ELECTRONIC REALIZATION OF HUMAN BRAIN'S NEO-CORTEX COLUMN USING FPGA

A thesis (or dissertation) submitted to the faculty of
San Francisco State University
In partial fulfillment of
The Requirements for
The Degree

Master of Science

In

Engineering
Embedded Electrical and Computer Systems

by

Padmavalli Vadali

San Francisco, California

December 2010

CERTIFICATION OF APPROVAL

I certify that I have read Electronic Realization of Human Brain's Neo-Cortex Column Using FPGA by Padmavalli Vadali, and that in my opinion this work meets the criteria for approving a thesis submitted in partial fulfillment of the requirements for the degree: Master of Science in Engineering: Embedded Electrical and Computer Systems at San Francisco State University.

Hamid Mahmoodi
Professor of Engineering

V.V. Krishnan
Professor of Engineering

Christopher Moffatt
Professor of Biology

ELECTRONIC REALIZATION OF HUMAN BRAIN'S NEO-CORTEX COLUMN USING FPGA

Padmavalli Vadali
San Francisco, California
2010

The objective of this thesis is a low-power, hardware implementation of columns of neurons in the human brain's neo-cortex. The biological neo-cortex of the human brain consists of innumerable number of columns. Each column is made of six layers with millions of neurons in each layer. This implementation consists of ten columns with thirty-six neurons in each, i.e. six neurons in each layer. This accounts to 300 neurons with 60 pulsars.

This thesis also proves the low power implementation of a layer of neurons as compared to the biological replica. Since it is a hardware implementation, the time taken to process a neural function is accelerated. Results from the Blue Brain Project [1] show that using a software simulation approach for whole brain simulation demands extremely high performance computing machines with significantly high resulting power consumption. We intend to show the benefits of hardware emulation of the brain in realizing a much faster and lower power method to simulate the brain's neural function.

The model has been coded in VERILOG providing the simulation results, starting with a single neuron to ten columns of neurons. The power calculations are also estimated. The results of a single neuron are also verified with the results of Neo-Cortical Simulator (NCS), an open source software by University of Nevada.

I certify that the Abstract is a correct representation of the content of this thesis.

Chair, Thesis Committee　　　　　　　　　　　　　　Date

Table of Contents

LIST OF TABLES

LIST OF FIGURES

LIST OF APPENDICES

INTRODUCTION

Considerable progress made in neuro-morphic engineering is not yet developed enough to get close to VLSI mimicry of the brain power efficiency. The brain consisting of 10^{10} neurons with 10^{14} neural connections is a very power efficient system that is still the most complex system to date [16]. Comparison of hardware/software implementation and software simulations shows how faraway humans are in achieving the same efficiency as biological neurons. The power consumed by the software implementations are up to 500 billion times more and neo-cortical simulator result is half of the biological data. An obvious reason for such error in the simulation is lack of physical interconnections that shall account for significant power consumption. Yet, the intelligence of the brain is not achieved. Hence, mapping the brain is crucial.

The human brain has trillions of synapses, billions of neurons, millions of proteins, and thousands of genes [16]. Each of these neurons is an elaborate electrochemical circuit with its own power management and information processing structure. The synapses, which are the junction of neurons, allow them to form circuits with the central nervous system. Neurons communicate by utilizing chemicals and electrical synapses activated by exploitation of the electrically excitable neuron membrane. The transmission and reception of information takes place among neurons, which is activated with the help of action potential process. Given the extensive interconnection and information processing that happens inside the brain, we are still able to supply its power within our daily food budget. Theories of evolution indicate that the brain had evolved to an efficient system that would maximize the daily nutrition budget.

Various organizations had taken interest in the brain's small volume and low power system. Prominent institutions such as the Defense Advanced Research Projects Agency (DARPA), Stanford University and IBM Corporation are conducting projects that attempt to generate an artificial brain with the help of reverse engineering. Most of these projects are software simulations as opposed to hardware implementation. The purpose of this thesis is the hardware, low power implementation of:

1. Neural functions of a single neuron, verification with Neo-Cortical Simulator [14].
2. Low power implementations.
3. Different layers of neurons and their interconnections
4. Columns and their interconnections.

The model proposed in this thesis is implemented in Verilog using Synopsys VCS simulator. The verification of a single neuron is done against NCS (Open source software) outputs by University of Nevada.

BACKGROUND

An average brain has a weight of 1.5 kg and surface area of 2,500 square centimeters [1]. With a density of 146,000 neurons per millimeter the cortex surface alone would have 36.5 billion neurons [2]. Taking into account the remaining tissue, the brain is estimated have 100 billion neurons. Each neuron is connected to thousand other neurons to communicate and process information. The information processing speed of the brain is slow compared to current electronic implementations. However, since each of the

Number of neurons (adult)	20,000,000,000 - 100,000,000,000
Number of neurons in cerebral cortex (adult)	about 20,000,000,000
Number of synapses (adult)	10^{14} (2,000-5,000 per neuron)
Weight	New Born = 0.3 kg, Adult = 1.4 kg
Power consumption (adult)	20 ~ 40 Watts
Percentage of body	2% weight 20-44% power consumption
Genetic code influence	1 bit per 10,000-1,000,000 synapses

neurons has extensive interconnections; the overall speed of the brain can even challenge the fastest supercomputers around. It is estimated that there are about 1 million synaptic events per second or 1×10^6. All the facts about brain are listed in the table I below.

TABLE I. Brain Facts

THE BRAIN POWER

The brain is the most metabolically active organ in the body, consuming 20% of its energy consumption while accounting for only 2% of its weight [3]. Furthermore, comparative studies suggest that the brain design, function and evolution were influenced by its energy requirements [4]. Power is the rate at which energy is converted, so in understanding the amount of power the brain consumes, we need to understand its source of energy and its daily consumption. The main source of energy for the brain is the glucose, found in most dietary carbohydrates. The brain utilizes this energy through the use of its cellular power plant, mitochondrion which is inside the cellular body of the

$$\left[\begin{aligned} 2400\ kcal\ calorie &= \frac{2400\ kcal}{24\ hr} \\ &= \frac{100\ kcal}{1\ hr}\, x \left(\frac{1hr}{3600sec}\right) x \left(\frac{4.184 joules}{1\ cal}\right) \\ &= \frac{116.38\ joules}{sec} = 116\ watts \end{aligned}\right]$$

neuron. Mitochondria usually get their cell’s power from glucose oxidation. Taking into account the 2400kcal intake of an average man based on the international dietary standard calorie allowance as computed using the Atwater system, the power consumption per day of an average man can be computed as follows.

CELLULAR BIOLOGY

BRAIN REGIONS

The brain is a soft mass composed of neural tissues and nerve cells linked to the spinal cord. It fits a surface area of about 2500 square centimeters into the skull by forming folds and grooves. Anatomist identifies four major lobes on the surface of each hemisphere of the brain divided by grooves or sulci. Several large sulci divide four lobes of different functions: frontal lobe, parietal lobe, occipital lobe and temporal lobe (Fig. 1).

The *frontal lobe*, located in the front of each cerebral hemisphere is involved with cognitive, speech and motor functions. Specifically, its responsibilities include reasoning, problem solving, judgment, and impulse control. Located behind it is t*he parietal lobe* which is concerned with sensory perceptions. It integrates sensory information from parts of the body and associates auditory and visual signals with the memory to give them meaning. The *temporal lobe* on the other hand, is concerned with auditory sensation, processing of speech and vision semantics, and formation of long term memory. The smallest of four lobes and the located in the rearmost of the skull is the occipital lobe. It is concerned with decoding of visual information.

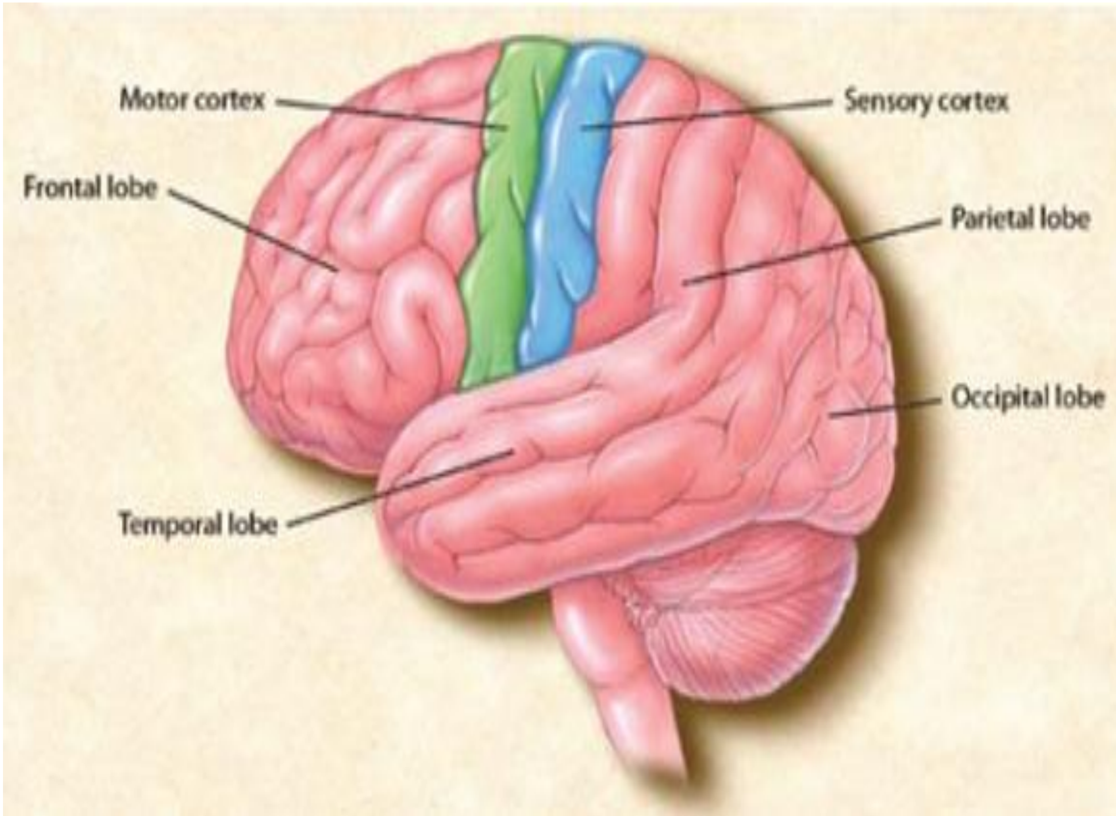


Fig. 1: Parts of the Brain [5]

BASICS OF HARDWARE IMPLEMENTATION

The main circuits mostly involved in hardware implementation of artificial brain models are Multiplication circuit, RC delay, Sigmoid Transfer function, Charge Storage [11]. We reviewed some technical IEEE papers dealing with the similar idea and came to this conclusion about the functional circuitry of brain.

PROJECTS AND SIMULATORS

PROJECTS

Over the years, many projects were developed targeting the computational speeds, number of neurons, and number of synapses. Many super computers have been used for this purpose. Various projects developed are Blue brain project, Facets, Neurogrid, Neuro-Morphic, and Synapse. A detailed discussion of some projects is as follows:

BLUE BRAIN PROJECT

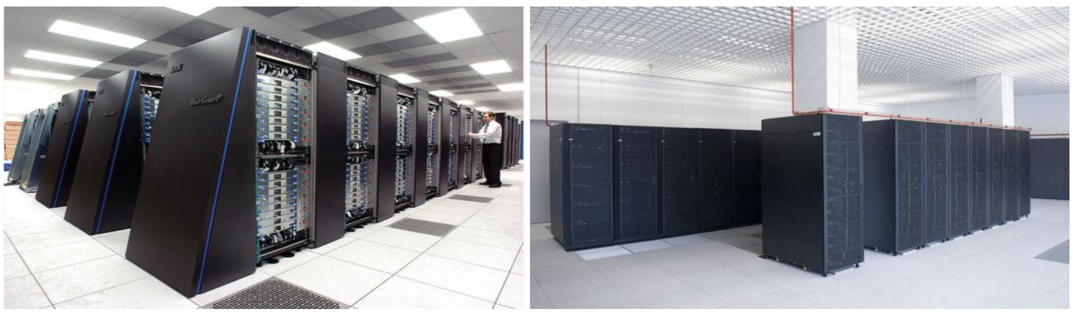


Fig. 2: IBM Blue Gene/P Super Computer [12]

The blue brain project is built by IBM, using IBM's Blue Gene /P super computer with 1,47,456 processors and 144TB of main memory. It can simulate one billion neurons and 10 trillion synapses. This project is the result of reverse engineering the mammalian brain. Firstly it replicated a rat's brain, and then was developed to simulate cat's brain. Recently, IBM developed it to replicate human brain. The supercomputer which was used to replicate the rat's brain was Magerit supercomputer. The software they have used for the project is 'Neuron'. In figure 7 are the images of the two supercomputers they used.

The simulator consumes one mega watt power to simulate one million neurons.

