Model-based Closed-loop Testing of Implantable Pacemakers

Zhihao Jiang, Miroslav Pajic and Rahul Mangharam
Department of Electrical and Systems Engineering
University of Pennsylvania
{zhihaoj, pajic, rahulm}@seas.upenn.edu

***Abstract*—The increasing complexity of software in implantable medical devices such as cardiac pacemakers and defibrillators accounts for over 40% of device recalls. Testing remains the principal means of verification in the medical device certification regime. Traditional software test generation techniques, where the tests are generated independently of the operational environment, are not effective as the device must be tested within the context of the patient's condition and the current state of the heart. It is necessary for the testing system to observe the system state and conditionally generate the next input to advance the purpose of the test. To this effect, a set of general and patient condition-specific temporal requirements is specified for the closed-loop heart and pacemaker system. Based on these requirements, we describe a closed-loop testing environment between a timed automata-based heart model and pacemaker. This allows for interactive and physiologically relevant model-based test generation for basic pacemaker device operations such as maintaining the heart rate and atrial-ventricle synchrony. We also demonstrate the flexibility and efficacy of the testing environment for more complex common timing anomalies such as reentry circuits, pacemaker mode switch operation and pacemaker-mediated tachycardia. This system is a step toward a testing approach for medical cyber-physical systems with the patient-in-the-loop.**

I. Introduction

Safety recalls of pacemakers and implantable cardioverter defibrillators due to firmware problems between 1990 and 2000 affected over 200,000 devices. This encompasses 41% of the devices recalled [1]. An estimated 1.3 million device checks and analyses and 36,187 device replacements resulted from the advisories at a cost of approximately $870 million. In 1996, 10% of medical devices recalls were caused by software-related issues. In June of 2006, software errors in medical devices made up 21% of recalls. During the first half of 2010, the Food and Drug Administration (FDA) issued 23 recalls of defective devices, all of which are categorized as *Class I*, meaning there is "reasonable probability that use of these products will cause serious adverse health consequences or death." At least six of the recalls were likely caused by software defects [2], [3].

The FDA currently does not request or review the medical device software during pre-market submission. While no specific requirements or software verification standards are issued, a set of general guidelines for software evaluation are recommended [4], [5], [6]. The responsibility to demonstrate the safety and efficacy of the device software is solely on the manufacturer. This is currently satisfied by the documentation of code inspections, static analysis, module-level testing and integration testing and their purpose is to establish "reasonable assurance of safety and effectiveness". These tests however fail to check for the correctness of the software and are largely open-loop tests that do not consider the context of the patient. Software is reviewed by the FDA only in the incident of a device recall. Software-related recalls are often issued in the form of *Safety Alerts* by the Food and Drug Administration (FDA) such as "Safety alert - Pacemaker may revert to VVI mode at 70 beats/min if programmed to one of several specific ventricular pulse widths"[2].

An effective software verification methodology is therefore needed for the risk analysis and certification of medical device software during the pre-market submission phase. Testing for medical device software currently is ad hoc, error prone, and very expensive. Traditional methods of testing do not suffice as the test generation cannot be done independently of the current state of the patient and organ. As the testing environment (i.e. patient condition) is not entirely under the control of the tester, the problem changes significantly as a degree of nondeterminism is introduced in the process. Implantable medical devices are a primary example of Medical Cyber-Physical Systems where the safety and efficacy of the device and device software must be evaluated within a closed-loop context of the patient. The key challenge is in the generation of physiologically relevant tests such that the device does not provide inappropriate therapy and does not adversely affect the safety of the patient. In addition, test generation must be interactive and adaptive such that the previous test stimulus affects the current state of the patient. The test generator must consider the current state when generating the next input in a way that advances the purpose of the test. The problem, therefore, becomes one of controller synthesis and cannot be addressed by an off-the-shelf model checker [7].

The focus of this effort is three-fold: (a) We provide a set of general and patient condition-specific pacemaker software requirements to ensure the safety of the patient, (b) We developed a timed automata-based heart model and pacemaker device model for interactive and clinically relevant test generation, and (c) We test the closed-loop system over a variety of basic operation tests where the heart rate must be maintained and the atrial-ventricle synchrony must be maintained. Furthermore, we present a set of complex, but common, cases such as supraventricular tachycardia due to timing derangements caused by reentry circuits, pacemaker mode-switch operation and pacemaker mediated tachycardia condition. With this approach of model-based testing, an executable functional model of the pacemaker is created at an early stage in the development process. The focus of our safety analysis is primarily on the timing behavior of the closed-loop system as heart is the most important natural real-time system. The timing of the intrinsic or induced pacing directly affects the hemodynamics of the heart and is a primary function of the stroke volume.

A. *Model-based Medical Device Testing*

Software embedded in a pacemaker or implantable defibrillator may have more than 80,000 lines of code [8]. There is currently no software testing standard for implantable medical devices. Testing is the observation of a program in execution under controlled conditions. The outcome of a test is compared with an oracle to determine the appropriateness or correctness of the therapy. While formal methods of verification is needed for medical device software [9], [10], [11], testing continues to be required because it can expose different kinds of problems (e.g., compiler bugs), can examine the program in its system context, and increases the diversity of evidence available. We do not have access to the source code and black-box testing limits the scope of evaluation. It is therefore necessary to develop a framework for model-based testing wherein the device itself, or a model of the device, is tested in closed-loop with the model of the patient or the organ of concern. The key challenge with model-based testing is having a good test generator. In the case for implantable medical devices, the challenges for software testing and test generation are manyfold:

1. Testing must be Physiologically-Relevant
Testing cannot be expected to catch every error in the program and it is impossible to evaluate every execution path. Many software errors, like unintended restarts or incorrect measurements, do not surface until after the devices are approved and in use. It is therefore necessary to provide the appropriate kind and level of abstraction of the patient and device model for testing. In our previous work [12], we developed a Virtual Heart Model (VHM) and dual chamber DDD pacemaker model in Simulink for medical device validation. The VHM is an electrophysiological model of the heart and models the timing and electrical conduction of the heart with both intrinsic and artificial pacing signals. In this paper, we use the VHM for closed-loop testing of pacemaker functionality for baseline and common complex test cases.

2. Non-deterministic factors in Testing
Traditional software testing is an open loop exercise to test for generic bugs and incorrect execution given a set of structural coverage criteria. This lends itself to Automated Test Generation by methods such as theorem proving, constraint logic programming and symbolic execution, model checking, using an event-flow model and using a Markov chains model [7]. The test-generation problem becomes significantly more complex and difficult when the program under test is non-deterministic or when the functionality of the program is a function of not only the computing and communication components but the operational environment. In the case of an implantable medical device such as an artificial pacemaker, the closed-loop system of the heart and the device must be tested as a whole, within the context of the patient's condition (see Fig. 1).

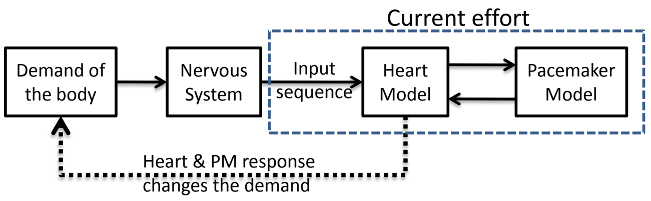


Figure 1. Closed-loop model of the heart and pacemaker in the context of the patient condition. Our focus is on the inner loop of the heart model and the pacemaker model.

