INTELLIGENT FUZZY CONTROLLER FOR EVENT-DRIVEN REAL TIME SYSTEMS

Janos Grantner[1]
Department of Electrical Engineering
University of Minnesota, Minneapolis, MN 55455

Marek Patyra and Marian S. Stachowicz
Department of Computer Engineering
University of Minnesota, Duluth, MN 55812

ABSTRACT

Most linguistic models 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 a model for synchronous finite state machines based on fuzzy logic. Such finite state machines can be used to build both event-driven time-varying rule-based systems and also the control unit section of a fuzzy logic computer. The architecture of a pipelined intelligent fuzzy controller is presented, and the linguistic model is represented by an overall fuzzy relation stored in a single rule memory. A VLSI integrated circuit implementation of the fuzzy controller is suggested. At a clock rate of 30 MHz, the controller can perform 3 MFLIPS on multi-dimensional fuzzy data.

KEYWORDS: Fuzzy Modeling, Intelligent Fuzzy Controller, Fuzzy Logic Hardware Accelerator, VLSI Implementation

1. FUZZY LOGIC FINITE STATE MACHINES

The general model of a finite state machine (FSM) is illustrated in Figure 1.1. Formally, a sequential circuit is specified by two sets of Boolean logic functions:

$$f_Z (X, y) \rightarrow Z, \text{ and}$$
$$f_y (X, y) \rightarrow Y,$$

where X, Z, y, and Y stand for a finite set of inputs, outputs, present and next state of the state variables, respectively. Functions f_Z and f_y map the inputs and the present states of the state variables to the outputs and the next states of the state variables, respectively.

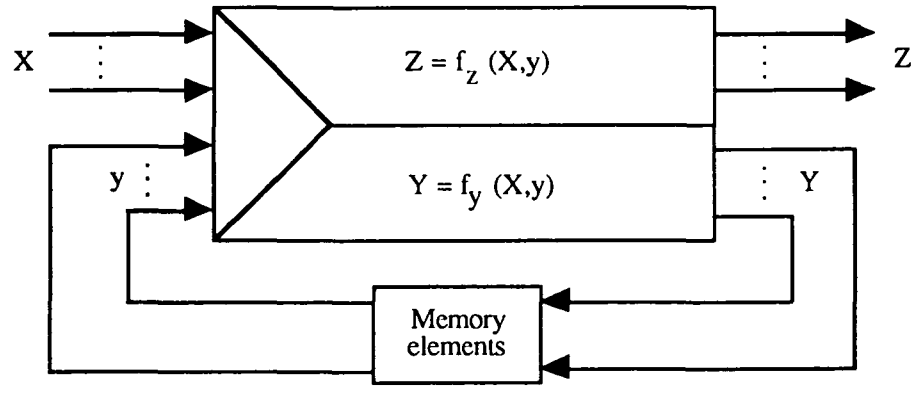

Figure 1.1 General model of a finite state machine (FSM).

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.

---

[1] *Visiting Professor from Technical University of Budapest, Hungary.*

We extend this model by introducing membership functions and fuzzy relations to map the changes which take place in fuzzy input data to fuzzy outputs and next states of the state variables.
With the model presented in this paper, the definition of states will 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 will be devised from a dynamically changing linguistic model since the response to a specific change at the fuzzy inputs will vary with different states of the FSM. We will refer to this model as Crisp-State-Fuzzy-Output FSM or CSFO FSM. A block diagram of the CSFO FSM is shown in Figure 1.2. X and Z stand for a finite set of fuzzy inputs and outputs, respectively.

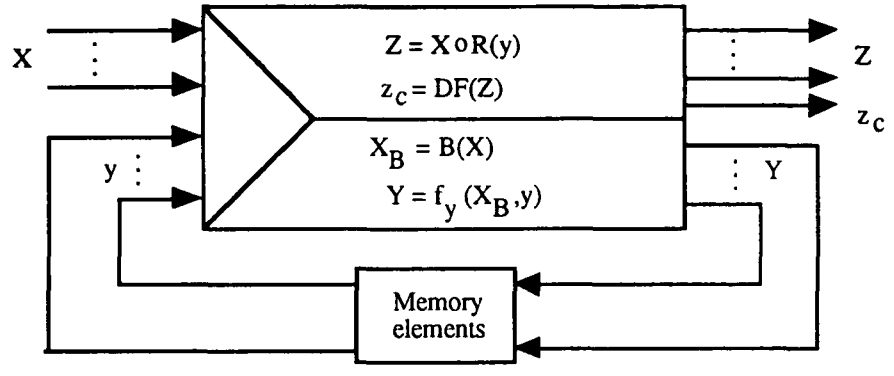

Figure 1.2 General model of the Crisp-State-Fuzzy-Output FSM (CSFO FSM).