FACETS (FAST ANALOG COMPUTING WITH EMERGENT TRANSIENT STATES)

The FACETS project is based in Europe using a very large scale integration based neural network ASIC (Application Specific Integrated Circuit). It uses both analog and digital architectures for replicating a rat's brain. The analog architecture emulates performance of the rat's brain and digital architecture is used for communications between neurons. A single chip used in the project emulates 348 neurons and 100,000 synapses. Eight chips emulating 200.000 neurons and fifty million synapses together are under development. It consumes 1 killo-watt per single wafer, and consumes 2.8 Megawatts for emulating 1 million neurons.

NEUROGRID

NeuroGrid is developed by the researchers at Stanford University. This is a hardware implementation of one million neurons and six billion synapses. It consists of 16 neurocores, each neurocore has 65536 silicon neurons. Total number of neurons in 16 neurocores is one million. Similar to the FACETS project, the analog circuits simulate the ion channels of the brain and digital circuits are for communications. The power consumed for simulating one million neurons is about 1MW. Below is a picture showing the neurogrid project.

Fig. 3: Neurogrid [13]

SIMULATORS

Many software implementations use simulators to replicate the neural networks. There are many such simulators available in the industry. Some of them are: Neo – cortical simulators, Cat brain simulator, C2 Simulator, Genesis, and Neuron etc. Among these various simulators, we have used the Neo-Cortical simulator for calculating the power consumption.

NEO-CORTICAL SIMULATOR

The neo-cortical simulator is a batch processing spiking neural network software. This was developed by Philip Goodman at University of Nevada, Reno in the year 1998. It is a open source software, simulating more than ten thousand (10,000) neurons and hundred thousand (100,000) synapses.

The brain input file consists of four major parts:

1. Brain
2. Columns
3. Layer
4. Cell/Neuron

The simulator simulates for one neuron and the output is shown as below

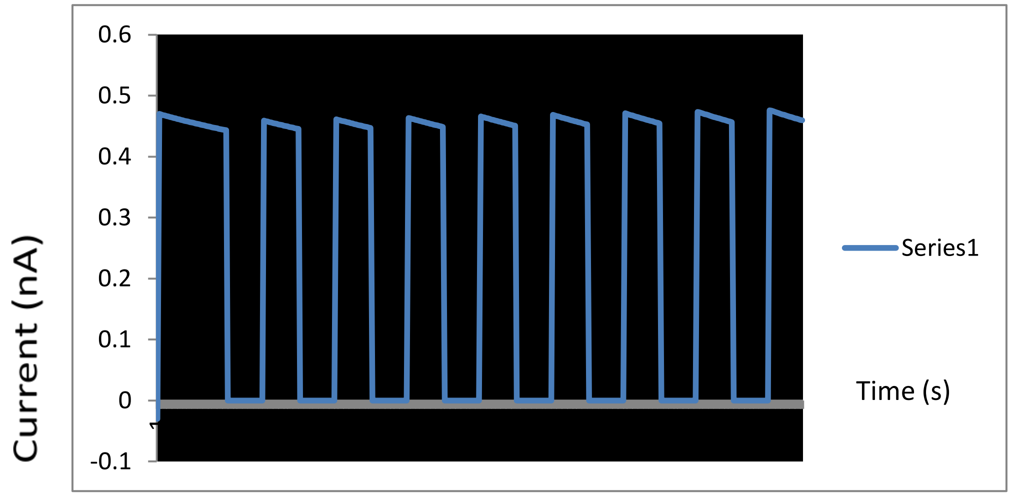


Fig. 4: NCS Current vs. Time Graph

TABLE II. Comparison of power consumption of existing projects

Projects	Power consumption	Per 100 billion neuron
Biological brain	20 – 40 W / 100 billion neurons	20-40W
Blue brain project	100 MW/1 million neurons	10000 GW
FACET	1KW / 348 neurons or 2.8MW / 1 million neurons	280GW
Neurogrid	1MW / 1 million neurons	1GW
Neo – cortical simulator	100 pW / neuron	10W

From the above plots:

The power consumption is 100pW for one neuron.

Comparing the above results:

From the above table, we can conclude that the biological brain consumes a lot less power as compared to hardware and software implementations. The hardware implementations consume less power than the software implementations. The power consumed by the neo-cortical simulator is much less than both hardware and software implementations. This is because the simulator is just simulating a single neuron. The power consumptions calculated for other projects involve many processors and supercomputers. The power consumptions of the implementations are so high. Yet, the intelligence of the brain is not achieved. Hence, mapping the brain is crucial.

CHAPTER 2

NEO-CORTEX

Neo-cortex basically consists of grey matter with unmyelinated fibers which surrounds the deeper white matter in the cerebrum. It consists of six layers of neurons starting from the first layer on surface of the cortex to the sixth layer. Most of the neurons in the neo-cortex are of two types: pyramidal neurons (80%) and inter neurons (20%). All the layers are connected to each other systematically, with inner most layers i.e., vi layer connecting outwards, ii and iii layers connecting inwards, and iv layer has connections sideways. The vertical arrangement of neurons is called neo-cortical columns. These columns are 0.5 mm in diameter and 2 mm deeper.

THE NEURON

The neuron is the basic functional block of the nervous system. It is a highly specialized cellular unit for information processing and transmission of electrochemical signals. With diameters ranging from 4 to 100 microns, each neuron contains millions of electrochemical pumping stations that shift about 200 sodium ions and 130 potassium ions per second [2]. The neuron fires electrochemical signals along its axon and interact with dendrites of another neuron at the point of virtual contact called as synapse. Parts and types of neurons are explained in the following sections to understand this concept better.

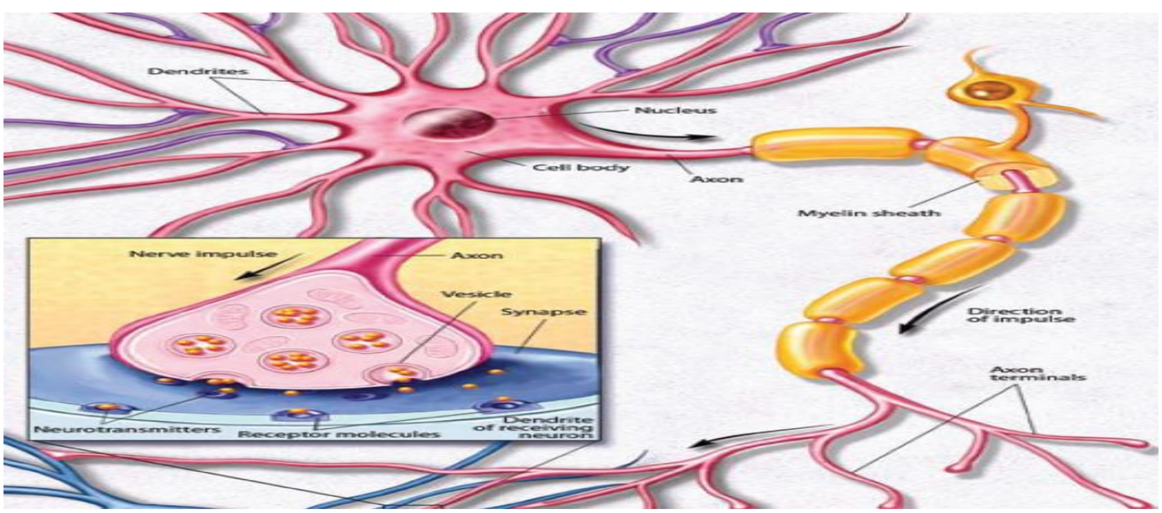


Fig. 5: Parts of a neuron [5]

PARTS OF THE NEURON

Neurons are specialized anatomically according to functions. However, most neurons are composed of four major parts: soma, dendrite, axon and synapse. The main part of the neuron that contains the nucleus is the *soma*. Also called the cell body, the soma contains all other necessary components of the cell such as ribosome and mitochondria which are in charge of building proteins and producing energy respectively. The transmitter and the receiver part of the neuron resemble a bush or a tree. The part of the neuron that functions as the receiver is called *dendrite*. A neuron contains multiple dendrites which are extended outward to communicate with thousand other neurons. Dendrites bring electrochemical stimuli to the cell body. The part of the neuron that carries away electrochemical output from the neuron towards other target cell is the *axon*. A neuron generally has only one axon which could be very long depending on the neuron function. Axons terminate in branches that interact with dendrites of another neuron. Axons and dendrites interact in specialized junctions called the *synapses*. These junctions may be electrical or chemical.

TYPES OF NEURON ACCORDING TO STRUCTURE

Anatomical structure of neurons is important for fulfilling its functions. Neurons are classified as anaxonic, bipolar, unipolar, or multipolar according to structural relationship of the main parts of the neuron. *Anaxonic* neurons have a structure that makes distinguishing dendrites from axon difficult. There are no anatomical clues to differentiating them. These types of neurons are the most common types of sensory neurons. *Bipolar* neurons on the other hand have a single dendrite and an axon with the cell body in between. They are small, measuring less than 30mm in length. Bipolar neurons are rare, but are often involved in special senses where they relay information to other neurons. In a *unipolar* neuron the dendrite and axon are continuous in one side with the cell body on the other side. Sensory neurons of peripheral nervous systems are often unipolar with axons extending to up to a meter or more. The most common type of the neuron is the *multipolar* neuron which has multiple dendrites and a single axon. Multipolar neurons can also be as long as unipolar neurons because they control skeletal muscles.

TYPES OF NEURON ACCORDING TO FUNCTION

Neurons have specialized functions which utilizes its structure and built-in electrochemical signals. Neurons can be categorized according to function as sensory neuron, motor neuron or interneuron. The first type is the *sensory neurons* which carry the information inward i.e. from the external or internal environment. Sensory neurons have specialized receptor that translates different stimuli from the internal or external environment. First they convert the stimuli to electrical signals and then into chemical signals that are passed along to other neurons. Almost all sensory neurons are unipolar neurons. The next type is the *motor neurons* which carries outward the stimuli from the brain. Motor neurons are multipolar neurons that transmit signals from the central nervous system to effectors such as muscles or glands. The type of neuron that provides connection between the sensory and motor neurons is the interneuron. Located entirely in the brain and the spinal cord, interneurons outnumber motor neurons and sensory neurons combined. Inter neurons handle the information processing by analyzing sensory inputs and coordinating motor outputs.

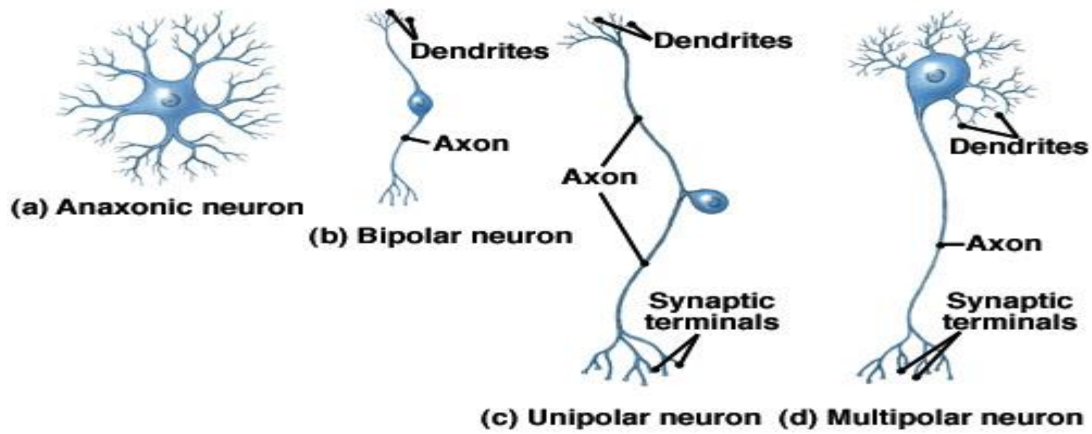


Fig. 6: Types of Neuron According to Anatomy [6]

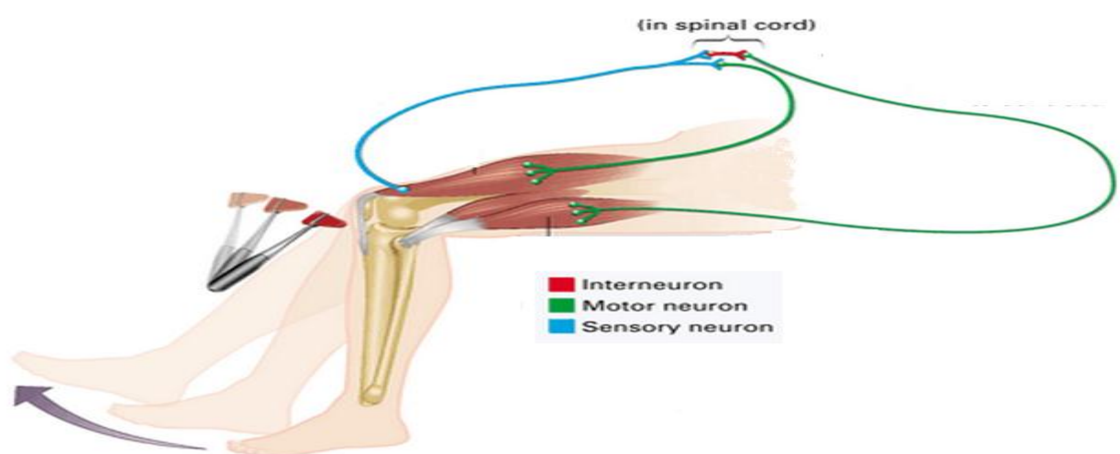


Fig 7: Types of Neuron According to Function [7]

The working of brain is easy to understand if we understand the functions it does. So the most important function that we need to understand is Action Potential which occurs in neurons.

