International Journal of Multimedia and Ubiquitous Engineering
Vol.11, No.9 (2016), pp.59-66
http://dx.doi.org/10.14257/ijmue.2016.11.9.07

Design of a Dual-Warp Scheduler for Streaming Multi-Processors Based GP-GPU

Do-Hyun Kim[1] and Jong Joon Park [2*]

[1]*Department of Computer Engineering, Seokyeong University, Korea*
[2]*Department of Computer Science, Seokyeong University, Korea*
kdh3190@skuniv.ac.kr, jong@skuniv.ac.kr

Abstract

In this paper, a warp scheduler is proposed for the improvement of multi-core stream processor based GP-GPU performance. The proposed warp schedulers are divided into odd and even warps, which are issued separately by applying the dual-warp issue. Furthermore, it can simultaneously process up to four instructions because each warp can issue two instructions through superscalar issue. The superscalar issue has a limitation in that it cannot simultaneously process two instructions having data dependence. To solve this limitation, the warp scheduler determines the instruction issuance by testing the issuing condition of the multi-core processor and the read/write register dependence. For scheduling algorithm, the round-robin algorithm was used. To measure the performance of multi-core stream processors, the Gaussian filter mask processing result of the GP-GPU using the proposed warp scheduler was compared with that of the multi-core CPU on various embedded platforms. The experiment results showed that the processing speed of the GP-GPU using the warp scheduler was 6-7 times faster. The GP-GPU also performed better on an image processing application.

Keywords*: GP-GPU; Warp; Scheduling; Round-Robin; SIMT*

1. Introduction

Recently, SIMT (Single Instruction Multiple Threads) architecture for most GP-GPU (General Purpose computing on GPU) implementations, which is developed from the conventional SIMD (Single Instruction Multiple Data) architecture, has been using the same instruction set as SIMD architecture with a set of instructions for processing the vector data in batches through a single execution flow. However, SIMT architecture differs from SIMD in that it can process different data simultaneously through the thread having a plurality of execution flows.

The SIMT based GP-GPU has introduced the concept of warp, which refers to the request of instructions in task unit. Each warp processor uses instructions in its own thread group. The processor uses the SIMT structure to divide tasks that can be accelerated into warps, and assigns each task to warps. The Warp Scheduler inside the processor of the SIMT structure selects a warp to process among the activated warps and transfers it to the streaming multi-processor (SM). Then the SM processes the warp in the multi-thread method using the stream processors that are loaded in the same number as the number of threads assigned to each warp.

This can be expanded to a multi-core SIMT structure by increasing the number of SMs. The design of the Warp Scheduler that efficiently distributes warps to each SM has a great effect on the performance of the multi-core GP-GPU. In other words, the Scheduling Algorithm in the Warp Scheduler is an important factor in the performance of GP-GPU.

* Corresponding Author

ISSN: 1975-0080 IJMUE

Therefore, one can greatly improve the performance of GP-GPU through the intensive study of Scheduling Algorithm and the right selection of an Algorithm.

2. Scheduling Algorithm

The usage of cores depends on the Scheduling Algorithm in a multi-core structured processor. The queuing time for a job in a process becomes longer when the rate of usages of cores is not regular, and the total performance of the system becomes lower. Thus the designer of the system should keep the rate of usages of the cores regular by selecting the right Scheduling Algorithm.

2.1. FCFS (First-Come, First Served Scheduling)

FCFS is the simplest form of the Scheduling Algorithm and is implemented by the method of FIFO (First-In, First-Out) Queue. FCFS is implemented simply by processing the first job first, but the average total queuing time is irregular and is long compared to other algorithms.

2.2. SJF (Shortest-Job-First Scheduling)

SJF processes the shortest job first among the core jobs. The processor first does the job that needs the shortest processing time expected among the multi-core jobs. Also, the average total queuing time can be reduced. So the SJF method is more appropriate for the higher performance of the system by reducing the queuing time. However, SJF should be informed of the expected job processing time before the job's scheduling.

