# SSD 아키텍처 탐구를 위한 PCIe 기반의 Open Source SSD 플랫폼 개발 및 성능 모니터링 방법

한양대학교 ▌정상혁 · 김영남 · 허태영 · 송용호*

## 1. 서 론

플래시 메모리 기술의 지속적인 발전에 의해 solid state drive(SSD)와 같은 플래시 메모리 기반 저장장치의 사용이 끊임없이 증가하고 있다. 플래시 메모리는 작고 싼 가격으로 고성능 대용량 저장장치를 구성할 수 있으므로, 모바일을 포함한 대다수의 임베디드 시스템 및 클라이언트/서버 시스템 등에서 다양하게 사용이 가능하다[1]. 하지만 플래시 메모리는 낸드 플래시 셀의 기본 구조에 의해 파생되는 불합리한 단점이 존재하기 때문에 이를 제거하기 위한 노력이 필요하다.

플래시 메모리 디바이스[2]는 저장 셀의 구조나 특성에 따라 혹은 error correction code(ECC)[3]의 포함 여부에 따라 다양하게 구분되며, 실제 응용 제품에 사용되기 위해서는 응용 제품의 특성에 맞게 구현된 저장장치 제어기의 지원이 필수적이다. 모바일 제품에 추가되는 플래시 저장장치의 제어 방식과 고성능 서버 시스템에 장착되는 플래시 저장장치의 동작 방식은 완전히 다르기 때문이다. 즉, 플래시 메모리의 제어장치는 플래시 디바이스의 구조적인 단점을 최대한 상쇄시키면서도 타깃 응용제품의 요구사항에 적합한 형태로 개발되어야 한다.

이와 같이, 다양한 시스템에서 널리 사용되고 있는 플래시 저장장치는 그 사용처에 따라 시스템의 구조와 성능 및 내구성 요구 조건이 다양할 수 있다. 따라서 SSD의 구조와 펌웨어는 타깃 시스템에 따라 다양하게 개발되어야 하며, time-to-market을 위해서는 신속한 SSD 아키텍처 exploration 및 알고리즘 검증이 필요하다[4]. 각 응용제품의 요구 조건과 최적의 성능 발휘를 위해 호스트 시스템에 적합한 저장장치의 내부 구조 및 알고리즘을 결정하는 작업은 모든 제품 개발에 앞서 선행되어야 하는 가장 주요한 작업 중 하나이다.

SSD 아키텍처 exploration 및 알고리즘 검증은 다양한 형태로 수행할 수 있다. Software 기반의 시뮬레이터들은 SSD 아키텍처와 구성 모듈을 상위 레벨의 프로그래밍 언어로 모델링하여, 그 동작을 구조적으로 검증한다. 이러한 방식은 구성 모듈을 빠르게 모델링할 수 있고 다양한 구조적 변경이 가능하여 time-to-market 요구를 충족시킬 수 있다. 하지만 실제의 복잡한 시스템 모델링에는 한계가 있어 정확성이 낮으며, 상위 레벨 언어의 시뮬레이션 수행 속도가 느려 다양한 구조적 검증에 문제가 될 수 있다. 반면, hardware 기반의 테스트 플랫폼들은 실제 저장장치 시스템 플랫폼을 제작하고 각 모듈을 개발하여 동작시킬 수 있어 실험 정확성 측면에서는 유리하지만, 다양하게 플랫폼 모듈을 변경하는 구성이 어렵고 개발비 및 개발기간이 증가하기 때문에 time-to-market에 불리하다.

따라서 본 논문에서 우리는 open source 기반의 SSD 플랫폼을 개발하고 실험 데이터를 빠르고 정확하게 추출할 수 있는 방법을 제안하여, 신속하고 효과적인 SSD 시스템 exploration이 가능하도록 한다. 호스트 인터페이스는 PCIe를 기반으로 하여 호스트와 SSD의 통신에서 성능 상의 병목현상이 없도록 최적화하고, SSD 플랫폼의 모든 모듈은 open source, 혹은 자체 제작한 제어기를 사용하여 추가비용 없이 빠르게 재사용 가능하도록 구성한다.

## 2. 관련 지식

SSD 플랫폼을 제작하여 시스템 exploration을 하기 위한 몇 가지 연구들이 수행되었다. 저장장치 시스템

* 종신회원

† 본 연구는 미래창조과학부 및 정보통신산업진흥원의 대학 IT연구센터 육성지원 사업의 연구결과로 수행되었음. (NIPA-2014-H0301-14-1018)

플랫폼의 exploration에 있어서 크게 두 가지 분류로 구분하여 구현될 수 있다. 편의상 software로 제작된 SSD 모델을 SW 시뮬레이터라 지정하고, hardware 제어기를 구현한 모형을 HW 테스트 플랫폼이라 정의하도록 한다. 두 가지 구현 모델은 각 장단점이 존재하므로 그 특성과 개발 실효성에 대해 숙지하여야 한다.

