

<https://doi.org/10.7236/IIIBC.2017.17.1.219>

IIIBC 2017-1-29

# 마이크로프로세서의 성능에 끼치는 DRAM의 영향에 관한 연구

## A Study in the Effects of DRAM on The Microprocessor Performance

이종복\*

Jongbok Lee\*

**요약** 최근에 이르러, 임베디드시스템, 이동단말기 뿐만이 아니라 고성능 마이크로프로세서 및 멀티코어프로세서에서 DRAM에 대한 중요성이 날로 증가되고 있다. 이에 발맞추어 산업계와 학계에서 미래의 DRAM에 대한 활발한 연구가 진행되고 있다. 따라서, 모의실험을 통하여 마이크로프로세서의 성능을 평가할 때 보다 정확한 DRAM 모델을 갖추는 것이 중요하다. 본 논문에서는 DRAM 시뮬레이터와 연동할 수 있는 명령어 자취형 (trace-driven) 마이크로프로세서 모의실험기를 개발하였다. 또한, SPEC 2000 벤치마크를 입력으로 모의실험을 수행하여, 사이클 단위로 정확하게 동작하는 DD3 모델이 마이크로프로세서의 성능에 끼치는 영향을 분석하였다.

**Abstract** Recently, the importance of DRAM is very significant not only in embedded systems and mobile devices but also in high-end modern microprocessors and multicore processors. To keep up with this, both industry and academia have actively studied various types of future DRAMs. Therefore, accurate DRAM model is requisite when evaluating the microprocessor performance. In this paper, a microprocessor trace-driven simulator which can couple with the cycle-accurate DRAM simulator has been developed. Using SPEC 2000 benchmarks as input, the effect of cycle-accurate DDR3 model on the microprocessor performance has been evaluated.

**Key Words** : DRAM, DDR3, Performance

### 1. 서 론

DRAM은 컴퓨터시스템에서 메인메모리를 구성하는 반도체 기억소자로서, 임베디드시스템, 이동단말기 뿐만이 아니라 고성능 마이크로프로세서 및 멀티코어프로세서의 성능에 큰 영향을 미친다. 따라서 과거에는 물론이고, 현재에서도 산업계와 학계에서 미래의 DRAM에 대

한 활발한 연구가 진행되고 있다<sup>[1-3]</sup>.

마이크로프로세서의 성능을 측정하기 위한 모의실험에서, 캐쉬 미스가 발생하여 DRAM을 접근하는 경우에 발생하는 손실을 고정된 사이클 수로 가정한다면 정확도가 낮아진다. 이 때, 실제로 읽기 및 쓰기 요청이 공급되어 DRAM에서 일어나는 상황을 사이클 단위로 모의실험할 수 있는 정확한 DRAM 모델을 이용한다면, 마이크로

\*정희원, 한성대학교 전자정보공학과  
접수일자 : 2016년 11월 1일, 수정완료 : 2016년 12월 25일  
게재확정일자 : 2017년 2월 3일

Received: 1 November, 2016 / Revised: 25 December, 2016 /  
Accepted: 3 February, 2017

\*Corresponding Author: jblee@hansung.ac.kr  
Dept. of Electronics & Information Engineering, Hansung  
University, Korea

프로세서의 성능을 보다 정밀하게 평가할 수 있다. 본 논문에서는 사이클 단위로 동작하는 DRAM 시뮬레이터와 연동할 수 있는 명령어 자취형 마이크로프로세서 모의실험기를 개발하였다. 또한, SPEC 2000 벤치마크를 입력으로 모의실험을 수행하여, 사이클 단위로 정확하게 동작하는 DDR3가 마이크로 프로세서의 성능에 끼치는 영향을 분석하였다.

본 논문은 다음과 같이 구성된다. 2 장에서 DRAM의 모델과 마이크로프로세서 모의실험기에 대하여 살펴보고, 3 장에서 모의실험 환경에 대하여 고찰한다. 4 장에서 모의실험 결과를 보이며, 5 장에서 결론을 맺는다.

