논문 2011-48SD-7-8

SSD 성능 향상을 위한 DRAM 버퍼 데이터 처리 기법

(DRAM Buffer Data Management Techniques to Enhance SSD Performance)

임 광 석*, 한 태 희**

(Kwang Seok Im and Tae Hee Han)

요 약

SSD(Solid State Disk)는 호스트 인터페이스와 낸드 플래시 메모리의 대역폭 차이를 완충하기 위한 버퍼로 DRAM을 적용하고 있다. 본 논문에서는 대역폭이 높은 고가의 DRAM을 사용하는 대신 저비용으로 SSD의 성능을 향상시킬 수 있는 효과적인 방법을 제안하였다. SSD 데이터는 사용자 데이터, 사용자 데이터 관리를 위한 메타데이터, 데이터의 오류 제어를 위한 FEC(Forward Error Correction) 패리티/CRC(Cyclic Redundancy Check) 등 크게 세 가지로 구분할 수 있다. 본 논문에서는 데이터 유형 별 특성을 고려하여 성능을 향상시키기 위해 모니터링 시스템을 통한 가변적인 버스트 데이터 처리 방법과 페이지 단위를 이용한 FEC 패리티/CRC 방식을 적용하였다. 실험을 통하여 0.07%의 무시할만한 칩 면적의 증가만으로 평균 25.9%의 SSD 성능 개선을 확인할 수 있었다.

Abstract

To adjust the difference of bandwidth between host interface and NAND flash memory, DRAM is adopted as the buffer management in SSD (Solid-state Disk). In this paper, we propose cost-effective techniques to enhance SSD performance instead of using expensive high bandwidth DRAM. The SSD data can be classified into three groups such as user data, meta data for handling user data, and FEC(Forward Error Correction) parity/ CRC(Cyclic Redundancy Check) for error control. In order to improve the performance by considering the features of each data, we devise a flexible burst control method through monitoring system and a page based FEC parity/CRC application. Experimental results show that proposed methods enhance the SSD performance up to 25.9% with a negligible 0.07% increase in chip size.

Keywords: SSD, DRAM buffer, Burst data control, Page based FEC parity/CRC

Ⅰ. 서 론

낸드 플래시 메모리를 기반으로 하는 SSD는 기존의 HDD에 비해 훨씬 더 높은 대역폭을 갖고 있고, 랜덤 입출력 데이터에 대해서도 우수한 성능을 보이고 있다. 또한 고속 회전하는 디스크를 헤드로 읽고 쓰는 기계적 방식의 HDD에 비해 내구성, 신뢰성이 뛰어나고, 전력 소모 측면에서도 많은 장점이 있으며 작은 폼 팩터(Form Factor)와 유연성 있는 설계를 통해서 더욱 혁신적인 제품 개발이 가능 하다.[1] 최근 SSD는 이런 이유로 모바일 향 제품에서 점차적으로 기업 형 서버제품으로 확대 적용되고 있는 추세이며 이러한 SSD의 확산에 따라 비용 효율적 성능 향상이 매우 중요한 이슈로 부각되고 있다.

SSD는 호스트와 낸드 플래시 메모리 인터페이스의 대역폭 차이를 완충하기 위한 버퍼로 DRAM을 적용하고 있다. 고속 호스트 인터페이스 규격은 SATA Ⅰ 1.5Gbps, SATAⅡ 3Gbps 에서 SATAⅢ 6Gbps 나아가 SAS , PCI-Express로 빠르게 변화하고 있고, 낸드 플래시 대역폭은 메모리 자체의 디바이스 적인 성능 향상과 더불어 구조적으로 다중 채널 병렬 처리를 통해 향상시킬 수 있다.[2]

* 학생회원, ** 평생회원, 성균관대학교 정보통신공학부
(School of Information Communication Engineering, Sungkyunkwan University)
접수일자: 2011년4월16일, 수정완료일: 2011년7월1일

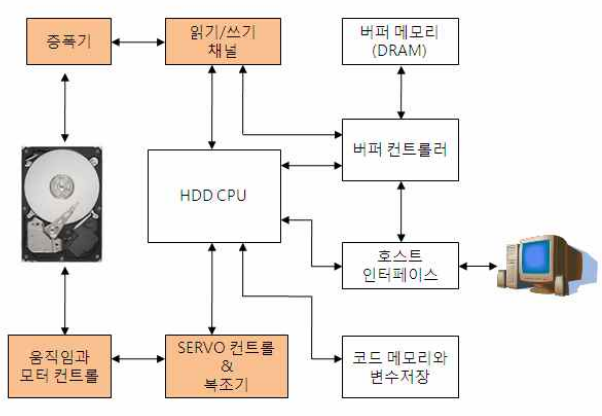
SSD의 데이터는 사용자 데이터, 메타데이터, FEC (Forward Error Correction) 패리티/ CRC(Cyclic Redundancy Check)[3] 등 세 가지로 구분 가능하다. 첫째 사용자 데이터는 저장 용량의 대부분을 차지하는 파일의 내용물 자체로 호스트 명령에 실린 시작주소와 섹터 크기 정보를 통해 전달된다. 둘째 메타데이터는 낸드 메모리의 Bad Block 처리와 호스트의 논리적 주소를 물리적 주소로 연결해주는 매핑(mapping) 테이블 관리에 사용된다.[4] 다중 채널의 메타데이터 처리 향상을 위해서는 멀티 CPU를 필요로 하는데[5], 그 예로 최근 출시된 삼성전자의 SSD인 S470 시리즈에는 세 개의 CPU가 탑재되어 있다.[6] 셋째 FEC 패리티/CRC 데이터는 사용자 데이터의 오류 제어를 위하여 사용된다.

