ANALOG CMOS IMPLEMENTATION OF NEURO-FUZZY SYSTEMS

by

Baykal SARIOĞLU

B.S., Electronical Engineering, Kadir Has University, 2004

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science

Graduate Program in Electronical Engineering
Boğaziçi University

2007

ANALOG CMOS IMPLEMENTATION OF NEURO-FUZZY SYSTEMS

APPROVED BY:

Prof. Günhan Dündar (Thesis Supervisor)	…….
Prof. H. Levent Akın	……
Assoc. Prof. Oğuzhan Çiçekoğlu	……

DATE OF APPROVAL:

ACKNOWLEDGEMENTS

First of all, I take this opportunity to express the profound gratitude from my deep heart to my beloved parents for their patience, love and continuous support.

I wish to express my very sincere gratitude and appreciation to Prof. Günhan Dündar for his attention, guidance, insight, and support during this research and the preparation of this thesis.

I would like to thank to Prof. Bogdan Wilamovski. His vision and support made this project possible.

In addition, special thanks are due to Selçuk Talay and Berke Yelten for constructive comments and suggestions during the development of this work.

Finally, I would like to thank for the support of TÜBİTAK.

ABSTRACT

ANALOG CMOS IMPLEMENTATION OF NEURO-FUZZY SYSTEMS

In this thesis, a novel neuro-fuzzy system presented and implemented in analog CMOS. The system is based on the fact that a rule in a zero order TSK fuzzy system can be represented as an area on an input space which is created by a neural network. If the rule output values of the fuzzy system are assigned to the corresponding areas and additionally, if the neuron threshold values and the weights of the neural network are selected suitably, the fuzzy system can be mapped on the neural network.

Implemented system consists of five main blocks; threshold block, area selection block, normalization block, weight assigning block and summing block. The blocks and their designs are introduced and additionally digital control units, which are used for determination of desired areas and values, are presented.

Implemented chip works in current mode, while the inputs are taken as voltages and the output is taken as a current. SPICE and theoretical MATLAB simulations and example fuzzy rule mappings show that implemented chip architecture works accurately and it is able to evaluate eight million fuzzy rules per second. The maximum power dissipation of the chip is equal to 37 mW.

ÖZET

BULANIK-SİNİR AĞLARININ ANALOG CMOS GERÇEKLEŞTİRİMİ

Bu çalışmada, geliştirilmiş olan yeni bir bulanık-sinir ağ mimarisi açıklanmış ve analog CMOS devre gerçekleştirimi yapılmıştır. Gerçekleştirilen sistem, bir sıfırıncı dereceden TSK bulanık mantık yapısındaki kuralların, bir yapay sinir ağı mimarisindeki sinir hücrelerinin belirlediği alanlar ile ifade edilebilmesi esasına dayanmaktadır. Eğer bulanık mantık yapısındaki kuralların çıkış değerleri ilgili alanlara atanır ve yapay sinir ağı mimarisindeki sinir hücrelerinin eşik değerleri ile ağırlık değerleri gerektiği gibi seçilebilirse, bulanık mantık yapısı, bir yapay sinir ağı mimarisi kullanılarak gerçekleştirilebilmektedir.

Gerçekleştirilmiş olan analog CMOS devre mimarisinin beş ana bölümü; eşik devresi, alan seçimi devresi, normalleştirme devresi, ağırlık atama devresi ve son olarak ta toplama devresi açıklanmıştır. Bu devrelerin yanında istenilen değerlerin istenilen alanlara ve ağırlıklara atanılmasında kullanılan sayısal denetim devreleri de tasarlanmış ve açıklanmıştır.

Gerçekleştirilen devre gerilim girişleri kabul ederken, çıkışı akım olarak vermekte ve iç yapı olarak akım bazlı çalışmaktadır. Benzetimler gerçekleştirilen yapının kuramsal yapıya uygun olarak çalıştığını ve saniyede sekiz milyon bulanık kuralı değerlendirebildiğini göstermektedir. Toplam güç tüketimi ise 37 mW olarak ölçülmüştür.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS/ABBREVATIONS

A_R	Process dependent matching parameter
I_{Th}	Threshold current
K	Process transconductance parameter
k_p'	PMOS process transconductance parameter
k_n'	NMOS process transconductance parameter
L	Length
R	Resistance
$R_{\square}$	Sheet resistance
V_G	Gate voltage
V_{SG}	Source-to-gate voltage
V_T	Thermal voltage
V_{Th}	Threshold voltage
W	Width
μ	Membership function
$\phi_{fuzzified}$	Fuzzified ϕ input
DAC	Digital-to-Analog Converter
LSB	Least Significant Bit
MSB	Most Significant Bit
NT	Threshold Neuron
NT_X	X dimension NT block
NT_Y	Y dimension NT block
TLU	Threshold Logic Unit
TSK	Takagi Sugeno Kang

1. INTRODUCTION

Neural networks and fuzzy systems are both very popular techniques in soft computing. The term soft computing was invented by Lotfi A. Zadeh, the founder of fuzzy logic [1]. Soft computing covers approaches to human reasoning that try to make use of the human tolerance for incompleteness, uncertainty, imprecision and fuzziness in decision making processes. Evolutionary computation and probabilistic reasoning are also included in soft computing in addition to neural networks and fuzzy systems. Soft computing is in particular concerned with combinations of these methodologies. Currently, neuro-fuzzy systems are the most visible approach to such combinations [27].

In this thesis a novel neuro-fuzzy system developed by Wilamovski [34] is examined and implemented in analog CMOS.

1.1. Neuro-Fuzzy Architecture

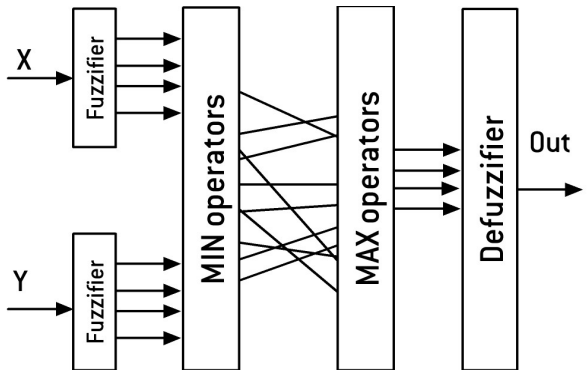


Figure 1.1. Block diagram of classical Mamdani type fuzzy controller

The block diagram of a typical Mamdani type fuzzy control system is shown in Figure 1.1 [2]. The system is composed of three main parts; fuzzifiers, a main processing unit with MIN and MAX operators and a defuzzifier.

Fuzzifiers convert crisp values into grades of membership for linguistic terms. This process is designated as fuzzification. The membership function is used to associate a

grade to each linguistic term. Various shapes of membership functions are shown in Figure 1.2.

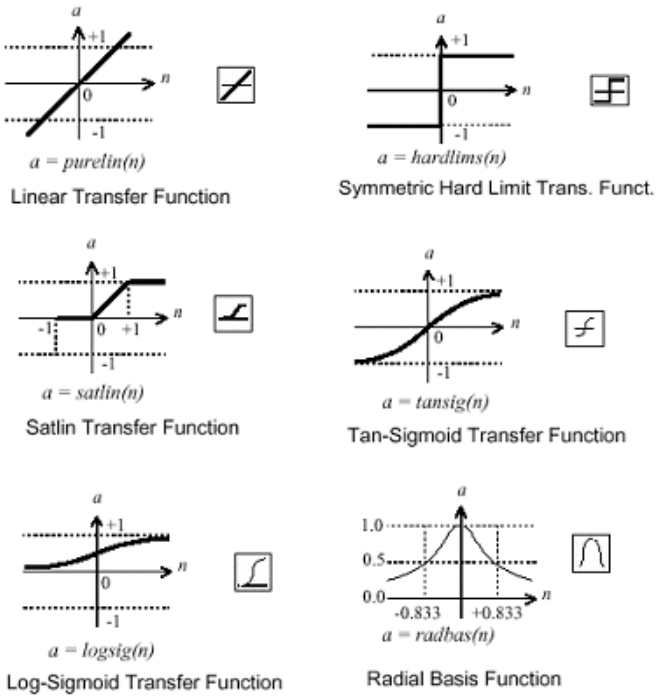


Figure 1.2. Various membership functions

In the center of the Mamdani controller, fuzzified inputs are processed by fuzzy logic blocks with MIN and MAX fuzzy logic operators. Fuzzy logic is a super set of conventional (or Boolean) logic and contains similarities and differences with Boolean logic. For instance, MIN operators are used instead of AND operators and in place of OR operators, MAX operators are implemented. As a matter of fact MIN and MAX operators represent intersection and union operations of membership functions. Intersection and union operations are generalized as t-norm and t-conorm (or s-norm) operators, respectively [3]. Several t-norm and t-conorm operators can also be used instead of MIN or MAX operators [4][5][6].

The last block of the controller is the defuzzification block, where the output analog variable is retrieved from a set of fuzzy variables. The centroid method, in which the "center of mass" of the result is provided as the crisp output value, is the most common type of defuzzification technique implemented in fuzzy systems today [28].

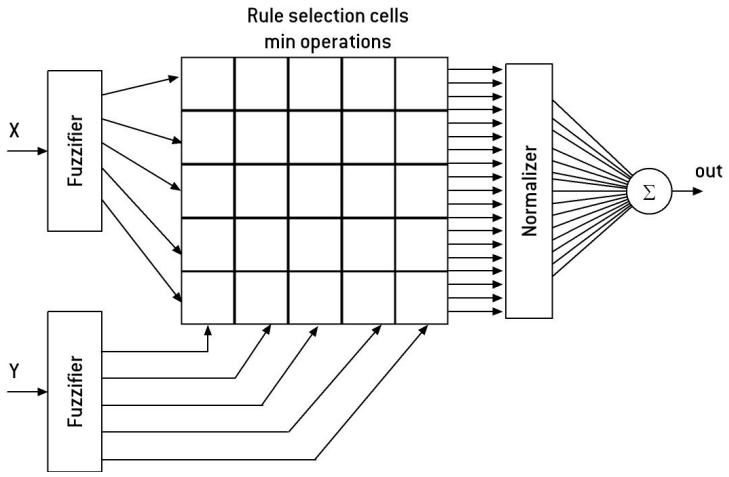


Figure 1.3. TSK (Takagi-Sugeno-Kang) fuzzy architecture

As an alternative to the Mamdani architecture, TSK (Takagi-Sugeno-Kang) architecture was introduced recently. In Mamdani model, fuzzy sets are used for conclusions, while conclusions are represented as functions in the TSK model. Therefore, TSK type fuzzy controller embodies a different type of defuzzification block, which is mostly composed of normalization and weighted sum blocks [7]. Rule configuration in (1.1) is Mamdani type where (1.2) shows TSK type rule representation.

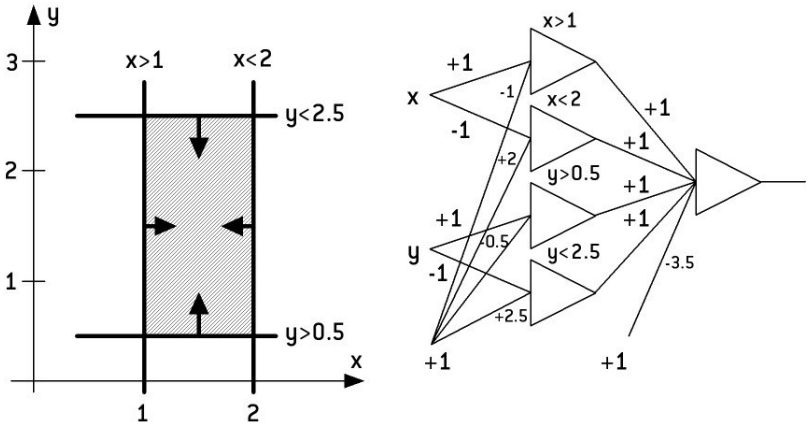
$$R_i\text{: If } x_1 \text{ is } A_{r1} \text{ AND } x_2 \text{ is } A_{r2} \ldots \text{ OR } xn \text{ is } A_{rn} \text{ then } u=B_i \tag{1.1}$$

$$R_i\text{: If } x_1 \text{ is } A_{r1} \text{ AND } x_2 \text{ is } A_{r2} \ldots \text{ AND } xn \text{ is } A_{rn} \text{ then } u=f_r(x_1, x_2, \ldots, x_n) \tag{1.2}$$

The TSK structure, as shown in Figure 1.3, also does not require MAX operators. Weighted average operation is applied directly to regions selected by MIN operators. What makes the TSK system really simple is that the output weights are proportional to the average function values at the selected regions by MIN operators.

A rule can be illustrated as a region on an input space. Since a single neuron in a neural network architecture can divide input space by a line, a plane, or a hyper plane, the rule selection implementation is possible in a neural network architecture. In order to select just one region in n-dimensional input space, more than n+1 neurons should be used. For example, to separate a rectangular pattern in two dimensions, four neurons are required, as is shown in Figure 1.4. The number of regions is proportional to number of first layer

neurons. Additionally number of selectable regions is proportional to number of hidden layer neurons. Therefore if more selectable input clusters are needed, then the number of neurons in the hidden layer should be properly multiplied. If there is no limitation for the number of neurons in the hidden layer, then all classification problems can be solved using a three layer neural network architecture.



(a) (b)

Figure 1.4. (a) Separation of the rectangular area on a two dimensional space and (b) desired neural network to fulfill this task

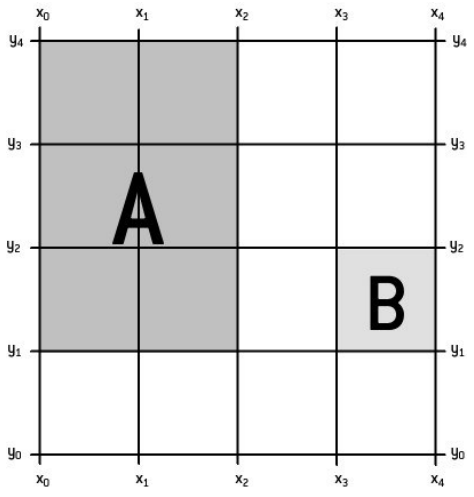


Figure 1.5. Two dimensional input plane separated vertically and horizontally by five neurons in each direction

As a result, with concept shown on Figure 1.4, fuzzifiers and MIN operators used for region selection of a TSK fuzzy system can be replaced by a simple neural network

architecture. Let us analyze Figure 1.5 where a two-dimensional input space was divided by five neurons horizontally and by five neurons vertically. The corresponding neural network architecture is shown in Figure 1.6. Each neuron is connected only one input. For each neuron input, weight is equal to +1 and threshold is equal to the value of the crossing point on the x or y axis. Neurons in the second layer have two connections to lower boundary neurons with weights of +1 and two connections to upper boundary neurons with weights of -1. Thresholds for all neurons in the second layer are set to +1. Only two of them are drawn on Figure 1.6. One of them selecting region A, which is bordered by x_0, x_2, y_1 and y_4 neurons and the second is selecting region B, which is bordered by x_3, x_4, y_1 and y_2 neurons.

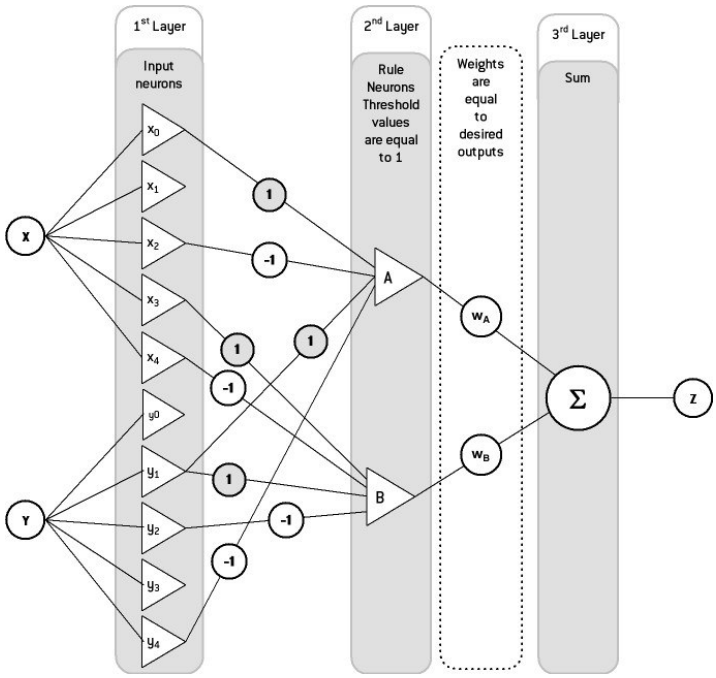


Figure 1.6. Neural network performing the function of TSK fuzzy system

Weights in the last layer have values in proportion to the expected function in selected areas. All neurons in Figure 1.6 have unipolar activation functions and if the system is properly designed, then for any input vector in certain areas only the neuron of this area produces +1, while all remaining neurons produce no output values. In the case when the input vector is close to a boundary between two or more regions, then all participating neurons are producing fractional values and the system output is generated as a weighted sum. For proper operation it is important that the sum of all outputs of the

second layer must be equal to +1. In order to assure above condition, an additional normalization block can be introduced, in a similar way as it is done in TSK fuzzy systems where mostly weighted average operation (1.3), where *w* is weight and *x* is fuzzified input value, is applied in defuzzification block [7]. The normalization block can be placed on some different positions on the system architecture. One example placement is shown in Figure 1.7. Some additional placement strategies are reviewed in section 2.5.

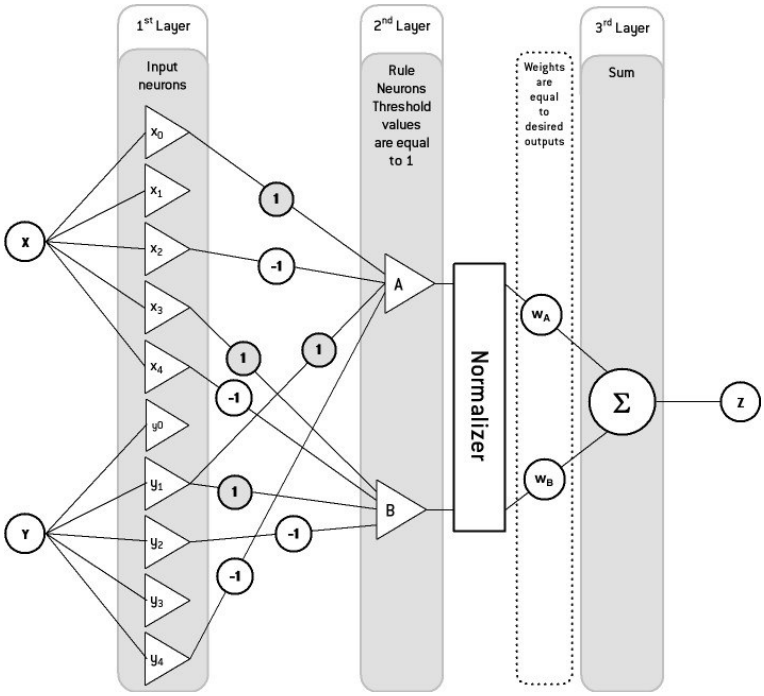


Figure 1.7. Neural network architecture with additional normalizer

$$Out = \frac{\sum w \cdot x}{\sum w} \tag{1.3}$$

The Neuro-Fuzzy architecture in Figure 1.7 is the final architecture which is implemented in this thesis.

1.1.1. Membership Functions

In fuzzy systems region selection is implemented by using fuzzification of inputs using membership functions [27]. Since the system is basically a fuzzy system on a neural network architecture, the fuzzification operation is basically the same. On the other hand, it

may be relatively hard to see how the membership functions are implemented in the architecture shown in Figure 1.7. The meaning of the membership functions and their implementation is in this architecture is discussed below.

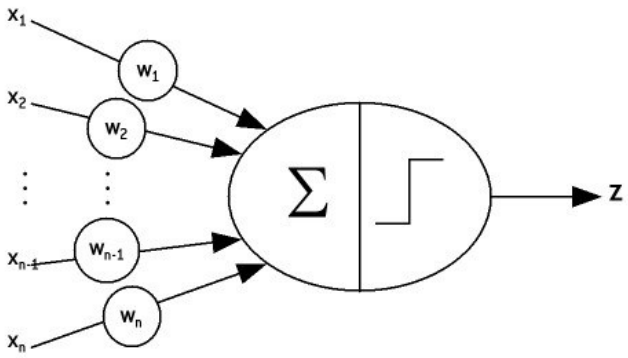


Figure 1.8. Basic artificial neuron

In Figure 1.8 a basic artificial neuron (or TLU) is shown [29]. The threshold relation for the basic neuron is written as

$$z = \begin{cases} 1 & \text{if} \quad a \geq \theta \\ 0 & \text{if} \quad a < \theta \end{cases} \tag{1.4}$$

where

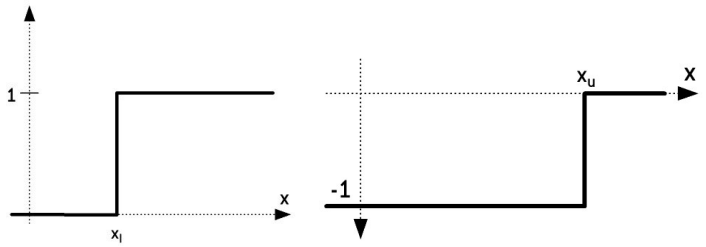
$$a = w_1 x_1 + w_2 x_2 + \cdots + w_n x_n \tag{1.5}$$

As stated before rule neurons are connected to four boundary neurons. Upper boundary neurons have -1 weight values while lower boundary neurons have +1 weight values. Let us assume that X and Y are the inputs of the system and that for a rule R the upper boundary neurons have x_u for X input and y_u for Y input. Lower boundary neurons are x_l for X and y_l for Y input. By implementing relation (1.4), relation (1.6) is obtained for lower boundary neurons.

$$z_{x_l} = \begin{cases} 1 & \text{if} \quad \mathrm{X} \geq x_l \\ 0 & \text{if} \quad \mathrm{X} < x_l \end{cases} \tag{1.6}$$

Additionally, for the upper boundary neuron, the relation obtained is (1.7)

$$z_{x_u} = \begin{cases} -1 & \text{if} \quad X \geq x_u \\ 0 & \text{if} \quad X < x_u \end{cases} \tag{1.7}$$



(a) (b)

Figure 1.9. Weighted (a) lower and (b) upper neuron output characteristics

If the upper and lower boundary relations are summed, relation (1.8), which is shown in Figure 1.10, can be obtained

$$z_x = \begin{cases} 0 & \text{if} \quad X \leq x_l \\ 1 & \text{if} \quad x_l \leq X \leq x_u \\ 0 & \text{if} \quad X < x_u \end{cases} \tag{1.8}$$

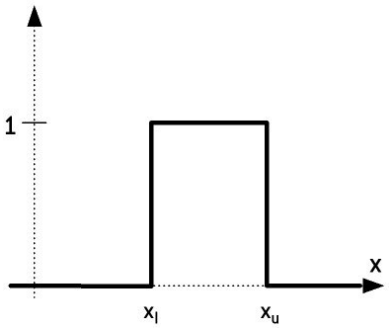


Figure 1.10. Membership function obtained in expression (1.8)

Obtained membership function is in a basic crisp form due to the simple activation function used in neurons. If another type of unipolar activation function is used then the obtained membership function will be shaped accordingly. If the same activation function is used for all first layer neurons then the obtained membership functions will mostly be in

a symmetrical form. Some example activation functions and their corresponding membership functions are shown in Figure 1.11.

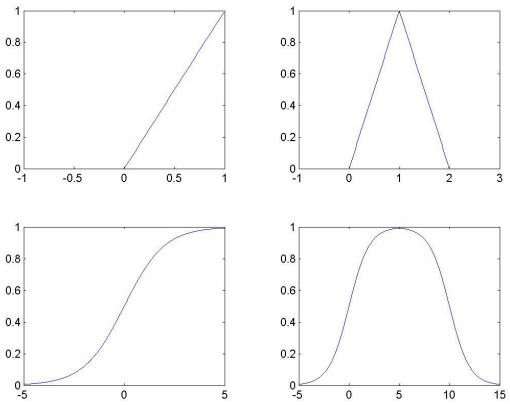


Figure 1.11. Example activation function to membership function conversions

1.1.2. T-norm Operator

A rule neuron (second layer neuron) in the network generates output according to its activation function. For example, the basic neuron in Figure 1.8 generates output while the sum of all inputs is greater than the threshold value. This condition prevents us from implementing the basic MIN operation, in which the minimum valued input is selected, because MIN operation requires a comparison, however no comparison operation takes place in neurons. Nevertheless instead of a basic MIN operation a t-norm operation could be defined for implementing the intersection operation.

Threshold block provides fuzzified inputs x_1, x_2, …, x_n and y_1, y_2, …, y_n where the weight values of corresponding synapses to second layer can be -1,+1 or 0. Let us assume a rule neuron having a configuration as basic neuron configuration shown in Figure 1.8, and lower boundary neurons are x_1, y_1 and the upper boundary neurons are x_2 and y_2; and moreover their activation functions are $A_{x_l}(X)$, $A_{x_u}(X)$, $A_{y_l}(X)$ and $A_{y_u}(X)$, respectively. Since the membership functions are basically the sum of the upper and lower boundary neuron activation functions in corresponding input dimension, X or Y, the fuzzified X and Y inputs can also be written as

$$X_{fuzzified} = A_{x_l}(X) + A_{x_u}(X)$$
$$Y_{fuzzified} = A_{y_l}(Y) + A_{y_u}(Y) \tag{1.9}$$

Since the rule neuron will produce +1 when only sum of the input values are greater than its threshold value, which is +1 in this case, the threshold relation or activation function for rule neurons can be written as (1.10).

$$z = \begin{cases} 1 & \text{if} \quad X_{fuzzified} + Y_{fuzzified} \geq 1 \\ 0 & \text{if} \quad X_{fuzzified} + Y_{fuzzified} < 1 \end{cases} \tag{1.10}$$

The relation (1.10) leads us to a t-norm operator instead of a basic MIN operator. Since the sum of fuzzified inputs can only be maximum +2 due to unipolar characteristic of activation functions, this t-norm operator can be written as

$$\mathrm{T}_{\min}(X_{fuzzified}, Y_{fuzzified}) = \max(0, X_{fuzzified} + Y_{fuzzified} - 1) \tag{1.11}$$

The obtained t-norm in (1.11) is identical to the t-norm operator so called Lukasiewicz's t-norm which is shown as $\mathrm{T}_{\mathrm{Luka}}(a,b) = \max(0, a+b-1)$ [8]. The graphical example is shown in Figure 1.12. In Figure 1.12 membership function, which is obtained in (1.8), is used for a linguistic term. As it can be seen there is output, which is also equal to the intersection of two membership functions, while only the sum of two membership functions is greater than the threshold value +1.

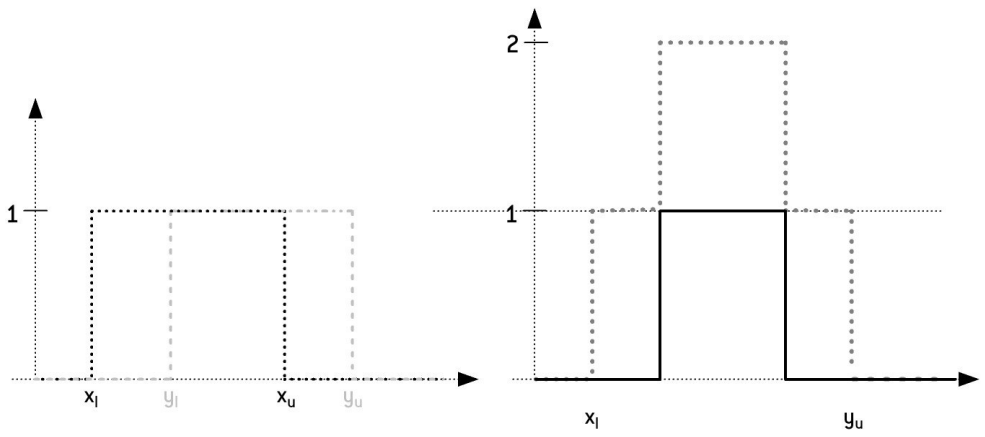


Figure 1.12. T-norm operation

1.1.3. Example Fuzzy Rule Mapping

In order to understand system properties further, an example fuzzy mapping operation is carried out in this section.

Let us have four neurons for each dimension

Let us have two fuzzy sets *X* and *Y* and four linguistic terms *A*, *B*, *C* and *D*, which have triangular membership functions as shown in Figure 1.13

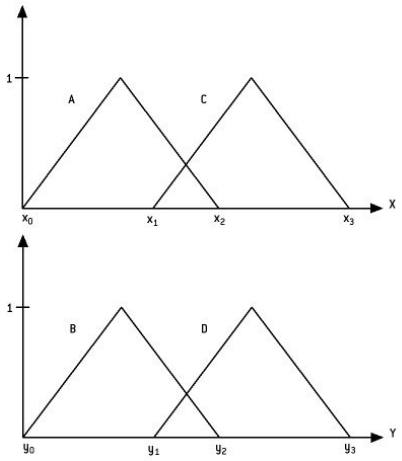


Figure 1.13. Example membership functions

Let us have following rules:

$$If\ X=A\ and\ Y=B\ then\ Z=v_1 \tag{1.12}$$

$$If\ X=C\ and\ Y=D\ then\ Z=v_2 \tag{1.13}$$

First of all we should try to find out the boundary values of two rules (or two areas), which will be called as V_1 and V_2, for the architecture. These boundary values depend on the activation functions of the first layer of neurons in the architecture. In this example we consider the activation functions set up same functions as membership functions shown in Figure 1.13.

Since the activation functions set up a function same as the membership functions, the lower boundary values of area V_1 are the minimum values of membership functions A and B. In other words V_1 area is bordered by two lower boundary neurons which are x_0 on X axis and y_0 on Y axis. By performing same logic the upper boundary neurons are achieved as x_2 and y_2. We obtain for area V_1, the boundary neurons are x_0, x_2, y_0 and y_2 , and for area V_2, they are x_1, x_3, y_1 and y_3.

After finding the boundary neurons, V_1 and V_2 areas can be shown as in Figure 1.14 on the input space.

