Design Method of Real-Time Parallel-Pipeline Processors Computing the Vector DFT [1]

Alexander A. Petrovsky [1,2], Michal V.Kachinsky [2]

[1] University of Technology Bialystok, Computer Science Department (Poland)

[2] Belarusian State University of Informatics and Radioelectronics, Computer Engineering Department
6, P.Brovky Str., 220027, Minsk, Republic of Belarus
E-mail: palex@org.by

Abstract - **Offered algorithm allows to built the structure of the parallel pipeline FFT processors (PPFFT-processor) computing the vector DFT in real time with the minimum structural complexity at given parameters: speed of input data receipt; structure of a computing element (arithmetic device) and time of the butterfly operation execution. The considered approach to structural synthesis of the PPFFT-processors for R-dimensional signal processing allows to receive the structure of the processor under given restrictions of a specific problem and is the basis for the decision of questions of automated design of PPFFT-processors at a structural level.**

I. INTRODUCTION

At the moment the special place at realisation of scientific researches is occupied by information systems, directly connected with real objects, ensuring the tax and processing of the received information, control of fast-following processes in real time. The number of channels (R) being subject to processing in such systems in various cases changes from 8 up to 256 and more, the rate of data receipt on an input of processing system in each channel reaches tens kHz, volume of fetch on each channel is 256 ...1024 and more points. Current data acquisition technology easily achieves data rates of 25-50 million samples per second (MSPS) for real time operation, with expected near term growth to 500 MSPS. The special difficulty is caused thus by realisation of DFT R-dimensional processes - vector DFT in real time.

Definition 1. *The R-dimensional process is the vector*

$$\mathbf{x} = [x_1(n)x_2(n)...x_R(n)]^{\mathrm{T}}, \quad (1)$$

where $x_r(n), r = \overline{1,R}; n = \overline{0,N-1}$ *is a time-sequence* $\{x_r(n)\}$ *with N elements. Based on the properties of DFT the equation (1) is written in D-dimensional complex vector*

$$\mathbf{z} = [z_1(n)z_2(n)...z_D(n)]^{\mathrm{T}}, \quad (2)$$

where $z_d(n) = x_{2d-1}(n) + jx_{2d}(n), d = \overline{1,D}; D = R/2,$ *and if R is even D=R/2 else D=(R+1)/2 (R is odd), T indicates the transpose.*

Definition 2. *The R-dimensional DFT (vector DFT)* **(VDFT)** *of vector (2) is then defined as vector*

$$\mathbf{U} = [U_1(k)U_2(k)...U_D(k)]^{\mathrm{T}},$$

where $U_d(k), d = \overline{1,D}; k = \overline{0,N-1}$ *is the DFT of the sequences* $\{z_d(n)\}$ *of D-dimensional vector* $\mathbf{z}$, *k is the frequency number.*

Thus: **VDFT**: $\mathbf{z} \Rightarrow \mathbf{U}$ *for each* $n,k,d;$ $n = \overline{0,N-1}; k = \overline{0,N-1}; d = \overline{1,D};$ $U_d(k) = \mathbf{DFT}[\, z_d(n)\,],$ *where* $\mathbf{DFT}[\, z_d(n)\,]$ *is the DFT of sequence* $\{\, z_d\,(n)\,\}$.

From definition 2 follows, that vector DFT is independent, joint transformation of a *D* components of a *D*-dimensional process $\mathbf{z}$ and represents set DFT of a format N. Based on the definitions 1 and 2 it is easily proofed (using the matrix form for DFT [3]) of the algorithms for computing of vector DFT for complex *D*-dimensional process $\mathbf{z}$ and real R-dimensional process $\mathbf{x}$.

The main characteristic of these real-time applications is that continuous flow of data has to be processed without interruption The pipelined FFT appears to be predominate architecture for high throughput real time applications [4, 5]. There are many papers have been written about high performance FFT-processor design [5, 6]. The purpose of the given article is to show the formalised approach to the decision of a problem of structural synthesis of parallel-pipeline FFT processors for R-dimensional signal processing. This is a new approach to high performance FFT hardware design. An automated design tool, that generates both the hardware and software for an FFT processor, was developed for programmable logic.

[1] This work was supported by the Computer Science Department at the University of Technology Bialystok (Poland) under the grant W/II/6/96

II. Computation of the vector DFT (vectors with larger lengths)

Proposition 1. *Let* $\mathbf{z}$ *is the D-dimensional vector process (2) and the sequences* $\{z_d(n), n = \overline{0, N-1}\}$ *is given by vector columns*

$$\mathbf{z}_d = [z_d(0) z_d(1) ... z_d(N-1)]^{\mathrm{T}},$$

then the vector DFT $\mathbf{U}$ *of vector* $\mathbf{z}$ *in matrix form is defined as follows:*

$$\mathbf{U} = (\mathbf{W}_N \otimes \mathbf{I}_D)\mathbf{z} \tag{3}$$

where $\otimes$ *the multiplication sign in equation (3) indicates the direct, or Kronecker, product of the submatrices.*

The proof of this proposition is based on the properties of direct multiplication of submatrices $\mathbf{W}_N \otimes \mathbf{I}_D$ and vector $\mathbf{z}$. In practice, the components $x_r(n)$ of a vector process $\mathbf{x}$ is the real sequences $\{x_r(n), n = \overline{0, N-1}\}$.

