A collection of Fuzzy Logic–Based Tools for the Automated Design, Modelling and Test of Analog Circuits

A.Torralba, J. Chávez, L. G. Franquelo

Dpto. de Ingeniería Electrónica

Escuela Superior de Ingenieros,

Avda. Reina Mercedes s / n, Sevilla–41012 (SPAIN)

e–mail: torralba@gtex02.us.es

Abstract

Analog design is quite more difficult than digital design. As a consequence, most of the effort in analog design still relies on designer's experience. New ideas and methodologies are required to make a progress towards automating the analog design. Computational intelligence techniques, such as those based on fuzzy logic, have provided excellent results in those fields traditionally reserved to human experts. This paper presents a collection of fuzzy logic–based tools which find an application in the three main design phases of the analog design: topology selection, modelling and optimization, and testing.

I. Introduction

The increasing demand of analog and mixed–signal application specific circuits (ASICs) has attracted the interest of the researchers towards the analog field. However, despite the great effort which has been devoted to the development of CAD tools for analog circuits, the design and test of the analog part of a circuit is still the most time–consuming task [1]–[2].

The difficulty in the analog field arises from the complex dependencies that exist between parameters and circuit specifications. Therefore, the design and test of analog circuits mainly relies on designer's experience.

It is in this kind of problems where fuzzy logic–based techniques have found an application. As stated in [3], fuzzy logic is a natural way of expressing the approximate, inexact nature of the human knowledge. This makes fuzzy logic a good candidate to aid in the analog design field.

This paper presents a collection of tools for the design, modelling and test of analog circuits. These tools share a common fuzzy logic–based framework, and they are part of FASY, an analog design package developed at the University of Seville.

The paper is organized as follows. Section II presents the fuzzy logic–based common framework. Section III presents an analog design tool which uses fuzzy logic in the topology selection process. In section IV fuzzy logic is used to model circuit performances. In section V a tool for the automatic testing of analog circuits using a neuro–fuzzy classifier is described. In each section, a brief description of the tool is made and experimental results are presented. Finally, in section VI some conclusions are drawn.

II. A Fuzzy Logic–Based Common Framework

Figure 1 shows the basic configuration of a fuzzy system. It has four principal components:

1. The *fuzzification interface*, which is a mapping from the input data to *fuzzy sets*. A *fuzzy set* U is characterized by a *membership function* $\mu^F : U \rightarrow [0,1]$. Membership functions can be labelled by a linguistic term F, such as "small","medium" or "large". The most commonly used membership functions are triangular, trapezoidal and gaussian.

2. The *fuzzy rule base*, which is a set of *fuzzy rules* in the form of *IF–THEN* clauses:

$$R_j : IF \;\; x_1 \; is \; A_{1j} \;\; and \;\; x_2 \; is \; A_{2j} \;\; and \;\; ... \;\; and \;\; x_n \; is \; A_{nj} \;\; THEN \;\; y \; is \; B_j \tag{1}$$

3. The *fuzzy inference machine*, which is a decision–making logic which employs the rules of the fuzzy rule base to produce a fuzzy output. The most commonly used inference mechanisms were proposed by Mamdani and Sugeno. A comprehensive review of the different kinds of fuzzy inference machines can be found in [4].

4. The *defuzzification interface* which produces a nonfuzzy action from the inferred fuzzy output. The most commonly used defuzzification method is the *centroid method*, which computes the center of gravity of the membership function of the inferred fuzzy output.

Figure 2 depicts a graphical representation of the principal components of the fuzzy mechanism used in this paper. Gaussian shaped membership functions have been selected for the input data

$$\mu_{A_{ij}}(x_i) \;\; = \;\; exp \;\; [\;\; -\frac{1}{2} \;\; (\frac{x_i \;\; - \;\; c_{ij}}{\sigma_{ij}})^2 \;\;] \tag{2}$$

Singletons are used for the output variable.

Fuzzy implication uses the Larsen's product operation rule. The connective and is implemented by means of the algebraic product. Fuzzy rules are combined by means of the sentence connective $also$, which is implemented by means of the algebraic addition. With this selection of parameters, an expression for rule j implication is given by:

$$\mu_{B'_j} = (\prod_{antecedents} \mu_{A_{ij}}(x_i)) \;\; \mu_{B_j} = \phi_j \;\; w_j \tag{3}$$

where w_j is the position of the singleton corresponding to the j–th rule.

The defuzzification process is carried out using the centroid method. As the membership functions of the output variable are singletons, the center of gravity of the inferred fuzzy set can be obtained by means of the expression

$$y = \frac{\sum_1^m \phi_j \ w_j}{\sum_1^m \phi_j} \tag{4}$$

A. Learning

Recently, it was proven that fuzzy systems are able to approximate any real continous function on a compact set to arbitrary accuracy [5]–[7]. Fuzzy systems can be trained to match unknown nonlinear maps. Different training procedures have been proposed in the literature, such as backpropagation [8], competitive learning [9], the orthogonal least square algorithm [7] and clustering [11].

The backpropgation technique has been selected here [12]:

1. Generate a list of input–output pairs in the map to be learnt.
2. Choose an input–output pair ($\mathbf{x}$,y^d) from the list to be learnt. $\mathbf{x}$ is the input vector and y^d the desired output.
3. Compute intermediate values ϕ_j, and the output y, corresponding to the input vector $\mathbf{x}$ (forward process).
4. Update parameters according to the following equations (backward process):

$$\Delta w_j = \eta_w \ [y^d - y] \ \frac{\partial y}{\partial w_j} \tag{5}$$

$$\Delta c_{ij} = \eta_c \ [y^d - y] \ \frac{\partial y}{\partial c_{ij}} \tag{6}$$

$$\Delta \sigma_{ij} = \eta_\sigma \ [y^d - y] \ \frac{\partial y}{\partial \sigma_{ij}} \tag{7}$$

5. Return to step 2 until

$$Error = \frac{1}{2} \sum_{learning \ \ points} (y^d - y)^2 \ < \ \epsilon \tag{8}$$

or until a predefined number of iterations is exceeded.

The derivatives $\partial y/\partial w_j$, $\partial y/\partial c_{ij}$ and $\partial y/\partial \sigma_{ij}$ can be obtained using the chain rule. η_w, η_c and η_σ are learning rate parameters.

III. A FUZZY LOGIC–BASED TOOL FOR ANALOG SYNTHESYS

Circuit design can be considered as a two step process: the designer first selects a circuit topology among a set of fixed alternatives and then optimizes the values of circuit parameters to meet a set of specifications.

Existing tools have been mainly focused in the parameter optimization step. The optimization process is converted into a minimization problem by means of the formulation of a cost function. Different techniques have been used to minimize the cost function: the steppest descendent method was used in OPASYN [13]. The method of *feasible directions* in DELIGHT.SPICE [14]–[15]. Sequential quadratic programming in [16]. Statistical techniques, such as simulated annealing [17]–[19] or tabu search [20] have been also used.

On the other hand, the topology selecttion process is usually left to the designer's experience. Existing references deal with the topology selection process by means of artificial intelligence techniques [13], or hierarchical decomposition [21]–[22].

This section describes a design system for analog synthesys which uses fuzzy–logic in the topology selection process. Afterwards, an optimizer which is described elsewhere [19], sizes all the components of the selected topology, to minimize an user defined cost function.

A. System Architecture

The architecture of the proposed design system is shown in figure 3. Two data flows are pointed out in this figure. In the top–down flow (called *design flow*), it acts as a design tool for analog cells. In the bottom-up flow (called *learning flow*), the topology selection rules are learnt from the experience gained with the optimizer. In a design, some performance specifications are defined by the designer. Table 1 shows an example of the specification set for operational amplifiers. The topology selection module uses fuzzy decision rules to select the circuit topologies that best accommodate the set of performance specifications. Selected topologies are displayed along with a grade from 0 to 100, indicating their relative suitability to performance specifications. The designer chooses one of them or allows the system to choose the highest rated topology. Then, the selected topology is given to the circuit parameter optimizer. If the final design is accepted, it is stored in the internal data base. The experience gained with these successful designs can be used to modify the topology decision rules. This process is called *learning rules process* and can be automatically made by means of the backpropagation method above described.