## II. DRAM의 모델 및 모의실험기

### 1. DRAM의 모델

DRAM을 모의실험하기 위하여 모델링을 할 때, 유한상태기계 (finite state machine)로 표시할 수 있다. 이러한 유한상태기계에서 여러 가지 DRAM의 상태를 노드로 표현하며, 외부입력에 의하여 한 상태에서 다른 상태로 전이가 일어난다. DDR3를 기준으로 했을 때, DRAM은 그림 1에 나타난 것과 같이 채널 (channel), 랭크 (rank), 뱅크 (bank), 행 (row) 및 열 (column)로 구성된다.

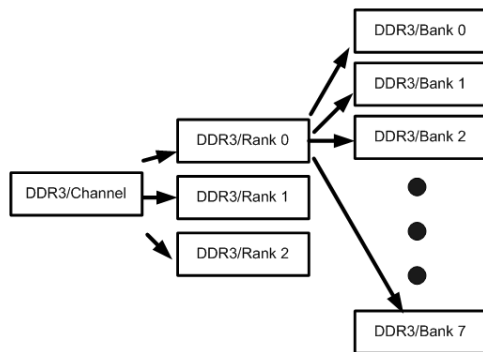


그림 1. DDR3 유한상태기계의 트리구조

Fig. 1. The tree structure of DDR3 state-machines

이 때, DRAM의 채널은 세 개의 랭크로 구성되며, 각 랭크는 여덟 개의 뱅크로 연결되는 계층적 구조를 가진다. 메모리 컨트롤러는 루트노드인 채널을 통해서 DRAM을 제어한다. 각 노드가 나타내는 DRAM의 상태 (status)는 열림(open)과 닫힘(closed)으로 표현되며, 활

성화 (activate), 선충전 (pre-charge), 읽기 (read), 쓰기 (write)에 각각 해당하는 ACT, PRE, RD, WR 명령어에 의해서 한 상태에서 다른 상태로 전이할 수 있다. DRAM을 나타내는 또 한 가지의 상태인 수평 상태(horizon)는 각 노드가 다음 명령어를 가장 빨리 받을 수 있는 시간을 알아낼 수 있는 참조가능한 테이블을 의미하며, 이것은 임의의 노드가 DDR3의 타이밍 파라미터를 준수하여 상태를 전이할 수 있도록 한다. 한편, 각 노드에는 해독 (decode), 검사 (check), 수정 (update)의 세 가지 기능을 적용할 수 있으며, 루트 노드인 채널에 인가했을 때 유한상태도로 구성된 트리의 하위계층으로 진행이 가능하다.

### 2. DRAM을 구성하는 유한상태기계의 개요

DRAM에 읽기 요청이 들어왔을 때, 메모리 컨트롤러는 해독, 검사, 수정의 세 가지 기능을 이용하여 요청에 응답한다. 읽기 요청의 최종 목표는 읽기 명령어에 의하여 DRAM으로부터 데이터를 읽어내는 것이다. 그러나, 랭크에 전원이 공급되지 않거나 뱅크가 닫혔을 때는 읽기 명령어를 수행할 수 없다. 따라서, 이러한 명령어를 수행하기 전에 주어진 어드레스에 대한 명령어가 발행 가능할지를 검사해야한다. 해독기능은 전원을 켜는 명령어나 활성화 명령어를 발행하기 전에 필요조건 명령어를 리턴한다. 이 때, 필요조건 명령어가 리턴되지 않았더라도, 뱅크가 최근에 활성화되었다면 읽기 명령어를 즉시 발행할 수 없다. 따라서 이 명령어는 주어진 어드레스와 주어진 명령어에 대하여 명령어가 현재의 사이클에 즉시 발행될 수 있는지 검사한다. 이러한 검사를 통과한 후에는 메모리 컨트롤러는 그 어떤 방해도 받지 않고 읽기 명령어를 발행할 수 있다.

