VLSI Implementation of Neural Networks

B. M. Wilamowski, J. Binfet, and M. O. Kaynak

Abstract

Currently, fuzzy controllers are the most popular choice for hardware implementation of complex control surfaces because they are easy to design. Neural controllers are more complex and hard to train, but provide an outstanding control surface with much less error than that of a fuzzy controller. There are also some problems that have to be solved before the networks can be implemented on VLSI chips. First, an approximation function needs to be developed because CMOS neural networks have an activation function different than any function used in neural network software. Next, this function has to be used to train the network. Finally, the last problem for VLSI designers is the quantization effect caused by discrete values of the channel length (L) and width (W) of MOS transistor geometries.

Two neural networks were designed in 1.5 μm technology. Using adequate approximation functions solved the problem of activation function. With this approach, trained networks were characterized by very small errors. Unfortunately, when the weights were quantized, errors were increased by an order of magnitude. However, even though the errors were enlarged, the results obtained from neural network hardware implementations were superior to the results obtained with fuzzy system approach.

I. INTRODUCTION

The analog approach is an attractive alternative for nonlinear signal processing. It provides parallel processing with a speed limited only by the delay of signals through the network. In recent years, a significant amount of research has been devoted in the development of fuzzy controllers [1]. In hardware, fuzzy systems dominate current trends in both microprocessor applications [2] and in custom designed VLSI chips [3]. Fuzzy controllers are especially useful for nonlinear systems, which are difficult to describe by mathematical models. Fuzzy controllers are also easier to implement [4][5][6][7]. Membership functions and fuzzy rules are chosen arbitrarily and therefore, fuzzy controllers are often good but not optimal.

Even though neural networks are primarily implemented in software, their good approximation properties make them an attractive alternative in hardware [8][9]. One concern in hardware implementation is related to the quantized values for the weights enforced by hardware [10][11][12]. Another difficulty is caused by fact that the activation functions obtained in VLSI implementation are different from these used in neural network software. Traditionally, neurons use sigmoidal type activation functions. Granted, other types of functions can be used, but sigmoidal type functions allow simple networks the ability to describe complex surfaces. This is due to the bell shaped derivative characteristics of a sigmoidal function. In the case presented, the neuron has a sigmoidal type activation. However, the function to describe this is not readily apparent such as the tangent hyperbolic commonly used in neural network software. This means that standard neural network training software cannot be used because it will produce incorrect solutions for the circuit realization.

In the presented approach, the difficulty with VLSI neural network implementation was overcome in the following way. The "measured" activation function is used for neural network training (Section III) and the training algorithm was adapted to quantized values of weights (Section IV).

II. NEURAL NETWORK CIRCUIT

Nonlinear activation functions of neurons are essential for neural network operation. Such sigmoidal functions can be created in the differential pair shown in Fig. 1.

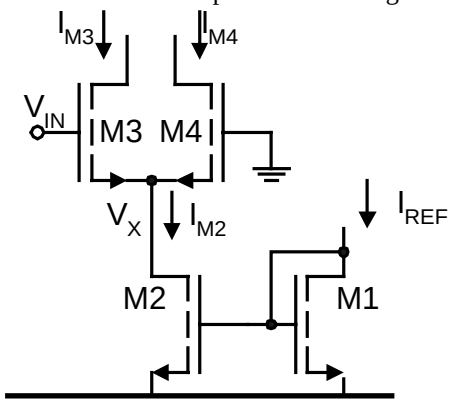


Fig. 1. Sigmoidal function generated by differential pair

Using a simple Shichman-Hodges MOS transistor model for strong inversion, the output currents for the MOS differential pair M3-M4 operating in strong inversion is given by:

$$I_{M3} = \frac{K}{2}\left(V_{IN} - V_X - V_{TN}\right)^2 \tag{1}$$

$$I_{M4} = \frac{K}{2}\left(0 - V_X - V_{TN}\right)^2 \tag{2}$$

and from Kirchhoff's Law

$$I_{M2} = I_{M3} + I_{M4} \tag{3}$$

combining equations (1) through (3) one may find [13]:

$$\text{for } \alpha \le -\frac{1}{\sqrt{\beta}} \quad I_{M3} = 0 \tag{4}$$

$$\text{for } \alpha \ge \frac{1}{\sqrt{\beta}} \quad I_{M3} = I_{M2} \tag{5}$$

$$\text{and for } -\frac{1}{\sqrt{\beta}} < \alpha < \frac{1}{\sqrt{\beta}}$$

$$I_{M3} = 0.5 I_{M2}\left(1 \pm \alpha\sqrt{2\beta - \alpha^2\beta^2}\right) \tag{6}$$

where:

$$\alpha = \frac{V_{IN}}{V_{TN}} \qquad \beta = K\frac{V_{TN}^2}{2I_{M2}} \tag{7}$$

This is a nonlinear transconductance circuit with voltage type input and current outputs. The transfer characteristics of the circuit obtained with SPICE program are shown in Fig. 2.