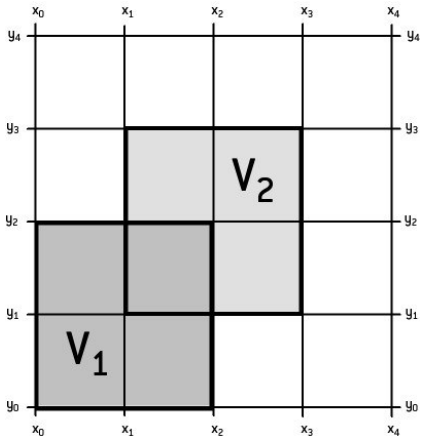


Figure 1.14. Obtained input space

The last operation we have to perform is assigning the weight values to the synapses between second and normalizer block as discussed before. Since we have v_1 for V_1 area and v_2 for V_2 area the weight values are v_1 and v_2, respectively.

Now we can map the values onto the neural network structure shown in Figure 1.7. The obtained mapping is shown in Figure 1.15, where x_0=0, x_1=0.25, x_2=0.5, x_3=0.75, y_0=0, y_1=0.25, y_2=0.5, y_3=0.75, v_1=10 and v_2=21. Control surfaces of the example fuzzy inference are shown in Figure 1.16 and 1.17. Figure 1.16 shows the resulting control surface of a Sugeno fuzzy system where the MIN operators are used. This simulation is done via MATLAB Fuzzy Logic ToolBox. Additionally, Figure 1.17 shows the resulting theoretical control surface of designed neuro-fuzzy architecture. Latter simulation is done via theoretical TSK type theoretical MATLAB model coded in the scope of this thesis.

One may notice that the input space actually represents the top view of the control surface without membership grades, which are shown by different colors. Latter feature may become beneficial to use for verification of results.

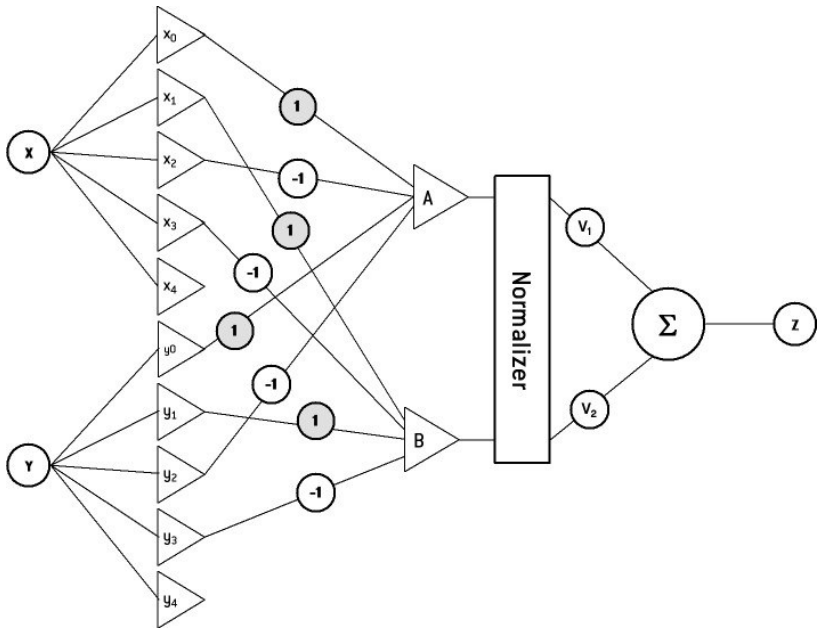


Figure 1.15. Obtained network structure

The general mapping procedure can be summarized as:

- Modify membership functions of the problem sets if needed.
- Find upper and lower boundary neurons for each membership function.
- Set (+1) connections for lower boundary neurons and set (-1) connections for upper boundary neurons to the corresponding second layer neuron. The other neurons remain disconnected.
- Assign consequence values of the rules to the corresponding connections of the second layer neurons as weight values.

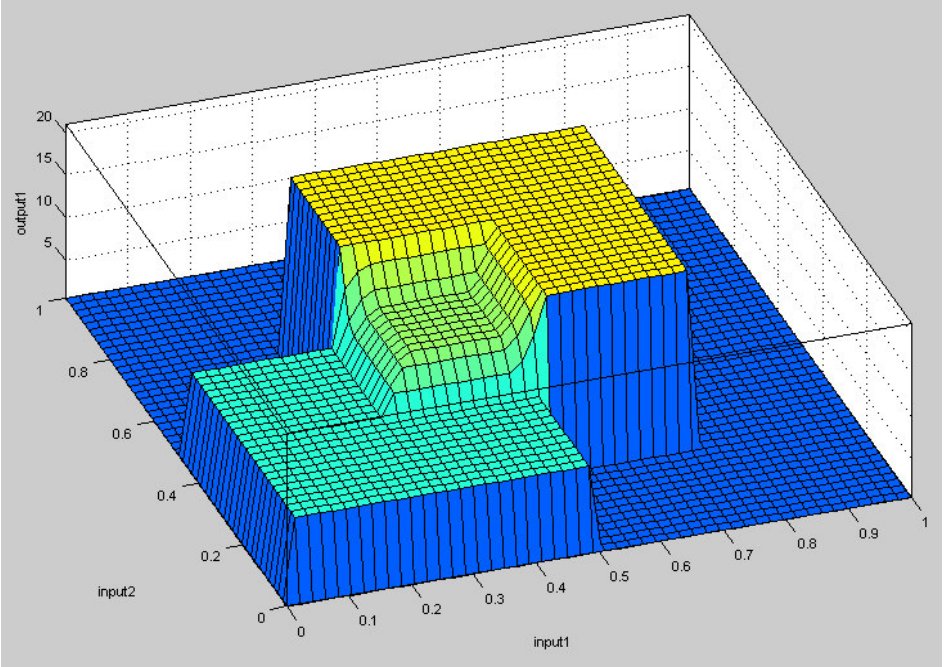


Figure 1.16. Control surface obtained via MATLAB Fuzzy Logic ToolBox

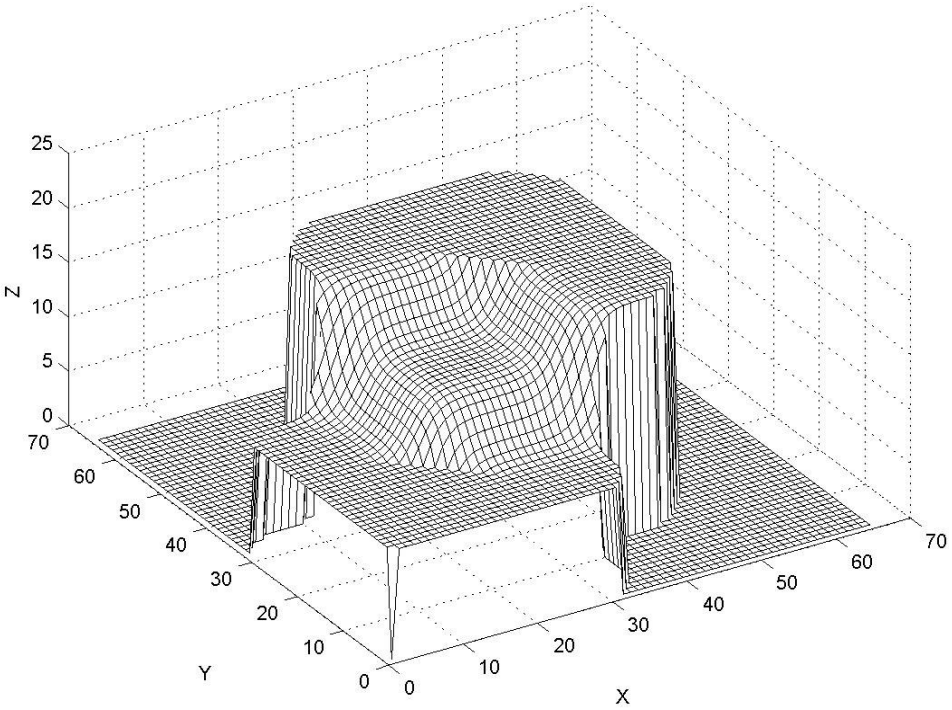


Figure 1.17. Control surface obtained via theoretical MATLAB model

2. MAIN BLOCKS OF THE NEURO-FUZZY SYSTEM IMPLEMENTATION

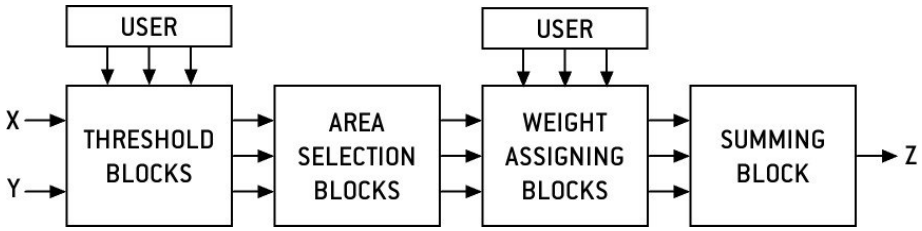


Figure 2.1 Block diagram of neuro-fuzzy system without normalization block

For actual circuit implementation of the introduced system, initially all main building blocks must be determined The block diagram of the neuro fuzzy architecture presented without the normalization block can be shown as in Figure 2.1. The architecture will be implemented for two dimensional input spaces, therefore it supports two inputs, labeled as X and Y and the output is called Z. The architecture is constituted of the following blocks; threshold blocks which correspond to the first layer in the neural network architecture, area selection blocks which correspond to the second layer neurons, weight assigning block which performs assignment of weight values to last layer synapses and summing block for summing the weighted values and obtaining the defuzzified crisp output. Later, the normalization block will be added to this diagram. Prior to adding the normalization block, some possible locations should be analyzed. Subsequent to this analysis the normalization block will be added to the architecture and the final block diagram will be shown.

2.1. Threshold Block

Threshold block corresponds to the first layer of the neural network architecture and the purpose of this block is to act as a fuzzifier and to determine boundary values of the desired areas on the input plane. There is one main block for each input. The block is composed of neuron blocks which contain activation functions used in fuzzification process.

A threshold block also has to have the ability of processing some user data, because of boundary neuron selection operations. The user should be able to change or determine the boundary neurons for desired regions on the input space.

2.2. Area Selection Block

Area selection block corresponds to the second layer neurons which determine if the area is selected. Area selection block can also be called rule neuron block since there is one block for one rule neuron. If there are n neurons for each dimension of a two dimensional input space, there can be maximum n^2 selectable region on the input plane. Therefore, the possible maximum number of area selection blocks in the system is equal to n^2.

Area selection block must generate a signal if each input is inside the boundaries of the area that is represented by the block. In other words it must generate a signal while the area is selected. This block is also responsible for implementation of the t-norm operation (1.9). Area selection block can be seen as processing unit of the controller.

2.3. Weight Assigning Block

Weight assigning block corresponds to the synapses between second and third layer. These blocks must have the ability to take user data as weight value and multiply them with the block inputs. Since area selection blocks can be seen as second layer neurons and weight assigning blocks as synapses, each area selection block has to be connected to only one weight assigning block as in the neural architecture neuron-synapse connection in Figure 1.7. Therefore, the number of weight assigning block is equal to the number of area selection blocks in the system. One may see this block as a multiplier where the user can only alter the value of the multiplier.

2.4. Summing Block

Summing Block sums the block inputs and generates the result as a signal which is the output of the whole system. This block acts as an adder.

2.5. Normalization Block

Normalization and summing block together form a defuzzification block which performs weighted average operation (2.1), where $x_i(i=1,2,..,n)$ values are the desired output values or in other words rule outputs, which are assigned in the weight assigning block by user, and $w_i(i=1,2,..,n)$ values are input values which are fuzzified in the threshold block.

$$f(x,w)=\frac{\sum x_i w_i}{\sum w_i} \tag{2.1}$$

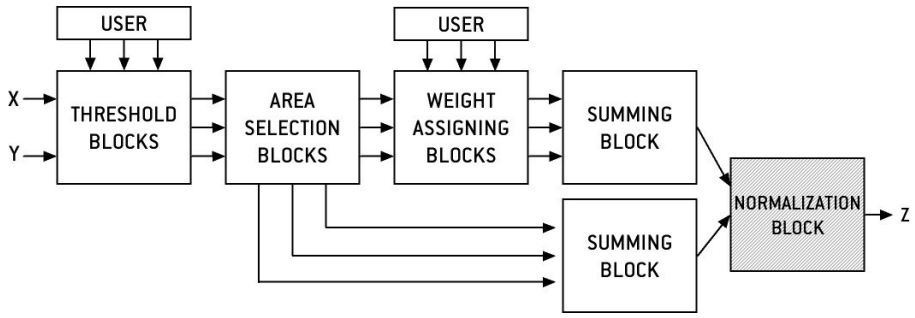


Figure 2.2. The first block diagram of system with normalization block

It is important to note that the weighted sum approach is also possible. Various fuzzy controllers reported in the literature implement weighted sum operation instead of weighted average [19][33]. However weighted average operation will be implemented in this thesis.

It is stated earlier that the additional normalization block can be placed on some different locations on the block diagram. The first possible location of the normalization block is after the weight assigning block as shown in Figure 2.2. It can be noted that this type of normalization block also needs an extra summing block for the summing operation of fuzzified inputs. For this realization it is needed to have a normalization block that provides a simple division function as shown in expression (2.2).

$$O=\frac{A}{B} \tag{2.2}$$

If $A = \sum x_i w_i$, the sum of weighted inputs (*xw*), and $B = \sum w_i$, the sum of weights (*w*), then the desired normalization operation (2.1) is achieved.

In analog MOS circuit implementations, division has always been a difficult operation to be implemented in terms of time and area. Many of the reported fuzzy controllers impose the condition that the denominator in expression (2.1) assumes the value 1 to avoid the division, or recur to the use of global normalization loops [30] [31], but this approach can lead to limited accuracy and stability problems.

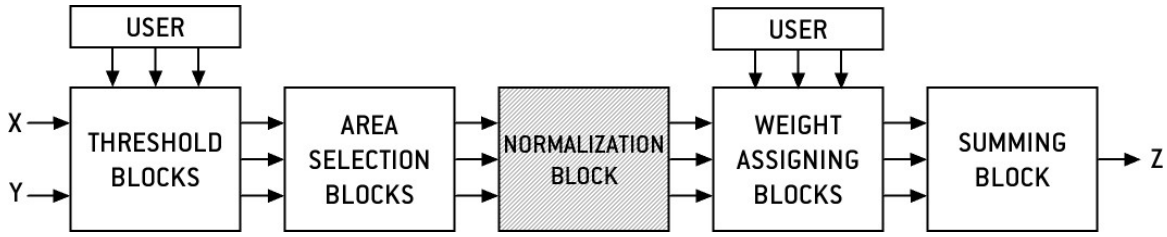


Figure 2.3. The second block diagram of system with normalization block

Another possible location for normalization block is between the area selection block and the weight assigning block as shown in Figure 2.3. For this sort of placement it is needed to have a multiple output normalization block which provides following relation for each input:

$$O_1 = \frac{x_1}{\sum x_i} \quad O_2 = \frac{x_2}{\sum x_i} \quad \cdots\cdots \quad O_n = \frac{x_n}{\sum x_i} \tag{2.3}$$

As shown above, this type of normalization block generates normalized outputs for each corresponding input; and these outputs are later introduced to the weight assigning block for performing the weight value multiplication. Later, multiplied input values are summed by the summing block and the desired function is obtained. This kind of normalization block can be constructed as a ‘Gilbert Normalizer’ [13], which is simple to be implemented and has been used in various proposed neural network and fuzzy system implementations [9][10][11][12]. This configuration is implemented in this thesis.

3. CIRCUIT REALIZATION

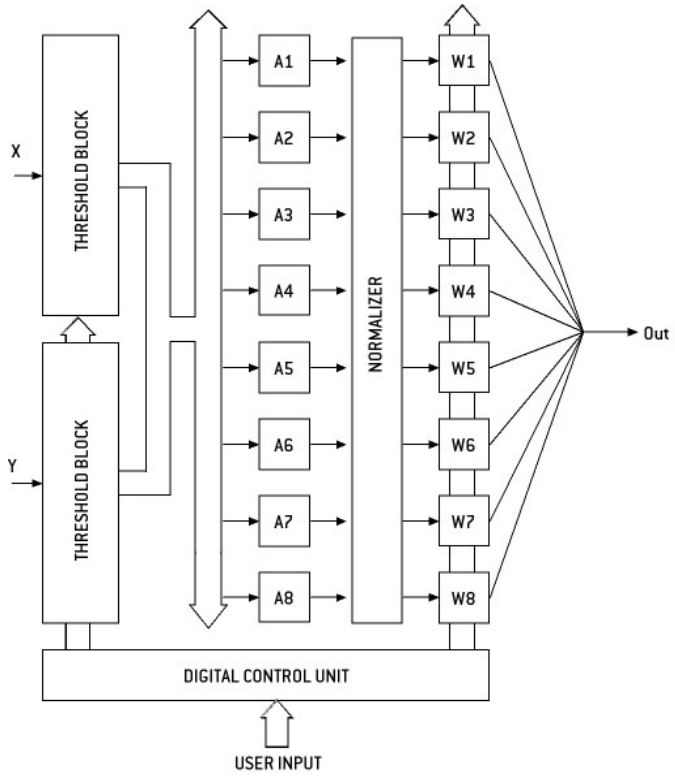


Figure 3.1. Chip architecture

The architecture designed for the chip implementation is shown in Figure 3.1. The chip architecture contains all the blocks previously discussed, switches and some control blocks for mapping boundary values and desired outputs. Main blocks, threshold, area selection, normalization and weight assigning blocks are analog CMOS implementations, while control blocks are digital implementations. It has decided that the first layer has 64 neurons. Additionally eight area selection block is present on the chip. Therefore eight weight assigning blocks are implemented.

Connections between first and second layer and weights are determined via the control unit. Therefore, the user is able to manipulate desired regions.

There are no non volatile memory elements in the chip. Additional external ROM may be needed for storing weight values or boundary neuron indexes permanently. For

correct operation it is important to set boundary neurons and weight values first. Until this setting is done, no correct value is observed. However once those values are set, internal memory holds them until user resets the chip or user changes them via the digital control block.

It should be noted that the chip is designed for realization of two dimensional input planes where the inputs are labeled as X and Y, respectively. The inputs are voltage while the output is current, but internal structure is working in current mode for the most part.

AMS 0.35 μm process is used in the design of circuits. SPICE codes, VHDL codes and layouts of all blocks can be found in appendix.

3.1. Threshold Block

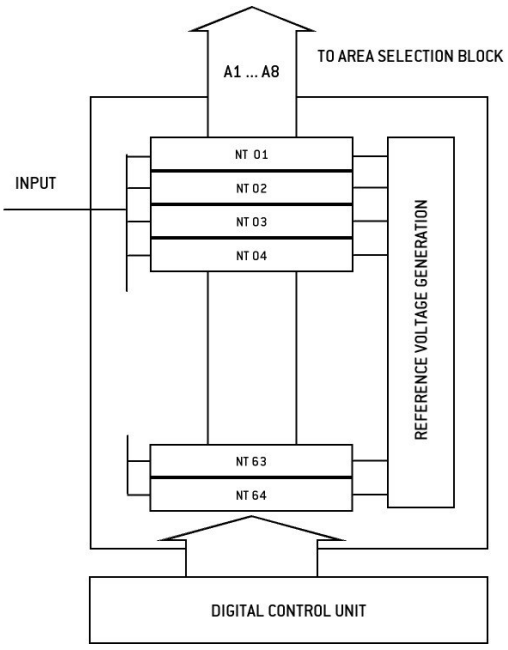


Figure 3.2. Threshold block

Threshold Block represents the first layer of network. For one input there is one threshold block in the architecture. The threshold block architecture is shown in Figure 3.2. The block consists of neuron blocks called NT Block, reference generation block and a digital control block for processing user data. Each NT Block represents a neuron in the first layer. Reference generation block generates the threshold values for all NT blocks.

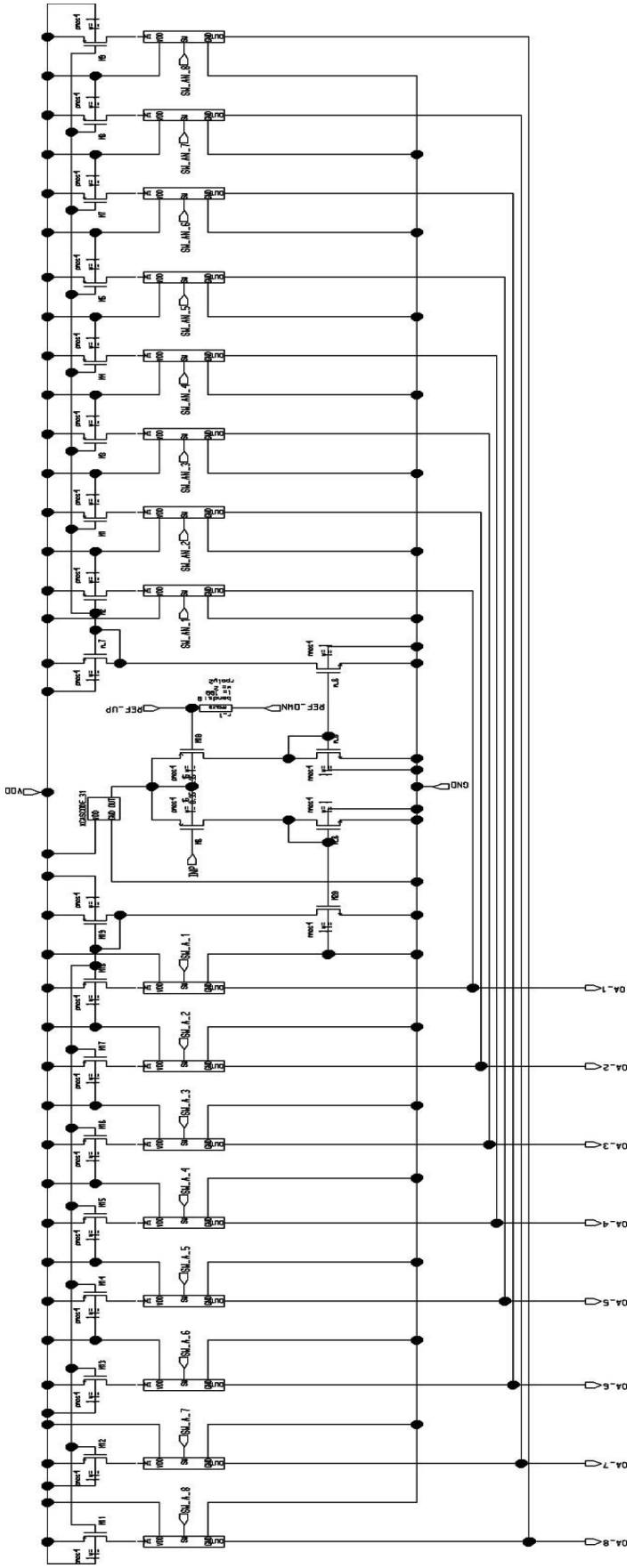


Figure 3.3. Neuron block (NT) circuitry

3.1.1. Neuron Circuitry

The neuron circuitry based on a MOS differential pair, where one input of differential pair is one of the system inputs and the other input is the threshold value (also called reference voltage). The MOS differential pair circuit has been widely used as a fundamental building block in most of the analogue implementations of neuron circuits and fuzzifier circuits [9][14][15][16][19]. In general, input signals to the pair are applied differentially to ensure that the nonlinearities of the circuits are absolutely symmetrical with respect to zero and also to minimize the impact of noise. Output signals, however, are taken either as voltages or currents, using the appropriate circuit technique, depending on the particular application. In the neuron realization in this thesis, the outputs are taken as currents and the threshold values of these neurons are determined by the reference voltage applied to one input of the differential pair. Connections to the second layer are provided by current mirrors, which simply copy the drain currents that are denoted as (+1), lower boundary and (-1), upper boundary node currents, respectively. Additionally user can control those connections by simply setting switches, which are placed on the outputs, on and off. Switches are basic transmission gates. Switch setting operation is controlled by a digital control block, which will be discussed later. Since the number of area selection block is eight, each threshold block has eight (-1) and eight (+1) output ports; and the inputs are eight (-1) switch control, eight (+1) switch control inputs and a reference voltage input.

Example simulations of (+1) node (M1 drain current) and (-1) (M2 drain current) node are shown in Figure 3.4 and Figure 3.5, respectively. Bias current which represents maximum output value (+1 in theory) is selected as $10\,\mu$ A.

Previously in section 1.1.1, it was discussed how the membership functions were generated in the structure. It was stated that the membership function is bounded to the activation function. Figure 3.4 shows a typical hyperbolic characteristic which could be used to construct membership function by performing same computations in section 1.1.1. Therefore M1 drain current could be called as (+1) output or lower boundary output. (+1) output is multiplied by -1 for obtaining negative activation function. However the (-1) output, or lower boundary output, is not obtained by simply multiplying the output by -1.

M2 drain current, which has characteristic shown in Figure 3.5, will be considered as (-1) output. Expected (-1) output has inverted characteristics of (+1) output, which would have a maximum value 0 A and minimum value -10 μA. Since (+1) characteristic is not multiplied by -1, it is not possible to obtain expected output. On the other hand one may notice the general form of (-1) node output is exactly the same as expected (-1) output. This characteristic will be used as negative output. In this situation, the characteristics could be represented as 10 μA up shifted version of expected function. This change in the characteristic has no major effect on the system and calculations. It only causes the threshold value of area selection neuron to increase by 30 μA

.

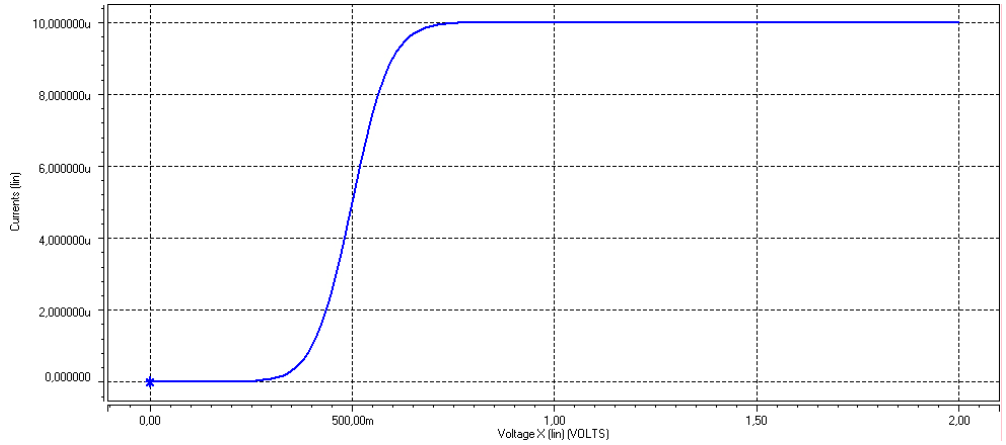


Figure 3.4. Lower boundary output, reference voltage is 0.5 V

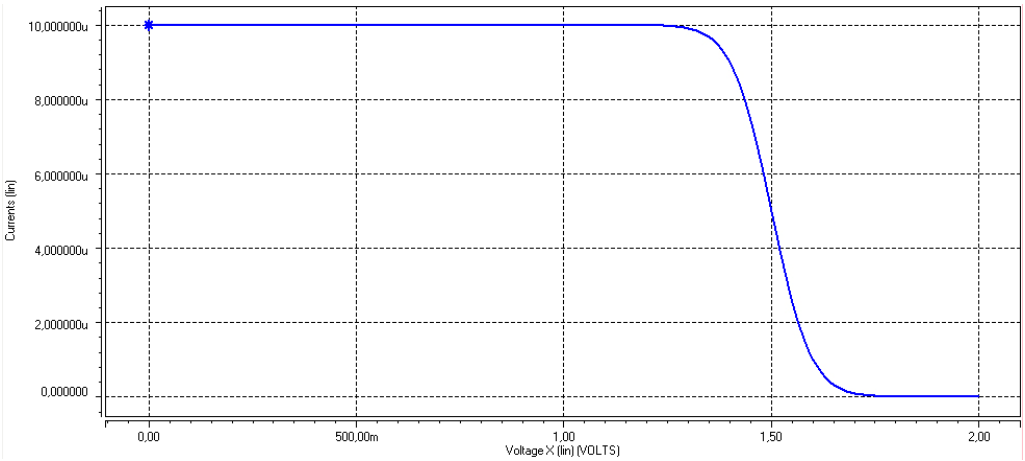


Figure 3.5. Upper boundary output, reference voltage is 1V

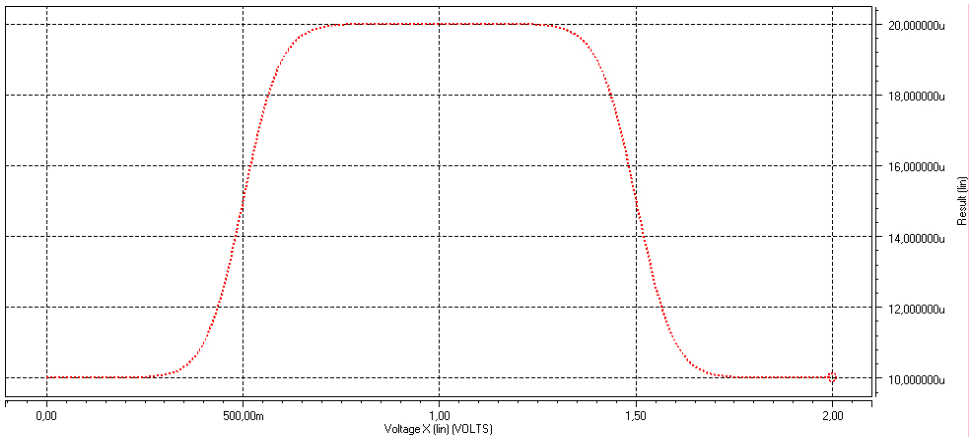


Figure 3.6. Obtained membership function in neuron circuitry

The membership function in the system is the sum of (+1) and (-1) outputs. If the two output currents are summed, the membership function shown in Figure 3.6 is obtained. As expected, the membership function is in the [10 μ A, 20 μ A] range. The shape of the membership function is related to reference input and the size of differential pair transistors. It is very important to calculate the expression of this membership function, since user is going to use this membership function with his/her fuzzy mapping process. This function is basically the sum of dc transfer characteristics of the MOSFET differential pair transistors.

