COURSE-EEC011P5G

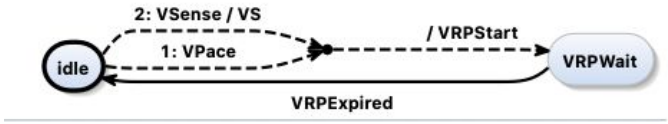
PACEMAKER REPORT

Design Logic and Road-Map

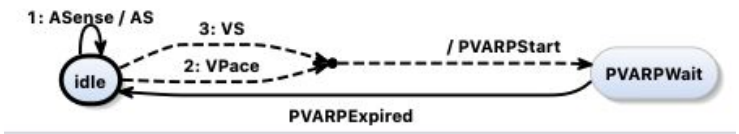
Introduction and Checking Validity of Pulses (3 hrs)

The pacemaker design starts with the implementation of 5 timers interacting with each other thus we here tried implementing the design as a system 5 timers i.e AVI Timer, AEI Timer, PVARP Timer, VRP timer, URI Timer, and LRI Timer. We started first implementing the design as a rough blueprint which helped us to found some logics to get correct signals to start the timers AEI, AVI, PVARP, URI, and LRI which restarted on getting a valid V Pulse here a valid V Pulse is a pulse which does not occur again after VRP_Expire seconds after a **V Pulse**. (Where VRP_Expire is the duration of VRP Timer).**[RoadBlock]**

To counter this we introduced a new signal in our design- **VS**, which communicates with other timers to denote an onset of a valid pulse. This can be shown in the following figure-



A similar technique was thus then used to generate a dummy signal **AS** to denote the onset of a valid **A Pulse**. A valid A-Pulse is a pulse generated after PVARP_Exp seconds after the onset of a valid V-Pulse thus we emitted the dummy signal **AS** when the PVARP Timer state is not in a waiting state (or is still ticking). Implementation can be understood using the following SCC.



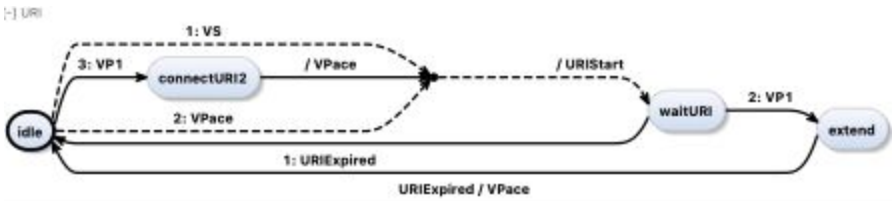
Taking care to check the validity of pulses we can start the corresponding timers on the being triggered by the VS and AS Pulses, generating their corresponding start signals. This will led all timers to wait until a corresponding Expired Signal is not received.

Adding the Pacing Logic (1 hr)

Now we try to inculcate the pacing logic into our SCC, the V-Pacing occurs in two cases -

1) AVI Timer expires after an A-Pulse and no V-Pulse is observed.
2) LRI Timer expires after a V-Pulse and another V-Pulse is yet not observed.

We address this issue by outputting a V-Pace when the corresponding timers expire, but here we end up at another **road-block,** the inclusion of a minimum URI time between the pacing which we need to address. Thus directly pacing on AVI-Timer expiration will not work. To tackle this issue we introduce an intermediate state **extension state** and an intermediate signal **VP1** in the URI timer SCC. This state transition is activated on receiving a VP1 signal and stays in the state until the URI timer expires, thus outputting the **V-Pace** signal. The following diagram can show a better picture of the representation.



Other case we need to take care was of that LRI_expiration time < AVI_expiration time + AEI_Expiration time. Thus we need to pace the heart even if the LRI-timer expires before thus pacing is required on both the instances and all the timers needed to be resetted when this pacing is outputted.

Dealing with not ASC-schedulable issue (1 hr)

A issue which occurred after the implementation of SCC was resulting SCC was not ASC-schedulable which on debugging and dry running our dummy logic occurred due to uncertainty on state transition. Also, some issues occurred due to immediate transition

cycle which was leding the system to run into an infinite loop. After the removal of these errors we were finally able to generate the C-code for our SCC chart.

Final Step: Implementation on PSOC (1-1.5 hr)

The code function was finally added to the psoc workspace along with a header files to communicate our psoc with the **tick, reset** functions and the Input and Output Variables.

[**roadblock**] We encountered a problem with the limit of the available number of fixed timers in the PSOC, (our design needed the 6 timers while available fixed timers were only 4). This was later solved by using the UDP implementation of timers provided by the PSOC. Each timer was set to have a period of 1ms, with interrupts on TC so that we can update each of the counter variables to match their corresponding timings.

The USBUART read and write functions were implemented to communicate with the heart emulator to read 'A' ,V' as the corresponding pulses and to pace the heart with the same symbols when necessary. The other functions implemented were the corresponding timer functions to communicate the timer with the SCC chart logic. The function allowed to start the timer, enable their interputs on receiving a start signal. Also, stop and reset the timer on receiving an expiration signal.

All these functions were encapsulated in a pacemaker function starting with USB read, timers and finally USB write to achieve correct pacing logic of our pacemaker.

Testing and fixing (2 hr)

On successful compilation of the code we ran our model on the heart emulator provided on various ticks and changing the timings of the model. Some errors occurred like double pacing on LRI expiration which was found due to no reset on A-pace in some timers which was leding to pacing again. On constant testing and fixing what we believe we finally get a accurate pacemaker design.