Silicon Implementation of Computational Intelligence for Mechatronics

Bogdan M. Wilamowski, *Fellow Member, IEEE*
Auburn University , USA

***Abstract* - Digital implementations of neurocomputers are presently quite expensive, they require excessive power, they suffer from a number of issues that cause performance characteristics to differ from the theoretical model of the system, and they are relatively intolerant of fault conditions. The inherent advantages of the massively parallel structure of these systems are also lost in the common practice of executing algorithms sequentially on a conventional computer. The paper presents nonlinear analog signal methodology where for nonlinear processing nonlinear characteristics of MOS transistors are used.**

I. INTRODUCTION

Electro-mechanical devices require sophisticated nonlinear controllers. Design and implementation of such controllers are not easy. Often computational methodologies are needed and this requires combination of several techniques such as neural networks fuzzy systems or evolutionary computation.

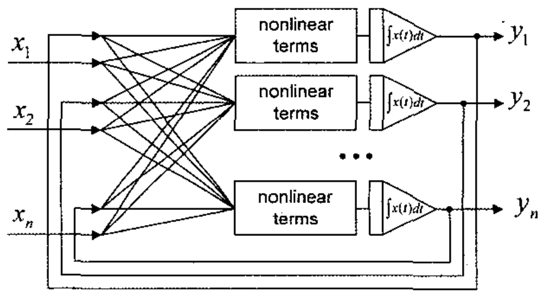
As a likely result of the on-going development of computer technology we may expect that massive parallel processing and soft computing will significantly enhance traditional computation methods. A natural consequence of this rapid growth is the emergence of the field of intelligent systems. The machine-intelligent behavior is determined by the flexibility of the architecture, the ability to realize machine incorporations of human expertise, laws of inference procedure and the speed of learning. All these titles are the main constituents of the research area named Computational Intelligence or Soft Computing. It is a practical alternative for solving mathematically intractable and complex problems. The main subdivisions of the area are artificial neural networks and fuzzy inference systems [1-4].

The mathematical power of machine intelligence is commonly attributed to the neural-like system architecture used and the fault tolerance arising from the massively interconnected structure. Such systems are characterized by heavy parallel processing. The last feature is unfortunately lost if algorithms are implemented using conventional microprocessors or digital computers.

Another aspect of soft computing systems is that instead of "zero" and "one" digital levels, they use fuzzy/continuous levels and in this way much more information is passed through the system. Conventional digital computers are not well suited for such signal processing.

A third feature of soft computing systems is their survivability in the presence of faults; this means they may work correctly if they are partially damaged. In contrast, a one-bit fault in traditional computers may lead to catastrophic results. Therefore there is a significant interest in development of special hardware for soft computing. Several special issues of various journals have been devoted to these topics and several reference books of collected articles on the subject have been published [5-7].

Computationally intelligent methodologies are commonly used for identification and/or control of nonlinear dynamic systems, the behaviour of which can usually be described by a set of nonlinear differential equations in state variable form as indicated in Fig. 1. Integrators can easily be implemented in silicon. It could be for example an operational amplifier with a feedback capacitor, but other simpler circuits are also possible. The difficult task in hardware realization is to implement arbitrary nonlinear terms. Fuzzy or neural systems are the prime candidates for that purpose. The focus of this proposal is on hardware implementation of computationally intelligent systems and on methods to design/train them.



$$y_1 = \int f_1(x_1, x_2, \cdots x_n, y_1, y_2, \cdots y_n)\ dt$$
$$y_2 = \int f_2(x_1, x_2, \cdots x_n, y_1, y_2, \cdots y_n)\ dt$$
$$\cdots$$
$$y_n = \int f_n(x_1, x_2, \cdots x_n, y_1, y_2, \cdots y_n)\ dt$$

Fig. 1. Block diagram of an arbitrary nonlinear dynamic system.

II. COMPARISON OF NEURAL AND FUZZY SYSTEMS FOR NONLINEAR FUNCTION MAPPING

Both neural networks and fuzzy systems are capable of approximating any nonlinear function, but their implementation in silicon is not easy. Both neural and fuzzy systems have their advantages and limitations.

Fuzzy systems utilize the expert information in the form of a set of rules. There are several reasons for using fuzzy systems in control engineering practice. First, the dynamics of the system under interest is generally complicated, but sometimes its behavior can be defined more easily in linguistic terms. Second, fuzzy systems are suitable architectures for modification and tuning process, which provides some kind of adaptiveness through the on-line adjustment of parameters. The major advantage of fuzzy logic based systems is their ability to utilize expert knowledge and perception based information.