3. Interactive and Adaptive Testing
The primary approach to system-level testing of medical devices is unit testing using a playback of pre-recorded electrogram and electrocardiogram signals [13], [14]. This tests if the input signal triggers a particular response by the pacemaker but has no means to evaluate if the response was appropriate for the patient condition. Furthermore, this approach of "tape testing" is unable to check for safety violations due to inappropriate stimulus by the pacemaker. Pacemaker Mediated Tachycardia (PMT), a condition that is described later in this paper, is a strong example of why we need an interactive and adaptive test generation systems such as the VHM. PMT is a condition where the pacemaker inappropriately drives the intrinsic heart-rate toward the upper rate limit. With a tape test, PMT would not occur and the response of the pacemaker could be classified as appropriate therapy.

The focus of this work is on the development of a system and methods for integration and system-level testing for implantable cardiac pacemakers. To address this, we specify a set of general and condition-specific requirements for the closed-loop system. We evaluate this by constructing a set of monitors to test for timing and safety conditions for each case.

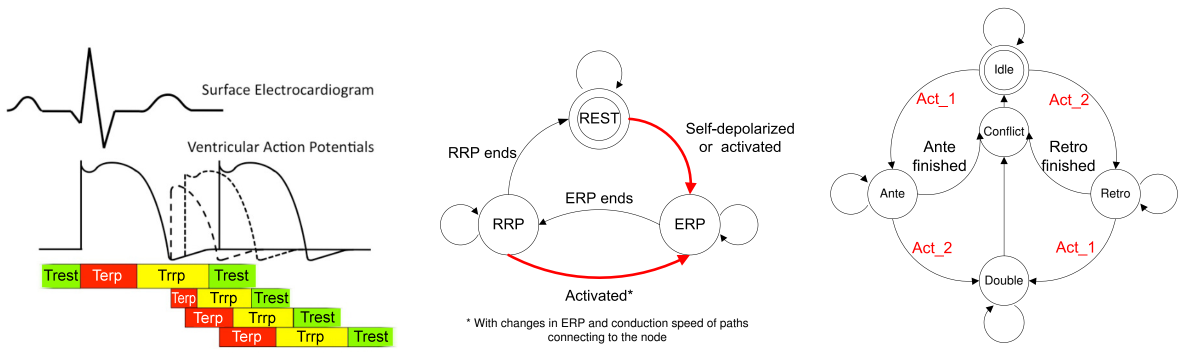
B. *Organization*

The rest of this paper is organized as follows. In Section II, we describe the construction and interfaces of a timed automata based heart model developed in Simulink. Section III presents the model of a pacemaker which interfaces with the heart model to generate a set of closed-loop physiological tests cases described in Section IV. Sections V-A and VI present the test generation and testing of the closed-loop system with a set of monitors. We then conclude the paper with a set of future directions toward more rigorous and formal medical system verification.

II. Heart Model

We developed a timed automata-based Virtual Heart Model (VHM) to model the timing and electrical conduction properties of the heart. The model, developed in Simulink, was constructed to serve as an adaptive test generator for a variety of heart conditions. The VHM responds to both intrinsic (natural) pacing and external (artificial) pacing stimulus and is capable of generating tests for the following common conditions:

1) Normal Sinus Rhythm and Sinus Bradycardia
2) Wenckebach Second-degree Heart Block



(a) (b) (c)

Figure 2. (a) Action potential recorded from ventricular tissue. The dashed lines show how action potential morphology changes when a stimulus is applied early to the tissue and how the corresponding timer values change.(b) Node automaton. (c) Path automaton

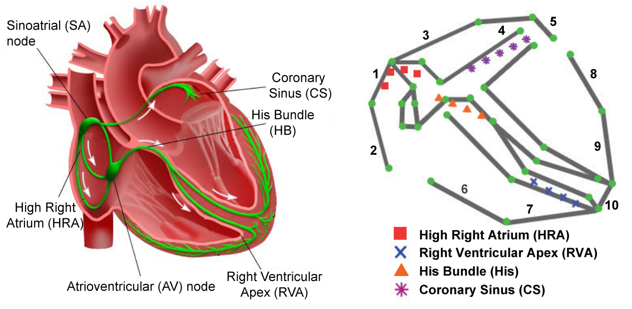
3) Atrial Flutter
4) Pacemaker Mediated Tachycardia

In our previous work [12], several arrhythmias were validated by an electrophysiologist to be clinically relevant. For completeness, we briefly describe the design of the VHM using timed automata. For more details, refer to [12].

A. Timing Model of the Virtual Heart Model

Modeling individual cells in order to obtain a view of the entire heart is processor heavy and contains extraneous information for purposes of device testing. Instead, the VHM utilizes the timing properties of the heart to obtain a macro-level view by lumping cells into *node* and *path automaton*.

The heart can be represented as a conduction network (Fig. 3(a)), because a section of activated heart tissue can only activate its neighboring tissue. For example, an a conduction path from tissue A to tissue B, a stimulus cannot reach B if A is in refractory (i.e. the cells are discharged), even if B is at rest. We can use this idea to model a section of tissue as two node automata connected by one path automaton. The refractory properties of the component are represented by the nodes and the conduction properties between the nodes are modeled by the path. We can represent different structures of the heart using nodes and paths with different parameters. The basic state transitions of the node automaton and the path automaton are shown in Fig. 2(b) and Fig. 2(c). In node automaton, the refractoriness is modeled as the Effective Refractory Period (*ERP*) when no conduction can occur, Relative Refractory Period (*RRP*) when attenuated conduction may occur and *Rest* states when the cell is fully excitable. In path automaton, the tissue conduction properties are modeled as no conduction (*Idle*), antegrade or forward conduction (*Ante*), retrograde or backward conduction (*Retro*), conduction in both directions (*Double*), and conflict (*Conflict*) state. Each state has an internal timer T_{state}, and the paths have conduction velocity V_{state}. The default refractory and conduction parameters are tuned in relation to the true refractory periods measured in clinical electrophysiology studies[15]. These allow us to produce clinically-relevant results [12].



(a) (b)

Figure 3. (a) The basic electrical conduction system of the heart. (b) Corresponding setup of nodes (dots), paths (lines) and probes (shapes) in our heart model.

It has been studied in [16] and [17] that the action potential duration is dependent on the timing of stimulus (see Fig. 2(a)). We use an exponential approximation of the trend to create similar behavior. In addition, the portion of the total refractory period that is unexcitable is dependent on the amplitude of the stimuli [18]. For simplification, we assume a fixed amplitude for all stimuli, which is the case in most implantable devices. The state transitions of node automaton and path automaton are shown below (all timer equations are described in [12]:

1. Node automata state transitions

- *When activation signal is received at a node*
  - In Rest state: go to ERP state, calculate the value of timer T_{erp}, reset the ERP timer, activate neighboring paths and change the T_{ante} or T_{retro} timers.
  - In ERP state: calculate T_{erp}, neighboring paths are not activated.
  - In RRP state: go to ERP state, activate neighboring paths, calculate T_{erp}, and change the conduction velocity of neighboring paths.
- *When no activation signal is received at node*
  - In Rest state: SA and AV nodes count down the Rest timer, when the timer times out, activate itself, go to ERP state, activate neighboring paths and change the conduction speed of neighboring paths. Other nodes stay in Rest state.
  - In ERP state: count down ERP timer and go to RRP state after the timer runs out.
  - In RRP state: count down RRP timer and go to Rest state after the timer runs out.

