Implementation of Fuzzy Controller in Verilog

Sandeep Sharma[1] Prof M C Patel[2]
[1]Student [2]Associate Professor
[1,2]Department of Instrument & Control Engineering
[1,2]LD College of Engineering, IC Department, GTU, Ahmedabad

Abstract— This paper pronounces the implementation for a basic fuzzy logic controller in Verilog Higher Descriptive Language (Verilog). This implementation is carried out through the use of the Verilog code. Use of the hardware description Verilog language in the application is suitable for being implemented into an Application Specific Integrated Circuit (ASIC) and Field Programmable Gate Array (FPGA). The advantages of using the Verilog language approach include prompt prototyping, and allowing usage of powerful synthesis tools such as Xilinx ISE, Synosys, Mentor Graphic, or Cadence which can be targeted easily and efficiently.

Key words: Verilog, Fuzzy Logic Controller, Fuzzzification, Rule Evaluation, Defuzzification, Application Specific Integrated Circuit (ASIC)

I. INTRODUCTION

The implementation of a fuzzy controller in Verilog language was motivated by the need for an inexpensive hardware implementation of a generic fuzzy controller for use in industrial and commercial applications. To demonstrate this implementation, an external device's information,(for instance say sesor), is converted into an output control signal to drive a device(s) such as motors, actuators etc., by the process of fuzzification, rule evaluation and defuzzification [1,2,3]. These processes are all based on a set of membership functions.

II. FUZZIFICATION

Among the different types of membership functions, the most commonly used in practice are the triangular and trapezoidal membership functions where the earlier function being a special case of the later functions. For this application we use a trapezodial function.

Each trapezoidal membership function is defined by two points and two slope values. The entire membership function can be divided into three segments: 0, 1 and 2 as shown in Figure 1. The Y axis displays the degree of membership (μ) having values between 0 and 1. The X axis displays the universe of discourse and is divided into three segments. The degree of membership is dependent on the location of the input value with reference to these three segments. Figure 1 shows the formation of trapezoidal input membership functions in the fuzzification process [8, 9, 10]".

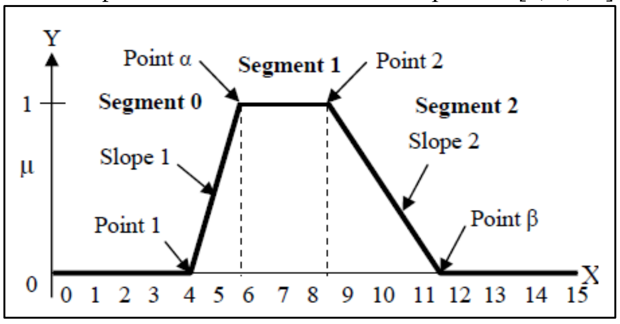


Fig. 1: Trapezoid Function Graph

$$trapezoid\,(x;a;b;c;d) = \begin{cases} 0 & x \le a \\ \dfrac{x-a}{b-a} & a \le x \le b \\ 1 & b \le x \le c \\ \dfrac{d-x}{d-c} & c \le x \le d \\ 0 & d \le x \end{cases}$$

The calculation of the degree of membership ((μ) can be categorized into three different segments:

- in part 0: μ = 0,
- in part 1: slope is upward from left to right, therefore: μ = (Input value – point 1) * slope 1, where μ is limited to a maximum value of 1,
- in part 2: slope is 1 downward from left to right, therefore: μ = 1 - (Input value – μ point 2) * slope 2, where μ is limited to a minimum value of 0.

