Journal of Computers Vol. 33 No. 6, December 2022, pp. 51-60
doi: 10.53106/199115992022123306004

Operator Hardware Realization using FPGA: CMAC Implementation as an Example

Kuan-Yi Lin[*], Shih-An Lee, Peter Liu

No. 151, Yingzhuan Rd., Tamsui Dist., New Taipei City 251301, Taiwan R. O. C.
805440012@gms.tku.edu.tw

Received 21 March 2022; Revised 1 April 2022; Accepted 2 April 2022

Abstract. We propose a hardware implementation on a field-programmable gate array (FPGA) of the cerebellar model articulation controller (CMAC) with an integer numeric system. Existing hardware implementations of floating-point processes on FPGAs have utilized the IEEE 754 standard, which introduces complex operations leading to long processing times. For the most complex part of the CMAC, that is, the exponential operator, traditional look-up table methods carry high memory costs. We, therefore, use a Taylor series to realize the exponential operator. Finally, we connect all the modules into the CMAC.

Keywords: CMAC, hardware implementation, integer numeric system

1 Introduction

1.1 Cerebellar Model Articulation Controller

The cerebellar model articulation controller (CMAC), proposed in 1975 by Albus, is a type of neural network model that imitates human learning [1-2]. The CMAC structure defines a series of mapping processes. After quantization, the input states are saved in the corresponding fixed memory units, and the output is the summation of the memory units. Fig. 1 shows the structure of the CMAC, which iteratively applies the corresponding process to finish learning. First, the learning space is defined. Second, the space is divided into many discontinuous states, which are the input states of the CMAC. Third, each element is projected onto the memory. Finally, the memory units are summed. Before the system finishes learning, it compares the real output and reference values to determine the error, which affects the values of memory units, such that the memory values become close to the correct values. The CMAC processing is completed when the system reaches a defined stopping condition.

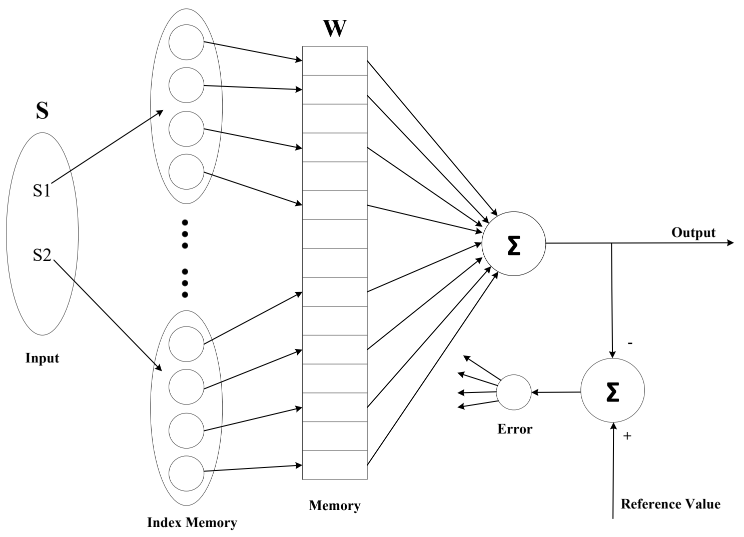


Fig. 1. CMAC structure

* Corresponding Author

Input Space. In Fig. 1, the input space is defined as , where represents the dimension of the input and represents the dominant factor of the plant system.

Index Memory. In this part, the system quantizes the value from the input space by using the Gaussian function:

$$\phi_{ij}(s_j) = \exp\left(\frac{-(s_j - m_{ij})^2}{2\sigma_{ij}^2}\right), \tag{1}$$

where φ_{ij} is the j-th layer of the i-th input s_i, and m_{ij} and σ_{ij} are the mean and variance terms of the Gaussian function, respectively. Here, each element is represented as:

$$m_{ij} = [m_{i1}^T m_{i2}^T \dots m_{in_R}^T]^T \in R^{n_i \cdot n_R} \text{ and} \tag{2}$$

$$\sigma_{ij} = [\sigma_{i1}^T \sigma_{i2}^T \dots \sigma_{in_R}^T]^T \in R^{n_i \cdot n_R}\ . \tag{3}$$