본 논문은 SSD의 DRAM 인터페이스 구조를 통해서 전송 데이터의 효율적인 접근 방법을 제시하고 최종적으로 전체 시스템의 성능향상을 위한 데이터 처리 기법을 제안한다. 논문은 다음과 같이 구성된다. Ⅱ장은 SSD 특성과 DRAM 버퍼 성능 개선 포인트를 설명하고, 제안된 성능 개선 방안은 Ⅲ장에서 보인다. Ⅳ장에서는 제안된 시스템의 실험결과를 나타내고, 마지막으로 Ⅴ장에서 결론을 맺는다.

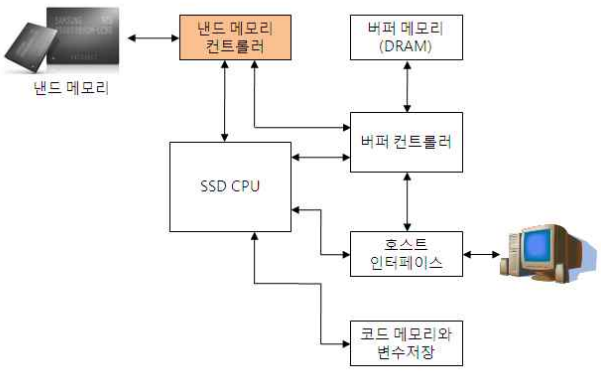
Ⅱ. SSD 특성과 DRAM 버퍼 성능 개선 포인트

일반적인 HDD의 구조는 그림 1(a)와 같다. HDD는 채널을 통해서 아날로그 신호를 처리해야 하며 모터를 통해서 디스크에 접근하는 움직임을 관리하기 때문에 모터 제어 부분과 servo 제어 부분이 존재한다. 이러한 기계적 특성 때문에 HDD는 외부 충격에 취약하고 전력 소모도 많을 뿐만 아니라, 랜덤데이터를 연속적으로 접근하는 경우에는 모터를 지속적으로 움직이면서 디스크 위치를 확인해야 하므로 성능저하가 발생한다. 반면에 SSD는 HDD에서 채널과 모터를 제어하는 부분을 낸드 메모리를 지원하는 컨트롤러가 대신하고 있어 그림 1(b)처럼 훨씬 간단한 구조로 되어있다.

그림 2는 SSD의 DRAM 인터페이스 구조이다. 호스트 버퍼 컨트롤러 부분은 SATA 인터페이스를 통해 호스트에서 요청하는 경우 데이터를 잠시 저장해 두었다가 Arbiter에게 요청한다. 낸드 버퍼 컨트롤러 부분은 낸드 메모리 컨트롤러에서 요청한 데이터를 처리하는 부분으로 다중 채널로 구성되어 있고 병렬 처리된다. 그리고 DRAM 에 접근하는 호스트, 낸드 메모리, CPU



(a) HDD 구조[7]



(b) SSD 구조

그림 1. HDD 와 SSD의 구조
Fig. 1. Structure of SSD and HDD.

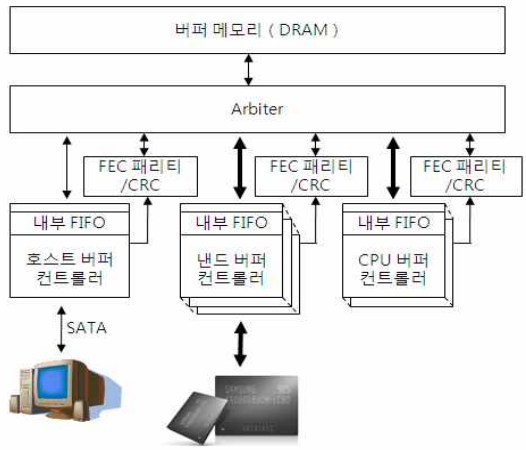


그림 2. SSD의 DRAM 인터페이스 구조
Fig. 2. Structure of DRAM interface in SSD.