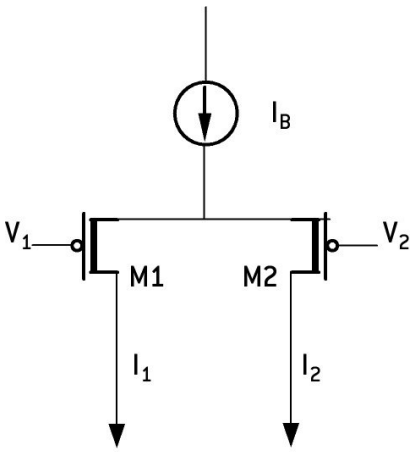


Figure 3.7. PMOS Differential pair

The dc transfer characteristics of the MOSFET differential pair shown in Figure 3.7 can be obtained using the circuit equations. We will first assume M1, M2 transistors are in saturation region. If two transistors are matched, we can write (3.1) and (3.2).

$$I_1 = \frac{k_p'}{2}\frac{W}{L}(V_{SG1} + V_{Th})^2 \tag{3.1}$$

$$I_2 = \frac{k_p'}{2}\frac{W}{L}(V_{SG2} + V_{Th})^2 \tag{3.2}$$

Taking the square roots of Equation (3.1) (3.2) and subtracting two equations, we obtain

$$\sqrt{I_1} - \sqrt{I_2} = \sqrt{k_p'\frac{W}{L}}(V_{SG1} - V_{SG2}) = \sqrt{k_p'\frac{W}{L}}V_D \tag{3.3}$$

, where $V_D = V_{SG1} - V_{SG2}$ is the differential-mode input voltage. If $V_D > 0$, then $V_{G1} > V_{G2}$ and $V_{SG1} > V_{SG2}$, which implies that $I_1 > I_2$ .Since

$$I_1 + I_2 = I_B \tag{3.4}$$

Then equation (3.3) becomes

$$(\sqrt{I_1} - \sqrt{I_B - I_1})^2 = (\sqrt{k_p'\frac{W}{L}} \cdot V_D)^2 = k_p'\frac{W}{L}{V_D}^2 \tag{3.5}$$

If we square both sides of this equation, we develop the quadratic equation

$${I_1}^2 - I_B I_1 + \frac{1}{4}(I_B - k_p'\frac{W}{L}{V_D}^2) = 0 \tag{3.6}$$

Applying the quadratic formula, rearranging terms, and noting that $I_1 > I_Q/2$ for $V_D>0$, we obtain

$$I_1 = \frac{I_B}{2} + \sqrt{\frac{k_p'WI_B}{2L}} \cdot V_D\sqrt{1 - (\frac{k_p'W}{2I_BL}){V_D}^2} \tag{3.7}$$

Using equation (3.4), we find that

$$I_2 = \frac{I_B}{2} - \sqrt{\frac{k_p' W I_B}{2L}} \cdot V_D \sqrt{1 - (\frac{k_p' W}{2 I_B L}) V_D^{\,2}} \tag{3.8}$$

Normalized drain currents are

$$I_1 = \frac{1}{2} + \sqrt{\frac{k_p'}{2 I_B} \frac{W}{L}} \cdot V_D \sqrt{1 - (\frac{k_p'}{2 I_B} \frac{W}{L}) V_D^{\,2}} \tag{3.9}$$

$$I_2 = \frac{1}{2} - \sqrt{\frac{k_p'}{2 I_B} \frac{W}{L}} \cdot V_D \sqrt{1 - (\frac{k_p'}{2 I_B} \frac{W}{L}) V_D^{\,2}} \tag{3.10}$$

and if $V_D \le -\sqrt{\frac{I_B L}{k_p' W}}$ M1 turns off and I_2 becomes equal to I_B and if $V_D \ge \sqrt{\frac{I_B L}{k_p' W}}$ M2 turns off and I_1 becomes equal to I_B.

As it can be seen in Figure 3.8 some hyperbolic characteristic is achieved. Expression (3.9) and (3.10) show how this characteristic changes proportional to size of transistors and reference voltage. Size increase causes wider membership functions while smaller sizes result tighter membership functions.

The subthreshold characteristics of differential pair should also be investigated. Let us assume M1 and M2 working in subthreshold region this time. By using subthreshold drain current equation of PMOS transistor, we can write

$$I_1 = I_0 e^{\frac{K V_1}{V_T}} e^{\frac{V}{V_T}} \tag{3.11}$$

$$I_2 = I_0 e^{\frac{K V_2}{V_T}} e^{\frac{V}{V_T}} \tag{3.12}$$

And from (3.4), we can write I_2 as

$$I_2 = I_1 e^{\frac{K(V_2 - V_1)}{V_T}} \tag{3.13}$$

If we write V_2-V_1 as V_D, we can obtain

$$I_2 = (I_B - I_2) e^{\frac{KV_D}{V_T}} \tag{3.14}$$

If further calculations are done, we obtain I_2 as

$$I_2 = I_B \frac{e^{\frac{KV_D}{V_T}}}{1 + e^{\frac{KV_D}{V_T}}} = I_B \frac{1}{1 + e^{\frac{-KV_D}{V_T}}} \tag{3.15}$$

and I_1 as

$$I_1 = I_B \frac{1}{1 + e^{\frac{KV_D}{V_T}}} \tag{3.16}$$

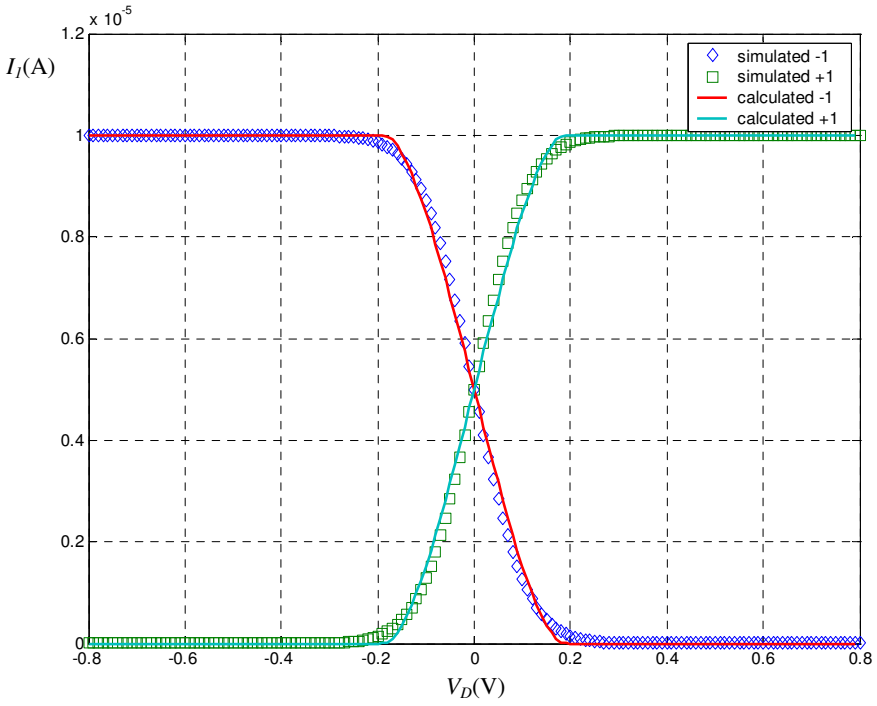


Figure 3.8. Saturation characteristic of MOS differential pair W/L=15 μ /0.35 μ

Comparison of HSPICE simulation versus MATLAB simulation is shown in Figure 3.9.

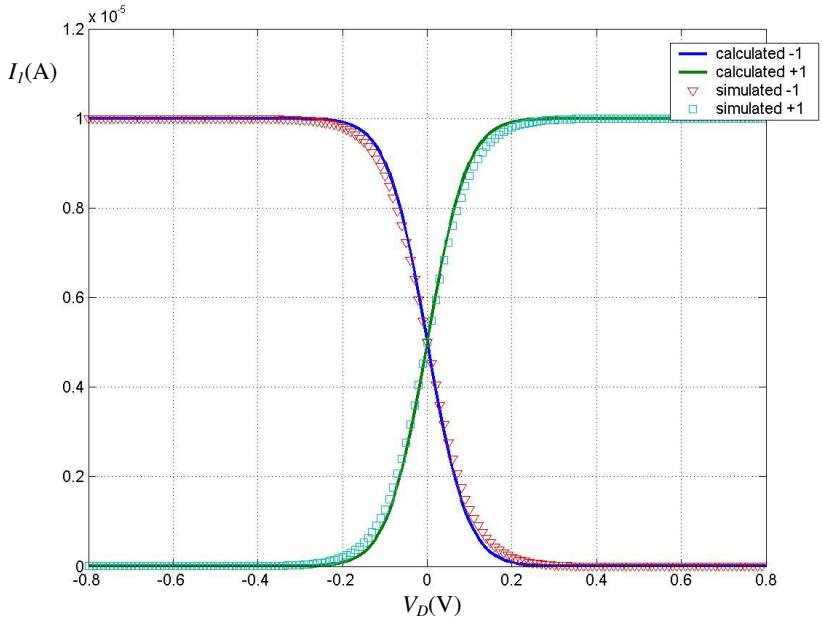


Figure 3.9. Subthreshold characteristic of differential pair, W/L=40 μ /0.35 μ

If we take some further calculations we can obtain also following results for drain currents of M1 and M2 transistors

$$I_1 = \frac{I_B}{2}\left(1 - \tanh\left(\frac{KV_D}{V_T}\right)\right) \tag{3.17}$$

$$I_2 = \frac{I_B}{2}\left(1 + \tanh\left(\frac{KV_D}{V_T}\right)\right) \tag{3.18}$$

After these calculations, we can obtain following definition for membership functions in the first layer of the system.

For a linguistic term L_ϕ where its membership function is generated by a lower boundary neuron ϕ_l and an upper boundary neuron ϕ_u, following relations can be written for the system:

If differential pair transistors are in subthreshold region

$$\phi_{fuzzified} = 5\mu\left(2 + \tanh\left(K\frac{(\phi-\phi_l)}{V_T}\right) - \tanh\left(K\frac{(\phi-\phi_u)}{V_T}\right)\right) \tag{3.19}$$

If the transistors are in saturation region:

$$\phi_{fuzzified} = I_B + \sqrt{\frac{k_p'WI_B}{2L}}(A-B) \tag{3.20}$$

$$A = (\phi-\phi_l)\sqrt{1-(\frac{k_p'W}{2I_BL})(\phi-\phi_l)^2} \tag{3.21}$$

$$B = (\phi-\phi_u)\sqrt{1-(\frac{k_p'W}{2I_BL})(\phi-\phi_u)^2} \tag{3.22}$$

where if $(\phi-\phi_l) \le -\sqrt{\frac{I_BL}{k_p'W}}$, $\phi_{fuzzified}$ is equal to I_B and if $(\phi-\phi_u) \ge \sqrt{\frac{I_BL}{k_p'W}}$ and $\phi_{fuzzified}$ is equal to I_B.

Example MATLAB simulation where ϕ_l=1 V and ϕ_u=2 V is shown in Figure 3.9 and 3.10.

When suitable W/L ratio is selected, saturation region characteristic is very close to the subthreshold region characteristic. Figure 3.10 and 3.11 shows this situation. The aim was implementing separate region selection and due to limited voltage interval (0-2.5 V), membership functions should have higher slope value. Considering there is no significant difference between subthreshold characteristic and saturation characteristic if suitable sizing is selected, the saturation region characteristic, where the width of transistors are 15 μm and lengths are 0.35 μm, is chosen. Since the saturation current characteristic is also hyperbolic, an approximate equation might be found. Equation (3.23) may also represent

the obtained membership function. Figure 3.12 shows the approximate characteristic where it is almost identical to the simulated response.

$$\phi_{fuzzified} = 5\mu\left(2 + \tanh\left(10(\phi - \phi_l)\right) - \tanh\left(10(\phi - \phi_u)\right)\right) \tag{3.23}$$

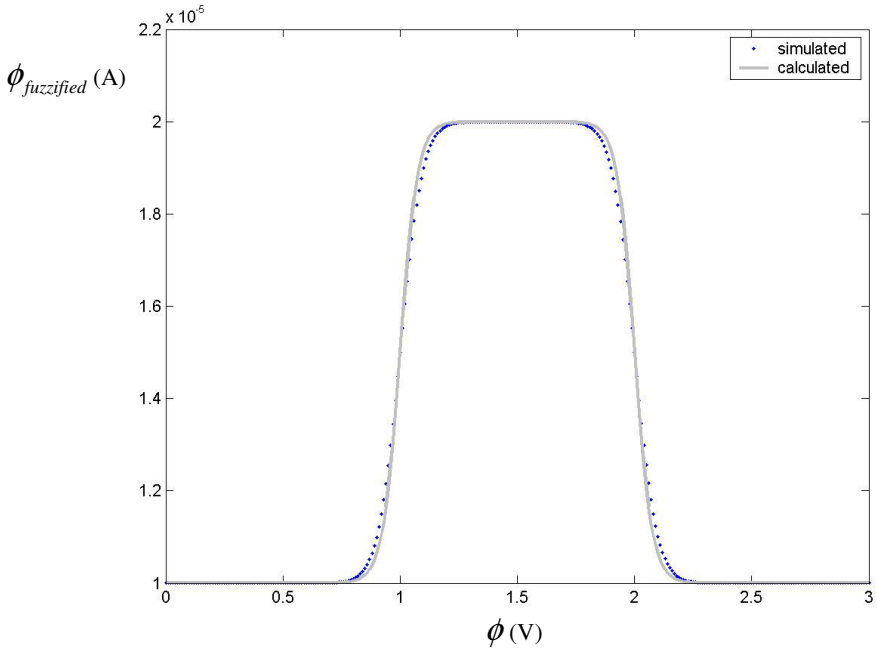


Figure 3.10. Comparison of simulated and theoretical subthreshold membership functions

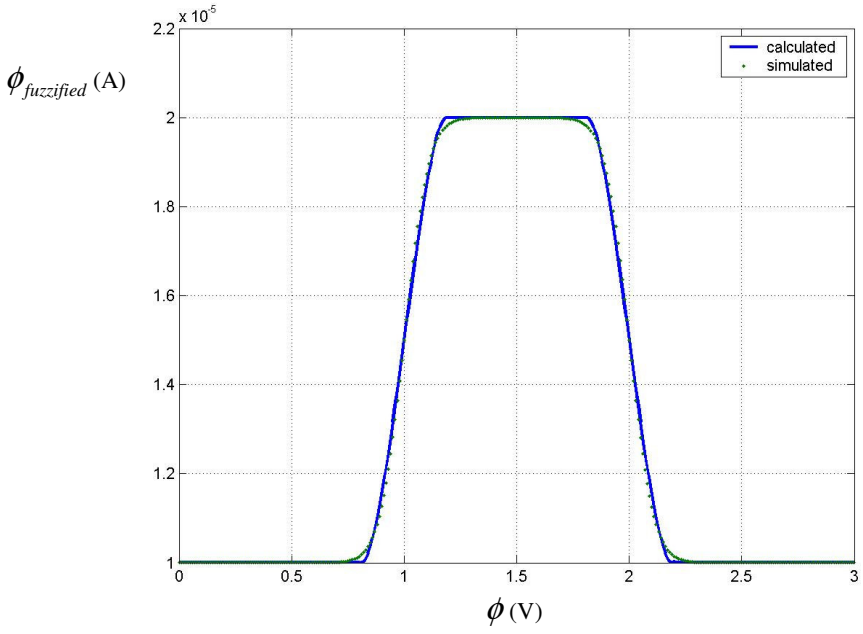


Figure 3.11. Comparison of simulated and theoretical saturation membership functions

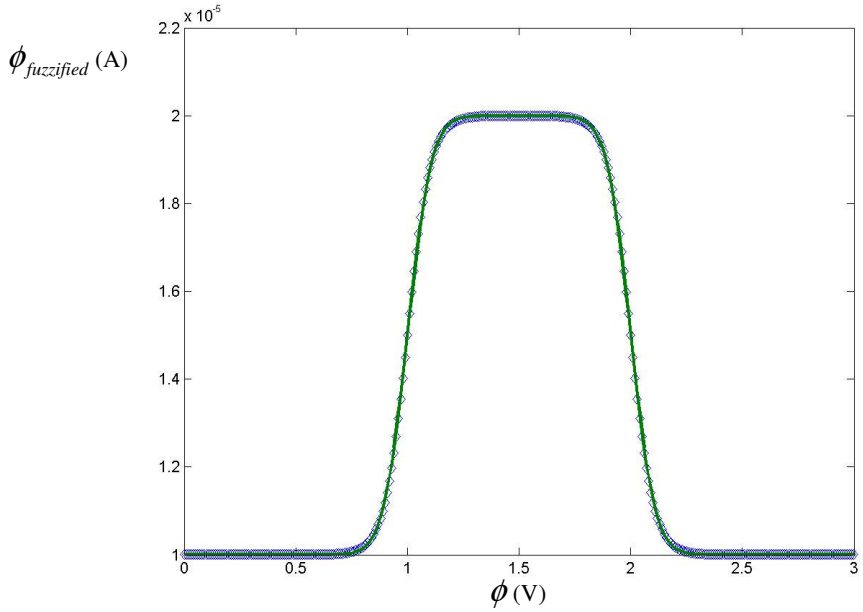


Figure 3.12. Comparison of simulated and approximate membership functions

Expression (3.21) will be assumed as membership function in further calculations for clarity.

3.1.2. Boundary Synapse Connecting

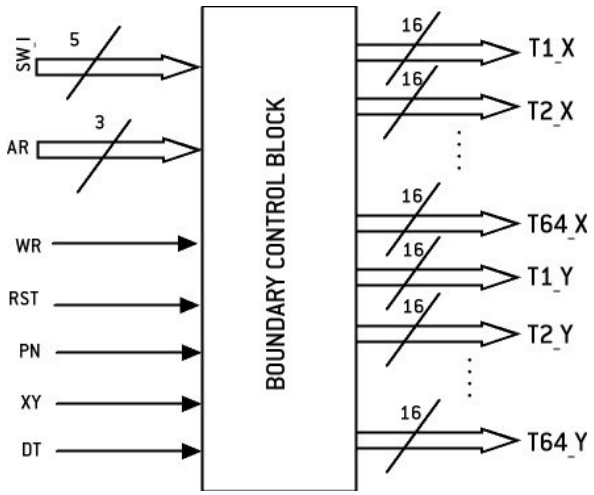


Figure 3.13. Block diagram for the control of the output current

In order to obtain correct area selection operation, only the correct four boundary neurons, which define the boundary of the area, must be connected to corresponding area

neuron. This operation is done by closing appropriate switches between boundary and area neuron blocks. The circuitry for such operation could easily be done by using digital design tools instead of designing RTL level analog counterparts. Designed digital control block diagram for synapse connecting operation is shown in Figure 3.13.

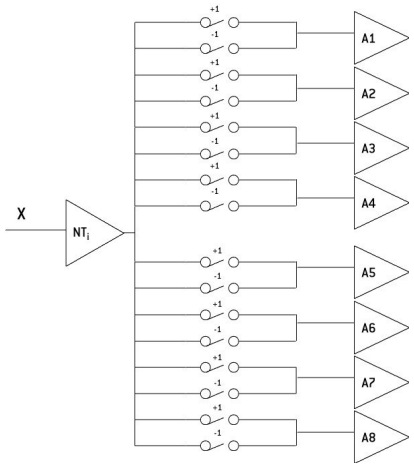


Figure 3.14. Connections between first and second layer neurons

First of all some memory elements should be in the design, since every synapse, which is represented by a switch in Figure 3.14, has to have some value to show if it is ON or OFF. Furthermore these memory elements prevent the user from selecting, setting or connecting the synapses every time whenever chip is restarted. The value that shows if synapse is connecting is logic ‘1’, which is later converted to 3.3 V and sent to the gates of transmission gate switches. When 3.3V applied to the gate, a connection between NT block and area neuron block is obtained. Conversely ‘0’ shows there is no connection.

Assigning ‘1’ and ‘0’ values to the memory is performed by using following block inputs; WR, SW_I, DT, XY, PN, AR and RST.

WR is the write input and if only WR is ‘1’, block accepts and stores inputs, otherwise stored values do not change and additionally user is not able to add new data and/or manipulate stored data. SW_I input determines the neuron which user want to connect. The block takes the SW_I value and converts it to the corresponding decimal index value which will be the neuron index number. Additionally user has to give the input

dimension which this neuron belongs to. This operation is performed via XY input. If XY input is equal to ‘0’, then X dimension neurons will be selectable, otherwise Y dimension neurons become selectable. Also user must set if he/she wants to connect lower (+1) or upper (-1) boundary synapse or in other words user must set if the synapse has +1 or -1 weight value. This operation is done via PN input. If PN is equal to ‘0’ then the lower boundary switch is selected, otherwise upper boundary neuron is selected. Last operation which user must do is the selecting the area selection block or area neuron to which the synapse will connect. User does this operation by setting AR input to desired area’s number. Block converts binary AR input to a decimal index value and connects the corresponding synapse to desired area. When area, dimension and upper/lower boundary is determined the corresponding area is set to DT input value.

Block also contains a RST input which breaks all synapse connections, if it is ‘1’. In other words RST input sets all memory elements to ‘0’.

Block outputs have names according to dimension and neuron order information. X dimension neuron outputs have suffix _X where Y dimension counterparts have _Y in their names. The name formation can be formulated as T(TNO)_(D) where (TNO) is neuron number and (D) input dimension. For example sixth neuron in X dimension is denoted as T6_X. All outputs are 16 bit long. The first eight bit section is the upper boundary connections while last eight bits represent lower boundary connections. The order is MSB to LSB where MSB is the first area LSB is eighth area connection. Here, the outputs can be simply formulated as:

T(TNO)_(D)=“U1 U2 U3 U4 U5 U6 U7 U8 L1 L2 L3 L4 L5 L6 L7 L8” (3.24)

As an example, in order to connect seventh neuron of X input dimension to fourth area selection block as an upper boundary (-1) neuron, the procedure is as follows. First of all WR must not be set to ‘1’ until area, dimension and upper/lower boundary information is introduced to the system. Please note that the introduction order of area, dimension or upper/lower boundary information is not important. AR is set to “011” (three in decimal), SW_I is “0110” (six in decimal), XY is ‘0’ and lastly PN is ‘0’. Then if WR is set to ‘1’ and DT to ‘1’, T7_X output should be observed as “0001000000000000”. Example

simulation of this process is shown in Figure 3.15. Additionally the VHDL simulation of various synapse connection control operations is shown in Figure 3.16.

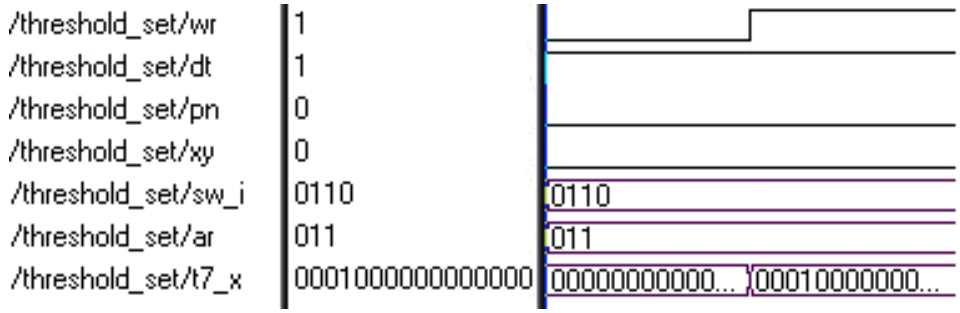


Figure 3.15. Example boundary synapse connection control process (I)

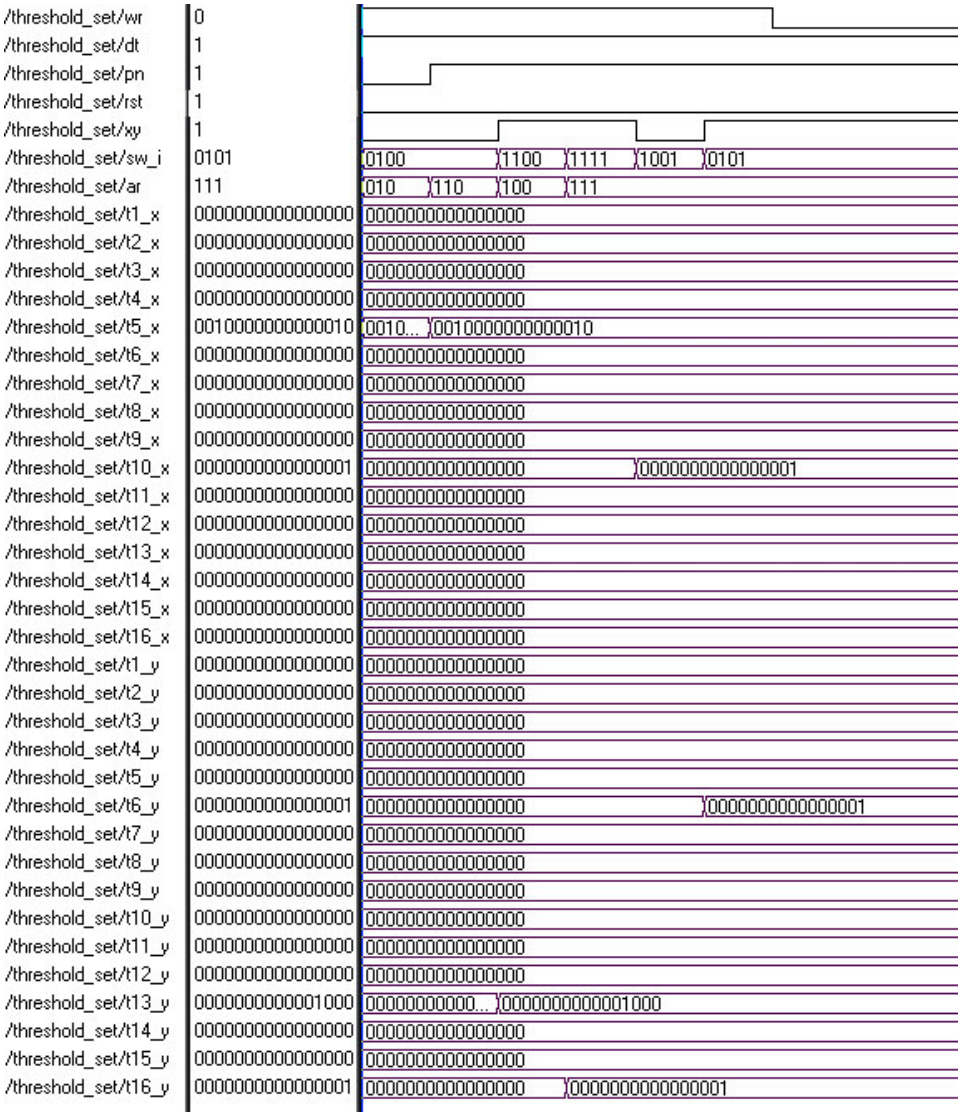


Figure 3.16. Example boundary synapse connection control process (II)

3.2. Reference Voltage Generation

Threshold value of first layer neuron is determined by a reference voltage applied to one input of the neuron circuit's differential pair. Every neuron must have a unique reference voltage between 0 and 2.5 V value range. The voltages may be arranged in a sequence or arbitrarily, however reference voltages will be assigned to every neuron by using following relation:

$$T_{NT_i} = i \cdot \frac{2.5}{64} \quad i = 1, 2, ..., 64 \tag{3.25}$$

where i is the index number and T_{NT_i} is the reference voltage of the corresponding neuron.

The reference voltages can be provided in two ways; in a serial manner or in a parallel manner.

3.2.1. Serial Reference Voltage Generation

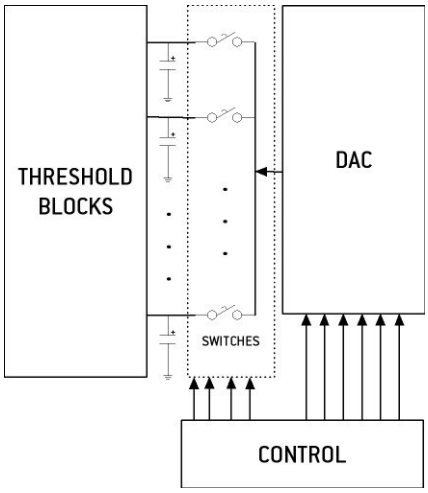


Figure 3.17. Block diagram of serial reference voltage generation

In serial reference generation method, reference voltages are generated serially. It should not be confused with a serial output generation. The general block diagram of serial reference voltage generation is shown in Figure 3.17. The digital control unit controls the

switches and sets the switches on or off in some order. Digital unit also provides a 6 bit digital input to a DAC to generate the corresponding reference inputs to the threshold block neuron circuits. The output voltage of the DAC is charged on a capacitor for storing the reference value until the end of reference generation cycle.

This kind of reference generation needs a clock input and a DAC. Since the system consists of 64 neurons for each dimension and besides two dimension threshold blocks working in parallel, the system output frequency equals to 1/64 of clock frequency. Maximum clock frequency depends on the performance of threshold block.

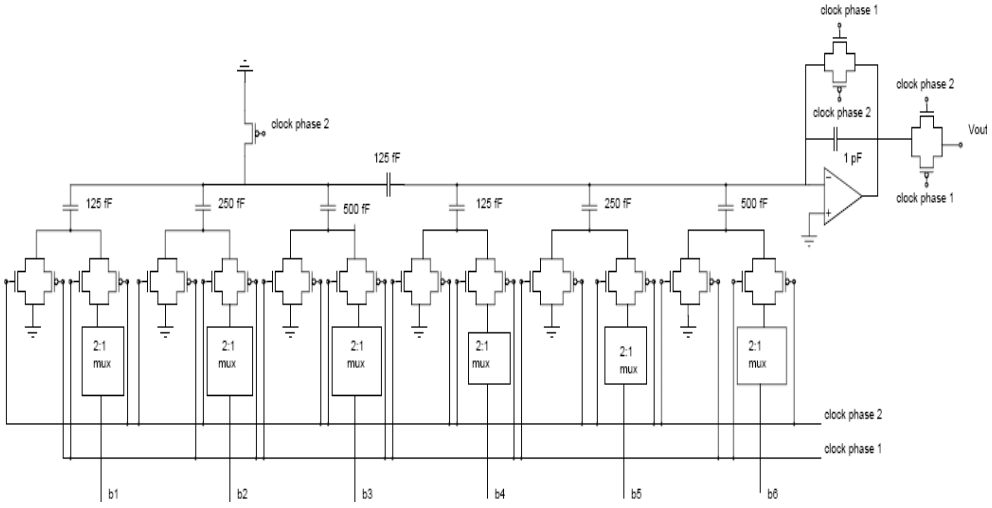


Figure 3.18. Circuit diagram of the transistor based switched capacitor charge redistribution 6-bit DAC

Since the references are simply voltages, DAC which generates those references should be designed in voltage mode. For this manner a switched capacitor charge redistribution 6-bit DAC was designed in a complementary BS project by B. Yelten [17].

