Asian Journal of Information Technology 5 (12): 1464-1469, 2006
© Medwell Journals, 2006

A Fast Testable Digital Fuzzy Logic Controller

Chemali, H., A. Khellaf, A. Benhamadouche and A. Ballouti
Department of Electronic University of Setif UFAS, Setif 19000, Algeria

Abstract: This study presents a hardware implementation of a two-input-one-output digital fuzzy logic controller. A new functional testing technique, based on fuzzy testing rules, is developed A fuzzy logic controller has been implemented on FPGA using VHDL hardware description language. A testable versatile soft core controller is designed with high degree of flexibility and portability and so numerous applications of this controller are possible. FPGA reprogramming and reconfigurable facilities are the driving force behind getting rapid hardware design improvements. High performance and fast controller responses are achieved by simple parameter tuning and pipelined methods. The pipelined structure adds to the soft core design efficient means for in-depth parameters controlling. As test is of prime interest, each part of this controller has been designed to grow up its testability. FPGA's Look-Up Table are used to prevent temporal fault occurring and to increase controller decision speed. This design approach is carried out not only to improve fuzzy controller speed and reliability but to get a well structured IP core.

Key words: Fuzzy controller, VHDL, test, scan, bist, signature

INTRODUCTION

In recent years, fuzzy systems have been implemented using different architecture and support, Fig. 1 dedicated VLSI and microprocessor are widely used, many applications require a high speed processing and short time decision. In this work, we present an implementation of fuzzy controller on FPGA XC4010 using high description language VHDL with new architecture which satisfies fuzzy system requirements and testability constraints.

The proposed architecture in Fig. 1 derives from the generally fuzzy systems and consists of three blocks, fuzzification block, rule base and inference engine block and finally defuzzification block [1,2].

Functional description of our fuzzy controller is obtained with VHDL; this description can handle multiple system inputs and outputs. The chosen design is a two-input single-output fuzzy controller, where a 7 triangular membership function with a maximum of 2 overlaps and 4 active rules among 49 rules and centre of area defuzzification method's are the main elements considered to build up the circuit in XC4000 FPGA Series which are covered by powerful software in every aspect of design from schematic or behavioural entry to programming. Multiple reprogramming of FPGA helped reconfiguration and thus hardware can be updated as software. All these features make fuzzy systems design easier to debug and modify rules or membership functions of the processing description[3,4].

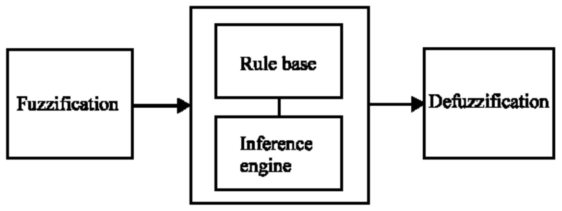

Fig 1: Fuzzy system architecture

To get benefits from FPGA solutions and get rid of external memories, we used functional descriptions which increase fuzzy controller performances in three points:

- Direct access to fuzzy membership Look-Up table (LUT).
- Simultaneous Rules activation.
- Implementation of arithmetic functions.

The different modules of this fuzzy controller are first described and implemented separately for a thorough exploration, optimisationand then the complete system is built relying on global performance and preserving the main features leading to a reconfigurable IP core.

Architecture: The VHDL program of the fuzzy controller is divided into three modules as shown in Fig. 2. A simple assignment produce the different block interlinks. 6, 7, 8 bit inputs and outputs are tried to determine the different issues which would be selected with respect to the targeted tasks and the required precision.

Corresponding Author: Chemali, H., Department of Electronic, University of Setif UFAS, Setif 19000, Algeria

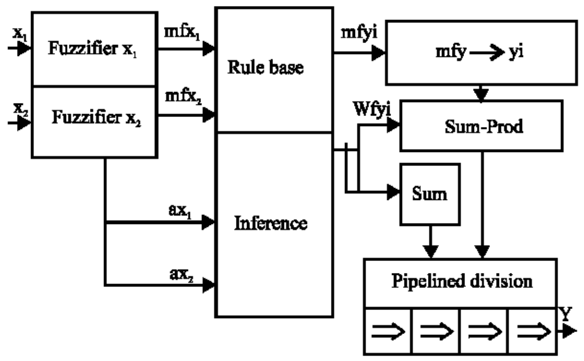

Fig. 2: Fuzzy controller diagram

Fuzzification module: A fuzzification module for each input is defined. Recall that fuzzification is transformation of the crisp data into a corresponding fuzzy set, where the input crisp data is converted and normalized as shown in Fig. 3.

