Eng.& Technology, Vol.25, No.9, 2007

Implementing Fuzzy Logic Controller Using VHDL

Yousra A. Mohammed* **Leena K. Hashim ****

Received on 19/4/2006
Accepted on 2/8/2007

Abstract

Design of a Fuzzy Logic Controller (FLC) requires more design decisions than usual, for example rule base, inference engine, defuzzifiction, and data pre- and post processing.

This paper describes a way to implement a simple (FLC) in VHDL, there are three parts to fuzzy controller, the fuzzification of the inputs, the defuzzification of the outputs, and the rule base. The controller that is implemented has demonstrated a 2-input, 1-output fuzzy controller with 5-membership functions.

This paper identifies and describes the design choices related to simple fuzzy logic controller, based on an international standard which is underway.

In this paper, we propose a VHDL-based logic synthesis approach for designing to reduce design time. A complete description of the controller (A fuzzier, defuzzifier parts and a rule based are written in VHDL by using Active_HDL and are assembled and synthesized using logic synthesis tools of ISE4.1 software. The efficiency of the generated hardware is explored for FPGAs technology.

Keywords: Fuzzy Logic Controller, VHDL, and FPGA.

تنفيذ مسيطر المنطق الضبابي باستخدام VHDL

الخلاصة

ان تصميم مسيطر المنطق الضبابي يتطلب قرارات فوق العادية، على سبيل المثال بخصوص أجزاءه الثلاثة (fuzzier, defuzzifier and a rule based).

تصف هذه المقالة كيفية تصميم هذا المسيطر البسيط بأجزاءه الثلاثة بأستخدام لغة VHDL كبيئة برمجية و الـ FPGA كبيئة مادية وذلك لتقليل زمن التصميم. ان هذا المسيطر مصمم ليعمل مع (2-input, 1-output & 5-Membership function). تصف هذه المقالة ايضا اختيارات التصميم المتعلقة بهذا المسيطر اعتمادا على القياسات الدولية الجارية.

واخيرا قد تم تصميم هذا المسيطر بأجزاءه الثلاثة بأستخدام برنامج Active_HDL لغرض الـ Functional Simulation, اما لغرض الـ synthesis والتنفيذ (Implementation) فقد تم بأستخدام برنامج ISE4.1.

1- Introduction

Fuzzy Logic Controller (FLC), has been introduced as controller to simulate complex system exploiting human operators experience and\or control engineers knowledge. Fuzzy controllers use a rule base to describe relationships between the input variables. Implementation of a detailed rule base increases in complexity as the number of input variables grows and the range of operation for the variables becomes more defined [1,2].

* Dept of Electrical & Electronic Eng., University of Technology
** Dept of Control & System Eng., University of Technology

https://doi.org/10.30684/etj.25.9.3
2412-0758/University of Technology-Iraq, Baghdad, Iraq

Today, still only a few full custom or semi-custom integrated fuzzy controllers exist and most of them are assembled from standard cells at the gate level [3]. Our design approach presented here is a high-level one. The usage of high-level modeling methodologies for modeling fuzzy controllers reduces development time significantly, making rapid design of custom fuzzy hardware possible.

VHDL [4] for design capture and VHDL-based logic synthesis are an efficient method for designing complex hardware.

The fuzzifier and defuzzifier parts of the system, which incorporate a lot of mathematical computations, are described in VHDL as a hand-coded design. We assemble the fuzzy control system, and synthesize a gate level description for field programmable gate array (FPGA) technology.

This paper is organized as follows: In section 2 we give a brief introduction to fuzzy controllers. Section 3 shows a proposed controller design description. Section 4 lists design choices made, and brings a description of the design flow. A fuzzifier, a defuzzifier and a rule-base of the controller are modeled in VHDL, as presented in section 5. The developed model is synthesized and implemented in section 6. Conclusions are presented in section 7.

2- Overview of Fuzzy Controllers

The fuzzy controller takes input values from the real world. These values are referred to as "crisp" since they are represented as a single number, not a fuzzy one. In order for the fuzzy controller to understand the inputs, the crisp input has to be converted to a fuzzy number. This process is called fuzzification.

The next step in the fuzzy control process is the implementation of the rule base. This is where the fuzzy inputs are compared and on the membership of each, the fuzzy output is chosen. For a 2 input system with 5 membership functions for each input, there would be 25 fuzzy rules to compute since they are anded together.

The final step is to convert the fuzzy outputs of the rule-base to crisp ones. This process is known as defuzzification. This process will take a fuzzy number and apply it to a membership function to achieve the crisp number that will be sent to the real world [5,6,7].

3- The proposed FLC

This section, introduces a simple controller system, which serves as an example to the various steps involved when designing a fuzzy controller. The controller is sensitive to two input variables (error (e) and change in error (ce)) and produces one output variable (control Action (u)).

For each of these inputs and output, five symmetric and triangular-shaped membership functions are defined and evenly distributed on the appropriate universe of discourse.

