MICROPROCESSORS WITH LOOK-UP TABLE (MLUT) FUZZY LOGIC CONTROLLERS

HENG SIEW TAN
Department of Electrical Engineering
University of Wyoming

BOGDAN M. WILAMOWSKI AND RICHARD S. SANDIGE
Department of Electrical Engineering
University of Wyoming

ABSTRACT

This paper investigates a hardware implementation of a multi-input single-output fuzzy logic controller using a microprocessor and a look-up table (MLUT) technique. This concept is simple yet has a very fast response time. This approach yields higher speeds compared to other fuzzy controllers that use software implementations with a PC or a workstation and microprocessor systems that executed the complete fuzzy inference processes. Simulation results for different resolutions are provided for the classical example of the backing truck to ramp problem. Simulation results show that fuzzy logic's interpolation capabilities work well with a fairly low number of bits, that is, 4 to 6 bits.

INTRODUCTION

A fuzzy controller can be designed either by programming a microprocessor system or by a digital hardware system composed of application specific integrated circuits (ASICs) or programmable logic devices (PLDs) [1][2][3]. Many current designs use a microprocessor system. Such software implementations of fuzzy logic controllers are accomplished using a microcontroller, a personal computer, or a workstation. Software implementations usually require several milliseconds to several hundreds of milliseconds to obtain the inference results. Processing time is further increased when the size of the knowledge base or the number of input variables is increased due to the large amount of real-time data processing required. Since fuzzy controllers are real-time systems, speed is important; the higher the speed the better the system. Hardware implementations, on the other hand, can be faster, cheaper, and more compact [1][2][3]. The first hardware fuzzy chip was developed by M. Togai and H. Watanabe [1] at AT&T Bell Laboratories using VLSI technology. T. Yamakawa [2] pioneered fuzzy logic at the hardware level by developing a system that can accept linguistic information and perform approximate reasoning-based inference at very high speeds (more than one million Fuzzy Logic Inferences Per Second or FLIPS). A PLD-based fuzzy logic controller system is developed and investigated by H. S. Tan, R. Sandige, and B. Wilamowski [3]. This system has a very fast fuzzy inference conversion time of more than eight million FLIPS.

This paper investigates a hardware implementation of a multi-input single-output fuzzy logic controller using a microprocessor and a look-up table (MLUT) technique. Total fuzzy inference process conversion time for a two-input single-output system using

a microprocessor such as the Motorola's MC68HC11 with an 8MHz crystal is about 310μs. This speed is more than 15 times faster than the MC68HC11 kernel designed by Motorola [4]. The processing time of the MLUT system is independent of the size of the knowledge base and the number of labels (terms), as opposed to other software implementations where the processing time is increased when size of the knowledge base or the number of terms is increased.

MLUT SYSTEM

Mathematical tables of logarithms, trigonometric functions, etc. are well known, their use being to convert one number into another number which is a function of the first number. Although it is usually possible to devise an algorithm to calculate the converted number this is often a fairly lengthy process. The alternative is to load a memory storage device with this information. In essence, a MLUT fuzzy logic controller can be accomplished by placing the results required by the fuzzy logic controller into a memory device or devices. The table is constructed by making the address correspond to the number being converted and the output data to the 'answer'.

DISCRETIZATION OF UNIVERSES OF DISCOURSE

A universe of discourse of a parameter can be either discrete or continuous. Usually, a universe of discourse in a fuzzy control system is continuous. In order to quantify such information for digital processing, a discretization of the continuous universe to form a discrete universe is performed. Discretization of a universe of discourse is frequently referred to as quantization. In effect, quantization discretizes a universe of discourse into a certain number of segments (quantization levels or resolutions). Each segment is labeled as a generic element, and forms a discrete universe. A look-up table based on the discrete universes, which defines the output of a controller for all possible combinations of the input signals, can be implemented by off-line processing.

