# Design Representation of the Multipurpose Fuzzy Logic Controller using Hardware Description Language

[1]Zeyad Assi Obaid, [1] Nasri Sulaiman and [1] M. N. Hamidon, [2] Mohammed Obaid Ali
[1] Department of Electrical & Electronic Engineering, Faculty of Engineering,
University Putra Malaysia,
43400 UPM Serdang, Selangor Darul Ehsan, Malaysia.
eng.alhamdany@yahoo.com,

***Abstract*- A large numbers of fuzzy control applications with the physical systems require a real-time operation to interface high speed constraints; higher density programmable logic devices such as field programmable gate array (FPGA) can be used to integrate large amounts of logic in a single IC. This paper presents a design representation of the multipurpose fuzzy logic controller using hardware description language, in order to make a comparison between other designs in the physical systems. The controller is (Proportional – integral – derivative Fuzzy Logic controller (PIDFLC)), with programmable fuzzy sets and programmable rule table using *VHDL* language. The method used to design the fuzzy logic controller is to design it with the aid of conventional PID control to serve wide range of the physical systems efficiently. The design representation is presented using RTL viewer in the ALTERA Quartus II program. Timing test results for the proposed controller was very fast ranging from 20.8 nano second, and the controller has the ability to serve a wide range of the systems.**

***Keywords* - Fuzzy logic controller, Altera, PID controller.**

## I. INTRODUCTION

Fuzzy Logic has been successfully applied to a large number of control applications. The most commonly used controller is the PID controller, which requires a mathematical model of the system. Fuzzy logic controller provides an alternative to PID controller since it is a good tool for the control of systems that are difficult in modeling. The control action in fuzzy logic controllers can be expressed with simple "if-then" rules [1]. Fuzzy controllers are more sufficient than classical Controllers because they can cover a much wider range of operating conditions than classical Controllers, and fuzzy controllers can operate with noise and disturbances of different nature. The used method most often to implement a fuzzy controller is to use it as a computer program on a general purpose computer, higher density programmable logic devices such as Field-Programmable Gate Array (FPGA) can be used to integrate large amounts of logic in a single IC. FPGA provide additional Flexibility: they can be used with tighter time-to-market schedules. FPGA places fixed logic cells on the wafer, and the FPGA designer constructs more complex functions from these Cells. The term field programmable highlights the customizing of the IC by the user, rather than by the foundry manufacturing the FPGA. Several researchers discussed the design of hardware fuzzy logic controller. Number of these works were specialized in control application [2], [3], and were aim to get better control responses. Others were concerned in developing general fuzzy logic processors. Their searches were concern using new techniques in fuzzy algorithm, to get higher processing speed versus low utilization of chip resource [4], [5].

## II. CLASSICAL PID CONTROLLER

In a P controller the control deviation e(t) is produced by forming the difference between the process variable yp(t) and the desired output yd(t); this is then amplified to give the anipu1ating variable, which operates a suitable actuator. The P controller simply responds to the magnitude of the deviation and amplifies it. As far as the controller is concerned, it is unimportant whether the deviation occurs very quickly or is present over a long period. Beside P component, there are other control components that behave in the same way mentioned below:

- The D component responds to changes in the process variable.
- The I component responds to the duration of the deviation. It sums the deviation applied to its input over a period of time.

The D and I components, are often combined with a P component to give PI, PD or PID controllers [6].

## III. STRUCTURE OF THE FUZZY LOGIC CONTROLLER

Fuzzy logic controller contains of three parts in any design, fuzzifier, inference engine or rule base and defuzzifier as shown with more details in the following sections. Figure (1) shown the structure of the fuzzy logic controller.

This limits the changing of the shapes of fuzzy sets. However, this restriction is widespread in many fuzzy control systems.

