ISSN:1226-7244 (Print)
ISSN:2288-243X (Online)
j.inst.Korean.electr.electron.eng.Vol.21,No.1,92~95,March 2017
논문번호17-01-15
http://dx.doi.org/10.7471/ikeee.2017.21.1.92


# 병렬 알고리즘의 가속화를 위한 GP-GPU의 Thread할당 기법

# Thread Distribution Method of GP-GPU for Accelerating Parallel Algorithms

이 관 호*, 김 치 용**★

Kwan-Ho Lee*, Chi-Yong Kim**★

## Abstract

In this paper, we proposed a way to improve function of small scale GP-GPU. Instead of using superscalar which increase scheduling-complexity, we suggested the application of simple core to maximize GP-GPU performance. Our studies also demonstrated that simplified Stream Processor is one of the way to achieve functional improvement in GP-GPU. In addition, we found that developing of optimal thread-assigning method in Warp Scheduler for specific application improves functional performance of GP-GPU. For examination of GP-GPU functional performance, we suggested the thread-assigning way which coordinated with Deep-Learning system; a part of Neural Network. As a result, we found that functional index in algorithm of Neural Network was increased to 90%, 98% compared with Intel CPU and ARM cortex-A15 4 core respectively.

## 요 약

본 논문에서는 적은 면적의 GP-GPU에서 성능을 향상시키기 위한 방법을 제안한다. 본 논문에서는 superscalar와 같이 과도하게 스케줄링 복잡성을 증가시키지 않는 대신 단순한 코어의 수를 늘려 성능을 극대화 시키는 방법을 제안한다. GP-GPU를 구성하는 Stream Processor의 구조를 단순화한다. 또한, Warp Schedule에서 thread 할당을 어플리케이션에 적합한 방법을 개발하여 성능을 개선한다. 성능을 검증하는 방안으로 neural network의 한 분야인 딥러닝에 대한 스레드 할당방식을 제안한다. Neural Network 알고리즘의 경우 Intel CPU 대비 90%에서 ARM Cortex-A15 4 core 대비 98% 성능 향상을 확인할 수 있었다.

*Key words : Stream Processor, Superscalar, Thread distribution, GP-GPU, Warp scheduler*

## Ⅰ. 서론

GPU의 성능이 향상되면서 다양한 분야에 GPU가 적용되고 있으며 특히, 성능을 극대화하기 위하여 정교한 병렬처리 방법이 중요한 문제가 되었다. 최근에는 스레드 병렬 처리 수준에서 데이터 병렬 처리와 작업 병렬 처리를 결합한 CPU 지원 GPU 스레드 풀 (CAGTP) 모델[1]이 제안되고 있다. 또한 GPU는 많은 양의 프로세싱 리소스를 최대한 활용하기 위해 수천에서 수만 개의 스레드를 동시에 활용하는 기술이 요구된다.

* NEXT CHIP Inc.
** Dept. of Computer Science, Seokyeong University
★ Corresponding author
e-mail: kcy@skuniv.ac.kr tel: ,02-940-7759
Manuscript received Mar. 24, 2017; revised Mar. 29, 2017; accepted Mar. 29, 2017
※ Acknowledgment
This work was supported by Seokyeong University in 2015.

그러나 GPU의 스레드 동시성은 사용 가능한 프로그램 카운터 및 단일 명령어 다중 스레드 스택과 같은 스레드 스케줄링 구조 (스케줄링 제한) 부족 또는 레지스터 파일 및 공유 메모리같은 온-칩 메모리 부족 (용량 제한)으로 인해 감소 될 수 있다. 본 논문에서는 superscalar와 같이 과도하게 스케줄링 복잡성을 증가시키지 않는 대신 단순한 코어의 수를 늘려 성능을 극대화 시키는 방법을 제안한다. 성능을 검증하는 방안으로 neural network의 한 분야인 딥러닝에 대한 스레드 할당방식을 제안한다.

## II. 제안하는 스레드 할당 방법

### 1. 스레드 할당방법

설계한 GP-GPU를 효율적으로 이용하기 위해서는 스레드를 제어하여 GP-GPU에서 처리할 알고리즘을 병렬 처리하여야 한다. 스레드 제어의 경우 알고리즘을 구현할 때 SP마다 할당된 스레드의 id를 이용한다[2]. 스레드의 id는 Special Register를 이용하도록 설계하였다.

GP-GPU를 이용한 알고리즘 병렬 가속화를 진행하는데, 해당 방법은 그림 1과 같다.

