UDC 004.94
DOI: 10.15587/1729-4061.2025.329139

SYNTHESIS OF RECURSIVE-TYPE NEURAL ELEMENTS WITH PARALLEL VERTICAL-GROUP DATA PROCESSING

Ivan Tsmots
Corresponding author
Doctor of Technical Sciences*
E-mail: ivan.h.tsmots@lpnu.ua
Vasyl Teslyuk
Doctor of Technical Sciences*
Yurii Opotyak
PhD*
Taras Mamchur
PhD Student*
Oleksandr Oliinyk
PhD Student*
*Department of Automated Control Systems
Lviv Polytechnic National University
S. Bandery str., 12, Lviv, Ukraine, 79013

The object of this study is the processes of parallel vertical-group data processing and minimization of equipment costs, which enable the synthesis of real-time recursive neural elements with high efficiency of equipment use. A model of a recursive-type neural element has been built, which, through the use of a parallel vertical-group method for calculating the scalar product and the ability to choose the number of bits in the group for the formation of partial products, coordinates the time of receipt of weights and input data with the time of calculating the result at the output of the neural element. This approach provides a hardware implementation of the neural element with minimal use of equipment.

The basic structure of the neural element has been designed, which, through the use of hardware mapping of the constructed graph model, regularity, and modularity of the structure, provides the synthesis of hardware for a specific application. The application of pipelines and spatial parallelism of data processing, as well as the organization of the process of calculating the scalar product, as the performance of a single operation, enables the implementation of a neural element for real-time operation.

Analytical expressions have been built to estimate the parameters of a neural element depending on the bit depth of operands, the number of data inputs, and the number of bits in the group. A method for synthesizing a recursive-type neural element has been devised, which, due to the use of the basic structure, enables mechanisms for matching the time of receipt of weight coefficients and input data with the time of calculating the output, thus ensuring its implementation for specific applications. Considering ways to minimize equipment costs ensures the construction of a neural element with minimal hardware costs.

The synthesized neural element for a data depth of 16 bits with an increase in the number of bits that are simultaneously processed in a group, from 2 to 8, provides a decrease in the processing time by 2.8 times with a reduction in the efficiency of using the equipment of the neural element by no more than 1.6 times

Keywords: model of a neural element, real-time calculations, hardware implementation of a neural element

Received 01.04.2025
Received in revised form 13.05.2025
Accepted date 11.06.2025
Published date 30.06.2025

How to Cite: *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*

1. Introduction

The current stage of development of neural network technologies is characterized by the expansion of application areas, a significant part of which requires real-time processing. Such hardware and software tools must take into account data streams of different intensity and meet restrictions on dimensions and power consumption [1, 2]. Most existing neural network tools have a structural organization of a universal type, which is functionally and structurally redundant, does not take into account the requirements of specific applications for performance, dimensions, power consumption, and has low efficiency of equipment use.

Designing highly efficient real-time neural network tools requires the development of new models and structures of a neural element (NE) and requires the widespread use of a modern element base (processors, programmable logic integrated circuits (PLD), etc.). Such NEs should be oriented towards implementation in very-large-scale integration (VLSI) circuits, for example, FPGA. When designing a model and a basic VLSI-structure of a NE, it is necessary to ensure the ease of adapting the NE to the requirements of specific applications, real-time operation, and high efficiency of equipment use [2]. These requirements can be met by using recursive methods and NE structures with parallel vertical-group data processing.

Real-time mode when processing data streams of different intensity in a recursive-type NE is enabled by coordinating the time of incoming input data with the time of calculating the output signal of NE. This is achieved by using parallelization and pipelined calculation processes, choosing the number of bits in a group for forming partial products. The orientation of NE structure to VLSI implementation with high efficiency of equipment use requires a reduction in the number of interface pins and hardware costs while ensuring real-time mode [3].

The design of a recursive-type NE with parallel vertical-group data processing is most expedient to be carried out on the basis of an integrated approach. It covers the modern element base, models, and structures of NEs, as well as methods for calculating the scalar product, and takes into account the intensity of data flow and the requirements of specific applications.

Therefore, research into designing recursive-type NEs with parallel vertical-group data processing in real time is relevant.

2. Literature review and problem statement

Analysis of the means for implementing artificial neural networks (ANNs) reveals [4, 5] that the overwhelming majority of the means used to implement ANN are software. The main disadvantages of software means for implementing ANNs are low performance. The unresolved issue is the provision of processing intensive data streams, which can be implemented by hardware means. Increasing the performance of software means for implementing ANNs can be achieved by jointly using general-purpose processors supplemented by a graphics processor [6]. However, unresolved issues for such systems are compliance with the limitations on dimensions and power consumption. Papers [7, 8] show that for neural network processing of data streams in real time it is advisable to use hardware VLSI implementation, which simultaneously provides small dimensions and high performance.

Our analysis of [9, 10] reveals that the main components on the basis of which hardware neural networks are synthesized are NEs, but the issue of synthesizing the NE architecture taking into account the specified parameters remains unresolved. The characteristics of NEs largely depend on the approaches to the hardware implementation of the scalar product calculation operation. However, the issue of synthesizing such elements oriented to data stream processing with specified characteristics is not considered. In [11], approaches to the hardware implementation of the scalar product calculation operation are investigated, the first of which is based on the operations of multiplication, addition, and the second – on elementary arithmetic operations of addition, inversion, and shift. However, in these approaches, the issue of optimizing the structure of the device and calculating its time parameters and equipment costs remains unresolved.

In [12], the architectures of neural networks and their practical application are considered; various types of neural networks are reviewed; the latest achievements are summarized. However, the issue of hardware implementation of such networks remains unresolved, which requires the development of appropriate structures of neural elements. In [13] it is shown that the use of the basis of elementary arithmetic operations and the multi-operand approach, in which the calculation of the scalar product is considered as the execution of a single operation, provides optimization of the device structure in terms of speed and hardware costs. Our analysis reveals that the algorithms for calculating the scalar product using the multi-operand approach and the basis of elementary operations mainly use the direct formation of partial products with their subsequent addition. The disadvantage of existing algorithms for calculating the scalar product is the complexity of implementation; the unresolved issue is the coordination of the intensity of the receipt with the intensity of the calculation.

Our analysis of work [14] revealed that the hardware implementation of scalar product calculation algorithms using a multi-operand approach, and a basis of elementary arithmetic operations can be carried out using devices with a recursive and non-recursive structure. Analysis of non-recursive scalar product calculation devices shows that the structural feature of such devices is the absence of inverse connections, and calculations in such devices are performed when data passes from inputs to outputs through all operational modules. Non-recursive scalar product calculation devices are divided into two classes: the first is matrix, which use parallel formation and summation of all partial products, the second is parallel-stream, which use sequential formation and addition with a corresponding shift of partial products. The disadvantage of non-recursive scalar product calculation devices is the difficulty of matching the intensity of the input with the intensity of the calculation and the high hardware costs for their implementation. The disadvantage of neural network structures using recursive devices is the presence of inverse connections; an unsolved problem is the calculation of the scalar product in several iterations, the number of which is determined by the partial product formation algorithm.

In [15], a modified mathematical model of the Izhykevych neuron is described to provide a simple digital hardware implementation. However, the specified implementation, although it has reduced hardware costs, provides low computational intensity. The modeling and hardware implementation of the neural element is described in [16]. The reported neural digital circuit of the piecewise linear neuron (PLSN) model is built and implemented on PLIC. However, this implementation of the neural element is not used for streaming computing, but for modeling various types of behavior of brain neurons. In [17], a model of a neuron with a delay in pulse propagation is proposed to solve the problem of signal propagation in the reverse direction for a more accurate implementation of its biological function. However, this NE model is oriented towards hardware implementation in an analog way using capacitance charging models. The analyzed models of NEs differ in the techniques of input data and weighting coefficients, in the techniques for calculating the postsynaptic excitation signal. The disadvantage of these solutions is the use of a spike model of a neural element, in which information is encoded by the duration of discrete pulses as the main information carrier, which limits the performance of NE.

