Hardware Implementation of Sigmoid Activation Functions using FPGA

Ivan Tsmots
Automated Control Systems dept.
Lviv Polytechnic National University
Lviv, Ukraine
https://orcid.org/0000-0002-4033-8618

Vasyl Rabyk
Department of Radio Physics and Computer Technologies
Ivan Franko National University
Lviv, Ukraine
rabykv@ukr.net

Oleksa Skorokhoda
Automated Control Systems dept.
Lviv Polytechnic National University
Lviv, Ukraine
https://orcid.org/0000-0002-1455-8553

***Abstract* — The methods of approximation of the sigmoid function are developed and modified, using piecewise linear approximation and approximation by a second order polynomial. The estimation of the accuracy of approximation by these methods of sigmoid function and its derivative is performed. The considered methods are implemented on FPGA using VHDL. The comparison of the required hardware resources and performance is given.**

Keywords — FPGA, VHDL, activation function, sigmoid function, piecewise linear approximation, hardware costs.

I. Introduction

In recent decades, interest in the hardware implementation of artificial neural networks (SNN) has grown. This is evidenced by the number of publications on this topic. This is primarily due to the rapid development of the element base, which is used for the implementation of digital ANN (very large scale integrated circuits – VLSI). One of the pressing problems that remain with this is the increase in the speed of the neural networks. One of the ways to speed up their work by exploiting of parallelism is to implement them on FPGA.

The speed of the work of artificial neurons depends on the speed of sigmoid activation function. Its implementation on FPGA requires significant hardware resources [1, 2, 3]. In [1], an overview of the basic methods used for implementation of the sigmoid function is presented.

Different methods of approximation are used for the digital implementation of nonlinear activation functions: tabular, Taylor transformation, piecewise linear approximation [1, 2]. When decomposing a Taylor series, many multiplications need to be performed, so this method is not acceptable for implementation on FPGA. Table method involves creating a table of possible values of the target function, taking into account their bit width. But the creation of a separate local table for each neuron will require significant hardware resources of the FPGA. Using the same table for different neurons will result in large time delays, since the spread of signals in the neurons of one layer is performed in parallel. Therefore, for implementation of the sigmoid type of activation functions the piecewise linear approximation (PWL approximation) is the most frequently used. PWL approximation of the sigmoid function consists of replacing the nonlinear function on each of its selected intervals by a straight line. This type of approximation is used in many papers [2, 3].

The choice of the method of approximation of the sigmoid function and its hardware implementation is the main aspects on which the accuracy and speed of the algorithm depend. With low accuracy of approximation we get high performance, and reducing the error of approximation leads to an increase in hardware resources and a decrease in the speed of data processing.

Another important point to consider is the differentiation of the approximated activation function [2, 3] since the ANN learning methods include both the activation function and its derivative.

II. Sigmoid Activation Functions of Neurons

Sigmoid activation functions most often are used for feedforward neural networks. They are monotonically growing, continuous and differentiated. In [1] for describing and analysis of the sigmoid activation functions a general class of functions is used:

$$f(x,k,b,T,c) = k + \frac{c}{1+be^{Tx}}\ ,\ \forall x \in R\ , \qquad (1)$$

where $k \in R$, $b \in R^+$, $T, c \in R\backslash\{0\}$, R – the set of real numbers $(-\infty, +\infty)$, R^+ – the set of real positive numbers $(0, +\infty)$, R\{0} – the set of real numbers except zero point $(-\infty, 0)$ and $(0, +\infty)$. Sigmoid nonlinearities usually belong to one of the following classes:

- classical sigmoid function ($k = 0$, $c = b = 1$, $T = -1$):

$$f(x) = \frac{1}{1+e^{-x}} \qquad (2)$$

- thermodynamic-like function ($k = 0$, $c = b = 1$, $T = -1/T'$):

$$t(x, T') = \frac{1}{1+e^{-x/T'}} \qquad (3)$$

- hyperbolic tangent ($k = 1$, $c = -2$, $b = 1$, $T = 2$):

$$h(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \qquad (4)$$

It should be noted that it's enough to calculate the sigmoid activation function (2) only for positive arguments x. For negative values of x it can be calculated using the following expression:

$$f(-x) = 1 - f(x) \qquad (5)$$

Indeed, by performing simple transformations for (2) and (4), we obtain:

978-1-7281-0053-1/19/$31.00 ©2019 IEEE

$$f(-x) = \frac{1}{1+e^{x}} = \frac{1}{1+\frac{1}{e^{-x}}} = \frac{e^{-x}}{1+e^{-x}} = \frac{1+e^{-x}-1}{1+e^{-x}} = 1 - f(x)$$

The nonlinear hyperbolic tangent function (4) can be calculated using the classical sigmoid function (2):

$$\mathrm{h}(x) = 1 + \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}} - 1 = \frac{2e^{x}}{e^{x}+e^{-x}} - 1 = \frac{2}{1+e^{-2x}} - 1 = 2f(2x) - 1, \quad (6)$$

$$\mathrm{h}(-x) = 2f(-2x) - 1 = 2[1 - f(2x)] - 1 = 1 - 2f(2x) \quad (7)$$

For estimation of the accuracy of approximation the maximum and average errors are used [2]. The average absolute error ε_{ave} and the maximum absolute error ε_{max} of the function $f(x)$, which is approximated by the function $\bar{f}(x)$ in the interval (x_{min}, x_{max}) are calculated as follow:

$$\varepsilon_{ave} = \frac{\sum_{i=0}^{N_p-1} |\bar{f}(x_i) - f(x_i)|}{N_p}, \quad (8)$$

$$\varepsilon_{max} = max\left|\bar{f}(x_i) - f(x_i)\right|, i = 0, \dots, N_p, \quad (9)$$

where N_p – number of points on which the interval (x_{min}, x_{max}) is divided.

The efficiency of approximation is compared by the achieved accuracy, speed, and hardware resources.

Arithmetic operations in neurons are performed over real numbers. When implementing arithmetic operations on FPGA over real numbers with fixed and floating points, the time for their calculation and hardware resources increase. Therefore, when implementing the components of artificial neurons using VHDL, real numbers are converted to integers using multiplication by 2^{10} and cut off the fractional part. The format of the representation of real numbers has a dimension of 16 bits: 1 sign bit and 15 bits to store the resulting integer. The maximum integer fraction of the real number should not exceed $2^4 - 1 = 15$. It should be noted that the input data in the neural networks are normalized. For example, the representation of the real number 3.1625 in the format of integers is shown in Fig. 1. To do this, the number is multiplied by 2^{10} = 1024 and the fractional part is cut off: 3.1625 * 1024 = 3238.4≈3238. In hexadecimal, this number is equal to: 3238_{10} = 0x0CA6.

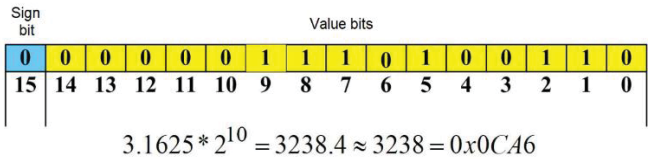


Fig. 1. Representation of the real number 3.1625 in the format of a signed integer

III. Methods of Approximation of Sigmoid Functions

Let's consider several PWL approximation algorithms that differ in the number of points, the placement of the initial and final points on the approximation lines and the criteria for their selection.

A. Piecewise linear approximation of the sigmoid function (PLAN approximation)

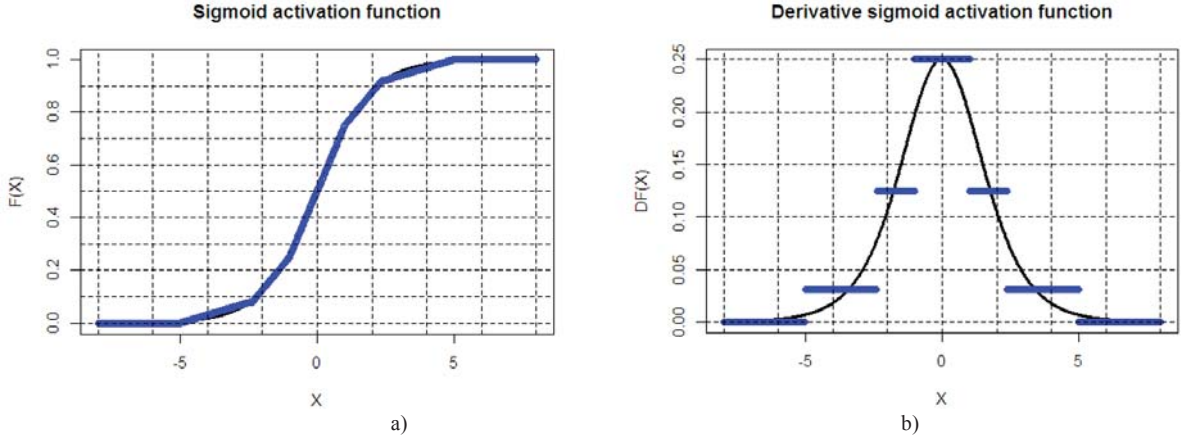
In this method, the approximation is performed using the following expression [2, 3]:

$$f(x) = \begin{cases} 1, & |x| \geq 5.0 \\ 0.03125 * |x| + 0.84375, & 2.375 \leq |x| < 5.0 \\ 0.125 * |x| + 0.625, & 1.0 \leq |x| < 2.375 \\ 0.25 * |x| + 0.5, & 0 \leq x < 1.0 \end{cases}, \quad (10)$$

Calculations must be performed only for positive input data x. For negative input data x, the sigmoid function is calculated using (5). One can convert (10) into an integer by multiplying its terms without the variable $|x|$ by 2^{10}:

$$f(x) = \begin{cases} 1024, & |x| \geq 5120 \\ 2^{-5} * |x| + 864, & 2432 \leq |x| < 5120 \\ 2^{-3} * |x| + 640, & 1024 \leq |x| < 2432 \\ 2^{-2} * |x| + 512, & 0 \leq |x| < 1024 \end{cases} \quad (11)$$

For the implementation of the last expression on the FPGA no multipliers are needed, it is enough to use shifting registers and adders. Sigmoid function and its PLAN approximation are depicted in Fig. 2a, and their derivatives – in Fig. 2b.



a) b)

Fig. 2. a) Sigmoid function and its PLAN approximation; b) Derivatives of the sigmoid function and its PLAN approximation

Let's divide the range (-8, 8) into $N_p = 1000$ points. In this range, the average and maximum errors of the PLAN approximation of the sigmoid function are $\varepsilon_{ave} = 0.00587$, $\varepsilon_{max} = 0.00587$ and for their derivatives $d\varepsilon_{ave} = 0.01412$, $d\varepsilon_{max} = 0.07088$.

The chart of absolute error between the sigmoid function and its PLAN approximation (the curve of black color), between the derivatives of the sigmoid function and its PLAN approximation (blue curve) is shown in Fig. 3.

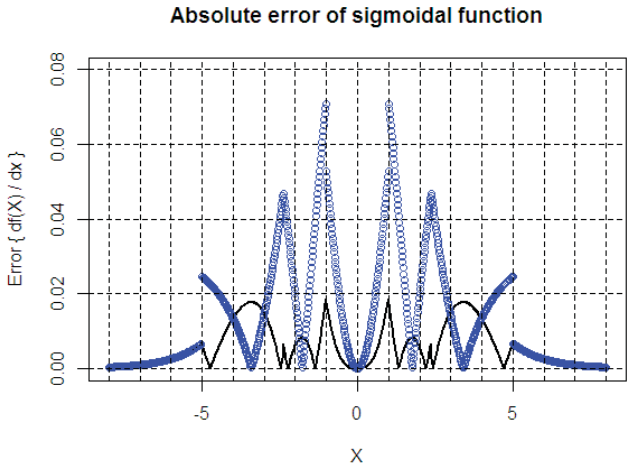


Fig. 3. The chart of absolute error between the sigmoid function and its PLAN approximation, and their derivatives

B. Approximation of the sigmoid function by a second-order curve.

For incoming data from the range $(0, x_{max})$, the sigmoid function is approximated by a polynomial [2, 3]:

$$\bar{f}(x) = c + bx + ax^2 \qquad (12)$$

In (12) unknown coefficients *a*, *b* and *c* are determined by the method of least squares. The system of equations for their calculation is as follows:

$$\begin{bmatrix} \sum_{i=0}^{N_p-1} x_i^2 & \sum_{i=0}^{N_p-1} x_i^3 & \sum_{i=0}^{N_p-1} x_i^4 \\ \sum_{i=0}^{N_p-1} x_i & \sum_{i=0}^{N_p-1} x_i^2 & \sum_{i=0}^{N_p-1} x_i^3 \\ N_p & \sum_{i=0}^{N_p-1} x_i & \sum_{i=0}^{N_p-1} x_i^2 \end{bmatrix} * \begin{bmatrix} c \\ b \\ a \end{bmatrix} = \begin{bmatrix} \sum_{i=0}^{N_p-1} f(x_i)x_i^2 \\ \sum_{i=0}^{N_p-1} f(x_i)x_i \\ \sum_{i=0}^{N_p-1} f(x_i) \end{bmatrix} \qquad (13)$$

