A Review of Discrete Wavelet Transformation Implementation in GPU through Register Based Strategy

Hemkant B. Gangurde, Dr. M. U. Kharat

[1]PG Student, Department of Computer Engineering, MET's institute of Engineering, Nashik, Maharashtra 422003
[2]Professor and Head Department of Computer Engineering, MET's institute of Engineering, Nashik, Maharashtra 422003.

---***---

Abstract - *The significant architectural changes made by Nvidia during the launch of Kepler architecture in the middle of 2012 has provided GPUs with a 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 context of the shared memory. This kind of approach can considerably improve the performance of applications which reuses implied data heavily. This work puts a review of register-based implementation of the Discrete Wavelet Transform (DWT), the data decorrelation technique in the field of image and video coding. Results of this particular approach indicates that this method is, at least, four times faster than the best GPU implementation of the DWT in past. Experimental tests also proves that the approach we are studying shows performance close to the GPUs performance limits.*

Key Words: Kepler, GPU, DWT.

1.INTRODUCTION

Computational power of GPUs 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 GPU's 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 has 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 spaces. Memory space of GPU is divided in three areas : global memory, shared memory, and register based memory. Global memory is the largest, situated off-chip DRAM which shows largest latency. Register and the shared memory are located on-chip and does get managed accordingly. In comparison they are much faster than the global memory, but their size is much smaller. The important deviation between the register memory and the shared memory is that the shared memory is more commonly used for storing 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 has strongly recommended to use shared memory for the operations such as sharing and reusing of data. But, 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 of the shared memory that, though it is very high, it might act as bottleneck for the applications which uses previously used data heavily. The next factor is that ALU dependant operations whose data is located in the shared memory must move it to registers before performing the required operations. Third one is that the shared memory space is smaller than register memory space. Volkov and Demmel [3], [4] shown that the its possible to gain the GPU's performance by direct use of the register memory, by minimizing the interference of the shared memory. These results suggested that maximum performance can be obtain when the registers are used as the main local storage space while reusing the data.

In Kepler architecture launched in 2012, the size of the register memory space is made twice as previous, and the number of registers that single thread can able to manage has been made four times as that of the previous, also a new instructions set is introduced to enable the data sharing in the register space. These improvements helped register-based implementations to program the GPU. Which is an emerging programming approach to have the better and faster implementation of the applications which can able to use GPUs efficiently.

The discrete wavelet transform (DWT) is an implementation of the wavelet transform using a discrete set of the wavelets and translations following some defined rules. This DWT transform decomposes the signal into mutually orthogonal set of wavelets, which is the main difference from the continuous wavelet transform or its implementation for the discrete time series sometimes called discrete-time continuous wavelet transform.

A GPU based implementation using Nvidia hardware needs the understanding of CUDA . CUDA(Compute Unified Device Architecture) is a parallel computing platform and application programming interface (API) model launched by Nvidia in 2006. It allows developers and software engineers to use a CUDA-enabled graphics processing unit (GPU) for general purpose processing an approach termed GPGPU (General-Purpose computing on Graphics Processing Units). The CUDA platform provides a software layer which gives direct access to the GPU's set of virtual instruction and parallel computational elements, for the execution of compute kernels. The CUDA platform is compatible with programming languages such as C, C++, and Fortran. This makes it easier for specialists in parallel programming to use GPU resources, in contrast to previous APIs like OpenGL and Direct3D, which required advanced skills in GPU programming. In addition to that, CUDA supports programming frameworks such as OpenACC and OpenCL.

In this paper we are reviewing the register-based implementation strategy for the DWT[1]. Discrete Wavelet Transformation which is a technique for data decorrelation in the area of video and image coding. It's usage found in international compression standards which include JPEG, also in number of coding schemes which includes SPIHT, EBCOT, or SPECK. In order to implement the DWT using GPU as hardware data reuse must be considered significantly. There are many different approaches present in the literature so that data could be reused efficiently. The use of a register based strategy allows a particular approach that differs significantly from the previous methods. In the register based implementation of DWT has to be implemented and rethought from scratch. The most important aspects are partitioning of data and thread-to-data mapping in GPU memory spaces. The strategy which we are reviewing here achieves speedups of 4 times in comparison with the best methods found in the literature.

2.RELATED WORK

Since lot of work and efforts has been taken while providing the efficient way to implement the DWT. In this section we had discussed about existing techniques used to implement DWT.

The implementations before emerging of CUDA, DWT were employed over number of devices and different programming languages based upon GPU as a hardware. The implementation proposed in [6], was based on OpenGL which introduced a decomposition of wavelet and reconstruction algorithm, which directly works on the graphics hardware of OpenGL capable workstations and accelerates the time consuming filtering steps results in saving the time. This particular approach has used the convolution and color matrix extensions together with OpenGL's facilities to scale images during copy instructions, they performed all necessary steps of 2D tensor product wavelet filtering without copying data from or to the machines main memory, thus succeeded in avoiding typical bottlenecks which could occur in the visualization cycle. Open Graphics Library (OpenGL) is a cross language, cross-platform application programming interface(API) to render 2D and 3D vector graphics. The API is commonly used to have interaction with a graphics processing unit (GPU), to achieve faster graphics rendering. Whereas [7], [8] employed OpenGL and Computer graphics together. Lots of these earliest methods were concentrated on convolution based operations.