CONCEPT OF ACTION POTENTIAL

Action Potential is the ionic activity. It involves mainly two ions- Sodium (Na^+) and Potassium (K^+). A cell (or a neuron) in an unexcited state is called in its resting state and the potential is called resting potential. The magnitude of resting potential is usually around -70mV. The resting state is also called polarized state of a neuron. During resting potential the Na^+ ions will be inside the membrane of the cell and the K^+ ions will be outside. When the cell is stimulated it affects the polarity of the cell and hence starts the depolarization. The Na^+ ions will be pumped out and K^+ ions will be allowed to pass inside the membrane. The exchange ratio of these ions is 3:2 respectively. A sufficient threshold level is to be reached in order to start the depolarization. Once it reaches the threshold level, the cell is in excited state and the maximum value of the potential at this point will be +40 mV. Figure1 shows the working of Action Potential inside a neuron [8].

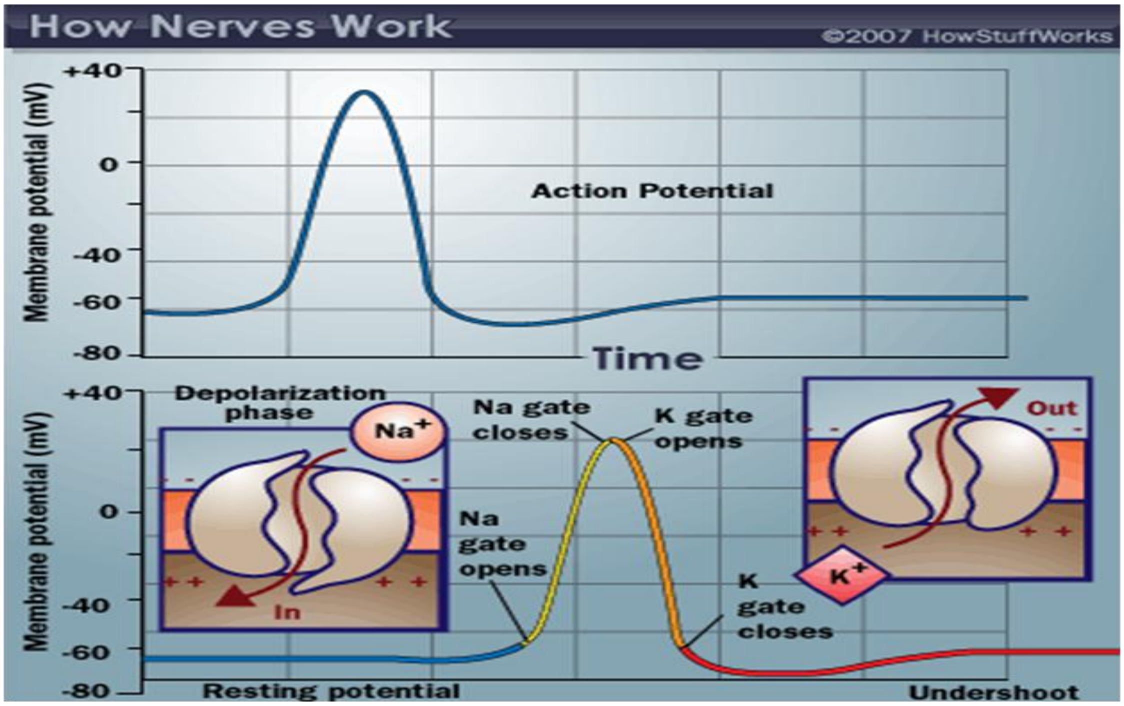


Fig. 8: Action Potential [9]

The unequal pumping activity of ion exchange will pass on to the neighboring cells and this is how the information will be transferred within and among the cells. The termination of this process will take place when the firing is not sufficient to excite the neighboring neurons.

REFRACTORY PERIOD

The firing frequency is the rate at which a cell can be stimulated again. It is an important measure to find out how fast any information can transmit through neurons. The maximum firing frequency of a neuron is 250 – 2000 Hz (0.5 – 4 ms), [10].

Another basic function that is followed in neurons is addition of synapses. Synapse is term given to the gap between the axon of one neuron to the dendrite of another. There are two different synapses namely, excitatory synapse (EPSP-Excitatory Post Synaptic Potential) and inhibitory synapse (IPSP- Inhibitory Post Synaptic Potential; also called Hyper polarization) [9]. The difference between the former and the latter is that the excitatory synapse will lead to firing of the neurons further which the latter will not.

OTHER ELECTRICAL PARAMETERS:

From the literature study we find that:

1. Power Consumption: The magnitude of the power consumption of an adult human brain is 20 – 40 Watts. The average power consumption per neuron is 0.5 – 4 nWatts [10].
2. Speed of Transmission inside axon: It is 90m/s in sheathed neuron and <0.1m/s in unsheathed neuron [10].

THE LAYERS AND COLUMNS OF NEO-CORTEX:

A layer is a network of number of neurons. There are about six horizontal layers of neurons in the neo-cortex. Each layer has its own composition in terms of connectivity and number of neurons. The vertical arrangement of neurons is called neo-cortical column. Below is the figure that describes the different layers:

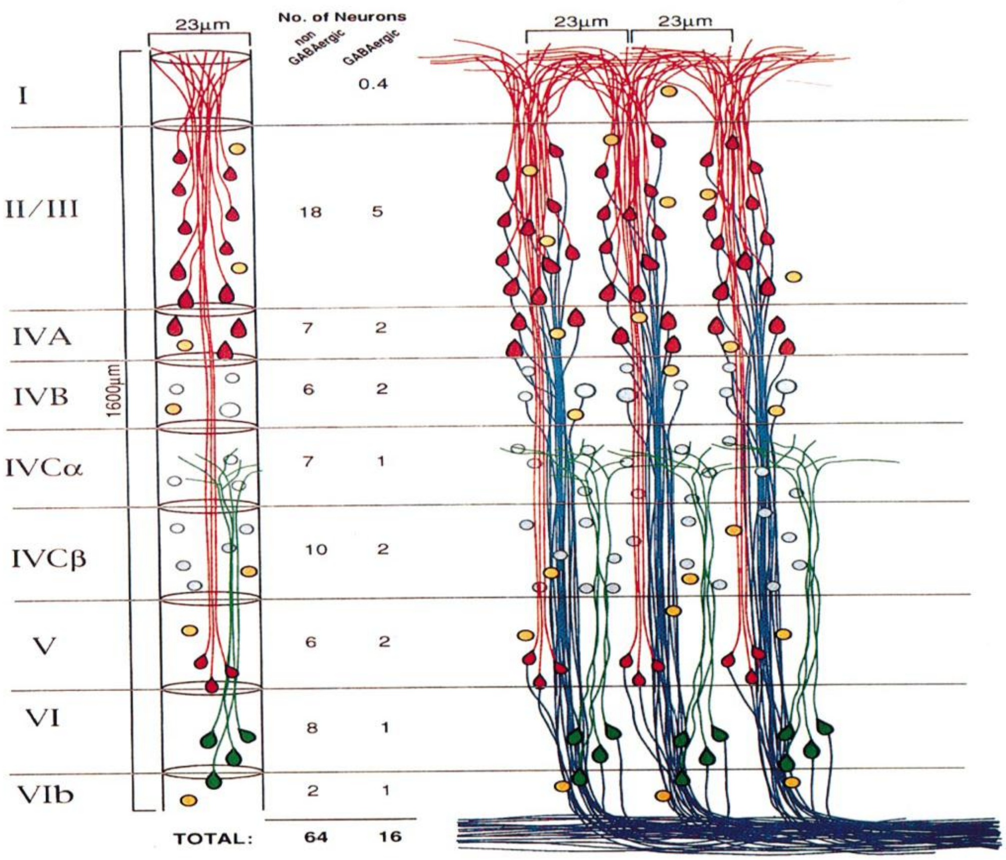


Fig. 9: Layers Interconnections [15]

CHAPTER 3

MODELS FOR ELECTRONIC REALIZATION OF NEURON

BASIC COMPONENTS

We know that the neurons consist of dendrites which carry information, cell nucleus which accumulates information and axons which send information in the form of electric pulses. We have considered Mcculloch's Pitts model also known as linear threshold gate model as a reference. This model was the earliest model ever proposed for the function of neuron. It is a neuron of a set of inputs $I_1,I_2.....I_n$ and output y. Then the output can be represented by the following equation:

$$X = \sum I_i W_i$$

Where $W_1,W_2......W_n$ are weight values normalized in the range of either (0,1) or (-1,1). The following figure represents the Mccculloch pitts model

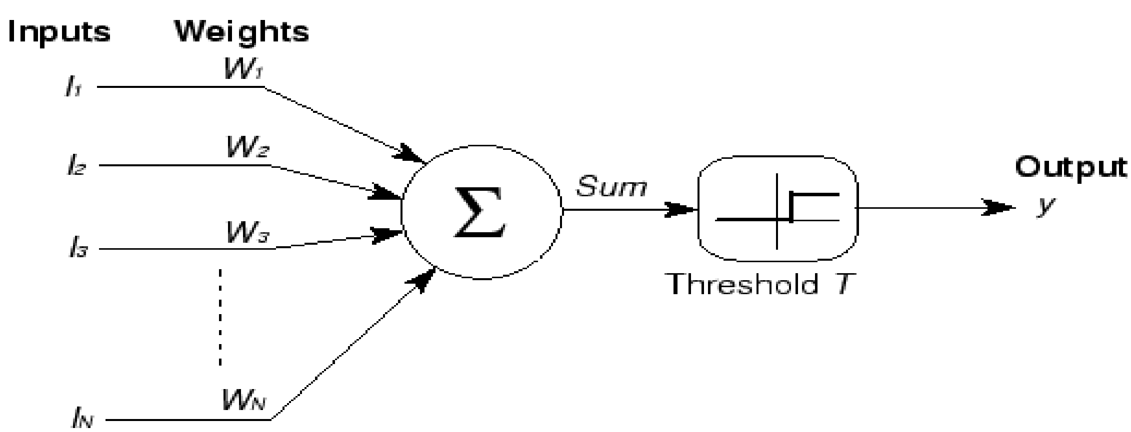


Fig. 10: McCulloch Pitts Model

The activation function is performed which gives the output of the neuron. The signals generated by actual biological neurons are the action-potential spikes, and the biological neurons are sending the signal in patterns of spikes rather than simple absence or presence of single spike pulse. For example, the signal could be a continuous stream of pulses with various frequencies. With this kind of observation, we should consider a signal to be continuous with bounded range.

$$y = \frac{1}{1 + exp(-x)}.$$

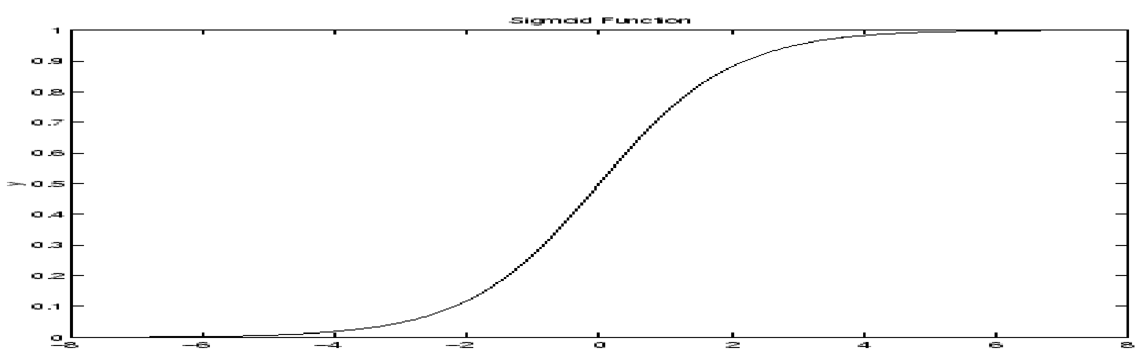


Fig. 11: Sigmoid Function

Additionally, the sigmoid function describes the ``closeness" to the threshold point by the slope. As x approaches to $-\infty$ or ∞ , the slope is zero; the slope increases as x approaches to 0. This characteristic often plays an important role in learning of neural networks.

