Modeling and Verification of a Dual Chamber Implantable Pacemaker*

Zhihao Jiang, Miroslav Pajic, Salar Moarref, Rajeev Alur, Rahul Mangharam
University of Pennsylvania, Philadelphia PA, USA

Abstract. The design and implementation of software for medical devices is challenging due to their rapidly increasing functionality and the tight coupling of computation, control, and communication. The safety-critical nature and the lack of existing industry standards for verification, make this an ideal domain for exploring applications of formal modeling and analysis. In this study, we use a dual chamber implantable pacemaker as a case study for modeling and verification of control algorithms for medical devices in UPPAAL. We begin with detailed models of the pacemaker, based on the specifications and algorithm descriptions from Boston Scientific. We then define the state space of the closed-loop system based on its heart rate and developed a heart model which can non-deterministically cover the whole state space. For verification, we first specify unsafe regions within the state space and verify the closed-loop system against corresponding safety requirements. As stronger assertions are attempted, the closed-loop unsafe state may result from healthy open-loop heart conditions. Such *unsafe transitions* are investigated with two clinical cases of Pacemaker Mediated Tachycardia and their corresponding correction algorithms in the pacemaker. Along with emerging tools for code generation from UPPAAL models, this effort enables model-driven design and certification of software for medical devices.

Keywords: Medical Devices, Implantable Pacemaker, Software Verification, Cyber-Physical Systems

1 Introduction

Over the past four decades, cardiac rhythm management devices such as pacemakers have expanded their role from "keeping the patient alive" to "making the patient's life comfortable". The addition of more safety and efficacy features has resulted in increased complexity, inevitably leading to more safety violations. From 1996-2006, the percentage of software-related causes in medical device recalls have grown from 10% to 21% [1]. During the first half of 2010, the US Food and Drug Administration (FDA) issued 23 recalls of defective devices, all of which are categorized as *Class I*, meaning there is a "reasonable probability that use of these products will cause serious adverse health consequences or death." At least six of the recalls were caused by software defects [2]. Unlike other industries such as aviation and automotive, the safety concern in the medical device domain is focused on the physical plant, the patient in this case, rather than the controller. As a result, although in aviation and automotive industries,

* This research was partially supported by NSF research grants MRI 0923518, CNS 0931239, CNS 1035715 and CCF 0915777.

standards are enforced during software development, manufacturing, and post-market change [3, 4], there are no well-established standards for development of software for medical devices. There is a pressing need for standards and tools to certify and verify the safety of software in medical devices. For device manufacturers, this has prompted recent interest in applying formal modeling and verification techniques in medical devices software development [5, 6].

In this effort, we propose a Timed Automata representation of the heart and a dual chamber pacemaker. Our models and specifications are designed based on descriptions available from Boston Scientific [7, 8], a leading manufacturer of pacemakers, and extensive medical literature on this topic. We then demonstrate how a model checker, like UPPAAL [9], can be used to find safety violations and prove the correctness of medical device algorithms. We define the state space of the closed-loop system based on its heart rate. Unsafe regions can then be specified and the closed-loop system is verified against corresponding safety requirements. We also define *unsafe transitions* as the controller drives the open-loop plant from a safe state into an unsafe closed-loop state. We focus on two cases of unsafe transitions which are referred to as "Pacemaker Mediated Tachycardia (PMT)". Modern pacemakers are equipped with correction algorithms to terminate these behaviors. We demonstrate how to identify known unsafe transitions and prove the correctness of corresponding correction algorithms using model checker. The UPPAAL model developed in this paper is freely available online [10]. These models can be used as a starting point for many purposes (e.g. to build models with costs and probabilities for quantitative analysis of the efficacy of pacemaker algorithms; development of patient-specific algorithms). In particular, the verified pacemaker model can be automatically translated into Stateflow charts in Simulink for test generation and code generation [11].

The paper is organized as follows: In Section 2, we introduce the physiological and timing basics of the heart and pacemaker. Section 3 presents UPPAAL models of the basic DDD pacemaker and the heart. In Section 4, we define unsafe regions and verify the basic pacemaker model against corresponding safety requirements. In Section 5, we proposed a procedure for identifying and verifying unsafe transitions and demonstrated using two cases of PMT.

2 Heart and Pacemaker Basics

The coordinated contraction of the heart is governed by its Electrical Conduction System (see Fig. 1). The Sinoatrial (SA) node, which is a collection of specialized tissue at the top of the right atrium, periodically spontaneously generates electrical pulses that can cause muscle contraction. The SA node is controlled by the nervous system and acts as the natural pacemaker of the heart. The electrical pulses first cause both atria to contract, forcing the blood into the ventricles. The electrical conduction is then delayed at the Atrioventricular (AV) node, allowing the ventricles to fill fully. Finally the fast-conducting His-Pukinje system spreads the electrical activation within both ventricles, causing simultaneous contraction of the ventricular muscles, and pumps the blood out of the heart.

Due to aging and/or diseases, the conduction properties of heart tissue may change. These changes may cause timing anomalies in heart rhythm, thus decrease the blood pumping efficiency of the heart. These timing anomalies are referred to as arrhythmias, and are categorized into *Tachycardia* and *Bradycardia*. Tachycardia features undesirable fast heart rate which impairs hemodynamics. Bradycardia features slow heart rate which results in insufficient blood supply. Bradycardia maybe due to failure of impulse generation with anomalies in the SA node, or failure of impulse propagation where the conduction from atria to the ventricles is delayed or blocked.