데이터 각각은 에러 검출과 복원을 위해 FEC 패리티/CRC 블록을 통해 DRAM과 통신한다.

최근에 EMC 와 HP 사에서는 서버 향 제품으로도 SSD를 출시하고 있다.[8] 서버 분야에서 보다 강조되는 문제는 데이터의 신뢰성에 관련한 부분으로 고속데이터

처리 때문에 발생할 수 있는 오류를 보완하기 위하여 FEC 패리티/CRC를 이용하여 에러를 제어할 수 있어야 한다. FEC에서는 대표적으로 BCH, Reed Solomon 코드, 그리고 최근에는 고성능을 위해 LDPC(Low Density Parity Check) 코드가 쓰이며, CRC로는 CRC16 또는 CRC32가 사용된다.

1. DRAM 버스트 모드를 통한 성능 향상

버스트 방식은 블록 단위로 된 일정량의 데이터를 인터럽트가 발생하지 않는 경우 모두 전달될 때 까지 중단하지 않고 연속으로 전송하는 방식을 의미한다. 따라서 버스트 크기가 증가할수록 처리할 수 있는 데이터는 증가하지만 내부에 저장하는 FIFO의 사이즈가 증가하는 단점이 있기 때문에 trade-off를 고려해야 한다. 식 (1)은 fMHz로 동작하는 32비트의 DDR 메모리에서의 유효 데이터 전송속도를 나타낸 것이다.

$$EF_{data} = \frac{Cycle_{burst} \times f \times 10^6[1/s] \times 2 \times 32[Bit]}{Cycle_{latency} + Cycle_{burst}} \quad (1)$$

EF_{data}는 유효 데이터 전송속도를 의미하며 DRAM에서 대기시간을 제외한 전송 데이터의 속도를 나타낸다. $Cycle_{burst}$는 버스트 사이클, $Cycle_{latency}$는 지연시간 사이클을 의미한다. 지연시간은 DDR2 스펙에 의해 CL(CAS Latency) 6사이클, T_{RCD}(RAS to CAS Delay) 6사이클, DDR2 PHY에 의해 2사이클, 컨트롤러와 DDR2 인터페이스에서 2사이클이 지연되어 평균 $Cycle_{latency}$은 16 사이클이 소요된다. 예를 들어 400MHz DDR2 메모리에서 133MByte/s의 8채널 낸드를 사용하고, SATAⅢ(대역폭은 600MByte/s)를 이용하면 전체 필요한 대역폭은 133×8 + 600 = 1.66GByte/s가 된다. 식(1)을 통하여 $Cycle_{burst}$가 16일 경우 대역폭이 1.6 GByte/s가 되므로, rising/falling edge를 고려하면 4Byte×32 = 128Byte 버스트 크기로 최적화된다. 버스트 크기가 결정된 후 버퍼 컨트롤러에서 사용되는 내부 FIFO 크기는 연속적인 버스트의 빠른 데이터 처리를 위해서 이중 버퍼인 256Byte를 사용하게 된다.

2. SSD 데이터 별 성능 향상 방안

사용자 데이터의 내부 FIFO는 연속적인 처리를 위해 버스트 크기의 두 배로 구성되어 있는데 DRAM 전송이 지연되어 FIFO에 데이터가 가득 찬 경우에는 버스트 크기를 FIFO 사이즈에 최적화하여 증가시키는 방법을 제안한다.

CPU로 액세스하는 메타데이터는 연속성이 보장되지 않고 랜덤접근이 많기 때문에 CPU 데이터 자체의 버스트 크기를 조절하는 것 보다는 SSD 데이터의 대부분을 차지하는 사용자 데이터에 대한 버스트 크기를 최적화하는 방법을 제안한다.

FEC 패리티/CRC 데이터의 접근 최소화 방법으로는 매 섹터가 아닌 페이지 단위의 동작을 통해 DRAM에 버스트 단위로 접근함으로써 성능을 향상시키는 방법을 제안한다.

Ⅲ. 제안하는 성능 개선 방안

본 논문에서는 SSD의 성능향상을 위하여 DRAM에서 병목 현상이 발생할 때의 지연시간을 최대한 줄이기 위한 방법으로 모니터링 기법을 통한 가변적인 DRAM 데이터 버스트 처리방법과, FEC 패리티/CRC 데이터를 매 섹터가 아닌 낸드 플래시의 페이지 단위로 한꺼번에 DRAM과 접근할 수 있는 방법을 제안한다.

1. DRAM 인터페이스의 모니터링 블록 구조

그림 3은 제안하는 DRAM 인터페이스 구조이다. 각 마스터에 대한 동작시간을 모니터링 할 수 있는 블록을 추가하여 지연이 발생하는 경우 Arbiter에 요청하는 버스트 데이터 크기를 최적화 하였다.