Figure 3.18 shows the complete schematic diagram for the two-stage switched capacitor charge-redistribution DAC. At the end, 14 transmission gates, 6 2:1 multiplexers, an operational amplifier and a single PMOS transistor are employed.

The multiplexer, shown in Figure 3.19, is built up on 3 NAND gates and 1 inverter. All transistors except pull down transistor of the inverter possess the same width, namely

10µm, whereas the other transistor has a width of 1 µm. The lengths of the transistors are minimum sized, i.e. it is 0.35 µm.

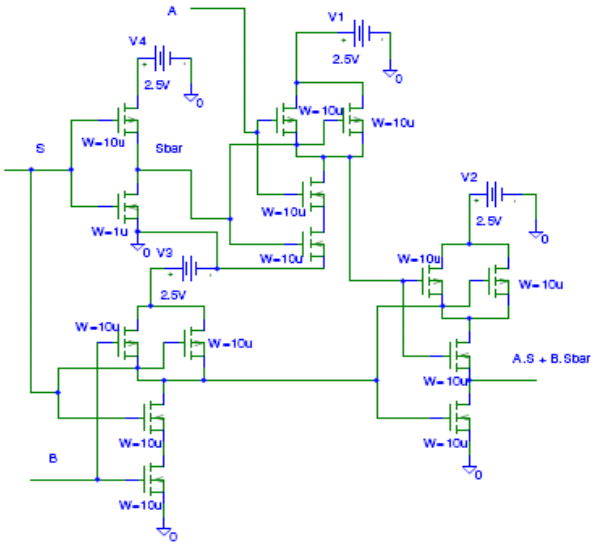


Figure 3.19. Circuit diagram for the 2:1 multiplexer

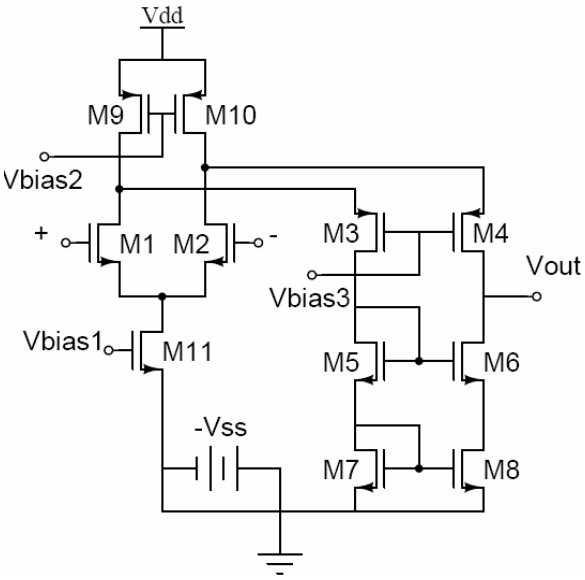


Figure 3.20. Folded cascode amplifier

A folded cascode amplifier, which is shown in Figure 3.20, is used in the output stage of DAC [18].

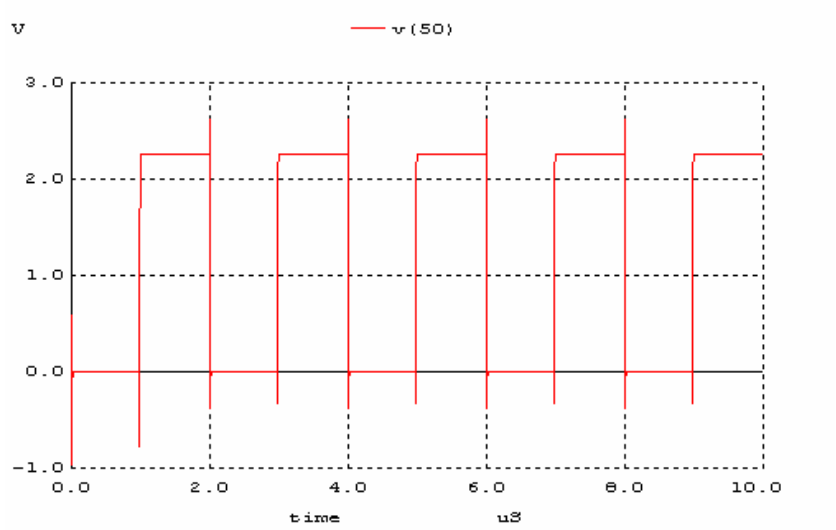


Figure 3.21. Maximum analog output voltage

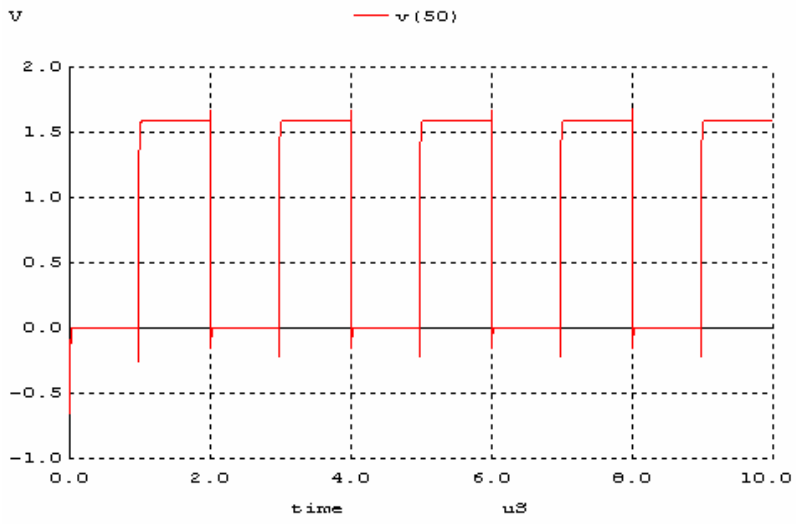


Figure 3.22. Simulation of DAC, V_{REF}= 2V

Example simulations of DAC circuit are shown in Figure 3.21 and Figure 3.22. Figure 3.21 shows maximum output voltage when the input is “111111” and reference voltage is 2.5 V. The second figure shows us when the input is “110011” and reference voltage is 2 V. Therefore the output must be equal to 51/63 of the reference voltage. That’s equal to 1.6 V. Latter simulation output value is almost same as the expected value.

The reader is referred to B. Yelten’s project report [17] for further detailed analysis of the DAC.

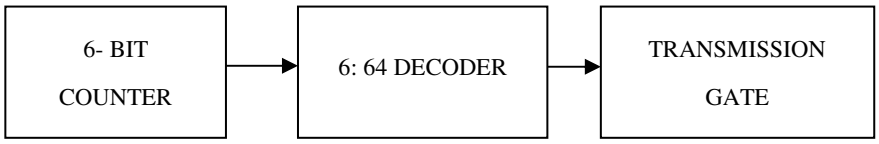


Figure 3.23. Control mechanism to apply reference voltages in serial manner

The block diagram of serial reference voltage control mechanism is shown in Figure 3.23. The block contains a 6 bit counter and a decoder. As the counter moves forward, the corresponding reference voltage produced by the DAC must be applied. This is the basic reason for the switches placed in front of the reference voltage. Whenever the counter is at a certain number the signal is passed through a decoder and the corresponding switch level is selected. At the gate of that switch, a positive pulse is given to activate the transmission gate in order to transmit the reference voltage to the gate of the differential pair reference input transistor.

As soon as the counter starts to operate the output of the counter is taken by the decoder. There are 64 transmission gates. Each of them can be activated if they are selected by the result of the counter. In the first cycle, counter shows "000000", thus the 0^{th} transmission gate must be activated. This means that the digitally produced 64 bit 'count' array should have logic '1' as its first element at that moment in order to let the transmission gate work. As soon as the transmission gate is on, the analog voltage output of the DAC is applied via the transmission gate to the differential pair. This process must be applied repeatedly for other numbers that the counter goes over. One of the biggest problems, that the series transmission has, is that the outputs of 64 threshold circuits arrive at different times since the comparison is conducted serially. This situation may cause intermediate result which could be undesirable. A remedy for that may be the increase in clock frequency. Nonetheless, this may have also its own consequences like the rise of the power consumption.

This procedure was applied via VHDL and the following figures from Modelsim were acquired. Figure 3.24 and 3.25 depict the operation methodology of the VHDL code. As can be seen, the counter goes on throughout the simulation beginning from zero. The variable 'result' in the simulation below is the output of the 6: 64 decoder. As the counter

points to one of the values in the range between 0 and 63, that transmission gate will become logic ‘1’ as can be seen from the result of the 6:64 decoder output array.

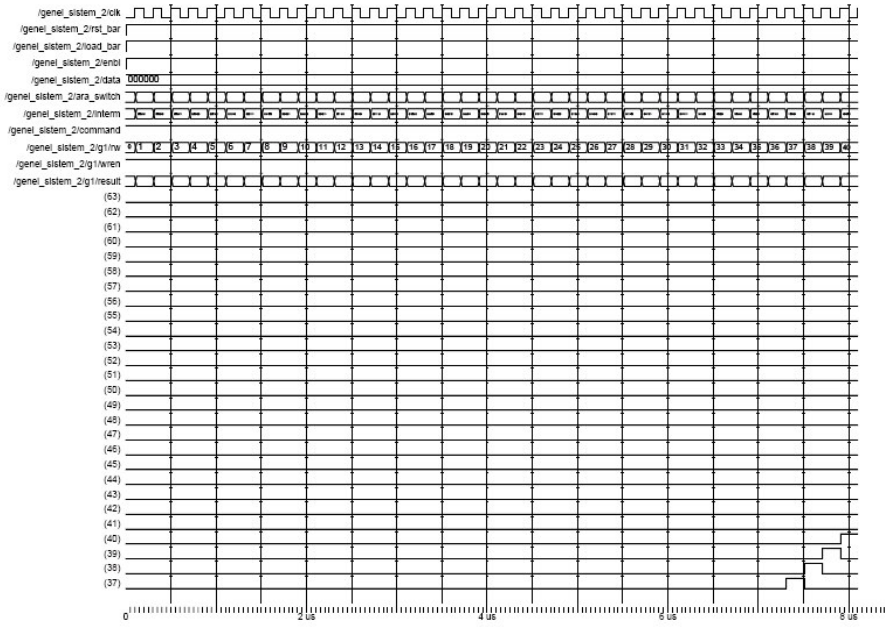


Figure 3.24. Simulation results for the serial reference control unit (I)

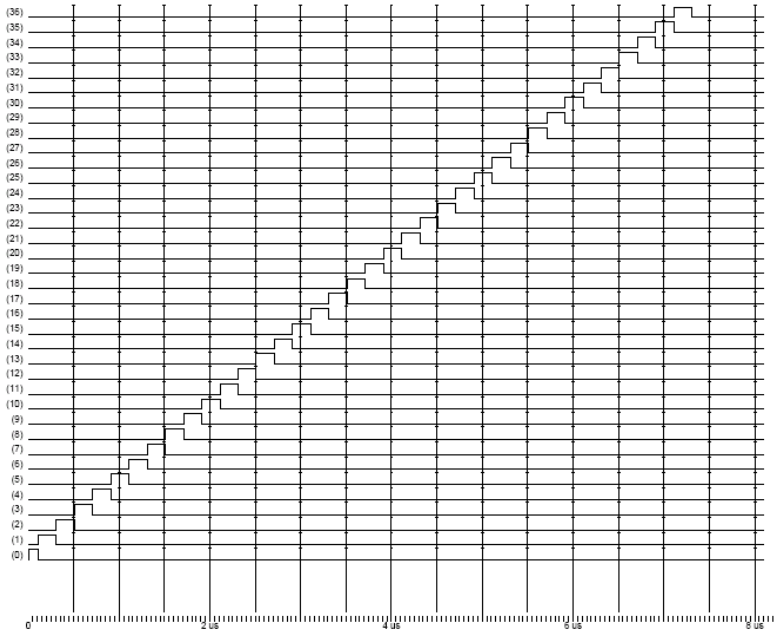


Figure 3.25. Simulation results for the serial reference control unit (II)

Figure 3.26 shows the basic schematic diagram for the synthesized VHDL code.

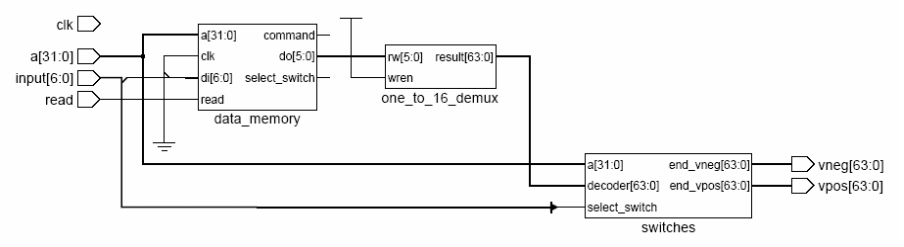


Figure 3.26. Synthesized VHDL code for the serial reference control unit

Since boundary neurons are known, the switches which must be closed are also known. Seeing this fact, alternatively a special counter which counts only the necessary switch numbers can be designed. Since the design has maximum eight possible active areas at the same time and each area has four connections to first layer, 8x4=32 memory elements, where each one is 6 bits, are needed. These memory elements hold the counter inputs. Please note that since capacitors hold the reference values until the end of cycle, closing order of switches is not important. Counter can only pass corresponding 6 bit memory element values to the DAC and the decoder. This idea was not pursued further in this work due to the increasing complexity of the digital circuit.

3.2.2. Parallel Reference Voltage Generation

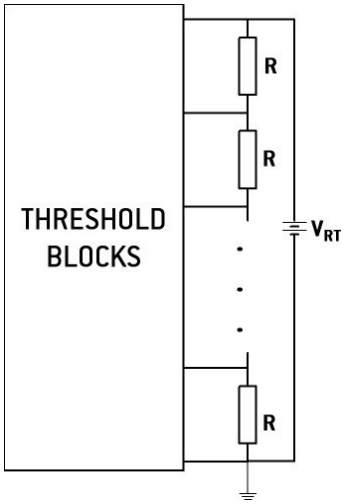


Figure 3.27. Block diagram of parallel reference voltage generation.

In parallel reference generation method, reference voltages are generated in parallel manner at the same instance of time via a stack of resistors. The general block diagram of

parallel reference voltage generation is shown in Figure 3.27. References are generated by a simple voltage divider, which divides V_{RT} voltage into 64 levels. Therefore i^{th} neuron in the transfer block is provided $i \cdot \frac{2.5}{64}$ V threshold voltage.

Matching is very important for parallel reference generation, since 64 resistors have to be equal. Resistor value is calculated by using AMS 0.35 micron technology mismatch and CMOS resistor model, (3.26) and (3.27) respectively. Power dissipation of resistor must be equal to 1 mW. Obtained resistor, width and length results for Poly 2 layer for different reference voltages are shown in Table 3.1. For 2.5 V reference voltage 24 Ohm is the optimal solution. This type of reference generation method is realized for the system, due to its simplicity.

$$\sigma\left(\frac{\Delta R}{R}\right) = \frac{A\_R}{\sqrt{W.L}} \tag{3.26}$$

$$R = \frac{L}{W} R_{\square} \tag{3.27}$$

Table 3.1. Possible resistor values according to the AMS mismatch model

RPOLY2			
V_{RT}(V)	$R(\Omega)$	$L(\mu m)$	$W(\mu m)$
2	15.62	17.171	54.95
2.1	17.22	18.03	52.33
2.3	20.66	19.74	47.78
2.5	24.41	21.46	43.96
2.7	28.47	23.18	40.70
2.9	32.85	24.90	37.89
3.1	37.53	26.61	35.45
3.3	42.53	28.33	33.30

3.3. Area Selection Block

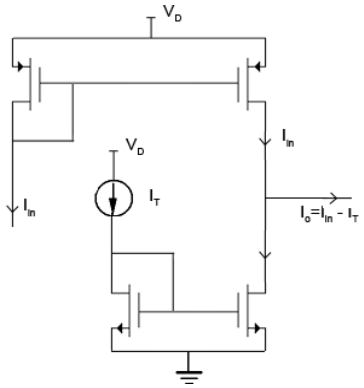


Figure 3.28. Current subtractor, basic area selection block

Area selection block corresponds to the second layer neurons, and it has connections to all NT blocks of two threshold blocks in the system; however by definition only four NT blocks output switches are set ON. This means only four NT blocks must obtain connections to an area selection block. Currents from these four connections are summed and taken as input to the block. The simplest area selection block form is a subtractor, where inputs are sum of block inputs and threshold current. The output of the block is simply the difference between the sum of inputs and the threshold. Since maximum output for NT block is $10\,\mu$A, there is $40\,\mu$A input current flows to the block if an area is fully selected. Therefore threshold current must be equal to $30\,\mu$A. This operation represents the t-norm examined in section 1.1.2. Figure 3.28 shows this configuration. Simulation is shown in Figure 3.30.

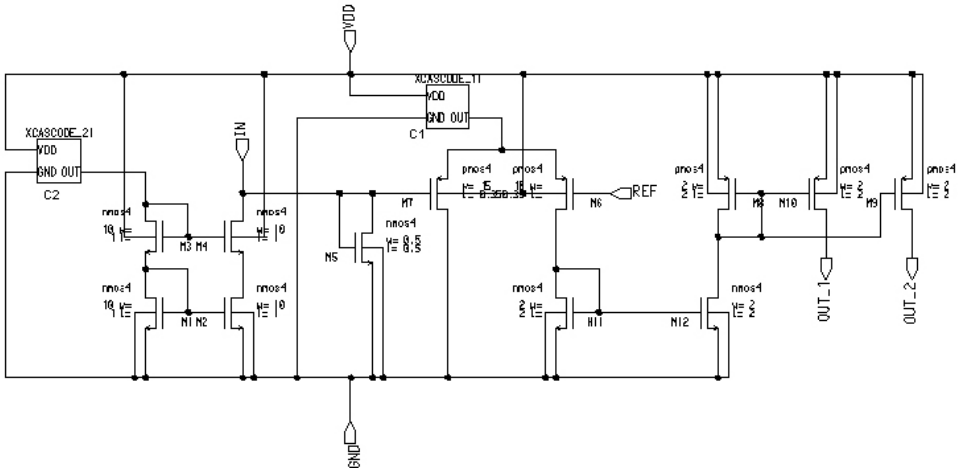


Figure 3.29. Area selection circuitry

However an activation function implementation where the user can alter the threshold value is more suitable for learning algorithms. Such a configuration is shown in Figure 3.29. It basically works as follows; the constant current source C2 operates as a threshold value which is equal to 30μA. Threshold current is subtracted from the input current and the result current generates voltage V_{M5} by help of M5 transistor. This generated voltage is one input of M6-M7 differential pair. The other input of differential pair is a reference voltage V_{ref} of which user can change for different applications. The drain current of M6, which is biased by C1 current source, is taken as output via current mirrors. This type of configuration is also based on differential pair characteristics. Therefore the computations, which are taken in section 3.1.1, are also valid for this configuration. As a consequence the characteristic of latter area selection block configuration can be derived as following:

For a rule ξ_i, where fuzzified inputs are $X_{fuzzified}$ and $Y_{fuzzified}$, respectively, following approximate relation for an area selection block output A_{ξ_i} can be written.

$$A_{\xi_i} = 5\mu\left(1+\tanh\left(10\left(V_{M5}-V_{ref}\right)\right)\right) \tag{3.28}$$

V_{M5} is actually gate to source voltage of M5 transistor. Its value depends on if M5 is in saturation or subthreshold region. If M5 is in saturation region

$$V_{M5} = V_{Th_{M5}} + \sqrt{\frac{2L_{M5}}{k_n' W_{M5}}} \cdot \sqrt{I_{X_{fuzzified}} + I_{Y_{fuzzified}} - I_{Th}} \tag{3.29}$$

If M5 is subthreshold region then

$$V_{M5} = V_{Th_{M5}} + V_T \cdot \ln(I_{M6}) \cdot \ln(I_{X_{fuzzified}} + I_{Y_{fuzzified}} - I_{Th}) \tag{3.30}$$

Simulation of block is shown in Figure 3.31. Figure shows fuzzified inputs, their sum which is the input of the area selection block, and output curves.

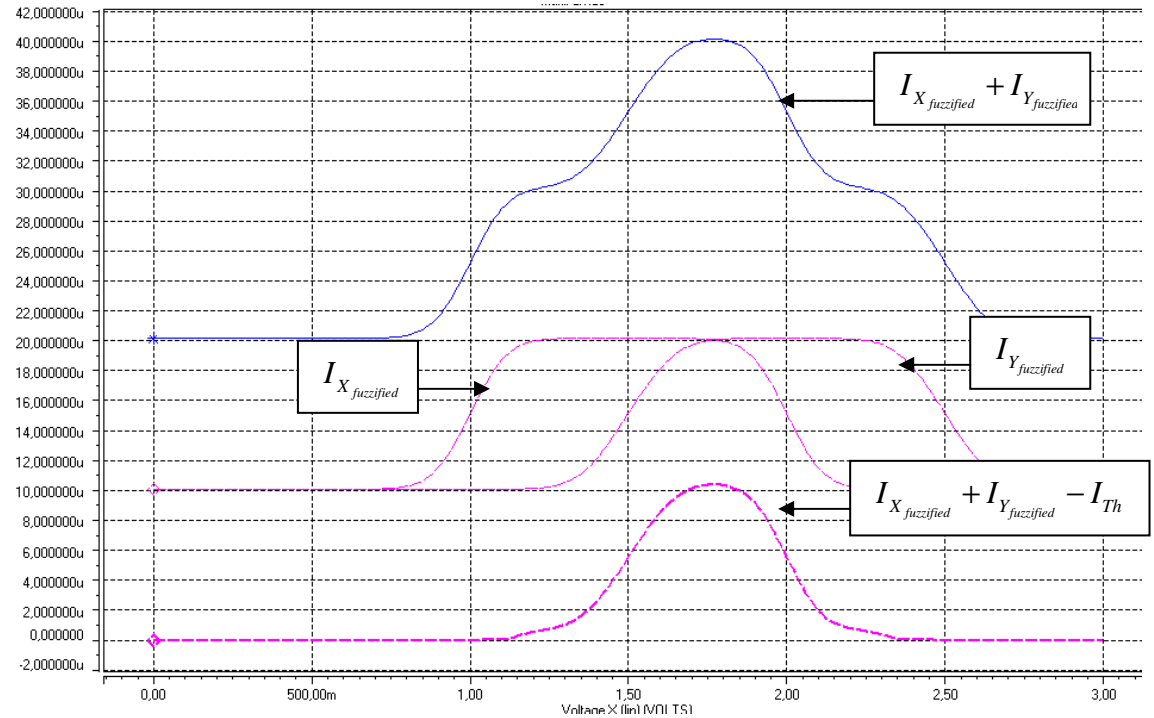


Figure 3.30. Simulation of basic area selection block

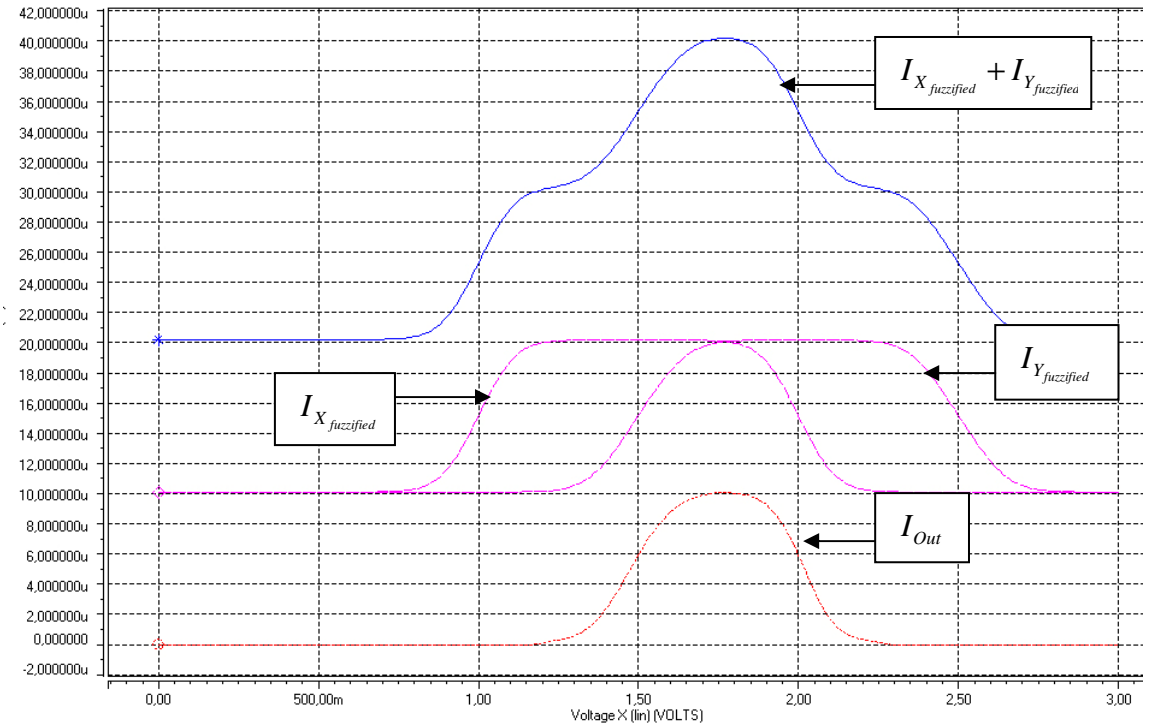


Figure 3.31. Simulation of second type area selection block

3.4. Normalization Block

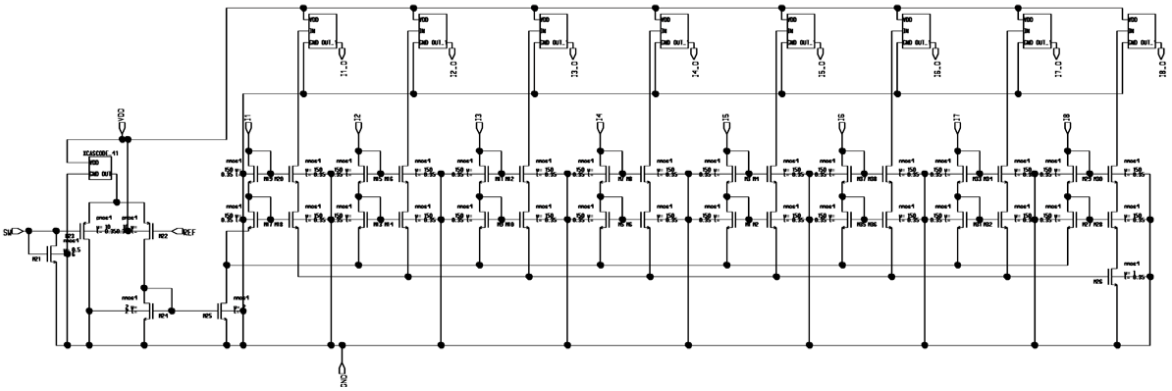


Figure 3.32. Normalization circuitry

Normalization operation is needed for correct defuzzification. A normalizer circuit for TSK type fuzzy system should perform the following computation, where *w* is weight value and *x* is the input:

$$OUT = \frac{\sum w_i x_i}{\sum w_i} \tag{3.31}$$

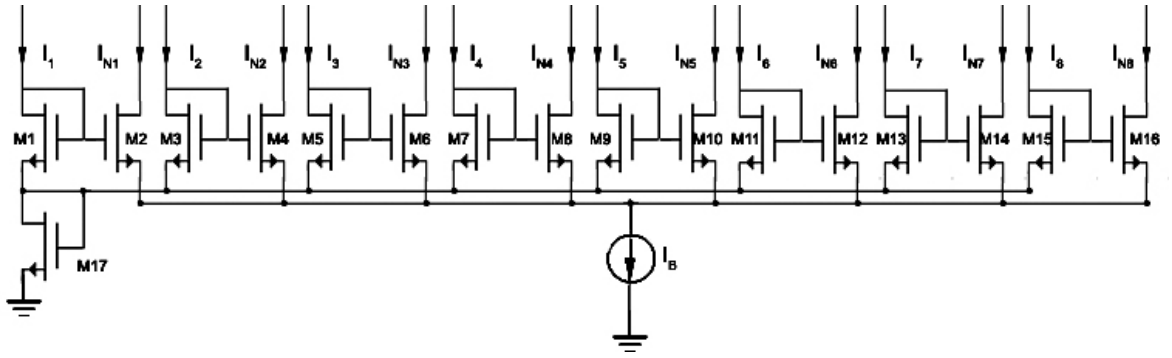


Figure 3.33. 8-output Gilbert normalizer

The required division is very difficult to implement in VLSI. Nevertheless there are some possible solutions to carry out this division. One is called 'Gilbert normalizer' [13], which is shown Figure 3.33. Designed normalization block, shown in Figure 3.32 is based on 8-output 'Gilbert normalizer'. For better accuracy purposes it is designed as cascoded.

Gilbert Normalizer circuit in Figure 3.33 has eight input currents I_j and provides eight output currents I_{Nj}. All transistors are matched and they are working in subthreshold region. For input drain currents we can write

$$I_1 = I_0 e^{\frac{KV_1}{V_T}} \quad I_2 = I_0 e^{\frac{KV_2}{V_T}} \quad \dots \quad I_8 = I_0 e^{\frac{KV_8}{V_T}} \quad . \tag{3.32}$$

and the output drain currents are

$$I_{N1} = I_0 e^{\frac{KV - V_1}{V_T}} \qquad I_{N2} = I_0 e^{\frac{KV - V_2}{V_T}} \quad \dots \quad I_{N8} = I_0 e^{\frac{KV - V_8}{V_T}} \tag{3.33}$$

Since the outputs are connected, we can write

$$\sum I_{Nj} = I_B = I_0 \sum \left(e^{\frac{KV_j}{V_T}} \right) e^{\frac{-V}{V_T}} = I_0 \sum \left(\frac{I_j}{I_0} \right) e^{\frac{-V}{V_T}} \tag{3.34}$$

We obtain

$$I_B = \sum \left(I_j \right) e^{\frac{-V}{V_T}} \Leftrightarrow e^{\frac{-V}{V_T}} = \frac{I_B}{\sum I_j} \tag{3.35}$$

Now we should try to write output current I_{Nj} (j=1,2,..,8) equations depending on the input currents I_j (j=1,2,..,8).

$$I_{N1} = I_B \frac{I_{X1}}{\sum I_{Xj}} \quad I_{N2} = I_B \frac{I_{X2}}{\sum I_{Xj}} \dots \; I_{N8} = I_B \frac{I_{X8}}{\sum I_{Xj}} \tag{3.36}$$

If we want to write a generalized formula, we can write

$$I_{Ni} = I_B \frac{I_{Xi}}{\sum I_{Xj}} \tag{3.37}$$

As it can be seen on expression (3.37) that each element in the output array is equal to the corresponding element in the input array, divided by the sum of all inputs and scaled

by current I_B. I_B is selected as $31\,\mu$A . There fore all input currents are normalized to [0,31 μ A] interval.