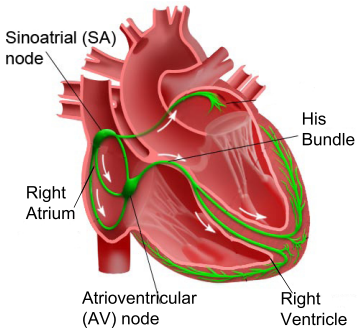


Fig. 1. Cardiac electrical system

Since the heart tissue can be activated by external electrical pulses, Bradycardia can be treated by providing electrical pulses when the heart rate is low. *Implantable Pacemakers* have been developed to deliver timely electrical pulses to the heart to maintain an appropriate heart rate and Atrial-Ventricular synchrony. Implantable pacemakers normally have two leads fixed on the wall of the right atrium and the right ventricle respectively. Activation of local tissue is sensed by the leads, triggering Atrial Sense (AS) and Ventricular Sense (VS) events. Atrial Pacing (AP) and Ventricular Pacing (VP) are delivered if no sensed events occur within deadlines.

In order to deal with different heart conditions, modern pacemakers are able to operate in different modes. The modes are labeled using a three character system. The first character describes the pacing locations, the second character describes the sensing locations, and the third character describes how the pacemaker software responds to sensing. In this work we describe the most commonly used mode of pacemaker, the dual-chamber DDD mode that paces both the atrium and the ventricle, senses both chambers, and sensing can both activate or inhibit further pacing. Similarly, the VDI mode paces only in the ventricle, senses both chambers, and inhibits pacing if event is sensed. [12]

3 System Modeling

3.1 Timed Automata and UPPAAL

Timed automaton [13] is an extension of a finite automaton with a finite set of real-valued clocks. It has been used for modeling and verifying systems which are triggered by events and have timing constraints between events. From the Boston Scientific pacemaker specification [7], the pacemaker can be modeled using this Extended Timed Automata notation, which is a subset of formal semantics in UPPAAL. UPPAAL ([9, 14]) is a standard tool for modeling and verification of real-time systems, based on networks of timed automata. The graphical and text-based interface makes modeling more intuitive. Requirements can be specified using Computational Tree Logic (CTL) [15] and violations can be visualized in the simulation environment.

3.2 System Overview

The function of a pacemaker is to manage the timing relationship between the atrial and ventricular events. Thus Timed Automata is suitable for modeling both the deterministic behavior of a pacemaker and the non-deterministic behavior of the heart. The overview of the closed-loop system is showed in Fig. 2(a). The heart and the pacemaker communicate with each other using broadcast channels. The heart generates *Aget!* and *Vget!* actions, representing atrial and ventricular events that the pacemaker take as inputs. The pacemaker processes the signals and generates pacing actions *AP!* and *VP!* to the corresponding components in the heart.

3.3 Basic DDD pacemaker modeling

The DDD pacemaker has 5 basic timing cycles triggered by events, as shown in Fig. 2(b). We decomposed our pacemaker model into 5 components which correspond to the 5 counters. These components communicate with each other using broadcast channels and shared variables (as shown in Fig. 3).

Lower Rate Interval (LRI): This component keeps the heart rate above a minimum value. In DDD mode, the LRI component models the basic timing cycle which defines the longest interval between two ventricular events. The clock is reset when a ventricular event (VS, VP) is received. If no atrial event has been sensed (AS), the component will deliver atrial pacing (AP) after TLRI-TAVI. The UPPAAL design of LRI component is shown in Fig. 3(a).

Atrio-Ventricular Interval (AVI) and Upper Rate Interval (URI): The function of the AVI component is to maintain the appropriate delay between the atrial activation and the ventricular activation. It defines the longest interval between an atrial event and a ventricular event. If no ventricular event has been sensed (VS) within TAVI after an atrial event (AS, AP), the component will deliver ventricular pacing (VP). In order to prevent the pacemaker from pacing the ventricle too fast, a URI component uses a global clock *clk* to track the time after a ventricular event (VS, VP). The URI limits the ventricular pacing rate by enforcing a lower bound on the times between consecutive ventricle events. If the global clock value is less than TURI when the AVI component is about to deliver VP, AVI will hold VP and deliver it after the global clock reaches TURI. The UPPAAL design of AVI and URI component is shown in Fig. 3(b) and (c).