Proposition 2. *Let* $\mathbf{x}$ *is the R-dimensional process (1), and obtain the new D-dimensional vector* $\mathbf{z}$ *in accordance with definition 1, and the DFT of a vector* $\mathbf{z}$ *is computed by using equation (3)*

$$\mathbf{U} = (\mathbf{W}_N \otimes \mathbf{I}_D)\mathbf{z} = \mathrm{Re}\,\mathbf{U} + j\,\mathrm{Im}\,\mathbf{U}$$

where $\mathrm{Re}\,\mathbf{U}$ *and* $\mathrm{Im}\,\mathbf{U}$ *are real and imaginary parts of the vector* $\mathbf{U}$ *, then the vectors*

$$\mathrm{Re}\,\mathbf{X}_r = [\mathrm{Re}(X_r(0))\,\mathrm{Re}(X_r(1)) ... \mathrm{Re}(X_r(N/2-1))]^{\mathrm{T}},$$

$$\mathrm{Im}\,\mathbf{X}_r = [\mathrm{Im}(X_r(0))\,\mathrm{Im}(X_r(1)) ... \mathrm{Im}(X_r(N/2-1))]^{\mathrm{T}}$$

are respectively defined as follows:

$$[\mathrm{Re}\mathbf{X}_1\,\mathrm{Im}\mathbf{X}_2\,\mathrm{Re}\mathbf{X}_3\,\mathrm{Im}\mathbf{X}_4 ... \mathrm{Re}\mathbf{X}_{2D-1}\,\mathrm{Im}\mathbf{X}_{2D}]^{\mathrm{T}} =$$

$$((\mathbf{E}_N + \mathbf{I}_{N/2} \otimes \mathbf{B}') \otimes \mathbf{E}_D)\,\mathrm{Re}\,\mathbf{U},$$

$$[\mathrm{Re}\mathbf{X}_2\,\mathrm{Im}\mathbf{X}_1\,\mathrm{Re}\mathbf{X}_4\,\mathrm{Im}\mathbf{X}_3 ... \mathrm{Re}\mathbf{X}_{2D}\,\mathrm{Im}\mathbf{X}_{2D-1}]^{\mathrm{T}} =$$

$$((\mathbf{E}_{N/2} \otimes \mathbf{B}'' + \mathbf{I}_N) \otimes \mathbf{E}_D)\,\mathrm{Im}\,\mathbf{U}, \tag{4}$$

where $\mathbf{E}_N$ *is the permutation N×N matrix,* $\mathbf{B}' = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}, \mathbf{B}'' = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$ *and* $\mathbf{I}_D$ *is the N×N unit matrix.*

The proof of this proposition is the same of the proof of proposition 1. The proposition permits to formulate integrated algorithm of computation DFT R-dimensional real process $\mathbf{x}$: 1) generate a complex D- dimensional vector $\mathbf{z}$; 2) calculate DFT of a vector $\mathbf{z}$; 3) define a vectors $\mathrm{Re}\,\mathbf{X}_r$ and $\mathrm{Im}\,\mathbf{X}_r, r = \overline{1, R}$.

Computation of the vector DFT (*vectors with larger lengths*) is based on the fact that the FFT algorithm (Stockham algorithm [4]) allows to reduce the computing of one-dimensional array of size DN. This method is based on the next **theorem [5]:** *let the elements of* $z_d(n), d = \overline{0, D-1}; n = \overline{0, N-1}$ *of* D*-complex sequences* $\{z_d(n)\}$ *of length* $N = 2^l$ *are structured as the one-dimensional array* $z(n^*)$ *of size DN as following:* $z(n^*) = z_d(n), \quad n^* = nD + d, n^* = \overline{0, DN-1}$, *then, applying the direct FFT algorithm to the base 2 to the array* $z(n^*)$ *and stopping the algorithm after l iterations yields the D Fourier transforms of size N.*

A similar theorem can be proved as well as for the inverse FFT algorithm only the D-dimensional vector process $\mathbf{z}$ is mapped into the one-dimensional array $z(n^*)$ of size DN by means of index transform:

$$z(n^*) = z_d(n), n^* = dN + n, n^* = \overline{0, DN-1}.$$

On the basis of propositions 1, 2 and the theorems follows the next proposition.

Proposition 3. *Let by means of index transform* $n^* = nD + d$ *or* $n^* = dN + n$ *where* $n^* = \overline{0, DN-1}$; $n = \overline{0.N-1}$; $d = \overline{0, D-1}$; $N = 2^l$, D*-dimensional vector process* $\mathbf{z}$ *given in (1) is mapped into the one-dimensional array* $z(n^*)$ *as following:* $z(n^*) = z_d(n)$, *where* $z_d(n)$ *- are the components of the vector process* $\mathbf{z}$. *Then the result of performance of first l iterations* $u(k^*), k^* = \overline{0, DN-1}$ *of direct and inverse FFT algorithm to the base 2 of the array* $z(n^*)$ *correspondingly is DFT of initial D-dimensional vector process* $\mathbf{z}$ *(vector DFT). At that the DFT of* $u_d(k)$ *components of D-dimensional vector process* $\mathbf{z}$ *is arriving by inverse mapping of one-dimensional array* $u(k^*)$ *of size* DN $u_d(k) = u(k^*), d = \overline{0.D-1}; k = \overline{0, N-1}$ *by means of index transform correspondingly*

$$d = k^* \bmod D, k = [k^*/D],$$

or

$$d = [k^*/N], k = k^* \bmod N,$$

where $[a]$ *- is the integer part of a.*

Proposition 3 has two important corollaries.

