KIPS Tr. Comp. and Comm. Sys.
Vol.4, No.2 pp.41~46 pISSN: 2287-5891

http://dx.doi.org/10.3745/KTCCS.2015.4.2.41

A Cache-based Reconfigurable Accelerator in Die-stacked DRAM

Yongjoo Kim†

ABSTRACT

The demand on low power and high performance system is soaring due to the extending of mobile and small electronic device market. The 3D die-stacking technology is widely studying for next generation integration technology due to its high density and low access time. We proposed the 3D die-stacked DRAM including a reconfigurable accelerator in a logic layer of DRAM. Also we discuss and suggest a cache-based local memory for a reconfigurable accelerator in a logic layer. The reconfigurable accelerator in logic layer of 3D die-stacked DRAM reduces the overhead of data management and transfer due to the characteristics of its location, so that can increase the performance highly. The proposed system archives 24.8 speedup in maximum.

Keywords : 3D, Reconfigurable, Accelerator, TSV, Memory, DRAM, RA, CGRA

3차원 구조 DRAM의 캐시 기반 재구성형 가속기

김 용 주†

요 약

컴퓨터 사용 환경이 모바일 시장 및 소형 전자기기 시장 등으로 다양해짐에 따라 저전력 고성능 시스템에 대한 요구도 커지고 있다. 3차원 die-stacking 기술은 한정된 공간에서 DRAM의 집적도과 접근 속도를 높여 차세대 공정방식으로 많은 연구가 되고 있다. 이 논문에서는 3차원 구조의 DRAM 로직층에 재구성형 가속기를 구현하여 저전력 고성능 시스템을 구성하는 방법을 제안한다. 또한 재구성형 가속기의 지역 메모리로 캐시를 적용하고 활용하는 방법에 대해서 논의한다. DRAM의 로직층에 재구성형 가속기를 구현할 경우 위치적인 특성으로 데이터 전송 및 관리에 필요한 비용이 줄어들어 성능을 크게 향상시킬 수 있다. 제안된 시스템에서는 최대 24.8의 스피드업을 기록하였다.

키워드 : 3차원, 재구성형, 가속기, 실리콘관통전극, 메모리, D램, 재구성형 프로세서, CGRA

1. 서 론

3D die-stacking 기술은 여러 개의 다이(die)를 한 공간에 여러 층으로 쌓음으로써 작은 공간에 많은 용량을 집적할 수 있고 각 다이 사이에 짧은 연결을 만들 수 있도록 해준다. 다이를 쌓는 방법에 대해서는 여러 가지 연구가 진행되었으며, 주로 작은 공간에 집적도를 높여가면서 연결을 효율적으로 하는 방법에 대해서 연구되었다[1-2].

3D die-stacking 방법에는 몇 가지가 있는데, 이 중 Fig. 1에서 보인 것처럼 Through-Silicon-Via(TSV)를 이용하는 방법이 빠르면서도 높은 집약도를 가질 수 있어 많이 사용되고 있다. 이 방식은 여러 개의 DRAM층(DRAM layer)을 쌓은 후, DRAM의 작업을 보조하기 위한 주변회로(peripheral circuitry), 인터페이스 회로(interface circuitry), built-in self-test(BIST) 모듈 등의 구성요소를 구현한 로직층(logic layer)과 합치는 방법이다. 로직층은 CMOS 공정으로 구현되고 DRAM층은 NMOS 공정으로 구현하는데, 성능에 최적화된 CMOS 공정은 빠른 반응 속도를 보이고, 저장 밀도에 최적화된 NMOS 공정은 많은 데이터를 집적할 수 있기 때문에, 둘의 장점을 다 살릴 수 있도록 로직층과 DRAM층을 다른 공정에서 제작한 뒤 둘을 TSV를 통해서 연결한다.

TSV를 사용해 다이를 쌓을 때에는 공정의 안정성을 위해서 Fig. 1에 나온 것처럼 DRAM층과 로직층의 크기를 똑같게 맞춘다. 로직층에 앞서 설명한 회로를 구현하는 데에는 많은 공간이 필요하진 않지만, DRAM층과 로직층의 크기를 똑같이 맞추는 게 반도체 수율을 증가시키는 데 도움이 되기 때문에 둘의 사이즈를 맞춘다.

