http://dx.doi.org/10.5392/JKCA.2016.16.10.477

낸드플래시 메모리의 효율적인 ECC 패리티 저장 방법

Efficient Policy for ECC Parity Storing of NAND Flash Memory

김석만, 오민석, 조경록
충북대학교 전자정보대학 정보통신공학과

Seokman Kim(smkim@cbnu.ac.kr), Minseok Oh(ohms747@cbnu.ac.kr),
Kyoungrok Cho(krcho@cbnu.ac.kr)

요약

본 논문은 ECC(error correcting code)의 오버헤드를 고려한 패리티의 저장 정책 및 그에 따른 낸드 플래시 메모리 컨트롤러의 구조를 제안한다. 일반적인 낸드 플래시 메모리의 용법은 데이터 영역과 스페어 영역을 분리하는 것이다. ECC 패리티는 낸드 플래시 메모리에 데이터가 입력될 때 생성된다. 일반적으로 ECC의 메시지 길이는 낸드 플래시 메모리의 한 페이지 보다 작기 때문에, 각 메시지의 패리티를 모두 모아 스페어 영역에 저장하게 된다. 읽기 동작 시에는 데이터 영역에 이어 스페어 영역의 ECC 패리티까지 모두 읽은 후에 ECC 처리를 통한 데이터 정정이 가능하다. 이 때 발생하는 오버헤드를 줄이기 위해 데이터/스페어 영역의 구분없이 ECC 처리된 데이터와 패리티를 연속으로 저장하는 분산형 정책을 사용하였다. 제안된 분산형 정책과 기존의 수집형 정책의 오버헤드를 설계적인 측면과 타이밍 측면으로 분석하고, 그에 맞는 낸드 플래시 메모리 컨트롤러의 구조를 제시한다. 페이지의 크기에 따른 액세스 시간을 시뮬레이션을 통해 분석한 결과, 읽기 동작 시, 분산형 정책의 액세스 시간이 수집형 정책에 비해 짧았고 페이지의 크기가 커질수록 감소율이 컸다. 실험에 사용된 16KB의 페이지 크기를 갖는 낸드 플래시 메모리의 경우 분산형 정책의 액세스 시간이 수집형 정책에 비해 13.6% 감소하였다. 이는 4GB 크기의 영상 파일을 읽을 때 약 1분가량의 시간이 단축되는 효과를 얻을 수 있다. 또한 읽기 동작이 많은 SSD(solid state drive)의 특성 상 전반적인 시스템의 성능 향상을 기대할 수 있다.

■ 중심어 : | SSD 컨트롤러 | ECC | 패리티 정책 |

Abstract

This paper presents a new method of parity storing for ECC(error correcting code) in SSD (solid-state drive) and suitable structure of the controller. In general usage of NAND flash memory, we partition a page into data and spare area. ECC parity is stored in the spare area. The method has overhead on area and timing due to access of the page memory discontinuously. This paper proposes a new parity policy storing method that reduces overhead and R(read)/W(write) timing by using whole page area continuously without partitioning. We analyzed overhead and R/W timing. As a result, the proposed parity storing has 13.6% less read access time than the conventional parity policy with 16KB page size. For 4GB video file transfer, it has about a minute less than the conventional parity policy. It will enhance the system performance because the read operation is key function in SSD.

■ keyword : | SSD Controller | ECC | Parity Policy |

* 본 논문은 충북대학교 학술연구지원사업의 연구비 지원에 의하여 연구되었음.
접수일자 : 2016년 08월 11일
심사완료일 : 2016년 09월 29일
수정일자 : 2016년 09월 29일
교신저자 : 조경록, e-mail : krcho@cbnu.ac.kr

I. 서 론

기술 발전과 맞물려 소비되는 디지털 콘텐츠의 데이터 양이 점점 방대해지고 있다. 단적으로 UHD, 4K, 3D 영상과 같은 디지털 콘텐츠는 기존의 Full HD 영상에 비해 수 배의 데이터 양을 갖는다. 따라서 이를 처리하기 위해 고성능의 시스템이 필요하며, 이러한 데이터를 저장하기 위한 고속의 대용량 저장장치도 필요하다.

