DESIGN REPORT DDD-MODE PACEMAKER

Daksh Chauhan
2018UCS0068

1. Project Overview

This project is an implementation of the DDD Mode PaceMaker. The steps taken while designing the PaceMaker:

1. SCChart implementation of the project which consists of 6 Timing Constraints to be maintained between Ventricular and Atrial Events.
2. Using the code generated in [1], integrate it with the PSoc by mapping the I/O with Timers and ISRs. Also linking with the Serial Connection for testing purposes.
3. Test the pacemaker via the heart-simulator via communicating over the USBUART and COM/Serial ports.

2. Chronology

Since designing with all the constraints at once is quite tedious, Splitting the tasks into 6 state-diagrams eases the job.

1. Develop a working VVI- Pacemaker. A VVI pacemaker is a subset of a DDD Mode Pacemaker. I used the SCChart model of a working VVI-Pacemaker as a starting model. It implements 2 / 6 of the timing constraints.
2. Add the URI timing Constraint. Implement the URI constraints on the VVI model. This step was the most tedious. A perfect sync between LRI_Restart and URI_Restart is required to avoid multiple paces.
3. Implement PVARP timing constraint. This is almost identical to the VRP constraint implemented in the VVI-pacemaker. Replacing V_Pulse with A_Pulse and VR and AR does the job here, while also adding an extra transition to generate AS inside the **idle state.**
4. Adding the AEI and AVI constraint. Since both are mere opposites of each other, coming up with a starting solution wasn't difficult. But syncing with the existing constraints was not trivial. Also an extra signal **VP_Req** was introduced to take care of delaying the VP generated by AVI, to not violate URI constraint. An extra state **paceURI** was also introduced to handle the **VP_Req** generated via the **AVI** process.

Following the steps above, the SCChart model was developed which consisted of **6** separate processes, each responsible for maintaining the respective Timing Constraint.

3. Hurdles and Bugs

Even after the radius process of developing the SCChart, there were several bugs. Bugs were present within the **SCChart.** Immediate transitions were the culprit. So many extra (*maybe unnecessary*) transitions were introduced to be 100% sure regarding the behaviour under many circumstances. Looking at the processes **AVI and URI**, one can easily make this out. Some tweaking in the immediate transitions and connector states finally got the SCChart to compile. Linking and Mapping with the signals in the PSoc was straightforward, though attention was needed while programming the ISR. The control flow of the event loop of the PSoc was:

1. Assert V_Pulse and A_Pulse to **LOW**
2. Scan the input for A and V signals
3. Handle the timers for Start/Stop or Reset Signals
4. Execute the SCChart **tick()** event
5. Output the Pacing Event (*If any)* on the serial connection

While testing with the **heart emulator,** I encountered 2 more bugs which were violating the ***URI Constraint*** and multiple pace-s were registered. Those were fixed by changing the immediate transitions in URI and LRI processes. Since the URI and LRI process can generate the ***VP*** signal, all the errors could only be in those processes.

Even after investing most of the time on debugging, a hard-to-reproduce bug, which ona rare event paces the heart twice is being observed on the **heart-emulator**. Hopefully this can be resolved in the near future.

4. Work-Division

Reading and Initial Research	2 hours
Planning	30 min
Modelling (SCChart)	2 hours
Coding (PSoc)	2 hours
Testing and Debugging	6 hours
Documentation	1 hour