DIGITAL IMPLEMENTATION OF THE SIGMOID FUNCTION FOR FPGA CIRCUITS

Alin TISAN, Stefan ONIGA, Daniel MIC, Attila BUCHMAN
Electronic and Computer Department, North University of Baia Mare, Baia Mare, Romania
Str. Dr. Victor Babeş, nr. 62A, tel. 0362-401265 ext. 234, alin.tisan@ubm.ro

Abstract: In this paper, is proposed a method to implement in FPGA (Field Programmable Gate Array) circuits different approximation of the sigmoid function. Three previously published piecewise linear and one piecewise second-order approximation are analyzed from point of view of hardware resources utilization, induced errors caused by the approximation function and bits representation, power consumption and speed processing. The major benefit of the proposed method resides in the possibility to design neural networks by means of predefined block systems created in System Generator environment and the possibility to create a higher level design tools used to implement neural networks in logical circuits.

Key words: *Sigmoid approximation, FPGA implementation, System Generator*

I. INTRODUCTION

A main component of an artificial neuron's behaviour and usually a bottleneck for its speed performance is the sigmoid activation function block. Hardware implementation of the considered function (sigmoid function), as defined in literature, [1, 2], implies important hardware resources consumption.

$$output(net) = f(bias + \sum_{k=1}^{N} w_k x_k)$$
$$output(net) = \frac{1}{1 + e^{-\left(bias + \sum_{k=1}^{N} w_k x_k\right)}} \quad (1)$$

In order to reduce such consumption is useful to adopt different approximations (in function of the available hardware resources) with minimum errors.

The principal classical methods to digitally implement the activation function are Look-up tables and truncation of the Taylor series expansion. The second method can further be divided in: sum-of-steps, piece-wise linear, combination of the previous, or others.

The best results reported in the literature show errors of 8% to 13.1% for sum-of-steps approximations and ± 2.45% to ± 1.14% [3,4] for piece-wise linear approximation. Also, there are approximations with lower errors, but uses floating-point multiplications, which makes it far too complicated for a practical VLSI implementation.

The hardware implementations of the sigmoid approximations are done in System Generator, part of the Simulink/Matlab environment.

In the following, the hardware resources consumption and the generated errors for different FPGA implementation of five approximations proposed in literature are analyzed.

II. HARDWARE IMPLEMENTATION OF THE SIGMOID FUNCTION

A. *Look-up Tables*

The hardware resources of an FPGA circuit utilized for Look-up Tables implementation are considerable large only in the case in which the ROM memory is not FPGA implemented with predefined RAM Block. In table I, are presented the equivalent gates counted for design.

TABLE I. HARDWARE RESOURCES UTILIZATION OF THE 4VSX35 FPGA CIRCUIT

Device Utilization Summary			
Logic Utilization	Used	Available	Utilization
Slices	0	15,360	0 %
LUTs	0	30,720	0 %
BRAMs	1	192	0.52 %
DSPs	0	192	0 %
Total equivalent gate count for design	196,60	3.5M	3.7 %

Manuscript received April 27, 2009, reviewed June 16, 2009

B. A-law approximation

The method is proposed by Myers and Hutchinson in order to obtain a modified curve with the gradient of each linear segment expressed as a power of two. In this way the multipliers are replaced with shifters. The curve is approximated by seven segments and its breakpoints are presented in Table II.

TABLE II. BREAKPOINTS OF A-LAW BASED SIGMOID APPROXIMATION

x	-8.0	-4.0	-2.0	-1.0	1.0	2.0	4.0	8.0
y	0.0	0.0625	0.12	0.25	0.75	0.87	0.937	1.0

In order to evidence the errors introduced by the A_law approximation, in fig. 1 and fig. 2 are shown the sigmoid function beside hardware implemented A_law function. The numbers of bits allocated for digital signal processing in FPGA circuit are: 32 bits with 16 bits binary point, noted with (32,16), 16 bits with 8 bits binary point, (16,8) and 8 bits with 4 bits binary point, (8,4).

