Fifth IFSA World Congress (1993), 781 - 784

VLSI Implementations of Fuzzy Logic Finite State Machines

Janos Grantner[1] and Marek J. Patyra[2]

[1] Department of Process Control, Technical University of Budapest, Budapest, 1111 Hungary
[2] Department of Computer Engineering, University of Minnesota, Duluth, MN 55812, USA

ABSTRACT: Most linguistic models of processes or plants known are essentially static, that is, time is not a parameter in describing the behavior of the object's model. In this paper we show two models for synchronous finite state machines (FSM) based on fuzzy logic, namely the Crisp-State-Fuzzy-Output (CSFO FSM) and Fuzzy-State-Fuzzy-Output (FSFO FSM). As a result of the introduction of the FSM models, the improved architectures for fuzzy logic controller have been defined. These architectures featuring pipelined intelligent fuzzy controller are discussed in terms of dimensionality of the model. VLSI integrated circuit implementation issues of the fuzzy logic controller are also considered. The presented approach can be utilized for fuzzy controller hardware accelerators intended to work in the real-time environment.

1. FUZZY LOGIC FINITE STATES MACHINES

The general model of a finite state machine (FSM) is illustrated in Figure 1. Formally, a sequential circuit is specified by two sets of Boolean logic functions: f_Z (X, y) $\rightarrow$ Z, and f_y (X, y) $\rightarrow$ Y; where X, Z, y, and Y stand for a finite set of inputs, outputs, the present and next states of the state variables, respectively. Function f_Z maps the inputs and the present state of the state variables to the outputs, and function f_y maps the inputs and the present state of the state variables to the next state of the state variables.

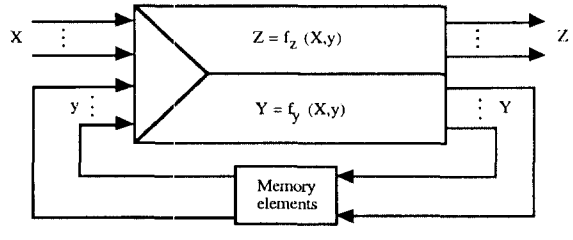

Figure 1 General Model of a Finite State Machine (FSM).

The current states of the memory elements hold information on the past history of the circuit. The behavior of a synchronous sequential circuit can be defined from the knowledge of its signals at discrete instants of time. Those time instants are determined by a periodic train of clock pulses. The memory elements hold their outputs until the next clock pulse arrives. We extended this model [2] by introducing membership functions and fuzzy relations to map the changes which take place in fuzzy input data to both the fuzzy outputs and next state of the state variables. The definition of states remain crisp, that is, the state of the system can be represented in one of the usual ways (i.e. by isolated flip-flops, registers or a microprogrammed control unit). The fuzzy outputs are devised from a dynamically changing linguistic model since the response to a specific change at the fuzzy input will vary with different states of the FSM. We are refer to this model as Crisp-State-Fuzzy-Output FSM or CSFO FSM. A block diagram of the CSFO FSM is shown in Figure 2. X and Z stand for a finite set of fuzzy inputs and outputs, respectively.

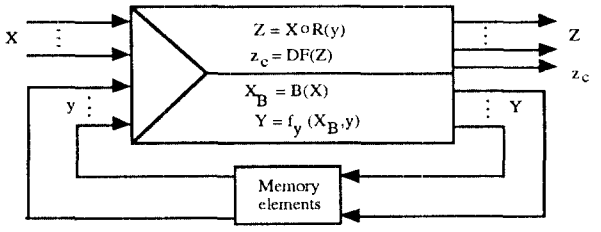

Figure 2 General Model of the Crisp-State-Fuzzy-Output Finite State Machine (CSFO FSM).