2. Path automata state transitions: The path automata is initially *Idle* until either of the nodes it connects to is activated (Act_1 or Act_2). The path starts the antegrade or retrograde conduction timer according to which node is activated. After T_{ante} or T_{retro} times out, the path activates the node at the opposite end. Because this node activates all paths it is connected to, the path where the activation originated goes to *Conflict* state to prevent back conduction. If a path is already in antegrade conduction when a second activation signal enters from the opposite end, the path enters the *Double* state. Both T_{ante} and T_{retro} count down until the timers correspond to the same location in space and the path goes to the *Conflict* state then back to *Rest* state.

B. Functional and Formal VHM interface

Our platform provides two interfaces, a formal signal for interfacing with medical device software and a functional electrogram for real device implementation. We introduce *probes* into the model which mimic sensing from electrodes in catheters or pacemaker leads. These probes generate synthetic unipolar electrograms by multiplying activations on neighboring paths by a distance-dependent Gaussian factor and summing them together. The bipolar electrograms are calculated as the differences between two unipolar signals. The bipolar formal signal is generated using an AND operation between the two unipolar formal signals. Varying the location of the probe pair and the conduction delay of the path can causes changes in electrogram morphology. The formal signal is useful for interacting with medical device software running at a lower frequency than the VHM.

C. Simulink Implementation of the VHM

The general automata and probe set used in our simulations is shown in Fig. 3(b) with 32 nodes and 33 paths. The probes are placed in specific areas to capture key activation timing intervals in the heart. A simulation GUI developed in Simulink and the automata network is superimposed on the heart anatomy. The length of paths are measured in pixels to provide a relative length relationship between different heart structures. Users can track the updated values of timers on the right side tables. Users can view electrograms and deliver programmed pacing in real-time. Simulink was chosen because both the functional and formal models could be developed from a common kernel. The VHM interfaces with the pacemaker model, which is described next.

III. PACEMAKER MODEL

A. Pacemaker model

As no pacemaker software or model is available, we developed a pacemaker model to mimic the behavior of a real pacemaker based on pacemaker timing cycles. The 5 basic timing cycles for a dual-chamber DDD mode pacemaker are shown in Fig. 4 [19]. Since these timing cycles are mostly independent of each other, we model them as 5 independent modules running in parallel. This also enable us to use the same model for different pacemaker modes by disabling the corresponding module. Pacemaker models for AAI and DDD modes were implemented and tested. AAI mode is a single chamber mode which senses and paces in the atrium. Its function is to keep the atrial rate above a certain threshold. DDD mode is a dual chamber mode which senses and paces in both atrium and ventricle. Besides maintaining both atrial and ventricular rate, DDD mode also includes A-V synchrony, which can optimize hemodynamics. The description and design of the 5 modules are introduced below:

1. Lowest Rate Interval (LRI): LRI is the most basic timing cycle for pacemaker. Its function is to maintain the heart rate above a certain level. It can be modeled as a timer with some logic. The timer is started and reset by ventricular event. If there is no ventricular event is sensed before the timer runs out, the pacemaker will deliver ventricular pacing.

2. Atrio-Ventricular Interval (AVI): The function of the AVI module is to keep A-V synchrony. The timer is started by atrial events. If no ventricular event happens before the timer runs out, the pacemaker will deliver ventricular pacing.

3. Three auxiliary modules: The post ventricular atrial refractory period (PVARP) module is a blocking period which is started by a ventricular event. During this period no atrial sense event can be triggered. The ventricular refractory period (VRP) module is a blocking interval which is started by a ventricular event. During this period no ventricular sense event can be triggered. These two modules are designed to filter out noise so that only real events are sensed. The upper rate interval (URI) component provides an upper bound for ventricular pacing. AVI timer is extended and the ventricular pacing is withheld to prevent the pacemaker from pacing the ventricle too fast.

Besides the basic timing cycles, a simplified version of Mode-switch operation was implemented. Wit Mode-swtich, the pacemaker should switch from DDD to VDI mode if the atrial rate is above certain threshold and switch from VDI to DDD mode if the atrial rate drops below a certain level. This function is to ensure the A-V synchrony function of the pacemaker does not track abnormally fast atrial rates and cause ventricular tachycardia. A timer is set to monitor the interval between two consecutive atrial events. Any value smaller than a specified threshold is considered as a fast atrial beat. A counter tracks the number of fast atrial beats. If there are certain number of consecutive fast atrial beats the pacemaker will switch from DDD to VDI. During VDI mode the pacemaker paces and senses the ventricle but still keeps track of the interval between atrial beats. If a slow atrial beats is detected, the pacemaker will switch from VDI to DDD mode.

IV. PHYSIOLOGICAL CASE STUDIES

We simulated four common clinically-relevant heart conditions using the closed-loop VHM and pacemaker model. Based on these conditions we derived a set of safety and efficacy requirements for the pacemaker (Section V-A). Following this, we applied monitors to the close-loop system to

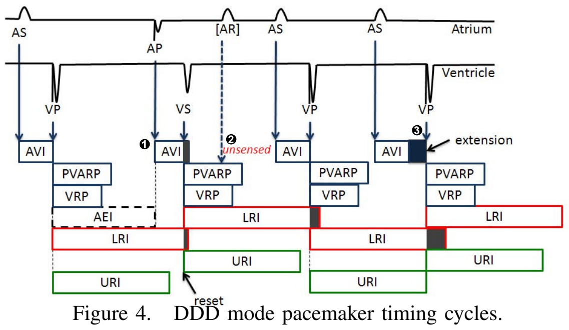


Figure 4. DDD mode pacemaker timing cycles.

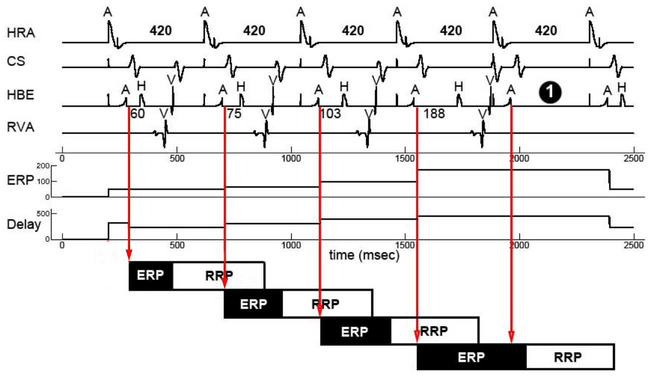


Figure 5. Electrograms of induced Wenckebach block in the heart model with a beat interval of 420ms. The heart model displays lengthening in the A-H interval and block in A-V node (1). Rows 5 and 6 show the increase in the ERP and conduction delay of the A-V node.

evaluate if the requirements are satisfied (Section VI). This section describes the electrophysiological details of each case so that we may understand the clinical relevance prior to testing. The detailed cases and detailed description can be found in [12].

