INFORMATION TECHNOLOGY

This study considers an approach to improving the information technology of structure synthesis for real-time multi-operand neural operation data processing. The task addressed relates to the lack of a formalized approach to the synthesis of such structures that would simultaneously take into account the parameters of data flows, the depth of the pipeline, the degree of parallelism, and hardware limitations while ensuring the specified time characteristics.

Methods for parallel vertical-group calculation of the scalar product, sum of squared differences, and search for maximum and minimum values have been devised. A method for concretizing flow graphs has been improved. Basic parallel-stream computing structures and analytical expressions for estimating hardware costs, pipeline cycle time, and efficiency of hardware resource use have been developed. Based on the above, the information technology for synthesizing parallel-stream structures for vertical-group calculation of real-time multi-operand neural operations has been improved. The task set was solved by combining vertical and group parallelism, conveyorization, modular organization, matching the intensity of data input with the intensity of their processing and a gradual transition from algorithmic description to hardware implementation.

The improved information technology provides a reduction in hardware costs, increased throughput, reduced latency, and the selection of optimal parameters of structures. Simultaneous processing of groups of bit slices reduces the number of pipeline steps, and the specification of flow graphs makes it possible to adapt the structure of calculations to real-time requirements.

In practice, the results could be used for synthesizing specialized FPGA-, ASIC-, SoC-tools for neural-oriented real-time systems with specified characteristics

Keywords: flow graphs, real-time systems, hardware costs, resource efficiency

UDC 004.94
DOI: 10.15587/1729-4061.2026.357361

IMPROVEMENT OF INFORMATION TECHNOLOGY FOR SYNTHESIZING PARALLEL-STREAM STRUCTURES FOR VERTICAL-GROUP COMPUTING OF MULTI-OPERAND NEURAL OPERATIONS IN REAL TIME

Ivan Tsmots
Doctor of Technical Sciences*
ORCID: https://orcid.org/0000-0002-4033-8618
Vasyl Teslyuk
Doctor of Technical Sciences*
ORCID: https://orcid.org/0000-0002-5974-9310
Yurii Opotyak
Corresponding author
Candidate of Technical Sciences*
ORCID: https://orcid.org/0000-0001-9889-4177
Bohdan Shtohrinets*
E-mail: bohdan.v.shtohrinets@lpnu.ua
ORCID: https://orcid.org/0009-0001-4956-3862
*Department of Automated Control Systems
Lviv Polytechnic National University
S. Bandery str., 12, Lviv, Ukraine, 79013

Received 04.03.2026
Received in revised form 19.05.2026
Accepted date 29.05.2026
Published date 30.06.2026

How to Cite: *Tsmots, I., Teslyuk, V., Opotyak, Y., Shtohrinets, B. (2026). Improvement of information technology for synthesizing parallel-stream structures for vertical-group computing of multi-operand neural operations in real time. Eastern-European Journal of Enterprise Technologies, 3 (2 (141)), 6–23. https://doi.org/10.15587/1729-4061.2026.357361*

1. Introduction

The modern development of real-time neural-oriented systems is due to the increasing requirements for autonomy, adaptability, and speed of data processing in complex dynamic environments [1–3]. Such systems are widely used in autonomous vehicles, mobile robotic platforms, vision systems, security devices, medical and embedded intelligent devices [2, 4]. Their effective functioning requires high-performance processing of multi-channel streaming data in real-time under conditions of limited hardware resources and low power consumption [3, 6 ,5].

The key issue of implementing neural algorithms in such systems is the high computational complexity of multi-operand neural operations, in particular, the scalar product, the sum of squared differences and group summation, which form the main part of the computational costs [1, 7]. Conventional universal processor architectures and horizontal parallel computing methods do not provide simultaneous fulfillment of the requirements for speed, scalability, regularity of structures, and guaranteed computation time, which is critical for real-time systems [8, 9].

State-of-the-art hardware solutions based on FPGA and ASIC make it possible to accelerate individual neural operations; however, they are characterized by limited scalability and lack of formal coordination between the intensity of incoming streaming data and the parameters of computational pipelines [3, 10]. This leads to inefficient use of hardware resources and complicates the adaptation of such structures to new classes of neural operations.

A promising direction for overcoming these limitations is the use of vertically parallel multi-operand parallel-stream computing structures. Such structures provide simultaneous processing of groups of bit slices, coordination of the data input frequency with the pipeline clock frequency, as well as formalized formation of partial and macro-partial products [11, 12]. This approach creates prerequisites for optimizing the time and hardware characteristics of computing resources and guaranteeing real-time operation. Therefore, the relevance of research relates to the development of methods for implementing multi-operand neural operations that are capable of providing real-time data processing. Based on the devised methods, it is necessary to improve the information technology for the synthesis of multi-operand parallel-stream computing structures focused on neuro-like data processing. Such information technology should ensure simultaneous consideration of the requirements for real-time data processing and achieving high efficiency in the use of hardware resources.

2. Literature review and problem statement

Analysis of the current stage of evolution of neural-oriented computing systems reveals that it is characterized by a rapid growth in data volumes and the complexity of artificial neural network models, which leads to increased requirements for speed and efficiency of use of computing resources [13, 14]. A review of studies [15, 16] shows that the main areas of research are the development of hardware accelerators of neural networks based on GPU, FPGA, and ASIC, as well as the optimization of computing structures for the implementation of basic neural operations. Papers [17, 18] consider approaches to building hardware accelerators of deep neural networks based on ASIC and FPGA with an emphasis on increasing computing performance and efficiency of use of hardware resources. At the same time, insufficient attention is paid in those studies to the synthesis of computing structures at the level of basic neural operations and the coordination of the time of receipt of input data with the calculation cycle.

In [19, 20] it is shown that the increase in GPU performance is achieved through massively parallel processing and the use of vector instructions. At the same time, such solutions remain energy-intensive and do not ensure the efficient use of hardware in embedded real-time systems. The issue of guaranteeing timing characteristics when processing intensive data streams remains unresolved, which is due to the orientation of universal processor architectures to a wide range of tasks and limits their ability to specialize for specific computational patterns of neural networks.

In current studies [21, 22] on FPGA and ASIC implementations of neural network components, most attention is on adapting the architecture to the algorithmic structure of neural operations. At the same time, the analysis of papers [23, 24] reveals that most of the existing solutions are focused on the implementation of individual types of neural algorithms. The issue of devising universal methods for synthesizing computational structures capable of effectively implementing various multi-operand neural operations taking into account real-time constraints remains unresolved. Its solution is due to the complexity of formalizing the transition from algorithmic description to hardware implementation taking into account data flow parameters.

In [25, 26] it is shown that a significant part of the research on increasing the speed of neural networks is focused on the hardware implementation of basic neural operations, in particular the scalar product [27], which is a key operation of most neural network algorithms. Existing approaches are mostly based on multiplication and addition or their simplified versions (shifts, addition, inversion), use mainly two-operand calculations and provide only sequential or limited parallel processing. The issue of effective hardware implementation of multi-operand calculations, characteristic of modern neural-oriented systems, remains unresolved.

Analysis of works [28, 29] reveals that parallel-vertical methods are used to calculate basic multi-operand neural operations, which involve simultaneous processing of a bit slice of all operands. This approach makes it possible to increase performance and reduce latency. The main disadvantage of the parallel-vertical method is the difficulty of matching the time of arrival of operands with the clock of the pipeline in parallel-stream structures, which complicates the effective implementation of multi-operand neural operations. Analysis of methods for synthesizing structures for computing basic neural operations in real time [30] showed the lack of formalized approaches capable of simultaneously taking into account the parameters of data flows, pipeline depth, and computation structure.

Our review of the literature demonstrates that the shortcomings of current methods for synthesizing parallel-stream structures for the calculation of multi-operand neural operations are the complexity of coordinating the time of data arrival with the pipeline clock and insufficient formalization of the transition from the algorithm to the hardware implementation. In addition, there are no effective methods for the calculation and synthesis of parallel-stream structures for basic multi-operand neural operations, capable of ensuring real-time operation and high efficiency of hardware resources. An unresolved task is the synthesis of parallel-stream structures for vertical-group calculation of multi-operand neural operations in real time, which simultaneously take into account the parameters of data flows, the depth of the pipeline, the degree of parallelism, as well as hardware limitations while ensuring the specified time characteristics.

The aforementioned allows us to assert that it is advisable to conduct research on the improvement and development of information technology for synthesizing these structures. It is based on the use of the methods devised in the work for parallel vertical-group calculation of the scalar product, sum of squared differences, search for maximum and minimum values, and an improved method for concretizing flow graphs.

3. The aim and objectives of the study

The aim of our work is to improve the information technology of synthesis of parallel-stream structures for vertical-group computing of multi-operand neural operations in real time, which ensures high efficiency of hardware resources use and compliance with the requirements of specific applications.

To achieve the goal, the following tasks were set:

– to devise a method of parallel vertical-group computing of basic multi-operand neural operations;

– to improve the method of concretization of flow graphs for the synthesis of parallel-stream computing structures in real time;

– to design basic parallel-stream structures for the implementation of multi-operand neural operations in real time and evaluate their parameters;

– to determine a set of requirements for information technology that form its functional capabilities and ensure its improvement through a systematic, formalized, and reproducible process of transition from algorithmic description of neural operations to effective hardware implementations.

4. The study materials and methods

The object of our study is the synthesis processes of parallel-stream computing structures designed to implement multi-operand neural operations in real-time systems with high efficiency of hardware resources use.

The hypothesis of the study assumes the possibility of ensuring the processing of multi-operand neural operation data at the rate of their arrival while simultaneously taking into account the requirements for the efficiency of hardware resources use. In this study, the use of hardware resources is assessed using measurable criteria, namely calculation of the duration of the pipeline cycle, hardware costs, and the efficiency of hardware resources use for the implementation of parallel-stream devices [6].

In the process of improving information technology, the following was assumed:

1) to implement multi-operand neural operations, computational structures are used that allow parallelization and pipelined data processing;

2) a possible redistribution of the total computational complexity of a specific algorithm between the number of simultaneously processed data processing streams and the data bit rate of a separate stream;

3) in the process of executing neural network algorithms, multi-operand neural operations are performed;

4) parallel-vertical calculation of multi-operand operations is based on simultaneous processing of groups of bit slices of all operands with subsequent reduction of partial results;

5) when implementing neural network algorithms, basic multi-operand operations are scalar product, convolution, sum of squares of differences, group summation, search for maximum and minimum values in arrays of numbers;

6) the calculation of multi-operand neural operations is reduced to the parallel execution of the same type of computing operations on groups of bit slices of the array of operands, which makes it possible to significantly reduce time costs;

7) parallel vertical group processing of data by groups of digits ensures the implementation of multi-operand neural operations in parallel-stream computing structures.

The simplification adopted in the study is the possibility of parallelization and conveyorization of data processing algorithms when implementing a wide class of multi-operand neural operations. However, neural networks and neuro-like structures have a regular structure oriented towards data stream processing, which makes the specified simplification not so restrictive.

The study used a method of spatial-temporal mapping of the algorithm, which ensures the detection of all forms of parallelism and allows constructing its stream graph in a tiered-parallel form. With this form of representation of the algorithm, all its functional operators are distributed by tiers, where operators of one tier can be executed independently.

To implement the data processing process, pipeline algorithmic structures were used, which provide high throughput.

Determining the balance between the depth of the pipeline and the complexity of operations makes it possible to achieve high efficiency in the use of hardware resources, taking into account the intensity of data input for processing.

In the process of improving information technology, the following research methods were used:

– methods from the theory of parallel computing – for the analysis and synthesis of parallel-stream computing structures designed to implement basic multi-operand neural operations in real time;

– methods from the theory of algorithms – for the development and formalization of algorithms for vertical-group computing of basic multi-operand neural operations;

– methods from graph theory – for the representation of algorithms for vertical-group computing of multi-operand neural operations in the form of flow graphs and their further specification;

– optimization methods – for ensuring the effective processing of continuous data streams in parallel-stream computing structures in real time with minimal hardware costs;

– methods from the theory of digital automata – for the synthesis and analysis of parallel-stream computing structures designed to compute basic multi-operand neural operations in real time.

