

# Formal Verification of Basal Ganglia Model (BGM) of Brain using UPPAAL



By  
**Arooj Nawaz**  
2017-NUST-MS(EE)-09-00000204545

Supervisor  
**Dr. Osman Hasan**  
**Department of Electrical Engineering**

A thesis submitted in partial fulfillment of the requirements for the degree  
of Masters of Science in Electrical Engineering (MS EE)

In  
School of Electrical Engineering and Computer Science,  
National University of Sciences and Technology (NUST),  
Islamabad, Pakistan.

(April 2021)

# Approval

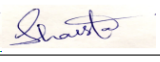
It is certified that the contents and form of the thesis entitled “**Formal Verification of Basal Ganglia Model (BGM) of Brain using UPPAAL**” submitted by **Arooj Nawaz** have been found satisfactory for the requirement of the degree.

Advisor: **Dr. Osman Hasan**

Signature: 

Date: 02-Aug-2019

Committee Member 1: **Dr. Shaista Jabeen**

Signature: 

Date: 01-Aug-2019

Committee Member 2: **Dr. Sidra Sultana**

Signature: 

Date: 02-Aug-2019

Committee Member 3: **Dr. Hassan Aqeel**

Signature: 

Date: 01-Aug-2019

# Dedication

I dedicate this thesis to my parents and family members who always motivated me in achieving goals throughout my life.

# Certificate of Originality

I hereby declare that this submission is my own work and to the best of my knowledge it contains no materials previously published or written by another person, nor material which to a substantial extent has been accepted for the award of any degree or diploma at NUST SEECS or at any other educational institute, except where due acknowledgement has been made in the thesis. Any contribution made to the research by others, with whom I have worked at NUST SEECS or elsewhere, is explicitly acknowledged in the thesis.

I also declare that the intellectual content of this thesis is the product of my own work, except for the assistance from others in the project's design and conception or in style, presentation and linguistics which has been acknowledged.

Author Name: Arooj Nawaz

Signature:  \_\_\_\_\_

# Acknowledgment

I am very grateful to Allah Almighty who always made a path for me even in the dark.

I would like to thank my supervisor, Dr. Osman Hasan, who gave me opportunity to work under his supervision. Thanks for his full time support, guidance and kindness. I am thankful to Dr. Shaista Jabeen for her guidance throughout my thesis. I would like to thank my parents and family members for their prayers and support. After that, I am very thankful to all my dear ones who always encouraged and supported me.

# Table of Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Background . . . . .	1
1.1.1	Parkinson’s Disease Treatment . . . . .	2
1.1.2	Open-Loop and Closed-Loop DBS . . . . .	2
1.2	Literature Review . . . . .	3
1.3	Problem Statement and our Proposed Solution . . . . .	3
1.4	Outline of Thesis . . . . .	4
<b>2</b>	<b>Preliminaries</b>	<b>5</b>
2.1	Model Checking . . . . .	5
2.2	UPPAAL Model Checker . . . . .	5
2.2.1	Timed Automata . . . . .	5
2.2.2	Queries . . . . .	6
<b>3</b>	<b>BG Implementation using Hodgkin-Huxley Neuron Model</b>	<b>8</b>
3.1	Hodgkin-Huxley Neuron Model . . . . .	8
3.2	BG Model . . . . .	8
3.2.1	Thalamic Neuron Model (TH) . . . . .	9
3.2.2	External Globus Pallidus (GPe) and Internal Globus Pallidus (GPi) Neuron Model . . . . .	10
3.2.3	Subthalamic Nucleus (STN) . . . . .	10
3.3	Experimental Detail and Results . . . . .	10
<b>4</b>	<b>Proposed Methodology</b>	<b>12</b>
4.1	System Overview: . . . . .	12
4.2	Modeling the Basal Ganglia: . . . . .	13
4.3	Modeling the DBS (Timed Automata): . . . . .	16
4.4	Verifying the DBS requirements . . . . .	18
4.4.1	Safety . . . . .	18
4.4.2	Liveness . . . . .	20

4.4.3	Supremum and infimum Queries for Time and Power Analysis . . . . .	20
4.4.4	Deadlock and Reachability . . . . .	23
4.5	Case Study for DBS Design and Parameter Selection . . . . .	23
4.5.1	Power comparison of open-loop and closed-loop DBS: .	24
4.5.2	Effect of programming Algorithm: . . . . .	25
<b>5</b>	<b>Conclusions</b>	<b>29</b>
5.1	Summary . . . . .	29
5.2	Future Work . . . . .	29

# List of Tables

4.1	Mean Firing Rate Values of Different BG Regions . . . . .	14
4.2	Power and Time analysis result for different range of stimulation parameters . . . . .	22
4.3	Time Analysis of Algorithms A and B. “Table. 4.3a and 4.3b” show the result of inf and sup queries with different stimulation parameter ranges for Algorithms A and B, respectively. From the overall comparison between these two algorithms, it can easily be observed that Algorithm B takes less time to achieve stability as compared to Algorithm A but stability is not always guaranteed in Algorithm B as compared to Algorithm A. . . . .	27



# List of Figures

3.1	BGM network with components modeling the basal ganglia regions. . . . .	9
3.2	Thalamus (TH) and Globus Pallidus Internus (GPi) cell neural activity pattern for Healthy and PD brain . . . . .	11
4.1	An overview of our proposed methodology representing the DBS and BG model communication through some synchronization channels . . . . .	13
4.2	Timed Automaton for the different regions of BG . . . . .	14
4.3	Timed Automaton Modeling the Refractory Period of different regions of BG . . . . .	15
4.4	Timed Automaton of Closed-loop DBS Controller. . . . .	16
4.5	Timed Automaton for Properties Verification . . . . .	18
4.6	Bounded Liveness Queries for Power and Time Analysis . . . . .	21
4.7	Timed Automaton of Open-loop DBS Controller . . . . .	24
4.8	Graph representing the instantaneous power behavior of Open-loop DBS Controller. Total instantaneous power comes out to be 6716uj approximately and average power 23uj approximately for 2.5s. . . . .	25
4.9	Graph representing the instantaneous power behavior of Closed-loop DBS Controller. Total instantaneous power comes out to be 4485uj approximately and average power 14.8uj approximately for 2.5s. . . . .	25
4.10	Example of two different Programming Algorithms for Power and Time Analysis . . . . .	26

# Abstract

Parkinson's disease (PD) is a neurological disorder that affects dopaminergic nerve cells in a specific area of brain called substantia nigra. PD symptoms include muscle rigidity, tremors, and changes in speech and gait. There is no cure for PD but surgical treatments, i.e., Deep Brain Stimulation (DBS), can help in relieving PD symptoms. Traditionally, DBS is done in an open-loop manner, where stimulation/pacing is always ON, irrespective of the patient need. As a consequence, patients can feel some side effects due to the continuous high frequency stimulation. Therefore, closed-loop DBS is preferred as it allows adjusting stimulation according to the patient need, and it consumes less power as compared to open-loop DBS. However, selection of an appropriate biomarker for closing the feedback loop is a major challenge in close-loop DBS. In this thesis, we propose to utilize model checking, i.e., a formal verification technique used to exhaustively explore the complete state space of a system, for analyzing DBS controllers. We model the basal ganglia (BG) region from hybrid automaton to timed automata using timed abstraction. Thereafter, we analyze the timed automata of the closed-loop DBS controller in response to the BG model. Furthermore, we formally verified the closed-loop DBS using timed computation tree logic (TCTL) properties, i.e., safety, liveness and deadlock. We show that closed-loop DBS significantly outperforms existing open-loop DBS controller in terms of energy efficiency. In order to demonstrate the practical effectiveness of our work, we formally analyze the closed-loop DBS for power efficiency and time behavior with two different algorithms, i.e., Algorithms A and B. Our results demonstrate that the closed-loop DBS running the Algorithm B is efficient in terms of time and power as compared to Algorithm A but stability is not always guaranteed in Algorithm B.

# Chapter 1

## Introduction

### 1.1 Background

Parkinson's disease (PD) is a nervous system disorder that affects the body movement. PD symptoms starts gradually and therefore patients even may not notice them at the early stages. With the passage of time, PD can have a major impact on how patients talk, sleep, think and walk. There are more chances to get PD in the age of sixty or later. After Alzheimer's, PD is considered to be the second major progressive neuro syndrome, which causes vital incapacity that affects patients daily routine task, their families and more importantly can imbalance their health system [1], [2]. One in seven patients with PD is under the age of fifty years, i.e., around 0.3-% of the under fifty years population is affected with PD in developed countries [3]. This frequency increases as age increases, e.g., at the age of sixty this rate rises to 1% and at the age of eighty this rises even more to 3%. The studies in UK reflect that in every 0.1M population, about 180 people are affected with this disease [4]. Parkinson's is probably expected to be a dominant disease with the passage of time as the population age increases [5].

PD originates in the basal ganglia (BG) that cause an imbalance in body movement. The BG is a collection of nuclei situated deeply within the brain and is responsible for motion related activities. Primate BG is composed of five different nuclei that are Striatum, Globus pallidus internus (GPi), Globus pallidus externus (GPE), Subthalamic nucleus (STN) and Substantia nigra (SN). Substantia nigra cells (SNc) play an important role in body movement by releasing a chemical called dopamine to carry out messages around the brain. In PD, SNc start dying, which results in reducing the dopamine levels and thus the patient's brain does not pass messages for regulating body movement. This, in turn, leads to body movement difficulties called akinesia,

muscular stiffness and slackening in physical movement, which is termed as bradykinesia, shakiness in standing position and tremor [6]. More importantly, SNc cannot be replaced with healthy body cells, so when the level of dopamine drops, the brain cannot pass as many messages to control the body as required. To the best of our knowledge, researchers have not been able to develop any remedies for PD, however there are several available ways to manage the symptoms.