A. *Maintain appropriate heart rate*

Maintaining appropriate heart rate is essential for keeping efficient blood flow throughout the body. Heart rate can be classified into Normal Sinus Rhythm (NSR, i.e. normal heart rate), Bradycardia (i.e. abnormally slow heart rate) and Tachycardia (i.e. abnormally fast heart rate). The mechanism of Sinus Bradycardia is when the *SA node*, which serves as natural pacemaker, fails to generate a fast enough heart rate to satisfy blood demand of the body. Sinus Bradycardia is the most basic and common arrhythmia which requires pacemaker therapy. The pacemaker can provide electrical impulses at the appropriate time intervals to maintain the heart rate above certain level. Pacemaker does not provide Tachycardia therapy but it needs to prevent itself from inducing the heart into the dangerous Tachycardia condition.

B. *Maintain Atrioventricular Synchrony*

Besides heart rate, atrioventricular (AV) synchrony is an important property to be satisfied to maintain efficient hemodynamics. AV heart block is a heart condition where the electrical conduction between the atria and ventricles is delayed or blocked. This is caused by the abnormal behavior of the *AV node*. In this case, we introduce the Wenchebach-type AV block, which features a progressively prolonged AV delay that eventually results in dropped beats. Fig. 5 shows the electrograms, the corresponding ERP period and conduction delay of the AV node automata in a VHM simulation.

From the His Bundle Electrogram (HBE in the figure), we observe the second stimulus which arrived at the AV node (marked as *A* in the figure) falls into the RRP period of the AV node. Consequently, the ERP period and the conduction delay of AV node are increased according to the timer equations referenced in Section II. After a series of stimuli, the ERP period is longer than the interval between two stimuli. This results in the blocking of the last stimulus at the AV node, causing a dropped beat (marker **1** in the figure). In the AV heart block situation, a dual chamber pacemaker should be employed to maintain the AV synchrony by appropriately pacing the atrium and ventricle in a coordinated manner.

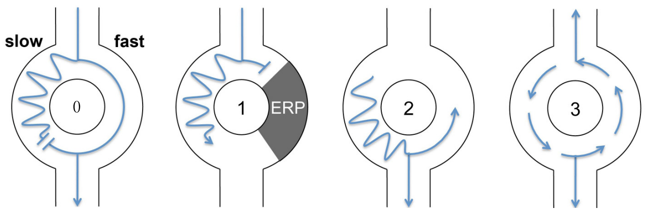
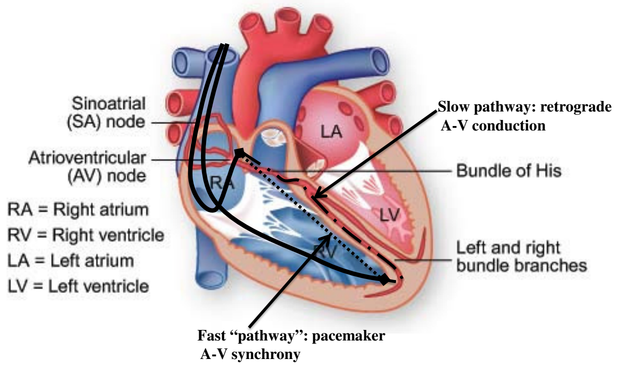


Figure 6. Mechanism of reentry circuit



Sinoatrial (SA) node
Atrioventricular (AV) node
RA = Right atrium
RV = Right ventricle
LA = Left atrium
LV = Left ventricle
LA
RA
LV
RV
Slow pathway: retrograde A-V conduction
Bundle of His
Left and right bundle branches
Fast "pathway": pacemaker A-V synchrony

Figure 7. Illustration of the reentry circuit in Endless Loop Tachycardia

C. *Pacemaker Mediated Tachycardia*

Pacemaker Mediated Tachycardia (PMT) is the circumstance where the pacemaker paces the heart at an inappropriately high rate. In the following two cases, we observe the pacemaker can introduce complications that may lead to unsafe heart conditions.

1) Endless Loop Tachycardia (ELT): is a situation where the pacemaker and heart tissue form a "circuit", which is known as reentry circuit. Reentry circuit is one of the most common reason for Tachy-arrhythmia. Its mechanism is based on timing anomalies caused by additional (accessory) conduction pathways within the heart. In this situation the "additional pathway" is the pacemaker AV synchrony function. Fig. 6 illustrates the mechanism of a reentry circuit.

A reentry circuit is formed by two conduction pathways. One has a short conduction delay but with a long blocking period (ERP), and the other has long conduction delay but with a short blocking period. During normal conduction (Fig. 6-0), stimulus propagates though both pathways. Since the signal through the fast pathway arrives at the circuit exit earlier, it will conflict with the signal through the slow pathway and they cancel each other. In this case, the circuit is equivalent to a single fast pathway. However, if another stimulus enters into the circuit shortly after the previous one (Fig. 6-1), it will be blocked in the fast pathway and go through the slow pathway as the fast pathway has a longer blocking period. When this stimulus arrives the circuit exit (Fig. 6-2), the blocking period of the fast pathway whould have ended and the stimulus propagates retrogradely (i.e. in reverse direction to normal conduction) through the fast pathway. The stimulus will then loop around the circuit and send stimuli to both the circuit's entry and exit (Fig. 6-3). Since the timing interval of circling around the circuit is shorter than the intrinsic heart rate, the circuit will become the dominating source of

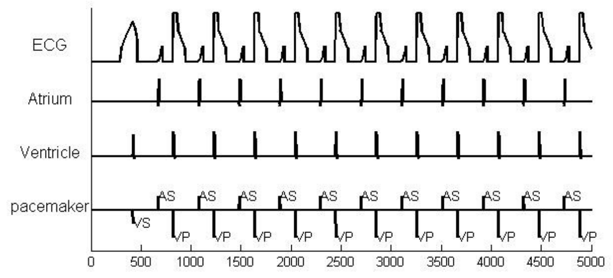


Figure 8. The first ventricle sense (VS) is a Premature Ventricular Contraction (PVC). It propagates retrogradely to the atrium and triggers atrial sense (AS). The AS-VP interval is equal to the AVI timer value of the pacemaker. The VP-AS intervals are equal to the first VS-AS interval, indicating all VPs are also going retrogradely to the atrium. The resulting heart rate is around 150bpm.

activation for the heart and cause Tachycardia. This condition must be detected by the pacemaker and must inhabit pacing that may lead to assisting the Tachycardia condition.

Fig. 7 shows the mechanism of ELT where natural conduction pathway serves as the slow pathway of the circuit and the A-V synchrony function of the pacemaker serves as the fast "pathway". A Premature Ventricle Contraction (PVC) can trigger retrograde conduction through the slow pathway and triggers atrial sense event in the pacemaker. The pacemaker will then pace the ventricle after the AVI timer runs out, triggering another retrograde conduction and causing reentrant tachycardia. The VHM's simulation result is shown in Fig. 8.

