7th International Conference on DEVELOPMENT AND APPLICATION SYSTEMS
S u c e a v a, R o m a n i a, M a y 27 – 29, 2 0 0 4

# A VHDL BASED APPROACH TO MODEL FUZZY LOGIC SYSTEMS

**Liviu ȚIGĂERU**[1]
**Ovidiu URSARU**[2]
*„Gh. Asachi” Technical University of Iasi,*
*Faculty of Electronics and Telecommunications*
*Bd. Carol I nr.11, Iasi, 6600 Romania*
*1) ltigaeru@etc.tuiasi.ro, 2) oursaru@etc.tuiasi.ro*

***Abstract.*** *Our aim was to find a solution to reduce the gap between the specific simulation tools for fuzzy logic systems and hardware design languages-based tools for electronic systems design. This paper presents a VHDL based approach to model fuzzy logic systems. This solution allows designing and simulation of the fuzzy logic systems in a standard hardware design environment, enabling the integration of the fuzzy models in complex hardware systems.*
***Keywords:*** *nonlinear systems models, fuzzy logic systems, VHDL*

## Introduction

The fuzzy logic is a computational paradigm that provides a mathematical tool to deal with the uncertainty and the imprecision. In the last decades, a lot of fuzzy logic system (FLS) based applications were reported in the literature. There are several areas where the fuzzy logic was successfully used: the automatic control, signal processing, data classification or decision making systems, computer vision, and so on. In this work, our goal was to find a hardware design language (HDL) based solution to model various configurations of FLSs. The HDLs have become an important factor in designing the VLSI circuits. These languages have been used to describe the circuits from the geometrical level up through the architectural level. The advantages of the HDL based FLS models approach derive in that:

- it allows the integration of the FLS model in a complex structure. As a result it can be described and synthesized a VLSI circuit with fuzzy computational features.
- the integration into a HDL based design environment that allows the verifications of the simulation results of a FLS configuration from a high (behavioral) description level to a structural synthesizable level (RTL or gate level).
- the integration into a unique HDL based design environment that allows the description of all components of a particular FLS based application. This approach lead to accurate simulation results.

One of the most popular hardware design language is VHDL (VHSIC Hardware Description Language). In the last few years it was investigated the possibility of using VHDL as a support for the description of the FLSs [1],[2]. In this work we propose a new VHDL package to model the FLSs.

## Description of the FLS

A FLS maps crisp inputs into crisp outputs. It contains four components: fuzzifier, fuzzy rulebase, the inference engine and the defuzzifier. The structure of the FLS, as was described in [3], is depicted in the Figure 1.

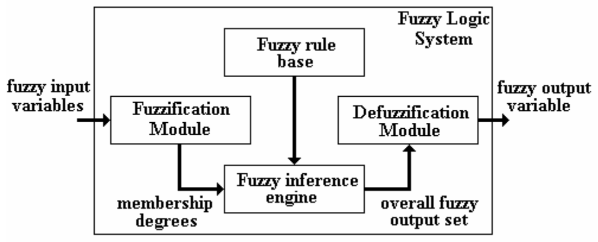

Figure 1 - The general structure of a FLS

The fuzzifier maps crisp numbers into fuzzy sets that are subsequently used as inputs to the inference engine. The fuzzy rulebase is a collection of rules that may be provided by experts or can be derived from numerical data. These rules are expressed as IF-THEN statements:

**IF** $X_1$ is $A_1^K$ **AND**…$X_N$ is $A_N^K$ **THEN** $Y$ is $B^K$

where $X_1,\ldots X_N$ and $Y$ are the inputs and the output of the system, $A_1^K\ldots A_N^K$ and $B^K$ are the linguistic labels (fuzzy sets). The $X_i$ is $A_i^K$ statement defines a fuzzy proposition. The fuzzy propositions are combined by means of the logic connectives (`AND`, `OR`, `NOT`) into the premise of the fuzzy rule. The inference engine maps the input fuzzy sets into the output fuzzy set and generates an overall output fuzzy set. Each fuzzy rule provides a partial contribution - a local output fuzzy set - to the overall output fuzzy set, that is an aggregation of the partial contributions. The defuzzifier maps the overall output fuzzy set produced by the inference engine into a crisp number.