Index Memory. Every element in the memory space corresponds to a receptive region. The multidimensional receptive field is defined as:

$$\begin{aligned} b_j(s_j, m_j, \sigma_j) &= \Pi \phi_{ij}(s_j) \\ &= \exp\left(\sum_{i=1}^{n_i} \frac{(s_i - m_{ij})^2}{\sigma_{ij}^2}\right) \text{ for } j = 1, 2, \dots, n_R\ , \end{aligned} \tag{4}$$

where b_i is associated with the i-th receptive field. The multidimensional receptive field function can be expressed as a vector $\Gamma(s, m, \sigma) = [b_1, b_2 \cdots b_{n_R}]^T \in R^{n_R}$.

Memory-Weight Space. The memory-weight space adjusts the weight of each value from the receptive space. The weight value v_j is linearly combined with the state variable z . The corresponding weight is denoted as:

$$w_j = v_j^T z\ . \tag{5}$$

After calculation, the weight is automatically adjusted.

Output. The output of the CMAC is the following sum of outputs of the memory-weight space:

$$y_c = \sum_{i=1}^{n_{nR}} \mu_i(s(t)) z^T v_i = \zeta^T \theta\ , \tag{6}$$

where $s = [s_1 s_2 \cdots s_j]^T$ are the previously defined variables, $\zeta = [z^T \mu_1\ z^T \mu_2\ \dots z^T \mu_{n_R}] \in R^{n_i \cdot n_R}$, and θ is a vector with appropriate dimensions.

Learning Mechanism. As previously stated, the CMAC is a type of learning controller, which requires the information of the system state and trajectory reference as the input to the system to facilitate the system's ability to learn. To provide the system with clear information, the difference between the system state and the trajectory reference becomes direct information. The learning mechanism is that each weight value is adjusted in the memory weight space and the value is updated by the linear equation as follows:

$$w_j(t+1) = w_j(t) + k_{j1}e(t) + k_{j2}\tilde{e}(t) \,, \tag{7}$$

where k_{j1} and k_{j2} represent the learning rates, $e(t)$ represents the error at time t, and $\tilde{e}(t)$ represents the error difference at time t.

2 Literature Review

The CMAC is a non-fully connected neural network similar to an associative memory network with an overlapping receptive field [3-4]. To avoid discontinuous output in typical CMAC approaches, fuzzy CMAC (FCMAC) networks use fuzzy sensors for more complex applications [5]. The CMAC has typically been implemented in software. Meanwhile, a field-programmable gate array (FPGA) has high speed, low cost, large memory, and short time-to-market platform [6]. Savran and Serkan provided a hardware implementation of the neural network using an FPGA [7].

A numeric system is most often used for hardware implementations of these algorithms and controllers. The floating-point (FLP) implementation is generally used to realize an algorithm or a controller. It is based on the Institute of Electrical and Electronics Engineers (IEEE) standard for FLP arithmetic (IEEE 754) established in 1985 [8-9], where the format is a “set of representations of numerical values and symbols”. A format may also include how the set is encoded. For 32-bit single precision, the FLP data format comprises a 23-bit fraction, an 8-bit exponent, and a 1-bit sign bit, and the format Radix is 2. Using the 32-bit FLP data format shown in Fig. 2, a nonzero number can expressed as:

$$y = (-1)^S \times 1.F \times 2^{E-bais} \,, \tag{8}$$

where *S* represents the sign flag, *F* represents the fraction part, and *E* represents the value of the exponent part.