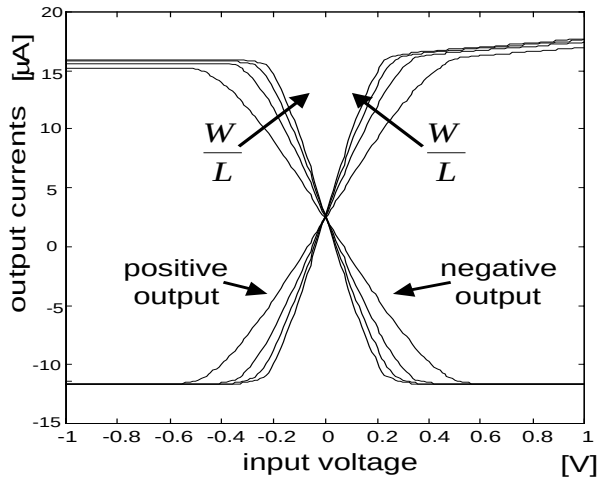


Fig. 2. Transfer characteristics of the circuit of Fig.1.

The circuit does yield a bipolar activation function and its shape can be adjusted by W/L ratios of M3 and M4 transistors. This circuit produces two types of current outputs positive and negative. The weight circuit shown in Fig. 3 can then multiply these currents from the differential pair of Fig. 1.

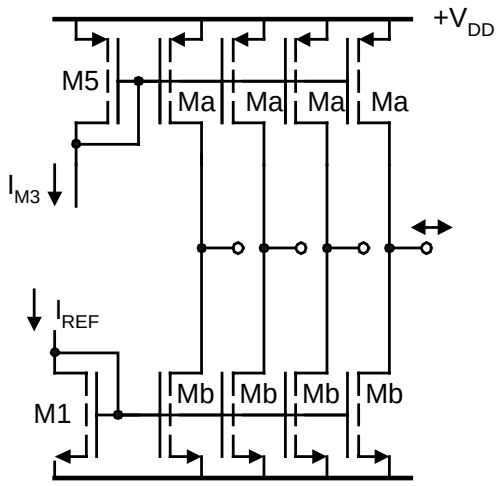


Fig. 3. Weight circuits

Transistor pairs Ma and Mb generate several currents. Each output may have a different weight adjusted by the W/L ratios of transistor pairs Ma and Mc. Each neuron has two weight circuits, one connected to positive output, which generate signals with positive weights, and another one connected to negative output, which generates signals with negative weights.

Fig. 4 shows connection of differential pair circuit of Fig. 1 combined with two weights circuits for positive and negative weights.

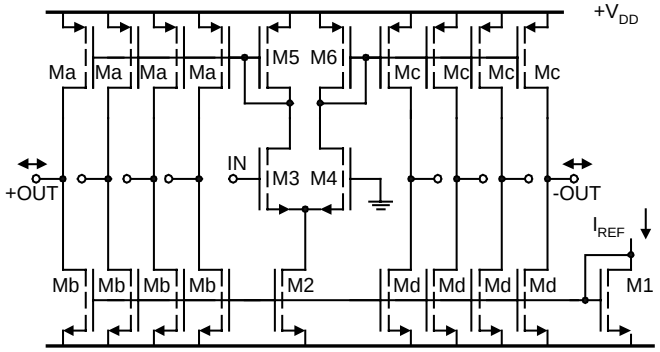


Fig. 4. The neuron circuit with voltage type input and weighted current outputs.

Since this neuron has a bipolar activation function, all output signals could have positive or negative direction on both positive and negative outputs depending on the excitation. A positive output current may have current flow in both directions and the same is possible on the negative output. All positive weights are connected to the positive outputs while all negative weights are connected to negative outputs. These are the desired characteristics, but the circuit had to be modified to remove the straying and biasing effects. Another problem with the conceptual neuron circuit is that the input is a voltage and the output is a current. The circuit had to be modified so that the input would also be a current.

