Electronic Letters on Science & Engineering 17(2) (2021)
Available online at dergipark.org.tr/else

An Implementation of Linear Regression using IP-Core and FPGA-based Microcomputer Architecture

Abdelkader Lazzem[1*], Halit Öztekin[1], Souad Cheurfi [3]

[1]Department of Electrical-Electronics Engineering, Sakarya University of Applied Sciences, Sakarya, Turkey
y190004033@subu.edu.tr, ORCID: 0000-0003-0136-356X,
halitoztekin@subu.edu.tr, ORCID: 0000-0001-8598-4763
[2] Department of Computer Engineering, Sakarya University of Applied Sciences, Sakarya, Turkey
d190004008@subu.edu.tr, ORCID: 0000-0003-4053-3971

Abstract: To generate more accurate results, machine learning approaches, particularly those based on neural networks, require the usage of accurate real values. Linear regression is a machine learning technique that is commonly used to identify the linear function that best fits a set of data. Due to current trends in systems need and the availability of Field-Programmable Gate Array (FPGA), floating-point implementations are becoming more widespread, and engineers are increasingly using FPGAs as a platform for floating-point implementations.This paper demonstrates the FPGA-based half-precision floating-point (FPU-16) implementation by proposing two different ways of linear regression implementation. The first way uses the assembler of BZK.SAU.FPGA-based microcomputer architecture. The second way uses the IP-Core of Xilinx simulated and tested with Vivado Design Suite software. After implementing both ways we have calculated the Mean Square Error MSE between the results and it was found to be equal to 7.73×10^{-4}.

Keywords: Artificial Neural Network (AAN), Machine Learning (ML) , Half-precision Floating-point (FPU-16), Linear Regression, Field-programmable gate array (FPGA), Assembler.

IP-Çekirdek ve FPGA-tabanlı Mikrobilgisayar Mimarisi kullanılarak Doğrusal Regresyon Uygulaması

Özet: Daha doğru sonuçlar elde etmek için, özellikle sinir ağlarına dayalı makine öğrenimi yaklaşımları, doğru gerçek değerlerin kullanılmasını gerektirir.Doğrusal regresyon, bir veri kümesine en uygun doğrusal fonksiyonu tanımlamak için yaygın olarak kullanılan bir makine öğrenimi tekniğidir.Sistem ihtiyacındaki mevcut eğilimler ve Alanda Programlanabilir Kapı Dizisinin (FPGA) kullanılabilirliği nedeniyle, kayan nokta uygulamaları giderek yaygınlaşıyor ve mühendisler kayan nokta uygulamaları için bir platform olarak FPGA'ları daha fazla kullanıyorlar. Bu makalede, doğrusal regresyon uygulamasının iki farklı yolu önerilerek FPGA tabanlı yarı duyarlıklı kayan nokta (FPU-16) uygulaması göstermektedir. İlk yol BZK.SAU.FPGA tabanlı mikrobilgisayar mimarisinin assembly dilini kullanır. İkinci yol Vivadi Design Suite yazılımıyla simüle edilmiş ve test edilmiş Xilinx'in IP çekirdeğini kullanır.Her iki yöntemi uyguladıktan sonra, aralarındaki Ortalama Kare Hata MSE'yi hesapladık ve sonucun 7.73×10^{-4} olarak bulduk.

Anahtar Kelimeler: Yapay Sinir Ağı (YSA), Makine öğrenmesi (MÖ), Yar-hassas Kayan-Nokt (FPU-16), Doğrusal Regresyon, Alanda programlanabilir kapı dizisi (APKD), Assembler.

Reference to this paper should be made as follows (bu makaleye aşağıdaki şekilde atıfta bulunulmalı):
Lazzem A., Öztekin H., Cheurfi S., 'An Implementation of Linear Regression using IP-Core and FPGA-based Microcomputer Architecture', Elec Lett Sci Eng, vol. 17(2), (2021),53-62.

1. Introduction

Since the invention of the first electronic component, electronics have continued to develop and spread all over the world which has led to the emergence of technology and its development. In view of current technological advances, the acceptance of Artificial Intelligence (AI) as an important technology and Its use in large-scale projects is increasing day by day.The ability to use micro-services containing AI greatly increases flexibility in implementing this technology in existing projects. Therefore, the development of AI has made Machine Learning techniques essential, which raises a question of how we can deploy these algorithms on embedded systems.