### **1.1.1 Parkinson's Disease Treatment**

L-dopa is traditionally used as a medicine for PD patients, as it helps to create dopamine molecules within the BG. However, the chances for L-dopa to successfully enter in dopaminergic neurons within SNc are quite less, typically between 1-5 %. Moreover, it works for a limited time due to the tolerance developed against this drug by people. Surgical options are also available as a treatment for PD. Deep brain stimulation (DBS) is a tested and certified treatment for motor symptoms where electric stimulations are delivered to the BG region of the brain [11,12]. It is a surgical method in which electrodes are implanted to generate pulses of high frequency (greater than 100 Hz) in the GPi or STN [7].

### **1.1.2 Open-Loop and Closed-Loop DBS**

The most commonly used DBS devices work in an open-loop manner where stimulation/pacing is always ON with fixed parameters (e.g., voltage, pulse repetition frequency, pulse width). In such devices, the stimulation parameters are tuned through manual adjustment by the physicians. In this method, the DBS battery drains faster due to the continuous stimulation and requires surgical replacement every two to five years. Furthermore, due to continuous high frequency stimulation, patients can experience some side effects [8], such as speech deficits and cognitive dysfunction.

Closed-loop DBS can cater for these issues of continuous high frequency stimulation side effects [8, 9] and the continuous drainage of battery. In closed-loop DBS, a sensor continuously monitors the patients state through a feedback signal, also termed as biomarker, and delivers stimulation accordingly. However, there are certain challenges in closed-loop DBS such as, selection of an appropriate biomarker reflecting PD symptoms, a suitable reference signal and implementing a controller to adapt to dynamic changes

in the reference signal [15].

## 1.2 Literature Review

A model based design framework to validate different levels of feedback in DBS controllers (open and closed-loop DBS), where BGM is implemented in software and hardware (FPGA), is proposed in [14]. This BGM platform generated the required responses to DBS and the framework can be used for design and test DBS controllers. Similarly, a deep reinforcement learning (RL)-based approach is introduced in [13] for analyzing the DBS controller patterns that are effective in reducing PD symptoms and are energy efficient. The BG region is modeled as a Markov decision process (MDP) and the brain-on-chip (BOC) on an FPGA is used to evaluate the performance. Several studies show that beta band activity can be a suitable feedback signal for closed-loop DBS. However, during voluntary movement, beta oscillations in BG desynchronize [10]. Therefore, a reference signal of constant beta power may not be suitable for DBS controlling mechanism. Thus, to include the ability in the controller to adapt to dynamic changes in the reference signal is quite beneficial. Su et al. [15] proposed the beta band power of GPi neuron that can be used as a biomarker of model state. They used a Proportional-Integral (PI) controller to calculate the current DBS frequency according to the dynamic variations in the beta band power. CTx-BG-Th network's computational model was used to test the closed-loop adjustment of stimulation frequency approach.

## 1.3 Problem Statement and our Proposed Solution

The implantable medical devices, such as DBS controllers, are highly safety-critical due to the dependency of human life on them. Therefore, such systems need to be rigorously analyzed to guarantee robustness and reliability. Traditionally, the analysis of DBS controllers has been conducted using simulation. Due to the sampling based nature of simulation, it is very difficult to guarantee that all bugs or corner cases are identified during the analysis phase. We propose to employ model checking, i.e., a formal verification technique, to verify design (system) and requirements (specifications) for a variety of real-time embedded and safety critical systems [20]. Generally, the

system is represented as an automaton (state space model) and specifications are written in temporal logic. Our DBS behavior can best be modeled as timed automaton due to its critical time nature. So, we used the UPPAAL model checker, that is based on the theory of timed automata. The query language of UPPAAL is a subset of timed computation tree logic (TCTL). There is a dire need of model checking for this problem, because a model checker rigorously explores the complete state space of the DBS controller with the given BG behavior, otherwise a missing test case can lead to a wrong prediction of the battery life or timing behavior. To the best of our knowledge, no formal verification method has been proposed in the context of verifying DBS controllers before.

The main contributions of this thesis are as follows:

1. Timed abstraction of the BG model from Hybrid Automata to Timed Automata.
2. Timed Automata of closed-loop DBS controller to generate relevant behavior in response to BG model.
3. Identification of TCTL properties to verify safety, liveness and deadlock freeness.
4. An approach for the formal power analysis of DBS controllers, i.e., open-loop DBS and closed-loop DBS, and their comparison.
5. A case study to analyze the power and timing behavior of DBS controllers running a specific algorithm.

## 1.4 Outline of Thesis

The rest of the thesis is organized as follows: Section II explains some fundamental details required for the better understanding of the rest of the thesis. Section II presents the Hodgkin-Huxley Neuron model implementation for all the BG region in MATLAB. Section III presents the formal verification of BG and DBS controller using the UPPAAL model checker. Section IV describes the power comparison between open-loop and closed-loop DBS and a case study on the analysis of time and power with two different algorithms. Finally, Section V concludes the thesis.

# Chapter 2

## Preliminaries

### 2.1 Model Checking

Model checking is a technique to verify design and requirements for a variety of real-time embedded and safety critical systems. A model checker exhaustively explores the complete state space of a system to automatically verifies if the given properties hold for the given system, otherwise it generates a trace of the counter example. By observing the generated trace, systems bugs can be easily identified for model correction. State space of a complex system can be very large and can lead to the infamous state-space explosion problem during properties verification, due to limited resources in terms of time and memory. Therefore, complex system models need to be abstracted in order to resolve this problem. There are many model checkers available, like NuSMV, PRISM, SPIN, UPPAAL etc. and we chose UPPAAL due to its distinct features including a user friendly GUI, counter example visualization and ease to model real-time systems.

### 2.2 UPPAAL Model Checker

UPPAAL is a toolbox for modeling, validation and verification of real-time systems, especially in those application areas where timing is critical, e.g., real-time controllers and communication protocols. UPPAAL model checker is based on the theory of timed automata, which is a finite state machine with clocks. The clocks allow us to keep track of continuous time.

#### 2.2.1 Timed Automata

A timed automaton is a tuple  $(C, A, F, f_0, E, I_{nv})$  where:

- $C$  is a set of clocks.
- $A$  is timed automata, i.e., set of all actions, co-actions and the internal  $\tau$ -action.
- $F$  is a finite set of locations.
- $f_0 \in F$  is an initial location.
- $E$  is the set of edges.
- $I_{nv}$  assigns invariants to locations.

In UPPAAL, a system is composed of concurrent processes, where each of them modeled as an automaton. The automaton has a finite set of locations as nodes, and edges as arcs between locations. Transitions are annotated with guards, selections, synchronization and updates. Guards and synchronizations on the transition edge are used to decide when to take a transition. At the time of transition, two updates are possible: reset of clocks or assignment of variables. Hand-shaking synchronization in UPPAAL allows two or more automaton to take a transition at the same time. One automaton transmits a signal using  $a!$  and the other automaton receive that signal using  $a?$ , where “ $a$ ” represents the synchronization channel, “ $!$ ” represents transmission of a signal and “ $?$ ” represents reception of that signal.

### 2.2.2 Queries

Model verification is a critical step in model-checking where properties are written in a formal language. UPPAAL utilizes a simpler version of TCTL properties. UPPAAL query language supports the following types of properties:

- Safety property:
  - $E \Box P$  (Exists globally  $P$ ): There exists a path where query  $P$  is always satisfies.
  - $A \Box P$  (Always globally  $P$ ): For all paths, query  $P$  always satisfies.
- Validation Property:
  - $E \langle \rangle P$  (Possibly): There exists a path at which query  $P$  possibly satisfies.



- Liveness property:
  - $A \langle \rangle P$  (Eventually): For all paths, query P eventually satisfies.
  - $P \rightarrow Q$  (Leads-to): Whenever P satisfies, query Q verifies eventually.
  
- Deadlock Property:
  - $E \langle \rangle \text{deadlock}$ : Exists deadlock.
  - $A \square \text{not deadlock}$ : There is no deadlock.

# Chapter 3

## BG Implementation using Hodgkin-Huxley Neuron Model

In this section, we present Hodgkin-Huxley(HH) neuron model along with implementation of BG region, i.e., healthy and PD brain. This section presents the BG model implementation on MATLAB with results differentiating the PD from healthy brain as reported in [14].

### 3.1 Hodgkin-Huxley Neuron Model

In 1952 Hodgkin and Huxley proposed a conductance-based mathematical model, which explains the ionic actions of neuron system as a system of coupled ODE's [16]. The mathematical equation representing the membrane voltage of a simple neuron is represented as below [16]

$$C_m \frac{dv}{dt} = I - g_k(V_m - v_k) - g_{Na}(V_m - v_{Na}) - g_l(V_m - v_l)$$

where  $I$  is the total membrane current per unit area,  $C_m$  is the membrane capacitance per unit area.  $g_k, g_{Na}$  and  $g_l$  are the potassium, sodium and the leak conductance's per unit area, respectively.  $V_K, V_{Na}$  and  $V_l$  are the potassium, sodium and leak reversal potentials, respectively.

### 3.2 BG Model

The general structure of the model is shown in Fig. 3.1 [14], with all basic components - subthalamic nucleus (STN), thalamus (TH), globus pallidus internus (GPi) and globus pallidus externus (GPe), showing their connections and capturing neural activity of the corresponding BG regions [14].

