ANNSyS (An Analog Neural Network Synthesis System)

İ. Bayraktaroğlu*, A. S. Öğrenci#, G. Dündar#, S. Balkır#, E. Alpaydın†
(*) Dept. of Computer Science and Engineering, UC San Diego, California -USA
(#) Electrical and Electronic Engineering Dept., Boğaziçi University, 80815 Bebek -İstanbul, Turkey
(†) Computer Engineering Dept., Boğaziçi University, 80815 Bebek -İstanbul, Turkey

Abstract

We present an Analog Neural Network Synthesis System based on a circuit simulator and a silicon assembler for neural networks. The circuit simulator makes use of the fact that neural networks with multilayer perceptron architecture consist of many decoupled blocks if the blocks are designed in MOS technology. We implement on-chip training on the software by incorporating the Madaline Rule III into our simulator. The assembler generates the layout by reading the standard cells from a library once the architecture of the network is given.

1. Introduction

Neural networks have found many applications in recent years. Research on neural networks started before the evolution of powerful computers. Neural networks are used when there is no algorithmic solution to a problem or a problem is too complicated to be solved by known algorithms [1]. Also, neural networks can be used when the definition of the problem does not exist, but samples of inputs and corresponding outputs are available. Applications of neural networks in general can be divided into two classes: Pattern recognition, and function approximation [1]. Most of these applications need real time processing. Neural networks are inherently parallel processors. When we implement a neural network with conventional computers, it has to be implemented in a serial manner. Besides this, the complexity of operations required in neural networks makes it impossible to use neural network simulators in many time critical applications. However, hardware implementations of neural networks can easily be made parallel. All parts of a neural network can be implemented in hardware. There are different approaches to hardware implementations. These are digital, analog, and mixed signal implementations. In mixed signal implementations the neural network is generally realized in analog hardware, whereas the inputs and outputs of such implementations are digital to enable easy interfacing with digital computers [2].

Analog implementations of neural networks have many advantages such as small size and high speed. Synapses, which are the most common elements in a neural network, can be represented at the circuit level by multipliers. For instance, parallel digital multipliers of 8x8 input word lengths have transistor counts on the order of at least several thousand [3]. Analog multipliers of comparable precision use less than 20 transistors. The speed of an analog multiplier is limited only by its bandwidth and can go up to the GHz range. When one looks at neurons, a similar picture can be seen. Again, an adder and the nonlinearity can be realized by less than 20 transistors, whereas the same operations require thousands of transistors in the digital domain [4]-[7].

However, analog neural network implementations have been rather *ad hoc* in that very few, if any, have explored the constraints that circuit non-idealities — like nonlinear synapses [8,9], neurons that deviate from ideal functions, or errors and limitations in storing weights [10] — bring about. It has been shown in [8] and [9] that multiplier nonlinearity can be a very severe problem even for nonlinearity factors of less than 10% for many applications. Limited precision in storing weights has also proven to be a crucial problem in analog neural network design. The work in this area has been mostly limited to predicting these effects either through simulation or through theoretical analysis and developing some methods to overcome these problems partially. In [9], the effects of some non-idealities have been studied through circuit simulation with SPICE and the importance of circuit level simulation in analog neural network design has been demonstrated.

Although SPICE is the standard tool for circuit simulation, it is not specially tailored for simulating neural networks. Neural networks consist of the interconnection of many identical blocks so that partitioning the network during simulation will increase the simulation speed tremendously.

Different approaches are used to obtain the weights of an analog neural network which also dictate the

This work is supported by TÜBİTAK under grant number EEEAG-183.

0-7803-4122-8/97 $10.00©1997 IEEE

implementation style and the architecture of the network [1]. These approaches are:

- Non-learning network: In this method, the weights are hardwired through the implementation of the fixed gain multiplier. In this case, the weights to be hardwired must be calculated before the operation of the neural network. The calculations are done on a computer which uses the model of the analog neural network. The performance of this method depends heavily on the matching between the model and the real circuit, which is a task that is very difficult to achieve.
- Neural-networks in analog hardware implementation with externally adjustable weight construction: For this realization, the weights are again computed on a host computer and downloaded to the chip. Then, the weights are fine tuned. The chip is used for forward pass, host computer is used for feedback (weight adaptation). By this way, the matching of the model and analog hardware is considerably increased.
- Neural network with on-chip learning: In this scheme, both the feedforward structure and all circuitry required to adapt the weights are realized on the chip. A major disadvantage of this approach and the previous one is that they both require additional hardware which is used only at the training stage.