Artificial neural networks are well known by their property of performing complex nonlinear mappings. Earlier works on the mapping properties of these architectures have shown that neural networks are universal approximators [8-10]. Even though neural networks today are primarily implemented in software, their good approximation properties make them an attractive alternative for hardware implementation. One concern in digital implementation is related to the quantization of weights required by digital hardware [11]. Another difficulty is caused by the fact that the activation functions obtained in practical digital VLSI implementation are different from those used in neural network design software. This means that standard neural network software cannot accurately represent the solutions one will get by actual circuit realization.

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 harder to train, but provide an outstanding control surface with much less error than that of a fuzzy controller. Figures 2, 3, and 4 show a comparison of fuzzy and neural network based system implemented in Motorola HC11 microcontroller. Motorola's 68HC711E9 is a low cost, 8-bit microprocessor; the on-board features of which are 512 bytes of RAM and EEPROM and 12K bytes of UV erasable EPROM. The processor was used with an 8 MHz crystal, allowing an internal clock frequency of 2 MHz.

A drawback of neural controllers is that the design process is more complicated than that of fuzzy controllers. However, this difficulty can be easily overcome with proper design tools. One severe disadvantage of a fuzzy system is its limited ability of handling problems with multiple inputs. In the case of neural networks such a limitation does not exist. Furthermore, control surfaces obtained from neural controllers also do not exhibit the roughness of fuzzy controllers that can lead to unstable or rough control.

Before neural networks can be implemented on VLSI chips several problems have to be solved. 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.

	Fuzzy System (Zadeh)	Fuzzy System (Tagagi-Sugeno)	Neural Network 2-1-1-1	Neural Network 2-1-1-1-1-1
Length of code in bytes	2324	1502	680	1119
Processing tin in ms	1.95	28.5	1.72	3.3
MSE Error	0.945	0.309	0.000578	0.000093

During the last decade many, more or less efficient algorithms for neural network traning have been developed. This proposal does not have the objective of developing better algorithms for the general purpose of neural network training, but it aims, based on the experience previously gained during a NSF sponsored project, to adopt or develop special algorithms for specific hardware architectures. During the course of the research activities, the project may also branch into easily retrainable algorithms where only a few new patterns are being replaced in the training batch.

This is especially important adaptive systems or adaptive critics metodology is used. Significant progress has already been made in learning algorithms and dedicated neural network architectures[12]. As has previously been discussed, the major advantage of neural systems is its parallel computation. Most current implementations of neural networks are done on Neumann type digital computers where all computation is done sequentially and as result, the unique advantage of neural network parallelism is being lost.

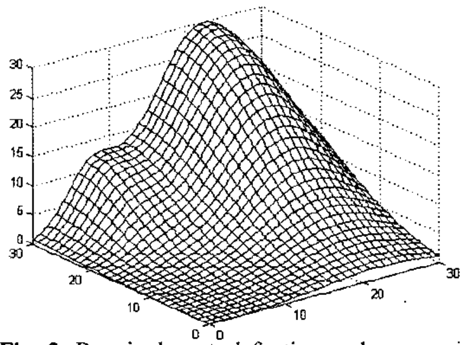
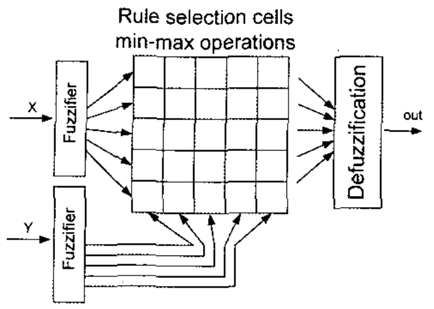


Fig. 2. Required control funtion and a comparison of the results obtained with microcontroller implementation using

fuzzy (details of which are given in Fig. 3) and neural (details of which are given in Fig. 4) approaches. [13]



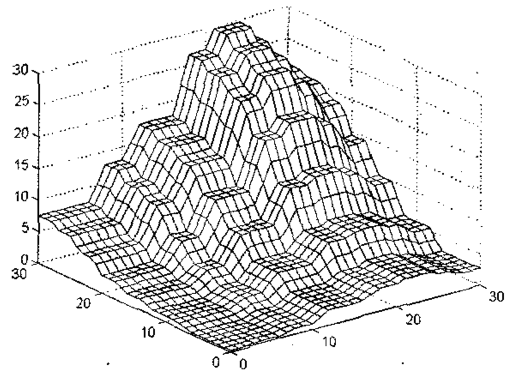
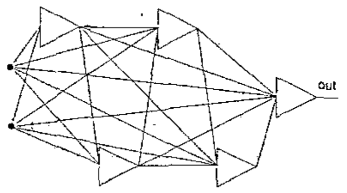


