FPGA Implementation of Range Addressable Activation Function for Lattice-Ladder Neuron

Tomyslav Sledevic[1], Dalius Navakauskas[1]
[1]*Department of Electronic Systems, Vilnius Gediminas Technical University, Naugarduko St. 41–422, LT-03227 Vilnius, Lithuania*
tomyslav.sledevic@vgtu.lt

***Abstract*—FPGA implementation of hyperbolic tangent activation function for multilayer perceptron structure seems attractive; however, there is a lack of preliminary results on the choice of memory size particularly, when LUT of the function is stored in dedicated on-chip block RAM. The aim of this investigation was to get insights on the distortions of the selected neuron model output by the evaluation of transfer function RMS error and neuron output signal mean and maximum errors while changing the gain and memory size of the activation function. Thus, the range addressable activation function for the second order normalized lattice-ladder neuron was implemented in Artix-7 FPGA. Various gain and memory constrains were investigated. The increase of LUT memory size and gain yielded smaller error of output signal and nonlinear influence on the transfer function. 2 kB of BRAM is sufficient to achieve tolerable less than 0.4 % maximum error utilizing only 0.36 % of total on-chip block memory.**

***Index Terms*—Lattice-ladder neuron;, nonlinear activation function; transfer function; high-level synthesis; fixed-point arithmetic; FPGA implementation.**

I. INTRODUCTION

Artificial neural network implementation in FPGA is attractive because of the hardware parallel and periodical structure, fast reconfigurability, hundreds of dedicated DSP and memory slices, convenient high-level synthesis tools [1]. The basic building blocks that are necessary for FPGA implementation of artificial neuron are adder, multiplier and nonlinear function that is hardware unfavourable.

The neuron activation functions such as sigmoid [2], logarithmic sigmoid [3] or hyperbolic tangent [4] are mostly used in the artificial neural networks. Such functions have easily obtainable derivative, which is important for training process, because that decreases the computational load. However, the precise implementation of the nonlinearity in FPGA gives cause for concern. The reasonable solutions while solving this issue can be combinational [5], [6], piecewise linear (PWL) [7], [8] or quadratic (PWQ) approximations [2], [9], look-up tables (LUT) synthesized in a logic [10], [11] or stored in on-chip memory [4]. The straightforward implementation of nonlinear function in hardware is not a correct approach, because both exponentiation and division operations are logic and arithmetic resource hungry [5], [11]. CORDIC algorithm introduces latency and has limited input domain [8].

Efficient FPGA implementation of activation function is a multi-criteria task. The balancing of the accuracy, resources and processing speed must be considered [8]. High precision implementation requires more resources. Signal routing through high amount of logic is a bottleneck to achieve higher clock frequency for the synthesized design, due to the growing delay in a critical path of not compact hardware. Moreover, resource reduction makes circuit faster, but also influences inaccuracy in approximated activation function.

When only few bits are used as the input of activation function, then it makes sense to use combinational approximation based on direct bit level mapping without arithmetic operators. It is shown in [5] that with 6 bits precision the maximal absolute error of the activation function is less than 1 %.

The polynomial approximation methods are based on the function input range division into equal parts, where each function subinterval is approximated by line or curve [8]. The PWL approximation of hyperbolic tangent with 256 linear sections provides a maximum error of 0.002 % using 32 bit arithmetic [7]. The controlled accuracy second order approximation of sigmoid function was implemented on single DSP slice with maximum allowable 1 % error [9]. The amount of additional logic resources depends on the required word-length and error. The PWQ approximation can be implemented by reusing already utilized multiplier and adder units in neuron block proposed in [2]. The main disadvantage of polynomial approximation technique is the increased latency due to the growing number of multiplication operations for the higher-order polynomial. The maximum allowable 2 % error with 9 bit input is achieved using hybrid PWL and LUT methods in [12] implementing hyperbolic tangent function in hardware.

The LUT-based approach works much faster than polynomial approximation, since LUT consumes memory. Small LUT is usually implemented in distributed memory, which does not require delay units, but has a limited size [4]. Newest FPGA chips have at least 0.5 MB of on-chip block RAM (BRAM) with one clock cycle latency to access stored data [13]. Therefore, large LUT is generally stored in BRAM. The accuracy of approximated hyperbolic tangent under various precision input signal and LUT size was investigated in [11]. Depending on the application and required accuracy for the activation function, different sizes

Manuscript received 5 January, 2015; accepted September 12, 2015.