5. Results of improving the information technology for synthesizing multi-operand parallel-stream computing structures of real time

5. 1. Method of parallel vertical-group calculation of basic multi-operand neural operations

For neural network algorithms, the basic multi-operand operations are scalar product, convolution, sum of squares of differences, group summation, search for maximum and minimum values in arrays of numbers. Parallel-vertical calculation of such multi-operand operations is based on simultaneous processing of groups of bit slices of all operands with subsequent reduction of partial results, which ensures the collapse of a set of intermediate values into a single final result. With this approach, the calculation of multi-operand neural operations is reduced to parallel execution of the same type of computational actions on groups of bit slices of the array of operands, which makes it possible to significantly reduce time costs.

Parallel vertical-group data processing by groups of k bits is an effective method for implementing multi-operand neural operations in parallel-stream computing structures. The main idea of the method is to simultaneously process vertically organized groups of bit slices of all operands, which makes it possible to achieve a high level of parallelism and reduce computing time. In vertical-group data processing, parallelism is implemented at two levels:

– vertical parallelism – simultaneous processing of the corresponding bits of all operands;

– group parallelism – parallel execution of calculations over k-bit groups, where $k = 1, 2, \ldots, n/2$, and n is the number of bits of the operands.

The choice of parameter k is a compromise between hardware costs and the time of computing the neural operation because increasing k increases the speed due to greater parallelism but increases the complexity and hardware costs.

To obtain the final result, a hierarchical reduction is used, which contracts the set of partial results obtained in each

k-bit group. Such reduction is implemented in the form of tree-like or pipeline structures.

In the vertical-group calculation method, the input data X_j and the weight coefficients W_j ($j = 1, ..., N$, where N is the number of data and the number of weight coefficients) are processed by groups of k-bits:

$$X_j = \sum_{h=1}^{m} 2^{-(h-1)k} x_{jh1} x_{jh2} ... x_{jhk},$$

$$W_j = \sum_{h=1}^{m} 2^{-(h-1)k} w_{jh1} w_{jh2} ... w_{jhk}, \quad (1)$$

where $m = \left\lceil \frac{n}{k} \right\rceil$, $x_{jh1} x_{jh2} ... x_{jhk}$, $w_{jh1} w_{jh2} ... w_{jhk}$ – value of the h-group of bits of the j-th input data X_j and the j-th weight coefficient W_j, respectively.

Parallel vertical-group method for calculating the scalar product. The calculation of the scalar product by the parallel vertical-group method is performed using the following formula

$$\begin{aligned} Y &= \sum_{j=1}^{N} W_j X_j = \\ &= \sum_{h=1}^{m} 2^{-(h-1)k} \sum_{j=1}^{N} \left(W_j x_{jh1} + 2^{-1} W_j x_{jh2} + ... + 2^{-(k-1)} W_j x \right) = \\ &= \sum_{h=1}^{m} 2^{-(h-1)k} \sum_{j=1}^{N} P_{jh} = \sum_{h=1}^{m} 2^{-(h-1)k} P_{Mh}, \end{aligned} \quad (2)$$

where P_{jh} is the jh-th group partial product, P_{Mh} is the h-th group macro-partial product, which is formed by adding N group partial products P_{jh}, i.e.

$$P_{Mh} = \sum_{j=1}^{N} P_{jh}.$$

From formula (2) it follows that the calculation of the scalar product is performed in m cycles, in each h-th cycle the following operations are performed:

– formation of k partial products using the following expression

$$P_{jhr} = W_j x_{jhr},$$

where $r = 1,...,k$;

– calculation of the group partial product P_{jh} using the following formula

$$P_{jh} = \sum_{r=1}^{k} 2^{-(r-1)} W_j x_{jhr};$$

– calculation of the h-th macro-partial product P_{Mh} by summing the h-th partial products P_{jh} using formula $P_{Mh} = \sum_{j=1}^{N} P_{jh}$;

– calculation of the final result is carried out by summing the macro-partial products P_{Mh} using the following expression

$$Y_h = 2^{-(h-1)k} Y_{h-1} + P_{Mh},$$

where $Y_0 = 0$.

The proposed parallel vertical-group method makes it possible to significantly increase the degree of parallelism of calculations and ensure efficient use of hardware resources due to the simultaneous processing of vertically organized groups of bit slices of operands.

The method of parallel vertical group calculation of the sum of squared differences. The calculation of the sums of squared differences is performed using the following formula

$$\begin{aligned} Y &= \left(X_1^e - X_1^b \right)^2 + \left(X_2^e - X_2^b \right)^2 + ... + \left(X_N^e - X_N^b \right)^2 = \\ &= \Delta X_1^2 + \Delta X_2^2 + ... + \Delta X_N^2 = \sum_{j=1}^{N} \Delta X_j^2. \end{aligned} \quad (3)$$

From formula (3) it follows that such a calculation is based on the squaring operation, which is implemented using the following formula

$$\Delta X_j^2 = \sum_{i=1}^{n} 2^{-(i-1)} P_{ji}, \quad (4)$$

where P_{jh} is the i-th partial result of squaring the j-th number, which is determined

$$P_{ji} = \left(0.x_{j1} x_{j2} ... x_{j(i-1)} 01 \right) \wedge x_{ji}. \quad (5)$$

In the vertical group calculation of the sum of squared differences, groups of k bits are processed simultaneously in each cycle. In this processing, the number of cycles is deter mined by expression $m = \left\lceil \frac{n}{k} \right\rceil$, where n is the number of bits ΔX_j. In each h cycle, where $h = 1, ..., m$, the group partial result is calculated using the following formula

$$P_{jh} = P_{jh1} + 2^{-1} P_{jh2} + ... + 2^{-(k-1)} P_{jhk} = \sum_{r=1}^{k} 2^{-(r-1)} P_{jhr}, \quad (6)$$

where $r = 1,...,k$.

The obtained group partial results are used to calculate the h-th macro-partial result of the square using the following formula

$$P_{Mh} = \sum_{j=1}^{N} P_{jh}. \quad (7)$$

The operation of squaring the j-th ΔX_j value with processing a group of k bits is performed using the following formula

$$\Delta X_j^2 = \sum_{h=1}^{m} 2^{-(h-1)k} P_{jh}. \quad (8)$$

Using formulae (7) and (8), the calculation of the sums of squared differences with simultaneous processing of groups of k digits is carried out as follows:

$$Y = \sum_{h=1}^{m} 2^{-(h-1)k} P_{Mh}. \quad (9)$$

Parallel vertical-group calculation of the sum of squared differences is performed in m cycles, in each h-th cycle the following operations are performed:

– k partial results are formed as a result of squaring P_{jh} according to formula (5);

– calculation of group partial results P_{jh} according to formula (6);

– calculation of macro-partial result P_{Mh} by adding group partial results Q_{jh} according to formula (7);

– calculation of the final result is carried out by summing macro-partial results P_{Mh} using the following expression

$$Y_h = 2^{-(h-1)k} Y_{h-1} + P_{Mh},$$

where $Y_0 = 0$.

The proposed method of parallel vertical-group calculation of the sum of squared differences provides a significant increase in the speed of calculations by combining vertical processing organization with group parallelism. Formalization of the squaring operation in the form of a system of partial, group and macro-partial results allows for the effective implementation of multi-bit multi-operand operations under strict time constraints.

The method of parallel vertical-group search for maximum and minimum values in a one-dimensional array of numbers. Parallel vertical-group search for maximum $X_{\max}$ and minimum $X_{\min}$ values in a one-dimensional $\{X_k\}_{h=1}^{N}$ array of numbers is performed in m cycles, where $m = \left\lceil \frac{n}{k} \right\rceil$, n is the number of digits of numbers, k is the number of digits in the group. In each h cycle ((h=1,...,m) there is a parallel receipt of vertically organized slices of k digits of all N numbers, starting from the highest. In each h-th cycle, the search for the maximum $X_{\max}$ and minimum $X_{\min}$ values in a one-dimensional array according to this method is based on the execution of k basic macro operations of the same type. Each r-th basic macro operation (r = 1, ..., k) is implemented based on the sequence of the following logical operations:

1) the value of the bits $x_{hr\max}$ and $x_{hr\min}$ is determined using the following formulae:

$$\overline{x_{hr\max}} = \bigwedge_{j=1}^{N} \overline{x_{jhr} \wedge y_{jhr}}, \quad (10)$$

$$x_{hr\min} = \bigwedge_{j=1}^{N} \overline{x_{jhr} \wedge z_{jhr}}. \quad (11)$$

where x_{jhr} – value of hr-th digit of j-th number of array, y_{jhr}, z_{jhr} – value of j-th digit of hr-th control word respectively for determination of maximum and minimum values, first control word equals $y_{j1} = z_{j1}$=1, j = 1,...,N;

2) formation of j-th digit of (hr + 1)-th control words is performed using the following formulae:

$$y_{j(hr+1)} = (\overline{x_{hr\max}} \vee x_{hr}) \wedge y_{jhr}, \quad (12)$$

$$z_{j(hr+1)} = (x_{hr\min} \vee x_{hr}) \wedge z_{jhr}. \quad (13)$$

The peculiarity of the considered parallel vertical group method for finding the maximum (minimum) number is that in each h-th cycle of operation, k digits of the maximum $X_{\max}$ and minimum $X_{\min}$ values are determined, which ensures a significant reduction of the total calculation time.

The main advantages of the proposed method are:

– use of a single basic macro operation, which simplifies hardware implementation;

– the possibility of deep parallelization and pipelined calculations, which ensures high speed;

– simultaneous processing of k bit slices of N numbers ensures the massive-parallel nature of the processing and increases the computational density;

– the calculation time is mainly determined by parameters k and n, and not by the number of elements of array N, which enables good scalability when working with large data arrays.

5. 2. Improvement and specification of flow graphs for synthesizing parallel-stream real-time computational structures

To assess the computational and structural characteristics of the algorithm for calculating multi-operand neural operations, its representation in the form of a functional graph of the algorithm F = (Φ, Γ) is used, where $\Phi = \{\Phi_1, \Phi_2, ..., \Phi_n\}$ is a set of functional operators, Γ is the law of mapping the connections between operators. The algorithm graph describes the sequence and mutual dependence of calculations. Graphically, the functional graph of the algorithm is displayed in the form of vertices corresponding to the operators of the algorithm Φ_i and arcs reflecting the connections between operators. For the hardware implementation of algorithms for calculating multi-operand neural operations, there is a need for their spatial-temporal mapping at the level of elementary arithmetic-logical operations. Such an algorithm mapping should ensure the detection of all forms of parallelism and finding the necessary space-time solutions for creating real-time hardware tools with high efficiency of equipment use.

A spatiotemporal representation of algorithms with the detection of all forms of parallelism is enabled by the algorithm flow graph in a tier-parallel form. In this form of representation of the algorithm, all its functional operators Φ_i are distributed by tiers. In this case, the j-th tier contains functional operators that depend on at least one functional operator of the (j – 1)-th tier and do not depend on the operators of the following tiers. All functional operators of one tier are executed independently of each other.

The parameters of the algorithm flow graph in most cases depend on the input data, the dimensions of which determine the number of operations performed, that is, the number of functional operators. For real-time data processing, it is necessary to find such spatiotemporal solutions that will provide hardware implementation of the algorithm with high efficiency of equipment use. The necessary spatiotemporal solutions are achieved by changing the parameters of the flow graph (the degree of detail of functional operators, the height and width of the graph).

The synthesis of real-time parallel-stream structures is based on the use of the method of adequate hardware mapping of the structure of flow graphs of algorithms. According to this method, each functional operator is assigned a separate operational block, and the arcs between functional operators are assigned hardware-implemented data transmission channels, which ensures the continuity of the calculation flow and the possibility of effective conveyorization. The hardware synthesized in this way is algorithmic. In such structures, the specified algorithm is implemented when passing and processing data from inputs to outputs through all operational blocks.

For processing data streams, the most suitable are pipeline algorithmic structures, which provide high throughput and efficient use of hardware resources. Pipelining involves the decomposition of the algorithmic structure into a sequence of steps (tiers) by introducing buffer memory between individual stages of calculations. Each step of the pipeline contains two main components: buffer memory, which provides temporary storage of intermediate results and synchronization of data streams; operational devices that implement the corresponding functional operators of this tier.

To achieve high performance and maximum efficiency of hardware resources, functional operators implemented in the pipeline steps must be structurally simple. In addition, they

must have approximately the same execution time, which ensures uniform loading of the steps and minimizes pipeline downtime.

