Neuro-Fuzzy Algorithm implemented in Altera's FPGA for Mobile Robot's Obstacle Avoidance Mission

Muhammad Nasiruddin Mahyuddin, Chan Zhi Wei and Mohd Rizal Arshad
School of Electrical and Electronic Engineering,
Universiti Sains Malaysia
14300, Nibong Tebal,
Penang, Malaysia.
Email: nasiruddin@eng.usm.my, trenoshinn@gmail.com and rizal@eng.usm.my
http://urrg.eng.usm.my

Abstract*— This paper presents the designed obstacle avoidance program for mobile robot that incorporates a neuro-fuzzy algorithm using Altera™ Field Programmable Gate Array (FPGA) development DE2 board. The neuro-fuzzy-based-obstacle avoidance program is simulated and implemented on the hardware system using Altera Quartus® II design software, System-on-programmable-chip (SOPC) Builder, Nios® II Integrated Design Environment (IDE) software, and FPGA development and education board (DE2). Nios® II IDE software is used to simulate and process the weight training. A mobile robot serves as the test platform of the program. An ultrasonic sensor and 3 servo motors are used as the test platform's sensing element and actuator, respectively with 2 servo motors used to move the robot and single servo motor used to adjust the sensing direction of the ultrasonic sensor. Altera™ FPGA development board proves to be a successful embedded system platform for the neuro-fuzzy algorithm to be implemented and tested. Remote testing is done on mobile robot for obstacle avoidance missions via an established radio frequency (RF) communication using DIGI™ XBee RF module.

Keywords-Neuro-fuzzy; FPGA; Mobile robot; DE II Board;

I. INTRODUCTION

The ability to easily reconfigure Field Programmable Gate Array (FPGA) makes the design less expensive than pre-designed hardware. FPGA provides suitable platform in realizing complex hardware system as well as implementing data intensive algorithm computation like sensing obstacle surrounding the environment of interest. These features bring convenience to incorporating an artificial intelligence-based-program for mobile robot navigation and obstacle avoidance task or mission. The general theory for mobile robotics navigation is based on a simple premise. For mobile robot to operate it must sense the known world, be able to plan its operations and then act based on this model. In real world, construction of mathematical models (white-box models) is very difficult because we rarely have a complete knowledge of the process that we need to model [1]. This theory of operation has become known as SMPA (Sense, Map, Plan, and Act) [2].

In our research, neuro-fuzzy algorithm is implemented as means to control the SMPA operation because the algorithm does not need a mathematical model of the real world.

According to research done by Al-Gallaf [5], he successfully implement the neuro-fuzzy system to drive a mobile robot run on a Transputer system. The Transputer ability, as an embedded controller, to perform a demanding parallel fuzzy controller, has been demonstrated in the real-time. This research shows that a parallel hardware architecture of Transputer able to interact with the neuro-fuzzy system in which we will able to implement the system on FPGA due to its higher speed and parallelism characteristic.

In other literatures, other researchers like Phinni [6], uses a genetic-neuro-fuzzy approach to deal with the obstacle avoidance problem. He proposes to generate collision free path and have proper motion planning as well as obstacle avoidance scheme. In the developed neuro-fuzzy approaches, training is done off-line with the help of Genetic Algorithm (GA), and the result shows that the system able to extract automatically the fuzzy rules and the membership functions in order to guide a wheeled mobile robot. The membership function for distance is generated by using Takagi and Sugeno Approach. His results seem to lack in hardware implementation and verification since he only produces simulation results.

Based on the research done by Pradhan [7], Rule-based-Neuro-Fuzzy Technique is used for navigation of multiple robots by simulation. Simulation result obtained shows that as many as one thousand mobile robots can successfully navigate neither colliding with each other nor colliding with obstacles present in the environment. They successfully shows that rule-based-neuro-fuzzy technique performs much better compared to the rule-based technique for navigation of multiple mobile robots where rule-based technique does not have learning capabilities.

