The Journal of The Institute of Internet, Broadcasting and Communication (IIBC)
Vol. 18, No. 3, pp.177-183, Jun. 30, 2018. pISSN 2289-0238, eISSN 2289-0246

https://doi.org/10.7236/JIIBC.2018.18.3.177
JIIBC 2018-3-23

디지털 신호처리 프로세서의 성능에 대한 DRAM의 영향 분석

Effects Analysis of DRAM for Digital Signal Processor Performance

이종복*

Jongbok Lee*

요 약 현재, 영상처리, 음성처리, 필터링, 등화 등의 분야에 디지털 신호처리 시스템이 광범위하게 쓰이고 있다. 더불어, 디지털 신호처리 시스템을 구성하는 디지털 신호처리 프로세서의 성능에 지대한 영향을 미치는 DRAM에 대한 연구가 산업계와 학계에서 활발하게 진행되고 있다. 따라서, 모의실험을 통하여 디지털 신호처리 프로세서의 성능에 대한 신뢰할만한 결과를 얻기 위하여, 보다 정확한 DRAM 모델을 갖추는 것이 중요하다. 본 논문에서는 싸이클 단위로 정확하게 동작하는 DRAM 시뮬레이터와 연동할 수 있는 디지털 신호처리 프로세서 모의실험기를 개발했다. 그리고 UTDSP 디지털 신호처리 벤치마크를 개발한 모의실험기에 대한 입력으로 하여, DRAM이 디지털 신호처리 프로세서의 성능에 끼치는 영향을 분석하였다.

Abstract Currently, digital signal processing systems are used extensively in image processing, audio processing, filtering, and equalizations, etc. In addition, the importance of DRAM, which has a great influence on the performance of an digital signal processor has been increased, making research on DRAM actively conducted in industry and academia. Therefore, it is important to have a more accurate DRAM model in order to obtain reliable results when evaluating the performance of a digital signal processor through simulation. In this paper, we developed a digital signal processor simulator capable of inter-working with a DRAM simulator. With the simulator, we analyzed the influence of the DRAM model which operates correctly on a cycle-by-cycle basis, on the performance of the digital signal processor by using the UTDSP digital signal benchmark.

Key Words : DRAM, digital signal processor, performance

Ⅰ. 서 론

디지털 신호처리 프로세서는 디지털 신호처리에 특화된 구조를 갖는 전문화된 마이크로프로세서로서, 음성신호, 영상신호, 전자기신호 또는 광학신호 등을 처리하는데 쓰인다. 디지털 신호처리의 목적은 실세계의 아날로그 신호를 ADC(Analog to Digital Converter)를 통하여 변환된 디지털 신호를 측정, 필터링, 압축하기 위한 것이 대부분이다. 디지털 신호처리 알고리즘은 일련의 데이터 샘플에 대하여 많은 양의 수학연산을 실시간으로 빠르고

*정회원, 한성대학교 전자정보공학과
접수일자 : 2018년 5월 1일, 수정완료 : 2018년 5월 28일
게재확정일자 : 2018년 6월 8일

Received: 1 May, 2018 / Revised: 28 May, 2018 /
Accepted: 8 June, 2018
*Corresponding Author: jblee@hansung.ac.kr
Dept. of Electronic & Information Eng., Hansung University, Korea

반복적으로 수행해야한다. 특히 최근에 이르러 그래픽 처리, 암호화, 압축 및 스마트 자동차 센서용 반도체칩을 위하여 디지털 신호처리 프로세서 성능의 고도화가 요구되고 있다 [1].

