Sensors & Transducers, Vol. 191, Issue 8, August 2015, pp. 135-151

Sensors & Transducers

http://www.sensorsportal.com

Multi Channel Sensor Linearization in Field Programmable Gate Array for Real Time Applications

*** Durlav Sonowal, Manabendra Bhuyan**
Tezpur University, Tezpur, 784028, India
Tel.: +91 03712 275267
*E-mail: dsn@tezu.ernet.in

Received: 31 July 2015 /Accepted: 25 August2015 /Published: 31 August 2015

Abstract: In industrial applications multi-channel data acquisition and logging is of prime importance for process monitoring and control. In such situations linearization of the non-linear responses obtained from multi-channel data acquisition systems is of prime importance for precise monitoring and control. Although various techniques have been developed for sensor linearization, real time multi channel linearization approaches are rare. This paper describes FPGA (Field Programmable Gate Array) implementation of different linearization techniques for multi-channel nonlinear sensor characteristics. The proposed multi-channel linearization techniques are implemented in real time using - piecewise linearization (PWL), look up table (LUT) based linearization, linearization by interpolation (LI) and artificial neural network (ANN) based linearization methods. We have discussed the different aspects of these linearization techniques and the trade-off between accuracy and implementation area in FPGA. The comparative analysis was performed by using identical thermistors connected to 8 channels for linearization and the performance of each method was analyzed. The performance of different techniques of linearization is estimated on the basis of logic utilization, linearization accuracy, execution time, noise immunity and speed of operation. Fixed point arithmetic has been used for data representation in the FPGA implementation.

Keywords: Multi channel, Sensor linearization, FPGA, ANN, Piecewise linearization, Linear interpolation and look up table.

1. Introduction

Sensors are the primary elements of any instrumentation system. Many of the sensors available for measurement of various physical quantities have nonlinear input-output characteristics. In many industrial applications multi-channel data acquisition and logging becomes essential for process monitoring and control. In such situations linearization of the non-linear responses obtained from multi-channel data acquisition systems is of prime importance for precise monitoring and control. Therefore it is necessary to process the output of the sensor with nonlinear characteristic to have a linear input-output relation for an accurate representation of a physical variable. In general, the prior developed techniques of sensor linearization can be broadly classified into two categories - hardware based linearization and software or PC based linearization [1].

Till date, a large number of techniques have been proposed on sensor linearization. For ease of explanation and discussion, we categorize these sensor linearization techniques in following ways:

1) Analog circuit and ADC based;
2) Microcontroller based;
3) VLSI and FPGA based;
4) Software based.

1.1. Analog Circuit and ADC Based

In [2], a transducer linearization method using ratiometric property of analog to digital converter (ADC) was proposed. In this technique the maximum relative deviation of nonlinearity is found as 0.5 kPa in a range of 100 kPa. In a similar type of work in [3], a nonlinear ADC with digitally selectable quantizing characteristics was proposed for linearization of nonlinear analog signals. In [4], a programmable nonlinear ADC was proposed that uses an optimal size of ROM to realize the nonlinear characteristics. In [5] a nonlinear ADC circuit based on switched capacitor architecture was proposed that uses piecewise linear approximation of the nonlinear conversion characteristics. The maximum nonlinear deviation was found as 0.0937 l/hr (1LSB in the range of 0-24 l/hr). In [6], a four constant curve fit was proposed and a temperature to frequency converter circuit was developed to employ the proposed fit. The peak absolute error of linearization is found to be 0.5 K. In [7], linearization of an NTC thermistor based on 555 timer circuit with frequency and analog output was proposed. In this method, the maximum percentage nonlinearity error claimed is ±1 % and ±1.7 % in two different ranges. In [8], functional link artificial neural network (FLANN) based linearization is proposed using a plug-in module type of digital circuit, where the maximum full scale error is claimed as ±2 %.

1.2. Microcontroller Based

In [9], an optimal method of sensor linearization using look up table in small embedded system was proposed. In this work a theory was also proposed for finding the minimum allowable size of a look up table without affecting the precision. However, this method is limited to LUT entries with only integer part, hence precision is also limited. In [10], a microcontroller based multiple sensor linearization using ANN has been proposed. The maximum approximation error of *tanh* function was predicted as less than 1%.

1.3. VLSI and FPGA Based

In another approach a MOS VLSI model was proposed for linearization of second and third order sensor models [11]. The circuit uses a simple continuous time analog signal processing cell that allows single chip fabrication. But in higher order linearization using MOS transconductance model, there is a problem of error due to amplifier dynamics and system stability due to the feedback. The maximum frequency of operation in this method is 1 kHz.

In an FPGA based sensor linearization and compensation of nonlinearity due to the environmental affect using ANN [12], the maximum full scale error is observed to be ±1.5 %. The method in [13], proposes a low cost microcontroller for linearization of a distance sensor by look up table (LUT), piece-wise linear interpolation and extrapolation (PWLI/E), fuzzy and ANN. The nonlinearity error analysis of this work shows a maximum relative error of more than 0.3 in Fuzzy approach, 0.15 in ANN and 0.10 in PWLI.

In earlier works, of the authors of this paper [14-15], FPGA implementation of thermistor linearization was presented, where a nonlinear thermistor characteristic for a temperature range of 0 – 100^{o}C was linearized in FPGA. However the data format used in FPGA was IEEE 745 32 bit format which consumes higher logical area. Moreover the methods were suggested for real time applications of a single channel. In this work we have used fixed point format for data representation in FPGA which reduces logical area of implementation and the performance of the system was examined using real time signal.

1.4. Software Based

In software based approach researchers developed several algorithms for transducer linearization. In [16], a graphical approach was proposed that uses a numerical method of iteration to obtain a linear solution. This method offers flexibility of online choice and replacement of thermistor over an extended range of operation. In [17], a method for function approximation in microcomputer memory implementing the inverse characteristics of the sensor for smart sensor system (SSS) was proposed. In [18], simulation of thermistor linearization using a two layer ANN was proposed. Some other software based sensor linearizations were- multi layer perceptron (MLP) ANN in [19-20], sensor calibration and compensation by adaptive linear element based neural network in [21], temperature compensation of a pressure sensor using BP neural network in [22].

From the above discussions it can be summarized that in a computer based linearization technique, different algorithms like interpolation, piecewise linearization, look up table, ANN etc. can easily be implemented using different software tools like MATLAB or programming language like C/C++. However, software modeled linearization methods are sequential with significant effects to time response for multichannel. In hardware or circuit level linearization techniques, both analog and digital methods are possible. There are applications where both digital and analog linearizations are combined in hybrid mode [23].

Sensor linearization can be achieved by different types of digital implementations such as - microcontroller based implementation, DSP based implementations and FPGA based implementation. Microcontroller and DSP based implementations are sequential and hence do not preserve the parallel

structure which is a vital requirement for high speed multi-input-multi-output (MIMO) systems. For example, DSP based or microcontroller based implementation of ANNs do not preserve the parallel architecture of the networks. On the other hand, FPGA devices offer a great flexibility in reconfiguration simultaneously maintaining the parallelism where required.

The use of ANN for sensor compensation and linearity correction is proposed in many literatures. ANN provides an efficient tool for mapping nonlinear input-output relations of sensor signals. Various ANN based linearization techniques are available such as - functional link artificial neural network (FLANN) based linearization [8]; ANN based sensor linearization and compensation of nonlinearity due to the environmental effect [12]; low cost microcontroller for ANN based linearization of a distance sensor [13]; simulation of thermistor linearization using a two layer ANN [18]; sensor linearization based on MLP ANN [19]; sensor calibration and compensation by adaptive linear element based neural network [21] and temperature compensation of a pressure sensor using BP neural network [22].

FPGAs are basically integrated circuits designed and configured by the user, which contains a matrix of configurable logic blocks and a hierarchy of reconfigurable interconnects that allow the blocks to be connected together as per the requirements of the user. The FPGA implementations of ANNs are discussed in many literatures [24-39]. A fully digital MLP ANN implemented with two XC3042 FPGAs and a 1 k × 8 EPROM was presented in [25-26]. In [27], authors discussed the FPGA implementation of ANN and showed that the high speed operation in real time applications of ANN could be achieved only if the networks were implemented using parallel hardware architecture. In [28], parameterized neuron was developed in FPGA. In [29], a neural network topology was described that implements the bit-serial architecture by reducing the requirement of hardware complexity. Earlier, researchers thought that the size of an ANN importantly determines the performance of ANN, however, recently a large number of researchers believe that the network architecture is responsible for determining the performance of the network [30-31]. In [32], a layer multiplexing technique for feedforward NN was proposed that reduces the requirement of hardware in ANN. In [33], the use of FPGAs in industrial control applications and neural network based FPGA systems were highlighted. In [34], generalized backpropagation multilayer perceptron architecture was described for online applications.

From the above literature survey it is observed that ANN has been implemented in FPGA for different applications – industrial control, vector control drive etc., however very little effort has been made for ANN implementation in FPGA for sensor linearization except in [16]. In [16], ANN has been used for sensor linearization and compensation of nonlinearity due to the environmental effect and the maximum full scale error is found to be ±1.5 %. However, FPGA based linearization by piecewise segmentation, linear interpolation and look-up table is not available in the literature so far.

