



This is to certify that the
thesis entitled

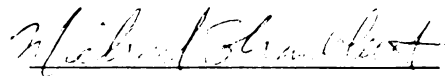
Analysis Of Neural Network Response
with Varied Neuron Models and Interconnection Patterns

presented by

David Barnard Pierce

has been accepted towards fulfillment
of the requirements for

Master's degree in Electrical
Engineering



Major professor

Date 13 Nov 1991

PLACE IN RETURN BOX to remove this checkout from your record.
TO AVOID FINES return on or before date due.

DATE DUE	DATE DUE	DATE DUE
MAR 3 1996		

MSU Is An Affirmative Action/Equal Opportunity Institution

c:\circ\datsdue.pm3-p.1

**ANALYSIS OF
NEURAL NETWORK RESPONSE
WITH VARIED NEURON MODELS
AND INTERCONNECTION PATTERNS**

By

David Barnard Pierce

A THESIS

**Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of**

MASTER OF SCIENCE

Department of Electrical Engineering

1991

ABSTRACT

ANALYSIS OF NEURAL NETWORK RESPONSE WITH VARIED NEURON MODELS AND INTERCONNECTION PATTERNS

By

David Barnard Pierce

This work addresses the performance of a neural network algorithm for constrained optimization. The network is simulated in VHSIC Hardware Description Language (VHDL). Comparisons are made for neurons with a step response and several levels of graded response, as well as several methods of interconnection. The response variables include response function, gain of the response function, and number of interconnections. The results show that a binary response is faster than a graded response. Additional interconnections resulted in an increase in speed of the network. Adding a capacitance factor to the input of the neuron model reduced oscillation and increased problem solving capability. Setting the gain and/or weights becomes the mechanism for implementing learning algorithms.

Contents

1	INTRODUCTION	1
1.1	Motivation	1
1.2	Scope	2
2	BACKGROUND	3
2.1	VHDL	3
2.1.1	Origin of VHDL	3
2.1.2	Structure of VHDL	4
2.1.3	Application of VHDL	6
2.2	Artificial Neural Networks	7
2.2.1	Artificial Neural Network Theory	7
2.2.2	Neuron Design	8
2.2.3	Interconnection Effects	11
3	NEURON AND NEURAL NETWORK SIMULATION	12
3.1	Neuron Model	12
3.1.1	Neuron Response Functions	12
3.1.2	Neuron Interconnections	16
3.1.3	Neuron VHDL Code	16
3.2	Network Model	17
3.2.1	Processor	17
3.2.2	Memory	17
3.2.3	Network	19
3.2.4	Test Bench	19
3.3	Simulation Inputs	19
3.4	Performance Measurement Technique	20
3.5	Procedure for Simulation	20
4	SIMULATION RESULTS	22
4.1	Neuron Model Results	22
4.2	Interconnection Pattern Results	23
5	DISCUSSION	27
6	CONCLUSIONS	30
A	APPENDIX A	32
A.1	Neuron Model VHDL Code	32
B	APPENDIX B	43
B.1	Network VHDL Code	43

C	APPENDIX C	54
C.1	Input Data Files	54
D	APPENDIX D	63
D.1	Simulation Outputs	63

List of Figures

1	Hopfield-Tank Network Model.	9
2	Sigmoid Transfer Function.	10
3	Binary Transfer Function.	13
4	Low Gain Ramp Transfer Function.	14
5	High Gain Ramp Transfer Function.	14
6	Low Gain Sigmoid Transfer Function.	15
7	High Gain Sigmoid Transfer Function.	15
8	Network Block Diagram.	18
9	VDHL Code for Binary Response Neuron.	33
10	VDHL Code for Binary Response Neuron with Capacitance.	34
11	VHDL Code for Low Gain Ramp Response Neuron.	35
12	VHDL Code for Low Gain Ramp Response Neuron with Capacitance.	36
13	VHDL Code for High Gain Ramp Response Neuron.	37
14	VHDL Code for High Gain Ramp Response Neuron with Capacitance.	38
15	VHDL Code for Low Gain Sigmoid Response Neuron.	39
16	VHDL Code for Low Gain Sigmoid Response Neuron with Capacitance.	40
17	VHDL Code for High Gain Sigmoid Response Neuron.	41
18	VHDL Code for High Gain Sigmoid Response Neuron with Capacitance.	42
19	VHDL Code for Neural Network.	44
20	Two Stage Interconnected Network Input Data.	55
21	Three Stage Interconnected Network Input Data.	56
22	Four Stage Interconnected Network Input Data.	57
23	Two Stage Interconnected Network with Sub-optimal Path Input Data.	59
24	Three Stage Interconnected Network with Sub-optimal Path Input Data.	60
25	Four Stage Interconnected Network with Sub-optimal Path Input Data.	61
26	Results of Binary Neuron Model.	64
27	Results of Low Gain Ramp Neuron Model.	68
28	Results of High Gain Ramp Neuron Model.	72
29	Results of Low Gain Sigmoid Neuron Model.	76
30	Results of High Gain Sigmoid Neuron Model.	80
31	Results of Binary Response Neuron With Capacitance.	84
32	Results of Low Gain Ramp Response Neuron With Capacitance.	88
33	Results of High Gain Ramp Response Neuron With Capacitance.	92
34	Results of Low Gain Sigmoid Response Neuron With Capacitance.	96
35	Results of High Gain Sigmoid Response Neuron With Capacitance.	102
36	Results of Three Stage Interconnection Pattern and Binary Neuron.	106
37	Results of Three Stage Interconnection Pattern and Ramp Neuron.	110
38	Results of Three Stage Interconnection Pattern and Sigmoid Neuron.	114
39	Results of Four Stage Interconnection Pattern with Binary Neuron.	118
40	Results of Four Stage Interconnection Pattern with Ramp Neuron.	121
41	Results of Four Stage Interconnection Pattern with Sigmoid Neuron.	125

42	Results of Two Stage Interconnection Pattern and Sub-optimal Path.	129
43	Results of Three Stage Interconnection Pattern and Sub-optimal Path.	133
44	Results of Four Stage Interconnection Pattern and Sub-optimal Path.	137

List of Tables

1	Results of Neuron Model Simulations.	23
2	Results of Varied Interconnections with Binary Neurons.	25
3	Results of Varied Interconnections with Low Gain Ramp Neurons. . .	25
4	Results of Varied Interconnections with Low Gain Sigmoid Neurons. .	25
5	Results of Varied Interconnections and Nonoptimal Path.	26
6	Results for Network with Capacitance Function.	29

1 INTRODUCTION

1.1 Motivation

The field of neural networks, although not a new field, is currently experiencing the focus of large amounts of research [1]. This research is exploring various methods of implementing networks, learning algorithms, different neuron models, and applications of the networks. The promise of faster solutions to difficult problems is a major reason for the focus upon neural networks.

Because neural networks are parallel processing structures, faster computations are expected. In addition, neural networks have fault-tolerant structures which provide additional incentive for use [2,3]. Neural networks are presently being utilized for many existing computation problems such as nonlinear programming [4], control applications [5,6], pattern recognition problems [2], and Content Addressable Memory (CAM) [9], to name a few. The parallel structure of the neural network allows fast comparison of patterns to compute a fast, 'near optimal' solution for a pattern, which can often be more useful than a slowly computed, optimal solution [9]. The neural network is also effective at the 'Traveling Salesman Problem', an np-complete problem that is time consuming for a single processor, sequential computing machine [11].

Despite the amount of research in the area of neural networks, there is still a large amount of knowledge to be gained. There is work being performed on network structures, learning algorithms, hardware implementations, and the specific neuron models to be utilized. One basic method of affecting the computational ability of the neural network is to vary the specific neuron response, the basic computation block of the network [12,13,14]. Also, the configuration of the network can have effects on the computational ability of the network. For example, the number and location of neuron interconnections will affect the computational ability.

Presently, there exists some work on the effect of different neuron responses and

their interconnections. Models with binary output and with graded response have been postulated [12,13]. Work has been published to show that a graded response will result in faster computations than a binary response [12]. However, there exists no actual simulation or comparison data for binary as well as graded response neurons. The effect of interconnections has also not been simulated for results. To generate an efficient network, these areas need to be explored.

1.2 Scope

Several possibilities exist for optimizing a neural network. The number of interconnections can be varied, the type of network algorithm can be varied, the method of interconnections or data transfer can be varied, and the specific neuron implementation can be varied. These parameters are the basic building blocks of each neural network.

As will be discussed, the type of neuron implementation can have a significant effect on the neural network and its ability to arrive at a decision in an efficient manner. This thesis addresses the performance of a neural network with changes in the implementation of individual neuron response curves and the number of interconnections.

These changes will be simulated and the results compared through the use of VHSIC Hardware Description Language (VHDL). VHDL allows hardware to be simulated and tested on a computer without actual implementation of the neural network hardware. This is important as actual hardware may require a great deal of money and time to generate results. VHDL allows simulations and results to be generated for a minimum of cost and time [16,18].

This work will explore the effects of varied neuron responses and the effect of varied interconnection patterns.

2 BACKGROUND

2.1 VHDL

2.1.1 Origin of VHDL

As the complexity of the engineer's design tasks increase, the methods of generating these designs must become more sophisticated. Without tools, such as computers and computer programs, the design task is completely manual. Manual drafting, development of schematics, tracking of part lists, and design analyses require large amounts of time and effort to generate and update. Manual methods are also more prone to errors in workmanship.

Because of the increasing complexity of the design tasks, it becomes necessary to develop tools to handle tasks like design analyses. Tools such as computer aided drafting, spreadsheets, SPICE (circuit analysis tool), and finite element analysis programs are available to aid the engineer in his/her design task.

For these same reasons, VHSIC Hardware Description Language (VHDL) was conceived and generated. The Department of Defense contracted for the development of VHDL to aid in the design of VHSIC hardware, which was a major item of defense expenditure. Without VHDL, the design of VHSIC hardware would be difficult to generate and to document.

VHDL will be utilized because of the amount of time and resources that would be required to prototype a neural network. To implement the necessary neural networks for the simulations would require many integrated circuit devices and much design work. This, in turn, requires a large amount of financial resources. By utilizing VHDL for simulations, the neural network can be generated and its operation verified in a fraction of the time. Any necessary changes can be coded and verified within minutes, which would not be possible with actual hardware. The cost to simulate the network in VHDL is only the time spent to design and code the problem.

2.1.2 Structure of VHDL

A major advantage of VHDL is the ability to utilize varied levels of hardware abstractions. The code can include exact models of gates with Boolean statements and combine these gates into a design. The code can also use hardware concepts to simulate a design. This can be performed by coding a block which performs complex Boolean and/or math functions on a set of data. These types of design can also be combined when desired.

The design is represented by a behavioral or structural model, or a combination of both. For the designer, this allows portions of uncreated hardware to be simulated and verified prior to spending large amounts of time implementing a specific hardware configuration. For example, a processor unit can be modeled behaviorally by sending certain output data when particular input data is sent to the processor. This can be coded without Boolean gate models allowing the designer to test a function or operation prior to generating all the internal structure necessary for operation.

Two basic elements of every system represented in VHDL are performed with an **entity statement** and an **architecture statement**. (Words that are reserved in VHDL are shown in **boldface** in this thesis). These two statements define an external and internal view of the system. The interface of the system is defined by an **entity statement**, which is the external view of the system. The behavior or structure of the system is given in the **architecture statement**, which is the internal view of the system.

The external view of the system described by the **entity statement** has commands to declare the system as an **entity** in addition to the system's **ports**. Each **port** defines a signal between the system and the systems around it. Signals defined by **ports** can include input, output, or bidirectional signals. Other information may also be declared in the **entity statement** concerning signals and values common to other systems.

The internal view of the system described by the **architecture** statement has commands to define what parts make up the system and how these parts define the system's operation. The system operation can be defined by a structural or a behavioral model. The structural model contains models of real hardware such as logic gates and memory devices and defines their connections. Behavioral models describe the system operation by logic statements such as **if** statements and **loop** commands. Both may perform the same function but are represented quite differently. For example, a counter could be implemented with Boolean gate models connected to operate like an actual counter. Or the Boolean logic of the counter could be implemented with **if** statements without detailed gate models.

Also within the **architecture** command, components (other systems) may be defined, signal values may be assigned, and other operations performed. The **component** statement places a previously defined system within the **architecture** being defined. The **architecture** of the counter example could include gate components by the instantiated **component** statement. The **component** statement includes a **port map** statement which defines the signal connections between the two systems.

The **architecture** of the system includes the code that defines the operation of the system. The code could include signal assignment statements that transmit a signal on a signal path. This can be performed as a result of a calculation, or a necessary condition being set. Also, variables can be calculated and arrays of components defined. These operations can be performed with **loops**, **if** statements, and other common software functions. For example, in this work the network uses a **loop** statement to calculate the summation for each neuron input. The neuron decision is made using an **if** statement.

The **architecture** statement also includes one other important concept which is not commonly found in software packages. The **process** statement defines a set of code which will run, given certain conditions. The **process** can be specified with sensitivity

to certain signals. The process will begin when a **transaction** or **event** occurs on a signal defined by the process as a sensitivity signal. The **process** statement also has a **wait** statement which suspends operation of the process when specified conditions occur. This capability will be utilized to update the network inputs and summations when a **transaction** has occurred.

VHDL allows the use of libraries for cataloging basic design elements. Elements such as gates, standard processors, or a hardware abstraction can be stored and recalled for use within any **entity** or **architecture** statement. The elements stored within a library are recalled by the **use** command. This command can recall another **entity**, or a group of entities within a sub-library.

One more significant ability of VHDL is the ability to declare **packages**. A **package** stores the basic elements of several designs and allows for **use** commands to address the **package** contents. A **package body** is the unit which contains functions or other commands of the **package**. System level details such as **system** constants, common functions, type declarations, and other similar information can be included in a **package** for use and modification for a single system.

In this work, the **package** and **package body** will be utilized to store the summation routine for each neuron's input. This routine will be implemented by a **function** command, which returns a value when called by the software. The value to be returned is the input summation for each neuron.

2.1.3 Application of VHDL

This work utilizes VHDL as the tool for simulating the artificial neural network and for simulating and gathering results. VHDL is a practical, realistic approach to designing a hardware neural network. It would be difficult and expensive to build a network with real hardware to test the network.

The neural network utilized was originally described in [15] and will be modified

for use in this work. The modifications and rationale are described in the next section.

2.2 Artificial Neural Networks

2.2.1 Artificial Neural Network Theory

Neural network theories have been in existence since 1943 [7]. The first work on neural networks concerned the mathematical concepts necessary for the network. Later work postulated actual hardware configurations and the mathematics supporting their use [9]. Much work has been centered around the Hopfield-Tank network [4,11,12,13,15].

Neural networks operate by computing a solution that reduces the network's energy function to a minimum value with certain structure conditions. These conditions allow the network to solve the problems detailed in this report. The energy function of the Hopfield-Tank network is

$$\begin{aligned}
E = & \frac{a}{2} \left(\sum_s \sum_i \sum_{j=i} V_{si} V_{sj} + \left(\sum_s \sum_i V_{si} - n \right)^2 \right) \\
& + \frac{b}{4} \sum_s \sum_i \sum_j \left(d_{si(s+1)j} V_{si} V_{(s+1)j} + d_{(s-1)j si} V_{si} V_{(s-1)j} \right) \\
& + \sum_s \sum_i \left(\frac{1}{R_{si}} \right) \int_{0.5}^{V_{si}} g_i^{-1}(\xi) d\xi.
\end{aligned}$$

This energy function contains terms for the network energy and the structural constraints of the network. This equation is minimized during the operation of a neural network. The equation forces one neuron to be enabled in each stage and forces the overall energy of the network to a minimum when one neuron is enabled within each stage. The last term on the right side can be ignored for a proper choice of gain (high gain).

The choice of weighting factors between each neuron is forced by the energy equation. The equation uses the weighting factors as part of the equation and the network is therefore highly dependent upon these choices. Without the proper choice of weighting factors, the network will not compute a valid solution.

Neural network theory generally attempts to draw parallels between biological neural networks and artificial neural networks (non-biological). Biological neural networks are responsible for the processing within the animal brain, such as sensory processing, pattern recognition, and speech. Because biological neural networks are very efficient at these tasks, it would be desirable to copy the network and utilize the network for tasks that general purpose, sequential instruction machines perform inefficiently.

The use of artificial neural networks is relatively new, due primarily to the inability of technology to produce a hardware implementation. Artificial neural networks require a large number of processors and interconnections, which has been beyond the capability of circuit technology.

The ability of manufacturers and the capabilities of equipment presently allows small neural networks to be implemented. Massively parallel structures necessary for large neural networks are not far away. These abilities have been progressing rapidly over the last several years [8].

