NOVEL NEUROCHIP DESIGN IMPLEMENTING ALOPEX FOR USE IN AN AUTOMATED DEEP BRAIN STIMULATION SYSTEM FOR PARKINSON’S PATIENTS

by

EMIR ELKHOLY

A dissertation submitted to the

Graduate School-New Brunswick

Rutgers, The State University of New Jersey

in partial fulfillment of the requirements

For the degree of

Doctor of Philosophy

Graduate Program in

Electrical and Computer Engineering under the direction of

Dr. Evangelia Micheli-Tzanakou

And approved by

________________________________________

________________________________________

________________________________________

________________________________________

New Brunswick, New Jersey

May, 2011
ABSTRACT OF THE DISSERTATION

Novel Neurochip Design Implementing ALOPEX For Use In An Automated Deep Brain Stimulation System For Parkinson’s Patients

By EMIR ELKHOLY

Dissertation Director:

Dr. Evangelia Micheli-Tzanakou

In order to counteract the debilitating effects of Parkinson’s disease, Deep Brain Stimulation (DBS) is used. However, due to the invasive nature of the procedure associated with DBS, there is a need for a system that can autonomously and reliably adjust itself based on the progression of the disease. The purpose of the neurochip in this system will be to autonomously counteract the abnormal brain behavior that causes tremors in Parkinson’s patients by sensing and stimulating portions of the brain over an extended period of time. Using the optimization technique ALOPEX developed in the Computational Intelligence Laboratory at Rutgers Biomedical Engineering, the developed hardware is able to reduce the unpleasant tremor and offer the subjects a much-improved quality of life. The promise of such research can lead to a device that would reduce the need for visits to doctors for parameter adjustment purposes. The approach of the research was to first enter recorded Parkinsonian brain activity in software, and develop the complete algorithm used to counteract the tremor. From this phase, the software is implemented using microcontrollers where it is tested for robustness in the laboratory. The final stage of research lays the groundwork for VLSI design of the device that can be fabricated. In theory, it has been shown that ALOPEX can successfully adapt the stimulation parameters in DBS in order to maintain the optimal
condition for the patient. Clinical testing on animals/humans will need to be done in order to verify these results.
Acknowledgements

Through the previous work of Dr. Evangelia Micheli-Tzanakou and her former students, much work has been done in the field of Parkinson’s research. Without her guidance and support none of the following research could have been conducted. I am also thankful for the access to the tools needed to conduct research in the Computational Intelligence Laboratory (CIL). Thanks must also be given to the members of the Electrical Engineering department whom advised me during my research including Dr. Michael Caggiano, Dr. Zoran Gajic, and Dr. Verica Gajic.

Through the continued support of my parents Nihad and Samy Elkholy, my brother Ryan, and my wife Dina, this research was possible.
Table of Contents

ABSTRACT OF THE DISSERTATION................................................................. ii
Acknowledgements ............................................................................................. iv
Table of Contents ................................................................................................. v
List of Tables .......................................................................................................... vii
List of Figures ......................................................................................................... viii
CHAPTER 1: Introduction ...................................................................................... 1
  1.1 Architecture of the Brain.................................................................................. 1
    1.1.1 Background .............................................................................................. 1
    1.1.2 Neurons .................................................................................................... 2
    1.1.3 Basal Ganglia ........................................................................................... 4
      1.1.3.1 Striatum .............................................................................................. 6
      1.1.3.2 Globus Pallidus .................................................................................. 6
      1.1.3.3 Substantia nigra and Ventral tegmental area ....................................... 6
      1.1.3.4 Subthalamic nucleus ......................................................................... 7
    1.1.3.5 Basal Ganglia Pathways ....................................................................... 7
    1.1.4 Action potentials ..................................................................................... 9
    1.1.5 Local field potentials (LFP) ..................................................................... 11
  1.2 Parkinson’s Disease ...................................................................................... 12
  1.3 Deep Brain Stimulation (DBS) ...................................................................... 14
    1.3.1 Description ............................................................................................. 14
    1.3.2 Theories on Effectiveness ....................................................................... 16
    1.3.3 Therapy Considerations ........................................................................... 18
  1.4 Commercialized Devices and Treatments ..................................................... 21
    1.4.1 Medication ............................................................................................. 22
    1.4.2 Pallidotomies ......................................................................................... 24
    1.4.3 Medical Devices for DBS ........................................................................ 24
CHAPTER 2: Purpose of the Dissertation ............................................................. 30
CHAPTER 3: Methods ............................................................................................ 34
  3.1 Signal Analysis .............................................................................................. 34
    3.1.1 ALOPEX ................................................................................................. 34
    3.1.2 LFP Frequency Analysis ......................................................................... 37
    3.1.3 LFP Processing Techniques ..................................................................... 38
  3.2 ADBS Parameter Targeting ............................................................................ 48
  3.3 Research Platform ......................................................................................... 52
    3.3.1 Hardware Tools ..................................................................................... 54
    3.3.2 Software Tools ....................................................................................... 56
CHAPTER 4: Software Results ............................................................................. 64
  4.1 ALOPEX Simulations ..................................................................................... 64
  4.2 Firmware Breakdown .................................................................................... 81
  4.3 VLSI Implementation .................................................................................... 90
  4.4 VLSI Approach ............................................................................................. 91
  4.5 Processor Specifications ............................................................................... 92
  4.6 ADBS Chip Design ...................................................................................... 94
    4.6.1 Processor ............................................................................................... 94
    4.6.2 PWM Module ......................................................................................... 108
    4.6.3 ADC Module ........................................................................................ 109
  4.7 Microcontroller Results .............................................................................. 112
  4.8 VLSI Results ............................................................................................... 122
CHAPTER 5: Hardware Breakdown ....................................................................... 131
  5.1 Negative Voltage Generation ...................................................................... 131
  5.2 Voltage Regulation Circuitry ........................................................................ 133
  5.3 Voltage Offset and Input Buffer Circuitry .................................................... 134
  5.4 PWM Post-filter .......................................................................................... 136
  5.5 Impedance Sensing Circuitry ........................................................................ 138
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.6</td>
<td>Power Considerations</td>
<td>140</td>
</tr>
<tr>
<td>5.7</td>
<td>Layout Considerations</td>
<td>143</td>
</tr>
<tr>
<td>5.8</td>
<td>ADBS Microcontroller Circuit</td>
<td>144</td>
</tr>
<tr>
<td></td>
<td>CHAPTER 6 : Discussion</td>
<td>147</td>
</tr>
<tr>
<td>6.1</td>
<td>Hardware Design Discussion</td>
<td>147</td>
</tr>
<tr>
<td>6.2</td>
<td>Software Design Discussion</td>
<td>149</td>
</tr>
<tr>
<td>6.3</td>
<td>VLSI Design Discussion</td>
<td>151</td>
</tr>
<tr>
<td></td>
<td>CHAPTER 7 : Summary and Conclusions</td>
<td>153</td>
</tr>
<tr>
<td></td>
<td>CHAPTER 8: References</td>
<td>159</td>
</tr>
<tr>
<td>APPENDIX A1</td>
<td></td>
<td>168</td>
</tr>
<tr>
<td>APPENDIX A2</td>
<td></td>
<td>171</td>
</tr>
<tr>
<td>APPENDIX A3</td>
<td></td>
<td>172</td>
</tr>
<tr>
<td>APPENDIX A4</td>
<td></td>
<td>174</td>
</tr>
<tr>
<td>APPENDIX A5</td>
<td></td>
<td>178</td>
</tr>
<tr>
<td>APPENDIX A6</td>
<td></td>
<td>180</td>
</tr>
<tr>
<td>APPENDIX A7</td>
<td></td>
<td>181</td>
</tr>
<tr>
<td>APPENDIX A8</td>
<td></td>
<td>185</td>
</tr>
<tr>
<td>APPENDIX A9</td>
<td></td>
<td>189</td>
</tr>
<tr>
<td>APPENDIX A10</td>
<td></td>
<td>194</td>
</tr>
</tbody>
</table>
List of Tables

Table 1.1: Action potential specifications
Table 3.1: Optimal Starting Parameters for Kuncel’s model
Table 4.1: Processor Specifications
Table 4.2: Arithmetic instructions supported by the processor
Table 4.3: Data transfer instructions supported by the processor
Table 4.4: Branch and set less than instructions supported by the processor
Table 4.5: Input and Outputs of the Processor
Table 4.6: Inputs and Outputs of the datapath
Table 4.7: Inputs and Outputs of the control
Table 4.8: ALU control information for supported instructions
Table 4.9: Register names corresponding to register number
List of Figures

Figure 1.1: The structure of the neuron [3]
Figure 1.2: Basal Ganglia Structures [24]
Figure 1.3: Basal Ganglia Pathways [14]
Figure 1.4: Depiction of an action potential [73]
Figure 1.5: Spatial Recordings of STN and GPi [7]
Figure 1.6: Human head with DBS lead implanted [46]
Figure 1.7: Sample DBS recording [31]
Figure 1.8: Different waveform types including an actual waveform from a Soletra unit [41]
Figure 1.9: Stimulation induced effects on adjacent structures of the STN [45]
Figure 1.10: Plot of variance function used in ALOPEX algorithm (left) and plot of gamma function used in ALOPEX algorithm (right)
Figure 3.2: Local Field Potential data sample
Figure 3.3: Single-sided amplitude spectrum of Local Field Potential data sample
Figure 3.4: Welch method for spectral density of Local Field Potential data sample
Figure 3.5: Local Field Potential data sample with artifact
Figure 3.6: Single-sided amplitude spectrum of Local Field Potential data sample with artifact
Figure 3.7: Welch method for spectral density of Local Field Potential data sample with artifact
Figure 3.8: ADBS System Overview [99]
Figure 4.1: The tracking of the optimal value for $x$ using ALOPEX (left) and Convergence of the optimal value for the $y$ cost function using ALOPEX (right)
Figure 4.2: Top View of 3D Gaussian Model (left) and Side View of 3D Gaussian Model (right)
Figure 4.3: Graph of ALOPEX tracking the optimal $X$ in a 3D Gaussian Function (left) and Graph of ALOPEX tracking the optimal $Y$ in a 3D Gaussian Function (right)
Figure 4.4: Graph of ALOPEX tracking the optimal cost function (3D Gaussian Function)
Figure 4.5: Model of DBS presented by Kuncel and Grille [61]
Figure 4.6: ALOPEX tracking the optimal voltage to stimulate with during DBS (left) and ALOPEX tracking the optimal frequency to stimulate with during DBS (right)
Figure 4.7: ALOPEX finding the minimum tremor response for a specific patient
Figure 4.8: Top-View of a function of two variables (left) and Side-View of a function of two variables (right)
Figure 4.9: Graph of ALOPEX tracking the optimal $X$ for a function of 2 variables (left) and Graph of ALOPEX tracking the optimal $Y$ for a function of 2 Variables (right)
Figure 4.10: Graph of ALOPEX tracking the optimal cost function (Function of 2 Variables)
Figure 4.11: Top level processor schematic showing control and datapath with I/O
Figure 4.12: Processor Block Diagram
Figure 4.13: Top level processor schematic showing datapath with I/O
Figure 4.14: Top level processor schematic showing control with I/O
Figure 4.15: Controller finite state machine diagram
Figure 4.16: Instruction flow of multi-cycle datapath
Figure 4.17: Instruction formats for R-type, I-type, and J-Type instructions
Figure 4.18: ADC Module using PWM module and ADBS Chip
Figure 4.19: top: Zoomed out view of stimulation waveform taken from MPLAB IDE showing the pulse-width and frequency characteristics. The x-axis is in terms of clock cycles. Bottom: A zoomed in view of the waveform showing the duty cycle of the PWM wave that dictates the amplitude of the resulting waveform.
Figure 4.20: Pulse-Width Modulation for 134 microseconds
Figure 4.21: Deep Brain Stimulation at a frequency of 15 Hz
Figure 4.22: Single pulse generation using dsPIC30F2011
Figure 4.23: Period of the pulsation in PWM
Figure 4.24: PWM signal after RC low-pass filtering
Figure 4.25: Figure 8.6: PWM signal after inadequate RC low-pass filtering
Figure 4.26: High frequency DBS with long pulse-widths
Figure 4.27: DBS pulse generated after filtering of PWM signal
Figure 4.28: Output from VLSI design of PWM unit
Figure 4.29: Output Waveforms for various different instructions
Figure 5.1: Negative voltage generation using microcontroller and two 2-bit demultiplexers
Figure 5.2: Zener Diode Regulator Circuit
Figure 5.3: Voltage Offset and Input Buffer Schematic
Figure 5.4: RC Filter to smooth PWM signal
Figure 5.5: RC filter with varying capacitance
Figure 5.6: Voltage divider used to sense impedance of contact
Figure 5.7: Impedance Sensing/ADC/PWM Interface
Figure 5.8: PCB Layout for ADBS Microcontroller Circuit
Figure 5.9: ADBS Microcontroller Circuit Schematic
Figure 7.1: Bi-directional I/O pad circuitry [82]
Figure A3.1: Step Response vs. Time of FIR Filter used in ADBS
Figure A3.2: Impulse Response of FIR Filter Used in ADBS
Figure A4.1: Graph of factor1 as iterations progress
Figure A4.2: Graph of factor2 as iterations progress
Figure A4.3: Graph of first Gaussian noise as iterations progress
Figure A4.4: Graph of second Gaussian noise as iterations progress
Figure A4.5: Graph of ALOPEX tracking the optimal X (Second Simulation of Gaussian Function)
Figure A4.6: Graph of ALOPEX tracking the optimal Y (Second Simulation of Gaussian Function)
Figure A4.7: Graph of ALOPEX tracking the optimal cost function (Second Simulation of Gaussian Function)
Figure A5.1: Graph of ALOPEX tracking the optimal X (2 Variable Function)
Figure A5.2: Graph of ALOPEX tracking the optimal Y (2 Variable Function)
Figure A5.3: Graph of ALOPEX tracking the optimal cost function (2 Variable Function)
Figure A6.1: Frequency analysis of post-filtered DBS pulse
Figure A6.2: Graph of 220-microsecond DBS pulse generated by microcontroller after post-filtering
Figure A8.1: Graph of ALOPEX tracking the optimal x parameter (Function of 6 Variables)
Figure A8.2: Graph of ALOPEX tracking the optimal y parameter (Function of 6 Variables)
Figure A8.3: Graph of ALOPEX tracking the optimal z parameter (Function of 6 Variables)
Figure A8.4: Graph of ALOPEX tracking the optimal a parameter (Function of 6 Variables)
Figure A8.5: Graph of ALOPEX tracking the optimal b parameter (Function of 6 Variables)
Figure A8.6: Graph of ALOPEX tracking the optimal c parameter (Function of 6 Variables)
Figure A8.7: Graph of ALOPEX tracking the cost function (Function of 6 Variables)
List of Abbreviations

EEG  Electroencephalogram
fMRI  functional Magnetic Resonance Imaging
PD    Parkinson’s disease
GPi   Globus Pallidus internus
STN   Subthalamic Nucleus
VIM   Ventral Intermediate Nucleus
GP    Globus Pallidus
SN    Substantia Nigra
VTA   Ventral Tegmental Area
SNr   Substantia Nigra pars reticulata
GPe   Globus Pallidus externus
SNc   Substantia Nigra pars Compacta
LFP   Local Field Potential
DBS   Deep Brain Stimulation
IPG   Implantable Pulse Generator
ADBS  Automated Deep Brain Stimulation
disp  Digital Signal Processing
PWM   Pulse Width Modulation
ADC   Analog to Digital Converter
DAC   Digital to Analog Converter
3D    Three-Dimensional
MHz   Megahertz
IC    Integrated Circuit
DC    Direct Current
ALU   Arithmetic Logic Unit
MIPS  Millions of Instructions Per Second
MIPS  Microprocessor without Interlocked Pipeline Stages
SPI   Serial Peripheral Interface Bus
FFT   Fast Fourier Transform
CDF   Cumulative Distribution Function
I/O   Input/Output
FSM   Finite State Machine
PLL   Phase Locked Loop
FIR Filter  Finite Impulse Response Filter
IIR Filter  Infinite Impulse Response Filter
DFT   Discrete Fourier Transform
CHAPTER 1
Introduction

1.1 Architecture of the Brain

1.1.1 Background

Regarded as one of the most complex biological structures in the human body, the human brain is an organ that has been heavily researched, yet a complete understanding of the underlying mechanisms has not been determined. Although much has been discovered in recent years, research has not reached the point of constructing a fully functional model, nor anything that can even be used for testing purposes. The main research approach has focused on modeling certain portions of the brain based on observed activity at certain distinguishable sites. One certainty with regard to the brain is that neurons are the fundamental component. These neurons are responsible for relaying and processing information dealing with movement and perception. Perhaps the complexity of the brain can be attributed to the sheer number of neurons that make up the brain. According to one estimate, there are between 10 billion and 1 trillion neurons in the human brain with the average human brain having 85 billion neurons [66]. Besides the enormous amount of neurons in the brain there exist multiple layers of interconnections between them. It has been estimated that within a cubic millimeter of human cortex there are 50,000 neurons that contain 150 meters of dendrites and 100 meters of axons with each projection from the neuron having 50 million synapses [67]. With this in mind, it is reasonable to expect that a single firing of a neuron in one portion of the brain can permeate to other sections of the brain. It is clear that understanding the activity of the brain by studying individual neurons is a difficult proposition. For this
reason researchers have taken another approach involving sensing aggregate brain activity. An attempt to view the activity of the brain is reduced to a top-level representation produced using an Electroencephalogram (EEG) or functional Magnetic Resonance Imaging (fMRI). Another method used to study the brain involves dissecting the brain into thin slices for examination. This technique is not used to observe the brain activity of a living subject (for good reason).

The section of the brain that is researched heavily with respect to Parkinson’s disease (PD) is the basal ganglia. Before the disease itself is described it is helpful to understand the site where the problems occur. However, in order to understand how different sites of the brain interact, it is important to start with the basic building blocks of the brain.

1.1.2 Neurons

The fundamental building block of the nervous system is the neuron [3]. The neuron is a nerve cell that is similar to other cells in many ways, but specialized in others. The specialty of the neuron is the ability to transmit information to other neurons using both electrical and chemical means. The uniqueness of the neuron is most apparent when understanding the structure of the nerve cell. The membrane of the neuron is what allows information to be sent to other cells. The connections between cells are called synapses, and it is the neurotransmitters (chemicals) that are released into these synapses that allow the communication with other neurons. In order to gain a better understanding of what the inherent properties of neurons are, one can examine the models proposed by Hodgkin and Huxley [5]. The equations presented by Hodgkin and Huxley describe action potentials within a cell. The model is based on a non-linear differential equation with
three supporting linear first-order differential equations that describe the characteristics of cell membranes. The equations mainly are based on the transfer of sodium and potassium ions across a cell membrane. Newer models have been presented which account for the complex geometries of the neuronal structures.

The neuron is basically composed of the cell body, the dendrites, and the axon (Figure 1.1). Depending on the function of the specific neuron, the shape of the cell can vary as well as the number of dendrites.

![Figure 1.1: The structure of the neuron [3]](image)

It is the dendrites that are covered with synapses, and this is the site where information is received from other neurons and electrically transmitted to the soma. The signals from the dendrites are joined at the site known as the soma. Located at the end of the soma is a structure known as the Axon Hillock. If the total strength of the signal exceeds that of the threshold limit of the Axon Hillock then a signal is sent via the axon in the form of an action potential (section 1.1.4). The larger the axon is in conjunction with the amount of myelin (substance which acts as an insulator) on the axon dictates the speed of the transmission. The synapses are gaps located at the end of the terminal buttons. Neurotransmitters are responsible for transmitting the signal across the synapse to
adjacent neurons. A neurotransmitter that plays a major role in the operation of the brain is dopamine. Dopamine is often associated with movement disorders such as Parkinson’s disease (PD). It is the lack of dopamine that is the key characteristic of the disease. Thus it is understandable that a key medicinal treatment for Parkinson’s uses dopamine to cope with the disease [68].

Each neuron is influenced by thousands of synaptic junctions, which means that each neuron affects its neighboring neurons in an excitatory or inhibitory manner. The group of Smith et al. [6] observed that single sub-thalamic neurons strongly influence a large population of pallidal neurons in a uniform manner, effectively regulating the gain of the pallidum. This “projection” of neuronal brain activity enables researchers to study neurons on the order of 40 micrometers by examining the larger brain structures that are influenced by it.

1.1.3 Basal Ganglia

When studying Parkinson’s disease it is crucial to understand the inner workings of the parts of the brain associated with the disease. The main portion of the brain associated with Parkinson’s disease is the basal ganglia. Within the basal ganglia the two main areas of interest are the Globus Pallidus internus (GPI) and the Subthalamic Nucleus (STN) [1]. These are the two areas that are examined most often for electrode implantation due to the research done associating them with PD. However, these are not the only areas examined with respect to PD. Another region of the brain that is of importance with respect to non-parkinsonian essential tremor is the Ventral Intermediate Nucleus of the thalamus (VIM) [2].
It is known that the basal ganglia contain the most dopamine neurons in the brain [4]. The basal ganglia are composed of nuclei that compose different cortical circuits that control various types of behavior including cognitive and emotional behavior. These nuclei are perhaps better known for the apparent effect on motor control. The basal ganglia are composed of the following nuclei: caudate nucleus, putamen, nucleus accumbens, globus pallidus (GP), subthalamic nucleus, and the mesencephalic nucleus of the substantia nigra (SN) and ventral tegmental area (VTA).

![Figure 1.2: Basal Ganglia Structures [24]](image)

The basal ganglia are located at the base of the forebrain. They are connected to the cerebral cortex and thalamus as well as other structures in the brain. The physical locations of the different structures within the Basal Ganglia can be seen in Figure 1.2 above.
1.1.3.1 Striatum

The largest section of the basal ganglia, the striatum, is composed of the caudate and the putamen. The striatum is known to be a major gate for the entrance of glutamatergic cortical and thalamic inputs to the basal ganglia. It is also a gate for the entrance of dopaminergic inputs from the SN and VTA. The neurons of the striatum project largely and for the most part exclusively to the Substantia Nigra pars reticulata (SNr)/GPi and Globus Pallidus externus (GPe) [4]. The striatum is also referred to as the striate body, and the term striatum refers to the layered appearance of bands of the brain gray matter [70].

1.1.3.2 Globus Pallidus

The globus pallidus is composed of three defined areas. The areas of the globus pallidus include the external portion (GPe), the internal portion (GPi), and the ventral pallidum. It is accepted that the GPe is involved with the communication among the different portions of the basal ganglia. The GPi is known to be one of the major inhibitory outputs of the basal ganglia [4]. It is the Globus Pallidus that can be examined with medical electrodes to quantize the effects of treatments [65]. It is involved in the regulation of voluntary movements, and thus is the target of procedures such as the pallidotomy. The Globus Pallidus (or pale globe) is characterized as a subcortical structure lying deep within the brain [71].

1.1.3.3 Substantia nigra and Ventral tegmental area

The Substantia nigra can be further divided into two sections known as the pars compacta (SNc) and the pars reticulata (SNr). The SNc and the Ventral tegmental area (VTA) contain predominately if not exclusively dopamine neurons [4]. These two sections are involved in the fluctuations of the neurotransmission within the basal
ganglia. They project to the striatum as well as to the GP and STN albeit to a lesser extent. When the subthalamic nucleus is excited during treatments such as deep brain stimulation this causes an increase in the dopaminergic neurons of the substantia nigra pars compacta which enhances dopamine release in the striatum [72].

1.1.3.4 Subthalamic nucleus

The Subthalamic nucleus (STN) is directly interconnected with most of the basal ganglia except for the striatum [4]. It is known that it receives excitatory inputs from the thalamus. Known as the primary target for treatments such as deep brain stimulation, the subthalamic nucleus is a key structure of the basal ganglia that is studied with respect to Parkinson’s disease. It is the stimulation of the glutamatergic neurons of the STN that indirectly activates nigrostriatal dopamine neurons via reciprocal excitatory innervation back to the STN, which leads to SNc activation [72].

1.1.3.5 Basal Ganglia Pathways

Modern understanding of the basal ganglia was derived from the Albin, Young, Penney [13] and Delong [14] model, which proposed a functional relationship between the nuclei of the basal ganglia. The basic model in Figure 1.3 depicts the different sections of the basal ganglia interacting in both an excitatory and inhibitory manner. Additional modified models were proposed by Delong [14] that display the inhibitory, excitatory, and neural paths (as well as feedback loops) influenced by dopamine.
This same model has been used by other groups including Moyer, Danish, and Finkel [15]. There are two classic basal ganglia pathways that have been described in research and a third pathway that has recently been discussed. Classically, the basal ganglia have two pathways that are not fully disconnected, the direct and indirect pathways [4]. The direct pathway has neurons that project to the GPi and the SNr. As a result, the output of the basal ganglia is inhibited which disinhibits the activity of the thalamus. The direct pathway is understood to run in the following chain: cortex→striatum→GPi→thalamus→cortex.

Of the four links, two are excitatory and the rest inhibitory, however the overall effect is the cortex exciting itself through the direct pathway. The indirect pathway involves neurons connected to the GPe projecting to the STN. In turn the neurons of the STN project to the GPi and the SN. When these neurons are active the GPe is inhibited which causes the disinhibition of the STN. The extra excitation to the GPi and SNr
inhibits the thalamus. The indirect pathway is understood to run in the following chain: 
cortex→striatum→GPe→STN→GPi→thalamus→cortex.

Three of the five links are inhibitory, thus the cortex inhibits itself through the 
indirect pathway. The third pathway that has been identified in the basal ganglia is called 
the hyper direct pathway. The striatum is completely avoided in this path. The series of 
connections involves the motor cortex, STN, and GPi. The role of this pathway is to 
inhibit actions that have already been initiated [4].

1.1.4 Action potentials

When the electrical activity of brain matter is observed intracellularly (within the 
cell) the potential difference that is recorded is known as an action potential. Such 
recordings can be taken using microelectrodes implanted into certain neurons. The 
voltage that is being measured is along the axon of the neuron. The following chart 
illustrates the basic action potential characteristics of a neuron that one can expect during 
intracellular recordings:

<table>
<thead>
<tr>
<th>Table 1.1: Action potential specifications</th>
</tr>
</thead>
<tbody>
<tr>
<td>Duration</td>
</tr>
<tr>
<td>----------</td>
</tr>
<tr>
<td>1 to 2 ms</td>
</tr>
</tbody>
</table>

As can be seen by the time duration of the action potential, the signal lasts for a 
time period of the order of a couple milliseconds. When observing the resting potential of 
the neuron it is found that generally this value is around –70 mV. The resting current 
under this normal condition is on the order of –50 to 20 pico-amps, a very minute current 
value. However, when the current in the cell exceeds a threshold value, this causes the
generation of a spike which is the action potential. $I_{\text{rheobase}}$ refers to the minimum current needed for the resting potential $V_m$ to reach the peak potential $V_t$ and generate an action potential [1]. As can be seen from the increase in current from the resting current ($I_m$) to the rheobase current, there is a substantial increase in current within the cell. This method of communication between neurons consumes little power, and it explains why the brain consumes less power than a standard light bulb (only a mere 20 watts) [69]. As it can be seen in the depiction of an action potential in Figure 1.4, the average numbers given in table 1.1 are roughly representative of the characteristics of an action potential.

