ISSN: 1226-7244 (Print)
ISSN: 2288-243X (Online)
논문번호 15-03-24
174

j.inst.Korean.electr.electron.eng.Vol.19,No.3,455~459,September 2015
http://dx.doi.org/10.7471/ikeee.2015.19.3.455

SIMT구조 GP-GPU의 명령어 처리 성능 향상을 위한 Dispatch Unit과 Operand Selection Unit설계

Design of a Dispatch Unit & Operand Selection Unit for Improving the SIMT Based GP-GPU Instruction Performance

곽 재 창*

Jae Chang Kwak*

Abstract

This paper proposes a dispatch unit of GP-GPU with SIMT architecture to support the acceleration of general-purpose operation as well as graphics processing. If all the information of an operand used instructions issued from the warp scheduler is decoded, an unnecessary operand load occurs, resulting in register loads. To resolve this problem, this paper proposes a method that can reduce the operand load and the load on the resister by decoding only the information of the operand using a pre-decoding method. The operand information from the dispatch unit is passed to the operand selection unit with preventing register bank collisions. Thus the overall performance are improved. In the simulation test, the total clock cycles required by processing 10,000 arbitrary instructions issued from the wrap scheduler using ModelSim SE 10.0b are measured. It shows that the application of the dispatch unit equipped with the pre-decoding function proposed in this paper can make an improvement of about 12% in processing performance compared to the conventional method.

요 약

본 논문은 그래픽 처리 뿐 만 아니라 범용 연산의 가속화를 지원하기 위한 SIMT 구조 GP-GPU의 Dispatch Unit과 Operand Selection Unit을 제안한다. Warp Scheduler로부터 발행된 명령어에서 사용되는 Operand의 모든 정보를 Decoding 하면 불필요한 Operand Load가 발생하여 레지스터 부하가 발생 한다. 이러한 문제점을 해결하기 위해 Pre-decoding방법을 사용하여 Operand의 정보만을 먼저 Decoding 하여 Operand Load를 줄이고, 레지스터의 부하를 줄일 수 있는 방법을 제안한다. 제안하는 Dispatch Unit에서 나온 Operand 정보들을 레지스터 뱅크 충돌을 방지하는 방법을 적용한 Operand Selection Unit에 전달해 전체적인 처리 성능을 향상 시켰다. Modelsim 10.0b를 이용하여 Warp Scheduler로부터 발행된 10,000개의 임의의 명령어를 처리하여 소요되는 총 Clock Cycle을 측정하였다. 본 논문에서 제안한 Pre-Decoding 기능을 탑재한 Dispatch Unit과 Operand Selection Unit을 적용하여 기존의 방법들 보다 각각 약 11%, 24%의 처리 효율이 증가한 것을 확인 할 수 있었다.

Key words : dispatch unit, pre-decoding, operand, register, operand sselection unit

* Dept. of Computer Science, Seokyeong University, jckwak@skuniv.ac.kr 02-940-7758
Manuscript received Sep. 9, 2015; revised Sep. 17, 2015 ; accepted Sep. 23. 2015
This is an Open-Access article distributed under the terms of the Creative Commons Attribution Non-Commercial License(http://creativecommons.org/licenses/by-nc/3.0) which permits unrestricted non-commercial use, distribution, and reproduction in any medium, provided the original work is properly cited.

Ⅰ. 서론

최근 GPU(Graphics Processing Unit)는 사용자가 원하는 어플리케이션에서 현실감 있는 3차원 그래픽 처리와 고사양의 기능을 제공하고 있다. 그러나 많은 연산량을 요구하는 어플리케이션들이 등장하게 되면서 CPU 단독으로 실행할 때 처리 속도가 늦어지는 문제가 발생하게 되었다. 이를 해결하기 위해 단순히 그래픽 처리 역할만 하였던 GPU를 많은 양의 범용 연산을 병렬로 처리하기 위해 GP-GPU(General Purpose computing on GPU)의 형태로 발전되었다. SIMT(Single Instruction Multiple Thread)구조의 GP-GPU는 하나의 명령어로 여러 Thread를 동작하여 많은 양의 범용 연산을 병렬로 처리해 처리 속도를 높였다[1]. 이러한 구조의 GP-GPU가 등장하면서 명령어 처리에 관련하여 다양한 연구 및 개발이 되어왔다. 본 논문은 SIMT구조의 GP-GPU에서 Warp Scheduler로부터 발행된 명령어를 Decoding 할 때 해당 명령어의 정보를 모두 Decoding 하는 기존의 방법과 다르게 Operand의 정보만 Decoding 하는 Pre-Decoding 방법을 적용한 Dispatch Unit을 제안한다. 제안하는 Dispatch Unit은 명령어를 처리할 때 불필요한 Operand의 Load를 방지하고 레지스터의 부하를 줄일 수 있다. 이 후 레지스터 뱅크 충돌을 방지하는 방법을 적용한 Operand Selection Unit을 설계하여 Dispatch Unit에서 전달 받은 Operand의 정보들을 이용해 레지스터 뱅크 충돌을 방지하여 명령어 처리 성능을 향상시켰다.

