Implementation of High Speed Tangent Sigmoid Transfer Function Approximations for Artificial Neural Network Applications on FPGA

Ismail KOYUNCU
Afyon Kocatepe University, Department of Electrical Electronics Engineering, 03200, Afyon, Turkey
ismailkoyuncu@aku.edu.tr

***Abstract*—Tangent Sigmoid (TanSig) Transfer Function (TSTF) is one of the nonlinear functions used in Artificial Neural Networks (ANNs). As TSTF includes exponential function operations, hardware-based implementation of this function is difficult. Thus, various methods have been proposed in the literature for the hardware implementation of TSTF. In this study, four different TSTF approaches on FPGA have been implemented using 32-bit IEEE 754–1985 floating point number standard, and their performance analyses and FPGA chip statistics are presented. The Van der Pol system ANN application was carried out using four different FPGA-based TSTF units presented. The Multilayer feed-forward neural network structure was used in the study. The FPGA chip statistics and sensitivity analyses were carried out by applying each TSTF structure to the exemplary ANN. The maximum operating frequency of ANNs designed on FPGA using the four different TSTF units varied between 184–362 MHz. The CORDIC-LUT-based ANN on FPGA was able to calculate 1 billion results in 3.284 s. According to the Van der Pol system ANN application carried out on FPGA, the CORDIC-LUT-based approach most closely reflected the reference ANN results. This study has a reference and key research for real-time artificial neural network applications used of tangent sigmoid one of the nonlinear transfer functions.**

***Index Terms*—real-time systems, field programmable gate arrays, artificial neural networks, approximation methods, transfer functions.**

I. Introduction

In recent years, **A**rtificial **N**eural **N**etworks (ANNs) are widely used in various fields. Biomedical [1], optimization [2], oscillator design [3], classification [4], synchronization [5], random number generator [6] and prediction [7] are examples of these areas of operation. In the literature, different structures can be found for implementing the ANNs as hardware, including the **D**igital **S**ignal **P**rocessor (DSP), **A**pplication **S**pecific **I**ntegrated **C**ircuits (ASIC) and **F**ield **P**rogrammable **G**ate **A**rray (FPGA). As a result of their overall network structure, ANNs operate in a parallel manner and therefore, when compared to ASIC and FPGA-based applications, a good speed performance cannot be obtained from DSP chips in ANN applications. Although quite a high performance can be achieved in ASIC-based ANN applications, the design process is lengthy. Furthermore, a single error during the design process of leads to substantial economic and time losses. In contrast, FPGA chips are capable of carrying out parallel processes, so an error during the design process can be fixed in a short amount of time and they can be reprogrammed at no extra cost [8]. Thus, as a result of the advantages of a flexible programming structure, FPGAs are preferrable in ANN-based applications.

Transfer functions used in ANNs are generally divided into two types: Linear and non-linear. **Rad**ial **Bas**is (RadBas), **Log**arithmic **Sig**moid (LogSig), wavelet, and **Tan**gent **Sig**moid (TanSig) are examples of non-linear transfer functions. As non-linear transfer functions contain exponential processes, hardware-based implementation of such processes is much more difficult than for other transfer functions. In this study, one of these non-linear transfer functions, the **T**an**S**ig **T**ransfer **F**unction (TSTF) was designed on FPGA using four different approaches. The design was coded in **V**ery High Speed Integrated Circuit (VHSIC) **H**ardware **D**escription **L**anguage (VHDL) using the 32-bit IEEE–754–1985 floating point number standard. The transfer functions designed were synthesized and tested for the Virtex–6 FPGA chip using ISE Design Tools System 14.1. In addition, the designed FPGA-based TSTF units were tested using the multilayer feed-forward ANN-based **V**an der **P**ol **S**ystem (VPS).

The FPGA chips, the 32-bit IEEE–754–1985 floating point number standard and the ANNs have been briefly mentioned in the second part of this study. General information on the TSTF approaches and the FPGA-based designs of these approaches is provided in the third part. Usage statistics for the FPGA chip and sensitivity analyses of the designs are also given. In the fourth part, the four different TSTF approaches of each design were tested on an FPGA-based exemplary ANN application. The last part presents assessments of the results obtained in the study and offers suggestions for future studies.

II. Background and Preliminaries

A. FPGA Chips

Currently, FPGA chips are used in many areas [9-11]. As FPGA chips are completely prefabricated and ready for customization, users can implement digital circuit designs by uploading the configuration file. Since the configuration of these chips takes very little time, circuit designs can be realized very quickly compared to ASIC implementations. The overall structure of the FPGA chip is shown in Figure 1. A typical FPGA device contains three configurable parts: an array of logic cells called **C**onfigurable **L**ogic **B**lock**s** (CLBs), a programmable interconnection network, and programmable **I**nput/**O**utput **B**lock**s** (IOBs) [12]. The CLBs are the most important part of the FPGA device. Each FPGA manufacturer implements a different type of CLB, including

Digital Object Identifier 10.4316/AECE.2018.03011

Look-**U**p **T**ables (LUTs), programmable flip-flops and several programmable multiplexers. The LUTs are function generators, capable of implementing any combinational logic function of their inputs.

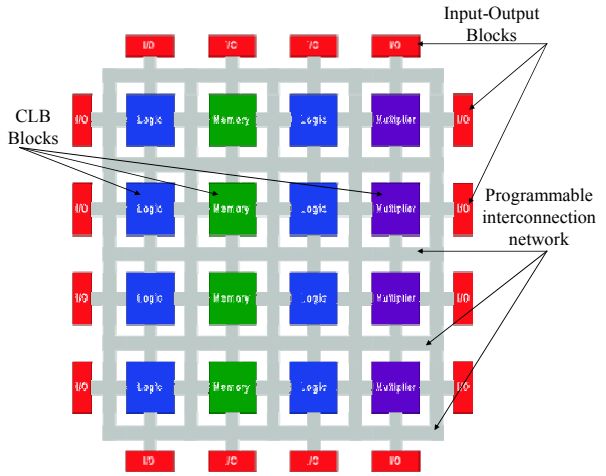

Figure 1. Basic structure of the FPGA chip

B. IEEE–754–1985 floating point number standard

The fixed point (IQ-Math) and IEEE–754–1985 floating point numbers are two examples of the many number standards used in FPGA chips. Although the numbers used in real time are infinite, in hardware platforms, the numbers are represented with approximate values depending on hardware capacity and restrictions. The IEEE–754–1985 floating point numbers ensures the representation of hardware restrictions in the closest way to reality by minimizing the effect of hardware restrictions. This standard has different specific formats such as the 16-bit half precision, 32-bit single precision and 64-bit double precision. The overall structure of the 32-bit IEEE–754–1985 floating point number standard is shown in Figure 2.

