# Closed-Loop Pacemaker Testing and Verification Test-Bed

# Real-Time Heart Model on a Chip

Sriram Radhakrishnan, Varun Sampath, Shilpa Sarode

Advisors: Dr. Rahul Mangharam, Miroslav Pajic, Zhihao Jiang

Penn University of Pennsylvania

mLAB Real-Time & Embedded Systems Lab

## Background and Motivation

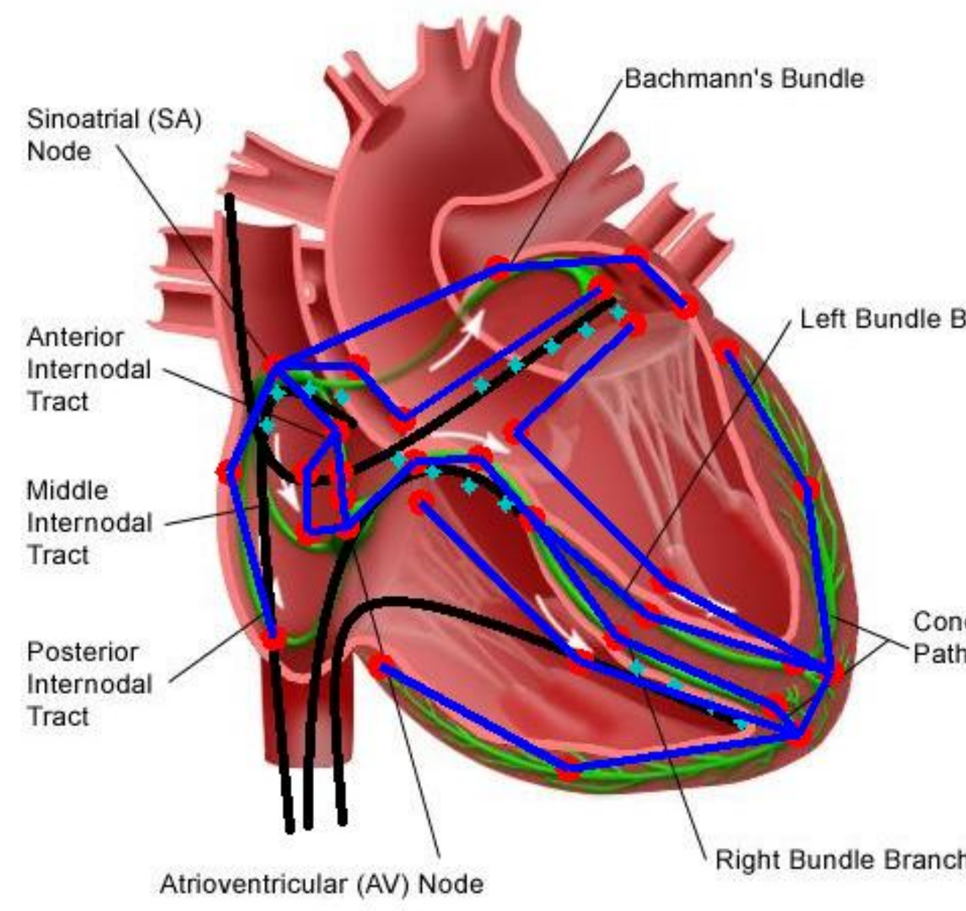
- From 1990-2000, over 200,000 pacemakers were recalled due to firmware issues
- Pacemakers are programmed with 80,000 to over 100,000 lines of code
- There is no standardized, closed-loop system for testing pacemaker functionality