![Figure 1.4: Depiction of an action potential](image_url)

As can be seen from Figure 1.4 the characteristics of the rising and falling phase depict a “spike” in activity. Action potential signals originate from a single neuron, and based on the aforementioned number of neurons it is understandable that in order to decode what an area of the brain is contributing to a process, a broader variety of signals must be examined.
1.1.5 Local field potentials (LFP)

Although much can be gained from examining the intracellular activity of specific neurons, based on the sheer number of neurons it is best to examine the aggregate effect of a large group of neurons. This electrophysiological signal which is denoted as a local field potential (LFP) is the sum of all dendritic synaptic activity within a certain volume of tissue. Fortunately, LFP’s can be distinguished from each other and categorized as to which portion of the brain they originated from.

Figure 1.5: Spatial Recordings of STN and GPi [7]

Danish, Moyer and Jaggi have presented several microelectrode recordings for both the STN and GPi tracks in the above figure [7]. It is clear that observing the LFP signals as an electrode is passed through the brain can make boundaries for different regions. The path of the electrode can be seen to follow the dotted lines that diagonally intersect the diagram of the Basal Ganglia. This discovery proved to be extremely useful in further studies of the brain as well as in neurosurgery.

When researchers examine LFP’s for pertinent information, the time response of the signal is less useful than the frequency response. For this reason, when examining LFP’s researchers categorize them according to the frequency ranges that compose the
signal. LFP’s are classified into four major categories; alpha, beta, high gamma, and very high frequency oscillations. It is known that the alpha band oscillations (7 – 13 Hz) are highly prevalent in human PD patients. This leads researchers to accept that the activity in this band underlies Parkinsonian tremor and is the result of changes in the STN-GPe connectivity following dopamine loss [23-25]. The next band, beta oscillations (11 – 30 Hz), represents baseline activity of the striatum. Upon initiation of movement, the signals in this band become desynchronized. In PD patients the oscillations are synchronized and only return to normal de-synchronization upon dopaminergic medication or initiation of movement [26-28]. The activity in the frequency band above the beta band is known as the gamma band. Next, the band that is rarely seen in PD patients off medication is the high gamma band (~70 Hz). These oscillations are inversely proportional to beta band oscillations because the oscillations increase with movement initiation and dopaminergic medication. Similarly, very high frequency oscillations around 300 Hz have similar characteristics as high gamma oscillations [27][29] [30].

1.2 Parkinson’s Disease

Parkinson’s disease is characterized by tremor, rigidity, and akinesia (inability to initiate movement) [1]. English physician James Parkinson initially described it in 1817 although others made earlier descriptions. When the affected patient is in a fixed position rather than in the midst of movement the effects of tremor are most visible. The rhythmic tremor is repetitive at the rate of 3 to 6 Hz. The rigidity encountered with Parkinson’s is described as an opposition to movement characterized by an increase in the tone of muscles with opposing action. Often the abnormal movements provide many challenges for patients when getting dressed, feeding themselves, or other routine activities. The
unfortunate aspect of the disease is that it is a progressive disease, causing the symptoms of the patient to worsen over time, even with treatment. Parkinson’s disease affects around 1.5 million Americans with 50,000 Americans newly diagnosed every year. With this growing population afflicted with this disease there is a growing need for treatments to ease the symptoms (mainly Parkinsonian tremor).

Parkinson’s disease can be observed in the brain by investigating the abnormal firing of neurons using standard medical electrodes. When there is a sufficient lack of dopamine, the pathways shown in Figure 1.3 become either over excited or inhibited. In turn, the STN and GPi become overly active, showing signals of synchronized oscillatory bursting [16]-[20] with a mean firing rate of 37+/−17 Hz [21] [22]. Through the interconnections of the different parts of the basal ganglia it is clear that any irregularity in a certain portion of the basal ganglia causes a chain reaction that can have unfortunate consequences. Such is the case in Parkinson’s disease with the dopamine deficiency eventually manifesting itself into motor problems for the patient.

Furthermore, in efforts to delve deeper into the studies of Parkinson’s disease the STN and GPi can be segregated into different areas that are associated with movement in different parts of the body [8]-[12]. Although the movements made by Parkinson’s patients seem complicated, the precise region of the brain responsible for such movements is known. Ultimately it is the lack of dopamine that causes the plethora of motor deficiencies encountered by patients of the disease. Treatments that target this source of the motor deficiencies yield dramatically positive results with respect to normalization of motor movements in chronically ill patients.
1.3 Deep Brain Stimulation (DBS)

Although neurosurgery is regarded as a higher risk procedure than other common surgeries, with the advent of Deep Brain Stimulation (DBS) the implantation devices used to counteract brain disorders has become more commonplace. DBS first originated in France in 1987 out of ablation procedures performed by doctors where areas of the brain were burned or lesioned to counteract the effects of certain disorders [74]. The origins of the research done by Dr. Benabid in France can be found in Dr. Benabid’s studies with the electrical stimulation of the brain of rats [86]. After opening a hole through the skull, an electrode that is roughly one millimeter thick is directed toward the STN (the most common target), which is an area less than the size of a pea. The settings of the unit take weeks to months to adjust due to the fact that there are 60,000 different ways the device can be configured (typical DBS system offered by Medtronic Inc, Minneapolis, MN). Through the deliverance of electrical impulses, DBS reduces 50% to 60% of the symptoms of Parkinson’s disease [74]. The symptoms of the disease that are not affected by DBS include depression, anxiety, balance problems, cognitive decline, and memory loss.

1.3.1 Description

Deep Brain Stimulation is a treatment used to treat chronic pain, Parkinson’s disease, tremor and dystonia [32]. DBS can also be used for disorders such as epilepsy and Tourette’s Syndrome [74]. DBS treatment involves the implantation of a medical device that transmits electrical impulses to a lead through an extension implanted in either the subthalamic nucleus, globus pallidus, or Ventral Intermediate Nucleus (VIM) of the thalamus (the last for non-Parkinsonian essential tremor). The effects of the
treatment are reversible, as opposed to some other treatments. The lead is typically a coiled wire that is insulated with four platinum iridium electrodes. The implantable pulse generator (IPG) consists of a lithium-ion battery and circuitry for pulsation and control of the device.

![Human head with DBS lead implanted](image)

**Figure 1.6: Human head with DBS lead implanted [46]**

As it can be seen in Figure 1.6, the implantation of the DBS electrode is a highly invasive procedure that requires the implantation of the electrode to the center of the head, near the base of the brain and close to the optic tract. If the electrode is placed too close to the optic tract patients can experience serious issues including blindness. DBS is based on the same concept as the pacemaker for the heart, thus at times it is referred to as the pacemaker for the brain. The devices associated with DBS are highly regulated by the FDA. One of the pre-market approvals issued for DBS is for unilateral thalamic stimulation for the suppression of tremor or Parkinsonian tremor not adequately controlled by medications and where the tremor constitutes a significant functional disability [75]. The second pre-market approval is for the bilateral stimulation of the GPi
or the STN as an adjunctive therapy in reducing some of the symptoms of advanced, levadopa-responsive Parkinson’s disease that are not adequately controlled with medication [75].

1.3.2 Theories on Effectiveness

There are three classical theories as to why DBS is an effective treatment for Parkinson’s disease. The first of these theories states that the regular pulses of DBS induce synchronization (either in STN/GPi axons or downstream structures) at 70 Hz, attenuating the 15-30 Hz oscillations in favor of high gamma oscillations. Furthermore, DBS might minimize the ~ 300 Hz oscillatory activities in downstream neurons at roughly twice the stimulation frequency [33]. The second theory proposes that DBS does not attenuate the beta oscillations in favor of high gamma oscillations, rather it simply desynchronizes the pathological oscillations [34] [35]. Lastly, some scientists believe that stimulation of the STN may activate dopaminergic neurons in the SNc, which allows for the release of dopamine and suppression of PD symptoms [36] [37]. The last theory could be the reason why DBS is only a temporary cure and the effectiveness degrades over the years.

Figure 1.7: Sample DBS recording [31]
When examining the sample DBS recording in Figure 1.7 there are key characteristics worth noting. First of all, it is evident that the neural signals are in fact relatively “slow” compared to other electrical signals encountered in other applications (i.e. wireless applications or processor clock rates). The frequency content of these biological signals is below 500 Hz, and this consideration simplifies the hardware of any sensing device being constructed. Secondly, any impulses seen in a DBS recording are usually one of two occurrences. The first one of these occurrences is simply artifact from a doctor moving the lead during a recording session. The second occurrence is actually the effects of stimulation at a given frequency. Due to the uniform frequency of these impulses (Figure 1.7) it is reasonable to attribute the impulses to stimulation. The actual local field potential activity from the section of the brain is the plethora of activity seen in between the stimulation pulses. The frequency analysis necessary to understand the brain activity is done on this section of data.

Although there have been relatively few advances in the way DBS is performed on patients, there has been interesting research that may explain the underlying reasons why DBS works. Until recently the underlying mechanisms of how DBS reduces tremor in Parkinsonian patients was not understood, however work done by Bekar et al. [38] has shed further light into the underlying workings of DBS. Bekar’s group showed in thalamic slices from mice that DBS causes nearby astrocytes to release adenosine triphosphate (ATP), a precursor to adenosine (through a catabolic process). In turn, adenosine A1 receptor activation depresses excitatory transmission in the thalamus, thus causing an inhibitory effect that mimicks ablation or "lesioning". If this research is
accurate, it can be used alongside other models involving adenosine to develop novel methods to stimulate, as well as to understand why certain DBS parameters are more effective than others. One promising avenue for research involves the use of understanding the amount of adenosine needed from a biological perspective in order to inhibit neurons. However, the approach used to successfully determine the required parameters for DBS are similar to navigating an airplane. It is not entirely plausible, nor necessary to measure every single parameter in the control problem in order to navigate the aircraft successfully. Through the monitoring of certain key outputs, as well as the monitoring of the state of other variables, the inputs can be adjusted to successfully navigate the aircraft. Although a model involving adenosine would be helpful (if the theory involving adenosine is the absolute theory describing why DBS works), through monitoring other parameters (primarily LFP data) the end result can be reached in an acceptable manner.

### 1.3.3 Therapy Considerations

With the DBS systems in use, the neuro-electrode interface impedance steadily varies over time [39]. With a constant pulsing voltage from the stimulator, the current will increase/decrease over time, which must be accounted for in programming sessions. Clinical impedance measurements of brain matter are typically in the 500-ohm to 1500-ohm range [39]. For this reason, if an impedance of less than 50 ohms is detected then it is assumed that there is a short circuit in the medical device. A broken lead is assumed to be the case when impedances greater than 2000 ohms are recorded. Since the impedance varies over time for human subjects due to factors such as glial scarring, the parameters must be frequently adjusted to obtain the optimal settings for the patient. A common
programming setting when initially starting the DBS therapy is to set an initial pulse width of 60-90 microseconds and a frequency of 130 Hz [40]. A higher frequency of 185 Hz can be used if the patient is tremor predominant. After the stimulator is turned on the general procedure involves increasing the voltage by 0.1-volt increments. When the voltage is raised too high, side effects such as paresthesia (sensation of tingling, prickling, or numbness) will be observed, and this will establish the upper end of the voltage range. When testing the contacts, they are tested sequentially with a rest time between the different contacts in order to isolate the effects of each contact. After the technician programs the settings, the patient can alter the voltage, but only by plus or minus 0.4 volts. Surprisingly, the optimal cathode anode pairs are often found by trial and error. When undesirable effects are noticed, the technician might change the active contact to the adjacent contact that was being stimulated. If that is ineffective, the technician can switch to bipolar mode in order to restrict the current spread. However, in this mode, at least 0.5 volts must be added additionally to the monopolar voltage setting to have the same effect as monopolar mode. The initial program is usually done over a period of several months. There is much time and money to be saved if an automated system for DBS can be commercialized.

The three main parameters of the stimulation waveform that are adjusted by health practitioners are amplitude, pulse-width, and frequency. Amplitude is defined as the magnitude of change in an oscillating variable. These parameters can be seen in the waveforms in Figure 1.8 [41]. For the first waveform the amplitude is defined as –1.0 volts for the first half millisecond. In terms of commercial DBS systems, the amplitude is defined as the difference between the peak voltages of the waveform. For the last
The second parameter that is adjusted is the pulse-width of the signal. For the first signal in Figure 1.8 [41] the period of time in which the signal deviates from zero is half of a millisecond (pulse-width of half of a millisecond). Finally, the frequency is defined as the inverse of the time period of the repeated signal. The first signal in Figure 1.8 [41], if repeated in a manner fit for DBS would yield a frequency of 500 Hz. Between the variation of the three parameters a theoretically infinite number of waveforms (limited by the hardware of the medical device) can be produced. However, there are certain biological considerations that limit the combination of parameters chosen.

DBS research over the past couple of decades has yielded three important principles [41]. Firstly, the neural response to stimulation is modulated by the shape of
the stimulation waveform. This statement is not surprising, but for completeness it must be said that different stimulation waveforms have different neural responses. This is clinically verified, and should be expected when testing any DBS system on a given subject. If the neural response remains constant or unresponsive over a wide range of stimulation parameters, it is likely that either the device is implanted incorrectly or it is malfunctioning. Secondly, it has been found that the waveforms must be bi-phasic to prevent tissue damage. Bi-phasic stimulation implies that after a negative voltage is outputted from the device, immediately a positive voltage is given to counteract the effects of the negative voltage stimulation. For a negative one-volt command given to the Soletra stimulator (Medtronic Inc, Minneapolis, MN), the waveform generated can be seen at the bottom of Figure 1.8. The reason for the waveform switching from negative voltages to positive voltages (bi-phasic stimulation) is to ensure that the total charge delivered is balanced to zero. As the voltage switches polarity, the current direction changes, and thus the amount of charge is balanced to ensure the tissue is not damaged much further. Finally, it is the cathodic (negative) phase of the waveform that has the greatest effect on neural activation. Given this observation multiple stimulation patterns can be constructed with the negative phase of the waveform actually affecting the neural activation and the positive portion of the waveform is present for charge balancing purposes.

1.4 Commercialized Devices and Treatments

To date there is no cure for Parkinson’s disease in any stage of development. Although this is unfortunate for the millions of people afflicted with the disease, there are a multitude of different treatments that are used to counteract the effects of this
progressive disease. The goal of such treatments is to allow the patient to enjoy a higher quality of life for as long as possible before the debilitating effects of the disease eventually take hold. The ideal form of treatment for Parkinson’s is clearly one that is associated with a pharmacological approach. As previously discussed, the origin of the disease is located in sections of the brain that are not easily accessible. If one were able to take medication to cure the disease the risks of surgery could be avoided altogether. When the surgery approach is chosen (as a last resort) there are other options besides DBS (although DBS is regarded as the best surgical option because its effects are reversible).

1.4.1 Medication

Known as the first line of treatment for patients with Parkinson’s, medication is a helpful tool that is tailored to the needs of each individual patient [76]. The primary factors that doctors consider when prescribing medication are the age of the patient, the symptoms, as well as any other pre-existing conditions. It is difficult to quickly prescribe the exact dosage and proper regimen of medications to take, but there are standard medications that have been proven to be effective.

The most common form of treatment for patients with Parkinson’s disease is a combination of levadopa and carbidopa known as Sinemet ®. It is the levadopa that is quickly converted to dopamine, however most of the levadopa is metabolized before it reaches the brain. The purpose of the carbidopa is to block the metabolization of the levadopa by the liver, thereby decreasing nausea and increasing the amount of dopamine that reaches the brain. This treatment is effective when treating bradykinesia and rigidity, but not as effective when treating the motor deficiencies associated with the disease.
Furthermore, as with any medication, the side effects are substantial including nausea, low blood pressure, dyskinesias, depression, confusion, and visual hallucinations.

The second line of treatments for Parkinson’s patients involves the use of dopamine agonists. These prescriptions are substitutes for Sinemet®, but are less effective. They attempt to mimic the effect of dopamine in the brain. The side effects are similar to those of levadopa. Dopamine agonists include Bromocriptine (Parlodel®), Pramipexole (Mirapex®), and Ropinirole (Requip®). Another drug that is commonly used is that of Amantadine (Symmetryl®) which is an antiviral drug that increases the release of dopamine that can be used in conjunction with levadopa for early stage Parkinson’s patients. Its effectiveness is of the order of 3 to 4 months and it has a multitude of side effects.

It is known that dopamine is oxidized by monoamine oxidase B (MAO-B), thus there are medications that inhibit MAO-B. These medications (Azilect® and Carbex®) increase the effectiveness of levadopa. Another line of drugs that works on an entirely different premise is that of the anticholinergics. These medications balance the over activity of the neurotransmitter acetylcholine. They are used to control tremor, and are used alongside levadopa. These drugs include Cogentine®, Akineton®, Benadryl®, and Artane®. The last class of medications used to treat Parkinson’s patients is COMT Inhibitors. As the name suggests, the COMT enzyme (which breaks down dopamine in the brain) is inhibited by this medication. These medications include Comtan® and Tasmar®.

It is clear from the number of different medications used that the goal is to alleviate the symptoms of the disease, not to cure the actual disease. They all involve
either the direct addition of dopamine or the inhibition of enzymes which breakdown dopamine. None of the medications target the cause of the lack of dopamine production. As previously mentioned, Parkinson’s is a progressive disease. Even with these medications, the patients know that the relief from the symptoms that they have are temporary, and unless a real cure is developed they will eventually succumb to the unfortunate effects of the disease.

1.4.2 Pallidotomies

The first type of surgical treatment for Parkinson’s patients was the pallidotomy [59][62][65]. This procedure involves the insertion of an electrical probe into the brain (specifically the globus pallidus). This probe is then heated to 80 degrees Celsius typically for about a minute. The goal of the treatment is to destroy the brain cells in the region. This procedure is used to relieve tremor and rigidity. Other versions of the procedure involve the implantation of a hollow probe that allows for the circulation of liquid nitrogen to damage the area of the brain being targeted [87]. This procedure is primarily used with patients that respond well to levadopa treatment. However, due to the irreversible effects of the procedure, pallidotomies are rarely performed anymore. The new technique that has replaced it is deep brain stimulation.

1.4.3 Medical Devices for DBS

There are numerous neurostimulators available on the market today, however the devices produced by Medtronic have been proven to take most of the market share. St. Jude’s Medical has recently gained approval from the EU to market their neurostimulators in Europe. The current DBS system manufactured by Medtronic has had some common problems and complications associated with the surgery as well as the
device itself. When the electrode is implanted in the desired region of the brain there is a risk of intracerebral hemorrhage [43] or improper lead placement [28]. There are numerous complications associated with the stimulator and electrode used currently including skin erosion, electrode or wire break, lead migration, and pulse generator malfunctioning. The reason the skin tends to erode in some patients after a year of the surgery is because the stimulating device is excessively thick, and after implantation it tends to erode the skin. For patients of a leaner build it is recommended to use a dual stimulator system that uses thinner components. The issue of the electrode or wire breaking must be considered when the connector between the extension cable and electrode is located below the mastoid [29]. Through constant head turning the cable and electrode have a tendency to break over time. The issue of pulse generator malfunctioning can be traced back to faulty hardware, and this can be improved with better hardware design techniques either at the embedded systems level or even VLSI level. Furthermore, there are stimulator related adverse events that are common in DBS. Stimulation of the STN may induce dyskinesia, hemiballismus, dysarthria, paresthesia, and diplopia.

Even after a DBS system is developed correctly and is implanted perfectly by the surgeon, the issue of programming the deep brain stimulator is crucial to the alleviation of the PD symptoms. The largest benefit of DBS over medication alone is the lack of motor fluctuations and/or disabling dyskinesia following the procedure [2]. This is seen immediately after surgery, however the improvement is temporary and only due to the lesioning effect the inserted electrode has on the brain. For this reason the programmer of
the system should wait a week after surgery to start programming in order to view results that are closer to the real effect of the system and not just the lesioning effect.

When looking at the electrodes used in the Medtronic systems, two different variations are offered. Both electrodes have 4 contacts located on the electrode with a diameter of 1.27 mm and the same contact length of 1.5 mm. The two models deviate with the spacing of the contacts and the total length. For model 3387 the spacing of the contacts is 1.5 mm and the total length is 10.5 mm. For model 3389 the spacing of the contacts is 0.5 mm and the total length is 7.5 mm. When programming the contacts, each contact can be cathodic, anodic, or switched off, however the case can only be cathodic or off. When using the stimulator case as the cathode, the stimulation is referred to as monopolar, and when the voltage is referenced between two contacts, it is referred to as bipolar.

There are three Medtronic stimulator models, the Itrel, Soletra, and the Kinetra. Of the three, the Itrel is no longer used, however it served the same purpose as the other two stimulators. Three general characteristics define the stimulus waveforms generated by the Soletra and Kinetra [41]. First of all, the charge injected into the tissue during the anodic and cathodic phases equal each other (charge balancing in order to prevent tissue damage). Secondly, as opposed to the external pulse generators used during surgery, the anodic phase duration is shorter than the stimulation period dependent on the stimulation frequency and the type of Implantable Pulse Generator (IPG). Most importantly, the programmed stimulation voltage is roughly equal to the difference between the cathodic and anodic peaks rather than the magnitude of the cathodic pulse alone. These three
factors contribute to the actual observed stimulation voltage to be substantially less than
the programmed voltage.

There are differences between the Soletra and Kinetra stimulators that are
substantial. The most obvious difference between the two stimulators is that the Kinetra
supports two DBS leads for bilateral stimulation and the Soletra only supports one lead.
Regarding the lithium-ion batteries, the Soletra has a battery voltage of 3.7 V and uses a
voltage multiplier circuit to achieve voltages higher than 3.7 V. Using this voltage
multiplier circuit substantially increases the power consumption of the device. For the
Kinetra, the battery voltage is 3.2 V, and the power consumption increases linearly as the
voltage is increased. Another difference between the two systems is the inter-pulse delays
of each. The inter-pulse delay of the Soletra lasts about 0.5 ms and the final delay lasts
about 3.8 ms. On the other hand, the Kinetra interleaves two stimulation waveforms (one
for each lead in bilateral stimulation) in each cycle so the effective time interval used to
construct the overall waveform is half the period defined by the stimulation frequency.
The inter-pulse delay of the Kinetra lasts about 0.2 ms and the final delay is about 0.4 ms
(until the start of the pulse on the bilateral lead). For both of these systems, the duration
of the anodic phase is the time remaining in the total period after subtracting the cathodic
pulse-width, inter-pulse delay, and final delay. Due to charge balancing, the ratio of
cathodic to anodic magnitude can vary substantially across stimulation parameter
settings. The effective cathodic pulse amplitude is reduced because the amplitude
registered on the programming device is actually representative of the peak-to-peak
voltage difference between the cathodic and anodic phases.
As mentioned earlier, one of the problems that can arise while performing DBS on a PD patient is improper placement or migration of the electrodes. If the contacts migrate away from the targeted region of the brain, then it is likely that the patient can experience undesired side effects.

Figure 1.9: Stimulation induced effects on adjacent structures of the STN [45]

Figure 1.9 shows that stimulating surrounding regions of the STN can lead to side effects such as dystonia, dysarthria, akinesia and sweating [45]. When one realizes that the entire target area of the STN is about the size of a pea, it becomes clear why the placement of the electrode is of the utmost importance. Although the issue of miniscule electrode
migration (or misplacement) is not life threatening and can be fixed through the use of different contacts, the problem can be avoided through the use of a smarter medical device.
CHAPTER 2
Purpose of the Dissertation

Although commercial DBS systems are currently available, in order to rate the effectiveness of one such system there must be feedback from either the patient or the brain itself. The DBS systems that are currently used only stimulate the brain with certain characteristic waveforms (varying frequencies), but do not sense the brain activity before, during, or after the stimulation to quantify the effect. Unable to fully understand how the brain operates, the current research approach is directed toward an analysis of the outputs of certain areas of the Parkinsonian brain with respect to what is considered “normal” brain activity. With the current devices on the market the only feedback available is the comfort of the patient as well as the reduction in the effects the disease has on the patient. Although the current DBS stimulators serve their purpose (because comfort and the reduction of Parkinsonian tremor is the ultimate goal), one can question the effectiveness of these systems. Using the same thought process one would further inquire as to why these commercially available systems do not have feedback within the system itself as opposed to merely non-quantifiable feedback from the patient. Such feedback would mainly involve a processor deciding on the optimal contacts to use in order to relieve the patient from the side effects of the disease. This can effectively reduce patient discomfort while also reducing the number of trips to a physician.

One issue with sensing brain activity over long periods of time through the implantation of a lead and electrode is that of scarring. However, there is research that supports methods to help extend the sensing lifetime of a contact [42]. One such study involved the delivery of electrical impulses to the probe contacts on a specified time interval to allow for continuous sensing from that electrode. This process has been termed
rejuvenation [42] and has been proven to “clean” any matter that is on or around the electrode that might impede the sensing process (counteracting glial scarring). It has been shown using iridium contacts that applying 1.5 volts for a period of 4 seconds reduces the site impedances of chronically implanted microarrays. It was found to cause a transient increase in the electrode conductivity through an iridium oxide layer and a decrease in the surrounding extracellular resistance by 85% plus or minus 1%. Also, the immediate site resistance was decreased by 44% plus or minus 7%. This process can be added to any DBS system to allow for accurate recordings a longer time after surgery.

Therefore, on a theoretical level there is little reason for any DBS system to not have sensing capability. Such functionality of the system would reduce the frequent visits to licensed professionals or doctors to adjust the parameters of the device. Furthermore, the optimal parameters can be found electronically without the chance of human error. With the vast array of different parameter configurations possible and the time necessary to find the optimal parameters it is clear that human error can become an issue. Using the optimal parameters from an Automated Deep Brain Stimulation (ADBS) system would have power consumption savings given that the stimulation parameters would never be overshot and extra power dissipated. Furthermore, with a smart device such as an ADBS system the mechanics of the device can be expanded to support many more contacts since the device itself would be responsible for finding the optimal parameters, not a human being. Different arrays can be constructed, allowing for different combinations of contacts to be stimulated with various waveforms in different patterns. The lead in such a system must have custom analog circuitry included in a package small enough to be implanted. This neurochip would remain in the same hermetically sealed case used in
current systems. The primary purpose of the neurochip is to analyze the LFP’s sensed by the connected electrode and to analyze the data in order to adjust the key parameters of DBS (pulse-width, frequency, and amplitude). Such a system could help to further unlock the secrets behind DBS and provide researchers the freedom to thoroughly stimulate different parts of different nuclei with different waveforms. Mimicking the stimulation done by commercial devices is not difficult, but the techniques involved with sensing LFP data and making sense of the frequency response are not trivial.

It is clear that Deep Brain Stimulation is a viable treatment for many Parkinson’s patients, however the system can be improved in many ways. When developing the ADBS system it is clear that an integral portion is the ADBS processor. The rest of the complete system consists of the actual lead/electrode and analog circuitry associated with it as well as the battery and casing that will withstand implantation over an extended period of time (multiple years). The primary purpose of this dissertation is to fully develop a prototype that can be used as a part of the overall system for testing in a laboratory scenario. Although the actual ADBS chip will not be fabricated in VLSI due to the cost, the groundwork will be done which can lead to fabrication in VLSI. The specific deliverables of this dissertation are as follows:

1. **Develop** the complete algorithm used to counteract the tremor (in conjunction with ALOPEX), which involves targeting the optimal contacts with the optimal settings in the shortest amount of time.

2. **Test** the software using the MPLAB IDE and ICD3 (debugging environment tool) to ensure the proper execution of code.
3. **Build** a working prototype using microcontrollers that will be used to debug the software.

4. **Test** the prototype in a laboratory scenario using preset waveforms that will be sensed by the device and checked for accuracy. Furthermore the stimulation capabilities of the device will be tested and observed for accuracy.