2.2.2 Neuron Design

The Hopfield-Tank network emulates the biological neural network by utilizing an operational amplifier for each neuron. Each amplifier would have inputs, both excitatory and inhibitory, from many other neurons [9]. An example is shown in Figure 1. This network is very large for nontrivial problems. An example would be the human brain, which is estimated to contain 10^{11} neurons [10].

One of the most significant aspects of an artificial neural network is the neuron model. The neuron has been postulated as an analog processor, such as an op-amp [12,17] with excitatory and inhibitory inputs. Most models utilize a graded response to the inputs multiplied by their respective weights.

The neuron transfer function of a biological neuron is a monotonically increasing

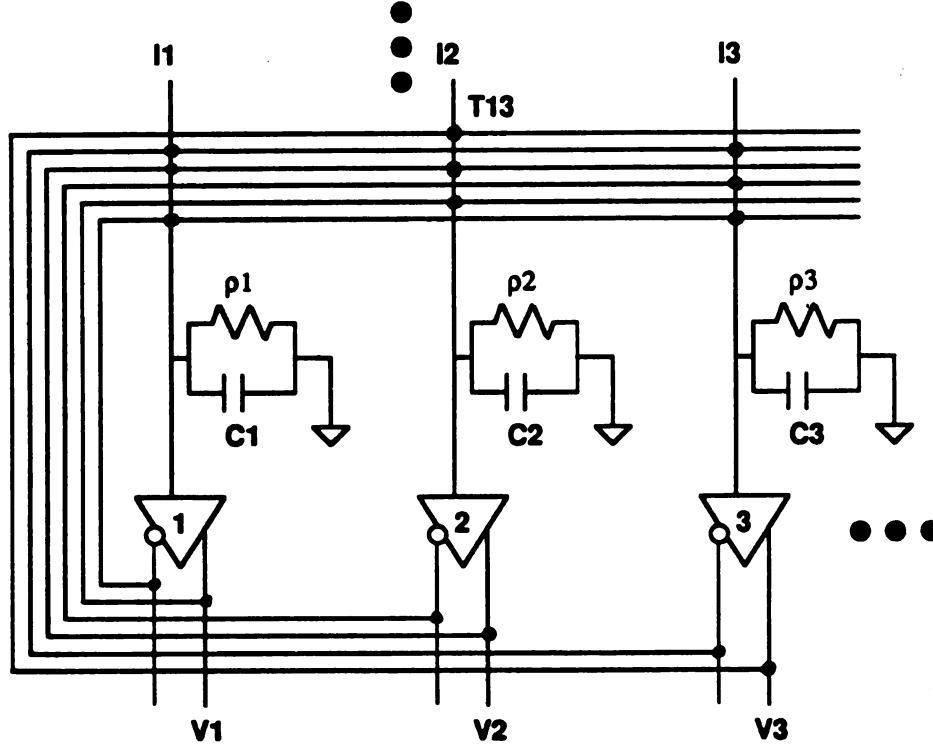


Figure 1: Hopfield-Tank Network Model.

and odd $[-g(-u)] = [g(u)]$ function, or sigmoid function [3]. An example is shown in Figure 2. The transfer function is commonly defined as

$$g(u) = \left(\frac{1}{2}\right) \left[1 + \tanh\left(\frac{u}{u_0}\right)\right].$$

The analog neuron model combines inputs and weights for each connection and sums these values. A bias term is added to the summation and then the summation value is utilized to calculate the output of each neuron.

The neuron has also been postulated as a discrete, or digital processor [8,9,15]. This model responds with a one or zero output, based on the summation of inputs multiplied by their respective weights. This model has also been utilized in the analysis and simulation of neural networks.

The digital model can be implemented easier than the analog version. Simple digital logic gates may be used in place of more complex op-amp or analog gate

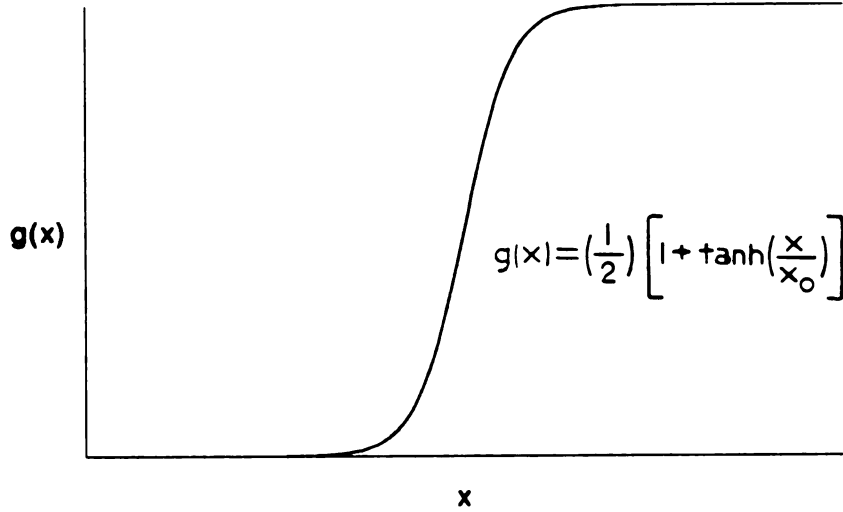


Figure 2: Sigmoid Transfer Function.

models. The number of transistors necessary for a standard op-amp (Texas Instruments uA741) is 22 [19]. For a standard digital gate (Texas Instruments SN5400) the number is 4 [20]. The digital version is faster at the individual component level due to the higher speed of digital electronics, 11 nanoseconds versus 10 microseconds [19,20]. However, the number of gate delays in a digital model is dependent on neuron implementation and would reduce the difference in speed.

Even with the listed advantages of the digital implementation, the network solution is not necessarily faster with digital neurons. It has been shown that an analog network would arrive at a solution faster than the digital network [14]. This would be true for equal time constants for each individual neuron. So it would be appropriate to ask, "For what networks would the analog neuron result in a more efficient computation? What is an ideal (or sufficient) neuron model?"

2.2.3 Interconnection Effects

A neural network utilizes massive parallelism perform computations. This parallelism is implemented through massive interconnection between individual neurons [2,14]. Each interconnection provides a signal which is multiplied by an appropriate weight and summed with the remaining products. This sum determines each neuron's output via the neuron response curve. Inhibitory and excitory inputs can be provided to each neuron. In some cases the inhibitory input is simply not excitory.

The effect of the number of interconnections to each neuron varies. The network response improves with additional inputs to each neuron. This is a result of each decision point having more information to apply toward that decision. However, additional interconnections can have a decreasing benefit as more are added [14].

The interconnections in typical artificial neural networks are symmetric and the interconnections to each neuron include a full stage or stages. A nonsymmetric network would connect parts of a stage or stages to a neuron. This type of network can solve specific problems but is too problem-specific and will not be utilized in this work. The number of stages that are interconnected to each neuron will be varied and the results used to determine the effect of increased interconnections.

The ease of implementation of a network is dependent upon the number of interconnections. Fewer interconnections reduce the complexity of the network. It is important to ask, "What is the optimum number of interconnections, and how many are necessary to solve a given class of problems?"

3 NEURON AND NEURAL NETWORK SIMULATION

3.1 Neuron Model

The neuron model utilized is similar to the model used by [15]. That model implemented a two-state output for each neuron. In this work, the neuron model is expanded to several states with different response curves.

3.1.1 Neuron Response Functions

Three types of response functions are investigated for each neuron. These include binary, ramp, and sigmoid functions. The ramp and sigmoid results will be compared to the binary model. This will provide a method of comparison for the different models.

The three response functions were chosen to model the biological neuron response. The binary function is the sigmoid function with an infinite gain. It is also the simplest model to implement. The ramp function is used because it is close to the actual sigmoid response of a neuron but does not have a complex math model, making it easy to implement. The sigmoid is used because it is the actual neuron response function. The simulations of each model will result in a comparison of the ability of each response function to arrive at a solution when used in the neural network.

By comparing the results for each neuron, the desired response can be chosen. The simpler models can be utilized if they provide the desired solution time, or the more complex function may be utilized if desired. The results will show the attributes of each of the functions.

For each response function, the gain of the curve is varied. By changing the gain of the curve, the effects of gain assumptions are determined. These variations in the neuron model provide coverage of factors that are important to the effectiveness of each neuron and the network.

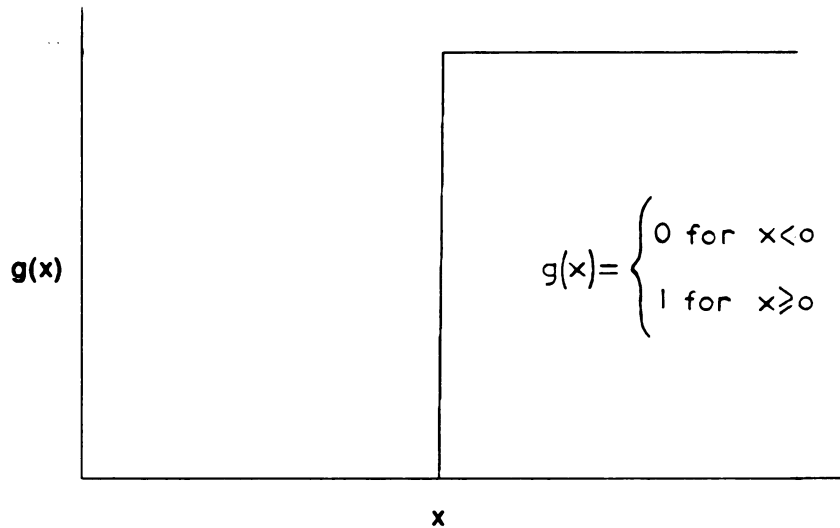


Figure 3: Binary Transfer Function.

The binary neuron response function is given in Figure 3. The two ramp neuron response functions (for both high and low gain) are shown in Figures 4 and 5. The two sigmoid neuron response functions (for both high and low gain) are shown in Figures 6 and 7. These figures give the actual response function coded in VHDL to implement the neuron. The neuron output values for each state (sum of input multiplied by weights) is listed in Appendix A.

The gain of each of the functions is chosen to have greater or lesser resolution than the problem set. That is, the decision of a neuron can be either the same as a binary response or the neuron decision can be at an intermediate point on the response function. Therefore, with a high gain function the neuron response will approximate a binary response. The low gain functions will allow neuron outputs that are not a one or a zero.

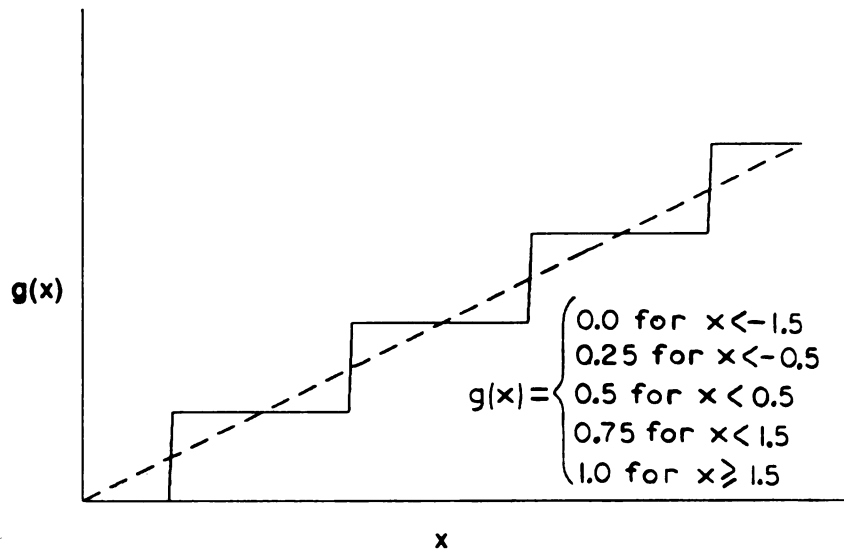


Figure 4: Low Gain Ramp Transfer Function.

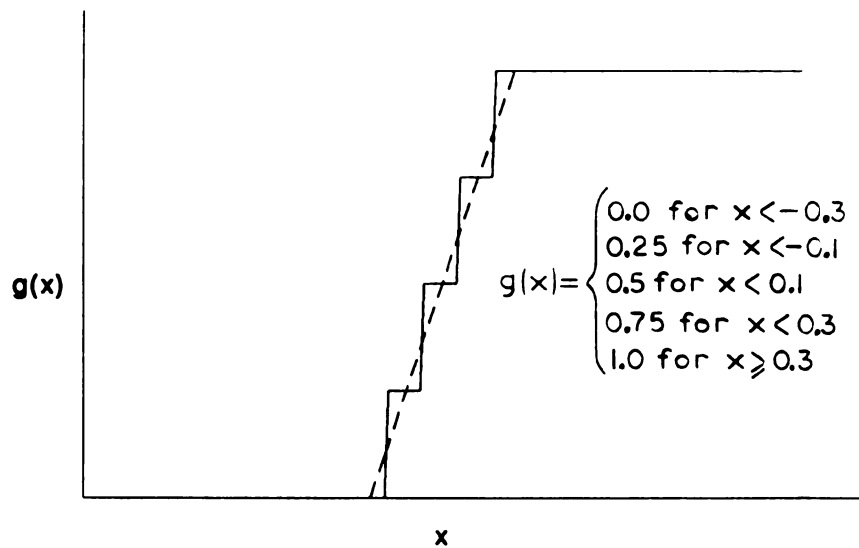


Figure 5: High Gain Ramp Transfer Function.

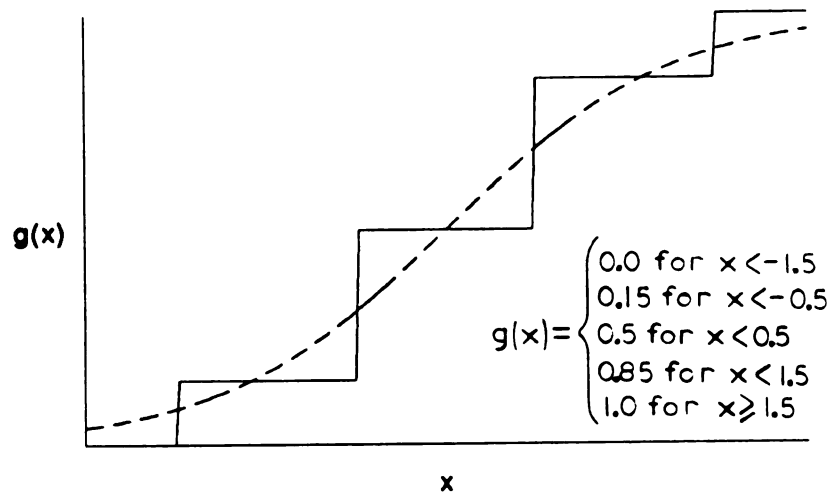


Figure 6: Low Gain Sigmoid Transfer Function.

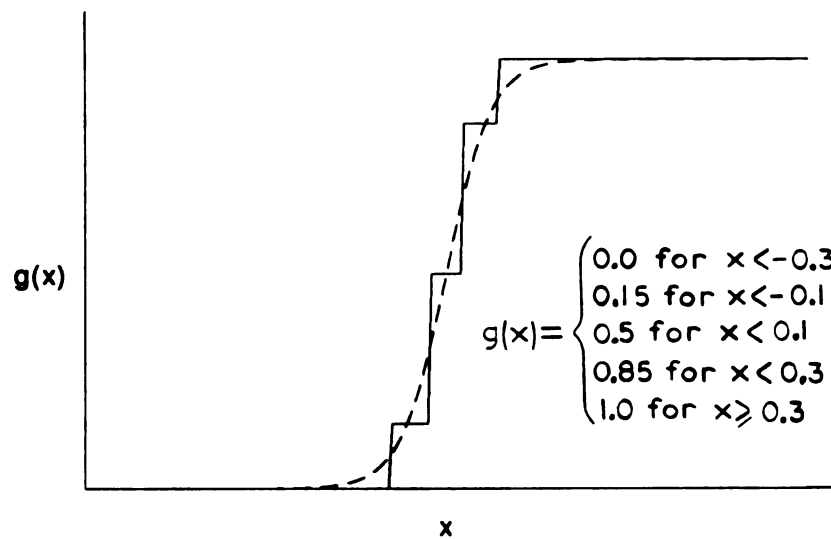


Figure 7: High Gain Sigmoid Transfer Function.

3.1.2 Neuron Interconnections

The number of interconnections for each neuron is varied to provide a comparison of the effect on the network. The first variations include interconnecting the stage that the current neuron is in and the previous stage. The inputs from these two stages are multiplied by their respective weights and summed at the neuron input. The neuron output is then computed. For the other two variations, the previous two or three stages and the current stage are connected to the current neuron.