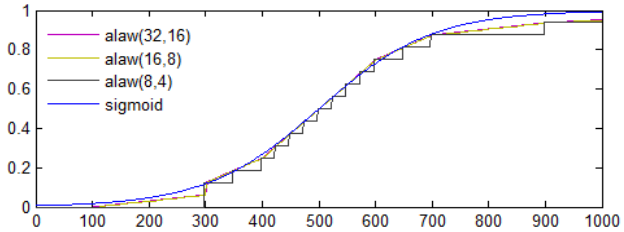

Figure 1. Comparative representation of the sigmoid function and alaw approximation

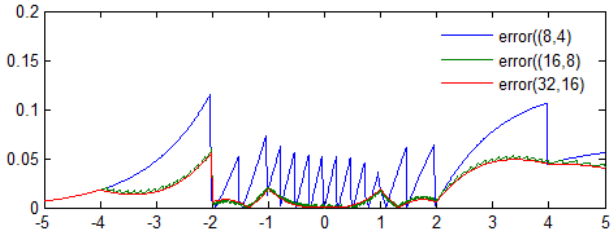

Figure 2. Errors introduced by the A_law approximation for different bits representation

After analyzing the results presented in fig.2, it can be conclude that for a (16,8) representation, the maximum error introduced by A_law approximation is 5.63% and the mean error is 0.6335%. These errors increase with decreasing of the bits representation: 11.51 % for (8,4) representation.

The hardware implementation of the A_law function is presented in fig.3.

The hardware resources utilized are resumed to 1 MCode block for compare functions, 4 shift registers, 2 multiplexers and 1 sum block. All the used blocks are part of the System Generator library (Xilinx Blockset).

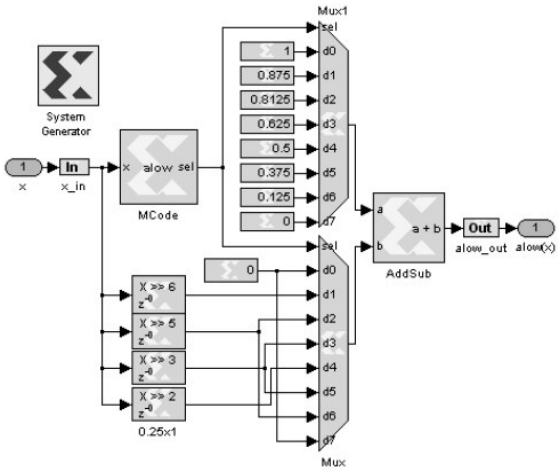

Figure 3. Hardware architecture of the A_law approximation

For determination of the total equivalent gates used for hardware implementation of the A_law function, ISE (Xilinx Integrated Software Environment) was used. Its report is presented in table III.

TABLE III. RESOURCES UTILIZATION OF THE 4VSX35 FPGA CIRCUIT FOR HARDWARE IMPLEMENTATION OF THE A_LAW APPROXIMATION

Device Utilization Summary				
Logic Utilization	Used			Available
	(32,16)	(16,8)	(8,4)	
Slices	185	74	23	15,360
LUTs	101	40	16	30,720
BRAMs	0	0	0	192
DSPs	0	0	0	192
Total equivalent gate count for design	1,653	1,440	204	3.5M

C. Alippi and Storti-Gajani Approximation

The Alippi and Storti-Gajani Approximation lies on the selecting of a set of breakpoints of the first derivate and setting the function as sum of power of two's numbers. For the reason that the sigmoid function has a symmetry point at coordinates (0, 0.5) only half pairs of x-y will be calculated, [3]:

$$y_{x>0} = 1 - y_{x\le 0} \tag{2}$$

Taking in consideration only the negative numbers, negative x-axis, and defining the integral part of x as INT(x), the decimal part of x with its own sign, denoted FRAC(x), is defined as follows:

$$FRAC(x) = x + \left|INT(x)\right| \tag{3}$$

Therefore, the expression of the Alippi function may be defined as:

$$Alippi(x)=\begin{cases}1-\dfrac{1/2+FRAC(-x)/4}{2^{|INT(x)|}} & \text{for } x>0\\ \dfrac{1/2+FRAC(x)/4}{2^{|INT(x)|}} & \text{for } x\le 0\end{cases} \quad (4)$$

Hardware implementation of the Alippi function is made in the System Generator Toolbox, part of Matlab/Simulink environment. The resources utilized are resumed at 8 shift registers used for dividing the Frac(x) signal to 4 and for shifting it with a number of Int(x) bits, 4 multiplexers, 3 subtraction blocks, 2 slice blocks and 1 comparator block. The numerical representation used for signals processing, i.e. number of bits and binary point are set in a pop-up parameters window as global variables. For each blocks in part the binary point is customized function of the maximum value of the function. In this way it can be obtained maximum performance at a given number of bits. The hardware architecture is shown in fig.4. In order to evidence the errors introduced by the Alippi approximation, in fig. 5 and fig. 6 are shown the sigmoid function beside Alippi function hardware implemented.