Since the neuron circuit of Fig. 4 has voltage type input and current type outputs a circuit for current to voltage conversion is required. This could be a simple resistor, but in

VLSI, a complex transistor circuit must be used. The current to voltage conversion is done with the circuit composed of transistors M11 and M12 (Fig. 5).

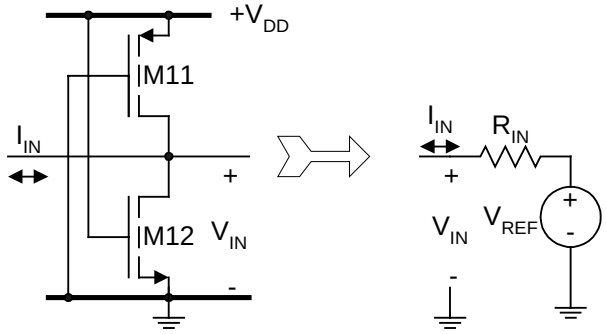


Fig. 5. Circuit for current to voltage conversion

The output characteristics seen from drains of M11 and M12 are given by the equation:

$$I_{IN} = K_N\left[(V_{DD} - V_{TN})V_{IN} - \frac{V_{IN}^2}{2}\right] - K_P\left[(V_{DD} + V_{TP})(V_{DD} - V_{IN}) - \frac{(V_{DD} - V_{IN})^2}{2}\right] \quad (8)$$

where K_P and K_N are transconductance parameters and V_P and V_N are the transconductance parameters for transistors M11 and M12. When W/L ratios are chosen properly ($K = K'*W/L$), then $K_P = K_N = K$, and equation (1) simplifies to

$$I_{IN} = K\left[V_{IN}(V_{DD} - V_{TN} + V_{TP}) - V_{DD}\left(\frac{V_{DD}}{2} + V_{TP}\right)\right] \quad (9)$$

If the input current $I_{IN} = 0$ and $K_P = K_N = K$ then the above equation can be solved for the input voltage.

$$V_{IN} = \frac{V_{DD}}{2}\frac{V_{DD} + 2V_{TP}}{V_{DD} - V_{TN} + V_{TP}} \quad (10)$$

If the threshold voltages are equal, $V_{TN} = -V_{TP}$, then for the case where $I_{TN} = 0$, $V_{IN} = V_{REF} = 0.5V_{DD}$. Note that for $K_P = K_N = K$, all nonlinear terms are canceled and the circuit composed of transistors M11 and M12 has linear input resistance equal to

$$R_{IN} = \frac{V_{IN}}{I_{IN}} = \frac{1}{K(V_{DD} - V_{TN} + V_{TP})} \quad (11)$$

With real transistors or with more accurate transistor models, slight non-linearity of the M11-M12 circuit in Fig. 5 may be observed. However, this is not important since it is superimposed with the nonlinear activation function shaped by M3-M4 differential pair. The value of V_{REF} can be adjusted by changing the relation of the W/L ratios for the PMOS and NMOS transistors. The input resistance is inversely proportional to the *K* parameters of transistors M11 and M12, which can be controlled by W/L ratio.

The positive and negative weights are set by the W/L ratios of transistors Ma and Mb respectively, while the relationship between the pair is kept constant. Note that each weight is set by one transistor and number of weights can be as large as needed.

III. Activation Functions

In traditional feed-forward neural networks, the most commonly used activation functions are sigmoidal and hyperbolic tangent. For microprocessor implementation, the Elliott function can be used since it can be easily evaluated with limited computation resources [2]. For special applications, Gaussian, sinusoidal, cosine, or linear functions can also be used but this is definitely not the current trend. In the case of the VLSI circuit, an activation function similar to the sigmoidal can be obtained, but it cannot be described by a simple function. The activation function for the circuit of Fig. 4 was obtained with a SPICE simulation using BSIM3 [14] transistor model and is shown in Fig. 6. This "measured" function was then approximated by various analytical formulas shown in Table 1 and are shown in Fig. 7. Note that there are very small differences between required and approximate functions. In order to evaluate the quality of approximation differences between required and approximation functions are plotted in Fig. 8.