In order to implement these algorithms on hardware, the researchers consider all options for creating devices with the required accuracy and efficiency, as well as a manageable power budget. The hardware implementations are used on hardware devices such as Application Specific Integrated Circuits (ASIC), Graphics processing unit (GPU), FPGA , Microcontrollers, and so on.where each has its own advantages and disadvantages.

Many studies on the hardware implementation of AI algorithms have been published in this area. Manar Abu Talib et al. [1] have discussed literature reviews covering related works. They summarized the results of the collected research papers in a pie chart, as is shown in Fig. 1.

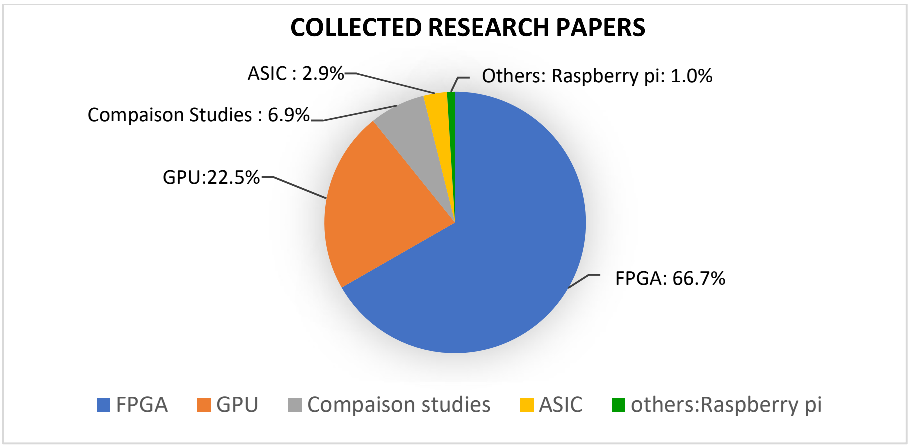


Fig. 1.Comparison between hardware devices [1].

From the pie chart, we can observe that most of the research was performed on FPGA. Because of its very suitable architecture which offers the ability to implement these kinds of algorithms, higher energy efficiency compared to the other devices, rapidity, and lower energy consumption. Ferreira et al [5] presented a fixed-point arithmetic linear regression algorithm implementation by using FPGA. VHSIC Hardware Description Language (VHDL) was the hardware design language used and implemented on the Altera DE2-115 development board. They used 8 training data points as samples and only one clock cycle was needed to implement their proposed algorithm. Lopes et al [12] proposed an FPGA -based fully parallel Support vector machine SVM using Stochastic Gradient Descent as a training method. where it increased the speed of implementation in comparison with relative software implementations and hardware implementations and requiring fewer hardware resources.Grout et al [13] presented an implementation of a 3-D multiple linear regression algorithm in hardware using FPGA. The algorithm was modeled and checked by using Python, NumPy and Matplotlib then converted to VHDL and implemented on Xilinx Spartan-3AN.

BZK.SAU.FPGA microcomputer architecture was proposed by Halit Öztekin [7] one of the authors of this paper and it has been used at the Computer Engineering Department of Sakarya and Yozgat Bozok Universities in Turkey since 2011. It also began to be used by Sakarya University of Applied Sciences during this year 2021. It was designed as a reconfigurable hardware educational tool to enhance the students' learning experience in hardware architecture . It is an FPGA-based modular logic gate architecture [6] based on the modularity of computer architecture. It has the property of reconfigurability without causing any downside to the system operation.

In this study, we will implement FPGA-based linear regression which represents one of the machine learning algorithms that uses FPU-16 in two different ways ; by using BZK.SAU.FPGA-based microcomputer architecture, by using IP-Core of Xilinx simulated and tested on Vivado software.

This study aims to illustrate the usability of BZK.SAU.FPGA microcomputer architecture in such applications by comparing its obtained results with the same application, but using another commonly used hardware architecture. where we can take advantage of its properties, especially that it is intended for educational purposes.

The rest of this paper is organized as follows. in Section 2,3,4 linear regression algorithm, Floating-Point Numbers, and BZK.SAU.FPGA are described respectively. In Section 5 the used Methodology is explained. In Section 6 the implementation of both proposed ways is presented. In section 7 evaluation of the Implementation of both ways is presented. In Section 8 the obtained Results are shown and discussed. In Section 9 the conclusion of this study is given.