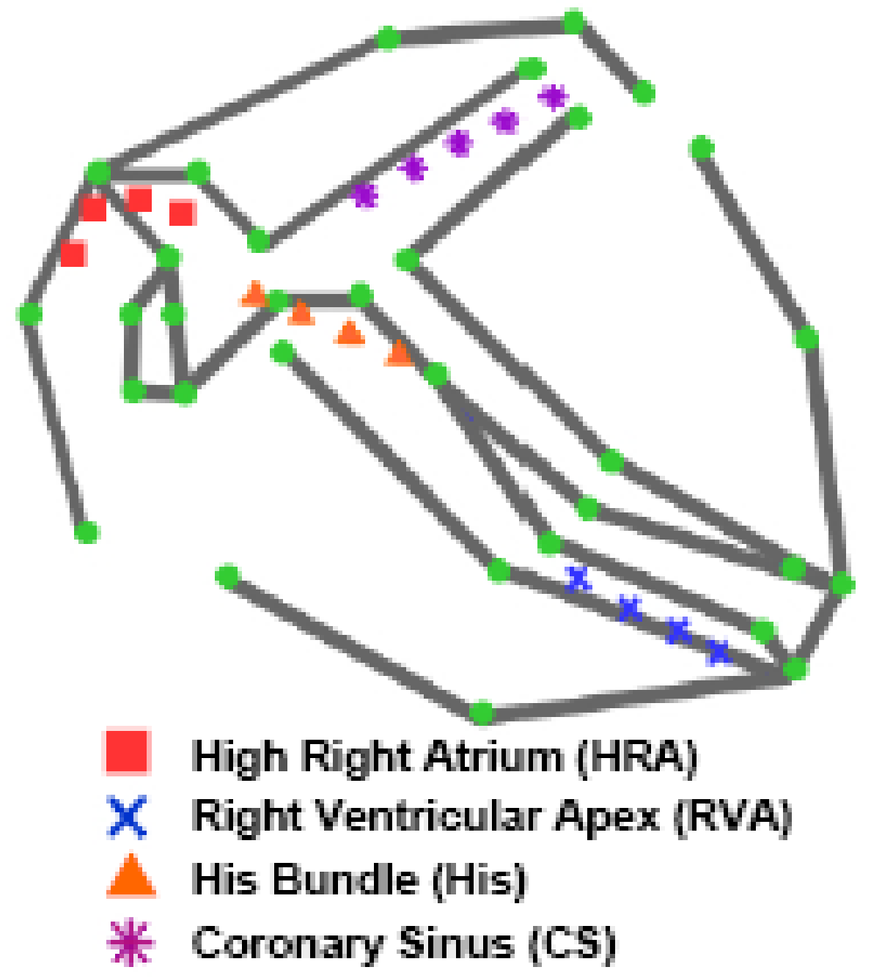
Idea: Build a "virtual heart" on a chip to provide a closed-loop testing environment for pacemakers

## Electrophysiology

- Natural pacing signal generated from sinoatrial node (heart's "pacemaker")
- Signal traverses atria and ventricles to activate cardiac muscle
- Model tissue regions as "nodes" and connections between regions as "paths"

