Application of Non-positional Codes for FIR-filter Implementation Using Computers with CUDA Technology

Dmitry I. Kaplun[1], Alexander S. Voznesenskiy[2], Vyacheslav V. Gulvanskii[3], Danil V. Bogaevskiy
Faculty of Computer Science and Technology
Saint-Petersburg Electrotechnical University "LETI"
Saint Petersburg, Russia
[1]dikaplun@etu.ru, [2]a-voznesensky@yandex.ru, [3]vvgulvanskii@gmail.com

***Abstract* – The paper discusses FIR-filter representation in non-positional codes and their implementation using the CUDA technology. We show the gain in performance for implementing FIR-filters in non-positional codes using the CUDA technology. The paper contains the comparison of FIR-filter implementation for different hardware platforms. We also provide information on further increasing the performance for FIR-filter implementation in non-positional codes using CUDA**

Keywords – non-positional codes, FIR-filter implementation, CUDA technology, GPU, CPU

I. INTRODUCTION

The existing contradictions in the field of digital filtering require us to search for new ways allowing us to provide necessary characteristics of digital filters. One of such ways is the application of non-positional codes for implementing digital filtering algorithms.

If we need to determine the performance characteristics of digital filters we need to specify the requirements. In many works we can see that the existing digital filtering algorithms do not provide the required accuracy and speed for calculating output samples.

In various applications we use non-recursive digital filters since they have several advantages in comparison with recursive digital filters. One of such advantages is linear phase response.

There are many approaches to implementing non-recursive digital filters. Nowadays we most widely use the approaches allowing us to most fully use modern hardware: FPGA, computers with CUDA technology. As we know this hardware makes it possible to implement distributed parallel computing. The algorithms should be parallelized. One of the solutions is the approach based on non-positional codes, in particular, the residue number system [1]. The residue number system the structure of a FIR-filter includes direct transform to non-positional code, modular addition, and modular multiplication for each parallel channel whose number is equal to the number of modules, and the inverse transform (Fig. 1).

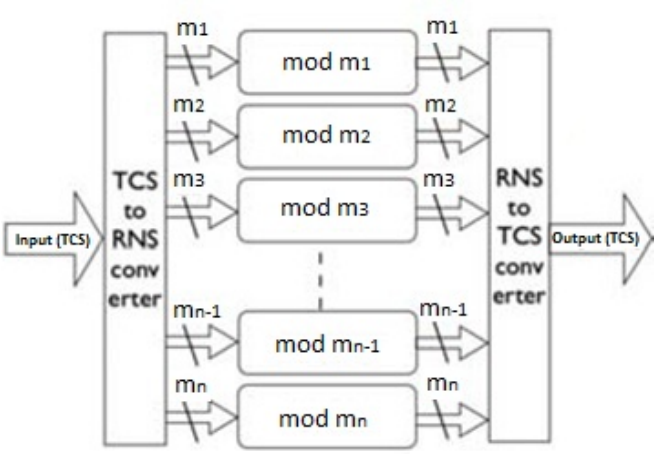

Fig. 1. Filtering process in the residue number system

Thus, addition and multiplication can be performed in a parallel way. Software-hardware implementation allows us to achieve the result that each core of GPU will perform an operation for a particular module. Software-hardware costs can be reduced *n* times, where *n* is the number of modules. It is obviuous to mention that the greatest gain can be achieved for a large filter order and large *n*.

The paper discusses the features of FIR-filter representation using non-positional codes and their implementation using the CUDA technology [2]. The paper also suggests possible ways of improving the performance for FIR filter implementation in non-positional codes using computers with CUDA technology.

II. FEATURES OF CUDA ARCHITECTURE

The computational architecture CUDA is based on the concept: one instruction for a set of data (Single Instruction Multiple Data, SIMD) and the concept of multiprocessor.

SIMD concept suggests that one instruction allows us to perform simultaneous processing of a set of data. .

Multiprocessor is a multicore SIMD processor allowing us in a particular time point to perform instructions for all cores.

978-1-5386-4340-2/18/$31.00 ©2018 IEEE

Each core of a multiprocessor is scalar, i.e. it does not support pure vector operations.

The device is a video adapter supporting the CUDA driver or another specialized device aimed at program execution using CUDA(such as NVIDIA Tesla).

Host is a program in an ordinary operating memory of a computer using CPU and performing control functions for working with such devices.

The part of the program working on CPU is a host and the video card is a device.

Logically the device can be represented as a set of multiprocessors (Fig. 2) and the driver of CUDA.