Take an illustrative, let's take a value "10" and we have to calculate the membership function regarding to this value. If we use 8 bit resolution, then μ = 1 equals to $FF or 255 in decimal (the "$" is hexadecimal value). The point 1 and point 2 values are $04 and $09, respectively, and the two slopes are:

- Slope 1 = 1 / (6 – 4) = $FF / 2 = 255 / 2 = 127 = $7F … (1)

And

- Slope 2 = 1 / (12 – 9) = $FF / 3 = 255 / 3 = 85 = $55…...(2)

As the value 10 ($0A) is greater than Point 2 and lies in part 2, Hence

Since the input value of 10 ($0A) is greater than Point 2 and lies in segment 2, hence

μ = $FF – (Input value – point 2) * slope 2 = $FF – ($0A - $09)*$55 = $AA (3)

In Verilog language, each membership function is defined by four 8-bit values,

```
if(X<=a)
        begin
                Trapezoidal=8'b00000000;
        end
        else if(X<b)
        begin
                slope1=8'b11111111/(b-a);
                Trapezoidal=(X-a)*slope1;
        end
        else if(X<=c)
        begin
                Trapezoidal=8'b11111111;
        end
        else if(X<=d)
        begin
          slope2=8'b11111111/(d-c);
         Trapezoidal=8'b11111111-((X-c)*slope2);
        End
else
```

```
        begin
                Trapezoidal=8'b00000000;
        end
end
end function
```

Where membership function can be described as:

```
        Assign Fuzzy NL= Trapezoidal (CrispData, 8'b0,
8'b0, 8'b00101010, 8'b01010101);    //a=0,b=0,c=2A,d=55
```

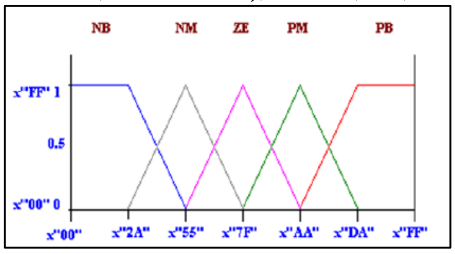


Fig. 2: Fuzzy Membership Function for error and change in error

Point 1 and point 2 for each membership function can be calculated by above figure 2

III. Rule Evaluation

A rule usually takes the form of *IF-THEN* statement as follow:

IF x is A AND y is B THEN z is C (4)

"AND" is a fuzzy operator which is the minimum operation between the two antecedents. In Verilog, the following "minimum" function is used to obtain the result of each rule evaluation between two variables:

```
function [7:0] Minimum;
input [7:0]a1,b1;
begin
        if(a1>b1)
        begin
                Minimum=b1;
        end
        else
        begin
                Mininimum=a1;
        end
end
end function
```

There is also an implied "OR" operation between successive rules when more than one rule is involved with the same output. The linguistic rule, IF (x_1 is A_1 AND y_1 is B_1) OR (x_2 is A_2 AND y_2 is B_2) THEN z is C, can be implemented by taking the maximum value of all the results with the same consequent block. In Verilog code, a function of taking the maximum of two input variables to determine the final result of all rules with the same output can be written as:

```
function [7:0] Maximum;
input [7:0]a2,b2;
begin
        if(a2>b2)
        begin
                Maximum=a2;
        end
        else
        begin
                Maximum=b2;
        end
```

E CE	NB	NM	ZE	PM	PB
PB	NM	NS	NB	PB	PB
PM	NM	NM	NB	PB	PB
ZE	NB	NB	ZE	PB	PB
NM	NB	NB	NB	PM	PM
NB	NB	NB	NB	PS	PM

Table 1: Fuzzy rules

By combining the "minimum" and "maximum' functions, the truth value for each rule can be obtained as

C <= max(min(X_1, Y_1), min(X_2, Y_2));

For suppose output rule NS to be fired we can write it as

```
        assign OutNS =Max (Max (Max (Min (ErPL,
ErChNL), Min(ErPL,ErChNS)), Min(ErPS, ErChNL)), Min
(ErPS, ErChNS));
```

IV. Defuzzification

During the defuzzification process, each fuzzy output is multiplied by its corresponding singleton position. The sum of this product is divided by the sum of all fuzzy output to obtain the result of the final output. In Verilog, this is implemented as:

```
        Assign product= (fuzzyPL*8'b0) + (fuzzyPS*8'b0)
+ (fuzzyZ*8'b0) + (fuzzyNS*8'b0) + (fuzzyNL*8'b0);
        Assign sum = fuzzyPL +fuzzyPS+ fuzzyZ
+fuzzyNS + fuzzyNL;
Assign CrispData=product/sum;
```