DRAM (Dynamic Random Access Memory)은 컴퓨터 시스템에서 메인 메모리를 구성하는 휘발성 반도체 기억소자로서, 고성능 마이크로프로세서 및 멀티코어프로세서 뿐만이 아니라 디지털 신호처리시스템, 이동단말기의 성능에 큰 영향을 미친다. 따라서 과거에는 물론이고, 현재에서도 산업계와 학계에서 미래의 DRAM에 대한 활발한 연구가 진행되고 있다[2]-[4]. 디지털 신호처리 프로세서의 성능을 측정하기 위하여 모의실험을 시행할 때 만일 DRAM 전용 시뮬레이터가 없으면, DRAM에 액세스할 때마다 고정된 싸이클 수를 취해야한다. 따라서, 이러한 방식으로 측정한 디지털 신호처리 프로세서의 성능은 그 정확도가 떨어진다. 만약에 읽기 및 쓰기 요청이 공급된 DRAM에서 일어나는 상황을 싸이클 단위로 시뮬레이션이 가능한 DRAM 모델을 이용한다면, 디지털 신호처리 프로세서의 성능을 보다 정확하게 평가할 수 있다.

본 논문에서는 싸이클 단위로 동작하는 DRAM 시뮬레이터와 인터페이스 할 수 있는 명령어 자취형 디지털 신호처리 프로세서 모의실험기를 개발했고, UTDSP 디지털 신호처리 벤치마크를 입력으로 모의실험을 수행하여, DDR3가 디지털 신호처리 프로세서의 성능에 끼치는 영향을 분석하였다 [5].

본 논문은 다음과 같이 구성된다. 2 장에서 DRAM의 모델을 살펴보고 3 장에서 디지털 신호처리 프로세서 모의실험기를 고찰한다. 4 장에서 모의실험 환경에 대하여 기술하고, 5 장에서 모의실험 결과를 보이며, 6 장에서 결론을 맺는다.

II. DRAM의 모델

1. DRAM의 모델

그림 1에서 볼 수 있듯이, DRAM이 프로세서로부터 읽기 및 쓰기 요청을 처리할 때, 서로 긴밀하게 동작하는 DRAM 컨트롤러, 채널, 랭크 및 뱅크로 구성된다. 또한, DRAM은 DRAM 제어기를 통하여 제어되며, 채널들을 통하여 연결된다. 각 채널은 여러 개의 랭크들로 이루어

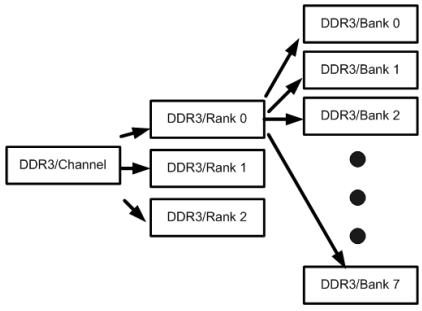

그림 1. DDR3 유한상태기계의 트리구조
Fig. 1. The Tree of DDR3 state-machines

진 DIMM 구조로 구성된다. 각 DRAM 장치는 트랜지스터와 캐패시터들로 배열된 DRAM 메모리 셀들과 행 디코더 (row decoder), 열 디코더 (column decoder), 행 버퍼 (row buffer), 센스 증폭기 (sense amplifier)로 이루어지며, 각 랭크는 8 개의 DRAM 장치로 구성된다.

본 논문에서 여러 가지 DRAM의 상태로 노드를 구성하고, 외부입력이 주어질 때 한 상태에서 다른 상태로 전이가 일어나는 유한상태기계 (finite state machine)로 DRAM의 모델을 정의하였다. 이 유한상태기계는 DRAM의 모델은 DDR3를 기준으로 했을 때, 그림 1과 같이 채널 (channel), 랭크 (rank), 뱅크 (bank), 행 (row), 열 (column)로 표현된다.

이 때, DRAM의 채널은 여덟 개의 뱅크를 갖는 계층적 구조를 갖는 세 개의 랭크로 구성되며, 각 랭크는 메모리 컨트롤러가 루트노드인 채널을 통해서 DRAM에 작용한다.

