Hardware implementation of the complex Hopfield neural network

Chih Kang Cheng

HARDWARE IMPLEMENTATION OF THE COMPLEX HOPFIELD NEURAL NETWORK

A Thesis

Presented to the

Faculty of

California State University,

San Bernardino

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

in

Computer Science

---

by

Chih Kang Cheng

December 1995

HARDWARE IMPLEMENTATION OF THE COMPLEX HOPFIELD NEURAL NETWORK

A Thesis

Presented to the

Faculty of

California State University,

San Bernardino

---

by

Chih Kang Cheng

December 1995

Approved by:

[redacted] 12/8/95

Dr. George M. Georgiou, Chair, Computer Science Date

[redacted]

Dr. Owen J. Murphy

[redacted]

Dr. Kerstin Voigt

ABSTRACT

This thesis examines the hardware implementation and simulation of complex analog Hopfield Neural Networks (HNN). In the evolution of Neural Networks, one rather unexplored area has been the use of complex domain computation. Although complex neural networks have been proposed by previous research (Birx and Pipenberg, 1989; Culhane and Pecker, 1989; Szilagyi et al., 1990; Georgiou, 1992; Yang et al., 1994), none has thus far implemented complex domain HNNs as a circuit.

First, the complex weights were designed as circuit components using SPICE which is a popular circuit design simulator. In order to test the correctness of the complex weight circuit components, the Discrete Fourier Transform (DFT) was implemented as a circuit using the same type of components. The DFT circuit output was compared with the DFT calculated output, and it was found that two values agreed. This was a validation of the circuit design of the complex weights.

Second, in the design a suitable activation function was used. The activation function was suggested by Georgiou (1992) as a suitable one for the complex Hopfield Neural Net. This is a normalizing function, as it normalizes a given complex value to have unity magnitude, while the phase remains unchanged. Thus, it limits the output of a neuron to having phase information only.

In attempting to simulate the complex HNN, it was discovered that the circuit simulator SPICE had a limitation which did not allow proper simulation of the complex HNN. This was overcome by decomposing the complex HNN into a real coupled network. The real coupled network was designed and was shown to be equivalent to the complex HNN. The weights of the real coupled network are purely resistive, as opposed to inductive ones in the complex HNN. The equivalent real network was tested using SPICE, and the results were verified by using a C program that simulated the same network.

ACKNOWLEDGMENTS

I would like to first thank my advisor Dr. Georgiou. He not only shared a lot of his knowledge in this field, but also gave me a lot of encouragement when things did not go as planned. Thanks to Dr. Murphy and Dr. Voigt who as committee members helped very much in this project. Also, I very much appreciate the help I received from Chi-Hung of UCLA. He helped me and gave me suggestions in the circuit parts of the thesis. I would like to say thanks to my parents; they always supported me when I felt lonely in this country. Their love gave me the strength to succeed. My wife, Ching-Ya, was also a power source for me to complete this thesis. Chun-Jei, a director of SPICE company in Taiwan, kindly helped me when I started learning SPICE. Finally, I would like to say thanks to my roommate Richard and my cousin Nelson for correcting my writing, my uncle Simon and aunt Grace for taking good care of me during the period of my stay in the USA. Thanks to all of you.

TABLE OF CONTENTS

ABSTRACT iii

ACKNOWLEDGMENTS v

LIST OF FIGURES ix

CHAPTER ONE

Hopfield Neural Networks 1

Introduction 1

Discrete-step HNN 4

The Energy Function 5

Continuous HNN 7

The Energy Function of Continuous HNN 9

CHAPTER TWO

Complex Neural Networks 10

Introduction 10

Implementations 12

Complex HNN 13

Discrete-step Complex HNN 13

Continuous Complex HNN 14

CHAPTER THREE

Implementations of Complex Neural Networks 15

Implementation of Complex Weights 15

The Weights in the Real Domain 16

An Inverter.. 17

The Weights in the Complex Domain 18

An Example Using Complex Weights 20

Results of the DFT Simulation 26

Inputs... 26

Outputs.. 26

CHAPTER FOUR

Decomposing the Complex HNN into a Real Coupled One...................... 29

The Limitation of SPICE 29

Decomposing the Complex HNN into a Real Coupled One 30

The Equivalency of the Complex and Real Circuits........................... 32

Mapping Complex Equations to Real Ones.. 34

CHAPTER FIVE

Results 37

The Input and Output of Simulating the Complex HNN 37

Input of Pulses [1, -1] 37

Input of Constants [1, -1] 41

The Complex HNN with Four Neurons... 44

CHAPTER SIX

Conclusions 46

APPENDIX A

The Simulation of the DFT with N =16, 32 47

APPENDIX B

The SPICE Code of Simulating Complex HNN 50

BIBLIOGRAPHY 53

LIST OF FIGURES

Figure 1.1 The Topology of Hopfield Neural Network (HNN) 2

Figure 1.2 Single-layer Feedback Neural Network 4

Figure 1.3 The Neural Network Model Using Electrical Components 8

Figure 1.4 The Input Node of the i'th Neuron 8

Figure 3.1 The Weights in the Real Domain 16

Figure 3.2 The Circuit of An Inverter 18

Figure 3.3 The Weights *a*+*bi* in the Complex Domain 18

Figure 3.4 The Weights *a-bi* in the Complex Domain 19

Figure 3.5 Another Simulation for the Complex Weight *a-bi*.......................... 19

Figure 3.6 The Circuit with N=2 22

Figure 3.7 The Circuit with N=4 24

Figure 3.8 The Circuit with N=8 25

Figure 3.9 The Input Data of Simulating Complex Weights 26

Figure 3.10a The Output of Simulating the Complex Weights (N=2) 27

Figure 3.10b The Output of Simulating the Complex Weights (N=4) 28

Figure 3.10c The Output of Simulating the Complex Weights (N=8) 29

Figure 4.1 Complex HNN with Two Neurons 30

Figure 4.2 The i'th Complex Neuron...31

Figure 4.3 Decomposing a Complex Neuron into Two Real Neurons 31

Figure 4.4 Decomposing the Complex HNN to Real Coupled Networks 33

Figure 5.1 Input of Pulses with Different Weights [0.004, 0.002].....................38

Figure 5.2 Input of Pulses with Different Weights [0.01, 0.03] 39

Figure 5.3 Input of Pulses with Different Weights [0.3, 0.4] 40

Figure 5.4 Input of Constants [1, -1] with Different Weights [0.004, 0.002]..... 41

Figure 5.5 Input of Constants [1, -1] with Different Weights [0.01, 0.03].........42

Figure 5.6 Input of Constants [1, -1] with Different Weights [0.3, 0.4]............43

Figure 5.7 Input of Pulses in the Complex HNN with Four Neurons 44

Figure 5.8 Input of Constants in the Complex HNN with Four Neurons.......... 45

Figure a The Simulation of the DFT with N = 16..48

Figure b The Simulation of the DFT with N = 32..49

CHAPTER ONE

Hopfield Neural Networks

1.1 Introduction

In the past few years, in order to solve difficult problems such as pattern recognition, researchers developed artificial neural networks that consist of massively interconnected neurons. John Hopfield (1982, 1984) introduced a recurrent neural network architecture for associative memories. He used it to demonstrate the computational properties of a fully connected network of units. The Hopfield Neural Network is an example of an associative learning network model. Such models store pattern vectors as memories, which can later be retrieved by providing corrupted versions of the stored pattern vectors. Associative memory associates or regenerates stored pattern vectors by means of specific similarity criteria. The purpose of the search is to output either one or all stored items that match the given search argument. The dynamics of neurons in Hopfield Nets can be described in both discrete and continuous space. These are discussed in the following sections. Hopfield and Tank (1985) showed that certain classes of optimization problems can be mapped and solved on the Hopfield model. They demonstrated the computational power and speed of their neural network model by (suboptimally) solving, among other problems, the classic NP-complete problem known as the Traveling Salesman Problem.

Hopfield and Tank have shown that if a proper topology is chosen and if interconnections for a given network is defined correctly, a solution can be obtained in reasonable time. Following the work of Hopfield, many researchers have used neural networks for signal processing such as bearing estimation (Luo & Bao, 1992; Rastogi et al., 1987) and maximum entropy deconvolution (Marrian & Peckerar, 1987). In these applications, high speed processing, or more precisely, real time processing, is desirable.

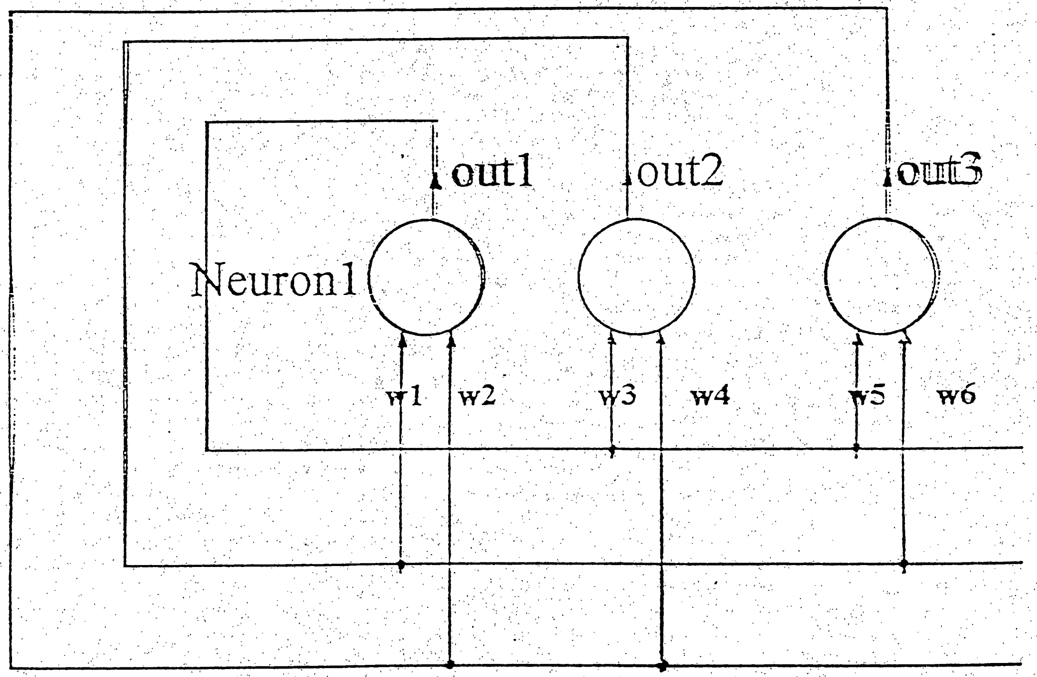


Figure 1.1 The Topology of Hopfield Neural Network (HNN)

The Hopfield Neural Network (HNN) model consists of nonlinear neurons which are fully connected with each other as shown in Figure 1.1. Out1, out2, out3 are the outputs of the neurons. The wi's are the weights which represent the interconnection strengths between neurons. The state of each neuron is determined by the outputs of neighboring neurons. The outputs generated in one step become the inputs during the next step. The dynamics of the neurons in HNN's can be described in both discrete and continuous (analog) space. The model designed for this thesis is described in continuous space.

There are certain aspects to consider when dealing with HNNs. If a HNN is used to solve a given problem, it usually has two major disadvantages. First, the output vector of the neurons is guaranteed to converge only to a local minimum instead of the global minimum, while the latter would be the best solution. Second, programming complexity when applying such HNN's to problems is usually very large. Considerable computation is invested in finding the correct neural interconnection strengths (weights) from the given data set of the problem to be solved (Dayhoff, 1990). Another issue involved is the choice of the proper energy function and activation function.

The programming complexity (Takeda et al., 1986) of a neural network was defined as the number of arithmetic operations that must be performed in order to determine the proper interconnection strengths (weights) for the problem to be solved. In conventional digital computers, once a program is compiled and stored in memory, it can be used on many different sets of input data. In neural networks, "learning" means finding a proper set of weights that result in the desired behavior of a neural network. The program and data are generally mixed and stored in the weights. A set of weights corresponds to a specific instance of a problem. Much time is thus invested to find the proper weights. For this reason, the concept of programming complexity becomes significant in the realm of neural networks.

It is not meaningful to compare the efficiencies of conventional digital computers to neural computers. Basically, they are two different kinds of machines. A digital computer can give an exact solution better than a neural computer can. Although neural networks are programmed in consideration of performing the proper number of operations specified by the programming complexity, they are not guaranteed to provide an exact solution. Another difference is that it is necessary to re-determine the interconnection strengths each time we use new data sets.

1.2 Discrete-step HNN

The discrete-time Hopfield network, being a feedback network, is characterized by a binary output vector at any given instant of time. The interactions of the processing elements cause the energy function to converge to a local minimum, which could be one of the desired solutions. In figure 1.2, a single-layer feedback neural network is shown, $i_1,\ldots,i_n$ are the external inputs, w_{ij} are the weights , and v_i are the outputs.

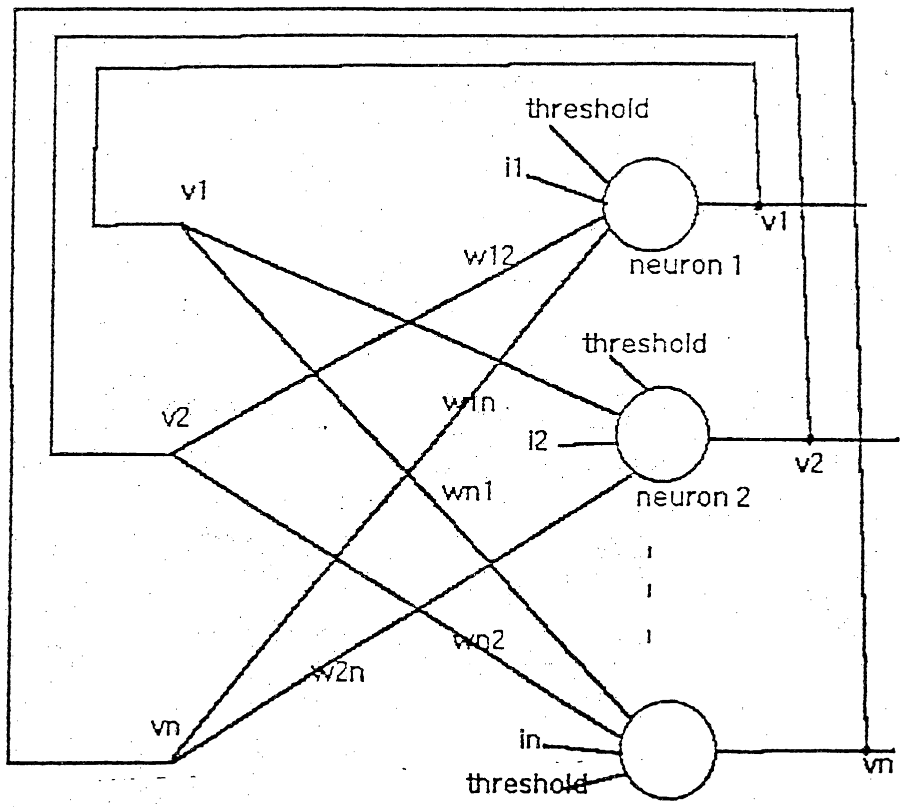


Figure 1.2 Single-layer Feedback Neural Network

1.2.1 The Energy Function

The energy function in a HNN provides us insight into the behavior of the network. It is used in proving that the network converges. The energy function is usually defined in n-dimensional output space v^n. The scalar-valued energy function for a discrete HNN has a quadratic form:

$$E = -1/2 v^t \mathrm{W} v + \theta^t v \qquad (1.1)$$

where θ is the threshold vector

W is the connectivity weight matrix

and v is the output vector.