### 3. DRAM을 구성하는 유한상태기계의 동작원리

DRAM의 표준 사양을 준수하기 위하여, 어떤 단계 및 상태에서 한 명령어가 다른 명령어에 선행해야하는가를 결정하기 위한 필요조건이 필요하다. 그리고, 각 단계 및 상태에서 명령어와 명령어 간에 어떤 타이밍 파라미터가 적용되어야 하는 가를 결정하는 것이 중요하다. 마지막으로, 각 단계에서의 어떤 명령어가 어느 상태로의 전이를 촉발시키는지 알아야한다. 따라서 위 세 가지 경우인 필요조건, 타이밍, 전이에 대한 세 가지 참조테이블을 제공하는 기능이 필요하다.

명령어의 해독 기능 내에서 필요조건 기능은, 해독 단

계와 해독 중인 명령어를 이용한다. 예를 들어서, 랭크 단계에서 해독하려는 명령어가 리프레쉬인 경우가 이에 해당한다. 이 때, 어떤 뱅크가 열려있다면, 전체 충전 명령어를 통하여 모든 뱅크를 닫고 이후의 리프레쉬 명령어를 가능하게 한다. 그러나, 어떤 뱅크가 열려있지 않다면 다른 명령어를 발행하지 않고 단지 리프레쉬 명령어를 리턴한다. 그 어떤 경우에도 명령어는 성공적으로 해독되며, 이 때는 DRAM의 유한상태도 트리를 더 이상 탐색할 필요가 없다. 한편, 랭크 단계에서 리프레쉬 명령이 발행된 이유는 상위 단계인 채널이 정보가 부족하여 하위 단계의 함수를 해독하기 위함이다. 어떤 명령어가 특정 단계에서 해독이 불가능하다면 유한상태도를 재귀적으로 탐색하여 명령어가 해독되도록 한다.

한편, 필요조건 기능에 추가하여, DDR3 모델에서 전이기능과 타이밍기능에 대한 테이블참조가 제공된다. 각 기능은 DRAM의 상태와 타이밍 파라미터를 인코딩하는데 쓰인다. 필요조건 기능과 마찬가지로, 전이기능과 타이밍기능은 단계, 상태, 명령어에 의하여 결정된다. 명령어가 발행되었을 때, 업데이트 기능은 테이블참조를 통하여 영향을 받는 유한상태도 트리의 모든 노드의 상태를 변경한다. 그러나 검사 기능은 DDR3 차원의 테이블참조를 하지 않고 DRAM 내부에 내장된 지역 참조테이블만을 조회하여, 한 명령어에 의하여 영향을 받는 모든 노드에 대하여 조건과 일치하는지 확인한다. 이렇게 함으로써, 현재 싸이클이 명령어를 발행할 수 있는 가장 이른 시점인지를 검증할 수 있다. 그러므로, 검사기능은 업데이트기능에 의하여 참조테이블을 최신 정보로 유지한다<sup>[4]</sup>.

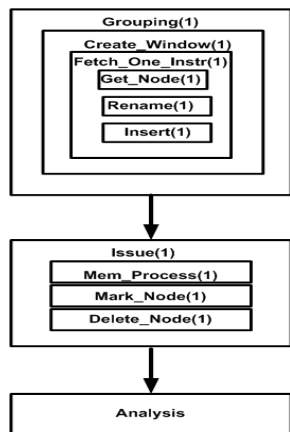


그림 2. 마이크로프로세서 모의실험기  
 Fig. 2. The microprocessor simulator

#### 4. 프로세서 모의실험기

프로세서는 제 1 단계 명령어 자취의 발생, 제 2 단계 명령어자취에 대한 프로세서의 실행으로 나누어진다. 제 1 단계에서 명령어 자취는 특정 하드웨어와 상관없이 소프트웨어의 특성에 따라 발생되었다.

제 2 단계 프로세서 실행은 다음과 같다. 본 모의실험기는 코어가 슈퍼스칼라 프로세서로 동작할 수 있도록 프로그래밍 되었다. 따라서 한 싸이클에 2 개 이상의 명령어를 인출받을 수 있다. 제 2 단계의 과정을 자세하게 기술하면 그림 2에 나타낸 것과 같으며, 설명하면 다음과 같다.

