Formal Verification of Deep Brain Stimulation Controllers for Parkinson's Disease Treatment

Arooj Nawaz[1], **Osman Hasan**[1], **Shaista Jabeen**[2]

[1]School of Electrical Engineering and Computer Science,

National University of Sciences and Technology, Islamabad, Pakistan.

[2]Electrical and Computer Engineering Department,

COMSATS University, Islamabad, Pakistan.

Keywords: Model Checking, Formal verification, Deep brain stimulation (DBS), Parkinson's disease (PD), Closed-loop DBS.

Abstract

Deep brain stimulation (DBS) is a widely accepted treatment for the Parkinson's disease (PD). Traditionally, DBS is done in an open-loop manner, where stimulation is always ON, irrespective of the patient needs. As a consequence, patients can feel some side effects due to the continuous high frequency stimulation. Closed-loop DBS can cater

for this problem as it allows adjusting stimulation according to the patient need. The selection of open or closed-loop DBS and an optimal algorithm for closed-loop DBS are some of the main challenges in DBS controller design and typically the decision is made through sampling based simulations. In this paper, 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 analyze the timed automata of the open-loop and closed-loop DBS controllers in response to the basal ganglia (BG) model. Furthermore, we present a formal verification approach for the closed-loop DBS controllers using timed computation tree logic (TCTL) properties, i.e., safety, liveness and deadlock freeness. We show that the closed-loop DBS significantly outperforms existing open-loop DBS controllers in terms of energy efficiency. Moreover, we formally analyze the closed-loop DBS for energy efficiency and time behavior with two different algorithms, i.e., Constant Update algorithm and Error Prediction Update algorithm. Our results demonstrate that the closed-loop DBS running the Error Prediction Update algorithm is efficient in terms of time and energy as compared to the Constant Update algorithm.

1 Introduction

Parkinson's disease (PD) is a nervous system disorder that affects the body movement. PD symptoms include muscle rigidity, tremors, and changes in speech and gait. 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 fam-

ilies and more importantly can imbalance their health system (Nussbaum et al., 2003; Chrischilles et al., 1998). Parkinson is expected to be a major disease worldwide with the passage of time as the population age increases (de Lau et al., 2006; Royal College of Physicians, 2006; Findley, 2007). PD originates in the basal ganglia (BG), which 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, Substantia nigra (SN), Subthalamic nucleus (STN), Globus pallidus externus (GPe) and Globus pallidus internus (GPi). Substantia nigra pars compacta (SNc) play an important role in body movement by releasing a chemical, called dopamine, to carry out messages around the brain. In PD, SNc starts 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 (Jankovic, 2008). 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.

L-dopa is traditionally used as a medicine for PD patients, as it helps to create dopamine molecules within the BG. However, it works for a limited time due to the tolerance developed against this drug by the patients. Deep Brain Stimulation (DBS) is a surgical treatment that can help in relieving PD symptoms, where electric stimulations are delivered to the BG region of the brain (Benabid, 2003; Okun, 2012). In this

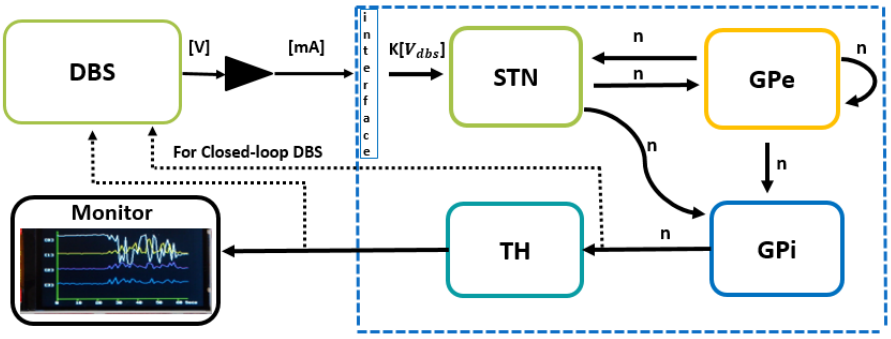


Figure 1: BGM network with components modeling the basal ganglia regions

method, electrodes are placed to generate pulses of high frequency (greater than 100 Hz) in the GPi or STN (Rodriguez-Oroz et al., 2005) while the neural activity of the GPi and TH assist in the PD detection as shown in Fig. 1 (Jovanov et al., 2018).

The most commonly used DBS devices work in an open-loop manner where stimulation/pacing is always ON with fixed parameters (e.g., amplitude (V), pulse repetition frequency (*Hz*), pulse width (*us*)). 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 (Rossi et al., 2016; Deuschl et al., 2006; Cyron, 2016), such as cognitive dysfunction and speech deficits. Closed-loop DBS can cater for these issues (Rossi et al., 2016; Hebb et al., 2014; Parastarfeizabadi et al., 2017). 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 a suitable biomarker reflecting PD symptoms, an appropriate reference signal and implementing a controller to adapt

to dynamic changes in the reference signal(Su et al., 2019; Kuncel et al., 2004).

1.1 Related Work

Krauss et al. (2021) presented a detailed overview of DBS technology evolution from external DBS system to internalized DBS system to predict the future advancement. Multiple DBS electrode configurations, battery designs, and modes of stimulation are discussed by keeping in view the existing technology. The type of DBS system determines the modes of stimulation, i.e., unipolar, bipolar, directional and interleaving. Adaptive or closed-loop DBS system is discussed in which selection of suitable biomarker is very important for a specific disease where sensing technology can play its role for the better efficacy of closed-loop DBS.

A model based design framework to validate different levels of feedback in DBS controllers (open and closed-loop DBS), where BGM is implemented on an FPGA, is proposed in (Jovanov et al., 2018). This BGM platform generates the physiological response that can be used to evaluate and test the the DBS controller design. Similarly, a deep reinforcement learning (RL)-based approach is introduced in (Gao et al., 2020) 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.

Wei et al. (2021) implemented a simplified model for basal ganglia (BG) network on embedded multi-processors that can generate the PD condition by producing abnormal beta bursts, while ensuring the accuracy. They executed a real-time testbed system to verify the usability of DBS mechanism using the implemented BG platform. Differ-

ent open-loop and closed loop DBS controllers were employed in real-time to evaluate the DBS technique optimization. Their implemented BG platform performed well with closed-loop DBS under different DBS strategies.

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 (Brittain et al., 2014). Therefore, a reference signal of constant beta power may not be appropriate 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. (2019) 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. Their proposed CTx-BG-Th network's computational model was used to test the closed-loop adjustment of stimulation frequency approach.

1.2 Our Novel Contribution

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 rigorously verify design (system) and requirements (specifications) for a variety of real-time embedded and safety critical systems (Hasan et al., 2015).