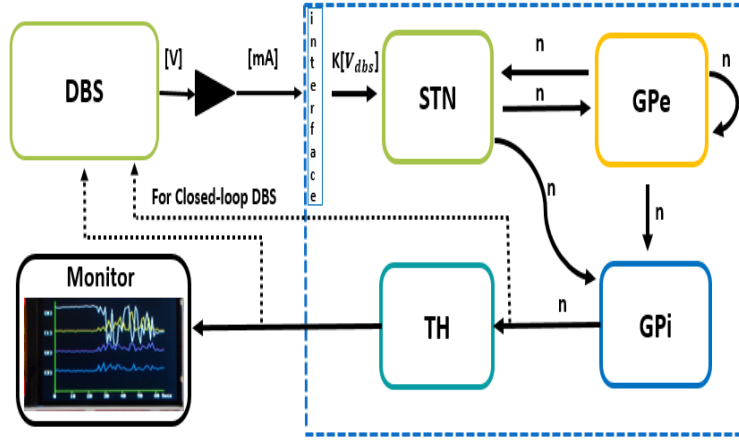


Figure 3.1: BGM network with components modeling the basal ganglia regions.

Note that, despite the fact that TH is actually not a technically part of BG, it reveals actions of GPi and assists in PD detection, and consequently is incorporated into the model.

Each nucleus of BGM consists of  $n$  neurons, where  $n$  is a design parameter, which is modeled using a variation of the Hodgkin-Huxley neuron model [16] based on the region; so, these are named from their origin (i.e., STN, GPe, GPi, TH). The electrical potential of each neuron mainly represents their activity, that we denoted as  $v_j^{TH}$ ,  $v_j^{STN}$ ,  $v_j^{GPi}$  and  $v_j^{GPe}$ , for neurons  $j \in \{1, \dots, n\}$  of the respective region. We provided the final ODE's for each neuron representing the membrane electric potential. The complete set of ODE's for all BG nuclei are given in [14].

### 3.2.1 Thalamic Neuron Model (TH)

The membrane electrical potential of a TH neuron,  $v_{TH}$ , is calculated using [14]

$$C_m \frac{dV_{th}}{dt} = -I_l - I_{Na} - I_K - I_T - I_{gith} + I_{appth}$$

Every TH neuron receives inhibitory input ( $I_{gith}$ ) from a GPi neuron.

### 3.2.2 External Globus Pallidus (GPe) and Internal Globus Pallidus (GPi) Neuron Model

They give intermediate processing between the input neurons (STN) that is directly excited by DBS and output neurons (TH), where the effects of DBS is observed. Therefore, sometimes they are considered as a hidden layer in the model. Still, GPi activity recording is considered important for model validation, because it gives good understanding about the working of the BG. Electrical properties of GPe are given below [14].

$$C_m \frac{dV_{GPe}}{dt} = -I_l - I_{Na} - I_K - I_T - I_{Ca} - I_{ahp} - I_{snge,ampa} - I_{snge,nmda} - I_{gege} - I_{strgpe} + I_{appgpe}$$

Electrical properties of GPi are given below [14].

$$C_m \frac{dV_{GPi}}{dt} = -I_l - I_{Na} - I_K - I_T - I_{Ca} - I_{ahp} - I_{snge} - I_{gege} - I_{strgpe} + I_{appgpi}$$

### 3.2.3 Subthalamic Nucleus (STN)

Every STN nuclei get events from two corresponding GPe nuclei, and stimulations from the DBS device. We can also consider it as system input neurons, because only STN cell get inputs from the DBS controller.

The electric potential emergence is given below as [14]

$$C_m \frac{dV_{STN}}{dt} = -I_a - I_L - I_{Na} - I_K - I_T - I_{Cak} - I_l - I_{gesn} - I_{cosn,ampa} - I_{cosn,nmda} + -I_{dbs}$$

## 3.3 Experimental Detail and Results

In this implementation, the first step is the modeling of Hodgkin Huxley equations for all the nuclei of BG region. There are different numerical methods to solve the Hodgkin-Huxley model. Six different numerical methods have been used by the researchers to solve the arbitrary ordinary differential equation. The numerical methods used are: forward Euler, modified Euler, backward Euler, Runge-Kutta, Adams-Bashforth-Moulton, predictor-corrector and MATLAB ODE45 function. We used the forward Euler's approach for solving and approximating the first order differential equation. We

have created the MATLAB code for all the nuclei of BG region, and solved the Hodgkin-Huxley model with Euler's numerical method.

Thereafter, the second step is to generate and differentiate the behavior of normal brain from PD brain through simulation. It has been reported [14] that PD symptoms are reflected in globus pallidus internal (GPi) and thalamus (TH) neural cell activity. TH cell exhibits regular spiking behavior in the healthy brain, and these spikes become irregular in the PD affected brain and the TH cell cannot fire properly with a regular pattern [14]. This also holds for the GPi cell that in healthy brain GPi exhibits regular spiking behavior, while in PD brain GPi spiking becomes more frequent and grouped (burst-like). We were able to generate and differentiate the healthy behavior from PD as can be seen in Figs. 3.2a & 3.2b, where the irregular spiking behavior of the TH cell is highlighted in red color to show the difference between neural activity in the healthy and PD brain. It can be noticed that the GPi cell produces more bursting in the PD condition as compared to healthy brain in Figs. 3.2c & 3.2d.

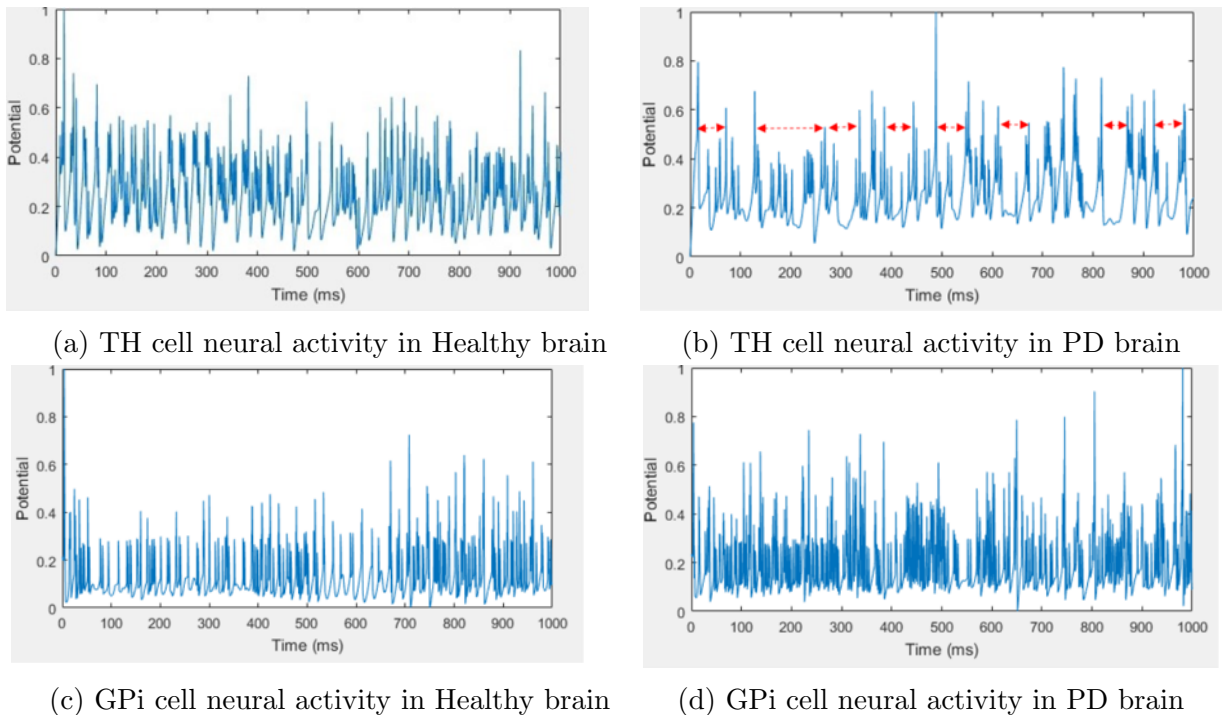


Figure 3.2: Thalamus (TH) and Globus Pallidus Internus (GPi) cell neural activity pattern for Healthy and PD brain

# Chapter 4

## Proposed Methodology

In this section, we present the formal verification of BG and closed-loop DBS controller. BG model is the timed abstraction of hybrid automata to timed automata. Thereafter, we present the timed automaton of closed-loop DBS controller behavior in response to BG. Furthermore, we describe the TCTL properties, i.e., safety, liveness and deadlock freeness, that can be used to verify our models.

### 4.1 System Overview:

We propose a design methodology, where TH neuron behavior and beta power of GPi neuron is monitored to detect the PD condition, as TH neuron pattern and beta power of GPi cell assist in PD detection [14]. Thereafter, if PD condition is detected then stimulations are delivered to STN.

The overview of the closed-loop system is shown in Fig. 4.1, where BG and DBS model behavior can be observed in terms of broadcast channels, i.e., *THget*, *THnotS*, *GPiget*, *BetaP* and *SP*. *THget* indicates TH activation in healthy or PD range, *THnotS* indicates that the TH activation is not sensed at all, *GPiget* indicates that the beta band power lies in the healthy range, *BetaP* indicates that the beta band power lies in the PD range and *SP* indicates pacing from DBS to BG. The BG generates some actions, i.e., *THget!*, *THnotS!*, *GPiget!* and *BetaP!*, representing BG condition that DBS takes as input. Thereafter, DBS processes these signals and generates the pacing action, i.e., *SP!*, to the corresponding nuclei of BG if PD is detected.