Corollary 1. The computing of vector DFT (the set *D* of one-dimensional transformations of size *N*) is reduced to computing of one one-dimensional *DN*-point DFT.

Corollary 2. The computing of one *DN*-point DFT where $D = 2^q, N = 2^l$ allows at the minimal quantity of additional equipment to compute from 1 to $DN/2$ transformations of length from *DN* to 2 correspondingly, changing only the number of performed iterations in the FFT algorithm to the base 2 from *q* + *l* to 1 correspondingly.

III. STRUCTURAL LANGUAGE OF PPFFT PROCESSOR DESCRIPTION

The parallel-pipeline structure is considered as application to an input data of the operators sequence, which transform an one-dimensional array in two-dimensional array and carry out butterfly operation of radix 2 FFT algorithm. By the description of the PPFFT-processor structure designations system offered by Parker [7] is accepted as a basis.

The parallel-pipeline structure is considered as application to an input data of the operators sequence, which transform an one-dimensional array in two-dimensional array and carry out butterfly operation of radix 2 FFT algorithm. By the description of the PPFFT-processor structure designations system offered by Parker [6] is accepted as a basis.

The index of element of two-dimensional array is defined as following:

$$[x,y]=[\{a_s a_{s-1}...a_1\},\{b_t b_{t-1}...b_1\}],$$

where $x=a_s 2^{s-1}+a_{s-1}2^{s-2}+...+a_1$ — is number of column, $a_i=\{0,1\}$, $y=b_t 2^{t-1}+b_{t-1}2^{t-2}+...+b_1$ — is number of row, $b_i=\{0,1\}$ define the position of given element in the array. The operators describing the parallel-pipeline structure are defined by their effect on the indexes of data elements and give some permutation of elements in the array [7]:

operator $\mu_{(j,i)}$ divides every row of the initial array into 2^j rows, processing the input data by blocks of 2^i columns:

$$\mu_{(j,i)}[x,y]=\mu_{(j,i)}[\{a_s ... a_1\},\{b_t ... b_1\}]=$$

$$\left[\left\{a_s ... a_{i+1}\, a_{i-j} ... a_1\right\},\left\{b_t ... b_1\, a_i ... a_{i-j+1}\right\}\right]$$

$$\mu^{-1}_{(j,i)}[x,y]=\left[\begin{array}{l}\left\{a_s ... a_{i-j+1}\, b_j ... b_1\, a_{i-j} ... a_1\right\},\\ \left\{b_t ... b_{j+1}\right\}\end{array}\right]$$

$$\mu_{(i)}=\mu_{(1,i)},\ \mu^{-1}_{(i)}=\mu^{-1}_{(1,i)},\quad j\le i\le s;$$

operator B describes the butterfly operation of radix 2 FFT algorithm. Every data pair of column is operated at this the butterfly operation result with sum is located at the row with lower index and the butterfly operation with difference is located at the row with higher index;

operator $\delta_{(i)}$ divides every row of the initial data array into 2^i rows:

$$\delta_{(i)}[x,y]=[\{a_s ... a_{i+1}\},\{b_t ... b_1\, a_i ... a_1\}],\quad i\le s;$$

operator T(i,j) implements a permutation of 2-dimensional data array elements within the blocks consisting from 2^i rows and 2^j columns after the following rule:

$$T_{(i,j)}[x,y]=\left[\left\{a_s ... a_{j+1}\, b_i ... b_1\right\},\left\{b_t ... b_{i+1}\, a_j ... a_1\right\}\right],\ j\le s,\ i\le t;$$

operator $R_{(i)}$ after every 2^i rows permutes even and odd rows within the data blocks of 2^i rows size and is defined as following

$$R_{(i)}[x,y]=[x,\{b_t ... b_2\,(b_1\oplus b_{i+1})\}],\quad i<t,$$

where $\oplus$ is XOR operation;

operator $\varphi_{(i,j)}$ implements a permutation of data within every column of data array after the following rule

$$\varphi_{(i,j)}[x,y]=\left[x,\left\{b_t ... b_{i+j+1}\left(b_{i+j}\oplus a_i\right)b_{i+j-1} ... b_1\right\}\right],$$

$$\varphi_{(i)}=\varphi_{(i,0)},\ i\le s,\ j\le t-s$$

The following identify holds true for the considered operations $\delta_{(j)}T_{(j,i-j)}=\mu_{(i-j,i)}$, $j\le 1$ and is proved in [8].

The operators describing the parallel-pipeline structure are defined by their effect on the indexes of data elements and give some permutation of elements in the array. When describing the parallel-pipeline structure the agreement taken in [7] is executed, that is the operators are written from left to right $\pi_1\pi_2[x,y]=\pi_2(\pi_1[x,y])$. This agreement is associated with accepted direction of data movement in the processor structure and is made for precise correspondence of operators sequences to structures images.

Thus, the above considered operators permit to functionally describe the parallel-pipeline structure. However, such a description is cumbersome and inconvenient to use as it has intermediate data permutations and the operator sequence does not correspond to the processor structure unambiguously.

