1-1-2011


Robert A. Nawrocki

University of Denver

Follow this and additional works at: https://digitalcommons.du.edu/etd

Part of the Artificial Intelligence and Robotics Commons, Computer Engineering Commons, and the Electrical and Computer Engineering Commons

Recommended Citation

https://digitalcommons.du.edu/etd/891
SIMULATION, APPLICATION, AND RESILIENCE OF AN ORGANIC NEUROMORPHIC ARCHITECTURE, MADE WITH ORGANIC BISTABLE DEVICES AND ORGANIC FIELD EFFECT TRANSISTORS

A Thesis
Presented to
The Faculty of Engineering and Computer Science
University of Denver

In Partial Fulfillment
of the Requirements for the Degree
Master of Science

by
Robert A. Nawrocki
August 2011
Advisor: Dr. Richard M. Voyles
ABSTRACT

This thesis presents work done simulating a type of organic neuromorphic architecture, modeled after Artificial Neural Network, and termed Synthetic Neural Network, or SNN. The first major contribution of this thesis is development of a single-transistor-single-organic-bistable-device-per-input circuit that approximates behavior of an artificial neuron. The efficacy of this design is validated by comparing the behavior of a single synthetic neuron to that of an artificial neuron as well as two examples involving a network of synthetic neurons. The analysis utilizes electrical characteristics of polymer electronic elements, namely Organic Bistable Device and Organic Field Effect Transistor, created in the laboratory at University of Denver. Polymer electronics is a new branch of electronics that is based on conductive and semi-conductive polymers. These new elements hold a great advantage over the inorganic electronics in the form of physical flexibility and low cost of fabrication. However, their device variability between individual devices is also much greater. Therefore the second major contribution of this thesis is the analysis of resilience of neural networks subjected to physical damage and other manufacturing faults.
Acknowledgements

First of all, I would like to express my deepest gratitude to both of my advisers, Professor Richard M. Voyles and Professor Sean E. Shaheen, for their guidance and patience with my graduate studies and my research. I would also like to thank Professor Roger Salters and Professor Wenzhong “David” Gao for serving as the committee members in my final oral examination. Additionally I would like to express a profound appreciation to Professor Nancy M. Lorenzon and Dr. Majid Shaalan for their direct and indirect help in my research at University of Denver.

Secondly, I thank all my lab mates, in both the Engineering Department as well as in the Physics Department, especially, Xiaoting Yang and Rachelle Cobb from Engineering Department as well as Alisha Humphries, Xin Jiang and Brian Bailey from Physics and Astronomy Department. Xiaoting performed majority of the work on Water Hammer actuation, which served as a test board for real-world application of SNN as well as its immediate future application. Rachelle was responsible for creation of an organic field effect transistor. Both Alisha and Xin were involved in the early stages of creation of organic Bistable Device. Brian was instrumental in creating the organic Bistable Device presented here.

Last but not least, I would like to give special thanks and dedicate this thesis to my immediate family, my mother and my father, and my sister and her husband, for their love, concern, support and encouragement. I thank you all.

My work was sponsored in part by NSF Safety, Security, and Rescue Research Center and the NSF under grants IIS-0841483, CNS-0923518 and OISE-1053249.
# Table of Contents

**CHAPTER ONE: INTRODUCTION** ............................................................................................................. 1

  Motivation .................................................................................................................................................. 1
  Thesis Overview ......................................................................................................................................... 4
  Review of Literature ................................................................................................................................. 5
  Background ................................................................................................................................................ 10

**CHAPTER TWO: ARCHITECTURE** ....................................................................................................... 12

  Bistable Device as a Synapse for Connection Weight ............................................................................... 15
  Single Transistor Circuit as a Soma for Summation and Activation ......................................................... 16

**CHAPTER THREE: INDIVIDUAL COMPONENTS** .................................................................................... 20

  Organic Electronics ..................................................................................................................................... 20
  Organic Bistable Device ............................................................................................................................... 24
  Fabrication .................................................................................................................................................. 25
    Recipe ...................................................................................................................................................... 27
  Switching Mechanism ................................................................................................................................. 34
    Electric Field Induction ............................................................................................................................. 34
    Formation of Conductive Filaments or Redox Behavior ......................................................................... 35
    Charge Trapping ....................................................................................................................................... 36
  Organic Field Effect Transistor .................................................................................................................. 38

**CHAPTER FOUR: SINGLE NEURON SIMULATION** ............................................................................... 42

  A Single Synthetic Neuron Behavior .......................................................................................................... 42
  Single Synthetic Neuron as a Logic Gate .................................................................................................... 44
  SNN Input-Output Relationship .................................................................................................................. 46

**CHAPTER FIVE: SIMULATION OF A NETWORK OF SYNTHETIC NEURONS** ........................................ 49

  Structured Computational Polymers .......................................................................................................... 49
    Actuation – Water Hammer ....................................................................................................................... 51
    WH on SNN .............................................................................................................................................. 52
  Synthetic Neuronal Network for Wall Following Robot ............................................................................. 54

**CHAPTER SIX: DAMAGE RESILIENCE AND MANUFACTURING ERRORS** ..................................... 59

  Background ................................................................................................................................................ 61
  Method ....................................................................................................................................................... 63
    Software .................................................................................................................................................. 63
    Network Architecture .............................................................................................................................. 63
    Data Sets ............................................................................................................................................... 66
    Optimization .......................................................................................................................................... 67
  Procedure ................................................................................................................................................ 69
  Results and Discussions ............................................................................................................................. 74
  Conclusion ................................................................................................................................................. 83

**CHAPTER SEVEN: SUMMARY AND FUTURE WORK** ........................................................................... 85

  Future Work ............................................................................................................................................. 86

**REFERENCES** ......................................................................................................................................... 88

**APPENDIX A: COMPREHENSIVE SET OF ANN ‘STUCK-AT-0’ FAULT GRAPHS** .................................. 97

**APPENDIX B: COMPREHENSIVE SET OF ANN ‘STUCK-AT-1’ FAULT GRAPHS** ............................. 105
List of Figures

Figure 1. (a) Biological neuron with dendrites and axon (input and output “wires”), synapse and soma being outlined (image from Wikipedia.org). (b) Graphical representation of operation of a single artificial neural, also shown in Equation 1: Dendrites and Axon act as input and output pathways, while artificial synapse (connection weight) propagates input signal proportional to the importance of that particular input. Soma produces proportional output (activation function) that is the sum of all of the input signals. .................................................................14

Figure 2. Operation of an ideal Bistable Device shown as an I-V characteristic curve of the device. A Bistable Device (BD) is either in the OFF state (R_{off} – high resistance, step 1 and 5, ‘red’ or light in grayscale) or ON state (R_{on} – low resistance, step 3, ‘blue’ or dark in grayscale) depending on past voltage. ‘Green’ represents switching, to either ON (step 2) or OFF (step 4) state. The element can be turned ON by increasing the input voltage above the positive threshold value (V_{ON}). Analogously, it can be turned OFF by decreasing the input voltage below the negative threshold value (V_{OFF}). ........................................................................................................16

Figure 3. (a) Schematics of a single neuron with two inputs (a binary connection weight, formed by a single BD), denoted by V_{IN1} and V_{IN2}, and one output, marked V_{OUT}. An organic field-effect transistor (OFET) is used for summation of input currents and to provide activation function. A symbol of a memristor in lieu of BD is used as such a device can be thought of as a memristor operating in its saturation region. (b) The same design showing, for the purposes of clarity, BD represented by two resistors: R_{ON} and R_{OFF}. ........................................................................................................18

Figure 4. Physically flexible organic light emitting diode (OLED) shown (a) being flexed and (b) lit-up. In this case the device activates (a dim light can be seen) at around 6 V. However, only around 12V can light become clearly visible. More advanced and commercially available OLEDs have much lower drive voltages. ........................................................................................................21

Figure 5. (a) Spin Coater, an inexpensive machine, which relies on centrifugal force, used to apply a uniform film of a material with the thickness of tens of nanometers with the accuracy of a few nanometers. (b) Spray coating gun used to deposit polymers such as a PEDOT (Figure 8) in OBDs. Hot plate is used to dissolve the solid material in the solvent as well as evaporate the solvent after the material has been deposited on the substrate via spin coating. ........................................................................................................24
Figure 6. Initial attempts of two separate samples at creating Organic Bistable Device. (a) The 1st scan produces switching behavior. However, subsequent scans do not have any switching behavior. (B) During the 1st scan a device briefly turns on. However, subsequent scans do not exhibit any switching behavior.

Figure 7. TEM image of ZnO QDs. (a) Particles formed from insufficiently dehydrated ZnAc formed larger, aggregated QDs with an average size above 10 nm. (b) Properly dehydrated ZnAc formed smaller diameter QDs of about 2 nm (picture taken the courtesy of NCF at University of Colorado, Boulder, Colorado).

Figure 8. Organic solution-processable Bistable Device. (a) Layered device structure. (b) Organic Bistable Device created on a flexible (PET) substrate.

Figure 9. I-V characteristics of Organic Bistable Device fabricated in Physics Department of DU laboratory. The ratio of $I_{on}$ to $I_{off}$ is about one order of magnitude. (a) linear-linear plot. (b) log-linear plot.

Figure 10. One possible explanation of switching mechanism in Organic Bistable Devices: conductive paths are being formed, by means of metal impurity injection, due to applied voltage, inside the insulating layer that result in decrease of the resistance and the device switches states from OFF to ON. (a) Device in OFF state. (b) Device in ON state.

Figure 11. Another possible explanation of switching mechanism in Organic Bistable Devices. (a) Non-conductive path with non-oxidized PEDOT chains that result in high resistance. (b) Application of sufficiently high voltage potential results in creation of oxidized PEDOT$^+$ chains, resulting in switching from OFF to ON state (low resistance).

Figure 12. A most likely explanation of switching mechanism of Organic Bistable Devices described in this experiment (Figure 9): charge trapping mechanism explained through electrons being embedded in conductive ZnO QDs, is believed to be responsible for the existence of two, ON and OFF, states, with the device architecture shown in Figure 8. (a) Device in OFF state. (b) Device in ON state.

Figure 13. Schematic of an Organic Field Effect Transistor, or OFET, showing the bottom electrodes (Source and Drain) and top electrode (Gate).

Figure 14. Device geometry and operation of OFET. PQT-12 is a hole-transporting semiconducting polymer. Application of negative voltage at the Gate induces a positive-charge buildup at the dielectric surface of the semiconductor. Red line indicates flow/direction of holes (current).
Figure 15. *I-V* characteristics of the top-gate OFET, shown in Figure 14, based on semiconducting polymer PQT-12. The figure depicts a family of Drain currents ($I_d$) measured for different Gate voltages ($V_g$). ........................................41

Figure 16. Comparison of activation functions of a conventional neural network unit (MATLAB’s tansig was modified to decrease the upper and lower limits as well as adjust its slope: $y = 0.376 * (2 ./ (1 + \exp(-1.5*x)) - 1)$) with the synthetic neuron shown in Figure 3 with binary connection weights for output of a neuron (measured at a drain of an OFET) for input between -6V and 6 V. The difference between these two functions, calculated according to Equation 2, was calculated to be approximately 3%. ..................................................43

Figure 17. CrossNet configuration of Organic Bistable Device-based artificial synapse increasing the total number of possible connection weight. ..................45

Figure 18. Synthetic Neural Network architecture with three distinct layers. Input layer has 4 inputs, hidden layer has 4 neurons, and output consists of a single neuron. Only neurons in Hidden layer and Output layer are needed when creating a hardware neural network as neurons in Input layer are only a conceptual representation of inputs (for instance from sensors) to the network.................................47

Figure 19. Nonlinear relationship of the SNN inputs ($A$) and output ($B$) as the network is stepped forward in time (data collected in experiment detailed in Figure 18)........................................................................................................48

Figure 20. Proposed Structured Computational Polymer design. Each cell, neighboring with 6 cells, will be equipped with a polymer micro sensor and actuator as well as distributed cognition (SNN). ........................................51

Figure 21. Setup of the water hammer experiment. Marks indicate points used to represent the shape of the hose; the first four yellow marks (light in grayscale) are used to show the points used for shape extraction by neural network........................................................................................................53

Figure 22. Comparison of outputs (given in radians) of a conventional ANN (blue or dark), trained on MATLAB, and the SNN presented here (red or light), trained to detect a directed shape due to water hammer effect on a hose. Shape 1 was used as a reference point (see Neuron Simulation section for explanation). Horizontal axis (value of $\pi/2$) is used as the reference: values above the axis indicate vector to the left of the reference direction, and values below the axis indicate vector to the right of the reference direction.........54