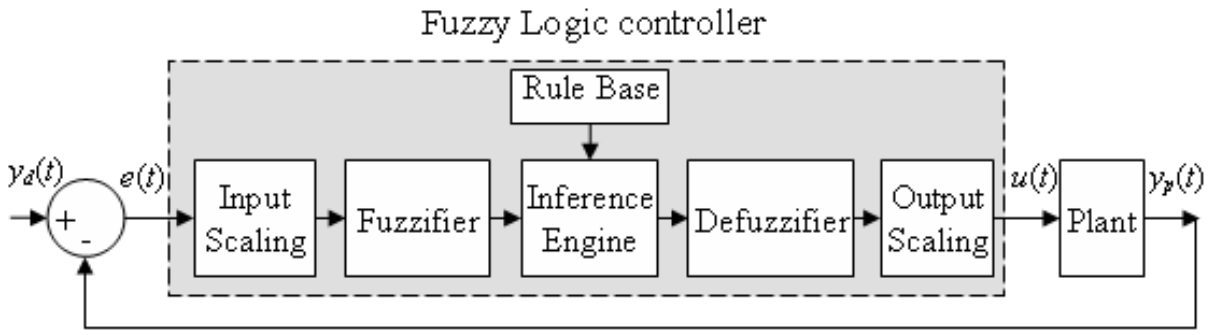


Fig. 1. Structure of fuzzy logic controller with unity feedback control system.

*A. The Fuzzifier Block*

Fuzzification process is performed using two fuzzifier blocks, one for each input variable. Each fuzzifier block takes the input variable and produces four output values represent the sequence numbers of the two active fuzzy sets, (i and i+1), and the membership degrees of the variable in each one of them, ($\mu_i$ and $\mu_{i+1}$). Fuzzifier block consists of three elements: memory module (called Input fuzzy sets' memory), inverter, and incremental. The memory module is used as a lookup table that stores membership values and active fuzzy set number for each entry value of input. Membership functions of any shape could be implemented in this memory by choosing the right memory words that represent the desired membership functions accurately. The memory module was implemented using core utility provided by Xilinx core generator system as a read-only memory (ROM). Each word in the input fuzzy sets' memory is divided into two parts. The first part is 3 bits data word represents the sequence number of the first active fuzzy set. The sequence number of the second active fuzzy set is obtained by adding one to the sequence number of the first active fuzzy set using the incremental. Assigning 3 bits for the sequence number of the fuzzy set will restrict the maximum number of fuzzy sets for each input variable to 8 fuzzy sets. The second part of memory word is 6 bits data word that represents the membership value of input in the first active fuzzy set. The membership value of input in the second active fuzzy set can be obtained by subtracting the membership value of the input in first active fuzzy set from one [4], [5], [7].

This dictates that the summation of membership values of two consecutive fuzzy set is always equal to one, as in the following equation:

$$\mu_i + \mu_{i+1} = 1 \tag{1}$$

*B. Inference Engine Block*

The Inference Engine Block used in the proposed design is based on active rule selector mechanism. Active rules selector block uses the information delivered from fuzzifier about active fuzzy sets, (have nonzero membership values), to launch only active rules. In this way, using an active rule selector, the number of rules to be processed will be reduced according to this equation:

$$\text{Number of active rules} = V^m \tag{2}$$

Where m is the number of inputs, and V is the maximum number of overlapped fuzzy set. In the proposed design, it assumes that m = 2 and V = 2. Hence, the number of active rules at each time is $V^m = 2^2 = 4$: rules. In addition to active rules selector block, inference engine involve two other blocks: rule memory (contains rule consequent) and minimum circuit (circuit to calculate the applicability degree for each active rule) [5].

