Digital Hardware Implementation of Sigmoid Function and its Derivative for Artificial Neural Networks

HASSENE FAIEDH, ZIED GAFSI, KAMEL BESBES
Laboratoire d'Electronique et de Microélectronique
Faculté des Sciences de Monastir, 5019 Monastir, TUNISIE
kamel.besbes@fsm.rnu.tn

KHOLDOUN TORKI
CMP-TIMA
INPG, 46 Av. Felix Viallet, 38031 Grenoble Cedex, FRANCE
kholdoun.torki@imag.fr

Abstract _ In this paper we propose a polynomial approximation of the sigmoid activation function and its derivative used in artificial neural networks, and we describe the design of the equivalent digital circuit using a floating-point representation for numbers. The simulation of the circuit realized with CMOS technology AMS 0.35_ m under a frequency of 300Mhz shows the efficiency of the implementation.

*Index Terms*_ Artificial neural networks, sigmoid function, polynomial approximation, floating-point, digital hardware implementation.

I. Introduction

In static networks, digital hardware implementation on silicon of neural networks must solve necessarily the implementation problem of the activation function. In dynamic networks where the learning phase is included in the circuit, the back-propagation algorithm [1], [2], [3], [4] and [5] (considered as a standard learning algorithm) deals with the activation function and its first derivative. As a consequence, in this later case, it is necessary to integrate not only the function but also its first derivative. Up to now, the implementation of these functions was done by software and most computers still use these functions through software code. Actually, a hardware implementation is feasible due to the ever-increasing possibilities of silicon integration. Castro et al [6] have shown that "neural networks with a continuous squashing function in the output are universal approximators". This justifies our choice of sigmoid function defined as: $f(x)=1/(1+\exp(-x))$ for the neuron activation function. The derivative is defined as $g(x)=\exp(-x)/(1+\exp(-x))^2$, often written as $g(x)= f(x)[1 - f(x)]$. The shape of these functions is reported on Fig.1.

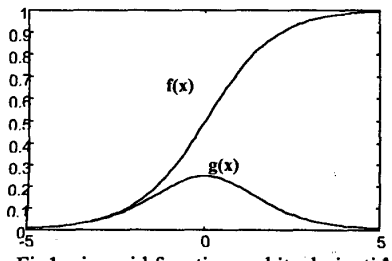

Fig1: sigmoid function and its derivative

One solution for the hardware implementation of these two functions is to use polynomial approximation [7], [8] and [9]. In this paper we propose a polynomial approximation by intervals for functions f and g and we describe the design of the equivalent circuit using a floating-point numbers representation [10] for the polynomials implementation.

II. Polynomial Approximation of Sigmoid Function and its Derivative

Let P_N be the set of polynoms of degree less or equal to N. We want to approximate the continuous function f in the interval [a , b] by an element p^* of P_N. It is important to note that the Taylor development of the function f for a given point in the interval is in general not satisfactory because the Taylor development gives a local and not a global approximation.

A- Mathematical Study

The following two theorems are fundamental results [11] and [12]. The first is due to Weierstrass in 1885 and the second to Chebyshev. We suppose that we want to approximate a continuous function f on interval [a , b].

Approximation theorem (Weierstrass 1885):

For any $\varepsilon>0$, there exist a polynom P such that $\mathrm{Sup}_{[a,b]} | f(x) - P(x) | \leq \varepsilon$.

This theorem shows that a continuous function can be approximated uniformly as far as possible by a polynom. But, it gives us no information on the starting degree of the polynom in function of the desired approximation error. A

theorem derived by Bernstein shows that this degree can be as large as we want.

Theorem (Chebyshev):
P^*_N is the best uniform approximation polynom of f on [a , b] in the set of polynoms of degree less than or equal to N if and only if there exists N+2 points:

$$a \le x_0 < x_1 < x_2 < \ldots < x_{N+1} \le b$$

verifying :

$$P^*_N(x_i) - f(x_i) = (-1)^i [P^*_N(x_0) - f(x_0)] = \pm \mathrm{Sup}_{[a,b]} | P^*_N(x) - f(x) |$$

B. Sigmoid function approximation

Fig. 1 shows that the sigmoid function is continuous and strictly monotonous on $[-\infty, +\infty]$. Physically, we may consider that this function tends to 0 from x = -5 and to 1 from x = 5. For complexity reasons, we consider only polynoms of degree equal to one. In terms of integration, it means we use only once the multiplication, addition and substraction operations.

We use the preceding theorem to find the best approximation of the sigmoid function on the intervals [-5 ,-4], [-4 , -3], [-3 , -2], [-2 , -1], [-1 , 0], [0 , 1], [0 , 1], [1 , 2], [2 , 3], [3 , 4] and [4 , 5]. We suppose that the function is constant on $[-\infty, -5]$ and on $[5 , +\infty]$.