Therefore it is proposed to put in two additional operators providing inter unambiguous correspondence of operator sequence in the PPFFT-processor structure description:

operator $M_{(i)}$ implements the following permutation of elements of 2-dimensional array:

$$M_{(i)}[x,y]=[\{a_s ... a_{i+1}b_1 a_{i-1} ... a_1\},\{b_t ... b_2 a_i\}],\quad i\le s;$$

operator $\beta_{(i)}$ permutes the rows of original data array after the following rule:

$$\beta_{(i)}[x,y]=[x,\{b_t ... b_{i+2}b_1 b_i ... b_2 b_{i+1}\}],\quad i<t.$$

Then the following proposition holds true.

Proposition 4. The parallel-pipeline structure of computation of 2^l FFT with $m=2^r$ computing elements at a stage implementing m sequential butterfly operations can be submitted as following:

direct PPFFT-structure

$$FFT^{d}_{(l,r)} = \delta_{(r)}\mu_{(l-r)}BM_{(l-r-1)}B\ldots BM_{(1)}B\beta_{(r)}B\ldots$$
$$\ldots B\beta_{(1)}B\mu^{-1}_{(1)}\mu^{-1}_{(r,r+1)}; \quad (8)$$

inverse PPFFT-structure

$$FFT^{i}_{(l,r)} = \mu_{(r,r+1)}\mu_{(1)}B\beta_{(1)}B\ldots B\beta_{(r)}BM_{(1)}B\ldots$$
$$\ldots BM_{(l-r-1)}B\mu^{-1}_{(l-r)}\mu^{-1}_{(r,r)}; \quad (9)$$

Operator $M_{(i)}$ describes functionally intermediate memory of $2\times2^{i-1}$ words, operator $\beta_{(i)}$ describes direct (without intermediate memory) communication between two stages of the PPFFT-processor.

By means of new operator $M_{(i)}$ the pipeline structure of 2^l FFT computation is described as following:

direct pipeline structure

$$FFT^{d}_{(l)} = \mu_{(l)}BM_{(l-1)}B\ldots BM_{(1)}B\mu^{-1}_{(1)};$$

inverse pipeline structure

$$FFT^{i}_{(l)} = \mu_{(1)}BM_{(1)}B\ldots BM_{(l-1)}B\mu^{-1}_{(l)}.$$

To describe a parallel-pipeline structure with simultaneous implementation of m butterfly operations which have the numbers with *DN/2m* difference let introduce one more additional operator $\sigma_{(i)}$ which permutes the original data array rows after the following rule:

$$\sigma_{(i)}[x, y] = [x, \{b_t \ldots b_{i+2}b_i \ldots b_1 b_{i+1}\}],$$
$$\sigma^{-1}_{(i)}[x, y] = [x, \{b_t \ldots b_{i+2}b_1 b_{i+1} \ldots b_2\}],\ i < t.$$

By means of this operator the direct parallel-pipeline structure of computation of 2^l FFT with $m=2r$ computing elements at a stage implementing m butterfly operators which have the numbers with *N/2m* ($N=2^l$) is described as following:

$$FFT^{d}_{(l,r)} = \mu_{(r,l)}\mu_{(l-r)}\sigma_{(r)}B\beta_{(r)}B\ldots B\beta_{(1)}BM_{(l-r-1)}B\ldots$$
$$\ldots BM_{(1)}B\mu^{-1}_{(1)}\mu^{-1}_{(r,l)}.$$

The operator combination $\mu_{(r,l)}\mu_{(l-r)}\sigma_{(r)}$, $\mu^{-1}_{(1)}\mu^{-1}_{(r,l)}$ describes input buffer memory mapping sequential input data flow on 2^{r+1} lines, and describes output buffer memory implementing inverse data transformation on the output.

The inverse parallel-pipeline structure of computation of 2^l FFT with $m=2^r$ computing elements at a stage implementing m butterfly operators which have the numbers with *N/2m* is presented in the similar way:

$$FFT^{i}_{(l,r)} = \mu_{(r,l)}\mu_{(1)}BM_{(1)}B\ldots BM_{[l-r-1]}B\beta_{(1)}B\ldots$$
$$\ldots B\beta_{(r)}B\sigma^{-1}_{(r)}\mu^{-1}_{(l-r)}\mu^{-1}_{(r,l)},$$

where operator combination $\mu_{(r,l)}\mu_{(1)}$, $\sigma^{-1}_{(r)}\mu^{-1}_{(l-r)}\mu^{-1}_{(r,l)}$ describes input and output buffer memory accordingly.

The operator $\sigma_{(i)}$ permits to present the descriptions (8) and (9) of the parallel-pipeline structure of computation of 2^l FFT with $m=2^r$ computing elements at a stage implementing m sequential butterfly operations in another way:

direct PPFFT structure

$$FFT^{d}_{(l,r)} = \delta_{(r)}\mu_{(l-r)}BM_{(l-r-1)}B\ldots BM_{(1)}B\beta_{(r)}B\ldots$$
$$\ldots B\beta_{(1)}B\sigma_{(r)}\mu^{-1}_{(1)}\mu^{-1}_{(r,r)},$$

inverse PPFFT structure

$$FFT^{i}_{(l,r)} = \delta_{(r)}\mu_{(1)}\sigma^{-1}_{(r)}B\beta_{(1)}B\ldots B\beta_{(r)}BM_{(1)}B\ldots$$
$$\ldots BM_{(l-r-1)}B\mu^{-1}_{(l-r)}\mu^{-1}_{(r,r)},$$

where the operator combinations $\delta_{(r)}\mu_{(l-r)}$ and $\delta_{(r)}\mu_{(1)}\sigma^{-1}_{(r)}$ describe input buffer memory in the direct and inverse PPFFT structures correspondingly and $\sigma_{(r)}\mu^{-1}_{(1)}\mu^{-1}_{(r,r)}$ and $\mu^{-1}_{(l-r)}\mu^{-1}_{(r,r)}$ describe output buffer memory.