The knowledge base can be directly obtained from an expert designer or automatically generated from the experience gained with earlier designs using the parameter optimizer that will be described in the following

section. Table 2 resumes a set of decision rules entered by a human expert to make a choice among three op.amp. topologies (figure 4). Following these rules, the three basic topologies are adequate when small dc gains and small unity gain frequencies are specified. For small and medium dc gains, the Basic Two Stage (BTS) op.amp. is preferred, while the Folded Cascode OTA op.amp. dominates in the case of large gains and frequencies. Note that these rules are defined using the natural language used by expert designers. Using the rules given and the inference process described in section II, the decision surfaces of figure 5a are obtained. These figures indicate the relative suitability of each topology to performance specifications, and are used in the topology decision process. This example has been deliberately made simple so that the results can be represented as simple decision surfaces.

Sometimes the topology decision rules cannot be acquired due to the lack of experience (e.g., in the case of new technology) or due to the existence of conflicting specifications. In those cases, the proposed system is able to automatically generate the decision rules by means of neuro–fuzzy techniques.

The automatic rules generation process carries out the following steps. The specifications space is clustered in an adequate number of cells, each one representing a set of user specifications. In the case of only two specifications, it leads to rectangular bi–dimensional cells. In the case of n specifications, it leads to n–dimensional cells. For each cell and each possible topology, an optimization process is carried–out using the circuit optimizer. The final value of the cost function is considered as a figure of merit of the related topology with this set of user specifications. It is convenient to repeat the optimization process several times, starting with randomly generated initial values of the system variables to obtain a result independent of the optimization procedure. Using this method, the decision surfaces of figure 5b have been obtained. When compared with the decision surfaces of figure 5a some interesting conclusions can be drawn. The general behaviour of these figures are similar, indicating that the experienced designer had a good qualitative feel for topology selection. There is a difference, whereas figure 5a represents the approximate reasoning of a human expert, figure 5b has been obtained as a consequence of an optimization process, and represents precise knowledge of circuit behaviour. Each surface in figure 5b was built from the results obtained with 100 designs. Technology files were taken from a commercial 1.5 μm CMOS process. It took approximately six hours on a 96 MIPs workstation to obtain the decision surfaces of figure 5b. Whereas it seems to be a time–consuming task, this process only needs to be carried out once for a given technology.

These surfaces can be stored as fuzzy decision rules by means of the learning process described in section II. Storing decision surfaces as fuzzy rules exhibits some advantages over a tabular representation. First of all, only a few decision rules are required for surface representation, saving memory. Secondly, fuzzy logic is a

structured way for knowledge representation and can be used by expert designers in future designs, and also in the evaluation of new incoming technologies. Finally, storing decision surfaces as fuzzy rules, allows the system to inherit some properties of fuzzy systems such as generalization, that is, the ability to obtain good responses when faced with new problems. After the learning process, the decision surfaces do not significatively differ from the surfaces of figure 5*b*. Figure 6 shows a view of the decision surfaces superimposed after normalization in the range [0..100].

As knowledge is represented by means of the exact shape of a set of membership functions and fuzzy rules, changing from one technology to another, only requires replacing them with a new set of membership functions and fuzzy rules, maintaining the structure of the decision making process.

Note that the technique used for the topology selection of operational amplifiers can also be used for other analog cells, and in the selection step of each level of a hierarchical design tool, such as OASYS. Using fuzzy logic for topology selection, and the ability to learn decision rules in an automatic way, are characteristics of the proposed system which are believed to be unique when it is compared with other existing analog CAD tools.

IV. A Fuzzy Logic–Based Tool for Circuit Performance Modelling of Analog Circuits

The design process of an electronic circuit is an iterative process that requires a large number of circuit simulations to meet the user specifications. An efficient way to reduce the time spent in these simulations is to build models of the circuit performance functions. These models find different applications, such as circuit optimization, design centering, tolerance assignment, delay modelling of VLSI circuits, etc.

Simple models can be employed in describing non–complex circuits with linear behaviour. However, in many problems, it is not possible to determine such simple models, because the inherent nonlinearities of the circuit cannot be approximated accurately by elementary functions, or because the complexity of the circuit makes it exhibit a complex behaviour.

Differente approximation methods have been proposed, such as quadratic interpolation [23]–[24], the Group Method of Data Handling (GMDH) [25]–[26], quasi–Newton optimization techniques [27], and statistical techniques [28].

In [29], a composite method was proposed which combines the best characteristics of the Maximally Flat Quadratic Interpolation technique (MFQI) and the Group Method of Data Handling (GMDH).

In this section, the use of Fuzzy Logic is proposed for accurate modelling of circuit performance functions. Fuzzy Logic is attractive in nonlinear circuit modelling for a number of reasons. First of all, fuzzy logic is a natural way of expressing the approximate nature of the human knowledge about the circuit behaviour. Secondly, as stated in [5]–[7], fuzzy models are universal approximators, so that there are no restrictions in the kind of functions that fuzzy systems can model. Finally, the parameters of the model can be trained to fit a set of selected input–output pairs, by means of neurofuzzy techniques, such as *backpropagation*. Thus, fuzzy models can be used in an automatic adaptive process which, starting from the selected input–output pairs, produces a model of high accuracy at a low cost.

A. Modelling Procedure

This subsection describes the modelling procedure.

1. The reference circuit simulator (SPICE) is used to generate a set of base points, called the *training set*. The *Latin Hypercube* experimental design technique [30] is used to generate the base points.

2. The fuzzy model described in section II is employed to model the circuit performance functions of interest. Although it is possible to start from values of the model parameters (c_{ij}, σ_{ij}, w_j) obtained from the approximate knowledge of the circuit behaviour, the results reported here have been obtained starting

from a set of randomly chosen values of the parameters.

3. The parameters of the model are tuned in an automatic way, by means of the process described in section II,

Using this procedure, circuit performance modelling is a simple process. There are not complex transformations of the input variables. No decisions based on fitting are required either. The parameters of the model adapt in an automatic way, and no human intervention is required.

B. Examples

Two examples, analog and digital, have been used to test the accuracy of the modelling technique. The examples have been taken from the reference [29]. Nevertheless, the value of the circuit parameters (MOS transistors widths and lenghts), as well as the technology files are, in general, different. Hence, the results reported here can only roughly be compared to the results reported in [29].

Example 1. *Modelling Time Delay of a Three–Input NAND Gate*

Figure 7 shows the schematic of a three–input NAND gate. In this example, the performance function of interest is the NAND gate–delay. The circuit has 7 independent variables, which are considered to be as independent, uniformly distributed random variables, whose variation ranges are displayed in table 3.

Table 4 shows the results obtained with the proposed fuzzy model and a variable number of rules. 200 base points were used in the training set. They were obtained using HSPICE level 6, and the parameters of a commercial 1.5 μm CMOS technology.

It can be observed the accuracy of the model, which provides a small value of the average and maximum errors, even for a reduced number of rules.

However, a complete test of the model accuracy requires the use of a set of independent test points. To this purpose a new set of 50 uniformly distributed base points was generated using the Latin Hypercube experimental technique once again. This new set of base points is called the *checking set*.

The last three columns in table 4 show the standard deviation, and the average and maximum errors of the models with the checking set. As expected, these errors are higher than the ones corresponding to the training set, although they are small enough to warranty that no overfitting occurs.

Example 2. *Modelling a CMOS Linear Transconductance Amplifier (TCA).*

Figure 8 depicts the schematic of a CMOS Linear Transconductance Amplifier (TCA). It is assumed perfect matching between transistors, as required by the symmetry of the circuit. The independent variables are the

width of the MOS transistors M9 (M14), M10 (M13), and M11 (M12), as well as the bias current, I_{ref}. These variables are considered to be as independent, uniformly distributed random variables, whose variation ranges are shown in table 5.

The performance functions of interest are the average dc transconductance A_g and the nonlinearity error NL. To obtain the base points, 50 dc swing analyses were performed. For every analysis (figure 8), $V_{in} = (V_{in}^{+} - V_{in}^{-})$ ranges from -3.6 V to 3.6 V, in 21 steps of 36 mV each. The performance functions of interest are defined as follows:

1. Average dc trans. gain ($\mu A/V$)

$$A_g = \frac{1}{21} \sum_{i=1}^{21} \frac{I_o}{V_{in}^i} \tag{9}$$

2. Nonlinear error (%)

$$NL = max\{\frac{A_g - I_o/V_{in}^i}{A_g}\} \quad i = 1,..,21 \tag{10}$$

A_g is a well–behaved function which can be accurately approached by the fuzzy model, as shown in table 6. The average and maximum errors in the training set are small, even for a reduced number of rules. On the other hand, the $NLerror$, is a max function which is difficult to model, as can be seen in table 7. However, the small value of the average errors and the standard deviations show that the fuzzy model still accurately represents the circuit behaviour.

