ISSN 2278-3091
Volume 9, No.3, May - June 2020
International Journal of Advanced Trends in Computer Science and Engineering
Available Online at http://www.warse.org/IJATCSE/static/pdf/file/ijatcse10932020.pdf
https://doi.org/10.30534/ijatcse/2020/10932020

A Real Time Hardware Co-Simulation of Linearization of Nonlinear Sensor using ANFIS Linearizer

Rajesh T. Jadhav [1], Dr.Vikram S. Patil [2]
[1] Research Scholar, Shivaji University, Kolhapur, Maharashtra, India, rajeshtukaramjadhav@gmail.com
[2] Director, ADCET, Ashta, & Research Guide, Shivaji University, Kolhapur, Maharashtra, India, vikrams.patil@gmail.com

ABSTRACT

Due to time to market constraints, the application of Xilinx System Generator (XSG) and Matlab-Simulink which are consider as one of the rapid prototyping tools, has become increasingly important. In association with Field Programmable Gate Array (FPGA) devices, hardware co-simulation allows designers for evaluation, testing and validation of a new algorithm, a new component or a new prototype without damaging the actual system. This paper presents a new methodology for implementing linearization of nonlinear sensor for temperature measurements, using a real-time hardware co-simulation. This methodology helps to improve the design verification efficiency for such a system. The present paper demonstrate the application of XSG tool to verify a new linearizer to linearize a nonlinear sensor such as NTC Thermistor based on Adaptive Neuro Fuzzy Inference System (ANFIS). ANFIS based inverse modeling technique is used for linearization. In the present paper, an intensive study of the effect of the type, number of membership functions and training method on the training error has been reported. The number of epochs needed to linearize the sensor is also presented to bring out the convergence time of the technique. The presented results may be of great importance to design engineers.

Key words : ANFIS, Hardware Co-Simulation, FPGA, Sensor linearization, XSG.

1. INTRODUCTION

Sensors are the fundamental elements which are used in most of the measurement circuits to monitor the physical quantity such as temperature, pressure, etc or to give feedback signals to the control unit. Sensors having low cost, better resolution, higher sensitivity and linear characteristics are required for industrial applications [1]. Generally Sensors gives analog output, which may sometimes shows nonlinear behaviour. This is due to natural nonlinear characteristic of sensor itself, environment's dynamic nature, sensor's inherent noise, aging and data loss due to transients or intermittent faults [2]. It is essential to have linear characteristics of the sensor as it will improve the system performance [3]. Linearization of this nonlinear behaviour of sensors has always been a designed challenge. Linearization of nonlinear sensor in the digital environment is a vital step in the instrument signal conditioning process [4].

Linearization of non-linear sensors characteristics is often a complex and difficult task. It is for this reason, neuro-fuzzy systems are getting importance in the field of soft computing. This is because of their learning, intrinsic parallelism and adaptation capabilities. Results revealed that this artificial intelligence technique utilizing the neuro-fuzzy system, optimizes the objective function [5]. Two main systems such as Mamdani-Zadeh (MZ) and Takagi–Sugeno–Kang (TSK) models have been developed [6], [7]. Among the two models, TSK system is consider as more universal and robust with many modifications [8]-[10]. Neuro-Fuzzy system's modeling capability was demonstrated by S. N. Engin et al. [11]. TakagiSugeno [12] studied the complete algorithms to shape the combinations of both Neural Networks and differential equations and thereafter the named it as ANFIS (Adaptive Neuro Fuzzy Inference Systems) [13].With the help of data acquired from mathematical model, a non linear system can be very accurately modelized by ANIFS [14].

Over the past decades, the popularity of FPGA has rapidly increased in embedded systems against microprocessors, microcontrollers or DSP. This is due to their parallel processing and increased number of gates [15]. Now days, for the prototyping of FPGA devices, Matlab - Simulink environment is most commonly used as compared to traditional design tools. This is because Matlab - Simulink environment doesn't require the use of complicated Hardware Description Languages (HDLs). Fortunately, the advent of Xilinx's XSG toolbox for Matlab - Simulink graphical environment, has made possible the hardware co-simulation of the system along with generation and implementation of HDL code in FPGA devices [15]-[17].This result in the reduction of time required between hardware implementation

and control design derivations. Within this environment, the software also provides hardware co-simulation [18], [19]. As compared to HDL based approach, this methodology provides easy verification and implementation of hardware. With respect to other methodologies, Matlab-Simulink simulation along with hardware co-simulation provides a more cost efficient solution [14]. The circuit is realized on Numato Lab's Waxwing Spartan 6 FPGA Development Board.