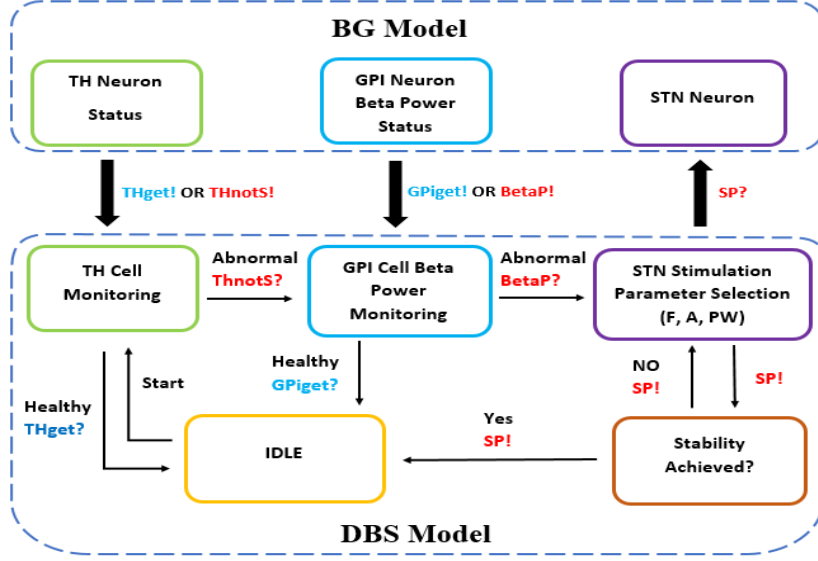


Figure 4.1: An overview of our proposed methodology representing the DBS and BG model communication through some synchronization channels

## 4.2 Modeling the Basal Ganglia:

In order to verify the functionality of the closed-loop DBS controller, we need a model of the BG that can capture how a human brain generates the sensory events. BG can be modeled using non-linear differential equation of Hodgkin-Huxley neuron model [14]. The model described by such non-linear differential equations, consists of multiple continuous state variables and comes under the category of a continuous-time systems. DBS algorithms can be best modeled as a composition of timed-automata, while the relevant features of BG can be described as a network of hybrid automata.

We model each neuron of BG as timed automata by abstracting the hybrid automata, keeping in view the required level of detail for the analysis. As mentioned in [18], a cell/neuron excitation (voltage change with time), upon stimulation, can be partitioned into some distinct phases that are upstroke, repolarization and resting. In each phase, the dynamics can be captured by a linear differential equation and this behavior is described as a hybrid automata. Thereafter, that hybrid automaton is further abstracted to timed automata based upon the timing that a signal takes to travel through a cell chain as mentioned in [18]. We abstracted the BG from hybrid automata to timed automata using the same approach, where timing values (stimulation time interval) for PD and healthy brain are calculated from the data available in literature [14]. Each BG neuron, i.e.,  $TH$ ,  $GPI$ ,  $STN$ , is modeled as

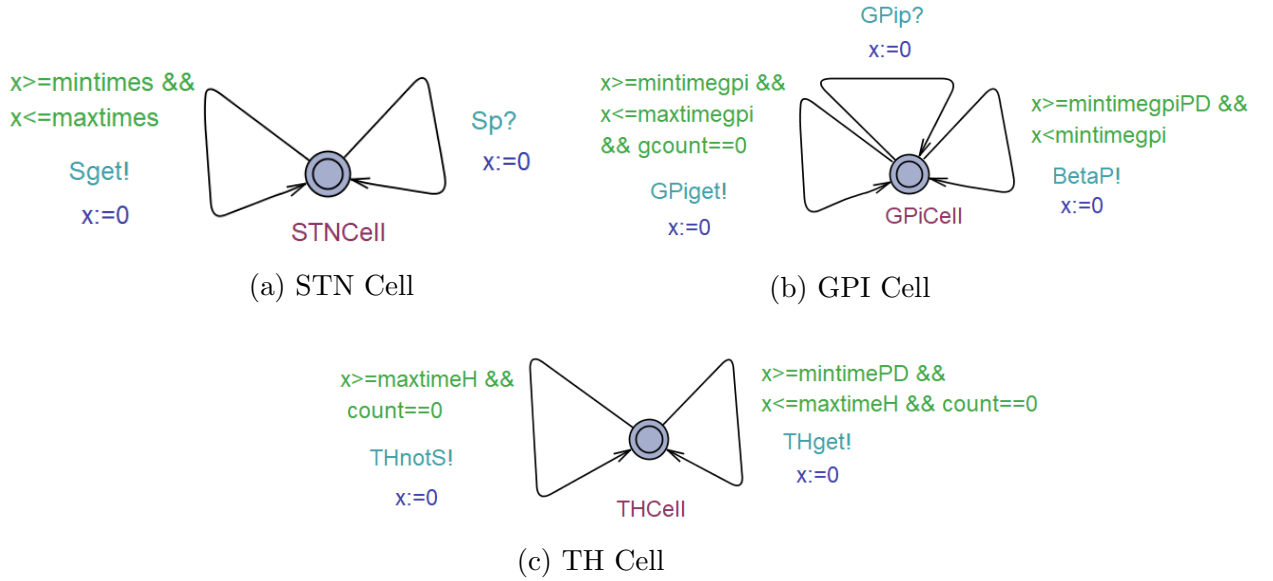


Figure 4.2: Timed Automaton for the different regions of BG

a separate process with a global clock named  $x$ , and some activation signals, i.e.,  $THget!$ ,  $THnotS!$ ,  $GPiget!$ ,  $BetaP!$ ,  $Sget!$ , as shown in Fig. 4.2. Each BG neuron automaton communicates with DBS timed automata using broadcast channels, i.e.,  $Sget!$ ,  $THget!$ ,  $THnotS!$ ,  $GPiget!$  and  $BetaP!$ .

Firing pattern/time detail of these nuclei are mentioned in Table 4.1 for both PD and healthy behavior, where the first two columns represent the mean firing rates available in literature [14] for these nuclei. We calculated the time interval of these spikes from these mean firing rates as presented in the last two columns of Table 4.1. Fig. 4.2a shows a healthy STN cell that relays an activation signal, i.e.,  $Sget!$ , with an interval of [34-111ms]. We aim to analyze TH and GPI cell beta power to detect the PD condition, so we randomly modeled TH and GPI cell such that they can fire non-deterministically in the healthy or PD range. For example, TH neuron can fire in any of these

Table 4.1: Mean Firing Rate Values of Different BG Regions

Cell	Healthy $m_{fr}$ [Hz]	PD $m_{fr}$ [Hz]	Healthy Spikes Interval [ms]	PD Spikes Interval [ms]
STN	[9,29]	[11,41]	[34.4,111.1]	[24.3, 90.9]
GPI	[59.8,101.2]	[76.6,135.4]	[9.8,16.7]	[7.3, 13.1]
TH	[10,20]	[5,166.6]	[50,100]	[6,200]



intervals, i.e., [50-100ms] for healthy and for PD the interval is [6-50ms] and [ $>100$ ms]. Once TH non-deterministically relays its activation signal to DBS, its behavior is analyzed according to the corresponding received signal, i.e., signal activation with respect to the corresponding time interval. GPi is also modeled non-deterministically which means that it can generate healthy or PD behavior by relaying its activation signal, i.e., *GPiget!* or *BetaP!*, where *GPiget!* indicates that the beta band power of GPi neuron lies in the healthy range and *BetaP!* indicates the beta band power value lies in the PD range. Thereby, DBS model takes TH and GPi activation signals as input and delivers stimulations to STN if required, i.e., *SP!*, from DBS to STN. A separate automaton is also developed for modeling the refractory period, i.e., amount of time it takes for an excitable membrane to be ready for a second stimulus once it returns to its resting state after excitation, for each cell of BG as shown in Fig. 4.3. Whenever it senses the STN activation, i.e., *Sget?*, it will go to the *SRP* state followed by the *inter* state, where it remains for the *TSRP* duration, i.e., STN refractory period duration. By doing so, we limit any excessive neuron spikes, i.e., *Sget?* cannot be sensed before the completion of the refractory period and again can be sensed after *TSRP* time duration. The other two neurons, i.e., TH and GPi, are similarly modeled as shown in Fig. 4.3.

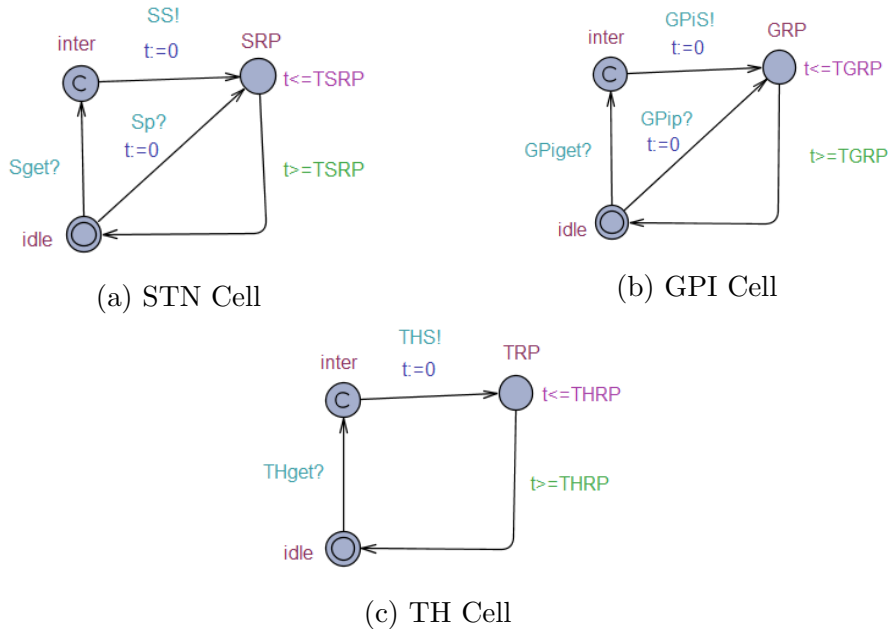


Figure 4.3: Timed Automaton Modeling the Refractory Period of different regions of BG

### 4.3 Modeling the DBS (Timed Automata):

We considered TH neuron pattern to detect any abnormality in BG, as reported in [14], by considering that TH is not a part of BG but is there to just assist in detecting the PD condition. After that, selection of a biomarker to close the feedback loop is quite challenging in the closed-loop system but we used the beta band power of GPi neuron to identify the PD condition [15], since its an appropriate biomarker reflecting PD symptoms. So, in order to detect PD, we used both signals, i.e., TH behavior and beta band power of GPi neuron. The DBS controller takes some input signals from BG, i.e., *THget!*, *THnotS!*, *GPiget!* and *BetaP!*, indicating the brain condition, and it delivers stimulations to STN, i.e., transmits the *SP!* signal to BG, if PD is detected otherwise remains in its *IDLE* state of sensing.