V. A Fuzzy Logic–Based Tool for Analog Testing

Fault detection and classification of analog circuits received a great deal of attention in the 70's and the early 80's, due to the increasing complexity of electronic circuits and the high realibility requirements in the spatial and military fields [31]–[32]. Recently, the extensive use of analog and mixed–signal ASICs, have attracted the interest of the research community towards analog and mixed–signal testing again [33]. A comprehensive review on the topic can be found in [31]–[32] and [34]. In [35]–[39] some fault detection and classification methods of analog circuits, which are directly related to this work, can be found.

In accordance with the classifications in [31], in this paper a new fault detection and classification method is proposed, which can be included among the *taxonometric* methods; that is, those which employ a *fault–dictionary* where the system reference responses, corresponding to each potential fault condition, are stored. This fault–dictionary is built before the on–line testing process, by means of a serie of Monte–Carlo simulations of the circuit, under nominal and faulty conditions. Thus, the proposed method can be also considered a *before–test* method. In the on–line process, the measured circuit responses are compared to the responses stored in the dictionary; the one which approximates the most, is selected following some predefined criterium.

The main contribution of this section is the proposal of a new fuzzy–neuron intended for classification problems, and its application to analog circuit testing. The proposed test system allows the recognition of arbitrarily shaped classification regions. It can be applied to test linear and non–linear circuits, with both parametric and functional tests. In addition, unlike other methods, no assumptions are made about the statistical distribution, or independence of the measurements.

A. A Fuzzy Neuron: Application to the Fault Detection of Analog ICs

Figure 9 shows a fuzzy–neuron specially intended for classification problems. The n–inputs of the neuron are rules of a fuzzy controller, whose inputs are, in turn, the circuit measurements.

Let $\mathbf{x} = (x_1, x_2, .., x_m)$ be the input set to the classifier (i.e. the circuit measurements). The input x_j in the rule i, is assigned a gaussian membership function $\mu_{ij}(x_j)$ following the model described in section II.

To perform as a classifier, the center of gravity defuzzification of equation (4) is replaced by a weighted sum and a sigmoidal function.

$$y = f\left(\sum_{i=1}^{m} w_i \ \phi_i\right) = \frac{1}{1+e^{-\sum_{i=1}^{m} w_i \ \phi_i}} \tag{11}$$

An analysis similar to that carried–out in [6], would demonstrate that this fuzzy–neuron is an *universal approximator* in the n–dimensional space. Then, arbitrarily shaped regions can be classified using this fuzzy–neuron.

B. Testing Procedure

The testing procedure consists of two phases: the *training process* and the *on–line testing process*.

1. *Training process*. This process is carried–out before test.

   (a) Perform a serie of Monte–Carlo simulations of the circuit, under nominal and faulty conditions. The set of measurements of each simulation is stored in a file. Once finished this process, the *training set* has been obtained.

   (b) Assign one fuzzy–neuron to each fault class. The fuzzy–neurons are then trained following the learning process presented in section A.

2. *Testing process*. This process is an on–line process carried–out in wafer–level. In this paper, this process has been replaced by a new serie of Monte–Carlo simulations. Each simulation is assigned to the fault class corresponding to the neuron with the highest output. Tally how often a given class is correctly classified. Also keep track of what classes a given fault is incorrectly classified under.

C. Fault Modelling

Although there are a large catalog of possible faults in analog circuits, only a reduced number of them are usually considered, containing short and open circuits in the terminal of the active devices. This limited catalog includes, however, the most common faults, as can be deduced from some empyrical studies [40].
In the examples of the next section, only the four most common faults in CMOS analog circuits have been included in the fault–list:

1. Gate–drain short.
2. Gate–source short.
3. Open contact in the drain.
4. Open contact in the source.

To warranty the convergence of the simulations, short–circuits are simulated by means of a 1 Ω resistor, while open contacts are simulated by means of a 100 $M\Omega$ resistor. All the faults are assumed to be equally probable.

D. Experimental Results

Figure 10 shows the schematic of two CMOS circuits which have been used to test the method proposed here. Table 8 resumes the results obtained with these two circuits. The *apparent error rate of misclassification* (APER), as defined in [41], has been used as a figure of merit. Let N_1, N_2, ... , N_g be the number of tests for each fault type, and the subscript m meaning misclassified, we define the APER:

$$APER = \frac{N_{1m} + N_{2m} + \ldots + N_{gm}}{N_1 + N_2 + + \ldots + N_g} \tag{12}$$

100 Monte–Carlo runs were generated for each fault type. It was assumed an uniform distribution of the circuit parameters, with the following maxima deviations:

- 10 % for MOS transistors physical dimensions W and L. Matched pairs remain matched within a 1 % of error.
- 10 % for MOS transistor SPICE parameters VTO and TOX.
- 10 % for bias currents.
- 20 % for all the passive components (resistors and capacitors).

The set of measurements is composed of:

- dc measurements in the nodes of the circuit.
- the magnitude of the ac analysis at the frequency of 100 Hz in the output node.

50 Monte–Carlo runs were used to build the training set, while the other 50 were used to build the checking set. Measured learning times in a SUN–10 workstation ranged from about 10 minutes for the single differential pair to approximately 20 minutes for the OTA. These times are much smaller than that for the Monte–Carlo analysis. It can be observed in table 8 that nearly 99 % of correct classification can be obtained using only dc measurements. Furthermore, using only 2 measurements (dc and ac magnitude in the output node), a large fraction of faulty circuits can be detected. Due to the extremely high classification capability obtained with dc measurements, only a minor improvement in the APER is observed whan one ac measurement in the output node is also included.

VI. Conclusions

In this paper, a collection of fuzzy logic–based tools for analog design automation has been presented. These tools share a common fuzzy–logic structure with learning capability.

The first tool is intended for the topology selection of analog cells. It uses fuzzy logic in the selection process. The decision rules can be directly entered by a human expert or they can be automatically generated from the experience with earlier designs. Secondly, a tool for circuit performance modelling has been presented. It can be used for a qualitative description of the circuit behaviour. Alternatively, it can accurately approach the circuit performances of interest by means of a learning process. Finally, a tool for analog testing has been described. It uses a fuzzy–neuron classifier to detect and classify faults in analog circuits.

Some examples of the application of these tools to analog circuits have been shown which demonstrate their abilities.

These tools represent an effort towards automating the analog design, combining new computational intelligence techniques with other conventional approaches based on simulation and optimization. Using the abilities of fuzzy systems, the proposed tools are able to combine experimental crisp data with the intuitive, fuzzy nature of the human knowledge about how analog circuits behave.

Acknowledgements

This work has been partially supported by the Spanish Comisión Interministerial de Ciencia y Tecnología (CICYT), under project TAP–0608–C03–03.

REFERENCES

[1] L.R.Carley and R.A.Rutenbar. "How to automate analog IC design". *IEEE Spectrum*, pp.26–30, Aug. 1988.

[2] M.Ismail and J.Franca, Eds. "Introduction to analog VLSI design automation". *Kluwer Academic Publishers*, 1990.

[3] C.C.Lee. "Fuzzy logic in control systems: fuzzy logic controller, part I". *IEEE Trans. on Syst., Man, And Cybern.*, vol. 20, no. 2, pp. 404–418, 1990.

[4] C.C.Lee. "Fuzzy logic in control systems: fuzzy logic controller, part II". *IEEE Trans. on Syst., Man, And Cybern.*, vol. 20, no. 2, pp. 419–435, 1990.

[5] B.Kosko. "Fuzzy systems as universal approximators". *in Proc. 1992 Int. Conf. Fuzzy Systems*, pp. 1153–1162, San Diego, Mar. 1992.

[6] L.X.Wang. "Fuzzy systems are universal approximators", *in Proc. 1992 Int. Conf. Fuzzy Systems*, pp. 1163–1170, San Diego, Mar. 1992.

[7] L.X.Wang and J.M.Mendel. "Fuzzy basis functions, universal approximation, and ortoghonal least–squares learning". *IEEE Trans. Neural Networks*, vol. 3, no.5, pp. 807–814, Sept. 1992.

[8] L.X.Wang and J.M.Mendel. "Back–propagation fuzzy systems as nonlinear dynamic system identifiers", *in Proc. 1992 Int. Conf. Fuzzy Systems*, pp. 1409–1418, San Diego, Mar. 1992.

[9] B.Kosko, *Neural Networks and Fuzzy Systems*, Prentice Hall, Englewood Cliffs, NJ 1987.

