CS 233: Software Development and Validation for Medical Cyber Physical Systems

Final Report

Software Development and Validation for a DDD Pacemaker

Vera Zhang
12-20-2018

Table of Contents

System Architecture

In this project we developed the software component for a DDD pacemaker. A DDD pacemaker is a dual chamber sensing and pacing system, it has two leads connected to the right atrial and right ventricle respectively. It possesses pacing and sensing capabilities both in atrium and ventricle, which is the most commonly used pacing mode nowadays to guarantee a normal myocardial function. If the rate of atrial (resp. ventricular) contractions is low, the DDD pacemaker will deliver electrical pacing through the lead in atrium (resp. ventricle), triggering atrial/ventricular contractions to increase the atrial (resp. ventricular) rate. And if the rate is high, the pacemaker won't make the situation worse.

Fig.1: System Architecture

The system architecture is shown in Fig.1. The specifications of the signals are as follows:

- Ain: Boolean event generated by the analog circuit indicating an atrial contraction.
- Vin: Boolean event generated by the analog circuit indicating a ventricular contraction.
- AP: Boolean event generated by the DDD pacemaker software indicating an atrial pacing should be delivered.
- VP: Boolean event generated by the DDD pacemaker software indicating a ventricular pacing should be delivered.
- EGM: Voltage sensed by the lead placed in the right ventricle.
- APace: The pacing voltage delivered from the atrial lead.
- VPace: The pacing voltage delivered from the ventricular lead.

Level of Concern

Major

A DDD pacemaker is a form of AV synchronous pacing system, it can generate the electrical signal required to keep the heart beat at a healthy rate. Any faulty in pacemaker may lead to fast or slow heart rate, which can cause harm or even death to the patients using them. In other words, it is a safety-critical system.

Software Description

Sensing and pacing both atria and ventricles, a DDD pacemaker is mainly designed to treat patients with Bradycardia, during which the ventricular rate is too slow. Compared with a VVI-mode pacemaker, it can guarantee the synchrony between A-V events, and filter the noise that may affect detection of real heartbeat. The lead can sense the right atrium for the atrial sense (AS: the electrical pulse that contracts the walls of the atria) and sense the right ventricle for ventricle sense (VS: the electrical pulse that contracts the walls of the ventricle). If no AS or VS occurs within a healthy heart's time limits, the pacemaker generates electrical pulses AP or VP to contract the atrium or the ventricle, respectively.

The critical timing cycles of a DDD pacemaker is described by Barold et al. as follows.

- LRI: LRI is the longest interval between a ventricle event $v \in \{VS, VP\}$ and the subsequent atrial pacing event (AP) with no intervening sensed events.
- URI: URI limits the ventricle pacing rate by imposing a lower limit on consecutive ventricle events $v \in \{VS, VP\}$.
- VRP: VRP is initiated by a ventricle event $v \in \{VS, VP\}$. During VRP, URI and LRI cannot be initiated or reset. During this period, a pacemaker does not respond to any incoming ventricle signals.
- AVI: AVI is the time interval between an atrial event $a \in \{AS, AP\}$ and the following ventricle event.
- PVARP: PVARP is initiated by an atrial event $a \in \{AS, AP\}$ During PVARP, a pacemaker does not respond to any incoming atrial signals. In other words, during this period no atrial events can initiate a new AVI.
- AEI: AEI is the interval between a ventricle event $v \in \{VS, VP\}$ and the subsequent atrial pacing event (AP) with no intervening sensed events, AEI=LRI-AVI.

Hazard Analysis

There are 6 hazards identified from the software design:

Index	Hazard	Severity	Frequency	Mitigation	Remaining Risks
1	Slow ventricular rate	Intolerable	Frequent	Ventricular pacing	None
2	Slow atrial rate	Intolerable	Frequent	Atrial pacing	None
3	Pace on T wave	Intolerable	Probable	Ventricular sensing	None
4	Pacemaker Syndrome	Minor	Frequent	Timing Cycles monitoring	None
5	Atrial Tachycardia Response (ATR)	Minor	Probable	Mode Switch Algorithm	None
6	Endless loop tachycardia (ELT)	Minor	Probable	None	All

Hazard 6 is a kind of tachycardia resulting from the sensing of retrograde P-waves by the pacemaker and due to the limitation of the current system architecture, it cannot be addressed. However, with its 'minor' severity and probable frequency, Hazard 6 is deemed "Tolerable". Hazard 1 -5 has been sufficiently mitigated by atrial/ventricular pacing and sensing with no remaining risks. The full hazard analysis report can be found in **Appendix 1**.

Software Requirements Specification (SRS)

The goal of a DDD pacemaker is to maintain a healthy heart beat in many abnormal heart conditions. Here are 4 basic functional requirements for the software:

1. The software should not have deadlocks
2. The v-v interval should be no bigger than TLRI no matter how we set the heart rate for random heart.
3. A ventricle pace (VP) can only happen at least TURI after a ventricle event (VS VP).
4. The ventricular rate should not be faster or equal to URL for more than 30 beats.

Architecture Design Chart

The interface for the DDD pacemaker software is shown in Fig. 2. The software takes Ain and Vin events as inputs, and outputs AP and VP events. There are eight parameters which are programmable for each individual patient.

- TLRI_def, TURI_def, TAVI_def, TPVARP_def, TVRP_def: 5 constant timing circles defined as before.
- triggerRate: The maximum intervals(ms) between two 'fast' atrial events (for ATR monitoring)
- entry_count_def: The minimal limit number of two consecutive 'fast' atrial events. And once reaches it, the system will start **Duration,** and counter for Duration is begin to counting.
- duration_def: The time interval used to confirm the detection of SVT in **Duration**, actually it is the minimal limit number of ventricle events in Duration. Once reached, our pacemaker will switch to VDI-mode. And any time the entry_count reaches zero, the **Duration** will be terminated and the pacemaker should switch back to DDD mode.

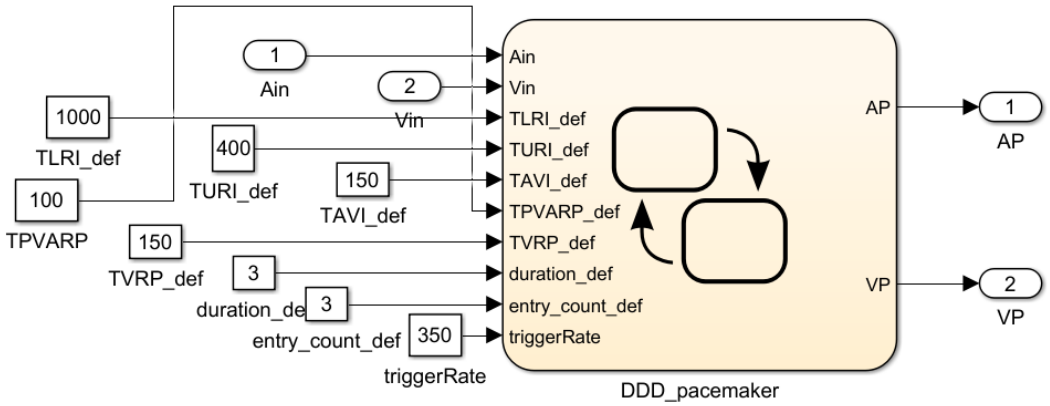

Fig 2: Interface for the DDD pacemaker software

The software design is shown in Fig. 3. There are 7 components which execute every 1ms following the Execution Order as follows.

- **VRP** component takes Vin as input and outputs VS event when the Vin event arrived TVRP after a ventricular event (VS, VP).
- **PVARP** component takes Ain as input and outputs AS (resp. AR) event when the Ain event (resp. don't) arrived TVRP after an atrial event (AS, AP).
- **URI** component monitors the time after each ventricular event (VS, VP), and set URI_block as false if the time exceeds TURI, otherwise true.
- **AVI** component monitors the time after each atrial event (AS, AP). If no ventricle events (VS, VP) are received, it will output VP event once the time exceeds TAVI and URI is not blocked.
- **LRI** component monitors the time after each ventricular event (VS, VP), and outputs VP event if the time exceeds TLRI.
- **Pre_duration_Counter** component takes AS or AR as inputs, and updates variable '*preCounter*' which denotes the number of fast consecutive 'fast' Atrial inputs (AS or AR).

- **Duration** component monitors *preCounter* and *durCounter* to start **Duration** once *preCounter* reaches entry_count_def and switch to VDI-mode once *durCounter* reaches duration_def. Finally switch back to DDD-mode once *preCounter* declines to zero.

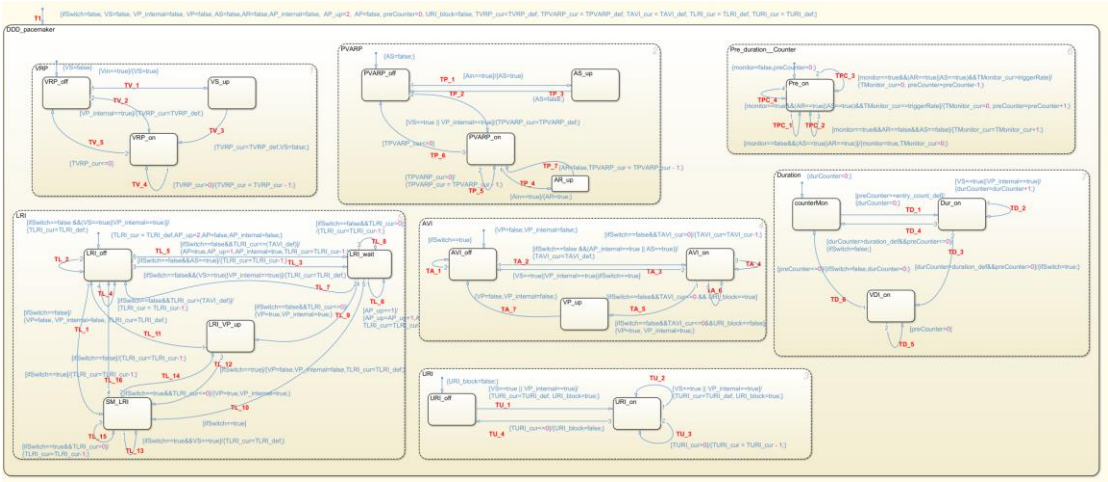

Fig 3: DDD pacemaker software design

Traceability Analysis

The DDD pacemaker software design started from a UPPAAL model of the software. The model was then manually translated into Stateflow chart and finally Matlab code. The test cases were identified from the Stateflow model based on transition coverage criteria. Details of the traceability among specifications, identified hazards and mitigations, and Verification and Validation testing can be found in **Appendix 2.**

Verification and Validation

During the software development, verification and validation activities were performed to ensure the safety and efficacy of the DDD pacemaker software. The confidence in safety and effectiveness can be illustrated using a safety assurance case. The top level of the safety assurance case is shown in Fig. 4.

