mathematics

MDPI

Article

A Model of a Universal Neural Computer with Hysteresis Dynamics for Avionics Problems

Andrey M. Solovyov [1,*], Nikolay I. Selvesyuk [2], Vladislav V. Kosyanchuk [2] and Evgeniy Y. Zybin [2]

[1] JSC "Concern "Sozvezdie", Plekhanovskaya Str. 14, 394018 Voronezh, Russia
[2] State Research Institute of Aviation Systems, Viktorenko Str. 7, 125167 Moscow, Russia; nis@gosniias.ru (N.I.S.); vvk@gosniias.ru (V.V.K.); eyzybin@2100.gosniias.ru (E.Y.Z.)
* Correspondence: darkzite@yandex.ru

Abstract: The paper proposes a method for implementing a universal neural network processor with hysteresis dynamics. This processor allows a wide range of heterogeneous tasks in real time to be performed without reprogramming and changing their internal structure. Adding hysteresis behavior to the system makes it possible to increase resistance to external influences, the complexity as well as non-linearity of intelligent output. The paper discusses the use of this processor as part of an on-board intelligent avionics system.

Keywords: neural network processor; avionics; on-board intelligent system; decision support system; hysteretic neural network; hysteresis; Preisach model

MSC: 68T42

Citation: Solovyov, A.M.; Selvesyuk, N.I.; Kosyanchuk, V.V.; Zybin, E.Y. A Model of a Universal Neural Computer with Hysteresis Dynamics for Avionics Problems. *Mathematics* **2022**, *10*, 2390. https://doi.org/10.3390/math10142390

Received: 27 May 2022
Accepted: 5 July 2022
Published: 7 July 2022

1. Introduction

Modern avionics systems are complex distributed computing systems, which must solve a wide range of problems and comply with stringent aviation security requirements [1,2]. The most complex and important element of such systems is an onboard computer network (OCN), which provides for the functioning of applications performing the diagnostics and management of the aircraft systems.

Distributed modular electronics (DME), i.e., second-generation integrated modular avionics (IMA), are considered to be the most promising basis for OCNs [3–6]. DME-based OCNs are distributed computing systems characterised by a large number of computational nodes with large computing power and a wide range of supported aircraft functions. Therefore, they must comply with the strict requirements of the capacity of the data transmission channels. The use of distributed computing systems for the design of OCNs makes it necessary to solve the problem of ensuring the fault tolerance of OCNs while maintaining the required functions.

Currently, the security problem is being solved by repeatedly duplicating critical avionics systems. It leads to significant material costs and deterioration of weight and size characteristics, which increases the cost of operating the aircraft [7].

Avionics suites (AS) based on IMA, which replaced the traditional federated avionics architecture, appeared as a result of the development of avionics engineering, and are intertwined with the design of new aircraft with enhanced flight performance and increased requirements to the reliability and unification of their structural elements. Modern avionics suites must have a larger number of functions, lower production and operation costs. It is also necessary to integrate the hardware, the software and the used algorithms. We can therefore name the following development areas of IMA:

- development of unified, open and fault tolerant architectures for avionics suites using promising high-speed network interfaces;

- unification of the equipment aiming to reduce the range of devices being developed, speed up the preparation of engineering documentation, enhance the weight and dimensions parameters of the equipment and improve the overall performance, reliability and fault resistance of avionics systems;
- extending the functions of avionics suites to ensure security and enhance their performance;
- development of effective integrated control methods (including intellectual ones) in order to increase fault resistance and ensure easy maintenance, including in the field;
- development and introduction of intellectual avionics systems which perform dynamic configuration control in order to ensure the durability of avionics suites and perform the set tasks despite either single or multiple faults.

To complete these listed tasks, it is necessary to develop information, logical and mathematical models of an IMA-based system, algorithms for integrated control over the state of the system based on these models, and algorithms for dynamic control over the configuration of the system during the operation of the whole suite. Due to complex intrasystem dependencies of the functioning of aircraft systems, their automated management, timely identification of emergencies and search for solutions enabling the systems to eliminate the occurring faults in real time proves to be a rather difficult task.

Traditional methods of solving these complex problems lead to unreasonably high material costs for hardware components. It should be particularly noted that avionics systems operate in real time, and aviation standards (such as ARINC 653) impose strict requirements on such systems in terms of time delays, time interval of decision making, etc.

This problem can be solved by means of artificial intelligence and machine learning. At the same time, one of the important applied problems determining the effectiveness of intellectual avionics systems is the design of specialised computers and coprocessors, which significantly increase the capacity of the intellectual output under rapidly changing conditions with high uncertainty level. The present paper presents a model of such a specialised computer which can be used to implement a large variety of artificial neural networks, including neural networks with hysteresis activation function [8].

2. Universal Neural Computer

One of the important requirements of an aircraft intellectual system as a part of an avionics suite is the use of various computational methods. These methods include the neural network method based on the use of an artificial neural network (ANN) as a specialised computer allowing for intellectual output. In our study, we developed a model of a universal neural computer (UNC) which can be used to implement a wide range of ANNs with different architectures. The model allows for the implementation of an ANN as an HDL-project for FPGA, a software unit for a microcontroller, or one of the elements of a system on a chip (SoC). The suggested model also allows for different types of computations by changing the architecture of the ANN without altering the structure of the UNC itself or its algorithms. The model of the UNC is shown in Figure 1.

Figure 1 demonstrates that the UNC consists of neurons N_j, whose inputs are multiplied by weighting coefficients a_{ij}. Then, for each neuron, the weighted sum of the inputs is transformed using the activation function F_j , and the output d_j is formed. The UNC can be described as a graph using an adjacency matrix and an incidence matrix. This approach, however, results in an overuse of resources, namely the amount of RAM. Taking into account the specifics of the neural architecture, namely the presence of multiple inputs and a single output, the UNC can be described using a neuron interaction matrix $\mathbf{M}^{n\times n}$ (NIM). Its off-diagonal elements are weighting coefficients of the neuron inputs with the number corresponding to the number of the row, and its diagonal element is the output of the neuron (Table 1).

