Journal of The Korea Society of Computer and Information
Vol. 17 No. 7 , July 2012

2012-17-7-1-1

# 2차원 구조 대비 3차원 구조 GPU의 메모리 접근 효율성 분석

전 형 규*, 안 진 우*, 김 종 면**, 김 철 홍*

# Memory Delay Comparison between 2D GPU and 3D GPU

Hyung Gyu Jeon*, Jin Woo Ahn*, Jong-Myon Kim**, Cheol Hong Kim*

## 요 약

최근 반도체 공정 기술이 발달함에 따라 단일 프로세서에 적재되는 코어의 수가 크게 증가하였고, 이는 프로세서의 성능을 급격하게 향상시키는 계기가 되고 있다. 특히, 많은 수의 코어들로 구성된 GPU(Graphics Processing Unit)는 대규모 병렬성을 활용하여 연산처리 성능을 크게 향상시키고 있다. 하지만, 주 메모리 접근 지연시간이 GPU의 성능 향상을 제약하는 심각한 요인 중 하나로 제기되는 상황이다. 본 논문에서는 3차원 구조를 통한 GPU의 메모리 접근 효율성 향상에 대한 정량적 분석과 3차원 구조 적용 시 발생 가능한 문제점에 대하여 살펴보고자 한다. 일반적으로 메모리 명령어 비율은 평균적으로 전체 명령어의 30%를 차지하고, 메모리 명령어 중에서 주 메모리 접근과 관련된 글로벌/로컬 메모리 명령어가 차지하는 비율 또한 평균 60%이므로 주 메모리로의 접근 지연시간을 크게 감소시키는 3차원 구조를 적용한다면 GPU의 성능 또한 크게 향상시킬 수 있을 것으로 예상된다. 그러나 본 논문에서 수행한 실험 결과에 따르면 메모리 병목현상으로 인해 3차원 구조 GPU의 성능이 2차원 구조 GPU에 비해 크게 향상되지는 않음을 확인할 수 있다. 분석 결과에 의하면, 3차원 구조 GPU는 2차원 구조 GPU와 비교하여 메모리 병목현상으로 인한 성능 지연이 최대 245%까지 증가하기 때문이다. 본 논문에서는 3차원 구조 GPU를 대상으로 메모리 접근의 효율성과 문제점을 함께 분석함으로써, 3차원 GPU에 적합한 메모리 구조를 설계하기 위한 가이드라인을 제시하고자 한다.

▸ Keywords : 3차원 구조 프로세서, 그래픽 전용 연산 유닛, 메모리, 성능 분석

## Abstract

As process technology scales down, the number of cores integrated into a processor increases dramatically, leading to significant performance improvement. Especially, the GPU(Graphics

• 제1저자 : 전형규 • 교신저자 : 김철홍
• 투고일 : 2011. 08. 25. 심사일 : 2011. 11. 07, 게재확정일 : 2012. 04. 20.
* 전남대학교 전자컴퓨터공학부(School of Electronics and Computer Engineering, Chonnam National University)
** 울산대학교 전기공학부 컴퓨터정보통신공학과(School of Computer Engineering and Information Technology, University of Ulsan)
※이 논문은 2011년도 정부(교육과학기술부)의 재원으로 한국연구재단 기초연구사업의 지원(2011-0003350)과 지식경제부 및 정보통신산업진흥원의 대학 IT연구센터 지원사업의 연구결과로 수행되었음(NIPA-2012-H0301-12-3005)

Processing Unit) containing many cores can provide high computational performance by maximizing the parallelism. In the GPU architecture, the access latency to the main memory becomes one of the major reasons restricting the performance improvement. In this work, we analyze the performance improvement of the 3D GPU architecture compared to the 2D GPU architecture quantitatively and investigate the potential problems of the 3D GPU architecture. In general, memory instructions account for 30% of total instructions, and global/local memory instructions constitutes 60% of total memory instructions. Therefore, the performance of the 3D GPU is expected to be improved significantly compared to the 2D GPU by reducing the delay of memory instructions. However, according to our experimental results, the 3D architecture improves the GPU performance only by 2% compared to the 2D architecture due to the memory bottleneck, since the performance reduction due to memory bottleneck in the 3D GPU architecture increases by 245% compared to the 2D architecture. This paper provides the guideline for suitable memory design by analyzing the efficiency of the memory architecture in 3D GPU architecture.

▸ Keywords : 3D Integrated Processor, Graphics Processing Unit, Memory, Performance Analysis

## I. 서 론

최근 반도체 공정 기술이 발달함에 따라, 단일 프로세서에 적재되는 코어의 수가 크게 증가하고 있다. 이와 같은 코어 개수의 증가는 프로세서의 성능을 크게 향상시키는 계기가 되고 있다. 특히, 코어 개수의 증가를 통한 성능 향상은 그래픽 전용 연산 유닛인 GPU(Graphics Processing Unit)에서 더욱 두드러지게 나타난다. 예를 들어, NVIDIA사의 최신 그래픽 프로세서 군인 GT500시리즈는 최대 512개의 코어를 적재함으로써 연산 능력을 크게 향상시켰다[1]. 이와 같이 다수의 코어로 구성된 GPU는 프로그램을 스레드(Thread) 단위로 구분하여 연산을 수행한다. GPU 내에 내장된 다수의 코어들은 독립적으로 스레드를 수행할 수 있기 때문에, 코어의 개수가 증가할수록 더욱 많은 수의 스레드를 동시에 처리함으로써 연산 속도를 향상시킬 수 있다. 그러나 단순히 코어의 개수를 증가시키는 것이 성능을 효율적으로 향상시킬 수 있는 것은 아니다. 이상적으로 GPU내의 코어들이 효율적으로 스레드를 처리한다면 성능 향상에 도움이 되지만, 코어에 스레드가 적절하게 분배되지 않으면 성능 향상의 이득을 볼 수 없다. 즉 GPU가 구조적 특성을 효율적으로 활용하기 위해서는 스레드가 여러 코어에 적절하게 분배되어야 한다.