The change in the number of interconnections between neurons is implemented by changing the following:

1. Modifying the initial constants for the network within the neural package module;
2. Modifying the memory module that holds weight factors for the neurons, and;
3. Modifying the bias term for each neuron response within the neuron modules.

The variations stated will allow visibility of the effect of interconnections in this network.

Also, the neuron models for the increased interconnection networks are similar but utilize a larger bias value to account for greater numbers of inputs.

3.1.3 Neuron VHDL Code

Appendix A contains the VHDL code for the neurons utilized in this work. The code implements the neurons by calculating the sum of each weight-input product. A series of **elsif** statements, which represent a lookup table, provide the output for the appropriate state.

The summation for each neuron is performed by a **function** call. This function computes the product of weight and input and sums all products. Each neuron has a **process** which is sensitive to the inputs. When a **transaction** occurs on an input

line, the neuron's process is initiated. The process stops when the output has been calculated until the next input transaction.

3.2 Network Model

The network model follows that of [15]. The network is setup as a processor with a neural network as a coprocessor. The neural network receives weights from memory as well as present neuron outputs and computes the solution. The results are then passed to the processor. A convergence sensor is employed to sense when the network has stabilized. The tolerance can be set within the package so it is easy to modify. A block diagram of the network is shown in Figure 8.

Appendix B contains the VHDL code utilized to initialize the network, implement the network, and set up a test bench.

3.2.1 Processor

The processor performs a management function for the network. The initial inputs and initial states are passed through the processor to the neurons. Results from the neural coprocessor are passed to the processor.

The processor module has a VHDL process which is sensitive to the outputs of the test bench module. When the test bench module sends the neuron outputs back to the processor, the processor's process begins. The process assigns each neuron's output to the input array for each neuron. This array is used by the neuron code to compute each neuron solution. The code utilizes if and loop statements to assign the correct neuron outputs to each neuron's input array.

3.2.2 Memory

A memory function is implemented to store the weights for each neuron. The weights are drawn from memory at the beginning of the simulation. They are not used again

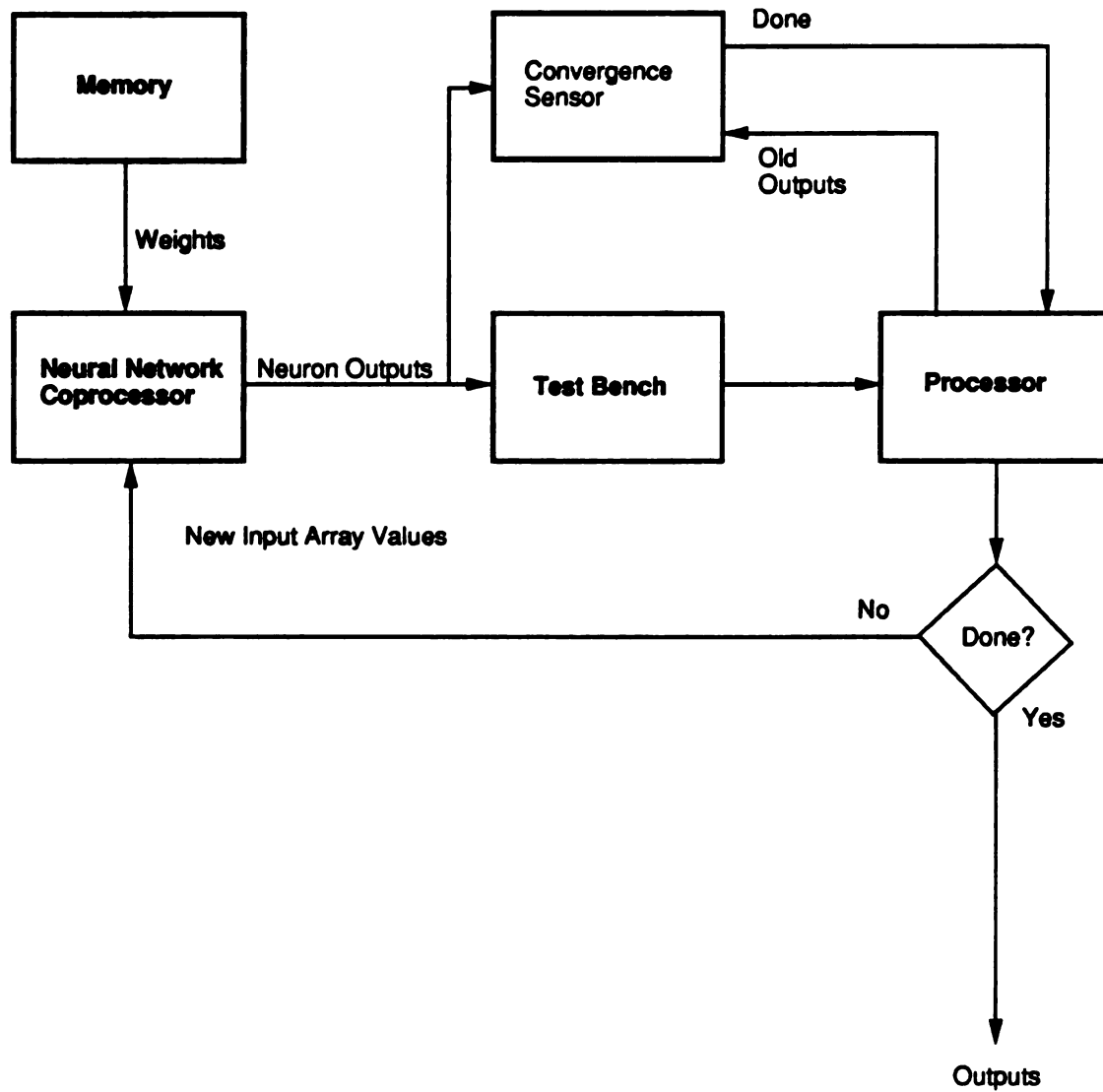


Figure 8: Network Block Diagram.

during the simulation, but could be updated if a learning algorithm was implemented.

The memory module has one signal assignment statement that sends the weight inputs for each neuron to the neuron. These weights are arranged in an array for each neuron. This array is multiplied by the input array for each neuron to compute the summation for each neuron.

3.2.3 Network

The network module uses a **generate** statement to define the network of neurons. The **generate** statement uses a loop to define an array of neurons in a single line of code.

3.2.4 Test Bench

A common method for running simulations in VHDL involves the use of a test bench. This test bench contains an **entity**, reference to modules in the library, and network stimulation inputs. The test bench also provides signals for observation at the output. The test bench used initializes the neuron states and begins the simulation.

The test bench module has a **process** that is sensitive to the neuron outputs. The test bench takes the neuron outputs and sends them to the processor to be assigned to the neuron inputs for the next cycle of calculations.

3.3 Simulation Inputs

Several data files are utilized to investigate the operation of the network. These data files contain the weight factors utilized for each of the problems simulated. The data (weights) are contained within the memory module. This is due to the inability of VHDL to utilize input files with real numbers as data. These data files provide a typical example of a problem to be solved by the neural network. The data files contain generated cost values which simulate network constraints. The network will seek a solution that reduces the cost along the path from start to finish.

The values utilized in the data sets consist of seven different values. The values, 0.0 and -9.0, either turn off or enable a neuron. This allows a starting point and ending point to be defined for the network. The first and last stages contain these points. The other five values are weights utilized for each neuron. These values will enable a neuron when the weight value is less than the bias value of the neuron.

Appendix C contains the data files used for the simulations.

3.4 Performance Measurement Technique

The performance of each neuron model is measured by the response time of the network. Each neuron model is implemented with the base network and the results are compared. The results include the final state (if one is reached) and the number of cycles necessary to reach that state. The final state must be a valid solution or the compute time will be meaningless.

Because the time between each update of the network state is 10 nanoseconds (10 nanoseconds was selected to represent five gate delays in 1 to 2 micron technology), this is the amount of time for one cycle. Using a time other than 10 nanoseconds would provide a more exact simulation for a particular network and particular neurons. To determine the number of cycles for convergence of the network, the setup time is subtracted from the completion time, and this result is divided by the cycle time. In this report the setup time is 4 nanoseconds, which is the time to feed the weights and initial conditions to the neurons.

3.5 Procedure for Simulation

The following steps are taken for each simulation run.

Step 1: Determine Neuron Output. The appropriate response curve and the discrete output levels are determined for each neuron.

Step 2: Code Neuron Output. The output levels are coded into the VHDL neuron model as a lookup table.

Step 3: Determine Weights and Bias. The weights for each input for each neuron are determined. The bias for the neuron response is also determined. The weights are then coded into the VHDL memory model and neuron model.

Step 4: Edit VHDL Network File. Edit the network memory and initialization code to utilize the appropriate neuron model, including weights and bias. The current neuron model is added to the network code and then the file is ready to be simulated.

Step 5: Simulate VHDL File. Next, the file is put into the VHDL analyzer and the code is compiled and used for simulation. The report output is printed as the last part of this step.

Step 6: Analyse Output. The output file is analyzed to verify that a valid solution has been computed. If no valid solution has been calculated, return to step 3 and repeat process. If a valid solution is computed, the number of cycles is calculated and the final state is printed.

4 SIMULATION RESULTS

4.1 Neuron Model Results

The neuron model results show that the binary response function provided the fastest network solution time. The ramp and sigmoid response functions were as fast when the gain was set very high approximating a binary response. When the gain was set to a lower value, the network would oscillate.

Many simulations were performed with the network and various problem sets. The weights and bias values were varied to arrive at a network and problem set that had a solution. The majority of the simulations resulted in oscillation or a solution set that was not valid.

The network would arrive at invalid solutions or oscillate for two reasons. The network was found to be very susceptible to the value of the gain of the neurons and to the value of the neuron bias used. The neuron bias, if not set to a value less than the weights in the nonoptimal path, results in many neurons being enabled incorrectly (including neurons not in optimal path). If the bias is set to a value more than some weights in the optimal path, then no solution will exist for some stages of the network.

The oscillation of the network occurred when the gain of the graded response neuron models was set to a low value. A high value of gain results in a response function that closely approximates a binary function with either a 0.0 or 1.0 output. Therefore, a low value of gain is a value that allows a resultant neuron decision to be other than 0.0 or 1.0. For example, if the sum of products at the neuron input is close to 0.0, then the resultant neuron decision is very close to 0.5.

This neuron then provides this output multiplied by a negative weight, to other neurons. If more than one neuron has other than a 0.0 or 1.0 output, other neurons are highly negatively driven. This results in a 0.0 output for many neurons in one cycle. The following cycle then contains many neurons with a 1.0 output (because

Table 1: Results of Neuron Model Simulations.

Response Function	Solution Time
Binary	15 Cycles
High Gain Ramp	15 Cycles
High Gain Sigmoid	15 Cycles
Low Gain Ramp	No Solution
Low Gain Sigmoid	No Solution

there is no negative bias). The network will continue to oscillate in this way because the low gain does not provide decision-making capability.

The high gain value provides only two decision points, so that each neuron has good decision making capability. This results in a solution in the fastest time. The weights and neuron bias values must be set to allow the binary neurons to arrive at a solution.

The problem set utilized allows the binary response network to arrive at a decision and also allows the graded response networks to arrive at a solution with the proper bias and gain.

The results are shown in Table 1. The neuron model response and number of cycles needed to converge are given. Appendix D contains the actual neuron outputs with times for examination.

4.2 Interconnection Pattern Results

The network results for varied interconnection patterns showed an increase in speed for increased numbers of interconnections. The increased numbers of interconnections provided each neuron with a greater knowledge of the problem set. This allows the

first stages to compute faster. The starting point is used by the first stages as part of the calculation and increased interconnections allowed more stages to use this known value, thus decreasing the time for these stages to compute.

The final stage also computed faster with more than two stage interconnection. This is due to the last stage having information from previous stages which have already reached a solution, in addition to the previous stage.

The increased number of interconnections allowed the low gain sigmoid neuron network to compute a solution, although this was not possible with only two stages of interconnections. The low gain ramp neuron network oscillated as with the two stages of interconnections. The additional knowledge combined with the better decision making capability of the sigmoid function resulted in a solution where the ramp function did not.

The difference in solution time between the low gain sigmoid and the binary response networks decreased as more interconnections were added. This is due to the added knowledge of each neuron overcoming the amount of ability of each response function to make a decision.

Table 2 shows the simulation results for the binary neuron network. Table 3 shows results for the low gain ramp neuron network. Table 4 shows the results for the low gain sigmoid neuron network. Appendix D contains the actual neuron outputs with times.

A second problem was simulated with the varied number of interconnected stages. This second problem contained a sub-optimal path between three stages that was not part of the overall optimal path. This would show the capability of the network to handle additional problems sets. Many applications of a neural network include sub-optimal paths, for which fast, near optimal solutions are desired.

This additional problem set utilized a sub-optimal path between stages three, four, and five. The sub-optimal path is therefore two stages long. The two- stage

Table 2: Results of Varied Interconnections with Binary Neurons.

Interconnections	Solution Time
Two Stages	15 Cycles
Three Stages	13 Cycles
Four Stages	9 Cycles

Table 3: Results of Varied Interconnections with Low Gain Ramp Neurons.

Interconnections	Solution Time
Two Stages	No Solution
Three Stages	No Solution
Four Stages	No Solution

Table 4: Results of Varied Interconnections with Low Gain Sigmoid Neurons.

Interconnections	Solution Time
Two Stages	No Solution
Three Stages	19 Cycles
Four Stages	13 Cycles

Table 5: Results of Varied Interconnections and Nonoptimal Path.

Interconnections	No Solution
Two Stages	No Solution
Three Stages	13 Cycles
Four Stages	9 Cycles

interconnected network could not compute a solution for this problem set. The three- and four-stage interconnected networks computed a solution in the same time as the problem set that did not include a sub-optimal path.

The two-stage network cannot arrive at a decision in the stage that has the two locally optimal paths. This stage sees two paths of a weight that is less than the neuron bias, thus enabling two neurons in this stage. The network solution requires that only one neuron is enabled per stage. So the effect of the sub-optimal path is to cause the network to oscillate at the stage where two locally optimal paths exist. The larger two networks have enough interconnections, or information, to see a larger solution set and resolve which of the two paths is the correct choice.

The three- and four-stage networks computed a solution in the same time as the problem set without the sub-optimal path. This shows that a solution can be computed for a network with an optimal path in a certain time and then sub-optimal paths can be introduced with the same solution time expected. The number of interconnected stages necessary for the problem set is the only variable that needs to be determined.

The results for this additional problem are shown in Table 5. The results of the simulation are given in Appendix D for examination.

5 DISCUSSION

Based upon the opening text, the simulation results for the varied neuron models were not expected. The graded response neuron models were not faster than the binary neuron models. In some cases the graded response neuron models could not achieve a solution.

The binary model proved to be easy to implement and simulate, whereas the graded response models were not. The graded response models were more difficult to implement due to variables such as gain and the number of discrete steps. These variables required several trials to achieve a valid solution.

The weights and bias of the neurons were revised many times during the simulation of each different network. This was necessary to make the network converge to a valid solution in a minimum amount of time. The network must be adjusted in this way for each new problem, as each problem set must utilize different weight values and different bias values. The bias values could be set at a particular value, if the weight values were multiplied by some factor. This would put each problem set within a particular range and allow the same bias value to be used for each simulation.

The majority of the results achieved through the successive trials were oscillating networks. The bias and/or weights would combine to drive all neurons to a high state, then a low state, and so on. By use of a problem that has only one optimal path, the network would converge. The optimal path must also contain each of the locally optimal paths. Locally optimal paths not in the optimal path could only be present if the number of interconnected stages could effectively 'step over' this sub-optimal solution in search of the overall optimal path.

The network would not converge if the gain of the neuron response function was low enough to allow the neurons to make 'fuzzy' or intermediate decisions. This caused the network to have final solutions that did not meet the validity test, that is, not all neurons had a one or zero output. This occurred for a low gain, when decision

points did not fall on the ends at 1.0 or 0.0, but in between.

The neuron model first coded into VHDL does not implement all factors of the neuron as proposed by [17]. The capacitance term in the network was not implemented. Additionally, the network does not accept negative input values. The input values are either positive or zero. The models account for negative inputs with zero input, but the capacitance function is not modeled.

Some further simulations were performed to determine if the simplified neuron model caused the results to be other than expected. Each of the neuron response functions was coded into a neuron model and a network that employed a form of capacitance. Because VHDL is intended for digital hardware simulations, no capacitor function is available. A capacitor was implemented by combining the previous input value and the present input value. This is done for each neuron at the neuron input. The VHDL code for this function is shown in Appendix A.