DWT was evaluated for the first time using lifting scheme [8]. In [8] they demonstrated a simple but powerful and cost effective solution to implement 2-D DWT on the consumer level GPU. No tailor made in that and no expensive DWT hardware is needed to achieve such performance. They have shown that 2-D DWT can be implemented on any SIMD-based GPU comes with normal configuration of PCs. This method unifies the mathematically-different forward and inverse DWT. Different wavelet filter kernels and boundary extension schemes are incorporated by modifying the filter kernel values and indirect address table respectively. [8] also demonstrated 2D-DWT applicability in wavelet-based geometric deformation, stylized image processing, texture-illuminance decoupling, and JPEG2000 encoding. Though the convolution based approach was popular previously as the lifting scheme requires intermediate values to be shared among coefficients. Tools were unavailable to implement lifting scheme efficiently in the past. Which was confirmed in [9] experimentally, in which both the convolution and the lifting approach were implemented and compared for the performance together.

The general-purpose GPU architecture as well as associated programming tools were lacking in the previously mentioned pre-CUDA implementations. The operations present in the DWT must be related to graphics operations, which found to be limited in past. Even these works were far more faster compared to a CPU-based implementation, their performance is could have been more with current GPUs that have an advanced memory hierarchy and support for general purpose GPU computing. One of the most important thing in current CUDA based implementations is the way image gets partitioned in order to allow parallel processing scheme. There are three main schemes employed in the for partitioning, called row-column scheme, row-block scheme, and block-based scheme.

The first implementation based upon CUDA of DWT was mentioned in [10]. Which uses the row column scheme. At start, a thread block loads a row of the image into shared memory then threads compute the horizontal filtering on that row. The first CUDA based implementation which combines lifting scheme as well as block-based scheme to implement DWT was addressed in [11]. The important factor of this approach was that it decreases transfers to the global memory as it evaluates horizontal and the vertical filtering in a one computation step. In this method the image is partitioned in rectangular blocks that needs to be loaded in the shared memory by a thread block. Then both horizontal filtering and the vertical filtering are applied to these blocks, in this their is no need of further memory transfers or need of a matrix transpose operation. The drawback in this kind of approach is that there is dependencies of data between blocks which are next to each other. These dependencies are not taken care in [11]. The simple solution is to extend all blocks with some rows and columns which overlaps with adjacent blocks.

The fastest implementation of the Discrete Wavelet Transformation was present in the literature, which was explored in [12]. In this approach the row-block scheme was used, which also extends to any number of dimensions. The method tries to maximize coalesced memory access. They compared their method to an optimized CPU implementation of the lifting scheme, to another (non-CUDA based) GPU wavelet lifting method, and also to an implementation of the wavelet transform in CUDA via convolution. They implemented method for both 2D and 3D data. The method is scalable and was shown to be the fastest GPU implementation among the methods considered. The first step involved in this approach is similar to the row-column based scheme. In that it first loads rows of an image to the shared memory in order to apply the horizontal filtering. Data are then stored in global memory segment. The second step shows similarity with block-based scheme, which do partition the image into vertically stretched blocks that are loaded to the shared memory. Rectangular blocks which are next to each other in the vertical axis are operated by the same thread block. This makes possible to thread block to reuse data which is situated at the borders of the blocks, results in handling the previously mentioned problem of data dependency of data. The setback of this particular approach is its requirement of two steps, which results in requirement of more accesses to the global memory. They have compared their implementation to convolution-based implementations and to the row-column scheme. Results conclude that the lifting scheme with the row-block partitioning scheme is one of the faster method.

Implementation in [1] is where it took to the next level considering the architectural changes made by Nvidia. [1] introduced an implementation of the DWT in a GPU through a register-based strategy. This kind of implementation approach has recently become possible in the latest CUDA architectures as they have expanded register memory space and also introduced new set of instructions. The most important feature of this method is the use of the register based memory space to carry all operations and an effective use of block-based partitioning scheme and thread to data mapping operation which permits the assignment of warps to process all data of a block. Experimental results indicate that the register-based strategy proved to be better in performance than the use of shared memory as it requires less number of instructions and able to achieve higher GPU utilization.