MODELING OF INPUT NEURON

Input neuron is the neuron which detects external signals and passes it on to inter neurons. These inputs can be of any type ranging from pulse, square, and sine. In this thesis, I have considered two pulse inputs which are counted and transmitted, after a certain delay. From the McCulloch Pitts model, there are two thresholds involved: synaptic gap threshold, and activation function threshold. Synaptic gap threshold is modeled as weight to the AND gate, and activation function threshold is modeled as comparator. Our model for input neuron is as shown below. This model has been verified using Synopsys VCS verification. Output for the model is provided in the appendix 1.

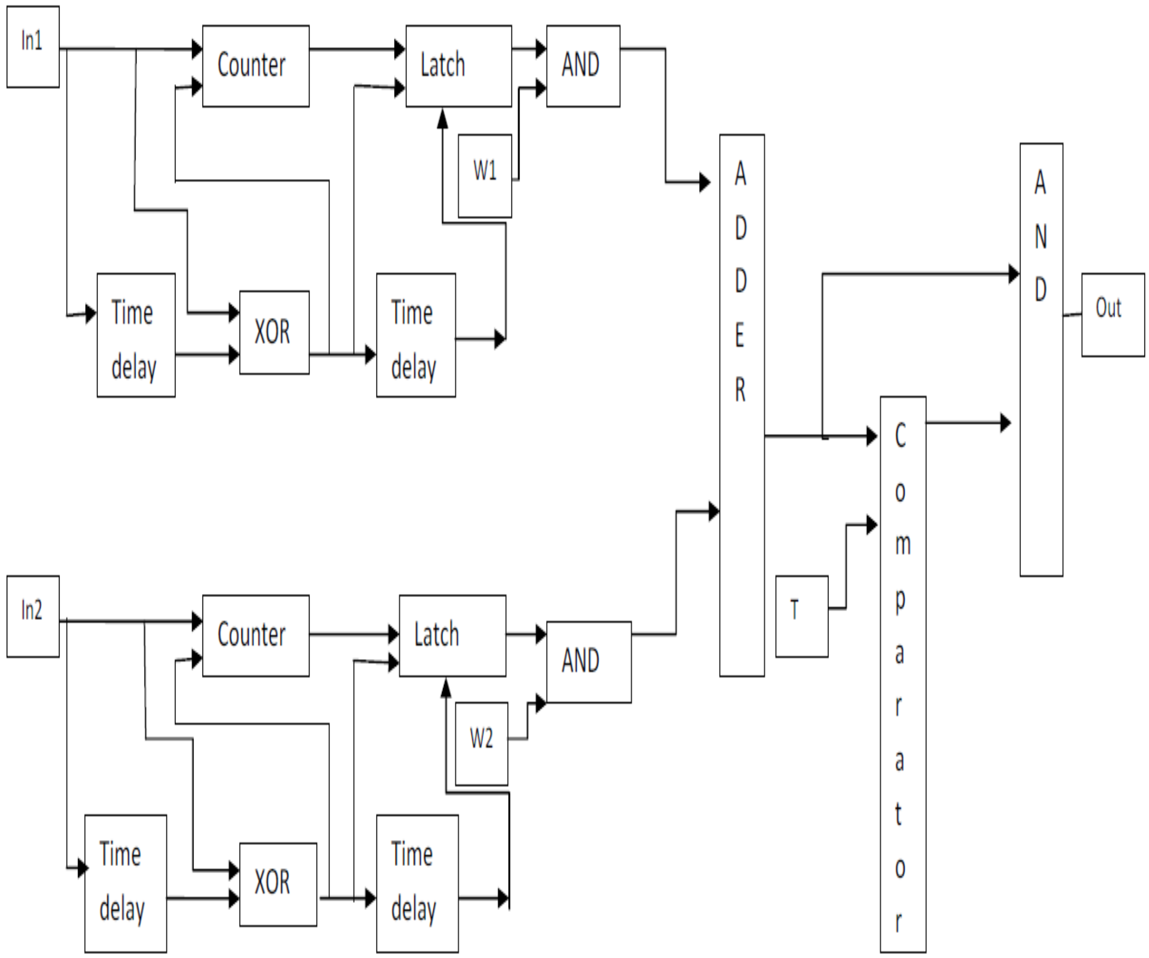


Fig. 12: Input Neuron Model

MODELING OF MIDDLE NEURON:

Middle neuron gets the inputs from input neurons to transmit onto internal neurons. Hence the inputs for middle neuron are internal. Hence there are no counters and latches required to process the inputs. There are two thresholds in the middle neuron too: One is the synaptic gap threshold, and the other activation function threshold. Below is the middle neuron model I proposed. Simulation output for the middle neuron using synopsys VCS is provided in appendix 2.

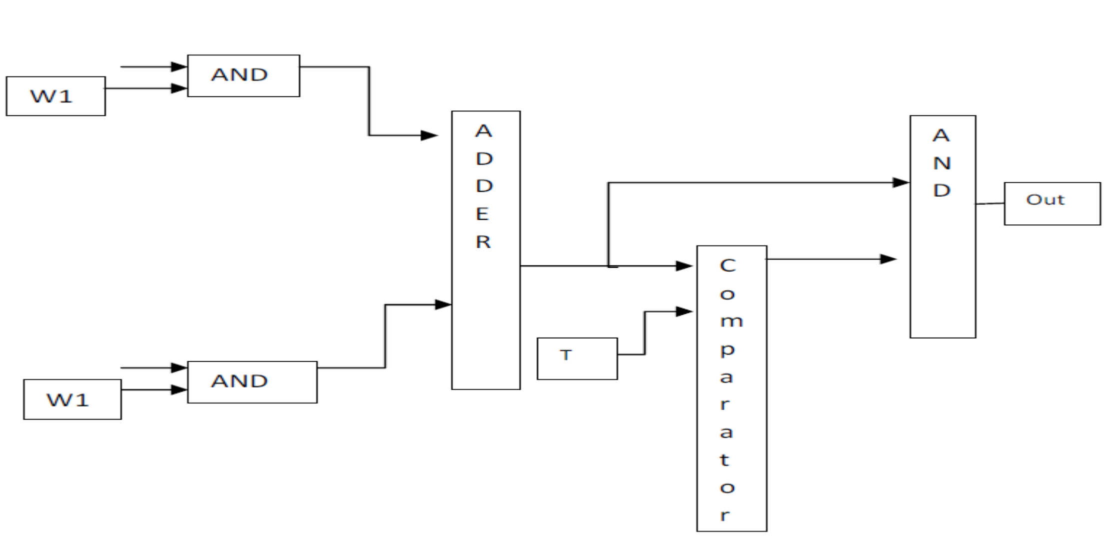


Fig. 13: Middle Neuron Model.

MODELING OF PULSAR

Pulsar converts the input signals into a pulse wave. Since, the input is processed by the input neuron, to reduce power consumption and speed. Output from middle neuron also has to be converted back to a pulse output to generate motor commands. Synopsys VCS verilog output for pulser is provided as appendix 3.

CHAPTER 4

POWER MEASUREMENTS AND OPTIMIZATION

In a biological neuron, the output from neurons is regenerated by every neuron. But, the model presented in this paper regenerates at the end of the each layer. This regeneration of power at the end of layer has reduced the power consumption; and increased the speed of the layer. Below is an example to demonstrate the power consumption:

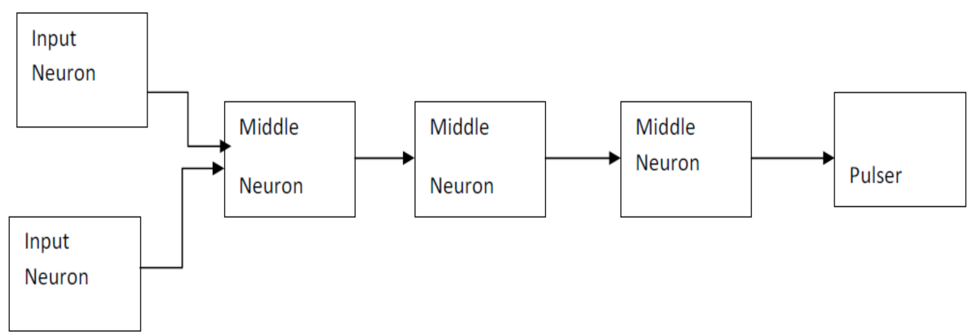


Fig. 14: Neuron Layer without Pulser Modules

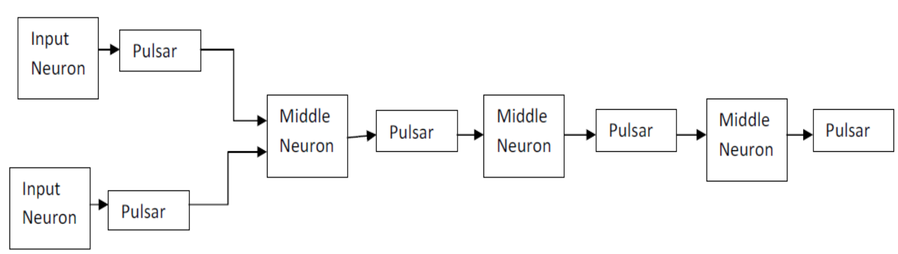
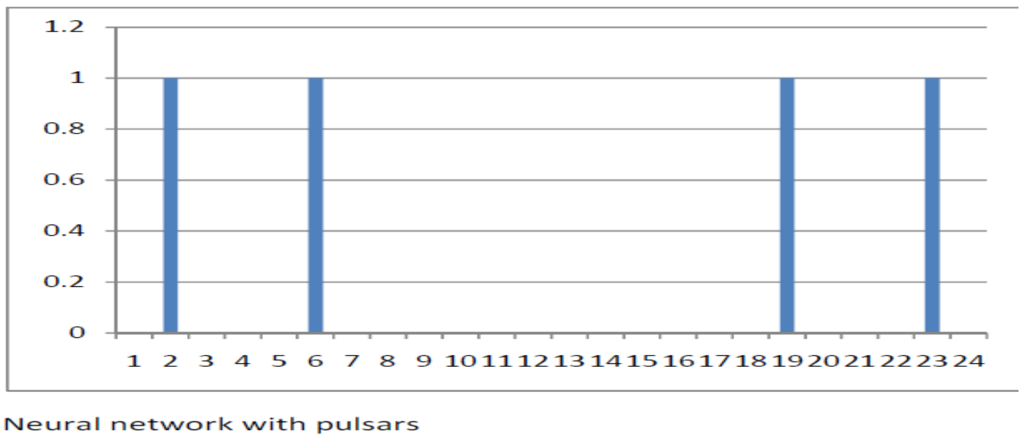


Fig. 15: Neuron Layer with Pulser Modules

Below is the output for neuron layer shown above with and without pulser modules:



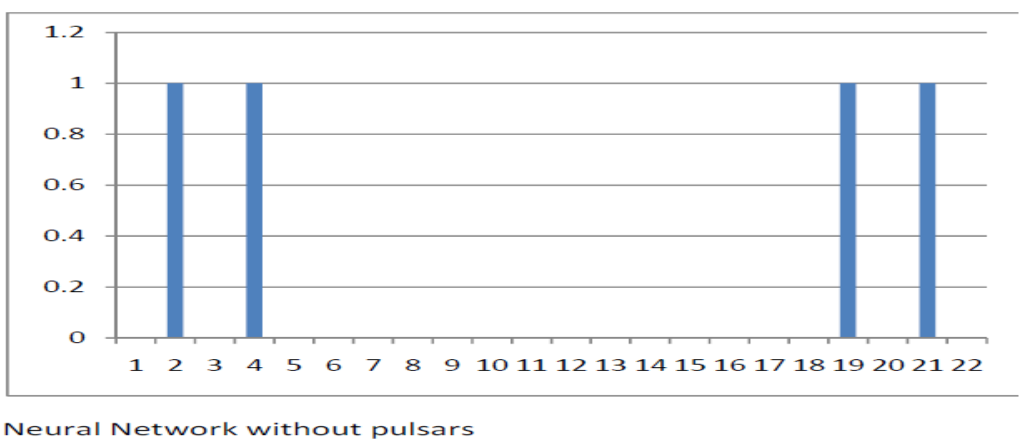


Fig. 16: Outputs for neuron layers with and without pulsers

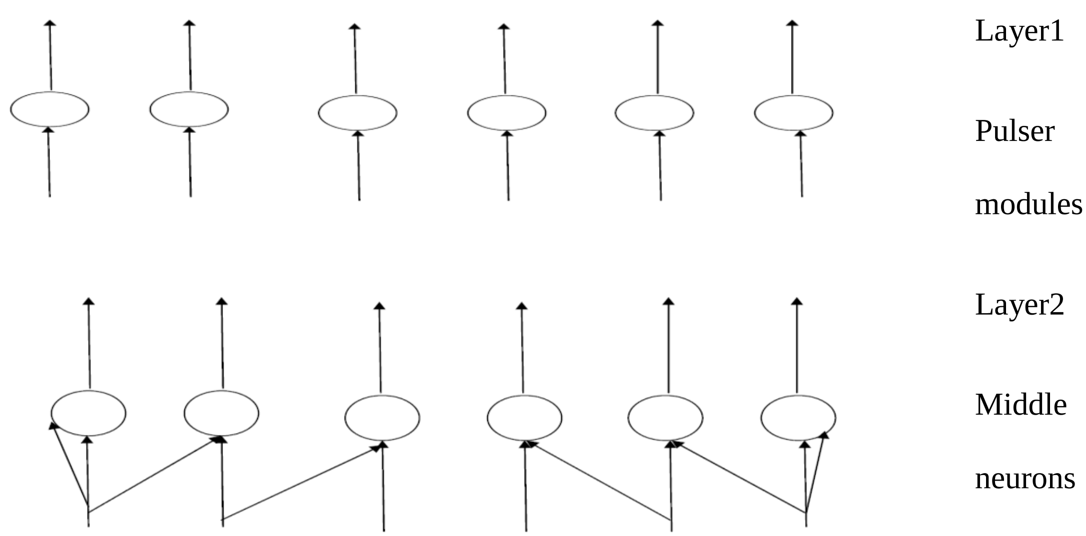
As seen from the outputs above both the modules produce same output. The power consumption is directly proportional to the switching time at every node. The switching time for the model without pulsars is 156ns. The switching time for model with pulsars is

211 ns. Hence, the power consumed by the model without pulsars at every stage is less compared to the model with pulsars. This proves that the power consumption of the optimized model proposed in this paper is much less than the biological representation.