SJF may not be the best algorithm, because it needs extra time for the additional calculation to calculate the expecting processing time, and it may be different from the real processing time, since one cannot know the exact processing time before it has done, and the expected processing time is only usually the approximate time.

2.2. Priority Scheduling

Priority Scheduling is the method of processing jobs by the priority of the jobs that are set by the processor. Priority Scheduling processes the job first that has the lowest priority value, while SJF processes the job first that has the shortest expecting job time. Hence it shows different results in total job processing. The decision of setting the priority value is the most important part in improvement of the performance of a system in Priority Scheduling. The priority values are not decided simply like the expected processing time as in SJF. There are two methods in deciding the priority values. The first method is Internally Defined Priority and their priority values are determined by the features of executing time, resource usage, and dependency features in each core process.

The second method is Externally Defined Priority and their priority values are determined by external features such as process scheduling rule, the rate of power consumption, *etc.* Additional time expenditure may arise to decide the priority values as in the case of SJF. The biggest demerits of this method is the occurrences of starvation, which is the case of never getting to the lowest priority job due to the continuous happenings of higher priority jobs.

2.3. Round-Robin

The system processes each core job within a time limit, and does the next job in turn in Round-Robin scheduling algorithm. The system may change the appropriate processing time according to the job environments and the performance of the system is improved. If a job has not been finished in the given time limit, then the system calls an interruption to do the next core job. The procedure of this processing method is called context switching.

There are a lot of overhead loads on the system if a context switching occurs. Therefore, the system performance of Round-Robin Scheduling would be improved by reducing the occurrences of Context Switching. The average usage ratio of the core processor in Round-Robin Scheduling is higher than the other algorithms, since Round-Robin processes the jobs turn by turn. It has proposed Warp Scheduler which uses a turn by turn way of Round-Robin Scheduling that minimizes the queuing time of each warp.

3. Proposed Warp Scheduler

The proposed Warp Scheduler of the SIMT structure GP-GPU can simultaneously issue two warps by applying the dual-warp issue. The warps are divided into odd and even warps, which are issued separately by two Warp Schedulers. Furthermore, superscalar issues were used so that two instructions with no register dependence can be issued simultaneously for each warp. Using this structure and the Warp Scheduler designed in this paper, up to four instructions can be issued for each stream processor.

3.1. Superscalar Issue

Superscalar issue is a technique for simultaneously processing multiple instructions by running multiple processing elements in parallel. In this paper, the processor performance was enhanced by simultaneously processing four instructions issued from two Warp Schedulers. The superscalar issue has a limitation in that it cannot simultaneously process two instructions having data dependence. To solve this limitation, the Warp Scheduler determines the instruction issuance by testing the issuing condition of the multi-core processor and the read/write register dependence. Each Warp Scheduler transfers the dependence test result to the Instruction Dispatch Unit, which determines the use of superscalar issue.

3.2. Warp Scheduler

The Warp Scheduler receives activated warp numbers and thread numbers assigned to the warps from the host CPU and manages them as active warps and active threads. Furthermore, it receives the initial PC value to get the starting position of the program and reads instructions sequentially or at the positions branched through the branch instruction. Figure 1 below shows the architecture of the Warp Scheduler designed in this paper.