효율적인 스레드 분배를 통한 GPU의 성능을 향상시키는 기법에 대해서는 이전부터 많은 연구들이 진행되고 있다[1-4]. 주로 스레드 분배와 관련된 연구들은 GPU 내의 스레드 스케쥴러의 구조를 변경하는 연구가 주를 이루었다. 그러나 근본적으로 많은 수의 스레드를 코어에 할당하기 위해서는 대용량의 주 메모리에서 데이터를 인출하는 과정이 필요하다. GPU에서는 주 메모리의 데이터를 인출하기 위해 메모리 인터페이스를 거치는데, 이 때 추가적으로 발생하는 지연시간이 400~600사이클에 육박한다[2]. 이는 마이크로프로세서에서 주 메모리를 접근하는데 소요되는 100사이클 내외의 지연시간과 비교하여 매우 크다는 것을 확인할 수 있다[3]. 그러므로 GPU내의 스레드 스케쥴러의 구조를 변경하여 효율적인 스레드 분배 기법을 적용하는 것도 성능 향상에 큰 도움이 될 수 있지만, 스레드를 분배하기 위해 주 메모리의 데이터를 인출하는 과정에서 발생하는 지연 시간을 줄인다면 성능 향상에 더욱 도움이 될 것이다. GPU와 주 메모리 사이에서 추가적으로 발생하는 지연 시간을 줄이기 위해서는 물리적인 해결책이 필요하다. 이러한 물리적인 해결책으로써 3차원 적층 기법을 활용할 수 있다. 3차원 적층 기법은 기존의 멀티 코어 프로세서에서 성능 향상을 위해 연구되었던 기법으로써, 여러 개의 코어를 수직으로 적층시켜 TSV(Through Silicon Via)로 연결함으로써 내부 연결망의 길이를 크게 줄여주어 프로세서의 성능을 향상시키는 기법이다[4]. 3차원 적층 기법을 활용한다면 주 메모리와 GPU 사이의 접근 지연시간 문제를 해결할 수 있으므로 GPU의 성능을 크게 향상시킬 수 있을 것으로 기대된다. 이에 본 논문에서는 GPU와 주 메모리를 수직으로 연결하는 3차원 구조 GPU를 통한 메모리 접

근 측면에서의 성능 향상 효과를 상세하게 분석하고자 한다.

본 논문의 구성은 다음과 같다. 2장에서는 연구 배경 및 관련 연구 내용을 기술하고, 3장에서는 실험 대상인 3차원 구조 GPU에 대하여 설명한다. 4장에서는 실험 환경 및 3차원 구조 GPU의 성능 분석에 관한 상세한 실험 결과를 기술하고, 5장에서 결론을 맺는다.

## II. 연구 배경

### 1. 3차원 구조 프로세서

표 1. CPU에서의 3차원 구조 연구
Table 1. 3D architecture research on CPU

| 연구명 | 내용 |
|---|---|
| Thermal Analysis of a 3D Die Stacked High Performance Microprocessor ( K. Puttaswamy, et.al. 2006)[6] | 2차원 CPU 구조에 비해 3차원 CPU 구조는 더 많은 다이를 쌓음으로써 배선 길이를 크게 감소시킬 수 있음을 제시한다. 하지만, 많은 다이를 쌓음으로 인해 발생하는 열 문제에 대해서 언급한다. 발열 문제는 3차원 구조의 치명적인 단점으로 성능 향상을 저해하는 주된 요인이 되고 있음을 실험 결과를 통해 제시한다. |
| A Novel Dimensionally-Decomposed Router for On-Chip Communication in 3D Architectures (J. Kim, et.al. 2007)[7] | 3차원 CPU 구조가 2차원 CPU 구조에 비해 배선 길이가 짧다는 장점을 상세하게 소개한다. 배선 길이가 짧아지면 전력 소모 또한 크게 줄어드는 이점을 활용하여 3D Networks-on-Chip (NoC) 구조를 제안한다. |
| Design and Management of 3D Chip Multiprocessors Using Network-in-Memory (Feihui Li, et.al. 2006)[8] | 2차원 CPU 구조에서 문제가 되고 있는 내부 연결망 문제를 해결하고자 새로운 L2 캐쉬 구조를 제안한다. 데이터 이동 기법을 활용하는 3차원 L2 캐쉬 구조를 활용하여 2차원 구조에 비해서 IPC를 평균 18% 향상시키는 실험 결과 또한 상세하게 기술한다. |

반도체 공정 기술의 발달로 인하여 트랜지스터 제조 공정이 미세화 됨에 따라 단일 프로세서에 여러 개의 코어를 적재시키는 것이 가능해졌다. 현재 이러한 추세와 맞물려 단일 프로세서에 많은 수의 코어가 적재된 2차원 구조 멀티코어 프로세서가 시장에 상용화 되었다.

일반적으로 2차원 구조 멀티코어 프로세서에서 코어 개수의 증가는 전체 프로세서의 성능을 높여줄 수 있다. 하지만 프로세서 내부에 적재되는 코어의 개수가 증가할 때, 이에 따른 부정적인 측면 또한 존재한다. 기존의 2차원 구조 멀티코어 프로세서에서 코어 개수를 증가시키면 코어 내부에 적재된 제어 및 연산 유닛들 사이의 통신을 위한 내부 연결망의 길이는 상대적으로 증가하게 된다. 이러한 내부 연결망의 증가는 유닛 간의 통신 지연을 발생시킬 가능성이 높다. 또한, 연결망의 길이 증가는 연결망에서 발생하는 전력 소모량을 증가시켜 프로세서의 신뢰성에 악영향을 미칠 수 있다. 위와 같은 문제점 때문에, 내부 연결망의 증가에 따른 통신 지연은 전체 프로세서의 성능을 저하 시키는 주된 요소 중 하나로 지적 받고 있다[5]. 2차원 구조 멀티코어 프로세서에서의 내부 연결망 지연 시간을 감소시키기 위하여 최근에는 3차원 구조 멀티코어 프로세서 제조 기법에 관한 연구가 활발히 진행되고 있다. 3차원 구조 멀티코어 프로세서는 2개 이상의 층이 수직으로 적층된 구조로 구성된다. 각 층에는 프로세서의 핵심 유닛들이 위치하게 되고, 각 층은 수직으로 연결된 TSV를 이용하여 통신이 가능하다. TSV를 이용한 내부 유닛 간의 수직 연결은 내부 연결망에서의 지연시간 증가 문제를 위한 좋은 해결책으로 기대된다.

