drones

MDPI

Article

An Approach to the Implementation of a Neural Network for Cryptographic Protection of Data Transmission at UAV

Ivan Tsmots [1], Vasyl Teslyuk [1,*], Andrzej Łukaszewicz [2,*], Yurii Lukashchuk [1], Iryna Kazymyra [1], Andriy Holovatyy [3] and Yurii Opotyak [1]

[1] Department of Automated Control Systems, Lviv Polytechnic National University, 79013 Lviv, Ukraine; ivan.h.tsmots@lpnu.ua (I.T.); urijlukas@gmail.com (Y.L.); iryna.y.kazymyra@lpnu.ua (I.K.); yurii.v.opotiak@lpnu.ua (Y.O.)
[2] Department of Machine Design and Exploitation, Faculty of Mechanical Engineering, Bialystok University of Technology, 15-351 Bialystok, Poland
[3] Department of Computer-Aided Design Systems, Lviv Polytechnic National University, 79013 Lviv, Ukraine; andrii.i.holovatyi@lpnu.ua
* Correspondence: vasyl.m.teslyuk@lpnu.ua (V.T.); a.lukaszewicz@pb.edu.pl (A.Ł.)

Abstract: An approach to the implementation of a neural network for real-time cryptographic data protection with symmetric keys oriented on embedded systems is presented. This approach is valuable, especially for onboard communication systems in unmanned aerial vehicles (UAV), because of its suitability for hardware implementation. In this study, we evaluate the possibility of building such a system in hardware implementation at FPGA. Onboard implementation-oriented information technology of real-time neuro-like cryptographic data protection with symmetric keys (masking codes, neural network architecture, and matrix of weighting coefficients) has been developed. Due to the pre-calculation of matrices of weighting coefficients and tables of macro-partial products and the use of tabular-algorithmic implementation of neuro-like elements and dynamic change of keys, it provides increased cryptographic stability and hardware–software implementation on FPGA. The table-algorithmic method of calculating the scalar product has been improved. By bringing the weighting coefficients to the greatest common order, pre-computing the tables of macro-partial products, and using operations of memory read, fixed-point addition, and shift operations instead of floating-point multiplication and addition operations, it provides a reduction in hardware costs for its implementation and calculation time as well. Using a processor core supplemented with specialized hardware modules for calculating the scalar product, a system of neural network cryptographic data protection in real-time has been developed, which, due to the combination of universal and specialized approaches, software, and hardware, ensures the effective implementation of neuro-like algorithms for cryptographic encryption and decryption of data in real-time. The specialized hardware for neural network cryptographic data encryption was developed using VHDL for equipment programming in the Quartus II development environment ver. 13.1 and the appropriate libraries and implemented on the basis of the FPGA EP3C16F484C6 Cyclone III family, and it requires 3053 logic elements and 745 registers. The execution time of exclusively software realization of NN cryptographic data encryption procedure using a NanoPi Duo microcomputer based on the Allwinner Cortex-A7 H2+ SoC was about 20 ms. The hardware–software implementation of the encryption, taking into account the pre-calculations and settings, requires about 1 msec, including hardware encryption on the FPGA of four 2-bit inputs, which is performed in 160 nanoseconds.

Keywords: neural network (NN); cryptographic protection; UAV; UAS; onboard system; encryption; decryption; tabular-algorithmic method; scalar product; real time

Citation: Tsmots, I.; Teslyuk, V.; Łukaszewicz, A.; Lukashchuk, Y.; Kazymyra, I.; Holovatyy, A.; Opotyak, Y. An Approach to the Implementation of a Neural Network for Cryptographic Protection of Data Transmission at UAV. *Drones* **2023**, *7*, 507. https://doi.org/10.3390/drones7080507

Academic Editor: Diego González-Aguilera

Received: 1 July 2023
Revised: 27 July 2023
Accepted: 30 July 2023
Published: 2 August 2023

1. Introduction

Cryptographic protection of data transmission between the UAV and the remote-control centre is important to ensure the confidentiality and integrity of the information

transmitted. However, there are certain practical problems and influencing factors that must be taken into account when developing a cryptographic protection system for data transmission, specifically for UAVs. The main ones are limited computing resources; ensuring requirements for energy consumption, dimensions, and weight of equipment; provision of data transmission between the UAV and the remote control centre in real time; ensuring the requirements for the cost of the system of cryptographic protection of data transmission; algorithmic problems of cryptographic algorithms, which are associated with security vulnerabilities and malicious attacks; limitation of physical access to means of cryptographic protection; organization of an effective key management system; damage or loss of data in the wireless transmission channel; and change in altitude and environmental conditions that affect the quality of communication and data transmission.

The key problem is to guarantee the cryptographic security of data transmission in the management of UAVs [1,2], intelligent robots [3], microsatellites [4], and various mobile transport systems [5]. Due to the security vulnerabilities of UAVs and illegal and malicious attacks against UAVs, especially against communication data and UAV control, solutions to prevent such attacks are needed, and one of them is to encrypt UAV's communication data [6–8]. Unmanned aerial vehicles (UAVs) must be energy-efficient, especially in data processing, because of limited battery capacity [9]. Solving this problem requires the development of neural network (NN) technology [10–12] for cryptographic data protection, which is focused on use in UAV onboard communication systems. When developing onboard cryptographic data protection systems, it is necessary to provide a real-time mode, increase cryptographic resistance and noise immunity, and reduce power consumption, weight, size and cost [13–23]. The usage of an auto-associative NN of direct propagation, which is trained on the basis of the principal components analysis, helps to conform to such requirements. A specific feature of such neural networks is the ability of weight pre-calculation and to apply the tabular-algorithmic method for the implementation of neuro-like elements using the basis of elementary arithmetic operations. For NN cryptographic encryption and decryption of data, it is proposed to use symmetric keys, which include masking codes, NN architecture and a matrix of weights [24,25].

Through the extensive use of a modern component base and the development of new VLSI methods, algorithms, and structures, high technical and operational rates of onboard cryptographic data protection systems are achieved. Onboard systems for NN cryptographic protection of data must have variable hardware for rapid changes in NN architecture. The use of modern element base (microcontrollers, programmable logic integrated circuits FPGA) in the development of onboard and embedded systems makes it possible to reduce their weight, size, and power consumption [26,27] and, in the development of onboard systems of NN cryptographic, data protection provides a quick change of encryption and decryption keys.

NN cryptographic encryption and decryption of data in real-time is achieved through the application of parallel encryption and decryption of data, hardware implementation of neuro-like elements based on a multi-operand approach and macro-partial products tables.

Therefore, the urgent problem is to propose an approach to the implementation of NN for cryptographic data protection, focused on implementing onboard systems with high technical and operational characteristics. The objective of the work is to study how to implement the onboard NN for real-time cryptographic data protection. In order to achieve this goal, the following tasks have to be solved:

- Development of the approach to NN cryptographic data protection;
- Development of the structure of the system of NN cryptographic protection and real-time data transmission;
- Development of components of onboard systems of NN cryptographic encryption–decryption of data;
- Implementation of the specialized hardware components of NN cryptographic data encryption on FPGA.

This article is structured as follows. In the Introduction, we have considered the problem relevancy and the main objectives of this research. Section 2 contains a brief review of the related works (the research context). The structure of NN technology for cryptographic data protection is described in Section 3, and the main stages of NN data encryption/decryption are considered here as well. Section 4 presents the structure of the system for NN cryptographic data protection and transmission (stationery and UAV onboard parts) developed using an integrated approach. The components of the onboard system for NN cryptographic data encryption and decryption are proposed in Section 5, and the diagrams for the specialized hardware are given.

2. Related Works

The study of the main trends in the area of UAV onboard systems development for real-time cryptographic data protection shows that NN methods are increasingly used for performing data encryption and decryption in such systems [28–32]. These publications show that the implementation of NN methods of cryptographic data protection is generally performed by software. The critical drawback of software implementation of NN cryptographic data protection is the difficulty of providing a real-time mode and the constraints imposed on onboard systems in terms of weight, size, power consumption, and cost.

The possibilities of adapting the auto-associative NN with non-iterative learning for data protection tasks are considered in [28–32]. The peculiarity of the functioning of such an NN is the preliminary calculation of weights as a result of its training based on the principal component analysis (PCA). In this case, a system of eigenvectors is used that corresponds to the eigenvalues of the covariance matrix of input data [33]. To encrypt and decrypt data, the auto-associative NN with pre-calculated weights is applied. In [34], it was shown that for the masking codes, the architecture of the NN and the matrix of weights are the basis for cryptographic encryption and decryption of data in neural networks.

Publications [35–37] are devoted to the hardware implementation of neural networks, showing that they are based on neural elements. The feature of such neural elements is that the number of inputs and their bit length are determined by the NN architecture, which is one of the characteristics of the data encryption key. The main operation of the neuro element is the calculation of the scalar product using pre-calculated weights.

In [37–39], the methods for calculating the scalar product using the basis of elementary arithmetic operations, addition and shift, are considered. The peculiarity of these methods is the formation of macro-partial products, their shift, and addition to the previously accumulated amount. Hardware implementation of such methods requires significant equipment costs. The implementation of a tabular-algorithmic method for calculating the scalar product, which is reduced to the operations of reading macro-partial products, addition and shift, requires fewer equipment costs and less computation time. The disadvantage of this method is that it is limited to fixed-point data format (for input data and weights).

Analyzing the works [31,40,41], it can be noted that NN tools for cryptographic symmetric encryption and decryption of data [42] are implemented on the basis of microprocessors supplemented by hardware that implements time-consuming computational operations using FPGA [43]. The high speed of NN tools for cryptographic encryption and decryption of data is achieved through parallelization, pipeline computing processes, and hardware implementation of neural elements. The disadvantage of the existing NN tools for cryptographic data protection is the difficulty of changing the encryption and decryption key rapidly.