The numerical representation used for the analysis is 32 bits with 16 bits binary point, (32,16), 16 bits with 8 bits binary point, (16,8), 8 bits with 4 bits binary point, (8,4) and 5 bits with 3 bits binary point, (5,3).

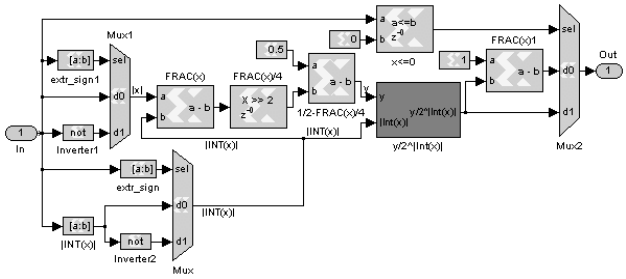

Figure 4. Hardware architecture of the Alippi function approximation

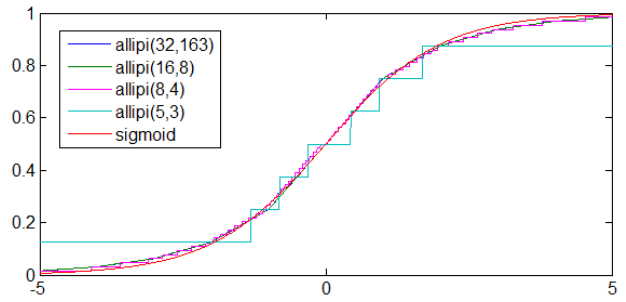

Figure 5. Comparative representation of the sigmoid function and Alippi approximation

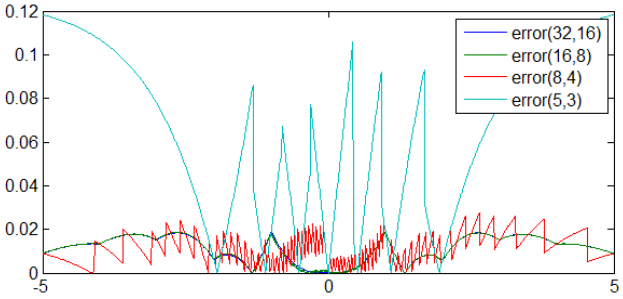

Figure 6. Errors introduced by the Alippi approximation for different bits representation

Analyzing the results presented in fig.6, can concludes that a (32,16) or (16,8) representation induces a maximum of 1.89 %, (8,4) representation produces an error of 2.77 % and for a (5,3) representation, the maximum error is of 11.83%.

In order to evidence the total equivalent gates used function of number of bits allocated for hardware implementation of the Alippi function, an ISE report about hardware implementation was made, presented in table IV.

D. PLAN Approximation

The PLAN approximation (Piecewise Linear Approximation of a Nonlinear function) was proposed by Amin, Curtis and Hayes–Gill [4]. The PLAN approximation uses digital gates to compounds the sigmoid function and its equations are presented in table V. The advantage of this approximation is giving by the fact that instead of multiplication, shifting operations are required.

TABLE IV. RESOURCES UTILIZATION OF THE 4VSX35 FPGA CIRCUIT FOR HARDWARE IMPLEMENTATION OF THE ALLIPI APPROXIMATION

Device Utilization Summary					
Logic Utilization	Used				Available
	(32,16)	(16,8)	(8,4)	(5,3)	
Slices	127	67	36	15	15,360
LUTs	218	110	56	24	30,720
BRAMs	0	0	0	0	192
DSPs	0	0	0	0	192
Total equivalent gate count for design	1812	912	456	219	3.5M

TABLE V. PLAN APPROXIMATION EQUATIONS

PLAN(X)	**Conditions**
1	$\|X\| \ge 5$
$0{,}03125 \cdot \|X\| + 0{,}84375$	$2{,}375 \le \|X\| < 5$
$0{,}0125 \cdot \|X\| + 0{,}625$	$1 \le \|X\| < 2.375$
$0{,}25 \cdot \|X\| + 0{,}5$	$0 \le \|X\| < 1$

The PLAN approximation and its introduced errors by representation on different number of bits are shown in

fig.7 and fig. 8. In order to calculate these errors a hardware implementation of the PLAN equations was made by means of System Generator blocks of the Matlab/Simulink environment, fig.9.