### 2.1 SW 시뮬레이터

SSD 플랫폼의 구조 및 내부 알고리즘의 동작 성능을 확인하기 위한 가장 간편한 방법으로 SW 시뮬레이터들이 개발되었다. SW 시뮬레이터들은 SSD시스템을 모델링하기 위해 호스트 인터페이스, 플래시 메모리 제어기, 버스 및 각종 메모리 디바이스를 상위 레벨 프로그래밍 언어로 모두 모델링한다. 필요한 경우 적절한 사용자 워크로드를 삽입하거나 혹은 자동 워크로드 생성기를 사용하여 시뮬레이터가 사용할 가상의 호스트 명령 및 데이터를 제공한다. 시뮬레이터의 수행 결과는 파일 형태로 저장되어 이후 시스템 성능의 변동을 가공하는 resource로 사용한다. 시뮬레이션 수행 결과의 정확성은 당연히 각 동작 모듈의 모델링 정확성에 의존하며, 상위 레벨 프로그래밍 언어로 개발되었기 때문에 시뮬레이션 수행 속도가 느려질 수 있다.

SSD 구조를 모델링하는 SW 시뮬레이터로는 대표적으로 SSDSim[5]과 QEMU-KVM으로 구현한 Sim[6] 등이 있다. SSDSim은 순수하게 trace driven으로 시뮬레이션을 수행하며 버스, 낸드 등 주요한 모듈들이 상위 레벨 언어로 모델링되어 있다. 각 모듈 구성의 변경은 용이하게 가능하지만 호스트 시스템에서 생성되는 워크로드와 데이터를 실시간으로 직접 사용할 수 없으며 시뮬레이션 결과의 정확성이 낮다는 측면에서 한계가 존재한다. QEMU-KVM으로 구현한 Sim은 가상머신인 QEMU의 저장장치 드라이버를 수정하여 SSD를 모델링한다. HW 테스트 플랫폼과 같은 실제 시스템 장치 단위의 정확성을 기대할 수는 없지만 가상머신에 포팅된 운영체제의 워크로드를 직접 받아서 사용하므로 저장장치의 동작 구현을 정확히 가상화할 수 있다는 장점이 있다.

### 2.2 HW 테스트 플랫폼

HW 테스트 플랫폼은 SSD 구현에 필수적인 모든 component를 집약한 SoC를 사용하고, 특정 디바이스에 적합하게 개발된 플래시 메모리 제어기를 사용한다. 한정된 플랫폼에서의 동작으로 제한된 시스템 제어기를 사용하기 때문에 하드웨어 구성 및 구조의 변경

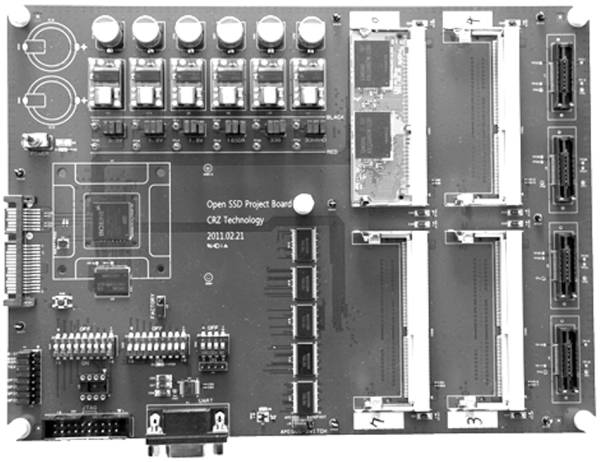


그림 1 Jasmine Open SSD Platform with Indilinx Barefoot Controller SoC

에는 한계가 있으며, 내부 펌웨어 및 소프트웨어 알고리즘을 다양하게 변경하는 실험을 수행할 수 있다. 이 방식은 테스트 플랫폼이 한번 개발되면 실제 시스템에서 다양한 소프트웨어 변경을 통해 매우 정확한 실험 결과치를 추출해 낼 수 있다. 가상의 워크로드가 아닌 실제 OS가 포팅된 호스트의 저장장치로 인식시킨 후 실험을 수행하기 때문에 실제 사용자 워크로드와 데이터를 정확하게 분석 가능하다.

한 예로 그림 1의 Jasmine OpenSSD 플랫폼[7]은 SATA 기반 인터페이스를 사용하며, 실제 호스트 시스템에 연결하여 real workload를 이용한 성능 분석이 가능하다. 플랫폼 보드 위에 추가 장착되는 낸드 디바이스 모듈의 개수에 따라 채널/웨이의 수는 한정적으로 변경이 가능하지만, Indilinx Barefoot Controller SoC[8]를 장착하여 낸드 제어기 및 하드웨어 아키텍처의 구조 변경은 근본적으로 불가능하다. 다만, 펌웨어를 다양하게 수정할 수 있으므로 ARM 프로세서 상에서 동작하는 소프트웨어 알고리즘을 변경해가며 성능 및 내구성 관점에서 검증이 가능하다. 결과적으로 Jasmine OpenSSD 플랫폼은 고정된 하드웨어에 의존한 시스템 구성을 유지하기 때문에, 저장장치의 하드웨어 구조 exploration을 수행하는데 부족함이 존재한다.