3. The Approach to NN Implementation for Cryptographic Data Protection

3.1. Structure of NN Technology of Cryptographic Data Protection

The implementation of NN for cryptographic protection of data transmission is focused on hardware and software implementation with high technical and operational

characteristics. It is proposed that to carry out such implementation on the basis of an integrated approach includes the following:

- Research and development of theoretical foundations of neuro-like cryptographic data protection;
- Research and development of new algorithms and structures of neuro-like encryption and decryption of data focused on modern element base;
- Modern element base with the ability to program the structure;
- The means for automated design of software and hardware.

Figure 1 shows the developed structure of NN technology for cryptographic data protection, which is focused on hardware implementation and provides encryption with symmetric keys. When implementing the symmetric cryptosystem, the encryption key and the decryption key are the same, or the decryption key is easily calculated from the encryption key.

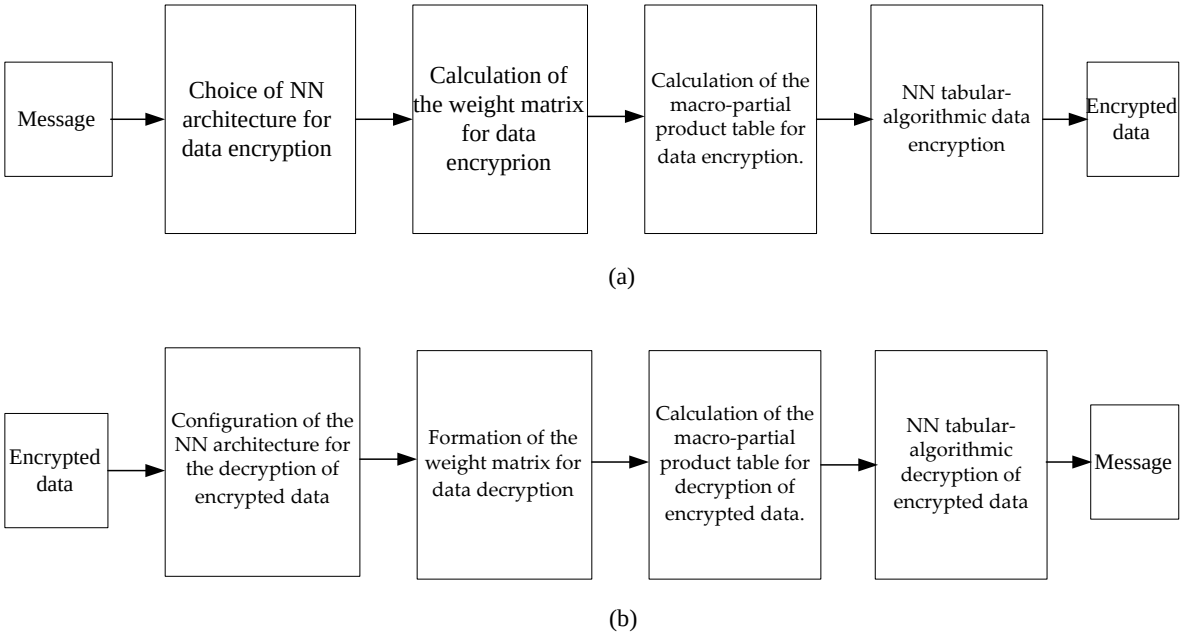

Figure 1. Structure of NN technology for cryptographic data protection: (**a**) the process of data encryption; (**b**) the process of decrypting data.

For hardware implementation, the proposed technology is based on the selection of auto-associative neural networks, which are trained non-iteratively. This allows us to calculate the matrix of weighting coefficients in advance and to store them in the lookup tables since they will be fixed for the selected NN configuration. The calculation of the output of the neuro-like element of this NN can be represented as the sum of the products of the weighting coefficients and the input data to be encrypted. To implement a quick calculation on the FPGA of the product of the fixed weighting coefficients and the input data, a table-algorithmic method of their calculation is applied. The tabular-algorithmic method makes it possible to implement high technical and operational characteristics of data encryption–decryption tools. A combination of these approaches ensures the effective implementation of FPGAs. The details of the above-mentioned steps are described further in the article.

A specific feature of the proposed technology is the pre-calculation of matrices of weighting coefficients for possible variants of neural networks and the use of the tabular-algorithmic method for the implementation of neuro elements. Such pre-calculation of matrices and tables provides the possibility of dynamically changing keys and, accordingly, increasing cryptographic stability. The use of elementary arithmetic operations in fixed-

point format for the hardware tabular-algorithmic implementation of the neuro element provides a reduction in hardware costs when building specialized hardware modules.

Ensuring the real-time mode when encrypting (decrypting) data can be achieved by selecting the necessary number of specialized hardware modules and reducing the time of calculating the scalar product in such modules. It is possible to reduce the time of calculating the scalar product by using an algorithm that provides for submitting g bit slices to the address inputs of g tables of macro-partial products. Using such an algorithm reduces the time of calculating the scalar product by g times.

3.2. Main Stages of NN Encryption

The encryption uses a key consisting of N neurons in the NN, a weight matrix, and masking operations. The main stages of message encryption are considered below.

Choice of NN architecture. The number of neuro elements N, the number of inputs k, and the bit inputs m determine the architecture of the NN. The number of neural elements is defined according to the following formula:

$$N = \frac{n}{m}, \tag{1}$$

where n is the bit length of the message, and m is the bit length of the inputs.

The incoming messages, which are encrypted, can have different bit lengths (n) and different inputs number (k), which is equal to the number of neuro elements N. The architecture of the NN depends on the value of the bit length of the message n and the number of inputs k. Such configuration of the NN architecture is available to encrypt the $n = 16$ bit message: $m = 2$, $k = 8$, $N = 8$; $m = 4$, $k = 4$, $N = 4$; $m = 8$, $k = 2$, $N = 2$, in case of $n = 24$ they are: $m = 2$, $k = 12$, $N = 12$; $m = 3$, $k = 8$, $N = 8$; $m = 4$, $k = 6$, $N = 6$, $m = 6$, $k = 4$, $N = 4$; $m = 8$, $k = 3$, $N = 3$; $m = 12$, $k = 2$, $N = 2$.

Calculation of the weight matrix. For data encryption–decryption, we will use an auto-associative NN, which learns non-iteratively using the principal components analysis (PCA), which performs a linear transformation following the formula

$$\overline{y} = W \cdot \overline{x} \tag{2}$$

According to Equation (2), the matrix $W \in R^{n \times n}$ is used to convert the input vector $\overline{x} \in R^n$ into the output vector $\overline{y} \in R^n$. The conversion is as follows. A system of linearly independent vectors selects an orthonormal system of eigenvectors corresponding to the eigenvalues of the covariance matrix of the input data.

The input data is a set of N vectors $\overline{x}_j, j = 1, \ldots N$, with dimension n, $\overline{x}_j = (x_{j1}, x_{j2}, \ldots, x_{jn})$:

$$X = (\overline{x}_1, \overline{x}_2, \ldots, \overline{x}_N)^t. \tag{3}$$

For N vectors, the autocovariance matrix $\overline{x}_j$ is

$$R = X^t \cdot X, \tag{4}$$

where each of the elements is expressed by

$$r_{jl} = \sum_{i=1}^{N} \overline{x}_{jl} \overline{x}_{il} = \sum_{i=1}^{N} (\overline{x}_{ji} - \mu_j)(\overline{x}_{il} - \mu_l), \tag{5}$$

where $j, l = 1, 2, \ldots, n$, and μ_j; μ_l—mathematical expectations of vectors $\overline{x}_j$, $\overline{x}_l$.

The eigenvalues of R symmetric non-negative matrix are real and positive numbers. They are arranged in descending order $\lambda_1 > \lambda_2 > \ldots > \lambda_n$. Similarly, the eigenvectors corresponding to λ_i are placed. Therefore, a linear transformation (2) is defined by the matrix W. Here, $\overline{y} = (y_1, y_2, \ldots, y_n)$ is a vector of the PCA principal components corresponding to the input data vector $\overline{x}$. The number of principal components vectors N conforms with

the number of input data vectors $\overline{x}$ [29]. The matrix of weights used to encrypt the data is as follows:

$$\begin{vmatrix} W_{11} & W_{12} & \cdots & W_{1k} \\ W_{21} & W_{22} & \cdots & W_{2k} \\ \vdots & \vdots & \cdots & \vdots \\ W_{N1} & W_{N2} & \cdots & W_{Nk} \end{vmatrix}. \quad (6)$$

The basic operation of the NN used to encrypt data is the operation of calculating the scalar product. This operation should be implemented using the tabular-algorithmic method because the matrix of weights W_{js}, where $j = 1, \ldots, N$, $s = 1, \ldots, k$, is pre-calculated.

Calculation of the table of macro-partial products for data encryption. The specificity of the scalar product calculation operation used in data encryption is that the weights are pre-calculated (constants) and set in floating point format, and the input data X_j is in fixed point format with its fixing before the high digit of a number. The scalar product is calculated by means of the tabular-algorithmic method according to the formula

$$Z = \sum_{j=1}^{N} W_j X_j = \sum_{i=1}^{n} 2^{-i} \sum_{j=1}^{N} W_j X_{ji} = \sum_{i=1}^{n} 2^{-i} \sum_{j=1}^{N} P_{ji} = \sum_{i=1}^{n} 2^{-i} P_{Mi}, \quad (7)$$

where N is the number of products; X_j is the input data; W_j is the j-th weight coefficient; n is the bit length of the input data; P_{ij} is the partial product; and P_{Mi} is the macro-partial product formed by adding N partial products P_{ij}, as follows: $P_{Mi} = \sum_{j=1}^{N} P_{ji}$.