Fig. 3. Control surfaces obtained with Motorola microcontroller HC11 using fuzzy approach with trapezoidal membership functions (7 functions per input) and Tagagi-Sugeno defuzzification [13]



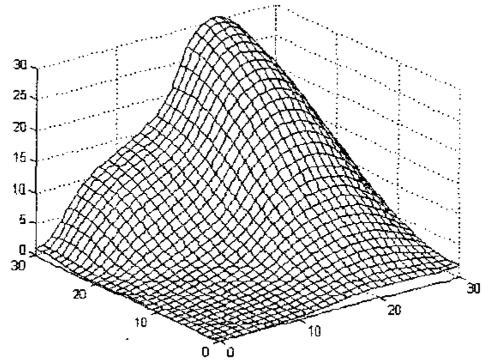
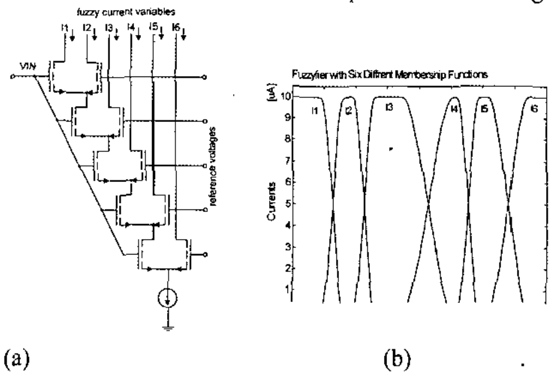


Fig. 4. Control surfaces obtained with Motorola microcontroller HC11 using fuzzy approach with six neurons 2-1-1-1-1-1 architecture and Elliot activation function. [13]

III. HARDWARE IMPLEMENTATION OF FUZZY SYSTEMS

Fuzzy systems dominate current trends in both microprocessor applications [13] and in custom designed VLSI chips [14]. Fuzzy controllers are especially useful in nonlinear systems since such systems are difficult to describe using conventional mathematical models. The use of an analog approach is an attractive alternative for this nonlinear signal processing. It provides parallel processing with a speed limited only by the delay of signals through the network. However, membership functions and fuzzy rules are chosen arbitrarily and therefore fuzzy controllers are often good but not optimal. Nevertheless, in many cases, simple fuzzy controllers perform better than traditional ones, especially if systems are very nonlinear. Their performance can significantly be improved when they are tuned with neural networks or by evolutionary algorithms [15]. It should be pointed out that the classical approach to fuzzy control [16], although possible [17-20] is difficult to implement in analog hardware.

Analog signal processing is usually much faster than digital. Several computation processes can be done simultaneously, and AD and DA conversion are not required. Analog integration or differentiation has already been used for many years. Analog summation and multiplication are also quite common. For computational intelligence, more sophisticated nonlinear function realizations are required such as WTA (Winner Takes All), fuzzy membership functions, normalization circuits, MIN and MAX operators, division circuits, and others [17-23]. In realizing such nonlinear signal processing, the nonlinear characteristics of MOS devices can provide an advantage.



(a) (b)

Fig. 5. Fuzzifier circuit with four differential pairs creating six membership functions: (a) circuit diagram and (b) fuzzifier characteristics.

As is stated above, the classical approach to fuzzy systems as presented by Zadeh [16] is difficult to implement in analog hardware. Especially difficult is the defuzzifier where signal division has to be implemented.

The division problem can be avoided through the use of feedback loops, but this approach can lead to limited accuracy and to stability problems. An alternative approach, taken here, is the use of simplified Takagi-Sugeno singleton inference rules [13], where normalization is used.

Consider, as an example, the analog implementation of a fuzzifier, which must convert crisp analog values into several fuzzy variables. The conversion takes place using membership functions of triangular, trapezoidal or Gaussian type shapes. Several different circuits have already been proposed [17-23]. In the circuit shown in Fig. 5, only one differential pair is required per membership function and one current source per fuzzifier. Note that several different shapes of the membership function can be obtained by selecting different W/L (width/length) ratios of MOS transistors. This circuit was used in the fabricated fuzzy chip shown in Fig. 6, and the measured characteristics are shown in Fig. 7. A relatively smooth control surface (better than what can be obtained by microprocessor implementation) was a result of non-ideality of transistor characteristics, which do not exhibit sharp transitions (which means that the first and the second derivatives are continuous).