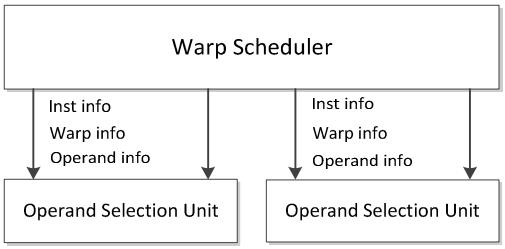

Fig. 1. Flow of the instruction data applying conventional method
그림 1. 기존의 방법이 적용된 명령어 Data 흐름

Ⅱ. 본론

1. 명령어 Decoding 방법

가. 기존의 방법

기존의 명령어를 처리 할 때 일반적인 Decoding 방법은 명령어의 타입, Operand의 종류, 상수 데이터 등을 Load한다[2]. 그림 1은 기존의 방법이 적용된 명령어 Data의 흐름이다. Warp Scheduler에서 발행된 명령어 Data에는 명령어의 정보, Warp의 정보, 두 개의 Operand 정보가 Operand Selection Unit에 전달되어진다. 해당 과정을 수행할 때 Operand가 레지스터를 참조하지 않는 명령어일 경우에도 불필요한 Operand를 Load하게 되는 단점이 존재하게 되는데 이는 레지스터에 부하를 발생시킨다. 특히 SIMT구조의 GP-GPU는 하나의 명령어를 다중 스레드로 처리하는데, 여러 개의 스레드가 동시에 하나의 레지스터에 접근하게 되면 레지스터 접근 충돌이 발생하게 된다. 레지스터 접근 충돌이 발생하게 되면 명령어를 처리하는데 소요되는 총 Clock Cycle이 증가되어 처리성능이 저하되는 단점이 존재한다.

나. 제안하는 방법

본 논문은 명령어를 Decoding할 때 명령어에 관련된 정보를 모두 Decoding 하는 기존의 방법과는 다르게 Pre-Decoding방법을 적용하여 불필요한 Operand의 Load를 줄였다.

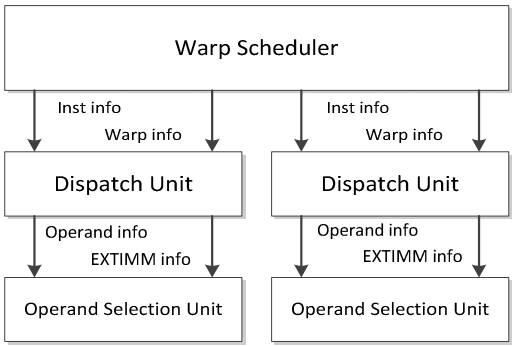

Fig. 2. Flow of the instruction data applying proposed method
그림 2. 제안하는 방법을 적용한 명령어 Data 흐름

그림 2는 제안하는 방법을 적용한 Data의 흐름을 나타낸다. Warp Scheduler 에서 명령어 정보와 실행 중인 Warp의 정보가 제안한 Dispatch Unit에 전달된다. Warp 의 정보에는 현재의 명령어와 사용되는 Thread를 알려주는 Thread Mask, 현재 Warp의 PC 값, Warp의 ID가 존재한다. Dispatch Unit에서는 명령어가 가진 Operand가 레지스터를 참조하는지 확인하여 Operand Fetch Unit에 Operand의 정보가 전달된다. Operand의 정보에는 Operand가 사용되는지를 확인 할 수 있는 Load 신호와 사용될 Operand의 Index 정보가 존재한다. 레지스터의 부하를 막기 위

해 불필요한 Operand Load 방지와 함께 Pre-Decoding을 통해 EXTIMM명령을 지원한다. EXTIMM명령의 경우 확장된 상수(Extend immediate)를 처리하는 명령어이다. 확장된 상수 명령어를 처리할 때 하위 27비트에 담긴 확장된 상수 값을 Operand Fetch Unit에 전달한다. 해당 명령어는 Operand를 사용하지 않는 경우이므로 본 논문에서 제안하는 Pre-Decoding 방법을 사용해 Opcode를 미리 확인하여 Operand Load를 방지한다.