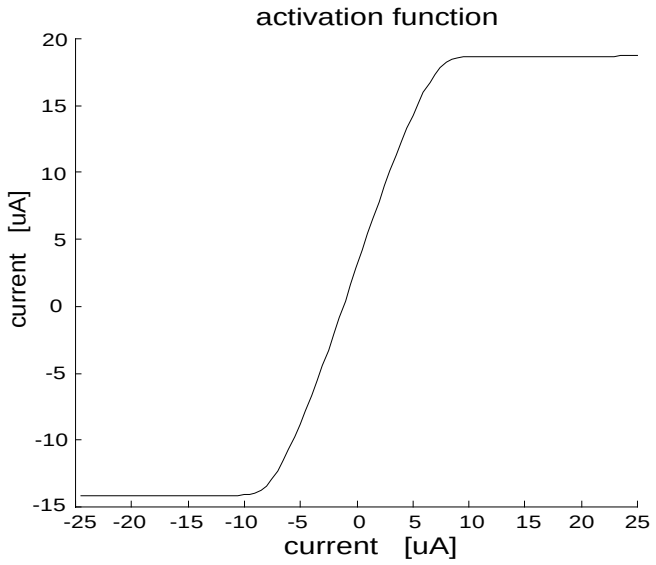


Fig. 6. Activation function of the enhanced neuron circuit obtained using SPICE simulation.

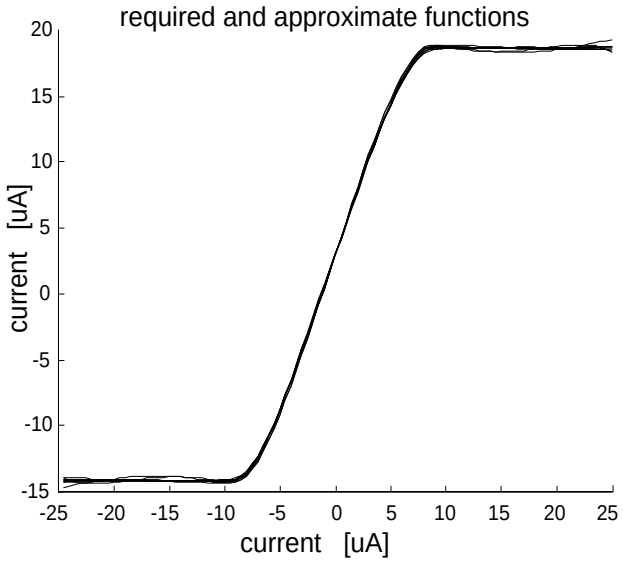


Fig. 7. Various approximations of the activation function obtained using functions of Table 1.

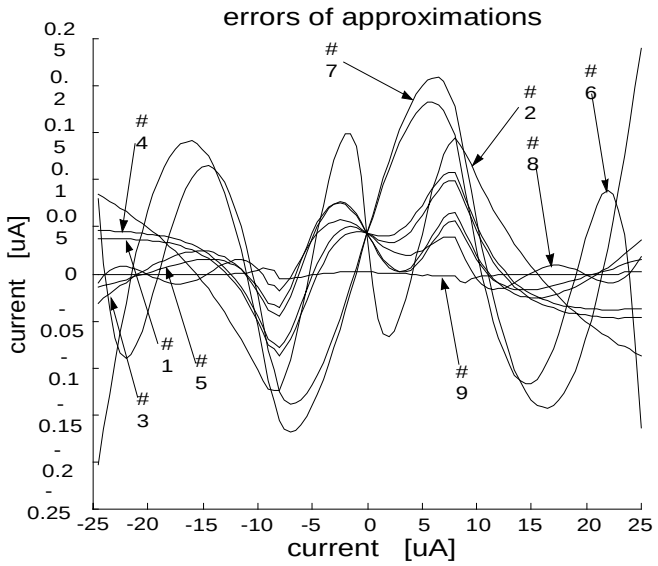


Fig. 8. Differences between the required activation function and approximation functions listed in Table 1.

Table 1. Functions used to approximate the transfer function in the neuron VLSI circuit

No	Function	Error
1	$f(net) = A\tanh(Bnet) = A\dfrac{2}{1-\exp(-2Bnet)} - 1$	0.2114
2	$f(net) = A\dfrac{Bnet}{1+\lvert Bnet\rvert}$ Elliott	0.5569
3	$f(net) = A\dfrac{Bnet}{1+\lvert Bnet\rvert} + C\tanh(Dnet)$	0.0978
4	$f(net) = A\tanh(Bnet) + C\tanh(Dnet)$	0.2183
5	$f(net) = A\tanh(Bnet) + C\tanh(Dnet) + E\tanh(Fnet)$	0.0761
6	$f(net) = Anet + Bnet^3 + Cnet^5 + Dnet^7 + Enet^9$	0.9121
7	$f(net) = A\sin(Bnet) + C\sin(Dnet) + E\sin(Fnet)$	1.4693
8	$f(net) = B\sin(Anet) + C\sin(2Anet) + D\sin(3Anet) + E\sin(4Anet) + F\sin(5Anet)$	0.0500
9	$f(net) = \begin{cases} E & for \quad net \le -A \\ C\,net\sqrt{2 - B^2net^2} + D & for \quad -A < net < A \\ F & for \quad net \ge A \end{cases}$	0.0005