In the expanded form, it is equal to

$$\mathbf{E} = -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij} v_i v_j + \sum_{i=1}^{n} \theta_i v_i \, , \; i \neq j \qquad (1.2)$$

The output of each neuron v_i is updated according to this formula:

$$v_i^{n+1} = \begin{cases} 1, & \text{if } \sum_j w_{ij} v_j^n - \theta_i > 0 \\ v_i^n, & \text{if } \sum_j w_{ij} v_j^n - \theta_i = 0 \\ -1, & \text{if } \sum_j w_{ij} v_j^n - \theta_i < 0 \end{cases} \qquad (1.3)$$

The network eventually converges, i.e. the outputs no longer change. The proof of convergence is following: The equation 1.2 can be expanded into

$$\mathbf{E} = -\frac{1}{2}\{\, v_k(\sum_j w_{kj} v_j + \sum_i v_i w_{ik}) + (v_k)^2 w_{kk} + \sum_{i=k}\sum_{j=k} v_i w_{ij} v_j \,\} + \theta_k v_k + \sum_{j=k} \theta_j v_j \tag{1.4}$$

So, when a new state is generated, the new energy becomes

$$\mathbf{E}^{\text{new}} = -\frac{1}{2}\{\, v_k^{\text{new}}(\sum_j w_{kj} v_j + \sum_i v_i w_{ik}) + (v_k^{\text{new}})^2 w_{kk} + \sum_{i=k}\sum_{j=k} v_i w_{ij} v_j \,\} + \theta_k v_k^{new}$$

$$+ \sum_{j=k} \theta_j v_j \tag{1.5}$$

$$\Rightarrow \Delta E_k = E^{\text{new}} - E$$

$$= -\frac{1}{2}\{\, (v_k^{\text{new}} - v_k)(\sum_j w_{kj} v_j + \sum_i v_i w_{ik}) + [(v_k^{\text{new}})^2 - (v_k)^2]\, w_{kk} \,\} + \theta_k(v_k^{\text{new}} - v_k) \tag{1.6}$$

since $\mathbf{w}_{ij} = \mathbf{w}_{ji}$, and $\mathbf{w}_{ii} = 0$

$$\therefore \quad \Delta E_k = -(v_k^{\text{new}} - v_k)(\sum_j w_{kj} v_j) + \theta_k(v_k^{\text{new}} - v_k)$$

$$= -(v_k^{\text{new}} - v_k)(\sum_j w_{kj} v_j - \theta_k)$$

suppose if $v_k^{\text{new}} = 1$, by equation 1.3

if $\sum_j w_{ij} v_j^n - \theta_i < 0$, and v_k equal to either 1 or -1, then $(v_k^{\text{new}} - v_k) \leq 0$

$$\Rightarrow \Delta E_k \leq 0$$

suppose if $v_k^{\text{new}} = -1$, by equation 1.3

if $\sum_j w_{ij} v_j^n - \theta_i > 0$, and v_k equal to either 1 or -1, then $(v_k^{\text{new}} - v_k) \geq 0$

$\Rightarrow \Delta E_k \leq 0$

From the above analysis, we can conclude that no matter what the new state of v_k is, 1 or -1, the Liapunov energy function always decreases, and since there is a lower bound, it converges to a local minimum.

1.3 Continuous HNN

Hopfield has also described a continuous-variable version of the binary-valued associative memory (1984). In this model, the output node (neuron) is uniquely determined by the instantaneous input to the node; that is,

$$v_{\text{output}} = g_i(u_i)$$

$$\Rightarrow \quad u_i = g_i^{-1}(v_{\text{output}})$$

where the output is a continuous and monotonically increasing function of the instantaneous input. Typically, the input-output relation $g_i(u_i)$, is a sigmoidal curve. For example, $g_i(u_i)$ might be $g_i(u_i) = 1 / (1+ exp(-u_i))$.

Continuous-time HNN perform similarly to discrete time networks. Time is assumed to be a continuous variable in continuous-type networks. In continuous HNN, continuous activation functions are necessary. Updating neurons occurs continuously in time with every network output. The continuous-type HNN converge to one of the stable minima in the state space. Evolution of this network model is in the general direction of negative gradient for the energy function. In applications, typically the energy function is made equivalent to a certain objective function that needs to be minimized, leading to an output which is close to the desired solution.

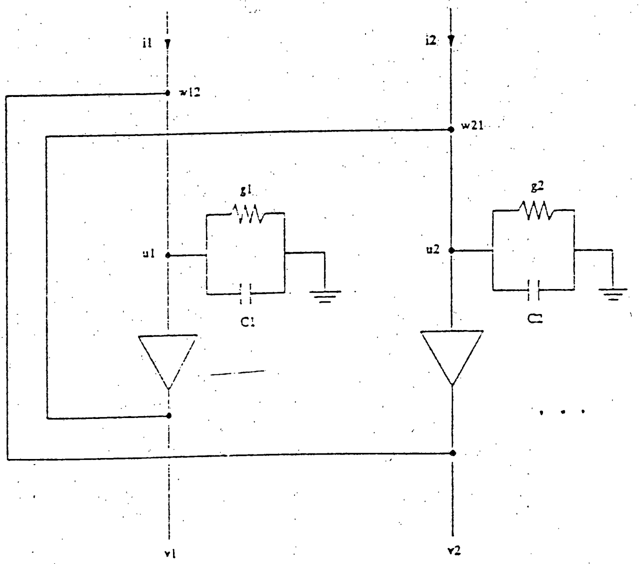


Figure 1.3 The Neural Network Model Using Electrical Components

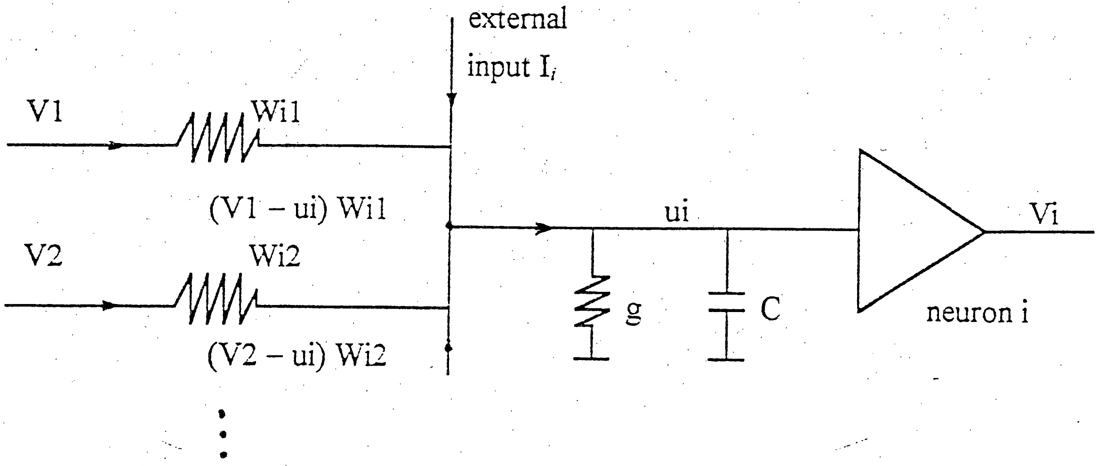


Figure 1.4 The Input Node of the i'th Neuron

In the continuous-valued Hopfield Net, the architecture of the network is specified so that changes in time are described continuously, rather than as discrete update times for individual units. With some inspiration from electrical circuits, the processing units are governed by the equation (Dayhoff, 1990):

$$C_i(du_i \;\; dt) = \sum_i w_{ij} v_j - (u_i / R_i) + I_i \qquad (1.7)$$

$$u_i = g_i^{-1}(v_i)$$

where C_i is a constant, R_i is controls unit j's decay resistance ($R_i > 0$), I_i is external input to unit i, and V_i is the output of unit i after the activation function is applied. W_{ij} is the weight between neuron i and neuron j. $w_{ij} v_j$ is the current which will be generated.

Electrically, $w_{ij} v_j$ might be understood to present the electrical current input to cell i due to the present potential of cell j. The term $(-u_i / R_i)$ is current flow due to finite transmembrane resistance R_i, and it causes a decrease in u_i. I_i is any other fixed input current to neuron i.

1.3.1 The Energy Function for Continuous HNN

The Liapunov function of interest for the continuous HNN is

$$E(v) = -1/2 \sum_{ij} \sum_{j} w_{ij} v_i v_j + \sum_i I_i v_i + \sum_i (1/R_i) \int_0^{v_i} g_i^{-1}(v) dv \qquad (1.8)$$

The proof of convergence follows: For symmetric w_{ij}, the time derivative is

$$dE / dt = -\sum_i (dv_i/dt)[\sum_i w_{ij} v_j - (u_i / R_i) + I_i] \qquad (1.9)$$

The quantity in square brackets is the right hand side of (1.7), so

$$dE / dt = -\sum_i (dv_i/dt) C_i \, dv_i/dt \tag{1.10}$$

or

$$dE / dt = -\sum_i C_i \, g_i^{-1'} (v_i) \, (dv_i/dt)^2 \tag{1.11}$$

because $g_i^{-1'}(v_i) \equiv \partial g_i^{-1} / \partial v_i$ is nonnegative, and so are C_i and the term $(dv_i /dt)^2$. Accordingly,

$$dE / dt \le 0, \text{ and } \quad dE / dt = 0 \text{ implies } dv_i/dt = 0 \quad \text{for all } i.$$

Since E is bounded, equation (1.11) shows that the system moves in state space toward lower and lower values of E, and comes to rest at one of the minima. The continuous-valued Hopfield net is thus shown to be globally stable. The memorization of patterns, however, is more complex than in the case of the binary-valued model. The stable points. The memorized patterns are determined not only by the prescription

$$w_{ij} = v_i v_j$$

but also by the shape of $g(u)$ and by the value of R_i.

CHAPTER TWO

Complex Neural Networks

2.1 Introduction

Complex-domain neurons are sometimes preferred in modeling since they are able to capture more complexity. For example, many signals are best described with complex values, such as in electric circuits, oscillators, and radar signals. Takeda and Goodman (1986) suggested Hopfield neural networks using complex numbers; Little et al. (1990)

discussed the complex weights of the backpropagation neural network; Georgiou (1992) discussed the activation functions of complex-domain neural networks; Georgiou and Koutsougeras (1992) derived the complex domain backpropagation so that it can accommodate suitable activation functions; a circuit implementation of the corresponding neuron with the complex domain was also proposed in the same paper.

Generally speaking, a complex-domain neuron will be more complex than a real-domain one. When extending a neural network from the real domain to the complex domain, certain problems arise. The energy and activation functions may not be suitable in the complex domain because singularities may be introduced when the domain is extended to the complex plane. For example, activation functions in the real domain such as $h(x) = 1/(1+exp(-x))$ and $\tanh(x)$, are undefined in the complex domain when $x= i\,(2n + 1)\pi$, $n = 0, 1, 2, \ldots$

Thus, it is necessary to use activation functions which can properly handle complex numbers. Three suitable activation functions for neural networks in the complex domain have been discussed in (Georgiou, 1992), which are the following:

1. $\boldsymbol{f_1(z) = rz / (rc + |z|)}$, r and c are positive real constants, which could both be taken to be 1. The property of this function is mapping a point $z = x + iy = (x, y)$ on the complex plane to a unique point $f(z) = (rx / (c+|z|), ry / (c+|z|))$ and on the open disc $\{z: |z|<r\}$. The phase angle of z is the same as that of $f(z)$. This function is suitable for complex domain backpropagation, as well as the complex HNN.

2. $f_2(z) = z / |z|$, with $f_2(0) = 0$. This function normalizes the magnitude of z to unity, while the phase remains the same. It limits the output of a neuron by having only phase information which can be exploited into having ultra-fast optical implementations (Noest, 1988). This function is used in the Complex Perceptron (Georgiou, 1992). Its use is also possible in the complex domain backpropagation. This function is the one used in this work.

3. $f_3(z) = \sigma^k$ *if* $|\arg(z/\sigma^k)| < \pi/q$, where $\sigma^k = e^{2\pi ki/q}$ and k = 0, 1, 2, ,.., $(q-1)$. In other words, the output of the neuron f_3(z) belongs to the set of the q roots of unity, which are evenly distributed over the unit circle on the complex plane. In arg(x), the phase angle x is taken to be in the interval $(-\pi, \pi]$. This function is used in the discrete version of the Complex Perceptron.

The validity of these functions is asserted by their applicability to three conceptually different models: perceptron, Backpropagation, and Hopfield models.

2.2 Implementations

In developing complex neural networks, several studies have mentioned the complex weights and activation functions with complex domain. For example, the backpropagation neural network learning algorithm (Little et al., 1983) is generalized to include complex-valued weights for possible optical implementation. In their approach, the outputs and the weights of neurons are allowed to be complex. A complex-valued neural network (Yang et al.,1994) for solving the direction of arrival estimation which can be solved by the maximum likelihood or linear prediction methods was presented. It shows that the processing time is greatly reduced so that the network can update the array weights very rapidly to accommodate any new arriving signals. Different generalizations of backpropagation (Birx and Pipenberg, 1989; Georgiou, 1992) for complex-valued weights were described. These papers suggested ways to define the weights and activation functions of complex neural networks.

Many phenomena in nature, engineering, and science are described best by using such complex-domain models. Examples include electric circuits, oscillators, and the processing of various signals such as Fourier Transformations. The difficulties of developing a complex neural network are in finding the weights and activation functions. Different kinds of problems require finding different kinds of weights and activation functions. It could possibly take a long time to find the proper weight and suitable activation function for a given problem.

2.3 Complex HNN

2.3.1 Discrete-step Complex HNN

In the discrete-step complex Hopfield net, the weight matrix is Hermitian, i.e. $w_{ji} = \overline{w_{ij}}$. A Liapunov function is:

$$E = -1/2 \sum_{j=1}^{N}\sum_{i=1}^{N} \overline{f_i}\, w_{ji} f_j$$

where f is an activation function.

Georgiou (1992) suggested that the activation functions f_2 and f_3 can be used in this discrete complex HNN. The stable memory vectors $V^{(1)}, V^{(2)}, \ldots, V^{(p)}$ are stored using the complex version of Hebbian learning (Georgiou, 1992):

$$w_{ji} = \sum_{s=1}^{p} V_j^{(s)} \overline{V_i^{(s)}}.$$

2.3.2 Continuous Complex HNN

The continuous complex Hopfield neural network (Szilagyi et al., 1990) is characterized by N coupled differential equations $dz_i / dt = \sum_{j=1}^{N} w_{ij} f_j (z_j(t)) + I_i$,

where I_i is the complex constant input to neuron i, and the complex variable z is what is commonly called the net input, or the weighted sum of the inputs, of a particular neuron. It is expressed as $z = x + iy$, where x and y are its real and imaginary parts. Likewise, u and v are the real and imaginary parts of the output of the activation function: $f(z) = u(x, y) + iv(x, y)$. It was shown (Szilagyi et al,1990) that function E is a Liapunov function, i.e. $dE / dt \leq 0$. $E = -1/2\sum_{i=1}^{N}\sum_{j=1}^{N} f_i W_{ij} f_j - \mathrm{Re}\ (\sum_{i=1}^{N} f_i I_i)$. It was proved (Georgiou, 1992) that activation functions for complex HNN satisfies the following condition:

$$dE / dt = -\sum_{j=1}^{N} (u_x (dx/dt)^2 + v_y(dy/dt)^2 + (dx/dt)(dy/dt)(u_y + v_x)) \leq 0.$$

Upon substituting the partial derivatives for f_2 we get

$$dE / dt = -\sum_{j=1}^{N} ((x\ dy/dt - y\ dx/dt)^2 / |z|^3) \leq 0 ,$$

which shows that f_2 can be used in the continuous complex Hopfield net.

The complex HNN was not thus far implemented as a circuit in neither the frequency (complex) domain nor in the time (real) domain. In the following chapters, circuit components which correspond to complex weights are designed. These components are used in the circuit design of the complex HNN. Furthermore, the designed circuit is mapped into an equivalent circuit that operates in the time domain, where all quantities are real.

CHAPTER THREE

Implementations of Complex Neural Networks

A complex Hopfield neural network is able to take real as well as complex numbers as input and still converge. In a circuit, the complex weights can be represented as optical signals when both magnitude and phase information are included (Szilagyi et al., 1990). The implementation of hardware circuits and using them to solve a given problem can yield faster results than when relying on software simulations. The circuits which designed for the complex weights of neural networks can provide a faster solution than the traditional ones. The circuit developed includes: inductors, OP-Amps(Operational amplifiers), capacitors, and resistors. The designed weights were tested on a DFT (Discrete Fourier Transforms) circuit in implemented hardware. In the designed complex circuits, the same frequency ω is used throughout each circuit.

3.1 Implementation of Complex Weights

The role the weights play in a neural network is to connect two neurons properly. Extending the weights from the real domain to the complex domain, a method to simulate the complex weight and generate correct outputs must be found. Using SPICE, a simulation of the implementation of complex weights was used to design such circuits. Several circuit blocks which can synthesize different types of complex weights is designed.

3.1.1 The Weights in the Real Domain

Several circuits were designed to represent the real weights shown below, where R_1, R_2 ... R_N and R_L are resistors, V_1, V_2 ... V_N are inputs, and V_{output} is output. Op is an ideal OP-Amp.

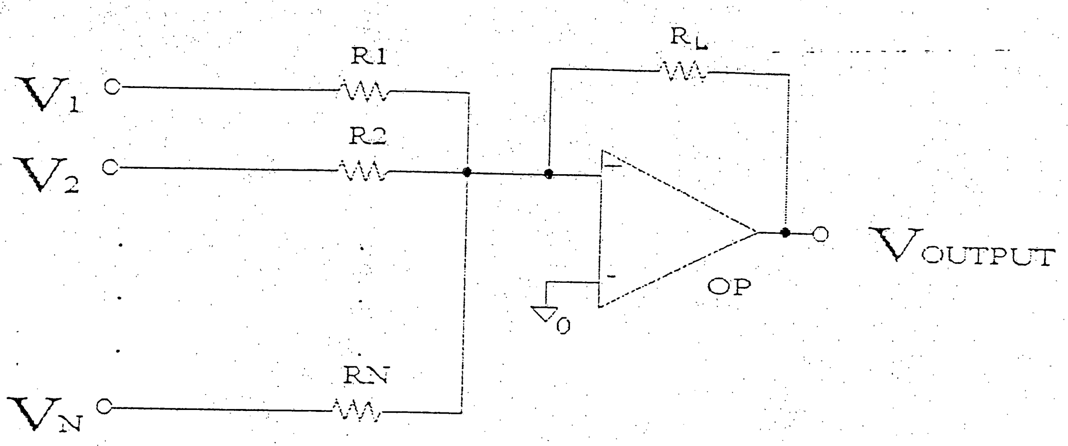


Figure 3.1 The Weights in the Real Domain

By setting the value of these resistors R_1, R_2,...., R_N, and R_L, the weight in a neural network can be simulated. The relationship between the input and the output in the circuit is similar to the one in the neural network.