Although microcontroller based linearization is cost effective than FPGA for a single channel application, FPGA based linearization is promising for multichannel sensor linearization. Although microcontroller can be interfaced to multichannel ADCs the computational bandwidth cannot be optimally utilized compared to FPGA. This is because, FPGA can be utilized in parallel hardware architecture. Therefore, FPGA based sensor linearization will be a better solution in multi-channel sensor array applications. In multi-channel sensor arrays applications different types (monotype or multi-type) of sensors are installed in different locations of an application and FPGA implementation becomes a simple task by routing the sensor outputs through an ADC to identical FPGA 'blocks' in hardware. Fig. 1 shows a proposed scheme for multichannel implementation of linearization in FPGA where the FPGA can be configured with parallel architecture of linearization blocks for parallel computation of sensor linearization.

In this work we have implemented multi channel linearization of sensors using the following techniques in FPGA, (1) Piecewise linearization, (2) Look up table (LUT) based linearization, (3) Linearization by interpolation and (4) Artificial Neural Network (ANN) based linearization.

For implementation of linearization technique in FPGA we have used a total of 8 channels, however due to overflow of FPGA LUTs the channel allocation has been done as - Piecewise linearization (8 channels), Linear Interpolation (8 channels), Look up Table (3 channels) and ANN (4 channels). Thermistors have been connected to all the 8 channels to acquire signals for linearization while a load cell has been used to analyze the speed of operation of the linearization techniques. This is because fast variation of temperature cannot be obtained from a thermistor while the load cell shows a fast response on loading and unloading.

The paper is organized as follows; Section-2 presents a background of linearization. In Section-3, We have discussed the FPGA implementation of linearization methods and results and discussion are presented in each subsection and the work is concluded in Section-4.

2. Background

A sensor or sensor system generally converts the physical quantities into an electrical signal, preferably into voltage signal. The basic linearization theory states that if $y = f(x)$ is a nonlinear function that describes the relationship between input variable x and its corresponding nonlinear output y shown in

Fig. 2, then the relationship between sensor output y and output of the linearization unit z must be an inverse function such that,

$$z = f^{-1}(y) = f^{-1}(f(x)) = x \quad (1)$$

Here we have linearized output voltage (y) of the voltage divider circuit resulting a linear output. To linearize the nonlinear function $y = f(x)$ the linearization unit has been configured digitally such that $g = f^{-1}$, i.e. $g(f(x)) = f^{-1}(f(x)) = x$ .For online implementation of the proposed techniques, 8 identical thermistors (with $R_0 = 10\,k\Omega$ and thermistor constant $\beta = 3750\,K$) have been connected with voltage divider circuits.

The output of the voltage divider circuits are then connected to all the 8 channels of the ADC. Fig. 3 shows the thermistor circuit connected to an 8-bit, 8 channel ADC (ADC0808) interfaced with Spartan-III XC3S400 FPGA. In the thermistor circuit a fixed resistance value of $1\,k\Omega$ (tolerance-0.005 %, temperature coefficient of 0.2 ppm/ºC and thermal stability less than 1 sec) is used which provides output voltage in the range from $0.1569\,\text{V} - 3.2079\,\text{V}$ for a temperature range of 0 – 100 ºC. The voltage source in the thermistor circuit is +5 V powered by a precision voltage source **(**ST4076) with a ripple noise specification of less than 1 mV (rms) and load regulation of ±0.05 %.

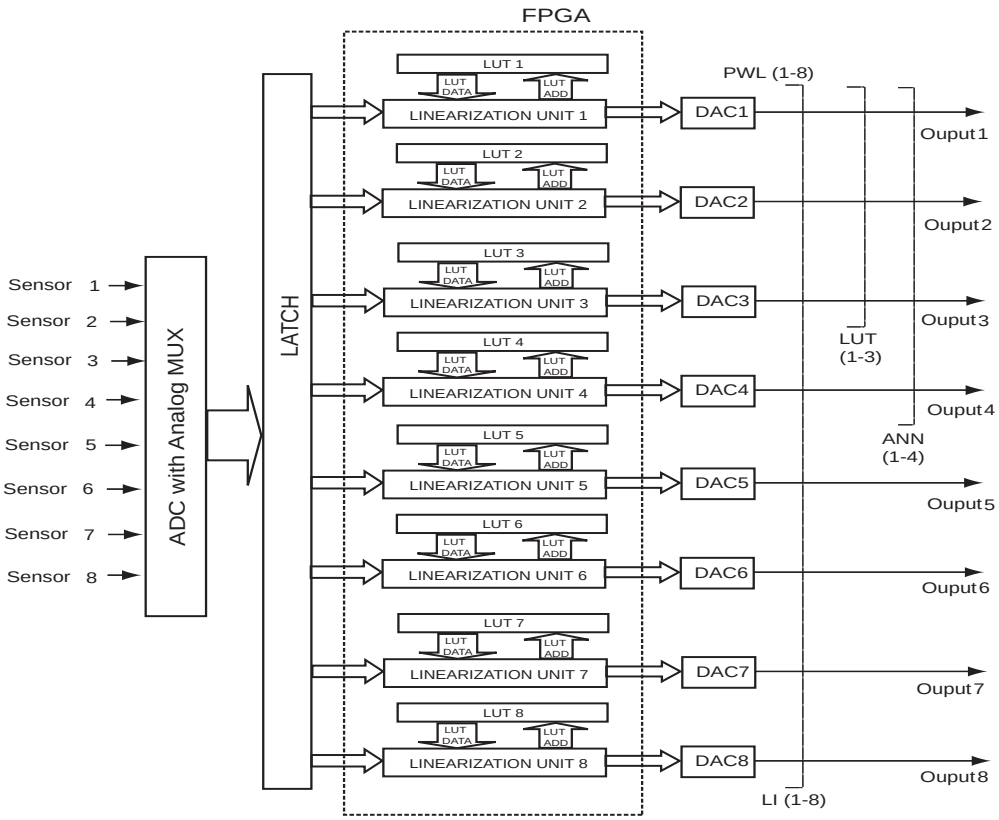

Fig. 1. Multiple sensor signal linearization in FPGA.

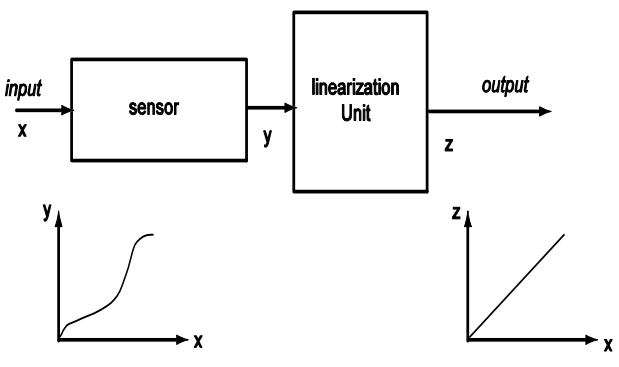

Fig. 2. Basic Linearization Process.

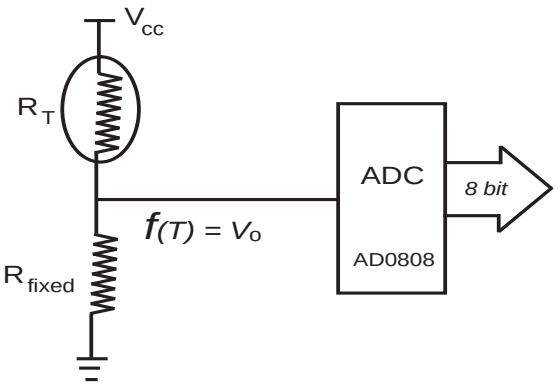

Fig. 3. Primary measurement circuit.

We have also implemented the finite state machines in FPGA to drive the ADC, DACs and LCD. DAC AD7541 provides analog version of the linearized output. Fig. 1 shows the block diagrams of the linearization unit. The Spartan-III XC3S400 FPGA was synthesized and configured using Xinilx ISE 11.

3. FPGA Implementation

As discussed we have implemented four different techniques of sensor linearization in which methods are described in details in the following subsections.

3.1. Piecewise Linearization (PWL)

In piecewise linearization method, the nonlinear sensor characteristic is divided into a certain number of linear segments. Each of the segments within its linear range is fitted to a known linear equation having a slope (m) and a bias (c). The number of segments in the nonlinear characteristic is dependent on accuracy of the linearization required and the length of operating range. However, the characteristic equation for each linear segment depends on the nonlinear characteristics of the sensor for the operating range. Considering n numbers of segments in the non-linear thermistor characteristics, each segment of the characteristic equation can be expressed by a straight line equation for which the slopes (m) and intercepts (c) are different. The equations of the straight line segments can be determined either graphically or using the coordinates of the two end points of the segments. The equation of the straight line can be written as –

$$y_{n-1,n} = m_n x_{n-1,n} + c_n, \quad (2)$$

where subscript n indicates the segment number and $y_{n-1,n}$ and $x_{n-1,n}$ indicate the values in $(n-1)^{th}$ to n^{th} range. Here, initially we have considered 20 segments for linearization of the nonlinear output voltage obtained from the thermistor measurement circuit and the corresponding linearized temperature values are obtained by using the following equations

$$\begin{aligned} T_{i,1} &= m_1 V_{o,1} + c_1 \\ T_{i,2} &= m_2 V_{o,2} + c_2 \\ &\cdots \\ T_{i,20} &= m_{20} V_{o,20} + c_{20} \end{aligned} \quad (3)$$

Before configuring the linearization algorithm in FPGA the slopes (m_n) and biases (c_n) given by Equation (3) are calculated. Table 1 shows the piecewise linear equations for 20 segments that we have used in this work. The same have been repeated for $n = 30$ and $n = 40$.