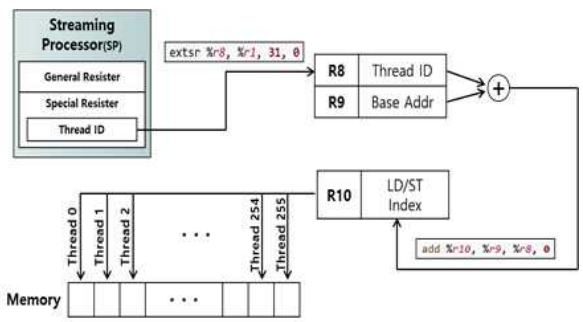


Fig 1. Thread distributing method for parallel acceleration
그림 1. 병렬 가속화를 위한 스레드 할당 방법

16 Warp, 16 Thread를 이용하게 되면 총 256개의 스레드를 이용이 가능하다. 스레드를 이용하고 메모리에 데이터를 Load하거나 Store 할 때 physical_tid를 이용한다. EXTSR 명령어를 통해 Special Register에 있는 physical_tid를 $r8에 저장하고 병렬 처리할 데이터의 주소를 $r9에 저장한다. 이후 두 레지스터에 저장된 스레드 id와 Base Address를 더하고 $r10에 저장하면 형태가 되고 thread id의 경우 offset 역할을 하게 된다. 이는 Base Address 로부터 각각의 스레드 id 만큼 증가한 주소에 있는 데이터를 메모리로부터 Load, 또는 Store 하는 작업을 동시에 처리하게 된다. 동일한 명령을 다수의 스레드를 이용하여 병렬 처리를 이용해 처리 속도를 향상 시킬 수 있다.

### 2. Deep Learning Algorithm의 메모리 접근 패턴

설계한 GP-GPU의 멀티 스레드를 이용해 메모리 접근이 많은 알고리즘과 공통된 연산이 많은 알고리즘을 병렬 가속화하여 처리 성능을 향상시킬 수 있다. 최근 인공지능 연구가 활발히 진행되면서 Neural Network 알고리즘 등 다양한 Deep Learning을 GP-GPU를 이용하여 학습 및 분류를 가속화시키는 방법들이 등장하였다. 하지만 단순히 많은 수의 SP를 이용하여 알고리즘을 처리하는 방법은 하드웨어의 한계가 존재하므로 해당 문제 해결하기 위해선 GP-GPU를 이용하더라도 알고리즘의 최적화 또한 필요하다.

Neural Network의 분류 과정은 다음과 같다.[3] Input layer에 입력 받은 데이터를 가중치(Weight) 값과 바이어스(Bias) 값을 이용하여 Hidden layer에 있는 모든 뉴런에 값을 전달한다. 이후 일련의 과정이 반복되어 최종적으로 Output layer에 출력된 값을 이용한다. 학습 과정의 경우는 분류 과정에서 얻어진 각 뉴런의 출력 값과 에러 값을 비교하여 가중치 값과 바이어스 값을 업데이트 한다.

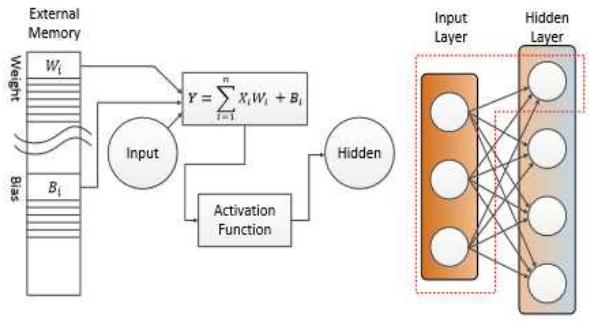


Fig 2. External Memory Access of Neural Network Algorithm
그림 2. Neural Network 알고리즘의 외부 메모리 접근

Neural Network 알고리즘을 처리하기 위해서는 그림 2와 같이 가중치 값과 바이어스 값을 이용하기 때문에 외부 메모리 접근이 많아지게 된다. 특히 망(Layer)의 깊이가 깊어지거나 뉴런의 개수가 많아지면 외부 메모리 접근은 기하급수적으로 증가하게 된다. 분류 과정뿐만 아니라 학습 과정이 포함된 알고리즘을 처리할 경우 외부 메모리 접근은 더욱 증가한다.

### 3. Neural Network 알고리즘의 메모리 접근 패턴에 따른 Thread 할당

본 논문에서 설계한 GP-GPU를 이용하여 Input layer, Hidden layer, Output layer가 존재하고 각 layer마다 3개, 256개, 3개의 뉴런이 포함된 Neural Network 알고리즘을 병렬 가속 처리하기 위한 방법을 제시한다. Neural Network 알고리즘의 경우 가중치와 바이어스의 값이 외부 메모리에 연속적으로 나열되어 있기 때문에 설계한 GP-GPU 포함된 많은 수의 SP를 통한 접근이 더 효율적이다.

