논문 2016-11-11

비휘발성 메모리 기반 캐시의 쓰기 작업 최적화를 위한 캐시 시뮬레이터 설계

(Cache Simulator Design for Optimizing Write Operations of Nonvolatile Memory Based Caches)

주 용 수, 김 명 회, 한 인 규, 임 성 수*
(Yongsoo Joo, Myeung-Heo Kim, In-Kyu Han, Sung-Soo Lim)

Abstract : Nonvolatile memory (NVM) is being considered as an alternative of traditional memory devices such as SRAM and DRAM, which suffer from various limitations due to the technology scaling of modern integrated circuits. Although NVMs have advantages including nonvolatility, low leakage current, and high density, their inferior write performance in terms of energy and endurance becomes a major challenge to the successful design of NVM-based memory systems. In order to overcome the aforementioned drawback of the NVM, extensive research is required to develop energy- and endurance-aware optimization techniques for NVM-based memory systems. However, researchers have experienced difficulty in finding a suitable simulation tool to prototype and evaluate new NVM optimization schemes because existing simulation tools do not consider the feature of NVM devices. In this article, we introduce a NVM-based cache simulator to support rapid prototyping and evaluation of NVM-based caches, as well as energy- and endurance-aware NVM cache optimization schemes. We demonstrate that the proposed NVM cache simulator can easily prototype PRAM cache and PRAM+STT-RAM hybrid cache as well as evaluate various write traffic reduction schemes and wear leveling schemes.

Keywords : Nonvolatile, Cache, Simulator, PRAM, STT-RAM

Ⅰ. 서 론

기존의 SRAM이나 DRAM 기반 메모리 시스템에서 공정 미세화에 따른 누설 전류 증가, 데이터 보존 능력의 저하 등의 한계점이 부각됨에 따라 차세대 비휘발성 메모리 (NVM, nonvolatile memory)를 기반으로 하는 메모리 시스템이 이를 극복하기 위한 대안으로 주목받고 있다.

대표적인 비휘발성 메모리로는 PRAM (상변화 메모리) [2], STT-RAM (스핀토크 자기메모리) [3], ReRAM (저항변화 메모리) [4] 등이 있다. 이들 메모리는 기존의 SRAM이나 DRAM과 비교하여 비휘발성, 고밀도, 낮은 누설 전류량 등의 특징을 지니고 있으며, NAND 플래시 메모리와 달리 쓰기 전 삭제 (erase-before-write) 작업이 필요치 않아 이로 인한 메모리 시스템 관리 비용 증가 및 성능 저하를 겪지 않는다.

이러한 장점에도 불구하고 비휘발성 메모리는 기존 메모리 기술과 비교하여 쓰기 작업(write operation)에서의 낮은 성능, 높은 전력소모, 최대 쓰기 횟수 제약이 심각한 단점으로 알려져 있으며, 비휘발성 메모리 시스템의 성공적인 구현을 위해서

*Corresponding Author (sslim@kookmin.ac.kr)
Received: 29 Jan. 2016, Revised: 9 Mar. 2016. Accepted: 28 Mar. 2016.
Y. Joo, I.-K. Han, S.-S. Lim: Kookmin University
M.-H. Kim: Hansol Technics
※ 이 논문은 2015년도 정부(교육부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업임(No. 2015R1D1A1A0105831).
※ 이 논문은 2015 한국컴퓨터종합학술대회에서 “비휘발성 메모리 캐시 시뮬레이터 설계 및 구현”이란 제목으로 발표된 논문 [1]을 확장한 것임.

 2016 Apr. 11(2) 87-95
ISSN : 1975-5066
http://dx.doi.org/10.14372/IEMEK.2016.11.2.87

는 비휘발성 메모리의 쓰기 성능, 전력 및 내구성을 향상시킬 수 있는 다양한 최적화 기법 [5-8]의 연구가 필수적이다.

연구자들은 이러한 최적화 기법의 연구를 위해 초기 설계 단계에서부터 메모리 시스템의 성능을 예측하고 최적화 기법의 성능을 평가할 수 있는 시뮬레이션 도구를 활용해 왔다. 하지만 기존의 시뮬레이션 도구들은 비휘발성 메모리의 특성에 대한 고려가 되어 있지 않아 비휘발성 메모리 시스템 설계에서 활용하는데 한계가 따른다.

예를 들어 기존의 캐시 시뮬레이터는 캐시를 구성하는 각각의 메모리 셀에 쓰여지는 데이터 값을 추적하는 기능을 지원하지 않는데, SRAM 기반 캐시 설계에서는 실제로 쓰여지는 데이터 값이 캐시 성능에 미치는 영향이 극히 제한적이기 때문에 해당 기능을 지원하지 않음으로써 시뮬레이션 속도를 높이는 것이 유리했다. 하지만 비휘발성 메모리 기반 캐시에서는 캐시의 전력 소모 및 유효 수명을 평가하기 위해 해당 기능이 반드시 필요하며, 이러한 특성을 반영하여 비휘발성 캐시를 위한 새로운 캐시 시뮬레이터를 개발할 필요가 있다.