R stands for the object's model which is now function of the y present state of the state variables, and "o" is the composition operator. The z_C crisp values of the fuzzy outputs are obtained by applying the defuzzification strategy DF. B stands for the transformation which maps the linguistic values of the X linguistic (fuzzy) variables to the X_B Boolean (two-valued) logic variables. Function f_y maps both the X_B Boolean logic variables and the y present state variables to the Y next state of the state variables. The object (linguistic) model R and the input variables X are used to derived the fuzzy outputs Z and the next state of the state variables. The fuzzy outputs Z are obtained from dynamically changing linguistic model R, since the response to the specific change of the fuzzy input X is not the same in different states of the CSFO FSM.

It is possible to extend the introduced CSFO FSM model even further. Hence, the Fuzzy-State-Fuzzy-Output finite state

machine is proposed (FSFO FSM). In this case the machine's state is represented by a fuzzy state, which can be defined by a fuzzy number.

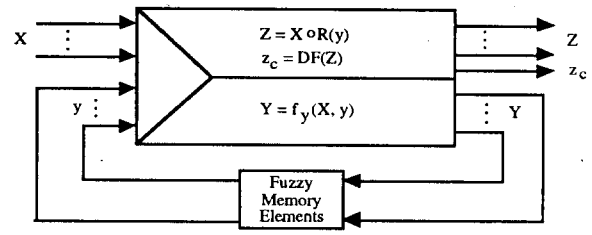

Figure 3 General Model of the Fuzzy-State-Fuzzy-Output Finite State Machine (FSFO FSM).

Now, fuzzy outputs are derived from fuzzy input variables X and present state of the state variables (fuzzy). Variables y_F and YF representing the present and the next state of the fuzzy state variable are stored in the fuzzy memory elements (Figure 3). FSFO FSM model can be implemented using CSFO FSM and the additional memory storing the State Membership Functions (SMF) (Figure 3). With the FSFO FSM it cannot be said that the system is in a particular crisp state, rather it is some different states simultaneously. Each state is occupied to a certain degree defined by the SMF. Thus the FSFO FSM can be considered as a general class of Fuzzy Logic Finite State Machines based on two-valued logic.

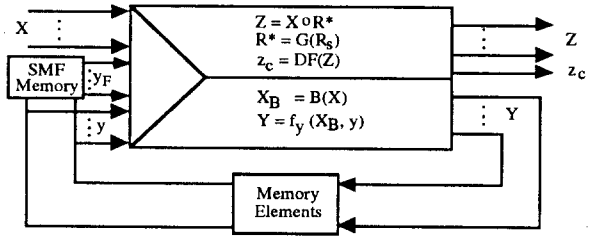

Figure 4 The Model of FSFO FSM implemented using CSFO FSM and SMF.

In Figure 4 the R^* stands for the composite linguistic model in a given fuzzy state and it is derived from all overall rules of the CSFO FSM by means of State Membership Functions.

2. FUZZY MODEL

A linguistic model of a process can be built using a specialized software; fuzzy inference and defuzzification strategies can also be computed without using any dedicated hardware. However, in the case of real-time control applications, the pure software approach may not offer sufficient performance. Later we discuss a hardware accelerator for a fuzzy logic controller. The accelerator is based upon the mathematical model of fuzzy inferencing. Three variants of such model are discussed below.

The process operation control strategy is created by analysis of input and output values, in which not only measurable quantities are taken into account but also parameters which cannot be measured, only observed [1]. On the basis of the verbal description, which is called a *linguistic model*, a fuzzy relation R is created:

$$R = \overset{N}{\underset{I=1}{*}} (XI \rightarrow YI)$$

In this formula $\rightarrow$ denotes the operation or operations by which fuzzy implications are defined, and the symbol * represents an operation which can be interpreted as the sentence connective ALSO. We shall present the algorithm not only intended for creating a fuzzy model when a verbal description is given, but also for determining the model's answer to a given input [2]. The verbal description of the process contains N relations, and fuzzy sets describe the particular states which occur in the verbal description of inputs $X^{(1)}$, $X^{(2}$, and $X^{(M)}$ and output Z be given by:

R_1: IF $X^{(1)}$ IS ... ($X^{(1)}1$) AND $X^{(2)}$ IS ... ($X^{(2)}1$)... AND $X^{(M)}$ IS ... ($X^{(M)}1$) THEN Z IS ... (Z1)

:

R_N: IF $X^{(1)}$ IS ... ($X^{(1)}N$) AND $X^{(2)}$ IS ... ($X^{(2)}N$)... AND $X^{(M)}$ IS ... ($X^{(M)}N$) THEN Z IS ... (ZN).