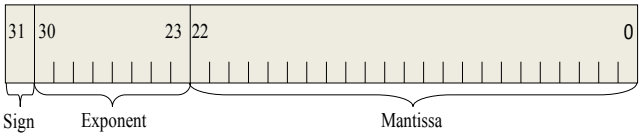

Figure 2. Representation of 32-bit IEEE–754–1985 single precision floating point number standard

The representation of the numbers according to the 32-bit IEEE–754–1985 floating point number standard is realized as shown in Eq. (1). In this equation, *v* is the decimal number value, *sign* is the bit value, *j* is the number of bits of the fractions (for 32-bit, *j=23*), *b* is the value of the fractional bits, and *exp* is the value of the exponent bits [13].

$$v = (-1)^{sign}\left(1 + \sum_{i=1}^{j} b_{23-i}\,2^{-i}\right)2^{(\exp-127)} \quad (1)$$

C. Artificial neural networks

Neuron cells used in ANNs generally consist of three parts: Inputs, the neuron body and outputs. The numerical model of the artificial neuron cell is shown in Eq. (2). Here; X_p represents the input value, *w* the weight, *b* the bias and O_p the output values. The X_p (input) values are multiplied by the *w* (weights) and summed with the *b* (bias). The neuron structures used in the application are divided into two groups: Those with bias and without bias. If the *b* (bias) value is used, this value is added to the total value obtained and it is put through a proper *f* transfer function suitable for the application. It is then transferred to the neuron outlet as the O_p value.

$$O_p = f\,(X_p.w + b) \quad (2)$$

The ANNs consist of three parts: The input layer, hidden layer and output layer. The overall structure of the ANN used in the design is shown in Figure 3 [14]. The input layer provides data input to the network, while the output layer ensures data output from the network. The number of input-output and hidden layers varies with the application. The transfer functions in the hidden layer are among the most important parts of the ANNs.

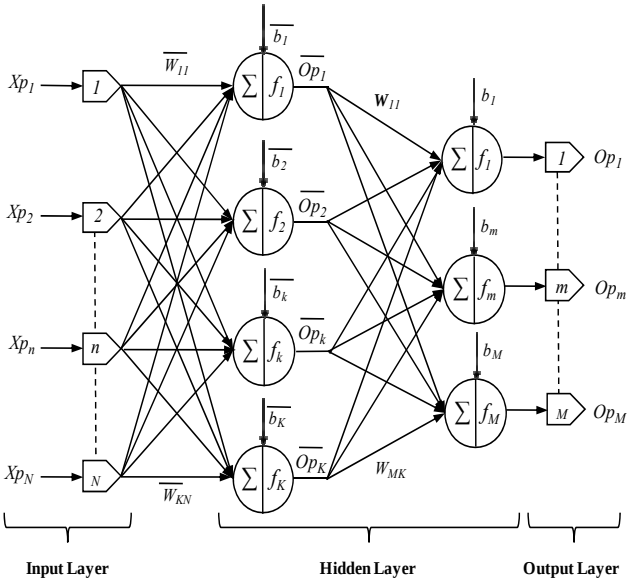

Figure 3. Multilayer feed-forward ANN structure

The $\overline{O}_{PK}$ expression calculated for the hidden layer outputs is shown in Eq. (3). Here, X_{pn} is the ANN input, $\overline{W_{kn}}$ the weight, $\overline{b_k}$ the bias, and f_K the transfer function for the hidden layer. The O_{PM} expression calculated for the ANN output layer is shown in Eq. (4). Here, W_{mk} is the weight, b_m the bias and f_M the TF for the output layer.

$$\overline{O}_{PK} = f_K\left[\left(\sum_{n=1}^{N} X_{pn}.\overline{W_{kn}}\right) + \overline{b_k}\right] \quad (3)$$

$$O_{PM} = f_M\left[\left(\sum_{k=1}^{K} \overline{O_{pk}}.W_{mk}\right) + b_m\right] \quad (4)$$

The matrix form of Eq. (4) is given in Eq. (5). In this equation, the calculation of the output of the neural networks requires matrix addition and multiplication operations which can be performed in parallel. The FPGAs are good candidates for exploiting parallelism in these kinds of calculations as they enable designers to place several functional units in their design and run them in parallel.

$$\begin{bmatrix} O_{p1} \\ O_{p2} \\ \vdots \\ O_{pm} \\ \vdots \\ O_{pM} \end{bmatrix} = f_M \left\{ \begin{bmatrix} W_{11} & W_{12} & \cdots & W_{1n} & \cdots & W_{1N} \\ W_{21} & W_{22} & \cdots & W_{2n} & \cdots & W_{2N} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ W_{k1} & W_{k2} & \cdots & W_{kn} & \cdots & W_{kN} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ W_{K1} & W_{K2} & \cdots & W_{Kn} & \cdots & W_{KN} \end{bmatrix} \cdot \begin{bmatrix} X_{p1} \\ X_{p2} \\ \vdots \\ X_{pn} \\ \vdots \\ X_{pN} \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_k \\ \vdots \\ b_K \end{bmatrix} \right\} \cdot \left\{ f_K \begin{bmatrix} W_{11} & W_{12} & \cdots & W_{1k} & \cdots & W_{1K} \\ W_{21} & W_{22} & \cdots & W_{2k} & \cdots & W_{2K} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ W_{m1} & W_{m1} & \cdots & W_{mk} & \cdots & W_{mK} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ W_{M1} & W_{M2} & \cdots & W_{Mk} & \cdots & W_{MK} \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \\ \vdots \\ b_M \end{bmatrix} \right\} \quad (5)$$