본 논문에서는 전체 메모리 시스템 중 캐시에 초점을 맞추어 비휘발성 캐시를 위한 최적화 기법의 성능 평가를 지원하는 캐시 시뮬레이터의 설계를 제안하고, 제안한 시뮬레이터를 사용하여 PRAM 캐시 및 STT-RAM+PRAM 하이브리드 캐시를 구현한 사례를 소개한다.

본 논문에서 제안한 캐시 시뮬레이터는 메모리 셀 단위의 쓰기 회수 추적 기능이 필수적인 NVM 기반 캐시 최적화 기법에 대한 신속한 프로토타이핑과 성능평가를 지원함으로써 관련 연구자들이 이를 사용하여 NVM 기반 캐시의 성능, 에너지뿐만 아니라 수명 향상 연구를 활성화하는 데 기여할 수 있을 것으로 예상된다.

Ⅱ. 관련 연구

현재 가장 널리 사용되고 있는 비휘발성 메모리인 플래시 메모리를 위한 대표적인 시뮬레이터로는 FlashSim [9], SSD extension for Disksim [10] 등이 있는데, 이들 시뮬레이터는 NAND 플래시 메모리 기반 솔리드 스테이트 드라이브(SSD)의 성능 평가를 위해 SSD 제어기의 정책에 따른 성능, 에너지 및 수명 예측 기능을 지원한다. 하지만 쓰기 회수 추적은 NAND 플래시의 페이지 및 블록 단위로만 이루어지며, 비휘발성 메모리 기반 캐시의 쓰기 작업 최적화를 위해 필요한 메모리 셀 단위의 쓰기 회수 추적 기능은 구현되어 있지 않다.

캐시 시뮬레이션을 지원하는 대표적인 시스템 수준 시뮬레이터로는 FLASH simulator [11], SimOS [12], SIMICS [13], SimpleScalar [14] 등이 있으며, 이들 시뮬레이터는 캐시나 주기억 장치 등의 특정 구성요소에 대한 자세한 시뮬레이션보다는 전체 시스템 수준에서 운영체제 및 사용자 애플리케이션에 대한 시뮬레이션을 위해 개발되었기 때문에 시뮬레이션 속도를 높이기 위한 방향으로 최적화가 이루어져 있다. 특히 캐시 시뮬레이션의 경우 실제 캐시 메모리 내부의 변화를 정확히 시뮬레이션하는 대신에 시스템 수준 성능평가를 수행하기 위한 최소한의 기능, 즉 입력 주소 값의 변화만을 추적하도록 구현되어 시뮬레이션 속도를 높인다. Dinero IV [15]와 Cachegrind [16]은 트레이스 수준에서 캐시 시뮬레이션을 수행하며 캐시 적중률, 메모리 참조 패턴, 명령어 실행 횟수 추적 등의 캐시에 특화된 성능 평가 지표를 제공한다. 특히 Cachegrind는 Valgrind 런타임 프레임워크에 포함되어 있어 실제 Linux 운영체제를 구동하면서 캐시의 성능을 평가할 수 있을 정도의 높은 시뮬레이션 속도를 제공한다. MemSpy [17]는 메모리 병목현상을 감지하는 성능 모니터링 도구이며, 캐시 시뮬레이터를 사용하여 메모리 접근 패턴 및 캐시 적중 실패의 원인 분석 기능을 제공한다.

위에 열거한 캐시 시뮬레이터들은 캐시 적중률에 따른 캐시의 성능을 분석하는 기능을 반드시 제공해야 하기 때문에 캐시 접근 시의 입력 주소 값의 변화를 정확히 추적할 수 있도록 구현되어 있다. 하지만 캐시 읽기/쓰기 작업을 수행할 때 실제로 캐시 내부의 메모리 셀에 값이 쓰여지고 읽히는 과정은 구현하지 않음으로써 시뮬레이터의 속도를 빠르게 하는데에 최적화되어 있다. 따라서 기존 캐시 시뮬레이션 도구들은 NVM 캐시의 쓰기 작업과 관련된 성능, 전력 소모 및 수명을 분석하기에는 적합하지 않으며, NVM 캐시의 쓰기 최적화 기법을 연구하기 위해서는 NVM 캐시에 특화된 새로운 캐시 시뮬레이터를 구현할 필요가 있다.

Ⅲ. NVM 캐시 시뮬레이터

본 연구에서 구현한 NVM 캐시 시뮬레이터의 주요 기능은 다음과 같다.