※ 본 연구는 미래창조과학부 및 정보통신기술진흥센터의 정보통신.방송 연구개발사업의 일환으로 수행하였음. [14-824-10-017, 오류 없는 시스템 통합을 위한 안전우선 분산 모듈형 SW 플랫폼]
† 정 회 원 : 한국전자통신연구원 임베디드SW부 연구원
Manuscript Received : October 30, 2014
First Revision : December 16, 2014; Second Revision : Junuary 7, 2015
Accepted : Junuary 19, 2015
* Corresponding Author : Yongjoo Kim(y.kim@etri.re.kr)

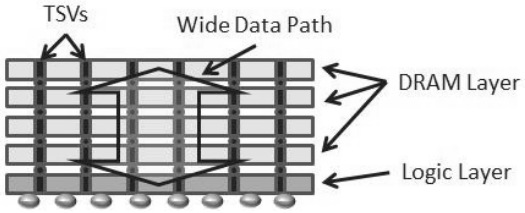

Fig. 1. Hybrid layer stacking

로직층 부분을 DRAM층의 크기만큼 크게 만들 경우 사용되지 않는 공간이 로직층에 생기게 된다. 이 공간은 이전까지는 낭비되는 공간으로 놔두었지만, 최근에는 이 공간을 활용하고자 하는 방법들이 시도되고 있다. [3]에서는 이 공간을 메모리 행 버퍼(row buffer)를 임시로 저장하기 위한 공간으로 사용하기를 제안하였다. 하지만 아직까지 많은 연구들이 진행되지 않아 좀 더 효율적인 공간 활용 방법에 대한 연구가 필요하다.

우리는 3차원 집적된 DRAM과 이 빈 공간의 특성을 분석하고 활용하여, 그 공간을 효율적으로 사용할 수 있는 재구성형 가속기를 제안하였다. 또한 이 가속기와 DRAM의 수행을 효율적으로 보조할 수 있는 캐시 기반의 지역 메모리(local memory)를 제안한다.

남은 부분은 다음과 같이 구성되었다. 2절에서는 3차원 구조의 DRAM과 재구성형 가속기의 기본 배경지식을 설명한다. 3절에서는 본 논문에서 제안하는 3차원 구조 DRAM에 대해서 설명하고, 4절에서는 실험을 통해 제안된 DRAM 시스템을 검증한다.

2. 배경 지식

2.1 3D 차원 구조 DRAM과 특성

3D Die-stacked DRAM은 Fig. 2처럼 여러 개의 DRAM층(DRAM layer)과 로직층(Logic layer)를 겹쳐놓은 구조로 되어있다. DRAM층은 기존의 DRAM 구조와 동일하게 NMOS 공정을 사용하여 제작되며 하나의 비트를 구성하는데 하나의 트랜지스터가 사용되어 적은 공간에 많은 데이터를 저장할 수 있다. 대신 값들을 처리하는 속도는 느린 편이다. 로직층은 CMOS 공정을 사용하고 여러 가지 속도에 최적화된 기법들이 사용된다. 로직층은 반응 속도가 빠르기 때문에 데이터 저장보다는 데이터에 접근하기 위한 여러 가지 주변 기기들이 주로 구현된다. 주로 구현되는 주변장치는 행 복호기(row decoder), 센스 증폭기(sense amplifier)와 같은 DRAM 관련 주변장치, off-chip 드라이버, 리시버와 같은 인터페이스 회로, Built-In Self-Test(BIST)와 같은 장치들이다. 로직층과 DRAM층은 각층을 수직으로 관통하는 TSV(Through-Silicon-Via)를 통해서 연결된다. 기존의 2차원 구조와 비교해서 각층이 물리적으로 가깝게 붙어있고, 그를 관통하는 넓은 채널을 통해서 각층은 대량의 데이터를 빠르게 주고받을 수 있다. CPU에서 3D 차원 구조 DRAM의 데이터에 접근할 때에는 로직층의 인터페이스를 통해 데이터를 요청하게 되고, 로직층에 있는 DRAM 관리 모듈은 TSV를 통해 DRAM층에 접근해 데이터를 주고받는다.

