Advanced Science and Technology Letters
Vol.43 (Multimedia 2013), pp.67-70
http://dx.doi.org/10.14257/astl.2013.43.13

Superscalar GP-GPU design of SIMT architecture for parallel processing in the embedded environment

Kwang-yeob Lee[1], Nak-woong Eum[2], Jae-chang Kwak[1] *

[1]Dept. of Computer Engineering, Computer Science*, SeoKyeong Univiersity,
Jeongneung 4-dong, Seongbuk-gu, Seoul, Korea
kylee@skuniv.ac.kr, jckwak@skuniv.ac.kr*
[2]Multi-media Processor Research Team, ETRI, Daejun, Korea
nweum@etri.re.kr

Abstract. Superscalar GP-GPU of SIMT architecture is implemented on VC707 SoC Platform. GP-GPU of current generation is used for processing graphics as well as a general purpose. GP-GPU becomes a credible alternative to general purpose processors for many applications. This paper proposes a design of superscalar GP-GPU of SIMT architecture for parallel processing in the embedded environment. In this paper, the application is processed parallel and its performance is compared using existing multi-core CPU of the embedded environment and the implemented GPU. The performance of parallel processing with the implemented GP-GPU was improved about 65%.

Keywords: SIMT, GP-GPU, Parallel Processing, Embedded System

1 Introduction

In recent times, parallelization of applications has become essential in an embedded environment. However, as a CPU in an embedded environment has low operating frequency and a small number of cores, the parallelization process is limited. A graphic processing unit (GPU), which was initially only used for graphic data processing, is increasingly being considered for use in general fields. The GPU, which consists of tens or thousands of effective cores, has a simpler structure than the CPU, facilitating easy parallelization [1].

In this study, we developed a GP-GPU with a superscalar single instruction multiple threads (SIMTs) structure for parallelization in the embedded environment. Comparisons between the performance of a conventional CPU in the embedded environment and that of the developed GPU with regard to parallelizing the application, confirmed that the GPU improved the parallelization performance by 65% on average.

2 SIMT GP-GPU Architecture

The structure of the GPU developed in this study is shown in Fig. 1. This GPU with an SIMT structure has 16 stream processors (SPs) per core. Each thread is grouped on the basis of the warp name [2], and two instructions are patched per warp. For a single SP, an odd warp and an even warp are assigned and processed. The GPUs with the conventional single instruction multiple data (SIMD) structure improved the parallelization performance by parallelizing SPs that have several ALUs [4, 5]. Each SP with the SIMD structure receives a single instruction and simultaneously processes several data. However, a module that can control the execution flow by enabling every SP to independently execute different instructions should exist as much as the number of SPs. Meanwhile, for the GPU with the SIMT structure, every SP processes the same instruction. Thus, a single SP control unit can control every SP, reducing the use rate of hardware resources and power consumption.

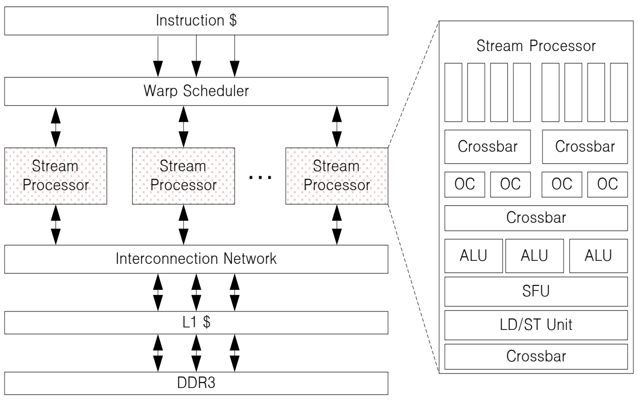

Fig. 1. GPU High Level Architecture

2.1 Latency Hiding

The cache accuracy rate tends to decrease in streaming applications. Thus, if cache misses often occur during memory instruction processing, a pipeline stalls with a decrease in the parallelization performance. Moreover, as hundreds of cycles are required to read data from the memory to the cache, a cache miss leads to performance degradation [6]. This study solved this problem using the multi-threading method as shown in Fig. 2. When a cache miss occurs among threads of a certain warp, the warp waits in the waiting list and a memory request of the following warp is processed until the data are hit. Warps in which the data of all the threads are not hit remain on the waiting list. While processing instructions of other warps, waiting warps are examined for whether every data element is hit by using a round robin method. The warp all of whose threads have hit data is sent to a write back module. As instructions of the following warp are processed in spite of a cache miss in a certain warp, the pipeline can maintain operation without a stall. Furthermore, the latency of the memory access time requiring a long cycle can be hidden.

