UDC: 519.6 https://doi.org/10.15407/jai2022.02.061

AVOIDING THE BIT-REVERSED ORDERING IN PARALLEL MULTI-DIGIT MULTIPLICATION BASED ON FFT

A. M. Tereshchenko[1], V. K. Zadiraka[2]
[1,2]Institute of Cybernetics of NAS of Ukraine, Ukraine
40, Academician Hlushkov av., Kyiv, 03680
teramidi@ukr.net
zvk140@ukr.net
[1]https://orcid.org/0000-0002-9549-9275
[2]https://orcid.org/0000-0001-9628-0454

Annotation. The paper for the parallel model of computation, a modification of the method of implementing the multiplication of multi-digit integers based on the fast Fourier transform (FFT) avoiding the bit-reversed ordering is proposed. The paper researches the calculation of FFT according to the "butterfly" scheme based on decimation-in-frequency and decimation-in-time methods, an input signal with elements in direct and bit-reversed order, with an increase and decrease in the number Fourier series coefficients at each step of the "butterfly", the use of a list of Fourier series coefficients in direct and bit-reversed order. The standard FFT-based multiplication algorithm uses the same "butterfly" operation to compute the forward and inverse Fourier transforms. The paper analyzes two combinations of the $FFTFD_N$–$FFTTB_N$ and $FFTFB_N$–$FFTTD_N$ "butterfly" calculation schemes for calculating forward and inverse discrete Fourier transforms (DFT) in the case of implementing the multi-digit operation in parallel computational model to exclude bit-reversed permutation. A scheme for distributing calculations among four processors is proposed, in which forward and inverse Fourier transform calculations are localized within one parallel processor. The proposed modification does not reduce the computational complexity in terms of the number of complex operations, but due to the exclusion of bit-reversed permutation, the number of synchronization commands between processors and data is reduced, which reduces the algorithm execution time. The scheme can be adapted to distribute the computations among a larger number of processors. Four algorithms for implementing FFT based on decimation-in-frequency and decimation-in-time methods, an input vector with elements in direct and bit-reversed orders are presented. To check the result of the calculation, the algorithm of multiplication avoiding the steps of bit-reversed ordering was implemented in the APL programming language. An example of calculation is given in the form of a table.

Keywords: parallel model of computation, fast Fourier transform, multi-digit multiplication, bit-reversed ordering, decimation-in-frequency, decimation-in-time.

Introduction

In today's world, digital signal processing (DSP) is an essential part of technologies used from the research of the earth's surface from space to real-time facial recognition using artificial intelligence techniques. One of the DSP methods is the Discrete Fourier Transform (DFT) calculation method and Fast Fourier Transform (FFT) algorithms. FFT is widely used to implement operations of multi-digit arithmetic, the speed of which have an influence on the speed of cryptographic hardware and software. The accuracy of the result depends on the number of digits of a multi-digit operation. The accumulation of rounding error in the case of performing tens of millions of operations can give a result that has no physical meaning.

It is believed that the fast algorithm for calculating the DFT has been discovered several times. The development of a fast algorithm for calculating the discrete transformation can be traced to the Carl Guess's unpublished work in 1805 when he needed it to interpolate the orbit of asteroids Pallas and Juno from sample observations. This work predated Furie's work [1], [2], but Carl Guess did not analyze the computation time and eventually used other methods to achieve his goal.

J.W. Cooley and John Tukey suggested the method [3], that recursively breaks down a DFT of any composite size $N=N_1 \cdot N_2$. When one of the numbers is a power of two, it is called decimation-in-time or decimation-in-frequency method, depending on how the "butterfly" operation is calculated. This method of breaking into two smaller DFTs is sometimes called Danielson–Lanczos lemma (1942) [4], influenced by Runge's work (1903) [2]. Although they apparently didn't realize the linearithmic asymptotic

complexity they had achieved.

The FFT calculation method consists of two parts: fast DFT calculation and bit-reversed ordering. After the appearance of the FFT, two of its parts were researched in detail for the purpose of acceleration.

For the sequential computational model, many original FFT acceleration methods are used, such as the method with precomputation of the Fourier series coefficients and the method of calculating the FFT using $N/4$ Fourier series coefficients instead of N^2 [5].

In this work, in the case of multi-digit multiplication operation in the parallel model of computation, various types of "butterfly" operation are investigated in order to speed up the calculation of the FFT. A modification of the algorithm in which bit-reversed ordering is not performed is considered. A scheme for calculation redistribution among processors is proposed, in which data is localized by the boundaries of a parallel processor.

