Design and Implementation of ANFIS on FPGA and Verification with Class Classification Problem

Moegi Utami[1(✉)], Yukinobu Hoshino[1], and Namal Rathnayake[2]

[1] Kochi University of Technology, 185 Miyanokuchi, Tosayamada-cho, Kami-shi Kochi 782-8502, Japan
275047e@gs.kochi-tech.ac.jp, hoshino.yukinobu@kochi-tech.ac.jp
[2] University of Tokyo, 7-3-1, Hongo Bunkyo-ku, Tokyo 113-8655, Japan

Abstract. In this research, we implemented ANFIS (Adaptive-Network-Based Fuzzy Inference System) on an FPGA(Field Programmable Gate Array) as a System-on-Chip (SoC) system and evaluated its performance using iris flower classification. Additionally, we developed and tested an IEEE 754 16-bit floating-point format tailored for ANFIS operations. The evaluation focused on computational accuracy and speed. The results demonstrate that this FPGA-based ANFIS implementation, coupled with the 16-bit floating-point format, yields high efficiency and superior computational performance.

Keywords: FPGA · ANFIS

1 Introduction

In recent years, there has been a notable paradigm shift in the domain of Internet of Things (IoT) technology, with increasing emphasis on a distributed architecture termed"edge computing". Diverging from the conventional approach, which predominantly centers on centralizing data processing within the cloud, edge computing leverages edge devices or servers embedded in IoT devices to conduct localized data processing. Consequently, only essential data is transmitted and shared with the cloud. This novel approach significantly enhances the real-time responsiveness of information processing while concurrently alleviating the cloud's computational burden by reducing data transmission and processing demands. An area of particular significance where edge devices demonstrate considerable promise is disaster prediction. Through strategic deployment of sensors near water bodies, accurate forecasts of water levels can be attained, enabling the proactive identification of floods and other potential hazards well in advance. This proactive strategy is pivotal in mitigating the impact of disasters, safeguarding communities, and preserving critical infrastructure from potential harm. However, because real-time performance is required, they must operate at

Supported by organization x.

B. Xin et al. (Eds.): IWACIII 2023, CCIS 1931, pp. 241–252, 2024.
https://doi.org/10.1007/978-981-99-7590-7_20

high speeds. Additionally, in the event of a disaster, they must be able to operate on standby power when the primary power supply is unavailable. Therefore, FPGAs are utilized in control units for designing edge devices in embedded systems due to their ability to operate with low power consumption, minimize heat generation during operation, and reduce the overall device footprint. Gomez-Pulid et al. conducted a performance comparison between FPGAs, CPUs, and GPUs, with their experimental results demonstrating that FPGAs outperform CPUs and GPUs in terms of speed and power efficiency when operated continuously for 24 h at an equivalent cost when clustered [1]. FPGAs offer the advantage of being suitable for outdoor installation due to their low heat generation, and their low power consumption contributes to reduced long-term operating costs. Moreover, FPGAs have a compact footprint when integrated into embedded systems, enabling installation in more locations compared to CPUs. Thus, FPGAs surpass CPUs as superior edge device controllers, capable of prolonged operation while occupying minimal space. In this study, we developed and implemented a program based on the learned ANFIS in both FPGA and CPU environments. Our primary focus was on designing the hardware logic circuit for ANFIS in FPGA, emphasizing power consumption and processing speed. Furthermore, we conducted a thorough verification process, comparing the results with CPU processing, to ensure the accuracy of our FPGA-based implementation falls within an acceptable range.