The best results were obtained with function number 9 of Table 1.

$$f(net) = \begin{cases} E & for \quad net \le -A \\ C\,net\sqrt{2 - B^2net^2} + D & for \quad -A < net < A \\ F & for \quad net \ge A \end{cases} \tag{12}$$

where
A = 9.25 μA
B = 0.1081 μA^{-1}
C = 1.7635
D = 2.2375 μA
E = -14.075 μA
F = 18.55 μA

This approximation was used in the training. As the derivative of the function 9 the following expression was used:

$$f'(net) = \begin{cases} 0.01 & for \quad net \le -A \\ \dfrac{2C\left(1 - B^2net^2\right)}{\sqrt{2 - B^2net^2}} + 0.01 & for \quad -A < net < A \\ 0.01 & for \quad net \ge A \end{cases} \tag{13}$$

The additional term 0.01 was essential to eliminate the flat-spot problem [15] which was more likely to occur with the used function than in traditional sigmoidal functions, due to zero derivative values for large net excitations.

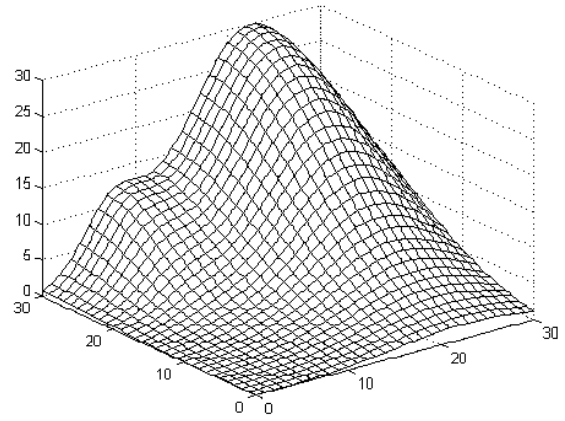


Fig. 9. Required control surface

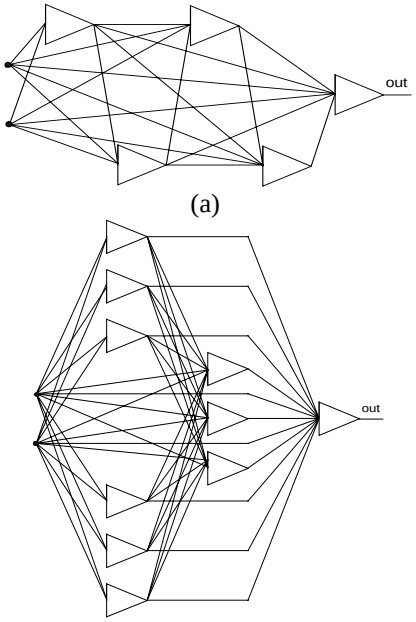
IV. EXPERIMENTAL EXAMPLES

A special MATLAB code was written to train the network, since a custom activation function was used. There are programs available to train neural networks, but using them would require difficult modifications in order to train with a custom activation function. While any training algorithm, such as error back propagation, can be used to train the network, an efficient one should be used in order to speed up the training process. The Lavenberg-Marquardt algorithm [16] was chosen for this task.

Various neural network architectures were trained to the required control surface shown in Fig. 9. During training, the activation function given by Eq. 9 was used. Table 2 shows errors obtained for different neural network architectures. Figure 10 shows the architectures used and Figure 11 shows the control surfaces obtained from training for 1-1-1-1-1 and 6-3-1 architectures.