In [18], the implementation of a neural element in an analog format is proposed, which requires the use of a specialized element base; the issue of providing an interface with digital means of computing systems remains unresolved. In [19, 20], the issues of implementing the neural element activation function are considered, which, unlike the one presented, can be implemented by a tabular method. In [21], a generalized model of a neural element of the parallel-stream type is considered, the peculiarity of which is the orientation towards VLSI implementation. The disadvantage of the considered NE models is the difficulty of coordinating the time of data arrival with the time of calculation of the output signal of the neural element. In addition, an unresolved problem is the assessment of the neural element parameters, namely, equipment costs, speed, and efficiency of equipment use. These parameters largely depend on the bit size of the operands, the amount of data, and the number of bits in the group.

Therefore, the disadvantages of existing parallel-vertical NE models are the difficulty of matching the time of arrival of input data and weighting coefficients with the time of calculation of the output signal. Their implementation does not enable the minimization of equipment use when designing the structure of the neural element.

3. The aim and objectives of the study

The aim of our work is to synthesize neural elements of the recursive type with parallel vertical-group data processing in real time, which meet the requirements of specific applications and have high efficiency of equipment use.

To achieve this goal, the following main research tasks have been defined:

– to build a generalized model and basic structure of the recursive-type NE with parallel vertical-group data processing;

– to estimate parameters of the recursive-type NE with parallel vertical-group data processing;

– to devise a method for synthesizing the recursive-type NE with parallel vertical-group data processing in real time.

4. The study materials and methods

The object of our study is the processes of parallel vertical-group data processing and minimization of equipment costs, which enable the synthesis of neural elements of the recursive type in real time with high efficiency of equipment use.

The subject of the study is models, methods, algorithms, and recursive structures for the implementation of NEs with parallel vertical-group data processing in real time.

NEs of the recursive type with parallel vertical-group data processing were designed taking into account the minimization of hardware costs while ensuring the calculation of the scalar product and the activation function in real time. Enabling parallel vertical-group data processing in real time was carried out by coordinating the time of arrival of weight coefficients and input data with the time of calculation of the NE output. Coordination of the time of arrival of weight coefficients and input data with the time of calculation of NE output was carried out by selecting the number of bits in the group for forming partial products and by pipelining the process of summing group partial products.

The following research methods were used in our research:

– the method of serial-parallel transformation, the method of recursive parallel vertical-group calculation of the scalar product, and the method of parallel-time coordination – to build a generalized model of recursive-type NE with parallel vertical-group data processing;

– the theory of digital automata, methods of synthesis of neural networks of coordinated-parallel data processing in real time, methods of calculating the scalar product;

– methods for evaluating the main parameters of NEs in the process of designing computational structures.

5. Results of research on the development of a recursive-type neural element with parallel vertical group processing

5. 1. Generalized model and basic structure of a recursive-type neural element with parallel vertical group data processing

The principal components on the basis of which neural networks are synthesized are Nes that function according to the following formula

$$y = f\left(\sum_{j=1}^{N} W_j X_j\right), \tag{1}$$

where y is the output signal of NE, X_j is the input data, W_j is the weight coefficients, N is the number of data inputs and weight coefficients, f is the activation function, $j = 1, ..., N$. From formula (1) it follows that the operation of NE is based on the mathematical operations of calculating the weighted sum (scalar product) $Z = \sum_{j=1}^{N} W_j X_j$ and the activation function $f(Z)$.

When building the NE model, it is necessary to use methods of calculating the scalar product, which have mechanisms for coordinating the time of arrival of weight coefficients and input data with the time of calculations. Such methods include the method of recursive parallel vertical-group calculation of the scalar product, in which the calculation time depends on the number of digits in the groups that are analyzed to obtain partial products. Parallel vertical-group calculation of the scalar product is performed in accordance with the following formula

$$\begin{aligned} Z &= \sum_{j=1}^{N} W_j X_j = \\ &= \sum_{j=1}^{N}\sum_{g=1}^{m} 2^{-(g-1)k} \begin{pmatrix} W_j X_{j[(g-1)k+1} + \\ +2^{-(r-1)} W_j X_{j[(g-1)k+r} + \\ +\ldots + 2^{-(k-1)} W_j X_{j[(g-1)k+k} \end{pmatrix} = \\ &= \sum_{j=1}^{N}\sum_{g=1}^{m} 2^{-(g-1)k} P_{Gjg}, \end{aligned} \tag{2}$$

where k is the number of bits of the input data in the group, which are analyzed to obtain partial products; $r = 1, ..., k$; m is the number of groups, which is calculated as $m = \left\lceil \frac{n}{k} \right\rceil$; $\lceil\ \rceil$ – rounding sign to a larger integer, n is the number of bits of the input data; P_{Gjg} is the j-th group product for the g-th group of bits; $g = 1, ..., m$.

In formula (2), the sum of all j-th group products for the g-th group of bits can be replaced by the g-th macropartial result P_{Mg}, which is calculated as follows

$$P_{Mg} = \sum_{j=1}^{N} P_{Gjg}. \tag{3}$$

The calculation of the scalar product using the macropartial result P_{Mg} is performed according to the following formula

$$Z = \sum_{g=1}^{m} 2^{-(g-1)k} P_{\grave{I}\,g}. \tag{4}$$

For recursive calculation of the scalar product in accordance with formula (4) it is necessary to perform m cycles of calculations. In each g-th cycle of such calculation the following operations are performed:

– calculation for each j-th pair of operands of k partial products in accordance with formula $P_{jgr} = W_j X_{jgr}$;

– calculation of the j-th group product for the g-th group of bits in accordance with formula $P_{Gjg} = P_{jg1} + ... + 2^{-(r-1)} P_{jg2r} + ... + 2^{-(k-1)} P_{jg2k}$;

– calculation of the g-th macropartial result P_{Mg} in accordance with formula $P_{Mg} = \sum_{j=1}^{N} P_{Gjg}$;

– addition of the g-th macropartial result P_{gM} to the summation result, which is shifted to the right by k digits, in accordance with expression $Z_g = 2^{-k} Z_{g-1} + P_{gM}$, where $Z_0 = 0$.

The generalized analytical model of the recursive-type NE with parallel vertical-group data processing is written as

$$y = f_{a(Z)} \times \times \left(f_{BSZ_g} \times \times \left(f_{Z_g} \times \times \left(f_{BSP_{Mg}} \times \times \left(f_{P_{Mg}} \times \times \left(f_{(BSP_{G1g},\dots,BSP_{GNg})} \times \times \left(f_{(P_{G1g},\dots,P_{GNg})} \times \times \left(f_{(P_{1g1},\dots,P_{1gk}),\dots,(P_{Ng1},\dots,P_{Ngk})} \times \times \left(f_{(X_1 \to x_{1g}),\dots,(X_N \to x_{Ng})} \times \times \left(f_{(W_j \to NW, X_j \to NX)} \right) \right) \right) \right) \right) \right) \right) \right) \right), \qquad (5)$$

where y is the output of the recursive-type NE; $f_{a(Z)}$ is the activation function; f_{BSZ_g} – buffer storage of the g-th partial result of the scalar product; f_{Z_g} – calculation of the g-th partial result of the scalar product according to formula $Z_g = 2^{-k} Z_{g-1} + P_{gM}$, where $Z_0 = 0$; $f_{BSP_{Mg}}$ – buffer storage of the g-th macropartial result P_{Mg}; $f_{P_{Mg}}$ – calculation of the g-th macropartial result $P_{Mg} = \sum_{j=1}^{N} P_{Gjg}$; $f_{(BSP_{G1g},\dots,BSP_{GNg})}$ – buffer storage of the g-th group products; $f_{(P_{G1g},\dots,P_{GNg})}$ – calculation of the g-th group products $P_{Gjg} = P_{jg1} + \dots + 2^{-(r-1)} P_{jg2r} + \dots + 2^{-(k-1)} P_{jg2k}$; $f_{(P_{1g1},\dots,P_{1gk}),\dots,(P_{Ng1},\dots,P_{Ngk})}$ – formation of the g-th partial products in accordance with formula $P_{jgr} = W_j X_{jgr}$; $f_{(X_1 \to x_{1g}),\dots,(X_N \to x_{Ng})}$ – parallel-group transformation of input data; $f_{(W_j \to NW, X_j \to NX)}$ – serial-parallel transformation of weight coefficients and input data.