Electrical System of the Heart





## System Block Diagram

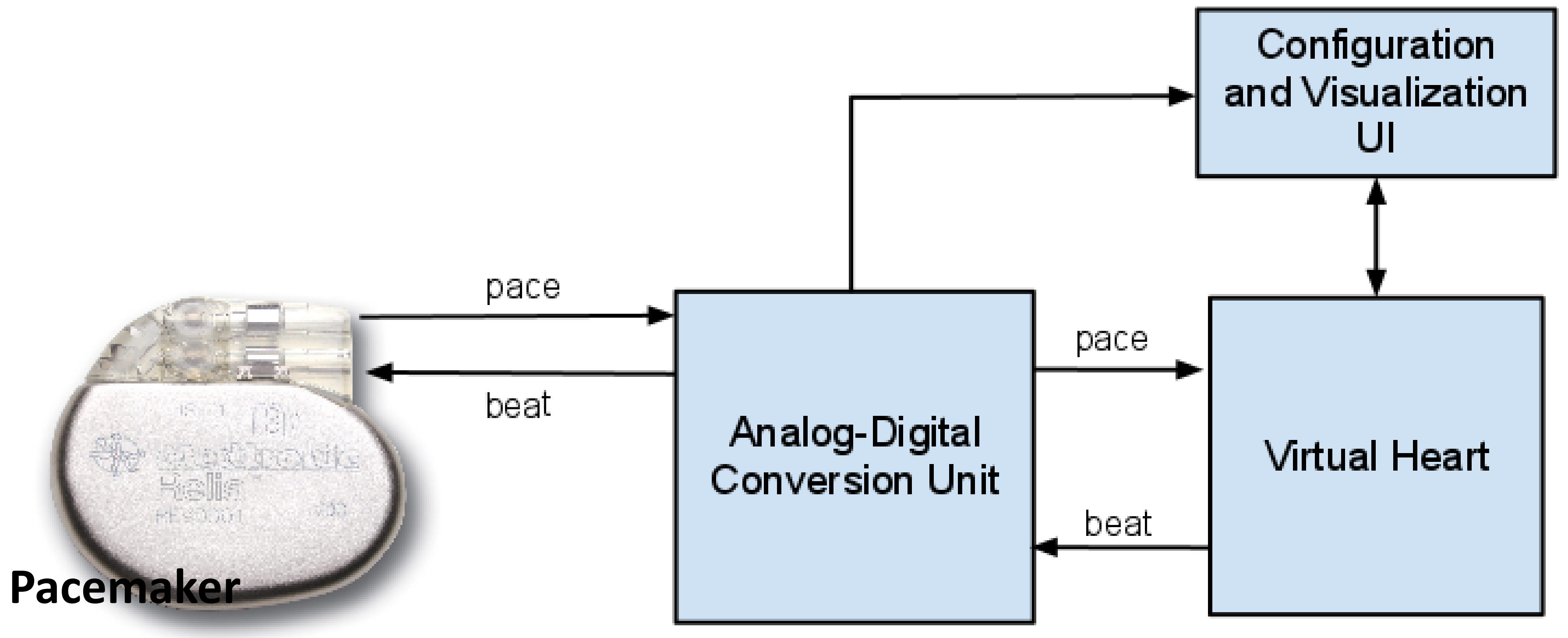
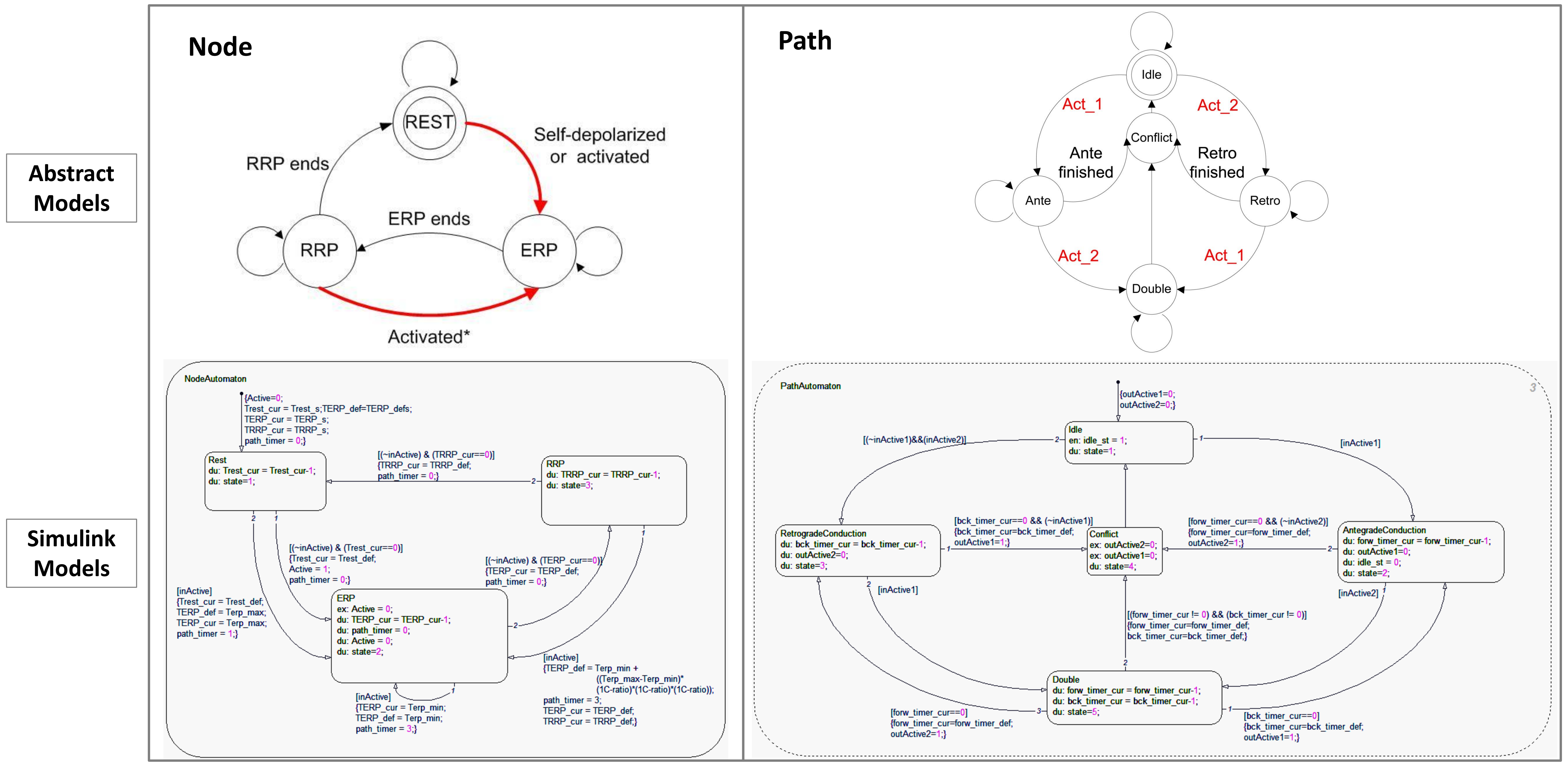