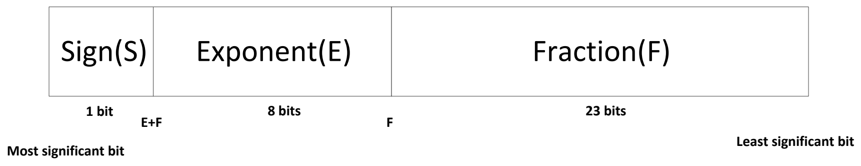


Fig. 2. IEEE 754 single-precision format

Although FLP design has the advantage of high accuracy, it is complex and has a high calculation cost. Therefore, the flexibility of realizing a tradeoff of operating performance, layout cost, and power consumption is limited when this type of numeric system design is considered.

3 Problem Statement and Motivations

A traditional controller cannot effectively control a complex system. Therefore, we need to choose a controller that can adapt to the control environment, such as a CMAC. However, such controllers have complex operation. Traditionally, they are implemented on software via a system-on-a-chip approach. However, this complex step-by-step process for operations decreases the controller’s performance. To enhance the process performance, the CMAC should be implemented with a pipeline design. Therefore, the hardware design value emerges in this scenario. In hardware implementation, the tradeoff of operating performance, layout cost, and power consumption is the most important issue. As described in the Literature Review, although IEEE 754 has high accuracy, it is only up to the standard in the three requirements of hardware implementation. Hence, this paper presents an alternative numeric system that can optimize the situation.

4 Integer Numeric System

To increase the operating performance and simplify the design procedure of the CMAC, we provide an integer numeric system that not only increases the processing speed but also maintains the accuracy of IEEE 754. To maintain the accuracy, we extend each element in the system to 1 million. That is, the elements from 10^{-1} and 10^{-6} become integers. Elements smaller than 10^{-6} are too small to affect the system; therefore, they can be neglected. Thus, the accuracy of the integer numeric system is maintained. As the operations of the integer numeric system are simpler than those of the FLP numeric system, the former can achieve faster processing speed. It can also be easily designed as software. We use the circle area problem with a 5-unit radius to compare the FLP and integer numeric systems. In the FLP process, π is equal to $3.141,592,653,59$, and in the integer numeric process, π is set as $3.141,592$ and the radius value is $5,000,000$. In this experiment, we use Quartus SignalTap as the environment as it can facilitate the real-time state. The experimental results shown in Fig. 3 and Fig. 4 demonstrate that the integer numeric system has better operating performance, with equal accuracy of both systems.

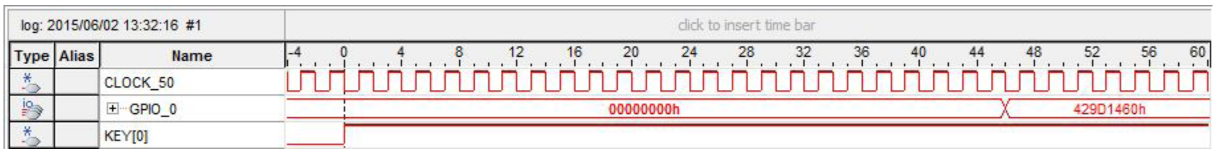


Fig. 3. Results of floating-point numeric system calculations

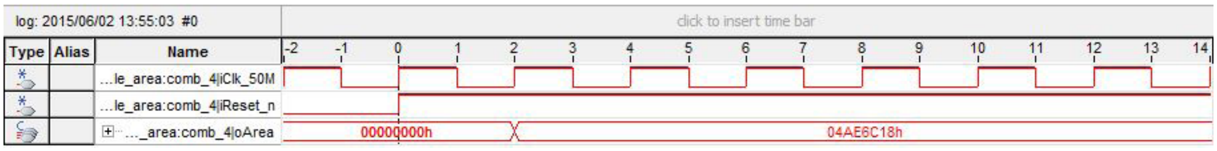


Fig. 4. Results of floating-point numeric system calculations

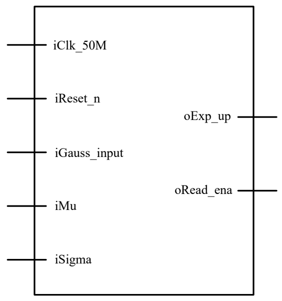
We choose Quartus's simulation of vector waveform timing as the simulation environment. This environment can present the real circuit behavior considering gate delay, settling time, and hold time.