Formation of the tables of macro-partial products for floating-point weights $W_j = w_j 2^{m_{W_j}}$ (where w_j is the mantissa of W_j weight coefficient; m_{W_j} is the order of W_j weight coefficient) foresees the following operations to be performed:

- Defining the largest common order of weights m_{Wmaxc};
- Calculation of the order difference for each W_j weight coefficient: $\Delta m_{W_j} = m_{Wmaxc} - m_{W_j}$;
- Shift the mantissa w_j to the right by a difference of orders Δm_{W_j};
- Calculation of P_{Mi} macro-partial product for the case when $x_{1i} = x_{2i} = x_{3i} = \ldots = x_{Ni} = 1$;
- Determining the number of overflow bits q in the P_{Mi} macro-partial product for the case when $x_{1i} = x_{2i} = x_{3i} = \ldots = x_{Ni} = 1$;
- Obtaining scalable mantissas w_j^h by shifting them to the right by the number of overflow bits;
- Adding to the largest common order of weight m_{Wmaxc} the number of overflow bits q, as per the formula $m_j = m_{Wmaxc} + q$.

The table of macro-partial products is calculated by the formula

$$P_{Mi} = \begin{cases} 0, \ if\ x_{1i} = x_{2i} = x_{3i} = \ldots = x_{Ni} = 0 \\ w_1^h, \ if\ x_{1i} = 1, \ x_{2i} = x_{3i} = \ldots = x_{Ni} = 0 \\ w_2^h, \ if\ x_{1i} = 0, x_{2i} = 1, x_{3i} = \ldots = x_{Ni} = 0 \\ w_1^h + w_2^h, \ if\ x_{1i} = 1, x_{2i} = 1, x_{3i} = \ldots = x_{Ni} = 0 \\ \vdots \\ w_2^h + \ldots + w_N^h, \ if\ x_{1i} = 0, x_{2i} = x_{3i} = \ldots = x_{Ni} = 1 \\ w_1^h + w_2^h + \ldots + w_N^h, \ if\ x_{1i} = x_{2i} = x_{3i} = \ldots = x_{Ni} = 1 \end{cases}, \quad (8)$$

where $x_{1i}, x_{2i}, x_{3i}, \ldots, x_{Ni}$ address inputs of the table, and w_j^h is the mantissa of W_j weight coefficient brought to the greatest common order.

The possible combinations number of P_{Mi} macro-partial products and the table size are as follows:

$$Q = 2^N. \quad (9)$$

By dividing all N products by parts $N1$ and $N2$, we can reduce the table size. For each of these parts, separate tables of macro-partial products P_{N1Mi} and P_{N2Mi} are formed and stored in separate memory blocks or a single memory block. When using two memory blocks, parts of the macro-partial products P_{N1Mi} and P_{N2Mi} are read in one clock cycle and in one memory block—in two clock cycles. The sum of two macro-partial products P_{N1Mi} and P_{N2Mi} gives us the macro-partial product P_{Mi}.

NN tabular-algorithmic data encryption. During the training of the NN, the matrix of weights W is determined. Figure 2 shows the structure of auto-associative NN used for data encryption. Here, M_j is the mask for the j-th input, x_j is the j-th input data, and XOR is the masking operation using the exclusive OR elements.

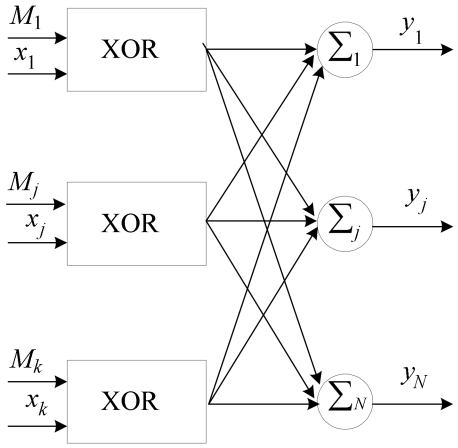

Figure 2. The structure of the data encryption NN.

To perform the NN data encryption, we multiply the W matrix by the input data vector x according to the formula

$$y_j = \begin{vmatrix} W_{11} & W_{12} & \cdots & W_{1k} \\ W_{21} & W_{22} & \cdots & W_{2k} \\ \vdots & \vdots & \cdots & \vdots \\ W_{N1} & W_{N2} & \cdots & W_{Nk} \end{vmatrix} \times \begin{matrix} x_1 \\ x_2 \\ \vdots \\ x_k \end{matrix}. \tag{10}$$

The multiplication of the matrix of weights W by the vector of input data x is reduced to performing N scalar product calculations:

$$y_j = \sum_{s=1}^{k} W_{js} x_s \tag{11}$$

where k—number of products, $s = 1, 2, \ldots, k; j = 1, 2, \ldots, N$.

The calculation of scalar products will be achieved using the tabular-algorithmic method, where the weights W_{js} are set in floating-point format, and the input data x_s is in a fixed-point format with fixation before the highest digit. Tabular-algorithmic calculation of the mantissa of the scalar product is reduced to reading the macro-partial product P_{Mi} from the j-th table (memory) at the address corresponding to the i-th bit slice of N input data, and adding it to the before accumulated sums according to

$$y_{Mji} = 2^{-1} y_{Mj(i-1)} + P_{Mji}, \tag{12}$$

where $y_{Mj0} = 0$, $i = 1, \ldots, m$, and m is the bit length of the input data. The number of tables of macro-partial products corresponds to N—the number of rows of the matrix (10). The result of calculating the scalar product y_j consists of the mantissa y_{Mj} and the order m_j.

The time required to compute the mantissa of the scalar product (SP) is determined by the formula

$$t_{SP} = m\left(t_{table} + t_{reg} + t_{add}\right), \tag{13}$$

where t_{SP} is the time of calculation of the scalar product, t_{table} is the time of reading from the table (memory), t_{reg} is the time of reading (writing) from the register, and t_{add} is the time of adding.

Data encryption can be performed either sequentially or in parallel, depending on the speed required. In the case of sequential encryption, the encryption time is the result of the formula

$$t_{encrypt} = Nm\left(t_{table} + t_{reg} + t_{add}\right), \tag{14}$$

where $t_{encrypt}$ is the time required for encryption. The encryption time can be reduced by performing N operations of calculating the scalar product in parallel.

As a result of NN data encryption, we obtain N encrypted data in the form $y_j = y_{Mj}2^{m_j}$, where y_{Mj} is the mantissa at the j-th output, and m_j is the order value at the j-th output. It is advisable to bring all encrypted data to the highest common order for transmission, and such reduction to the greatest common order is performed in three stages:

- Define the greatest order m_{encr};
- For each encrypted data y_j, calculate the difference between the orders $\Delta m_j = m_{encr} - m_j$;
- By performing shift of the mantissa y_{Mj} to the right by the difference of orders Δm_j, we obtain mantissa of the encrypted data y^h_{Mj} reduced to the greatest common order.

The mantissa of the encrypted data y^h_{Mj} reduced to the largest common order and the largest common order m_{encr} are sent for decryption.

3.3. The Main Stages of NN Cryptographic Data Decryption

Now the encrypted data presented by mantissa y^h_{Mj} reduced to the largest common order m_{encr} need to be decrypted. The encrypted data will be decrypted according to the following procedure.

Configuration of the NN architecture for the decryption of encrypted data. The architecture of the NN for the decryption of encrypted data, in terms of the number of neural elements, is the same as the architecture of the NN used for the encryption of data. In this NN, the number of inputs and the number of neurons corresponds to the number of the encrypted mantissa y^h_{Mj}. The NN architecture used to decrypt encrypted data is presented in Figure 3.

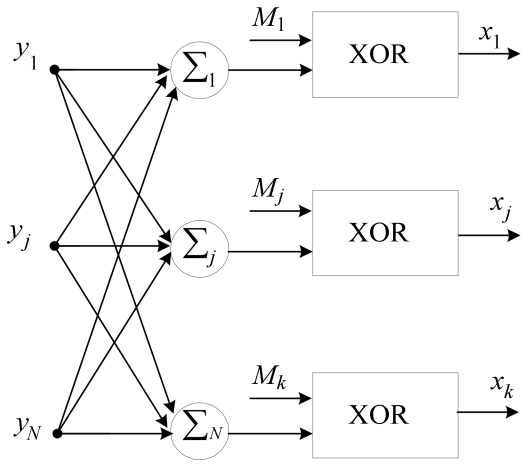

Figure 3. The NN architecture for decryption of encrypted data.

The bit rate of the inputs during decryption corresponds to the bit rate of the encrypted mantissa y^h_{Mj}. Its value determines the decryption time, and to reduce it, the lower bits of the mantissa may be discarded because they will not affect the original message recovery.

Formation of the weight matrix. The matrix of weights for decrypting encrypted data is formed from a matrix of weights for encrypting input data by transposing it:

$$\begin{vmatrix} W_{11} & W_{12} & \cdots & W_{1k} \\ W_{21} & W_{22} & \cdots & W_{2k} \\ \vdots & \vdots & \cdots & \vdots \\ W_{N1} & W_{N2} & \cdots & W_{Nk} \end{vmatrix}^{T} = \begin{vmatrix} W_{11} & W_{21} & \cdots & W_{N1} \\ W_{12} & W_{22} & \cdots & W_{N2} \\ \vdots & \vdots & \cdots & \vdots \\ W_{1k} & W_{2k} & \cdots & W_{Nk} \end{vmatrix}. \quad (15)$$

The basic operation for the encryption of input data and decryption of encrypted data is the calculation of the scalar product, which is implemented using a tabular-algorithmic method.