2) Atrial Flutter and Pacemaker Mode-Switch function: Supraventricular Tachycardia (SVT) is the most common Tachy-arrhythmia and features a very fast atrial rate. SVT itself is not fatal since a functioning AV node will filter out some of the stimuli to the ventricle and the ventricular rate remains relatively normal. However, complications may arise if there exist other means, besides the AV node, to provide A-V synchrony. This is likely in the form of additional pathways or due to fast atrial "tracking" by a DDD dual chamber pacemaker.

Atrial Flutter is one of the most common SVTs and is caused by reentry circuits. In a large population group, there exists a slow conduction pathway in the right atrium which forms a reentry circuit along with the normal conduction pathway. Fig. 9-a shows synthetic EGMs of an intermittent Atrial flutter case without pacemaker. During Atrial flutter, the atrial rate is around 240bpm but only around half of them conduct to the ventricle so the ventricular rate is around 100bpm. When the Atrial flutter terminates, the heart condition changes to bradycardia.

With a dual chamber pacemaker in DDD mode implanted, the heart rate and A-V synchrony are maintained during Bradycardia. However, during Atrial flutter, the A-V synchrony function of the pacemaker will pace the ventricle for every sensed atrial event and lead to unsafe ventricular tachycardia.(Fig. 9-b) During SVT-like Atrial Flutter, maintaining an appropriate ventricular rate is much more important than maintaining A-V synchrony. In order to prevent inappropriate A-V synchrony, the pacemaker should be able to detect SVT and disable the A-V synchrony function during SVT while keep A-V synchrony during bradycardia and A-V block.

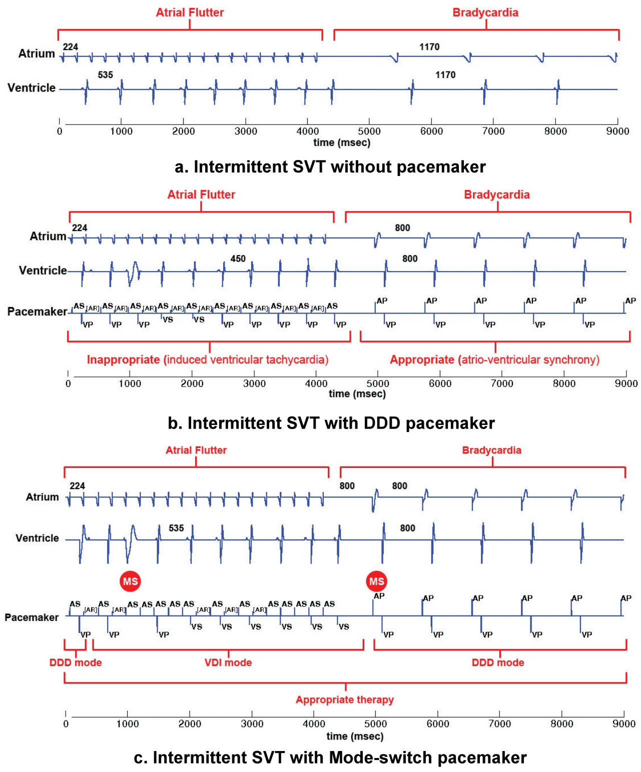


Figure 9. Simulation results for intermittent Supraventricular Tachycardia

A Mode-switch function has been introduced to switch to single chamber mode (VDI) during SVT. Fig. 9-c shows the simulation result for pacemaker with Mode-switch function during SVT.

After detecting five consecutive fast atrial events, the pacemaker switchs from DDD mode to VDI mode (first MS marker in Fig. 9-c), which is a single chamber mode that paces and senses only in the ventricle while monitoring the interval between atrial events. After the SVT terminates, the pacemaker detects a long coupling interval between atrial events and switches back to DDD mode (second MS marker in Fig. 9-c). With the Mode Switch function, the pacemaker delivers appropriate therapy during different heart conditions. A more detailed description can be found in [20].

V. Closed-loop Requirements

In this section, we describe a set of requirements that are imposed on the closed-loop system to guarantee the safety and efficacy of the system. These requirements can not only be used for testing methods proposed here, but also for verification of the closed-loop system.

A. *Patient condition*

From a physiological point of view there are certain requirements that the closed-loop system needs to satisfy. Some properties are absolute and some of them are conditional. For conditional properties, we specify heart states as patient conditions (see Fig. 10). In the closed-loop system these heart states are connected to the heart model parameters and won't be changed by the pacemaker therapy. This will allow us to evaluate whether the pacemaker provides an appropriate therapy for certain types of arrhythmias.

State	Conditions	Equivalence in VHM
NSR	Intrinsic heart rate between 60-120bpm	Trest_SA within [500 1000];
Brady	Intrinsic heart rate<60bpm	Trest_SA>1000
Sinus Tachy	Intrinsic heart rate>120bpm	Trest_SA<500
A-V Block	AV node ERP is long so that A-V conduction is slow or blocked	TERP_AV>400
SVT	Atrial rate above 200 due to reentry circuit	Interval between two Activation_SA is shorter than 250ms && Trest_SA>600

Figure 10. Patient state and equivalence in VHM

B. General requirements

Fig. 11 shows several general requirements that a rhythm-management implantable device should be able to satisfy, as well as the corresponding VHM logic. The requirements range from safety requirements to efficacy requirements and each requirement is assigned a priority. The device should always try to satisfy a higher priority requirement even if a lower priority requirement may be violated. The highest priority requirements have the smallest priority value.

The parameters of the close-loop system which are exposed to the monitors are also specified. *State_[node name]* and *Activation_[node name]* are the current state and activation status of the node automata. *Condition_[heart state name]* is a boolean value for whether the heart state is true. a_p, v_p, a_s and v_s are the pace and sense events of the pacemaker.

1) No Ventricular pace should happen during ventricle refractory (Priority 1): This requirement is the most important safety requirement for implantable cardiac device. Pacing during the ventricle refractory period creates derangements in timing across the ventricle. This disturbs the normal coordinated contraction of the ventricle and can lead to unsafe ventricle fibrillation.

Index	General Requirements	Correspondence in the close-loop system	Priority
1	No ventricular pace should happen during ventricle refractory	If State_RVA=ERP, no v_p should happen	1
2	Each atrial and ventricular event should be sensed by the corresponding lead	Interval between two Activation_RVA should between 500ms to 1000ms	2
3	Ventricular rate should be maintained between 60bpm and 120bpm	If Condition_NSR=true, Interval between two Activation_RVA shouldn't be larger than Trest_SA; If Condition_Brady=true, Interval between two a_p should be equal to LRI timer of pacemaker	3

4	Without activity sensor, the pacemaker should not increase ventricular rate above it's programmed LRI during Brady and above intrinsic heart rate during NSR	If Activation_SA=true, a_s=true;
		If Activation_RVA=true, v_s=true;
			4

5	If the intrinsic heart rate is below some threshold, After each atrial event there should be a ventricular event within some interval(1:1 conduction)	If Interval between two Activation_SA is larger than 600, after each Activation_SA, there should be a Activation_RVA within [100, 150]	5
6	No activation conflict should happen within muscle tissue	musclepaths=path automata 1-10 in Fig 3(b); State_musclepaths should never be *Double*	6

Figure 11. General requirements for the close-loop system

2) Each atrial and ventricular event should be sensed by the corresponding lead (Priority 2): Sensed intrinsic heart activation signals are essential inputs for the pacemaker. The pacemaker should ensure that the actual heart activations are accurately sensed and noise should be filtered.

