https://doi.org/10.31891/csit-2025-3-9
UDC 004.383.4 + 004.896

Ivan TSMOTS
Lviv Polytechnic National University
Oleh BEREZSKY
West Ukrainian National University
Taras MAMCHUR
Lviv Polytechnic National University

SYNTHESIS OF RECURSIVE DEVICES FOR VERTICAL-GROUP CALCULATION OF BASIC MULTI-OPERAND NEUROOPERATIONS

The operational basis of artificial neural networks has been determined, comprising groups of the following neurooperations: preprocessing, processing, and computation of transfer functions. A set of basic multi-operand neurooperations was selected for hardware implementation, including: finding maximum and minimum values in a one-dimensional data array, calculation of the sum of squared differences, scalar product calculation. The methods of vertical-group computation of basic multi-operand neurooperations (finding for maximum and minimum values in a one-dimensional array, calculation of the sum of squared differences, and scalar product calculation) have been improved. Using the selection of number of bits for operands group for single-cycle processing, these methods enable synchronization of data arrival time with calculation time and ensure high hardware utilization efficiency during the hardware implementation. It's proposed a recursive devices design for vertical-group computation of basic multi-operand neurooperations based on an integrated approach. This approach leverages the capabilities of modern element base, incorporates vertical methods, algorithms, and recursive device structures for implementing basic neurooperations and considers the requirements of specific applications. The principles for designing recursive devices for vertical-group calculation of basic multi-operand neurooperations have been chosen. These include: the use of a basis of elementary arithmetic operations and a multi-operand approach; modularity; pipelining and spatial parallelism; homogeneity and regularity of the structure; synchronization between data arrival time and neurooperation calculation time; specialization and adaptation of structure to specific application requirements. A format converter has been developed to transform a flow of serial input data from a one-dimensional array into a parallel-serial data output by group of bits. Basic structures have been developed. They represent calculation algorithms in terms of hardware and serve as the foundation for synthesizing of recursive devices for vertical-group calculation of basic multi-operand neurooperations with specified parameters. The method for synthesis of recursive devices for calculation of basic multi-operand neurooperations with vertical-group data processing has been improved. Through the use of mechanisms for synchronizing calculation time with data arrival time, this method provides the selection of structure which performs real-time data processing and with high hardware utilization efficiency. It has been demonstrated that the use of the improved vertical-group methods, designed basic structures of devices for finding maximum and minimum numbers in one-dimensional arrays, calculation of the sum of squared differences and scalar product, as well as the improved synthesis method, enables real-time mode and the implementation of devices for calculation of basic multi-operand neurooperations with vertical-group data processing with high hardware utilization efficiency.

Keywords: multi-operand neurooperations, recursive devices, vertical-group data processing, real-time operation, hardware utilization efficiency.

Іван ЦМОЦЬ
Національний університет «Львівська політехніка»
Олег БЕРЕЗЬКИЙ
Західноукраїнський національний університет
Тарас МАМЧУР
Національний університет «Львівська політехніка»

СИНТЕЗ РЕКУРСИВНИХ ПРИСТРОЇВ ВЕРТИКАЛЬНО-ГРУПОВОГО ОБЧИСЛЕННЯ БАЗОВИХ БАГАТООПЕРАНДНИХ НЕЙРООПЕРАЦІЙ

Виділено операційний базис штучних нейронних мереж, який складається із груп таких нейрооперацій: попередньої обробки, процесорних та обчислення передатних функцій. Вибрано для апаратної реалізації базові багатооперандні нейрооперації: пошуку максимальних і мінімальних значень у одновимірному масиві даних, обчислення суми квадратів різниць та обчислення скалярного добутку. Вдосконалено методи вертикально-групового обчислення базових багатооперандних нейрооперацій (пошуку максимальних і мінімальних значень у одновимірному масиві даних, обчислення суми квадратів різниць, обчислення скалярного добутку), які шляхом вибору кількості розрядів у зрізі операндів для опрацювання в одному такті забезпечують узгодження часу надходження даних з часом обчислення та високу ефективність використання обладнання при їх апаратній реалізації. Запропоновано розробку рекурсивних пристроїв вертикально-групового обчислення базових багатооперандних нейрооперацій виконувати на основі інтегрованого підходу, який ґрунтується на можливостях сучасної елементної бази, охоплює вертикальні методи, алгоритми та рекурсивні структури пристроїв для реалізації базових нейрооперацій і враховує вимоги конкретних застосувань. Вибрано принципи розробки рекурсивних пристроїв вертикально-групового обчислення базових багатооперандних нейрооперацій, основними з яких є: використання базису елементарних арифметичних операцій та багатооперандного підходу; модульності; конвеєризації та просторового паралелізму; однорідності та регулярності структури; узгодження часу надходження даних з часом обчислення нейрооперації; спеціалізації та адаптації структури пристрою до вимог конкретного застосування. Розроблено перетворювач форматів, який перетворює потік послівних вхідних даних одновимірного масиву у паралельно-послідовну видачу даних групами розрядів. Розроблено базові структури, які апаратно відображають алгоритми обчислень і є основою для синтезу рекурсивних пристроїв вертикально-групового обчислення базових багатооперандних нейрооперацій

із заданими параметрами. Вдосконалено метод синтезу рекурсивних пристроїв обчислення базових багатоопераднних нейроопераций з вертикально-груповою обробкою даних, який за рахунок використання механізмів узгодження часу обчислення з часом надходження даних забезпечує вибір структури, що виконує обробку даних у реальному часі та має високу ефективність використання обладнання. Показано, що використання вдосконалених вертикально-групових методів, розроблених базових структур пристроїв пошуку максимальних і мінімальних чисел у одновимірних масивах, обчислення сум квадратів різниць і скалярного добутку та вдосконаленого методу синтезу дає змогу забезпечити режим реального часу та реалізацію пристроїв обчислення базових багатоопераднних нейроопераций з вертикально-груповою обробкою даних з високою ефективністю використання обладнання.

Ключові слова: багатоопераднні нейрооперації, рекурсивні пристроїв, вертикально-групова обробка даних, реальний час, ефективність використання обладнання.

Received / Стаття надійшла до редакції 01.09.2025
Accepted / Прийнята до друку 22.09.2025

Introduction

When using real-time neural network technologies in industry (management of technological processes and complex objects), energy (optimization of load in power grids), military affairs (technical vision, mobile robot motion control, cryptographic data protection), transport (traffic and engine control), medicine (disease diagnosis), and instrument engineering (image recognition and control optimization), it is necessary to process intensive data flows using means that meet restrictions on size, weight, and energy consumption and have high equipment utilization efficiency [1]. To ensure a wide range of applications, it is necessary to identify basic neurooperations and develop devices that can be easily adapted to the requirements of specific tasks and can be used to synthesize a wide range of real-time hardware neural networks [2].

An analysis of the operational basis of artificial neural networks (ANNs) shows [3] that neural network operations depending on the number of operands processed simultaneously can be divided into single-operand (square root, transfer functions), two-operand (addition, division, multiplication), and multi-operand (determination of minimum and maximum numbers, group summation, scalar product calculation, calculation of the sum of squared differences). The speed of ANN is mainly determined by the execution time of multi-operand basic neurooperations. A distinctive feature of such neurooperations is that they are performed on a set of operands, and the result of the operation is a single number. Hardware neural networks which widely use devices for implementing multi-operand basic neural operations are used to process intensive flows in real time [4].

The synthesis of highly efficient devices for real-time calculation of multi-operand basic neurooperations requires the widespread use of modern element base (half-specific and half-specific very-large-scale integration (VLSI) circuits, single-chip neuroprocessors, system-on-chip (SoC) architectures, microcomputers, and microcontrollers), the development of new methods and VLSI structures. The orientation of devices for the calculation of basic multi-operand neurooperations on VLSI implementation with high hardware utilization efficiency requires a reduction in the number of interface outputs, inter-neuron connections, and synchronization of data arrival time with processing time. The main ways of such synchronization are the selection of the clock frequency, the number and bit width of data processing cycles. These requirements can be met by using parallel vertical-group data processing methods and recursive structures that adapt to the requirements of a specific application [5].

It is most efficient to develop recursive structures for vertical-group calculation of basic multi-operand neurooperations on the basis of integrated approach that covers the modern element base, methods, algorithms, structures, and takes into account the time of data arrival and the requirements of specific applications.

Therefore, the problem of developing methods and recursive structures with high hardware utilization efficiency for vertical-group calculation of basic multi-operand neurooperations becomes especially relevant.

The object of the research is the processes of synchronized data arrival time with vertical-group data processing time and hardware cost minimization, which provide the development of recursive devices for real-time calculation of basic multi-operand neurooperations with high hardware utilization efficiency.

The subject of the research is vertical-group methods, recursive structures of devices for real-time calculation of basic multi-operand neurooperations with high hardware utilization efficiency.

The aim of the work is to develop methods and recursive structures of devices for vertical-group calculation of basic multi-operand neurooperations in real time with high hardware utilization efficiency.

To achieve this goal, the following main ***tasks of the study are defined:***

determine the operational basis of the ANN and select the most complex basic multi-operand neurooperations, the hardware implementation of which will provide real-time neural network processing of intensive data flows;

improve methods of vertical-group calculation of basic multi-operand neurooperations and focus them on implementation based on recursive structures;

develop basic structures of recursive devices for vertical-group calculation of basic multi-operand neurooperations, which are the basis for the synthesis of real-time devices with high hardware utilization efficiency;

develop a method for synthesizing recursive devices for real-time calculation of basic multi-operand neurooperations, which uses the developed basic structures and provides synthesis of real-time devices based on them, which meet the requirements of a specific application and have high hardware utilization efficiency.

