Implementation of Hopfield Neural Network Using Double Gate MOSFET

A thesis presented to

the faculty of

the Russ College of Engineering and Technology of Ohio University

In partial fulfillment

of the requirements for the degree

Master of Science

Amit Parasmal Borundiya

March 2008

This thesis titled

Implementation of Hopfield Neural Network Using Double Gate MOSFET

by

AMIT PARASMAL BORUNDIYA

has been approved for

the School of Electrical Engineering Computer Science

and the Russ College of Engineering and Technology by

Abstract

BORUNDIYA AMIT ,M.S., March 2008, Electrical Engineering

Implementation of Hopfield Neural Netwrok Using Double Gate MOSFET (94 pp.)

Director of Thesis: Janusz A. Starzyk

Hopfield Neural Network has been used to solve the constraints satisfaction problems. To make these networks solve problem in real time, independent of the size, would require building a massively parallel structure. A CMOS circuit can be used to construct such network to find the solution. Current CMOS technology is reaching its physical limitation in deep submicron regime and new devices are explored which can provide scalability in accordance to Moore's Law. To further increase the network capacity double gate transistors can be used. Double gate MOSFET model of the hysteresis neuron proposed in this thesis utilizes 8 transistors as compared to 60 transistors needed with an operational amplifier's model. This structure not only reduces the count of transistors by 86% but also demonstrates that larger circuits of double gate MOSFETs can be built, bolstering the faith in double gate MOSFET devices as a possible substitute of CMOS devices in a near future.

Approved: ____

Janusz A. Starzyk

Professor of Electrical and Computer Science

Acknowledgments

“A good teacher Explains, a Superior teacher Demonstrates while a Great teacher Inspires” unknown.

Dr. Starzyk truly epitomizes these words. He is one of the most influential people in my life. I have gained immense knowledge and experience from him, which is invaluable. Without his expertise in the field of Neural Network and analog circuit design, this thesis wouldn’t have been possible. I sincerely thank him for demonstrating confidence in me and providing an opportunity to work under his able guidance.

I am also thankful to Dr. Savas Kaya, for allowing me to use the Double Gate MOSFET model and other resources. His advice and guidance was of great help in shaping the thesis. The various discussions with him helped me.

I thank Dr. Jeffrey Dill and Dr. Xiaoping Shen for being on the committee and reviewing my thesis.

I am grateful to Mrs. Zofia Starzyk for all her support, encouragement and providing me with a helping hand. Without her help and consideration it wouldn’t have been a smooth sailing for me.

I take this opportunity to thank my beloved parents and all my family members for believing in my ability and providing constant encouragement and support.

I also thank Ihitashri, for her patience and support which she constantly provided during my Masters program.

I also thank my friends Lankesh, Santosh, Basawaraj and Anish for their help. Lastly I thank GOD, the Almighty for providing me with strength and for His choicest blessings.

Dedicated to

Mr. and Mrs. Starzyk

Table of Contents

Page

Abstract 3

Acknowledgments 4

List of Tables 10

List of Figures 11

1. Introduction 13

1.1 Background and Motivation 13

1.2 Research Goals 14

1.3 Thesis Organization 14

2. Chaos 16

2.2 History: 16

2.3. Negative Resistance: 18

2.3.1 Negative Resistance Converter: 18

2.4 Chua’s Circuit 20

2.4.1 Chua’s Circuit Using Opamp 21

2.4.2 Simulation Results 23

3. Hopfield Neural Network 25

3.1 Introduction 25

3.1.1 Historical Background 26

3.1.2 Neural Network vs Conventional Computer 27

3.2 Biological Neuron Model ... 28

3.3 Artificial Neuron Model ... 29

3.4 Neural Network Architectures ... 33

3.5 Hopfield neural network ... 34

3.6 Hysteresis Neuron Model ... 38

3.6.1 Hysteresis Neural Network System ... 39

3.6.2 Hysteresis Activation Function ... 41

3.6.3 Equilibrium Points ... 41

3.7 Simulations of Hopfield Neuron Model ... 42

3.7.1 Simulation using Multisim 8.0: ... 43

3.7.2 Simulation using OrCAD 10.3 ... 46

3.7.3 Simulation using Spice3 ... 49

4. N-queens Problem ... 52

4.1 Introduction ... 52

4.2 System Stability for Hysteresis Neural Network: ... 52

4.3 Problem Statement and Solution ... 53

4.4 Network Topology ... 58

5. Double Gate MOSFET implementation of HNN ... 60

5.1 Introduction ... 60

5.2 Double Gate MOSFET Structure ... 62

5.3 Double gate Spice Model UFDG ... 63

5.4 Double Gate Structures Self Bias Amplifier ... 64

5.4.1 Spice 3 Code 65

5.4.2 Simulations Results 66

5.5 Double gate Structure for Schmitt Trigger 66

5.5.1 Spice 3 Code 67

5.5.2 Simulation Results 68

5.6 Double gate Neuron Model 69

5.6.1 Spice 3 Code 70

5.6.2 Simulation Results 73

5.7 Summary 75

5.7.1 Hysteresis Neuron Model using OTA 76

6. Conclusion and Future Work 79

6.1 Conclusion 79

6.2 Future Work 80

References 81

Appendix 1: Multisim Model Details 86

1.1LM 741 Models Details 86

1.2 BZX83C10 Model Details 86

Appendix 2: OrCAD Model Details 87

2.1 Opamp 741 87

2.2 Zener Diode 1N4744A Model 88

Appendix 3: Spice3 Models Details 89

3.1 Spice3 Zener Model: 89

3.2 Spice3 opamp Model: 89

3.3 Running the Netlist file 90

Appendix 4: Matlab Code for N-Queen Problem 91

Appendix 5: DG/SOI MOSFET Device-Line Parameters 92

Appendix 6: DG MOSFET Model Line and Structural Parameters used in Simulation .. 93

6.1 NMOS Double Gate Model 93

6.2 PMOS Double Gate Model 93

6.3 Running UFDG Simulation on Spice3 94

List of Tables

Page

Table 1: Brain vs Computer Comparison 27
Table 2: Number of Solutions for Different Size of Queen Problem 57

List of Figures

Page

Figure 1: Opamp based negative impedance converter. 19
Figure 2: I-V graph for negative impedance converter. 19
Figure 3: Approximate I-V characteristics of opamp in all regions of operations. 20
Figure 4: Chua’s circuit. 21
Figure 5: Opamp based implementation of Chua’s circuit. 22
Figure 6: Simulation result for Chua’s circuit. 23
Figure 7: Simulation result for Chua’s circuit. 24
Figure 8: Anatomy of biological neuron [Lode07]. 28
Figure 9: Mathematical neuron model. 29
Figure 10: Threshold logic function. 31
Figure 11: Piecewise linear function. 31
Figure 12: Log sigmoid function. 32
Figure 13: Multilayer feed forward architecture. 33
Figure 14: Recurrent neural network architecture. 34
Figure 15: State of Hopfield neural network. 36
Figure 16: Hopfield neural network architecture. 37
Figure 17: General hysteresis neuron. 39
Figure 18: Hysteresis neural network system. 40
Figure 19: Hysteresis function. 41
Figure 20: Multisim schematic of hysteresis neuron. 43
Figure 21: Double circular chaos phenomenon using HNN. 44
Figure 22: Double well defined circular chaos using HNN. 45
Figure 23: Doubly attractors using HNN. 45
Figure 24: Hysteresis characteristics using HNN. 46
Figure 25: Hysteresis neuron circuit with OrCAD 10.3. 47
Figure 26: OrCAD simulation for chaos. 47
Figure 27: Circular chaos using OrCAD. 48
Figure 28: Hysteresis characteristics using OrCAD. 48
Figure 29: Chaotic attractor from hysteresis neuron using Spice3. 50
Figure 30: Double circular chaotic attractors using Spice3. 51
Figure 31: Hysteresis characteristics using Spice3. 51
Figure 32: One of the solutions for 8 queen problem. 54
Figure 33: Matlab simulation results for 8x8 queen problem. 56
Figure 34: Connection topology for 8-queen network. 58
Figure 35: Moore’s law graph [Inte07]. 60
Figure 36: Non classical CMOS devices [Hutc02]. 61
Figure 37: Double gate MOSFET structure. 62
Figure 38: General double gate operation modes. 63
Figure 39: Self bias configuration of double gate MOSFET. 65
Figure 40: Double gate MOSFET self-bias amplifier configuration. 66

Figure 41: Schmitt trigger using double gate MOSFET. 67
Figure 42: Schmitt trigger with conjugate bias ±0.5V... 68
Figure 43: Schmitt trigger with conjugate bias ±0.3V... 69
Figure 44: Double gate neuron model. 70
Figure 45: Double gate MOSFET hysteresis neuron simulation one. 74
Figure 46: Double gate MOSFET neuron model simulation two. 74
Figure 47: RC OTA design for hysteresis chaos generator. 76
Figure 48: OTA bases hysteresis neuron. 77
Figure 49: Chaos using RC OTA. 77
Figure 50: Hysteresis using RC OTA. 78

1. Introduction

1.1 Background and Motivation

Constraint satisfaction problems have been the subject of research for many years. Efforts have been made to solve them by writing efficient algorithms [Sosi90] and coding efficient computer programs [Fink87]. All these approaches were hampered by the sequential nature of the computer simulation. J.J. Hopfield proposed the Neural Network approach to find the solution for these NP hard problems [Hopf85]. In recent times the chaotic phenomenon was explored helping to solve these problems [Toku97]. Chaotic phenomenon helps to search for the global optimal solution. An example application is used in this thesis to illustrate this approach. In the approach that combines chaotic search with neural network solution, hysteresis neural network is used to solve the N-queens problem [Naka99]. The solution can be efficiently implemented based on hysteresis neurons whose analog implementation uses just a few components. The silicon chips are approaching VLSI scaling limit [ITRS07] and different design approaches and improved silicon devices are studied to extend the Moore's laws [More69] in coming years. Double gate MOSFET has been studied for various digital and analog applications [Vara05] but this thesis is the first attempt to use it in the field of neural networks. It shows how to efficiently implement hysteresis neural network that can solve N-queens problem.

To summarize, if we leverage the capabilities of double-gate MOSFET to build the hysteresis neuron with significantly less transistors, larger network can be designed in the same silicon area. These networks have regular structures of identical building blocks;

hence the size of the building block is crucial for the overall size of the network. These networks can be made to solve various constraint satisfaction problems in real time independent of the problem size.

The objective of this thesis work is to show how double-gate MOSFET can be used to build a hysteresis neuron and using this neuron build a Hopfield neural network to solve a selected application problem. Meeting these objectives bolsters the faith that double-gate MOSFET can be utilized to build larger integrated circuits helping to improve design efficiency, reduce the design area and internal delays. Other objective of this thesis is to analyze the chaotic phenomenon and study organization of the hysteresis neural networks to propose a circuit topology that efficiently solves the 8-queen problem.

1.2 Research Goals

Specific research goals of this thesis are

- Broadly study the chaos phenomenon and its occurrence in electrical circuits,
- Analyze the hysteresis neuron and its stability,
- Analyze the hysteresis neural network for solving the N-queen problem,
- Propose the circuit topology for solving 8-queen problem,
- Implement the hysteresis neuron using double gate MOSFET.

1.3 Thesis Organization

The thesis is organized in six chapters.

Chapter 2, broadly studies the chaos phenomenon in nature and in the electrical circuits. The non-linearity of the electrical circuit characteristics is discussed to get better understanding of the chaos phenomenon in electric circuits. One of such nonlinear circuits that demonstrates various chaotic phenomenons is Chua's circuit [Mats84]. In Chapter 2, Chua's circuit is implemented and simulated using Multisim software.

Chapter 3 discusses general principles of Hopfield neural networks. The chapter introduces the concept of the neural network and compares it with the conventional computer. Biological and artificial neuron models are explained to get better understanding of how the network works. Feed forward and recursive network architectures are explained. The Hopfield hysteresis neuron model is analyzed with respect to the hysteresis activation function and its equilibrium points. This neuron model is implemented using operational amplifier and is simulated using Multisim, OrCAD and Spice3 simulators.

Chapter 4 states the N-queen problem and discusses the stability criterion of the hysteresis neural network as a cost function to find the solution. The network topology for solving 8 queens problem is also proposed in this chapter. This approach can be easily extended to a general N-queens problem.

Chapter 5 discusses the double-gate MOSFET structure and its UFDG model [UFDG06]. Self bias amplifier and Schmitt trigger circuits are implemented and simulated using Spice3. Using these blocks, a double-gate MOSFET hysteresis neuron is built and simulated.

Chapter 6 provides the conclusion for the thesis work and discusses future work.

2. Chaos

2.1 Introduction:

Chaos theory describes the behavior of certain non linear dynamical systems which under specific conditions exhibit dynamics that are very sensitive to initial conditions [Wiki07]. Most of the systems are deterministic, meaning that if provided with the same initial conditions to the system, the system could predict the final result. It is certain that, the result obtained for any number of iterations, using the same initial conditions, will be the same [Utex07]. But, in the case of non linear dynamic systems a small error in the initial condition can produce a different result. This is because the non linear systems are very sensitive in nature and as a consequence the results observed seems to be random [Wiki07], [Utex07].

2.2 History:

The notion of determinism was challenged with the discovery of the chaos phenomenon.

The various laws of physics have given a good connection between cause and effect around us. These laws helped to get almost accurate predictions of any physical system if the initial conditions are known. Newton's laws of motion best demonstrate the philosophy of determinism. Using these laws we can predict the time the planets take to orbit around sun, projectiles, time for tides [Utex07]. To make a correct prediction based on these laws, it is required to provide precise initial conditions to an infinite precision. However, providing the infinite precision is almost impossible for any known system.

Thus, if accurate predictions cannot be made using almost the same initial conditions the system can be chaotic.

Edward Lorenz, a professor at Massachusetts Institute of Technology in early 1960 was the first to study the chaos phenomenon. He wrote a simplified weather model program which can be simulated on a digital computer. Precisely, he studied the nature of the air flow current when the air is heated by the Sun rays [Utex07]. His mathematical model contained various equations that modeled this air current. Even though he had a primitive computer to work with, few weather predictions were made. While carrying out these experiments he accidentally discovered the phenomenon of chaos [Utex07], [Thin07].

Lorenz code was running on a computer, which is a completely deterministic system. Therefore, he expected to get same results when he provided the same set of initial conditions. To his surprise he did not get the same results, and the patterns developed differed a lot from the previous iterations. Lorenz believed that he had provided the same numbers in all the iterations for initial conditions, but after careful study of these numbers he came to conclusion that what he believed as the same input values differed slightly from each other [Utex07]. This is due to the precision of the computer. While his print copy could show numbers up to three decimal places, internally the computer had six decimal places precision. He did not notice this difference because the values differed very slightly and could be taken as the same set of values. This laid the foundation for the study of chaotic systems [Thin07]. Thus, in the chaotic system even an imaginably small error in two sets of initial conditions can produce a large discrepancy with the previously obtained results.

Chaos is also manifested in some electrical circuits. Dr. Yoshisuke Ueda, Professor, Department of Electrical Engineering, Kyoto University, independently identified the chaos phenomenon using analog computer in 1961 [Ueda07]. He performed the analog simulation of a non linear oscillator. He stated the chaos demonstrated by the analog computer is truly a natural phenomenon [Ueda07]. Later in 1983, Dr. Leon O. Chua, Professor, Department of Electrical Engineering and Computer Science, at University of Berkeley, designed a simple circuit that exhibit the chaos phenomenon [Mats84]. The circuit was build with few basic components and became a classical example to demonstrate the chaotic phenomenon in the electrical circuits.

2.3. Negative Resistance:

Negative resistance, also known as negative differential resistance is a property of the electrical circuit elements composed of active devices in which, over a certain voltage ranges, current is a decreasing function of voltage [Wiki07]. This range is called negative resistance region.

2.3.1 Negative Resistance Converter:

The operational amplifier (opamp) based schematic diagram for negative resistance converter is as shown in Figure 1, redrawn as described by Chua, [Chua07] has both the positive feedback and the negative feedback.

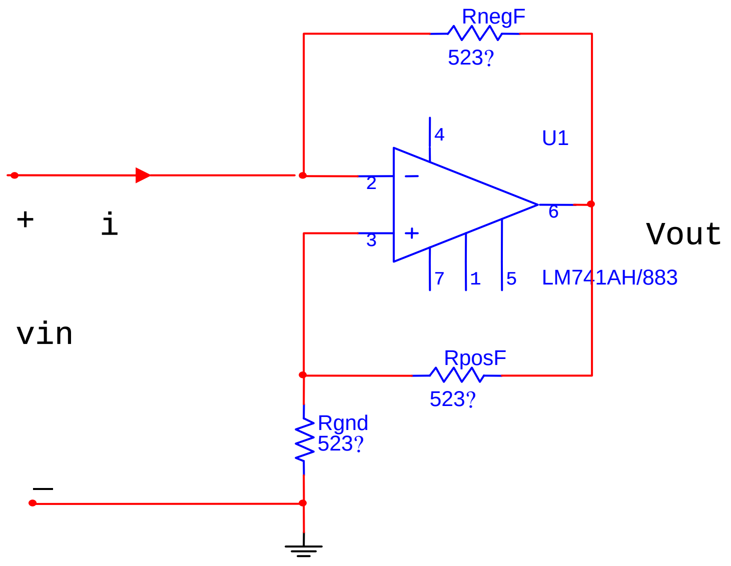


Figure 1: Opamp based negative impedance converter.

Equation 2.3-1 shows the relationship between current and voltage for a negative impendence converter and the results are plotted as shown in Figure 2

$$i = \frac{-\left(R_{posF}\right)}{R_{gnd} \times R_{negF}} \times v_{in} \qquad \textbf{[Chua07] (2.3-1)}$$

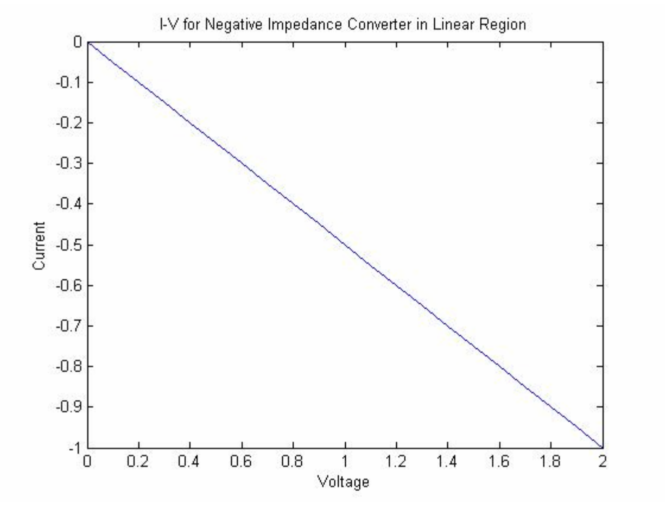


Figure 2: I-V graph for negative impedance converter.

The slope of the curve is negative, and hence the circuit is called the “negative impedance converter”. The opamp is operating in a linear mode; therefore the curve is a straight line. Opamp causes non-linearity and occurs when opamp operates in the saturated region. Figure 3, shows the approximate I-V characteristics of the opamp in all regions of operation as observed in the article by Chua [Chua07].

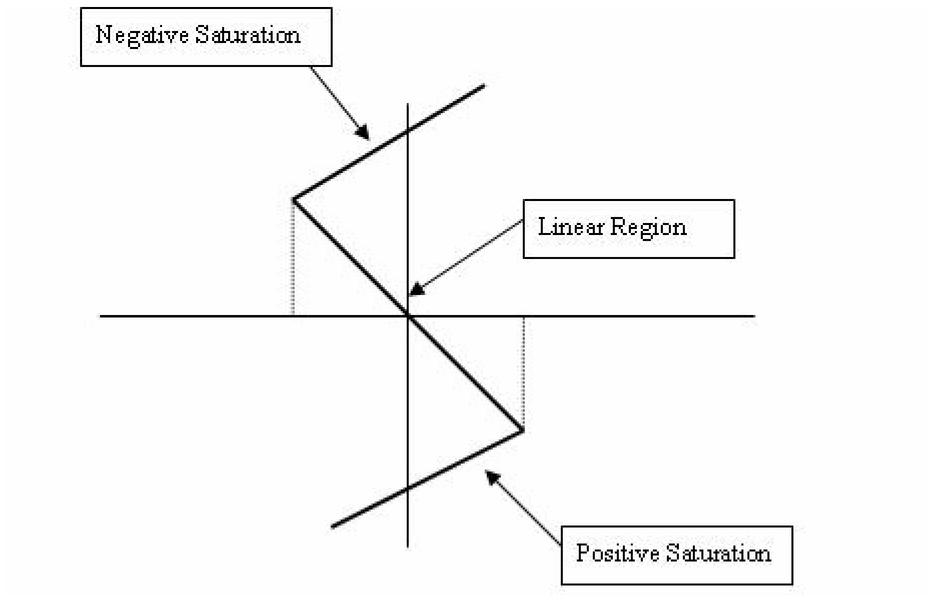


Figure 3: Approximate I-V characteristics of opamp in all regions of operations.

