

Parallel Execution of FFT Algorithms on NeuroMatrix® Architecture

Vitaly Kashkarov, Sergey Mushkaev
RC MODULE, Moscow

This article studies the possibility of parallel computing applied to FFT. It examines an approach to FFT radix 16 implementation and makes a comparative analysis of the discussing approach with a standard method of FFT computation on radix 2. In addition, the article describes the basic principles of 256 point complex FFT implementation on NeuroMatrix® architecture. It estimates accuracy of calculation and NM6403 performance.

Introduction

A significant part of the tasks of time series analysis is connected with Fourier transform and methods of its effective computing. In these tasks Fourier transform plays an important role as a necessary intermediate step in defining power spectrum densities, cross-spectral densities, transfer functions, convolutions, correlation functions and so on.

In practice the most widely spread FFT algorithms are those on radix 2, where each functional node performs a basic operation – a two-point “butterfly”. These algorithms were primarily oriented on reducing the number of multiplication operations to the minimum. Nevertheless, with vector processors appearing this criterion becomes inessential. On the contrary, the number of simultaneously made multiplications is the main indicator of the processor performance. Therefore the question of parallel FFT computation with higher radices and their possible combinations arises.

The steps of FFT computing can be described in the form of a graph, the nodes of which perform a discrete transform but on a less dimension vectors that are the parts of the input vector. Depending on the radix choice, both the total number of arithmetic operations and the number of graph layers change (Fig. 1).

Table 1. FFT Computing Complexity

N	Forward computation of DFT (radix N)			Computation of FFT on radix 2			Computation of FFT with combined radices 2,16,32			
	Complex muls	Complex adds	Number of graph layers	Complex muls	Complex Adds	Number of graph layers	Complex muls	Complex adds	Combinati on of radices	Number of graph layers
N	N ²	N ² -N		(N/2)log ₂ N	Nlog ₂ N					
256	65536	65280	1	1024	2048	8	8192	7680	16-16	2
512	262144	261632	1	2304	4608	9	16384	16384	2-16-16	3
1024	1048576	1047552	1	5120	10240	10	49152	49152	32-32	2
2048	4194304	4194304	1	11264	22528	11	131072	131072	2-32-32	3

Complex muls – Number of complex multiplications

Complex adds – Number of complex additions

In FFT radix 2 algorithms the number of processing layers achieves its maximum (Table 1). Increasing the number of layers causes increasing of round off error as floating point arithmetic requires normalization at each layer. It is especially crucial in those cases when computations are given in fixed point arithmetic or with an insufficient data word-length. It should be also noted that in this case in order to prevent overflow it is necessary to normalize the intermediate results after each or after a group of graph layers applying the operation of right shift (Fig.1). Apart from the shift, the normalization may include the rounding procedure which also contributes to additional computational burden.

A possible trade-off decision might be the approach based on increase of radix in FFT algorithms. Below we focus at a 256 point FFT radix 16. On one hand the choice of this radix provides the possibility of organizing parallel computations, and on the other hand it reduces the number of graph layers to two.

The discrete 256-point Fourier transform is determined by the vector matrix formula:

$$Y(k) = \sum_{n=0}^{255} W_{256}^{k \cdot n} \cdot X(n), k = 0..255, \text{ where } W_{256}^k = \exp\left(-\frac{2\pi \cdot i \cdot k}{256}\right)$$

After identical transformation this formula takes on the form more suitable for calculation the FFT-256 radix 16:

$$Y(k) = \sum_{n=0}^{15} W_{256}^{k \cdot n} \cdot \sum_{i=0}^{15} W_{256}^{16k \cdot i} \cdot X(16 \cdot i + n); k = 0..255$$

The 256 point complex FFT radix 16 graph (see Fig.1) gives an idea on how this formula works:

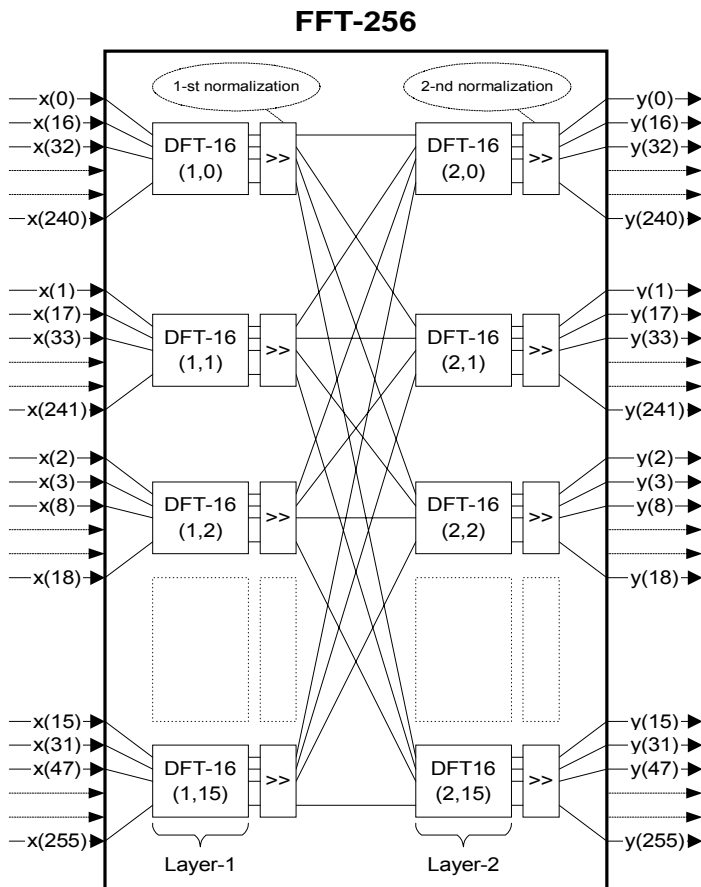


Fig.1 Generalized graph of computing the FFT-256 on radix 16.

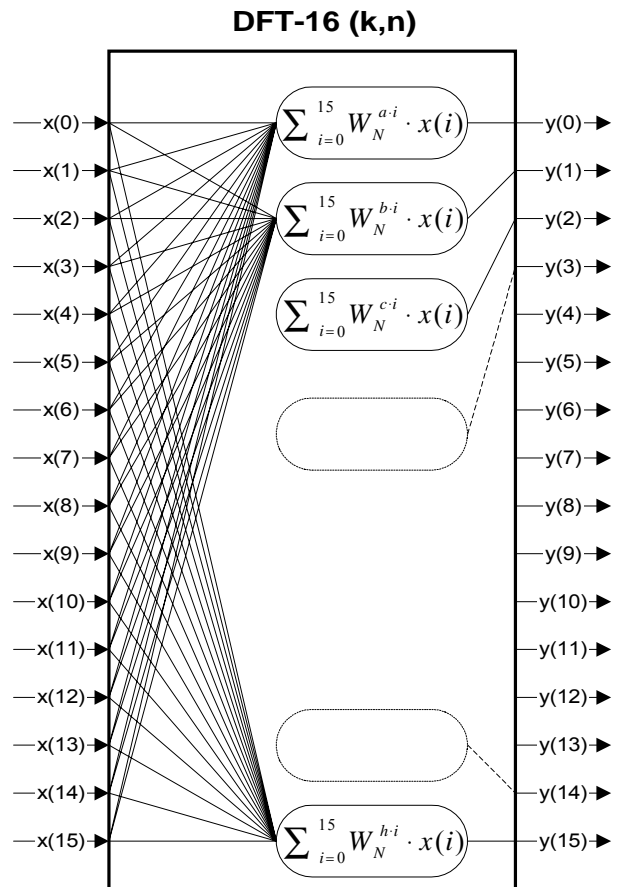


Fig.2 Expanded pattern of a block of 16-point discrete Fourier transform

The graph consists of two layers with sixteen blocks in each layer. Each block in the graph has sixteen complex inputs and outputs. As shown in Fig. 2, each block in the graph performs a 16-point discrete Fourier transform and differs from other blocks only in complex coefficients W . Thus, parallel FFT processing actually comes to effective implementation of DFT-16, i.e. determination of sixteen scalar products of different vectors $[W]$ with one vector $[x]$, which is equivalent to multiplication of the matrix of coefficients of Fourier transform $-[W]$ of the size 16×16 elements by 16 elements input vector $[x]$.