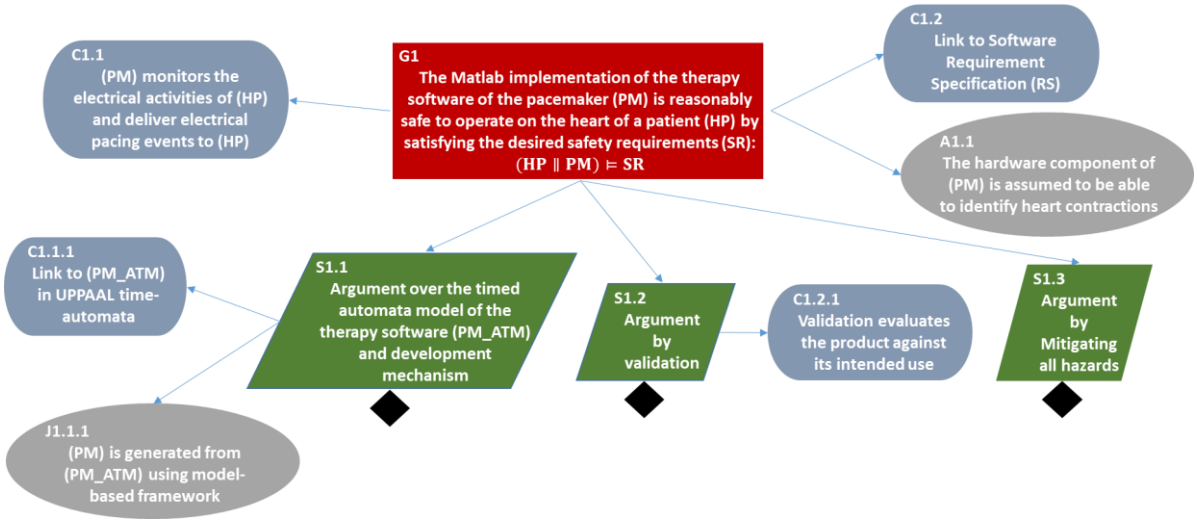

Fig. 4: Top level of safety assurance case

We have validated our DDD pacemaker software from the following 3 perspectives:

1. Model-based software development from validated model to validated code

We have adopted model-based design framework for DDD pacemaker software. A timed-automata model of the software (PM_ATM) was first developed in UPPAAL. By pairing PM_ATM with a series of heart models (HP_ATM), the closed-loop system was model checked against the two software requirements, and the requirements were satisfied. The safety argument is shown in Fig.5 and details of the model checking results can be found in **Appendix 2**.

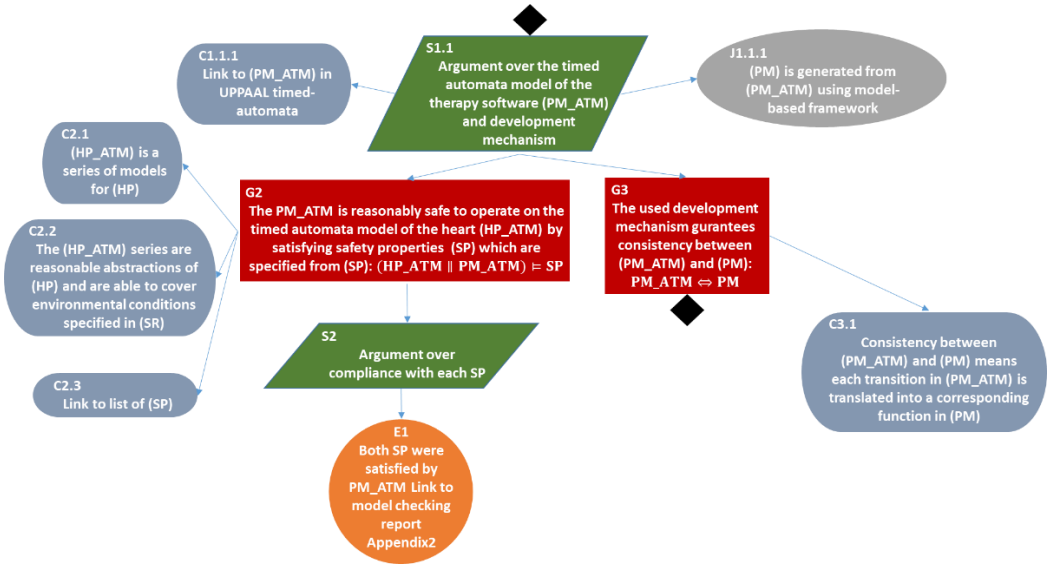

Fig. 5: Model checking in UPPAAL

PM_ATM was then manually translated into a Stateflow model PM_S, and then to Matlab code PM. The safety argument is shown in Fig. 6. The manual translation process ensures all transitions in PM_ATM have correspondence in PM. Details of traceability can be found in **Appendix 3**. The conformance between PM_S to PM was further verified using conformance testing with transition coverage criteria. Details of the conformance testing results can be found in **Appendix 4**.

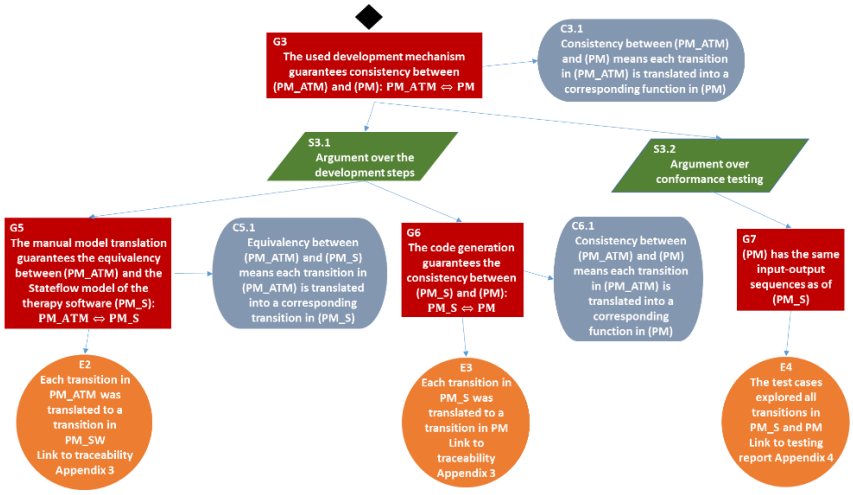

Fig. 6: From validated model to validated code

2. Functional Validation with Physiological Models

The Matlab code PM was then validated against common heart scenarios modeled by heart models HM. The test result demonstrated that PM was able to satisfy SR under those heart scenarios. The safety argument is shown in Fig. 7, and details of the functional testing can be found in **Appendix 4**.

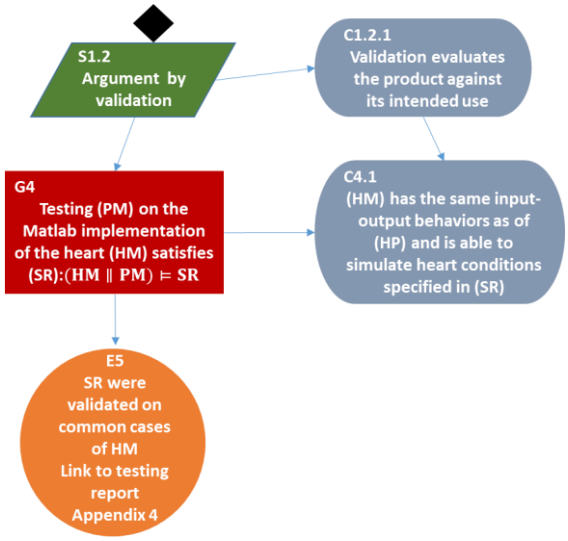

Fig. 7: Functional Testing

3. Hazards Mitigation

During the software design, all identified hazards have been sufficiently mitigated. The safety argument is shown in Fig. 8 and Fig. 9. Details of hazard analysis can be found in **Appendix 1**.