최근 각광을 받고 있는 SSD(solid-state drive)는 HDD(hard-disk drive)에 비해 빠른 액세스 시간과 높은 내구성을 갖는다. MLC(multilevel cell)나 TLC(triple-level cell)를 채택한 낸드 플래시 메모리가 사용되면서 SSD의 용량이 증가하게 되었다. 그로인해 SSD의 용도가 운영체제용에서 주요 데이터 저장 장치로 확대되고 있다. 또한 SSD는 기계적인 구동부가 없기 때문에 충격에 강해 모바일 환경에서 HDD에 비해 강점을 갖는다. 노트북 PC와 같이 최근에 출시되는 모바일 컴퓨팅 기기들은 SSD가 장착되는 것이 기본으로 여겨지며 기존의 하드디스크 드라이브를 빠르게 대체하고 있다.

고성능의 대용량 SSD를 구현하기 위해서는 여러 개의 낸드 플래시 메모리를 병렬로 구성하는 멀티 채널, 한 채널 내에서 버스 사용율을 높이는 멀티 웨이 구성 등의 기법이 사용된다. MLC나 TLC와 같은 방식을 사용하여 낸드 플래시 메모리의 집적도를 높이기도 한다. 하지만 MLC나 TLC 방식은 SLC에 비해 비트에러율(BER, bit-error ratio)이 높기 때문에 ECC(error correction code)의 사용이 필수적이다. 또한 적용된 시스템의 특성에 적합한 FTL(flash translation layer) 알고리즘의 선택이 중요하다. 이런 여러 가지 조건에 따른 SSD의 성능을 분석하기 위한 시뮬레이터에 대한 다양한 연구가 진행되었다[1-4].

이런 분석들은 FTL 알고리즘의 종류, 낸드 플래시 메모리의 특성 등 다양한 파라미터를 사용하여 시스템을 분석하지만, 낸드 플래시 메모리를 직접 제어하는 낸드 플래시 메모리 컨트롤러의 오버헤드는 고려되지 않는 이상적인 환경에서 분석을 실시한다. 그러나 낸드 플래시 메모리 컨트롤러의 설계 방법이나 off-chip ECC의 적용 유무에 따라 낸드 플래시 메모리 컨트롤러에서의 오버헤드가 분명히 존재한다. 일반적인 낸드 플래시 메모리의 용법은 데이터 영역과 스페어 영역을 분리하는 것이다. ECC 패리티는 낸드 플래시 메모리에 데이터가 입력될 때 생성된다. 일반적으로 ECC의 메시지 길이는 낸드 플래시 메모리의 한 페이지 보다 작기 때문에, 각 메시지의 패리티를 모두 모아 스페어 영역에 저장하게 된다. 읽기 동작 시에는 데이터 영역에 이어 스페어 영역의 ECC 패리티까지 모두 읽은 후에 ECC 처리를 통한 데이터 정정이 가능하다. 본 논문에서 제안하는 ECC 저장 정책은 낸드 플래시 메모리의 데이터 영역과 스페어 영역을 구분하지 않고 ECC 패리티를 메시지 데이터에 연달아 저장하는 분산형 정책이다. 본 논문에서는 이러한 ECC의 패리티를 저장하는 정책에 따른 오버헤드를 설계적인 측면과 타이밍의 측면에서 분석하고 그에 따른 설계 방식을 제시한다.

II. 낸드 플래시 메모리 컨트롤러의 구조

일반적인 SSD의 구조는 [그림 1]과 같다. SSD는 호스트 시스템(PC)과 명령 및 데이터를 전송하기 위한 호스트 인터페이스, 데이터를 저장하기 위한 메모리(버퍼), FTL 알고리즘을 위한 프로세서 코어, 낸드 플래시 메모리를 제어하기 위한 낸드 플래시 컨트롤러 등으로 구성된다. 낸드 플래시 컨트롤러는 SSD의 프로세서 코어가 낸드 플래시를 제어할 수 있도록 인터페이스를 제공한다. SSD가 고성능 대용량이 되면서 여러 개의 낸드 플래시 메모리를 사용하고 이를 채널과 웨이로 구성한다. 대용량 SSD의 경우 MLC(혹은 TLC) 낸드 플래시 메모리를 사용한다. MLC 낸드 플래시 메모리는 데이터의 안정성이 떨어지기 때문에 ECC의 사용이 필수적이다. ECC는 낸드 플래시 메모리 내부의 컨트롤러에 존재하는 on-chip ECC와 낸드 플래시 메모리 외부의 컨트롤러가 사용하는 off-chip ECC로 나뉠 수 있다. Off-chip ECC는 보드 레벨에서 신뢰성 확보를 위해 사용되고, on-chip ECC는 MLC 및 TLC에서 메모리 자체의 신뢰성 확보를 위해 주로 사용된다[5][6]. 본 논문에서는 off-chip ECC를 사용하는 경우를 대상으로 한다.