According to the *Kirchoff's* rule,

$$V_{output} = -(\frac{R_L}{R_1})V_1 - (\frac{R_L}{R_2})V_2 - \dots\dots\dots -(\frac{R_L}{R_N})V_N \qquad (3.2)$$

Here, $-(\frac{R_L}{R_1}), -(\frac{R_L}{R_2}), \dots\dots\dots, -(\frac{R_L}{R_N})$ represent the weights.

In the matrix forms:

$$\text{Output} = \text{Input} \times \text{Weights}$$

$$\Rightarrow \left[Vo_0, Vo_1, Vo_2 \, \ldots\ldots\ldots\ldots \, Vo_{N-1}\right]$$

$$= [V_{i0}, V_{i1}, V_{i2} \ldots V_{i(N-1)}] \times \begin{bmatrix} W_{00}, & W_{01}, \ldots & W_{0(N-1)} \\ W_{10}, & W_{11}, \ldots & W_{1(N-1)} \\ W_{20}, & W_{21}, \ldots & W_{2(N-1)} \\ & \vdots & \\ W_{(N-1)0}, & W_{(N-1)1}, \ldots & W_{(N-1)(N-1)} \end{bmatrix}$$

where V_{O0}, $V_{O1}, \ldots, V_{O(N-1)}$ are the output vectors, V_{i0}, V_{i1}, $V_{i2} \ldots V_{i(N-1)}$ are the input vectors, W_{00}, $W_{01}, \ldots, W_{(N-1)(N-1)}$ are the weights and N is the dimension. The relationship between the input and the output in the circuits can be represent as the weights in neural networks.

3.1.2 An Inverter

An inverter is used in the representing of negative weights in the circuit design. Negative weights are simulated by having negative input to positive resistance. Thus, negative resistance is avoided in the design. As shown in Figure 3.2,

$$V_{\text{output}} = -(R/R)V_{\text{netinput}} = -V_{\text{netinput}}$$

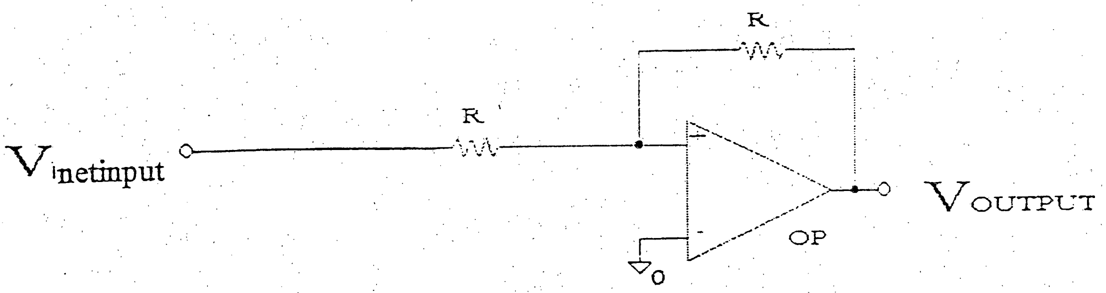


Figure 3.2 The Circuit of An Inverter

3.1.3 The Weights in the Complex Domain

Three basic elements are used to construct the entire circuit representing the weights.

1. The Type ***a+bi*** of Complex Weights, where *a* and *b* are positive real numbers. C in figure 3.3 is a capacitor.

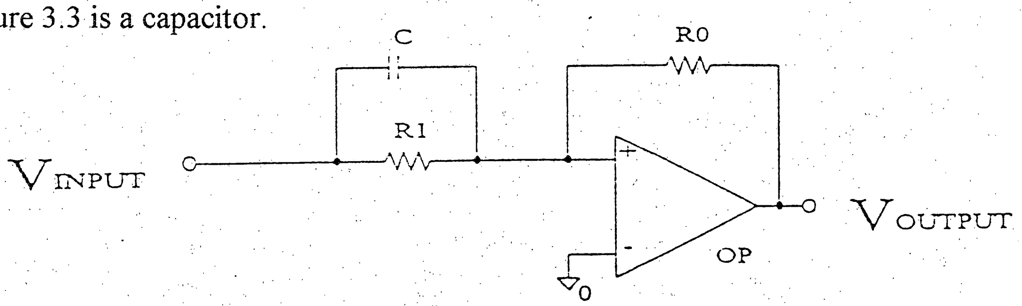


Figure 3.3 The Weights $a + bi$ in the Complex Domain

According to the *Kirchoff's* rule: $V_{output} = -\,(R_0/R_1 + i\omega R_0C)\,V_{input}$

where $\omega = 2\pi f$, and f is the frequency. (3.3)

So we know if the weight = $(a + bi)$, an inverter is used to obtain a negative input V_{INPUT}, and proper values of R_0, R_1, C, and ω are chosen so that $a = R_0 / R_1$, $b=\omega R_0 C$.

2. The Type **a-b*i*** of Complex Weights. When the weight is represented by $a - bi$, where *a* and *b* are positive real numbers, another circuit to represent $a - bi$ is used. Here, an inductor L is used.

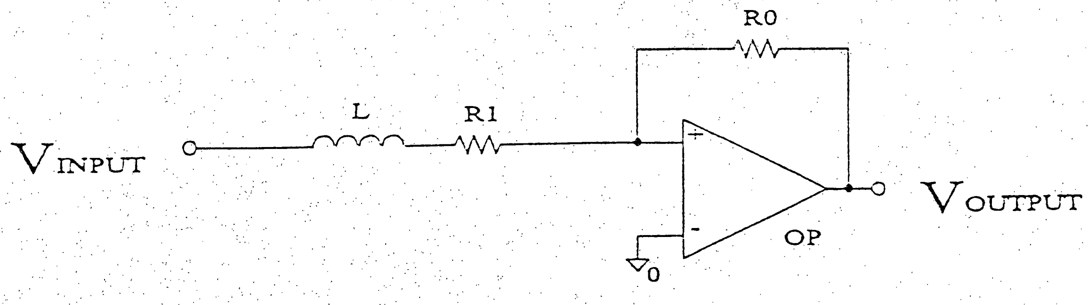


Figure 3.4 The Weights $a - bi$ in the Complex Domain

According to the *Kirchoff's* rule:

$$V_{output} = -R_0 / (R_1 + i\omega L)\, V_{input}$$

$$= -R_0\,(R_1 - i\omega L) / (R_1^2 + \omega^2 L^2)\, V_{input}$$

$$= -((R_0 R_1 / (R_1^2 - \omega^2 L^2) - iR_0\omega L / (R_1^2 + \omega^2 L^2))\, V_{input}$$

So, if the weight = $(a - bi)$, an inverter is used to obtain a negative input V_{INPUT}, and proper values of R_0, R_1, C, and ω are chosen so that

$$a = R_0 R_1 / (R_1^2 + \omega^2 L^2),\ b = R_0 \omega L / (R_1^2 + \omega^2 L^2) \qquad (3.4)$$

A passive circuit that implements type *a-bi* appears in figure 3.6. It can be used when $0 < a < 1$.

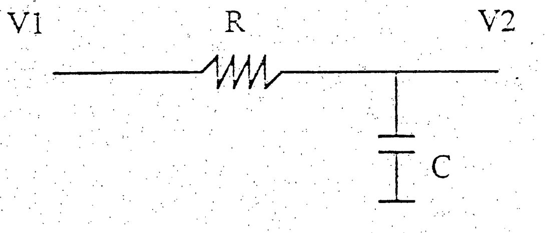


Figure 3.5 Another Simulation for the Complex Weight $a - bi$

In Figure 3.5, according to the *Kirchoff's* rule:

$$V2 / V1 = (1 / i\omega C) / (R + (1 / i\omega C))$$

$$= (1 - i\omega RC) / (1 + (\omega RC)^2)$$

$$\text{so, } a = 1 / (1 + (\omega RC)^2), \quad b = \omega RC / (1 + (\omega RC)^2).$$

3.2 An Example Using Complex Weights

Signals are expressed as a function of time in the time-domain representation. In the 18th Century, J. B. Fourier showed that almost any signal can be expressed as a sum of sinusoids of various frequencies. In signal processing, the Fourier Transform is a very powerful mathematical tool for understanding, analyzing, and solving problems such as filtering (Elliott and Rao,1982).

Engineering problems often require information about the spectral content of signals. The Fourier Transform, $X(f)$ of a continuous signal $x(t)$, is (Limited, 1989):

$$X(f) = \int_{\infty}^{-\infty} x(t)\, e^{-2\pi i f t}\, dt \qquad (3.5)$$

t : time

f: frequency.

$X(f)$ is called the spectrum of the signal $x(t)$. By sampling the time and frequency variables and limiting the computations to a finite set of data points, the continuous-time Fourier Transform can be made suitable for digital computation. This modified version of the Fourier Transform is usually referred to as DFT. The input of DFT is a sequence of numbers rather than a continuous function of time $x(t)$. The sequence of numbers, referred to as a discrete time signal, usually result from the periodical sampling of the continuous signal $x(t)$. The DFT equation for the evaluation of $X(k)$ gives,

$$X(k) = \frac{1}{N}\sum_{n=0}^{N-1} x(n)\, e^{-2\pi ikn/N} \qquad (3.6)$$

$k = 0, 1, 2, \ldots. N\text{-}1$

using matrix form, it becomes

$$\mathrm{X} = \mathrm{x\,W} \qquad (3.7)$$

where

$$\mathrm{X} = \begin{bmatrix} X_0 & X_1 & X_2 & \ldots\ldots\ldots\ldots & X_{N-1} \end{bmatrix}$$

$$\mathrm{x} = \begin{bmatrix} x_0 & x_1 & x_2 & \ldots\ldots\ldots\ldots & x_{N-1} \end{bmatrix}$$

$$\mathrm{W} = \begin{bmatrix} W_{00} & W_{01} & \cdots\cdots & W_{0N-1} \\ W_{10} & W_{11} & \cdots\cdots & W_{1N-1} \\ . & . & & . \\ . & . & & . \\ . & . & & . \\ W_{N-10} & W_{N-11} & \cdots\cdots & W_{N-1N-1} \end{bmatrix}$$

where $W_{kn} = e^{-2\pi ikn/N}$

W is the weight matrix, which is fixed:

$$\mathrm{Weight} = e^{-2\pi ikn/N} = \cos(2\pi kn/N) + i\cdot\sin(2\pi kn/N) \qquad (3.9)$$

$n = 0, 1, \ldots. N-1;$

$k = 0, 1, \ldots N\text{-}1;$

N: dimension of the matrix.

Since the traditional way of computing DFT is software sequential processing. It is unable to operate on the entire vector of sampled data simultaneously (Culhane and

Peckerar, 1989). A circuit in hardware could operate on the entire vectors at the same time and thus overcome this disadvantage. First, the 2-dimensioned (N=2) DFT circuit is designed.

The weight matrix $= \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$, and the circuit is shown in Figure 3.6.

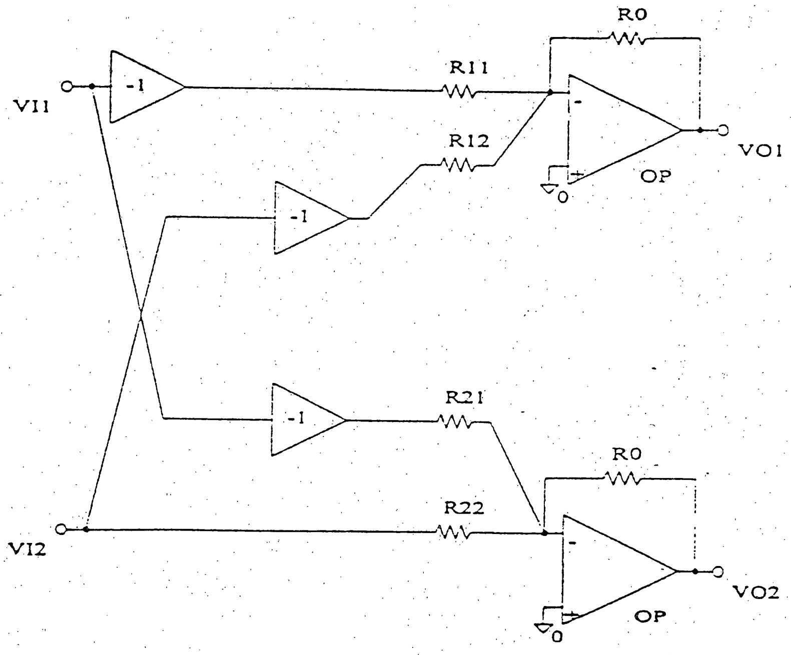


Figure 3.6 The Circuit with **N** = **2**

If **N=4**, the weight matrix $= \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -i & -1 & i \\ 1 & -1 & 1 & -1 \\ 1 & i & -1 & -i \end{bmatrix}$, and the circuit is shown in Figure 3.7.

If **N=8,** the weight matrix $= \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & \frac{1-i}{\sqrt{2}} & -i & \frac{-1-i}{\sqrt{2}} & -1 & \frac{-1+i}{\sqrt{2}} & i & \frac{1+i}{\sqrt{2}} \\ 1 & -i & -1 & i & 1 & -i & -1 & i \\ 1 & \frac{-1-i}{\sqrt{2}} & i & \frac{1-i}{\sqrt{2}} & -1 & \frac{1+i}{\sqrt{2}} & -i & \frac{-1+i}{\sqrt{2}} \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & \frac{-1+i}{\sqrt{2}} & -i & \frac{1+i}{\sqrt{2}} & -1 & \frac{1-i}{\sqrt{2}} & i & \frac{-1-i}{\sqrt{2}} \\ 1 & i & -1 & -i & 1 & i & -1 & -i \\ 1 & \frac{1+i}{\sqrt{2}} & i & \frac{-1+i}{\sqrt{2}} & -1 & \frac{-1-i}{\sqrt{2}} & -i & \frac{1-i}{\sqrt{2}} \end{bmatrix}$,

and the circuit is shown in Figure 3.8. The previously designed complex weights are used in these circuits.

In order to obtain the weights easily, we set the value of the resistor $R_0 = 10^3 = 1K\Omega$, $\omega = 2\pi * 10^3 = 2\pi K$. Let us consider the type $a + bi$ of a complex weight, from the equation (3.3), we know that

$$a = -\frac{1}{R_1}$$

$$\Rightarrow R_1 = -\frac{1}{a}(K\Omega)$$

$$b = -2\pi \cdot 1K \cdot 1K \cdot C$$

$$\Rightarrow C = -\frac{b}{2\pi}(\mu F).$$

For the type $a\text{-}bi$, from the equation (3.4),

$$a = -\frac{R_0 R_1}{R_1^2 + \omega^2 L^2} \text{ and } b = -\frac{R_0 \omega L}{R_1^2 + \omega^2 L^2}$$