## 3. Open Source SSD Platform 개발 및 성능 모니터링

### 3.1 Open Source SSD Platform Board

본 논문에서 제시하는 PCIe 기반 Open Source SSD 플랫폼의 모형은 그림 2와 같으며, 그 제어기의 구조는 그림 3과 같다.

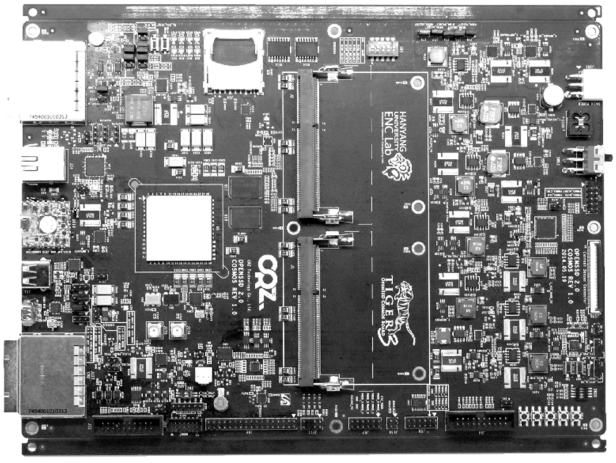

그림 2 PCIe 기반 Open Source SSD Platform 보드

이 플랫폼에는 Xilinx FPGA가 탑재되어 SSD의 동작을 정의하는 하드웨어 모듈 제어기를 자유롭게 구성할 수 있다. 탑재된 FPGA는 xilinx zynq[9]모델로서 ARM dual core와 DRAM 제어기와 같은 필수적인 하드웨어 모듈 IP가 내장되어 있으며 ARM compiler를 통해 편리한 펌웨어 개발이 가능하다. 또한 PCIe 인터페이스를 사용하기 위해서는 Xilinx coregen을 통해 phy 모듈을 생성하고 자체 제작한 wrapper 모듈을 이용하여 호스트와 통신할 수 있도록 구성할 수 있다. 기본적으로 제공되는 IP 이외에 SSD 제어를 위한 낸드 플래시 메모리 제어기, 채널/웨이 구성을 위한 아키텍처 및 DMA, Arbiter 등과 버스 프로토콜 관리 모듈을 모두 제작하여 연결한다. SSD 하드웨어의 내부 모듈 수정을 원한다면 플랫폼의 동작 주파수 변경은 물론이고, 하드웨어 Accelerator, Async/Sync 낸드 모드 변경, 채널/웨이 구조, 버스 및 내부 ECC까지 변경이 가능하다.

또한 SSD 하드웨어 플랫폼에 적합한 펌웨어 및 FTL[10,11]을 적절히 구현하여 최적화된 형태의 소프트웨어 지원으로 SSD 시스템 구축 및 실험이 가능하다. 기존의 HW 테스트 플랫폼에서는 펌웨어 및 FTL의 개발에 있어 정해진 하드웨어 환경에 국한되어 제한적으로 개발을 수행해야 한다. 예를 들어, 하드웨어 제어기의 채널/웨이 Arbiter가 입력 데이터를 static 방식으로 균등하게 분할하여 각 디바이스에 나눠준다면, FTL은 Dynamic Map 할당 방식을 아예 사용할 수도 없다. 하드웨어 플랫폼이 호스트 인터페이스를 통해 데이터를 받아들이고 플래시 메모리에 저장하는 모든 순서는 이미 지정된 sequence에 의해 진행되므로 펌웨어와 FTL 알고리즘 개발에도 한계가 있을 수 있다.

제안하는 Open Source SSD 플랫폼은 PCIe 2.x 4-Lane 기반의 인터페이스(2GB/s)[12]를 사용한다. 이러한 고성능 인터페이스를 사용하는 이유는 SSD와 호스트 사이에서 발생 가능한 병목현상을 제거하기 위한 것이다. 같은 이유로 SATA2 [13]정도의 호스트 인터페이스를 사용한다면 데이터 병목현상으로 인해 SSD 내부에서 처리 가능한 최고 성능을 검증할 수 없다. 특정 환경, 혹은 특정 하드웨어 장비에서만 동작되는 SSD가 아니라 많은 세부 모듈에서의 성능 측정치를 정확히 추출해야하는 플랫폼의 개발이 목적이기 때문에, 최적의 성능을 발휘하는 호스트 인터페이스 구현은 필수적이다.

SSD 플랫폼 하드웨어의 각 동작 모듈은 Verilog RTL로 코딩하고 EDK[14]툴에서 모듈을 연결하여 구현한다. Xilinx에서 제공하는 툴을 사용하여 직관적이고 효율적으로 개발을 수행할 수 있다. SSD 플랫폼 소프트웨어는 SDK를 사용하여 C언어로 개발한다. ARM dual core 프로세서가 내장되어 있으므로 arm code compiler가 내장된 SDK를 통해 편리하게 펌웨어 및 FTL 제작이 가능하다.