DFT-16 Implementation on NeuroMatrix® Architecture

Effective parallel processing of DFT-16 is achieved by means of hardware support. NeuroMatrix® architecture provides fast vector-matrix multiplication. All arithmetic computations referring directly to DFT-16, are made on the vector co-processor (VCP). As VCP allows operating with data of variable word-length, we allocate 32 bits for the Imaginary (Im) part and 32 bits for the Real (Re) part for input data and computation results, and 8 bits Re and 8 bits Im to store complex coefficients W . Thus, one complex value is contained in one 64bit word. Imaginary and Real parts of the W coefficients are also packed to 64bit words. All W coefficients are calculated in advance and therefore they are stored in an array in the order convenient for further calculations (Fig.3).

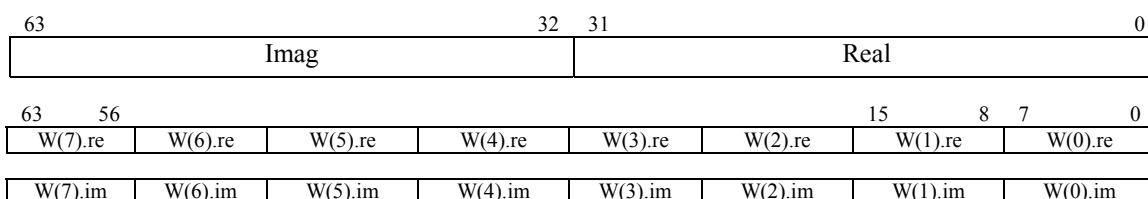


Fig.3 Format of Storage of Input Data and Transform Coefficients

Due to such data representation the VCP multiply unit operates in two configurations:

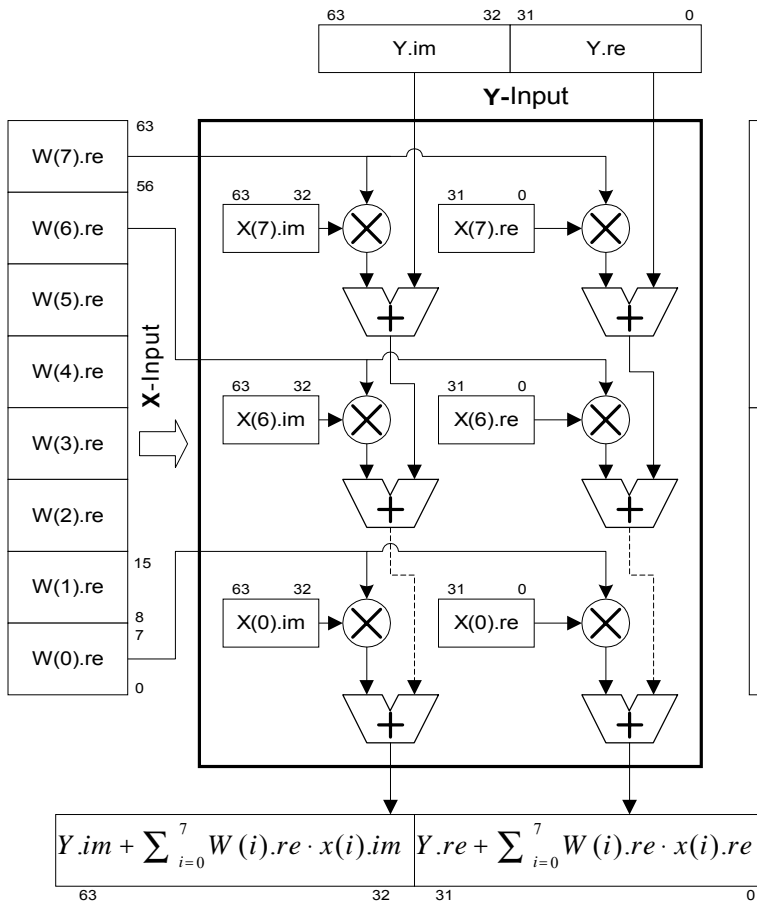


Fig.4 Equivalent diagram of NeuroMatrix VCP multiply unit configured to eight 8bit inputs and two 32bit outputs.

