17-18
SCOPUS

International Journal on Recent and Innovation Trends in Computing and Communication
Volume: 5 Issue: 6

ISSN: 2321-8169
1140 – 1145

Discrete Wavelet Transformation Implementation in GPU through Register Based Strategy

Hemkant Balasaheb Gangurde
Department of Computer Engineering
MET's Institute of Engineering
Nashik, India
gangurdehemkant@gmail.com

M. U. Kharat
Department of Computer Engineering
MET's Institute of Engineering
Nashik, India
mukharat@rediffmail.com

Abstract— The significant architectural changes made by Nvidia during the launch of Kepler architecture in 2012, upgraded its GPUs with greater register memory and rich instructions set to have communication between registers through available threads. This created a potential for new programming approach which uses registers for sharing and reusing of data in the context of the shared memory. This kind of approach can considerably improve the performance of applications which reuses implied data heavily. This work is based upon of register-based implementation of the Discrete Wavelet Transform (DWT) with the help of CUDA and openCV. DWT is the data decorrelation approach in the area of video and image coding. Results of this particular approach indicate that this technique performs at least four times better than the best GPU implementation of the DWT in past. Experimental tests also prove that this approach shows the performance close to the GPUs performance limits.

Keywords- *CUDA, DWT, Kepler, Nvidia, OpenCV*

*****

I. INTRODUCTION

The computational power of GPUs is growing notably day by day. Previously GPUs were only used to decrement the graphics rendering burden due to CAD (computer-aided design) or high graphics video games on CPUs. GPUs are currently used for mainstream applications as well. During the evolution of GPUs, it has gone through major changes in its architecture. The most important change was the release of Compute Unified Device Architecture (CUDA) in 2006 of the Nvidia, which provided architectural tools for general purpose computing together along with C-compiler for the GPU to have GPU programming. While using GPU for the implementation of mainstream application one must have to take care so that the potential of the GPU capacities get fully exploited. Managing the data internally is the important aspect. The important factor in GPU-based implementation is storing required data in the legitimate memory areas. Memory space of GPU is divided into three areas: global memory, shared memory, and register-based memory. Global memory is the largest, situated off-chip DRAM which shows the largest latency. Register and the shared memory are present on-chip and do get managed accordingly. In comparison, they are much faster in comparison with global memory, but their size is much smaller. The important deviation between the register memory and the shared memory is that use of shared memory is more common in order to store and reusing intermediate results and sharing data between threads efficiently. The arithmetic and logical operations are performed in register memory where threads are private in registers.

During the launch of CUDA, Nvidia has released the guidelines [2] which have strongly recommended using shared memory space for the operations such as sharing and reusing of data. These recommendations were challenged by Volkov and Demmel [3][4], who explored that an extreme use of the shared memory may decrement the level of performance. Three factors are responsible for this. The first one is the bandwidth associated with shared memory that, though it is very high, it might act as the bottleneck for the applications which uses previously used data heavily. The next factor is that ALU dependent operations whose data is present in the shared memory space must be required to move it to registers before performing the operations. The last one is that the size of shared memory space is considerably lesser in comparison with register memory space. Volkov and Demmel [3][4] shown that it is possible to gain the GPU's level of performance by directly using the register memory, by decreasing the interference of the shared memory. These results suggested that performance can be maximized by making register memory space as local storage place when data reusing is the must.