[10] Chi–Cheng Jou. "Supervised learning in fuzzy systems: algorithms and computational capabilities". *in Proc. 2nd Int. Conf. Fuzzy Systems*, pp. 1–6, Mar. 1993.

[11] L.X.Wang. "Training of fuzzy logic systems using nearest neighborhood clustering". *in Proc. 2nd. IEEE Int. Conf. Fuzzy Systems*, pp. 13–17, San Francisco, Mar. 1993.

[12] D.E.Rumelhart and J.L.McCleland (eds.). *Parallel and Distributed Processing I,II*, Cambridge: MIT Press, 1986.

[13] H.Y.Koh, C.H.Séquin, P.R.Gray. "OPASYN: A Compiler for CMOS Operational Amplifiers". *IEEE Trans. Computer Aided Des.*, vol. CAD–9, no. 2, February 1990, pp. 113–125.

[14] W.Nye, D.C.Riley, A.Sangiovanni–Vincentelli, André L.Tits. "DELIGHT.SPICE: An Optimization–Based System for the Design of Integrated Circuits". *IEEE Trans. Computer Aided Des.*, vol. CAD–7, no. 4, April 1998, pp. 501–519.

[15] J.M.Shy, A.Sangiovanni–Vincentelli. "ECTASY: A new environment for IC design optimization". *Proc. ICCAD 1988*, pp. 484–487.

[16] P.C.Maulik, L.R.Carley, D.J.Allstot. "Sizing of Cel-level Analog Circuits using Constrained Optimization Techniques". *IEEE J. Solid–State Circuits*, vol. 28, no. 3, March 1993, pp. 233-241.

[17] G.G.E.Gielen, H.C.C.Walscharts, W.M.C.Sansen. "Analog Circuit Design Optimization Based on Symbolic Simulation and Simulated Annealing". *IEEE J. Solid–State Circuits*, vol. SC–25, no. 3, June 1990, pp. 707–713.

[18] E.S.Ochotta, R.A.Rutenbar, L.R.Carley. "Equation–free synthesis of high–performance linear analog circuits". *Proc. 1992 Brown/MIT Conf. Advanced Research in VLSI and Parallel Systems*, Cambridge, MA:MIT Press 1992, pp. 129–143.

[19] J.Chávez, M.A. Aguirre, A.Torralba. "Analog Design Automation: A Case Study". *Proc. of the IEEE 1993 Int. Symp. on Circuits and Systems, ISCAS'93*, May 1993, Chicago, USA, pp. 2083–2085.

[20] M.A.Aguirre, J.Chávez, A.Torralba and L.G.Franquelo. "Analog design automation by means of a Tabu Search approach". *Proc. of the IEEE Int. Symp. on CAS, ISCAS'94*, vol. 1, pp.375–378, Londres, May 1994.

[21] F.El–Turkey and E.E.Perry. "BLADES: an artificial intelligence approach to analog circuit design". *IEEE Trans. Computer–Aided Design*, vol. 8, no. 6, pp. 680– 692, June 1989.

[22] R.Harjani, R.A.Rutenbar and L.R.Carley. "OASYS: a framework for analog circuit synthesis". *IEEE Trans. on Computer–Aided Design*, vol. 8, no. 12, pp. 1247–1266, Dec. 1989.

[23] J.W.Bandler and H.L.Abdel–Malek. “Optimal centering, tolerancing and yield determination via updated approximations and cuts”. *IEEE Trans. Circuits Syst.*, vol. 25, pp. 853–871, 1978.

[24] D.E.Hecevar, M.R.Lightner, and T.N.Trick. “An extrapolated yield approximation technique for use in yield maximization”. *IEEE Trans. Comp.–Aided Design*, vol. 3, pp. 279–287, Oct. 1984.

[25] R.M.Biernacki and M.A.Styblinski. “Statistical circuit design with a dynamic constraint approximation scheme”, *in Proc. IEEE Int. Symp. Circuits Syst.*, pp. 976–979, San José, 1986.

[26] S.Ikeda, M.Ochiai, and Y.Sawargi, “Sequential GMDH and its application to river flow prediction”, *IEEE Trans. Syst. Man Cybern.*, vol. 6, no. 7, July 1976.

[27] H.U.Huang, “Unified approach to quadratically convergent algorithm for function minimization”, *JOTA*, no. 5, pp. 405–423.

[28] T.K.Yu, S.M.Kang, I.N.Hajj and T.N.Trick, “Statistical performance modeling and parametric yield estimation of MOS VLSI”, *IEEE Trans. Comp.–Aided Design*, vol. 6, pp. 1013, 1022, Nov. 1987.

[29] M.A.Styblinski and S.A.Aftab. “Combination of interpolation and self–organizing approximation techniques – a new approach to circuit performance modeling”, *IEEE Trans. Comp.–Aided Design*, vol. 12, pp. 1775–1785, Nov. 1993.

[30] M.D.Mckay et al. “A comparison of three methods for selecting initial values of input variables in the analysis of output from a computer code”. *Technometrics*, vol. 21, pp. 239–245, 1979.

[31] P.Duhamel and J.C.Rault. “Automatic test generation techniques for analog circuits and systems: a review”. *IEEE Trans. Circuits Syst.*, vol. 26, pp. 411–440, Jul. 1979.

[32] J.W.Bandler and A.E.Salama. “Fault diagnosis of analog circuits”. *Proc. IEEE*, vol. 73, pp. 1270–1325, Aug. 1985.

[33] M.Ismail and M.Soma, eds. (Special issue on “analog testing and mixed–signal testing”). *Journal of Electronic Testing*, vol. 4, no.4, 1993.

[34] R.Liu, ed. *Testing and diagnosis of analog circuits and systems*. Van Nostrand Reinhold, 1991.

[35] S.D.Bedrosian and J.H.Lee. “Automatic fault test generation for nonlinear analog circuits using the fuzzy concept”, in *Proc. 1978 21st Midwest Symp. Circuits Syst.*, pp. 399–403.

[36] J.Lee and S.D.Bedrosian. “Fault isolation algorithm for analog electronic systems using the fuzzy concept”. *IEEE Trans. Circuits Syst.*, vol. 26, pp. 518–522, Jul. 1979.

[37] W.Maly and Z.Pizlo. “Tolerance assignment for IC selection tests”. *IEEE Trans. Computer–Aided Design*, vol. 4, pp. 156–162, 1985.

[38] L.Milor and V.Visvanathan. “Detection of catastrophic faults in analog integrated circuits”. *IEEE Trans. on Computer–Aided Design*, vol. 8, pp. 114–130, Feb. 1989.

[39] B.R.Epstein, M.Czigler, and S.R.Miller. “Fault detection and classsification in linear integrated circuits: an application of discrimination analysis and hypothesis testing”. *IEEE Trans. on Computer–Aided Design*, vol. 12, pp. 102–113, Jan. 1993.

[40] P.Banerjee and J.A.Abraham. “Fault characterization of VLSI MOS circuits”, in *Proc. IEEE Int. Conf. Circuits and Computers*, New York, pp. 564–568, Sept.–Oct. 1982.

[41] R.A.Johnson and D.W.Wirchern. *Appliedd Multivariate Statistical Analysis*. Englewood Cliffs, NJ: Prentice Hall, 1988.

Specification name	Spec.
DC Gain (dB)	90
Unity Gain Freq. (MHz)	1
Phase Margin (deg)	60
Slew Rate ($V/\mu sec$)	1
$1/f$ Noise at 1 KHz ($nV/\sqrt{Hz}$)	150
Power Dissip. (mW)	1
Active Area (μm^2)	min.
Load Capacitance (pF)	10

Table 1: An example of user specifications for opamps.

Gain / Freq.	S	M	L
S	VL	VL	M
M	VL	L	S
L	L	M	VS

a)

Gain / Freq.	S	M	L
S	VL	L	S
M	L	M	VS
L	M	S	VS

b)

Gain / Freq.	S	M	L
S	VL	L	S
M	VL	L	S
L	VL	M	S

c)

VL	*Very Large*
L	*Large*
M	*Medium*
S	*Small*
VS	*Very Small*

Table 2: Decision rules defined by an expert. The tables show the adequacy of each topology in the case of a large active area. a) BTS, b) OTA, c) Folded Cascode OTA.