D. VPS

The VPS [15] is considered to have positive real-power nonlinearities in the restoring and damping force, including for fractional powers. The second order differential equation of the VPS is given in Eq. (6). The VPS's damping parameter value is $\mu=0.5$ in this equation. The values of typical initial conditions are $x_0= 1.0$ and $u_0=-0.97$.

$$\begin{aligned} \frac{dx}{dt} &= u \\ \frac{du}{dt} &= \mu(1-x^2)u - x \end{aligned} \quad (6)$$

III. FPGA-BASED TANSIG TRANSFER FUNCTION DESIGNS AND CHIP STATISTICS

A. FPGA-based TanSig transfer function designs

Transfer functions used in ANNs are generally divided into two groups: Linear and nonlinear. Nonlinear transfer functions like TSTF include exponential processes, making its hardware-based implementation quite difficult. To this purpose, various methods for the implementation of its exponential function such as the LUT, Taylor Series and Elliott approaches have been presented in the literature.

If the value interval to be used in the LUT-based approach is definite and low, the result values of the function are recorded as e^u. The result of the e^u function according to the u input value is transferred to the output without the need for any calculation process. While quite rapid results can be obtained with this method, it requires a very large hardware resource in order to obtain precise results [16]. Thus, the LUT approach is generally preferred for specific hardware applications.

Another e^u value calculation approach is the Taylor Series expansion. With a being a real or complex number, the Taylor Series of an $f(u)$ function is defined as shown in Eq. (7).

$$f(u) = f(a) + \frac{f'(a)}{1!}(u-a) + \frac{f''(a)}{2!}(u-a)^2 + \cdots + \frac{f^n(a)}{n!}(u-a)^n \quad (7)$$

If $a=0$, the series is identified as the Maclaurin series. The Taylor Series for e^u exponential function is as shown in Eq. (8).

$$e^u = 1 + u + \frac{u^2}{2!} + \frac{u^3}{3!} + \frac{u^4}{4!} + \frac{u^5}{5!} + \ldots\ldots + \frac{u^n}{n!} \quad (8)$$

After obtaining the e^u exponential function, the TSTF is calculated by applying it to the expression given in Eq. (9).

$$TanSig(u) = \left(e^{2u}-1\right)/\left(e^{2u}+1\right) \quad (9)$$

The convergence of the function generally increases as the expansion order of the Taylor Series increases [17]. However, the increase in the number of processes in hardware implementation also leads to an increase in hardware rate. Thus, a fifth order TS expansion (TS-5) was used in order to achieve convergence of the TSTF in this study. A design was made using VHDL in order to implement the TSTF on FPGA via the Taylor series. The block diagram of the TS-5 based unit is shown in Figure 4. The 32-bit IEEE–754–1985 single precision floating point number standard was used in the implementation of all approaches presented on FPGA. The **I**ntellectual **P**roperty (IP) cores used in the designs were generated using the Xilinx Core Generator System. The *Mult.1-Mult.5* units perform multiplication, the *Subt.* unit performs subtraction, the *Adder1-Adder5* units perform addition, and the *Div.1-Div.5* units perform division in accordance with the IEEE 754–1985 floating point standard. The *Latency1-Latency4* units ensure the delay for the processes as they are realized simultaneously as pipelines. The *1.0f*, *2.0f*, *6.0f*, *24.0f* and *120.0f* are the floating point fixed values that are defined in accordance with the IEEE 754–1985 standard in VHDL. The unit produces the first result after 77 clock cycles. Because the unit operates as a pipeline, it can produce new results during each clock cycle after this initial period.

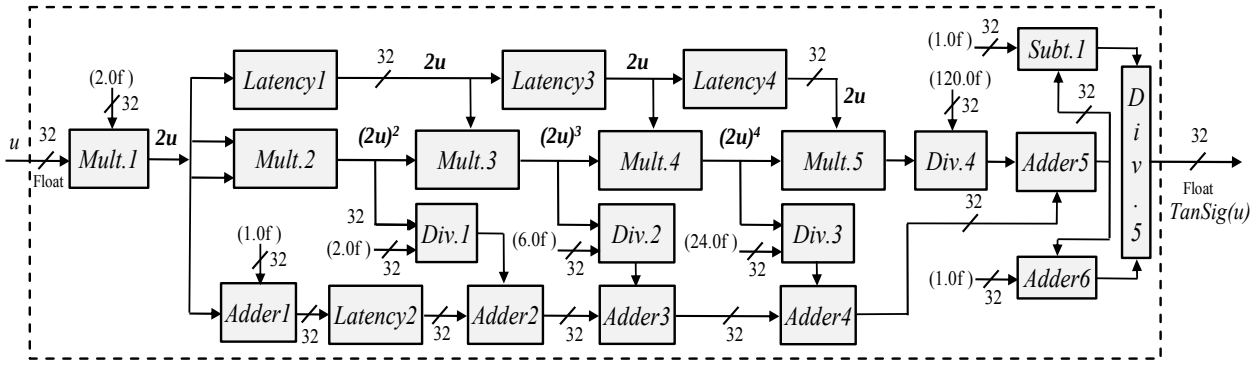

Figure 4. Block diagram of TS-5 based TSTF on FPGA

Another TSTF approach was put forth by D. L. Elliott, as shown in Eq. (10). The Elliott-93 approach does not require a great many numerical processes in order to calculate the TSTF, and the calculation of the e^u function is also not required. Thus, the Elliott-93 approach is preferred as its hardware is easily implemented, although it does not produce very precise results.

$$\sigma_{e_{93}}(u) = u/(1+|u|) \tag{10}$$

The unit was designed using VHDL in order to implement TSTF on FPGA using the Elliott-93 approach. The block diagram of the unit is shown in Figure 5. As is also seen in the figure, its hardware implementation is quite easy as it contains very few processes when compared to other approaches. The *Abs* unit was used in order to obtain the absolute value of the received u value. The unit operates as a pipeline. After 21 clock cycles, the unit produces the first result and can produce new results throughout each subsequent clock cycle. The Elliott-2 is another approach that was development from the Elliott-93 approach. The expression of the Elliott-2 approach is shown in Eq. (11). Here, *sgn(u)* is the signum function. The calculation of TSTF in the Elliott-2 approach requires more processes than the Elliott-93; the convergence is greater when compared to the real TSTF. Thus, in the applications, the Elliott-2 approach is more advantageous than the Elliott-93 approach as it produces more precise results.