2. ACTUAL SYSTEM DESCRIPTION

The actual system block diagram is shown in Figure 1 and Figure 2 shows actual implemented system.

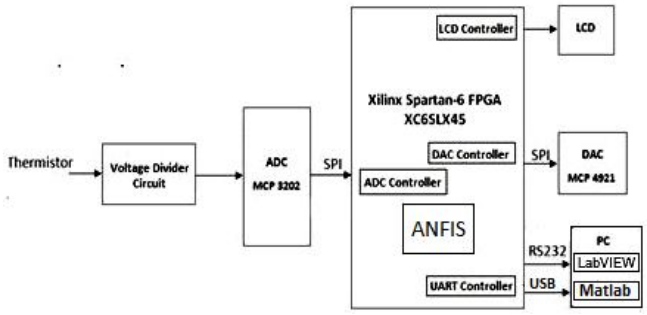

Figure 1: Actual System Block Diagram

Figure 2: Actual System

The NTC thermistor, 1K resistor and +5V voltage source are connected in series to form a voltage divider circuit whose schematic is shown in Figure 3.

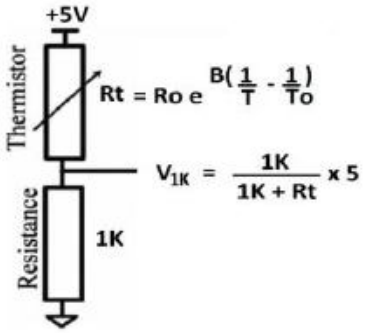

Figure 3: Schematic of voltage divider circuit

The thermistor used in a voltage divider circuit is a NTC Thermistor whose resistance R_T at temperature T can be modeled by

$$R_T = R_O \, e^{xp}\left[\beta\left(\frac{1}{T} - \frac{1}{T_O}\right)\right] \quad (1)$$

where the NTC thermistor used in this work has R_O = 10,000 ohms, is the resistance at a reference temperature T_O = 298 K (25 °C) and β = 3950, with a tolerance of ±10%. The non linear analog voltage across 1K resistor is given to the ADC (MCP 3202). Further the digital output of ADC is given to the digital device named Waxwing Spartan 6 FPGA (XC6SLX45) Development Board. An ANFIS used for sensor linearization, for temperature measurements, is implemented in FPGA through hardware co-simulation.

3.XSG DESIGN FLOW FOR FPGA IMPLEMENTATION

High level programming languages like C, Matlab-Simulink, etc are used for modeling the given system. XSG comes along with Matlab-Simulink software packet. XSG is a predefined block set which is used to implement system defining algorithms [18]. The accuracy of the algorithms can also be verified using these high level technical programming languages. Matlab-Simulink provides an interactive environment for analyzing data, developing algorithm, numerical computation and data visualization [19]. For embedded systems, Matlab - Simulink provides model based design and multidomain simulation environment. Matlab - Simulink also provides an extensive library of parameterizable functions, event driven simulator and interactive graphical environment that allows designing, simulation, implementation and testing of time varying systems [19].

In this application, XSG and Matlab-Simulink are used as a system level modeling tool (for facilitating FPGA hardware design) and a high level development tool respectively. It extends Matlab - Simulink by providing a hardware design, a well suited modeling environment. Xilinx's ISE tools are used for synthesizing the result to Xilinx FPGA technology. FPGA implementation steps which includes synthesizing, translating, mapping, placing and routing are automatically executed for generation of programming bit file for FPGA devices. Figure 4 shows the actual XSG design flow.