Figure 3.34 shows the simulation results. Note that sum of normalized currents is always constant and that the same current ratios are preserved after normalization. The latter feature is very difficult to accomplish in other solutions of normalizer circuits.

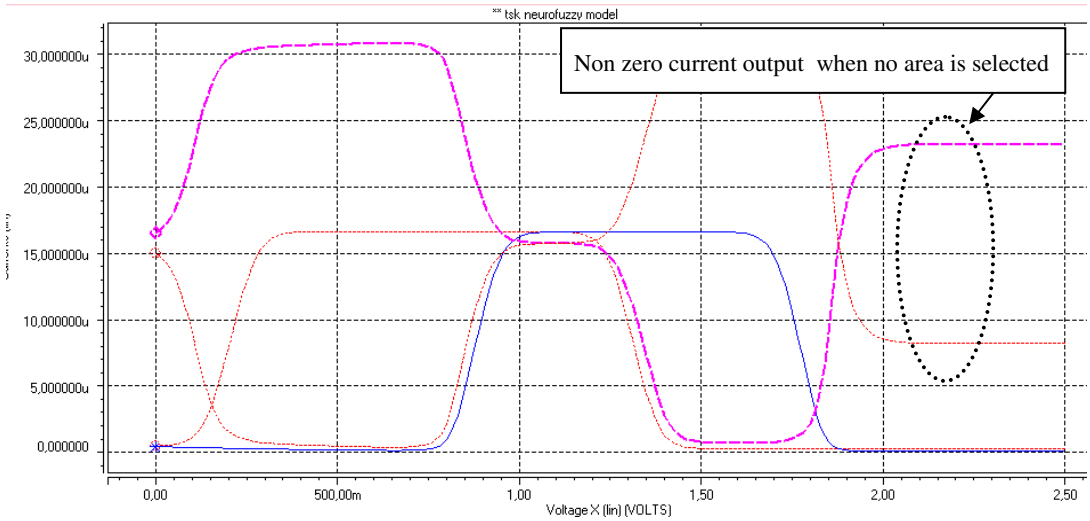


Figure 3.34. Problematic simulation result of the normalizer

Nonzero output while no area is selected is an undesirable situation. Since area selection blocks generate no output current while no area is selected, normalizer block inputs I_j are become 0 A. That causes the transistors, which is connected to input nodes, turn off and each output current becomes 1/8 of I_B current. Another issue is that normalizer tries to normalize every possible input current value. Area selection block generates some small currents that have order of magnitude in nano-amperes, even if no area is selected. In spite of this, normalized outputs are in micro ampere range. This situation is shown in Figure 3.34.

One possible solution to the problem is turning off the current source I_B while there is no input or there is a small input to the normalizer. For instance a current source which is controlled by the total output current of the area selection blocks is a simple solution.

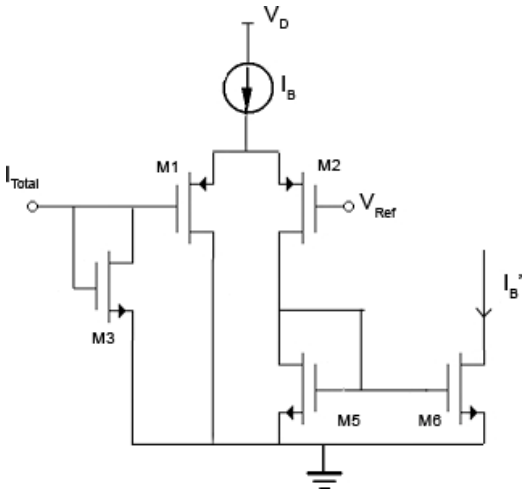


Figure 3.35. Current source controlling circuit

Implemented current source controller is shown in Figure 3.35. The differential pair principle applied in area selection block is also implemented for current source controlling, due to its simplicity and no need for high precision control. If one or more area is active, an output current from the area selection blocks is observed. Total output current of area selection blocks can be used for generating a voltage by the help of M3, additionally this voltage is compared with reference voltage via differential pair and the output current copied to the I_B node of normalizer. Since I_B should rise to its peak value as quickly as possible, differential pair transistors should work in subthreshold region. The widths of the differential pair transistors M1 and M2 are 40 μm and the lengths are 0.35 μm.

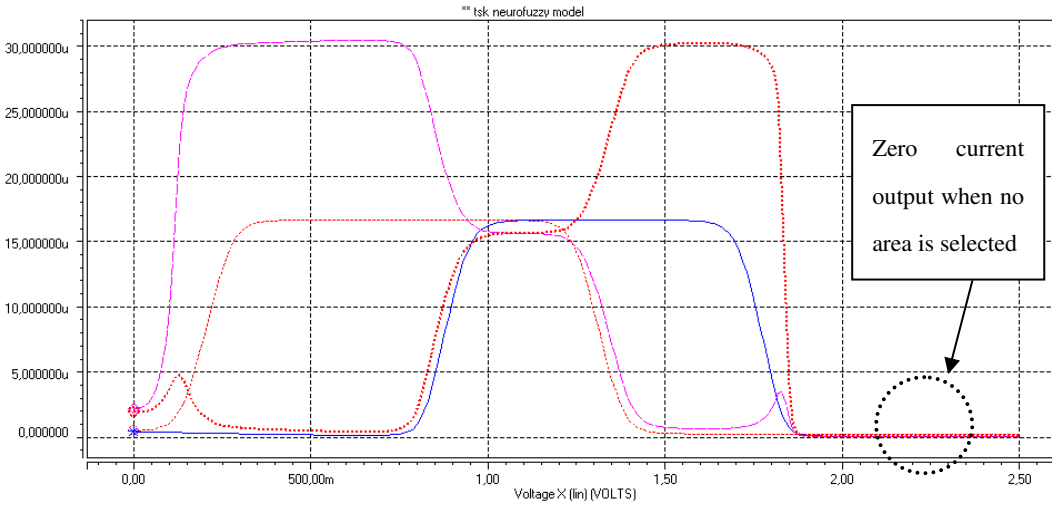


Figure 3.36. Simulation of normalizer with current control circuit

Simulation for new normalizer architecture with designed current source controller is shown in Figure 3.36. While there is no input current to normalizer, the output current is zero, which means no area is selected. This type of normalization block also supports resolving an undesired situation which occurs when some small values are normalized. It was commented that small values were normalized to a value between 0 and 31 μA. Even if the value normalization is correct, due to hyperbolic characteristic of membership functions normalization of very small values results a rotated version of rectangular pattern desired. This undesired situation is shown in Figure 3.37. It should be also mentioned that changing of activation function of threshold neurons may resolve the problem. However the reference value in the current source controller in Figure 3.35 could be used as a threshold value of which the sum of area selection block outputs must pass to trigger the normalizer. Since the reference value of current source is controllable user can change the resulting surface for the needs. Simulation of corrected version is shown in Figure 3.38. The figures show top views of a control surface of a singe rule fuzzy problem.

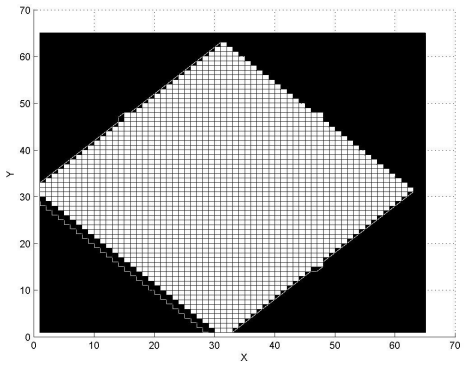


Figure 3.37. Top view of erroneous output surface, Rule(16,16,48,48)

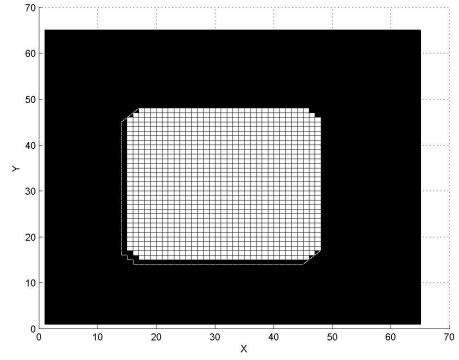


Figure 3.38. Top view of corrected output surface, Rule(16,16,48,48)

3.5. Weight Assigning Block

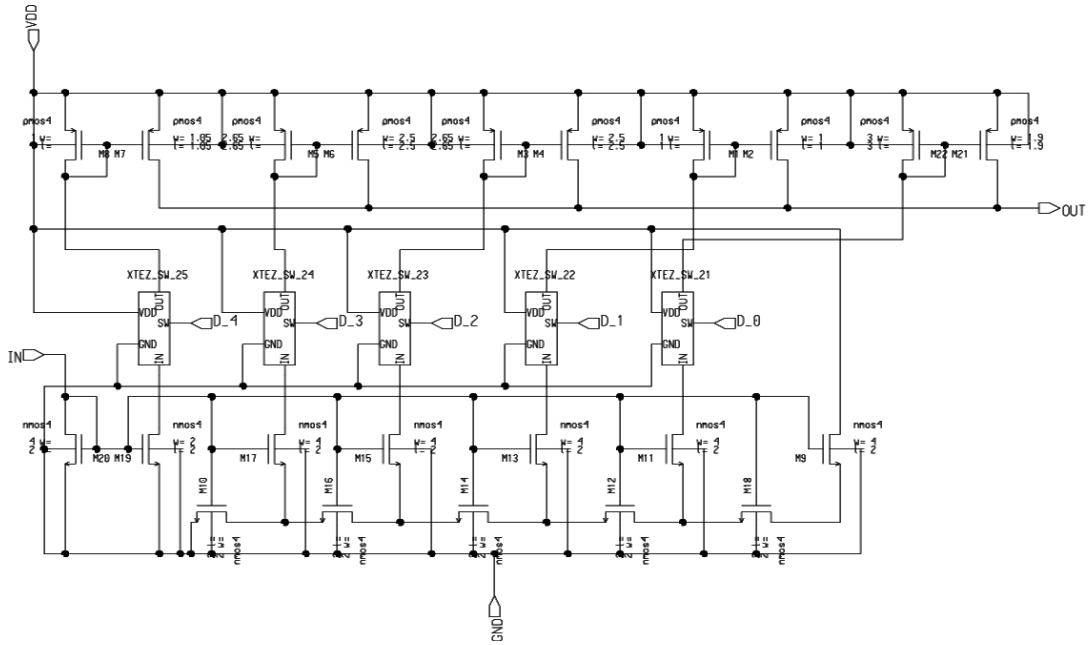


Figure 3.39. Weight assigning circuitry

Fuzzified values are normalized in normalization block. In order to obtain defuzzified output value, normalized values must be multiplied by corresponding weight values. Weight assigning block provides this multiplication.

Basically weight assigning block is a multiplier where one value could be determined by user. The main issue for the system is how to appropriately store the weights on chip in a non-volatile manner. If the weights are stored as charge on a capacitor [24][25], they will decay due to parasitic conductance. One method would use analog memory cell [20][21]. However this technique requires using large voltages. Another possible method is representing weights digitally [22][23]. Digital weight values are converted to analog values and used in analog multipliers. In digital weight approach the accuracy is a problem for certain learning algorithms. Therefore the size of digital word which represents weights should be selected carefully. Digital weights with current multiplying M2M DAC configuration showed in Figure 3.39 are employed. In this configuration weight values are determined by digital inputs of DAC. The reference or bias current is the normalized input current value that is taken from the normalization block.

3.5.1. Weight Representation

It was decided to represent the weights by 5 bit words. Therefore, the DAC which processes the weight data must be 5 bit. A 5 bit DAC has a resolution equals to 2^5 or 32, which means DAC is able to multiply the reference current by a maximum value of 1/31. This is the real reason why normalizer has a 31 μA bias current. Therefore user will be able to compute and see the exact result, corresponding to the fully selected area.

Let us have a rule R like following:

$$R: If\ X = A\ then\ Z = 5 \tag{3.38}$$

For the rule R, an output value 5 is wanted to be observed if R is fully selected. Remembering the rule output values are the weight values of the last layer of synapses, R area weight will be set to 5 or "00101". Please note that this setting procedure is done by a digital control. 5 also means fuzzified and normalized input, which can be maximum 31 μA while R area is fully selected, will be multiplied by 5/31;

$$31\mu \times \frac{5}{31} = 5\mu \tag{3.39}$$

5 μA corresponds the value which user set. User will see the exact result of R rule output value which implies the area is fully selected. Additionally user may want to set a value between 0 and 1. For such weight values user has 32 levels between 0 and 1, where 31 depicts 1 while 0 represents 0. In other words step size for such weight values is 1/31 = 0,03226. On the other hand user may want to set the weight value to a higher value than 31. So user must decide a way to represent this value between 0 and 31. One way for assigning a higher range values to a value between 0 and 31 is normalization. Please note that by using normalization you can also change the range to [0,1]. For example, if the maximum and minimum values of weights are known, following normalization transformation, which will change the range to [0,31], can be used.

$$\delta_j = 31\left(\frac{w_j - w^{\min}}{w^{\max} - w^{\min}}\right) \tag{3.40}$$

Additionally if all weights are known, the sum of weight values can also be used for obtaining normalized weights as shown in equation (3.41).

$$\delta_j = 31\left(\frac{w_j}{\sum w_i}\right) \tag{3.41}$$

3.5.2. 5 bit Current Multiplying DAC

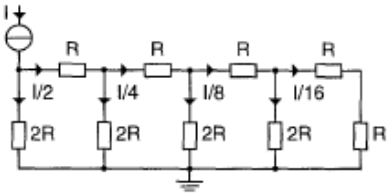


Figure 3.40. Basic R2R ladder

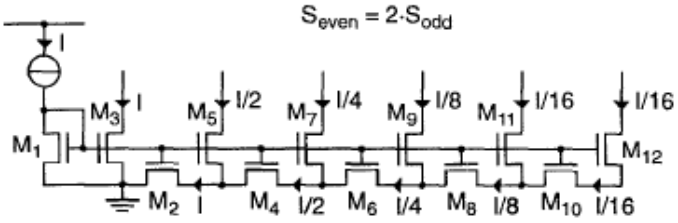


Figure 3.41. M2M ladder

Any network of linear resistors can be implemented by means of MOS transistors, with the additional possibility of electrically controlling value of each equivalent resistor if the transistor is operating in subthreshold region. This concept can be applied to the R2R ladder shown in Figure 3.40, resulting MOS only ladder (or in other words M2M ladder) [32] shown in Figure 3.41. In order to split the current equally at each node, the aspect ratios of the even numbered transistors need to be twice that of the odd numbered transistors, which can be simply realized either by connecting two unit transistors in parallel for the series branch and one for the shunt branch (or inversely one for the series

branch and two in series for the shunt branch). It is interesting to note that the current division is independent of the current and therefore it is also independent of the region where MOS transistors operate.

The performance of the circuit is limited by a number of second order effects. Mismatch of device geometry and oxide thickness will only affect the accuracy of the division.

The current output could be taken in various ways. In this implementation shown in Figure 3.39 current outputs are taken from drains of odd numbered transistors via current mirrors. Outputs are controlled by switches. MSB represented by M3 while LSB is M11 and M12. Simulation output is shown in Figure 3.42. As it can bee seen the precision of the circuit is very high. Every step form 0 to 31 μ A is achieved accurately.

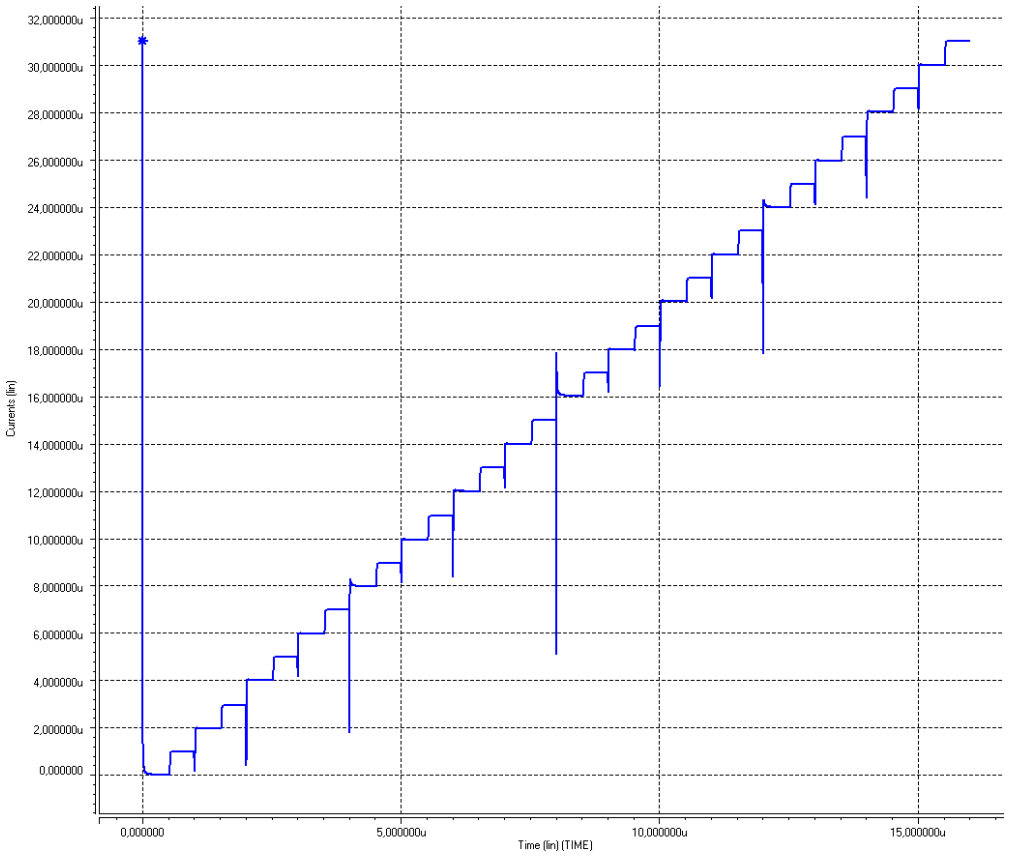


Figure 3.42. Simulation of DAC circuit, levels between 0 and 31 μ A

3.5.3. Weight Control Unit

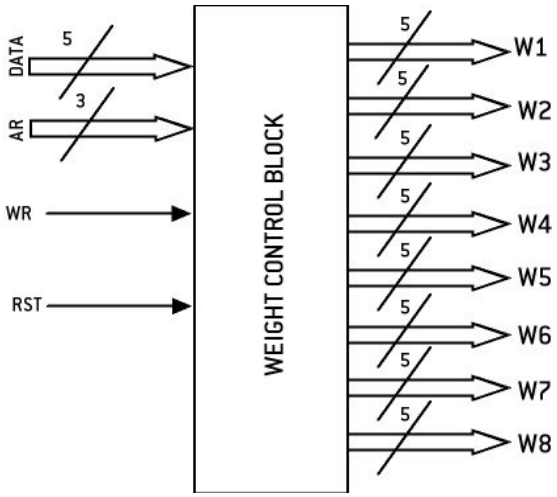


Figure 3.43. Block diagram of weight control unit

Since DAC inputs are indicating the weight values, these inputs must be assigned in some way. As in setting boundary neuron synapse connections, weight setting operation is done by a digital control block.

Designed weight controlling (or setting) block is shown in Figure 3.43. The block simply takes 5 bit input and writes it to memory and applies it to the switch inputs of corresponding weight assigning block.

Block has following inputs DATA, RST, WR and AR. DATA is 5 bit user input which represents the switch value of the weight assigning block. That means DATA must be equal to the value which user wants to set. Desired weight assigning block is assigned by AR input. Block converts binary AR data to a decimal value which denotes the index number of weight assigning block. DATA binding can only be done while WR input is '1'. RST input resets the switch values to '0'. In other words it resets the memory to '0' entirely.

Block outputs are named according to the order of weight assigning blocks. For example W1 represents the synapse between the first area selection block and the output node. Output configuration matches the configuration of the DAC in the weight assigning

blocks. For example MSB of W1 stands for MSB of the DAC input and LSB corresponds to the LSB input of the DAC.

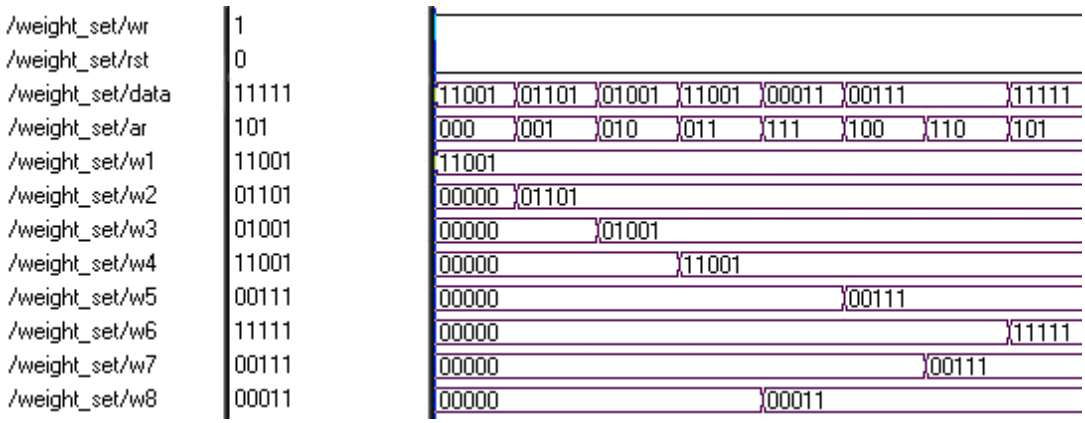


Figure 3.44. VHDL simulation of weight control unit

3.6. Summing Block

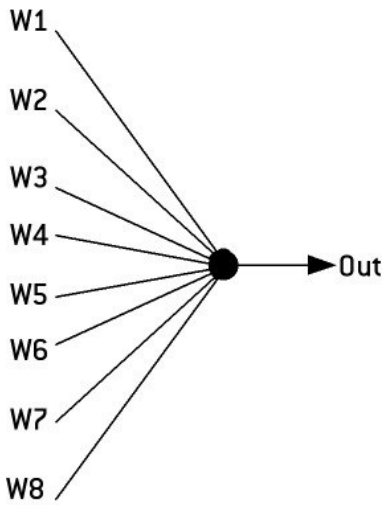


Figure 3.45. Summing node

It was noted that for correct defuzzification expression (3.28) is needed. Normalized and weighted currents are obtained in previous blocks of the design and the summing block completes the system and sums all normalized and weighted currents. This block is basically a node which all current outputs of weight assigning blocks, Wi i=1, 2, … , 8, are connected to. Figure 3.45 shows this simple configuration.

4. CHIP DESIGN & EXAMPLES

4.1. Chip Design

The complete circuit was designed and tested via various tools. A MATLAB model was also programmed for comparing the theoretical results with simulated ones. VHDL-AMS and H-SPICE simulations were carried out. Figure 4.3 shows the schematic of designed analog circuitry of the architecture. The digital circuitry is shown in appendix due to its size. Figure 4.4 shows the completed circuitry where analog and digital blocks are connected. While digital block generates the corresponding switch control voltages which is used in the process of the input data in analog circuitry. Due to limited chip area 16 neuron blocks were designed for each threshold block. Also eight area blocks, normalizer and eight weight assigning block are shown in Figure 4.3.

Total number of inputs is equal to 29, where two of them are the dimension inputs and the other ones are digital block inputs.

Complete circuit has an ability to process maximum eight million fuzzy rule evaluations per second. An example is shown in Figure 3.49, where X input is 1 MHz pulse signal and Y input is constant. Another example is shown in Figure 3.50, where this time X input is sinusoidal wave. Please note that while maximum input frequency is 1 MHz, maximum number of possible fuzzy rule evaluations per second is equal to eight million due to parallel evaluation of eight fuzzy rules.

Estimated power dissipation for analog circuitry is 37 mW, where power consumption of digital circuitry can be neglected. Total number of transistors is equal to 1862, while total number of digital gates is equal to 4018.

