FUZZY LOGIC CONTROLLED MODEL CAR

Gerardo Aranguren
Dpto de Electrónica y Telecomunicaciones
jtpararg@bi.ehu.es

Luis Nozal
Dpto de Electrónica y Telecomunicaciones
jtplonol@bi.ehu.es

Miguel Rodríguez
Dpto de Matemática Aplicada
maprogom@bi.ehu.es

Enrique García
Dpto de Electrónica y Telecomunicaciones
enrique@bise08.bi.ehu.es

E.T.S. Ing. Ind. y Telec., UPV/EHU, Alameda Urquijo s/n, 48013 Bilbao

Abstract

On this paper, an application of a high performance fuzzy logic hardware controller is explained, a model car is given autonomous character. Development of a fuzzy rule base, aided by a simulator program, is followed by trials made to come to conclusions regarding ideal and real functioning of the hardware developed.

Key words: Fuzzy Controller, FPGA.

1 INTRODUCTION

Many struggles during the last years have been oriented towards the consecution of faster microprocessors and controllers based on smaller instruction sets (RISC technologies), which applied to the new faster RAM memories easily available, permit a faster execution of programs and control tasks [1], [2] y [3]. On this line, a specific fuzzy logic controller (FICIC) has been developed at the Electronic Design Group (G.D.E.) on the University of the Basque Country, Spain [4]. Heavy workloads consisting of thousands of fuzzy rules can be executed in real time [5], [6], and [7],. As an instance application, a model car with five ultrasonic sensors has been endowed with the FICIC circuit as a controller. The complexity of autonomous driving has been an ideal test for the hardware developed, which has had to face with fuzzy rule bases of up to 10,000 fuzzy rule code lines.
FICIC (Fuzzy Inference Controller Integrated Circuit) is a pipeline sequential fuzzy controller endowed with great flexibility and high performance (it obtains the advantageous qualities of both kinds of design: a sequential microprocessor-like processing, that allows for a great flexibility, but with parallel elements, mainly pipeline, that provide an improved efficiency in times). Its design allows for the implementation on PFGAs or ASICs. In order to optimise the design and performance of the circuit, it is necessary to approach the concept of a RISC (Reduced Instruction Set Computer) processor; for that purpose, it is necessary to adapt the fuzzy algorithm and make some simplifications.
Reducing the complexity of a rule-based fuzzy system is essential to optimise the process, decreasing execution times and operation complexity. To design the sequential fuzzy controller, we will consider three simplifications [8]:
- <u>First simplification</u>: term decomposition
- <u>Second simplification</u>: application of Heuristics
-<u>Third simplification</u>: reduction of rules and premises.

The sequential character of the controller proposed can notably reduce the execution time of the rule set, since neither the maximum number of rules, nor the maximum number of subpremises for each rule, need to be processed.

2 SPECIFIC RULE BASE EDITOR AND COMPILER TOOL: FUZZYSOFT

A software package has been developed for the generation of control rules for any given fuzzy system. It facilitates the edition of the input variables of the permitted types, the output variables, and the rules with their corresponding hierarchies, as well as compilation and simulation. This software has been written in C++ and offers the facilities described in [9].

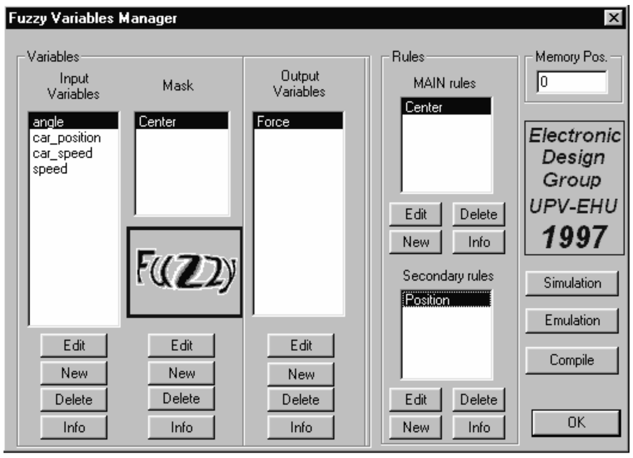


Figure 1. Main screen of the program

3 MODEL CAR APPLICATION

3.1 HARDWARE IMPLEMENTATION

An application has been done in a prototype to drive a car (figure 2).
Figure 3 shows its structure. The Fuzzy Controller is implemented in a FPGA. It reads the program from a

memory and executes it. A microcontroller interfaces input data from a set of sensors and output data to drive and turn the car.