XSG automates the design process which includes debugging, implementation and verification of Xilinx's FPGA. XSG provides a system-level resource estimation, high-speed hardware description language co-simulation interface, and accelerated simulation through hardware in the loop

co-simulation interfaces that boost the simulation performance up to 1000x [21]. For the design of DSP FPGAs, XSG provides a system integration platform that brings together RTL components of a DSP system and Matlab – Simulink under one environment of simulation and implementation. XSG also provides black box block. With the help of black box block, Matlab - Simulink environment can import RTL and co-simulate it with the help of ModelSim or Xilinx ISE Simulator.

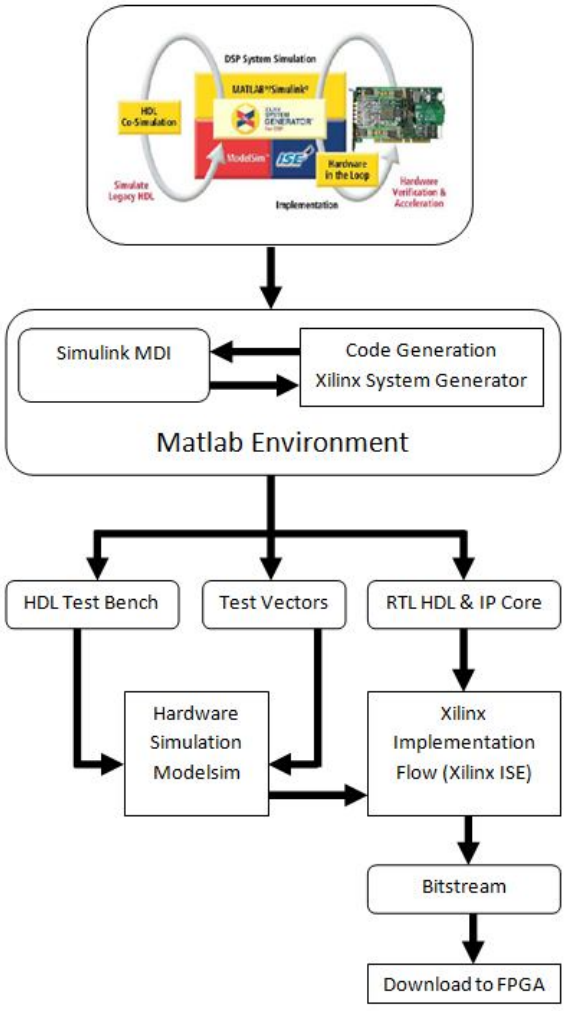

Figure 4: System Generator design flow [17]

4. ANFIS LINEARIZER MODELIZATION

4.1 Elements of ANFIS Architecture

As ANFIS is going to be hardware implemented, hence it is necessary to have a detailed knowledge of its architecture. Figure 5 shows the ANFIS architecture wherein the square nodes denotes the functions with parameters to be learnt whereas circular nodes represent fixed operations.

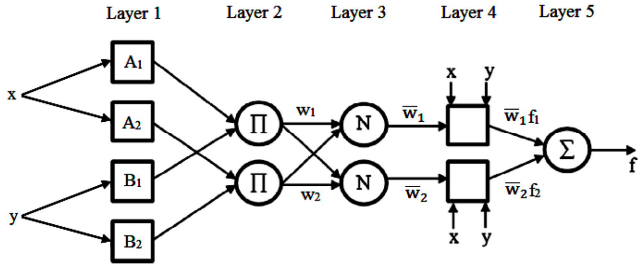

Figure 5: ANFIS Architecture

If x is A_1 and y is B_1 then according to Sugeno rule form $f_1 = p_1x + q_1y + r_1$
(2)

Here inputs x and y represents premise variables of the fuzzy rule. A_1, B_1 represents premise parameters and p_1, q_1 and r_1 represents consequence parameters. Using f_1 and f_2 functions and w_1 and w_2 weight values, the final output function F is given by (3).

$$F = \frac{w_1 f_1 + w_2 f_2}{w_1 + w_2} = \overline{w_1} f_1 + \overline{w_2} f_2 \tag{3}$$

With reference to Fig. 5, ANFIS Linearizer shows five layers [14].