Analyzing the results presented in fig.8, can concludes that (32,16) representation induces a maximum error of 1.89 % and a mean error of 0.81%, (16,8) representation has a maximum error of 2.14 % and a mean error of 0.90%, (8,4) representation has a maximum error of 10.75 % and a mean error of 5.06% and (5,3) representation has a maximum of 24.33 % and a mean error of 15.09%.

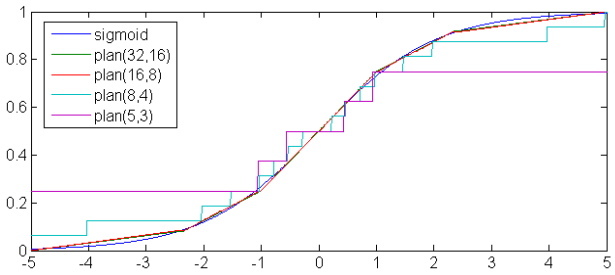

Figure 7. Comparative representation of the sigmoid function and PLAN approximation

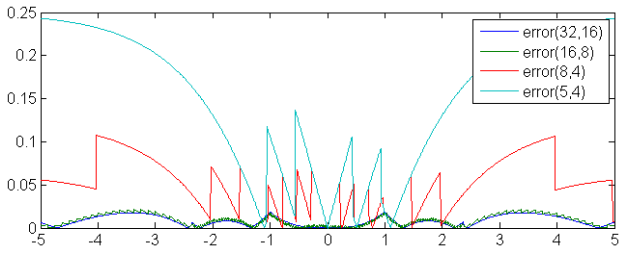

Figure 8. Errors introduced by the PLAN approximation for different bits representation

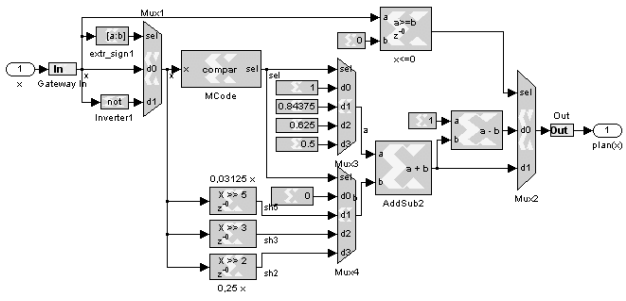

Figure 9. Hardware architecture of the Alippi function approximation

In table VI are presented the resources utilized in function of numerical representation for hardware implementation of the PLAN approximation.

TABLE VI. RESOURCES UTILIZATION OF THE 4VSX35 FPGA CIRCUIT FOR HARDWARE IMPLEMENTATION OF THE PLAN APPROXIMATION

Device Utilization Summary					
Logic Utilization	Used				Available
	(32,16)	(16,8)	(8,4)	(5,3)	
Slices	109	44	24	11	15,360
LUTs	138	67	32	11	30,720
BRAMs	0	0	0	0	192
DSPs	0	0	0	0	192
Total equivalent gates count for design	1164	561	270	114	3.5M

E. Piecewise second-order approximation

The sigmoid function can also be implemented as a piecewise second-order approximation. This kind of approximation implies using of generic square functions that involve using of the multiplications blocks. In order to avoid this, Zhang, Vassiliadis and Delgado–Frias have presented a second-order approximation scheme defined in the interval (-4, 4) that requires only one multiplier, (Zhang approximation), [5]. There are reported others second-order approximation, but the hardware resources involved for implementation are three times higher, [8].

$$y = \begin{cases} \frac{1}{2}\left(\frac{x}{2^2}-1\right)^2 & \text{for } -4 > x < 0 \\ 1-\frac{1}{2}\left(\frac{x}{2^2}+1\right)^2 & \text{for } 4 > x \geq 0 \end{cases} \quad (5)$$

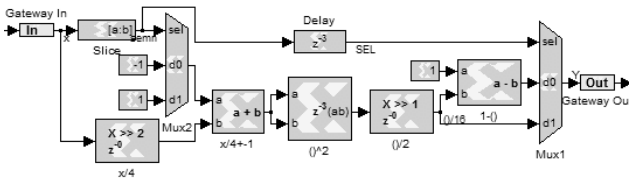

Figure 10. Hardware architecture of the Zhang function approximation

Analyzing the errors introduced by the Zhang approximation

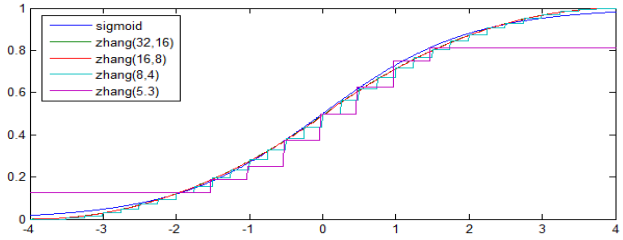

Figure 11. Comparative representation of the sigmoid function and Zhang approximation

