ISSN:1226-7244 (Print)
ISSN:2288-243X (Online)
논문번호17-01-11
j.inst.Korean.electr.electron.eng.Vol.21,No.1,77~80,March 2017
http://dx.doi.org/10.7471/ikeee.2017.21.1.77

슈퍼스칼라 구조를 갖지 않는 고성능 Stream Processor 설계

A Design of a High Performance Stream Processor without Superscalar Architecture

이 관 호*, 김 치 용**★

Kwan-Ho Lee*, Chi-Yong Kim**★

Abstract

In this paper, we proposed a way to improve performance of GP-GPU by deletion of superscalar issue from its original form. At first, we simplified the structure of stream processor in order to eliminate superscalar issue. Under this condition, preservation of hardware size and increasing of thread number were followed by functional improvement of GP-GPU. As the number of thread was getting larger, we proposed the new model of warp scheduler which adjusts the group of thread. This superscalar issue-deleted warp scheduler transferred the instructions to warp which was activated by Round Robin Scheduling. Performance comparison was conducted by Gaussian filtering and the results indicated that our newly designed GP-GPU showing 7.89 times better in its performance than original one.

요 약

본 논문에서는 기존의 superscalar GP-GPU 구조와 달리 superscalar issue를 제거하여 GP-GPU성능을 향상하는 방법을 제안한다. superscalar issue를 제거하기 위해 stream processor의 구조를 단순화했다. stream processor의 구조가 단순화 됨에 따라 하드웨어의 크기를 크게 늘리지 않고 thread 개 수가 늘려 성능을 개선하였다. thread 개수가 늘어남에 따라 thread의 묶음인 warp을 관리하는 warp scheduler 구조를 새롭게 제안하였다. 제안하는 warp scheduler는 superscalar issue가 제거 되어 있기 때문에 warp 활성화 정보를 통해 라운드 로빈 스케쥴링을 통해 활성화 된 warp에게 명령어를 전달한다. 성능 비교는 가우시안 필터링 연산을 사용하였으며 기존의 GP-GPU의 비해 7.89배의 성능향상을 보였다.

Key words : GP-GPU, Warp scheduler, Stream processor, sueperscalar, issue, SIMT

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

Ⅰ. 서론

최근 CPU 성능이 한계에 다다르면서 GPU를 범용 목적으로 개발한 GP-GPU(General Purpose on Graphic Processing Unit)가 고성능 CPU를 대체하고 있다. GP-GPU는 대부분 SIMT(Single Instrction Mutiple Thread) 구조[1]를 가진다. SIMT는 단일 명령어로 여러 개의 thread가 동시에 병렬 처리하는 구조이다. 이 때 thread의 집합을 warp이라 부르며 warp을 제어하는 것이 GP-GPU의 핵심적인 기능이다.

Warp의 제어는 warp scheduler의 기능이다. warp scheduler는 명령어 메모리로부터 명령어를 가지고 오는 명령어 발행과 발행된 명령어를 활성화된 warp에게 전달하는 역할을 수행한다. 명령어를 전달 받은 warp은 stream processor를 통해 명령어를 수행한다.

기존의 연구[2]에서 warp scheduler는 하나의 stream processor에서 두 개의 명령어를 수행하는 superscalar issue 기능이 있었다. superscalar issue는 두 개의 명령어를 동시에 수행할 수 있지만 stream processor의 하드웨어 복잡도가 증가하게 된다.

본 연구에서는 warp scheduler의 superscalar issue 기능을 제거함에 따라 성능향상을 기대할 수 있다.

II. 본론

1. 제안하는 stream processor의 구조

Stream processor는 warp scheduler를 통해 발행 받는 명령어를 수행한다. 기존의 연구에서는 stream processor의 성능을 향상시키기 위해서 superscalar issue를 사용하였다. superscalar issue는 한 개의 stream processor에서 두 개의 명령어를 동시에 수행하는 기능이다. 하지만 많은 기능 모듈이 필요했다. 제안하는 stream processor는 superscalar issue 기능을 제거하고 구조를 단순화 했다. 기존의 stream processor 구조와 제안하는 stream processor의 구조 변화는 그림 1에서 확인할 수 있다.

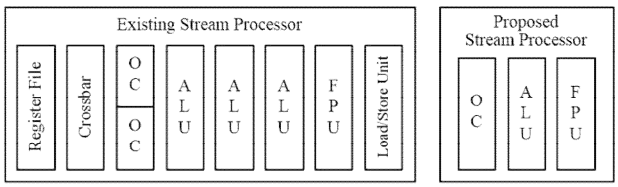

Fig. 1. Existing SP and proposed SP architecture
그림 1. 기존의 SP와 제안하는 SP의 구조

표 1은 기존의 stream processor와 제안하는 stream processor의 자원 사용량과 감소량이다.