In such description of PPFFT structure with simultaneous execution of m sequential butterfly operations, in contrast to (8) and (9), on the input and output they use operators implementing inter reverse transformations permutations) as it is easy to show that $\delta_{(r)} = \mu_{(r,r)}$.

The developed by the authors version of language [8] of the structural description provides an one-to-one correspondence of a sequence of the operators to the processor structure in its description.

The structure of the parallel-pipeline FFT processor for computation of DFT vector process consisting of $D = 2^q$ components of the size $N = 2^l$ with $m = 2^r$ computing elements at a stage can be submitted as follows:

Proposition 5. ***Forward PPFFT-processor:*** *m of consecutive butterfly operations are simultaneously carried out*

$$FFT^{F}_{(q,l,r)1} = \begin{cases} \delta_{(r)}\mu_{(q+l-r)}BM_{(q+l-r-1)}B\ldots \\ \ldots BM_{(q-r+1)}B\mu^{-1}_{(q-r+1)}\mu^{-1}_{(r,r)}, r \le q; \\ \delta_{(r)}\mu_{(q+l-r)}BM_{(q+l-r-1)}B\ldots BM_{(1)}B\beta_{(r)}B\ldots \\ \ldots B\beta_{(q+1)}B\delta^{-1}_{(q)}\sigma_{(r)}\mu^{-1}_{(1)}\mu^{-1}_{(r,r)}, r > q; \end{cases}$$

Proposition 6. ***Inverse PPFFT-processor:*** *m of consecutive butterfly operations are simultaneously carried out*

$$FFT^{I}_{(q,l,r)1} = \begin{cases} \delta_{(r)}\mu_{(1)}\delta^{-1}_{(r)}B\beta_{(1)}B\ldots B\beta_{(r)}BM_{(1)}B\ldots \\ \ldots BM_{(l-r+1)}B\mu^{-1}_{(l-r)}\mu^{-1}_{(r,r)}, r \le l-1; \\ \delta_{(r)}\mu_{(1)}\delta^{-1}_{(r)}B\beta_{(1)}B\ldots \\ \ldots B\beta_{(l-1)}B\delta^{-1}_{(l-1)}\sigma_{(r)}\mu^{-1}_{(1)}\mu^{-1}_{(r,r)}, r > l-1; \end{cases}$$

The combinations of the operators $\delta_{(r)}\mu_{(q+l-r)}$, $\delta_{(r)}\mu_{(1)}\delta^{-1}_{(r)}$ describe input buffer memory in the appropriate structures; $\mu^{-1}_{(q-r+1)}\mu^{-1}_{(r,r)}$, $\delta^{-1}_{(q)}\sigma_{(r)}\mu^{-1}_{(1)}\mu^{-1}_{(r,r)}$, $\mu^{-1}_{(l-r)}\mu^{-1}_{(r,r)}$, $\delta^{-1}_{(l-1)}\sigma_{(r)}\mu^{-1}_{(1)}\mu^{-1}_{(r,r)}$ — output buffer memory; *B* - computing element (CE) (arithmetic device of butterfly operation); $M_{(i)}$ — intermediate memory $2x2^{i-1}$ of words volume; $\beta_{(i)}$ — direct (without intermediate memory) communication between two stages of the PPFFT-processor.

IV. Example

The structure of the PPFFT-processor for computation of inverse vector DFT for $D = 4$, $N = 8$, and m=2 is shown on a fig1. In parallel-pipeline processor of vector FFT with *m* CEs on every stage the CE calculate the butterfly operations which have the numbers with $DN/2m$ difference.

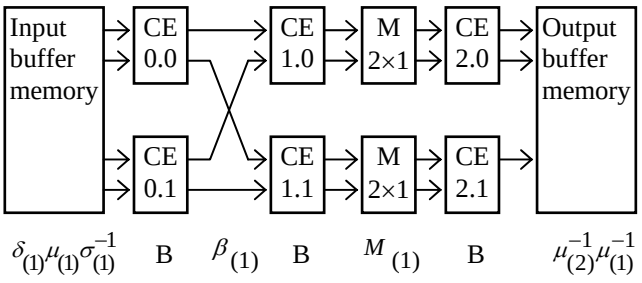

Fig. 1.Structures of the PPFFT-processor

If $D < m$ then the stages from 0 to $\log_2(DN/m)-1$ form *m* smaller independent pipeline FFTs of length DN/m and the stages from $\log_2(DN/m)-1$ to $\log_2 N-1$ are connected without intermediate memory; if $D \ge m$ then all the stages form *m* smaller independent pipeline FFTs of length DN/m and are connected using the intermediate memory. All computation processes of *R*-dimensional signal processing by PPFFT-processor $D = 4$, $N = 8$ and $m=2$ according fig1a are shown on the fig. 2.

The timing diagram (table 1) indicates the time and order in which the butterflies in each stage are computed. The dashes represent butterflies performed on data from the processing ($D = 4$, $N = 8$, m=2)-transform and the crosses represent butterflies performed on data in the following ($D = 4$, $N = 8$, m=2)-transform. The $B_i(n)$ -is *i*th butterflies of *n*th component of vector process, and T_B is the time of the butterfly operation execution (cycle duration).