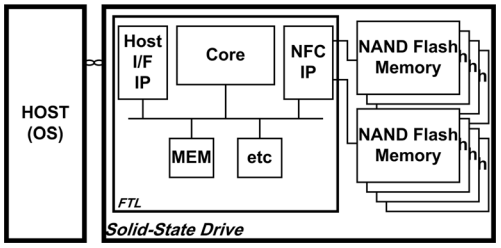

그림 1. SSD 시스템의 구조

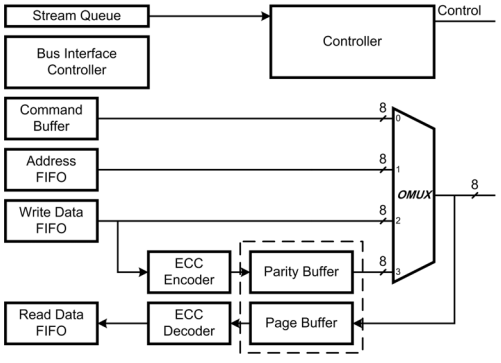

그림 2. 낸드 플래시 메모리 컨트롤러의 구조

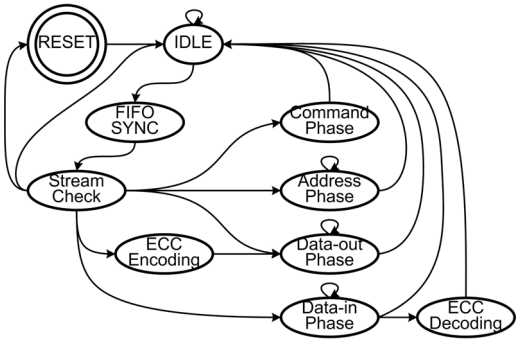

그림 3. 제어기 FSM의 상태도

낸드 플래시 메모리 컨트롤러의 구조는 [그림 2]와 같으며, SSD의 프로세서 코어로부터 전달받은 명령어 및 주소, 데이터를 저장하기 위한 FIFO 형태의 버퍼와 제어기, ECC 블록들로 구성이 된다. 낸드 플래시 메모리는 명령어와 주소 전달을 통해 동작한다. 명령어, 주소, 데이터 버퍼에 할당된 주소를 기반으로 메모리를 액세스 하는 단계를 구분하여 Stream Queue에 저장한다. 이를 바탕으로 제어기가 각 블록들을 순서에 맞게 제어한다.

제어기 FSM(finite state machine)의 상태도는 [그림 3]과 같다. 낸드 플래시 메모리의 명령, 주소, 데이터 입출력 단계를 진행하는 FSM이 각각 존재하며, Stream Check 상태에서 Stream Queue의 내용에 따라 각 단계의 다른 동작 상태로 분기한다. ECC를 사용하는 경우에는 데이터 출력 단계 전에 ECC Encoding 상태를 거치며, 데이터 입력 단계 후에 ECC Decoding 상태를 거친다.

III. 패리티 저장 정책에 따른 액세스 시간 모델링

ECC의 패리티를 저장하는 두 가지 정책이 있다. 첫 번째는 패리티를 모두 모아 스페어 영역에 저장하는 수집형, 두 번째는 ECC의 처리 단위에 따른 패리티를 데이터에 연달아 저장하는 분산형이다. [그림 4]는 패리티 정책에 따라 페이지 공간이 할당되는 예를 보여준다.

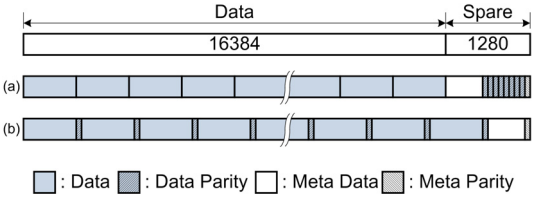

그림 4. 페이지 내의 패리티 저장 정책
(a) 수집형 (b) 분산형

수집형 정책은 일반적인 SSD 환경에서 주로 사용되는 방법이다. 낸드 플래시 메모리의 기본 기록 단위는 페이지이다. 페이지는 데이터 영역과 스페어 영역으로 구분되는데 일반적인 SSD 환경에서는 데이터 영역에 데이터를 저장하고 ECC의 패리티 정보를 비롯한 관리 정보들을 스페어 영역에 저장한다. 수집형 정책에서는 데이터를 기록할 때, 순수 데이터와 메타 데이터가 전송되면서 ECC에서 발생한 패리티들을 임시로 저장하기 위한 버퍼가 필요하다. 데이터를 읽을 때 에러 정정을 위해서는 ECC의 패리티가 필요하므로 패리티가 포

함된 스페어 영역까지 모두 읽어서 컨트롤러 내부에 저장해두어야 한다. 따라서 설계 시 면적의 오버헤드가 발생하게 된다.