Layer 1: Every node is adaptive in this layer. Here fuzzification process takes place. Output of each node is given by (4).

$$O_{1,i} = \mu_{A_i}(x) \quad for\ i = 1,2$$
$$O_{1,i} = \mu_{B_{i-2}}(x) \quad for\ i = 3,4 \tag{4}$$

Thus $O_{1,i}(x)$ represents membership grade for inputs x and y. The membership functions could be trapezoidal, triangular or any other type.

Layer 2: In this layer, nodes are fixed and output of each node is given by (5) which represents a weight of the rule.

$$O_{2,i} = w_i = \mu_{A_i}(x)\mu_{B_i}(y) \quad for\ i = 1,2 \tag{5}$$

Layer 3: In this layer, nodes are fixed. The ratio of the i^{th} rule's firing weight to the sum of all rules weights is computed and is given by (6).

$$O_{3,i} = \overline{w_i} = \frac{w_i}{w_1 + w_2} \tag{6}$$

Layer 4: In this layer, nodes operate as a function block, whose variables represents the input values and parameters are adaptive. Overall output (TSK output) of this layer is given by (7).

$$O_{4,i} = \overline{w_i} f_i = \overline{w_i}(p_i x + q_i y + r_i) \tag{7}$$

Here p_i, q_i and r_i denotes consequent parameters to be determined.

Layer 5: Output of this layer is the summation of all the input signals. The final output is given by (8).

$$O_{5,i} = \sum_i \overline{w_i} f_i = \frac{\sum_i w_i f_i}{\sum_i w_i} \tag{8}$$

In this work, ANFIS Linearizer is going to be hardware implemented. Analysis of the learning phase results presented in Table 2 guide us to choose the method highlighted in gray; two input triangle membership functions and two linear output membership functions with three parameters each one. ANFIS architecture for linearization of nonlinear sensor's characteristic is illustrated by Figure 6.

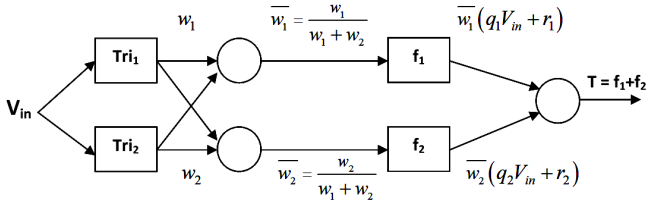

Figure 6: ANFIS architecture for linearization

With reference to Fig. 6, the overall output is given by (9). Equation (10) gives the expression for $Tri_i(x)$ function

.

$$f = \frac{(q_1 x + r_1) Tri_1(x) + (q_2 x + r_2) Tri_2(x)}{Tri_1(x) + Tri_2(x)} \tag{9}$$

$$Tri_i(x) = \begin{cases} 0 & if\ x \le a_i \\ \frac{x - a_i}{b_i - a_i} = \frac{1}{b_i - a_i} x - \frac{a_i}{b_i - a_i} & if\ a_i \le x \le b_i \\ \frac{c_i - x}{c_i - b_i} = \frac{-1}{c_i - b_i} x + \frac{c_i}{c_i - b_i} & if\ b_i \le x \le c_i \\ 0 & if\ c_i \le x \end{cases} \tag{10}$$

4.2 Generating Input and Target Data

Input data is generated with the help of data sheet provided by the manufacturer of thermistor. The data sheet provides the values of thermistor resistance with respect to temperature. From these values the input data that is thermistor non linear voltage across 1k ohm resistor is calculated with the help of voltage divider formula. The non linear data generated that is V_{1K} is then plotted with respect to temperature by taking the help of third party software. The temperature v/s input data plot is shown in Figure 7. Once again by taking the help of third party software, the linear fit is obtained along with the slope and intercepts values. The corresponding linear fit is shown in Figure 8. From this input data, ANFIS is trained to generate the target data for sensor linearization.