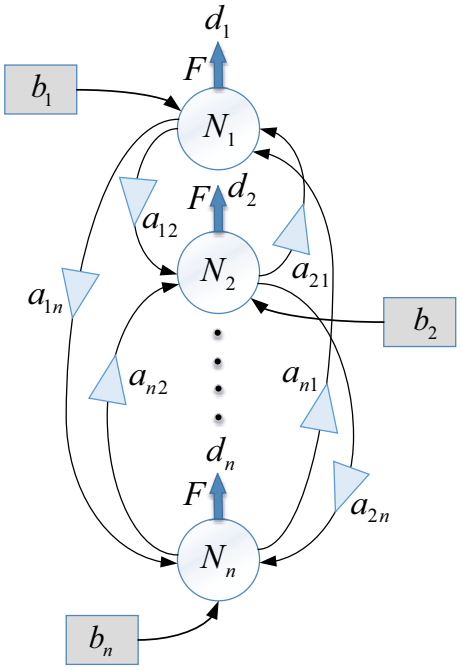

Figure 1. The UNC model.

Table 1. The neuron interaction matrix of the UNC.

N	**1**	**2**	**…**	***n***
1	d_1	a_{12}	…	a_{1n}
2	a_{21}	d_2	…	a_{2n}
…	…	…	…	…
n	a_{n1}	a_{n2}	…	d_n

Besides the NIM, we also need to describe the set of activation functions used by the UNC with regard to each neuron. In order to do this, let us introduce, in addition to the NIM, an activation function vector $\mathbf{F}^n$ (AFV), whose elements f_i indicate the number of the activation function used for the neuron N_i. Coefficients a_{ij} in the NIM affect the slope of the activation function of each neuron. However, to increase the flexibility of the UNC, we should also introduce displacement coefficients b_i, which allow us to shift the activation function of each neuron either to the left or to the right. To do so, we introduce a displacement coefficient vector $\mathbf{B}^n$, containing b_i. Thus, we can mathematically describe the UNC by the following set

$$\left\{\mathbf{M}^{n\times n}, \mathbf{B}^n, \mathbf{F}^n\right\}, \tag{1}$$

where n is the number of neurons in the UNC. The dynamics of the UNC can be described by the following transformation

$$\mathbf{D}_t = \mathbf{F}(\mathbf{D}_{t-1} \times \mathbf{M} + \mathbf{B}), \tag{2}$$

where

$$\mathbf{D} = Diag\left(\mathbf{M}^{n\times n}\right). \tag{3}$$

Here vectors **D** and **B** are rows, and the t and $t-1$ labels mean the current and previous iterations, respectively.

3. Neuron Activation Function

One of the key aspects determining the capacity of an ANN is the choice of the activation function (AF) [9,10]. There is a wide range of AFs, from step and linear functions to sigmoid and hyperbolic functions. The choice of the AF involves using the computer's

resources and, at the same time, determines the behaviour of the UNC and its ability to generalise and identify complex nonlinear dependences. It is obvious that the use of step and linear activation functions is beneficial as these employ a minimum of the computer's resources. However, their use deprives the UNC of nonlinear behaviour. Nonlinear activation functions require significant amounts of computing power, but allow for training the UNC on complex sets with hidden nonlinear dependences.

Taking into account all the advantages and disadvantages, consider the following standard functions: the hyperbolic tangent, the Softmax function, the linear function, and the hysteresis function.

The hyperbolic tangent. This activation function has a larger gradient than the sigmoid function, which has a positive effect on the backpropagation training.

The Softmax function. It is a logistic function in multiple dimensions. Clearly this function is applied to a vector, rather than to a specific value. Softmax can be used for classification problems. The output value of the Softmax function can be interpreted as the probability of occurrence of each class.

The linear function. This activation function can be used in data approximation problems.

The hysteresis function. This function significantly increases the flexibility of the UNC. It is obvious that this activation function helps to enhance the robustness of the neural network to various types of noise and improves the capacity of the intellectual output by adding degrees of freedom (i.e., parameters of the hysteresis model) [11–16] which determine the nonlinearity of the dynamics of the whole ANN. It is also important that the use of hysteresis functions with feedforward neural networks results in short-term memory effects. In other words, feedforward neural networks with hysteresis functions demonstrate the properties and the dynamics traditionally characterising recurrent neural networks [17]. Thus, by changing the parameters of the hysteresis activation functions, we can make the ANN behave both as a feedforward neural network and as a recurrent neural network.

With regard to the UNC model, the AFV will contain the values from an interval $k = \overline{0,4}$, where

0—transformation of f_i is off (the output is always 0);
1—the activation function is the hyperbolic tangent;
2—the activation function is the Softmax function;
3—the activation function is the linear function;
4—the activation function is the hysteresis function.

Taking into account the fact that the hardware implementation of nonlinear functions (for instance, by means of FPGA) is rather problematic, the following approach is used to solve this problem:

- The design process involves piecewise linear approximation of the activation function in the input interval $[x_{\min}, x_{\max}]$ with the sampling interval h (the number of intervals is m) ensuring the required accuracy of the UNC.
- As a result of the approximation, we obtain a table of values of coefficients $\{a_i, b_i\}$, $i = \overline{0, m-1}$ determining the slope of the linear function $y = a_i x + b_i$ in the segment $[x_{\min} + ih, x_{\min} + (i+1)h]$.
- The implementation of the function suggests that the input value x is used to find a corresponding segment, i.e., such an index i, for which $x \in [x_{\min} + ih, x_{\min} + (i+1)h]$. If x lies outside the $[x_{\min}, x_{\max}]$ interval, we use the segment closest to x. Then the following calculation is performed: $y = a_i x + b_i$.

The selected approach (piecewise linear approximation) is a compromise between using sets $f(x_i)$ with a small sampling interval h (which requires large amounts of memory) and calculating the nonlinear function directly (which requires significant amounts of computing power as well as specialised blocks performing multiplication, division, etc.).