LUTs (from 28 samples [10] to 215 samples [4]) are used.

This paper presents FPGA implementation of range addressable LUT approximation of hyperbolic tangent function. Using nowadays FPGA there is not much concern about memory, therefore LUT stored in BRAM is preferred in our implementation. In Section II the implementation of lattice-ladder neuron (LLN) and its activation function are presented. In Section III the error estimation parameters are described in details. In Section IV results on experimental investigation of created LLN working under different LUT size and output gain are summarized. General conclusions are stated in Section V.

II. Implementation of Lattice-Ladder Neuron and Its Nonlinear Activation Function

The hardware implementation of lattice-ladder neuron [14] with on-chip training circuit and linear activation function was presented in [15]. The normalized lattice outperforms other IIR structures due to the stability. However, it requires four multiplication operations in each section and 11 multiplications in total for the second order LLN. The output of the second order LLN can be expressed

$$s_{\text{out}}(n) = \Phi\left\{ g \sum_{j=0}^{2} v_j b_j(n) \right\}, \tag{1}$$

when local flow of information in the lattice part for all j sections (see Fig. 1) is defined by:

$$\begin{bmatrix} f_{j-1}(n) \\ b_j(n) \end{bmatrix} = \begin{bmatrix} \cos\Theta_j & -\sin\Theta_j \\ \sin\Theta_j & \cos\Theta_j \end{bmatrix} \begin{bmatrix} f_j(n) \\ z^{-1} b_{j-1}(n) \end{bmatrix}, \tag{2}$$

with initial and boundary conditions $b_0(n) = f_0(n)$ and $f_2(n) = s_{\text{in}}(n)$, where $s_{\text{in}}(n)$ – input signal; v_j – weight of the ladder; Θ_j – rotation angle of the lattice; $f_j(n)$ and $b_j(n)$ – forward and backward signals of the lattice correspondingly; $\Phi\{\cdot\}$ – hyperbolic tangent as neuron activation function; g – amplification coefficient.

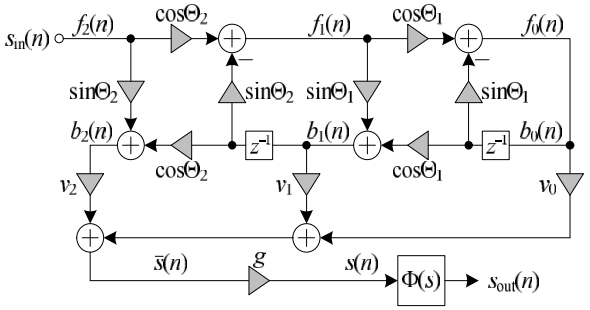

Fig. 1. The structure of the second order lattice-ladder neuron.

The hyperbolic tangent function is defined as follows

$$\Phi\left(s(n)\right) = \frac{e^{2.03 g\bar{s}(n)} - 1}{e^{2.03 g\bar{s}(n)} + 1}, \tag{3}$$

where $s(n) = g\bar{s}(n)$ is the input of activation function.

The proposed neuron activation function is divided in three regions (see Fig. 2): pass, processing and saturation. In a pass region $\Phi \equiv s$, the signal $s(n)$ is directly transferred to the output $s_{\text{out}}(n)$. The processing region of the function $\Phi(s)$ is implemented in LUT and stored in BRAM. In a saturation region, the LLN output is always $\Phi \equiv 1$.

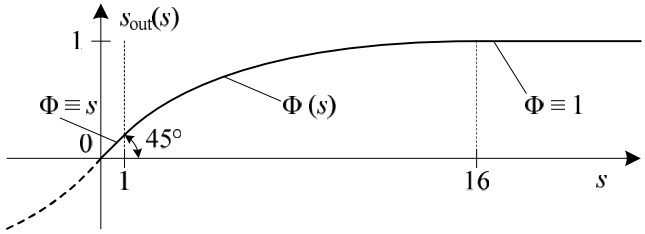

Fig. 2. The plot of the proposed activation function.