Based on the analytical model (5), a generalized graph model of recursive-type NE with parallel vertical-group data processing has been built, which is shown in Fig. 1.

The main components of the generalized graph model of the recursive-type NE with parallel vertical-group data processing are a serial-parallel converter of weight coefficients and input data $f_{(W_j \to NW, X_j \to NX)}$, N parallel-group converters of input data $f_{(X_1 \to x_{1g}),\dots,(X_N \to x_{Ng})}$, ($N \times k$) nodes for calculating partial products P_{jgr}, N k-input adders for calculating group partial products P_{Gjg}, N-input adder for calculating the g-th macropartial result P_{Mg}, a macropartial result adder $Z_g = 2^{-k} Z_{g-1} + P_{gM}$, an activation function calculator $f_{a(Z)}$.

To reduce the bit size of the macropartial result adder P_{Mg}, it is advisable to calculate the partial products P_{jgr} starting from the lower bits of the input data. A feature of the proposed model is the simultaneous calculation of k partial products for each j-th pair of operands, which provides a reduction in the number of cycles of calculating the scalar product by k times. One of the criteria for choosing the number of bits in the group for calculating partial products is to enable real-time data processing.

A feature of the generalized graph model of the recursive-type NE with parallel vertical-group data processing is enabling the combination in time of the execution of data input operations of the next array with N weight coefficients W_j and N input data X_j with the processing of the previous data array.

At the current stage of development of integrated technology, the main goal of designing appropriate data processing tools is to obtain a modular and regular structure oriented towards the VLSI implementation. The initial information for designing the basic structure of the recursive-type NE with parallel vertical-group data processing in real time is:

– generalized graph model of the recursive-type NE with parallel vertical-group data processing;

– algorithm of parallel vertical-group calculation of the scalar product;

– number of weight coefficients and input data N;

– intensity of the input data flow Pd;

– requirements for the NE interface, including the frequency of receiving and issuing information;

– bit depth of input data and accuracy of calculations;

– technical and economic requirements and limitations.

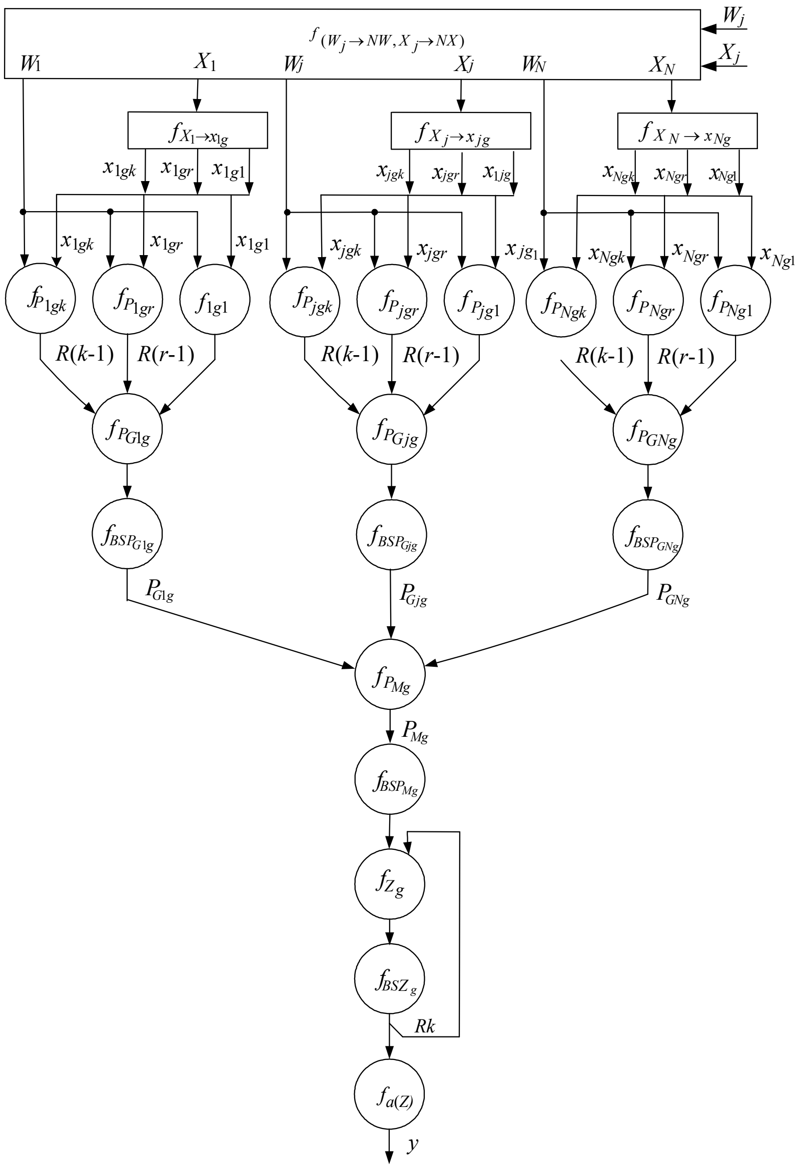

Fig. 1. Generalized graph model of a recursive-type neural element with parallel vertical group data processing

Using the graph model of NE (Fig. 1), the basic structure of the recursive-type NE with parallel vertical-group data processing was designed. The basic recursive-type NE structure with parallel vertical group data processing was designed according to the following principles:

– hardware mapping of the constructed graph model of NE;

– ensuring regularity and modularity of the structure;

– enabling a balance between the duration of I/O and calculation operations;

– extensive use of pipelines and spatial parallelism for data processing;

– performing calculations based on elementary arithmetic operations – addition, inversion, and shift;

– calculating the scalar product as a single operation;

– reducing the number of outputs of the NE interface.

The basic structure of a recursive-type NE with parallel vertical-group data processing (Fig. 2) was built by hardware mapping of the generalized graph model of NE.

The process of recursive parallel-vertical calculation of the scalar product in NE can be divided into two stages.