$$\text{since } a^2 + b^2 = R_0^2 = 10^6 = 1M$$

$$\Rightarrow R_1 = aK \ , \ \ L = \frac{b}{2\pi}$$

$$\text{if } \frac{R_0}{R_1^2 + \omega^2 L^2} = 10^{-3} \text{ is given.}$$

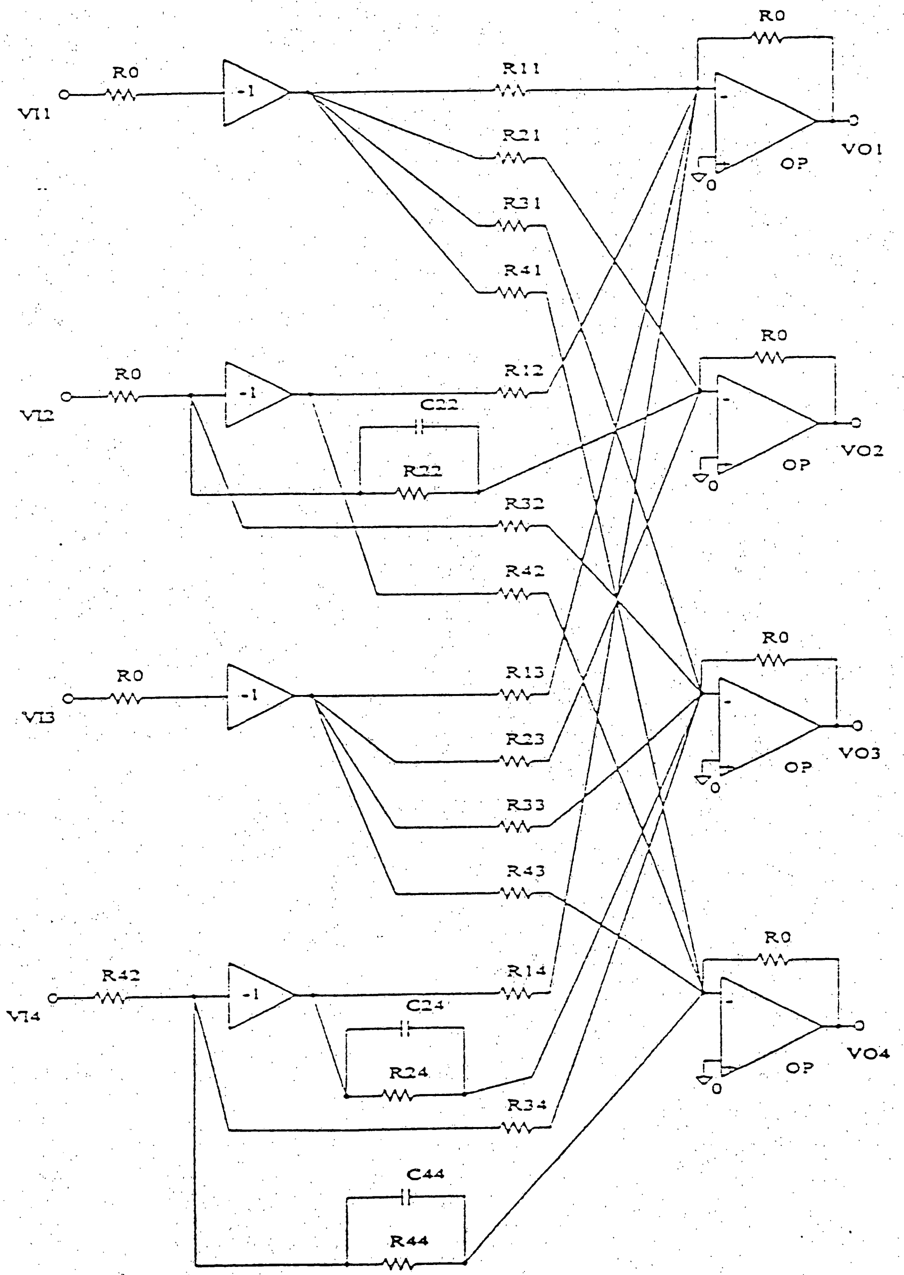


Figure 3.7 The Circuit with N=4

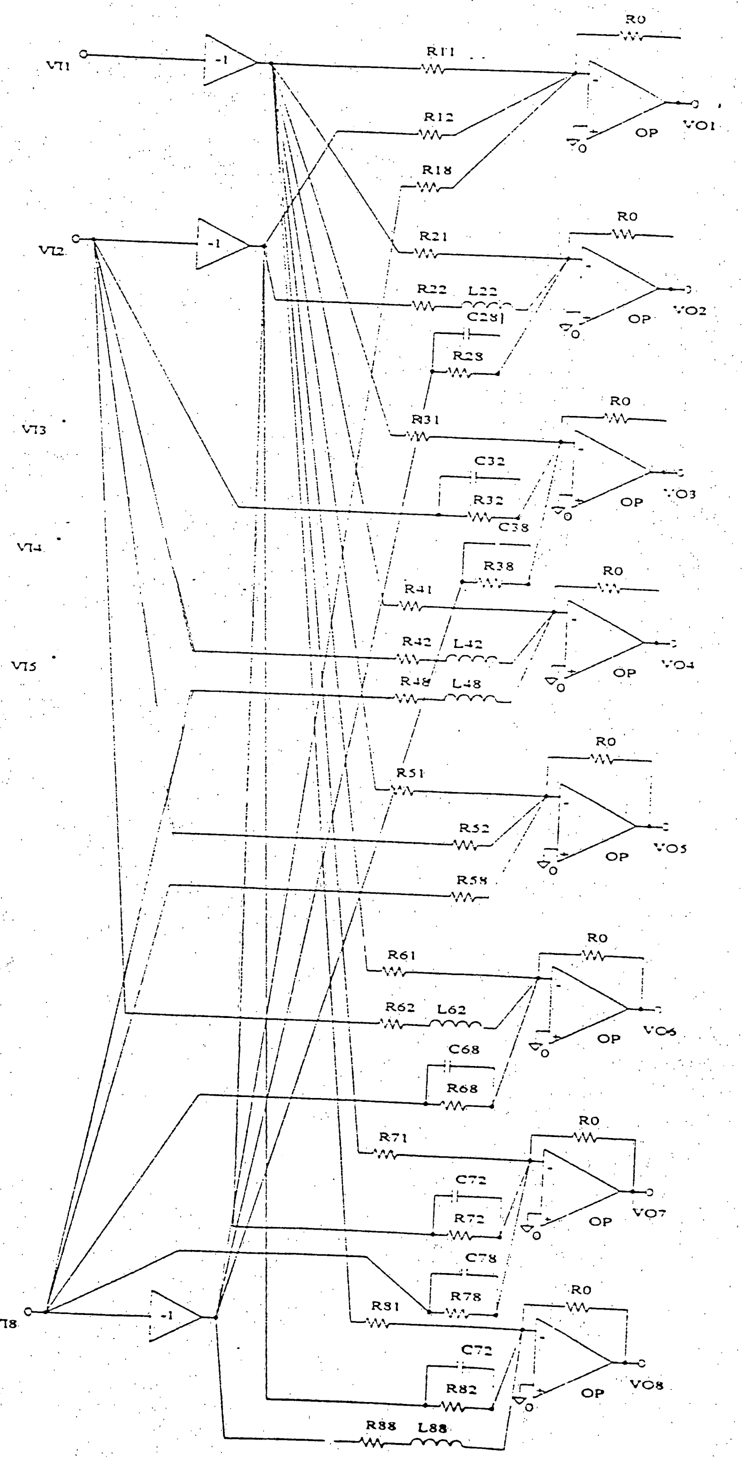


Figure 3.8 The Circuit with **N**= 8

3.3 Results of the DFT Simulation

3.3.1 Inputs

The dimension of the input data for DFT circuits are N=2, 4, 8, 16, and 32 respectively. The data to be transformed is a simple unit down-step shown in Fig 3.9. The first half is 1 Volt (high) and the other half is 0 Volt (low).

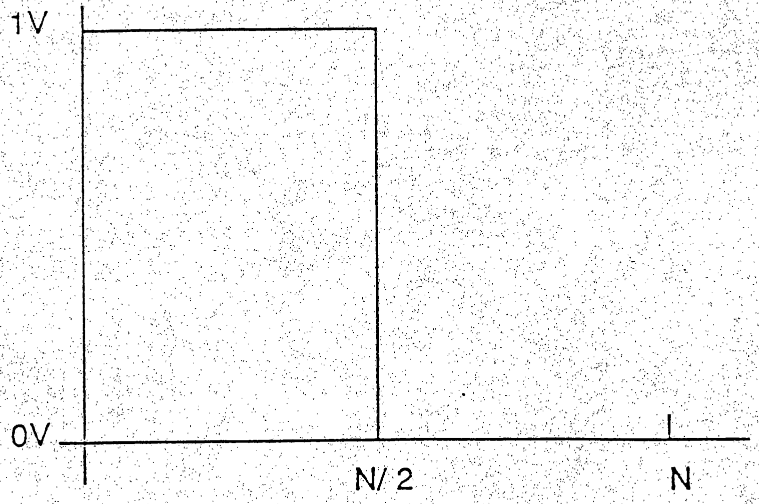


Figure 3.9 The Input Data of Simulating Complex Weights

3.3.2 Outputs

Figure 3.10a, 3.10b, 3.10c are the results of simulating the DFT's. The upper figures are the true output from traditional DFT computing (software). The lower figures are the simulating results which generated from the designed circuits (hardware). Their results are the same. It shows that the designed circuits correctly simulate the complex weights.

In figure 3.10a, the upper plots are the output from DFT program; the lower plots are the result from the circuit simulation. The plots on the left side are the real parts; and the right side are imaginary parts. Both results are identical.

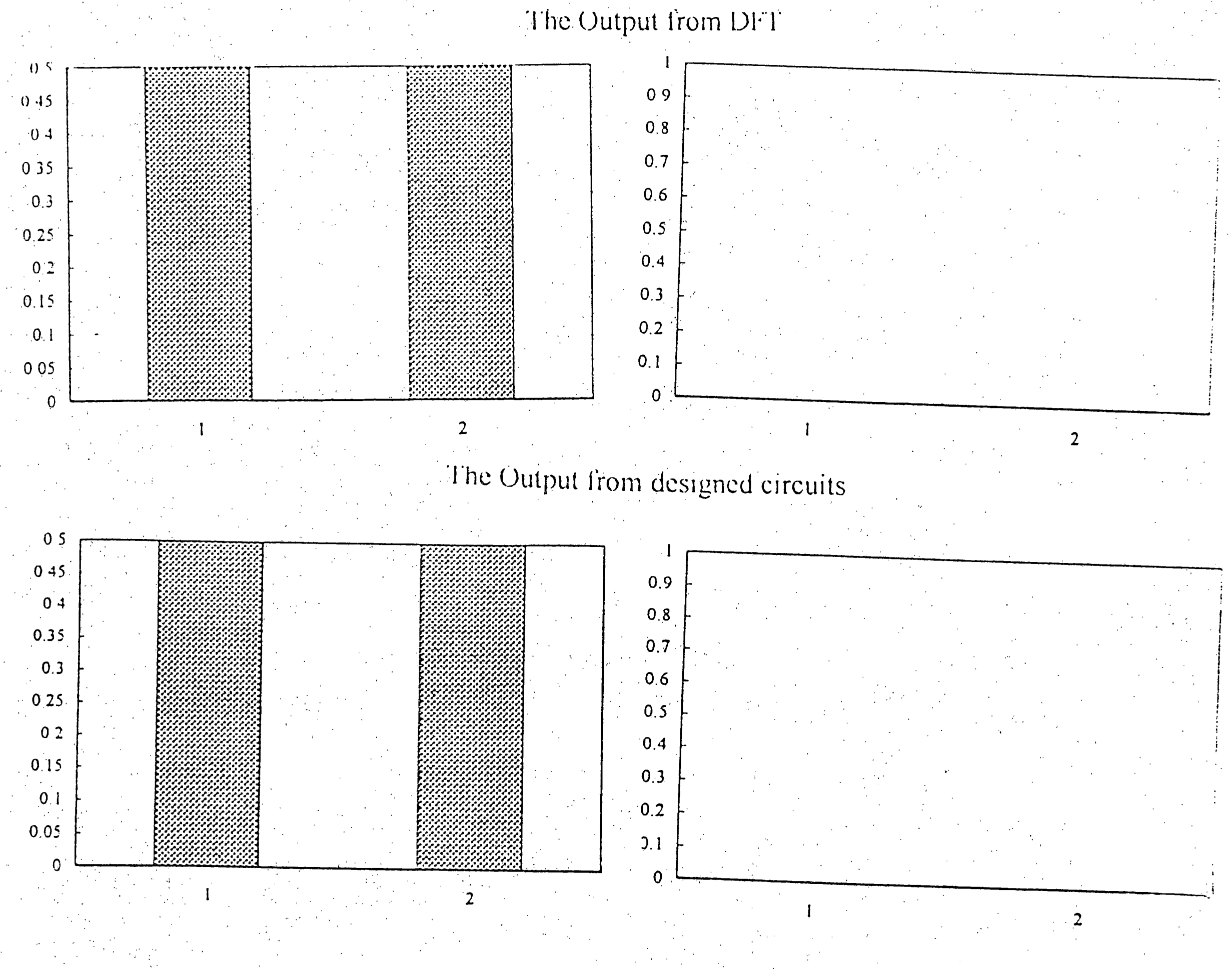


Figure 3.10a The Output of Simulating the Complex Weights (N=2)

In figure 3.10b, the upper plots are the output from DFT program; the lower plots are the result from the circuit simulation. The plots on the left side are the real parts; and the right side are imaginary parts.

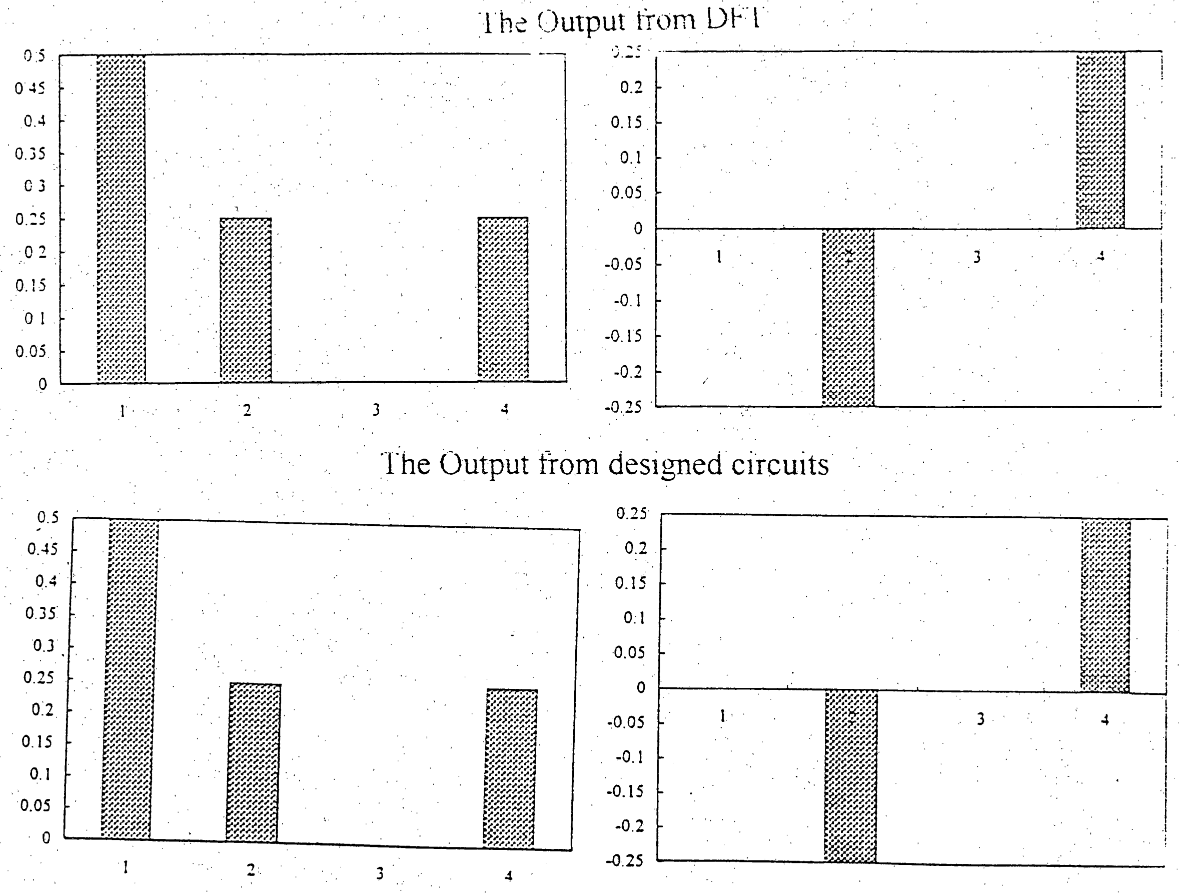


Figure 3.10b The Output of Simulating the Complex Weights (N=4)

In figure 3.10c, the upper plots are the output from DFT program; the lower plots are the result from the circuit simulation. The plots on the left side are the real parts; and the right side are imaginary parts. The designed circuit was tested up to N = 32. (see

Appendix A). These results showed that the output which was generated from the circuits simulation is identical to the actual output from a DFT computer program.