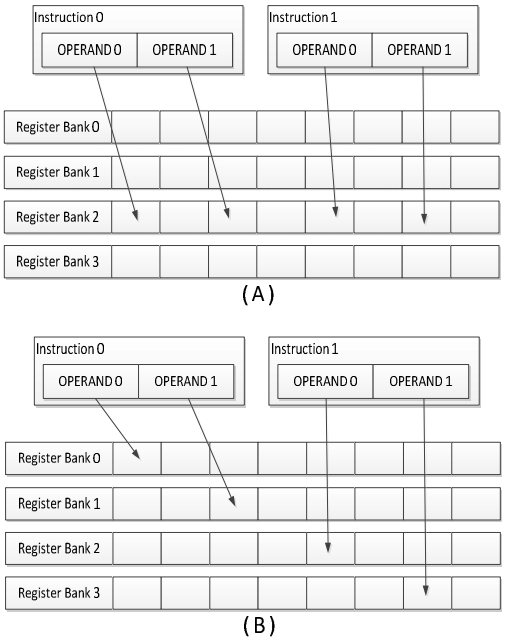

Fig. 3. (A) Register access conflict, (B) Register access non-conflict

그림 3. (A) 레지스터 접근 충돌 발생, (B) 레지스터 접근 충돌 미발생

2. Operand Selection Unit 설계

가. 기존의 방법

Dispatch Unit에서 전달받은 Operand의 정보를 이용해 Pipeline 형태의 Operand Selection Unit에서 명령어 처리를 진행한다. 두 개의 Warp에서 각각 하나의 명령어, 총 2개의 명령어를 처리하기 위해 2개의 Operand Selection Unit으로 구성되어 있다. Operand을 Load할 때 레지스터 뱅크 충돌 문제를 피할 수 없다[3]. 그림 3은 SIMT 구조를 갖는 GP-GPU가 명령어를 처리할 때 레지스터의 접근할 때 발생하는 충돌의 일례를 나타낸다. 그림 3의 (A)와 같이 4개의 Operand가 모두 하나의 레지스터 뱅크에 존재하는 경우 뱅크 충돌이 발생하게 되어 Pipeline Stall이 발생하게 된다. 그림 3의 (B)와 같이 각각의 Operand가 모두 다른 레지스터 뱅크에 존재하는 레지스터를 참조한다면 뱅크 충돌은 발생하지 않게 된다.

그림 4의 (A)는 레지스터 뱅크에 접근할 Operand의 Index를 FIFO에 저장한다. 이 후 FIFO에 저장되어있는 순서대로 레지스터 뱅크에 접근하여 Operand를 Load하는 방법이다. 명령어에서 요구하는 Operand의 개수가 2개 이하인 경우 가장 최소의 Clock Cycle을 소모하게 된다. 하지만 뱅크 충돌이 발생하게 되는 경우 많은 처리 Clock Cycle을 소모하게 된다. 그림 4의 (B)는 한 Clock Cycle에 오직 하나의 Operand Index만 레지스터 뱅크에 접근하도록 설계하여 뱅크 충돌을 제거하는 방법이다. 뱅크 충돌이 발생 하더라도 Full Pipeline 방법으로 설계되었기 때문에 고정된 처리 Clock Cycle을 소모하게 된다. 하지만 명령어에서 요구하는 Operand의 개수가 2개 이하인 경우에도 뱅크 충돌이 발생한 경우와 같은 Clock Cycle을 소모하게 되어 처리속도가 늦어지는 것을 확인 할 수 있었다[4].

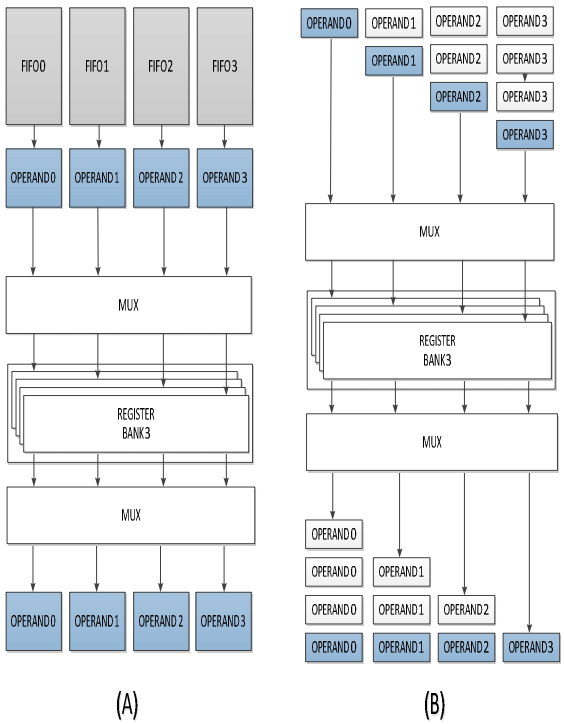

Fig. 4. (A) FIFO Memory based Operand Selection Unit, (B) Full Pipeline based Operand Selection Unit