5. Lay the groundwork for VLSI design of the device in VERILOG that can be fabricated when future funding is allocated.

The research from this dissertation is sufficient enough to be applied to testing on animals. A separate study can be completed on battery technology and medical casing for use in implantable devices. Finally, the entire system can be tested together and eventually used in clinical trials with humans.
CHAPTER 3
Methods

3.1 Signal Analysis

3.1.1 ALOPEX

When attempting to stimulate with the optimal parameters during DBS the number of different combinations do not allow for an exhaustive search through all of the possible stimulation patterns. Furthermore, finding these optimal parameters is further complicated by the fact that the medical device interacts with a biological system, not a discrete circuit that provides the exact same response to the same input over time (biological systems are often time-varying systems). For this reason, the optimization technique must be one that can find a rough estimation of the parameters in a timely manner. An optimization technique that is suitable for such an application is ALOPEX [62][88][93][95][96][97]. ALOPEX is an optimization procedure that can be used to optimize the weights in a neural network. Although a neural network will not be used for the ADBS system, the optimization technique adequately finds the optimal parameters to minimize the power sensed by the electrode. The main advantage ALOPEX has over other techniques such as backpropagation [98], Boltzmann machines [100], or a linear search of combinations is that as the number of parameters increases the speed of convergence is affected much less than other techniques. As the number of factors to optimize increases, ALOPEX becomes a better optimization technique to use over other methods. This is observed when understanding the nature of the convergence of the parameters, primarily the fact that they converge in parallel.
The basic premise of the optimization technique is that a cost function exists which is dependent on multiple parameters. For the ADBS system the parameters are the frequency of the pulse, the pulse-width, and the amplitude. For each combination of these three parameters, a different power response can be observed from the portion of the brain where the electrode is implanted. Therefore, the power observed is a function of three variables that can be adjusted by the device, and the power is the output that is monitored and minimized. It is important to understand that the trade off for the speed of convergence of the algorithm is the accuracy of the solution. However, it has been shown that the near optimal solution can be repeatedly determined in a timely manner. This optimization technique works in an iterative fashion allowing all of the parameters $X_i$ to be changed simultaneously each iteration according to the following equations:

\[ X_i(n) = X_i(n-1) + \gamma \Delta X_i(n) \Delta R(n) + r_i(n) \]  
\[ \Delta X_i(n) = X_i(n-1) - X_i(n-2) \]  
\[ \Delta R(n) = R(n-1) - R(n-2) \]

The key variables in the above equations are as follows. Each parameter associated with DBS is designated by $X_i$. The global response change $\Delta R(n)$ is defined as the difference in the value of the cost function. In the application of DBS the cost function is the power of the signal that is sensed at the contact location. This power should be minimized after the parameters have been updated through the iterations. The cross correlation of local changes is noted as $\Delta X(n)$. It is the product of the global response change and the cross correlation of the local changes that helps the process converge in the correct direction. Within the algorithm there are two factors that can be adjusted, the first of which is the gain factor gamma. The second factor that can be
adjusted is the additive noise $r(n)$. It is the presence of the second factor that makes ALOPEX such a powerful optimization technique. Inherently, noise must be added to the process in order for the results to converge correctly. It is clear that this is not an algorithm that is easily disrupted by the noise of the system itself. This further adds to the robustness of the optimization technique allowing ALOPEX to be used to optimize the most complicated of processes. The additive noise is of Gaussian distribution with a mean of zero and an initially large standard deviation (Figure 3.1). The standard deviation decreases as the process converges to ensure a stable stopping point. The factor of gamma increases with the iterations (right) to compensate for $\Delta R(n)$ decreasing (left) as the process converges. One safeguard that is built into the algorithm is that there is a maximal change permitted for $X_i$ per iteration. This limitation on the step size prevents drastic changes that can lead to oscillations preventing convergence.

![Plot of variance function used in ALOPEX algorithm (left) and plot of gamma function used in ALOPEX algorithm (right)](image)

Another key factor that must be accounted for when using ALOPEX is the number of iterations the algorithm is allowed to run for before expecting convergence. This number can be estimated from experimentation with the process, however as the
number of iterations is increased the chance of convergence to the correct result increases. It is important to note the stochastic nature of ALOPEX, and that all initial values of $X_i$ are random. This further shows the versatility of the algorithm and its ability to be used in a wide array of applications [88][89][91].

3.1.2 LFP Frequency Analysis

When sensing LFP data from different areas of the brain it has been found that certain frequencies correlate to certain symptoms of Parkinson’s disease. There exist clear peaks in the frequency response of LFP’s that distinguish a Parkinsonian signal from a normal signal. Generally, PD is associated with exaggerated oscillatory synchrony in the basal ganglia at frequencies over the 8-35 Hz range [55]. Furthermore, for patients with PD at rest, the beta frequency band has increased power, but the spectral profiles are different between patients [47]. The same study found that this “signature” rhythm for each patient was consistent between the right and left STN of each patient. However, the key factor that was realized from the aforementioned study was that longer periods of DBS attenuated beta power for a longer period of time suggesting there may be long acting functional changes to networks in the STN in PD after chronic DBS. It was found that after a short session of DBS the power in the beta band was attenuated for about 10 seconds after the stimulation was turned off. Although it was clear that beta band activity increased for patients with PD, it was also found that there was a clear peak in activity within the 2-13 Hz band [50]. It was further found in this study that activity increased after the administration of levadopa and apomorphine. Other research groups reached similar conclusions when examining an increase in low frequency oscillations in the 1-2 Hz range [54] as well as an increase at 7 Hz [51].
From the wealth of research on the subject there is evidence that if certain areas of the basal ganglia are regulated to a certain pulsation frequency then the motor movements can be altered. It was found that when Parkinson’s patients were stimulated at 20 Hz (which is a common synchronization frequency of patients with PD) that the physical tapping rates slowed down by 8.2 +/- 3.2% [56]. This is evidence that excessive beta synchrony within the basal ganglia-cortical loop may contribute to the slowing of movements in PD. It was further noted that activity around 20 Hz is suppressed prior to and during self/externally paced voluntary movements and also during motor imagery [57]. The same study linked the 10 Hz and 5 Hz oscillatory activities to the generation of tremor generation and other motor deficits. Overall, it is clear that activity below 40 Hz contains the information necessary to identify certain movement disorders like PD. However, one group reported that the beta band activity remained unchanged after treatments of levadopa, which is in direct contradiction of most other research [54]. However, they did attempt to link the peak in the 4-10 Hz peak with a specific neuronal pattern associated with the dyskinesias elicited by dopaminergic drugs [58]. This study cannot be overlooked when considering the feasibility of a medical device based on these principles. According to pallidotomy data from Dr. Tzanakou’s group [59][65][90][94], lesioning of the brain has shown to decrease the total power along the entire frequency range of the signal. It is this key observation that will be the foundation for the ADBS device fabricated.

3.1.3 LFP Processing Techniques

Researchers have used available equipment to sense and analyze LFP activity, however most of the systems that are in use for sensing brain activity from electrodes are
large systems that are used in conjunction with a computer. For an ADBS system to be feasible it must possess the same sensing capabilities as the large systems used in research yet small enough to be implanted in the patient. When sensing the LFP data from test subjects the following steps are typically taken to obtain the data [47-51]. The low voltage level of LFP data are in the range of tens to hundreds of microvolts, which is easily distorable when transmitted through a long wire. In order for this signal to transmit through the wire connecting the lead to the neurochip with a minimal amount of noise distortion the signal must first undergo pre-amplification. After pre-amplification the signal is then bandpass filtered between 1 Hz and 800 Hz and then further amplified. The exact filtration range is not critical, which has allowed researchers to band-pass filter from 2 Hz to 1000 Hz when sensing LFP data. The frequency components below and above these frequencies are not within the frequency ranges that are relevant to Parkinsonian tremor. Further amplification after band-pass filtering is usually of the differential variety, and is used to raise the signal to the level that can be interpreted by the analog to digital converter (ADC) of a processor. Once the signal has reached the ADC it is digitized for further processing. Researchers have used either a 10 or 12 bit ADC that samples anywhere from 1 kHz to 10 kHz with a quantization range of 5 volts. After the data was quantized, each research group resorted to examining the frequency characteristics of the signal. However, the intermediate steps and their interpretations of the signals slightly varied.

The strategy of some of the research groups involved normalizing the recording and imposing the same background noise to each of the recordings. For instance, the group of Marceglia and Foffani [51] normalized each recording by subtracting the mean
and dividing by the standard deviation of the 600-1000 Hz band-pass filtered signal in order to impose the same background noise to all recordings. The signals were then digitally band-pass filtered (2-45 Hz) and down-sampled at 125 Hz. A group of researchers noted that the group of Kuhn and Brown used the discrete fourier transform (DFT) for analysis as well as the following technique for normalization of data [52]. The autospectra of the LFP were estimated by dividing the records into a number of disjoint sections of equal duration (1024 points) and estimating the spectra by averaging across these discrete sections affording a frequency resolution of 1 Hz. When referring to the electrode contacts on a standard Medtronic recording probe the contacts are numbered from 0 to 3. When measuring electric potential between two contacts any combination of two contacts can be selected to record from. For instance, when a recording is made from a bipolar contact pair of (12), this means that the electric potential difference is being measured between the first contact and the second contact. The group of Kuhn and Brown recorded the LFP power from bipolar contact pairs (01, 12, 23) and expressed the power as a percentage of total power in the 5-95 Hz range. Frequencies below 5 Hz and in the 44-56 Hz band were excluded since they are prone to movement artifacts and main noise (noise from the power of the equipment) respectively. A subsidiary power analysis was performed using the mean power value of the 105-145 Hz frequency band to normalize data across subjects. Three frequency bands of interest were selected for further analysis: sub-beta (5-12 Hz), low beta (13-20 Hz), and the upper beta band (21-30 Hz). They then picked the contact pair of each side that displayed the maximum activity in the 13-30 Hz range for further analysis.
Although the aforementioned methods of LFP processing were adequate for the needs of those researchers, other groups implemented strategies that could be applied to an actual medical device that is implantable. The group of Stewart and Wingeier [48-49] calculated the power spectral density using an averaged periodogram method with Hanning windows of 1-second width and 0.5 second overlap, yielding a frequency resolution of 1 Hz. All time frequency spectrograms were calculated similarly, yielding a temporal resolution of 0.5 seconds and frequency resolution of 1 Hz. Frequencies below 3 Hz were deemed inseparable from low frequency artifact and were not shown in spectrograms. Segments were further divided into 1-second windows and a separate beta-band power estimate was calculated using the Fast Fourier Transform (FFT) for each 1-second window. When recording from the basal ganglia the length of the recording can be on the order of less than a second to yield representative results. For this reason the periodogram method used is closer to the analysis that can be done in real time by a neurochip.

The group of Priori and R.M. Villani analyzed the inputs using the following method [50]. The oscillatory activity of the STN at rest was quantified by power spectral analysis of LFP’s. Spectra were calculated using Welch’s averaged, modified periodogram method [53] implemented by the Matlab function psd in the following way. Signals were divided into sections of 2048 samples with no overlap; each section was detrended (i.e. the mean was subtracted to eliminate DC offsets) and windowed by a Hanning window. The squared magnitude of the DFT of the sections was averaged to estimate the power spectral density of the signal. To compare data before and after medication LFP’s were normalized by subtracting the mean and dividing by the standard
deviation of the 600-1000 Hz band-pass filtered signals. This procedure imposes the same background noise to all the recordings, hence reducing variability [54]. Five ranges were examined: low (2-7 Hz), alpha (7-13 Hz), low beta (13-20 Hz), high beta (20-30 Hz), and gamma (35-45 Hz). For each frequency band the power was defined as the average power in that band, expressed after decimal log transformation (log power). The log spectrum was transformed into a probability distribution function by normalizing it to its integral. The Kolmogorov-Smirnov statistic provided the maximum difference between the cumulative distribution functions (CDF’s). Significant activity was defined as the mean of the log spectrum plus the critical value at which the Kolmogorov-Smirnov test statistic became significant (P<0.05).

The LFP processing techniques to be used when analyzing brain activity during a DBS session were verified using data taken from recordings of pallidotomy procedures done by the group of Dr. Tzanakou. The data taken by Dr. Tzanakou’s group was during a pallidotomy procedure in which they were attempting to locate the optimal depth where the pallidotomy procedure would take place [59][65][90][94]. The equipment utilized possessed filtering elements, therefore the raw data that was given for this study was already filtered. The data were taken from a text file and imported into Matlab. When in Matlab the original function was plotted, the DFT of the signal was plotted, as well as the spectral plot using Welch’s method [Appendix A1][53]. The advantage of using the Welch method is for its additional low pass filtering capability that can easily be seen when comparing to the original data.
From Figure 3.2 it is evident that the LFP data are on the order of magnitude of hundreds of microvolts. The data are clearly highly stochastic, however it is not completely random. The frequency composition of the LFP is the key to the research in DBS. Behind this initially random function lies clear information in the frequency domain that will provide the basis for the entire automated system. When observing the waveform, it is evident that there is some consistency within the waveform. The main signal is clearly composed of smaller signals with distinct frequency components. This can be verified by observing the frequency response of Figure 3.2.
Figure 3.3 consists of the frequency spectrum of the signal in Figure 3.2. From Figure 3.3 it can be seen that the heavy activity of Parkinsonian brain activity in the subthalamic nucleus is less than 100 Hz. This is consistent with the research described earlier that ascertained the frequency characteristics of LFP’s in patients with Parkinson’s. However, from the single-sided amplitude spectrum above it is evident that there is baseline activity that is of no concern to a deep brain stimulation device. When using Welch’s method [Appendix A1][53] to calculate spectral density its main advantage is evident as much of the “noise” (as far as this research is concerned) is removed, allowing for easier analysis of the frequency spectrum. When looking at Figure 3.3 a baseline of activity is present throughout the total frequency range (similar to white noise).
Using the Welch method on the same data drastically reduced the noise that was present in the signal. From the original plot in Figure 3.3, the data that has undergone the filtering process is shown in Figure 3.4. Inspection of Figure 3.4 further allows one to surmise that the bulk of the characterizing signal is even less than 50 Hz when observing the Parkinsonian brain activity. With this information, the frequency range over which the chip is observing can be tightened even further, lessening the computational burden on the main ADBS chip. A second example of an LFP can be seen in Figure 3.5 where the LFP data are recorded after a pallidotomy is performed. It is clear that the magnitude of the signal is reduced heavily across the entire time the signal is recorded. However, in the frequency domain there is a slight difference between the LFP before the pallidotomy and after. There seems to be a shift in the location of the maximum in terms of frequency. But, our main concern is the total energy within the spectrum, which according to Parseval’s Theorem [Appendix A2] is equal in both the frequency and time domains. The
energy in the post-pallidotomy signal possesses less energy than that of the pre-
pallidotomy signal.

![Local Field Potential data sample with artifact](image)

**Figure 3.5: Local Field Potential data sample with artifact**

In Figure 3.5 there is a spike in voltage in the period between 5 seconds and 6 seconds. This is artifact due to the perturbation of the electrode during the surgery. This undoubtedly contributed to errant high frequency data in the signal. Once again, this illustrates the benefit of Welch’s method with its filtering effect. Although the high frequency data would be disregarded by any person examining the frequency content of the signal with respect to DBS, from a device perspective there must be filtering of some sort involved.
When comparing multiple spectrum graphs of pallidotomy data we can determine that the aberration at roughly 60 Hz (Figure 3.6) is due to the movement of the electrode during surgery (not noise from the power supplied to the equipment). This errant spike can be eliminated using a variety of low-pass filtering techniques such as Welch’s method. The graphs using the Welch method have eliminated the baseline noise associated with the entire signal. For the application of ADBS this is imperative to reduce data storage. The algorithms within ADBS are only concerned with the frequency ranges with high power spikes. Furthermore, from common LFP research associated with Parkinson’s it is known that the activity in the beta frequency region (below 35 Hz) is the frequency range of interest [51]. The use of Welch’s Method is further support for the use of digital filtering in software after analog anti-aliasing filtering.
Ignoring the aberration found at roughly 60 Hz in Figure 3.7 the main peak is much less than 50 Hz. The bulk of the activity is below 25 Hz, which will allow the device to focus on a small range of frequencies. If desired, the pre-filter can be tuned to filter out everything over this frequency range (if it is found in clinical tests that this range is in fact the only substantial range).

### 3.2 ADBS Parameter Targeting

When beginning to develop a methodology for finding the optimal parameters for deep brain stimulation time must be considered as a limiting factor for the approach used. Before considering the multitude of contact sites that exist which can be stimulated, we have a wide range of values for the three parameters of frequency, amplitude, and pulse width. The typical range of the settings according to Kuncel and Grille [60] are as follows:

- **Amplitude:** 1-3.5 V in 0.1 V steps (25 options)
Pulse-Width: 60-210 microseconds in 30 microsecond increments (6 options)

Frequency: 130-185 Hz in 5 Hz increments (12 options)

This yields 1800 different possibilities for parameter settings, and this is for a single electrode. This would take 30 minutes alone to process if each parameter setting test took only 1 second with sampling time, pulsating time, and processing time included. If we were to select six stimulation contacts to stimulate from this would yield 63 different location settings for which to test. Thus, we are given $63 \times 1800 = 113,400$ possibilities to test. If each test took a second of time to process, this is equivalent to 31.5 hours, and this is only testing the parameter range that has been experimentally determined. If the testing ranges of the parameters are expanded, we can see that there is a need for an algorithm to expedite the process of finding the optimal parameters as opposed to testing every parameter over a long period of time.

One consideration that must be taken into account is the charge density delivered by the electrode [60]. The recommended maximum value for the charge density is 30 microcoulombs per cm$^2$. This safety consideration implies that charge densities above the specified value cause damage to the brain tissue. Furthermore, the charge density during DBS is directly related to the voltage and pulse-width of the stimulation pulses.

$$\text{CHARGEDENSITY} = \frac{(V \times PW)}{\text{IMPEDANCE} \times \text{AREA}}$$  \hspace{1cm} (3.4)

In the above equation the charge density is directly proportional to the product of the stimulation voltage (amplitude) and the pulse-width of the waveform. The charge density is inversely proportional to the characteristics of the electrode including the impedance of the electrode and the geometric surface area of the electrode. Typically the
impedance is about 500 ohms and the surface area of the electrode is .06 cm\(^2\). This leads to the Voltage/Pulse-Width restriction for DBS in equation 3.5 below:

\[ V \times PW < 900 \mu\text{C} \Omega \]

Equation 3.5 clearly places an upper limit on the product of the voltage and pulse-width during DBS. As long as the product is kept below 900 microcoulomb*ohms then that parameter can be used for stimulation. Any other combination that does not satisfy equation 3.5 must be disregarded for the safety of the patient.

The one aspect that DBS research is lacking is the gathering of data relating all different combinations of frequency, amplitude, pulse-width, and tremor response. A primary reason for this is that the underlying nature of why DBS actually works is still uncertain, thus creating a model based on physical parameters is difficult. One group that attempted to produce a model was that of Kuncel and Grille [61]. Although a definitive model was not produced, their model could be used to assist doctors finding the optimal parameter settings. Although it was clear that the expected response from patient to patient was not the same, there were some trends that were noticed that could be used in the development of an ADBS system. The group used nine patients that exhibited essential tremor and performed DBS on 14 of their thalami while varying the parameters of frequency, pulse-width, and amplitude. It was found that low frequency stimulation aggravated tremor, and that this effect increased when the voltage was increased. It was also found that high frequency stimulation had a U-shaped relation to voltage. When the optimal voltage was set, tremor was optimally reduced, but any increases beyond that voltage increased tremor. A hypothesis was made in order to model the process: the authors believed that tremor response to DBS resulted from two “competing processes”.\]
With this hypothesis the relationship of tremor to voltage and frequency of stimulation were mathematically modeled. It was found that the optimum voltage varies across patients, and that this is related to electrode position. The authors initially concluded that the primary parameters that affected tremor were voltage and frequency, and that pulse-width was least important. The model that they developed was a nonlinear model of VIM DBS. Using the hypothesis of the two “competing processes” for VIM DBS, the following non-linear mathematical model of DBS was presented by Kuncel [61]:

\[ T = a \left[ 1 - e^{-\beta_1} + \frac{\alpha_1 - \beta_1}{1 + e^{\alpha_1 f - w}} \right] - b \left[ 1 - e^{-\beta_2} + \frac{\alpha_2 - \beta_2}{1 + e^{\alpha_2 f - w}} \right] \]

(3.6)

In the above equation there are three variables, \( v \) (stimulation voltage), \( f \) (stimulation frequency), and \( \beta_1 \). The range of values that were experimentally observed in table 3.1 completes the model that was presented in equation 3.6. Observing the wide range of the values in table 3.1 can make one skeptical to the legitimacy of such a model, although it was a fine initial effort.

**Table 3.1: Optimal Starting Parameters for Kuncel’s model [61]**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Estimate</th>
<th>Lower Confidence Limit</th>
<th>Upper Confidence Limit</th>
</tr>
</thead>
<tbody>
<tr>
<td>( a )</td>
<td>24.9</td>
<td>12.6</td>
<td>34.9</td>
</tr>
<tr>
<td>( b )</td>
<td>6.11</td>
<td>3.26</td>
<td>11.8</td>
</tr>
<tr>
<td>( a_1 )</td>
<td>0.00288</td>
<td>0.0009</td>
<td>.0069</td>
</tr>
<tr>
<td>( a_2 )</td>
<td>-.00456</td>
<td>-.0281</td>
<td>.018</td>
</tr>
<tr>
<td>( \beta_2 )</td>
<td>0.186</td>
<td>-.532</td>
<td>.615</td>
</tr>
<tr>
<td>( s )</td>
<td>0.0236</td>
<td>-6.46</td>
<td>5.516</td>
</tr>
<tr>
<td>( w )</td>
<td>94.7</td>
<td>63.0</td>
<td>134</td>
</tr>
</tbody>
</table>

With these initial estimates, the model developed by the researchers was able to match some of the experimental data. It is clear that their model has some severe limitations, although it is a good first step toward modeling DBS based on data recorded.
The limitations include the following:

1. It is not a complete model of physical processes. The values of parameters have been fit to achieve a match to the experimental data rather than measured physical quantities like distance, electrical impedance, etc.
2. The model does not address effects of varying the active contact on the electrode used for stimulation.
3. To predict the optimal voltage for each thalamus the model needs an estimate of the parameter $\beta_1$ for that thalamus which needs at a minimum tremor measurements at low stimulation frequency.
4. The model accounts for less than 100% of the variance of the data. As a matter of fact, it seems to be accurate to the data far less than 100% of the time (more along the lines of 46% of the variance).

When attempting to use a model for a control application, this success rate is unacceptable, especially for a medical device implanted in humans. However, this model can serve as a rough starting point for a more flexible algorithm such as ALOPEX [62] [88] to find the optimal parameters.

3.3 Research Platform

When developing a medical device there are two main parts of the design that must interact together to provide a complete product. These two parts of the design are the mechanical design, and the electronics controlling the mechanism. Furthermore, the electronics in the medical device can be subdivided into two separate areas of development. These two areas are the hardware design as well as the software design. With the rapid growth in the semiconductor industry over the past ten years, many
portions of designs that were done in hardware can be substituted with software. With powerful microcontrollers capable of over a hundred million instructions per second, algorithms that were only theoretically viable are being implemented in practice. The overall embedded system design that will serve as the testing device for researchers in the CIL laboratory (Rutgers) is shown in Figure 3.8.

![Figure 3.8: ADBS System Overview [99]](image)

The mechanical portion of the design is depicted simply in the block labeled “electrode”. The design of the pre-amplifier and the pre-filter is also included within the parallel thesis that designs the electrode for this project [78]. Besides the aforementioned components of the system, only minor circuitry is required besides the items in the box labeled as dsPIC30F2011. The dsPIC30F2011 is a microcontroller manufactured by Microchip Corporation that offers digital signal processing capability in a small package [79]. The sub-blocks within the box labeled as dsPIC30F2011 are all operations done in software by the microcontroller. Due to the computationally intensive nature of the sub-blocks within the processor block it is understandable that the processor selection is crucial when developing a robust device.
3.3.1 Hardware Tools

The primary hardware tools that are used for the development of the board necessary for testing in a laboratory environment are primarily integrated circuits (IC’s). As mentioned before, Microchip’s line of digital signal processors is the core of the design. This line of chips is used because the main processor must be capable of supporting digital signal processing algorithms as well as performing the analog to digital conversions from the electrode. When used in conjunction with standard operational amplifiers, resistors, and capacitors a fully functional board can be developed for testing on animals the algorithms developed. However, the line of microcontrollers from Microchip is capable of consuming power on the order of nano-watts. For this reason, these chips can even be considered for an actual device that is implanted in human subjects. Furthermore, the programming of the microcontrollers is easily done using Microchip’s ICD3 Debugger/Programmer. This tool produced by Microchip is capable of running the microcontroller a single instruction at a time in order to gain more control of the debugging.

As previously stated, the microcontroller selected as the main processor was the dsPIC30F2011 (Figure 3.9) manufactured by Microchip Corporation. This high-performance, 16-bit digital signal controller is capable of being programmed in C with 24-bit wide instructions, however the datapath is 16-bits wide. With an internal phase locked loop (PLL) available the chip is capable of up to 30 MIPS (millions of instructions per second), which is ample processing power for our application. Furthermore, the digital signal processing (DSP) features of the chip include a dual data fetch, two 40-bit wide accumulators, a 17-bit x 17-bit single-cycle multiplier, as well as the capability to
execute all DSP instructions in a single clock cycle. With the use of timers, the 16-bit Pulse Width Modulation (PWM) feature can provide the digital to analog conversion (DAC) used as the stimulating output to the electrode. Furthermore, there is a high resolution 12-bit ADC with a 200 ksps conversion rate and up to 10 input channels. Lastly, using Microchip's patented “nano-watt” power consumption, this chip can be used in a wide variety of low power applications.

The different components of the overall system are as follows. The electrode that will need to be designed will have external circuitry to handle the amplification as well as analog filtering [78]. This analog circuitry will be connected to the main processor (dsPIC30F2011) where all of the processing of signals will take place. This chip is located in the casing of the stimulation device. For the purposes of research, the immense processing power of Microchip’s line of products is not needed for the DBS system. A more basic chip can be developed in VLSI that satisfies the requirements of the ADBS device. Furthermore, not all of the attributes of the dsPIC must be used in an actual application, and the power consumed can still remain low, therefore it can still be considered for in vitro testing.
3.3.2 Software Tools

The software used for the development of the ADBS system is standard in academia. For all simulations of algorithms, Matlab was called upon to model the processes before algorithms were tested using the microcontroller. The development of all firmware for the microcontrollers was done using the programming language C. When the programs written in C are compiled, the assembly language disassembly is also given using Microchip’s development environment named MPLAB IDE. The advantage of such a tool is that the processor that is developed in VLSI can be developed to have the same instruction set as those used by MPLAB IDE (which is the standard set of instructions). The final piece of software that is essential for research is that of Cadence. Using this software dedicated for VLSI, most processors can be custom designed to suit the needs of the developer. VERILOG is the programming language that is used to develop hardware. Not only can digital circuitry be designed, but also it can be simulated and laid out for actual production. Using this powerful set of tools, the archaic Deep Brain Stimulation Systems can be automated to provide better treatment to ailing patients.