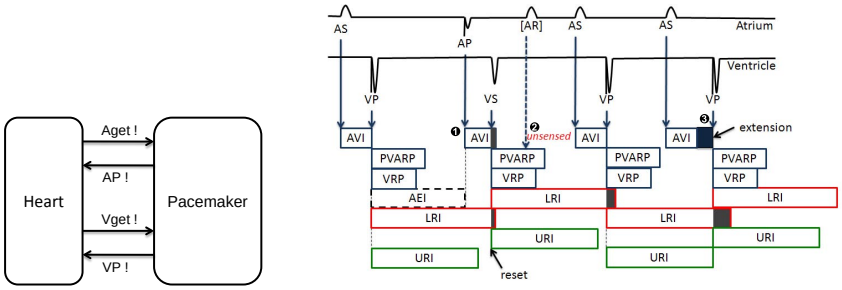


Fig. 2. (a) System Overview, (b) Basic 5 timing cycles of DDD pacemaker

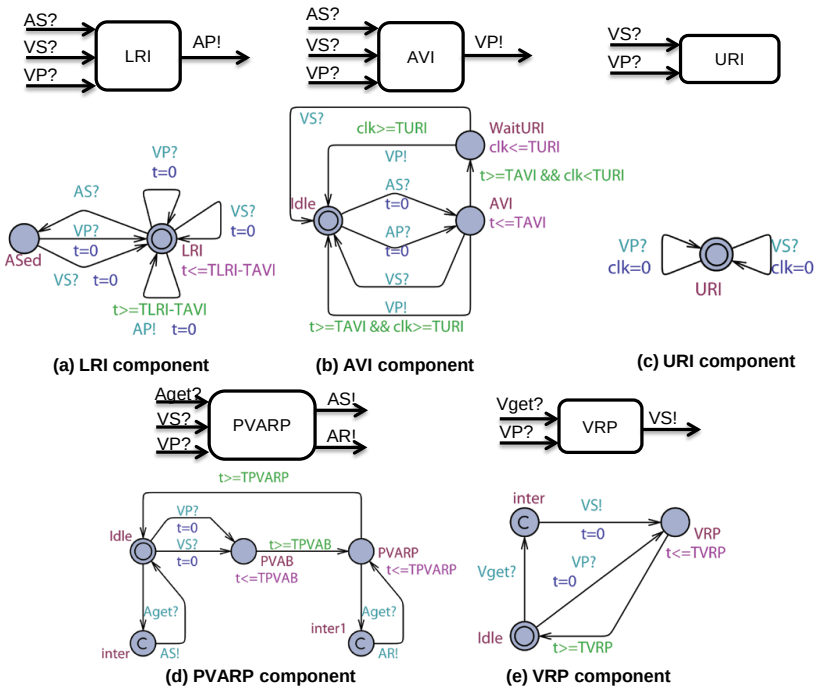


Fig. 3. Components of the pacemaker model in UPPAAL

Post Ventricular Atrial Refractory Period (PVARP) and Post Ventricular Atrial Blanking (PVAB): Not all atrial events (*Aget!*) are recognized as Atrial Sense (*AS!*). After each ventricular event, there is a blanking period (PVAB) followed by a refractory period (PVARP) for the atrial events in order to filter noise. Atrial events during PVAB are ignored and atrial events during PVARP trigger *AR!* event which can be used in some advanced diagnostic algorithms. The UPPAAL design of PVARP component is shown in Fig. 3(d).

Ventricular Refractory Period (VRP): Correspondingly, the VRP follows each ventricular event (VP, VS) to filter noise and early events in the ventricular channel which could otherwise cause undesired pacemaker behavior. Fig. 3(e) shows the UPPAAL design of VRP component.

Parameter Selection: Each timing parameter of the pacemaker has a feasible range. However, after those parameters are programmed, they are fixed during pacemaker operation. Consider all possible combinations of feasible parameter values is infeasible. In this work, we only verify one instance of a DDD pacemaker with nominal values in clinical settings [8]. The values we choose are TAVI=150, TLRI=1000, TPVARP=100, TVRP=150, TURI=400, TPVAB=50.

3.4 Random Heart Model (RHM)

In order to verify pacemaker algorithm, we need to first define the state space for the closed-loop system. The state space definition should not only cover all possible pacemaker operations, but also be

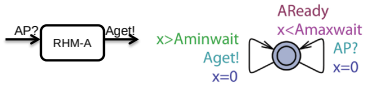


Fig. 4. RHM for the Atrial Channel

physiologically intuitive for safety requirement specification. To this end, we define the state space of the closed-loop system by the atrial interval (interval between atrial events $\in \{AS, AP\}$) and ventricular interval (interval between ventricular events $\in \{VS, VP\}$). This heart rate representation enables us to define unsafe regions for bradycardia and tachycardia.

The Random Heart Model (RHM) is designed to cover open-loop heart behaviors. It non-deterministically generates an intrinsic heart event *Xget!* within [*Xminwait*, *Xmaxwait*] after each intrinsic heart event *Xget* or pacing *XP*. Here we use two RHMs for the atrial and ventricular channel where X can be atrial (A) or ventricular (V). RHM covers all possible input to the pacemaker if the interval [*Xminwait*, *Xmaxwait*] is set to $[0, \infty]$. It can also cover subset of possible heart conditions by assigning appropriate values to those two parameters. The UPPAAL model of the atrial RHM is shown in Fig. 4.

4 Verification regarding unsafe regions

In this section, we define unsafe regions regarding bradycardia and tachycardia and specify two basic safety properties. These two basic safety properties are strict so that they must be satisfied by any pacemaker under all heart conditions. We then discuss refinement of the safe regions and make stronger assertions.

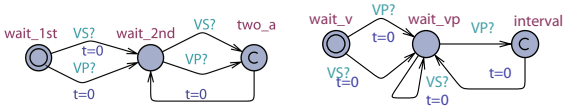
4.1 Lower Rate Limit

The most essential function for the pacemaker is to treat bradycardia by maintaining the ventricular rate above a certain threshold. We define the region where the ventricular rate is slow, as *unsafe*. The monitor *Pvv* is designed to measure interval between ventricular events and is shown in Fig. 5(a). The property *A[] (Pvv.two_a imply Pvv.t≤TLRI)* is satisfied by the basic DDD pacemaker.

4.2 Upper Rate Limit

The pacemaker is not designed to treat tachycardia so it can only pace the heart to increase its rate and cannot slow it down. However, it is still important to guarantee it does not pace the ventricles beyond a maximum rate to ensure safe operation. To this effect, an upper rate limit is specified such that the pacemaker can increase the ventricular rate up to this limit.