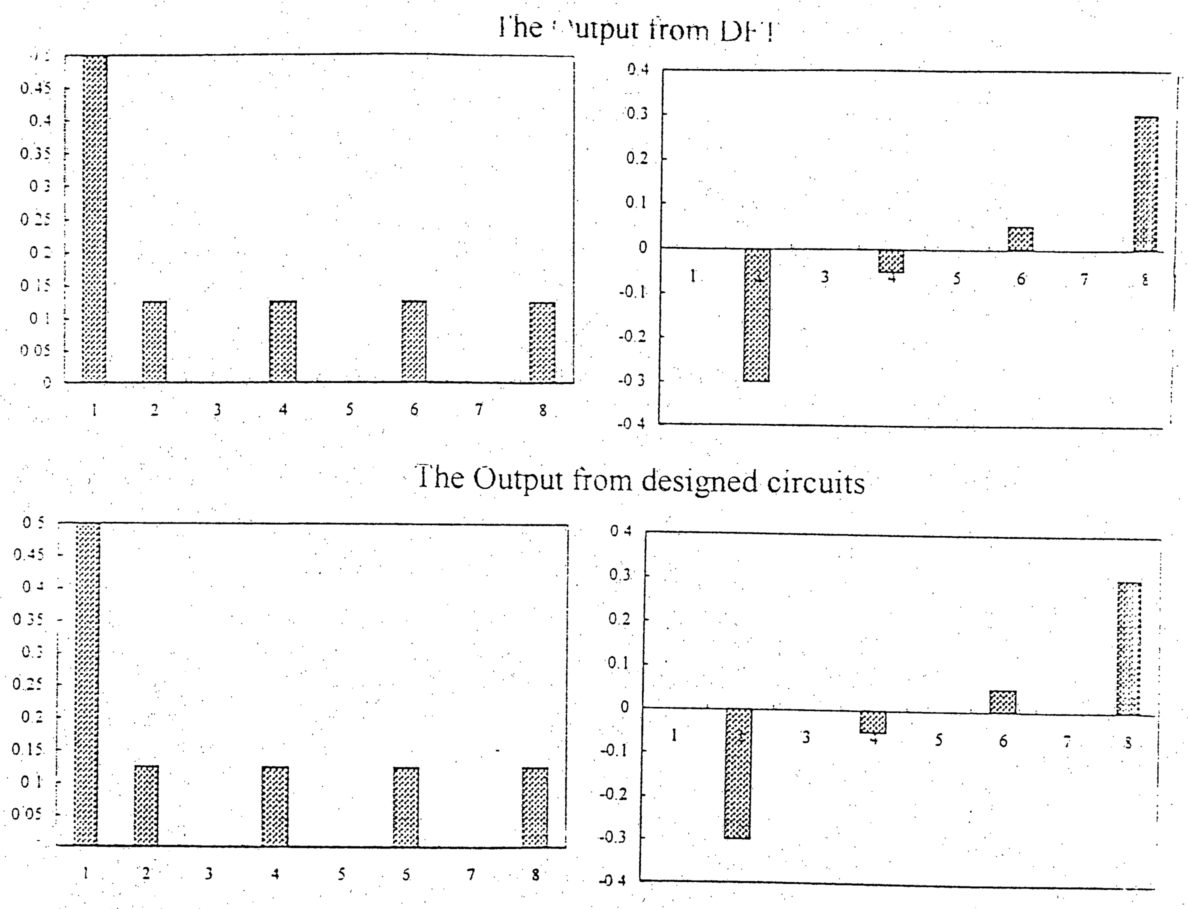


Figure 3.10c The Output of Simulating the Complex Weights (N=8)

CHAPTER FOUR

Decomposing the Complex HNN into a Real Coupled One

4.1 The Limitation of SPICE

Using the circuits designed for the complex weights, a complex HNN with two neurons is shown in Figure 4.1. As mentioned before, there are suitable activation functions for neurons in the complex domain. In the circuit design, the magnitude of a wave form is represented by the absolute value, which is part of the activation function used. The problem of using SPICE to simulate the complex HNN is that the current version of SPICE cannot determine the instantaneous magnitude of a complex voltage. Therefore, another method to represent the complex HNN is needed, in order to be simulated.

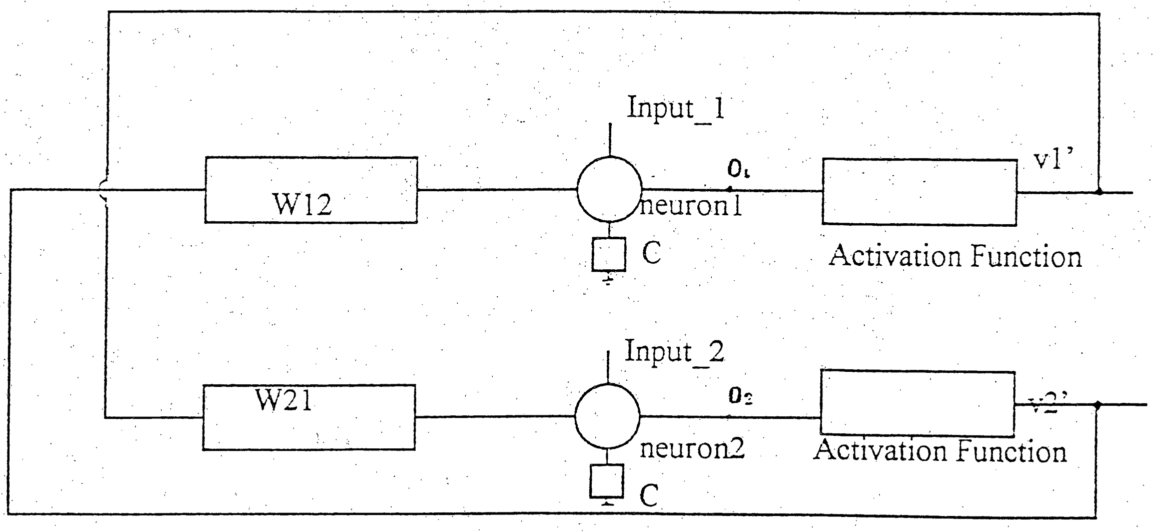


Figure 4.1 Complex HNN with Two Neurons

In figure 4.1, a two-neuron complex HNN is shown. W_{12} and W_{21} are the complex weights. Input_1 and input_2 are external inputs, $V_{1'}$ and $V_{2'}$ are outputs. C is a capacitor. and the activation functions are properly designed for the complex domain.

4.2 Decomposing the Complex HNN into a Real Coupled One

The complex HNN is decomposed into one in which every complex neuron is replaced by two real coupled neurons. Figure 4.2 shows a complex neuron I with output

$$Vi = O / \sqrt{Re(O)2 + Im(O)2}$$

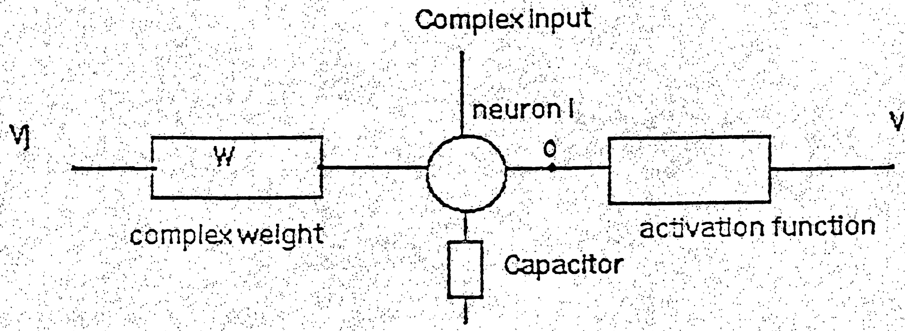


Figure 4.2 The i'th Complex Neuron

The equivalent of neuron i in the real domain is shown in figure 4.3. It consists of two real coupled neurons. In figure 4.3, W_R is the real part and W_I is the imaginary part of the complex weight W. Input_R is the real part and input_I is the imaginary part of the external input. V_j_R is the real part and Vj_I is the imaginary part of input V_j which is the output from neuron j, and C_i is the imaginary part of the impedance capacitor $j\omega C$. The capacitor is treated as a weight with zero inputs since it is grounded.

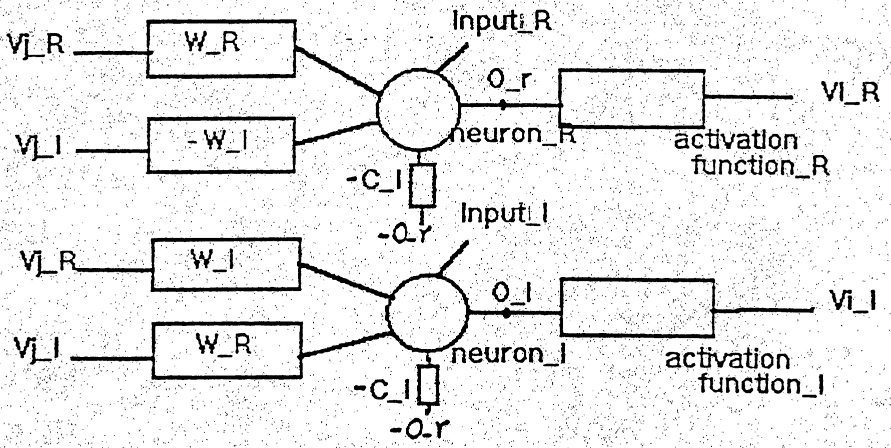


Figure 4.3 Decomposing a Complex Neuron into Two Real Neurons

The activation function $f(z) = z / | z |$ in this model is represented by a dependent voltage source. The absolute value is determined by the real part and the imaginary part of the outputs. So, in figure 4.3,

$$V_{i}\_R = O\_r / \sqrt{(O\_r)2 + (O\_i)2},$$

$$V_{i}\_I = O\_i / \sqrt{(O\_r)2 + (O\_i)2}.$$

Unlike other complex neurons which need capacitors and inductors, only resistors are required to represent the weight in figure 4.3. This is one of the advantages of this model.

The complex HNN in Figure 4.1 can be decomposed as shown in figure 4.4. In figure 4.4, V_{1_r} and V_{2_r} are the real parts, and V_{1_i} and V_{2_i} are the imaginary parts of the complex number V_1 and V_2 which are shown in Figure 4.3. Input1_r is the real part and input1_i is the imaginary part of the external input1. W_{12_R} is the real part and W_{12_I} is the imaginary part of the complex weight W_{12}, and C_i is a resistor, as explained above.

4.3 The Equivalency of the Complex and Real Circuits

To see that the circuit of two real coupled neurons (figure 4.4) is equivalent to a single complex neuron (figure 4.2), consider the net input to complex neuron j:

$$\text{net}_j = \sum_j w_j v_j$$

then, $\text{Re}(\text{net}_j) = \text{Re}(\sum_j wjvj) = \text{Re}(\sum_j ((w_{j\_R} + w_{j\_I}) + (v_{j\_R} + i^* v_{j\_I}))$

$$= \sum_j (w_{j\_R}v_{j\_R} - w_{j\_I}v_{j\_I}).$$

$$\mathrm{Im}(net_j) = \mathrm{Im}(\sum_j wjvj) = \mathrm{Im}(\sum_j ((w_{j\_R} + w_{j\_I}) + (v_{j\_R} + i^* v_{j\_I}))$$

$$= \sum_j (w_{j\_R}v_{j\_I} - w_{j\_I}v_{j\_R}).$$

By inspection of figure 4.3, it can be seen that the quantities Re(net$_j$) and Im(net$_j$) correspond to the net inputs of the two real neurons.

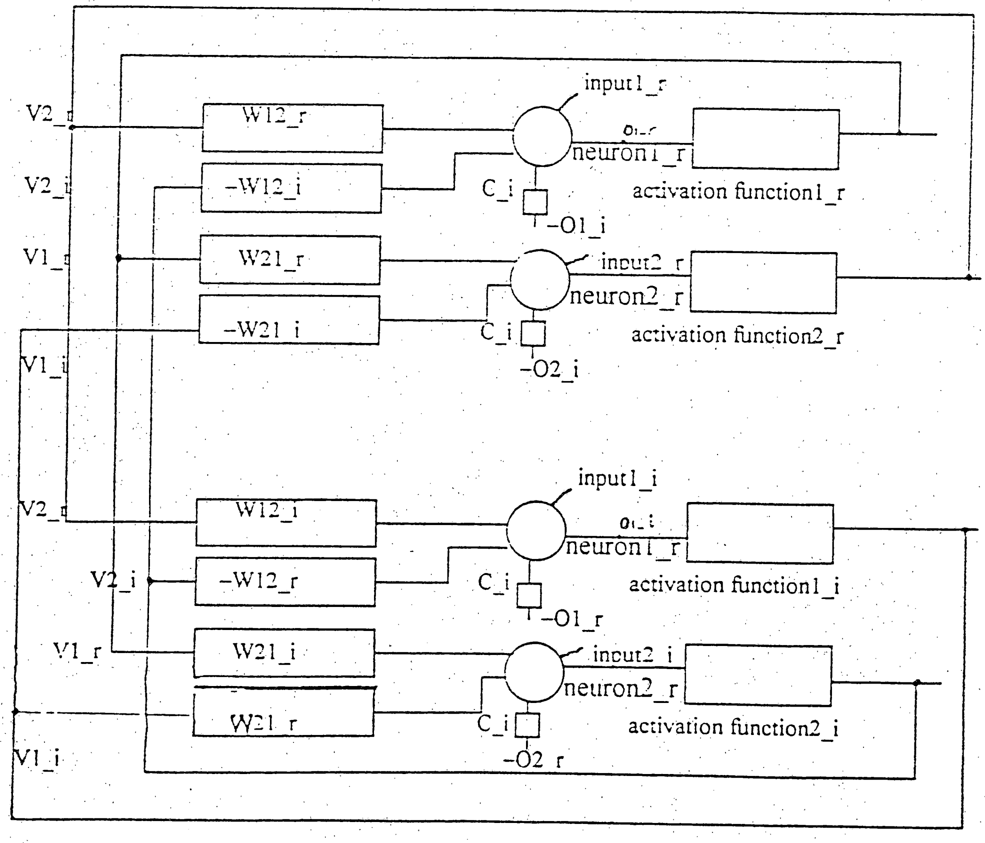


Figure 4.4 Decomposing the Complex HNN to Real Coupled Networks

It is clear that $\mathrm{Re}(net_j)$ corresponds to the net input of one real neuron in figure 4.4, and $\mathrm{Im}(net_j)$ to another, and both neurons correspond to complex neuron j.

4.4 Mapping Complex Equations to Real Ones

In this section, it will be shown how to solve a simultaneous system of complex equations in the real domain. Although the mapping used was previously discussed (Eaton, 1983), to our knowledge it was not used for this purpose.

Suppose $\mathbb{C}$ is the field of complex numbers, $\mathbb{C}^n$ is the *n*- dimensional complex vector space of *n*-tuples (columns) of complex numbers, and C_n is the set of all n * n complex matrices. Each $x \in \mathbb{C}^n$ has the unique representation $x = u + iv$ with $u, v \in R^n$. The u is the real part of x, v is the imaginary part of x, and $i = \sqrt{-1}$ is the imaginary unit.

This representation of x defines a real vector space isomorphism between $\mathbb{C}^n$ and R^{2n}.

for $x \in \mathbb{C}^n$, let

$$[x] = \begin{bmatrix} u \\ \\ v \end{bmatrix} \in R^{2n}$$

where $x = u + iv$,

and $[ax + by] = a[x] + b[y]$, for $x, y \in \mathbb{C}^n$, $a,b \in R$.

If $C \in \mathbb{C}^n$, then $C = A + iB$ where A and B are n * n real matrices.

Thus for $x = u + iv \in \mathbb{C}^n$,

$$Cx = (A + iB)(u + iv) = Au - Bv + i(Av + Bu).$$

So

$$[\text{Cx}] = \begin{bmatrix} \text{A}u - \text{B}v \\ \\ \text{A}v + \text{B}u \end{bmatrix} = \begin{bmatrix} \text{A}, & \text{-B} \\ \\ \text{B}, & \text{A} \end{bmatrix} \begin{bmatrix} u \\ \\ v \end{bmatrix}$$

The above mapping can be used in solving a simultaneous system of complex equation.

For example, if there is a linear problem,

$$\text{W X} = \begin{bmatrix} 2\text{-}3i \\ \\ 3+5i \end{bmatrix}$$

where weight matrix

$$\text{W} = \begin{bmatrix} 3+i, & 2\text{-}i \\ \\ 5+2i, & 6\text{-}7\text{i} \end{bmatrix},$$

the unknow vector X

$$X = \begin{bmatrix} v_1 \\ \\ v_2 \end{bmatrix}, \text{ where } v_1 = u_1 + iu_3,\ v_2 = u_2 + iu_4 .$$

The system is mapped into the real domain as follows:

$$A = \begin{bmatrix} 3 & , & 2 \\ \\ 5 & , & 6 \end{bmatrix}, B = \begin{bmatrix} 1 & , & -1 \\ \\ 2 & , & -7 \end{bmatrix}, \text{ and}$$

$$\begin{bmatrix} A & , & -B \\ \\ B & , & A \end{bmatrix} \begin{bmatrix} v_1 \\ \\ v_2 \end{bmatrix} = \begin{bmatrix} 2-i \\ \\ 3+5i \end{bmatrix}$$

$$\Rightarrow \begin{bmatrix} A & , & -B \\ \\ \\ B & , & A \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} = \begin{bmatrix} 2 \\ 3 \\ -1 \\ 5 \end{bmatrix}$$

Solving the above real system, we obtain $u_1 = 60/73$, $u_2 = -74/73$, $u_3 = -87/73$, $u_4 = 27/73$. It can be verified that the answer to the original complex system is $(60/73 - i*87/73, -74/73 + i*27/73)$ as expected.

CHAPTER FIVE

Results

5.1 The Input and Output of Simulating the Complex HNN

The complex HNN was simulated by first being mapped an equivalent real circuit. Consider the case of a complex HNN with two complex neurons as shown in figure 4.1. After decomposing the complex numbers, figure 4.4 shows that input1_r is the real part and input1_i is the imaginary part of the external input1 shown in figure 4.1; input2_r is the real part and input2_i is the imaginary part of the external input2 shown in figure4.1. In figure 4.4, V1_r is the real part and V1_i is the imaginary part of the complex output V1 shown in figure 4.1; V2_r is the real part and V2_i is the imaginary part of the complex output2 shown in figure 4.1.