2 Applying AFIS to the Iris Classification

In this paper, we apply a 4-input, 3-MF (membership function) Adaptive-Network-Based Fuzzy Inference System [2,3] to create a program for AI in edge devices. The objective of this program is to identify three types of iris based on four characteristics. The dataset used for machine learning and iris identification is the iris-dataset. It consists of 150 data items, including measurements of petal and sepal length (cm) and width (cm) for three species of iris (setosa, versicolor, and virginica), with 50 data items per variety. These four data items are input into the ANFIS, and the processing results determine the classification of each data set. By assigning a unique value to each iris variety, the ANFIS classification calculation produces the output based on those distinct values. ANFIS for iris identification in this study was designed by using "Neuro-FuzzyDesigner" in MATLAB R2017a and loading the teacher data into the ANFIS automatic design function. The dataset prepared for machine learning is a modified version of iris-dataset. The input data was prepared as the teacher data, which consisted of a total of 150 rows of five items, four of which were the length (cm) and width (cm) of the petals and sepals of the iris, and the eigenvalues of each cultivar were recorded. The eigenvalues for each variety were set as follows. In this study, ANFIS for iris identification was designed using the "Neuro-Fuzzy Designer" in MATLAB R2017a. The teacher data was loaded into the ANFIS automatic design function. For machine learning, a modified version of the iris-dataset was prepared. The input data was arranged as teacher data, comprising

a total of 150 rows, with five items each. Among these, four items represented the length (cm) and width (cm) of the petals and sepals of the iris, while the remaining item recorded the eigenvalues for each cultivar (Fig. 1).

The eigenvalues for each variety were set as follows:
setosa : 100
versicolor : 200
virginica : 300

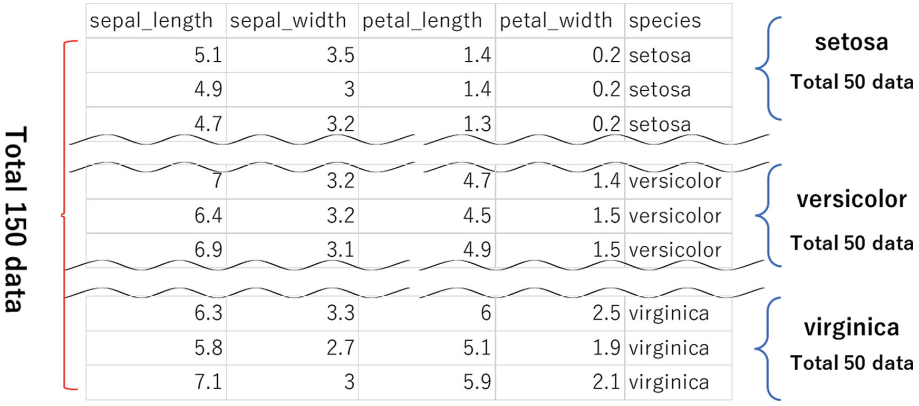

Fig. 1. irisdataset

During machine learning, the MFs are set and learnt for the previous case section. The error values of each MF are shown in Table 1 after learning until the error value of each MF is minimised. The function with the smallest error value that could be designed using only four arithmetic operations was pimf, so the ANFIS was designed with pimf (Pi-shaped membership function) as the MF for the precondition. The function of pimf is as in Eq. (1).

$$f(x,a,b,c,d) = \begin{cases} 0 & (x \le a) \\ 2(\frac{x-a}{b-a})^2 & (a \le x \le \frac{a+b}{2}) \\ 1-2(\frac{x-b}{b-a})^2 & (\frac{a+b}{2} \le x \le b) \\ 1 & (b \le x \le c) \\ 1-2(\frac{x-c}{d-c})^2 & (c \le x \le \frac{c+d}{2}) \\ 2(\frac{x-d}{d-c})^2 & (\frac{c+d}{2} \le x \le d) \\ 0 & (d \le x) \end{cases} \tag{1}$$

x:input value
a,b,c,d: constant parameter

Table 1. Minimal error value due to the MF of the previous case.

MF	error value
trimf	1.0951
trapmf	3.7536
gbellmf	0.021761
gaussmf	0.15818
gauss2mf	0.13541
pimf	0.66251
dsigmf	0.026067
psigmf	0.026067

Figure 2 shows the classification results after educating the iris dataset with pimf in MATLAB R2017a and Fig. 3 shows the pimf graph for the petal lengths created.

