**Adam DĄBROWSKI, Paweł PAWŁOWSKI, Mateusz STANKIEWICZ, Filip MISIOREK**

Poznan University of Technology, Department of Computing, Division of Signal Processing and Electronic Systems

# Fast and accurate digital signal processing realized with GPGPU technology

***Abstract**. An idea of the so-called quasi-maximum accuracy computations for improvement of precision of the floating-point digital signal processing with graphic processing units (GPUs) is presented in this paper. In the presented approach, the increase of the precision of computations does not need any increase of the length of the data words. Special attention has been paid to efficiency and precision of computations. The maximum accuracy has been analyzed and technically realized with no additional costs in hardware and computation time.*

***Streszczenie.** W artykule zaproponowano wykorzystanie obliczeń zmiennoprzecinkowych o quasi-maksymalnej dokładności do cyfrowego przetwarzania sygnałów za pomocą procesorów graficznych (GPU). W zaprezentowanym rozwiązaniu, zwiększenie precyzji obliczeń nie wymaga zwiększenia długości słów danych. Szczególną uwagę zwrócono na efektywność przeprowadzanych obliczeń. Idea użycia obliczeń o maksymalnej precyzji została technicznie zrealizowana bez dodatkowych kosztów w sprzęcie i w czasie obliczeniowym.*
***(Szybkie i dokładne cyfrowe przetwarzanie sygnałów z wykorzystaniem technologii GPGPU)***

**Słowa kluczowe**: liczby zmiennoprzecinkowe, GPGPU, CUDA, maksymalna precyzja obliczeń, dwa akumulatory, generator cyfrowy
**Keywords**: floating-point numbers, GPGPU, CUDA, maximum accuracy, two accumulators, digital generator

## Introduction

A famous Fettweis-Orchard theorem formulated in the 60-thies of the previous Century explains marvelous properties of passive lossless electronic filters [1, 2]. Furthermore, almost since the beginning of the era of digital signal processing (DSP), it is known that digital lossless systems, i.e. those, which losslessly and passively (or more precisely – structurally losslessly) transmit the signal power (or the "pseudopower" as it was originally referred to) are stable (even under looped conditions), insensitive to coefficient errors, and tolerant to computational inaccuracies [3].

Examples of structurally lossless digital procedures are: discrete Fourier transformation (DFT) together with typical fast Fourier transform (FFT) algorithms, finite impulse response (FIR) filters (based on lossless delay line), and properly designed wave digital filters (WDF's) [3, 4].

It has to be stressed that the kern operation for all mentioned systems is an inner product of vectors. One of these vectors is composed of signal samples and the other one is built with the algorithm coefficients [4, 5].

## Maximum accuracy property

In order to guarantee passivity and approximate losslessness of a technically realized digital system based on the vector inner product, it is reasonable to achieve the so-called "maximum accuracy" of computations. According to the original Kulisch idea, an arithmetic procedure based on a specific number (quantization) grid possesses the maximum accuracy property if in between the exact result of an inner product of two vectors and the computed one, there exists no other number in the grid [6].

In order to guarantee the maximum accuracy property the whole inner product must be computed exactly and only at the end a single rounding of the final result should be made.

A technical realization of the maximum accuracy for fixed-point computations is quite easy. The only prerequisites are: an arithmetic unit adopted to computations with fractions among –1 and 1 and the so-called "long accumulator", i.e., that by some bits longer than the double precision accumulator (long enough to carry a series of exact multiply and add operations).

Realization of the maximum accuracy for floating-point computations is much more involved. The algorithms originally proposed by Kulisch and Bollender never became popular in digital signal processing as the number and thus the time of the required computations depends on the numeric values of the processed data [6, 7]. In consequence one of the most important requirements of the real-time DSP computations, namely the regularity, cannot be guaranteed with these algorithms [8].

This is just one of reasons that over many years the architectures of floating-point digital signal processors (DSP's) were not only not optimized for the maximum accuracy arithmetics but they were quite inefficient with this kind of algorithms. Thus up to now this important concept remained almost forgotten.