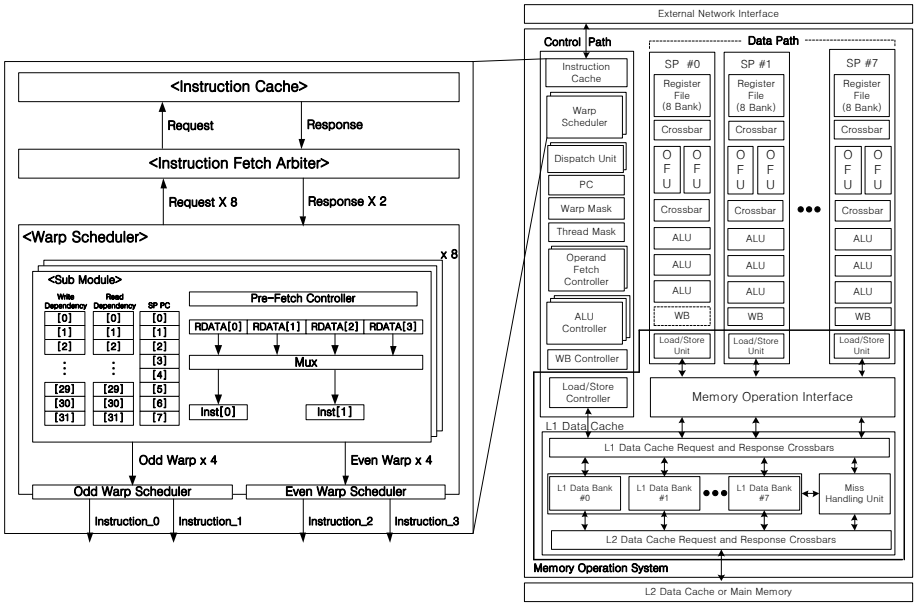

Figure 1. Warp Scheduler

3.3. Warp Scheduler Sub Module

The sub modules of the Warp Scheduler are matched to warps and then manages them. According to this structure, as many sub modules of the Warp Scheduler are created as the number of used warps. In this paper, a total of eight warps were used in the processor, and eight sub modules were created. Of these, four sub modules each are matched to the odd and even Warp Schedulers, respectively.

3.4. Scheduling

Each sub module requests the instruction cache for the address of the main memory from which the required instruction is located through the PC value received from the host CPU, and this request occurs in every sub module that has an active warp. Because it is impossible to simultaneously process the memory access request of every sub module, the requests of sub modules are processed according to the round-robin method. The warp waiting time can be reduced by equally distributing the used amount of each warp using the round-robin scheduling such as Figure 2.

3.5. Scheduling Example

Figure 3 shows the execution result for an example that where Round-Robin Scheduling is applied. This paper proposes a scheduler that can process two sub-modules simultaneously using dual warp issue. In this example sub module 0, 3, 7, 10, 12, 14 are Non-Active. So sub module 1, 2 is Round robin's starting point. In the first step is represented by T0. Sub module 1, 2 are issued using two warp scheduler. In the second step T1, except for sub module 3 - because it is non-active - sub module 4, 5 are issued. In the third step, sub module 6 is issued as an even warp and sub module 9 is issued as an odd warp. Sub module 8 is issued in the next step as even warp with sub module 11.