Let's divide the range (0, 4) into $N_p = 1000$ points. Using the values x_i and $f(x_i)$ at these points, we form a system of equations (13). Having solved it, we obtain the coefficients *a*, *b*, and *c*. The expression for the approximation of sigmoid function:

$$f(x) = \begin{cases} 1, & x \geq 4.0 \\ -0.03577 * x^2 + 0.25908 * x + 0.5038, & 0 \leq x < 4.0' \end{cases} \qquad (14)$$

Integer form of (14) is as follows:

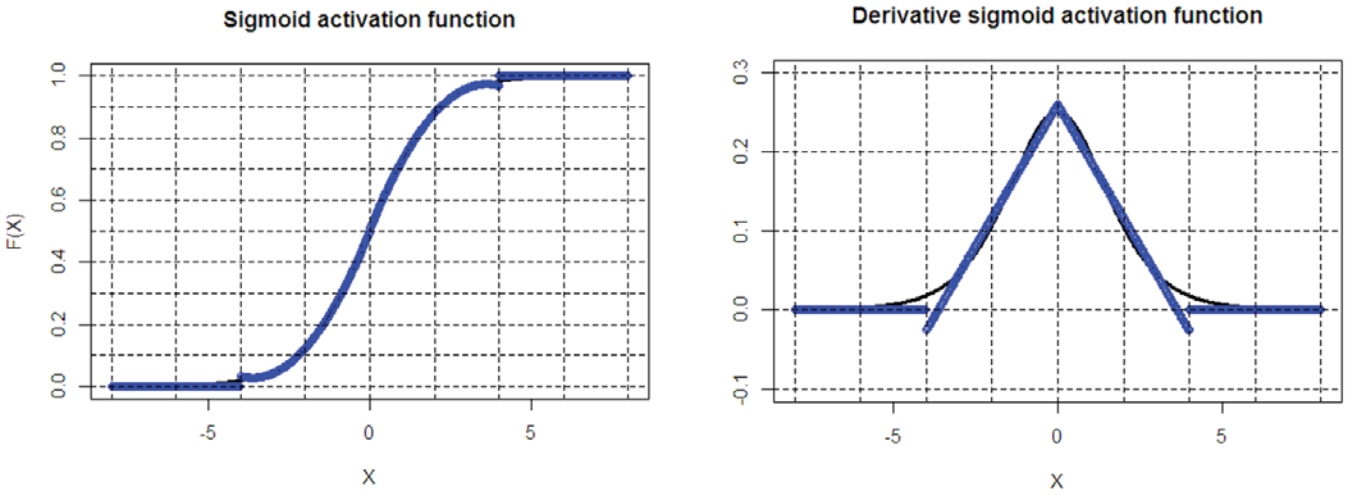
$$f(x) = \begin{cases} 1024, & x \geq 4096 \\ -36 * 2^{-20} * x^2 + 265 * 2^{-10} * x + 515, & 0 \leq x < 4096 \end{cases} \qquad (15)$$

In the range (-8, 8) the average and maximum errors of approximation of the sigmoid function by a second-order curve are $\varepsilon_{ave} = 0.00426$, $\varepsilon_{max} = 0.01798$, and for their derivatives – $d\varepsilon_{ave} = 0.00769$, $d\varepsilon_{max} = 0.04388$.

Sigmoid function and its approximation by a second-order curve are shown in Fig. 4a, and their derivatives – in Fig. 4b. The black color depicts the sigmoid function and its derivative, and the blue color – their approximation.

The chart of absolute error between the sigmoid function and its approximation by a second-order polynomial (the curve of black color), between the derivatives of the sigmoid function and its approximation by a second-order polynomial (blue curve) is shown in Fig. 5.