Generally, in model checking, the system that needs to be analyzed is represented as an automaton (state space model) and specifications are written in temporal logic. The verification of the given properties is done by an exhaustive state-space exploration to ensure the validity of the given temporal logic properties. Our DBS behavior can best be modeled as timed automaton due to its time critical nature. So, we used the UPPAAL model checker, which is based on the theory of timed automata (Behrmann et al., 2011). The query language of UPPAAL is a subset of timed computation tree logic (TCTL). There is a dire need of model checking to rigorously explore the complete state space of DBS controllers for this problem, otherwise a missing test case can lead to a wrong prediction of the battery life or timing behavior. In our work, we have used the simplified abstraction of BG network using timed automata to verify the exhaustive behavior of DBS controllers. 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 paper are as follows:

1. Timed abstraction of the BG model from Hybrid Automata to Timed Automata.

2. Timed Automata of open-loop and closed-loop DBS controllers 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 analysis of energy consumption of DBS controllers, i.e., open-loop DBS and closed-loop DBS, and their comparison.

5. A case study to analyze the amount of energy delivered and timing behavior of closed-loop DBS controllers running a specific algorithm.

The rest of the paper is organized as follows: Section II provides an overview of model checking and UPPAAL for the better understanding of the paper. Section III presents the formal models and the proposed verification approach of BG and DBS controller using the UPPAAL model checker. Section IV describes the amount of delivered energy comparison between open-loop and closed-loop DBS and a case study on the analysis of time and energy efficiency with two different algorithms. Finally, Section V concludes the paper.

2 Model Checking and UPPAAL

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 verify 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 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, the most popular are NuSMV (Cimatti et al., 2000), UPPAAL (Behrmann et al., 2011), PRISM (Kwiatkowska, 2003) and SPIN (Holzmann, 2003). A model checker tool is selected based on its applicability to model the real-time systems, asynchronous systems, synchronous digital logic, clock synchro-

nized protocols, etc. Every model checker accepts its own input language with strict notation and features. The NuSMV tool (Cimatti et al., 2000) is intended to formally verify finite state systems with specifications given as Computational Tree Logic (CTL) and Linear Temporal Logic (LTL) formulas. UPPAAL (Behrmann et al., 2011) is intended to formally verify real-time systems where timing aspects are critical. UPPAAL query language is a subset of CTL. PRISM (Kwiatkowska, 2003) is a probabilistic symbolic model checker that is used for automatic formal verification of probabilistic systems. It accepts the system specifications written in the temporal logic Probabilistic Computation Tree Logic (PCTL). SPIN model checker (Holzmann, 2003) is used to verify concurrent and distributed systems. It accepts the system specification written in the verification language Promela.

Table 1 shows the model checker tools and their applicability along with some other useful information. Thus, a model checker tool is selected depending upon the application domain and system characteristics. We have chosen UPPAAL model checker as the most suitable option for modeling and analysing the DBS systems due to its ability to model real-time systems along with its distinct features of having a user friendly GUI and counter-example visualization.

Table 1: Model Checker Tools and their Applicability

	NuSMV	UPPAAL	PRISM	SPIN
Types of Semantic Model	Label Transition System	Timed Transition System	Probabilistic Semantic Model	Concurrent and Distributed Systems
Domain	Digital Circuits	Timed Systems	Health Care	Communication Protocols
GUI	No	Yes	Yes	Yes
Counter Example Generation	Yes	Yes	No	Yes
Counter Example Visualization	No	Yes	No	Yes

UPPAAL is a toolbox for modeling, validation and verification of real-time systems, especially in those application areas where timing is critical, e.g., communication protocols and real-time controllers. UPPAAL 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. 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 τ -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 is 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.

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 [] P (Exists globally P): There exists a path where query P is always satisfies.
  - A [] P (Always globally P): For all paths, query P always satisfies.
- Reachability Property:
  - E <> P (Possibly): There exists a path at which query P possibly satisfies.
- Liveness property:
  - A<> P (Eventually): For all paths, query P eventually satisfies.
  - P $\rightarrow$Q (Leads-to): Whenever P satisfies, query Q verifies eventually.
- Deadlock Property:
  - E <> deadlock: Exists deadlock.
  - A [] not deadlock: There is no deadlock.

3 Proposed Methodology

In this section, we present the proposed methodology for the formal modeling and verification of BG, open-loop DBS and closed-loop DBS controller. BG model is the timed

abstraction of hybrid automata to timed automata. Thereafter, we present the timed automaton of the 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.

3.1 System Overview:

In the proposed methodology, TH behavior and beta power of GPi is monitored to detect the PD condition (Jovanov et al., 2018). Thereafter, if PD condition is detected then stimulations are delivered to STN. The overview of the closed-loop system is shown in Fig. 2, 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,

