Implementations Approches of Neural Networks Lane Following System

Klabi Imen
METS Research Group-National Engineers School of Sfax, Tunisia

Afef Benjemma
METS Research Group-National Engineers School of Sfax, Tunisia

Mohamed Slim Masmoudi
METS Research Group-National Engineers School of Sfax, Tunisia

ABSTRACT
Nowadays, the techniques based on the use of artificial neural networks are instigating increasing interest in the fields of control and robotics. The rapidity of processing, the ability to learn and adapt as well as the robustness of these approaches, are motivating this work.

To help this system be embedded in a wheelchair, it is imperative to respect the functional constraints and those of resource allocation, weights, consumption, cost...

So conceiving an embedded system is ultimately an exercise in optimization: minimizing production costs for optimal functionality. The objective of this work is FPGA implementation of an optimal architecture of neuronal network.

Keywords:
Robotic Mobile, neural networks, FPGA, sigmoid function

1. INTRODUCTION
Intelligent algorithms [2-3] which have parallel data flow in her nature have the ability to solve complex problems on the areas, such as modeling, control, image processing, guidance/assistance and medical diagnosis [4-10].. The most used algorithms are neural network, fuzzy logic and genetic algorithm. Real-time applications which employ ANN (Artificial Neural Network) requiring faster response time can show their real performance only on parallel running architectures. To take profit from the parallelism of neural networks, we had thought about new ways of implementation different from usual ones and based on the use of standard processors. A suitable and adequate alternative consists in using dedicated hardware systems operating different means of achieving hardware and software.

This strategy is based on both the advent of submicron technologies (FPGA and ASIC) [5] and the development of conception (methodology and CAD tools).

In fact, these tools have many advantages: a beneficial hardware implementation in terms of area and execution time, affordable business costs, a satisfying response of functional and productive requirements and short deadline of conception. The purpose of this article is to conceive an optimal architecture of neuronal network in order to implements it on FPGA. The paper is organized as follows:

Section 2 describes lane following system. Section 3 descried the design of the neural network and simulation results obtained by Matlab Toolbox. Section 4 presents the simulation and implementation approaches of the neural network system in FPGA.

2. DESCRIPTION OF THE LANE FOLLOWING
Our robot has a same configuration as normal car. Five infrared detectors which are used to find the distance between the vehicle and its obstacles (see Figure1). Three sensors were placed at the right side of the vehicle facing the parking area and two detectors were installed on the front and rear of the vehicle.

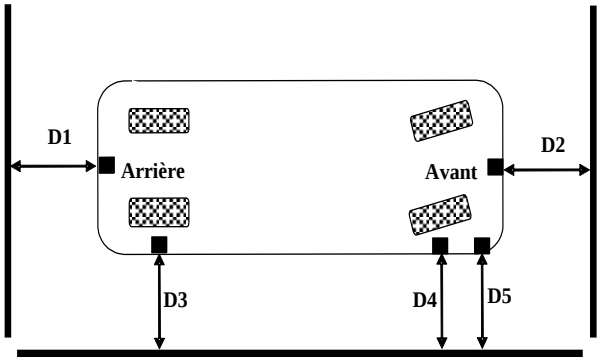

Figure 1: Positioning of the various detectors

3. NEURAL NETWORK AND SIMULATIONS
Several simulations were performed in Matlab to find the optimal architecture for our network of neurons.
To guarantee efficiency, time and the minimum resources on the FPGA, we propose a network that contains two hidden layers: the first layer has ten neurons, the second seven neurons, and the output layer has one single neuron.
We can analyze realization of MLP in two parts. Learning stage consists of updating network parameters defined by learning algorithm. Test and generalization stage composes that network is tested with different data by using same parameters which are achieved at the end of the training.
The network's input and output data training sets are normalized to [0.05, 0.95], this improve effectiveness regarding to the domain definition of the sigmoid function. The weights will be adjusted to minimize the error between the desired output and the network output. The simulation results using MATLAB gives a smallest error for 5000 epochs. We have conducted a learning machine using the MATLAB software where we get the smallest square error (1.8126e-004) See Figure2.

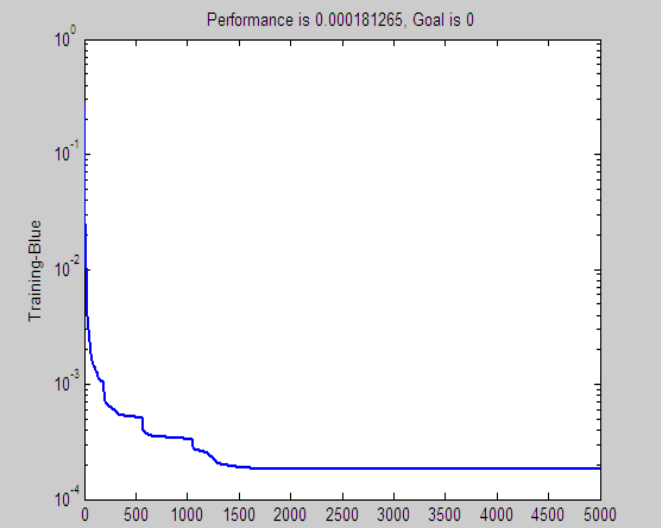

Figure 2. Quadrature error curve during training

This error is relatively low (see Figure 2).