Another example of related research is done by Cao [2] on Reactive Navigation for Autonomous Guided Vehicle Using the Neuro-Fuzzy Techniques. In this research, neuro-fuzzy algorithm is used as the system that is able to control the operation such as SMPA (Sense, Map, Plan, and Act). Neuro-

This work was supported by Universiti Sains Malaysia, under Short Term Grant 304/PELECT/6035251

fuzzy system is able to modify and extract new rules from a properly training. It offers the precision and learning capabilities which are suitable for obstacle avoidance task.

The first successful implementation of artificial neural network was published over a decade ago [8]. Upegui and Sanchez [9] describe that FPGA are very well suited for evolving systems with cellular structures, such as neural networks, fuzzy system rules, or cellular automata because it allows a great degree of flexibility for evolving circuits. Cavuslu [10] has demonstrated the hardware implementation of neural network using Xilinx FPGAs, an Intel based FPGA architecture. They show that hardware implementation of fully parallel ANN's is possible in real time using general FPGAs features. Other work shows the implementation of artificial neural network in FPGA for insect robots [11].

Referring to Fig.1, the neuro-fuzzy algorithm used in our research is the cooperative neuro-fuzzy system, which comprises the fuzzy sets (i.e. membership functions) of the fuzzy system, predetermined by the neural network [12]. The neural network mechanism (embedded inside the Altera™ FPGA) initially acquires the sampled training data relayed from the remote mobile robot via DIGI™ XBee RF modules. Both the neural network and fuzzy systems used complement each other and provide improved real-time control and navigation performance.

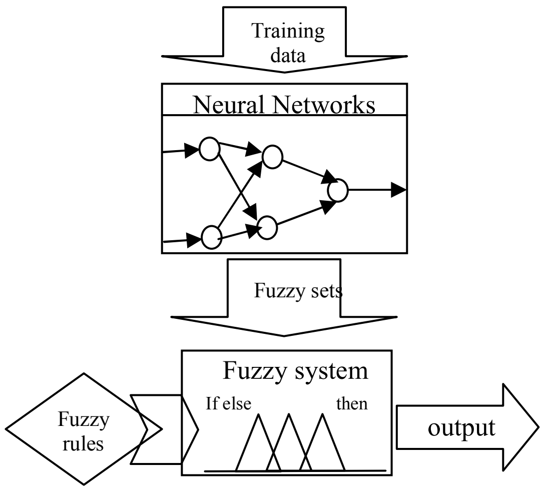


Figure 1. Neuro-fuzzy algorithm used features membership functions of the fuzzy system that are predetermined by the neural network.

The designed neuro-fuzzy algorithm in obstacle avoidance program is realized using Altera FPGA DE2 development board with Cyclone II chipset. The board allows high-speed hardware applications, provides a high degree of parallelism [3] and offers acceptable densities without the cost and length design cycles of full custom circuits [4].

The integration between hardware and software system design for neuro-fuzzy algorithm will be carried out using Altera Quartus® II and Nios® II IDE software. This paper presents the simulation results as well as the results of the obstacle avoidance experiment to assess the performance of the mobile robot in various scenarios of static hindrance.

II. Formulation of Neuro-Fuzzy Algorithm For Obstacle Avoidance Program

The neuro-fuzzy algorithm is formulated in four layers as shown in Fig. 2. The layers are:

1) Input layer
2) Rule layer
3) Consequent layer
4) Output layer

Input Layer

In this layer, the inputs are fuzzified. The inputs are the distances sensed (by ultrasonic sensor) between the obstacle and the mobile robot. The system requires 3 types of input:

1) Left direction obstacle distance
2) Middle direction obstacle distance
3) Right direction obstacle distance

Each of these inputs classified to the fuzzy set memberships function as shown in Fig. 3.