2. Linear Regression

Linear regression is a supervised learning algorithm used in Machine Learning that involves finding the best function to define the linear relationship between an output variable y and an input variable x. According to the number of variables of x, the linear regression is divided into two types. With a single input, it's called a simple regression, and with more than one input, it's called a multiple regression [8]. The modeling of this algorithm graphically is a question of finding the best possible line that can explain a model (x, y). The advantage of the linear regression algorithm is its ease of interpretation and ease of calculation.

In this research, we will use simple linear regression. So, the function to look for is on the form: $y = a2 + x\, a1$, where a1 and a2 respectively represents the coefficients of the slope and the value at index 0 to be estimated [11]. Since we are going to use several sets (x, y) as data to estimate the values of the coefficients, the function becomes in a matrix form:

$$Y = \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix} = \begin{pmatrix} 1 & x_1 \\ \vdots & \vdots \\ 1 & x_n \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \end{pmatrix} \qquad (1)$$

The fundamental basis for the solution of linear systems is the concept of Matrices and Vectors in Linear Algebra. The most significant expression used to calculate the results of Linear Regression is one of the optimization techniques known as the Least-Square approximation. To obtain optimized results, the coefficient values must be changed to obtain the closest values to the original outputs. We use the ordinary least squares method for this. The matrix expression for least-square solution to find the value of a1 and a2 is shown in the equation below. The suggested technique for implementing this function on FPGA is shown in the methodology section.

$$\begin{pmatrix} a_1 \\ a_2 \end{pmatrix} = (X^T\, X)^{-1}\, X^T Y \qquad (2)$$

3. Floating-Point Numbers

A floating-point is a method of encoding and storing real numbers in electronic devices. It allows representing approximately a part of the real numbers that can be small or too large [2]. Floating-point numbers have the advantage of maintaining precision over a large dynamic range, while fixed-point numbers lose precision [3]. Floating-point numbers are composed of three parts: the sign, the exponent, and the mantissa. Two standard formats were released to represent this type of number: IEEE 754 (the most used format), and IEEE 854.

The IEEE 754 standard format [4] has basic and interchange formats . The basic formats are three binary formats (Single precision (32 bits), Double precision (64 bits),Quadruple precision (128 bits)). And two decimal formats(64 and 128 bits). The interchange formats are divided into binary interchange formats of range 16, 32, 64, and 128 bits. The decimal interchange formats are defined for any multiple of 32 bits of at least 32 bits . In the study, the preferred format is IEEE 754 FPU-16 as shown in Fig. 2. below. That is because we are going to use BZK.SAU.FPGA which has 16-bits registers microcomputer architecture.In order to use another format in this architecture, either hardware additions or software improvements should be made in accordance with this hardware.

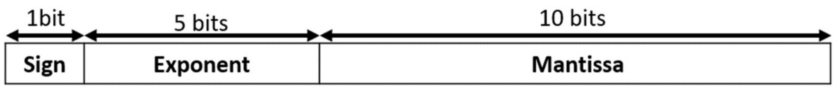


Fig. 2. IEEE 754 FPU-16 Format .

4. BZK.SAU.FPGA

BZK.SAU.FPGA microcomputer architecture is a modular logic gate-based architecture designed in the FPGA development environment [6]. Based on the modularity of computer architecture, BZK.SAU.FPGA can show its superiority when it comes to the User-system relationship, it provides the users with the reconfigurability required to add their own designs instead of a component in the system without causing any negative effect to the system operation. The Computer Engineering Department at Sakarya and Yozgat Bozok Universities in Turkey have been using BZK.SAU.FPGA since 2011 which is a 16-bit microcomputer architecture that is designed on altera FPGA board accordingly with a Computer Architecture Simulator named BZK.SAU[7]. Table 1. shows a summary of the characteristics of BZK.SAU.FPGA microcomputer architecture.

TABLE 1. Characteristics of BZK.SAU.FPGA Microcomputer Architecture [10].