Problem statement

Consider the calculation in the parallel model of computation. Let $R_{2N} = U_N \cdot V_N$, where U_N, V_N – N-digit positive integers, R_{2N} – $2N$-digit integer. Let us present the number in the form:

$$U_N = (u_{N-1}u_{N-2}...u_0) = \sum_{i=0}^{N-1} u_i 2^{\omega i},$$

$$V_N = (v_{N-1}v_{N-2}...v_0) = \sum_{i=0}^{N-1} v_i 2^{\omega i},$$

$$R_{2N} = (r_{2N-1}r_{2N-2}...r_0) = \sum_{i=0}^{2N-1} r_i 2^{\omega i},$$

where ω – the length of the machine word in bits (further we will assume ω=16, 24, 32 or 64 bits), $0 \le u_i, v_i, r_i < 2^{\omega}$.

It needs for the parallel model of computation to develop an algorithm for the implementation of the operation of multiplication of two N-digit numbers based on FFT avoiding the bit-reversed ordering.

The basis multiplication algorithm calculating DFT in the field of complex numbers (DFT- $2N$ **)**[6]. In this algorithm, N leading zeros are added to each of N-digit numbers. The $2N$-digit numbers are represented in the form of real $2N$-digit column vectors, which are input parameters in the DFT. The transition from N- to $2N$-digit numbers and the use of DFT of the length of $2N$ elements is explained by the fact that the result of multiplication of two N-digit numbers gives the number of the length of $2N$ digits.

Algorithm 1.

Multiplication of two N-digit numbers calculating DFT of the length of $2N$ in the field of complex numbers.

1. Add leading zeros:

$$X_{2N}(N+r) \leftarrow Y_{2N}(N+r) \leftarrow 0, r = \overline{0, N-1}.$$

2. Calculate **DFT of the length of 2*N***:

$$\hat{X}_{2N} \leftarrow W_{2N.2N} \cdot X_{2N}, \ \hat{Y}_{2N} \leftarrow W_{2N,2N} \cdot Y_{2N},$$

де $W_{2N}^{\langle r \cdot k \rangle_{2N}} = e^{-\frac{2\pi \cdot i}{2N} \cdot r \cdot k} = e^{-\frac{\pi \cdot i}{N} \cdot r \cdot k}$, $i = \sqrt{-1}$, $r = \overline{0, 2N-1}$, $k = \overline{0, 2N-1}$, елементи матриці $W_{2N.2N}$.

3. Multiply **DFT of the length of 2*N***:

$$\hat{Z}_{2N}(r) \leftarrow \hat{X}_{2N}(r) \cdot \hat{Y}_{2N}(r), \ r = \overline{0, 2N-1}.$$

4. Calculate **IDFT**:

$$Z_{2N} \leftarrow \frac{1}{2N} \cdot W_{2N.2N} \cdot \hat{Z}_{2N}^*$$

$$(\text{or } \frac{1}{2N} \cdot W_{2N,2N}^* \cdot \hat{Z}_{2N}).$$

5. Calculate the result:

$$Z_{2N}(r) \leftarrow [\mathrm{Re}\, Z_{2N}(r)], \ r = \overline{0, 2N-1},$$

where $[\mathrm{Re}\, Z_{2N}(r)]$ – rounding to the nearest integer of real part of $Z_{2N}(r)$.

Further on in the text, a complex signal is used, each element of which is a complex number. A complex signal, in which each element in the complex part is zero, represents a multi-digit number that is an input parameter or the result of an algorithm. A vector whose elements are complex numbers represents a complex signal in the case of software implementation.

Analysis of calculation types of "butterfly" operation

16 "butterfly" calculation types were investigated according to four characteristics: "butterfly" calculation using decimation-in-time and decimation-in-frequency, performing bit-reversed ordering before or after FFT, increasing or decreasing the usage of Fourier series coefficients at each step, using the list of Fourier series coefficients in direct or bit-reversed order. For further analysis, 4 types were chosen, which are presented in Fig. 1–4.

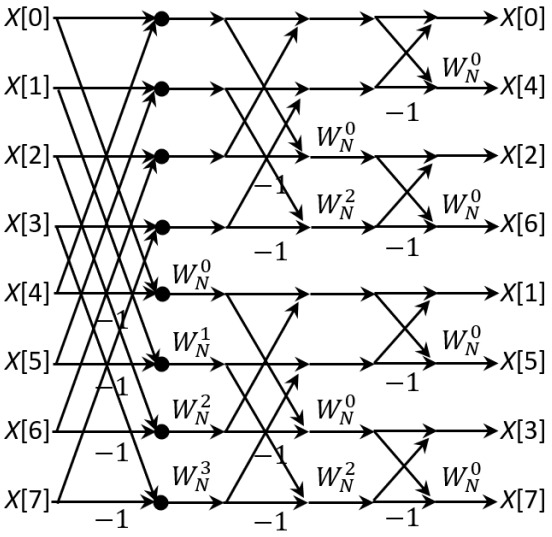

Fig. 1. "Butterfly" calculation based on decimation-in-frequency method, input signal in direct order, Fourier series coefficients in direct order, N=4, $FFTFD_8$