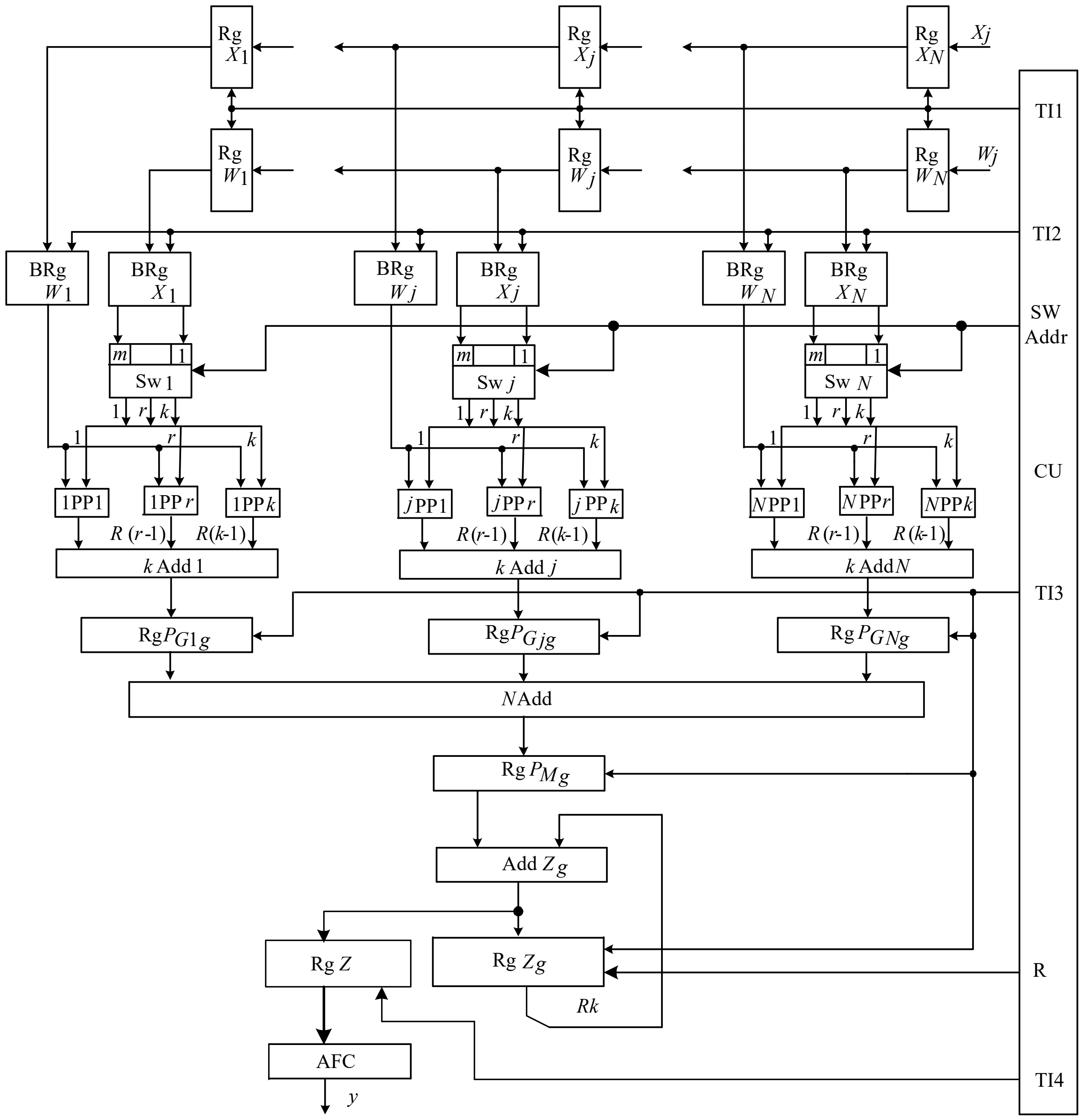

Fig. 2. The basic structure of a recursive-type neural element with parallel vertical group data processing, where TI1, TI2, TI3, and TI4 are the first, second, third, and fourth inputs of clock signals, respectively, SWAddr is the address for switches, R is the reset input, *y* is the output of NE. The designed structure includes: Rg – register, Sw – switch, BRg – buffer register, *k*Add – *k*-input adder, *N*Add – *N*-input adder, Add – adder. The components of NE are PP – partial product receiving node and AFC – activation function calculator. Control over NE operation is provided by control unit CU

At the first stage, a serial-parallel conversion of input data and weight coefficients is performed. In the process of such conversion, an array of N input data and N weight coefficients is sequentially written to registers RgX_1, ..., RgX_N and RgW_1, ..., RgW_N using clock pulses TI1. After N clock pulse TI1, an array of N input data X_1, ..., X_N and N weight coefficients W_1, ..., W_N will be stored in registers RgX_1, ..., RgX_N and RgW_1, ..., RgW_N, which are rewritten to buffer registers BRgX_1,..., BRgX_N and BRgW_1,..., BRgW_N by clock pulse TI2.

At the second stage, a clock-by-clock parallel-vertical calculation of the scalar product is performed.

In the first cycle of the calculation, the output of the switch Swj receives k lower bits of the input data X_j, which come from nodes jPP1,..,jPPk. At the outputs of nodes jPP1,...,jPPk , we receive partial products P_{j11},..., P_{j1k}, which, shifted to the right by (r–1) bits, are fed to the inputs of adder $k$Add$j$. At the output of this adder, we receive the group partial product P_{Gj1}, which is fed to the input of register RgP$_{Gjg}$. The leading edge of pulse TI3 writes the group partial products P_{Gj1} to registers RgP$_{Gjg}$.

In the second cycle of the calculation, switch Swj is set to the position when its output receives the second group of k subsequent bits of the input data X_j. The data from the output of switch Swj are fed to the inputs of nodes jPP1,..,jPPk, at the outputs of which we obtain the second partial products P_{j21},..., P_{j2k}. These partial products with a shift to the right by (r – 1) bits are fed to the inputs of adder $k$Add$j$, at the output of which we obtain the group partial product P_{Gj2}. The first group partial products P_{Gj1} from the outputs of registers RgP$_{Gjg}$ are fed to the inputs of adder NAdd, where they are summed. At the output of adder NAdd we obtain the first macropartial result P_{M1}. The leading edge of the second pulse TI3 writes the second group partial products P_{Gj2} and the first macropartial result P_{M1} to registers RgP$_{Gjg}$ and RgP$_{Mg}$, respectively.

In the third calculation cycle, switch Swj is set to the position when the third group of k subsequent digits of input number X_j arrives at its output, and the pulse from input R resets register RgZ_g to zero. In this cycle, we obtain the third group partial products P_{Gj3} at the outputs of adders $k$Add$j$, the second macropartial result P_{M2} at the output of adder NAdd, and the first macropartial result P_{M1} at output AddZ_g. The leading edge of the third pulse TI3 writes the third group partial products P_{Gj3}, the second macropartial result P_{M2}, and the first macropartial result P_{M1} to registers RgP$_{Gjg}$, RgP$_{Mg}$ and RgZ_g, respectively.

In the following cycles of operation, the scalar product calculation is performed similarly. The result of the scalar product calculation Z is obtained after (m + 3)-th cycle of operation. This result is written to register RgZ by the leading edge of clock pulse TI4 and from its outputs is fed to the AFC inputs. The value of the output signal y of NE is obtained at the AFC outputs.

5. 2. Estimating the parameters of a recursive-type neural element with parallel vertical-group data processing

The basic parameters used to evaluate the recursive-type NE with parallel vertical-group data processing are equipment costs, the time of calculating the output signal y, and the efficiency of equipment use.

Since the structure of the recursive-type NE with parallel vertical-group data processing is oriented towards the VLSI implementation, it is advisable to take a logic gate (inverter, AND, OR type element) as the unit of calculation of equipment costs.

The equipment costs for implementing the recursive-type NE with parallel vertical-group data processing are determined from the following expression

$$W_{\text{NE}} = (5N+3)W_{\text{Rg}} + NW_{\text{Sw}} + kNW_{\text{PP}} + \\ + NW_{k\text{Add}} + W_{N\text{Add}} + W_{\text{Add}} + W_{\text{AFC}} + W_{\text{CU}}, \quad (6)$$

where W_{Rg} – equipment costs for implementing the register; W_{Sw} – equipment costs for implementing the switch; W_{PP} – equipment costs for implementing the PP node; $W_{k\text{Add}}$ – equipment costs for implementing the k-input adder; $W_{N\text{Add}}$ – equipment costs for implementing the N-input adder; W_{AFC} – equipment costs for implementing the activation function calculator; W_{CU} – equipment costs for implementing the control unit.