현재 진행되고 있는 3차원 구조에 대한 연구들은 대부분 2차원 구조를 3차원 구조로 적용하는 경우 발생되는 이점(내부 연결망 감소로 인한 성능 향상 및 전력 소모량 감소 등)과 문제점(전력 밀도 증가 및 온도 문제 등)을 정량적으로 분석하는 데 초점을 맞추고 있다. 하지만, 대부분의 3차원 구조 연구는CPU에 대상이 국한되어 진행되고 있다. 급증하는 연산량을 만족시키는 장점으로 인해 GPU가 크게 각광받고 있는 현 상황에서, 본 논문에서는 GPU를 대상으로 3차원 구조에 대한 연구를 진행하고자 한다. 표 1은 기존에 수행되었던 CPU를 대상으로 하는 대표적인 3차원 구조 연구들을 보여주고 있다. 본 논문에서는 표 1에서 보이는 기존의 연구들과 달리 연구 대상을 GPU로 설정하고 3차원 구조를 적용하는 경우에 발생되는 장점과 단점에 대하여 정량적으로 분석하고자 한다.

### 2. GPU의 메모리 접근 경향

본 논문의 분석 대상인 3차원 구조 GPU는 성능 향상을 위한 기법중 하나로써, 주 메모리와 GPU를 수직으로 적층시킨 형태로 구성된다. 이러한 3차원 구조 GPU는 주 메모리에 접근할 때 TSV롤 통하여 접근을 하기 때문에 2차원 구조 GPU와 비교하여 주 메모리의 접근이 활발히 일어날 때, 더욱 우수한 성능을 제공할 수 있을 것으로 기대된다.

표 2. 기존의 GPU 연구
Table 2. Previous GPU research

| 논문명 | 저자<br>(학회, 발행년도) | 내용 |
|---|---|---|
| A Processor Architecture For Horizon (Horizon Processor) | Mark R. Thistle et al.<br>(SC, 1988) | SISD(Single Instruction Single Data stream) 구조에서 큰 공유 메모리로 인해 유발되는 메모리 접근 지연시간을 문맥 교환(Switching context)을 활용하여 감출 수 있도록 하는 Horizon System을 제안한다. Horizon System에서는 높은 연산 처리량을 확보하기 위하여 FU(Functional Unit)들을 파이프라인화 하고 있을 뿐 아니라, 하드웨어 기법을 통해 자원들의 익명성과 독립성 또한 확보한다. |
| Dynamic Warp Subdivision for Integrated Branch and Memory Divergence Tolerance (DWS) | Jiayuan Meng et al.<br>(ISCA, 2010) | 단일 워프(warp)가 추가적인 레지스터 파일 공간을 요구하지 않으면서 스케줄러의 슬롯 중 하나 이상을 차지하도록 허용하는 DWS(Dynamic Warp Subdivision) 기법을 제안한다. 독립적인 스케줄링 개체(Entity)들은 프로그램 수행 중에 분기를 허용하며, 앞서 적중되어 수행된 스레드들 또한 포함할 수 있도록 허용한다. 이를 통해, DWS 기법은 메모리 수준의 병렬성을 향상시키고 지연시간을 감출 수 있게 된다. 로컬 L1 캐쉬와 공유 L2 캐쉬를 가진 프로세서에서 DWS 기법을 적용하면 칩 공간 오버헤드는 1%에 불과한 반면 성능은 70% 향상된다. |
| Many-Thread Aware Prefetching Mechanisms for GPGPU Applications (MT-Prefetching) | JaeKyu Lee et al.<br>(MICRO, 2010) | 다중 스레드 선출 기법(Many-Thread aware Prefetching)을 활용하여 GPGPU 시스템에 적합한 새로운 하드웨어와 소프트웨어 선인출 기법을 제안한다. Inter-Thread Prefetching이라 불리는 소프트웨어적인 MT-Prefetching 매커니즘은 fine-grained 스레드들 사이의 일반적인 메모리 접근 작용을 이용하므로, 소프트웨어적인 MT-Prefetching 기법은 평균 16%, 하드웨어적인 MT-Prefetching 기법은 평균 15%의 성능 향상을 보인다. |
| On-the-Fly Elimination of Dynamic Irregularities for GPU Computing (G-streamline) | Eddy Z. Zhang et al.<br>(ASPLOS , 2011) | GPU 컴퓨팅에서 생기는 동적 불규칙 문제들을 해결할 수 있는 소프트웨어적인 기법으로 G-Streamline을 제안한다. G-Streamline은 하드웨어 확장이나 오프라인 자료 수집을 요구하지 않는 순수한 소프트웨어적인 해결책으로 불규칙 문제들을 처리하고 최적화 사이에 발생되는 충돌을 해결함으로써 전체 프로그램의 성능을 최대화한다. |

본 논문에서는 GPU에서 주 메모리 접근이 어느 정도 발생할 수 있는지를 분석하기 위해 그림 1에서 보이는 바와 같이 GPU에서 수행 가능한 22가지 벤치마크 프로그램들의 메모리 명령어 비율을 측정해 보았다. 대부분의 벤치마크 프로그램들이 전체 명령어 중 메모리 명령어가 차지하는 비율이 10%가 넘음을 확인할 수 있다. 특히, AES와 같은 경우에는 메모리 명령어의 비율이 최대 75%까지 나타남을 확인할 수 있다. 전체적으로는 평균 30% 정도의 비율을 나타냄을 확인할 수 있다. 이러한 결과를 통해 GPU에서 수행되는 연산 명령의 상당 부분들이 메모리 접근 명령임을 확인할 수 있다. 그러나 GPU의 메모리 접근 방식은 다양하기 때문에, 단순히 메모리 명령어 비율이 높다고 하여 실제 주 메모리 접근이 높을 수 있다고 단정 짓는 것은 다소 무리가 있을 수 있다. 본 논문에서 사용된 벤치마크 프로그램들은 메모리 참조를 수행할 때 CUDA 메모리 구조에 의해 참조하게 된다.

CUDA에서 메모리는 글로벌(Global)/로컬(Local)/컨스턴트(Constant)/공유(Shared)/파라미터(Parameter)/텍스쳐(Texture)와 같이 6개로 구분된다. 이들 중 글로벌 메모리와 로컬 메모리가 GPU의 주 메모리에 공간을 할당받는다. 그러므로 특정 벤치마크에서 메모리 명령어 비율이 높지만, 글로벌/로컬 메모리와 같이 주 메모리 접근이 아닌 캐시 메모리에 존재하는 공유, 컨스턴트 또는 텍스쳐 접근 명령어가 많다면 해당 벤치마크는 2차원 구조 GPU와 비교하여 3차원 구조에서의 성능 향상의 폭이 크지 않을 가능성이 있는 것이다. 그림 2는 벤치마크 별로 메모리 명령어들을 접근하는 메모리 종류에 따라 구분하여 정리한 것이다. 그래프에서 알 수 있는 바와 같이, CP, mri-q의 경우를 제외하고 대다수의 벤치마크 프로그램들에서는 글로벌/로컬 메모리와 같이 주 메모리에 접근하는 명령어(L/G_Store, L/G_Load)의 비중이 높음을 확인할 수 있다. WP와 같은 벤치마크 프로그램의 경우에는 글로벌/로컬 메모리 명령어 비율이 약 94%임을 확인할 수 있다. 또한 Blackschole, NN, FFT의 경우 각각 92%, 90%, 83%의 높은 비율로 주 메모리를 접근함을 확인할 수 있다.

