EISC 프로세서를 이용한 NVMe Device 설계

김영근, 송용호
한양대학교 전자컴퓨터통신공학부
ykkim@enc.hanyang.ac.kr, yhsong@hanyang.ac.kr

Design of NVMe Device Using EISC Processor

Youngkuen Kim, Yong Ho Song
Department of Electronics and Computer, Hanyang University, Seoul, Korea

요 약

최근 PCIe기반 SSD에서 사용하는 NVMe 인터페이스에 대한 연구가 활발히 진행되고 있다. 기존 NVMe기반 SSD는 내부동작을 정확하게 알기 어렵고 개발목적으로 사용하는데 어려움이 있다. 본 논문에서는 EISC프로세서 소프트코어와 오픈소스인 OpenSSD 플랫폼을 이용한 NVMe 디바이스 시스템을 설계함으로써, 내부동작을 파악할 수 있고 환경설정을 통해 최적화가 가능하다. PCIe를 지원하는 FPGA board에서 성능을 검증한 결과, 랜덤 4KB 읽기 I/O의 경우 평균 4356 IOPS, 랜덤 4KB 쓰기의 경우 평균 4395 IOPS의 성능을 보였다. 또한 시퀀셜 128KB 읽기 I/O의 경우 평균 106.5 MB/s, 시퀀셜 128KB 쓰기 I/O의 경우 평균 106.1 MB/s의 성능을 보였다.

Ⅰ. 서 론

비휘발성 메모리는 빠른 응답속도, 낮은 전력 소비 등의 장점을 가지고 있어 많은 컴퓨팅 시스템에서 하드디스크를 대체하는 저장장치로서 사용되고 있다. 많은 비휘발성 메모리 저장장치는 대역폭을 향상시키기 위해 비휘발성 메모리를 병렬적으로 접근한다. 멀티채널, 웨이 구조에 의해 비휘발성 메모리 저장장치의 대역폭이 향상됨에 따라 하드디스크 기반의 저장장치에서 사용하던 호스트 인터페이스의 대역폭이 비휘발성 메모리 저장장치의 대역폭을 제한하게 되었다. 따라서 PCI Express (PCIe) 인터페이스가 비휘발성 메모리 저장장치의 호스트 인터페이스로서 사용되게 되었다. 또한 PCIe 기반 SSD를 위해서 새로운 NVM Express (NVMe) 인터페이스가 고안되었다.[1]

NVMe 인터페이스는 최적화된 레지스터, 명령어 집합 등을 정의하며 하나의 I/O큐를 가지고 동작하던 기존 인터페이스와는 다르게 다중 큐 구조를 가지고 어플리케이션 마다 개별적인 큐를 사용가능하게 해서 I/O 병렬성을 높였다. 또한 NVMe SSD에서 발생한 단일 인터럽트로 다중 큐/다중 명령에 대한 완료 처리가 가능하도록 지원한다[2].

본 논문에서는 Extendable Instruction Set Computer (EISC)기반 프로세서를 이용한 NVMe 디바이스 시스템을 제안한다. EISC는 ADchips에서 개발한 Instruction Set Architecture (ISA)로서 확장 레지스터와 확장 플래그를 통해 피연산자 길이를 필요한 만큼 임의로 확장이 가능하고, 길이가 16비트로 고정된 명령어 구조를 갖는다. 또한 간단한 명령어 셋을 가짐으로써 크기가 작고, 저전력, 고성능의 특징을 갖기 때문에 임베디드 시스템에서 사용하기 적합하다[3].

Ⅱ. 본 론

그림 1은 제안하는 NVMe 디바이스 시스템의 구조를 보여준다. NVMe 디바이스 시스템은 MCU, NVMe host controller, DRAM controller, DRAM을 포함한다. NVMe 호스트 컨트롤러는 호스트 시스템의 메인 메모리에 존재하는 호스트 명령어를 가져온다. 또한 PCIe 인터페이스를 통해 NVMe 디바이스 시스템과 호스트 시스템 간의 데이터전송을 수행한다. MCU는 NVMe 명령어를 처리하기 위한 NVMe 펌웨어를 동작시킨다. NVMe 펌웨어는 NVMe 명령어를 처리하기 위해 submission 큐, command 큐 등을 생성하거나 제거하며 NVMe 디바이스의 정보를 호스트 시스템에게 제공하여 호스트 시스템이 NVMe 디바이스를 인식하도록 한다. 각 구성요소는 AXI 버스를 통해 통신한다.