We require that a ventricle pace (VP) can only occur at least *TURI* after a ventricle event (VS, VP). The monitor for the property is shown in Fig. 5(b) and the property *A[] (PURI_test.interval imply PURI_test.t≥TURI)* is satisfied by the basic DDD pacemaker model.



(a) (b)

Fig. 5. (a) Monitor for LRL: Interval between two ventricular events should be less than TLRI, (b) Monitor for URL: Interval between a ventricular event and a VP should be longer than TURI

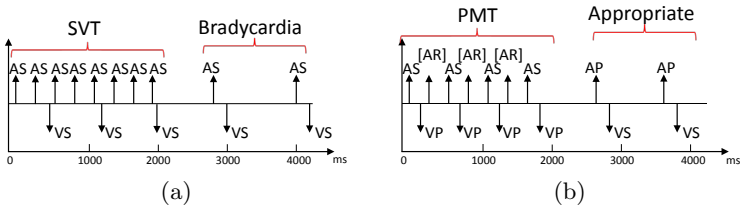


Fig. 6. (a) SVT with ODO pacemaker (b) SVT with DDD pacemaker

5 Verification regarding unsafe transitions

The two unsafe regions, introduced above, are intuitive but provide for loose safety properties. One may wonder if we can further reduce the safe region. When the closed-loop system is in some unsafe state, there are two possible scenarios. One is when, the open-loop plant without the controller, is also in unsafe state. In our case, if the heart is in tachycardia, the pacemaker is not supposed to react so that this case is of little value to us. The other scenario is that the open-loop plant is in a safe state and the controller is driving the closed-loop system into some unsafe states. We call this scenario *Unsafe Transition*. In our case, the pacemaker may increase the heart rate inappropriately, which is referred to as Pacemaker Mediate Tachycardia (PMT).

We now introduce two cases of PMT and their corresponding correction algorithms. Since one closed-loop state may correspond to multiple execution traces, these PMT scenarios will not be returned by the model checker as counter-examples of safety requirements. However, we can still identify known PMT by adding constraints to the heart model or developing more complex requirements.

5.1 Verification Procedure

The pacemaker manufacturers have developed anti-PMT algorithms to terminate different PMT scenarios. In this section, we propose a general procedure to identify PMT scenarios and verify the safety and correctness of anti-PMT algorithms. The general steps for the procedure include:

1. Show existence of PMT behaviors in the closed-loop system
2. Introduce anti-PMT algorithms and check whether the two basic safety requirements still hold
3. Prove correctness of anti-PMT algorithms by showing the non-existence of PMT scenarios

Here we use two well-identified PMT cases to demonstrate the methodology.

5.2 Verification of the Mode-Switch algorithm

Supraventricular Tachycardia (SVT): SVT is an arrhythmia which features an abnormally fast atrial rate. Typically the AV node, which has a long refractory period, can filter most of the fast atrial activations during SVT thus the ventricular rate remains relatively normal. Fig. 6(a) demonstrates a pacemaker event trace during SVT, with a ODO mode pacemaker which just sensing in both channels. In this particular case, every 3 atrial events (AS) correspond to 1 ventricular event (VS) during SVT.

As an arrhythmia, SVT is still considered as a safe heart condition since the ventricles operate under normal rate can still maintain adequate cardiac output. However, the AVI component of a dual chamber pacemaker is equivalent to a virtual pathway in addition to the intrinsic conduction pathway between the atria and the ventricles. The pacemaker tries to maintain 1:1 A-V conduction and thus increases the ventricular rate inappropriately. Fig. 6(b) shows the pacemaker trace of the same SVT case with DDD pacemaker. Although half of the fast atrial events are filtered by the PVARP period ([AR]s), the DDD pacemaker still drives the closed-loop system into 2:1 A-V conduction with faster ventricular rate, which is inappropriate. This problem can be resolved by switching pacemaker into single chamber mode to maintain appropriate ventricular rate.

Existence of PMT during SVT: Since PMT during SVT is an unsafe transition, we need to first adjust the heart model so that the open-loop behaviors covers SVT and are in the safe region. To this end, the interval for the ventricular RHM is set to [500,800]. This rate is slow enough not to be considered as tachycardia, but faster than the Lower Rate Limit of the pacemaker so that pacemaker should not intervene. The monitor *Pv_v* is designed to show existence of PMT during SVT. It goes to the error state if the ventricular rate drops below the Upper Rate Limit (Fig. 7).

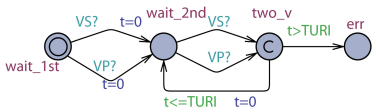


Fig. 7. Monitor for SVT: Check existence of an endless sequence where the ventricular event interval $\leq$TURI

The existence property $E[](notPv\_v.err)$ is specified, which verifies if there exists an execution in which the ventricular interval is always less or equal to TURI. The property is first verified on pacemaker without the mode-switch algorithm. The property is satisfied during verification.

Mode-Switch algorithm: Intuitively, the mode-switch algorithm first detects SVT. After confirmed detection, it switches the pacemaker from a dual-chamber mode to a single-chamber mode. During the single-chamber mode, the A-V synchrony function of the pacemaker is deactivated thus the ventricular rate is decoupled from the fast atrial rate. After the algorithm determines the end of SVT, it will switch the pacemaker back to the dual chamber mode.