Thus, the algorithm with an approximated AF can be presented as follows (Figure 2):

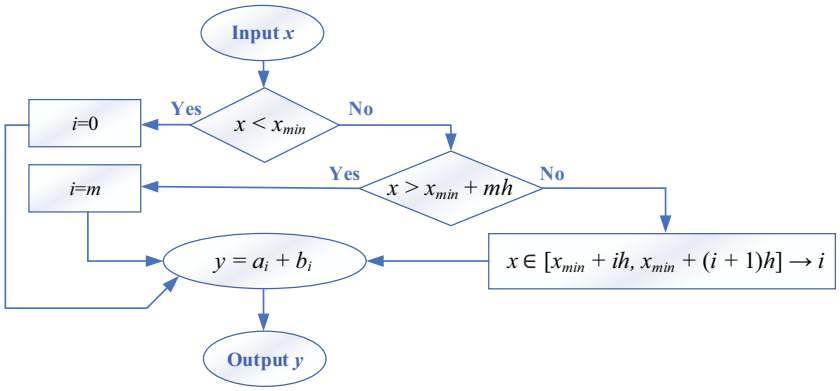

Figure 2. The algorithm with an approximated activation function.

4. Hysteresis Activation Function

The effectiveness of an ANN depends on a number of factors, including the network's architecture, implementation methods, training algorithms and reproduced biological properties of neurons in the human brain. Thus, neural plasticity allows for self-training neural networks. Another important property of biological neurons is their short-term memory. When designing an ANN, this property can be reproduced by means of the hysteresis activation function (HAF). At the moment, there are a large number of mathematical models of hysteresis, which can be used to construct the HAF, for instance the Bouc–Wen model [18–21] or the S-converter model [22]. However, with regard to the hardware implementation of hysteresis in the UNC, the Preisach model [23–28] proves to be more practical, since it is the simplest and the most scalable from the point of view of finite automata theory.

Let us consider the HAF based on the Preisach model in more detail. Let $R[\alpha, \beta, x_0]$ denote a two-position relay with thresholds α and β. The state space of the non-ideal relay is a pair of numbers $\{-1, 1\}$. The connection between the input $u(t) \in c_{[0,t]}$ and the alternating output $x(t) \in \{-1, 1\}$ is set by the operator $R[\alpha, \beta, x_0]$:

$$x(t) = R[\alpha, \beta, x_0]u(t), \tag{4}$$

where x_0 is the initial state of the converter. The relations between the input and the output are shown in Figure 3.

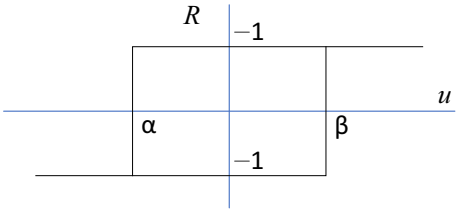

Figure 3. The input–output relation of the converter R.

The initial state x_0 of the converter must comply with the following conditions:

$$\begin{cases} x_0 = -1, & u(0) \leq \alpha, \\ x_0 = 1, & u(0) \geq \beta, \\ x_0 \rightarrow unchanging, & \alpha < u(0) < \beta. \end{cases} \tag{5}$$

The Preisach converter is called a continuous analogue of a converter consisting of non-ideal relays connected in parallel. We will further use the Preisach converter consisting of a finite number of operators $R[\alpha, \beta, x_0]$ as the activation function of the ANN. The scheme of the converter is shown in Figure 4.

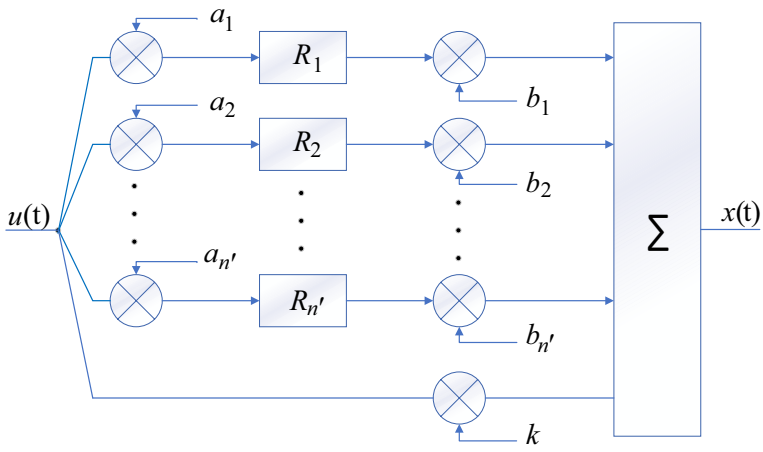

Figure 4. The scheme of the Preisach converter.

Thus, the activation function of each neuron contains n non-ideal relays $R[\alpha, \beta, x_0]$, whose characteristic curve is shown in Figure 3. Additional coefficients a'_n, b'_n, and k determine the slope of the characteristic curves of the non-ideal relays and, as a result, the shape of the characteristic curve of the Preisach converter.

If necessary, we can assume that coefficients $a_{n'} = \{1\}$, $b_{n'} = \{1\}$, $k = 0.1$ and the shift interval of the thresholds α and β is $h = 0.01$. In this case, the characteristic curve of the converter will be as shown in Figure 5. The slope of the characteristic curve provides for differentiability and thus for better correlation when using the gradient method to find an optimal solution for the training of the ANN.

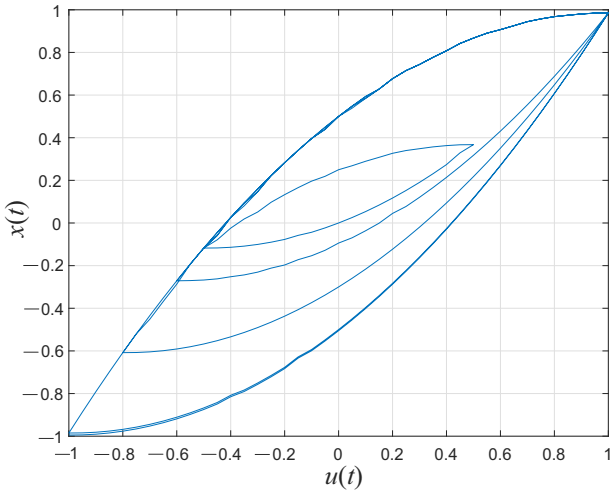

Figure 5. Characteristic curve of the Preisach converter.