One of the main parameters of pipeline algorithmic structures is computational capacity, which is defined as follows

$$D_k = \frac{m_k n_k}{t_M + t}, \quad (14)$$

where t_M – buffer memory access time, t_O – calculation time of the most complex functional operator Φ_{jk} by the operational block, m_k – number of data input channels in the pipeline stages, n_k – bit depth of data input channels in the pipeline stages.

The main requirements for the synthesis of highly efficient algorithmic structures for processing data streams are to ensure real-time processing with minimal hardware costs. To ensure real-time processing of data streams arriving with an intensity $P_d = F_d m_v n_v$, where F_d – data input frequency, m_v – number of data input channels, n_v – bit depth of input channels, the following condition must be met

$$P_d \le D_k. \quad (15)$$

The synthesis of highly efficient real-time algorithmic structures requires the development of appropriate algorithms and flow graphs that must ensure the fulfillment of condition (15). This development process can be divided into the following four stages:

– decomposition of the algorithm for solving the problem;

– design of communications (data exchange) between functional operators;

– aggregation of functional operators;

– planning of calculations.

At the decomposition stage, the algorithm for solving the problem Φ is divided into functional operators Φ_i between which connections are established that correspond to this algorithm. The greater the degree of detail of the algorithm obtained as a result of decomposition, the more flexible the algorithm will be and the easier it is to adapt it to the fulfillment of condition (15). In practice, the synthesis of highly efficient real-time algorithmic structures uses the method of functional decomposition, in which the algorithm Φ is divided into operations Φ_i, each of which can be implemented by operational blocks of a certain level of hierarchy. The time and method of performing the Φ_i operation by the operational block are among the main parameters in determining the pipeline clock speed T_c and the bit depth of the data input channels n_k for real-time algorithmic structures. The result of the first stage of development is a graph diagram of the algorithm where functional operators Φ_i have approximately the same execution time, and their complexity is determined by the means of implementation.

At the stage of designing communications for the pipeline implementation of the algorithm, it is necessary to determine the structure of data exchange channels between functional operators Φ_i. To do this, a transition is made from the graph-scheme of the algorithm to a flow graph, in which the spatial-temporal placement and fixing of functional operators Φ_i by tiers is carried out. The structure of connections in the flow graph between functional operators Φ_{jk} of neighboring tiers determines the number of data input channels and the structure of connections between operational blocks during the hardware implementation of the algorithm.

According to the results of the first two stages of development, it is possible to estimate the computational capacity of the pipeline algorithmic structure D_k. In the case when the intensity of data input P_d coincides with the estimated throughput of the pipeline D_k, determined based on the analysis of the flow graph of the algorithm, the algorithmic device synthesized according to this graph provides high efficiency in the use of hardware resources.

If the throughput of pipeline D_k is less than the intensity of data arrival P_d, then to ensure the processing of data streams in real time, it is necessary to parallel enable algorithmic devices, the number of which is determined using the following expression

$$S = \left\lceil \frac{P_d}{D_k} \right\rceil, \quad (16)$$

where $\lceil\ \rceil$ – rounding sign to a larger integer.

In the case when $P_d < D_k$, to ensure high efficiency of equipment use it is necessary to proceed to the third stage of development – operation consolidation. At this stage, operations and data transmission channels are combined in the tiers of the flow graph. The algorithm graph that we obtain as a result of such a combination is a specified flow graph. The value of the combination coefficient R for real-time algorithmic devices is determined as follows

$$R = \left\lfloor \frac{D_k}{P_d} \right\rfloor, \quad (17)$$

where $\lfloor\ \rfloor$ is the rounding sign to a smaller integer.

The rounding stage is closely related to the planning stage where, after combining functional operators to preserve information about the structure of the algorithm's flow graph, calculations are planned, delays and data permutations are determined. To reproduce calculations, delay control and data permutation operators are introduced into each tier of the specified graph.

5. 3. Basic parallel-flow structures for the implementation of multi-operand neural operations in real time and estimation of their parameters

The development of parallel-flow devices with parallel-vertical calculation of basic multi-operand neural operations should be carried out on the basis of an integrated approach. This approach combines the capabilities of a modern element base (FPGA, ASIC, SoC), vertical calculation methods and algorithms, parallel-flow architectures, and the requirements for applied neural-oriented real-time systems.

In order to fully use the potential of modern hardware platforms, it is suggested to adhere to the following basic development principles:

– the multi-operand principle, which involves the simultaneous processing of multiple operands and ensures a significant reduction in the time of performing neural operations;

– the principle of vertical parallelism, which consists in the simultaneous processing of bit slices of all operands and the formation of macro-partial results;

– calculation pipeline, which provides deep temporal parallelization and maximum throughput;

– modularity of construction, which consists in forming a device from functionally completed unified pipeline steps;

– localization of calculations and minimization of intermodular connections, which makes it pos-

sible to reduce signal propagation delays and power consumption;

– regularity and structural uniformity, which ensures effective synthesis and tracing on FPGA and ASIC;

– coordination of data arrival time with calculation time, which guarantees continuous operation of the pipeline without downtime;

– adaptation of the architecture to the algorithmic structure of neural operation, which makes it possible to minimize hardware costs and latency.

The set of these principles enables the development of a hardware structure optimized according to the criteria of speed, energy efficiency, scalability, and regularity.

The synthesis of structures of real-time parallel-stream devices is carried out on the basis of specified flow graphs of algorithms for calculating multi-operand neural operations. Depending on the intensity of data flow P_d, various variants of structures of parallel-streaming devices can be synthesized, which differ in both the organization of calculations and technical parameters.

Since the task of fully describing all possible structures is combinatorially complex and practically insoluble, it is advisable to isolate and study generalized basic parallel-stream structures for the implementation of multi-operand neural operations. Such basic structures serve as architectural templates on the basis of which efficient specialized real-time computing devices can be synthesized, optimized for specific applications and given requirements for performance, latency, and hardware costs.

The basic structure of a parallel-stream device with vertical-group dot product calculation. The basic structure of a parallel-stream device with vertical-group dot product calculation is shown in Fig. 1, where DFC is a data format converter, TI_w is a write clock, TI_c is a pipeline clock, Rg is a register, Ad is an adder, BFGPP is a block for forming group partial products, PPS is a partial product shaper, PS is a pipeline step, *kAd* is a *k*-input adder.

The device consists of a DFC data format converter and m identical stages of the PS pipeline. The number of stages is determined by ratio $m = \left\lceil \frac{n}{k} \right\rceil$, where n is the bit depth of the input data X_j, k is the number of bits of multipliers X_j, which are analyzed to calculate the group partial products P_{jh}.

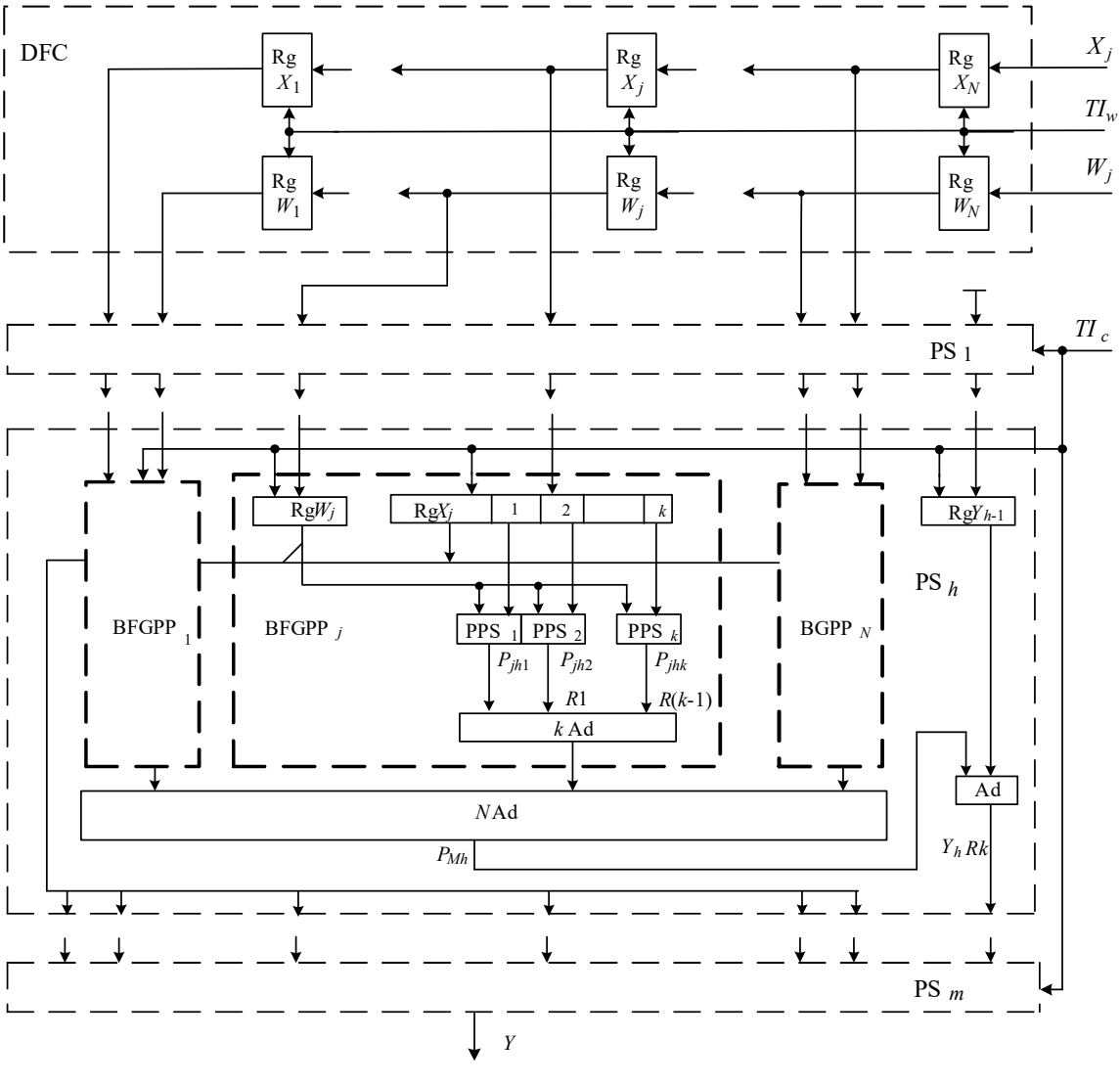

Fig. 1. Basic structure of a parallel-streaming device with vertical-group scalar product computation

The input data X_j and the weighting coefficients W_j are fed to the device sequentially through two independent channels with a clock TI_w and are written to registers RgX_1,..., RgX_N and RgW_1,..., RgW_N. After N TI_w clocks, the data from the outputs of registers RgX_1,..., RgX_N and RgW_1,..., RgW_N are written to the corresponding registers RgX_1,..., RgX_N and RgW_1,..., RgW_N of the first stage of the PS_1 of pipeline by the clock pulse TI_c.

The device operates on the pipeline principle with a cycle time TI_c, which, to ensure real time, must be $TI_k \le N \times TI_1$. In each h-th cycle of operation ($h = 1, ..., m$) in registers RgW_1,..., RgW_N, RgX_1,..., RgX_N and RgY_{h-1} of the h-th pipeline stage PS_h, data from outputs of the (h – 1)-th pipeline stage PS_{h-1} are written. In the pipeline stage PS_h for the h-th group of multiplier bits X_{jh1}, X_{jh2}, ..., X_{jhk} at outputs PPS_1,..., PPS_k, k partial products are formed in accordance with $P_{jhs} = W_j X_{jhs}$, where ($s = 1,...,k$). The formed partial products are fed to the input of the k-input adder kAd, and the s-th partial product $W_j X_{jhs}$ is shifted relative to the (s – 1)-th partial product $W_j X_{jh(s-1)}$ by one digit to the right.

By adding the partial products at the output of the k-input adder kAd, the group partial product P_{jh} is obtained. The calculated group partial product P_{jh} is fed to the j-th input of the N-input adder NAd, at the output of which we obtain the h-th macro-partial product P_{Mh}. The calculated h-th macro-partial product P_{Mh} is fed to the input of adder Ad, where it is added to the (h – 1)-th partial result Y_{h-1}. The result of calculating the first scalar product is obtained at the output of the device after the m-th clock cycle. In each subsequent cycle, the output forms the result of calculating the next scalar product, which ensures continuous pipeline operation.