To implement the approximation of the sigmoid function (15) on the FPGA it is necessary to use multipliers, which will increase the hardware resources and reduce the speed.



a) b)

Fig. 4. a) Sigmoid function and its approximation by a second-order curve;
b) Derivative of the sigmoid function and its approximation by a second-order curve

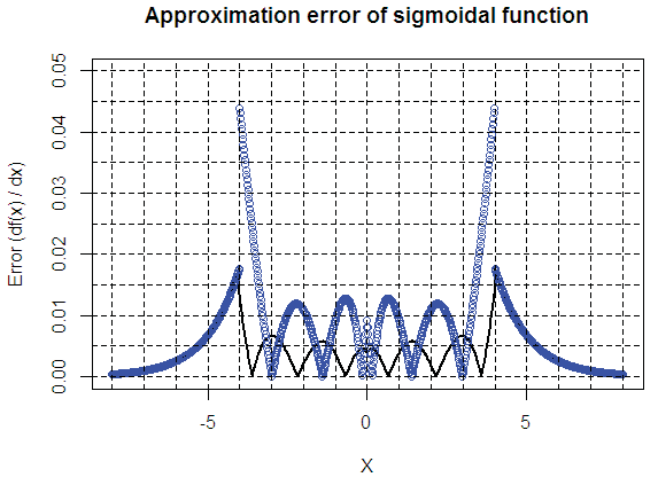


Fig. 5. The chart of absolute error between the sigmoid function and its approximation by a second-order polynomial

In [2, 3] for the approximation of the sigmoid function by the second-order polynomial a simplified expression is used, the implementation of which requires only one multiplier, shift registers, and adders:

$$f(x) = \begin{cases} 1, & x \geq 4.0 \\ -0.03125 * x^2 + 0.25 * x + 0.5, & 0 \leq x < 4.0' \end{cases} \quad (16)$$

Integer form of (16) is as follows:

$$f(x) = \begin{cases} 1024, & x \geq 4096 \\ -2^{-15} * x^2 + 2^{-2} * x + 512, & 0 \leq x < 4096' \end{cases} \quad (17)$$

Sigmoid function and its approximation by (17) in the range (-8, 8) are shown in Fig. 6a, and their derivatives – in Fig. 6b. The black color depicts the sigmoid function and its derivative, and the blue color – their approximation.

In the range (-8, 8) the average and maximum error of approximation of the sigmoid function by (17) are $\varepsilon_{ave} = 0.00774$, $\varepsilon_{max} = 0.02160$, and for their derivatives – $d\varepsilon_{ave} = 0.00877$, $d\varepsilon_{max} = 0.02375$. The chart of absolute errors is depicted in Fig. 7 (black – an error between the sigmoid function and the approximation expression (17), blue color – an error between their derivatives).

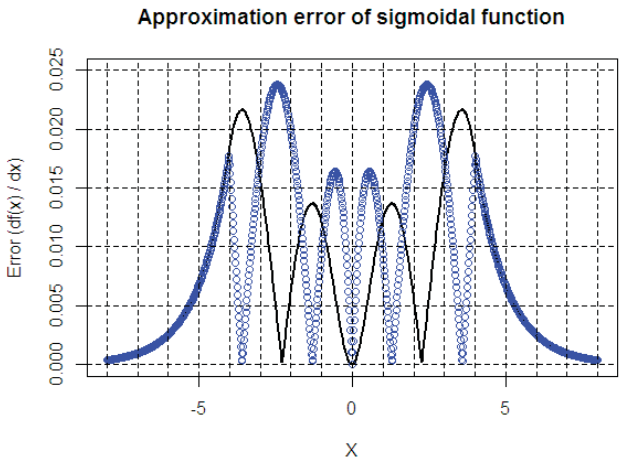
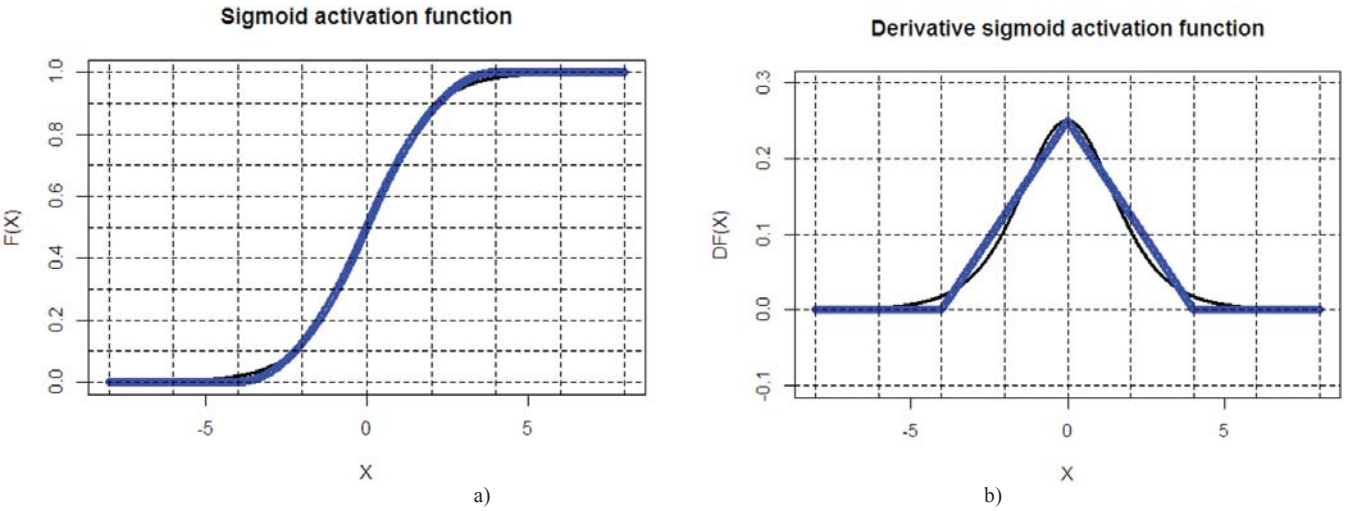