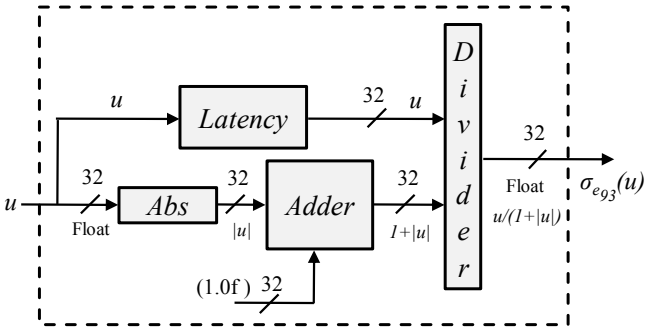

Figure 5. Block diagram of Elliott-93-based TSTF approach on FPGA

$$\sigma_{e_2}(u) = \mathrm{sgn}(u)\cdot u^2/(1+u^2) \tag{11}$$

The block diagram of the unit designed using the Elliott-2 approach for the TSTF on FPGA is shown in Figure 6. The unit produces the first result after 35 clock cycles and works as a pipeline, after which each unit can produce new results throughout each clock cycle.

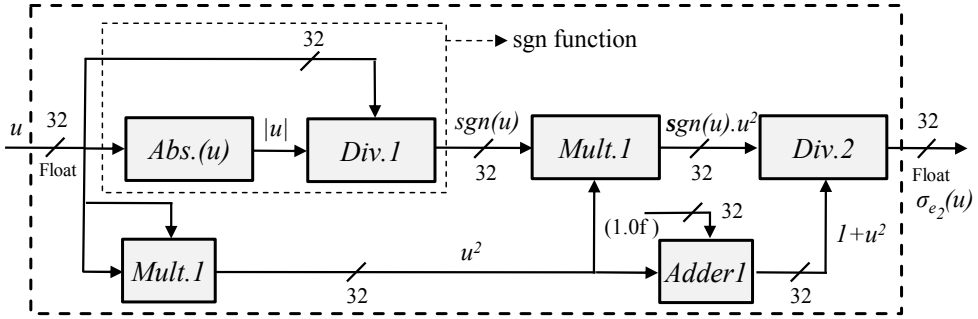

Figure 6. Block diagram of Elliott-2-based TSTF approach on FPGA

Another approach that ensure the calculation of TSTF on FPGA is the **CO**ordinate **R**otation **DI**gital **C**omputer (CORDIC) and LUT-based approach. In this approach, e^u is calculated using the *Cosh(u)* and *Sinh(u)* hyperbolic function values [18]. However, the CORDIC unit can only calculate the *Sinh(u)* and *Cosh(u)* values in the interval between $-\pi/4$ and $\pi/4$, i.e. the CORDIC unit can only make calculations in the interval between $e^{-0.7853981}$ and $e^{0.7853981}$. The block diagram of this approach is shown in Figure 7.

The unit calculates the required e^u value in two parts, as shown in Eq. (12). In the first part, the ζ remainder from the division of u value received in the unit within the ψ value is calculated, ψ being the range that the CORDIC unit can calculate. The ζ value, once calculated, is converted to the 23-bit fixed point number type. Here, the ζ value is divided into two parts: a 7-bit whole number part (ω) and a 16-bit decimal part (λ). It was made suitable for the multiplication with an 18-bit ψ value by adding bits (00) to the least significant bit of the λ value. The τ value to be calculated by the CORDIC unit was then obtained by multiplying the λ and ψ values. After the CORDIC unit calculated the e^τ value, this value was again converted to a floating point number. In the second part, the $e^{0.75\omega}$ value that is equal to the whole number section of the part obtained is found from LUT. The e^{2u} value is calculated by multiplying the two $e^{0.75\omega}$ and e^τ intermediate values obtained. The *TanSig(u)* TF is then calculated by performing a floating addition and subtraction. Although the FPGA chip source rates used in the design were high, the e^u calculation range was very high.

$$TanSig(u) = \frac{\left((e^{(2u \bmod \psi)} \cdot e^{\mathrm{int}(2u/\psi)} - 1\right)}{\left(e^{(2u \bmod \psi)} \cdot e^{\mathrm{int}(2u/\psi)} + 1\right)} \tag{12}$$

The unit inputs and outputs are suitable for the 32-bit IEEE–754–1985 floating point number standard, but the CORDIC unit operates in accordance with the fixed number standard; the fixed point number standard was used in these sections. The unit produces the first result after 76 clock cycles. Because it operates as a pipeline, the unit can produce new results throughout each clock cycle after this initial period.