In addition to this software, dsPICfdLite (Microchip Corporation) was purchased in order to design the digital filter used for the dsPIC. Furthermore, dsPICworks (Microchip Corporation) was used to generate simulated inputs such as simulated LFP’s that can be used to test the code. The basic operating cycle of this device is described in Figure 3.10. It can be seen that the major selection that the user must make is between the Automated DBS mode or the manual mode where the user chooses a certain waveform. There is also a mode that will allow the technician or medical professional to observe the
brain activity as well as check the electrode-brain impedance. One of the key reasons systems shy away from sensing brain activity is the fact that the scarring of the tissue causes the impedance of the contact-brain interface to rise over time, not allowing accurate LFP information to be detected. However, there are things that can be done to counteract this. Even if the sensing capability of the device degrades over time due to the change in the impedance, the ability to sense the brain activity as the device is being lowered into the base of the brain is a major improvement over the current system.

Currently, a test probe that senses LFP data are lowered into the brain in order to locate the exact site where implantation will take place based on the highest LFP activity. This is then removed and the actual stimulating electrode implanted. In this process, it is certain that the stimulating electrode was not placed in the same exact position as the sensing electrode. If the new design were used, a single electrode both stimulates and senses, therefore the positioning only has to be performed one time, and the intended site of stimulation will be the actual site for stimulation because the electrode will not be moving once correctly dropped into place.

The main blocks of the code written for the dsPIC include a portion of code to convert the analog signals to digital signals (after an analog anti-aliasing pre-filter), code to account for the correct pulse-width modulation in order to produce an analog output, code to digitally filter the incoming data, as well as code for power estimation of the incoming signal. Based on the amount of power (the squared amplitude of the signal) we can arbitrate between the different contacts as to which one to isolate for pulsation. The ALOPEX algorithm is run for a set amount of iterations which can be altered in order to compromise between speed and accuracy.
Figure 3.10: Flowchart of ADBS System
The three parameters that will be changing in the program are the pulsation frequency, pulse amplitude, as well as the pulse-width (Figure 3.10). The algorithm is designed to optimize the required values for these three parameters in order to obtain the desired response (maximum reduction in tremor). Utilizing internal timers and their interrupt capabilities PWM cycles can be setup in order to achieve a desired DBS pulse amplitude. There is a distinction between the pulse train coming from the PWM output and that of the DBS system. The principle of pulse-width modulation works on varying the duty cycle of a waveform with set amplitude in order to achieve an analog value between zero and the amplitude used. In our application, the $V_{dd}$ of the processor is 3.6 volts, and for this reason our PWM operation can only produce voltages between 0 and 3.6 volts. Our PWM uses 8 bit resolution, therefore we are able to resolve voltages down to $(3.6 \, \text{V}/(2^8)) = 14 \, \text{mV}$. Considering the technician can only raise the voltage in 100 mV increments this is adequate resolution. The pulses that are desired for DBS will be composed of the pulses from the PWM module when the DBS pulse is desired to be high. Turning off the PWM module turns off the DBS pulse. Therefore, our program produces a pulse train of pulse trains alternatively turning on and off.

The digital filtering operation was constructed using dsPIC\textsuperscript{fdLite} as previously stated. After the specifications for the filter are entered (filter type, tap lengths, etc), the graphs showing the magnitude plots over time as well as the frequency response of the filter are constructed. After the specifications of the digital filter were determined according to our filtering needs, the option was given whether to construct code in C or assembly. Considering the code consisted of calculated filter taps for the FIR (Finite Impulse Response) filter, assembly code is suitable. These filter taps are stored in a
separate file and called upon using DSP instructions that are found in libraries attached to
the code. This filtered data are now “clean” and in digital form. It is ready to be further
processed using unique algorithms in order to make DBS adaptive. When choosing the
type of digital filter to design another option would have been to use an IIR filter (Infinite
Impulse Response filter). The filtering equation that is used is similar to the FIR (finite
impulse response) filter case. When calculating the filtered response for an FIR filter
equation 3.7 is used [80].

\[
y(n) = \sum_{m=0}^{M} h(m)x(n-m)
\]  

(3.7)

The equation for the IIR (infinite impulse response) filter is only different in the
upper limit of the summation. Instead of the terms being summed in a finite manner to M,
they are summed to infinity (equation 3.8) [80].

\[
y(n) = \sum_{m=0}^{\infty} h(m)x(n-m)
\]  

(3.8)

The filter design problem is that of constructing a transfer function of a filter that
meets prescribed frequency response specifications. For the case where an FIR filter is
used, the two main advantages are the property of linear phase response as well as
guaranteed stability [80]. The tradeoff for these two advantages is that there could be a
high computational cost. The two main advantages of the IIR filter design are the low
computational cost as well as the efficiency when cascading filters. The disadvantages of
the filter include the potential for instability as well as the fact that linear phase can only
be approximated over the passband. For these reasons the FIR filter was chosen because
the computational cost of the filter was not found to be overwhelming for the processor
being used.
When viewing the Magnitude vs. Frequency plot for the digital FIR filter constructed the pertinent factors to keep in mind are the frequencies that are passed by the filter as well as the passband/stopband ripples. As can be seen in the graph, the passband frequency of 40 Hz allows for minute attenuation of the input signal up to that value. This value was chosen because the information that is pertinent within the signal is less than 35 Hz. The value of 40 Hz was chosen to allow for a buffer considering the variability of the incoming signals from different patients. Furthermore, the frequency information received will not be completely neglected up to the stopband frequency at a
value of 150 Hz. Anything above this value is of little importance to ADBS, and thus it is attenuated heavily. The length of this FIR filter was 19 taps, which was designed to be shorter allowing for less delay (as can be seen in Figure 3.12).

![Figure 3.12: Group Delay of FIR Lowpass Filter used in ADBS](image)

When examining the graph of the group delay of the filter, the delay is the same across all frequencies and it is 18 milliseconds. This factor is most pertinent when actually running the chip in real-time because for the first 18 seconds of data that is filtered, the output should be neglected. This delay was expected based on known DSP principles, primarily that the delay is a function of the number of taps and the sampling
frequency (equation 3.9). In figures 3.11 and 3.12, $F_s$ represents the sampling frequency, in the case of the ADBS processor this is the sampling frequency of the Analog to Digital Converter [80].

$$delay = \frac{(N - 1)}{2 * F_s}$$

(3.9)

The above simulations of the filter were done with a filter tap length of 19 and a sampling frequency of 500 Hz. These numbers yield a result of 18 milliseconds as shown in Figure 3.12. This group delay can further be seen in the plots of the impulse response and step response for this particular filter (figures A.3.1 and A.4.1). If a one second sample is taken at a time, this yields 18 milliseconds of unusable data from the start (or less than 2 percent of the signal). It can be seen that as the number of taps increases to cater for a more stringent filter (and better filter response), the group delay increases in a linear fashion. For this reason the designer of a filter must decide how stringent the filter must be based on the requirements of the application. The key metrics discussed in this section with regard to the digital filter are representative of what will be used when sensing biosignals. As stated within this section, the pertinent data are found within the lowest 35 Hz of the signal, and the slow nature of this biological signal allows for a sampling frequency of 500 Hz to be adequate. As shown in Figure 3.12, these key metrics yield a negligible delay time which is easily counteracted within the algorithm running on the microcontroller.
4.1 ALOPEX Simulations

Due to the variability and uncertainty associated with modeling the brain, ALOPEX was deemed the optimal approach for efficiently traversing the range of stimulation parameters in order to determine the optimal stimulation setting for the patient. Primarily used in conjunction with neural networks and pattern recognition, the goal was to first establish ALOPEX as an efficient optimization technique. Furthermore, it was necessary to observe the nature of the convergence associated with the algorithm in order to determine the proper method to use when the algorithm is used for an ADBS system. Initially, a simple simulation was performed using ALOPEX on a one-dimensional optimization problem. The task at hand was to observe the accuracy of the algorithm when finding the local minimum of the equation \( y = x^2 + 4 \). From this simulation the complexity of the equations was increased incrementally to ensure ALOPEX was in fact optimizing the parameters as desired. The goal in the end was to establish ALOPEX as an effective optimization technique for generic multidimensional functions. For a multi-dimensional optimization problem that would be encountered during ADBS the effects of ALOPEX can only be approximated. For this reason the simplest optimization problem is provided first to present the accuracy desired using this optimization technique. After the first optimization problem is explained, levels of complexity are added in order to demonstrate that ALOPEX works best as an optimization technique when multiple parameters are being optimized at the same time. Although there are no live subjects (animal or human) to perform the clinical evaluation
of the algorithm, a firm procedure will be established allowing the testing in vitro/in vivo to occur in a timely manner when clinical testing is pursued.

In accordance with the ALOPEX algorithm, the variance of the additive noise $r(n)$ (equation 3.1 and Figure 3.1) is initially set at its highest value and then reduced as the number of iterations increases. As the number of iterations increases and the convergence process begins, the variance decreases steadily as a function of the number of iterations as the final result is determined. The ALOPEX algorithm is first demonstrated on the simplest of optimization problems using the equation $y = x^2 + 4$. It is known that the local extrema of $y = x^2 + 4$ is the minimum value of $y = 4$, and this is located at $x = 0$. If one were to imagine ADBS in a simplistic form then the value of $y$ represents the cost function in this example. For ADBS this implies that the value of $y$ produced from this quadratic equation is the power of the LFP signal as the value of $x$ is varied. In ADBS the parameters that are altered are frequency, pulse-width, and amplitude. For this example one can imagine that ADBS only depends on a single parameter (either frequency, pulse-width, or amplitude) and that the LFP power sensed depends only on the variation of a single parameter (i.e. frequency). The plots in Figure 4.1 show the convergence of this single parameter optimization problem. It can be seen that the convergence is stochastic, and almost violent in nature, but clearly controlled. The convergence to the correct solution is evident from Figure 4.1. The value of $x$ should converge to 0, and as can be seen below, it quickly converges to a value in the 0.1 range, which is not far from the ideal solution. For the cost function, the ideal value of convergence is 4, and the algorithm converges to a value of 4.0004. One would relate this example to a clinical trial by concluding that as the value of $x$ (any of the stimulation parameters) converges, the
minimum value of the LFP power sensed was found. In our pre-ordained example the
convergence to the correct solution is dictated by the equation that was used in the
algorithm. However, in a clinical trial the LFP response will be altered in a distinct
manner as each stimulation parameter is adjusted in the algorithm. It is this LFP response
that is dictated by the DBS response of the patient. The exact model or equation which
governs DBS is not known, however using this technique (as opposed to a control
algorithm) the knowledge of the model is less important.

Although this is an example of optimizing a function of only one variable, it is the
same concept that is used to optimize a function of three or more variables. It can be
expected that the convergence will be stochastic in nature, demonstrating a “random
walk” as it eventually finds the optimal parameter settings for amplitude, pulse-width,
and frequency. As can be seen from the plots, the movement in the convergence is
initially large. Furthermore, the convergence is extremely fast without even accounting
for the ability of the algorithm to work in parallel on many parameters at once. However,
the slightest difference in the initial tuning of the algorithm has a large effect on the
convergence abilities of the algorithm. For a dedicated application (such as a Deep Brain
Stimulator) the initial tuning would encompass the testing of the algorithm on multiple
patients in order to evaluate whether a general set of initial tuning parameters apply to all
patients or whether it must be fine tuned on an individual patient basis. In either case, for
a person experienced with running ALOPEX simulations, the tuning should not take
more than a couple of minutes to perform, and these settings would last the remainder of
the life of the device.
From examining the nature of the convergence in Figure 4.1 it is clear that the step size (amount of fluctuation of the cost function) per iteration initially is large allowing for the algorithm to find the general range of the solution quickly. Midway through the iteration cycle the step size is reduced considerably allowing for fine-tuning and steady convergence to the optimal result. The self-adjusting nature of the algorithm can be seen from Figure 4.1 as the cost function reaches the optimal parameter multiple times before eventually resting at the optimal solution. This movement is what allows the algorithm to avoid local minima that are not the global minimum that is desired. The speed of the procedure is dependent on the number of iterations as well as the processing speed of the chip being used to run the algorithm. However, as can be seen from the above example, the iterations are on the order of thousands, providing little processing challenge to any microcontroller found on the market today. Even if one were to select a processor with a processing frequency of the order of tens of kilohertz (slow), thousands of iterations would be computed in a fraction of a second.
The goal of the processor is to digitize the analog signal taken from the electrodes and minimize the power of the undesired signal using ALOPEX. As previously described, ALOPEX uses local correlations between changes in individual weights and changes in the global error measure. Its goal is to solve the optimization problem of minimizing the error with respect to network weights. Using ALOPEX, the optimal DBS settings can be programmed without the use of trial and error, leading to a DBS system that is less prone to human error. As previously discussed, the model of DBS presented by Kuncel and Grill [61] (equation 3.6) is far from a fully accurate model (for reasons outlined in section 3.2), however it is possible for this model and future rough models to have practical considerations for this application. Using these models to find the rough optimal stimulation parameters, these points can be used to limit the range of the parameters that are traversed by ALOPEX when being used in the laboratory setting. This technique can be used by future researchers to reduce the time it takes for ALOPEX to find the optimal stimulation parameters. These values can be programmed into the microcontroller prior to use on the patient to assist the algorithm when trying to find the optimal stimulation parameters for each patient. However, as can be seen from the parameter ranges in Table 3.1 pertaining to equation 3.6 (Kuncel’s model), the model accounts for high variability from patient to patient. In essence, the doctor or health practitioner responsible for the commissioning of the ADBS device would then roughly categorize the patient under the Kuncel model first by specifying specific values instead of ranges for variables in Table 3.1. Once these values are entered, the ADBS microprocessor could limit the range of stimulation values for example limiting the possible stimulation voltages to a range of 1 volt to 2 volts instead of the entire range if
the model estimated the optimal stimulation voltage to be 1.5 volts. Instead of having ALOPEX traverse the entire possible range of parameter values the commissioning personnel can use the model to direct ALOPEX to a range that would allow for quick optimization of the parameters as long as the Kuncel model predicts correct values. However, if the researcher’s model is proven to be ineffective for the majority of patients in a clinical setting, then ALOPEX will be left to operate in its traditional form (initially selecting a random stimulation parameter to start the algorithm). The idea is that this ADBS chip can be used by other researchers in conjunction with their own proposed models in their own research goals of finding a more accurate model of DBS. The advantage of using ALOPEX is the ability of the algorithm to optimize multiple parameters in a parallel fashion. This can be exploited by researchers that are testing their models which are dependent on a multitude of parameters. All of the parameters can be entered into the algorithm in order to assist the researcher when determining the accuracy of their own model. Modeling the brain is of little consequence to this research due to the fact that ALOPEX circumvents the need to have a model in order to operate.

Figure 4.2: Top View of 3D Gaussian Model (left) and Side View of 3D Gaussian Model (right)
A demonstration of the effectiveness of ALOPEX to optimize two parameters uses the model in Figure 4.2. When examining the model from a top-view it is apparent that the maximum is located at an \( x \)-value of zero as well as a \( y \)-value of zero. It is clear that over the entire span of the function, this maximum value is located on a small surface area of the entire function. As previously discussed, ALOPEX is able to locate the extrema of a multi-dimensional function in a timely manner, however it does not produce the exact answer. This 3D Gaussian model is a good representation of a localized area within the individual function that each patient undergoing DBS possesses. This function is dependent on the frequency, pulse-width, and amplitude of the DBS signal. In this three-dimensional model it can be imagined that one of the parameters is neglected (for instance pulse-width). As a matter of fact, in the model for DBS presented by Kuncel and Grille [61] the pulse-width is in fact neglected. The function presented is only dependent on the amplitude of the signal as well as the frequency. When viewing the actual tracking of the parameters that are optimized, it can be seen that the \( x \) and \( y \) values converge to a value that is close to the actual optimized values. These convergence values (Figure 4.3) are located near the point depicted in Figure 4.2 \((X,Y,Z = -1, 1, 1.698)\).

![Figure 4.3: Graph of ALOPEX tracking the optimal \( x \) in a 3D Gaussian Function (left) and Graph of ALOPEX tracking the optimal \( y \) in a 3D Gaussian Function (right)](image)
For both the optimal $x$ and optimal $y$ tracking done by ALOPEX the movement from the initial number dictated by the random number generator is large. The algorithm actually is near the final resting value when tracking after less than 500 iterations (almost an immediate response on the order of a fraction of a second when running on any modern processor), however the fine adjustments that proceed afterwards help to obtain a solution that is even more accurate. Although ALOPEX was allowed to run for 10,000 iterations it is clear in this example that little change occurs after 4000 iterations. Running on any processor with a clock speed of over 1 MHz allows ALOPEX to be run in well under a second. It could be assumed that for functions that are more simplistic than others that the full 10,000 iterations can be reduced by as much as an order of magnitude to 1,000 iterations, however when being applied to an unknown system it is best to leave the number of iterations higher than what is found to be clinically acceptable. The clinically acceptable number of iterations can be determined as follows. Given a sample population of patients willing to have the ADBS device implanted, the processor can be programmed to find the lowest number of iterations needed to be within 1% of the resting value of the algorithm. In Figure 4.4 this number of iterations is roughly around 4000. Although the algorithm will be allowed to run a larger number of iterations for each patient, the actual number of iterations needed will be saved for each patient. The stopping criteria of the algorithm can either be a set number of iterations or it can be ran until the fluctuation of the cost function is within a desired threshold. Based on the results of the entire population the 99% confidence interval can be determined for each parameter. A factor of safety can be used for each of these intervals to ensure that the vast
majority of patients will be able to have their parameters converge to the correct values. Although it may be found that half the population only needs 4000 iterations to converge to a solution, if the other half of the population needed 8000 iterations then the default setting from the factory would allow for well over 10,000 iterations. These numbers are hypothetical and would need to be tested clinically before the correct number of iterations are determined. Once these ranges are clinically established, optimization can be done by the processor on an individual basis autonomously using the same principle. The processor can keep track of the actual number of iterations necessary to be within 1% of the final resting value, and over time increase or decrease the number of iterations necessary in the algorithm. This adjustment to the method in which ALOPEX is used can save an exorbitant amount of time for certain patients (and not as much time for others that will require the full number of allowed iterations).

![Graph of ALOPEX tracking the optimal cost function (3D Gaussian Function)](image)

**Figure 4.4: Graph of ALOPEX tracking the optimal cost function (3D Gaussian Function)**

As the optimal cost function is tracked it is expected that the convergence would mimic that of the parameters that are optimized since it is a function of those parameters. As
expected it can be seen in Figure 4.4 that initially there is a large jump in the value of the cost function, however by iteration 4000 the final solution is almost reached. It is important to understand that ALOPEX yields different results every time that it is ran, however these results are always close to the optimum (the solution converges in the same region on a consistent basis). A second simulation using the function from Figure 4.2 can be found in Appendix A4. It can be observed that similar results are obtained, and this is attributed to the random number generator. As can be seen from these figures a good approximation is still produced using ALOPEX. As discussed in section 3.1.1, the Gaussian noise is reduced as the number of iterations increases. As a result of this, the portions of the ALOPEX equation that are functions of the decaying Gaussian noise will exhibit a decaying shape (Appendix A4.1-A4.4). It can be expected that every time ALOPEX is executed the “optimized” value that is found will be different each time. If Figure A4.5 is examined it can be seen that as ALOPEX is used to track the optimal $x$-value the convergence is of a similar shape to the convergence in Figure 4.3, however a different result is obtained (roughly $-0.6$). The optimal $y$-value that is reached in Figure A4.6 is roughly 0.4, and the cost function converges to a value of roughly 1.91 (Figure A4.7) after about 4000 iterations. Clearly on these iterations of ALOPEX a more accurate solution was found, and it is a testament to the variability of the algorithm. Such convergence is analog to observing the path a liquid takes as it is directed through a funnel. Due to the gradient effect produced by the shape of the funnel and the force of gravity, when liquid is poured into the funnel, the path the liquid takes to the bottom of the funnel may be different each time the liquid is poured. However, the liquid generally reaches the same resting point each time it is poured through the funnel. The path that the
parameters traverse as ALOPEX runs is analogously different each time, however due to
the gradient nature of the problem, the algorithm is designed to allow for the final resting
values of the parameters to reach the optimal values desired.

The previously discussed model of DBS presented by Kuncel and Grille [61]
(equation 3.6) is plotted in Figure 4.5. Such a function was not expected due to the lack
of resolution with respect to local maxima and minima. It was quickly determined that
this model was a general depiction, and its purpose was not to be accurate, but to give a
general fitting to different curves observed from data from different patients. The
parameter $\beta_1$ (Table 3.1) is the parameter that is adjusted to account for the variability
between different patients. When this value was adjusted across a large range, the only
difference that was noticed was the change in the convexity of the curve produced by the
model, however the lack of local extrema was evident. What can be implied from this
model is that the tremor response of a patient can follow a smooth trajectory as
parameters are adjusted to optimal values. Perhaps this is the only thing that can be taken
from the model presented by Kuncel’s group. Although this is clearly not an accurate
model due to the lack of resolution with respect to optimal parameters, the fact that their
data confirms the assumption that a smooth trajectory is followed as opposed to erratic
movement (errant spikes) demonstrates that an optimization strategy can be carried out
with respect to DBS parameter optimization. The verification of such a conclusion is
beyond the scope of this research (it would require the modeling of the tremor response
of a large population of patients).
However, running ALOPEX on such a curve is not an unnecessary task given the fact that the reality of DBS could mimic this curve. If given such a set of data it is important to examine how ALOPEX behaves. As expected, ALOPEX quickly proceeded to push the parameters in the direction of largest difference (on the curve). On the slope shown in Figure 4.5 this would imply the parameters of voltage to increase quickly as well as the parameter of frequency to oscillate depending on the accuracy of the values entered (in practice).
Referring to Figure 4.6, as expected the voltage was quickly increased to near the maximum value of 5V, however the unexpected occurred with the frequency response. Immediately a frequency for stimulation of roughly 142 Hz was locked on and maintained throughout the iterations. In Figure 4.7 the tremor response started negative and continued to decrease violently until it settled at roughly 4000 iterations. The tremor was measured as a relative term, the higher the value the more tremor was observed by the device used to measure it on the patients. The above example was shown to demonstrate ALOPEX’s response to less than ideal conditions. Although this exercise revealed the nature of the algorithm, it did not suffice to show how ALOPEX would deal with multiple local extrema and many parameters.

![Figure 4.7: ALOPEX finding the minimum tremor response for a specific patient](image)

The function that was selected to show the effectiveness of ALOPEX is shown in Figure 4.8. From the top perspective (left) it is evident that there are four local extrema generated by this three-dimensional plot. As shown by the side-view of the function (right) two of the extrema are maxima and two of them are minima. When dealing with
such a situation one can be unsure as to how ALOPEX will manage to produce accurate results. Such a situation is dependent on the cost function itself, and cannot be avoided by the researcher if it is a characteristic of the cost function.

![Figure 4.8: Top-View of a function of two variables (left) and Side-View of a function of two variables (right)](image)

Multiple simulations were ran in order to determine the nature of the algorithm with respect to a function of this nature. In figures 4.9 and 4.10 the optimal parameters are tracked alongside the cost function. It can be seen that the random number generator starts at roughly −1.7 for the value of $x$ and −0.8 for the value of $y$. ALOPEX allows the value of $x$ to converge to roughly −1.05 and -0.85 for the $y$-value. When examining Figure 4.8 (left) this places the convergence point of the algorithm at the upper left hand circle nearly at the center. When translating this point to the plot of Figure 4.8 (right) it is clear that this is nearly the maximum value of the function (the optimal value of $x$ is -1.05 and the optimal value of $y$ is -0.85). The parameters within the ALOPEX algorithm are what dictated the convergence to this point. It is those parameters that must be adjusted by the researcher to allow ALOPEX to converge to the desired optimization point.
The cost function in Figure 4.10 shows convergence around 0.14, which is near the actual max value of the function. Over multiple simulations it was realized that the starting point of the algorithm can dictate the manner of the convergence. However, this could be adjusted by increasing the allowable fluctuation per iteration, not allowing the solution to converge quickly. The predictability of the algorithm would then be sacrificed, which is highly undesirable. Having four distinct extrema along a single function is unlikely, however if this were the case, the algorithm can be ran multiple times to provide all of the extrema over time. Another solution to an optimization problem such as this is to run multiple simulations and average the results. It is important to recognize that inherent to the ALOPEX algorithm is the fact that local extrema are avoided due to the addition of noise in the algorithm. If the algorithm were to temporarily converge at a local extremum the noise from the subsequent iterations would work to move the convergence away from that point and toward the global optimum value. In addition, the researcher has the ability to observe the convergence as it is occurring, which allows the researcher to tweak the
parameters within a particular run if the researcher notices the algorithm converging to a
local minimum.

Figure 4.10: Graph of ALOPEX tracking the optimal cost function (Function of 2 Variables)

A second example of ALOPEX converging to a different extremum is found in
appendix A5. This is a good example of a mediocre convergence of the algorithm where
parameters of the algorithm are not finely tuned. The initial starting values were
approximately –1.5 and 2.0 for x and y respectively which is one of the farthest points
from the four extrema on the map. For this reason, the algorithm eventually came close to
the area of the extrema, but was not as precise as one would like. The exact application
being used for ALOPEX dictates the parameters within it, allowing for predictable
convergence. Once again, this stresses the versatility of ALOPEX, and the need for
clinical testing with respect to the algorithm in order to determine the values of various
parts of the algorithm such as the maximum step size the parameter can increase/decrease
from iteration to iteration.

From the above results of the ALOPEX runs, the nature of the convergence is
better understood. Therefore, demonstrating the optimization of a function of 3 variables
serves little purpose. However, one claim that has been repeated yet not demonstrated is the ability of ALOPEX to optimize multiple parameters in parallel. Based on the nature of the algorithm itself, it makes little difference whether 100 parameters are optimized or 10. The convergence time is the only aspect that would differ when running ALOPEX on multiple parameters. ALOPEX was run on a multi-dimensional function (function of 6 variables) in order to show the effectiveness of the technique. Although graphically such a function is difficult to represent, the convergence graphs of the algorithm can still be shown. For simplicity, the function that was chosen is shown in equation 4.1.

\[ Q = x^2 + y^2 + z^2 + a^2 + b^2 + c^2 + 4 \]  (4.1)

Even in this multidimensional function it is trivial to recognize the minimum point is located at \( x = y = z = a = b = c = 0 \). This multidimensional quadratic function has a value of 4 when all of the parameters of the function are equal to zero. In a quadratic fashion, if any other values are substituted into the input variables the value of the function increases. Therefore, the goal of ALOPEX in this example would be to either optimize the input variables to reach the global minimum or maximum point dictated by the range of the function that is selected.

For instance, if \(|x| \leq 1, |y| \leq 1, |z| \leq 1, |a| \leq 1, |b| \leq 1, \) and \(|c| \leq 1\), the global maximum is the value of 10. Regardless of which points we are attempting to optimize, the input parameters in order to reach the desired extrema, ALOPEX can be adjusted to reach these points. As with the earlier examples, ALOPEX will be ran in order to find the global minima at the point \( Q(0,0,0,0,0,0) \). Due to the high dimensionality of the equation being optimized, the multiple plots of ALOPEX optimizing the parameters are in Appendix A8. It has already been established that the optimization algorithm can be run
for tens of thousands of iterations in order to achieve near exact results, however the plots in Appendix A8 demonstrate the near accurate results that can be achieved with multi-dimensional functions in little time (same number of iterations as lower dimension function optimization problems). Provided in Appendix A7 is the Matlab code that produced the plots in Appendix A8. It can be seen that as the complexity of the equation that is being optimized grows, the code required to optimize the equation grows in nearly a linear fashion. This observation further demonstrates the necessity of a processor that is able to process the algorithm in a timely manner for an ADBS application.