We have used a C++ program to get corresponding fuzzy sets of each input crisp data with respect to the appropriate universe of discourse of the FLC.

All possible membership forms (triangular, trapezoidal, Gaussian…) with 2 overlaps at most can be usedand then discrete fuzzy sets are saved in LUT as illustrated in Fig. 4.

When using 8 bit inputs, we can store fuzzification data in 16x256 bits LUT.

The data is stored in 4 field format in the following order:

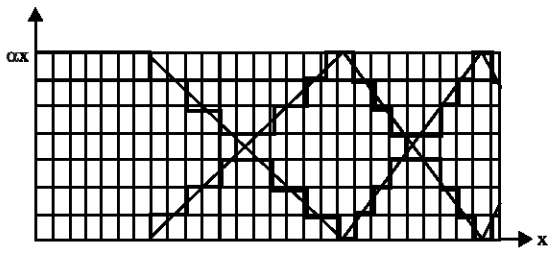

Fig. 3: Normalized data

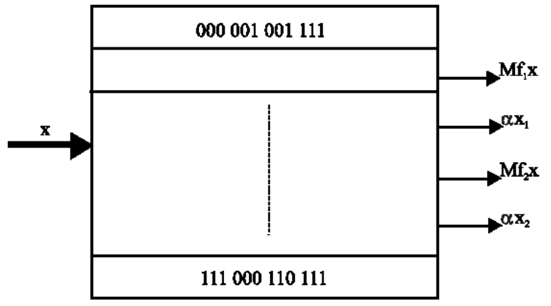

Fig. 4: Fuzzification LUT

Mf1 (3bits), αx1(5bits), Mf2 (3bits) and αx2 (5bits)

Where Mf1, Mf2 are the membership functions and α x 1, α x 2 are their corresponding degrees.

We note that simple input assignments produce the corresponding fuzzy sets of the normalized data and this derives directly from VHDL functional description advantages.

Fuzzy rules and inference

Rule selection: The fuzzy rules represent the core of an FLC and are expressed in linguistic format. We used a Mamdani's method to represent the inputs where the implication of each rule is a singleton[5,6].

If x1 is Mf1-x1 and x2 is Mf2-x2 then U is Mfy

The implication of each rule is stored in the 3 x 49 look Up Table which forms the command matrix of the fuzzy system and in which the outputs singleton are 3 bit-codes.

In this configuration, 4 rules are selected by addressing LUT with the result of membership's combination in pairs as shown in Fig. 5 and expressed as follows:

LUT (Mf1-x1 & Mf2-x2):= 3bit-code singletons.

This task can be executed at once, so all active rule selection should be performed simultaneously.

Inference Engine: The minimum inference method is applied to compute the antecedent membership degrees. To implement this, we have used 4 blocks of Min as showed in Fig. 6.

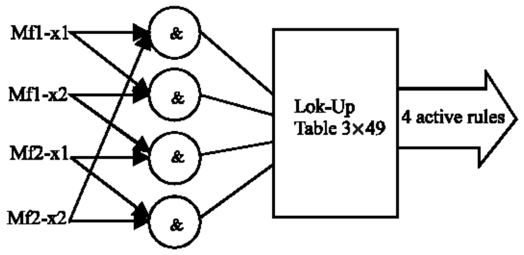

Fig. 5: Active rules

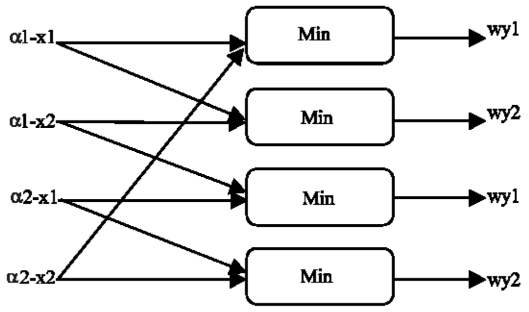

Fig. 6: Minimum computing

Proceeding as defined above has yield to obtain the following advantages:

- Multiple accesses to the matrix rules implied a reduction in rules reading time.
- Rule base replaced by coded singleton reduce FPGA used area.
- The 4 active rules and their corresponding membership degrees are extracted out at the same rising clock edge.