5.1.1 Input of Pulses [1, -1]

When the complex weight

$$W = \begin{bmatrix} 0\,, & 0.004+0.002i \\ & \\ 0.004-0.002i, & 0 \end{bmatrix}$$, the output is shown in figure 5.1.

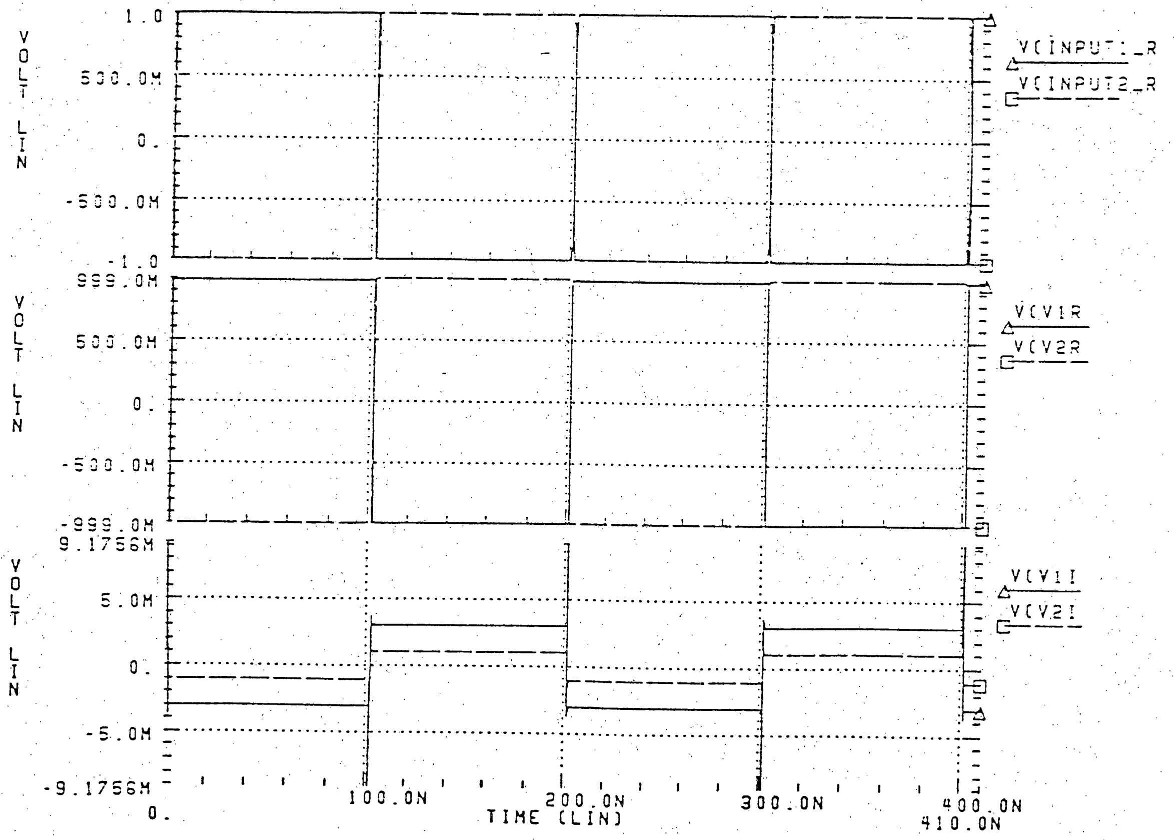


Figure 5.1 Input of Pulses with Different Weights [0.004, 0.002]

In figure 5.1, the first plot is the Inputs which are pulses [1, -1]. The second plot is the real parts of the outputs which are pulses [0.999000, -0.999000]. The third plot is the imaginary parts of the outputs which are pulses [-0.003000, -0.001000].

When the complex weight

$$W = \begin{bmatrix} 0 & 0.01+0.03i \\ & \\ 0.01-0.03i & 0 \end{bmatrix}$$, the output is shown in figure 5.2.

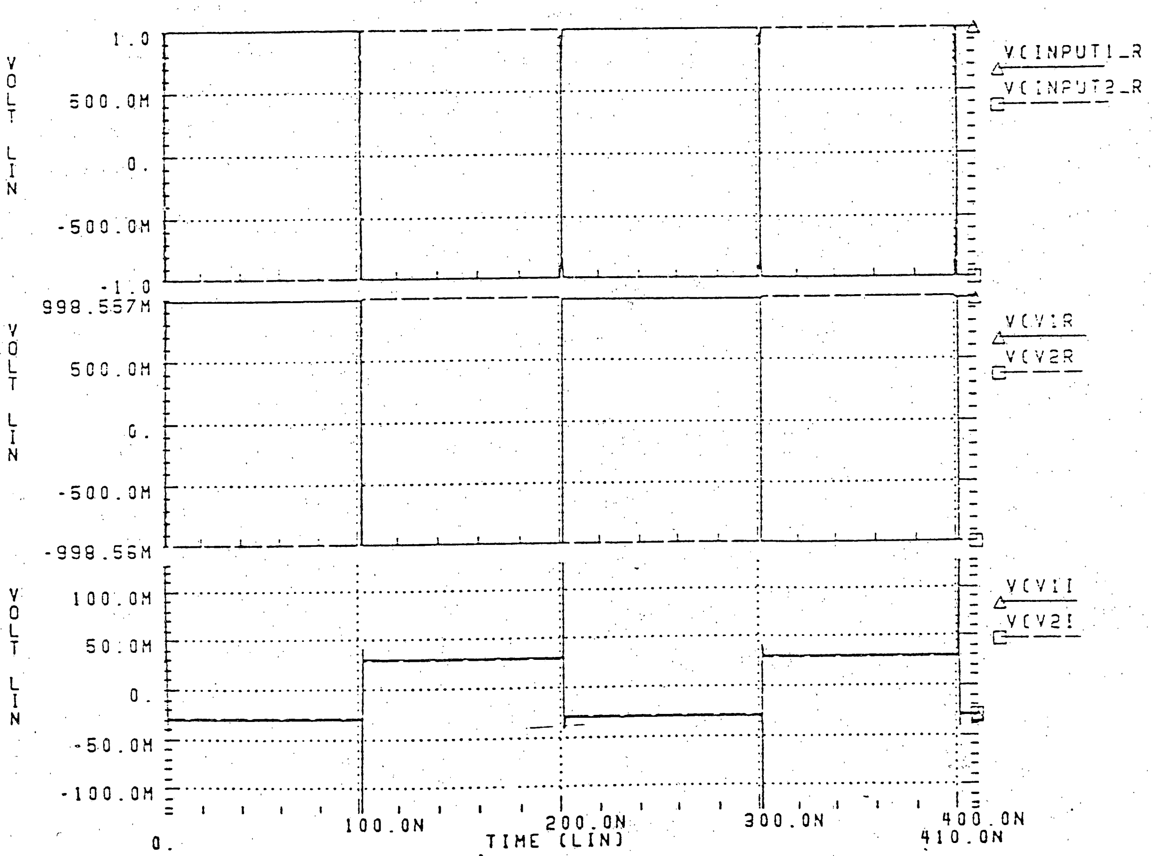


Figure 5.2 Input of Pulses with Different Weights [0.01, 0.03]

In figure 5.2, the first plot is the Inputs which are pulses [1, -1]. The second plot is the real parts of the outputs which are pulses [0.998557, -0.99856]. The third plot is the imaginary parts of the outputs which are pulses [-0.050000, -0.050000].

When the complex weight

$$W = \begin{bmatrix} 0, & 0.3+0.4i \\ & \\ 0.3-0.4i, & 0 \end{bmatrix}$$, the output is shown in figure 5.3.

In figure 5.3, the first plot is the Inputs which are pulses [1, -1]. The second plot is the real parts of the outputs which are pulses [0.943280, -0.943280]. The third plot is the imaginary parts of the outputs which are pulses [-0.375000, -0.375000].

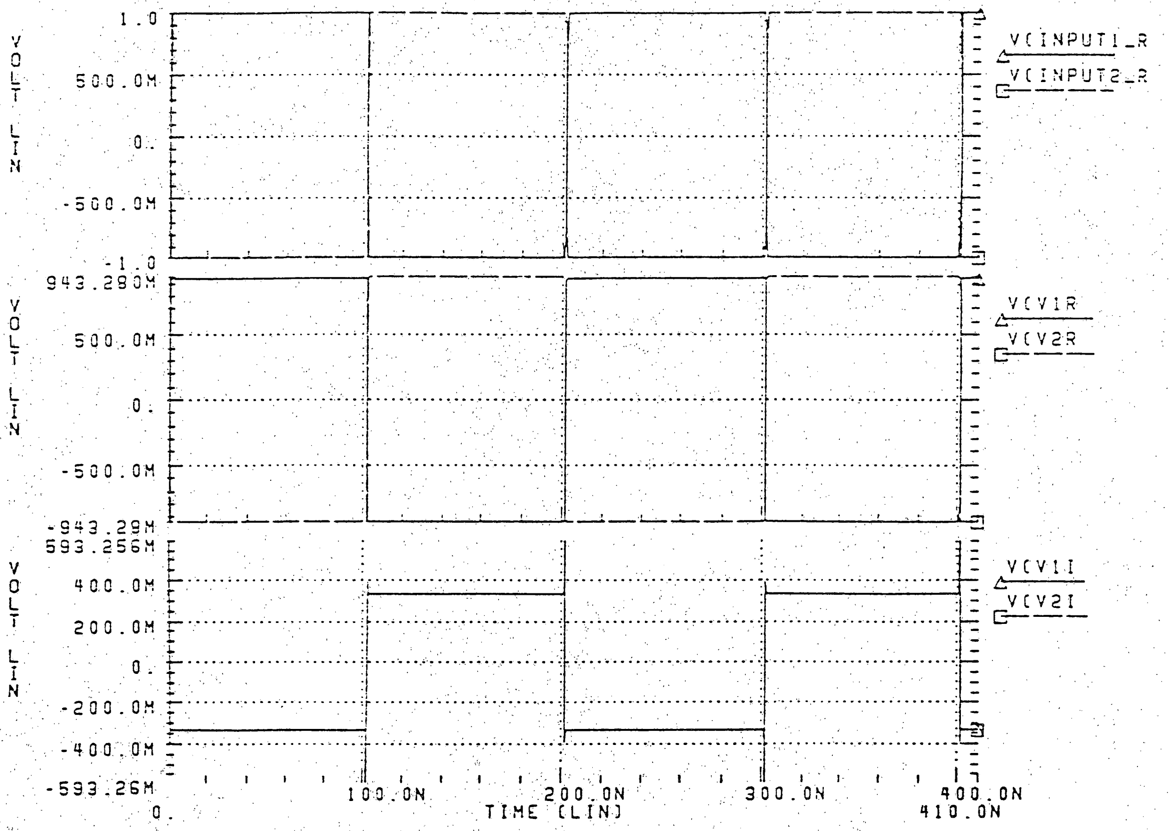


Figure 5.3 Input of Pulses with Different Weights [0.3, 0.4]

5.1.2 Input of Constants [1, -1]

When the complex weight

$$W = \begin{bmatrix} 0, & 0.004+0.002i \\ & \\ 0.004-0.002i, & 0 \end{bmatrix}$$, the output is shown in figure 5.4.

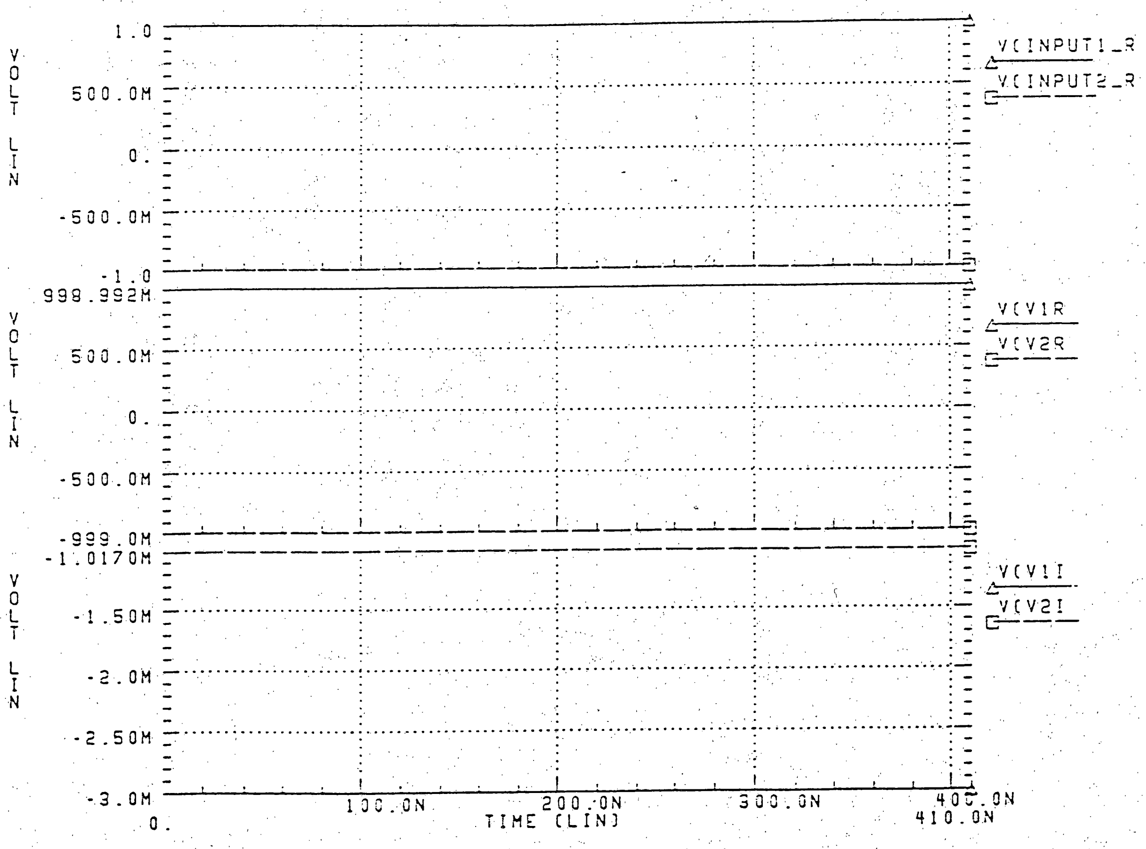


Figure 5.4 Input of Constants [1, -1] with Different Weights [0.004, 0.002]

In figure 5.4, the first plot is the Inputs which are constants [1, -1]. The second plot is the real parts of the outputs which are constants [0.998992, -0.999000]. The third plot is the imaginary parts of the outputs which are constants [-0.003000, -0.001017].

When the complex weight

$$W = \begin{bmatrix} 0, & 0.01+0.03i \\ & \\ 0.01-0.03i, & 0 \end{bmatrix}$$, the output is shown in figure 5.5.

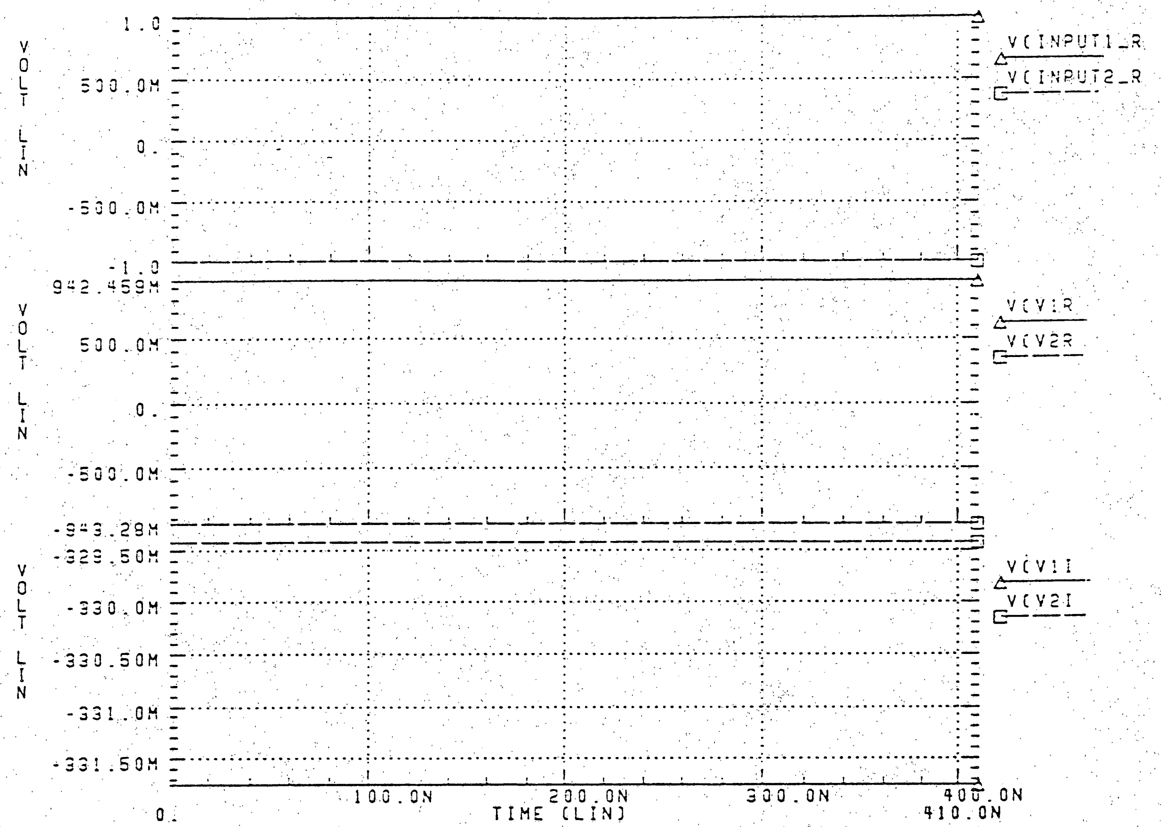


Figure 5.5 Input of Constants [1 , -1] with Different Weights [0.01, 0.03]

In figure 5.5, the first plot is the Inputs which are constants [1, -1]. The second plot is the real parts of the outputs which are constants [0.998496, -0.998560]. The third plot is the imaginary parts of the outputs which are constants [-0.029502, -0.031483].