The mode-switch algorithm specification we use is the same as the one used in Boston Scientific pacemakers [8]. The algorithm first measures the interval between atrial events outside the blanking period (AS, AR). The interval is considered as *fast* if it is above a threshold (*Trigger Rate*) and *slow* otherwise (see Fig. 8 (1)). A counter increments for *fast* events and decrement for *slow* events (see Fig. 8 (2)). After the counter value reaches the *Entry Count*, the algorithm will start a *Duration* which is a time interval used to confirm the detection of SVT (see Fig. 8 (3)). In the *Duration*, the counter keeps counting. If the counter value is still positive after the *Duration*, the pacemaker will switch to the VDI mode (*Fallback mode*). In the VDI mode, the pacemaker only senses and paces the ventricle. At any time if the counter reaches zero, the *Duration* will terminate and the pacemaker is switched back to DDD mode.

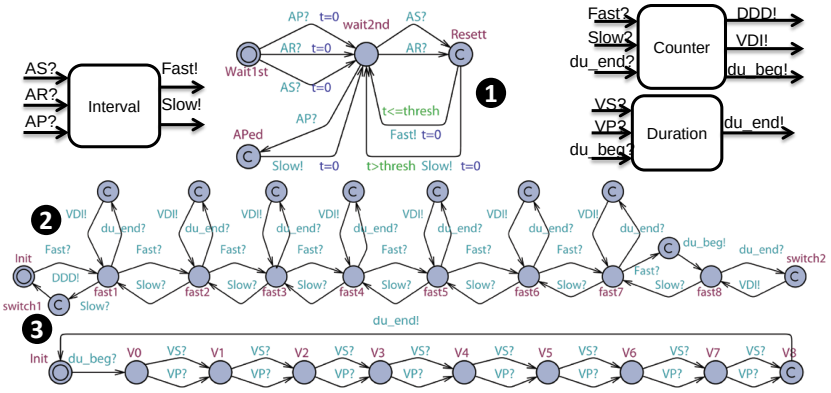


Fig. 8. Mode-Switch algorithm

In our UPPAAL model of the mode-switch algorithm, we use nominal parameter values from the clinical setting. We define *trigger rate* at 170bpm (350ms), *entry count* at 8, *duration* for 8 ventricular events and *fallback mode* as VDI.

In order to model both DDD and VDI modes and the switching between them, we made modifications to the AVI and LRI components. In each component two copies for both modes are modeled, and switch between each other when switching events (DDD, VDI) are received. During VDI mode, VP is delivered by the LRI component instead of the AVI component. The clock values are shared between both copies in order to preserve essential intervals even after switching. The modified AVI and LRI components are shown in Fig. 9.

Verification against basic safety requirements: We verify the same basic safety requirements on the pacemaker model with mode-switch algorithm. The Upper Rate Limit property still holds but the Lower Rate Limit property is violated. When the pacemaker is switching from VDI mode to DDD mode, the responsibility to deliver VP switched from LRI component to AVI component. Since the clock reference is different (Ventricular events in LRI and Atrial events in AVI), the clock value for delivering the next VP is not preserved. As a result, if an atrial event which triggered the mode-switch from VDI to DDD happens within [TLRI-TAVI, TLRI) after the last ventricular event, the next ventricular pacing will be delayed by at most TAVI time, which violates the Lower Rate Limit property (Fig. 11(a)).

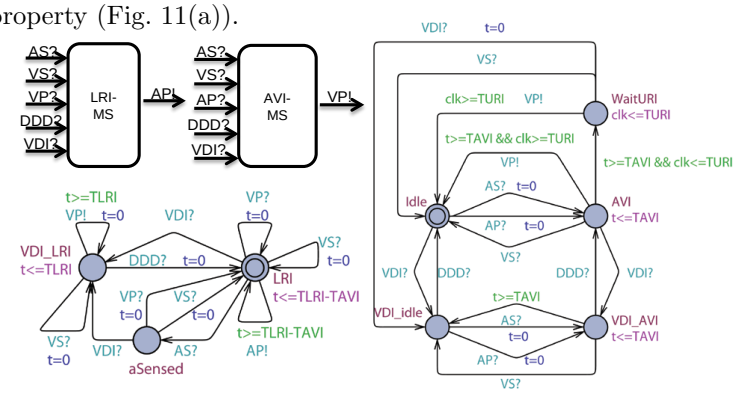


Fig. 9. New LRI & AVI components

Fig. 10. Monitor for Mode-Switch: Check if mode-switch to VDI mode will always eventually happen.

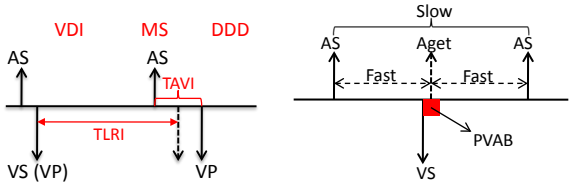
Verification of the algorithm: We now present the verification of the correctness of the mode-switch algorithm by checking the same existence property *E[] (not Pv_v.err)* on pacemaker with mode-switch algorithm. We expect the violation of this property, since during VDI mode the ventricular rate of the heart model is less than the Upper Rate Limit and will not trigger ventricular pacing. The counter example of the violation should show that mode-switch algorithm successfully switches the mode of the pacemaker to VDI mode. However, this property is still satisfied, indicating the mode-switch algorithm failed to eliminate the PMT scenario. Then we further restrict the atrial interval of RHM to [100, 200]. Since the atrial rate for the new heart model is always above the trigger rate, mode switch to VDI mode should always eventually happen. The monitor PMS for the new property is shown in Fig. 10.