##### (1) 명령어 인출 및 재명명

Initialize 함수에서 초기화 작업을 거친 후에, Grouping 함수가 Create\_Window 함수를 부르고 이것은 다시 Fetch\_One\_Instr 함수를 불러서 각 코어는 매 싸이클마다 새로운 명령어를 인출받는다. 한 싸이클에 1 개의 명령어를 인출받을 때와 달리, 2 개 이상의 명령어를 인출받을 때, 분기명령어를 만나면 그 이상의 인출을 하지 않고 인출을 멈춘다.

Get\_Node 함수에서 인출한 명령어는 Rename 함수에서 재명명 (renaming) 작업을 거치면서 명령어 종속에 의한 타임스탬프(timestamp) 값을 설정받는다. 타임스탬프 방식은 명령어 자취를 이용하는 모의실험에서 데이터 종속성을 신속하고 효율적으로 부여할 수 있는 핵심적인 방법이다.

모든 명령어는 인출 초기에 현재 싸이클에 명령어 자체의 지연 싸이클을 더한 값을 타임스탬프로 설정 받는다. 이후의 디코드 및 재명명 단계에서, 각 명령어의 소오스 레지스터 1과 소오스 레지스터 2에 대하여 레지스터 화일을 검색하여 같은 이름을 갖는 레지스터를 찾아서 해당하는 타임스탬프 값을 읽는다. 만일 현재 명령어의 타임스탬프 값보다 소오스 레지스터 1 또는 2의 타임스탬프 값이 더 크다면, 명령어의 타임스탬프는 그 값으로 대체된다. 또한, 명령어의 목적레지스터의 타임스탬프도 같은 값으로 대체되어, 추후에 이 목적 레지스터를 소오스 레지스터로 참조하는 명령어에 대하여 올바른 타임스탬프 값을 가질 수 있도록 한다. 레지스터 화일의 타임스탬프 값에 의하여, 프로세서 명령어 간의 종속성이 유지되어 성능을 구하는데 반영된다. 이와 같이 재명명을 거친 명령어는 insert 함수에서 각 코어의 명령어 윈도우에 삽입된다.

## (2) 명령어 이슈

Issue 함수로 진행하면, 사이클이 증가함에 따라서 윈도우 내의 명령어는 자체의 타임스탬프 값이 현재 사이클 보다 작거나 같을 때 삭제될 수 있다. 그러나, 첫 단계인 Mark\_Node 함수에서 즉각 삭제하지 않고 삭제 가능 표시만 하며, 다음 단계인 Delete\_Node 함수에서 삭제 가능 표시를 한 명령어를 실제로 삭제한다. 이 과정에서 만일 삭제 가능 표시를 하지 않은 명령어를 만나면 그 이후의 명령어는 삭제를 즉각 종료한다. 이렇게 함으로써, 코어가 비순차실행(out-of-order execution) 방식의 슈퍼스칼라 프로세서인 경우에도 순차종료(in-order completion)를 보장할 수 있다.

## (3) 시뮬레이션

해당 코어의 윈도우 공간에 Grouping 함수를 이용하여 명령어를 인출해서 채우고, Issue 함수로 각 코어에 대하여 명령어를 실행하면서 종속성에 의하여 부여된 명령어의 타임스탬프가 충족되면 삭제한다. 위의 Issue 동작은 명령어가 삭제되어 윈도우가 빈 상태가 될 때까지 반복적으로 실행된다. 윈도우가 비어있는 상태가 되면, 다시 Grouping 함수를 통하여 윈도우를 명령어로 채운다. 이 과정은 입력으로 주어진 벤치마크 프로그램의 모든 명령어가 소진될 때까지 반복된다.

위 과정이 한번 실행될 때 마다 사이클이 증가하므로 모의실험에 입력으로 쓰인 명령어의 총 개수를 처리하기 위하여 소요된 사이클 수로 나누어, 마이크로프로세서 시스템의 IPC(Instruction Per Cycle)을 계산할 수 있다<sup>[5]</sup>.