2.4 Chua’s Circuit

Figure 4, shows the basic Chua’s circuit. This circuit [Mats84], exhibit a chaos. The circuit consists of

- A linear resistor R
- Two capacitors C1 and C2
- One inductor L

- One non-linear resistor RN

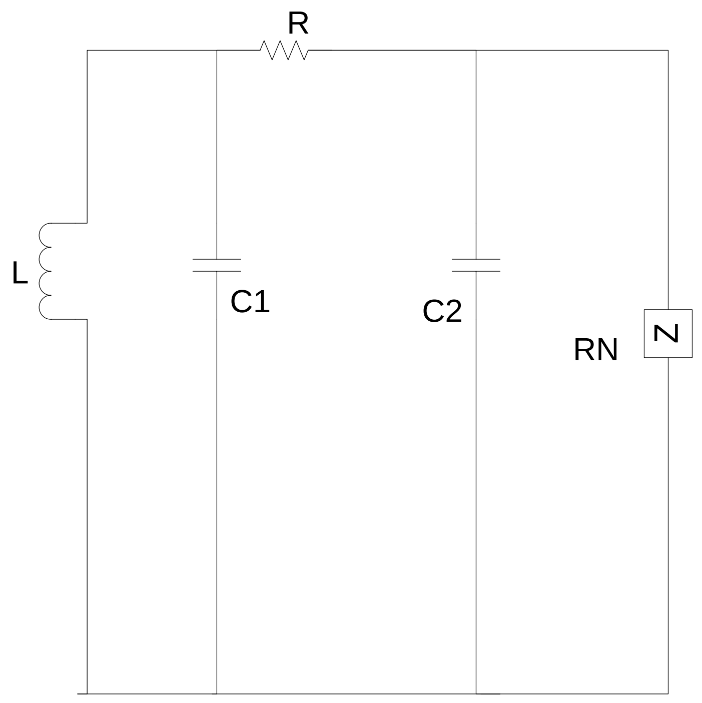


Figure 4: Chua's circuit.

This circuit can be easily built using all basic elements and non-linear resistor, which can be implemented using negative resistance converter [Kenn92]. The current in the non-linear resistance depends upon the applied voltage across its terminal. Hence, it can be considered as a voltage controlled device. From the discussion in Section 2.3.1, opamp can be used to develop negative impedance converter (NIC) and the I-V characteristics consists of three piecewise linear segments. This NIC is the basic block for building the non linear resistance for Chua's circuit.

2.4.1 Chua's Circuit Using Opamp

The complete Op-amp based implementation of Chua's circuit is as shown in the Figure 5

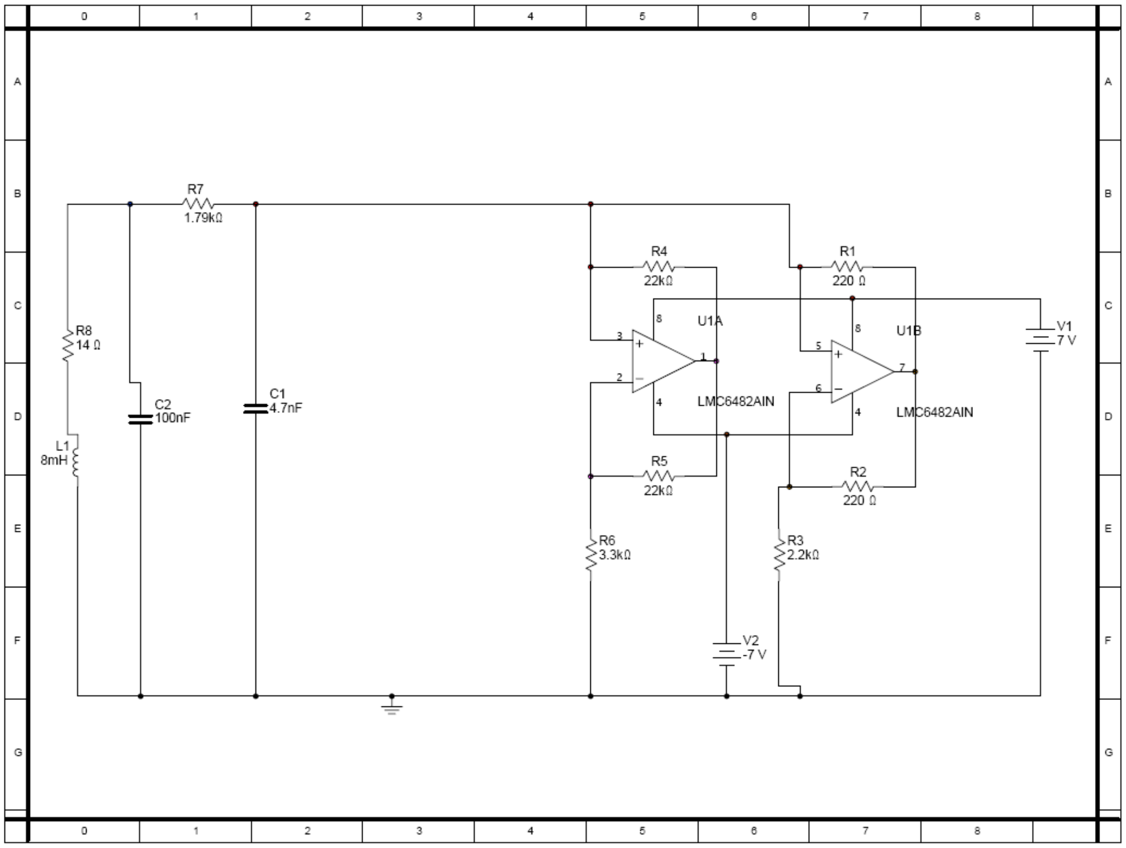


Figure 5: Opamp based implementation of Chua’s circuit.

The above circuit was redrawn in Multisim software using the schematic from Chua’s website [ChMs07]. The two NIC in parallel provides with the non-linear resistance component in the Chua’s circuit as shown in Figure 5. In the above circuit the inductor, resistor and capacitors together implement a conventional oscillator which gives damped oscillations if isolated from other circuit elements [Cros07]. The opamp and other resistors together yield the nonlinear negative resistance. This circuit also provides the energy for the dynamic circuits.

2.4.2 Simulation Results

The circuit was implemented and simulated using Multisim software. The voltage across capacitors C1 was plotted against the voltage across C2. The result was viewed using the simulated Agilent oscilloscope available in Multisim software and the chaos was seen using X-Y display on oscilloscope with probes placed across C1 and C2 as shown in Figure 5, 6 and 7.

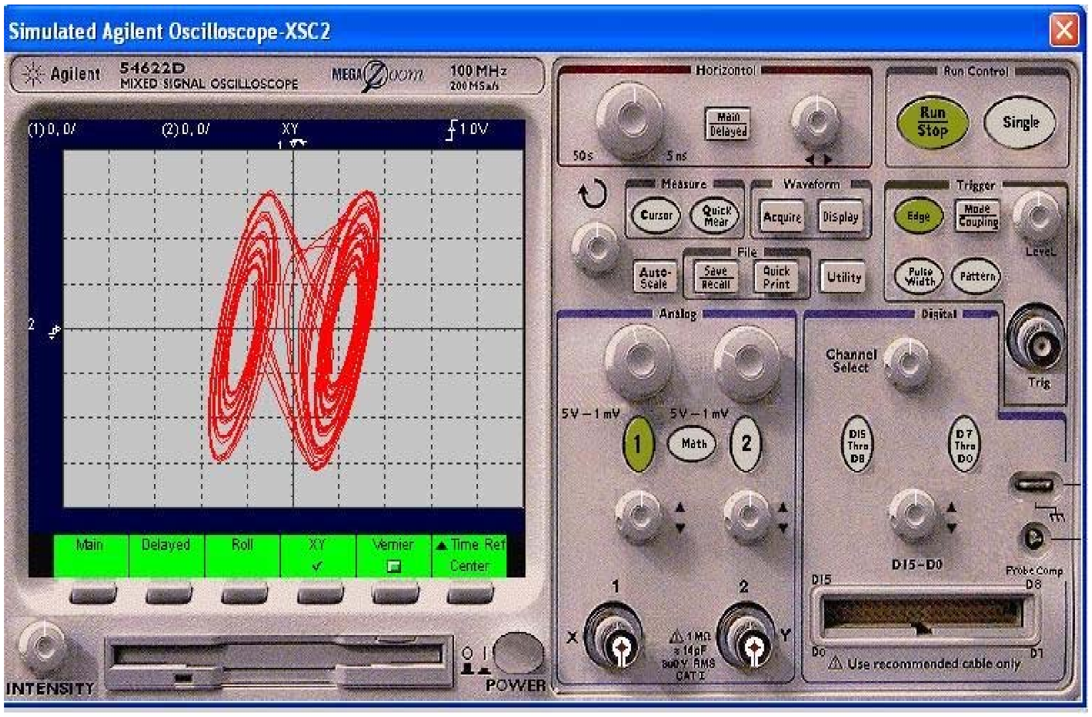


Figure 6: Simulation result for Chua's circuit.

From Figure 6, it can be seen that double scroll attractors can be obtained using relatively simple circuit proposed by Chua.

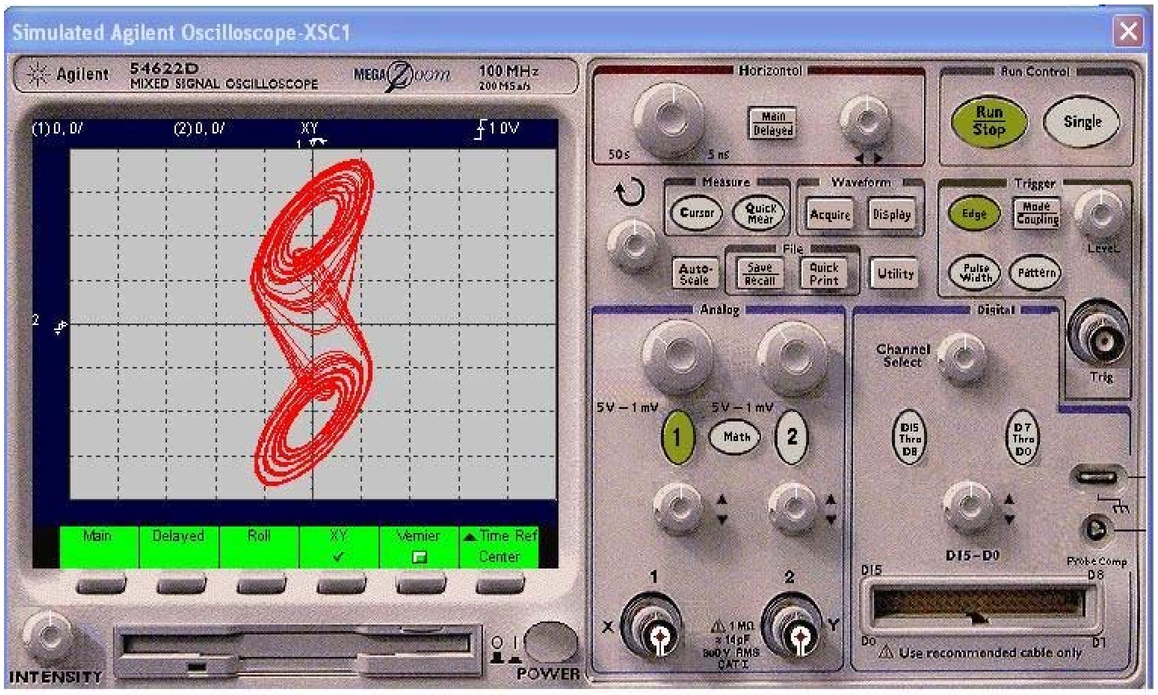


Figure 7: Simulation result for Chua’s circuit.

Figure 7, demonstrates vertical double scroll attractors using different value resistors, capacitors. Depending on the values of circuit parameters used, Chua’s circuit demonstrates various chaos phenomenons.

One of the important limitations of the Chua’s circuit implemented in Figure 5 is that it consists of an inductor which is not easily fabricated in VLSI. This motivated for building circuits which can exhibit chaos phenomenon using the components that can be easily fabricated in VLSI. One of the circuits is discussed in the Chapter 3 using opamp.

3. Hopfield Neural Network

3.1 Introduction

According to Haykin [Simo98], “neural network is a massively parallel distributed processor that has a natural propensity for storing experimental knowledge and making it available for use.” Neural network is similar to the brain in

- The way network acquires the knowledge through a learning process.
- The way network used synaptic weights to store the knowledge.

Thus, the human brain is source of inspiration for all neural networks study. The brain is a highly complex, non-linear, dynamic and parallel processing unit with complex associations between inputs, outputs and internal states. A neural network tries to simulate some of the properties of the brain for solving the real world problems, such as speech recognition, image analysis, and some NP hard problems which are difficult to solve using the conventional computers. This is achievable because the neural network structure is highly parallel (thus improving the speed of operations) and has a property to learn and generalize the input [Simo98]. Following are few important properties of artificial neural network (ANN) described by Haykin [Simo98],

- ANN can be linear as well as non-linear. Many phenomena in the nature are non-linear and are important.
- The ANN learns by constructing the input-output mapping.
- The ANN has the ability to adapt its weight to the changing ambience.
- Each neuron in the network is affected by the activity of the other neurons.

- ANN has ability to self-organize and represents the information it receives during the learning phase.
- The parallel nature of ANN makes it suitable for implementation in VLSI and is inherently fault tolerant.

Most of the discussion in this chapter is based on a Simon Haykin’s [Simo98] book on neural networks and discussion presented by Wenle Zhang in his course on Neural Network. Some of the figures were redrawn using Microsoft Visio for the better understanding of the architecture.

3.1.1 Historical Background

McCulloch and Pitts are pioneers in the field of neural networks (NN); they introduced the first mathematical model of biological neurons [Pitt43]. This neuron model was considered as a competitor for Von Neumann model for basic computing machine as it was capable of performing all Boolean operations. Further development of the NN concepts and applications occurred with the invention of Rosenblatt’s Perceptron [Rose58]. The Perceptron was a single layer and single neuron with ability to perform binary classification for linearly separable patterns. Perceptron model was not able to implement XOR function since it did not have the ability to classify the non-linearly separable patterns. In the last few decades there has been a greater interest in this field attributed to discoveries of back-propagation learning model, Hopfield neural network, recurrent neural networks, and associative memories. Some impressive applications such as image recognition, speech recognition, and associative memory were developed based on various neural network models.

3.1.2 Neural Network vs Conventional Computer

A neural network (NN) does not use the conventional algorithmic approach like a computer does. The computer can solve problems that have a predefined methodology, thus limiting the problem solving ability. But the NN learns like the brain from the associations between data and hence do not have to be programmed for a specific task. Choosing the problem becomes important and since no predefined steps are present, the operation can be unpredictable as compared to computers whose behavior is completely predictable. On average the neuron in the brain works at a speed of 100 Hz as compared to several millions of instructions per second performed by a computer [Simo98]. Despite this slow speed compared to a computer, the brain's network can perform some tasks like recognition, planning, and action control at a remarkable ease as opposed to the computer. Most of the computers cannot be useful if a single component is damaged but the brain network is able to work appreciably well even if there is a significant damage to the parts of the brain.

Table 1: Brain vs Computer Comparison

	Processing Elements	Size	Speed	Architecture	Fault Tolerant	Learning Ability	Energy Per operation
Brain	10^{11} neurons	10^{-6} m	100 Hz	parallel, distributed	Yes	Yes	10^{-16} Joules/Sec
Computer	10^{8} transistors	10^{-7} m	10^{9} Hz	serial, centralized	No	Little	10^{-6} Joules/Sec

Above, Table 1 summaries comparison between the brain and the modern computer based on the readings in the book [Simo98], Wiki [Wiki07].

Computers can be better suited for algorithmic problems while the neural networks can be better for nonalgorithmic problems like classification and recognition.

3.2 Biological Neuron Model

Neurons are the basic functional elements of the neural network in the brain. It is a massively parallel information processing network with about 10^{11} neurons. Each neuron is connected to many (10,000 or more) synapses.

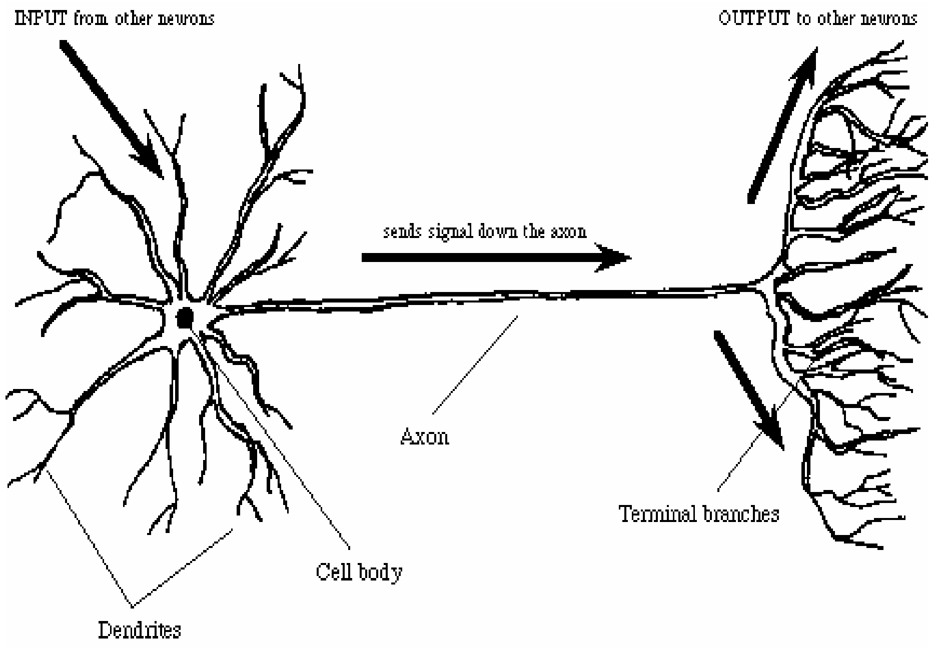


Figure 8: Anatomy of biological neuron [Lode07].

The neuron has many dendrites, an axon and a body. Neurons transmit the signals in the form of the electrical impulses. Dendrites act as an input source to the neuron, as they carry the signal from the other neurons to the cell body of the neuron. The axon conducts the signals away from the cell body. The neurons can either be in a firing or in a resting state. The neuron fires when the total incoming stimulus received is greater than the threshold. The neurons are connected through synapses. A synapse is a tiny gap between the axon of one neuron and a dendrite of other one. When there is an impulse through the neuron cell it transmits a chemical called neuron-transmitter into the synapse. The neurotransmitters, depending upon their strength, induce or inhibit the impulses in the connecting neuron [Simo98], [Wiki07], [Lode07].

3.3 Artificial Neuron Model

The mathematical model of an artificial neuron can be described in the following figure

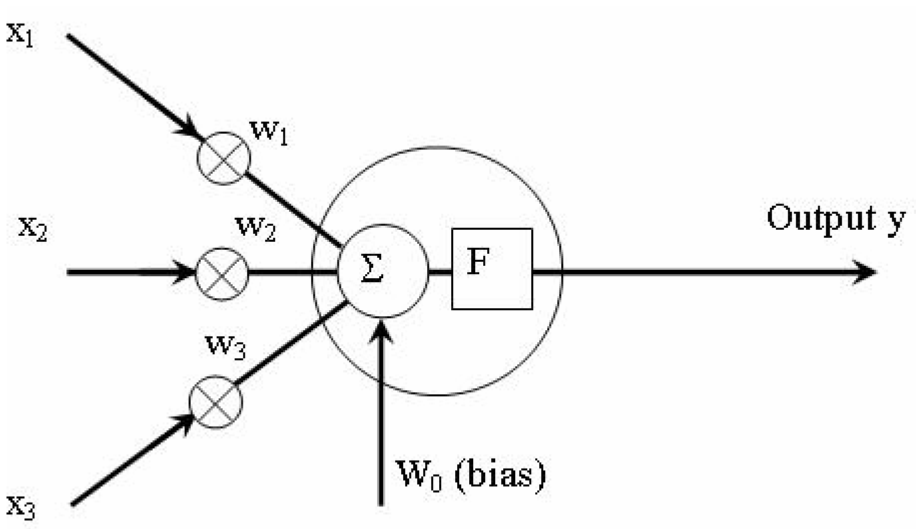


Figure 9: Mathematical neuron model.

The general artificial neuron model has the following five components,

1. Inputs (X_i): the inputs to the neuron come from the external environment or through other neuron outputs. They can have a discrete or a real value.

2. Weights (W_i): The weights are the real-valued numbers. They determine the contribution of each neuron to the system. One of the first things the model does is to calculate the weighted sum of inputs [Scho07].

$$W = w_1 x_1 + w_2 x_2 + w_3 x_3 \qquad \textbf{(3.3-1)}$$

3. Threshold or Bias (u): This is the quantity which is subtracted from the weighted sum to get the input for the transfer function [Scho07]. For simplicity most of the time the bias is regarded as an additional input with values w_0=u and x_0=-1.

$$W = w_0 x_0 + w_1 x_1 + w_2 x_2 + w_3 x_3 \qquad \textbf{(3.3-2)}$$

4. Transfer Function: The transfer function gives the relationship between the input and the output of the neuron. There are various function used to describe the networks transfer function, most of them are nonlinear with a limited output range.

