 사용하는 스케줄링 기법을 제안하였다. 각 마스터들의 요청 시, 이들이 가진 bank 주소를 확인하고, 연속으로 특정 bank의 접근을 피하여 DRAM Multi-bank를 활용할 수 있게 하였다. 이를 이용해서 bank 활성화를 위한 시간과 RAS와 CAS로 생기는 지연 시간을 줄여 DRAM 버퍼 대역폭의 효율적인 활용이 가능하게 되었다. 제안하는 스케줄링 방식을 가지는 Arbiter 면적은 전체 칩에서 0.01% 증가로 무시할 만한 수준이며 기존의 스케줄링 기법에 비해 SSD 읽기 및 쓰기 성능은 각각 최대 47.4% 와 47.7%의 개선됨을 확인할 수 있었다.

참 고 문 헌

[1] http://www.segate.com/www/en-us/support/before_you_buy/speed_considerations

[2] L.P. Chang, “Hybrid solid-state disks: Combining heterogeneous NAND flash in large SSDs”, Design Automation Conference, pp.428-433, April 2008.

[3] J. Ryu and C. Park, “Analysis of Embedded Software Design Affecting SSD Performance”, Journal of KIISE, vol.27, no.5, pp.58-68, May 2009.

[4] http://www.samsung.com/global/business/semiconductor/products/flash/Products_Toggle_DDR_NAND Flash.html

[5] H. Zheng, J. Lin, Z. Zhang and Z. Zhu, “Decoupled DIMM: building high-bandwidth memory system using low-speed DRAM devices”, 36th annual international symposium on Computer architecture, pp.255-266, June 2009.

[6] L. Chang, “Hybrid solid-state disks: Combining heterogeneous NAND flash in large SSDs”, Asia and South Pacific Design Automation Conference ASPDAC 2008, pp.428-433, March 2008.

[7] J. H. Yoon, E. H. Nam, Y. J. Seong, H. Kim, S. L. Min and Y. Cho, “A High Performance Flash Memory Solid State Disk”, Journal of KIISE, Vol.14, No.4, pp.378-388, June 2008.

[8] B. Ha, H. Cho and Y. I. Eom, “WADPM : Workload-Aware Dynamic Page-level Mapping Scheme for SSD based on NAND Flash Memory”, Journal of KIISE, Vol.37, No.4, pp.215-225, August 2010.

[9] K. Lahiri, A. Raghunathan and G. Lakshminarayana, “The LOTTERYBUS on-chip communication architecture”, VLSI(Very Large Scale Integration) Systems, Vol.14, No.6, pp.596-608, June 2006.

저 자 소 개

박 준 수(학생회원)

2010년 광운대학교 정보제어 공학과 학사 졸업.

2010년 3월～현재 성균관대학교 임베디드소프트웨어학과 석사과정.

<주관심분야 : 임베디드 시스템, SSD 컨트롤러>

한 태 희(평생회원) - 교신저자

1992년 KAIST 전기 및 전자공학과 학사.

1994년 KAIST 전기 및 전자공학과 석사.

1999년 KAIST 전기 및 전자공학과 박사.

1999년 3월～2006년 8월 삼성 전자 통신연구소 책임 연구원.

2006년 9월～2008년 2월 한국산업기술대학교 전자공학과 조교수.

2008년 3월～현재 성균관대학교 정보통신공학부 반도체시스템공학 전공 부교수.

<주관심분야 : IT SoC 설계 및 설계 방법론, 단말 시스템, IT 융합 기술>

황 용 중(학생회원)

2010년 성균관대학교 전자전기 공학과 학사 졸업.

2010년 3월～현재 성균관대학교 정보통신공학부 석사과정.

<주관심분야 : 임베디드 시스템, SoC 설계>