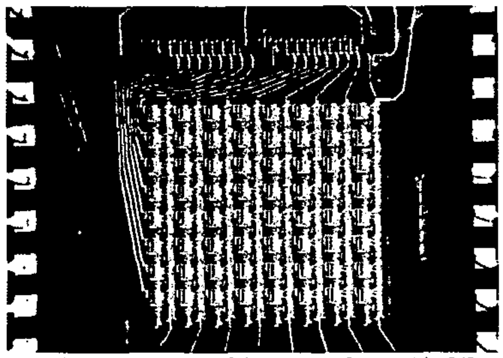


Fig. 6. Microphotograph of the analog fuzzy chip [4]

The design of an FPGA (Field Programmable Gate Array)-based fuzzy controller can be very simple. As shown in Fig.8, it consists of an FPGA, analog-to-digital (A/D) converters for the inputs, a digital-to-analog (D/A) converter for the output and a ROM chip. The design is done on the 4010 FPGA using 97% of the chips resources. The maximum delay between flip-flops determines the clock frequency. The maximum clock frequency for this design is 9.6MHz. The size of the look up table grows exponentially as the number of inputs increases. This is the downfall of the solution of Fig. 4(a). The size of ROM can be significantly reduced if the concept presented in Fig. 5 is used. In this approach only three or four most significant bits of each input determine the address for the lookup table. A weighted average of least significant bits is used to eliminate rawness (Fig. 8).

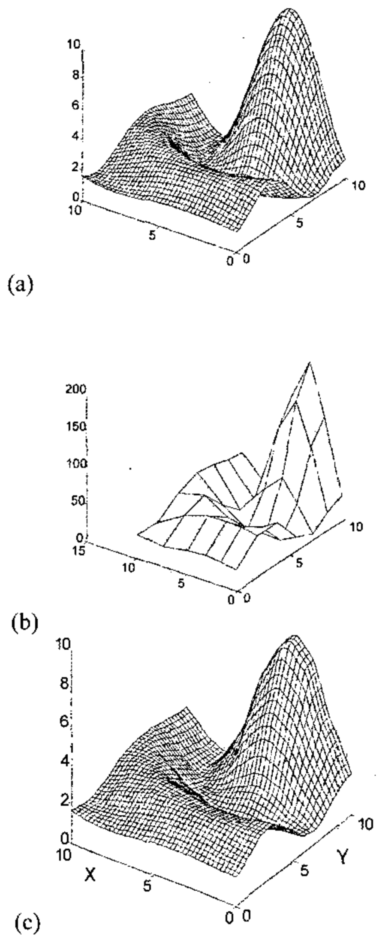
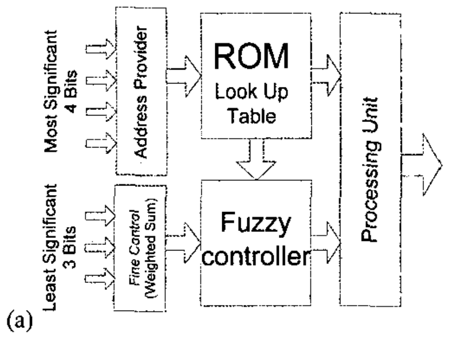
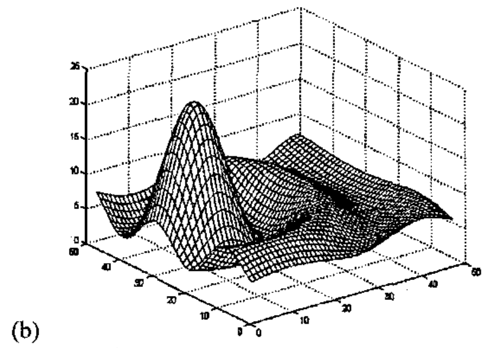


Fig. 7. Control surfaces: (a) desired control surface, (b) information stored in defuzzifier as weights, and (c) measured control surface of VLSI [4]

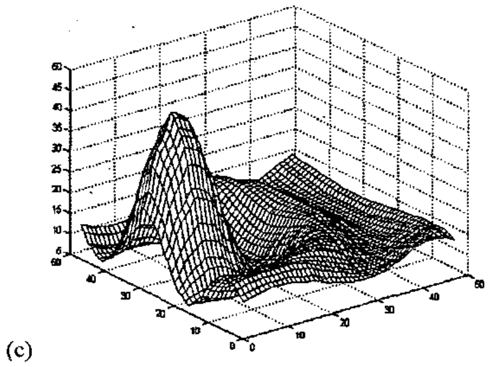
Microprocessor systems are fast enough for most mechatronics systems. The control surfaces obtained by their use can be rather rough but the roughness can partially be compensated by using product encoding instead of min encoding or by the use soft membership functions (for example Gaussian). Another way to eliminate roughness is to use second order approximation algorithms [25-26]. Unfortunately these improvements are more computationally intensive and result in much slower performance.