Calculation of the table of macro-partial products for decryption of encrypted data. A specific feature of the scalar product calculation operation used to decrypt encrypted data is that the weights are pre-calculated (constants) and set in floating-point format, while the encrypted data y_j are received in block-floating-point format. The calculation of the scalar product using the tabular-algorithmic method is performed by Equation (7). Preparation and calculation of possible variants of macro-partial products are performed as in the previous case under Equation (8).

The amount of encrypted data determines the number of macro-partial products P_{Mi} and the size of the table. The largest common order m_{Pms} is computed for each table.

NN tabular-algorithmic decryption of encrypted data. The NN decryption is specified by multiplying the W matrix by the encrypted data vector y:

$$x_s = \begin{vmatrix} W_{11} & W_{21} & \cdots & W_{N1} \\ W_{12} & W_{22} & \cdots & W_{N2} \\ \vdots & \vdots & \cdots & \vdots \\ W_{1k} & W_{2k} & \cdots & W_{Nk} \end{vmatrix} \times \begin{matrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{matrix}. \quad (16)$$

The multiplication of the weights matrix W^T by the input data vector y is reduced to performing N scalar product calculations:

$$x_s = \sum_{j=1}^{N} W_{sj} y_j \quad (17)$$

where N is the number of products, and $s = 1, 2, \ldots, k; j = 1, 2, \ldots, N$.

Tabular-algorithmic calculation of the mantissa of the scalar product is reduced to reading the macro-partial product P_{Mi} from the table (memory) at the address corresponding to the i-th bit-slice of k input data, and adding it to the previously accumulated sums, according to the formula

$$x_{Msi} = 2^{-1} y_{Ms(i-1)} + P_{Msi}, \quad (18)$$

where $x_{s0} = 0, i = 1, \ldots, g$, and g is the bit rate of the encrypted data. The time necessary to calculate the scalar product mantissa is defined under the formula

$$t_{SP} = g\left(t_{table} + t_{reg} + t_{add}\right), \quad (19)$$

where t_{SP} is the time for scalar product calculation, t_{table} is the time for reading from a table (memory), t_{reg} is the time of reading (writing) from the register, and t_{add} is the time for adding. The result of the calculation of the x_s scalar product consists of a mantissa x_{Ms} and order, which is equal to $m_{decrs} = m_{PMs} + m_{encr}$.

At the output of the NN (see Figure 3), we obtain k decrypted data in the following form $x_s = x_{Ms} 2^{m_{decrs}}$, where x_{Ms} is the mantissa at the s-th output, and m_{decrs} is the value of the order at the s-th output. To obtain the input data, it is necessary to shift the s-th mantissa x_{Ms} by the value of the order m_{decrs}.

4. The Structure of the System for NN Cryptographic Data Protection and Transferring in Real-Time Mode

The development of the structure of the system for NN cryptographic data protection and transmission in real-time will be carried out using an integrated approach, which contains the following:

- Research and development of theoretical foundations of NN cryptographic data encryption and decryption;
- Development of new tabular-algorithmic algorithms and structures for NN cryptographic data encryption and decryption;
- Modern element base, development environment and computer-aided design tools.

A system for NN cryptographic data protection in real-time was developed using the following principles:

- Changeable composition of the equipment, which foresees the presence of the processor core and replaceable modules, with which the core adapts to the requirements of a particular application;
- Modularity, which involves the development of system components in the form of functionally complete devices;
- Pipeline and spatial parallelism in data encryption and decryption;
- The openness of the software, which provides opportunities for development and improvement, maximising the use of standard drivers and software;
- Specialising and adapting hardware and software to the structure of tabular algorithms for encrypting and decrypting data;
- The programmability of hardware module architecture through the use of programmable logic integrated circuits.

In order to provide neural-like encryption and decryption of data arrays in real time, it is necessary that encryption and decryption occur without accumulating delays. Encrypting (decrypting) an array of h messages in real time imposes a time limit for their encryption (decryption), which must meet the following:

$$ht_{E/De} \leq t_a, \tag{20}$$

where $t_{E/De}$ is the time of encryption (decryption) of one message, and t_a is the time of arrival of h messages, which is determined as follows:

$$t_a = \frac{hn}{F_d s n_k}, \tag{21}$$

where n is the bit rate of the message, s is the number of channels through which the message is received, n_k is the bit rate of the channels, and F_d is the frequency of message arrival.

Knowing the time t_a, it is possible to determine the encryption (decryption) time of one message $t_{E/De}$ according to the following formula:

$$t_{\text{Ш/Дш}} \leq \frac{n}{F_d s n_k}. \tag{22}$$

In the case of the NN approach to encryption (decryption), it is proposed to supplement the processor core with the specialized modules that implement neural elements in hardware to ensure real time. The number of specialized modules and the time of calculation of the scalar product in such modules should ensure the fulfilment of the condition $t_{E/De} \leq t_a/h$. It is possible to choose the time of calculation of the scalar product by using an algorithm that involves the use of q tables of macropartial products for calculation by applying q bit slices to their address inputs. The use of such an algorithm reduces the time of calculating the scalar product by q times.

The system of NN cryptographic real-time data protection and transmission consists of a stationary part, which is a remote-control centre, and a UAV onboard part. The structure of the stationary part of the system of NN cryptographic data protection and transmission is shown in Figure 4.

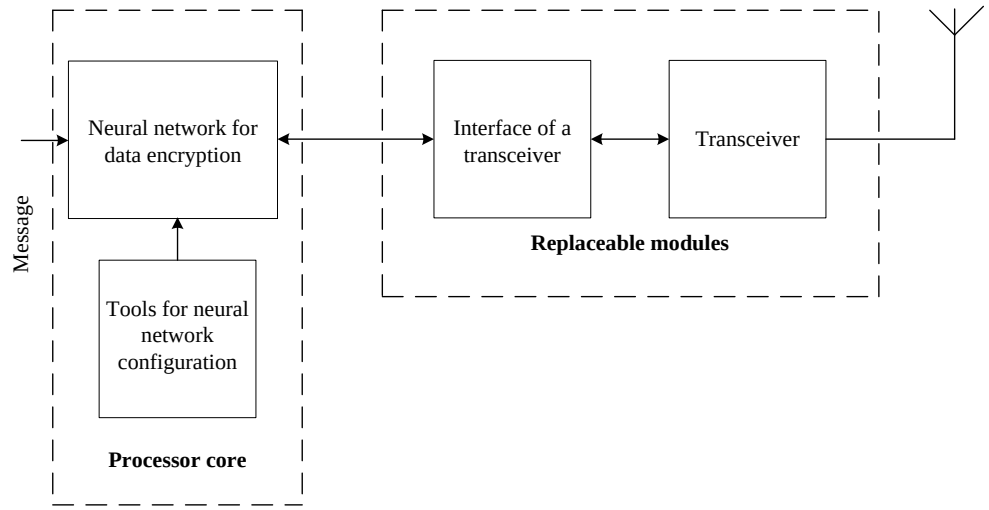

Figure 4. Structure of the stationary part of the system of NN cryptographic data protection and transmission.

The processor core of the remote-control center is implemented on the basis of a personal computer. The transceiver is used to transmit encrypted data; it communicates with the processor core through the interface based on a microcontroller.

The UAV onboard part of the system for NN cryptographic real-time data protection and transmission is implemented on the processor core, which is supplemented by dedicated hardware and software. The processor core of the UAV onboard part of the system is designed on a microcomputer. The structure of the onboard part of the system of NN cryptographic data protection and receiving is depicted in Figure 5.

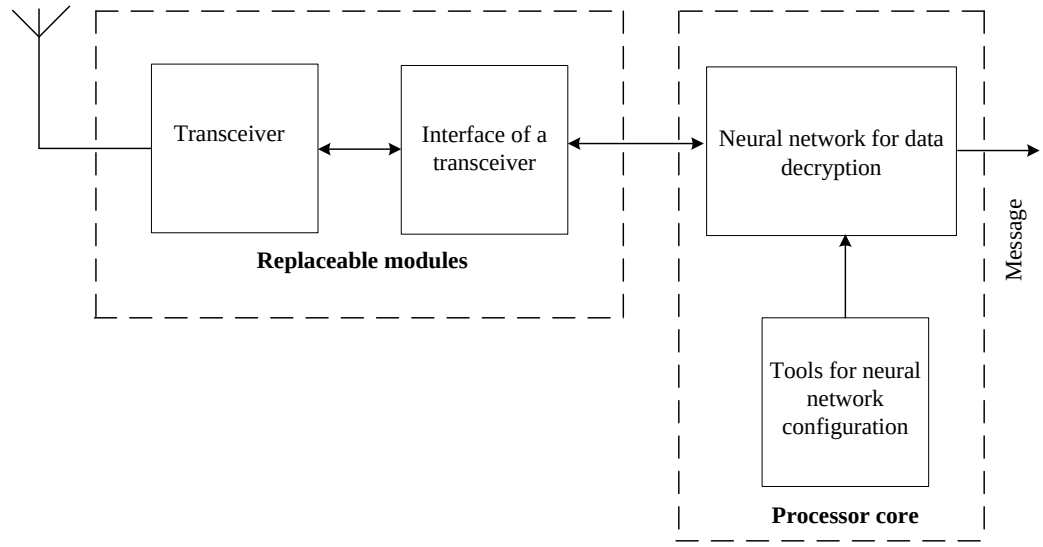

Figure 5. Structure of the UAV onboard part of the system of NN cryptographic data protection and transmission.

The effective implementation of NN encryption–decryption and encoding–decoding algorithms in real time is achieved by combining universal and customized software and hardware. The use of modern elements (microcomputer, microcontroller, FPGA) in the

development of the UAV onboard part ensures the accomplishment of the requirements for weight, dimensions and energy consumption.

The effectiveness of the system for NN cryptographic real-time data protection and transmission is directly associated with the choice of both hardware and software implementation.