5. An ANN with Hysteresis

Consider the construction of a feedforward neural network with HAF [8] which can be implemented by the UNC. The ANN is shown in Figure 6. As a part of an intellectual avionics system, this ANN can be used for object classification, image recognition, route building, searching for a suitable configuration of the distributed computer network in an emergency situation, etc.

$\mathbf{Y}^m$—the vector of input values;
$\mathbf{W}^{mn}$—the weighting matrix of the hidden layer;
$\mathbf{U}^n$—the vector of influences of the hidden layer;
$\mathbf{Q}^n$—the output vector of the hidden layer;
$\mathbf{V}^{np}$—the weighting matrix of the output layer;
$\mathbf{E}^p$—the vector of influences of the output layer;
$\mathbf{X}^p$—the target vector;

$G = f(U, X, \dot{X}, t)$—the activation function.

The number of neurons n in the hidden layer must be 30–50% of m. This number is determined empirically. A too-large n can result in the overtraining of the neural network, which will negatively affect its generalization ability. If n is too small, the ANN can lose its capacity to learn.

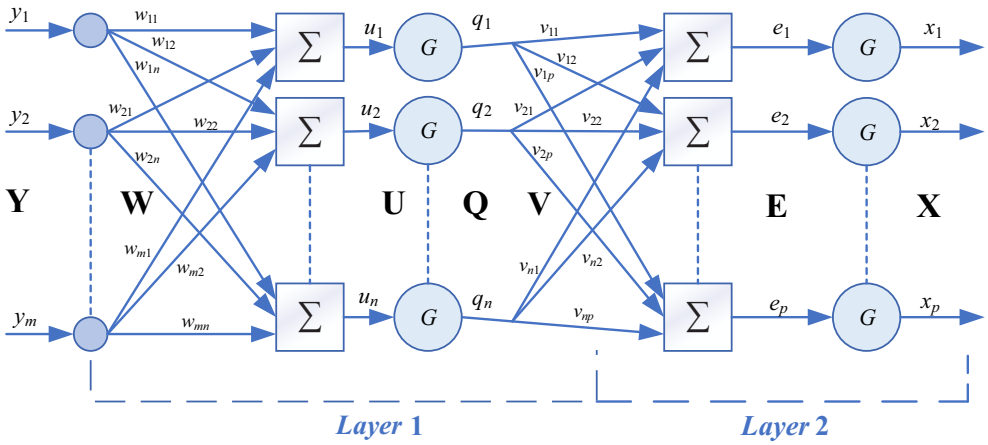

Figure 6. The architecture of a two-layer ANN with HAF.

Next, we introduce the following designations for the indices: the $i = \overline{1,m}$ index numbers the inputs, the $j = \overline{1,n}$ index numbers the elements of the hidden layer, and the $k = \overline{1,p}$ index numbers the outputs. The l index indicates the number of the iteration.

The forward propagation proceeds as follows:

$$\mathbf{U}^n = \mathbf{Y}^m \mathbf{W}^{mn}, \tag{6}$$

$$\mathbf{Q}^n = G(\mathbf{U}^n), \tag{7}$$

$$\mathbf{E}^p = \mathbf{Q}^n \mathbf{V}^{np}, \tag{8}$$

$$\mathbf{X}^p = G(\mathbf{E}^p). \tag{9}$$

To train this neural network we should use the algorithm for the minimisation of the target error function, which is calculated using the formula

$$E(W, V) = \frac{1}{2} \sum_{k=1}^{p} (x_k - d_k)^2. \tag{10}$$

Here, x_k is the calculated real value of the k-th output, when the neural network receives one of the input patterns of the training set $(\mathbf{Y}^t, \mathbf{D}^t)$, $t = \overline{1,T}$. Here d_k is the target value of the k-th output for this pattern.

The neural network is trained using the gradient descent optimisation method, i.e., the weights are reset during each iteration using the following formulas:

$$w_{ij}{}^{N+1} = w_{ij}{}^{N} - \alpha \frac{\partial E}{\partial w_{ij}}, \tag{11}$$

$$v_{jk}{}^{N+1} = v_{jk}{}^{N} - \alpha \frac{\partial E}{\partial v_{jk}}, \tag{12}$$

where α is a coefficient determined during the training process.

Then, we modify the training algorithm by adding numerical differentiation of the HAF to the backpropagation procedure. This results in the following relations for the correction of weights of the output layer **V**:

$$\frac{\partial E}{\partial v_{jk}} = \delta_k q_j, \tag{13}$$

$$\delta_k = (x_k - d_k)\frac{{x_k}^{(l+1)} - {x_k}^{(l)}}{{e_k}^{(l+1)} - {e_k}^{(l)}}, \tag{14}$$

and for the hidden layer **W** they are

$$\frac{\partial E}{\partial w_{ij}} = y_i \frac{{q_j}^{(l+1)} - {q_j}^{(l)}}{{u_j}^{(l+1)} - {u_j}^{(l)}} \sum_{k=1}^{p} \left(\delta_k v_{jk}\right). \tag{15}$$

6. Hardware Implementation of the UNC

Obviously, there are a number of implementations of the UNC, each having its own advantages and disadvantages. They can all be divided into three groups:

- *Parallel implementation*. In this case, the calculation of the dynamics is performed in parallel (row by row) using n independent processes (according to the number of rows in the NIM). This type of implementation requires significant amounts of computing power, but has the minimum UNC response time.
- *Sequential implementation*. In this case, the calculation of the dynamics is performed as a single process step by step. It has long UNC response time, but does not require a lot of computing power.
- *Parallel-sequential implementation*. In this case, some of the calculations are performed in parallel (a certain number of independent processes) and the others are subsequent. This approach is a compromise between the first two and ensures the required UNC response time while using a limited amount of computing power. Thus, the a priori requirements to the resources used and the response time determine the implementation method of the UNC.

