PACEMAKER - A Modelling, Design and Analysis Paradigm

Project Description

Pacemaker is a small embedded device that is placed in the chest or in the abdomen of a person to help control abnormal heart results. In general, the device uses electric pulses to prompt the heart to beat at the normal rate. Pacemakers are used to treat arrhythmias. Arrhythmias are problems with the rate of the heartbeat that is the heart can beat too fast called the Tachycardia or too slow called the Bradycardia or with an irregular rhythm.

The pacemaker monitors and controls your heartbeat. The electrode in the pacemaker detects your heart's electrical activity and sends data through the wires to the generator in the dual chamber pacemaker system and then responds back by sending pulses from the generator to the right atrium and the right ventricle. The pulses help coordinate the timing of these two chambers' contractions.

Video Demo Link
https://drive.google.com/a/iiits.in/file/d/13n6zNU8Gg-dqy2hsqyAGfz6heNkuhQaj/view?usp=drivesdk
Design Report Link
https://docs.google.com/document/d/1-L5rXNNU4sJh73Ejr20c7-m5t5xO9E5uVveLPtmCR-Y/edit?usp=sharing

UPPAAL

Uppaal is a tool box for modeling, simulation and verification of real-time systems, based on constraint-solving.It is appropriate for systems that can be modeled as a collection of nondeterministic processes with finite control structure and real-valued clocks, communicating through channels and/or shared variables.

Uppaal consists of three main parts: a description language, a simulator, and a model-checker. The description language is a non-deterministic guarded command language with data types. It serves as a modeling or design language to describe

system behavior as networks of timed automata extended with data variables. The simulator and the model-checker are designed for interactive and automated analysis of system behavior by manipulating and solving constraints that represent the state space of a system description. The simulator enables examination of possible dynamic executions of a system during early modeling (or design) stages and thus provides an inexpensive means of fault detection prior to verification by the model-checker which covers the exhaustive dynamic behavior of the system.

Modelling in UPPAL

To facilitate modeling, Uppaal provides both graphical and textual formats for the description language. One can use either the textual format or the Autograph-based graphical user interface to define system descriptions, namely networks of timed automata. We mostly used the Autograph-based graphical user interface to model our system descriptions and requirements. Also in uppaal models can be thought of as individual functions and we have used textual format to mention their input/output relationships.

Analysis in UPPAL

1. **Model-checking**

The model-checker is designed to check for invariant and reachability properties, in particular whether certain combinations of control-nodes and constraints on clocks and integer variables are reachable from an initial configuration.

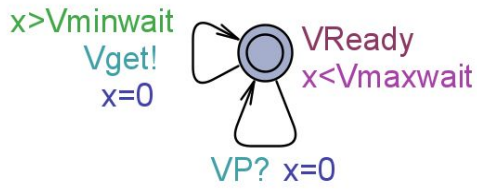
2. **Simulation**

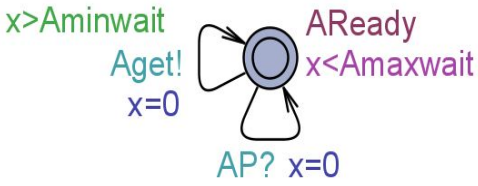
The simulator allows the user to examine in an interactive and graphical fashion the dynamic behavior of a system.

Modelling

In the modeling part of the project, we define our system to be simply a combination of Heart and Pacemaker models. A physical system is one realized in matter, in contrast to a conceptual or logical system such as software and algorithms where our physical system includes the human heart . The heart model imitates and abstracts the properties of a real human heart system which is an open-loop system, and yields insights into that system.

1. **Random Heart Model (RHM):** We define the atrial interval (interval between atrial events ∈ {AS,AP}) and ventricular interval (interval between ventricular events ∈ {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 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,∞]. It can also cover a subset of possible heart conditions by assigning appropriate values to those two parameters.