Configuration and Visualization UI
pace
beat
Analog-Digital Conversion Unit
pace
Virtual Heart
beat
Pacemaker

Image Credit Medtronic

## Finite State Automata

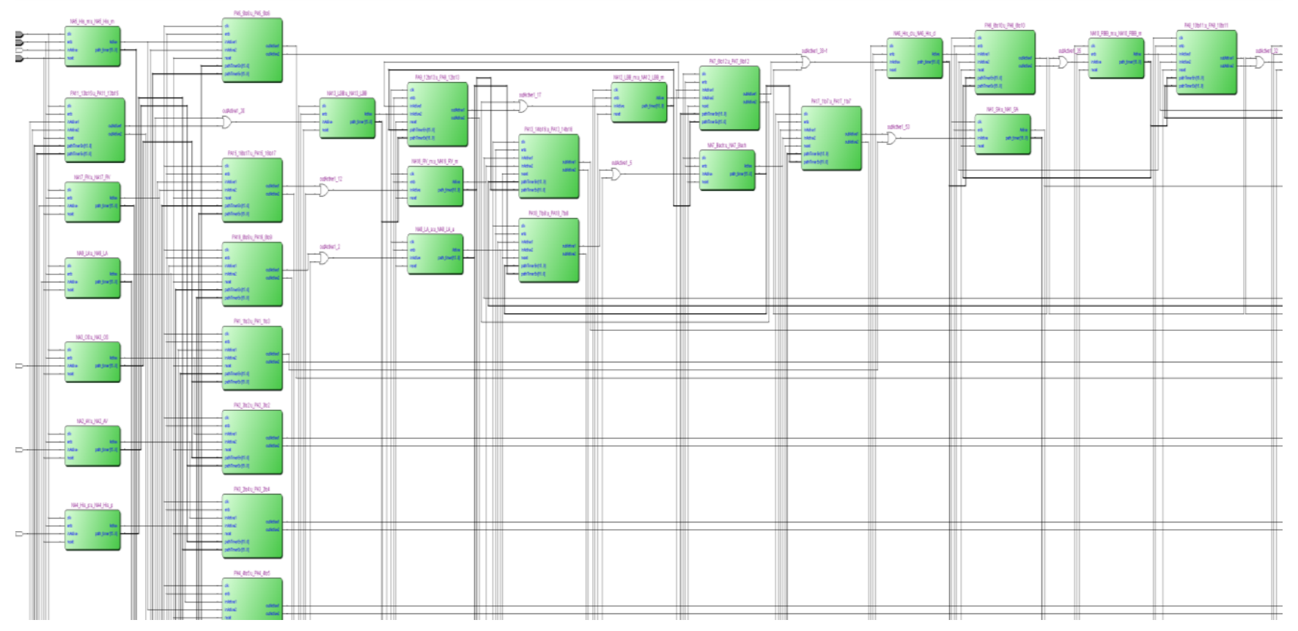
- Each node and path is represented by a finite state machine
- State transitions are governed by timing parameters, based on signal propagation properties of the heart

