# Assignment 4

Part- I

**1.Why cardiac pacemaker requires formal verification?**

Pacemaker is a life-critical real-time system which has complex timing constraints. Formal verification with timed automate is checked with respect to timing properties. Guaranteeing timing properties on it implementation is essential because the pacemaker is a safety-critical real-time system. Violation of timing properties can result in fatal loss of properties or life.

**2.VDD mode of operation** – (V- Ventricle D- Dual atrium and ventricle, D- both inhibit and trigger pacing)

VDD one of non-rate adaptive operating modes of a pacemaker, Ventricle paced, atrium and ventricle sensed and either inhibition or triggering of the pacemaker in response to a sensed beat. It is may be appropriate for the patient with normal sinus node function and conduction disease of the AV node (Chambers paced and Chamber sensed A-Atrium and V –Ventricle). It is Dual chamber (two lead) pacing systems. Atrial sensing is accomplished from floating sending electrodes on the atrial portion of the ventricular pacing lead. The initial portion of the cycle consists of the ventricular refractory period(VRP) 150-500ms.

**3a) what is being modeled in the case study?**

Formal modeling with timed automata of real-time software system(pacemaker) consists modeling of ventricles and heart. During first phase (requirements and design phases of the software life cycle), formal modeling is started by using Boston’s Scientific system specification to make timed automata. During second phase the model checking is performed on timed automata model with respect to timed properties of pacemaker in VVI mode using UPPAAL model checker tool.

Timing properties LRI, HRI and VRP are considered most important timing periods which needs to be verified.

**Ventricle Model:**

Ventricle automation captures sending signals from the ventricle of the heart and emitting ventricular pacing signals to the heart.

There are two states WaitRI and WaitRP in the ventricle timed automation.

**Heart Model:**

It simulates a human heart and environment for us to simulate and verify the pacemaker software in modeling phase. The heart waits a ventricular pacing event from the pacemaker or randomly sends a ventricular sensing event to the pacemaker during time interval between lower bound (minwait) and higher bound(maxwait) for two consecutive heart beats in Ready state.

3b) **Temporal properties considered in verification of the pacemaker**

In VVI mode of pacemaker three properties are considered most important timing periods which should be guaranteed. They are LRI (Lower Intrinsic Rate), HRI (Heuristic Intrinsic Rate) and VRP (Ventricle Refractory Period).

HRI: when hysteresis pacing is enabled, pacemaker uses a pre-defined value HRI. The HRI allows the sensed Intrinsic rate to decrease to a level lower than that of the programmed lower rate. HRI provides the capability to patients own intrinsic rate while pacing a faster rate if the intrinsic rate falls below the HRI. The heuristic rate is always slower than that of the programmed lower rate. The LRI timer is initiated by the programmed event while the HRI is sensed by the non-refractory event.

3c) **how to use model checking to check whether the model satisfies the property**

From the ventricle controller model, we observed two states (WaitRI and WaitVRP) and 3 transitions in the Ventricle timed automation. Model checking is performed using UPPAAL tool. Initially we map timing requirements of verification queried in UPPAAL. There are 4 properties (PropDeadlock, PropLRI, PropHRI, PropVRP) used in the verification. Using Deadlock, we ensure there are no deadlocks in any execution sequence. We check LRI value when hysteresis pacing is disabled. We check HRI when hysteresis pacing is enabled. The pacemaker uses pre-defined value HRI instead of LRI after every sensing. VRP is used to check the ventricular controller follows the VRP period requirement or not by checking whether it stays in the WaitRI state when pacemaker detects a ventricular sensing or pacing event. From these constraints on each property, the model checking on timed automata model is verified.

Model checking using Uppaal

<http://dmi.uib.es/~jproenza/SistEncTR/TheUppaalModelChecker.pdf>

3d) **Arguments**

Overall assurance of the system depends on the system specification. For example, pacemaker depends of Boston Scientific’s system specification. Later it needs to map to model checking tool. Clearly it tells that specification intuitive and straightforward to use timed automata. Developer can not design formal model unless he/she finds intuitiveness. How to make that kind of document so that model can be designed well? How does can be done in the scope of software life cycle development phases.

II. **Identify an article or resource that describes model-checking**

Probabilistic model checking for comparative analysis of automated air traffic control systems

<http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=7001427&tag=1>

**summary:**

Air traffic control ensures aircraft stay safely separated. To achieve the required level of assurance for this safety-critical application, the Automated Airspace Concept (AAC) proposes a network of components providing multiple levels of separation assurance, including conflict detection and resolution. In our previous work, we conducted a formal study of this concept including specification, validation, and verification utilizing the NuSMV and CadenceSMV model checkers to ensure there are no potentially catastrophic design flaws remaining in the AAC design before the next stage of production. In this paper, we extend that

workto include probabilistic model checking of the AAC .we are motivated by the system designers requirement to compare different design options to optimize the functional allocation of the AAC components. Probabilistic model checking provides quantitative measures for evaluating different design options, helping system designers to understand the impact of parameters in the model on a given critical safety requirement. We detail our approach to modeling and probabilistically analyzing this complex system consisting of a real-time algorithm, a logic protocol, and human factors. We utilize both Discrete Time Markov Chain (DTMC) and Continuous Time Markov Chain (CTMC) models to capture the important behaviors in the AAC components. The separation assurance algorithms, which are defined over specific time ranges, are modeled using a DTMC. The emergence of conflicts in an airspace sector and the reaction times of pilots, which can be simplified as Markov processes on continuous time, are modeled as a CTMC. Utilizing these two models, we calculate the probability of an unresolved conflict as a measure of safety and compare multiple design options.

III. **Include relevant questions and concerns.**

Q). Code is synthesized from the timed automata model once the model is verified. The generated code behavior varies based on the platform on which it is run. How to ensure the timing properties across all the platforms?