Let approximate p1 (x) = ax+b on [-5 , -4]. Following the theorem, the maximal error ε is reached in three points and takes alternate signs; furthermore, the convexity of the function f on [-5 , -4] implies that the 2 extreme points with the greatest error are the limits of the interval (Fig.2).

Lets define α the third point where this maximal error is reached, we have :

$f(-5) - p_1(-5) = -\varepsilon$, $f(\alpha) - p_1(\alpha) = \varepsilon$ et $f(-4) - p_1(-4) = -\varepsilon$

As the error is maximum on α, we have: $f'(\alpha) - p_1'(\alpha) = 0$. We deduce easily:

a = 0.01129, α = -4.5, b = 0.06248 et ε = 0.0006.

This means that if we limit the degree of the polynom to 1, the function f is approximated on [-5 , -4] by the polynom P1 (x) = 0.01129*x + 0.06248 and the maximum error (0.0006) is reached at the points -5, -4.5 and –4.

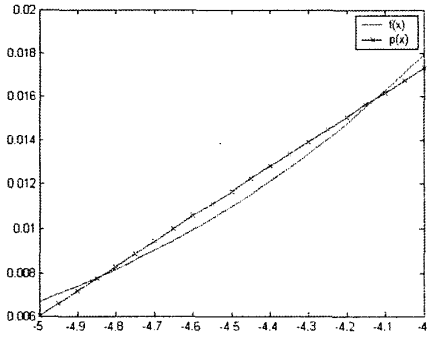

Fig. (2): Polynomial approximation of the function f on [-5 , -4].

The detailed approximation results of function f are summarized on table1.

Table1: polynomial approximation of the function f(x)= 1/(1+exp(-x)).

Interval	a	b	_	_	Polynom
[-5 , -4]	0.01129	0.06248	0.0006	-4.5	P1(x)=0.01129*x + 0.06248
[-4 , -3]	0.02943	0.13404	0.001	-3.5	P1(x)=0.02943*x + 0.13404
[-3 , -2]	0.07177	0.25902	0.003	-2.5	P1(x)=0.07177*x + 0.25602
[-2 , -1]	0.14973	0.41285	0.005	-1.5	P1(x)=0.14973*x + 0.41285
[-1 , 0]	0.23105	0.49653	0.003	-0.5	P1(x)=0.23105*x + 0.49653
[0 , 1]	0.23105	0.50346	-0.003	0.5	P1(x)=0.23105*x + 0.50346
[1 , 2]	0.14973	0.58714	0.005	1.5	P1(x)=0.14973*x + 0.58714
[2 , 3]	0.07177	0.74097	0.003	2.5	P1(x)=0.07177*x + 0.74097
[3 , 4]	0.02943	0.86595	0.001	3.5	P1(x)=0.02943*x + 0.86595
[4 , 5]	0.01129	0.93751	0.0006	4.5	P1(x)=0.01129*x + 0.93751

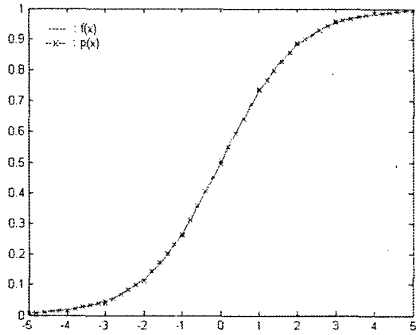

Fig.(3) : Polynomial approximation of the sigmoid function

C. Approximation of the derivative function :

Using the same method, the function g(x)=f(x)[1-f(x)] can be approximated by a polynom of degree 1 of the form P1(x)= a*x +b, on the different intervals.

On interval[-5, -4] :

a=0.01101, α=-4.5, b=0.06107 and ε=0.0006.

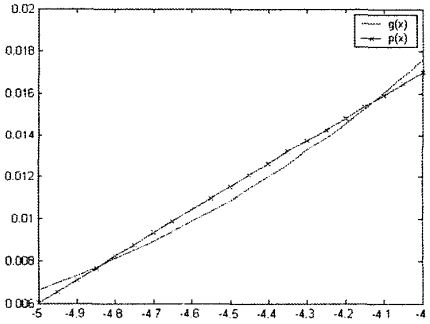

Fig. (4): Polynomial approximation of the derivative function g on [-5 , -4].

The detailed approximation results of the derivative function g are summarized on table2.

Table 2: polynomial approximation of the derivative function g(x)=f(x)[1- f(x)].