## The VHDL package

In this section it is proposed a VHDL package to model various FLS configurations. First, it were defined a set of basic data types that store the fuzzy information. Figure 2 presents the data structures.

```
type FS is record
    name  : string(1 to 2);
    shape : string (1 to 4);
    a     : real;
    b     : real;
    c     : real;
    d     : real;
end record;
type FSs is array (1 to numberFS) of FS;
type INPUT is record
    variable_name : string(1 to 2);
    N             : natural;
    fuzzy_sets    : FSs;
    min           : real;
    max           : real;
end record;
type OUTPUT is record
    variable_name : string (1 to 2);

    N             : natural;
    fuzzy_sets    : FSs;
    min           : real;
    max           : real;
    samples       : natural;
end record;
type  FZOUT is array (1 to numberSAMPLES) of real;
type RULEBASE is  array (natural range<>) of FZOUT;
```

Figure 2 - The data structures of the package

A fuzzy set is described by the record data type called `FS`, formed by six fields: `name`, `shape`, `a`, `b`, `c`, and `d`. The first field stores the linguistic label that identifies the fuzzy set. The second field defines the shape of the membership function of the fuzzy set. It can be defined five membership functions: trapezoidal, triangular, S-shaped, Z-shaped and singleton. The last four fields represent the parameters of the membership function. The fuzzy sets defined for the same fuzzy variable can be grouped into a vector of `FS` data type denoted `FSs`. A fuzzy input variable is described by means of the record data type `INPUT`, formed by five fields: `name`, `N`, `fuzz_sets`, `min` and `max`. The first field identifies the input variable, the second represents the number of the defined fuzzy sets, the third assigns the defined fuzzy sets to the concerned input variable and the last two fields define the universe of discourse of the input variable. The same manner is used to describe the output fuzzy variable of the system. This is stored in the record data type `OTUPUT` that is formed by six fields: `name`, `N`, `fuzzy_sets`, `min`, `max` and `sample`. The last field of the record represents the number of the samples used to represent in a discret form the output fuzzy sets of the system. The other fields preserve the same meaning as the fields that form the record `INPUT`. The `FZOUT` data type is a vector that keeps the samples of the partial contribution generated by the inference engine. The `RULEBASE` data type is a vector of `FZOUT`. It is used as a data support in the aggregation stage of the local output fuzzy sets.

The fuzzy algorithm is described by a set of functions that handle the introduced data types. It were developed several functions that can be divided into three distinct groups. First group contains the initialization functions. The function `init_FS` initializes the fuzzy sets of the system. The `init_INP` and `init_OUT` functions initialize the input and output fuzzy variables. Next group of functions are used to describe the fuzzy rules and inference mechanism of the system. The `FUZZ` function performs the fuzzification of the data input. It computes the membership degrees of the input values to the input fuzzy sets. The logic connectives, used in the fuzzy rules, are defined by means of three functions: `fzAND`, `fzOR` and `fzNOT`. Table 1 presents the mathematical operators provided by the VHDL package to implement the and/or logic connectives.

Table 1. The mathematical operators used to implement the logic connectives

| | AND | OR |
|---|---|---|
| Zadeh | $\min(A, B)$ | $\max(A, B)$ |
| Probabilistic | $A \cdot B$ | $A + B - A \cdot B$ |
| Lukasiewicz | $\max(A + B - 1, 0)$ | $\min(A + B, 1)$ |

Subsequently, it was introduced the function `IMPL` that performs the fuzzy implication. We have implemented several implication functions. These are presented in Table 2.

Table 2. The fuzzy implication functions

| implication function | definition |
|---|---|
| Mamdami | $\min(\beta, C)$ |
| Larsen | $\beta \cdot C$ |
| Kleene-Dienes | $\max(1-\beta, C)$ |
| Goguen | $\begin{array}{ll} 1 & \beta = 0 \\ \min\left(\frac{C}{\beta}, 1\right) & \beta \neq 0 \end{array}$ |

where β represents the firing degree and C the consequence of the fuzzy rule. The aggregation of the partial contributions is performed by the `AGG` function. Table 3 summarizes the mathematical operators used to implement this function.

Table 3. The mathematical operators used to implement the aggregation

| mathematical operator | definition |
|---|---|
| maximum | $\max(PC^1, ..., PC^K)$ |
| algebraic sum | $PC^1 + ... + PC^K$ |
| bounded sum | $\min(PC^1, ....PC^K, 1)$ |

$PC^i$ represents the partial contribution of the i fuzzy rule. Finally, it was introduced the `DEFF` function that performs the defuzzification. There are provided the most popular defuzzification methods: Center of Gravity (COG), Mean of Maxima (MOM), First of Maxima (FOM) and Last of Maxima (LOM).