5 CMAC Hardware Implementation and Simulation Results

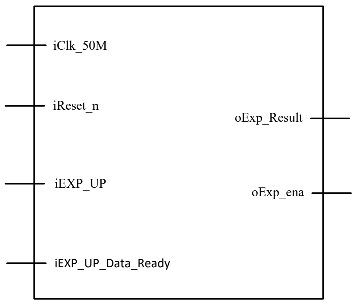
The CMAC structure can be divided into the Gaussian function operation module, Gaussian function output multiplication and summation operation module, error value calculation module, and weight update operation module.

In general design, a continuous mathematical function is usually implemented in the hardware system by using the method of look-up tables. However, this table should be memorized in a Flash memory, which requires a long communication time. Therefore, although the look-up table method can save layout cost, it may decrease the operating precision and increase the operating time of the system. To maintain the precision and operating performance, we implement the mathematical function in hardware and use the integer numeric system to avoid complex operation.

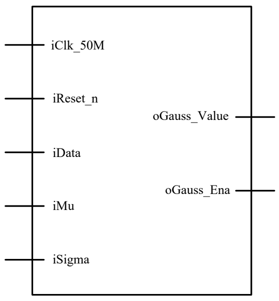
Block diagrams of each circuit module are shown in Fig. 5(a)-Fig. 5(f). Each circuit module provides the handshake signal to inform the receive module whether the data is ready. The user sends a clock input with 50 MHz, one bit of the Reset signal, 32 bits of the system goal, and 32 bits of the system state. Then, the CMAC circuit outputs a 32 bit response and a handshake signal.



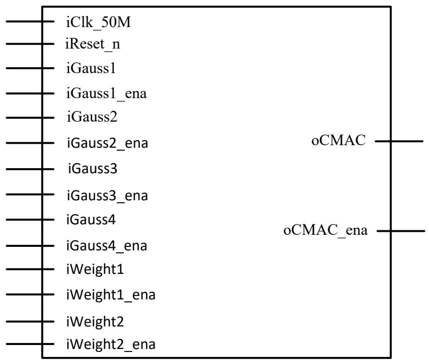
(a) Power value calculation in the exponential function



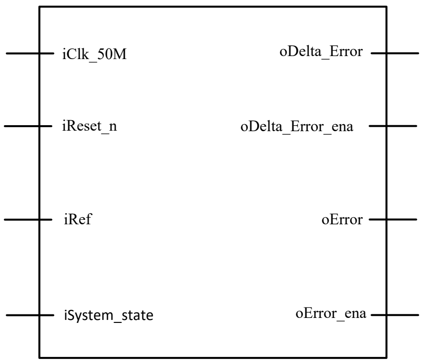
(b) Exponential function



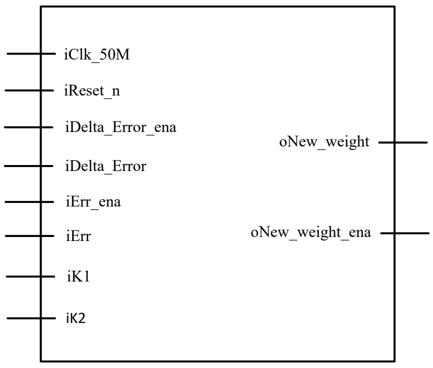
(c) Gaussian function



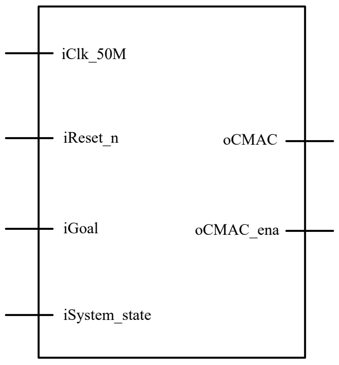
(d) Multiplication and summation function



