GPU Implementation of the STA Algorithm on I/Q Data

M. Lewandowski, P. Karwat
Department of Ultrasound
Institute of Fundamental Technological Research PAS
Warsaw, Poland
mlew@ippt.pan.pl

J. Kudelka, T. Kleczek
Informatics
University of Warsaw
Warsaw, Poland

***Abstract*— GPU computing is a new paradigm in high performance signal and image processing. Massive parallel processing offered by the GPUs provides high acceleration of computations when they are properly implemented. Ultrasound image reconstruction is one of these highly parallel classes of algorithms. Massive amount of multichannel input data and deterministic order of execution makes US applications good candidates for high performance GPU implementation. Our goal is to design a versatile ultrasound platform with GPU real-time processing. The project is based on a new system architecture allowing for bandwidth and scalability of processing power. We implemented and optimized SAFT image reconstruction algorithms on CUDA. The study shows that a single GTX-580 GPU card is capable of reconstructing the 128-channel I/Q data into 256×256 High Resolution Images (HRI) at a frame-rate of 44.64 fps which is an equivalent of 5700 Low Resolution Images (LRI) per second.**

Keywords*— ultrasonic imaging; synthetic aperture; GPU; CUDA

I. INTRODUCTION

The medical ultrasound scanners are complex electronic systems with multi-channel signal acquisition and real-time processing. Up to now a huge throughput and an enormous number of operations necessary to perform the imaging functions (beamforming) required application of hardware processing. Nowadays, due to the rapid development of software processing, in particular the GPU processors, a fully software implementation of these functions became possible. However, it should be noted that obtaining an efficient algorithm implementation on the GPU is not an easy task. The complexity of the internal architecture and memory hierarchy of the GPUs causes that slight code variation can cause dramatic changes in the performance.

A growing interest in the implementation of the ultrasound imaging algorithms on the GPU is observed in the literature. Kim et al. [1] presented the GPU implementation of a complete path of the ultrasound imaging based on the classical single-line beamforming method. Performance of the algorithm was tested on the Nvidia GTX-285 card. For the stored 64-channel RF data the obtained frame-rate was equal to 55 Hz. It provided the real-time operating at the same rate as the ultrasonic data are acquired. Yiu et al. [2] implemented a synthetic aperture imaging algorithms and plane wave compounding on a set of three GPU cards (2x Nvidia GTX-480, 1x Nvidia GTX-470). For the 32-channel beamformer the resulting rate of the 512×255 LRI (Low-Resolution Image) generation was 4700-3000 Hz for 5-15 cm depth, respectively. When the beamformer size was increased to 128-channel, the LRIs were obtained at the rate of 1400-800 Hz for mentioned depths. In another publication So et al. [3] reported about the result of 2200 Hz for reconstruction of 512x256 LRI images of 9 cm depth from 128-channel data with use of a single GTX-480. The paper also presented a comparison of the performance and the energy efficiency of the synthetic aperture algorithm implementations on different platforms - CPU, GPU, CPU+GPU and multi-GPU. The problem of data throughput between the CPU and the GPU processor, which limits the processing performance of the system as a whole, was highlighted in all these works.

In this study we implemented an efficient tool for reconstruction of synthetic aperture images [4], which will be applied in our newly developed ultrasound platform [5]. The aim was to limit the bandwidth requirements and to obtain the highest possible frame-rate. For this purpose a number of algorithms were implemented, optimized and then compared in terms of their efficiency.

II. IMPLEMENTATION

A. System Architecture

All the algorithms were implemented in C++ with the aid of Nvidia CUDA C++ API. We chose CUDA over OpenCL because a lot of work is done implicitly and thus it is easier for the developer. This has the drawback of being able to run only on Nvidia GPUs. However, although the OpenCL runs on a wider range of devices it doesn't ensure an optimality of its work. There are also some low level problems like the inability to use pinned memory which can cause additional limitations.

The data processing was performed on Nvidia GTX-580. The utilized PC hardware was as follow:

- Motherboard - Asus Supercomputer P6T7 WS,
- CPU - Intel Core i7 960 @ 3.2 GHz,
- RAM - 6 GB DDR3.