Variable Name	$\bar{x}_i$	$\pm\Delta x_i$
x_1: Capacitor C_{aa} (pF)	0.20	0.10 (50.0 %)
x_2: Capacitor C_{bb} (pF)	0.20	0.10 (50.0 %)
x_2: Capacitor C_{cc} (pF)	0.55	0.45 (81.8 %)
x_4: Width of transistor M1, W_1 (μm)	14.00	10.50 (75.0 %)
x_5: Width of transistor M2, W_2 (μm)	14.00	10.50 (75.0 %)
x_6: Width of transistor M3, W_3 (μm)	14.00	10.50 (75.0 %)
x_7: Rise time of input waveform, T (ns)	4.10	3.90 (95.1 %)

Table 3: Input Variable Ranges for the Three–Input NAND Gate.

	Training Set			Checking Set		
N.Rules	Std.Dev	Avg.Error	Max.Error	Std.Dev.	Avg.Error	Max.Error
4	1.947	1.500	6.670	3.520	2.507	9.852
8	1.044	0.797	4.017	2.841	1.900	10.020
16	0.593	0.476	1.695	2.824	1.888	8.810

learning in 1000 iterations

Table 4: Model Accuracy for the Three–Input NAND Gate. Results in %.

Variable Name	$\bar{x}_i$	$\pm\Delta x_i$
x_1: Width of M9 and M14 (μm)	250.00	150.00 (60.00 %)
x_2: Width of M10 and M13 (μm)	300.00	66.67 (22.22 %)
x_3: Width of M11 and M12 (μm)	4.00	0.75 (18.75 %)
x_4: Bias Current I_{ref} (μA)	50.00	0.50 (1.00 %)

Table 5: Input Variable Ranges for the TCA.

	Training Set			Checking Set		
N.Rules	Std.Dev	Avg.Error	Max.Error	Std.Dev	Avg.Error	Max.Error
4	0.744	0.610	1.857	0.956	0.773	2.506
8	0.339	0.472	1.713	0.865	0.667	2.473
16	0.318	0.246	0.944	0.584	0.376	2.552

learning in 1000 iterations

Table 6: Model accuracy for the CMOS linear TCA. Average dc gain (A_g). Results in %.

	Training Set			Checking Set		
N.Rules	Std.Dev	Avg.Error	Max.Error	Std.Dev	Avg.Error	Max.Error
4	1.385	1.076	5.175	3.271	2.628	8.309
8	0.975	0.662	4.837	2.414	1.712	8.420
16	0.883	0.617	4.234	2.088	1.585	6.038

learning in 1000 iterations

Table 7: Model accuracy for the CMOS linear TCA. Nonlinearity error ($NLerror$). Results in %.

	Go/No Go	Fault Classes	Fault Classification (Average)
Singel Differential Pair		15	
Only output node. ac and dc (2 measur.)	7.67		13.53
dc in all nodes (4 measurements)	1.73		0.47
dc in all nodes and ac in output node (5 measurements)	0.4		0.4
Output Trans. Amplifier		23	
Only output node. ac and dc (2 measur.)	8.04		24.17
dc in all nodes (5 measurements)	1.13		5.48
dc in all nodes and ac in output node (6 measurements)	1.04		0.7

Table 8: Analog testing tool: summary of results. Data show the APER (Apparent Rate of Error) in percentage.

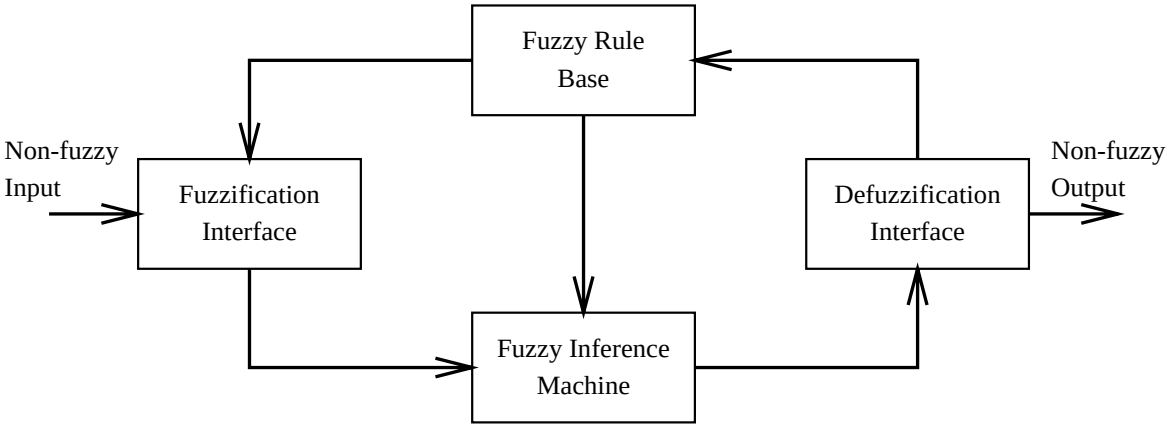
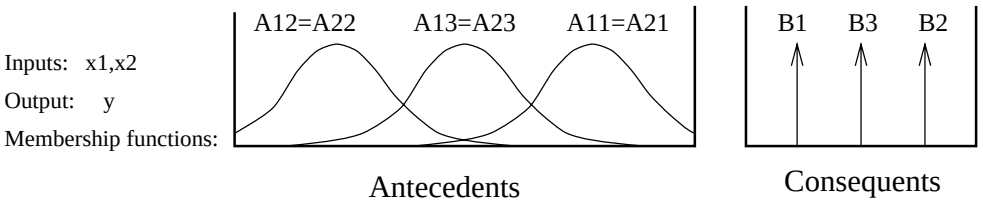


Figure 1: The structure of a fuzzy model.



Rules:

R1	If	x1	is	A11	and	x2	is	A21	then	y is B1
R2	If	x1	is	A12	and	x2	is	A22	then	y is B2
R3	If	x1	is	A13	and	x2	is	A23	then	y is B3

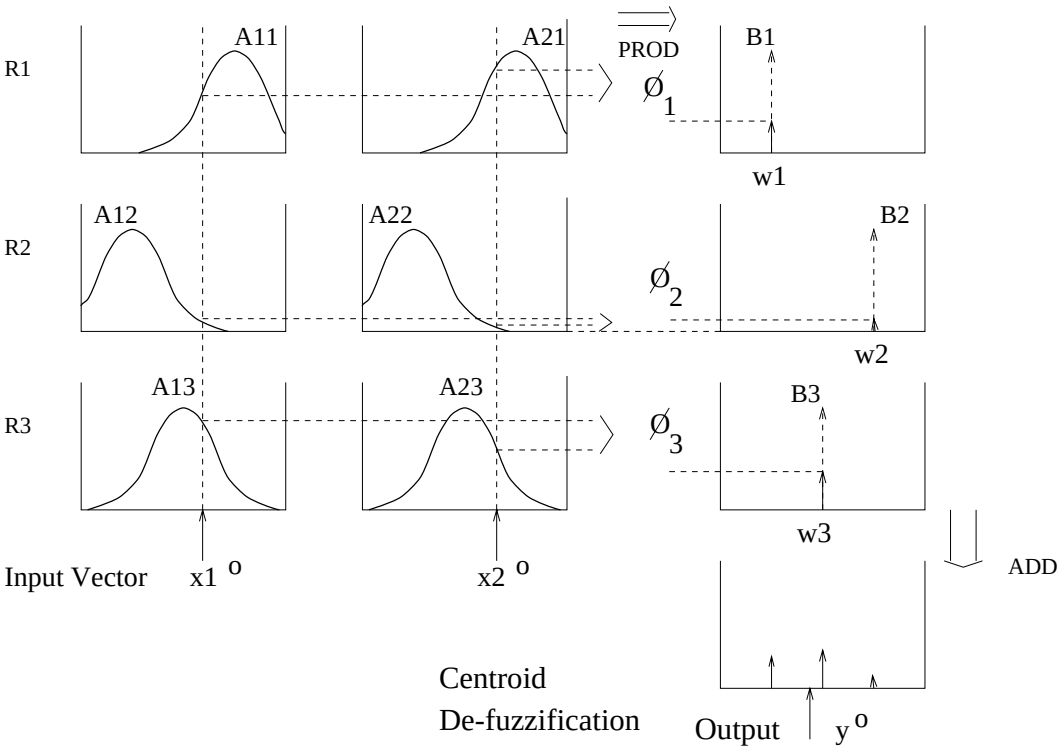


Figure 2: The inference process.