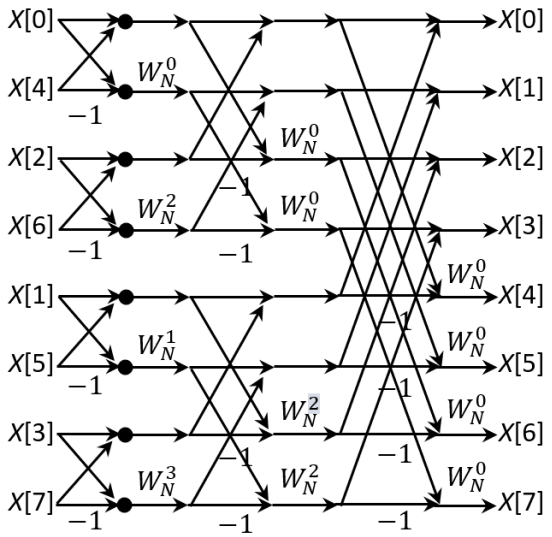

Fig. 3. "Butterfly" calculation based on decimation-in-frequency method, input signal in bit-reversed order, Fourier coefficients in bit-reversed order, N=4, $FFTFB_8$

In Fig. 2 and Fig. 3 the input signal must be bit-reversed. In Fig. 1 the type of "butterfly" calculation based on decimation-in-frequency is presented. In the first step of the calculation scheme (Fig. 1), Fourier series coefficients are used in direct order W_4^0, W_4^1, W_4^2, W_4^3. Obtaining the result requires performing a bit-reversed ordering to complete calculation.

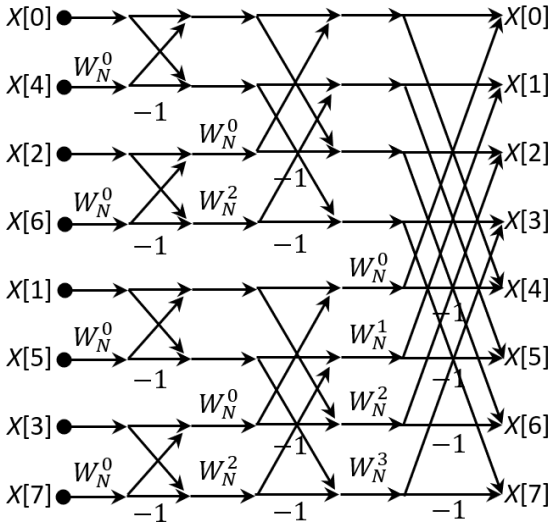

Fig. 2. "Butterfly" calculation based on decimation-in-time method, input signal in bit-reversed order, Fourier series coefficients in direct order, N=4, $FFTTB_8$

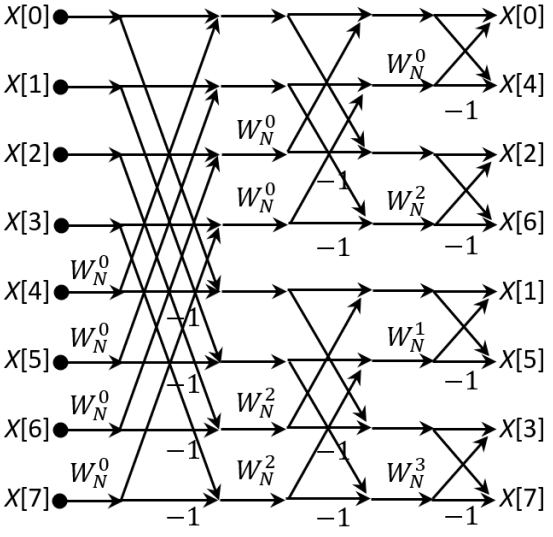

Fig. 4. "Butterfly" calculation based on decimation-in-time method, input signal in direct order, Fourier series coefficients in bit-reversed order, N=4, $FFTTD_8$

In Fig. 2 “butterfly” type based on decimation-in-time method is presented. In the last step of the calculation, Fourier series coefficients are used in bit-reversed order W_4^0, W_4^2, W_4^1, W_4^3. It needs to perform a bit-reversed ordering before the scheme (Fig. 2).

In “butterfly” based on decimation-in-frequency method, addition and subtraction operations are first calculated, then Fourier series coefficient is multiplied by the result of the subtraction. In “butterfly” operation based on decimation-in-time, the calculation is done in the opposite way: first, the Fourier series coefficient is multiplied by an element, which is then added and subtracted accordingly.