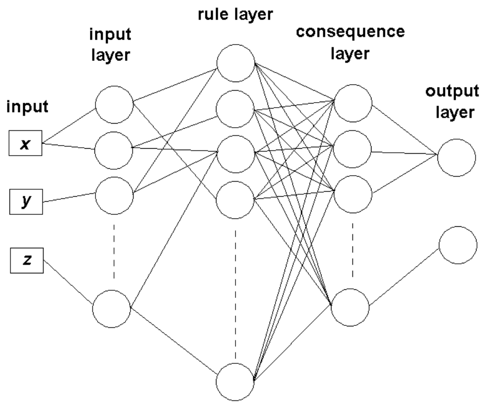


Figure 2. Structure of Designed Neuro-fuzzy

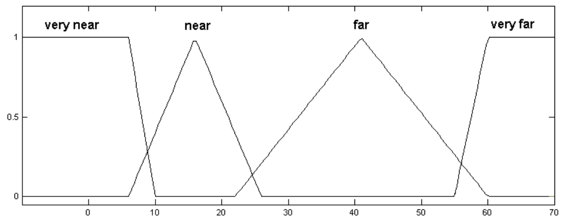


Figure 3. Input Membership Function

The fuzzy set for direction distance is tabulated as in Table 1, shown on the next page.

Rule Layer

Each node in this layer calculates the firing strength of a rule via multiplication:

$$r_i = \mu_{Ai}(x) \times \mu_{Bi}(y) \times \mu_{Ci}(z) \quad i=1,2,3. \tag{1}$$

TABLE I
DIRECTION DISTANCE FUZZY SET

Fuzzy Set	Description	Variation Range
LVN	left very near	<10
LN	left near	6~26
LF	left far	22~60
LVF	left very far	>55
MVN	mid very near	<10
MN	mid near	6~26
MF	mid far	22~60
MVF	mid very far	>55
RVN	right very near	<10
RN	right near	6~26
RF	right far	22~60
RVF	right very far	>55

In this layer, the firing strength depends on the condition given by the fuzzy rule. If the antecedent of the rule consists only the input x and input y, input z will not be taken into consideration of firing strength. Each neuron represents the fuzzy rule of the system. 40 neurons are utilized in the Rule Layer, giving rise to 40 rules embedded inside the Fuzzy Knowledge Base System.

Consequence Layer

There are 6 nodes in consequence layer which specify the direction of the mobile robot to be executed. Node in this layer computes the sum of all firing strengths of the rules using feed-forward neural network algorithm as shown in Fig. 4.

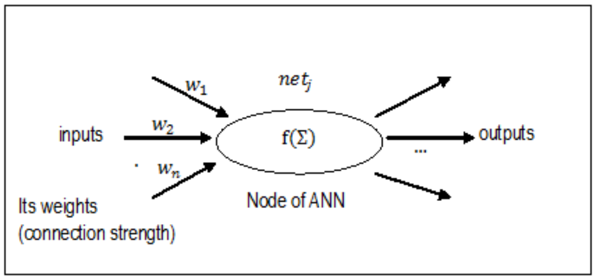


Figure 4. Feed-forward Neural Network Structure [13]

Each output from the Rule Layer is multiplied with the weight:

$$i_i = r_i \times w_i \qquad (2)$$

where i_i is the result of multiplying the output from rules layer, r_i, and the weight, w_i.

It is eventually fed into the consequence layer where activation process of neural network using sigmoid function takes place.

$$f_{Ai}(i_i)=sigmoid[\textstyle\sum_i i_i] \qquad (3)$$

where $f_{Ai}(i_i)$ is the output for the consequence layer.

Output Layer

Defuzzification takes place in this layer by using Largest of Maximum (LOM) method which is suitable for decision making task.

There are two nodes in this layer because only two outputs are needed. One node represents the direction of moving (forward, stop, and reverse) and the other node represents the angle of deviation or the desired heading angle of the mobile robot (left, middle, and right).

The node will determine the maximum value from the output of the consequence layer and determine the final direction that will be taking placed.

Back-propagation

Back-propagation is used to feedback the error signal in the system through one layer by another, updating the weights of the neuro-fuzzy system.