Built in media	Altera DE2-70 FPGA
Built in	Schematic design
Keyboard	Full stroke “clicky”.. keys
Text modes (display)	24 lines × 40 columns
Graphics modes (display)	320 × 384
COLO rsc	Monochrome in VGA mode
RAM	64 KB
ROM	4 MB
Memory endianness	Big-Endian
CPU design	CISC
CPU architecture	Von-neumann (SISD)
Address and data bus	16-bit
The number of GPRs/data and address registers	16
Control unit	Hardware control
The processing of instructions	Non-pipeline
ALU	16 bit (only integers)
Data representation	2's complement
OS	Single user-single task
Written in language	BZK.SAU assembly language
File system	FAT

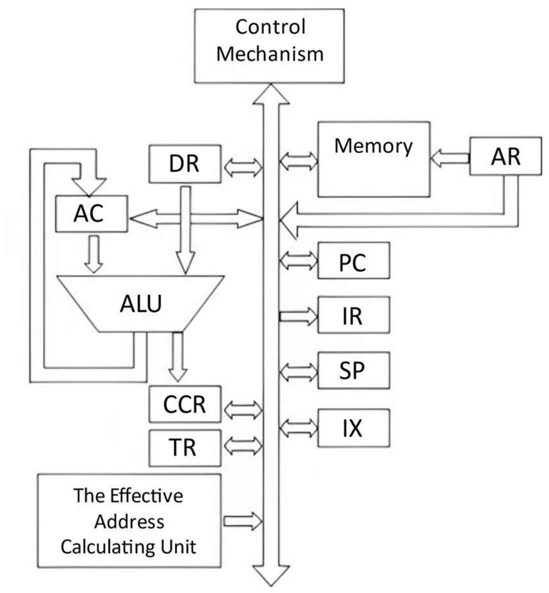


Fig. 3. Block Diagram of BZK.SAU.FPGA. Microcomputer Architecture [7].

5. Methodology

In our study , we create the linear regression model which is used for training datasets by implementing the matrix of least-square solution that helps to find the coefficients of the linear regression function.The first step was by importing the data (x, y) that describe our model and dividing it into two groups inputs and outputs. The next step was the calculation of the least-square coefficients (equation 2). This step was carried out by computing in floating-point terms the matrix transpose, inverse, and multiplication. This is achieved using the steps given below :

$$X^T = \begin{pmatrix} 1 & \cdots & 1 \\ x_1 & \cdots & x_n \end{pmatrix} \tag{3}$$

$$X^T\,X = \begin{pmatrix} 1 & \cdots & 1 \\ x_1 & \cdots & x_n \end{pmatrix}\begin{pmatrix} 1 & x_1 \\ \vdots & \vdots \\ 1 & x_n \end{pmatrix} \tag{4}$$

$$X^T\,X = \begin{pmatrix} \sum_1^n 1 & \sum_1^n x_i \\ \sum_1^n x_i & \sum_1^n x_i^2 \end{pmatrix} \tag{5}$$

$$X^T X = \begin{pmatrix} n & \sum_1^n x_i \\ \sum_1^n x_i & \sum_1^n x_i^2 \end{pmatrix} \tag{6}$$

The matrix inverse was calculated by using 2x2 matrix inverse technique :

$$(X^T X)^{-1} = \frac{1}{n\sum_1^n x_i^2 - (\sum_1^n x_i)^2}\begin{pmatrix} \sum_1^n x_i^2 & -\sum_1^n x_i \\ -\sum_1^n x_i & n \end{pmatrix} \tag{7}$$

$$X^T\,Y = \begin{pmatrix} 1 & \cdots & 1 \\ x_1 & \cdots & x_n \end{pmatrix}\begin{pmatrix} y_1 \\ \vdots \\ y_2 \end{pmatrix} \tag{8}$$

$$X^T\,Y\; = \begin{pmatrix} \sum_1^n y_i \\ \sum_1^n x_i y_i \end{pmatrix} \tag{9}$$

In the end we have:

$$\begin{pmatrix} a_2 \\ a_1 \end{pmatrix} = \frac{1}{n\sum_1^n x_i^2 - (\sum_1^n x_i)^2} \begin{pmatrix} \sum_1^n x_i^2 & -\sum_1^n x_i \\ -\sum_1^n x_i & n \end{pmatrix} \begin{pmatrix} \sum_1^n y_i \\ \sum_1^n x_i y_i \end{pmatrix} \tag{10}$$

Where :

$$a_2 = \frac{1}{n\sum_1^n {x_i}^2 - (\sum_1^i x_i)^2} * (\sum_1^n {x_i}^2 \sum_1^n y_i - \sum_1^n x_i \sum_1^n x_i y_i) \tag{11}$$

$$a_1 = \frac{1}{n\sum_1^n {x_i}^2 - (\sum_1^i x_i)^2} * (-\sum_1^n x_i \sum_1^n y_i + n\sum_1^n x_i y_i) \tag{12}$$

which will be implemented in both IP-Core and FPGA-based Microcomputer Architecture.