The latency of the pipeline step PS, which simultaneously determines the period of the pipeline cycle, is calculated using the following formula

$$T_{\text{kPS}} = t_{\text{Rg}} + t_{\text{AND}} + t_{k\text{Ad}} + t_{N\text{Ad}} + t_{\text{Ad}}, \qquad (18)$$

where t_{Rg}, t_{AND}, $t_{k\text{Ad}}$, $t_{N\text{Ad}}$, t_{Ad} – the operation time of register Rg, the logical element AND, k-input adder kAd, N-input adder NAd and adder Ad, respectively.

The equipment costs for the implementation of this device are determined using the following expression

$$W_{\text{SMU}} = 2NW_{\text{Rg}} + m\left[\begin{matrix} N\left(2W_{\text{Rg}} + kW_{\text{PPS}} + W_{k\text{Ad}}\right) + \\ + W_{\text{NAd}} + W_{\text{Ad}} + W_{\text{Rg}} \end{matrix}\right], \qquad (19)$$

where W_{Rg}, W_{PPS}, $W_{k\text{Ad}}$, W_{NAd}, W_{Ad} – equipment costs, respectively, for register Rg, the partial product shaper PPS, k-input adder kAd, N-input adder NAd and adder Ad.

The basic structure of a parallel-stream device with vertical-group calculation of the sum of squares of differences is shown in Fig. 2. The device includes DFC – data format converter, PRS – partial result shaper, GPRS – group partial result shaper, Su – subtractor, Rg – register, kAd – k-input adder, NAd – N-input adder, Ad – adder. The input and output signals are TI_w – data recording clock pulses, X_j^e and X_j^b – data inputs, TI_c – pipeline clock pulses, Y – output of the sum of squares of differences. The number of pipeline steps (PS) is determined during the design of a specific device.

The main components of the parallel-stream device for vertical-group calculation of the sum of squared differences are the data format converter DFC and m of the same type of stages of pipeline PS. The input data X^e and X^b are supplied to the device sequentially by two independent channels with a cycle TI_w. In the data format converter DFC, at each cycle TI_w, the $\Delta X_j = X_j^e - X_j^b$, difference is calculated, which is sequentially written to registers RgX_1,..., RgX_N. After N cycles TI_w, the data from the outputs of registers RgX_1,..., RgX_N are written to registers RgX_1,..., X_N of the first stage of pipeline PS_1 by the clock pipeline pulse TI_c.

In the device, the calculation of the sum of squared differences is carried out with a cycle TI_c, which, to ensure real time, must be $TI_k \le N \times TI_1$. In each h-th stage of pipeline PS_h, the h-th iteration of the algorithm for vertical group calculation of the sum of squares of differences is implemented by hardware. In each j-th group partial result shaper $GPRS_j$, the bits x_{j1}, x_{j2}, ..., $x_{j[(h-1)k]r}$ are input to the r inputs of the partial result shaper PRS_r. At output r of the partial result shaper PRS_r, we obtain the partial result of the square using the following expression

$$P_{jhr} = (0.x_{j1}x_{j2}...x_{j[(h-1)k]r-1}01) \wedge x_{j[(h-1)k]r}. \qquad (20)$$

The formed partial result of raising to the square P_{jhr} is supplied with a right shift of r-1 digit to the r-th input of the k-input adder kAd. At the output of the k-input adder kAd we obtain the group partial result P_{jh}. The group partial results are supplied to the inputs of the N-input adder NAd, at the output of which we obtain the macro-partial result P_{Mh}. At adder Ad the macro-partial result P_{Mh} is added to the previously accumulated macro-partial results P_{Mh} using expression $Y_h = 2^{-(h-1)k} Y_{h-1} + P_{Mh}$, where $Y_0 = 0$.

The latency of the pipeline step PS, which simultaneously determines the period of the pipeline clock, is calculated using the following formula

$$T_{k\text{PC}} = t_{\text{Rg}} + t_{\text{AND}} + t_{k\text{Ad}} + t_{N\text{Ad}} + t_{\text{Ad}2n}, \qquad (21)$$

where t_{Rg}, t_{AND}, $t_{k\text{Ad}}$, $t_{N\text{Ad}}$, t_{Ad} is the operation time of register Rg, logical element AND, k-input adder kAd, N-input adder NAd, and adder Ad, respectively.

The equipment costs for the implementation of the device for calculating the sum of squares of differences are determined using the following expression

$$W_{\text{SSU}} = W_{\text{Su}} + NW_{\text{Rg}} + m\left[\begin{matrix} N\left(W_{\text{Rg}} + kW_{\text{PRS}} + W_{k\text{Ad}}\right) + \\ + W_{\text{NAd}} + W_{\text{Ad}} + W_{\text{Rg}} \end{matrix}\right], \qquad (22)$$

where W_{Su}, W_{Rg}, W_{PRS}, $W_{k\text{Ad}}$, W_{NAd}, W_{Ad} – equipment costs, respectively, for subtractor Su, register Rg, the partial result shaper PRS, the k-input adder kAd, the N-input adder NAd, and adder Ad.

The basic structure of a parallel-stream device with vertical-group search for maximum and minimum values in a one-dimensional array of numbers. The basic structure of a parallel-stream device with vertical-group search for maximum and minimum values in a one-dimensional array of numbers is in Fig. 3, where TI_w – data recording clock pulses, X_j – data input, TI_c – pipeline clock pulses, DFC – data format converter, Rg – register, Tr – trigger, PS – pipeline step.

The parallel-stream device for vertical-group search of maximum and minimum values operates on the pipeline principle and is implemented on the basis of a data format converter DFC, m pipeline result registers and m identical stages of pipeline PS. Each PS_h hardware implements k basic macro operations for calculating the maximum and

minimum numbers in a one-dimensional data array. Each r-th basic macro operation ($r = 1,...,k$) is implemented on the basis of logical operations: determining the value of r bits of the maximum $x_{hr\max}$ and minimum $x_{hr\min}$ values according to formulae (10), (11), and forming the bits of control words $y_{j(hr+1)}$ and $z_{j(hr+1)}$ according to formulae (12), (13). In the h-th stage of pipeline PS_h, k bits of the maximum $X_{\max}$ and minimum $X_{\min}$ values are determined, which are written to the h-th pipeline registers Rg$hX_{\max}$ and Rg$hX_{\min}$, respectively. Simultaneous processing of k bit slices in PS_h provides massively parallel processing and increases processing efficiency. The time for searching for maximum and minimum values is mainly determined by parameters k and n, and not by the number of elements in array N.

The latency of pipeline stage PS, which simultaneously determines the period of the pipeline clock, is calculated using the following formula

$$T_{\kappa\text{MMU}} = t_{\text{Rg}} + 3kt_{\text{AND}}, \quad (23)$$

where t_{Rg}, t_{AND} is the activation time of register Rg, the logical element AND, respectively.

The equipment costs for implementing a parallel-streaming device with vertical-group search for maximum and minimum values in a one-dimensional array of numbers are determined using the following expression

$$W_{\text{MMU}} = NW_{\text{Rg}} + 2W_{\text{Rg}} + Nm\left(2W_{\text{Tr}} + W_{\text{Rg}} + 6kW_{\text{AND}}\right), \quad (24)$$

where W_{Rg}, W_{Tr}, W_{AND} – equipment costs, respectively, for register Rg, trigger Tr, and the logical element AND.

Estimation of the parameters of parallel-streaming devices for vertical-group computing of multi-operand neural operations. The main functional units on the basis of which parallel-streaming devices for vertical-group computing of multi-operand neural operations are synthesized are registers, triggers, adders, a subtraction unit, and logical elements. Since the developed structures are oriented towards the FPGA implementation, a logic gate (inverter, AND, OR type element) was chosen as the unit of calculation of equipment costs, and for the estimation of time parameters – the value of delay of the logical gate τ. Equipment costs for the implementation of functional units in gates and their speed are given in Table 1.

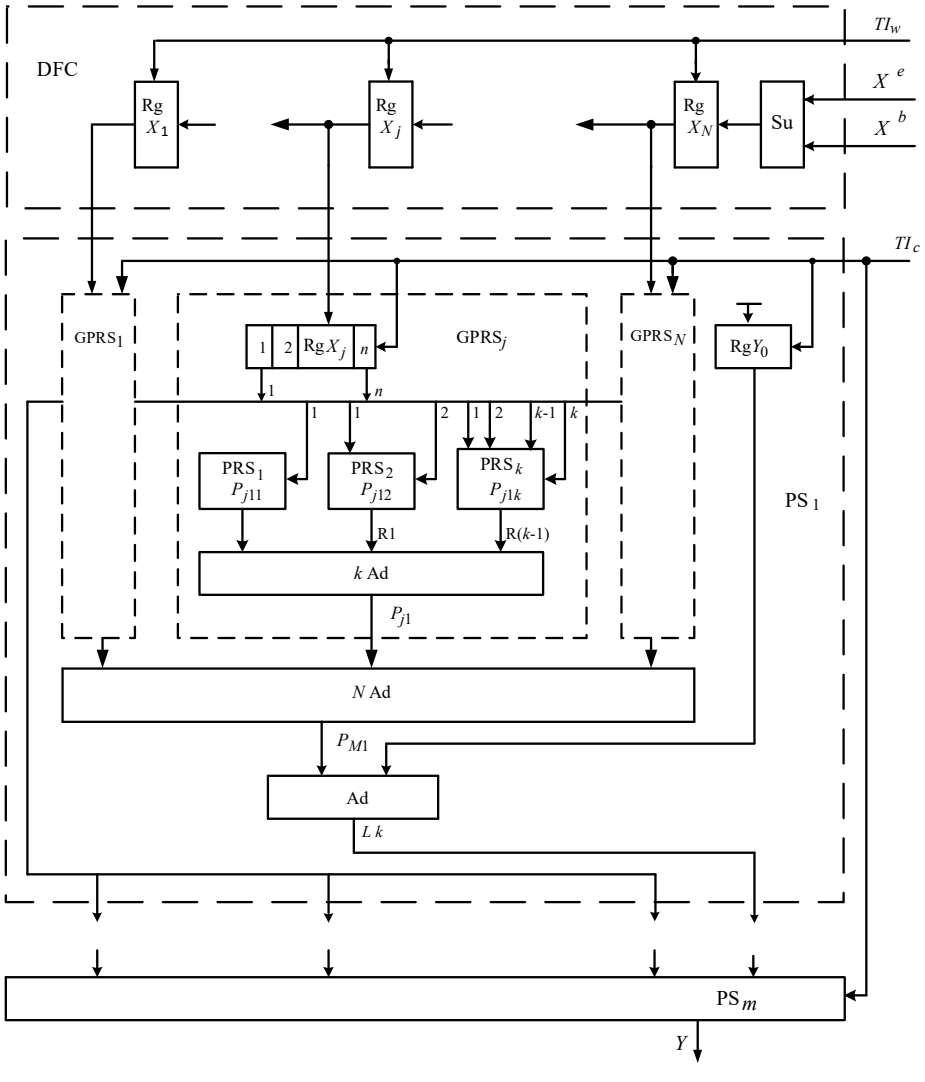

Fig. 2. Basic structure of a parallel-streaming device with vertical-group calculation of the sum of squares of differences