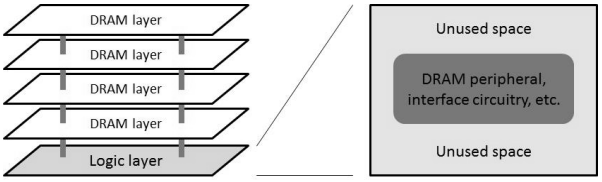

Fig. 2. 3D die-stacked DRAM and the unused space in logic layer

로직층과 DRAM층은 안정성을 위해 동일한 사이즈로 구현되어 적층된다. 로직층에 구현되는 주변장치와 인터페이스 모듈은 공간을 많이 차지하지 않기 때문에 약간의 빈 공간이 로직층에 생긴다. Fig. 2에서 오른쪽에 로직층을 보인 것처럼 인터페이스와 DRAM 주변장치를 구현한 후 일부는 빈 공간으로 남아있게 된다. 이 논문에서는 이 빈 공간을 활용하여 메모리 관련 작업의 일부를 가속할 수 있는 장치를 구현하여 DRAM을 좀 더 효율적으로 사용하고자 한다.

이 빈 공간은 일반적인 SoC의 공간과 다른 몇 가지 특징이 있다. 첫째, 빈 공간이 DRAM층과 물리적으로 가까운 곳에 있어 데이터에 빠르게 접근할 수 있다. 이 빈 공간은 DRAM 내부의 로직층에 있기 때문에, 여러 단계의 버스를 거칠 필요 없이 DRAM층에 바로 접근할 수 있어 데이터 접근이 빠르다. 둘째, 이 공간에서는 TSV의 넓은 대역폭(bandwidth)을 활용해 대량의 데이터에 한 번에 접근할 수 있다. DRAM 내부에서는 DRAM층에서 광역폭의 TSV 다발을 통해서 데이터를 꺼내기 때문에 대량의 데이터를 동시에 꺼낼 수 있다. 하지만 이 데이터가 외부로 나갈 때에는 메모리 인터페이스를 통해 폭이 좁은 버스를 지나 데이터가 전달되므로 외부에서는 대량의 데이터에 한 번에 접근할 수 없다. DRAM의 로직층에서는 이 데이터에 직접적으로 접근할 수 있어 한 번에 대량의 데이터에 접근할 수 있다.

하지만 이 공간은 다른 공간에 비해서 제약 조건도 있다. 첫째, 이 공간은 로직층의 남는 공간을 활용하기 때문에 활용할 수 있는 공간이 작다. 로직층은 원래 DRAM의 작업을 보조하기 위한 DRAM 주변회로(peripheral circuitry), 인터페이스 회로(interface circuitry), Build-In Self-Test(BIST) 모듈 등이 들어가는 공간으로 이를 제외한 공간은 그렇게 크지 않다. 둘째, 이 공간에 구현될 로직은 발열이 적어야 한다. 이 공간은 적층된 로직층에 존재하는데, 적층 구조는 단층 구조에 비해서 구조적으로 효율적인 열 발산이 어렵다. 또한 3차원으로 적층된 여러 다른 층에서 동시에 열을 발생시키기 때문에 발열량이 적어야 한다.

위의 몇 가지 특징상 로직층의 빈 공간에는 대량으로 데이터를 사용하고 처리하는 장치가 적합하다. 또한 적은 공간을 차지하고 파워를 적게 소모하면서 작업을 처리할 수 있는 장치가 적합하다. 재구성형 가속기(Coarse-Grained

Reconfigurable Architecture)는 크기도 작으면서 높은 성능을 발휘하며, 상황에 따라 수행 작업을 바꿀 수도 있어 DRAM의 로직층의 빈 공간을 활용하기에 적합하다.

2.2 재구성형 가속기(Coarse-Grained Reconfigurable Architecture, CGRA or RA)

Fig. 3은 재구성형 가속기(Coarse-Grained Reconfigurable Architecture, CGRA or RA)의 구조를 보여준다. 재구성형 가속기는 PE(Processing Element)라고 불리는 작은 계산장치가 격자 모양으로 배열되어있고, 인급한 PE끼리는 서로 계산한 결과값들을 주고받을 수 있도록 연결되어있다. PE는 기본적인 산술연산과 로직연산을 수행할 수 있다. 그리고 종류에 따라 일부 PE는 곱셈이나 나눗셈, 메모리 접근과 같은 특수 연산을 수행할 수 있다. 각 PE가 수행하는 내용은 컨피규레이션(configuration)에 의해 결정된다. 이 컨피규레이션은 컨피규레이션 메모리(configuration memory)에 저장되며 매 사이클마다 다른 컨피규레이션을 불러와서 수행할 내용을 바꿀 수 있다. 재구성형 가속기는 ASIC(Application Specific Integrated Circuit)나 다른 하드웨어 가속기와 마찬가지로 특정 작업을 가속하는 역할을 한다. 많은 PE를 활용해 CPU보다 빠르게 계산을 처리한 후 그 값을 메모리로 다시 보내게 된다.