세부적인 지연 모니터링 블록과 순서도는 그림 4와 같다. 그림 4(a)에서 wait 신호는 지연이 발생할 때 생

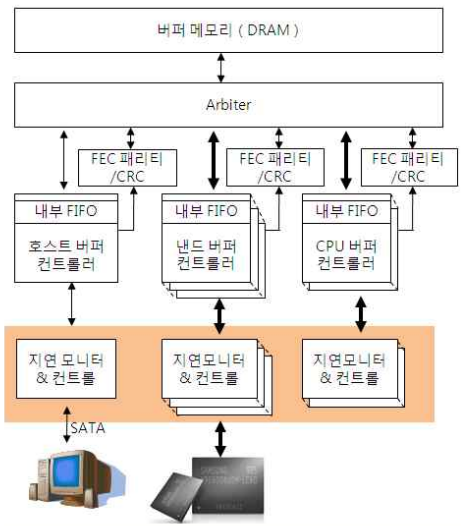
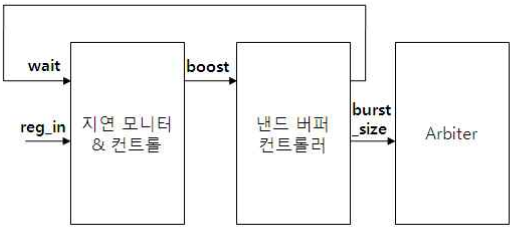
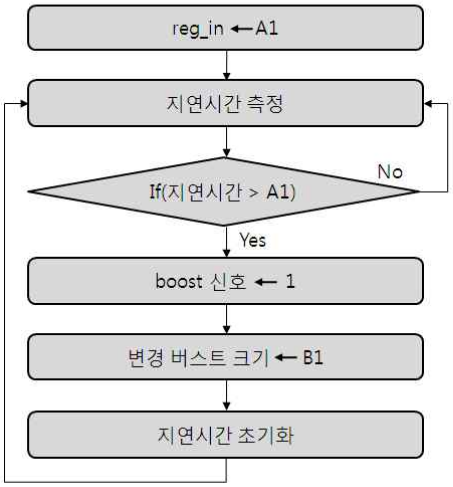


그림 3. 제안하는 DRAM 인터페이스 구조
Fig. 3. Proposed structure of DRAM interface.



(a) 지연 모니터링 블록 다이어그램



(b) 지연 모니터링 순서도

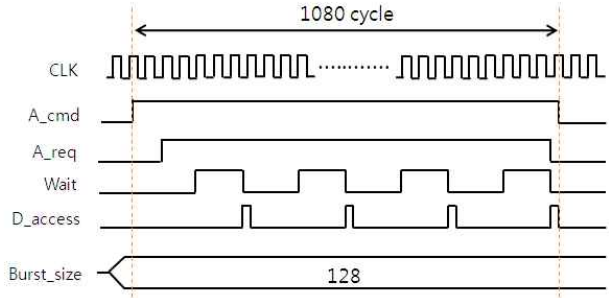
그림 4. 지연 모니터링 블록 다이어그램과 순서도
Fig. 4. Delay monitoring block diagram and flowchart.

성되고, reg_in은 레지스터 값으로 wait 신호의 지연시간과 일치되었을 때 boost 신호에 의해 burst_size를 최적화 한다.

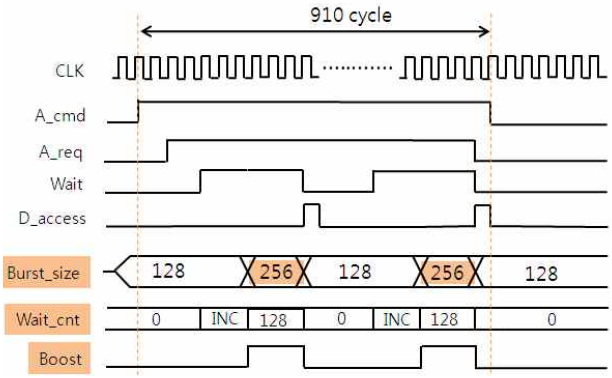
$$\text{reg\_in(A1)} = \frac{BurstSize_{aft} - BurstSize_{bef}}{DataSize_{1cycle}} \quad (2)$$

reg_in(A1)은 버스트 크기의 변경 조건을 만족하는 지연시간 사이클을 의미하고, $DataSize_{1cycle}$ 은 하나의 사이클 당 데이터 전송량, $BurstSize_{aft}$ 는 변경 버스트 크기, $BurstSize_{bef}$는 기존 버스트 크기를 의미한다. DDR2, SATAⅢ, 8채널 낸드의 경우 변경 가능한 최대 버스트 사이즈는 내부 FIFO 크기인 256Byte이므로 그림 4(b)의 B1인 $BurstSize_{aft}$ 는 256Byte가 되고 $BurstSize_{bef}$는 128Byte이다. $DataSize_{1cycle}$은 낸드 메모리의 경우 1Byte이고 호스트는 4Byte 이므로, 낸드 메모리의 A1 값은 128이고 호스트의 A1 값은 32가 된다.