Table 1. Piecewise Linear Equations of Thermistor Characteristics for the Temperature Range of 0-100ºC with 20 Segments.

Temperature Range, (ºC)	Output Voltage Range, (V)	Linear Equations
[0-5]	[0.1569-0.2019]	$T_i = 123.852V_o - 19.631$
[5-10]	[0.2019-0.2567]	$T_i = 103.8084V_o - 15.6448$
[10-15]	[0.2567-0.3230]	$T_i = 87.9795V_o - 11.7344$
[15-20]	[0.3230-0.4018]	$T_i = 75.4031V_o - 7.9128$
[20-25]	[0.4018-0.4945]	$T_i = 65.3598V_o - 4.1949$
[25-30]	[0.4945-0.6021]	$T_i = 57.3075V_o - 0.5982$
[30-35]	[0.6021-0.7251]	$T_i = 50.835V_o + 2.8577$
[35-40]	[0.7251-0.8637]	$T_i = 45.6279V_o + 6.1501$
[40-45]	[0.8637-1.0178]	$T_i = 41.445V_o + 9.2532$
[45-50]	[1.0178-1.1866]	$T_i = 38.1001V_o + 12.1383$
[50-55]	[1.1866-1.3686]	$T_i = 35.4491V_o + 14.7727$
[55-60]	[1.3686-1.5619]	$T_i = 33.33806V_o + 17.12$
[60-65]	[1.5619-1.7644]	$T_i = 31.8081V_o + 19.1399$
[65-70]	[1.7644-1.9732]	$T_i = 30.6654V_o + 20.7875$
[70-75]	[1.9732-2.1856]	$T_i = 29.9016V_o + 22.0133$
[75-80]	[2.1856-2.3987]	$T_i = 29.4786V_o + 22.7628$
[80-85]	[2.3987-2.6098]	$T_i = 29.3687V_o + 22.9762$
[85-90]	[2.6098-2.8164]	$T_i = 29.5523V_o + 22.5884$
[90-95]	[2.8164-3.0163]	$T_i = 30.017V_o + 21.5284$
[95-100]	[3.0163-3.2079]	$T_i = 30.7563V_o + 19.719$

3.1.1. FPGA Implementation of PWL

Fig. 4 illustrates the piecewise linearization algorithm in FPGA. Here, the output voltages (V_o) from the multi-channel voltage divider circuits are fed to an ADC and the digital versions of V_o are compared online with previously stored voltage ranges ($V_{o,n}$). The comparator generates slopes m_n

and bias c_n for the n^{th} range of input data. The corresponding slope (m_n) is multiplied with input voltage which is then added with the corresponding bias c_n to get the linear temperature value T_i given by a straight line equation.

$$T_{i,1} = m_1 V_{o,1} + c_1$$

of the i^{th} piecewise linear segment. The linearized outputs obtained from the FPGA system are processed through the DACs to obtain the analog outputs.

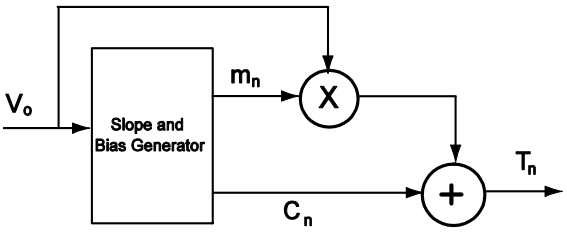

Fig. 4. Block Diagram of piecewise linearization.

3.1.2. Data Representation

In FPGA devices, arithmetic operations are performed using either floating point or fixed point arithmetic. For higher degree of accuracy and precision; floating point arithmetic such as IEEE 754 std. 32 bit and 64 bit are used. However, the floating point arithmetic is complex as compared to fixed point arithmetic and consumes higher implementation area in FPGA.

Fixed point arithmetic is a trade-off between conventional integer arithmetic and floating point and therefore we have adopted fixed point arithmetic operation for implementation of linear interpolation in FPGA. The major advantage of fixed point arithmetic is that it follows the same basic mathematical rule for addition, subtraction, multiplication and division that is applicable for conventional binary numbers. For this reason fixed point arithmetic can easily be implemented in FPGA with minimum resource consumption and minimum complexity.

In this work we have used different bit lengths for input voltage (V_o), slope (m_n), bias (c_n) and output (T_i). The input data V_o is represented using $S2.10$ fixed point format, i.e. MSB is the sign bit, two bits for the integer (I) and ten fractional places (F) as shown below-

SII.FFFFFFFFFF

This format is suitable to represent the voltage levels ranging from 0 V to 3.9 V which is sufficient for this work. The format also provides a data precision of 2^{-10} which is 9.7656×10^{-4} in this case. For illustration, let us consider a voltage level of '1.35V' for representation in $S2.10$ fixed point format. Here the number is positive, therefore $S = 0$ and the integer part '1' is represented using 2 bit binary as '01'. The fractional part '0.35', is multiplied by 1024 (2^{10}) and then converted to binary, i. e., 0.35 × 1024 = 358.4~ 358. The 10 bit binary equivalent of '358' is '0101100110'. Therefore the number '1.35' can be represented in $S2.10$ fixed point formats as '010101100110'.

The slope (m_n) and bias (c_n) are also represented in the same $S2.10$ format and output temperatures (T_i) are represented in $S8.8$ fixed point format for getting higher ranges.

3.1.3. Results of PWL

In this work we have implemented piecewise linearization of an array of 8 nonlinear thermistors in FPGA for three different numbers of piecewise segments, i.e. 20, 30 and 40. We have used performance measures- root mean square error (RMSE), mean absolute deviation (MAD) and standard deviation (SD) for evaluation of performance with varied number of segments. The experiment is performed with real time signals. The thermistors connected to the voltage divider circuits are placed in temperature controlled chamber. The temperature is varied from 0 – 100°C. The outputs of the voltage divider circuits and the outputs of DACs after linearization are recorded using LabVIEW based data acquisition system (NI DAQ USB 6351). Outputs of the voltage divider circuits and FPGA outputs are compared with the output of a K-type thermocouple by using LabVIEW based standard temperature measurement system.

Fig. 5 shows the full scale percentage error for 8 channels with 40 segments of each channel and Fig. 6 shows the full scale percentage error for channel 1 with $n = 20$, $n = 30$ and $n = 40$. Table 2 shows RMSE, MAD, SD, Maximum full scale error for all 8 channels with 40 segments. In a similar way, RMSE, MAD, SD, Maximum full scale error was calculated for linearization with 20 and 30 piecewise segments. Table 3 presents a comparative analysis for different number of segments for one channel. A maximum full scale (FS) error of 2.5064 % and 1.5027 % are observed with 20 and 40 piecewise linear segments respectively.

The figure shows a 45.24 % reduction in RMSE due to the increase in number of linearization segments from 20 to 40. However, 4 % to 7 % increase in LUT area utilization is observed when the number of segment is increased from 20 to 40. The device uses 290 elements of 4-input LUTs with 20 segments with maximum FS error of 2.5064 %; whereas with 556 elements of 4-input LUTs for 40 segments it results a 1.5027 % maximum FS error, i.e. the maximum error is reduced by 40 % by increasing the number of linear segments from 20 to 40.

It is important to know that how much noise has been contributed by the linearization process since the overall performance of the proposed method does not depend only on the linearity error, but also on the

noise contributed by the electronics such as fluctuation of electron (Johnson-Nyquist noise) due to the thermal agitation of the carrier electrons or noise due to nonlinear temperature dependent mobility in MOS transistors. The noise analysis of the sensor signals has been done by performing Fast Fourier Transform (FFT) on the sensor output signal and the linearized signal. FFT of zero level signals, viz. zero level sensor and zero level FPGA outputs are also shown in Fig. 7 which indicates that the FPGA contributes very less noise to the sensor output.

The FFT of the linearized output shown in Fig. 8 reveals that the thermistor and the measuring circuit produces low frequency or $1/f$ noise where the noise density decreases at higher frequency.

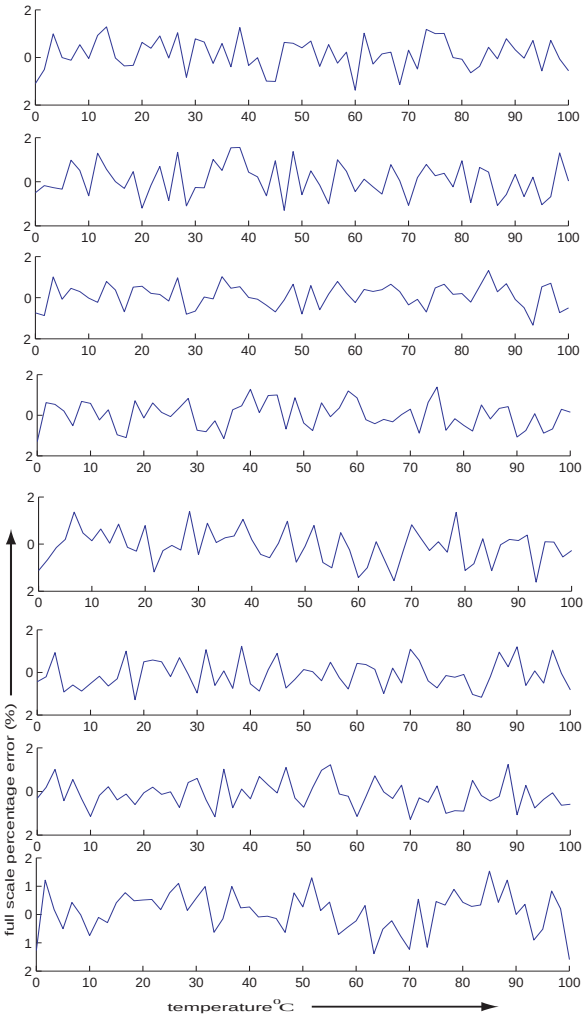

Fig. 5. Full scale percentage error in piecewise linearization with 40 segments for 8 channels from 0ºC to 100ºC.