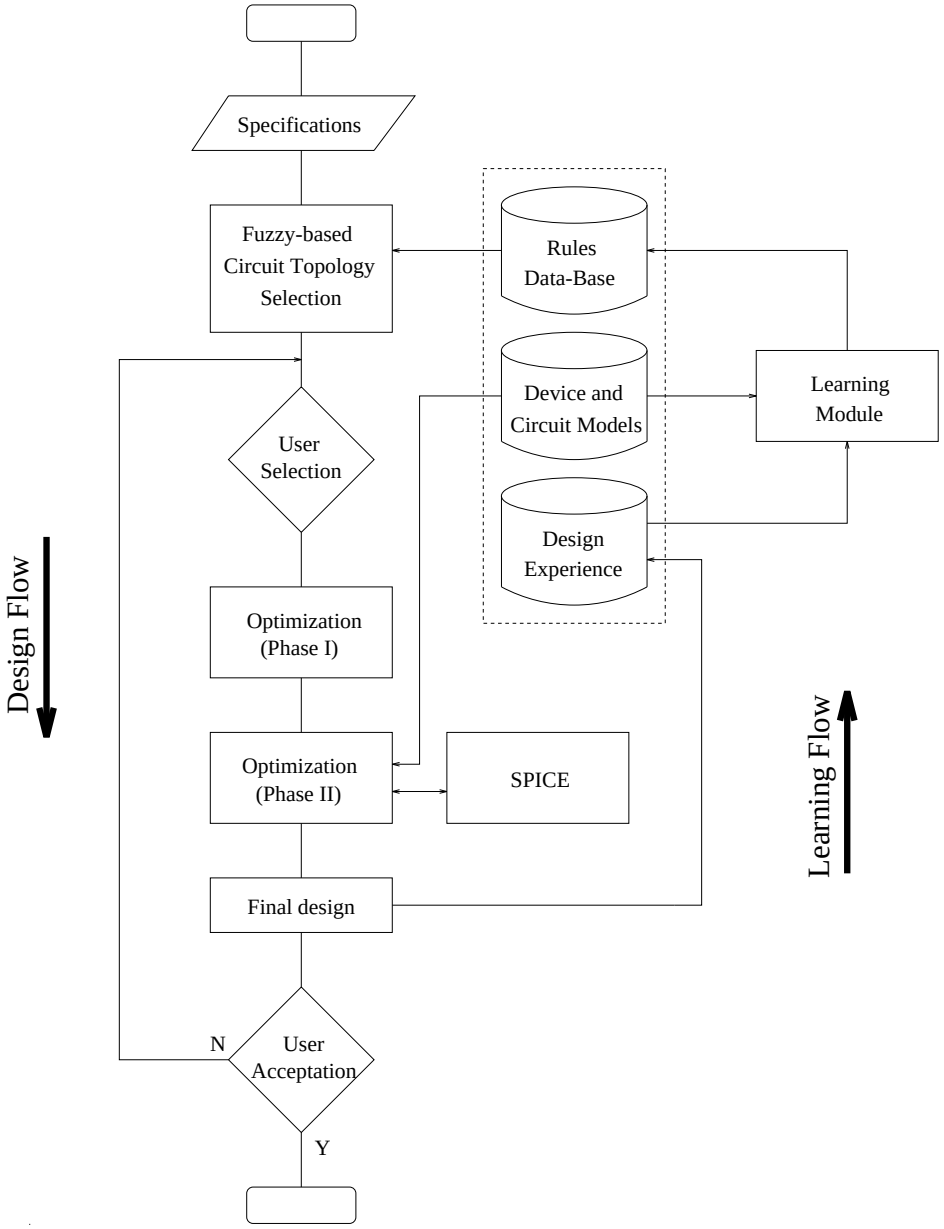


Figure 3: Block Diagram of the design system.

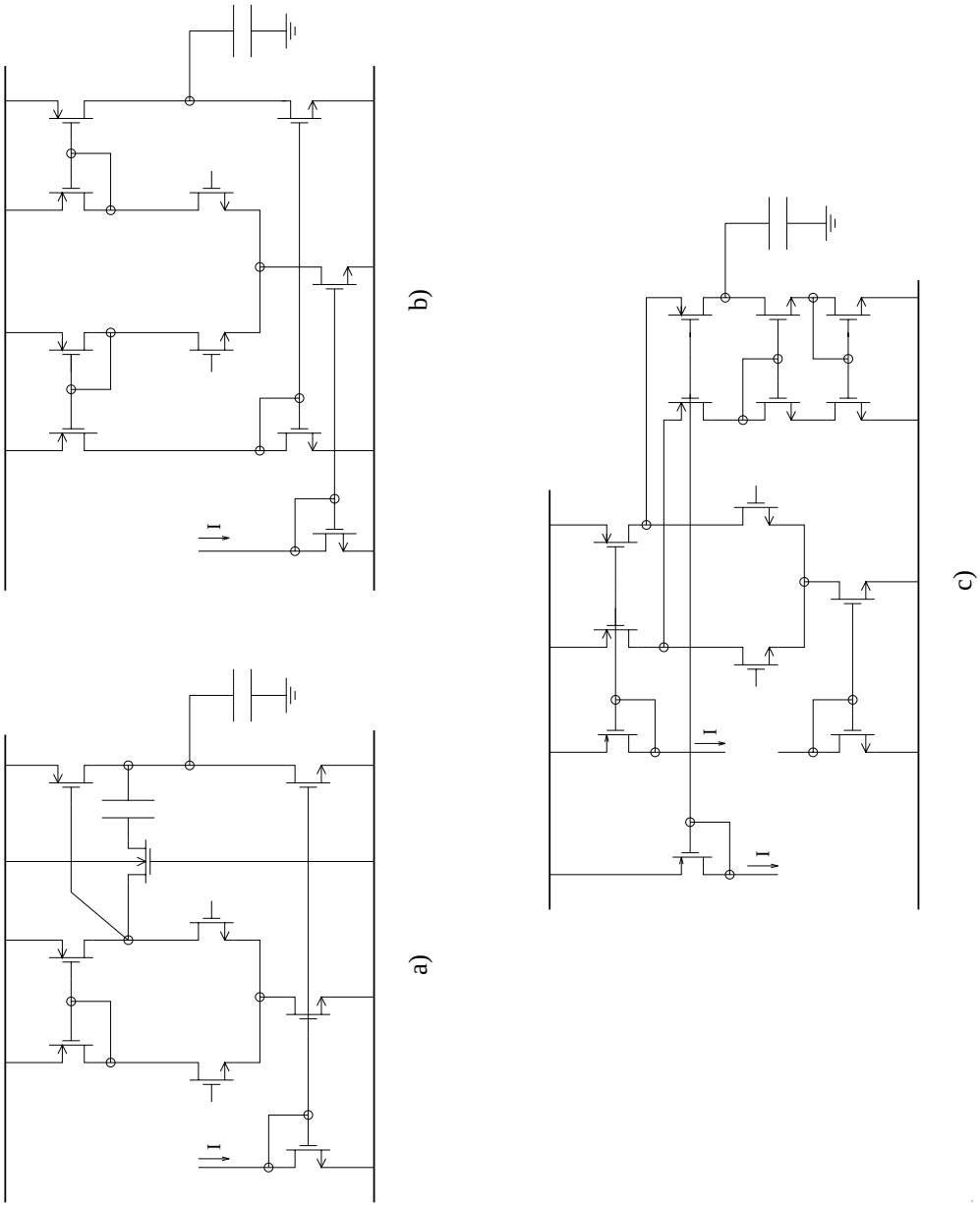
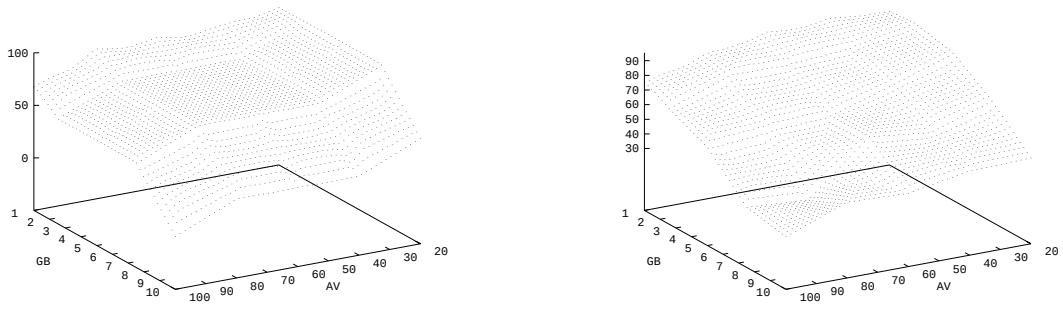
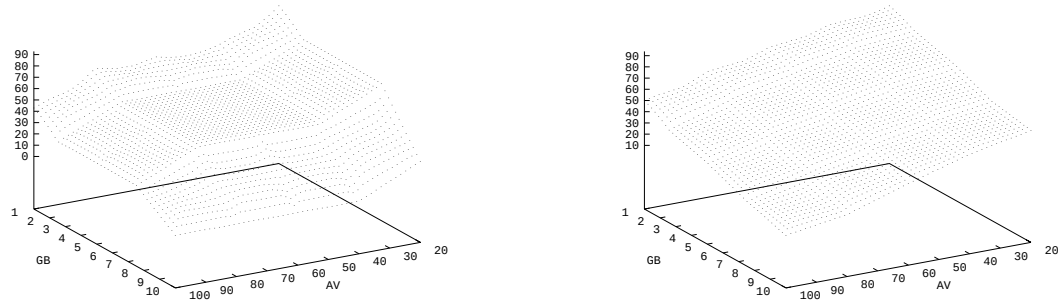