B. *Algorithms*

The synthetic aperture technique is known for its improved imaging quality when compared to classical beamforming. However, it also involves a huge computational load. Despite the continuously increasing computing power of modern GPU, the real-time reconstruction of ultrasound multi-channel data is still a difficult task. To get the highest frame-rate possible to obtain with use of a certain GPU the issues of data bandwidth, number of the read/write operations and computational load need to be balanced.

Systems working on high-frequency data, i.e. the raw RF signals or obtained by means of the Hilbert transform I/Q signals (I - in phase, Q - quadrature), are often limited with their memory bandwidth. However, the I/Q data can be produced with use of the quadrature demodulation which returns a signal whose spectrum is moved to lower frequencies. Therefore, a decimation can be applied. In most cases it solves the problem of limited memory bandwidth. However, the use of the low-frequency I/Q data requires to perform the phase correction [5] which involves additional calculations of the sine and cosine values in the main loop of the algorithm. Nevertheless, it might be a fair price for reduction of input data size and possession of the data format which is potentially useful for other processing e.g. Doppler. The quadrature demodulation is computationally demanding task and calculating it with the GPU would obviously negatively influence the overall system performance. Nonetheless, since our platform uses FPGAs for data accumulation, its free logic resources can be designed to calculate the demodulated signal.

A number of further improvements were developed in order to shorten the time needed to reconstruct a single HRI. Most of the used acceleration techniques were applied in order to minimize the cost of data readings and writings that the algorithm has to perform. This was achieved by means of several optimizations.

- For the classical synthetic aperture imaging every pair of transducers participates in the data acquisition twice, exchanging roles of transmitter and receiver. The data obtained at these acquisitions are processed with use of the same time delay values. Thus, instead of doing the reconstruction twice for each transducer pair, the I/Q signals can be summed in pairs before the reconstruction. This, in turn, reduces the input data nearly by half. The idea is presented in fig. 1. Each transmitter-receiver pair is represented by a single element of the scheme. The I/Q data related to the elements placed symmetrically with respect to the scheme diagonal, are summed together: $IQ_{COMPR}(n_{TX},n_{RX}) = IQ(n_{TX}, n_{RX}) + IQ(n_{RX}, n_{TX})$.
- The spatial locality is optimized by issuing readings that are close to one another on the same GPU SM (Streaming Multiprocessor), which allows to utilize the L1 and L2 cache. This is done by batch processing of few adjacent horizontal image lines.
- The writing operations are nearly eliminated by saving the processing results in per-chip registers and accessing them only at the very end of the processing. This in turn eliminates the issue of reading/writing race conditions.
- Multiple low-level optimizations are made. These operations include setting proper compiler flags (e.g. -use_fast_math), using specialized functions (e.g. sincosf) and minimizing the number of registers needed to improve overall occupancy.
- Computation of the indexes for the reading operations is accelerated by a memory trade-off by preprocessing the distances and saving them in shared memory or computing inner frames in the right order.

The above optimizations are used in NAIVE implementation of the reconstruction algorithm. They proved to be effective, however the resulting performance is still far from expected. Therefore, further improvements are developed and introduced in DIST-SHARED and STRIPES versions of the algorithm.

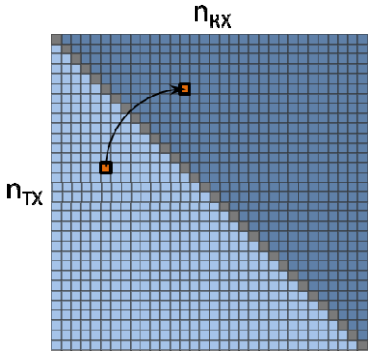

Figure 1. Transmission-reception scheme for the STA technique. The symmetry of the scheme allows for an initial compression of the data.