- Threshold function or Step function: This transfer function was used in the McCulloch-Pitts [Pitts43] neuron model called as threshold logic unit (TLU) [Wiki07].

$$\begin{aligned} f(x) &= 1, \quad if \quad x \geq 0 \\ &= 0, \quad if \quad x < 0 \end{aligned} \qquad \textbf{(3.3-3)}$$

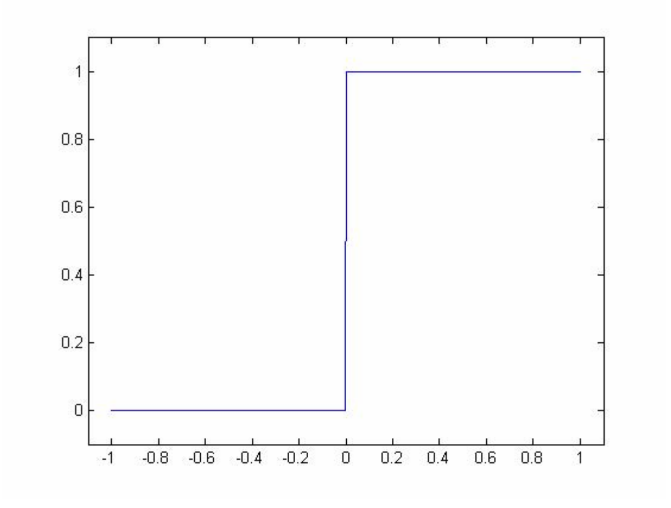


Figure 10: Threshold logic function.

- Piecewise linear function: This transfer function has the form [Wiki07]

$$\begin{aligned} f(x) &= 1, \text{ if } x \ge 1/2 \\ &= x, \text{ if } -1/2 < x < 1/2 \\ &= 0, \text{ if } \mathrm{x} \le -1/2 \end{aligned} \qquad \textbf{(3.3-4)}$$

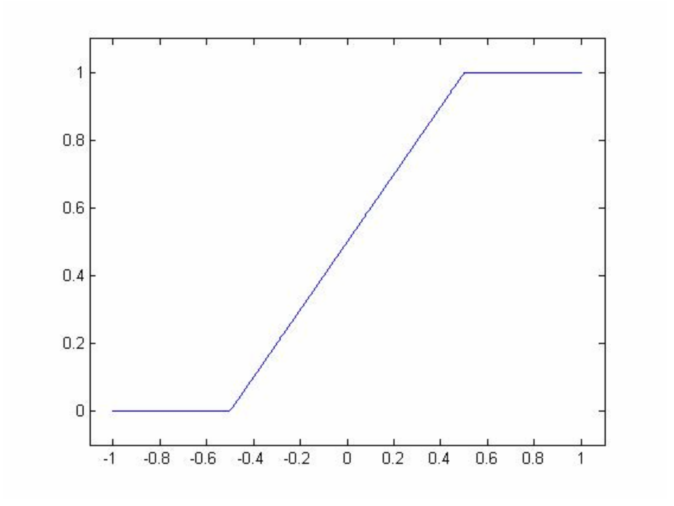


Figure 11: Piecewise linear function.

- Sigmoidal function: The most commonly used transfer function in the neural networks is a sigmoidal function. For instance, a sigmoidal function can be defined as [Wiki07]

$$f(x)=\frac{1}{1+e^{-ax}} \tag{3.3-5}$$

Where a, is the slope parameter.

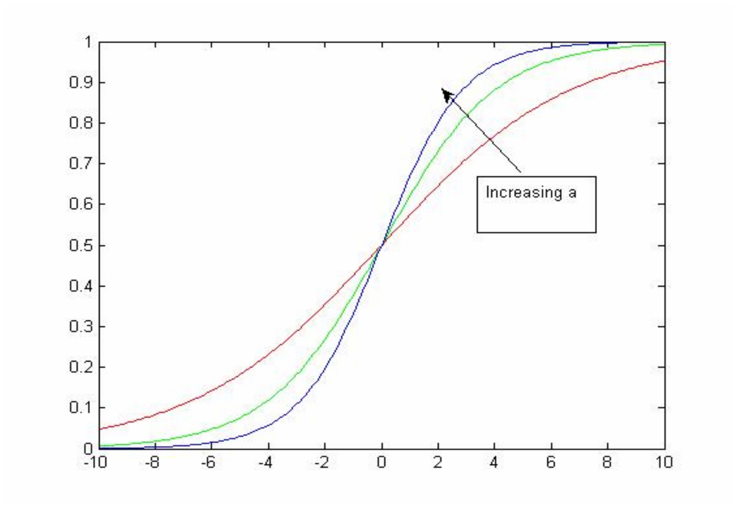


Figure 12: Log sigmoid function.

5. Output (Y_i): The neuron calculates the output [Scho07] according to the following formula

$$\begin{aligned} y &= f\left(\sum_{i=1}^{n} w_i x_i - u\right) \\ &= f\left(\sum_{i=0}^{n} w_i x_i\right) \end{aligned} \tag{3.3-6}$$

For the example elaborated in the Figure 9 the output can be written as

$$y = f\left(w_0 x_0 + w_1 x_1 + w_2 x_2 + w_3 x_3\right) \qquad \textbf{(3.3-7)}$$

The final output value depends upon the transfer function and can be discrete or continuous.

3.4 Neural Network Architectures

In any neural network the neurons are grouped together into layers. Data is fed through the input layer, processed by one or several hidden layers and finally the network provides the output through the output layer.

- Feed Forward Network:

  In this kind of the Networks there is no feedback in the system [Simo98]. The input after processing through the hidden layers is directly given out as output.

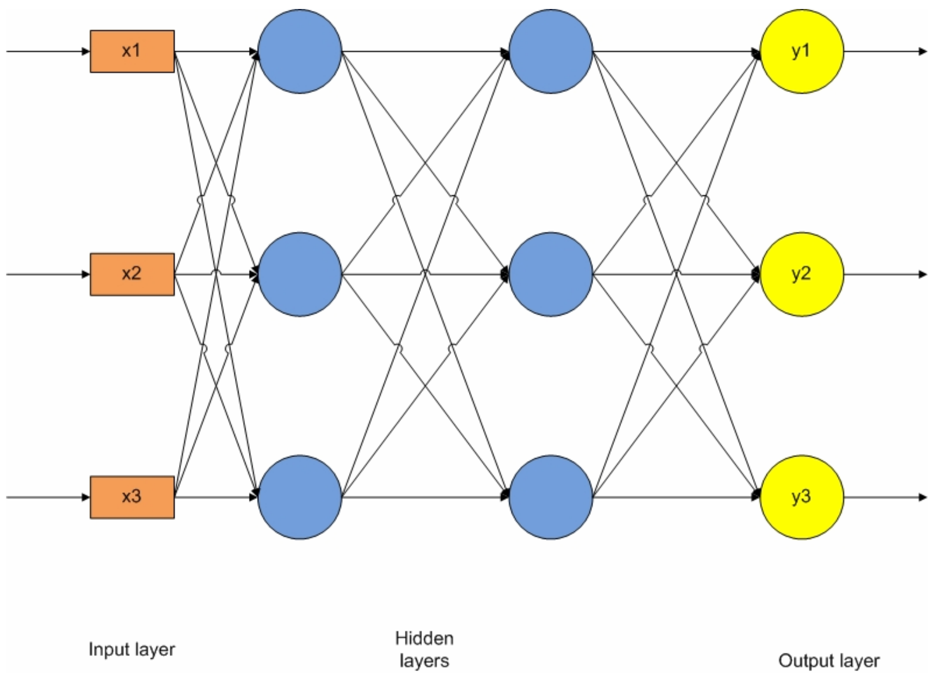


Figure 13: Multilayer feed forward architecture.

- Recurrent neural network architecture:

  Feedback occurs in almost every part of the human brain. Thus a recurrent neural network was introduced and can be defined as a network that has feedback to all the inputs [Simo98]. The feedback can be to all input neurons including the neuron's own input.

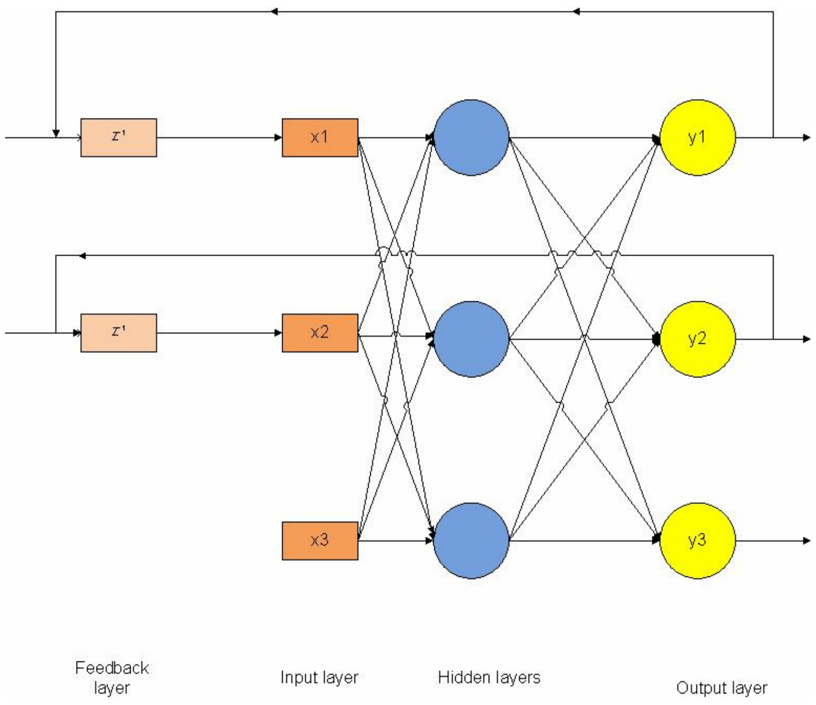


Figure 14: Recurrent neural network architecture.

3.5 Hopfield neural network

Hopfield Neural Networks (HNN) was introduced by J.J. Hopfield [Hopf82]. The introduction of HNN reignited the interest in the field of neural networks. The neural network research was making a slow progress after it began with great enthusiasm, due to various shortcomings of the already existing neural network models. HNN can serve as a content-addressable memory (CAM). A CAM is able to retrieve the complete memory based on sufficient partial information provided to the network [Wiki07]. HNN was also used to solve optimization problems like traveling salesman problem [Hopf85].

The unit of HNN has a single layer of processing units with binary threshold limits. This means that the output of HNN unit is either 0 or 1 depending whether the input is less or greater than the threshold value. In some cases the output can also take values -1 or 1. Mathematically it can be illustrated as [Wiki07]

$$\begin{aligned} a_i &= 1, \quad \text{if} \sum_{\mathrm{j}} \mathrm{w}_{\mathrm{ij}} s_j \geq u_i \\ &= 0, \quad \text{otherwise} \\ a_i &= 1, \quad \text{if} \sum_{\mathrm{j}} \mathrm{w}_{\mathrm{ij}} s_j \geq u_i \\ &= -1, \quad \text{otherwise} \end{aligned} \qquad \textbf{(3.5-1)}$$

where, w_{ij} is weight of connection unit i and j, s_j is the state of unit j and

u_i is the threshold of the unit.

These activity values are called, states. At each moment the network has a specific state, and a state can be represented by a vector of zeros and ones, such as [Simo98]

$$U = (u_1, u_2, \ldots. u_n) \qquad \textbf{(3.5-2)}$$

In the following figure the state vectors is

$$U = (1, 0, 0, 1) \qquad \textbf{(3.5-3)}$$

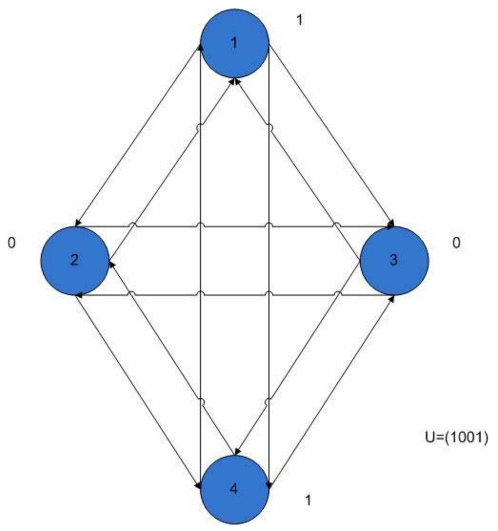


Figure 15: State of Hopfield neural network.

The state changes as the value of each unit changes with time. The HNN are recursive type of networks. The processing units in the HNN are fully connected, that is every unit is connected to every other unit and output of each unit has a feedback into input of each unit. This recursive arrangement helps HNN to be in a stable state if no external input is applied.

The connections in the HNN have the following restrictions [Wiki07]:

$$w_{ii} = 0, \forall i \quad \textbf{(3.5-4)}$$

In this case none of the units in HNN has connection to itself, and

$$w_{ij} = w_{ji} \ , \forall i, j \quad \textbf{(3.5-5)}$$

In the above case all connections are symmetric.

The Hopfield Neural network architecture [Simo98] is as shown below

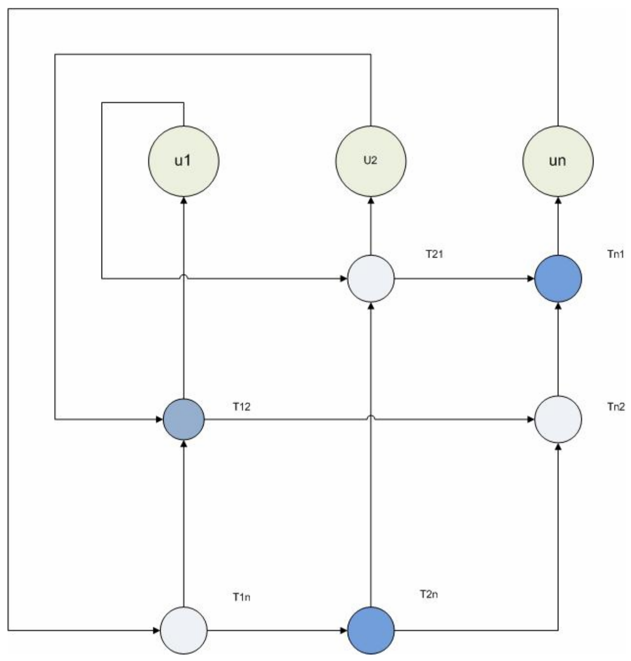


Figure 16: Hopfield neural network architecture.

Each network state in HNN has the energy associated with it given by [Wiki07]

$$E = -1/2 \sum_{j} \sum_{i(i \neq j)} W_{ij} u_j u_i \qquad \textbf{(3.5-6)}$$

This energy function is not any physical energy of the system but can be used as an objective function which is minimized by the network while updating its states. The successive updating of the state in the network is a convergence process and as a result of this, the energy of the system gets minimized [Simo98]. The energy is either reduced or unchanged during the updating process. Even if the units are chosen randomly to update their states; the network will converge to a local minima or global minima in the above given energy function. The convergence conditions can be linked to a Lyapunov function [Wiki07]. Once the system reaches the local minima it is not possible for the system to reach the global minima unless the process of updating with different initial values is started again. If the weights are symmetric the energy function will decrease monotonically and if non-symmetric weights are used in the network, the network may exhibit some periodic or chaotic behaviors [Wiki07]. This chaotic behavior was seen in a relative small phase space and did not hamper the performance of the network ability [Hopf85]. The HNN is used to solve many combinatorial problems but, the solutions obtained may or may not be the optimum based on the value of the energy function. If the energy function reaches the local minimum, rather than the global minimum of the system, the solution cannot be considered as an optimum. To enhance the search ability of the network chaotic neural networks (CNN) have been proposed [Toku97]. Tokuda observed that CNN provides an excellent search ability to find the optimal solution to the combinatorial problems. In the current work the hysteresis neural network is studied. In related research the hysteresis phenomenon was used to solve N-queen problem [Naka99]. This approach is analyzed and used in this thesis.

3.6 Hysteresis Neuron Model

In the hysteresis neuron the activation function is a hysteretic function. This neuron is similar to a conventional neuron as it calculates the weighted sum of the inputs and weights. It differs from the conventional Neuron model by having a nonlinear hysteresis activation function. The following figure illustrates the basic neuron model.

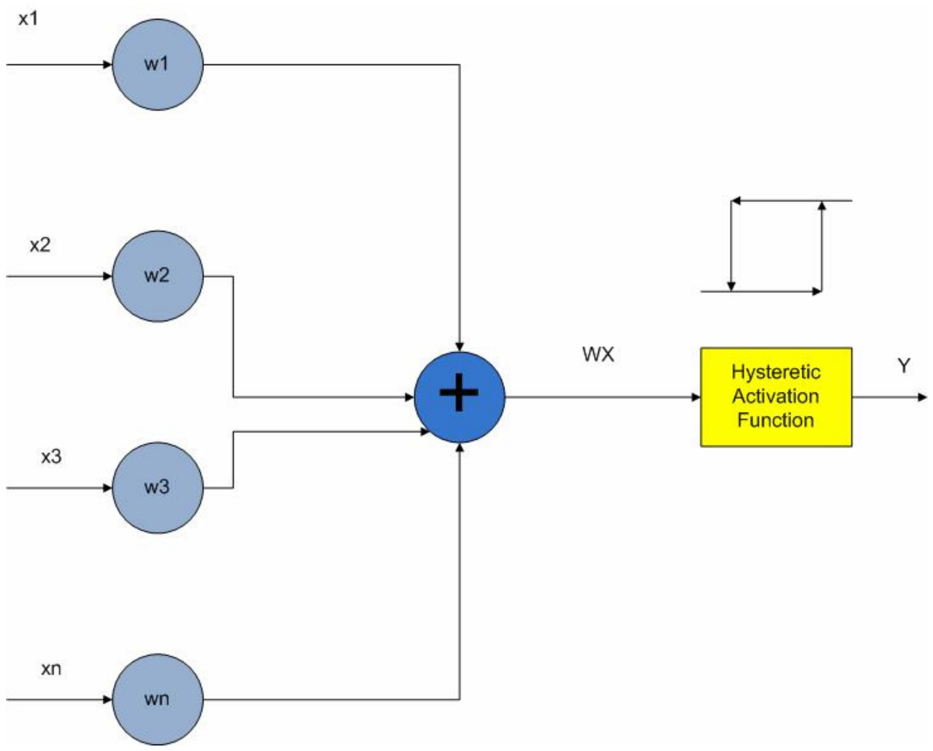


Figure 17: General hysteresis neuron.

The weighted sum of the input functionality can be implemented using the integrator and a summing amplifier using the opamp. The hysteresis activation function can be implemented using the Schmitt trigger.

3.6.1 Hysteresis Neural Network System

Nakaguchi [Naka99], in his work described the implementation of hysteresis neural network system, shown in Figure 18.

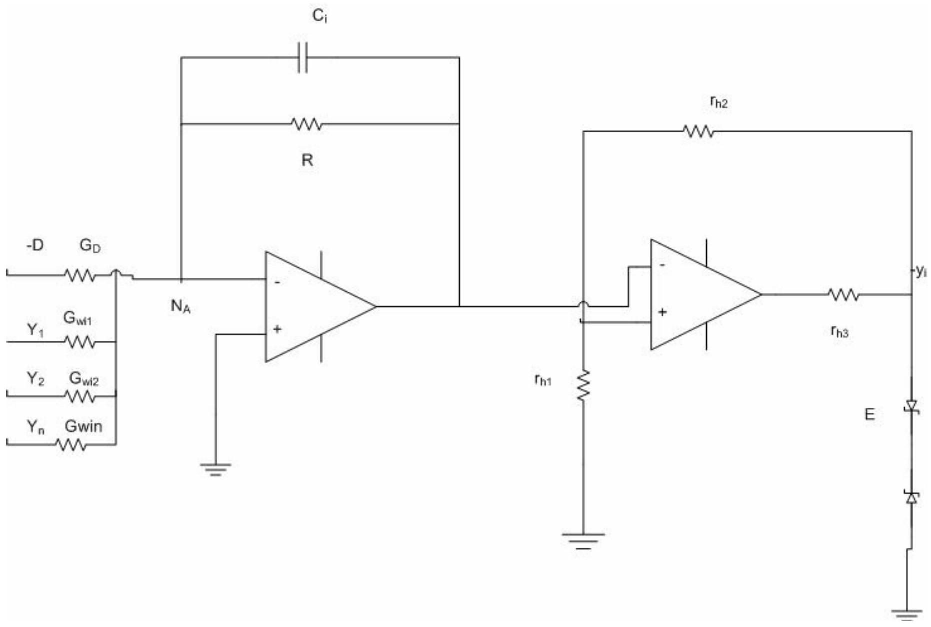


Figure 18: Hysteresis neural network system.

The complete analysis of the above circuit was carried out by Nakaguchi [Naka99]; some of the important equations are used in this section in order to understand the hysteresis neural network. Following is the simplified equation which represents the above circuit,

$$\lambda_i \frac{d}{dt} x_i = -x_i - \sum_{j=1}^{N} w_{ij} y_j + u_i$$

$$y_i = h(x_i) \qquad \textbf{[Naka99]} \quad \textbf{(3.6-1)}$$

$$= 1 \ \text{ if } \ x_i \geq 0$$

$$= 0 \ \text{ if } x_i \leq 1$$

where ,

$t \in R$, is time,

$x_i \in$ R, is the internal state of the neuron.

$y_i \in \{1,0\}$ is the output of the neuron

$\lambda_i \rangle\ 0$ is a time constant

$w_{ij} \in$ R, is the weight of the output between

$h(x_i)$ is a hysteresis function.

3.6.2 Hysteresis Activation Function

$h(x_i)$ switches from 0 to 1 if x_i reaches the right threshold 1 and switched from 1 to 0 if x_i reaches the left threshold 0. The region between $0<x_i<1$ is called hysteresis region [Naka99]. This is shown in Figure.19.