그림 5는 사용자 데이터의 모니터링 시스템을 통한



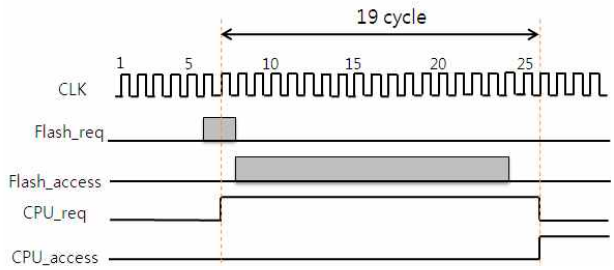
(a) 기존 방식의 사용자 데이터 타이밍도



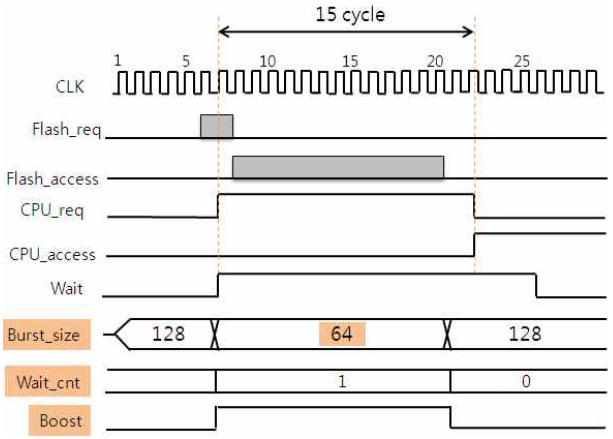
(b) 모니터링 시스템을 통한 사용자 데이터 타이밍도

그림 5. 기존 방식과 모니터링 시스템을 통한 사용자 데이터의 타이밍도 비교
Fig. 5. Comparison of timing diagram of conventional method and monitoring system in terms of user data.

가변 버스트 적용 전후의 예상되는 타이밍도이다. 그림 5(a) 에서 CLK 은 내부 버퍼 컨트롤러 클록으로 DRAM 컨트롤러의 클록보다 2배 느리다. A_cmd 신호는 낸드 채널의 한 섹터(512Byte)의 쓰기 명령 수행시간을 의미하고, A_req 신호는 FIFO에 128Byte의 데이터가 전송되었을 때 DRAM에 요청하는 신호이다. Wait 신호는 낸드 버퍼 컨트롤러에서 FIFO가 비워있는 경우(낸드 쓰기 동작)나 가득 찬 경우(낸드 읽기 동작)의 지연신호를 의미하는데 본 논문에서는 16채널을 모두 사용하는 경우에 wait 신호가 발생하게 된다. D_access 신호는 DRAM 인터페이스에서 데이터를 주거나 받는 경우에 발생되며 그림 5의 경우에는 DRAM의 읽기 명령에 해당된다. 그리고 Burst_size는 DRAM 동작에서 사용되는 버스트 크기를 의미한다. 그림 5(b)는 지연 모니터 및 컨트롤 블록을 이용한 경우의 타이밍도이다. 낸드 채널의 지연이 발생하는 시점부터 Wait_cnt 는 하나씩 증가하게 되고 128이 되는 순간부터는 boost 신호에 의해 버스트 크기가 내부 FIFO 최대 크기인



(a) 기존 방식의 메타데이터 타이밍도



(b) 모니터링 시스템을 통한 메타데이터 타이밍도

그림 6. 기존 방식과 모니터링 시스템을 통한 메타 데이터의 타이밍도 비교

Fig. 6. Comparison of timing diagram of conventional method and monitoring system in terms of meta data.

256Byte로 바뀌는 동작을 한다. 두 번의 DRAM 접근 동작만으로 명령이 수행되기 때문에 DRAM 인터페이스에서 발생하는 지연시간이 감소되어 기존의 1080사이클에서 910사이클로 시간이 단축된다.

메타데이터를 긴급하게 처리하는 경우는 호스트에서 NCQ(Native Command Queuing) 명령이 들어오는 경우이다. 이때에는 32개 명령까지 Queuing 되어 동시에 수행될 수 있기 때문에 각각의 명령에 대한 메타데이터의 빠른 처리를 위해 지연 시간이 발생한 순간부터 사용자 데이터의 버스트 크기를 최적화 한다.