5. Development of the Components of the Onboard System for NN Cryptographic Data Encryption and Decryption

In general, the problem of developing onboard systems for NN cryptographic encryption–decryption of data can be formulated as follows:

- To develop an algorithm for the onboard system of NN encryption–decryption of data and present it in the form of a specified flow graph;
- To design the structure of the onboard system for NN data encryption–decryption with the maximum efficiency of equipment use, taking into account all the limitations and providing real-time data processing;
- To determine the main characteristics of neural elements and carry out their synthesis;
- To choose exchange methods, determine the necessary connections and develop algorithms for exchange between system components;
- To determine the order of implementation in time of NN data encryption–decryption processes and develop algorithms for their management.

Components of the onboard system of NN cryptographic data encryption and decryption should provide the implementation of the selected NN, ability to change masks, and calculate matrices of weights W_j and tables of macro-partial products P_{Mi} for possible NN options. To effectively implement the components of the onboard system of NN cryptographic encryption–decryption of data, it is proposed to use hardware–software implementation of the algorithms based on a microcontroller supplemented by specialized hardware. The structure of the component of NN cryptographic data encryption, which meets such requirements, is presented in Figure 6, where MC is the microcontroller, MN is the mask node, MP is the macro-partial product, Rg is the register, and Add is the adder.

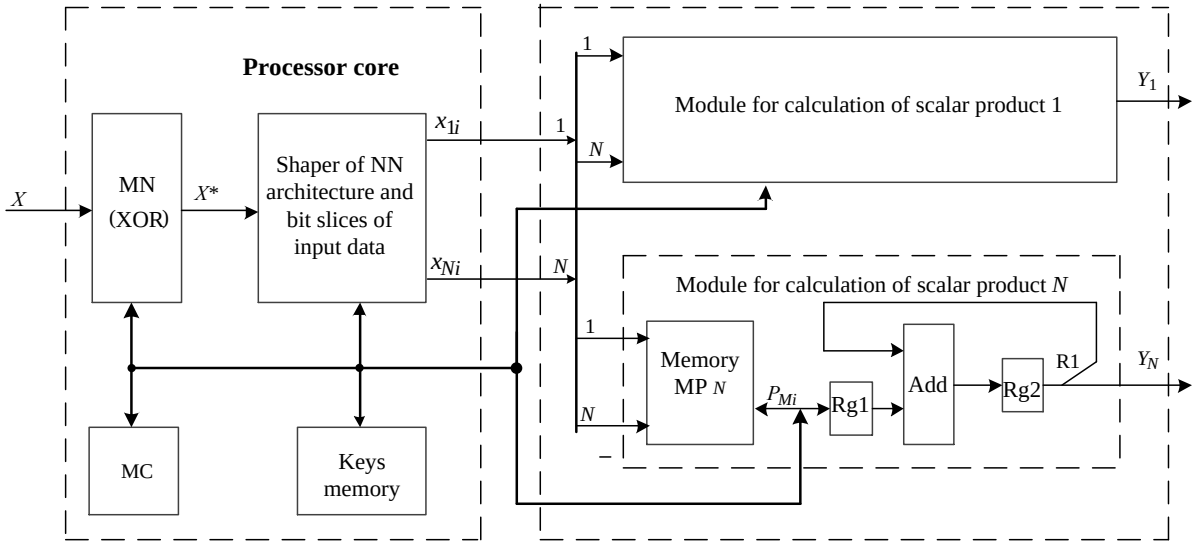

Figure 6. Structure of the component of NN cryptographic encryption of data.

The developed component of NN cryptographic data encryption has a variable composition of equipment, which is based on the core of the system and a set of modules for calculating the scalar product. The system core is constant for all applications and consists of microcontroller MC, mask node MN, keys memory, and module of the shaper of the NN architecture and bit slices of input data. The scalar product calculation modules implement

the basic operation of the tabular-algorithmic method of scalar product calculation under the formula

$$Z_i = 2^{-1} Z_{i-1} + P_{Mi}, \tag{23}$$

where $Z_0 = 0$.

The number of modules for calculating the scalar product depending on the required speed is determined by the following formula:

$$s = \frac{N}{2^v}, \tag{24}$$

where N is the number of neuro-like elements, and $v = 0, \ldots, d, d = log_2 N$. The system of NN cryptographic data encryption reaches its highest speed when the number of computational modules of the scalar product corresponds to the number of neural elements N. To ensure real-time data encryption, it is proposed to implement the scalar product calculation modules, mask node module (MN), and module of the shaper of NN architecture and bit slices of the input data in the form of specialized hardware.

The NN cryptographic data encryption component works as follows. Before encrypting the data, the MC configures the NN architecture (determines the number of neural elements N, the number of inputs k and their bit-size m). For the selected NN architecture matrix of weights W_j and tables of P_{Mi} macro-partial products are calculated by MC, and then they are written in the memory of MP. In addition, the masks selected from the keys' memory are stored in the MN node. The message X to be encrypted comes to input of MN in fixed-point format; here, it is masked. The masked message X^* from the output of MN comes to input of the module of the shaper of NN architecture and bit slices, where it is divided into N groups with m bit rate and bit slices are formed $x_{1i}, \ldots, x_{Ni}$. It should be noted that forming of bit slices $x_{1i}, \ldots, x_{Ni}$ begins with lower bits. The formed bit slices $x_{1i}, \ldots, x_{Ni}$ are the addresses for reading macro-partial products P_{Mi} from the MP memory. The read macro-partial product P_{Mi} is written to the Rg1 register. The adder (Add) performs a summation of macro-partial products P_{Mi} as per Equation (23). The number of cycles required to calculate the scalar product is determined by the bit size of input m. Control of the encryption process in the onboard system of NN cryptographic data encryption is performed by MC.

The structure of the component of NN cryptographic data decryption is shown in Figure 7, where DCSB is the decryption component setting block, and x_j^*-j-th masked initial data.

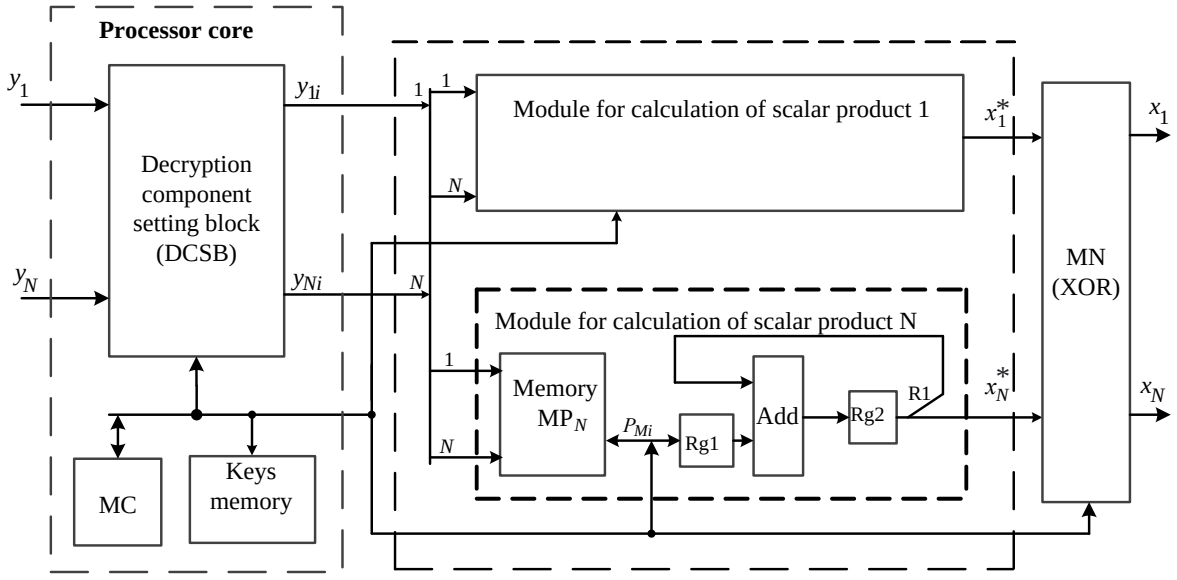

Figure 7. Structure of the component of NN cryptographic decryption of data.

The NN cryptographic data decryption component with symmetric keys works as follows. Before the start of data decryption, a key arrives, which, with the help of DCSB, configures the architecture (the number of N neuro-like elements) of the NN. For the selected NN architecture, the matrix of weighting coefficients W_j and the table of macro-partial products P_{Mi} are calculated using DCSB, and the mask digits are recorded in the MN node. Encrypted data $y_1, \ldots, y_N$ in floating-point format are sent to the input of the DCSB, in which the order alignment of the encrypted data and the formation of bit sections of the mantissas of the encrypted data $y_{1i}, \ldots, y_{Ni}$ are performed. Alignment of the orders of the encrypted data $y_1, \ldots, y_N$ is performed by determining the maximum order m_{maxy}, calculating the difference of orders for each y_j of the encrypted number $\Delta m_{y_j} = m_{maxy} - m_{y_j}$, and shifting the mantissa of each number to the right by the amount Δm_{y_j}. After the alignment of the orders, the formation of bit cuts of the mantissa of the encrypted numbers $y_{1i}, \ldots, y_{Ni}$ is performed, starting with the lowest digits.

The bit cuts $y_{1i}, \ldots, y_{Ni}$ obtained at the output of the DCSB are the address for reading from the MP memory of the macro-partial product P_{Mi}, which are used in the proposed table-algorithmic calculation of the scalar product. Calculated macro-partial P_{Mi} product is recorded in the register Rg1. With the help of the adder Add, the summation of macro-partial products P_{Mi} is performed according to Equation (23). The number of cycles required to calculate the scalar product is determined by the mantissas of encrypted numbers n_y. Management of the process of decryption of encrypted data is performed using MC. Decrypted masked initial data $x_1^*, \ldots, x_N^*$ are received at the inputs of the MN, at the output of which we receive the initial data $x_1, \ldots, x_N$.