One commercial implementation of analog neural networks is the ETANN 80170NX chip [5]. This chip has been plagued by limited resolution in storing the synapse weights in that the long time resolution of the weights is not more than five bits. Implementing Madaline Rule III [11] has been suggested for the ETANN chip; however, this requires a host computer and excessive external hardware besides many timing problems which limit the performance of training. Problems like these have prevented the success of this chip in the market so that commercial applications using this chip and similar ones have been very few.

Outline of the paper is as follows. Section 2 introduces the Analog Neural Network Simulation System (ANNSiS) which is based on circuit partitioning techniques. This simulation system is used to simulate neural networks composed of the building blocks discussed in section 3. SAFANN (Silicon Assembler For Analog Neural Networks) is examined in section 4. Training using Madaline Rule III on ANNSyS is proposed in section 5 whereas section 6 concludes the paper.

2. Analog Neural Network Simulation System (ANNSiS)

Most of the neural networks used for pattern recognition or function approximation applications are of the multi-layer perceptron structure. Determination of the weights is performed prior to operation, at the learning phase, which in general, requires the adaptation of weights iteratively to reduce the error (distance between the actual output and the desired output of the network) to zero. Different algorithms can be applied during the learning phase. The most commonly used and well known algorithm is the backpropagation algorithm [11].

Feedforward multilayer neural networks are regular structures where every neuron is connected to every other neuron in the previous layer through synapses. Therefore, they yield themselves easily to partitioning and automatic netlist generation.

Analog neural networks are specially designed to be used in real time applications. The speed of analog neural networks is a very important issue during the design process. However, the feedforward structure and the regularity of the neural network allows the determination of the overall speed of the network easily. If the building blocks of the neural network are designed carefully to give a fast response, the overall structure is assured to fulfill the desired response time. This means that, for most applications, the DC transfer characteristics of the neural networks will be the important issue for the designer. Therefore, DC analysis will be sufficient for most cases. A simulator designed specifically for this purpose can be applicable to analog neural networks.

The most commonly used tool for circuit simulation is SPICE. However, the size of a neural network circuit for a practical example is very large to be simulated with SPICE. The simulation time of SPICE for neural network circuits increases almost quadratically with the circuit size. Besides, when the circuit size is increased beyond a limit, SPICE starts to have difficulties in simulating the network. This problem can be solved by using partitioning techniques. For DC analysis, if the layers are completely decoupled, by which we mean that the outputs of the neurons are not loaded by the inputs of the synapses of the next layer, the circuit can be partitioned into decoupled blocks which can be simulated separately starting from the input layer. Most of the neural network implementations use CMOS technology. Considering this technology, the assumption of being completely decoupled holds and allows us to partition the network.

ANNSiS is initiated by simulating all partitions in the first layer and finding the outputs. The output values of the first layer are then applied to the next layer as independent voltage sources being input to the synapses of the neurons in that layer. This way, the input is propagated to the output.

We first created different sized analog neural network structures and simulated them without partitioning. Next, we simulated them by partitioning into blocks which consist of a neuron and all synapses connected to that neuron. Simulation results of SPICE2G6 and ANNSiS are compared for accuracy and found out to be exactly matching. Table 1 shows the sizes, CPU times, and the

Table 1. Simulation results.