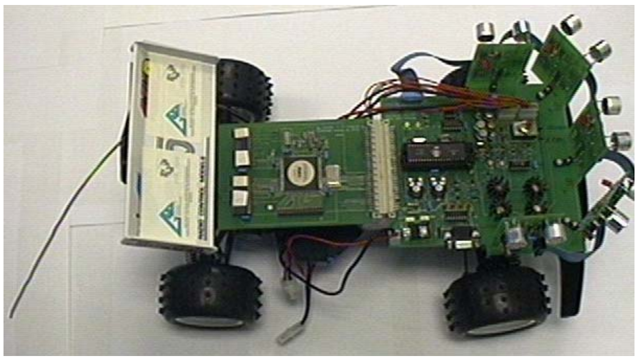


Figure 2. Prototype

The FPGA used is the EPF10K50RC240-3 device of Altera. It reads instructions from the memory, as it has been explained before. The PIC16C74 μC of Microchip receives the distances from sensors to give them to fuzzy controller, and receives results from fuzzy controller to send them for actuators to drive and turn the car.
Ultrasonic sensors measure distances to the wall or obstacles from the time employed by an ultrasonic wave for going and returning. Drives adapt digital signals to act over motors that move the car and turn the wheels.

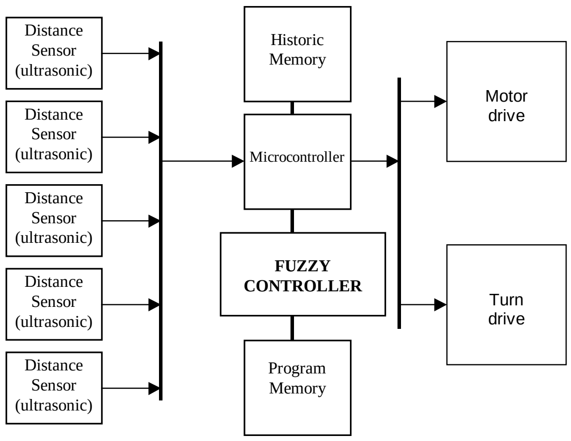

Distance Sensor (ultrasonic)
Distance Sensor (ultrasonic)
Distance Sensor (ultrasonic)
Distance Sensor (ultrasonic)
Distance Sensor (ultrasonic)
Historic Memory
Microcontroller
FUZZY CONTROLLER
Program Memory
Motor drive
Turn drive

Figure 3. Block diagram of prototype

3.2 CAR SIMULATION AND DEVELOPMENT OF THE RULE BASE

As a necessary step towards the development of a useful model car application, a testing tool has been implemented to verify the correct functioning of the system and to test the knowledge base the expert has previously compiled. Such task will be accomplished through the necessary testing in a simulator. Such simulator has been developed with very specific objectives:
- Closest possible approach to the environment and conditions the model will be used on. Design of the simulating path or road, realistic model for the ultrasonic sensors, etc.
- Flexibility of the simulating functions for the consecution of a work bench capable of testing different simulating conditions, environments and inference strategies; all through the use of one sole user interface never minding to change the application code, whatever new conditions it faces.
- High fidelity simulation of the FICIC hardware. This hardware is in charge of the fuzzy control and, therefore, its characteristics (operations order, word width, physical limitations, total memory size, instructions syntax, number of in-variables and actuators, etc...) are simulated just the way they are, in order to obtain the closest possible simulation results to the real conditions.

As shown in figure 4, a driving expert must previously write and compile a fuzzy rule base program (knowledge base) using the FUZZYSOFT. Then, the simulator user will load that program into the simulator SIMCOCHE.EXE (through the use of the functions compiled in FICICDLL.dll, a FICIC controller simulator set of functions), as well as a map for the road or circuit where the car will run the simulation.