Using FPGA we can employ all three types of the UNC implementation. While using a processor we can only resort to the sequential implementation of the UNC. Since there are strict requirements to the response time of an intellectual avionics system in an emergency situation (in fact, the response time is crucial), we suggest using parallel implementation of the UNC by means of FPGA.

In what follows, the main advantages and disadvantages of the UNC outlined in this paper are described.

Advantages:

- it can be used to implement a large number of ANNs of various types (multilayer perceptrons, Hopfield network, Boltzmann machine, deep learning networks, autoencoder, etc.), both feedforward and recurrent, without changing the architecture of the UNC;
- it allows for a dynamic change in the architecture of the ANN without altering the architecture of the UNC due to its uniformity (can be useful for solving different problems by means of a single UNC);
- when we use parallel implementation of the UNC based on FPGA, the UNC response time is determined by the FPGA production technology and at the moment can be measured in nanoseconds.

Disadvantages:

- the UNC requires large amounts of computing power, as does any system with parallel data processing. It is practical when used for real-time problems in situations when decisions must be made in limited time;

- when implementing standard ANNs, the neurons of the input layer are only used as memory cells, while their computing power is not employed. This is a consequence of the uniformity of the UNC, which can be avoided by partially deviating form the uniformity principle.

The structure of the HDL-project for FPGA implementing the UNC is shown in Figure 7. As we can see, the project describes parallel implementation of the UNC as an isolated coprocessor accessed via a standard PCI-E interface. Obviously, this type of project implementation minimizes the reaction time of the system to the change in the input state from the complete parallelism of all processes inside this coprocessor. As was presented in the advantages of the presented approach, the time delays are determined by the FPGA production technology and amount to units of nanoseconds (in accordance with modern technological processes).

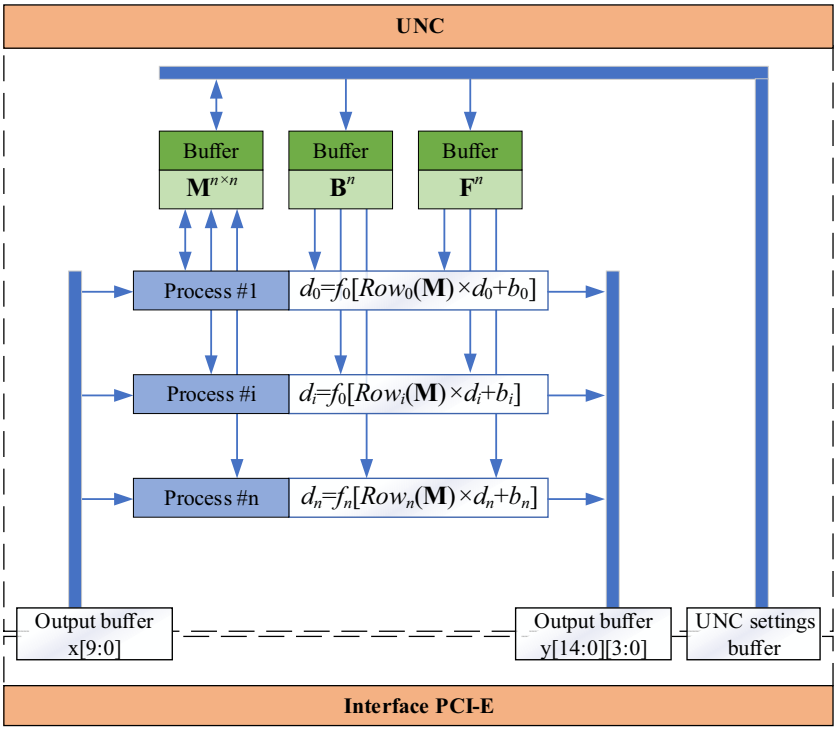

Figure 7. The structure of the HDL-project for FPGA implementing the UNC.

Currently, avionics is increasingly using systems on the chip in which the neural matrix is implemented in hardware, for example, chips from NVIDIA (such as Jetson Nano) or from Xilinx (such as Versal). These solutions have a wide range of different libraries and tools that facilitate their use. However, these are very expensive devices, and their use requires the transfer of the entire complex of on-board equipment to a fundamentally different technological level. This may be appropriate when creating completely new systems. The approach proposed in the manuscript is an inexpensive solution that allows the user to modify the already existing intelligent systems of the onboard equipment complex. The proposed coprocessor does not require new systems on a chip and can be integrated into an existing standard FPGA project.

7. Application of the UNC

Let us consider one of the possible applications of the UNC as a coprocessor for an expert decision support system which reconfigures the avionics suit in emergency situations. We assume that the avionics suit is based on the principles of distributed modular avionics whose compute nodes are connected via an on-board all-optical network [29].

In this case, the decision support system has the following functional components:

- a supervisor;
- a knowledge base;
- an expert system.

The supervisor continuously analyses the state of the on-board network and its elements in real time. The knowledge base includes a set of solutions enabling the system to parry any equipment failures. In particular, it contains a set of possible configurations of the avionics suite.

The expert system produces intellectual output (searches for the optimal solution) based on the information from the supervisor and the possible solutions in the knowledge base.

Since the knowledge base has a large set of heterogeneous data with nonlinear dependences which cannot always be formalised, it is reasonable to use a neural coprocessor as a functional element of the expert system. The UNC can well serve as such a coprocessor.

Let us consider a simplified example of the application of the UNC as an element of the expert system. It is obvious that, in an emergency situation when it is necessary to reconfigure the on-board network, the response time of the expert system is critical. The response time, in turn, depends on the time required to find the necessary information in the knowledge base and the time of the intellectual output. Since the UNC runs all the processes in parallel, it can significantly reduce the response time of the expert system in an emergency situation.