4. FPGA IMPLEMENTATION

To reduce the computation time and increase accuracy, we work in fixed-point code for the various parameters of the network given the simplicity of implementation and low cost. The VHDL implementation of neural network algorithm is done by two steps: the first step is the implementation of the neurone function; the second step is the implementation of the network architecture design.

4.1 Fixed points coding

In this case implementation, we chose the natural binary coding for positive numbers between 0 and 2n-1 and 2's complement coding for negative numbers. The principle of the fixed-point coding is given in Figure 3.

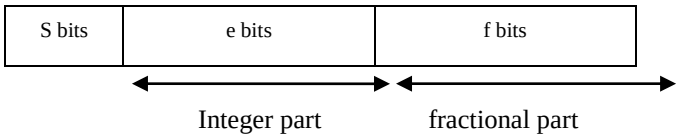

Figure 3. Format of a fixed point number.

To increase the precision of a real number N, N is multiplied by a constant 2^f with f the number of bits of the fractional part. So the number of bit coding of N will be: N=e+f+1.The estimated number is given by this equation:

$$N_{est} = round\left(N \times 2^f\right) / 2^f \qquad (1)$$

The round function is used in Matlab; it gives the rounding of a real number. These estimated parameters can experience errors that may affect the accuracy of the calculation.

4.2 Sigmoïde function coding

The simulation in digital technology of the activation function which is obtained from the basic formula:

$$f(x) = \frac{1}{1+e^{-x}} \qquad (2)$$

requires a relatively large computation time compared to the time required for the multiplier and adder. To reduce this computation time, three methods are often cited:

1. The use of a calculation unit specific to the evaluation of the function activation
2. polynomial methods
3. Coding of the function in an array of values ("lookup table").

In the first time of this work, we use the third solution

4.2.1 Look up table approach

In order to increase accuracy of coding the sigmoid function values, we took only the values of X (the weighted sum of output) belonging to the interval [-7.7] because the value of the sigmoid function Y is equal to 1 for values greater than 7, and it is less than 10^{-3} for values less than -7.

X addresses was multiplied by 2^{10}, so the maximum address (corresponding to X = 7) is equal to 7 * 1024 = 7168, so the addresses will be coded on 14 bits. We chose a memory size of 16384 memory cells in order to contain the positive and negative values of the addresses of X.

The quantization error of sigmoid function is 0.0052%. The error is low due to the bits number coding chose. To improve accuracy, the outputs Y are multiplied by 2^{13}. The values of Y will be coded on 14 bits. The values of Y estimated the sigmoid function is given by the following equation:

$$Y_{iest} = round\left(Y_{iest} \times 2^{13}\right) / 2^{13} \qquad (3)$$

The quantization error is given by the following relation

$$E_{rr} = norm\left(Y - Y_{est}\right) = \sqrt{\sum_{i=1}^{i=16383} \left(\left|Y_i^2 - Y_{iest}^2\right|\right)} \qquad (4)$$

So, to an accuracy of 13 bits of the fractional part there is the quantization error of the sigmoid function is 0.0013.

In order to use the designed ROM, we reduce the output bits vector using truncate and division block. As shown in Figure 6 the final block diagram of neural network blocks is composed of multipliers, adder, divider, truncate and ROM modules.

4.2.2 Simulation results

The simulation of the VHDL model is performed using Quartus II version 9.

Our sigmoid function (see figure 4) has been compiled by QUARTUS II and implemented on a CycloneII EP2C35F484C6 FPGA.

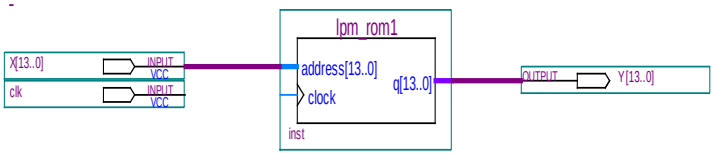

Figure 4. Implementation of sigmoid function implemented with Look up table approach

The results of the compilation process are described in table 1

Table 1. Hardware resources of the sigmoid function implemented with *Look up table approach*

Total logic elements	Total memory bits
34/33.216(1%)	229.376/483.840(47%)

The results of implementation of our neuronal networks with sigmoid function developed with this solution are described in table 2.