Fig. 8: Hazard Mitigation (1)

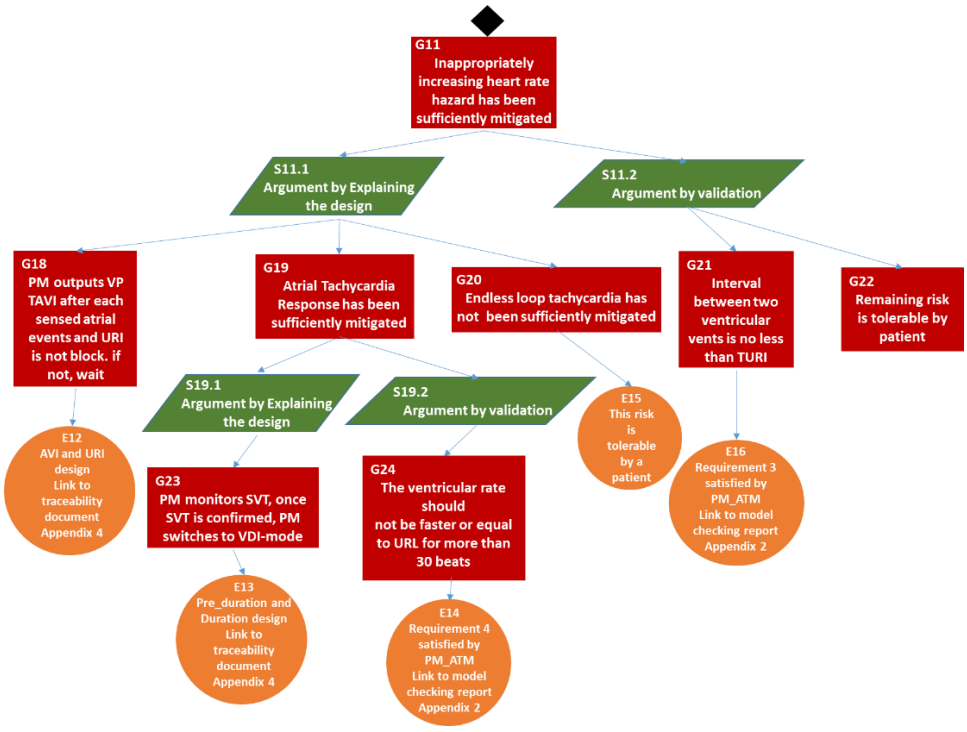

Fig. 9: Hazard Mitigation (2)

Revision History

Compared with the first version of our DDD pacemaker, the newly one has removed some redundant transitions in our UPPAAL model PM_ATM. What's more, when translating it into a Matlab code PM_S, with the help of test cases in Appendix 4, we again removed the last one extra transition in TLRI component of our DDD pacemaker safely without losing any functionality.

Remaining Problems

The remaining problem for this DDD pacemaker is ELT Hazard. According to the ELT Termination algorithm, we can confirm an ELT Hazard if the difference between the current VP-AS interval with previous intervals are within $\pm$32ms for 8 consecutive times. Then the pacemaker will increase the PVARP period to 500ms and ELT will then be terminated. However, in this solution, different AV conductions require different timing-limit. For example, if a patient has a 150ms conduction delay from A to V (or V to A), then the threshold of this termination algorithm should be set as [110ms,182ms]. However, for another patient who has a slower conduction delay with 200ms, such setting in advance will leading a false negative judgement. So a feasible solution is to set another lead to help check the flowing-direction of electrical signal, which means this new pacemaker can detect the retro-conduction from ventricle to Atrium that causes the fast ventricular rate at URL. So with this method, we don't need to set a time-limit which will be different for different conduction delay, and get a more general solution. But currently, it can't be implemented in this double-lead system design.

However, we build it in UPPAAL successfully (which means we assume another electrical signal can be sensed in AV-node by an extra lead). And we argument this solution to ELT hazard by validation. For more information, please refer to ELT part in Requirement 4 of model checking report from Appendix 2.

Appendix

Appendix 1. Hazard Analysis Report

Hazard 1: Slow ventricular rate

- Hazardous situation
  - Pacemaker fail to increase heart rate above 60bpm
- Sequence of events
  - A ventricular event happened
  - 1000ms has passed
  - No ventricular event
- Harm
  - Insufficient blood supply
- Risk Control Measure: Pace the ventricle if no ventricular event 1000ms after the previous ventricular event or Pace the ventricle if no ventricular event 150ms after the previous sensed atrial event.

Hazard 2: Slow atrial rate

- Hazardous situation
  - Pacemaker fail to increase heart rate above 60bpm
- Sequence of events
  - An atrial event happened
  - 1000ms has passed
  - No atrial event
- Harm
  - Insufficient blood supply
- Risk Control Measure: Pace the atrium if no ventricular event 850ms after the previous ventricular event

Hazard 3: Pace on T wave

- Hazardous situation
  - Pacemaker triggers ventricular fibrillation
- Sequence of events
  - The patient has a sudden increased intrinsic heart rate due to physiological need
  - A ventricular contraction occurred shortly before a ventricular pacing
  - Ventricular pacing during ventricular refractory period
  - Ventricular fibrillation triggered
- Harm
  - Insufficient blood supply
  - **Death**
- Risk Control Measure: introduce ventricular and atrial sensing and given VP when no ventricular event 1000ms after the previous ventricular event or no ventricular event 150ms after the previous sensed atrial event.

Hazard 4: Pacemaker Syndrome

- Hazardous situation
    - Atrial and ventricular contractions are too close to each other
- Sequence of events
    - DDD pacemaker delivers a ventricular pace
    - An intrinsic atrial contraction happens shortly before/after
- Harm
    - Discomfort
    - Low blood pressure
- Risk Control Measure: introduce both atrial and ventricular sensing and give a VP when no sensed ventricular events within TAVI after sensing an atrial event. Or release a ventricular pace when no sensed ventricular events nor VP within TLRI. And also give an AP when no sensed atrial event within TAEI after sensing a ventricular event.

Hazard 5: Atrial Tachycardia Response (ATR)

- Hazardous situation
    - Pacemaker triggers fast ventricular pacing due to atrial fibrillation, i.e. turn atrial flutter into ventricular flutter).
- Sequence of events
    - An atrial event happened
    - 150ms has passed
    - Pacemaker gives a ventricular pace
    - 250ms has passed
    - An atrial event happened
    - 150ms has passed
    - Pacemaker gives a ventricular pace
    - Repeat {4,5,6,7}
- Harm
    - Discomfort
    - Insufficient blood supply
- Risk Control Measure: automatic mode switching algorithm, which monitors Atrial Tachycardia and switch to VDI mode once a supraventricular tachycardia (SVT) is confirmed.

Hazard 6: Endless loop tachycardia (ELT)

- Hazardous situation
    - Random heart event trigger retrograde conduction and cause fast ventricular contraction
- Sequence of events
    - A ventricular event or a ventricular pace given by pacemaker happened
    - A premature ventricular complex (PVC) was triggered and the electrical signal became a retrograde conduction.
    - An atrial event was triggered due to the retrograde conduction caused by PVC
    - 150ms has passed

- Pacemaker gave a ventricular pace
- An atrial event was triggered again due to the retrograde conduction of VP signal
- Repeat {4,5,6}

- Harm
  - Discomfort
  - Low blood pressure
- Risk Control Measure
  - None

Appendix 2: Model checking report

The DDD pacemaker software was developed as a timed-automata model in UPPAAL as shown in Fig. 10.

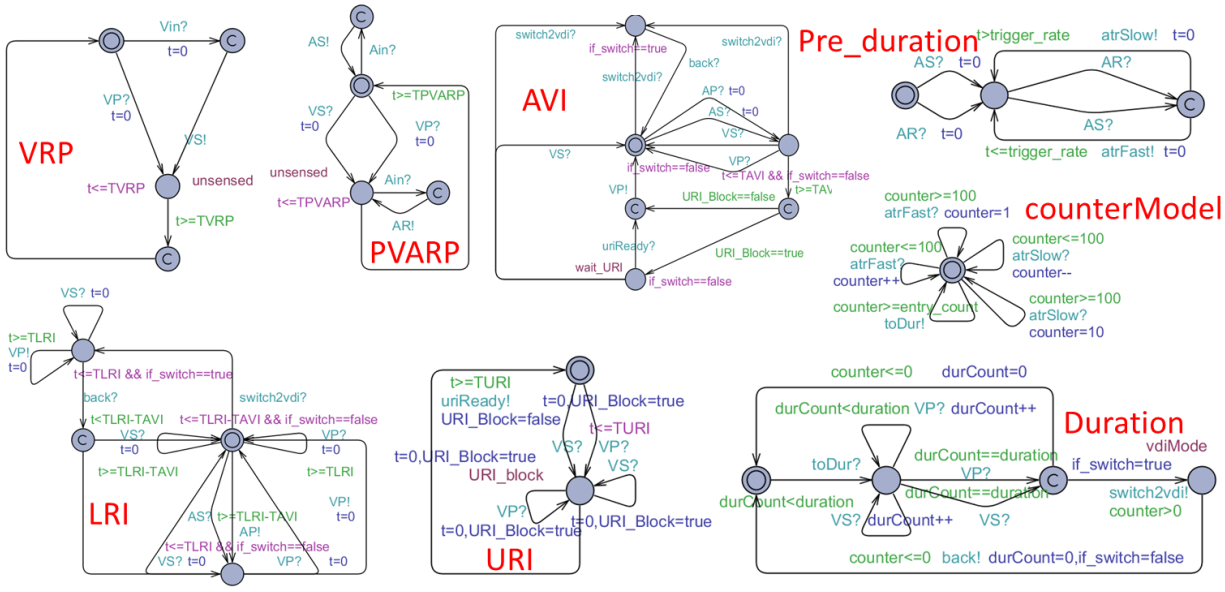

Fig. 10: UPPAAL model of the DDD pacemaker software

A heart model (Fig. 11) which can generate Ain and Vin at any time was paired with the DDD pacemaker model when [Aminwait,Amaxwait] and [Vminwait,Vmaxwait] are both set to [200,300], so that all possible inputs a pacemaker may encounter are covered.