To make the smoothed junction between the pass and processing regions, the exponent value is set to 2.03 instead of original 2. The approximated LLN output can be expressed

$$s_{\text{out}}(n) = \begin{cases} 1, & s(n) \geq 16, \\ \Phi\left(s(n)\right), & 16 < s(n) < 1, \\ s(n), & -1 \leq s(n) \leq 1, \\ \Phi\left(s(n)\right), & -16 < s(n) - 1, \\ -1, & s(n) \leq -16. \end{cases} \tag{4}$$

The activation function output of negative value is obtained using asymmetry principle $\Phi\left(-s(n)\right) = -\Phi\left(s(n)\right)$. The amplification coefficient g controls the output range of the hyperbolic tangent. If $g = 1$, then activation function is linear. For $g > 1$, the nonlinearity in the LLN output grows. The higher gain is the wider range of LUT is accessible. If $g = 16$, then $s_{\text{out}}(n)$ varies in range [–1, 1]. The amplification of $\bar{s}(n)$ more than 16 times is equivalent to the squeeze of activation function or similarly to increase of the slope of hyperbolic tangent for $s(n)$ in range [–1, 1].

Program *Matlab* is used to create a reference design working on floating-point data. Program *Vivado high-level synthesis* (HLS) from *Xilinx* is used to create, simulate and analyse constrained design. After the simulations of floating-point and fixed-point designs, the corresponding output signals $s_{\text{out}}(n)$ and $\hat{s}_{\text{out}}(n)$ are compared to determine the LLN transfer function and the output signal discrepancies.

III. Evaluation Criteria

The accuracy of the hyperbolic tangent approximation is evaluated using average and maximum error with variable m size memory and g amplification of signal $\bar{s}(n)$ [8], [15]:

$$\varepsilon_{\text{avg}}\left(g, m\right) = \frac{1}{N} \sum_{n=0}^{N-1} \left| s_{\text{out}}(n) - \hat{s}_{\text{out}\,g,m}(n) \right|, \tag{5}$$

$$\varepsilon_{\max}\left(m\right) = \max \left| s_{\text{out}}(n) - \hat{s}_{\text{out}}(n) \right|. \tag{6}$$

The description in the literature of limits of the accuracy of activation function is fragmented and it lacks commonly acceptable values. The ε_{avg} and $\varepsilon_{\max}$ values usually depend on the precision requirements for the application and in many cases maximum error 0.4 % [15], 1.5 % [8] or even 2 %

[6], [12] is tolerable.

The lattice-ladder can be set to work as nonlinear low, high, band-pass or band-stop filter with additional gain control (see Fig. 1). Such a system has the transfer function, which will have distortions dependent on the limited LUT size dedicated to the hyperbolic tangent function. To check the accuracy of the LLN transfer function, the LLN must be scanned by $s_{\text{in}}(n)$ signal, which contains all the frequency components in range $[0, f_S/2]$, where f_S is signal sampling frequency. The frequency of $s_{\text{in}}(n)$ must linearly change in time. This property has the chirp signal $s_{\text{chirp}}(n)$

$$s_{\text{chirp}}(n) = \cos\left(2\pi f_i(n) + \phi_0\right), \tag{7}$$

with instantaneous frequency sweep function expressed by

$$f_i(n) = f_0 + \frac{f_1 - f_0}{n_1} n, \tag{8}$$

where f_0 and f_1 are desired starting and breakpoint frequencies at time $n = 0$ and $n = n_1$; $\phi_0 = 0$ – signal initial phase.

To check the worst case of LLN transfer function implementation, the parameters $\Theta_2 = \Theta_1 = v_2 = v_1 = 0$ and $v_0 = 1$ are set to pass through all frequency components to the input of activation function. The transfer function of LLN activation function is obtained by the estimator [16]

$$T(f) = \frac{P_{\text{cross}}(f)}{P_{\text{auto}}(f)}, \tag{9}$$

where $P_{\text{cross}}(f)$ – cross power spectral density of $s_{\text{in}}(n)$ and $s_{\text{out}}(n)$; $P_{\text{auto}}(f)$ – auto power spectral density of $s_{\text{in}}(n)$.