Table 2. Hardware resources of our neuronal networks

Total logic elements	Total memory bits
11.234/85200(13%)	4.128.768/8.248.320(50%)

The disadvantage of this solution comes from the area occupied by memory in FPGA.To remedy the problems of this solution, we thought about using the polynomial methods to approximate the sigmoid function. There are several methods of approximation, but the one used is the method in the sense of least squares.

4.2.3 Polynomial approach

In purpose to choose the smallest order of approximation polynomial, that allows obtaining the closest polynomial to the function f at some points xi, several simulations were carried out under Matlab. Figure 5 shows an approximation by a polynomial of order 6 and the other by a polynomial of order 7.

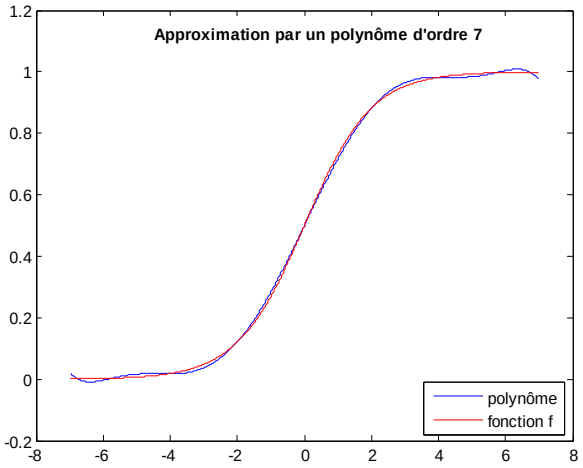

Figure 5.Approximation the sigmoid function by a polynomial of order 6 and 7.

We note here that the polynomial of order 7 gives a better approximation of the sigmoid.
Thus, the polynomial approximation obtained is:

$$P = a_1 \times x^7 + a_2 \times x^6 + a_3 \times x^5 + a_4 \times x^4 + a_5 \times x^3 + a_6 \times x^2 + a_7 \times x + a_8 \quad (5)$$

By eliminating insignificant coefficients, the polynomial becomes:

$$P = a_1 \times x^7 + a_3 \times x^5 + a_5 \times x^3 + a_7 \times x + a_8 \quad (6)$$

To increase the accuracy of calculation the coefficients a_i of polynomial P was multiplied by 2^{19}. Once we have obtained a polynomial approximating the function and whose coefficients are numbers that can be manipulated by machine, it remains to implement this polynomial. To do that, we have to choose an evaluation diagram that describes the order in which operations are conducted. The simplest diagram is the one of Horner. This diagram has good properties in terms of accuracy, but the sequential structure forbidden to exploit the parallelism exposed by the materiel[8]. That is why we thought about using Estrin's family of diagram which is some kind of Horner diagram parallelized [8].
Our polynomial of degree 7 is evaluated by the following diagram:

$$P = (a_8 + a_7 \times x) + x^2 \times (a_5 \times x) + x^4 \times [(a_3 \times x) + x^2 \times (a_1 \times x)] \quad (7)$$

This diagram can be run in parallel.
Every line contains the different expressions that can be run in parallel:

$$x2 = x \times x \quad P1 = (a_8 + a_7 \times x) \quad P2 = a_5 \times x \quad P3 = a_3 \times x$$
$$P4 = a_1 \times \; ; \quad ; \quad ; \quad ; \quad ; \quad ;$$
$$x4 = x2 \times x2 \quad P21 = P1 + x2 \times P2 \quad P22 = P3 + x2 \times P4$$
$$P = P21 + x4 \times P22$$

This solution (see figure 6) has been compiled by QUARTUS II and implemented on a Cyclone EP2C35F484C6 FPGA.

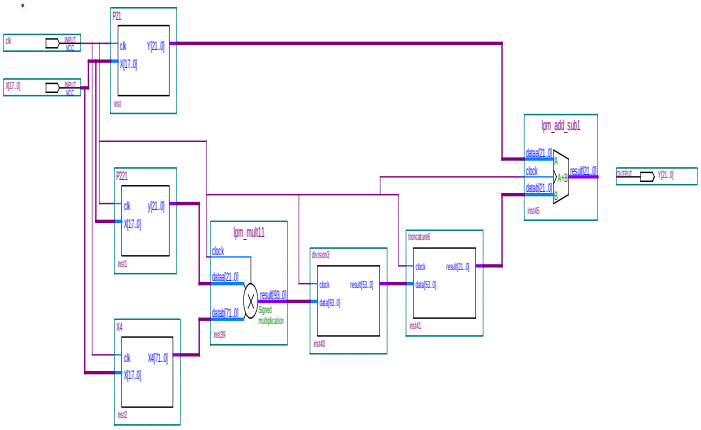

Figure 6. Implementation of sigmoid function approximated by a polynomial

Figure 7 presents the results of functional simulation of the sigmoid function approximated by a polynomial.