그림 4. (A) FIFO 메모리 기반 Operand Selection unit, (B) 풀 파이프라인 기반 Operand Selection unit

나. 제안하는 방법

본 논문은 기존에 존재하는 두 가지의 방법을 절충한 그림5와 같은 구조가 가장 효율적임을 확인하였다.

설계한 Operand Selection Unit은 그림 4의 (A)와 (B)의 중간 단계라고 볼 수 있으며 총 4개의 Operand 중 2개의 Operand를 우선적으로 레지스터 뱅크에서 Load한 후에 나머지 두 개의 Operand를 Load 하는 방법으로 처리가 진행된다. 이러한 방법으로 Operand를 Load하게 되면 2개 이하의 Operand를 요구하는 명령어를 처리 할 때에는 최소 4 Clock Cycle이 소요된다. 해당 방법은 레지스터 뱅크 충돌이 발생하지 않은 FIFO Memory based Operand Selection Unit 보다 처리 속도는 많이 느려지지 않지만 추가적인 FIFO를 필요 하지 않아 자원 소비량은 적게 소비된다. 또한 Full Pipeline based Operand Selection Unit 방법과 비교할 경우 Pipeline Stage가 줄어들게 되어 처리 속도가 빨라졌으며 마찬가지로 자원 소비량은 줄어들게 되어 자원 대비 처리 성능이 올라 처리 속도가 빨라진다.

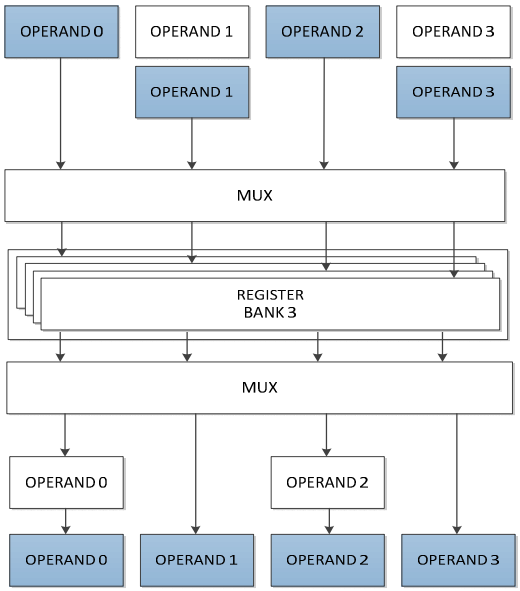

Fig. 5. Proposed Operand Selection Unit
그림 5. 제안하는 Operand Selection Unit

III. 실험 결과

가. 실험 환경

본 논문에서는 기존의 명령어 Decoding방법을 이용하지 않고 Pre-Decoding 방법을 적용한 Dispatch Unit과 이 후 처리되는 Operand Selection Unit을 설계하여 SIMT 구조 GP-GPU를 구현하였다. 구현을 위해 VerilogHDL를 이용하여 RTL 회로를 설계 하였으며, 시뮬레이션 결과는Modelsim SE 10.0b 시뮬레이터를 이용하여 10000개의 임의의 명령어를 발행하는 테스트벤치를 이용하여 시뮬레이션 결과를 얻었다. 합성결과는 Xilinx ISE 14.4 [5] 를 이용하였다.

나. 실험 결과

표 1 은 10000개의 임의의 명령어를 발행하는 테스트벤치를 작성하여 기존의 방법과 본 논문에서 제안한 방법의 총 수행 Clock Cycle를 비교한 결과이다.

기존의 방법 중 (A) 결과는 10000개의 임의의 명령어를 처리할 때 총 7,902회의 수행 Clock Cycle 성능을 보였고 실제로 합성하였을 때 사용된 Flip-Flop의 수는 8,732개였다. (B)의 결과는 총 9,341회의 Clock Cycle 성능을 보였고 총 10,728개의 Flip-Flop의 수를 확인 할 수 있었다. 본 논문에서 제안하는 방법으로 설계된 결과 (C)는 총 8,231회의 수행 Clock Cycle 성능을 보였고 소모된 Flip-Flop의 수는 8,156개였다. 실험 비교 결과 사용된 자원 대비 처리 소요 Clock Cycle이 크게 차이가 나지 않은 것을 확인 하였고 본 논문에서 제안한 방법이 가장 높은 효율임을 확인 할 수 있었다.