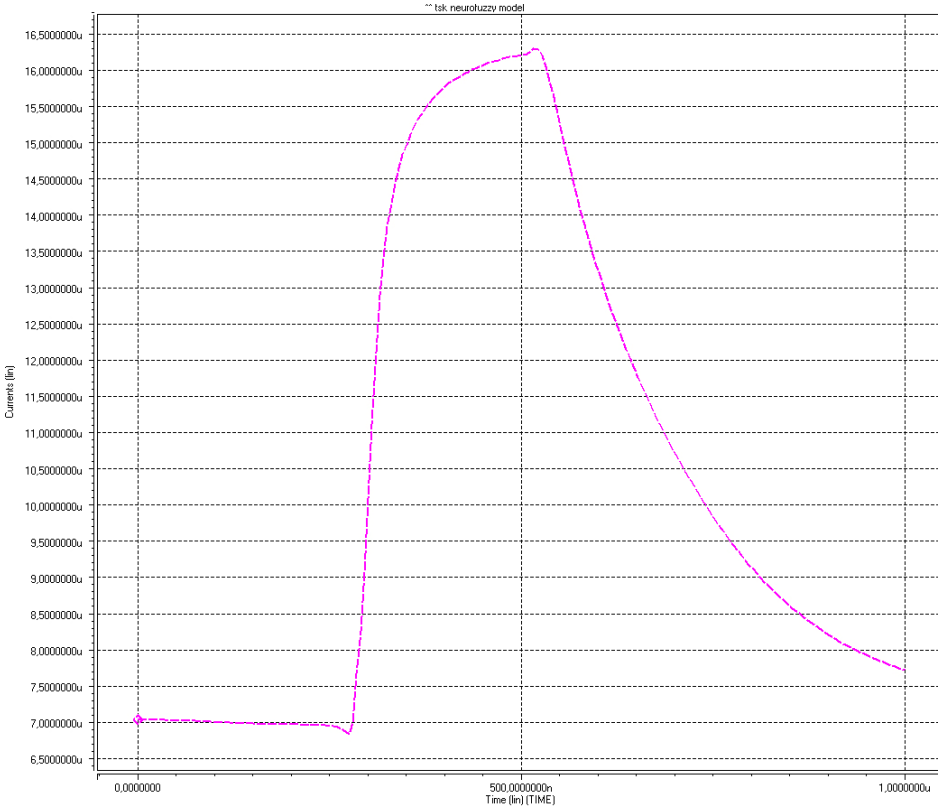


Figure 4.1. Maximum frequency, pulse input

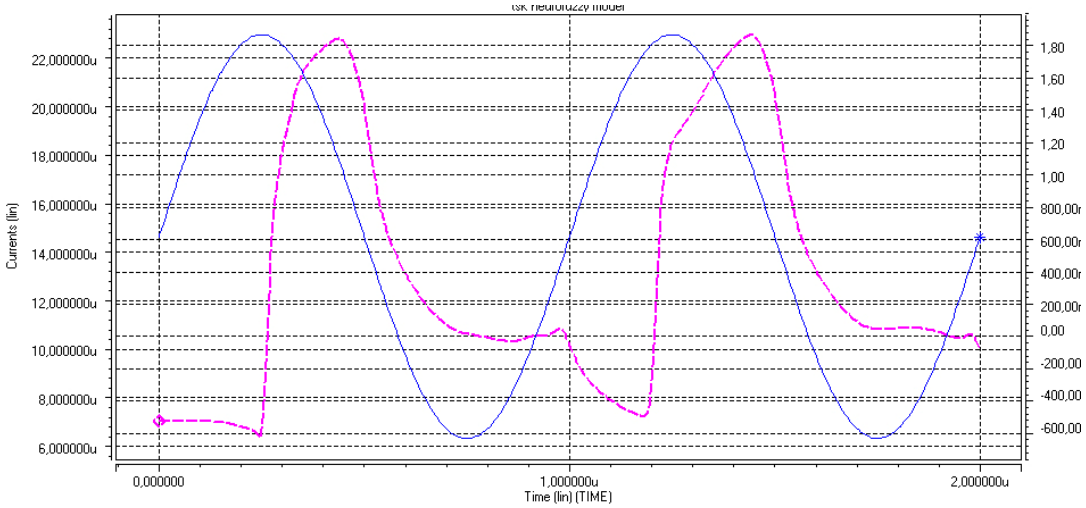


Figure 4.2. Maximum frequency, sinusoidal input

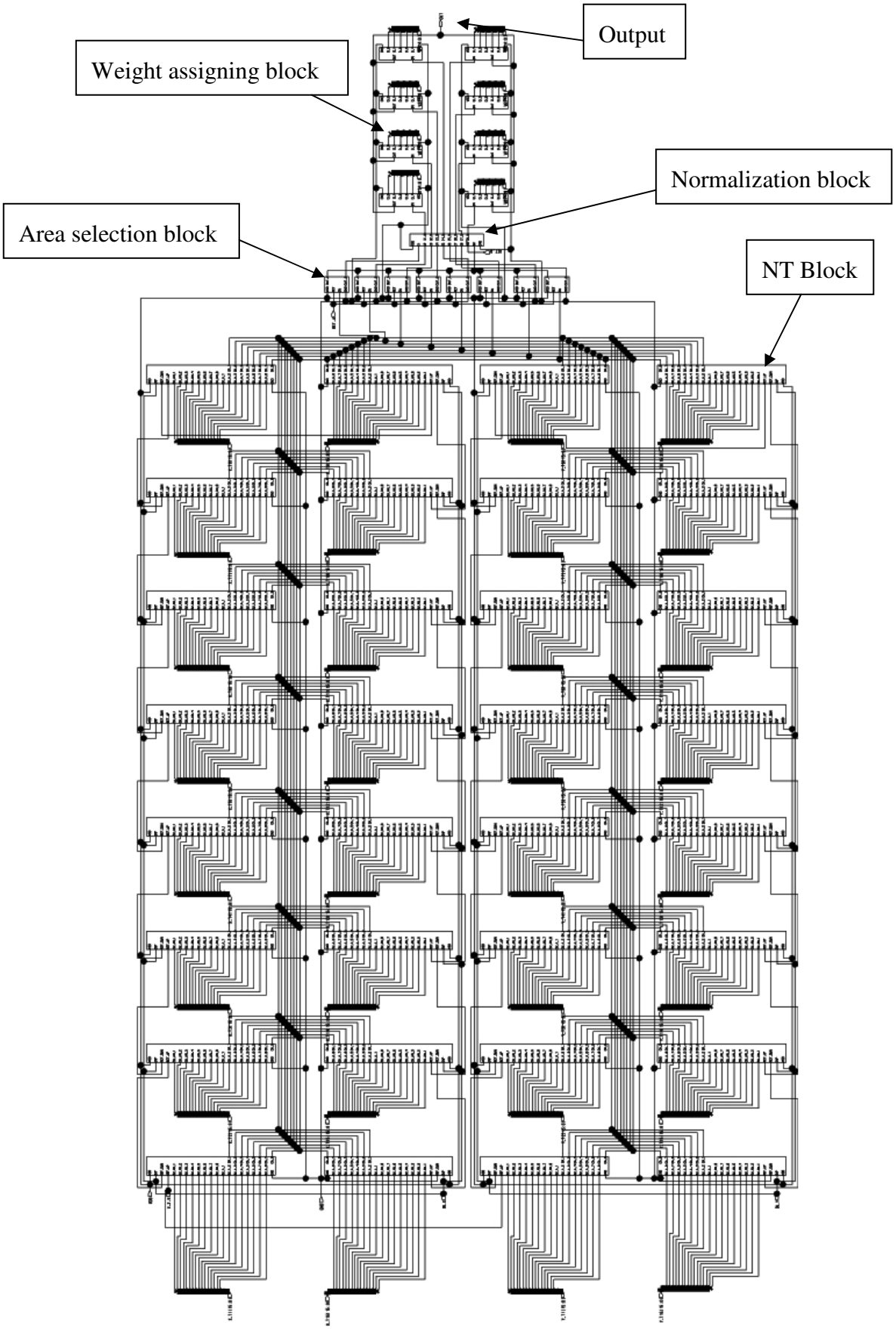


Figure 4.3. Complete analog circuit

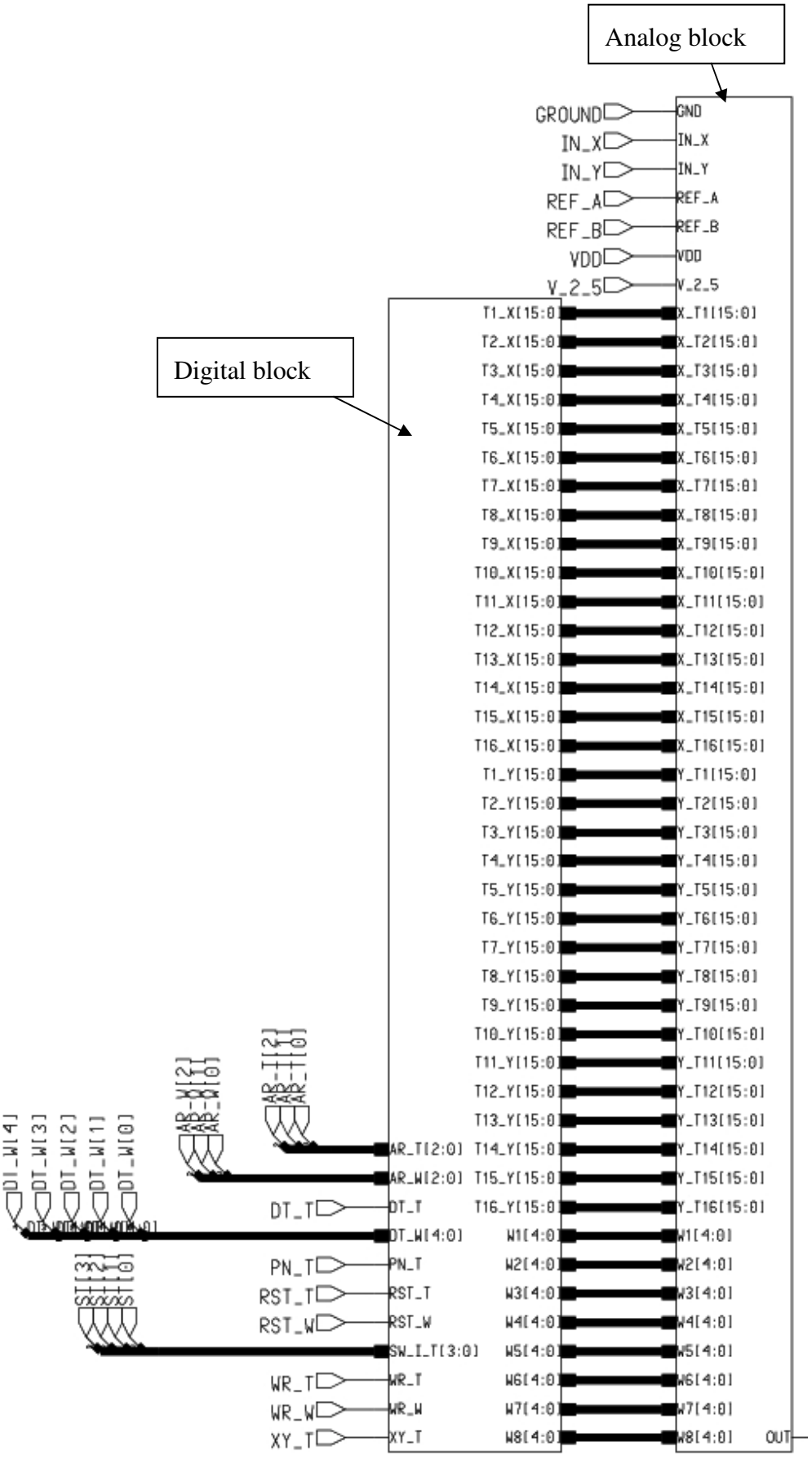


Figure 4.4. Analog and digital blocks connected

4.2. Examples

4.2.1. Implementation Of One Dimensional Rules

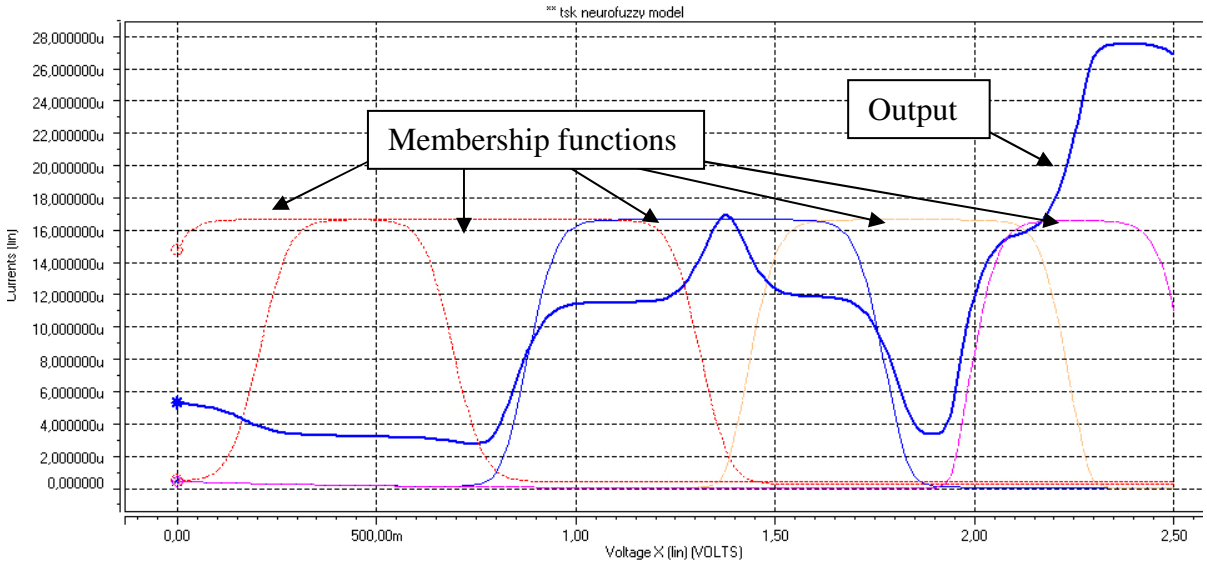


Figure 4.5. Simulation of example one dimensional fuzzy system

It is stated that design is implemented for two dimensional input spaces. However most of two dimensional fuzzy systems include some one dimensional fuzzy rules as shown below.

$$\text{If } X=A \text{ then } Z=c \tag{4.1}$$

Such one dimensional rules can also be implemented in proposed design. For the example rule, lower boundary neuron for Y dimension is set to the first neuron and upper boundary neuron is set to the last neuron (16 in implementation) of the system. Therefore, Y input will always be in the boundaries of the rule area.

A simulation example is shown in Figure 4.5. Figure 4.5 shows a DC simulation output of the chip where X input sweeps form 0V to 2.5 V and Y input is 1.25 V. System consists of five rules that are independent of Y dimension. Therefore, the output represents the control surface of the one dimensional fuzzy system.

4.2.2. Area Selection

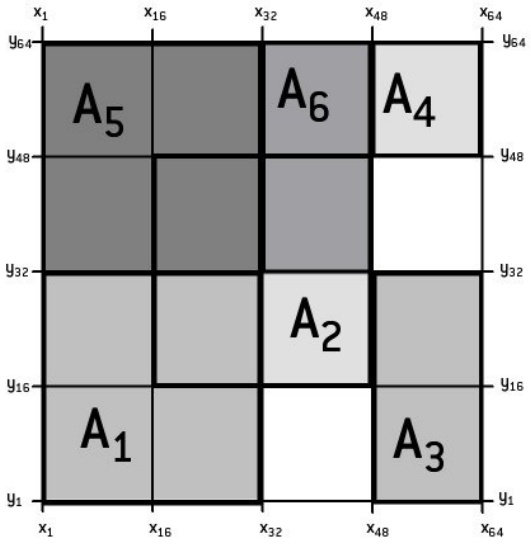


Figure 4.6. Example input space

A desired input space is shown in Figure 4.6. Desired areas and their corresponding boundary neurons are listed in Table 4.1:

Table 4.1. Example input space values

Area	NT_X		NT_Y		Output
A_1	1	32	1	32	10
A_2	16	48	16	48	21
A_3	48	64	1	32	6
A_4	48	64	48	64	31
A_5	1	32	32	64	7
A_6	32	48	32	64	26

Obtained control surface via theoretical MATLAB simulation is shown in Figure 4.7. Additionally obtained control surface via SPICE simulation is shown in Figure 4.8. T-norm operation is used in the theoretical model, while the second type of area selection block is used in the chip. The small differences in the edges of the simulated surface are based on second type of area selection block which does not perfectly utilize desired T-min operation.

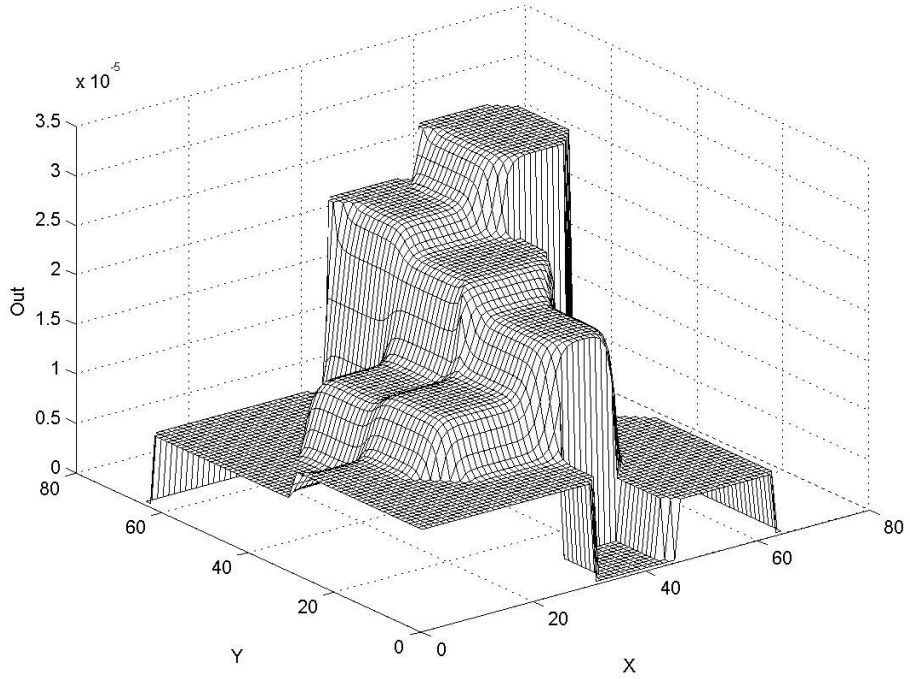


Figure 4.7. Theoretical control surface for the example

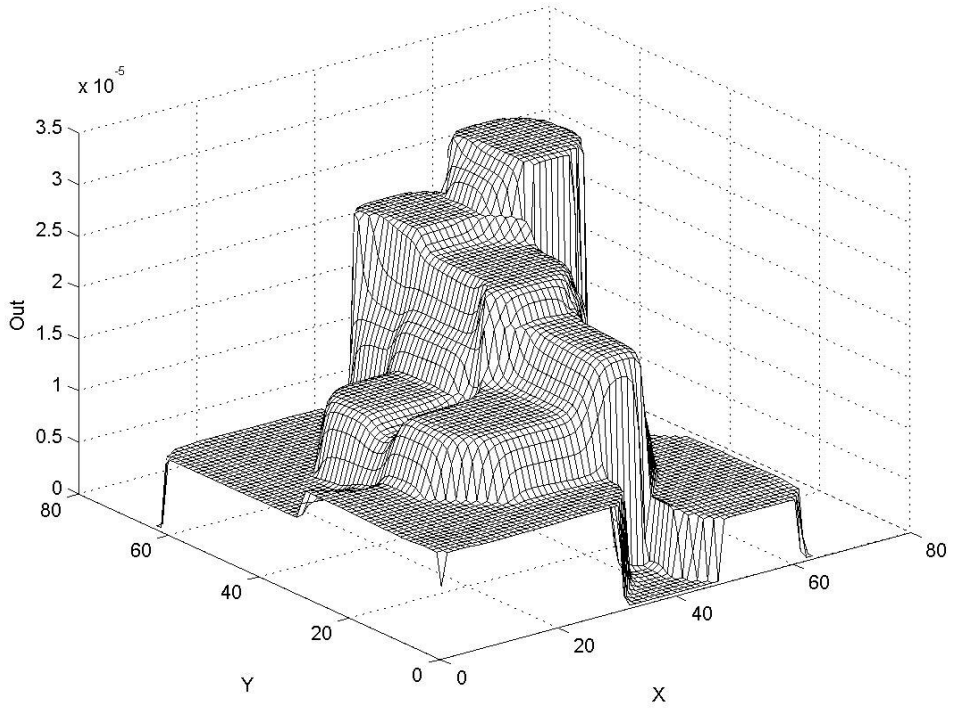


Figure 4.8. Simulated control surface for the example

4.2.3. Robotic Pool System

The decision of the difficulty of a shot on the pool table for a robotic pool system was chosen as another example control problem for the system. A theoretical example zero-order TSK type fuzzy control approach is carried out in [26]. Same approach will be followed and same fuzzy problem will be mapped onto the chip and the simulated SPICE results are compared with theoretical ones.

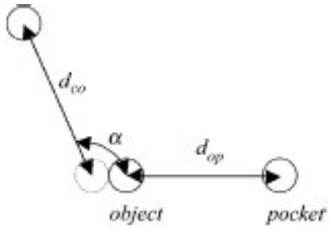


Figure 4.9. Geometry of a shot

In order to estimate the difficulty of a shot, there are three main parameters, which are shown in Figure 4.9, to be considered.

- The distance traveled by the cue ball before the collision with object ball, denoted as d_{co}.
- The distance from the object ball to pocket, denoted as d_{op}.
- The angle between the line joining both the distances d_{co} and d_{op}, denoted as α.

d_{co}, d_{op} and α is considered as inputs of the system in [26]. But since the chip has two inputs, the effect of d_{op} will be neglected in the system and d_{op} will be considered as an average distance value. Membership functions for d_{co} and d_{op} sets are shown in Figure 4.10. Membership functions for α are shown in Figure 4.11.

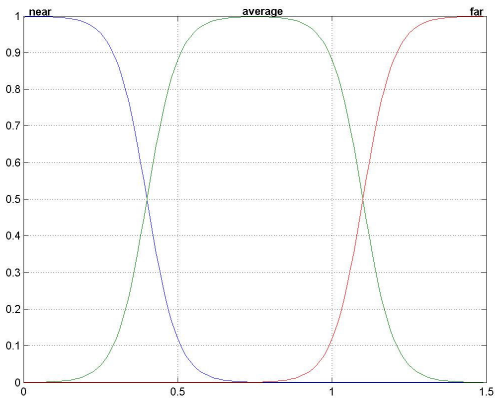


Figure 4.10. Membership functions of d_{co} and d_{op}

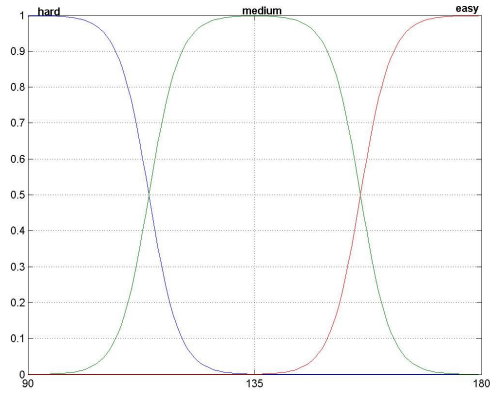


Figure 4.11. Membership functions of α

And the membership values of the output, which is denoted as Δ, are

$$\begin{aligned} \mu_{\Delta_{simple}}(0.0) &= 1.0 \\ \mu_{\Delta_{mediate}}(0.5) &= 1.0 \\ \mu_{\Delta_{high}}(1.0) &= 1.0 \end{aligned} \tag{4.2}$$

The rules are defined as

$$\mathrm{R_i}\text{: IF } d_{co} \text{ is LV } d_{co}^{(\mathrm{i})} \text{ AND } \alpha \text{ is LV } \alpha^{(\mathrm{i})} \text{ THEN } \Delta \text{ is LV } \Delta^{(\mathrm{i})} \tag{4.3}$$

where LV terms are linguistic values. The rule base is shown in Table 4.2.

Table 4.2. Rule base for the example

Rules	Inputs		Outputs
	d_{co}	α	Δ
1	Near	Easy	Intermediate
2	Near	Medium	Intermediate
3	Near	Hard	Tough
4	Average	Easy	Intermediate
5	Average	Medium	Intermediate
6	Average	Hard	Tough
7	Far	Easy	Intermediate
8	Far	Medium	Tough
9	Far	Hard	Tough

One can notice there are nine rules in the system. However the chip is restricted to eight rules for experimental purposes. One rule might be neglected or an extra area selection block can be added to the chip for just reviewing simulation results. Since adding extra components will change the architecture, the ninth rule will be neglected. Resulting theoretical control surface is shown in Figure 4.12. Simulated control surface is shown in Figure 4.13.

Theoretical response includes nine rules while simulated one consists of eight rule outputs. Absence of ninth area is clearly seen in Figure 4.13.

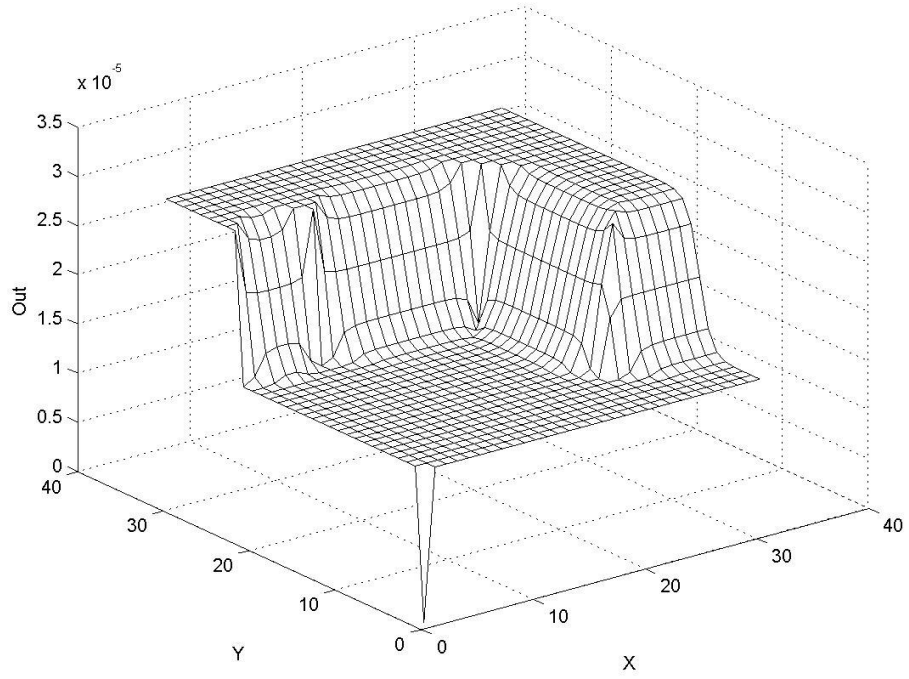


Figure 4.12. Theoretical control surface for the example

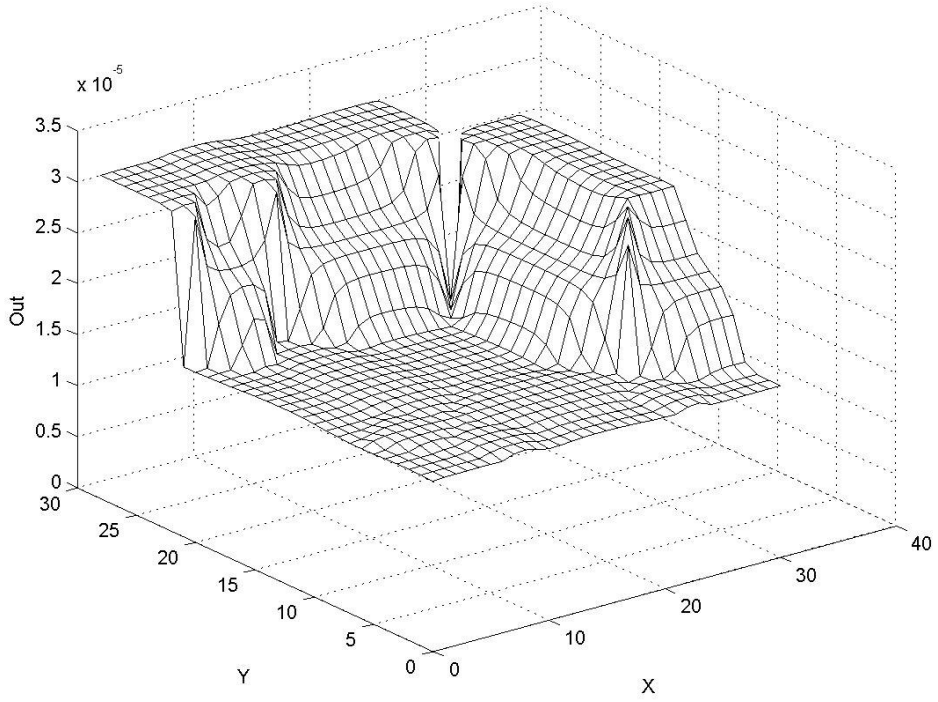


Figure 4.13. Simulated control surface for the example

5. CONCLUSION AND FUTURE WORK

5.1. Conclusion

A novel neuro-fuzzy architecture, which realizes zero order TSK type fuzzy system, is defined, tested, modeled and additionally analog CMOS implementation for two dimensional input spaces is realized. This implementation consists of the following blocks; threshold block, area selection block, normalization block, weight assigning, and summing block.

Threshold block is simply the first layer of the neuro-fuzzy architecture and includes neuron circuits. The analog implementation of this block mostly based on the differential pair characteristic. This characteristic is playing an important role in construction of membership functions. The saturation region is selected, but where it is similar to the subthreshold counterpart. If it is needed, another membership function is possible if sizes of transistors are selected suitably. Additionally neuron circuit may be replaced with another analog circuit if membership function is not obtainable via differential pair. Also serial and parallel reference voltage generation methods and their digital control blocks are presented, where parallel reference generation method is selected for implementation. Second layer connections are set by a digital control blocks which simply sets ‘ON’ or ‘OFF’ desired connection switches.

Area selection blocks represent rule neurons which are in the second layer of the system. The purpose of this block is to show if the rule is obtained. Two different possible implementations are introduced. First one is simply a current subtractor where a threshold current is subtracted from the sum of input currents. The other implementation is based on differential pair characteristic as threshold block. When inputs are inside the area of corresponding rule, then the block generates a current output which flows through the normalization block input nodes.

Normalization block normalizes each fuzzified input in order to obtain correct defuzzification operation. Additionally, some control problems, which result from the

incorrect operation of normalization circuit, are reviewed. A current source controller which generates reference current for normalization circuit is realized for correct operation. Controller is also based on differential pair characteristic. It simply sets a threshold value for normalization operation. If the voltage generated by total area selection output current is greater than this reference, then controller generates bias current for normalization in order to normalize the area selection outputs.

Weight assigning block represents the connections between second layer and third layer. This block can be seen as a multiplier where user can determine the multiplication value which stands for weight of a synapse. 5 bit current multiplying DAC is implemented and weights are decided to be represented digitally. The reference current of this DAC is normalized and fuzzified inputs. Therefore, outputs correspond to weighted inputs. For correct defuzzification additional summing operation is required.

Summing block sums all weighted inputs and finishes the fuzzy rule evaluation process. Since the values are represented by currents, the summing block is simply a node where all weight assigning block outputs are connected to.

Examples show us implemented chip works accurately and the control surface, which is constructed by simulations, are very close to the theoretically created control surfaces. Implemented chip is also able to evaluate eight million fuzzy rules per second. Estimated power dissipation is equal to 37 mW.

5.2. Future Work

5.2.1. Additional Circuit Implementations

Hyperbolic type activation function and membership function implementations are possible in proposed design. Additional neuron circuit implementations for the other type of activation functions should also be determined.

Normalization block is the bottleneck of the system. Due to operating in subthreshold region, it causes a major drop in the maximum possible bandwidth value. New

normalization block realization should be tested and additionally new normalization block strategies should be introduced.

5.2.2. Learning Algorithm

Realization of a learning algorithm has not been achieved. This thesis actually focuses on the hardware implementation, however some attempts to realize a learning procedure was made. Nevertheless some key points, which may help realizing a learning procedure, are detected:

- There are four synapse connections from first layer to a second layer neuron. Two of them are (+1) while the other two are (-1) connections. That means, if there are n second layer neurons, there must be 4n number of connections between first and second layer, where 2n of them are (+1) connections and the other 2n are (-1) connections.
- If there is a (+1) connection in the first layer there must be at least another connection which has (-1) value for corresponding dimension. Additionally there must be at least two connections for other dimension, (+1) and (-1), respectively.

The most characteristic feature of the architecture is that there are equal number of (+1) an (-1) weight valued synapses between first and second layer. Due to not all of the first layer neurons have connections to second layer, learning algorithm must also consider 0 value, which represents no connection.