6. Implementation

As we can see in equations 11 and 12, the a1 and a2 coefficients depend on the input and output data and the number of data values that are corresponding to the iteration number. For the input data, two summations are required, one for the values themselves and the other one for the squared values. For the output data, one summation is required for the values themselves. For the input and output data, there is one summation that is required for the multiplication of input and output values. As seen also, equations 11 and 12 depend on other operations like the division and multiplication of the precedent summations. To implement equations 11 and 12 and find their corresponding coefficients using single-precision floating-point on FPGA. The proposed two different ways are as follow :

6.1. IP-Core of Xilinx

Firstly, we used xilinx® Floating-Point Operator core v7.1 which is provided by Vivado Design Suite software. This core provides means to perform floating-point arithmetic on an FPGA which can be customized for multiple arithmetic operations with different word lengths, latency, and interface. We customized the IP-core to perform a FPU-16 with the needed arithmetic operations to implement equations 11 and 12. In Fig. 4. An example of implementing linear regression function by using Floating-Point Operator core v7.1 is given.

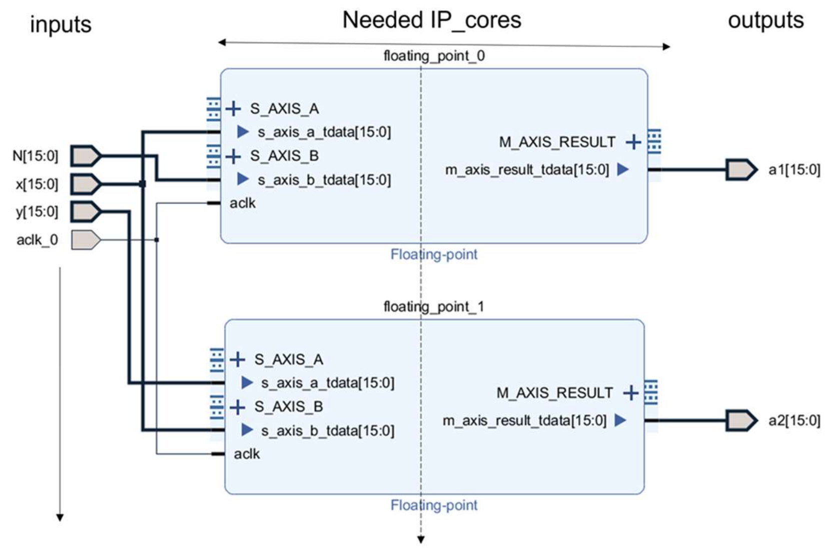


Fig. 4. An Example of Implementing Linear Regression by Using IP-Core of Xilinx.

6.2. BZK.SAU.FPGA- Based Microcomputer Architecture

In the second proposed way, we used the BZK.SAU.FPGA-based microcomputer architecture, which uses a bunch of assembly language instructions to implement the linear regression function. Table 2. shows the main instructions used for the implementation of the FPU-16 unit by BZK.SAU.FPGA.

TABLE 2. Instructions Used in The Implementation of Fpu-16 Unit Using BZK.SAU.FPGA [9] .

Mnemonic	Description		Mnemonic	Description
LDA	AC←M[AR]		SHR	AC← AC ≫ 1
STA	M[AR] ← AC		SHL	AC← AC ≪ 1
BZR	PC← EA ,if Zero_flag=1		INC	AC← $\overline{\text{AC}}$ + 1
BRA	PC← EA		NEG	AC← A C + 1
BMI	PC← EA ,if Zero_flag=1 (negative)		MUL	AC← AC × DR
XOR	AC←AC ⊕ DR		DIV	AC← AC ÷ DR
OR	AC←AC v DR		JMP	AC← EA , M[SP] ← AR
AND	AC←AC ^ DR		RTS	PC← M[SP]
SUB	AC← AC - DR		ADD	AC← AC + DR

EA(effective Adress): PC← PC + Offset .