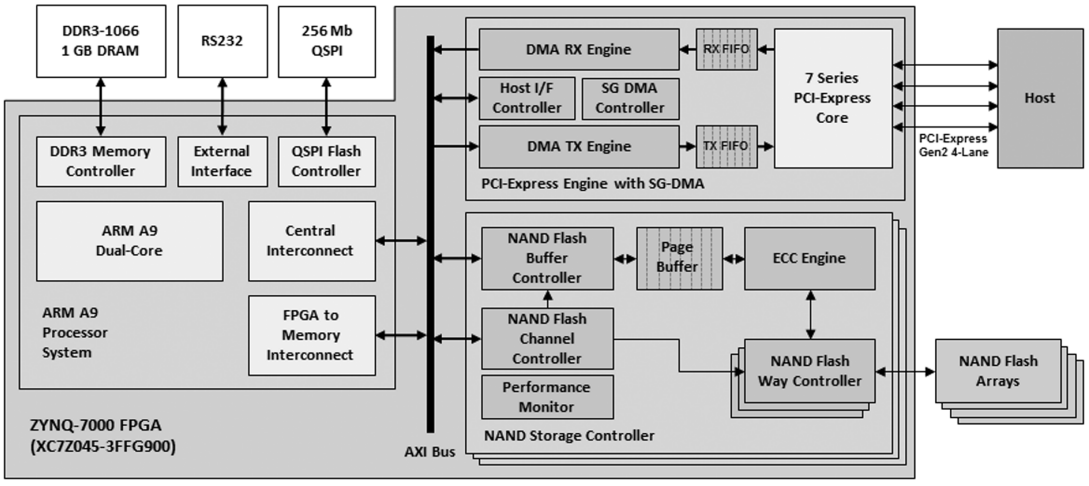


그림 3 Open Source SSD 플랫폼 제어기 구조

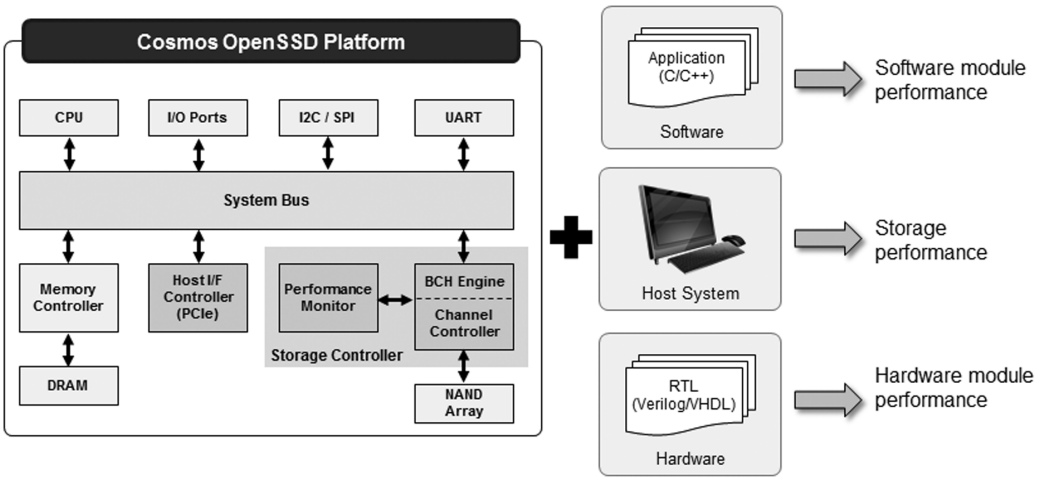


그림 4 Open Source SSD 플랫폼 Performance Monitor를 위한 모듈 구성

Open Source SSD 플랫폼을 정확하게 구현하여 사용하더라도 제작 의도에 맞는 정확한 실시간 성능 및 내구성 측정이 가능해야 한다. 실제 사용되는 SSD와는 달리 아키텍처 exploration을 목적으로 하므로, 어느 구조와 어느 알고리즘에서 최적의 성능이 발휘되는지 파악하기 위해서는 저장장치의 성능 및 내구성을 하드웨어 기반 모듈에서 직접적으로 추출하는 방식을 사용하여야 한다. 플랫폼에서 추출한 데이터를 가공하고 외부 인터페이스로 전송하는데 플랫폼 내부의 자원을 크게 소모하게 된다면 성능을 모니터링 하는 자체가 저장장치 성능의 병목이 될 수 있다. 또한 실시간으로 SSD 플랫폼의 성능을 모니터링 할 수 있다면 특정 time domain에서 성능 저하 요소를 찾아내는 검사를 효과적으로 수행할 수 있을 것이다.