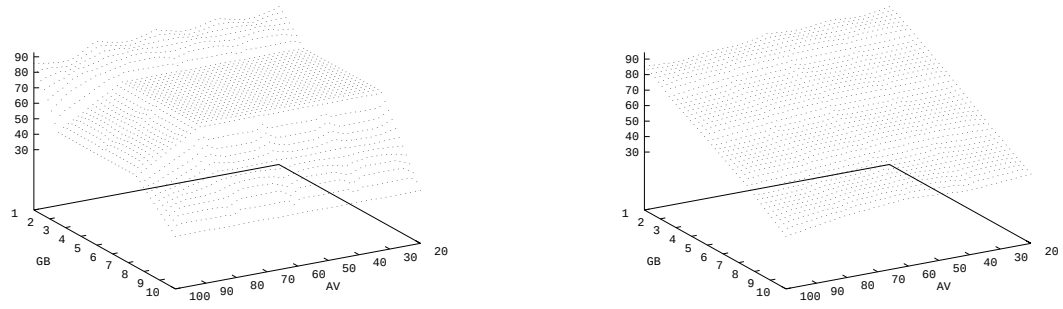
Figure 4: Basic op.amp. topologies: a) Basic Two Stage (BTS) op.amp., b) Output Transconductance Amplifier (OTA) and c) Folded Cascode OTA.



a) BTS b)



a) OTA b)



a) Folded Cascode OTA b)

Figure 5: Decision surfaces: a) Obtained from a human expert. b) Obtained from the optimization process.

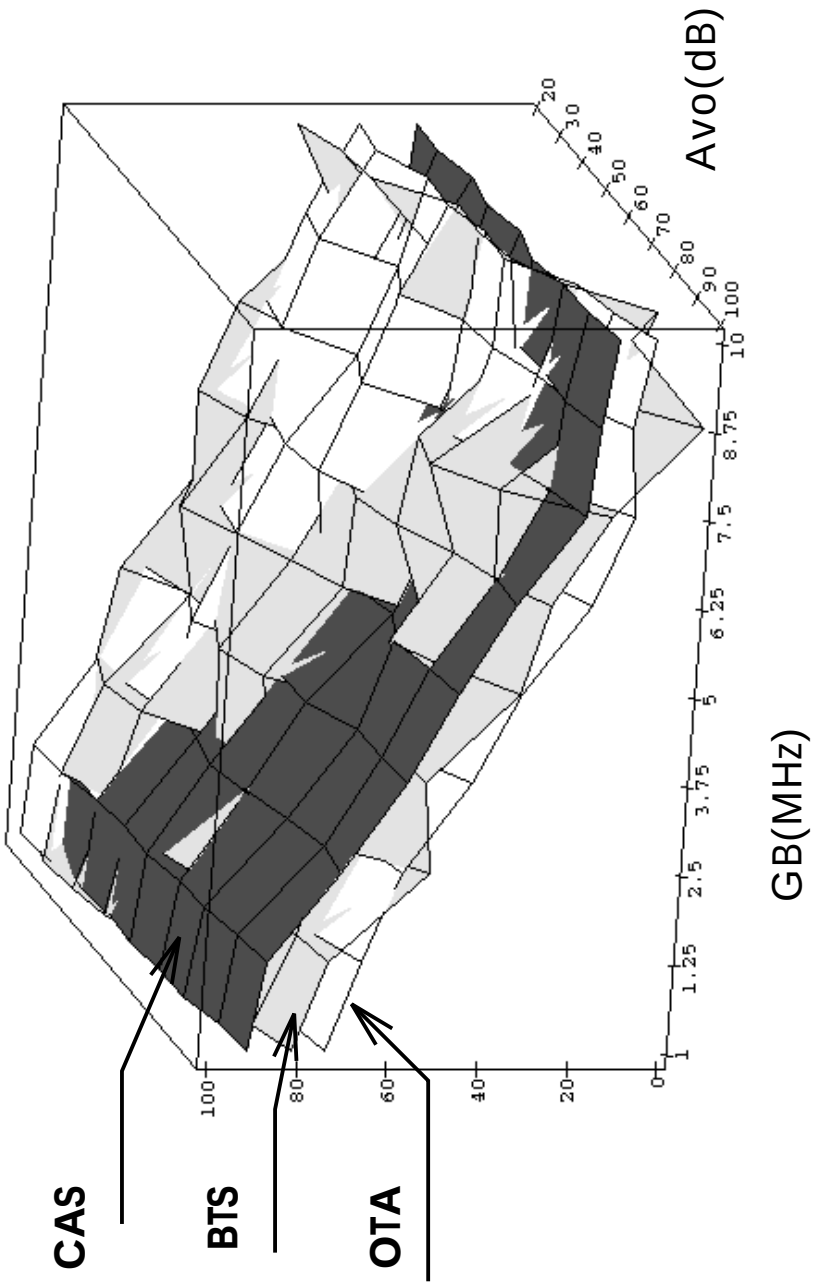


Figure 6: Superimposed decision surfaces.

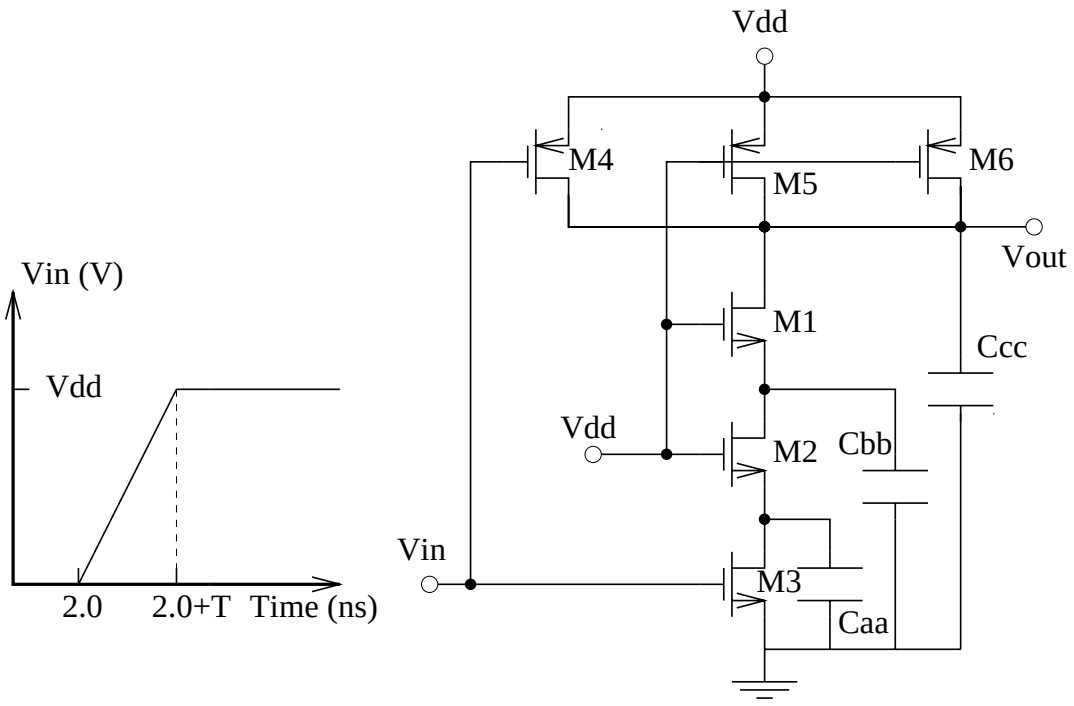


Figure 7: Three–Input NAND gate.