R stands for the object's model which is now function of the y present states of the state variables, and o is the operator of composition. The z_C crisp values of the fuzzy outputs are obtained by computing the DF defuzzification strategy. 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 also the y present states to the Y next states of the state variables.
To accelerate the mapping of the fuzzy inputs X to a new set of fuzzy and crisp outputs Z and z_C, respectively, (i.e. to compute fuzzy inference and the DF defuzzification strategy) our pipelined fuzzy logic hardware accelerator model [5] will also be employed with the CSFO FSM. The next states of the state variables will be devised from the present states and the X_B Boolean logic variables. For instance, a Boolean variable X1LOW is true if the position of the maximum in the membership function for linguistic variable X1 falls in the range 1 to 5. X1LOW is otherwise false.
The state transients will be completed simultaneously with the fuzzy pre-processing pipeline step (Figure 3.3). A new S_K state of the CSFO FSM will then select an overall fuzzy relation R_K which will in turn be used as the linguistic model in the fuzzy inference pipeline step while the system is in state S_K. With this model, the state variables will take their new values at the rate at which the pipeline steps proceed. The fuzzy outputs will be defuzzified in the last pipeline step.
In the course of the learning process (eq. 2.3), an overall relation R_I is created for each state S_I (I = 1,..., N) of the CSFO FSM.

2. ALGORITHM OF CREATING A MULTIPLE-INPUT FUZZY MODEL

A linguistic model of a process can be built by software; fuzzy inference and defuzzification strategies can also be computed without using any dedicated hardware. However, in case of real-time control applications, the pure software approach may not be sufficient. We suggest a hardware accelerator for a multiple-input fuzzy logic controller. The accelerator is based upon the mathematical model as follows.
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:

$$R = \ast_{I=1}^{N} (XI \rightarrow YI). \qquad (2.1)$$

In formula (2.1) $\rightarrow$ is a symbol of the operation or operations by which fuzzy implications are defined, and the symbol $*$ represents an operation which interprets the sentence connective ALSO.
We shall present the algorithm not only intended for creating a fuzzy model with a given verbal description is given, but also for determining the model's answer to a given input [2].
The verbal description of the process performance contains N relations, and fuzzy sets describe the particular states which occur in the verbal description of inputs $X^{(1)}$ and $X^{(2)}$ and output Y be given in formula (2.2). The graphic interpretations [4] of fuzzy sets $X^{(1)}$, $X^{(2)}$, and Y are illustrated in Figure 2.1.

$$\begin{aligned}&\text{R1: IF } X^{(1)} \text{ is very small } (X^{(1)}1) \text{ AND } X^{(2)} \text{ is medium } (X^{(2)}1) \text{ THEN Y is medium (Y1)}\\ &\quad \text{ALSO}\\ &\quad \vdots\\ &\text{RN: IF } X^{(1)} \text{ is very big } (X^{(1)}N) \text{ AND } X^{(2)} \text{ is medium } (X^{(2)}N) \text{ THEN Y is medium (YN)}\end{aligned} \tag{2.2}$$

The paragraphs below illustrate in turn:
Fuzzy Learning
A method of creating fuzzy relation R1 which represents the first fuzzy implication in the verbal description is interpreted as intersection. The remaining relations R2, R3,..., RN are created analogously by application of the same definition of fuzzy implication.

$$\begin{gathered}R1 = X1 \times Y1\\ \forall (u,w) \in U \times W \; R1(u,w) = \min\,(X1(u), Y1(w))\\ \forall u \in U \; X1(u) = \min(X^{(1)}(u), X^{(2)}(u))\end{gathered} \tag{2.3}$$