To estimate the costs in logic gates, it is necessary to reveal the structure of the PP node, the activation function calculator AFC, and the control unit CU. The PP node is implemented on the basis of N 2-input AND gates. Depending on the tasks for which the artificial neural network is oriented, the following activation functions can be used in NE: sigmoid, hyperbolic tangent, ReLU, Leaky ReLU, ELU, Swish, and threshold. Each of these activation functions has its own advantages and disadvantages. The choice of activation functions for NE depends on the specific task and architecture of the neural network. For the recursive-type NE with parallel vertical-group data processing, the ReLU activation function was selected, which is implemented on the basis of a 2-input n-bit switch.

In the CU control unit, the generation of clock pulses and control signals is carried out by firmware. The implementation of such a firmware control unit CU requires the following hardware costs

$$W_{\text{CU}} = W_{\text{ROM}} + W_{\text{Cnt}} + W_{\text{Rg}}, \quad (7)$$

where W_{Cnt} – equipment costs for implementing the counter, W_{ROM} – equipment costs for implementing ROM.

From formulas (6) and (7) it is clear that the main functional units on the basis of which a recursive-type NE with parallel vertical-group data processing is synthesized are adders, registers, switches, counter, and memory. To estimate the equipment costs for functional units (in the number of logic gates), analytical expressions have been built, which are given in Table 1, where n – bit depth of functional units, m – number of inputs. The specified values were obtained by modeling functional units when implementing specialized VLSI processors for parallel-stream data processing.

Table 1

Equipment costs for implementing functional nodes and corresponding delay time

No.	Names of functional units	Cost of equipment (gates)	Number of delay stages (τ gates)
1	n-bit register	$7n$	3
2	n-bit adder	$20n$	$7 \log_2 n$
3	m-input n-bit adder	$(m-1)\, 20n$	$7 \log_2 n \log_2 m$
4	m-input n-bit switch	$3mn$	m
5	m-address n-bit ROM	$2^m n$	$(m+3)$
6	binary counter	$12\, n$	$5 \log_2 n$

To estimate the costs in logic gates, the value of the equipment costs for the implementation of individual functional units is substituted into formula (6) from the table. As a result, we obtain:

$$W_{\text{NE}} = (5N+3)W_{\text{Rg}} + NW_{\text{Sw}} + kNW_{\text{PP}} + \\ +NW_{k\text{Add}} + W_{N\text{Add}} + W_{\text{Add}} + W_{\text{AFC}} + W_{\text{CU}},$$

$$W_{\text{NE}} = (5N+3)W_{\text{Rg}} + NW_{\text{Sw}\,m\to k} + kNW_{\text{PP}} + \\ +NW_{k\,\text{Add}} + W_{N\,\text{Add}} + W_{\text{Add}} + W_{\text{Sw}\,2\to n} + \\ +W_{\text{ROM}\log_2 N\to 6} + W_{\text{Cnt}} + W_{\text{Rg}} = \\ =(5N+3)7n + 3Nn + kNn + N(k-1)20n + \\ +(N-1)20n + 20n + 6n + 2^{\log_2 N}6 + 42 = \\ =\left(38Nn + 21Nkn + 27n + 2^{\log_2 N}6 + 42\right) gates, \quad (8)$$

where N is the number of input data (weight coefficients), n is the number of bits of input data and weight coefficients, k is the number of bits in the group.

The estimation of the time of calculation of the output signal y in NE is carried out in units of the delay of data passage through the logic gate τ. The calculation time of the output signal y in the recursive-type NE with parallel vertical-group data processing depends on the number of clocks and their duration. To determine the duration of the clock, the values of the delay of data passage through the functional nodes, which are given in the table, were used. Applying values from Table 1, an analytical expression was derived for calculating the time of calculation of the output signal y in the recursive-type NE with parallel vertical-group data processing. The analytical expression for estimating the time of calculation of the output signal y in NE is written as

$$t_{\text{NE}} = \left(\left\lceil \frac{n}{k} \right\rceil + 3\right) t_{N\,\text{Add}} = \left(\left\lceil \frac{n}{k} \right\rceil + 3\right)\left(7\log_2\left\lceil \log_2 N \right\rceil\right)\tau. \quad (9)$$

For the integral assessment of NE, the equipment efficiency criterion E was used, which links performance to equipment costs and gives an assessment of elements (logic gates) by performance. The quantitative value for assessing equipment efficiency is determined from the following formula

$$E_{\text{NE}} = \frac{R_{\text{NE}}}{t_{\text{NE}} W_{\text{NE}}}, \quad (10)$$

where R_{NE} – complexity of the algorithm for calculating the NE output signal y, W_{NE} – equipment costs in the logic gates for implementing NE, t_{NE} – time for calculating the output signal y NE. The complexity of the algorithm for calculating the output signal y NE in elementary arithmetic operations (addition, shift) is determined as follows

$$R_{\text{NE}} = 2nN. \quad (11)$$

$$E_{\text{NE}} = \\ = \frac{2Nn}{\left(\left\lceil \frac{n}{k} \right\rceil + 3\right)\left(7\log_2\left\lceil \log_2 N \right\rceil\right)\left(38Nn + 21Nkn + 27n + 2^{\log_2 N}6 + 42\right)}. \quad (12)$$

By substituting the R_{NE} values from formula (11), t_{NE} from formula (9), and W_{NE} from formula (8) into formula (10), an analytical expression for calculating the efficiency of equipment use is derived

For operands with a bit size of $n = 24$ and $k = 2$, $k = 4$, $k = 6$, $k = 8$, $k = 12$, using the analytical expression (8), plots of equipment costs for the implementation of NEs were constructed, which are shown in Fig. 3.

Analysis of the plots (Fig. 3) reveals that the highest equipment costs when implementing NE will be for value $k = 12$, the lowest – for $k = 2$.

For operands with a bit depth of $n = 24$ and $k = 2$, $k = 4$, $k = 6$, $k = 8$, $k = 12$, using the analytical expression (9), plots of the calculation time of NE output signal were constructed, which are shown in Fig. 4.

The plots (Fig. 4) demonstrate that the smallest calculation time of the output signal of NE is obtained for $k = 12$, and the largest for $k = 2$.

For operands with a bit depth of $n = 24$ and $k = 2$, $k = 4$, $k = 6$, $k = 8$, $k = 12$, using the analytical expression (12), plots of the efficiency of using the NE equipment are constructed (Fig. 5).

From the plots (Fig. 5) it is clear that the highest efficiency of NE equipment use will be for $k = 4$, and the lowest for $k = 12$. At the same time, for a data bit depth of 16 bits, increasing the number of k bits that are simultaneously processed in a group from 2 to 8 leads to a 2.8-fold reduction in processing time while simultaneously reducing the efficiency of equipment use of the neural element by no more than 1.6 times.