7. Evaluation

In order to evaluate our implementation of the linear regression function using FPU-16 in both proposed ways. The data of the increasing linear regression proposed by Ferreira et al [5] is used see Table 3.

TABLE 3. Data Of The Linear Regression.

x_i	-1.1	0.1	1.2	2.3	3.1	4.1	4.8	5.7
y_i	-1.7	2.4	5.0	7.3	10.9	12.5	16.2	19.7

From Table 3 we can observe eight decimal training data points that are applied to both proposed ways. These decimal values are converted to FPU-16 format before using them in equations 11 and 12 to get the a1 and a2 coefficients. For the IP-Core of Xilinx, we used Vivado software for the evaluation. The Altera DE2-70 Cyclone II-EP235F672C6 chip is used in the evaluation of BZK.SAU.FPGA.

8. Results And Discussion

After implementing the Linear Regression by using IP-Core and FPGA-based microcomputer architecture and finding their resulting coefficients for each iteration we have summarized the results in Tables 4. and 5.

TABLE 4. The Coefficients Values Obtained From IP-Core of Xilinx.

iteration NO.	a1	a2
1.	1.546	0
2.	2.01	0.3384
3.	3	0.638
4.	2.906	0.717
5.	3.075	0.834
6.	2.914	1.018
7.	2.975	1.305
8.	3.022	1.416

TABLE 5. The Coefficients Values Obtained From BZK.SAU.FPGA.

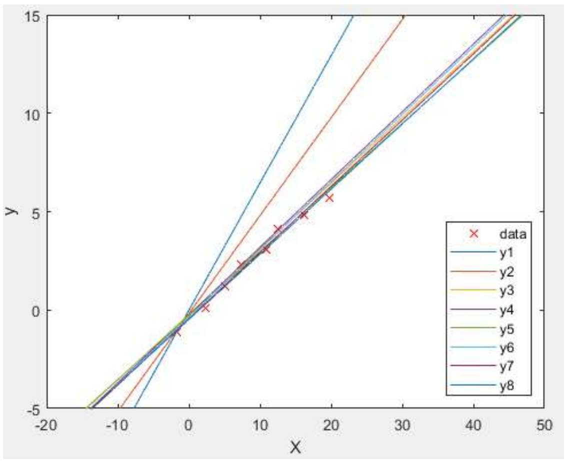
iteration NO.	a_1	a_2
1.	1.551	0
2.	2.004	0.3384
3.	2.996	0.6367
4.	2.908	0.718
5.	3.076	0.8345
6.	2.91	1.017
7.	2.969	1.195
8.	3.012	1.41

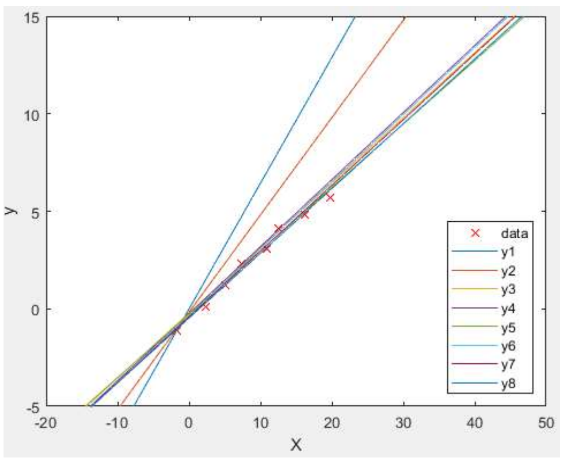
From Tables 4 and 5 we can observe that the obtained results from BZK.SAU.FPGA is so close to the ones obtained from the IP-core of Xilinx despite it being made as an educational tool which proves its usability in such applications.

To evaluate our results we have used MATLAB software to plot the training data points and resulting linear regression function $y{=}a2{+}\ x\ a1$ graphs as given in Table 6. below .

TABLE 6. The Resulting Linear Regression Functions.

IP-Core of Xilinx	BZK.SAU.FPGA
y1=0 +x * 1.546	y1= 0 + x * 1.551
y2=0.3384 +x * 2.01	y2= 0.3384+ x * 2.004
y3=0.638 +x * 3	y3=0.6367+ x * 2.996
y4=0.717 +x* 2.906	y4=0.718+ x * 2.908
y5=0.834 +x * 2.075	y5=0.8345+ x * 3.076
y6=1.018 +x * 2.914	y6=1.017+ x * 2.91
y7=1.305+x * 2.975	y7=1.195+x* 2.969
y8=1.416 +x * 3.022	y8=1.41+x*3.012





(a) (b)

Fig. 5. The Training Data Points And Resulting Linear Regression Functions Graph.(a) IP-Core of Xilinx,(b) BZK.SAU.FPGA.