*C. Defuzzifier Block*

The defuzzification process is performed in the Defuzzifier block using the Centroid method defined by Equation below:

$$z = \frac{\sum_{k=1}^{N} \mu_k * \beta_k}{\sum_{k=1}^{N} \mu_k} \tag{3}$$

Where N represents the number of the rules $\mu_k$ is the degree of the applicability of the kth rule; _k is the defuzzified value of the output membership function of the kth rule [8]. The

Defuzzifier involves two accumulators, one multiplier, and one divider. The defuzzifier block accepts four rules consequent and their membership degrees from the inference engine, (sequentially, in four clock cycles), and produces a crisp output to the output gain block, The membership degrees and rules consequents are delivered from the inference engine in a sequential manner in four consecutive clock cycles, instead of being produced in parallel in one clock cycle. This will enhance (reduce) the used area of the target FPGA, at the expense of increasing time interval between input latching and output producing [5].

## IV. STRUCTURE OF THE PROPOSED CONTROLLER

The main block in the PDFLC is the fuzzy inference block. The proposed fuzzy inference block is two inputs, one output fuzzy system of Mamdani type that uses singleton membership functions for the output variable (it could also be considered as a Sugeno type with constant rule consequents. The first input is the error signal e(n), and the second input is the rate of change of error signal defined as the difference between two consecutive error values and given as:

$$\Delta e(n) = e(n) - e(n-1) \tag{4}$$

Where e(n-1) is the error from previous sampling. Before entering the fuzzy inference block, each one of these two inputs is multiplied by a gain coefficient inside the PDFLC, (Kp and Kd or Kp and Ki). In similar manner, the output of the fuzzy inference block is multiplied by a gain coefficient inside the PDFLC, (Ko). At the same time, the output of the fuzzy inference block in the second PDFLC is multiplied by a gain coefficient then accumulated to form the uPIFLC. Both outputs (uPDFLC and uPIFLC) are added together to form the PIDFLC output (uPIDFLC) as shown in figure (2).

## V. FPGA DESIGN CONSEDERATIONS

The chosen target device family in the proposed design is Virtex FPGAs family form Xilinx Company. Virtex FPGAs family offers a useful criterion to the proposed design, which is the internal RAM block. Virtex FPGAs incorporate several large block memories. These complement the distributed Look Up Table (LUT), that provides shallow RAM structures implemented in configurable logic blocks (CLBs). This criterion is very useful because fuzzy system almost needs large storage element to store fuzzy sets information and rule table [5], [8].