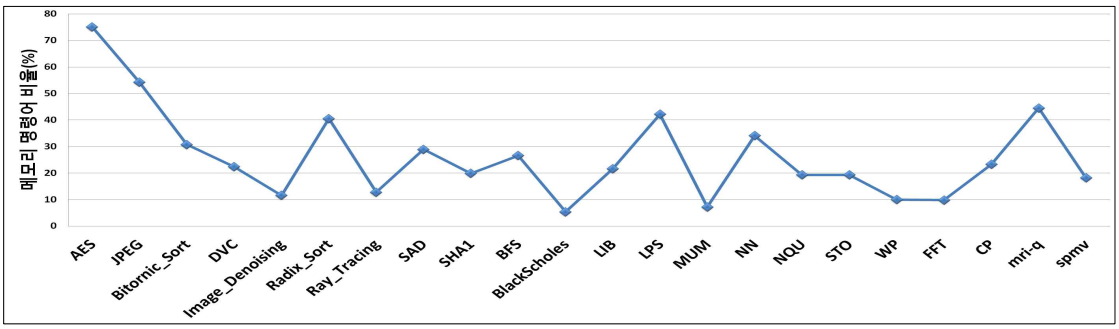



그림 1. 벤치마크 별 메모리 명령어 비율
Fig 1. Memory Instruction rates of the Benchmark

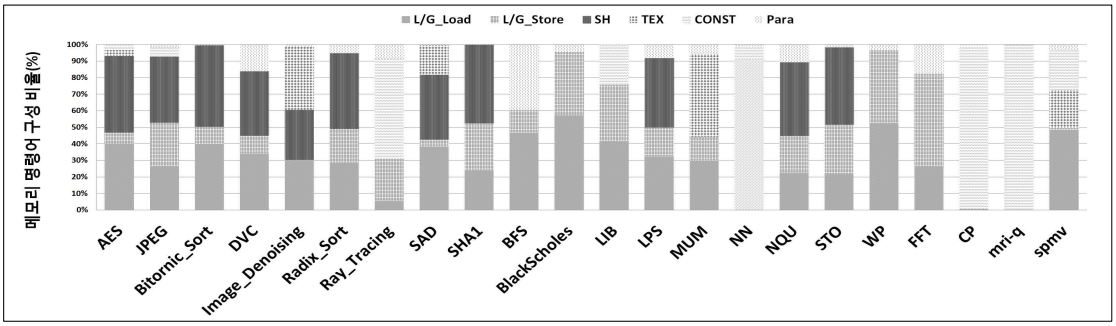



그림 2. 벤치마크 별 메모리 명령어 구성 비율
Fig 2. Memory Instruction Type of each Benchmark

전반적으로 글로벌/로컬 메모리 명령어 비율은 평균 60%의 비중을 차지하므로, 실제 GPU에서 수행되는 프로그램들의 주 메모리 참조율은 매우 높을 수 있음을 예상할 수 있다. 글로벌/로컬 메모리 접근 명령어 외에도, 텍스쳐(TEX), 콘스턴트(CONST), 파라미터(PARA)와 같은 메모리 명령들은 각각의 전용 캐쉬가 존재하여 캐쉬로 접근을 하게 된다. 벤치마크 프로그램들의 메모리 명령어 비율 및, 메모리 명령어 종류에 따른 구성 비율을 분석한 결과, 3차원 구조 GPU의 이점을 활용한다면 대부분의 프로그램들에서 높은 성능 향상을 보일 수 있을 것으로 기대된다.

## III. 실험 환경

### 1. 모의실험 환경

본 논문에서는 3차원 구조 GPU를 통한 성능 향상 효과를 분석하기 위해 Simplescalar[14]를 기반으로 개발된 GPU 성능 평가 시뮬레이터인 GPGPU-SIM[15]을 사용한다. 실험에 사용된 벤치마크 프로그램은 GPU 성능 평가 시 주로 사용되는 ERC와 CUDA SDK에서 총 22개의 프로그램들을 선별하여 이용한다[16-17]. 실험에 사용된 GPU 구성변수들은 표 3서 보이는 바와 같다. GPU의 각 구성요소를 설정할 때에는, 정확한 값을 사용하기 위하여 기존 연구[18]와 Samsung GDDR3 명세서[19]를 참고하여 설정한다.

표 3. GPU 구성 변수
Table 3. GPU Parameters

| GPU Parameters | |
|---|---|
| Number of Shader Cores | 28 |
| Warp Size | 32 |
| SIMD Pipeline Width | 8 |
| Number of Thread / Core | 512 |
| Number of CTAs / Core | 8 |
| Number of Registers / Core | 16384(Byte) |
| Shared Memory / Core | 32(KB) |
| Number of Memory Channels | 8 |
| Bandwidth per Memory Module | 8(Byte/Cycle) |
| DRAM request queue capacity | 32 |
| GDDR3 Memory Timing | |
| tCL=9, tRP=13, tRC=34, tRAS=21, tRCD=12, tRRD=8 | |

### 2. GPU 내부 연결망 모델링

GPU 내부 코어들은 통신을 위해 내부 연결망을 사용하기 때문에, GPGPU-SIM에서는 코어들의 내부 통신을 위한 네트워크 구조를 다양하게 지원하는 시뮬레이터인 Booksim[20]을 사용한다. Booksim은 연결망의 형태, 네트워크를 구성하는 라우터의 특성 등을 설정할 수 있으며 다른 시뮬레이터와 협동이 가능하다. GPU의 통신 네트워크 구조 및 내부 연결망 구성변수는 표 4에 보이는 바와 같다.