Figure 23. NOMAD SCOUT2 mobile robot used to obtain the necessary data (inputs from 9 sonars, and outputs from 2 motors) for wall following routine. ........................................................................................................56
Figure 24. Comparison of the ANN (MATLAB) and SNN outputs for a robotic wall following task: the average SNN error was approximately 9.6%. (a) The output of the right wheel, with SNN accuracy of 8.6%. (b) The output of the left wheel, with an average the SNN error of about 10.5%. ................................58

Figure 25. Motivation behind investigation of ANN resilience: given a fixed number of neurons, is it better to arrange them into: (a) single hidden layer or (b) multiple hidden layers. .................................................................62

Figure 26. Various arrangements of hidden layers investigated: all cells in a single hidden layer (100), cells split into two hidden layers (50/50), cells split into three hidden layers (33/34/33), and cells split into four hidden layers (25/25/25/25). .................................................................64

Figure 27. (a) No use of feedback (feedforward) and the use of feedback (recurrent) architectures. The feedback is only employed from the last hidden layer to the first hidden layer. (b) Snapshot from MATLAB simulation illustrating a recurrent network with two hidden layers (50 cells each) with a feedback from the output to the 1st hidden layer. .....................................................66

Figure 28. (a) Plot of \textit{tansig} activation function used for the experiment, demonstrating perpetrating the ‘stuck-at-1’ fault by either fixing the inputs to be equal to ‘10’ (faults to input layer) or all the connection weights of the affected neuron to ‘10’ (faults to hidden layer(s)). (b) With all the connection weights set to ‘10’, the output of a neuron will be ‘1’ for all input values, resulting in the ‘stuck-at-1’ condition. ........................................71

Figure 29. Graphical representation of the damage pattern - the order in which the individual cells were removed. Figures (a) through (i) represent sequential removal of individual cells. ........................................................................73

Figure 30. ‘Stuck-at-0’ faults in hidden layer, uncorrelated data, unoptimized network. (a) recurrent architecture. (b) feedforward architecture: for ‘stuck-at-0’ occurring in hidden layer(s) increasing the number of layers results in increasing the error. ........................................................................76

Figure 31. ‘Stuck-at-0’ in input layer, correlated data, optimized network. (a) recurrent architecture. (b) feedforward architecture: for ‘stuck-at-0’ occurring in input layer increasing the number of layers results in decreasing the error. ........................................................................77

Figure 32. (a) When ‘stuck-at-0’ faults occur in the hidden layer, it is better (network experiences less performance degradation) to arrange all the cells into a single hidden layer. (b) When ‘stuck-at-0’ faults occur in the input layer, it is better to arrange the cells into multiple hidden layers. ..........78
Figure 33. 'Stuck-at-1' fault in NN. (a) With faults occurring in Hidden layer(s), network with the highest number of hidden layers (four layers) experiences the least performance degradation. (b) With faults occurring in input layer, network with the highest number of hidden layers (four layers) experiences the least performance degradation.

Figure 34. ‘Stuck-at-1’ fault is administered exclusively to individual hidden layers: either only to the 1st, only to the 2nd, only to the 3rd, or only to the 4th hidden layer.

Figure 35. Preliminary design of a single, self-contained synthetic neuron (see Figure 3 for neuron architecture) outlining individual manufacturing layers that allows for four \((n+1)\) possible connection weight values (this can be easily increased by increasing the number of Organic Bistable Devices per input). Resistive strips, made from PEDOT, are used for setting individual OBDs to ON/OFF states as well as resistors.
# List of Equations

<table>
<thead>
<tr>
<th>Equation</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equation 1</td>
<td>13</td>
</tr>
<tr>
<td>Equation 2</td>
<td>43</td>
</tr>
<tr>
<td>Equation 3</td>
<td>63</td>
</tr>
<tr>
<td>Equation 4</td>
<td>63</td>
</tr>
</tbody>
</table>
List of Tables

Table 1. A single synthetic neuron, with four OBDs per input (sixteen quantized connection weight values), trained to work as AND and OR logic gates. ..........46

Table 2 Number of neurons used with different network architectures (number of hidden layers used)........................................................................................................69
List of Acronyms

ANN – Artificial Neural Network
BD – Bistable Device
BE – Bottom Electrode
DU – University of Denver
ITO – Indium Tin Oxide
OBD – Organic Bistable Device
OFET – Organic Field Effect Transistor
OLED – Organic Light Emitting Diode
PEDOT – Poly(3,4-ethylendioxythiophene)
PET – Polyethylene Terephthalate
PMMA – Poly(methylmethacrylate)
PQT12 – Poly(3,3’’’ dialkyldialkylquaterthiophene)
SCP – Structured Computational Polymer
SNN – Synthetic Neural Network
TE – Top Electrode
TEM – Transmission Electron Microscopy
ZnO QDs – Zinc Oxide Quantum Dots
Chapter One: Introduction

Motivation

The majority of today’s computers are built based on the basic architecture proposed in 1937 by Alan Turing (Cragon, 2000). They are founded on the idea of universal computation of sequential instructions. This paradigm has proven to be unimaginatively successful with inexpensive computers making their way into just about every aspect of human existence. Transistors, being at the core of this architecture, have shrunk in size by more than six orders of magnitude and their speed has also been increased by over ten million (www.intc.com). However, even though significant progress has been made in relation to the power consumption of an individual transistor, the principle of the paradigm is a deterministic approach, with transistors operating in their saturation regions, which necessitates significant margins of error achieved by operating transistors at high voltages. This in turn results in high power consumption (Fox, 2009; Versace, 2010). IBM’s Blue Gene/P (BG/P) super computer, fit with 147,456 CPUs and 144 TB of main memory, designed to simulate the cortical function of a cat, consumes about 1.3 megawatts of electricity (Jo, 2010). In contrast, the human brain, on the average, consumes only about 20 watts of power. The brain of a small fruit fly uses energy on the order of microwatts, for such complex tasks as flight control and visual information processing to find food and avoid predators. A super computer uses megawatts of energy and still has difficulty executing such tasks in real-time. The
massive parallelism, distributed architecture, and the merger of information and information processing are believed to be responsible for such spectacular efficiency of these biological systems (Fox, 2009).

The term *neuromorphic* was coined by Carver Mead (1990) to describe systems that mimic neuro-biological architectures. *Neuromorphic engineering* is a concept of biologically inspired physics, mathematics, computer science and engineering that aims to develop neural systems with applications such as computer vision, machine learning, and autonomous robots. Living organisms process information in a starkly different fashion than human-made computers. At the core of a biological brain is biological neuron which is responsible for processing, transmitting, and storing information. A single neuron can receive information from as many as 10,000 neurons. It, then, can be connected to as many as few million other neurons. A human brain consists of about 50-100 billion neurons and about 1000 trillion synaptic connections arranged into various areas each responsible for different brain functions, which themselves are being coordinated by yet other brain regions (Atallah, 2004). This massive parallelism and modular design are believed to be behind such spectacular energy efficiency, high-speed of operation as well as generic intelligence and robustness against damage (Versace, 2010).

Artificial Neural Networks, or ANNs, being the earliest example of neuromorphic systems, are mathematical constructs that attempt to capture the essential components and functionalities of networks of such processing elements, constituting the brain. Modeled after biological brains, they consist of a set of interconnected group of artificial neurons
and process information using a connectionist approach to computation. They are most often used to model systems where the relationship is not explicitly known.

The most common neural architecture is feedforward where the information is unidirectional; there is no explicit feedback. The other common neural network type, termed recurrent neural network, employs an explicit feedback in the network. There are two main learning paradigms. Supervised learning involves training data with explicit input and output training sets. Unsupervised learning employs training data with only input patterns and a cost function that the network tries to minimize (Haykin, 1999).

A very common application of ANNs is in control systems where the system is non-linear and the exact relationship is difficult or impossible to determine analytically. ANNs have been employed in applications such as image processing (Danchenko, 2007), and hand-writing recognition (Srihari, 2008) where the system is expected to learn over time, and also in prediction of performance of the stock market (Yugong, 2009), fraud management and loan defaulting in the banking industry (Witkowska, 1999), control and sensor monitoring in industrial plants (Bouhouche, 2006), memory allocation in embedded systems (Chtourou, 2006), and target detection in vision-based systems (Resko, 2004). However, the power consumption of majority of ANN’s is as high as that of other conventional information processing systems due to the fact that they are emulated on serial, electronic Turing machines. Additionally, the real-time performance is often limited due to the same reason.
Thesis Overview

Given the limitation of the serially emulated artificial neural networks, the shortcomings of the currently available hardware neural networks, the potential of biologically inspired neuromorphic architectures to build machines that can learn and reason (www.darpa.mil), and my personal interest in biologically inspired computing, I decided to investigate the possibility of developing a simple parallel information processing system that would be more energy efficient, could be employed for a wide variety of tasks, would be easy to manufacture and would possess a certain kind of information processing pattern akin to the type of reasoning observed in nature.

The principal contributions of this thesis are: development of the single-transistor-single-organic-bistable-device-per-input circuit that approximates the behavior of a single artificial neuron; development of polymer electronic elements necessary for realization of the aforementioned circuit, termed synthetic neuron, as it will be realized using synthetic (polymers) materials; analysis of performance of a network of the synthetic neurons; and, in the face of deployment of polymer-based synthetic neural network, analysis of resilience of various network architectures subjected to damage or other manufacturing error.

This thesis is organized as follows. First I present a review of the currently available hardware neural network implementations followed by a background of the discovery of my own architecture. Next, I will demonstrate my own proposal with the emphasis on simplicity of this design. Chapter Three will discuss the creation, in the University of Denver laboratory, of the individual components needed to realize a single
synthetic neuron proposed in the previous chapter. Following this demonstration I will present simulation of an individual synthetic neuron based on the electrical characteristics outlined in the previous chapter. I will demonstrate that a single synthetic neuron can be trained to function as a logic gate. Chapter Five is dedicated to computational analysis, meaning computer simulation, of employing synthetic neural network to perform real world tasks, namely classification. I will show that SNN can be used to predict directionality of propulsion due to water hammer effect. I will also make evident that SNN can be used to perform, with a great deal of accuracy, a wall following routine commonly associated with mobile robots. In the light of realizing hardware neural network, the possibility of a single or a group of neurons becoming un-operational due to physical damage or manufacturing error, is real. Therefore, in Chapter Six I will present investigation of robustness or resilience of various architectural paradigms. Specifically, I will answer the following question: “given a fixed number of neurons, is it better to arrange them into a single hidden layer or split them into multiple hidden layers.” This thesis will conclude with summing up the main points of this work as well as discussing future directions that will lead to creation of tangible SNN.

**Review of Literature**

A number of possible realizations of hardware neural networks have been proposed over the years. Deshmukh *et al.* (2005) suggested realizing the functionality of a neuron via a conventional CMOS technology. In simulation they constructed a discrete neuron (a neuron based on the operation of a transistor in its saturation region, as opposed to its linear region) with a single NAND, NOR and XOR gates as well as ten flip flops
that were used to store binary connection weights. The neuron was based on a weighter circuit used to perform the connection weight functionality, with a single NAND and XOR gates and two flip flops. Two-phase clocking with no overlap was used to ensure that the weights were shifted without corruption. They demonstrated a network consisting of three layers, each fit with five neurons, capable of performing a task of simple pattern recognition.

Similar to Deshmukh’s design, in the most basic form, my synthetic neuron is also fit with binary connection weights. However, even though networks with binary connection weights are often cited as capable of performing useful tasks (Johansson, 2004; Li, 2010), such a scheme suffers from the limitation of applications that it can be applied for. As such, my neuron design improves on this shortcoming by allowing, with a relative ease, to increase the granularity of connection weights thereby increasing the usefulness of the design.

Gupta et al. (2005) proposed a design aiming at realizing the sigmoidal activation function, as well as its derivative, commonly used with neural networks. This design is based on asymmetry of transistors, operating in their linear regions, connected as cross-coupled differential pairs. The strength of this proposal is that an external signal is used to obtain either the sigmoidal or the derivative of sigmoidal activation function from the same circuit. This design featured fifteen transistors, cross-coupled and operating in their subthreshold regions, as well as four current sources. The authors demonstrated a proof-of-concept neuron capable of generating both tan-sigmoidal (bounded between +/- 10 volts) and log-sigmoidal (bounded between 0 and 20 volts) activation functions as well as
the bell-shaped derivative (with a peak at around 4.5 volts). Additionally, experimental results showing family of outputs with various saturation levels of the sigmoidal shape curve, along with their derivatives were shown. The difference between the positive saturation ($1^{st}$ quadrant) and negative saturation ($3^{rd}$ quadrant) authors determined to be up to 200%.