3) Ventricular rate should be maintained between 60bpm and 120bpm (Priority 3): Ventricular rate determines the actual cardiac output. It is necessary for the pacemaker to keep the ventricular rate above a certain level to ensure the requisite cardiac output and below certain level to ensure efficient pumping.

4) Without an activity sensor the pacemaker should not increase the ventricular rate above its programmed LRI during Bradycardia and above the intrinsic heart rate during NSR (Priority 4): In some cases, the pacemaker may try to pace the heart at inappropriate high rate (like the two PMT cases in Section IV). The pacemaker should always give intrinsic heart signal higher priority and increase the heart rate only when it's necessary.

5) If the intrinsic heart rate is below a certain threshold, after each atrial event there should be a ventricular event within a predefined interval (Priority 5): A 1:1 A-V conduction should be maintained and coordinated to ensure efficient pumping. This is an efficacy requirements of the closed-loop system.

6) No activation conflict should happen within muscle tissue (Priority 6): Activation conflicts within the muscle tissues disturbs the normal contraction pattern which might compromise the efficacy of cardiac output. Consequently, intrinsic and artificially paced signal should ensure coordinated contraction without conflicts across the muscle paths 12-16 (as shown in Fig. 3(a)).

C. Conditional requirements

Besides the general requirements, additional conditional requirements are specified for specific patient condition such as Normal Sinus Rhythm, Bradycardia, Heart Block, Supraventricular tachycardia and Endless Loop Tachycardia. These requirements are shown in Fig. 12 and cover common critical patient conditions.

Condition	Requirement given condition	Correspondence in VHM
NSR=true && A-V block=false	No a_p and v_p should happen	a_p=false; v_p=false
Brady=true && A-V block=false	no a_s and v_p should happen	a_s=false; v_p=false
Brady=true && A-V block=true	no a_s and v_s should happen	a_s=false; v_s=false
SVT=true && State_PM=DDD	Mode switch to VDI should happen within 5s	PM_ModeSwitch=true within 5s && PM_NextState=VDI
SVT=false && State_PM=VDI	Mode switch to DDD should happen if the atrial rate is lower than 60bpm	if Interval between two Activation_SA is larger than 1000, PM_ModeSwitch=true && PM_NextState=DDD

Figure 12. Conditional requirements for the close-loop system

VI. Closed-loop System Testing

Currently there does not exist a unified approach for Simulink models verification. Therefore, we use a testing procedure similar to Instrumentation Based Verification (IBV) [21]. In this procedure, monitors are designed and utilized to check for any violation of the requirements. The proposed approach uses coverage-based testing and, thus, it can be used only in combination with tools for automatic test data generation. Several such commercial (e.g., Reactis, Design Verifier) and non-commercial (e.g. [22], [23]) tools exist. However, to test case studies described in Section IV the closed-loop system input test vectors are created based on physiological requirements.

In this section, we describe design of the monitors used for testing of the requirements presented in Section V-A, a method used to generate input test cases, along with results obtained with these test vectors.

A. Monitor Designs

For each requirement described in Section V-A, a monitor is designed to generate an event when a violation of the property occurs. Using *Assertion block* in Simulink we are able to detect any violation of the imposed closed-loop system requirement and capture the execution scenario (i.e, simulation trace) which has led to the violation.

Design monitors utilized for the closed-loop system testing are grouped into the following categories:

M1. *Monitors for logical conditions:* When a requirement can be described using logical operations over system's signals, it is necessary to check whether at each time instance the predefined combination of signals and/or (VHM and pacemaker's) states might occur. For example, the first condition from Fig. 11 can be described as:

$$(State_{RVA} == ERP) \wedge (\neg V_{pace}).$$

M2. *Single event monitor:* A subset of the requirements can be described as a combination of time dependencies between consecutive appearances of a single event. For example, the second requirement from Fig. 11 will be satisfied if time durations between all consecutive RVA activation signals belong to a region $[0.5s,\ 1s]$. A single event monitor shown in Fig. 13 is utilized to detect any violations for this type of requirements. Values for T_{low} and T_{high} are used to determine the required time span between consecutive manifestations of the same event.

M3. *Monitors for timing dependencies between different events:* A subset of requirements imposes conditions that an event E_2 always appears only after an event E_1. In addition, the time duration between these two events has to be within a predefined bound. For example, consider Requirement 5 from Fig. 11. It imposes a condition that the SA activation signal is always followed by the RVA activation signal. In addition, the RVA activation signal must not occur before 100ms have elapsed from the SA activation or later than 150ms after the SA activation signal (i.e., lower and upper bound). The monitor used to check for violations for this type of requirements is presented in Fig. 14 where the time interval is described as $[T_{low},\ T_{high}]$.

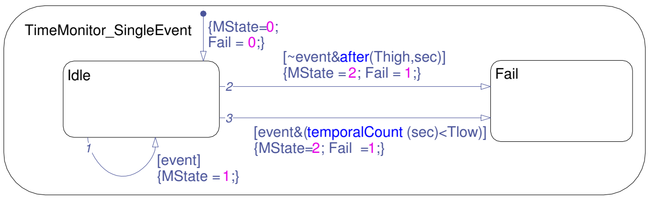


Figure 13. A Single Event Monitor.

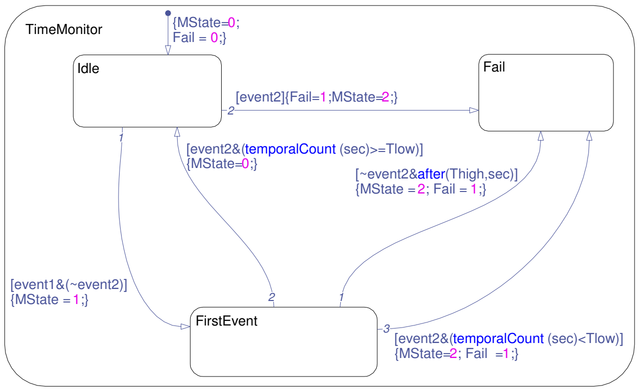


Figure 14. Design of a monitor used to check timing dependencies between events.

M4. *Combination monitors:* In general, a requirement can be described as a combination of the previously described requirements (from **M1-3**). In this case, the aforementioned monitors are composed to generate intermediate signals/events. However, it is necessary to modify the monitors presented in Fig. 13 and Fig. 14. These 'modified' monitors have a similar structure as **M1-M3** with a difference that they do not contain the $Fail$ state and all transitions to the $Fail$ state in the initial monitor designs are altered to become transitions to the $Idle$ state. These modifications enable a continuous tracking of the intermediate properties that would not be possible if the monitor gets blocked in the $Fail$ state, after the appropriate scenario occurs.