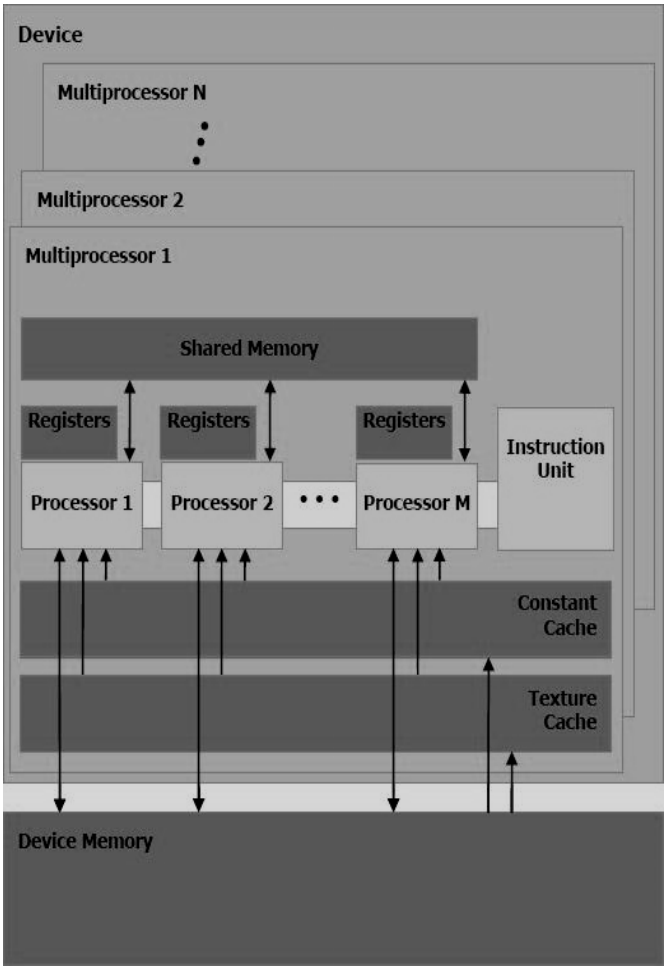

Fig. 2. Device

Imagine we would like to perform on our device a procedure in *N* flows (i.e. we would like to parallelize its performance). We will call this procedure a core.

The feature of the CUDA architecture is a block network organization uncommon for multiflow applications (Fig. 3). The driver of CUDA distributes the resources of the device between flows.

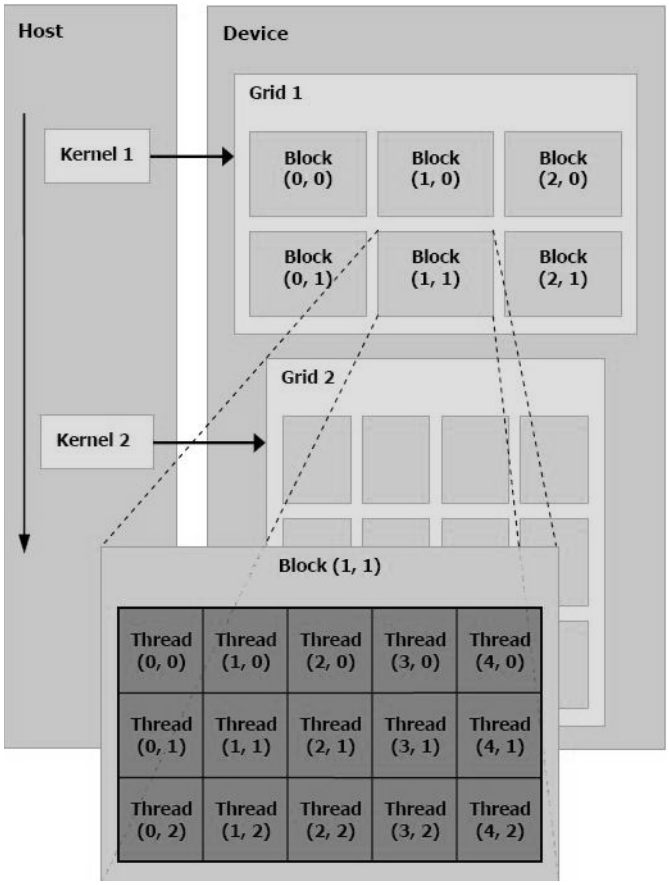

Fig. 3. Organization of flows

Fig. 3 shows the core indicated as kernel. All flows performing this core are joined in blocks (Block) and the blocks are joined in the grid (Grid).

III. Data transfer between GPU and CPU

The main drawback of the NVIDIA CUDA technology is the necessity of data necessity of data transfer between graphical and central processors. Such a drawback is critical in applications working in real time. This is caused by the fact t that a graphical processor has its own address space.

In the case of a small number of operations performed on a graphical processor anв high degree of algorithm parallelization data copying can take nearly 90% of the total execution time.

There are different approaches for reducing data copying time from a host to a device and back:

- use of asynchronous copying , CUDA Streams and CUDA events.
- use of pinned memory for working with data on a host.

In the case of ordinary data transfer between a device and a host we need to allocate memory on a device and copy data from the host to this memory domain.

One of the approaches to reducing the transfer time is number representation in the residue number system, which

reduces the capacity of number representation and the size of transmitted data.

IV. FIR-FILTER IMPLEMENTATION USING THE RESIDUE NUMBER SYSTEM AND CUDA

Based on CUDA specification FIR-filters [3,4] were implemented as a set of functions using the C++ language.