				SPICE2G6		ANNSiS Without partitioning		ANNSiS With partitioning	
Structure	synapses +opamps	Fets	Nodes	Memory (k)	Time (sec)	Memory (k)	Time (sec)	Memory (k)	Time (sec)
2x1	3	49	148	672	3.6	448	0.8	560	0.8
2x2x1	9	147	426	908	13.3	596	2.7	812	1.7
2x3x1	13	213	611	980	19.6	672	4.3	956	2.4
2x4x1	17	279	796	1040	26.3	748	6.6	1084	3.0
2x5x1	21	345	981	1100	31.9	820	8.8	1208	3.7
2x5x2	27	445	1258	1212	49.3	928	14.5	1400	5.1
2x5x3	33	545	1535	*	*	1028	20.6	1564	6.3
2x5x4	39	645	1812	*	*	1128	26.6	1696	7.0
2x5x5	45	745	2089	*	*	1264	32.3	1912	8.8
4x8x7	103	1721	4766	*	*	2136	128.0	3112	19.4

* SPICE2G6 did not converge up to the predefined number of iterations of SPICE.

memory requirements for different structures for simulations performed on SunSPARC2 workstations.

When the analog neural network circuit was partitioned, the simulation time decreased remarkably. As seen in Figures 1 and 2, the simulation time increases almost linearly for the partitioned case and almost quadratically for the non-partitioned case. Remarkable decreases in simulation time show the effectiveness of partitioning. Thus, it is possible to simulate large neural network circuits in a faster manner and without any convergence problems.

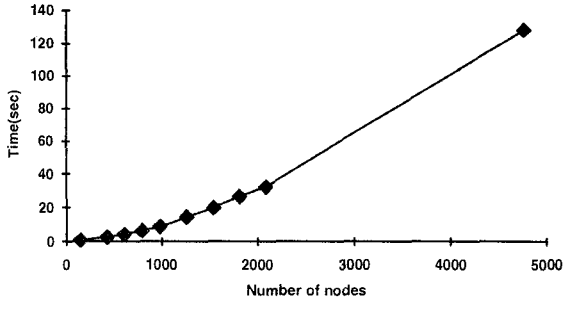

Figure 1. Simulation time for different MLP structures(without partitioning).

3. Neural Network Circuitry

Figure 3 shows the circuit used as a synapse. This circuit is a modified version of the well known Gilbert multiplier [12,13]. The inputs are in the form of voltage differences and are denoted by the couples $x_1 - x_2$ and $y_1 - y_2$. The output of the original Gilbert multiplier is a current difference and this difference is converted to a single ended current (Z) through current mirrors. This improves the linearity of the multiplier as well as providing easy interfacing to the following circuitry. The output voltage characteristics of the synapse circuit is shown in Figure 5.

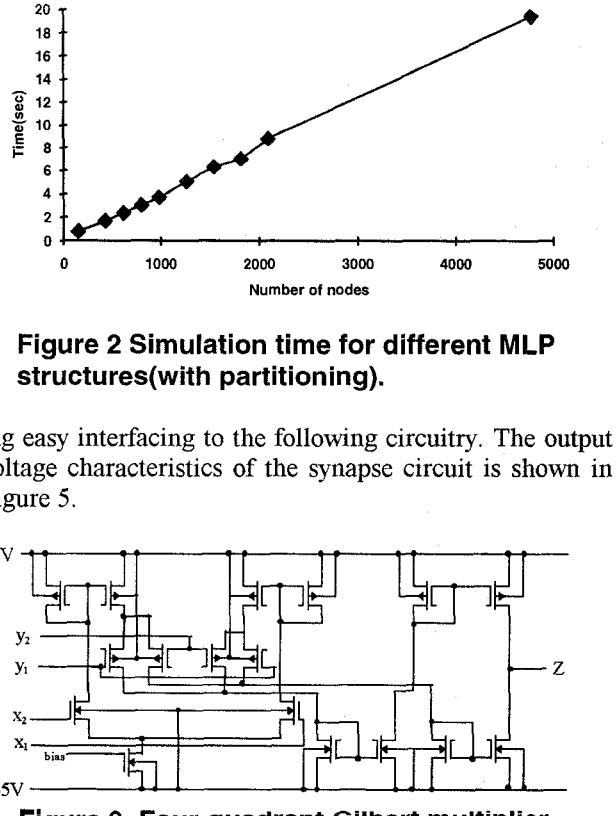

Figure 2 Simulation time for different MLP structures(with partitioning).