To obtain the look-up table, the desired resolutions of all the input and output variables must be selected. The resolution for the input and output variables should be a power 2, i.e., 2^n, where n is the number of bits. The result of $\log_2$(resolution) gives the number of bits required for each input and output variable. After the input and output parameters have been determined, the look-up table is obtained via an off-line computer software program that performs the fuzzy inference process computations for all possible input combinations. This process includes fuzzification of the input signals, rules evaluation ("min-max inference), and defuzzifications. The size of the look-up table is directly proportional to the resolution of the inputs. Mathematically, the number of rows in the look-up table is equal to the product of the resolutions of the inputs universes of discourse:

$$\text{Number of rows} = \prod_{i=1}^{n} RES_i \qquad (1)$$

where n is the total number of inputs in the system and RES_i is the resolution of the i-th input.

DEGREE OF MEMBERSHIP FOR AN INPUT TO ITS BOUNDARY VALUES

By programming a microprocessor, a technique utilizing the degree of membership concept for the inputs and their boundary values can be realized. This technique allows partial membership for an input to its boundary. In other words, an input may partially belong to a boundary value. For instance, if an input value falls in between two boundary values, two boundary-value conditions are considered for the final output result.

To illustrate the approach, let us assume that input variables x and y have the membership functions as shown in Figure 1. For simplicity, assume that a resolution of 4 is selected for both of the inputs and the output, therefore two bits are required for each input and the output to represent the quantization of their universes of discourse.

Triangular membership functions are modeled on the boundary values for each of the inputs. Each vertex of a triangle falls directly onto a boundary value and the triangle is overlapping 50% of its area with the adjacent triangles (25% each with the right and left adjacent triangles) accept for the leftmost and rightmost boundary values where one side of the triangle is overlapping 25% of its area with the adjacent triangle while the other side is extended to infinity. This is shown in Figure 2 for inputs x and y.

Now, assume that the input values for x and y are x_0=2.5 and y_0=2.0 respectively. For the given x_0 and y_0, the degree of membership for x_0 and y_0 to their lower boundary (LB) and upper boundary (UB) values represented by $\mu_{LB}(x_0)$, $\mu_{UB}(x_0)$, $\mu_{LB}(y_0)$ and $\mu_{UB}(y_0)$ respectively can be obtained by matching x_0 and y_0 against their boundary values membership functions respectively as shown in Figure 3. From Figure 3, the boundary values for x_0 are LB_{x0}=2 and UB_{x0}=4 respectively. The degree of membership for x_0 to their boundary values are $\mu_{LB}(x_0)$=0.75 and $\mu_{UB}(x_0)$=0.25 respectively. For y_0, we have LB_{y0}=0 and UB_{y0}=2.5. The $\mu_{LB}(y_0)$ and $\mu_{UB}(y_0)$ are 0.2 and 0.8 respectively.

Mathematically, the $\mu_{LB}(i)$ and $\mu_{UB}(i)$ for an input i, in general, that falls in between two boundary values can be obtained by

$$\mu_{LB}(i) = 1 - \frac{i - LB_i}{|UB_i - LB_i|} \qquad (2)$$

$$\mu_{UB}(i) = \frac{i - LB_i}{|UB_i - LB_i|} \qquad (3)$$

where LB_i and UB_i are the lower and upper boundary values where i falls in between respectively. One can see that the sum of $\mu_{LB}(i)$ and $\mu_{UB}(i)$ is always equal to one.

THE REQUIRED ADDRESSES AND DATA

With two boundary-value conditions each for x and y, there are 2^2=4 different required input combinations or “addresses” to be considered for the final output. Four different addresses must be accessed to obtain four unique data through the look-up table. The four required addresses in binary format with x, referred to as the first input, in the most significant place followed by y, referred to as the second input, are A_{11}, A_{12}, A_{21}, and A_{22}; where A_{ij} represents the address obtained by the combination of the first input’s lower (if i=1) or upper (if i=2) boundary condition and the second input’s lower (if j=1)

or upper (if j=2) boundary condition. The four unique data are D_{11}, D_{12}, D_{21}, and D_{22}; where d_{ij} represents the data obtained by the combination of the first input's lower (if i=1) or upper (if i=2) boundary condition and the second input's lower (if j=1) or upper (if j=2) boundary condition address. Each data has two degrees of membership associated with it. One for the input x and the other for the input y. Table 1 shows the summary for the above information together with the associated degree of membership for each data.