This operation is carried out very fast so all active rule selection and minimum computations are done simultaneously at the same rising clock edge.

Defuzzification module: This module converts the inference results into real values to control the process. As known, controller performance in terms of speed and area depends tightly on the defuzzification design [1, 7]. Thus, appropriate algorithms should be selected for this module.

We have used the centre of area defuzzification method's COG as below:

$$Y = \frac{\sum_{i=1}^{m} y_i wy_i}{\sum_{i=1}^{m} wy_i}$$

Defuzzification module consists on three pipelined functions: multiplication, addition and division. After converting activated rules to a crisp data, we can then get addition and multiplication using usual methods as defined.

It is well known that addition and multiplication actions do not severely limit controller performance, however controller frequency depends firmly on the applied division algorithm and generally consumes a great part of controller area. To avoid performance degradation

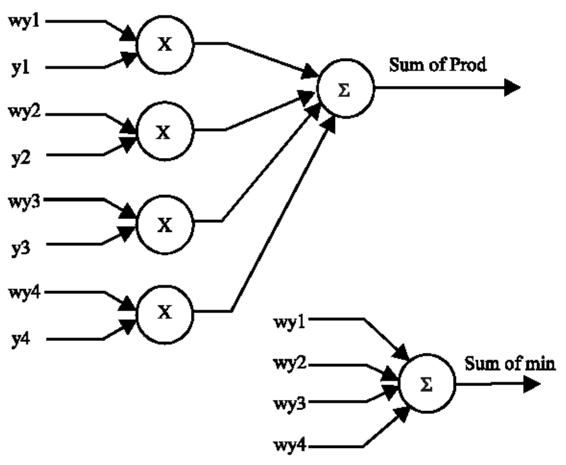

Fig. 7: Multiplication and addition

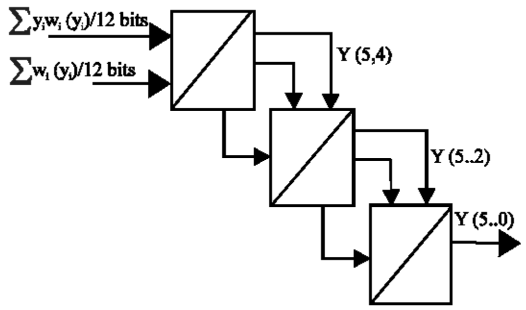

Fig. 8: Pipelined division

and slowing down, we used binary shift division in pipelined data path represent in Fig. 7.

Figure 8 illustrate an examplem of pipelined division (6 bits input/out put), ahere division is divided into 3 successive separate steps: 2 bits then 2 bits then 2 bits.

So each step can treat a data pattern path in a same time as the other steps do, providing a result for the next step at the right tigme. This division takes 3 periods of time.

We can get the division bit per bit or 3 bits per 3bits in several steps according to speed and area limitations.

IMPLEMENTATION

Once detailed description of the different components of our fuzzy controller is obtained, the processing strategy is then defined.

The fuzzification and inference modules are set firstand then a description of the pipelined structure of defuzzification module is derived. This latter depends closely on the number of input-output bits and the selected division algorithm.

To improve fuzzy decision, we extended pipeline mechanism to the global design.

As seen in the Fig. 9 the timing diagram is that of a 6 bit inputs/output fuzzy controller; each function is executed in a single clock time starting at a rising edge.

After simulating and implementing each part of the fuzzy controller, we note that fuzzification and rule-inference modules could be processed at the same clock period, then defuzzification is carried out in 2 steps: Sum and Sum-of-prod in the first step, followed in the second step by the division. In this way, fuzzy controller response will take 5 clock periods instead of 6 periods.