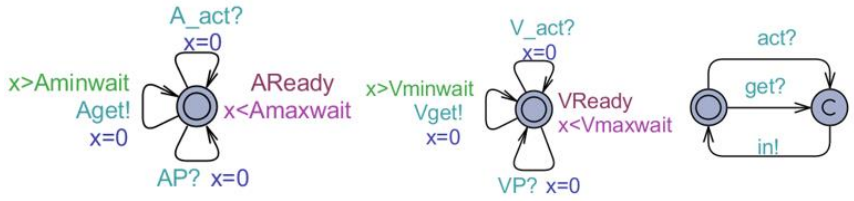

Fig. 11: UPPAAL heart model

Requirement 1 was written in TCTL as:

A[] (not deadlock)

For Requirement 2, a monitor PLRL (Fig. 12) was introduced for simpler requirement specification:

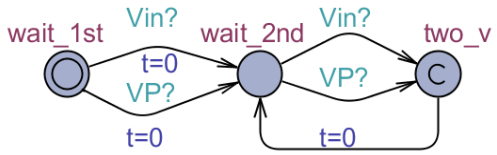

Fig. 12: Monitor for Requirement 2

Requirement 2 was written in TCTL as:

A[] (PLRL.two_v imply PLRL.t<=TLRI)

For Requirement 3, a monitor PURL (Fig. 11) was introduced for simpler requirement specification:

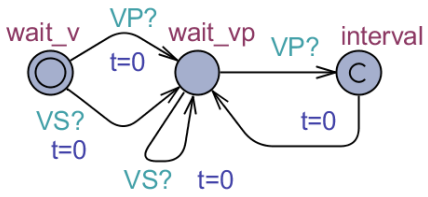

Fig. 13: Monitor for Requirement 3

Requirement 3 was written in TCTL as:

A[] (PURL.interval imply PURL.t==TURI)

All above 3 requirements were satisfied by the model, as shown in Fig. 12

```
A[] (not deadlock)
满足该性质.
A[] (PLRL.two_v imply PLRL.t<=TLRI)
满足该性质.
A[] (PURL.interval imply PURL.t>=TURI)
满足该性质.
```

Fig. 14: Model checking result (1)

For Requirement 4, we need refined the heart model for a more specific situation, such as: ATR and ELT. To achieve that, we introduce ***'cond_d'*** and ***'NAV'*** templates to model conduction path and AV node respectively. And a monitor ***'PPersist'*** was also introduced for a simpler requirement specification:

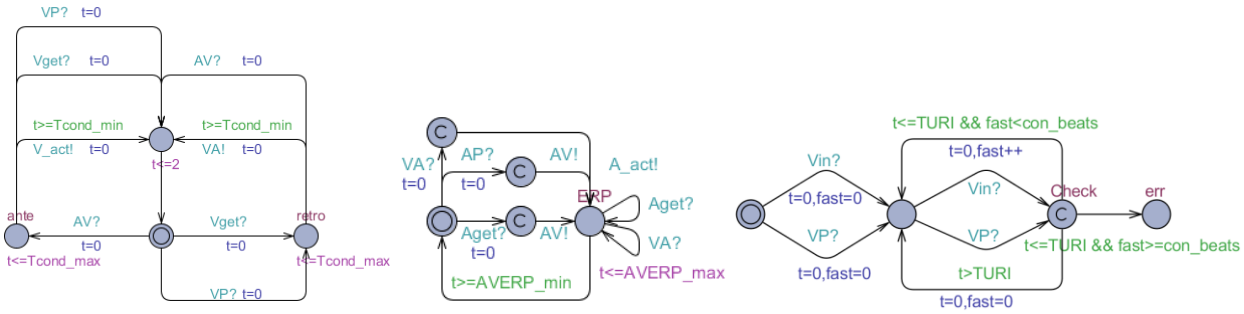

Fig. 15: Additional templates for Requirement 4 (left to right: cond_d, NAV and PPersist)

Requirement 4 was written in TCTL as:

A[] (not PPersist.err)

For **ATR** situation, we refined the heart model as Fig. 16, which implements a SVT with 300 bpm in Atria.

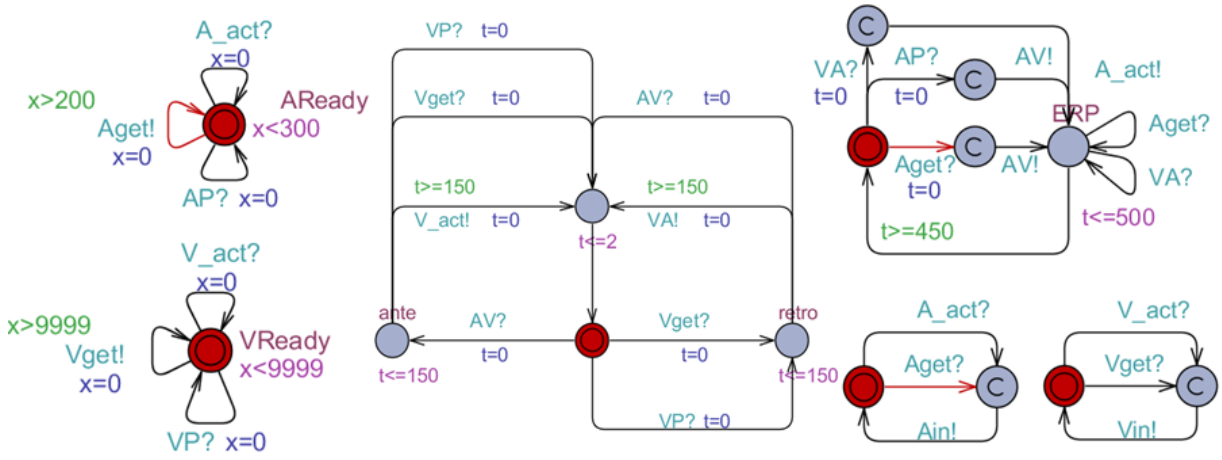

Fig. 16: A heart model in SVT

And **for ELT situation**, we add another variable '***myELTpath***' in cond_d component of our heart model (shown in Fig. 17), which indicates that we can capture the retrograde conduction of electrical signal with an extra lead.

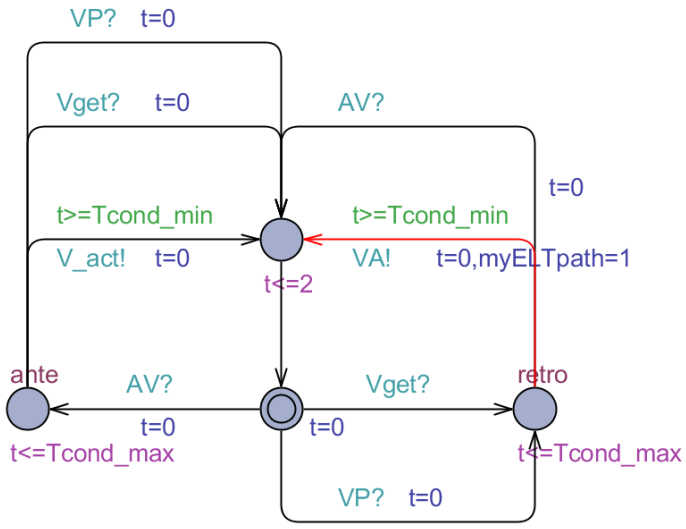

Fig. 17 Monitor variable 'myELTpath' of retrograde conduction

Then we need modify our VRP, PVARP, AVI and LRI component a little with the addition of 'myELTpath' variable to detect if the retro-grade conduction would lead a fast ventricular rate at URL. The transitions modified have been marked in red. We also add an extra component 'ELT' as an ELT monitor (Fig. 18).

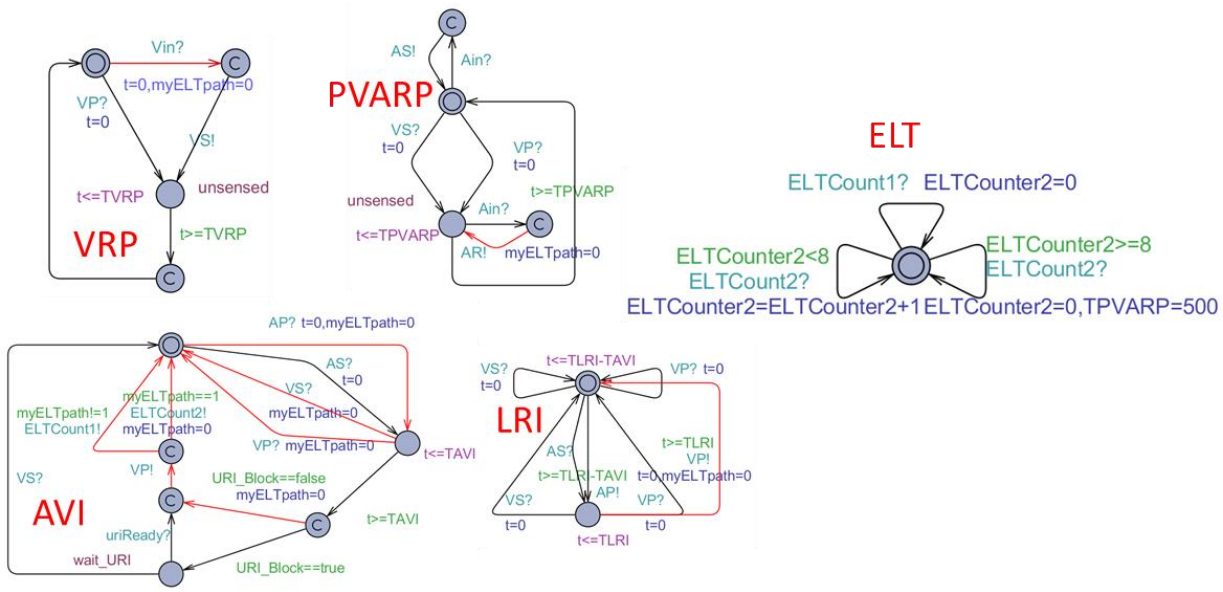

Fig. 18 New pacemaker with ELT-termination algorithm

The model checking result of requirement 4 both for ATR and ELT case is shown in Fig. 17.

```
A[] (not PPersist.err)
满足该性质.
```

Fig. 19: Model checking result (2)

Appendix 3: Traceability Document

This documents maintains traceability among the UPPAAL model, Stateflow model, Matlab code, software specification and test cases for the DDD pacemaker software. The UPPAAL model is shown in Fig. 20. Note that transition 1 in AVI of UPPAAL model will be mapped to TA_1 transition in AVI component of Stateflow model in Matlab. For others, the mapping rules are similar.