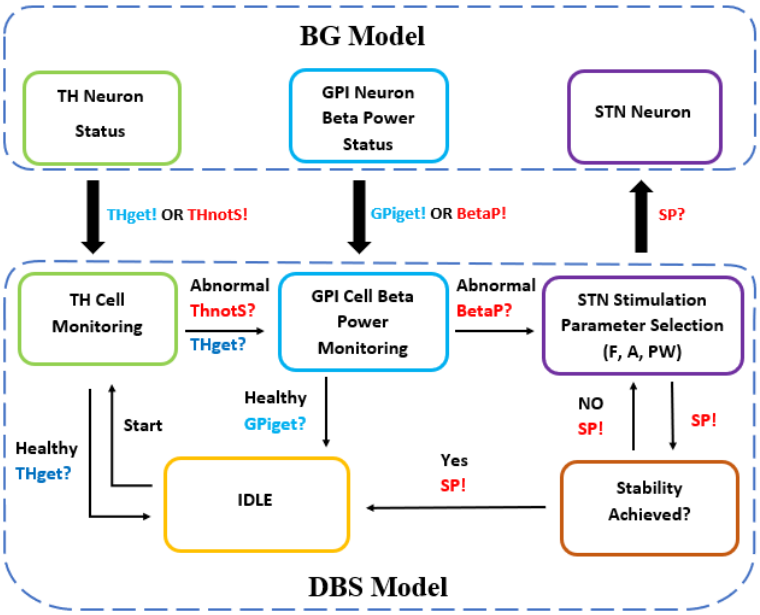


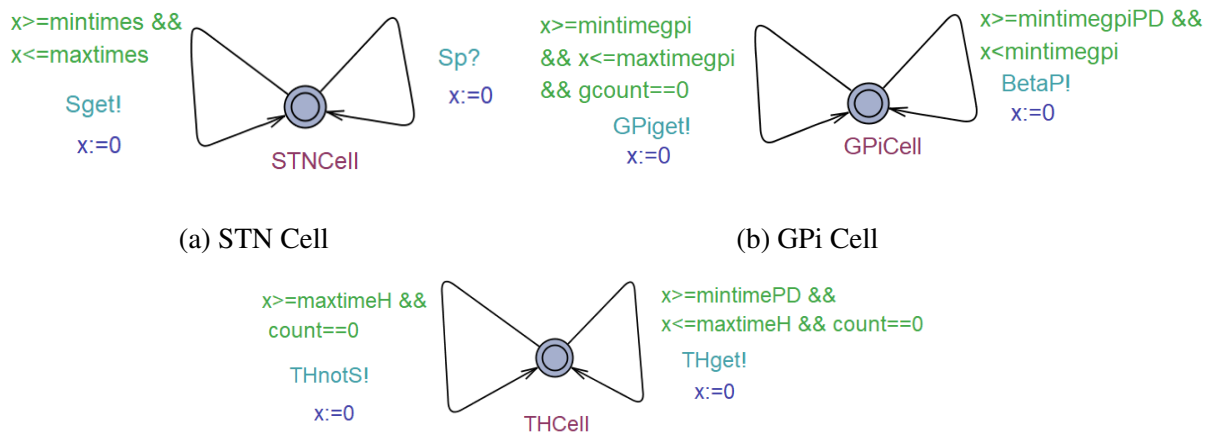
Figure 2: An overview of our proposed methodology representing the closed-loop DBS and BG model communication through some synchronization channels.

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 condition of the BG is indicated by *THget!, THnotS!, GPiget!* and *BetaP!*. Thereafter, DBS processes these signals and generates the pacing action, i.e., *SP!*, to the corresponding region of BG if PD is detected.

3.2 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 (Jovanov et al., 2018; HODGKIN et al., 1952). 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 represented 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 accuracy for the analysis. As mentioned in (Alur et al., 2019), a cell/neuron excitation (voltage change with time), upon stimulation, can be partitioned into some well defined 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 (Alur et al., 2019). We abstracted the



(a) STN Cell

(b) GPi Cell

(c) TH Cell

Figure 3: Timed Automaton for the different regions of BG

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 the literature (Jovanov et al., 2018). Each BG neuron, i.e., *TH, GPi, STN*, is modeled as a separate process with a global clock named *x*, and some activation signals, i.e., *THget!*, *THnotS!*, *GPiget!*, *BetaP!*, *Sget!*, as shown in Fig. 3. Each BG neuron automaton communicates with DBS timed automata using the broadcast channels, i.e., *Sget!, THget!, THnotS!, GPiget! and BetaP!*. The *count* and *gcount* varible detail is discussed in Subsection 3.2 of Section 3.3.

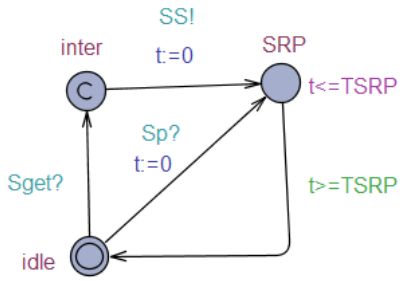
Firing patterns and the time interval details for these BG nuclei are mentioned in Table 2 for both PD and healthy behavior, where the first two columns presents experimental results available in literature (Jovanov et al., 2018). We calculated the time interval of these spikes from the mean firing rates, m_{fr}, as presented in the last two columns of Table 2. Fig. 3a 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

Table 2: Mean Firing Rate Values of Different BG Regions

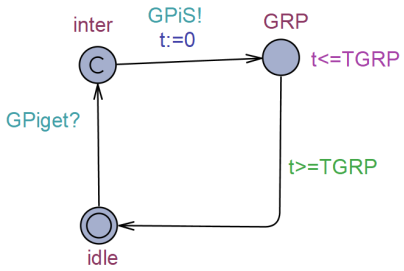
Cell	Healthy m_{fr} [Hz]	PD m_{fr} [Hz]	Healthy Spikes Interval [ms]	PD Spikes Interval [ms]
STN	[9,29]	[11,41]	[34.4,111.1]	[24.3, 90.9]
GPi	[59.8,101.2]	[76.6,135.4]	[9.8,16.7]	[7.3, 13.1]
TH	[10,20]	[5,166.6]	[50,100]	[6,200]

beta power to detect the PD condition, so we 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 intervals, i.e., [50-100ms] for healthy and for PD the interval is [6-50ms] and [>100ms]. 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.

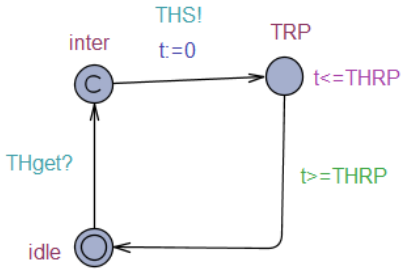
A separate automaton as shown in Fig. 4 is also developed for modeling the refractory period, i.e., a period for an excitable membrane to be responsive for a second stimulus once it returns to its resting state after excitation. Fig. 4 shows the automaton for single cells of *STN*, *GPi* and *TH* regions of BG. We explain automaton for a *STN*



(a) STN Cell



(b) GPi Cell



(c) TH Cell

Figure 4: Timed Automaton Modeling the Refractory Period of different regions of BG

cell and the remaining two (*GPi* and *TH*) are modeled in a similar manner. For the *STN* automaton, whenever the *STN* activation, i.e., *Sget?* is sensed, the automaton goes to the *SRP* (STN Refractory Period) state followed by the *inter* state. While making the transition from *inter* state to *SRP* state, it transmits or passes an activation signal *SS!* that is detected by the controller automaton by using *SS?*, where *SS* indicates that the *STN* activation is sensed by the controller. It remains in the *SRP* state for the *TSRP* duration, where *TSRP* indicates the *STN* refractory period. 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 activation behaviors of GPi and TH cells are captured in the same way in Fig. 4b and Fig. 4c, respectively, with the similar notation and respective names for cell signals.

We have integrated all the components of BG model, as mentioned in Fig. 3, and

Fig. 4 by keeping in view the required level of accuracy for the analysis as mentioned in (Alur et al., 2019). We believe that this abstraction from hybrid automaton to timed automata would not compromise the real world scenarios as timing aspects (stimulation/response time interval) for PD and healthy brain has been taken from the available literature (Jovanov et al., 2018). Our abstracted BG model generates the human brain sensory events just like the real possible scenarios that can be used to verify the functionality of the closed-loop DBS controller.

3.3 Modeling the DBS (Timed Automata):

3.1 Open-Loop DBS Modeling

The Open-loop DBS does not sense the BG condition and continuously deliver 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 parameters (e.g., amplitude(A) =1V, frequency(f) =115Hz, impedance=1000Ω and pulse width(PW) =200us) are delivered to STN. An open-loop DBS automaton is shown in Fig. 5, with a delay activation signal, i.e., *Sp!*. The electrical energy delivered per unit time (E) is discussed in detail in Subsection 3.3 of Section 3.4.