A membership function gives the degree of membership of an input value to every fuzzy set. The input may belong to more than one fuzzy set. In the example application, the error and change in error are divided into five overlapping fuzzy sets called "Large Positive", "Small Positive", "Zero", "Small Negative", and "Large Negative".

The fuzzified input variables are input to the rule base, the sample controller uses 25 rules, displayed in Table (1) to find an output fuzzy set.
In our implementation we used one of the most commonly used defuzzification methods which is the Center of Areas (COA), which is computationally expensive since it includes division.

4- Design choices, design flow and environment

We have implemented a dedicated fuzzy chip, capable of performing stand alone operations, rather than extend some general purpose processor with fuzzy instructions. Our decision is influenced by several facts:
_ a dedicated chip offers the highest execution speed,
_ design time is short, due to the usage of synthesis and high-level design tools,
_ low cost of real-estate.

For designing three parts (fuzzier, rule-base, and defuzzier) of FLC, we have used a behavioral architecture of VHDL language approach that means encoding these parts directly in VHDL. The reasons for this choice are:
_ it reduces the design time,
_ global design functionality is evaluated in a short time,
_ different design choices are quickly explored,
_ edit-compile-debug cycle is fastened [8],
_ it provides little or no insight into the physical implementation of the entity but provides very flexible code that can be modified quickly [9].

Modeling a rule base, a fuzzifier and a defuzzifier of a fuzzy control system perform mathematical operations, such as multiplications and divisions. For describing such parts, direct encoding in VHDL at the register-transfer level is more efficient in VHDL
We have used field-programmable gate arrays (FPGAs) as hardware platform, because they are complex enough to be used for rapid prototyping of complex systems, and for the final implementation of a chip, if only a small number of pieces is needed. They offer more flexibility than ASICs, as when a design is not needed any more; the chip can be reprogrammed for a new hardware.
For the purpose of design and implementation of this controller, a VHDL model describing the design operation is presented, this VHDL model uses Xilinx families (virtex2-x2v4000) FPGA [10], the model is verified and synthesized using analysis and compiled tools provided by ISE4.1 software.

5- VHDL descriptions of the fuzzy logic controller

The fuzzifier, the defuzzifier and the rule-base used in our fuzzy controller have been described in hand-optimized VHDL.
Inputs to the fuzzifier are input variable, and outputs are variables containing membership degrees for all input fuzzy sets. A part of the VHDL description of a fuzzifier is given in Fig.(1). In the architecture body, membership degrees of the input signal are calculated for each fuzzy set.
The defuzzification is modeled in a similar way, a part of the VHDL

description of the defuzzifier is listed in Fig.(2). The input is a membership set of variables and the output is the control variable.
Fig.(3) shows part of the VHDL description of the rule-base of the controller, in the VHDL coded of the rule-base; if-then statement is used, which may be a more natural choice in the coding process, also each rule is represented in the distinct processes.

6- Synthesis and technology mapping

After design verification, the ISE4.1design compiler is used to perform logic synthesis. The result of synthesis is a gate-level description of the controller.
Fig.(4) shows the results of synthesis of the example fuzzy controller with two input and one output variables, and a rule base consisting of twenty-five rules.
We have compiled our models targeting technologies: Xilinx FPGAs (x2v4000). For FPGA implementation, the measure area efficiency is expressed in a number of CLBs (configurable logic blocks).

7- Conclusion

In this paper, we have presented approach for describing fuzzy logic controller using VHDL description and FPGA technology.
The selection of a certain FPGA devices depends on many parameters like: number of input and output variables of the system, the cost and delay required in implementing the system and reprogramming capability. The Xilinx (x2v4000) FPGA device is the one suitable for implementing FLC because it satisfies all above mentioned requirements with reasonable optimization.
The FLC based FPGA technology substantially simplifies the design loop. The result in some significant benefits, such as development time, simpler design and faster time to market.

8-References

[1] Rokaia Sh.; Tuning of Fuzzy Logic Controller Using Genetic Algorithm; Thesis, Department of Electrical Engineering, Al-Mustansyriah University, October 2001.
[2] Charles P. & Datta G.; A Comparison of Robustness Fuzzy Logic PID, Sliding Mode Control; Technical report; University of California; 1991.
[3] Aptoronix Inc.; Why Use Fuzzy Logic; 2000.
[4] IEEE; IEEE Standard VHDL Language Reference Manual, IEEE, NY,
1988. IEEE Standard 1076-1987.
[5] Dan Simon; Introduction to Fuzzy Control; August 2002.
[6] Kevin M. Passino & Stephen Yurkovich; Fuzzy Control; Addison Wesley Longman; 1998.
[7] An Introduction to FL with Simple BASIC Implementation can be found at: http:// www.fuzzy-logic.com/.
[8] Brown S. & Vranesic Z.; Fundamentals of Digital Logic with VHDL Design ; MCGrow-Hill Company, 2002.
[9] Thararam S. Muk; Design and FPGA Implementation of an Adaptive Demodulator; M.SC. Thesis, University of Kansas; 2000.
[10] Xilinx; Vertex Series Configuration Architecture User Guide, Application Note; 2006.