In Fig. 1–4 notations $FFTFD_8$, $FFTTB_8$, $FFTFB_8$, $FFTTD_8$ are used. *FFT* is a common notation for Fast Fourier Transform algorithm. The next letter in the 4th position from the left, *F* (*Frequency*), denotes decimation-in-frequency, and the letter *T* (*Time*) means decimation-in-time. The last letter in the 5th position from the left *D* (*Direct*) means that the input signal is in direct order, the letter *B* (*Bit-reversed*) – the input signal is in bit-reversed order. The subscript in the lower case indicates the length of the FFT in complex elements.

The decimation-in-frequency schemes are derived from each other (see Fig. 1 and Fig. 3). In order to go to the scheme of Fig. 3, it needs to perform a bit-reversed ordering of the horizontal lines of the scheme in Fig. 1. Similarly, it can be shown decimation-in-time schemes (Fig. 2 and Fig. 4) are derived from each other.

Examining the types of schemes (Fig.1–4), you can see that if a direct signal is applied to decimation-in-frequency scheme, then the scheme uses a direct list of Fourier series coefficients. If a bit-reversed complex signal is applied to the scheme, then it needs a list of Fourier series coefficients in bit-reversed order. For types of decimation-in-time schemes, this relationship is opposite.

Next, in the form of a table for a more convenient analysis, the main characteristics of the types of calculations according to the “butterfly” scheme, shown in Fig. 1–4.

Table 1. The main characteristics of the analysed “butterfly” scheme types (Fig. 1–4)

Characteristics of scheme	Calculation “butterfly” type according to scheme			
	$FFTFD_8$ (Fig. 1)	$FFTTB_8$ (Fig. 2)	$FFTFB_8$ (Fig. 3)	$FFTTD_8$ (Fig. 4)
Decimation-in-	frequency	time	frequency	time
Bit-reversed ordering is performed	at the end	at the beginning	at the beginning	at the end
The number of Fourier series coefficients at each step	decreased	increased	decreased	increased
The list of Fourier series coefficients is used in order	direct	direct	bit-reversed	bit-reversed

Implementation of the multiplication based on the "butterfly" calculation

From a practical point of view, not all of the 16 investigated types of the "butterfly" scheme are suitable for the implementation of the multiplication based on the FFT. So, for example, for the scheme type based on decimation-in-time method, a decrease in the number of Fourier series coefficients, the first step involves the coefficients multiplication by zero elements in the high part of the signal.

In case of implementation of the multiplication operation in Algorithm 1 the calculation of forward and inverse FFT only one type of calculation "butterfly" scheme from Fig. 1–4 is used.

For finding the result of the forward and inverse Fourier transform in Algorithm 1, the different types of the “butterfly” schemes could be used considering the bit-reversed ordering (Fig. 1–4).

Bit-reversed ordering

Many versions of the multiplication operation prefer the input signal in direct order to calculate “butterfly”. Although a

single bit-reversed ordering step can have a non-negligible effect on the overall speed and can be performed with linear complexity, much work has been devoted to the optimisation of bit-reversed ordering since the time of FFT. In the case of software implementation for longer FFTs, reorder algorithms become more complicated.

The Stockham auto-sort algorithm [7], [8] typically writing back and forth between two arrays, transposing one "digit" of the indices with each stage, and has been especially popular on SIMD architectures [9], [10]. Such algorithms maintain direct ordering at each step, but at the expense of additional sorting costs.

Implementation of the multiplication operation based on the FFT in the parallel model of computation

The number of synchronization steps has a great influence on the effective implementation of algorithms with the possibility of redistributing operations among processors. During the execution of a synchronization instruction, the processors are stopped to wait for the rest of the processors to finish executing, considering the SIMD architecture. The bit-reversed ordering stage requires synchronization. The longer the bit-reversed ordering length, the more processors stop at the synchronization stage. Localizing the reordering size reduces the overall execution time. It is shown further that the bit-reversed ordering steps can be avoided completely.

As mentioned earlier, the forward and inverse Fourier transforms in Algorithm 1 can be computed using different “butterfly” schemes. Consider the schemes in Fig. 1 and 2. If for the scheme on Fig. 1 bit-reversed reordering is required after calculation, then for the schema in Fig. 2 the bit-reversed ordering needs to be done before starting the scheme. In Algorithm 1, steps 2 and 4 of calculating the forward and inverse Fourier transforms are separated by step 3 of element-wise multiplication. The direct or bit-reversed order of the element-wise multiplication does not affect the result of step 3. Given the fact that the bit-reversed ordering is performed consecutively twice, leads to the original order, it is possible to exclude the bit-reversed ordering, if in step 2 of Algorithm 1 the calculation takes place according to the scheme in Fig. 1, and at step 4, the calculation takes place according to the scheme in Fig. 2.

One of the advantages of eliminating bit-reversed ordering is the separation of computation with the localization of the processed data. In Fig. 5 the calculation distribution scheme among four processors for implementation of two 4-digit numbers multiplication based on the FFT of the length of 8 elements is presented. Bit-reversed ordering is excluded in the scheme (Fig. 5).