Nowadays the situation has drastically changed due to sudden and rapid growth of new technology referred to as the GPGPU computations (general purpose computations with graphic computing units) [9]. In this paper we show that using our approach of two accumulators together with the GPGPU technology, the technical realization of the so-called quasi maximum accuracy floating-point arithmetic is relatively easy and cheap as it can be realized with no extra investments in time of computations and hardware.

## Two accumulators approach

In contrast to multiplication, the floating-point addition is a much more complicated operation than its fixed-point counterpart. It has to be split into four steps:

a) comparison of exponents of the numbers to be added,
b) equalization of exponents of both numbers; an exponent of the smaller number should be increased together with the right shift of the mantissa; it should be noticed that during this shifting the hidden "one" bit (the MSB of the fraction) appears as a physical bit,
c) addition of the mantissas,
d) normalization of the result (if a mantissa of the result remains in range $\langle 1,2)$, it should be unchanged, while if the mantissa is greater than or equal to 2, it should be right shifted by one bit to the normalized value together with the decrease by one of the exponent).

In case of adding a negative number to a positive number or a subtraction, the whole process should be realized as that described above except for replacing the addition of the mantissas with the subtraction of them. Normalization of the mantissa, which is less than 1 consists in left shifting together with the exponent increment.

Sources of errors in the floating-point addition described above are in steps b) and d). An important error is made at the correction step of the fraction of the smaller number after the equalization of the exponents. In extreme case, i.e., if we assume m-bit fraction and the difference between exponents greater than m+1 bits, the smaller number will be zeroed, i.e., the result of addition will stay equal to the greater number and the error of this addition will be equal just to the smaller number. Thus we will lose as many bits of the fraction as they are needed to represent the difference between the exponents of the added numbers.

Mainly because of this, Analog Devices (in the SHARC DSP family) but also other companies extended the single precision IEEE-754 standard and offer fractions longer by 8 bits, i.e., 32-bit long fractions (including the hidden bit) [10].

The second error source, namely the mantissa normalization of the result is less important as it can at most bring a loss of merely one bit.

Rarely overflow and underflow errors can occur, if the results exceed the dynamic range of the arithmetic, but these situations should not appear in the properly functioning algorithms.

In order to reduce errors produced during a long series of additions (accumulations), the authors proposed to use two accumulators [11]. The proposed algorithm of the floating-point addition computes a difference between exponents of the numbers to be added. If this difference is lower than the given threshold Tr, the input number is accumulated to the first accumulator. Otherwise two following steps are made: the value of the first accumulator is added to the number stored in the second accumulator and the input number is moved to the first accumulator. Thus accumulator 1 works periodically with small numbers, while accumulator 2 stores large portions of the intermediate results. At the end, the contents of both accumulators should be added. In our previous works we shown that this algorithm substantially improves accuracy and is tolerant to threshold values.

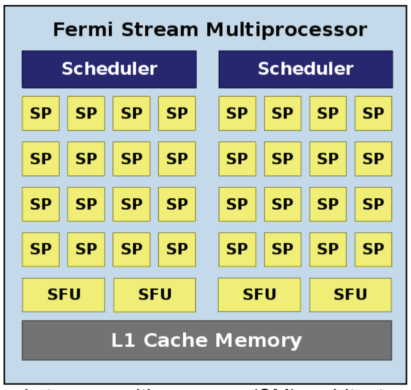


Figure 1. Fermi stream multiprocessor (SM) architecture [15]

## GPGPU technology

General purpose computing on graphics processing units (GPGPU) is a modern and especially nowadays a vital programming technology [12, 13, 14].

Only after the year 2000 new GPU's have been equipped with parallel stream processors also called the programmable shaders. First shaders were split into pixel shaders (PS) and vertex shaders (VS). Afterwards both of them have, however, evolved into the so-called unified shaders, which generally have become as flexible as entire CPU's.

The main advantage of modern GPU's lies in parallelization of their structure. Due to many cores (shaders), many programs can be run at the same time. Table 1 shows the growth of the number of shaders in subsequent NVIDIA GPU models. It can be seen that nowadays the graphics cards possess hundreds of the stream processors.