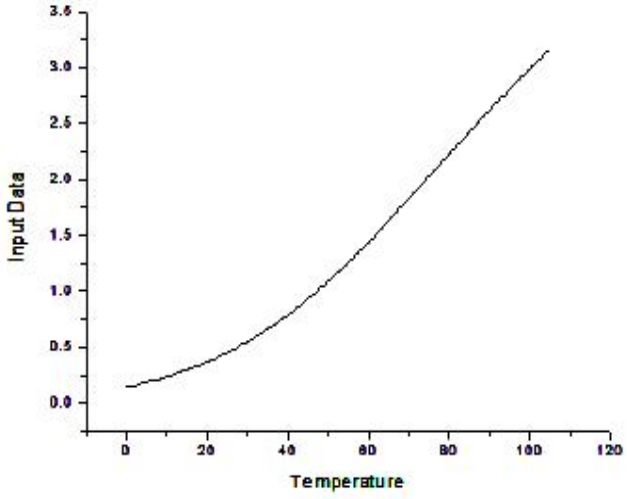

Figure 7: Temperature v/s Input Data Plot

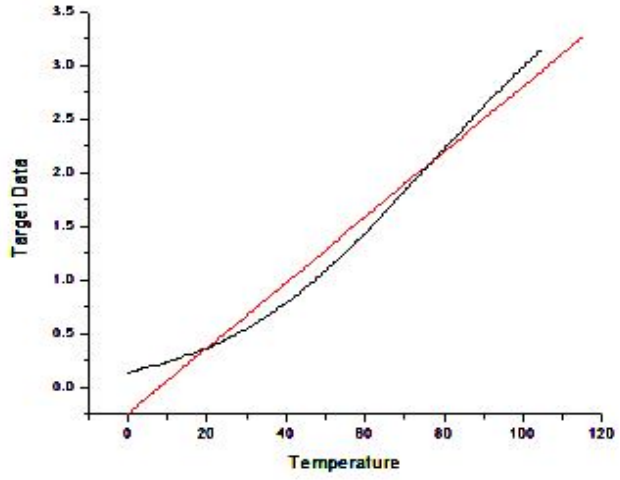

Figure 8: Linear Fit Plot

4.3 Training ANFIS

For linearization of nonlinear sensor's characteristic, a neuro-fuzzy toolbox from Matlab-Simulink software was used for training the ANFIS linearizer. With the help of given i/p – o/p data set of modeled system, ANFIS creates a Takasi-Sugeno-Kang fuzzy inference system (TSK FIS). This makes fuzzy system to learn from data set of modeled system. The computation of membership parameters is assured by gradient vector.

5. H/W & S/W CO-SIMULATION IN XSG

Sometimes it is necessary to incorporate HDL modules to a XSG design. XSG's black box block allows the use of VHDL/Verilog HDL in XSG design. With the help of black box block, HDL modules can be incorporated into XSG

design. When black box blocks are compiled, XSG automatically wires the incorporated module and corresponding files into the surrounding netlist [18]. Figure 9 shows the design of our architecture with XSG. The black box block contains the VHDL code defined for ANFIS linearizer used for linearization of nonlinear sensor's characteristics.

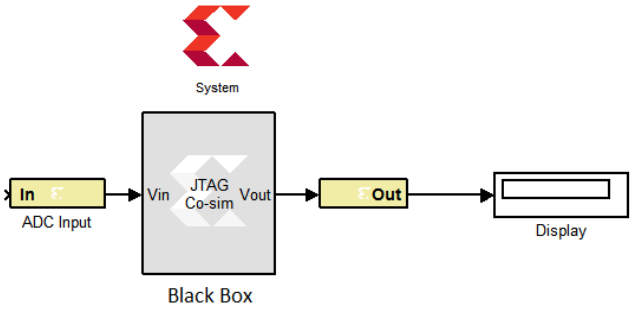

Figure 9: XSG Simulation Project

HDL co-simulation process involves simulation of black boxes using HDL simulator, compilation of HDL, generation of additional HDL test benches, scheduling simulation events and handling exchange of data between HDL simulator and Matlab-Simulink. XSG supports hardware co-simulation feature in which a real time design executing on an FPGA chip can be directly imported into a Matlab - Simulink simulation environment. After hardware co-simulation compilation, a bitstream is automatically created and associated to a corresponding. Figure 10 shows the same.