그림 6은 메타데이터의 모니터링 시스템을 통한 사용자 데이터의 가변 버스트 적용 전후의 예상되는 타이밍도이다. 그림 6(a) 에서 Flash_req 신호는 낸드 채널의 요청 신호를 의미하며 Flash_access는 낸드 채널의 DRAM 접근신호이다. CPU_req는 메타데이터를 요청하는 신호이고, CPU_access 는 메타데이터가 DRAM에 접근함을 의미한다. 메타데이터는 사용자 데이터의 버스트 길이에 따라 지연시간이 증가하므로 전송속도 향상을 위해서는 SSD 데이터의 대부분을 차지하는 사용자 데이터의 버스트 크기를 줄여서 메타데이터가 빨리 DRAM을 점유할 수 있어야 한다.

그림 6(b)는 모니터링 시스템을 통한 사용자 데이터의 가변 버스트를 적용한 경우이다. 메타데이터를 우선적으로 처리하기 위하여 지연시간이 발생하는 순간부터 사용자 데이터의 버스트 크기를 최적화 하였다. 버스트 크기는 사용자 데이터의 성능 저하를 최소화하기 위해 128Byte에서 64Byte로 감소되어 메타 데이터의 지연시간이 19사이클에서 15사이클로 줄어든다.

2. 제안하는 FEC 패리티/CRC 데이터 인터페이스

기존의 HDD는 디스크로부터 데이터를 섹터 단위로 요청하지만, SSD는 최소 페이지 단위로 요청하기 때문에 페이지 단위의 접근을 통해 성능을 향상시킬 수 있었다. 제안하는 인터페이스에는 페이지 단위로 FEC 패리티/CRC 데이터를 저장할 수 있는 내부 FIFO가 필요하다. 현재 하나의 페이지를 4KB로 가정하면 FEC 패리티/CRC 크기는 매 섹터(512Byte) 당 16바이트가 필요하므로 16바이트 FIFO 8개가 필요하다.

그림 7은 낸드 메모리의 DRAM 쓰기 동작에서 FEC 패리티/CRC 동작을 나타낸 순서도이다. 기존에는 섹터당 매번 DRAM에 요청하였지만 개선 후에는 섹터 데이터가 연속적인 DRAM 주소를 접근하는 경우 해당페이지 만큼을 FIFO에 저장한 후에 버스트로 DRAM에 접근하는 방식으로 변경되었다. 예외적으로 DRAM 주소가 연속적이지 않은 경우에는 연속적인 부분까지의

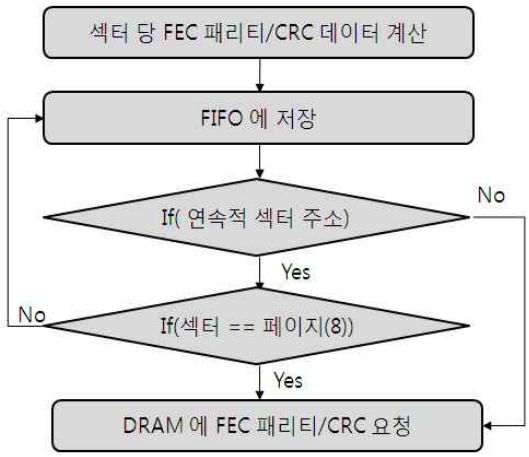


그림 7. 페이지에 따른 FEC패리티/CRC 요청 순서도

Fig. 7. Flowchart of FEC parity/CRC request according to page.

FEC 패리티/CRC 데이터만 DRAM에 전송되도록 설계 되었다.

Ⅳ. 실험 결과

가변적인 DRAM 데이터 버스트 처리 방법과, 페이지 단위의 FEC 패리티/CRC 동작을 적용한 SSD 컨트롤러는 Verilog HDL로 기술되었고, 삼성 65nm CMOS 라이브러리를 이용하여 합성하였다.

호스트는 SATAⅢ(대역폭은 600MByte/s)가 적용되었고 DRAM 인터페이스를 위해 순차데이터를 32비트의 병렬데이터로 가정하였다. 호스트 주파수를 150MHz로 하여 150MHz × 4Byte = 600MByte/s 로 대역폭을 맞추었고, 호스트 버퍼 컨트롤러에서 지연이 발생하는 경우에는 데이터가 전송되지 않도록 제어하였다. 낸드 메모리의 경우는 삼성제품인 토글 낸드 32Gbit K9GBGD8U0A[9]를 모델링 하였다. 입출력 핀 하나당 133MBit/s 로 전송을 하고 총 핀 수는 8핀이기 때문에 주파수는 133MHz 로 정하였고, 매 클록 당 1Byte로 하여 133MHz × 1Byte = 133MByte/s 로 맞추어 주었다. DRAM은 삼성전자의 1Gbit DDR2 K4T1G044QF[10]로 400MHz를 적용하였고, 내부 버퍼 컨트롤러는 완전 비동기적인 FIFO 구조로 200MHz로 설계 되었다. PVT(Process, Voltage, Temperature)를 고려하여 합성한 결과 8채널 전체 칩 500만 게이트에 비하면 0.07% 밖에 증가하지 않아 전체 면적에 미치는 영향은 무시할 만 하였다.