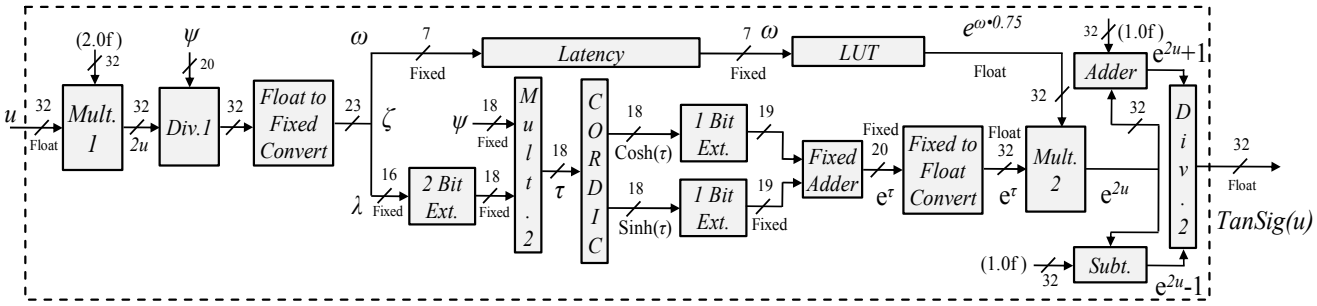

Figure 7. Block diagram of CORDIC-LUT-based TSTF approach on FPGA

B. Performance of FPGA-based TSTF approximations and chip statistics

In the study presented, four different TSTFs were designed on FPGA: The TS-5, Elliott-93, Elliott-2 and CORDIC-LUT approach units. The designs were coded in VHDL in accordance with the 32-bit IEEE–754–1985 floating point number standard. The designed TSTFs were tested and synthesized for the Virtex–6 (XC6VLX240T–3FF784) FPGA chip using Xilinx ISE Design Suite 14.1. The testbench structure was created through Xilinx ISE Design Suite for the test processes and each FPGA-based TSTF unit using an exemplary data set (-6, -5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5, 6) during the testing stage was tested. Test results obtained via the Xilinx ISE Design Tools Simulator (DTS) for the TSTF units of the four different structures implemented on FPGA and the numerically calculated real TSTF results are shown in Figure 8.

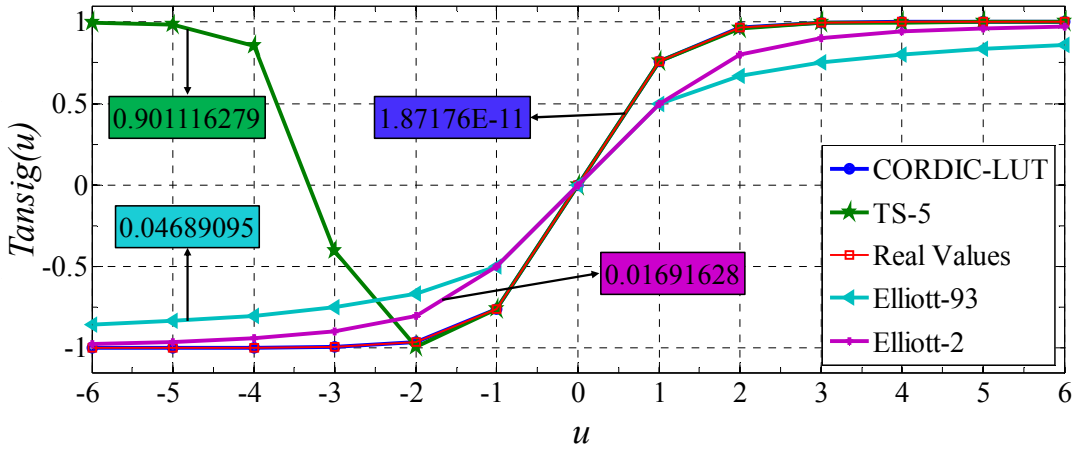

Figure 8. Tests and MSE results of FPGA-based TSTF units

The **M**ean **S**quared **E**rror (MSE), the **R**oot **M**ean **S**quare **E**rror (RMSE), the **M**ean **A**bsolute **E**rror (MAE) and the **M**ean **A**bsolute **P**ercentage **E**rror (MAPE) analyses were carried out on the results using Eqs. (13), (14) and (15). The analysis results are presented in Figure 8 and Table I. According to the results obtained, the CORDIC-LUT-based TSTF unit achieved the closest convergence with the real values.

$$RMSE = \sqrt{MSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(\hat{u}_i - u_i\right)^2} \quad (13)$$

$$MAE = \frac{1}{n}\sum_{i=1}^{n}\left|\hat{u}_i - u_i\right| \quad (14)$$

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{\hat{u}_i - u_i}{\hat{u}_i}\right| \quad (15)$$

TABLE I. MSE, RMSE, MAE AND MAPE ANALYSES OF FPGA-BASED TSTF DESIGNS

TSTF Structure	MSE	RMSE	MAE	MAPE
Elliott-93	0.046891	0.216543	0.201963	0.295233
Elliott-2	0.016916	0.130063	0.099109	0.189652
TS-5 based	0.901116	0.949271	0.496668	0.574049
CORDIC-LUT	1.87E–11	4.33 E–06	1.81 E–06	0.076925
CORDIC [19]	2.84E–05	5.33E–03	--	--
PMFEP [20]	5.14E–05	7.17E–03	--	--

Four different FPGA-based TSTF designs were synthesized using the Xilinx ISE 14.1 DTS for the Virtex-6. The FPGA chip statistics obtained from the Place and Route operation in Xilinx Virtex-6 chip, the pipeline latency of the units and their maximum operating frequencies obtained after the Place and Route process are shown in Table II.

TABLE II. CHIP STATISTICS OF FPGA-BASED TSTF DESIGNS

TanSig TF		Number of Slice Registers	Number of Slice LUTs	Number of Occupied Slices	Number of IOBs	Latency (Clock cycle)	Maximum Frequency (MHz)
Elliott-93 based	Used	1,048	1,194	412	67	21	362.344
	Usage Rate (%)	1	1	1	16		
Elliott-2 based	Used	3,056	3,225	1,067	67	35	362.344
	Usage Rate (%)	1	2	2	16		
TS-5 based	Used	7,739	9,774	3,087	67	77	184.751
	Usage Rate (%)	2	6	8	16		
CORDIC-LUT based	Used	4,711	5,228	1,604	67	76	361.461
	Usage Rate (%)	1	3	4	16		