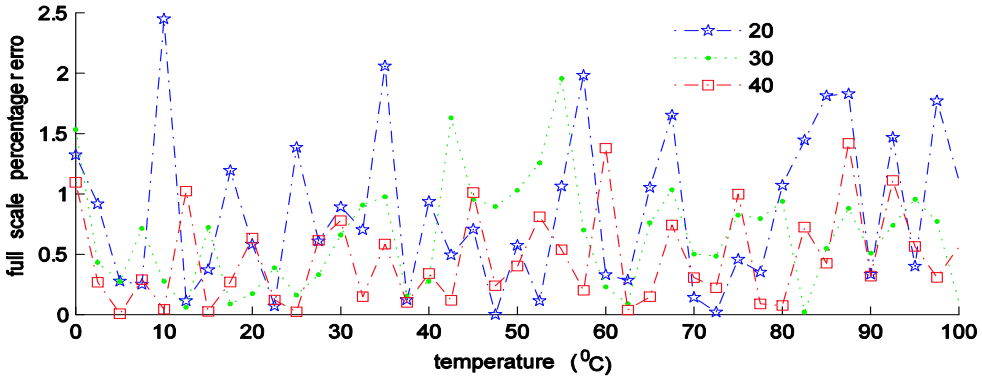

Fig. 6. Full scale percentage error in piecewise linearization with 20, 30, 40 segments for a single (CH1) channel from 0 ºC to 100 ºC.

Table 2. Error analysis of PWL in FPGA for 8 Channels with $n = 40$.

Channel	RMSE	MAD	SD	Maximum FS error, (%)
1	0.6238	0.5030	0.6237	1.5027
2	0.6562	0.5382	0.6560	1.5896
3	0.6270	0.5157	0.6259	1.4576
4	0.6523	0.5308	0.6516	1.5262
5	0.7004	0.5638	0.7004	1.5799
6	0.5646	0.4595	0.5647	1.3904
7	0.6446	0.5168	.6449	1.6245
8	0.6997	0.5655	0.6991	1.6950

Table 3. Comparative error analysis of PWL in FPGA for channel-1 with different number of piecewise segments.

Number of segments	20	30	40
RMSE	1.1391	0.8438	0.6238
MAD	0.9266	0.6903	0.503
SD	1.14	0.8445	0.6237
Maximum FS error (%)	2.5064	1.956	1.5027
No. of 4 input LUTs (out of 7168)	290 (4 %)	433 (6 %)	556 (7 %)

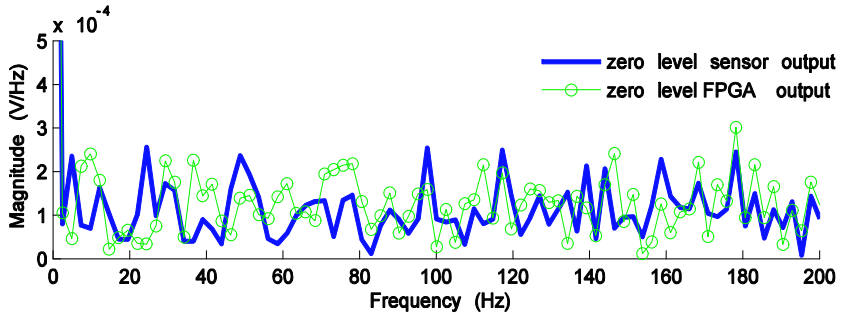

Fig. 7. FFT of zero level signal for sensor output and FPGA output.

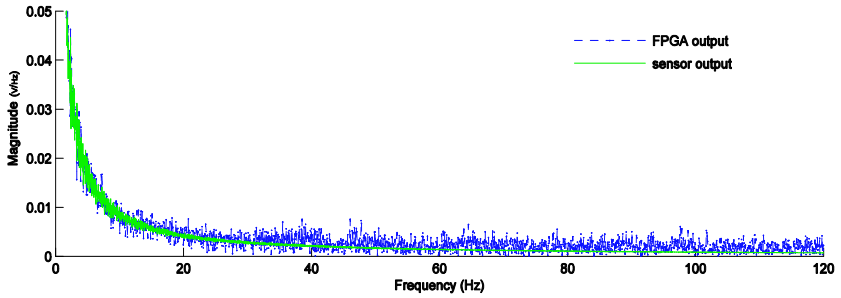

Fig. 8. FFT of sensor output and FPGA output in PWL.

It is observed that the noise density in the sensor output and the FPGA output has very little difference. However spectral leakages are observed in the linearized output which is due to random fluctuations of the mobility carriers in the electronics. The white noise density level of the signal is found in the order of 4 mV/Hz.

3.2. Linearization by Interpolation (LI)

Interpolation is a method to estimate unknown values by interpolating known data. In piecewise linear interpolation method, a set of known data points of the nonlinear sensor characteristic is considered. Any two subsequent points of the known data form a straight line in which the unknown data points can be interpolated. The number of known data points which are equally separated from the subsequent one is dependent on the accuracy of the linearization required and the length of operating range. Let us consider two known point a($V_{o,1}$; $T_{i,1}$) and b($V_{o,2}$; $T_{i,2}$) in the nonlinear thermistor characteristics shown in Fig. 9.

For any value of V_o within the range [$V_{o,1}$;$V_{o,2}$] the unknown value T_i can be determined or interpolated by using the straight line equation given as

$$T_i = T_{i,1} + (T_{i,2} - T_{i,1}) \frac{(V_o - V_{o,1})}{(V_{o,2} - V_{o,1})} \quad (4)$$

The concatenation of linear interpolants between each pair of data points of the data set ($V_{o,1}$; $T_{i,1}$); ($V_{o,2}$; $T_{i,2}$), … ($V_{o,n}$; $T_{i,n}$) form a continuous curve as shown in Fig. 9.

Initially the interpolation technique for linearization of the nonlinear thermistor is simulated in MATLAB. The data points and corresponding equations are shown in Table 4. In this work we have implemented the interpolation method in FPGA for 20, 30 and 40 sets of data points.

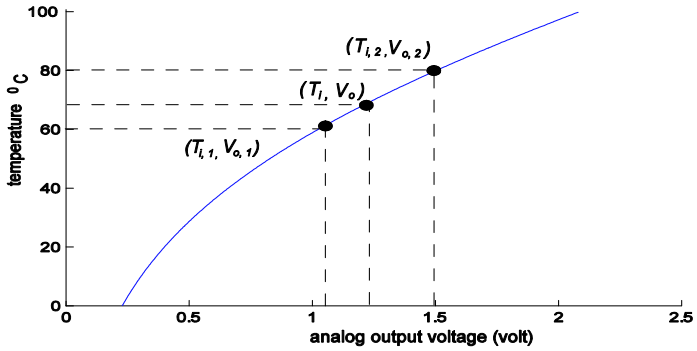

Fig. 9. Interpolation technique.

Table 4. Interpolation of Thermistor Characteristics for the Temperature Range of 0ºC - 100ºC with 20 Data Points.

Data Points	Linearized Temperature Equation
(0.1585,0) – (0.1989,5)	$T_i = 123.852V_o - 19.631$
(0.1989,5) – (0.247,10)	$T_i = 103.8084V_o - 15.6448$
(0.247,10) – (0.3039,15)	$T_i = 87.9795V_o - 11.7344$
(0.3039,15) – (0.3702,20)	$T_i = 75.4031V_o - 7.9128$
(0.3702,20) – (0.4467,25)	$T_i = 65.3598V_o - 4.1949$
(0.4467,25) – (0.5339,30)	$T_i = 57.3075V_o - 0.5982$
(0.5339,30) – (0.6323,35)	$T_i = 50.835V_o + 2.8577$
(0.6323,35) – (0.7419,40)	$T_i = 45.6279V_o + 6.1501$
(0.7419,40) – (0.8625,45)	$T_i = 41.445V_o + 9.2532$
(0.8625,45) – (0.9937,50)	$T_i = 38.1001V_o + 12.1383$
(0.9937,50) – (1.1348,55)	$T_i = 35.4491V_o + 14.7727$
(1.1348,55) – (1.2846,60)	$T_i = 33.33806V_o + 17.12$
(1.2846,60) – (1.4418,65)	$T_i = 31.8081V_o + 19.1399$
(1.4418,65) – (1.6048,70)	$T_i = 30.6654V_o + 20.7875$
(1.6048,70) – (1.772,75)	$T_i = 29.9016V_o + 22.0133$
(1.772,75) – (1.9417,80)	$T_i = 29.4786V_o + 22.7628$
(1.9417,80) – (2.1119,85)	$T_i = 29.3687V_o + 22.9762$
(2.1119,85) – (2.2811,90)	$T_i = 29.5523V_o + 22.5884$
(2.2811,90) – (2.4477,95)	$T_i = 30.017V_o + 21.5284$
(2.4477,95) – (2.6102,100)	$T_i = 30.7563V_o + 19.719$

3.2.1. FPGA Implementation of LI

The linear interpolation methodology is represented in Fig. 10. Here the range selector unit generates $T_{i,m}$, $T_{i,m+1}$, $V_{o,m}$ and $V_{o,m+1}$ for the corresponding value of V_o.