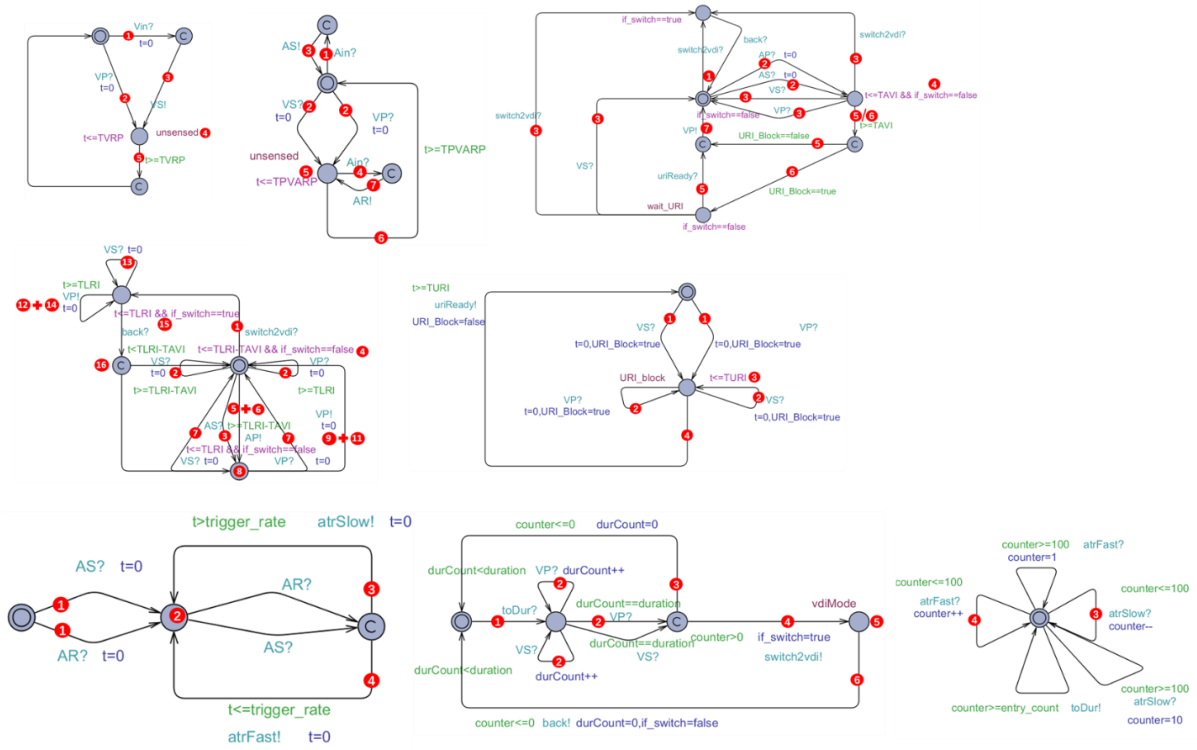

Fig. 20: UPPAAL model with notations

The Stateflow model of the DDD pacemaker software is shown in Fig. 21:

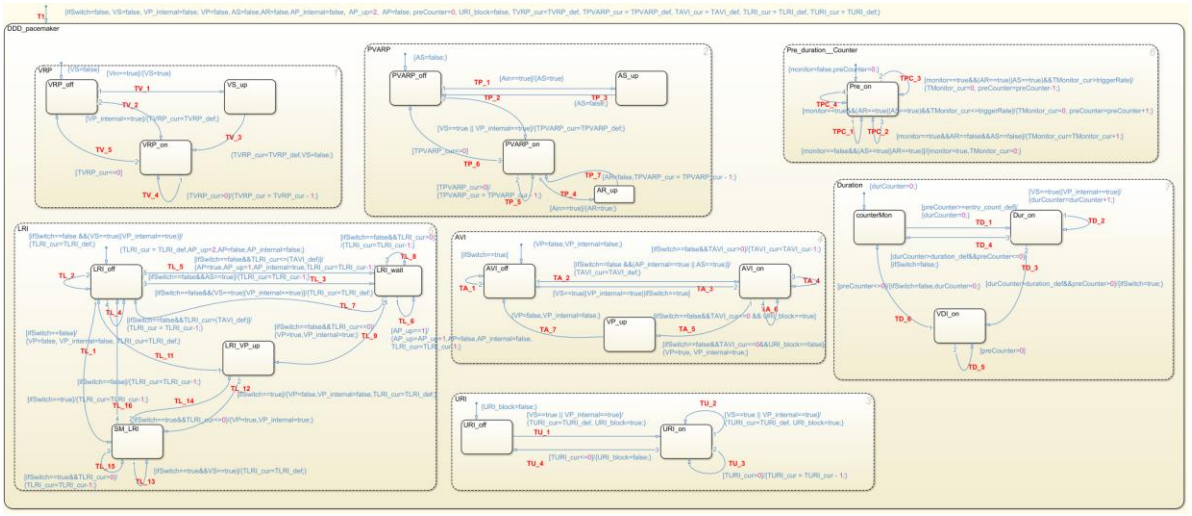

Fig. 21: Stateflow model with notations

The Matlab function of the DDD pacemaker is shown below. The function is called every msec.

```
function [States,AP,VP]=HW3_YediZhang_PM(Ain,Vin,States,Param)
 AP=States.AP_internal;
 VP=States.VP_internal;

 switch States.Cur_stateVRP
     case 'VRP_off'
         if Vin==1 %%% TV_1
```

```
            States.VS=1;
            States.Cur_stateVRP='VS_up';
        elseif States.VP_internal==1 %%% TV_2
            States.TVRP_cur=Param.TVRP_def;
            States.Cur_stateVRP='VRP_on';
        else
        end
    case 'VS_up' %%% TV_3
        States.TVRP_cur=Param.TVRP_def;
        States.VS=0;
        States.Cur_stateVRP='VRP_on';
    case 'VRP_on'
        if States.TVRP_cur>0 %%% TV_4
            States.TVRP_cur=States.TVRP_cur-1;
        elseif States.TVRP_cur<=0 %%% TV_5
            States.Cur_stateVRP='VRP_off';
        else
        end
end

switch States.Cur_statePVARP
    case 'PVARP_off'
        if Ain==1 %%% TP_1
            States.AS=1;
            States.Cur_statePVARP='AS_up';
        elseif States.VS==1 || States.VP_internal==1 %%% TP_2
            States.TPVARP_cur=Param.TPVARP_def;
            States.Cur_statePVARP='PVARP_on';
        else
        end
    case 'AS_up' %%% TP_3
        States.AS=0;
        States.Cur_statePVARP='PVARP_off';
    case 'PVARP_on'
        if Ain==1 %%% TP_4
            States.AR=1;
            States.Cur_statePVARP='AR_up';
        elseif States.TPVARP_cur>0 %%% TP_5
            States.TPVARP_cur = States.TPVARP_cur-1;
        elseif States.TPVARP_cur<=0 %%% TP_6
            States.Cur_statePVARP='PVARP_off';
        else
        end
    case 'AR_up' %%% TP_7
        States.AR=0;
        States.TPVARP_cur = States.TPVARP_cur-1;
        States.Cur_statePVARP='PVARP_on';
end

switch States.Cur_stateURI
    case 'URI_off'
        if States.VS==1 || States.VP_internal==1 %%% TU_1
            States.TURI_cur=Param.TURI_def;
            States.URI_block=1;
            States.Cur_stateURI='URI_on';
        else
        end
    case 'URI_on'
        if States.VS==1 || States.VP_internal==1  %%% TU_2
            States.TURI_cur=Param.TURI_def;
            States.URI_block=1;
        elseif States.TURI_cur>0  %%% TU_3
            States.TURI_cur=States.TURI_cur-1;
        elseif States.TURI_cur<=0  %%% TU_4
            States.URI_block=0;
            States.Cur_stateURI='URI_off';
        else
        end
end

switch States.Cur_stateAVI
```

```
    case 'AVI_off'
        if States.ifSwitch==0 && (States.AP_internal==1 || States.AS==1) %%% TA_2
            States.TAVI_cur=Param.TAVI_def;
            States.Cur_stateAVI='AVI_on';
        elseif States.ifSwitch==1 %%% TA_1
            States.Cur_stateAVI='AVI_off';
        else
        end
    case 'AVI_on'
        if States.ifSwitch==0 && States.TAVI_cur<=0 && States.URI_block==0 %%% TA_5
            VP=1;
            States.VP_internal=1;
            States.Cur_stateAVI='AVI_VP_up';
        elseif States.VS==1 || States.VP_internal==1 || States.ifSwitch==1 %%% TA_3
            States.Cur_stateAVI='AVI_off';
        elseif States.ifSwitch==0&& States.TAVI_cur>0 %%% TA_4
            States.TAVI_cur = States.TAVI_cur-1;
            States.Cur_stateAVI = 'AVI_on';
        elseif States.ifSwitch==0 && States.TAVI_cur<=0 && States.URI_block==1 %%% TA_6
            States.Cur_stateAVI='AVI_on';
        else
        end
    case 'AVI_VP_up' %%% TA_7
        VP=0;
        States.VP_internal=0;
        States.Cur_stateAVI='AVI_off';

end

switch States.Cur_stateLRI
    case 'LRI_off'
        if States.ifSwitch==1 %%% TL_1
            States.TLRI_cur = States.TLRI_cur-1;
            States.Cur_stateLRI='SM_LRI';
        elseif States.ifSwitch==0&&(States.VS==1||States.VP_internal==1) %%% TL_2
            States.TLRI_cur=Param.TLRI_def;
        elseif States.ifSwitch==0&&States.AS==1 %%% TL_3
            States.TLRI_cur = States.TLRI_cur-1;
            States.Cur_stateLRI='LRI_wait';
        elseif States.ifSwitch==0 && States.TLRI_cur>Param.TAVI_def %%% TL_4
            States.TLRI_cur=States.TLRI_cur-1;
        elseif States.ifSwitch==0 && States.TLRI_cur<=Param.TAVI_def %%% TL_5
            AP=1;
            States.AP_up=1;
            States.AP_internal=1;
            States.TLRI_cur = States.TLRI_cur-1;
            States.Cur_stateLRI='LRI_wait';
        else
        end
    case 'LRI_wait'
        if States.AP_up==1 %%% TL_6
            States.AP_up=States.AP_up+1;
            AP=0;
            States.AP_internal=0;
            States.TLRI_cur = States.TLRI_cur-1;
        elseif States.ifSwitch==0&&(States.VS==1 || States.VP_internal==1) %%% TL_7
            States.TLRI_cur=Param.TLRI_def;
            States.Cur_stateLRI='LRI_off';
        elseif States.ifSwitch==0 && States.TLRI_cur>0 %%% TL_8
            States.TLRI_cur=States.TLRI_cur-1;
        elseif States.ifSwitch==0&&States.TLRI_cur<=0 %%% TL_9
            VP=1;
            States.VP_internal=1;
            States.Cur_stateLRI='LRI_VP_up';
        else
        end
    case 'LRI_VP_up'
        if States.ifSwitch==0 %%% TL_11
            VP=0;
            States.VP_internal=0;
            States.TLRI_cur=Param.TLRI_def;
```

```
            States.Cur_stateLRI='LRI_off';
        elseif States.ifSwitch==1 %%% TL_12
            VP=0;
            States.VP_internal=0;
            States.TLRI_cur=Param.TLRI_def;
            States.Cur_stateLRI='SM_LRI';
        else
        end
    case 'SM_LRI'
        if States.ifSwitch==1&&States.VS==1 %%% TL_13
            States.TLRI_cur=Param.TLRI_def;
        elseif States.ifSwitch==1&&States.TLRI_cur<=0 %%% TL_14
            VP=1;
            States.VP_internal=1;
            States.Cur_stateLRI='LRI_VP_up';
        elseif States.ifSwitch==1&&States.TLRI_cur>0 %%% TL_15
            States.TLRI_cur=States.TLRI_cur-1;
        elseif States.ifSwitch==0 %%% TL_16
            States.TLRI_cur = States.TLRI_cur-1;
            States.Cur_stateLRI='LRI_off';
        else
        end
end

switch States.Cur_statePreDur
    case 'Pre_on'
        if States.monitor==0&&(States.AS==1||States.AR==1) %%% TPC_1
            States.monitor=1;
            States.TMon_cur=0;
        elseif States.monitor==1&&States.AR==0&&States.AS==0 %%% TPC_2
            States.TMon_cur=States.TMon_cur+1;
        elseif
States.monitor==1&&(States.AR==1||States.AS==1)&&(States.TMon_cur>Param.triggerRate) %%% TPC_3
            States.TMon_cur=0;
            States.preCounter=States.preCounter-1;
        elseif
States.monitor==1&&(States.AR==1||States.AS==1)&&(States.TMon_cur<=Param.triggerRate) %%% TPC_4
            States.TMon_cur=0;
            States.preCounter=States.preCounter+1;
        else
        end
end

switch States.Cur_stateDur
    case 'counterMon'
        if States.preCounter>=Param.entry_count_def %%% TD_1
            States.durCounter=0;
            States.Cur_stateDur='Dur_on';
        end
    case 'Dur_on'
        if States.VS==1||States.VP_internal==1 %%% TD_2
            States.durCounter=States.durCounter+1;
        elseif States.durCounter>Param.duration_def && States.preCounter>0 %%% TD_3
            States.ifSwitch=1;
            States.Cur_stateDur='VDI_on';
        elseif States.durCounter>Param.duration_def && States.preCounter<=0 %%% TD_4
            States.ifSwitch=0;
            States.Cur_stateDur='counterMon';
        else
        end
    case 'VDI_on'
        if States.preCounter>0 %%% TD_5
            States.Cur_stateDur='VDI_on';
        elseif States.preCounter<=0 %%% TD_6
            States.ifSwitch=0;
            States.durCounter=0;
            States.Cur_stateDur='counterMon';
        else
        end
  end
end
```

The Fig. 22 shows the traceability for each state and transition among the models.

Index	In matlab	In Simulink	test case(part)	Correspondence in UPPA	Specification
				Parameters	
1	Param.TLRI_def	TLRI_def		TLRI	The maximum interval between 2 ventricular events
2	Param.TAVI_def	TAVI_def		TAVI	The minimum delay between a ventricular event and an atrial event
3	Param.TPVARP_def	TPVARP_def		TPVARP	The period after each atrial event in which no Ain is accepted
4	Param.TVRP_def	TVRP_def		TVRP	The period after each ventricular event in which no Vin is accepted
5	Param.TURI_def	TURI_def		TURI	The minimal interval between 2 ventricular events
6	Param.URI_block	URI_block		URI_Block	The variable to record the state of URI (if it is in block state)
7	Param.duration_def	duration_def		duration	A monitor counter to start Duration from Pre-duration
8	Param.entry_count_def	entry_count_def		entry_count	A counter measures fast-events in Pre-duration
9	Param.triggerRate	triggerRate		triggerRate	sensing threshold for SVT Detector(ATR)
				Local Varibales/Signals	
10	States.ifSwitch	ifSwitch		Switch!	Switch to VDI Pacemaker
11	States.VS	VS		VS	Internal event indicating sensed ventricular event
12	States.VP_internal	VP_internal		all VP	Internal event indicating VP is true
13	States.AS	AS		PVARP-3	Internal event indicating sensed atrial event
14	States.AR	AR		PVARP-7	Internal event indicating unsensed atrial event
15	States.AP_internal	AP_internal		LRI-5,6	Internal event indicating AP is true
16	States.AP_up	AP_up			additional variable used in simulink&matlab for track in LRI: To classify If there exists a AS when LRI is in the initial state or tigger a VP
17	States.URI_block	URI_block		URI_Block	A varibale monitoring if URI is in block state, avi need to wait until URI unlocked to send a VP
18	States.TVRP_cur	TVRP_cur		VRP-t	clock in VRP
19	States.TPVARP_cur	TPVARP_cur		PVARP-t	clock in PVARP
20	States.TAVI_cur	TAVI_cur		AVI-t	clock in AVI
21	States.TLRI_cur	TLRI_cur		LRI-t	clock in LRI
22	States.TURI_cur	TURI_cur		URI-t	clock in URI
23	States.TMon_cur	TMon_cur		CounterModel-t	clock in CounterModel
24	States.monitor	monitor		Pre_duration-2	Start monitors on intervals between two atrial events
25	States.preCounter	preCounter		CounterModel-counter	A counter to record number of fast-interval
26	States.durCounter	durCounter		Duration-durCount	Ventricular-event-timer for duration
				Initialization	
0	T1	T1		Initial State	Initial internal events, output events and timers
				VRP	
27	TV_1	TV_1	6,12,61,65,69,73,79	1	receive a sensed Vin, sent VS to Pacemaker, and get into VRP
28	TV_3	TV_3	7,13,62,66,70,74,80	3	
29	TV_2	TV_2	21,27,34,41,48,87	2	receive VP and get into VRP-period
30	TV_4	TV_4	8,9,10,14,21,28,35,4	4	VRP-period
31	TV_5	TV_5	11,15,22,29,36,43,60	5	get out of VRP-period
				PVARP	
32	TP_1	TP_1	2,4,16,24,31,38,83	1	receive a sensed Ain, sent AS to Pacemaker
34	TP_3	TP_3	3,5,17,25,32,39,84	3	
33	TP_2	TP_2	6,12,20,27,34,41,48,(	2	receive Ventricular event and get into PVARP-period
36	TP_5	TP_5	7,13,62,66,70,74,90	5	PVARP-period
35	TP_4	TP_4	8,49,51,53,55,57,75,(	4	receive a unsensed Ain, sent AR to Pacemaker
38	TP_7	TP_7	9,50,54,56,68,76,89	7	
37	TP_6	TP_6	10,14,21,28,35,42,59	6	get out of PVARP-period
				Pre_duration	
39	TPC_1	TPC_1	2	1	get into pre-duration monitor
40	TPC_2	TPC_2	3,4,6,7,10,11-15,17-	2	
41	TPC_3	TPC_3	24,31,38,75,83,88	3	a Fast-event
42	TPC_4	TPC_4	5,8,9,16,49,51,53,55,	4	a slow-event
				Duration	
43	TD_1	TD_1	16,57	1	Enter Duration in SVT-algorithm
44	TD_2	TD_2	19,26,33,61,65,69	2	Duartion terminate
45	TD_3	TD_3	73	3	after duration, counter value is negative, still remain in DDD-mode
46	TD_4	TD_4	40	4	after duration, counter value is positive, switch to VDI Pacemaker
47	TD_5	TD_5	74-88	5	stay in VDI-mode
48	TD_6	TD_6	89	6	switch back to DDD-mode
				AVI	
49	TA_1	TA_1	74-89	1	Switch to VDI Pacemaker, and switch back, since AVI shouldn't work in VDI-mode
50	TA_2	TA_2	2,16,24,31,38,46	2	Recevie an sensed atrial event, begin AVI
51	TA_3	TA_3	6	3	receive Ventricular event and restart
52	TA_4	TA_4	3,4,5,17,25,32,39,47	4	wait AVI-period ends
53	TA_5	TA_5	19,26,33,40	5	pacemaker is not in URI, a VP is triggered
54	TA_6	TA_6	18	6	pacemaker is in URI(URI is blocked), a VP won't be triggered until URI is unblocked
55	TA_7	TA_7	20,27,34,41,48	7	An interal transition in uppaal
				LRI	
56	TL_1	TL_1	74	1	Switch to VDI Pacemaker, and the timer is kept
57	TL_2	TL_2	12,61,65,69,73	2	Receive a Ventricular event, restart
58	TL_3	TL_3	2,16,24,31,38	3	receive an AS, then won't sent AP after AEI, wait to send VP directly
59	TL_4	TL_4	1,7-11,13-15,20-23	4	not receive an AS, wait to sent AP
60	TL_5	TL_5	45	5	no AS, no VP/VS, then send AP after AEI
61	TL_6	TL_6	46	6	
62	TL_7	TL_7	6,19,26,33,40	7	Receive a Ventricular event, restart
63	TL_8	TL_8	3,4	8	not receive any Ventricular event, wait to sent VP after LRI
64	TL_9	TL_9	47	9	not receive any Ventricular event, wait to sent VP after LRI
66	TL_11	TL_11	48	11	
67	TL_12	TL_12	86	12	sent VP after LRI
69	TL_14	TL_14	87	14	
68	TL_13	TL_13	79,80	13	Receive a Ventricular event, restart
70	TL_15	TL_15	75-78,81-85,88,89	15	not receive any Ventricular event, wait to sent VP after LRI
71	TL_16	TL_16	90	16	switch back to DDD-mode
				URI	
72	TU_1	TU_1	6,20,27,34,41,48,87	1	Receive a Ventricular event, restart URI
73	TU_2	TU_2	12,61,65,69,73,79	2	
74	TU_3	TU_3	7-11,13-18,21,22,28	3	stay in URI-period
75	TU_4	TU_4	19,23,30,37,44,85	4	get out of URI-period