The process of decrypting encrypted data takes much longer than the encryption process. The number of cycles required to calculate the scalar product during data decryption has increased by $q = \left\lceil \frac{n_y}{m} \right\rceil$ times, where $\lceil\ \rceil$—the sign of rounding up to a larger whole number, n_y is the digits number of the mantissa of the encrypted data, and m is the digits number of the input data $x_1, \ldots, x_N$. It is possible to reduce the time of calculating the scalar product by using an algorithm that provides for the submission of q bit slices to the address inputs of q tables of macro-partial products. The use of such an algorithm reduces the time of calculating the scalar product by q times.

6. Results and Discussion

For experimental verification of the proposed NN technology for cryptographic protection of data transmission system, the simulation was performed. Currently, the hardware description languages such as VHDL, VHDL-AMS, Verilog, and Verilog-AMS are widely used for creating behavioral descriptions and models of digital, analog, and mixed-signal devices and systems [44,45].

The design of specialized onboard hardware systems for NN cryptographic data encryption was performed in the VHDL hardware programming language in the Quartus II ver. 13.1 development environment using its libraries. The Quartus II development environment supports the entire process of designing specialized hardware, from user input to FPGA programming and debugging of both the chip itself and the tools as a whole.

A schematic diagram of the specialized hardware components of NN cryptographic data encryption is shown in Figure 8. The inputs of module XOR_Mask1_4_2: X [7..0]—are the input data; Clk—input sync for input data download; X_Mask [7..0]—8-bit mask. At the output of this block, N vectors with bit length m are formed. Synchronization is implemented on the leading edge of Clk pulses.

Block V_Cutter with $N = 4$ input vectors of bit length $m = 2$ consists of N registers of parallel-serial type and forms vertical bit slices. Input data: Data_1 [n-1..0], . . ., Data_N [n-1..0]—N input vectors with bit length n; Clk—pulses of synchronization of forming vertical bit slices; Reset—the signal of the initial reset in the "0" output of the registers R_Par_Ser; Load—the signal to allow data to be loaded into the R_Par_Ser registers. Outputs: V_Out1, . . ., V_OutN—vertical bit slice. The formation of vertical sections begins with the lower bit.

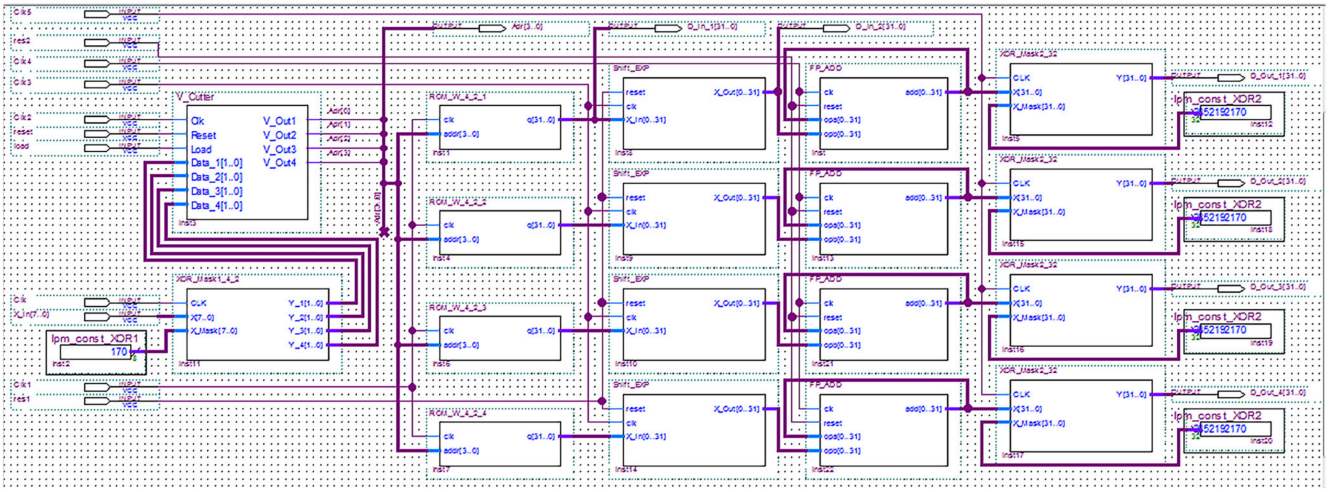

Figure 8. A circuit of the specialized hardware components of NN cryptographic data encryption.

The weights of the NN with $N = 4$ inputs with a bit length of $m = 2$ are stored in the FPGA ROM in the form of four tables. Each of them consists of 16 words with a bit length of 32 bits. Reading data from these tables is performed using blocks ROM_W_4_2_1, . . ., ROM_W_4_2_4.

Inputs of these blocks: addr [3..0]—the address of the cell of the table from which the data will be read; clk—synchronization pulses for reading data from the table. Synchronization is implemented on the leading edge of the pulses clk. Output: q [31..0]—data read from the cell with the input address.

The data read from the tables is transmitted to the input blocks Shift_EXP, which perform their multiplication by 2^j, where $j = 0, \ldots, n-1$. Upon receipt of this block of data corresponding to the zero digit, the bit counter is reset. Synchronization of this block is carried out by means of clock pulses Clk. At the output X_Out [0..31], we obtain the input data multiplied by 2^j.

From the output of the Shift_EXP blocks, the data are sent to one of the inputs of the adders FP_ADD. The other input of the adders is connected to their output. Adder input signals: clk—synchronization pulses; reset—signal to reset the input data opa when implementing the adder with the battery; opa [0..31], opb [0..31]—terms. On the leading edge of the first pulse clk, the adders are loaded into the adder, and on the leading edge of the second pulse, the received sum is displayed. Adder output: the sum add [0..31].

From the output of the adders, FP_ADD data is fed to the input of the block XOR_Mask2_32, which performs the overlay of the 32-bit mask. Inputs of the block XOR_Mask2_32: X [31..0]—encrypted output data; Clk—synchronization of input data download; X_Mask [31..0]—32-bit mask. Block output: vector Y [31..0]. Synchronization is implemented on the leading edge of Clk pulses. The encrypted data are obtained at the outputs D_Out_1, D_Out_2, D_Out_3, D_Out_4.

The timing diagram of the specialized hardware of NN cryptographic data encryption is presented in Figure 9.

The time diagram (Figure 9) shows an example of NN cryptographic encryption of eight-bit data, which are received in binary code at inputs X_In X [7..0]. An 8-bit mask 170 = 0xAA is received at the X_Mask [7..0] inputs, which is set using the lpm_const_XOR1 component (Figure 7). It is used to mask input data using the XOR operation. For input X_In_1—01001100 XOR 10101010 = 11100110; for input X_In_2—01010100 XOR 10101010 = 11111110. For the first number 01001100 at the outputs Y_1[1. . .0], Y_2[1. . .0], Y_3[1. . .0], Y_4[1. . .0] of the XOR_Mask1_4_2 block, we obtain 11, 10, 01, and 10, respectively. When encrypting the first vector of input data at the Adr outputs, we obtain 4-bit slices starting from the lowest bits, which are sent to the address inputs of ROM_W4_2_1,

ROM_W4_2_2, ROM_W4_2_3, and ROM_W4_2_4 blocks. These lookup tables contain pre-calculated neuro elements' weights.

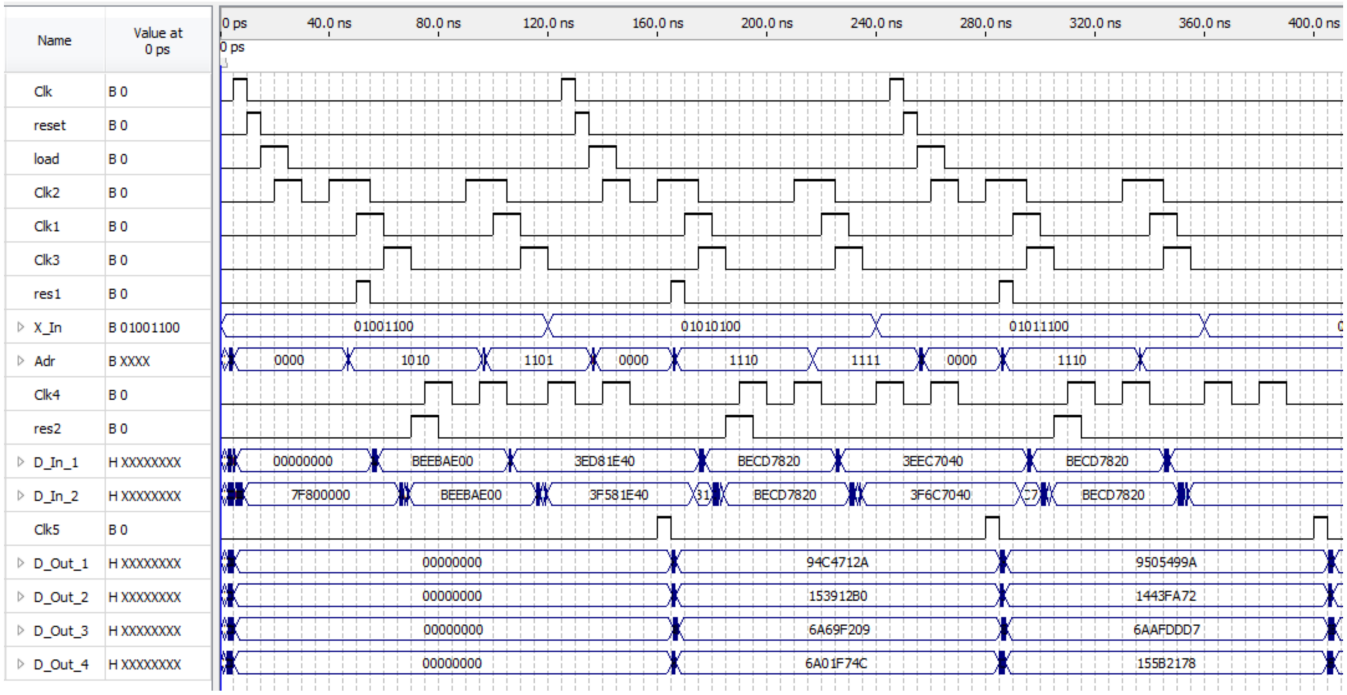

Figure 9. The timing chart of the specialized hardware of NN cryptographic data encryption.