Similar to Gupta’s proposal, my design can also obtain a family of activation functions that can either approximate either tansigmoidal or logsigmoidal activation functions with various saturation levels. However, the strength of my design is that it is based only on a single transistor as compared to fifteen transistors accompanied by four current sources proposed by Gupta et al. While their approach is based on asymmetry of transistors, my proposal utilizes the symmetry of organic field effect transistors. Additionally, my design is that of a complete neuron, which incorporates not only the sigmoidal activation function but also neuronal connection weights along with summing functionality of a soma.

A number of companies, including Intel, IBM, AMD, Hitachi, Siemens and Samsung, to name just a few, offer commercial chips that, to a various extent (degree of precision, including or excluding learning ability, various network architecture), realize the functionality of neural networks (Dias, 2004). Hitachi, for instance, offers neural chips with either Hopfield or Backpropagation learning functions with up to 574 neurons and 32k synapses. IBM developed a number of chips that employ radial basis function (RBF) architecture and can use up to 36 neurons per chip with a maximum of 64 synapses per neuron. Intel offers chips that do (NI1000, via RCE, or restricted coulomb
energy, and PNN, or probabilistic neural network, algorithms) and do not (ETANN, or Electrically Trainable Analog Neural Network) possess learning capabilities. A chip with no learning paradigm needs to have its weights imported after an external ANN has been trained (for instance using software such as MATLAB’s ANN Toolbox).

At present my design does not incorporate any learning algorithms. Pre-computed connection weights, via external software training such as the aforementioned MATLAB ANN Tool Box, have to be manually imported. Additionally, as already mentioned, there is a limit to the precision of both the activation function and the connection weights. The accuracy of activation function is explicitly related to the physical characteristics of a transistor used. The precision of connection weights, however, can be arbitrarily extended (for more information see section Bistable Device as a Synapse for Connection Weight).

The simplest designs approximating behavior of a neuron were proposed and demonstrated by two separate groups working under Alibart and Jo. Both of them concentrated on the behavior of a spiking synapse, which is more similar to the behavior to a biological synapse than that of an artificial synapse found in ANNs.

A device made of molecules and nanoparticles, termed nanoparticle organic memory field-effect transistor, or NOMFET (Alibart, 2010), was shown to exhibit a facilitating (STP, or short-term potentiation) and depressing (STD, or short-term depression) behavior of a biological spiking synapse. The same device was shown to exhibit the aforementioned functions commonly found in biological brains. By applying positive or negative Gate voltage, while holding Source and Drain contacts grounded, facilitating or depressing behavior can be programmed respectively. This device makes
use of the charge storage capability of nanoparticles embedded in an insulating material that is sandwiched between source and drain of a transistor. Applying voltage repeatedly results in steadily increasing or decreasing voltage response of the NOMFET due to individual charges being continuously and cumulatively retained by gold nanoparticles. The transconductance gain of a transistor is used to dynamically tune the amount of charge stored by the nanoparticles. That, in effect, makes the device mimic the short-term plasticity of a biological synapse.

Jo et al. (2010) have suggested the possibility of employing a silicon-based memristor to realize the time-dependent characteristics of a biological spiking synapse. This design utilizes metal-oxide semiconductor neurons and silicon-based memristors as synapses for synaptic functions such as spike, timing dependent plasticity. In this design, memristors are arranged into a cross-bar (similar to that proposed by Likharev, discussed later in the text) with pre- and post-synaptic neurons being located on adjacent sides of the cross-bar. This allows for every post-synaptic neuron to receive information from every pre-synaptic neuron. The use of memristive synapses results in neural behavior akin to spiking, biological neurons.

With the aforementioned Alibart and Jo proposals in mind, the strength of a design that exhibits spiking behavior is that it does not require an external learning circuit. The value (charge) of a synapse is directly related to the time-dependent input signal; the synapse learns continuously. In biology this concept is known as a synaptic plasticity. The strength, however, is also the weakness of this approach. Lack of stimuli results in synapse de-learning or forgetting information stored. The design that I propose
is based not on the memristive synapse operating in its linear region but rather on the saturation property of Organic Bistable Devices (discussed later in text). Once synaptic weights are set, during the learning stage, the synapse will retain this information indefinitely (the limitation being physical degradation of the device). This property makes it much more analogous to conventional ANNs and easier to employ in standard engineering applications.

**Background**

Likharev (2003) proposed the idea of realizing a nanoscale hybrid semiconductor/nanodevice integrated circuit with a neuromorphic network architecture. His hybrid CMOS/nanowire/molecular-nanodevice (CMOL) would be created from CMOS op-amps acting as somas, and synapses from two-terminal, bistable memristive nanodevices formed at intersections of nano-scale axonic and dendritic wires (Jo’s cross-bar arrangement of memristive synapses, though un-credited, is identical to the one proposed by Likharev). The arrangement of these devices would produce a device with a behavior similar to that of a neural network. He demonstrated, though only theoretically, a possibility for this system to perform functions such as image recognition and pattern classification (Türel, 2004; Ma, 2007).

Inspired by Likharev’s work I proposed an analogous architecture that uses an Organic Bistable Device to realize the multiplicative property of a synapse (Nawrocki, 2010b; Nawrocki, 2011b). Additionally, in contrast to Likharev’s architecture, which is significantly more complex as it is based on the use of operational amplifiers, I will

---

1 Both OBDs and memristors will eventually discharge. However, at present the retention time of OBDs is reported in literature to be in months or greater.
demonstrate a greatly simplified architecture where the summing functionality and sigmoidal output of a soma can be realized by employing only a single transistor operating in its linear region. The architecture operates in a time domain, as opposed to the frequency domain of biological neural networks, meaning it is a subject to the same operating principles that govern conventional Artificial Neural Networks that have been analyzed and applied in the last few decades. Additionally, the use of Organic Bistable Devices in lieu of memristive synapses, results in time invariant network: the system retains the information even with the absence of external power source.
Chapter Two: Architecture

In a biological neuron the synapse is responsible for converting the chemical or electrical input signal and passing it onto the soma, or neuron body, for further processing (Kandel, 2000). Its purpose is to provide a neuron with an input that is proportional to the importance of the signal from that specific neuron: An input from an “important” neuron is passed much more easily (the synapse is more sensitive to the signal, meaning that a synapse will pass a signal of a minimal strength) to the neuron body than an input from an “un-important” neuron (the synapse is less sensitive to the signal, meaning that a signal will have to be very strong in order for the synapse to respond to it). Learning (neuro- or synaptic plasticity) is accomplished by modifying, either increasing or decreasing, the strength (sensitivity) of the synapse.

In an artificial neuron the functionality of a synapse is realized via a connection weight. Analogously to the biological synapse, connection weight from an “important” neuron is significantly greater than a connection weight from an “un-important” neuron. Learning in an ANN is accomplished by modifying the numerical value of the aforementioned connection weight.

The function of a soma, in both biological and artificial neurons, is to sum all the input signals and produce an output signal when a predefined threshold value has been exceeded by all these summed inputs. Whereas biological neurons encode the
information by modifying the frequency of firing, in artificial neural networks it is the amplitude of the signal that changes.

Equation 1 represents the algorithm commonly used to compute the output of an artificial neuron. In general, it performs three functions: it multiplies all the inputs \( (x_i) \) by their corresponding connection weights \( (w_i) \), it sums all of these products \( \sum(x_iw_i) \), and it produces the output based upon the utilized activation function (Haykin, 2000), denoted by \( \varphi \) in Equation 1. Conceptually, the multiplication is accomplished by the artificial synapse, while the summation and activation are performed by the artificial soma. Hence every representation of a neuron must be able to perform those three functions, namely multiplication, summation, and activation. More generally, many neural representations also include bias value commonly added to the product of input values and connection weights. However, this value, for the purposes of simplicity, can be treated as equal to zero (0). Figure 1B graphically demonstrates Equation 1.

\[
f(x) = \varphi\left(\sum_i (x_i w_i)\right)
\]

Equation 1
Figure 1. (a) Biological neuron with dendrites and axon (input and output “wires”), synapse and soma being outlined (image from Wikipedia.org). (b) Graphical representation of operation of a single artificial neural, also shown in Equation 1: Dendrites and Axon act as input and output pathways, while artificial synapse (connection weight) propagates input signal proportional to the importance of that particular input. Soma produces proportional output (activation function) that is the sum of all of the input signals.
Bistable Device as a Synapse for Connection Weight

In the most basic form a Bistable Device, or BD, can be thought of as a memory device represented by two resistors; it can be in one of two possible states, with its resistance corresponding to being either in the ON (low resistance) or OFF (high resistance) state. In general it is desired for a Bistable Device to have a high ratio of ON (R\text{ON}) and OFF (R\text{OFF}) resistances (typically several orders of magnitude). The operation of an ideal BD, demonstrated in Figure 2, can be summarized as follows. It is in the OFF state (high resistance – step 1 in Figure 2) until the input voltage is increased past a threshold voltage (V\text{ON} – step 2 in Figure 2). It stays in the ON state (low resistance – step 3 in Figure 2) until the voltage is reduced below a threshold voltage (V\text{OFF} – step 5 in Figure 2). Steps 2 and 4 correspond to ON and OFF switching respectively. Keeping operating (read) voltage (V\text{read}) below threshold voltages (absolute values of V\text{ON} or V\text{OFF}) allows for a possibility of two distinct outputs (output can be set by utilizing V\text{write} > V\text{ON} or reset by V\text{erase} < V\text{OFF}). In theory V\text{OFF} = -V\text{ON}. With the output of a Bistable Device (current) equal to a product of input (voltage) and connection weight (conductance), with its bistable nature, it seems a natural choice to realize the functionality of a binary synapse (Choi, 2009). For more in-depth explanation of the operation, design, and switching properties of Bistable Devices see sections Organic Bistable Device and Switching Mechanism in the next chapter.
A Bistable Device (BD) is either in the OFF state (R_{off} – high resistance, step 1 and 5, ‘red’ or light in grayscale) or ON state (R_{on} – low resistance, step 3, ‘blue’ or dark in grayscale) depending on past voltage. ‘Green’ represents switching, to either ON (step 2) or OFF (step 4) state. The element can be turned ON by increasing the input voltage above the positive threshold value (V_{ON}). Analogously, it can be turned OFF by decreasing the input voltage below the negative threshold value (V_{OFF}).

**Single Transistor Circuit as a Soma for Summation and Activation**

In both biological neurons and artificial neurons, the soma performs two functions. First, it sums all of the dendritic inputs. It then fires, along the axon, producing a bounded output proportional to the input; there are maximum and minimum values (saturation) that the signal will never surpass. This is a critical property as it ensures that for an unbounded input there is a bounded output.
I have developed a simple, single-transistor circuit that produces a suitable approximation to a neural soma. Given the weighted inputs described above, the single-transistor circuit sums the inputs and produces a nonlinearly scaled output that approximates a sigmoid. To emulate sigmoidal behavior, I used a single characteristic Source-Drain voltage, $V_{SD}$ curve by choosing Gate voltage, $V_G$ based on utilized $R_{ON}$ and $R_{OFF}$ resistances (discussed in Organic Bistable Device).

Figure 3A and Figure 3B demonstrate design of a single synthetic neuron. Figure 3A shows the actual design of a synthetic neuron incorporating BDs as artificial synapses. In the subthreshold region (or ‘operating region’ in Figure 2), below $V_{ON}$ or above $V_{OFF}$, a BD operates as a resistor and for the purposes of explanation and simplicity of simulation can be approximated as such. Figure 3B illustrates such an approximation that was used for simulation discussed later in text (see Chapter Four: Single Neuron Simulation and Chapter Five: Simulation of a Network of Synthetic Neurons). This design additionally allows for different neural behaviors (for different types of neurons) to be achieved by employing different gate voltages. For an extended range of output values, $V_G$ could be made variable. Furthermore, positive and negative bias values can be easily realized by either connecting (in Figure 3A and Figure 3B) a variable voltage supply (for varying bias values) or adding another input voltage to the voltage divider formed between BDs and $R_{base}$.
Figure 3. (a) Schematics of a single neuron with two inputs (a binary connection weight, formed by a single BD), denoted by $V_{IN1}$ and $V_{IN2}$, and one output, marked $V_{OUT}$. An organic field-effect transistor (OFET) is used for summation of input currents and to provide activation function. A symbol of a memristor in lieu of BD is used as such a device can be thought of as a memristor operating in its saturation region. (b) The same design showing, for the purposes of clarity, BD represented by two resistors: $R_{ON}$ and $R_{OFF}$.

