Adapting data processing methods to modern GPU architecture

Bartłomiej Szczepaniak

Lodz University of Technology
Institute of Applied Computer Science
ul. Stefanowskiego 18/22, 90-924 Lodz, Poland
bartlomiej.szczepaniak@p.lodz.pl

1. Abstract

Wavelet transform have a wide area of application in many scientific areas, for example signal processing, image compression [6] or data mining [4] [5]. Present requirements demand preforming large amount of calculations in the minimum time. For that reason the goal of this paper is to present an approach that will fulfill mentioned requirements, by adapting algorithms for signal processing to GPU demand.

The article analyzes potential benefits of usage of the implementation of lattice structure on Graphics Processing Unit (GPU) and compares obtained results with the implementation on Central Processing Unit (CPU).

2. Introduction

Nowadays signal processing is applicable in many scientific areas like computer tomography, medical ultrasonography or data compression [2]. Among signal processing methods we can find such approaches like sorting, calculation of Fourier and wavelet transform, convolution or correlation. The increasing popularity and demand of signal processing caused the necessity of adapting this methods to requirements established by modern technologies.

For that reason it appeared necessary to consider adaptation signal processing methods to the modern architecture of Graphics Processing Units (GPU). Implementation algorithms on GPU has one significant advantage over standard CPU implementation - multithreading. Execution of multiple threads is also possible on modern multi-core processors (CPU), however it will never be as efficient and fast as multithreading performed by GPU. The article compares efficiency obtained by implementing specific algorithms in CUDA (designed for Nvidia GPU) with implementation of the same algorithm in C language executed on standard Central Processing Unit. The main issue analyzed in this work is implementation of the lattice structure being a model representing discrete wavelet transform (DWT). Article analyses obtained results and determines next steps that should be performed in future experiments and scientific work in order to develop described approach.

3. Lattice structure

Nowadays there is plenty models representing wavelet transforms or discrete wavelet transform (DWT). The simplest model is a two channel filter bank. It is a structure having one input and output calculated according to the values obtained from filters h (highpass filter) and g (lowpass filter). As a result of such filtering we can obtain two signals having the same length as the input signal. In order to obtain the same signal length on the output, decimation after each filter is necessary.

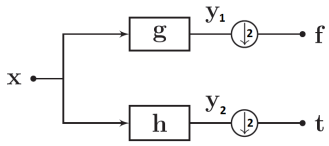

Figure 1. two channel filter bank with with decimation factor

Another model representing wavelet transforms may be wavelet transform matrix or a lattice structure which was analyzed in this article. The Lattice Structure is a mathematical schema enabling, for instance, an implementation of wavelet transform. It was introduced by prof. Yatsymirskyy in one of his publications [1]. the basis for this structure is a two-point base operation D_1 presented below.

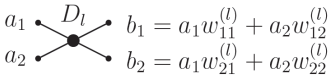

Figure 2. A two-point base operation [1]

Output of two-point base-operation (b_1, b_2) is calculated on the basis of two input values $(a_1,\ a_2)$ and proper factor values (ω_{11}, ω_{12}, ω_{21}, ω_{22}). The lattice structure analyzed in this article consist of 3 layers, with 3 different base-operations (D_1, D_2, D_3). Factor values which were used in experiments described in this article are presented in the Table 1.

Factor	D_1	D_2	D_3
ω_{11}	0.924508	0.479286	-0.105301
ω_{12}	0.381162	0.877659	0.994440
ω_{21}	-0.381162	0.877659	0.994440
ω_{22}	0.924508	-0.479286	0.105301

Table 1. Factor values

Schema presenting architecture of lattice structure is presented in the picture below. In analyzed experiments the input was a set of random float numbers (the same set for CPU implementation as well as for the GPU solution).

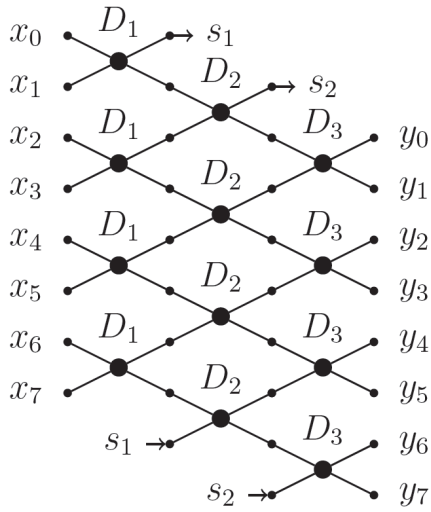

Figure 3. The lattice structure [1]