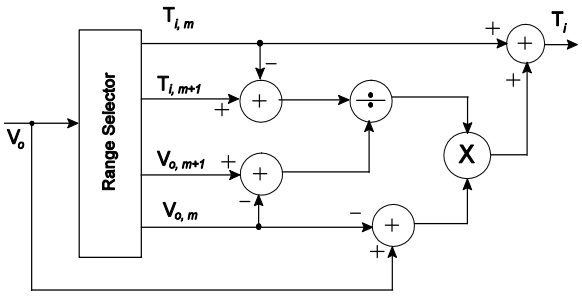

Fig. 10. Linearization by interpolation in FPGA.

For any voltage, e.g. 0.2 V, which is in the range of the data points (0.1989; 5) to (0.2470; 10); the corresponding temperature can be determined by interpolation as –

$$T_i = 5 + (10 - 5)\frac{(0.2 - 0.1989)}{(0.2470 - 0.1989)} = 5.118\,℃ \quad (5)$$

Similar to the data representation as in PWL, we have used $S4.8$ bit format to represent input (V_o) voltage and $S8.8$ bit format to represent output (T_i) and intermediate data to implement LI in FPGA.

3.2.2. Results of LI

The FS error in linearization by interpolation with $n = 40$ is shown in Fig. 11. Table 5 shows RMSE, MAD, SD, Maximum full scale error for all 8 channels with 40 segments. Table 6 shows the comparison of RMSE, MAD and SD respectively for different number of data points. A maximum FS error

of 2.6329% is observed with 20 data points while it is found to be 0.6237% for 40 data points. A RMSE of 1.1793 is observed for 20 data points, while it is 0.6238 for 40 data points. Thus a 47.1 % reduction in RMSE is observed by increasing the number of data points from 20 to 40.

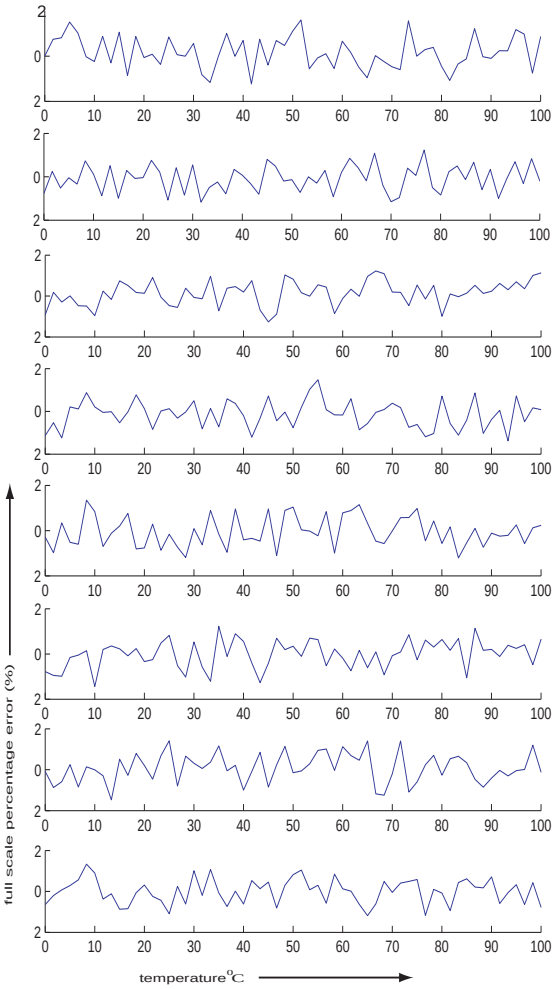

Fig. 11. Full scale percentage error in linearization using interpolation with 40 segments for 8 channels from 0 ºC to 100 ºC.

It is also observed that for 20 data points, FPGA device uses 290 elements of 4-input LUTs; while it is 433 and 566 for 30 and 40 data points respectively, i.e. 4 %, 6 % and 8 % for 20, 30 and 40 data points respectively. An improvement in accuracy can be observed when data points are increased from 20 to 40. Also, the LUT address size increases from 9 bits to 10 bits as the number of data points increases from 20 to 40.

Table 5. Error analysis of LI in FPGA for 8 channels with n=40.

Channel	RMSE	MAD	SD	Maximum FS error (%)
1	0.7040	0.5755	0.7040	1.6308
2	0.5780	0.4758	0.5784	1.3946
3	0.5959	0.4833	0.5963	1.4736
4	0.6321	0.5165	0.6326	1.4750
5	0.6444	0.5245	0.6441	1.5671
6	0.6696	0.5443	0.6702	1.4817
7	0.7095	0.5788	0.7084	1.5907
8	0.5788	0.4632	0.5781	1.4335

Table 6. Comparative error analysis of LI in FPGA for Channel-1 with different Number of data points.

Number of segments	20	30	40
RMSE	1.1793	0.8262	0.6238
MAD	0.9621	0.679	0.503
SD	1.1784	0.8268	0.6237
Maximum FS error (%)	2.6329	2.0191	1.5027
No. of 4 input LUTs (out of 7168)	290 (4 %)	433 (6 %)	556 (7 %)

The FFT analysis in Fig. 12 shows that the FPGA system has incorporated with higher spectral leakage however the white noise level is same as that of PWL method.

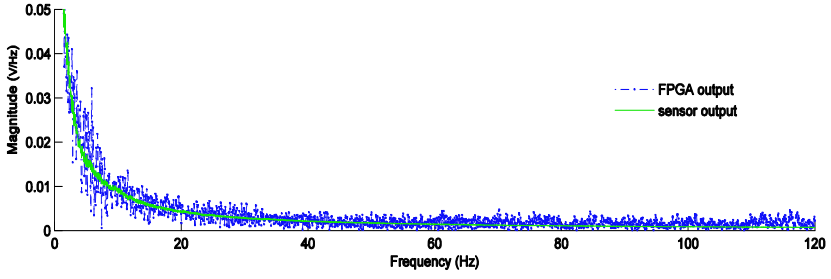

Fig. 12. FFT of sensor output and FPGA output in LI.

3.3. Look up Table (LUT) Based Linearization

A look up table (LUT) provides an interpretation between input-output values for any nonlinear system. They are used to realize a wide variety of nonlinear functions [9]. Thus a LUT can be used for mapping the input-output characteristics of the thermistor. Here, the outputs of the voltage divider circuit and the corresponding temperatures are stored

in memory in a tabular form. The output of the ADC is applied as input to the linearization unit online, that addresses the corresponding linearized temperature.

3.3.1. FPGA Implementation of LUT Based Linearization

LUTs can be implemented in simplest controllers, but it requires much higher program memory.

Typically f^{-1} is implemented in FPGA using the block RAMs of the device; however add-on memory devices like flash memory can be used to implement complex nonlinear functions with higher degree of nonlinearity and with higher precision. In this case y in Fig. 2 is 8 bit data from ADC and used as pointer to the memory locations. The FPGA architecture for implementation of a LUT linearization can be realized as shown in Fig. 13. The 8 bit ADC output which can address a maximum of 256 (2^8) data is used as pointer to the memory locations of LUT where $f^{-1}(y)$ is stored.

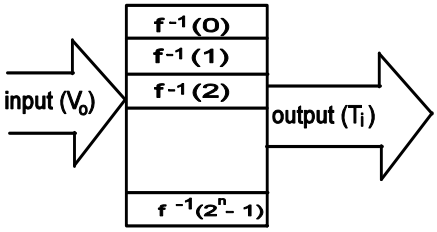

Fig. 13. Look up table based linearization.

3.3.2. Data Representation

In this method, the 8-bit ADC outputs (V_o) address the LUT data in FPGA. The LUT output data (T_i) are represented using $S8.8$ fixed point format. For a maximum temperature range of 100ºC, the 8 bit fractional part provides a resolution of 2^{-8} i.e. 3.9×10^{-3}ºC, while the microcontroller based LUT in [9] would provide only a resolution of 100×2^{-3}ºC, i.e. 3ºC. In this work we have used 128 and 256 LUT data for the temperature range of 0ºC - 100ºC.

3.3.3. Results of LUT Based Linearization

The errors in linearization for different numbers of LUT data elements are shown in Fig. 14. It is observed that in case of 128 data elements maximum FS error is 1.5214%, while it is 1.1427 % in case 256 LUT data elements. Similarly RMSE, MAD and SD are shown in Table 7. The results indicates a 27% reduction in RMSE when the number of LUT data elements are increased from 128 to 256 and this increase in LUT data elements consumes 14% to 35% more logic area. It may be mentioned that unlike PWL and LI, in LUT based technique, linearization units were implemented for only 3 and 2 channels with 128 and 256 LUT data respectively in Spartan-III XC3S400 FPGA. However, number of channels can be increased by using high capacity FPGAs.

Table 7. Comparative error analysis of LUT based Linearization in FPGA for Channel-1 with different LUT sizes.