Table 1. Existing SP and proposed SP logic usage
표 1. 기존의 SP와 제안하는 SP의 자원 사용량

Logic	Existing SP	Proposed SP	Decrement
Slice register	3,008	1,830	39.16%
Slice LUT	5,612	5,050	10.01%
Block RAM	29	0	100%
DSP	15	6	60%

자원 사용량이 줄어든다는 의미는 하드웨어 복잡도가 낮아졌다는 의미이며 하드웨어 복잡도가 낮아짐에 따라 기존의 GP-GPU보다 많은 수의 stream processor를 병렬화 할 수 있게 되었다. 기존의 GP-GPU는 1개의 stream processor에 8개의 threa로 구성되며, 총 8 개의 stream processor로 구성된다. 기존의 GP-GPU는 총 64개의 thread로 동작하였다. 제안하는 stream processor를 사용한다면 1개의 stream processor에 16개의 thread로 동작할 수 있으며 총 16개의 stream processor을 사용할 수 있게 되어 256개의 thread로 동작할 수 있다. 그에 따라 thread의 묶음인 warp scheduler의 기능이 변경되어야 하며 해당 내용은 본론 2에서 기술한다.

2. 제안하는 Warp scheduler

warp scheduler의 근본적인 기능은 명령어 발행과 전달이다. 명령어 발행은 insturction cache가 명령어 전달은 instruction fetch가 필요하게 된다. 명령어 발행과 전달은 활성화되어 있는 warp에 따라 발행과 전달을 수행한다.

명령어 전달을 위해 stream processor에 개수에 맞는 instruction fetch unit이 필요하다. 본론 1에서 stream processor는 16개 이며 그에 따라 instruction fetch unit을 16개가 필요하다. 명령어 발행을 위해서는 instruction cache가 필요하며, instruction fetch unit 네 개 당 한 개가 필요하므로 총 4개의 instruction cache가 설계되어 있다. 그림 2는 warp scheduler의 구조를 보여준다.

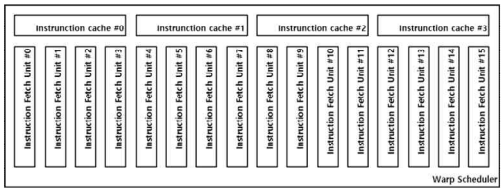

Fig. 2. Proposed warp scheduler architecture
그림 2. 제안하는 warp scheduler의 구조

제안하는 warp scheduler는 superscalar issue가 제거 되어 있기 때문에 warp 활성화 정보를 통해 라운드 로빈 스케쥴링[3]을 통해 활성화 된 warp에게 명령어를 전달한다. 그림 3을 통해 warp scheduler의 동작을 설명한다.

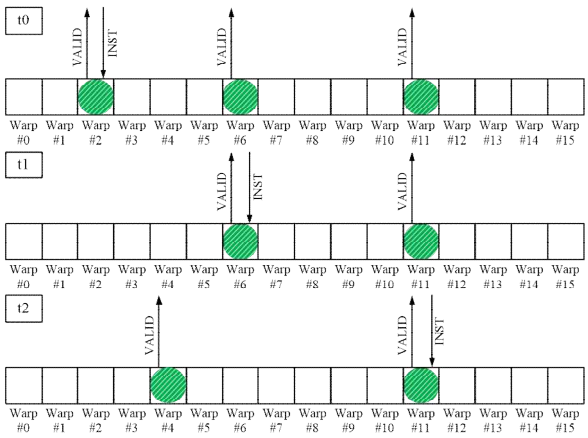

Fig. 3. Warp scheduler process
그림 3. Warp scheduler의 동작

t0의 시간에서 활성화 된 2번, 6번, 11번 warp은 warp scheduler에게 VALID 신호를 보낸다. VALID 신호를 받은 warp scheduler는 활성화 된 2번warp에게 명령어를 전달한다. t1의 시간에서 2번 warp은 비활성화 되었다. 6, 11번 warp은 VALID 신호를 보내고 6번 warp에게 명령어를 전달하게 된다. t2의 시간에서는 6번 warp은 비활성화 되고 4번 warp이 활성화 되었지만 아직 warp scheduler의 명령어 전달이 한 회차를 돌지 못했으므로 11번 warp에게 명령어가 전달되게 된다.

본론 3에서는 본론 1과 2에서 변경된 구조를 활용해 GP-GPU의 성능 비교를 했다.

3. 실험

본론 1과 2에서 제안하는 stream processor와 warp scheduler의 구조와 동작원리를 설명하였다. 해당 구조를 반영한 GP-GPU와 기존의 GP-GPU의 구조와 성능을 비교했다.

GP-GPU의 성능 비교를 위해 사용한 FPGA 보드는 Xilinx 사의 VC-707 board[4]를 사용했다.

Xilinx VC-707 보드를 사용하여 기존의 GP-GPU와 본론 1, 2에서 제안하는 기술을 적용한 GP-GPU를 FPGA로 설계하여 성능을 비교했다. GP-GPU의 제어와 데이터 통신을 쉽게 사용하기 위해서 VC-707 board에 PCIe 버스를 사용했다.