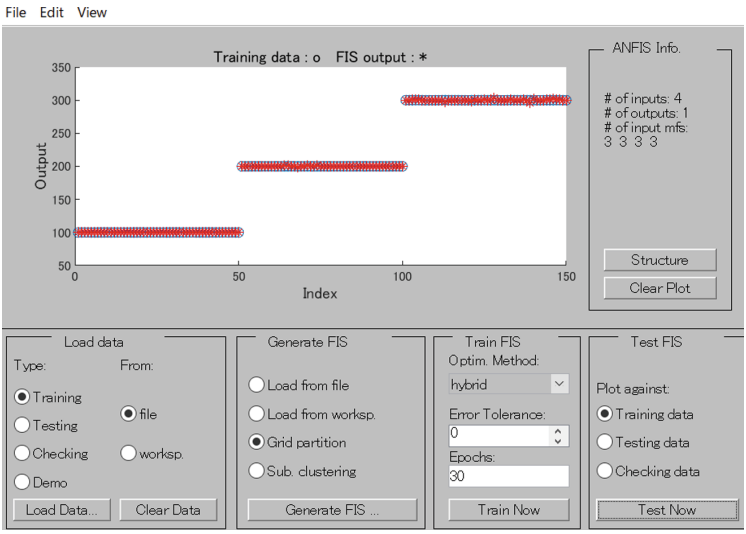

Fig. 2. Post-training classification results for iris-dataset by MATLAb 2017a

ANFIS is structured as a hierarchical network with an input layer and five processing layers. As illustrated in Fig. 4, which represents a 2-input, 2-MF ANFIS, the processing at each layer is summarized as follows. In the input layer, two data items are input to each neuron. Operations are then conducted in the processing layers, spanning from layer 1 to layer 5, following a series of steps.

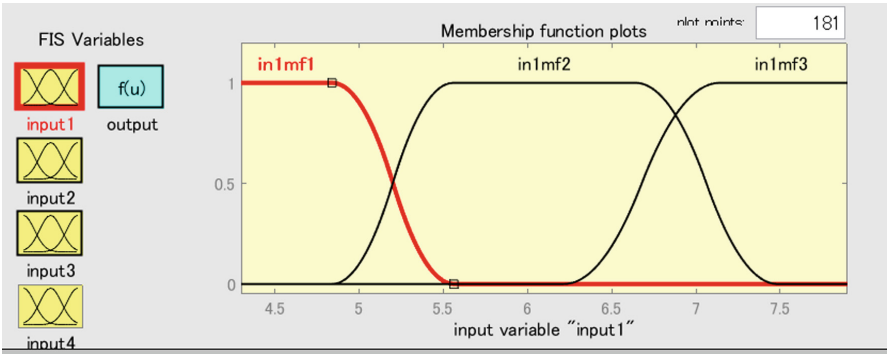

Fig. 3. Graph of pimf for petal length created in MATLAB R2019a.

$$w = MF * MF \qquad \overline{w_i} = \frac{w_i}{\sum_{n=1}^{4} w_n} \qquad Fi = k_1 * x1 + k_2 * x2 + k_3$$