Analysis of the latest research and publications

An analysis of neural network implementation methods shows [6-8] that for processing intensive data flows in real time, it is reasonable to use hardware implementation of neural networks with pipelining and spatial parallelism.

Neural networks are synthesized on the basis of hardware devices which implement the most complex operations and are focused on synchronization of calculation time and input data arrival time. In papers [9-11], methods of calculations and structures of devices for parallel-vertical calculation of basic neurooperations are reviewed. The disadvantage of the reviewed methods and structures is low performance, which largely depends on the bit depth of operands that are simultaneously processed during the implementation of algorithms.

The characteristics of ANNs largely depend on the hardware implementation options for basic multi-operand neurooperations such as finding maximum and minimum numbers in arrays, calculation of sums of squared differences and scalar products [12-14]. The analysis of structures for the implementation of basic multi-operand neurooperations [15,16] showed that two types of structures are used for hardware implementation: recursive and non-recursive. A structural feature of recursive devices is the presence of inverse connections. In such devices, the calculation of operations is performed in several iterations, the number of which mainly depends on the bit depth of the operands being analyzed. The disadvantage of recursive devices for the implementation of basic multi-operand neurooperations is their relatively low speed [17]. Non-recursive devices do not have inverse connections, they have higher speed, and their implementation requires high hardware costs [18,19]. The disadvantage of non-recursive devices that implement basic multi-operand neurooperations is the large number of interface outputs and high hardware costs [20].

The papers [20, 21] consider two approaches to the hardware implementation of basic multi-operand neurooperations (calculation of the sum of squared differences and scalar product), the first of which is based on multiplication and addition operations, and the second on elementary arithmetic operations of addition, inversion, and shift. However, these approaches leave unresolved the problem of optimizing device structures and calculating their time parameters and hardware costs.

Papers [22, 23] examine a multi-operand approach, in which the calculation of basic neurooperations is considered as the execution of a single operation based on elementary arithmetic operations. The use of this approach to implement basic multi-operand neurooperations will provide optimization of calculation time.

The analysis of the above publications shows that the hardware implementation of basic multi-operand neurooperations for searching for maximum (minimum) numbers in arrays, calculating sums of squared differences and scalar products requires the development of new vertical methods, algorithms, and non-recursive structures focused on processing continuous data flows in real time with high hardware utilization efficiency.

Research results and their discussion

1. Determining the operational basis of neural networks

The current stage of development of real-time neural network technologies is characterized by the widespread use of these technologies for the implementation of intelligent components for data processing, obstacle recognition, control of mobile robotic platforms, and data protection. The main requirements for such components are real-time operation and providing high hardware utilization efficiency. Achievements in NIS technologies make it possible to increasingly transfer the implementation of neural algorithms to hardware devices that distribute the computing process both in time and space. The structural organization of such hardware is based on the principle of adequate hardware representation of neural algorithm graphs. The hardware is characterized by high performance along with the complexity of modification and change of working algorithms.

Based on the analysis of neural network algorithms [11], an operational basis was determined, which is shown in Fig. 1.

The operational basis of neural networks consists of three groups of operations:
the first – pre-processing neuro-operations;
the second – processing neuro-operations;
the third – activation functions.

The first group of neural operations provides the conversion of input data to a form that will return the best results. The learning vector contains one value for each neural network input and one value for each neural network output, depending on the type of training (supervised or unsupervised). Training the network on a “raw” set usually does not give good results. To improve the quality of the neural network usage, the input data is pre-processed, which refers to the following operations: normalization, quantization, and filtering.

Normalization is a procedure for pre-processing input data (training, testing, and working samples), in which the values of the features that form the input vector are reduced to a certain specified range. After normalization, all values of the input features will be reduced to some narrow range [0, 1] or [–1, 1].

Normalization of input data to the range [0, 1] is performed as follows:

$$x_i^{\times} = \frac{x_i - x_{min}}{x_{max} - x_{min}}, \quad (1)$$

where x_i is the input data, x_{max} is the maximum value of the input data, x_{min} and is the minimum value of the input data.

Normalization of input data to the range [-1, 1] is as follows:

$$x_i^{\times} = \frac{x_i}{|x|_{max}}, \quad (2)$$

These kinds of normalization do not require complex calculations and are widely used for *xi* inputs that tightly fill a certain gap.

After normalizing the input data in the RBF and GRNN networks, the Euclidean distance from each input vector to all the others must be calculated. This calculation of the Euclidean distance is performed using the neurooperation:

$$y = ||x_i^e - x_i^b||^2 = (x_1^e - x_1^b)^2 + (x_2^e - x_2^b)^2 + \cdots + (x_N^e - x_N^b)^2, \quad (3)$$

For other types of neural networks, filtering can be used, which is performed on noisy input data and is reduced to discarding values that are invalid. In addition, quantization is performed on continuous quantities, which involves the determination of a finite set of discrete values.

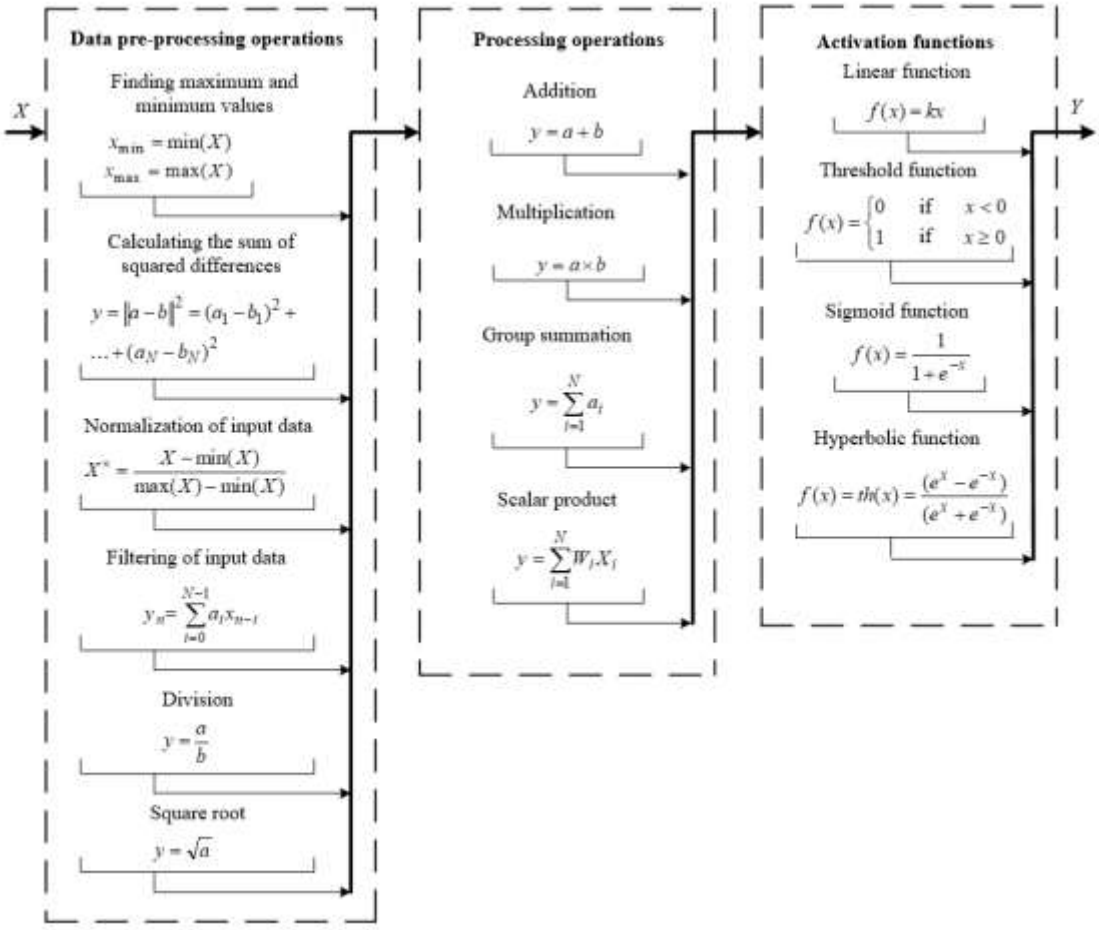

Fig. 1. The operational basis of ANN

Processor operations on input data and weighting coefficients are performed directly in the neural network itself during training and functioning and are reduced to the calculation of a weighted sum. When processing data in the neural network itself, the following operations can be used: addition, multiplication, group summation, and scalar product calculation.

The weighted sum value is converted into an output signal through an algorithmic process known as an activation function or transfer function. Neural networks can use different activation functions, which are selected depending on the tasks being solved and the type of neural network. The most commonly used activation functions in neural networks are linear, threshold, sigmoid, and hyperbolic.

From the analysis of the operational basis of neural networks (Fig. 1), it can be seen that the performance of hardware neural networks depends most on the following operations: finding the maximum and minimum values in a one-dimensional data array, calculation of the sum of squared differences and scalar product calculation. A distinctive feature of these neural operations is that they are multi-operand and are performed on a set of operands. The result of a multi-operand neural operation is a single number. It is proposed to perform multi-operand neuro-

operations based on a multi-operand approach, in which the process of calculating a neuro-operation is considered as the execution of a single operation based on elementary arithmetic operations.