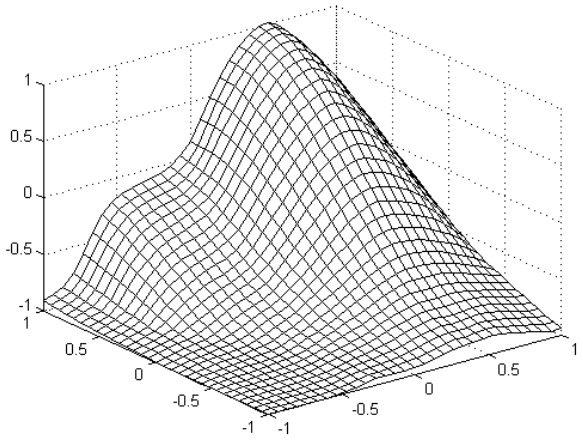
Table 2. Training errors obtained for various neural network architectures

Neural network architectures			Error
No. Layers	No. Weights	Architecture	(SSE)
3	9	1 1 1	1.207
4	14	1 1 1 1	0.2336
5	20	1 1 1 1 1	0.07
3	18	2 2 1	1.6268
2	17	5 1	0.8713
3	29	3 3 1	0.4844
3	49	5 4 1	0.0843
3	47	6 3 1	0.0312
4	53	4 4 1 1	0.0173
4	55	4 3 2 1	0.0525
2	32	10 1	0.1258

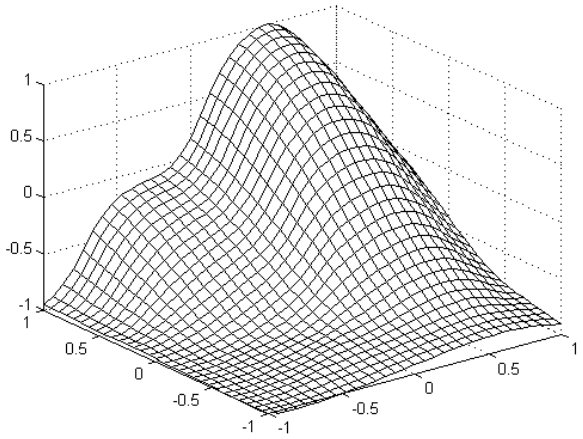


(b)

Fig. 10. Neuron structures (a) for 1-1-1-1-1 architecture and (b) for 6-3-1 architecture



(a)



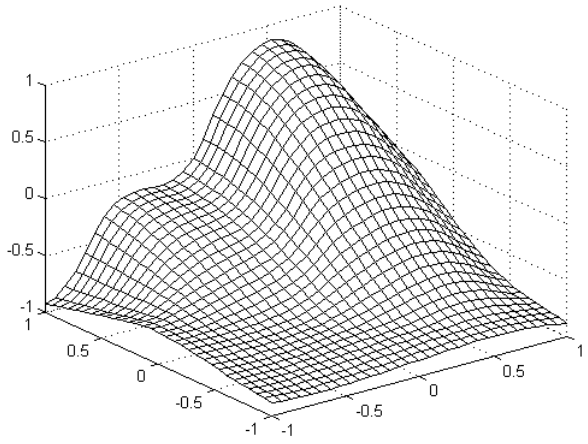
(b)

Fig. 11. Examples control surfaces obtained after training (a) for 1-1-1-1-1 architecture and (b) for 6-3-1 architecture

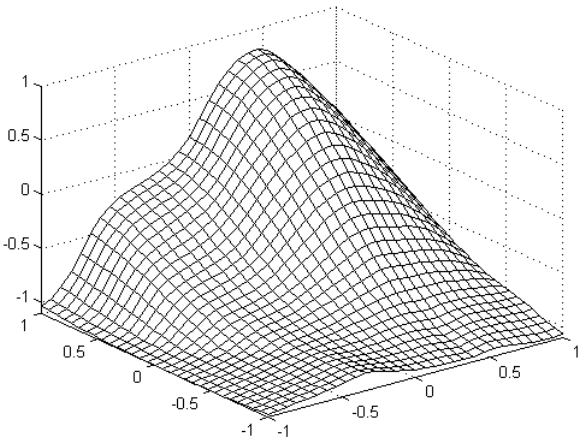
These two architectures were implemented in 1.5μm n-well technology using the MAGIC package for VLSI layout. During the design process, the required W and L values were rounded to the nearest integer values. The maximum transistor size was limited to $3\lambda \le W \le 40\lambda$ and $2\lambda \le L \le 40\lambda$. This way there are 38*39 = 1482 possible combinations of W/L ratios and some of them give repeated values. Removing the repeated values yielded 922 possible combinations. Finally, applying an additional constraint that the total gate area must be smaller than $100\lambda^2$, the number of possible weights is limited to 166.