그림 3은 Neural Network 알고리즘 병렬 가속화하기 위해 설계한 GP-GPU에서 스레드 할당하는 방법이다. 가중치와 바이어스의 값을 접근하기 위한 주소의 경우 각 $r9, $r11에 저장되어 있다. 이후 스레드를 이용해 메모리에 접근하기 위해 $r8에 저장되어 있는 스레드 id를 각각의 가중치, 바이어스의 저장된 베이스 주소와 합하여 각각 $r10, $r12에 저장한다

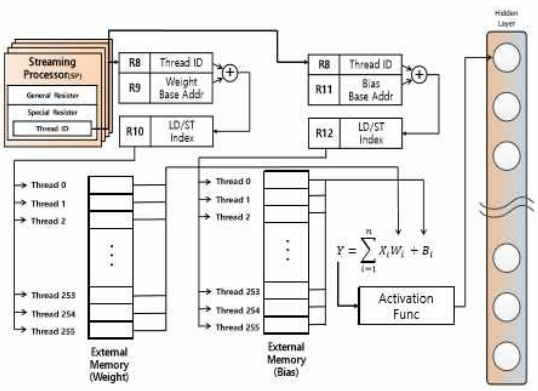


Fig 3. Thread distribution of GP-GPU for Parallel Acceleration of Neural Network

그림 3. Neural Network의 병렬 가속화를 위한 GP-GPU의 스레드 할당

메모리에 연속적으로 저장되어 있는 가중치와 바이어스값을 메모리로부터 읽어 Input layer에 입력된 데이터와 연산을 한 후 Activation function을 통해 최종 결과값이 Hidden layer에 저장된다.

## III. 실험

VC707 보드 상에 GP-GPU를 구현하기 위한 시스템 버스 인터페이스[4][5]는 AXI4와 APB Bus를 사용하였다. 최초 GP-GPU의 기본 초기화 과정과 처리 시작 제어 신호를 처리하기 위해 외부 Host PC의 CPU를 사용한다. Host PC의 CPU는 PCI-E 인터페이스를 사용하여 보드 상의 DDR3 메모리를 초기화 작업을 수행하거나 GP-GPU IP에 레지스터 초기화, 처리 시작 신호등을 전송한다. 전체적인 시스템 구현은 그림 4와 같다.

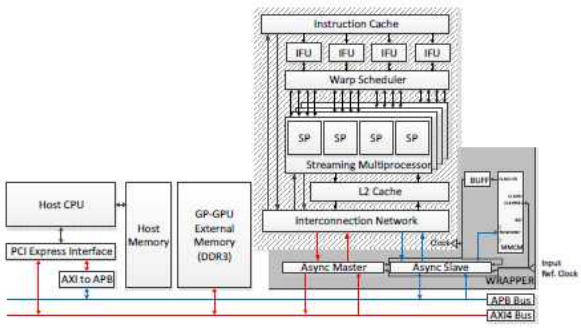


Fig 4. Implementation of FPGA System

그림 4. FPGA 시스템 구현

본 논문의 실험에서는 GP-GPU 제어 프로그램에서 PCI Express를 통해 해당 모듈을 제어할 수 있게 설계하였다. System bus clock은 200MHz로 고정 되어 있으며, GP-GPU IP의 입력 클럭 주파수는 최저 30MHz에서 최고 100MHz로 입력 클럭 주파수의 변경이 가능하다.

설구현한 시스템의 사양과 FPGA의 자원 사용량은 표 1과 같다.

실험에 사용된 Neural Network는 Input layer, Hidden layer, Output layer로 구성되어있으며, 각각 뉴런 3개, 256개, 3개로 구성되어 있다. 총 5000번의 반복 학습 및 분류를 진행하는 Neural Network 알고리즘의 병렬처리 실험 결과는 그림

5와 같다. 그림과 같이 실험에 쓰인 플랫폼 마다 탑재된 코어의 동작 주파수가 다르기 때문에 총 처리 시간의 차이가 발생하여 1회 처리하는데 소모되는 클럭수를 비교하였다. (학습 및 분류 1회 처리 클럭수 = 동작 주파수 x 수행 시간 / 학습 및 분류 반복 횟수) Neural Network 알고리즘의 경우 분류와 학습이 동시에 진행되며 학습 횟수가 증가하면 처리 시간이 증가하지만 분류 성능은 더욱 향상된다.