DRAM의 유한상태기계에서 각 노드가 나타내는 DRAM의 상태는 열림과 닫힘 및 수평상태의 세 가지로 표현된다. 한편, DRAM 노드는 한 상태에서 활성화, 프리차지 (precharge), 읽기, 쓰기 내부 명령어에 의해서 다른 상태로 전이한다. 열림과 닫힘 외에 DRAM을 나타내는 또 한 가지의 상태인 수평 상태는 각 노드가 다음 명령어를 가장 빨리 받을 수 있는 시간에 대한 정보를 제공함으로써, 임의의 노드가 DDR3의 타이밍 파라미터의 제약 하에서 한 상태에서 다른 상태로 전이할 수 있도록 한다. 한편, DRAM을 구성하는 각 노드에는 세부 동작을 위하여 해독, 검사, 정정의 세 가지 기능을 적용할 수 있다. 이러한 기능은 DRAM을 구성하는 유한상태기계의 루트 노드인 채널에 인가했을 때, 루트로부터 트리의 하위계층으로 전파된다.

2. DRAM을 구성하는 유한상태기계의 개요

DRAM에서 읽기 요청의 궁극적인 목표는 읽기 명령어로 DRAM에서 데이터를 읽는 것이다. DRAM의 메모리 컨트롤러는 프로세서로부터 읽기 요청이 들어왔을 때, 1절에서 기술한 것과 같이 해독, 검사, 정정의 세 가지 기능으로 응답한다. 또한, 랭크에 전원이 공급되지 않거나 뱅크가 닫혔을 때는 읽기 명령어를 수행할 수 없기 때문에 읽기 명령어를 수행하기 전에 주어진 어드레스에 대한 명령어가 발행 가능한지를 검사한다. 해독 기능은 주어진 어드레스와 주어진 명령어에 대하여 명령어가 현재의 싸이클에 즉시 발행될 수 있는지를 검사하며, 이것을 통과한 후에는 메모리 컨트롤러가 읽기 명령어를 발행할 수 있다.

3. DRAM을 구성하는 유한상태기계의 동작원리

DRAM의 표준 사양을 준수하려면, 세 가지 요소가 필요한데, 이것은 DRAM 모델에 명령어 발행 조건, 타이밍 정보, 상태 전환에 대한 정보이다. 명령어 발행 조건은, 어떤 상태에서 한 명령어가 다른 명령어보다 먼저 실행되어야 하는가를 결정하기 위한 필요조건이다. 타이밍 정보는 각 상태에서 DRAM의 내부 명령어와 명령어 간에 어떤 타이밍 파라미터가 적용되어야 하는가를 결정한다. 마지막으로, 상태 전이는 각 단계에서의 명령어가 어떤 상태로 전이되는지를 결정한다.

리프레쉬 (refresh) 명령은 임의의 뱅크가 열려있는 가의 여부에 따라 달라진다. 즉, 임의의 뱅크가 열려있다면, 전체 충전 명령어를 통하여 모든 뱅크를 닫은 이후에 리프레쉬 명령어를 수행한다. 그러나, 모든 뱅크가 닫혀있다면 추가의 명령어를 발행하지 않고 리프레쉬 명령어를 보낸다. 이 때 명령어는 성공적으로 디코드되며, DRAM의 유한상태도 트리에 대한 탐색을 멈춘다.

DRAM의 표준사양을 준수하기 위하여, DDR3 모델에서 상태의 전이기능과 타이밍기능에 대한 정보가 필요한데, 이것은 DRAM의 상태와 타이밍 파라미터를 부호화하는데 쓰인다. 명령어 발행 조건 기능과 마찬가지로, 상태의 전이기능과 타이밍기능은 단계, 상태, 명령어에 의하여 결정된다. 명령어가 발행되었을 때, 정정 기능은 전역 테이블 참조를 통하여 영향을 받는 유한상태도 트리의 모든 노드의 상태를 변경한다. 그러나, 검사 기능은 전역 테이블 참조를 하지 않고 오직 DRAM 내부에 내장된 지역 테이블만을 참조하여 한 명령어에 의하여 영향을 받는 모든 노드에 대하여 조건과 일치하는지를 확인한다. 이렇게 함으로써, DRAM 제어기가 특정 명령어를 발행할 수 있는 가장 빠른 싸이클 시점을 확인할 수 있다. 이와 같이, DRAM 제어기는 검사를 통한 업데이트 기능으로 DRAM의 정보를 기록하는 참조테이블을 최신으로 유지한다[6].