4.2 Firmware Breakdown

As is common with most microprocessors, code is executed in a sequential fashion when the dsPIC30F2011 is given power. Although the code is designed to never terminate (the microcontroller would be doing nothing if it did), the code executes until the end of the program and then stops. This is unlike many other controller systems such as programmable logic controllers that continuously loop through the code for an infinite amount of time constantly scanning inputs and delegating which outputs to turn on. This concept of infinite surveillance of the system (the brain tissue) must be inherent in the structure of the code. As was done with the design of the hardware, a state machine was designed for the flow of the software. Structuring the code in this manner ensured that the processor was in a recognizable state at all times, and could be manipulated reliably and predictably. However, if the code methodically dictates that the processor must be in certain states at certain times, how could the processor handle outside requests that affect the activity of the processor? The solution to this common need is the accommodation of interrupts in the hardware. Since the interrupt signal is inherently an asynchronous signal,
there must be hardware internal to the microcontroller that allows a subroutine to be run once an interrupt has been decoded. In practice, the processor is forced to save its current state of execution and begin the execution of an interrupt handler. Interrupts are invaluable in real-time computing and essential to the successful operation of the code in general. Interrupts can be both from software (an instruction within the instruction set), or hardware as previously mentioned.

Once the structure of the code is known, the next factor to consider when writing software is the instruction set itself. The dsPIC30F2011 has a dedicated instruction set with special instructions dedicated to digital signal processing (DSP). Extra variable types must be understood for successful execution of any algorithm due to the expanded instruction set. The one variable type that is unique to standard variable types in C is the “fractional” data type. The need for an extra data type became apparent during DSP operations where overflow was an imminent threat to the correct execution of the algorithm. If standard data types were used such as “integer” or “float” there is always the risk of overflow, even with the dedicated 40 bit accumulators that are available. If a million iterations are executed of an algorithm with a single multiplication of any minute number greater than one, overflow is a real threat. For this reason fractional format is the format of choice for DSP operations. Using the 1.15 fractional type (or Q15 as it is also known), the range of values is from –1 (0x8000) to 0.999969482 (0x7FFF). Once the nuances of the correct data types are known the hardware of the digital signal processors can be fully utilized.

Due to environment noise or even hardware malfunction, at times the microcontroller does not perform as it should. To avoid occurrences such as freezing
(which would be catastrophic for an implanted device) safeguards are built into the hardware such as the watchdog timer. However, even in software the programmer can implement safeguards in the form of traps, which can keep the execution of code running in the event of unexpected occurrences. The primary purpose is to scan the registers for states that should not occur, and if one of these states is observed, clear the register and continue execution of code. This can be done with the use of interrupts where a priority level is assigned to a certain section of code which is executed when certain registers toggle. For example, when the processor enters an unknown state certain registers will flag the program counter to execute a user defined section of code that can alert the user of the faulty state of the processor. Another crucial factor that must be kept in consideration is the memory management with respect to the processor. Many debugging errors were found to originate from a lack of available memory during execution of code. Careful and clever memory management is critical when developing software for microcontrollers.

The primary theory involved with the software being written for the ADBS device involves many different subjects. When referring to “software”, this is actually the firmware being written for use on a microcontroller. The bulk of the software concentrates solely on the programming of the hardware in order to operate the device. The main portions of code include the following modules:

1. Analog to Digital Conversion (ADC)
2. ALOPEX Algorithm
3. Band-pass filtering with windowing
4. Serial Peripheral Interface Bus code (SPI communication algorithm between microcontrollers)

5. Stimulation (pulse-width modulation to output analog values)

6. Power Calculation/Decision Making

7. Peripheral input/output (I/O) control

Although there are numerous subroutines that are called within the main program, the following segments of code are the most crucial to the operation of the microcontroller. All the “intelligence” of the ADBS system is within the confines of the following subroutines.

(a) Light1.c

Light1.c is the main program written to control the dsPIC30F2011. It is from this main program where all of the subsequent subroutines are called. The code begins with the initialization of the hardware characteristics of the microcontroller. Perhaps the most important setting that is initialized is the speed of the oscillator that is selected. Although an external oscillator can be used with this chip that can run as fast as 80 MHz, the power consumed by a chip is a function of the operating frequency. Therefore, for this application the internal RC oscillator can be used and multiplied by different factors depending on the setting of certain registers. Using an internal phase locked loop the microcontroller is able to multiply the clock speed of the RC oscillator if needed. Certain safeguards provided with the use of a dsPIC can be configured such as the watchdog timer as well as the brown-out voltage reset. If the code starts to idle and not execute instructions the watchdog timer resets the instruction flow as well as the registers in volatile memory and the code is executed from the beginning as if the power was first
applied to the chip. The brown-out voltage reset is provided as protection so that the chip only operates within a specific voltage range. Also, in software the master clear pin can be enabled or disabled to allow the processor to be reset manually using a pushbutton or other type of method using circuitry. Manual pushbuttons are available with the lab prototype, however when the actual ADBS device is produced for patients the processor will be reset wirelessly.

The beginning of the code consists of the prototypes of all the other subsequent functions, as well as the declaration of all key data structures. Most of the data structures are standard (i.e. arrays), however the use of pointers should be noted for the filtering operations. The use of pointers greatly reduces the code necessary for filtering and power estimation. The code proceeds to call the main functions initializing them, as well as transmitting 10 bytes of data to a subsequent processor using SPI, which is done to ensure all the hardware is operating correctly. Intermittent pauses are inserted into the code to account for the timing necessary for the slave processor to respond to the master.

The remainder of the code in the main program is meant to direct the processor through the necessary order of functions that should be called based on what the user would like to accomplish (i.e. manual stimulation, ADBS, waveform recording, impedance sensing, etc.). The major functions that are called directly within light1.c are the VectorScale function as well as the FIR function. Due to the high sampling speed of the ADC, many vectors are necessary to compose a signal LFP signal. When summing the data within the vectors the danger of overflow must be accounted for. For this reason each array of values is scaled down before being summed. The FIR function is used to filter the data using the FIR filter (list of filter taps) developed using dsPICfdLite
(Microchip Corporation). Appendix A1 gives background on the theory behind the modules used to estimate the power spectrum.

(b) Handsample.c

Although the code written in handsample.c seems trivial, it is actually the control of the entire device. When the researcher using the ADBS device would like to adjust the parameters manually this only requires a change in the values of a few choice variables within this function. Besides the ability to change the stimulation characteristics, the handsample function possesses the different flags that direct the main code to perform whichever task is desired. Adbs.h contains the values of the variables that are necessary to direct the code through the different operations. If an external interface were desired it would operate through the setting of variables in this function.

(c) Adctest.c

Before the LFP data can be analyzed the analog information (LFP voltages) must be converted to the digital realm using the ADC on the microcontroller. Generally, the LFP data are slow signals with respect to other electrical signals observed in electronics, therefore the sampling rate is chosen accordingly. Referring back to section 1.1.5 which describes the frequency components of the LFP signal, it is understood that the parts of the signal that are pertinent to ADBS are below 100 Hz. Based on the frequency content of the signals, the sampling frequencies of the neural recording systems in use today are in the range of 1 kHz to 10 kHz. Even the use of 1 kHz is considered oversampling, and this gives the ADC considerable leeway when selecting the sampling frequency. The settings of the ADC are programmed in the firmware and they control the operation of the device using a series of interrupts and accompanying C code. The analog values are
set to be stored in fractional format for ease of use when calculating the attributes of the
signal. On the microcontroller itself, there are two analog reference pins that are named
$AV_{dd}$ and $AV_{ss}$. The voltage that is applied to these pins dictates the range of voltages
that the ADC can convert. Negative voltages cannot be used, only positive voltages or
zero. For example, if 3.6 V is connected to $AV_{dd}$ and 0 V is connected to $AV_{ss}$, when an
analog voltage is sensed on an analog pin, it is scaled against that voltage range. If 1.8 V
is sensed on the analog pin then in the analog buffer a value of 0.5 is recorded in
fractional format. Accordingly, if half that voltage is sensed, a value of 0.25 will be
recorded. However, the values in the buffer cannot exceed 1, and cannot be less than
zero. When performing operations this is helpful to counteract overflow since we have a
limited number of bits with which to operate.

Timing is the main issue encountered when using an ADC. The analog buffer
internal to the Microchip line of microcontrollers is limited to 16 registers. After each
sample/convert sequence, a number is placed in the buffer. It is crucial to understand the
number of clock cycles necessary to ensure proper acquisition and conversion. Once the
timing is understood with respect to the clock speed of the microcontroller, the buffer can
be used efficiently to obtain the required data points of the analog signal. The concept of
analog to digital conversion is trivial, but the details involved with its implementation
with respect to timing are far from trivial in practice.

The code within adctest.c mainly sets up the ADC to convert in the format that is
most desired. For this application, the numbers are stored in fractional format as opposed
to integer or float format. The sampling rate is also set which is a function of both the
acquisition time and the conversion time. The actual analog pin that is used is assigned in
this function as well as the interrupt code that is executed after 16 samples are obtained. When the buffer is full of data it is transferred to the array named inputSignal0 by means of pointer assignment. This interrupt has immediate priority and is executed regardless of what other code is currently running at the time the buffer for analog values is filled. However, there may be precedence for other interrupts that have higher priority over the ADC interrupt and this is dependent on the current state of the processor.

(d) **Concisealopex.c**

The automated portion of our design is based on the code within concisealopex.c (found in Appendix A9). Within concisealopex.c is the function named ALOPEX. The basic premise besides running the ALOPEX algorithm repeatedly is that the chip must stimulate an area of the brain through the electrode, and then subsequently measure the power at that site. The power measurement is the cost function within the ALOPEX algorithm. The parameters that are adjusted are the pulse-width, period, and amplitude. After a brief period of stimulation, the ADC measures the signal, the vector is scaled down, the signal is filtered, and then the power is measured. Using the ALOPEX algorithm the same process is repeated within the confines of the algorithm until the optimal parameters for pulse-width, frequency, and amplitude are reached. The typical values for stimulation are already known in advance, and they are a pulse-width of 60-210 microseconds, a frequency of 130-185 Hz, and an amplitude of 1-3.5 V. The number of iterations ALOPEX runs for can be adjusted within this function as well as the length of time needed for intermittent stimulation/sensing performed by the algorithm. The researcher can best adjust these parameters on an individual basis for different types of live subjects (i.e. rats, monkey, humans).
(e)  *Spittransfemaster.c*

The code within spittransfemaster.c is intended to establish the SPI protocol for communication with other devices. With this chip as the master, multiple slaves can be setup to interact with the ADBS chip simply by setting low a control line that is connected to the slave. Through loading and unloading the buffer at specific times, efficient and fast communication can take place between the ADBS chip and other peripheral devices including handheld devices or a PC if needed. Although the SPI driver has been written for the ADBS, a separate driver must be installed on the slave devices that are attempting to connect to the ADBS chip. Such code is used in the peripheral dsPIC30F2011 that was used to demonstrate the SPI communication capabilities. Perhaps the most important attribute of the secondary microcontroller is the ability to check for the normal operation of the primary processor through consistent communication with the secondary processor. When activity that is unwarranted is detected by the secondary processor, the state of the primary processor can be reset by the accompanying processor. Such constant verification allows for a more robust testing unit for the researcher using the device.

(f)  *Stimulate.c*

As previously mentioned, the stimulation required in DBS consists of a signal composed of certain amplitude over a certain duration followed by zero voltage. Controlling the three parameters of amplitude, pulse-width, and frequency easily creates these signals. However, the task in practice involves sourcing an analog voltage from a digital output pin (since we do not want to add the hardware of a DAC). To do this, the concept of PWM is used to create analog voltages. Based on the duty-cycle of the wave,
any analog voltage can be created from zero volts to the maximum voltage sourced by the microcontroller. If a pin sources a square wave with amplitude of 5 volts with the voltage high for half the period of the wave, this square wave can be repeated in order to give the effect of having an analog voltage equal to one half of the amplitude (because the duty-cycle was set to half). If a voltage equal to one quarter of the amplitude were desired, the duty-cycle would be set to a quarter of the period (25%). Now that an analog voltage has been produced, in order to create the waves necessary for DBS the PWM procedure is simply turned on and off at the required times.

4.3 VLSI Implementation

Although the microchip chosen to fabricate the ADBS system proved adequate to suit the needs of the application, it was also determined that much of the resources of the processor were wasted. The interaction with the human brain is a biological process, and the frequencies of the signals sensed and stimulated are slow (with respect to the processing speed found in microcontrollers). However, when running computationally intensive algorithms such as ALOPEX, the extra processing power of the microcontrollers can be utilized. For this reason, a CPU clock speed even greater than that of the dsPIC30F2011 can be used to reduce the processing time. Another feature of the aforementioned microcontroller that is not needed for a dedicated chip in an ADBS system is the high pin count. Even in its most basic form, the dsPIC30F2011 has 18 pins that are used for inputs/outputs as well as programming. When constructing a dedicated piece of hardware for the ADBS system the exact amount of pins needed can be constructed. Perhaps the largest portion of the digital signal processor that is not needed for this application is the immense instruction set. The dsPIC30F2011 has 84 base
instructions standard that are used in a vast array of applications. The tasks to be handled by the ADBS system are easily programmed by a set of 10 instructions. The reduction from 84 instructions to 10 instructions surely provides large savings with respect to die space. Practically, if the ADBS system were to be implanted it would be shielded in a case similar to the DBS systems offered commercially. When examining these devices it is found that the bulk of the volume within the device is reserved for battery storage. Although the space savings of a custom chip would not benefit the patient much in this instance, future designs could benefit from a much smaller processor. If researchers were able to circumvent the issue of powering the chip from a large battery (i.e. use solar power through the human eye for instance) then the circuitry for the ADBS system could be incorporated onto the electrode itself, eliminating the need to implant a device in the chest as well as eliminating the lead connecting that device to the electrode. With the aforementioned considerations in mind, the following processor was designed for use in an ADBS system.

4.4 VLSI Approach

The general approach to the design of the processor is standard for the design of any digital hardware in VLSI. Using Cadence, code written in the hardware language VERILOG was compiled. The hardware that is produced from the compiler is far from adequate as far as readiness for production. The actual hardware that is produced must be sifted through to find problem areas that must be rectified for the processor to work correctly. Using a custom test bench, the final hardware must be tested thoroughly in order to further find faults in the hardware. The issue of timing within the processor is a concern even when all of the hardware is found to be fully functional. Generally, the
amount of time debugging the hardware surpasses the actual development of the hardware itself. Since the processor design that was selected for this project is of MIPS (Microprocessor without Interlocked Pipelining Stages) Architecture, the instructions read into the control and executed by the datapath are of a standard assembly code variety. A useful feature of the MPLAB IDE that was used was the disassembly listing from the original written C code to assembly language. Without much software development time, the same instructions written in assembly code can be executed on the custom processor developed in VERILOG.

### 4.5 Processor Specifications

The key specifications of the processor are the processor speed and the number of instructions that are valid. The other major concern when designing the ADBS chip is the power consumption because it dictates the heat dissipated as well as the battery life of the device. The ADBS chip was designed with multi-cycle architecture with the following specifications.

<table>
<thead>
<tr>
<th>Processor Speed</th>
<th>90 MHz</th>
</tr>
</thead>
<tbody>
<tr>
<td>Instructions</td>
<td>10</td>
</tr>
<tr>
<td>Supply Voltage</td>
<td>1.8V</td>
</tr>
<tr>
<td>Power Consumption</td>
<td>1mW per cm³ according to IEEE/ANSI standards</td>
</tr>
<tr>
<td>Register File</td>
<td>32 x 32 bit</td>
</tr>
<tr>
<td>ALU (Arithmetic Logic Unit)</td>
<td>Simple ALU (unsigned) with operations needed for our specific use.</td>
</tr>
</tbody>
</table>

The instruction set is minimized to only 10 instructions, and the power consumption is within the 1mW per cm³ according to IEEE/ANSI standards for implantable devices. It is necessary to understand that this is solely the digital portion of the hardware for the ADBS chip. For this section of the chip the information is entered in
digital format and outputted in digital format (it is strictly a processor). The three basic types of instructions are arithmetic instructions, data transfer instructions, and branch/set less than instructions. Since all of the instructions to be executed are generated internally (with no code developed by an outside source such as the medical practitioner or patient) the method in which data are handled allows a much lower level architecture chip to perform the computations of a much more advanced processor. For instance, the divide instruction was removed because it could be replaced in code by a multiplication of the inverse (along with scaling the result). Furthermore, floating point considerations can be dealt with by using strictly integers scaled by factors of ten. With a few considerations in mind, a basic processor can be fully utilized in an efficient manner allowing for all the processing power that is needed.

**Instruction Set Descriptions:**

(a) *Arithmetic*

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add</td>
<td>Adds two registers and stores the result in a register</td>
</tr>
<tr>
<td>Addi</td>
<td>Adds a register and an unsigned immediate value and stores the result in a register</td>
</tr>
<tr>
<td>sub</td>
<td>Subtracts two registers and stores the result in a register</td>
</tr>
<tr>
<td>mult</td>
<td>Multiplies $s$ by $t$ and stores the result in a register</td>
</tr>
</tbody>
</table>

(b) *Data Transfer*

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>lw</td>
<td>A word is loaded into a register from the specified address</td>
</tr>
<tr>
<td>sw</td>
<td>The contents of $t$ is stored at the specified address</td>
</tr>
</tbody>
</table>
(c) Branches and Set Less Thans

Table 4.4: Branch and set less than instructions supported by the processor

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>beq</td>
<td>Branches if the two registers are equal</td>
</tr>
<tr>
<td>j</td>
<td>Jumps to the calculated address</td>
</tr>
<tr>
<td>slt</td>
<td>If $s$ is less than $t$, $d$ is set to one. It gets zero otherwise</td>
</tr>
<tr>
<td>slti</td>
<td>If $s$ is less than immediate, $t$ is set to one. It gets zero otherwise</td>
</tr>
</tbody>
</table>

Code can be compiled from higher-level languages such as C or Java and run on this processor. These basic instructions are sufficient to run all of the firmware for the ADBS system as well as sufficient enough to process the multiple iterations associated with ALOPEX.

4.6 ADBS Chip Design

4.6.1 Processor

The processor for the ADBS system is internal to the general circuitry and only has a total of 5 inputs/outputs (I/O’s). The names of each of the inputs/outputs are listed in table 4.5.

Table 4.5: Input and Outputs of the Processor

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>MemData</td>
<td>Address</td>
</tr>
<tr>
<td>Clock</td>
<td>Writedata (b)</td>
</tr>
<tr>
<td>Reset</td>
<td></td>
</tr>
</tbody>
</table>

The most crucial input to any processor is the clock that dictates the speed of the processor. Based on the architecture of the processor, the speed the instructions can be executed are directly related to the speed of the clock. This line can be connected to a standard crystal oscillator at various frequencies. The other crucial input to the processor
is the reset line that is used to reset the program counter of the processor to its original state as well as clear all registers. Besides the required clock and reset lines, the other three I/O’s to this chip are associated with the movement of information to and from memory. The MemData corresponds to information entering the processor from memory. All information is stored in memory including the actual program that is executed. An important output of the processor is the address line. The address line tells the memory where to store the information on the writedata line that is outputted by the processor. In other words, the address line holds an address in memory where information will be stored, and the writedata line contains data from the processor that will be stored in memory. The organization of a processor is composed of two main subsections, the control and the datapath. The datapath consists of the units of the processor that perform data processing such as arithmetic logic units (ALU’s). The control gives commands to the datapath to control the flow of data within the datapath. Also, the control unit regulates the interaction between main memory and the datapath. The control can be thought of as the “brain” of the processor. Together, the control and the datapath along with the multiple interconnections between them comprise the processor. In Figure 4.11 the control and the datapath are both depicted. The heavy lines are meant to represent a 32-bit bus as opposed to a single bit line (which is depicted by the lighter lines).
Accordingly, the specific parts of the processor that were designed consisted of the control and the datapath. The specific elements of the datapath include the following: instruction register, memory data register, register file, sign extension unit, two “shift left 2” units, multiplexers, ALU, EPC, and Cause registers. The architecture of the processor is based on a 32-bit version of the MIPS architecture, which includes circuitry for interrupts and exceptions [63]. The control lines with the bit sizes of each line are given on the block diagram of the MIPS based processor (Figure 4.12).

The datapath has all of the inputs that are outputted from the control unit as well as the memory data that comes from an external memory unit. In total it has a combined 21 inputs and outputs (Table 4.6).
Table 4.6: Inputs and Outputs of the datapath

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>PCWriteCond</td>
<td>Address</td>
</tr>
<tr>
<td>PCWrite</td>
<td>Writedata (b)</td>
</tr>
<tr>
<td>IorD</td>
<td>Function</td>
</tr>
<tr>
<td>MemData</td>
<td>Overflow</td>
</tr>
<tr>
<td>Reset</td>
<td>OpCode</td>
</tr>
<tr>
<td>MemtoReg</td>
<td></td>
</tr>
<tr>
<td>IntCause</td>
<td></td>
</tr>
<tr>
<td>PCSource</td>
<td></td>
</tr>
<tr>
<td>ALUSrcB</td>
<td></td>
</tr>
<tr>
<td>ALUSrcA</td>
<td></td>
</tr>
<tr>
<td>RegWrite</td>
<td></td>
</tr>
<tr>
<td>RegDst</td>
<td></td>
</tr>
<tr>
<td>CauseWrite</td>
<td></td>
</tr>
<tr>
<td>ALUControl</td>
<td></td>
</tr>
<tr>
<td>Clock</td>
<td></td>
</tr>
<tr>
<td>EPCWrite</td>
<td></td>
</tr>
</tbody>
</table>
Figure 4.12: Processor Block Diagram [63]
Using PCWriteCond and PCWrite with a small amount of combinational logic the program counter register is told when to read in the value calculated by the datapath. The IorD control line alerts the multiplexer located in front of the memory whether the information entered into the address input to the memory is an instruction or a calculated address based on an instruction. Memdata either consists of a given instruction or of a piece of information being retrieved from memory. MemtoReg is another control line to a multiplexer that determines the information placed on the write data line of the register file. IntCause is used to determine whether the exception that takes place in the processor is due to internal or external reasons. PCSource is used to determine the input to the program counter, which can either be the result of the ALU, jump address, or exception address. ALUSrcA is used to determine whether the first input to the ALU is from the A register or the address on the program counter. The ALUSrcB control line designates the second input of the ALU from the B register, a constant number 4 (used to increment the program counter), a sign extended portion of the instruction, and a shifted left, sign extended portion of an instruction. The RegWrite control line determines whether the register file can be written to or not. RegDst determines which register will be written to, based on the instruction. Causewrite and EPCWrite control lines determine whether the respective registers can be written to. The address line is an output of the datapath used to determine the location in memory to be accessed. The Writedata line contains the information to be written to memory. Function, overflow, and opcode are three signals from the datapath to the control used to determine the state of the processor.
Finally, there is the control that consists of 20 inputs and outputs (Table 4.7):

Table 4.7: Inputs and Outputs of the control

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Function</td>
<td>PCWriteCond</td>
</tr>
<tr>
<td>Overflow</td>
<td>PCWrite</td>
</tr>
<tr>
<td>Reset</td>
<td>IorD</td>
</tr>
<tr>
<td>OpCode</td>
<td>MemRead</td>
</tr>
<tr>
<td>Clock</td>
<td>MemWrite</td>
</tr>
<tr>
<td></td>
<td>MemtoReg</td>
</tr>
<tr>
<td></td>
<td>IntCause</td>
</tr>
<tr>
<td></td>
<td>PCSource</td>
</tr>
<tr>
<td></td>
<td>ALUControl</td>
</tr>
<tr>
<td></td>
<td>ALUSrcB</td>
</tr>
<tr>
<td></td>
<td>ALUSrcA</td>
</tr>
<tr>
<td></td>
<td>RegWrite</td>
</tr>
<tr>
<td></td>
<td>RegDst</td>
</tr>
<tr>
<td></td>
<td>CauseWrite</td>
</tr>
<tr>
<td></td>
<td>EPCWrite</td>
</tr>
</tbody>
</table>
The inputs and outputs of the control correspond to the inputs and outputs that were previously discussed in the datapath for the most part. The I/O is better shown in the schematic given by design analyzer (Cadence) in Figure 4.14.

![Figure 4.14: Top level processor schematic showing control with I/O](image)

In order to build the control unit, a finite state machine (FSM) was constructed which depicts the flow of control decisions. For this processor a 14 part Mealy state machine was used which dictates that the output depends on the inputs and current state. When examining the FSM in Figure 4.15 the control is broken up by the various cycles (Instruction Fetch, Decode, etc.) The first two cycles occur independent of the instruction (inputs), but thereafter are dependent on the Opcode, Function, and Overflow. All
instructions are performed in four cycles except for a “load word” (LW), which is executed in five cycles. The control has also accounted for exceptions which can be caused by an ambiguous Opcode or Overflow from the ALU. In such a case, the state is reset back to state 0. Furthermore, when looking at the FSM one will notice the control lines in each state bubble which are asserted and sent to the datapath at the appropriate time. In this convention, when a control line is not given a value, it is assumed that it is de-asserted to logic ‘0’. The diagram was also used in verification during debugging to check if the appropriate states occurred during a given instruction.

For example, when the load word (LW) instruction is executed the following sequence of events occurs (following Figure 4.24). The processor always starts in State 0 with Memread asserted, ALUSrcA = 0, IorD = 0, IRWrite asserted, ALUSrcB = 01, ALUOp = 00, PCWrite asserted, and PCSource = 00. This sequence of control lines is what is necessary for the datapath to fetch the instruction. On the next clock cycle State 1 is entered which sets ALUSrcA = 0, ALUSrcB = 11, and ALUOp = 00. This consists of the instruction decode/register fetch portion of the process. From this point onward the actual instruction is what dictates the state that the processor is currently in. A portion of the instruction that is fetched consists of an “Op” section which narrows down the type of instruction to be executed. For the LW instruction the processor enters State 4. At this stage the processor does not know whether the instruction being executed is a load word, store word, or an immediate addition. Logically, these instructions are all associated with this state because they require the same tasks to be performed by the hardware. Further differentiation is made by the instruction as State 5 is entered and finally State 6 before the processor returns to the initial state of State 0.
Figure 4.15: Controller finite state machine diagram
The nature of the multi-cycle architecture is shown in Figure 4.16. In a continuous fashion each instruction is fetched from memory before it is decoded and sent to the control. The control then sets the appropriate control lines to their respective values, which in turn dictates the datapath to handle the data in the desired fashion.

![Instruction Sequence](image)

**Figure 4.16: Instruction flow of multi-cycle datapath**

After the datapath has finished the desired task the processed information is either sent back to memory or another region of the datapath for further processing. The task of the datapath is to use the control lines from the control to manipulate a given instruction. When the instruction is first placed on the MemData line it is read into the instruction register as well as the memory data register where the different parts of the instruction are distributed to various sections of the datapath as well as the control. The three types of instruction formats that are supported in this processor are R-type, I-type, and J-type instructions. The breakdown of each instruction with respect to field names is shown in Figure 4.17. R-Type instructions are generally arithmetic instructions that require the use of the ALU to perform mathematical operations. I-Type instructions categorize the instructions that move information between registers such as loading registers with data. J-Type instructions allow for jumping to a certain address in program memory. These are
used to load the program counter with a specific address referencing a certain line in program memory.