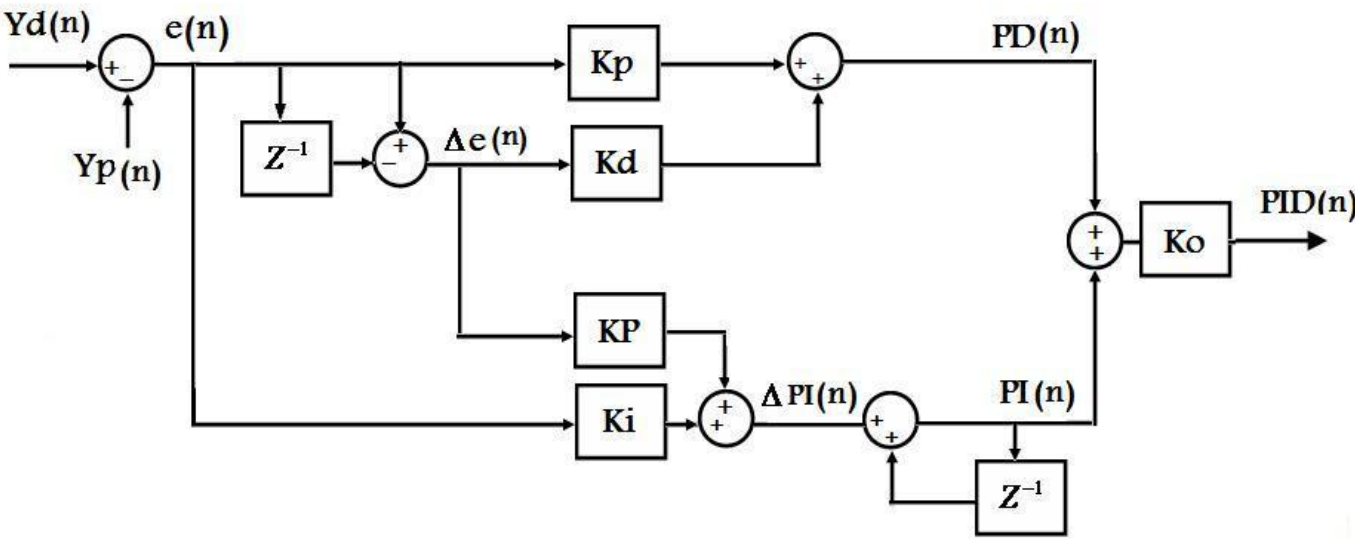


Fig. 2. The main structure of the proposed PID fuzzy logic controller.

## VI. COMPILATIONS AND TIMING TEST RESULTS

The proposed design was done using VHDL language; the proposed design was tested during ALTERA Quartus II 6.1 program with RLT viewer in order to check the design inputs/outputs pins. Figure (3) shows the flow summary for

the design after testing and compilation.

| | |
|---|---|
| Flow Status | Successful - Sun Mar 22 16:29:53 2009 |
| Quartus II Version | 6.1 Build 201 11/27/2006 SJ Web Edition |
| Revision Name | PID |
| Top-level Entity Name | controller |
| Family | Cyclone II |
| Device | EP2C50U484C7 |
| Timing Models | Final |
| Met timing requirements | Yes |
| Total logic elements | 231 / 50,528 ( < 1 % ) |
| Total combinational functions | 215 / 50,528 ( < 1 % ) |
| Dedicated logic registers | 54 / 50,528 ( < 1 % ) |
| Total registers | 54 |
| Total pins | 127 / 294 ( 43 % ) |
| Total virtual pins | 0 |
| Total memory bits | 0 / 594,432 ( 0 % ) |
| Embedded Multiplier 9-bit elements | 2 / 172 ( 1 % ) |
| Total PLLs | 0 / 4 ( 0 % ) |

Fig. 3. Compilation reports flow summary.

Analysis & Synthesis Resource Usage Summary

| | Resource | Usage |
|---|---|---|
| 1 | Estimated Total logic elements | 216 |
| 2 | | |
| 3 | Total combinational functions | 216 |
| 4 | ⊟ Logic element usage by number of LUT inputs | |
| 5 | -- 4 input functions | 43 |
| 6 | -- 3 input functions | 105 |
| 7 | -- <=2 input functions | 68 |
| 8 | | |
| 9 | ⊟ Logic elements by mode | |
| 10 | -- normal mode | 97 |
| 11 | -- arithmetic mode | 119 |
| 12 | | |
| 13 | ⊟ Total registers | 54 |
| 14 | -- Dedicated logic registers | 54 |
| 15 | -- I/O registers | 0 |
| 16 | | |
| 17 | I/O pins | 0 |
| 18 | Embedded Multiplier 9-bit elements | 2 |
| 19 | Maximum fan-out node | clk |
| 20 | Maximum fan-out | 54 |

Fig. 4. Analysis & synthesis resource usage summary.

After design test and design compilation, for every design we have to synthesis and analysis our design, figure (4) shown Analysis & synthesis resource usage summary.

In our design representation, the first part of the fuzzy logic controller is fuzzifier, this part in our design contain of two inputs as mansion above, the first input is the error, and the