표 1. SSD 컨트롤러 합성 결과
Table 1. Result of SSD controller synthesis.

	변경 전	변경 후
전체 면적(um^2)	48862	52203
로직 게이트(gate)	38173	40783

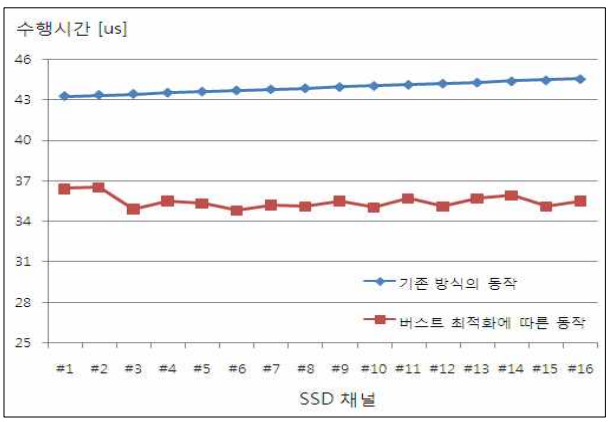


그림 8. 기존 방식과 버스트 크기 최적화에 따른 사용자 데이터의 성능 비교
Fig. 8. Comparison in performance of conventional method and burst size optimization in terms of user data.

1. 제안 적용 전/후의 동작 비교

가. 사용자 데이터의 전송 성능 비교

사용자 데이터의 성능을 측정하기 위하여 실험에서는 128바이트 버스트의 이중 버퍼인 256바이트 FIFO 크기에서 낸드 플래시 16채널을 사용하였을 때를 가정하였다. 위 실험에서는 16채널 모두 한 페이지(8섹터)를 낸드에 쓰기와 읽기 전송 후의 평균시간을 비교하였다. 전체적으로 최대 시간이 소요된 채널을 비교한 결과 44.54us에서 36.5us로 18%의 성능향상을 확인 할 수 있었다.

나. 메타 데이터의 처리속도 비교

표 2는 사용자 데이터의 지연에 의한 영향을 고려하지 않기 위해 8채널 낸드 동작에서 연속되지 않은 주소의 4바이트 데이터를 한번에서 16번까지 DRAM에 요청했을 때 DRAM까지 쓰기 읽기 동작을 측정한 평균 데이터이다. 평균적으로 11.8% 이상의 성능향상 수치를 얻을 수 있었다.

표 2. 기존 방식과 버스트 크기 최적화에 따른 메타 데이터의 성능 비교
Table 2. Comparison in performance of conventional method and burst size optimization in terms of meta data.

메타 데이터 (byte)	기존방식 (ns)	버스트 최적화 (ns)	향상도 (%)
4	125	105	16
8	275	235	14.5
12	425	365	14.1
16	640	560	12.5
32	1385	1205	12.9
64	2965	2265	11.8

다. FEC 패리티/CRC 동작의 전송 성능 비교

그림 9는 버스트 크기를 조절하지 않는 16채널 SSD 환경에서 섹터와 페이지에 따른 FEC 패리티/CRC 데이터를 추가한 후의 비교 결과이다. 낸드 메모리의 1번

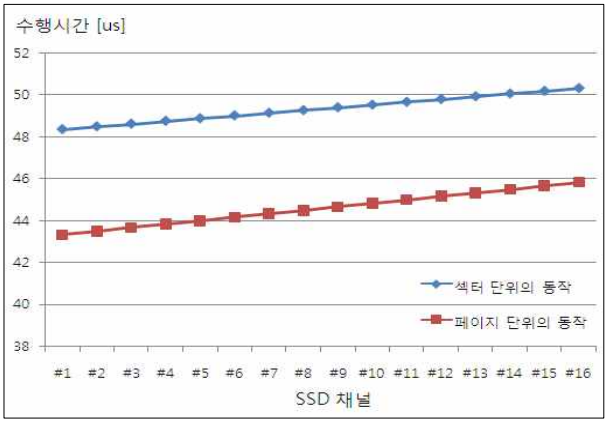


그림 9. 섹터와 페이지에 따른 FEC 패리티/CRC 성능 비교

Fig. 9. Comparison in performance of FEC parity/CRC according to sector and page.