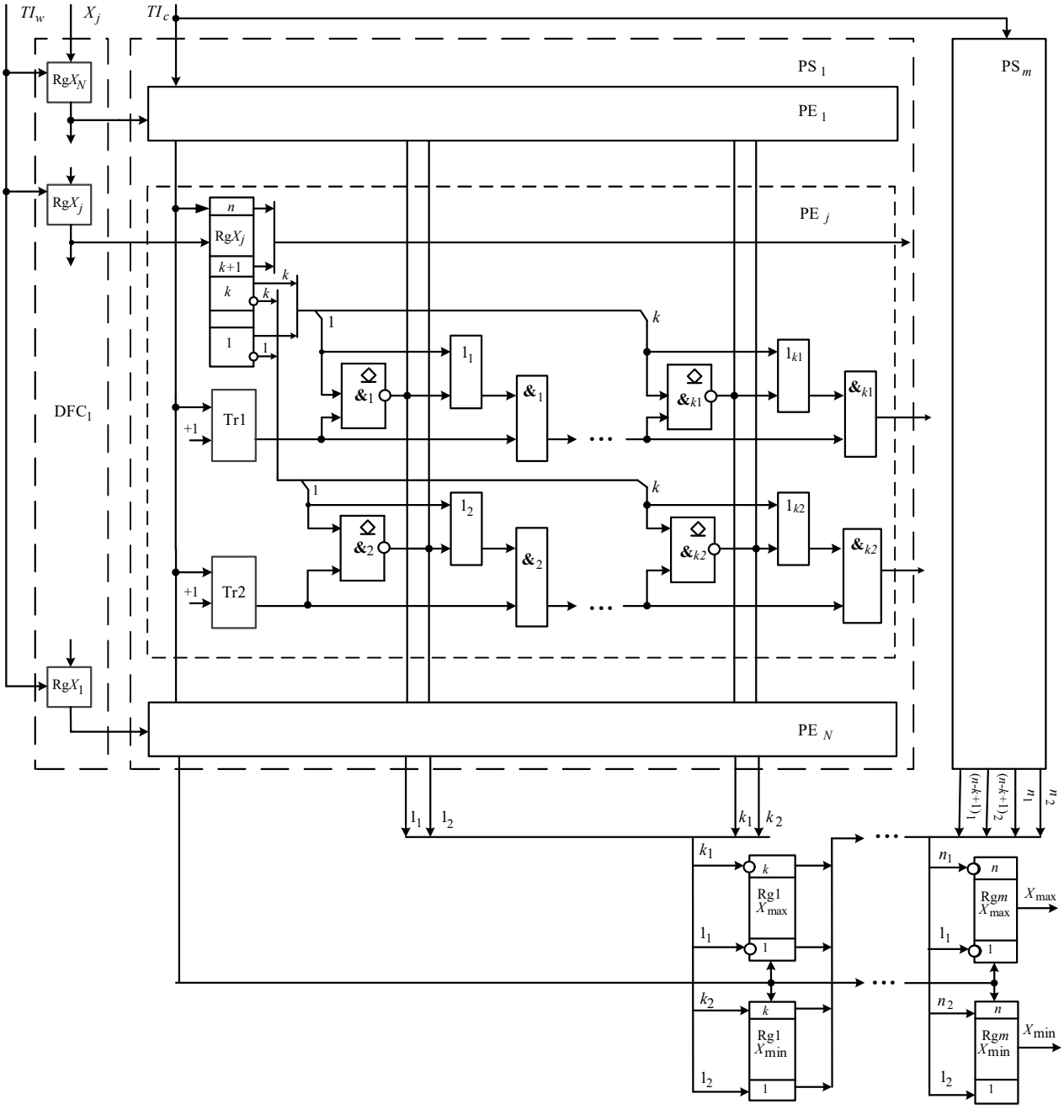

Fig. 3. Basic structure of a parallel-streaming device with vertical-group search for maximum and minimum values in a one-dimensional array of numbers

Table 1

Equipment costs for implementing the functional units and their performance

No.	Name of functional units	Equipment costs (gates)	Number of delay stages (gate τ)
1	trigger	6	3
2	n-bit register	$7n$	3
3	n-bit adder	$20n$	$7\log_2 n$
4	n-bit subtractor	$20n$	$7\log_2 n$
5	N-input n-bit adder	$(N-1)\,20n$	$7\log_2 n \log_2 N$

To estimate the equipment costs in gates (logic elements) for the implementation of each of the designed basic parallel-stream devices of vertical-group calculation of multi-operand and neural operation, a generalized analytical expression is formed. Such an expression is defined as the sum of the hardware costs of all functional units that are part of the device. Taking into account the features of the developed basic structures of parallel-stream devices, the analytical dependences for estimating the equipment costs take the following form:

1) device for vertical-group calculation of the scalar product

$$W_{\text{SMU}} = 14Nn + \frac{n}{k}\left(21Nkn + 14Nn + 7n\right); \quad (25)$$

2) device for vertical group calculation of the sum of squared difference

$$W_{\text{SSU}} = 7Nn + 20n + \frac{n}{k}\left(21Nkn + 7Nn + 7n\right); \quad (26)$$

3) device for vertical-group search for maximum and minimum values in a one-dimensional array of numbers

$$W_{\text{MMU}} = 7Nn + 14n + \frac{n}{k}\left(6Nk + 7Nn + 12N\right). \quad (27)$$

The pipeline clock (clock period) is determined by the maximum latency of the pipeline steps of the designed parallel-stream devices for vertical-group computation of multi-operand neural operation. Analytical expressions for estimating the duration of the pipeline cycle of the corresponding devices take the following form:

1) device for vertical-group computation of the scalar product

$$T_{\text{kSMU}} = 4 + 7\log_2 n\left(\log_2 k + \log_2 N + 1\right); \quad (28)$$

2) vertical group sum of squared difference calculation device

$$T_{\text{kSSU}} = 4 + 7\log_2 n\left(\log_2 k + \log_2 N\right) + 7\log_2 2n; \quad (29)$$

3) device for vertical-group search for maximum and minimum values in a one-dimensional array of numbers

$$T_{\text{kMMU}} = 3 + 3k. \quad (30)$$

To evaluate the developed structures of parallel-streaming devices for vertical group computing of multi-operand neural operations, it is proposed to use the criterion of efficiency of hardware resource utilization, which connects the complexity of algorithms with the duration of their execution and hardware costs. Quantitatively, the efficiency of using hardware resources is determined using the following expression

$$E_{\text{MNO}} = \frac{R_{\text{MNO}}}{t_{\text{MNO}} W_{\text{MNO}}}, \quad (31)$$

where R_{MNO} – complexity of algorithm of calculation of multi-operand neural operation, t_{MNO} – time of calculation of multi-operand neural operation, W_{MNO} – hardware costs for realization of parallel-streaming device of vertical-group calculation of multi-operand neural operation.

Analytical expressions for estimating the efficiency of using hardware resources of designed parallel-streaming devices take the following form:

1) device of vertical-group calculation of scalar product

$$E_{\text{SMU}} = \frac{N + Nn}{\left[4 + 7\log_2 n\left(\log_2 k + \log_2 N + 1\right)\right]\left[14Nn + \frac{n}{k}\left(21Nkn + 14Nn + 7n\right)\right]}; \quad (32)$$

2) device for vertically grouping the sum of squared differences:

$$E_{\text{SSU}} = \frac{2N + Nn}{\left[4 + 7\log_2 n\left(\log_2 k + \log_2 N\right) + 7\log_2 2n\right]\left[7Nn + 20n + \frac{n}{k}\left(21Nkn + 7Nn + 7n\right)\right]}, \quad (33)$$

3) device for vertically grouping maximum and minimum values in a one-dimensional array of numbers

$$E_{\text{MMU}} = \frac{Nn}{\left(3 + 3k\right)\left[7Nn + 14n + \frac{n}{k}\left(6Nk + 7Nn + 12N\right)\right]}. \quad (34)$$

The parameters of the designed parallel-stream devices for vertical group calculation of the scalar product, sum of squared differences, and search for maximum and minimum values in a one-dimensional array of numbers were evaluated. The calculations were performed for the number of operands $N = 16$, the data bit depth $n = 24$, and the following values of the vertical grouping parameter (number of bits in the group): $k = 1, 2, 3, 4, 6, 8, 12$.

The plots of hardware costs for implementing the designed parallel-stream devices, indicated in the number of gates, are shown in Fig. 4, where W_{SMU}, W_{SSU} and W_{MMU} are the hardware costs for implementing devices for the scalar product, sum of squared differences, and search for maximum and minimum values in a one-dimensional array of numbers, respectively.

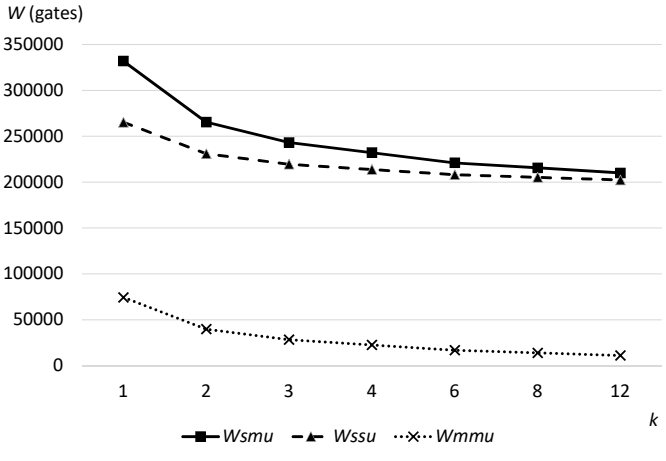

Fig. 4. Plots of hardware cost for implementing parallel-streaming devices for vertically grouped computation of the scalar product, sum of squared differences, and finding maximum and minimum values in a one-dimensional array of numbers

Analysis of Fig. 4 reveals that with an increase in the vertical grouping parameter k, the hardware costs for implementing all designed parallel-stream devices decrease. This is explained by the fact that with an increase in k, the number of pipeline stages m decreases, and therefore the number of registers, adders, and other functional units in the device structures decreases.

At the same time, the most significant decrease in hardware costs is observed for small values of k, while with a further increase in k, the reduction effect gradually decreases, which is associated with the complexity of functional units that process larger groups of bits.

It was also established that the highest hardware costs are for the scalar product calculation device, where multiplication and multi-level summation operations are implemented. The device for calculating the sum of squared differences has comparable but somewhat lower costs, while the lowest hardware costs are for the maximum and minimum value search device, which is based mainly on logical operations.

The plots of duration of the pipeline cycle of the designed parallel-stream devices, expressed in the number of gates τ, are shown in Fig. 5, where T_{SMU}, T_{SSU} and T_{MMU} are the pipeline cycle of the devices for the scalar product, sum of squared differences, and search for maximum and minimum values in a one-dimensional array of numbers, respectively.

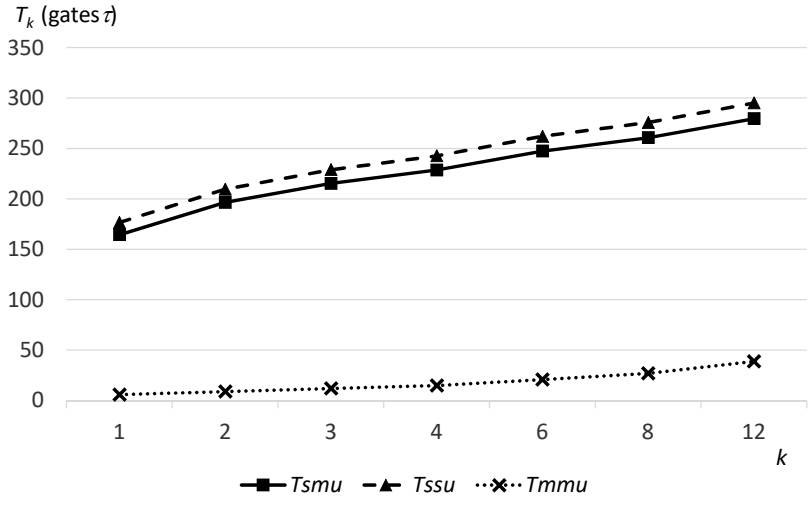

Fig. 5. Plots of duration of the pipeline cycle of parallel-streaming devices for vertically grouping the scalar product, the sum of squared differences, and finding the maximum and minimum values in a one-dimensional array of numbers

Analysis of Fig. 5 reveals that increasing the vertical grouping parameter k leads to an increase in the duration of the pipeline cycle of the designed devices. This is due to the increase in the complexity of calculations within one pipeline step because at larger values of k a larger number of bits are processed simultaneously, which leads to an increase in the critical path delay.

The longest duration of the pipeline cycle is for the device for calculating the scalar product, which is due to the implementation of complex arithmetic operations of multi-level summation of partial products. The pipeline cycle is somewhat shorter for the device for calculating the sum of squares of differences, which is explained by the implementation of operations of multi-level summation of partial results of raising to the square. The shortest duration of the pipeline cycle is for the device for searching for maximum and minimum values because it is based mainly on simple logical operations and does not require complex arithmetic calculations.

Thus, the increase in parameter k is accompanied by an increase in the duration of pipeline cycle, which must be taken into account when choosing the optimal value of k, taking into account the compromise between hardware costs and time characteristics of the device.