RESULTS

Our Fuzzy controller has two inputs one output, characterized by 7 triangular membership functions with maximum of 2 overlaps and 4 active rules from 49 stored rules.

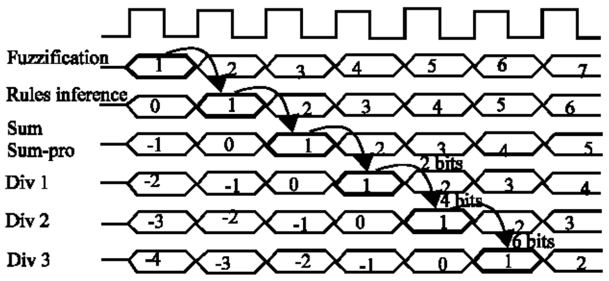

Fig. 9: Timing diagram

After implementing all modules, in a Xilinx VHDL platform on XC4036XL FPGA, for 6 bits fuzzy logic controller through pipelined and non- pipelined architectures, we note that the speed is doubled for our pipelined method in comparison to the other methods as shown in Table 1.

The maximum frequency for this fuzzy controller in XC4000 FPGA series is 18.416MHz. So it can treat inputs each 54ns and make decision every 270ns with 18MFIPS according to pipeline design.

Implementation of different input/output FLC bit-widths in different FPGA devices are shown in Table 2. When varying the number of bits per inputm it is recommended to modify the pipelined division structure in order to boost up fuzzy controller performance.

As far as the quantity of FPGA's CLB (Configurable Logic Blocks) used, an exponential growth resulted despite of the significant improvement in FLC defuzzification architecture.

Fuzzy controller response: Figure 10 illustrates an example of 6 bits fuzzy controller response of VHDL stimuli, this response was transformed with MATLAB to get system surface control.

We can see in Fig. 11 that the surface is smooth with 6bit inputs, so there is no need to enlarge input widthand this controller can perform 18 Mega inferences per second which is very important for Fuzzy FPGA Controller based applications.

Table1: Frequency details for 6 bits

	Pipelined	Non Pipelined
	XC4036XL	XC4036XL
Fuzzification	46.768MHz	46.768MHz
Rules/inferences	42.173MHz	42.173MHz
Defuzzification	19.035MHz	9.035 MHz
Fuzzy controller	18.416MHz	8.904MHz

Table 2:6/7/8 bits Fuzzy controller performance

Bits	Frequency	Area (CLBs) (%)	FPGAs
6	16.459MHz	69	XC4010XL
7	15.222MHz	78	XC4013XL
8	9.652MHz	99	XC4013XL

Fig. 10: Surface control of FPGA

Fig. 11: Fault injection in fuzzification

Fig. 12: Fault injection in defuzzification

CONTROLLER TEST

CLBs are the basic FPGA functional elements to construct fuzzy controller designs. They are a suite of sequential and combinational components structured in matrix like form. Throughout designing our controller, we tried to minimize sequential elements in the implementation of FPGA by selecting VHDL instructions which use solely combinational logic and we configured

LUT of CLB as a functional generator to confine sequential part.

To evaluate testability of our controller we have injected faults into modules of our VHDL description, thus extra information is extracted in order to select the appropriate testing strategy.

We can inject faults into any modules of VHDL description by making changes in instruction or in data content; changing membership function, changing assignment instruction…etc.

Figures 11 and 12 clearly show fault propagation through the entire surface control; in the experiment (Fig. 11), fault is injected in fuzzification LUT. The effect of changing data path in the pipelined division is shown in Fig. 12.

Fault is not masked but it is propagated in all the system, that's why we can use standard techniques of test like BIST with 100% fault covering with 2% add CLB or Boundary scan to test our controller. For Boundary scan, FPGA already contain BS cells usable to get on- line tests without adding any extra module.

Fuzzy testing rules: After injecting faults and studying the resulting traces, we note that all faults are propagatedand we can detect all these faults by scanning the diagonal of the surface control which is done only by scanning the diagonal of fuzzy memory rules.

In order to perform this method of testing, we have added dedicated fuzzy rules TR-Test Rules in the free space of memory rules without any change in original parameter Fig. 13.