The property *A<> (PMS.err)* is not satisfied. The counter-example shows that some of the atrial events fall into the Post Ventricular Atrial Blanking period (PVAB) and got ignored. As a result, two *fast* intervals may be considered as one *slow* interval (see Fig. 11(b)). If this happens more than one out of the *Entry Count*, mode-switch from DDD to VDI may never happen.

Discussion: We demonstrated that model checking techniques can be used to identify unknown violations which cannot be identified during open-loop testing, showing the necessity and usefulness of formal verification in medical device software development and certification. We also showed that adding new features to the verified system is a potential source for safety violations.

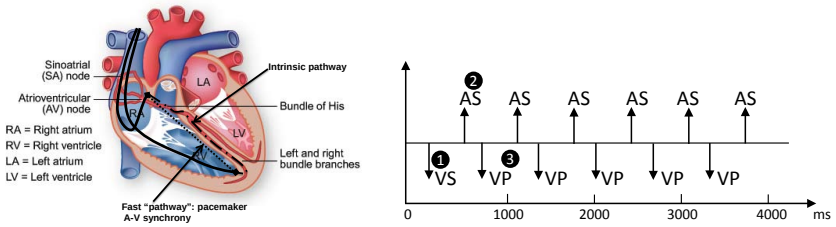
5.3 Verification of Endless Loop Tachycardia (ELT) algorithm

ELT overview: The AVI component of a dual-chamber pacemaker introduces a virtual A-V pathway which forms a loop with the intrinsic A-V conduction pathway (see Fig. 12(a)). In this scenario, a ventricular event (VS) triggers a V-A conduction through the intrinsic pathway (Marker 1 in Fig. 12(b)). The pacemaker registers this signal as an Atrial Sense (AS) (Marker 2 in Fig. 12(b)). This event triggers VP after TAVI, as if the signal conducts through the virtual A-V pathway (Marker 3 in Fig. 12(b)). The VP will trigger another V-A conduction and this VP-AS-VP-AS looping behavior will continue (see Fig. 12(b)).



(a) (b)

Fig. 11. (a) Safety Violation: VP is delayed due to the reset of timer during mode-switch, (b) Correctness Violation: The blocking period may block some atrial events, turning two *Fast* events to one *Slow* event



(a) Virtual circuit formed by the pacemaker and the heart

(b) Pacemaker trace for ELT initialized by a early ventricular signal

Fig. 12. Endless Loop Tachycardia case study demonstrating the situation when the pacemaker drives the heart into an unsafe state [16]

The interval between atrial events is TAVI plus the V-A conduction delay, which will drive the ventricular rate as high as the Upper Rate Limit.

From the pacemaker's point of view, the pacemaker paces the ventricles as specified for every AS. That is why open-loop testing is unable to detect this closed-loop behavior. Modern pacemakers are equipped with anti-ELT algorithms to identify and terminate potential ELT. One common algorithm identifies ELT by the ELT pattern and terminates ELT by increasing TPVARP time once to block the AS caused by the V-A conduction.

Existence of ELT: As another case of unsafe transition, we again constrain the open-loop heart model into healthy heart. We set both the atrial interval and the ventricular interval above TURI so that ELT behavior is not covered by the heart model. Two monitors were designed to show the existence of ELT. One monitor, *PELT_det*, shows the persistence of the VP-AS pattern and the other monitor, *Pvv*, shows that the ventricular rate is always no slower than the upper rate limit (Fig. 13). The existence property *E[] ((not PELT_det.err) && (not Pvv.err))* fails on pacemaker without an anti-ELT algorithm.

The reason for the failure is that in our closed-loop system, AS can only be triggered by *Aget* signal from the atrial heart model, where in ELT case the AS is triggered by backward V-A conduction, which is not covered by our heart model. In order to solve this problem, we model the A-V conduction of the heart in addition to the orignal RHM. The adjusted RHM and the conduction component is shown in Fig. 14. For each atrial event *Aget*, the conduction component generates *V_act* after certain delay and vice versa. The conduction is non-deterministic so that the old RHM is a special case for the new RHM. The PVARP and VRP components are also modified to accommodate new events *A_act* and *V_act*.

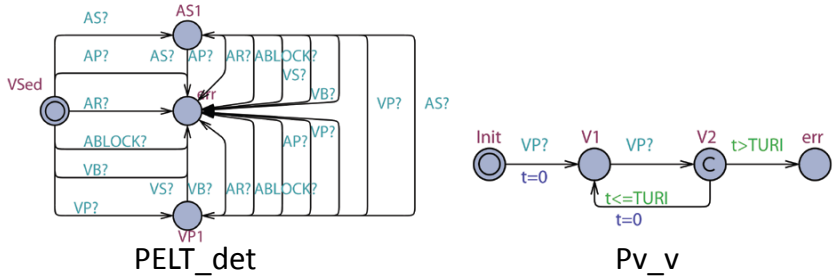


Fig. 13. Monitor for ELT: VP-AS pattern detection and Upper Rate detection

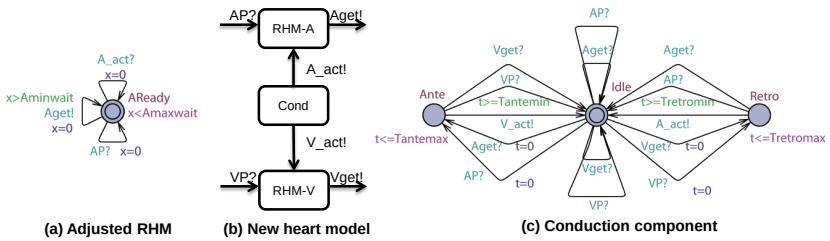


Fig. 14. Modified heart model and the conduction component

After introducing the conduction component, the existence property holds, indicating the closed-loop system with new heart model covers ELT.