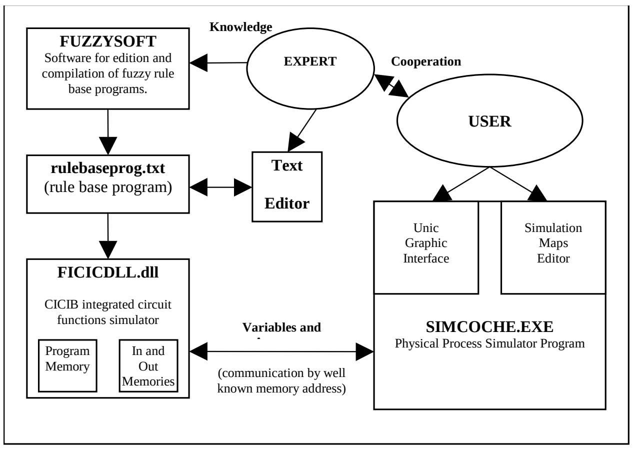


Figure 4. Simulator function scheme

Simulation results will be observed and compared, and the simulator user will suggest the necessary changes in the rule base to the driving expert.

3.2.1 Environment Model. Map Editor

There is a big increase in the difficulty involved in designing a controller and a fuzzy rule base for an autonomous car when it is meant to be capable of working not on one specific circumstance, but on a broad enough set of different environments to ensure viability as a driving application. One rule base can seem to be apparently well designed if it's successfully tested under the restricted conditions of one only driving circuit. Nevertheless that same rule base may fail if original driving circuit conditions change (road width, newer obstacles, more distinct turns, etc.).

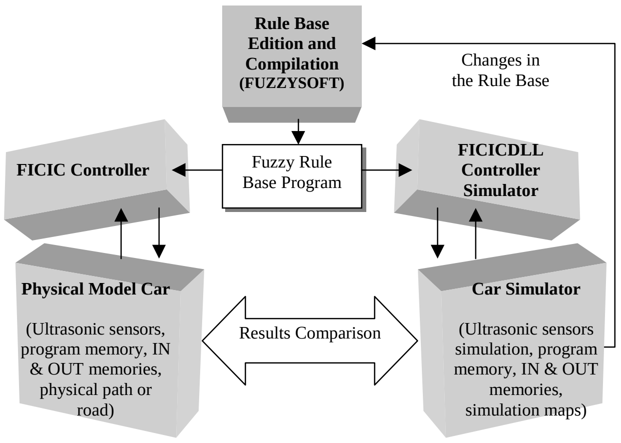

Rule Base Edition and Compilation (FUZZYSOFT)
Changes in the Rule Base
FICIC Controller
Fuzzy Rule Base Program
FICICDLL Controller Simulator
Physical Model Car
(Ultrasonic sensors, program memory, IN & OUT memories, physical path or road)
Results Comparison
Car Simulator
(Ultrasonic sensors simulation, program memory, IN & OUT memories, simulation maps)

Figure 5. Parallelism between hardware implementation and simulation.

Therefore, in order to developing a new universal rule base valid on all types of environments, proofs will have to be done on different simulation scenarios. A **map editor** has been included into the simulator so the user can create and edit the roads on which the car will run. It is presented as a user friendly editor that can generate or load maps saved in binary files with '.map' extension. In this way proofs can be done of different sets of rules (rules bases) on different roads, with different obstacles, etc.

3.2.2 Vehicle Model. Ultrasonic Sensors

The physical implementation of the car is a non-holonomous model car carrying five ultrasonic sensors. It will be simulated as a rectangle, with 5 ultrasonic sensors located in the front side, all oriented with different symmetric angles with respect to the longitudinal axis of the car, just the way it is in the physical model.

The sensors ideal behavior consists on measuring the distance over the straight line originated at the sensor situation and oriented within a determinate angle. That distance measured will be that to the closest object that crosses the sensor line. Nevertheless, the real characteristics of the ultrasonic sensors that will be implemented on the physical model force the simulator to include real ultrasonic sensor effects. Therefore a maximum measurable distance is set, as well as a Maximum Incidence Angle Without Rebound (M.I.A). Any surface (map fence) whose normal angle forms, with the sensor line, an angle superior to the M.I.A., will be considered as a rebound loss and therefore that surface won't be detected by the sensor.