Figure 3. Four-quadrant Gilbert multiplier.

Figure 4 shows an OPAMP that can be used to sum the outputs of all the synapses connected to the neuron and convert the current sum into voltage. This OPAMP consists of two stages, where the first stage is a differential amplifier whose differential current output is mirrored into the next stage and converted to a single ended output through circuitry very similar to the synapse

circuit above. A sigmoid generator introduced in [14] is used after the OPAMP to generate the activation function for the neuron. This generator is depicted in Figure 6, and the characteristics are plotted in Figure 7.

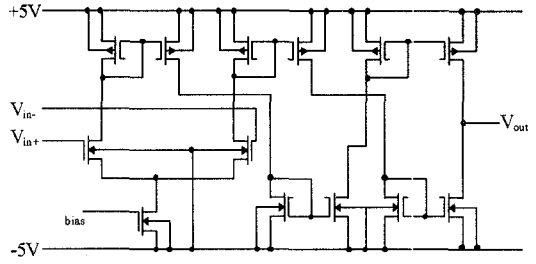

Figure 4. General purpose OPAMP.

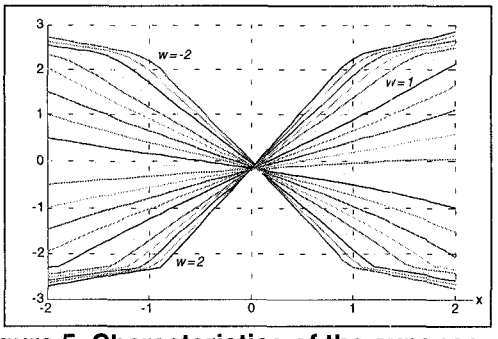

Figure 5. Characteristics of the synapse.

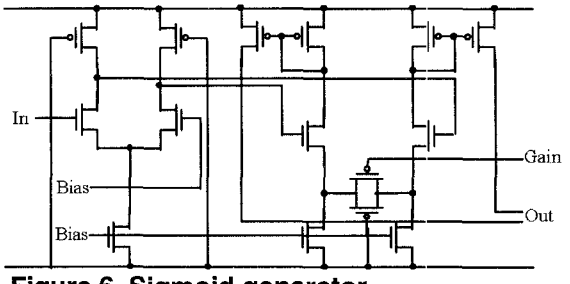

Figure 6. Sigmoid generator.

3. Silicon Assembler

The automatic generation of the analog neural network layout is possible if there are suitably designed synapses and neurons. These basic blocks (subcells) can be placed in arrays and the complete circuit layout will be obtained. Once the *cif* (*Caltech Intermediate Form*) files for the building blocks are available, the final layout can be obtained as a *cif* file by appending that placement information to the *cif* definitions of the blocks. As part of this study, a silicon assembler (SAFANN) is developed to accomplish this task.

The starting point is the layout for a single cell which consists of three types of subcells and some interconnections. A sample 3-input cell structure is given in Figure 8. It is mainly made up of 3 multipliers, a neuron,

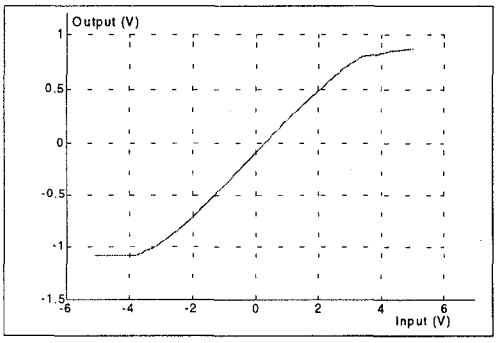

Figure 7. Characteristics of the sigmoid generator.

and 3 of the so called channels. In fact, each channel consists of 3 subchannels employing 2 lines each. This type of a topology has its reasons: First of all, the design has to be modular; that is, it should support any number of inputs. Hence, the weights for the inputs are carried on channels whose number can easily be manipulated. It should also be noticed that only one weight; i.e., 2 weight lines should be connected to each multiplier, so that a decoding scheme is necessary. Next, the input lines have to travel throughout the cell in horizontal direction because that input will also be required in the next cell placed to the right of the first one. Finally, the output lines of the multiplier have to be aligned such that they are common in vertical direction so that they will be connected together which is also part of the abutment property. The supply and bias lines will also run through the cells for alignment.