To illustrate this, consider a monitor in Fig. 15 used to check the fifth requirement from Fig. 11. The monitor combines a 'modified' Single Event Monitor. In this case, by setting the monitor's parameters $T_{low} = 0s$ and $T_{high} = 0.6s$, its output signal (i.e., $TM\_SingleEvent.Fail$) represents the condition that the heart rate is below the predefined threshold. In addition, a 'modified' monitor of type **M3** with parameters $T_{low} = 0.1s$ and $T_{high} = 0.15s$ is used to check time durations between SA and RVA activation signals. Outputs of these 'modified' monitors (i.e., signals $TM\_SingleEvent.Fail$ and $TM.Fail$) are combined to generate a logical signal whose occurrence represents a violation of the tested property.

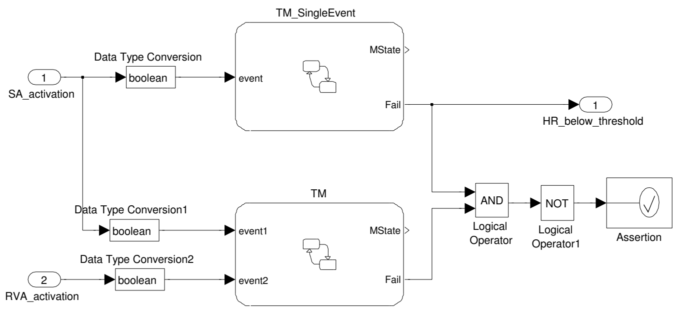


Figure 15. An example of a combination monitor derived as a composition of the basic monitors from Fig. 13 and Fig. 14.

Case 1 & 2: Maintaining heart rate and A-V synchrony												
Requirements	NSR						Brady					
	No AV block			AV block			No AV block			AV block		
	No PM	AAI PM	DDD PM	No PM	AAI PM	DDD PM	No PM	AAI PM	DDD PM	No PM	AAI PM	DDD PM
1: No v_p during refractory	N/A	Check	Check	N/A	Check	Check	N/A	Check	Check	N/A	Check	Check
2: Events sensed	N/A	Check	Check	N/A	Check	Check	N/A	Check	Check	N/A	Check	Check
3: Maintain Ventricle rate	Check	Check	Check	Violated	Violated	Check	Violated	Check	Check	Violated	Violated	Check
4: Rate increase when necessary	N/A	Check	Check	N/A	Check	Check	N/A	Check	Check	N/A	Check	Check
5: A-V synchrony	Check	Check	Check	Violated	Violated	Check	Check	Check	Check	Violated	Violated	Check
6: No muscle tissue conflict	Check	Check	Check	Check	Check	Check	Check	Check	Check	Check	Check	Check
NSR=true && A-V block=false	N/A	Check	Check	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
Brady=true && A-V block=false	N/A	N/A	N/A	N/A	N/A	N/A	N/A	Check	Check	N/A	N/A	N/A
Brady=true && A-V block=true	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	Check	Check

Figure 16. Simulation results for Maintain heart rate and A-V synchrony

B. Generation of Test Vectors

To test the close-loop system we need to specify the inputs for the heart model to cover certain physiologically relevant heart conditions. We now specify inputs of the VHM for the four cases discussed in section IV. In our test approach, the inputs of the heart model are only those heart parameters which can change over time. This property significantly reduces the number of inputs and the model complexity.

1) Maintaining the heart rate: The heart rate is a function of the blood demand of the body and is controlled by the nervous system. The nervous system controls the heart rate by changing the firing rate of the SA node. Failure to increase the heart rate when needed leads to Bradycardia. Correspondingly, we can alter the heart rate of the VHM by changing the T_{rest} timer value of the SA node automata. In this base case, we model the heart rate variation of a patient with intermittent Bradycardia. The range of T_{rest} of the SA node automata is confined to $[600, 1500]ms$ so the corresponding heart rate is between 40bpm and 100bpm. This input sequence for SA node automata will also be used for the latter cases.

2) Maintaining A-V synchrony: The nervous system controls the A-V conduction delay/block by changing the ERP timer of the AV node. Correspondingly, we mimic the behavior in VHM by changing the T_{ERP} timer value of AV node automata. For simplicity, we only specify two distinct values for the T_{ERP} timer value of AV node automata, which correspond to the state “AV-block” and “No AV block”.

3) Endless Loop Tachycardia (ELT): In this case, we are interested in how Pacemaker Mediated Tachycardia can influence the requirements satisfaction. Here we use the same input in Case 2 for the SA and AV node automata, and introduce Premature Ventricle Contraction (PVC) to induce Endless Loop Tachycardia. In this case, we set the RV node automata to be active 200ms after the simulation begin.

4) Atrial flutter and pacemaker Mode-switch function: In this case we are interested in checking if the pacemaker with Mode-switch function can switch from DDD mode to VDI mode during SVT and switch back after SVT terminates. For simplicity and clinical relevance, we induce the Atrial flutter by setting the fast pathway to refractory and activate the node automata at the circuit entry. This will mimic the behavior when an early atrial signal successfully gets into the circuit at the right time and induce Atrial flutter. We terminate atrial flutter by disabling the slow pathway.

C. Results

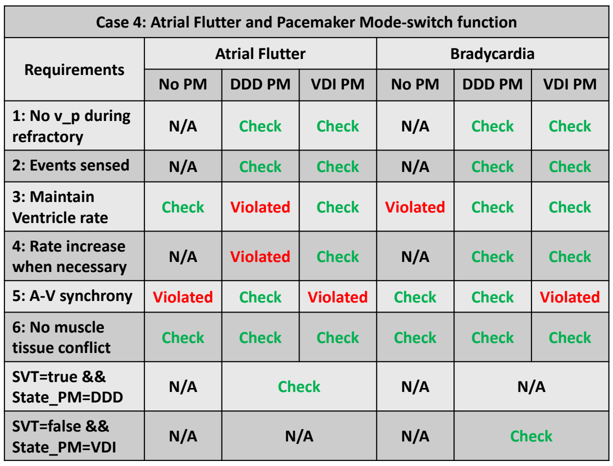
1) Maintain heart rate and A-V synchrony: We combined Case 1 and 2 together and ran the simulation with monitors. The results are shown in Fig. 16. From the table we can see that during AV block, general Requirements 3 and 5 are violated and applying an AAI pacemaker will not help. In this situation, a DDD pacemaker can satisfy all the requirements so the therapy is appropriate for a patient with AV-block.

2) Endless Loop Tachycardia: The simulation result is shown in Fig. 17. We can see from the table that after the ELT has been induced, varying the heart condition will not stop it and does not affect the outcome. In fact the pacemaker “Hijacked” the heart rate and puts the patient into a fixed and potentially unsafe condition.