After that , the Mean Square Error (MSE) was calculated for the results of both the BZK.SAU.FPGA-based microcomputer architecture and IP-Core of Xilinx by using the MATLAB software and we obtained the result of 7.73×10^{-4}.

9. Conclusion

Our goal in this study is to be able to clearly understand the implementation of FPGA- based half-precision floating-point, as it is becoming increasingly important to apply the arithmetic operations in a more accurate and fast way as the technology is still getting developed fast. To do that we have proposed two different ways of implementing FPGA-based FPU-16. Assembler of BZK.SAU.FPGA-based microcomputer architecture was used for the first way .and for the second way IP-Core of Xilinx is used which is simulated and tested using Vivado software. After implementing both ways we have calculated the Mean Square Error MSE between them and obtained the result of 7.73×10^{-4}.

References

[1] TALIB, Manar Abu, MAJZOUB, Sohaib, NASIR, Qassim, et al. A systematic literature review on hardware implementation of artificial intelligence algorithms. The Journal of Supercomputing, 2021, vol. 77, p. 1897-1938.

[2] PURNIMA, Shrivastava, MUKESH, Tiwari, JAIKARAN, Singh, et al. VHDL Environment for Floating point Arithmetic Logic Unit-ALU Design and Simulation. Research Journal of Engineering Sciences. ISSN, 2012, vol. 2278, p. 9472.

[3] GUMBER, Karan et THANGJAM, Sharmelee. Performance analysis of floating point adder using vhdl on reconfigurable hardware. International Journal of Computer Applications, 2012, vol. 46, no 9, p. 1-5.

[4] IEEE Standard for Floating-Point Arithmetic," in IEEE Std 754-2019 (Revision of IEEE 754-2008) , vol., no., pp.1-84, 22 July 2019.

[5] W. de Assis Pedrobon Ferreira, I. Grout and A. C. Rodrigues da Silva, "FPGA hardware linear regression implementation using fixed-point arithmetic," 2019 32nd Symposium on Integrated Circuits and Systems Design (SBCCI), 2019, pp. 1-6.

[6] H. Öztekin , A. Gülbağ and F. Temurtaş , "Assembler Design for BZK.SAU.FPGA Micro Computer Architecture", Electronic Letters on Science and Engineering, vol. 13, no. 1, pp. 1-9, Jul. 2017.

[7] H.Öztekin, F. Temurtas and A. Gulbag, "BZK.SAU.FPGA10.0: Microprocessor architecture design on reconfigurable hardware as an educational tool," 2011 IEEE Symposium on Computers & Informatics, 2011, pp. 385-389.

[8] Kavitha S, Varuna S and Ramya R, "A comparative analysis on linear regression and support vector regression," 2016 Online International Conference on Green Engineering and Technologies (IC-GET), 2016, pp. 1-5.

[9] H.Öztekin, F. Temurtas and A. Gulbag, "BZK.SAU: Implementing a hardware and software-based Computer Architecture simulator for educational purpose," 2010 International Conference On Computer Design and Applications, 2010, pp. V4-90-V4-97.

[10] H. Öztekin; Temurtas, F; Gulbag, A; (2018). On the improvement of the teaching quality and learning effectiveness in the computer organization course through FPGA and modular centered microcomputer design. Computer Applications In Engineering Education, 26, 1840-1825

[11] D. C. Montgomery, E. A. Peck, and G. G. Vining, Introduction to Linear Regression Analysis (4th ed.). Wiley & Sons, 2006.

[12] F. F. Lopes, J. C. Ferreira, and M. A. C. Fernandes, "Parallel Implementation on FPGA of Support Vector Machines Using Stochastic Gradient Descent," Electronics, vol. 8, no. 6, p. 631, Jun. 2019.

[13] I. Grout, W. d. A. P. Ferreira and A. C. R. d. Silva, "Implementation of 3-D Multiple Linear Regression in Hardware using the Xilinx Spartan-3AN FPGA," 2019 16th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON), 2019, pp. 171-174,