FINAL OUTPUT

In general, for a two-input single-output controller, the final output, O_f, can be obtained by

$$O_f = \mu_{11}[\mu_{21}d_{11} + \mu_{22}d_{12}] + \mu_{12}[\mu_{21}d_{21} + \mu_{22}d_{22}] \quad (4)$$

where μ_{ij} is the degree of membership for the i-th input to its lower (if j=1) or upper (if j=2) boundary value and d_{ij} represents the data obtained by the combination of the first input's lower (if i=1) or upper (if i=2) boundary condition and the second input's lower (if j=1) or upper (if j=2) boundary condition address. Equation (4) can be further extended to accommodate a controller with more inputs. For example, for a three-input single-output controller, Equation (4) becomes

$$\begin{aligned} O_f = {} & \mu_{11}[\mu_{21}\mu_{31}d_{111} + \mu_{21}\mu_{32}d_{112} + \mu_{22}\mu_{31}d_{121} + \mu_{22}\mu_{32}d_{122}] \\ & + \mu_{12}[\mu_{21}\mu_{31}d_{211} + \mu_{21}\mu_{32}d_{212} + \mu_{22}\mu_{31}d_{221} + \mu_{22}\mu_{32}d_{222}] \end{aligned} \quad (5)$$

where d_{ijk} represents the data obtained by the combination of the first input's lower (if i=1) or upper (if i=2) boundary condition, the second input's lower (if j=1) or upper (if j=2) boundary condition, and the third input's lower (if k=1) or upper (if k=2) boundary condition address.

MICROPROCESSOR SOFTWARE PROCESS

The process of determining the degree of membership, obtaining data from the look-up table, and calculating the final output discussed in the previous sections can be accomplished by programming a microprocessor. The microprocessor software process first reads the inputs through the A/D converters. These inputs are then matched against the boundary values membership functions to find the degrees of membership to their boundary values. The process then proceeds to obtain the data for the required combination addresses (obtained from the different combinations of the boundary-value conditions) through the existing look-up table. Finally, with all the data and their associated degrees of membership, the final output of the control signal is obtained. The overall flowchart of the software process is shown in Figure 4.

MLUT FUZZY LOGIC CONTROLLER HARDWARE ARCHITECTURE

There are two possible hardware implementations with this technique. The first implementation is with a microprocessor such as the Motorola's MC68HC11 wired in a single chip mode. The on-chip electrically erasable programmable ROM (EEPROM) is used as the look-up table to store the necessary data. Data can be programmed into the

EEPROM or erased from the EEPROM under software control. Figure 5 shows the block diagram of this system.

The second implementation is with the microprocessor wired in an expanded mode. An external PROM or PROMs are used as the look-up table device. Figure 6 shows the block diagram for this system. If a MC68HC11 microprocessor is used, external A/D converters are not needed in both implementations since an on-chip A/D converter is included in the MC68HC11.

SIMULATIONS

The classical example of the backing truck to ramp example by Kong and Kosko [5][6][7] is examined. Simulations are performed for different resolutions for the MLUT fuzzy logic controller. When using a look-up table the fuzzy logic's interpolation capabilities are shown to work well with a fairly low number of bits.

Figure 7 shows the geometry of the simulated truck and ramp in the classical example of the backing truck to ramp example. The input fuzzy variables are x (distance) and ϕ (angle) of the truck's position. The output fuzzy variable is θ, the truck's steering-angle signal. Assume enough clearance between the truck and the ramp so the truck's y-coordinate, y, could be ignored. The x-coordinate ranges from -50 to 50, ϕ ranges from -180 to 180, and θ ranges from -30 to 30. Positive values of θ represented clockwise rotations of the steering wheel, while negative values represented counterclockwise rotations.