## III. 모의실험 환경

본 논문의 모의실험은 운영체제 Fedora 22에서 3.1GHz로 동작하는 Intel Core i5-2400가 탑재된 데스크탑 PC에서 시행하였다. 표 1은 모의실험에 이용된 프로세서 아키텍처의 사양을 나타낸 것이다.

각 코어는 슈퍼스칼라 방식으로 운영되므로, 매 사이클 마다 2 개의 명령어를 인출하고 윈도우의 크기는 8이다. 각 코어의 연산유닛은 정수형 유닛, 로드 스토어 유닛, 그리고 분기명령어로 구성된다. 1 차 명령어 캐쉬와 데이터 캐쉬는 각 코어마다 설치되며, 코어의 개수가 증가할 수록, 그 용량이 증가하므로 성능에 유리하다. 따라서 공정한 성능 비교를 위하여, 단일코어 프로세서의 명

령어 캐쉬와 데이터 캐쉬는 64 KB의 용량을 갖도록 설정하였다. 1 차 명령어 캐쉬는 2 차 연관도(set associativity)를 가지나, 1 차 데이터 캐쉬는 직접 매핑을 통하여 접근된다. 분기 명령어는 2 단계 적응형 분기 예측 방식을 적용하였다. 표 2는 모의실험에 이용된 여덟 개의 SPEC 2000 정수형 벤치마크 프로그램이다. SimpleScalar를 통하여 MIPS IV 5천만 개의 명령어 자취를 모의실험에 적합하도록 발생시켰다<sup>[6]</sup>.

표 1. 프로세서 아키텍처 하드웨어의 사양

Table 1. The architecture specification of each core

항목	값
1 차 명령어 캐쉬	64KB
1 차 데이터 캐쉬	64KB
1차 명령어 캐쉬 공통 사항	2 차 연관, 16 B 미스 페널티 10 사이클
1차 데이터 캐쉬 공통 사항	직접, 32 B 미스 페널티 10 사이클
명령어 윈도우의 크기	8
인출율, 이슈율, 퇴거율	2
연산유닛 사양	산술논리(2), 로드 & 스토어(1), 분기(1)
분기 어드레스 캐쉬	1 K 엔트리
분기 예측기	8 비트 전역 히스토리 방식 미스 페널티 6 사이클
이슈 지연 사이클	산술논리(1), 분기(1), 로드(1), 스토어(1),
결과 지연 사이클	산술논리(1), 분기(1), 로드(1), 스토어(1),

표 2. SPEC 2000 정수형 벤치마크 프로그램

Table 2. SPEC 2000 integer benchmark programs

벤치마크	설 명
bzip2	압축
crafty	체스 경기 놀이
gap	그룹 이론 해석기
gcc	C 프로그래밍 언어 컴파일러
gzip	압축
mcf	조합 최적화
paser	워드 프로세서
twolf	배선 및 배치 모의실험기

그림 3은 본 모의실험의 전체 흐름도를 나타낸 것으로서, 크게 세가지 단계로 나뉜다. 첫 단계에서, 초기 DRAM 프로파일러를 통하여 DRAM에 접근하는 모든 명령어와 데이터의 어드레스와 읽기/쓰기 여부를 구분하여 화일로 저장한다. 마이크로프로세서에서 DRAM을 접근하는 경우는 명령어를 인출하거나, 데이터를 로드 및 스토어할 때 각각 명령어 캐쉬와 데이터 캐쉬에서 캐쉬 미스가 발생하는 경우이다. 두 번째 단계에서 DRAM 시뮬레이터가 위 화일을 입력으로 받아서 DRAM 내부에서

처리하기 위한 사이클 수를 계산한다. 본 모의실험에서 설정한 DRAM은 DDR3이며, 이것은 JEDEC 표준에 따라서 1600 MT/s의 전송률, 11-11-11 타이밍, 11.9 GB/s의 대역폭을 갖는다<sup>[7]</sup>. 마지막인 세 번째 단계에서, DRAM 시뮬레이터로부터 계산한 DRAM 소요 사이클 수를 입력으로 마이크로프로세서의 성능을 계산한다.