Number of LUT DATA	**128**	**256**
RMSE	0.6477	0.4692
MAD	0.3172	0.2311
SD	0.3794	0.2742
Maximum FS error (%)	1.5214	1.2427
No. of 4 input LUTs (out of 7168)	975 (14 %)	2500 (35 %)

The FFT of the sensor output and FPGA output of LUT based linearization are shown in Fig. 15. In this case the spectral density of noise is found to be higher than that of PWL and LI.

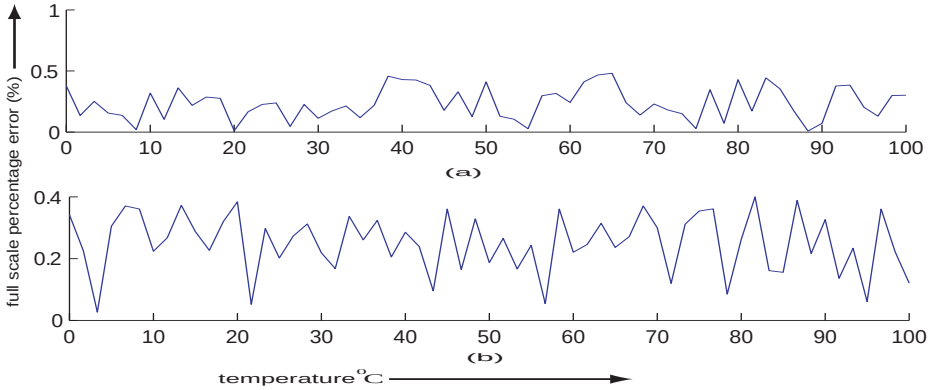

Fig. 14. Full scale percentage error in LUT based linearization for (a) 128 and (b) 256 LUT data from 0 ºC to 100 ºC.

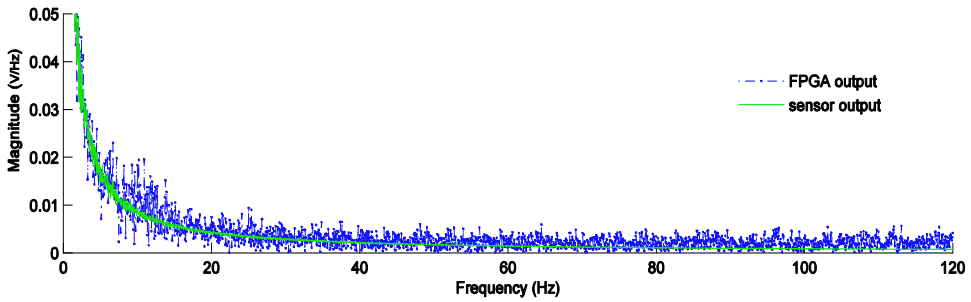

Fig. 15. FFT of sensor output and FPGA output in LUT.

3.4. ANN Based Linearization

There are different architectures of ANN for mapping nonlinear input-output functions. Based on the degree of nonlinearity of the original function, there may be different numbers of layers or different numbers of neurons in hidden layer. N. J. Medrano-Marques and B. Martin-del-Brio in [19] proposed an optimized architecture of ANN for linearization of thermistor. The network is a MLP neural network consisting only two neurons in hidden layer and one neuron in output layer as shown in Fig. 16. They have simulated the network for implementation in low cost processors like microcontrollers. In this work, an array of 4 such ANN structures has been implemented in FPGA for the 4 channels of sensor signals.

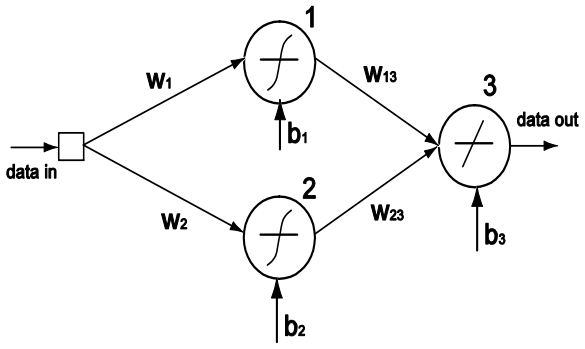

Fig. 16. An artificial neural network for sensor linearization.

We have taken the activation function of the hidden layer neurons as a hyper tangent activation function given by-

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

whereas the output neuron has linear activation function. The network is initially trained in MATLAB using input-output data set given in Equation (3). The weight and bias of the ANN are determined during training by simulating the network in MATLAB. The training parameters obtained are shown in Table 8.

Table 8. Matlab simulation results of ANN for Linearization Channel-1.

Network Type	**MLP Feed Forward Netwrok**			
Activation Function	**Neuron1**	**Neuron2**	**Neuron3**	
	$f(x) = tanh(x)$	$f(x) = tanh(x)$	$f(x) = x$	
Training Algorithm	Back Propagation			
Training data size	600			
Learning rate	2.0			
Goal	10^{-6}			
Epochs	1000			
Weights	$W1$	$W2$	$W13$	$W23$
	-0.026	1.284	-87.348	159.44
Bias	$b1$	$b2$	$b3$	
	1.195	2.733	-85.909	

3.4.1. FPGA Implementation of a Neuron

The processing element of an ANN is the neuron which can be expressed by an equation of the form, $y_i = \sum(w_i x_i + b_i)$, where x, w and b represent input, weight and bias respectively. Fig. 17 shows the FPGA implementation of a neuron which includes, multiplier, adder and activation function.

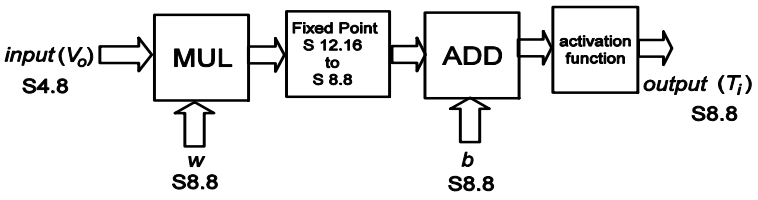

Fig. 17. Block Diagram of FPGA implementation of a neuron.

3.4.2. Implementation of Activation Function in FPGA

Efficient implementation of the nonlinear activation function on an FPGA is a difficult task faced by designers as it is not suitable for direct implementation [35-36]. In most cases computationally simplified alternatives of nonlinear activation function are used [37-38]. In this work we have used Piece- Wise Linear Approximation (PWL) method for implementation of the $tanh(x)$ activation function for neurons 1 and 2. The hyper tangent curve has been segmented into 10 linear segments as shown in Fig. 18. The piecewise linear equations for each segment are shown in Table 9. The Number of linear pieces can be increased to achieve the precision of the activation function at the cost of memory and computational complexity [39].

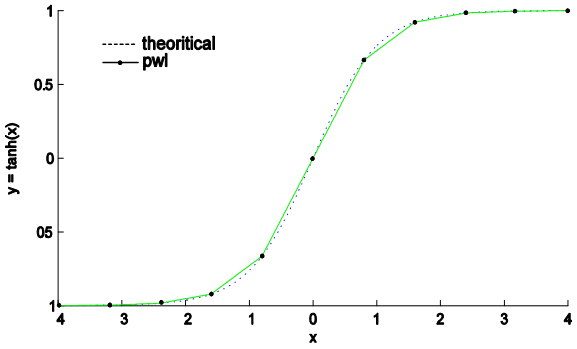

Fig. 18. Piecewise linear approximation of $y = tanh(x)$ function.

Table 9. Approximation of $y = tanh(x)$ function in FPGA.

Range	Equation
[-5,-4.1]	$y = 0.000x - 0.997$
[-4,-3.1]	$y = 0.003x - 0.985$
[-3,-2.1]	$y = 0.026x - 0.918$
[-2,-1.1]	$y = 0.177x - 0.626$
[-1,-0.1]	$y = 0.739x - 0.064$
[0,1]	$y = 0.768x + 0.044$
[1,2]	$y = 0.177x + 0.626$
[2,3]	$y = 0.026x + 0.918$
[3,4]	$y = 0.003x + 0.985$
[4,5]	$y = 0.000x + 0.997$

In ANN based linearization in FPGA input voltage (V_o) is represented using S4.8 bit fixed point format, while temperature (T_i), weight (w) and bias (b) are represented using S8.8 bit fixed point format.

3.4.2. Results of FPGA Implementation of ANN for Linearization

ANNs have been implemented in FPGA for linearization of the nonlinear thermistors for the range 0 ºC to 100 ºC as discussed above. The networks implemented here are a MLP networks in which 600 data sets are used to train each of the networks with back propagation algorithm. The result in Fig. 19 shows that ANN can perform linearization with better accuracy as compared to the other methods discussed above.

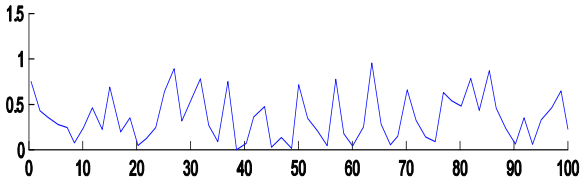

Fig. 19. Full scale percentage error of ANN based linearization for 4 channels from 0 ºC.

As shown in Table 10,a maximum FS error of 1.241 % is observed in ANN based linearization with average logic utilization of 19 % per channel and 89 % for 4 channels.

Table 10. Error analysis of ANN based linearization in FPGA for 4 channels.

Channel	RMSE	MAD	SD	Maximum FS error (%)
1	0.4805	0.2376	0.2857	1.1901
2	0.4549	0.2195	0.2707	1.1981
3	0.4958	0.2357	0.2811	1.1914
4	0.4751	0.224	0.2709	1.2410

The FFT of the sensor output and FPGA output in Fig. 20 of ANN based linearization shows that the noise voltage fluctuation is less severe than that of LUT based linearization and similar to LI, however higher than that of PWL based linearization.