It should be mentioned that my design is based on a BD which is essentially a memristor operating in saturated regions, except for when switching states. Such a choice
results in behavior approximating a conventional Artificial Neural Network where, once the training has finished, the network retains its values and does not need to be refreshed. Employing a memristor operating in a linear region, such as the one described by Alibart (2010), with the same neural architecture, would result in a time-varying, frequency-based behavior much more similar to a behavior of a biological neuron, such as the one described by Jo (2010). A great benefit of such a design stems from the fact that the system would not need an additional circuit for the purposes of training, as the synapses (memristors) would be trained during the operation (similar to how a biological brain functions). However, as already mentioned, use of a network of such neurons would require the need for a periodic refreshing of the information stored in the artificial synapses, or connection weights, resulting in the network “forgetting” all the information learned if left unused.

Another consideration should be given to the fact that the use of BDs as artificial synapses results in non-negative connection weights. In ANNs, connection weights are customarily initialized in the +/-1 interval. A non-negative restriction on connection weights results in increasing the time needed to train the network. In some situations a network may not be able to train at all. However, this might be at least partially alleviated by including a negative bias, realized by the use a negative voltage source and a resistor connected via a voltage divider. Another solution to circumvent this limitation would be to include inverting, or negative neurons (neurons that produce negative output for positive input), accomplished by including two (N-type and P-type) transistors connected to form a canonical inverter.
Chapter Three: Individual Components

The long term goal, as discussed later in the text (see Structured Computational Polymers) is a creation of a tangible Synthetic Neural Network. At present, BDs and organic field effect transistors are relatively new elements, and very few analysis tools are available. Because of the constraint of physical flexibility (see Structured Computational Polymers), and the desire for a low-cost, easily manufactured technology, the choice of the technology was organic electronics. It was desirable, then, to create both of these devices in order to obtain necessary electrical characteristics needed for the simulation.

Organic Electronics

Organic electronics is a branch of electronics that is based on semiconducting and conducting polymers and small molecules. The term “organic” stems from the fact that the materials are carbon-based and rely on the transport of pi-electrons as their conduction mechanism. Elements such as organic light emitting diodes (Park, 2009), organic field effect transistors (Usta, 2009), organic photovoltaic cells (Shaheen, 2005), and even organic electrochemical sensors (Svensson, 2008) have been created. Organic electronics possess several advantages over traditional, inorganic electronics. One such advantage is physical flexibility; whereas traditional electronics are typically rigid and
inflexible, organic electronics allows for creation of non-rigid, flexible devices. This property is due to the fact that polymeric and other organic materials are held together by relatively weak intermolecular bonds and bending does not critically affect their mechanical integrity or charge transport properties (Ouyang, 2005). Figure 4 shows the flexibility of an organic light emitting diode (OLED) created in our laboratory. The first commercial product with a flexible display was demonstrated by Seiko in 2009 (www.seikowatches.com).

![Figure 4. Physically flexible organic light emitting diode (OLED) shown (a) being flexed and (b) lit-up. In this case the device activates (a dim light can be seen) at around 6 V. However, only around 12 V can light become clearly visible. More advanced and commercially available OLEDs have much lower drive voltages.](image)

Another major advantage of organic electronics is the ease and low cost of manufacturing. Solution processing, involving such processes as spin or spray coating, doctor blading, laminating, screen printing or inkjet printing is an attractive alternative to expensive inorganic processes, such as chemical etching, e-beam evaporation or

---

2 This is not always the case. A number of research groups, such as (Hu, 2011), have produced physically flexible inorganic electronic components.
photolithography, that require expensive equipment and clean room facilities. Figure 5 illustrates two examples from our laboratory of solution processing equipment; spin coating (A) and spray coating (B). Large-scale integrated circuits have been fabricated using organic materials and the aforementioned solution processes (Drury, 1998; Pinto, 2006)
Figure 5. (a) Spin Coater, an inexpensive machine, which relies on centrifugal force, used to apply a uniform film of a material with the thickness of tens of nanometers with the accuracy of a few nanometers. (b) Spray coating gun used to deposit polymers such as a PEDOT (Figure 8) in OBDs. Hot plate is used to dissolve the solid material in the solvent as well as evaporate the solvent after the material has been deposited on the substrate via spin coating.

Organic Bistable Device

The earliest (known to the author) publications of electrical switching behavior in thin-film devices were reported about three decades ago (Potember, 1982). Organic Bistable Devices, however, did not appear until the early 2000’s (Ma, 2002). In the next few years these devices acquired a necessary momentum to propel the research into a widespread acceptance (Kanwal, 2005; Ouyang, 2005).
Fabrication

Inspired by the relative ease of fabrication due to the materials used (only a single layer of organic material), as well as outstanding device characteristics (over four orders of magnitude difference between the ON and OFF states) I decided to replicate the work of (Son, 2009b) to obtain an OBD for use as a synthetic synapse. However, this work turned out to be challenging, as the authors omitted important information about the device fabrication recipe, such as the solution concentration or the thickness of individual layers.

The device that I was attempting to replicate consisted of only three layers. The bottom electrode (BE) was formed from Indium-Tin-Oxide, or ITO. On top of the BE was a thin layer of a composite material, poly(methylmethacrylate) (PMMA) mixed with an unknown concentration of ZnO QD. The top electrode (TE) was formed by thermally evaporated layer of Al to a thickness of about 150 nm.

Initial attempts to replicate this work proved dubious and inconclusive. The devices undoubtedly showed hysteretic behavior. However, the ON to OFF ratio was well below a single order of magnitude. Additionally, this behavior was unreliable as multiple scans would result in different behavior. Figure 6A and Figure 6B illustrate the lack of necessary switching behavior. Possible explanation of this behavior includes manufacturing errors, such as incomplete wiping of contacts or contact misalignment, and recipe errors, such as too high annealing temperature, large aggregates of ZnO NP due to presence of moisture in ZnAc crystals, or individual material layers being too thin or too thick.
Figure 6. Initial attempts of two separate samples at creating Organic Bistable Device. (a) The 1st scan produces switching behavior. However, subsequent scans do not have any switching behavior. (B) During the 1st scan a device briefly turns on. However, subsequent scans do not exhibit any switching behavior.

As a result, the basic device structure, meaning the individual layered architecture, was altered. Figure 8A demonstrates the final device structure used to obtain
the $I$-$V$ characteristics demonstrated in Figure 9. The layered device operates in the vertical direction: ZnO being an electron donor (and a hole blocking layer) results in the current propagation from silver top electrode down to ITO bottom electrode. PEDOT:PSS, or Poly(3,4-ethylenedioxythiophene) poly(styrenesulfonate), aids aluminum in donating positively charged holes; silver has a higher work function, -5.1 eV, than ITO, -4.8 eV, and is a poor hole conductor, hence the use of PEDOT, which has a work function of -4.9 eV, closer to ITO, and is much better hole-transporting medium (Huang, 2005). ITO, or Indium-Tin Oxide, a highly-conductive transparent crystal, was patterned to form a cross-bar with an organic the top electrode, forming a device of about 9 mm$^2$.

**Recipe**

The following is a recipe used for creation of the OBD. A particular acknowledgement for this recipe should be given to Brian Bailey for his help in the recipe of ZnO layer. A substrate coated with ITO was first patterned by covering portions of ITO with paint and rubbing the exposed ITO off with Zn dust dissolved in HCl. Subsequently, ITO was cleaned by sonicating the samples in Acetone and IPA respectively for 15 minutes each.

Zinc acetate (ZnAc) was first dehydrated on a hot plate at 120°C for 10 minutes and placed in a desiccator to cool down. A solution was formed by mixing 137.6mg of ZnAc with 45µL of monoethanolamine per 1 mL of 2-methoxyethanol. This solution was then dissolved at 60°C overnight. Before depositing on substrate, the solution was filtered (pore size of 450 nm). The solution was spin coated at 2000 rpm for 60 seconds, forming
a film about 180 nm thick. Before the samples were annealed, the ZnAc was removed from the contacts with IPA. As a side note, it should be stated that annealing not only evaporates the solvents but also has an effect on the overall charge transfer of the material (Akande, 2008). Samples were annealed in air, for 10 minutes at 300°C, to form ZnO. Subsequently the samples were rinsed with DI water, acetone, ethanol and IPA, to remove any organic compounds. The samples were dried at 200°C for 10 minutes in air.

Next, a recipe to form ZnO QD is given. ZnAc was first dehydrated on a hot plate at 120°C for 10 minutes and placed in a desiccator to cool down. A 1% solution of ZnAc was formed by dissolving it in dimethylformamide (DMF). The solution was stirred on a hot plate at 20°C for 10 minutes, before it was raised to 105°C with a rate of 2°C/min. At this point, the solution was left at 120°C for 90 minutes to form quantum dots. Subsequently the solution was allowed to return to room temperature before it was stored in a refrigerator. The size of ZnO QD were verified using TEM to be about 2 nm in diameter. Figure 7 displays an image of ZnO QD with individual lattices being visible.

It should be noted that the dehydration of ZnAc is critical before being dissolved in a solvent for the formation of particles of sufficiently small size. Omission of this step resulted in formation of particles with an average size of about 20-30 nm in diameter. Also, increasing the time the solution is kept on hot place at 105°C past 2 hours commonly resulted in particle aggregation, forming large blocks about 50 nm in size.
A 5% solution of poly(methylmethacrylate) (PMMA) was obtained by dissolving the substance in DMF, at 60°C overnight. Subsequently, a 1.5% solution of ZnO QD in PMMA was formed. The PMMA:ZnO solution was spin coated, at 2000 rpm, forming a uniform film with a thickness of about 200 nm. After the contacts were wiped off with DMF, the samples were annealed at 120°C for 10 minutes under nitrogen.

A solution of PEDOT:PSS, mixed 1:1 with IPA (to increase its wetting properties) was first filtered (pore size of 450 nm) and then deposited by spin coating at 4000 rpm for 60 seconds forming a 30 nm thick layer. Subsequently the samples had their
contacts wiped off with IPA before they were annealed for 10 minutes at 120°C under nitrogen.

The top electrode, or TE, was formed by thermally depositing a 200 nm thick layer of silver. The pressure inside the chamber was commonly below $1.5 \times 10^{-6}$ Torr when the deposition started, and reaching $8 \times 10^{-7}$ Torr towards the end of the deposition. The rate was kept constant at $5 \text{Å/s}$.
Figure 8. Organic solution-processable Bistable Device. (a) Layered device structure. (b) Organic Bistable Device created on a flexible (PET) substrate.

Figure 9 demonstrates the I-V characteristics of the OBD. It can be seen that, in the initial scan the device is turned OFF with resistance being high ($R_{\text{off}} \rightarrow I_{\text{off}}$). However,
at around $V=0.5$ V the device turns ON (positive switching) and the resistance switches from high to low ($R_{on} \rightarrow I_{on}$). When the voltage is decreased the device remains in the ON state until the voltage is reduced below negative threshold voltage of about $-0.5$ V. This device possesses the necessary switching characteristics of an OBD. However, the ratio of $I_{on}$ to $I_{off}$ is only about one and a half orders of magnitude. For the purposes of demonstrating the basic functionality this is satisfactory. However, for all practical implementations, this needs to be improved to at least two or three orders of magnitude in order to avoid erroneous readings due to noise or device degradation.
Switching Mechanism

There are multiple physical mechanisms by which an OBD can operate (store a given state and switch between states). A given device may rely on one or more such mechanisms, and it can occur in the literature that a device is reported without full understanding of which mechanism(s) are responsible. Three of the most common explanations of mechanisms are discussed next.

Electric Field Induction

The first explanation is called electric field induction (Ouyang, 2005). In a metal-insulator-metal (MIM) sandwich, the device is initially in the OFF state (high resistance). However, when the potential exceeds threshold, small amounts of impurities or hot
electrons are injected into the polymer insulator sandwiched between two metal electrodes. This in effect decreases the physical separation of the two electrodes reducing the total resistance (changing the state from OFF to ON). The effect is illustrated in Figure 10. When a reverse potential is applied these impurities, or electrons, are “pushed out” of the insulator back to the metal contacts thereby increasing the total resistance across the polymer (corresponding to switching from ON to OFF state). This explanation often applies to insulators with impurities, such as gold nanoparticles. The low- and high-conductive states are explained by direct tunneling and Fowler-Nordheim tunneling respectively (Wang, 2003).