표 7 . 내부 연결망 구성 변수
Table 4. Internal Interconnection Network Parameters

| Parameters | Value |
|---|---|
| Topology | Crossbar |
| Routing Mechanism | Dimension Order |
| Routing Delay | 1 |
| Virtual Channels | 2 |
| Virtual Channel Buffers | 4 |
| Virtual Channel Allocator | iSLIP |
| Alloc iters | 1 |
| VC alloc Delay | 1 |
| Input Speedup | 2 |
| Flit Size(Bytes) | 16 |

## IV. 실험 결과

### 1. 성능 변화 분석

각 벤치마크 프로그램별로 분석 대상 구조인 3차원 구조를 통한 GPU의 성능 향상에 대하여 분석한 결과는 그림 3에서 보이는 바와 같다. 그래프를 살펴보면, 3차원 구조 GPU의 성능 향상은 MUM에서 최대 35%까지 증가함을 확인할 수 있다. 또한, BFS, NN, JPEG, WP에서는 각각 30%, 23%, 15%, 10%의 성능 향상이 나타남을 확인할 수 있다. 나머지 벤치마크 프로그램들의 결과에서도 Image_Denoising을 제외하고는 모든 경우에 성능 향상이 나타남을 확인할 수 있다. 하지만, 전체 벤치마크 프로그램들 중에서 메모리 명령어 비율이 가장 높은 AES의 경우에는 약 1%의 매우 낮은 성능 향상을 나타내어 메모리 명령어 비율이 높을수록 3차원 구조 GPU에서 성능 이득을 많이 얻을 수 있을 것이라는 예상과는 다른 결과를 보임을 확인할 수 있다.

위의 결과가 나타나는 이유를 분석하기 위해 메모리 명령어 비율과 실제 벤치마크 프로그램을 수행할 때 발생하는 메모리 접근 비율을 분석하고자 한다. 그림 4는 벤치마크 별로 분석한 사이클 당 메모리 접근 횟수를 보인다. 그림 1에서 분석한 벤치마크 별 메모리 명령어 비율과 그림 4에서 보이는 사이클 당 메모리 접근 횟수 사이에는 상관관계가 거의 없음을 확인할 수 있다. 예를 들어 AES의 경우, 메모리 명령어의 비율은 매우 높지만 실제 사이클 당 메모리 접근 비율은 매우 낮다. 이는 AES가 주 메모리 접근 사이클을 줄인 3차원 구조 GPU에서 성능 향상의 폭이 크지 않을 가능성이 있음을 의미한다. 실제 실험 결과에서 MUM, BFS, NN, JPEG, WP와 같이 높은 성능 향상을 나타낸 벤치마크 프로그램들은 사이클 당 메모리 접근 횟수가 높음을 확인할 수 있다. 싸이클 당 메모리 접근 횟수가 낮은 AES, Bitonic_sort, DVC, NQU, STO, CP, mri-q 와 같은 벤치마크 프로그램들에서는 3차원 구조를 통한 성능 향상이 매우 미미함을 확인할 수 있다. 하지만, 사이클 당 메모리 접근 횟수가 높은 경우에도 3차원 구조에서 매우 낮은 성능 향상을 보이거나, Image_Denoising의 경우와 같이 성능이 하락하는 경우도 발생하고 있다.

한 성능 향상이 매우 미미함을 확인할 수 있다. 하지만, 사이클 당 메모리 접근 횟수가 높은 경우에도 3차원 구조에서 매우 낮은 성능 향상을 보이거나, Image_Denoising의 경우와 같이 성능이 하락하는 경우도 발생하고 있다.

더 자세한 분석을 위하여, 2차원 구조 GPU와 비교하여 3차원 구조 GPU에서 사이클 당 메모리 접근 횟수 증가 양상을 정리한 결과는 그림 5에서 보이는 바와 같다. 그래프에서 나타내는 바와 같이, 3차원 구조 GPU에서는 사이클 당 주 메모리 접근 횟수가 2차원 구조와 비교하여 증가함을 확인할 수 있으며, 이는 Image_Denoising을 제외한 모든 프로그램들에서 성능 증가 양상과 매우 유사한 패턴을 보임을 알 수 있다. 3차원 구조를 통해 주 메모리 접근 지연시간이 짧아지면서, 사이클 당 메모리 접근 횟수 또한 증가하게 되는 것이다. 3차원 구조 GPU에서 MUM, BFS, NN, JPEG, WP와 같이 10%이상의 성능 향상을 보인 벤치마크의 경우에는 사이클 당 주 메모리 접근 횟수 역시 높은 증가율을 나타낸다. 그러나 다른 벤치마크 프로그램들은 위의 벤치마크들과 비교하여 주 메모리 접근 횟수의 증가 폭이 상대적으로 크지 않으며 Image_Denoising의 경우는 매우 큰 폭(약 60%)으로 증가하였으나 성능은 오히려 감소함을 확인할 수 있다.

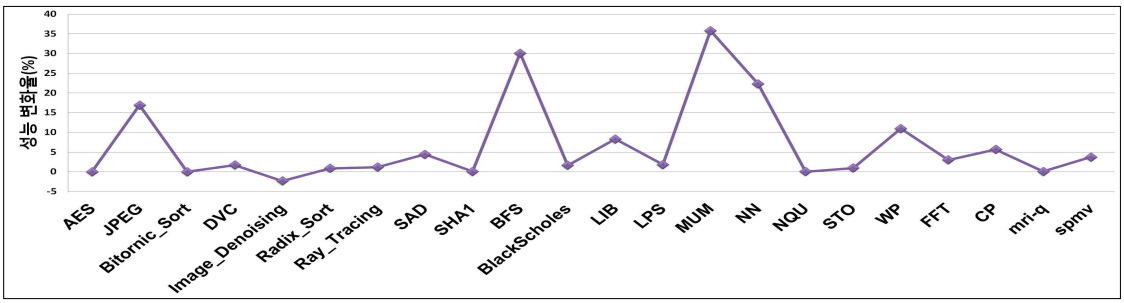



그림 3. 2차원 구조 GPU 대비 3차원 구조 GPU의 성능 향상
Fig. 3. Performance Improvement of the 3D GPU Compared to 2D GPU