3.2 Closed-Loop DBS Modeling

We considered TH neuron pattern to detect any abnormality in BG, as reported in (Jovanov et al., 2018), by considering that TH is not a part of BG but assist in PD detection. 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

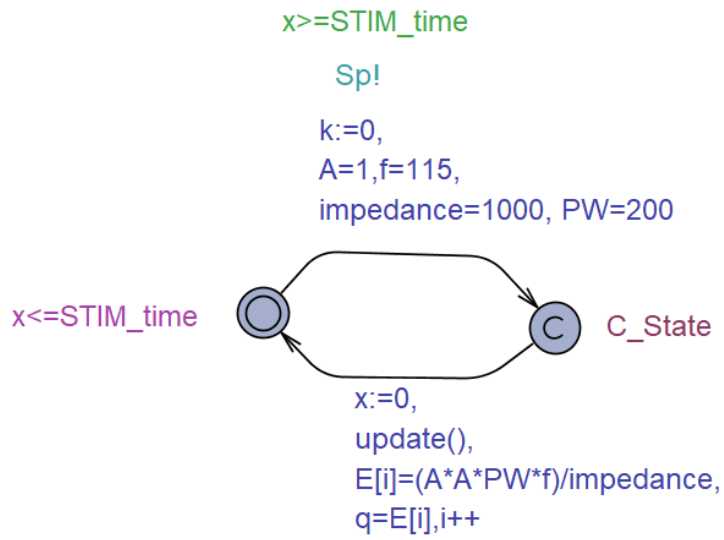


Figure 5: Timed Automaton of Open-loop DBS Controller

condition (Su et al., 2019), since it is 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. 6. 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. 2, i.e., to transition back to the *IDLE* state if normal TH behavior is observed

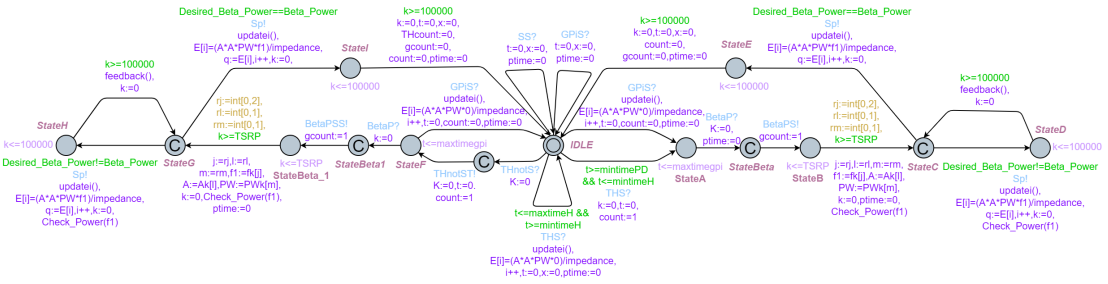


Figure 6: Timed Automaton of Closed-loop DBS Controller.

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 stimulations 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., f (Hz), A (V) and PW (us).

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. (2018) 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 (Su et al., 2019) are quite useful in the context of suppressing PD symptoms. Su et al. (2019) implemented a PI controller, that delivers the stimulation of randomly selected frequency from the

given range 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 frequency selected from the given range 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 (Su et al., 2019).

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 (Su et al., 2019) 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. 6, 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.

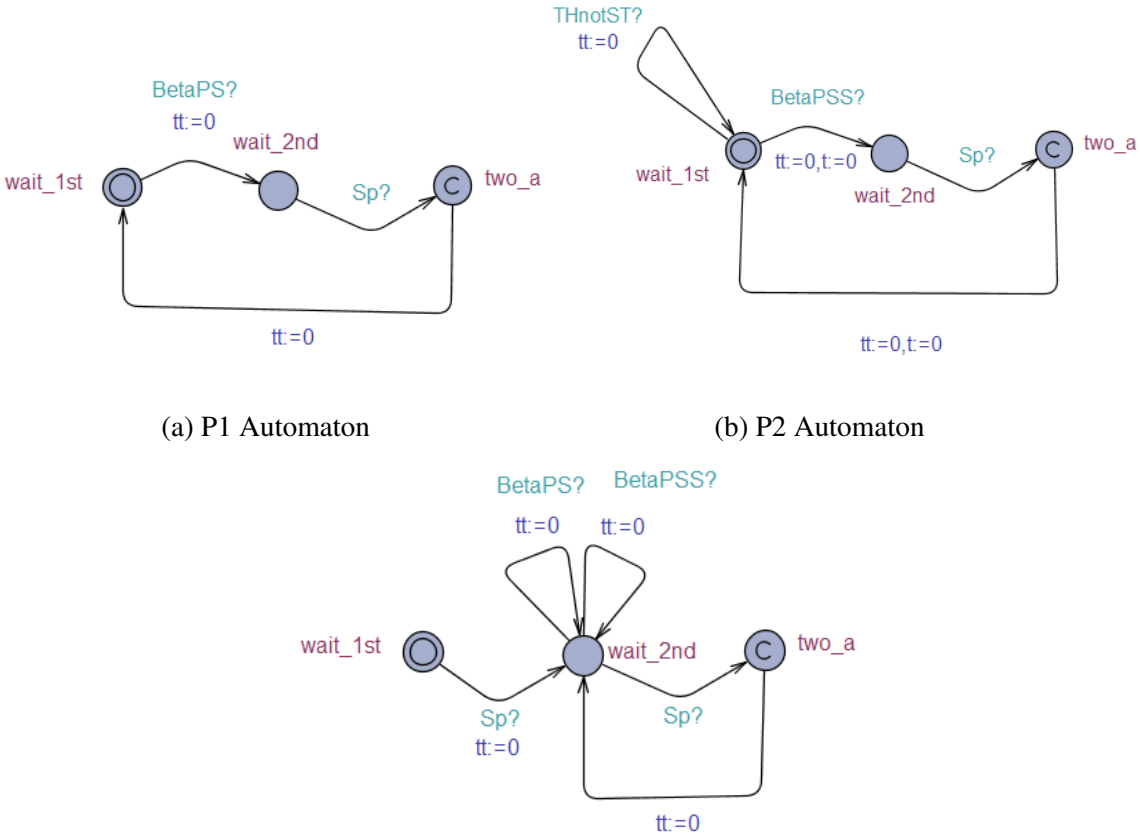
The *count* and *gcount* are variables used as flags for *TH* and *GPi*, respectively. These flags are set to 1 to prevent the interruption caused by TH and GPi cell activation when the controller is pacing to STN in the pacing phase. These flags are reset to 0 when the controller is reset or it goes back to the sensing phase because *TH* and *GPi* cell activation detection is required in the sensing phase. The natural pacing activity of *TH* and *GPi* is, however, not affected by using these flag variables.