The calculation of the forward DFT is based on the decimation-in-frequency scheme $FFTFD_4$ of the length of 4 elements over the signal presented in the direct order (Fig. 1). The calculation of the inverse DFT is based on the decimation-in-time scheme $FFTTB_4$ of the length of 4 elements and the result will be in direct order (Fig. 2). The scheme distributes the computation among four processors to compute two DFTs and one IDFT of the length of 8 elements. Processors 2 and 3 are partially used only for calculating the DFT of the Y_8.

Three vertical lines (Synchronization) in Fig. 5 indicate calculation steps relatively when synchronization of processors (or memory cells) is required. Although synchronization is marked relatively and there might be more synchronization commands in the case of implementing the scheme, in case of avoiding the bit-reversed ordering the number of synchronization commands is reduced and, accordingly, the algorithm execution time is reduced.

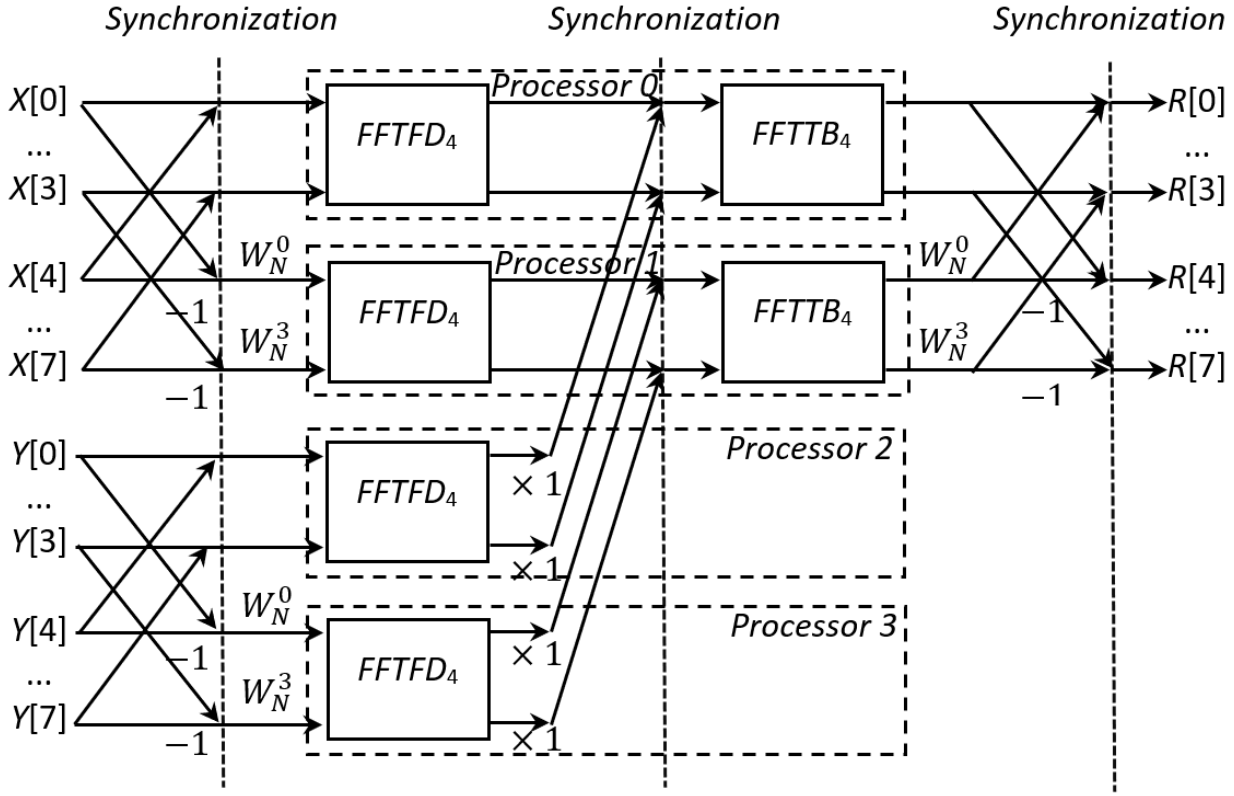

Fig. 5. The calculation distribution scheme on four processors of the implementation of two 4-digit numbers multiplication based on the FFT and avoiding the bit-reversed ordering, N=4, $FFTFD_4$ – scheme "butterfly" is based on decimation-in-frequency (Fig. 1) and uses Fourier series coefficients in direct order $(W_4^0, W_4^1, W_4^2, W_4^3)$, $FFTTB_4$ – scheme "butterfly" is based on decimation-in-time (Fig. 2) and uses Fourier series coefficients in bit-reversed order $(W_4^0, W_4^2, W_4^1, W_4^3)$

According to Algorithm 1 and the scheme in Fig. 5, the implementing of the two 4-digit numbers multiplication is based on FFT of the length of 8 elements. This length of 8 elements is explained by the fact that the result of multiplication of two 4-digit numbers will be the number of the length of 8 digits. Therefore, to the initial signals X_4, Y_4 of the length of 4 elements at the beginning four zeros are added to the higher digits (X[4]...X[7], Y[4]...Y[7] (Fig. 5). The first step of the "butterfly" on the left is optional but it is presented to preserve the general understanding of the calculation.