The capacitor implemented takes the past and present input and uses the following calculation

$$NeuronInput = 0.6(Present) + 0.4(Past).$$

This calculation limits a graded neuron output from going 1.0 to 0.0, and vice versa. This allows the network to retain some memory and keep neurons from oscillating in certain cases. The past value was also limited to a maximum absolute value equal to the value of the neuron bias used. This prevents a highly negative input from keeping the neuron off at all times.

The simulations were performed on the modified network and results taken. The modified network results are shown in Table 6. The network did converge in one additional case. However, the network did oscillate for the low gain ramp function as before. The low gain sigmoid function converged to invalid solutions, but did so relatively close to the values of the valid solution. For example, many stages with the low gain sigmoid response had a result of 0.85, and the valid result was 1.0. The

Table 6: Results for Network with Capacitance Function.

Response Function	Solution Time
Binary	15 Cycles
High Gain Ramp	16 Cycles
High Gain Sigmoid	16 Cycles
Low Gain Ramp	No Solution
Low Gain Sigmoid	26 Cycles

result could be interpreted as a 1.0 because the other values in these stages were 0.0.

This network took longer to converge than the binary network, but did converge. The capacitance allowed the network to retain memory and arrive at a solution.

6 CONCLUSIONS

The work performed showed that neural networks and their solution time is very strongly dependent upon the neuron model and the interconnection pattern. The speed of the network can be increased by modifying the neuron model and increasing the number of interconnections.

The best network in the cases studied in this work utilizes neuron models with a response function approximating a binary neuron. This neuron model provides the fastest solution because no intermediate cycles were needed to converge. The graded response neuron models require extra cycles to reach the decision point because of intermediate 'decisions' by each neuron made during simulation.

By increasing the number of interconnections, the difference in speed between the binary and low gain sigmoid function is decreased. This decreases the need to employ a binary neuron. A neuron with a moderate to high gain could be employed in a network with many interconnections with little or no penalty in speed.

The sigmoid function would also provide more information to a learning algorithm. The learning algorithm can use this information to determine the magnitude of the neuron input. The learning algorithm can also reset the gain within this network to allow modest gain at the start and increase the gain as a solution is near.

Additionally, the best network contains many interconnected stages. This also results in a faster solution time. Additional interconnected stages also eliminate solutions that are not part of the overall optimal path because of the larger picture seen by each neuron.

By adding a capacitance term to the input of each neuron, it was shown that the time to reach a solution for the network was increased. However the network would cease to oscillate and, if interpreted correctly, could provide a solution where a simplified network would oscillate.

The hardware available today can be employed in a digital model of a neuron

for the best network. The slower analog components and slower speed of the graded response networks does not compare to the binary neuron implemented with digital gates. The benefit of the analog network would have be to great in the area of learning algorithms to make the analog network desirable.

As research on learning algorithms grows, the optimal neuron model may be a graded response function with the gain set to a high enough value to allow a small degree of the switching decision to be on the response function and utilized in the learning algorithm. It may also be advantagous for the learning algorithm to revise the gain of the neuron model. Although this is not employed in biological neural networks, an artificial neural network could put this increased capability to use.

APPENDIX A

A.1 Neuron Model VHDL Code

The response curves are modeled in VHDL by incorporating a simple lookup table. Each table was copied to the network code and a simulation run was made. Figures 9 through 18 show the VHDL code for each neuron model.

– Model of a Neuron Element

```
use work.Neural_Package.all;
entity Neural_element is
  port (
    Stimulus : in Input_Array;
    Weights : in Input_Array;
    Output : out Real := 0.0);
end Neural_Element;

architecture behavior of neural_element is
begin
  NeuralProcess :
  process(Stimulus'Transaction)
    variable Sum : Real;
  begin
    Sum := CalculateSum(Stimulus, Weights) + 1.9;
    if Sum > (0.0) then
      Output <= 1.0 after 4 ns;
    else
      Output <= 0.0 after 4 ns;
    end if;
  end process;
end behavior;
--
```

Figure 9: VHDL Code for Binary Response Neuron.

– Model of a Neuron Element

```
use work.Neural_Package.all;
entity Neural_element is
  port (
    Stimulus : in Input_Array;
    Weights : in Input_Array;
    Output : out Real := 0.0);
end Neural_Element;

architecture behavior of neural_element is
  component Capacitor port (Past : in Real; Present : out Real);
  end component;
  for all : Capacitor use entity work.Capacitor(behavior);
  signal Past_Out : Real;
  signal Present_In : Real;
begin
  cap : Capacitor port map (Past_Out, Present_In);
  NeuralProcess :
  process(Stimulus'Transaction)
    variable Sum : Real;
    variable Cap_Value : Real;
    variable In_Voltage: Real;
  begin
    Sum := CalculateSum(Stimulus, Weights) + 1.9;
    Cap_Value := Present_In;
    if Cap_Value < -1.9 then Cap_Value := -1.9;
    end if;
    In_Voltage := (0.6*Sum) + (0.4*Cap_Value);
    if In_Voltage > (0.0) then
      Output <= 1.0 after 4 ns;
    else
      Output <= 0.0 after 4 ns;
    end if;
    Past_Out <= Sum;
  end process;
end behavior;
```

Figure 10: VHDL Code for Binary Response Neuron with Capacitance.

– Model of a Neuron Element

```
use work.Neural_Package.all;
entity Neural_element is
  port (
    Stimulus : in Input_Array;
    Weights : in Input_Array;
    Output : out Real := 0.0);
end Neural_Element;

architecture behavior of neural_element is
begin
  NeuralProcess:
  process(Stimulus'Transaction)
    variable Sum : Real;
  begin
    Sum := CalculateSum(Stimulus, Weights) + 1.9;
    if Sum < (-1.5) then
      Output <= 0.0 after 4 ns;
    elsif Sum < (-0.5) then
      Output <= 0.25 after 4 ns;
    elsif Sum < (0.5) then
      Output <= 0.50 after 4 ns;
    elsif Sum < (1.5) then
      Output <= 0.75 after 4 ns;
    else
      Output <= 1.0 after 4 ns;
    end if;
  end process;
end behavior;
```

Figure 11: VHDL Code for Low Gain Ramp Response Neuron.

```

-- Model of a Neuron Element

use work.Neural_Package.all;
entity Neural_element is
  port (
    Stimulus : in Input_Array;
    Weights : in Input_Array;
    Output : out Real := 0.0);
end Neural_Element;

architecture behavior of neural_element is
  component Capacitor port (Past : in Real; Present : out Real);
  end component;
  for all : Capacitor use entity work.Capacitor(behavior);
  signal Past_Out : Real;
  signal Present_In : Real;
begin
  cap : Capacitor port map (Past_Out, Present_In);
  NeuralProcess :
  process(Stimulus'Transaction)
    variable Sum : Real;
    variable Cap_Value : Real;
    variable In_Voltage: Real;
  begin
    Sum := CalculateSum(Stimulus, Weights) + 1.9;
    Cap_Value := Present_In;
    if Cap_Value < -1.9 then Cap_Value := -1.9;
    end if;
    In_Voltage := (0.6*Sum) + (0.4*Cap_Value);
    if In_Voltage < (-1.5) then
      Output <= 0.0 after 4 ns;
    elsif In_Voltage < (-0.5) then
      Output <= 0.25 after 4 ns;
    elsif In_Voltage < (0.5) then
      Output <= 0.50 after 4 ns;
    elsif In_Voltage < (1.5) then
      Output <= 0.75 after 4 ns;
    else Output <= 1.0 after 4 ns;
    end if;
    Past_Out <= Sum;
  end process;
end behavior;

```

Figure 12: VHDL Code for Low Gain Ramp Response Neuron with Capacitance.

– Model of a Neuron Element

```
use work.Neural_Package.all;
entity Neural_element is
  port (
    Stimulus : in Input_Array;
    Weights : in Input_Array;
    Output : out Real := 0.0);
end Neural_Element;

architecture behavior of neural_element is
begin
  NeuralProcess:
  process(Stimulus'Transaction)
    variable Sum : Real;
  begin
    Sum := CalculateSum(Stimulus, Weights) + 1.9;
    if Sum < (-0.3) then
      Output <= 0.0 after 4 ns;
    elsif Sum < (-0.1) then
      Output <= 0.25 after 4 ns;
    elsif Sum < (0.1) then
      Output <= 0.50 after 4 ns;
    elsif Sum < (0.3) then
      Output <= 0.75 after 4 ns;
    else
      Output <= 1.0 after 4 ns;
    end if;
  end process;
end behavior;
```

Figure 13: VHDL Code for High Gain Ramp Response Neuron.

```

-- Model of a Neuron Element

use work.Neural_Package.all;
entity Neural_element is
  port (
    Stimulus : in Input_Array;
    Weights : in Input_Array;
    Output : out Real := 0.0);
end Neural_Element;

architecture behavior of neural_element is
  component Capacitor port (Past : in Real; Present : out Real);
  end component;
  for all : Capacitor use entity work.Capacitor(behavior);
  signal Past_Out : Real;
  signal Present_In : Real;
begin
  cap : Capacitor port map (Past_Out, Present_In);
  NeuralProcess :
  process(Stimulus'Transaction)
    variable Sum : Real;
    variable Cap_Value : Real;
    variable In_Voltage : Real;
  begin
    Sum := CalculateSum(Stimulus, Weights) + 1.9;
    Cap_Value := Present_In;
    if Cap_Value < -1.9 then Cap_Value := -1.9;
    end if;
    In_Voltage := (0.6*Sum) + (0.4*Cap_Value);
    if In_Voltage < (-0.3) then
      Output <= 0.0 after 4 ns;
    elsif In_Voltage < (-0.1) then
      Output <= 0.25 after 4 ns;
    elsif In_Voltage < (0.1) then
      Output <= 0.50 after 4 ns;
    elsif In_Voltage < (0.3) then
      Output <= 0.75 after 4 ns;
    else Output <= 1.0 after 4 ns;
    end if;
    Past_Out <= Sum;
  end process;
end behavior;

```

Figure 14: VHDL Code for High Gain Ramp Response Neuron with Capacitance.

```

-- Model of a Neuron Element

use work.Neural_Package.all;
entity Neural_element is
  port (
    Stimulus : in Input_Array;
    Weights : in Input_Array;
    Output : out Real := 0.0);
end Neural_Element;

architecture behavior of neural_element is
begin
  NeuralProcess:
  process(Stimulus*Transaction)
    variable Sum : Real;
  begin
    Sum := CalculateSum(Stimulus, Weights) + 1.9;
    if Sum < (-1.5) then
      Output <= 0.0 after 4 ns;
    elsif Sum < (-0.5) then
      Output <= 0.15 after 4 ns;
    elsif Sum < (0.5) then
      Output <= 0.50 after 4 ns;
    elsif Sum < (1.5) then
      Output <= 0.85 after 4 ns;
    else
      Output <= 1.0 after 4 ns;
    end if;
  end process;
end behavior;

```

Figure 15: VHDL Code for Low Gain Sigmoid Response Neuron.

```

-- Model of a Neuron Element

use work.Neural_Package.all;
entity Neural_element is
  port (
    Stimulus : in Input_Array;
    Weights : in Input_Array;
    Output : out Real := 0.0);
end Neural_Element;

architecture behavior of neural_element is
  component Capacitor port (Past : in Real; Present : out Real);
  end component;
  for all : Capacitor use entity work.Capacitor(behavior);
  signal Past_Out : Real;
  signal Present_In : Real;
begin
  cap : Capacitor port map (Past_Out, Present_In);
  NeuralProcess :
  process(Stimulus'Transaction)
    variable Sum : Real;
    variable Cap_Value : Real;
    variable In_Voltage: Real;
  begin
    Sum := CalculateSum(Stimulus, Weights) + 1.9;
    Cap_Value := Present_In;
    if Cap_Value < -1.9 then Cap_Value := -1.9;
    end if;
    In_Voltage := (0.6*Sum) + (0.4*Cap_Value);
    if In_Voltage < (-1.5) then
      Output <= 0.0 after 4 ns;
    elsif In_Voltage < (-0.5) then
      Output <= 0.15 after 4 ns;
    elsif In_Voltage < (0.5) then
      Output <= 0.50 after 4 ns;
    elsif In_Voltage < (1.5) then
      Output <= 0.85 after 4 ns;
    else Output <= 1.0 after 4 ns;
    end if;
    Past_Out <= Sum;
  end process;
end behavior;

```

Figure 16: VHDL Code for Low Gain Sigmoid Response Neuron with Capacitance.

– Model of a Neuron Element

```
use work.Neural_Package.all;
entity Neural_element is
  port (
    Stimulus : in Input_Array;
    Weights : in Input_Array;
    Output : out Real := 0.0);
end Neural_Element;

architecture behavior of neural_element is
begin
  NeuralProcess:
  process(Stimulus'Transaction)
    variable Sum : Real;
  begin
    Sum := CalculateSum(Stimulus, Weights) + 1.9;
    if Sum < (-0.3) then
      Output <= 0.0 after 4 ns;
    elsif Sum < (-0.1) then
      Output <= 0.15 after 4 ns;
    elsif Sum < (0.1) then
      Output <= 0.50 after 4 ns;
    elsif Sum < (0.3) then
      Output <= 0.85 after 4 ns;
    else
      Output <= 1.0 after 4 ns;
    end if;
  end process;
end behavior;
```

Figure 17: VHDL Code for High Gain Sigmoid Response Neuron.

```

-- Model of a Neuron Element

use work.Neural_Package.all;
entity Neural_element is
  port (
    Stimulus : in Input_Array;
    Weights : in Input_Array;
    Output : out Real := 0.0);
end Neural_Element;

architecture behavior of neural_element is
  component Capacitor port (Past : in Real; Present : out Real);
  end component;
  for all : Capacitor use entity work.Capacitor(behavior);
  signal Past_Out : Real;
  signal Present_In : Real;
begin
  cap : Capacitor port map (Past_Out, Present_In);
  NeuralProcess :
  process(Stimulus'Transaction)
    variable Sum : Real;
    variable Cap_Value : Real;
    variable In_Voltage : Real;
  begin
    Sum := CalculateSum(Stimulus, Weights) + 1.9;
    Cap_Value := Present_In;
    if Cap_Value < -1.9 then Cap_Value := -1.9;
    end if;
    In_Voltage := (0.6*Sum) + (0.4*Cap_Value);
    if In_Voltage < (-0.3) then
      Output <= 0.0 after 4 ns;
    elsif In_Voltage < (-0.1) then
      Output <= 0.15 after 4 ns;
    elsif In_Voltage < (0.1) then
      Output <= 0.50 after 4 ns;
    elsif In_Voltage < (0.3) then
      Output <= 0.85 after 4 ns;
    else Output <= 1.0 after 4 ns;
    end if;
    Past_Out <= Sum;
  end process;
end behavior;

```

Figure 18: VHDL Code for High Gain Sigmoid Response Neuron with Capacitance.

APPENDIX B

B.1 Network VHDL Code

The network is simulated by use of the VHDL code shown in Figure 19. This code contains each of the parts of the network and the code necessary to implement them.

```

-- Neural Package. Contains network parameters and function Sum.

package Neural_Package is
  constant Network_Stages : natural := 8;
  constant Neuron_Interconn : natural := 12;
  constant Neurons_per_Stage : natural := 6;
  constant Total_Neurons : natural := 48;
  constant Tolerance : Real := 0.1;
  type Neural_Array is array (Natural range 1 to Total_Neurons) of Real;
  type Input_Array is array (Natural range 1 to Neuron_Interconn) of Real;
  type Weights_Matrix is array (Natural range 1 to Total_Neurons) of Input_Array;
  type Stimulus_Matrix is array (Natural range 1 to Total_Neurons) of Input_Array;

  function CalculateSum (
    Stimulus : Input_Array;
    Weights : Input_Array)
    return Real;

end Neural_Package;

package body Neural_Package is
  function CalculateSum (
    Stimulus : Input_Array;
    Weights : Input_Array)
    return Real
  is
    variable Sum : Real := 0.0;
  begin
    for I in 1 to Neuron_Interconn loop
      Sum := Sum + Stimulus(I) * Weights(I);
    end loop;
    return Sum;
  end CalculateSum;
end Neural_Package;

-- Model of a Capacitor Element

use work.Neural_Package.all;

entity Capacitor is
  port (Past : in real; Present : out real := 0.0);
end Capacitor;

architecture behavior of Capacitor is
begin
  CapProcess :
  process(Past*Transaction)
  begin
    Present <= Past;
  end process;
end behavior;

```

Figure 19: VHDL Code for Neural Network.