IV. REAL TIME ANN APPLICATION OF VPS ON FPGA

In this study, the VPS dataset was generated using the fifth-order Runge-Kutta-Butcher algorithm (RK5B) in order to test the TSTFs in four different structures designed on FPGA. There are two inputs (x and u) in the dataset which are state variables of the VPS. For the VPS based application, the multilayer feed forward ANN structure was first modelled in the Matlab® program. The dataset of the VPS (1000 values) was divided into two parts during the ANN-based modelling stage, 800 of the values being training data and 200 being test data. The FPGA-based multilayer feed-forward ANN contains two inputs in the input layer, four neurons containing TSTF in the hidden layer and two neurons containing Pureline TF in the output layer. The Levenberg-Marquardt algorithm was used during the training stage and the ANN training performance reached 2.87×10^{-12} (MSE) at the end of 20,000 epochs. After this process, the ANN was tested using 200 test data and the test performance of the ANN was found to be 1.87×10^{-10} (MSE). The bias and weight values used in the ANN were taken as a reference after the test stage and these data were used in the structure of the FPGA-based feed-forward ANN via conversion to IEEE–754–1985 number standard. The ANN based VPS in this study was modelled so as to be studied on FPGA using the 32-bit IEEE–754–1985 floating point number standard and coded in VHDL. Units such as divider, multiplier and adder that are in line with the floating point number standard used in the designs were generated using the IP Core Generator developed via Xilinx ISE DTS. Simulation results obtained from the designs generated via Xilinx ISE DTS on FPGA are shown below. Xilinx ISE simulation results for the ANN-based VPS Elliott-93 approach are shown in Figure 9, with the ANN-based VPS Elliott-2 approach in Figure 10, with the ANN-based VPS TS-5 approach in Figure 11, and with the ANN-based VPS CORDIC-LUT approach in Figure 12. The ANN-based VPS TS-5 approach produced its first results after 134 clock cycles, the ANN-based VPS Elliott-93 approach after 78 clock cycles, the ANN-based VPS Elliott-2 approach after 87 clock cycles, and the ANN-based VPS CORDIC-LUT approach after 130 clock cycles. With all four ANN-based TSTF approaches on FPGA, after this initial clock cycle, a result was produced in each subsequent clock cycle. Moreover, the CORDIC-LUT-based VPS ANN on FPGA was able to calculate 1 billion results in 3.284 s.

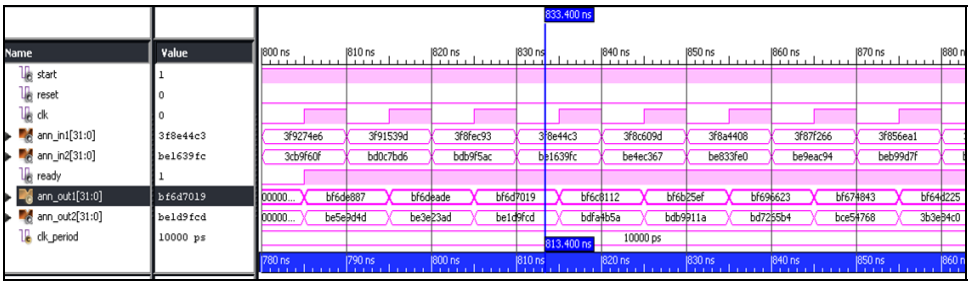

Figure 9. ANN outputs of FPGA-based VPS Elliott-93 approach obtained via Xilinx ISE DTS

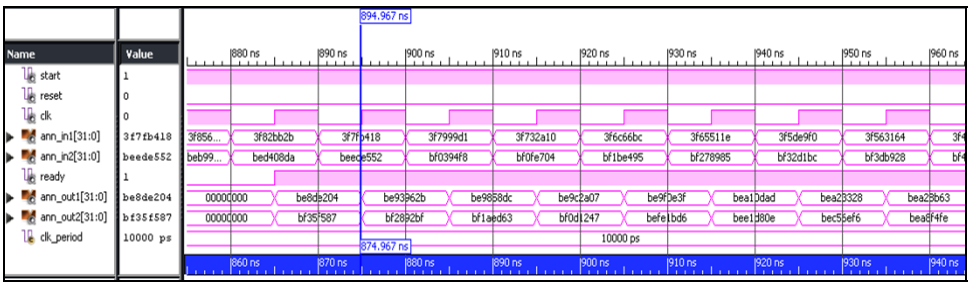

Figure 10. ANN outputs of FPGA-based VPS Elliott-2 approach obtained via Xilinx ISE DTS

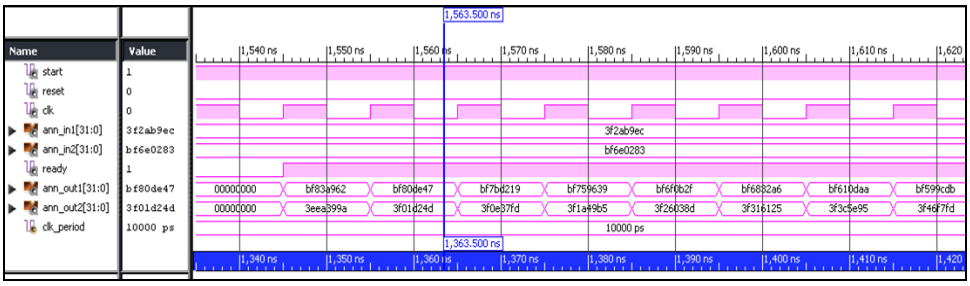

Figure 11. ANN outputs of FPGA-based VPS TS-5 approach obtained via Xilinx ISE DTS

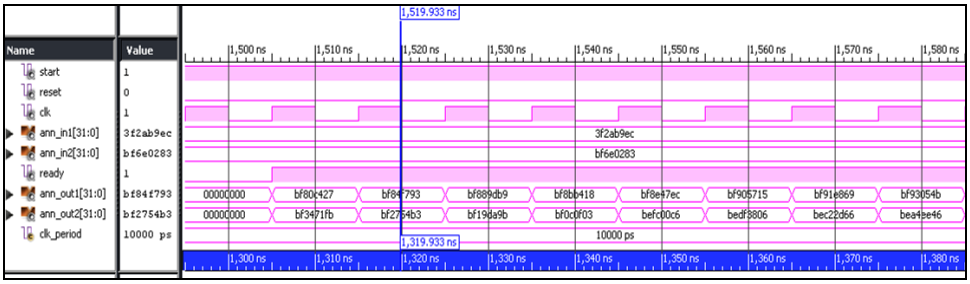

Figure 12. ANN outputs of FPGA-based VPS CORDIC-LUT approach obtained via Xilinx ISE DTS

In this study, MSE, RMSE, MAE and MAPE analyses were performed on the results obtained from the ANN-based VPSs on FPGA and the analyses results were presented in Table III. The ANN outputs of the FPGA-based VPS CORDIC-LUT approach unit, according to the results obtained, achieved the closest convergence to the real values. In addition, output signals of the ANN-based VPS approaches on FPGA obtained via Xilinx ISE DTS are given in Figure 13 and the absolute error results of the ANN-based VPS approaches on FPGA obtained via Xilinx ISE DTS are given in Figure 14.