In Fig. 5 the calculation distribution scheme for the implementation of the multiplication of two 4-digit numbers is shown. The calculation distribution scheme on four processors can be used to implement the multiplication of N-digit numbers. Then the calculation of the forward and inverse Fourier transform should be based on the $FFTFD_N$ and $FFTTB_N$ schemes.

Similarly to the scheme in Fig. 5, it is possible to build a calculation distribution scheme between processors based on the $FFTFB_4$ (Fig. 3) and $FFTTD_4$ (Fig. 4) "butterfly" schemes. Then the calculation of the forward DFT should be based on the $FFTTD_4$ scheme, the input of which is in direct order, and the bit-reversed signal is an input for the $FFTFB_4$ scheme, the result of which will be a in direct order. Unlike the scheme in Fig. 5, according to Table 1, the calculation of the $FFTTD_4$ and $FFTFB_4$ schemes takes place using the list of Fourier series coefficients in bit-reversed order.

The scheme in Fig. 5 can be adapted to distribute the computation over a larger number of processors. In this case, it is necessary to use an FFT of a shorter length and the volume of calculations at the beginning and at the end of the scheme increases.

```
Xc←FFTFreqDirect arg;Bv;N;S;j;d;n;p;k;s1;s2;r;i1;
XDc WDv N←arg
⍝ Decimation in Frequency FFT algorithm
⍝ Input signal in direct order
⍝ XDc - vector of complex signal in
⍝       the direct order of length N
⍝ WDv - vector of Furie cofficients in
⍝       the direct order of length of N/2
⍝ N   - number elements in the vector XDc
f←⎕IO ⋄ j←1 ⋄ d←N÷2 ⋄ n←log2 N
Xc←XDc
:For p :In 0,⍳n-1
    :For r :In 0,⍳j-1
        s1←r×(2×d) ⋄ s2←s1+d
        :For k :In 0,⍳d-1
            i1 i2←(s1 s2)+k
            X←(Xc[f+i1]-Xc[f+i2])×WDv[f+k×j]
            Xc[f+i1]←Xc[f+i1]+Xc[f+i2]
            Xc[f+i2]←X
        :EndFor
    :EndFor
    j←j×2 ⋄ d←d÷2
:EndFor
```

Fig. 6. Algorithm for calculating FFT based on decimation-in-frequency method and input vector in direct order, $FFTFD_N$

Peculiarities of the implementation of algorithms for calculating the FFT with decimation-in-frequency and decimation-in-time. In Fig. 6–9 the 4 algorithms for implementing the FFT algorithm based on decimation-in-frequency and decimation-in-time methods, with elements of the input vectors provided in direct and bit-reversed orders. In each implementation algorithm there is a variable *f*, the value of which is the initial index of the element in array. The value of this variable considers that different technologies have different initial array indices (0 or 1).

```
Xc←FFTTimeBitReversed arg;Bv;N;S;j;d;n;p;k;s1;s2;r
XBc WDv N←arg
⍝ Decimation in Time FFT algorithm
⍝ Input signal in bit-reversed order
⍝ XBc - vector of complex signal in
⍝       the bit-reversed order of length N
⍝ WDv - vector of Furie cofficients in
⍝       the direct order of length of N/2
⍝ N   - number elements in the vector XBc
f←⎕IO ⋄ j←1 ⋄ d←N÷2 ⋄ n←log2 N
Xc←XBc
:For p :In 0,⍳n-1
    :For r :In 0,⍳d-1
        s1←r×(2×j) ⋄ s2←s1+j
        :For k :In 0,⍳j-1
            i1 i2←(s1 s2)+k
            X←Xc[f+i2]×WDv[f+k×d]
            Xc[f+i2]←Xc[f+i1]-X
            Xc[f+i1]←Xc[f+i1]+X
        :EndFor
    :EndFor
    j←j×2 ⋄ d←d÷2
:EndFor
```