The distortions between reference and proposed LLN transfer functions are evaluated taking root mean square (RMS) metric ε_T [11], [17]

$$\varepsilon_T(g,m) = \sqrt{\frac{2}{f_S} \sum_{f=0}^{f_S/2} \left| T_g(f) - \hat{T}_{g,m}(f) \right|^2}, \tag{10}$$

where $T_g(f)$, $\hat{T}_{g,m}(f)$ – floating-point and fixed-point transfer functions.

IV. Results of Experimental Investigation

We are interested in compact and enough precise hyperbolic tangent activation function for lattice-ladder neuron implementation. Therefore, the transfer function and output signal discrepancies are measured under different BRAM size. During experiments BRAM size is increased 2 times in each step from 64 B to 64 kB. The gain is increased linearly from 2 to 16. Other LLN parameters are kept unchanged. Independent on the BRAM size two bytes precision is used for each sample of the approximated hyperbolic tangent. The average and maximum errors (ε_{avg} and ε_{max}) are obtained by uniformly sampling $s_{\text{in}}(n)$ on 10^6 equally spaced points in the range of [–1, 1].

The average and maximum errors of the activation function output depends almost linearly on the LUT size (Table I). In comparison with the [8], the ε_{max} less than 1.5 % can be achieved using 512 B LUT size or only 0.09 % of total BRAM in *xc7z020* FPGA chip [13] used here for experimental investigation. Maximum error less than 0.4 % [5] can be achieved using 2 kB of BRAM (one BRAM block from 280 available in *xc7z020*) utilizing only 0.36 % of total block memory in FPGA chip. The large LUT ensures small error, *e.g.*, 64 kB memory (see Fig. 3) yields $\varepsilon_{\text{max}} = 0.01$ % and $\varepsilon_{\text{avg}} = 0.002$ %, however utilizes 11.4 % of available memory. Such high precision is very "expensive" as for single LLN, however is useful for big neural networks with shared activation function.

TABLE I. THE HYPERBOLIC TANGENT FUNCTION IMPLEMENTATION ERRORS.

BRAM size, B	128	256	512	1k	2k	4k
ε_{avg}, %	0.54	0.27	0.13	0.06	0.03	0.02
ε_{max}, %	5.48	2.74	1.39	0.70	0.35	0.16

The maximum ε_{avg} is observed at $g = 3$ for all tested memory sizes (Fig. 3). The rising gain allows to access wider range of LUT addresses, therefore decreases the quantization error and ε_{avg} falls slowly.

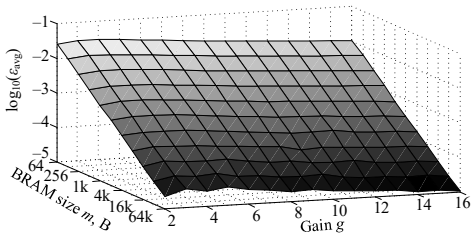

Fig. 3. The average error of the LLN output, while changing gain and memory size for approximated hyperbolic tangent function.

The chirp signal with sampling frequency $f_S = 11025$ Hz and duration $n_1 = 10$ s is used for the transfer function error estimation experiments. The results of transfer function distortions $\varepsilon_T(g, m)$ under two margin gains and three different filter bandwidths $f_{B1} = 5512$ Hz, $f_{B2} = 2756$ Hz, $f_{B3} = 200$ Hz, are presented in Fig. 4, as a dependence on BRAM size. Decrease of the bandwidth will reduce the gap between ε_T limits. The ε_T for LLN with f_{B1} decreases almost linearly when memory size grows. The lower $\varepsilon_T = 1.2 \cdot 10^{-4}$ and upper $\varepsilon_T = 1.8 \cdot 10^{-3}$ limits belong to the LLN with widest bandwidth using 2 kB BRAM for activation function.

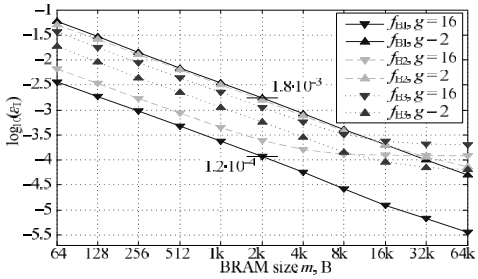

Fig. 4. RMS errors of the LLN transfer function using different size LUTs.