![Figure 10. One possible explanation of switching mechanism in Organic Bistable Devices: conductive paths are being formed, by means of metal impurity injection, due to applied voltage, inside the insulating layer that result in decrease of the resistance and the device switches states from OFF to ON. (a) Device in OFF state. (b) Device in ON state.](image)

**Formation of Conductive Filaments or Redox Behavior**

Alternative explanation was proposed that is based on redox behavior (Ha, 2010). When a voltage is applied to the electrode, such as PEDOT (a highly conductive polymer), the polymer chains will be oxidized to PEDOT$^+$ chains by the injected carriers.
The current paths will, as a result, form by the PEDOT$^+$ chains, and the device will switch from the OFF to the ON state (high resistance to low resistance). The PEDOT$^+$ chains are reduced to PEDOT$^0$ chains by injection of carriers when a voltage is scanned in the opposite direction. The current paths, then, are destroyed, and the device switches from ON to OFF state. This switching behavior is graphically illustrated in Figure 11.

![Figure 11](image)

**Figure 11.** Another possible explanation of switching mechanism in Organic Bistable Devices. (a) Non-conductive path with non-oxidized PEDOT chains that result in high resistance. (b) Application of sufficiently high voltage potential results in creation of oxidized PEDOT$^+$ chains, resulting in switching from OFF to ON state (low resistance).

**Charge Trapping**

The Electric Field Induction and Formation or Conductive Filaments are two possible explanation of the switching behavior. A third hypothesis, the one the most pertinent to the switching behavior observed in the devices that I created, is called Charge Trapping (Son, 2009a; Son, 2009b). The insulating PMMA is embedded with conductive ZnO QDs. At first, when the device is created and scanned, the resistance is high. However, when the voltage potential exceeds a critical limit, individual electrons become
trapped in ZnO QDs. Subsequently, the resistance between top and bottom electrodes reduces significantly, with the embedded electrons forming a conductive path, and allows for the current to flow freely and the device in considered being in the ON state. This results from the carrier concentration dependence of the mobility in organic semiconductors, which is well established. A higher concentration of carriers increases the carrier mobility, and hence the conductivity, by filling trap states that would otherwise impede carrier transport and also by increasing the dielectric constant that changes the activation energy for carrier transport. Reversing the voltage polarity has the effect of pulling the electrons out of the ZnO QDs and turns the device back to the OFF state. It is the difference in work functions of elements surrounding the active layer (PMMA:ZnO QD) that allows for the electrons to be embedded and form these two, ON and OFF states.

Figure 12. A most likely explanation of switching mechanism of Organic Bistable Devices described in this experiment (Figure 9): charge trapping mechanism explained through electrons being embedded in conductive ZnO QDs, is believed to be responsible for the existence of two, ON and OFF, states, with the device architecture shown in Figure 8. (a) Device in OFF state. (b) Device in ON state.
Organic Field Effect Transistor

Organic Field Effect Transistors, or OFETs, are a type of a field effect transistor that uses an organic semiconductor in its active channel. Primarily these devices are fabricated using layered approach; individual materials, such as electrodes, semiconductor, insulator, are deposited sequentially. The more common device geometry is of bottom-gate with top drain and source electrodes. However, depending on the materials used, the top-gate with bottom drain and source electrodes might be a preferred choice. Figure 13 illustrates the design of an OFET created in laboratory at DU. Because the current flows between two adjacent electrodes, the bottom electrode is often patterned to form what appears to be two forks intertwined with one another. This increases the overall area of the field effect resulting in increased source-drain current ($I_d$).

![Figure 13. Schematic of an Organic Field Effect Transistor, or OFET, showing the bottom electrodes (Source and Drain) and top electrode (Gate).](image)

Because of the availability of a flexible substrate (PET) coated with a conductive sheet ($\text{In}_2\text{O}_3, \text{Ag}, \text{Au}$) I was interested in alternative means of patterning. I decided to
investigate the possibility of employing laser engraver (Versa Laser VLS 3.50; www.versalaser.com) for this patterning task. Additionally, I investigated the silk screen printing process for the purposes of depositing the top electrode (Gate). However, because this process requires physically pressing the metallic pain through the screen, it would result in puncturing through the very thin layer (few hundred nanometers) of the dielectric. Subsequently this approach was abandoned in favor of thermally evaporating the metal electrode. What follows was the result of work performed by Rachelle Cobb.

The bottom electrode was laser-patterned on flexible substrate (PET, or polyethylene terephthalate) coated with In₂O₃, Ag, Au, which is a transparent, conductive material, with a sheet resistance of 25 Ω/□, that is often used for creation of organic photovoltaic cells (OPVs) or electronic displays. A 60 nm thick layer of hole-transporting
semiconducting polymer PQT12, or Poly(3,3‴ dialkyloquaterthiophene) was spin coated on top of the bottom source and drain electrodes. A layer of insulating PMMA, or Poly(methyl methacrylate), was spin coated on top of the semiconductor, with the average thickness of about 100 nanometers. A 120 nm thick layer of aluminum was thermally deposited to function as the top gate.

Figure 14 illustrates the device geometry of the OFET, with the aforementioned layers, as well as pictorially presents the operation of the device. When a negative voltage is applied to the Gate (top electrode) it induces a negative charge-buildup on the gate side of the dielectric and the positive change-buildup on opposite side as well as inside the semiconductor. As a result, the normally non-conductive semiconductor becomes conductive allowing for the transport of holes. Figure 15 demonstrates a family of drain currents \( I_d \) with different gate voltages \( V_g \).

It should be noted that, because Source and Drain electrodes are made of the same material (actually, as already stated, before the patterning those two electrodes formed one block of material) their assignment of Source and Drain is purely arbitrary. Also, the physical layout (the aforementioned intertwined forks’ pattern) is symmetrical. As a result, the \( I-V \) characteristic of an OFET is also symmetrical resulting in great similarity to the sigmoidal function.
Figure 15. I-V characteristics of the top-gate OFET, shown in Figure 14, based on semiconducting polymer PQT-12. The figure depicts a family of Drain currents ($I_d$) measured for different Gate voltages ($V_g$).
Chapter Four: Single Neuron Simulation

The simulation and analysis presented in this and the following chapters are based on the electrical characteristics of the elements, OBDs and organic field effect transistors presented in Chapter Three: Individual Components. Additionally, in order to simplify the analysis, the actual model used was the one seen in Figure 3B, that is two resistors were substituted for a single OBD (for justification please see Bistable Device as a Synapse for Connection Weight).

A Single Synthetic Neuron Behavior

As can be seen in Figure 3, for a binary connection weight, a single neuron consists of a single OBD per input (Figure 3A represents the actual circuit while, for clarity purposes, Figure 3B represent two distinct states of a OBD, ON and OFF, with two resistors – see Bistable Device as a Synapse for Connection Weight section for justification), a single transistor, and two resistors. A single OBD, being in either ON or OFF state, corresponding to either low ($R_{ON}$) or high ($R_{OFF}$) resistance, is used as a binary connection weight. With the aforementioned constraint on input voltage, in a single neuron only the transistor is an active element with the rest being passive elements. Therefore Figure 16, depicting output voltage plotted against input voltage, reveals that the activation function of the synthetic neuron looks remarkably similar to a sigmoidal activation function. It demonstrates that only a single transistor, with one OBD per input,
is sufficient to obtain a sigmoidal activation function commonly associated with analog artificial neural networks. Equation 2 shows the formula used to calculate the error:

\[
err = \left| \frac{|p| - |q|}{|p|} \right|
\]

Equation 2

where \( p \) denotes ANN value calculated in MATLAB and \( q \) is the value obtained from the SNN simulation.

Figure 16. Comparison of activation functions of a conventional neural network unit (MATLAB’s tansig was modified to decrease the upper and lower limits as well as adjust its slope: \( Y = 0.376 \times (2 \div (1 + \exp(-1.5 \times x)) - 1) \)) with the synthetic neuron shown in Figure 3 with binary connection weights for output of a neuron (measured at a drain of an OFET) for input between -6 V and 6 V. The difference between these two functions, calculated according to Equation 2, was calculated to be approximately 3%.
As already noted that the symmetrical behavior of the activation function is realized because OFETs have a truly symmetrical design (source and drain are made of the same material and their assignment is purely arbitrary) which results in a remarkably symmetrical I-V characteristics (see Figure 15).

**Single Synthetic Neuron as a Logic Gate**

For some tasks binary connection weights are sufficient for proper operation (Likharev, 2003). However, other tasks may require a finer granularity of the connection weights. Because OBD are formed at a cross-point of two wires, their physical size can be minimal. Therefore, increasing the smoothness of connection weights can be easily accomplished by increasing the total number of OBD assigned to an individual synapse (at present, the theoretical limit for a single OBD created in our laboratory is 1μm²). Three such devices, with only a single one being in an ON state, would result in a synapse with 4 possible values. However, setting a subset of OBDs to an ON state would result in $2^n$ possible values, with $n$ being the number of OBDs employed per connection weight. Making OBD with variable ON and OFF resistances (accomplished by varying their size) would result in more linear distribution of quantized connection weight values. Figure 17 illustrates one possible physical arrangement of OBD-based artificial synapses used to increase the total number of possible connection weights.
From the pattern recognition or classification perspective, a single artificial neuron can be used to perform a binary or two-dimensional (two-class) classification (Haykin, 1999). Table 1 demonstrates an example where a single synthetic neuron, with four binary devices per input (sixteen quantized connection weight values), was trained to behave like a logic gate. Two logic gates, AND and OR, were obtained. The ability of a single neuron to perform a binary or two-dimensional classification can be easily
extended to a multi-dimensional classification capability of a network of neurons (which will be demonstrated in *Chapter Five: Simulation of a Network of Synthetic Neurons*).

Table 1. A single synthetic neuron, with four OBDs per input (sixteen quantized connection weight values), trained to work as AND and OR logic gates.

<table>
<thead>
<tr>
<th></th>
<th>AND Gate</th>
<th>OR Gate</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>IN_1 [V]</td>
<td>IN_2 [V]</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0 (OFF)</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0.3 (OFF)</td>
</tr>
<tr>
<td>0</td>
<td>5</td>
<td>0.3 (OFF)</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>2.2 (ON)</td>
</tr>
</tbody>
</table>

**SNN Input-Output Relationship**

To investigate the input-output relationship of the SNN, I set up an experiment with five synthetic neurons (four hidden neuron and one output neuron, input neurons are only a conceptual representation of network inputs) connected in a network analogous to an artificial neural network. Figure 18 depicts the network architecture. The actual network architecture was deliberately chosen to be identical to the architecture used to test the feasibility of emulating an artificial neural network in hardware (see section *WH on SNN* for more details). For the purposes of clarity of analysis only input values IN₁ and IN₂ were changed linearly while IN₃ and IN₄ were kept constant.
Figure 18. Synthetic Neural Network architecture with three distinct layers. Input layer has 4 inputs, hidden layer has 4 neurons, and output consists of a single neuron. Only neurons in Hidden layer and Output layer are needed when creating a hardware neural network as neurons in Input layer are only a conceptual representation of inputs (for instance from sensors) to the network.

Figure 19 was constructed with data pertaining to an experiment discussed in *Chapter Five: Simulation of a Network of Synthetic Neurons*, which relates information obtained from bend sensors (input to the network) to force directionality (output of the network) of water hammer effect. Figure 19A demonstrates the linearly varying input values. Figure 19B shows the output of the SNN. It can be clearly seen that the input-output relationship is nonlinear. This is due to the nonlinear activation function (seen in Figure 16) most profoundly manifested at the extreme values of the input (saturation region of the output).
Figure 19. Nonlinear relationship of the SNN inputs (A) and output (B) as the network is stepped forward in time (data collected in experiment detailed in Figure 18).
Chapter Five: Simulation of a Network of Synthetic Neurons

I have already demonstrated the similarity of a single synthetic neuron to that of an artificial neuron. I have not, however, yet shown the feasibility of employing the network of my synthetic neurons to a real world task. ANNs are often employed for classification tasks where there network learns to find a pattern. I have already demonstrated that my single synthetic neuron can be trained to perform a binary classification, or to act like a logic gate (see Single Synthetic Neuron as a Logic Gate). However, I still have not demonstrated the ability of a network of synthetic neurons to perform a task that involves much more complex classification of non-linear relationships. Next, using an example of a proposed all-polymer soft robot that aims to employ my proposed distributed cognition with distributed form of robotic actuation, I will demonstrate that my Synthetic Neural Network is indeed fully suitable for a complex, non-linear, real-world task (Nawrocki, 2011a).