Let us now review some major aspects of fuzzy inference for different model dimensions.

A. Single-Input-Single-Output

Fuzzy Learning

A method of creating fuzzy relation R_1 which represents the first fuzzy implication in the verbal description is interpreted as intersection. The remaining relations R_2, R_3,..., R_N are created analogously by application of the same definition of fuzzy implication.

$$R_1 = X1 \times Z1 \qquad \text{(EQ 1)}$$

$$\forall(u,w) \in U \times W \quad R_1(u,w) = \min(X1(u), Z1(w)) \qquad \text{(EQ 2)}$$

The final relation R (being the object's model) is obtained as the union of R_1, R_2,..., R_N, since the sentence connective ALSO is defined as union:

$$R = R_1 \cup R_2 \cup ... \cup R_N \qquad \text{(EQ 3)}$$

$$\forall(u,w) \in U \times W \quad R(u,w) = \max[R_1(u,w), R_2(u,w),..., R_N(u,w)] \qquad \text{(EQ 4)}$$

Fuzzy Inference

The method of creating fuzzy answer Z to a fuzzy input X is to apply max-min composition.

$$\forall w \in W \quad Z(w) = \max[\min(X(u), R(u,w))] \qquad \text{(EQ 5)}$$

B. Double-Input-Single-Output

The inputs are normalized into the same universe of discourse.

Fuzzy Learning

A method of creating fuzzy relation R1 which represents the first fuzzy implication can be derived by the decomposition of the first rule into two parts:

$$R_1^{(1)} = X1^{(1)} \times Z1 \qquad \text{(EQ 6)}$$

and:

$$R_1^{(2)} = X1^{(2)} \times Z1 \qquad \text{(EQ 7)}$$

In the enhanced form (EQ 6) and (EQ 7) can be rewritten as:

$$R_1^{(1)}(u, w) = \min(X1^{(1)}(u) \times Z1(w)) \qquad \text{(EQ 8)}$$

$$R_1^{(2)}(u, w) = \min(X1^{(2)}(u) \times Z1(w)) \qquad \text{(EQ 9)}$$

The remaining relations R_2, R_3,..., R_N are created analogously by application of the same decomposition technique. The overall rule for the $X^{(1)}$ and Z is then given by:

$$R^{(1)} = R_1^{(1)} \cup R_2^{(1)} \cup ... \cup R_N^{(1)} \qquad \text{(EQ 10)}$$

and for $X^{(2)}$ and Z by:

$$R^{(2)} = R_1^{(2)} \cup R_2^{(2)} \cup ... \cup R_N^{(2)} \qquad \text{(EQ 11)}$$

The enhanced forms of (EQ 6) and (EQ 6) are given below:

$$R^{(1)}(u, w) = \max\{R_1^{(1)}(u, w) \cup ... \cup R_N^{(1)}(u, w)\} \quad \text{(EQ 12)}$$

$$R^{(2)}(u, w) = \max\{R_1^{(2)}(u, w) \cup ... \cup R_N^{(2)}(u, w)\} \quad \text{(EQ 13)}$$

As a result two overall rules coexists in this model.