The timed automata of DBS for a closed-loop DBS behavior in response to BG is shown in Fig. 4.4. In this design implementation, the DBS model remains in its initial state named *IDLE* as long as the received BG behavior is healthy. Initially, the DBS controller keeps sensing the TH behavior in its *IDLE* state by receiving the signals from TH automaton, i.e., *THget!* or *THnotS!*. TH automaton is modeled non-deterministically because we want to observe all combinations, i.e., either healthy or PD. The DBS model takes further actions according to the received TH pattern with respect to time as shown in Fig. 4.1, i.e., to transition back to the *IDLE* state if normal TH behavior is observed otherwise transition to the next state where GPi cell beta power is monitored for further actions. Beta band power of GPi is monitored through a signal from GPi automaton, i.e., *GPiget!*, which indicates that the beta band power value lies within the normal range and *BetaP!* indicates that the beta band power value of GPI lies in the PD range. The DBS takes further actions after monitoring the GPi cell condition, i.e., delivers stimula-

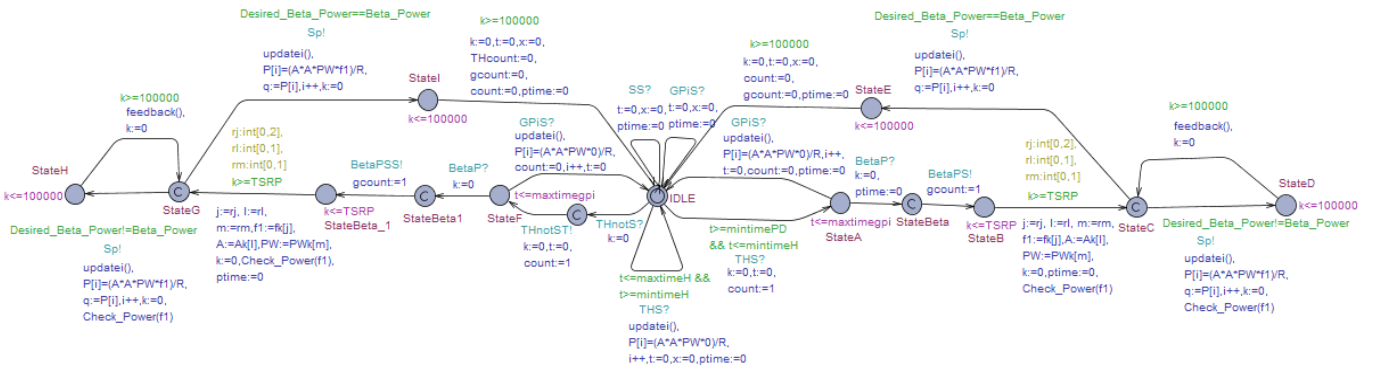


Figure 4.4: Timed Automaton of Closed-loop DBS Controller.

tions to STN if PD is detected otherwise it transitions to the *IDLE* state if healthy behavior is detected. Once PD is detected, stimulations are delivered to STN by relaying the activation signal, i.e., *Sp!*, from DBS to BG. There are certain stimulation parameters that need to be considered while applying stimulation, i.e., frequency, amplitude and pulse width.

After detecting the PD condition, stimulation is applied to STN but the question is how do we ensure that applied stimulation of any random frequency helps to suppress the PD symptoms. Jovanov et al. [14] provided a range of beta band power that can be used to estimate the BG behavior, i.e., healthy or PD. These frequency stimulations that can shift the beta band power range from PD to healthy [15] are quite useful in the context of suppressing PD symptoms. Fei et al. [15] implemented a PI controller, that delivers the stimulation of randomly selected frequency and estimates the beta band power of that applied stimulation. The calculated beta band power is provided as a feedback signal to the PI controller and the error is estimated, as the difference between the desired beta band power (reference signal) and calculated beta band power (feedback signal). Then the PI controller changes the frequency of its current stimulation according to the error, and keeps on changing the stimulation frequency until the error is minimized or until the calculated beta band power (feedback signal) becomes equal to the desired beta band power (reference signal). So, we implemented the same methodology, where DBS delivers the stimulation of randomly selected frequency and calculates the beta power of the applied stimulation. Accordingly, we considered the relation between the applied DBS frequency and beta power of that stimulation frequency from [15].

The DBS controller keeps on changing the stimulation frequency parameter based upon the error between the calculated and desired beta band power until the error is zero. The desired beta band power is known [15] as we considered its value to be 110. Our DBS model keeps on iterating the loop of *StateC* to *StateD* or *StateG* to *StateH* as shown in Fig. 4.4, where two functions are called named *Check\_Power* and *Feedback*. *Check\_Power* is used to calculate the beta band power in response to that selected/applied frequency and the *Feedback* function is used to check the error between the calculated beta band power and the desired beta band power. Once the beta band power of that applied frequency signal becomes equal to the desired beta band power, then the model transitions to *StateE* or *SateI* (stability is achieved at this state). After achieving stability, stimulations are applied for some time interval according to the design parameters of the considered algorithm and transitions back to its *IDLE* state. Once the *IDLE* state is reached, the same procedure is repeated, i.e., to detect PD and apply stimulation and return to the *IDLE* state once stability is achieved.

## 4.4 Verifying the DBS requirements

Model verification is a critical step where the requirements are verified against their real-time embedded system or controller, i.e., DBS in our case, using a formal tool. UPPAAL uses a simpler version of TCTL properties. We formally verified our DBS model requirements using the following properties:

### 4.4.1 Safety

Safety properties are of the form: “something bad will never happen”. We aim to deliver stimulations to STN whenever PD or any abnormality in BG is detected. We made a separate automaton named *P1* as shown in Fig. 4.5a with a local clock named *tt*. *BetaPS* and *Sp* are broadcast channels, i.e., we want to detect these signals whenever they are transmitted by the DBS timed automaton. The clock is reset on the reception of *BetaPS* signal, i.e., when PD is detected, and the clock keeps on counting until STN pacing, i.e., reception of SP signal form DBS automaton. By doing this, we actually count the total time it takes to deliver stimulations to STN after PD detection.

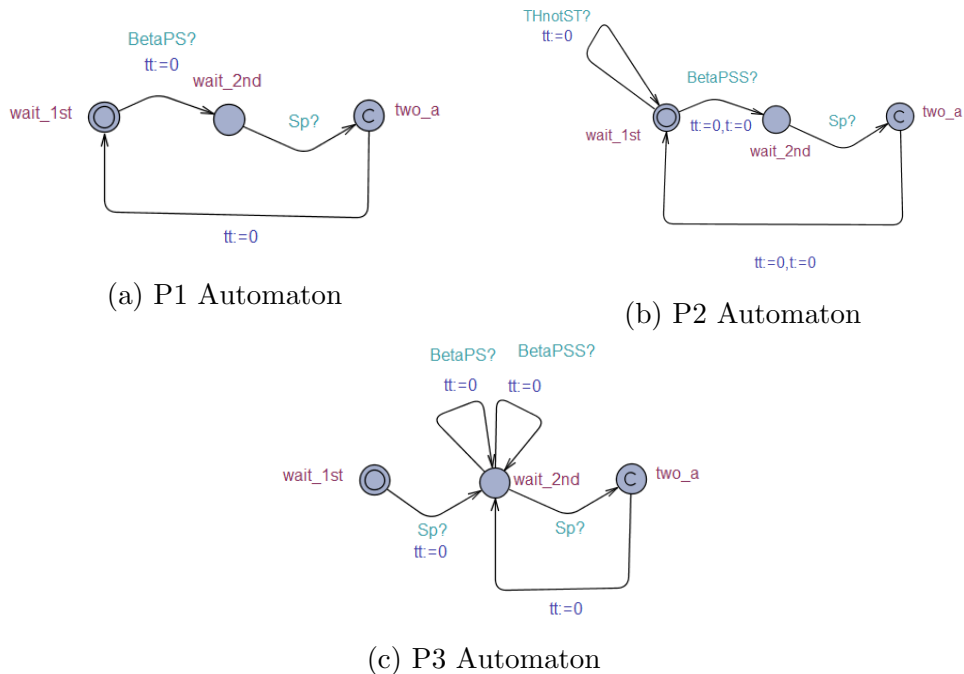


Figure 4.5: Timed Automaton for Properties Verification

Our model delivers stimulations to STN during its normal spike time interval, i.e., not before the refractory period and not after the maximum spike time interval of healthy STN cell.  $TSRP$  indicates the STN refractory period and  $TSLRI$  indicates the longest rate interval, i.e., a longest time interval that a healthy STN neuron can take to fire. Verified safety properties for this requirement are mentioned below

- $A[] P1.two\_a \text{ imply } (P1.tt \geq TSRP \ \&\& \ P1.tt \leq TSLRI)$
- $A[] \text{ not}(P1.two\_a \ \&\& \ P1.tt > TSLRI)$
- $A[] \text{ not}(P1.two\_a \ \&\& \ P1.tt < TSRP)$

By using the above mentioned properties, we verified that stimulations to STN are delivered within  $TSRP$  and  $TSLRI$  interval. We want to make sure that bad state should never happen while fulfilling safety requirements. There are two bad states possible in our case, i.e., STN stimulation delivery before  $TSRP$  or after  $TSLRI$ , so the last two properties are just used to ensure that the bad state never happens.