For lower bits 1010 from ROM_W4_2_1 block, the 32-bit macro-partial product BEEBAE00 is read, which is fed to the input D_In_1 and to the input of the first block Shift_EXP, which performs the multiplication operation by shifting by 2^j, where $j = 0, \ldots, n-1$. At the output of the first block Shift_EXP and at the input D_In_2, we obtain BEEBAE00. For the next 1101 bits, the 32-bit macro-partial product 3ED81E40 is read from the ROM_W4_2_1 block. At the output of the first block Shift_EXP and at the input D_In_2, we obtain the macro-partial product multiplied by two, which is equal to 3F581E40.

In the first adder FP_ADD, we sum up the data from the outputs of the first block Shift_EXP and obtain the sum (its value is not displayed on the time charts), which is sent to the first block XOR_Mask2_32. In the first block, XOR_Mask2_32, the XOR operation is performed with the sum in IEEE 754 format and mask 2852192170 = 0xAA00FFAA. At the D_Out_1 output, we obtain the encrypted value 0x94C4712A.

For input data with a dimension of 1 byte X_In = {01001100}, we obtain an encrypted value with a dimension of 16 bytes D_Out_1 = 0x94C4712A; D_Out_2 = 0x153912B0; D_Out_3 = 0x6A69F209; D_Out_4 = 0x6A01F74C.

The implementation of the specialized hardware for NN cryptographic data encryption based on the FPGA EP3C16F484C6 Cyclone III family [46] requires 3053 logic elements and 745 registers. Approximately 160 nanoseconds are required to encrypt one input vector.

For comparison with the above-described hardware implementation on FPGA, the same components were implemented exclusively as the software. The components were created in the C language using the Code::Blocks development environment version 20.03. The execution time of a similar NN cryptographic data encryption procedure using a NanoPi Duo microcomputer based on the Allwinner Cortex-A7 H2+ SoC was about 20 ms. The results of the comparison allow us to see a significant gain in time for the implementation of NN cryptographic data encryption and decryption.

The authors understand the importance of the issue of cryptographic stability. However, this is beyond the scope of this study. The security of the neural network cryptographic approach mainly depends on the length of the key, which is determined by the masking codes, the neural network architecture, and the floating-point weighting matrix, as well

as on the frequency of its change. The length of the key depends on the number of neural elements N, which determine the size of the matrix of weighting coefficients.

The operation of onboard communication cryptographic systems for UAVs can be exposed to an attack on the secret key by breaking, through which it is possible to gain access to protected data. However, the time and resources required to crack the key and decrypt the encrypted data depend on the complexity of the algorithm for calculating the floating-point weighting matrix and the decryption algorithm. The number of operations required to calculate the matrix of weighting coefficients is approximately equal to N^2n arithmetic operations (where n is the data bit width), and the number of operations required to decrypt encrypted data approximately equals N^2 operations of multiplying floating-point numbers and N^2 operations of adding floating-point numbers. Therefore, the computational complexity of the proposed NN approach is high. Obviously, the evaluation of security analysis could be performed in further studies.

7. Conclusions

The approach to the implementation of neural networks for cryptographic protection of data transmission at UAV onboard communication systems has been presented in this work. This paper describes the UAV onboard system for NN cryptographic data protection in real-time using an integrated approach based on the following principles: variable equipment composition; modularity; conveyorization and spatial parallelism; software openness; and suitability for hardware implementation on FPGA.

The information technology of real-time neuro-like cryptographic data protection with symmetric keys (masking codes, neural network architecture, and matrix of weighting coefficients) oriented for onboard implementation has been developed. Due to the pre-calculation of matrices of weighting coefficients and tables of macro-partial products, use of tabular-algorithmic implementation of neuro-like elements, and dynamic change of keys, it provides increased cryptographic stability and hardware–software implementation on FPGA.

The table-algorithmic method of calculating the scalar product has been improved, by bringing the weighting coefficients to the greatest common order, pre-calculating the tables of macro-partial products and using instead of floating-point multiplication and summation the operations of reading from memory, fixed-point summation and shift, it provides a reduction hardware costs for its implementation and calculation time.

A real-time neural network cryptographic data protection system has been developed on the basis of a processor core supplemented with specialized hardware modules for calculating the scalar product, which, due to the combination of universal and specialized approaches, software and hardware, ensures the effective implementation of neuro-like algorithms for real-time cryptographic encryption and decryption of data.

The specialized hardware for NN cryptographic data encryption was developed in the VHDL equipment programming language in the Quartus II environment and implemented using family Cyclone III FPGA EP3C16F484C6.

Author Contributions: Conceptualization, I.T. and V.T.; methodology, I.T., Y.O. and Y.L.; software, Y.L. and Y.O.; validation, Y.L., Y.O. and I.K.; formal analysis, I.T. and V.T.; investigation, A.Ł. and A.H.; resources, A.Ł. and I.K.; data curation, Y.L. and I.K.; writing—original draft preparation, I.T., I.K. and Y.O.; writing—review and editing, I.K., A.H. and A.Ł.; visualization, I.K. and Y.O.; supervision, V.T. and A.Ł.; project administration, I.T. and V.T. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: No new data were created or analyzed in this study. Data sharing is not applicable to this article.

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

References

1. Han, B.; Qin, D.; Zheng, P.; Ma, L.; Teklu, M.B. Modeling and performance optimization of unmanned aerial vehicle channels in urban emergency management. *ISPRS Int. J. Geo-Inf.* **2021**, *10*, 478. [CrossRef]
2. Śledź, S.; Ewertowski, M.W.; Piekarczyk, J. Applications of unmanned aerial vehicle (UAV) surveys and Structure from Motion photogrammetry in glacial and periglacial geomorphology. *Geomorphology* **2021**, *378*, 107620. [CrossRef]
3. Zhang, C.; Zou, W.; Ma, L.; Wang, Z. Biologically inspired jumping robots: A comprehensive review. *Robot. Auton. Syst.* **2020**, *124*, 103362. [CrossRef]
4. Li, D.; Ma, G.; He, W.; Ge, S.S.; Lee, T.H. Cooperative Circumnavigation Control of Networked Microsatellites. *IEEE Trans. Cybern.* **2020**, *50*, 4550–4555. [CrossRef]
5. Boreiko, O.; Teslyuk, V.; Zelinskyy, A.; Berezsky, O. Development of models and means of the server part of the system for passenger traffic registration of public transport in the "smart" city. *East.-Eur. J. Enterp. Technol.* **2017**, *1*, 40–47. [CrossRef]
6. Kim, K.; Kang, Y. Drone security module for UAV data encryption. In Proceedings of the 2020 International Conference on Information and Communication Technology Convergence (ICTC), Jeju, Republic of Korea, 21–23 October 2020; pp. 1672–1674. [CrossRef]
7. Samanth, S.; Prema, K.V.; Balachandra, M. Security in Internet of Drones: A Comprehensive Review. *Cogent Eng.* **2022**, *9*, 2029080. [CrossRef]
8. Kong, P.-Y. A survey of cyberattack countermeasures for unmanned aerial vehicles. *IEEE Access* **2021**, *9*, 148244–148263. [CrossRef]
9. Shafique, A.; Mehmood, A.; Elhadef, M.; Khan, K.H. A lightweight noise-tolerant encryption scheme for secure communication: An unmanned aerial vehicle application. *PLoS ONE* **2022**, *17*, e0273661. [CrossRef] [PubMed]
10. Verma, A.; Ranga, V. Security of RPL based 6LoWPAN Networks in the Internet of Things: A Review. *IEEE Sens. J.* **2020**, *20*, 5666–5690. [CrossRef]
11. Srivastava, S.; Bhatia, A. On the Learning Capabilities of Recurrent Neural Networks: A Cryptographic Perspective. In Proceedings of the 2018 IEEE International Conference on Big Knowledge (ICBK), Singapore, 17–18 November 2018; pp. 162–167. [CrossRef]
12. Zhu, Y.; Vargas, D.V.; Sakurai, K. Neural Cryptography Based on the Topology Evolving Neural Networks. In Proceedings of the 2018 Sixth International Symposium on Computing and Networking Workshops (CANDARW), Takayama, Japan, 27–30 November 2018; pp. 472–478. [CrossRef]
13. Duan, X.; Han, Y.; Wang, C.; Ni, H. Optimization of Encrypted Communication Length Based on Generative Adversarial Network. In Proceedings of the 2021 IEEE 4th International Conference on Big Data and Artificial Intelligence (BDAI), Qingdao, China, 2–4 July 2021; pp. 165–170. [CrossRef]
14. Grodzki, W.; Łukaszewicz, A. Design and manufacture of unmanned aerial vehicles (UAV) wing structure using composite materials. *Mater. Werkst.* **2015**, *46*, 269–278. [CrossRef]
15. Łukaszewicz, A.; Szafran, K.; Jóźwik, J. CAx techniques used in UAV design process. In Proceedings of the 2020 IEEE 7th International Workshop on Metrology for AeroSpace (MetroAeroSpace), Pisa, Italy, 22–24 June 2020; pp. 95–98. [CrossRef]
16. Łukaszewicz, A.; Skorulski, G.; Szczebiot, R. The main aspects of training in the field of computer aided techniques (CAx) in mechanical engineering. In Proceedings of the 17th International Scientific Conference on Engineering for Rural Development, Jelgava, Latvia, 23–25 May 2018; pp. 865–870. [CrossRef]
17. Łukaszewicz, A.; Miatluk, K. Reverse Engineering Approach for Object with Free-Form Surfaces Using Standard Surface-Solid Parametric CAD System. *Solid State Phenom.* **2009**, *147–149*, 706–711. [CrossRef]
18. Miatliuk, K.; Łukaszewicz, A.; Siemieniako, F. Coordination method in design of forming operations of hierarchical solid objects. In Proceedings of the 2008 International Conference on Control, Automation and Systems, ICCAS 2008, Seoul, Republic of Korea, 14–17 October 2008; pp. 2724–2727. [CrossRef]
19. Puchalski, R.; Giernacki, W. UAV Fault Detection Methods, State-of-the-Art. *Drones* **2022**, *6*, 330. [CrossRef]
20. Zietkiewicz, J.; Kozierski, P.; Giernacki, W. Particle swarm optimisation in nonlinear model predictive control; comprehensive simulation study for two selected problems. *Int. J. Control* **2021**, *94*, 2623–2639. [CrossRef]
21. Kownacki, C.; Ambroziak, L. Adaptation Mechanism of Asymmetrical Potential Field Improving Precision of Position Tracking in the Case of Nonholonomic UAVs. *Robotica* **2019**, *37*, 1823–1834. [CrossRef]
22. Kownacki, C.; Ambroziak, L.; Ciężkowski, M.; Wolniakowski, A.; Romaniuk, S.; Bożko, A.; Ołdziej, D. Precision Landing Tests of Tethered Multicopter and VTOL UAV on Moving Landing Pad on a Lake. *Sensors* **2023**, *23*, 2016. [CrossRef]
23. Basri, E.I.; Sultan, M.T.H.; Basri, A.A.; Mustapha, F.; Ahmad, K.A. Consideration of Lamination Structural Analysis in a Multi-Layered Composite and Failure Analysis on Wing Design Application. *Materials* **2021**, *14*, 3705. [CrossRef]
24. Al-Haddad, L.A.; Jaber, A.A. An Intelligent Fault Diagnosis Approach for Multirotor UAVs Based on Deep Neural Network of Multi-Resolution Transform Features. *Drones* **2023**, *7*, 82. [CrossRef]
25. Yang, J.; Gu, H.; Hu, C.; Zhang, X.; Gui, G.; Gacanin, H. Deep Complex-Valued Convolutional Neural Network for Drone Recognition Based on RF Fingerprinting. *Drones* **2022**, *6*, 374. [CrossRef]
26. Duan, X.; Han, Y.; Wang, C.; Ni, H. Optimization of Encrypted Communication Model Based on Generative Adversarial Network. In Proceedings of the 2022 International Conference on Blockchain Technology and Information Security (ICBCTIS), Huaihua City, China, 15–17 July 2022; pp. 20–24. [CrossRef]