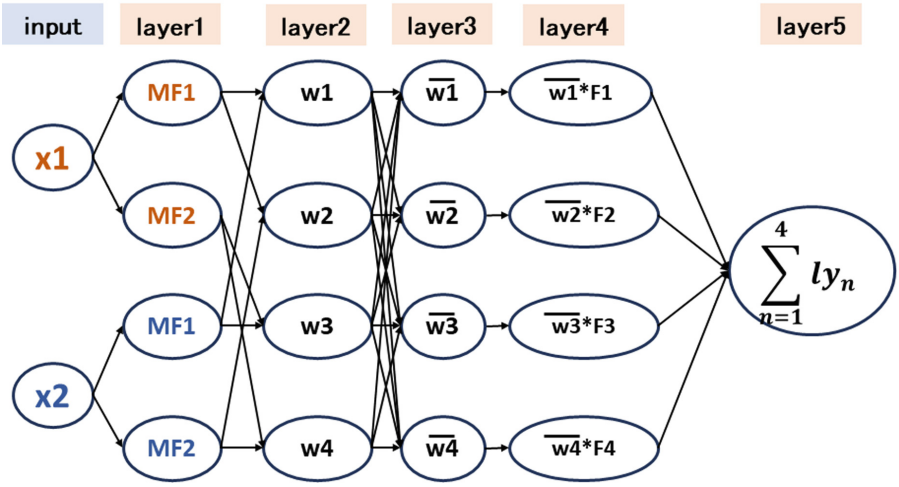

Fig. 4. ANFIS structure with 2 inputs and 2 MFs

1. In the first layer, the values of each neuron input from the input layer are processed by three neurons that store the membership functions (MFs) to compute the degree of truth for each classification.
2. In the second layer, the results of processing by the MF of each neuron in the first layer are combined and multiplied by one from each classification.
3. In the third layer, the result obtained in the second layer is divided by the sum of all neurons in the second layer, and the normalized result is output to each neuron in the next layer.

4. In the fourth layer, the normalized values input from the third layer, the constants set for each neuron, and the values of the four items input in the first layer are processed by linear functions and output.
5. The fifth layer outputs the sum of all processing results from the fourth layer.

The linear function of the fourth layer is as follows, as shown in Eq. (2). The constant k is defined as one more than the number of elements to be input, which is five for four inputs.

$$\overline{w_i} \times (k_{i1} \times (input1) + k_{i2} \times (input2) + k_{i3}) \tag{2}$$

i:Number of neurons in layers 3 and 4

k_1, k_2, k_3:Constants in the posterior function in the "i" neuron

input1,input2: values of the input layer

As the ANFIS created in this paper has 4 inputs and 3 MFs, the ANFIS created in MATLAB has the structure shown in the Fig. 5. There are four inputs, layer 1 has 12 neurons as each input has 3 MFs, and layers 2 to 4 have 81 neurons in the fourth power of 3.

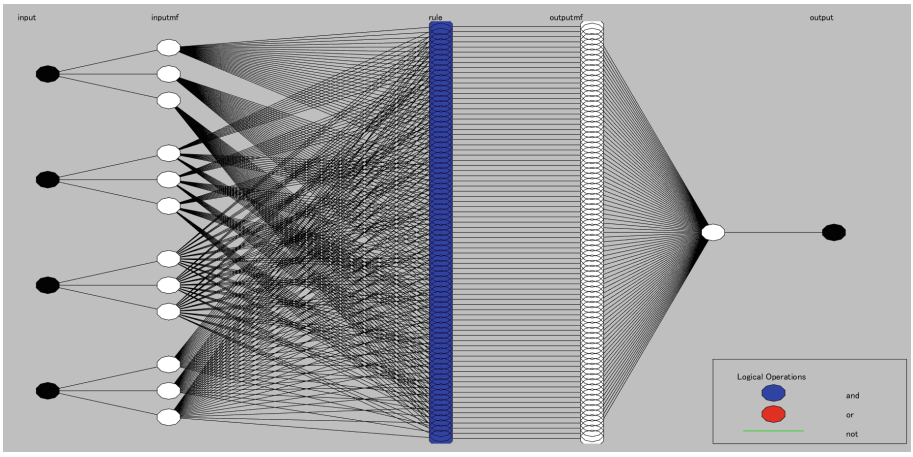

Fig. 5. Post-training classification results for iris-dataset by MATLAb 2017a

3 Hardware Program Design for 16bit ANFIS

The logic circuit was designed using VerilogHDL as the hardware description language and Quartus II 13.1 for development. When designing a hardware logic circuit for ANFIS to identify iris, it is essential to convert floating-point values into a format that can be processed in VerilogHDL. In this paper, the floating-point data handled in the logic circuit is converted into a 16-bit binary single-precision

floating-point data type (IEEE754) for arithmetic operations. The decision to choose fp16, which offers lower precision compared to fp32 in processing results, was driven by the emphasis on cost and processing speed in the hardware logic circuit's design. Although fp16 may yield lower accuracy compared to CPU's 32-bit floating-point operations, its implementation is expected to provide sufficient accuracy for class classification. Additionally, fp16 requires fewer resources in terms of memory usage, making it more efficient for the overall logic circuit. Achieving accurate class classification allows for increased processing speed and reduced power consumption despite the slight reduction in processing precision compared to the CPU's fp32 operations (Fig. 6).