When LLN is set to pass through all frequencies $f_{B1} = 5512$ Hz, then the error ε_T decreases almost linearly increasing both gain and memory size (Fig. 5). The gain set to 2 yields highest error for a given memory size, because at $g = 2$ with the $s(n)$ signal only 1/16 of the approximated

hyperbolic tangent can be accessed. Setting the LLN to work in narrower frequency band f_{B3} increases the error ε_T for activation function with higher gain and vice versa, smaller gain (less nonlinearity in the activation function) decreases the ε_T. The increase of BRAM size more than 2 kB nonlinearly improves the error of transfer function. Therefore, 2 kB BRAM is identified as sufficient memory size for activation function implementation as a trade-off between resources and precision.

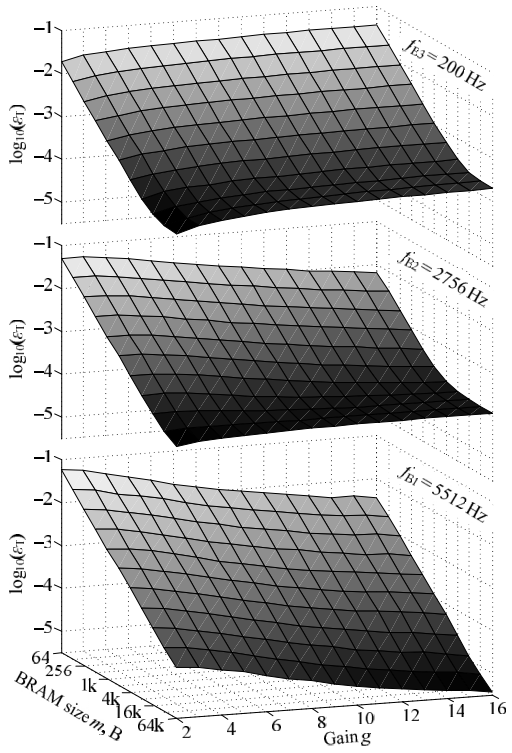

Fig. 5. The RMS error of the LLN transfer function, while changing gain and memory size for approximated hyperbolic tangent function.

The LLN implemented in HLS tool was translated to the low-level model described in VHDL for further synthesis and uploading to the FPGA. The Place and Route tool shows that, in the generated LLN circuit the hyperbolic tangent samples can be accessed with frequency f_{max} = 312 MHz, while f_{max} > 300 MHz is considered as a high maximum frequency achievable in a practical FPGA systems [18].

V. CONCLUSIONS

The range addressable hyperbolic tangent activation function for second order lattice-ladder neuron in a fixed-point arithmetic is successfully implemented on *xc7z020* FPGA chip.

Carried out experimental investigation affirms that:

1. BRAM size of 2 kB is sufficient to achieve tolerable less than 0.4 % maximum error of the approximated activation function output with small RMS error $\varepsilon_T = 1.8 \cdot 10^{-3}$ of the lattice-ladder neuron transfer function.
2. The narrower band lattice-ladder neuron has, the larger error ε_T is, when operating in whole range of hyperbolic tangent activation function.
3. Relatively high frequency f_{max} = 312 MHz is achieved for activation function LUT access in a final synthesized circuit.

REFERENCES

[1] J. Misra, I. Saha, "Artificial neural networks in hardware: A survey of two decades of progress", *Neurocomputing*, vol. 74, pp. 239–255, 2010. [Online]. Available: http://dx.doi.org/10.1016/j.neucom.2010.03.021

[2] V. P. Nambiar, M. Khalil-Hani, R. Sahnoun, M. N. Marsono, "Hardware implementation of evolvable block-based neural Networks utilizing a cost efficient sigmoid-like activation function", *Neurocomputing*, vol. 140, pp. 228–241, 2014. [Online]. Available: http://dx.doi.org/10.1016/j.neucom.2014.03.018

[3] V. Vaitkus, G. Zylius, R. Maskeliunas, "Electrical spare parts demand forecasting", *Elektronika ir Elektrotechnika*, vol. 20, no. 10, 2014, pp. 7–10. [Online]. Available: http://dx.doi.org/10.5755/j01.eee.20.10.8870