GP-GPU의 명령어는 32bit로 구성되며 PCIe 버스에서 PCIe-AXI bridge와 AXI-APB bridge를 통해 변환되어 전달된다. GP-GPU의 데이터는 128bit로 구성되며 PCIe 버스에서 PCIe-AXI bridge와 AXI interconnect를 통해 DDR 메모리에 저장되며, 내부에서 AXI 버스를 통해 데이터가 이동한다. 기존의 GP-GPU와 제안하는 GP-GPU는 같은 interface를 사용한다. 그림 4는 GP-GPU의 내・외부 인터페이스[5]를 도식화했다.

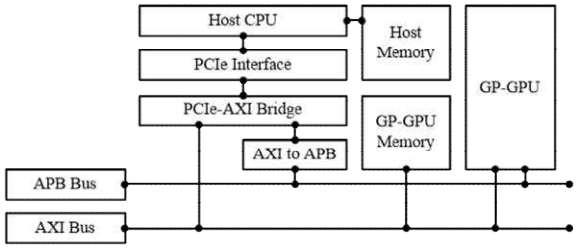

Fig. 4. GP-GPU bus interface
그림 4. GP-GPU 버스 인터페이스

그림 4와 같은 interface를 설계하고 GP-GPU를 FPGA로 설계했다. FPGA 합성에는 Xilinx Vivado 16. 1을 사용했다. 두 개의 GP-GPU의 자원 사용량은 표 2와 같다.

Table 2. GP-GPU logic usage
표 2. GP-GPU 자원 사용량

	Logic	Used	Available	Utilization
Existing GP-GPU	Slice Register	104,051	607,200	17.13%
	Slice LUT	195,948	303,600	64.54%
	Block RAM	95	3,090	3.07%
	DSP	121	2,800	4.32%
Proposed GP-GPU	Slice Register	128,863	607,200	21.22%
	Slice LUT	178,677	303,600	58.85%
	Block RAM	106	3,090	3.43%
	DSP	112	2,800	4%

두 개의 GP-GPU의 자원 증감량을 비교해보면 slice register는 23.84%의 증가, slice LUT는 8.81%의 감소, block RAM은 11.57% 증가, DSP는 7.43%의 감소량을 보였다. slice register와

DSP의 증가는 stream processor의 개수의 증가로 증가한 것으로 예상되며, slice LUT와 block RAM의 감소는 stream processor의 구조의 단순화로 인해 예상된다.

두 개의 GP-GPU에 동일한 application을 수행하여 두 개의 성능 비교를 했다. 수행하고자 하는 application은 가우시안 필터링[6]이며 3x3과 5x5의 필터 사이즈를 필터링 했다. 실험에서 사용된 이미지의 사이즈는 VGA 해상도(640 x 480)이며 32bit의 색공간을 사용했다. 제안하는 GP-GPU의 경우 동작 주파수를 100Mhz로 수행할 수 있으나 동일한 실험 조건을 맞추기 위해 50Mhz로 낮추어 수행하였다.

Table 3. Result of experiment
표 3. 실험 결과

	Filter size	Thread	Time
Existing GP-GPU	3 x 3	64	236 ms
Proposed GP-GPU		256	30 ms
	Filter size	thread	time
Existing GP-GPU	5 x 5	64	307 ms
Proposed GP-GPU		256	76 ms

표 4=3는 기존의 GP-GPU와 제안하는 GP-GPU의 가우시안 필터링을 수행한 결과이다. 3x3 가우시안 필터링의 경우 기존의 GP-GPU는 236 ms, 제안하는 GP-GPU에서는 30 ms의 수행 시간을 보이며 이 것은 7.89배의 성능향상을 보였다. 5x5 가우시안 필터링의 경우는 4.03배의 성능향상을 보였다.

III 결론

본 논문에서는 GP-GPU에서의 superscalar issue 기능을 제거한 후 stream processor의 개수를 늘려 성능 향상을 제안했다. 50Mhz의 동작 주파수에서 가우시안 필터링 연산에 대해 기존의 GP-GPU보다 3x3 필터에서는 7.89배, 5x5 필터에서는 4.03배의 성능향상을 보였다.

References

[1] Sungsu Kim, “Table-based thread reconvergence mechanism on SIMT processer,” Master thesis, Yonsei University, 2011.

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

[3] Honghun Choi, Jongmyon Kim, Cheol Hong, “Analysis of GPU Performance Depending on Warp Scheduling Schemes,” *THE JOURNAL OF KOREAN INSTITUTE OF NEXT GENERATION COMPUTING,* vo 9, no. 2, pp.54-56, Apr, 2013

[4] Xilnx, “VC707 User Guide,” http://xilinx.com

[5] 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
DOI : 10.7471/ikeee.2016.20.1.089

[6] Seon-gye Hwang, *“Image processing by Visual C++,”* Hanbit Media, 2011.