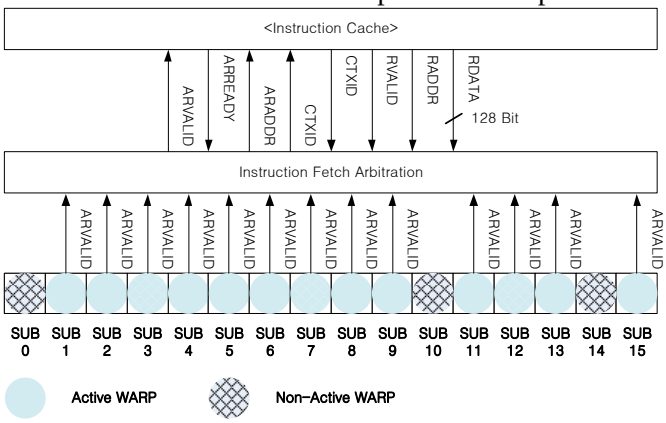

Figure 2. Round-Robin Scheduling

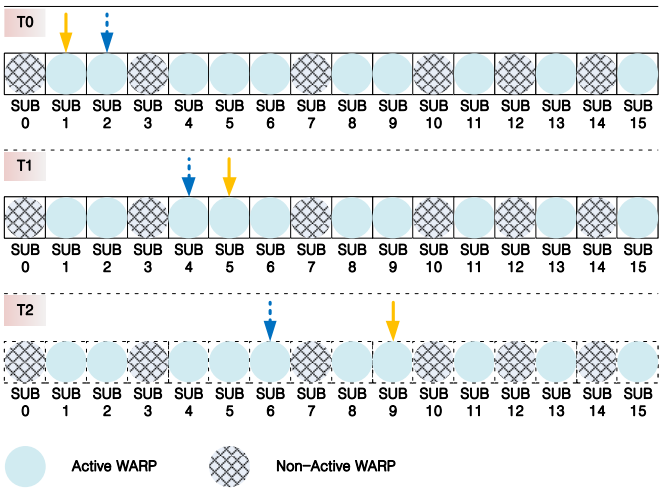

Figure 3. Scheduling Example

4. Results

The usage of a GPU has been extended to general processing by an accelerated computing, which uses high parallel processing, while the previous GPU is simply for the graphic processing. With these procedures, the GPU has developed into a form of GPGPU, and the index of performance is not only for graphic processing but also for general computability.

This paper includes two experiments that have been conducted to confirm the higher performance of Warp Scheduling. The first experiment is to test a general computing for the accelerated computability. The second one is to test a graphic process for the improvement of an image, to enhance the original image.

4.1. General Processing Experiment

The Warp Scheduler designed in this paper can simultaneously process four instructions. To measure the parallel processing performance of instructions, a Gaussian filter mask operation with multiple calculations and high parallelism was used for this experiment. For the experimental environment, Xilinx's VC-707 FPGA platform board was used. Table 1 shows the result of the 5*5 Gaussian filter mask operation using the SIMT based GP-GPU to which the designed Warp Scheduler was applied, compared with the results of the same operation using a multi-core CPU on the embedded platforms. Because the operating frequency of each processor is different, the performance time when calculating at 100 MHz operating frequency was compared for every processor (unit: ms). Table 2 shows the 3*3 Gaussian filter mask operation results in the same environment as in Table 1. The experiment results showed improved performance for both 5*5 and 3*3 Gaussian filter mask operation.

Table 1. Comparison of Processing Time - 5*5 Gaussian Filter Mask Operation (ms)

Processor	1Core	2Core	3Core	4Core
ARM Cortex-A15(1.6GHz)	587.84	298.56	201.60	156.80
ARM Cortex-A9(1.7GHz)	1065.73	621.69	418.37	323.51
ARM Cortex-A9(1.4GHz)	999.60	703.78	590.94	484.68
SIMT based GPGPU(50MHz)	153.625	N/A	N/A	N/A

Table 2. Comparison of Processing Time - 3*3 Gaussian Filter Mask Operation (ms)

Processor	1Core	2Core	3Core	4Core
ARM Cortex-A15(1.6GHz)	441.61	222.88	153.12	124.96
ARM Cortex-A9(1.7GHz)	778.43	444.21	299.03	234.94
ARM Cortex-A9(1.4GHz)	705.18	526.26	428.26	360.08
SIMT based GPGPU(50MHz)	118.17	N/A	N/A	N/A

4.2. Image Processing Experiment

An important application of this new GP-GPU is image processing. It is applied on an image enhancement algorithm to improve the brightness of an image. The size of the image used in this experiment is QVGA (320×240), and the experiment for improving the brightness of an image has been done with the same state as in the general processing environment. Figure 4 is the original image that is used in this experiment, and Figure 5 is the improved image to which was applied the enhancement algorithm with the developed GP-GPU. From the comparison of these two images, it is clear that the roads, the lanes, and the sign post in Figure 5 are more clarified than the original ones. Also the test result of the PSNR of these two images is 17.77 and has been proved by the improvement of the brightness by which one can see the parts which cannot be differentiated in the original image.

Figure 4. Original Image

Figure 5. Enhancement Image

5. Conclusions

The Warp Scheduler is indispensable in GP-GPU for multi-processing according to the SIMT architecture. In this paper, a new Warp Scheduler was designed to process instructions in the superscalar issue. The proposed Warp Scheduler issues warps from the even and odd Warp Schedulers by applying the dual-warp issue. Furthermore, to solve the limitation of the superscalar issue, the Warp Scheduler performs a dependence test and simultaneously issues two instructions for instructions that have no dependence. Through the aforementioned process, the application of the Warp Scheduler designed in this paper can simultaneously process four instructions because two instructions are issued separately from two warps. The warp waiting time was reduced by evenly distributing the used amount of each warp using the round-robin method as the scheduling algorithm.