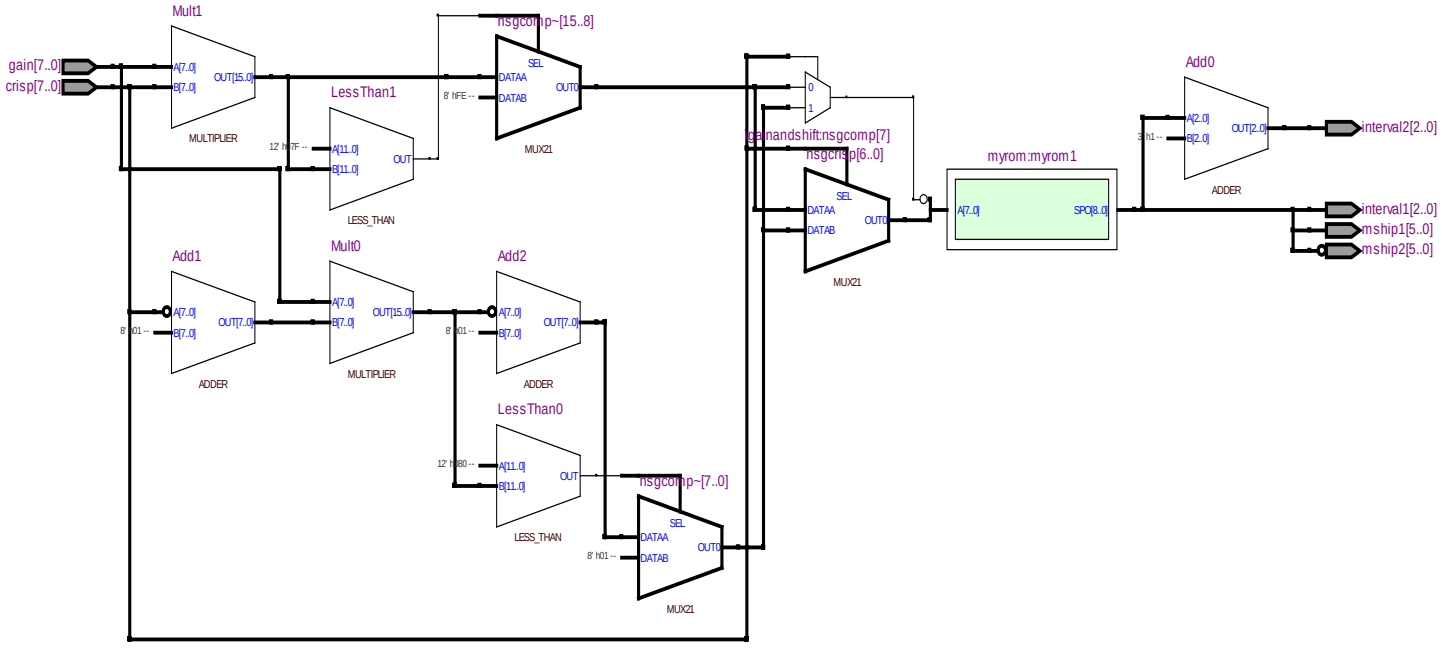


Fig. 5. One input fuzzifier.