All functions allow us to work with any type of data, when we specify the operations of addition, subtraction, and transformation into int32 format. The implemented functions ions include data copying from a host to a device and back, calculation of inverse inversions for RNS modules, decomposition of coefficients and input samples in RNS, application of the filtering algorithm to input samples in RNS and the conversion of the result to the decimal number system. The most time-consuming function is the calculation of multiplicative inversions due to the necessity of their selection.

In the case of digital filtering in real time in a digital signal processing system it is not necessary to perform constant calculation of multiplicative inversions. In the case of constant modules of RNS it is enough to perform it once, which reduces the algorithm performance time. It is not necessary to transfer filter coefficients in RNS.

V. FIR-FILTER TESTING IN RNS USING CUDA

The digital filtering algorithm in RNS [5] provides processing with high accuracy and can perform in a wide frequency band. The testing was based on using numbers of the Big Integer type.

Big Integer are integer numbers using special algorithms for processing. Their range is not limited.

Using Big Integer in C++ and CUDA was based on the open library Mett Mak Kulchen and adapted for use with GPU with CUDA technology.

Performance analysis was carried out on the video card GeForce GTX 750 Ti and central processor Intel Core i5-4670. Performance analysis was based on using the output of a FIR-filter on CPU, implementation of a FIR-filter on GPU without RNS and implementation of a FIR-filter in RNS.

Testing was based on using numbers of the order 1030. RNS modules encompassed 919, 929, 937, 941, 947, 953, 967, 971, 977, 983. Multiplicative inversions were calculated in advance. Filters with identical coefficients used a random array of samples of the size 8192, 16384, and 32768. The order of FIR-filters was 2048 and 4096.

The results for a FIR-filter of the order 204 are represented in Table 1.

TABLE I. FIR-FILTER OF 204 ORDER

Number of input samples	Execution time		
	CPU, ms	GPU, ms	GPU in RNS, ms
8192	6383	390	201
16384	13722	720	418
32768	28436	1301	765

TABLE II. FIR-FILTER OF 4096 ORDER

Number of input samples	**Execution time**		
	CPU, ms	GPU, ms	GPU in RNS, ms
8192	11152	657	259
16384	26166	1246	508
32768	56301	2633	1034

As can be seen from the tables, implementation of a FIR-filter in RNS has the greatest performance. This is due to the fact that the use of RNS leads to the decrease in the number capacity, but the number of computations increases. For numbers of the type Big Integer arithmetic operations take much more time than for int32 numbers and in this case we can observe the obvious increase of the performance. However, if the gain in the number of operations is insignificant in the case of capacity decrease, it is reasonable not to use RNS. We should remember that in the case of using RNS the number of parallel computing increases several times.

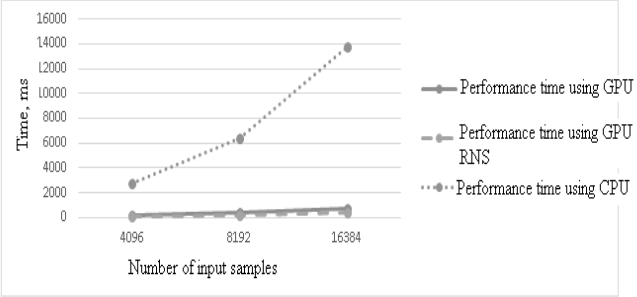

Fig. 4. FIR-filter implementation results

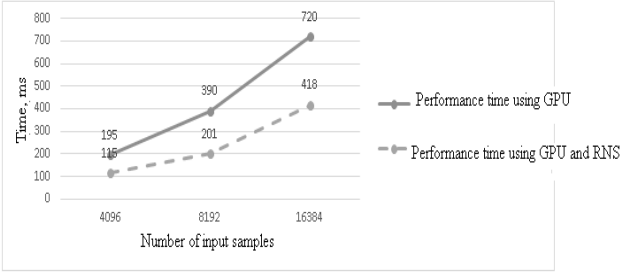

Fig. 5. FIR-filter implementation results in RNS

VI. CONCLUSIONS

Analyzing the figures we can draw the following conclusions:

- implementation performance on GPU is much higher than on CPU;
- application of RNS for reducing the capacity allows us to improve the performance 1.5-2 times;
- by analyzing possible ways of improving the performance we have found that we can improve the

performance if we get rid of multiplication by replacing it with a search table. When we choose small modules of RNS the size of the search table will be small.

ACKNOWLEDGMENT

The research is supported by the grant of the Russian Science Foundation (Project № 17-71-20077).

REFERENCES

[1] A. Omodi, B. Prekumar: Residue Number System – Theory and implementation, Imperial College Press, London, UK, 2007, 293 p.

[2] J. Sanders, E. Kandrot CUDA by Example: An Introduction to General-Purpose GPU Programming // Addison-Wesley Professional, 2010, 312 p.

[3] R. E. Blahut Fast Algorithms for Digital Signal Processing, Addison–Wesley Press, 1985, 455 p.

[4] S.K. Mitra Digital Signal Processing: A Computer-Based Approach /. – USA: McGraw-Hill, 1998, 940 p.

[5] S. Negovan Digital fir filter architecture based on the residue number system, Facta universitatis-series: Electronics and Energetics, vol. 22, No. 1, 2009, p. 125-140.