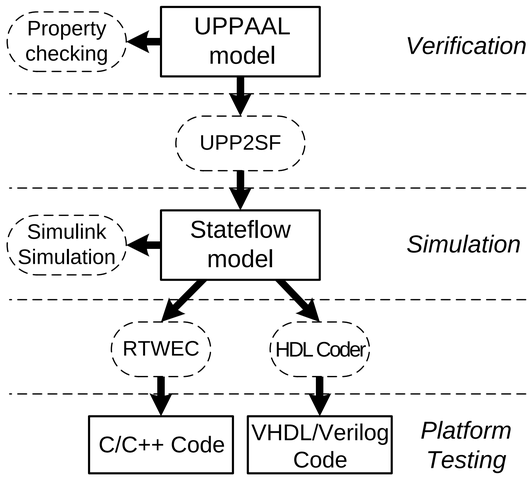
From Verification to Implementation: A Model Translation Tool and a Pacemaker Case Study

Miroslav Pajic, Zhihao Jiang, Insup Lee, Oleg Sokolsky, Rahul Mangharam

University of Pennsylvania — PRECISE: Penn Research in Embedded Computing and Integrated Systems Engineering

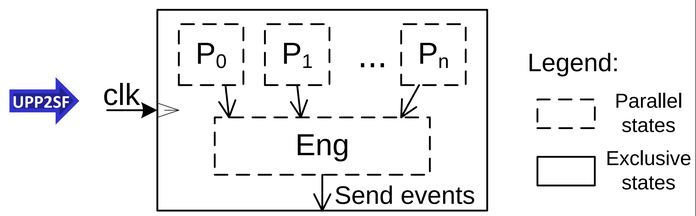
1 Model-Driven Development

Goal is to integrate
- system modeling
- verification
- model-based WCET analysis
- simulation
- code generation
- testing



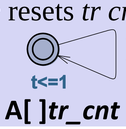
2 UPP2SF: UPPAAL → Stateflow

Network of UPPAAL automata $P_0, P_1 \ldots P_n$



Translation procedure maps:
- Guards and clock invariants
- Synchronization over binary and broadcast channels
- Committed and urgent states

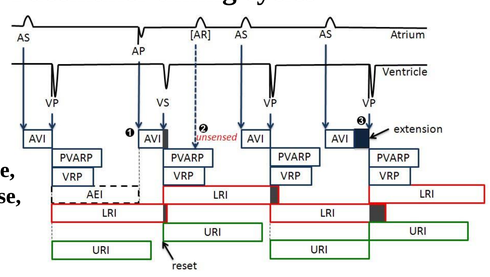
WCET Analysis
- add variable *tr_cnt*
- resets *tr cnt* = *tr cnt* + 1



$A[\,]tr\_cnt \leq N$

3 Pacemaker Case Study

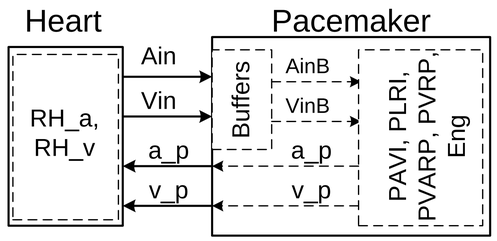
Pacemaker timing cycles

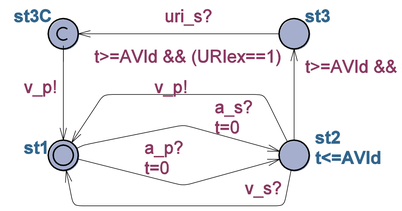


VP - ventricular pace,
VS - ventricular sense,
AP - atrial pace,
AS - atrial sense.