The final relation R (being the object's model) is obtained as the union of R1, R2,..., RN, since the sentence connective ALSO is defined as union.

$$\begin{gathered}R = R1 \cup R2 \cup \ldots \cup RN\\ \forall (u,w) \in U \times W \quad R(u,w) = \max(R1(u,w), R2(u,w),\ldots, RN(u,w))\end{gathered} \tag{2.4}$$

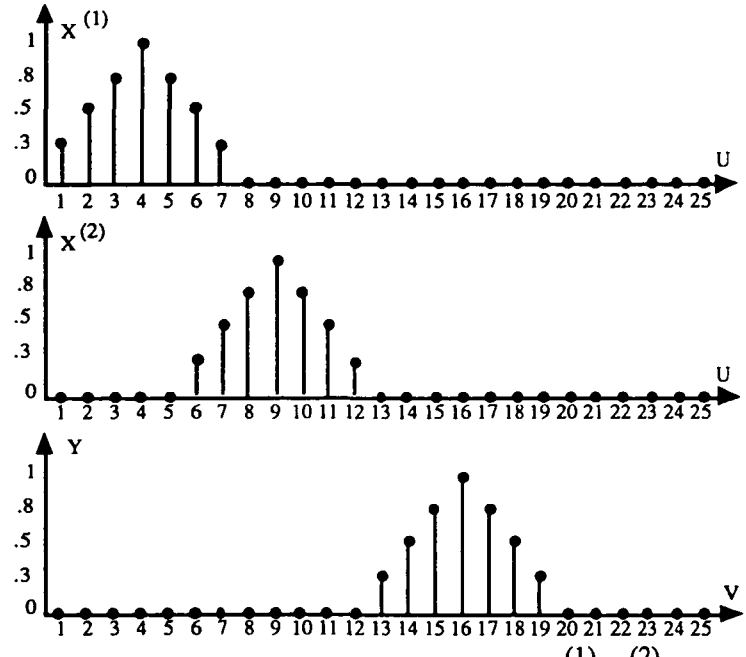

Figure 2.1. Graphic interpretations of fuzzy sets $X^{(1)}$, $X^{(2)}$, and Y.

Fuzzy Inference

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

$$\forall w \in W \; Y(w) = \max_{u \in U} \left(\min(X(u), R(u,w))\right) \tag{2.5}$$

Defuzzification Strategy

The deterministic value of the answer (crisp value) is determined using the formula

$$y_C = \frac{1}{L} \sum_{J=1}^{L} w_J \tag{2.6}$$

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

3. HARDWARE ACCELERATOR

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.

Currently, in our research the degree of membership function is a discrete valued function with a 5-element domain set. With two-valued logic, three bits are used to represent each element of the set. The number of levels can be extended up to eight. The universe of discourse of a fuzzy subset is limited to a finite set with 25 elements ($u_{max} = w_{max} = 25$). Seventy five bits are used for digitization of the membership function.

The accelerator consists of four basic units: the host interface, the fuzzy pre-processing unit, the combined fuzzy model/fuzzy inference unit, and the defuzzifier unit. The last two are referred to as the fuzzy engine [3]. The functional block diagram of the accelerator is shown in Fig. 3.1. To achieve a high processing rate for real time applications, the units are connected in a four-level pipeline.