3) Atrial flutter and pacemaker Mode-switch function: The result of the simulation is shown in Fig. 18. We observe that during Atrial flutter, the close-loop system with a DDD pacemaker can satisfy Requirement 5 but will violate Requirements 3 and 4. With a VDI pacemaker, Requirements 3 and 4 are satisfied but Requirement 5 is violated. Since there’s no single mode to satisfy Requirements 3, 4 and 5 during SVT, using VDI mode during Atrial flutter is more appropriate since Requirements 3 and 4 have higher

Case 3: Endless Loop Tachycardia				
Requirements	NSR		Bradycardia	
	No AV block	AV block	No AV block	AV block
1: No v_p during refractory	Check	Check	Check	Check
2: Events sensed	Check	Check	Check	Check
3: Maintain Ventricle rate	Violated	Violated	Violated	Violated
4: Rate increase when necessary	Violated	Violated	Violated	Violated
5: A-V synchrony	Check	Check	Check	Check
6: No muscle tissue conflict	Check	Check	Check	Check

Figure 17. Simulation results for Endless Loop Tachycardia



Case 4: Atrial Flutter and Pacemaker Mode-switch function						
Requirements	Atrial Flutter			Bradycardia		
	No PM	DDD PM	VDI PM	No PM	DDD PM	VDI PM
1: No v_p during refractory	N/A	Check	Check	N/A	Check	Check
2: Events sensed	N/A	Check	Check	N/A	Check	Check
3: Maintain Ventricle rate	Check	Violated	Check	Violated	Check	Check
4: Rate increase when necessary	N/A	Violated	Check	N/A	Check	Check
5: A-V synchrony	Violated	Check	Violated	Check	Check	Violated
6: No muscle tissue conflict	Check	Check	Check	Check	Check	Check
SVT=true && State_PM=DDD	N/A	Check		N/A	N/A	
SVT=false && State_PM=VDI	N/A	N/A		N/A	Check	

Figure 18. Simulation results for Atrial flutter and acemaker Mode-switch

priority. During Bradycardia a DDD pacemaker can satisfy all general requirements thus it's the appropriate therapy. From the conditional requirements (last two row in the table) we observe that the mode-switch function is working properly. The pacemaker mode-switch function is appropriate for patient with intermittent SVT. Failure to switch modes in corresponding heart condition will result in violating the requirements and cause inappropriate therapy.

VII. Discussion

A. Future Work

As a part of our future efforts we plan to improve generation of the test vectors that would allow heart rate smoothening over a desired interval of time. In addition, it would be beneficial to consider a more complex pacemaker model which would include Rate Response modes (e.g., DDDR mode) specified in [9], [10]. Finally, as an avenue of the future work we plan to investigate methods for the closed-loop system verification. With this approach our goal is to translate the VHM from Simulink into UPPAAL [24], a widely used tool for system verification, where properties have to be expressed using temporal logic formulas before they are checked. Due to the limitations of the monitor based approach, a migration to UPPAAL presents a natural way to verify closed-loop system using the timed automata framework.

B. Conclusion

There is currently no software testing standard for implantable medical devices such as pacemakers and cardioverter defibrillators. We present a method for testing of pacemakers within the closed-loop context of a heart model. A set of general and patient condition-specific temporal requirements is specified for the closed-loop system. Based on these requirements, we presented an interactive and physiologically relevant model-based test generation for basic and complex pacemaker operations. With the use of monitors, we demonstrate that the proposed system is capable of testing common and complex heart conditions across a variety of pacemaker modes. This system is a step toward a testing approach for medical cyber-physical systems with the patient-in-the-loop.

References

[1] W. H. Maisel et. al. Recalls and Safety Alerts involving Pacemakers and Implantable Cardioverter-Defibrillator Generators. *JAMA*, 286(7), 2001.

[2] List of Device Recalls, U.S. Food and Drug Admin., (last visited Jul. 19, 2010).

[3] K. Sandler, L. Ohrstrom, L. Moy, and R. McVay. Killed by Code: Software Transparency in Implantable Medical Devices. *Software Freedom Law Center*, 2010.

[4] US FDA Center for Devices and Radiological Health, Medical Devices; Current Good Manufacturing Practice (CGMP) Final Rule; Quality System Regulation, Oct. 1996.

[5] US FDA Center for Devices and Radiological Health, Guidance for the Content of Premarket Submissions for Software Contained in Medical Devices, May 2005.

[6] US FDA Center for Devices and Radiological Health, General Principles of Software Validation; Final Guidance for Industry and FDA Staff, Jan. 2002.

[7] J. Rushby. Automated test generation and verified software. 4171:161–172, 2008.

[8] Personal communication with Paul L. Jones, Center for Devices and Radiological Health, Office of Science and Engineering Laboratories, FDA. August, 2010.

[9] PACEMAKER System Specification. Boston Scientific. 2007.

[10] A. O. Gomes and M. V. Oliveira. Formal specification of a cardiac pacing system, 2009.

[11] E. Jee, I. Lee, and O. Sokolsky. Assurance cases in model-driven development of the pacemaker software. In *Leveraging Applications of Formal Methods, Verification, and Validation*, volume 6416 of *LNCS*, pages 343–356. 2010.

[12] Z. Jiang, M. Pajic, A. T. Connolly, S. Dixit, and R. Mangharam. Real-time heart model for implantable cardiac device validation and verification. In *22nd Euromicro Conference on Real-Time Systems, (IEEE ECRTS'10)*, July 2010.

[13] Testing Implantable Medical Devices, J. M. Cortner, Guidant, Evaluation Engineering, June 2004.

[14] *Medtronic ViP-II Virtual Interactive Patient: User's Manual Software v1.5*. Rivertek Medical Systems, 2006.

[15] M. E. Josephson. *Clinical Cardiac Electrophysiology*. Lippincot Williams and Wilkins, fourth edition, 2008.

[16] P. Denes, D. Wu, R. Dhingra, R. J. Pietras, and K. M Rosen. The Effects of Cycle Length on Cardiac Refractory Periods in Man. *Circulation*, 49:32–41, 1974.

[17] M. R. Franz, C. D. Swerdlow, L. B. Liem, and J. Schaefer. Cycle Length Dependence of Human Action Potential Duration In vivo. *J. Clin. Invest.*, 82(3):972–979, Sep 1988.

[18] A. E. Buxton et. al. The Human Atrial Strength-Interval Relationship. *Circulation*, 79(2), 1989.

[19] S. S. Barold, R. X. Stroobandt, and A. F. Sinnaeve. *Cardiac Pacemakers Step-by-Step: An Illustrated Guide*. Wiley-Blackwell, 2003.

[20] Z. Jiang, A. Connolly, and R. Mangharam. Using the virtual heart model to validate the mode-switch pacemaker operation. *32nd Intl. Conf. IEEE Engineering in Medicine and Biology Society*, 2010.

[21] C. Ackermann et. al. Model based design verification - a monitor based approach. In *Society of Automotive Engineers, World Congress*, 2008.

[22] R. Alur, A. Kanade, S. Ramesh, and K. C. Shashidhar. Symbolic analysis for improving simulation coverage of simulink/stateflow models. In *EMSOFT '08: Proc. 8th ACM Intl. Conf. on Embedded software*, pages 89–98, 2008.

[23] A. Kanade et. al. Generating and analyzing symbolic traces of simulink/stateflow models. In *21st Intl. Conf. on Computer-Aided Verification*, 2008.

[24] K.G. Larsen, Paul Pettersson, and Wang Yi. Uppaal in a nutshell. *International Journal on Software Tools for Technology Transfer (STTT)*, 1997.