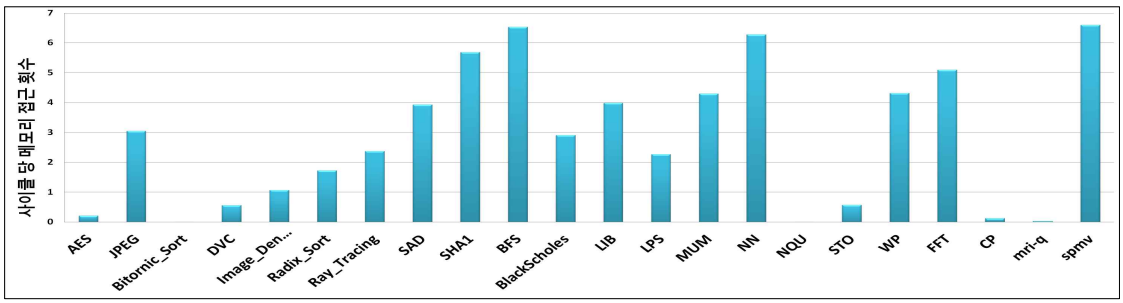



그림 4. 3차원 구조 GPU의 사이클 당 메모리 접근 횟수
Fig. 4. Number of Memory Access per Cycle for 3D GPU

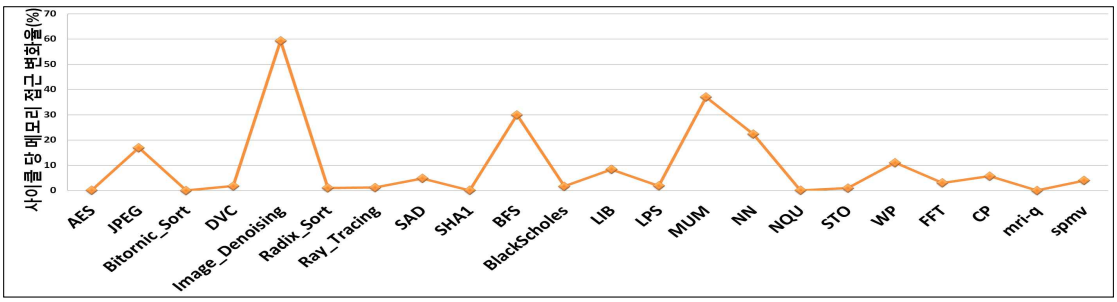



그림 5. 2차원 구조 GPU 대비 3차원 구조 GPU의 사이클 당 메모리 접근 증가율
Fig. 5. Increase of Memory Accesses per Cycle of the 3D GPU Compared to 2D GPU

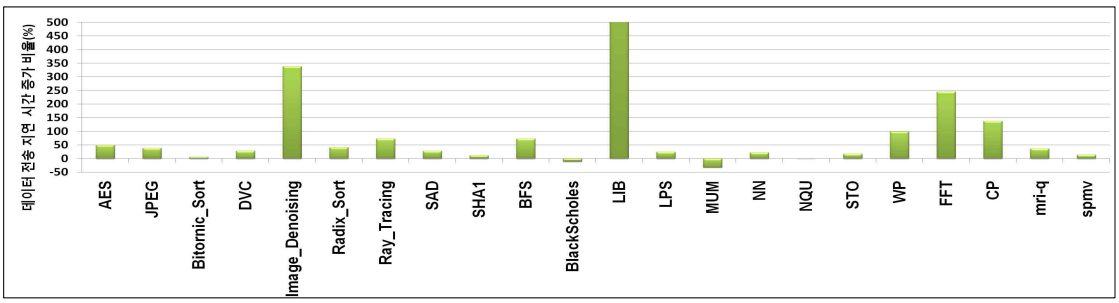



그림 6 . 2차원 구조 GPU 대비 3차원 구조 GPU에서의 데이터 전송 지연시간 증가 비율
Fig. 6. Increased Data Transfer Delay of the 3D GPU Compared to the 2D GPU due to Memory Bottleneck

결과적으로, 높은 성능 향상을 나타내는 벤치마크 프로그램들은 메모리 접근 비율이 높으며, 2차원 구조와 비교하여 3차원 구조에서 주 메모리 접근 지연시간 감소에 의해 메모리 접근 비율이 큰 폭으로 상승한다. 이에 비해 낮은 성능 향상을 나타내는 벤치마크 프로그램들은 메모리 접근 비율이 낮고 3차원 구조 GPU의 가장 큰 이점인 주 메모리 접근 시간의 감소에도 불구하고 접근율 향상이 거의 없다. 또한, 메모리 접근율이 높음에도 3차원 구조에서 접근율 향상이 미미하여 성능적인 이득을 보지 못하는 경우도 존재한다. 그리고 Image_Denoising의 경우와 같이 높은 메모리 접근 비율을 나타내며 3차원 구조를 통해 메모리 접근 비율이 상당히 증가함에도 불구하고, 오히려 성능적인 측면에서는 감소하는 경우도 존재한다. 즉, 3차원 구조 GPU에서 2차원 구조 GPU와 비교하여 낮은 성능 향상을 보이는 이유는 GPU에서 주 메모리 접근 시간의 감소를 통한 전체 수행 사이클 감소 효과를 제대로 활용하지 못하기 때문인 것으로 분석된다.

## 2. 데이터 전송 지연시간

기존의 2차원 구조 GPU와 달리 3차원 구조 GPU에서는 코어와 주 메모리가 TSV를 이용하여 연결되기 때문에, 주 메모리 접근 지연시간이 매우 큰 폭으로 감소한다. 이러한 구조적 특징을 지닌 3차원 구조 GPU는 기존의 2차원 구조 GPU와 비교하여 짧은 시간 내에 메모리의 데이터가 GPU에 도착하므로 이에 따른 메모리 병목 현상이 발생할 가능성이 존재한다.그림 6은 2차원 구조 GPU와 3차원 구조 GPU에서 주 메모리에 요청된 데이터가 코어로 전달될 때 발생하는 메모리 병목 현상으로 인한 데이터 전송 지연시간의 증가 비율을 정리한 것이다. 그래프에서 알 수 있듯이, NQU, MUM 그리고 BlackScholes를 제외한 나머지 벤치마크 프로그램들에서 지연시간의 증가 비율이 매우 높게 나타난 것을 확인할 수 있다. NQU의 경우에는 지연시간 추가 발생 비율이 0%로 아무런 변화가 나타나지 않는데, 이는 그림 4의 결과와 같이 사이클 당 메모리 접근 횟수가 매우 낮기 때문이다. 또한, 사이클 당 메모리 접근 횟수 변화 양상 역시 거의 0%로 변화가 없음을 그림 5에서도 확인할 수 있다.