**A VHDL based Fuzzy Logic Controller model**

To validate the proposed VHDL package, it was described a Fuzzy Logic Controller (FLC) that is used to command a buck converter. The goal of the application is to keep the output voltage at a constant value ($V_{OUT}$=5V) in the presence of various working perturbations (change of input voltage, change of load). Figure 3 presents the structure and the circuit parameters of the buck converter.

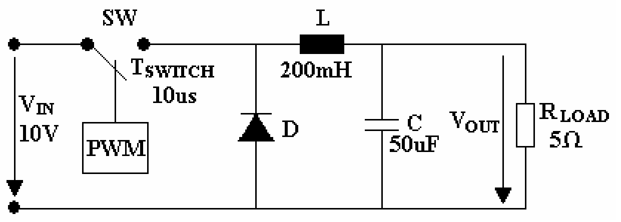

Figure 3 - The buck converter

To perform this task, $V_{OUT}$ is sampled with a sampling period $T_S$=10us and compared to a reference voltage $V_{REF}$=5V such that an error signal is generated:

$$E(K) = V_{OUT}(K) - V_{REF} \quad (1)$$

where K is the sampling moment. The error signal and the change of error signal

$$CE(K) = E(K) - E(K-1) \quad (2)$$

are supplied to the FLC inputs, that outputs the duty cycle of the PWM signal, used to command the buck converter. Figure 4 presents the structure of the control application. With except of the FLC, the other blocks used in the application are modeled in VHDL-AMS, the analog extension of the VHDL language. The proposed VHDL package allows to model various configurations for the FLC.

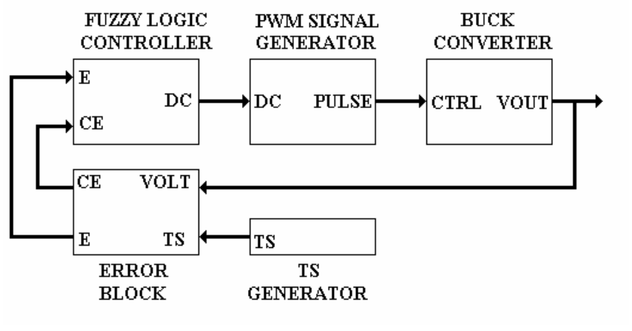

Figure 4-The structure of the control application

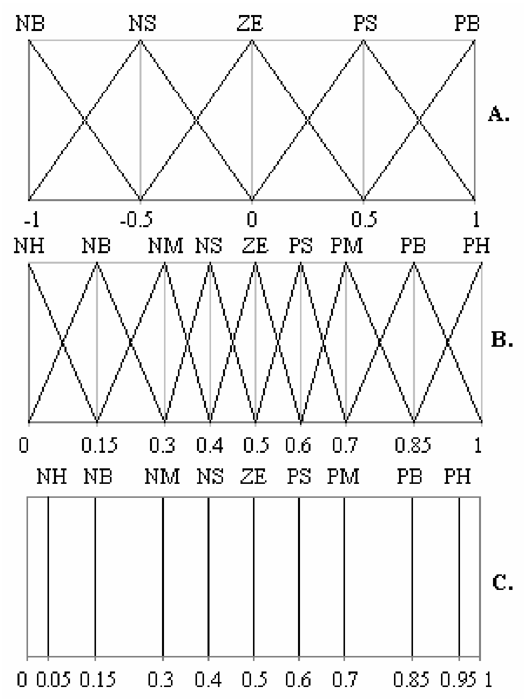

Figure 5-The fuzzy sets of the controller: a-the input fuzzy sets; b-the output fuzzy sets; c-the output fuzzy sets (Takagi Sugeno FLC case)

Table 4. The fuzzy rulebase

| CE / E | **NB** | **NM** | **ZE** | **PM** | **PB** |
|---|---|---|---|---|---|
| **NB** | PH | PB | PM | PS | ZE |
| **NS** | PB | PM | PS | ZE | NS |
| **ZE** | PM | PS | ZE | NS | NM |
| **PS** | PS | ZE | NS | NM | NB |
| **PB** | ZE | NS | NM | NB | NH |