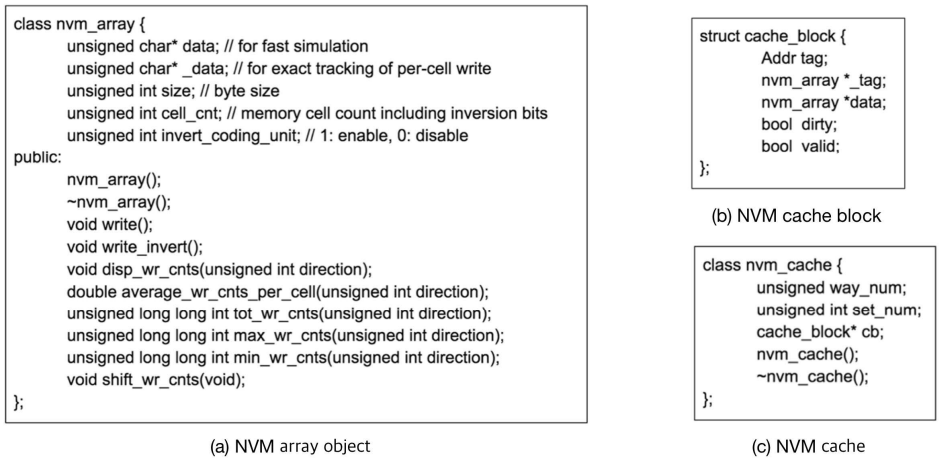


그림 1. NVM 클래스
Fig. 1 NVM class

1. NVM 셀 단위 쓰기 횟수 추적 기능

각각의 NVM 셀마다 NVM 쓰기 작업횟수를 기록하여 정확한 수명 예측을 지원한다. 그림 1(a)의 NVM 배열 객체는 한 워드를 표현하고 있으며, 워드의 크기는 바이트 단위로 설정 가능하다(unsigned int size). 데이터는 CPU 또는 메모리 등 다른 구성요소와의 통신에 적합하도록 바이트 단위로 저장되어 있으며(unsigend char* _data), 각각의 NVM 셀마다 쓰기 회수를 기록하기 위해 비트 단위로도 풀어서 저장된다(unsigend char* _data). NVM 캐시는 쓰기 회수 절감을 위해 원래 데이터를 invert시켜 저장시키는 data inverting 기법 [18, 19]을 사용하는데, 해당 기능이 활성화되어 있는지의 여부를 나타내기 위해 상태 변수를 포함한다(unsigned int invert_coding_unit). NVM 캐시의 전력 소모 또는 수명을 평가하지 않아도 되는 경우는 셀별 쓰기 횟수 추적 기능을 비활성화하여 시뮬레이션 속도를 높이도록 설계하였다.

2. 객체지향 설계 기반의 NVM 모델링

NVM의 일반적인 속성을 모델링한 NVM 클래스(그림 1(b), 그림 1(c))를 정의하고 이를 상속하여 PRAM, STT-RAM, ReRAM 등의 다양한 NVM으로 확장 가능하도록 하였다.

- **NVM 캐시 블록**: 그림 1(b)는 NVM 캐시 블록을 표현하는데, 입력 주소에 따른 해당 캐시 블록을 찾기 위해 tag 값이 Addr tag 변수와 nvm_array *_tag에 각각 기록된다. 전자는 캐시 동작 자체를 빠르게 시뮬레이션하기 위해 바이트 단위로 tag 값을 저장하며 후자는 tag에 해당하는 NVM 셀의 쓰기 회수 추적을 위해 비트 단위로 tag 값을 풀어서 저장한다.
- **NVM 캐시**: 그림 1(c)는 NVM 캐시 블록을 배열로 구성하여 NVM 캐시 객체를 표현하는데, 해당 캐시 객체에는 set의 개수(unsigned int set_num)와 way의 개수(unsigned int way_num)의 곱만큼의 캐시 블록 배열(cache_block* cb)에 대한 포인터가 연결되어 있으며, 캐시 초기화 및 할당된 메모리 반납을 위한 생성자와 소멸자 함수가 함께 제공된다.

이러한 객체지향 기반 설계를 통해 NVM 캐시 시뮬레이터는 다양한 종류의 NVM 캐시에서 공통적으로 지원해야 하는 기능들을 기본적으로 제공하며, 특정 NVM에 필요한 기능을 유연하게 확장할 수 있다. 이뿐만 아니라 객체를 상속하여 다른 종류의 NVM을 조합한 새로운 하이브리드 캐시 구조에 대한 신속한 프로토타이핑을 하는 것이 가능하다.