V. Division Algorithm

Since direct division is not possible in FPGA, hence following Verilog code is used synthesis the division algorithm

```
input [7:0]a,b;
reg [7:0] a1,b1,product;
integer i;
begin
        a1=a;
        b1=b;
        product=0;
        for(i=0;i<8;i=i+1)
        begin
                product={product[6:0],a1[7]};
                a1[7:1]=a[6:0];
                product=product-b1;
                if(product[7]==1)
                begin
                        a1[0]=0;
                        product=product+b1;
                end
                else
                begin
                        a1[0]=1;
                end
        end
        Divide=a1;
```

VI. Simulation and Results

A. *Case of Study 1*

The inputs are: error = -45 and cerror= 5.

1) *System 1: Simulink with Xilinx calculates the output*

$$out = \frac{(128 * 43) + (140 * 128) + (242 * 128)}{(43 + 128 + 128)} = 181$$

The value 181 decimal (0B5 hex) is **33.57** in the original universe of discourse [-80,80].

2) *System 2: Matlab/Simulink.*

$$out = \frac{(0 * 0.1667) + (8 * 0.5) + (72 * 0.5)}{(0.1667 + 0.5 + 0.5)} = 34.2847$$

Now, it is clear that 33.57-34.29=-0.72, hence the percentage difference between systems is 2.14%

B. *Case of Study 2*

Error = -28 and cerror= -10, using

1) *System 1: Simulink with Xilinx calculates the output*

$$out = \frac{(128 * 17) + (140 * 170)}{(17 + 170)} = 138$$

The value 138 decimal (8A hex) is 6.58 in the original universe of discourse [-80,80].

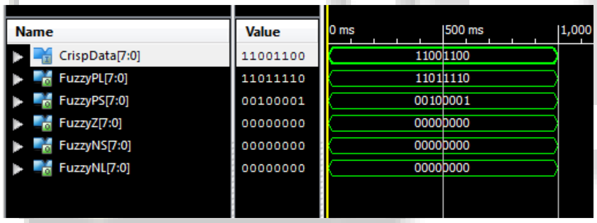
2) *System 2: Matlab/Simulink.*

$$out = \frac{(0 * 0.006667) + (8 * 0.6667)}{(0.06667 + 0.6667)} = 7.273$$

In the same way as in the previous case, we have 6.58-7.23= 0.65, hence the percentage difference between systems is 10.48%.

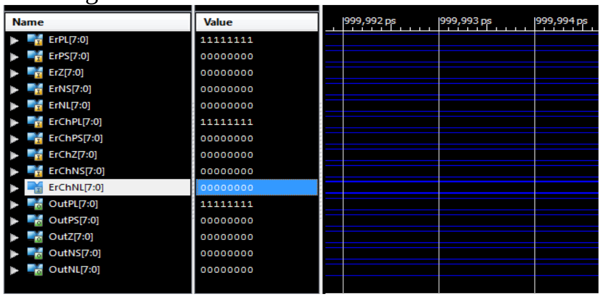
Three module in program is created fuzzification, rule base and defuzzification, each module is tested separately and at last combined to form a full program.

Simulation diagram for fuzzification is given below.



```
Name              Value       0 ms        500 ms        1,000
CrispData[7:0]    11001100    11001100
FuzzyPL[7:0]      11011110    11011110
FuzzyPS[7:0]      00100001    00100001
FuzzyZ[7:0]       00000000    00000000
FuzzyNS[7:0]      00000000    00000000
FuzzyNL[7:0]      00000000    00000000
```