Fig. 22 The traceability for each state and transition in PM_S

Appendix 4: Test Report

1. Functional Testing

Heart model HM was used to test both PM_S and PM using common heart scenarios. The tests cover all 4 interaction patterns shown below, while satisfying Requirement 2. The test results are shown in Fig. 21.

a) Ain, Vin

b) Ain, VP
c) AP, Vin
d) AP, VP
e) Mode switch

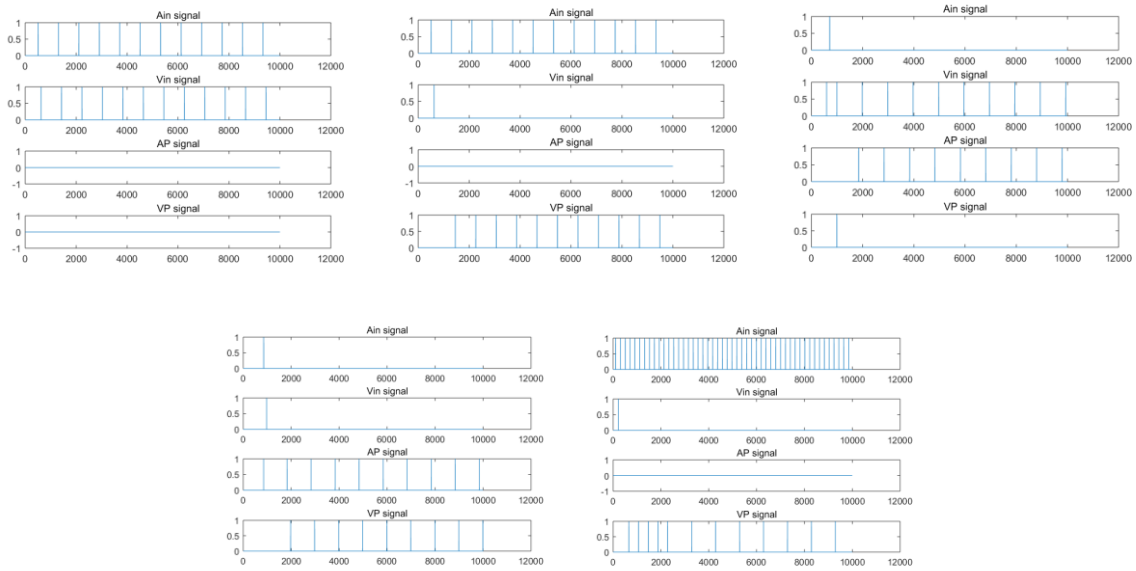

Fig. 23 Functional testing results

2. Conformance Testing

The goal of conformance testing is to establish a conformance relationship between PM_S and PM. The test cases will be generated from the Stateflow model, and the combined states which can be reached (31 reachable states) are listed as shown in Fig. 23. Note that unreachable states (855) are not list here.

VRP_off & PVARP_off & AVI_off & LRI_off & URI_off & Pre_on & counterMon	S1
VRP_off & PVARP_off & AVI_off & LRI_off & URI_off & Pre_on & Dur_on	S2
VRP_off & PVARP_off & AVI_off & LRI_off & URI_on & Pre_on & counterMon	S4
VRP_off & PVARP_off & AVI_off & LRI_off & URI_on & Pre_on & Dur_on	S5
VRP_off & PVARP_off & AVI_off & LRI_wait & URI_off & Pre_on & counterMon	S7
VRP_off & PVARP_off & AVI_off & LRI_VP_up & URI_off & Pre_on & VDI_on	S15
VRP_off & PVARP_off & AVI_off & SM_LRI & URI_off & Pre_on & VDI_on	S21
VRP_off & PVARP_off & AVI_off & SM_LRI & URI_on & Pre_on & VDI_on	S24
VRP_off & PVARP_off & AVI_on & LRI_wait & URI_off & Pre_on & counterMon	S31
VRP_off & PVARP_off & AVI_on & LRI_wait & URI_off & Pre_on & Dur_on	S32
VRP_off & PVARP_off & AVI_on & LRI_wait & URI_on & Pre_on & Dur_on	S35
VRP_off & PVARP_off & AVI_on & LRI_VP_up & URI_off & Pre_on & counterMon	S37
VRP_off & PVARP_off & AVI_VP_up & LRI_off & URI_off & Pre_on & counterMon	S49
VRP_off & PVARP_off & AVI_VP_up & LRI_off & URI_off & Pre_on & Dur_on	S50
VRP_off & AS_up & AVI_off & SM_LRI & URI_on & Pre_on & VDI_on	S96
VRP_off & AS_up & AVI_on & LRI_wait & URI_off & Pre_on & counterMon	S103
VRP_off & AS_up & AVI_on & LRI_wait & URI_off & Pre_on & Dur_on	S104
VRP_off & AS_up & AVI_on & LRI_wait & URI_on & Pre_on & Dur_on	S107
VS_up & PVARP_on & AVI_off & LRI_off & URI_on & Pre_on & counterMon	S436
VS_up & PVARP_on & AVI_off & LRI_off & URI_on & Pre_on & Dur_on	S437
VS_up & PVARP_on & AVI_off & LRI_off & URI_on & Pre_on & VDI_on	S438
VS_up & PVARP_on & AVI_off & SM_LRI & URI_on & Pre_on & VDI_on	S456
VRP_on & PVARP_off & AVI_off & LRI_off & URI_on & Pre_on & counterMon	S580
VRP_on & PVARP_off & AVI_off & LRI_off & URI_on & Pre_on & Dur_on	S581
VRP_on & PVARP_off & AVI_off & SM_LRI & URI_on & Pre_on & VDI_on	S600
VRP_on & PVARP_on & AVI_off & LRI_off & URI_on & Pre_on & counterMon	S724
VRP_on & PVARP_on & AVI_off & LRI_off & URI_on & Pre_on & Dur_on	S725
VRP_on & PVARP_on & AVI_off & SM_LRI & URI_on & Pre_on & counterMon	S742
VRP_on & PVARP_on & AVI_off & SM_LRI & URI_on & Pre_on & VDI_on	S744
VRP_on & AR_up & AVI_off & LRI_off & URI_on & Pre_on & counterMon	S796
VRP_on & AR_up & AVI_off & LRI_off & URI_on & Pre_on & Dur_on	S797
VRP_on & AR_up & AVI_off & SM_LRI & URI_on & Pre_on & VDI_on	S816

Fig. 24: Combined states

Test cases shown in Fig. 25 were generated to cover all transitions (100%) in Fig. 22. There are totally 90 test-cases used in this test procedure, and Ain (resp. Vin) signal has been marked in red (resp. orange). All output signals are in blue. It contains input, state transition covered and output in each test case.