3. NVM 모델 통합

NVM 캐시의 성능을 다각도에서 평가하기 위해 성능, 에너지, 면적, 내구성 모델을 시뮬레이터에 통합 구현하였고 파라미터 값을 조정하여 시뮬레이터 상에서 다양한 종류의 NVM을 손쉽게 구현할 수 있도록 하였다. 표 1은 PRAM을 사용하여 구현한 NVM 캐시에서 세부 구성요소별로(표 2 참조) 동작에 필요한 지연시간, 동적 에너지 (dynamic energy)

표 1. PRAM 기반 캐시의 성능, 에너지, 면적 모델
Table 1. Performance, energy, and area model of a PRAM-based cache

Timing value (unit: ns)							
T_{wsor} T_{sh_d}	0.5 0.4	T_{add} T_{dec_d}	0.2 1.2	$T_{wd}+T_{wl\_d}$ T_{set}	2.05 150.0	T_{pcm_rd} T_{omux}	0.82 1.48
Energy value (unit: pJ)							
E_{set} E_{omux} E_{ht_rd}	4.5 0.0192 536.14	E_{reset} E_{sa} E_{ht_wr}	6.9 0.535 537.187	E_{pcm_rd} $E_{wd}+E_{wl\_d}$ E_{dec_t}	0.82 2.33 0.097	E_{add} E_{sh_d} E_{sh_d}	0.026 0.0136 0.0136
Power value (unit: nW)							
P_{add} P_{sh}	125.67 17771.8x2	P_{ht}	1511.5	P_{sa}	cbs x wds x 8 x snum x (1.42877+352.8)	P_{wd}	15.175
Area value (unit: um^2)							
A_{pcm} A_{add}	0.06075 313	A_{wd} A_{wsor}	31.617 3.6855	A_{sa} A_{omux}	cbs x wds x 8 x snum x 25.1364 cbs x wds x 8 x snum x 42.93	A_{sh}	85759.9922

표 2. PRAM 캐시의 주요 구성요소
Table 2. Major components in a PRAM cache

Component	Meaning
wsor	wordline shift offset register
add	adder
wd	decoder
wl	word-line
sh	shifter
omux	output mux
set	PRAM set operation
reset	PRAM reset operation
sa	sense amplifier
ht	H-tree route wire
pcm	PRAM cell

소모치, 정적 전력 (static power) 소모치, 면적 값에 대한 파라미터의 종류와 예시값을 보여주고 있다.

시뮬레이터를 사용하는 연구자들은 PRAM뿐만 아니라 STT-RAM이나 RRAM 등 다양한 NVM에 대하여 표 1의 파라미터 값을 대상 NVM에 맞게 설정함으로써 손쉽게 새로운 NVM 캐시를 구현할 수 있다.

4. NVM 최적화 기법 지원

본 연구에서 개발한 NVM 캐시 시뮬레이터는 최근의 NVM 기반 메모리 시스템 연구 [18-20]에서 제안된 RBW (read-before-write) 기법, DI (data inverting) 기법, 블록 시프팅 (periodic block shifting) 기법 등을 기본적으로 제공한다. 아래에 소개되는 각 기법의 동작원리에 따르면 이들

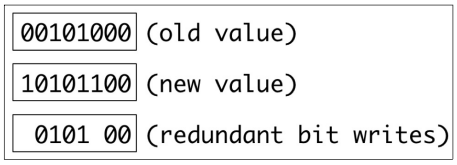


그림 2. Read-before-write (RBW) 기법
Fig. 2 Read-before-write (RBW) method

기법의 성능평가를 위해 본 시뮬레이터의 메모리 셀 단위 쓰기 회수 추적 기능이 필요함을 확인할 수 있다.

- **RBW (read-before-write) 기법**: NVM 메모리에서는 메모리 셀에 가해지는 쓰기의 양을 줄이기 위해 RBW 기법이 일반적으로 사용되며, 본 캐시 시뮬레이터 또한 RBW 기법을 지원한다. RBW 기법은 워드 단위로 데이터를 기록할 때 워드를 구성하는 모든 비트에 쓰기 작업을 수행하는 대신 이미 기록되어 있는 데이터 값과 새로 쓰여질 데이터 값을 비트 단위로 먼저 비교하는 작업을 수행한다. 이때 새로 쓰여지는 값이 예전 값과 같을 경우 해당 비트는 쓰기 작업을 수행하지 않도록 하여 불필요한 쓰기 전력 및 수명의 소모를 방지한다. 그림 2의 경우 새로 쓰여지는 값을 기존 값과 비교한 결과 8개의 비트 중 6비트가 동일하기 때문에 이를 제외한 나머지 두 개의 비트에 대해서만 쓰기 작업을 수행하게 된다. 그림 1(a)의 NVM 모델은 객체 생성자에서 비트 별로 데이터를 기록하고 새로운 값이 쓰여질 때 RBW 모드를 적용할지의 여부를 설정하도록 구현되어 있다.