3.EXISTING METHODOLOGY

The DWT is one of the signal processing technique which is derived from the Fourier transformation analysis. DWT applies various banks of filters to the input which then decompose the low frequencies and high frequencies of the signal. In the field of image coding, the forward operation of the Discrete Wavelet Transformation is applied to the original set of pixels present in an image in the first step of the encoding operation. Generally, coding systems use a dyadic decomposition of the DWT which results into a multiresolution representation of the image. This kind of representation organizes the wavelet coefficients in different levels of resolution and subbands that capture features of the image in the vertical, horizontal, and diagonal form. To reconstruct the image decoder is applied at last stage. Depending upon the filter bank used, determines some of the features of the transform. Most commonly, irreversible CDF 9/7 and the reversible CDF 5/3 filter banks are used. The method[1] we are reviewing implements both these filter banks. This system deals with the implementation of DWT which is the core part in most of the systems.

The DWT can be implemented using either convolution operation or by the lifting scheme. The lifting scheme decreases the usage of memory and the number of operations performed, thats why it is commonly employed in DWT. It performs numerous steps on a discretely-sampled one-dimensional signal, which is represented by an vector. Every step results into the (intermediate) wavelet coefficients which are then assigned to the even, or to the odd, positions of the an array. Each coefficient is computed using three samples: the even (or odd) position of the array, and its adjacent neighbors. This procedure can be repeated number of times based upon the filter bank used. The most important advantage of the lifting scheme is that all coefficients in the odd (or even) positions can be calculated parallely as they do not have any dependencies among them.

Most of the modern implementations, do use the lifting scheme. The approach we are studying here stores the image partitions data in the register memory. It's very much known that the row-column scheme is not much popular due to its slower nature. The results also do indicate that the block-based is not much effective as it uses large amount of shared memory. This is the important reason while introducing the row-block scheme. This analysis is for CUDA architectures before the kepler architecture. Both the row-block and the block based schemes allow data transfers from/to the global memory. The important deviation here is the number of global memory accesses required. The row block scheme requires the read and write operation of the image (or the LL subband) two times. Entire data are accessed in a row-by-row technique in the starting step. After the horizontal filtering performed, the data are returned to the global memory space. The whole image is accessed again using vertical blocks so as to perform the vertical filtering. The images which are larger in width, it is more efficient to divide the rows in slices that are processed independently due to fulfill the memory requirements.

4.CONCLUSION

We studied the DWT(Discrete Wavelet Transformation) implementation by various ways including non-CUDA based approaches. Furthermore, we studied how the lifting scheme for DWT implementation is better than the convolution approach adopted previously before introduction of GPU based architecture. In this paper we studied, what are the main reasons behind the DWT implementation using register based strategy and how the architectural changes made by Nvidia helped for adopting this particular approach.

REFERENCES

[1] Pablo Enfedaque, Francesc Aul-Llinas and Juan C. Moure, "Implementation of the DWT in a GPU through a Register-based Strategy", in IEEE Trans.Parallel Distrib. Syst, 2015.

[2] Nvidia, CUDA C Programming guide, 2014. Available: http://docs.nvidia.com/cuda/cuda-c-programming-guide.

[3] V. Volkov and J. W. Demmel, "Benchmarking GPU's to tune dense linear algebra", in Proc. ACM/IEEE Conf. Supercomputing, Nov. 2008, pp. 3142

[4] V. Volkov, "Better Performance at Lower Occupancy", in IEEE Int. Conf. Image Process., 2010.

[5] F. N. Iandola, D. Sheffield, M. Anderson, P. M. Phothilimthana, and K. Keutzer, "Communication-minimizing 2D convolution in GPU registers", in Proc. IEEE Int. Conf. Image Process., Sep. 2013, pp. 21162120.

[6] M. Hopf and T. Ertl, "Hardware accelerated wavelet transformations", in Proc. EG/IEEE TCVG Symp, 2000.

[7] A. Garcia and H.-W. Shen, "GPU-based 3D wavelet reconstruction with tileboarding", in Vis. Comput, 2005.

[8] T.-T. Wong, C.-S. Leung, P.-A. Heng, and J. Wang, "Discrete wavelet transform on consumer-level graphics hardware", in IEEE Trans. Multimedia, 2007.

[9] C. Tenllado, J. Setoain, M. Prieto, L. Pinuel, and F. Tirado, "Parallel implementation of the 2D discrete wavelet transform on graphics processing units: filter bank versus lifting", in IEEE Trans. Parallel Distrib. Syst, 2008.

[10] J. Franco, G. Bernabe, J. Fernandez, and M. E. Acacio, "A parallel implementation of the 2D wavelet transform using CUDA", in Proc. 17th Euromicro Int. Conf. Parallel, 2007.

[11] J. Matela, "GPU-based DWT acceleration for JPEG2000", in Proc. Annu. Doctoral Workshop Math. Eng. Meth. Comput. Sci, 2009.

[12] W. J. van der Laan, A. C. Jalba, and J. B. Roerdink, "Accelerating wavelet lifting on graphics hardware using CUDA,", in IEEE Trans. Parallel Distrib, 2011.