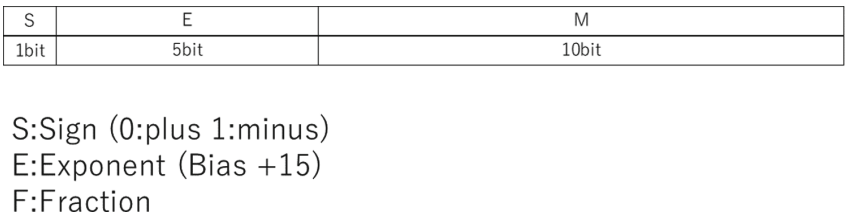

Fig. 6. IEEE 754 16bit floating point

Hardware-software co-design was employed to input the four items of the dataset into the hardware logic circuit. After converting the decimal point values to fp16, the four items are fed into the hardware logic circuit by outputting them to the physical addresses of the pins on the FPGA board. Subsequently, the processing results are retrieved from the pin's virtual address by the software program, converted back to 32-bit floating point, and verified.

The operations executed in layers 1 through 5 of ANFIS were designed by incorporating the logic circuits for the four arithmetic operations and comparison operations of fp16. The logic circuits for the four arithmetic operations of fp16 were designed with reference to Reference [4]. The structure of the ANFIS logic circuitry for layers 1 through 5 is illustrated in Fig. 7. The four pin-input items are labeled as x1 to x4, and the ANFIS processing results are output to the pins labeled as output.The FPGA is always running, but a reset signal is generated when the input x1 x4 is changed. The software programme then retrieves the processing results from the virtual address of the pin, converts them to 32-bit floating point and outputs the processing results for each individual Iris in a csv file. Each layer processing is handled by a pipelined data-flow type calculation [5].

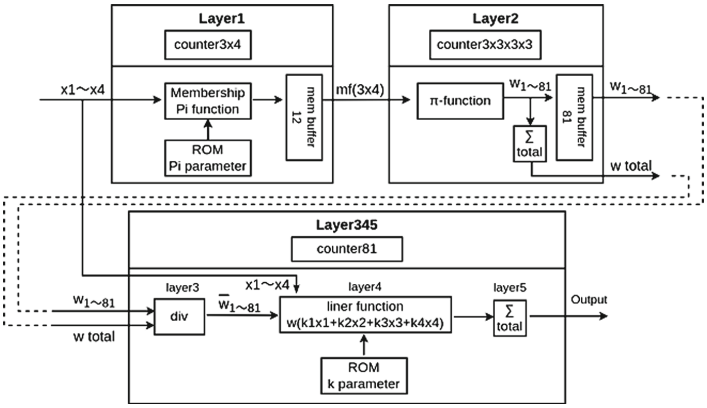

Fig. 7. Structure of the ANFIS logic circuit with 4 inputs and 3 MFs.

The number of pins and logic elements used are shown in Table (2).

Table 2. Hardware Logic Circuit Functions

clock frequency	50 MHz
logic utilization	9,038/32,070 (28%)
Total number of registers	13245
Total number of pins	368/457 (81%)
Total number of block memory bits	103,262/4,065,280 (3%)
DSP block total	16/87 (18%)

Table 3 provides a summary of the processing times for each layer circuit and the entire logic circuit. The simulation results clearly demonstrate that the processing speed approximately doubles when the frequency of the FPGA-implemented logic circuit is doubled. The busy signals of each layer of ANFIS are as shown in Fig. 8, the processing time of each layer is the time from the rise to fall of the busy signal, and the operating end time of the logic circuit is simultaneous with the fall of the busy signals of ly345.