(e) Error value calculation



(f) Weight update operation module



(g) CMAC

Fig. 5. Circuit block diagrams

5.1 Gaussian Function

The Gaussian function, which is vital to implementation, comprises three parts: power value calculation, exponential calculation, and reciprocal calculation. In the power value calculation part, we calculate $(s_j - m_{ij})^2 / 2\sigma_{ij}^2$ is calculated. This requires subtraction, multiplication, and division operators. For subtraction, we design 32-bit input and output operators. Multiplication is used to complete the square term. For multiplication, because 1 million is multiplied twice, the multiplication result is divided by 1 million. The numbers obtained after multiplication need to be further divided. We first divide σ_{ij}^2 by 1 million to prevent 1 million from disappearing. In the exponential part, the exponential function can be defined as a Taylor series as follows:

$$e^x = \sum_{n=0}^{\infty} \frac{x^n}{n!} = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!} + \frac{x^5}{5!} + \cdots. \tag{9}$$

In order to maintain the layout cost and precision, we only use seven elements in the series because after seven elements the impact value will be less than 10^{-6}. Each element is calculated and then summed, and after seven elements the impact value is less than 10^{-6}. After the exponential operation, the $1/e^x$ in the Gaussian function can easily be defined. To keep the 1 million in the reciprocal operation, we multiply the denominator by 1 million twice. This allows us to maintain accuracy.

To prevent the receive circuit module from receiving data at the signal setup time, we let the oRead_ena signal rise at the next clock. Fig. 6 to Fig. 8 respectively show the power value, exponential, and reciprocal calculations of the exponential function, which can have equal valence to the Gaussian function output. In the simulation result for the power value in the exponential function part, the input value is set as 5, mean value is set as 2, and σ is set as 2. As shown in Fig. 6, the operating result is 1.125. In the exponential simulation part, the input value is set as 2, and, as shown in Fig. 7, the operating result is 7.387,298. Finally, in the Gaussian simulation part, the input data is set as 5, mean value is set as 2, and σ is set as 2. As shown in Fig. 8, the operating result is 0.325,005 and the operating time is less than 1.44 μs.

5.2 Gaussian Function Output Multiplication and Summation Module

This module is the output part of the CMAC, wherein the result of the Gaussian function and the weight space are multiplied and summed. The waveform simulation result is shown in Fig. 9, which indicates two inputs to the module. In the first part, the first, second, third, and fourth Gaussian functions are set as 3, −2, 2, and −1, respectively; the first and second weights are set as 2 and 4 and the operating result is −2. In the other part, the first, second, third, and fourth Gaussian functions are set as 1, −1, 2, and −3, respectively; the first and second weights are set as 1 and 0.005 and the operating result is −1.03. The simulation results indicate that the handshake signal in this module ensures that each of the input data is ready, and the operating time is approximately 100 ns.

5.3 Error Value Calculation Module

To provide the CMAC learning resource, the error between the desired reference and the system output state, as well as the difference in errors, are necessary information. The error value calculation module computes these two inputs and connects to the weight update operation module. Fig. 10 shows the simulation results for the relationship between the error and the error difference. The tracking goal is set as 5 and the initial system state is set as 3. After the operation, the error value is 2 and the error difference is 0 because the initial error value is 0. Next, when the system state changes to −1, which represents the CMAC affecting the plant system, the error value becomes 6 and the error difference becomes −2. The simulation result indicates that the operating time for the error value calculation is approximately 160 ns.

5.4 Weight Update Operation Module

In this part, we simulate the learning response of the CMAC through the weight update operation module. Fig. 11 shows the waveform of the weight update. The constant k_1 is set as 5 and k_2 is set as 3. If the error is set as 3 and the error difference is set as 5, then the weight update value is 30. Next, if the error changes to 0 and the error difference changes to −2, the weight update is 24. Each time update in the simulation takes approximately 240 ns.

5.5 Whole CMAC Circuit

Finally, the whole CMAC is simulated and the results are shown in Fig. 12. The circuit takes less than 1.6 μs to operate a CMAC learning period.