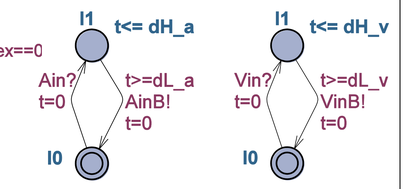
4 Pacemaker Modeling in UPPAAL

Interaction between the pacemaker and heart

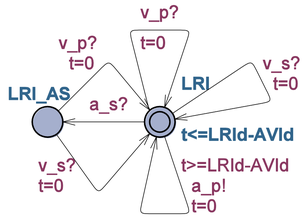




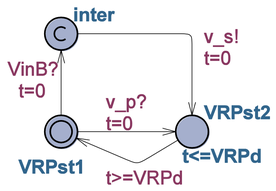
AVI automaton



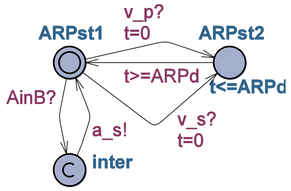
Atrial and Ventricular buffers



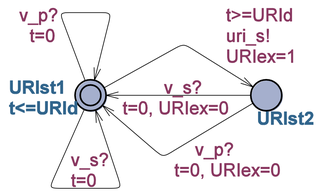
LRI automaton



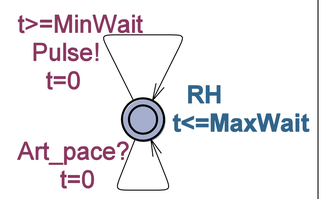
VRP automaton



PVARP automaton

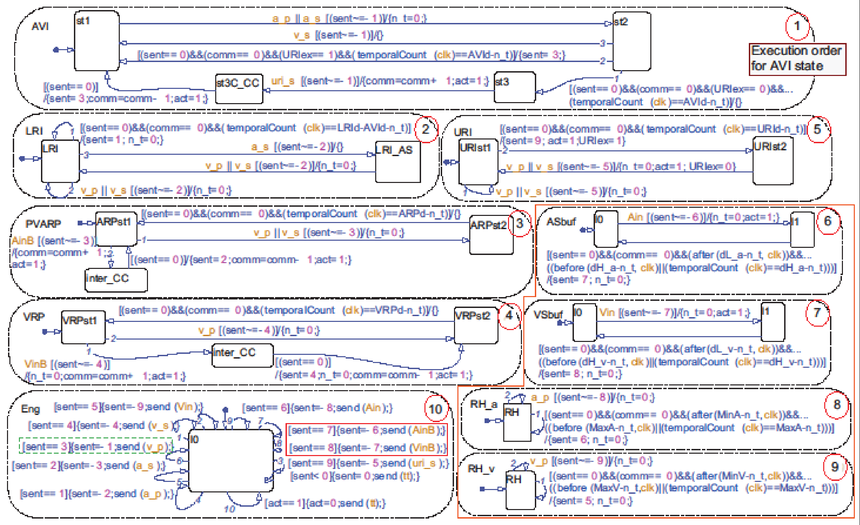


URI automaton



Random Heart

5 Pacemaker Stateflow Design



6 Code Synthesis

Listing 1. `bitsForTID0` definition

```
struct {
uint_T is_AVI:3;
uint_T is_LRI:2;
uint_T is_PVARP:2;
uint_T is_VRP:2;
uint_T is_URI:2;
uint_T is_active_AVI:1;
uint_T is_active_LRI:1;
uint_T is_active_PVARP:1;
uint_T is_active_VRP:1;
uint_T is_active_URI:1;
uint_T is_active_Eng:1;
uint_T is_Eng:1;
uint_T URI_ex:1;
} bitsForTID0;
```

Listing 2. `Rt_OneStep` procedure

```
detect which of the input events are active;
for each of the input events {
if EventName is active   {
sf_previousEvent = _sfEvent_;
   _sfEvent_ = EventName;
   cl_ChartName();
   _sfEvent_ = sf_previousEvent;
  }
}

update the outputs;
update the input events states;
```

Listing 3. `cl_ChartName()` procedure

```
increase counters associated with _sfEvent_;
for each of the parallel states {
 processState();
}
```

Listing 4. `processState()` procedure

```
if (rtDWork.bitsForTID0.is_active_NAME != 0) {
switch (rtDWork.bitsForTID0.is_NAME) {
case SubStateName1:
/* the loop below is - checkTrans();*/
for all transitions in ex. order {
if transition enabled {
execution transition actions;
reset the corresponding temporal counters;
update rtDWork.bitsForTID0.is_NAME;

break for
        }
      }
break;

case SubStateName2:
checkTrans();
break;
    ...
default:
rtDWork.bitsForTID0.is_NAME=IN_NO_ACTIVE_CHILD;
break;
  }
}
```

Listing 5. `broadcast_tt()` procedure

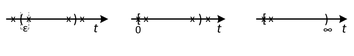
```
static void broadcast_tt(void) {
int16_T sf_previousEvent;

sf_previousEvent = _sfEvent_;
  sfEvent_ = event_tt;
  cl_ChartName();
 _sfEvent_ = sf_previousEvent;
}
```

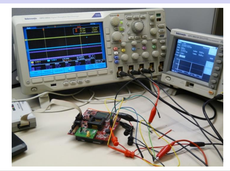
7 Testing of the Physical Implementation

Real-Time requirements - e.g., Pacing in the atrium:
1. AP cannot occur during the interval $0 \leq t_v \leq LRI_d\text{-}AVI_d$;
2. If AS does not occur within the interval, AP should occur at $t_v = LRI_d - LRI_d$
3. If AS occurs at t_v ($0 \leq t_v \leq LRI_d\text{-}AVI_d$), AP should not be applied in the atrium within the interval $0 \leq t_v \leq LRI_d\text{-}AVI_d$.

Test Scenarios



Implementation – on top of the *nanoRK* RTOS



CPU frequency	Average ex. time	Minimal ex. time	Maximal ex. time	Standard deviation
4MHz, OL	176.1μs	167.6μs	462.9μs	14.2μs
4MHz	180.9μs	167.6μs	738.2μs	17.3μs
8MHz, OL	89.5μs	84.7μs	234.6μs	7.2μs
8MHz	92.0μs	84.9μs	370.4μs	13.7μs