(a)



(b)



(c)

Fig. 8. FPGA implemented fuzzy controller(a) , the required (b) , and obtained (c) control surfaces.

IV. HARDWARE IMPLEMENTATIONS OF NEURAL NETWORKS

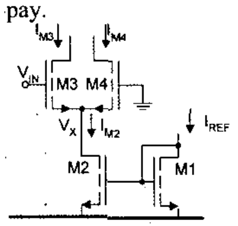
It has been stated above that, in hardware implementations of *soft computing methodologies, fuzzy systems dominate* current trends in both microprocessor applications [13] and in custom designed VLSI chips [4]. However control surfaces obtained from fuzzy controllers are rough, which can cause unstable control. On the other hand, neural networks usually require a computation of tangent hyperbolic activation functions. This task it often too complex for simple microprocessors. When the tangent hyperbolic function is replaced by the Elliott function then the computations are relatively simple.

$$f(net) = \tanh(net) = \frac{2}{1-\exp(-2net)} - 1 \qquad (1)$$

$$f(net) = \frac{net}{1+|net|} \qquad (2)$$

With such an approach, neural network implementation can be done with shorter code, resulting in faster operation, and much more accurate results. Figs. 2, 3 and 4 show the comparison of several controllers for the same desired control surface implemented in the popular HC11 micro-controller using various fuzzy and neural network architectures. A microprocessor based system can easily be reprogrammed, but training them on fly would be a real challenge (fuzzy approaches are better from this point of view). The major disadvantage of microprocessor-based implementations is that they do not take advantage of massive parallel computing as occurs in nervous systems.

Most common neural systems use neurons with sigmoid activation functions. Other activation functions such as Gaussian or cosine are also used. The problem is that neither of the latter functions can easily be implemented in silicon. However, this can be solved by the use of existing *nonlinear characteristics of silicon devices, rather than* focusing on implementing traditional functions (like tanh()) in silicon which is rather difficult). Such an approach would require a modification of training algorithms, but this is a small price to pay.



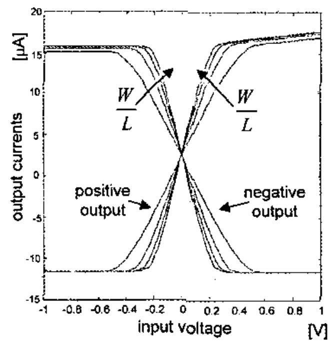


Fig. 9. "Squashed function" generated by differential pair

There are some problems that have to be solved before neural networks can be implemented in VLSI chips in an effective manner. First, a new approximation function needs to be developed because CMOS neural networks can produce activation functions different than any function used in standard neural networks. One of the simplest approaches would be the usage of a "squashed function" produced by a differential pair as shown in Fig. 9. Next, the network must be trainable using this function and this requires a specially developed algorithm.

The on-going miniaturization of silicon chips leads to problems with device tolerance limits. Cutting edge technology is limited by tolerances. When we become able to produce devices with acceptable tolerances, size is reduced once more and we are brought back to facing the tolerance dilemma again. However, the biological nervous system is not so sensitive to such tolerance elements [1]. The prime reason is that in the vertebrate nervous system, communication between distant neurons is accomplished using encoded pulse streams [29]. Pulse-stream encoding techniques use pulse streams to carry information and control analog circuitry, while storing further analog information on the time axis. The firing rate of action potentials in biological neurons is roughly proportional to the change in the original graded potential, which is categorized as *frequency modulation*. Padgett, Werbos, and Kohonen [2] present an overview of the use of PCNN in pattern recognition applications. In a special issue of IEEE Trans. on Neural Networks [28] a number of good references of PCNN technologies are given. PCNN are not sensitive to the magnitude of the signal since information is coded in signal timing. In the case of digital implementations there are several ways to code the information as is illustrated in Fig. 10-12.