TABLE III. MSE, RMSE, MAE AND MAPE ANALYSES OF ANN-BASED VPS TSTF DESIGNS ON FPGA.

TSTF Structure	MSE	RMSE	MAE	MAPE
Elliot-93	0.162015	0.402511	0.345268	0.887363
Elliot-2	0.028407	0.168543	0.153141	0.532057
TS-5	0.669619	0.818302	0.716914	3.425274
CORDIC-LUT	1.02E–08	1.00 E–04	7.83 E–05	3.58 E–04

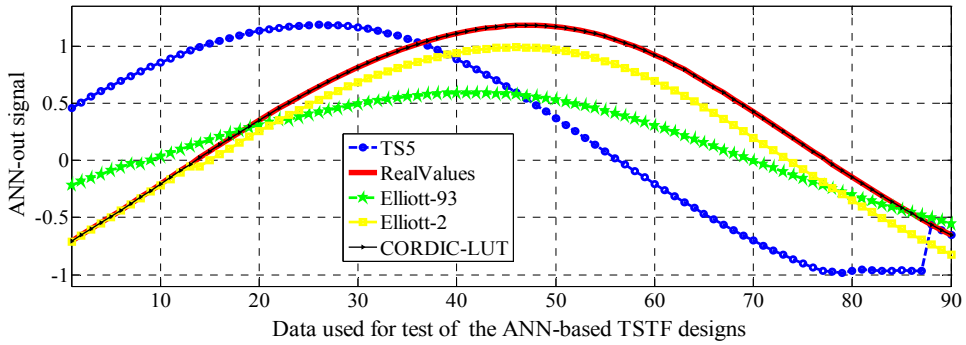

Figure 13. Output signals of the ANN-based VPS approaches on FPGA obtained via Xilinx ISE DTS

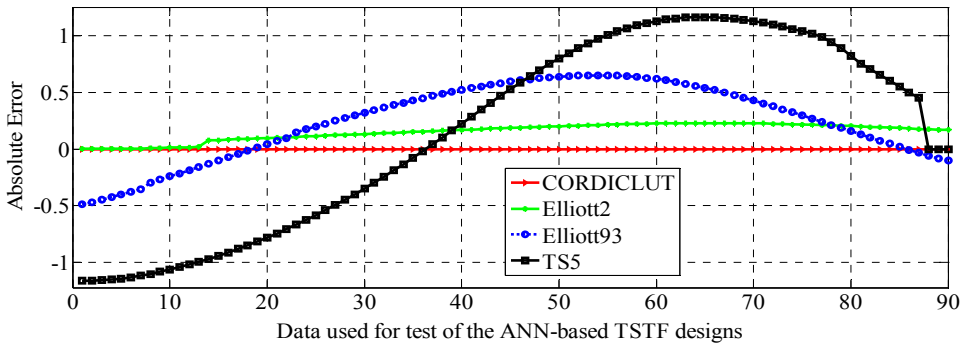

Figure 14. Absolute errors of ANN-based VPS approaches on FPGA obtained from Xilinx ISE DTS

The real time multilayer feed-forward ANN-based VPS-TSTF approaches were synthesized for the Xilinx Virtex–6 (XC6VLX240T–3FF784) chip via Xilinx ISE 14.1 DTS. The FPGA chip statistics and the operating frequencies obtained after the Place and Route process are shown in Table IV. All FPGA-based ANN designs work as pipeline.

TABLE IV. FPGA CHIP STATISTICS AND ERROR ANALYSIS OF REAL-TIME ANN-BASED VPS TSTF APPROACHES

ANN Structure		Number of Slice Registers	Number of Slice LUTs	Number of Occupied Slices	Number of IOBs	Maximum Frequency (MHz)
Elliot-93 based	Used	20,864	20,757	6,420	131	362.344
	Usage Rate (%)	6	13	17	32	
Elliot-2 based	Used	28,820	28,648	8,278	131	356.010
	Usage Rate (%)	9	19	21	32	
TS-5 based	Used	47,477	55,030	17,126	131	184.751
	Usage Rate (%)	15	36	45	32	
CORDIC-LUT based	Used	34,493	36,773	11,795	131	304.534
	Usage Rate (%)	11	24	31	32	

V. CONCLUSION

The TanSig transfer function (TSTF), a nonlinear transfer function used in ANN applications, was coded in VHDL so as to be studied on FPGA using four different approaches. The 32-bit IEEE–754–1985 floating point number standard was used in the designs. The precision, performance and chip usage rate of the four different TSTF approaches designed on FPGA were analyzed. According to the results, CORDIC-LUT based TSTF unit was the closest to the real TanSig value. The ANN application of VPS was conducted in an FPGA-based multilayer feed-forward structure in order to test the TSTF approaches, and an error analyses were performed using the results. The maximum operating frequency of the ANN-based VPS on FPGA realized by using four different TSTF units varied between 184-362 MHz. The CORDIC-LUT based ANN on FPGA was able to calculate 1 billion results in 3.284 s. According to the ANN-based VPS application on FPGA, the unit with the lowest RMSE, MSE, MAE and MAPE results was the CORDIC-LUT based approach. Future studies can be carried out by adapting the e^{u} units presented in this study to other transfer functions containing exponential processes. Moreover, real time high speed ANN applications can be conducted in different areas using the TSTFs presented in this study.