SAFANN achieves the automatic placement of the layout building blocks and routing between them by placing instances of symbols created for those blocks, namely the multiplier, neuron and subchannel; and routing channels as collection of boxes into a *cif* file. The structure of this methodology can be easily understood by

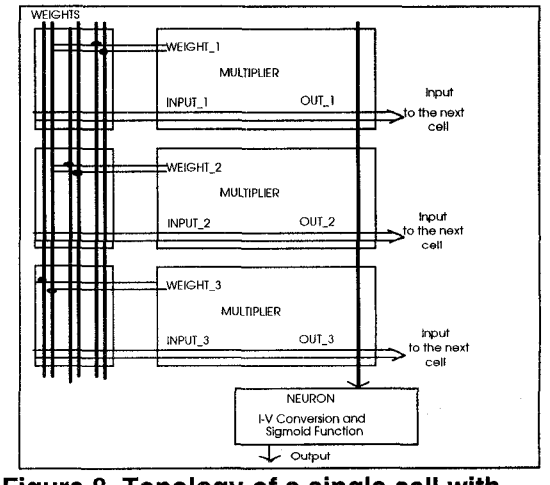

Figure 8. Topology of a single cell with three inputs.

considering a single layer of an ANN given in Figure 9. Here, the channels will be described by collection of boxes representing metal-1, metal-2 lines and vias. Neuron cells, however will be called by the symbols. Figure 10 presents the layout of a test chip containing 4 neurons (cell of Figure 8) with 5 inputs for each. This chip is generated by SAFANN and it contains the building blocks explained in section 2. The chip is sent to foundry for fabrication in a 2μ technology.

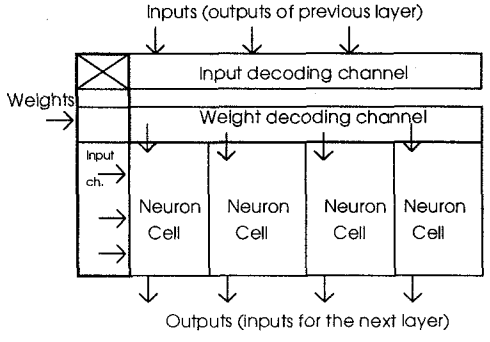

Figure 9. Topology of a single layer with three inputs and four neurons.

5. Training

There are three different approaches to the hardware implementation of neural networks as explained in the introduction. The non-learning implementation requires minimal chip size, whereas, due to the mismatches between the models and real circuitry, its performance is the worst one, and the on-chip learning implementation gives the best performance. However, if the model of real circuitry perfectly matches the real neural network, the non-learning implementation will be equivalent to on-chip training. An almost perfectly matching model can be the SPICE model of the analog network. Having developed a tool for simulating analog neural networks with SPICE models, Madaline Rule III [11] was chosen to fine tune the weights obtained from the backpropagation algorithm which uses models for the neurons and synapses.

Although we developed a method for the simulation of analog neural networks, it still takes considerable amount of time to simulate them. Therefore, before starting Madaline Rule III, an approximate starting point for the weights should be found. Approximate determination of weights is done using the modified backpropagation algorithm, which incorporates the approximate models of the synapses and neurons. We modeled the synapses as polynomial functions with errors of less than 1%. The sigmoids are modeled in a more general form of sigmoid function, given as $f(x) = A + \frac{B}{1+e^{(Cx+D)}}$. The model we used has a mismatch below 5%.