3.4 Verifying the closed-loop 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:

3.1 Safety

The Safety property asserts that something bad will never happen. We aim to deliver stimulations to the STN whenever PD or any abnormality in the BG is detected. We made a separate automaton named *P1* as shown in Fig. 7a with a local clock named *tt*. *BetaPS* and *Sp* are broadcast channels, where *BetaPS* indicates that beta band power of *GPi* lies in the PD range and PD condition is sensed by the controller. We want to detect these signals, i.e., *BetaPS* and *Sp*, whenever they are transmitted by the DBS timed automaton. The clock is reset on the reception of *BetaPS* signal, i.e., when PD is



(a) P1 Automaton

(b) P2 Automaton

(c) P3 Automaton

Figure 7: Timed Automaton for Properties Verification

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.

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 property for this requirement is mentioned below

- A[] P1.two_a imply (P1.tt>=TSRP && P1.tt<=TSLRI)

By using the above mentioned property, 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 we verified A[] not (P1.two_a && P1.tt>TSLRI) and A[] not (P1.two_a && P1.tt<TSRP) properties just to ensure that the bad states are never reached.

Another automaton named *P2*, as shown in Fig. 7b, 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. Here, *THnotST* indicates the abnormal behavior of *TH* cell (no spikes at all) that can happen when the *TH* activation is not sensed by the controller. The controller automaton relays an activation signal *THnotST!* whenever *TH* generates no spikes. This activation signal is then detected by the *P2* automaton through *THnotST?*. The following safety requirement for this automaton was verified successfully.

- A[] P2.two_a imply (P2.tt>=TSRP && P2.tt<=TSLRI)

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. 7c. 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 property for these safety requirements.

- A[] P3.two_a imply (P3.tt>=TSRP && P3.tt<=TSLRI)

3.2 Liveness

The liveness property asserts that, under certain conditions, some event will eventually 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<>(T1.StateBeta) imply (T1.StateE or T1.StateD)
- A<>(T1.StateBeta_1)imply (T1.StateH or T1.StateI)

As can be observed from Fig. 6, i.e., the *T1* automaton, where PD is detected in *StateBeta* or *StateBeta_1* and consequently STN pacing is applied in *StateE, SatetD, 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.

3.3 Supremum and Infimum Queries for Time and Amount of delivered Energy Analysis

In this case study, we aim to analyze the amount of delivered energy and timing constraints that provide an important design guideline in selecting the DBS parameters. By doing so, we can estimate the total electrical energy delivered per unit time from DBS

to STN. Thereby, the total energy delivered per unit time (E) can easily be estimated by using the below mentioned formula (Cagnan et al., 2017).

$$E = (A * A * PW * f)/impedance$$

There are certain parameters that need to be considered for the estimation of delivered energy, i.e., A (V), PW (us), f (Hz) and impedance (Ω). All parameters except impedance are programmable (Cagnan et al., 2017). Thereby, we considered a range of these parameters based on an existing work (Cagnan et al., 2017) as can be observed from Fig. 6, where the DBS model randomly selects any value of these parameters from the given range of A (V), PW (us) and f (Hz). We consider a constant value of tissue impedance faced by electrodes while delivering stimulations (Cagnan et al., 2017).

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 energy bounds of delivered stimulations, where q represents delivered electrical energy per unit time.

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

By using the above-mentioned queries, amount of delivered energy 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 energy value, i.e., sup generates the max energy value and inf generates the min energy value. The get trace option can also be used to identify the parameters responsible for these energy values generation. Some results are shown in Table 3, for both inf and sup queries for different parameters range with constant tissue impedance.

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

Table 3: Amount of delivered Energy and Time analysis result for different range of stimulation parameters

Properties	f $\in \{60,130,180\}Hz$ A $\in \{1,3\}V$ PW $\in \{60,200\}us$	f $\in \{80,105,155\}Hz$ A $\in \{1,3\}V$ PW $\in \{60,200\}us$	f $\in \{90,145,195\}Hz$ A $\in \{1,2\}V$ PW $\in \{50,100\}us$	f $\in \{105,170,190\}Hz$ A $\in \{1,2\}V$ PW $\in \{50,100\}us$
inf{T1.StateE or T1.StateD} : q	3uj	4uj	4uj	5uj
sup{T1.StateE or T1.StateD} : q	324uj	279uj	78uj	76uj
inf{T1.StateE } : T1.ptime	200000us	200000us	300000us	400000us
sup{T1.StateE } : T1.ptime	600000us	500000us	900000us	700000us

- inf{T1.StateE} : T1.ptime
- sup{T1.StateE} : 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 Table 3 with the same parameters ranges considered for delivered energy analysis.

Amount of delivered energy and time analysis results are used to select DBS design and parameters. A detailed depiction of how the energy and time varies with these parameters is shown in Table 3 with different stimulation parameters ranges, i.e., f (*Hz*), A (*V*) and PW (*us*). Table 3 shows the result of inf and sup queries (listed on the extreme left side of the table) for both energy and time analysis with stimulation parameters ranges mentioned on the top of the table. Table 3 clearly depicts the effect of these parameters on delivered energy values, the higher the stimulation parameter values the higher the delivered energy value is. Likewise, the time analysis strongly depends on the selected parameters, the stimulation time keeps on increasing until parameters are fine tuned to achieve the stability.

3.4 Reachability and Deadlock

Reachability properties are often used to ensure that some particular situation can be reached. For example, when designing 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<> T1.StateE
- E<> T1.StateD
- E<> T1.StateH
- E<> GPiComponent.GPiCell

Deadlock is a state in which progress is impossible. A state is termed as a deadlock state if there are no outgoing action transitions neither from the state itself or any of its delay successors.The ideal system/model should be deadlock free. The deadlock property is used to verify that the system is deadlock-free as follows:

- A[] not deadlock
- E<> deadlock

We verified the above mentioned requirements for the DBS controller to ensure that the system will never get stuck as it is a very important requirement for the real-time systems, i.e., DBS in our case.

4 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 the closed-loop DBS design parameters.