The method for weight training is given as follows [13]:

Calculate the error gradient for the neuron in the output layer:

$$e_k = d_k - y_k \qquad (4)$$

$$\delta_k(i) = y_k \times [1 - y_k] \times e_k(i) \qquad (5)$$

where,
e_k = the error calculated
d_k = desired output
y_k = actual output
$\delta_k(i)$ = error gradient
The calculated weight corrections:

$$\Delta w_{jk}(i) = \alpha \times \delta_k(i) \times y_k \qquad (6)$$

where α is the learning rate which have the value between 0 and 1.

The updated weight for consequence layer:

$$w_{jk}(i) = w_{jk}(i) + \Delta w_{jk}(i) \qquad (7)$$

III. FPGA HARDWARE SYSTEM DESIGN

Quartus® II and SOPC Builder are used for the hardware system design. SOPC Builder is used to define the structure of a hardware system by generating them in Verilog HDL files. The GUIs featured in SOPC Builder allows the component to be added or removed conveniently. The component is configurable and the connection between the components can be easily modified.

The designed hardware system consists of CPU, memory and I/O peripherals. In this design, we require additional JTAG UART interface (which is defined in the SOPC Builder) to be able to connect the DE2 board to the host computer via a Universal Serial Bus (USB). The block diagram for the system generation design is shown in Fig. 5.

In order to implement the neuro-fuzzy algorithm into FPGA board, software programming is required where the neuro-fuzzy algorithm is written in C++ program which can be compiled and simulated by using Nios® II IDE software.

Fig. 6 shows the flow chart of neuro-fuzzy algorithm training which can be programmed using Nios® II IDE software.

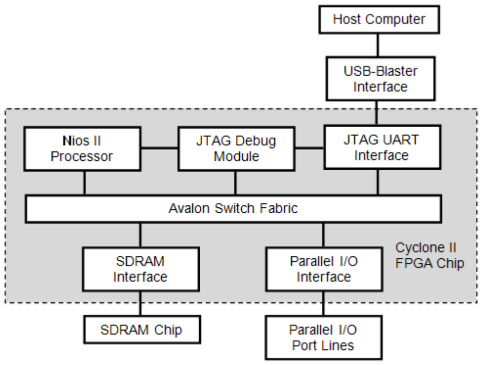
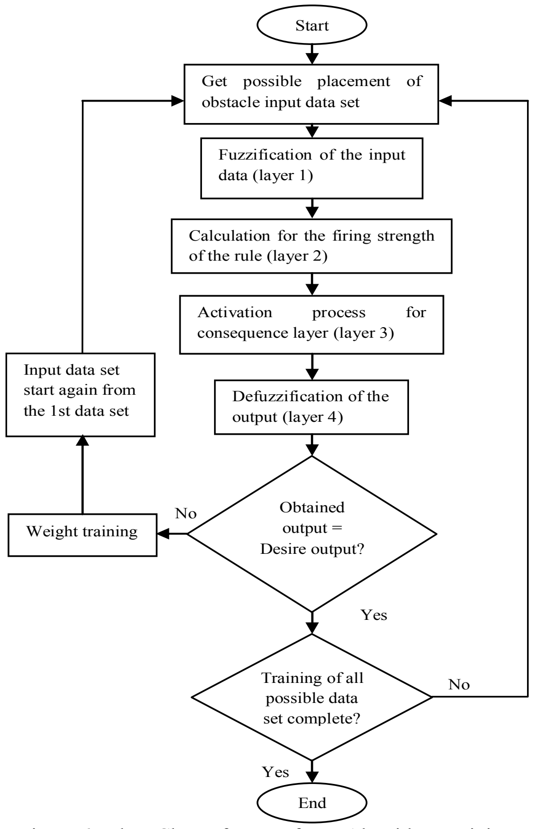