### 3.2 SSD Platform Performance Monitoring

하드웨어 플랫폼과 소프트웨어 개발 키트까지 갖추어진 상태이기 때문에, 원하는 데이터를 어느 위치에서든 손쉽게 추출할 수 있는 솔루션을 구성할 수 있다. CPU나 BUS의 지연시간만을 따로 추출하여 가공할 수 있으며 가비지 컬렉션[15,16]등의 특정 알고리즘 동작 중에 누적된 지연만을 계산하여 표시할 수 있다. 즉, SSD 플랫폼에서 발생하는 모든 지연을 break down하여 어느 지점에서 병목이 발생하는지를 정확히, 그리고 용이하게 추출이 가능하다.

그림 4는 Open Source SSD 플랫폼에서 소프트웨어/하드웨어 모듈의 성능 모니터링을 위한 구성을 보인다. 또한 그림 5와 6은 성능 모니터링 모듈의 내부 구현 구조와 데이터 추출에 필수적인 레지스터 집합을 나타낸다. 원하는 위치의 하드웨어 모듈에서 추출한 성능 정보 데이터는 UART를 통해 호스트로 전달된다. 추출한 모든 데이터를 호스트에 전달하는 것은 SSD 내부 동작 성능에 영향을 미치므로 평균값이나 표준편차를 누적하여 가공한 데이터를 송부하는 방식을 사용한다. 이러한 연산은 보통 그림 5의 Performance Monitor라는 하드웨어 모듈을 통해 이루어지며, 필요한 경우에는 CPU를 통해 연산된 값을 호스트로 전달한다. 데이터 가공에 필요한 모든 연산을 하드웨어 모듈에서만 수행할 수는 없으므로 추가적인 CPU 연산이 필요할 수 있다. 하지만 CPU의 연산지연과 SSD 플랫폼의 UART 통신은 큰 지연을 발생할 수 있기 때문에, 이러한 영향을 상쇄하기 위한 추가적인 CPU를 사용한다. SSD 플랫폼에 탑재된 Zynq FPGA는 ARM dual core 프로세서를 지원하므로 사용하지 않는 core를 성능 모니터링 용도로 할당하여 데이터 추출 동작에만 사용한다. 현재 플랫폼은 single core 동작만을 정의하며 이 후 dual core를 펌웨어 및 알고리즘 처리에 사용한다면 수정이 필요할 것이다.