MODELING OF LAYERS AND COLUMNS

Layers are a network of neurons. The neurons can be categorized into six layers of neuron, with the 6th layer at the bottom of neo-cortex and the rest towards the surface.

In this model, layer 6 has input neurons; layer 5, layer 4, layer3, and layer 2 have middle neurons; layer 1 has series of pulsars. The connections between the neurons and intra layers are as shown below. Outputs from the bottom layers are connected inputs of the next layer. The outputs of neurons in Layer 4 are connected to inputs of other neurons (refer layers from Chapter 2).



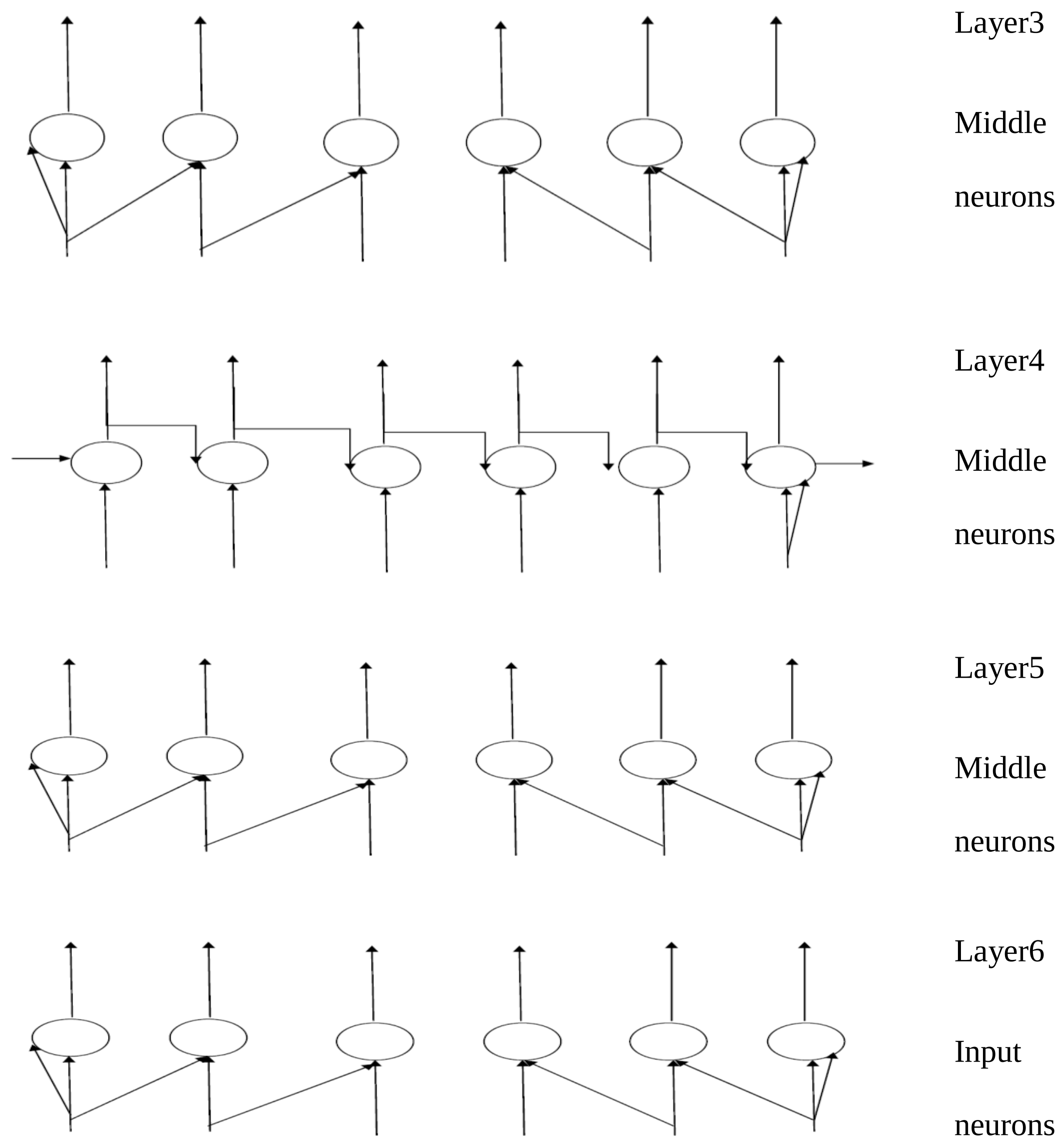


Fig. 17: Models for six layers

All the layers together form a column. Below is the model for ten columns, and each column is connected to the other through fourth layer.

Below is the model for ten columns:

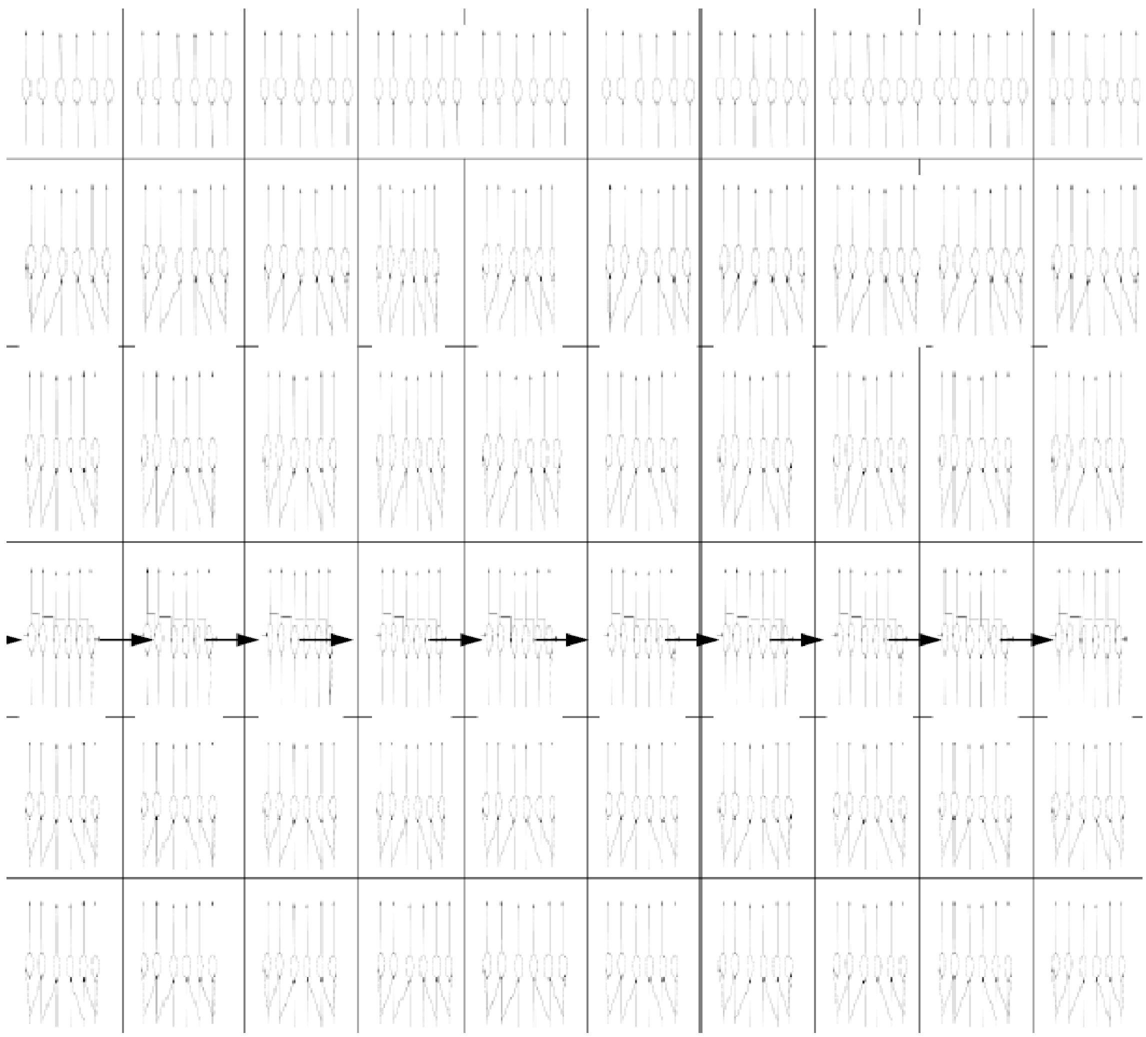


Fig. 18: Ten Columns Model

As seen in the model the outputs from layer 4 of one column are connected to the inputs of next column. Synopsys VCS Verilog output for ten columns is provided in appendix 4.

CHAPTER 5

IMPLEMENTATION AND RESULTS

The hardware and design flow used for this project are:

- Spartan3E family
- XC3S500E device
- FG320 package
- ISE Simulator (VHDL/Verilog)

SYNTHESIS RESULTS OF TEN COLUMNS USING XILINX ISE WEB PACK:

Using Xilinx, I have synthesized the simulated program of ten columns. Table III below shows the number of components used for synthesis of ten columns. Turns out that the number of components used is as expected. For example, the number of counters used for 60 input neurons are 120. i.e 2 counters for each input neuron, which matches the expected number.

Table III: HDL Synthesis Report

```
==========================================================
*                    HDL Synthesis                    *
==========================================================

HDL Synthesis Report
Macro Statistics
# Adders/Subtractors                    : 300
 4-bit adder carry out                  : 300
# Counters                              : 120
 4-bit up counter                       : 120

# Registers                             : 120
 4-bit register                         : 120
# Comparators                           : 360
 4-bit comparator greatequal            : 60
 4-bit comparator greater               : 300
# Xors                                  : 60
 2-bit xor2                             : 60
```

As seen from the table IV, the combinational delay is 225.039ns for ten columns, which is much less when compared to NCS (neo-cortical simulator) outputs. The speed grade is a parameter which shows the performance of the circuit. The performance of the circuit is directly proportional to the speed grade. -4 of speed grade is considered as a relatively good performance circuit.

Table IV: Timing Summary

Speed Grade: -4

Minimum period: 2.656ns (Maximum Frequency: 376.506MHz)

Minimum input arrival time before clock: No path found

Maximum output required time after clock: 4.496ns

Maximum combinational path delay: 225.039ns

==

==========

Process "Synthesize - XST" completed successfully

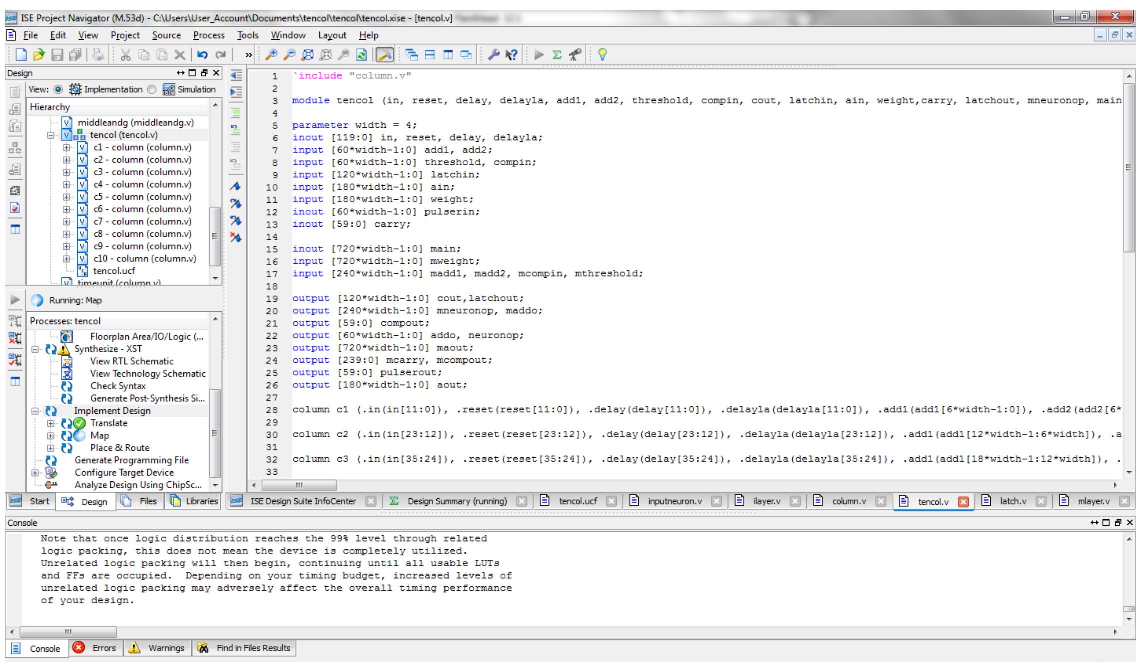


Fig 19: Translate Results

The verilog code has to be translated into an executable schematic. This process of converting the code into schematic by assigning pins for inputs and outputs is called translation. The translation summary can be seen in the fig. 19 which was successful.