Table 1. Amount of shaders in consecutive generations of NVIDIA GPU's [15]

| GeForce model | Pixel shaders | Vertex shaders |
|---|---|---|
| 5950 Ultra | 4 | 3 |
| 6800 GS | 12 | 5 |
| 7900 GT | 24 | 8 |
| Quadro FX580 | 32 | |
| 8800 GT | 112 | |
| 9800 GTX | 128 | |
| Quadro 2000 | 192 | |
| GTX 480 | 480 | |
| GTX 590 | 1024 | |

The following GPGPU programming techniques are used: SDK CUDA (software development kit to compute unified device architecture) by NVIDIA, FireStream by AMD/ATI, OpenCL by Khronos Group and DirectCompute by Microsoft [15, 16]. All these API's include software drivers for convenient GPGPU computing.

In our previous work we have presented methods of parallel programming of GPU's and shown that SDK CUDA can be useful to realize complex digital signal processing (DSP) tasks [14].

The newest architecture of GPU from NVIDIA is called Fermi and it introduces innovative changes in construction of the core. Mainly the organization of streaming multiprocessors (SM's) has changed. Now a single core contains 32 stream processors (SP's).

Figure 1 shows the core of the stream multiprocessor. Besides of 32 SP's it contains two schedulers. Each of these subsystems is divided into: instruction cache, register file, two schedulers, and two dispatchers. These components are responsible for managing a group of 32 parallel threats (called warps in the CUDA terminology). Another element is the special function unit (SFU), which enables to execute transcendental instructions such as sine, cosine, reciprocal, and square root [4].

The most important improvements are in the memory. Elder versions of CUDA processors contained 16 KB shared memory per SM. For GPU it was a read only memory used to read constants and textures, which could be set by a host. Now the Fermi stream multiprocessor has got a 64 KB read/write enabled Level 1 (L1) cache memory per each SM. NVIDIA gives an opportunity to split this memory to 16 KB shared memory plus 48 KB L1 cache memory or 48 KB shared memory plus 16 KB L1 cache memory. It is a much more flexible solution.

Fermi GPU contains 768 KB L2 cache memory with load and store operations enabled. The third level is DRAM, installed on a graphics card, usually it is a fast DDR memory. Fermi has got 384-bit wide memory interface. It enables to support maximum 6 GB DDR5 DRAM. The chip additionally supports error correcting codes (ECC's). By this means the developer can be much more confident that the data are correct despite hardware defects.

NVIDIA classifies its products into groups and gives them the so called compute capability number. Currently there are two main compute capabilities 1.x and 2.x [15].

All mentioned hardware improvements allowed to add new features in the SDK like warp vote functions, double-precision floating-point arithmetic and floating-point atomic addition operating on 32-bit words.

Finally CUDA gives the user better support for scientific computing and DSP. There exist MAD (multiply and add) and FMA (fused multiply and add) operations that support inner product computations. Both are performed in a single clock period. However MAD offers a single precision, i.e., rounded result while FMA guarantees the exact (double precision multiplication result).

FMA instruction together with the two accumulator concept have been used to realize the quasi maximum accuracy arithmetic. In order to present advantages of this approach results of experiments with two FIR filters are shown in next Section.

Subnormal floating-point numbers, which are also supported, fill the gap around zero in the floating-point number representations of former DSP's.

Another advantage of new SDK is that it supports high-level languages like C++. However actually it supports C++ with important restrictions, e.g., classes are not able to use virtual functions.

Fermi architecture enables also pointers, try-catch blocks, and better support for the debuggers.

To be objective one has to add that there are also some disadvantages, mainly because of limitations of the CUDA architecture. First, the user must remember that the SIMT (single instruction multi thread) model, represented by CUDA architecture, has limitations in relation to the single instruction multiple data (SIMD) model. CUDA puts 32 threads into one warp and performs one instruction. This looks similar to the SIMD model, however, in this case the thread cannot use data contained in the registers that are owned by another thread. Fortunately this is not a problem in the proposed two accumulator method of the realization of the maximum accuracy arithmetic.

Some consolation is that the software can communicate between threads by the shared memory only.

Another drawback comes from the fact that devices still need to switch between different contexts (e.g., graphics context and computing). Thus it is not possible to run them simultaneously.