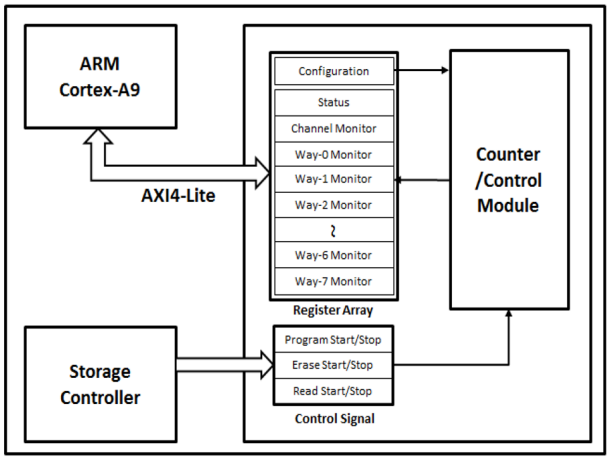


그림 5 성능 및 내구성 모니터링 모듈 내부 구조

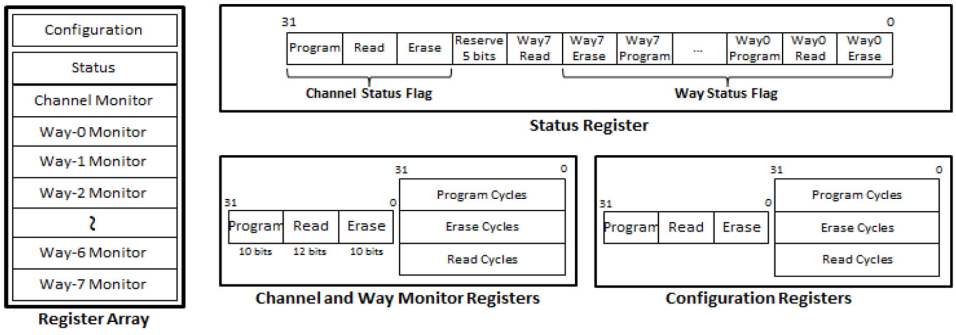


그림 6 성능 및 내구성 관련 데이터 추출을 위한 모니터링 레지스터

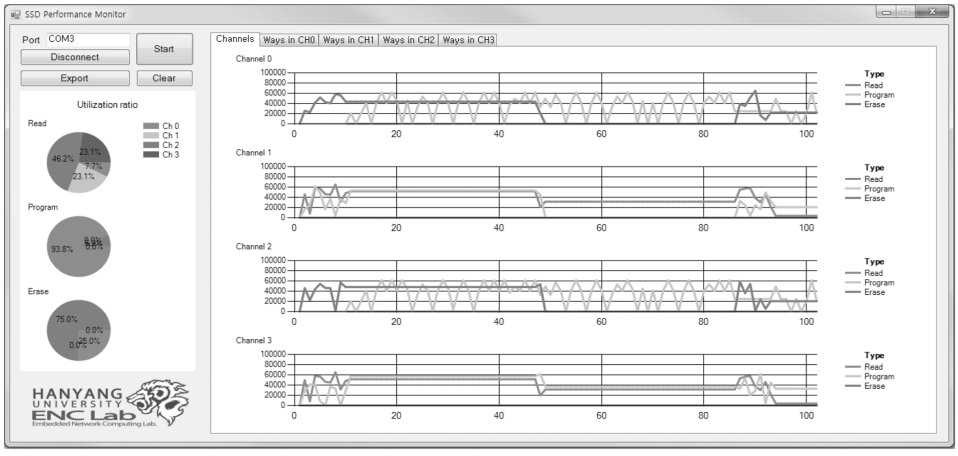


그림 7 실시간 SSD 플랫폼 채널 Utilization 및 R/W/E 동작 비율

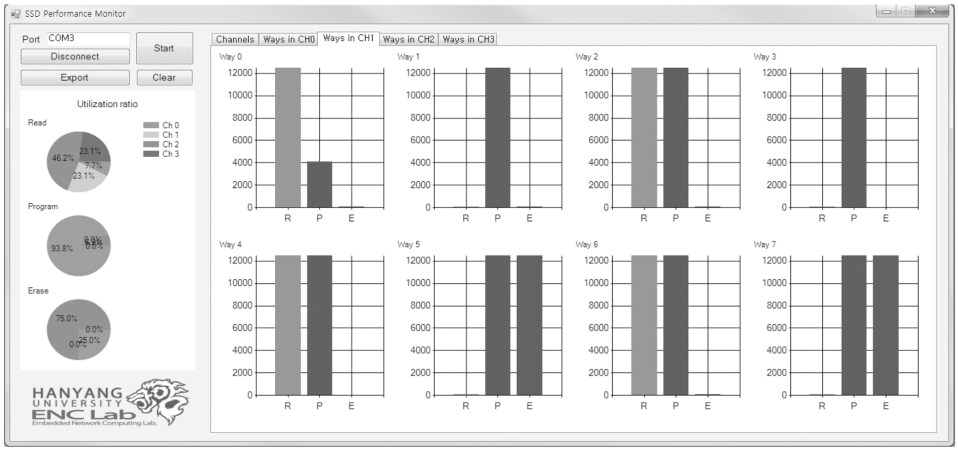


그림 8 실시간 SSD 플랫폼 웨이 Utilization 및 R/W/E 동작 비율

그림 7과 8에서 확인할 수 있듯이, 호스트 시스템으로 전달된 가공된 데이터는 즉각적으로 성능을 보여주는 Graphic 환경에서 표시된다. 현재 SSD 플랫폼 시스템이 수행 중인 작업에 대해 처리 속도를 bandwidth 형태로 보이며, 각 알고리즘에서 누적되는 수행 횟수를 파이 다이어그램으로 표현한다. 또한 호스트에 전달된 모든 데이터는 데이터 가공에 재사용된다.

## 4. Performance Monitor를 이용한 Open Source SSD 실험 결과

다음은 Performance Monitor를 이용하여 Open Source SSD 플랫폼 실험 결과를 보인다. 실시간으로 추출하여 입력 데이터 패턴에 따라 변화하는 데이터 처리 성능을 항목별로 보이며, 실시간 데이터를 누적하여