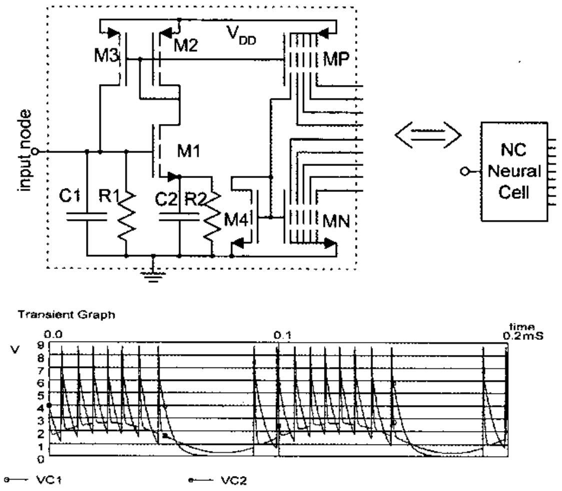


Fig. 10 Electronic implementation of Pulse Couple Neural Networks

V. SILICON DEVICES FOR NONLINEAR SIGNAL PROCESSING

It is also possible to use other approaches than fuzzy or neural systems for nonlinear mapping. For example, several nonlinear blocks implemented in VLSI could be linked together, more or less in a random way, and the desired nonlinear characteristics could be obtained by tuning links and device parameters. Several development methods could be implemented for such systems. One way would be to use genetic algorithms to find close to optimum architectures and then gradient based methods could be used (similar to those used for neural networks) for final tuning.

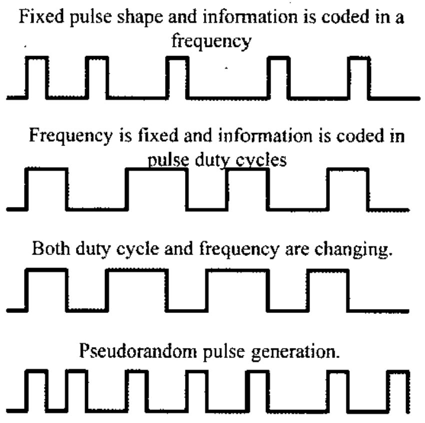

Fixed pulse shape and information is coded in a frequency
Frequency is fixed and information is coded in pulse duty cycles
Both duty cycle and frequency are changing.
Pseudorandom pulse generation.

Fig. 11. Various method of implementation of the pulse nature of real nervous system into digital hardware.

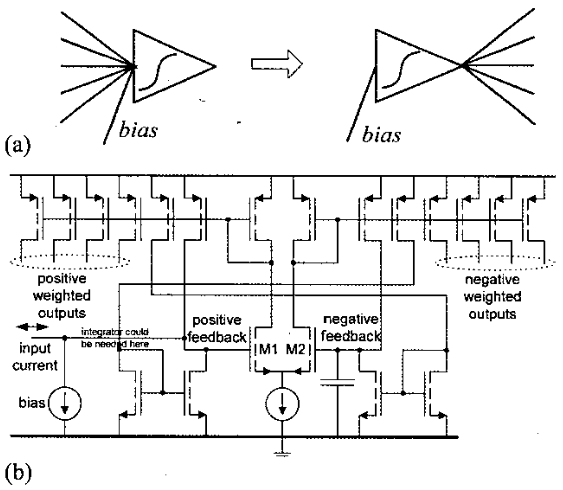

(a)
bias
bias
positive weighted outputs
negative weighted outputs
positive feedback
negative feedback
integrator could be needed here
input current
bias
M1 M2
(b)

Fig. 12. Neuron circuit working in the pulse mode with duty cycle modulation. It has two outputs, positive and negative, therefore it is relatively easy to implement both positive and negative weights. Nonlinear activation function (squashed type) is automatically generated by the differential pair M1 M2.

As is stated above, nonlinear functions can be directly implemented by VLSI circuits. For example, exponential and logarithmic functions can be accomplished by using the nonlinear characteristics of p-n junction, bipolar transistor characteristics, or MOS transistor characteristics operating in sub threshold conduction mode. There are several examples that can be given to illustrate the creation of arbitrary nonlinear blocks using nonlinear characteristics of MOS transistors. For the square relationship, the MOS transistor characteristics operating in strong inversion mode can be used. The circuit shown in Fig. 13 can approximate sine and cosine functions, which are required for robot kinematics. Figure 14 shows a four quadrant multiplier, implemented with MOS transistors operating in sub-threshold conduction mode.

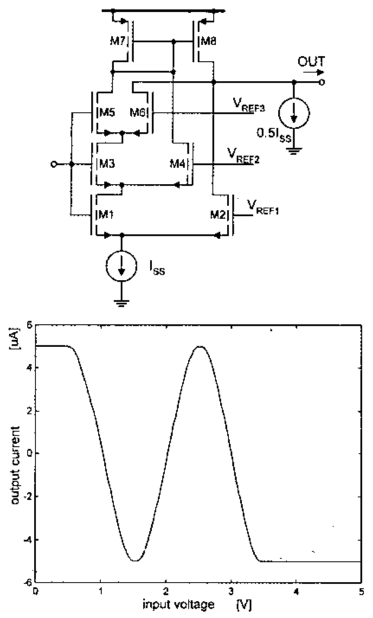


Fig. 13. Hardware implementation of cosine function using several differential pairs.