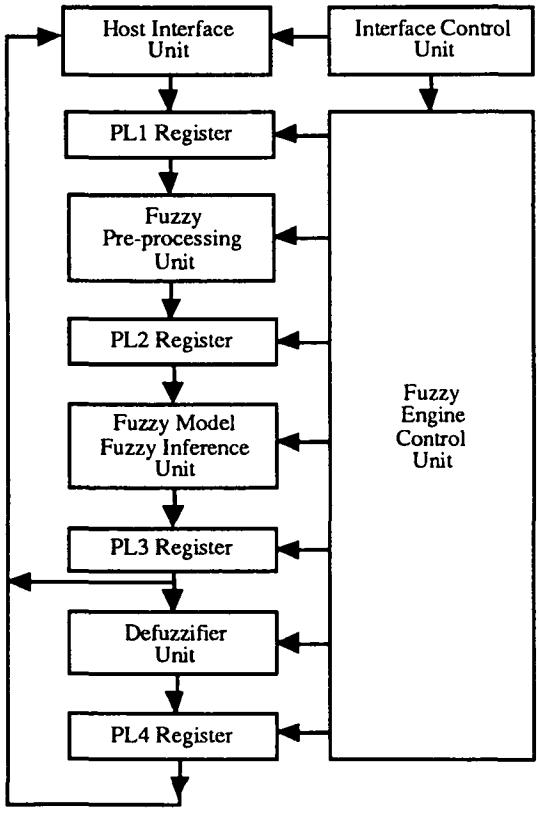

Figure 3.1. Pipeline architecture of the hardware accelerator.

The core of the hardware accelerator is a fuzzy engine which implements the formuli (eq. 2.4) to (eq. 2.6). 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 3.2.
After the XI and YI registers have been loaded, learning a multi-dimensional rule RK takes u_{max} clock periods. The MUX2 multiplexer at the input of the minimum unit selects the YI register. During the first clock step, u_1 is paired with all w elements of YI and these pairs are fed to the inputs of the minimum unit. If the current rule is the first in a learning sequence, throughout the learning cycle 0 (non membership) elements will be paired with the outputs of the minimum unit and fed to the inputs of the maximum unit. The whole word of maximum values is stored at the first location of the R rule memory. During the jth clock step, u_j is compared to all w elements of YI simultaneously and the vector of the max elements is stored in the jth location of R.
If the current rule is not the first one in the learning process, the MUX3 multiplexer at the input of the maximum unit selects the ith row of R($1 \le i \le u_{max}$) during the ith clock step and the contents of this row in R will be updated from the outputs of the maximum unit.