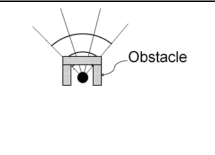
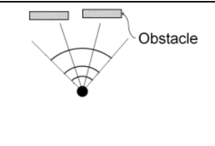
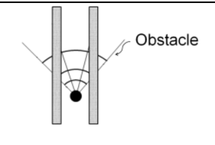
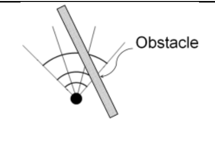
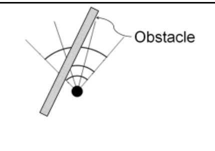
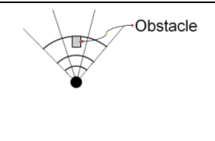
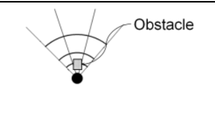
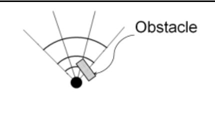
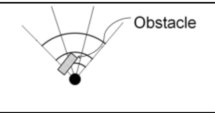
Figure 5. Designed Hardware Systems



Start
Get possible placement of obstacle input data set
Fuzzification of the input data (layer 1)
Calculation for the firing strength of the rule (layer 2)
Activation process for consequence layer (layer 3)
Defuzzification of the output (layer 4)
Obtained output = Desire output?
No
Weight training
Input data set start again from the 1st data set
Yes
Training of all possible data set complete?
No
Yes
End

Figure 6. Flow Chart of Neuro-fuzzy Algorithm Training

TABLE 2
SIMULATION OF NEURO-FUZZY ALGORITHM

No.	Situation	Result of Simulation
1		left sensor input =6.000000 middle sensor input =15.000000 right sensor input =7.000000 direction = reverse angle = mid
2		left sensor input =200.000000 middle sensor input =99.000000 right sensor input =80.000000 direction = forward angle = mid
3		left sensor input =12.000000 middle sensor input =300.000000 right sensor input =13.000000 direction = forward angle = mid
4		left sensor input =300.000000 middle sensor input =40.000000 right sensor input =20.000000 direction = forward angle = left
5		left sensor input =12.000000 middle sensor input =35.000000 right sensor input =300.000000 direction = forward angle = right
6		left sensor input =300.000000 middle sensor input =35.000000 right sensor input =300.000000 direction = forward angle = right
7		left sensor input =300.000000 middle sensor input =6.000000 right sensor input =300.000000 direction = reverse angle = right
8		left sensor input =300.000000 middle sensor input =12.000000 right sensor input =6.000000 direction = stop angle = left
9		left sensor input =8.000000 middle sensor input =15.000000 right sensor input =300.000000 direction = stop angle = right

IV. RESULT AND DISCUSSION

A random set of data with different situation is applied to test the output result. The simulation results are shown in Table 2.

Situation 1

The mobile robot is placed in a deadlock situation. From the simulation result shown in Table 2, the mobile robot tries to escape the dead lock by moving in reverse direction.

Situation 2

The obstacles are placed very far (or no obstacle) from mobile robot. The result shows that the mobile robot still moving in a straight direction.

Situation 3

Two walls of obstacles are placed at left and right direction of the mobile robot. No obstacle at the middle path. Result shows the mobile robot continues to move forward and following the wall.

Situation 4

In this case, the obstacle is placed on the right side of the mobile robot. The robot tries to avoid it by move to left direction.

Situation 5

Similar to case 4, the robot tries to avoid the obstacle by move to right direction.

Situation 6

Now, a square obstacle is placed at the middle direction. The result successfully shows that the mobile robot does not move in the middle path that may cause collision with the obstacle.

Situation 7

A square obstacle is placed very near to the middle path of the mobile robot. Result shows that the robot tries to avoid the obstacle by move in right and reverse direction. This shows that the mobile will always keep a long distance with the obstacle to avoid collision.

Situation 8

The obstacle is placed very close to robot in the right direction. The robot stopped and turns in left direction.

Situation 9

Similar to case 9 but with obstacle placed at the left direction. The robot tries to avoid collision by stop and turn in right direction.

From the result obtained, it is observed that the mobile try to avoid the obstacle sensed and finding a collision-free path. The mobile robot is able to avoid the given obstacle situation.