Fig. 7. The chart of absolute error between the sigmoid function and the approximation expression (17), and between their derivatives

IV. Implementation of Sigmoid Functions on FPGA

The implementation of sigmoid functions was carried out in the development environment Quartus II for FPGA EP3C16F484C6 of the Cyclone III family using hardware definition language VHDL.

For this purpose, sigmoid function approximation methods discussed in section 2 ((11), (15), and (17)) have been used. Each of these expressions is implemented as a symbol. Fig. 8 shows the symbol for the PLAN approximation of the sigmoid function (11). The other two symbols (FA_Sigm_N2, FA_Sigm_N3) have the same inputs and outputs. The inputs of these symbols are Clk – synchronization input, IN [15..0] – the sum of the weighted inputs of the neuron (16 bit width), and the output – Out [15..0] – the value of the approximation of the activation function (16 bit width as well). The sigmoid function is calculated along the front of the pulses entering the Clk input.



a) b)

Fig. 6. a) Sigmoid function and its approximation by (17); b) Derivative of the sigmoid function and its approximation by (17)

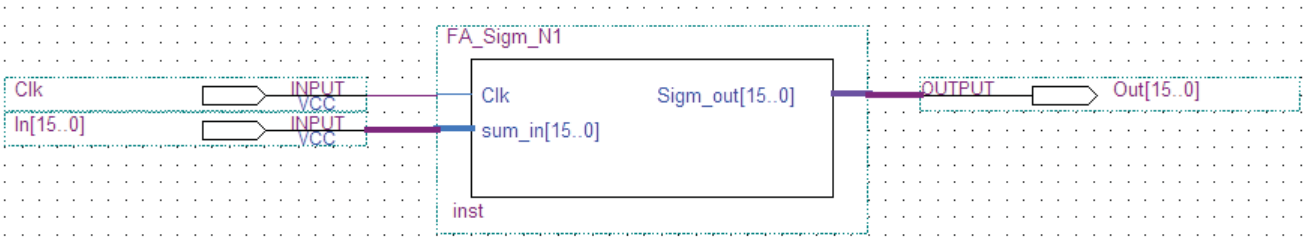


Fig. 8. Appearance of the symbol FA_Sigm_N1

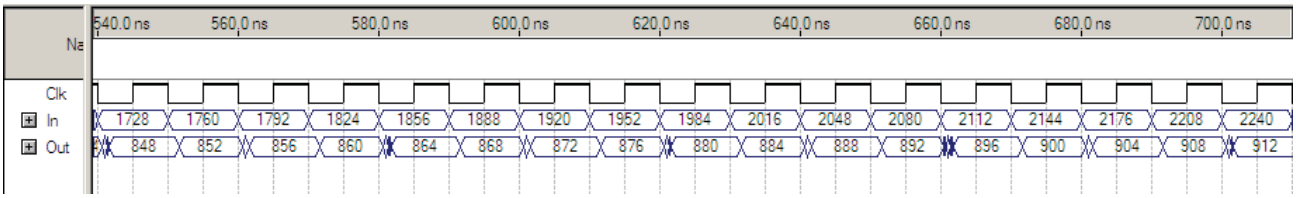


Fig. 9. Time charts of the module of the sigmoid function calculation FA_Sigm_N1

Fig. 9 shows a fragment of the time chart of the module's work, which calculates the sigmoid function using the PLAN approximation. The time chart is obtained using timing modulation in the development environment Quartus II.