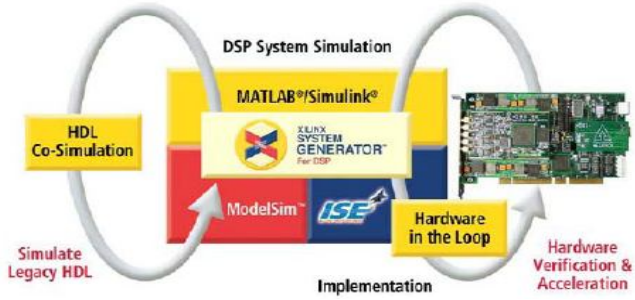

Figure 10: FPGA based Hardware-Software (HW-SW) co-simulation environment [18]

When the simulation of design is carried out in Matlab-Simulink environment, the results are calculated in actual FPGA chip. This results in very fast simulation times while verifying hardware's functionality correctness.XSG provides JTAG interface between Matlab – Simulink and FPGA hardware platform with the help of Xilinx programming cable. Model with hardware in the loop testing on FPGA Spartan 6 platform is shown in Figure 11.

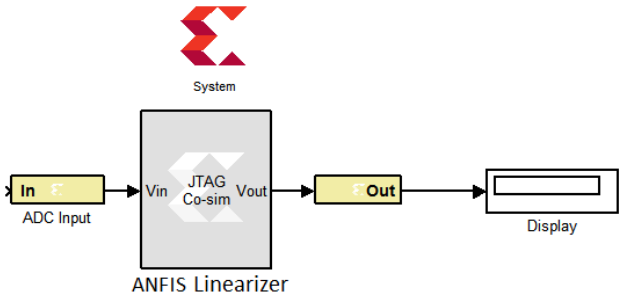

Fig. 11. XSG project for hardware co-simulation testing on Xilinx's Spartan 6 FPGA devices

Xilinx's Spartan 6-XC6SLX45 FPGA device with a clock speed of 100 MHz is used. Table 1 shows the synthesis report for optimization setting.

Table 1: Xilinx Synthesis Report

Device Utilization Summary			
Slice Logic Utilization	**Used**	**Availabl e**	**Utilizatio n**
Number of LUTs	282	27,288	1%
Number of occupied Slices	142	6,822	1%
Number of MUXCYs	52	13,644	1%
Number of bounded IOBs	65	218	29%
Number of DSP48A1s	9	56	15%

In practice, extra blocks are required for input/output interfaces, and synchronization.

6. RESULT AND DISCUSSION

With reference to Figure 1, ANFIS linearizer block has single input and single output. The block has to learn sensor's inverse characteristic. Many different methods were analyzed by making changes in the type and number of input membership functions and the type of output membership function. Table 2 represents learning phase results for different approaches. The error between the ANFIS output and sensor's inverse characteristic output represents mean square errors (MSE).

Table 2: Learning Phase Results

Input Membership		Training Method	Output Membership Type	Error	Epoch
Type	**Number**				
Triangle	2	Hybrid	Constant	0.089562	500
			Linear	0.042277	200
		Back Propagation	Constant	0.10639	500
			Linear	0.20319	500
	3	Hybrid	Constant	0.065469	250
			Linear	0.069862	50
		Back Propagation	Constant	0.073769	500
			Linear	0.066717	500
	4	Hybrid	Constant	0.028184	100
			Linear	0.039662	20
		Back Propagation	Constant	0.035134	500
			Linear	0.031191	500

Trapeze	2	Hybrid	Constant	0.052064	300
			Linear	0.043712	200
		Back Propagation	Constant	0.051347	500
			Linear	0.14182	500
	3	Hybrid	Constant	0.043692	200
			Linear	0.018114	200
		Back Propagation	Constant	0.044042	500
			Linear	0.040508	500
	4	Hybrid	Constant	0.032136	200
			Linear	0.0081451	220
		Back Propagation	Constant	0.034315	500
			Linear	0.029078	500
Bell Shape	2	Hybrid	Constant	0.068148	500
			Linear	0.029257	500
		Back Propagation	Constant	0.11602	500
			Linear	0.17479	500
	3	Hybrid	Constant	0.014575	500
			Linear	0.0094959	230
		Back Propagation	Constant	0.066668	500
			Linear	0.032527	500
	4	Hybrid	Constant	0.0080987	500
			Linear	0.0068352	120
		Back Propagation	Constant	0.063041	500
			Linear	0.026443	500
Gauss	2	Hybrid	Constant	0.21132	500
			Linear	0.044937	500
		Back Propagation	Constant	0.23609	500
			Linear	0.2041	500
	3	Hybrid	Constant	0.059039	500
			Linear	0.017174	420
		Back Propagation	Constant	0.12191	500
			Linear	0.041409	500
	4	Hybrid	Constant	0.018454	500
			Linear	0.010377	250
		Back Propagation	Constant	0.12445	500
			Linear	0.036323	500
Gauss2	2	Hybrid	Constant	0.074492	500
			Linear	0.027118	500
		Back Propagation	Constant	0.11443	500
			Linear	0.17679	500
	3	Hybrid	Constant	0.03082	500
			Linear	0.016948	100
		Back Propagation	Constant	0.083049	500
			Linear	0.055935	500
	4	Hybrid	Constant	0.0150537	500
			Linear	0.0089628	400
		Back Propagation	Constant	0.062011	500
			Linear	0.030194	500