4.1 Energy Efficiency comparison of open-loop and closed-loop DBS:

Electrical energy delivered per unit time (E) is calculated whenever stimulation is delivered to STN, i.e., *Sp!*, by using the following formula (Cagnan et al., 2017).

$$E[i] = (A * A * PW * f)/impedance$$

Where *i* indicates the index number of the *E* array to calculate the electrical energy delivered per unit time. We simulated the timed automaton of the open-loop DBS controller with stimulation parameters based on the existing work (Cagnan et al., 2017), i.e., A =1V, f =115Hz, impedance=1000Ω and PW =200us, for 292 iterations that is approximately equal to 2.5s, as shown in Fig. 8. Due to fixed parameters, the delivered energy value remains the same for the whole time and average amount of delivered en-

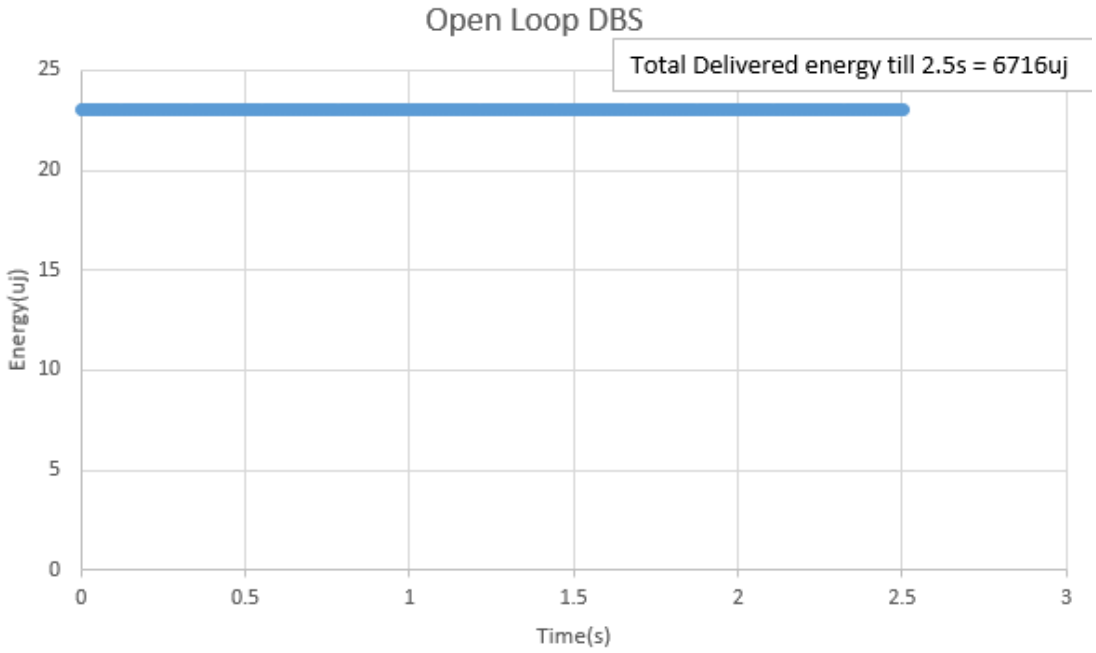


Figure 8: Graph representing the delivered electrical energy behavior of Open-loop DBS Controller. Total delivered energy comes out to be 6716uj approximately and average amount of delivered energy is approximately 23uj for 2.5s.

ergy comes out to approximately 23uj.

In order to do a fair comparison between both controllers, we apply the same methodology to the closed-loop DBS model with the same parameters, i.e., A=1V, impedance=1000Ω, PW=200us, 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. 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 on, as can be observed throughout the graph in Fig. 9. The average amount of delivered energy is approximately equal to 14.8uj.

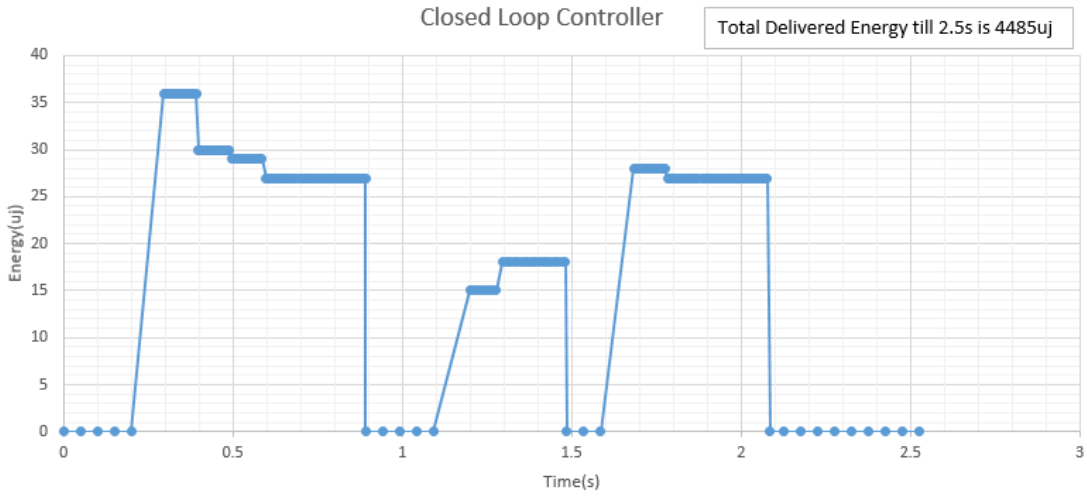


Figure 9: Graph representing the delivered electrical energy behavior of Closed-loop DBS Controller. Total delivered energy comes out to be 4485uj approximately and average amount of delivered energy is approximately 14.8uj for 2.5s.

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 energy 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.

4.2 Effect of closed-loop DBS 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 stimulation 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 which are named as Constant Update algorithm and Error Prediction Update algorithm, 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 the Constant Update algorithm, the 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

Algorithm 1: In the **Constant Update algorithm**, the DBS selects a random frequency for stimulation from the allowed permissible 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.

$intfeedback()$

$\{$

if $Beta\_Power\ != Desired\_Beta\_Power$ **then**

  if $f1 == 200$ **then**

    $f1 \leftarrow 0$

  end if

  $f1 \leftarrow f1 + 5$

else

  $err \leftarrow Desired\_Beta\_Power - Beta\_Power$

  $f1 \leftarrow f1 - err$

end if

$\}$

checks all the available frequency range with a step size of 5. For example, if the DBS initially chooses randomly f=100Hz 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 Constant Update algorithm, and the new updated frequency will become f=105Hz. 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 becomes equal to the desired beta band power.

Algorithm 2: In the **Error Prediction Update algorithm**, the DBS selects a frequency randomly from the available permissible range and the error is calculated based on the beta band power. The stimulation frequency is then updated depending on the value of error until stability is achieved.