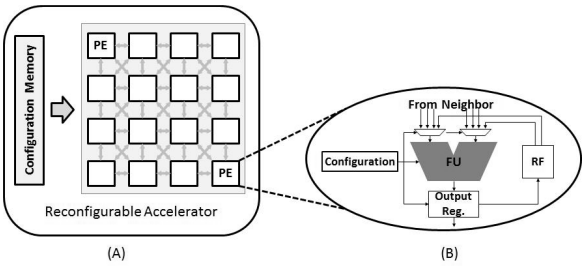

Fig. 3. Structure of CGRA

재구성형 가속기는 많은 수의 PE로 구성되지만 일반적인 CPU에 들어가는 분기(branch) 관련 장치 등이 없어서 크기가 작다. 하지만 많은 PE로 인해 복잡한 분기가 없는 단순 반복 작업에서는 강력한 성능을 보여준다. 재구성형 가속기가 DRAM의 로직층에 구현된다면 대량의 데이터에 동시에 접근하여 빠르게 작업을 처리할 수 있다. 또한 GPU나 ASIP (Application-Specific Instruction-set Processor) 등의 다른 가속기들에 비해서 크기가 작아서 로직층의 작은 빈 공간에 들어가기 적합하다. 또한 성능에 비해 발열량도 작은 편이라[4-5] 발열량이 작아야 하는 3D DRAM 구조에 적합하다.

3. 제안하는 3차원 구조 DRAM

3.1 3D 차원 구조 DRAM과 특성

Fig. 4는 이 논문에서 제안한 전체 시스템을 나타낸다. 왼쪽의 CPU chip 부분이 일반적인 구조의 시스템을 나타낸다. 그리고 오른쪽의 DRAM chip 부분이 이 논문에서 제안한 3차원 구조의 DRAM이다. 재구성형 가속기(RA)는 CPU chip과 떨어져서 DRAM chip의 로직층에 위치한다. 이 시스템에서 보통의 작업을 수행할 때에는 일반적인 시스템처럼 DRAM에서 데이터를 CPU chip의 캐시로 옮겨와 작업을 수행한다. 하지만 데이터 접근 위주의 작업이나, 데이터 병렬성이 있어 가속이 가능한 작업을 수행할 경우에는 데이터를 DRAM에서 CPU chip으로 옮기지 않고 DRAM의 RA에서 직접 메모리에 접근하여 작업 수행을 한다.

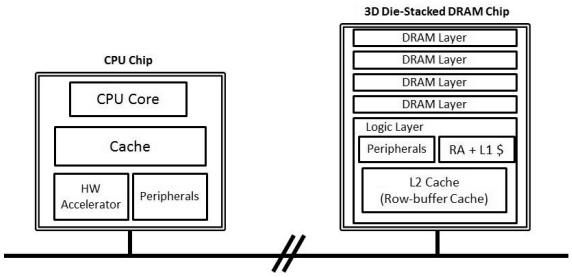

Fig. 4. Block diagram of proposed system

3.2 캐시 기반 재구성형 가속기

재구성형 가속기는 PE의 배열에서 수행할 데이터를 저장할 지역 메모리가 있는데, 기존의 재구성형 가속기는 이 메모리가 스크래치 패드 메모리(Scratch Pad Memory, SPM)로 구성된다. 재구성형 가속기에서 수행할 작업과 사용될 데이터들은 컴파일 단계에서 미리 분석된다. 실제 작업을 수행할 때에는, 미리 분석된 사용할 데이터들을 DRAM에서 재구성형 가속기의 지역 메모리로 전달하여 사용할 준비를 한 후 작업을 수행한다. 일반적인 재구성형 가속기에서 스크래치 패드 메모리를 사용하는 이유는, 재구성형 가속기는 많은 PE를 사용해 많은 데이터를 동시에 처리하기 때문에 중간에 작업이 지연(stall)되면 작업처리량에 많은 손실이 발생한다. 그래서 스크래치 패드 메모리를 사용해 데이터 미스로 인한 작업 지연을 방지한다.