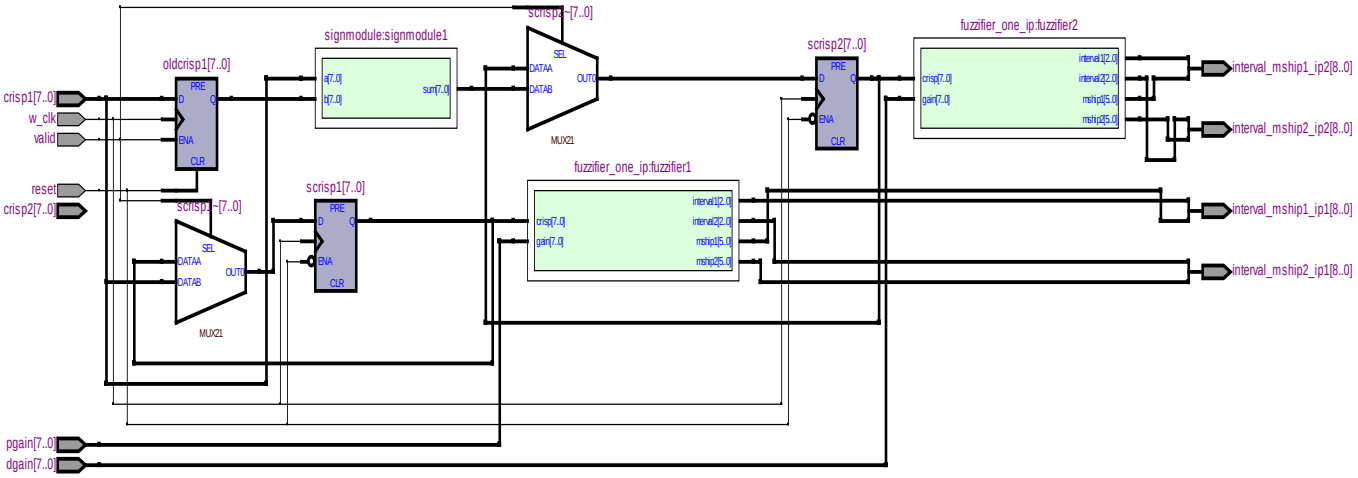


Fig. 6. Two input fuzzifier.

For other two part of our design (inference engine and defuzzifier), figure (7) shown the design representation of the inference engine for our design.

For the third part of our design, figure (8) shown the design representation for the defuzzifier which contain of the output

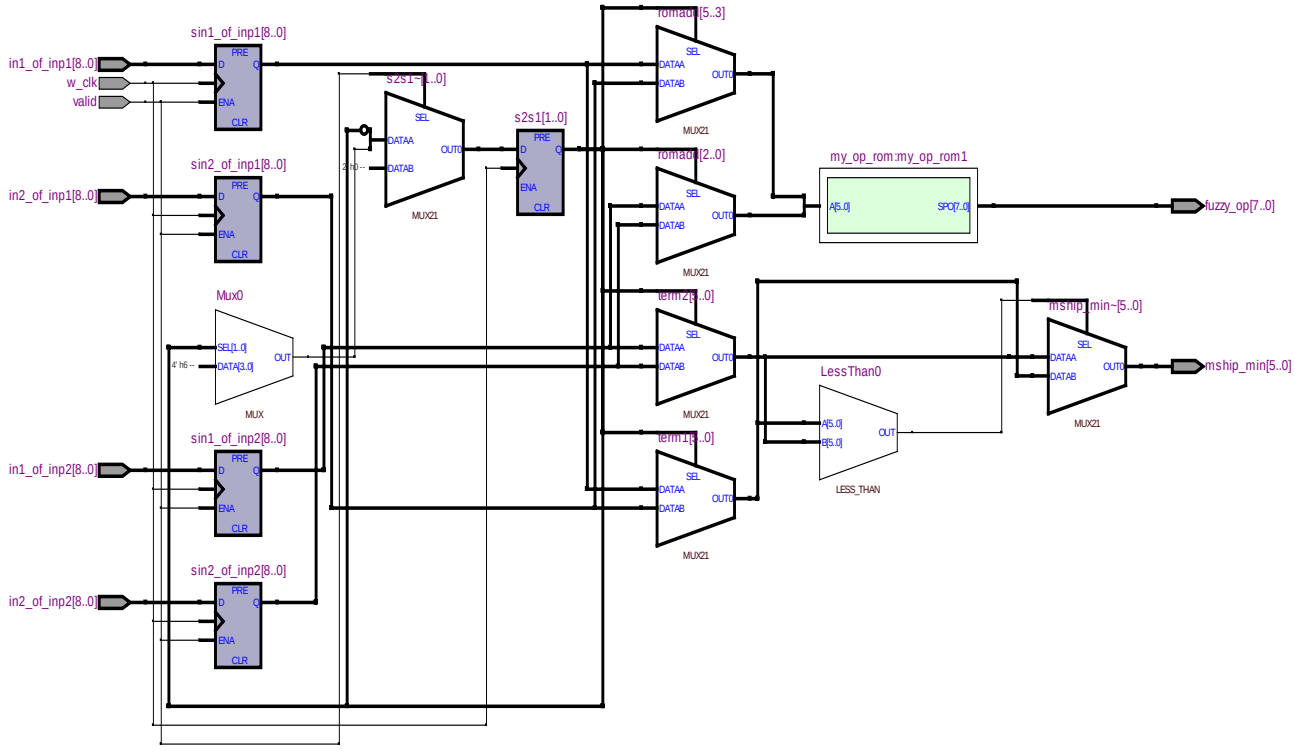


Fig. 7. Inference engine or fuzzy inference.