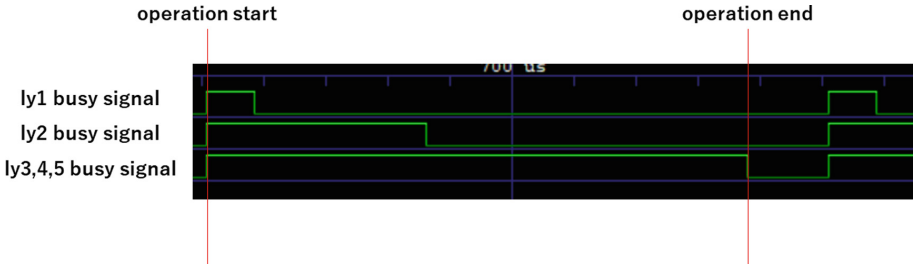

Fig. 8. Busy signals at each layer of ANFIS

Table 3. ANFIS layers of hardware logic circuits and overall processing time [us]

frequency	layer1	layer2	layer3,4,5	entire circuit
50[MHz]	7.717	27.588	59.630	87.218
100[MHz]	3.863	13.789	25.950	43.602

In order to compare the results of hardware AFIS, we designed an AFIS software program for iris identification using C language. This software program was implemented on two different devices: a PC equipped with a high-speed CPU mounted on an SoC FPGA and an ARM processor. The purpose of this comparison is to evaluate the processing performance and computational speed of the 16-bit AFIS designed for FPGAs. Additionally, we developed a software program to read four data items from the iris dataset and perform iris identification. The hardware logic was implemented on the two different devices with varying frequencies and numbers of CPU chips. The average processing time for 150 data points from the 1st to the 5th layers is presented in Table 4.

Table 4. Processing speed by implementation on CPU

CPU	frequency	Number of cores	processing time
ARM®CortexA9	925[MHz]	2	160.431[us]
Intel Core i7	2.8[GHz]	8	12.738[us]

4 Results and Comparison

Based on the data presented in Tables 3 and 4, we observe that Intel Core i7, Model-sim, Cyclone V SEA5 SoC, and ARM CortexA9, in that order, exhibit

faster processing speeds. Comparing the Cyclone V SEA5 SoC with the ARM CortexA9, despite operating at about 1/18.5 of the frequency of the ARM CortexA9, the Cyclone V SEA5 SoC achieves a processing speed approximately 1.84 times faster. Additionally, Model-sim's processing speed is around 0.292 times that of an Intel Core i7, even though it operates at 1/28th of the frequency. Interestingly, an FPGA with an operating frequency of 400 [MHz] theoretically matches the processing time of an Intel Core i7. These comparisons demonstrate that FPGAs possess one core but boast faster processing speeds in ANFIS logic circuits, allowing them to handle data operations within the data input wait time required for an edge device. Furthermore, the FPGA cluster incurs lower costs for cluster design and electricity consumption when operating 24 h a day. Consequently, the FPGA cluster designed in this study, implementing the ANFIS hardware logic circuitry, exhibits sufficient processing speed for edge device control and proves to be a more cost-effective alternative compared to the CPU cluster. We conducted a comparison of software programs and hardware logic circuits for iris identification using ANFIS. Table 5 presents a comparison of correlation coefficients and accuracy between the software program executed on a Dual ARM CortexA9 (CPU) and the hardware logic circuit executed on a Cyclone V SEA5 SoC (FPGA) with the results processed by ANFIS using eigenvalues for each iris variety. The correlation coefficients demonstrate that the CPU identification results exhibit a very strong correlation with each eigenvalue, and similarly, the FPGA identification results also show a strong correlation with each eigenvalue. However, it is worth noting that the mean relative error between the identification results and the eigenvalues is approximately 3.4 times higher for FPGA than for CPU. Additionally, the standard deviation is approximately 3 times higher for FPGA than for CPU.

Table 5. CPU Execution Result

Control devices	correlation coefficient	Average of relative error	standard deviation
CPU	1.0000	0.12[%]	0.21
FPGA	0.9998	0.41[%]	0.61