반면 분산형 정책에서는 페이지를 데이터 영역과 스페어 영역의 구분 없이 ECC의 처리 단위마다 데이터와 패리티를 연속으로 저장하는 것이다. 분산형 정책에서 데이터를 기록할 때, 패리티를 수집할 필요가 없으므로 패리티를 저장하기 위한 버퍼가 필요 없다. 마찬가지로 데이터를 읽을 때도 페이지를 저장하기 위한 버퍼가 필요 없다. [표 1]은 두 정책의 오버헤드를 비교한 것이다.

표 1. 낸드 플래시 메모리 컨트롤러의 설계 오버헤드

동작	오버헤드	수집형	분산형
쓰기	설계	패리티 버퍼	×
	타이밍	×	zero padding
읽기	설계	페이지 버퍼	×
	타이밍	ECC 레이턴시 + 페이지 데이터 액세스 시간	ECC 레이턴시

낸드 플래시 메모리의 동작 특성과 ECC 정책에 따른 오버헤드를 고려한 컨트롤러의 제어방법에 의해 낸드 플래시 메모리의 액세스 시간을 다음과 같이 구할 수 있다. 식 (1)과 (2)는 ECC 정책에 따른 읽기 동작 시 낸드 플래시 메모리 컨트롤러가 소요하는 시간이다. 식 (3)과 (4)는 ECC 정책에 따른 액세스 시간을 컨트롤러에서 사용하는 클럭 사이클을 기준으로 표현한 것이다. 컨트롤러가 사용하는 클럭 사이클은 낸드 플래시 메모리의 타이밍 특성 중 동작의 기본이 되는 t_{RC}(read cycle time)와 t_{WC}(write cycle time)에 의해 결정된다. 일반적으로 낸드 플래시의 읽기/쓰기 사이클은 6단계를 거쳐 생성이 되므로, 컨트롤러의 클럭 주기는 tRC, tWC의 1/6정도로 정할 수 있다.

$$T_{r_{dist}} = T_{cmd} + T_{addr} + tR + tRC \times pageSize + L_{ECC} \quad (1)$$

$$T_{r_{coll}} = T_{cmd} + T_{addr} + tR + tRC \times pageSize + L_{ECC} + T_{dk} \times pageSize \quad (2)$$

$$C_{r_{dist}} = C_{cmd} + C_{addr} + \lceil \frac{tR}{T_{dk}} \rceil + phaseCycles \times pageSize + L_{ECC} \quad (3)$$

$$C_{r_{coll}} = C_{cmd} + C_{addr} + \lceil \frac{tR}{T_{dk}} \rceil + phaseCycles \times pageSize + L_{ECC} + pageSize \quad (4)$$

T_{cmd}는 명령어 전달 시간, T_{addr}는 주소 전달 시간, tR은 메모리 내에서 셀의 데이터를 읽어 내부 버퍼에 저장하는 시간, tRC는 컨트롤러에서 메모리의 데이터를 읽어오는 시간, $phaseCycles$는 메모리를 액세스하는 각 단계에 소요되는 클럭 사이클 수, $pageSize$는 메모리의 한 페이지 크기, L_{ECC}는 ECC에 데이터가 입력된 뒤 출력을 얻을 수 있는 ECC의 레이턴시이다.

IV. 실 험

ECC의 패리티 저장 정책에 따른 오버헤드를 실제 낸드 플래시 메모리의 특성을 적용하여 확인하였다. 본 논문에서 사용된 낸드 플래시 메모리 컨트롤러는 낸드 플래시 메모리의 각 동작 단계 별로 FSM이 동작하고 각 단계는 6개의 상태를 순차적으로 실행하도록 구현되었다. 따라서 낸드 컨트롤러의 동작 클럭의 주기는 낸드 플래시 메모리의 tWC와 tRC의 약 1/6이다. 실험에 사용된 낸드 플래시 메모리는 SK Hynix의 8Gb 낸드 플래시 이며, tWC와 tRC는 100ns 이고, 페이지의 크기는 데이터 영역이 16,384 바이트, 스페어 영역이 1,280 바이트이다.