We assign to each line and column of memory rules one rule T-S, values of each rule can be determined by signature compaction (MISR) of rules[8] or by adding randomly one of the original rules in each line or column.

In test mode, membership functions activate 4 adjacent fuzzy rules which activate 4 test fuzzy rules, then the fuzzy controller continue normal functioning mode Fig. 14, the result of defuzzification is injected in the both of fuzzification modules as a test vector, only with this condition we can activate the diagonal of memory rules.

With this technique there is no need to add any extra module for testing; a simple organization of the fuzzy controller structure can make testing more effective and faster. The pipeline structure allows testing mode to be executed on online mode by using one rising clock edge of operating cycle.

This fuzzy testing method is a first step leading to consider a new concept of fuzzy generation, fuzzy compactionand fuzzy fault modelling for on line testing and diagnosis. The developed testing method showed that it is 64 faster than standard testing methods such as full scan, partial scan and AD-HOC means experienced on locally defined benchmarks and 0% of area overhead.

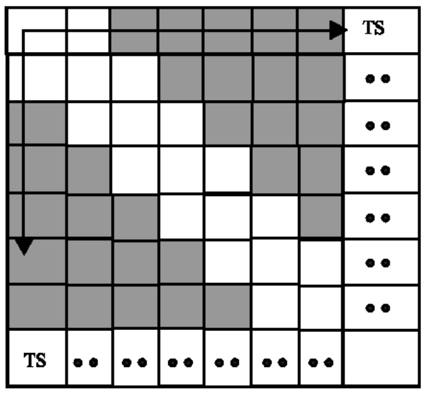

Fig. 13: Fuzzy testing rules

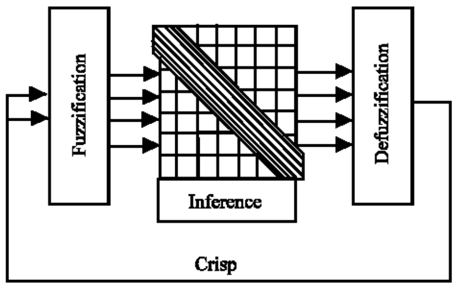

Fig. 14:Fuzzy test mode

CONCLUSION

In conclusion, three new major actions are reported and used to design the fuzzy controller: multiple accesses to look up tables, fuzzy test rules and pipeline methods for computing defuzzification data; this makes our contribution valuable and deserves extra focus to present a structured design method which could lead to get better controller performance for any application.

The implementation of fuzzy systems using VHDL on FPGAs and where each design stage is revisited to improve testability as circuit grows has become a structured design technique but to get more benefits from FPGA and to introduce new testing method, we used VHDL behavioural description that allowed us to introduce new pipelined structures for enhancing global system performance.

Now our efforts are directed to design a fuzzy controller with high degree of reconfiguration in the form of a well structured IP software-hardware core. The fuzzy signature introduced in this design will certainly get more support and interest in the future.

REFERENCES

1. Kandel, A. and G. Langholz, 1998. Fuzzy hardware: Architectures and Applications, Kluwer Academic Publishers Bostan.
2. Dubois, D. and and H. Prade, 1980. Fuzzy sets and systems: Theory and applications, Academic Press.
3. Xilinx, 1999. XC4000E and XC4000X Series Field Programmable Gate Arrays, Xilinx Inc.
4. Muresan, V., D. Crisu and X. Wang, 1997. From VHDL to FPGA a case study of a fuzzy logic controller, proceeding of international conference of young lectures, 83-90, pp: 11-17.
5. Marek, J., L. Grantnerand K. Koster, 1996. Digital fuzzy logic controller: Design and implementation, IEEE transaction on fuzzy systems, pp: 4.
6. Passino, K.M. and S. Yurkovich, 1998. Fuzzy Control. An Imprint of Addison-Wesley Longman.
7. del Campo, I. and J.M. Tarela, 1999. Consequences of the Digitization on the Performance of a Fuzzy Logic Controller, IEEE transaction on fuzzy systems, pp: 1.
8. Laoumri, A., 2002. Test fonctionnel d'un controleur flou these magister Universite de Setif Algerie Dept. electronics.