R-Type

<table>
<thead>
<tr>
<th>Opcode</th>
<th>Rs</th>
<th>Rt</th>
<th>Rd</th>
<th>Shamt</th>
<th>funct</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>26</td>
<td>25</td>
<td>21</td>
<td>20</td>
<td>16</td>
</tr>
</tbody>
</table>

I-Type

<table>
<thead>
<tr>
<th>Opcode</th>
<th>Rs</th>
<th>Rt</th>
<th>immediate</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>26</td>
<td>25</td>
<td>21</td>
</tr>
</tbody>
</table>

J-Type

<table>
<thead>
<tr>
<th>Opcode</th>
<th>address</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>26</td>
</tr>
</tbody>
</table>

Figure 4.17: Instruction formats for R-type, I-type, and J-Type instructions

Based on the nature of each instruction, the different fields are distributed accordingly. After the control decodes the instruction from the opcode field, the control lines for the multiplexers and registers are set to perform a specific instruction. The general flow of information includes routing through the register file, various sign extension units, multiple multiplexers, the ALU, and a final routing back to the program counter and memory on the write data line.

Perhaps the most important structure in the processor, the ALU requires the most attention during the design process. The designed 32-bit ALU receives the 3-bit ALU control bits from the ALU Control Unit, which is located in the control. The general decoding technique used with the ALU involves decoding the ALUOp using the control.
This is decoded into the ALU control bits that are then entered into the datapath. The control bits for the ALU are shown in table 4.8 for each instruction used.

Table 4.8: ALU control information for supported instructions

<table>
<thead>
<tr>
<th>Instruction Opcode</th>
<th>ALUOp</th>
<th>Instruction Operation</th>
<th>Function Field</th>
<th>ALU Action</th>
<th>ALU Control Input</th>
</tr>
</thead>
<tbody>
<tr>
<td>LW (35dec) 00</td>
<td>load word</td>
<td>xxxxxxx</td>
<td>add</td>
<td>010</td>
<td></td>
</tr>
<tr>
<td>SW (43dec) 00</td>
<td>store word</td>
<td>xxxxxxx</td>
<td>add</td>
<td>010</td>
<td></td>
</tr>
<tr>
<td>BEQ (4dec) 01</td>
<td>branch equal</td>
<td>xxxxxxx</td>
<td>subtract</td>
<td>110</td>
<td></td>
</tr>
<tr>
<td>R-type (0dec) 10</td>
<td>Add</td>
<td>100000</td>
<td>add</td>
<td>010</td>
<td></td>
</tr>
<tr>
<td>R-type (0dec) 10</td>
<td>Subtract</td>
<td>100010</td>
<td>subtract</td>
<td>110</td>
<td></td>
</tr>
<tr>
<td>R-type (0dec) 10</td>
<td>Multiply</td>
<td>011000</td>
<td>multiply</td>
<td>000</td>
<td></td>
</tr>
<tr>
<td>R-type (0dec) 10</td>
<td>set on less</td>
<td>101010</td>
<td>set on less</td>
<td>111</td>
<td></td>
</tr>
<tr>
<td>Addi (8dec) 00</td>
<td>addi</td>
<td>xxxxxxx</td>
<td>add</td>
<td>010</td>
<td></td>
</tr>
<tr>
<td>SLTI (10dec) 11</td>
<td>set on less imm</td>
<td>xxxxxxx</td>
<td>set on less</td>
<td>111</td>
<td></td>
</tr>
<tr>
<td>J-type (2dec) xx</td>
<td>jump</td>
<td>xxxxxxx</td>
<td>add</td>
<td>010</td>
<td></td>
</tr>
</tbody>
</table>

The “x’s” in the above table refer to “don’t cares”. The inputs to the ALU are from the 32-bit registers A and B. These registers provide the two operands for the ALU. The ALU output is 32 bits that are fed into the ALUout register for the next clock cycle. The other output is the zero output, which is used for branch operations. When a branch instruction is executed, the control checks for the output of this single bit output. This zero output is ANDed with the PCWriteCond control bit, which decides whether the
branch was successful, and the new address to be fed in the Program Counter (PC). It is important to note that the ALU is functioning during every single instruction that is executed (arithmetic or not). Without an efficiently designed ALU the processor cannot be efficient.

The registers of the processor are stored in a structure called the register file. These registers can be read or written using the required input and the specific number of the register. For a read operation from the register file, the register number is required, whereas for a write, it requires the register number where the new data has to be written and the actual data that has to be stored. For the ADBS chip a 32 word by 32-bit instruction register was constructed. This was estimated from the number of registers required for execution of code using the dsPIC30F2011 (which contained only 16x16 bit working registers). The extra registers were provided for future expansion of the entire circuitry if needed. The architecture of the processor was designed to be a 32-bit architecture as opposed to the 16-bit architecture of the dsPIC series of microcontrollers. This allows for larger numbers being processed within each instruction, which is a helpful tool for this application. Furthermore, there is a set of temporary registers that were constructed. For the multi-cycle datapath implementation, at the end of every clock cycle, all data that are used in subsequent clock cycles must be stored in a state element. For this reason the registers denoted as A, B, ALUOut, EPC, Cause, PC and MemoryData hold the values of the subsequent paths on the next clock cycle. Another piece of hardware that was included in this processor was the sign extension unit. The sign extension unit has a 16-bit input that is sign extended into a 32-bit result. This is used to make the memory offset to 32 bits which will be used on immediate operations.
Due to a multitude of features including electromagnetic interference or overflow from an ALU operation, at times the opcode of the instruction may be unrecognizable to the processor. To account for odd occurrences the processor has been designed with exception handling hardware. Whenever an exception occurs, the address of the instruction has to be stored. In the ADBS processor the ALU provides the next PC value and this is saved in the EPC register. The main control sends an INTCause signal to the Cause Vector that helps to write it in the CAUSE register. From the state machine (Figure 4.15) it can be seen that this can originate from internal or external errors. States 13 and 14 represent this when looking at the finite state machine diagram.

4.6.2 PWM Module

An essential portion of the ADBS chip is the hardware responsible for the stimulation of the brain tissue. In the microcontroller design presented earlier, the internal hardware of the microcontroller necessary to perform the pulse-width modulation is the Output Compare module. Using this module of the dsPIC alongside the available timers and custom firmware the desired PWM signal was produced. However, instead of utilizing auxiliary modules and timers, a PWM module can be constructed that has the dedicated purpose of stimulation. An added benefit of such customization is that the module can be replicated multiple times for the multitude of contacts available on an ADBS electrode.

The design for the PWM Module was first started in VERILOG in the same manner as the main processor. Also, the simulation procedure was the same for the PWM module as it was for the main processor, producing the same type of testing environment. The test bench that was written for the PWM module followed the following procedure.
The basis of the test bench is the alternating nature of the signal that is the clock. Every 50,000 clock cycles the signal alternates to the opposite state. The first instruction in the test bench involves writing a value to the period and pulse-width (after an initial reset of the processor). This is done twice with two different sets of parameters (to show the capability of different pulse-widths and periods). The basis for the VERILOG that creates the PWM module is a counter that increments until the desired pulse-width or period is reached. Once the conditions are satisfied the signal is set low and the signal is constructed in an iterative fashion. When the VERILOG is compiled into hardware gates, it becomes almost unrecognizable, but the basic premise is as was described. The same logic was used in the microcontroller implementation of the PWM wave using the output compare module and a timer. The module that is created in VLSI is dedicated to the task of creating the signals required for ADBS.

4.6.3 ADC Module

The concept behind implementing an ADC module is not a difficult one especially if a PWM module has already been constructed. It is a further testament of the versatility of the dsPIC30F2011 that contains multiple ADC’s in such a compact package. The basic premise behind the construction of the ADC is shown in Figure 4.18.

![Figure 4.18: ADC Module using PWM module and ADBS Chip](image-url)
In Figure 4.18, the building blocks of the design have already been constructed for a Successive-approximation ADC. The only thing that is missing is the algorithm the processor must undertake to coordinate the task. The only additional piece of hardware that is present in Figure 4.18 is the comparator, which is merely an operational amplifier that is configured in the manner above. For a realizable comparator, the output voltage is a function of the input voltage as described in equation 4.2.

\[
V_O = \begin{cases} 
V_{OH} & \text{if } (V_{in+} - V_{in-}) > V_{IH} \\
A_v (V_{in+} - V_{in-}) & \text{if } V_{IL} < (V_{in+} - V_{in-}) < V_{IH} \\
V_{OL} & \text{if } (V_{in+} - V_{in-}) < V_{IL}
\end{cases}
\]  

(4.2)

As can be seen in equation 4.2 the op-amp will output a “high” signal if the difference between the positive and negative terminals is greater than the input high voltage. The “low” signal will be generated if the difference is below the input low voltage. In the analog region of operation, the output is the gain factor multiplied by the difference of the two input signals. For this reason, the gain must be made large enough to saturate the output guaranteeing that the output is always digital (high or low) allowing for accurate reading by the digital input pin of the ADBS chip. After pre-amplification, the signal that is read into the ADC is roughly between 600 and 700 millivolts. For the ADBS chip, roughly 1.8 volts of variation are needed to distinguish between a high and low signal. Therefore, for the comparator there exists an input swing of roughly 100 mV to an output swing of 1.8 volts requiring the gain of the comparator to be at least \(\frac{1.8}{0.1}\)
= 18. To assure the reliable operation of the device in the digital region, the gain should be in the vicinity of 200 as opposed to 18.

The algorithm that is implemented by the ADBS chip is as follows. In software, a register will start with the most significant bit turned on, and the rest of the bits turned off. From this digital value, the PWM module generates an analog voltage that is proportional to it. Therefore, when this register begins with the value of 1000000000 (in the case of a 10-bit ADC), the PWM module will be given a duty-cycle of 50% (because 1000000000 is half of the range of 1111111111). This duty-cycle of 50% will then force the PWM module to provide a voltage equal to 50% of the supply voltage. This analog voltage will be low-pass filtered (attempting to eliminate any aberrations in line voltage) and then fed to the negative terminal of the comparator. The positive terminal of the comparator is the ADC input, and this will have an arbitrary voltage on it. Based on equation 4.2, if the ADC input voltage is greater than that on the negative terminal of the op-amp, a “high” signal is generated by the op-amp which is fed back to the ADBS chip through the digital input. If a “high” signal is read by the ADBS chip, the most significant bit in the aforementioned register is turned off, because clearly the voltage on the ADC line is larger than that generated by the PWM module. If the signal remains “low” the most significant bit will remain high, and the next highest significant bit will be turned on (and the process repeated until the least significant bit is tested). Using this algorithm the ADC can be constructed from simple building blocks without the need for fancy circuitry.
4.7 Microcontroller Results

One of the measurable attributes of the ADBS design using microcontrollers is the pulsation during the stimulating sessions. If live subjects were available then the ADBS algorithm would be tested allowing for the optimization of the stimulation parameters similar to the examples in section 4.1. The first place where simulation can take place is in the actual microchip development environment (MPLAB IDE) where any outputs can be monitored with respect to the number of clock cycles that have elapsed. In Figure 4.19 the output compare module (pin OC1) is captured in the middle of a session of DBS. Arbitrarily, a pulse-width and duty-cycle were selected (in number of cycles). It is apparent that in terms of programming, the microcontroller is only concerned with the unit of clock cycles, and if the speed of the processor was adjusted to a different value the factors within the programming would have to be adjusted in a proportional manner (i.e. if the clock speed were halved the number of clock cycles programmed would also have to be halved in order to maintain the same actual time).

As expected, the simulator provided results with 100% accuracy including near zero rise and fall times for the ideal square waves generated. Although this is a fine graphic representation of the intention of the programming on the microcontroller, one would be naive to neglect the capacitive and inductive effects present in practice. In order to view the culmination of these real-world effects, an oscilloscope was necessary in conjunction with the actual circuit to be tested.
Figure 4.19: top: Zoomed out view of stimulation waveform taken from MPLAB IDE showing the pulse-width and frequency characteristics. The x-axis is in terms of clock cycles. Bottom: A zoomed in view of the waveform showing the duty cycle of the PWM wave that dictates the amplitude of the resulting waveform.
The general procedure involved commanding the microcontroller to stimulate at a desired frequency, pulse-width, and amplitude, and then observing the resulting waveform. It was apparent immediately that an oscilloscope with a high sampling rate and advanced triggering capability was needed.

![Figure 4.20: Pulse-Width Modulation for 134 microseconds](image)

In Figure 4.20, the pulse-width of the DBS signal was commanded as 134 microseconds. As can be seen from the above plot, the pulsation lasts approximately 132 microseconds (according to the oscilloscope). This is within the tolerance of what is acceptable since the DBS devices are usually adjusted in increments of more than 10 microseconds. Although the general shape of the waveform was acceptable, the rise-times and fall-times of the signal became apparent from the initial plot. In later plots these attributes were further exaggerated by expanding the time axis on the oscilloscope.
If the time axis of the oscilloscope is compressed (time intervals made shorter) then the plot is effectively being viewed from a zoomed out vantage point. The resulting waveform can be seen in Figure 4.21.

In Figure 4.21, the frequency of the DBS is roughly 15 Hz, and this is shown by the time period equaling roughly 67.4 ms. The pulses are quite faint (almost non-existent) due to the short time period of actual stimulation. One has to focus on the plot to see the delta impulses that are present at a uniform frequency. An exact frequency of 15 Hz would have given a time period of 66.67 ms, however as can be seen by the crudeness of the measurement, 67.4 ms is acceptable. From laboratory notes it is apparent that 15 Hz was the frequency commanded by the microcontroller. It is certain that this frequency is as accurate as the internal oscillator. The signal is produced pulse by pulse, and as can be
shown by the other figures the microcontroller is accurate at the pulse level (order of microseconds). The deviation between the measurement of the frequency and the commanded frequency can be attributed to oscillator error as well as the rise and fall times associated with the capacitive/inductive effects within the real-world circuit.

Figure 4.22: Single pulse generation using dsPIC30F2011

The individual pulse that is generated is shown above, it is on the order of 1.35 microseconds. As can be seen from the plot of Figure 4.32, the period of the signal is 6.79 microseconds. As programmed by the microcontroller the duty cycle of the PWM wave is set at 20% which should yield a DC value of (0.2)*Vdd after post-filtering. The most important observation from this plot of the waveform generated by the microcontroller is the method by which the signal attains a commanded value. It is
apparent that the microcontroller oscillates both as it reaches its peak voltage and as it settles to a voltage of zero.

The concept of PWM relies on the post-filter (a low-pass filter) to smooth out the individual pulses into the desired DC voltage that is dictated by the duty-cycle of the PWM wave. It is generally accepted that the resistance multiplied by the capacitance of the low-pass filter (RC time-constant) should be dimensioned so that it is 100 times higher than the period of the signal. For this application the period varies by a factor of roughly 15 in practice (i.e. the period of time between pulses can vary by as much as 15 times). For this reason the variable capacitance in the post-filter was introduced in order to produce waveforms with the desired time characteristics (delay effects minimized).
Figure 4.24 shows one such waveform with the time-axis expanded to exaggerate the delay effects.

After post-filtering the PWM wave from earlier figures the above waveform results. It can be seen that as the time constant of the filter is adjusted the waveform either becomes smoother or less smooth over the rising portion of the signal. Two attributes are needed in order to have the ideal square wave, a flat steady state portion, and rise/fall times of zero. In the middle portion of the waveform the ultimate smoothness is required (ideally a slope of zero) and when the waveform drops to zero the filter should allow the waveform to react as quickly as possible. For commanded waveforms with short pulse-widths this can be an issue because it is not possible to have an infinite slew rate with a perfectly flat dc portion of the wave following it. Additional filters with
variable gain can be inserted into the signal path in order to pre-shape the signal prior to
the low-pass filtering in order to obtain a more idealized square wave, but for the general
signals that are commanded by ADBS chip, this issue is not a large one.

When a smaller time constant filter is used the wave becomes excessively choppy
as in Figure 4.25. However, as previously stated, it must be realized that this problem is
only existent when small pulse-widths are demanded. For larger pulse-widths the time to
ramp up to the desired voltage and ramp down to zero are less crucial. Referring to
Figure A6.1 in the appendix, the frequency analysis of a post-filtered DBS pulse is
depicted. It can be seen that the majority of the frequency response is in the lower
frequency region; most of the high frequency components were suppressed by the low-
pass filtering. As opposed to the issue of the short-pulse width signal, the other extreme exists that must be examined (Figure 4.26).

In the Figure 4.26 the other extreme case is depicted with one of the higher frequencies being commanded and longest pulse-widths required in DBS. As can be seen, even under these conditions, the period of the PWM signal is still substantially larger than the pulse-width of the stimulating section. If the stimulating section is zoomed in on (time-axis expanded) the plot in Figure 4.27 is seen.
As it can be seen, after post filtering the wave is slower to reach its maximum value and slower to reach a value of zero again. Of course, the main goal of this was to smooth out the pulse-train and provide a smooth DC voltage. As can be seen, the ideal case is almost achieved, but in practice this should suffice. Another similar figure can be found in Figure A6.2 that shows a graph of a 220 microsecond DBS pulse after post-filtering. When looking at the total charge delivered as opposed to the ideal case it is similar. Keeping in mind Figure 4.26, in the worst case the initial stimulating portion of the signal will not interfere with the next stimulating portion and thus the timing is less critical.
4.8 VLSI Results

Although the computer architecture scheme developed in theory was relatively straightforward, the actual coding of the hardware as well as the debugging was anything but trivial. Unlike coding software in languages such as C, there are additional considerations that must be accounted for when coding hardware in VERILOG. Most of the concern when coding software in C centers on memory usage issues as well as the logic of the process being coded. When coding in VERILOG, the compiler does not always produce the expected hardware from what was compiled. The additional work is encountered when the hardware is optimized from its original form. The additional step of looking into the hardware itself and debugging at the gate level was required. The optimization done by the software eliminated redundancies within the hardware and streamlined it allowing for fewer gates to be required. However, in the process some key structures can be deleted, and wires can be left errantly hanging causing inaccuracies when actually simulating using the processor. One module that was designed in VLSI was the actual PWM module. The purpose of designing this piece of dedicated hardware is to output a desired PWM wave based on a commanded frequency and pulse-width (giving the desired duty-cycle of the wave). As was done with the microcontroller, the PWM module will be turned on and off to give the correct pulse-width for the final signal. As can be seen in the VLSI simulation in Figure 4.28, the PWM module performs superbly when given the desired parameters of pulse-width and period. In Figure 4.28 it can be seen that the last two registers (period and pulse_width) are altered to provide a second PWM wave. When looking at the pwm_out signal which is the actual PWM wave that is created, it produces two distinct PWM waves with different frequencies and pulse-
width. It is evident that this module is straightforward with few interfacing wires. The integration of this hardware with the rest of the ADBS chip is a trivial task that can be done during layout (if this is a goal of future research).
Figure 4.28: Output from VLSI design of PWM unit
Most of the structures within the datapath consist of registers (flip-flops). These registers and multiplexers contain lines that are connected to the control that dictate how the structures in the datapath operate. The main structure within the datapath is the ALU, which is the heart of the processor itself (it performs all of the calculations). It is not possible to view the detail of the complete datapath on a single screen, however if one were to expand the figure many times then the inputs and outputs to the datapath are evident.

There are no control lines originating from the datapath. All of the structures shown in the datapath are fully dependent on the control lines from the control in order to sequence the entire process of decoding and executing an instruction. The datapath is analogous to the human body without a brain present. Although all of the physical structures are available for use, without the brain/control no activity is taking place. It is also worth noting that the datapath shown is optimized by the compiler due to many redundancies that can be exploited from the original designed datapath. The effects of the optimization are quite drastic in that they reduce an incomprehensible cluster of flip-flops, gates and wires into a much reduced and more orderly logic gate layout. The same optimization procedure is done to the control portion of the VLSI implementation. As opposed to the optimized hardware of the datapath, the optimized hardware of the control path yielded many more logic gates. The power of the optimization software minimized the size of the hardware significantly.
After the compiler generated the hardware, the arduous task of processor verification began where a test bench was created to test the various sections of the processor. Primarily, every instruction was entered from program memory and ran on the processor. The various outputs were monitored to ensure that the hardware was performing as expected. Various different instructions were loaded as well as sample data entered in registers. The different registers from the MIPS nomenclature map to the register file as shown in table 4.9. When compiling the instruction to machine code from assembly language, these register numbers are what represent the names of the registers on the left.

<table>
<thead>
<tr>
<th>Name</th>
<th>Register Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>$zero</td>
<td>0</td>
</tr>
<tr>
<td>$v0-$v1</td>
<td>2-3</td>
</tr>
<tr>
<td>$a0-$a3</td>
<td>4-7</td>
</tr>
<tr>
<td>$t0-$t7</td>
<td>8-15</td>
</tr>
<tr>
<td>$s0-$s7</td>
<td>16-23</td>
</tr>
<tr>
<td>$t8-$t9</td>
<td>24-25</td>
</tr>
<tr>
<td>$gp</td>
<td>28</td>
</tr>
<tr>
<td>$sp</td>
<td>29</td>
</tr>
<tr>
<td>$fp</td>
<td>30</td>
</tr>
<tr>
<td>$ra</td>
<td>31</td>
</tr>
</tbody>
</table>

The various instructions that were used as part of testing to ensure that the processor was running correctly are given below. The plotted waveforms from figure 4.29 represent a typical testing cycle resulting from the execution of this assembly code.

Addi $s1, $zero, 128 % adding the zero register to 128 and store in register 17
Sw $s1, 0($zero) % store the value in register 17 to memory
Add $s4, $zero, $s1 % add register zero and 17 and place in register 20
Sub $fp,$s4,$s1 % register 20 minus register 17 and place in register 30
Mult $s4,$s1,$at % multiply register 20 and 17 and place in register 1
Slt $zero,$s1,$v0 % check if register 0 is less than register 17, if so set register 2 high
Beq $s4, $s1,8 % branch to 8 if registers 20 and 17 are equal
J 8 % jump to pc +4+8
Figure 4.29: Output Waveforms for various different instructions
Table 4.9 shows the MIPS convention used when compiling assembly code into register numbers. When $s4$ is written in assembly the compiler tells the processor to refer to register number 20. When referring to Figure 4.29 it is easy to follow the code written in assembly by observing the row labeled “state” and following the finite state machine (Figure 4.15) to see the instruction that is executed. The signals listed in Figure 4.29 can be seen in the block diagram from Figure 4.15. Figure 4.12 is the roadmap to understanding how the ADBS processor decodes the instructions that are executed. As can be seen from the top of Figure 4.12, the control unit supplies the control lines to the datapath (mid to lower section of the same figure). With the sequential turning on/off of these control lines the datapath is forced to process the incoming binary information (instruction) in the manner which is dictated by the assembly code. Midway through the assembly code listed above can be noticed a MUL (multiplication) instruction. The proper method to understand how the instruction is executed is as follows. Referring to the lines of code which are executed it is known that this line is executed midway through the simulation. Using the state machine diagram from Figure 4.15 we know that the processor should follow the following state transitions. The processor always starts in state 0 before a path must be chosen depending on the instruction. In state 0 it is known from the information presented over the past chapter that the instruction is fetched from memory. At this point in time, the processor has no idea regarding the contents of the instruction. Once the instruction is fetched the processor is forced into state 1 where the instruction is decoded and the registers are fetched. This can be visually seen when following the state of the control lines in Figure 4.12. Once the instruction is decoded the processor must move to the next state depending on the content of the instruction. Since
this instruction consists of a multiplication (R-type) the opcode has a value of 0 which dictates that the processor must move to state 9. The transition from state to state consists of the alteration of the values of the control lines from the control unit. The datapath is simply slave hardware which is told what to do based on the control lines. From state 9 according to the state diagram the next state is state 10 which is the completion of the execution of the instruction. If no exceptions are found the state of the processor returns to state 0 where the process is repeated for the next instruction. Looking at Figure 4.29 one will notice the signal labeled as dut.controlunit.state[3:0]. This signal holds the value of the state of the processor which has been the topic of discussion. Since the multiplication instruction takes place midway through the code that was simulated we must look toward the middle of the simulation to notice the state transitions that were discussed above. When observing the state transitions one will notice the transition from state 0 to state 1 to state 9 to state a (a in hexadecimal format is 10) which is actually repeated (due to multiple R-type instructions executed). Other signals worth observing are the data that is executed in the instruction, the address of the data, as well as the clock itself. The instructions were all successfully executed at a clock speed of 100 MHz, which is fast enough for the tasks required by the ADBS system. Not only is this design faster and more efficient for the purposes of ADBS, but also it serves as a dedicated piece of hardware for ADBS that can be used by future researchers. In the same manner as described above the remaining instructions can be seen in the simulation in Figure 4.29. The actual Verilog code that constructed the control unit in VLSI is provided in Appendix A10. Even if one is not familiar with the Verilog language, the flow of the code resembles that of a state machine (which is precisely the main format of the control
unit). The remainder of the VLSI code looks similar due to the fact that it is written entirely in Verilog. However, after the code has been compiled and simulated, the optimization process occurs. After much debugging one will notice that the actual Verilog code is changed to a format that is almost unrecognizable (a list of nets and gates). This is understandable due to the fact that Verilog is a programming language that constructs actual hardware to be fabricated.
CHAPTER 5
Hardware Breakdown

The ADBS system is implemented on the Microchip platform using a combination of peripheral hardware. The software is written in both assembly language and C, and is considered dedicated firmware for the microcontroller itself. When laying out a circuit on a board there are a multitude of factors to consider. Furthermore, since the microcontroller used was not designed solely for the purposes of this application, space considerations must be taken into account when utilizing the peripheral hardware. In the end, the goal is to construct a functioning board which can be used by clinical researchers to test the deep brain stimulation algorithms (primarily ALOPEX) on live subjects. All software written is burned on the microcontroller which is found on the board. The peripheral circuitry performs the tasks the microcontroller is not capable of performing including negative voltage generation and general voltage regulation. The combination of these different circuits culminates in the construction of the ADBS unit that can be used during clinical testing.

5.1 Negative Voltage Generation

One issue when observing the stimulation waveforms used in DBS systems is the polarity of the signal. As a matter of fact, it is imperative in DBS systems to use both positive and negative voltages in the same waveform for charge balancing reasons (Figure 1.8). The question arises whether microcontrollers are capable of generating negative voltages. As expected, this is not a standard option on any microcontroller. However, there exist dedicated chips that can be used to generate negative voltages.
When examining the limited space available within the stimulator device that is implanted, it is clear that a separate chip is not used to generate negative voltages. In practice, there is a principle within power electronics that allows for the generation of negative voltages known as charge pumping [92]. With a few capacitors and transistors, a microcontroller can be used to generate negative voltages or even higher voltages than the supply of the microcontroller. However, it is even possible to avoid external circuitry completely using our advanced microcontroller. A potential difference between two pins dictates the direction that the current flows between the two pins. In the example of two pins, if initially an analog voltage of 5V is applied to the first pin and ground is applied to the second pin we can refer to the potential difference between the two pins as $+5$ V. If we apply the same 5V to the second pin and the ground to the first pin this is the same as applying $-5$V. The direction of the current was reversed which is the purpose of applying the negative voltage. In order to route the two signals (the analog positive voltage and the ground) some external circuitry must be added. Furthermore, the type of circuitry is dependent on the number of contacts on the electrode being used. For the standard DBS electrode used in practice the following design can be used.