V. PPFFT-processor structure synthesis algorithm for R-dimensional signal processing

The number of parallel working CEs on the stage can be various. The lower bound of *m* is 1 that corresponds to one-pipeline FFT processor. The upper bound of *m* is *DN*/2 that corresponds to matrix processor. However, it is more convenient to limit *m* by values equal to powers of 2 because in this case every CE calculates the same number of butterfly operations and the performance is uniformly distributed between the CEs.

The basic characteristics of the pipeline processor of vector FFT are following: the number of inputs, 2*m;* the number of stages, $\log_2 N$; the number of CEs on the stage, *m.;* the common number of CEs on the stage, $m\log_2 N$; the intermediate memory volume, *DN*-2*m* words; the delay, $\log_2 N + DN/2m - 1$ cycles; the time of the butterfly operation execution (cycle duration T_B), $2\Delta tm/D$.

The basic criteria at an vector processor structure synthesis estimation is maintenance of computation of vector DFT in real time with the minimum structural complexity. At development of the FFT-processor on this criteria on the basis of parallel - pipeline structure it is required to solve the following two basic problems of structural synthesis:

1) define optimum number of computing elements (CE) at a stage in the PPFFT-processor ensuring at given time of base operation execution computation of vector DFT in real time with the minimum structural complexity

$$m_{\min} = \begin{cases} 1, r = 0, if\ DT_B < 2\Delta t, \\ 2^r, r = \left]\log(DT_B/2\Delta t\right[, if\ DT_B \ge 2\Delta t, \end{cases}$$

where $]a[$ is nearest integer, greater than *a* or equal to it (if *a* is an integer); T_B is a time of butterfly operation execution of algorithm FFT; Δt is a sampling period; m is number of CE at a stage; *D* is number of components of vector process; *N* is size of a component;

2) define topology of structure of the PPFFT-processor for this number of CE at a stage.