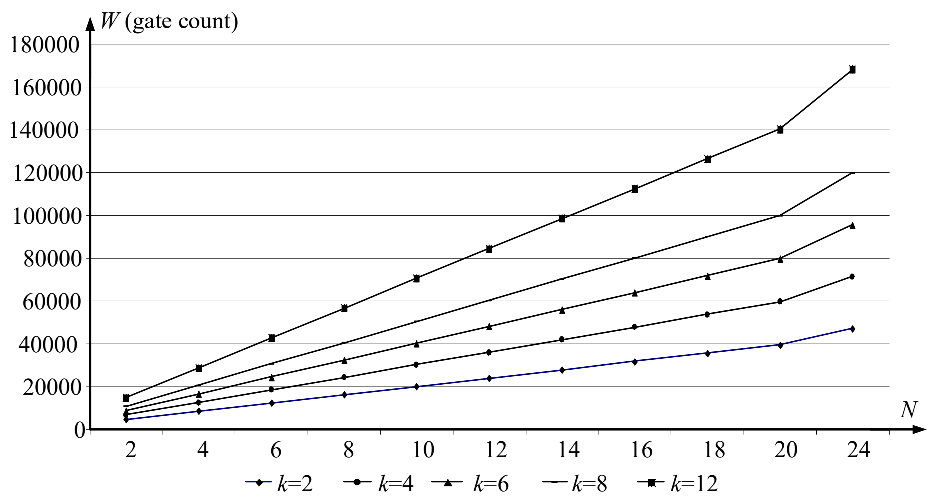

Fig. 3. Equipment cost plots for implementing a neural element

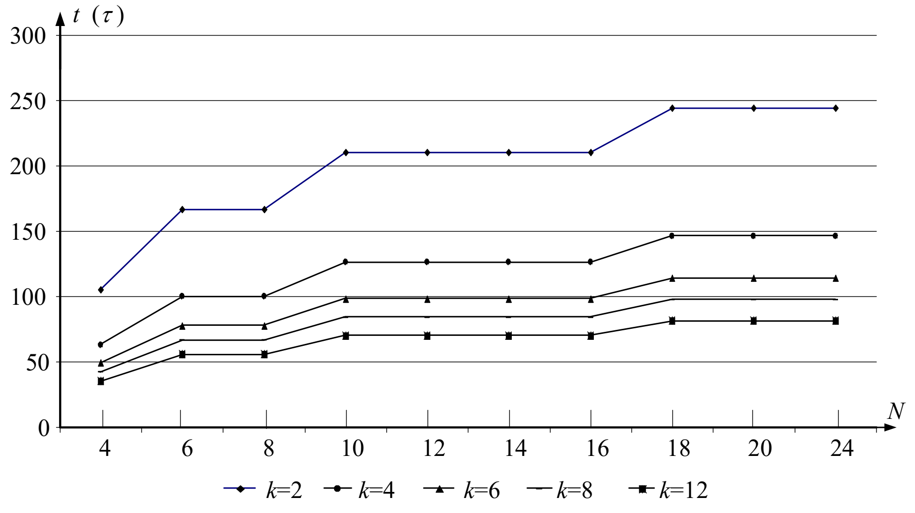

Fig. 4. Time plots of calculating the output signal of a neural element

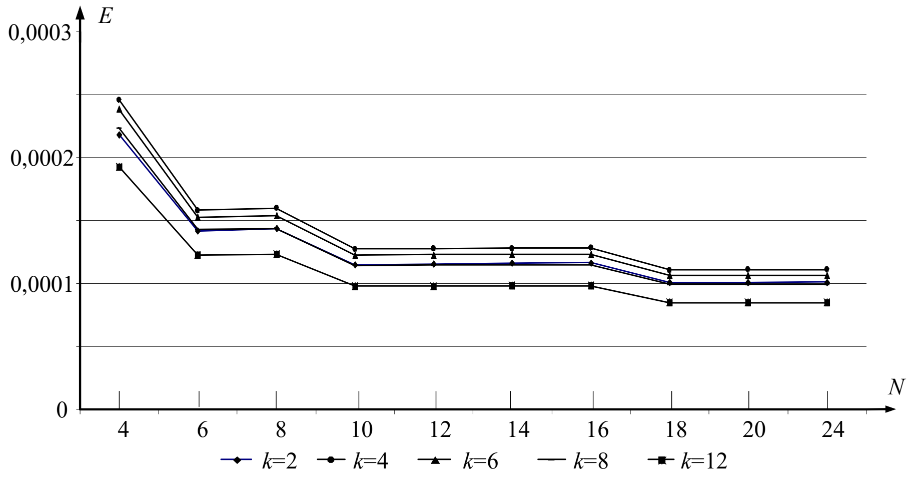

Fig. 5. Neural element equipment utilization efficiency plots

5. 3. Method for synthesizing a recursive-type neural element with parallel vertical-group data processing in real time

The task of synthesizing a recursive-type NE with parallel vertical-group data processing is reduced to enabling real-time operation with minimal hardware costs for its implementation. The output data for synthesis of a recursive-type NE with parallel vertical-group data processing is the time of arrival of an array of N input data X_1, .., X_N and N weight coefficients W_1, .., W_N, which is determined from

$$t_d = NT_{TI1}, \quad (13)$$

where T_{TI1} is the duration of the period of arrival of input data X_j and weight coefficients W_j.

To enable real-time operation of NE, the following condition must be met

$$t_d \geq t_{\mathrm{NE}}, \quad (14)$$

where t_{NE} is the time of calculation of the output signal y NE.

The time of calculation t_{NE} depends on the number of bits in group k, which are used to obtain partial products, the volume of data N, the bit depth of the input data n, and is determined from formula (9).

The main parameter that reduces the time of calculation t_{NE} is k, the value of which can vary in the range $k = 2, ..., n/2$. Increasing the value of k leads to a decrease in the number of processing cycles m and increases the hardware costs for the implementation of PP receiving nodes and k-input adders of the nodes. The second parameter on which the time of calculation t_{NE} depends is the time of summing N group partial products P_{G1g}, ..., P_{GNg}, which determines the period of clock pulses TI3. This time can be reduced by pipelining the N-input adder, that is, dividing it into steps using registers.

To select a variant of implementing recursive-type NE with parallel vertical-group data processing in real time, the criterion of efficiency of using the equipment E_{NE} is used. High efficiency of using the equipment E_{NE} when implementing recursive-type NE with parallel vertical-group data processing in real time is achieved by matching the time of data arrival t_d with the time of calculation of the output signal y NE t_{NE}. Such matching may require both an increase and a decrease in t_{NE}.

The main ways to reduce t_{NE} are:

– increasing the number of bits k in the group, which are used to obtain partial products;

– pipelining the N-input adder by dividing it into stages;

– parallel inclusion of two or more devices for calculating the scalar product, the number of which is determined mainly by the time of arrival of input data t_d.

In the case when the data arrival time t_d is significantly greater than time t_{NE}, to ensure high efficiency of equipment use, they must be coordinated. Such coordination can be achieved as follows:

– reducing the number of bits k in the group, which are used to obtain partial products;

– using devices with smaller numbers of product pairs to calculate the scalar product

$$Z_1=\sum_{l=1}^{N/2}W_lX_l \text{ or } Z_2=\sum_{p=1}^{N/4}W_pX_p. \tag{15}$$

For the synthesis of a recursive-type NE with parallel vertical-group data processing in real time with a given time t_d of data arrival, the designed NE (Fig. 2) was used as the base. The synthesis of such an NE requires the following stages:

1) estimate the time $t_{N\text{Add}}$ of summing N group partial products P_{G1g}, ..., P_{GNg}, (duration of the period of clock pulses TI3);

2) determine the number of groups of bits m (processing cycles) from formula $m=\left\lceil\frac{t_d}{t_{N\,\text{Add}}}\right\rceil-3$, where m must be with in $n/2\geq m\geq 2$;

3) for the case when $m<2$, it is necessary to reduce time $t_{N\text{Add}}$ by dividing the N-input adder into stages so that the operation time of the conveyor stage t_{CS} ensures the fulfill ment of condition $n/2>m\geq 2$, where $m=\left\lceil\frac{t_d}{t_{\text{CS}}}\right\rceil$, or by using two or more parallel operating devices for calculating the scalar product;