Ⅲ. 디지털 신호처리 프로세서 모의실험기

본 연구에서 개발된 디지털 신호처리 프로세서는 모의실험을 위하여 제 1 단계 명령어 자취의 발생, 제 2 단계 명령어 자취를 위한 디지털 신호처리 프로세서의 실행으로 나누어진다. 제 1 단계에서 명령어 자취는 일반적인 명령어자취 모의실험 (trace-driven simulation)에서 선행되어야 하는 작업으로서, 특정 하드웨어와 상관없이 디지털 신호처리 소프트웨어의 특성에 따라 발생되었다. 이에 대한 구체적 환경은 4 장에서 자세히 다룬다.

본 모의실험기는 디지털 신호처리 프로세서가 수퍼스칼라 방식으로도 동작할 수 있도록 한 싸이클에 2 개 이상의 명령어를 인출할 수 있다. 그림 2에 나타낸 제 2 단계의 과정을 자세하게 기술하면 다음과 같다.

1. 명령어 인출, 재명명

초기화 작업을 거친 후에, 그루핑 함수가 명령어 윈도우 생성 함수를 부르고 이것이 다시 명령어 인출 함수를 호출한다. 결과로써, 디지털 신호처리 프로세서는 매 싸이클마다 새로운 명령어를 인출받는데, 만일 분기명령어를 만나면 인출을 중단한다.

겟노드 함수에서 인출한 명령어는 재명명 함수에서 명령어 재명명 (renaming) 작업을 거치면서 명령어 종속에 의한 타임스탬프 (timestamp) 값을 설정받는다.

모든 명령어는 인출 초기에 현재 싸이클에 명령어 자체의 결과 지연 싸이클을 더한 값을 타임스탬프로 초기화한다. 이후의 디코드 및 재명명 단계에서, 각 명령어의 모든 소오스 레지스터에 대하여 레지스터 화일을 검색하여 같은 레지스터의 타임스탬프 값을 찾는다. 이것은 현재 명령어에 종속을 야기하는 원인을 제공하는 목적 레지스터를 찾는 과정이다. 만일 이렇게 해서 찾은 레지스터의 타임스탬프 값이 현재 명령어의 소오스 레지스터의

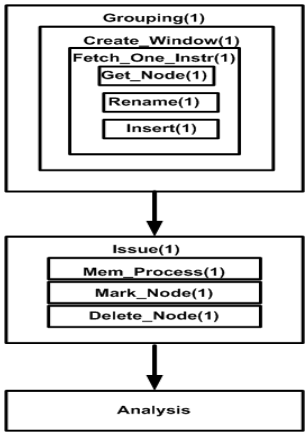

그림 2. 모의실험기
Fig. 2. The simulator

타임스탬프 값보다 더 크다면, 소오스 레지스터의 타임스탬프는 그 값으로 대체되어야 한다. 또한, 현재 명령어의 목적 레지스터의 타임스탬프도 갱신된 소오스 레지스터에 결과지연 싸이클 수를 더한 값으로 대체한다. 이러한 타임스탬프 방식으로 인하여 명령어 자취를 이용하는 모의실험에서 데이터 종속성을 신속하고 효율적으로 부여할 수 있다. 타임스탬프에 의하여 재명명을 거친 명령어는 인서트 함수에서 각 코어의 명령어 윈도우에 삽입된다.