Another automaton named  $P2$ , as shown in Fig. 4.5b, works with its own local clock named  $tt$ , and two broadcast signals, i.e.,  $BetaPSS$  and  $SP$ . All constraints and requirements are the same as mentioned above in the  $P1$  automaton with just a different broadcast channel, i.e.,  $BetaPSS$  instead of  $BetaPS$ .  $BetaPSS$  in  $P2$  automaton indicates the detection of PD when the TH activation is not sensed at all, while in  $P1$  automaton the  $BetaPS$  signal indicates the detection of PD with abnormal TH activation. So both cases and requirements are the same with just different TH neuron pattern detection. The following safety requirements for this automaton were verified successfully.

- $A[] P2.two\_a \text{ imply } (P2.tt \geq TSRP \ \&\& \ P2.tt \leq TSLRI)$
- $A[] \text{ not}(P2.two\_a \ \&\& \ P2.tt > TSLRI)$
- $A[] \text{ not}(P2.two\_a \ \&\& \ P2.tt < TSRP)$

Another safety constraint is to not have an excessive STN stimulation, i.e., the interval between two consecutive STN pacing  $\in [TSRP, TSLRI]$ . We made another automaton, named  $P3$ , for verifying this safety requirement as shown in Fig. 4.5c. In this design implementation, local clock named  $tt$  is

reset whenever the  $Sp$  signal is received. By doing this, we actually measure the time between two consecutive STN pacing. We verified the following properties for these safety requirements.

- $A[] P3.two\_a \text{ imply } (P3.tt \geq TSRP \ \&\& \ P3.tt \leq TSLRI)$
- $A[] \text{ not}(P3.two\_a \ \&\& \ P3.tt > TSLRI)$
- $A[] \text{ not}(P3.two\_a \ \&\& \ P3.tt < TSRP)$

#### 4.4.2 Liveness

The liveness property states that, under certain conditions, some event will ultimately occur. In our design methodology, the considered event is STN pacing under the PD condition. So, we want to verify that whenever the PD is detected, STN pacing will eventually happen. We used and verified the below mentioned properties in order to satisfy liveness requirements in our design implementation.

- $A \langle \rangle (T1.StateBeta) \text{ imply } (T1.StateE \text{ or } T1.StateD)$
- $A \langle \rangle (T1.StateBeta\_1) \text{ imply } (T1.StateH \text{ or } T1.StateI)$

As can be observed from Fig. 4.4, i.e., the  $T1$  automaton, where PD is detected in  $StateBeta$  or  $StateBeta\_1$  and consequently STN pacing is applied in  $StateE$ ,  $StateD$ ,  $StateH$  and  $StateI$ . So, by the verification of the above-mentioned liveness properties we can conclude that whenever PD is detected the STN pacing will eventually occur.

#### 4.4.3 Supremum and infimum Queries for Time and Power Analysis

In this case study, we aim to analyze power and timing constraints that provide an important design guideline in selecting the DBS parameters. By doing so, we can estimate the instantaneous power of delivered stimulations from DBS to STN. Thereby, the total power consumption of the DBS controller can easily be estimated by using the below mentioned formula [19].

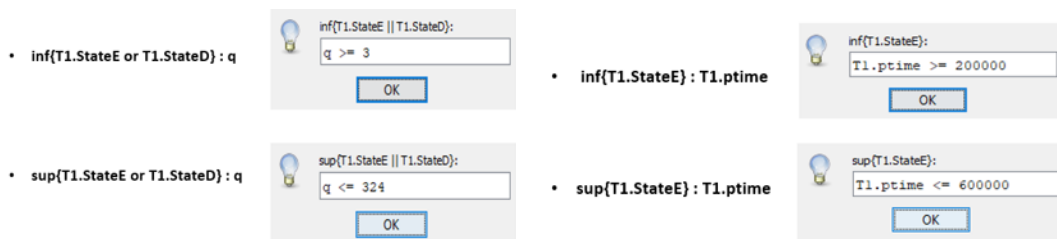
$$P = (A * A * PW * f) / R$$

There are certain parameters that need to be considered for power estimation, i.e., amplitude (A), pulse width (PW), frequency (f) and resistance (R). All parameters except resistance are programmable. Thereby, we considered a range of these parameters as can be observed from Fig. 4.4, where DBS model randomly selects any value of these parameters from the given range of amplitude, pulse width and frequency. We consider a constant value of R corresponding to the tissue impedance faced by electrodes while delivering stimulations [19].

Generally, UPPAAL return the status of queries by indicating whether they are satisfied or not. However, for some queries, some additional information is also provided, like for the supremum and infimum queries. These queries find the supremum (max) or infimum (min) for a given expression for all reachable states that satisfy a particular predicate. We use the following properties to find the maximum and minimum instantaneous power of delivered stimulations, where  $q$  represents delivered instantaneous power.

- $\sup\{\text{T1.StateE or T1.StateD}\} : q$
- $\inf\{\text{T1.StateE or T1.StateD}\} : q$

By using the above-mentioned queries, power bounds can be easily estimated for a given range of parameters. UPPAAL exhaustively explores the whole state space for all possible combinations of these parameters and generates maximum and minimum power value, i.e.,  $\sup$  generates the max power value and  $\inf$  generates the min power value. The get trace option can also be used to identify the parameters responsible for these power values generation. Some results are shown in Fig. 4.6a, for both  $\inf$  and  $\sup$  queries with these parameters - frequency = [60,130,180], amplitude = [1,3] and pulse width = [60,200] with R constant.



(a) Sup and Inf queries for power analysis (b) Sup and Inf queries for time analysis

Figure 4.6: Bounded Liveness Queries for Power and Time Analysis

Stimulation timing analysis is also a critical step for any DBS controller design, i.e., for how much time stimulation are given to achieve stability. We aim to check for how much time these stimulations are given until stability is achieved. Whereas, as mentioned in the last section, stability is achieved when the beta band power of applied signal becomes equal to the desired beta band power. By using these properties, we can find the maximum and minimum time to achieve stability for a given range of values. The get trace option can also be used to trace out the parameters responsible for that time value generation. Properties used for timing analysis are

- $\text{inf}\{\text{T1.StateE}\} : \text{T1.ptime}$
- $\text{sup}\{\text{T1.StateE}\} : \text{T1.ptime}$

Whereas *ptime* is a local clock in the *T1* automaton, that we reset once we achieve stability, i.e., *StateE* or *StateI*. Results for inf and sup query are shown in Fig. 4.6b with the same parameters ranges considered for power analysis.

Power and time analysis results are used to select DBS design and parameters. A detailed depiction of how power and time varies with these parameters is shown in Table 4.2 with different stimulation parameters ranges, i.e., frequency, amplitude and pulse width. Table 4.2 shows the result of inf and sup queries (listed on the extreme left side of the table) for both power and time analysis with stimulation parameters ranges mentioned on the top of the table.

Table 4.2: Power and Time analysis result for different range of stimulation parameters

Properties	$f \in \{60,130,180\}$	$f \in \{80,105,155\}$	$f \in \{90,145,195\}$	$f \in \{105,170,190\}$
	$A \in \{1,3\}$ $PW \in \{60,200\}$	$A \in \{1,3\}$ $PW \in \{60,200\}$	$A \in \{1,2\}$ $PW \in \{50,100\}$	$A \in \{1,2\}$ $PW \in \{50,100\}$
$\text{inf}\{\text{T1.StateE or T1.StateD}\} : q$	3uj	4uj	4uj	10uj
$\text{sup}\{\text{T1.StateE or T1.StateD}\} : q$	324uj	279uj	78uj	152uj
$\text{inf}\{\text{T1.StateE}\} : \text{T1.ptime}$	200000us	200000us	300000us	400000us
$\text{inf}\{\text{T1.StateE}\} : \text{T1.ptime}$	600000us	500000us	900000us	700000us



#### 4.4.4 Deadlock and Reachability

Reachability properties are often used while designing a model to perform sanity checks. For instance, when creating a model of a communication protocol involving a sender and a receiver, it is quite desirable to know whether it is possible for the sender to send a message at all or whether a message can possibly be received. In our design, we intend to ask whether it is possible for the DBS to send stimulations to STN or to achieve stability. We used the following properties to verify reachability requirement:

- $E \langle \rangle T1.StateE$
- $E \langle \rangle T1.StateD$
- $E \langle \rangle T1.StateH$
- $E \langle \rangle GPiComponent.GPiCell$

The deadlock property is used to check that the system is deadlock-free as follows:

- $A[]$  not deadlock
- $E \langle \rangle$  deadlock

### 4.5 Case Study for DBS Design and Parameter Selection

In this section, we present a case study on analyzing open-loop and closed-loop DBS. We show how our proposed models and properties in UPPAAL can be used to compare these two types of controllers in term of energy efficiency. Our results demonstrate that the closed-loop DBS outperforms the existing open-loop DBS in term of energy efficiency. We also present an analysis of stimulation time for a given programming algorithm to select DBS design parameters.