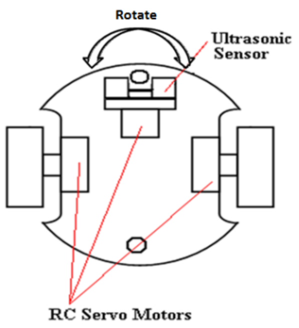


Figure 7. Top View of Mobile Robot Platform

Fig. 7 shows the simple mobile robot platform that used for the obstacle avoidance task. The communication between the mobile robot and the Altera DE2 FGPA Development Board is shown in Fig. 8. Fig. 9 shows the experimental mobile robot used to test the program.

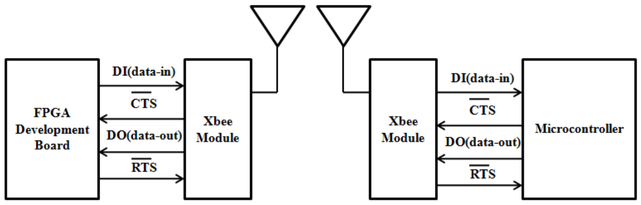


Figure 8. Communication Between FPGA and Mobile Robot

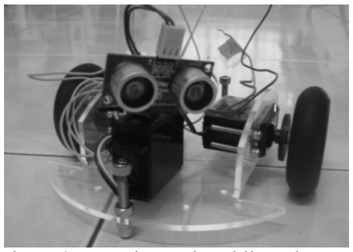


Figure 9. Experimental Mobile Robot

Example calculation for situation 4:

Using Trapezium equation:

$$\mu_{Ai}(x_i) = \begin{cases} 0, & x < a \text{ or } x > d \\ \frac{x-a}{b-a}, & a < x < b \\ 1, & b < x < c \\ \frac{d-x}{d-c}, & c < x < d \end{cases} \quad (8)$$

A Trapezium membership function generated as shown in Fig. 10.

Using Triangle equation:

$$\mu_{Ai}(x_i) = \begin{cases} 0, & x < a \text{ or } x > c \\ \frac{x-a}{b-a}, & a < x < b \\ \frac{d-x}{d-c}, & b < x < c \end{cases} \quad (9)$$

A Triangle membership function generated as shown in Fig. 11.

Calculation:

For left direction input:
Left sensor input = 300 cm

Calculation using equation 8:

$$\mu_{A1}(300) = 1$$

The fuzzy input for left sensor is equal to 1 for very far position.

For middle sensor input:
Middle sensor input = 40 cm

Using equation 9, a membership function generated as shown in Fig. 11.

$$\mu_{B2}(40) = \frac{40-22}{41-22} = 0.947$$

The fuzzy input for middle sensor is equal to 0.947 for far position.

For right sensor input:
Right sensor input = 20 cm

Using equation 9, a membership function generated as shown in Figure 12.

$$\mu_{C1}(20) = \frac{20-6}{26-6} = 0.7$$

The fuzzy input for right sensor is equal to 0.7 for 'near position'

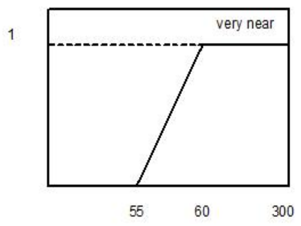


Figure. 10. Membership Function for Left Sensor Input

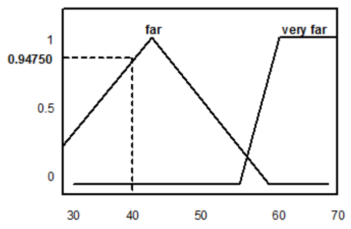


Figure. 11. Membership Function for Middle Sensor Input

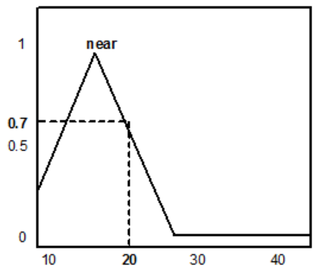


Figure. 12. Membership Function for Right Sensor Input

After calculating the fuzzy input, the fuzzy input are matched with the rule in the rule layer by using AND operation (equation 2).