[4] M. Bahoura, "FPGA Implementation of high-speed neural network for power amplifier behavioral modeling", *Analog Integrated Circuits and Signal Processing*, vol. 79, no. 3, pp. 507–527, 2014. [Online]. Available: http://dx.doi.org/10.1007/s10470-014-0263-7

[5] M. T. Tommiska, "Efficient digital implementation of the sigmoid function for reprogrammable logic", *IEE Proc. – Computers and Digital Techniques*, vol. 150, no. 6, pp. 403–411, 2003. [Online]. Available: http://dx.doi.org/10.1049/ip-cdt:20030965

[6] B. Zamanlooy, M. Mirhassani, "Efficient VLSI implementation of neural networks with hyperbolic tangent activation function", *IEEE Trans. Very Large Scale Integration (VLSI) Systems*, vol. 22, no. 1, pp. 39–48, 2014.

[7] P. Ferreira, P. Ribeiro, A. Antunes, F. M. Dias, "A high bit resolution FPGA implementation of a FNN with a new algorithm for the activation function", *Neurocomputing*, vol. 71, pp. 71–77, 2007. [Online]. Available: http://dx.doi.org/10.1016/j.neucom.2006.11.028

[8] A. Armato, L. Fanucci, E. P. Scilingo, D. De Rossi, "Low-error digital hardware implementation of artificial neuron activation functions and their derivative", *Microprocessors and Microsystems*, vol. 35, no. 6, pp. 557–567, 2011. [Online]. Available: http://dx.doi.org/10.1016/j.micpro.2011.05.007

[9] I. D. Campo, R. Finker, J. Echanobe, K. Basterretxea, "Controlled accuracy approximation of sigmoid Function for efficient FPGA-based implementation of artificial neurons", *Electronics Letters*, vol. 49, no. 25, pp. 1598–1600, 2013. [Online]. Available: http://dx.doi.org/10.1049/el.2013.3098

[10] U. Lotric, P. Bulic, "Applicability of approximate multipliers in hardware neural networks", *Neurocomputing*, vol. 96, pp. 57–65, 2012. [Online]. Available: http://dx.doi.org/10.1016/j.neucom.2011.09.039

[11] A. Gomperts, A. Ukil, F. Zurfluh, "Development and implementation of parameterized FPGA-based general purpose neural networks for online applications", *IEEE Trans. Industrial Informatics*, vol. 7, no. 1, pp. 78–89, 2011. [Online]. Available: http://dx.doi.org/10.1109/TII.2010.2085006

[12] P. K. Meher, "An optimized lookup-table for the evaluation of sigmoid function for artificial neural networks", in *18th IEEE/IFIP Int. Conf. VLSI System on Chip*, 2010, pp. 91–95.

[13] *Zynq-7000 all programmable SoC overview*, Xilinx Inc., San Jose, CA, 2014.

[14] D. Navakauskas, "A Reduced size lattice-ladder neural network", in *Proc. IEEE Signal Processing Society Workshop on Neural Networks for Signal Processing VIII*, 1998, pp. 313–322. [Online]. Available: http://dx.doi.org/10.1109/nnsp.1998.710661

[15] T. Sledevic, D. Navakauskas, "The lattice-ladder neuron and its training circuit implementation in FPGA", in *Proc. IEEE 2nd Workshop on Advances in Information, Electronic and Electrical Engineering*, 2014, pp. 1–4. [Online]. Available: http://dx.doi.org/10.1109/aieee.2014.7020327

[16] P. M. T. Broersen, "A comparison of transfer function estimators", *IEEE Trans. Instrumentation and Measurement*, vol. 44, no. 3, pp. 657–661, 1995. [Online]. Available: http://dx.doi.org/10.1109/19.387302

[17] L. Stasionis, A. Serackis, "Burst signal detector based on signal energy and standard deviation", *Elektronika ir Elektrotechnika*, vol. 20, no. 2, pp. 48–51, 2014. [Online]. Available: http://dx.doi.org/10.5755/j01.eee.20.2.6384

[18] G. Dessouky, M. J. Klaiber, D. G. Bailey, S. Simon, "Adaptive dynamic on-chip memory management for FPGA-based reconfigurable architectures", in *Proc. 24th Int. Conf. Field Programmable Logic and Applications*, 2014, pp. 1–8. [Online]. Available: http://dx.doi.org/10.1109/fpl.2014.6927471