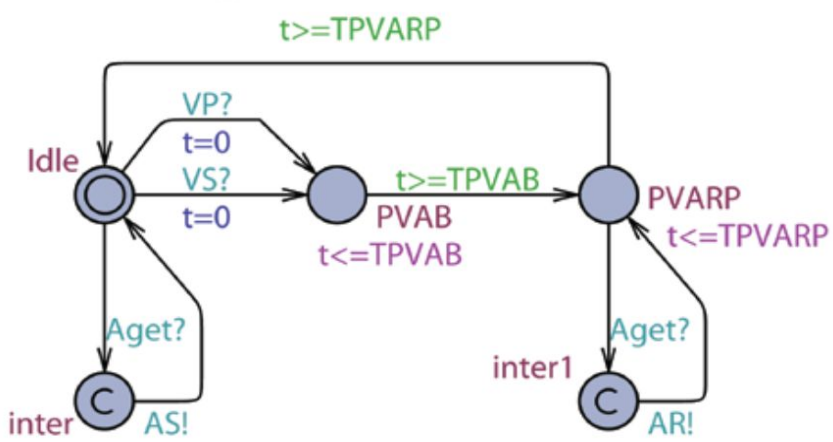




2. Interfacing Models

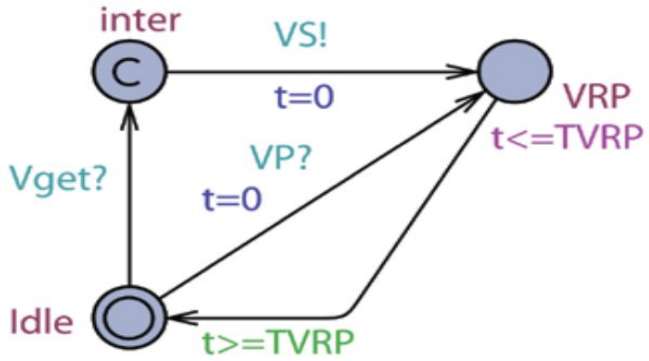
1) **Post Ventricular Atrial Refractory Period (PVARP) and Post Ventricular Atrial Blanking (PVAB):**

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.



2) **Ventricular Refractory Period (VRP):**

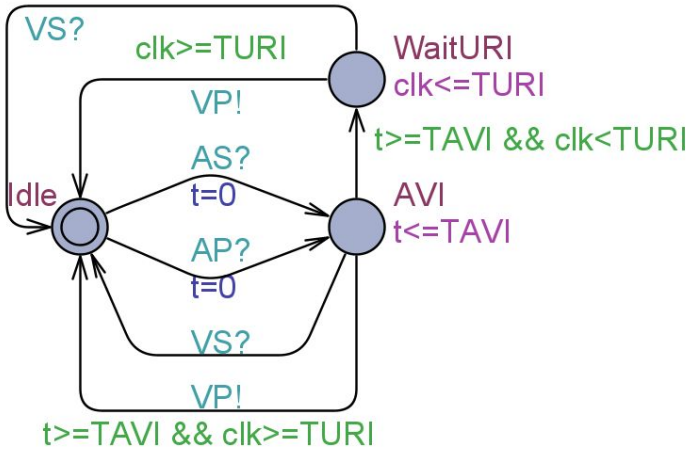
Correspondingly, the VRP follows each ventricular event (VP, VS) to filter noise and early events in the ventricular channel which could otherwise cause undesired pacemaker behavior.