Fig. 7. Algorithm for calculating FFT based on decimation-in-time method and input vector in bit-reversed order, $FFTTB_N$

```
Xc←FFTFreqBitReversed arg;XBc;WBv;N;f;j;d;n;p;k;s1;s
XBc WBv N←arg
⍝ Decimation in Frequency FFT algorithm
⍝ Input signal in bit-reversed order
⍝ XBc - vector of complex signal in
⍝       the bit-reversed order of length N
⍝ WBv - vector of Furie cofficients in
⍝       the bit-reversed order of length of N/2
⍝ N   - number elements in the vector XBc
f←⎕IO ⋄ j←1 ⋄ d←N÷2 ⋄ n←log2 N
Xc←XBc
:For p :In 0,⍳n-1
    :For k :In 0,⍳d-1
        s1←k×(2×j) ⋄ s2←s1+j
        :For r :In 0,⍳j-1
            i1 i2←(s1 s2)+r
            X←(Xc[f+i1]-Xc[f+i2])×WBv[f+k]
            Xc[f+i1]←Xc[f+i1]+Xc[f+i2]
            Xc[f+i2]←X
        :EndFor
    :EndFor
    j←j×2 ⋄ d←d÷2
:EndFor
```

Fig. 8. Algorithm for calculating FFT based on decimation-in-frequency method and input vector in bit-reversed order, $FFTFB_N$

```
Xc←FFTTimeDirect arg;Bv;N;S;j;d;n;p;k;s1;s2;r;i1;
XDc WBv N←arg
⍝ Decimation in Time FFT algorithm
⍝ Input signal in direct order
⍝ XDc - vector of complex signal in
⍝       the direct order of length N
⍝ WBv - vector of Furie cofficients in
⍝       the bit-inverse order of length of N/2
⍝ N   - number elements in the vector XDc
f←⎕IO ⋄ j←1 ⋄ d←N÷2 ⋄ n←log2 N
Xc←XDc
:For p :In 0,⍳n-1
    :For k :In 0,⍳j-1
        s1←k×(2×d) ⋄ s2←s1+d
        :For r :In 0,⍳d-1
            i1 i2←(s1 s2)+r
            X←Xc[f+i2]×WBv[f+k]
            Xc[f+i2]←Xc[f+i1]-X
            Xc[f+i1]←Xc[f+i1]+X
        :EndFor
    :EndFor
    j←j×2 ⋄ d←d÷2
:EndFor
```

Fig. 9. Algorithm for calculating FFT based on decimation-in-time method and input vector in direct order, $FFTTD_N$

For the convenience of analysing the features of the algorithm implementations, the FFT calculations are listed in Table 2.

The algorithms are implemented using the same variables and the same structure to see similarity or difference. All algorithms have three levels of cycles. The variable *k* is used to find the element from the list of Fourier series coefficients that are passed to the algorithm in direct (*WDv*) or bit-reversed (*WBv*) order. Similarly, the input vector with elements in direct (*XDv*) and bit-reversed (*XBv*) order is marked.

An example of calculation (Fig. 5). The algorithm is implemented in the APL[11] programming language to check the correctness of the algorithm. which uses the $FFTFD_4$ scheme (decimation-in-frequency) to calculate the forward DFT, and $FFTTB_4$ (decimation-in-time) to calculate the IDFT.

The step-by-step calculation results are shown in Table 3 on the example of multiplication 1112·1112=1236544 two 4-digit numbers. To simplify, the same numbers are multiplied. The series of digits is presented in the form of vectors, in which the elements are indexed from left to right.

Table 2. Peculiarities of the algorithm implementations for calculating the FFT based on decimation-in-frequency and decimation-in-time, input vector in direct and bit-reversed orders

	FFT calculation algorithm			
	$FFTFD_N$ (Fig. 6)	$FFTTB_N$ (Fig. 7)	$FFTFB_N$ (Fig. 8)	$FFTTD_N$ (Fig. 9)
Decimation-in-	frequency	time	frequency	time
Order of input vector is	direct	bit-reversed	bit-reversed	direct
Fourier series coefficients are in order	direct	direct	bit-reversed	bit-reversed
Input parameters	*XDc, WDv*	*XBc, WDv*	*XBc, WBv*	*XDc, WBv*
The cycle of the 2nd level is built on variables	*r, j*	*r, d*	*k, d*	*k, j*
The cycle of the 3rd level is built on variables	*k, d*	*k, j*	*r, j*	*r, d*

XDc – elements of input vector are in direct order
XBc – elements of input vector are in bit-reversed order
WDv – vector with Fourier series coefficients in direct order
WBv – vector with Fourier series coefficients in bit-reversed order