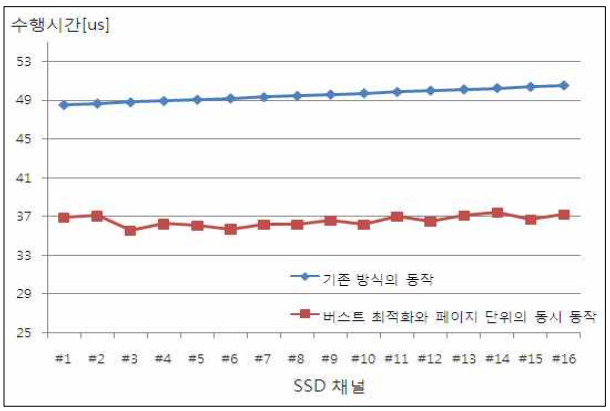


그림 10. 기존 방식과 버스트 크기 최적화와 페이지에 따른 FEC 패리티/CRC 성능 비교

Fig. 10. Comparison in performance of conventional method and burst size optimization and FEC parity/CRC according to page.

채널부터 16번 채널까지 한 페이지를 낸드에 쓰기와 읽기 전송 후의 평균시간을 측정하였다. 최대 시간이 소요된 16번 채널의 데이터를 비교해 본 결과 50.3us에서 45.82us 로 9% 의 성능향상을 확인할 수 있었다.

그림 10은 버스트 크기 최적화와 페이지 단위의 FEC 패리티/CRC 동작을 동시에 적용시킨 경우이다. 메타데이터는 평균적으로 DRAM을 한번 읽고 한번 쓰는 것으로 가정하였고, 16채널 모두 한 페이지를 낸드에 쓰기와 읽기 수행 후 평균시간을 측정하였다. 적용 전에는 16번 채널, 적용 후에는 2번 채널에서 수행시간이 오래 소요되었고, 이 두 채널을 비교해 본 결과 50.5us 에서 37.4us로 25.9% 의 성능향상을 확인 할 수 있었다.

채널당 낸드에 한 페이지를 쓰는 경우 사용자 데이터는 DRAM 읽기 명령이 32번에서 16번으로 감소하였고, FEC 패리티/CRC 데이터는 DRAM 읽기 명령이 8번에서 한번으로 줄어들어 전체적으로 DRAM 인터페이스에서 발생하는 지연시간이 단축되었다.

V. 결 론

본 논문은 SSD의 DRAM 인터페이스 구조를 통해서 전송 데이터의 효율적인 접근 방법을 제시하고 최종적으로 전체 시스템의 성능향상을 위한 데이터 처리 기법을 제안하였다. 사용자와 메타 데이터의 전송시간이 지연되어 대기 시간이 증가하는 경우에 가변적인 버스트 크기를 이용하여 DRAM에 접근하는 대기 시간을 줄임으로써 성능을 향상시킬 수 있었고, FEC 패리티/CRC 데이터를 위해서는 낸드 메모리의 특성을 이용하여 기존의 매 섹터마다 전송이 아닌 페이지 단위로 전송하여 DRAM 인터페이스에서 발생하는 지연시간을 줄일 수 있었다. 제안한 방법을 통해서 성능을 비교해 본 결과 0.05%의 미미한 칩 면적의 증가를 통하여 사용자 데이터는 18%, 메타데이터는 11.8%, FEC 패리티/CRC 데이터에 의한 동작은 9% 의 향상이 이루어져 전체적으로 25.9%의 SSD 성능 개선을 확인할 수 있었다.

참 고 문 헌

[1] N Agrawal, V Prabhakaran, “Design tradeoffs for SSD performance”, USENIX 2008.

[2] SW Lee, “A case for flash memory ssd in enterprise database applications”, ACM SIGMOD 2008.

[3] Chung-Li Yu, Ho-Ming Leung, “ECC/CRC error detection and correction system” US Patent, no. 5,027,357 1991.

[4] C Park, “A reconfigurable FTL(flash translation layer) architecture for NAND flash-based applications” ACM TECS 2008.

[5] SW Lee, “A case for flash memory ssd in enterprise database applications” ACM SIGMOD 2008.

[6] http://www.samsung.com/sec/consumer/it/harddiskdrives/ssd/MZ-5PA256/KR/index.idx?pagetype=prd_detail&returnurl=

[7] James J, Allen W, “Hard Disk Controller: the Disk Drive's Brain and Body”, Computer Design 2001 262~267

[8] http://h20000.www2.hp.com/bc/docs/support/SupportManual/c01580706/c01580706.pdf

[9] Flash memory K9GBGD8U0A
http://www.samsung.com/global/business/semiconductor/products/flash/Products_Toggle_DDR_NANDFlash.html
[10] DDR2 SDRAM K4T1G044QF data sheet
http://www.samsung.com/global/system/business/semiconductor/product/2010/10/5/537329ds_k4t1gxx4qf_rev111.pdf

저 자 소 개

임 광 석(학생회원)
2000년 2월 건국대학교 전자공학과 학사 졸업.
2000년 3월～현재 삼성전자 책임연구원.
2010년 3월～현재 성균관대학교 반도체디스플레이공학과 석사과정.
<주관심분야 : SOC 설계, 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>