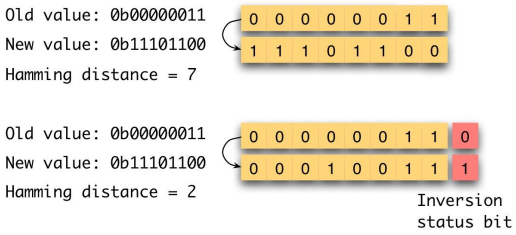


그림 3. Data inverting (DI) 기법
Fig. 3 Data inverting (DI) method

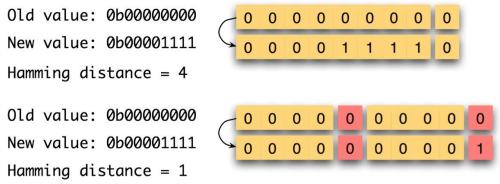


그림 4. Sub-block DI 기법
Fig. 4 Sub-block DI method

- **DI (data inverting) 기법**: DI 기법은 기존의 데이터 값과 새로운 데이터 값의 HD(hamming distance) 값을 먼저 계산한다. HD 값이 워드 크기의 절반을 넘는 경우에는 새로운 데이터 값 전체를 뒤집어 인코딩함으로써 HD 값을 절반 이하로 줄일 수 있는데, 최근의 NVM 메모리 시스템 연구 [18, 19]에서는 NVM 쓰기 회수를 줄이기 위해 DI 기법을 적용할 것을 제안하였다. 그림 3의 예에서는 기존 값과 새로운 값을 비교한 결과 8비트 중 7비트가 다르기 때문에 HD 값이 7이 되며, 따라서 데이터를 반전시켜 저장한 결과 최종 HD 값이 1이 됨을 알 수 있다. 이때 데이터를 반전시켜 저장했는지의 여부를 기록하기 위한 추가의 inversion status 비트가 필요하게 되고, 따라서 한 워드의 비트 수는 9로 증가하며 유효 HD 값도 2가 됨을 볼 수 있다.
- **Sub-block DI 기법**: 본 캐시 시뮬레이터는 워드당 inversion status 비트를 하나 할당하는 기본적인 DI 기법 외에도 하나의 워드에 2개 이상의 inversion status 비트를 할당하여 추가로 HD 값을 줄일 수 있는 sub-block DI 모드도 지원하도록 구현되었다. 그림 4의 경우 inversion status 비트를 하나만 할당할 경우 HD 값이 4여서 inversion을 적용할 수 없음을 볼 수 있는데, 4비트마다 inversion status 비트를 할당하는 경우 추가의 NVM 셀을 사용하는 오버헤드는 발생하지만 HD 값을 1까지 줄일 수 있음을 보여준다.
- **비트라인 시프팅 (bit-line shifting) 기법**: NVM 캐시는 특정 NVM 셀에 데이터가 집중적으로 쓰여지는 현상으로 인한 수명 저하를 방지하기 위해 웨어레벨링 기법을 적용하게 되는데, 본 NVM 캐시 시뮬레이터는 각각의 캐시 블록에 대한 웨어레벨링 효과를 얻기 위해 [18]에서 제안한 비트라인 시프팅 기법을 기본적으로 지원한다. 비트라인 시프팅 기법은 캐시 블록마다 가해진 쓰기 작업의 회수를 세는 카운터를 두고 카운터의 값이 특정 값에 도달할 때마다 캐시 블록의 데이터 비트를 한칸씩 옆으로 밀어서 회전시키는 구조로 되어 있다. 그림 1(a)의 NVM 객체는 이를 표현하기 위해 shift offset 값을 변수로 두고 쓰기 작업이 일정 회수만큼 누적될 때마다 shift offset 값을 1씩 증가시키도록 구현되었다.
- **워드라인 리매핑 (word-line remapping) 기법**: NVM 캐시에서는 비트라인 시프팅 기법이 적용되어 각 캐시 블록 별로 쓰기 회수가 균일해진 이후에도 특정 캐시 블록에 쓰기 작업이 집중되어 “hot cache block”이 발생하게 되며, 해당 캐시 블록은 다른 캐시 블록보다 수명이 짧아지게 된다. [18]에서는 이를 해결하기 위해 특정 시간마다 캐시 블록의 주소 사상을 변경시키는 워드라인 리매핑 기법을 제안하였으며, 본 캐시 시뮬레이터는 워드라인 리매핑 기법을 그림 1(c)의 NVM 캐시 객체 수준에서 구현하였다.