The hardware implementation of the Zhang approximation is shown in fig. 10

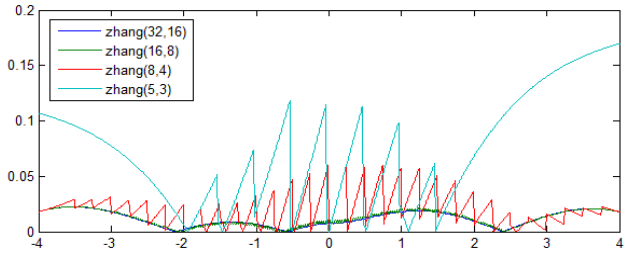

Figure 12. Errors introduced by the Zhang approximation for different bits representation

presented in fig. 11 and 12 it can be saw that the maximum error induced in function of the used numerical representation are 16.95% for (5,3), 6% for (8,4), 2.27% for (16,8) and 2.24% for (32,16), where the first number in the parenthesis represents the number of bits and the second is the binary point. The mean of absolute errors is distributed as follow: 7.21% for (5,3), 2.03% for (8,4), 1.21% for (16,8) and 1.18% for (32,16).

The resources utilized for hardware implementation of the Zhang function are shown in table VII.

TABLE VII. HARDWARE IMPLEMENTATION OF THE ZHANG APPROXIMATION

Device Utilization Summary					
Logic Utilization	Used				Available
	(32,16)	(16,8)	(8,4)	(5,3)	
Slices	93	29	18	10	15,360
LUTs	86	46	26	14	30,720
DSPs	1	1	1	1	192
Total equivalent gates count for design	1,169	513	309	201	3.5M

III CONCLUSIONS

In this paper are investigated different approximations functions reported in literature, from point of view of hardware resources utilization and induced errors. The hardware design is made in System Generator from the Simulink Matlab environment and the utilized resources reports are made in ISE environment.

Analyzing the report of the introduced errors vs. utilized resources, we get the conclusion that the best approximation method is the PLAN function in the case in which the number of the artificial neurons hardware implemented that use sigmoid function as fire function is larger than the number of the BRAM blocks available in the FPGA circuit. In the case in which the number of artificial neurons is lower than the total BRAM blocks available in the FPGA circuit the best way to approximate the sigmoid function is Lookup Tables method.

The results of hardware implementations of the considered approximation functions are shown in table VIII

TABLE VIII. ERRORS AND RESOURCES UTILIZATION OF THE 4VSX35 FPGA CIRCUIT FOR HARDWARE IMPLEMENTATION OF THE SIGMOID APPROXIMATION

Approximation function	Maximum error (%)	Mean error (%)	Total equivalent gates count for design
Lookup Table	0	0	131.072
A-low	5.63	0.63	411
Allipi	1.89	1.11	877
Plan	1.89	0.63	351
Zhang	2.16	1.10	314

All the approximation functions were grouped into a library in Simulink/System Generator environment and used as fired function for the neurons of an artificial neural network hardware implemented.

The neuron was designed to suit the calculus algorithms of the learning phase, in which the weights are calculated and stored in the RAMW memory block, and the propagation phase, in which the fired output is determined.

The blocks that compound the neuron are: a MAC unit, a multiplexer with two inputs, a memory block and the firing block.

The architecture of the neuron is presented in figure 13.

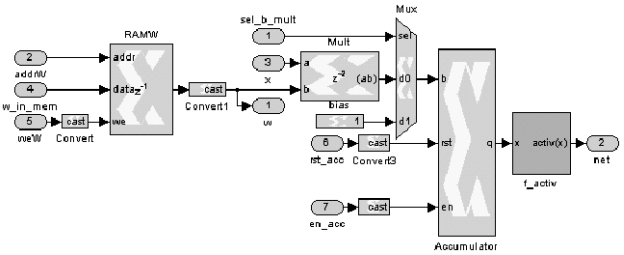

Figure 13. Architecture of the neuron

The configuration of the RAM memory used for the weights storage consists of weights initialization and the setting of the number of bits used for data processing and representation. For that, a pop-up parameterization window is used, figure 14.