4) for the case when $n/2\geq m\geq 2$, the number of digits in the group to obtain partial products is determined from formula $k=\left\lceil\frac{n}{m}\right\rceil$;

5) for the case when $m>n/2$, the calculation of the scalar product is implemented on devices with lower hard ware costs that calculate scalar products $Z_1=\sum_{l=1}^{N/2}W_lX_l$ or $Z_2=\sum_{p=1}^{N/4}W_pX_p$;

6) coordinate time t_d of data arrival with time t_{NE} of calculating the output signal y NE by using appropriate coordination procedures. To do this, one can change the number of bits in group k to obtain partial products, divide the N-input adder into stages, or use two or more parallel operating scalar product calculation devices to calculate the scalar product. One can implement the calculation of the scalar product $Z=\sum_{j=1}^{N}W_jX_j$ on devices that calculate scalar products $Z_1=\sum_{l=1}^{N/2}W_lX_l$ or $Z_2=\sum_{p=1}^{N/4}W_pX_p$;

7) perform an assessment of the efficiency of equipment use for different implementation options for recursive-type NE with parallel vertical-group data processing and select the NE option that has the highest efficiency of equipment use.

6. Discussion of results related to the synthesis of a recursive-type neural element with parallel vertical-group data processing

A feature of parallel-vertical type NE models described in [22, 23] is the complexity of matching the time of data arrival with the time of calculation of the NE output signal. The proposed recursive-type NE model with parallel vertical-group data processing ensures the matching of the time of arrival of weight coefficients and input data with the time of calculation of the NE output. This is achieved by choosing the number of bits in the group for forming partial products. The basis of the analytical (5) and graph models (Fig. 1) is the operation of calculating the scalar product by the parallel vertical-group method (2), which, through the choice of the number of bits in the group for forming partial products, provides a change (decrease or increase) in the time of calculation of the scalar product in NE. Due to the use of a serial-parallel converter of weight coefficients and input data, parallel-group converters of input data and nodes for calculating partial products, the timing of the arrival of weight coefficients and input data with the time of calculating the output of NE is ensured.

In [24], an FPGA-based implementation of NE with four inputs and a sigmoid activation function using 16-bit fixed-point numbers is described. The designed recursive-type neural element with parallel vertical-group data processing allows the use of various activation functions by using their tabular implementation.

Unlike the NE model described in [25] that has a more complex implementation, in our NE model the process of calculating the scalar product is reduced to the operation of group summation of partial products. Additionally, the use of a parallel vertical-group algorithm and the pipeline process provides the possibility of hardware implementation of the real-time neural element on FPGA.

In [26], a bit-wise implementation of NE for a spiking neural network based on FPGA is reported to construct a neural model scheme based on a biologically plausible model of a neuron with quadratic spikes. However, the proposed sequential bit-wise design of NE does not work for data with a small bit depth. In the case of a larger data depth, such an NE implementation does not require additional data processing cycles. In the case of the neural element proposed in the work, the data depth can be 8–32 bits, which increases computing performance.

The authors of [27] consider implementations of NE for Spiking Neural Networks (SNNs), which represent neural network models as an analogy to biological neurons. Information during transmission between neurons is encoded using time and amplitudes (weights). This approach slows down information transmission and reduces the performance of the neural network as a whole. The proposed NE operates on high-dimensional data in a parallel format, which provides high performance and streaming data processing in real time. By using the analytical expression for evaluating the efficiency of equipment utilization (12), the choice of the implementation option of recursive-type NE with parallel vertical-group data processing in real time is made.

In [28], the task of designing the architecture of a neural network during its implementation is considered. As the authors note, such an architecture is obtained manually by exploring its hyperparameter space and is kept fixed during training, and modern applications require small models due to the imposed resource limitations of embedded devices. The cited paper considers the SCANN methodology, which uses three main operations of changing the architecture, namely, increasing the number of connections, increasing the number of neurons, and cutting off connections. In the proposed synthesis method, minimizing the cost of equipment during the synthesis of recursive-type NE with parallel vertical-group data processing is achieved by using matching mechanisms that can reduce or increase the time of computing the NE output t_{NE}. The main mechanisms for reducing t_{NE} are increasing the number of bits k in the group; pipelining the N-input adder; parallel inclusion of two or more dot product calculation devices. To increase time t_{NE}, the following mechanisms are used: reducing the number of bits k in the group; reducing the number of product pairs in the dot product calculation device (15).

However, it is necessary to take into account the limitations of our study, which imply the implementation of a recursive-type neural element focused on parallel streaming data processing. The considered NE model is focused specifically on hardware implementation based on FPGA and on the possibility of parallel data processing. Software implementations of the developed neural element will not be appropriate.

The lack of analysis of implementation options for the proposed neural element with reference to specific architectures of programmable logic integrated circuits, for example, CPLD, can be considered a disadvantage of this study.

Further research on the synthesis of recursive-type NE with parallel vertical-group data processing will be aimed at designing high-speed pipeline multi-input adders used to sum partial products and group partial products. Also relevant is research aimed at developing algorithms and structures for recursive-type NEs with tabular formation of macropartial results.

7. Conclusions

1. A generalized recursive-type NE model with parallel vertical-group data processing has been built, the main components of which are nodes for calculating partial products, multi-input adders of group partial products, a multi-input adder for calculating the macropartial result, and a macropartial result summator. The model uses a serial-parallel converter of weight coefficients and input data, parallel-group converters of input data, and an activation function calculator. The designed NE model uses a parallel vertical-group method for calculating the scalar product with the ability to select the number of bits in the group for forming partial products. In the model, the coordination of the time of arrival of weight coefficients and input data with the time of calculating the NE output provides real-time data processing. The designed graph model of the neural element is oriented towards hardware implementation. The basic structure of NE has been developed, which, due to the use of hardware mapping of the designed graph model of NE, provides a reduction in the time of NE synthesis for specific applications. The specified result is achieved due to the regularity and modularity of the structure, pipelining, and spatial parallelism of data processing, as well as organization of the process of calculating the scalar product as the execution of a single operation.

2. For the designed recursive-type neural element with parallel vertical-group data processing, the parameters for estimating equipment costs, speed, and efficiency of equipment use have been determined. The estimation of NE parameters is carried out depending on the bit depth of the operands, the amount of data, and the number of bits in the group to obtain partial products. The use of the proposed estimation parameters allows us, at the stage of synthesis of neural elements for specific applications, to estimate their quantitative parameters and select the optimal structure with given time characteristics.

3. A method for synthesizing recursive-type NEs has been devised, which, by using the basic structure of NEs, mechanisms for coordinating the time of arrival of weight coefficients and input data with the time of output calculation, ensures the implementation of NEs for specific applications. The proposed method takes into account ways to minimize equipment costs and provides the synthesis of recursive-type NEs with parallel vertical-group data processing, which implements data processing 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, as well as the results reported in this paper.

Funding

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

Data availability

All data are available, either in numerical or graphical form, in the main text of the manuscript.

Use of artificial intelligence

The authors confirm that they did not use artificial intelligence technologies when creating the current work.