Table 1. Comparison of the proposed method and conventional method
(A) Dispatch Unit & FIFO memory based Operand Selection Unit, (B) Dispatch Unit & Full Pipeline based Operand Selection Unit,(C) Proposed Dispatch Unit & Proposed Operand Selection Unit

표 1. 기존의 방법과 제안하는 방법의 성능 비교
(A) Dispatch Unit & FIFO 메모리 기반 Operand Selection Unit, (B) Distpatch Unit & 풀 파이프라인 기반 Operand Selection Unit, (C) 제안하는 Dispatch Unit & 제안하는 Operand Selection Unit

	Flip-Flop	Clock cycle	Flip-Flop/ Cycle
(A)	8,732	7,902	0.442
(B)	10,728	9,341	0.517
(C)	8,156	8,213	0.392

Ⅳ. 결론

본 논문은 SIMT기반의 GP-GPU에서 명령어를 Decoding하는 과정에서 불필요한 Operand Load를 방지하는 Pre-Decoding 방법을 적용한 Dispatch Unit과 레지스터 뱅크 충돌을 방지하기 위한 Operand Selection Unit을 설계 하였다. 명령어의 정보를 모두 Decoding 하지 않고 Operand에 관련된 정보만을 우선적으로 Decoding하여 불필요한 Operand Load를 방지하였고 레지스터 뱅크 충돌을 방지하여 레지스터의 부하를 줄였다. VerilogHDL을 이용하여 제안하는 Dispatch Unit을 적용한 SIMT 구조 GP-GPU를 설계하였다. 성능 확인을 위해 작성한 테스트벤치를 ModelSim SE 10.0b으로 시뮬레이션 테스트를 진행하였다. 사용한 자원대비 처리 효율을 확인하기 위해 Xilinx ISE 14.4를 이용하여 합성결과를 얻었다. 실험결과 Pre-Decoding 방법이 적용된 Dispatch Unit과 레지스터 뱅크 충돌을 방지하기 위한 방법이 적용된 Operand Selection Unit을 사용한 GP-GPU가 기존의 방법들을 사용한 GP-GPU보다 사용된 자원 대비 처리 성능 효율이 각각 약 11%, 24% 증가 함을 알 수 있었다.

본 논문에서 제안한 Pre-Decoding방법을 적용한 Dispatch Unit과 레지스터 뱅크 충돌을 방지하기 위한 방법을 적용한 Operand Selection Unit을 기반으로 설계된 SIMT구조의 GP-GPU에서는 불필요한 연산과 레지스터 접근 및 레지스터 뱅크 충돌을 방지하여 명령어 처리 속도가 향상되는 구조임을 증명하였다.

References

[1] Lashgar. A, Baniasadi. A, Khonsari. A, “Dynamic warp resizing: Analysis and benefits in high-performance simt,” Computer Design, ICCD, 2012 IEEE 30th International Conference, pp. 502-503, 2012.

[2] Xue Yang, Lixin Yu, Wei Zhuang, Yingpan Wu, Li Hao, “Design of instruction decode logic for dual-issue superscalar processor based on leon2,” Consumer Electronics, Berlin (ICCE-Berlin), 2013. ICCEBerlin 2013. IEEE Third International Conference, pp. 1-4, September 2013.

[3] Sohl, J., Jian Wang Karlsson, A., Liu, D., “Conflict-free data access for multi-bank memory architectures using padding,” High Performance Computing (HiPC), 2013 20th International Conference, pp. 425-432, 2013.

[4] Qi Zhang, Li, Qing, Yunyang Dai, Kuo, C.-C.J., “Reducing memory bank conflict for embedded multimedia systems,” Multimedia and Expo, 2004. ICME '04. 2004 IEEE International Conference, pp. 471-474 Vol.1, June 2004.

[5] Xilinx, “VC707 User Guide,” http://www.xilinx.com

BIOGRAPHY

Jae Chang Kwak (Member)

Feb. 1983 BA in Yonsei Univ.
Dec. 1989 MS in Univ. of Iowa
Aug. 1993 PhD in Univ. of Iowa
Mar. 1995 - present Professor at Dept. of Computer Science, Seo Kyeong Univ.
<Research Interest> Network Traffic Control, QoS, Realtime Scheduling, Embedded System