Interval	a	b	_	_	Polynom
[-5 , -4]	0.01101	0.06107	0.0006	-4.5	P1(x)=0.01101*x + 0.06107
[-4 , -3]	0.02751	0.12623	0.001	-3.5	P1(x)=0.02751*x + 0.12623
[-3 , -2]	0.05981	0.22213	0.002	-2.5	P1(x)=0.05981*x + 0.22213
[-2 , -1]	0.09161	0.28729	0.0009	-1.7	P1(x)=0.09161*x + 0.28729
[-1 , -1/2]	0.07678	0.27442	0.001	-0.7	P1(x)=0.07678*x + 0.27442
[-1/2 , 0]	0.02999	0.25175	0.001	-0.2	P1(x)=0.02999*x + 0.25175
[0 , 1/2]	-0.02999	0.25175	0.001	0.2	P1(x)=-0.02999*x + 0.25175
[1/2 , 1]	-0.07678	0.27442	0.001	0.7	P1(x)=-0.07678*x + 0.27442
[1 , 2]	-0.09161	0.28729	0.0009	1.7	P1(x)=-0.09161*x + 0.28729
[2 , 3]	-0.05981	0.22213	0.002	2.5	P1(x)=-0.05981*x + 0.22213
[3 , 4]	-0.02751	0.12623	0.001	3.5	P1(x)=-0.02751*x + 0.12623
[4 , 5]	-01101	0.06108	0.0006	4.4	P1(x)=-0.01101*x + 0.06108

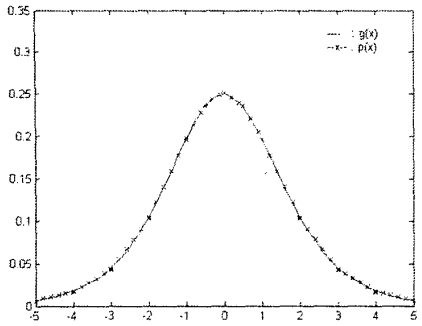

Fig.(5) : Polynomial approximation of the derivative function .

III. The floating-point polynomial circuit

A. The floating-point in simplified format

We use a representation of floating point numbers without keeping the rules of the IEEE normalisation [5]. This representation is done on 24 bits. A positive mantissa and the sign are stored in a 16 bits register; the exponent, a signed integer, is stored in a 8 bit register. The dynamic of the numbers is weaker than the one given by the IEEE standard but covers a largely sufficient area for real numbers. Every real number can be represented in the following form $x=10^{exp}.mant$, where the exponent exp is a signed integer and the mantissa mant is the signed real. In binary representation, the number is written: $x=2^{exp}.mant$. To be unique, the representation imposes a normalized mantissa: $1/2 \leq |mant| < 1$.

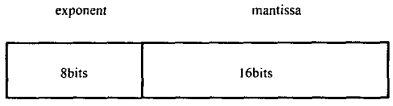

Fig.(6) : A 24 bit floating point number. The high order 8 bits constitute the exponent; the remaining 16 bits, the mantissa or "fractional" part.

Table3. Characteristics of the floating-point representation adopted.

Mantissa register size	Signe + 15 bits
Exponent register size	8 bits
Total size	24 bits
Highest positive number	$2^{127} - 1 = 1,7.\ 10^{38}$
Lowest negative number	$-2^{127} = -1,7.\ 10^{38}$
Approximative precision	2^{-15} _3,05. 10^{-5}

B. Structure of the polynom circuit

The activation function f is now represented by a polynom on a given interval. The choice of a polynom of degree 1 is justified by two reasons, (i) the maximum in the errors is affordable and (ii) the surface on silicon must be kept to a minimum. A polynom of degree2 uses 3 multiplications and 2 adders.

The schematic of the circuit for a polynom of degree 1 is given in figure 7. A simulation example at the logical gate level of this circuit for x = -2.25 gives the following results. The mantissa gives the result 0.0244 and the exponent gives the value of 2; the result can be written as 2^{exp}.mantissa. It is equal to 0.09795. The value of $f(-2.25)$ gives a result of 0.09534 and the error is 0.0022301.

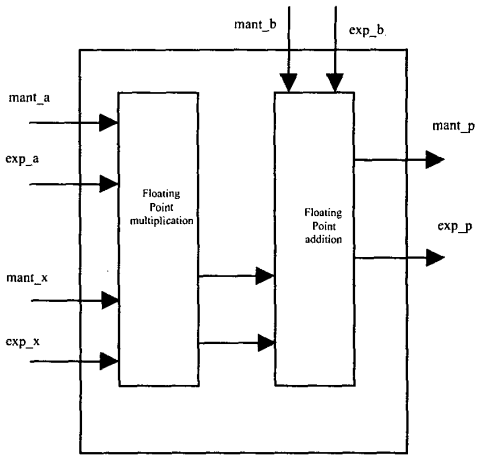

Fig.(7) : Structure of the polynom circuit.