APPENDIX A: SPICE, VHDL AND MATLAB CODES

A.1. Spice Codes

A.1.1. NT Block

```
.subckt thresh in ref1 ref2 oA1 oA2 oA3 oA4 oA5 oA6 oA7 oA8 oA1n oA2n oA3n oA4n
oA5n oA6n oA7n oA8n sA1 sA2 sA3 sA4 sA5 sA6 sA7 sA8 sA1n sA2n sA3n sA4n sA5n
sA6n sA7n sA8n
vdd dd 0 3.3
x1 s cascode7
m1 s in o1 dd pfet W= 15u L=0.35u
m2 s  ref1 o2 dd pfet W= 15u L=0.35u
mncm1 o1 o1 0 0 nfet w=1u l=1u
mncm2 o1b o1 0 0 nfet w=1u l=1u
mnpm1 o1b o1b dd dd pfet w=1u l=1u
mnpm_A1 oA1b o1b dd dd pfet w=1u l=1u
mnpm_A2 oA2b o1b dd dd pfet w=1u l=1u
mnpm_A3 oA3b o1b dd dd pfet w=1u l=1u
mnpm_A4 oA4b o1b dd dd pfet w=1u l=1u
mnpm_A5 oA5b o1b dd dd pfet w=1u l=1u
mnpm_A6 oA6b o1b dd dd pfet w=1u l=1u
mnpm_A7 oA7b o1b dd dd pfet w=1u l=1u
mnpm_A8 oA8b o1b dd dd pfet w=1u l=1u
mncm3 o2 o2 0 0 nfet w=1u l=1u
mncm4 o2b o2 0 0 nfet w=1u l=1u
mnpm2 o2b o2b dd dd pfet w=1u l=1u
mnpm_A1n oA1bn o2b dd dd pfet w=1u l=1u
mnpm_A2n oA2bn o2b dd dd pfet w=1u l=1u
mnpm_A3n oA3bn o2b dd dd pfet w=1u l=1u
mnpm_A4n oA4bn o2b dd dd pfet w=1u l=1u
mnpm_A5n oA5bn o2b dd dd pfet w=1u l=1u
```

```
mnpm_A6n oA6bn o2b dd dd pfet w=1u l=1u
mnpm_A7n oA7bn o2b dd dd pfet w=1u l=1u
mnpm_A8n oA8bn o2b dd dd pfet w=1u l=1u
xsA1 oA1 oA1b sA1 dd switch2
xsA2 oA2 oA2b sA2 dd switch2
xsA3 oA3 oA3b sA3 dd switch2
xsA4 oA4 oA4b sA4 dd switch2
xsA5 oA5 oA5b sA5 dd switch2
xsA6 oA6 oA6b sA6 dd switch2
xsA7 oA7 oA7b sA7 dd switch2
xsA8 oA8 oA8b sA8 dd switch2
xsA1n oA1n oA1bn sA1n dd switch2
xsA2n oA2n oA2bn sA2n dd switch2
xsA3n oA3n oA3bn sA3n dd switch2
xsA4n oA4n oA4bn sA4n dd switch2
xsA5n oA5n oA5bn sA5n dd switch2
xsA6n oA6n oA6bn sA6n dd switch2
xsA7n oA7n oA7bn sA7n dd switch2
xsA8n oA8n oA8bn sA8n dd switch2
R ref1 ref2 20
.subckt switch2 in out sw dd
X1 sw swn dd inverter2
ms1a in swn out dd pfet w=5.25u l=0.35u
ms1b in sw out 0 nfet w=1.75u l=0.35u
.ends
.subckt inverter2 inn outn dd
m1 dd inn outn dd pfet w=3.15u l=0.35u
m2 outn inn 0 0 nfet w=1.05u l=0.35u
.ends
.subckt cascode7 5
vdd 1 0 3.3
m1 2 2 1 1 pfet W=10u L=1u
m3 5 2 1 1 pfet W=10u L=1u
```

```
m4 20 20 0 0 nfet w=1u l=29.5u
m5 30 20 0 0 nfet w=1u l=29.5u
Rref 1 50 10k
Vin 50 20 0
Vo 2 30 0
.ends cascode7
.ends  thresh
```

A.1.2. Area Selection Block

```
.subckt areasel 30 40 10 34
xc2 32 cascode2
xc 16 cascode1
mt1 32 32 35 0 nfet w=10u l=1u
mt2 30 32 36 0 nfet w=10u l=1u
mt3 35 35 0 0 nfet w=10u l=1u
mt4 36 35 0 0 nfet w=10u l=1u
mo1 16 34 0 10 pfet W=15u L=0.35u
mo2 16 20 40 10 pfet W=15u L=0.35u
mres1 34 34 0 0 nfet W=0.5u L=0.5u
vref 20 0 0.625
.subckt cascode1 5
vdd 1 0 3.3
m1 2 2 1 1 pfet W=10u L=1u
m3 5 2 1 1 pfet W=10u L=1u
m4 20 20 0 0 nfet w=1u l=21u
m5 30 20 0 0 nfet w=1u l=21u
Rref 1 50 10k
Vin 50 20 0
Vo 2 30 0
.ends cascode1

.subckt cascode2 5
```

```
vdd 1 0 3.3
m1 2 2 1 1 pfet W=10u L=1u
m3 5 2 1 1 pfet W=10u L=1u
m4 20 20 0 0 nfet w=5u l=42u
m5 30 20 0 0 nfet w=5u l=42u
Rref 1 50 10k
Vin 50 20 0
Vo 2 30 0
.ends cascode2
.ends
```

A.1.3. Normalization Block

```
.subckt normalizer i1 i2 i3 i4 i5 i6 i7 i8 i1o i2o i3o i4o i5o i6o i7o i8o isw
vdd dd 0 3.3
m11 i1 i1 i1a 0 nfet w=150.0u l=0.35u
m12 i1o i1 i1b 0 nfet w=150.0u l=0.35u
m13 i1a i1a 12 0 nfet w=150.0u l=0.35u
m14 i1b i1a 442 0 nfet w=150.0u l=0.35u
m21 i2 i2 i2a 0 nfet w=150.0u l=0.35u
m22 i2o i2 i2b 0 nfet w=150.0u l=0.35u
m23 i2a i2a 12 0 nfet w=150.0u l=0.35u
m24 i2b i2a 442 0 nfet w=150.0u l=0.35u
m31 i3 i3 i3a 0 nfet w=150.0u l=0.35u
m32 i3o i3 i3b 0 nfet w=150.0u l=0.35u
m33 i3a i3a 12 0 nfet w=150.0u l=0.35u
m34 i3b i3a 442 0 nfet w=150.0u l=0.35u
m41 i4 i4 i4a 0 nfet w=150.0u l=0.35u
m42 i4o i4 i4b 0 nfet w=150.0u l=0.35u
m43 i4a i4a 12 0 nfet w=150.0u l=0.35u
m44 i4b i4a 442 0 nfet w=150.0u l=0.35u
m51 i5 i5 i5a 0 nfet w=150.0u l=0.35u
m52 i5o i5 i5b 0 nfet w=150.0u l=0.35u
```

```
m53 i5a i5a 12 0 nfet w=150.0u l=0.35u
m54 i5b i5a 442 0 nfet w=150.0u l=0.35u
m61 i6 i6 i6a 0 nfet w=150.0u l=0.35u
m62 i6o i6 i6b 0 nfet w=150.0u l=0.35u
m63 i6a i6a 12 0 nfet w=150.0u l=0.35u
m64 i6b i6a 442 0 nfet w=150.0u l=0.35u
m71 i7 i7 i7a 0 nfet w=150.0u l=0.35u
m72 i7o i7 i7b 0 nfet w=150.0u l=0.35u
m73 i7a i7a 12 0 nfet w=150.0u l=0.35u
m74 i7b i7a 442 0 nfet w=150.0u l=0.35u
m81 i8 i8 i8a 0 nfet w=150.0u l=0.35u
m82 i8o i8 i8b 0 nfet w=150.0u l=0.35u
m83 i8a i8a 12 0 nfet w=150.0u l=0.35u
m84 i8b i8a 442 0 nfet w=150.0u l=0.35u
mr 12 12 0 0 nfet w=1u l=0.35u
xcas 443 cascode3
mr1 0 isw 443 dd pfet w=10.0u l=0.35u
mr2 444 ir2 443 dd pfet w=10.0u l=0.35u
mhcm11 444 444 0 0 nfet w=2u l=2u
mhcm21 442 444 0 0 nfet w=2u l=2u
VR2 ir2 0 1.1
ms isw isw 0 0 nfet w=0.5u l=6u
.subckt cascode3 5
vdd 1 0 3.3
m1 2 2 1 1 pfet W=40u L=1u
m3 5 2 1 1 pfet W=40u L=1u
m4 20 20 0 0 nfet w=5u l=57.75u
m5 30 20 0 0 nfet w=5u l=57.75u
Rref 1 50 10k
Vin 50 20 0
Vo 2 30 0
.ends cascode3.ends
```

A.1.4. Weight Assigning Block

```
.subckt dac ss in1 in2 in3 in4 in5 m_out dd
**********CURRENT DIVIDING**********
m1 ss ss 0 0 nfet w=4u l=2u
m2 d1 ss out 0 nfet w=2u l=2u
m3 10 ss out 0 nfet w=4u l=2u
m4 d2 ss 10 0 nfet w=2u l=2u
m5 20 ss 10 0 nfet w=4u l=2u
m6 d3 ss 20 0 nfet w=2u l=2u
m7 30 ss 20 0 nfet w=4u l=2u
m8 d4  ss 30 0 nfet w=2u l=2u
m9 40 ss 30 0 nfet w=4u l=2u
m10 d5 ss 40 0 nfet w=2u l=2u
m11 40 ss 30 0 nfet w=4u l=2u
m12 dd ss 40 0 nfet w=2u l=2u
*******SWITCHES FOR DIGITAL INPUT***************
Xs1 o1 d1 in1 dd switch
Xs2 o2 d2 in2 dd switch
Xs3 o3 d3 in3 dd switch
Xs4 o4 d4 in4 dd switch
Xs5 o5 d5 in5 dd switch
****************************
Vt 55 50 0
**********CURRENT MIRRORS**********************
mcm11 dd o1 o1 dd pfet w=1u l=1u
mcm12 dd o1 m_out dd pfet w=1.05u l=1.05u
mcm21 dd o2 o2 dd pfet w=2.65u l=2.65u
mcm22 dd o2 m_out dd pfet w=2.5u l=2.5u
mcm31 dd o3 o3 dd pfet w=2.65u l=2.65u
mcm32 dd o3 m_out dd pfet w=2.5u l=2.5u
mcm41 dd o4 o4 dd pfet w=1u l=1u
mcm42 dd o4 m_out dd pfet w=1u l=1u
```

```
mcm51 dd o5 o5 dd pfet w=3u l=3u
mcm52 dd o5 m_out dd pfet w=1.9u l=1.9u
**********SWITCH SUB**********
.subckt switch in out sw dd
ms1a dd sw out dd pfet w=5.25u l=0.35u
ms1b in sw out 0 nfet w=1.75u l=0.35u
.ends
*****INVERTER SUB******
.subckt inverter inn outn dd
m1 dd inn outn dd pfet w=3.15u l=0.35u
m2 outn inn 0 0 nfet w=1.05u l=0.35u
.ends
**************************
vo out 0 0
.ends
```

A.1.5. Complete Analog Circuit

```
** TSK NeuroFuzzy Model
** by Baykal Sarioglu
vdd 100000 0 3.3
vtezt 100000 10 0
vref 20 0 1
********** REFERENCE VOLTAGES **********
vrxn ref_x1 0 2.5
vryn ref_y1 0 2.5
********** INPUT VOLTAGES **********
vinp1 in 0  sin(.618 1.25 1000k)
vinp2 in2 0 2.0
********** THRESHOLD BLOCK INSTANCES **********
** x axis
xth_x1 in ref_x1 ref_x2         a1 a2 a3 a4 a5 a6 a7 a8     a1n a2n a3n a4n a5n a6n a7n a8n
0 0 10 10 0 0 0 0    0 0 0 0 0 0 0 0  thresh
```

xth_x2 in ref_x2 ref_x3 a1 a2 a3 a4 a5 a6 a7 a8 a1n a2n a3n a4n a5n a6n a7n a8n
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 thresh
xth_x3 in ref_x3 ref_x4 a1 a2 a3 a4 a5 a6 a7 a8 a1n a2n a3n a4n a5n a6n a7n a8n
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 thresh
xth_x4 in ref_x4 ref_x5 a1 a2 a3 a4 a5 a6 a7 a8 a1n a2n a3n a4n a5n a6n a7n a8n
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 thresh
xth_x5 in ref_x5 ref_x6 a1 a2 a3 a4 a5 a6 a7 a8 a1n a2n a3n a4n a5n a6n a7n a8n
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 thresh
xth_x6 in ref_x6 ref_x7 a1 a2 a3 a4 a5 a6 a7 a8 a1n a2n a3n a4n a5n a6n a7n a8n
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 thresh
xth_x7 in ref_x7 ref_x8 a1 a2 a3 a4 a5 a6 a7 a8 a1n a2n a3n a4n a5n a6n a7n a8n
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 thresh
xth_x8 in ref_x8 ref_x9 a1 a2 a3 a4 a5 a6 a7 a8 a1n a2n a3n a4n a5n a6n a7n a8n
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 thresh
xth_x9 in ref_x9 ref_x10 a1 a2 a3 a4 a5 a6 a7 a8 a1n a2n a3n a4n a5n a6n a7n a8n
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 thresh
xth_x10 in ref_x10 ref_x11 a1 a2 a3 a4 a5 a6 a7 a8 a1n a2n a3n a4n a5n a6n a7n a8n
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 thresh
xth_x11 in ref_x11 ref_x12 a1 a2 a3 a4 a5 a6 a7 a8 a1n a2n a3n a4n a5n a6n a7n a8n
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 thresh
xth_x12 in ref_x12 ref_x13 a1 a2 a3 a4 a5 a6 a7 a8 a1n a2n a3n a4n a5n a6n a7n a8n
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 thresh
xth_x13 in ref_x13 ref_x14 a1 a2 a3 a4 a5 a6 a7 a8 a1n a2n a3n a4n a5n a6n a7n a8n
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 thresh
xth_x14 in ref_x14 ref_x15 a1 a2 a3 a4 a5 a6 a7 a8 a1n a2n a3n a4n a5n a6n a7n a8n
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 thresh
xth_x15 in ref_x15 ref_x16 a1 a2 a3 a4 a5 a6 a7 a8 a1n a2n a3n a4n a5n a6n a7n a8n
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 thresh
xth_x16 in ref_x16 ref_x17 a1 a2 a3 a4 a5 a6 a7 a8 a1n a2n a3n a4n a5n a6n a7n a8n
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 thresh
xth_x17 in ref_x17 ref_x18 a1 a2 a3 a4 a5 a6 a7 a8 a1n a2n a3n a4n a5n a6n a7n a8n
0 10 0 0 0 10 0 0 0 0 10 10 0 0 0 0 thresh
xth_x18 in ref_x18 ref_x19 a1 a2 a3 a4 a5 a6 a7 a8 a1n a2n a3n a4n a5n a6n a7n a8n

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 thresh

xth_x19 in ref_x19 ref_x20 a1 a2 a3 a4 a5 a6 a7 a8 a1n a2n a3n a4n a5n a6n a7n a8n

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 thresh

xth_x20 in ref_x20 ref_x21 a1 a2 a3 a4 a5 a6 a7 a8 a1n a2n a3n a4n a5n a6n a7n a8n

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 thresh

xth_x21 in ref_x21 ref_x22 a1 a2 a3 a4 a5 a6 a7 a8 a1n a2n a3n a4n a5n a6n a7n a8n

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 thresh

xth_x22 in ref_x22 ref_x23 a1 a2 a3 a4 a5 a6 a7 a8 a1n a2n a3n a4n a5n a6n a7n a8n

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 thresh

xth_x23 in ref_x23 ref_x24 a1 a2 a3 a4 a5 a6 a7 a8 a1n a2n a3n a4n a5n a6n a7n a8n

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 thresh

xth_x24 in ref_x24 ref_x25 a1 a2 a3 a4 a5 a6 a7 a8 a1n a2n a3n a4n a5n a6n a7n a8n

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 thresh

xth_x25 in ref_x25 ref_x26 a1 a2 a3 a4 a5 a6 a7 a8 a1n a2n a3n a4n a5n a6n a7n a8n

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 thresh

xth_x26 in ref_x26 ref_x27 a1 a2 a3 a4 a5 a6 a7 a8 a1n a2n a3n a4n a5n a6n a7n a8n

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 thresh

xth_x27 in ref_x27 ref_x28 a1 a2 a3 a4 a5 a6 a7 a8 a1n a2n a3n a4n a5n a6n a7n a8n

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 thresh

xth_x28 in ref_x28 ref_x29 a1 a2 a3 a4 a5 a6 a7 a8 a1n a2n a3n a4n a5n a6n a7n a8n

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 thresh

xth_x29 in ref_x29 ref_x30 a1 a2 a3 a4 a5 a6 a7 a8 a1n a2n a3n a4n a5n a6n a7n a8n

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 thresh

xth_x30 in ref_x30 ref_x31 a1 a2 a3 a4 a5 a6 a7 a8 a1n a2n a3n a4n a5n a6n a7n a8n

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 thresh

xth_x31 in ref_x31 ref_x32 a1 a2 a3 a4 a5 a6 a7 a8 a1n a2n a3n a4n a5n a6n a7n a8n

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 thresh

xth_x32 in ref_x32 ref_x33 a1 a2 a3 a4 a5 a6 a7 a8 a1n a2n a3n a4n a5n a6n a7n a8n

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 thresh

xth_x33 in ref_x33 ref_x34 a1 a2 a3 a4 a5 a6 a7 a8 a1n a2n a3n a4n a5n a6n a7n a8n

10 0 0 0 10 0 0 0 0 0 0 0 0 10 0 0 thresh

xth_x34 in ref_x34 ref_x35 a1 a2 a3 a4 a5 a6 a7 a8 a1n a2n a3n a4n a5n a6n a7n a8n

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 thresh

xth_x35 in ref_x35 ref_x36 a1 a2 a3 a4 a5 a6 a7 a8 a1n a2n a3n a4n a5n a6n a7n a8n
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 thresh
xth_x36 in ref_x36 ref_x37 a1 a2 a3 a4 a5 a6 a7 a8 a1n a2n a3n a4n a5n a6n a7n a8n
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 thresh
xth_x37 in ref_x37 ref_x38 a1 a2 a3 a4 a5 a6 a7 a8 a1n a2n a3n a4n a5n a6n a7n a8n
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 thresh
xth_x38 in ref_x38 ref_x39 a1 a2 a3 a4 a5 a6 a7 a8 a1n a2n a3n a4n a5n a6n a7n a8n
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 thresh
xth_x39 in ref_x39 ref_x40 a1 a2 a3 a4 a5 a6 a7 a8 a1n a2n a3n a4n a5n a6n a7n a8n
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 thresh
xth_x40 in ref_x40 ref_x41 a1 a2 a3 a4 a5 a6 a7 a8 a1n a2n a3n a4n a5n a6n a7n a8n
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 thresh
xth_x41 in ref_x41 ref_x42 a1 a2 a3 a4 a5 a6 a7 a8 a1n a2n a3n a4n a5n a6n a7n a8n
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 thresh
xth_x42 in ref_x42 ref_x43 a1 a2 a3 a4 a5 a6 a7 a8 a1n a2n a3n a4n a5n a6n a7n a8n
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 thresh
xth_x43 in ref_x43 ref_x44 a1 a2 a3 a4 a5 a6 a7 a8 a1n a2n a3n a4n a5n a6n a7n a8n
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 thresh
xth_x44 in ref_x44 ref_x45 a1 a2 a3 a4 a5 a6 a7 a8 a1n a2n a3n a4n a5n a6n a7n a8n
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 thresh
xth_x45 in ref_x45 ref_x46 a1 a2 a3 a4 a5 a6 a7 a8 a1n a2n a3n a4n a5n a6n a7n a8n
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 thresh
xth_x46 in ref_x46 ref_x47 a1 a2 a3 a4 a5 a6 a7 a8 a1n a2n a3n a4n a5n a6n a7n a8n
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 thresh
xth_x47 in ref_x47 ref_x48 a1 a2 a3 a4 a5 a6 a7 a8 a1n a2n a3n a4n a5n a6n a7n a8n
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 thresh
xth_x48 in ref_x48 ref_x49 a1 a2 a3 a4 a5 a6 a7 a8 a1n a2n a3n a4n a5n a6n a7n a8n
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 thresh
xth_x49 in ref_x49 ref_x50 a1 a2 a3 a4 a5 a6 a7 a8 a1n a2n a3n a4n a5n a6n a7n a8n
0 0 0 0 0 0 0 0 0 0 10 0 0 0 0 0 0 0 0 thresh
xth_x50 in ref_x50 ref_x51 a1 a2 a3 a4 a5 a6 a7 a8 a1n a2n a3n a4n a5n a6n a7n a8n
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 thresh
xth_x51 in ref_x51 ref_x52 a1 a2 a3 a4 a5 a6 a7 a8 a1n a2n a3n a4n a5n a6n a7n a8n

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 thresh
xth_x52 in ref_x52 ref_x53 a1 a2 a3 a4 a5 a6 a7 a8 a1n a2n a3n a4n a5n a6n a7n a8n
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 thresh
xth_x53 in ref_x53 ref_x54 a1 a2 a3 a4 a5 a6 a7 a8 a1n a2n a3n a4n a5n a6n a7n a8n
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 thresh
xth_x54 in ref_x54 ref_x55 a1 a2 a3 a4 a5 a6 a7 a8 a1n a2n a3n a4n a5n a6n a7n a8n
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 thresh
xth_x55 in ref_x55 ref_x56 a1 a2 a3 a4 a5 a6 a7 a8 a1n a2n a3n a4n a5n a6n a7n a8n
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 thresh
xth_x56 in ref_x56 ref_x57 a1 a2 a3 a4 a5 a6 a7 a8 a1n a2n a3n a4n a5n a6n a7n a8n
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 thresh
xth_x57 in ref_x57 ref_x58 a1 a2 a3 a4 a5 a6 a7 a8 a1n a2n a3n a4n a5n a6n a7n a8n
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 thresh
xth_x58 in ref_x58 ref_x59 a1 a2 a3 a4 a5 a6 a7 a8 a1n a2n a3n a4n a5n a6n a7n a8n
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 thresh
xth_x59 in ref_x59 ref_x60 a1 a2 a3 a4 a5 a6 a7 a8 a1n a2n a3n a4n a5n a6n a7n a8n
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 thresh
xth_x60 in ref_x60 ref_x61 a1 a2 a3 a4 a5 a6 a7 a8 a1n a2n a3n a4n a5n a6n a7n a8n
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 thresh
xth_x61 in ref_x61 ref_x62 a1 a2 a3 a4 a5 a6 a7 a8 a1n a2n a3n a4n a5n a6n a7n a8n
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 thresh
xth_x62 in ref_x62 ref_x63 a1 a2 a3 a4 a5 a6 a7 a8 a1n a2n a3n a4n a5n a6n a7n a8n
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 thresh
xth_x63 in ref_x63 0 a1 a2 a3 a4 a5 a6 a7 a8 a1n a2n a3n a4n a5n a6n a7n a8n 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 thresh
xth_x64 in 0 0 a1 a2 a3 a4 a5 a6 a7 a8 a1n a2n a3n a4n a5n a6n a7n a8n
0 0 0 0 0 0 0 0 10 0 0 0 10 0 0 0 thresh
y axis********
xth_y1 in2 ref_y1 ref_y2 a1 a2 a3 a4 a5 a6 a7 a8 a1n a2n a3n a4n a5n a6n a7n a8n
0 0 0 10 10 10 0 0 0 0 0 0 0 0 0 0 thresh
xth_y2 in2 ref_y2 ref_y3 a1 a2 a3 a4 a5 a6 a7 a8 a1n a2n a3n a4n a5n a6n a7n a8n
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 thresh
xth_y3 in2 ref_y3 ref_y4 a1 a2 a3 a4 a5 a6 a7 a8 a1n a2n a3n a4n a5n a6n a7n a8n

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 thresh

xth_y4 in2 ref_y4 ref_y5 a1 a2 a3 a4 a5 a6 a7 a8 a1n a2n a3n a4n a5n a6n a7n a8n

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 thresh

xth_y5 in2 ref_y5 ref_y6 a1 a2 a3 a4 a5 a6 a7 a8 a1n a2n a3n a4n a5n a6n a7n a8n

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 thresh

xth_y6 in2 ref_y6 ref_y7 a1 a2 a3 a4 a5 a6 a7 a8 a1n a2n a3n a4n a5n a6n a7n a8n

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 thresh

xth_y7 in2 ref_y7 ref_y8 a1 a2 a3 a4 a5 a6 a7 a8 a1n a2n a3n a4n a5n a6n a7n a8n

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 thresh

xth_y8 in2 ref_y8 ref_y9 a1 a2 a3 a4 a5 a6 a7 a8 a1n a2n a3n a4n a5n a6n a7n a8n

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 thresh

xth_y9 in2 ref_y9 ref_y10 a1 a2 a3 a4 a5 a6 a7 a8 a1n a2n a3n a4n a5n a6n a7n a8n

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 thresh

xth_y10 in2 ref_y10 ref_y11 a1 a2 a3 a4 a5 a6 a7 a8 a1n a2n a3n a4n a5n a6n a7n a8n

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 thresh

xth_y11 in2 ref_y11 ref_y12 a1 a2 a3 a4 a5 a6 a7 a8 a1n a2n a3n a4n a5n a6n a7n a8n

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 thresh

xth_y12 in2 ref_y12 ref_y13 a1 a2 a3 a4 a5 a6 a7 a8 a1n a2n a3n a4n a5n a6n a7n a8n

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 thresh

xth_y13 in2 ref_y13 ref_y14 a1 a2 a3 a4 a5 a6 a7 a8 a1n a2n a3n a4n a5n a6n a7n a8n

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 thresh

xth_y14 in2 ref_y14 ref_y15 a1 a2 a3 a4 a5 a6 a7 a8 a1n a2n a3n a4n a5n a6n a7n a8n

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 thresh

xth_y15 in2 ref_y15 ref_y16 a1 a2 a3 a4 a5 a6 a7 a8 a1n a2n a3n a4n a5n a6n a7n a8n

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 thresh

xth_y16 in2 ref_y16 ref_y17 a1 a2 a3 a4 a5 a6 a7 a8 a1n a2n a3n a4n a5n a6n a7n a8n

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 thresh

xth_y17 in2 ref_y17 ref_y18 a1 a2 a3 a4 a5 a6 a7 a8 a1n a2n a3n a4n a5n a6n a7n a8n

0 10 0 0 0 0 0 0 0 0 0 10 0 0 0 0 thresh

xth_y18 in2 ref_y18 ref_y19 a1 a2 a3 a4 a5 a6 a7 a8 a1n a2n a3n a4n a5n a6n a7n a8n

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 thresh

xth_y19 in2 ref_y19 ref_y20 a1 a2 a3 a4 a5 a6 a7 a8 a1n a2n a3n a4n a5n a6n a7n a8n

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 thresh

xth_y20 in2 ref_y20 ref_y21 a1 a2 a3 a4 a5 a6 a7 a8 a1n a2n a3n a4n a5n a6n a7n a8n
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 thresh
xth_y21 in2 ref_y21 ref_y22 a1 a2 a3 a4 a5 a6 a7 a8 a1n a2n a3n a4n a5n a6n a7n a8n
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 thresh
xth_y22 in2 ref_y22 ref_y23 a1 a2 a3 a4 a5 a6 a7 a8 a1n a2n a3n a4n a5n a6n a7n a8n
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 thresh
xth_y23 in2 ref_y23 ref_y24 a1 a2 a3 a4 a5 a6 a7 a8 a1n a2n a3n a4n a5n a6n a7n a8n
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 thresh
xth_y24 in2 ref_y24 ref_y25 a1 a2 a3 a4 a5 a6 a7 a8 a1n a2n a3n a4n a5n a6n a7n a8n
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 thresh
xth_y25 in2 ref_y25 ref_y26 a1 a2 a3 a4 a5 a6 a7 a8 a1n a2n a3n a4n a5n a6n a7n a8n
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 thresh
xth_y26 in2 ref_y26 ref_y27 a1 a2 a3 a4 a5 a6 a7 a8 a1n a2n a3n a4n a5n a6n a7n a8n
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 thresh
xth_y27 in2 ref_y27 ref_y28 a1 a2 a3 a4 a5 a6 a7 a8 a1n a2n a3n a4n a5n a6n a7n a8n
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 thresh
xth_y28 in2 ref_y28 ref_y29 a1 a2 a3 a4 a5 a6 a7 a8 a1n a2n a3n a4n a5n a6n a7n a8n
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 thresh
xth_y29 in2 ref_y29 ref_y30 a1 a2 a3 a4 a5 a6 a7 a8 a1n a2n a3n a4n a5n a6n a7n a8n
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 thresh
xth_y30 in2 ref_y30 ref_y31 a1 a2 a3 a4 a5 a6 a7 a8 a1n a2n a3n a4n a5n a6n a7n a8n
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 thresh
xth_y31 in2 ref_y31 ref_y32 a1 a2 a3 a4 a5 a6 a7 a8 a1n a2n a3n a4n a5n a6n a7n a8n
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 thresh
xth_y32 in2 ref_y32 ref_y33 a1 a2 a3 a4 a5 a6 a7 a8 a1n a2n a3n a4n a5n a6n a7n a8n
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 thresh
xth_y33 in2 ref_y33 ref_y34 a1 a2 a3 a4 a5 a6 a7 a8 a1n a2n a3n a4n a5n a6n a7n a8n
10 0 10 0 0 0 0 0 0 0 0 0 10 10 0 0 thresh
xth_y34 in2 ref_y34 ref_y35 a1 a2 a3 a4 a5 a6 a7 a8 a1n a2n a3n a4n a5n a6n a7n a8n
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 thresh
xth_y35 in2 ref_y35 ref_y36 a1 a2 a3 a4 a5 a6 a7 a8 a1n a2n a3n a4n a5n a6n a7n a8n
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 thresh
xth_y36 in2 ref_y36 ref_y37 a1 a2 a3 a4 a5 a6 a7 a8 a1n a2n a3n a4n a5n a6n a7n a8n

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 thresh

xth_y37 in2 ref_y37 ref_y38 a1 a2 a3 a4 a5 a6 a7 a8 a1n a2n a3n a4n a5n a6n a7n a8n

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 thresh

xth_y38 in2 ref_y38 ref_y39 a1 a2 a3 a4 a5 a6 a7 a8 a1n a2n a3n a4n a5n a6n a7n a8n

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 thresh

xth_y39 in2 ref_y39 ref_y40 a1 a2 a3 a4 a5 a6 a7 a8 a1n a2n a3n a4n a5n a6n a7n a8n

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 thresh

xth_y40 in2 ref_y40 ref_y41 a1 a2 a3 a4 a5 a6 a7 a8 a1n a2n a3n a4n a5n a6n a7n a8n

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 thresh

xth_y41 in2 ref_y41 ref_y42 a1 a2 a3 a4 a5 a6 a7 a8 a1n a2n a3n a4n a5n a6n a7n a8n

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 thresh

xth_y42 in2 ref_y42 ref_y43 a1 a2 a3 a4 a5 a6 a7 a8 a1n a2n a3n a4n a5n a6n a7n a8n

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 thresh

xth_y43 in2 ref_y43 ref_y44 a1 a2 a3 a4 a5 a6 a7 a8 a1n a2n a3n a4n a5n a6n a7n a8n

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 thresh

xth_y44 in2 ref_y44 ref_y45 a1 a2 a3 a4 a5 a6 a7 a8 a1n a2n a3n a4n a5n a6n a7n a8n

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 thresh

xth_y45 in2 ref_y45 ref_y46 a1 a2 a3 a4 a5 a6 a7 a8 a1n a2n a3n a4n a5n a6n a7n a8n

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 thresh

xth_y46 in2 ref_y46 ref_y47 a1 a2 a3 a4 a5 a6 a7 a8 a1n a2n a3n a4n a5n a6n a7n a8n

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 thresh

xth_y47 in2 ref_y47 ref_y48 a1 a2 a3 a4 a5 a6 a7 a8 a1n a2n a3n a4n a5n a6n a7n a8n

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 thresh

xth_y48 in2 ref_y48 ref_y49 a1 a2 a3 a4 a5 a6 a7 a8 a1n a2n a3n a4n a5n a6n a7n a8n

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 thresh

xth_y49 in2 ref_y49 ref_y50 a1 a2 a3 a4 a5 a6 a7 a8 a1n a2n a3n a4n a5n a6n a7n a8n

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 thresh

xth_y50 in2 ref_y50 ref_y51 a1 a2 a3 a4 a5 a6 a7 a8 a1n a2n a3n a4n a5n a6n a7n a8n

0 0 0 0 0 0 0 0 0 10 0 0 0 0 0 0 thresh

xth_y51 in2 ref_y51 ref_y52 a1 a2 a3 a4 a5 a6 a7 a8 a1n a2n a3n a4n a5n a6n a7n a8n