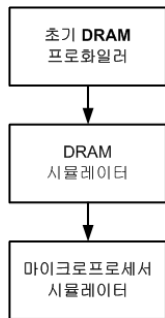


그림 3. 모의실험의 흐름도  
 Fig. 3. The simulation flow

#### IV. 모의실험 및 결과

그림 4에 초기 DRAM 프로파일러에 의하여 생성되어 DRAM 시뮬레이터에 입력으로 공급되는 입력파일의 예를 나타냈다. 이 파일은 5 천만 개의 명령어 중에서 명령어 인출을 위한 어드레스나 로드/스토어 명령어로 인하여 DRAM에 접근하는 데이터 어드레스를 16진수로 표현한 값과, 읽기 또는 쓰기에 따라 각각 R, W로 구분된 양식의 쌍으로 표현된다.

```

0x41b7c R
0xffffd7bfc W
0x41b7d R
0x41b7e R
0x41b7f R
0xffffd7c00 W
    
```

그림 4. DRAM 시뮬레이터에 공급되는 입력파일의 예  
 Fig. 4. The example input file for the DRAM simulator

표 3은 모의실험에서 마이크로 프로세서의 명령어를 실행하기 위하여 소요되는 사이클 수와, DRAM에 접근하는데 소요된 사이클 수를 각 벤치마크에 대하여 나타낸 것이다. *Gap*과 *gcc*는 DRAM에서 소비되는 사이클 수가 명령어를 실행하는데 드는 사이클 수보다 오히려 크

다. 따라서, 전체 성능이 매우 낮을 것으로 예측할 수 있다. 그리고, *crafty*는 DRAM에서 소비되는 사이클 수가 명령어 처리 사이클 수의 58 %에 육박하므로 중간 정도의 손실을 가져오게 된다. 반면에, *bzip2*, *gzip*, *mcf*, *parser*는 DRAM에서의 소비되는 사이클 수가 명령어가 처리되는데 필요한 사이클 수보다 훨씬 적어서 DRAM을 접근함으로써 발생하는 성능의 감소가 거의 없다. 따라서, 이 경우는 각 벤치마크 프로그램 고유의 명령어 수준 병렬성에 의하여 전체 성능이 좌우된다.

표 3. 모의실험 총 사이클 수와 DRAM 소비 사이클 수의 비교  
 Table 3. The comparison number of simulated cycles vs. DRAM active cycles

벤치마크	명령어 처리 소요 사이클 수	DRAM 소비 사이클 수
bzip2	36,864,225	5,621
crafty	14,560,224	8,457,345
gap	26,971,987	98,518,767
gcc	28,756,971	172,353,744
gzip	27,335,975	650
mcf	32,224,007	129,782
parser	28,445,264	510
twolf	29,228,897	2,576,279

그림 5에 여덟 개의 SPEC 벤치마크에 대하여 DRAM의 동작을 포함한 모의실험을 수행하여 성능을 IPC로 나타냈다.

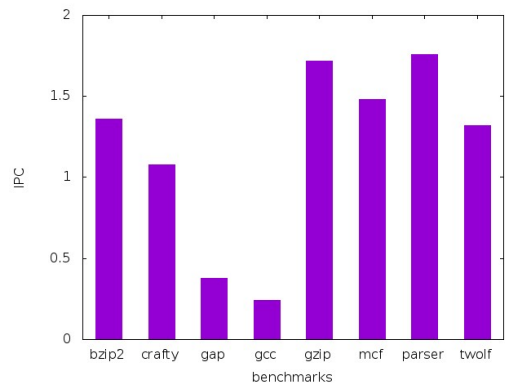


그림 5. DDR3 모델을 적용한 마이크로프로세서의 SPEC 2000 벤치마크에 대한 모의실험 결과

Fig. 5. The simulation results of SPEC 2000 benchmarks reflecting the DDR3 model on a microprocessor