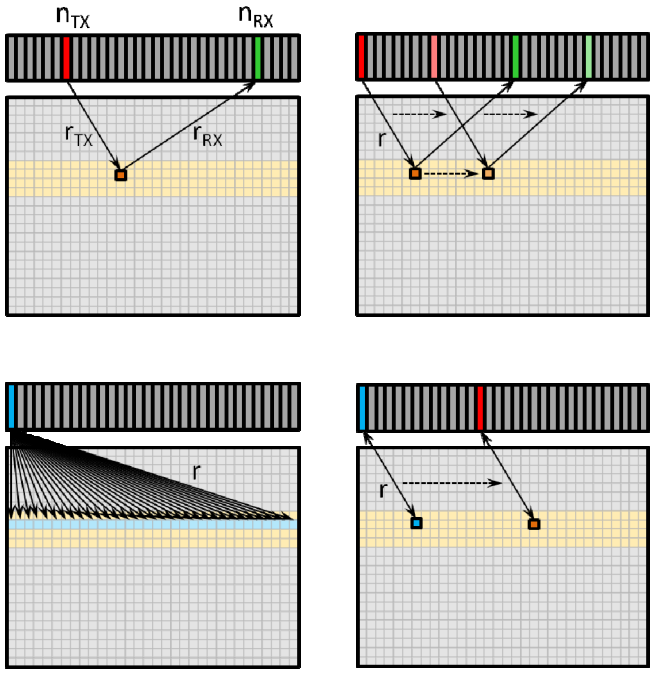

Figure 2. Schemes of a) the standard STA approach implemented in the NAIVE algorithm, b) the STRIPES optimization, c) and d) the DIST-SHARED optimization.

The DIST-SHARED initially computes the distances from the first transducer to every pixel (fig. 2c) in a considered image horizontal line and saves them in the shared memory. When the distances from other transducers are to be calculated, their equivalents are simply read from the memory (fig. 2d).

The STRIPES optimization groups transmitter-pixel-receiver sets into geometrically identical families. Delays for each family are computed only once. The idea of the algorithm is shown in fig. 2b.

C. *GPU Framework*

The GPU Framework, where the stream of processing is defined as a graph (XML file), was also implemented. The stream consists of processing elements (nodes) and data connections (edges). Each node is composed of three software components: input strategy, GPU processing kernel and output strategy. The strategies are responsible for formatting the input and output data. They also enable the implementation of two types of patterns: fork-join and scatter-gather. Each kernel is parameterized in the XML file (static and dynamic parameters). The data stream loaded from the XML file is launched and run by the Framework. It receives data from the acquisition cards and then generates results for the visualization application. The Framework supports the operations on many GPU cards and the various methods of data transfer depending on the support offered by the cards.

III. RESULTS

The STRIPES version of the algorithm appeared to be the most efficient. For the non-decimated 128-channel IQ data it is able to produce 512×512 HRIs of 55 mm depth at a rate of 8.5 Hz (fig. 3) which is an equivalent of nearly 1100 LRI/s.

Reduction of the image resolution results in proportional increase of the frame-rate for all algorithms except for the NAIVE. The STRIPES version allows to obtain 256×256 HRI images at a frame-rate equal to 31 Hz which is roughly 4000 LRI/s (fig. 4).

The decimation gives only a slight acceleration since our hardware provides enough bandwidth for the input data. However, the increased number of receiving channels or higher sampling frequency would become a limitation and then the decimation would be more useful.

The framework utilizes less than 10% of processing power, thus its influence on the results is negligible.

The results show that the optimization of the implementation of the reconstruction algorithms can bring a substantial improvement in their efficiency. To compare the obtained frame-rates with those reported in [2] and [3] we need to interpolate them. For image resolution 512×256 used in above papers we expect to achieve the performance of 2000 LRI/s which is far better than reported in [2] (1400 LRI/s). However, the result of 2200 LRI/s declared in [3] proves that our implementation can be still optimized.