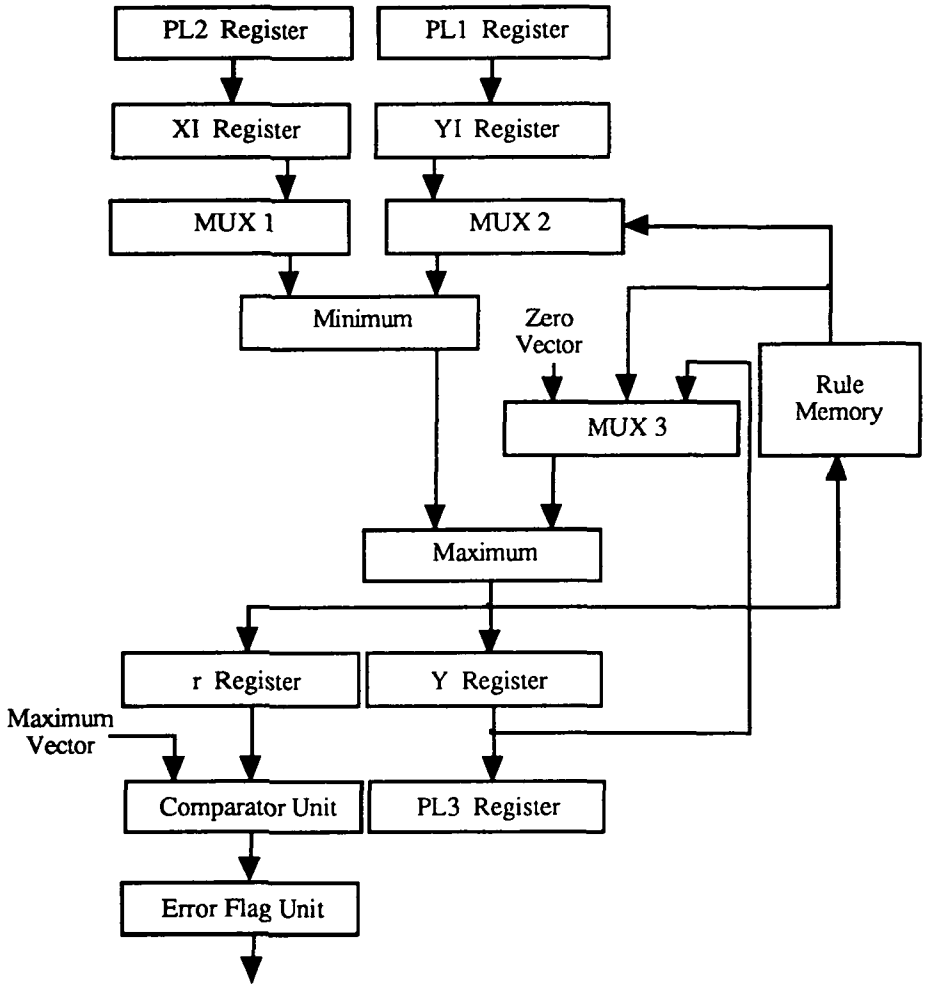

Figure 3.2 Functional block diagram of the fuzzy model/fuzzy inference unit.

Therefore the learning process of N rules takes $N \times u_{max}$ clock periods with the architecture shown in Figure 3.2. The clock steps needed to load registers XI and YI are ignored at this point. Computing the fuzzy inference (max-min composition) also takes u_{max} clock periods. This time the MUX2 multiplexer at the input of the minimum unit pairs the u_i element of XI with all r elements of the ith row in R. If i = 1 (first clock step), then the MUX3 multiplexer at the input of the maximum unit selects 0 as the other operand for each element at the output of the minimum unit. The outputs of the maximum unit are fed to

the inputs of the Y register. From the second to the last clock steps, outputs of the Y register are fed back to the inputs of the maximum unit through the MUX3 multiplexer. 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. If the error flag is activated at the completion of the learning of a new rule, then all elements in the R rule memory equal 1 (full membership). This flag can be used to generate an interrupt request to the host machine. The system can then recover from this erroneous state by either downloading a "safe" model to the R memory or starting over the learning process with a modified model.

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 space-time diagram in Figure 3.3 illustrates the overlapped operations of the pipeline units. Assuming that the downloading of the fuzzy data from the host system to the accelerator and the reading of the fuzzy and/or crisp output data from there (subtask T_1) does not exceed $[u_{max}+4]+2$ (9) clock periods, the accelerator produces new fuzzy and/or crisp output data every $[u_{max}+4]+2$ clock periods once the pipeline is filled. Thus, at a clock rate of 30 MHz the fuzzy engine can perform over 3,000,000 fuzzy logical inferences per second with the current fuzzy data format.

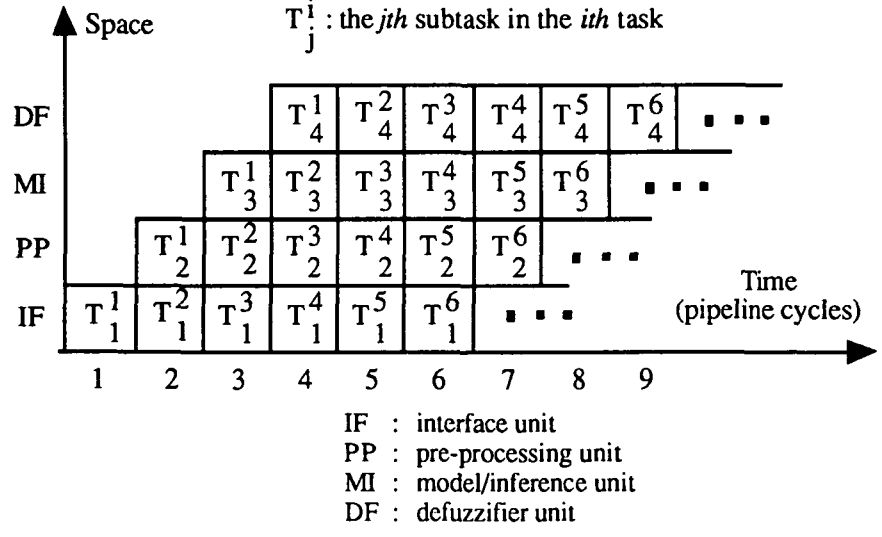

Figure 3.3. Overlapped operations of the pipeline units.

4. VLSI IMPLEMENTATION

One of the most difficult issues coming from the practical realization is associated with the VLSI implementation, therefore the information provided in this section are based on our estimates and previous experience with projects of a similar nature.