Every clock period, new values are calculated for car position and orientation as well as new sensors readings.

Steer angle and car speed will be determined by the fuzzy controller, based upon the five sensors readings at each instant. Once the new car situation has been calculated the simulator finds out whether it crashed into any map fence or not. All significant parameters can be changed from the user interface, not being necessary any code rearranging whatsoever.

3.2.3 Fuzzy Control

Fuzzy control in the simulation is realized by FICICDLL, a dynamic link library developed to meet the necessities derived from simulation of FICIC hardware. This dll is in charge of all the processes that involve simulating the working method and capabilities hosted in FICIC hardware.

As a preliminary step, a fuzzy rule base must have been developed using the FUZZYSOFT editor. This rule base must meet several conditions that ensure its viability with the car hardware.

Five inference strategies have been developed to be used in the simulator and in the real car: centre, left,right, park and stop.

As a result, we obtain a file containing the rules ready to be interpreted by the FICIC hardware or the FICICDLL software.

Instruction Code	Observations
0000.XXXX.XXXX.XXXX	No operation
0001.DBBB.XXXV.VVVV Add16b	Jump to Add16b if the bit BBB of variable VVVVV is equal to D
0010.XXXX.XXXX.XXXX Add16b	Unconditional jump to absolute address Add16b
0011.AAAA.AXXX.XXXX	End of actuator AAAAA. (to defuzzyfication)
0100.CCCC.CUXX.XXXX	THEN consequent CCCCC. (Last consequent if U =1)
0101.XXXX.XXXX.XXXX	End of the program. Wait for a new START.
1ZZZ.ZZYY.YYYV.VVVV	Subpremise (IF variable VVVVV IS AS_i, or AZ_i)

Table 1: Instructions Set.

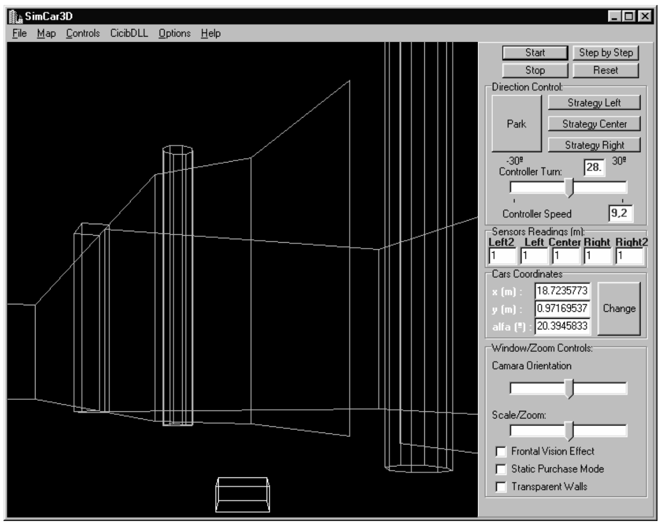
This rule base can have thousands of rules inside it, and run-time execution of the program could be a problem since FICIC hardware is perfectly capable of executing it in real time, but a software application running in a computer doesn't. For that reason the FICICDLL.dll functions execute a pre-interpretation of the code to free run-time execution from most of its workload. In this way the difference in speed between hardware execution and the simulation (software) is lessened considerably. Rule bases of up to 10,000 code lines have been tested without any problem. The rule base must be executed entirely on every simulation step (one per simulation clock period).

The instruction set the simulator uses features that supported by the hardware, and the instruction execution is exactly the same. Therefore the same RISC orientation has been used. The instructions are described in table 1.

Communication between the car simulator and the FICICDLL controller is realized exactly the same way it is done on the hardware implementation: through the use of two memories (IN and OUT memories) were variables discrimination is bound to the addresses they have in those memories (well-known addresses).