2. Improvement of methods for vertical-group calculation of basic multi-operand neurooperations

Vertical-group calculation of basic multi-operand neuro-operations assumes that data arrives in parallel by groups of bit slices (vertically). The vertical-group method of processing data arrays assumes that the weighting coefficients W_j and input data X_j are received in parallel by slices of k bits according to the following formulas:

$$W_j = \sum_{i=1}^{n} 2^{-(i-1)} w_{ji} = \sum_{g=1}^{m} 2^{-(g-1)k} (w_{j[(g-1)k+1]} + 2^{-1} w_{j[(g-1)k+2]} + \cdots + 2^{-(k-1)} w_{j[(g-1)k+k]}), \quad (4)$$

$$X_j = \sum_{i=1}^{n} 2^{-(i-1)} x_{ji} = \sum_{g=1}^{m} 2^{-(g-1)k} (x_{j[(g-1)k+1]} + 2^{-1} x_{j[(g-1)k+2]} + \ldots + 2^{-(k-1)} x_{j[(g-1)k+k]}), \quad (5)$$

where w_{ji}, x_{ji} are the values of the *i-th* bits of the weighting coefficients and input data; n is the bit depth of the weighting coefficients and input data; m is the number of bit groups $m = \left\lceil \frac{n}{k} \right\rceil$ into which the weighting coefficients W_j and input data X_j are divided; k is the number of bits in a group.

2.1. Method of vertical-group finding for maximum and minimum values in a one-dimensional data array

The vertical-group method for finding the maximum X_{max} and minimum X_{min} values in a one-dimensional $\{X_k\}_{h=1}^{N}$ array assumes that in each *g-th* cycle (*g=1*, …, *m* where $m = \left\lceil \frac{n}{k} \right\rceil$, *k* is the number of bits in the group, *n* is the bit depth) has the parallel input of *N* numbers by higher digits first, by slices of *k* bits [9]. Finding the maximum X_{max} and minimum X_{min} numbers in a one-dimensional array $\{X_k\}_{h=1}^{N}$ using this method is based on performing the same basic macro-operations for each *r-th* bit slice (*r=1, …, k*), which are based on three simple operations.

To find the maximum number X_{max} in a one-dimensional array $\{X_k\}_{h=1}^{N}$, the following operations are used:

1) Creation of the value of the *r-th* bit slice P_r using the formula:

$$P_r = \bigvee_{h=1}^{N} X_{rh} \wedge y_{rh}, \quad (6)$$

where X_{rh} is the value of the *r-th* bit of the *h-th* number in the array, y_{rh} is the value of the *h-th* bit of the *r-th* control word, the value of the 1-st control word is equal to $y_{11}=y_{12}=\ldots=y_{1N}=1$;

2) Determination of the *r-th* bit of the maximum number X_{maxr} using the expression:

$$Xr \begin{cases} 0, & if \quad P_r = 0 \\ 1, & if \quad P_r = 1_{max} \end{cases}, \quad (7)$$

3) Creation of *h* bits of *(r+1)-th* control word using the formula:

$$y_{(r+1)h} = \begin{cases} 0, & if \quad P_r = 1, \quad X_{hr} \neq y_{hr} \\ 1, & if \quad P_r = X_{hr} = y_{hr} = 1 \\ y_{hr}, & if \quad P_r = 0 \end{cases}, \quad (8)$$

To find the minimum number X_{min} in a one-dimensional array $\{X_k\}_{h=1}^{N}$, the following operations are used:

1) Creation of the value of the *r-th* bit slice P_r, which is performed using the formula:

$$P_r = \bigvee_{h=1}^{N} \bar{X}_{rh} \wedge y_{rh}, \quad (9)$$

where $\bar{X}_{rh}$ is the inverse value of the *r-th* bit of the *h-th* number of the array, y_{rh} is the value of the *h-th* bit of the *r-th* control word, the value of the 1-st control word is equal to $y_{11}=y_{12}=\ldots=y_{1N}=1$;

2) Determination of the *r-th* bit of the minimum number X_{minr} according to the formula:

$$X_{\min r} \begin{cases} 0, & if \quad P_r = 1 \\ 1, & if \quad P_r = 0 \end{cases}, \quad (10)$$

3) Creation of *h* bits of the *(r+1)-th* control word, which is performed according to the formula:

$$y_{(r+1)h} = \begin{cases} 0, & if \quad P_r = 1, \quad \bar{X}_{khr} \neq y_{rh} \\ 1, & if \quad P_r = \overline{X}_{hr} = y_{hr} = 1 \\ y_{hr}, & if \quad P_r = 0 \end{cases}, \quad (11)$$

A key feature of mentioned parallel vertical-group method for finding the maximum (minimum) number is that in each *g-th* clock cycle, *k* bits of the maximum X_{max} (minimum X_{min}) number are determined.

The distinctive features of the vertical-group search for maximum and minimum numbers in an array are next:

use of a single basic macro-operation;

the possibility of using parallelization and pipelining of calculations;

the possibility of simultaneous processing of N-bit slices;

the calculation time is mainly determined by both the number of bits in group k and the bit depth of numbers n, rather than their amount N.

2.2. Method of vertical-group calculation of the sum of squared differences

The vertical-group calculation method of the sum of squared differences requires that each operand be represented as groups of *k* bits. In this representation, the operands are written as follows:

$$X_j = \sum_{i=1}^{n} 2^{-(i-1)} x_{ji} = \sum_{g=1}^{m} 2^{-(g-1)k} (x_{j[(g-1)k+1]} + 2^{-1} x_{j[(g-1)k+2]} + \cdots + 2^{-(k-1)} x_{j[(g-1)k+k]}), \quad (12)$$

where x_{ji} is the value of the *i-th* digit of the j-*th* operand; *n* is the operand's bit depth, $m = \left\lceil \frac{n}{k} \right\rceil$ is the number of groups into which the operand is broken down.

Squaring is the main operation in the calculation of the sum of squared differences. To perform this operation, the vertical algorithm is used:

$$X^2 = (0.01) \wedge x_1 + 2^{-1}(0.x_1 01) \wedge x_2 + 2^{-2}(0.x_1 x_2 01) \wedge x_3 + \cdots + 2^{-(n-1)}(0.x_1 x_2 \ldots x_{n-1} 01) \wedge x_n = \sum_{i=1}^{n} 2^{-(i-1)} Q_i, \quad (13)$$

where Q_i is the partial result of squaring, which is determined as follows:

$$Q_i = (0.x_1 x_2 \ldots x_{i-1} 01) \wedge x_i, \quad (14)$$

The evolution of the above algorithm is the creation of a Q_{Gg} group partial result of squaring for a group of *k* bits:

$$Q_{Mg} = Q_{g1} + 2^{-1} Q_{g2} + \cdots + 2^{-(k-1)} Q_{gk} = \sum_{r=1}^{k} 2^{-(r-1)} Q_{gr}, \quad (15)$$

where Q_{gr} is the partial result of squaring.

The algorithm for squaring with the use of group partial result creation Q_{Gg} is written as follows:

$$X^2 = \sum_{g=1}^{m} 2^{-(g-1)k} Q_{Gg}, \quad (16)$$

The calculation of *N* sums of squared differences will be performed based on a multi-operand approach, which consists of simultaneously processing all operands and creating group partial results of the sum of squared differences for them. The calculation of *N* sums of squared differences will be performed using a parallel vertical-group method, which is written as follows:

$$Y = (X_1^e - X_1^b)^2 + (X_2^e - X_2^b)^2 + \cdots + (X_N^e - X_N^b)^2 = \Delta X_1^2 + \Delta X_2^2 + \cdots + \Delta X_N^2 =$$

$$= \sum_{g=1}^{m} 2^{-(g-1)k} Q_{1Gg} + \cdots + \sum_{g=1}^{m} 2^{-(g-1)k} Q_{NGg} = \sum_{j=1}^{N} \sum_{g=1}^{m} 2^{(g-1)k} Q_{jGg} = \sum_{g=1}^{m} 2^{(g-1)k} \sum_{j=1}^{N} Q_{jGg} = \sum_{g=1}^{m} 2^{(g-1)k} Q_{Mg}, \quad (17)$$

where Q_{Mg} is the *g-th* micro-partial result of the sum of squared differences.

The main steps of the vertical-group method for calculating the sum of squared differences are:

simultaneous sequential-group arrival of operands X_j^e, X_j^b and calculation of N of the difference modules ΔX_j;

creation for each j-th module ΔX_j in the g-th cycle k of partial results of squaring Qn-(kg-1),..,Qn-k(g-1);

summation $N \times k$ of partial results of squaring;

creation of a macro-partial result of the sum of squared differences QMg by summing N(k) partial results of squaring;

obtaining the result of the sum of squared differences by summing the macro-partial results of squaring QMg with a right shift on k bits.

2.3. Vertical-group method for the scalar product calculating

The method of vertical-group calculation of scalar product is implemented on the basis of elementary arithmetic operations and is oriented on VLSI implementation. The use of this method provides a reduction in the number of clock cycles and, consequently, in the calculation time. The scalar product calculation using this method comes down to creating and summing of partial products according to the following formula:

$$Z = \sum_{j=1}^{N} W_j X_j = \sum_{j=1}^{N} \sum_{g=1}^{m} 2^{-(g-1)k} \left(W_j X_{j[(g-1)k+1} + 2^{r-1} W_j X_{j[(g-1)k+r} + \cdots + 2^{-(k-1)} W_j X_{j[(g-1)k+k} \right), \quad (18)$$

where *r=1, …, k*.

After making the necessary changes to formula (18), the scalar product calculation can be written as follows:

$$Z = \sum_{g=1}^{m} 2^{-(g-1)k} \sum_{j=1}^{N} \left(W_j X_{j[(g-1)k+1} + 2^{r-1} W_j X_{j[(g-1)k+r} + \cdots + 2^{-(k-1)} W_j X_{j[(g-1)k+k} \right) = \sum_{g=1}^{m} 2^{-(g-1)k} P_{\text{gM}}, \quad (19)$$

where P_{gM} is the *g-th* macro-partial result of the scalar product calculation.