Ⅳ. 사례 연구

1. PRAM 캐시

본 연구팀은 NVM 캐시 시뮬레이터를 사용하여 그림 5와 같은 구조의 PRAM 캐시[18]를 구현하고 3장에서 소개된 바와 같이 PRAM의 쓰기 트래픽 양을 절감할 수 있는 최적화 기법인 RBW 기법 및 DI 기법의 성능을 평가하였다. 또한 쓰기 트래픽 양을 줄인 이후에도 각 셀별로 쓰기 횟수가 불균등하게 분포하는 문제를 완화하기 위하여 비트라인 시프팅 기법 및 워드라인 리매핑 기법을 적용하고 그 효과를 검증하였다. 또한 그림 6과 같이 NVM 캐시 시뮬레이터의 heat-map 기능을 통해 실제로 웨어레벨링 기법이 어떻게 동작하는지를 시각적으로 확인할 수 있었다. [18]에서 보고된 바에 의하면 NVM 캐시 최적화 기법을 적용하지 않은 기본 상

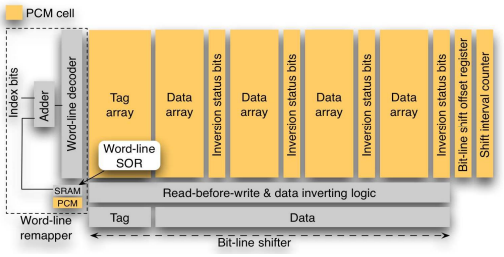


그림 5. PRAM 캐시 구조도 [18]
Fig. 5 PRAM cache structure [18]

태의 PRAM 캐시에서 약 1시간 정도로 측정된 예상 수명을 최적화 기법의 적용을 통해 최대 4년까지 연장시킬 수 있음이 확인되었다.

2. PRAM+STT-RAM 하이브리드 캐시

본 연구에서 구현한 NVM 캐시 시뮬레이터의 확장성 및 프로토타이핑의 용이성을 평가하기 위해 그림 7과 같이 PRAM과 STT-RAM을 결합한 하이브리드 캐시[20]를 구현하였다. STT-RAM은 PRAM과 비교하여 최대 쓰기 횟수가 1000배 이상 큰 장점을 가지고 있지만 셀 크기가 PRAM의 4배 정도로 크다. 따라서 다수의 PRAM 캐시 블록으로 구성된 캐시 세트마다 하나의 STT-RAM 쓰기 버퍼를 할당하는 하이브리드 구조를 채택하여 PRAM과 STT-RAM의 장점을 동시에 취할 수 있다. PRAM+STT-RAM 하이브리드 캐시를 프로토타이핑하기 위해 NVM 캐시 시뮬레이터를 다음과 같이 설정하였다.

- NVM 캐시 시뮬레이터의 기본 NVM 클래스를 상속받아 PRAM과 STT-RAM 파티션을 구성.
- PRAM과 STT-RAM 파티션 각각에 대해 성능, 에너지, 면적, 내구성 모델 파라미터를 수정.
- 하이브리드 캐시의 파티션 별 쓰기 트래픽을 계산하는 통계 함수 추가 구현.

NVM 캐시 시뮬레이터의 객체지향 기반 설계로 인해 위의 일련의 작업을 빠르게 수행할 수 있었고 NVM 캐시 시뮬레이터가 신속한 프로토타이핑을 지원함을 확인할 수 있었다.

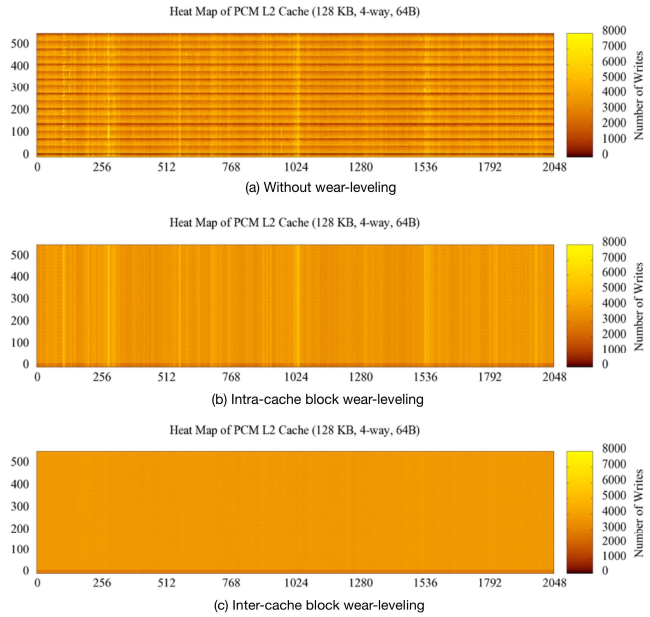


그림 6. NVM 캐시 시뮬레이터의 heat-map 기능. x축: 캐시 블록 주소 (단위: 512 bits), y축: 각 캐시 블록 내부의 메모리 셀 주소 (단위: bit)
Fig. 6 Heat-map feature of the NVM cache simulator. x-axis: cache block address (unit: 512 bits), y-axis: memory cell address in each cache block (unit: bit)