In the consequence layer, activation process (equation 3) takes place using the weighted outputs from rule layer as input.

At the final stage, defuzzification is performed on the data by using Largest of Maximum (LOM) method. The result of motion given is making the mobile robot to move in reverse and in middle direction.

V. Conclusion

The simulation result of motion planning for the mobile robot is compared and verified through hardware implementation on FPGA. The neuro-fuzzy algorithm-based-obstacle avoidance program is successfully tested on the experimental mobile robot.

VI. Acknowledgment

The authors would like to thank Universiti Sains Malaysia for the awarded Short Term Grant 304/PELECT/6035251 to support the research project on obstacle avoidance of an intelligent mobile robot. The authors would like to express gratitude and thanks to Altera Corporation (M) Sdn. Bhd. for their donation of the FPGA Development Board (DE2) to School of Electrical and Electronic Engineering, USM and also their support in terms of trainings and advices.

VII. References

[1] M. A. Grima, Neuro-*fuzzy Modeling In Engineering Geology,* Balkema, 2000, p.1, 53 – 59.
[2] J. Cao, X.Q. Liao, E. Hall, *Reactive Navigation for Autonomous Guided Vehicle Using the Neuro-fuzzy Techniques,* Available: www.min.uc.edu/robotics/papers/ paper1999/spiejin/jin99.pdf 16/01/09.
[3] R.H. Katz, G. Borriello, *Contemporary Logic Design*, 2nd ed., Pearson Prentice Hall, Upper Saddle River, NJ, 2005 p. 421-451.
[4] N.M. Botros, M. Abdul-Aziz, *Hardware implementation of an artificial neural network*, IEEE International Conference on Neural Network, Vol. 3, 1993 p.1252 – 1257
[5] E. Al-Gallaf, *Transputer Neuro-Fuzzy Controlled Behavior-Based Mobile Robotics System*, Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1 .1.91.4667&rep=rep1&type=pdf 05/02/09
[6] M.J. Phinni, A.P. Sudheer, M. RamaKrishna, K.K. Jemshid, *Obstacle Avoidance of a wheeled mobile robot: A Genetic-neurofuzzy approach*, IISc Centenary – International Conference on Advances in Mechanical Engineering (IC-ICAME), Bangalore, India, July 2-4, 2008
[7] S.K. Pradhan, D.R. Parhi, A.K. Panda, *Navigation of Multiple Mobile Robots using Rule-based-Neuro-Fuzzy Technique*, International Journal of Computational Intelligence 3, 2007
[8] J. Zhu, P. Sutton, *FPGA Implementations of Neural Networks – a Survey of a Decade of Progress*, Vol 2778, 2003 p.1062-1066
[9] A. Upegui, E. Sanchez, *Evolving Hardware by Dynamically Reconfiguring Xilinx FPGAs*, Available: http:// slwww.epfl.ch/~upegui/docs/Upegui_ICES05.pdf 16/02/09
[10] M.A. Çavuşlu, C. Karakuzu, S. Şahin, *Neural Network Hardware Implementation Using FPGA*, ISEECE 3rd International Symposium on Electrical, Electronic and Computer Engineering Symposium Proceedings, ISEECE 2006 3rd International Symposium on Electrical, Electronic and Computer Engineering, 2006 p. 287-290
[11] R.J. Mitchell, D.A. Keating, C. Kambhampati, *Neural Network Controller For Mobile Robot Insect*, Available: http://www.personal.rdg.ac.uk/~shsmchlr/misc file/e94www.pdf 09/02/09.
[12] D. Nauck, F. Klawonn and R. Kruse, *Foundations of Neuro-Fuzzy Systems*, West Sussex: John Wiley, 1997.
[13] A. Garrido, *Fusion Between Fuzzy Systems, Generic Algorithms and Artificial Neural Networks*, ACTA Universitatis Apulensis, No. 12, 2006
[14] M. Negnevitsky, *Artificial intelligence*, 2nd edi., Addison-Wesley, 2005 p.166-217, 260-299.