From formula (19) it follows that the scalar product calculation is performed in *m* cycles, in each *g-th* cycle the following operations are performed:

creation for each j-th pair of operands k partial products in accordance with the formula $P_{j[(g-1)k+r]} = W_j X_{j[(g-1)k+r]}$;
calculation of the g-th macro-partial result of the scalar product calculation PgM by summing $N \times k$ partial products in accordance with the formula $P_{\text{gM}} = \sum_{j=1}^{N}(W_j X_{j[(g-1)k+1} + 2^{r-1} W_j X_{j[(g-1)k+r} + \cdots + 2^{-(k-1)} W_j X_{j[(g-1)k+k})$;
adding the g-th macro-partial result of the scalar product calculation PgM to the sum result, which is shifted to the right by k bits, in accordance with the expression $Z_g = 2^{-k} Z_{g-1} + P_{gM}$, where Z0=0.

3. Development of basic structures of recursive devices for vertical-group calculation of basic multi-operand neuro-operations

3.1. Design principles of recursive devices for vertical-group calculation of basic multi-operand neuro-operations

It is proposed to develop recursive devices for vertical-group calculation of basic multi-operand neuro-operations based on an integrated approach, which is based on the capabilities of modern element base, covers vertical methods, algorithms, and recursive device structures for implementing basic neuro-operations, and takes into account the requirements of specific applications. To make full use of the advantages of modern element base, it is proposed to develop recursive devices for vertical-group calculation of basic multi-operand neuro-operations according to the following principles:
use of the basis of elementary arithmetic operations for the implementation of basic multi-operand neuro-operations;
use of a multi-operand approach for the implementation of devices for the calculation of basic multi-operand neuro-operations;
modularity, which involves the development of devices for implementing basic multi-operand neural operations in the form of functionally complete modules;
localization and reduction of the number of connections between device elements;
pipelining and spatial parallelism in the development of device structures for the implementation of basic multi-operand neuro-operations;
homogeneity and regularity of structure;
synchronization of data arrival time with the calculation time of the basic multi-operand neuro-operation;
specialization and adaptation of the device structure to the structure of the calculation algorithm of the basic multi-operand neuro-operation.

3.2. Development of a data format converter

The format converter must provide conversion of a serial input data flow of a one-dimensional array into parallel-serial data output in groups of *k* bits. The format conversion is performed in two stages. At the first stage, *N* numbers of the array are received sequentially, and at the second stage, parallel-serial conversion is performed at each *g* clock cycle of the converter, and k digits are obtained at its *j-th* output. The structure of the data format converter is shown in Fig. 2, where *Rg* is a register, *BRg* is a buffer register, *Cm* is a commutator, *Cnt* is a counter, X_j is a data input, TI_1, TI_2 are clock pulses, respectively, the first and second, and *WrX* is a data write signal.

The format converter consists of groups of registers *Rg1 - RgN*, buffer registers *BRg1 - BRgN*, and commutators *Cm1 - CmN*. Registers *Rg1 - RgN* are serial connected to each other and provide sequential recording of a one-dimensional array with *N* input data. The array of *N* input data is recorded using clock pulses TI_1. Input data $X_1,\ldots,X_N$ from outputs *Rg1 - RgN* are recorded in buffer registers *BRg1 - BRgN* by the signal *WrX*. Data from the outputs of buffer registers *BRg1 - BRgN* are fed to the inputs of switches *Cm1 - CmN*, at the outputs of which we obtain a slice of k bits in each TI_2 clock cycle. The conversion of data in parallel code to serial-group code is performed in *m* clock cycles equal to the TI_2 period.

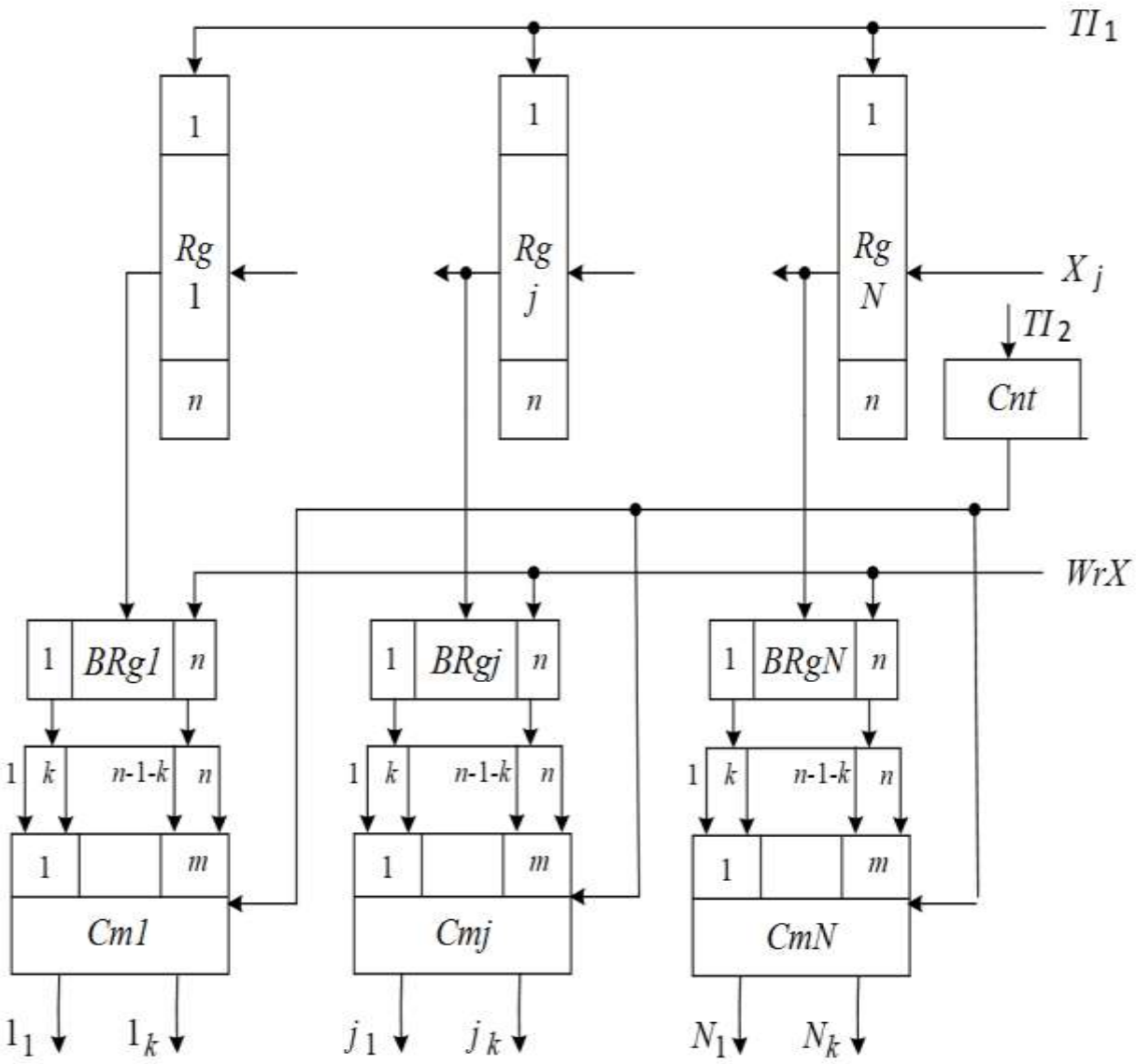

Fig. 2. The structure of the data format converter

3.3. Structure of a recursive device for finding maximum and minimum values

A recursive device for finding maximum and minimum values is implemented on the basis of *N* identical processor elements (PE). Each PE implements on hardware level *k* basic macro-operations for the calculation of maximum and minimum numbers.

The structure of the recursive device for searching for maximum and minimum numbers in a one-dimensional array $\{X_k\}_{h=1}^{N}$ using the parallel vertical-group method is shown in Fig. 3, where *TI* are clock pulses, *IS* is the initial setting, *Tg* is a trigger, Rg is a register, $X_{h1} - X_{hk}$ is a *h-th* input of a group of *k* bits, $X_{h1min} - X_{hkmin}$ and $X_{h1max} - X_{hkmax}$ – outputs of groups of k bits of the minimum and maximum numbers, respectively.

The number of PEs connected to the common results bus, when simultaneously finding the maximum and minimum numbers for a one-dimensional array $\{X_k\}_{h=1}^{N}$, is determined by its size. The use of common results buses provides parallelization of the processing of the bit slice, the processing time of which determines the clock frequency of the device. The finding for maximum and minimum numbers using the parallel vertical-group method in such a device is performed in a time determined as follows:

$$t_{FMMN} = \left\lceil \frac{n}{k} \right\rceil \left(t_{Rg} + k3t_{logI}\right), \tag{20}$$

where t_{Rg} and t_{logI} are the response times of the register and logical elements of the *OR*, *AND*, *AND-NOT* types, respectively, and *k* is the number of bits in the group.