사용된 ECC는 Reed-Solomon이 사용되었으며 RS(207, 205)가 적용되었다. ECC의 에러 정정 능력은 40 bits/1,024 Bytes로 적용하였다. 심볼의 크기가 1바이트인 Reed-Solomon의 최대 코드 길이는 255 바이트이므로 1,024 바이트를 한 번에 처리를 할 수 없다.[7] 따라서 1,024 바이트를 효율적으로 나누어 처리하기 위한 코드 길이를 205 바이트로 선정하였고, 각 처리 단위 내에서 8 비트의 연집오류를 정정할 수 있도록 하였다.

1,024 바이트를 5번에 걸쳐 ECC 처리를 하고 1 심볼의 zero 패딩이 추가되었다. 따라서 1,024 바이트 내에서 8 비트의 연집오류 수정이 가능하고, 총 40 비트의 분산 오류 수정이 가능하다. 메타 데이터를 제외한 한 페이지의 데이터를 처리하기 위한 패리티는 총 160 바이트이다. 쓰기 동작 시 ECC의 레이턴시는 3 사이클, 읽기 동작 시는 ECC의 코드 워드 길이인 207 사이클이다.

쓰기 동작의 경우, 수집형 정책에서는 160 바이트의 패리티를 저장하기 위한 버퍼가 필요하다. 타이밍 오버헤드는 발생하지 않는다. 반면 분산형 정책에서는 데이터와 패리티가 순차적으로 저장이 되므로 패리티를 수집하기 위한 별도의 버퍼가 필요하지 않다. 다만 1,024 바이트 마다 발생하는 1 심볼의 zero 패딩을 저장해야 하기 때문에 zero 패딩 저장에 따른 타이밍 오버헤드가 발생한다. 16,384 바이트의 데이터를 저장할 경우 총 16 바이트에 해당하는 타이밍 오버헤드(16*tWC)와 스페어 영역 오버헤드가 발생한다.

읽기 동작의 경우, 수집형 정책에서는 데이터 크기만큼의 버퍼가 필요하므로 16,384 바이트의 버퍼가 필요하다. 한 페이지의 데이터를 읽는데 필요한 클럭 사이클의 수는 식 (4)에 의해 120,232 사이클이다. 분산형 정책에서는 데이터 저장을 위한 버퍼가 필요 없고, 한 페이지의 데이터를 읽는데 필요한 클럭 사이클의 수는 식 (3)에 의해 103,848 사이클이다. 분산형 정책이 수집형 정책에 비해 13.6% 적은 사이클로 한 페이지의 데이터를 읽을 수 있다.

페이지의 크기가 다른 다수의 디바이스를 선정하고 4GB의 영상 파일을 읽는 것을 가정하였다. 각 디바이스들이 갖는 파라미터들을 사용하여 식 (3), (4)에 의한 한 페이지의 액세스 시간을 구하고, 4GB 크기의 영상 파일 전체를 읽는데 걸리는 시간을 시뮬레이션을 통해 비교했다. 또한 분산형 정책의 액세스 시간이 수집형 정책에 비해 감소된 비율을 각 정책의 액세스 시간과 함께 [그림 5]에 나타냈다. 페이지 크기가 커질수록 분산형 정책의 효과가 커지며 10%내외의 감소율을 갖는다. 일반적으로 많이 사용되는 4GB 크기의 영상 파일의 경우 1분 가량 전송시간이 단축되는 효과를 갖는다.