After designing the VLSI layout using the above constraints, the circuit net list was extracted. Then using the extracted net list, the circuits were simulated with the SPICE program VBase from Veribest using the BSIM3 [14] transistor model. The resulting surfaces for the two implemented circuits are shown in Fig. 12. The error (SSE) was increased from 0.07 to 2.8111 and from 0.0312 to 6.0012 for 1-1-1-1-1 and 6-3-1 architectures respectively. Note that the weight quantization has very significant effect on the quality of control surface.

The quantization process strongly depends on the quality of the layout tools. With a simple tool and scalable design rules, the minimum raster size is comparable to the minimum feature size. For example, with MAGIC layout software and the scalable CMOS technology, the minimum channel length is equal to 2λ, where λ is the raster size (0.8μm in our case). With this approach, both W and L may have only integer values and this enforces a very strong quantization effect. With more powerful layout design tools such as Mentor Graphics or LASI, where pattern rotation is allowed, the quantization effect is not as critical, since the raster has at least 10 times larger resolution.

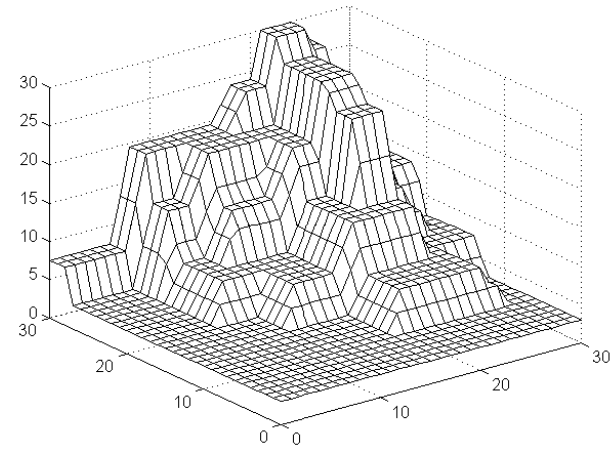


(a)

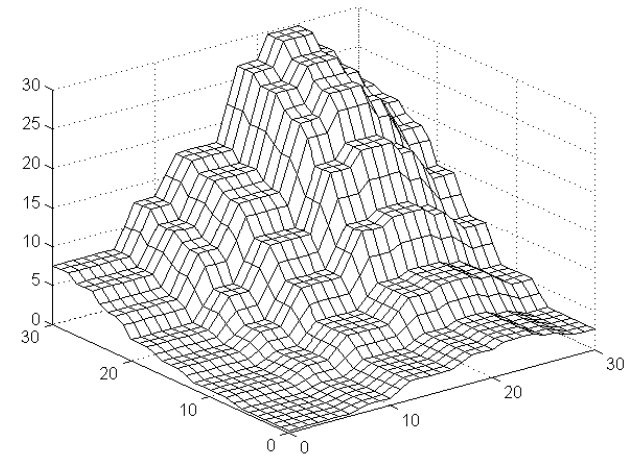


(b)

Fig. 12. Control surfaces obtained with SPICE simulation (weights were quantized for transistors Ma, Mc and Mbd). (a) for 1-1-1-1-1 architecture and (b) for 6-3-1 architecture



(a)



(b)

Fig. 13. Control surface using fuzzy approaches (a) obtained with trapezoidal membership functions and Zadeh approach and (b) obtained with trapezoidal membership functions and Tagagi-Sugeno approach.

V. CONCLUSION

Fuzzy controllers do have several advantages such as simple rule based design, but they usually produce relatively raw control surfaces, which are not acceptable for precision control [2]. The control surfaces obtained with fuzzy systems are shown in Figure 13 for the same required surface as shown in Fig. 8. No matter if the Zadeh [17] (Fig. 12(a)) or Tagagi-Sugeno [18] (Fig. 13(b)) approach was used, a relatively raw control surface was obtained. These fuzzy control surfaces also exhibit larger errors, 908.4 and 296.5 for Fig. 13 (a) and (b) respectively. With the neural network approach presented in this paper, the resulting control surfaces are very smooth.

Although the presented examples were for a two input case, the general nature of neural systems is such that they can easily handle multidimensional problems. This is not true for the fuzzy systems where the number of inputs is severely limited because with an increased number of inputs, the size of the rule table grows exponentially.

References

[1] Wilamowski B. M. "Neuro-Fuzzy Systems and its applications" tutorial at 24th *IEEE International Industrial Electronics Conference - IECON'98* August 31 - September 4, 1998, Aachen, Germany, vol. 1, pp. t35-t49.