– Neural Package. Contains network parameters and function Sum.

```
package Neural_Package is
  constant Network_Stages : natural := 8;
  constant Neuron_Interconn : natural := 12;
  constant Neurons_per_Stage : natural := 6;
  constant Total_Neurons : natural := 48;
  constant Tolerance : Real := 0.1;
  type Neural_Array is array (Natural range 1 to Total_Neurons) of Real;
  type Input_Array is array (Natural range 1 to Neuron_Interconn) of Real;
  type Weights_Matrix is array (Natural range 1 to Total_Neurons) of Input_Array;
  type Stimulus_Matrix is array (Natural range 1 to Total_Neurons) of Input_Array;
  function CalculateSum (
    Stimulus : Input_Array;
    Weights : Input_Array)
  return Real;
end Neural_Package;
```

```
package body Neural_Package is
  function CalculateSum (
    Stimulus : Input_Array;
    Weights : Input_Array)
  return Real
  is
    variable Sum : Real := 0.0;
  begin
    for I in 1 to Neuron_Interconn loop
      Sum := Sum + Stimulus(I) * Weights(I);
    end loop;
    return Sum;
  end CalculateSum;
end Neural_Package;
```

– Model of a Capacitor Element

```
use work.Neural_Package.all;
entity Capacitor is
  port (Past : in real; Present : out real := 0.0);
end Capacitor;
```

```
architecture behavior of Capacitor is
begin
  CapProcess :
  process(Past'Transaction)
  begin
    Present <= Past;
  end process;
end behavior;
```

Figure 19. (continued).

- This module is meant to be used as a memory to hold the input values
- (Weights). A centralized memory is visualised as the change required
- for a different network would be minimal this way.

```

use work.Neural_Package.all;
entity memory is
port (
    memory_output : out Weights_Matrix);
end memory;

architecture memory_arch of memory is
begin
    process
    begin
        memory_output <= (
            -- Neuron Stage 1
            (0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0),
            (-9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0),
            (-9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0),
            (-9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0),
            (-9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0),
            (-9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0),
            -- Neuron Stage 2
            (-2.5, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, -5.0, -5.0, -5.0, -5.0, -5.0),
            (-3.5, 0.0, 0.0, 0.0, 0.0, 0.0, -5.0, 0.0, -5.0, -5.0, -5.0, -5.0),
            (-0.5, 0.0, 0.0, 0.0, 0.0, 0.0, -5.0, -5.0, 0.0, -5.0, -5.0, -5.0),
            (-4.5, 0.0, 0.0, 0.0, 0.0, 0.0, -5.0, -5.0, -5.0, 0.0, -5.0, -5.0),
            (-2.5, 0.0, 0.0, 0.0, 0.0, 0.0, -5.0, -5.0, -5.0, -5.0, 0.0, -5.0),
            (-4.5, 0.0, 0.0, 0.0, 0.0, 0.0, -5.0, -5.0, -5.0, -5.0, -5.0, 0.0),
            --Neuron Stage 3
            (-2.5, -4.5, -4.5, -3.5, -2.5, -3.5, 0.0, -5.0, -5.0, -5.0, -5.0, -5.0),
            (-4.5, -2.5, -3.5, -4.5, -3.5, -2.5, -5.0, 0.0, -5.0, -5.0, -5.0, -5.0),
            (-2.5, -2.5, -4.5, -4.5, -3.5, -3.5, -5.0, -5.0, 0.0, -5.0, -5.0, -5.0),
            (-2.5, -2.5, -1.5, -4.5, -2.5, -3.5, -5.0, -5.0, -5.0, 0.0, -5.0, -5.0),
            (-4.5, -4.5, -2.5, -2.5, -2.5, -2.5, -5.0, -5.0, -5.0, -5.0, 0.0, -5.0),
            (-2.5, -2.5, -3.5, -4.5, -3.5, -2.5, -5.0, -5.0, -5.0, -5.0, -5.0, 0.0),
            -- Neuron Stage 4
            (-2.5, -3.5, -4.5, -4.5, -3.5, -2.5, 0.0, -5.0, -5.0, -5.0, -5.0, -5.0),
            (-2.5, -3.5, -3.5, -3.5, -4.5, -2.5, -5.0, 0.0, -5.0, -5.0, -5.0, -5.0),
            (-2.5, -3.5, -4.5, -2.5, -2.5, -2.5, -5.0, -5.0, 0.0, -5.0, -5.0, -5.0),
            (-3.5, -3.5, -4.5, -2.5, -2.5, -2.5, -5.0, -5.0, -5.0, 0.0, -5.0, -5.0),
            (-4.5, -3.5, -2.5, -0.5, -2.5, -4.5, -5.0, -5.0, -5.0, -5.0, 0.0, -5.0),
            (-4.5, -3.5, -2.5, -2.5, -4.5, -4.5, -5.0, -5.0, -5.0, -5.0, -5.0, 0.0),
            -- Neuron Stage 5
            (-2.5, -2.5, -2.5, -4.5, -3.5, -2.5, 0.0, -5.0, -5.0, -5.0, -5.0, -5.0),
            (-4.5, -4.5, -2.5, -4.5, -1.5, -4.5, -5.0, 0.0, -5.0, -5.0, -5.0, -5.0),
            (-4.5, -2.5, -2.5, -3.5, -3.5, -4.5, -5.0, -5.0, 0.0, -5.0, -5.0, -5.0),
            (-2.5, -4.5, -2.5, -3.5, -3.5, -4.5, -5.0, -5.0, -5.0, 0.0, -5.0, -5.0),
            (-4.5, -3.5, -2.5, -3.5, -3.5, -4.5, -5.0, -5.0, -5.0, -5.0, 0.0, -5.0),
            (-4.5, -3.5, -4.5, -4.5, -3.5, -3.5, -5.0, -5.0, -5.0, -5.0, -5.0, 0.0),
        );
    end process;
end architecture;

```

Figure 19. (continued).

```

-- Neuron Stage 6
(-4.5, -3.5, -2.5, -2.5, -4.5, -3.5, 0.0, -5.0, -5.0, -5.0, -5.0, -5.0),
(-4.5, -2.5, -3.5, -4.5, -2.5, -2.5, -5.0, 0.0, -5.0, -5.0, -5.0, -5.0),
(-4.5, -3.5, -3.5, -4.5, -2.5, -2.5, -5.0, -5.0, 0.0, -5.0, -5.0, -5.0),
(-2.5, -4.5, -2.5, -2.5, -4.5, -4.5, -5.0, -5.0, -5.0, 0.0, -5.0, -5.0),
(-3.5, -3.5, -4.5, -2.5, -3.5, -4.5, -5.0, -5.0, -5.0, -5.0, 0.0, -5.0),
(-3.5, -0.5, -3.5, -2.5, -2.5, -2.5, -5.0, -5.0, -5.0, -5.0, -5.0, 0.0),
-- Neuron Stage 7
(-2.5, -3.5, -4.5, -2.5, -3.5, -2.5, 0.0, -5.0, -5.0, -5.0, -5.0, -5.0),
(-4.5, -2.5, -3.5, -2.5, -2.5, -2.5, -5.0, 0.0, -5.0, -5.0, -5.0, -5.0),
(-4.5, -3.5, -2.5, -2.5, -3.5, -3.5, -5.0, -5.0, 0.0, -5.0, -5.0, -5.0),
(-3.5, -2.5, -2.5, -3.5, -2.5, -0.5, -5.0, -5.0, -5.0, 0.0, -5.0, -5.0),
(-2.5, -3.5, -2.5, -2.5, -4.5, -4.5, -5.0, -5.0, -5.0, -5.0, 0.0, -5.0),
(-4.5, -4.5, -2.5, -4.5, -3.5, -3.5, -5.0, -5.0, -5.0, -5.0, -5.0, 0.0),
-- Neuron Stage 8
(-2.5, -3.5, -4.5, -0.5, -2.5, -3.5, 0.0, -5.0, -5.0, -5.0, -5.0, -5.0),
(-9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0),
(-9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0),
(-9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0),
(-9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0),
(-9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0));
wait;
end process;
end memory_arch;

-- A network of Neural Elements
use work.Neural_Package.all;
entity network is
    port (
        Network_Stimulus : in Stimulus_Matrix;
        Network_Weights : in Weights_Matrix;
        Network_Output : out Neural_Array);
end network;

architecture network_structure of network is
    component Neural_Node
        port (
            Stimulus : in Input_Array;
            Weights : in Input_Array;
            Output : out Real := 0.0);
    end component;

```

Figure 19. (continued).

```

-- Instantiation of all Neural_Nodes to the
-- Neural_Element design unit

for all : Neural_Node use entity work.Neural_element(behavior);
begin
  element_generate:
    for i in 1 to Total_Neurons generate
      Nodes : Neural_Node
        port map ( Network_Stimulus(i), Network_Weights(i), Network_Output(i) );
    end generate;
end network_structure;

-- This unit is used to sense the convergence of the network
-- to a solution within the tolerance limit

use work.Neural_Package.all;
entity conv_sensor is
  port (
    Old_Outputs : in Neural_Array;
    New_Outputs : in Neural_Array;
    Sensor_Out : out integer);
end conv_sensor;

architecture sensor_behavior of conv_sensor is
begin
  loop_process:
  process(Old_Outputs'Transaction)
    variable Sum : integer := 0;
    variable difference : real := 0.0;
  begin
    Sensor_Out <= 1 after 0 ns;
    Loop1: for I in 1 to Total_Neurons loop
      difference := abs (Old_Outputs(I) - New_Outputs(I));
      if difference > Tolerance then
        Sum := Sum + 1;
      end if;
    end loop Loop1;
    if Sum = 0 then
      Sensor_Out <= 0;
    end if;
  end process;
end sensor_behavior;

```

Figure 19. (continued).

Figure 19. (continued).

```

for all : network use entity work.network(network_structure);
for all : memory use entity work.memory(memory_arch);
for all : conv_sensor use entity work.conv_sensor(sensor_behavior);

begin
  memory_read : memory port map (Matrix_Weights);
  run_network : network port map (Matrix_Stimulus, Matrix_Weights, Outputs);
  sensor : conv_sensor port map (Old_Out, New_Out, Done);
  process
    variable tmp1 : integer := 1;
    variable tmp2 : integer := 1;
    variable tmp3 : integer := 1;
    variable tmp4 : integer := 0;
    begin
      wait on Stimuli"Transaction until Done = 1;

      -- Loop to do stimulus for each neuron
      All_Neurons_Loop:
        for C_Neurons in 1 to Total_Neurons loop

          -- determine number of stages for each neuron and set
          tmp3 := Neuron_Interconn/Neurons_per_Stage - 2;

          --loop for the number of stages
          Each_Stage:
            while tmp3 > -2 loop

              -- loop for each neuron in stage
              Update_Stimulus_Loop:
                for C_Input in 1 to Neurons_per_Stage loop

                  -- determine current stage for calculation
                  tmp1 := ((C_Neurons - 1)/Neurons_per_Stage) - tmp3;

                  -- if stage is negative set for proper wraparound
                  if tmp1 < 1 then
                    tmp1 := Network_Stages + tmp1;
                  end if;

                  -- set tmp2 for correct stimulus input from current neuron
                  tmp2 := ((tmp1 - 1) * Neurons_per_Stage) + C_Input;
                  Matrix_Stimulus(C_Neurons)(tmp4 + C_Input) <= Stimuli(tmp2) after 2 ns;
                end loop Update_Stimulus_Loop;

                  -- set for next stage of neurons
                  tmp3 := tmp3 - 1;
            end while Each_Stage;
        end for All_Neurons_Loop;
    end process;
end begin;

```

Figure 19. (continued).

```

-- set for next stage of inputs
    tmp4 := tmp4 + Neurons_per_Stage;
    end loop Each_Stage;
    tmp4 := 0;
    end loop All_Neurons_Loop;

-- send new values to convergence sensor
    New_to_Old_Loop: for i in 1 to Total_Neurons loop
        Old_Out(i) <= New_Out(i);
        New_Out(i) <= Stimuli(i);
    end loop New_to_Old_Loop;
end process;
end processor_structure;

-- This is the test bench for testing the whole system. Whole system is
-- integrated in ann_processor.

use work.Neural_package.all;
entity test_bench is
end test_bench;

architecture test_bench_arch of test_bench is

    component ann_processor
        port (
            Stimuli : in Neural_Array;
            Outputs : out Neural_Array);
    end component;

    signal kIn : Neural_Array :=
(0.0, 1.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 1.0, 1.0, 0.0,
1.0, 0.0, 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, 1.0, 1.0, 0.0,
0.0, 1.0, 0.0, 0.0, 0.0, 0.0,
1.0, 1.0, 0.0, 1.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0);
    signal kOut : Neural_Array;
    signal Stop : Boolean := false;

    for all : ann_processor use entity work.ann_processor(processor_structure);
    begin
        tester : ann_processor port map (kIn, kOut);
        process
        begin
            wait on kOut until Stop = false;
            kIn <= kOut after 4 ns;
            Stop <= True after 304 ns;
        end process;
    end test_bench_arch;

```

Figure 19. (continued).

APPENDIX C

C.1 Input Data Files

The input data files used for the simulations are shown in Figures 20 through 25. These files contain the weights used for the simulations. The weights can be interpreted as cost values from one neuron to the next neuron. The more negative a value is, the less likely that path will be chosen.

The values are arranged in arrays by neuron and by stage. There are six neurons in each stage, and there are six stages. The first array of values contains the weights for the first neuron in the first stage. The last six values in each neuron array are weights for neurons within the same stage as the neuron being considered. The values before these last six are from previous stages.

For example, with a three stage interconnected network, the first six values in an array are the weights from neurons two stages prior, the next six values are from neurons one stage prior, and the last six values are from neurons in the same stage. The 0.0 value within the last six values is a direct feedback from the neuron to itself, which is neither enabled or disabled because of the 0.0 value.

– Neuron Stage 1
 (0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0),
 (-9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0),
 (-9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0),
 (-9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0),
 (-9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0),
 (-9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0),
 – Neuron Stage 2
 (-2.5, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, -5.0, -5.0, -5.0, -5.0, -5.0),
 (-3.5, 0.0, 0.0, 0.0, 0.0, 0.0, -5.0, 0.0, -5.0, -5.0, -5.0, -5.0),
 (-0.5, 0.0, 0.0, 0.0, 0.0, 0.0, -5.0, -5.0, 0.0, -5.0, -5.0, -5.0),
 (-4.5, 0.0, 0.0, 0.0, 0.0, 0.0, -5.0, -5.0, -5.0, 0.0, -5.0, -5.0),
 (-2.5, 0.0, 0.0, 0.0, 0.0, 0.0, -5.0, -5.0, -5.0, -5.0, 0.0, -5.0),
 (-4.5, 0.0, 0.0, 0.0, 0.0, 0.0, -5.0, -5.0, -5.0, -5.0, -5.0, 0.0),
 – Neuron Stage 3
 (-2.5, -4.5, -4.5, -3.5, -2.5, -3.5, 0.0, -5.0, -5.0, -5.0, -5.0, -5.0),
 (-4.5, -2.5, -3.5, -4.5, -3.5, -2.5, -5.0, 0.0, -5.0, -5.0, -5.0, -5.0),
 (-2.5, -2.5, -4.5, -4.5, -3.5, -3.5, -5.0, -5.0, 0.0, -5.0, -5.0, -5.0),
 (-2.5, -2.5, -1.5, -4.5, -2.5, -3.5, -5.0, -5.0, -5.0, 0.0, -5.0, -5.0),
 (-4.5, -4.5, -2.5, -2.5, -2.5, -2.5, -5.0, -5.0, -5.0, -5.0, 0.0, -5.0),
 (-2.5, -2.5, -3.5, -4.5, -3.5, -2.5, -5.0, -5.0, -5.0, -5.0, -5.0, 0.0),
 – Neuron Stage 4
 (-2.5, -3.5, -4.5, -4.5, -3.5, -2.5, 0.0, -5.0, -5.0, -5.0, -5.0, -5.0),
 (-2.5, -3.5, -3.5, -3.5, -4.5, -2.5, -5.0, 0.0, -5.0, -5.0, -5.0, -5.0),
 (-2.5, -3.5, -4.5, -2.5, -2.5, -2.5, -5.0, -5.0, 0.0, -5.0, -5.0, -5.0),
 (-3.5, -3.5, -4.5, -2.5, -2.5, -2.5, -5.0, -5.0, -5.0, 0.0, -5.0, -5.0),
 (-4.5, -3.5, -2.5, -0.5, -2.5, -4.5, -5.0, -5.0, -5.0, -5.0, 0.0, -5.0),
 (-4.5, -3.5, -2.5, -2.5, -4.5, -4.5, -5.0, -5.0, -5.0, -5.0, -5.0, 0.0),
 – Neuron Stage 5
 (-2.5, -2.5, -2.5, -4.5, -3.5, -2.5, 0.0, -5.0, -5.0, -5.0, -5.0, -5.0),
 (-4.5, -4.5, -2.5, -4.5, -1.5, -4.5, -5.0, 0.0, -5.0, -5.0, -5.0, -5.0),
 (-4.5, -2.5, -2.5, -3.5, -3.5, -4.5, -5.0, -5.0, 0.0, -5.0, -5.0, -5.0),
 (-2.5, -4.5, -2.5, -3.5, -3.5, -4.5, -5.0, -5.0, -5.0, 0.0, -5.0, -5.0),
 (-4.5, -3.5, -2.5, -3.5, -3.5, -4.5, -5.0, -5.0, -5.0, -5.0, 0.0, -5.0),
 (-4.5, -3.5, -4.5, -4.5, -3.5, -3.5, -5.0, -5.0, -5.0, -5.0, -5.0, 0.0),
 – Neuron Stage 6
 (-4.5, -3.5, -2.5, -2.5, -4.5, -3.5, 0.0, -5.0, -5.0, -5.0, -5.0, -5.0),
 (-4.5, -2.5, -3.5, -4.5, -2.5, -2.5, -5.0, 0.0, -5.0, -5.0, -5.0, -5.0),
 (-4.5, -3.5, -3.5, -4.5, -2.5, -2.5, -5.0, -5.0, 0.0, -5.0, -5.0, -5.0),
 (-2.5, -4.5, -2.5, -2.5, -4.5, -4.5, -5.0, -5.0, -5.0, 0.0, -5.0, -5.0),
 (-3.5, -3.5, -4.5, -2.5, -3.5, -4.5, -5.0, -5.0, -5.0, -5.0, 0.0, -5.0),
 (-3.5, -0.5, -3.5, -2.5, -2.5, -2.5, -5.0, -5.0, -5.0, -5.0, -5.0, 0.0),
 – Neuron Stage 7
 (-2.5, -3.5, -4.5, -2.5, -3.5, -2.5, 0.0, -5.0, -5.0, -5.0, -5.0, -5.0),
 (-4.5, -2.5, -3.5, -2.5, -2.5, -2.5, -5.0, 0.0, -5.0, -5.0, -5.0, -5.0),
 (-4.5, -3.5, -2.5, -2.5, -3.5, -3.5, -5.0, -5.0, 0.0, -5.0, -5.0, -5.0),
 (-3.5, -2.5, -2.5, -3.5, -2.5, -0.5, -5.0, -5.0, -5.0, 0.0, -5.0, -5.0),
 (-2.5, -3.5, -2.5, -2.5, -4.5, -4.5, -5.0, -5.0, -5.0, -5.0, 0.0, -5.0),
 (-4.5, -4.5, -2.5, -4.5, -3.5, -3.5, -5.0, -5.0, -5.0, -5.0, -5.0, 0.0),
 – Neuron Stage 8
 (-2.5, -3.5, -4.5, -0.5, -2.5, -3.5, 0.0, -5.0, -5.0, -5.0, -5.0, -5.0),
 (-9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0),
 (-9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0),
 (-9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0),
 (-9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0),
 (-9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0),
 (-9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0);