2. 명령어 이슈

이슈 함수로 진행하면, 싸이클이 증가함에 따라서 윈도우 내의 명령어는 자체의 타임스탬프 값이 현재 싸이클 보다 작거나 같고 연산유닛을 활용할 수 있을 때 윈도우로부터 삭제될 수 있다. 그러나, 첫 단계인 마크노드 함수에서 즉각 삭제하지 않고 삭제 가능 표시만 하며, 다음 단계인 딜리트 노드 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 수퍼스칼라 프로세서인 상황에서도 순차종료(in-order completion)를 보장할 수 있다.

3. 시뮬레이션

디지털 신호처리 프로세서의 윈도우 공간에서 그루핑 함수를 사용하여 명령어를 인출하고, 이슈 함수로 명령어를 실행하는 동안 종속성에 의해 주어진 명령어의 타임스탬프가 충족되면 명령어를 삭제한다. 위의 이슈 동작은 명령어가 삭제되어 윈도우가 빈 상태가 될 때까지 반복적으로 실행된다. 윈도우가 비어있는 상태가 되면, 다시 그루핑 함수를 통하여 윈도우를 명령어로 채운다. 이 과정은 입력으로 주어진 벤치마크 프로그램의 모든 명령어가 소진될 때까지 반복된다. DRAM 모델을 반영하기 위하여, 로드나 스토어 명령어를 만났을 때는 어드레스를 이용하여 캐쉬와 DRAM을 접근하기 위하여 소비된 싸이클 수를 늘린다.

위 과정이 한번 실행될 때 마다 싸이클이 증가하므로 모의실험에 입력으로 쓰인 명령어의 총 개수를 처리하기 위하여 소요된 싸이클 수로 나누어, 디지털 신호처리프로세서 시스템의 IPC(Instruction Per Cycle)을 계산할 수 있다[7].

IV. 모의실험 환경

모의실험은 운영체제 Fedora 25에서 3.1 GHz로 작동하는 Intel Core i5-2400에서 수행하였다. 표 1은 모의실험에 사용된 디지털 신호처리 프로세서 아키텍쳐의 사양을 보인다.

표 1. 디지털 신호처리 프로세서 아키텍쳐 하드웨어
Table 1. The architecture specification of DSP

Item	Value
primary instruction cache	64KB, 2-way set associative, 16 B, miss-penalty of 10 cycles
primary data cache	64KB, direct, 32 B, mis-penalty of 10 cycles
instrucion window	8
fetch rate, issue rate, retire rate	2
functional units	ALU(2), load & store(1), branch(1), fp_add(1), fp_mul(1)
branch address cache	1 K entry
branch predictor	8 bit global history mis-penalty of 6 cycles
issue latency	ALU(1), branch(1), load(1), store(1), fp_mul(1), fp_div_single(4), fp_div_double(7)
result latency	ALU(1), branch(1), load(1), store(1), fp_mul(3), fp_div_single(6), fp_div_double(9)

디지털 신호처리 프로세서는 수퍼스칼라 방식으로 운영되어, 매 싸이클 마다 2 개의 명령어를 인출 가능하고

윈도우의 크기는 8이다. 연산유닛은 정수형 유닛, 로드 스토어 유닛, 분기 유닛, 실수형 덧셈기, 실수형 곱셈기로 구성된다.

1 차 명령어 캐쉬와 데이터 캐쉬는 그 용량이 증가할수록 성능에 유리하므로, 64 KB의 용량을 갖도록 설정하였다. 1 차 명령어 캐쉬는 2 차 연관도(set associativity)를 가지나, 1 차 데이터 캐쉬는 직접 매핑으로 설정하였다. 분기 명령어는 2 단계 적응형 분기 예측 방식을 적용하였다. 표 2는 모의실험에 이용된 아홉 개의 UTDSP 디지털 신호처리 벤치마크 프로그램이다. SimpleScalar를 통하여 MIPS IV 5천만 개의 명령어 자취를 모의실험에 적합하도록 발생시켰다[8].