In this work we have considered the most used FLC configurations in control applications: Mamdami (min-max), Larsen (product-bounded sum) and zero order Takagi-Sugeno. For the Mamdami FLC we have chosen two different cases: one case with COG and another case with MOM defuzzification method. For Larsen FLC we have considered only COG defuzzification method. For all configurations, the input variables are defined over a [-1,1] universe of discourse and the output variable is defined over a [0,1] universe of discourse. Five and nine respectively fuzzy sets are defined for both input and output variable respectively. The fuzzy sets of the controller are shown in Figure 5. The rulebase of the controller is presented in Table 4. The VHDL based model of the Mamdami COG FLC is presented in Figure 6. The fuzzy algorithm of the controller is described in several stages:

- the initialization of the controller's fuzzy sets, inputs and output variable.
- the fuzzy rulebase description
- the aggregation of the local inferred output fuzzy sets
- the defuzzification of the overall output fuzzy set and the assignation of the result to the output

The model for the others FLC configurations can be described similarly.

## Simulation results

The VHDL based FLC models were verified by time domain simulations. The control application was described and simulated in a unique design environment that allows to run both VHDL and VHDL-AMS models. For each FLC configuration it was considered the same control algorithm. The performance of the FLC was verified by simulating the line regulation and load regulation of the buck converter. For the first situation, the input voltage (line voltage) has suddenly changed from 10V to 20V and back to 10V. For the load regulation, the load has suddenly changed from 5Ω to 10Ω and back to 5Ω. Figure 7(a)-(d) shows the line regulation and Figure 8(a)-(d) shows the load regulation. Figure 9(a)-(d) shows the control surface generated by each considered FLC configuration. These results confirm the ability of the proposed models for the FLCs in controlling the buck converter and validate the VHDL package developed in this work.

## Conclusion

Our aim was to find a solution to reduce the gap between the specific simulation tools for fuzzy systems and HDL-based tools for hardware design. Thus, it was developed a VHDL package to model various configurations of FLS that were validated by simulation. As a final consequence, these models can be integrated in complex structures to describe very large systems that exhibit fuzzy computational features.

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;
use work.fuzzy.all;
entity controller is
       port(
      e_val  : in  data :=0.0;
      ce_val : in  data :=0.0;
      dc_val : out real :=0.0);
end controller;
architecture model of controller is
 constant opAND :string(1 to 4):="ZADH";
 constant f_imp :string(1 to 4):="MAMD";
 constant opAGG :string(1 to 3):="MAX";
 constant method:string(1 to 3):="COG";
begin
algorithm: process(e_val,ce_val)
  variable LE,LCE,LDC : FSs;
  variable ER,CE: INPUT;
  variable DC: OUTPUT;
  variable R: RULEBASE(1 to 25);
  variable Total_contribution: FZOUT;
  variable Result:real:=0.0;
begin
LE:= (
init_FS("NB","ZRMP",-1.0,-0.5,0.0,0.0),
init_FS("NS","TRNG",-1.0,-0.5,0.0,0.0),
init_FS("ZE","TRNG",-0.5,0.0,0.5,0.0),
init_FS("PS","TRNG",0.0,0.5,1.0,0.0),
init_FS("PB","SRMP",0.5,1.0,0.0,0.0),
others=>Undefined);
LCE:=(
 init_FS("NB","ZRMP",-1.0,-0.5,0.0,0.0),
LDC:=(
init_FS("NH","ZRMP",0.00,0.15,0.00,0.0),
init_FS("NB","TRNG",0.00,0.15,0.30,0.0),
init_FS("NM","TRNG",0.15,0.30,0.40,0.0),
init_FS("NS","TRNG",0.30,0.40,0.50,0.0),
init_FS("ZE","TRNG",0.40,0.50,0.60,0.0),
init_FS("PS","TRNG",0.50,0.60,0.70,0.0),
init_FS("PM","TRNG",0.60,0.70,0.85,0.0),
init_FS("PB","TRNG",0.70,0.85,1.00,0.0),
init_FS("PH","SRMP",0.85,1.00,0.10,0.0),
others=>Undefined);

ER:=init_INP("ER",5,LE,-1.0,1.0);
CE:=init_INP("CE",5,LCE,-1.0,1.0);
DC:=init_OUT("DC",9,LDC,0.0,1.0,10);

R(1):=IMPL(
fzAND(FUZZ("PB",ER,e_val),
FUZZ("NB",CE,ce_val),opAND),
"ZE",DC,f_imp);