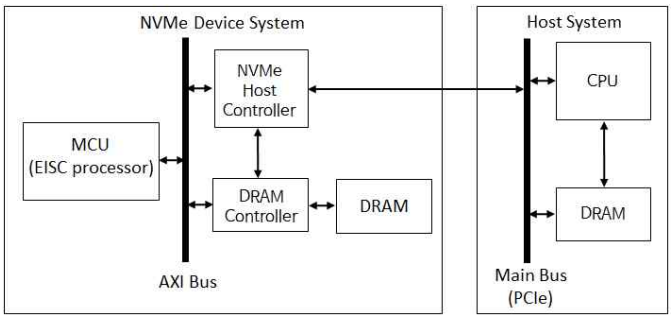

그림 1. EISC processor with NVMe host controller

그림 2는 EISC processor의 구조를 보여준다. JTAG 디버거는 JTAG 호스트에서 파일 I/O 명령어를 이용해서 시스템버스를 통해서 메모리에 접근하거나, 프로세서 제어하는 명령어를 통해서 프로그램 디버깅을 수행할 수 있도록 해준다. BIU는 EISC 자체 프로토콜을 AXI 프로토콜로 변환해주는 역할을 한다. 또한 interrupt controller를 통해 시스템 버스에 연결되어 있는 NVMe 호스트 컨트롤러가 발생시키는 인터럽트를 처리할 수 있다. 그림 3은 NVMe 디바이스 시스템이 호스트에서 발생한 명령어를 처리하는 과정을 보여준다. 호스트 시스템 내부 DRAM에는 처리해야 할 명령어를 쌓아두는 submission 큐와 명령어 수행완료 명령어를 쌓아두는 completion 큐가 존재한다. NVMe 호스트 컨트롤러는 submission 큐와 completion큐의 상태를 나타내는 doorbell 레지스터를 가지고 있다. 호스트 CPU는 NVMe 디바이스 시스템에서 처리해야할 명령어를 만들고 submission 큐에 저장한다. 그리고 NVMe 호스트 컨트롤러 내부에

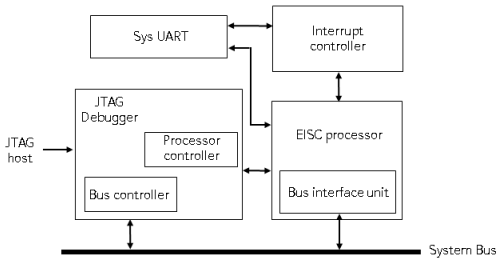

그림 2. EISC processor system architecture

doorbell 레지스터 값을 변경한다. NVMe 펌웨어는 주기적으로 NVMe 호스트 컨트롤러의 doorbell 레지스터 값을 확인한다. 만약 레지스터 값의 변화가 있으면 NVMe 호스트 컨트롤러에게 호스트 시스템 내부 DRAM의 submission 큐에 접근해서 명령어를 NVMe 디바이스 시스템 내부 DRAM에 저장하도록 명령을 내린다. EISC 프로세서에서 동작하고 있는 NVMe 펌웨어는 NVMe 호스트 컨트롤러에서 대기 중인 명령어를 처리한다. 해당 명령어에서 필요한 데이터의 전송이 끝났거나 NVMe 펌웨어에서 해당 명령어를 완료하라고 하면, NVMe 호스트 컨트롤러는 호스트 DRAM의 completion 큐에 completion 명령을 복사하고 호스트로 인터럽트를 보내서 처리 완료를 알린다.

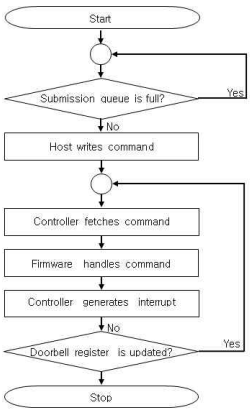

그림 3. Command processing with NVMe device system

Ⅲ. 실 험

제안하는 EISC 프로세서를 이용한 NVMe 디바이스 시스템의 성능을 평가하기 위하여 Xilinx의 ZYNQ-7000 XC7Z045-3FFG900 기반 테스트 보드[5]에 PCIe GEN2 4-Lane을 사용했다. ZYNQ-7000 All Programmable System on Chip (APSoC)내부에 Processing System (PS) 단의 메모리 컨트롤러를 이용해서 1GB DDR3 DRAM을 사용하고[4] Programmable Logic (PL) 영역에 NVMe 호스트 컨트롤러와 EISC기반 Empress 프로세서를 구현했다[5]. Empress 프로세서는 100MHz로 동작하고 캐쉬와 Scratch Pad Memory (SPM)는 사용하지 않았다. NVMe host controller는 200MHz로 동작한다. 표 1 은 합성결과 사용한 리소스를 나타낸다.