하지만 DRAM에 존재하는 재구성형 가속기는 수행 환경이 다르다. DRAM의 로직층에 재구성형 가속기가 위치할 경우 데이터 미스가 발생하더라도 바로 DRAM층에 접근하여 데이터를 받아올 수 있어 작업 지연시간이 적다. 그리하여 우리는 DRAM의 로직층에 구현하는 재구성형 가속기의 지역 메모리로 캐시를 적용하여 구현하였다.

Fig. 5는 제안한 재구성형 가속기와 DRAM과의 연결 구조를 나타낸다. 재구성형 가속기는 여러 개의 작은 L1 캐시와 하나의 L2 캐시를 지역 메모리로 사용한다. L1 캐시는 여러 개의 작은 캐시를 사용하여 데이터를 주소에 따라 다른 위치의 캐시에 저장하도록 하여 재구성형 가속기에서의 동시 작업 처리 능력을 향상시켰다. L2 캐시는 한 라인의 사이즈가 DRAM의 한 블록 사이즈(row buffer 사이즈)와 동일하도록 구현하여 DRAM에서 나온 데이터를 한 번에 담고 있도록 하였다. DRAM에 접근하는 것은 파워 소모와 접

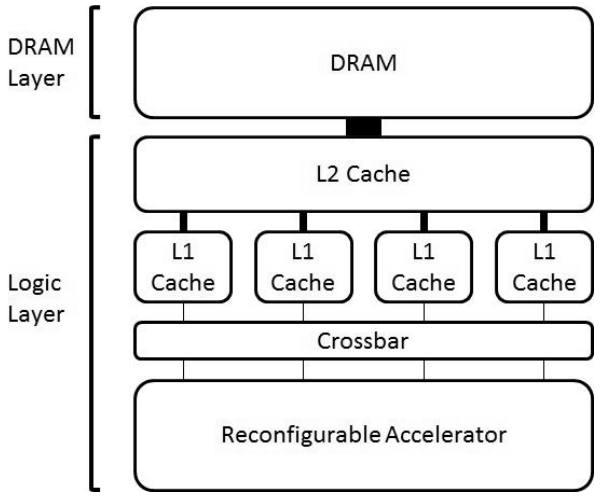

Fig. 5. Structure of reconfigurable accelerator and local memory

근 소요 시간이 크므로 L2 캐시를 이용해 DRAM 접근을 줄이도록 하였다. 재구성형 가속기를 사용하지 않고 CPU에서 작업을 수행하는 경우, L2 캐시를 DRAM의 블록에서 나온 데이터를 담는 버퍼로 재활용하여 CPU chip의 성능 향상에도 기여할 수 있다.

재구성형 가속기의 지역 메모리를 스크래치 패드 메모리에서 캐시로 바꾸는 경우 몇 가지 장점이 있다. 첫째, 데이터 관리에 드는 비용을 줄일 수 있다. 기존의 재구성형 가속기는 수행하기 전에 사용할 데이터를 DRAM에서 지역 메모리(스크래치 패드 메모리)에 미리 옮겨놓는 작업이 필요하다. 이 작업은 CPU chip에 있는 CPU에서 DRAM에 있는 재구성형 가속기의 메모리 관리자로 수행에 오래 걸리는 off-chip 통신을 하여 명령을 전달한다. 하지만 캐시를 사용하면 스크래치 패드 메모리를 위한 관리를 하지 않아도 되어서 off-chip 버스를 통한 통신과 데이터 전송이 없어지고, 그로 인해 수행시간이 감소하고 통신을 위한 파워 소모가 줄어든다. 둘째, 재구성형 가속기를 좀 더 쉽게 사용할 수 있다. 스크래치 패드 메모리에 데이터를 옮겨놓기 위해서는 컴파일러에서 사용할 데이터 분석이 있어야 한다. 컴파일러에서 이 분석을 지원하지 못할 경우 스크래치 패드 메모리를 사용할 수 없다. 또한 응용의 구현 형태에 따라 사용할 데이터를 분석할 수 없는 경우도 있는데 이런 경우에도 캐시를 사용한 경우에는 재구성형 가속기를 사용할 수 있다. 마지막으로 지역 메모리를 재활용할 수 있다. 지역 메모리를 스크래치 패드 메모리로 구현한 경우에는 재구성형 가속기를 사용하지 않을 때에는 지역 메모리도 사용되지 않았다. 하지만 지역 메모리를 캐시로 구현한 경우에는 DRAM의 로우 버퍼(row buffer)를 담는 캐시로 재활용할 수 있어 CPU에서 작업을 수행할 경우에도 성능 향상에 기여할 수 있다.