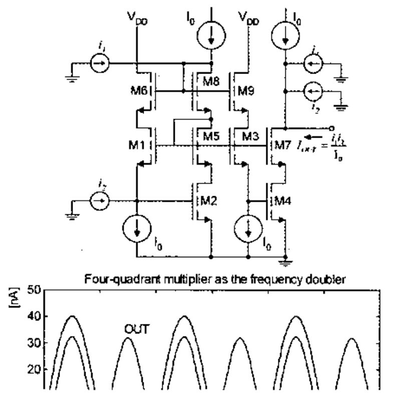

V_{DD}
I_0
V_{DD}
I_0
i_1
M6
M8
M9
i_2
M1
M5
M3
M7
$I_{OUT} = \frac{i_1 i_2}{I_0}$
i_2
M2
M4
I_0
I_0
Four-quadrant multiplier as the frequency doubler
[nA]
OUT

Fig. 14. Four quadrant multiplier (a) circuit diagram (b) multiplier used as the frequency doubler.

Neural networks could be implemented as VLSI chips in several ways. The circuit in Fig. 15 works as a nonlinear transconductance amplifier where inputs and outputs are currents. Weights are implemented by selecting different W/L ratios. With positive and negative outputs of the differential pair both positive and negative weights can be implemented.

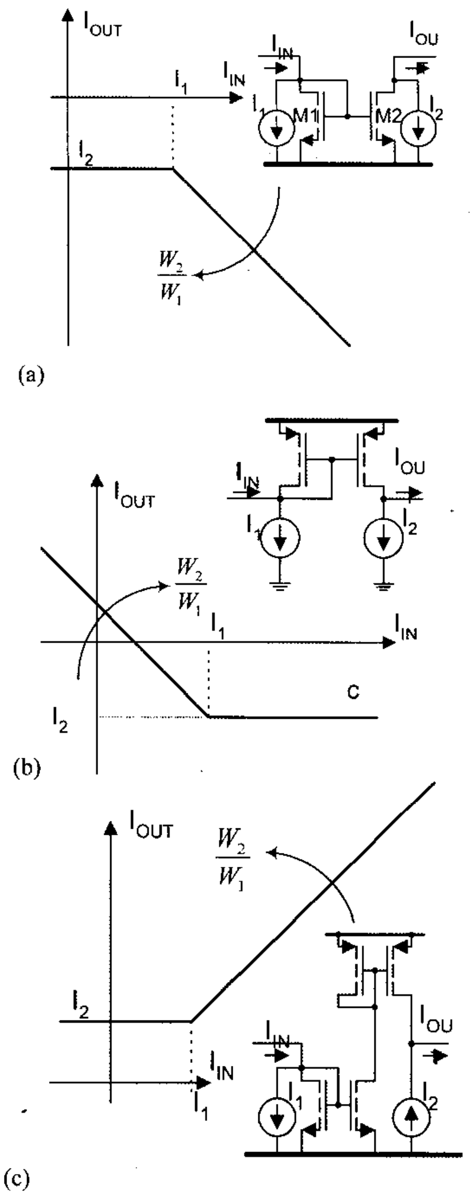


Fig. 15. Current mirrors and nonlinear processing units [29] for piecewise approximations: (a) using NMOS transistors, (b) using PMOS transistors, (c) combining NMOS and PMOS mirrors.

VI. CONCLUSION

Nonlinear characteristics of semiconductor devices in VLSI circuits were used to synthesize arbitrary nonlinear functions. Several ways of nonlinear signal processing were proposed including neural networks, fuzzy systems, and application specific synthesis. The later one includes piece-wise approximations.

References

[1] S. Deutsch and A. Deutsch, *Understanding the Nervous System: An Engineering Perspective*, IEEE Press, Piscataway, NJ, 1993.

[2] M. L. Padgett, P. J. Werbos, and T. Kohonen, *Strategies and Tactics for the Application of Neural Networks in Industrial Electronics*, CRC Handbook for Industrial Electronics, J. D. Irwin Ed., CRC Press and IEEE Press, pp. 835-857, 1997.

[3] Zurada, J., *Introduction to Artificial Neural Systems*, West Publishing Company, 1992.

[4] Wilamowski, B. M., R. C. Jaeger, and M. O. Kaynak, "Neuro-Fuzzy Architecture for CMOS Implementation" *IEEE Transaction on Industrial Electronics*, vol. 46, No. 6, pp. 1132-1136, Dec. 1999.

[5] Special Issue on New Trends in Neural Network Implementations in *International Journal of Neural Systems (IJNS)* Vol. 10, No. 3, June, 2000.