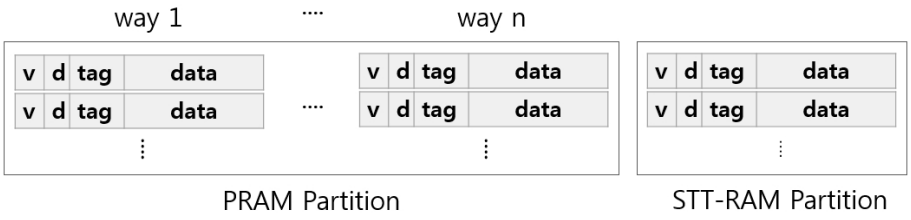


그림 7. 하이브리드 캐시 라인 구조 [20]
Fig. 7. Cache line structure of the PRAM+STT-RAM hybrid cache [20]

IV. 결 론

본 논문에서는 NVM 기반 캐시의 쓰기 성능, 전력 및 내구성 향상을 위한 다양한 최적화 기법을 구현하고 평가하기 위해 메모리 셀 단위로 정확한 쓰기 회수를 측정 가능한 NVM 기반 캐시 시뮬레이터의 설계를 제시하였다. 또한 기존의 캐시 시뮬레이터와 비교하여 NVM 기반 캐시 시뮬레이터에서 제공되어야 하는 주요 기능을 논의하였다. PRAM 캐시 및 PRAM+STT-RAM 하이브리드 캐시 프로토타입 구현 사례를 통해 제안된 캐시 시뮬레이터가 NVM 기반 캐시의 쓰기 최적화 기법을 평가하는데 적합함을 보였다.

본 논문에서 제안한 캐시 시뮬레이터는 메모리 셀 단위의 쓰기 회수 추적 기능이 필수적인 NVM 기반 캐시 최적화 기법에 대한 신속한 프로토타이핑과 성능평가를 지원함으로써 관련 연구자들이 이를 사용하여 NVM 기반 캐시의 성능, 에너지뿐만 아니라 수명 향상 연구를 활성화하는 데 기여할 수 있을 것으로 예상된다.

제안한 캐시 시뮬레이터는 메모리 셀 단위의 쓰기 회수 측정을 위해 필연적으로 시뮬레이션 속도를 희생하는 한계를 가지고 있다. 이를 보완하기 위해 메모리 셀 단위의 쓰기 회수 측정 기능을 선택적으로 활성화하는 기능이 구현되어 있으나, NVM 기반 캐시의 쓰기 최적화 기법을 평가하기 위해서는 해당 기능의 사용이 필수적이다. 따라서 해당 기능이 활성화된 상태에서의 시뮬레이션 성능을 개선하는 연구를 계속 수행할 예정이다.

References

[1] M.H. Kim, I.K. Han, Y. Joo, S.S. Lim, "Design and Implementation of Nonvolatile Memory-Based Cache Simulators," Proceedings of Korea Computer Congress, Vol. 42, No. 1, pp. 1498-1500, 2015 (in Korean).

[2] Y. Zhang, S.B. Kim, J.P. McVittie, H. Jagannathan, J.B. Ratchford, C.E.D. Chidsey, Y. Nishi, H.S.P. Wong, "An integrated phase change memory cell with Ge nanowire diode for cross-point memory," Proceedings of IEEE Symposium on VLSI Technology, pp. 98-99, 2007.

[3] S. Chung, K.M. Rho, H.J. Suh, D.J. Kim, H.J. Kim, S.H. Lee, J.H. Park, H.M. Hwang, S.M. Hwang, J.Y. Lee, Y.B. An, J.U. Yi, Y.H. Seo, D.H. Jung, M.S. Lee, S.H. Cho, J.N. Kim, G.J. Park, G. Jin, A.D. Smith, V. Nikitin, A. Ong, X. Tang, Y. Kim, J.S. Rho, S.K. Park, S.W. Chung, J.G. Jeong, S.J. Hong, "Fully Intergrated 54nm STT-RAM with the smallest bit cell dimension for high density memory application," Proceedings of IEEE International Electron Devices Meeting, pp. 12.7.1-12.7.4, 2010.

[4] A. Kawahara, R. Azuma, Y. Ikeda, K. Kawai, Y. Katoh, Y. Hayakawa, K. Tsuji, S. Yoneda, A. Himeno, K. Shimakawa, T. Takagi, T. Mikawa, K. Aono, "An 8 Mb Multi-Layered Cross-Point ReRAM Macro With 443 MB/s Write Throughput," IEEE Journal of Solid-State Circuits, Vol. 48, No. 1, pp. 178-185, 2013.

[5] J. Yue, Y. Zhu, "Accelerating write by exploiting PCM asymmetries," Proceedings of IEEE 19th International Symposium on High Performance Computer Architecture, pp. 282-293, 2013.