The ELT-termination algorithm: The ELT detection algorithm by Boston Scientific [7] utilizes these three features:

- Ventricular rate at Upper Rate Limit
- VP-AS pattern
- Fixed V-A conduction delay

The pacemaker first monitors VP-AS pattern with ventricular rate at upper rate limit. Then it compares the VP-AS interval with previous intervals. ELT is confirmed if the difference between the current VP-AS interval and the first VP-AS interval are within ±32ms for 16 consecutive times. Then the pacemaker increases the PVARP period to 500ms once so that the next AS will be blocked and will not trigger a VP. ELT will then be terminated.

As the V-A conduction delays are patient-specific, the algorithm compares VP-AS interval to a previously sensed value instead of an absolute value. Since we can not store past clock values in UPPAAL, we can not explicitly model this ELT detection algorithm. However, since the conduction delay in our heart model is within a known range, we can compare the VP-AS interval with this range. The VP-AS pattern detection module for our anti-ELT algorithm is shown in Fig. 15 (1). It detects the VP-AS pattern with ventricular rate at upper rate limit and sends out *VP_AS* event if the interval qualifies.

A counter counts the number of qualified VP-AS patterns. It increases the PVARP period to 500ms if eight consecutive VP-AS patterns are detected. (Fig. 15 (2)) The PVARP component is also modified so that the PVARP period can only be changed once by the anti-ELT algorithm. (Fig. 15 (3))

Verification against bottom-line safety requirements: The two bottom-line safety requirements still hold when the anti-ELT algorithm is introduced.

Verification of the algorithm: The existence property *E[]((not PELT_det.err) && (not Pvv.err))* is not satisfied after the anti-ELT algorithm is introduced, indicating the algorithm successfully terminates ELT. We successfully reproduced the case when the algorithm works in the simulation environment of UPPAAL.

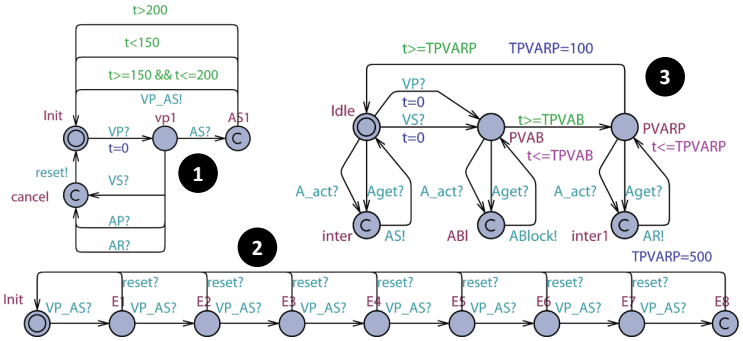


Fig. 15. Counter for VP-AS pattern

Discussion: In this case study, we showed that we may require the heart model to provide more physiological details when verifying more complex properties. We also observed some limitations of Timed Automata when modeling more complex algorithms.

6 Related Work

Jee et. al present a safety assured development approach of real-time software using pacemaker as their case study in [17]. They formally model and verify a single chamber VVI pacemaker using UPPAAL and then implement it and check the preservation of properties transferred from model to implementation code. Tuan et. al propose an RTS formal model for pacemaker and its environment and verified it against number of safety properties and timed constraints using PAT model checker [18]. They have modeled the pacemaker for all 18 operating modes as described in Boston scientific, but their work lacks specification and analysis os complex behaviors of the pacemaker, such as mode-switch.

Wiggelinkhuizen uses mCRL2 and UPPAAL to formally model the pacemaker from the firmware design of Vitatron's DA+ pacemaker [19]. Two main approaches have been used to investigate the feasibility of applying formal model checking to the design of device firmware. The main approach consists of verifying the firmware model in context of a formal heart model and a formal model of a hardware module which fails for high heart rates because of the state explosion. Another approach is to verify a part of firmware design which was feasible and was able to detect a known deadlock rather soon.

Macedo et. al have developed a concurrent and distributed real-time model for a cardiac pacemaker through a pragmatic incremental approach. The models are expressed using the VDM and are validated primarily by scenario-based test, where test scenarios are defined to model interesting situations such as the absence of input pulses [20]. The models cover 8 modes of pacemaker operation.

Gomes et. al present a formal specification of pacemaker system using the Z notation in [21]. They have also tried to validate that the formal specification satisfies the informal requirements of Boston Scientific by using a theorem prover, ProofPower-Z. They have partially checked the consistency of their specification

through reasoning. No validation experiment regarding safety conditions were performed yet. [21]

Mery et. al in [22], formally model all operational modes of a single electrode pacemaker system using event-B and prove them. They use an incremental proof-based approach to refine the basic abstract model of the system and add more functional and timing properties. They use the ProB tool to validate their models in different situations such as absence of input pulses.