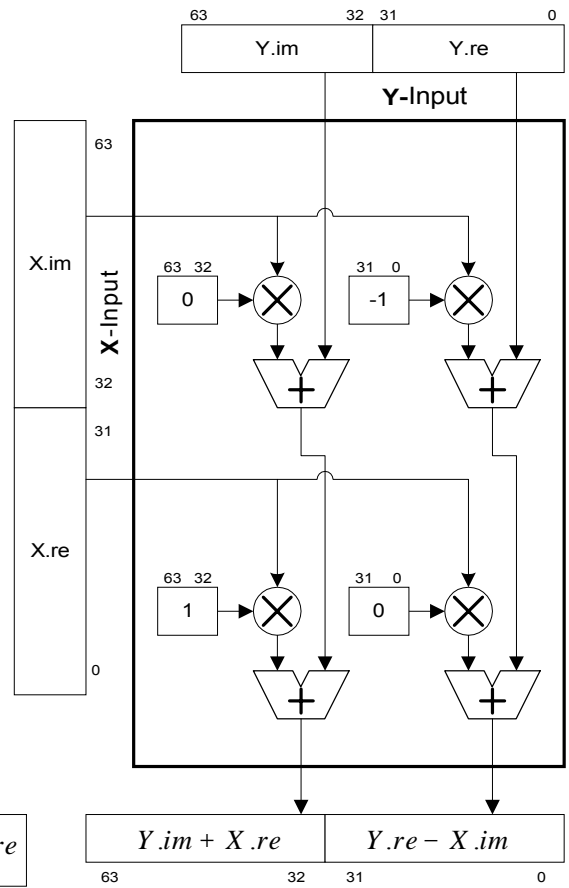


Fig.5 Equivalent diagram of NeuroMatrix VCP multiply unit configured to two 32bit inputs and two 32bit outputs.

The two given variants of VCP multiply unit configuration describe the full process of scalar multiplication of two complex vectors. The first diagram performs sixteen multiplications and accumulations per one processor cycle and serves for calculation of sums of pairwise products of Im and Re parts, the second diagram performs four multiplications and accumulations per cycle, but in fact it serves only for final addition of the obtained partial sums. The full diagram of multiplication of two complex sixteen element vectors is shown in Fig.6. Since only eight elements of the input vector $[x]$ can be loaded to the multiply unit at once, loading of the whole vector $[x]$ takes place in two stages.

The entire scalar product $y(k) = \sum_{i=0}^{15} W_{256}^{a \cdot i} \cdot X(i)$ consists of three stages:

1. Eight complex elements $x(0)..x(7)$ are loaded to the VCP multiply unit. The vector of eight *Re* parts of complex coefficients $W(0)..W(7)$ comes from memory to the X input of the multiply unit followed by the vector of eight *Im* parts. The multiplication is made according to diagram shown in Fig.4. The results of the multiply-add operation is stored to the accumulation FIFO (AFIFO) for further processing. While the first eight elements of input vector take part in calculations the rest eight elements are loaded to the Shadow Matrix of the VCP multiply unit. After the first stage of calculations is finished, the processor copies the contents of the Shadow Matrix to the Active Matrix of the VCP multiply unit.
2. At the second stage the rest eight elements $x(8)..x(15)$ of the sixteen element input vector are multiplied by the FFT weight coefficients $W(8)..W(15)$ and the result this product adds to the result of the first stage. So, after two stages we have two 64bit words containing packet results of calculation as shown below:

$A = \sum_{i=0}^{15} W_{256}^{a \cdot i} \cdot re \cdot X(i).im$	$B = \sum_{i=0}^{15} W_{256}^{a \cdot i} \cdot re \cdot X(i).re$
$C = \sum_{i=0}^{15} W_{256}^{a \cdot i} \cdot im \cdot X(i).im$	$D = \sum_{i=0}^{15} W_{256}^{a \cdot i} \cdot im \cdot X(i).re$

3. To obtain the ultimate result $y(k)$, we have to perform $A + D$ and $B - C$ operations. To carry out this operation in VCP we load 0,1 and -1 to the Active Matrix of the VCP multiply unit matrix. These values are used as weight coefficients allowing us to execute ADD and SUBTRACT operations on X and Y operands in parallel. The result of the third stage is $y(k)$.