$int feedback()$

$\{$

if $Beta\_Power > Desired\_Beta\_Power$ **then**

$err \leftarrow Beta\_Power - Desired\_Beta\_Power$

$f1 \leftarrow f1 + err$

end if

if $Desired\_Beta\_Power > Beta\_Power$ **then**

$err \leftarrow Desired\_Beta\_Power - Beta\_Power$

$f1 \leftarrow f1 - err$

end if

if $Desired\_Beta\_Power == Beta\_Power$ **then**

$err \leftarrow Desired\_Beta\_Power - Beta\_Power$

$f1 \leftarrow f1 - err$

end if

$\}$

In the Error Prediction Update algorithm, the DBS controller initially selects a frequency randomly from the available permissible range 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 algorithm. 2, and the DBS keeps on updating the stimulation frequency until stability

is achieved. A comparison of both cases is given in Table 4, with the same parameters, i.e., A (V), PW (us), f (Hz) and impedance (Ω), 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

Properties	f $\in \{90,140,190\}Hz$ A = $1V$ PW = $100us$	f $\in \{100,150,195\}Hz$ A = $1V$ PW = $100us$	f $\in \{80,130,185\}Hz$ A = $1V$ PW = $100us$	f $\in \{70,100,130\}Hz$ A = $1V$ PW = $100us$
infT1.StateE : T1.ptime	$200000us$	$300000us$	$200000us$	$200000us$
supT1.StateE : T1.ptime	$3300000us$	$3100000us$	$2400000us$	$700000us$

(a) Constant Update algorithm

Properties	f $\in \{90,140,190\}Hz$ A = $1V$ PW = $100us$	f $\in \{100,150,195\}Hz$ A = $1V$ PW = $100us$	f $\in \{80,130,185\}Hz$ A = $1V$ PW = $100us$	f $\in \{70,100,130\}Hz$ A = $1V$ PW = $100us$
infT1.StateE : T1.ptime	$200000us$	$300000us$	$200000us$	$200000us$
supT1.StateE : T1.ptime	$700000us$	$900000us$	$400000us$	$800000us$

(b) Error Prediction Update algorithm

Table 4: Time Analysis of Constant Update algorithm and Error Prediction Update algorithm. "Table. 4a and 4b" show the result of inf and sup queries with different stimulation parameter ranges taken from (Cagnan et al., 2017) for Constant Update algorithm and Error Prediction Update algorithm, respectively. From the overall comparison between these two algorithms, it can easily be observed that Error Prediction Update algorithm takes less time to achieve stability as compared to Constant Update algorithm but stability is not always guaranteed in Error Prediction Update algorithm as compared to Constant Update algorithm.

achieve stability.

It can be clearly observed that Constant Update algorithm takes more time to achieve stability as compared to Error Prediction Update algorithm, because Constant Update algorithm checks all frequency ranges until the stability is achieved while on the other side Error Prediction Update algorithm just updates the frequency based on the error and achieves stability in less time due to less number of iterations. In Constant Update algorithm, stability is guaranteed because it checks all frequency combinations, but in Error Prediction Update algorithm, 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, Constant Update algorithm takes more time to achieve stability but stability is always guaranteed, Error Prediction Update algorithm 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 analysis for the amount of delivered energy for Constant Update algorithm and Error Prediction Update algorithm. Delivered electrical energy depends upon certain factors, i.e., A (V), PW (us), f (Hz) and impedance (Ω). The results in Table 3 clearly demonstrate how energy value varies by changing any of these parameters. Table 3 simply depicts the effect of these parameters on energy values. As mentioned previously, Constant Update algorithm takes more time to achieve stability as compared to Error Prediction Update algorithm, which means stimulations are deliv-

ered for a longer period of time in order to achieve stability, that in turn leads to more power consumption for a specific time duration as compared to Error Prediction Update algorithm. Hence, we can say that Error Prediction Update algorithm is best if we consider the power aspect only, but at the same time stability is not always guaranteed in Error Prediction Update algorithm, so there are pros and cons for both options and our approach allows us to choose the best option according to the given requirements.

We analyzed the closed-loop DBS controller response rigorously with two different error correction schemes, i.e, Constant Update algorithm and Error Prediction Update algorithm, on a single platform with the same stimulation parameter settings, i.e., A (V), PW (us), f (Hz) and impedance (Ω). To the best of our knowledge, we have never come across any work like this, where two different error correction schemes/algorithms can be compared on a single platform with the same stimulation parameter settings. Therefore, these findings clearly indicate the usefulness of this platform that can then be used to compare and choose different design algorithms of DBS according to the need and given requirements.

5 Conclusion

In this paper, we presented a verification methodology to formally analyze the DBS controllers for PD and to investigate the effects of design parameters on energy consumption and timing analysis of DBS controllers. 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 freeness. 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 energy consumption or stimulation time. The analysis results are found to be quite useful in the context of DBS programming algorithms and design parameters selection.

References

Nussbaum, R. L., & Ellis, C. E.(2003). Alzheimer's disease and Parkinson's disease. *The New England journal of medicine*, *348(14)*, 1356–1364. https://doi.org/10.1056/NEJM2003ra020003

Chrischilles, E. A., Rubenstein, L. M., Voelker, M. D., Wallace, R. B., & Rodnitzky, R. L. (1998). The health burdens of Parkinson's disease. *Movement disorders : official journal of the Movement Disorder Society*, *13(3)*, 406-13. doi:10.1002/mds.870130306

de Lau, L. M., & Breteler, M. M. (2006). Epidemiology of Parkinson's disease. *The Lancet. Neurology*, *5(6)*, 525–535. https://doi.org/10.1016/S1474-4422(06)70471-9

National Collaborating Centre for Chronic Conditions (UK). *(2006). Parkinson's Disease: National Clinical Guideline for Diagnosis and Management in Primary and Secondary Care.* Royal College of Physicians (UK).

Findley, L. J. (2007). The economic impact of Parkinson's disease. *Parkinsonism & related disorders*, *13* Suppl, S8–S12. https://doi.org/10.1016/j.parkreldis.2007.06.003

Jankovic, J. (2008). Parkinson's disease: clinical features and diagnosis. *Journal of neurology, neurosurgery, and psychiatry*, *79(4)*, 368–376. https://doi.org/10.1136/jnnp.2007.131045

Rodriguez-Oroz, M. C., Obeso, J. A., Lang, A. E., Houeto, J. L., Pollak, P., Rehncrona, S., Kulisevsky, J., Albanese, A., Volkmann, J., Hariz, M. I., Quinn, N. P., Speelman, J. D., Guridi, J., Zamarbide, I., Gironell, A., Molet, J., Pascual-Sedano, B., Pidoux, B., Bonnet, A. M., Agid, Y., ... Van Blercom, N. (2005). Bilateral deep brain stimulation in Parkinson's disease: a multicentre study with 4 years follow-up. *Brain : a journal of neurology*, *128(Pt 10)*, 2240–2249. https://doi.org/10.1093/brain/awh571