Due to our objective constraints, i.e. the MOSIS service is available for chip fabrication at this time, the full design version of the proposed controller will be designed, along with a scaled-down version which will pass the constraints and will finally be fabricated.

There are two different 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 starting with the full scale design. According to our preliminary assumption we come up with the descriptions of design signals which are summarized in Table 1.

Full scale HC version	Full scale SA version
64b scaled address/data bus	64b digital address/data bus @
~55 signal for data bus control *	32b XPROM interface
(Bus Parity), (Command)	4 XPROM control signals
(Status and CP), (Capability)	4 analog inputs %
(Synchronization)	3 signals for fuzzification control #
(Arbitration)	3 signal for inference control #
(Addresses)	2 signals for defuzzification control #
3 signals for fuzzification control #	4 signals for mode control
3 signals for inference control #	2 or 3 analog outputs
3 signals for defuzzification control #	1 CLK global clock
1 CLK global clock	1 STB strobe signal
1 RESET input	1 EN synchronization input
1 CS chip select	1 RESET input
~8 power supply inputs	~8 power supply inputs

* We are currently working on the data bus control so this number can be changed.
# These options could be programmable.
% This number is a subject of investigation and can be changed.
@ Can be used to substitute for a single analog input/output.

Table 1. Preliminary definitions of signals for full scale versions of HC and SA Fuzzy Logic Controller.

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. In our first approach this version will not be cascadable. The SA version of the chip will input the analog data and perform the fuzzification operation by itself. The stand-by mode will be common for both versions.

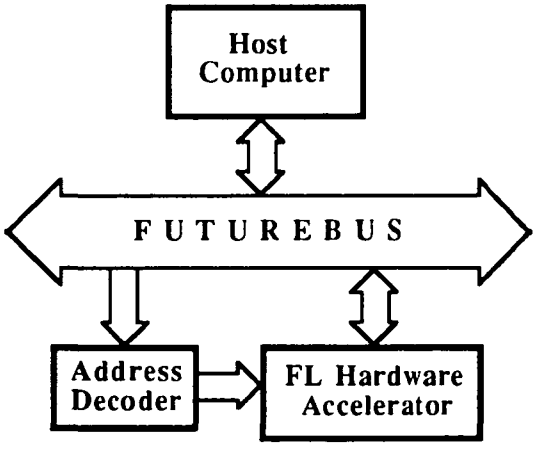

Fig. 4.1. Configuration for the Hardware Accelerator working under the host computer (HC version).

One can also see that the HC version will require a very detailed design of the interface to the bus system used by the host computer for data transmission while the SA version will need an A/D converter and a few D/A converter,which will be included in the chip design.

It is assumed that such a version will be communicating with the host computer through FUTUREBUS (Fig. 4.1).

Both versions has its own advantages and disadvantages basically due to communications issues, the number of pins and the design effort. It is important to point out that one can expect some instant differences in the performance of the four versions which will further be investigated in detail.

Let us now focus on the scaled down implementations of the SA version of the proposed fuzzy logic controller. There will be designed two basic modes for chip operation: normal and programmed. The normal mode will include cascaded (parallel or serial) and non cascaded operation. The block diagrams illustrating these modes are shown in Figs. 4.2 and 4.3.

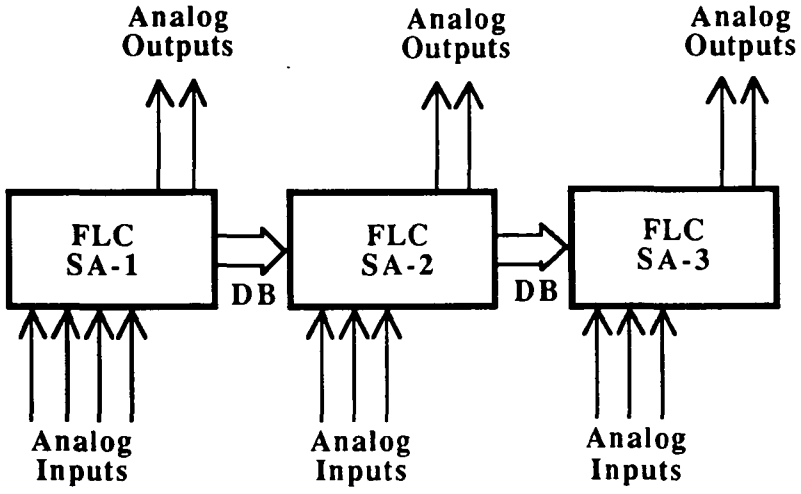

Figure. 4.2. Serial configuration for Fuzzy Logic Controller (SA version).