Fuzzy Inference

Having two overall rules the model output can be obtained using the min-superposition of two relations: input X and rule R:

$$Z = \min\{X^{(1)} \circ R^{(1)}), (X^{(2)} \circ R^{(2)})\} \quad \text{(EQ 14)}$$

C. Multiple-Input-Single-Output

Fuzzy Learning

It is assumed that the first and following rules can be decomposed into M separate subrules as follows: (for the first rule)

$$R_1^{(1)} = X1^{(1)} \times Z1 \quad \text{(EQ 15)}$$

$$\vdots$$

$$R_1^{(M)} = X1^{(M)} \times Z1 \quad \text{(EQ 16)}$$

In the enhanced form (EQ 6) through (EQ 7) can be rewritten as:

$$R_1^{(1)}(u, w) = \min(X1^{(1)}(\upsilon) \times Z1(w)) \quad \text{(EQ 17)}$$

$$R_1^{(M)}(u, w) = \min(X1^{(M)}(\upsilon) \times Z1(w)) \quad \text{(EQ 18)}$$

for the Nth rule:

$$R_N^{(1)} = X1^{(1)} \times Z1 \quad \text{(EQ 19)}$$

$$\vdots$$

$$R_N^{(M)} = X1^{(M)} \times Z1 \quad \text{(EQ 20)}$$

The overall rule for the X and Z is then given by:

$$R^{(1)} = R_1^{(1)} \cup R_2^{(1)} \cup ... \cup R_N^{(1)} \quad \text{(EQ 21)}$$

$$\vdots$$

$$R^{(M)} = R_1^{(M)} \cup R_2^{(M)} \cup ... \cup R_N^{(M)} \quad \text{(EQ 22)}$$

The enhanced forms of (EQ 6) through (EQ 6) are given below:

$$R^{(1)}(u, w) = \max\{R_1^{(1)}(u, w) \cup ... \cup R_N^{(1)}(u, w)\} \quad \text{(EQ 23)}$$

$$\vdots$$

$$R^{(M)}(u, w) = \max\{R_1^{(M)}(u, w) \cup ... \cup R_N^{(M)}(u, w)\} \quad \text{(EQ 24)}$$

Fuzzy Inference

In this case the model output can be obtained using the min-superposition of all M relations: input $X^{(i)}$ and rule $R^{(i)}$:

$$Z = \min\{X^{(1)} \circ R^{(1)}), ..., (X^{(M)} \circ R^{(M)})\} \quad \text{(EQ 25)}$$

In each state of the CSFO FSM just one R overall rule memory is selected, there are s instances of R total for s states with the SISO configuration. With the DISO controller, a pair of R(1) and R(2) overall rule memories are assigned to each state, and finally, in the MISO case, a set of R(1), ..., R(M) rule memories belongs to each state. Due to a compact implementation of the knowledge base, the memory requirement of storing a R overall rule is about 1/4 kilobyte with the digitized membership function employed [6]. The rule memory need of the FSFO FSM model is the same as that of the CSFO FSM one.

D. Defuzzification Strategy

The defuzzification strategy is common for all described versions of the fuzzy model. The z_c deterministic value of the answer (crisp value) is determined using the formula:

$$Z_c = \frac{1}{L} \sum_{J=1}^{L} w \quad \text{(EQ 26)}$$

where L is the number of points $w_J \in W$ in which output set Z reaches a maximum.

3. FUZZY-TO-BOOLEAN MAPPING (B) ALGORITHM

The general form of the B algorithm can be used to convert either fuzzy inputs or outputs to sets of Boolean variables. The mapping of a single fuzzy input X to a set of Boolean input variables XB is illustrated in Figure 5.

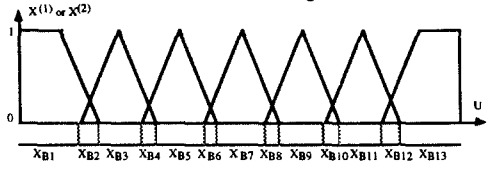

Figure 5 Mapping of fuzzy input X to a set of Boolean input variables X_B.

The universe of discourse is broken up into k overlapping sub-intervals where k is the number of linguistic values. These linguistic sub-intervals are mapped to n disjoint sub-intervals associated with Boolean variables where n = 2k-1. A Boolean input variable X_{Bi}=1 if the position of the input maximum falls into Boolean sub-interval i (i=1,..., 2k-1), and all other X_{Bj}=0 (j ≠ i, j=1,..., 2k-1). In case of the MISO model, multiple sets of Boolean inputs are created. The state transients of either the CSFO or the FSFO FSM models can be specified in terms of a sequence of changes at the X_B Boolean inputs.