FA_Sigm_N1 module inputs provide a sequence of synchronization pulses with a period of 10 ns and a sequence of weighted sums from 0 to 8192 in step 32. At the output of the adder we obtain the value of the approximated sigmoid function. Fig. 9 shows a fragment with input values ranging from 1728 to 2240. For example, the value of the input signal In = 1728 corresponds to the value of the signal $Out = 2^{-3} * 1728 + 640 = 856$. Output signal delay relative to synchronization pulses (t_{CO}) for modules FA_Sigm_N1 and FA_Sigm_N3 is 27…28 ns, and for the module FA_Sigm_N3 – 37 ns.

A comparison is made on the accuracy of the approximation of the sigmoid function by (11), (15) and (17) and the hardware resources of the FPGA spent on their implementation. Hardware resources are compared by the number of logical elements (LEs) and pins of FPGA EP3C16F484C6 from the Altera Cyclone III family. The maximum number of LEs for this FPGA [4] is 15408, and the number of pins is 347. The comparison results are given in Table 1.

In addition to the simulation of sigmoid functions they have been implemented at the DE0 board [5]. The value of the weighted sum applied to the input of modules FA_Sigm_N1, FA_Sigm_N2, FA_Sigm_N3 is set using switches, and the resulting value of the activation function is shown on the seven-segment LED indicators.

TABLE 1. COMPARISON OF IMPLEMENTATIONS OF SIGMOID FUCNTION APPROXIMATIONS

Symbol	ε_{ave}	ε_{max}	$d\varepsilon_{ave}$
FA_Sigm_N1	0.00587	0.01850	0.01412
FA_Sigm_N2	0.00426	0.01798	0.00769
FA_Sigm_N3	0.00774	0.02160	0.00877
Symbol	$d\varepsilon_{max}$	**LEs**	**Pins**
FA_Sigm_N1	0.07088	246/15408	33/347
FA_Sigm_N2	0.04388	368/15408	33/347
FA_Sigm_N3	0.02375	167/15408	33/347

V. CONCLUSION

For the approximation of the sigmoid activation function, the methods of piecewise linear approximation (module FA_Sigm_N1) and second-order polynomial approximation (modules FA_Sigm_N2, FA_Sigm_N3) are used. In the first method shifting registers are used for multiplication. The second method uses three multipliers, and in the third method only one multiplier and shifting registers are used. A comparison is made on the accuracy of the implementation of three sigmoid functions. The digital hardware implementation of the sigmoid activation function was performed in the VHDL language in the Quartus II development environment. Modules of sigmoid functions are implemented on FPGA EP3C16F484C6 of the Cyclone III family. The simulation of their work and comparison of the necessary hardware resources and performance are performed. The implemented modules of sigmoid functions will be used in the design of multilayer feedforward neural networks.

REFERENCES

[1] Beiu V., Peperstraete J.A., Vandewalle J., Lauwereins R.: Close Approximations of Sigmoid Functions by Sum of Steps for VLSI Implementation of Neural. [Online resource]. Availible at: https://pdfs.semanticscholar.org/fdef/62a66787929bb80163219744d9eab041b203. pdf

[2] Tommiska M.T. Efficient digital implementation of the sigmoid function for reprogrammable logic. [Online resource]. Availible at: https://pdfs.semanticscholar.org/9bf6/4bae3f8528cd5ac72a4ae869a74563ff6c26.pdf

[3] Tisan Alin, Oniga Stefan, Mic Daniel, Buchman Attila: Digital Implementation of the Sigmoid Function for FPGA Circuits. [Online resource]. Availible at: http://users.utcluj.ro/~ATN/papers/ATN_2_2009_4.pdf

[4] Cyclone III Device Handbook. [Online resource]. Availible at: http://www.altera.com/literature/hb/cyc3/cyc3_ciii51001.pdf

[5] DE0 User Manual. [Online resource]. Availible at: http://esca.korea.ac.kr/teaching/FPGA_boards/DE0/DE0_User_Manual.pdf