[6] Y. Wang, Y. Han, L. Zhang, H. Li, X. Li, "ProPRAM: exploiting the transparent logic resources in non-volatile memory for near data computing," Proceedings of the 52nd Annual Design Automation Conference, No. 47 pp. 1-6, 2015.

[7] R. Maddah, S. Mohammad, Seyedzadeh, R. Melhem, "CAFO: Cost aware flip optimization for asymmetric memories," Proceedings of IEEE 21st International Symposium on High Performance Computer Architecture, pp. 320-330, 2015.

[8] J. H. Lee, "PCM Main Memory for Low Power Embedded System," IEMEK J. Embed. Sys. Appl., Vol. 10, No. 6, pp. 391-397, 2015 (in Korean).

[9] Y. Kim, B. Tauras, A. Gupta, B. Urgaonkar, "FlashSim: A Simulator for NAND Flash-Based Solid-State Drives," Proceedings of IEEE 1st International Conference on Advances in System Simulation, pp. 125-131, 2009.

[10] V. Prabhakaran, T. Wobber, "SSD Extension for DiskSim Simulation Environment," Microsoft Resarch, Available: http://research.microsoft.com/en-us/downloads/b41019e2-1d2b-44d8-b512-ba35ab814cd4

[11] J. Gibson, R Kunz, D. Ofelt, M. Horowitz, J. Hennessy, M. Heinrich, "FLASH vs. (simulated) FLASH: closing the simulation loop," Proceedings of the 9th international conference on Architectural support for programming languages and operating systems, pp. 49-58, 2000.

[12] S.A. Herrod, "Using Complete Machine Simulation to Understand Computer System Behavior," Ph.D. thesis, 1998.

[13] P.S. Magnusson, B. Werner, "Efficient Memory Simulation in SimICS," Proceedings of the 28th Annual Simulation Symposium, pp. 62-73, 1995.

[14] T. Austin, E. Larson, D. Ernst, "SimpleScalar: an infrastructure for computer system modeling," Proceedings of IEEE Computer, Vol. 35, No. 2, pp. 59-67, 2002.

[15] J. Edler, M.D. Hill, Dinero IV Trace-Driven Uniprocessor Cache Simulator. Available: http://pages.cs.wisc.edu/~markhill/DineroIV

[16] Cachegrind: a cache and branch-prediction profiler Available: http://valgrind.org/docs/manual/cg-manual.html

[17] M. Martonosi, A. Gupta, T. Anderson, "Tuning memory performance of sequential and parallel programs," Proceedings of IEEE Computer, Vol. 28, No. 4, pp. 32-40, 1995.

[18] Y. Joo, D. Jiu, X. Dong, G. Sun, N. Chang, Y. Xie, "Energy- and endurance-aware design of phase change memory caches," Proceedings of the Conference on Degign, Automation and Test in Europe, pp. 136-141, 2010.

[19] S. Cho, H. Lee, "Flip-N-Write: A Simple Deterministic Technique to Improve PRAM Write Performance, Energy and Endurance," Proceedings of IEEE 42nd Annual IEEE/ACM International Symposium on Microarchitecture, pp. 347-357, 2009.

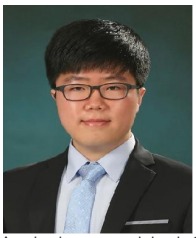
[20] Y. Joo, S. Park, "A Hybrid PRAM and STT-RAM Cache Architecture for Extending the Lifetime of PRAM Caches," IEEE Computer Architecture Letters, Vol. 12 No. 2, pp. 55-58, 2013.

Yongsoo Joo (주 용 수)

He received the Ph.D. degree in School of Computer Science and Engineering from Seoul National University, Seoul, Korea, in 2007. He is an assistant professor in the School of Computer Science in Kookmin University, Korea. His research interests include embedded systems and memory systems.

Email: ysjoo@kookmin.ac.kr

Myeung-Heo Kim (김 명 희)



He received the M.S. degree in Computer Science from Kookmin University, Seoul, Korea, in 2016. He is currently a SW architect in Hansol Technics Co., Ltd. His research interests include emedded SW, virtualization, and cache memorty systems.

Email: kim6515516@gmail.com

In-Kyu Han (한 인 규)

He received the M.S. degree in Computer Science from Kookmin University, Seoul, Korea, in 2015. He is currently a Ph.D. candidate in Computer Science at Kookmin University, Seoul, korea. His research interests include Internet of Things (IOT) and virtualization systems.

Email: in1004kyu@gmail.com

Sung-Soo Lim (임 성 수)

He received the Ph.D. degree in Electrical and Computer Engineering from Seoul National University, Seoul, Korea, in 2002. He is a professor in the School of Computer Science in Kookmin University, Korea. His research interests include virtualization, real-time systems, and mobile power management.

Email: sslim@kookmin.ac.kr