4. HARDWARE IMPLEMENTATION

The hardware accelerator which performs the fuzzy learning, fuzzy inference, and defuzzification computation, that is, which maps the fuzzy inputs to fuzzy and/or crisp outputs, is summarized in this section. A SISO CSFO FSM model is assumed for further discussions and just the active overall rule memory section of the whole knowledge base will be shown. The accelerator consists of four basic units: the interface, the Fuzzy-to-Boolean mapping unit, the combined fuzzy model/fuzzy inference unit, and the defuzzifier unit. The last two are referred to as the fuzzy engine [6]. The functional block diagram of the accelerator is shown in Figure 6. To achieve a high processing rate for real time applications, the units are connected in a four-level pipeline.

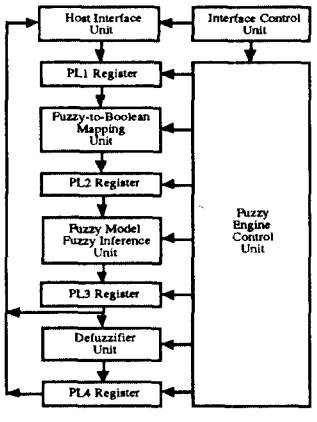

Figure 6 Pipeline Architecture of the Hardware Accelerator.

The core of the hardware accelerator is a fuzzy engine which implements the equations (EQ 2), (EQ 5), and (EQ 26). It is split into the fuzzy model/fuzzy inference unit and the defuzzifier unit. The functional block diagram of the fuzzy model/fuzzy inference unit (without increased parallelism) is shown

in Figure 7.

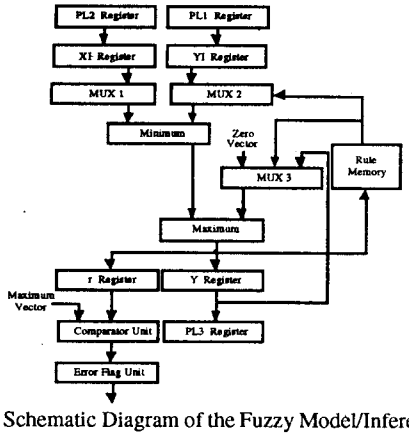

Figure 7 Schematic Diagram of the Fuzzy Model/Inference Unit.

Contents of the R rule memory remain unchanged during the fuzzy inference process. After the last clock step, register Y holds the result of the XoR operation in the digitized fuzzy data format. To detect whether the condition: $\forall(u,w) \in U \times W$, $R(u,w) = 1$ is met, an error flag was added to the fuzzy engine.

Due to the linear property of the max-min composition, by quadrupling the functional units of the basic architecture, the time required to complete the pipeline steps for either the fuzzy learning or the fuzzy inference process can be reduced to $[u_{max} + 4] + 2$ clock periods. Since the precedence relation of the subtasks (I/O data transfer (T_1), the pre-processing of the multiple fuzzy inputs (T_2), the learning of a new rule or the performing a fuzzy inference operation (T_3), and the defuzzification (T_4)) are all linear operations, the four basic units of the hardware accelerator form a linear pipeline. The pipeline architecture allows the simultaneous operation of the four units.

The defuzzifier unit performs the defuzzification operation. Finding both the maximum and its position takes at most 4 clock periods. Two parallel adder networks are used to sum up the position codes of the maximum and obtain the total number of maximum simultaneously. Then, the crisp value is taken from the look-up table. If maximum value is zero then the crisp value is flagged.

5. VLSI IMPLEMENTATION ISSUES

Although fuzzy logic has been successfully applied to control problems over the last 10 years, extensive integrated circuits implementations of fuzzy logic circuits have yet to be expected. Few interesting approaches to the implementation of fuzzy logic circuits have been published recently: [4], [8], and [9].