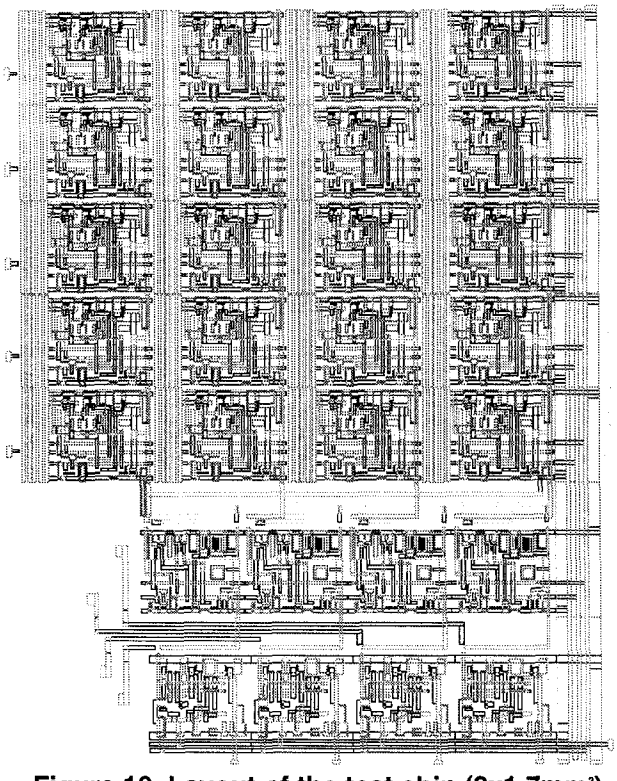

Figure 10. Layout of the test chip (2x1.7mm²)

The approximate weights are calculated on a computer with backpropagation algorithm and these weights are downloaded to the SPICE representation of the network to start Madaline Rule III. Madaline Rule III is an algorithm that is used when the backpropagation algorithm cannot be applied due to the impossibility of finding the derivatives of error with respect to the weights. Instead of using the derivatives, a small amount of disturbance is added to the summation of the synapse outputs and the difference in the error is measured. Dividing the error difference by the disturbance, we find the derivative of the error with respect to the input of the neuron if the disturbance is small enough. Then, assuming linear neurons and using the chain rule, one can find the derivative of error with respect to the weights. However, if the neurons are not perfectly linear, this method will not produce the real derivatives. Another method could be adding disturbance to the weights and then finding the derivative with respect to the weights directly. However, this will increase the simulation time considerably. Thus another method was devised: assuming that the synapses are linear again, we applied disturbances to all the weights, which indeed produces a disturbance at the input of the neuron. The simulations show that this will improve the convergence properties of Madaline Rule III. This method is Madaline Rule III with a minor modification.

The method is tested on two examples, the classical XOR problem and a sine function generator. For the XOR problem, a 2x3x1 structure was used, while a 1x10x1 structure was employed for the sine generator. The weights are calculated via modified backpropagation using the models for the synapse and neuron circuits and the error decreased below 1% for both cases. At this time, the network was simulated by ANNSiS and the error was found to be 13% for the XOR example and 30% for the sine generator example. Figure 11 shows the sine approximation result of the neural network after modified backpropagation algorithm. These errors were larger than the errors estimated by the backpropagation algorithm. This shows that, even in simple examples, small model mismatches (less than 5%) can create big problems. After that we applied Madaline Rule III for 50 epochs and the error obtained using the simulator decreased below 1% for the XOR case and below 3% for the sine generator case. Figure 12 shows the sine function obtained from training on ANNSyS.

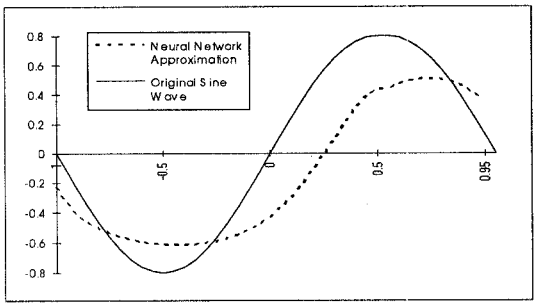

Figure 11.Neural network approximation of sine wave after modified backpropagation.