R(2):=IMPL(
fzAND(FUZZ("PB",ER,e_val),
FUZZ("NS",CE,ce_val),opAND),
"NS",DC, f_imp);
.......................................................................................................
R(25):=IMPL(
fzAND(FUZZ("NB",ER,e_val),
FUZZ("PB",CE,ce_val),opAND),
"ZE",DC, f_imp);

Total_contribution:=AGG(R,DC,opAGG);
init_FS("NS","TRNG",-1.0,-0.5,0.0,0.0),
init_FS("ZE","TRNG",-0.5,0.0,0.5,0.0),
init_FS("PS","TRNG",0.0,0.5,1.0,0.0),
init_FS("PB","SRMP",0.5,1.0,0.0,0.0),
others=>Undefined);
```

```
Result:=
DEFUZZ(Total_contribution,DC,deffm);
dc_val<=Result;
end process algorithm;
end model;
```

Figure 6- The VHDL based model of a Mamdami Fuzzy Logic Controller

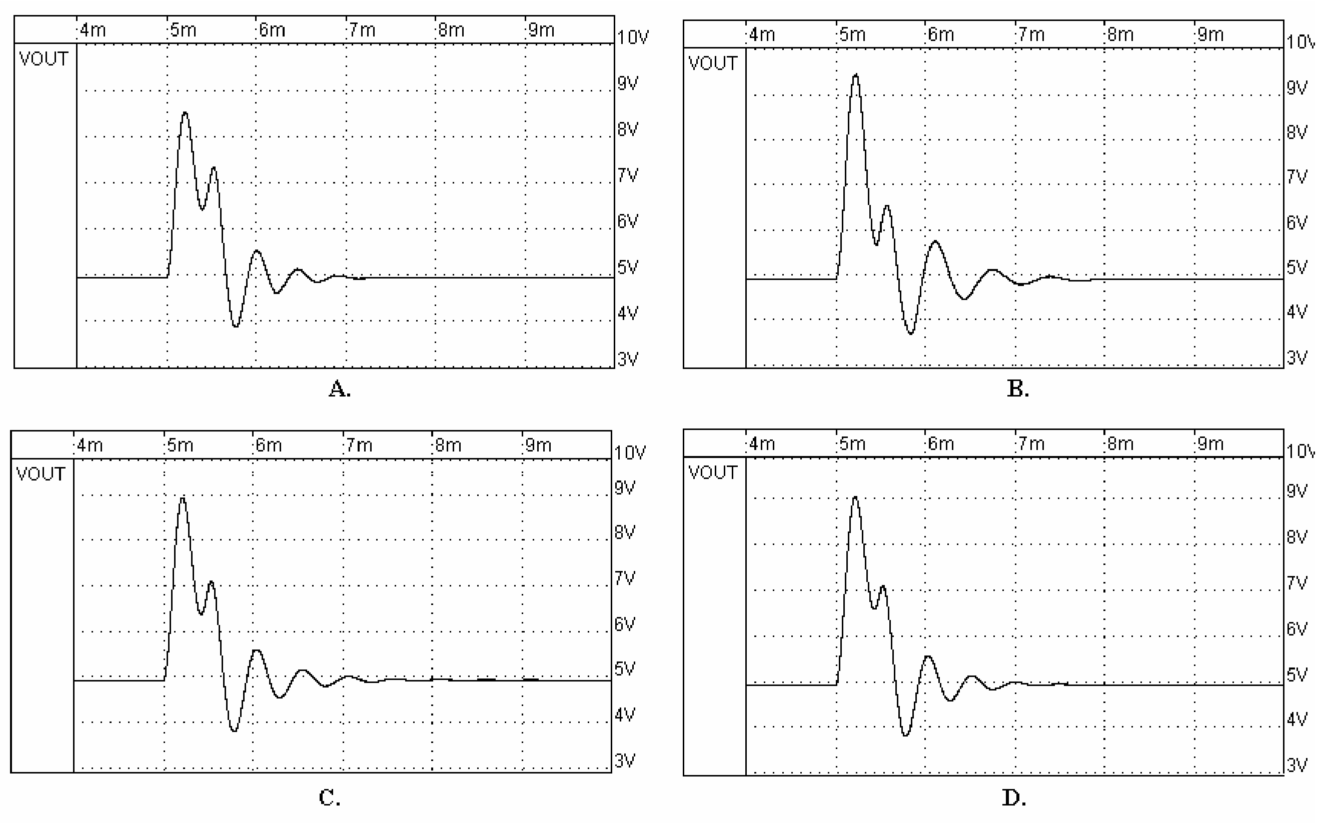

Figure 7 - Simulation results-line regulation: a-Mamdami FLC with COG defuzzification; b-Mamdami FLC with MOM defuzzification; c-Larsen FLC; d- zero order Takagi Sugeno FLC