표 2. UTDSP 벤치마크 프로그램
Table 2. UTDSP benchmark programs

벤치마크	설 명	유 형
compress	압축 프로그램	실수형
edge detect	256 그레이 수준 128x128 픽셀 이미지의 에지를 탐지	정수형
FFT	1024 개 점의 복소수 고속 푸리에 변환	실수형
FIR	64 개의 점을 처리하는 256 탭 필어	실수형
histogram	히스토그람 균등화를 통헌 256 그레이 수준 128x128 픽셀 이미지의 개선	정수형
IIR	64 개 점에 대한 4 개 직렬 IIR 바이쿼드 필터	실수형
lpc	선형 예측 코딩 부호화기의 구현	실수형
multiplication	두 개 10x10 행렬의 곱셈	실수형
spectral estimation	피리어드그램 평균을 이용하여 음성의 입력 샘플에 대한 전력 스펙트럼 예측	실수형

그림 2는 본 DRAM 시뮬레이터와 연동하는 디지털 신호처리 프로세서 모의실험의 전체 흐름도를 나타낸 것으로서, 크게 세가지 단계로 나뉜다.

첫 단계에서, 초기 DRAM 프로화일러를 통하여 DRAM에 접근하는 어드레스와 읽기/쓰기 여부를 구분하여 저장한다. 디지털 신호처리 프로세서에서 DRAM을 접근하는 경우는 명령어를 최초로 인출하거나, 각각 명령어 캐쉬와 데이터 캐쉬에서 캐쉬 미스가 발생하는 경우이다.

두 번째 단계에서 DRAM 시뮬레이터가 위 파일을 입력으로 받아서 DRAM 내부에서 처리하기 위한 싸이클 수를 계산한다. 본 모의실험에서 설정한 DRAM은 1600 MT/s의 전송률, 11-11-11 타이밍, 11.9 GB/s의 대역폭을 갖는 DDR3이다[9].

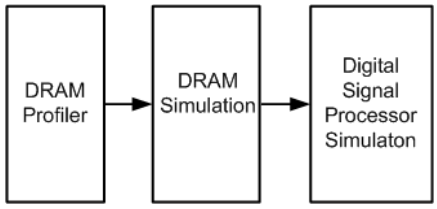

그림 3. 모의실험의 흐름도
Fig. 3. The Simulation Flow

마지막인 세 번째 단계에서, DRAM 시뮬레이터로부터 계산한 DRAM 소요 싸이클 수를 입력으로 디지털 신호처리 프로세서의 성능을 계산한다.

V. 모의실험 및 결과

초기 DRAM 프로화일러에 의하여 생성되어 DRAM 시뮬레이터에 입력으로 제공되는 입력화일은 5천만 개의 명령어 중에서 명령어 인출이나 로드/스토어 명령으로 인하여 DRAM에 접근하는 명령어들만 별도로 분리하고 16 진수의 어드레스와 읽기, 쓰기에 따라 각각 R, W로 구분하여 표현된다.

표 3은 모의실험에서 디지털 신호처리 프로세서의 명령어를 실행하기 위하여 소요되는 싸이클 수와, DRAM에 접근하는데 소요된 싸이클 수를 각 벤치마크에 대하여 나타낸 것이다. *FIR, histogram, lpc, spectral estimation*은 DRAM에서 소비되는 싸이클 수가 상대적으로 높기 때문에, DRAM에 의하여 성능 손실을 상대적으로 크게 입을 것으로 예측할 수 있다. 그리고, *compress, edge detect,*