27. Karakaya, B.; Celik, V.; Gulten, A. Realization of Delayed Cellular Neural Network model ON FPGA. In Proceedings of the 2018 Electric Electronics, Computer Science, Biomedical Engineerings' Meeting (EBBT), Istanbul, Turkey, 18–19 April 2018; pp. 1–4. [CrossRef]
28. Volna, E.; Kotyrba, M.; Kocian, V.; Janosek, M. Cryptography Based on Neural Network. In *Proceedings of the 26th European Conference on Modeling and Simulation (ECMS 2012), Koblenz, Germany, 29 May–1 June 2012*; Troitzsch, K.G., Moehring, M., Lotzmann, U., Eds.; European Council for Modeling and Simulation: Caserta, Italy, 2012; pp. 386–391. [CrossRef]
29. Shihab, K. A backpropagation neural network for computer network security. *J. Comput. Sci.* **2006**, *2*, 710–715. [CrossRef]
30. Sagar, V.; Kumar, K. A symmetric key cryptographic algorithm using counter propagation network (CPN). In Proceedings of the 2014 ACM International Conference on Information and Communication Technology for Competitive Strategies, (ICTCS'14), Udaipur, India, 14–16 November 2014. [CrossRef]
31. Arvandi, M.; Wu, S.; Sadeghian, A.; Melek, W.W.; Woungang, I. Symmetric cipher design using recurrent neural networks. In Proceedings of the IEEE International Joint Conference on Neural Networks, Vancouver, BC, Canada, 16–21 July 2006; pp. 2039–2046. [CrossRef]
32. Tsmots, I.; Tsymbal, Y.; Khavalko, V.; Skorokhoda, O.; Teslyuk, T. Neural-like means for data streams encryption and decryption in real time. In Proceedings of the 2018 IEEE Second International Conference on Data Stream Mining & Processing (DSMP), Lviv, Ukraine, 21–25 August 2018; pp. 438–443. [CrossRef]
33. Scholz, M.; Fraunholz, M.; Selbig, J. Nonlinear principal component analysis: Neural network models and applications. In *Principal Manifolds for Data Visualization and Dimension Reduction*; Gorban, A.N., Kégl, B., Wunsch, D.C., Zinovyev, A.Y., Eds.; Lecture Notes in Computational Science and Engineering; Springer: Berlin/Heidelberg, Germany, 2008; Volume 58. [CrossRef]
34. Rabyk, V.; Tsmots, I.; Lyubun, Z.; Skorokhoda, O. Method and Means of Symmetric Real-time Neural Network Data Encryption. In Proceedings of the 2020 IEEE 15th International Scientific and Technical Conference on Computer Sciences and Information Technologies (CSIT 2020), Zbarazh, Ukraine, 23–26 September 2020; Volume 1, pp. 47–50. [CrossRef]
35. Chang, A.X.M.; Martini, B.; Culurciello, E. Recurrent Neural Networks Hardware Implementation on FPGA. *arXiv* **2015**, arXiv:1511.05552. [CrossRef]
36. Nurvitadhi, E.; Venkatesh, G.; Sim, J.; Marr, D.; Huang, R.; Ong Gee Hock, J.; Tat Liew, Y.; Srivatsan, K.; Moss, D.; Subhaschandra, S.; et al. Can FPGAs beat GPUs in accelerating next-generation deep neural networks? In Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, Monterey, CA, USA, 22–24 February 2017; pp. 5–14. [CrossRef]
37. Misra, J.; Saha, I. Artificial neural networks in hardware: A survey of two decades of progress. *Neurocomputing* **2010**, *74*, 239–255. [CrossRef]
38. Guo, K.; Sui, L.; Qiu, J.; Yao, S.; Han, S.; Wang, Y.; Yang, H. From model to FPGA: Software-hardware co-design for efficient neural network acceleration. In Proceedings of the 2016 IEEE Hot Chips 28 Symposium (HCS), Cupertino, CA, USA, 21–23 August 2016; pp. 1–27. [CrossRef]
39. Ovtcharov, K.; Ruwase, O.; Kim, J.Y.; Fowers, J.; Strauss, K.; Chung, E.S. Accelerating Deep Convolutional Neural Networks Using Specialized Hardware. Microsoft Research Whitepaper. 2016. Available online: https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/CNN20Whitepaper.pdf (accessed on 29 April 2022).
40. Wang, Y.; Xu, J.; Han, Y.; Li, H.; Li, X. DeepBurning: Automatic generation of FPGA-based learning accelerators for the neural network family. In Proceedings of the 53rd Annual Design Automation Conference (DAC'16), Austin, TX, USA, 5–9 June 2016; Association for Computing Machinery: New York, NY, USA, 2016; pp. 1–6. [CrossRef]
41. Nurvitadhi, E.; Sheffield, D.; Sim, J.; Mishra, A.; Venkatesh, G.; Marr, D. Accelerating Binarized Neural Networks: Comparison of FPGA, CPU, GPU, and ASIC. In Proceedings of the 2016 International Conference on Field-Programmable Technology (FPT), Xi'an, China, 7–9 December 2016; pp. 77–84. [CrossRef]
42. Yayik, A.; Kutlu, Y. Neural Network Based Cryptography. *Neural Netw. World* **2014**, *24*, 177–192. [CrossRef]
43. Govindu, G.; Zhuo, L.; Choi, S.; Prasanna, V. Analysis of high-performance floating-point arithmetic on FPGAs. In Proceedings of the 18th International Parallel and Distributed Processing Symposium (IPDPS 2004), Santa Fe, NM, USA, 26–30 April 2004; p. 149. [CrossRef]
44. Khalil, K.; Dey, B.; Abdelrehim, M.; Kumar, A.; Bayoumi, M. An Efficient Reconfigurable Neural Network on Chip. In Proceedings of the 2021 28th IEEE International Conference on Electronics, Circuits, and Systems (ICECS), Dubai, United Arab Emirates, 28 November–1 December 2021; pp. 1–4. [CrossRef]
45. Dumesnil, E.; Beaulieu, P.-O.; Boukadoum, M. Fully parallel FPGA Implementation of an Artificial Neural Network Tuned by Genetic Algorithm. In Proceedings of the 2018 16th IEEE International New Circuits and Systems Conference (NEWCAS), Montreal, QC, Canada, 24–27 June 2018; pp. 365–369. [CrossRef]
46. *Cyclone III Device Handbook*; Altera Corporation: San Jose, CA, USA, 2012. Available online: https://www.intel.com/content/www/us/en/content-details/655197/cyclone-iii-device-handbook-volume-2-chapter-1-cyclone-iii-device-datasheet.html (accessed on 29 April 2022).