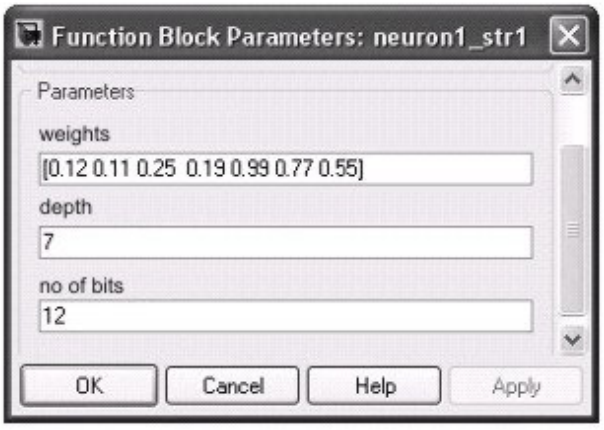

Figure 14. Neuron parameterisation window

In order to underline the main characteristics of the developed firing blocks, such as the hardware resources utilization, processing frequency and power consumption,

TABLE IX. RESOURCE DISTRIBUTIONS FOR HARDWARE IMPLEMENTATION OF ARTIFICIAL NEURON WITH DIFFERENT FIRING FUNCTION

Resources distribution	Neuron lookup_table	Neuron f_zhang	Neuron f_alippi	Neuron f_Alow	Neuron f_plan	Available resources 4VSX35
Slices	5	28	59	29	12	15.360
LUTs	0	25	89	31	10	30.720
RAMBs	2	1	1	1	1	192
DSPs	1	2	1	1	1	192
Total equivalent gates count for design	131.120	65.898	66.418	65.911	65.680	3.5 M
Max frequency (MHz)	227.790	255.860	290.613	268.168	234.467	-
Estimated power consumed	609	608	607	607.02	608	-

hardware implementations of neurons with different firing approximation function were made. The FPGA circuit used for implementation was 4VSX35.

The reports are presented in table IX.

The maximum frequency supported by the hardware implementation of the approximations reported in the literature [3 - 6] shows much smaller performances comparative with the same approximation but hardware implemented with the proposed method described in this paper.

TABLE X. CLOCK RATE OF SIGMOID FUNCTION APPROXIMATIONS REPORTED IN LITERATURE [7,9]

Approximation	Max frequency (MHz)
A-law based approximation	36
Alippi and Storti–Gajani Approximation	36
PLAN approximation	39
Zhang approximation	176

The results obtained in this paper are means to be guidance tool in selecting and using of most appropriate approximation function of the sigmoid function.

The design of the all firing function considered is made by the authors in Simulink/System Generator environment in order to create a library of components used to develop neurons and neuron networks with different topologies. Finally, the library of the neuronal components will represent a useful instrument for an easier designing of a neural network system.

REFERENCES

[1] Cirstea, M., et all.: *Neural and Fuzzy Logic Control of Drives and Power Systems,* Newnes, pp 77-112, 2002.

[2] E. Won, "A hardware implementation of artificial neural networks using field programmable gate arrays", *Nuclear Instruments and Methods in Physics Research,* 581, pp 816–820, 2007.

[3] Alippi, C., and Storti–Gajani, G.: "Simple approximation of sigmoidal functions: realistic design of digital neural networks capable of learning". *Proc. IEEE Int. Symp. on Circuits and Systems*, Singapore, pp. 1505–1508, 1991.

[4] Amin, H., et all: "Piecewise linear approximation applied to nonlinear function of a neural network", *IEE Proc. Circuits, Devices Sys*., 144, (6), pp. 313–317J, 1997.

[5] Zhang, M., et all.: "Sigmoid generators for neural computing using piecewise approximations", *IEEE Trans. Comput.,* 45, (9), pp. 1045–1049, 1996.

[6] M.T. Tommiska: "Efficient digital implementation of the sigmoid function for reprogrammable logic", *IEE Proceedings – Computers and Digital Techniques* 150, number 6, pp. 403-411, 2003.

[7] Basterretxea, K., et all.: "Digital design of sigmoid approximator for artificial neural networks", *Electron. Letters*, 38, (1), pp. 35–37, 2002.

[8] N. Nedjah, et all.: "Dynamic MAC-based architecture of artificial neural networks suitable for hardware implementation on FPGAs", *Neurocomputing* 72(10-12), pp 2171-2179, 2009.

[9] Holt, J.L., and Hwang, J-N.: "Finite precision error analysis of neural network hardware implementations", *IEEE Trans. Comput*, 42, (3), pp. 281–290, 1993.