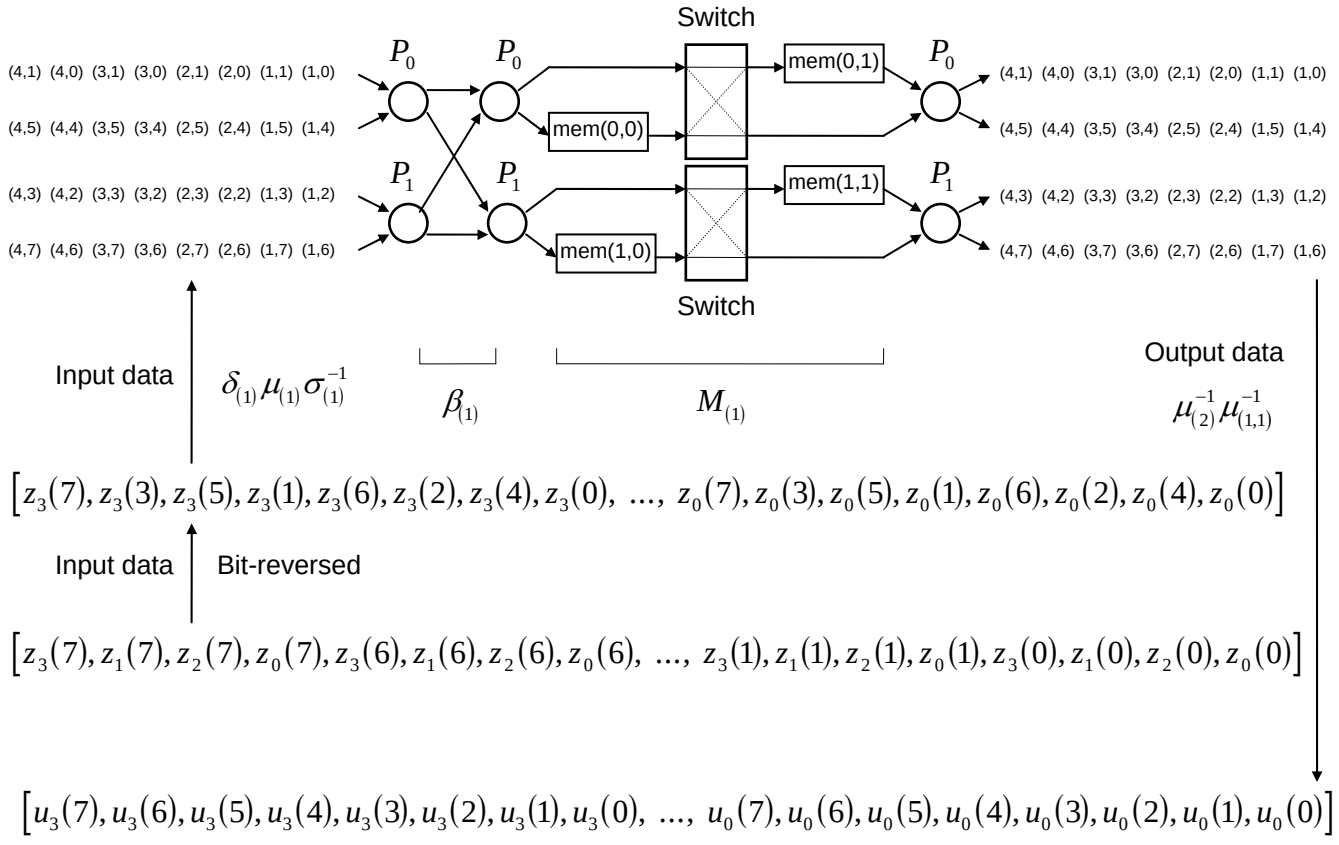

Fig. 2. The synthesis diagram of PPFFT-processor

TABLE 1
THE TIMING DIAGRAM INDICATES THE TIME AND ORDER IN WHICH THE BUTTERFLIES IN EACH STAGE ARE COMPUTED

Stage	Processor (CE)	T_B										
		1	2	3	4	5	6	7	8	9	10	11
0	P_0	$B_0(1)$	$B_2(1)$	$B_0(2)$	$B_2(2)$	$B_0(3)$	$B_2(3)$	$B_0(4)$	$B_2(4)$	×	×	×
	P_1	$B_1(1)$	$B_3(1)$	$B_1(2)$	$B_3(2)$	$B_1(3)$	$B_3(3)$	$B_1(4)$	$B_3(4)$	×	×	×
1	P_0	-	$B_0(1)$	$B_2(1)$	$B_0(2)$	$B_2(2)$	$B_0(3)$	$B_2(3)$	$B_0(4)$	$B_2(4)$	×	×
	P_1	-	$B_1(1)$	$B_3(1)$	$B_1(2)$	$B_3(2)$	$B_1(3)$	$B_3(3)$	$B_1(4)$	$B_3(4)$	×	×
2	P_0	-	-	-	$B_0(1)$	$B_2(1)$	$B_0(2)$	$B_2(2)$	$B_0(3)$	$B_2(3)$	$B_0(4)$	$B_2(4)$
	P_1	-	-	-	$B_1(1)$	$B_3(1)$	$B_1(2)$	$B_3(2)$	$B_1(3)$	$B_3(3)$	$B_1(4)$	$B_3(4)$

Algorithm.

INPUT: $D = 2^q, N = 2^l, \Delta t, T_B$ **;**

OUTPUT: $PPFFT_{(q,l,r)}, m = 2^r$ **:**

#1. To define $q = \log_2 D$.

#2. To define the number of stages in the PPFFT-processor $l = \log_2 N$.

#3. To define a parameter r and number of computing elements at a stage *m* ensuring computation of the vector DFT in real time at the minimum structural complexity of the processor.

#4. If $m \le DN/2$ or, that is the same, $r \le q + l - 1$ then pass to the step 5. Otherwise at given time of the butterfly operation execution T_B the PPFFT-processor with given parameters *q* and *l* cannot be constructed, and pass to the step 11.

#5. To define the common number of computing elements in the PPFFT-processor: *ml***.**

#6. If $m < N/2$ or, that is the same, $r < l - 1$ then pass to the step 7, if $m \ge N/2$ or $r \ge l - 1$, then pass to the step 9.

#7. To write down the description of structure $PPFFT_{(q,l,r)}, m = 2^r$ of PPFFT-processor.

#8. To define total volume of intermediate memory N-2m in words in the processor. Pass to the step 10.
#9. To write down the description of structure $PPFFT_{(q,l,r)}, m = 2^r$ of PPFFT-processor.
#10. To construct the structure of the PPFFT-processor on received expression.
#11. End.

VI. REALISATION

The most efficient realisation of the PPFFT processor is achieved by means of FPGA technology. FPGA unites the flexibility of programmable processors (DSP processors of TMS320 family) and the performance of dedicated hardware. The PPFFT processor structure includes two types of modules: a processing element in which the basic operation of FFT algorithm is realized and an intermediate memory which acts as a buffer between the different stages of the PFT algorithm. Every type of modules requires its reconfiguration depending on the synthesized structures. The reconfiguration of the processing element depends on transform length and the number of the processing elements in a stage since in this case the rotating multiplicand addressing is changed. The reconfiguration of the intermediate memory depends moreover on its position between the stages. Such modules can be realized by means of SRAM-based FPGAs. At this the modules are adjusted via the FPGA programming in accordance with expressions obtained on the stage of structural synthesis of the PPFFT processor.

Processing element. The processing element carries out the basic operation of the FFT algorithm to the base 2. The basic operation calculation includes such actions as a record of six real numbers (two complex values of input data and one rotating coefficient), an execution of four multiplication operations and six addition ones over these numbers and a reading of four result values. In general the processing element consists of a data exchange unit, arithmetic core immediately executing the basic operation and a coefficient memory (fig. 3).

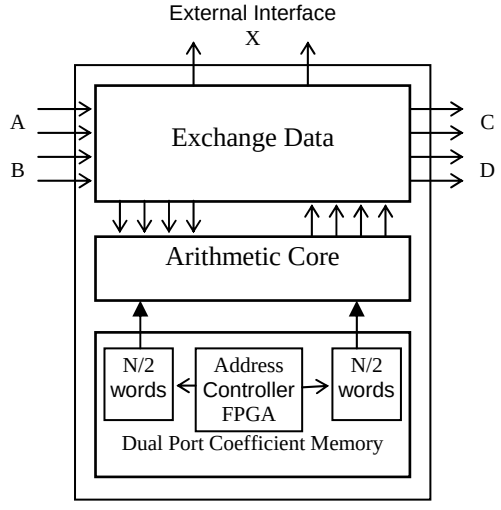

Fig. 3 . Processing element

The characteristic property of the processing element is a presence of three external buses. The input bus (A and B) and the output one (C and D) unite the processing elements as a parallel-pipeline structure. The external bus X is used for organization of the processor external trunk providing an obtaining of results of any FFT algorithm iteration. The data exchange unit provides a simultaneous work of all three buses that is the necessary condition for organization of the processor parallel-pipeline work.