When the complex weight

$$W = \begin{bmatrix} 0, & 0.3+0.4i \\ 0.3-0.4i, & 0 \end{bmatrix}$$, the output is shown in figure 5.6.

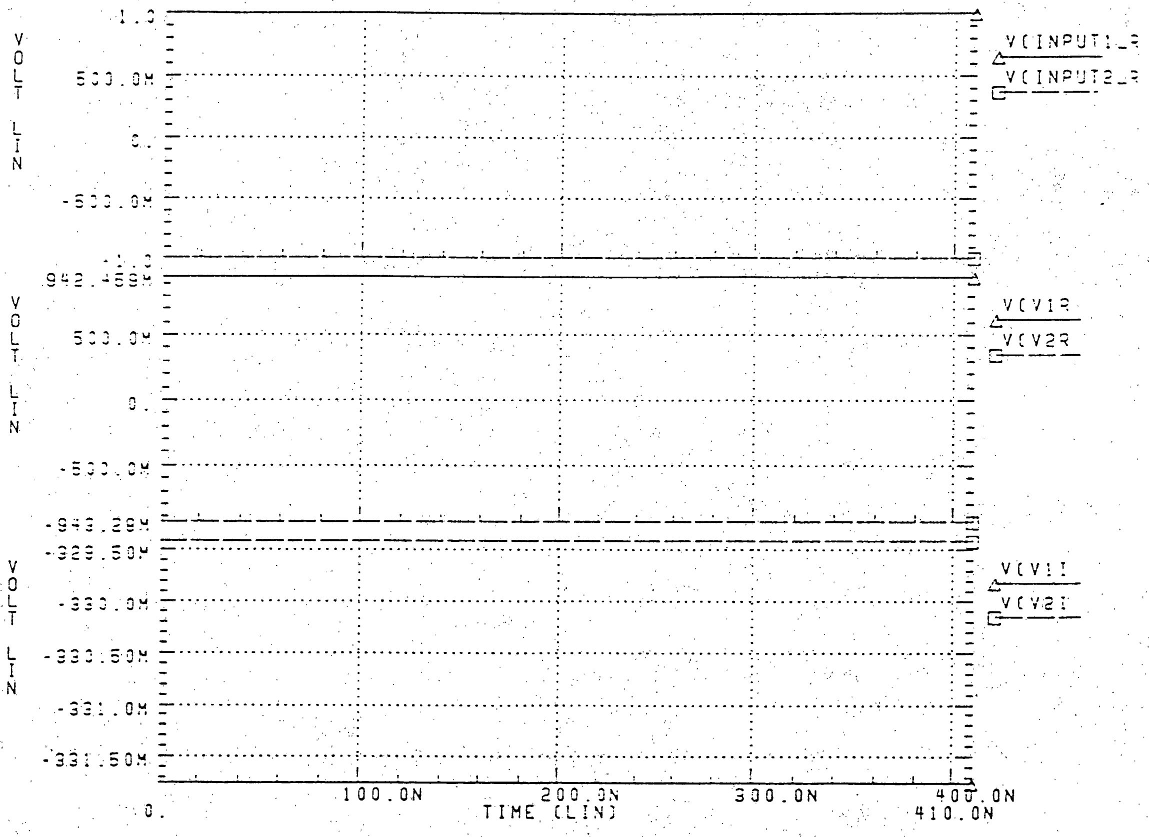


Figure 5.6 Input of Constants [1 , -1] with Different Weights [0.3, 0.4]

In figure 5.6, the first plot is the Inputs which are constants [1, -1]. The second plot is the real parts of the outputs which are constants [0.942459, -0.943280]. The third plot is the imaginary parts of the outputs which are constants [-0.331750, -0.329600].

5.1.3 The Complex HNN with Four Neurons

The extension of the complex HNN was also simulated. The performance of this model with four neurons is the following: when the complex weight

$$W = \begin{bmatrix} 0 & 0.001+0.001i, & 0.005+0.006i, & 0.001+0.002i \\ 0.001-0.001i, & 0 & 0.002+0.003i, & 0.001+0.003i \\ 0.005-0.006i, & 0.002-0.003i, & 0 & 0.002+0.004i \\ 0.001-0.002i, & 0.001-0.003i, & 0.002-0.004i, & 0 \end{bmatrix}$$

and the input are pulses [-1, 1, -1, 1], the output is shown in figure 5.7.

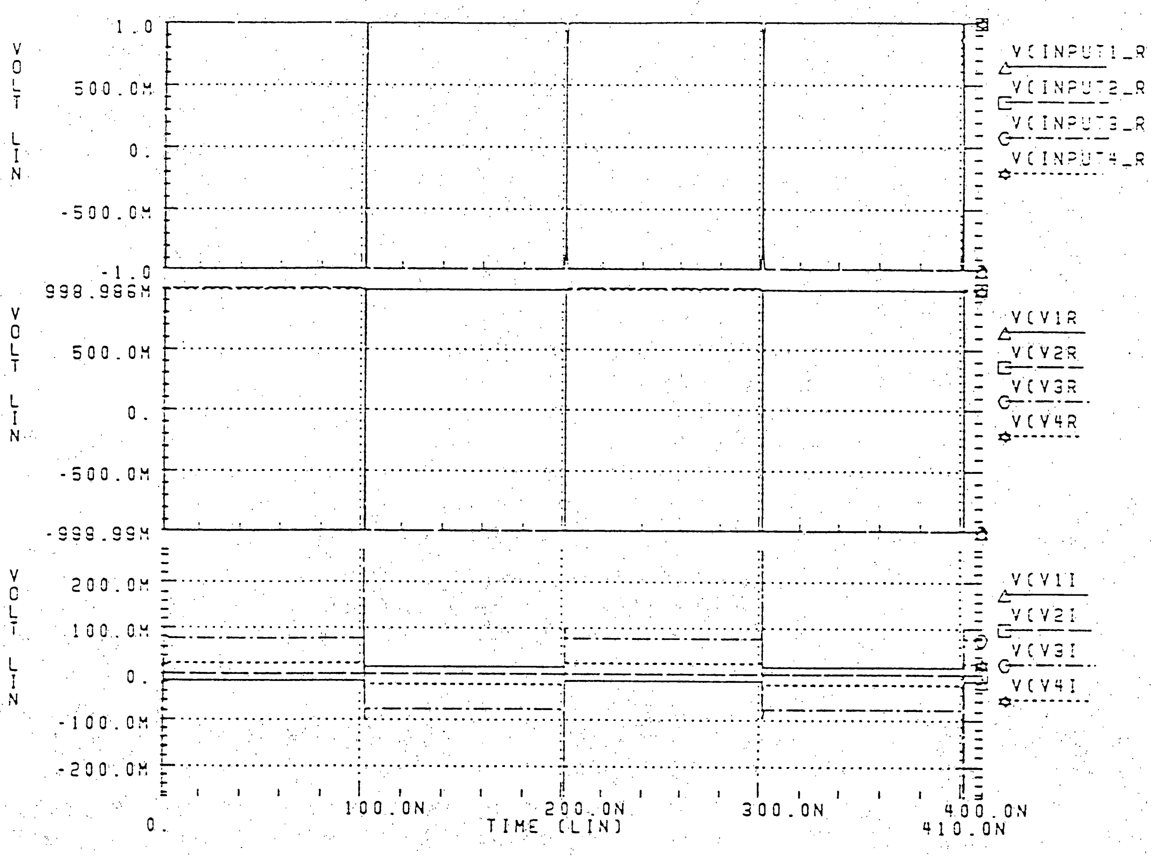


Figure 5.7 Input of Pulses in the Complex HNN with Four Neurons

In figure 5.7, the first plot is the Inputs which are pulses [-1, 1, -1, 1]. The second plot is the real parts of the outputs which are pulses [-0.99899, 0.998986, -0.99899, 0.998986]. The third plot is the imaginary parts of the outputs which are pulses [-0.020000, 0, 0.080000,0.020000]. Using the same weights and constant inputs [1, -1, 1, -1], the output is shown in figure 5.8.

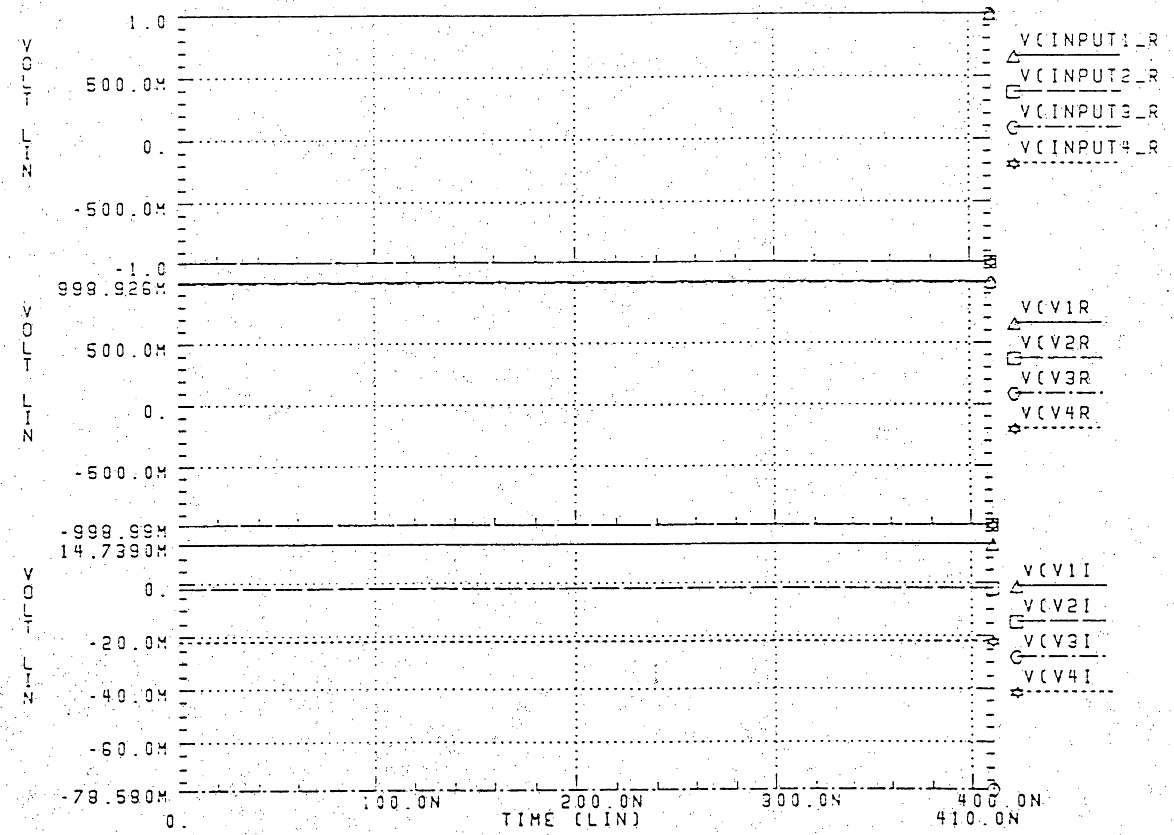


Figure 5.8 Input of Constants in the Complex HNN with Four Neurons

In figure 5.8, the first plot is the Inputs which are pulses [1, -1, 1, -1]. The second plot is the real parts of the outputs which are constants [0.998926, -0.998990, 0.998926, -0.998990]. The third plot is the imaginary parts of the outputs which are the constants [0.014739, 0.003000, -0.078580, -0.022000].

As shown above, when the inputs are pulses, the real parts of the outputs take symmetric values, and the imaginary parts remain the same. The values for the weights were varied, but the output remained close to the input as shown. The correctness of the design was verified with the aid of a C program. To avoid posing singularities, a small constant was added to the activation function which became $f(z) = z / (|z| + 10^{-3})$.

CHAPTER SIX

Conclusions

A circuit corresponding to the complex HNN has been designed in this study. In particular, complex weights were designed as circuit components. The same type weights were applied to design DFT circuit for validation.

To overcome the limitation of SPICE, a mathematical method was found to map a complex HNN into an equivalent real coupled one with weights represented as resistors. In general, complex weights include resistors, capacitors, and inductors. The correctness of the design was experimentally confirmed with the aid of a C program. A systematic method was used to solve a simultaneous system of complex linear equations by mapping them into the real domain. Using this method, many problems involving complex values can be solved by decomposition into the real domain. When compared to traditional

HNN models, the complex HNN is more general, as it contains the real valued HNN as a special case.

It is hoped a future version of SPICE can run the complex HNN simulation directly, and that the complex HNN will be manufactured as a chip.

APPENDIX A

The Simulation of the DFT with N =16, 32

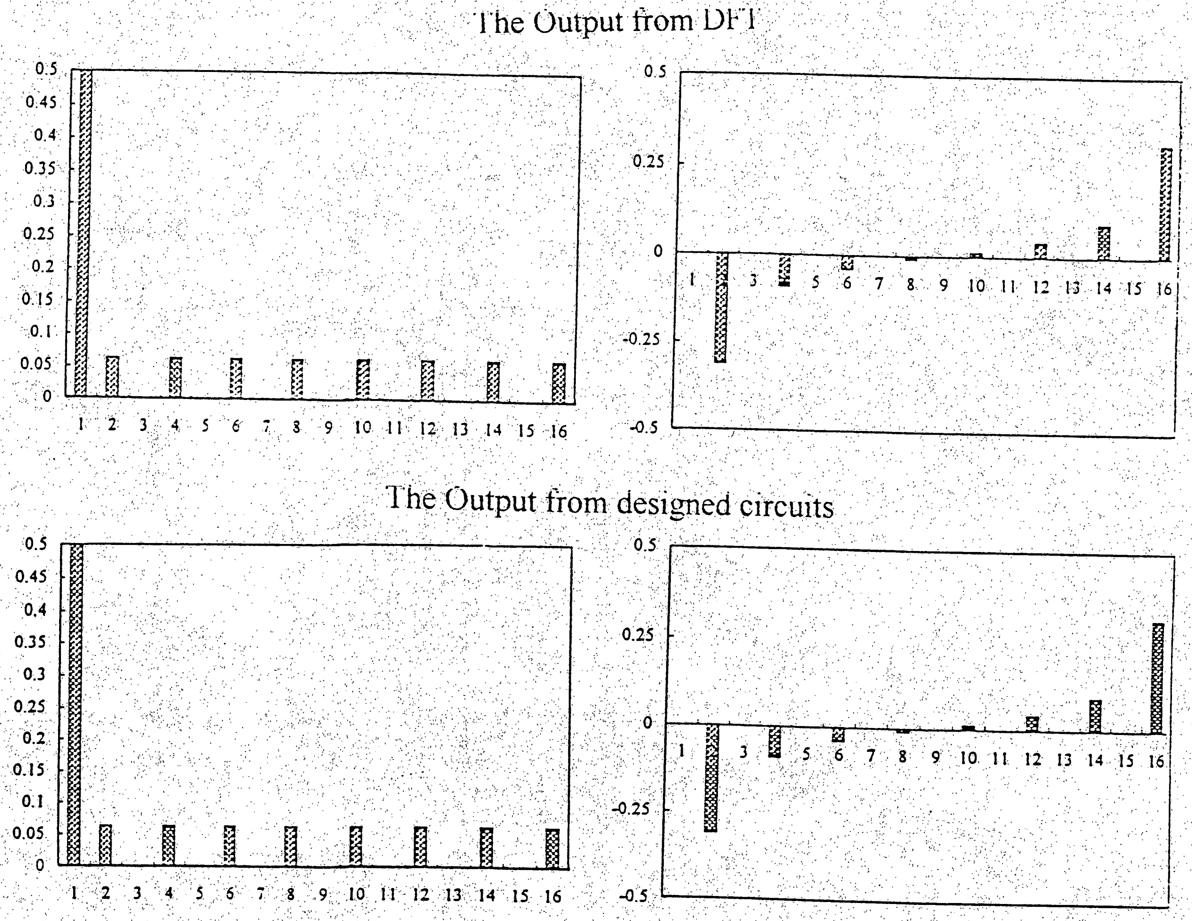
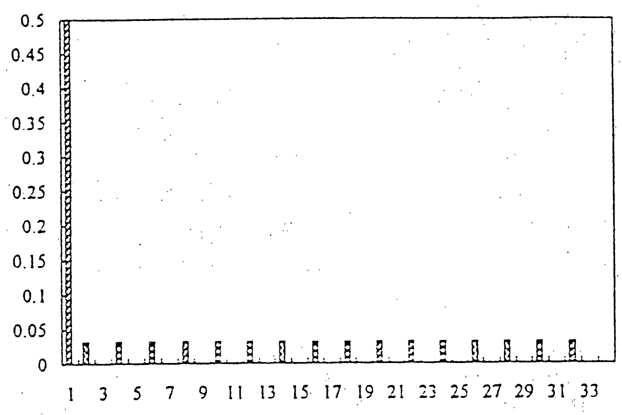
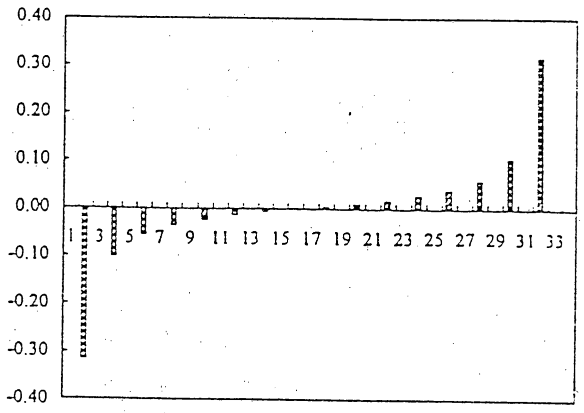