There are two possible versions of the fuzzy logic controller that could be useful in most practical implementations: a controller working stand alone (SA), or with an appropriate host computer (HC). These options will be taken into consideration.

Let us discuss the VLSI implementation issues in more detail. We assume that the proposed fuzzy controller will have three basic cycles of operation: fuzzy learning, fuzzy inference and stand-by. In case of the fuzzy learning and fuzzy inference operations the HC version will be supplied with fuzzy data through the host computer which performs the fuzzification of the analog inputs. It is obvious that HC version will be able to process only digital representation of the fuzzy data prepared by the host computer.

The SA version of the VLSI implementation will input the analog data and perform the fuzzification operation by itself. The stand-by mode will be common for both versions. There are other several vertsion-specific implementation issues which are discussed more elaboratively in [2].

6. CONCLUSIONS

The general model for fuzzy state machine used to formulate fuzzy controllers for event-driven real-time systems is described. An improved architecture for fuzzy logic controller is defined based on the introduced Crisp-State-Fuzzy-Output Finite State Machine (CSFO FSM). The defined architecture provides a novel strategy for fuzzy model building enabling fuzzy inferences to be performed in a single stage of a hardware accelerator, an ability not common to previously published architectures. The Fuzzy-State-Fuzzy-Output Finite State Machine (FSFO FSM) is introduced as an extension to the CSFO FSM. These proposed architecture for the fuzzy controller hardware accelerator, appropriately pipelined, reaches the speed of at least few M FLIPS and have the ability to work in a real time environment. The VLSI Implementation issues of a proposed architecture are discussed.

7. REFERENCES

[1] FC 110 Digital Fuzzy Processor DFP™, Togai InfraLogic, Inc., October 1991.

[2] J. Grantner, M. Patyra, and M. Stachowicz, "Architecture for Event-Driven Intelligent Fuzzy Logic Controller ", Proceedings of *IEEE International Conference on Fuzzy Systems*, San Francisco, CA, pp. 273-278, March/April 1993.

[3] M.J. Patyra, "VLSI Implementation of Fuzzy-Logic Circuits", *International Fuzzy Systems Association World Congress*, Brussels, Belgium, June 1991.

[4] M.Sasaki, F.Ueno, and T.Inoue, "7.5MFLIPS Fuzzy Microprocessor Using SIMD and Logic-in-Memory Structure", Proceedings of *IEEE International Conference on Fuzzy Systems*, San Francisco, CA, pp.527-534, March/April 1993.

[5] M. Stachowicz, J.Grantner, L. Kinney, "Two-Valued Logic for Linguistic Data Acquisition", *NAFIPS Workshop '91*, University of Missouri-Columbia, pp. 168-172, May 14-17, 1991.

[6] M. Stachowicz, J.Grantner, L. Kinney, "Pipeline Architecture Boosts Performance of Fuzzy Logic Controller", *IFSICC'92 International Fuzzy Systems and Intelligent Control Conference*, Louisville, Kentucky, March 15-18, pp. 190-198, 1992.

[7] M. Togai, H.Watanabe, "Expert System on a Chip: An Engine for Real-Time Approximate Reasoning", *IEEE Expert*, pp. 55-62, Fall 1986.

[8] A.Ungering, K.Thuener, and K.Goser, "Architecture of a PDM VLSI Fuzzy Logic Controller with Pipelining and Optimized Chip Area", Proceedings of *IEEE International Conference on Fuzzy Systems*, San Francisco, CA, pp. 447-452, March/April 1993

[9] H.Watanabe, W.Dettloff, K.Yount, "A VLSI Fuzzy Logic Controller with Reconfigurable, Cascadable Architecture", *IEEE Journal of Solid-State Circuits*, vol. 25, pp. 376-381, 1990.

[10] L.Zadeh, "A Fuzzy Algorithmic Approach to the Definition of Complex or Imprecise Concepts", *International Journal Man Machine Studies*, vol. 8, pp. 249-291, 1976.