메모리 접근 활동이 활발하지 않은 경우에는, 3차원 구조 GPU에서의 지연시간 증가 비율 또한 증가할 가능성은 크지 않으며 성능 변화 역시 그림 3의 결과에서와 같이 매우 낮음을 확인할 수 있다. MUM의 경우에는, 지연시간 증가 비율이 오히려 33% 감소함을 확인할 수 있다. MUM은 전체 벤치마크 프로그램들 중에서 가장 높은 성능 향상을 나타내는데, 이러한 높은 성능 향상의 이유 중 하나는 위와 같은 지연시간의 감소 효과에 의한 것으로 분석된다. 그러나 Blackschole은 지연시간이 10% 감소함에도 불구하고 성능 증가율은 1%로 매우 낮은 결과를 나타낸다. 그림 6에서 지연시간이 증가한 벤치마크 프로그램들을 분석하면, 성능이 저하되는 Image_Denoising과 같은 경우는 약 320%의 높은 증가율을 나타내고 FFT, CP, WP의 경우에서도 각각 245%, 137%, 98%의 높은 증가율을 나타낸다. 이 밖에도 대다수의 벤치마크 프로그램들은 평균 31%의 지연시간 증가율을 나타내어 이로 인한 성능 저하가 무시하지 못할 수준임을 확인할 수 있다. 이러한 3차원 구조 GPU에서 발생하는 데이터 전송시간의 지연증가 문제는 주 메모리 접근 지연시간 감소에 의한 성능 향상 효과를 저해시키는 주된 요인으로 분석된다. 즉, 3차원 구조 GPU에서 메모리 인터페이스에서 발생하는 병목현상으로 인한 데이터 전송 지연시간 증가 문제가 성능 향상을 저해하는 가장 큰 원인으로 분석된다.

본 논문에서는 3차원 구조 GPU에서 성능 향상을 저해하는 요소가 메모리 인터페이스에서 발생하는 병목 현상임을 정확히 보임과 동시에 메모리 접근을 효율적으로 하기 위하여 메모리 인터페이스의 수를 변경하여 실험을 수행한다. 본 실험에서 가장 큰 병목 현상이 나타난 응용프로그램인 Image_Denoising 을 사용한다. 또한, 메모리 인터페이스 수 이외의 시스템 구성은 표 3의 값을 그대로 적용한다.

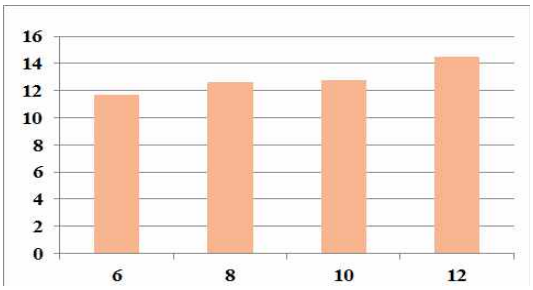



그림 7. 메모리 인터페이스 수에 따른 성능 비교
Fig. 7. Performance comparison with the number of memory interface

그림 7은 메모리 인터페이스 수에 따른 성능 비교 결과를 메모리 인터페이스 개수가 4개인 경우를 기준으로 IPC 향상도를 통해 보여준다. 그래프에서 세로축은 IPC 증가를 나타내며 가로축은 메모리 인터페이스 개수를 나타낸다. 본 논문의 분석 결과와 동일하게, 메모리 인터페이스 수가 4개에서 6개, 8개, 10개, 12개로 증가할수록 성능 또한 비례하여

증가하는 것을 알 수 있다. 그러므로 메모리 인터페이스에서 발생하는 병목현상을 제거할 수 있는 방법을 적용하면, 3차원 구조 GPU의 성능을 더욱 향상시킬 수 있을 것으로 판단된다.

## V. 결 론

본 논문에서는 기존의 2차원 구조 GPU와 비교하여 주 메모리 접근 시간을 줄여줌으로써 성능을 크게 향상시킬 수 있는 3차원 구조 GPU의 성능 변화 양상 및 추가적으로 발생할 수 있는 문제점에 대하여 분석하였다. 실제 GPU에서 수행되는 벤치마크 프로그램들을 사전 분석한 결과로는 주 메모리로의 접근이 많을 것으로 예상되어, 3차원 구조를 통해 GPU의 성능이 크게 개선될 것으로 예상되었다. 하지만, 실제 실험결과에서는 평균 2%의 매우 낮은 성능 향상만을 나타냈다. 이와 같은 결과의 이유를 본 논문에서는 벤치마크별로 나타나는 메모리 명령어 비율이 사이클 당 메모리 접근과는 상관이 없기 때문으로 분석하였다. 실험 결과에서 알 수 있듯이, 높은 성능 향상을 나타내는 벤치마크 프로그램들은 메모리 접근비율이 매우 높음을 확인할 수 있었지만, 메모리 접근 비율이 높음에도 불구하고 성능 향상을 보이지 못한 경우들이 종종 존재하였다. 이는 코어와 주 메모리 사이에서 발생하는 내부 병목현상 문제가 3차원 구조 GPU에서는 2차원 구조와 비교하여 더욱 심화되었기 때문으로 분석된다. 내부 병목현상 문제를 분석한 결과, 3차원 구조 GPU에서 메모리 인터페이스로부터 코어까지 데이터가 전송될 때 발생하는 지연시간이 기존의 2차원 구조 GPU와 비교하여 최대 245%까지 증가함을 확인할 수 있었다. 이러한 결과는 3차원 구조 GPU를 구성할 때, 단순히 구조 변경을 통한 주 메모리 접근 지연시간 감소 효과에 의한 성능 향상만을 고려하는 것이 아니라, 코어와 주 메모리 사이의 병목현상에 의한 데이터 전송 지연시간의 증가 문제 또한 고려해야 함을 의미한다. 이와 같은 문제점을 고려하여 3차원 구조 GPU를 구성한다면, 주 메모리 접근 지연 시간을 줄임으로써 얻을 수 있는 성능 향상의 이점을 충분히 활용할 수 있을 것으로 기대된다.

## 참고문헌

[1] Cena. G, Cereia. M, Scanzio. S, Valenzano. A, Zunino. C, “A high-performance CUDA-based computing platform for industrial control systems,” In Proceedings of the Industrial Electronics 2011 IEEE International Symposium, pp.1169-1174, Gdansk, Poland, Jun. 2011.

[2] Maruyama N, Nukada A, Matsuoka S, “A High-Performance Fault-Tolerant Software Framework for Memory on Commodity GPUs,” In Proceedings of 24th IEEE International Symposium on Parallel & Distributed Processing, pp.1-12, Atlanta, USA, Apr. 2010.

[3] Jishen Zhao, Xiangyu Dong, Yuan Xie, “An Energy-Efficient 3D CMP Design with Fine-Grained voltage Scaling,” In Proceedings of Design, Automation & Test in Europe Conference & Exhibition, pp.1-4, Grenoble, France, Mar. 2011.