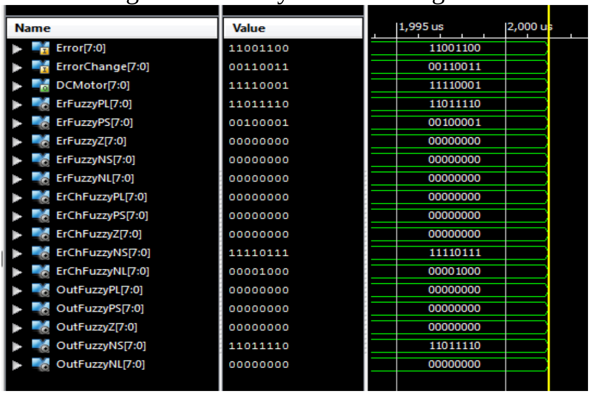
Fig. 3: Simulation window for Fuzzification.



```
Name              Value       999,992 ps   999,993 ps   999,994 ps
ErPL[7:0]         11111111
ErPS[7:0]         00000000
ErZ[7:0]          00000000
ErNS[7:0]         00000000
ErNL[7:0]         00000000
ErChPL[7:0]       11111111
ErChPS[7:0]       00000000
ErChZ[7:0]        00000000
ErChNS[7:0]       00000000
ErChNL[7:0]       00000000
OutPL[7:0]        11111111
OutPS[7:0]        00000000
OutZ[7:0]         00000000
OutNS[7:0]        00000000
OutNL[7:0]        00000000
```

Fig. 4: Simulation window for Rule base.

Simulation diagram for Fuzzy controller is given below.



```
Name                Value       1,995 us    2,000 us
Error[7:0]          11001100    11001100
ErrorChange[7:0]    00110011    00110011
DCMotor[7:0]        11110001    11110001
ErFuzzyPL[7:0]      11011110    11011110
ErFuzzyPS[7:0]      00100001    00100001
ErFuzzyZ[7:0]       00000000    00000000
ErFuzzyNS[7:0]      00000000    00000000
ErFuzzyNL[7:0]      00000000    00000000
ErChFuzzyPL[7:0]    00000000    00000000
ErChFuzzyPS[7:0]    00000000    00000000
ErChFuzzyZ[7:0]     00000000    00000000
ErChFuzzyNS[7:0]    11110111    11110111
ErChFuzzyNL[7:0]    00001000    00001000
OutFuzzyPL[7:0]     00000000    00000000
OutFuzzyPS[7:0]     00000000    00000000
OutFuzzyZ[7:0]      00000000    00000000
OutFuzzyNS[7:0]     11011110    11011110
OutFuzzyNL[7:0]     00000000    00000000
```

Fig. 5: Simulation window for Fuzzy controller.

VII. Conclusion

The design of a typical fuzzy logic controller using VERILOG is presented in this paper. Once the basic design of the fuzzy logic control system has been defined, the each component of the fuzzy inference system in VERILOG according to the design specifications. The above algorithm is synthesis in FPGA xc3s1000-5fg320. Here no of slices used is 616 out of 7680 giving out 8% area is 616 out of 7680 giving out 8% area utilization. Xilinx ISE 13.2 is used to synthesis the proposed algorithm.

References

[1] Madni Asad M, Wan L. A, Hansen, Robert K, Vuong Jim B, “Adaptive Fuzzy Logic Based Control System For Rifle Stabilization”, Proceedings, 1998 World Automation Congress (WAC ’98), Anchorage,Alaska, May 10-14, 1998, pp.103-112.

[2] Madni Asad M, Wan L.A, Vuong J. and Vuong P, “Fuzzy Logic Based Smart Controller for a Cryogenic Cooler”, Proceedings 1996 World Automation Congress (WAC ’96), International Symposium of Soft Computing in Industry (ISSCI), May 27-30, 1996, Volume 5, pp.481-488.

[3] Madni Asad M, Wan L. A, Kuo D and Vuong J, “Sensor Based Microcontroller Unit with Built In Fuzzy Inference for an Endometrium Ablator”,Proceedings 1995, 3rd European Congress on Intelligent Techniques and Soft Computing (EUFIT ’95), August 28-31, 1995, pp. 1621- 1624.

[4] Zadeh, L.A., "Fuzzy Sets," Information and Control, 8,338353, 1965.

[5] Brubaker, David I., "Fuzzy Logic Basics: Intuitive Rules ReplaceComplex Math," EDN, June 18, 1992, pp.111.

[6] Glenn A, “Fundamentals of Fuzzy Logic Part I & II”, SENSORS, April 1993.

[7] Earl Cox, The Fuzzy Systems Handbook (1994), ISBN 0-12-194