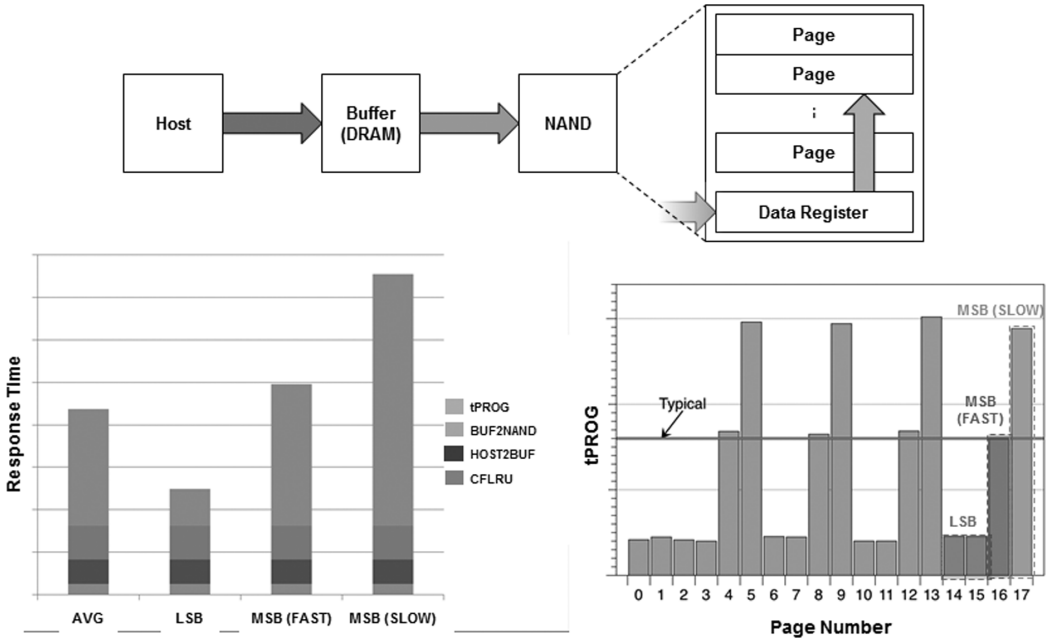


그림 9 동작 수행 시간 Break-Down

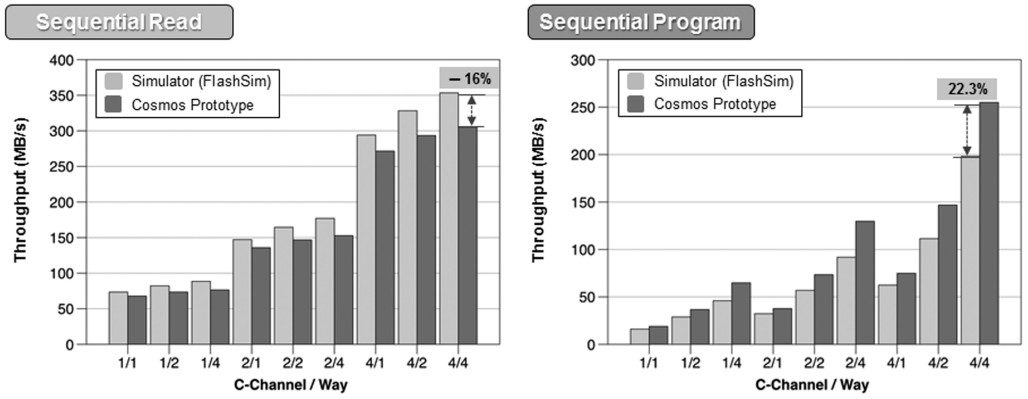


그림 10 아키텍처 Exploration 결과 모형

elapsed 성능도 표현한다. Performance Monitor를 이용한 Open Source SSD 플랫폼은 실제 SSD 시스템의 데이터 추출에서부터 지정된 모듈의 세부 동작 횟수까지 즉각적으로 손쉽게 유도가 가능하다.

그림 9는 호스트에서 낸드 플래시 디바이스까지 데이터의 전송과 각 모듈의 연산에 따라 처리되는 지연시간을 breakdown하여 보인다. 버퍼 관리 알고리즘은 CFLRU[17]를 사용하였다. 호스트에서 전달된 데이터가 디바이스에 저장되기까지 각 연산지연을 전체 수행시간과 비교하여 보이며, MSB 프로그램의 속도가 uniform하지 않음을 확인할 수 있다. 그림 10은 SW 시뮬레이터[5,6] 등이 실제 워크로드를 사용한 실험에서 그 측정 결과가 얼마나 부정확한지를 나타낸다. 제안하는 Open Source SSD 플랫폼은 실제 SSD에서 발생할 수 있는 세세한 타이밍 지연 등을 모두 반영하므로 일반적인 시뮬레이션과 달리 매우 정확한 성능 예측이 가능하다. 그림에서 확인할 수 있듯이, 다양한 채널 및 웨이 개수를 변경하며 실험해 보았을 때, 시뮬레이션과 제안하는 SSD 플랫폼 사이에는 throughput 측면에서 16%~22.3% 정도의 커다란 차이가 발생함을 확인할 수 있었다. 즉, SSD 시뮬레이션에서 각 모듈을 정확히 모델링하지 못한다면 아키텍처 exploration이 불가능할 정도의 예측 오차가 발생할 수 있다는 것이다.

## 5. 결 론

본 논문에서 우리는 Open Source SSD 플랫폼을 활용하여 SSD 개발에 필수적인 아키텍처 exploration과

성능 모니터링 방법을 제안하였다. 개발된 SSD 플랫폼을 통해서, 성능 관련 데이터의 실시간 추출과 가공이 용이하며 time-to-market에 효과적이고 하드웨어 모듈 변경에 유연한 SSD 개발이 가능하다는 것을 확인할 수 있었다. 우리는 Open Source SSD 플랫폼이 최근 사용량이 계속 증가하고 있는 플래시 메모리 기반 저장장치에서 그 용도와 구조 및 성능의 검증을 용이하게 할 수 있는 방안으로 기여되기를 기대한다.

## 참고문헌


[ 1 ] G. Lawton, "Improved flash memory grows in popularity", IEEE Comput., vol. 39, no. 1, pp. 16-18, Jan. 2006.