IV. Implementation of the sigmoid function and its derivative

To explore the whole domain of validity, we need to conceive a circuit that takes into account the polynom describing f in respect to the value of x. The structure of this circuit is given by the schematic circuit of Fig. 8.

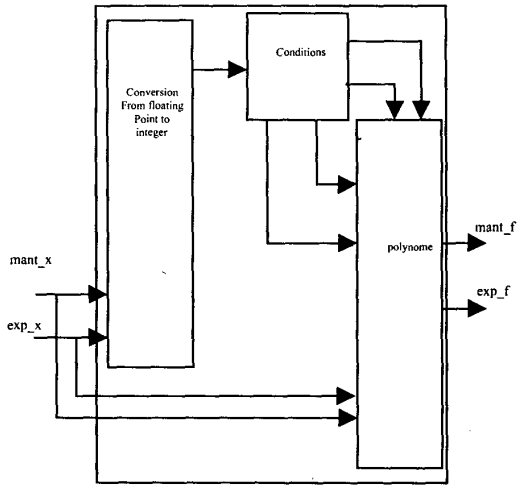

Fig.8: Principle of the circuit.

Layout of the circuit:

With the 0.35 µm technology of AMS, the net compact surface of the circuit is (354x200) μm^2; the brute surface is (385x275) μm^2. The circuit has 764 cells, which are equivalent to 2292 logical gates. Its layout is presented in fig. 9. Simulation results under a frequency of 300Mhz exhibit the efficiency of the implementation.

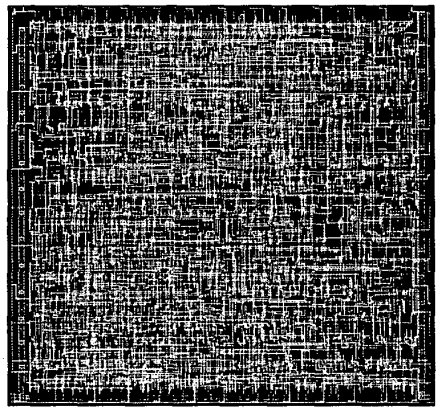

Fig.(9) : The circuit describing the function f after place and route step.

V. Conclusion

A digital hardware implementation of sigmoid function and its derivative for artificial neural networks, using floating-point numbers representation, has been realized. This work will be very useful in future integration of learning on chip digital neural networks on ASICs or FPGA.

REFERENCES

[1] Y.Le Cun. *"A learning scheme for asymmetric threshold network "*, COGNITIVA 85, Paris (France), 1985, pp. 599-604.

[2] B.Parker, *"Learning-logic"*, TR-47, Center for Computational Research in Economics and Management Sciences, MIT, April 1985.

[3] D.E. Rumelhart, L.L.Mc Clelland and the PDP research group. *"Parallel distributed processing exploration in the microstructure of cognition"*, vol. I, II and III. A Bradford book, MIT press, Cambridge (MA), 1986.

[4] Y.Le Cun. *"Modèle connexionniste de l'apprentissage"*. PhD Thesis, Paris VI University, 1987.

[5] J.Hérault and C.Jutten. *"Réseaux neuronaux et traitement du signal"* Hermés, Paris, 1994.

[6] J.L. Castro, C. J. Mantaset, and J.M.Benitez, « Neural Networks with a continuous squashing function in the output are universal approximators », Neural Networks 13 (2000), pp. 561-563.

[7] J.M. Muller « Arithmétiques des ordinateurs, opérateurs et fonctions élémentaires » , Masson, Paris, 1989.

[8] M. Zhang, S.Vassiliadis, and J.G. Delgado-Frias ; " Sigmoid Generators for Neural Computing Using Piecewise Approximations ". IEEE Transactions on Computers, Vol. 45, No 9, pp. 1045-1049, 1996.

[9] K. Basterrextea, J.M. Tarela, and I. Del Campo , « Approximation of Sigmoid Function and the Derivative for Artificial Neurons ». 2001 WSES International Conference on Neural Network and applications, Puerto de la Cruz, Tenerife, Canary Islands, Spain, February 11-15, 2001.

[10] C. Souani, « PhD Thesis, Sciences Faculty of Monastir, Tunisia, june 2000, Appendix1 and Appendix2 ».

[11] E. Rémés, « Sur un procédé convergent d'approximations successives pour déterminer les polynômes d'approximation », C.R. Acad. Sci. Paris, 198, pp. 2063 – 2065, 1934.

[12] P.-J Laurent, « Approximation et optimisation », Collection Enseignement des Sciences, 13, Hermann, Paris, 1972.