[2] Wilamowski B.M. and J. Binfet, "Do Fuzzy Controllers Have Advantages over Neural Controllers in Microprocessor Implementation" Proc. of. 2-nd *International Conference on Recent Advances in Mechatronics - ICRAM'99*, Istanbul, Turkey, pp. 342-347, May 24-26, 1999.

[3] Wilamowski B. M. and R. C. Jaeger, "Neuro-Fuzzy Architecture for CMOS Implementation" *IEEE Transaction on Industrial Electronic*s vol. 46, no.6, Dec. 1999.

[4] Choi J., B.J.Sheu, and J.C.F. Chang, (1994) A Gaussian Synapse Circuit for Analog VLSI Neural Networks. *IEEE Trans. on Very Large Scale Integration (VLSI) Systems*, vol. **2**, no. 1, pp. 129-133.

[5] Rodriguez-Vazquez A. and F. Vidal-Verdu, Learning in Neuro/Fuzzy Analog Chips, *IEEE International Symposium on Circuits and Systems*, Seattle WA, vol. **3**, pp. 2325-2328, April 30-May 3 1995.

[6] Yamakawa, A fuzzy Inference Engine in Nonlinear Analog Mode and its Application to a Fuzzy Logic Control, *IEEE Trans. on Neural Networks*, vol. 4, pp. 496-522, 1993.

[7] Ota, Y. and B. M. Wilamowski, " CMOS Implementation of a Voltage-Mode Fuzzy Min-Max Controller", *Journal of Circuits, Systems and Computers*, vol. 6, No 2, pp. 171-184, April 1996.

[8] Hornik, K., "Multilayer Feedforward Networks as Universal Approximators," *Neural Networks*, v.2, pp 359-366, 1989.

[9] Funahashi, K., "On the Approximate Realization of Continuous Mappings by Neural Networks," *Neural Networks*, v.2, pp 183-192, 1989.

[10] Draghici S., Sethi, I.K. - On the possibilities of the limited precision weights neural networks in classification problems in Biological and Artificial Computation: From Neuroscience to Technology, *Lecture Notes in Computer Science*, J. Mira, R. Moreno-Diaz, J. Cabestany (Eds.), p. 753-762, Springer Verlag, 1997

[11] Beiu, V, Draghici S. - Limited weights neural networks: very tight entropy based bounds in Proc. *2nd Intl. ICSC Symposium On Soft Computing, Fuzzy Logic, Artificial Neural Networks, and Genetic Algorithms - SOCO'97*, Nimes, France, September 14-17, 1997).

[12] Cupal, J.J., B. M. Wilamowski, R. S. Sandige, and J. Miller, " A Fractional Powers-of-Two Number System for Digital Neural Networks, *International Journal of Modeling and Simulation*, vol. 16, No 3, pp. 123-128, 1996

[13] Wilamowski, B. M. and R. C. Jaeger, "VLSI Implementation of a Universal Fuzzy Controller," *Intelligent Engineering Systems Through Artificial Neural Networks*, vol. 7, ed. C. H. Dagli and others, by ASME Press, New York 1997, pp. 351- 356.

[14] Huang J. H., Z. H. Liu, M. C. Jeng, K. Hui, M. Chan, P. K. Ko, and C. Hu, "BSIM3 Manual," Department of Electrical Engineering and Computer Science, University of California, Berkeley.

[15] Wilamowski, B.M. and L. Torvik, "Modification of Gradient Computation in the Back-Propagation Algorithm", presented at *Artificial Neural Networks in Engineering - ANNIE'93*, St. Louis, Missouri, November 14-17, 1993; also in *Intelligent Engineering Systems Through Artificial Neural Networks* vol. 3, pp. 175-180, ed. C. H. Dagli, L. I. Burke, B. R. Fernandez, J. Gosh, R.T., ASME PRESS, New York 1993.

[16] Hagan M. T. and M. Menhaj, "Training feedforward networks with the Marquardt algorithm," *IEEE Transactions on Neural Networks*, vol. 5, no. 6, pp. 989-993, 1994.

[17] Zadeh L. A., "Fuzzy sets". *Information and Control*, New York, Academic Press vol **8**, pp. 338-353, 1965.

[18] Takagi T. and M. Sugeno, Derivation of Fuzzy Control Rules from Human Operator's Control Action. *Proc. of the IFAC Symp. on Fuzzy Inf. Knowledge Representation and Decision Analysis*, pp. 55-60, July 1989.