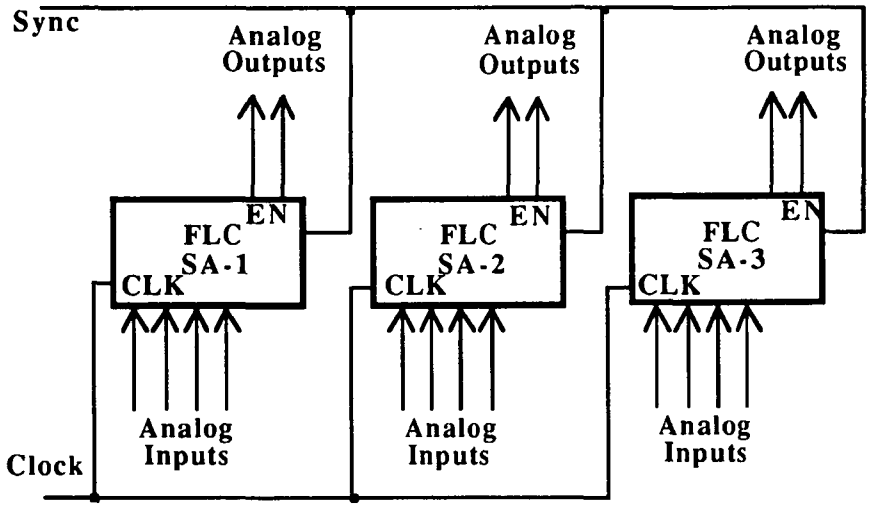

Figure. 4. 3. Parallel configuration for Fuzzy Logic Controller (SA version).

In the programming mode, we assume that it will be possible to preprogram the fuzzifier, defuzzifier or inference engine, or any combination of these, in order to preserve a flexible operation tuned to the actual user. In order to achieve programmability, an EPROM/EEPROM type of memory block will be built-into the chip, and will be controlled by the external source through a memory I/O port and control signals. Our decision to fabricate this version is based on both the number of pins and also the number of signals needed to implement this version (no data interface is needed). The scaled down implementation of our design matching the objective constraints is presented as follows.

• Chip size and package

The reasonable MOSIS package has 132 pins and can contain a chip occupying 7.8mm*9.2mm of silicon area (max). The choice of CMOS technology leads us to the variety of available processes starting from *lambda*=1μm to *lambda*=0.6μm. Keeping in mind the maximum signal frequency for chip operation, which was originally set between 25MHz and 30MHz, as well as the maximum chip size, the n-well, double-metal CMOS technology with lambda=0.6μm will be adequate to achieve the design goals.

• Chip area and number of transistors

The maximum chip area of 72.68mm^2 (132 pins package) can contain about 600000 transistors for highly regular structure[2] with the standard CMOS technology (*lambda*=0.6μm). According to our estimations we will be able to put at most as four parallel fuzzy data processing paths into the chip. The single data processing path including the programming options (memoryless) is estimated to have about 50,000 transistors..

• Clock strategy and clock distribution

We decided to use single external clock signal (CLK) to generate an on-chip, two-phase non overlapping internal clock signals (Φ1 and Φ2), the two phase-clock system having the advantage of making hazard problems within the pipeline paths more easily identifiable. These phases will be distributed over the whole chip using a second metallization layer. Because the longest possible metal line is about 10mm, we chose a tree-like structure for phase distributions driven by high gain clock drivers. These drivers will be designated to drive an appropriate capacitive load of the whole clock line tree. According to the results of our previous research, the single processing path will have the ability to operate at 8 clock cycles/pipeline step. Setting the external clock rate at around 30MHz will enable us to operate the processor at a high processing rate. A future detailed investigation will help us to determine the highest possible clock rate.