3.2.4 Graphic Interface

Trough a complete graphic interfaces, the user can have access to all the simulation parameters and representation modes. This includes: loading a rule base, loading and editing maps, controlling simulation execution, different types of simulation, car length, maximum car speed, wheel's maximum angle, sensor's orientation an properties, scale (Zoom), 2D and 3D viewer options, etc.
The simulator is a user-friendly application, which includes all the tools necessary for the user. It provides not only full access to all the simulation parameters, rule base and map edition, but also full view of the situation of the car, sensors readings, decisions taken by the controller, historic files with all data acquired during the simulation (very useful feature for neural nets rule base optimization).
Two-dimensional and three-dimensional views, with many different points of view and camera effects, provide complete information of the car motion.



-30º Controller Turn: 28. 30º
Controller Speed 9,2
Sensors Readings [m]:
Left2 1 Left 1 Center 1 Right 1 Right2 1
Cars Coordinates
x (m) : 18.7235773
y (m) : 0.97169537
alfa (º) : 20.3945833
☐ Frontal Vision Effect
☐ Static Purchase Mode
☐ Transparent Walls

Figure 6. Three dimensional representation and simulator controls

4 CONCLUSIONS

A 7,300 lines code rule base has been developed using the FUZZYSOFT and it has been successfully tested by the simulator. This rule base has been tested on many different scenarios and many different obstacles have been saved by the car. Also important information about the orientation and proper use of the ultrasonic sensors has been acquired, as well as maximum car speed, control frequency, etc.
This includes the five strategies described before: follow left wall, follow right wall, center in the road, park and stop. These strategies can be chosen by the simulator user by means of a set of buttons that permits to alternate from one strategy to another. In the physical model, this is achieved through the radio controls.
The rule bases developed using the simulator have been loaded into the physical model program memory so the FICIC hardware could control the car. In that way, several trials have been done alternating different strategies, rule bases and driving circuits. The results have been parallel to those obtained in the simulation, and rule bases obtained on the computer model have proofed to be successful on the real scenarios. Very few differences have been noticed, mostly due to wheels sliding and terrain irregularities. Real time execution of the final 7,300 fuzzy rule base code has been realized with a 12 MHz clock on the control process.

Acknowledgement

This work has been sponsored by the Universidad del País Vasco/Euskal Herriko Unibertsitatea with the code: UPV 147.345-TB058/96.

References

[1] C. Von Altrock, B. Krause and H. J. Zimmermann, "Advance Fuzzy Logic Control of a Model Car in Extreme Situations", Fuzzy Sets and Systems, vol. 48, nº 1, pp. 41-52, 1992.
[2] H. Eichfeld, M. Klimke, M. Menke, J. Nolles and T. Künemund, "A General-Purpose Fuzzy Inference Processor", IEEE Micro mag., vol. 15, nº 3, 1995.
[3] H. Surmann and A. P. Ungering, "Fuzzy Rule-based systems on General-purpose Processors", IEEE Micro Mag., vol. 15, nº 4, pp. 40-48, 1995.
[4] G. Aranguren, M. Barron, J. López de Arroyabe and A. Chavarría, "Controlador borroso implementado en FPGA", ESTYLF'98, Pamplona, pp. 425-430, 1998.
[5] H. Watanabe, W. D. Dettloff and K. E. Yount, "A VLSI Fuzzy Logic Controller with Reconfigurable, Cascadable, Architecture", IEEE J. Solid State Circ., vol. 25, nº 2, pp. 376-381, 1990.
[6] V. Samoladas, L. Petrou, "Special-purpose Architectures for Fuzzy Logic Controllers", Microprocessing and Microprogramming, vol. 40, nº 4, pp. 275-289, 1994.
[7] D. L. Hung, "Dedicated Digital Fuzzy Hardware", IEEE Micro Mag., vol. 15, nº 4, pp. 31-39, 1995.
[8] G. Aranguren, M. Rodríguez y M. Barrón, "Controlador Secuencial Segmentado para Lógica Borrosa Implementado en FPGA", ESTYLF'96, Oviedo, pp. 255-259, 1996.
[9] G. Aranguren, M. Rodríguez, J. López de Arroyabe and N. Altarriba, "Software de compilación, simulador y emulación borrosos", ESTYLF'97, Tarragona, pp. 77-82, 1997.