As a result of the parallel processing performance experiment, the GP-GPU to which the proposed Warp Scheduler was applied to had the processing speed that was at least 2 percent faster than various embedded CPUs in the 5*5 Gaussian filter mask and at least 5 percent faster in the 3*3 Gaussian filter mask. Also, the image has been improved using the improvement of brightness algorithm with GPGPU, and has clarified the various elements of the image from the result of the above experiment.

Increasing the used amount of multiple cores by applying the designed Warp Scheduler when configuring the GP-GPU of multi-core SIMT by expanding the SIMT architecture is expected to improve the processing efficiency of the processor and increase the total performance of the GP-GPU.

Acknowledgments

This work was supported by the Industrial Core Technology Development Program (10049192, Development of a smart automotive ADAS SW-SoC for a self-driving car) funded By the Ministry of Trade, industry & Energy and was supported by Seokyeong University in 2014.

References

[1] M. Garland, S. L. Grand, J. Nickolls, J. Anderson, J. Hardwick, S. Morton, E. Phillips, Y. Zhang and V. Volkov, “Parallel computing experiences with CUDA", IEEE Micro, Vol. 28, no. 4, pp. 13-27, 2008.
[2] B. H. Lee, “A Design of a GP-GPU using the Multi-Threaded Pipeline and the General Register Architecture”, The Graduate School of Seokyeong University, **(2010)**.
[3] J. L. Hennessy and D. A. Patterson, “Instruction-level parallelism and its exploitation”, in Computer Architecture: A Quantitative Approach, 4th ed., San Francisco, CA: Morgan Kaufmann Pub., **(2007)**, pp. 66-153.
[4] A. Levinthal and T. P. Chap, “A SIMD graphics processor”, In SIGGRAPH, **(1984)**, pp. 77-82.
[5] K. Sharma, Saifullah and I. Moon, “GPU-Based Optimization of Self-Organizing Map Feature Matching for Real-Time Stereo Vision”, Journal of information and communication convergence engineering, vol. 12, no. 2, **(2014)**, pp. 128-134.
[6] W. W. L. Fung, I. Sham, G. Yuan and M. Tor, “DynamicWarp Formation and Scheduling for Efficient GPU Control Flow”, Microarchitecture 2007. MICRO 2007. 40th Annual IEEE/ACM International Symposium, **(2007)**, pp. 407-420.
[7] H. J. Seo and H. W. Kim, “Study of Modular Multiplication Methods for Embedded Processors”, Journal of information and communication convergence engineering, vol. 12, no. 3, **(2014)**, pp. 145-153.
[8] W. W. L. Fung, I. Sham, G. Yuan and M. Tor, “DynamicWarp Formation and Scheduling for Efficient GPU Control Flow”, Microarchitecture 2007. MICRO 2007. 40th Annual IEEE/ACM International Symposium, **(2007)**, pp. 407-420.
[9] R. A. Lorie and H. R. Strong, “Method for conditional branch execution in SIMD vector processors”, US Patent, vol. 4, **(1984)**, pp. 435,758.
[10] E. Lindholm, J. Nickolls, S. Oberman and J. Montrym, “NVIDIA Tesla: A Unified Graphics and Computing Architecture”, Micro IEEE, vol. 28, no. 2, **(2008)**, pp. 39-55.

Authors

Do-Hyun Kim, 2014: BS degree in Computer Engineering, Seokyeong University, 2014~present: Master course in Electronics and Computer Engineering, Seokyeong University

Jong-Joon Park, 1978 BS degree in Physics, Sogang University. 1980: MS degree in Electronics Engineering, Yonsei University. 1980~1990: Assistant Professor in the Dept. of Computer Science, College of Industry, Chosun University. 1994: Ph.D in Computer Science, College of Art and Science, Florida State University. 1995~Present: Professor in Computer Science, Seokyeong University.