• Rule memory

The major problem with the limited capacity of the internal SRAM (for HC version) of EPROM/EEPROM (for SA version) memory for the storage of rules of inferences in previous works [6-8] does not exist in our approach due to the strategy of building the global rule for the whole linguistic model described in Chapter 2. In our case only 1/4 Kbyte SRAM or EPROM/EEPROM is needed to store the global rule. Such an approach creates a luxury of increasing the parallelism of the internal structure by a factor of four, which is discussed in the next section.
It also should be noted that the idea of CSFO FSM is intended to be implemented in the SA version. Furthermore, the required extension of the rule memory (every FSM state will have assigned rule memory) will be evaluated. It is however unlikely that overall number of transistors for a single path will reach 100,000 transistors.

• Pipeline architecture

The estimated number of transistors for a single fuzzy data processing path is around 50,000. This means that the chip under consideration has the capacity containing at least four separate fuzzy data processing paths plus rule memory, which gives total estimation of about 300,000 transistors (look-up table used for defuzzification is included). The estimated area occupied by transistors is about 45 mm^2. The rest of the chip area will be used to provide high speed communication between processing units and the built-in memory (EPROM/EEPROM). It is expected that four parallel data paths will be designed in the chip increasing the actual speed of operation twice. In the proposed design, 3 MFLIPS performance is expected assuming the clock rate will be up to 30MHz.

5. CONCLUSIONS

The paper describes the general model for fuzzy state machine (FSM) which is used to formulate the fuzzy controller for event-driven real-time systems. As a result the improved architecture for fuzzy logic controller has been defined.
The improvement with respect to already published architectures [5-9] comprises in a novel strategy for fuzzy model building, which enables fuzzy inferences to be performed in a single stage of a hardware accelerator. As it has been estimated the proposed architecture, appropriately pipelined, for the hardware accelerator will profit in reaching at least 3M fuzzy logical operations per second.
The presented approach can be utilized for fuzzy controller hardware accelerators intended to work in the real time environment.

---

[2] Excluding the area occupied by the chip frame.

REFERENCES

1. L. A. Zadeh, "A Fuzzy Algorithmic Approach to the Definition of Complex or Imprecise Concepts", *International Journal Man Machine Studies*, 8, pp. 249-291, 1976.
2. M. S. Stachowicz, "The Application of Fuzzy Modeling in Real-Time Expert Systems for Control", *Proc. 49th Ironmaking Conference*, Detroit, pp. 503-512, March 25-28, 1990.
3. M. S. Stachowicz, J. Grantner, L. L. Kinney, "Two-Valued Logic for Linguistic Data Acquisition", *NAFIPS Workshop '91*, University of Missouri-Columbia, Proc. pp. 168-172, May 14-17, 1991.
4. M. S. Stachowicz, M. E. Kochanska, "Graphic Interpretation of Fuzzy Sets and Fuzzy Relations", *Mathematics at the Service of Man*, (eds. A. Ballester, D. Cardins, E. Trillas), Springer-Verlag, West Berlin, pp. 620-629, 1982.
5. M. S. Stachowicz, J. Grantner, L. L. Kinney, "Pipeline Architecture Boosts Performance of Fuzzy Logic Controller", *IFSICC'92 International Fuzzy Systems and Intelligent Control Conference*, Louisville, Kentucky, Proc. pp. 190-198, March 15-18, 1992.
6. M. Togai, H. Watanabe, "Expert System on a Chip: An Engine for Real-Time Approximate Reasoning", *IEEE Expert*, pp. 55-62, Fall 1986.
7. H. Watanabe, W. D. Dettloff, K. E. Yount, "A VLSI Fuzzy Logic Controller with Reconfigurable, Cascadable Architecture", *IEEE Journal of Solid-State Circuits*, pp. 376-381, Vol. 25, No. 2, April 1990.
8. FC 110 Digital Fuzzy Processor DFP^{TM}. Togai InfraLogic, Inc. 10/1991.
9. M. J. Patyra, "VLSI Implementation of Fuzzy-Logic Circuits", International Fuzzy Systems Association World Congress, Brussels, Belgium, June, 1991.