The plots of efficiency of using hardware resources for the designed parallel-stream devices for vertical-group calculation of the scalar product and the sum of squares of differences are shown in Fig. 6. In it, E_{SMU} and E_{SSU} are the efficiency of using hardware resources for devices, respectively, of the scalar product and the sum of squares of differences.

For a parallel-stream device for vertically grouping the scalar product, the maximum efficiency of using hardware resources is achieved at $k = 6$, while for a device for calculating the sum of squares of differences, it is achieved at $k = 1$ (Fig. 6). This is explained by the differences in the structural organization of calculations and the relationship between hardware costs and the duration of pipeline cycles.

The efficiency plot of hardware resources utilization by the designed parallel-stream device for searching for maximum and minimum values in a one-dimensional array of numbers is shown in Fig. 7.

As revealed by our analysis, for a parallel-stream device for searching for maximum and minimum values in a one-dimensional array of numbers, the maximum efficiency of hardware resource use is achieved at $k = 4$. This is due to the optimal ratio between hardware costs and the duration of the pipeline cycle for this type of operations, which are based mainly on logical transformations.

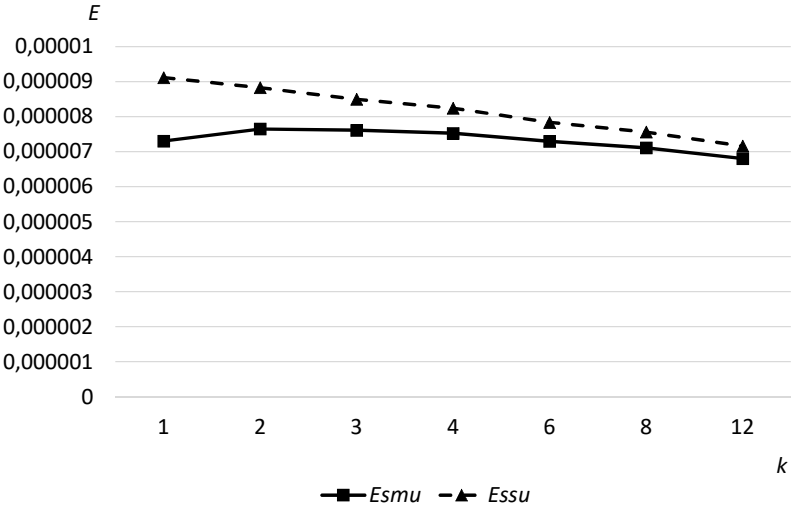

Fig. 6. Efficiency plots of hardware resources utilization for the designed parallel-stream devices for the vertical-group calculation of the scalar product and the sum of squares of the differences

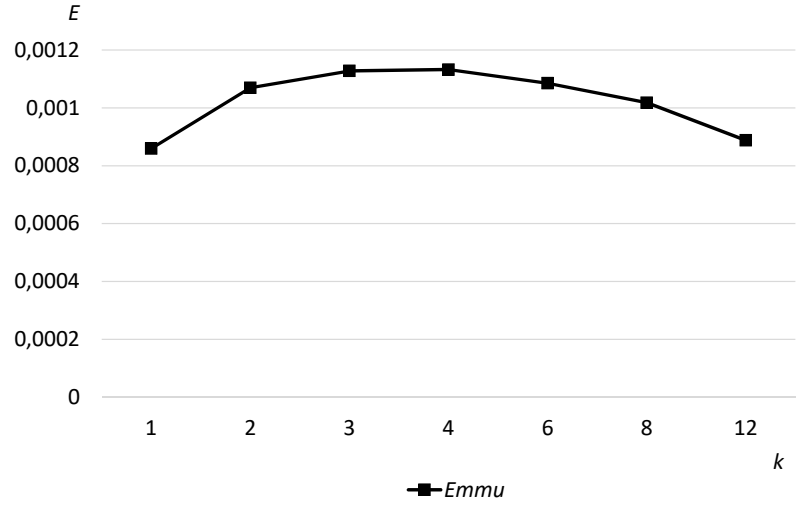

Fig. 7. Efficiency plot of hardware resources utilization by the designed parallel-stream device for searching for maximum and minimum values in a one-dimensional array of numbers

5. 4. Defining a set of requirements and improvement of the information technology for synthesizing computational structures for performing multi-operand neural operations in real time

Improvement of the information technology for synthesizing parallel-stream structures of vertical-group computing of multi-operand neural operations in real time requires defining a set of requirements that form its functional capabilities. These requirements should provide a systematic, formalized, and reproducible process of transition from algorithmic description of neural operations to effective hardware implementations.

The basic requirements for the information technology that synthesizes multi-operand parallel-stream computational structures are:

– formalization and systematicity of synthesis – ensuring a formalized description of all stages of synthesis, from the decomposition of neural operations to the formation of an implementation description of hardware structures;

– support for multi-operand neural operations – focus on computational structures for implementing complex operations (scalar product, sum of squared differences, group summation, etc.);

– parallel-stream and vertically-parallel organization of calculations – the ability to simultaneously process groups of bit slices and form macro-partial products;

– coordination of algorithmic and hardware parameters – synchronization of data flow intensity, pipeline depth, degree of parallelism, and clock frequency to enable real-time mode;

– parameterization and scalability – support for adapting structures by the number of operands, data bit depth, pipeline depth, and level of parallelism;

– optimization of hardware and time costs – minimizing resource use and delays when processing multi-operand operations;

– focus on hardware implementation – use of modern elemental FPGA, ASIC, and system on a chip (SoC).

The information technology for synthesizing multi-operand parallel-stream computing structures is based on a hierarchical synthesis model, which includes the following interconnected levels:

– a functional level at which neural-oriented calculations are represented as a set of basic multi-operand neural operations;

– an algorithmic level that provides the selection and parameterization of parallel-stream and vertically-parallel computing algorithms;

– a structural level at which the synthesis of regular hardware structures is performed taking into account pipeline and parallelism;

– an implementation level focused on FPGA implementation.

The improvement of information technology for synthesizing multi-operand parallel-stream computing structures involves the implementation of a clear sequence of interconnected stages that provide a formalized transition from the algorithmic description of neural-oriented calculations to hardware-implementation structures with guaranteed time characteristics.

The structure of such information technology in the form of a block diagram is shown in Fig. 8.

The improvement of information technology is based on a gradual transition from an algorithmic description of neurocomputation to an implementation description of a specialized hardware structure, taking into account real-time requirements and hardware limitations.

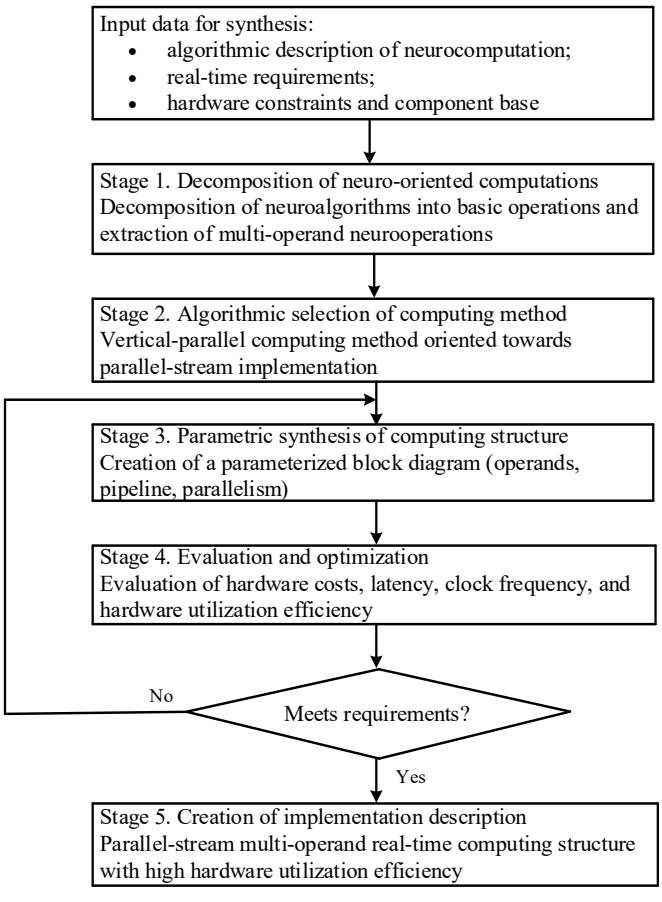

Fig. 8. Flowchart of information technology for synthesizing multi-operand parallel-stream computing structures for real-time neural-oriented systems

The input data of the information technology for synthesizing multi-operand parallel-stream computing structures are:

– a description of a neural-oriented problem or a neural network model, which includes a computation graph $G = (V, E)$, where V is a set of vertices corresponding to basic operations or neural operations, E is a set of information connections that determine data flows between them and are oriented towards the synthesis of parallel-stream hardware structures;

– a set of basic multi-operand neural operations $V_O = \{V_{O1}, V_{O2}, ..., V_{Ok}\}$, among which the scalar product $Z = \sum_{j=1}^{N} W_j X_j$, the sum of squares of differences $Y = \sum_{j=1}^{N} \left(X_j^e - X_j^b \right)^2$, as well as the operations of finding the maximum $Y_{\max} = \max X_j$, $j = 1,...,N$ and the minimum $Y_{\min} = \min X_j$, $j = 1,...,N$, which are characteristic of neural-oriented real-time systems;

– number of operands N and bit depth n of data;

– constraints on execution time $T_k \le T_o$, where T_k is the pipeline cycle, T_o is the exchange time;

– constraints on hardware resources $W_p \le W_{\max}$, where W_p is the hardware costs for implementing the device, $W_{\max}$ is the maximum allowable hardware costs;

– constraints on power consumption $P_p \le P_{\max}$, where P_p is the power consumption of the device, $P_{\max}$ is the maximum allowable power consumption.

At the first stage, the decomposition of neural-oriented calculations is performed, which consists in the decomposition of the neural algorithm into a set of basic computational operations. From this set, multi-operand neural operations ($N > 2$) are distinguished, which are decisive in terms of computational complexity and real-time requirements. The decomposition of neural algorithms is performed taking into account the following criteria:

– the number of operands N in the neural operation;

– the frequency of execution of the operation in the computational graph;

– the impact of the operation on the overall latency of calculations;

– the suitability of the operation for vertical-parallel and parallel-stream implementation.

As a result of the decomposition, a set of multi-operand neural operations is formed, for which it is advisable to apply specialized methods of vertical-parallel computing and subsequent parametric synthesis of parallel-stream hardware structures.

At the second stage, an algorithmic selection of the calculation method is carried out, which involves determining the method for implementing the obtained multi-operand neural operations, which ensures the fulfillment of real-time requirements with minimal use of hardware resources. The choice of the method is based on the results of the decomposition of neural-oriented calculations and the analytical assessment of the parameters of multi-operand neural operations. When choosing a calculation method for multi-operand neural operations, it is advisable to take into account the following criteria:

– the execution time of operations required to ensure real-time;

– parallelism, which determines the ability to simultaneously process a group of operands;

– the efficiency of using hardware resources, which involves minimizing hardware resources while ensuring real-time;

– scalability, which ensures an increase in the number of operands without a significant increase in latency.

Existing conventional sequential and pipeline calculation methods do not always provide the necessary speed and efficient use of hardware resources in real-time systems. To implement multi-operand neural operations, it is advisable to use a vertically parallel method, which involves simultaneous processing of bit slices of a group of operands. The use of this method provides for the following:

– reduced latency – the calculation occurs due to the simultaneous execution of operations on bit slices of all operands;

– high performance – parallel processing ensures significant acceleration of multi-operand operations compared to sequential methods;

– savings in hardware resources – due to a reduction in the number of intermediate registers and logical nodes required for intermediate data storage;

– scalability and versatility – the method is easily adapted to a different number of operands and types of neural operations.

Thus, the vertically parallel method provides an optimal combination of high performance and efficient use of hardware resources, making it the most appropriate for calculating multi-operand neural operations in real-time systems.

At the third stage, a parametric synthesis of a parallel-stream computing structure is implemented, which involves quantitative determination of the parameters of the operands, the depth of the pipeline and the degree of parallelism. Within the framework of this stage, the formation of a parameterized structural scheme of the hardware implementation of the vertically parallel calculation of a multi-operand neural operation is ensured, which guarantees the fulfillment of the time constraints of the real-time neural-oriented system with minimal hardware and energy costs.