## Results of experiments

The proposed approach to the quasi maximum accuracy in floating-point digital signal processing has been verified with realization of sine waveform generators. Many such generators realized in parallel, with parameters controlled in real-time, with frequency modulation feature, are very useful in medical applications (e.g. audio-, photo- and magneto-stimulation). The real-time control is crucial for fast and exact feedback during a medical procedure [18].

Here we only write the final expressions for computing of $n$-th sample of the sine signal with frequency modulated by the triangle signal [11]:

$$x_n = A \cdot \sin(\varphi_n) \tag{1}$$

where:

$A$ - amplitude of the sine signal,

$\varphi_n = \varphi_{n-1} + W_g \cdot (M_n + 1)$, $\quad M_n = M_{n-1} + W_m$

$W_g = 2\pi f_g / f_s$, $\quad W_m = \pm 4 \cdot g \cdot f_m / f_s$,

$f_g$ - frequency of the sine signal (without modulation),

$f_s$ – sampling frequency, $f_m$ – modulation frequency, frequency deviation:

$g = \Delta f_g / f_g$, $\quad \Delta f_g = f_{\max} - f_g = f_g - f_{\min}$

For approximation of sine function we used Taylor series of 9-th order [18].

It can be noticed that we have two accumulation procedures: increment of phase φn (consecutive samples) and frequency modulation (accumulator $M_n$).

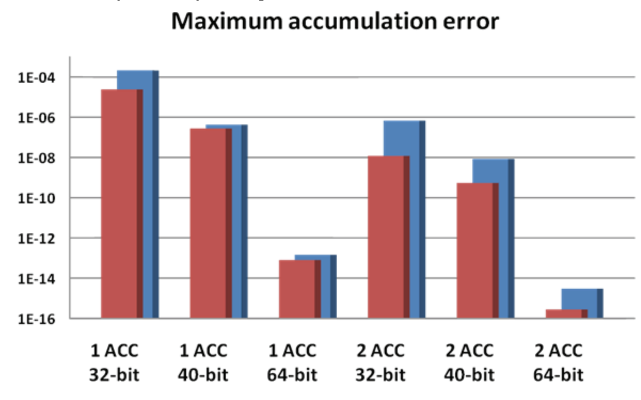
If the modulation signal is common for all generators of particular sine components of the generated signal (what is used in the most cases), parameter Mn can be calculated only once and used for all generators, because it does not depend on parameters of the modulated signal.

We assumed typical parameters of generators used for medical stimulation in audio range and realized with GPGPU technique and CUDA technology in C language. Very slow frequency modulation (e.g. with frequency equal to $f_m$ = 0.5 Hz, deviation $g$ = 0.3 and $f_s$ = 48 kS/s) produces modulation parameter that needs to be accumulated equal to $W_m = 1.25 \cdot 10^{-5}$. This value cannot be precisely accumulated using a 32-bit floating-point representation (so-called single precision). Unfortunately some GPUs with CUDA compute capability version 1.1 and lower do not support efficient double precision floating-point computations (with 64-bits). To overcome this drawback we propose to use the two-accumulators approach (see chapter 3).

The maximum accumulation error during accumulation of modulation parameter Wm was calculated in MATLAB with two options: with and without rounding. Numbers representation and rounding to nearest method was used according IEEE 754 standard [19]. Results are presented in Fig. 2. The outcomes show that two-accumulators approach reduces accumulation error more than 100 times for the same length of a word. The rounding also significantly reduces errors, especially in two-accumulator method.

For practical reasons, and to proof that the two-accumulator method can be useful, especially in multi-core processing environment (e.g. GPGPU), the experiment of up to 1000 modulated sine generators was performed.