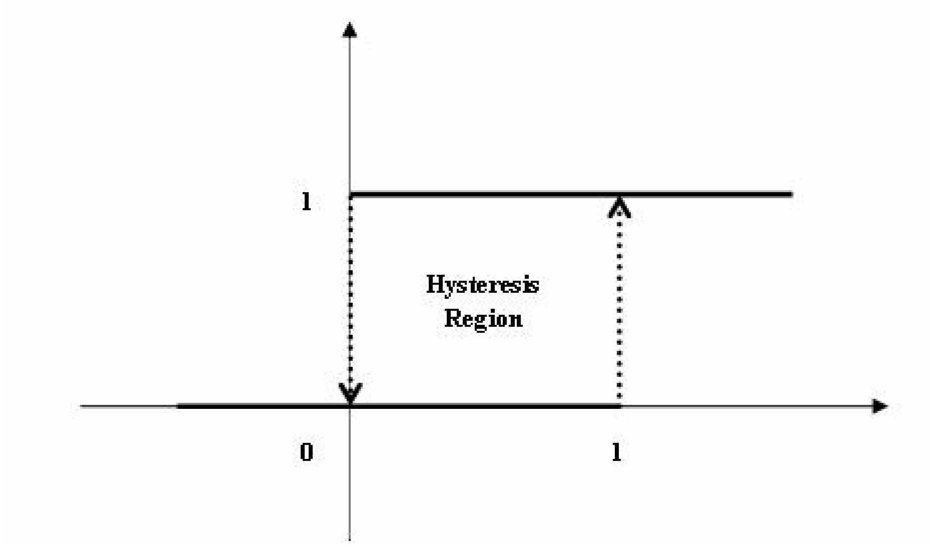


Figure 19: Hysteresis function.

3.6.3 Equilibrium Points

The equilibrium point in this network is be given by Equation 3.6-2

$$p_i = -\sum_{j=1}^{N} w_{ij} y_j - u_i \qquad \textbf{[Naka99] (3.6-2)}$$

where, p_i is the equilibrium point of the ith neuron

u_i is the input value to the ith neuron.

The equilibrium points are dependent on the output of the system and remain constant in the system as long as the outputs are unchanged. There are two types of equilibrium points depending upon the output of the system.

p_i is said to be a real equilibrium point if following condition is satisfied.

$$(2p_i - 1)(2y_i - 1) > -1$$ **[Naka99] (3.6-3)**

The cell (neuron) corresponding to this point is said to be stable.

p_i is said to be a imaginary equilibrium point if following condition is satisfied.

$$(2p_i - 1)(2y_i - 1) \leq -1$$ **[Naka99] (3.6-4)**

The cell (neuron) corresponding to this point is said to be unstable.

Hence, if $p_i \leq 0$ the cell has to cross zero, the left threshold of the hysteresis region and the output of the cell changes from $y_i = 1$ to $y_i = 0$

$$\therefore \; p_i \leq 0 \text{ For } y_i = 1, \; y_i \text{ changes from } 1 \to 0$$

Similarly if $p_i \geq 1$ the cell has to cross the right threshold of the hysteresis region and the output of the cell changes from $y_i = 0$ to $y_i = 1$

$$\therefore \; p_i \geq 1 \text{ For } y_i = 0, \; y_i \text{ changes from } 0 \to 1$$

Thus the trajectory of the cell is always directed to the equilibrium point.

3.7 Simulations of Hopfield Neuron Model

The simulations were done using Multisim 8.0, OrCAD 10.3 and Spice3 software.

3.7.1 Simulation using Multisim 8.0:

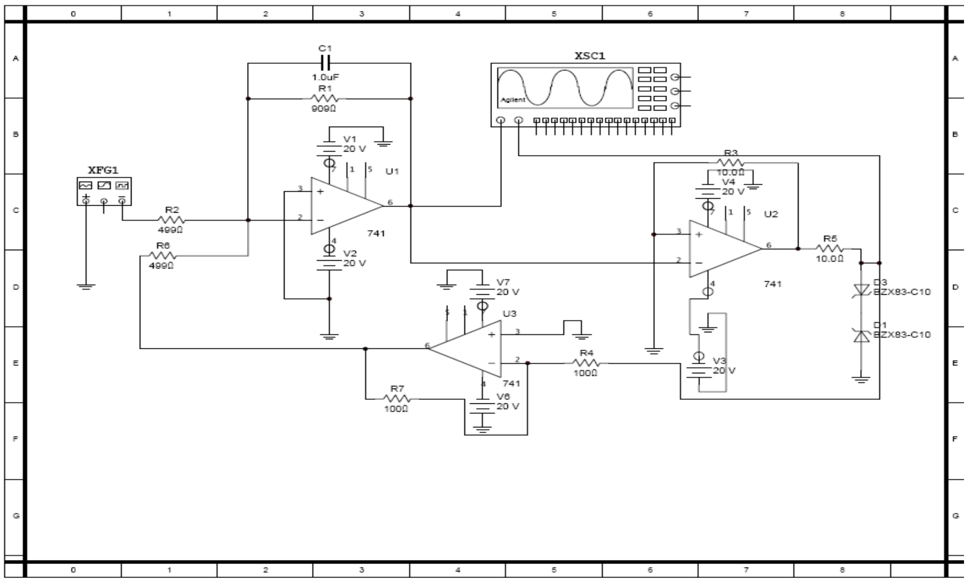


Figure 20: Multisim schematic of hysteresis neuron.

Figure 20 shows the schematic of hysteresis neuron using Multisim, the opamp used in the figure is LM741 and a BZX83_C10 Zener diode is used. These models (Appendix 1 gives the details for these models) are available in the Multisim library. The above circuit is based on the neural network circuit discussed in Section 3.6.1 [Naka99]. Multisim is very useful for this research since it contains the Simulated Agilent Oscilloscope (SAO). The SAO helped to work in a simulated laboratory environment without actually performing the experiments in a laboratory.

For certain values of the inputs, the hysteresis neuron showed the chaotic phenomenon and the following simulation results that were obtained for various inputs confirm this.

A square waveform with the frequency of 1000 Hz and duty cycle 50% was used in the simulation. Only the amplitude of the square waveform was changed to observe different chaos and hysteresis characteristics.

Initially the amplitude of the square wave was 70 mV and the following results was observed

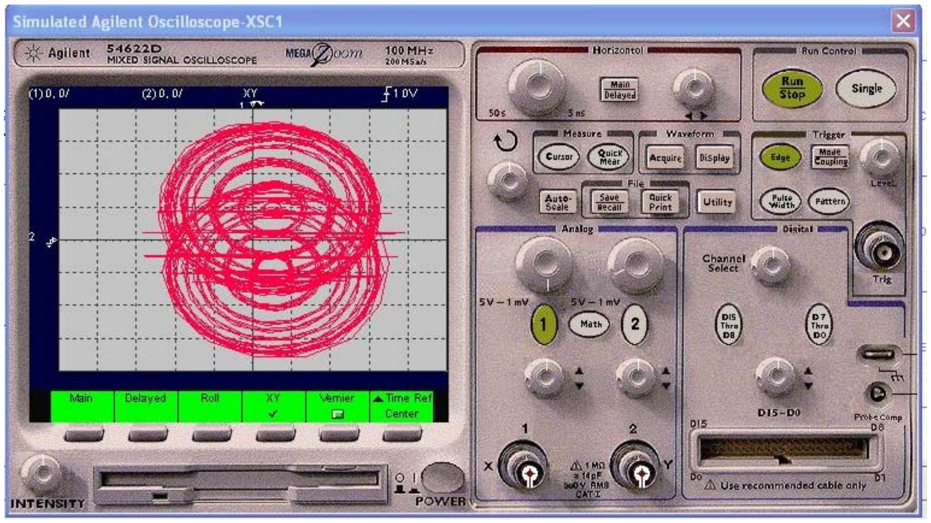


Figure 21: Double circular chaos phenomenon using HNN.

The simulation was run again with the same configuration but increasing the amplitude to 100 mV. The simulated output is as shown in the Figure 22, giving well defined double circular chaos.

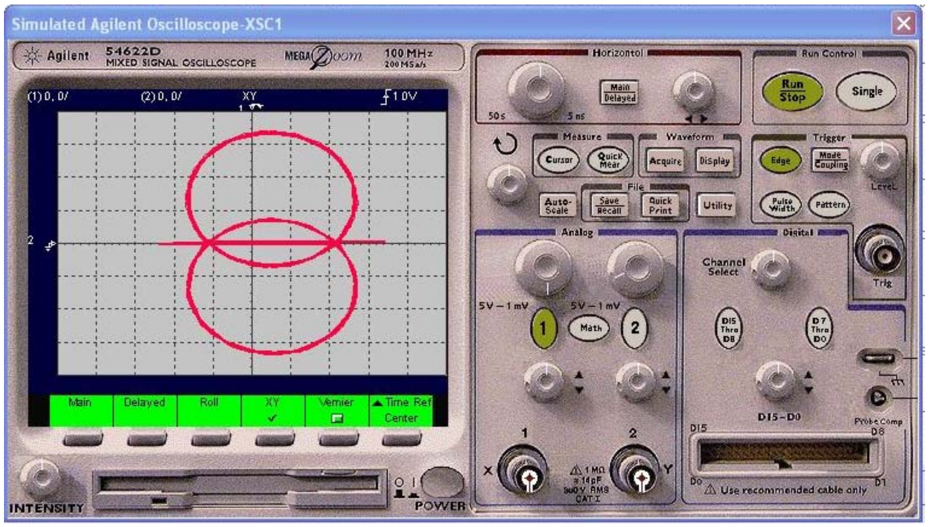


Figure 22: Double well defined circular chaos using HNN.

If the amplitude is further increased the same circuit shows double attractors as shown in the following figure.

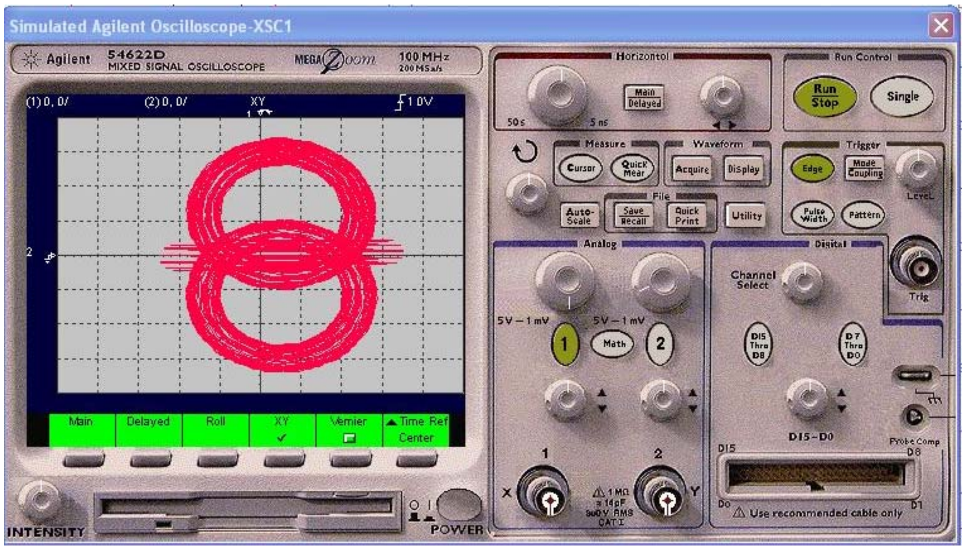


Figure 23: Doubly attractors using HNN.

Increasing the amplitude to 700 mV, the hysteretic characteristics can be obtained with the same circuit as shown in Figure 24.

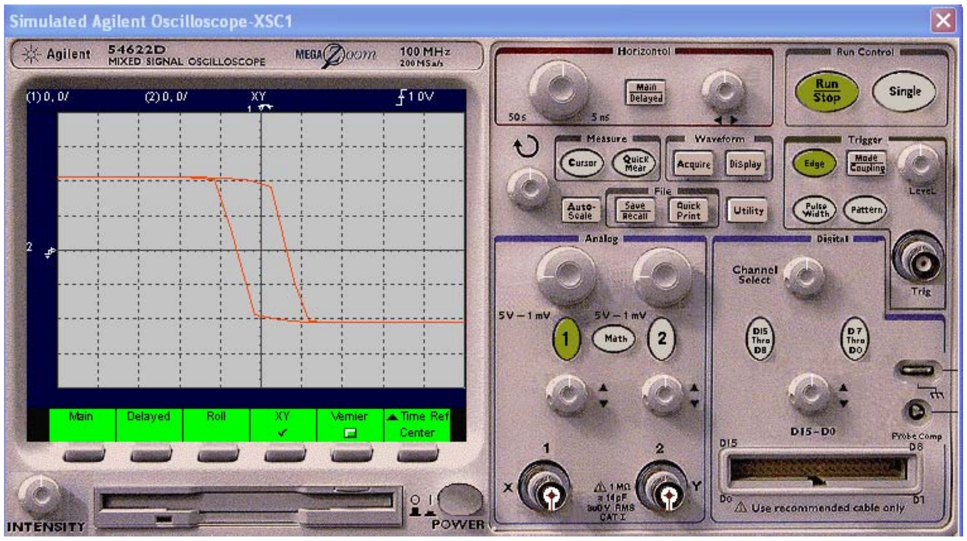


Figure 24: Hysteresis characteristics using HNN.

From the above discussion, the hysteresis neural network can be considered as a chaotic circuit which can go to the stable state with proper initial conditions and hence a solution can be obtained.

3.7.2 Simulation using OrCAD 10.3

The next step for building the hysteresis neural network circuit using double gate MOSFET is to build a Spice netlist for the circuit and to be able to generate the same kind of simulation waveforms as discussed in Section 3.7.1 using OrCAD. The approach taken in this thesis to move design obtained in OrCAD to Spice3 simulator. Spice3 simulator is used because the double gate model [UFDG06] is available for simulation only in Spice3 simulator. OrCAD is a good tool for schematic entry for the prototype design and has an extensive component library, as it was necessary to have a nonlinear opamp 741 model. On the other hand Spice3 uses a command line interface and models of devices and components need to be built or acquire from CAD vendors. Thus it was convenient to test the design using the symbols provided in OrCAD libraries rather than writing the netlist for each Opamp 741 Model in Spice3.

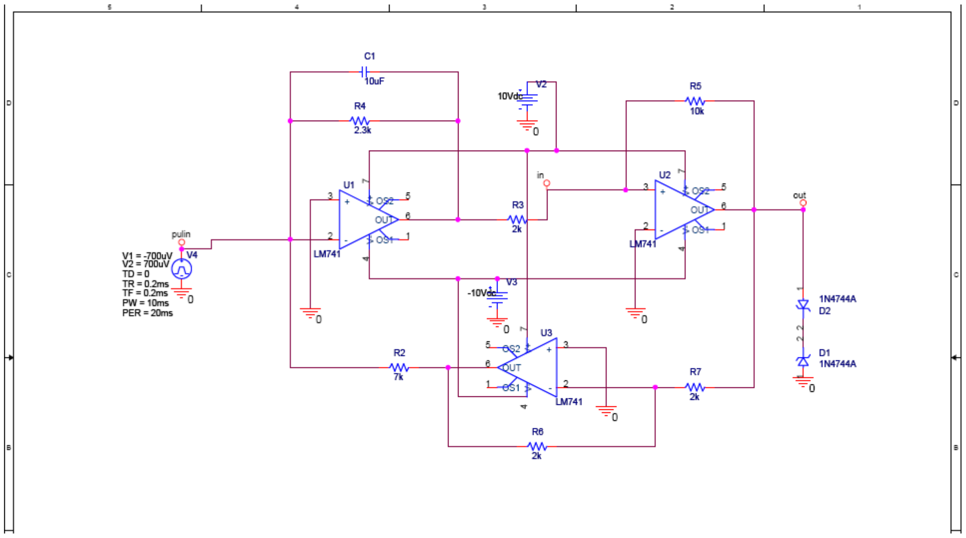


Figure 25: Hysteresis neuron circuit with OrCAD 10.3.

The above figure shows the schematic of hysteresis neuron using OrCAD 10.3, based on the discussion in the Section 3.6.1 [Naka99]. The opamps used in the figure are LM741 and the Zener diode is IN4744A (Appendix 2 gives the details for these models). Both models were present in the libraries of OrCAD. The simulations were run for different values of the initial conditions and signal frequencies.

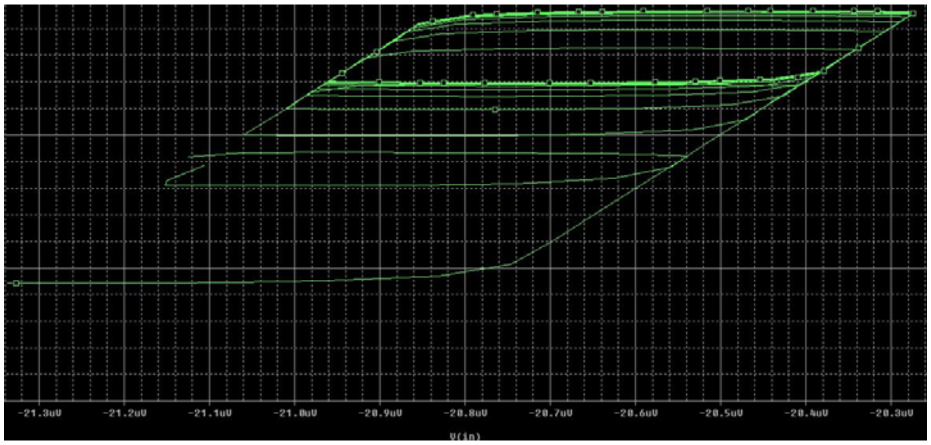


Figure 26: OrCAD simulation for chaos.

By changing the initial conditions, like increasing the amplitude, we get circular chaos as shown below

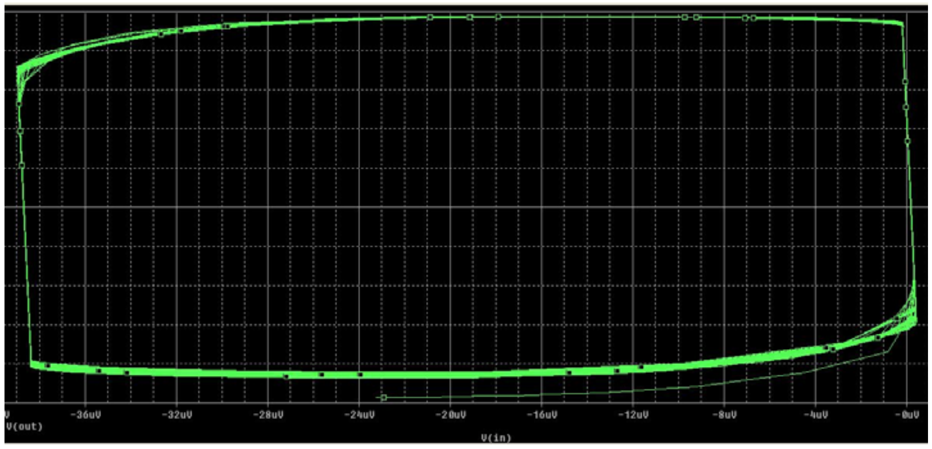


Figure 27: Circular chaos using OrCAD.

Modifying the initial conditions in the same circuit, a hysteresis characteristic is obtained as shown in Figure 28.

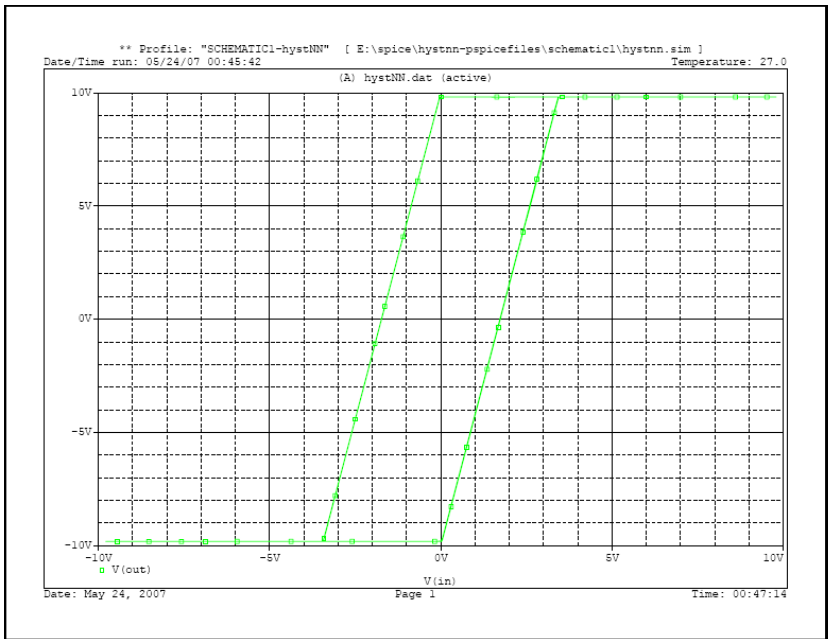


Figure 28: Hysteresis characteristics using OrCAD.

Thus approximately the same characteristics for the hysteresis neuron like those obtained earlier by using Multisim were simulated using OrCAD 10.3 schematic design tool. The next step was to generate a netlist for this circuit and simulate it using Spice3 simulator. Here the double gate model [UFDG] was available in Spice3 was used in simulation.