5 Conclusions and Future Work

In this study, we designed and implemented an ANFIS system for iris identification to compare FPGAs and CPUs as edge device controllers. The results demonstrate that FPGAs can achieve faster processing speeds than CPUs, even at lower operating frequencies suitable for edge device control. Moreover, the FPGA implementation of ANFIS for iris recognition outperforms CPUs with multiple cores and similar frequencies, all at a lower cost. For future work, we

aim to design a logic circuit with higher precision using 16-bit arithmetic and implement it in a robot. While our current hardware logic circuit utilizes 16fp, research by Henry et al. indicates that a design employing bfloat16 (bf16) can attain accuracy and speed equivalent to fp32 in deep learning [6]. By modifying the logic circuits for quadrature and comparison operations, which currently operate with fp16, to work with bf16, we aspire to implement a hardware logic circuit that matches the precision and speed of a 32-bit floating-point CPU. Additionally, we operated FPGA ANFIS by reading a dataset from a SoC-FPGA, but this does not allow us to verify real-time performance. As a future step, we plan to validate the FPGA's performance as a controller by implementing the hardware logic circuit on a smaller FPGA board and executing robot control. This will enable us to assess the FPGA's capabilities for real-time applications effectively.

Furthermore, while ANFIS for iris identification was implemented in FPGAs in this paper, previous work has included a photovoltaic panel emulator [7], a heater plate Algorithm for adjusting the pulse-width modulation (PWM) duty cycle [8], bilateral teleoperation system [9]. We believe that it is possible to apply ANFIS to classification and learning problems as there have been successful simulations using ANFIS implemented on FPGAs, and we intend to develop a new ANFIS for the future. We plan to verify whether the logic circuit created in this paper can be applied to ANFIS with other data sets. Integrating these technologies into IoT devices for disaster monitoring holds vast potential. AI camera devices can create an efficient disaster monitoring network, enabling real-time data and early detection. The AI accelerator designed for disaster prediction can revolutionize management practices, improving accuracy and response strategies. Our commitment to pushing technological boundaries for disaster management stems from the desire to safeguard lives and communities globally. With IoT devices and AI accelerators, we envision swift and effective disaster responses, minimizing casualties and environmental impact. We're dedicated to making the world more resilient and will persist in advancing science and technology toward this goal.

Acknowledgment. This work was supported by JSPS KAKENHI Fostering Joint International Research(B) 22KK0160.

References

1. Gomez-Pulido, J.A., Vega-Rodriguez, M.A., Sanchez-Perez, J.M., et al.: Accelerating floating-point fitness functions in evolutionary algorithms: a FPGA-CPU-GPU performance comparison. Genet. Program Evolvable Mach. **12**, 403–427 (2011)
2. Takagi, T., Sugeno, M.: Fuzzy identification of systems and its applications to modeling and control. IEEE Trans. Syst. Man Cybern. SMC-15(1), 116–132 (1985)
3. Jang, J.S.: ANFIS: adaptive-network-based fuzzy inference system. IEEE Trans. Syst. Man Cybern. **23**(3), 665–685 (1993)
4. Shirazi, N., Walters, A., Athanas, P.: Quantitative analysis of floating point arithmetic on FPGA based custom computing machines. In: Proceedings IEEE Symposium on FPGAs for Custom Computing Machines, pp. 155–162. IEEE (1995)

5. Kim, H., Choi, K.I.: A pipelined non-deterministic finite automaton-based string matching scheme using merged state transitions in an FPGA. PloS one **11**(10), e0163535 (2016)
6. Henry, G., Tang, P.T.P., Heinecke, A.: Leveraging the bfloat16 artificial intelligence datatype for higher-precision computations. In: 2019 IEEE 26th Symposium on Computer Arithmetic (ARITH), pp. 69–76 (2019)
7. Gómez-Castañeda, F., Tornez-Xavier, G.M., Flores-Nava, L.M., Arellano-Cárdenas, O., Moreno-Cadenas, J.A.: Photovoltaic panel emulator in FPGA technology using ANFIS approach. In: 2014 11th International Conference on Electrical Engineering, Computing Science and Automatic Control (CCE), Ciudad del Carmen, Mexico, pp. 1–6 (2014). https://doi.org/10.1109/ICEEE.2014.6978289
8. Huang, C.W., Pan, S.T., Zhou, J.T., Chang, C.Y.: Enhanced temperature control method using ANFIS with FPGA. Sci. World J. 2014, 8 (2014). Article ID 239261. https://doi.org/10.1155/2014/239261
9. Khati, H., et al.: Neuro-fuzzy control of bilateral teleoperation system using FPGA. Iran. J. Fuzzy Syst. **16**(6), 17–32 (2019)