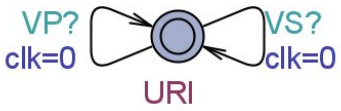


3. Pacemaker

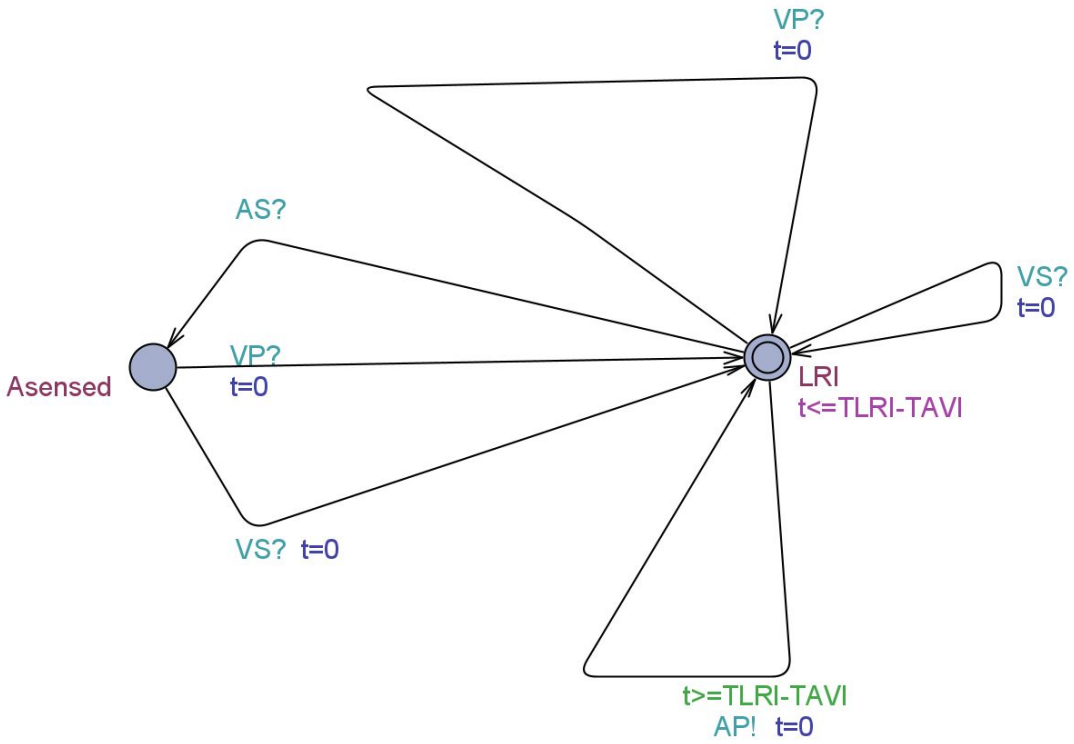
1) **Atrio-Ventricular Interval (AVI) and Upper Rate Interval (URI):** The function of the AVI component is to maintain the appropriate delay between the atrial activation and the ventricular activation. It defines the longest interval between an atrial event and a ventricular event. If no ventricular 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.





2) **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.

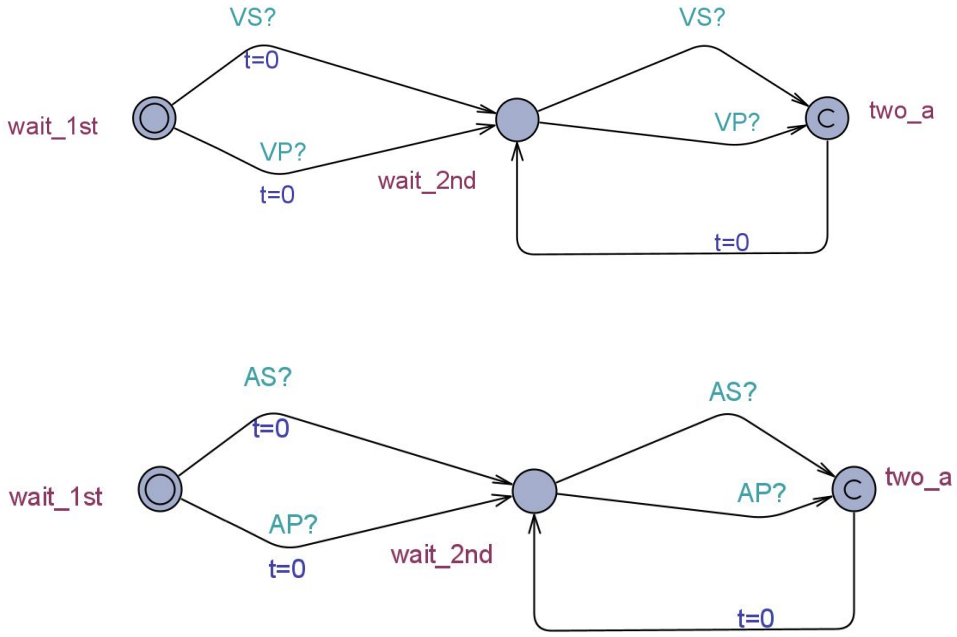


Analysis

Here, we define unsafe regions regarding bradycardia and specify basic safety properties. These two basic safety properties are strict so that they must be satisfied by any pacemaker under all heart conditions.