3.7.3 Simulation using Spice3

In order to implement the hysteresis neuron using double gate MOSFET, the following Spice3 netlist was generated.

```
VIN 4 0 PULSE (0 0.1MV 0 0 0 2MS)
dD1 3 11 1N4744A__ZENER__1
dD2 0 11 1N4744A__ZENER__1
rR2 4 5 1.500e+003
rR1 5 8 9.090e+002
rR7 10 13 1.00e+002
rR4 13 3 1.00e+002
rR3 0 9 10
rR5 9 3 10
cC1 5 8 1.0E-6
VV7 12 0 dc 15 ac 0 0
xU3 10 13 0 U3_OPEN_VS- 12 OPAMP1
rR6 10 5 4.990e+002
VV6 0 V6_OPEN_2 dc 15 ac 0 0
VV4 7 0 dc 15 ac 0 0
```

```
VV3 0 6 dc 15 ac 0 0

xU2 9 8 0 6 7 OPAMP1

VV2 0 2 dc 15 ac 0 0

VV1 1 0 dc 15 ac 0 0

xU1 8 5 0 2 1 OPAMP1
```

The model information used in this netlist of opamp 741 and 1N4744A Zener diode and the methodology to run the program is detailed in the Appendix 3. This netlist was run on Solaris 10 operating system and results were plotted and captured using the screen shots.

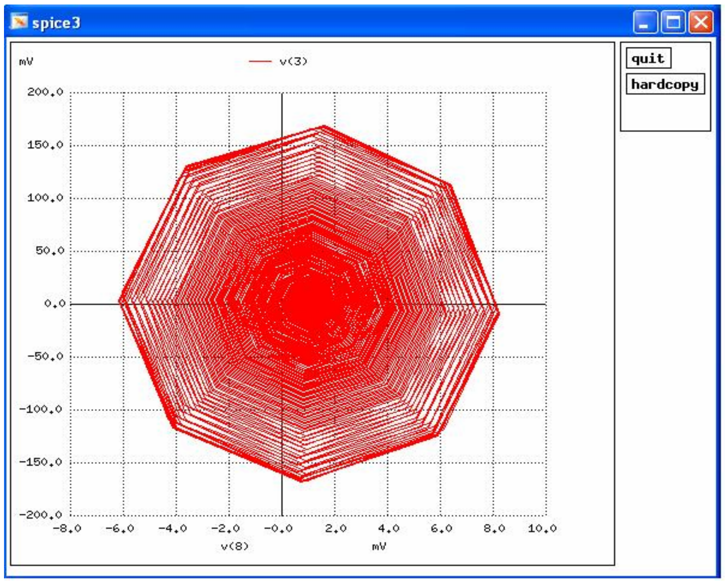


Figure 29: Chaotic attractor from hysteresis neuron using Spice3.

In the above figure the chaotic attractor has an octagonal shape.

Changing the amplitude of the input wave, yields double circular chaotic attractors shown in Figure 30.

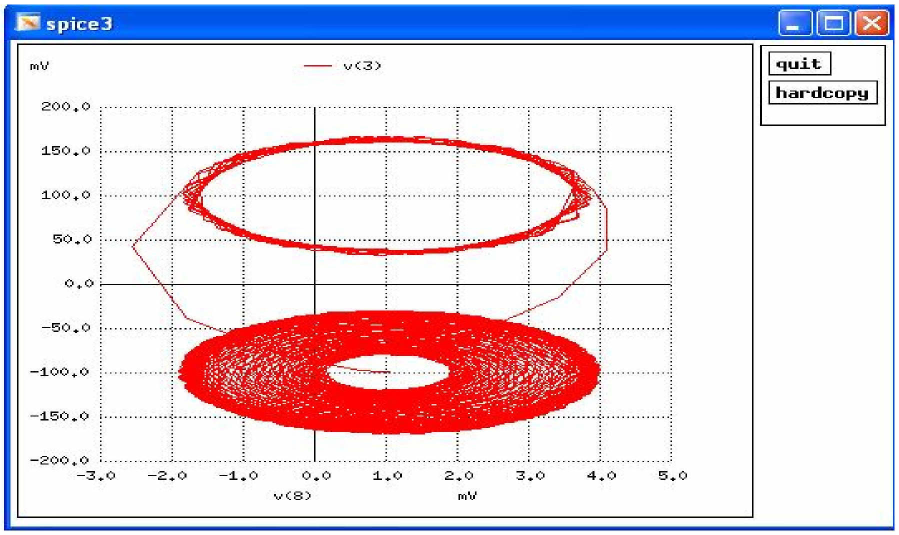


Figure 30: Double circular chaotic attractors using Spice3.

The following figure illustrates the Hysteretic phenomenon demonstrated with the same circuit but with different set of the initial conditions.

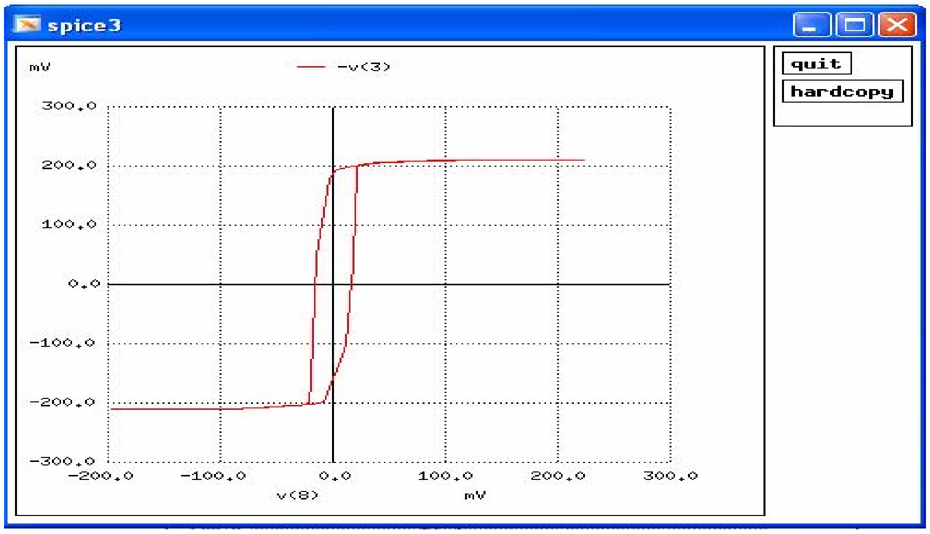


Figure 31: Hysteresis characteristics using Spice3.

The next step is to use the same netlist and to produce hysteretic characteristics using the double gate MOSFET. This simulation will be discussed in Chapter 5 after the description of N-queens problem.

4. N-queens Problem

4.1 Introduction

The 8-queen puzzle was proposed by a German chess player Max Bazzel [Wiki07] in 1848. Later this puzzle was studied extensively by mathematicians and was extended to N-queens problem. This is one of the constraint satisfaction problems and hence it is NP-complete. It is difficult for all classical search algorithms to find solution for the N-queen problem in an acceptable time. Various approaches [Sosi90], [Fink87] has been used to solve the N-queen problem efficiently. Since the problem has a simple and regular structure this can be used in finding different strategies using neural networks.

4.2 System Stability for Hysteresis Neural Network:

In the Section 3.6.3 the stability of single hysteresis neuron was introduced. The output of the system doesn't change when all the cells in the network have a real equilibrium point and the system is believed to reach a stable state.

Extending the conditions for the real and imaginary points in the Equations 3.6.4 and 3.6.5 to the complete system, we obtain the following conditions:

$$if \qquad (2p_i - 1)(2y_i - 1) > -1 \qquad (\forall i)$$

[Naka99] (4.2-1)

the system is said to be stable, and

$$if \qquad (2p_i - 1)(2y_i - 1) < -1 \qquad (\exists i)$$

[Naka99] (4.2-2)

the system is considered to be unstable.

One can infer that

- If $p_i > 1$ $(\forall i)$

  only $y_i = 1$ have real equilibrium points and $y_i = 0$ have imaginary equilibrium points. These imaginary equilibrium points have the possibilities to change their output from $y_i = 0$ to $y_i = 1$

- If $p_i < 0$ $(\forall i)$

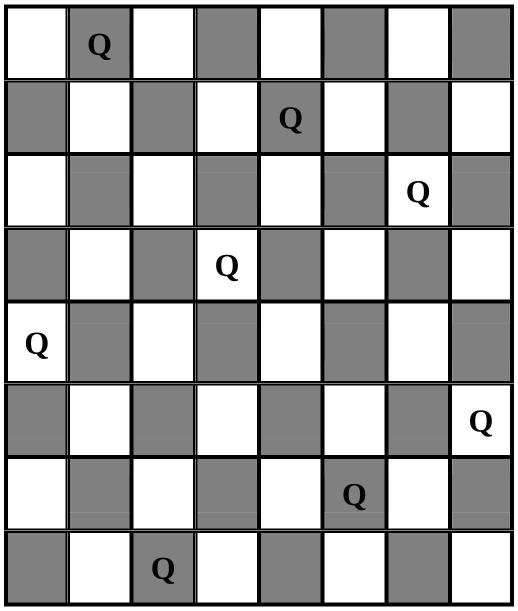
  only $y_i = 0$ have real equilibrium points and $y_i = 1$ have imaginary equilibrium points. These imaginary equilibrium points have the possibilities to change their output from $y_i = 1$ to $y_i = 0$

From the above discussion, it can be concluded that it is only the imaginary equilibrium points which change their states. These cells try to change their output in the hysteresis region which is in the range [0, 1]. When these imaginary equilibrium points locate the hysteresis region they become stable [Naka99]. Once all these points become stable, the system achieves the stability, which corresponds to the solution of the problem for which the network is designed.

4.3 Problem Statement and Solution

Place N (N>3) queens, on the N x N board such that no two queens attack each other. In 8-queen problem it is necessary to place 8 queens in such a way that no two queens are able to attack each other.

One of the 8 queen problem’s solutions is as shown in Figure 32



a)

0	1	0	0	0	0	0	0
0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	1
0	0	0	0	0	1	0	0
0	0	1	0	0	0	0	0

b)

Figure 32: One of the solutions for 8 queen problem.

In Figure 32 (b), 1 corresponds to the place where a queen is placed while a 0 corresponds to the place where no queen is placed. In the N-queen problem no two queens can occupy the same row, column and diagonal, transforming these conditions into various constraints.

The sum of the outputs on the rows is 1 [Naka99].

$$\sum_{m=1}^{N} y_{im} = 1 \qquad (\forall i, j) \qquad \textbf{[Naka99] (4.3-1)}$$

The sum of outputs on the columns is 1 [Naka99]

$$\sum_{m=1}^{N} y_{mj} = 1 \qquad (\forall i, j) \qquad \textbf{[Naka99] (4.3-2)}$$

According to Nakaguchi [Naka99] the sum of the outputs for the diagonal can be either 1 or 0, as a queen can be in the diagonal or may not be present.

$$0 \le \sum_{1\le i\text{-}k,\, j\text{-}k \le N} y_{i\text{-}k,\, j\text{-}k} \le 1 \qquad (\forall i, j)$$

[Naka99] (4.3-3)

$$0 \le \sum_{1\le i\text{-}k,\, j+k \le N} y_{i\text{-}k,\, j+k} \le 1 \qquad (\forall i, j)$$

The above equations are the constraints for solving the N-queen problem. There can be many different solutions for the same size of chess board depending upon the initial conditions. The stable system is the one that has stable equilibrium point and these constraints can be used to obtain the equilibrium for the hysteresis neural network. Substituting the values from Equation 4.3-1, 4.3-2 and 4.3-3 into the Equation 3.6.3 for the equilibrium point becomes Equation 4.3-4 [Naka99]

$$p_{ij} = \left[\left(1-\sum_{m=1}^{N} y_{im}\right)+\left(1-\sum_{m=1}^{N} y_{mj}\right)+\alpha_{ij}\left(1-\sum_{1\le i-j,\, j-k\le N} y_{i-j,\, j-k}-\sum_{1\le i-j,\, j+k\le N} y_{i-j.\, j+k}\right)+u_{ij}\right] \qquad \textbf{(4.3-4)}$$

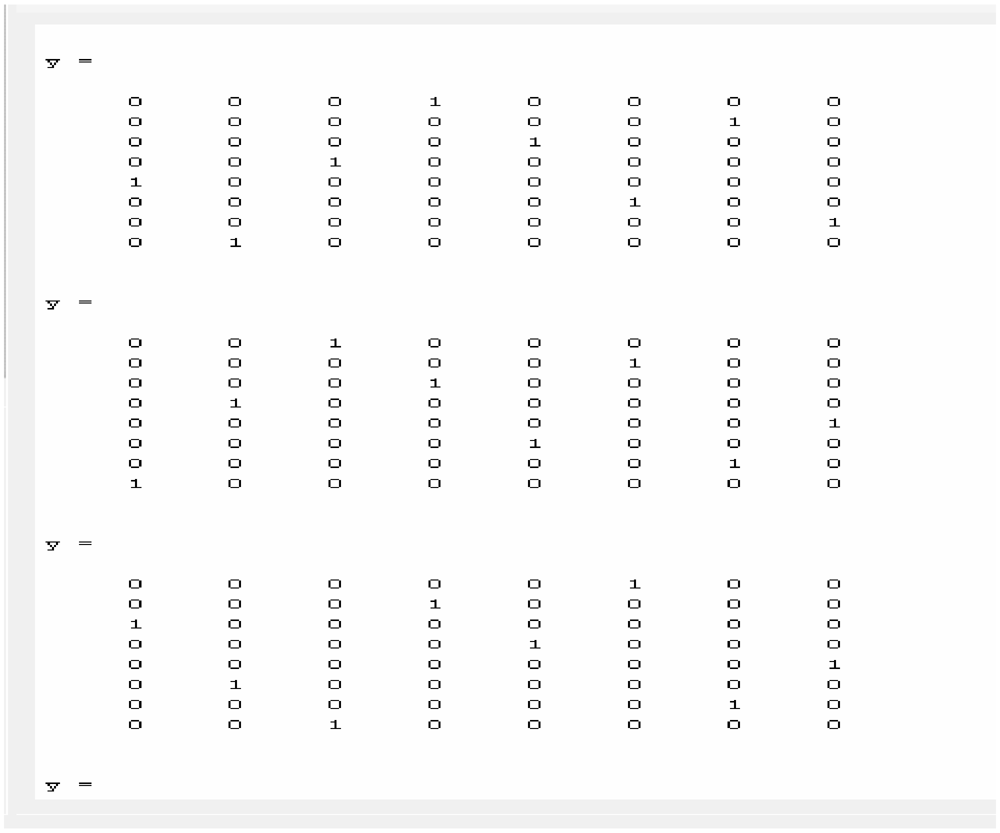
Substituting the above value of equilibrium points into Equation 3.6.2 gives [Naka99]

$$\lambda_{ij}\dot{x}_{ij} = -x_{ij} + \left[\left(1-\sum_{m=1}^{N} y_{im}\right)+\left(1-\sum_{m=1}^{N} y_{mj}\right)+\alpha_{ij}\left(1-\sum_{1\le i-j,\, j-k\le N} y_{i-j,\, j-k}-\sum_{1\le i-j,\, j+k\le N} y_{i-j.\, j+k}\right)+u_{ij}\right]$$

(4.3-5)

The solution to the N-queen problem can be obtained by having a stable system. The stable system can be obtained when all the equilibrium points are in the hysteresis region. The equilibrium point has the value inside the hysteresis region once all the constraints are satisfied. Using these constraints a Matlab code was written and the simulation result for 8 queens' problem is shown in Figure 33 (refer Appendix 4 for Matlab code).

If all the equilibrium points of the system are in the hysteresis region the system is said to be stable and this corresponds to the solution for the N-queen problem.



```
y =

     0     0     0     1     0     0     0     0
     0     0     0     0     0     0     1     0
     0     0     0     0     1     0     0     0
     0     0     1     0     0     0     0     0
     1     0     0     0     0     0     0     0
     0     0     0     0     0     1     0     0
     0     0     0     0     0     0     0     1
     0     1     0     0     0     0     0     0


y =

     0     0     1     0     0     0     0     0
     0     0     0     0     0     1     0     0
     0     0     0     1     0     0     0     0
     0     1     0     0     0     0     0     0
     0     0     0     0     0     0     0     1
     0     0     0     0     1     0     0     0
     0     0     0     0     0     0     1     0
     1     0     0     0     0     0     0     0


y =

     0     0     0     0     0     1     0     0
     0     0     0     1     0     0     0     0
     1     0     0     0     0     0     0     0
     0     0     0     0     1     0     0     0
     0     0     0     0     0     0     0     1
     0     1     0     0     0     0     0     0
     0     0     0     0     0     0     1     0
     0     0     1     0     0     0     0     0


y =
```

Figure 33: Matlab simulation results for 8x8 queen problem.

There are 92 solutions for the 8-queen problem which depends on the initial condition. Table 2 [Naka99] gives the number of solutions for various sizes of chess board and the number of iterations required to obtain these solutions.

Table 2: Number of Solutions for Different Size of Queen Problem

Table N	# of Solutions	N^2C_N
4	2	$1.83x10^3$
5	10	$5.31x10^4$
6	4	$1.94x10^6$
7	40	$8.59X10^7$
8	92	$4.42X10^9$
9	352	$2.60X10^{11}$
10	724	$1.73X10^{13}$
11	2680	$1.27x10^{15}$
12	14200	$1.04x10^{17}$
13	73712	$9.18x10^{18}$
14	365596	$8.81x10^{20}$

The N-queen system has a large set of differential equations and when executed in the software each one of them is executed sequentially. Due to the inherently sequential nature of computer, as the number of queens to be placed increases, the time taken to execute all the equations increases exponentially. The differential equations can be implemented using the analog circuit. Each cell (neuron) in the system can switch its state in parallel, independently of each other. Thus in a resulting parallel network, the execution time is independent of the number of queens to be placed, considering that such a network is realizable physically, meaning implementation using hardware.

4.4 Network Topology

The N-queen problem requires only one queen to be placed in a single row, column or a diagonal. As a result each neuron is connected to all the neurons whose output influences the output of the neuron and feedback to itself. Consider the following figure. The simulated circuit topology is shown for two cases (neuron number 6: yellow and 13: green). This can be easily extended to all neurons.

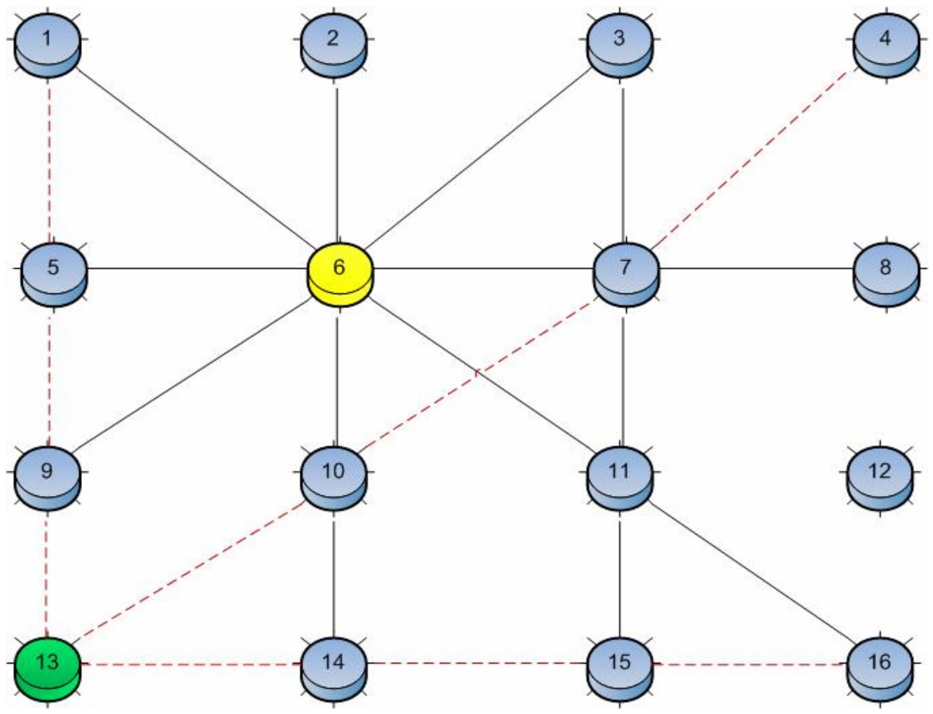


Figure 34: Connection topology for 8-queen network.

Consider the queen at location (6). This neuron (representing the queen) is connected to all neurons from row 2 (neuron numbers 5, 7, 8) and column 2 (neuron numbers 2, 10, 14) and all the neurons along the forward diagonal (neuron number 1, 11, 16) and backward diagonal (3, 9) in addition it has its own feedback. The connections are shown in the black line. All neurons surrounding it affect the output of this neuron.

Neuron representing the queen at location (13), is connected to all neurons from row 4 (neuron numbers 14, 15, 16) and column 1 (neuron numbers 1, 5, 9) and all the neurons along the forward diagonal (neuron number 10, 7, 4) and in addition it has its own feedback. The connections are shown with the red dashed line. The neuron (number 13) is at the corner of the 8x8 board; hence it doesn’t consider any connection with the backward diagonal neurons. Only the neurons in forward diagonal affect the output.

5. Double Gate MOSFET implementation of HNN

5.1 Introduction

Coming decade will have a plethora of challenges for the silicon industry which successfully provided us with many system improvements over the last few decades.

Gordon More predicted that the number of transistors on a chip would double every two years [More65]. For more than 40 years the silicon industry proved this prediction, which is now commonly known as "Moore's law".