Structured Computational Polymers

As already mentioned neuromorphic architecture aims to mimic neuro-biological architectures present in the nervous systems. This design can be employed with various systems. However, it seems the most appropriate to be utilized with novel concepts, such as biomorphic or soft robotics. Soft robots are a new group of mechanisms where their structure is not rigid but can adjust to the changing environment (Albu, 2008). Commonly these systems are made from flexible materials, such as polymers.
Structured Computational Polymers (SCPs) is a concept of layered class of smart materials that combine perception, cognition, actuation, power, and structure. These highly integrated smart materials will be used as building blocks for new type of devices. Because the SCPs will be highly distributed, they will be fabricated using a concept called shape-deposition manufacturing (Cham, 2002; Kim, 2007) that allows for iterative combination of dissimilar material addition and removal with the use of sacrificial support materials. This process permits creation of distributed, self-contained cells each with their own sensing, cognition and actuation embedded in shape.

SCP will have the capacity to sense its immediate environment and process that data for intelligent actuation – actuation where the applied force is based on the type of material handled. SCP will also have a need for on-board information processing allowing for intelligent actuation that can be customized based upon the application and the operational environment. Because grasping a tomato requires different operational mode than tightening a metal bolt, SCP would allow for creating smart robotic actuators with the capacity to adjust grasping force for different objects.

I am part of a group at University of Denver, called Collaborative Mechatronics Lab, where we are involved in a large project aiming at developing all-polymer soft robot. It will be propelled using a concept called water hammer (discussed later in the text) where SCP will be used to aid in steering. SCP will comprise of multiple cells where each cell has the following capabilities: sensing, actuation, cognition, and data transfer. Figure 20 demonstrates the concept of SCP.
Sensing and actuation will be performed using polymer sensors and actuators. The data will be processed in a distributed fashion requiring distributed data processing algorithm as well as the capability for each cell to exchange the necessary information with other cells. For this data-processing task a form of a neural network is perfectly suited as it naturally allows for data processing to be distributed and, in the case of individual SCP cells becoming non-operational, it does not result in a catastrophic failure (Nawrocki, 2010a; Nawrocki, 2011c).

**Actuation – Water Hammer**

The water hammer effect, also known as fluid hammer, has been known since the introduction of the modern plumbing. Until recently it was viewed as a negative effect with capacity to destroy indoor and outdoor plumbing and, as a result, a number of remedies were developed to mitigate its effects. This phenomenon occurs when water
traveling through a pipe experiences a rapid and sharp change in pressure usually facilitated by a fast closure of a valve. Increase in pressure, at the point of the closure, is brought about by the continuous motion of the flowing liquid. The intensity of the water hammer effect is inversely proportional to the time in which the valve is closed: the shorter the shutoff time the greater the force of the effect.

We have verified, both analytically and experimentally (Yang, 2010) the efficacy of water hammer effect as either a sole source of propulsion of a soft robot or used to aid conventional sources of propulsion. We have also demonstrated, with 20 distinct examples, that the shape of the hose affects the directionality of propulsion (Nawrocki, 2011a). However, because the relationship of the overall shape to the directionality is complex and non-linear, as well as the need for processing of this relationship to be performed in a distributed fashion, I proposed that this cognitive task be performed by the Synthetic Neural Network.

**WH on SNN**

Because the currently proposed SNN does not have on-board learning capability, I first trained a conventional, emulated artificial neural network to perform the aforementioned shape-of-the-hose-to-water-hammer-directionality matching. The 4-input, 4-hidden neurons, and one-output network (shown in Figure 18) was trained in MATLAB to an accuracy of less than one percent. Having verified that conventional neural network is capable of extrapolating the non-linear water hammer pattern I manually exported those weights to an SNN. Each synthetic neuron had inputs composed of three OBDs which quantized the input weights to eight levels (weight quantization
error = 7.9%) used for quantized connection weights. Figure 21 demonstrates the setup of the water hammer experiment, while Figure 22 demonstrates 20 distinct propulsion directions (given in radians) associated with 20 distinct shapes, with outputs from MATLAB and SNN. Shape 0 (direction along a straight, or non-bent hose) is a reference vector (angle equal to 1.57 [rad]): shapes above or below the horizontal axis indicate shapes that resulted in angle of propulsion either to the left or to the right of the reference shape. It can be seen that the SNN is not only able to correctly classify two distinct patterns (left or right propulsion), but is remarkably accurate (average error = 8.8%) in relation to the ability to correctly identify the angle. This highly complex example demonstrates the feasibility of employing SNN to a real-world task and further validates usefulness of SNN.
shape of the hose; the first four yellow marks (light in grayscale) are used to show the points used for shape extraction by neural network.

![Comparison of ANN (Matlab) and SNN](image)

Figure 22. Comparison of outputs (given in radians) of a conventional ANN (blue or dark), trained on MATLAB, and the SNN presented here (red or light), trained to detect a directed shape due to water hammer effect on a hose. Shape 1 was used as a reference point (see Neuron Simulation section for explanation). Horizontal axis (value of $\pi/2$) is used as the reference: values above the axis indicate vector to the left of the reference direction, and values below the axis indicate vector to the right of the reference direction.

**Synthetic Neural Network for Wall Following Robot**

Robotic wall following is one of the most common examples used to demonstrate the simplest form of artificial intelligence for a mobile robot. Fit with range finders, such as sonar or laser range finder, the robot follows a wall, as in a corridor, at a pre-defined distance while attempting to fulfill its mission, such as finding food or other objects of
interest. This seemingly simple task often results in programming difficulties as the programmer needs to account for such unexpected occurrences as crossing hallways or open doors. The use of ANN allows freeing the programmer from tedious explicit programming in lieu of “programming by example”.

To further elucidate the effectiveness of the SNN to map complex input-output relationships, I trained it to perform a task of a robotic wall following. I obtained input data from 9 sonar sensors mounted on a mobile NOMAD SCOUT2 robot (shown in Figure 23), with the output being motor commands sent to two individual motors. With this data set, a conventional ANN was trained in MATLAB to an error of less than one percent. The data set that the network was trained on consisted of 100 data points. However, the testing set consisted only of a subset of the training data set, with a total of 20 examples (seen in Figure 24). The neural network contained 9 input cells, 15 hidden cells, and two output cells. The connection weights were then manually exported to SNN with three OBDs per input. Figure 24 demonstrates the comparison of the output of ANN trained in MATLAB to the normalized SNN output, with Figure 24A showing the right wheel output, while Figure 24B depicts the output of the left wheel. The average error, computed using Equation 6, of the right wheel output was approximately 8.8% while the average error of the left wheel output was approximately 11.6%, resulting in an average output error of 10.2%.
Figure 23. NOMAD SCOUT2 mobile robot used to obtain the necessary data (inputs from 9 sonars, and outputs from 2 motors) for wall following routine.
Comparison of ANN (MATLAB) and SNN output (right wheel)

Comparison of ANN (MATLAB) and SNN output (left wheel)
Figure 24. Comparison of the ANN (MATLAB) and SNN outputs for a robotic wall following task: the average SNN error was approximately 9.6%. (a) The output of the right wheel, with SNN accuracy of 8.6%. (b) The output of the left wheel, with an average SNN error of about 10.5%.
Chapter Six: Damage Resilience and Manufacturing Errors

A conventional microprocessor, when subjected to physical damage, will cease to operate resulting in a catastrophic failure. A hardware neural network, such as the Synthetic Neural Network, is based on a connection of semi-independent processing elements. Therefore, intuitively, a physical damage should not result in a catastrophic failure but only in performance degradation. Additionally, the aforementioned SCP (see Structured Computational Polymers for more details) will be fit with SNN based on polymer electronics technology. Semiconducting and conducting polymers are based on long molecular chains as opposed to the crystal structure of inorganic electronics. As a result, organic electronics are much more unpredictable; each electronic element can have a slightly different behavior (Calamia, 2011). Hence, the distributed nature of the proposed neuromorphic architecture presents another advantage when the building blocks are made of organic electronics components.

To this day very few studies have been conducted that characterize the performance degradation of a neural network subjected to neural damage or other manufacturing faults. The first in-depth study was published by Bolt (1991). He constructed fault models, such as Fault Injection, Mean-Time-Between-Failure, and Service Degradation Method, to measure the reliability of various neural networks. The Fault Injection technique involved subjecting a system to a known number of faults and measuring the subsequent degradation. Mean-Time-Before-Failure method was based on
measuring the average time period before failure first occurs. Service Degradation Method was a combination of Fault Injection and Mean-Time-Between-Failure methods. All of the aforementioned faults were discussed in a context of neural network models such as Associative Neural Networks and Hopfield Neural Network but were only examined in the context of Multi-Layer Perceptron. Unfortunately the author did not reach any universal conclusions except for stating that neural networks offer advantages with respect to fault tolerance and reliability characteristics.

Another major work in this field was published by Protzel et al. (1993) who investigated recurrent NNs. They applied the Traveling Salesman Problem (TSP) and the Assignment Problem (AP) for analysis of performance degradation under two main conditions: ‘stuck-at-0’ and ‘stuck-at-1’ faults. Their analysis primarily concentrated on the difference in degradation between the two investigated faults. They examined three cases, where the number of cities was equal to ten, twenty, and thirty. Their main finding was that solution quality to TSP and AP tasks of the network is significantly more present in ‘stuck-at-0’ fault than in ‘stuck-at-1’; throughout the progression of cell removal, in all the test cases, the ‘stuck-at-0’ suffered negligible degradation. The ‘stuck-at-1’, however, showed a significant degradation.

Tchernev et al. (2005) focused on finding the optimal size of a feedforward NN, trained on classification problems, which exhibited the most fault tolerance. As such his study is similar to the neural network optimization paradigm discussed later in this chapter (see Optimization section). The study concluded with a finding that, if the training sample is non-complex, there is an optimal size of a trained network that
produces the smallest fully fault-tolerant network when the network is replicated; there exists a network size, across different trials, that reliably produces fault-tolerant networks. However, for highly complex training sets different trials will result in fault-tolerant networks of different sizes.

Because of the interest of realizing the SNN using polymer electronics technology that is more prone to device unpredictability and in manufacturing faults, I aimed at understanding performance degradation when neurons are physically damaged or experience other manufacturing faults. A physical damage is equivalent with the ‘stuck-at-0’ scenario. The ‘stuck-at-1’ error would, in the case of the proposed SNN architecture, physically correspond to the output of the OFET being equal to that of the Gate voltage (such a fault could be due to manufacturing failure or physical damage). I investigated both types of faults, namely ‘stuck-at-0’ and ‘stuck-at-1’ faults. Additionally, optimization of the network size was explored as a part of the intrinsic property of the neural network resilience.

**Background**

In ANNs, most commonly the neurons, or cells, are arranged into three separate layers: input, hidden, and output. Input cells provide the network with the input data. Output cells provide the result of mapping. Hidden neurons are responsible for the actual computation of the mapping.

When constructing an artificial neural network the number of necessary hidden neurons needs to be determined. Networks constructed with too few neurons cannot represent the inherent complexity of the data and will converge slowly or not at all.
Providing too many neurons may result in overfitting the data (Lawrence, 1998). Also, the topology, that is the number of hidden layers that the neurons will be arranged in, needs to be decided. It is generally accepted that one or two hidden layers are sufficient for most applications (Ostafe, 2005).

I aimed to investigate the performance degradation of trained neural networks that were subjected to faults in order to determine if a particular network topology, namely the number of hidden layers, offers benefits based on damage location. I aimed to answer the following question: given a fixed number of neurons, is it better to arrange them into a single hidden layer or into multiple, smaller layers. Figure 25 illustrates the concept visually. The goal was to compare the performance of network architectures when damage occurs in the input layer against that of the damage in the hidden layer(s). Monte Carlo analysis of four cases of optimized and unoptimized networks (explained in the next section), trained using highly correlated and highly uncorrelated data were examined in order to generalize the results. Additionally, I inspected these four cases using both feedforward and recurrent networks.