Data size	CPU time	Clock ticks CPU	GPU time	Clock ticks GPU	Ratio 1	Ratio 2
50 000	20206,36	52490,15	4227,69	10982,3	4,78	4,77
1 million	49890,92	129601,9	4584,14	11908,25	10,88	10,88
2 millions	107941,16	280399,3	4835,98	12562,45	22,32	22,32
3 millions	140645,8	534652,4	4153,84	15790,45	33,86	33,85

Table 2. Obtained results

4. Implementation

Multithreading implementation of lattice structure requires applying changes in the standard architecture of algorithm. As it was presented in the picture, after each layer in lattice structure, a displacement of one variable takes place. The first output variable from each layer is, at the same time, an input variable for the next layer. Such displacement seems to be very tricky in multithreading implementation. For that reason, in order to eliminate inconvenient displacement, an additional set of input values has been added to the initial set of random float numbers. This new, additional set was actually a copy of the beginning of the input set of variables. Such change does not extend considerably calculations of algorithm and seem to be the easiest solution for mentioned problem.

In multithreading implementation the main aim is to execute as many parallel calculations as possible. Applying lattice structure for a set of two-million set of input variables makes it possible to perform one million calculations (two-point base-operations) at the same time. This benefit should exhibit an advantage that GPU multithreading implementation should have over standard CPU implementation.

Investigations of two approaches in implementations were performed on two types of hardware listed below:

- CPU(Intel Core i7 CPU 2,67 GHz)
- GPU (NVIDIA NVS 3100M)

During the experiments, not only the time of calculations was measured. In order to get precise differences between calculations on CPU and GPU, time needed to sent data between host and device in GPU implementation was also taken into account. The time precision of performed calculations was 10^{-3} s.

Table presenting results obtained during the experiment consist of 7 columns (Table 2). The first one (”Data size”) represents input data size (amount of float random numbers being in the input set). Second and fourth columns (”CPU time”, ”GPU time”) represent time of lattice structure calculations obtained on CPU and GPU. Column third and fifth (”Clock ticks CPU”, ”Clock ticks GPU”) contains number of clock ticks obtained when both implementations were performed. Sixth column (”Ratio 1”) represents the ratio between GPU and CPU clock ticks whereas the last column (”Ratio 2”) contains ratio between CPU and GPU time.

5. Conclusion

As we can see from obtained results (Table 2), it is highly advisable to consider adapting data processing methods to modern GPU architecture. In the example analyzed in this article the GPU implementation of lattice structure for 1 million elements appeared to be ten times faster than standard implementation of

the same structure performed on CPU. Future experiments should take into account an efficiency comparison of wavelet transforms using the lattice structure and the lifting scheme which is also acknowledged in literature as a very fast method [3]. Further development of multithreading approach in fast data processing methods may have significant influence in such scientific areas like computer tomography, medical ultrasonography, data compression. It is easy to imagine a plenty of scientific examples in which it is necessary to analyze and calculate 1 million elements (numbers). For instance in image compression if we take into account a standard two-dimensional image which size is 1000 x 1000, than it will be necessary to analyze 1 million pixels. As it was proofed in the article in such size of problems, applying wavelet transform for GPU architecture may decuple the acceleration of calculations.

References

[1] M. Yatsymirskyy *Lattice structures for synthesis and implementation of wavelet transforms*,Journal of Applied Computer Science, 17(1):133-141, 2009.

[2] J. Stolarek *Synteza falek ortogonalnych na podstawie oceny przetworzonego sygnału*,rozprawa doktorska, Politechnika Łódzka, 2011.

[3] G. Uytterhoeven, D. Roose, A. Bultheel *Wavelet Transforms Using the Lifting Scheme*,Report ITA-Wavelets-WP1.1 (Revised version), 1997.

[4] T. Li, Q. Li, S. Zhu, M. Ogihara. *A survey on Wavelet Applications in Data Mining*, SIGKDD Explorations, Volume 4, Issue 2, 2002.

[5] P. K. Dash, Iian Lee Wen Chun, M. V. Chilukuri *Power Quality Data Mining Using Soft Computing And Wavelet Transform*, TENCON 2003.Conference on Convergent Technologies for the Asia-Pacific Region, Bangalore, India, 2003.

[6] A. R. Colderbank, I. Daubechies, W. Sweldens, Boon-Lock Yeo *Lossless Image Compression Using Integer to Integer Wavelet Transform*, Proceedings International Conference On Image Processing, Volume 1, 1997.

Author

mgr inż. Bartłomiej Szczepaniak
Institute of Applied Computer Science,
Lodz University of Technology
ul. Stefanowskiego 18/22,
90-924 Lodz, Poland
bartlomiej.szczepaniak@p.lodz.pl