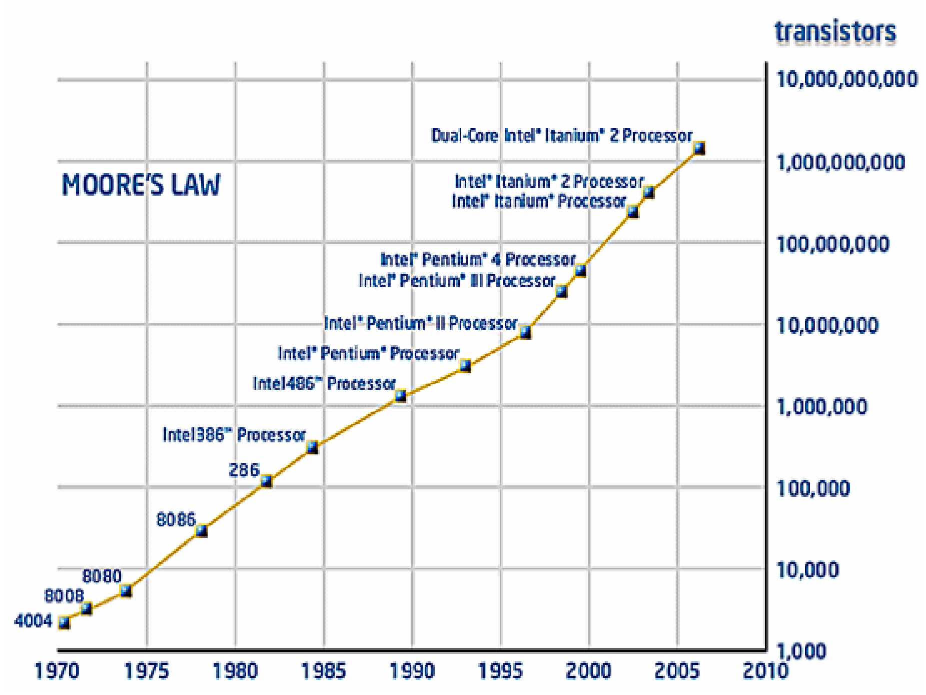
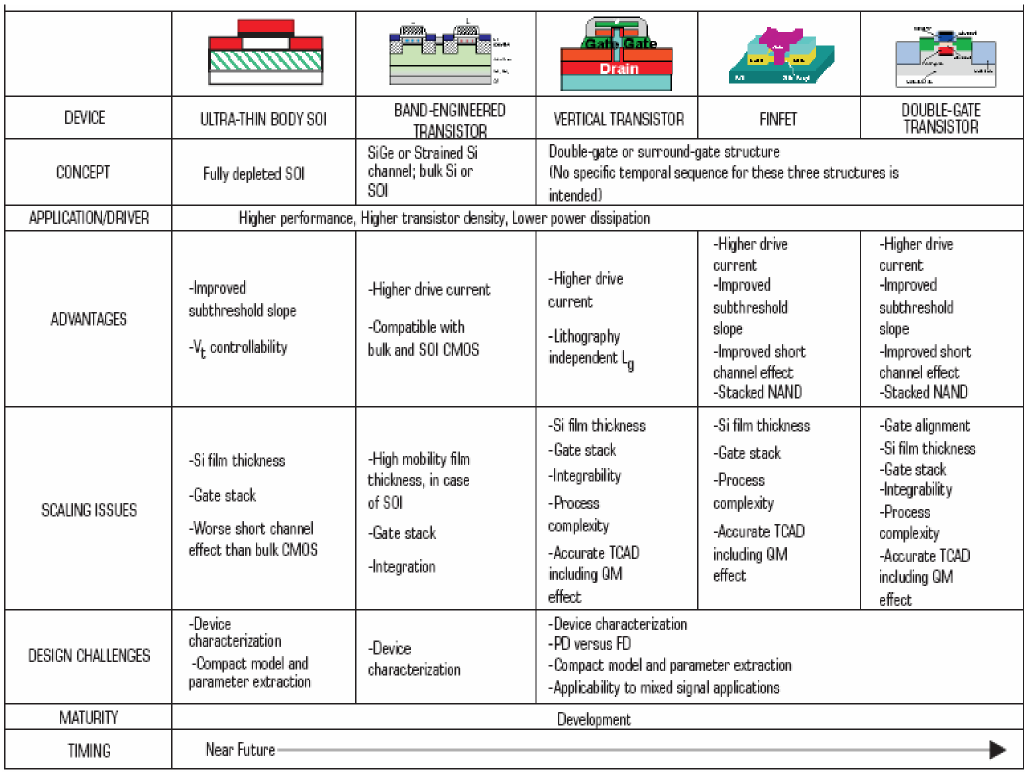


Figure 35: Moore's law graph [Inte07].

The number of transistors per microprocessor chip increased from some thousands in 1970 to nearly a billion in recent times [Inte07]. The industry managed to squeeze more and more transistors by scaling down the size of a single transistor. Reducing the physical dimensions of the transistor increased the system speed and density. According to International Technology Roadmap for Semiconductors [ITRS07] the industry will soon hit a roadblock in coming decade which will hamper the progress of device shrinking. The devices are reaching the physical limits of shrinking; short channel effects and quantum phenomenon will prohibit the smooth scaling of the devices in the near future. To combat such challenges, the silicon industry is using a dual approach; first, it is putting hard efforts to utilize the existing technology to shrink the devices as much as possible and second, it is also considering the feasibility for new devices, which will provide the same or better performance and scalability as the current MOSFET technology. The following figure describes new devices, their advantages, their scaling issues and design challenges as illustrated in the paper, "Extending the road beyond CMOS" [Hutc02].


Gate Gate
Drain

DEVICE	ULTRA-THIN BODY SOI	BAND-ENGINEERED TRANSISTOR	VERTICAL TRANSISTOR	FINFET	DOUBLE-GATE TRANSISTOR
CONCEPT	Fully depleted SOI	SiGe or Strained Si channel; bulk Si or SOI	Double-gate or surround-gate structure (No specific temporal sequence for these three structures is intended)		
APPLICATION/DRIVER	Higher performance, Higher transistor density, Lower power dissipation				
ADVANTAGES	-Improved subthreshold slope -V_t controllability	-Higher drive current -Compatible with bulk and SOI CMOS	-Higher drive current -Lithography independent L_g	-Higher drive current -Improved subthreshold slope -Improved short channel effect -Stacked NAND	-Higher drive current -Improved subthreshold slope -Improved short channel effect -Stacked NAND
SCALING ISSUES	-Si film thickness -Gate stack -Worse short channel effect than bulk CMOS	-High mobility film thickness, in case of SOI -Gate stack -Integration	-Si film thickness -Gate stack -Integrability -Process complexity -Accurate TCAD including QM effect	-Si film thickness -Gate stack -Process complexity -Accurate TCAD including QM effect	-Gate alignment -Si film thickness -Gate stack -Integrability -Process complexity -Accurate TCAD including QM effect
DESIGN CHALLENGES	-Device characterization -Compact model and parameter extraction	-Device characterization	-Device characterization -PD versus FD -Compact model and parameter extraction -Applicability to mixed signal applications		
MATURITY	Development				
TIMING	Near Future →				

Figure 36: Non classical CMOS devices [Hutc02].

5.2 Double Gate MOSFET Structure

Sekigawa proposed the double-gate transistor structure in early 1980s [Seki84]. The Double gate transistor consists of a conducting channel surrounded by gates on both sides. Having two gates ensures that no part of a channel is far away from a gate. This gives better control of the channel by the gate electrode. In addition, the voltage applied to the gate terminal determines the amount of current flowing through the channel.

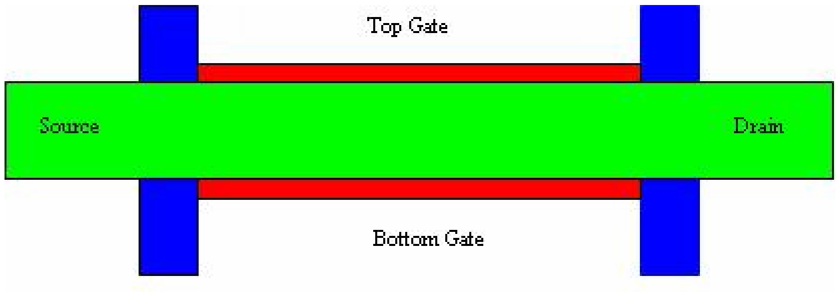


Figure 37: Double gate MOSFET structure.

The salient features of the double gate are [Wong02]

- Better control of short channel effects.
- Reductions of I_{off} ; drain current when gate voltage is zero.
- Higher current drive capabilities.

The double gate can be operated in two modes. One of the common modes is switching both gates simultaneously. This mode is called simultaneously driven double gate (SDDG) MOSFET. Another mode is applying a bias to the second gate (back-gate) to dynamically change the threshold voltages. This mode is called as independently driven double gate (IDDG) MOSFET [Wong02].

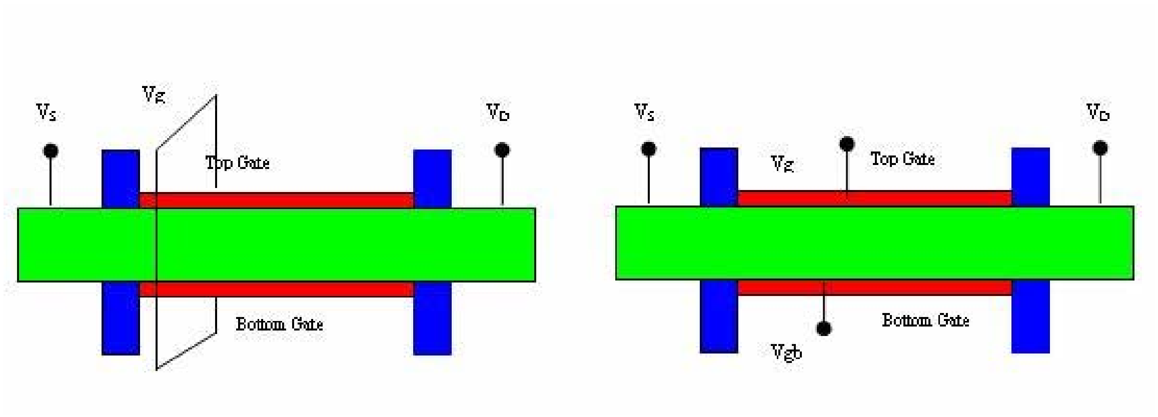


Figure 38: General double gate operation modes.

Kaya proposed the use of double gate MOSFET for implementing the tuneable analog circuit blocks via varying the bottom gate bias. The major importance of this is stated below [Kaya06]

- Increasing the functionality of the device.
- Reducing the area and other parasitic effects.
- High-speed and low power consumption.

Most of the devices implemented using double gates MOSFET are individual circuit blocks with some specific functionality. This research uses the DG MOSFET model as a part of the larger circuit to implement the hysteresis neuron and the entire system of cells based on the hysteresis neuron circuits. The DG MOSFET model used in this research was developed by University of Florida [UFDG06].

5.3 Double gate Spice Model UFDG

The UFDG is a generic model for double gate developed by SOI group at University of Florida [UFDG06]. The model is compatible with the Spice3 simulator and is faster than TCAD model. The UFDG model has five terminals designated as drain, front gate, source, back gate and body. The syntax for this model in the Spice3 is given [UFDG06] as

MXXXX **ND NGF NS NGB** [**NB**] **MNAME L**=exp **W**=exp [**M**=exp] [**AD**=exp] [**AS**=exp] [**AB**=exp] [**NRD**=exp] [**NRS**=exp] [**NRB**=exp] [**PDJ**=exp] [**PSJ**=exp] [**RTH**=exp] [**CTH**=exp] [off] [**IC**= V_{ds}, V_{gfs}, V_{gbs}, V_{bs}]

where ND: Drain node, NGF: Front gate node, NS: Source node, NGB: Back gate node, NB: Body node, MNAME: Model name, L: Gate length, W: Gate width; the quantities in the bracket are optional and (for the remaining device line parameters refer Appendix 5). In the thesis the length and width of the double gate model used are L=100E-9 and W=1E-6 for NMOS transistor and L=100E-9 and W=2E-6 for PMOS transistor. Various parameters of the model used in simulation are defined in the Appendix 6 for NMOS and PMOS double gate. Appendix 6 also illustrates the methodology for running the Spice3 simulation using the UFDG model.

5.4 Double Gate Structures Self Bias Amplifier

A high-gain push pull amplifier using DG CMOS inverter, when biased in the transition region, using the TCAD simulator is proposed [Kaya06]. The amplifier characteristics can be altered by varying the bias of the bottom gate. The bias for the DG CMOS can be of the same voltage i.e. $V_{bg}{}^{p}=V_{bg}{}^{n}$ or it can be of conjugate pair $V_{bg}{}^{p}= -V_{bg}{}^{n}$ with equal magnitude but having opposite polarity. A third configuration of the amplifier called self bias can be obtained if the output of the IDDG CMOS pair drives the bottom-gates respective transistors i.e. $V_{bg}{}^{p}=V_{bg}{}^{n}=V_{out}$ as shown in Figure 39.

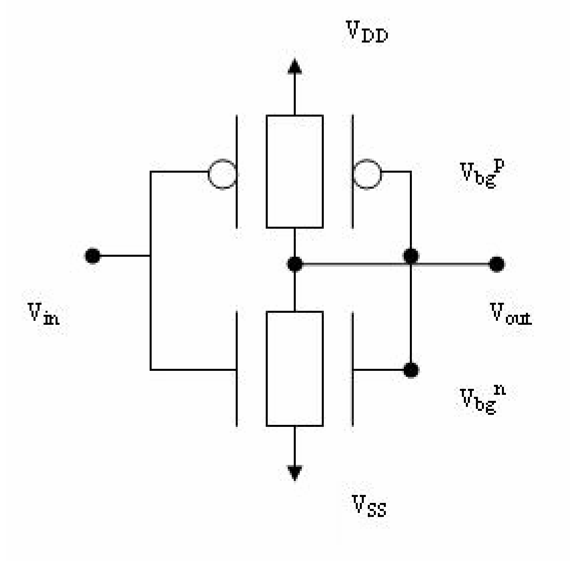


Figure 39: Self bias configuration of double gate MOSFET.

5.4.1 Spice 3 Code

Following is the spice3 netlist for the self bias configuration of double gate MOSFET.

```
.width in=80 out=80

.option post

MN0 out in ss out NDG L=100E-9 W=1E-6 IC=0, 0, 0, 0

MP0 out in dd out PDG L=100E-9 W=2E-6 IC=0, 0, 0, 0

VSS ss 0 DC -0.5

VIN in 0 DC 0.5

VDD dd 0 DC 0.5

.DC VIN -0.5 0.5 0.005V

.PRINT DC out

.OPTION

+ ABSTOL=1E-18 PIVTOL=1E-30 GMIN=1E-18 ITL1=1000
```

+ VNTOL=1E-6 RELTOL=1E-3 CHGTOL=1E-18

The NDG (NMOS) and PDG (PMOS) double gate model characteristics are given in the Appendix 6. As seen from the code the power supply used is ± 0.5 V, which is typically used for the low power applications.

5.4.2 Simulations Results

The gain for the self bias configuration is significantly lower and the curve obtained is linear.

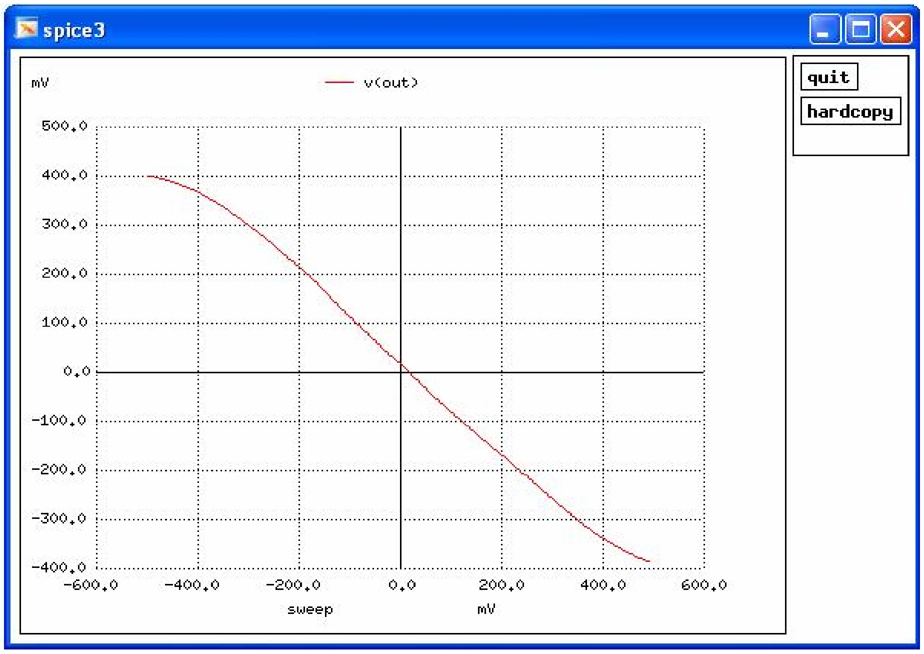


Figure 40: Double gate MOSFET self-bias amplifier configuration.

5.5 Double gate Structure for Schmitt Trigger

An efficient and programmable Schmitt Trigger structure using only 4 double gate MOSFET using the TCAD simulator is proposed [Kaya06]. The double gate CMOS amplifier's lateral shift of transfer function is demonstrated and can be easily utilized to build Schmitt trigger. The Schmitt trigger is a two stage device in which the response of the first stage is shifted using the conjugate bias of the second stage. The width of hysteresis depends upon the conjugate bias of the second stage [Kaya06].

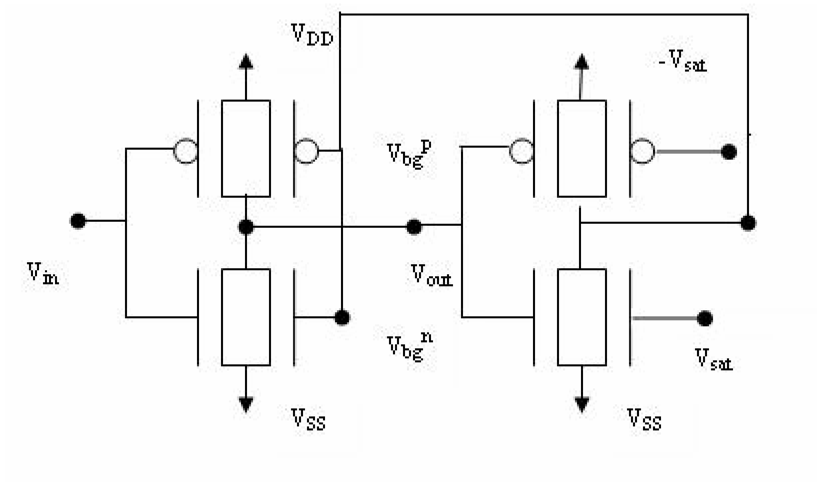


Figure 41: Schmitt trigger using double gate MOSFET.

5.5.1 Spice 3 Code

The spice3 netlist for the Schmitt trigger configuration of double gate MOSFET is

```
.width in=80 out=80

.options post

MN1 3 1 7 4 GN1 L=100E-9 W=1E-6 IC=0, 0, 0, 0

MP1 3 1 2 4 GP1 L=100E-9 W=2E-6 IC=0, 0, 0, 0

MN2 4 3 7 3 GN2 L=100E-9 W=1E-6 IC=0, 0, 0, 0

MP2 4 3 2 3 GP2 L=100E-9 W=2E-6 IC=0, 0, 0, 0

VSS 7 0 DC -0.3V

VDD 2 0 DC 0.3V

VIN 1 0 DC 1V
```

VIN 1 0 PWL(0nS -1 100nS 1 200nS -1)

.OPTIONS

+ ABSTOL=1E-18 PIVTOL=1E-30 GMIN=1E-10 ITL1=1000

+ VNTOL=1E-6 RELTOL=1E-3 CHGTOL=1E-18

The GN1 GN2 (NMOS) and GP1, GP2 (PMOS) model characteristics are given in the Appendix 6. As seen from the code the power supply used is ± 0.3 V, which is typically used for the low power applications.

5.5.2 Simulation Results

The Schmitt Trigger's width is tunable using the conjugate bias in the second stage, illustrated from the Figure 42 and 43. In Figure 42 the conjugate bias applied was ±0.5V.

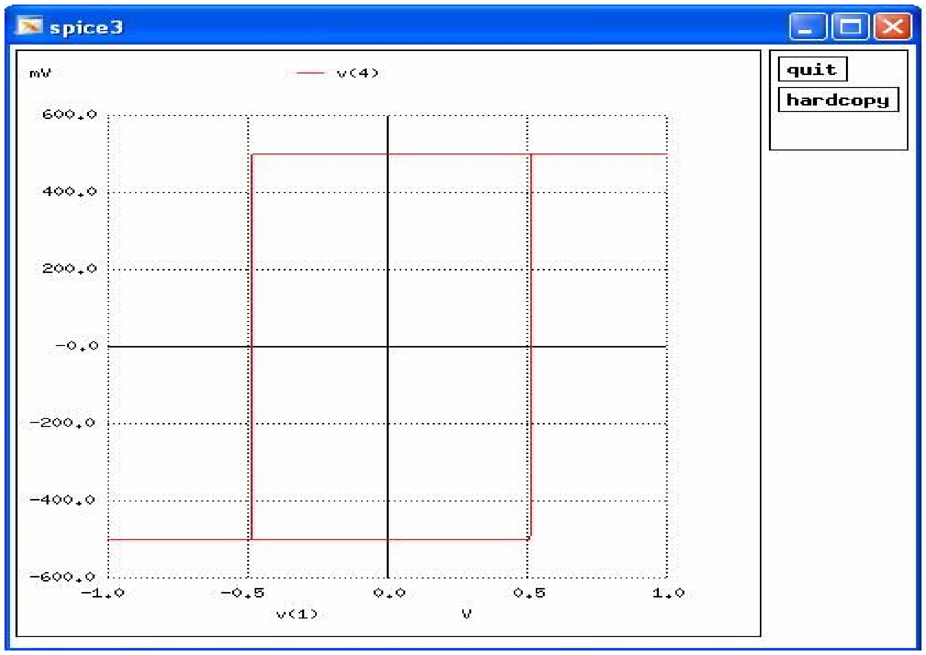


Figure 42: Schmitt trigger with conjugate bias ±0.5V.