4. 실 험

4.1 실험 세팅

실험은 x86 데스크톱에서 진행하였고 CPU와 DRAM의 동작 시뮬레이션을 위해서 GEM5 시뮬레이터[6]를 이용하였다. DRAM의 3차원 메모리 시스템 구현과 캐시 동작 테스트는 GEM5 시뮬레이터의 메모리 시뮬레이션 모듈인 RUBY를 실험 환경에 맞게 수정하여 시뮬레이션하였다. 재구성형 가속기의 동작은 in-house 시뮬레이터[4]를 GEM5에 구현하여 같이 성능을 측정하였다. 재구성형 가속기는 각각 500MHz의 속도로 동작하도록 하였다. DRAM의 세부 동작 수치는 CACTI-3DD[7]를 사용하여 측정 후 시뮬레이션에 반영하였다. CPU칩과 DRAM칩 간의 Off-chip 통신 지연시간은 10ns로 가정하여 실험하였다[8]. 재구성형 가속기의 캐시에 사용한 파라미터는 CACTI[9]를 사용해 반응속도를 측정하였다.

실험에 사용한 CPU는 3GHz의 4-Issue Out-of-order 프로세서를 사용하였다. CPU는 6개의 정수연산 유닛, 2개의 곱셈-나눗셈 계산 유닛, 4개의 실수 계산 유닛, 2개의 실수 곱셈-나눗셈 계산 유닛, 4개의 메모리 접근 유닛을 가지고 있고 32KB의 L1 캐시와 128KB의 L2 캐시를 가지고 있다. 재구성형 가속기는 4x4 PE array로 구성되어있고 4개의 곱셈-나눗셈 유닛, 4개의 메모리 접근 유닛을 가지고 있다. 재구성형 가속기의 하드웨어 세팅은 [4]를 참조하였다. DRAM은 총 4개의 DRAM층이 적층된 4GB의 메모리를 사용하였다. 자세한 세팅 내용은 Table 1과 같다.

Table 1. Experiment system setting

	Specification
CPU	3GHz, Out-of-order 4-Issue
	6 Int + 2 MultDiv + 4 FP + 2 FPMultDiv + 4 LD/ST
	32KB L1 I-Cache, 32KB L1 D-Cache, 128KB L2 Cache
Bus	1GHz, Width = 64bit, Latency 20ns
DRAM	4GB DRAM, 4 Dies, tRCD = 9.2ns, tRP = 3.2ns
RA	500Mhz, 16 PE, 4 MultDiv, 4 LD/ST
Local Memory(RA)	1KB L1 Cache x 4
	64KB L2 Cache

실험에 사용한 프로그램은 SPEC CPU 2000의 swim과 MPEG2 디코더, 그리고 SHA이다. 실험내용은 총 3가지로 비교하였다.

- CPU only : DRAM의 로직층에 아무런 추가 구현을 하지 않은 상태에서 CPU상에서 벤치마크 프로그램을 실행함
- CPU+DRAM Cache : DRAM의 로직층에 구현된 재구성형 가속기의 L2 캐시를 DRAM의 로우 버퍼 캐시로 사용함

- CPU+RA : 기본적인 작업은 CPU에서 수행하고 가속이 가능한 부분이 나오면 DRAM의 재구성형 가속기를 사용하여 작업을 수행함(RA=재구성형 가속기, Reconfigurable Accelerator)