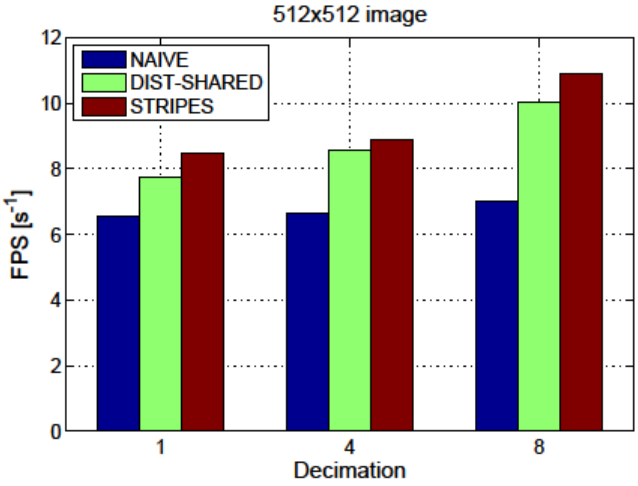

Figure 3. Benchmark of the introduced algorithms for 512×512 pixel images.

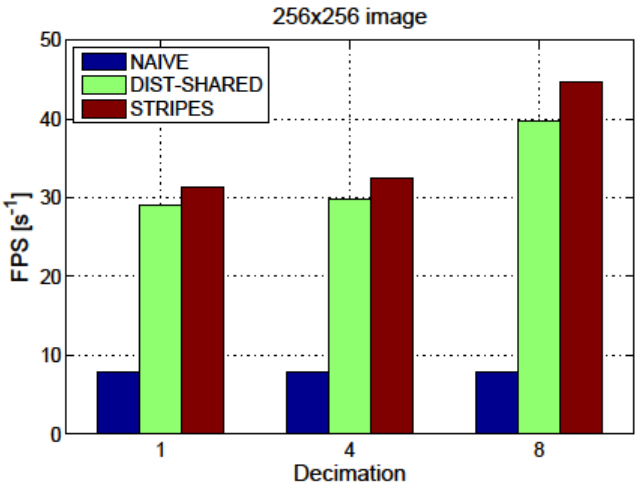

Figure 4. Benchmark of the introduced algorithms for 256×256 pixel images.

I. CONCLUSIONS

The results show that with optimization strategies we greatly increase performance by full utilization of the device. However, there is still much space for improvement in the field of the new generation GPUs.

The implemented synthetic aperture imaging algorithms do not directly benefit from reduced size of the input I/Q data because the number of operations at the image reconstruction depends on the output image size. From the system perspective the decimation of the I/Q data helps to decrease the required CPU→GPU bandwidth.

In future our activity will involve works on subsequent kernels for the ultrasound processing (including Doppler), an adaptation of the Framework for the new version 5 of the CUDA, as well as an integration of the Framework with the hardware layer of the multi-channel acquisition RX64 cards [5] built in our lab.

ACKNOWLEDGMENT

Project POIG.01.03.01-14-012/08-00 co-financed by the European Regional Development Fund under the Innovative Economy Operational Programme.

REFERENCES

[1] Seokhyun Kim; Hak-yeol Sohn; Jin Ho Chang; Tai-kyoung Song; Yangmo Yoo; "A PC-based fully-programmable medical ultrasound imaging system using a graphics processing unit," Ultrasonics Symposium (IUS), 2010 IEEE , pp.314-317, 11-14 Oct. 2010.

[2] Yiu, B.Y.S.; Tsang, I.K.H.; Yu, A.C.H.; "GPU-based beamformer: Fast realization of plane wave compounding and synthetic aperture imaging," Ultrasonics, Ferroelectrics and Frequency Control, IEEE Transactions on , vol.58, no.8, pp.1698-1705, August 2011.

[3] So, H.K.-H.; Junying Chen; Yiu, B.Y.S.; Yu, A.C.H.; "Medical Ultrasound Imaging: To GPU or Not to GPU?," Micro, IEEE , vol.31, no.5, pp.54-65, Sept.-Oct. 2011.

[4] J. A. Jensen, S. I. Nikolov, K. L. Gammelmark, M. H. Pedersen, “Synthetic Aperture Ultrasound Imaging,” *Ultrasonics*, vol. 44, pp. e5-e15, 2006.

[5] M. Lewandowski, M. Walczak, B. Witek, P. Kulesza, K. Sielewicz, “Modular & Scalable Ultrasound Platform with GPU Processing,” Ultrasonics Symposium (IUS), 2012 IEEE, unpublished, 7-10 Oct. 2012.