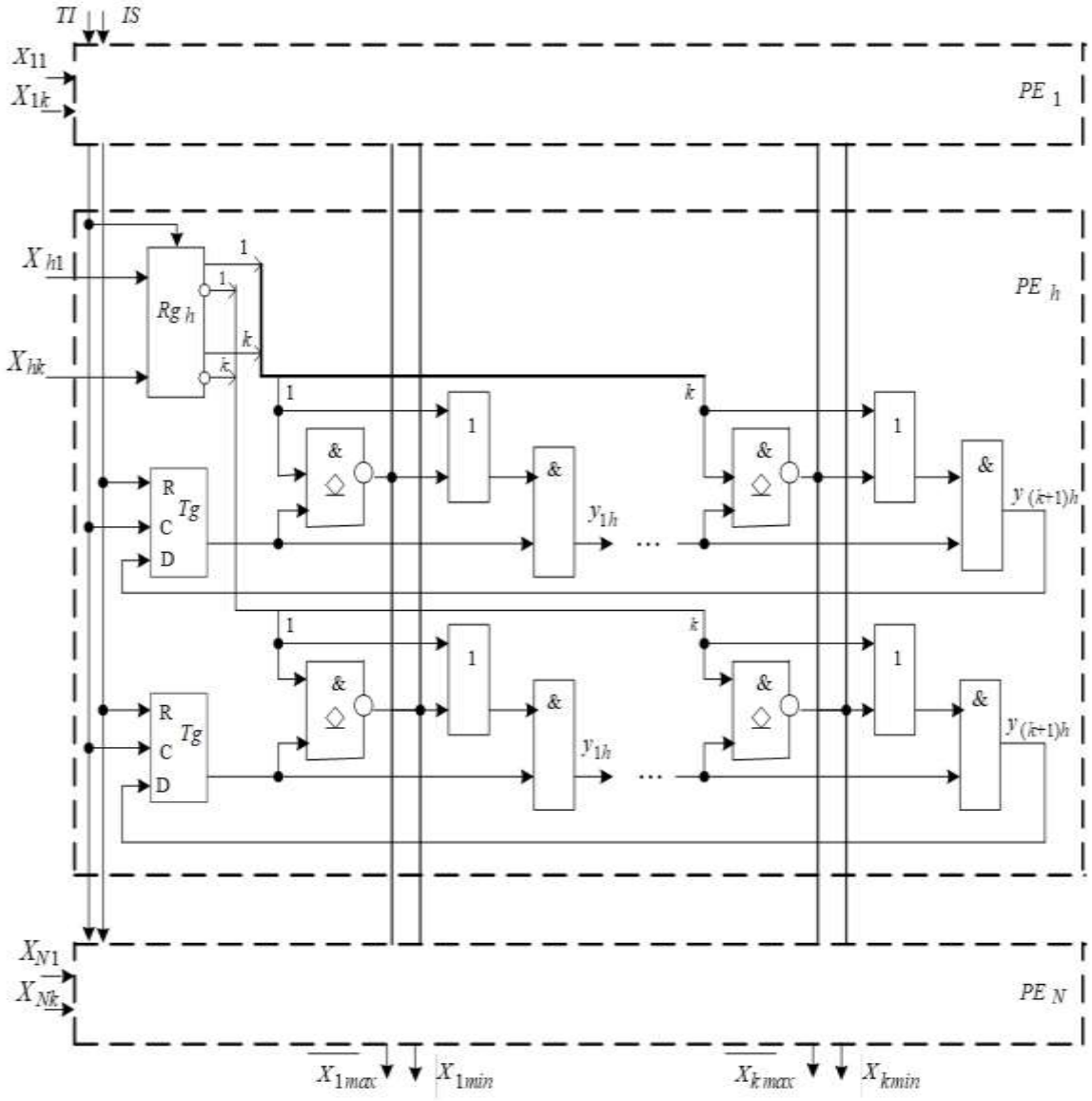

Fig. 3. Structure of a recursive device for finding maximum and minimum numbers in a one-dimensional array

3.4. Structure of a recursive device for vertical-group calculation of the sum of squared differences

Depending on the method of creating and summarizing the macro-partial results of the sum of squared differences Q_{Mg}, the following options for implementing a recursive device for calculating the sum of squared differences are possible:

with serial creation and summarization of macro-partial results of squaring QMg;

with parallel formation and serial summation of macro-partial results of squaring QMg;

with parallel formation and summation of macro-partial results of squaring QMg.

The structure of the recursive device for calculating the sum of squared differences with parallel formation and sequential summation of macro-partial results of squaring Q_{Mg} is shown in Fig. 4, where *Rg* is a register, *PMIA* is a pipelined multi-input adder, *Ad* is an adder, *PE* is a processor element.

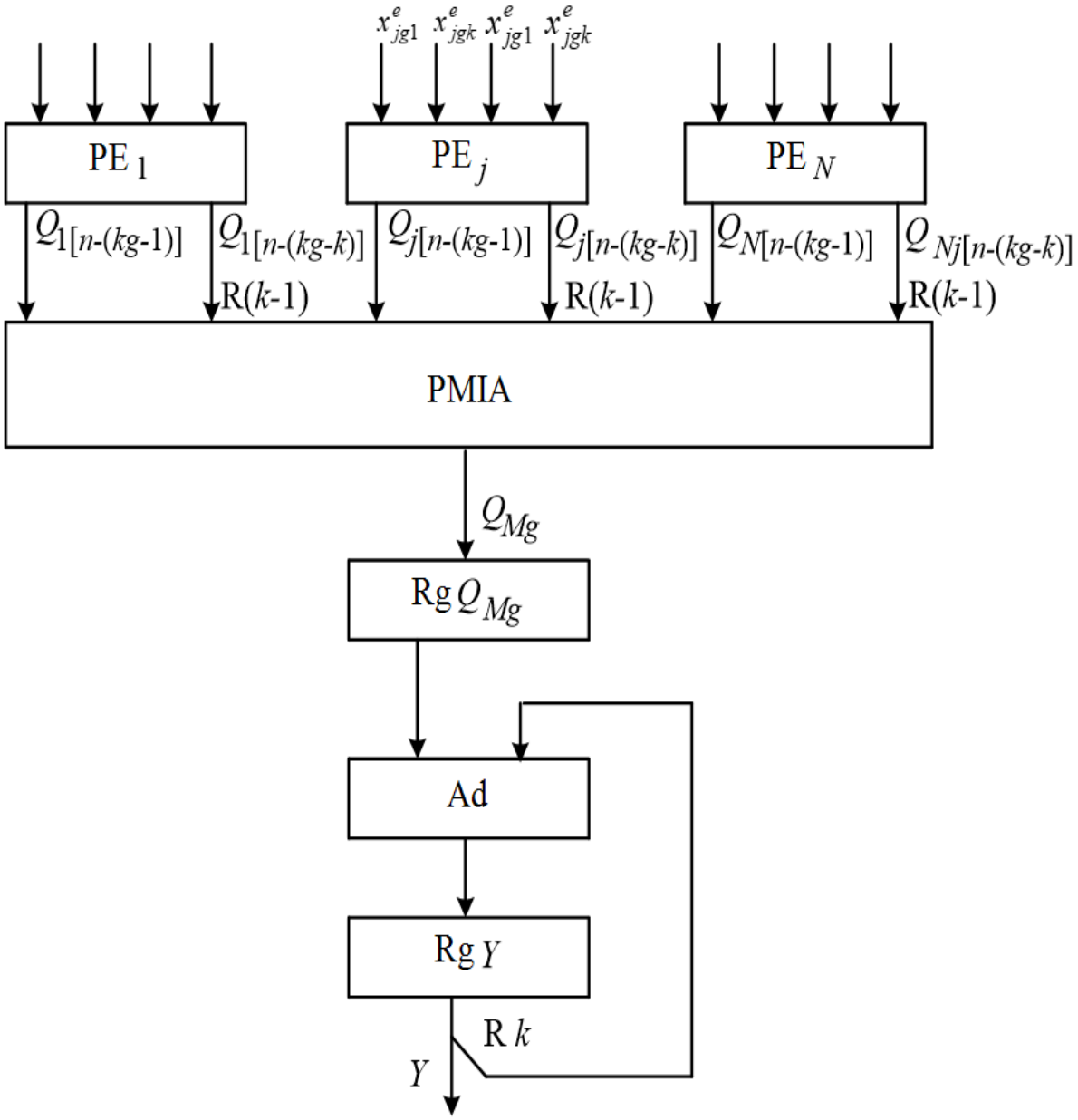

Fig. 4. Structure of a recursive device for calculating the sum of squared differences

The main components of this structure are N processor elements *PEj* and $N \times k$-input pipeline adder *PMIA*. We will perform the summation of $N \times k$ partial results of squaring using a cascade algorithm. The number of steps required to implement such summation is calculated using the following formula:

$$s = \lceil log_2(N \times k) \rceil, \quad (21)$$

At each step, the operands are split into pairs, and the sum is calculated for each pair. For VLSI implementations, modified cascade summation algorithms without carry propagation can be used.

For the calculation of the difference module ΔX_j and the creation of k partial results of squaring $Q_{j[n-(kg-r)]}$, a PEj structure was developed, which is shown in Fig. 5, where *Sub* is a subtractor, *Tg* is a trigger, *Rg* is a register, $|\Delta X_j|$ is the difference module.