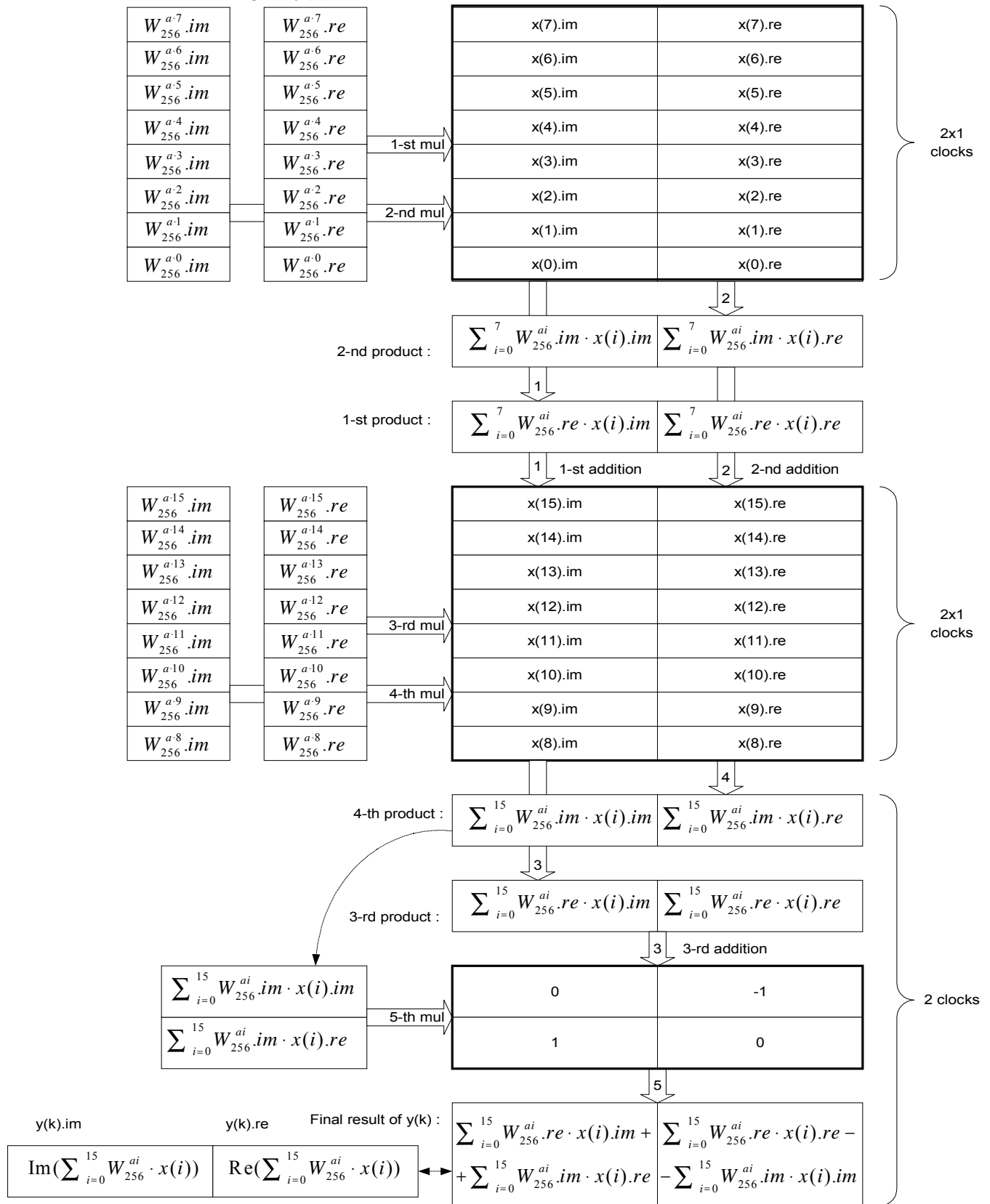


Fig.6 The order of DFT-16 $y(k) = \sum_{i=0}^{15} W_{256}^{a-i} \cdot X(i)$ calculation.

Fig.6 illustrates the process of scalar multiplication of two vectors only. Actually, to achieve the maximum performance of the NeuroMatrix VCP the data are loaded in blocks of thirty-two 64-bit words. In the result, taking into account the data transfer time, every step of multiplication (Fig.4, Fig 5) takes one clock cycle. It becomes possible due to simultaneous use of two data buses. Loading of input data vectors $x(i)$ via the first bus is combined

with loading of DFT coefficients $W(i)$ and storing of the results $y(i)$ via the second bus. Thus, the entire procedure of scalar multiplication of two complex 16-element vectors actually takes seven processor cycles in average over the whole FFT-256.