REFERENCES

[1] O. Geman, H. Costin, "Automatic Assessing of Tremor Severity Using Nonlinear Dynamics, Artificial Neural Networks and Neuro-Fuzzy Classifier," Advances in Electrical and Computer Engineering, vol. 14(1), pp. 133-138. 2014. doi:10.4316/AECE.2014.01020

[2] E. Betiku, A. E. Taiwo, "Modeling and optimization of bioethanol production from breadfruit starch hydrolyzate response surface methodology and artificial neural network," Renewable Energy, In vivo thermography-based image for early detection of breast cancer using two-tier segmentation algorithm and artificial neural network vol. 74, pp. 87–94, 2015. doi 10.1016/j.renene.2014.07.054

[3] M. Alçın, İ. Pehlivan, İ. Koyuncu, "Hardware design and implementation of a novel ANN-based chaotic generator in FPGA," Optik-International Journal for Light and Electron Optics, vol. 127, pp. 5500–5505, 2016. doi:10.1016/j.ijleo.2016.03.042

[4] W. Zang, X. Liu, W. Bi, "An artificial neural network classification model based on DNA computing," Human Centered Computing Lecture Notes in Computer Science, vol. 8944, pp. 880–889, 2015. doi:10.1007/978-3-319-15554-8_82

[5] G. Zhang, Y. Shen, "Exponential synchronization of delayed memristor-based chaotic neural networks via periodically intermittent control," Neural Networks, vol. 55, pp. 1–10, 2014. doi:10.1016/j.neunet.2014.03.009

[6] M. Tuna, C. B. Fidan, "A Study on the importance of chaotic oscillators based on FPGA for true random number generating (TRNG) and chaotic systems," Journal of the Faculty of Engineering and Architecture of Gazi University, vol. 33, pp. 469–486, 2018. doi:10.17341/gazimmfd.416355

[7] S. Roy, R. Banerjee, P. K. Bose, "Performance and exhaust emissions prediction of a CRDI assisted single cylinder diesel engine coupled with EGR using artificial neural network," Applied Energy, vol. 119, pp. 330–340, 2014. doi:10.1016/j.apenergy.2014.01.044

[8] M. Alcın, I. Pehlivan, I. Koyuncu, "The Performance Analysis of Artificial Neural Network Based Shimizu-Morioka Chaotic System with Respect to Sample Numbers," Balkan Journal of Electrical and Computer Engineering, vol. 3(4), pp. 252–255, 2015. doi:10.17694/bajece.24157

[9] M. Tuna, C. B. Fidan, "Electronic circuit design, implementation and FPGA-based realization of a new 3D chaotic system with single equilibrium point," Optik-International Journal for Light and Electron Optics, vol. 127(24), pp. 11786–11799, 2016. doi:10.1016/j.ijleo.2016.09.087.

[10] A. Melnyk, V. Melnyk, "Self-Configurable FPGA-Based Computer Systems," Advances in Electrical and Computer Engineering, vol. 13(2), pp. 33-38. 2013. doi:10.4316/AECE.2013.02005

[11] M. Tuna, I. Koyuncu, C. B. Fidan, İ. Pehlivan, "Real time implementation of a novel chaotic generator on FPGA," 23nd Signal Processing and Communications Applications Conference (SIU), pp. 698–701, 2015. doi:10.1109/SIU.2015.7129921

[12] İ. Koyuncu, A. T. Özcerit, "The design and realization of a new high speed FPGA-based chaotic true random number generator," Computers and Electrical Engineering, vol. 58, pp. 203–214, 2017. doi:10.1016/j.compeleceng.2016.07.005

[13] M. A. Çavuşlu, C. Karakuzu, S. Şahin, M. Yakut, "Neural network training based on FPGA with floating point number format and it's performance," Neural Computing and Applications, vol. 20, pp. 195–202, 2011. doi:10.1007/s00521-010-0423-3

[14] A. Saad, K. Hany, A. Amin, "Three-dimensional turbulent swirling flow reconstruction using artificial neural networks," Journal of Mechanical Engineering and Automation, vol. 4, pp. 1–9, 2014. doi:10.5923/j.jmea.20140401.01

[15] S. Brezetskyi, D. Dudkowski, T. Kapitaniak, "Rare and hidden attractors in Van der Pol-Duffing oscillators," The European Physical Journal Special Topics, vol. 224, pp. 1459–1467, 2015. doi:10.1140/epjst/e2015-02471-2

[16] D. Suzuki, M. Natsui, A. Mochizuki, S. Miura, H. Honjo, H. Sato, S. Fukami, S. Ikeda, T. Endoh, H. Ohno, T. Hanyu, "Fabrication of a 3000-6-input-LUTs embedded and block-level power-gated nonvolatile FPGA chip using p-MTJ-based logic-in-memory structure," IEEE Symposium on VLSI Circuits Digest of Technical Paper, pp. 172–173, 2015. doi:10.1109/VLSIT.2015.7223644

[17] P. Nilsson, A. U. R. Shaik, Gangarajaiah R., Hertz E., "Hardware implementation of the exponential function using Taylor series," 32nd NORCHIP Conference, pp. 1–4, 2014. doi:10.1109/NORCHIP.2014.7004740

[18] İ. Koyuncu, İ. Şahin, C. Gloster, N. K. Sarıtekin, "A Neuron Library for Rapid Realization of Artificial Neural Networks on FPGA: A Case Study of Rössler Chaotic System," Journal of Circuits, Systems and Computers, vol. 26(01), pp. 1750015, 2017. doi:10.1142/S0218126617500153

[19] V. Tiwari, N. Khare, "Hardware implementation of neural network with sigmoidal activation functions using CORDIC," Microprocessors and Microsyststems, vol. 39, pp. 373–381, 2015. doi:10.1016/j.micpro.2015.05.012

[20] D. Baptista, D. F. Morgado, "Low-resource hardware implementation of the hyperbolic tangent for artificial neural networks," Neural Computing and Applications, vol. 23, pp. 601–607, 2013. doi:10.1007/s00521-013-1407-x