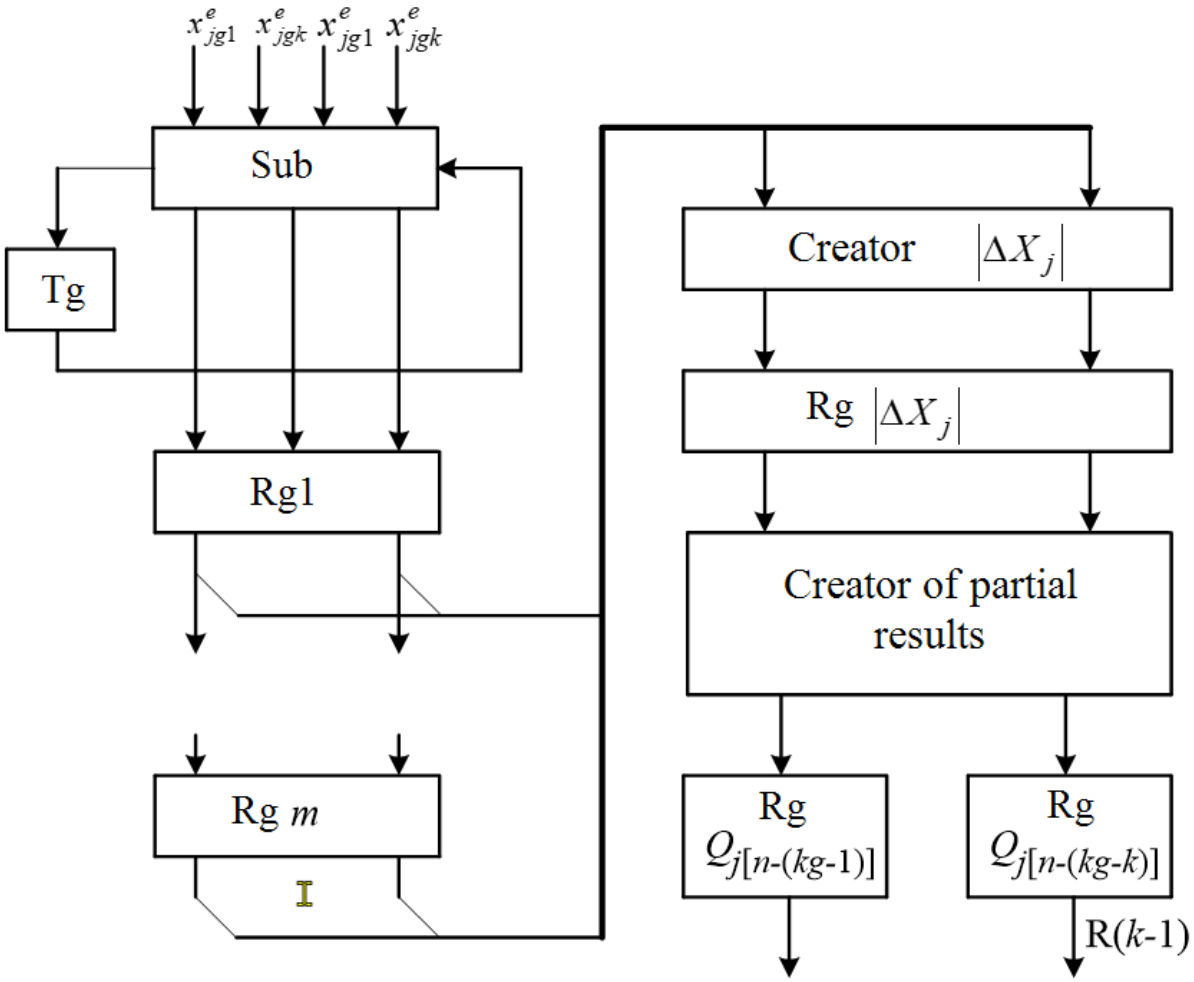

Fig. 5. Structure of a PE recursive device for calculating the sum of squared differences

The developed structure of PE_j is oriented on the alignment in time of the processes of calculation of the difference module $|\Delta X_j|$ for one input array and the creation of k partial results of squaring $Q_{j[n-(kg-r)]}$ for another input array.

The operands X_j^e and X_j^b are fed to the PE_j input sequentially in groups of k bits, starting with the lower bits. In each PE_j, using the subtractor *Sub* for m cycles, the difference ΔX_j is calculated and written to registers *Rg1*, …, *Rgm*. The calculated difference ΔX_i is fed to the inputs of the creator $|\Delta X_j|$, at the output of which its modulus is obtained $|\Delta X_j|$. In the following clock cycles, k partial results of squaring $Q_{j[n-(kg-r)]}$ are formed at the outputs of the partial result creators. The creation of partial results of squaring $Q_{j[n-(kg-r)]}$ is carried out starting from the higher bits of the modulus $|\Delta X_j|$. The k partial results of squaring $Q_{j[n-(kg-r)]}$ formed at the output of PE_j are fed with a right shift of *(r-1)* bits to the inputs of the pipeline $N \times k$ -input adder *PMIA*. The sum obtained at the output of *PMIA* is the macro partial result of squaring Q_{Mg}, which is written to the register RgQ_{Mg}. At each clock cycle, the *Ad* adder adds the data from the output of the RgQ_{Mg} register to the previously accumulated sum from the *RgY* register shifted k digits to the right according to the following formula:

$$Y_g = 2^{-k} Y_{g-1} + P_{Mg}, \tag{22}$$

where Y_0=0.

The time for calculating the sum of squared differences is determined by the following formula:

$$t_{SSD} = \left(\left\lceil \frac{n}{k} \right\rceil + 4 + log_2 N\right)\left(t_{Rg} + t_{Sm}\right), \tag{23}$$

where t_{SSD} is the time for calculation of the sum of squared differences, t_{Rg} is the register response time, and t_{Sm} is the time for addition of two numbers.

3.5. Structure of a recursive device for vertical-group calculation of scalar product

The structure of the hardware component that implements the vertical-group calculation method of the scalar product depends on the following:
the use of separate or multiplexed buses for inputting input data Xj and weighting coefficients Wj;
parallel vertical-group method of scalar product calculation of serial or parallel formation of g-th macro-partial result of scalar product calculation PgM;
separation or combination of processes of receiving operands of one array and scalar product calculation for operands of the second array.

For the VLSI implementation of the parallel vertical-group method of scalar product calculation, we choose a structure that provides:
use of 2N channels with a bit depth of k for inputting input data Xj and weighting coefficients Wj;
input of input data Xj and weighting coefficients Wj in groups of k bits starting from the lower bits;
use of N paths for data processing;
parallel formation of partial products in each g-th cycle $N \times k$;
calculation of the g-th macro-partial result of the scalar product PgM by parallel-pipeline summation $N \times k$ of partial products;
serial summation of macro-partial results of the scalar product PgM;
alignment in time of the processes of receiving weighting coefficients Wj and scalar product calculation.

The structure of the recursive device for vertical-group calculation of the scalar product is shown in Fig. 6, where *PE* is a processor element, *Rg* is a register, *PMIA* is a pipelined multi-input adder, and *Ad* is an adder.

The main element of this structure is PE_j, at the output of which k partial products are created, where each *r-th* partial product $P_{j[(g-1)k+r]}$ is shifted to the right by *(r-1)* bits. Parallel vertical-group calculation of the scalar product in this device is broken down into two steps, each of which is performed in *m* cycles.

In the first step, in each *g-th* clock cycle, *k* bits of input data X_j and *k* bits of weighting coefficients W_j are fed to the input of PE_j. The input of groups of input data X_j and weighting coefficients W_j starts from the lower bits. The first step ends at *(m+1)* with the recording of the weighting coefficient W_j in the register RgW_j and k lower bits in the register Rgx_{jg}.

At the second step, in each g at the clock frequency in PE_j for a group of input data bits $X_{j[(g-1)k+1]} \dots X_{j[(g-1)k+r]} \dots X_{j[(g-1)k+k]}$, k partial products are formed in accordance with the expression $P_{j[(g-1)k+r]} = W_j X_{j[(g-1)k+r]}$. The partial products formed in PE_j are fed to the input of the *PMIA* pipeline multi-input adder, with the *r-th* (r=1, …, k) partial product $P_{j[(g-1)k+r]}$ shifted relative to the *(r-1)-th* partial product $P_{j[(g-1)k+r-1]}$ by one digit to the right. By adding $N \times k$ partial products at the output of *PMIA*, we obtain the macro-partial result of the scalar product P_{gM}, which is written to the register RgP_{gM}. The *Ad* adder adds the macro-partial result of the scalar product P_{gM} to the previously calculated sum shifted to the right by *k* digits, in accordance with the expression $Z_g = 2^{-k} Z_{g-1} + P_{gM}$, where $Z_0=0$.

The developed recursive device for vertical-group calculation of scalar product operates on a pipeline principle and is oriented towards processing continuous data flows. In the developed device, the time for scalar product calculation is determined by the formula:

$$t_{SP} = \left(\left\lceil \frac{n}{k} \right\rceil + 3 + log_2 N\right)\left(t_{Rg} + t_{Sm}\right), \quad (24)$$

where t_{SP} is the time of scalar product calculation.

4. The method of synthesis of recursive devices for calculation of basic multi-operand neuro-operations has been improved

Recursive devices for calculation of basic multi-operand neuro-operations must provide the following requirements:
high hardware utilization efficiency;
adaptation to the requirements of specific applications;
synchronization of input data arrival time with the calculation time of the basic multi-operand neuro-operation;
real-time operation;
focus on VLSI implementation;
short development time and low cost;
small number of interface outputs.

To evaluate the developed recursive devices for the calculation of basic multi-operand neuro-operations, the criteria of hardware utilization efficiency of the E_{DBMN} are used, which take into account the complexity of the algorithm for implementing a multi-operand neuro-operation, the number of external interface outputs, the homogeneity of the structure, and links the execution time of the basic multi-operand neuro-operation with hardware costs and estimates the performance of the device elements. The quantitative value of hardware utilization efficiency for devices for calculating basic multi-operand neuro-operations is determined as follows:

$$E_{DBMN} = \frac{R_{BNO}}{t_{BNO}(k_1 W_{DBMN} + k_2 Q)}, \quad (25)$$

where E_{DBMN} is the hardware utilization efficiency of the device for calculating basic multi-operand neuro-operations; R_{BNO} is the complexity of the algorithm for implementing basic neuro-operations; t_{BNO} is the execution time of basic neuro-operations; W_{DBMN} – equipment costs for the implementation of the device for the calculation of basic multi-operand neuro-operations; *k1* – coefficient for taking into account the homogeneity of the structure, *k2* – coefficient for taking into account the number of external interface outputs; *Q* – number of external interface outputs.