Performance and Accuracy of Computation

The number of bits allocated for presentation of the FFT coefficients $W(i)$ defines the accuracy of computation. There are two ways of presenting values of cosine and sine in an 8-bit grid:

1. $W = \text{round}(64.0 * \cos(x))$ - (six significant bits)
2. $W = \text{round}(127.0 * \cos(x))$ - (seven significant bits)

In the first case, there are 65 different values of cosine in the range from 0 to 1, while in the second case 128 different values are available. Of course, we achieve better accuracy in the second case. Below in the chapter we will evaluate the difference in accuracy.

However, if only operations of integer multiplications and accumulations are made during the calculations, it is necessary to normalize the results of those calculations, i.e. every element of the output array should be divided by the corresponding scaling coefficient. For the first case, it is equal to 64^2 , and for the second to 127^2 . In both cases, it is expedient to replace the division with a right shift, but unlike the 6-bit case the replacement of the division by 127^2 by the shift 14 bits right causes a small systematic error. Normalization can be made either once in the end of multiplications, or twice – after each step of multiplication (Fig. 1) (intermediate normalization serves to avoid overflow during computations). The number of necessary stages of normalization is determined depending on the range of input data (see Table 3).

For accuracy evaluation the arbitrary signal was processed using the forward and inverse Fourier transform and then the input x was compared with the restored signal x' . In particular, the average of distribution M and the standard deviation σ of relative error δ were calculated for accuracy estimation. $\delta(i) = \frac{x'(i).re - x(i).re}{x(i).re}$, $i=0..255$.

Table 2. Comparative characteristics of the restored signal accuracy after forward and inverse FFT with different radices

Fourier transform	Systematic error-M		SD - σ	
	6 bit/1.0	7 bit/1.0	6 bit/1.0	7 bit/1.0
FFT-256 radix 2	-1%	-12%	2.0%	0.9%
FFT-256 radix 16	-0.4%	-3%	1.2%	0.6%

Sign “-” means that the restored signal weakens by M percent if compared to the input signal.

As it shown in the table above, the 7-bit representation of FFT coefficients for FFT-256 radix 2 causes an essential systematic error and makes such an algorithm less effective. The FFT-256 radices 16 and 32 allow more effective use of the bit grid for the DFT coefficients. Those algorithms have higher accuracy due to 4-5 times fewer number of graph layers, which as well decreases additional expenses for normalization and adjustment. The adjustment operation on the NeuroMatrix VCP is made by means of the vr vector register [1] and combined with main computations without causing any additional time expenses. There is also a possibility of making additional optimization combining the normalization procedure (arithmetic right shift) with the last stage of the scalar matrix vector product computation (Fig. 6).

General Characteristics of FFT Functions

The FFT algorithm implementation on NeuroMatrix architecture has the following features:

Input and output data are 32-bit integer complex values; the storage format is shown in Fig.3

Input data range is shown in Table 3.

Depth of FFT coefficients is 8 bit (two variants of representation: 6 and 7 significant bits)

Fixed point arithmetic.

Output data elements are stored in the same order as the input ones.

Table 3. Performance of Forward and Inverse FFT on NeuroMatrix NM6403 Processor

Number of Complex Points	Without normalization			One normalization			Two normalizations		
	Cycles	Time msec	Input data range	Cycles	Time, msec	Input data range	Cycles	Time, msec	Input data range
256	3662	0.092	± 512 (7bit) ± 2047 (6bit)	4053	0.1	± 512 (7bit) ± 2048 (6bit)	4429	0.11	$\pm 2^{18}$ (7bit) $\pm 2^{18}$ (6bit)
512	8180	0.2	± 256 (7bit) ± 1023 (6bit)	8930	0.22	± 256 (7bit) ± 1023 (6bit)	9524	0.24	$\pm 2^{18}$ (7bit) $\pm 2^{18}$ (6bit)
1024	18900	0.47	± 128 (7bit) ± 511 (6bit)	20234	0.5	± 128 (7bit) ± 511 (6bit)	22630	0.56	$\pm 2^{17}$ (7bit) $\pm 2^{17}$ (6bit)
2048	47624	1.2	± 64 (7bit) ± 255 (6bit)	50289	1.25	± 64 (7bit) ± 255 (6bit)	52665	1.32	$\pm 2^{17}$ (7bit) $\pm 2^{17}$ (6bit)

NM6403 cycle time=25 ns (40MHz)

References: [1] – NeuroMatrix architecture documentation is available at <http://www.module.ru/products/nm/nm6403.html>