Apart from analyzing the error incorporated in linearization process, we have also tested the dynamic response of the linearization unit. Since temperature variation in a thermistor itself is a slow process, therefore to test the speed of operation of the linearization unit we have used a load cell of capacity 100 kg as input sensor that is interfaced with FPGA through the ADC. The signal obtained from the Wheatstone bridge configured load cell has been linearized by using LUT based linearization technique. The response obtained from the load cell were mapped to the linearized output using a LUT of 150 data points in the full scale rang of 100 kg of the load cell. A random load up to 90 kg was applied and unloaded to test the speed of operation of the FPGA. Fig. 21 shows the load cell response and FPGA output for loading and unloading for a time duration of 8 sec. From the Fig. 21 for testing speed of operation of the FPGA system, it is observed that the sharp loading and unloading of load cell is processed by FPGA with a very little amount of time delay

error. Further, for quantification of the speed of response of the FPGA system, a fast varying analog input has been applied to the ADC and the FPGA processed output in LUT based linearization through the DAC has been recorded as shown in Fig. 22. With an ADC conversion time of 100 µs per channel (for a clock frequency of 200 kHz), DAC settling time of about 150 ns and FPGA execution time of 90 ns, the maximum speed of linearization per channel was estimated as 100.24 µs (100 µs + 150 ns + 90 ns). For 8 channels this figure will be 801.29 µs (100 µs×8 + 150 ns×8 + 90 ns). This gives a maximum frequency of 9.976 kHz for a single channel while 1.248 kHz for 8 channels.

A linearity plot for the four types of linearization methods is also shown in Fig. 23 along with the nonlinear thermistor response.

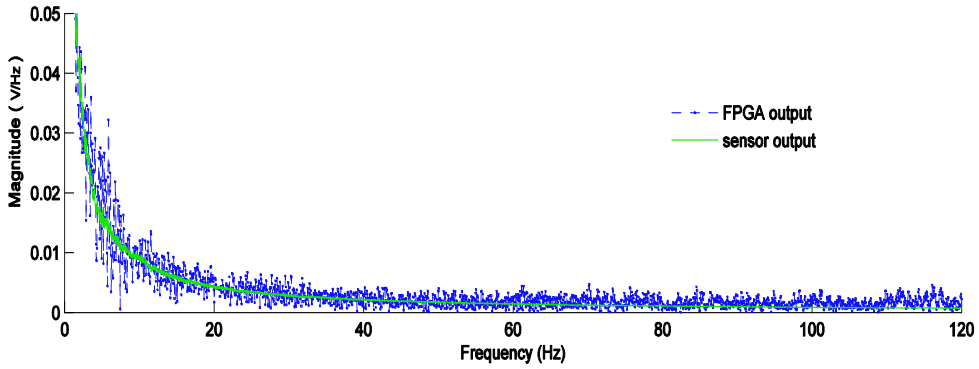

Fig. 20. FFT of sensor output and FPGA output in ANN based linearization.

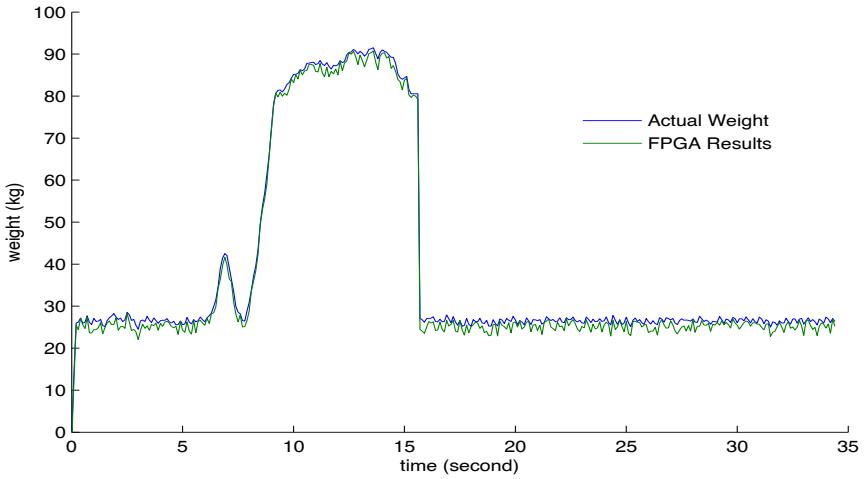

Fig. 21. Signals showing fast variation of load applied to a load cell and corresponding FPGA based linearized (LUT) output

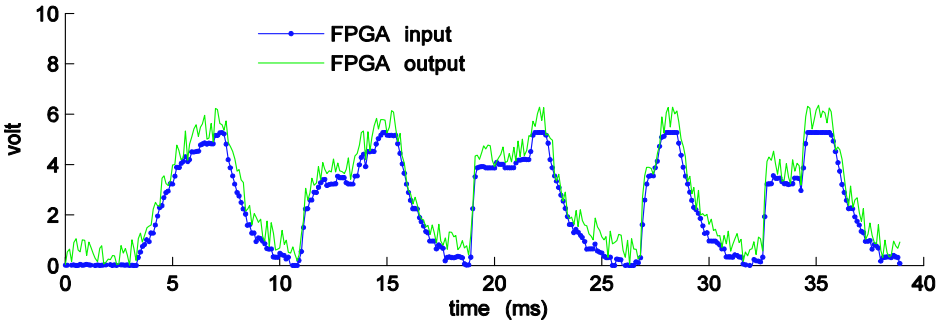

Fig. 22. Signals showing fast variation of an analog signal and corresponding FPGA based linearized (LUT) output.

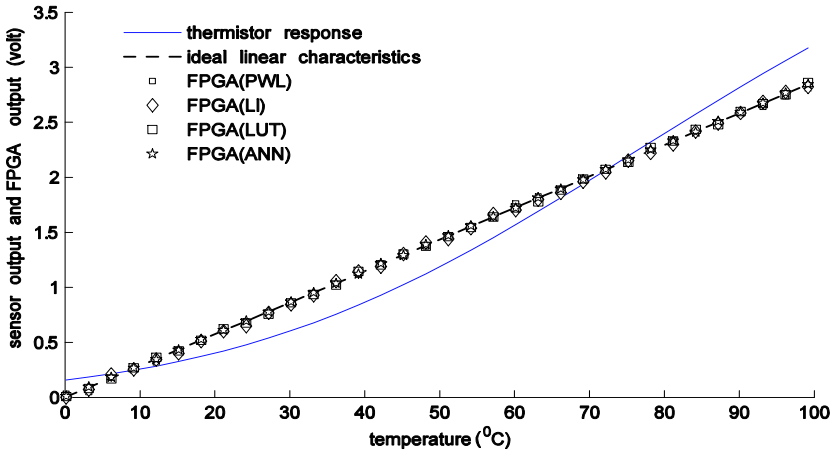

Fig. 23. Linearity plot for all 4 methods of linearization in FPGA.

4. Conclusions

In this work we have implemented the four different techniques of thermistor linearization in FPGA for simultaneous 8-channel sensor linearization. We have tested the techniques using real time signals from thermistors using LabVIEW data acquisition system. The performance of each of the linearization scheme has been estimated considering mainly accuracy, speed of operation and cost of implementation. In the result sections of each implementation, we have shown full scale percentage error, RMSE, MAD and SD by comparing with different numbers of data sets.

A comparative result of all the four methods of linearization is shown in Table 11. It is observed that ANN is showing better accuracy, while LUTs consumes highest logic area of FPGA device. The piecewise linearization and interpolation consumes almost same logic area in FPGA, but in case of linear interpolation the execution time of the linearization block is higher than that of PWL. While in case of ANN based implementation, the logic area utilization is almost 19 % as compared to 7 % in case of PWL and LI. It can be estimated that for same amount of full scale error of 1.19 % in ANN; PWL and LI would utilize logic area of about 8 % and LUT based linearization would utilize 36 %.

Table 11. Comparison of four methods of linearization in FPGA.

	PWL	LI	LUT	ANN
RMSE	0.6238	0.6238	0.4692	0.4805
MAD	0.503	0.503	0.2311	0.2376
SD	0.6237	0.6237	0.2742	0.2857
Execution Time	120 ns	125 ns	90 ns	140 ns
FPGA LUTsused (%)	7 %	7 %	35 %	19 %
FS error (max)	1.5 %	1.5 %	1.24 %	1.19 %

Thus PWL and LI are better option for this case. Further, the implementation complexity is higher in case of ANN and the accuracy of the method is dependent on several steps of implementations - e.g. accuracy of activation function. However cost of implementation is less in case of ANN as compared to LUT. The FPGA implementation using LUT is the simplest method to map the nonlinear thermistor characteristics in comparison to the other three methods. From the point of execution time PWL, LI, LUT outperforms ANN based linearization technique. On the other hand, in spite of lower performance functions, ANN can be improved considerably due to its learning capability for higher degree of nonlinearity.

The FPGA based linearization system has been tested with fast varying analog inputs from load cell and analog inputs. It has been observed that the FPGA based linearization can handle the fast varying signals also. It is estimated that the system can linearize analog signals with a maximum frequency of 9.976 kHz for a single channel while 1.248 kHz for 8 channel operation.