When implementing the parametric synthesis of a parallel-stream computing structure, it is advisable to take into account the following criteria:

– fulfillment of real-time time constraints – ensuring the specified latency, requirements for the pipeline cycle $T_k \leq T_o$;

– degree of parallelism of calculations – the number of simultaneously used computational channels and the level of spatial-temporal parallelization of multi-operand neural operations;

– depth and balance of the pipeline – optimal distribution of computational operations between the stages of the pipeline with minimization of the critical path;

– efficiency of use of hardware resources – minimization of the number of logical elements, registers and connections while maintaining the required performance;

– energy efficiency – reduction of power consumption of the computational structure due to the optimal choice of parallelism and pipeline parameters;

– scalability of the structure – the ability to adapt the computational structure to changes in the number of operands, data bit size and performance requirements without significantly complicating the hardware implementation;

– regularity and structural uniformity – formation of regular hardware structures suitable for effective implementation on FPGA and ASIC.

The result of the third stage is a parameterized structural diagram of a multi-operand parallel-stream computing structure, which:

– formalizes the relationship between the number of operands, bit depth, pipeline depth, and level of parallelism;

– ensures the fulfillment of real-time requirements;

– is the basis for further structural-hardware and implementation synthesis on FPGA, ASIC, or SoC.

At the fourth stage, the synthesized parallel-stream computing structure is evaluated and optimized for vertically parallel execution of multi-operand neural operation in order to ensure that its parameters meet real-time requirements and hardware limitations. At this stage, the cost of hardware resources, operation execution time, and equipment efficiency are evaluated. At the same time, the pipeline parameters, the degree of parallelism, and the organization of operand processing are optimized to ensure minimal hardware and time costs while maintaining the specified performance.

A number of parameters are used to evaluate the synthesized parallel-stream computing structure. Hardware costs are calculated in the number of gates, and the operation execution time is calculated in the number of gate delay cascades when data passes through the longest calculation path (latency). The efficiency of hardware use in this case is defined as the ratio of performance to hardware costs, which makes it possible to us to assess the contribution of each gate to the overall performance.

The result of the fourth stage is an evaluated and optimized parallel-stream computing structure for vertically parallel execution of multi-operand neural operations, which ensures compliance of hardware costs, operation execution

time, and equipment utilization efficiency with real-time requirements.

At the fifth stage, an implementation description of the synthesized parallel-streaming multi-operand computing structure is formed, which ensures the vertical-parallel execution of multi-operand neural operations. Within the framework of this stage, the following steps are performed:

– transformation of the parameterized structural diagram into an implementation description suitable for specific hardware platforms (FPGA, ASIC, SoC);

– creation of pipeline and parallel blocks in accordance with the specified parameters of the pipeline depth and the degree of parallelism;

– optimization of the interconnections of functional nodes to minimize delays and hardware costs;

– preparation of a description for synthesis, tracing, and verification, which ensures compliance with time constraints and resource characteristics.

The result of the fifth stage is an implementation description of the parallel-streaming multi-operand computing structure for the vertical-parallel execution of neural operations in real time. The resulting structure is optimized for hardware cost, latency, and hardware utilization efficiency, and is ready for synthesis on FPGA, ASIC, or SoC.

6. Discussion of results related to improving the information technology for synthesizing the structures for calculating multi-operand neural operations in real time

Improving the information technology of synthesis of parallel-stream computing structures for vertical-group execution of multi-operand neural operations in real time required the development of a number of appropriate methods.

The devised method of parallel vertical-group calculation of multi-operand neural operations is intended for determination of the scalar product, sum of squares of differences, and search for maximum and minimum values in a one-dimensional array. The main advantages of the devised method are achievement of high level of parallelism of calculations and an increase in the efficiency of use of hardware resources, which is provided by the integrated combination of vertical and group parallelism. As follows from formulas (2), (9) to (13), calculation of such multi-operand neural operations is performed by simultaneous processing of groups of bit slices, which makes it possible to reduce number of steps $m=\frac{n}{k}$. This is directly reflected in plots of hardware costs (Fig. 4), where with increase of k the corresponding decrease of number of steps is observed. At the same time, as shown by formulae (28) to (30) and plots in Fig. 5, increase in k leads to increase of duration of pipeline cycle due to complication of calculations within one step. The generalized effect of these two opposing trends is reflected in formulae (32) to (34) and efficiency plots (Fig. 6, 7), where the presence of optimal values of parameter k (in particular, $k=6$ for the scalar product, $k=1$ for the sum of squared differences, $k=4$ for the search problem) is observed, which ensure maximum efficiency in the use of hardware resources.

Conventional approaches to implementing neural operations are based on two-operand or horizontally parallel computations [25–28], where processing is performed sequentially or with a limited level of parallelism. The proposed method provides simultaneous processing of a set of operands at the bit-slice level. This makes it possible to significantly reduce the number of calculation cycles (formulae ((2), (9) to (13)) and increase the throughput. Known FPGA/ASIC solutions [8, 9, 17, 18] are focused on the implementation of individual neural operations without coordination of data flow parameters. The proposed approach implements formalized coordination of the data flow intensity with the pipeline parameters (condition (15)), which provides a guaranteed real-time mode. The advantages of the proposed solution are a high degree of parallelism due to the vertical-group organization of calculations; the possibility of parametric optimization (through the choice of k); regularity and modularity of structures, which simplifies hardware implementation; increased efficiency of hardware resource use (Fig. 6, 7).

The advantages of the proposed method of stream graph synthesis are the formation of a specified stream graph of the algorithm based on a step-by-step approach, which includes algorithm decomposition, communication design, functional operator aggregation, and calculation planning. In particular, algorithm decomposition makes it possible to represent the computational process in the form of a set of functional operators $\{\Phi_1, \Phi_2, ..., \Phi_n\}$, which creates prerequisites for detecting all forms of parallelism. This is directly reflected in the formation of a stream graph in a tiered-parallel form, where operators of one tier can be executed independently. Designing communications between operators ensures the determination of the structure of data transmission channels, which affects the parameters of the pipeline and the coordination of the intensity of data flows (condition (15)). The aggregation of functional operators (the union coefficient R) makes it possible to adapt the graph structure to real-time requirements, which is confirmed by the reduction of hardware costs (Fig. 4) while maintaining acceptable time characteristics (Fig. 5). Computational planning with the introduction of delay and permutation operators ensures the correct spatial-temporal mapping of the algorithm and the continuity of pipeline processing. As a result, a specified flow graph is formed, which is the basis for the synthesis of an effective parallel-flow structure.

In known methods of constructing computational structures [22, 27], flow graphs are used mainly to describe algorithms without taking into account the parameters of data flows. The proposed method provides a formalized specification of graphs taking into account the intensity of data flow and the parameters of the pipeline. This makes it possible not only to describe the algorithm but also directly proceed to its hardware implementation. Unlike approaches focused on a static structure of calculations, the proposed method makes it possible to adapt the algorithm graph by enlarging operators and planning calculations. This provides a balance between the depth of the pipeline and the complexity of operations and makes it possible to achieve high efficiency in the use of hardware resources (Fig. 6, 7). The advantages of the proposed method are formalized transition from the algorithm to the hardware structure; the ability to match the intensity of data flows with the parameters of the pipeline; ensuring a high level of parallelism and regularity of structures; adaptability to various types of multi-operand neural operations.

The designed basic parallel-stream computing structures are used to synthesize efficient computing structures based on an integrated combination of the multi-operand principle, vertical parallelism, conveyorization, and coordination of computing intensities and data arrival. In particular, the use

of vertical-group computing makes it possible to process k-bit slices simultaneously, which reduces the number of pipeline stages m and, accordingly, hardware costs (formulae (25) to (27), Fig. 4).

At the same time, as follows from formulae (28) to (30) and confirmed by the plots in Fig. 5, the duration of pipeline cycle is determined by the complexity of calculations within one step and remains relatively stable or changes slightly when k changes because the critical path is formed by a limited number of functional nodes. The increase in the efficiency of using hardware resources (formulae (32) to (34), Fig. 6, 7) is explained by the simultaneous reduction in hardware costs and the preservation of acceptable time characteristics, which ensures an improvement in the "performance/resources" ratio.

Unlike conventional computational structures for implementing neural operations, which are based on two-operand operations and sequential or limitedly parallel summation [25, 28, 30], the proposed structures provide simultaneous processing of a set of operands within one cycle. This makes it possible to reduce latency and increase throughput. Unlike existing FPGA/ASIC solutions, where the architecture is often rigidly tied to a specific algorithm [8, 17–19], the designed structures are parameterized and scalable, which allows them to be adapted to different values of N, n, and k. This is made possible by the modular organization and the use of unified pipeline steps. In addition, unlike approaches without formalized data flow coordination, the proposed structures ensure coordination of the data flow intensity with the pipeline bandwidth (condition (15)), which guarantees real-time operation.

The improvement of information technology for synthesizing parallel-stream computing structures for vertical-group execution of multi-operand neural operations in real time described in the study uses a formalized approach. The specified approach involves a phased multi-level implementation of such structures, which ensures a coordinated transition from algorithmic description to hardware implementation taking into account the parameters of the computing process. In particular, at the algorithmic level stage, the vertical-group calculation method is used (formulae (2), (9) to (13)), which determines the structure of calculations and the degree of parallelism. At the structural level, a specified flow graph is formed, which provides a spatial-temporal representation of the algorithm and determines the parameters of the pipeline. The coordination of the intensity of data arrival with the throughput of the computing structure is implemented through the fulfillment of condition (15), which guarantees the real-time mode. The evaluation results (formulas (25) to (34), Table 1, Fig. 4–7) confirm that the proposed information technology makes it possible to achieve a reduction in hardware costs, maintain an acceptable duration of the pipeline cycle, and increase the efficiency of resource use due to the optimal choice of parameters k, pipeline depth, and degree of parallelism.

Unlike known approaches to the synthesis of computational structures, where the transition from algorithm to hardware implementation is carried out without formal consideration of data flow parameters [22, 27], the proposed information technology ensures their coordination at all stages of synthesis. This makes it possible to obtain structures that are guaranteed to satisfy real-time requirements. Unlike existing FPGA/ASIC solutions focused on the implementation of individual neural operations [8, 9, 17, 18], the proposed technology provides a universal approach to the synthesis of structures for various multi-operand operations. This becomes possible due to the hierarchical synthesis model and parameterization of computational structures. Thus, the advantages of the developed information technology for synthesis are formalized and reproducible synthesis process; coordination of algorithm and hardware implementation parameters; ensuring high performance with minimal hardware costs; the ability to adapt to different classes of neural operations and application requirements.

A limitation of our research is the assumption about the possibility of parallelization and pipelined data processing operations. However, in the case of neural networks and neural-like systems that involve the implementation of multi-operand neural operations, it should not be considered unfounded. The architectures of these data processing systems have a regular structure and, therefore, provide the potential for both parallelization and pipelined operations.

Further studies may consider the design of automated tools for synthesizing parallel-stream structures based on the proposed technology and expanding the technology to complex neural network models and hybrid computing structures. Devising multi-criteria optimization methods could make it possible to take into account energy characteristics and adapt to uneven and stochastic data flows, which would provide experimental implementation and verification on FPGA/ASIC. Advancing these areas is appropriate because it could increase the practical significance and versatility of information synthesis technology, as well as ensure its effective application in modern neural-oriented real-time systems.

7. Conclusions

1. A method for parallel vertical-group computation of basic multi-operand neural operations (scalar product, sum of squared differences, and finding the maximum and minimum values in a one-dimensional array) has been devised. Due to vertical (simultaneous processing of the corresponding bits of all operands) and group parallelism (parallel execution of computations over k-bit groups) and the optimal choice of parameter k, high efficiency of hardware resource use is ensured, combined with speed and hardware optimization. A feature of the devised method for computing the scalar product and sum of squared differences is the use of hierarchical reduction of partial results, which provides fast folding of the results of each k-bit group using tree-like or pipeline structures. This significantly increases the computation speed and optimizes the use of hardware resources by reducing the volume of intermediate operations. The peculiarity of the devised method for finding the maximum (minimum) number is the use of a single basic macro operation, deep parallelization, and pipelined search process. This approach provides scalability for working with large data sets and reduces the computation time, which is determined mainly by parameters k and n, and not by the number of N array elements.