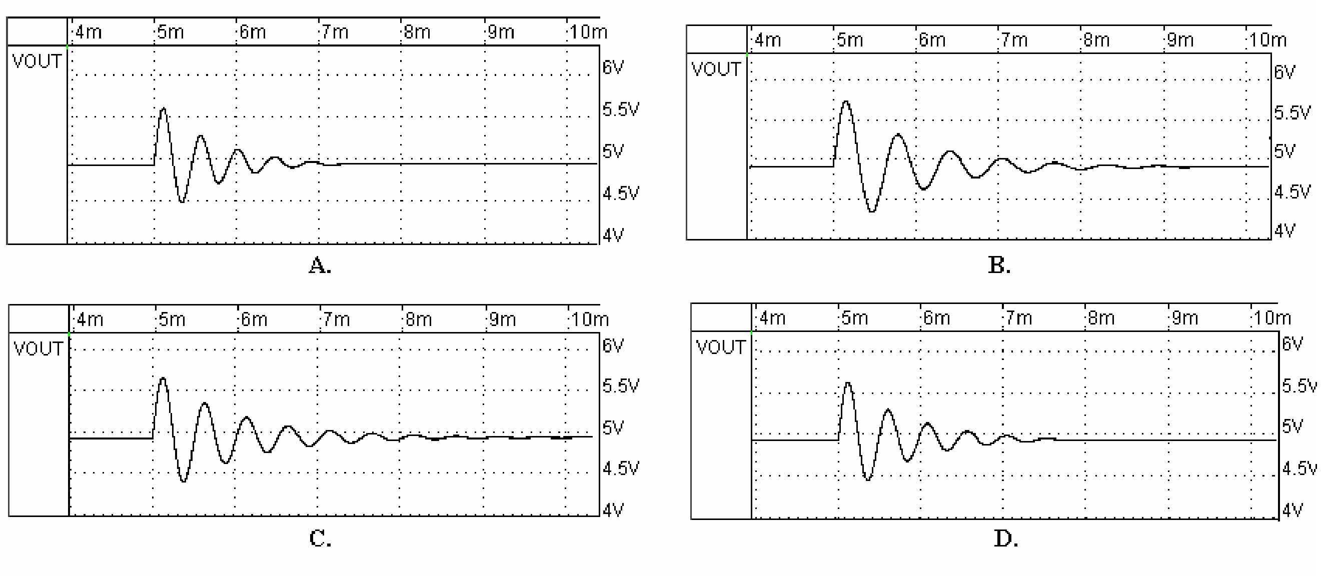

Figure 8 - Simulation results-load regulation: a-Mamdami FLC with COG defuzzification; b- Mamdami FLC with MOM defuzzification; c-Larsen FLC; d- zero order Takagi Sugeno FLC

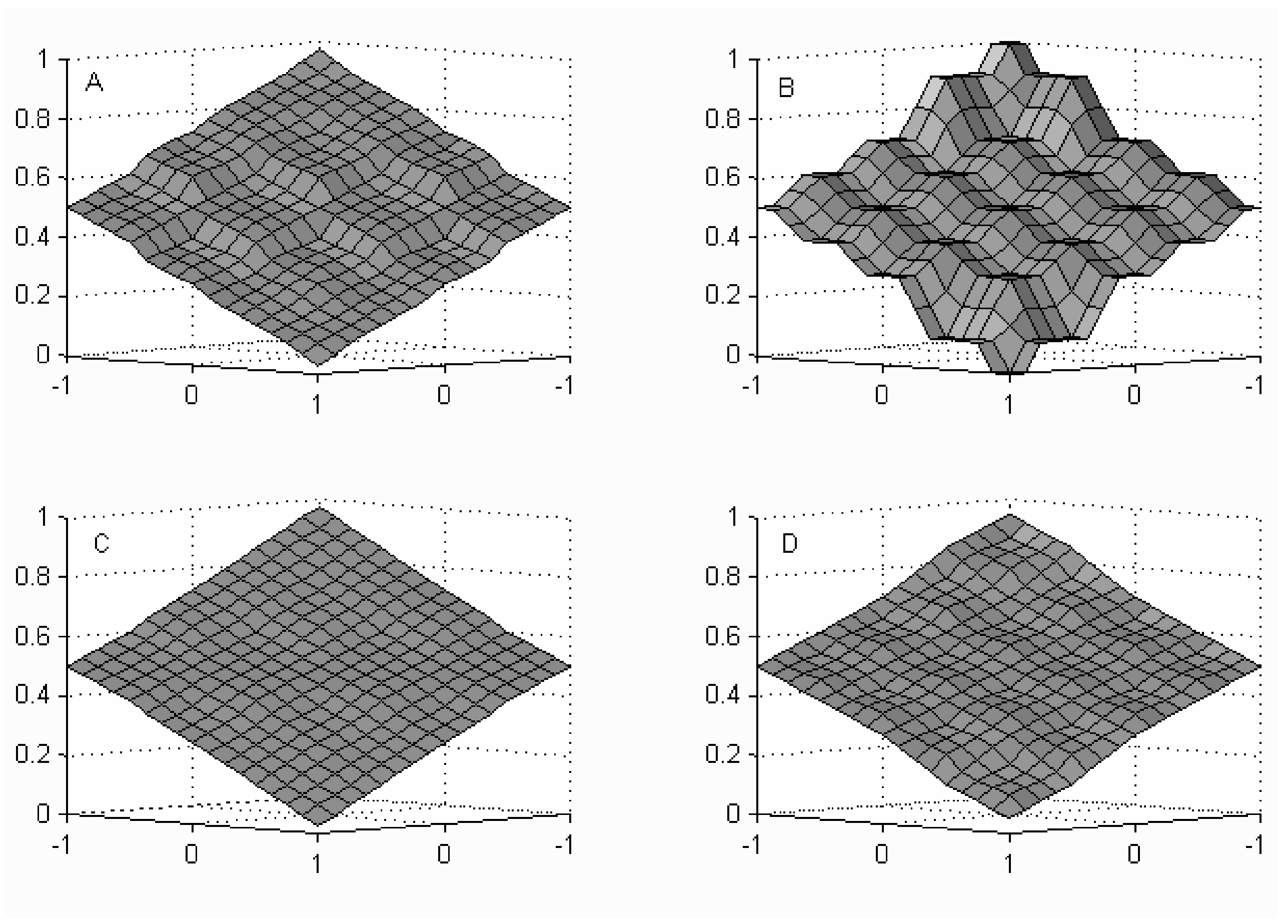

Figure 9 - The control surface generated by each FLC configuration: a-Mamdami FLC with COG defuzzification; b- Mamdami FLC with MOM defuzzification; c-Larsen FLC; d- zero order Takagi Sugeno FLC

## References

[1] Hollstein T., Halgamuge S.K., Glesner M. (1996) *Computer Aided Design of Fuzzy Systens Based on Generic VHDL Specifications*,IEEE Trans.on Fuzzy Systems, no.4 pp. 403-417.
[2] D. Galán, C. J. Jiménez, A. Barriga, S. Sánchez Solano (1995) *VHDL Package for Description of Fuzzy Logic Controllers*, European Design Automation Conference (EURO-VHDL'95), pp. 528-533, Brighton - Great Britain.
[3] Driankov D., Hellendorn H., Reinfrank M., (1993) *An introduction to fuzzy control*, Springer Verlag.
[4] N.Mohan, T.Undeland and W.Robbins: (1995) *Power Electronics - Converters, Applications and Design*, John Wiley&Sons inc..
[5] W.C. So, C.K. Tse, Y.S. Lee, (1996) *Development of a Fuzzy Logic Controller for DC/DC Converters: Design, Computer, Simulation and Experimental Evaluation*, IEEE Trans. on Power Electronics, vol.11 no.1 pp. 24-31.
[6] P.J. Ashenden, G.D. Peterson, D.Teegarden, (2002) *The System Designer's GuideVHDL-AMS*, Morgan Kaufman Publishers.
[7] R. E. Harr and A. G. Stanculescu, ed. (1991) *Applications of VHDL to Circuit Design*. Kluwer Academic Publishers, Boston.