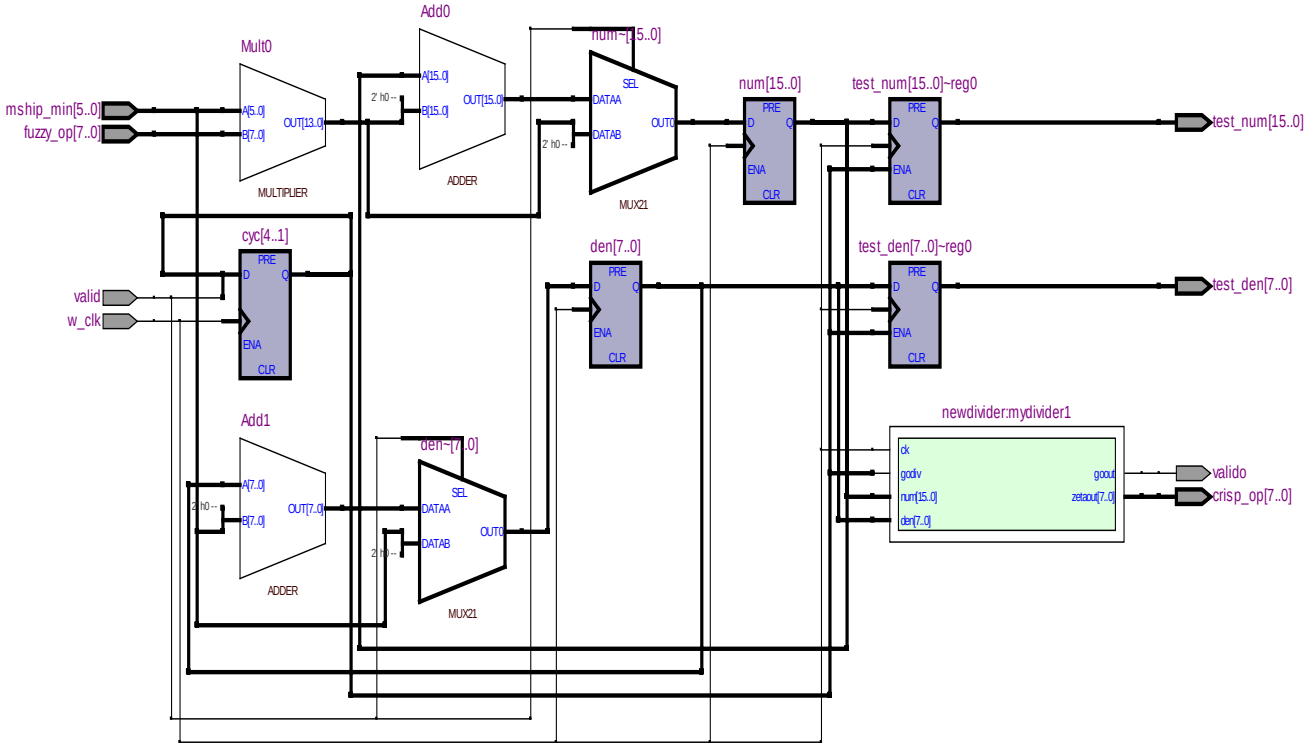


Fig. 8. Structure of the defuzzifier block.

In the timing test for our design, we got that the maximum actual time required for PD controller is 20.803 ns between opgain (2) and the output (3), as shown in figure (9).