표 3. 모의실험과 DRAM 소비 싸이클 수의 비교
Table 3. The comparison of simulated cycles vs. DRAM active cycles

benchmarks	number of cycles for instructions	number of cycles spent on DRAM
compress	11,668,776	2,392,442
edge detect	27,923,682	6,842,961
FFT	26,930,372	9,853,523
FIR	2,245,745	209,620
histogram	26,246,665	24,520,075
IIR	313,719	33,739
lpc	4,389,624	8,737,119
multiplication	8,582,771	2,869,723
spectral estimation	14,184,771	24,573,743

표 4. DSP의 성능에 영향을 끼치는 성분의 백분율
Table 4. The percentage of components that affect the performance of DSP

벤치마크	분기 페널티	명령어캐쉬	데이터 캐쉬	분기캐쉬	DRAM
compress	8.97	38.46	3.21	14.10	35.26
edge detect	5.78	35.26	13.29	13.29	32.37
FFT	5.53	37.33	14.75	11.06	31.34
FIR	5.23	38.24	16.34	8.50	31.70
histogram	3.83	43.55	4.18	13.24	35.19
IIR	5.95	40.48	10.71	9.52	33.33
lpc	2.72	46.53	3.63	8.46	38.67
multiplication	4.21	29.89	8.43	32.18	25.29
spectral estimation	2.66	44.38	7.69	8.58	36.69

*FFT, multiplication*은 DRAM에서 소비되는 싸이클 수가 중간 정도이므로 그 정도의 손실이 예상된다. 반면에, *IIR*은 DRAM에서의 소비되는 싸이클 수가 명령어가 처리되는데 필요한 싸이클 수보다 상대적으로 훨씬 적어서 DRAM을 접근함으로써 발생하는 성능의 감소가 거의 없다. 따라서, 이 경우는 프로그램 고유의 명령어 수준 병렬성 및 캐쉬에 의하여 전체 성능이 좌우된다.

표 4에 각 벤치마크의 성능을 저해하는 다섯 가지 요소인 분기 미스페널티, 명령어 캐쉬 미스, 데이터 캐쉬 미스, 분기 어드레스 캐쉬 미스, DRAM 접근시 소요되는 싸이클 수를 백분율로 나타냈다. 본 연구의 디지털 신호처리 프로세서는 표 3 과 표 4에 의하여 최종 성능이 결정된다.

그림 4에 최종적으로 아홉 개의 UTDSP 벤치마크에 대하여 DRAM의 동작을 포함한 모의실험을 수행하여 성능을 IPC로 나타냈다. *Lpc*는 명령어캐쉬 미스에 의하여 그 성능이 83 %, DRAM을 접근하는데 69 % 감소하여, 가장 낮은 수준인 0.34 IPC를 얻는데 그쳤다. 그러나, *IIR*은 명령어 캐쉬에 의한 손실이 18 %에 불과하고 DRAM에 의하여 불과 15 %의 성능 감소로 인하여 가장 높은 1.36 IPC를 기록하였다. *FIR*은 명령어 캐쉬에 의하여 66 %, DRAM으로 인하여 55 %의 성능 손실로 중간에 해당하는 0.50 IPC를 기록하였다.

결과적으로, 각 디지털 신호처리 벤치마크 프로그램별로 디지털 신호처리 프로세서는 최저 0.34 IPC에서 최고 1.36 IPC 범위를 나타내고 평균 0.67 IPC의 성능을 기록하였다.