In the Figure 43 the conjugate bias applied was ±0.3V. Thus it can be concluded that the width of the Schmitt trigger can be easily tuned by applying the proper conjugate bias voltages.

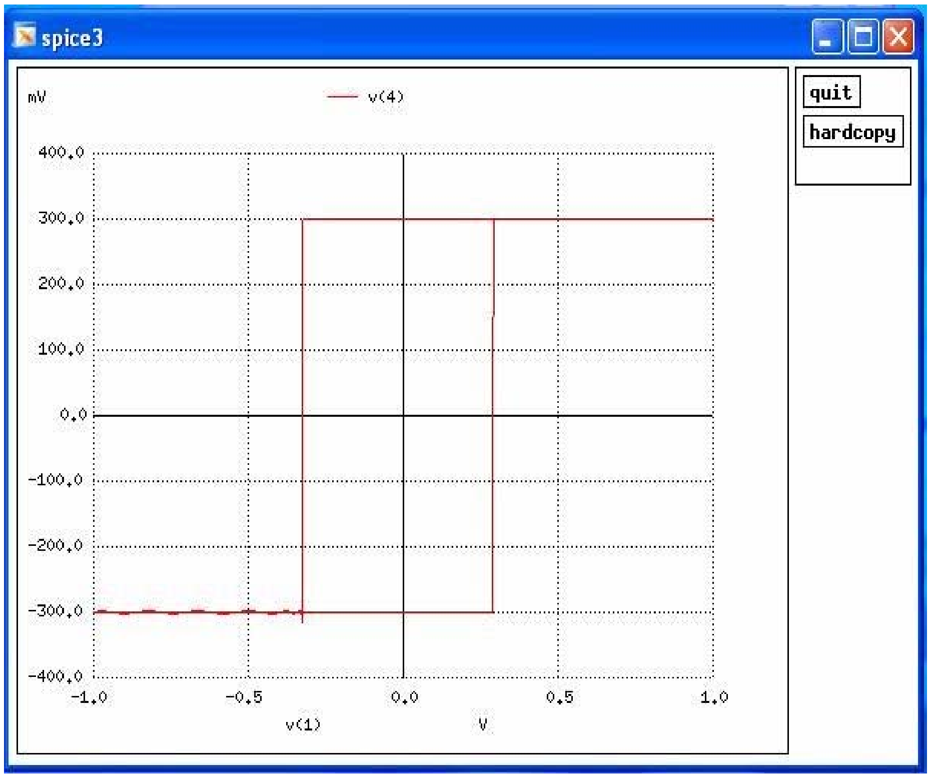


Figure 43: Schmitt trigger with conjugate bias ±0.3V.

5.6 Double gate Neuron Model

Figure 44 illustrates the double gate neuron model using the self bias configuration and Schmitt trigger configuration. As seen from the figure this model utilizes only 8 transistors. The transistor sizes used in the simulation are

- NMOS : Length = 100E-9 and Width = 1E-6
- PMOS : Length = 100E-9 and Width = 2E-6

The transistor models of NMOS and PMOS used in these simulations are given in the Appendix 6. A 1N4744A Zener diode is used in this simulation. The model details for the Zener diode are given in Appendix 3.1.

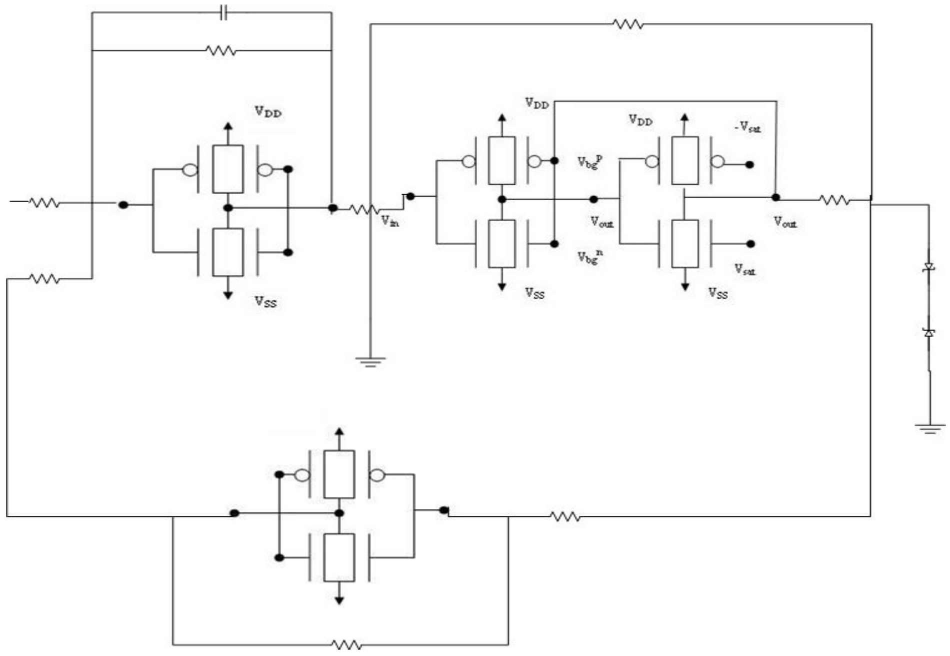


Figure 44: Double gate neuron model.

5.6.1 Spice 3 Code

This spice3 netlist written for the double gate MOSFET neuron model is as follows

```
xV5 4 0 Pulse_V_SourceV5

.SUBCKT Pulse_V_SourceV5 1 2

V 1 2 dc 0 ac 1 0

+   pulse(-1V 1V 0NS 0NS 0NS 1MS 2MS)

.ENDS

dD1 3 11 1N4744A__ZENER__1

dD2 0 11 1N4744A__ZENER__1
```

```
rR2 4 5 1.500e+003

rR1 5 8 9.090e+002

rR7 10 13 1.00e+002

rR4 13 3 1.00e+002

rR3 0 9 100

rR5 9 3 100

cC1 5 8 1.0E-6

VV7 12 0 dc 15 ac 0 0

xU3 10 13 self_bias

.SUBCKT self_bias 13  10

.width  in=80 out=80

.option post

MN01 10 13 ss 10  NDG L=100E-9 W=1E-6 IC=0,0,0,0

MP01 10 13 dd 10  PDG L=100E-9 W=2E-6 IC=0,0,0,0

VSS01 ss 0 DC -0.5

VDD01 dd 0 DC 0.5

.OPTION

+ ABSTOL=1E-18 PIVTOL=1E-30 GMIN=1E-18 ITL1=1000

+ VNTOL=1E-6 RELTOL=1E-3 CHGTOL=1E-18

.MODEL NDG NMOS LEVEL=10

        See Appendix 6 for NDG model

.MODEL PDG PMOS LEVEL=10
```

```
        See Appendix 6 for PDG model

.ENDS

rR6 10 5 4.990e+002

VV6 0 V6_OPEN_2 dc 15 ac 0 0

VV4 7 0 dc 15 ac 0 0

VV3 0 6 dc 15 ac 0 0

VV2 0 2 dc 15 ac 0 0

VV1 1 0 dc 15 ac 0 0

xU1 8 5 self_bias

xU2   9 8 schmitt_trigger

.SUBCKT schmitt_trigger 9 8

MN1 14 8 17 9  GN1 L=100E-9 W=1E-6 IC=0,0.5,0,0

MP1 14 8 16 9  GP1 L=100E-9 W=2E-6 IC=0,0,0,0

MN2 9 14 17 14  GN2 L=100E-9 W=1E-6 IC=0,-0.5,0,0

MP2 9 14 16 14  GP2 L=100E-9 W=2E-6 IC=0,0,0,0

VSS 17 0 DC -0.5V

VDD 16 0 DC 0.5V

.OPTIONS

+ ABSTOL=1E-18 PIVTOL=1E-30 GMIN=1E-10 ITL1=1000

+ VNTOL=1E-6 RELTOL=1E-3 CHGTOL=1E-18

.MODEL GN1 NMOS LEVEL=10

        See Appendix 6 for GN1 model
```

```
.MODEL GN2 NMOS LEVEL=10
        See Appendix 6 for GN2 model
.MODEL GP1 PMOS LEVEL=10
        See Appendix 6 for GP1
.MODEL GP2 PMOS LEVEL=10
        See Appendix 6 for GP2
.ENDS
rR18 3 18 10
.MODEL 1N4744A__ZENER__1 D
        See Appendix 3.1 for Zener 1N4744A model
.ENDS
.TRAN 0.1MS 50MS
.END
```

5.6.2 Simulation Results

The methodology to run the netlist in elaborated in the Appendix 6.3. Figures 45 and 46 are the double gate MOSFET model hysteresis characteristics, obtained after the netlist was run on Solaris 10 machine. The double gate hysteresis neuron can be used as a building block to build a larger Hopfield neural network. This network can solve several constraint satisfaction problems including the N-queens problem discussed in chapter 4. If a network with topology, suggested in Section 4.4 is implemented, the network can be utilized to solve the 8 queen problems.

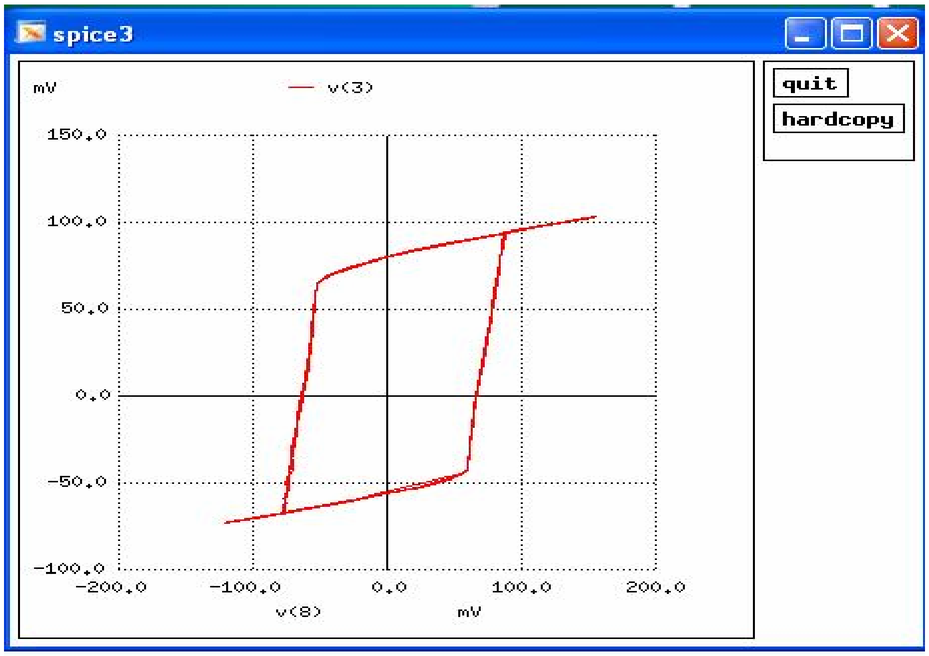


Figure 45: Double gate MOSFET hysteresis neuron simulation one.

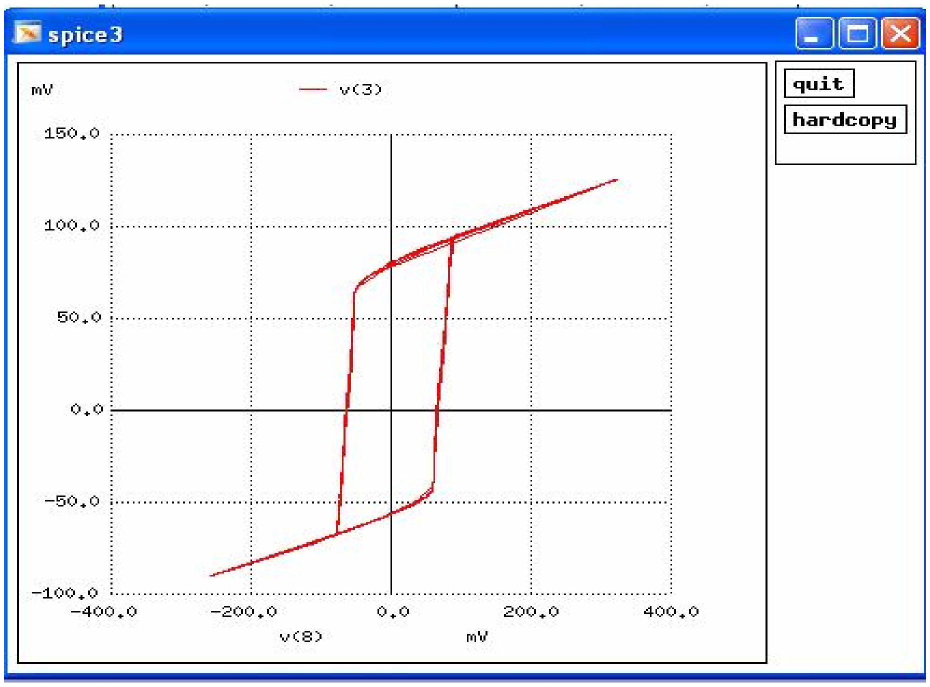


Figure 46: Double gate MOSFET neuron model simulation two.

5.7 Summary

Double gate MOSFET not only saves the transistor count but also operates at lower voltages than present CMOS devices. Suppose we build a network for solving the 8-queen problem, using hysteresis neuron. 60 transistors are needed to build the single opamp based neuron, as a one LM741 operational amplifier requires 20 transistors [N74107]. The number of transistors required for 8x8 networks will be 3840 while the count of transistors using double gate neurons is 512 since it requires only 8 transistors to build a single neuron. This is 3328 transistors less than opamp implementation which is a significant reduction in the transistor count. The supply voltage required for opamp based neuron is between ±12V and ±15V while it is ±0.5V in double gate MOSFET based neuron. Any application utilizing a neural network principle has hundreds of neurons and the size of network is comparatively large for 8x8 neural networks. Thus, using the double gate neuron model, the same functionality can be obtained using less number of transistors operating at lower voltages. This thesis also demonstrated that we can build larger circuits using the double gate MOSFET rather than using it for designing small individual components. This attempt of implementing a neural network using double gate MOSFET can be considered as a proof of concept for implementing larger integrated circuits using double gate MOSFET.

Double gate MOSFET can be considered as an option for futuristic device which might be able to replace the conventional CMOS in coming decades.

5.7.1 Hysteresis Neuron Model using OTA

Operational transconductance amplifier (OTA) is a voltage controlled current source (VCCS). It takes differential input voltage and produces output current. A simple design of hysteresis chaos generator using just six elements - two capacitors, two resistors, one linear VCCS and one hysteresis VCCS is proposed [Naka96]. Figure 47 gives a block level circuit design of hysteresis chaos generator using the hysteresis VCCS.

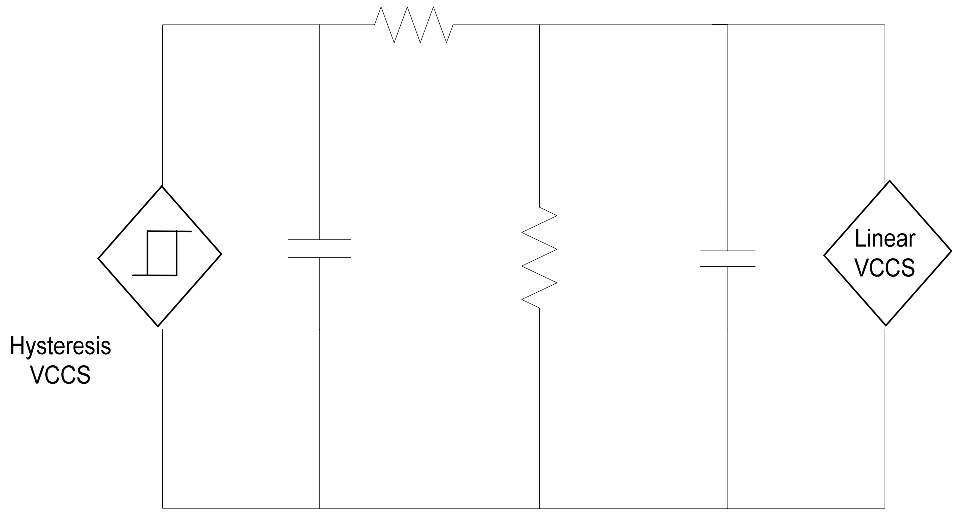


Figure 47: RC OTA design for hysteresis chaos generator.

Kim proposed the Schmitt trigger design [Kim97], using OTA and a register. Using the Schmitt trigger design using OTA, hysteresis OTA is build with the help of built in models of OTA in Multisim software. Figure 48 gives complete realization for the block level design of hysteresis chaos generator illustrated in the Figure 47.

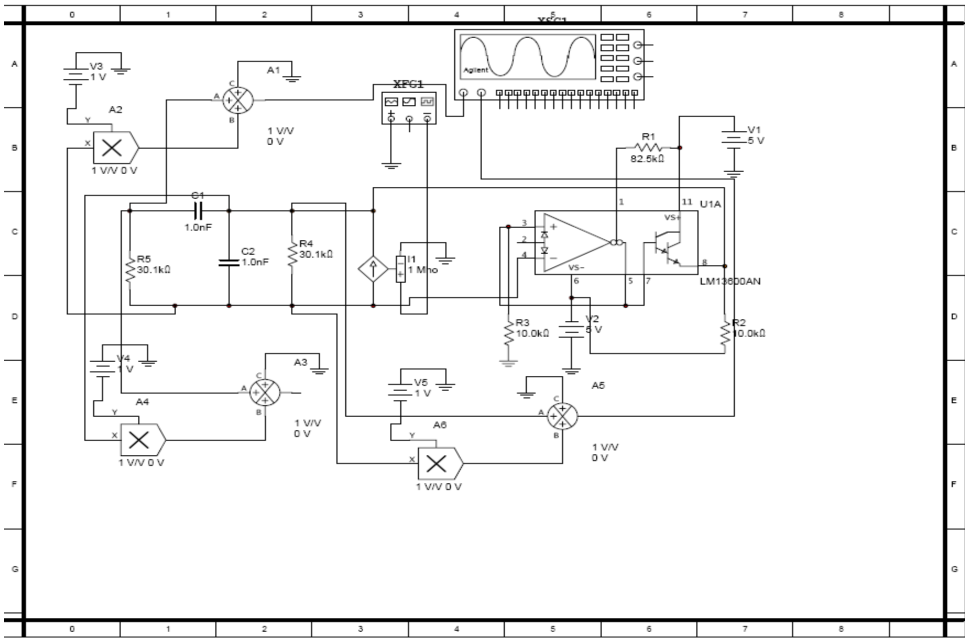


Figure 48: OTA bases hysteresis neuron.

The circuit shown in Figure 48 was simulated in the Multisim software and the results are shown in the Figures 49 and 50. Figure 49 demonstrated the chaos phenomenon using OTA based hysteresis chaos generator.

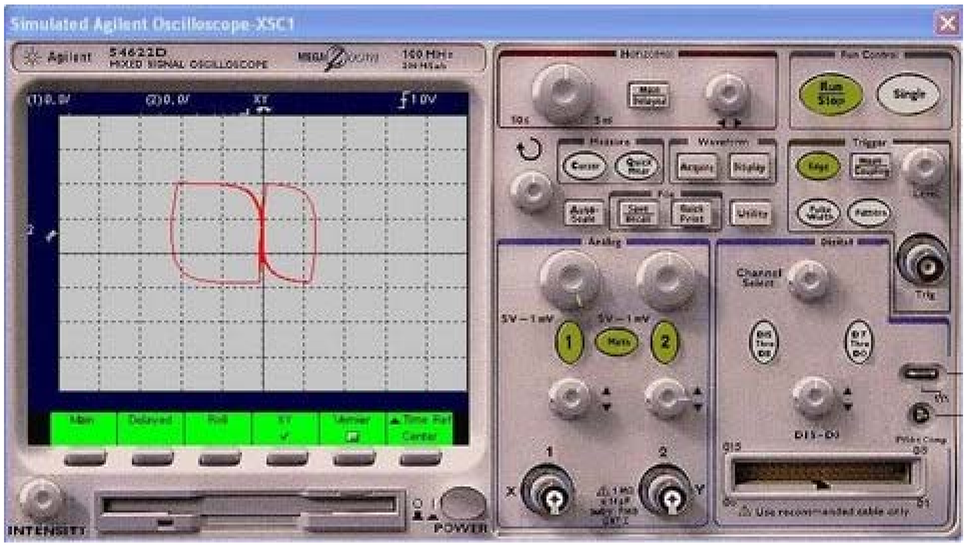


Figure 49: Chaos using RC OTA.

Changing the initial conditions the hysteresis characteristics with the same circuit can be obtained.