Assume, for example, that the on-board all-optical network consists of 10 optical channels and the knowledge base includes 14 possible configurations of the network. Assume, furthermore, that the knowledge base also stores information about the active optical channels (being used by the current configuration) and reserve channels (not used by the current configuration). Then the data in the knowledge base can be presented as the following table:

Here, 1 corresponds to the active channels and 0 corresponds to the reserve channels.

Assume, moreover, that the supervisor analyses the state of the optical transceivers of the on-board network and transmits this information to the expert system as the following state vector:

State x is determined as follows: 0—normal operation, 1—failure.

It is obvious that, in order to accelerate the intellectual output, a neural network can be used as a coprocessor. In this case, the neural network will be a multilayer perceptron. Analogous to the image classification problem, the neural network can be trained using a set where the input vectors are the failure vectors (obtained by the supervisor) and the output vectors are the most optimal configurations from the knowledge base. By an optimal configuration we mean the best of the existing configurations, i.e., the one that can be used to parry the largest number of failures and has the smallest number of active optical channels. Failures are parried by switching the faulty optical channel to the reserve state within the selected configuration.

Let us use the UNC to deploy a multilayer perceptron (Figure 8).

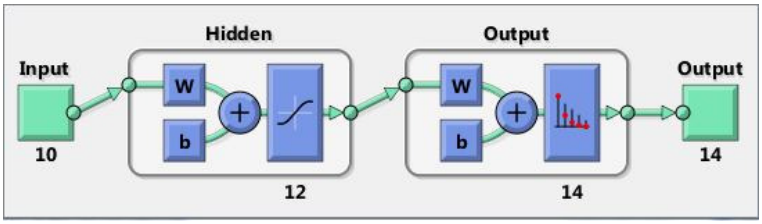

Figure 8. A two-layer perceptron for the expert system.

The neural network is trained using the backpropagation method and a pre-formed training set. The input vectors in the training set are random, while the output vectors are determined using the knowledge base (Table 2) and the principle of selecting the optimal configuration described above.

Table 2. Knowledge base.

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
1	0	1	1	1	0	1	0	0	0	0
2	0	1	1	0	0	1	0	1	1	0
3	1	0	1	0	0	0	0	0	0	0
4	1	1	0	0	0	0	0	0	1	1
5	1	0	0	0	0	0	1	1	0	1
6	0	0	1	1	1	0	1	0	0	1
7	0	0	0	0	1	0	1	0	0	1
8	0	0	0	1	0	1	0	1	0	0
9	1	1	1	0	0	0	0	1	0	1
10	0	0	0	0	0	0	0	1	0	0
11	1	1	0	0	0	1	0	0	0	0
12	0	1	1	1	0	1	0	0	1	0
13	0	1	0	0	0	1	1	1	0	1
14	0	0	0	0	1	0	0	1	0	1

We conducted an experiment to test the UNC. For this we used an Artix-7 FPGA AC701 Evaluation Kit. The response of the trained UNC to a random (test) failure vector (Table 3) is shown in Figure 9. This test vector has the structure given in the Table 4.

Table 3. Test failure vector.

Channel	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
State	1	0	0	1	0	0	0	0	0	1

Table 4. Failure vector.

Channel	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
State	*x*	*x*	*x*	*x*	*x*	*x*	*x*	*x*	*x*	*x*

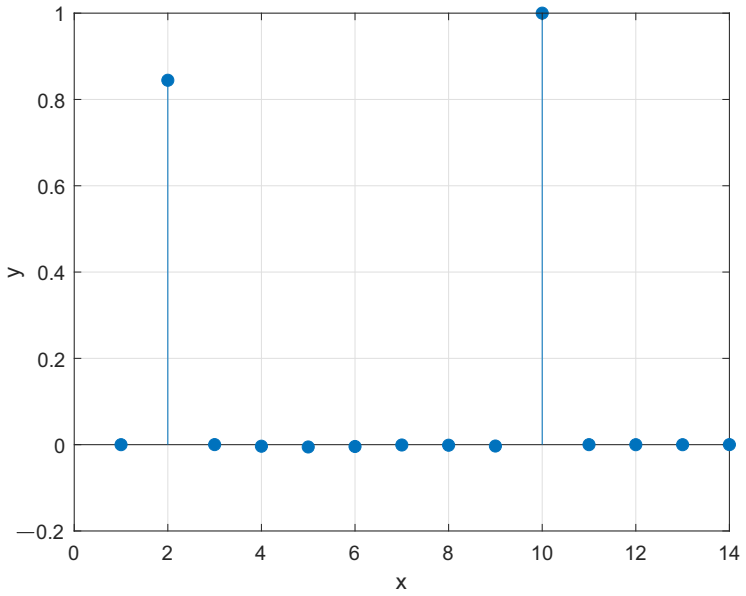

Figure 9. The response of the trained UNC to the test failure vector.

The functioning of the UNC modelled using Vivado 2018.2 (Xilinx), taking into account the delays in the FPGA, is presented in Figure 10.

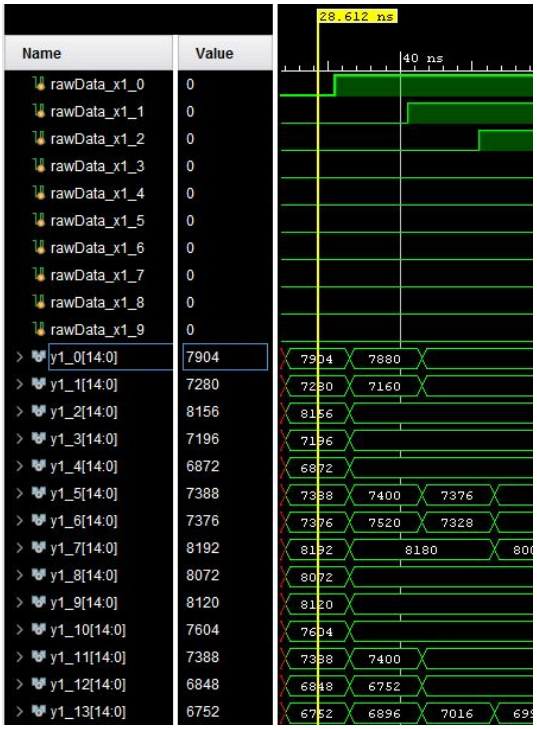

Figure 10. Functioning of the UNC modelled using an Artix-7 FPGA AC701 Evaluation Kit.