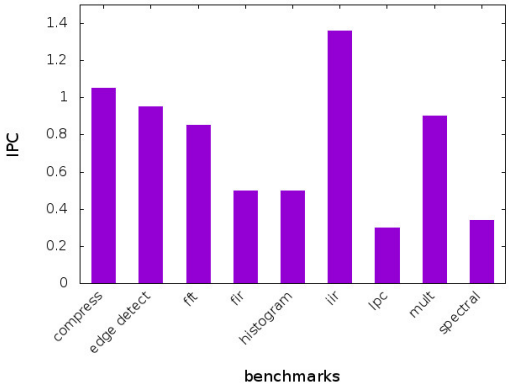

그림 4. UTDSP 벤치마크에 대한 DSP의 성능
Fig. 4 The performance of DSP for UTDSP

본 연구에 따르면, 디지털 신호처리 프로세서의 성능이 명령어 수준 병렬성 및 캐쉬와 더불어 DRAM을 접근하는데 얼마나 많은 싸이클을 소비하느냐에 따라서 결정됨을 알 수 있다. 아홉 개의 벤치마크에 대하여 명령어 캐쉬에 의한 성능의 손실이 평균 39 %로 가장 높게 나타났으며, DRAM에 의한 손실이 두 번째로 높은 33 %를 기록하여, 성능에 무시하지 못하는 영향을 주는 것을 알 수 있다.

VI. 결 론

본 논문에서는 싸이클 단위로 동작하는 DRAM 시뮬레이터와 함께 디지털 신호처리 프로세서의 성능을 측정할 수 있는 디지털 신호처리 프로세서 시뮬레이터를 개발하였다. 이것을 위하여 초기 DRAM 프로화일러에 의하여 얻은 데이터를 DDR3를 모델링한 DRAM 시뮬레이터에 입력하여 DRAM에서 소요되는 싸이클 수를 획득하였으며, UTDSP 디지털 신호처리 벤치마크를 입력으로 실행하는 디지털 신호처리 프로세서의 성능을 측정하였다.

그 결과, 디지털 신호처리 프로세서가 DRAM을 접근하는데 드는 손실이 명령어 캐쉬 미스에 의한 성능 손실 다음으로 영향력이 크기 때문에 성능을 보다 정밀하게 측정하려면 정확한 DRAM 모델이 필요하다는 것을 알 수 있었다.

추후로, DDR3 이외에 DDR4, SALP, LPDDR3, LPDDR4, GDDR5 등의 기타 DRAM을 이용하는 디지털 신호처리 프로세서의 성능을 평가하고 비교하는 연구가 필요하다.

References

[1] L. J. Karam, I. AlKamal, A. Gatherer, G. A. Frantz, D. V. Anderson, B. L. Evans, "Trends in Multi-core DSP Platforms," IEEE Signal Processing Magazine, pp. 1- 10, Nov. 2009

[2] P. Rosenfeld et al. "DRAMSim2: A Cycle Accurate Memory System Simulator," IEEE Computer Architecture Letters, 2011.

[3] Y. Kim et al. "A Case for Exploiting Subarray-Level Parallelism (SALP) in DRAM," ISCA, 2012.

[4] D. Lee et al. "Tiered-Latency DRAM : A Low Latency and Low Cost DRAM Architecture," HPCA, 2013.

[5] C. G. Lee, "UTDSP Benchmark," http://www.eecg.toronto.edu/~corinna/DSP/infrastructure/UTDSP.html, May 1998.

[6] Y. Kim, W. Yang, and O. Mutlu, "Ramulator : A Fast and Extensible DRAM Simulator," IEEE Computer Architecture Letters, 2015.

[7] 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.

[8] T. Austin, E. Larson, and D. Ernest, "SimpleScalar : An Infrastructure for Computer System Modeling," Computer, Vol. 35, No. 2, pp. 59-67, Feb. 2002.

[9] JEDEC, JESD79-3 DDR3 SDRAM Standard, Jun. 2007.

저자 소개

이 종 복(정회원)

- 1964년 8월 20일생.
- 1988년 서울대 컴퓨터공학과 졸업.
- 1998년 동 대학 전기공학부 졸업(공박).
- 1998년 ~ 2000년 LG반도체 선임연구원.
- 2000년 ~ 현재 한성대 전자정보공학과 교수
- Tel : 02-760-4497
- Fax : 02-760-4435
- E-Mail : jblee@hansung.ac.kr
- 관심분야 : 임베디드 프로세서, 멀티코어 프로세서, 인공지능

※ 본 연구는 한성대학교 교내학술연구비 지원과제임.