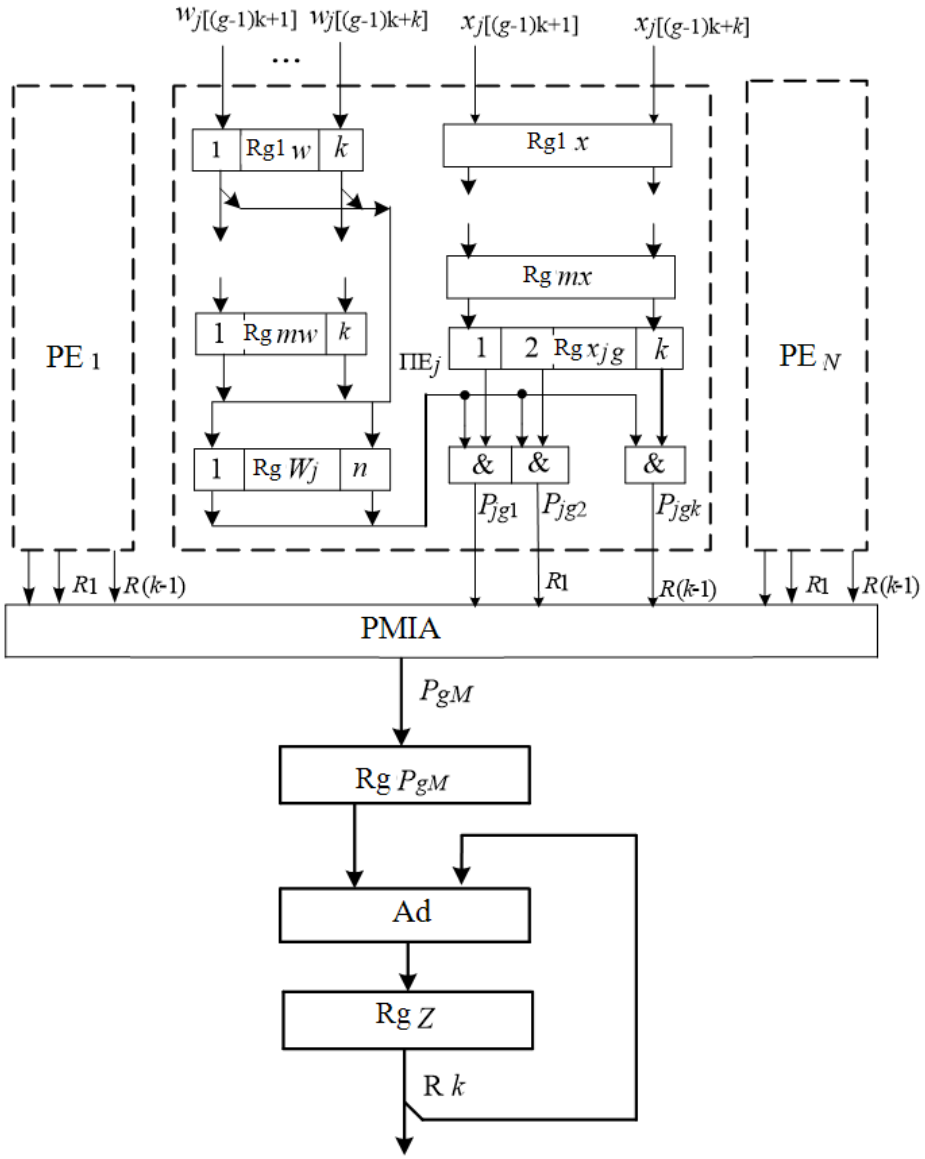

Fig. 6. Structure of a recursive device for vertical-group scalar product calculation

The task of synthesizing recursive devices for the calculation of basic multi-operand neuro-operations with vertical-group data processing is reduced to providing real-time work with minimal hardware costs for their implementation. The output data for the synthesis of recursive devices with vertical-group data processing is the arrival time of arrays from N input data X_1, ..., X_N:

$$t_d = NT_{TI1}, \tag{26}$$

where T_{TI1} is the duration of the input data X_j arrival period.

To provide the work of recursive devices for the calculation of basic multi-operand neuro-operations in real time, the following condition must be met:

$$t_d \geq t_{BMN}, \tag{27}$$

where t_{BMN} is the calculation time of a multi-operand basic neuro-operation.

The calculation time t_{BMN} depends on the number of bits in group *k*, which are simultaneously processed during the implementation of algorithms for calculating basic multi-operand neuro-operations, the amount of data *N*,

and the bit depth of the input data *n*. The time for searching for maximum and minimum numbers in a one-dimensional array, calculating the sum of squared differences and the scalar product is determined by formulas (20), (23) and (24), respectively.

The main parameter that reduces the calculation time t_{BMN} is *k*, whose value can vary in the range $k=2, \ldots, n/2$. Increasing the value of *k* leads to a decrease in the number of clock cycles *m* and an increase in hardware costs. The second parameter on which the calculation time of the sum of squared differences and scalar product depends is the time of summing of *N* numbers. This time can be reduced by pipelining the *N*-input adder, i.e., dividing it into steps using registers.

To select an option for implementing a recursive device for calculating a basic multi-operand neuro-operation in real time, we will use the criteria of hardware utilization efficiency of the E_{DBMN}. High efficiency of hardware utilization E_{DBMN} in the implementation of a recursive device for calculating basic multi-operand neuro-operations in real time is achieved by synchronizing the data arrival time t_d with the calculation time of the basic multi-operand neuro-operation t_{BMN}. Such synchronization may require both an increase and a decrease in t_{BMN}.

The main ways to reduce t_{BMN} are:

increasing the number of k bits in a group that are simultaneously processed when implementing algorithms for calculating basic multi-operand neuro-operations;

pipelining of an N-input adder by dividing it into steps;

parallel inclusion of two or more devices for the calculation of basic multi-operand neuro-operations, the number of which is mainly determined by the time td of input data arrival.

In the case when the data arrival time t_d is significantly greater than the time t_{BMN}, it is necessary to synchronize them to provide high hardware utilization efficiency. Such synchronization can be achieved by reducing the number of bits k in the group that are simultaneously processed when implementing algorithms for calculating basic multi-operand neuro-operations or by using devices that process data arrays of dimensions *N*/2 and *N*/4 for calculation.

For the synthesis of recursive devices for the calculation of basic multi-operand neuro-operations with vertical-group data processing in real time with a given data arrival time t_d, it is advisable to use the devices developed in Fig. 3, Fig. 4, and Fig. 6 as a basis. The synthesis of such recursive devices requires the following steps:

1) determine the number of bits *k* in the group necessary for synchronization of data arrival time and calculation time;

2) for the case when $t_d < t_{BMN}$ and $k=n/2$, the reduction in the calculation time of the basic multi-operand neuro-operation can be achieved by using two or more devices operating in parallel;

3) for the case when $t_d > t_{BMN}$ and $k=1$, the calculation of the basic multi-operand neuro-operation is implemented on devices with lower hardware costs that process data arrays of dimensions *N*/2 and *N*/4;

4) evaluate the hardware utilization efficiency for different variants of recursive devices for calculating basic multi-operand neuro-operations with vertical-group data processing and select the device option with the highest hardware utilization efficiency.

Discussion of the results

A distinctive feature of the methods and structures of parallel-vertical calculation devices for basic multi-operand neuro-operations (finding maximum and minimum values in a one-dimensional data array, calculating the sum of squared differences, scalar product calculation), described in [20, 21], is the complexity of synchronizing the data arrival time with the calculation time. The developed basic structures of recursive-type devices with vertical-group data processing provide synchronization of the time of input data arrival with the calculation time. This is achieved by selecting the number of bits in a group that are simultaneously processed at each clock cycle of the device. Thanks to the use of a data format converter, which converts the stream of sequential input data of a one-dimensional array into parallel-serial data output in groups of bits, synchronization of the input data arrival time with the calculation time is ensured.

A disadvantage of the study is the lack of analysis of options for implementing devices for parallel-vertical calculation of basic multi-operand neuro-operations with reference to specific architectures of programmable logic integrated circuits, such as CPLD for example.

Further research on the synthesis of parallel-vertical calculation devices for basic multi-operand neuro-operations will be focused on the development of high-speed pipelined multi-input adders used for summing partial products and group partial products. Research focused on developing tools for automating the process of synthesizing devices for parallel-vertical calculation of basic multi-operand neuro-operations in real time with high hardware utilization efficiency is also relevant.

Thus, based on the results of the paper, the following scientific novelty and practical significance of the research results can be formulated.

Scientific novelty of the research results obtained:

Improved methods of vertical-group calculation of basic multi-operand neuro-operations (finding maximum and minimum values in a one-dimensional data array, calculating the sum of squared differences, scalar product

calculation) have been improved, which, by selecting the number of bits in the operand slice for processing in one cycle, provide synchronization of data arrival time with calculation time and high hardware utilization efficiency in their hardware implementation;

The method of synthesis of recursive devices for calculating basic multi-operand neuro-operations with vertical-group data processing has been improved, which, through the use of mechanisms for synchronizing the calculation time with the data arrival time, provides the selection of a structure that performs data processing in real time and has high hardware utilization efficiency.