Timing Analyzer Summary

| | Type | Slack | Required Time | Actual Time | From | To | From Clock | To Clock | Failed Paths |
|---|---|---|---|---|---|---|---|---|---|
| 1 | Worst-case tsu | N/A | None | 4.423 ns | validin | cyc1 | -- | clk | 0 |
| 2 | Worst-case tco | N/A | None | 33.129 ns | defuzzifier:defuzzifier1\|newdivider:mydivider1\|sig_divtemp_1[1] | output[1] | clk | -- | 0 |
| 3 | Worst-case tpd | N/A | None | 20.803 ns | opgain[2] | output[3] | -- | -- | 0 |
| 4 | Worst-case th | N/A | None | -4.175 ns | validin | cyc1 | -- | clk | 0 |
| 5 | Clock Setup: 'clk' | N/A | None | 75.85 MHz ( period = 13.184 ns ) | defuzzifier:defuzzifier1\|test_den[1] | defuzzifier:defuzzifier1\|newdivider:mydivider1\|sig_divtemp_1[1] | clk | clk | 0 |
| 6 | Total number of failed paths | | | | | | | | 0 |

Fig. 9. Timing analyzer summary.

## VII. CONCLUSION

This paper proposed the design representation of the multi purpose fuzzy logic controller using hardware description language, in order to make a comparison with other design using software especially with the physical systems which require a real-time operation. Compilation and timing test results for our design method show that the design using hardware description language is able to produce the same structure for the design using MATLAB. In the timing test for our design, we got that the maximum actual time required for PD controller is 20.803 ns between opgain (2) and the output (3). Therefore the proposed controller will contains the same specification to other design in order to control a wide range of the physical systems with sampling time ranging from milliseconds. This small-size high-speed chip is able to offer adequate accuracy. From this compression, we got that the responses of the proposed controller are very close to the responses of the software-based controllers which designed using MATLAB.

## ACKNOWLEDGMENT

The authors would like to thank firstly, our god, and all UPM staff and all friends who gave us any help related to this work. Finally, the most thank is to our families and to our countries which born us.

## REFERENCES

[1] S.Poorani, T.V.S.Urmila Priya, K.Udaya Kumar and S.Renganarayanan,"FPGA based fuzzy logic controller for electric vehicle", Journal of The Institution of Engineers, Singapore, Vol. 45 Issue 5 2005.

[2] Onur KARASAKAL, Engin YES," Implementation of a New Self-Tuning Fuzzy PID Controller on PLC", Turk J Elec Engin, VOL.13, NO.2 2005.

[3] V. Tipsuwanporn,3 S. Intajag and V. Krongratana, "Fuzzy Logic PID controller based on FPGA for process control", IEEE International Symposium on Industrial Electronics, Vol. 2, pp. 1495-1500, 4-7 May 2004.

[4] Mohammed Y. Hassan and Waleed F. Sharif, " Design of FPGA based PID-like Fuzzy Controller for Industrial Applications", IAENG International Journal of Computer Science, 34:2, IJCS_34_2_0517 November 2007.

[5] Zeyad Assi Obaid, Nasri B Sulaiman, Mazin T. Muhssin "Design of Fuzzy Logic Controller Using FPGA For The Non-Linear Systems" proceeding of the 3rd International Conference on Postgraduate Education In Penang, Malaysia, 16-17 December 2008.

[6] M.Schleicher and F. Blasinger," Control Engineering a Guide for Beginners" Jumo Gmbh & Co. KG, Fulda, Germany, 3rd edition, 2003.

[7] S. Sánchez Solano, A. Barriga, C. J. Jiménez, J. L. Huertas," design and application of digital fuzzy controllers" the Sixth IEEE International Conference on Fuzzy Systems, Vol. 2, pp. 869-874, Barcelona - Spain, July 1-5, 1997.

[8] "Virtex™ 2.5 V Field Programmable Gate Arrays" data sheet DS003, www.xilinx.com.