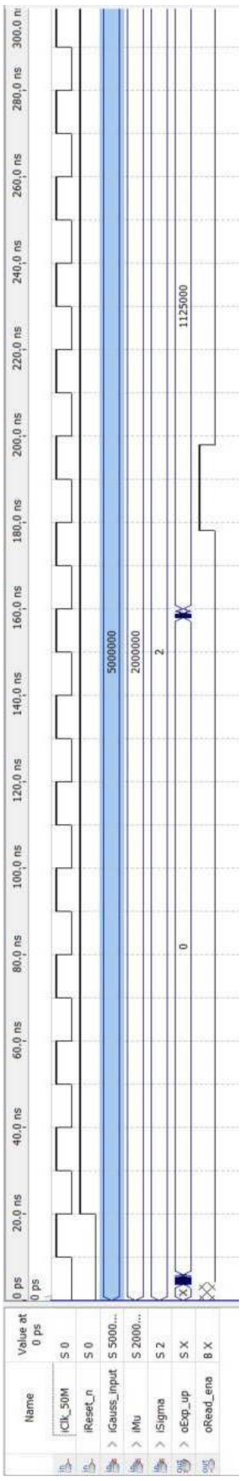


Fig. 6. Waveform of the power value in the exponential function

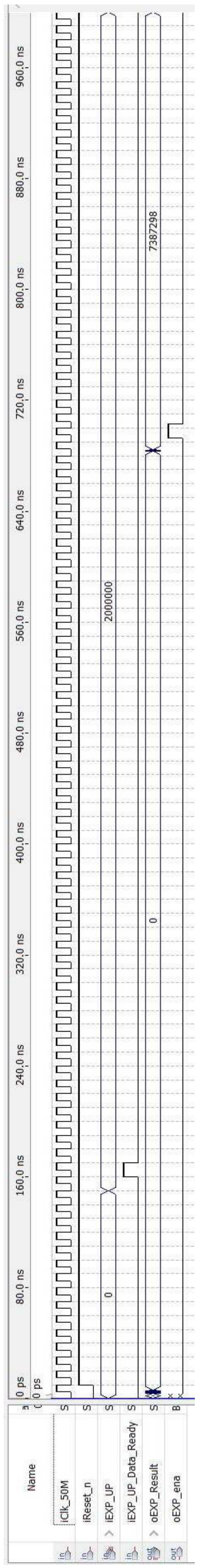


Fig. 7. Waveform of the exponential function

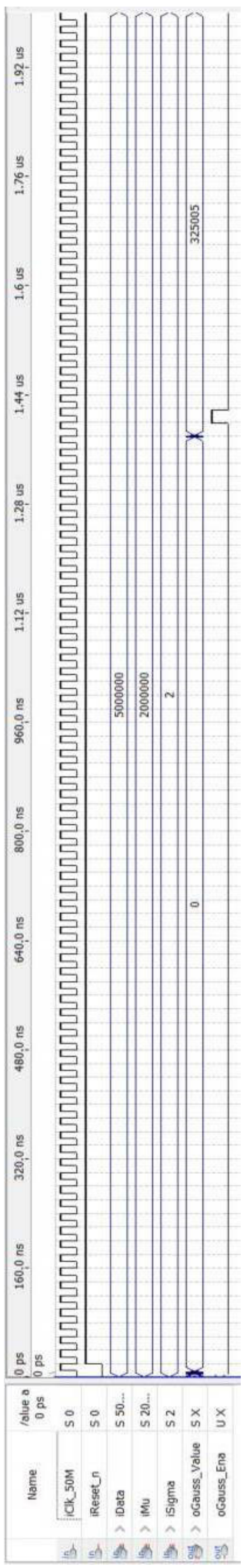


Fig. 8. Waveform of the Gaussian function

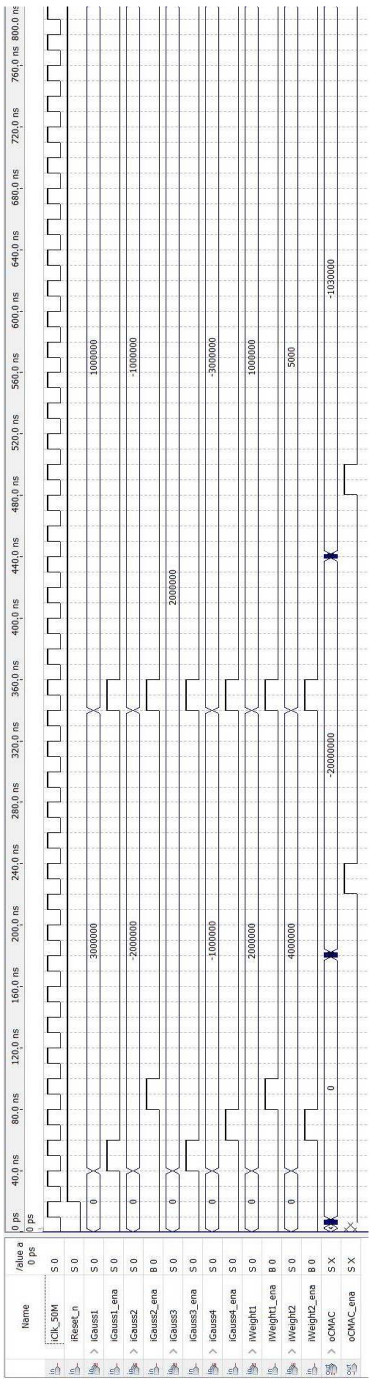


Fig. 9. Waveform of the multiplication and summation function

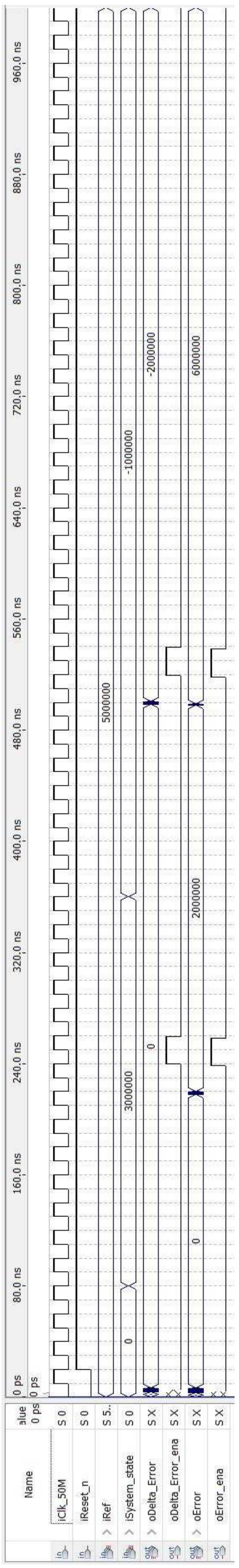


Fig. 10. Waveform of the error value calculation function

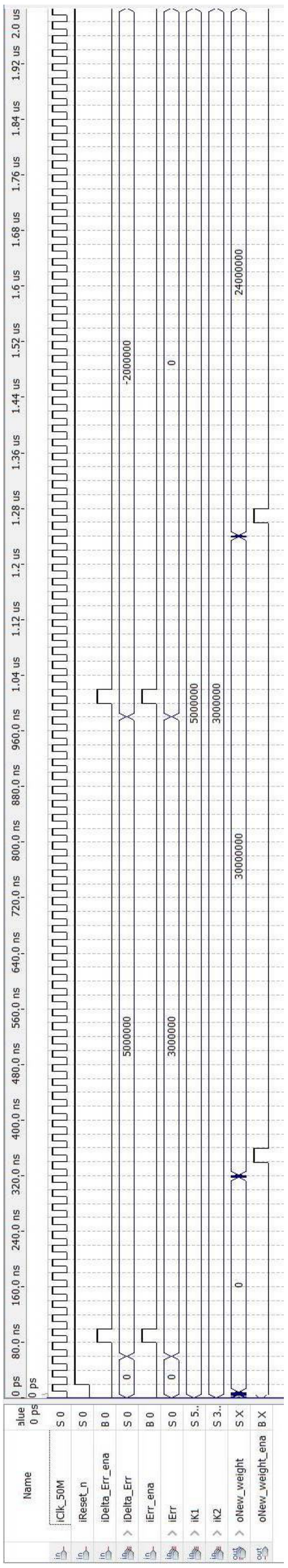


Fig. 11. Waveform of the weight update

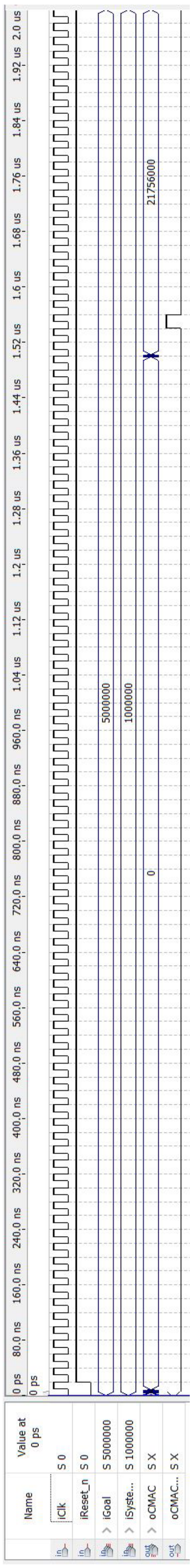


Fig. 12. Waveform of CMAC

6 Conclusion

The cerebellar model articulation controller (CMAC) is a highly adaptable controller; however, its learning model requires a lengthy processing time. Therefore, hardware implementation is an avenue to enhance processing performance. The traditional hardware implementation using the IEEE 754 standard to process the floating-point system uses complex operating elements, which increases the processing time. In