[4] D. H. Kim, K. Athikulwongse, S. K. Lim, “A Study of Through-Silicon-Via Impact on the 3D Stacked IC Layout,” In Proceedings of the 2009 International Conference on Computer-Aided Design, pp.674-680, California, USA, Nov. 2009.

[5] Joyner J. W, Zarkesh Ha P, Meindl J. D, “A Stochastic Global Net-length Distribution for a Three-Dimensional System on Chip (3D-SoC),” In Proceedings of the 14th IEEE International ASIC/SOC Conference, pp.147-151, Arlington, USA, Sep. 2001.

[6] K. Puttaswamy, and G. H. Loh, “Thermal Analysis of a 3D Die Stacked High Performance Microprocessor,” In Proceedings of ACM GreatLakes Symposium on VLSI, pp.19-24, Philadelphia, USA, May. 2006.

[7]] J. Kim, C. Nicopoulos, D. Park, R. Das, Y. Xie, V. Narayanan, M. Yousif, and C. Das, “A Novel Dimensionally-Decomposed Router for On-Chip Communication in 3D Architectures,” In Proceedings of the International Symposium on Computer Architecture, pp.138-149, San Diego, USA, Jun. 2007.

[8] F. Li, C. Nicopoulos, T. Richardson, Y. Xie, V.

Narayanan, and M. Kandemir, “Design and Management of 3D Chip Multiprocessors Using Network-in-Memory,” In Proceedings of the International Symposium on Computer Architecture, pp.130-141, Boston, USA, May. 2006.

[9] J. D. Owens, M. Houston, D. Luebke, S. Green, J. E. Stone, J. C. Phillips, “GPU computing,” In Proceedings of IEEE, Vol. 96, no. 5, pp.879-899, California, USA, May. 2008.

[10] M. R. Thistle, B. J. Smith, “A processor architecture for Horizon,” In Proceedings of SuperComputing, Vol. 1, Florida, USA, Nov. 1988.

[11] Jiayan Meng, David Tarjan, Kevin Skadron, “Dynamic Warp Subdivision for Integrated Branch and Memory Divergence Tolerance,” In Proceedings of the 37th annual international symposium on Computer architecture, pp.235-246, Saint-Malo, France, Jun. 2010.

[12] Jaekyu Lee, Lakshminarayana N. B, Hyesoon Kim, Vuduc R, “Many-Thread Aware Prefetching Mechanisms for GPGPU Applications,” In Proceedings of 43rd Annual IEEE/ACM International Symposium on Microarchitecture, pp.213-224, Georgia, USA, Dec. 2010.

[13] Eddy Z. Zhang, Yunlian Jiang, Ziyu Guo, Kai Tian, Xipeng Shen, “On-the-Fly Elimination of Dynamic Irregularities for GPU Computing,” In Proceedings of the 16th International Conference on Architectural support for programming languages and operating systems, pp.369-380, California, USA, Mar. 2011.

[14] D. Burger, T. M. Austin, S. Bennett, “Evaluating future microprocessors: the SimpleScalar tool set,” Technical Report TR-1308, University of Wisconsin-Madison Computer Sciences Department, Jul. 1997.

[15] A. Bakhoda, G. L. Yuan, W. W. L. Fung, H. Wong, T. M. Aamodt, “Analyzing CUDA Workloads Using a Detailed GPU Simulator,” In Proceedings of IEEE International Symposium on Performance Analysis of Systems and Software, pp.163-174, Miami, USA, Apr. 2009.

[16] Chang D. W, Jenkins C. D, Garcia P. C, Gilani S. Z, Aguilera P, Nagarajan A, Anderson M. J, Kenny M. A, Bauer S. M, Schulte M. J, Compton K, “ERCBench: An Open-Source Benchmark Suite for Embedded and Reconfigurable Computing,” In Proceedings of International Conference on Field Programmable Logic and Applications, pp.408-413, Milano, Italy, Sep. 2010.

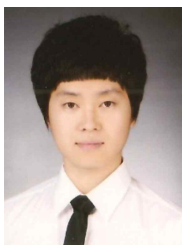
[17] Goswami N. Shankar R. Joshi M. Tao Li, “Exploring GPGPU Workloads: Characterization Methodology, Analysis and Microarchitecture Evaluation Implications,” In Proceedings of IEEE International Symposium on Workload Characterization, pp.1-10, Georgia, USA, Dec. 2010.

[18] Bakhoda A, Kim J, Aamodt T. M, “Throughput-Effective On-Chip Networks for Manycore Accelerators,” In Proceedings of the 43th Annual IEEE/ACM International Symposium on Microarchitecture, pp.421-432, Georgia, USA, Dec. 2010.

[19] Samsung 512Mbit GDDR3 SDRAM, http://www.samsung.com/global/system/business/semiconductor/product/2008/5/22/841580ds_k4j52324qh_rev10.pdf.

[20] Booksim interconnection network simulator, http://nocs.stanford.edu/booksim.html.

## 저 자 소 개



**전 형 규**

2011: 전남대학교 전자컴퓨터공학부 공학사

2011-현재: 전남대학교 전자컴퓨터공학과 석사과정

관심분야: 컴퓨터구조, 임베디드 소프트웨어

Email : hggodman1108@gmail.com

안 진 우

2010: 전남대학교 전자컴퓨터공학부 공학사

2012: 전남대학교 전자컴퓨터공학과 석사

관심분야: 컴퓨터구조, 임베디드 하드웨어 설계

Email : ajw0411@gmail.com

김 종 면

1995: 명지대학교 전기공학사

2000: University of Florida ECE 석사

2005: Georgia Institute of Technology ECE 박사

2005-2007: 삼성종합기술원 전임연구원

2007-현 재: 울산대학교 전기공학부 교수

관심분야: 임베디드 SoC, 컴퓨터구조, 프로세서 설계, 병렬처리

Email: jmkim07@ulsan.ac.kr

김 철 홍

1998: 서울대학교 컴퓨터공학사

2000: 서울대학교 대학원 컴퓨터공학부 석사

2006: 서울대학교 대학원 전기컴퓨터공학부 박사

2005-2007: 삼성전자 반도체총괄 SYS.LSI사업부 책임연구원

2007-현재: 전남대학교 전자컴퓨터공학부 교수

관심분야: 임베디드시스템, 컴퓨터구조, SoC 설계, 저전력 설계

Email : cheolhong@gmail.com