The values and the membership functions for the inputs and output are given in Figure 8. Table 2 shows the Fuzzy Associative Memory (FAM) of the fuzzy rules used in the system. A total of 35 (5×7) fuzzy rules are applied to the system.

The original control surface using the Kong and Kosko's controller is shown in Figure 9 with the corresponding truck trace shown in Figure 10. The truck always starts with the same 35 positions corresponding to the center of the 35 fuzzy rules.
In all of the following examples, 8 bits of output resolution are used for the D/A converter. A classical fuzzy controller was used to generate the contents of the look-up table. First consider the case of 4 bits (a resolution of 16) for each of the two inputs. In this case, a 8-bit address is required and the total number of byte-size storage locations in the look-up table (assuming 8 bits of output resolution for the D/A converter) is 2^8=256 locations. Using 4 bits for each input, the corresponding control surface is shown in Figure 11 and the truck trace is shown in Figure 12. The control surface and corresponding truck trace for 5 bits (a resolution of 32) for each input are shown in Figure 13 and Figure 14, respectively. Finally, with 6 bits (a resolution of 64) for each input, the control surface and the corresponding truck trace are shown in Figure 15 and Figure 16, respectively.

As one can see from the simulations, 4 bits (a resolution of 16) for each input provide reasonable accuracy. In this case, the required storage locations are 2^8=256 byte size locations, and this is a relatively small memory. One can see that it is practical to increase the number of inputs to 4. For the case of 4 bits for each input, the memory address is 16 bits which corresponds to 2^{16} = 64 K byte-size storage locations.

CONCLUSION

A hardware implementation technique for a multi-input single-output MLUT system fuzzy logic controller has been presented. This approach yields higher speeds compared to other fuzzy controllers that use software implementations with a personal computer or a workstation. Simulation results show that the fuzzy logic's interpolation capabilities work well with a fairly low number of bits, that is, 4 to 6 bits. Distinctive features of the implementation of the MULT system fuzzy logic controller can be summarized as follows

- The processing time is independent of the size of the knowledge based and the number of labels (terms) used in the system.
- Microprocessors such as MC68HC11s offer an affordable technology; they are relatively inexpensive devices. These devices can be programmed using a PC or workstation by the user.
- With user programmable and configurable capabilities, microprocessors allow design changes for a more optimal solution. This results in significant cost savings by reducing the risk of hardware design changes.
- The design is compact and expandable. Any multi-input multi-output fuzzy controller can be realized into n number of multi-input single output fuzzy controllers, where n is the number of outputs.
- The design has no device and technology restriction.
- The off-line software can be modified to allow for design changes and for other types of fuzzy control systems.

REFERENCES

[1] Togai, M., Watanabe, H., (1985). "A VLSI implementation of a fuzzy inference engine: Toward an expert system on a chip," Proc. of 2nd Int. Conf. on AI and Applications, Dec, 192-197.

[2] Yamakawa, T., (1988). "High-Speed Fuzzy Controller Hardware Systems: The Mega FLIPS Machine," Information Science, Elsevier Science Publishing Company, Inc., 113-128.

[3] Tan, H. S., Sandige, R. S., Wilamowski, B. M., (1994). "Hardware Implementation of PLD-based Fuzzy Logic Controllers Using a Look-up Table Technique," Intelligent Engineering System Through Artificial Neural Networks, ASME Press, (4), 89-94.

[4] Motorola, Inc., (1992). "Fuzzy Logic Education Program," Motorola Inc.

[5] Kong, S., Kosko, B., (1992). "Adaptive fuzzy system for backing up a truck-and-trailer," IEEE Trans. on Neural Networks, (3), March, 211-223.

[6] Kosko, B., (1992). Neural Networks and Fuzzy Systems - A Dynamical Systems Approach to Machine Intelligence, Prentice Hall.

[7] Wilamowski, B. M., Sandige, R. S., (1993). "Trainable Fuzzy Controller," Intelligent Engineering System Through Artificial Neural Networks, ASME Press, (3), 561-566.