this paper, we presented the integer numeric system in the CMAC structure and scaled the floating point between 10^{-1} and 10^{-6} to improve the system's performance. This method is quicker than IEEE 754 for processing. In future, the hardware module could be integrated into application-specific integrated circuits (ASIC) for implementation on any device requiring a high learning ability.

7 Acknowledgements

This paper was supported by Ministry of Science and Technology, Taiwan, under grant numbers MOST 107-2218-E-032-004, MOST 108-2221-E- 032-040, and MOST 109-2221-E-032-033.

References

[1] J. Albus, A New Approach to Manipulator Control: The Cerebellar Model Articulation Controller (CMAC), Dynamic System, Measurement, and Control (1975) 220-227.

[2] J. Albus, Data Storage in the Cerebellar Model Articulation Controller (CMAC), Dynamic System, Measurement, and Control (1975) 220-227.

[3] J.-Q. Huang, F. L. Lewis, Neural-network predictive control for nonlinear dynamic systems with time-delay, IEEE Transactions on Neural Networks 14(2)(2003) 377-389.

[4] F. Hong, S.S. Ge, T.H. Lee, Practical adaptive neural control of nonlinear systems with unknown time delays, in: Proc. of the 2004 American Control Conference, 2004.

[5] S.-F. Su, Z.-J. Lee, Y.-P. Wang, Robust and fast learning for fuzzy cerebellar model articulation controllers, IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) 36(1)(2006) 203-208.

[6] D. Job, V. Shankararaman, J. Miller, Combining CBR and GA for designing FPGAs, in: Proc. of the Third International Conference on Computational Intelligence and Multimedia Applications, 1999.

[7] A. Savran, S. Unsal, Hardware Implementation of a Feed forward Neural Network Using FPGAs, EGE University, Dep. of Electrical and Electronic Engineering, 2003.

[8] D. Goldberg, What every computer scientist should know about floating-point arithmetic, ACM Computing Surveys (CSUR) (1991) 5-48.

[9] A. Sasidharan, P. Nagarajan, VHDL implementation of IEEE 754 floating point unit, in: Proc. of the International Conference on Information Communication and Embedded Systems (ICICES2014), 2014.