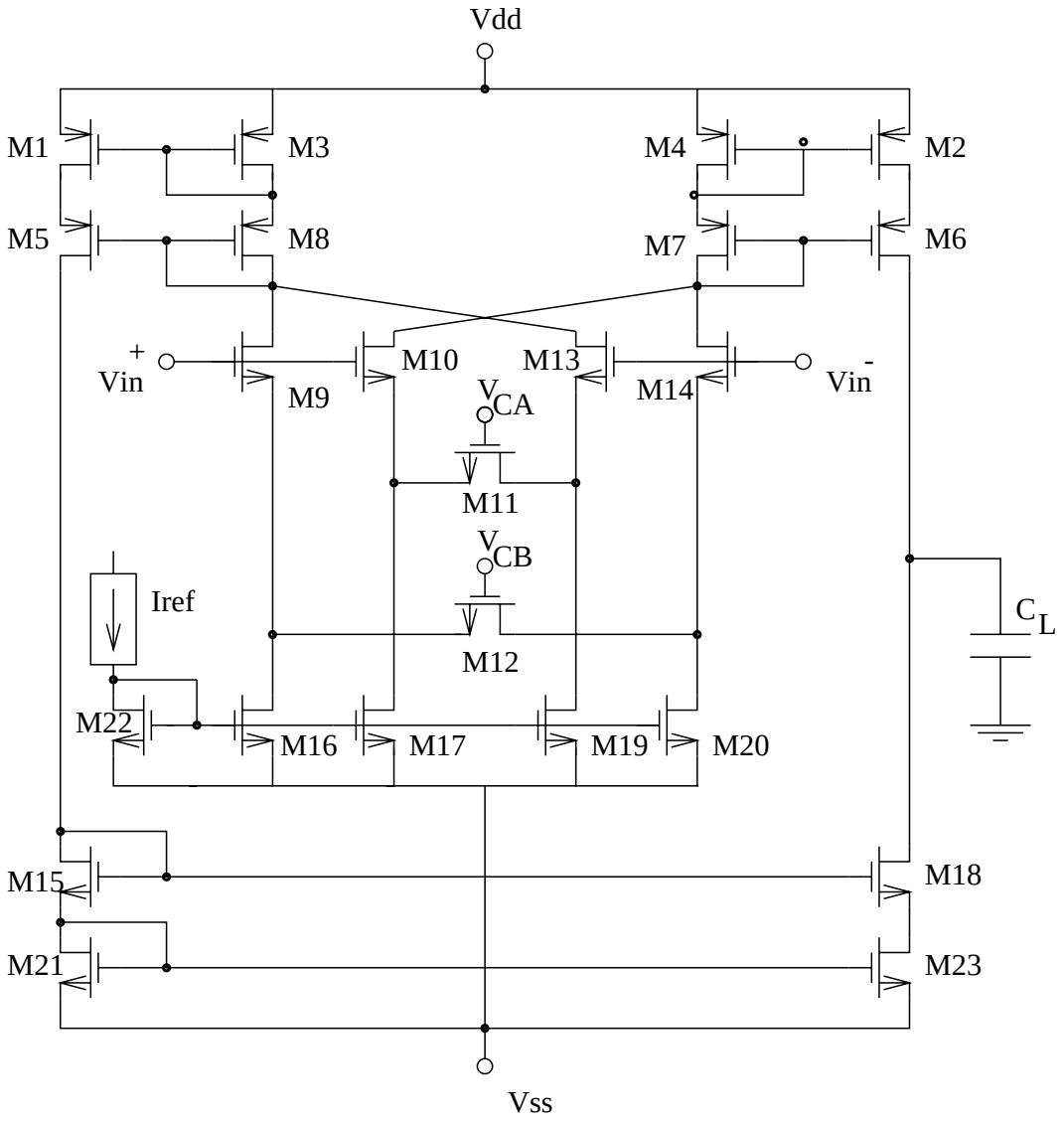
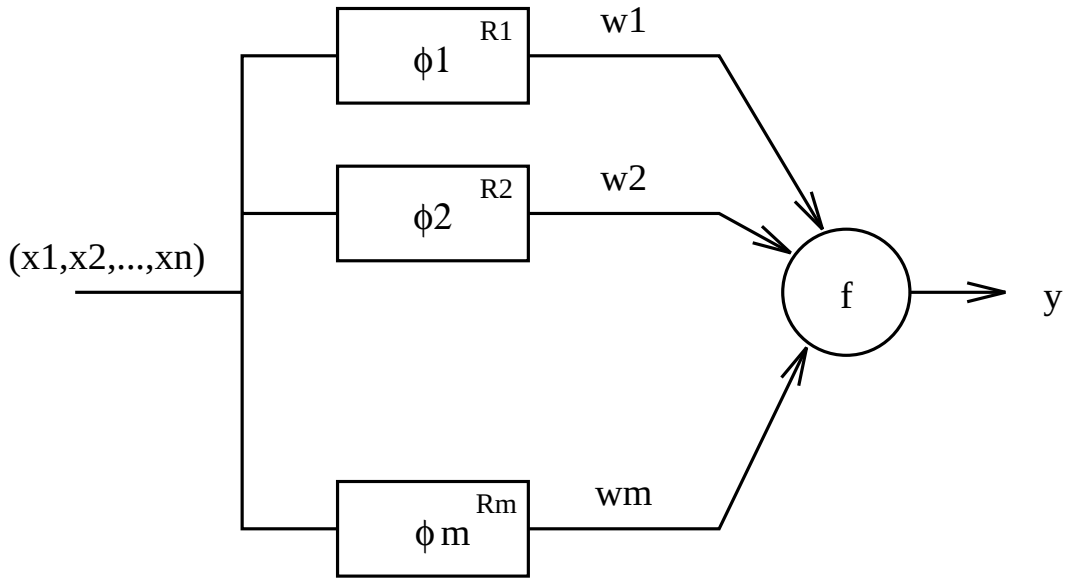


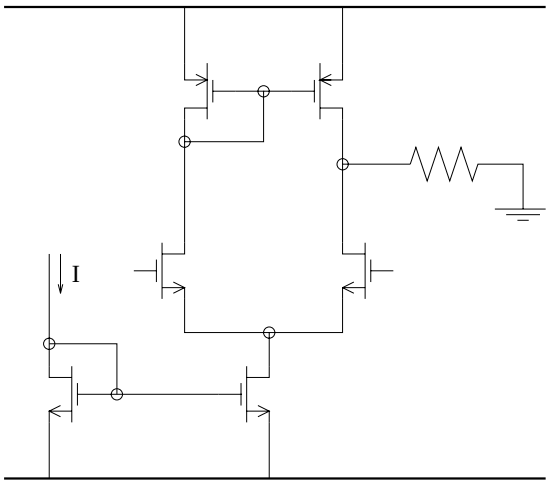
Figure 8: CMOS linear TCA.



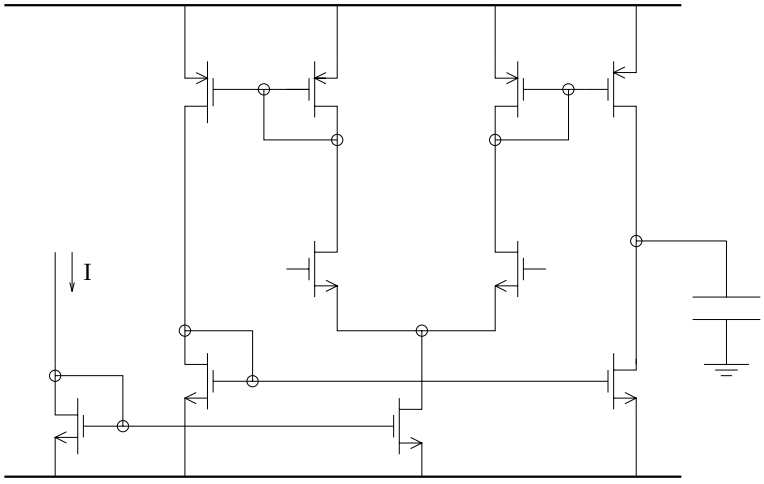
$$\phi_i = \prod_{j=1}^{n} \mu_{ij}(c_{ij}, \sigma_{ij}) \qquad y = f\left(\sum_{i=1}^{m} w_i \phi_i\right)$$

$$\mu_{ij} = e^{-1/2\left(\frac{x_j - c_{ij}}{\sigma_{ij}}\right)^2} \qquad f(x) = \frac{1}{1 + e^{-x}}$$

Figure 9: Fuzzy neuron–based classifier.



a) Differential Pair Circuit (DPC)



b) Output Transconductance Amplifier (OTA)

Figure 10: Circuit examples.