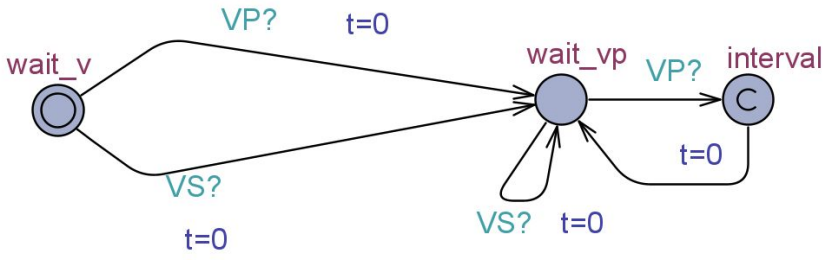
1. **Lower Rate Limit**

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 intervals between ventricular events.The property A[] (A[] mon_lri.two_a imply mon_lri.t<=TLRI) is satisfied by the basic DDD pacemaker.



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 property (A[] mon_uri.interval imply mon_uri.t>=TURI) is satisfied by the basic DDD pacemaker model.



3. **Endless Tachycardia Loop**

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 and VP-AS-VP-AS looping behavior will continue. 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.

The ELT-termination algorithm

The ELT detection algorithm these three features:

– Ventricular rate at Upper Rate Limit

– VP-AS pattern

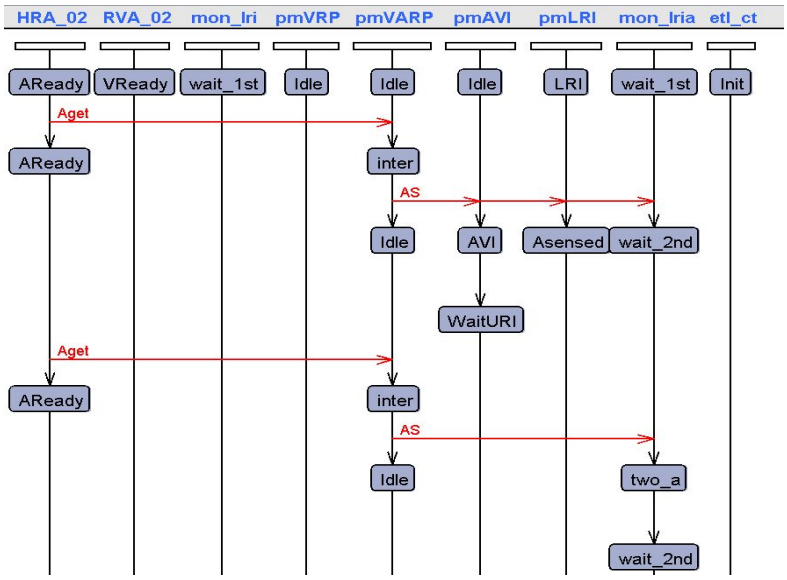
– V-A conduction delay

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.

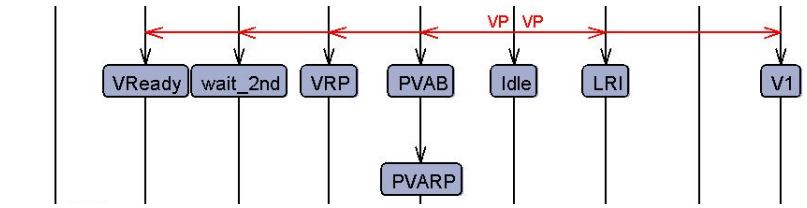
Analysis equation: E<> etl.err

Simulation of the possible wrong event:

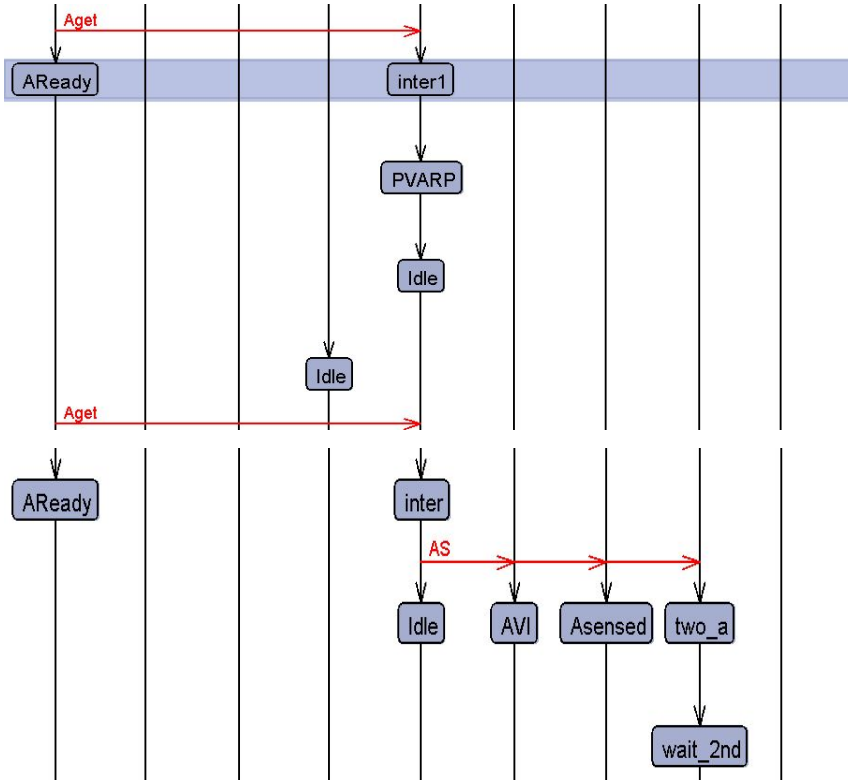
Two heart beats in 1 second.



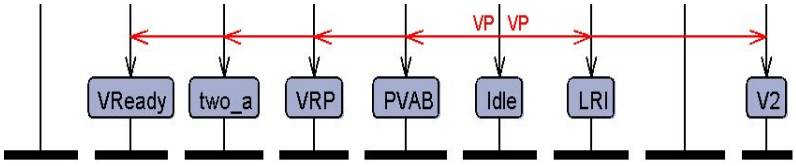
clk=400



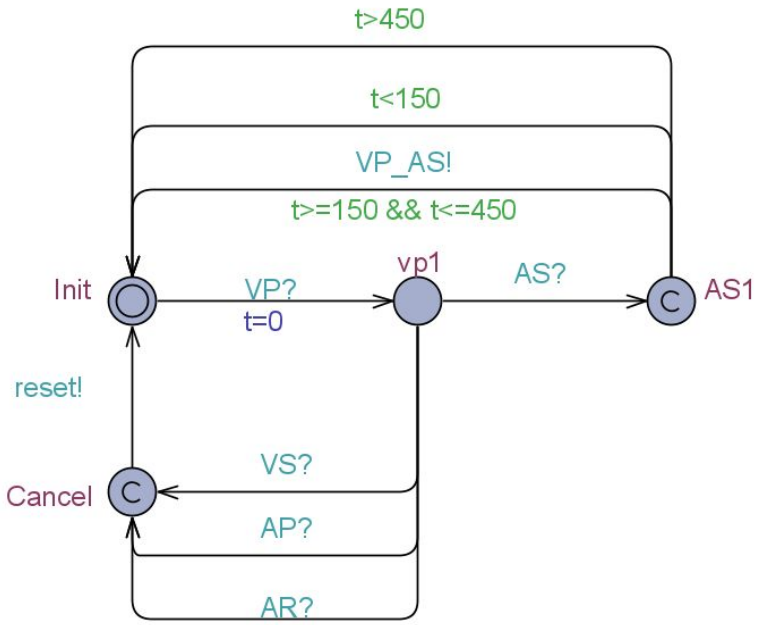
clk=600

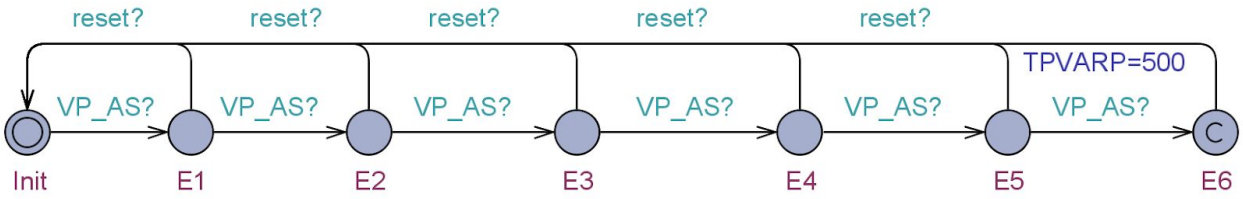


clk=800



Additional models





Analysis models