### 4.5.1 Power comparison of open-loop and closed-loop DBS:

An open-loop DBS automaton is shown in Fig. 4.7, with a delay activation signal, i.e.,  $Sp!$ . Open-loop DBS does not sense and send stimulations to STN with fixed parameters. Therefore, we also did not include any sensing or feedback signal in this design model, and stimulations with fixed parameter are delivered to STN as can be observed in Fig. 4.7, i.e.,  $A=1$ ,  $f=115$ ,  $R=1000$  and  $PW=200$ . Instant power is calculated whenever stimulation is delivered to STN, i.e.,  $Sp!$ , by using the following formula.

$$P[i] = (A * A * PW * f) / R$$

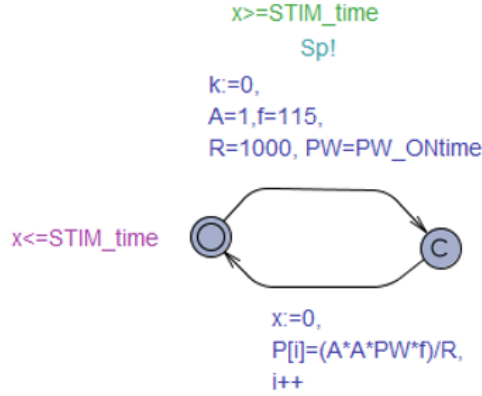


Figure 4.7: Timed Automaton of Open-loop DBS Controller

Where  $i$  indicates the index number of the  $P$  array. We simulated this automaton for 292 iterations that is approximately equal to 2.5s, as shown in Fig. 4.8. Due to fixed parameters, the instant power value remains the same for the whole time and average power comes out to approximately 23 $\mu$ j.

In order to do a fair comparison between both controllers, we apply the same methodology to closed-loop DBS model with the same parameters, i.e.,  $A=1$ ,  $R=1000$ ,  $PW=200$ , except the frequency variation as mentioned in the previous section. We simulated this closed-loop DBS timed automaton for 190 iterations that is approximately equal to 2.5s, as shown in Fig. 4.9. Closed-loop DBS pattern can easily be observed that it randomly selects any frequency from the given range and feedback its error signal, and achieves stability after some time. This behavior is randomly simulated where both healthy and PD behavior are considered non-deterministically, i.e., healthy behavior generated in  $[0-0.2s]$  and PD behavior generated in  $[0.3-0.9s]$  and so

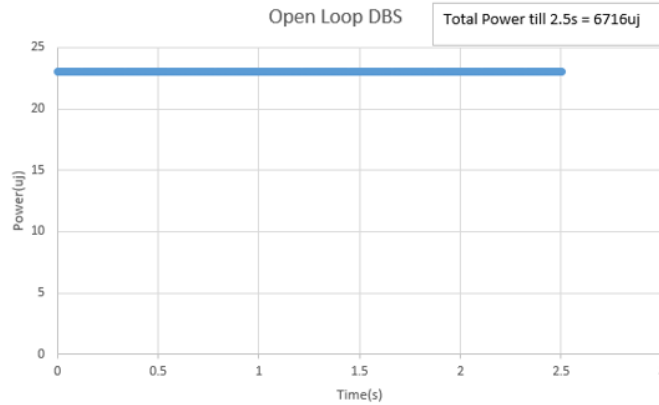


Figure 4.8: Graph representing the instantaneous power behavior of Open-loop DBS Controller. Total instantaneous power comes out to be 6716 $\mu\text{j}$  approximately and average power 23 $\mu\text{j}$  approximately for 2.5s.

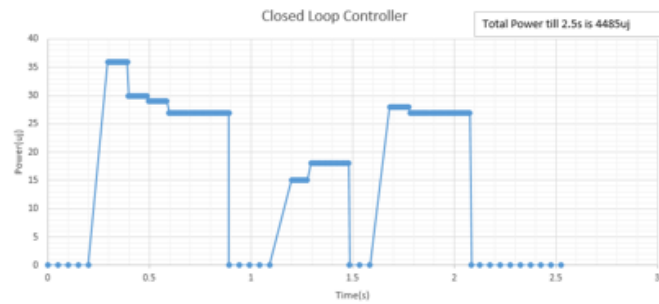


Figure 4.9: Graph representing the instantaneous power behavior of Closed-loop DBS Controller. Total instantaneous power comes out to be 4485 $\mu\text{j}$  approximately and average power 14.8 $\mu\text{j}$  approximately for 2.5s.

on, as can be observed throughout the graph in Fig. 4.9. The average power is approximately equal to 14.8 $\mu\text{j}$ .

As we know in closed-loop DBS, stimulations are only applied when PD is detected in order to prolong the battery life. Open-loop and closed-loop DBS power analysis is made only for 2.5s, and it can be clearly observed that the closed-loop DBS outperforms open loop DBS in terms of energy efficiency and thus prolongs the battery life.

#### 4.5.2 Effect of programming Algorithm:

There can be multiple programming algorithms even for the same hardware, so the selection of an optimal algorithm for DBS can be a challenging task. As mentioned in our design methodology, in Section IV, DBS delivers the stim-

ulation of randomly selected frequency and beta band power of that applied stimulation is calculated. After that, the error is calculated, i.e., difference between the desired beta band power (reference signal) and calculated beta band power of the current stimulation (feedback signal). The DBS controller keeps changing its stimulation frequency until the error is zero, i.e., stability is achieved.

The response of controller and the error correction scheme depends on the design methodology. For this case study, we designed two different algorithms for error calculation and DBS frequency updates as shown in Figs. 4.10a & 4.10b, and we name them as Algorithms A and B, respectively. We used the same parameters and functions as mentioned in Section IV with just a different variation in *feedback* function, i.e., where the error is calculated and frequency is updated. In Algorithm A, DBS chooses random frequency stimulation according to the available frequency range and the frequency is updated with a step size of 5 until the beta band power of that signal becomes equal to the desired beta band power. By doing so, it checks all the available frequency range with a step size of 5. For example, if the DBS initially chooses randomly  $f=100$  and if its beta band power is not equal to the desired beta band power then it will update its frequency with an increment of 5 according to Algorithm A, and the new updated frequency will become  $f=105$ . After the frequency update, it again checks the beta band power of the updated frequency stimulation and keeps on updating its frequency until stability is achieved, i.e., beta band power of that applied frequency signal

```

int feedback()
{
  if (Beta_Power != Desired_Beta_Power)
  {
    f1=f1+5;
  }
  if (f1== 200)
  {
    f1=0;
  }
  else if (Desired_Beta_Power == Beta_Power)
  {
    err=Desired_Beta_Power-Beta_Power;
    f1=f1-err;
  }
  return err;
  return f1;
}

```

(a) Algorithm A

```

int feedback()
{
  if (Beta_Power > Desired_Beta_Power)
  {
    err=Beta_Power-Desired_Beta_Power;
    f1=f1+err;
  }
  else if (Desired_Beta_Power > Beta_Power)
  {
    err=Desired_Beta_Power-Beta_Power;
    f1=f1-err;
  }
  else if (Desired_Beta_Power == Beta_Power)
  {
    err=Desired_Beta_Power-Beta_Power;
    f1=f1-err;
  }
  return err;
  return f1;
}

```

(b) Algorithm B

Figure 4.10: Example of two different Programming Algorithms for Power and Time Analysis

becomes equal to the desired beta band power.

In Algorithm B, DBS chooses random frequency stimulation according to the available frequency range in that particular algorithm and the error is calculated, i.e., difference between the desired beta band power and beta band power in response to the applied signal. The frequency update depends upon the value of error as can be seen in Fig. 4.10b, and the DBS keeps on updating the stimulation frequency until stability is achieved. Comparison of both cases is given in Table 4.3, with the same parameters, i.e., A, PW, F and R, for a fair comparison. We used inf and sup queries for time analysis, i.e., to calculate the upper and lower bounds of time to achieve stability.

It can be clearly observed that Algorithm A takes more time to achieve stability as compared to Algorithm B, because Algorithm A checks all frequency ranges until the stability is achieved while on the other side Algorithm B just updates the frequency based on the error and achieves stability in less time due to less number of iterations. In Algorithm A, stability is guaranteed because it checks all frequency combinations, but in Algorithm B, stability is not always guaranteed because we can never certainly say that the DBS will achieve the desired frequency or not, i.e., a large error may lead to frequent frequency updates and may get stuck in a loop where it iterates over a particular value and thus never gets out of it. So Algorithm A takes more time

Properties	F ∈ {90,140,190} A = 1 PW =100	F ∈ {100,150,195} A = 1 PW =100	F ∈ {80,130,185} A = 1 PW =100	F ∈ {70,100,130} A = 1 PW =100
infT1.StateE : T1.ptime	200000	300000	200000	200000
supT1.StateE : T1.ptime	3300000	3100000	2500000	700000

(a) Algorithm A

Properties	F ∈ {90,140,190} A = 1 PW =100	F ∈ {100,150,195} A = 1 PW =100	F ∈ {80,130,185} A = 1 PW =100	F ∈ {70,100,130} A = 1 PW =100
infT1.StateE : T1.ptime	200000	300000	200000	200000
supT1.StateE : T1.ptime	700000	900000	400000	800000

(b) Algorithm B

Table 4.3: Time Analysis of Algorithms A and B. “Table. 4.3a and 4.3b” show the result of inf and sup queries with different stimulation parameter ranges for Algorithms A and B, respectively. From the overall comparison between these two algorithms, it can easily be observed that Algorithm B takes less time to achieve stability as compared to Algorithm A but stability is not always guaranteed in Algorithm B as compared to Algorithm A.

to achieve stability but stability is always guaranteed, Algorithm B takes less time to achieve stability but stability is not always guaranteed. These findings clearly indicate the usefulness of the proposed analysis of a given algorithm for DBS time, i.e., maximum and minimum time to obtain stability, that can help to design or select different DBS design parameters.