Table (1) Membership functions distribution

Change In Error \ Error	NB	NS	Z	PS	PB
NB	NB	NB	NB	NS	Z
NS	NB	NB	NS	Z	PS
Z	NB	NS	Z	PS	PB
PS	NS	Z	PS	PB	PB
PB	Z	PS	PB	PB	PB

```
 --to find the member function
CENB<=(abs(cnb-e))/cj ;
CENS<=(abs(cns-e))/cj;
CEZ<=(abs(cz-e))/cj;
CEPS<=(abs(cps-e))/cj;
CEPB<=(abs(cpb-e))/cj;
CCNB<=(abs(cnb-ce))/cj;
CCNS<=(abs(cns-ce))/cj;
CCZ<=(abs(cz-ce))/cj;
CCPS<=(abs(cps-ce))/cj;
CCPB<=(abs(cpb-ce))/cj;
end process;
MENB<= 0.0 WHEN (CENB>1.0) ELSE (1.0-CENB);
MENS<= 0.0 WHEN (CENS>1.0) ELSE (1.0-CENS);
MEZ<= 0.0 WHEN (CEZ>1.0) ELSE (1.0-CEZ);
MEPS<= 0.0 WHEN (CEPS>1.0) ELSE (1.0-CEPS);
MEPB<= 0.0 WHEN (CEPB>1.0) ELSE (1.0-CEPB);
MCNB<= 0.0 WHEN (CCNB>1.0) ELSE (1.0-CCNB);
MCNS<= 0.0 WHEN (CCNS>1.0) ELSE (1.0-CCNS);
MCZ<= 0.0 WHEN (CCZ>1.0) ELSE (1.0-CCZ);
MCPS<= 0.0 WHEN (CCPS>1.0) ELSE (1.0-CCPS);
MCPB<= 0.0 WHEN (CCPB>1.0) ELSE (1.0-CCPB);
```

Figure (1)A part of the VHDL description of a fuzzifier

```
SCA<=(pub11+pub21+pub31+pub41+pub12+pub22+pub32+pub42
+    pub13+pub23+pub33+pub43+pub14+pub24+pub34+pub44+
   pub15+pub25+pub35+pub45);
NNCA<=(((cpb*pub11)+(cpb*pub21)+(cpb*pub31)+(cps*pub41))+
            ((cpb*pub12)+(cps*pub22)+(cps*pub32)+(cns*pub42))+
            ((cpb*pub13)+(cps*pub23)+(cns*pub33)+(cnb*pub43))+
            ((cpb*pub14)+(cns*pub24)+(cns*pub34)+(cnb*pub44))+
            ((cns*pub15)+(cnb*pub25)+(cnb*pub35)+(cnb*pub45)));

process(SCA,NNCA)
 begin
          if (sca=0.0) then
                    U<=0.0;
          else
                    U<=((nnca/sca)/sf);
          end if;
          end process;
```

Figure (2) A part of the VHDL description of defuzzifier

```
          --TO REPRESENT THE RULES (25-RULES) THE DOF PART
--the rules of mcnb----------
PROCESS(MCNB)
BEGIN
          if (mcnb = 0.0) then pub11<=0.0;
          elsif (mcnb>menb) then pub11<=menb;
          else                pub11<=mcnb;
          end if;

          if (mcnb = 0.0) then pub21<=0.0;
          elsif (mcnb>mens) then pub21<=mens;
          else                pub21<=mcnb;
          end if;
          if (mcnb = 0.0) then pub31<=0.0;
          elsif (mcnb>mez) then pub31<=mez;
          else                pub31<=mcnb;
          end if;
          if (mcnb = 0.0) then pub41<=0.0;
          elsif (mcnb>meps) then pub41<=meps;
          else                pub41<=mcnb;
          end if;
                    end process;
```

Figure (3) A part of the VHDL description of rule-base (MCNB)

```
Design Information
------------------
Command Line   : map -p xc2v4000-ff1152-5 -cm area -k 4 -c 100 -tx off aa.ngd
Target Device  : x2v4000
Target Package : ff1152
Target Speed   : -5
Mapper Version : virtex2 -- $Revision: 1.58$
Mapped Date    : Fri Jan 13 20:08:40 2006

Design Summary
--------------
  Number of errors:      0
  Number of warnings:    0
  Number of Slices:              2,006 out of  23,040    8%
  Number of Slices containing
    unrelated logic:               0 out of   2,006    0%
  Total Number 4 input LUTs:     3,902 out of  46,080    8%
    Number used as LUTs:             3,785
    Number used as a route-thru:       117
  Number of bonded IOBs:           736 out of     824   89%
  Number of MULT18X18s:             56 out of     120   46%
Total equivalent gate count for design:  259,478
Additional JTAG gate count for IOBs:  35,328
```

Figure (4) shows the results of synthesis