Figure 10 demonstrates that the UNC suggests the following solution for the test failure vector (Table 3): configuration No. 10 or configuration No. 2 from the knowledge base can be used. Configuration No. 10 is optimal, and therefore of the first priority.

Figure 10 shows that the response time of the UNC to the input vector is 28.612 ns. We should note that, when other development environments are used for the hardware implementation of the neural processor as a SoC (nVidia Jetson or Xilinx Versal) in order to solve the problem of image recognition in real time [30,31], the delays are from several milliseconds to tens of milliseconds. Thus, the suggested approach helps to minimise the delays, which is crucial for enhancing the reliability and fault tolerance of aircraft systems.

We should note that according to DO-254 and EUROCAE ED-80 standards, the suggested approach can only be used with DAL D and DAL E systems. For DAL A, DAL B, and DAL C systems, equipment with UNC should undergo flight testing and be certified. We should also note that in our experiment demonstrating the acceleration of the intellectual output for the dynamic reconfiguration problem, we used the UNC as a coprocessor together with an on-board computer with JetOS real time OS. This operating system was developed in full compliance with the ARINC 653 standard and is now being certified. The ARINC 653 standard sets the requirements for resource sharing and deterministic character of all the processes in an on-board real time OS. Since the architecture of the UNC does not include any elements or software processes that can "freeze", when used together with JetOS, it cannot affect the determined execution time for the functional tasks implemented by the OS.

8. Conclusions

The UNC described in this article is a universal computer which can perform a wide range of tasks, both deterministic (approximation, search for optimal route, search in a large data array) and nondeterministic, whose formalisation is rather problematic due to the lack of information and nonlinearity of internal dependences (classification, clustering, and decision making based on human experience). The use of the suggested neurocontroller as one of the blocks of a complex system (for instance an intellectual decision support system for aircrew) makes it possible to broaden the spectrum of problems the UCN is applicable to and make the decision making process more flexible and uniform from the point of view

of hardware implementation. It also ensures the required response time to the input signals (for example, emergencies), i.e., minimises the time needed to make a crucial decision.

We should also note that the introduction of hysteretic dynamics to the behaviour of the UNC helps to enhance its robustness to various types of noise, and improves the capacity of the intellectual output by adding degrees of freedom (i.e., parameters of the hysteresis model) which determine the nonlinearity of the dynamics of the whole ANN. It is also important that the use of the hysteresis activation function for a feedforward neural network results in dynamics and properties characteristic of recurrent ANNs. Thus, by changing the parameters of hysteresis activation functions we can make the ANN behave both as a feedforward neural network and as a recurrent neural network.

One of the many possibilities of using the presented universal neural network processor is the following example. Modern aircraft are increasingly equipped with special systems for reconfiguring the avionics complex, which can significantly increase their reliability in the event of an emergency. The reconfiguration process consists in finding the optimal configuration from the supervisor's memory (knowledge base). At the same time, the response time to equipment failure and the search interval are critical parameters. The use of a neural network computer to search for such an optimal configuration has great advantages in these parameters due to the high speed of information processing (due to the parallelism of processes). At the same time, the neural network is able to use hidden nonlinear patterns in the supervisor's knowledge base data, which increase the flexibility and depth of intelligent output. Thus, the use of the presented neural network computer as part of an on-board expert system is an urgent and promising task.

Author Contributions: Conceptualization, A.M.S. and N.I.S.; Data curation, V.V.K.; Investigation, A.M.S.; Methodology, E.Y.Z.; Project administration, N.I.S.; Resources, V.V.K.; Writing—original draft, A.M.S.; Writing—review & editing, E.Y.Z. All authors have read and agreed to the published version of the manuscript.

Funding: Research was conducted with financial support of the Russian Federation Ministry of Higher Education and Science within the scope of agreement No. 075-11-2020-024.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Informed consent was obtained from all subjects involved in the study. Written informed consent has been obtained from the patient(s) to publish this paper.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

OCN	Onboard computer network
DME	Distributed modular electronics
IMA	Integrated modular avionics
ANN	Artificial neural network
UNC	Universal neural computer
NIM	Neuron interaction matrix
AFV	Activation function vector
AF	Activation function
HAF	Hysteresis activation function

References

1. Moir, I. *Military Avionics Systems*; John Wiley & Sons: Hoboken, NJ, USA, 2019.
2. Spitzer, C.R. *Avionics: Elements, Software and Functions*; CRC Press: Boca Raton, FL, USA, 2018.
3. Kovalov, A.; Franz, T.; Watolla, H.; Vishav, V.; Gerndt, A.; Lüdtke, D. Model-Based Reconfiguration Planning for a Distributed On-board Computer. In Proceedings of the 12th System Analysis and Modelling Conference, Montreal, QC, Canada, 19–20 October 2020; pp. 55–62.