Figure 25. Motivation behind investigation of ANN resilience: given a fixed number of neurons, is it better to arrange them into: (a) single hidden layer or (b) multiple hidden layers.
Method

Software

The analysis presented in this paper was performed on networks obtained from MATLAB™ (Neural Network Toolbox version 4.0.6). Both networks, feedforward and recurrent, were trained using resilient backpropagation \textit{trainrp} algorithm and the activation functions used were \textit{tansig}

\[
\varphi = \frac{2}{1 + e^{-2n}} - 1
\]

Equation 3

for hidden layer and \textit{purelin}

\[
\varphi = n
\]

Equation 4

for output layer (www.mathworks.com). MATLAB uses Mean Square Error (MSE) to calculate the train error.

Network Architecture

I examined four different network architectures. In the case of unoptimized networks, with both correlated and uncorrelated training set, the total number of cells in hidden layer(s) was kept fixed at 100 and the cells were arranged into a single hidden layer, separated into two layers with 50 cells in each layer, separated into three layers with 33, 34 and 33 cells in consecutive layers, or separated into four layers with 25 cells
in each layer. See Figure 26 for graphical representation of the arrangement and Table 2 for more details.

Figure 26. Various arrangements of hidden layers investigated: all cells in a single hidden layer (100), cells split into two hidden layers (50/50), cells split into three hidden layers (33/34/33), and cells split into four hidden layers (25/25/25/25).

In the case of optimized networks the total number of neurons in all hidden layers was not kept constant as, due to the optimization algorithm and depending on the number of hidden layers, the total number of neurons varied. In the case of uncorrelated data (feedforward/recurrent) the number of neurons was 94/98 for a single hidden layer, 43/47 for dual hidden layer, 41/45 for triple hidden layer, and 35/38 for quadruple hidden layer. The number of neurons across multiple hidden layers was always kept the same for all of the trials. In the case of correlated data (feedforward/recurrent) the following number of neurons was used: 15/17 for a single hidden layer, 13/15 for dual hidden layer, 12/14 for
triple hidden layer, and 23/24 for quadruple hidden layer. The higher number of cells in the case of four hidden layers was due to the fact that with a smaller number of neurons the network failed to train to an acceptably low error.

Two different network types were used with all of the aforementioned hidden layer division; feedforward, employing no feedback, and recurrent, with feedback. All the networks were trained using backpropagation training algorithm (discussed in the previous subsection). Both, feedforward and recurrent networks were used to investigate the aforementioned uncorrelated and correlated data sets with unoptimized and optimized networks. The feedback path for recurrent network was only from the output layer to the first hidden layer. Figure 27 shows the use and lack of feedback.
Figure 27. (a) No use of feedback (feedforward) and the use of feedback (recurrent) architectures. The feedback is only employed from the last hidden layer to the first hidden layer. (b) Snapshot from MATLAB simulation illustrating a recurrent network with two hidden layers (50 cells each) with a feedback from the output to the 1st hidden layer.

Data Sets

Two different data sets were used. The correlated data set (www.ncrg.aston.ac.uk) consisted of 404 test vectors with 34 inputs and 6 outputs. This data set was randomly divided into a training set (S₁) of 304 vectors and test set (S₂) of 100 vectors (S₁ ∩ S₂ = Ø).
Uncorrelated data set consisted of 1500 test vectors each with 30 inputs and 10 outputs: 450,000 numbers (30\times10\times1500), in the range between 0 and 1, were generated using random number generator in C language (C language has a pseudo random capacity, hence a program was written where the generator was tied to the clock when the numbers were generated). The numbers were then mixed before they were used as a training set in order to achieve the most randomized training. The purpose was to achieve a highly uncorrelated data and the network would learn chaos. The testing set (S_2) was equal to the training set (S_1) or (S_1 = S_2).

Optimization

A number of optimization algorithms exist that can be divided into two broad categories: constructive and destructive optimizations (Fahlman, 1990; Drucker, 1992). Generally speaking constructive optimization algorithms start with a minimum number of cells with additional cells added as needed throughout training. Destructive algorithms, also called pruning, start with larger-than-necessary networks with extraneous cells pruned upon finishing training.

A simple algorithm for pruning networks proposed by Suzuki (2001) was used. This method is based on the influence of the removed neuron on the error and the consequent retraining of the network. The following summarizes the algorithm:

1. train the network until the error is satisfactorily small
2. virtually remove a unit and calculate the error
3. repeat (2) for all units in a layer
4. remove the unit that resulted in the smallest error change (largest error)
5. retrain the network

6. if the retrained network converges (error is satisfactorily small) stop the algorithm, else go back to (2) and repeat the process anew.

The algorithm used to constructively optimize the network is called Cascade-Correlation Learning Algorithm (CAS) and was first proposed by Fahlman (1990), and elaborated by Teng (1996). The algorithm, modified for maximum size control, consists of three phases: TRAIN_OUTPUT, TRAIN_INPUT and SIZE_CHECK. The following is a summary:

- **TRAIN_OUTPUT** trains network until there is no significant improvement in error
- **TRAIN_INPUT** adds a single cell, with full-mesh connectivity
- **SIZE_CHECK** checks if maximum allowable size is reached

Before the networks were optimized, both the uncorrelated and correlated data patterns were first normalized and then reduced in size using Principal Component Analysis (PCA) with 2% reduction (Gaussian distribution of the data is calculated, followed by removal of components that fall outside of the 2% range). This procedure, also carried out in MATLAB, resulted in reduction of correlated input of 34 variables to 13 variables and output from 6 to 5 variables. The uncorrelated data was reduced from 30 to 28 variables in the input vector and the size of the output vector remained unchanged as 10 variables. 50 constructive and 50 destructive trials were conducted in order to obtain the average number of cells needed (the rationale was to keep the number of cells
per layer constant in both optimized and non-optimized networks for the purposes of comparison). The network sizes used can be found in Table 2.

Table 2 Number of neurons used with different network architectures (number of hidden layers used).

<table>
<thead>
<tr>
<th></th>
<th>Number of hidden layers</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>One</td>
</tr>
<tr>
<td>un-optimized &amp; optimized (feedforward &amp; recurrent)</td>
<td></td>
</tr>
<tr>
<td>un-correlated</td>
<td>Input 30</td>
</tr>
<tr>
<td></td>
<td>Hidden 100</td>
</tr>
<tr>
<td></td>
<td>Output 10</td>
</tr>
<tr>
<td>correlated</td>
<td>Input 34</td>
</tr>
<tr>
<td></td>
<td>Hidden 100</td>
</tr>
<tr>
<td></td>
<td>Output 6</td>
</tr>
<tr>
<td>optimized (feedforward)</td>
<td></td>
</tr>
<tr>
<td>un-correlated</td>
<td>Input 28</td>
</tr>
<tr>
<td></td>
<td>Hidden 94</td>
</tr>
<tr>
<td></td>
<td>Output 10</td>
</tr>
<tr>
<td>correlated</td>
<td>Input 13</td>
</tr>
<tr>
<td></td>
<td>Hidden 15</td>
</tr>
<tr>
<td></td>
<td>Output 5</td>
</tr>
<tr>
<td>optimized (recurrent)</td>
<td></td>
</tr>
<tr>
<td>un-correlated</td>
<td>Input 28</td>
</tr>
<tr>
<td></td>
<td>Hidden 98</td>
</tr>
<tr>
<td></td>
<td>Output 10</td>
</tr>
<tr>
<td>correlated</td>
<td>Input 13</td>
</tr>
<tr>
<td></td>
<td>Hidden 17</td>
</tr>
<tr>
<td></td>
<td>Output 5</td>
</tr>
</tbody>
</table>

As already mentioned in the case of correlated data the training set and testing set were different (available data was separated into two, disjoint sets: training set and testing set). Because of the nature of uncorrelated set, the entire set was used for training and testing purposes.

**Procedure**

For the purposes of obtaining a reliable average a total of 50 different trials with new randomized weights (between 0 and 1) were conducted. Upon training the network
(when the train error was less than 0.1%) the network was subjected to ‘stuck-at-0’ or ‘stuck-at-1’ faults. ‘Stuck-at-0’ fault was perpetrated by fixing the connection weights and the bias of that neuron equal to zero.

Because, with the activation function used ($tansig(10)=1$), the ‘stuck-at-1’ fault is analogous to a neuron constantly producing a maximum output of ‘1’ (see Figure 28 for illustration), in the case of input-layer faults, the input vector of affected neuron was modified to a value of ‘10’ (equivalent with the neuron producing a constant value of ‘10’). When the hidden layer(s) were subjected to ‘stuck-at-1’ faults, all of the connection weights of affected neuron were set to ‘10’, which resulted in the neuron generating an output equal to ‘1’.
Two types of neural faults were administered; faults to input layer and faults to hidden layer(s). Faults to neurons in an individual layer (input or hidden layer(s)) were carried out in a random fashion. Faults in the hidden layer(s) was conducted consecutively in successive layers (i.e. for a network with three hidden layers a damage in the first hidden layer was followed by a fault in the second layer, followed by fault in the third layer, followed by a fault in the first layer, etc.) until the maximum number of cells was affected. The following outlines successive faults perpetrated to a network with three hidden layers:

- Fault in 1 cell in Hidden 1 (1st cell affected)
- Fault in 1 cell in Hidden 2 (2nd cell affected)
- Fault in 1 cell in Hidden 3 (3rd cell affected)
- Fault in 1 cell in Hidden 1 (4th cell affected)
- ...
- Fault in 1 cell in Hidden 3 (12th cell affected)
- Fault in 1 cell in Hidden 1 (13th cell affected)
- Fault in 1 cell in Hidden 2 (14th cell affected)
- Fault in 1 cell in Hidden 3 (15th cell affected)
For the purposes of clarity Figure 29 graphically demonstrates the pattern in which the cells were sequentially affected.
Figure 29. Graphical representation of the damage pattern - the order in which the individual cells were removed. Figures (a) through (i) represent sequential removal of individual cells.
There is a wide body of evidence that suggests that the number of cells in a hidden layer should not exceed the number of training examples (Lawrence, 1998), in order not to overfit the data: there is a minimum number of cells needed to coarsely classify a training set, with additional cells used for fine tuning. Removal of 'critical' cells would then result in a significant drop of performance while a removal of 'non-critical' cells would result in either slight degradation or even improvement of performance (one form of optimization of ANNs is based on cell removal). A network that only contains necessary or critical cells is considered to be optimized. Because this research aimed at comprehensiveness of observation the goal was to investigate both optimized and unoptimized networks.

**Results and Discussions**

In all of the ‘stuck-at-0’ cases, that is with correlated and uncorrelated training data as well as with optimized and unoptimized networks, the conclusions are universal: for ‘stuck-at-0’ fault incurred to hidden layer(s) increasing the number of layers results in increasing the error (the network that experiences the least performance degradation is the network with a single hidden layer) – see Figure 30A and Figure 30B. However, when the ‘stuck-at-0’ fault occurs to the input layer, increasing the number of layers results in decreasing the error (the network that experiences the least performance degradation is the network with the most hidden layers or, in this case, four hidden layers) – see Figure 31A and Figure 31B. Increasing the number of hidden layers usually results in increasing the training time (significantly complex data might be trained relatively easily with a single hidden layer; however it might present great difficulty at
training networks with multiple hidden layers). This finding shows that there is at least one benefit resulting from increasing the number of hidden layers, namely the resilience of the network when affected by the ‘stuck-at-0’ fault in the input layer. Figure 32 illustrates preferred network architecture based on location of ‘stuck-at-0’ faults.

(a)
Figure 30. ‘Stuck-at-0’ faults in hidden layer, uncorrelated data, unoptimized network. (a) recurrent architecture. (b) feedforward architecture: for ‘stuck-at-0’ occurring in hidden layer(s) increasing the number of layers results in increasing the error.
Figure 31. ‘Stuck-at-0’ in input layer, correlated data, optimized network. (a) recurrent architecture. (b) feedforward architecture: for ‘stuck-at-0’ occurring in input layer increasing the number of layers results in decreasing the error.
Figure 32. (a) When 'stuck-at-0' faults occur in the hidden layer, it is better (network experiences less performance degradation) to arrange all the cells into a single hidden layer. (b) When 'stuck-at-0' faults occur in the input layer, it is better to arrange the cells into multiple hidden layers.