표 1. Resource utilization

	Resource	Utilization	Available	Utilization %
EISC	FF	8588	437200	1.96
	LUT	19304	218600	8.83
	Memory LUT	0	70400	0
	BRAM	14	545	2.57
NVMe host controller	FF	11531	437200	2.64
	LUT	8749	218600	4
	Memory LUT	56	70400	0.08
	BRAM	28	545	5.14

표 2 는 제안한 NVMe 디바이스 시스템의 성능을 나타내고 있다. 벤치마크는 IOmeter를 사용했고, 4KB 단위로 데이터를 요청하고 랜덤한 주소에 접근하는 경우와 128KB 단위로 데이터를 요청하고 연속적인 주소에 접근하는 경우에 대해서 실험을 진행하였다[6]. 실험 결과 랜덤 4KB 읽기 I/O의 경우 평균 4356 Input/Output Operations Per Second (IOPS), 랜덤 4KB 쓰기의 경우 평균 4395 IOPS의 성능을 보였다. 또한 시퀀셜 128KB 읽기 I/O의 경우 평균 106.5 MB/s, 시퀀셜 128KB 쓰기 I/O의 경우 평균 106.1 MB/s의 성능을 보였다. 하지만 ARM Coretex-A9 프로세서를 사용하면 랜덤 4KB 읽기, 쓰기 I/O 모두 평균 300K IOPS의 성능을 보이고 시퀀셜 128KB 읽기, 쓰기 I/O의 경우 모두 평균 1.7GB/s의 성능을 보인다[7]. 이러한 성능 차이가 나는 이유는 Empress 프로세서는 100MHz로 ARM Coretex-A9 프로세서는 최대 1GHz로 동작한다. 또한 시스템 구조상 ARM Coretex-A9 프로세서는 바로 DRAM에 접근할 수 있지만, Empress 프로세서는 DRAM에 접근하려면 반드시 AXI버스를 거쳐야하는 오버헤드가 발생한다.

표 2. RAM disk performance

Workload	Read	Write
Random 4KB	4356 IOPS	4395 IOPS
Sequential 128KB	106.5 MB/s	106.1 MB/s

Ⅳ. 결 론

본 논문에서는 EISC 프로세서를 이용하여 펌웨어를 구동시키는 NVMe 디바이스 시스템을 제안하였다. 실험 결과 제안하는 구조는 랜덤 4KB 읽기 I/O의 경우 평균 4356 IOPS, 랜덤 4KB 쓰기의 경우 평균 4395 IOPS의 성능을 보였다. 또한 시퀀셜 128KB 읽기 I/O의 경우 평균 106.5 MB/s, 시퀀셜 128KB 쓰기 I/O의 경우 평균 106.1 MB/s의 성능을 보였다.

ACKNOWLEDGMENT

This work was supported by the IT R&D program of Ministry of Trade, Industry and Energy (10049192, Development of a Smart Automotive ADAS SW-SoC for a Self-Driving Car)

참 고 문 헌

[1] Huffman, A. and Juenemann, D., "The Nonvolatile Memory Transformation of Client Storage," Computer, 46(8):38 - 44, August 2013.

[2] NVM Express Revision 1.2.1. (2016). [online]. Available: http://www.nvmexpress.org/wp-content/uploads/NVM_Express_1_2_1_Gold_20160603.pdf

[3] 이희, "고성능 국산 임베디드 마이크로프로세서(EISC)", 한국통신학회지(정보와통신) 제23권 제5호, pp.28-44, 5월, 2006.

[4] Xilinx, "UG585 Zynq-7000 All Programmable SoC Technical Reference Manual (v1.11)," September, 2016.

[5] http://www.adc.co.kr/technology/procnplat/processor/ae3200empress.php

[6] http://www.iometer.org/

[7] Y. H. Song et. al., "Cosmos openSSD: A PCIe-based open source SSD platform," in Proc. Flash Memory Summit, August. 2014.