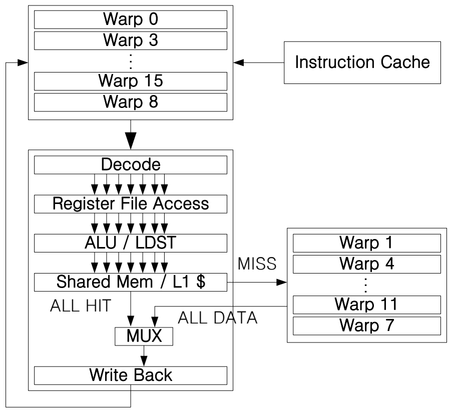

Fig. 2. Method to Memory Latency Hiding

3 Experimental Results

As the experimental environment, the VC707 FPGA Platform by Xilinx was used. The operating frequency in the platform is 50 MHz. A verification application was used for parallelizing an integral image creation algorithm, which is frequently used in the field of image processing and recognition.

Table 1. Comparison of runtime for parallelization of integral image creation (unit: ms)

Heading level	1 Core	2 Core	3 Core	4 Core	Frequency
ARM Cortex-A9	31.24	19.46	14.57	11.46	1.4 GHz
ARM1176JZF	318.14	N/A	N/A	N/A	700 MHz
Ours GPU	150.89	N/A	N/A	N/A	50 MHz

For parallelization of the CPU in the embedded platform, OpenMP was used. An image of 307,200 pixels was converted into the integral image. The experiment result is presented in Table 2. As runtime difference occurred because of the different core operating frequency of each platform used in the experiment, clocks used for processing a single pixel were compared; the experiment result is shown in Fig. 3 (The clock used for processing a single pixel = operating frequency × runtime/number of pixels). As a result of the experiment, it was verified that the GPU increased the parallelization performance by approximately 34% more than when parallelization was performed using the Odriod-X quad core in the embedded platform and by approximately 96% more than when the Raspberrypi single core was used, showing an average performance improvement of 65.7%.

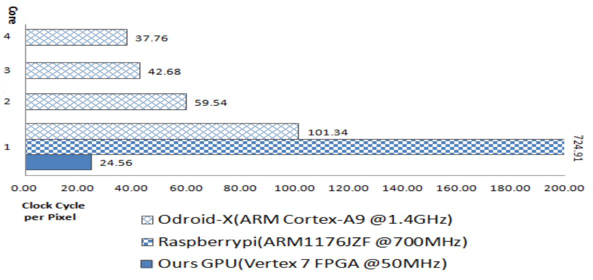

Fig. 3. Results of comparison of clock cycles used for processing a single pixel

4 Conclusion

In this study, we developed a superscalar GP-GPU with an SIMT structure on the Vertex7 VC707 FPGA platform. The parallelization performance of the developed GPU and the conventional CPU in an embedded environment were compared. For the comparison, the execution speed for parallelizing an integral image creation algorithm was considered. It was found that the parallelization performance improved by approximately 65% on average more than when the conventional CPU in the embedded environment was used. Given the experiment result, the GPU is expected to be used for parallelization in an embedded environment in the future. Furthermore, the developed GPU will be modified to be multi-core to increase the effect of parallelization.

Acknowledgments

This work was supported by the IT R&D program of MOTIE/KEIT. [10035152 , Energy Scalable Vector Processor - Primary Technology]

References

1. Advanced Micro Devices, Inc."ATI CTM Guide", 1.01 edition, (2006)
2. Erik Lindholm, John Nickolls, Stuart Oberman, John Montrym,"NVIDIA Tesla: A Unified Graphics and Computing Architecture", Micro IEEE (Volume:28,Issue:2), 39--55 (2008)
3. NVIDIA," CUDA Technology", http://www.nvidia.com/CUDA.
4. A. Levinthal and T. Porter.Chap, "a SIMD graphics processor". In SIGGRAPH, 77--82, (1984)
5. R. A. Lorie and H. R. Strong."Method for conditional branch execution in SIMD vector processors", US Patent 4,435,758, (1984)
6. Wilson W. L. Fung, Ivan Sham, George Yuan, Tor M., "DynamicWarp Formation and Scheduling for Efficient GPU Control Flow", Microarchitecture 2007. MICRO 2007. 40th Annual IEEE/ACM International Symposium, 407--420 (2007)