[6] Computational Intelligence Imitating Life edited by J. Zurada and others, IEEE Press 1994

[7] Silicon Implementation of Pulse Coded neural Networks edited by M. Zaghloul and others, Kluwer Academic 1994.

[8] Analog VLSI Signal and Information Processing edited by M. Ismail and T. Fiez, McGraw-Hill 1994

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

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

[11] 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

[12] B. Wilamowski, "Neural Network Learning and Architectures" tutorial at *ICONIT/IFSA Multiconference* Istanbul, Turkey, June 30, 2003.

[13] B. Wilamowski and Jeremy Binfet " Microprocessor Implementation of Fuzzy Systems and Neural Networks ", *International Joint Conference on Neural Networks (IJCNN'01)*, pp. 234-239, Washington DC, July 15-19, 2001.

[14] B. Wilamowski, J. Binfet, and O. Kaynak, "VLSI Implementation of Neural Networks," *International Journal of Neural Systems (IJNS)* Vol. 10, No. 3, pp. 191-197, June, 2000.

[15] Xin Yao, "Evolving artificial neural networks," *Proceedings of the IEEE*, vol. 87, no. 9, 1999, pp. 1423-1447.

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

[17] Yamakawa T., "A Fuzzy Inference Engine in Nonlinear Analog Mode and its Application to a Fuzzy Logic Control," *IEEE Trans. on Neural Networks*, vol. 4, no. 3, pp. 496-522, May 1993.

[18] Baturone I., A. Barriga, and J. L. Huertas, "Multi-input Voltage and Current-Mode Min/Max Circuits," *Proceedings: 3rd Int. Conf. on Fuzzy Logic, Neural Networks and Soft Computing*, Iizuka, Japan, 1994.

[19] Ota Y. and B. Wilamowski, Current-Mode CMOS Implementation of a Fuzzy Min-Max Network, *World Congress on Neural Nerworks* vol. II, pp. 480-483, 1995.

[20] Ota Y., B.M.Wilamowski, Analog Hardware Implementation of a Voltage-Mode Fuzzy Min-Max Controller, *Journal of Circuits, Systems, and Computers*, Vol. **6**, No.2, pp. 171-184, 1996.

[21] Choi J., B. J. Sheu, and J. C.-F. Chang, "A Gaussian Synapse Circuit for Analog VLSI Neural Networks,"

[22] Ramirez-Angulo J., A BiCMOS Universal Membership Function Circuit with Fully independent, Adjustable Parameters, *IEEE International Symposium on Circuits and Systems*, Seattle WA, vol. I, pp. 275-278, April 30-May 3 1995.

[23] Ahmadi S., L. Sellami, and R. W. Newcomb, A CMOS PWL Fuzzy Membership Function, *IEEE International Symposium on Circuits and Systems*, Seattle WA, vol. 3, pp. 2321-2324, April 30-May 3 1995

[24] Angulo J.R., (1995) A BiCMOS Universal Membership Function Circuit with Fully Independent, Adjustable Parameters, *Proc. IEEE International Symposium on Circuits and Systems*, pp. 275-278.

[25] N. Dharia, J. Gownipalli, B. Wilamowski, and O. Kaynak "Multidimensional Second Order Defuzzification Algorithm (M-SODA)" Multidimensional Second Order Defuzzification Algorithm (M-SODA)" *Proc. of the 28th Annual Conference of the IEEE Industrial Electronics Society – IECON 2002*, pp. 3280-3284, Sevilla, Spain, Nov 5-8, 2002.

[26] N. Dharia, J. Gownipalli, O. Kaynak and B. Wilamowski, " Fuzzy Controller With Second Order Defuzzification Algorithm," *The 2002 IEEE World Congress on Computational Intelligence: WCCI, 2002, International Joint Conference on Neural Networks: IJCNN 2002*, May 12-17, 2002, Honolulu, USA, pp.2327-2332, 2002.

[27] Ota, Y. and B. Wilamowski, "Analog Implementation of Pulse-Coupled Neural Networks", *IEEE Transaction on Neural Networks*, vol 10, No 3, May 1999, pp. 539-544.

[28] J. Johnson, M. L. Padget, O. Omidvar " Guest Editorial Overview of Pulse Coupled Neural Networks (PCNN) Special Issue", *IEEE Trans. on Neural Networks* vol. 10, no3, pp. 461-463, May 1999.

[29] B. Wilamowski, E. Ferre-Pikal, and O. Kaynak, "Low Power Current Mode CMOS circuits for synthesis arbitrary nonlinear functions" *9the NASA Symposium on VLSI Design, Albuquerque*, New Mexico, pp. 7.3.1-7.3.8, November 8-9, 2000.