4.2 수행시간 비교

Fig. 6은 각 실험에서의 수행시간을 보여준다. 모든 벤치마크에서 CPU only보다 CPU+RA에서 수행시간이 훨씬 짧아짐을 확인할 수 있다. CPU+DRAM Cache의 경우 벤치마크에 따라서 수행시간 감소 효과가 있고 없음이 차이가 있는데, 이는 벤치마크의 메모리 접근 패턴이 달라 DRAM 캐시의 활용성이 달라짐에 의한 것이다. CPU+RA는 CPU only보다 평균 11의 스피드업(speedup)을 보였다. Swim은 24.8로 가장 높은 수치를 기록했고 MPEG2 디코더는 3.6으로 가장 낮은 수치를 보였다. 이 수치의 차이는 각 벤치마크의 코드 병렬성, 그리고 CPU chip과 DRAM chip 간의 데이터 전달 빈도 및 전달량에 따라 달라진다.

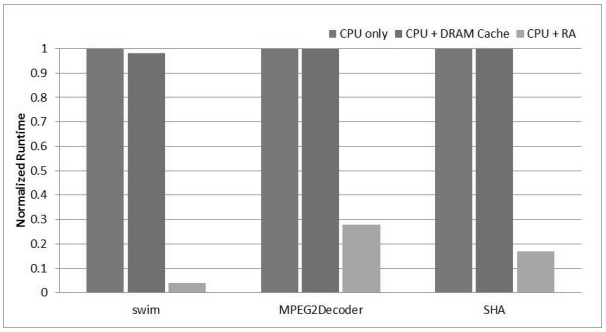

Fig. 6. Runtime Comparison in 3 Cases

4.3 코드 병렬성과 성능

재구성형 가속기는 단순한 구조 특성상 분기문이 많은 코드는 수행할 수가 없고 반복수행을 처리하는 데 특화되어있어 병렬성이 높은 코드를 수행하는 것이 유리하다. 코드의 병렬성에 따른 성능의 변화에 대해서 분석해보았다.

Table 2. The Runtime Breakdown (Normalized to CPU only runtime)

	CPU only		CPU+RA	
	Sequential Region Runtime (in CPU)	Parallel Region Runtime (in CPU)	Sequential Region Runtime (in CPU)	Parallel Region Runtime (in RA)
Swim	0.001>	0.999<	0.001>	0.040
Mpeg2Dec	0.247	0.753	0.251	0.027
SHA	0.132	0.868	0.132	0.036

Table 2는 각 벤치마크의 코드를 순차적으로 수행해야 하는 순차적 영역과 병렬성이 있는 병렬적 영역으로 나누어, 각각을 수행하는 데 걸리는 시간을 분석하였다. 병렬적 영역은 재구성형 가속기에서 수행할 수 있는 영역이다. 각 수치는 직관성 향상을 위해 CPU only의 전체 수행시간을 1로 정하고 그에 맞게 보정하였다. CPU only에서는 두 영역 다 CPU에서 수행을 하고 CPU+RA에서는 순차적 영역은 CPU에서 수행하고 병렬적 영역은 RA에서 수행하였다.

Swim 벤치마크의 경우 대부분의 수행시간은 병렬적 영역에서 소모된다. 반면에 Mpeg2 디코더는 순차적 수행 영역의 비중이 Swim 벤치마크에 비해 크기가 커서 재구성형 프로세서를 사용할 수 있는 부분이 더 적다. 이 비율은 앞서 Fig. 6에서 보인 수행시간 감소 차이에 영향을 준 것으로 분석된다. Mpeg2 디코더의 경우 CPU에서 수행하는 순차적 영역의 수행시간이 CPU only보다 CPU+RA 때 조금 더 늘어나는 것을 볼 수 있는데, 이는 재구성형 가속기를 수행하기 위한 통신 코드를 수행하는 시간의 비중이 크기 때문이다. 전체적으로 병렬적 영역을 CPU에서 수행할 때보다 재구성형 가속기에서 수행할 때 수행시간이 많이 줄어든 것을 볼 수 있다. 이는 재구성형 가속기의 많은 PE를 통해 병렬성이 있는 부분을 가속하였고, CPU와 DRAM이 데이터를 주고받는 시간을 줄였기 때문으로 분석된다.