Table 3 illustrates the parameters for input and output membership functions obtained in the learning phase for ANFIS architecture.

Table 3: Parameters for ANFIS architecture

Input Membership			**Output Membership**		
Tri_1	a	-3.13	f_1	p	0
	b	-35		q	4.5
	c	5.169		r	-0.03
Tri_2	a	0.21	f_2	p	0
	b	3		q	1.225
	c	6.305		r	0.5

Figure 12 shows the result for real time hardware co-simulation of sensor linearization for temperature measurements.

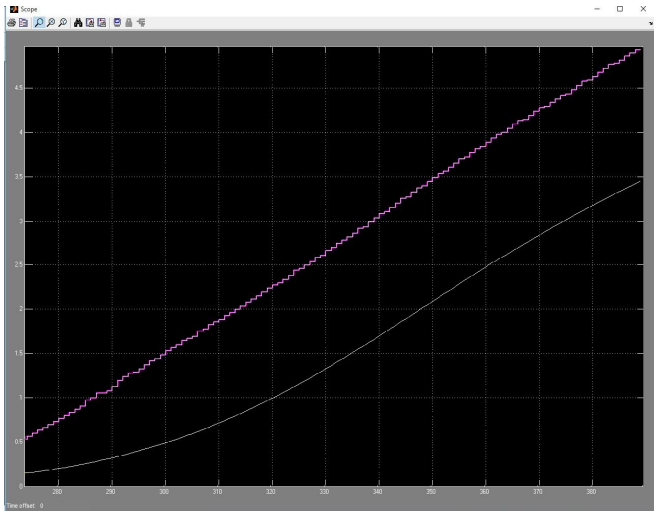

Figure 12: Real Time Hardware Co-simulation of Sensor Linearization for Temperature Measurements

7. CONCLUSION

With the help of XSG hardware co-simulation feature, simulation and simultaneous verification of design hardware was greatly accelerated. The purpose of this paper was to design an ANFIS linearizer for linearization of nonlinear sensor. This design is implemented on a low cost Waxwing Spartan 6 FPGA (XC6SLX45) Development Board. Test results and analysis of an implemented architecture shows that the device does an accurate linearization.

Future works include implementation of hardware co-simulation of some smart features like auto calibration of sensor, sensor drift compensation and sensor fault detection on Xilinx's FPGA.

REFERENCES

[1] G. Bucci, M. Faccio and C. Landi, New ADC with Piecewise Linear Characteristic: Case Study – Implementation of a Smart Humidity Sensor, IEEE Trans, Instrumentation and Measurement, vol. 49, No. 6, pp. 1154-1166, Dec. 2000. https://doi.org/10.1109/19.893250

[2] N. N. Charniya, Some Features of Neural Networks based Intelligent Sensors and Design Issues, International Journal of Computer Applications, vol. ICCIA, No.2, pp. 1-4, March 2012.