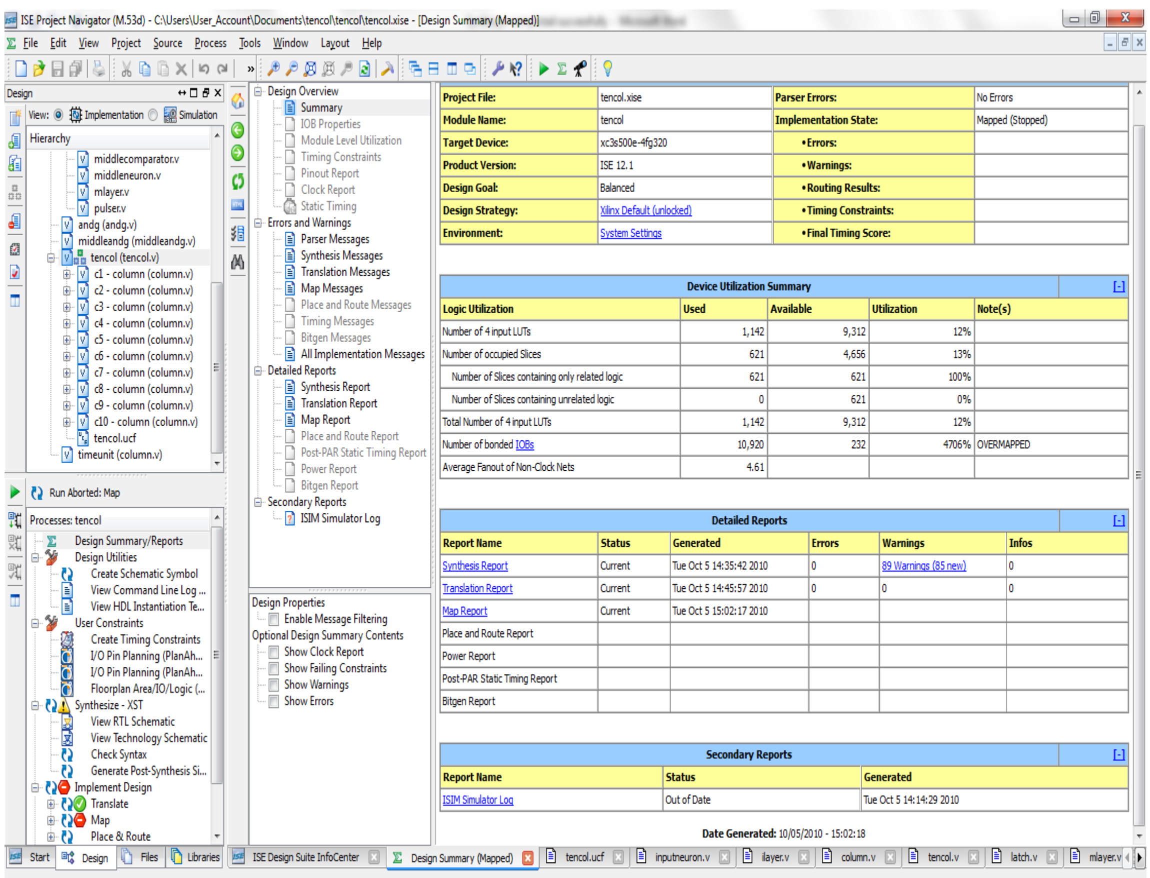


Fig 20: Summary of synthesis

The summary of synthesis can be seen in fig. 20. The number of look up tables (LUTs) used for the synthesis of ten columns is 1142. The total LUTs available for synthesis is

9312. This shows that the percentage of utilization is 12%. We can fit in a model of 3000 neurons into this FPGA which is reasonable. The synthesis had no errors and was successful.

CONCLUSION

The ten columns of a human brain are simulated and synthesized using Synopsys VCS and Xilinx ISE web pack digitally. The output of a single neuron is verified with the output of Neo-cortical simulator (NCS), a software implementation by the University of Nevada. The neuron models proposed in this thesis are optimized for optimal power consumption and time.

Appendix 1

Input neuron output

time= 0

ns,in1=0,in0=0,delay=xx,reset=xx,cout=xxxxxxxx,delayla=xx,latchout=xxxxxxxx,aout=xxxxxxxx,addo=xxxx,compout=x,neuronop= xxxx

time= 1

ns,in1=0,in0=0,delay=00,reset=00,cout=00000000,delayla=xx,latchout=xxxxxxxx,aout=xxxxxxxx,addo=xxxx,compout=x,neuronop= xxxx

time= 2

ns,in1=1,in0=1,delay=00,reset=11,cout=00010001,delayla=xx,latchout=xxxxxxxx,aout=xxxxxxxx,addo=xxxx,compout=x,neuronop= xxxx

time= 3

ns,in1=1,in0=1,delay=11,reset=00,cout=00000000,delayla=xx,latchout=00010001,aout=00010001,addo=0010,compout=1,neuronop= 0010

time= 7

ns,in1=0,in0=0,delay=11,reset=11,cout=00000000,delayla=00,latchout=00000000,aout=00000000,addo=0000,compout=0,neuronop= 0000 time= 8

ns,in1=1,in0=0,delay=00,reset=10,cout=00010000,delayla=00,latchout=00000000,aout=

00000000,addo=0000,compout=0,neuronop= 0000

time= 9

ns,in1=0,in0=1,delay=10,reset=11,cout=00010001,delayla=00,latchout=00000000,aout=

00000000,addo=0000,compout=0,neuronop= 0000

time= 10

ns,in1=1,in0=1,delay=01,reset=10,cout=00100000,delayla=00,latchout=00000000,aout=

00000000,addo=0000,compout=0,neuronop= 0000

time= 11

ns,in1=1,in0=1,delay=11,reset=00,cout=00000000,delayla=10,latchout=00100000,aout=

00100000,addo=0010,compout=1,neuronop= 0010

time= 13

ns,in1=1,in0=0,delay=11,reset=01,cout=00000000,delayla=10,latchout=00100000,aout=

00100000,addo=0010,compout=1,neuronop= 0010

time= 14

ns,in1=1,in0=1,delay=10,reset=01,cout=00000001,delayla=10,latchout=00100000,aout=

00100000,addo=0010,compout=1,neuronop= 0010

Appendix 2

Middle neuron output

time= 0 ns,main=00000000, maout=00000000,maddo=0000,mcompout=0,mneuronop=
0000

time= 2 ns,main=11111111, maout=11111111,maddo=1110,mcompout=1,mneuronop=
1110

time= 3 ns,main=00001101, maout=00001101,maddo=1101,mcompout=1,mneuronop=
1101

time= 4 ns,main=11111111, maout=11111111,maddo=1110,mcompout=1,mneuronop=
1110

time= 5 ns,main=00000000, maout=00000000,maddo=0000,mcompout=0,mneuronop=
0000

time= 6 ns,main=01011010, maout=01011010,maddo=1111,mcompout=1,mneuronop=
1111

time= 7 ns,main=00001111, maout=00001111,maddo=1111,mcompout=1,mneuronop=
1111

time=   8 ns,main=01011010, maout=01011010,maddo=1111,mcompout=1,mneuronop=
1111

time=   9 ns,main=00000000, maout=00000000,maddo=0000,mcompout=0,mneuronop=
0000

time=   10 ns,main=00001010,maout=00001010,maddo=1010,mcompout=1,mneuronop=
1010

time=   11 ns,main=00000000,maout=00000000,maddo=0000,mcompout=0,mneuronop=
0000

time=   12 ns,main=11111111,maout=11111111,maddo=1110,mcompout=1,mneuronop=
1110

time=   13 ns,main=00000000,maout=00000000,maddo=0000,mcompout=0,mneuronop=
0000

time=   14 ns,main=11111111,maout=11111111,maddo=1110,mcompout=1,mneuronop=
1110

time=   15 ns,main=00000000,maout=00000000,maddo=0000,mcompout=0,mneuronop=
0000

time=   16 ns,main=01011010,maout=01011010,maddo=1111,mcompout=1,mneuronop=
1111

Appendix 3

Pulser Output:

```
time=    0 ns, pulserin=xxxx, pulserout=x

time=    1 ns, pulserin=1111, pulserout=x

time=    2 ns, pulserin=0101, pulserout=x

time=    3 ns, pulserin=0000, pulserout=0

time=    4 ns, pulserin=0110, pulserout=0

time=    5 ns, pulserin=0010, pulserout=0

time=    6 ns, pulserin=1011, pulserout=0

time=    7 ns, pulserin=0000, pulserout=1

time=    8 ns, pulserin=0110, pulserout=0

time=    9 ns, pulserin=1110, pulserout=0

time=   10 ns, pulserin=0000, pulserout=1

time=   11 ns, pulserin=1010, pulserout=0
```

Appendix 4

Ten column output:

time=   19

ns,in=0000000000000000000000000000000000000000000000000000000000000000000

00000000000000000000000000000000000000000000000000000,

pulserout10=x01011,pulserout9=100000,pulserout8=000011,pulserout7=100011,pulsero

ut6=100011,pulserout5=100000,pulserout4=000011,pulserout3=100011,pulserout2=1000

11,pulserout1=100000

time=   20

ns,in=0000000000000000000000000000000000000000000000000000000000000000000

00000000000000000000000000000000000000000000000000000,

pulserout10=x10100,pulserout9=010100,pulserout8=000000,pulserout7=001000,pulsero

ut6=011100,pulserout5=010100,pulserout4=000000,pulserout3=001000,pulserout2=0111

00,pulserout1=010100

time=   21

ns,in=0001110001110001110001110001110001110001110001110001110001110001110

00111000111000111000111000111000111000111000111000111,

pulserout10=x01001,pulserout9=101001,pulserout8=111101,pulserout7=110101,pulsero

ut6=100001,pulserout5=101001,pulserout4=111101,pulserout3=110101,pulserout2=1000

01,pulserout1=101001

time=   22

ns,in=0000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000,

pulserout10=x10100,pulserout9=010100,pulserout8=000000,pulserout7=001000,pulsero
ut6=011100,pulserout5=010100,pulserout4=000000,pulserout3=001000,pulserout2=0111
00,pulserout1=010100

time=   23

ns,in=1111110111111111110111111111110111111111110111111111110111111111110
11111111111011111111111011111111111011111111111011111,

pulserout10=x01001,pulserout9=101001,pulserout8=111101,pulserout7=110101,

pulserout6=100001,pulserout5=101001,pulserout4=111101,pulserout3=110101,pulserout
2=100001,pulserout1=101001

All tests completed successfully

Hence, simulations for ten columns with 6 layers in each column are simulated. Next step in the process is implementation.