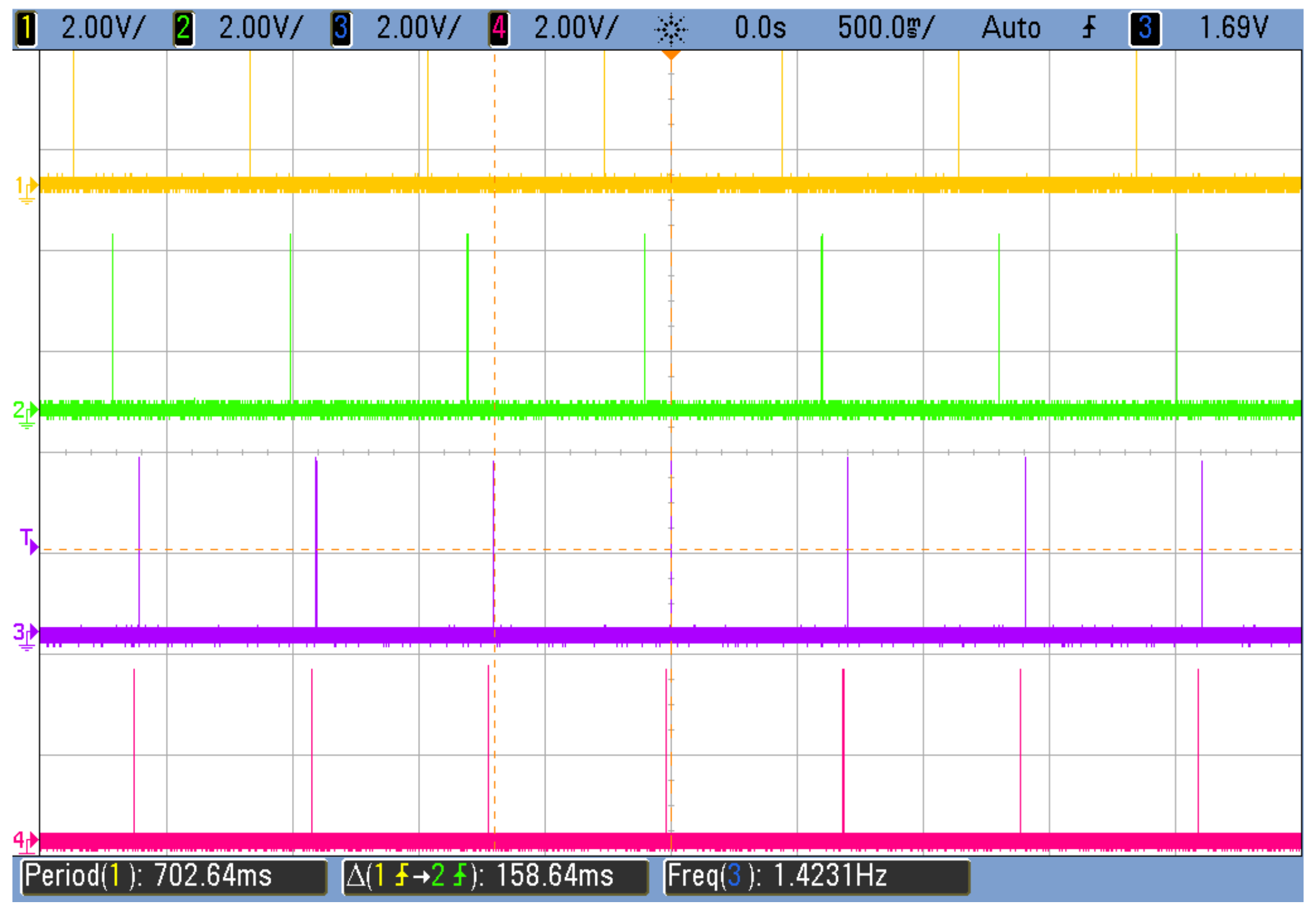


## FPGA Implementation

Terasic DE0-Nano Board
Altera Cyclone IV FPGA

- Current Model: 19 Nodes and Paths
- LUT Utilization: 34%
- RTL schematic subset below




1 2.00V/ 2 2.00V/ 3 2.00V/ 4 2.00V/ 0.0s 500.0ms/ Auto 3 1.69V
Period(1): 702.64ms
Δ(1→2): 158.64ms
Freq(3): 1.4231Hz

- Preliminary Heart Model Output
- Spikes represent node activation, when the node receives the signal
- Shows signal propagation through the heart

## Model Based Development

- Use models (finite state automata) to represent relevant systems and states
- Use simple models of the nodes and paths to build a complex model of the heart using MATLAB Simulink (heart model shown on right)
- Use automatic code generation tools to convert models to HDL for the FPGA board
- Substantially reduces human coding errors
- Can use model-checking tools (e.g. UPPAAL) to verify model properties and specifications