[3] Alaa Abdul Hussein Salman , Fadhil Rahma Tahir, Mofeed Turky Rashid, ―Design and implementation model for linearization sensor characteristic by FPAA, Iraq J. Electrical and Electronic Engineering, Vol. 11 No.2, 2015. https://doi.org/10.33762/eeej.2015.106238

[4] C. Alippi, A. Ferrero and V. Piuri, "Artificial intelligence for instruments and measurement applications", IEEE Instrumentation and Measurement Magazine, vol. 1, No. 1, pp. 9-17, March 1998. https://doi.org/10.1109/5289.685492

[5] Caryl Charlene Escolar-Jimenez, Kichie Matsuzaki and Reggie C. Gustilo, "A Neural-Fuzzy Network Approach to Employee Performance Evaluation". International Journal of Advanced Trends in Computer Science and Engineering, Vol. 8.3, pp. 573- 581, 2019. https://doi.org/10.30534/ijatcse/2019/37832019

[6] J. S. Jang, C. T. Sun, and E. Mizutani, *Neuro-Fuzzy and Soft Computing*. Englewood Cliffs, NJ: Prentice-Hall, 1997.

[7] T. Takagi and M. Sugeno, "Fuzzy identification of systems and its application to modeling and control," *IEEE Trans. Syst. Man Cybern.*, vol. 15, pp. 116–132, Feb. 1985.

[8] C. F. Juang and C. T. Lin, "An on-line self-constructing neural fuzzy inference network and its applications," *IEEE Trans. Fuzzy Syst.*, vol. 6, pp. 12–32, Feb. 1998. https://doi.org/10.1109/91.660805

[9] M. R. Emami, I. B. Turksen, and A. A. Goldenberg, "Development of a systematic methodology of fuzzy logic modeling," *IEEE Trans. Fuzzy Syst.*, vol. 6, pp. 346–368, Apr. 1998.

[10] S. Abe and R. Thawonmas, "A fuzzy classifier with ellipsoidal regions," *IEEE Trans. Fuzzy Syst.*, vol. 5, pp. 358–368, Apr. 1997.

[11] S. N. Engin, J. Kuvulmaz and V. E. Ömurlü, "Fuzzy control of an ANFIS model representing a nonlinear liquid-level system," *Neural Computing & Applications*, September 2004, Volume 13, Issue 3, pp 202-210. https://doi.org/10.1007/s00521-004-0405-4

[12] T. TAKAGI and M. SUGENO, "Fuzzy identification of systems and its applications to modeling and control", IEEE transactions on systems, man, and cybernetics, vol. 15, no. 1, pp. 116–132, 1985.

[13] Sri Hari Nallamala , Dr. Pragnyaban Mishra and Dr. Suvarna Vani Koneru, "Qualitative Metrics on Breast Cancer Diagnosis with Neuro Fuzzy Inference Systems". International Journal of Advanced Trends in Computer Science and Engineering, Vol. 8.2, pp. 259- 264, 2019. https://doi.org/10.30534/ijatcse/2019/26822019

[14] Jyh-Shing Roger Jang, "ANFIS: adpative network based inference system," *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 23, N°3, May/June 1993.

[15] O.AKIN, I. ALAN, "The use of FPGA in field-oriented control of an induction machine," Turk J Elec Eng & Comp Sci, Vol. 18, No.6, pp. 943-962, 2010.

[16] S. Simard, J. G. Mailloux, and R. Beguenane, "Prototyping advanced control systems on FPGA," EURASIP J. on Emb. Sys. Vol. 2009, pp 1–12, 2009. https://doi.org/10.1155/2009/897023

[17] T. Saidani, D. Dia, W. Elhamzi, M. Atri, and R. Tourki, "Hardware Co-simulation For Video Processing using Xilinx System Generator," Proc, of the World Congress on Eng, (IWCE), London, UK, July 1–3, 2009.

[18] Xilinx System Generator User's Guide, www. Xilinx.com.

[19] The MathWorks Inc. Embedded Matlab Language User Guide (2013).

[20] P.N. Mahana, F.N. Trofimenkoff, "Transducer output signal processing using an eight-bit microcomputer," *IEEE Trans. Instrum. Meas.*, Vol. 35, 1986, pp. 182-186. https://doi.org/10.1109/TIM.1986.6499087

[21] A. Flammini, D. Marioli, A. Taroni, "Transducer output signal processing using an optimal look-up table in microcontroller based systems," *Electron. Lett.*, Vol. 33, 1997, pp 1197-1198. https://doi.org/10.1049/el:19970809