[ 2 ] A. Leventhal, "Flash storage memory", Commun. ACM, vol. 51, no. 7, pp. 47 – 51, Jul. 2008.

[ 3 ] S. Jung, S. Lee, H. Jung, and Y. H. Song, "In-page management of error correction code for MLC flash storages", in Proc. 54th IEEE Int. Conf. Midwest Symp. Circuits Syst., Aug. 2011.

[ 4 ] H. Jung, S. Jung, and Y. H. Song, "Architecture exploration of flash memory storage controller through a cycle accurate profiling", IEEE Trans. Consum. Electr., vol. 57, no. 4, pp. 1756-1764, Nov. 2011.

[ 5 ] Y. Hu, H. Jiang, D. Feng, L. Tian, H. Luo, and S. Zhang, "Performance impact and interplay of SSD parallelism through advanced commands, allocation strategy and data granularity", ACM Conference on ICS, Jun. 2011.

[ 6 ] J. Yoo, Y. Won, J. Hwang, S. Kang, J. Choi, S. Yoon, J. Cha, "VSSIM: Virtual machine based SSD simulator", IEEE Symposium on MSST, May 2013.

[ 7 ] OpenSSD-Project. Indilinx Jasmine Platform Specification. http://www.openssd-project.org/wiki/The_OpenSSD_Project.

[ 8 ] http://en.wikipedia.org/wiki/Indilinx.

[ 9 ] http://www.xilinx.com/products/silicon-devices/soc/zynq-7000.html.

[10] J. Kim, J. M. Kim, S. H. Noh, S. L. Min, and Y. Cho, "A space-efficient Flash Translation Layer for compact flash systems", IEEE Trans. Consum. Electr., vol. 48, no. 2, pp. 366-375, May 2002.

[11] S.-W. Lee, D.-J. Park, T.-S. Chung, D.-H. Lee, S.-W. Park, and H.-J. Song, "A log buffer-based flash translation layer using fully-associative sector translation", ACM Trans. Embed. Comput. Syst., vol. 6, no. 3, article 18, Jul. 2007.

[12] http://en.wikipedia.org/wiki/PCI_Express.

[13] ATA/ATAPI Command Set – 2 (2009, Aug.). T13/2015-D, Revision 2. http://www.t13.org.

[14] http://www.xilinx.com/tools/platform.htm.

[15] L.-P. Chang, T.-W. Kuo, and S.-W. Lo, "Real-time garbage collection for flash-memory storage systems of real-time embedded systems", ACM Trans. Embed. Comp. Syst., vol. 3, no. 4, pp. 837-863, Nov. 2004.

[16] J. Lee, Y. Kim, G. M. Shipman, S. Oral, and J. Kim, "Preemptible I/O scheduling of garbage collection for solid state drives", IEEE Trans. Comput.-Aid. Design Integrated Circuits Syst., vol. 32, no. 2, pp. 247-260, Feb. 2013.

[17] S.-Y. Park, D. Jung, J.-U. Kang, J.-S. Kim, and J. Lee, "CFLRU: A replacement algorithm for flash memory", CASES'06, Oct. 2006.


## 약 력


### 정 상 혁

2008 한양대학교 미디어통신공학과 졸업(학사)
2010 한양대학교 전자컴퓨터통신공학과 졸업(석사)
2014 한양대학교 전자컴퓨터통신공학과 졸업(박사)
2014～현재 한양대학교 융합전자공학부 연구교수
관심분야 : 시스템 아키텍처, 플래시 메모리 기반 응용 저장장치, 차세대 메모리
Email : shjung@enc.hanyang.ac.kr



### 김 영 남

2011년 강원대학교 전자공학과 졸업(학사)
2011년～현재 한양대학교 전자컴퓨터통신공학과 재학(석사)
관심분야 : 시스템 아키텍처, 플래시 메모리, MPEG
Email : ynkim@enc.hanyang.ac.kr



### 허 태 영

2012년 한국해양대학교 전파공학과 졸업(학사)
2014년 한양대학교 전자컴퓨터통신공학과 졸업(석사)
2014년～현재 한양대학교 MSRC 멀티미디어 리서치 센터 연구원
관심분야 : 시스템 아키텍처, 플래시 메모리, 차세대 메모리
Email : tyhuh@enc.hanyang.ac.kr



### 송 용 호

1989년 서울대학교 컴퓨터공학과 졸업(학사)
1991년 서울대학교 컴퓨터공학과 졸업(석사)
1991년～1996년 삼성전자 컴퓨터 시스템, 연구원
2002년 미국 Univ. of Southern California졸업 (박사)
2003년～현재 한양대학교 융합전자공학부 부교수
2011년～현재 MSRC 멀티미디어 리서치 센터 센터장
관심분야 : 컴퓨터 구조, 플래시 메모리, 차세대 메모리, 빅데이터, NoC, MPEG
Email : yhsong@hanyang.ac.kr