*Gap*과 *gcc*는 DRAM을 접근하는데 성능이 80 % 이상 감소하여, 가장 낮은 수준인 0.38 IPC이하를 얻는데 그쳤

다. *Twolf*는 DRAM으로 인하여 23 %의 성능 손실로 중간에 해당하는 1.32 IPC를 기록하였으며, *parser*는 명령어 수준 병렬성이 상대적으로 높고 DRAM 접근에 의한 성능이 거의 없어 가장 높은 1.76 IPC를 나타냈다. 모의 실험 결과에서 알 수 있듯이, 명령어 수준 병렬성과 더불어 DRAM을 접근하는데 얼마나 많은 사이클을 소비하는가에 따라서 각 벤치마크의 전체 성능이 결정됨을 알 수 있다. 결과적으로, 각 벤치마크 프로그램 별로 최저 0.24 IPC에서 최고 1.76 IPC 범위를 나타내고 평균 0.97 IPC의 성능을 기록하였다.

## V. 결 론

본 논문에서는 사이클 단위로 정확하게 동작하는 DRAM 시뮬레이터와 연동하여 마이크로프로세서의 성능을 측정할 수 있는 명령어 자취 모의실험기를 개발하였다. 이것을 위하여, 초기 DRAM 프로파일러에 의하여 얻은 데이터를 DDR3를 모델링한 DRAM 시뮬레이터에 입력하여 DRAM에서 소요되는 사이클 수를 획득하였으며, 그 결과를 마이크로프로세서에 입력하여 SPEC 2000 벤치마크의 성능을 측정하였다.

추후로, DDR3 이외에 DDR4, SALP, LPDDR3, LPDDR4, GDDR5 등의 기타 DRAM을 이용하는 마이크로프로세서의 성능을 평가하고 비교하는 연구가 필요하다. 나아가, DRAM 시뮬레이터와 연동하여 성능을 측정하는 본 연구는 일반적인 멀티코어 프로세서, 임베디드 멀티코어 프로세서 및 디지털 신호처리 멀티코어 프로세서에도 확대가 가능하다.

## References

- [1] P. Rosenfeld et al. "DRAMSim2: A Cycle Accurate Memory System Simulator," IEEE Computer Architecture Letters, 2011.  
DOI: <https://doi.org/10.1109/L-CA.2011.4>
- [2] Y. Kim et al. "A Case for Exploiting Subarray-Level Parallelism (SALP) in DRAM,"

ISCA, 2012.

- [3] D. Lee et al. "Tiered-Latency DRAM: A Low Latency and Low Cost DRAM Architecture," HPCA, 2013.
- [4] Y. Kim, W. Yang, and O. Mutlu, "Ramulator: A Fast and Extensible DRAM Simulator," IEEE Computer Architecture Letters, 2015.
- [5] J. Lee, "A Study of Trace-driven Simulation for Multi-core Processor Architectures," Journal of The Institute of Internet, Broadcasting and Communication, vol. 12, no. 3, pp. 9-13, Jun. 2012.
- [6] T. Austin, E. Larson, and D. Ernest, "SimpleScalar: An Infrastructure for Computer System Modeling," Computer, Vol. 35, No. 2, pp. 59-67, Feb. 2002.
- [7] JEDEC, JESD79-3 DDR3 SDRAM Standard, June 2007.

## 저자 소개

### 이 중 복(정회원)



- 1964년 8월 20일생.
- 1988년 : 서울대 컴퓨터공학과 졸업.
- 1998년 : 동 대학 전기공학부 졸업(공학박).
- 1998 ~ 2000 : LG반도체 선임연구원.
- 2000년 ~ 현재 : 한성대 전자정보공학과 교수

• Tel : 02-760-4497

• Fax : 02-760-4435

• E-Mail : jblee@hansung.ac.kr

<주관심분야 : 멀티코어 프로세서>

※ 본 연구는 한성대학교 교내학술연구비 지원과제임.