The proposed FPGA based linearization techniques are suitable for systems with multiple sensor inputs as FPGA devices are suitable for multi-channel operation with parallel architecture with high speed signal processing capability.

Acknowledgements

The authors of this paper would like to thank Tezpur University, Assam, India for partially supporting the work under start up grant scheme.

References

[1]. M. Bhuyan, Intelligent Instrumentation: Principles and Applications, *CRC Press*, 2010.

[2]. G. Iglesias, E. Iglesias, Linearization of transducer signals using an analog-to-digital converter, *IEEE*

Transactions on Instrumentation and Measurement, Vol. 37, No. 1, 1988, pp. 53–57.

[3]. J. Lygouras, Nonlinear ADC with digitally selectable quantizing characteristic, *Transactions on Nuclear Science, IEEE*, Vol. 35, No. 5, 1988, pp.1088-1091.

[4]. D. Anvekar, B. S. Sonde, A programmable nonlinear ADC using optimal-sized rom, *IEEE Transactions on Instrumentation and Measurement*, Vol. 40, No. 6, 1991, pp. 1031–1035.

[5]. E. Volpi, N. Nizza, P. Bruschi, A non linear ADC for sensor linearization, in *Proceedings of theInternational Conference on PhD Research in Microelectronics and Electronics (PRIME'07)*, 2007, pp. 5–8.

[6]. S. Kaliyugavaradan, P. Sankaran, V. Murti, A new compensation scheme for thermistors and its implementation for response linearization over a wide temperature range, *IEEE Transactions on Instrumentation and Measurement*, Vol. 42, No. 5, 1993, pp. 952–956.

[7]. Z. Nenova, T. Nenov, Linearization circuit of the thermistor connection, *IEEE Transactions on Instrumentation and Measurement*, Vol. 58, No. 2, 2009, pp. 441–449.

[8]. J. Patra, G. Panda, R. Baliarsingh, Artificial neural network based nonlinearity estimation of pressure sensors, *IEEE Transactions on Instrumentation and Measurement*, Vol. 43, No. 6, 1994, pp. 874–881.

[9]. L. E. Bengtsson, Lookup table optimization for sensor linearization in small embedded systems, *Journal of Sensor Technology*, Vol. 2, 2012, pp. 177-184.

[10]. N. Cotton, B. Wilamowski, Compensation of nonlinearities using neural networks implemented on inexpensive microcontrollers, *IEEE Transactions on Industrial Electronics*, Vol. 58, No. 3, 2011, pp. 733–740.

[11]. N. Khachab, M. Ismail, Linearization techniques for nth-order sensor models in MOSVLSI technology, *IEEE Transactions on Circuits and Systems*, Vol. 38, No. 12, 1991, pp. 1439–1450.

[12]. J. Patra, G. Chakraborty, P. Meher, Neural-network-based robust linearization and compensation technique for sensors under nonlinear environmental influences, *IEEE Transactions on Circuits and Systems I: Regular Papers*, Vol. 55, No. 5, 2008, pp. 1316–1327.

[13]. H. Erdem, Implementation of software-based sensor linearization algorithms on low-cost microcontrollers, *ISA Transactions*, Vol. 49, No. 4, 2010, pp. 552–558.

[14]. D. Sonowal, M. Bhuyan, FPGA implementation of neural network for linearization of thermistor characteristics, in *Proceedings of the International Conference on Devices, Circuits and Systems (ICDCS'12)*, March 2012, pp. 422–426.

[15]. D. Sonowal, M. Bhuyan, Linearizing thermistor characteristics by piecewise linear interpolation in real time FPGA, in *Proceedings of the2nd International Conference on Advances in Computing, Communications and Informatics (ICACCI'13)*, August 2013, pp. 1976–1980.

[16]. D. Patranabis, D. Ghosh, A novel software-based transducer linearizer, *IEEE Transactions on Instrumentation and Measurement*, Vol. 38, No. 6, 1989, pp. 1122–1126.

[17]. J. Sturcel, M. Kamensky, Function approximation and digital linearization in sensor system, *AT&P Journal*, Vol. 13, No. 7, 2006, pp. 81-82.

[18]. M. Attari, F. Boudjema, M. Heniche, Linearizing a thermistor characteristic in the range of zero to 100 degree C with two layer artificial neural networks, in *Proceedings of theIEEE Instrumentation and Measurement Technology Conference on Integrating Intelligent Instrumentation and Control (IMTC'95)*, 1995, pp. 119-122.

[19]. N. Medrano-Marques, M. del Brio, A general method for sensor linearization based on neural networks, in *Proceedings of theIEEE International Symposium on Circuits and Systems, (ISCAS'2000)*, Geneva, Vol. 2, 2000, pp. 497–500.

[20]. N. Medrano-Marques, B. Martin-del-Brio, Sensor linearization with neural networks, *IEEE Transactions on Industrial Electronics*, Vol. 48, No. 6, 2001, pp. 1288-1290.

[21]. S. A. Khan, D. Shahani, A. Agarwala, Sensor calibration and compensation using artificial neural network, *ISA Transactions*, Vol.42, No. 3, 2003, pp. 337-352.

[22]. X. Wei, Sensor temperature compensation technique simulation based on BP neural network, *TELKOMNIKA*, Vol. 11, No. 6, 2013, pp. 3304-3313.

[23]. D. K. Ghara, D. Saha, K. Sengupta, Implementation of linear trace moisture sensor by nano porous thin film moisture sensor and NLAmp, *International Journal on Smart Sensing and Intelligent Systems*, Vol. 1, No. 4, December 2008, pp. 955-968.

[24]. A. Muthuramalingam, S. Himavathi, E. Srinivasan, Neural network implementation using FPGA: Issues and application, *International Journal of Information Technology*, Vol. 4, No. 2, 2008, pp. 86–92.

[25]. N. Botros, M. Abdul-Aziz, Hardware implementation of an artificial neural network using field programmable gate arrays (FPGA's), *IEEE Transactions on Industrial Electronics*, Vol. 41, No. 6, 1994, pp. 665-667.

[26]. N. Botros, M. Abdul-Aziz, Hardware implementation of an artificial neural network, in *Proceedings of the IEEE International Conference on Neural Networks*, Vol. 3, 1993, pp.1252-1257.

[27]. X. Yu, D. Dent, Implementing neural networks in FPGAs, in *Proceedings of theEE Colloquium on Hardware Implementation of Neural Networks and Fuzzy Logic*, Vol. 61, 1994, pp. 1/1–1/5.

[28]. J. Zhu, G. J. Milne, B. K. Gunther, Towards an FPGA based reconfigurable computing environment for neural network implementations, in *Proceedings of theNinth International Conference on Artificial Neural Networks (ICANN'99)*, (Conf. Publ. No. 470), Vol. 2, 1999, pp. 661-666.

[29]. Y. Chen, W. du Plessis, Neural network implementation on a FPGA, in *Proceedings of the IEEE 6th Africon Conference in Africa (AFRICON'02)*,Vol. 1, 2002, pp. 337–342.

[30]. J. M. Granado, M. A. Vega, R. Perez, J. Sanchez, J. Gomez, Using FPGAs to implement artificial neural networks, in *Proceedings of the13th IEEE International Conference on Electronics, Circuits and Systems (ICECS' 06)*, 2006, pp. 934-937.

[31]. J. Liu, D. Liang, A survey of FPGA-based hardware implementation of ANNs, in *Proceedings of the International Conference on Neural Networks and Brain (ICNN B'05)*,Vol. 2, 2005, pp. 915-918.

[32]. S. Himavathi, D. Anitha, A. Muthuramalingam, Feed-forward neural network implementation in FPGA using layer multiplexing for effective resource utilization, *IEEE Transactions on Neural Networks*, Vol. 18, No. 3, 2007, pp. 880-888.

[33]. E. Monmasson, L. Idkhajine, M. Cirstea, I. Bahri, A. Tisan, M.-W. Naouar, FPGAs in industrial control applications, *IEEE Transactions on Industrial Informatics*, Vol. 7, No. 2, 2011, pp. 224-243.

[34]. A. Gomperts, A. Ukil, F. Zurfluh, Development and implementation of parameterized FPGA-based general purpose neural networks for online applications, *IEEE Transactions on Industrial Informatics*, Vol. 7, No. 1, 2011, pp. 78–89.

[35]. M. Tommiska, Efficient digital implementation of the sigmoid function for reprogrammable logic, *IEE Proceedings on Computers and Digital Techniques*, Vol. 150, No. 6, 2003, pp. 403–411.

[36]. K. Basterretxea, J. Tarela, I. Del Campo, Approximation of sigmoid function and the derivative for hardware implementation of artificial neurons, *IEE Proceedings Circuits, Devices and Systems*, Vol. 151, No. 1, 2004, pp. 18-24.

[37]. A. Mishra, Z. Zaheeruddin, K. Raj, Implementation of a digital neuron with nonlinear activation function using piecewise linear approximation technique, in *Proceedings of the International Conference on Microelectronics (ICM' 07)*, 2007, pp. 69-72.

[38]. A. Tisan, S. Oniga, D. Mic, A. Buchman, Digital implementation of the sigmoid function for FPGA circuits, *ACTA Technica Napocensis – Electronics and Telecommunication*, Vol. 50, No. 2, 2009, pp. 15-20.

[39]. Z. Xie, A non-linear approximation of the sigmoid function based on FPGA, in *Proceedings of theIEEE 5th International Conference on Advanced Computational Intelligence (ICACI' 12)*, 2012, pp. 221–223.