		P0		P1				P2								P3/D0
Test-Case -- Mode Switch (3,3)																
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
time(ms)	49	50	51	150	151	169	170	220	221	270	320	330	331	441	480	500
start state	S1	S1	S103	S31	S103	S31	S436	S724	S796	S724	S580	S4	S436	S724	S580	S4
Input	49ms	Ain	1ms	Ain	1ms	Vin	1ms	Ain	1ms	49ms	50ms	Vin	1ms	100ms	50ms	Ain
VRP						1	3	4	4	4	5	1	3	4	5	
PVARP		1	3	1	3	2	5	4	7	6		2	5	6		1
AVI		2	4	4	4	3										2
LRI	4	3	8	8	8	7	4	4	4	4	4	2	4	4	4	3
URI						1	3	3	3	3	3	2	3	3	3	3
Pre		1	2	2	4	2	2	4	4	2	2	2	2	2	2	4
Dur																1
output		AS		AS		VS		AR				VS				AS
finish state	S1	S103	S31	S103	S31	S436	S724	S796	S724	S580	S4	S436	S724	S580	S4	S107

	P3/D0			D1					P2		D2					P1
Test-Case -- Mode Switch (3,3)																
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
time(ms)	500	501	651	731	732	832	882	1132	1210	1211	1361	1362	1462	1512	1762	1860
start state	S4	S107	S35	S35	S50	S725	S581	S5	S2	S104	S32	S50	S725	S581	S5	S2
Input	Ain	1ms	150ms	80ms	1ms	100ms	50ms	250ms	Ain	1ms	150ms	1ms	100ms	50ms	250ms	Ain
VRP					2	4	5					2	4	5		
PVARP	1	3			2	6			1	3		2	6			1
AVI	2	4	6	5	7				2	4	5	7				2
LRI	3	8	8	7	4	4	4	4	3	8	7	4	4	4	4	3
URI	3	3	3	4	1	3	3	4				1	3	3	4	
Pre	4	2	2	2	2	2	2	2	3	2	2	2	2	2	2	3
Dur	1			2							2					
output	AS			VP					AS		VP					AS
finish state	S107	S35	S35	S50	S725	S581	S5	S2	S104	S32	S50	S725	S581	S5	S2	S104

			D3					P0		D4						
Test-Case -- Mode Switch (3,3)																
	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46
time(ms)	1860	1861	2011	2012	2112	2162	2412	2420	2421	2571	2572	2672	2722	2972	3422	3423
start state	S2	S104	S32	S50	S725	S581	S5	S2	S104	S32	S49	S724	S580	S4	S1	S7
Input	Ain	1ms	150ms	1ms	100ms	50ms	250ms	Ain	1ms	150ms	1ms	100ms	50ms	250ms	450ms	1ms
VRP				2	4	5					2	4	5			
PVARP	1	3		2	6			1	3		2	6				
AVI	2	4	5	7				2	4	5	7					2
LRI	3	8	7	4	4	4	4	3	8	7	4	4	4	4	5	6
URI				1	3	3	4				1	3	3	4		
Pre	3	2	2	2	2	2	2	3	2	2	2	2	2	2	2	2
Dur			2							4						
output	AS		VP					AS		VP					AP	
finish state	S104	S32	S50	S725	S581	S5	S2	S104	S32	S49	S724	S580	S4	S1	S7	S31

				P-1		P0		P1		P2		P3/D0				D1
Test-Case -- Mode Switch (3,3)																
	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61
time(ms)	3423	3572	3574	3600	3601	3610	3611	3620	3621	3630	3631	3640	3641	3674	3724	3730
start state	S7	S31	S37	S724	S796	S724	S796	S724	S796	S724	S796	S724	S797	S725	S581	S5
Input	1ms	149ms	1ms	Ain	1ms	Ain	1ms	Ain	1ms	Ain	1ms	Ain	1ms	33ms	50ms	Vin
VRP			2	4	4	4	4	4	4	4	4	4	4	4	5	1
PVARP			2	4	7	4	7	4	7	4	7	4	7	6		2
AVI	2	4	7													
LRI	6	9	11	4	4	4	4	4	4	4	4	4	4	4	4	2
URI			1	3	3	3	3	3	3	3	3	3	3	3	3	2
Pre	2	2	2	4	2	4	2	4	2	4	2	4	2	2	2	2
Dur												1				2
output		VP		AR		AR		AR		AR		AR				VS
finish state	S31	S37	S724	S796	S724	S796	S724	S796	S724	S796	S724	S797	S725	S581	S5	S437

	D1				D2				D3				D4		P2	
Test-Case -- Mode Switch (3,3)																
	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76
time(ms)	3730	3731	3831	3881	3900	3901	4001	4051	4100	4101	4201	4251	4300	4301	4350	4351
start state	S5	S437	S725	S581	S5	S437	S725	S581	S5	S437	S725	S581	S5	S438	S744	S816
Input	Vin	1ms	100ms	50ms	Vin	1ms	100ms	50ms	Vin	1ms	100ms	50ms	Vin	1ms	Ain	1ms
VRP	1	3	4	5	1	3	4	5	1	3	4	5	1	3	4	4
PVARP	2	5	6		2	5	6		2	5	6		2	5	4	7
AVI														1	1	1
LRI	2	4	4	4	2	4	4	4	2	4	4	4	2	1	15	15
URI	2	3	3	3	2	3	3	3	2	3	3	3	2	3	3	3
Pre	2	2	2	2	2	2	2	2	2	2	2	2	2	2	3	2
Dur	2				2				2				3	5	5	5
output	VS				VS				VS				VS		AR	
finish state	S437	S725	S581	S5	S437	S725	S581	S5	S437	S725	S581	S5	S438	S744	S816	S744

								P1					P0			
Test-Case -- Mode Switch (3,3)																
	76	77	78	79	80	81	82	83	84	85	86	87	88	89	90	
time(ms)	4351	4401	4451	4500	4501	4601	4651	4800	4801	4901	5501	5502	5550	5551	5552	
start state	S816	S744	S600	S24	S456	S744	S600	S24	S96	S24	S21	S15	S744	S816	S742	
Input	1ms	50ms	50ms	Vin	1ms	100ms	50ms	Ain	1ms	100ms	600ms	1ms	Ain	1ms	1ms	
VRP	4	4	5	1	3	4	5					2	4	4	4	
PVARP	7	6		2	2	6		1	3			2	4	7	5	
AVI	1	1	1	1	1	1	1	1	1	1	1	1	1	1		
LRI	15	15	15	13	13	15	15	15	15	15	12	14	15	15	16	
URI	3	3	3	2	3	3	3	3	3	4		1	3	3	3	
Pre	2	2	2	2	2	2	2	3	2	2	2	2	3	2	2	
Dur	5	5	5	5	5	5	5	5	5	5	5	5	5	6		
output				VS				AS			VP		AR			
finish state	S744	S600	S24	S456	S744	S600	S24	S96	S24	S21	S15	S744	S816	S742	S724	

Fig. 25: Test cases

The test cases were executed on both the Stateflow model and the Matlab code, and both tests passed (shown in Fig. 26).

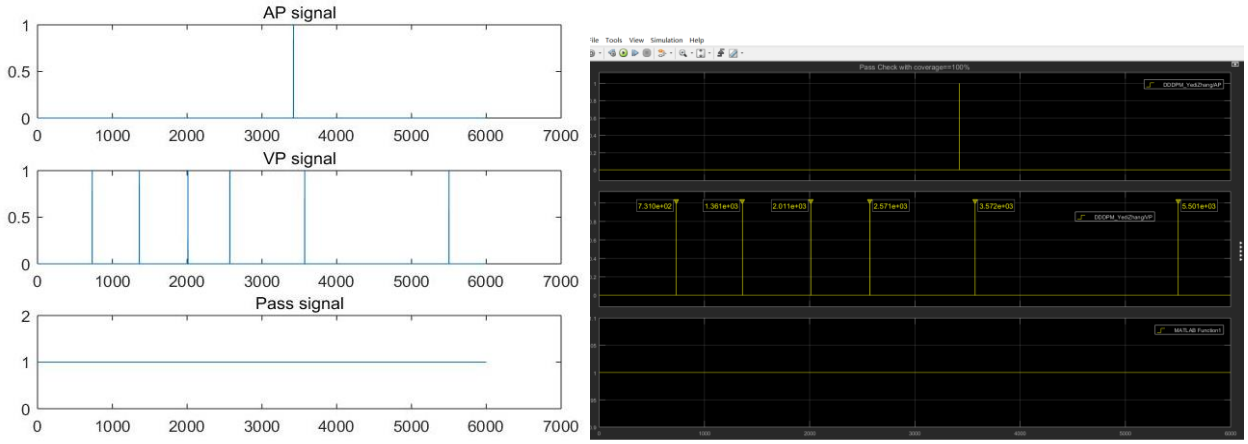

Fig. 26 Test cases running results

The visualization of the test cases is shown in Fig. 27.

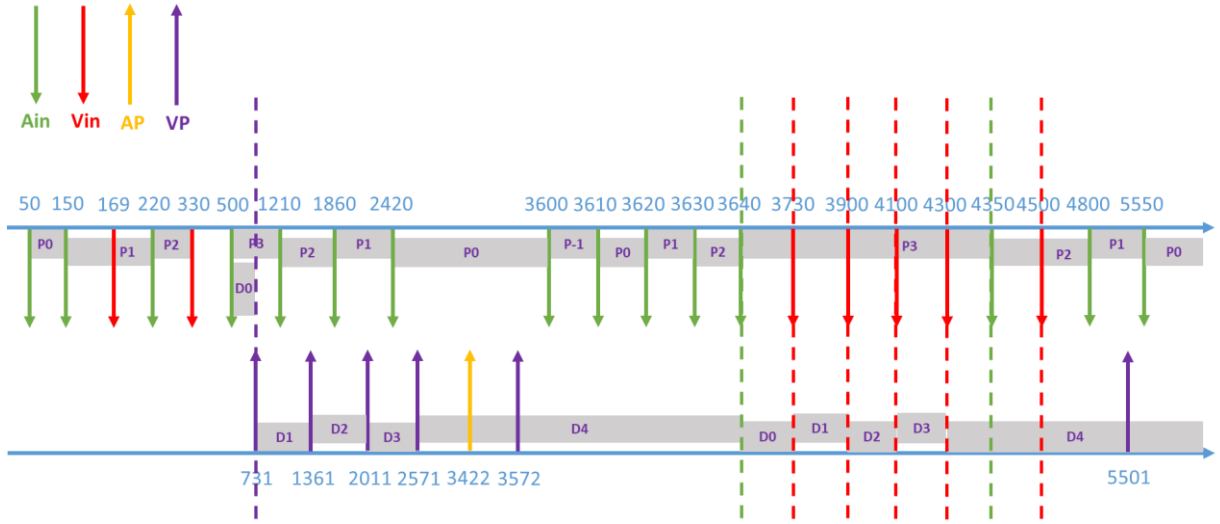

Fig. 27: Test case visualization

For those gray regions:

P1 denotes “preCounter==1”

P-1 denotes “preCounter==-1”

D1 denotes “durCounter==1”.