Figure 20: Two Stage Interconnected Network Input Data.

[illegible]

[illegible]

(0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, -7.5, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, -5.0, -5.0, -5.0, -5.0, -5.0, -5.0),
(0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, -10.5, 0.0, 0.0, 0.0, 0.0, 0.0, -5.0, 0.0, -5.0, -5.0, -5.0, -5.0),
(0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, -4.5, 0.0, 0.0, 0.0, 0.0, 0.0, -5.0, -5.0, 0.0, -5.0, -5.0, -5.0),
(0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, -13.5, 0.0, 0.0, 0.0, 0.0, 0.0, -5.0, -5.0, -5.0, 0.0, -5.0, -5.0),
(0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, -7.5, 0.0, 0.0, 0.0, 0.0, 0.0, -5.0, -5.0, -5.0, -5.0, 0.0, -5.0),
(0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, -13.5, 0.0, 0.0, 0.0, 0.0, 0.0, -5.0, -5.0, -5.0, -5.0, -5.0, 0.0),

(0.0, 0.0, 0.0, 0.0, 0.0, 0.0, -2.5, -3.5, -2.5, -4.5, -2.5, -3.5, -2.5, -4.5, -4.5, -3.5, -2.5, -3.5, 0.0, -5.0, -5.0, -5.0, -5.0, -5.0),
(0.0, 0.0, 0.0, 0.0, 0.0, 0.0, -2.5, -4.5, -4.5, -3.5, -3.5, -4.5, -4.5, -2.5, -3.5, -4.5, -3.5, -2.5, -5.0, 0.0, -5.0, -5.0, -5.0, -5.0),
(0.0, 0.0, 0.0, 0.0, 0.0, 0.0, -4.5, -3.5, -4.5, -2.5, -2.5, -3.5, -2.5, -2.5, -4.5, -4.5, -3.5, -3.5, -5.0, -5.0, 0.0, -5.0, -5.0, -5.0),
(0.0, 0.0, 0.0, 0.0, 0.0, 0.0, -1.5, -2.5, -3.5, -4.5, -3.5, -4.5, -2.5, -2.5, -1.5, -4.5, -2.5, -3.5, -5.0, -5.0, -5.0, 0.0, -5.0, -5.0),
(0.0, 0.0, 0.0, 0.0, 0.0, 0.0, -4.5, -2.5, -3.5, -2.5, -2.5, -4.5, -4.5, -4.5, -2.5, -2.5, -2.5, -2.5, -5.0, -5.0, -5.0, -5.0, 0.0, -5.0),
(0.0, 0.0, 0.0, 0.0, 0.0, 0.0, -4.5, -4.5, -4.5, -3.5, -2.5, -2.5, -2.5, -2.5, -3.5, -4.5, -3.5, -2.5, -5.0, -5.0, -5.0, -5.0, -5.0, 0.0),

(-2.5, -3.5, -4.5, -4.5, -2.5, -3.5, -2.5, -3.5, -4.5, -4.5, -2.5, -3.5, -2.5, -3.5, -4.5, -4.5, -3.5, -2.5, 0.0, -5.0, -5.0, -5.0, -5.0, -5.0),
 (-3.5, -3.5, -4.5, -4.5, -2.5, -3.5, -4.5, -3.5, -2.5, -4.5, -4.5, -2.5, -2.5, -1.5, -3.5, -3.5, -4.5, -2.5, -5.0, 0.0, -5.0, -5.0, -5.0, -5.0),
 (-4.5, -2.4, -3.5, -3.5, -3.5, -4.5, -2.5, -2.5, -3.5, -4.5, -2.5, -4.5, -2.5, -3.5, -4.5, -2.5, -2.5, -2.5, -5.0, -5.0, 0.0, -5.0, -5.0, -5.0),
 (-4.5, -4.5, -4.5, -2.5, -3.5, -2.5, -3.5, -4.5, -2.5, -2.5, -4.5, -2.5, -3.5, -3.5, -4.5, -2.5, -2.5, -2.5, -5.0, -5.0, -5.0, 0.0, -5.0, -5.0),
 (-0.5, -2.5, -4.5, -2.5, -3.5, -4.5, -2.5, -4.5, -1.5, -2.5, -4.5, -3.5, -4.5, -3.5, -2.5, -0.5, -2.5, -4.5, -5.0, -5.0, -5.0, -5.0, 0.0, -5.0),
 (-2.5, -3.5, -3.5, -3.5, -4.5, -4.5, -2.5, -4.5, -3.5, -3.5, -2.5, -4.5, -4.5, -3.5, -2.5, -2.5, -4.5, -4.5, -5.0, -5.0, -5.0, -5.0, -5.0, 0.0),

57

– Neuron Stage 1
 (0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0),
 (-9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0),
 (-9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0),
 (-9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0),
 (-9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0),
 (-9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0),
 – Neuron Stage 2
 (-2.5, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, -5.0, -5.0, -5.0, -5.0, -5.0),
 (-3.5, 0.0, 0.0, 0.0, 0.0, 0.0, -5.0, 0.0, -5.0, -5.0, -5.0, -5.0),
 (-1.5, 0.0, 0.0, 0.0, 0.0, 0.0, -5.0, -5.0, 0.0, -5.0, -5.0, -5.0),
 (-4.5, 0.0, 0.0, 0.0, 0.0, 0.0, -5.0, -5.0, -5.0, 0.0, -5.0, -5.0),
 (-2.5, 0.0, 0.0, 0.0, 0.0, 0.0, -5.0, -5.0, -5.0, -5.0, 0.0, -5.0),
 (-4.5, 0.0, 0.0, 0.0, 0.0, 0.0, -5.0, -5.0, -5.0, -5.0, -5.0, 0.0),
 – Neuron Stage 3
 (-2.5, -4.5, -4.5, -3.5, -2.5, -3.5, 0.0, -5.0, -5.0, -5.0, -5.0, -5.0),
 (-4.5, -2.5, -3.5, -4.5, -3.5, -2.5, -5.0, 0.0, -5.0, -5.0, -5.0, -5.0),
 (-2.5, -2.5, -4.5, -4.5, -3.5, -3.5, -5.0, -5.0, 0.0, -5.0, -5.0, -5.0),
 (-2.5, -2.5, -1.5, -4.5, -2.5, -3.5, -5.0, -5.0, -5.0, 0.0, -5.0, -5.0),
 (-4.5, -4.5, -2.5, -2.5, -2.5, -2.5, -5.0, -5.0, -5.0, -5.0, 0.0, -5.0),
 (-2.5, -2.5, -3.5, -4.5, -3.5, -2.5, -5.0, -5.0, -5.0, -5.0, -5.0, 0.0),
 – Neuron Stage 4
 (-2.5, -3.5, -4.5, -4.5, -3.5, -2.5, 0.0, -5.0, -5.0, -5.0, -5.0, -5.0),
 (-2.5, -3.5, -3.5, -1.5, -4.5, -2.5, -5.0, 0.0, -5.0, -5.0, -5.0, -5.0),
 (-2.5, -3.5, -4.5, -2.5, -2.5, -2.5, -5.0, -5.0, 0.0, -5.0, -5.0, -5.0),
 (-3.5, -3.5, -4.5, -2.5, -2.5, -2.5, -5.0, -5.0, -5.0, 0.0, -5.0, -5.0),
 (-4.5, -3.5, -2.5, -0.5, -2.5, -4.5, -5.0, -5.0, -5.0, -5.0, 0.0, -5.0),
 (-4.5, -3.5, -2.5, -2.5, -4.5, -4.5, -5.0, -5.0, -5.0, -5.0, -5.0, 0.0),
 – Neuron Stage 5
 (-2.5, -2.5, -2.5, -4.5, -3.5, -2.5, 0.0, -5.0, -5.0, -5.0, -5.0, -5.0),
 (-4.5, -4.5, -2.5, -4.5, -1.5, -4.5, -5.0, 0.0, -5.0, -5.0, -5.0, -5.0),
 (-4.5, -2.5, -2.5, -3.5, -3.5, -4.5, -5.0, -5.0, 0.0, -5.0, -5.0, -5.0),
 (-2.5, -1.5, -2.5, -3.5, -3.5, -4.5, -5.0, -5.0, -5.0, 0.0, -5.0, -5.0),
 (-4.5, -3.5, -2.5, -3.5, -3.5, -4.5, -5.0, -5.0, -5.0, -5.0, 0.0, -5.0),
 (-4.5, -3.5, -4.5, -4.5, -3.5, -3.5, -5.0, -5.0, -5.0, -5.0, -5.0, 0.0),
 – Neuron Stage 6
 (-4.5, -3.5, -2.5, -2.5, -4.5, -3.5, 0.0, -5.0, -5.0, -5.0, -5.0, -5.0),
 (-4.5, -2.5, -3.5, -4.5, -2.5, -2.5, -5.0, 0.0, -5.0, -5.0, -5.0, -5.0),
 (-4.5, -3.5, -3.5, -4.5, -2.5, -2.5, -5.0, -5.0, 0.0, -5.0, -5.0, -5.0),
 (-2.5, -4.5, -2.5, -2.5, -4.5, -4.5, -5.0, -5.0, -5.0, 0.0, -5.0, -5.0),
 (-3.5, -3.5, -4.5, -2.5, -3.5, -4.5, -5.0, -5.0, -5.0, -5.0, 0.0, -5.0),
 (-3.5, -0.5, -3.5, -4.5, -2.5, -2.5, -5.0, -5.0, -5.0, -5.0, -5.0, 0.0),
 – Neuron Stage 7
 (-2.5, -3.5, -4.5, -2.5, -3.5, -2.5, 0.0, -5.0, -5.0, -5.0, -5.0, -5.0),
 (-4.5, -2.5, -3.5, -2.5, -2.5, -2.5, -5.0, 0.0, -5.0, -5.0, -5.0, -5.0),
 (-4.5, -3.5, -2.5, -2.5, -3.5, -3.5, -5.0, -5.0, 0.0, -5.0, -5.0, -5.0),
 (-3.5, -2.5, -2.5, -3.5, -2.5, -0.5, -5.0, -5.0, -5.0, 0.0, -5.0, -5.0),
 (-2.5, -3.5, -2.5, -2.5, -4.5, -4.5, -5.0, -5.0, -5.0, -5.0, 0.0, -5.0),
 (-4.5, -4.5, -2.5, -4.5, -3.5, -3.5, -5.0, -5.0, -5.0, -5.0, -5.0, 0.0),
 – Neuron Stage 8
 (-2.5, -3.5, -4.5, -0.5, -2.5, -3.5, 0.0, -5.0, -5.0, -5.0, -5.0, -5.0),
 (-9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0),
 (-9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0),
 (-9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0),
 (-9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0),
 (-9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0, -9.0);

Figure 23: Two Stage Interconnected Network with Sub-optimal Path Input Data.

[illegible]

Figure 24: Three Stage Interconnected Network with Sub-optimal Path Input Data.

(-4.5, -2.5, -2.5, -2.5, -4.5, -3.5, -2.5, -3.5, -3.5, -2.5, -4.5, -4.5, -2.5, -2.5, -2.5, -4.5, -3.5, -2.5, 0.0, -5.0, -5.0, -5.0, -5.0, -5.0),
 (-4.5, -4.5, -1.5, -3.5, -2.5, -4.5, -2.5, -3.5, -4.5, -1.5, -2.5, -4.5, -4.5, -4.5, -2.5, -4.5, -1.5, -4.5, -5.0, 0.0, -5.0, -5.0, -5.0, -5.0),
 (-4.5, -2.5, -2.5, -4.5, -2.5, -4.5, -2.5, -3.5, -3.5, -3.5, -4.5, -2.5, -4.5, -2.5, -2.5, -3.5, -3.5, -4.5, -5.0, -5.0, 0.0, -5.0, -5.0, -5.0),
 (-3.5, -2.5, -2.5, -2.5, -4.5, -3.5, -2.5, -1.5, -4.5, -4.5, -4.5, -2.5, -2.5, -1.5, -2.5, -3.5, -3.5, -4.5, -5.0, -5.0, -5.0, 0.0, -5.0, -5.0),
 (-3.5, -4.5, -4.5, -2.5, -3.5, -4.5, -2.5, -4.5, -2.5, -2.5, -2.5, -4.5, -4.5, -3.5, -2.5, -3.5, -3.5, -4.5, -5.0, -5.0, -5.0, -5.0, 0.0, -5.0),
 (-2.5, -2.5, -2.5, -3.5, -4.5, -3.5, -4.5, -4.5, -4.5, -3.5, -2.5, -3.5, -4.5, -3.5, -4.5, -4.5, -3.5, -3.5, -5.0, -5.0, -5.0, -5.0, -5.0, 0.0),

(-2.5, -4.5, -4.5, -4.5, -3.5, -3.5, -4.5, -4.5, -3.5, -2.5, -2.5, -3.5, -4.5, -3.5, -2.5, -2.5, -4.5, -3.5, 0.0, -5.0, -5.0, -5.0, -5.0, -5.0),
 (-3.5, -2.5, -2.5, -4.5, -2.5, -3.5, -2.5, -3.5, -3.5, -3.5, -4.5, -4.5, -4.5, -2.5, -3.5, -4.5, -2.5, -2.5, -5.0, 0.0, -5.0, -5.0, -5.0, -5.0),
 (-4.5, -2.5, -3.5, -3.5, -3.5, -4.5, -4.5, -2.5, -3.5, -2.5, -4.5, -2.5, -4.5, -4.5, -3.5, -3.5, -4.5, -2.5, -2.5, -5.0, -5.0, 0.0, -5.0, -5.0, -5.0),
 (-3.5, -2.5, -4.5, -4.5, -2.5, -2.5, -4.5, -4.5, -3.5, -4.5, -3.5, -2.5, -2.5, -4.5, -2.5, -2.5, -4.5, -4.5, -5.0, -5.0, -5.0, 0.0, -5.0, -5.0),
 (-4.5, -4.5, -3.5, -2.5, -2.5, -2.5, -4.5, -4.5, -2.5, -2.5, -3.5, -3.5, -3.5, -3.5, -4.5, -2.5, -3.5, -4.5, -5.0, -5.0, -5.0, -5.0, 0.0, -5.0),
 (-4.5, -2.5, -2.5, -1.5, -4.5, -3.5, -2.5, -1.5, -2.5, -3.5, -1.5, -4.5, -3.5, -0.5, -3.5, -4.5, -2.5, -2.5, -5.0, -5.0, -5.0, -5.0, -5.0, 0.0),