The actions defined by the basic operation can be carried out under the various parallelism degree that leads to various structural constructions of the arithmetic core. The maximum performance can be achieved by use of pipeline structure with complete paralleled computation of the basic operation. At this the simple functional units such as adders can be realized by means of programmable logic. On the other hand, multiplication is a critical operation taking into account the performance and requires a lot of logical elements for its realization.

Therefore it has been found advantageous to carry out the multiplication by means of special-purpose high-functional modules like matrix multipliers. Fig.2 shows the structure of such an arithmetic unit with four multipliers and the programmable FPGA logic.

Processor interconnection element.

In general, some stages of the parallel-pipeline processor are connected through the intermediate memory which buffirizes a serial data flow on the input of the processing element with the object of its permutation and execution of essential delays according to the FFT algorithm. The processor interconnection elements unite the processing elements of the parallel-pipeline processor stages (fig. 3).

The processor interconnection element is based on the dual port memory including two random access memory blocks of N words volume. The intermediate memory provide simultaneous access of writing and reading to the every memory block. In addition to the intermediate memory the processor interconnection elements consist of data exchange (data path) unit and local control block. The data exchange unit provides the memory block switching in line with four working modes of the intermediate memory (fig. 4).

The local control block synchronizes the work of the memory blocks and addresses them. All the processor interconnection elements of the parallel-pipeline processor implement the same consequence of actions but they operate with different intermediate memory volume and switching rate of the working modes.

Parallel-pipeline processor.

Fig. 5 shows the structure of the parallel-pipeline processor based on the considered above elements for D=4, N=8 and m=2. The processor consists of three stages of the processing elements. Each stage has two elements. As it is not required the intermediate memory between the first and second stages the processing elements of these stages are connected immediately. Between the second and third stages it is necessary to use the intermediate memory of 2x1 words size. The processing elements of these stages are connected through the processor interconnection elements.

By comparison with the usual pipeline processor the FFT control of the intermediate memory in the parallel-pipeline processor is simpler as the number of stages requiring the intermediate memory is less. At this the intermediate memories are parallel and demands only one set of control signals.

Except the intermediate memory on the input and output of vector DFT processor it is necessary to use buffer memory which maps the input data flow onto rn processor inputs and makes a reverse mapping on the output.

The complete structure of the vector DFT processor is shown on fig. 6. The processor memory unites two banks of the input buffer and two banks of the output one. One bank of the input memory sends the data to the reading bus for processing in the parallel-pipeline processor. The other bank receives the new input data over the system bus. Simultaneously, one bank of the output memory sends the result of previous transform to the system bus for the farther processing and the other one receives the result of current transform from the parallel-pipeline processor over the writing bus. Such a memory design as well as bus structure provides a continual data receipt and permits execution of input-output operations simultaneously with processing.

The considered above analysis and principles of design of the parallel-pipeline vector DFT processor can be used by FFT processor building both as LSIs and VLSIs.

CONCLUDING REMARKS

In some signal processing applications, it is desirable to build very high performance FFT-processors. To meet the performance requirements, these processors are typically highly pipelined. A notation is then presented which allows parallel-pipeline versions of FFT-processors to be developed for multidimensional signal processing, in particular, for R-dimensional signal. These versions are well suited for use in VLSI implementation. An example, a hardware solution for this purpose has been put into practice by means of SRAM-based field programmable gate arrays (FPGA) (using distributed arithmetic).

REFERENCES

[1] A.A.Petrovsky. Architecture of real-time programmable digital system and matrix algorithms for control of R-dimensional space random vibration / 3rd IFAC/IFIP workshop on “Algorithms and Architecure for Real-Time Control” AARTC’95, Ostend, Belgium, pp. 425-430, 1995.

[2] A.A.Petrovsky, M.V.Kachinsky. Structural synthesis of parallel-pipeline FFT-processors for R-dimensional signal processing/ 4th Intern. Workshop on Systems, Signals and image processing, Poznan, Poland, pp.215-218, 1997.

[3] D.G.Korn, J.J.Lambiotte, Computing the fast Fourier transform on a vector computer./ Mathematics of computation, vol. 33, no. 147, pp.977-992, 1979

[4] J.A.Johnston. Parallel pipeline fast Fourier transfomer, IEE Proceedings, Vol.130,Part F, No 6, pp.564-572, 1983.

[5] S.F.Gorman, M.Wills, Partial column FFT pipeline./ The Trans. Circuits and Systems-II: analog and digital proces., vol. 42, no. 6, pp. 414-423, 1995

[6] N.Koziris, G.Economakos, T.Audronikos and et al. Optimal automatic hardware synthesis for signal processing algorithms / Int. conf. “Digital Sygnal Processing-97” DSP97, Greece, pp.1011-1014.

[7] D.S. Parker, Notes on shuffle/exchange-type switching networks./ IEEE The Trans .Comput., vol. C-29, no. 3, pp. 213-222, Mar. 1980.

[8] E.H. Wold, A.M. Despain, Pipeline and parallel-pipeline FFT processors for VLSI implementations./ IEEE The Trans .Comput., vol. C-33, no. 5, pp. 414-426, May. 1984.