Appendix 5

Ten Column Code:

```
 [pvadali@hafez parameterizedtencol1]$ vi tencol.v
`include "column.v"

module tencol (in, reset, delay, delayla, add1, add2, threshold, compin, cout, latchin, ain,
weight,carry, latchout, mneuronop, main,maout, mweight, madd1, madd2, maddo,
mcompin, mthreshold, mcarry, mcompout, pulserin, pulserout, compout, addo, neuronop,
aout);

parameter width = 4;
input [119:0] in, reset, delay, delayla;
input [60*width-1:0] add1, add2;
input [60*width-1:0] threshold, compin;
input [120*width-1:0] latchin;
input [180*width-1:0] ain;
input [180*width-1:0] weight;
input [60*width-1:0] pulserin;
input [59:0] carry;

input [720*width-1:0] main;
input [720*width-1:0] mweight;
input [240*width-1:0] madd1, madd2, mcompin, mthreshold;

output [120*width-1:0] cout,latchout;
"tencol.v" [dos] 73L, 12362C                        1,1          Top
`include "column.v"
```

```
module tencol (in, reset, delay, delayla, add1, add2, threshold, compin, cout, latchin, ain,
weight,carry, latchout, mneuronop, main,maout, mweight, madd1, madd2, maddo,
mcompin, mthreshold, mcarry, mcompout, pulserin, pulserout, compout, addo, neuronop,
aout);

parameter width = 4;
input [119:0] in, reset, delay, delayla;
input [60*width-1:0] add1, add2;
input [60*width-1:0] threshold, compin;
input [120*width-1:0] latchin;
input [180*width-1:0] ain;
input [180*width-1:0] weight;
input [60*width-1:0] pulserin;
input [59:0] carry;
input [720*width-1:0] main;
input [720*width-1:0] mweight;
input [240*width-1:0] madd1, madd2, mcompin, mthreshold;

output [120*width-1:0] cout,latchout;
output [240*width-1:0] mneuronop, maddo;
output [59:0] compout;
output [60*width-1:0] addo, neuronop;
output [720*width-1:0] maout;
output [239:0] mcarry, mcompout;
output [59:0] pulserout;
output [180*width-1:0] aout;
```

```
column c1 (.in(in[11:0]), .reset(reset[11:0]), .delay(delay[11:0]), .delayla(delayla[11:0]),
.add1(add1[6*width-1:0]), .add2(add2[6*width-1:0]), .threshold(threshold[6*width-1:0]),
.compin(compin[6*width-1:0]), .cout(cout[12*width-1:0]), .latchin(latchin[12*width-
1:0]), .ain(ain[18*width-1:0]), .weight(weight[18*width-1:0]),.carry(carry[5:0]),
.latchout(latchout[12*width-1:0]), .mneuronop(mneuronop[24*width-1:0]),
.main(main[72*width-1:0]), .maout(maout[72*width-1:0]), .mweight(mweight[72*width-
1:0]), .madd1(madd1[24*width-1:0]), .madd2(madd2[24*width-1:0]),
.maddo(maddo[24*width-1:0]), .mcompin(mcompin[24*width-1:0]),
.mthreshold(mthreshold[24*width-1:0]), .mcarry(mcarry[23:0]),
.mcompout(mcompout[23:0]), .pulserin(pulserin[6*width-1:0]),
.pulserout(pulserout[5:0]), .compout(compout[5:0]), .addo(addo[6*width-1:0]),
.neuronop(neuronop[6*width-1:0]), .aout(aout[18*width-1:0]));

column c2 (.in(in[23:12]), .reset(reset[23:12]), .delay(delay[23:12]),
.delayla(delayla[23:12]), .add1(add1[12*width-1:6*width]), .add2(add2[12*width-
1:6*width]), .threshold(threshold[12*width-1:6*width]), .compin(compin[12*width-
1:6*width]), .cout(cout[24*width-1:12*width]), .latchin(latchin[24*width-1:12*width]),
.ain(ain[36*width-1:18*width]), .weight(weight[36*width-
1:18*width]),.carry(carry[11:6]), .latchout(latchout[24*width-1:12*width]),
.mneuronop(mneuronop[48*width-1:24*width]), .main(main[144*width-1:72*width]),
.maout(maout[144*width-1:72*width]), .mweight(mweight[144*width-1:72*width]),
.madd1(madd1[48*width-1:24*width]), .madd2(madd2[48*width-1:24*width]),
.maddo(maddo[48*width-1:24*width]), .mcompin(mcompin[48*width-1:24*width]),
.mthreshold(mthreshold[48*width-1:24*width]), .mcarry(mcarry[47:24]),
.mcompout(mcompout[47:24]), .pulserin(pulserin[12*width-1:6*width]),
.pulserout(pulserout[11:6]), .compout(compout[11:6]), .addo(addo[12*width-
1:6*width]), .neuronop(neuronop[12*width-1:6*width]), .aout(aout[36*width-
1:18*width]));
```

column c3 (.in(in[35:24]), .reset(reset[35:24]), .delay(delay[35:24]), .delayla(delayla[35:24]), .add1(add1[18*width-1:12*width]), .add2(add2[18*width-1:12*width]), .threshold(threshold[18*width-1:12*width]), .compin(compin[18*width-1:12*width]), .cout(cout[36*width-1:24*width]), .latchin(latchin[36*width-1:24*width]), .ain(ain[54*width-1:36*width]), .weight(weight[54*width-

1:36*width]),.carry(carry[17:12]), .latchout(latchout[36*width-1:24*width]), .mneuronop(mneuronop[72*width-1:48*width]), .main(main[216*width-1:144*width]), .maout(maout[216*width-1:144*width]), .mweight(mweight[216*width-1:144*width]), .madd1(madd1[72*width-1:48*width]), .madd2(madd2[72*width-1:48*width]), .maddo(maddo[72*width-1:48*width]), .mcompin(mcompin[72*width-1:48*width]), .mthreshold(mthreshold[72*width-1:48*width]), .mcarry(mcarry[71:48]), .mcompout(mcompout[71:48]), .pulserin(pulserin[18*width-1:12*width]), .pulserout(pulserout[17:12]), .compout(compout[17:12]), .addo(addo[18*width-1:12*width]), .neuronop(neuronop[18*width-1:12*width]), .aout(aout[54*width-1:36*width]));

column c4 (.in(in[47:36]), .reset(reset[47:36]), .delay(delay[47:36]), .delayla(delayla[47:36]), .add1(add1[24*width-1:18*width]), .add2(add2[24*width-1:18*width]), .threshold(threshold[24*width-1:18*width]), .compin(compin[24*width-1:18*width]), .cout(cout[48*width-1:36*width]), .latchin(latchin[48*width-1:36*width]), .ain(ain[72*width-1:54*width]), .weight(weight[72*width-1:54*width]),.carry(carry[23:18]), .latchout(latchout[48*width-1:36*width]), .mneuronop(mneuronop[96*width-1:72*width]), .main(main[288*width-1:216*width]), .maout(maout[288*width-1:216*width]), .mweight(mweight[288*width-1:216*width]), .madd1(madd1[96*width-1:72*width]), .madd2(madd2[96*width-1:72*width]), .maddo(maddo[96*width-1:72*width]), .mcompin(mcompin[96*width-1:72*width]), .mthreshold(mthreshold[96*width-1:72*width]), .mcarry(mcarry[95:72]),

.mcompout(mcompout[95:72]), .pulserin(pulserin[24*width-1:18*width]), .pulserout(pulserout[23:18]), .compout(compout[23:18]), .addo(addo[24*width-1:18*width]), .neuronop(neuronop[24*width-1:18*width]), .aout(aout[72*width-1:54*width]));

column c5 (.in(in[59:48]), .reset(reset[59:48]), .delay(delay[59:48]), .delayla(delayla[59:48]), .add1(add1[30*width-1:24*width]), .add2(add2[30*width-1:24*width]), .threshold(threshold[30*width-1:24*width]), .compin(compin[30*width-1:24*width]), .cout(cout[60*width-1:48*width]), .latchin(latchin[60*width-1:48*width]), .ain(ain[90*width-1:72*width]), .weight(weight[90*width-1:72*width]),.carry(carry[29:24]), .latchout(latchout[60*width-1:48*width]), .mneuronop(mneuronop[120*width-1:96*width]), .main(main[360*width-1:288*width]), .maout(maout[360*width-1:288*width]), .mweight(mweight[360*width-1:288*width]), .madd1(madd1[120*width-1:96*width]), .madd2(madd2[120*width-1:96*width]), .maddo(maddo[120*width-1:96*width]), .mcompin(mcompin[120*width-1:96*width]), .mthreshold(mthreshold[120*width-1:96*width]), .mcarry(mcarry[119:96]), .mcompout(mcompout[119:96]), .pulserin(pulserin[30*width-1:24*width]), .pulserout(pulserout[29:24]), .compout(compout[29:24]), .addo(addo[30*width-1:24*width]), .neuronop(neuronop[30*width-1:24*width]), .aout(aout[90*width-1:72*width]));

column c6 (.in(in[71:60]), .reset(reset[71:60]), .delay(delay[71:60]), .delayla(delayla[71:60]), .add1(add1[36*width-1:30*width]), .add2(add2[36*width-1:30*width]), .threshold(threshold[36*width-1:30*width]), .compin(compin[36*width-1:30*width]), .cout(cout[72*width-1:60*width]), .latchin(latchin[72*width-1:60*width]), .ain(ain[108*width-1:90*width]), .weight(weight[108*width-1:90*width]),.carry(carry[35:30]), .latchout(latchout[72*width-1:60*width]), .mneuronop(mneuronop[144*width-

1:120*width]), .main(main[432*width-1:360*width]), .maout(maout[432*width-1:360*width]), .mweight(mweight[432*width-1:360*width]), .madd1(madd1[144*width-1:120*width]), .madd2(madd2[144*width-1:120*width]), .maddo(maddo[144*width-1:120*width]), .mcompin(mcompin[144*width-1:120*width]), .mthreshold(mthreshold[144*width-1:120*width]), .mcarry(mcarry[143:120]), .mcompout(mcompout[143:120]), .pulserin(pulserin[36*width-1:30*width]), .pulserout(pulserout[35:30]), .compout(compout[35:30]), .addo(addo[36*width-1:30*width]), .neuronop(neuronop[36*width-1:30*width]), .aout(aout[108*width-1:90*width]));

column c7 (.in(in[83:72]), .reset(reset[83:72]), .delay(delay[83:72]), .delayla(delayla[83:72]), .add1(add1[42*width-1:36*width]), .add2(add2[42*width-1:36*width]), .threshold(threshold[42*width-1:36*width]), .compin(compin[42*width-1:36*width]), .cout(cout[84*width-1:72*width]), .latchin(latchin[84*width-1:72*width]), .ain(ain[126*width-1:108*width]), .weight(weight[126*width-1:108*width]),.carry(carry[41:36]), .latchout(latchout[84*width-1:72*width]), .mneuronop(mneuronop[168*width-1:144*width]), .main(main[504*width-1:432*width]), .maout(maout[504*width-1:432*width]), .mweight(mweight[504*width-1:432*width]), .madd1(madd1[168*width-1:144*width]), .madd2(madd2[168*width-1:144*width]), .maddo(maddo[168*width-1:144*width]), .mcompin(mcompin[168*width-1:144*width]), .mthreshold(mthreshold[168*width-1:144*width]), .mcarry(mcarry[167:144]), .mcompout(mcompout[167:144]), .pulserin(pulserin[42*width-1:36*width]), .pulserout(pulserout[41:36]), .compout(compout[41:36]), .addo(addo[42*width-1:36*width]), .neuronop(neuronop[42*width-1:36*width]), .aout(aout[126*width-1:108*width]));

column c8 (.in(in[95:84]), .reset(reset[95:84]), .delay(delay[95:84]), .delayla(delayla[95:84]), .add1(add1[48*width-1:42*width]), .add2(add2[48*width-

1:42*width]), .threshold(threshold[48*width-1:42*width]), .compin(compin[48*width-1:42*width]), .cout(cout[96*width-1:84*width]), .latchin(latchin[96*width-1:84*width]), .ain(ain[144*width-1:126*width]), .weight(weight[144*width-1:126*width]),.carry(carry[47:42]), .latchout(latchout[96*width-1:84*width]), .mneuronop(mneuronop[192*width-1:168*width]), .main(main[576*width-1:504*width]), .maout(maout[576*width-1:504*width]), .mweight(mweight[576*width-1:504*width]), .madd1(madd1[192*width-1:168*width]), .madd2(madd2[192*width-1:168*width]), .maddo(maddo[192*width-1:168*width]), .mcompin(mcompin[192*width-1:168*width]), .mthreshold(mthreshold[192*width-1:168*width]), .mcarry(mcarry[191:168]), .mcompout(mcompout[191:168]), .pulserin(pulserin[48*width-1:42*width]), .pulserout(pulserout[47:42]), .compout(compout[47:42]), .addo(addo[48*width-1:42*width]), .neuronop(neuronop[48*width-1:42*width]), .aout(aout[144*width-1:126*width]));