For the tests we have used Dell Precision T1500R PC computer with CPU Intel i7 870 2.93 GHz, RAM 4 GB DDR3, two GPU cards: NVIDIA Quadro FX580 (32 processor cores 800MHz, RAM 512 MB GDDR3, 450MHz) and NVIDIA Quadro 2000 (192 processor cores 1250MHz, RAM 1GB DDR5 1300MHz), HDD 500GB Sata II, operating system Debian GNU/Linux (kernel 2.6.37 i686). The graphic processor Quadro FX580 is compatible with CUDA compute capability version 1.1 and Quadro 2000 with CUDA compute capability version 2.0.



| | 1 ACC 32-bit | 1 ACC 40-bit | 1 ACC 64-bit | 2 ACC 32-bit | 2 ACC 40-bit | 2 ACC 64-bit |
|---|---|---|---|---|---|---|
| 🟥 with rounding | 2,39E-05 | 2,73E-07 | 8,00E-14 | 1,19E-08 | 5,36E-10 | 2,78E-16 |
| 🟦 without rounding | 2,13E-04 | 4,31E-07 | 1,43E-13 | 6,74E-07 | 8,57E-09 | 3,00E-15 |

Fig. 2. Maximum accumulation error for 24000 accumulations of $W_m = 1.25 \cdot 10^{-5}$

Seven versions of generators have been tested. They are listed in Table 2. Performance results are presented in Figure 3. The program calculates one vector of output samples divided between particular generators. The solution is universal: the output signal can be a sum of all or selected generators or as many output signals as generators can be produced. Additionally, it is not important which thread is related to a given generator. From Fig. 3 it can be deduced the two-accumulators approach computed by CPU is significantly slower than typical solution, even if we compare 32-bit to 64-bit precision. On the other side, the GPU calculates both approaches in comparable time and does it much faster than CPU, especially for large number of generators. Notice that the used Quadro FX580 has only 32 processor cores (most advanced GTX590 form NVIDIA houses dual 512-core graphics processing unit [15] – compare Tab. 1) and is clocked 3.6 times slower than CPU whereas Quadro 2000 GPU has 192 processor cores and is clocked 2.9 times slower than CPU.

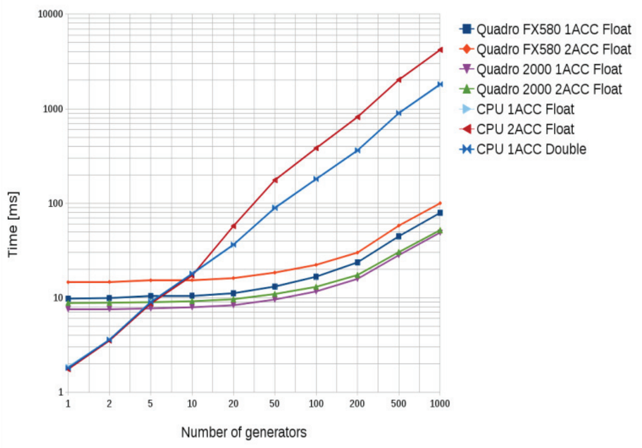


Fig. 3. Comparison of computing time: CPU vs. GPU
It should be noted that the two accumulator approach significantly outperforms other approaches.

Table 2. Versions of tested generation algorithms

| Version indication | Processor | Computing precision | No. of accumulators |
|---|---|---|---|
| Quadro FX580 1ACC 32-bit | GPU | Single (32-bits) | 1 |
| Quadro FX580 2ACC 32-bit | GPU | Single (32-bits) | 2 |
| Quadro 2000 1ACC 32-bit | GPU | Single (32-bits) | 1 |
| Quadro 2000 2ACC 32-bit | GPU | Single (32-bits) | 2 |
| CPU 1ACC 32-bit | CPU | Single (32-bits) | 1 |
| CPU 2ACC 32-bit | CPU | Single (32-bits) | 2 |
| CPU 1ACC 64-bit | CPU | Double (64-bits) | 1 |

## Conclusions

In this paper we have shown that the general purpose computing using graphics units (GPGPU) can be successfully applied to perform floating-point computations with practically maximum accuracy with no additional costs in hardware and time of computations. This has been achieved by means of the described concept of two accumulators.

The regularity of computations, necessary to realize various tasks of digital signal processing, is guaranteed. We have proven this by means of experiments with precise frequency modulated sine generators.

Realization of FM sine generators with quasi maximum accuracy arithmetic improves accuracy of the frequency deviation.

Using the two-accumulator approach with the GPU is much faster than with the CPU and does not prolong computing time significantly.

An additional improvement in the GPGPU solution can be achieved by using the so called streams. Thanks to this, several programs can be run in parallel and asynchronously. It means that each generator would produce each own output vector as fast as it is possible.