In contrast to the findings of the 'stuck-at-0' fault, the 'stuck-at-1' error does not result in a difference of performance dependent on the fault location. Regardless of whether the fault occurs in the input layer or in the hidden layer(s), the network that experiences the least performance degradation is the network where the fixed number of neurons was split into the highest number of hidden layers; Figure 33A and Figure 33B demonstrate this finding. It should be noted that there is usually a noticeable improvement in resilience (decrease of performance degradation) when the number of hidden layers if increased from one to two and, to a lesser degree, from two to three (as can be seen in Figure 33B). However, a further increase from three to four hidden layers either results in a minimal improvement of network resilience or, temporarily, may even result in a decrease of resilience (increase of the average error). This can be seen in Figure 33A when 4 and 8 cells are removed. Additionally, while Figure 33A and Figure 33B show the average increase of error, individual trials often result in errors varying widely (as much as 250%) from the average performance.
Protzel et. al. (1993) noted that, with Traveling Salesman Problem (TSP) and the Assignment Problem (AP), the performance degradation of a network subjected to ‘stuck-at-0’ faults is negligibly small, while the ‘stuck-at-1’ fault results in a significant decrease of network’s fidelity. This is greatly in line with my findings: fault administered to input layer, with feedforward architecture, optimized network, and correlated data, with a single hidden layer, resulted in significantly lower error (0.22% with 7 cells removed) when the type of fault was ‘stuck-at-0’ (see Figure 31), while the ‘stuck-at-1’ error affected networks suffered error about one order of magnitude higher (2.25% with 7 cells removed) (see Figure 33).
A likely explanation of the observed trend is that when the faults occur, regardless on the location, the error is propagated throughout the entirety of the network, resulting in "cushioning" effect: the closer the faults occur to the output of the network (single hidden layer is directly connected to the output), the less of the “cushioning” will the network experience and the greater the performance degradation. When the network contains only a single hidden layer, the error, measured at the output, travels only through a single layer (when the fault is administered to input layer) or through no layers (when the fault is administered to hidden layer). However, when the network consists of multiple hidden
layers, because of how the fault was being administered to the hidden layers (sequentially), the error will at least propagate through one of more layers resulting in network “cushioning” the error. This can be especially seen when the error occurs in the input layer.

This finding can also be explained by the analysis of the input values propagated to an output neuron. Because 'stuck-at-1' fault means that a neuron produces an output equal to '1', when the error occurs in the network with only a single hidden layer, this 'high' or maximum value will be immediately sent to the output neuron resulting in a significant difference from the original (desired) value. However, when the fault occurs in a network with multiple hidden layers, for instance with four hidden layers, this 'high' or maximum output of a hidden neuron will in 25% of the cases occur in 1st hidden layer, resulting in the error traveling through another 3 hidden layers before this signal arrives at the output layer. This signal, then, has three chances of being attenuated (if the connection weight of that neuron is sufficiently low) before being received at the output layer.

To further verify this relationship, I conducted another experiment where I kept the number of hidden layers fixed at four. I then administered the 'stuck-at-1' faults exclusively to individual hidden layers. Figure 34 demonstrates the findings. When the faults occur in the 1st hidden layer (the furthest away from the output layer), the error is significantly smaller than if the faults occur to the 4th hidden layer (directly providing the input values to neurons in the output layer). This, I believe, validates my assumption about the "cushioning" effect of multiple hidden layers.
In the 'stuck-at-0' analysis I identified a consistent difference in performance between feedforward and recurrent networks. I observed that in about 80% of the cases with the same settings (network architecture, data set) the recurrent networks perform modestly better than feedforward architecture when subjected to neural damage (observation valid only for unoptimized networks as optimization resulted in different network sizes). Recurrent networks with a single hidden layer experienced performance degradation of about 13% compared to 16% degradation of a network with feedforward architecture. I did not observe any such meaningful relationship between feedforward and recurrent networks when the error that the network is subject to was the 'stuck-at-1' fault.
Conclusion

The conclusion of 'stuck-at-0' analysis points to the fact that network with fewer number of hidden layers provides a better resilience to faults occurring in the input layer. However, when the ‘stuck-at-0’ fault occurs to hidden neurons, it is the network with the greater number of hidden layers that experiences the least performance degradation. The simulation of 'stuck-at-1' faults results in a uniform conclusion, that which applies to all types of networks (feedforward and recurrent), regardless of where the error would occur: arranging a fixed number of neurons into a greater number of hidden layers results in decrease of the performance degradation or increase of network resilience.

A fault in input layer is equivalent to a fault of input sensors, while a fault in hidden layer(s) is analogous to a fault in the processor itself. A robot that is trained to perform certain functions based on sensory readings will currently most likely utilize a serial computing machine, a CPU, to emulate the parallel processing environment of an artificial neural network. However, with the emergence of neuromorphic technologies based on memristive artificial synapses, or architectures proposed in here, based on OBDs, soon tangible neural networks employed to control physical systems will become a reality. Analysis of benefits of different neural architectures will become a necessity. This study suggests that the use of multiple hidden layers may offer an advantage in a form of resilience against physical damage or other manufacturing faults.

Biological neural networks are arranged into a highly modular arrangement of many different layers (cell types) that are commonly controlled by yet other layers (Atallah, 2004). During a progression of a disease or aging, when either cells deaths or
cell mutations occur, the animal does not suffer a fatal error due to a complete brain malfunctioning but experiences a gradual loss of a skill or a function. This study might shed a light as to a reason for highly modular and inter-dependent, multi-layered biological neural architecture.
Chapter Seven: Summary and Future Work

I have demonstrated, via computer simulation, that a simplified hardware neuron can be implemented with only three distinct electrical elements; two OBDs (for two-input neuron), one transistor, and two resistors. This neuron will possess the necessary functionality of an artificial neuron, namely synaptic multiplication, somatic summation and activation function. I have also demonstrated that a single neuron, with a connection weights quantized to sixteen levels, can successfully function as either AND or OR logic gate. Additionally, I have shown that SNN can be trained to perform, with a great deal of accuracy, real-world classification tasks, such as the ability to distinguish the directed propulsion of a hose due to water hammer effect or the ability to perform the wall following task for a mobile robot. Because of the plans of implementing such a hardware neural network using SNNs based on organic electronic materials and devices, my analysis was based on existing characteristics of electrical elements that I have created. Finally I have shown that, depending on the type of fault that a neuron can experience, namely ‘stuck-at-0’ or ‘stuck-at-1’ fault, and the prediction of where the physical damage is most likely to occur (input layer corresponding to sensors, or hidden layer corresponding to the actual SNN) the network should be appropriately arranged into either a single or multiple hidden layers, as the performance degradation seems to be facilitated by the number of hidden layers as well as where the physical damage occurred. It should be noted that, at least preliminarily, from the manufacturing perspective,
arranging all the hidden neurons into a single layer would be easier than arranging them into multiple hidden layers.

**Future Work**

This thesis concentrated on the primarily theoretical aspect of Synthetic Neural Network, namely simulation of an individual synthetic neuron as well as verification of the performance and efficacy of a network of such elements. This work, however, will be continued in my doctoral studies aiming at realization of a fully functioning SNN that will be created using low-cost solution processing on a flexible substrate and will be used in a soft robot powered by water hammer effect and aided by *Structured Computational Polymers*.

The first step will be to create a fully functioning single synthetic neuron. A preliminary design, shown in Figure 35, depicts the necessary layers (for the purpose of clarity the design only shows the top and bottom electrodes). Upon the creation of a single synthetic neuron a whole network of such neurons will be created with the goal of employment in a real world task, such as mobile robotic wall following.
Figure 35. Preliminary design of a single, self-contained synthetic neuron (see Figure 3 for neuron architecture) outlining individual manufacturing layers that allows for four \((n+1)\) possible connection weight values (this can be easily increased by increasing the number of Organic Bistable Devices per input). Resistive strips, made from PEDOT, are used for setting individual OBDS to ON/OFF states as well as resistors.
References


http://cns.bu.edu/nl/moneta.html (last accessed on 8 December 2010)

http://cns.bu.edu/nl/synapse.html (last accessed on 8 December 2010)

http://bluebrain.epfl.ch/ (last accessed on 20 December 2010)

http://www.neuron.yale.edu/neuron/ (last accessed on 20 December 2010)


http://www.intc.com/releasedetail.cfm?ReleaseID=386254&ReleasesType=Technology %20Leadership (last accessed on 10 May 2010)

http://www.ncrg.aston.ac.uk/NN/databases.html (last accessed on 10 May 2010)
http://www.mathworks.com/products/neuralnet/ (last accessed on 10 May 2010)


http://www.stanford.edu/group/braininsilicon/index.html (last accessed on 10 Dec 2010)

Appendix A: Comprehensive set of ANN ‘Stuck-at-0’ fault graphs

Stuck-at-0 faults in HIDDEN layer:
recurrent architecture, UNoptimized network, UNcorrelated data

Stuck-at-0 faults in INPUT layer:
recurrent architecture, UNoptimized network, UNcorrelated data
Stuck-at-0 faults in HIDDEN layer:
recurrent architecture, optimized network, UNcorrelated data

Stuck-at-0 faults in INPUT layer:
recurrent architecture, optimized network, UNcorrelated data
**Stuck-at-0 faults in HIDDEN layer:**
recurrent architecture, UNoptimized network, correlated data

**Stuck-at-0 faults in INPUT layer:**
recurrent architecture, UNoptimized network, correlated data
Stuck-at-0 faults in HIDDEN layer:
recurrent architecture, optimized network, correlated data

Stuck-at-0 faults in INPUT layer:
recurrent architecture, optimized network, correlated data
Stuck-at-0 faults in HIDDEN layer:
feedforward architecture, UNoptimized network, UNcorrelated data

Stuck-at-0 faults in INPUT layer:
feedforward architecture, UNoptimized network, UNcorrelated data
Stuck-at-0 faults in HIDDEN layer:
feedforward architecture, optimized network, UNcorrelated data

Stuck-at-0 faults in INPUT layer:
feedforward architecture, optimized network, UNcorrelated data
Stuck-at-0 faults in HIDDEN layer:
feedforward architecture, UNoptimized network, correlated data

Stuck-at-0 faults in INPUT layer:
feedforward architecture, UNoptimized network, correlated data
Stuck-at-0 faults in HIDDEN layer:
feedforward architecture, optimized network, correlated data

![Graph showing error with varying number of cells removed for different hidden layers.

Stuck-at-0 faults in INPUT layer:
feedforward architecture, optimized network, correlated data

![Graph showing error with varying number of cells removed for different hidden layers.]
Appendix B: Comprehensive set of ANN ‘Stuck-at-1’ fault graphs

**Stuck-at-1 faults in HIDDEN layer:**
feedforward architecture, optimized network, correlated data

**Stuck-at-1 faults in HIDDEN layer:**
feedforward architecture, UNoptimized network, correlated data
Stuck-at-1 faults in HIDDEN layer:
feedforward architecture, optimized network, UNcorrelated data

Error

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
Cells Removed

1 hidden layer
2 hidden layers
3 hidden layers
4 hidden layers

Stuck-at-1 faults in HIDDEN layer:
feedforward architecture, UNoptimized network, UNcorrelated data

Error

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
Cells Removed

1 hidden layer
2 hidden layers
3 hidden layers
4 hidden layers
Stuck-at-1 faults in INPUT layer:
feedforward architecture, optimized network, correlated data

Stuck-at-1 faults in INPUT layer:
feedforward architecture, UNoptimized network, correlated data
Stuck-at-1 faults in INPUT layer:
feedforward architecture, optimized network, UNcorrelated data

![Graph showing error vs. cells removed with different layers](image)

Stuck-at-1 faults in INPUT layer:
feedforward architecture, UNoptimized network, UNcorrelated data

![Graph showing error vs. cells removed with different layers](image)
Stuck-at-1 faults in HIDDEN layer:
recurrent architecture, optimized network, correlated data

Stuck-at-1 faults in HIDDEN layer:
recurrent architecture, UNoptimized network, correlated data
Stuck-at-1 faults in HIDDEN layer:
recurrent architecture, optimized network, UNcorrelated data

Stuck-at-1 faults in HIDDEN layer:
recurrent architecture, UNoptimized network UNcorrelated data
Stuck-at-1 faults in INPUT layer:
recurrent architecture, optimized network, correlated data

Stuck-at-1 faults in INPUT layer:
recurrent architecture, UNoptimized network, correlated data
Stuck-at-1 faults in INPUT layer: recurrent architecture, optimized network, UNcorrelated data

Stuck-at-1 faults in INPUT layer: recurrent architecture, UNoptimized network, UNcorrelated data