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 thresh

xth_y52 in2 ref_y52 ref_y53 a1 a2 a3 a4 a5 a6 a7 a8 a1n a2n a3n a4n a5n a6n a7n a8n

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 thresh

```
xth_y53 in2 ref_y53 ref_y54 a1 a2 a3 a4 a5 a6 a7 a8    a1n a2n a3n a4n a5n a6n a7n a8n
0 0 0 0 0 0 0 0    0 0 0 0 0 0 0 0  thresh
xth_y54 in2 ref_y54 ref_y55 a1 a2 a3 a4 a5 a6 a7 a8    a1n a2n a3n a4n a5n a6n a7n a8n
0 0 0 0 0 0 0 0    0 0 0 0 0 0 0 0  thresh
xth_y55 in2 ref_y55 ref_y56 a1 a2 a3 a4 a5 a6 a7 a8    a1n a2n a3n a4n a5n a6n a7n a8n
0 0 0 0 0 0 0 0    0 0 0 0 0 0 0 0  thresh
xth_y56 in2 ref_y56 ref_y57 a1 a2 a3 a4 a5 a6 a7 a8    a1n a2n a3n a4n a5n a6n a7n a8n
0 0 0 0 0 0 0 0    0 0 0 0 0 0 0 0  thresh
xth_y57 in2 ref_y57 ref_y58 a1 a2 a3 a4 a5 a6 a7 a8    a1n a2n a3n a4n a5n a6n a7n a8n
0 0 0 0 0 0 0 0    0 0 0 0 0 0 0 0  thresh
xth_y58 in2 ref_y58 ref_y59 a1 a2 a3 a4 a5 a6 a7 a8    a1n a2n a3n a4n a5n a6n a7n a8n
0 0 0 0 0 0 0 0    0 0 0 0 0 0 0 0  thresh
xth_y59 in2 ref_y59 ref_y60 a1 a2 a3 a4 a5 a6 a7 a8    a1n a2n a3n a4n a5n a6n a7n a8n
0 0 0 0 0 0 0 0    0 0 0 0 0 0 0 0  thresh
xth_y60 in2 ref_y60 ref_y61 a1 a2 a3 a4 a5 a6 a7 a8    a1n a2n a3n a4n a5n a6n a7n a8n
0 0 0 0 0 0 0 0    0 0 0 0 0 0 0 0  thresh
xth_y61 in2 ref_y61 ref_y62 a1 a2 a3 a4 a5 a6 a7 a8    a1n a2n a3n a4n a5n a6n a7n a8n
0 0 0 0 0 0 0 0    0 0 0 0 0 0 0 0  thresh
xth_y62 in2 ref_y62 ref_y63 a1 a2 a3 a4 a5 a6 a7 a8    a1n a2n a3n a4n a5n a6n a7n a8n
0 0 0 0 0 0 0 0    0 0 0 0 0 0 0 0  thresh
xth_y63 in2 ref_y63 ref_y64 a1 a2 a3 a4 a5 a6 a7 a8    a1n a2n a3n a4n a5n a6n a7n a8n
0 0 0 0 0 0 0 0    0 0 0 0 0 0 0 0  thresh
xth_y64 in2 0 0                a1 a2 a3 a4 a5 a6 a7 a8    a1n a2yl a3n a4n a5n a6n a7n a8n
0 0 0 0 0 0 0 0    10 0 10 0 0 0 0 0  thresh
vin1 a1 a1_in 0
vin2 a2 a2_in 0
vin3 a3 a3_in 0
vin4 a4 a4_in 0
vin5 a5 a5_in 0
vin6 a6 a6_in 0
vin7 a7 a7_in 0
vin8 a8 a8_in 0
vin1n a1n a1_in 0
```

```
vin2n a2n a2_in 0
vin3n a3n a3_in 0
vin4n a4n a4_in 0
vin5n a5n a5_in 0
vin6n a6n a6_in 0
vin7n a7n a7_in 0
vin8n a8n a8_in 0
xarea1 a1_in a1_out 10 a1_outt  areasel
xarea2 a2_in a2_out 10 a2_outt  areasel
xarea3 a3_in a3_out 10 a3_outt  areasel
xarea4 a4_in a4_out 10 a4_outt  areasel
xarea5 a5_in a5_out 10 a5_outt  areasel
xarea6 a6_in a6_out 10 a6_outt  areasel
xarea7 a7_in a7_out 10 a7_outt  areasel
xarea8 a8_in a8_out 10 a8_outt  areasel
vtat1 a1_in a1_outt 0
vtat2 a2_in a2_outt 0
vtat3 a3_in a3_outt 0
vtat4 a4_in a4_outt 0
vtat5 a5_in a5_outt 0
vtat6 a6_in a6_outt 0
vtat7 a7_in a7_outt 0
vtat8 a8_in a8_outt 0
vo1 a1_out a1_outb 0
vo2 a2_out a2_outb 0
vo3 a3_out a3_outb 0
vo4 a4_out a4_outb 0
vo5 a5_out a5_outb 0
vo6 a6_out a6_outb 0
vo7 a7_out a7_outb 0
vo8 a8_out a8_outb 0
*CURRENT MIRRORS
mcma1_1 a1_outb a1_outb 0 0 nfet w=2u l=2u
```

```
mcma1_2 a1_im1 a1_outb 0 0 nfet w=2u l=2u
mcma1_3 a1_im1 a1_im1 10 10 pfet w=2u l=2u
mcma1_4 a1_copy1 a1_im1 10 10 pfet w=2u l=2u
mcma1_5 a_t1 a1_im1 10 10 pfet w=2u l=2u
mcma2_1 a2_outb a2_outb 0 0 nfet w=2u l=2u
mcma2_2 a2_im1 a2_outb 0 0 nfet w=2u l=2u
mcma2_3 a2_im1 a2_im1 10 10 pfet w=2u l=2u
mcma2_4 a2_copy1 a2_im1 10 10 pfet w=2u l=2u
mcma2_5 a_t1 a2_im1 10 10 pfet w=2u l=2u
mcma3_1 a3_outb a3_outb 0 0 nfet w=2u l=2u
mcma3_2 a3_im1 a3_outb 0 0 nfet w=2u l=2u
mcma3_3 a3_im1 a3_im1 10 10 pfet w=2u l=2u
mcma3_4 a3_copy1 a3_im1 10 10 pfet w=2u l=2u
mcma3_5 a_t1 a3_im1 10 10 pfet w=2u l=2u
mcma4_1 a4_outb a4_outb 0 0 nfet w=2u l=2u
mcma4_2 a4_im1 a4_outb 0 0 nfet w=2u l=2u
mcma4_3 a4_im1 a4_im1 10 10 pfet w=2u l=2u
mcma4_4 a4_copy1 a4_im1 10 10 pfet w=2u l=2u
mcma4_5 a_t1 a4_im1 10 10 pfet w=2u l=2u
mcma5_1 a5_outb a5_outb 0 0 nfet w=2u l=2u
mcma5_2 a5_im1 a5_outb 0 0 nfet w=2u l=2u
mcma5_3 a5_im1 a5_im1 10 10 pfet w=2u l=2u
mcma5_4 a5_copy1 a5_im1 10 10 pfet w=2u l=2u
mcma5_5 a_t1 a5_im1 10 10 pfet w=2u l=2u
mcma6_1 a6_outb a6_outb 0 0 nfet w=2u l=2u
mcma6_2 a6_im1 a6_outb 0 0 nfet w=2u l=2u
mcma6_3 a6_im1 a6_im1 10 10 pfet w=2u l=2u
mcma6_4 a6_copy1 a6_im1 10 10 pfet w=2u l=2u
mcma6_5 a_t1 a6_im1 10 10 pfet w=2u l=2u
mcma7_1 a7_outb a7_outb 0 0 nfet w=2u l=2u
mcma7_2 a7_im1 a7_outb 0 0 nfet w=2u l=2u
mcma7_3 a7_im1 a7_im1 10 10 pfet w=2u l=2u
mcma7_4 a7_copy1 a7_im1 10 10 pfet w=2u l=2u
```

mcma7_5 a_t1 a7_im1 10 10 pfet w=2u l=2u
mcma8_1 a8_outb a8_outb 0 0 nfet w=2u l=2u
mcma8_2 a8_im1 a8_outb 0 0 nfet w=2u l=2u
mcma8_3 a8_im1 a8_im1 10 10 pfet w=2u l=2u
mcma8_4 a8_copy1 a8_im1 10 10 pfet w=2u l=2u
mcma8_5 a_t1 a8_im1 10 10 pfet w=2u l=2u
vgreat a_t1 a_t1b 0
xnorm a1_copy1 a2_copy1 a3_copy1 a4_copy1 a5_copy1 a6_copy1 a7_copy1 a8_copy1 a1_norm a2_norm a3_norm a4_norm a5_norm a6_norm a7_norm a8_norm a_t1b normalizer
mcp1a a1_norm a1_norm 10 10 pfet w=2u l=2u
mcp1b a1_normb a1_norm 10 10 pfet w=2u l=2u
mcp2a a2_norm a2_norm 10 10 pfet w=2u l=2u
mcp2b a2_normb a2_norm 10 10 pfet w=2u l=2u
mcp3a a3_norm a3_norm 10 10 pfet w=2u l=2u
mcp3b a3_normb a3_norm 10 10 pfet w=2u l=2u
mcp4a a4_norm a4_norm 10 10 pfet w=2u l=2u
mcp4b a4_normb a4_norm 10 10 pfet w=2u l=2u
mcp5a a5_norm a5_norm 10 10 pfet w=2u l=2u
mcp5b a5_normb a5_norm 10 10 pfet w=2u l=2u
mcp6a a6_norm a6_norm 10 10 pfet w=2u l=2u
mcp6b a6_normb a6_norm 10 10 pfet w=2u l=2u
mcp7a a7_norm a7_norm 10 10 pfet w=2u l=2u
mcp7b a7_normb a7_norm 10 10 pfet w=2u l=2u
mcp8a a8_norm a8_norm 10 10 pfet w=2u l=2u
mcp8b a8_normb a8_norm 10 10 pfet w=2u l=2u
vnorm1 a1_normb a1_normc 0
vnorm2 a2_normb a2_normc 0
vnorm3 a3_normb a3_normc 0
vnorm4 a4_normb a4_normc 0
vnorm5 a5_normb a5_normc 0
vnorm6 a6_normb a6_normc 0
vnorm7 a7_normb a7_normc 0

```
vnorm8 a8_normb a8_normc 0
**WEIGHT ASSIGNING***
*****W1 OUT******
Va1_d1 ina1_1 0 0
Va1_d2 ina1_2 0 3.3
Va1_d3 ina1_3 0 0
Va1_d4 ina1_4 0 3.3
Va1_d5 ina1_5 0 0
XD1 a1_normc ina1_1 ina1_2 ina1_3 ina1_4 ina1_5 m_out1 10 dac
*****W2 OUT******
Va2_d1 ina2_1 0 3.3
Va2_d2 ina2_2 0 0
Va2_d3 ina2_3 0 3.3
Va2_d4 ina2_4 0 0
Va2_d5 ina2_5 0 3.3
XD2 a2_normc ina2_1 ina2_2 ina2_3 ina2_4  ina2_5 m_out2 10 dac
*****W3 OUT******
Va3_d1 ina3_1 0 0
Va3_d2 ina3_2 0 0
Va3_d3 ina3_3 0 3.3
Va3_d4 ina3_4 0 3.3
Va3_d5 ina3_5 0 0
XD3 a3_normc ina3_1 ina3_2 ina3_3 ina3_4 ina3_5 m_out3 10 dac
*****W4 OUT******
Va4_d1 ina4_1 0 3.3
Va4_d2 ina4_2 0 3.3
Va4_d3 ina4_3 0 3.3
Va4_d4 ina4_4 0 3.3
Va4_d5 ina4_5 0 3.3
XD4 a4_normc ina4_1 ina4_2 ina4_3 ina4_4 ina4_5 m_out4 10 dac
*****W5 OUT******
Va5_d1 ina5_1 0 0
Va5_d2 ina5_2 0 0
```

```
Va5_d3 ina5_3 0 3.3
Va5_d4 ina5_4 0 3.3
Va5_d5 ina5_5 0 3.3
XD5 a5_normc ina5_1 ina5_2 ina5_3 ina5_4 ina5_5 m_out5 10 dac
*****W6 OUT******
Va6_d1 ina6_1 0 3.3
Va6_d2 ina6_2 0 3.3
Va6_d3 ina6_3 0 0
Va6_d4 ina6_4 0 3.3
Va6_d5 ina6_5 0 0
XD6 a6_normc ina6_1 ina6_2 ina6_3 ina6_4 ina6_5 m_out6 10 dac
*****W7 OUT******
Va7_d1 ina7_1 0 0
Va7_d2 ina7_2 0 0
Va7_d3 ina7_3 0 0
Va7_d4 ina7_4 0 0
Va7_d5 ina7_5 0 0
XD7 a7_normc ina7_1 ina7_2 ina7_3 ina7_4 ina7_5 m_out7 10 dac
*****W8 OUT******
Va8_d1 ina8_1 0 0
Va8_d2 ina8_2 0 0
Va8_d3 ina8_3 0 0
Va8_d4 ina8_4 0 0
Va8_d5 ina8_5 0 0
XD8 a8_normc ina8_1 ina8_2 ina8_3 ina8_4 ina8_5 m_out8 10 dac
****************************************
vout1 m_out1 tet 0
vout2 m_out2 tet 0
vout3 m_out3 tet 0
vout4 m_out4 tet 0
vout5 m_out5 tet 0
vout6 m_out6 tet 0
vout7 m_out7 tet 0
```

vout8 m_out8 tet 0
VHOLY_OUT tet 0 0

A.2. VHDL Codes

A.2.1. Boundary Synapse Connectting

```
USE WORK.basic_utilities.ALL;
entity Threshold_Set is
  port( WR,DT,PN,RST,XY:in bit;
   SW_I: in bit_vector( 5 downto 0 );
   AR: in bit_vector( 2 downto 0);
         SW_X_P,SW_X_N,SW_Y_P,SW_Y_N :out bit_2d_64);
end;
architecture behavioral of Threshold_Set is
begin
   process
      variable ar_no,sw_no:integer:=0;
   begin
     if( WR = '1' ) then
        if( RST = '1') then
          SW_X_P<= (OTHERS => (OTHERS => '0'));
          SW_X_N<= (OTHERS => (OTHERS => '0'));
          SW_Y_P<= (OTHERS => (OTHERS => '0'));
          SW_Y_N<= (OTHERS => (OTHERS => '0'));
        else
          binary_to_integer( SW_I,sw_no);
          binary_to_integer( AR,ar_no);
          if( XY = '0') then
            if( PN = '1') then
              SW_X_P(ar_no)( sw_no) <= DT;
            else
              SW_X_N(ar_no)( sw_no) <= DT;
```

```
            end if;
          else
            if( PN = '1') then
              SW_Y_P(ar_no)( sw_no) <= DT;
            else
              SW_Y_N(ar_no)( sw_no) <= DT;
            end if;
          end if;
        end if;
      end if;
      wait for 1 ns;
    end process;
end;
```

A.2.2. Weght Setting and Controlling

```
USE WORK.basic_utilities.ALL;
entity Weight_Set is
  port( WR,RST:in bit;
   DATA: in bit_vector( 4 downto 0 );
   AR: in bit_vector( 2 downto 0);
         D_O :out bit_2d_5
         );
end;
architecture behavioral of Weight_Set is
begin
   process
      variable ar_no:integer:=0;
   begin
     if( WR = '1' ) then
       if( RST = '1') then
         D_O <= (OTHERS => (OTHERS => '0'));
       else
```

```
            binary_to_integer( AR,ar_no);
            D_O(ar_no) <= DATA;
          end if;
       end if;
       wait for 1 ns;
    end process;
end;
```

A.2.3. Complete Digital Circuit

```
LIBRARY WORK;
use WORK.ALL;
entity final is
  port(
   WR_T,DT_T,PN_T,RST_T,XY_T:in bit;
   SW_I_T: in bit_vector( 3 downto 0 );
   AR_T: in bit_vector( 2 downto 0);
   WR_W,RST_W:in bit;
   DT_W: in bit_vector( 4 downto 0 );
   AR_W: in bit_vector( 2 downto 0);
        T1_X,T2_X,T3_X,T4_X,T5_X,T6_X,T7_X,T8_X,T9_X,T10_X,T11_X,T12_X,T1
3_X,T14_X,T15_X,T16_X,
        T1_Y,T2_Y,T3_Y,T4_Y,T5_Y,T6_Y,T7_Y,T8_Y,T9_Y,T10_Y,T11_Y,T12_Y,T1
3_Y,T14_Y,T15_Y,T16_Y: OUT bit_vector( 15 downto 0);
        W1,W2,W3,W4,W5,W6,W7,W8:out bit_vector( 4 downto 0));
end;

architecture structural of final is
begin
x1: ENTITY work.Threshold_Set(behavioral)
        PORT MAP ( WR_T, DT_T, PN_T, RST_T, XY_T, SW_I_T, AR_T, T1_X,
T2_X,T3_X,T4_X,T5_X,T6_X,T7_X,T8_X,T9_X,T10_X,T11_X,T12_X,T13_X,T14_X,
T15_X,T16_X,T1_Y,T2_Y,T3_Y,T4_Y,T5_Y,T6_Y,T7_Y,T8_Y,T9_Y,T10_Y,T11_Y,T
```

```
12_Y,T13_Y,T14_Y,T15_Y,T16_Y);
x2: ENTITY work.Weight_Set(behavioral)
        PORT MAP(WR_W,RST_W,DT_W,AR_W,W1,W2,W3,W4,W5,W6,W7,W8);
end;
```

A.3 Matlab Codes

A.3.1. Theoretical Model

```
count=1;
val=0;
TotalOut=zeros(30,30);
TotalWeighted=zeros(65,65);
TotalAR=zeros(65,65);
while val <= 2.5
   layer1_X(count)=val;
   layer1_Y(count)=val;
   connect1_X(count)=0;
   connect1_Y(count)=0;
   connectA1_X(count)=0;
   connectA1_Y(count)=0;
   connectA2_X(count)=0;
   connectA2_Y(count)=0;
   connectA3_X(count)=0;
   connectA3_Y(count)=0;
   connectA4_X(count)=0;
   connectA4_Y(count)=0;
   connectA5_X(count)=0;
   connectA5_Y(count)=0;
   connectA6_X(count)=0;
   connectA6_Y(count)=0;
   connectA7_X(count)=0;
   connectA7_Y(count)=0;
```

```
    connectA8_X(count)=0;
    connectA8_Y(count)=0;
    connectA9_X(count)=0;
    connectA9_Y(count)=0;
    count=count+1;
    val=val+0.0390625;
end
count=1;
while count<=9
    AR(count)=0;
    weight(count)=0;
     weight_con(count)=0;
    count=count+1;
end
%%%%%%%%%1st area
connectA1_X(1)=1;
connectA1_X(10)=-1;
connectA1_Y(1)=1;
connectA1_Y(10)=-1;
%%%%%%%%%2nd area
connectA2_X(1)=1;
connectA2_X(10)=-1;
connectA2_Y(10)=1;
connectA2_Y(29)=-1;
%%%%%%%%%2nd area
connectA3_X(1)=1;
connectA3_X(10)=-1;
connectA3_Y(29)=1;
connectA3_Y(64)=-1;
%%%%%%%%%1st area
connectA4_X(10)=1;
connectA4_X(29)=-1;
connectA4_Y(1)=1;
```

```
connectA4_Y(10)=-1;
%%%%%%%%%2nd area
connectA5_X(10)=1;
connectA5_X(29)=-1;
connectA5_Y(10)=1;
connectA5_Y(29)=-1;
%%%%%%%%%2nd area
connectA6_X(10)=1;
connectA6_X(29)=-1;
connectA6_Y(29)=1;
connectA6_Y(64)=-1;
%%%%%%%%%1st area
connectA7_X(29)=1;
connectA7_X(64)=-1;
connectA7_Y(1)=1;
connectA7_Y(10)=-1;
%%%%%%%%%2nd area
connectA8_X(29)=1;
connectA8_X(64)=-1;
connectA8_Y(10)=1;
connectA8_Y(29)=-1;
%%%%%%%%%2nd area
connectA9_X(29)=1;
connectA9_X(64)=-1;
connectA9_Y(29)=1;
connectA9_Y(64)=-1;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
weight(1)=0.5*31;
weight(2)=0.5*31;
weight(3)=1*31;
weight(4)=0.5*31;
weight(5)=0.5*31;
weight(6)=1*31;
```

```
weight(7)=0.5*31;
weight(8)=1*31;
weight(9)=1*31;
count=1;
val=0;
countx=1;
county=1;
X=0;
FL_weightX =0;
FL_weightY =0;
while X<=1.55
  ax(countx)=countx;
  Y=0;
  county=1;
  while Y<=1.55
    ay(county)=county;
    ar=1;
    %%%%1st LAyer to 2nd LAyer
    while ar<=9
      AR(ar)=0;
      count=1;
      while(count<=65)
        if( ar==1)
          FL_weightX = connectA1_X(count);
          FL_weightY = connectA1_Y(count);
        end
        if(ar==2)
          FL_weightX = connectA2_X(count);
          FL_weightY = connectA2_Y(count);
        end
        if(ar==3)
          FL_weightX = connectA3_X(count);
          FL_weightY = connectA3_Y(count);
```

```
            end
            if(ar==4)
              FL_weightX = connectA4_X(count);
              FL_weightY = connectA4_Y(count);
            end
            if(ar==5)
              FL_weightX = connectA5_X(count);
              FL_weightY = connectA5_Y(count);
            end
            if(ar==6)
              FL_weightX = connectA6_X(count);
              FL_weightY = connectA6_Y(count);
            end
            if(ar==7)
              FL_weightX = connectA7_X(count);
              FL_weightY = connectA7_Y(count);
            end
            if(ar==8)
              FL_weightX = connectA8_X(count);
              FL_weightY = connectA8_Y(count);
            end
            if(ar==9)
              FL_weightX = connectA9_X(count);
              FL_weightY = connectA9_Y(count);
            end
            AR(ar)=AR(ar)+(10e-6+10e-6*tanh(10*(X-layer1_X(count))))*0.5*
FL_weightX;
            AR(ar)=AR(ar)+(10e-6+10e-6*tanh(10*(Y-layer1_Y(count))))*0.5*
FL_weightY;
            count=count+1;
          end
          ar=ar+1;
      end
```

```
      %%% 2ndLayer to 3rdLAyer
      ar=1;
      while ar<=9
       if(AR(ar)-10e-6 > 0)
          TotalWeighted(countx,county)=
TotalWeighted(countx,county)+weight(ar)*(AR(ar)-10e-6);
          TotalAR(countx,county)=TotalAR(countx,county)+AR(ar)-10e-6;
       end
        ar=ar+1;
      end
      if(TotalAR(countx,county)>0)
          if(TotalWeighted(countx,county)>0.0e-4)
             TotalOut(countx,county)=10e-6*TotalWeighted(countx,county)/
TotalAR(countx,county);
          end;
      end;
      Y=Y+0.05;
      county=county+1;
   end
   X=X+0.05;
   countx=countx+1;
end
[X,Y]=meshgrid(1:1:countx-1,1:1:county-1);
Z=TotalOut;
colormap(white)
surf(Y,X,Z)
```

REFERENCES

1. Zadeh, L. A., "Fuzzy Sets, Information and Control", *New York Academic*, Vol. 8, pp. 338–353, 1965.

2. Mamdani, E.H. and S. Assilian, "An Experiment in Linguistic Synthesis with A Fuzzy Logic Controller", *International Journal of Man-Machine Studies*, Vol. 7, No. 1, pp. 1–13, 1975.

3. Menger, K., "Statistical Metrics", *Proceedings of National Academy of Sciences*, pp. 535–537, 1942.

4. Hohle, U., "Probabilistic Uniformization of Fuzzy Topologies", *Fuzzy Sets and Systems*, Vol. 1, Issue 4, 1978.

5. Klement, E.P., "A Theory of Fuzzy Measures: A Survey", in Gupta, M. and Sanchez, E., North Holland (eds.), *Fuzzy Information and Decision Processes*, pp. 59–66, Amsterdam, 1982.

6. Dubois, D. and H. Prade, "Triangular Norms for Fuzzy Sets", *Proceedings to International Symposium of Fuzzy Sets*, pp. 39–68, Linz, 1981.

7. Takagi, T. and M. Sugeno, "Derivation of Fuzzy Control Rules from Human Operator's Control Action", *Proceedings of IFAC Symposium Fuzzy Information Knowledge Representation and Decision Analysis*, pp. 55–60, July 1989.

8. Detyniecki, M., R. R. Yager, and Bouchon-Meunier, "Reducing T-norms and Augmenting T-conorms", *International Journal of General Systems*, Vol. 31, pp. 265–276, 2002.

9. Wilamovski, B. M., "Analog VLSI Hardware for Fuzzy Systems", *IEEE Industrial Elec. Society IECON '98 Proc. of the 24th Annual Conference 1*, pp. 52–55, 1998.

10. Izhizuka, O., K. Tanno, Z. Tang and H. Matsumoto, “Design of A Fuzzy Controller with Normalization Circuits”, *IEEE International Conference on Fuzzy Systems*, 8-12 March 1992, pp. 1303–1308, 1992.

11. Vidal-Verdu, F., R. Navas and A. Rodriguez-Vazquez, “A Modular CMOS Analog Fuzzy Controller”, *Proceedings of the Sixth IEEE International Conference on Fuzzy Systems*, 1-5 July 1997, Vol. 2, pp. 647–652, 1997.

12. Baturone, I., S. Sanchez-Solano, A. Barriga and J.L Huertas, “Implementation of CMOS Fuzzy Controllers as Mixed-Signal Integrated Circuits”, *IEEE Transactions on Fuzzy Systems*, Vol. 5, Issue 1, pp. 1–19, 1997.

13. Gilbert, B., “A Monolithic 16-channel Analog Array Normalizer”, *IEEE Journal of Solid-State Circuits*, Vol. 19, pp. 956–963, 1984.

14. Maher, M.C., S.P. Deweerth, M.A. Mahowald and C.A. Mead, “Implementing Neural Architectures Using Analog VLSI Circuits”, *IEEE Transactions on Circuits and Systems*, Vol. 36, pp. 643–652, 1989.

15. Harrer, H., J.A. Nossek, and R. Stelzl, “An Analog Implementation of Discrete-Time Cellular Neural Networks”, *IEEE Transactions on Neural Networks*, Vol. 3, Issue 3, pp. 466–476, 1992.

16. Vittoz, W.A., “Analog VLSI Implementation of Neural Networks”, *Proceedings of IEEE international symposium on Circuits and Systems*, Vol. 4, pp. 2524–2527, New Orleans, 1990.

17. Yelten, B., *Hardware Design and Simulation of Artificial Neural Networks Blocks*, B.S. Project, Boğaziçi University, 2006.

18. Sedra, A.S. and K.C. Smith, *Microelectronic Circuits*, Oxford University Press, New York, 2004.

19. Wilamovski, B. M., R.C. Jaeger and M. O. Kaynak,, "Neuro-Fuzzy Architecture for CMOS Implementation", *IEEE Transactions on Industrial Electronics*, Vol. 46, Issue 6, pp. 1132–1136, 1999.

20. Diorio, C., P. Hasler, A. Minch and C.A. Mead, "A Single-Transistor Silicon Synapse", *IEEE Transactions on Electron Devices*, Vol. 43, Issue 11, pp. 1972–1980, 1996.

21. Diorio, C., P. Hasler, B. A. Minch and C. A. Mead, "A Complementary Pair of Four-Terminal Silicon Synapses*"*, *Analog Integrated Circuits and Signal Processing*, Vol. 13, No. 1–2, pp. 153–166, 1997.

22. Koosh, V. F. and R. Goodman, "VLSI Neural Network with Digital Weights and Analog Multipliers", *IEEE International Symposium on Circuits and Systems*, 6-9 May 2001, Vol. 3, pp. 233–236, 2001.

23. Hollis, P. W. and J. J. Paulos, "Artificial Neural Networks Using MOS Analog Multipliers", *IEEE Journal of Solid-State Circuits*, Vol. 25, Issue 3, pp. 849–855, 1990.

24. Eberhardt, S., T. Duong, and A. Thakoor , "Design of Parallel Hardware Neural Network Systems from Custom VLSI Building Block Chips", *Proceedings of International Joint Conference on Neural Networks*, , Vol. 11, pp. 183, 12-22 June 1989.

25. Kub, F., K. Moon and I. Mack, "Cascadable 32 X 32 Vector-Matrix Multiplier for Artificial Neural Networks", *International Joint Conference on Neural Networks Presentation*, 12-22 June 1989.

26. Chua, S. C., E. K. Wong and V. C. Koo, "Intelligent Pool Decision System Using Zero-Order Sugeno Fuzzy System*", Journal of Intelligent and Robotic Systems*, Vol.44, No. 2, October 2005.

27. Nauck. D., F. Klawonn and R. Kruse, *Foundations of Neuro-Fuzzy Systems*, John Wiley & Sons, New York, 1997.

28. Malki, H. A. and C. G. Umeh, "Design of A Fuzzy Logic Based Level Controller", *Journal of Engineering Technology*, pp. 32–38, Spring 2000.

29. Gurney K., *An Introduction to Neural Networks*, Routledge, London, 1997.

30. Yamakawa, T., "A Fuzzy Inference Engine in Nonlinear Analog Mode and Its Application to a Fuzzy Logic Control", *IEEE Transaction on Neural Networks*, Vol. 4, No. 3, pp. 496–522, May 1993.

31. Sasaki, M., N. Ishikawa, F. Ueno, and T. Inoue, "Current Mode Analog Fuzzy Hardware with Voltage Input Interface and Normalization Locked Loop", *Second IEEE International Conference on Fuzzy Systems*, Vol. E57-A (6), pp. 451–457, June 1992.

32. Enz, C. and E. Vittoz, "CMOS Low-Power Analog Circuit Design", *Designing Low Power Digital Systems, Emerging Technologies*, pp. 79–133, 1996.

33. Sasaki, M. and F. Ueno, "A VLSI Implementation of Fuzzy Logic Controller Using Current Mode CMOS Circuits", *Third International Conference on Industrial Fuzzy Control and Intelligent Systems*, pp. 215 – 220, 1-3 December 1993.

34. Wilamovski, B., Private Correspondence, Auburn University, Alabama, 2006.