![Figure 5.1: Negative voltage generation using microcontroller and two 2-bit demultiplexers](image)

Figure 5.1: Negative voltage generation using microcontroller and two 2-bit demultiplexers
In Figure 5.1 the neurochip is either a standard microchip or a custom neurochip fabricated in VLSI. The digital I/O (input/output) lines will be turned on/off in software according to which contacts must be stimulated. Using this wiring the PWM signal and the ground signal are each placed on separate contacts based on the algorithm programmed in the processor. However, if a novel electrode is used that utilizes 15 contacts, the two bit DE-MUX’s must be replaced with 4-bit DE-MUX’s to accommodate the extra contacts. Although this circuitry is not complex, there are other considerations when connecting an electrode to an ADBS chip, and thus there will be much more circuitry that works in conjunction with this concept. However, this is how negative voltages would be applied in a multi-contact application. For the single contact application discussed in this dissertation these factors do not have to be considered.

Another solution providing negative voltage generation involves using dedicated integrated circuits such as the MAX828 offered from Maxim Corporation in conjunction with external capacitors and resistors [83]. However, for a practical solution that can be surgically implanted, the use of multiple IC’s is prohibited.

5.2 Voltage Regulation Circuitry

When supplying the power to any microcontroller or integrated circuit it is imperative that the power supply is a regulated supply. This entails that the supply voltage will remain fixed no matter what load is present. Simply connecting a battery to an IC provides an unregulated power supply, and can lead to faulty operation of the IC. If the power line is noisy the microcontroller may not be able to operate properly (which is the case of the dsPIC30F2011). In order to power the microcontroller either a regulated
power supply can be used in the laboratory environment or the following circuit can be connected between the portable battery and the power input of the chip (Figure 5.2).

![Diagram of Zener Diode Regulator Circuit](image)

**Figure 5.2: Zener Diode Regulator Circuit [81]**

The premise of operation for this circuit is that given an input voltage that is roughly higher than the desired regulated voltage by roughly a couple of volts, a stable output should be expected. Even if the input voltage increases, the zener diode maintains the voltage drop that it was designed to maintain. For the purposes of standard microcontrollers a 5V Zener diode can be used (most IC’s accept a 5V input). However, as the current increases the voltage across the resistor increases and this can be wasteful if not designed properly. The voltage across the resistor is the difference between the supply voltage and the voltage across the Zener diode, therefore the Zener diode’s rated voltage should be as close as possible to the input voltage (within a couple of volts).

### 5.3 Voltage Offset and Input Buffer Circuitry

Aside from the inability to produce negative voltages, standard microcontrollers with analog to digital converters are unable to sense negative voltages. Local Field Potential data often is negative (as described earlier) and circuitry is necessary to process the data. A solution for such a dilemma is circuitry that offsets the voltage accordingly, providing only positive voltages for the ADC of the processor to digitize. However, in the
case of the sensitive ADC, the resistance on the pin has a large effect on the accuracy of the module. Therefore, additional circuitry must be added as a buffer between the voltage offset circuitry and the input to the ADC.

![Figure 5.3: Voltage Offset and Input Buffer Schematic](image)

As it can be seen in the schematic of Figure 5.3, the LFP signal first encounters a capacitor that blocks any DC voltage allowing only the fluctuations of the LFP to pass. This voltage will eventually be added to the offset voltage provided by the two equal resistors connected to the 5V supply. This supplies a constant 2.5V that is added to the fluctuating LFP signal ensuring a positive voltage. The standard op-amp shown with negative feedback provides large input impedance (theoretically infinity) and an output impedance of a couple ohms (which is highly desirable for the ADC input pin of the microcontroller). In the processor the adjustment for the additional 2.5V is made in the firmware of the microcontroller. Furthermore, the input stage of the resistance with the capacitance provides for input filtering. If the equation for the cutoff frequency is followed as shown in equation 5.1, then the values of the resistors and capacitors can be determined accordingly.

\[
f_c = \frac{1}{2\pi RC}
\]  

(5.1)
For the application of DBS all frequencies above 250 Hz can be filtered out. Therefore using equation 5.1 this provides a time constant of 637 microseconds (the product of the resistance and capacitance). In the case of Figure 5.3 the equivalent resistance of the two resistors (of value R) is equal to R/2 because they are in parallel. According to the size requirements of the layout the resistor and capacitor can be sized to this design constraint.

5.4 PWM Post-filter

After the microcontroller has successfully produced the necessary pulse-train required, the analog voltage necessary for DBS requires further filtering. Using a simple RC filter that is relatively slower than the PWM wave, the digital signal can nearly produce a constant analog voltage. When dimensioning the filter there are a couple of concerns. If the time constant of the filter is too large the output will react too slowly to the changing input not allowing for a responsive stimulation waveform. Conversely, if the filter is dimensioned too small then the smooth analog voltage that is expected will appear to have the jagged characteristics of the unfiltered pulse-train. A happy medium must be found in order to ensure successful pulse-width modulation. Typically, the time constant of the RC filter should be 100 times larger than the period of the signal. This issue is most apparent when trying to resolve crisp waveforms with small pulse-widths (time constant too large).
Equation 5.1 governs the cutoff frequency for the low-pass filter in Figure 5.4. The main issue is that the pulse-width and frequency vary across a large range, and having a single capacitance provides only a single cut-off frequency. This would not be a problem if a variable capacitance were to be used. For the purposes of DBS, it is possible to use a simplistic filter as shown is Figure 5.4, however if a precise filter were needed, the following circuitry could be used (Figure 5.5).

This simplistic circuit applies the ground line through the three-bit demultiplexer to terminate the circuit at different junctions. It can be seen that if ground is applied to the first output of the demultiplexer (from the left) then there is only a single capacitance
present in the circuit. If the second line of the demultiplexer is selected then there are two capacitances in series (which add in parallel) in the circuit. If each subsequent line in the demultiplexer is selected then the original capacitance of the circuit will be halved each time. The sizing of the passive elements in Figure 5.5 was based on values that were used during experiments in the Computational Intelligence Laboratory.

5.5 Impedance Sensing Circuitry

A critical attribute to the ADBS system is the ability to sense the impedance on each contact of the electrode. As mentioned earlier the impedance of brain matter is between 500 and 1500 ohms. Any impedance sensed less than 50 ohms indicates a short circuit in the device. If the impedance were greater than 2000 ohms this most likely indicates a broken lead. Besides these two extreme cases, the impedance in human subjects varies over time due to glial scarring, and adjustments must be made in the stimulation parameters to counteract the scarring effect. Present systems have no mechanism to handle these occurrences, and rely purely on the ability of the doctor to determine the extreme conditions as well as provide the subtle adjustments the patient requires. To sense the impedance of the electrode a simple voltage divider is used alongside simple calculations done by the processor.

![Figure 5.6: Voltage divider used to sense impedance of contact](image)
In the voltage divider circuit of Figure 5.6 the voltage sensed by the ADC will vary as the contact resistance varies. The “power” given to the entire processor (Figure 5.6) will vary between different generations of designs, so calculations will be shown for a supply of 5V. For the given circuit the contact resistance is given by equation 5.2.

\[ R_c = \frac{100 \cdot V_{ADC}}{1 - 0.2 \cdot V_{ADC}} \]  

(5.2)

To show the range of the values read by the processor and how it corresponds to actual resistances, numbers can be substituted in equation 5.2. It can be seen that for an infinite contact resistance the voltage of the ADC will sense 5 V and for zero resistance a value of zero will be read. A more realistic representation of values shown is for a contact resistance of 500 ohms the ADC will sense a value of 2.5V, which is in the middle of the quantization range. If the supply voltage were to decrease (as is the case for the dedicated processor fabricated in VLSI) the equation would vary slightly in accordance with the standard voltage divider equation. If extremely accurate results were necessary the supply voltage could be monitored using an auxiliary ADC pin, and this value substituted into the voltage divider equation in real-time providing accurate results. However, due to the slow changing nature of brain matter impedance, this is unnecessary (and would actually consume much needed power).

Due to the fact that each contact on the electrode is responsible for not only stimulation, but also sensing and impedance recording, additional circuitry must be included. Although the in depth circuitry is included in a parallel dissertation [78], in Figure 5.7 the basic premise is outlined.
In Figure 5.7 the NPN and PNP transistors are wired together forming a pass gate allowing a strong logic “1” and strong logic “0” to be passed. When the digital I/O connected to the gate is excited, this indicates that the impedance circuitry will be connected to the contact (Vdd connected through a 500 ohm resistor to the contact). When this output pin has a value of logic “0” then either the ADC is sensing alone or the PWM signal is being passed to the contact (this is dependent on the demultiplexer controlled by the neurochip). This section of the ADBS system is not fabricated in this dissertation and is only provided as a reference to show how the neurochip interacts with the physical contact.

5.6 Power Considerations

When dealing with any electronic circuit that is embedded into a medical device the power consumption is of utmost importance. Whether referring to the hardware
designed using the dsPIC30F2011 or the custom VLSI design, the following principles apply. All power that is drawn from the chips is drawn from a voltage source attached to the \(V_{dd}\) pins of a chip. The total power drawn by the chip is a sum of the static power consumption and the dynamic power consumption [82].

\[
P_{TOTAL} = P_{STATIC} + P_{DYNAMIC} \tag{5.3}
\]

Static power dissipation is the baseline power that is dissipated when the chip is powered on and no extraneous operations are taking place. This power dissipation is due to a myriad of different causes [82]. A major cause of static power dissipation is subthreshold conduction through OFF transistors. Also, the tunneling current through the gate oxide as well as the leakage through reverse-biased diodes has a substantial effect as well. Another factor that leads to static power dissipation is contention current in ratioed circuits. Dynamic power dissipation is primarily due to a couple of causes. The first main source of dynamic power consumption is the charging and discharging of load capacitances. The second major source of dynamic power dissipation is the short-circuit current while both pMOS and nMOS networks are partially ON. Although it is possible to estimate the power that is dissipated, it is easier to use software to obtain power consumption numbers. However, it is important to realize the variables that influence the power consumption.

\[
P_{STATIC} = I_{STATIC} \cdot V_{DD} \tag{5.4}
\]
It is clear from equation 5.4 that the static power dissipation is a function of the static current consumption and the voltage of the power supply. This is the first clue for the hardware chip designer or even the microcontroller user that the lower the supply voltage that is used, the less power will be dissipated (in a linear fashion). When operating the dsPIC30F2011 the operating range is from a couple volts up to a little above 5V. If used in an actual medical device, it would be advantageous to use the lowest supply voltage possible. For the dynamic power dissipation the reduction of the supply voltage is even more of a factor.

\[ P_{DYNAMIC} = C \cdot V_{DD}^2 \cdot f_{sw} \]  

(5.5)

From equation 5.5 it can be seen that the dynamic power dissipation is a function of the square of the supply voltage. Although this is immensely greater than the effect the supply voltage had on the static power dissipation, another key factor is introduced in equation 5.5. The operating frequency denoted by \( f_{sw} \) refers to the operating frequency of the chip. The dynamic power dissipation is a linear function of the operating frequency of the chip. This implies that for minimal dynamic power dissipation the processor should be ran at the slowest possible processing speed. The capacitance in the dynamic power dissipation equation cannot be altered for a microcontroller that is purchased, however it can be changed in a custom VLSI design to obtain an even lower dynamic power dissipation Figure. As shown by the above equations, if the designer is able to minimize both the supply voltage and the operating frequency of the processor, a lower power dissipation figure can be obtained.
5.7 Layout Considerations

There are general considerations to take into account when fabricating a circuit on a Printed Circuit Board (PCB). One general practice is to provide bypass capacitors in various places on the board such as between the power and ground lines to reduce the noise, as well as on special modules like the ADC module. Furthermore, typically a 1uF low ESR capacitor should be located close to the microchip to stabilize the core logic’s internal voltage regulator. Generally the bypass capacitors are included at two places on a board, at the power supply, and at each active device (analog and digital). If the bandwidth of the device is about 1 MHz a 1uF will reduce injected noise dramatically. If the bandwidth is above 10 MHz a 0.1 uF capacitor is appropriate. In between those frequencies both or either one could be used. Furthermore, the bypass capacitor should be placed as close to the power supply pin of the device. If two bypass capacitors are used, the smaller one should be placed closer to the device pin and the lead length of the capacitor should be as short as possible.

Typically the power and ground traces are 40-60 millimeters wide decreasing down to 12 millimeters when connecting to the actual microcontroller (power traces are the widest on the board). For any analog signals in the millivolt range shorter lines are used, and they are further shielded from noise and environmental effects. The goal is to have all signal traces as short as possible whether digital or analog. Finally, the placement of components on the board also has an effect on the stability of the circuit. The devices on the board should be separated into two categories, high speed (greater than 40 MHz) and low speed. Furthermore, those categories should be further separated into three sub-
categories of pure digital, pure analog, and mixed signal. The high frequency components should be placed as close as possible to the connector of the board.

5.8 ADBS Microcontroller Circuit

The circuits presented in Chapter 5 all support the operation of the microcontroller (dsPIC30F2011). Primarily in testing environments the microcontroller is simply placed on a breadboard with the surrounding circuitry arranged around it in any fashion. However, if one were interested in producing a printed circuit board, the following figure shows the layout of the silkscreen, top copper level, and lower copper level.

![Figure 5.8: PCB Layout for ADBS Microcontroller Circuit](image-url)
As it can be seen from the simplistic layout, there is a place for two microcontrollers as well as the op-amp and the transistor IC’s. The IC used for the transistors is the ALD1115 which is composed of an NMOS and PMOS transistor. These five IC’s are on the right side of the PCB layout. Since there are five IC’s there exist 10 transistors which are used to facilitate the operation of the microchip. The secondary microchip is provided for any expansion that may be needed by the researcher using the device. Even for our lab use, on the breadboard there exist two microcontrollers in the same fashion as above. The rest of the PCB consists of the MAX477 op-amp, Zener diode, resistors, and the connectors for wires (in order to connect to the PC and power supply). The schematic for the above layout is provided in Figure 5.9. As can be seen in the schematic, the basic circuits presented throughout the chapter are all present in the schematic. The secondary microcontroller is left out of the schematic due to the fact that it was not needed for the one contact that is the focus of this dissertation. As described previously the transistors serve as simple pass gates for the signals that originate from the microcontroller.
Figure 5.9: ADBS Microcontroller Circuit Schematic
CHAPTER 6
Discussion

6.1 Hardware Design Discussion

The device that will be used in the laboratory setting in order to test the optimization algorithms consists of the various circuits discussed in chapters 4 and 5. Clearly, the portion of the hardware where much of the efforts were concentrated is the microcontroller that is programmed with the optimization algorithm. The main purpose of the surrounding circuitry is to either facilitate the inputs to the microcontroller or to generate the proper output from the stimulating contact. In order to input the LFP data into the microcontroller for further processing the LFP voltage must first be offset in circuitry due to the inability of the microcontroller to sense negative voltages. In order for the microcontroller to operate reliably the voltage supply must be regulated by the voltage regulation circuitry. When attempting to sense the impedance from the contact the impedance sensing circuitry is necessary. On the output side of the microcontroller the PWM wave that is generated must be filtered by the analog post filter, and this stimulation waveform must be connected to negative voltage generation circuitry in order to produce negative voltages when needed. None of the peripheral circuitry provides any intelligence to the ADBS device, nor does it contribute to the algorithms in any way. The only purpose of the peripheral circuitry is to allow the processor to properly sense the voltages from its environment, and properly stimulate the contact when needed.

Since the ADBS hardware developed is intended to be used in a laboratory environment (until clinical tests are completed), the use of the MPLAB IDE software in conjunction with the device is helpful in many respects. First of all, the MPLAB software
offered from Microchip easily connects to the ADBS device providing multiple windows in a user friendly environment that can be used to monitor key registers in the dsPIC30F2011 in real time. Furthermore, due to the fact that the processor is programmed in C, the MPLAB software serves as the user interface for the device. As the researcher uses the ADBS device on live subjects the microcontroller’s code can be altered in real-time if the algorithm used to optimize the parameters needs to be enhanced. It was found that this feature of the device interface was extremely helpful when testing the device even in the commissioning phase. Another useful feature of the ADBS device that was constructed was the secondary processor that is provided with the primary processor. This processor can be used for a multitude of different tasks including data storage or even board expansion if auxiliary inputs/outputs are needed in the future. This processor can even be connected to simple switches or other outputs if needed. Due to the connection to the Microchip software on the PC (MPLAB), none of these features are needed initially, however the option is available for the researcher using the ADBS device.

There are substantial advantages that the ADBS device possesses over the standard DBS stimulators offered by Medtronic for clinical testing. First of all, (and most importantly) the code written for this device is open source, nothing is write protected on the microchip (purposely). The reason for doing so is to allow the researcher the ability to modify the firmware of the microchip in order to expand the capabilities of the device not only in software but also in hardware. The standard devices offered from the large medical device companies offer the stimulation of up to 4 contacts (not simultaneously) in a bi-lateral fashion (if necessary). This entails that when stimulating bi-laterally, two
contacts can be stimulated at any given time. With microchip’s line of DSP chips the I/O pin count is adequate for any custom electrode a researcher may develop. Using our software alongside our firmware (for the DSP line of microchips) the researcher can utilize a device that stimulates tens of contacts at a time (if this is proven to be clinically effective). One downside of multiple contact stimulation is the power drain provided by each additional stimulating contact. With the battery technology available today, if one were to stimulate tens of contacts at a time the battery pack would have to be changed regularly (which is not practical). However, in a laboratory setting (which is the intended purpose of this device) the novel algorithms for DBS can be developed while the battery technology reaches the desired point which will allow implantation of a device that stimulates many contacts simultaneously (given multiple contact stimulation is an effective practice).

6.2 Software Design Discussion

When examining the software written in C much of the code that is written involves setting parameters of the microcontroller which control different aspects of the microcontroller such as the I/O ports and processor speed. Besides the basic commissioning of the microcontroller there exists code which is meant to interface the microcontroller to the researcher using the ADBS chip. This code allows the researcher to vary the stimulation parameters or to adjust different variables involved in the ADBS process. Much of the code written in assembly language was used for filtering purposes or for vector processing using DSP instructions integrated in the microcontroller. The remainder of the programming in C involves the algorithm required to run ALOPEX within the optimization procedure. The advantage of using the MPLAB environment
when running the code is that the debugging features allow the code to be executed one instruction at a time. Besides the convenient debugging, the MPLAB IDE allows the user the ability to time how long the processor takes to execute certain areas of code. It was found that when running the processor at the speeds allowed by the RC oscillator, the algorithm’s speed can be evaluated even when varying the number of iterations requested by the researcher using the ADBS device.

The robustness of the code is inherent to the structure of the code itself. Besides the use of the secondary processor to monitor the activity of the primary processor, each of the processors has the watch dog timer enabled in the event that a fault can be detected even faster. With the sequence of the code following the model of a state machine the code structure is easily followed by outside researchers with little training on the device. The operation of the ADBS device is controlled by the use of the mouse in the MPLAB environment, and is user friendly with the output window in the MPLAB environment.

Perhaps the largest benefit the ADBS device provides to the user is the ability to optimize the stimulation parameters. As discussed earlier, the different stimulation combinations available consume much time when performing an exhaustive search to find the optimal stimulation parameters. Instead of focusing on optimizing the parameters for a single contact the researcher using this device will have the ability to focus on other strategies such as stimulating different sets of contacts in different patterns. Any alteration to the standard DBS strategy can be viewed as a different parameter which is altered. For example, instead of only optimizing the standard three stimulation parameters a fourth parameter can be added for the number of contacts which are stimulated. A fifth parameter that can be added to the process and optimized using
ALOPEX can be an orientation parameter which is user defined based on the pattern with which the contacts are stimulated (for a multi-contact scenario). ALOPEX facilitates the clinical testing for any researcher involved with DBS for Parkinson’s (or other diseases/disorders where DBS is a viable treatment). Any researcher can theorize the different parameters which contribute to the reaction of the brain to DBS, and these parameters can be optimized using our software (on the ADBS device). The testing efficacy of the device is limited only to the creativity of the researcher using the device.

6.3 VLSI Design Discussion

The processor that was designed in VLSI is used in conjunction with the VLSI versions of the PWM module and ADC module. In section 5.1 the hardware was designed in a manner which allowed for the microcontroller to control the peripheral circuitry in order to execute the ADBS algorithm. In a similar manner, for the VLSI design of the ADBS chip the main portion of the hardware consists of the main processor which connects to the peripheral hardware (PWM Module and ADC Module). The PWM Module and ADC Module are intertwined (as described in the previous chapter), and both need control from the main processor. In general, the peripheral hardware is used solely for interfacing to the contacts of the electrode (both in the sensing and stimulation sense).

When observing the simulation of the processor it is clear that the clock speed of 100 MHz is used without any problems. Each instruction that is executed in the simulation is executed with the expected timing and accuracy. It is worth noting that this implies that the processor will run in the same fashion at lower clock speeds. Higher clock speeds could have been simulated, however even the processor speed of 100 MHz is beyond what is needed for this application. Once the exact nature of the ADBS device
is known through clinical testing and the proper funds are acquired, then this design can be laid out and fabricated. The issues that cannot be foreseen at this point all relate to the interface with the contacts (the electrode). The highest number of contacts that can be used clinically will be determined during clinical trials. Once the highest contact number is determined, the correct number of I/O pads can be laid out alongside the ADBS processor (which is already completed).
CHAPTER 7
Summary and Conclusions

Once a live specimen is provided for testing purposes novel clinical research can be conducted with the ADBS device. Not only can the optimization abilities of the device be used, but the researcher can experiment with different types of waveforms in order to see if an alternate waveform can be produced that is more effective. Another group has begun research into this topic [85] and have claimed that alternate waveforms can be used besides the typical pulse-trains used in DBS. Since the ADBS device is completely open source (unlike the devices made by Medtronic) the researcher will be able to produce any waveform within the electrical confines of the chip. Through experimental work the users of this device can open an entire field within DBS dedicated to alternate stimulating waveforms. For instance, in traditional DBS the stimulation waveform can be characterized by three parameters which can be optimized (frequency, pulse-width, and amplitude). If a researcher desired to find an alternate waveform that was clinically more effective than the square waves used they could characterize the stimulation waveform by a larger number of parameters. For instance, parameters could be added which would allow for the generation of triangular waveforms or sinusoidal waveforms (even complex waveforms composed of an infinite number of waveforms). One could decompose the Fourier analysis of a waveform into different parameters which could be optimized using this algorithm in order to find the ideal waveform for DBS that yielded the best tremor response. Due to the flexibility of ALOPEX, it is possible to find other stimulation waveforms that are more effective than the traditional square waveform used in practice.
If the ADBS chip is ever commercialized, the one portion of the design process that was left out was that of the I/O pad circuitry. It is in the best interest of the designer to use optocouplers integrated on the chip to protect the ADBS device. The opto-isolation circuitry would ensure that any artifact sensed by the chip would not damage the device (leading to the surgical removal of the device). The optocoupler is composed of two devices, a transmitter (typically a gallium arsenide LED) and an optical receiver such as a phototransistor or light-triggered diac [44]. These two devices are electrically isolated allowing only light to be transmitted between them. As is the design of the dsPIC30F2011, the VLSI fabricated ADBS chip should use the same pin as an input and as an output (to conserve space). This general design can be found in any VLSI textbook and has been provided in Figure 7.1 for convenience. Generally the I/O pad size is from 100 to 150 µm². The minimum pitch at which bonding machines operate dictates the minimum spacing of the IO pads (150 to 200 µm).

![Bi-directional I/O pad circuitry](image.png)

**Figure 7.1: Bi-directional I/O pad circuitry [82]**

There are two separate items that have been developed that have their own limits as far as actual use. The first item includes the microcontroller design with surrounding
circuitry that is fully functional and intended for use in a laboratory setting. The second item is the actual VLSI design of the device that maintains the same functionality as the microcontroller design, however is a dedicated piece of hardware for the purposes of ADBS. The limit of the VLSI design of the neurochip is the substantial initial cost of fabricating the chip. For a foundry to construct the hardware necessary to fabricate the custom design the cost is on the order of many thousands of dollars. This option can be explored at a later date if other students continue the project. Although a less dedicated piece of hardware, the microcontroller design is able to perform the same tasks as the neurochip designed in VLSI. However, practically when such an electronic device is implanted into a human subject, considerations such as biocompatibility and battery usage are an issue. As a matter of fact, before such a device can be implanted research must be done regarding efficient power delivery from the battery to the neurochip. Furthermore, novel methods can be used to charge the battery such as miniaturized solar panels implanted behind the eyes which recharge the battery pack that is surgically implanted. At the current moment, the microcontroller design can be utilized in a laboratory setting, possibly on laboratory animals with the proper IRB’s.

A second task at hand for any researcher that wishes to use this device is that of fabricating a suitable electrode. Although standard electrodes can be purchased which can be used on laboratory animals, the full capability of the device can only be shown for the multi-contact solution. This will be done once the single contact solution is demonstrated on animals to be an effective solution for Parkinson’s. A benefit to using microcontrollers for laboratory work is the fact that they are not dedicated pieces of hardware. Algorithms can be tweaked, firmware can be upgraded, and circuitry can be easily added based on
the needs of the researcher in a real laboratory setting. If the current design of the neurochip was fabricated in VLSI and used in a laboratory setting, it would certainly be proven to be useless quickly due to real world considerations that could only be taken into account in a real laboratory setting. These adjustments can be made in a matter of minutes on the ADBS design using microcontrollers. A single transistor change for the VLSI design would require an entirely new chip to be manufactured forcing the researcher to incur unneeded costs multiple times. However, the solution provided thus far is in fact complete, the only task necessary for future researchers to perform is exhaustive testing on animals, and eventually human subjects. The resources of this laboratory at this time do not permit for the substantial expenses of the research required, but if future funding is acquired this is definitely a research path worth pursuing.

The current device is capable of neurostimulation at multiple programmable frequencies, pulse-widths, and amplitudes. This capability is similar to that of the commercially available DBS systems available on the market. Such systems retail for tens of thousands of dollars, and thus the system that was presented in this dissertation can be used as a replacement at the very least. Of even greater importance is the automated nature of the device that should be tested on live subjects (animal first then human). If testing in the laboratory validates the algorithms, this contribution can be utilized by millions of patients that use neurostimulators. The neurostimulation business is currently worth over one billion dollars, and the automation method used for ADBS can be used for every type of stimulation device. The neurostimulation industry is projected to be a 5.2 billion dollar industry by 2012 [84], allowing for the use of the technology presented in this dissertation to have an impact in the medical industry.
Typically neurostimulation is reserved as a last resort for chronically ill patients, and these patients usually only resort to surgery when medications fail. Any patients that opt for neurostimulation are clearly in need of the alleviation of the symptoms of their conditions. The automated system using ALOPEX can help to further alleviate the symptoms of their conditions by choosing the optimal settings for stimulation. Furthermore, the effort exerted by the patient as well as the doctor is reduced due to the automated method in which the device adjusts itself. When a medical professional examines a patient to observe the effects of stimulation, the time taken to do so is enormous compared to the time taken by the ADBS device. Considering each separate combination is tested and recorded in less than a second using ADBS, the hardest working medical professionals would be hard pressed to obtain the optimal parameters in a similar time with the same accuracy. When all this is coupled with the fact that the patient’s optimal comfort is achieved in the quickest manner possible, there are few reasons why the whole industry would not move in the direction of automating the stimulation process.