(-2.5, -3.5, -4.5, -4.5, -3.5, -4.5, -3.5, -3.5, -4.5, -2.5, -4.5, -2.5, -2.5, -3.5, -4.5, -2.5, -3.5, -2.5, 0.0, -5.0, -5.0, -5.0, -5.0, -5.0),
 (-4.5, -4.5, -4.5, -2.5, -3.5, -4.5, -4.5, -3.5, -2.5, -4.5, -2.5, -3.5, -4.5, -2.5, -3.5, -2.5, -2.5, -2.5, -5.0, 0.0, -5.0, -5.0, -5.0, -5.0),
 (-4.5, -4.5, -2.5, -2.5, -3.5, -2.5, -4.5, -4.5, -4.5, -2.5, -3.5, -4.5, -4.5, -3.5, -2.5, -2.5, -3.5, -3.5, -5.0, -5.0, 0.0, -5.0, -5.0, -5.0),
 (-2.5, -2.5, -3.5, -2.5, -0.5, -3.5, -3.5, -1.5, -4.5, -2.5, -4.5, -3.5, -3.5, -2.5, -2.5, -3.5, -2.5, -0.5, -5.0, -5.0, -5.0, 0.0, -5.0, -5.0),
 (-4.5, -2.5, -2.5, -2.5, -3.5, -2.5, -4.5, -4.5, -2.5, -4.5, -3.5, -3.5, -2.5, -3.5, -2.5, -2.5, -4.5, -4.5, -5.0, -5.0, -5.0, -5.0, 0.0, -5.0),
 (-4.5, -4.5, -3.5, -2.5, -2.5, -3.5, -4.5, -2.5, -2.5, -2.5, -3.5, -2.5, -4.5, -4.5, -2.5, -4.5, -3.5, -3.5, -5.0, -5.0, -5.0, -5.0, -5.0, 0.0),

[illegible]

62

APPENDIX D

D.1 Simulation Outputs

Figures 26 through 44 show the results for all of the simulations performed. The results show the final outputs of the neural coprocessor and the time of completion. The time of completion minus 4 nanoseconds and then divided by 10 nanoseconds equals the number of cycles needed to arrive at a solution.

TIME	SIGNAL NAMES					
(NS)	KOUT(1)	KOUT(2)	KOUT(3)	KOUT(4)	KOUT(5)	KOUT(6)
0	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
4	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00
14	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
24	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
34	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
44	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
54	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
64	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
74	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
84	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
94	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
104	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
114	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
124	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
134	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
144	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
154	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
164	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
174	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
184	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
194	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
204	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
214	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00

Figure 27: Results of Low Gain Ramp Neuron Model.

TIME	SIGNAL NAMES					
(NS)	KOUT(25)	KOUT(26)	KOUT(27)	KOUT(28)	KOUT(29)	KOUT(30)
0	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
4	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00
14	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
24	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00
34	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
44	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00
54	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
64	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00
74	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
84	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00
94	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
104	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00
114	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
124	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00
134	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
144	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00
154	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
164	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00
174	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
184	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00
194	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
204	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00
214	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00

Figure 27. (continued).

TIME	SIGNAL NAMES					
(NS)	KOUT(25)	KOUT(26)	KOUT(27)	KOUT(28)	KOUT(29)	KOUT(30)
0	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
4	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00
14	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
24	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00
34	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
44	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00
54	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
64	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00
74	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
84	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00
94	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
104	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00
114	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
124	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00
134	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
144	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00
154	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
164	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00
174	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
184	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00
194	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
204	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00
214	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00

Figure 29. (continued).

TIME	SIGNAL NAMES					
(NS)	KOUT(37)	KOUT(38)	KOUT(39)	KOUT(40)	KOUT(41)	KOUT(42)
0	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
4	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00
14	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
24	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00
34	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
44	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00
54	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
64	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00
74	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
84	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00
94	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
104	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00
114	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
124	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00
134	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
144	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00
154	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
164	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00
174	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
184	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00
194	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
204	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00
214	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00

Figure 29. (continued).

TIME	SIGNAL NAMES					
(NS)	KOUT(13)	KOUT(14)	KOUT(15)	KOUT(16)	KOUT(17)	KOUT(18)
0	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
4	7.500000E-01	7.500000E-01	7.500000E-01	7.500000E-01	7.500000E-01	7.500000E-01
14	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
24	5.000000E-01	5.000000E-01	5.000000E-01	5.000000E-01	5.000000E-01	5.000000E-01
34	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
44	2.500000E-01	2.500000E-01	2.500000E-01	5.000000E-01	5.000000E-01	2.500000E-01
54	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
64	0.000000E+00	2.500000E-01	0.000000E+00	5.000000E-01	2.500000E-01	2.500000E-01
74	0.000000E+00	0.000000E+00	0.000000E+00	2.500000E-01	0.000000E+00	0.000000E+00
84	0.000000E+00	0.000000E+00	0.000000E+00	5.000000E-01	2.500000E-01	0.000000E+00
94	0.000000E+00	0.000000E+00	0.000000E+00	5.000000E-01	0.000000E+00	0.000000E+00
104	0.000000E+00	0.000000E+00	0.000000E+00	5.000000E-01	0.000000E+00	0.000000E+00
114	0.000000E+00	0.000000E+00	0.000000E+00	7.500000E-01	0.000000E+00	0.000000E+00
124	0.000000E+00	0.000000E+00	0.000000E+00	7.500000E-01	0.000000E+00	0.000000E+00
134	0.000000E+00	0.000000E+00	0.000000E+00	7.500000E-01	0.000000E+00	0.000000E+00
144	0.000000E+00	0.000000E+00	0.000000E+00	7.500000E-01	0.000000E+00	0.000000E+00
154	0.000000E+00	0.000000E+00	0.000000E+00	7.500000E-01	0.000000E+00	0.000000E+00
164	0.000000E+00	0.000000E+00	0.000000E+00	7.500000E-01	0.000000E+00	0.000000E+00
174	0.000000E+00	0.000000E+00	0.000000E+00	7.500000E-01	0.000000E+00	0.000000E+00
184	0.000000E+00	0.000000E+00	0.000000E+00	7.500000E-01	0.000000E+00	0.000000E+00
194	0.000000E+00	0.000000E+00	0.000000E+00	7.500000E-01	0.000000E+00	0.000000E+00
204	0.000000E+00	0.000000E+00	0.000000E+00	7.500000E-01	0.000000E+00	0.000000E+00
214	0.000000E+00	0.000000E+00	0.000000E+00	7.500000E-01	0.000000E+00	0.000000E+00

Figure 32. (continued).

TIME	SIGNAL NAMES					
(NS)	KOUT(1)	KOUT(2)	KOUT(3)	KOUT(4)	KOUT(5)	KOUT(6)
0	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
4	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00
14	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
24	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
34	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
44	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
54	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
64	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
74	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
84	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
94	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
104	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
114	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
124	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
134	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
144	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
154	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
164	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00

TIME	SIGNAL NAMES					
(NS)	KOUT(7)	KOUT(8)	KOUT(9)	KOUT(10)	KOUT(11)	KOUT(12)
0	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
4	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00
14	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
24	0.000000E+00	0.000000E+00	5.000000E-01	0.000000E+00	0.000000E+00	0.000000E+00
34	0.000000E+00	0.000000E+00	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
44	0.000000E+00	0.000000E+00	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
54	0.000000E+00	0.000000E+00	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
64	0.000000E+00	0.000000E+00	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
74	0.000000E+00	0.000000E+00	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
84	0.000000E+00	0.000000E+00	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
94	0.000000E+00	0.000000E+00	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
104	0.000000E+00	0.000000E+00	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
114	0.000000E+00	0.000000E+00	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
124	0.000000E+00	0.000000E+00	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
134	0.000000E+00	0.000000E+00	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
144	0.000000E+00	0.000000E+00	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
154	0.000000E+00	0.000000E+00	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
164	0.000000E+00	0.000000E+00	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00

Figure 33: Results of High Gain Ramp Response Neuron With Capacitance.

144	0.000000E+00	0.000000E+00	8.500000E-01	0.000000E+00	0.000000E+00	0.000000E+00
154	0.000000E+00	0.000000E+00	8.500000E-01	0.000000E+00	0.000000E+00	0.000000E+00
164	0.000000E+00	0.000000E+00	8.500000E-01	0.000000E+00	0.000000E+00	0.000000E+00
174	0.000000E+00	0.000000E+00	8.500000E-01	0.000000E+00	0.000000E+00	0.000000E+00
184	0.000000E+00	0.000000E+00	8.500000E-01	0.000000E+00	0.000000E+00	0.000000E+00
194	0.000000E+00	0.000000E+00	8.500000E-01	0.000000E+00	0.000000E+00	0.000000E+00
204	0.000000E+00	0.000000E+00	8.500000E-01	0.000000E+00	0.000000E+00	0.000000E+00
214	0.000000E+00	0.000000E+00	8.500000E-01	0.000000E+00	0.000000E+00	0.000000E+00
224	0.000000E+00	0.000000E+00	8.500000E-01	0.000000E+00	0.000000E+00	0.000000E+00
234	0.000000E+00	0.000000E+00	8.500000E-01	0.000000E+00	0.000000E+00	0.000000E+00
244	0.000000E+00	0.000000E+00	8.500000E-01	0.000000E+00	0.000000E+00	0.000000E+00
254	0.000000E+00	0.000000E+00	8.500000E-01	0.000000E+00	0.000000E+00	0.000000E+00
264	0.000000E+00	0.000000E+00	8.500000E-01	0.000000E+00	0.000000E+00	0.000000E+00

Figure 34. (continued).

[illegible]

Figure 34. (continued).

144	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	5.000000E-01	0.000000E+00
154	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	8.500000E-01	0.000000E+00
164	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	8.500000E-01	0.000000E+00
174	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	8.500000E-01	0.000000E+00
184	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	8.500000E-01	0.000000E+00
194	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	8.500000E-01	0.000000E+00
204	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	8.500000E-01	0.000000E+00
214	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	8.500000E-01	0.000000E+00
224	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	8.500000E-01	0.000000E+00
234	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	8.500000E-01	0.000000E+00
244	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	8.500000E-01	0.000000E+00
254	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	8.500000E-01	0.000000E+00
264	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	8.500000E-01	0.000000E+00

Figure 34. (continued).

144	5.000000E-01	5.000000E-01	5.000000E-01	5.000000E-01	5.000000E-01	5.000000E-01
154	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
164	5.000000E-01	5.000000E-01	5.000000E-01	5.000000E-01	5.000000E-01	5.000000E-01
174	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
184	5.000000E-01	5.000000E-01	5.000000E-01	5.000000E-01	5.000000E-01	5.000000E-01
194	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
204	5.000000E-01	5.000000E-01	1.500000E-01	5.000000E-01	1.500000E-01	1.500000E-01
214	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
224	1.500000E-01	1.500000E-01	1.500000E-01	5.000000E-01	0.000000E+00	1.500000E-01
234	0.000000E+00	0.000000E+00	0.000000E+00	5.000000E-01	0.000000E+00	0.000000E+00
244	0.000000E+00	0.000000E+00	0.000000E+00	5.000000E-01	0.000000E+00	0.000000E+00
254	0.000000E+00	0.000000E+00	0.000000E+00	8.500000E-01	0.000000E+00	0.000000E+00
264	0.000000E+00	0.000000E+00	0.000000E+00	8.500000E-01	0.000000E+00	0.000000E+00

Figure 34. (continued).

TIME	SIGNAL NAMES					
(NS)	KOUT(1)	KOUT(2)	KOUT(3)	KOUT(4)	KOUT(5)	KOUT(6)
0	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
4	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00
14	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
24	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
34	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
44	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
54	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
64	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
74	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
84	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
94	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
104	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
114	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
124	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
134	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
144	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
154	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
164	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00

TIME	SIGNAL NAMES					
(NS)	KOUT(7)	KOUT(8)	KOUT(9)	KOUT(10)	KOUT(11)	KOUT(12)
0	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
4	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00
14	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
24	0.000000E+00	0.000000E+00	5.000000E-01	0.000000E+00	0.000000E+00	0.000000E+00
34	0.000000E+00	0.000000E+00	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
44	0.000000E+00	0.000000E+00	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
54	0.000000E+00	0.000000E+00	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
64	0.000000E+00	0.000000E+00	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
74	0.000000E+00	0.000000E+00	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
84	0.000000E+00	0.000000E+00	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
94	0.000000E+00	0.000000E+00	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
104	0.000000E+00	0.000000E+00	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
114	0.000000E+00	0.000000E+00	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
124	0.000000E+00	0.000000E+00	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
134	0.000000E+00	0.000000E+00	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
144	0.000000E+00	0.000000E+00	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
154	0.000000E+00	0.000000E+00	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
164	0.000000E+00	0.000000E+00	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00

Figure 35: Results of High Gain Sigmoid Response Neuron With Capacitance.

TIME	SIGNAL NAMES					
(NS)	KOUT(1)	KOUT(2)	KOUT(3)	KOUT(4)	KOUT(5)	KOUT(6)
0	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
4	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00
14	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
24	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
34	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
44	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
54	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
64	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
74	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
84	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
94	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
104	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
114	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
124	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
134	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00

TIME	SIGNAL NAMES					
(NS)	KOUT(7)	KOUT(8)	KOUT(9)	KOUT(10)	KOUT(11)	KOUT(12)
0	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
4	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00
14	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
24	0.000000E+00	0.000000E+00	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
34	0.000000E+00	0.000000E+00	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
44	0.000000E+00	0.000000E+00	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
54	0.000000E+00	0.000000E+00	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
64	0.000000E+00	0.000000E+00	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
74	0.000000E+00	0.000000E+00	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
84	0.000000E+00	0.000000E+00	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
94	0.000000E+00	0.000000E+00	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
104	0.000000E+00	0.000000E+00	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
114	0.000000E+00	0.000000E+00	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
124	0.000000E+00	0.000000E+00	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
134	0.000000E+00	0.000000E+00	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00

Figure 36: Results of Three Stage Interconnection Pattern and Binary Neuron.

TIME	SIGNAL NAMES					
(NS)	KOUT(13)	KOUT(14)	KOUT(15)	KOUT(16)	KOUT(17)	KOUT(18)
0	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
4	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00
14	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
24	7.500000E-01	7.500000E-01	2.500000E-01	1.000000E+00	2.500000E-01	2.500000E-01
34	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
44	7.500000E-01	7.500000E-01	2.500000E-01	1.000000E+00	2.500000E-01	2.500000E-01
54	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
64	7.500000E-01	7.500000E-01	2.500000E-01	1.000000E+00	2.500000E-01	2.500000E-01
74	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
84	7.500000E-01	7.500000E-01	2.500000E-01	1.000000E+00	2.500000E-01	2.500000E-01
94	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
104	7.500000E-01	7.500000E-01	2.500000E-01	1.000000E+00	2.500000E-01	2.500000E-01
114	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
124	7.500000E-01	7.500000E-01	2.500000E-01	1.000000E+00	2.500000E-01	2.500000E-01
134	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
144	7.500000E-01	7.500000E-01	2.500000E-01	1.000000E+00	2.500000E-01	2.500000E-01
154	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
164	7.500000E-01	7.500000E-01	2.500000E-01	1.000000E+00	2.500000E-01	2.500000E-01
174	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
184	7.500000E-01	7.500000E-01	2.500000E-01	1.000000E+00	2.500000E-01	2.500000E-01
194	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
204	7.500000E-01	7.500000E-01	2.500000E-01	1.000000E+00	2.500000E-01	2.500000E-01
214	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00

Figure 37. (continued).

TIME	SIGNAL NAMES					
(NS)	KOUT(25)	KOUT(26)	KOUT(27)	KOUT(28)	KOUT(29)	KOUT(30)
0	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
4	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00
14	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
24	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00
34	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
44	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00
54	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
64	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00
74	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
84	8.500000E-01	1.000000E+00	8.500000E-01	5.000000E-01	8.500000E-01	8.500000E-01
94	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
104	8.500000E-01	1.000000E+00	5.000000E-01	1.500000E-01	8.500000E-01	5.000000E-01
114	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
124	1.500000E-01	8.500000E-01	0.000000E+00	0.000000E+00	1.500000E-01	0.000000E+00
134	0.000000E+00	1.500000E-01	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
144	0.000000E+00	8.500000E-01	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
154	0.000000E+00	8.500000E-01	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
164	0.000000E+00	8.500000E-01	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
174	0.000000E+00	8.500000E-01	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
184	0.000000E+00	8.500000E-01	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
194	0.000000E+00	8.500000E-01	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00

Figure 38. (continued).