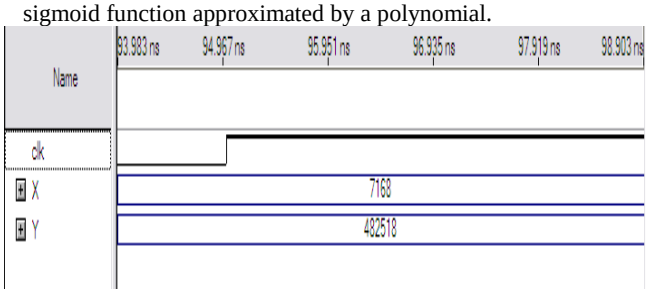

Figure 7. Simulation result of polynomial approximation of the sigmoid function

For x = 7, we obtain:

- The theorique result : y = 0,9761
- The simulation result : y = 0,9203($482518/2^{19}$)

The results of the compilation process are described in table 3.

Table 3. Result of compilation (synthesis) of the polynomial solution

Total logic elements	Memory Ressources
2.198/33.216(7%)	0/483.840(0%)

The results of implementation of our neuronal networks with sigmoid function approximated by a polynomial function are described in table 4.

Table 4. Hardware resources of our neuronal networks

Total logic elements	Memory Ressources
45.583/85.200(55%)	0/8.248.320(0%)

4.3 Comparaison

In this section we will compare the implementation results of our NNC developed

Table 5. Comparison

	Total logic elements	Memory ressource
NNC (look up table approach)	11.234/85.200 (13%)	4.128.768/8.248.320 (50%)
NNC(polynomial approach)	45.583/85.200(55%)	0/8.248.320 (0%)

The step of synthesis and implementation of our NNC has shown that with the first solution (use look up table approach) we have occupied more memory resources (50%) than with the second solution (use polynomial approach) developed (0%).

5. CONCLUSION

In This paper we have described the design and implementation of neural network control algorithm for a mobile robot car. We have chosen to work with programmable circuits and in particular to use FPGA.

To implement our NNC we have developed two solutions for sigmoid function optimization: look up table approach and polynomial approach. The step of synthesis and implementation shas shown that with the first solution we have occupied more memory resources (50%) than with the second solution developed (0%).

6. REFERENCES

[1] A. Abedenour"Outil d'analyse et de partitionnement/ ordonnancement pour les systèmes temps réel embarqués "Thèse de doctorat, l'université Bretagne sud 2004.

[2] Yeong-Chan Chang and Bor-Sen Chen "Intelligent robust tracking controls for holonomic and nonholonomic mechanical systems using only position measurements" IEEE Trans on Fuzzy System, Vol. 13, No.4, August 2005

[3] Joseba L. Arroyabe, Gerardo Aranguren, Luis A. L. Nozal, Jose L. Martin "Autonomous vehicle guidance with fuzzy algorithm" in Proceedings of IEEE International Conference I.E.C.O.N. Japan, 2000.

[4] C¸avus¸lu MA, Karakaya F, Altun H (2008) C¸KA Tipi Yapay Sinir Ag˘i Kullanılarak Plaka Yeri Tespitinin FPGA'da Donanımsal Gerc¸eklenmesi. In: Proceedings of Akıllı Sistemlerde Yenilikler ve Uygulamalar Sempozyumu 2008 (ASYU 2008) Isparta, Turkey (in Turkish)

[5] Wafa Makni Ben Ayed, « Implémentation de réseaux de neurones sur FPGA Appliqués à la Robotique Mobile », Mastère, Ecole Nationale d'Ingénieur à Sfax, juillet 2007.

[6] Sang-Woo Moon and Seong-Gon Kong, "Block-based neural networks" IEEE Trans Neural Networks, Vol. 12, No. 2, pp. 307-317, March 2001.

[7] Frank Elie « Conception et réalisation d'un système utilisant des réseaux de neurones pour l'identification et la caractérisation, au bord de satellites, de signaux transitoires de type sifflement », thèse de doctorat à l'université de l'Orléans, 1997, pp.101-118

[8] Florent de Dinechin, « Matériels et logiciels pour l'évaluation de fonction numériques Précision, performance et validation », mémoire d'habilitation à diriger des recherches, numéro d'ordre 22-2007, l'université Claude Bernard Lyon1.

[9] Wafa Makni Ben Ayed, « A Neural Controller for lane/wall following system», Ecole Nationale d'Ingénieur à Sfax, juillet 2008.

[10] Mehmet Ali çavuslu «Neural network training based on FPGA with floating point number format and it's performance»,Neural Comput & Applic (2011)