Table 3. An example of calculating the multiplication of two 4-digit numbers

Step	Scheme	The result of calculating the elements of the vectors at each step
	Input	$X_4=Y_4=(2, 1, 1, 1)$ $W_4=(1, 0.70711–j0.70711, –j, –0.70711–j0.70711)$
1		$X_8=Y_8=(1, 1, 1, 1, 0, 0, 0, 0)$
2	$FFTFD_4$, $p=0$	$X_8=Y_8=(2, 1, 1, 1, 2, 0.70711–j0.70711, –j, –0.70711–j0.70711)$
2	$FFTFD_4$, $p=1$	$X_8=Y_8=(3, 2, 1, 0, 2–j, –j1.41422, 2+j, –j1.41422)$
2	$FFTFD_4$, $p=2$	$X_8=Y_8=(5, 1, 1, 1, 2–j2.41422, 2+j0.41422, 2–j0.41422, 2+j2.41422)$
2		Bit-reversed ordering is not performed
3		$Z_8= X_8·Y_8 =(25, 1, 1, 1, –1.82846–j9.65688, 3.82842+j1.65688, 3.82842–j1.65688, –1.82846+j9.65688)$
3	*conjugate element*	$Z_8=(25, 1, 1, 1, –1.82846+j9.65688, 3.82842–j1.65688, 3.82842+j1.65688, –1.82846–j9.65688)$
4		Bit-reversed ordering is not performed
4	$FFTTB_4$, $p=0$	$Z_8=(26, 24, 2, 0, 1.99993+j8, –5.65688+j11.31376, 1.99996–j8, 5.65688+j11.31376)$
4	$FFTTB_4$, $p=1$	$Z_8=(28, 24, 24, 24, 3.99992, 5.65688+j5.65688, j16, –16.97064+j16.97064)$
4	$FFTTB_4$, $p=2$	$Z_8=(31.99992, 32.00007, 40, 48.00022, 24.00008, 15.99993, 8, ¯0.00022)$
4	*Division by 8*	$Z_8= (3.99999, 4.00001, 5, 6.00003, 3.00001 1.99999, 1, ¯0.00003)$
5	*Rounding*	$Z_8=(4, 4, 5, 6, 3, 2, 1, 0)$

The result of the calculation is a number 1236544=1112·1112

Conclusions. In the paper for the parallel model of computation, a modification of the method of implementing the multiplication of multi-digit numbers based on the fast Fourier transform with avoiding bit-reversed ordering is proposed. The paper investigates the calculation of the FFT based on decimation-in-frequency "butterfly" scheme and the calculation of the IFFT based on decimation-in-time "butterfly" scheme allows to exclude bit-reversed ordering. A scheme for distributing calculations between four processors is proposed, in which forward and inverse transformation calculations are localized within one parallel processor. The proposed modification does not reduce the computational complexity in terms of the number of complex operations, but due to the exclusion of bit-reversed ordering, the number of synchronizations among processors and data is reduced, which reduces the algorithm execution time. The scheme can be adapted to distribute the computations among a larger number of processors. Four algorithms for implementing FFT based on decimation-in-frequency and decimation-in-time methods, input vector with elements in direct and bit-reversed orders are presented. To check the result of the calculation, the algorithm of multiplication with avoiding the bit-reversed ordering was implemented in the APL programming language. An example of calculation is given in the form of a table.

References

1. Gauss, Carl Friedrich (1876) [n.d.]. Theoria Interpolationis Methodo Nova Tractata. Carl Friedrich Gauss Werke. Vol. Band 3. Göttingen: Königliche Gesellschaft der Wissenschaften. pp. 265–327.
2. Heideman M. T., D. H. Johnson, and C. S. Burrus, Gauss and the history of the fast Fourier transform, IEEE ASSP Magazine, 1, (4), 14-21 (1984).
3. James W. Cooley, John W. Tukey: An algorithm for the machine calculation of complex Fourier series. In: Math. Comput. 19, 1965, S. 297—301.
4. G. C. Danielson and C. Lanczos, Some improvements in practical Fourier analysis and their application to X-ray scattering from liquids, J. Franklin Inst. 233, 365–380 and 435–452 (1942).
5. V. K. Zadiraka and S. S. Melnykova, Digital Signal Processing. [in Russian], Naukova Dumka, Kyiv. 294 p. (1993).
6. V. K. Zadiraka and A. M. Tereshchenko, Computer Arithmetic of Multi-Bit Numbers in Sequential and Parallel Computational Models [in Ukrainian], Naukova Dumka, Kyiv (2021).
7. Originally attributed to Stockham in W. T. Cochran et al., What is the fast Fourier transform, Proc. IEEE vol. 55, 1664–1674 (1967).
8. P. N. Swarztrauber, FFT algorithms for vector computers, Parallel Computing vol. 1, 45–63 (1984).
9. Swarztrauber, P. N. (1982). Vectorizing the FFTs. In Rodrigue, G. (ed.). Parallel Computations. New York: Academic Press. pp. 51–83.
10. Pease, M. C. An adaptation of the fast Fourier transform for parallel processing. J. ACM. 15 (2), 1968. P. 252–264. doi:10.1145/321450.321457.
11. TryAPL page https://tryapl.org/.

The article has been sent to the editors 30.11.22
After processing 15.12.22