More advanced algorithms can be executed in real-time thanks to the increased computing capacities of modern GPU's using parallelism.

*The paper was supported by INDECT and DS 2012 projects.*

## REFERENCES

[1] Fettweis, A., Filters met willekeurig gekozen deningspolen en Tschebyschew-karakteristik in het doorlaatgebied, Tijdschrift van het Nederlands Radiogenootschap, Vol. 25, No. 2 1960, pp. 337 – 382.
[2] Orchard, H. J., Inductorless filters, Electron. Lett., Vol. 2, No. 6, 1966, pp. 224 – 225.
[3] Fettweis, A., Digital filter structures related to classical filter networks, Arch. Elek. Uebertrag., Vol. 26, No. 5, 1972, pp. 201 – 206.
[4] Dabrowski, A., Recovery of effective pseudopower in multirate signal processing, PUT, Poznan 1988.
[5] Dabrowski, A., Multirate and multiphase switched-capacitor circuits, Chapman & Hall, London 1997.
[6] Kulisch, U. W. and Miranker, W., Computer Arithmetic in Theory and Practice, Academic Press, 1984.
[7] Bohlender, G., Floating-point computation with maximum accuracy, IEEE Trans. Computers, Vol. C-26, No. 7, 1977, pp. 621 – 632.
[8] Dabrowski, A. and Olejniczak M., Rounding noise in digital floating-point FIR filters, Proc. Latvian Signal Proc. Int. Conf. (EURASIP), 1990, pp. 223 – 229.
[9] Rosenband, D. L. and Rosenband, T., A design case study: CPU vs. GPGPU vs. FPGA, Formal Methods and Models for Co-Design, 2009. MEMOCODE '09. 7th IEEE/ACM Int. Conference on Digital Object Identifier, 2009 , pp. 69 – 72.
[10] Analog Devices, “ADSP-2106x SHARC DSP Microcomputer Family, ADSP-21061/ADSP-21061L,” Analog Devices, Inc., Rev B, 2000.
[11] Pawłowski, P. and Dabrowski, A., Extended precision method for accumulation of floating-point numbers in digital signal processors, UTP, Elektronika 10, Vol. 249, 2007, pp. 77 – 84.
[12] Harris, M., Coombe, G., Scheuermann, T., Lastra, A., Physically-based visual simulation on graphics hardware, 2002 SIGGRAPH / Eurographics Workshop on Graphics Hardware.
[13] Rumpf, M. and Strzodka, R., Using graphics cards for quantized FEM computations. Proc. VIIP’01, 2001, 193 – 202.
[14] Misiorek, F., Dabrowski, A, Pawłowski, P., The ability to use parallel programming on graphics processor unit in digital signal processing, Proc. of New Trends in Audio and Video / IEEE Signal Processing Conference NTAV/SPA 2008, Poznań 2008, pp. 225 – 228.
[15] NVIDIA webpage, www.nvidia.com
[16] Han,. T. D., Abdelrahman, T. S., hiCUDA: High-Level GPGPU Programming, IEEE Transactions on Parallel and Distributed Systems, Volume: 22 , Issue: 1, 2011, pp. 78 – 90.
[17] MathWorks, MATLAB - The Language Of Technical Computing, Parallel Computing Toolbox, www.mathworks.com, 2011
[18] Portalski M., Pawłowski P., Dąbrowski A., Synthesis of some class of nonharmonic tones, XXVIII IC-SPETO International Conference on Fundamentals of Electrotechnics and Circuit Theory 2005, vol. 2. pp. 439 – 442.
[19] IEEE Standard Board, IEEE Standard for Floating-Point Arithmetic, IEEE Std 754–2008

***Authors**: Adam Dąbrowski (adam.dabrowski@put.poznan.pl), Paweł Pawłowski (pawel.pawlowski@put.poznan.pl), Mateusz Stankiewicz (mateusz.stankiewicz@put.poznan.pl), Filip Misiorek (filip.misiorek@gmail.com), Poznan University of Technology, Department of Computing, Division of Signal Processing and Electronic Systems, Piotrowo 3a, 60-965 Poznań*