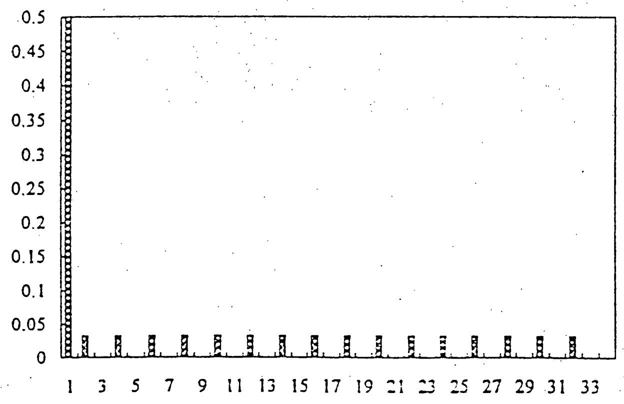
Figure a The Simulation of the DFT with N =16

The Output from DFT





The Output from designed circuits



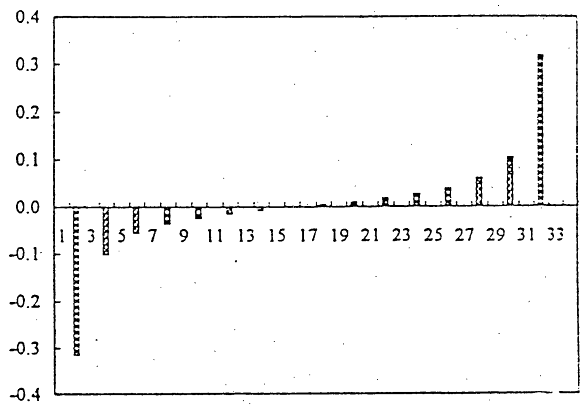


Figure b The Simulation of the DFT with N =32

APPENDIX B

The SPICE Code of Simulating Complex HNN

```
******* Subckts
.subckt OP p n out
Eop  out 0 OPAMP p n
.ends OP

.subckt adder a b c d  virtual_gnd out
x1 0 virtual_gnd out1 OP
r1 a virtual_gnd 1g
r2 b virtual_gnd 1g
r3 c virtual_gnd 1g
r4 d virtual_gnd 1g
r5 out1 virtual_gnd 1g
e_a1 out 0 VCL='v(out1) * -1'
.ends adder

.subckt mul in out m=wm
e_mul1 out 0 VOL='V(in) * m'
.ends mul

******* Parameters
.param w12r=0.01 w12i=0.03
.param w21r=0.01 w21i=-0.03
.param ci=0.001
.param vi=0 v1=1.0 v2=-1.0 td=1ns tr=1ns tf=1ns pw=99ns pt=200ns

******* Main Ckts
Vi1_r input1_r 0 pulse v1 v1 td tr tf pw pt
Vi1_i input1_i 0 pulse vi vi td tr tf pw pt
Vi2_r input2_r 0 pulse v2 v2 td tr tf pw pt
Vi2_i input2_i 0 pulse vi vi td tr tf pw pt

x1  v2r o1rr mul wm='w12r'
x2  v2i o1ri mul wm='-1 * w12i'
x3  o1i o1ci mul wm='ci'
x4  v1r o2rr mul wm='w21r'
x5  v1i o2ri mul wm='-1 * w21i'
x6  o2i o2ci mul wm='ci'
x7  v2r o1ir mul wm='w12i'
x8  v2i o1ii mul wm='w12r'
x9  o1r o1cr mul wm='-1 * ci'
x10 v1r o2ir mul wm='w21i'
x11 v1i o2ii mul wm='w21r'
x12 o2r o2cr mul wm='-1 * ci'

x13 input1_r o1rr o1ri o1ci vtg_o1r o1r adder
x14 input2_r o2rr o2ri o2ci vtg_o2r o2r adder
x15 input1_i o1ir o1ii o1cr vtg_o1i o1i adder
x16 input2_i o2ir o2ii o2cr vtg_o2i o2i adder

e_norm1  v1r  0 VOL='V(o1r)/(sqrt(V(o1r)*V(o1r)+V(o1i)*V(o1i))+1m)'
e_norm3  v1i  0 VOL='V(o1i)/(sqrt(V(o1r)*V(o1r)+V(o1i)*V(o1i))+1m)'
e_norm2  v2r  0 VOL='V(o2r)/(sqrt(V(o2r)*V(o2r)+V(o2i)*V(o2i))+1m)'
e_norm4  v2i  0 VOL='V(o2i)/(sqrt(V(o2r)*V(o2r)+V(o2i)*V(o2i))+1m)'

.end
```

```
******* Parameters
.param w12r=0.01 w12i=0.01
.param w13r=0.05 w13i=0.06
.param w14r=0.01 w14i=0.02
.param w21r=0.01 w21i=-0.01
.param w23r=0.02 w23i=0.03
.param w24r=0.01 w24i=0.03
.param w31r=0.05 w31i=-0.06
.param w32r=0.02 w32i=-0.03
.param w34r=0.02 w34i=0.04
.param w41r=0.01 w41i=-0.02
.param w42r=0.01 w42i=-0.03
.param w43r=0.02 w43i=-0.04
.param ci=0.01

.param vi=0 v1=1 v2=-1 td=1ns tr=1ns tf=1ns pw=99ns pt=200ns


******* Main Ckts
Vi1_r input1_r 0 pulse v1 vi td tr tf pw pt
Vi1_i input1_i 0 pulse vi vi td tr tf pw pt
Vi2_r input2_r 0 pulse v2 v2 td tr tf pw pt
Vi2_i input2_i 0 pulse vi vi td tr tf pw pt
Vi3_r input3_r 0 pulse v1 v1 td tr tf pw pt
Vi3_i input3_i 0 pulse vi vi td tr tf pw pt
Vi4_r input4_r 0 pulse v2 v2 td tr tf pw pt
Vi4_i input4_i 0 pulse vi vi td tr tf pw pt
*Vi1_r input1_r 0 pulse v1 vi td tr tf pw pt ac 1
*Vi1_i input1_i 0 pulse vi vi td tr tf pw pt ac 0
*Vi2_r input2_r 0 pulse v2 v2 td tr tf pw pt ac 1, 180
*Vi2_i input2_i 0 pulse vi vi td tr tf pw pt ac 0
*Vi1_r n1_r 0 sin (0.0 2.5 100Meg 0ns) ac 1
*Vi1_i n1_i 0 sin (0.0 0.0 100Meg 0ns) ac 0
*Vi2_r n2_r 0 sin (0.0 2.5 100Meg 0ns 0 180) ac 1, 180
*Vi2_i n2_i 0 sin (0.0 0.0 100Meg 0ns) ac 0

x1  v2r c1r_2r mul wm='w12r'
x2  v2i c1r_2i mul wm='-1 * w12i'
x3  v3r c1r_3r mul wm='w13r'
x4  v3i c1r_3i mul wm='-1 * w13i'
x5  v4r c1r_4r mul wm='w14r'
x6  v4i c1r_4i mul wm='-1 * w14i'

x7  v1r c2r_1r mul wm='w21r'
x8  v1i c2r_1i mul wm='-1 * w21i'
x9  v3r c2r_3r mul wm='w23r'
x10 v3i c2r_3i mul wm='-1 * w23i'
x11 v4r c2r_4r mul wm='w24r'
x12 v4i c2r_4i mul wm='-1 * w24i'

x13 v1r c3r_1r mul wm='w31r'
x14 v1i c3r_1i mul wm='-1 * w31i'
x15 v2r c3r_2r mul wm='w32r'
x16 v2i c3r_2i mul wm='-1 * w32i'
x17 v4r c3r_4r mul wm='w34r'
x18 v4i c3r_4i mul wm='-1 * w34i'

x19 v1r c4r_1r mul wm='w41r'
x20 v1i c4r_1i mul wm='-1 * w41i'
x21 v2r c4r_2r mul wm='w42r'
x22 v2i c4r_2i mul wm='-1 * w42i'
x23 v3r c4r_3r mul wm='w43r'
x24 v3i c4r_3i mul wm='-1 * w43i'

x25 v2r c1i_2r mul wm='w12i'
x26 v2i c1i_2i mul wm='w12r'
x27 v3r c1i_3r mul wm='w13i'
x28 v3i c1i_3i mul wm='w13r'
x29 v4r c1i_4r mul wm='w14i'
x30 v4i c1i_4i mul wm='w14r'

x31 v1r c2i_1r mul wm='w21i'
x32 v1i c2i_1i mul wm='w21r'
x33 v3r c2i_3r mul wm='w23i'
x34 v3i c2i_3i mul wm='w23r'
x35 v4r c2i_4r mul wm='w24i'
x36 v4i c2i_4i mul wm='w24r'
```

```
x37 v1r o3i_1r mul wm='w31i'
x38 v1i o3i_1i mul wm='w31r'
x39 v2r o3i_2r mul wm='w32i'
x40 v2i o3i_2i mul wm='w32r'
x41 v4r o3i_4r mul wm='w34i'
x42 v4i o3i_4i mul wm='w34r'

x43 v1r o4i_1r mul wm='w41i'
x44 v1i o4i_1i mul wm='w41r'
x45 v2r o4i_2r mul wm='w42i'
x46 v2i o4i_2i mul wm='w42r'
x47 v3r o4i_3r mul wm='w43i'
x48 v3i o4i_3i mul wm='w43r'


x49 input1_r o1r_2r o1r_2i o1r_3r o1r_3i o1r_4r o1r_4i o1r_o1i vtg_o1r o1r adder

x50 input2_r o2r_2r o2r_2i o2r_3r o2r_3i o2r_4r o2r_4i o2r_o2i vtg_o2r o2r adder
x51 input3_r o3r_1r o3r_1i o3r_2r o3r_2i o3r_4r o3r_4i o3r_o3i vtg_o3r o3r adder
x52 input4_r o4r_1r o4r_1i o4r_2r o4r_2i o4r_3r o4r_3i o4r_o4i vtg_o4r o4r adder
x53 input1_i o1i_2r o1i_2i o1i_3r o1i_3i o1i_4r o1i_4i o1i_o1r vtg_o1i o1i adder
x54 input2_i o2i_1r o2i_1i o2i_3r o2i_3i o2i_4r o2i_4i o2i_o2r vtg_o2i o2i adder
x55 input3_i o3i_1r o3i_1i o3i_2r o3i_2i o3i_4r o3i_4i o3i_o3r vtg_o3i o3i adder
x56 input4_i o4i_1r o4i_1i o4i_2r o4i_2i o4i_3r o4i_3i o4i_o4r vtg_o4i o4i adder

x57 o1i o1r_o1i mul wm='ci'
x58 o2i o2r_o2i mul wm='ci'
x59 o3i o3r_o3i mul wm='ci'
x60 o4i o4r_o4i mul wm='ci'
x61 o1r o1i_o1r mul wm='-1 * ci'
x62 o2r o2i_o2r mul wm='-1 * ci'
x63 o3r o3i_o3r mul wm='-1 * ci'
x64 o4r o4i_o4r mul wm='-1 * ci'

e_norm1  v1r  0 VOL='V(o1r)/(sqrt(V(o1r)*V(o1r)+V(o1i)*V(o1i))+1m)'
e_norm3  v1i  0 VOL='V(o1i)/(sqrt(V(o1r)*V(o1r)+V(o1i)*V(o1i))+1m)'
e_norm2  v2r  0 VOL='V(o2r)/(sqrt(V(o2r)*V(o2r)+V(o2i)*V(o2i))+1m)'
e_norm4  v2i  0 VOL='V(o2i)/(sqrt(V(o2r)*V(o2r)+V(o2i)*V(o2i))+1m)'
e_norm5  v3r  0 VOL='V(o3r)/(sqrt(V(o3r)*V(o3r)+V(o3i)*V(o3i))+1m)'
e_norm6  v3i  0 VOL='V(o3i)/(sqrt(V(o3r)*V(o3r)+V(o3i)*V(o3i))+1m)'
e_norm7  v4r  0 VOL='V(o4r)/(sqrt(V(o4r)*V(o4r)+V(o4i)*V(o4i))+1m)'
e_norm8  v4i  0 VOL='V(o4i)/(sqrt(V(o4r)*V(o4r)+V(o4i)*V(o4i))+1m)'


****** Analysis
.option probe post=1 delmax=2n
.op 100ns
*.ac dec 10 1 100k
.tran 2n 410n
.print tran v(input1_r)
.print tran v(input2_r)
.print tran v(input3_r)
.print tran v(input4_r)
.print tran v(input1_i)
.print tran v(input2_i)
.print tran v(input3_i)
.print tran v(input4_i)
.print tran v(o1r)
.print tran v(o1i)
.print tran v(v1r)
.print tran v(v1i)
.print tran v(o2r)
.print tran v(o2i)
.print tran v(v2r)
.print tran v(v2i)
.print tran v(o3r)
.print tran v(o3i)
.print tran v(v3r)
.print tran v(v3i)
.print tran v(o4r)
.print tran v(o4i)
.print tran v(v4r)
.print tran v(v4i)

.end
```

BIBLIOGRAPHY

Birx, D. L., & Pipenberg, S. J. (1989). Neural network structures for defect discrimination. Fifth Annual Aerospace Applications of Artificial Intelligence Conference, 24-26.

Culhane, A. D., & Peckerar, M. C. (1989). A neural net approach to discrete Hartley and Fourier Transforms. IEEE Transaction, 695-703.

Dayhoff, J. (1990). Neural network architectures. Van Nostrand Reinhold, 52-57.

Elliott, D. F., & Rao, K. R. (1982). Fast transforms: algorithms, analyses, applications. Prentice Hall.

Eaton, M. L. (1983). Multivariate statistics - a vector space approach. John Wiley & Sons, Inc., 370-374.

Georgiou, G. M. (1992). Activation functions for neural networks in the complex domain. First International Conference on Fuzzy Theory and Technology.

Georgiou, G. M., & Koutsougeras, C. (1992). Complex domain backpropagation. IEEE Transactions on circuits and systems,vol. 39, No. 5.

Hopfield, J. J., (1982). Neural network and physical systems with emergent collective computational abilities. Proceedings of the National Academy of Sciences of the U.S.A. 79, 2554-2558.

Hopfield, J. J., (1984). Neurons with graded response have collective computational properties like those of two-state neurons. Proceedings of the National Academy of Sciences of the U.S.A. 81, 3088-3092.

Hopfield, J. J., & Tank, D. W. (1986). Simple neural optimization networks: an A/D converter, signal decision circuit, and a linear programming circuit. *IEEE* Trans. Circuits Syst. vol. CAS-36, 533-541.

Little, G., Steven, R., Gustafson, C., & Senn, R. A. (1990). Generalization of the backpropagation neural network learning algorithm to permit complex weights. APPLIED OPTICS, vol. 29, No. 11.

Limited, I. (1989). Digital signal processing. Prentice Hall. 132-133.

Noest, A. J. (1988). Neural information processing systems. American Institute of Physics, New York. 584-591.

Szilagyi, M., Mikkelsen, J. C., & Mortansen, K. H.(1990). Some new thoughts on neural networks with complex connection matrices. Technical Report DAIMI 332.

Takeda, M., & Goodman, J. W. (1986). Neural networks for computation: number representations and programming complexity. APPLIED OPTICS, vol. 25, No. 18..

Yang, W .H., Chan, K. K., & Chang, P. R. (1994). Complex-valued neural network for direction of arrival estimation. Electronics Letters, vol. 30.No. 7.

Zurada, J. M. (1992). Introduction to artificial neural systems. West Publishing Company. 251-300.