References

1. Leigh, A. J., Heidarpur, M., Mirhassani, M. (2022). A Low-Resource Digital Implementation of the Fitzhugh-Nagumo Neuron. 2022 17th Conference on Ph.D Research in Microelectronics and Electronics (PRIME), 369–372. https://doi.org/10.1109/prime55000.2022.9816797
2. Renteria-Cedano, J., Rivera, J., Sandoval-Ibarra, F., Ortega-Cisneros, S., Loo-Yau, R. (2019). SoC Design Based on a FPGA for a Configurable Neural Network Trained by Means of an EKF. Electronics, 8 (7), 761. https://doi.org/10.3390/electronics8070761
3. Tsmots, I. G., Opotyak, Yu. V., Shtohrinets, B. V., Mamchur, T. B., Oliinyk, O. O. (2024). Operational basis of artificial neural networks and evaluation of hardware characteristics for its implementation. Ukrainian Journal of Information Technology, 6 (2), 125–138. https://doi.org/10.23939/ujit2024.02.125

4. Kundu, S., Banerjee, S., Raha, A., Basu, K. (2022). Special Session: Effective In-field Testing of Deep Neural Network Hardware Accelerators. 2022 IEEE 40th VLSI Test Symposium (VTS), 1–4. https://doi.org/10.1109/vts52500.2021.9794227
5. Wu, J., Zhao, B., Wen, H., Zhao, Q. (2022). Design of Neural Network Accelerator Based on In-Memory Computing Theory. 2022 4th International Conference on Natural Language Processing (ICNLP), 547–551. https://doi.org/10.1109/icnlp55136.2022.00100
6. Sarg, M., Khalil, A. H., Mostafa, H. (2021). Efficient HLS Implementation for Convolutional Neural Networks Accelerator on an SoC. 2021 International Conference on Microelectronics (ICM), 1–4. https://doi.org/10.1109/icm52667.2021.9664920
7. Nouacer, R., Hussein, M., Espinoza, H., Ouhammou, Y., Ladeira, M., Castiñeira, R. (2020). Towards a framework of key technologies for drones. Microprocessors and Microsystems, 77, 103142. https://doi.org/10.1016/j.micpro.2020.103142
8. Saini, S., Lata, K., Sinha, G. R. (2021). VLSI and Hardware Implementations Using Modern Machine Learning Methods. CRC Press. https://doi.org/10.1201/9781003201038
9. Davies, M., Srinivasa, N., Lin, T.-H., Chinya, G., Cao, Y., Choday, S. H. et al. (2018). Loihi: A Neuromorphic Manycore Processor with On-Chip Learning. IEEE Micro, 38 (1), 82–99. https://doi.org/10.1109/mm.2018.112130359
10. Hager, G., Wellein, G. (2010). Introduction to High Performance Computing for Scientists and Engineers. CRC Press. https://doi.org/10.1201/ebk1439811924
11. 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
12. Liu, W., Wang, Z., Liu, X., Zeng, N., Liu, Y., Alsaadi, F. E. (2017). A survey of deep neural network architectures and their applications. Neurocomputing, 234, 11–26. https://doi.org/10.1016/j.neucom.2016.12.038
13. Tsmots, I. G., Opotyak, Y. V., Shtohrinets, B. V., Mamchur, T. B., Holubets, V. M. (2024). Model, structure and synthesis method of matrix-type neural element. Scientific Bulletin of UNFU, 34 (4), 68–77. https://doi.org/10.36930/40340409
14. Tsmots, I., Skorokhoda, O., Ignatyev, I., Rabyk, V. (2017). Basic vertical-parallel real time neural network components. 2017 12th International Scientific and Technical Conference on Computer Sciences and Information Technologies (CSIT), 344–347. https://doi.org/10.1109/stc-csit.2017.8098801
15. Leigh, A. J., Mirhassani, M., Muscedere, R. (2020). An Efficient Spiking Neuron Hardware System Based on the Hardware-Oriented Modified Izhikevich Neuron (HOMIN) Model. IEEE Transactions on Circuits and Systems II: Express Briefs, 67 (12), 3377–3381. https://doi.org/10.1109/tcsii.2020.2984932
16. Pi, X., Lin, X. (2022). An FPGA-based Piecewise Linear Spiking Neuron for Simulating Bursting Behavior. 2022 IEEE 12th Annual Computing and Communication Workshop and Conference (CCWC), 0415–0420. https://doi.org/10.1109/ccwc54503.2022.9720886
17. Wang, G., Fu, D. (2024). Spike Neural Network with Delayed Propagation Characteristics and Hardware Implementation. 2024 6th International Conference on Electronic Engineering and Informatics (EEI), 1181–1185. https://doi.org/10.1109/eei63073.2024.10696338
18. Ramirez-Morales, R. R., Ponce-Ponce, V. H., Molina-Lozano, H., Sossa-Azuela, H., Islas-García, O., Rubio-Espino, E. (2024). Analog Implementation of a Spiking Neuron with Memristive Synapses for Deep Learning Processing. Mathematics, 12 (13), 2025. https://doi.org/10.3390/math12132025
19. Xu, Q., Ding, S., Bao, H., Chen, M., Bao, B. (2022). Piecewise-Linear Simplification for Adaptive Synaptic Neuron Model. IEEE Transactions on Circuits and Systems II: Express Briefs, 69 (3), 1832–1836. https://doi.org/10.1109/tcsii.2021.3124666
20. Bao, B., Zhu, Y., Li, C., Bao, H., Xu, Q. (2020). Global multistability and analog circuit implementation of an adapting synapse-based neuron model. Nonlinear Dynamics, 101 (2), 1105–1118. https://doi.org/10.1007/s11071-020-05831-z
21. Tsmots, I. H., Shtohrinets, B. V., Kazymyra, I. Y., Lytvyn, A. A. (2023). Model and method for synthesis of neural element of parallel-streaming type. Scientific Bulletin of UNFU, 33 (2), 92–100. https://doi.org/10.36930/40330213
22. Tsmots, I. H., Skorokhoda, O. V., Medykovskyi, M. O. (2017). Pat. No. 118596 UA. Prystriy dlia obchyslennia skaliarnoho dobutku. No. a201700835; declareted: 30.01.2017; declareted: 11.02.2019.
23. Tsmots, I. H., Tesliuk, V. M., Lukashchuk, Yu. A., Kazymyra, I. Ya. (2021). Pat. No. 127774 UA. Prystriy dlia obchyslennia skaliarnoho dobutku. No. a202104653; declareted: 12.08.2021; declareted: 28.12.2023..
24. Shymkovych, V., Doroshenko, A., Mamedov, T., Yatsenko, O. (2022) Automated design of an artificial neuron for field-programmable gate arrays based on an Algebra-Algorithmic approach. International Scientific Technical Journal «Problems of Control and Informatics», 67 (5), 61–72. https://doi.org/10.34229/2786-6505-2022-5-6
25. Tsmots, I., Opotyak, Y., Shtohrinets, B. (2023). Method of Synthesis of Devices for Parallel Stream Calculation of Scalar Product in Real Time. Vìsnik Nacìonalʹnogo Unìversitetu "Lʹvìvsʹka Polìtehnìka". Serìâ Ìnformacìjnì Sistemi Ta Merežì, 14, 248–266. https://doi.org/10.23939/sisn2023.14.248
26. Lin, X., Lu, H., Pi, X., Wang, X. (2020). An FPGA-based Implementation Method for Quadratic Spiking Neuron Model. 2020 11th IEEE Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON), 0621–0627. https://doi.org/10.1109/uemcon51285.2020.9298029
27. Zi, H., Zhao, K., Zhang, W. (2024). Designing and Accelerating Spiking Neural Network Based on High-Level Synthesis. 2024 Conference of Science and Technology for Integrated Circuits (CSTIC), 1–3. https://doi.org/10.1109/cstic61820.2024.10531920
28. Hassantabar, S., Wang, Z., Jha, N. K. (2022). SCANN: Synthesis of Compact and Accurate Neural Networks. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 41 (9), 3012–3025. https://doi.org/10.1109/tcad.2021.3116470