6. Conclusion and Future Work

An Analog Neural Network Synthesis System (ANNSyS) was developed. This package consists of an Analog Neural Network Simulation System (ANNSiS), a Silicon Assembler For Analog Neural Networks (SAFANN), a function approximator for synapses and neurons, a modified backpropagation algorithm, and a circuit level neural network trainer using modified Madaline Rule III as well as a control shell. ANNSiS is based on circuit partitioning techniques and has superior performance compared to other circuit simulators for simulating analog neural networks. ANNSyS has been applied to a number of neural network problems one of which has been illustrated in this paper and excellent results have been obtained. As for the future work, ANNSyS will be applied to real life examples and the fabricated chip will be tested for performance evaluation.

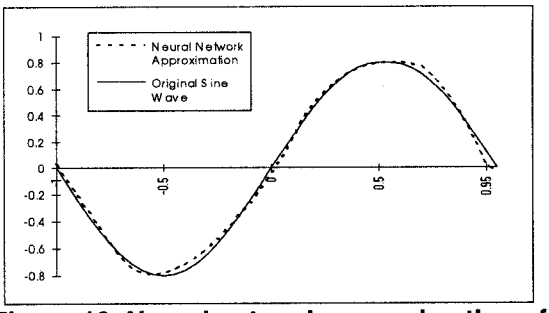

Figure 12. Neural network approximation of sine wave after Madaline Rule III

References

[1] A-J. Annema, *Feed-forward Neural Networks, Vector Decomposition Analysis, Modelling and Analog Implementation*, Kluwer Academic Pub., Boston 1995.

[2] C.S. Lindsey and T. Lindblad, Plenary Talk given at the Third Workshop on Neural Networks: From Biology to High Energy Physics, Marciana Marina, Isola d'Elba, Italy, Sept. 26-30, 1994.

[3] H. Binici, G. Dündar, and S. Balkır, "A new multiplier architecture based on radix-2 conversion scheme," *Proceedings of ECCTD '95*, pp 439-442, İst., 1995.

[4] P. Treleaven, M. Pacheco, and M. Vellasco, "VLSI architectures for neural networks," *IEEE Micro. Mag.*, pp 8-27, Dec 1989.

[5] Intel 80170NX ETANN Data Sheets, Feb. 1991.

[6] O. Rossetto *et. al.*, "Analog VLSI synaptic matrices as building blocks for neural networks," *IEEE Micro. Mag.*, pp 56-63, Dec 1989.

[7] G. Dündar and K. Rose, "Analog neural network circuits for ASIC fabrication," *Proc. of the 5th. IEEE ASIC Conference*, Rochester, 1992, pp 419-422.

[8] G. Dündar, F-C. Hsu, and K. Rose, "Effects of nonlinear synapses on the performance of multilayer neural networks," *Neural Computation*, Vol.8, No:5, pp. 939-949, July 1996.

[9] A. Şimşek, M. Civelek, and G. Dündar, "Study of the effects of nonidealities in multilayer neural networks with circuit level simulation," *Proc. of Melecon'1996*, Bari, Vol.1, pp. 613-616, May 1996.

[10] G. Dündar and K. Rose, "The effects of quantization on multilayer neural networks," *IEEE Trans. on Neural Networks*, Vol.6, No. 6, pp. 1446-1451, Nov. 1995.

[11] B. Widrow and M. Lehr, "30 years of adaptive neural networks: Perceptron, Madaline, and Backpropagation," *Proc. of IEEE*, Vol.78, No:9, pp. 1415-1442, Sept. 1990.

[12] B. Gilbert, "A precise four-quadrant multiplier with subnanosecond responce," *IEEE Journal of Solid State Circuits*, Vol. 3, pp. 365-373, 1968.

[13] F.J. Kub *et. al.*, "Programmable analog matrix multipliers," *IEEE Journal of Solid State Circuits*, Vol. 25, pp. 207-214, 1990.

[14] T. Shima *et. al.*, "Neurochips with on-chip backpropagation and/or Hebbian learning," *IEEE Journal of Solid State Circuits*, Vol. 27, pp. 1868-1876, Dec. 1992.