column c9 (.in(in[107:96]), .reset(reset[107:96]), .delay(delay[107:96]), .delayla(delayla[107:96]), .add1(add1[54*width-1:48*width]), .add2(add2[54*width-1:48*width]), .threshold(threshold[54*width-1:48*width]), .compin(compin[54*width-1:48*width]), .cout(cout[108*width-1:96*width]), .latchin(latchin[108*width-1:96*width]), .ain(ain[162*width-1:144*width]), .weight(weight[162*width-1:144*width]),.carry(carry[53:48]), .latchout(latchout[108*width-1:96*width]), .mneuronop(mneuronop[216*width-1:192*width]), .main(main[648*width-1:576*width]), .maout(maout[648*width-1:576*width]), .mweight(mweight[648*width-1:576*width]), .madd1(madd1[216*width-1:192*width]), .madd2(madd2[216*width-1:192*width]), .maddo(maddo[216*width-1:192*width]), .mcompin(mcompin[216*width-1:192*width]), .mthreshold(mthreshold[216*width-1:192*width]), .mcarry(mcarry[215:192]), .mcompout(mcompout[215:192]), .pulserin(pulserin[54*width-1:48*width]), .pulserout(pulserout[53:48]), .compout(compout[53:48]), .addo(addo[54*width-

```
1:48*width]), .neuronop(neuronop[54*width-1:48*width]), .aout(aout[162*width-
1:144*width]));
column c10 (.in(in[119:108]), .reset(reset[119:108]), .delay(delay[119:108]),
.delayla(delayla[119:108]), .add1(add1[60*width-1:54*width]), .add2(add2[60*width-
1:54*width]), .threshold(threshold[60*width-1:54*width]), .compin(compin[60*width-
1:54*width]), .cout(cout[120*width-1:108*width]), .latchin(latchin[120*width-
1:108*width]), .ain(ain[180*width-1:162*width]), .weight(weight[180*width-
1:162*width]),.carry(carry[59:54]), .latchout(latchout[120*width-1:108*width]),
.mneuronop(mneuronop[240*width-1:216*width]), .main(main[720*width-
1:648*width]), .maout(maout[720*width-1:648*width]), .mweight(mweight[720*width-
1:648*width]), .madd1(madd1[240*width-1:216*width]), .madd2(madd2[240*width-
1:216*width]), .maddo(maddo[240*width-1:216*width]),
.mcompin(mcompin[240*width-1:216*width]), .mthreshold(mthreshold[240*width-
1:216*width]), .mcarry(mcarry[239:216]), .mcompout(mcompout[239:216]),
.pulserin(pulserin[60*width-1:54*width]), .pulserout(pulserout[59:54]),
.compout(compout[59:54]), .addo(addo[60*width-1:54*width]),
.neuronop(neuronop[60*width-1:54*width]), .aout(aout[180*width-1:162*width]));
assign main[683*width-1:682*width] = {width{1'b1}};
assign main[611*width-1:610*width] = mneuronop[223*width-1:222*width];assign
main[539*width-1:538*width] = mneuronop[199*width-1:198*width];assign
main[467*width-1:466*width] = mneuronop[175*width-1:174*width];assign
main[395*width-1:394*width] = mneuronop[151*width-1:150*width];assign
main[323*width-1:322*width] = mneuronop[127*width-1:126*width];assign
main[251*width-1:250*width] = mneuronop[103*width-1:102*width];assign
main[179*width-1:178*width] = mneuronop[79*width-1:78*width];assign
main[107*width-1:106*width] = mneuronop[55*width-1:54*width];assign
main[35*width-1:34*width] = mneuronop[31*width-1:30*width];
endmodule
```

Appendix 6

TEN COLUMNS TEST BENCH:

```
[pvadali@hafez parameterizedtencol1]$ vi tencol_tb.v
module tencol_tb ();

parameter width = 4;

wire [119:0] reset, delay, delayla;
wire [60*width-1:0] add1, add2;
wire [60*width-1:0] compin;
wire [120*width-1:0] latchin;
wire [180*width-1:0] ain;
wire [60*width-1:0] pulserin;
wire [59:0] carry;
wire [720*width-1:0] main;
wire [240*width-1:0] madd1, madd2, mcompin;
wire [120*width-1:0] cout,latchout;
wire [240*width-1:0] mneuronop, maddo;
wire [59:0] compout;
wire [60*width-1:0] addo, neuronop;
wire [720*width-1:0] maout;
wire [239:0] mcarry, mcompout;
wire [59:0] pulserout;
wire [180*width-1:0] aout;
integer i;
reg [119:0] in;
reg [60*width-1:0] threshold;
reg [180*width-1:0] weight;
```

```
reg [720*width-1:0] mweight;
reg [240*width-1:0] mthreshold;
tencol dut (.neuronop(neuronop[60*width-1:0]),
        .cout(cout[120*width-1:0]),
        .latchout(latchout[120*width-1:0]),
        .aout(aout[180*width-1:0]),
        .addo(addo[60*width-1:0]),
        .compout(compout[59:0]),
        .mneuronop(mneuronop[240*width-1:0]),
        .maddo(maddo[240*width-1:0]),
        .maout(maout[720*width-1:0]),
        .mcarry(mcarry[239:0]),
        .mcompout(mcompout[239:0]),
        .pulserout(pulserout[59:0]),
        .delayla(delayla[119:0]),
        .delay(delay[119:0]),
        .reset(reset[119:0]),
        .latchin(latchin[120*width-1:0]),
        .ain(ain[180*width-1:0]),
        .add1(add1[60*width-1:0]),
        .add2(add2[60*width-1:0]),
        .compin(compin[60*width-1:0]),
        .threshold(threshold[60*width-1:0]),
        .weight(weight[180*width-1:0]),
        .in(in[119:0]),
        .main(main[720*width-1:0]),
        .mweight(mweight[720*width-1:0]),
        .madd1(madd1[240*width-1:0]),
```

```
.madd2(madd2[240*width-1:0]),
.mcompin(mcompin[240*width-1:0]),
.carry(carry[59:0]),
.mthreshold(mthreshold[240*width-1:0]),
.pulserin(pulserin[60*width-1:0]));

initial
begin

$monitor ("time=%5d ns,in=%b,
pulserout10=%b,pulserout9=%b,pulserout8=%b,pulserout7=%b,pulserout6=%b,pulserou
t5=%b,pulserout4=%b,pulserout3=%b,pulserout2=%b,pulserout1=%b",$time, in[119:0],
pulserout[59:54],pulserout[53:48],pulserout[47:42],pulserout[41:36],pulserout[35:30],pul
serout[29:24],pulserout[23:18],pulserout[17:12],pulserout[11:6],pulserout[5:0]);

threshold = 0;
mthreshold = 0;

in= 0;in = #1
120'b0001110001110001110001110001110001110001110001110001110001110001110
00111000111000111000111000111000111000111000111000111;//11in = #1
120'b0000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000;//12in = #1

120'b1111110111111111110111111111110111111111110111111111110111111111110
11111111111011111111111011111111111011111111111011111;//13in = #1
120'b0000001000000000001000000000001000000000001000000000001000000000001
00000000000100000000000100000000000100000000000100000;//14in = #1
```

120'b11100011111111100011111111100011111111100011111111100011111111110001
111111110001111111110001111111110001111111110001111111;//15in = #1
120'b00011010011100011010011100011010011100011010011100011010011100011101
0011100011010011100011010011100011010011100011010011 1;//16in = #1
120'b11100100000011100100000011100100000011100100000011100100000011110010
000001110010000001110010000001110010000001110010000000;//17in = #1
120'b00000000000001110010000001110010000001110010000001110010000001110010
000001110010000001110010000001110010000001110010000000;//1#in = #1
120'b00011010011100011010011100011010011100011010011100011010011100011101
0011100011010011100011010011100011010011100011010011 1;//2in = #1
120'b11100101100011100101100011100101100011100101100011100101100011110010
1100011100101100011100101100011100101100011100101 1000;//3in = #1
120'b00011010011100011010011100011010011100011010011100011010011100011101
0011100011010011100011010011100011010011100011010011 1;//4in = #1
120'b11100001100011100001100011100001100011100001100011100001100011110000
1100011100001100011100001100011100001100011100001 1000;//5in = #1
120'b00011010011100011010011100011010011100011010011100011010011100011101
0011100011010011100011010011100011010011100011010011 1;//6in = #1
120'b11111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111;//7in = #1
120'b00011011100000001101110000000110111000000011011100000001101110000001101
1100000001101110000000110111000000011011100000001101 11000;//8in = #1
120'b11100110011111100110011111100110011111100110011111100110011111110011
0011111100110011111100110011111100110011111100110 0111;//9in = #1
120'b11111011100011111011100011111011100011111011100011111011100011111101
1100011111011100011111011100011111011100011111011 1000;//10in = #1
120'b00011100011100011100011100011100011100011100011100011100011100011110
00111000111000111000111000111000111000111000111000111;//11in = #1

120'b0000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000;//12in = #1
120'b0000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000;in= 0;in = #1
120'b0001110001110001110001110001110001110001110001110001110001110001110
00111000111000111000111000111000111000111000111000111;//11in = #1
120'b0000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000;//12in = #1
120'b1111110111111111110111111111110111111111110111111111110111111111110
11111111111011111111111011111111111011111111111011111;//13in = #1
120'b0000001000000000001000000000001000000000001000000000001000000000001
00000000000100000000000100000000000100000000000100000;//14
in= 0;in = #1
120'b0001110001110001110001110001110001110001110001110001110001110001110
00111000111000111000111000111000111000111000111000111;//11in = #1
120'b0000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000;//12in = #1
120'b1111110111111111110111111111110111111111110111111111110111111111110
11111111111011111111111011111111111011111111111011111;//13in = #1
120'b0000001000000000001000000000001000000000001000000000001000000000001
00000000000100000000000100000000000100000000000100000;//14in = #1
120'b1110001111111110001111111110001111111110001111111110001111111110001
11111111000111111111000111111111000111111111000111111;//15in = #1
120'b0001101001110001101001110001101001110001101001110001101001110001101
00111000110100111000110100111000110100111000110100111;//16in = #1
120'b1110010000001110010000001110010000001110010000001110010000001110010
00000111001000000111001000000111001000000111001000000;//17in = #1
120'b0000000000001110010000001110010000001110010000001110010000001110010

```
00000111001000000111001000000111001000000111001000000;//1#in = #1
120'b0001101001110001101001110001101001110001101001110001101001110001101
00111000110100111000110100111000110100111000110100111;//2in = #1
120'b1110010110001110010110001110010110001110010110001110010110001110010
11000111001011000111001011000111001011000111001011000;//3in = #1
120'b0001101001110001101001110001101001110001101001110001101001110001101
00111000110100111000110100111000110100111000110100111;//4in = #1
120'b1110000110001110000110001110000110001110000110001110000110001110000
11000111000011000111000011000111000011000111000011000;//5in = #1
120'b0001101001110001101001110001101001110001101001110001101001110001101
00111000110100111000110100111000110100111000110100111;//6in = #1
120'b1111111111111111111111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111111;//7in = #1
120'b0001101110000000110111000000011011100000001101110000000110111000000011101
11000000011011100000001101110000000110111000000110111000;//8in = #1
120'b1110011001111110011001111110011001111110011001111110011001111110011
00111111001100111111001100111111001100111111001100111;//9in = #1
120'b1111101110001111101110001111101110001111101110001111101110001111101
11000111110111000111110111000111110111000111110111000;//10in = #1
120'b0001110001110001110001110001110001110001110001110001110001110001110
00111000111000111000111000111000111000111000111000111;//11in = #1
120'b0000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000;//12in = #1
120'b0000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000;in= 0;in = #1
120'b0001110001110001110001110001110001110001110001110001110001110001110
00111000111000111000111000111000111000111000111000111;//11in = #1
120'b0000000000000000000000000000000000000000000000000000000000000000000
```

```
00000000000000000000000000000000000000000000000000000;//12in = #1
120'b1111110111111111110111111111110111111111110111111111110111111111110
11111111111011111111111011111111111011111111111011111;//13in = #1
120'b0000001000000000001000000000001000000000001000000000001000000000001
00000000000100000000000100000000000100000000000100000;//14

$display("All tests completed sucessfully\n\n");
$finish;end
initial
begin$dumpfile ("tencol.dump");
$dumpvars (0, tencol_tb);end
endmodule
```

REFERENCES

[1] Blinkov, S.M. and Glezer, I.I, "*The Human Brain in Figures and Tables.* A Quantitative Handbook," New York: Plenum Press, 1968.

[2] Shadbolt, Nigel, "Brain Power,". IEEE Intelligent Systems. 2003.

[3] Harris R. Lieberman, Robin B. Kanarek, and Chandan Prasad, "Nutrition, Brain and Behavior," New Orleans : CRC Press Taylor and Francis Group, 2005

[4] Simon B. Laughlin, Rob R. de Ruyter van Steveninck, John C. Anderson, "The metabolic cost of neural information," Nature Neuroscience . 1, 1998, Vol. 36, 41.

[5] Frederic H. Martini and Judi L. Nath, "**Fundamentals of Anatomy & Physiology," 8th Edition**.

[7] Pearson Success net: http://www.pearsonsuccessnet.com/snpapp/iText/products/0-13-115075-8/text/chapter28/concept28.1.html

[8] Information processing in human body: http://vadim.oversigma.com/MAS862/Project.html

[9] How nerves work: http://health.howstuffworks.com/nerve.htm/printable

[10] I.E. Ngoledingc, R.N.G. Nnguib mid S.S. D l q. A New Class of Analogue CMOS Neural Network Circuits, Department of Electrical and Electronic Engineering University of Newcastle upon Tyne, Newcastle Upon Tyne, NE1 7RU, UK.

[11] Blue Brain Projects: http://domino.watson.ibm.com/comm/pr.nsf/pages/rsc.bluegene_cognitive.html

[12] Neurongrid: https://www.stanford.edu/group/brainsinsilicon/about.html

[13] Neo- Cortical simulator: www.brain.unr.edu

[14] James A. Anderson,Department,“A Brain-Like Computer for Cognitive Software Applications:The Ersatz Brain Project,” Cognitive and Linguistic Sciences, Brown University Providence

[17] Brain Facts and Figures: http://faculty.washington.edu/chudler/facts.html