5. 결 론

우리는 이 논문에서 3차원 구조의 DRAM의 로직층에 재구성형 가속기를 구현하는 것을 제안하였다. 3차원 구조의 DRAM에서는 로직층에 빈 공간이 발생하는데, 3차원 구조 DRAM과 로직층의 특징상 재구성형 가속기를 구현하는 것이 적합하였다. 또한 위치적 특징을 살리기 위해 재구성형 가속기의 지역 메모리로 캐시를 적용하고 활용하는 방법에 대해서 논의하였다. 데이터 접근이 많은 응용 프로그램의 경우 DRAM의 재구성형 가속기에서 작업을 수행할 경우 데이터 전송 및 관리에 필요한 비용이 줄어들어 성능을 크게 향상(최대 24.8 스피드업)시킬 수 있었다.

References

[1] Uksong Kang, Hoe-Ju Chung, Seongmoo Heo, Soon-Hong Ahn, Hoon Lee, Soo-Ho Cha, Jaesung Ahn, DukMin Kwon, Jin Ho Kim, Jae-Wook Lee, Han-Sung Joo, Woo-Seop Kim, Hyun-Kyung Kim, Eun-Mi Lee, So-Ra Kim, Keum-Hee Ma, Dong-Hyun Jang, Nam-Seog Kim, Man-Sik Choi, Sae-Jang Oh, Jung-Bae Lee, Tae-Kyung Jung, Jei-Hwan Yoo, and Changhyun Kim, "8Gb 3D DDR3 DRAM Using Through-Silicon-Via Technology," *in ISSCC*, pp.129-131, 2009.

[2] Tezzaron Semiconductor, www.tezzaron.com/

[3] Gabriel H. Loh, "A register-file approach for row buffer caches in die-stacked DRAMs," *In Proceedings of the 44th Annual IEEE/ACM International Symposium on Microarchitecture(MICRO-44 '11). ACM*, New York, NY, USA, pp.351-361, 2011.

[4] Yongjoo Kim, Jongeun Lee, Aviral Shrivastava, and Yunheung Paek, "Memory Access Optimization in Compilation for Coarse-Grained Reconfigurable Architectures," *ACM Transactions on Design Automation of Electronic Systems,* Vol.16, No.4, Article 42, 2011.

[5] Yongjoo Kim, Jongeun Lee, Aviral Shrivastava, Jonghee W. Yoon, Doosan Cho, and Yunheung Paek, "High Throughput Data Mapping for Coarse-Grained Reconfigurable Architectures," *IEEE Transactions on Computer-Aided Design of integrated circuits and systems,* Vol.30, No.11, Nov., 2011.

[6] Nathan Binkert, Bradford Beckmann, Gabriel Black, Steven K. Reinhardt, Ali Saidi, Arkaprava Basu, Joel Hestness, Derek R. Hower, Tushar Krishna, Somayeh Sardashti, Rathijit Sen, Korey Sewell, Muhammad Shoaib, Nilay Vaish, Mark D. Hill, and David A. Wood. 2011. The gem5 simulator. SIGARCH Comput. Archit. News 39, 2 (May, 2011), pp.1-7.

[7] Ke Chen, Sheng Li, Muralimanohar, N., Jung Ho Ahn, Brockman, J.B., and Jouppi, N.P., "CACTI-3DD: Architecture-level modeling for 3D die-stacked DRAM main memory," *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2012,* vol., no., pp.33-38, 12-16 Mar., 2012.

[8] Liu, C.C. Ganusov, I. Burtscher, and M. Sandip Tiwari, "Bridging the processor-memory performance gap with 3D IC technology," *Design & Test of Computers, IEEE*, Vol.22, No.6, pp.556-564, Nov.-Dec. 2005.

[9] Steven J. E. Wilton, and Norman P. Jouppi, "CACTI: An Enhanced Cache Access and Cycle Time Model," *IEEE Journal of solid-state circuits*, Vol.31, No.5, May, 1996, Online http://www.hpl.hp.com/research/cacti/

김 용 주

e-mail : y.kim@etri.re.kr

2006년 서울대학교 전기공학부(학사)

2012년 서울대학교 전기컴퓨터공학부 (Ph.D., 석박사통합)

2012년~2013년 서울대학교 박사후연구원

2013년~현 재 한국전자통신연구원 임베디드SW부 연구원

관심분야 : CGRA, MPSoC, 임베디드 시스템, 메모리 최적화