7 Conclusion and Future Work

In this paper, we modeled a dual-chamber pacemaker with advanced features using Timed Automata. Timed automaton captures key features of the closed-loop system and enables the use of tools like UPPAAL in verification. We then verified one instance of a dual chamber pacemaker model with nominal parameter values since it is impossible to consider all possible combinations. We defined a heart rate representation of closed-loop state space and identified unsafe regions and unsafe transitions. We demonstrated that model checking techniques can be used to reveal safety violations which cannot be identified during open-loop testing. We also showed that adding features to previously verified system may result in safety violations. Furthermore, we showed that more complex heart model is need to provide more physiological insights during property specification. The UPPAAL model developed in this paper is freely available online [10]. We hope that these models can be used as a starting point for many purposes (e.g. to build models with costs and probabilities for quantitative analysis).

In this paper, we only verified the safety and correctness of pacemaker algorithms. However, the ultimate goal for a pacemaker is to maintain the efficiency of the heart. As future work, we would like to evaluate the efficiency of those algorithms by assigning costs for different heart conditions. The evaluation can be used to develop better treatment for general and specific patients. More complex heart models are therefore needed to provide physiological insights. However, rigorous heart model refinement should be considered to ensure model consistency. While Timed Automata is a good fit for the problem studied here, it also has some drawbacks as it can not capture certain behaviors of some advanced algorithms like memorizing difference of clocks, and is also not scalable enough. Our future work will also focus on improving the efficiency of verification toolchain for medical device certification.

References

[1] List of Device Recalls, U.S. Food and Drug Admin., (last visited Jul. 19, 2010).
[2] K. Sandler, L. Ohrstrom, L. Moy, and R. McVay. Killed by Code: Software Transparency in Implantable Medical Devices. *Software Freedom Law Center*, 2010.
[3] AUTOSAR website: www.autosar.org/.
[4] AVSI website: http://www.avsi.aero.
[5] R. Alur, D. Arney, E. L. Gunter, I. Lee, J. Lee, W. Nam, F. Pearce, S. Van Albert, and J. Zhou. Formal Specifications and Analysis of the Computer-Assisted

Resuscitation Algorithm (CARA) Infusion Pump Control System. *Intl. Journal on Software Tools for Technology Transfer (STTT)*, 5:308–319, 2004.

[6] Annette ten Teije et. al. Improving medical protocols by formal methods. *Artificial Intelligence in Medicine*, 36(3):193 – 209, 2006.

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

[8] The Compass - Technical Guide to Boston Scientific Cardiac Rhythm Management Products. 2007.

[9] K.G. Larsen, Paul Pettersson, and Wang Yi. Uppaal in a Nutshell. *International Journal on Software Tools for Technology Transfer (STTT)*, pages 134–152, 1997.

[10] Zhihao Jiang, Miroslav Pajic, Salar Moarref, Rajeev Alur, and Rahul Mangharam. Pacemaker UPPAAL model download: http://www.seas.upenn.edu/∼zhihaoj/VHM/PM_verify.zip.

[11] M. Pajic, Z. Jiang, O. Sokolsky, I. Lee, and R. Mangharam. From Verification to Implementation: A Model Translation Tool and a Pacemaker Case Study. In *18th IEEE Real-Time and Embedded Technology and Applications Symposium (IEEE RTAS)*, 2012.

[12] S. Barold, R. Stroobandt, and A. Sinnaeve. *Cardiac Pacemakers Step by Step*. Blackwell Futura, 2004.

[13] R. Alur and D. L. Dill. A Theory of Timed Automata. *Theoretical Computer Science*, 126:183–235, 1994.

[14] Gerd Behrmann, Alexandre David, and Kim G. Larsen. A Tutorial on Uppaal. *Formal Methods for the Design of Real-Time Systems, Lecture Notes in Computer Science*, pages 200–236, 2004.

[15] Edmund M. Clarke and E. Allen Emerson. Design and synthesis of synchronization skeletons using branching-time temporal logic. In *Logic of Programs, Workshop*, pages 52–71, 1982.

[16] Z. Jiang, M. Pajic, and R. Mangharam. Model-based Closed-loop Testing of Implantable Pacemakers. In *ICCPS'11: ACM/IEEE 2nd Intl. Conf. on Cyber-Physical Systems*, 2011.

[17] E. Jee, S. Wang, J. K. Kim, J. Lee, O. Sokolsky, and I. Lee. A Safety-Assured Development Approach for Real-Time Software. *The Proceedings of 16th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications*, pages 133–142, 2010.

[18] L. A. Tuan, M. C. Zheng, and Q. T. Tho. Modeling and Verification of Safety Critical Systems: A Case Study on Pacemaker. *Fourth International Conference on Secure Software Integration and Reliability Improvement*, pages 23–32, 2010.

[19] J. E. Wiggelinkhuizen. Feasibility of Formal Model Checking in the Vitatron Environment. *Master thesis, Eindhoven University of Technology*, 2007.

[20] Macedo H. D., Larsen P. G., and Fitzgerald J. Incremental Development of a Distributed Real-Time Model of a Cardiac Pacing System using VDM. *Formal Methods*, pages 28–30, 2008.

[21] A. O. Gomes and M. V. Oliveira. Formal Specification of a Cardiac Pacing System. *In Proceedings of the 2nd World Congress on Formal Methods (FM '09)*, pages 692–707, 2009.

[22] D. Mery and N. K. Singh. Pacemaker's Functional Behaviors in Event-B. *Research report, INRIA*, 2009.