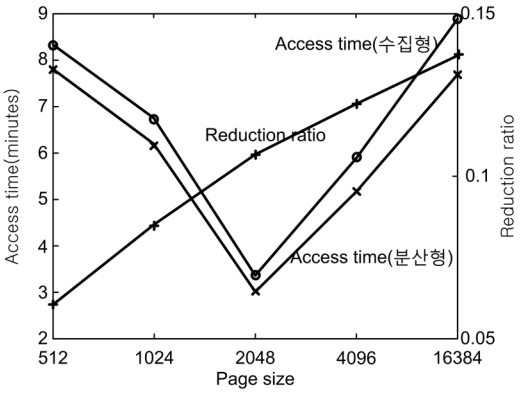

그림 5. 페이지 크기에 따른 정책 별 액세스 시간

V. 결 론

본 논문에서는 SSD를 위한 시스템 구성에서 ECC 사용에 의한 패리티 저장 정책과 그에 따른 낸드 플래시 메모리 컨트롤러의 구조에 대해 제안하였다. 패리티 저장 정책에 따라 수집형과 분산형으로 나누었고 각 정책의 오버헤드를 설계적인 측면과 타이밍 측면에서 분석하였다. 기존에 일반적으로 사용되고 있는 수집형 정책에 비해 분산형 정책을 채택하는 경우 설계적인 측면에서 버퍼의 사용이 줄어들어 구조가 간단해지고, 타이밍에 대한 오버헤드가 줄어들기 때문에 읽기 동작 시 처리 속도도 빨라지는 것을 알 수 있었다. 16KBytes 의 페이지 크기를 갖는 낸드 플래시 메모리의 경우 한 페이지의 데이터를 읽을 때 수집형 정책에 비해 분산형 정책이 약 13.6%의 시간을 단축하였다. 이는 4GB의 영상 파일의 경우 약 1분가량 읽는 시간이 단축되는 효과이다. 쓰기 동작에 비해 읽기 동작이 많은 SSD의 사용 특성상 전체적인 성능 향상을 얻을 수 있다.

참 고 문 헌

[1] D. Kim, K. Bang, S. H. Ha, S. W. Chung, and E. Y. Chung, “A Transaction Level Simulator for Performance Analysis of Solid-State

Disk(SSD) in PC Environment," Journal of The Institute of Electronics Engineers of Korea, Vol.45, No.12, pp.57-64, 2008.

[2] C. Sun, T. O. Iwasaki, T. Onagi, K. Johguchi, and K. Takeuchi, "Cost, Capacity, and Performance Analyses for Hybrid SCM/NAND Flash SSD," IEEE Trans.action on Circuits and Systems-I: Regular Papers, Vol.61, No.8, 2014(8).

[3] Y. Kim, B. Tauras, A. Gupta, and B. Urgaonkar, "FlashSim: A Simulator for NAND Flash-based Solid-State Drives," 2009 First International Conference on Advances in System Simulation, Porto, Portugal, pp.125-131, 2009(9).

[4] L. Zuolo, C. Zambelli, R. Micheloni, S. Galfano, M. Indaco, S. Di Carlo, P. Prinetto, P. Olivo, and D. Bertozzi, "SSDExplorer: a Virtual Platform for Performance/Reliability-oriented Fine-Grained Design Space Exploration of Solid State Drives," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems(TCAD), Vol.34, No.10, pp.1627-1638, 2015.

[5] S. Gregori, A. Cabrini, 0. Khouri, and G. Torelli, "On-chip error correction techniques for new-generation flash memories," Proc. of IEEE, Vol.91, No.4, pp.602-616, 2003(4).

[6] W. Liu, J. Rho, and W. Sung, "Low-power high-throughput BCH error correction VLSI design for multi-level cell NAND fash memories," IEEE Workshop on Signal Processing Systems (SIPS), pp.248-253, 2006.

[7] https://en.wikipedia.org/wiki/Reed-Solomon_error_correction

저 자 소 개

김 석 만(Seokman Kim) 정회원

- 2005년 2월 : 충북대학교 전기전자컴퓨터공학부(공학사)
- 2008년 2월 : 충북대학교 정보통신공학과(공학석사)
- 2008년 3월 ~ 현재 : 충북대학교 정보통신공학과(박사과정)

<관심분야> : 저전력 회로 설계, SoC 설계

오 민 석(Minseok Oh) 준회원

- 2016년 2월 : 충북대학교 정보통신공학과(공학사)
- 2016년 3월 ~ 현재 : 충북대학교 정보통신공학과(석사과정)

<관심분야> : 통신시스템설계, 저전력 회로 설계

조 경 록(Kyoungrok Cho) 정회원

- 1977년 : 경북대학교 전자공학과(공학사)
- 1989년 : 일본 동경대학교 전자공학과(공학석사)
- 1992년 : 일본 동경대학교 전자공학과(공학박사)
- 1979년 ~ 1986년 : ㈜금성사TV연구소 선임연구원
- 1999년 ~ 2005년 : Oregon State University 객원교수
- 1992년 ~ 현재 : 충북대학교 전자정보대학 교수
- 2010년 ~ 현재 : IDEC 충북대 지역센터장

<관심분야> : 통신시스템LSI설계, 저전력 고속회로 설계, Platform 기반의 SoC 설계