Table 1. FPGA Resource Usage of GPU
표 1. GPU의 FPGA 자원 사용량

| | Used | Available | Utilization (%) |
|---|---|---|---|
| LUT | 178,677 | 303,600 | 58.85 |
| LUTRAM | 11,099 | 130,800 | 8.49 |
| Flip Flop | 128,863 | 607,200 | 21.22 |
| BRAM | 106 | 1,030 | 10.29 |
| DSP | 112 | 2,800 | 4 |
| IO | 163 | 700 | 23.29 |
| GT | 4 | 28 | 14.29 |
| BUFG | 14 | 32 | 43.75 |
| MMCM | 5 | 14 | 35.71 |
| PLL | 1 | 14 | 7.14 |
| PCIe | 1 | 4 | 25 |

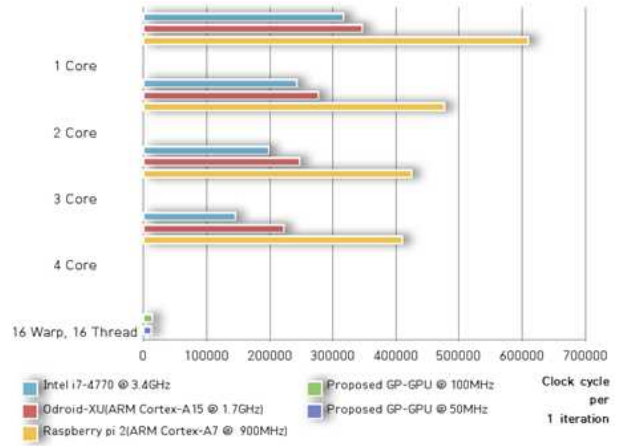


Fig 5. Comparison of Clock Cycle number for training and classification using Neural Network Algorithm
그림 5. Neural Network 알고리즘의 학습 및 분류를 위해 소요하는 클럭 사이클 비교

## IV. 결론

본 논문은 메모리 접근이 많은 애플리케이션을 다수의 스레드를 이용한 병렬 처리를 위한 SIMT 구조의 멀티 코어 GP-GPU의 설계와 접근 방법의 대한 방법을 제안하였다. 설계한 GP-GPU는 범용적인 CPU의 코어 수보다 많으면서 자원 사용량을 최대한 줄이고 설계 면적 대비 성능을 높이고자 SP(Stream Processor)는 16개로 구성하였다. SP는 합성 시 2의 제곱수로 SP의 수를 최대 32개까지 합성할 수 있도록 설계하였다. 설계한 GP-GPU는 Warp Scheduler을 통해 효율적인 스레드 분배를 하여 애플리케이션의 병렬 처리 성능을 향상시켰다. 본 논문 에서는 설계한 GP-GPU를 이용한 병렬 처리 성능을 검증하기 위하여 구현한 GP-GPU와 기존 PC 환경의 Intel i-7 CPU와 ARM Cortex-A15, A7을 사용한 임베디드 플랫폼인 Odroid-XU[6], Raspberrypi 2[7] 플랫폼에서 알고리즘 처리 성능을 비교 검증하였다. 처리 성능을 비교한 Neural Network 알고리즘의 경우 Intel CPU 대비 90%에서 ARM Cortex-A15 4 core 대비 98% 성능 향상을 확인할 수 있었다.

## References

[1] Shuai , Tao Li, Qiankun Dong, Xuechen Liu, Yule Yang, "CPU-assisted GPU thread pool model for dynamic task parallelism," *Networking, Architecture and Storage (NAS), 2015 IEEE International Conference on,* 2015
DOI: 10.1109/NAS.2015.7255234

[2] Seonghyeon Han, Sukwon Yoo, "The parallelization of binarization using a GP-GPU," *The International Journal of Advanced Culture Technology*, vol. 4, no. 4,, 2016

[3] Tariq Rashid, *"Make Your Own Neural Network,"* Hanbit media, 2017

[4] Gyutaek Kyung, "A design of a SIMT architecture based GP-GPU using multi-banked cache memory structure," Master thesis, Seokyeong University, 2015.

[6] Yun-Seop Hwang, Hee-Kyeong Jeon, Kwan-ho Lee, Kwang-yeob Lee, *"Implementation of the SIMT based image signal processor for the image processing," j.inst.Korean.electr.electron.eng, vol 20, no.1, pp89-93, Apr, 2016*

[6] Odroid, "Odroid-XU," http://www.hardkernel.com

[7] Raspberrypi, "raspberrypi," http://www.raspberrypi.org