Practical significance of the research results: the use of improved vertical-group methods, developed basic structures of devices for finding maximum and minimum numbers in one-dimensional arrays, calculation of sums of squared differences and scalar products, and an improved synthesis method makes it possible to provide real-time mode and implementation of devices for calculating basic multi-operand neuro-operations with vertical-group data processing with high hardware utilization efficiency.

Conclusions

The operational basis of ANNs has been identified, basic multi-operand neuro-operations have been selected for hardware implementation, the methods of vertical-group calculation of selected basic neuro-operations have been improved, the basic structures and method of synthesis of recursive devices for parallel vertical-group calculation of basic multi-operand neuro-operations in real time have been developed. Based on the results of the research, the following main conclusions can be drawn.

1. The operational basis of the ANNs has been identified, which consists of groups of the following neuro-operations: pre-processing, processing, and calculation of transfer functions. Basic multi-operand neuro-operations have been selected for hardware implementation: finding maximum and minimum values in a one-dimensional data array, calculating the sum of squared differences, and scalar product calculation.

2. Methods of vertical-group calculation of basic multi-operand neuro-operations (finding maximum and minimum values in a one-dimensional data array, calculating the sum of squared differences, scalar product calculation), which, by selecting the number of bits in the operand slice for processing in one cycle, provide synchronization of data arrival time with calculation time and high hardware utilization efficiency in their hardware implementation.

3. It is proposed to develop recursive devices for vertical-group calculation of basic multi-operand neuro-operations based on an integrated approach, which is based on the capabilities of modern element base, covers vertical methods, algorithms, and recursive device structures for implementing basic neuro-operations, and takes into account the requirements of specific applications.

4. The principles for the development of recursive devices for vertical-group calculation of basic multi-operand neuro-operations have been selected, the main ones being: the use of a basis of elementary arithmetic operations and a multi-operand approach; modularity; pipelining and spatial parallelism; homogeneity and regularity of structure; synchronization of data arrival time with the calculation time of the neural operation; specialization and adaptation of the device structure to the requirements of a specific application.

5. A format converter has been developed that converts a stream of serial input data from a one-dimensional array into parallel-serial data output in groups of bits.

6. Basic structures have been developed that represents from hardware side the calculation algorithms and form the basis for the synthesis of recursive devices for vertical-group calculation of basic multi-operand neuro-operations with specified parameters.

7. The method of synthesis of recursive devices for calculation of basic multi-operand neuro-operations with vertical-group data processing has been improved, which, due to the use of mechanisms for synchronization of calculation time with data arrival time, provides the selection of a structure that performs data processing in real time and has high hardware utilization efficiency.

References

1. Lee D., Park M., Kim H., Jeon M. AI-based mobile robot navigation using deep neural networks and reinforcement learning. IEEE Access, 2021. №9. P. 329-345. https://doi.org/10.1109/ACCESS.2021.3102345
2. Tsmots I., Skorokhoda O., Ignatyev I., Rabyk V. Basic Vertical-Parallel Real Time Neural Network Components. Proceedings of XIIth International Scientific and Technical Conference CSIT 2017, 5-8 Sept. 2017, Lviv, Ukraine, pp. 344-347.
3. Chang A. X. M., Martini B., Culurciello E. Recurrent neural networks hardware implementation on FPGA: arXiv preprint arXiv:1511.05552. 2015.
4. Tsmots I. H., Antoniv V. Ya. Methods and tools for vertical-parallel searching of maximum and minimum numbers in arrays. Ukrainian Journal of Information Technology. 2022. 4(1). P. 68–77. https://doi.org/10.23939/ujit2022.01.0068.
5. Kundu S., Banerjee S., Raha A., Basu K. Special Session: Effective In-field Testing of Deep Neural Network Hardware Accelerators. 2022 IEEE 40th VLSI Test Symposium (VTS), San Diego, CA, USA, 2022. Pp. 1-4. DOI: 10.1109/VTS52500.2021.9794227/.
6. Wu J., Zhao B., Wen H., Zhao Q. Design of Neural Network Accelerator Based on In-Memory Computing Theory. 2022 4th International Conference on Natural Language Processing (ICNLP), Xi'an, China, 2022. Pp. 547-551, doi: 10.1109/ICNLP55136.2022.00100.
7. Strategy for Artificial Intelligence Development in Ukraine: monograph / [Under the general editorship of A. Shevchenko]. Kyiv: IAIP, 2023. 305 p. https://doi.org/10.15407/development_strategy_2023.
8. Sarg M., Khalil A. H., Mostafa H. Efficient HLS Implementation for Convolutional Neural Networks Accelerator on an SoC. 2021 International Conference on Microelectronics (ICM), New Cairo City, Egypt, 2021. Pp. 1-4. DOI: 10.1109/ICM52667.2021.9664920.

9. Nandhini M., Duraiswamy K. A systematic literature review on hardware implementation of artificial intelligence algorithms. Microprocessors and Microsystems. 2020. 77, 103142. https://doi.org/10.1016/j.micpro.2020.103142.
10. Saha S., Patnaik S. VLSI and hardware implementations using modern machine learning methods. Springer. 2021. https://doi.org/10.1007/978-3-030-70713-7.
11. Kozhemyako V. P., Kozhemyako A. V., Vasylkiva O. S. The current status, element base and comparative analysis of characteristics neurochips. Optoelectronic information-power technologies. 2017. Vol. 32. No. 2. Pp. 29–38.
12. Davies M. et al. Loihi: A Neuromorphic Manycore Processor with On-Chip Learning. IEEE Micro. Jan.–Feb. 2018. Vol. 38. No. 1. Pp. 82–99. DOI: 10.1109/MM.2018.112130359.
13. Hager G., Wellein G. Introduction to High Performance Computing for Scientists and Engineers. Boca Raton: CRC Press, 2010. 356 p.,
14. Hennessy J. L., Patterson D. A. Computer Architecture: A Quantitative Approach. 6th ed., Morgan Kaufmann, 2017. 936 p.
15. Tsmots I. H., Lukashchuk Yu. A., Ihnatyev I. V., Kazymyra I. Ya. Components of hardware neural networks for coordinated parallel-vertical data processing in real time. Ukrainian Journal of Information Technology. 2021. 3(1). P. 63–72. https://doi.org/10.23939/ujit2021.03.063
16. Liu W., Wang Z., Liu X., Zeng N., Liu Y., Alsaadi F. E. A survey of deep neural network architectures and their applications. Neurocomputing. 2017. V. 234. P. 11-26. DOI: 10.1016/j.neucom.2016.12.038.
17. Rashkevych Yu. M., Tkachenko R. O., Tsmots I. H., Pelesko D. D. Neiropodibni metody, algorytmy ta struktury obrobky syhnaliv i zobrazhen u realnomu chasi: monohrafiia [Neuro-like methods, algorithms and structures for real-time signal and image processing: monograph]. Lviv: Lviv Polytechnic Publishing House, 2014. (in Ukrainian)
18. Kaul H. et al. Hardware for machine learning: Challenges and opportunities. In 2017 IEEE Custom Integrated Circuits Conference (CICC). 2017. P. 1–8. IEEE. https://doi.org/10.1109/CICC.2017.7993622.
19. Wang G., Fu D. Spike Neural Network with Delayed Propagation Characteristics and Hardware Implementation. 6th International Conference on Electronic Engineering and Informatics (EEI), Chongqing, China, 2024. Pp. 1181-1185. DOI: 10.1109/EEI63073.2024.10696338.
20. Tsmots I. H., Teslyuk V. M., Opotiak Yu. V. Prystrii dlia vyznachennia maksymalnoho ta minimalnoho chysel u dvovymirnomu masyvi chysel [Device for determining the maximum and minimum numbers in a two-dimensional array of numbers], Patent of Ukraine, no. 128150, published Apr. 14, 2024, Bull. no. 16. (in Ukrainian)
21. Tsmots I. H., Teslyuk V. M., Lukashchuk Yu. A., Kazymyra I. Ya. Prystrii dlia obchyslennia skalyarnoho dobuktu [Device for calculating the scalar product], Patent of Ukraine, no. 127774, IPC G06G 6/33, appl. no. u202010852, filed May 19, 2020, published Dec. 27, 2023, Bull. no. 52. (in Ukrainian)
22. Guo K., Zeng S., Yu J., Wang Y., Yang H. A Survey of FPGA BasedNeural Network Accelerator. ACM Trans. Reconfig. Technol. Syst. 2017. Vol. 9, No. 4. 21 p. https://doi.org/10.1145/3106709.
23. Liu A. C.-C., Law O. M. K. Artificial Intelligence Hardware Design: Challenges and Solutions. John Wiley & Sons. 2021. 240 p.

Ivan Tsmots **Іван Цмоць**	Doctor of Engineering Sciences, Professor. Department of Automated Control Systems, Lviv Polytechnic National University https://orcid.org/0000-0002-4033-8618 e-mail: ivan.tsmots@gmail.com	доктор технічних наук, професор кафедри Автоматизованих систем управління. Національний університет «Львівська політехніка»
Oleh Berezsky **Олег Березький**	Doctor of Engineering Sciences, Professor. Department of Computer Engineering. West Ukrainian National University https://orcid.org/0000-0001-9931-4154 e-mail: olber62@gmail.com	доктор технічних наук, професор кафедри Комп'ютерної інженерії. Західноукраїнський національний університет
Taras B. Mamchur **Тарас Мамчур**	PhD student. Department of Automated Control Systems. Lviv Polytechnic National University https://orcid.org/0009-0006-0593-7937 e-mail: taras.b.mamchur@lpnu.ua	аспірант кафедри Автоматизованих систем управління. Національний університет «Львівська політехніка»