4. Annighöfer, B.; Thielecke, F. A systems architecting framework for optimal distributed integrated modular avionics architectures. *CEAS Aeronaut. J.* **2015**, *6*, 485–496. [CrossRef]
5. Zhou, X.; Xiong, H.; Feng, H.E. Hybrid partition-and network-level scheduling design for distributed integrated modular avionics systems. *Chin. J. Aeronaut.* **2020**, *33*, 308–323. [CrossRef]
6. Neretin, E.S.; Lunev, E.M.; Ivanov, A.S.; Budkov, A.S. Application of distributed integrated modular avionics concept for perspective aircraft equipment control systems. *J. Phys. Conf. Ser.* **2019**, *1353*, 012005. [CrossRef]
7. Hitt, E.F.; Mulcare, D. Fault-tolerant avionic. In *Digital Avionics Handbook*; CRC Press: Boca Raton, FL, USA, 2015.
8. Soloviev, A.M.; Semenov, M.E. Artificial Neural Networks with Hysteresis Activation Function. In Proceedings of the XVI All-Russian Scientific Engineering and Technical Conference "Neuroinformatics-2014", Moscow, Russia, 27–31 January 2014; Conference Proceedings, Part 1; NRNU MEPhI: Moscow, Russia, 2014; pp. 31–40
9. Kanwar, N.; Goswami, A.K.; Mishra, S.P. Design Issues in Artificial Neural Network. In Proceedings of the 2019 4th International Conference on Internet of Things: Smart Innovation and Usages (IoT-SIU), Ghaziabad, India, 18–19 April 2019; pp. 1–4.
10. Hayou, S.; Doucet, A.; Rousseau, J. On the impact of the activation function on deep neural networks training. In Proceedings of the International Conference on Machine Learning, Long Beach, CA, USA, 10–15 May 2019; pp. 2672–2680.
11. Litak, G.; Margielewicz, J.; Gąska, D.; Rysak, A.; Trigona, C. On Theoretical and Numerical Aspects of Bifurcations and Hysteresis Effects in Kinetic Energy Harvesters. *Sensors* **2022**, *22*, 381. [CrossRef]
12. Borzunov, S.V.; Semenov, M.E.; Sel'vesyuk, N.I.; Meleshenko, P.A. Hysteretic converters with stochastic parameters. *Math. Models Comput. Simul.* **2020**, *12*, 164–175. [CrossRef]
13. Semenov, M.E.; Reshetova, O.O.; Borzunov, S.V.; Meleshenko, P.A. Self-oscillations in a system with hysteresis: The small parameter approach. *Eur. Phys. J. Spec. Top.* **2021**, *230*, 3565–3571. [CrossRef]
14. Semenov, M.E.; Solovyov, A.M.; Meleshenko, P.A.; Reshetova, O.O. Efficiency of hysteretic damper in oscillating systems. *Math. Model. Nat. Phenom.* **2020**, *15*, 43. [CrossRef]
15. Semenov, M.E.; Reshetova, O.O.; Tolkachev, A.V.; Solovyov, A.M.; Meleshenko, P.A. Oscillations under hysteretic conditions: From simple oscillator to discrete Sine-Gordon model. *Top. Nonlin. Mech. Phys.* **2019**, *228*, 229–253.
16. Medvedskii, A.L.; Meleshenko, P.A.; Nesterov, V.A.; Reshetova, O.O.; Semenov, M.E.; Solovyov, A.M. Unstable Oscillating Systems with Hysteresis: Problems of Stabilization and Control. *J. Comput. Syst. Sci. Int.* **2020**, *59*, 533–556 . [CrossRef]
17. Medsker, L.; Jain, L.C. *Recurrent Neural Networks: Design and Applications*; CRC Press: Boca Raton, FL, USA, 1999.
18. Ikhouane, F.; Rodellar, J. *Systems with Hysteresis: Analysis, Identification and Control Using the Bouc-Wen Model*; John Wiley & Sons: Hoboken, NJ, USA, 2007.
19. Solovyov, A.M.; Semenov, M.E.; Meleshenko, P.A.; Barsukov, A.I. Bouc-Wen model of hysteretic damping. *Procedia Eng.* **2017**, *201*, 549–555. [CrossRef]
20. Semenov, M.E.; Solovyov, A.M.; Meleshenko, P.A.; Kanishcheva, O.I. Stabilization of a Flexible Inverted Pendulum via Hysteresis Control: The Bouc-Wen Approach. In *Vibration Engineering and Technology of Machinery*; Springer: Cham, Switzerland, 2021; pp. 267–279.
21. Ismail, M.; Ikhouane, F.; Rodellar, J. The hysteresis Bouc-Wen model, a survey. *Arch. Comput. Methods Eng.* **2009**, *16*, 161–188. [CrossRef]
22. Krasnosel'skii, M.A.; Pokrovskii, A.V. *Systems with Hysteresis*; Springer: Berlin/Heidelberg, Germany, 1989.
23. Mayergoyz, I.D. Dynamic Preisach models of hysteresis. *IEEE Trans. Magn.* **1988**, *24*, 2925–2927. [CrossRef]
24. Kuczmann, M. Dynamic Preisach hysteresis model. *J. Adv. Res. Phys.* **2010**, *1*, 011003.
25. Iyer, R.V.; Tan, X.; Krishnaprasad, P.S. Approximate inversion of the Preisach hysteresis operator with application to control of smart actuators. *IEEE Trans. Autom. Control* **2005**, *50*, 798–810. [CrossRef]
26. Hussain, S.; Lowther, D.A. An efficient implementation of the classical Preisach model. *IEEE Trans. Magn.* **2017**, *54*, 1–4. [CrossRef]
27. Semenov, M.E.; Borzunov, S.V.; Meleshenko, P.A. Stochastic Preisach operator: Definition within the design approach. *Nonlinear Dyn.* **2020**, *101*, 2599–2614 . [CrossRef]
28. Krejčí, P.; O'Kane, J.P.; Pokrovskii, A.; Rachinskii, D. Properties of solutions to a class of differential models incorporating Preisach hysteresis operator. *Phys. D Nonlinear Phenom.* **2012**, *241*, 2010–2028. [CrossRef]
29. Solovyov, A.M.; Semenov, M.E.; Selvesyuk, N.I.; Kosyanchuk, V.V.; Zybin, E.Y.; Glasov, V.V. Dynamic Reconfiguration of a Distributed Information-Computer Network of an aircraft. In Proceedings of the International Conference for Information Systems and Design, Virtual, 3–5 September 2021; pp. 120–133.
30. Gh, M.B. A novel Face Recognition System based on Jetson Nano developer kit. *IOP Conf. Ser. Mater. Sci. Eng.* **2020**, *928*, 032051.
31. Kaarmukilan, S.P.; Hazarika, A.; Poddar, S.; Rahaman, H. An accelerated prototype with movidius neural compute stick for real-time object detection. In Proceedings of the 2020 International Symposium on Devices, Circuits and Systems, Higashi-Hiroshima, Japan, 6–8 March 2020; pp. 1–5.