TIME	SIGNAL NAMES					
(NS)	KOUT(1)	KOUT(2)	KOUT(3)	KOUT(4)	KOUT(5)	KOUT(6)
0	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
4	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00
14	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
24	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
34	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
44	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
54	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
64	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
74	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
84	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
94	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00

TIME	SIGNAL NAMES					
(NS)	KOUT(7)	KOUT(8)	KOUT(9)	KOUT(10)	KOUT(11)	KOUT(12)
0	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
4	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00
14	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
24	0.000000E+00	0.000000E+00	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
34	0.000000E+00	0.000000E+00	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
44	0.000000E+00	0.000000E+00	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
54	0.000000E+00	0.000000E+00	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
64	0.000000E+00	0.000000E+00	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
74	0.000000E+00	0.000000E+00	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
84	0.000000E+00	0.000000E+00	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
94	0.000000E+00	0.000000E+00	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00

TIME	SIGNAL NAMES					
(NS)	KOUT(13)	KOUT(14)	KOUT(15)	KOUT(16)	KOUT(17)	KOUT(18)
0	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
4	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00
14	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
24	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00
34	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
44	0.000000E+00	0.000000E+00	0.000000E+00	1.000000E+00	0.000000E+00	0.000000E+00
54	0.000000E+00	0.000000E+00	0.000000E+00	1.000000E+00	0.000000E+00	0.000000E+00
64	0.000000E+00	0.000000E+00	0.000000E+00	1.000000E+00	0.000000E+00	0.000000E+00
74	0.000000E+00	0.000000E+00	0.000000E+00	1.000000E+00	0.000000E+00	0.000000E+00
84	0.000000E+00	0.000000E+00	0.000000E+00	1.000000E+00	0.000000E+00	0.000000E+00
94	0.000000E+00	0.000000E+00	0.000000E+00	1.000000E+00	0.000000E+00	0.000000E+00

Figure 39: Results of Four Stage Interconnection Pattern with Binary Neuron.

TIME	SIGNAL NAMES					
(NS)	KOUT(37)	KOUT(38)	KOUT(39)	KOUT(40)	KOUT(41)	KOUT(42)
0	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
4	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00
14	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
24	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00
34	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
44	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00
54	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
64	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00
74	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
84	0.000000E+00	0.000000E+00	0.000000E+00	1.000000E+00	0.000000E+00	0.000000E+00
94	0.000000E+00	0.000000E+00	0.000000E+00	1.000000E+00	0.000000E+00	0.000000E+00

TIME	SIGNAL NAMES					
(NS)	KOUT(43)	KOUT(44)	KOUT(45)	KOUT(46)	KOUT(47)	KOUT(48)
0	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
4	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00
14	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
24	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00
34	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
44	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00
54	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
64	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00
74	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
84	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
94	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00

Figure 39. (continued).

TIME	SIGNAL NAMES					
(NS)	KOUT(13)	KOUT(14)	KOUT(15)	KOUT(16)	KOUT(17)	KOUT(18)
0	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
4	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00
14	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
24	1.000000E+00	1.000000E+00	7.500000E-01	1.000000E+00	7.500000E-01	7.500000E-01
34	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
44	2.500000E-01	5.000000E-01	0.000000E+00	1.000000E+00	2.500000E-01	0.000000E+00
54	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
64	2.500000E-01	5.000000E-01	0.000000E+00	1.000000E+00	2.500000E-01	0.000000E+00
74	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
84	2.500000E-01	5.000000E-01	0.000000E+00	1.000000E+00	2.500000E-01	0.000000E+00
94	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
104	2.500000E-01	5.000000E-01	0.000000E+00	1.000000E+00	2.500000E-01	0.000000E+00
114	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
124	2.500000E-01	5.000000E-01	0.000000E+00	1.000000E+00	2.500000E-01	0.000000E+00
134	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
144	2.500000E-01	5.000000E-01	0.000000E+00	1.000000E+00	2.500000E-01	0.000000E+00
154	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
164	2.500000E-01	5.000000E-01	0.000000E+00	1.000000E+00	2.500000E-01	0.000000E+00
174	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
184	2.500000E-01	5.000000E-01	0.000000E+00	1.000000E+00	2.500000E-01	0.000000E+00
194	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
204	2.500000E-01	5.000000E-01	0.000000E+00	1.000000E+00	2.500000E-01	0.000000E+00
214	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00

Figure 40. (continued).

TIME	SIGNAL NAMES					
(NS)	KOUT(1)	KOUT(2)	KOUT(3)	KOUT(4)	KOUT(5)	KOUT(6)
0	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
4	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00
14	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
24	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
34	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
44	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
54	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
64	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
74	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
84	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
94	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
104	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
114	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
124	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
134	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00

TIME	SIGNAL NAMES					
(NS)	KOUT(7)	KOUT(8)	KOUT(9)	KOUT(10)	KOUT(11)	KOUT(12)
0	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
4	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00
14	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
24	0.000000E+00	0.000000E+00	8.500000E-01	0.000000E+00	0.000000E+00	0.000000E+00
34	0.000000E+00	0.000000E+00	8.500000E-01	0.000000E+00	0.000000E+00	0.000000E+00
44	0.000000E+00	0.000000E+00	8.500000E-01	0.000000E+00	0.000000E+00	0.000000E+00
54	0.000000E+00	0.000000E+00	8.500000E-01	0.000000E+00	0.000000E+00	0.000000E+00
64	0.000000E+00	0.000000E+00	8.500000E-01	0.000000E+00	0.000000E+00	0.000000E+00
74	0.000000E+00	0.000000E+00	8.500000E-01	0.000000E+00	0.000000E+00	0.000000E+00
84	0.000000E+00	0.000000E+00	8.500000E-01	0.000000E+00	0.000000E+00	0.000000E+00
94	0.000000E+00	0.000000E+00	8.500000E-01	0.000000E+00	0.000000E+00	0.000000E+00
104	0.000000E+00	0.000000E+00	8.500000E-01	0.000000E+00	0.000000E+00	0.000000E+00
114	0.000000E+00	0.000000E+00	8.500000E-01	0.000000E+00	0.000000E+00	0.000000E+00
124	0.000000E+00	0.000000E+00	8.500000E-01	0.000000E+00	0.000000E+00	0.000000E+00
134	0.000000E+00	0.000000E+00	8.500000E-01	0.000000E+00	0.000000E+00	0.000000E+00

Figure 41: Results of Four Stage Interconnection Pattern with Sigmoid Neuron.

TIME	SIGNAL NAMES					
(NS)	KOUT(13)	KOUT(14)	KOUT(15)	KOUT(16)	KOUT(17)	KOUT(18)
0	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
4	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00
14	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
24	1.000000E+00	1.000000E+00	8.500000E-01	1.000000E+00	8.500000E-01	8.500000E-01
34	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
44	1.500000E-01	5.000000E-01	0.000000E+00	1.000000E+00	0.000000E+00	0.000000E+00
54	0.000000E+00	0.000000E+00	0.000000E+00	1.500000E-01	0.000000E+00	0.000000E+00
64	0.000000E+00	1.500000E-01	0.000000E+00	1.000000E+00	0.000000E+00	0.000000E+00
74	0.000000E+00	0.000000E+00	0.000000E+00	1.000000E+00	0.000000E+00	0.000000E+00
84	0.000000E+00	0.000000E+00	0.000000E+00	1.000000E+00	0.000000E+00	0.000000E+00
94	0.000000E+00	0.000000E+00	0.000000E+00	1.000000E+00	0.000000E+00	0.000000E+00
104	0.000000E+00	0.000000E+00	0.000000E+00	1.000000E+00	0.000000E+00	0.000000E+00
114	0.000000E+00	0.000000E+00	0.000000E+00	1.000000E+00	0.000000E+00	0.000000E+00
124	0.000000E+00	0.000000E+00	0.000000E+00	1.000000E+00	0.000000E+00	0.000000E+00
134	0.000000E+00	0.000000E+00	0.000000E+00	1.000000E+00	0.000000E+00	0.000000E+00

TIME	SIGNAL NAMES					
(NS)	KOUT(19)	KOUT(20)	KOUT(21)	KOUT(22)	KOUT(23)	KOUT(24)
0	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
4	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00
14	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
24	1.000000E+00	1.000000E+00	8.500000E-01	8.500000E-01	1.000000E+00	1.000000E+00
34	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
44	1.500000E-01	1.500000E-01	0.000000E+00	0.000000E+00	1.000000E+00	5.000000E-01
54	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
64	0.000000E+00	1.500000E-01	0.000000E+00	0.000000E+00	1.000000E+00	1.500000E-01
74	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	8.500000E-01	0.000000E+00
84	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	1.000000E+00	0.000000E+00
94	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	1.000000E+00	0.000000E+00
104	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	1.000000E+00	0.000000E+00
114	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	1.000000E+00	0.000000E+00
124	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	1.000000E+00	0.000000E+00
134	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	1.000000E+00	0.000000E+00

Figure 41. (continued).

TIME	SIGNAL NAMES					
(NS)	KOUT(1)	KOUT(2)	KOUT(3)	KOUT(4)	KOUT(5)	KOUT(6)
0	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
4	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00
14	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
24	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
34	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
44	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
54	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
64	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
74	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
84	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
94	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
104	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
114	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
124	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
134	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00

TIME	SIGNAL NAMES					
(NS)	KOUT(7)	KOUT(8)	KOUT(9)	KOUT(10)	KOUT(11)	KOUT(12)
0	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
4	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00
14	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
24	0.000000E+00	0.000000E+00	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
34	0.000000E+00	0.000000E+00	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
44	0.000000E+00	0.000000E+00	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
54	0.000000E+00	0.000000E+00	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
64	0.000000E+00	0.000000E+00	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
74	0.000000E+00	0.000000E+00	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
84	0.000000E+00	0.000000E+00	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
94	0.000000E+00	0.000000E+00	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
104	0.000000E+00	0.000000E+00	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
114	0.000000E+00	0.000000E+00	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
124	0.000000E+00	0.000000E+00	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
134	0.000000E+00	0.000000E+00	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00

Figure 43: Results of Three Stage Interconnection Pattern and Sub-optimal Path.

TIME	SIGNAL NAMES					
(NS)	KOUT(1)	KOUT(2)	KOUT(3)	KOUT(4)	KOUT(5)	KOUT(6)
0	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
4	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00
14	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
24	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
34	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
44	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
54	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
64	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
74	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
84	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
94	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00

TIME	SIGNAL NAMES					
(NS)	KOUT(7)	KOUT(8)	KOUT(9)	KOUT(10)	KOUT(11)	KOUT(12)
0	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
4	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00
14	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
24	0.000000E+00	0.000000E+00	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
34	0.000000E+00	0.000000E+00	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
44	0.000000E+00	0.000000E+00	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
54	0.000000E+00	0.000000E+00	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
64	0.000000E+00	0.000000E+00	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
74	0.000000E+00	0.000000E+00	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
84	0.000000E+00	0.000000E+00	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
94	0.000000E+00	0.000000E+00	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00

TIME	SIGNAL NAMES					
(NS)	KOUT(13)	KOUT(14)	KOUT(15)	KOUT(16)	KOUT(17)	KOUT(18)
0	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
4	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00
14	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
24	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00
34	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
44	0.000000E+00	0.000000E+00	0.000000E+00	1.000000E+00	0.000000E+00	0.000000E+00
54	0.000000E+00	0.000000E+00	0.000000E+00	1.000000E+00	0.000000E+00	0.000000E+00
64	0.000000E+00	0.000000E+00	0.000000E+00	1.000000E+00	0.000000E+00	0.000000E+00
74	0.000000E+00	0.000000E+00	0.000000E+00	1.000000E+00	0.000000E+00	0.000000E+00
84	0.000000E+00	0.000000E+00	0.000000E+00	1.000000E+00	0.000000E+00	0.000000E+00
94	0.000000E+00	0.000000E+00	0.000000E+00	1.000000E+00	0.000000E+00	0.000000E+00

Figure 44: Results of Four Stage Interconnection Pattern and Sub-optimal Path.

TIME	SIGNAL NAMES					
(NS)	KOUT(37)	KOUT(38)	KOUT(39)	KOUT(40)	KOUT(41)	KOUT(42)
0	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
4	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00
14	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
24	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00
34	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
44	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00
54	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
64	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00
74	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
84	0.000000E+00	0.000000E+00	0.000000E+00	1.000000E+00	0.000000E+00	0.000000E+00
94	0.000000E+00	0.000000E+00	0.000000E+00	1.000000E+00	0.000000E+00	0.000000E+00

TIME	SIGNAL NAMES					
(NS)	KOUT(43)	KOUT(44)	KOUT(45)	KOUT(46)	KOUT(47)	KOUT(48)
0	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
4	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00
14	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
24	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00
34	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
44	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00
54	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
64	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00	1.000000E+00
74	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
84	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
94	1.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00

Figure 44. (continued).

References

References

- [1] R. Jurgen, "The Specialties", *IEEE Spectrum*, January 1991, p. 79.
- [2] R. Lippmann, "An Introduction To Computing With Neural Nets", *IEEE ASSP Magazine*, April 1987, pp. 4-22.
- [3] F. Salam, "A Tutorial Workshop On Neural Nets And Their Engineering Implementations", *31st Midwest Symposium On Circuits And Systems*.
- [4] M. Kennedy and L. Chua, "Neural Networks For Nonlinear Programming", *IEEE Transactions on Circuits and Systems*, Vol. 35, No. 5, pp. 554-562.
- [5] M. Hanes, S. Ahalt, K. Mirza, and D. Orin, "A Neural Network Interface to the DIGITS Grasping System", *ICJNN Conference on Neural Networks*, 1990, Vol. III, pp. 343-348.
- [6] S. Wang and H. Yeh, "Self-Adaptive Neural Architectures for Control Applications", *ICJNN Conference on Neural Networks*, 1990, Vol. III, pp. 309-313.
- [7] W. McCulloch and W. Pitts, "A Logical Calculus of the Ideas Imminent in Nervous Activity", *Bulletin of Mathematical Biophysics*, Vol. 5, pp. 115-133, 1943
- [8] J. Vidal, J. Pemberton, and J. Goodwin, "Implementing Neural Nets with Programmable Logic", *IEEE 1st Conference on Neural Networks*, 1987, Vol. III, pp. 539-545.
- [9] J. Hopfield, "Neural Networks and Physical Systems with Emergent Collective Computational Abilities", *Proceedings of National Academy of Science*, Vol. 79, pp. 2554-2558.
- [10] B. Shriver, "Artificial Neural Systems", *Computer*, March 1988, pp. 8-9.

- [11] C. Chiu, C. Maa, and M. Shanblatt, "An Artificial Neural Network Algorithm for Dynamic Programming", *International Journal of Neural Systems*, Vol. 1, No. 3, pp. 211-220.
- [12] J. Hopfield, "Neurons with Graded Response have Collective Computational Properties like those of Two-State Neurons", *Proceedings of National Academy of Sciences*, Vol. 81, pp. 3088-3092.
- [13] P. Bozovsky, "Discrete Hopfield Model with Graded Response", *ICJNN Conference on Neural Networks*, Vol. III, pp. 851-856.
- [14] J. Hopfield and D. Tank, "'Neural' Computation of Decisions in Optimization Problems", *Biological Cybernetics*, Vol. 52, pp. 141-152.
- [15] C. Keshavachandra and M. Shanblatt, "Modeling Artificial Neural Networks Using VHDL", Technical Report, MSU-ENGR-90-009, Michigan State University Technical Report.
- [16] R. Lipsett, C. Schaefer, and C. Ussery, VHDL: Hardware Description and Design, Kluwer Academic Publishers, 1989.
- [17] D. Tank and J. Hopfield, "Simple 'Neural' Optimization Networks: An A/D Converter, Signal Decision Circuit, and a Linear Programming Circuit", *IEEE Transactions on Circuits and Systems*, Vol. CAS-33, No. 5, pp. 533-541.
- [18] User's Manual for the Standard VHDL 1076 Support Environment, Intermetrics, 1988.
- [19] Linear Circuits Data Book, Texas Instruments, 1984, pp. 3-224 and 3-227.
- [20] The TTL Data Book, Volume 2, Texas Instruments, 1985, pp.3-4 and 3-5.

MICHIGAN STATE UNIV. LIBRARIES



31293007845997