In summary, a system has been designed, simulated, and tested that can be used by clinical researchers to optimize Deep Brain Stimulation in a laboratory setting. After proper clinical trials are completed, researchers will be able to use this device on a larger sample of subjects, ultimately leading to the fabrication of a device that can be offered to the community of Parkinson’s patients. This device will allow the medical personnel that are responsible for the commissioning of the device to obtain the optimal stimulation characteristics faster than the current methods employed. Although such a device shows great promise, the downfall of any device that is implanted into the human brain is that
the regulations required to bring one such device to market requires an exorbitant amount of money and time, however once these non-technical problems are resolved the community of afflicted patients will be able to benefit greatly from this helpful medical device.
CHAPTER 8

References


APPENDIX A1

Power Spectrum Estimation

After the LFP activity has been obtained from the electrode and saved on the chip, the method of analysis used involves power spectrum estimation. In order to understand what is happening we first discuss spectral density, the concept of the periodogram, and finally Welch’s Method.

(a) Spectral Density

The spectral density is often referred to as the spectrum of the signal (it is basically the frequency content of the signal). For ADBS the concern is with the power spectral density which describes how the power of a signal or time series is distributed with frequency. The power can be the actual physical power or just the squared sum of the value of the signal. If the signal does not have a Fourier transform, the Wiener-Khinchin theorem provides an alternative. The PSD is the Fourier transform of the autocorrelation function (equation A1.1), R(τ) of the signal if the signal is treated as a wide-sense stationary random process (a necessary requirement).

\[ S(f) = \int_{-\infty}^{\infty} R(\tau) e^{-2\pi i f \tau} d\tau \quad (A1.1) \]

In order to obtain the power of the signal at a certain frequency we must integrate over the frequency range for positive and negative frequencies (equation A1.2).
\[ P = \int_{f_1}^{f_2} S(f) df + \int_{-f_2}^{f_1} S(f) df \]  

If given a sequence of time samples, we can estimate the spectral density of a random signal. There are both parametric and non-parametric approaches based on either time-domain or frequency-domain analysis. A common non-parametric technique is the periodogram. Also, the spectral density is usually estimated using Fourier transform methods, but another technique that can be used is Welch’s method.

(b) Periodogram

The periodogram is computed from a finite-length digital sequence using the Fast Fourier Transform (FFT). The raw periodogram is not a good spectral estimate because of spectral bias and the fact that the variance at a given frequency does not decrease as the number of samples used in the computation increases. The spectral bias problem arises from the sharp truncation of the sequence, which can be solved by using windowing first. Smoothing the periodogram can reduce the variance problem. One technique used is the method of averaged periodograms. This technique involves dividing the set of N samples into L sets of M samples, computing the DFT of each set, squaring it to get the power spectral density and compute the average of all of them. This leads to a decrease in the standard deviation as \(1/\sqrt{L}\).

(c) Welch’s Method

Welch’s method is used for estimating the power of a signal vs. frequency while reducing noise compared to the methods it is based on [53]. The method is based on the concept of using periodograms, which convert the signal from the time domain to the frequency domain. It is an improvement over the standard method and the Barlett method because it reduces noise in the estimated power spectra in exchange for reducing the
frequency resolution. Due to the noise caused by imperfect and finite data, the noise reduction from Welch’s method is desired. The Welch method is based on the Bartlett method and differs in two ways:

1. The signal is split into overlapping segments. The original data segment is split up into L data segments of length M, overlapping by D points. If $D = \frac{M}{2}$, the overlap is said to be 50%.

2. The overlapping segments are then windowed: After the data are split up into overlapping segments, the individual L data segments have a window applied to them (in the time domain).

After the above process the periodogram is calculated by computing the DFT and then calculating the squared magnitude of the result. The individual periodograms are then time-averaged, which reduces the variance of the individual power measurements. The end result is an array of power measurements vs. frequency bin. The above steps are used by MATLAB’s `pwelch` command to calculate spectral density. This method results in a power spectrum that is a smoothed version of the original, with less noise. In this way, the Welch method is a way of low pass filtering data. Also, this method is used by MATLAB’s PSD command to calculate spectral density.
APPENDIX A2

Parseval’s Theorem

\[ \int_{-\infty}^{\infty} |x(t)|^2 \, dt = \int_{-\infty}^{\infty} |X(f)|^2 \, df \]  \hspace{1cm} (A2.1)

Where \( X(f) \) equals the Fourier transform of \( x(t) \). The significant lesson taken from this theorem is that the total energy in a time signal summed across all of time is equal to the energy of the same signal in the frequency domain across all frequencies.
APPENDIX A3

Step Response vs. Time of FIR Filter used in ADBS

Figure A3.1: Step Response vs. Time of FIR Filter used in ADBS
Impulse Response of FIR Filter Used in ADBS

Figure A3.2: Impulse Response of FIR Filter Used in ADBS
APPENDIX A4

ALOPEX Supplementary Graphs for Gaussian Example

Figure A4.1: Graph of factor1 as iterations progress

Figure A4.2: Graph of factor2 as iterations progress
Figure A4.3: Graph of first Gaussian noise as iterations progress

Figure A4.4: Graph of second Gaussian noise as iterations progress
Figure A4.5: Graph of ALOPEX tracking the optimal X (Second Simulation of Gaussian Function)

Figure A4.6: Graph of ALOPEX tracking the optimal Y (Second Simulation of Gaussian Function)
Figure A4.7: Graph of ALOPEX tracking the optimal cost function (Second Simulation of Gaussian Function)
APPENDIX A5

ALOPEX Supplementary Graphs (2 Variable Example)

Figure A5.1: Graph of ALOPEX tracking the optimal X (2 Variable Function)

Figure A5.2: Graph of ALOPEX tracking the optimal Y (2 Variable Function)
Figure A5.3: Graph of ALOPEX tracking the optimal cost function (2 Variable Function)
APPENDIX A6

Microcontroller PWM Waves

Figure A6.1: Frequency analysis of post-filtered DBS pulse

Figure A6.2: Graph of 220-microsecond DBS pulse generated by microcontroller after post-filtering
Multi Dimensional ALOPEX Matlab Simulation Code

\[ k = 1:1:10002; \]

\[ \text{std} = 6 \times \exp(-k/1500); \]
\[ \gamma = (-0.5 \times \cos(\pi \times k/10000) + 0.5); \]

```
figure(2);
plot(k, std)
title('Plot of standard deviation');
xlabel('iterations');
ylabel('standard deviation');
```

```
figure(3);
plot(k, gamma)
title('Plot of gamma');
xlabel('iterations');
ylabel('gamma');
```

```
xopt(1) = rand(1) \times \text{sign(rand(1) - 0.5)};
xopt(2) = rand(1) \times \text{sign(rand(1) - 0.5)};
```

```
yopt(1) = rand(1) \times \text{sign(rand(1) - 0.5)};
yopt(2) = rand(1) \times \text{sign(rand(1) - 0.5)};
```

```
zopt(1) = rand(1) \times \text{sign(rand(1) - 0.5)};
zopt(2) = rand(1) \times \text{sign(rand(1) - 0.5)};
```

```
aopt(1) = rand(1) \times \text{sign(rand(1) - 0.5)};
aopt(2) = rand(1) \times \text{sign(rand(1) - 0.5)};
```

```
obopt(1) = rand(1) \times \text{sign(rand(1) - 0.5)};
obopt(2) = rand(1) \times \text{sign(rand(1) - 0.5)};
```

```
copt(1) = rand(1) \times \text{sign(rand(1) - 0.5)};
copt(2) = rand(1) \times \text{sign(rand(1) - 0.5)};
```

```
rcost(1) = xopt(1)^2 + yopt(1)^2 + zopt(1)^2 + aopt(1)^2 + bopt(1)^2 + (copt(1)^2 + 4);
rcost(2) = xopt(2)^2 + yopt(2)^2 + zopt(2)^2 + aopt(1)^2 + bopt(1)^2 + (copt(1)^2 + 4);
```

```
factor1(1) = 0;
factor1(2) = 0;
factor2(1) = 0;
factor2(2) = 0;
factor3(1) = 0;
factor3(2) = 0;
factor4(1) = 0;
factor4(2) = 0;
factor5(1) = 0;
factor5(2) = 0;
```
factor6(1)=0;
factor6(2)=0;

j=0;
p=0;
v=0;
d=0;
t=0;
u=0;

%sqrt(N)*randn will generate a random noise with desired variance of N.
%but sqrt(N) is equal to sigma which is standard deviation v(n)

for n=3:1:10002
    c1(n) = rand(1) .* sign(rand(1) - 0.5);
    elk1(n) = c1(n)*std(n);
    c2(n) = rand(1) .* sign(rand(1) - 0.5);
    elk2(n) = c2(n)*std(n);
    c3(n) = rand(1) .* sign(rand(1) - 0.5);
    elk3(n) = c3(n)*std(n);
    c4(n) = rand(1) .* sign(rand(1) - 0.5);
    elk4(n) = c4(n)*std(n);
    c5(n) = rand(1) .* sign(rand(1) - 0.5);
    elk5(n) = c5(n)*std(n);
    c6(n) = rand(1) .* sign(rand(1) - 0.5);
    elk6(n) = c6(n)*std(n);

    %divided by 15 to be a scaling on the gamma scaling factor to make the
    %solution converge
    factor1(n) = 0.05*(gamma(n)*(xopt(n-1)-xopt(n-2))*(rcost(n-1)-rcost(n-2))+ elk1(n)/20);
    factor2(n) = 0.05*(gamma(n)*(yopt(n-1)-yopt(n-2))*(rcost(n-1)-rcost(n-2))+ elk2(n)/20);
    factor3(n) = 0.05*(gamma(n)*(zopt(n-1)-zopt(n-2))*(rcost(n-1)-rcost(n-2))+ elk3(n)/20);
    factor4(n) = 0.05*(gamma(n)*(aopt(n-1)-aopt(n-2))*(rcost(n-1)-rcost(n-2))+ elk4(n)/20);
    factor5(n) = 0.05*(gamma(n)*(bopt(n-1)-bopt(n-2))*(rcost(n-1)-rcost(n-2))+ elk5(n)/20);
    factor6(n) = 0.05*(gamma(n)*(copt(n-1)-copt(n-2))*(rcost(n-1)-rcost(n-2))+ elk6(n)/20);

    if (abs(factor1(n)) > .1)
        if(factor1(n)>=.1)
            factor1(n) = .1;
        else
            factor1(n) = -.1;
        end
    end

    j=j+1;
end

if (abs(factor2(n)) > .1)
    if(factor2(n)>=.1)
        factor2(n) = .1;
    else
        factor2(n) = -.1;
    end
end

p=p+1;
end
if (abs(factor3(n)) > .1)  
    if (factor3(n) >= .1)  
        factor3(n) = .1;  
    else  
        factor3(n) = -.1;  
    end  

    v = v + 1;  
end

if (abs(factor4(n)) > .1)  
    if (factor4(n) >= .1)  
        factor4(n) = .1;  
    else  
        factor4(n) = -.1;  
    end  

    d = d + 1;  
end

if (abs(factor5(n)) > .1)  
    if (factor5(n) >= .1)  
        factor5(n) = .1;  
    else  
        factor5(n) = -.1;  
    end  

    t = t + 1;  
end

if (abs(factor6(n)) > .1)  
    if (factor6(n) >= .1)  
        factor6(n) = .1;  
    else  
        factor6(n) = -.1;  
    end  

    u = u + 1;  
end

xopt(n) = xopt(n-1) + factor1(n);  
yopt(n) = yopt(n-1) + factor2(n);  
zopt(n) = zopt(n-1) + factor3(n);  
aopt(n) = aopt(n-1) + factor4(n);  
bopt(n) = bopt(n-1) + factor5(n);  
copt(n) = copt(n-1) + factor6(n);  
rcost(n) = xopt(n).^2 + yopt(n).^2 + zopt(n).^2 + aopt(n).^2 + bopt(n).^2 + copt(n).^2 + 4;

end

iter = 1:1:10002;

figure(4);  
plot(iter, xopt)  
title('optimal x');
xlabel('iterations');
ylabel('x');

figure(5);
plot(iter,yopt)
title('optimal y');
xlabel('iterations');
ylabel('y');

figure(6);
plot(iter,zopt)
title('optimal z');
xlabel('iterations');
ylabel('z');

figure(7);
plot(iter,aopt)
title('optimal a');
xlabel('iterations');
ylabel('a');

figure(8);
plot(iter,bopt)
title('optimal b');
xlabel('iterations');
ylabel('b');

figure(9);
plot(iter,copt)
title('optimal c');
xlabel('iterations');
ylabel('c');

figure(10);
plot(iter,rcost)
title('rcost');
xlabel('iterations');
ylabel('rcost');
APPENDIX A8

Matlab Simulations for Multi Dimensional ALOPEX

Figure A8.1: Graph of ALOPEX tracking the optimal x parameter (Function of 6 Variables)

Figure A8.2: Graph of ALOPEX tracking the optimal y parameter (Function of 6 Variables)
Figure A8.3: Graph of ALOPEX tracking the optimal $z$ parameter (Function of 6 Variables)

Figure A8.4: Graph of ALOPEX tracking the optimal $a$ parameter (Function of 6 Variables)
Figure A8.5: Graph of ALOPEX tracking the optimal b parameter (Function of 6 Variables)

Figure A8.6: Graph of ALOPEX tracking the optimal c parameter (Function of 6 Variables)
Figure A8.7: Graph of ALOPEX tracking the cost function (Function of 6 Variables)
APPENDIX A9

Microcontroller Firmware for Parameter Optimization

//concise alopex code

#include "p30f2011.h"
#include "dsp.h"
#include "math.h"
#include "stdlib.h" /* for rand, srand */

extern FIRStruct lpfFilter;

extern volatile int doFilterFlag;
extern volatile unsigned int pw;
extern volatile unsigned int period;
extern volatile unsigned int ampl;

float pwopt1;
float pwopt2;
float pwopt3;
float amplopt1;
float amplopt2;
float amplopt3;
float freqopt1;
float freqopt2;
float freqopt3;

int n,q,p,r,f,h;
float c;
float elk;
float factor1;
float factor2;
float factor3;

volatile unsigned int minpw;
volatile unsigned int minperiod;
volatile unsigned int minampl;

extern volatile fractional inputSignal0[80];
extern volatile fractional filteredSignal0[80];
extern volatile fractional* o0Ptr;

float std;
float gamma1;

extern volatile fractional wavepower1;
extern volatile fractional wavepower2;
extern volatile fractional wavepower3;
typical ranges for pulse-width, frequency, and amplitude

pw: 60-210 microseconds
frequency: 130-185 Hz
Amplitude: 1-3.5 V

Medtronic Soletra model 7426 and Itrel II model 7424 System settings:

pw: 60-450 microseconds in 30 microsecond increments
frequency: 2-185 Hz (5-100 Hz in increments of 5 Hz, and 2, 33, 130, 135, 145, 160, 170, 185)
Amplitude: 0-10.5 V in 0.115 V increments

recommended charge density of 30 microcoulombs per centimeter squared limits the number of combinations
this limits us to having the product of the voltage and pulsewidth less than 900 microseconds*volts
*/

void alopex()
{
    srand(TMR1);

    pw = 9 + 0.002*(rand());
    period = 750 + 1.808*(rand());
    ampl = .001495*(rand());

    // to turn on the stimulation in the code include these two lines
    OC1CONbits.OCM = 6;
    T2CONbits.TON = 1;

    pwopt1 = pw;
    freqopt1 = period;
    amplot1 = ampl;

    // delay for time while it stimulates (roughly a second at 30 Mhz processor clock)
    for(f=0; f<550; f++)
    {
        for(h=0; h<550; h++)
        {
            asm volatile("nop");
        }
        // to turn off the stimulation in the code include these two lines
        OC1CONbits.OCM = 0;
        T2CONbits.TON = 0;

        ADCON1bits.ADON = 1;  // turn ADC on
        while(!doFilterFlag)
        {
            // output "processing"
        }
        doFilterFlag = 0;

        VectorScale(80, &inputSignal0[0],&inputSignal0[0], 0x0180); // scale value is in fractional 1.15 format, 15'th bit is sign bit scaled by 0.5
//vector power calculation

o0Ptr = FIR(80,&filteredSignal0[0],&inputSignal0[0], &lpfFilter);

//should scale vector down by 16 to ensure no overflow

wavepower2 = VectorPower(80,&filteredSignal0[0]);

pw = 9+ 0.002*(rand());
period = 750+1.808*(rand());
ampl = .001495*(rand());

pwopt2 = pw;
freqopt2 = period;
amplopt2 = ampl;

// to turn on the stimulation in the code include these two lines
OC1CONbits.OCM = 6;
T2CONbits.TON = 1;

//delay for time while it stimulates (roughly a second at 1 Mhz processor clock)

for(f=0;f<550;f++)
{for(h=0;h<550;h++)
    {asm volatile("nop");}
}

// to turn off the stimulation in the code include these two lines
OC1CONbits.OCM = 0;
T2CONbits.TON = 0;

//problem area...when the ADC is getting turned on the while loop executes for too long, i gues
//doFilterFlag never gets set high

ADCON1bits.ADON = 1;  //turn ADC on
while(!doFilterFlag)
{
    }

VectorScale(80, &inputSignal0[0],&inputSignal0[0], 0x0180);//scale value is in fractional 1.15 format, 15'th bit is sign bit scaled by 0.5

//vector power calculation

o0Ptr = FIR(80,&filteredSignal0[0],&inputSignal0[0], &lpfFilter);

wavepower1 = VectorPower(80,&filteredSignal0[0]);

for(n=2;n<1000;n++)
{
    std = 6*expf(-n*.001);
gamma1 = (-0.5*cosf(3.14159*n*.001)+0.5);
    if((rand())<16383)
        c = (rand())*.00003;
    else

\[ c = -1 \times (\text{rand}()) \times (0.0003); \]

elk = c * std;

\[
\text{factor1} = 0.05 \times (\text{gamma1} \times (\text{pwopt2} - \text{pwopt1}) \times (\text{wavepower2} - \text{wavepower1}) + \text{elk} \times 0.05);
\]

\[
\text{pwopt1} = \text{pwopt2};
\]

\[
\text{pwopt3} = \text{pwopt2} + \text{factor1};
\]

\[
\text{pwopt2} = \text{pwopt3};
\]

if (\text{rand}()) < 16383
\[\]
\[
\text{c} = (\text{rand}()) \times 0.0003;
\]

else
\[\]
\[
\text{c} = -1 \times (\text{rand}()) \times 0.0003;
\]

elk = c * std;

\[
\text{factor2} = 0.05 \times (\text{gamma1} \times (\text{amplopt2} - \text{amplopt1}) \times (\text{wavepower2} - \text{wavepower1}) + \text{elk} \times 0.05);
\]

\[
\text{amplopt1} = \text{amplopt2};
\]

\[
\text{amplopt3} = \text{amplopt2} + \text{factor2};
\]

\[
\text{amplopt2} = \text{amplopt3};
\]

if (\text{rand}()) < 16383
\[\]
\[
\text{c} = (\text{rand}()) \times 0.0003;
\]

else
\[\]
\[
\text{c} = -1 \times (\text{rand}()) \times 0.0003;
\]

elk = c * std;

\[
\text{factor3} = 0.05 \times (\text{gamma1} \times (\text{freqopt2} - \text{freqopt1}) \times (\text{wavepower2} - \text{wavepower1}) + \text{elk} \times 0.05);
\]

\[
\text{freqopt1} = \text{freqopt2};
\]

\[
\text{freqopt3} = \text{freqopt2} + \text{factor3};
\]

\[
\text{freqopt2} = \text{freqopt3};
\]

if (fabs(factor1) > 0.1)
\[\]
\[
\text{if} (\text{factor1} >= 0.1)
\]
\[
\text{factor1} = 0.1;
\]

else
\[\]
\[
\text{factor1} = -0.1;
\]

\[\]
\[
\text{q} = \text{q} + 1;
\]

\[\]

if (fabs(factor2) > 0.1)
\[\]
\[
\text{if} (\text{factor2} >= 0.1)
\]
\[
\text{factor2} = 0.1;
\]

else
\[\]
\[
\text{factor2} = -0.1;
\]

\[\]
\[
\text{p} = \text{p} + 1;
\]

\[\]

if (fabs(factor3) > 0.1)
\[\]
\[
\text{if} (\text{factor3} >= 0.1)
\]
\[
\]
factor3 = .1;
else
factor3 = -.1;

r = r + 1;
}
pw = pwopt3;
period = freqopt3;
ampl = amplopt3;

// stimulate with the parameters ALOPEX found
// to turn on the stimulation in the code include these two lines
OC1CONbits.OCM = 6;
T2CONbits.TON = 1;

// delay for time while it stimulates (roughly a second at 30 Mhz processor clock)
for(f = 0; f < 550; f++)
    {for(h = 0; h < 550; h++)
        {asm volatile("nop");}
    }

// to turn off the stimulation in the code include these two lines
OC1CONbits.OCM = 0;
T2CONbits.TON = 0;
ADCON1bits.ADON = 1; // turn ADC on

while(!doFilterFlag)
    { // output "processing"
    }
doFilterFlag = 0;

VectorScale(80, &inputSignal0[0], &inputSignal0[0], 0x0180);// scale value is in fractional 1.15 format, 15'th bit is sign bit scaled by 0.5
// vector power calculation
o0Ptr = FIR(80, &filteredSignal0[0], &inputSignal0[0], &lpfFilter);

// should scale vector down by 16 to ensure no overflow
wavepower3 = VectorPower(80, &filteredSignal0[0]);
wavepower1 = wavepower2;
wavepower2 = wavepower3;

} return;
}
APPENDIX A10

VLSI Code for Control Unit (Verilog Code)

```verilog
module control (input [5:0] op, fcn,
    input clock, reset, overflow,
    output reg [2:0] alucontrol,
    output reg [1:0] aluop, pcsource, alusrcb,
    output reg irwrite, regwrite, regdst, alusrca, memtoreg, iord, memread, memwrite,
    output reg pcwritecond, pcwrite, causewrite, epcwrite, intcause,
    output reg [3:0] state, nextstate);

initial
begin
    nextstate = 4'd0;
    irwrite = 1'd1;
end

always @(posedge clock)
if(reset) state <= 4'd0;
else state <= nextstate;

always @(*)
begin
    alusrca <= 1'd0;
    alusrcb <= 2'd0;
    aluop <= 2'd0;
    causewrite <= 1'd0;
    epcwrite <= 1'd0;
    intcause <= 1'd0;
    iord <= 1'd0;
    irwrite <= 1'd0;
    memread <= 1'd0;
    memtoreg <= 1'd0;
    memwrite <= 1'd0;
    pcsource <= 2'd0;
    pcwrite <= 1'd0;
    pcwritecond <= 1'd0;
    regdst <= 1'd0;
    regwrite <= 1'd0;

    case(state)
    4'd0:
        begin
            nextstate <= 4'd1;
            alusrca <= 1'd0;
            alusrcb <= 2'd1;
            aluop <= 2'd0;
            iord <= 1'd0;
            irwrite <= 1'd1;
            memread <= 1'd1;
            pcsource <= 2'd0;
        end
    endcase
end
```

pcwrite <= 1'd1;
end

4'd1:
begin

case(op)
6'd10: nextstate <= 4'd2;
6'd35: nextstate <= 4'd4;
6'd8: nextstate <= 4'd4;
6'd43: nextstate <= 4'd4;
6'd0: nextstate <= 4'd9;
6'd4: nextstate <= 4'd11;
6'd2: nextstate <= 4'd12;
default:nextstate <= 4'd13;
endcase
alusrc <= 1'd0;
alusrcb <= 2'd3;
aluop <= 2'd0;
end

4'd2:
begin
nextstate <= 4'd3;
alusrc <= 1'd1;
alusrcb <= 2'd2;
aluop <= 2'd3;
end

4'd3:
begin

case(overflow)
1'b1: nextstate <= 4'd14;
default:nextstate <= 4'd0;
endcase
regdst <= 1'd0;
regwrite <= 1'd1;
memtoreg <= 1'd0;
end

4'd4:
begin

case(op)
6'd35: nextstate <= 4'd5;
6'd8: nextstate <= 4'd7;
6'd43: nextstate <= 4'd8;
default:nextstate <= 4'd0;
endcase
alusrc <= 1'd1;
alusrcb <= 2'd2;
aluop <= 2'd0;
end

4'd5:
begin
nextstate <= 4'd6;
memread <= 1'd1;
iord <= 1'd1;
end

4'd6:
begin
nextstate <= 4’d0;
regdst <= 1’d0;
regwrite <= 1’d1;
memtoreg <= 1’d1;
end

4’d7:
begin
  case(overflow)
    1’b1:  nextstate <= 4’d14;
    default: nextstate <= 4’d0;
  endcase
  regdst <= 1’d0;
  regwrite <= 1’d1;
  memtoreg <= 1’d0;
end

4’d8:
begin
  nextstate <= 4’d0;
  memwrite <= 1’d1;
  iord <= 1’d1;
end

4’d9:
begin
  nextstate <= 4’d10;
  alusrc <= 1’d1;
  alusrch <= 2’d0;
  aluop <= 2’d2;
end

4’d10:
begin
  case(overflow)
    1’b1:  nextstate <= 4’d14;
    default: nextstate <= 4’d0;
  endcase
  regdst <= 1’d1;
  regwrite <= 1’d1;
  memtoreg <= 1’d0;
end

4’d11:
begin
  nextstate <= 4’d0;
  alusrc <= 1’d1;
  alusrch <= 2’d0;
  aluop <= 2’d1;
  pcwritecond <= 1’d1;
  pcsource <= 2’d1;
end

4’d12:
begin
  nextstate <= 4’d0;
  pcwrite <= 1’d1;
  pcsource <= 2’d2;
end

4’d13:
begin
nextstate <= 4'd0;
alusrca <= 1'd0;
alusr cb <= 2'd1;
aluop <= 2'd1;
causewrite <= 1'd1;
epcwrite <= 1'd1;
intcause <= 1'd0;
pwrite <= 1'd1;
pcsource <= 2'd3;
end

4'd14:
begin
  nextstate <= 4'd0;
alusrca <= 1'd0;
alusr cb <= 2'd1;
aluop <= 2'd1;
causewrite <= 1'd1;
epcwrite <= 1'd1;
intcause <= 1'd1;
pwrite <= 1'd1;
pcsource <= 2'd3;
end

default: nextstate <= 4'd0;
endcase

case(aluop)
2'd0: alucontrol <= 3'd2;
2'd1: alucontrol <= 3'd6;
default: case(fcn)
  6'd24: alucontrol <= 3'd0;
  6'd26: alucontrol <= 3'd1;
  6'd32: alucontrol <= 3'd2;
  6'd34: alucontrol <= 3'd6;
  6'd42: alucontrol <= 3'd7;
default: alucontrol <= 3'd2;
endcase
end

case(fcn)
6'd24: alucontrol <= 3'd0;
6'd26: alucontrol <= 3'd1;
6'd32: alucontrol <= 3'd2;
6'd34: alucontrol <= 3'd6;
6'd42: alucontrol <= 3'd7;
default: alucontrol <= 3'd2;
endcase
end

endmodule
Curriculum Vitae

Emir Elkholy

May 2011  PhD in Electrical/Biomedical Engineering
Rutgers University, School of Engineering, New Brunswick, NJ

May 2007  M.S. in Electrical Engineering
Columbia University, The Fu Foundation School of Engineering,
New York, NY

May 2006  B.S. in Electrical Engineering
Rutgers University, School of Engineering, New Brunswick, NJ

Principal Occupations Held

May 2006-May 2011  Part time Project Engineer
Esco Precision, Inc.

September 2007-May 2011  Full-Time Researcher in the Computational Intelligence
Laboratory under the direction of Dr. Evangelia Tzanakou
(Rutgers University)