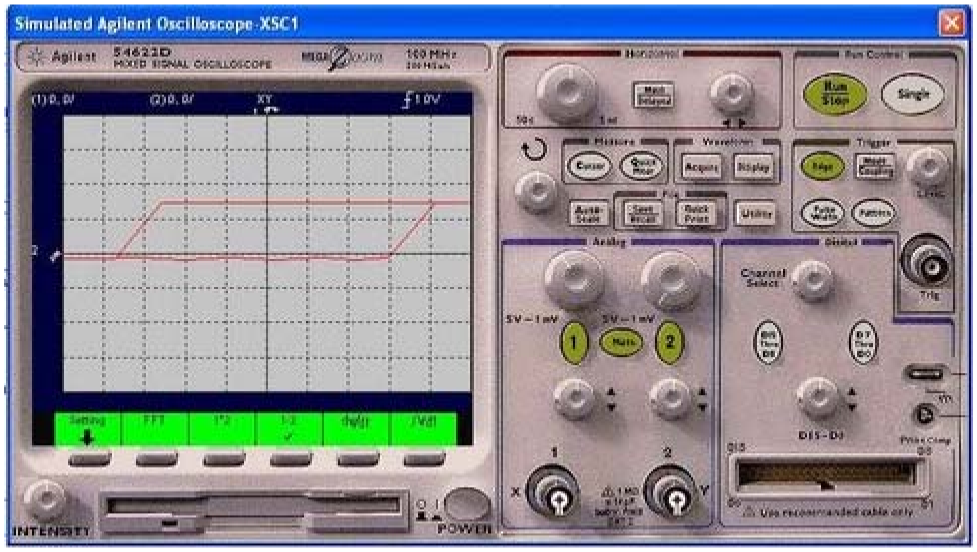


Figure 50: Hysteresis using RC OTA.

6. Conclusion and Future Work

6.1 Conclusion

The work presented in this thesis explores the relatively new devices - double-gate MOSFET- for the implementation of the hysteresis neural network. It is the first attempt to use these devices in the neural network field, presenting a large scale application of double gate MOSFETs.

To conclude most of the research goals have been accomplished as mentioned in Section 1.2.

- Chaotic phenomenon has a good ability to search for an optimal solution of a problem with many local minima. The current work uses the chaos theory with focus on electrical circuits. Chua's circuit is a good example to demonstrate the type of non-linearity in the electrical circuit which causes chaos.
- The opamp based hysteresis neuron model is implemented and its stability is explained. Stability criterion of the network can be used as a cost function and when the network reaches its stable point, this corresponds to a solution. This is explained in regards to N-queens problem.
- The circuit topology for solving 8-queens problem is proposed that can be extended to N-queens.
- The hysteresis neuron model is implemented using the double gate MOSFET. Once we are able to build this analog circuit representation of the network, the solution can be obtained in a constant time irrespective of the network size.

6.2 Future Work

The CMOS implementation of OTA based hysteresis chaos generator circuit has been designed with additional functionality to control the chaos [Naka97]. The Hysteresis VCCS is designed using 11 transistors. The ratio of width and length (W/L) of these transistors varies from 10/20 to 50/10. The bias used in simulation of the circuit is ±8.5V and the source voltage around ±10V. Double gate implementation of OTA has been proposed [Kaya06] using only one capacitor and four transistors with relatively small voltage levels and bias conditions. This implementation has been utilized to build the hysteresis neuron. The future challenge is to develop the hysteresis VCCS using double-gate MOSFET so that it can be used to build chaos generator using less number of components.

To manually implement neural network circuit in a schematic editor is a challenge. As the number of neurons in the circuit increases the connections between neurons are complex to design using schematic editor. Another area for future work would be to build a schematic editor tool that can be use to build neural network circuits efficiently with ease.

References

[ChMs07] Chua Leon O., Nonlinear Electronics lab, University of California at Berkeley, “Working with chaos: Simulating Chua’s circuit”, August 2007, http://nonlinear.eecs.berkeley.edu/chaos/StrangeAttractor.jpg.

[Chua87] Chua, Leon O., Desoer, Charles A. and Kuh, Ernest S., "Linear and Nonlinear Circuits” McGraw-Hill, 1987.

[Chua07] Chua Leon O., Nonlinear Electronics lab, University of California at Berkeley, “Working with chaos: Simulating Chua’s circuit”, August 2007, http://nonliner.eecs.berkeley.edu/chaos/introduction_nonliner_circuit _analysis.pdf.

[Cros07] Cross Michael, Chaos Demonstration, California Institute of Technology September2007, http://www.its.caltech.edu/~mcc/chaos_new/Chua_docs/ works.html.

[Fink87] Finkel Raphel, Manber Udi, “DIB-a distributed implementation of backtracking”, *ACM TOPLAS*, vol. 9, issue 2, pp. 235-256 1987.

[Hopf82] Hopfield J.J., “Neural networks and physical systems with emergent collective computational properties”, *PNAS* vol. 79, no.8, pp. 2554-2558, April 1982.

[Hopf85] Hopfield, J.J. and Tank, D. "Neural computation of decisions in optimization problems", *Biological Cybernetics* vol.52, pp. 141-152, 1985.

[Hutc02] Hutch James A. ,Bourianoff George I., Zhirnov Victor V., Brewer Joe E., “Extending the Road beyond CMOS”, *IEEE Circuits and Devices Magazine*, vol. 18, no.2, pp. 28-41 March 2002.

[Inte07] Moore Gordon E., Intel Inc, Technology and Research, Silicon, September 2007, http://www.intel.com/technology/mooreslaw.

[ITRS07] International Technology Roadmap for Semiconductor, August 2007 http://www.itrs.net/Links/2006Update/FinalToPost/00_ExecSum2006Update.pdf.

[Kaya06] Kaya S., Hamed H. F. A., Starzyk J., “Low-power Tuneable Analog Circuit Blocks Based on Nanoscale Dual-Gate MOSFETs”, in *IEEE Nano*, vol. 1, June 2006, pp. 355-358.

[Kenn92] Kennedy M. P., “Robust Op-amp realization of Chua’s Circuit”, *Ferquenz*, vol .46, pp66-80, March-April1992.

[Kim97] Kim k., Cha H.W. and Chung W.S., “OTA-R Schmitt triggers with independently controllable threshold and output voltage levels”, *Electronics letters* vol.33, no.13,pp. 1103-1105, June 1997.

[Lode07] Loder C., “Neural Networks: An Overview”, October 2007, at Northeastern University, http://www.ccs.neu.edu/groups/honors-program/freshsem/ 19951996/colder.

[Mats84] Matsumoto T., “A Chaotic Attractor from Chua’s Circuit." *IEEE Trans. CAS* vol.31, no.21, pp.1055-1058, December 1984.

[More65] Moore Gordon E., “Cramming More Components onto Integrated Circuits”, *Electronics*, vol. 38, no.8, pp. 114-117, April 1965.

[Mult07] MultiSim NI website, “The NI Electronics Education Platform” September 2007, http://www.ni.com/swf/presentation/us/ewb/eep.

[N74107] National Semiconductor, LM741 operational amplifier datasheet, August 2007, http://cache.national.com/ds/LM/LM741.pdf.

[Naka96] Nakagawa S. and Saito T., “An RC OTA Hysteresis Chaos Generator”, *IEEE transaction on circuits and systems I*, vol. 43, no. 12, pp. 1019-1021, December 1996.

[Naka97] Nakagawa S. and Saito T., “CMOS Hysteresis Chaos generator with ISS control”, in *Proc.* o*f IEEE/ISCAS*, Hong Kong, June1997, pp.979-800.

[Naka99] Nakaguchi T., Jin’no K. and Tanaka M., “Hysteresis Neural Network for N-Queen problem”, *IECIE TRANS. Fundamentals*, vol. E82-A, no.9, pp.1851-1859, September 1999.

[Orca07] OrCAD Tutorial, “An OrCAD tutorial for Elec 424”, spring 2002, http://www.picvietnam.com/download/ORCAD/orcad_tutor.pdf.

[Pitt43] McCulloh,W.S. and Pitts, W.H., “A logical calculus of the ideas immanent in nervous activity”, in *Bulletin of Mathematical Biophysics,* vol.5, no. 4, pp.115-133, December1943.

[Rose58] Rosenblatt F. “The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain”, in *Psychological Review*, vol. 65, no. 6, pp. 386-408, November, 1958.

[Seki84] Sekigawa T., Hayashi Y., "Calculated Threshold-voltage Characteristics of an XMOS transistor having an Additional Bottom Gate", in *Solid State Electron*. vol.27, pp. 827-828, 1984.

[Simo98] Simon H., "Neural Networks: A comprehensive Foundation", 2nd Edition, Pearson Education, 1998.

[Scho07] Schoenauer Marc, Equipe Evolution Artificielle et Apprentissage de l'x, Université Paris Sud, Septembre 2007, http://www.lri.fr/~marc/EEAAX/Neurones/tutorial/aneuron/html/ index.html.

[Sosi90] Sosic R., Gu J., "A polynomial time algorithm for the N-Queens problem", in *ACM SIGART Bulletin*, vol 1, issue 3, pp: 7-11, 1990.

[spic07] Spice3 user manual, University of California at Berkeley, August 2007, http://bwrc.eecs.berkeley.edu/Classes/IcBook/SPICE/index.html.

[Thin07] Thinkquest Library, Making order out of chaos", August 2007, http://library.thinkquest.org/12170/history/lorenz.html.

[Toku97] Tokuda I., Nagashima T. and Aihara K., "Global Bifurcation structure of Chaotic Neural Network and its application to Traveling Salesman problem", *Neural Networks*, vol.10, no.9, pp.1673-1690, 1997.

[Ueda07] Ueda Yoshisuke., Kyoto University, September 2007, http://muratopia.org/NUGW/People/uedaY.html.

[UFDG06] SOI Group, University of Florida,”UFDG MOSFET user guide” ver. 3.6 April 2006.

[Utex07] Trump M.,” what is chaos? A five part online course”, August 2007 http://order.ph.utexas.edu/chaos/index.html.

[Vara05] Varadharajan S., “Digital and Analog application of Double gate MOSFETS”, M.S. Thesis, Ohio University, Athens OH, USA 2005.

[Wong02] Wong S.P., “Beyond the conventional transistors”, IBM J. Res. & Dev. vol. 46 no.2/3, March/May 2002.

[Wiki07] Wikipedia, web based encyclopedia, August 2007, www.wikipedia.org.

Appendix 1: Multisim Model Details

1.1LM 741 Models Details

```
.SUBCKT LM741 1 2 3
* EWB Version 4 - 3 Terminal Opamp Model
* A= 200000 RI= 2e+006 RO= 75 VSW+= 21 VSW-= -21
* Vos= 0.001 Ibs= 8e-008 Ios= 2e-008 SR+= 500000
* fu= 1.5e+006 fp2= 1e+032 CC= 3e-011
Vos 4 1 DC 0.001V
Ib1 4 0 9e-008A
Ib2 2 0 7e-008A
G1 0 5 4 2 0.0584804
G2 0 6 5 0 0.779738
G3 0 3 6 0 0.779738
Ri 4 2 2e+006ohm
R1 5 0 1000ohm
R2 6 0 75ohm
R3 3 0 75ohm
C1 5 0 2.12207e-005
C2 6 0 2.12207e-035
Cc 5 0 3e-011
.ENDS
```

1.2 BZX83C10 Model Details

```
.MODEL BZX83C10 D IS= 2.524e-016 RS= 0.5164 CJO= 0.999e-12 VJ= 0.75
+ TT= 5e-009 M= 0.3333 BV= 10.11 IBV= 0.3572 N= 1
+ EG= 1.11 XTI= 3 KF= 0 AF= 1 FC= 0.5
+ TNOM= 27
```

Appendix 2: OrCAD Model Details

2.1 Opamp 741

```
.SUBCKT 741__OPAMP__1 1 2 3 4 5
* EWB Version 4 - 5 Terminal Opamp Model
* nodes: 3=+ 2=- 1=out 5=V+ 4=V-
* VCC= 15 VEE= -15 CC= 3e-011 A= 200000 RI= 2e+006
* RO= 75 VOS= 0.001 IOS= 2e-008 IBS= 8e-008
* VSW+= 14 VSW-= -14 CMMR= 90
* ISC= 0.025 SR= 0.5 Fu= 1e+006 Pm= 6.09112e-007
    VC 5 15 DC 1.68573V
    VE 12 4 DC 1.68573V
    IEE 10 4 DC 1.516e-005A
    R1  10  0 10Gohm
    R6  11  0 100Kohm
    R7   5  4 1Kohm
    Rc1  6  5 5305.16ohm
    Rc2  5  7 5305.16ohm
    Re1  9 10 1839.19ohm
    Re2  8 10 1839.19ohm
    Ro1  1 14 37.5ohm
    Ro2 14  0 37.5ohm
    Ree 10  0 1.31926e+007ohm
    Rcc  0 13 2.20906e-005ohm
    Cee  0 10 1e-012
    Cc  14 11 3e-011
    C1   6  7 1e-016
    GA  11  0  6  7 0.000188496
    GC   0 13  1  0 45268.1
    GB  14  0 11  0 282.942
    GCM  0 11 10  0 5.96075e-009
    D1  14 13 Dopamp1
    D2  13 14 Dopamp1
    D3   1 15 Dopamp2
    D4  12  1 Dopamp2
    Qt1  6 2 9 Qopamp1
    Qt2  7 3 8 Qopamp2
.MODEL Dopamp1 D (Is=7.53769e-014A Rs=0 Cjo=0F Vj=750mV Tt=0s M=0)
.MODEL Dopamp2 D (Is=8e-016A Rs=0 Cjo=0F Vj=750mV Tt=0s M=0)
.MODEL Qopamp1 NPN (Is=8e-016A BF=83.3333 BR=960m
+       Rb=0ohm Re=0ohm Rc=0ohm Cjs=0F Cje=0F Cjc=0F
+       Vje=750m Vjc=750m Tf=0 Tr=0 mje=0 mjc=0 VA=50)
.MODEL Qopamp2 NPN (Is=8.30948e-016A BF=107.143 BR=960m
+       Rb=0ohm Re=0ohm Rc=0ohm Cjs=0F Cje=0F Cjc=0F
+       Vje=750m Vjc=750m Tf=0 Tr=0 mje=0 mjc=0 VA=50)
.ENDS
```

2.2 Zener Diode 1N4744A Model

```
.MODEL 1N4744A__ZENER__1 D IS= 5.038e-012 RS= 0.001 CJO= 1e-012 VJ=
0.75
+                TT= 5e-009 M= 0.3333 BV= 15.05 IBV= 0.1087 N= 1
+                EG= 1.11 XTI= 3 KF= 0 AF= 1 FC= 0.5
+                TNOM= 27
```

Appendix 3: Spice3 Models Details

3.1 Spice3 Zener Model:

```
.MODEL 1N4744A__ZENER__1 D IS= 5.038e-012 RS= 0.001 CJO= 1e-012 VJ=
0.75
+                TT= 5e-009 M= 0.3333 BV= 15.05 IBV= 0.1087 N= 1
+                EG= 1.11 XTI= 3 KF= 0 AF= 1 FC= 0.5
+                TNOM= 27
```

3.2 Spice3 opamp Model:

```
.SUBCKT OPAMP1 1 2 3 4 5
* EWB Version 4 - 5 Terminal Opamp Model
* nodes: 3=+ 2=- 1=out 5=V+ 4=V-
* VCC= 15 VEE= -15 CC= 3e-011 A= 200000 RI= 2e+006
* RO= 75 VOS= 0.001 IOS= 2e-008 IBS= 8e-008
* VSW+= 14 VSW-= -14 CMMR= 90
* ISC= 0.025 SR= 0.5 Fu= 1e+006 Pm= 6.09112e-007
    VC 5 15 DC 1.68573V
    VE 12 4 DC 1.68573V
    IEE 10 4 DC 1.516e-005A
    R1  10  0 10Gohm
    R6  11  0 100Kohm
    R7   5  4 1Kohm
    Rc1  6  5 5305.16ohm
    Rc2  5  7 5305.16ohm
    Re1  9 10 1839.19ohm
    Re2  8 10 1839.19ohm
    Ro1  1 14 37.5ohm
    Ro2 14  0 37.5ohm
    Ree 10  0 1.31926e+007ohm
    Rcc  0 13 2.20906e-005ohm
    Cee  0 10 1e-012
    Cc  14 11 3e-011
    C1   6  7 1e-016
    GA  11  0  6  7 0.000188496
    GC   0 13  1  0 45268.1
    GB  14  0 11  0 282.942
    GCM  0 11 10  0 5.96075e-009
    D1  14 13 Dopamp1
    D2  13 14 Dopamp1
    D3   1 15 Dopamp2
    D4  12  1 Dopamp2
    Qt1  6 2 9 Qopamp1
    Qt2  7 3 8 Qopamp2
.MODEL Dopamp1 D (Is=7.53769e-014A Rs=0 Cjo=0F Vj=750mV Tt=0s M=0)
.MODEL Dopamp2 D (Is=8e-016A Rs=0 Cjo=0F Vj=750mV Tt=0s M=0)
.MODEL Qopamp1 NPN (Is=8e-016A BF=83.3333 BR=960m
+       Rb=0ohm Re=0ohm Rc=0ohm Cjs=0F Cje=0F Cjc=0F
+       Vje=750m Vjc=750m Tf=0 Tr=0 mje=0 mjc=0 VA=50)
```

```
.MODEL Qopamp2 NPN (Is=8.30948e-016A BF=107.143 BR=960m
+       Rb=0ohm Re=0ohm Rc=0ohm Cjs=0F Cje=0F Cjc=0F
+       Vje=750m Vjc=750m Tf=0 Tr=0 mje=0 mjc=0 VA=50)
.ENDS
```

3.3 Running the Netlist file

1. log onto Solaris machine
2. navigate to the folder containing the spice netlist
3. type command spice3
4. press return key
5. type command source filename
6. type command run
7. type command plot

To run the spice netlist, navigate to the folder that contains the netlist file and follow the instruction given below

```
P2% spice3 (enter)

Spice3 -> source workingSpice.in

Spice3-> run

Spice3-> plot -v(3) vs v(8)
```

Appendix 4: Matlab Code for N-Queen Problem

```
x=random('Normal',0,1,4,4);
N=8;
y=zeros(N);
cont=0;
l=1;
m=1;
n=0;
o=0;
conty=0;
while (cont~=8)
cont=1;
a=1;
b=8;
l =round( a + (b-a) * rand(1))
m =round( a + (b-a) * rand(1))
y=zeros(N);
y(l,m)=1;
for i=1:63
   i =round( a + (b-a) * rand(1))
   j =round( a + (b-a) * rand(1))
        sumcol=sum(y(:,j));
        sumrow=sum(y(i,:));
       k=i+j;
       sumdiag1=sum(diag(y,j-i));
       sumdiag2=0;
       if (k<=N)
           for l=1:k-1
           sumdiag2=sumdiag2+y(l,k-l);
           end
       else
           for l=k-N:N
           sumdiag2=sumdiag2+y(l,k-l);
           end
       end
         sumdiag=sumdiag1-sumdiag2;
         p(i,j)=(1-sumrow)+(1-sumcol)+(1-sumdiag1-sumdiag2);
         if (p(i,j)==3)
             y(i,j)=1;
             cont=cont+1;
         end
    end
end
```

Appendix 5: DG/SOI MOSFET Device-Line Parameters

Name	Description	Units	Defaults
M	Number of gate fingers	m	1
AD	Area of drain region	m^2	0.0
AS	Area of source region	m^2	0.0
AB	Area of body	m^2	0.0
NRD	Squares of drain region	sq.	0.0
NRS	Squares of source region	sq.	0.0
NRB	Squares of body-contact region	sq.	0.0
PDJ	Perimeter of drain-body junction	m	W
PSJ	Perimeter of source-body junction	m	W
RTH	Thermal resistance	kW	0.0
CTH	Thermal capacitance	kW	0.0
IC	Initial conditions	V	0.0

Appendix 6: DG MOSFET Model Line and Structural Parameters used in Simulation

6.1 NMOS Double Gate Model

```
.MODEL GN1 NMOS LEVEL=10
+ WKFG=4.40
+ WKBG=4.40
+ NSF=1e10
+ NSB=1e10
+ TOXF=2n
+ TOXB=2n
+ TSI=10.0n
+ NBODY=1E15
+ UO=1400./500.
+ THETA=1.0
+ VSAT=1e7
+ VO=0
+ BLIM=0
+ QMX=1.0
+ QMD=1.0
+ NDS=1E20
+ JRO=1E-11
+ M=2
+ SEFF=1E5
+ SCEB=0
+ DL=1e-9
+ LES=0.0n
+ LED=0.0n
+ RD=5E-6
+ RS=5E-6
+ SFACT=5
+ WFACT=5
+ SO=1
+ BJT=0
+ DG=1
```

6.2 PMOS Double Gate Model

```
.MODEL GP1 PMOS LEVEL=10
+ WKFG=4.80
+ WKBG=4.80
+ NSF=1e10
+ NSB=1e10
+ TOXF=2n
+ TOXB=2n
+ TSI=10.0n
+ NBODY=1E15
+ UO=1400./470.
+ THETA=0.8
+ VSAT=1e7
+ VO=0
+ BLIM=0
+ QMX=1.0
+ QMD=1.0
+ NDS=1E20
+ JRO=1E-11
+ M=2
+ SEFF=1E5
+ SCEB=0
+ DL=1e-9
+ LES=0.0n
+ LED=0.0n
+ RD=5E-6
+ RS=5E-6
+ SFACT=5
+ WFACT=5
+ SO=1
+ BJT=0
+ DG=1
```

6.3 Running UFDG Simulation on Spice3

- Go to the directory where your spice file is located
- On console type command **run ufdg –r <filename.in**
- After simulation is complete use command **nutmeg,** this is used to plot the results.