We also conducted the power analysis of Algorithms A and B. Instant power depends upon certain factors, i.e., amplitude, pulse width, frequency and tissue impedance. The result in Table 4.2 clearly demonstrate how instant power value varies by changing any of these parameters. Table 4.2 simply depicts the effect of these parameters on power values. We also considered another case where we made the power comparison of Algorithms A and B. As mentioned previously, Algorithm A takes more time to achieve stability as compared to Algorithm B, which means stimulations are delivered for a longer period of time in order to achieve stability in Algorithm A, that in turn leads to more power consumption for a specific time duration as compared to Algorithm B. So, we can say that Algorithm B is best if we consider the power aspect only, but at the same time stability is not always guaranteed in Algorithm B, so there are pros and cons for both options and our approach allows us to choose the best option according to the given requirements.

# Chapter 5

## Conclusions

### 5.1 Summary

In this thesis, we presented a verification methodology to formally analyze the DBS controllers for PD in response to BG activity, i.e., for analyzing the effect of design parameters on power and timing analysis of DBS controller. In order to verify the functionality of closed-loop DBS controller, we developed a model of BG by abstracting the hybrid automaton to timed automata. Thereafter, we developed a closed-loop DBS controller model and studied its behavior in response to BG model. We then formally verified the closed-loop DBS controller requirements using TCTL properties, i.e., safety, liveness and deadlock. We also developed an open-loop DBS model to compare both open and closed loop controllers in terms of energy efficiency and our results demonstrate that the closed-loop DBS outperforms the open loop DBS in terms of energy efficiency and prolongs the DBS battery life. To illustrate the effectiveness of the proposed methodology, we analyzed two algorithms with nominal parameter values to see their effect on DBS power consumption or stimulation time. The analysis results were found to be quite useful in the context of DBS programming algorithms and design parameter selection.

### 5.2 Future Work

In this thesis, we verified the closed-loop DBS requirements by abstracting the BG (can best be modeled as non-linear differential equation of H-H neuron model) as otherwise we observed the state-space explosion problem during properties verification, due to limited resources in terms of time and memory. Therefore, we abstracted the complex BG model from hybrid automata to timed automata, in order to resolve this problem.

The DBS design algorithm to detect the PD behavior and stimulation pattern of DBS to BG are the main design steps of any DBS controller that can also vary from manufacturer to manufacturer. The design of CL-DBS was formulated based on the available data and resources. A case study was designed based on possible real scenarios as the algorithms of commercially available DBS controllers are not available. The purpose of this case study is to analyze a given DBS controller behavior in terms of time and power with given stimulation parameters. As future work, we would like to evaluate the efficiency of any DBS with given algorithm because the ultimate goal for a pacemaker is to maintain the efficiency of the BG behavior.



# Bibliography

- [1] R. L. Nussbaum, C. E. Ellis. "Alzheimer's disease and Parkinson's disease." *The New England journal of medicine* vol. 348,14 (2003): 1356-64. doi:10.1056/NEJM2003ra020003
- [2] E. A. Chrischilles, L. M. Rubenstein, M. D. Voelker, R. B. Wallace, R. L. Rodnitzky. "The health burdens of Parkinson's disease." *Movement disorders : official journal of the Movement Disorder Society* vol. 13,3 (1998): 406-13. doi:10.1002/mds.870130306
- [3] L. M. de Lau, M. M. Breteler. "Epidemiology of Parkinson's disease." *The Lancet. Neurology* vol. 5,6 (2006): 525-35. doi:10.1016/S1474-4422(06)70471-9
- [4] National Collaborating Centre for Chronic Conditions (UK). *Parkinson's Disease: National Clinical Guideline for Diagnosis and Management in Primary and Secondary Care*. London: Royal College of Physicians (UK); 2006. PMID: 21089238.
- [5] L. J. Findley. "The economic impact of Parkinson's disease." *Parkinsonism related disorders* vol. 13 (2007): S8-S12. doi:10.1016/j.parkreldis.2007.06.003
- [6] J. Jankovic. "Parkinson's disease: clinical features and diagnosis." *Journal of neurology, neurosurgery, and psychiatry* vol. 79,4 (2008): 368-76. doi:10.1136/jnnp.2007.131045
- [7] M. C. Rodriguez-Oroz, J. A. Obeso, A.E Lang, J. L. Houeto, P. Pollak, S. Rehncrona, J. Kulisevsky, A. Albanese, J. Volkmann, M. I. Hariz, N. P. Quinn, J. D. Speelman, J. Guridi, I. Zamarbide, A. Gironell, J. Molet, B. Pascual-Sedano, B. Pidoux, A. M. Bonnet, Y. Agid, J. Xie, A. L. Benabid, A. M. Lozano, J. Saint-Cyr, L. Romito, M. F. Contarino, M. Scerrati, V. Fraix, N. Van Blercom. "Bilateral deep brain stimulation in Parkinson's disease: a multicentre study with 4 years

- follow-up.” *Brain : a journal of neurology* vol. 128,Pt 10 (2005): 2240-9. doi:10.1093/brain/awh571
- [8] P. J. Rossi, A. Gunduz, J. Judy, L. Wilson, A. Machado, J. J. Giordano, W. J. Elias, M. A. Rossi, C. L. Butson, M. D. Fox, C. C. McIntyre, N. Pouratian, N. C. Swann, C. de Hemptinne, R. E. Gross, H. J. Chizeck, M. Tagliati, A. M. Lozano, W. Goodman, J. P. Langevin, R. L. Alterman, U. Akbar, G. A. Gerhardt, W. M. Grill, M. Hallett, T. Herrington, J. Herron, C. van Horne, B. H. Kopell, A. E. Lang, C. Lungu, D. Martinez-Ramirez, A. Y. Mogilner, R. Molina, E. Opri, K. J. Otto, K. G. Oweiss, Y. Pathak, A. Shukla, J. Shute, S. A. Sheth, L. C. Shih, G. K. Steinke , A. I. Tröster, N. Vanegas, K. A. Zaghloul, L. Cendejas-Zaragoza, L. Verhagen, K. D. Foote, M. S. Okun. “Proceedings of the Third Annual Deep Brain Stimulation Think Tank: A Review of Emerging Issues and Technologies.” *Frontiers in neuroscience* vol. 10 119. (2016), doi:10.3389/fnins.2016.00119
- [9] A. O. Hebb, J. J. Zhang, M. H. Mahoor, C. Tsiokos, C. Matlack, H. J. Chizeck, N. Pouratian. “Creating the feedback loop: closed-loop neurostimulation.” *Neurosurgery clinics of North America* vol. 25,1 (2014): 187-204. doi:10.1016/j.nec.2013.08.006
- [10] J. S. Brittain, P. Brown. “Oscillations and the basal ganglia: motor control and beyond.” *NeuroImage* vol. 85 Pt 2,Pt 2 (2014): 637-47. doi:10.1016/j.neuroimage.2013.05.084
- [11] A. L. Benabid. “Deep brain stimulation for Parkinson’s disease.” *Current opinion in neurobiology* vol. 13,6 (2003): 696-706. doi:10.1016/j.conb.2003.11.001
- [12] M. S. Okun, “Deep-brain stimulation for parkinson’s disease,” *New England Journal of Medicine*, vol. 367,16 (2012) : 1529–1538.
- [13] Q. Gao, M. Naumann, I. Jovanov, V. Lesi, K. Kamaravelu, W. M. Grill, M. Pajic. “Model-Based Design of Closed Loop Deep Brain Stimulation Controller using Reinforcement Learning,” 2020 ACM/IEEE 11th International Conference on Cyber-Physical Systems (ICCPS), Sydney, NSW, Australia, 2020, pp. 108-118, doi: 10.1109/ICCPS48487.2020.00018.
- [14] I. Jovanov, M. Naumann, K. Kumaravelu, W. M. Grill and M. Pajic, “Platform for Model-Based Design and Testing for Deep Brain

- Stimulation,” 2018 ACM/IEEE 9th International Conference on Cyber-Physical Systems (ICCPS), Porto, 2018, pp. 263-274, doi: 10.1109/ICCPS.2018.00033.
- [15] F. Su, K. Kumaravelu, J. Wang, W. M. Grill. “Model-Based Evaluation of Closed-Loop Deep Brain Stimulation Controller to Adapt to Dynamic Changes in Reference Signal.” *Frontiers in neuroscience* vol. 13,956. (2019), doi:10.3389/fnins.2019.00956
- [16] A. L. HODGKIN, A. F. HUXLEY. “A quantitative description of membrane current and its application to conduction and excitation in nerve.” *The Journal of physiology* vol. 117,4 (1952): 500-44. doi:10.1113/jphysiol.1952.sp004764
- [17] G. Behrmann, A. David, K. Larsen, P. Pettersson, W. Yi. “Developing uppaal over 15 years.” *Software—Practice & Experience*. 41(2), 133–142 (2011)
- [18] R. Alur, M. Giacobbe, T. Henzinger, K. G. Larsen, M. Mikučionis. Continuous-Time Models for System Design and Analysis. In: *Lecture Notes in Computer Science*, Vol. 10000, 2019, p. 452–477, doi: [https://doi.org/10.1007/978-3-319-91908-9\\_22](https://doi.org/10.1007/978-3-319-91908-9_22)
- [19] H. Cagnan, D. Pedrosa, S. Little, A. Pogosyan, B. Cheeran, T. Aziz, A. Green, J. Fitzgerald, T. Foltynie, P. Limousin, L. Zrinzo, M. Hariz, K. J. Friston, T. Denison, P. Brown. “Stimulating at the right time: phase-specific deep brain stimulation.” *Brain : a journal of neurology* vol. 140,1 (2017): 132-145. doi:10.1093/brain/aww286
- [20] O. Hasan and S. Tahar, “Formal Verification Methods”, In: Mehdi Khosrow-Pour (Eds.), *Encyclopedia of Information Science and Technology*, IGI Global Pub, pp 7162-7170, 2015.