2. A method for stream mapping of the algorithm graph for calculating basic multi-operand neural operations has been improved, which provides concretization of the graph in accordance with the requirements for the application and coordination of the intensity of data receipt with the intensity of their processing. Such concretization of the graph is carried out by means of algorithm decomposition, communication

design, enlargement of functional operators, and calculation planning. Representation of the algorithm in the form of a concretized and coordinated stream graph provides an optimal distribution of functional operators by tiers. This allows for their independent and simultaneous execution, and pipelining using buffer memory and structurally simple operators contributes to high processing intensity and uniform loading of hardware resources. Hardware mapping of the concretized and coordinated flow graph provides a parallel-stream computing structure for processing data streams in real time with high efficiency of hardware use.

3. Basic parallel-stream computing structures have been designed for the implementation of multi-operand neural operations (scalar product, sum of squared differences, and search for maximum and minimum values in a one-dimensional array) in real time. These structures are based on an integrated combination of a multi-operand approach, vertical parallelism, conveyorization, modular organization, and coordination of computational intensities and data input. The designed structures are characterized by regularity, scalability, as well as orientation towards hardware implementation based on FPGA and serve as the basis for the synthesis of parallel-stream computing tools with given parameters. Analytical expressions have been derived for estimating hardware costs, pipeline cycle time, and resource efficiency in basic parallel-stream computing structures, which provide the possibility of a reasonable choice of computational structure parameters at the stage of their synthesis. According to the results of the evaluation of the designed basic parallel-stream computing structures, it was found that with an increase in the vertical group parameter k, hardware costs are significantly reduced due to a reduction in the number of pipeline stages. The duration of the pipeline cycle under the specified conditions remains practically unchanged while the efficiency of hardware resource use increases.

4. A set of requirements for the information technology of synthesis of parallel-stream computing structures, which form the functional capabilities and ensure its improvement, has been defined. The use of a systematic, formalized, and reproducible process of transition from algorithmic description of neural operations to effective hardware implementations is the basis for improving the corresponding information technology. The improved technology for synthesizing parallel-stream computing structures for the implementation of multi-operand neural operations in real time is based on a phased multi-level transition from a vertical-group algorithm to hardware implementation. In the process of this transition, the parameters of the computing process (intensity of data flows, pipeline depth, and degree of parallelism) are coordinated with hardware constraints, which enables the achievement of the specified time characteristics with minimal resource consumption. The use of advanced information synthesis technology provides increased productivity and efficiency of hardware resource use, reduced latency, as well as the implementation of highly efficient parallel-stream computing structures for performing multi-operand neural operations in real time.

Conflicts of interest

The authors declare that they have no conflicts of interest in relation to the current study, including financial, personal, authorship, or any other, that could affect the study and the results reported in this paper.

Funding

The study was carried out at the Lviv Polytechnic National University within the framework of research work "Methods and means for intelligent measurement of movement parameters and determining spatial orientation of ground mobile robotic platforms" (State registration number 0124U000822).

Data availability

All data are available in the main text of the manuscript.

Use of artificial intelligence

The authors acknowledge that artificial intelligence tools were used during the preparation of the manuscript to check grammar, spelling, and punctuation without changing the text, as well as to partially search for sources published over the last 5 years, using keywords and criteria entered by the authors. The results were verified by the authors. For these purposes, GPT-5.3 from OpenAI was used. No part of the manuscript was generated or modified using artificial intelligence tools. The authors bear full responsibility for the final version of the manuscript.

Authors' contributions

Ivan Tsmots: Conceptualization, Methodology, Writing – original draft; **Vasyl Teslyuk**: Methodology, Formal analysis, Investigation, Supervision; **Yurii Opotyak**: Validation, Investigation, Project administration, Writing – review & editing; **Bohdan Shtohrinets**: Formal analysis, Visualization.

References

1. Izonin, I., Tkachenko, R., Hovdysh, N., Berezsky, O., Yemets, K., Tsmots, I. (2025). Cascade-Based Input-Doubling Classifier for Predicting Survival in Allogeneic Bone Marrow Transplants: Small Data Case. Computation, 13 (4), 80. https://doi.org/10.3390/computation13040080
2. Tsmots, I., Teslyuk, V., Łukaszewicz, A., Lukashchuk, Y., Kazymyra, I., Holovatyy, A., Opotyak, Y. (2023). An Approach to the Implementation of a Neural Network for Cryptographic Protection of Data Transmission at UAV. Drones, 7 (8), 507. https://doi.org/10.3390/drones7080507
3. Juracy, L. R., Garibotti, R., Moraes, F. G. (2023). From CNN to DNN Hardware Accelerators: A Survey on Design, Exploration, Simulation, and Frameworks. Foundations and Trends® in Electronic Design Automation, 13 (4), 270–344. https://doi.org/10.1561/1000000060
4. Deng, B. L., Li, G., Han, S., Shi, L., Xie, Y. (2020). Model Compression and Hardware Acceleration for Neural Networks: A Comprehensive Survey. Proceedings of the IEEE, 108 (4), 485–532. https://doi.org/10.1109/jproc.2020.2976475

5. Mohaidat, T., Khalil, K. (2024). A Survey on Neural Network Hardware Accelerators. IEEE Transactions on Artificial Intelligence, 5 (8), 3801–3822. https://doi.org/10.1109/tai.2024.3377147
6. Tsmots, I., Teslyuk, V., Kryvinska, N., Skorokhoda, O., Kazymyra, I. (2022). Development of a generalized model for parallel-streaming neural element and structures for scalar product calculation devices. The Journal of Supercomputing, 79 (5), 4820–4846. https://doi.org/10.1007/s11227-022-04838-0
7. Xie, Y., Oniga, S. (2024). A Comprehensive Review of Hardware Acceleration Techniques and Convolutional Neural Networks for EEG Signals. Sensors, 24 (17), 5813. https://doi.org/10.3390/s24175813
8. Wang, Z. (2025). Accelerating Transformer Models: FPGA-Based Hardware Optimization and Heterogeneous Computing Strategies. Applied and Computational Engineering, 138 (1), 86–92. https://doi.org/10.54254/2755-2721/2025.21360
9. Zeng, K., Ma, Q., Wu, J. W., Chen, Z., Shen, T., Yan, C. (2022). FPGA-based accelerator for object detection: a comprehensive survey. The Journal of Supercomputing, 78 (12), 14096–14136. https://doi.org/10.1007/s11227-022-04415-5
10. Tsmots, I., Teslyuk, V., Opotyak, Y., Mamchur, T., Oliinyk, O. (2025). Synthesis of recursive-type neural elements with parallel vertical-group data processing. Eastern-European Journal of Enterprise Technologies, 3 (2 (135)), 6–16. https://doi.org/10.15587/1729-4061.2025.329139
11. Geng, S., Wang, Z., Liu, Z., Zhang, M., Zhu, X., Dan, Y. (2025). Hardware implementation of FPGA-based spiking attention neural network accelerator. PeerJ Computer Science, 11, e3077. https://doi.org/10.7717/peerj-cs.3077
12. Carpegna, A., Savino, A., Carlo, S. D. (2025). Spiker+: A Framework for the Generation of Efficient Spiking Neural Networks FPGA Accelerators for Inference at the Edge. IEEE Transactions on Emerging Topics in Computing, 13 (3), 784–798. https://doi.org/10.1109/tetc.2024.3511676
13. Gong, Y., Xu, Z., He, Z., Zhang, W., Tu, X., Liang, X., Jiang, L. (2022). N3H-Core. Proceedings of the 2022 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, 112–122. https://doi.org/10.1145/3490422.3502367
14. Tsai, T.-H., Ho, Y.-C., Sheu, M.-H. (2019). Implementation of FPGA-based Accelerator for Deep Neural Networks. 2019 IEEE 22nd International Symposium on Design and Diagnostics of Electronic Circuits & Systems (DDECS), 1–4. https://doi.org/10.1109/ddecs.2019.8724665
15. Kachris, C. (2025). A Survey on Hardware Accelerators for Large Language Models. Applied Sciences, 15 (2), 586. https://doi.org/10.3390/app15020586
16. Silvano, C., Ielmini, D., Ferrandi, F., Fiorin, L., Curzel, S., Benini, L. et al. (2025). A Survey on Deep Learning Hardware Accelerators for Heterogeneous HPC Platforms. ACM Computing Surveys, 57 (11), 1–39. https://doi.org/10.1145/3729215
17. Kang, B. J., Lee, H. I., Yoon, S. K., Kim, Y. C., Jeong, S. B., O, S. J., Kim, H. (2024). A survey of FPGA and ASIC designs for transformer inference acceleration and optimization. Journal of Systems Architecture, 155, 103247. https://doi.org/10.1016/j.sysarc.2024.103247
18. Bjerge, K., Schougaard, J. H., Larsen, D. E. (2021). A scalable and efficient convolutional neural network accelerator using HLS for a system-on-chip design. Microprocessors and Microsystems, 87, 104363. https://doi.org/10.1016/j.micpro.2021.104363
19. Yan, Y., Ling, Y., Huang, K., Chen, G. (2023). An efficient real-time accelerator for high-accuracy DNN-based optical flow estimation in FPGA. Journal of Systems Architecture, 136, 102818. https://doi.org/10.1016/j.sysarc.2022.102818
20. Tasci, M., Istanbullu, A., Tumen, V., Kosunalp, S. (2025). FPGA-QNN: Quantized Neural Network Hardware Acceleration on FPGAs. Applied Sciences, 15 (2), 688. https://doi.org/10.3390/app15020688
21. Liu, Y., Yenamachintala, S. S., Li, P. (2019). Energy-efficient FPGA Spiking Neural Accelerators with Supervised and Unsupervised Spike-timing-dependent-Plasticity. ACM Journal on Emerging Technologies in Computing Systems, 15 (3), 1–19. https://doi.org/10.1145/3313866
22. Parashar, A., Raina, P., Shao, Y. S., Chen, Y.-H., Ying, V. A., Mukkara, A. et al. (2019). Timeloop: A Systematic Approach to DNN Accelerator Evaluation. 2019 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS), 304–315. https://doi.org/10.1109/ispass.2019.00042
23. Chen, Y.-H., Yang, T.-J., Emer, J. S., Sze, V. (2019). Eyeriss v2: A Flexible Accelerator for Emerging Deep Neural Networks on Mobile Devices. IEEE Journal on Emerging and Selected Topics in Circuits and Systems, 9 (2), 292–308. https://doi.org/10.1109/jetcas.2019.2910232
24. Jouppi, N. P., Young, C., Patil, N. et al. (2017) In-datacenter performance analysis of a tensor processing unit. arXiv. https://doi.org/10.48550/arXiv.1704.04760
25. Dhilleswararao, P., Boppu, S., Manikandan, M. S., Cenkeramaddi, L. R. (2022). Efficient Hardware Architectures for Accelerating Deep Neural Networks: Survey. IEEE Access, 10, 131788–131828. https://doi.org/10.1109/access.2022.3229767
26. Zhang, C., Li, P., Sun, G., Guan, Y., Xiao, B., Cong, J. (2015). Optimizing FPGA-based Accelerator Design for Deep Convolutional Neural Networks. Proceedings of the 2015 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, 161–170. https://doi.org/10.1145/2684746.2689060
27. Chen, T., Moreau, T., Jiang, Z. et al. (2018). TVM: an automated end-to-end optimizing compiler for deep learning. In Proceedings of the 13th USENIX Symposium on Operating Systems Design and Implementation (OSDI), 579–594. Available at: https://www.usenix.org/system/files/osdi18-chen.pdf
28. Sze, V., Chen, Y.-H., Yang, T.-J., Emer, J. S. (2017). Efficient Processing of Deep Neural Networks: A Tutorial and Survey. Proceedings of the IEEE, 105 (12), 2295–2329. https://doi.org/10.1109/jproc.2017.2761740
29. Jouppi, N. P., Hyun Yoon, D., Ashcraft, M., Gottscho, M., Jablin, T. B., Kurian, G. et al. (2021). Ten Lessons From Three Generations Shaped Google's TPUv4i : Industrial Product. 2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA), 1–14. https://doi.org/10.1109/isca52012.2021.00010
30. Mittal, S. (2018). A survey of FPGA-based accelerators for convolutional neural networks. Neural Computing and Applications, 32 (4), 1109–1139. https://doi.org/10.1007/s00521-018-3761-1