Rossi, P. J., Gunduz, A., Judy, J., Wilson, L., Machado, A., Giordano, J. J., Elias, W. J., Rossi, M. A., Butson, C. L., Fox, M. D., McIntyre, C. C., Pouratian, N., Swann, N. C., de Hemptinne, C., Gross, R. E., Chizeck, H. J., Tagliati, M., Lozano, A. M., Goodman, W., Langevin, J. P., ... Okun, M. S. (2016). Proceedings of the Third Annual Deep Brain Stimulation Think Tank: A Review of Emerging Issues and Technologies. *Frontiers in neuroscience*, *10*, 119. https://doi.org/10.3389/fnins.2016.00119

Hebb, A. O., Zhang, J. J., Mahoor, M. H., Tsiokos, C., Matlack, C., Chizeck, H. J., & Pouratian, N. (2014). Creating the feedback loop: closed-loop

neurostimulation. *Neurosurgery clinics of North America*, *25(1)*, 187–204. https://doi.org/10.1016/j.nec.2013.08.006

Brittain, J. S., & Brown, P. (2014). Oscillations and the basal ganglia: motor control and beyond. *NeuroImage*,*85 Pt 2(Pt 2)*, 637–647. https://doi.org/10.1016/j.neuroimage.2013.05.084

Benabid, A. L. (2003). Deep brain stimulation for Parkinson's disease. *Current opinion in neurobiology*, *13(6)*, 696–706. https://doi.org/10.1016/j.conb.2003.11.001

Okun, M. S. (2012). Deep-brain stimulation for Parkinson's disease. *The New England journal of medicine*, *367(16)*, 1529–1538. https://doi.org/10.1056/NEJMct1208070

Gao, Q., Naumann, M., Jovanov, I., Lesi, V., Kamaravelu, K., Grill, W. M.,& Pajic, M. (2020). 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 pp. 108-118*, doi: 10.1109/ICCPS48487.2020.00018.

Jovanov, I., Naumann,M., Kumaravelu, K., Grill, W. M., & M. Pajic. (2018). Platform for Model-Based Design and Testing for Deep Brain Stimulation. *2018 ACM/IEEE 9th International Conference on Cyber-Physical Systems (ICCPS), Porto*, *pp. 263-274*, doi: 10.1109/ICCPS.2018.00033.

Su, F., Kumaravelu, K., Wang, J., & Grill, W. M. (2019). Model-Based Evaluation of Closed-Loop Deep Brain Stimulation Controller to Adapt to Dy-

namic Changes in Reference Signal. *Frontiers in neuroscience*, *13*, 956. https://doi.org/10.3389/fnins.2019.00956

HODGKIN, A. L., & HUXLEY, A. F. (1952). A quantitative description of membrane current and its application to conduction and excitation in nerve. *The Journal of physiology*, *117(4)*, 500–544. https://doi.org/10.1113/jphysiol.1952.sp004764

Behrmann, G., David, A., Larsen, K., Pettersson, P., & Yi. W. (2011). Developing uppaal over 15 years. *Software—Practice & Experience*. *41(2)*, 133–142.

Alur, R., Giacobbe, M., Henzinger, T., Larsen, K.G., & Mikučionis, M. (2019). Continuous-Time Models for System Design and Analysis. *In: Lecture Notes in Computer Science*,*Vol. 10000*, p. 452–477, doi: https://doi.org/10.1007/978-3-319-91908-9_22

Cagnan, H., Pedrosa, D., Little, S., Pogosyan, A., Cheeran, B., Aziz, T., Green, A., Fitzgerald, J., Foltynie, T., Limousin, P., Zrinzo, L., Hariz, M., Friston, K. J., Denison, T., & Brown, P. (2017). Stimulating at the right time: phase-specific deep brain stimulation. *Brain : a journal of neurology*, *140(1)*, 132–145. https://doi.org/10.1093/brain/aww286

Hasan, O., & Tahar, S. (2015). Formal Verification Methods, In: Mehdi Khosrow-Pour (Eds.), *Encyclopedia of Information Science and Technology, IGI Global Pub*, *pp 7162-7170*.

Krauss, J. K., Lipsman, N., Aziz, T., Boutet, A., Brown, P., Chang, J. W., Davidson, B., Grill, W. M., Hariz, M. I., Horn, A., Schulder, M., Mammis, A., Tass,

P. A., Volkmann, J., & Lozano, A. M. (2021). Technology of deep brain stimulation: current status and future directions. *Nature reviews. Neurology*, *17(2)*, 75–87. https://doi.org/10.1038/s41582-020-00426-z

Wei, X., Zhang, H., Gong, B., Chang, S., Lu, M., Yi, G., Zhang, Z., Deng, B., & Wang, J. (2021). An Embedded Multi-Core Real-Time Simulation Platform of Basal Ganglia for Deep Brain Stimulation. *IEEE transactions on neural systems and rehabilitation engineering : a publication of the IEEE Engineering in Medicine and Biology Society*,*29*, 1328–1340. https://doi.org/10.1109/TNSRE.2021.3095316

Parastarfeizabadi, M., & Kouzani, A. Z. (2017). Advances in closed-loop deep brain stimulation devices. *Journal of neuroengineering and rehabilitation*, *14(1)*, 79. https://doi.org/10.1186/s12984-017-0295-1

Kuncel, A.M., & Grill, W. M. (2004). Selection of stimulus parameters for deep brain stimulation. *Clinical neurophysiology : official journal of the International Federation of Clinical Neurophysiology 115(11)* (2004): 2431-41. doi:10.1016/j.clinph.2004.05.031

Deuschl, G., Herzog, J., Kleiner-Fisman, G., Kubu, C., Lozano, A. M., Lyons, K. E., Rodriguez-Oroz, M. C., Tamma, F., Tröster, A. I., Vitek, J. L., Volkmann, J., & Voon, V. (2006). Deep brain stimulation: postoperative issues. *Movement disorders : official journal of the Movement Disorder Society*, *21 Suppl 14*, S219–S237. https://doi.org/10.1002/mds.20957

Cyron, D. (2016). Mental Side Effects of Deep Brain Stimulation (DBS) for Movement

Disorders: The Futility of Denial. *Frontiers in integrative neuroscience*, *10*, 17. https://doi.org/10.3389/fnint.2016.00017

Cimatti, A., Clarke, E., Giunchiglia, F. Roveri, M. (2000) NUSMV: a new symbolic model checker. *STTT 2, 410–425 (2000)*. https://doi.org/10.1007/s100090050046

Kwiatkowska, M. (2003). Model checking for probability and time: from theory to practice. 18th Annual IEEE Symposium of Logic in Computer Science, 2003. *Proceedings., 351-360.*

Holzmann, G.J., 2003. The SPIN Model Checker: Primer and Reference Manual. 1st Edn., AddisonWesley, Germany, *ISBN-10: 0321228626, pp: 608*