Comparison of Classical IF-clause Modelling and Statechart Modelling for ARGESIM Benchmark C3 'Class-E amplifier' using MOSILAB/Modelica

Günther Zauner, Gemma Ferdinand Kaunang, Vienna Univ. of Technology, Austria;
gzauner@osiris.tuwien.ac.at

Simulator: Mosilab (Modeling and Simulation Laboratory) is a simulation tool for complex technical systems developed by Fraunhofer Gesellschaft. For the modeling process, MOSILAB uses the object- and equation-oriented model description language Modelica®, with a backwards-compatible extension to incorporate elements for describing model structure dynamics. This extension is defined in the model description language MOSILA. An integrated development environment offers users support in every work step – from model building over simulation to post-processing. Besides the traditional component diagram and textual modelling layer, class and statechart diagrams are available to users for the model-based development process.

MOSILAB can be used for applications in automotive and energetic systems, mechatronics, multi domain physics, block digram modelling and others.

Model: The basic class-E power amplifier was introduced by N.O. Sokal and A.D. Sokal in their classic paper from 1975. It is a switching-mode amplifier that operates with zero voltage and zero slope across the switch at switch turn-off. The equations describing the circuit are the state-equations where inductor currents and capacitor voltages are chosen as system variables. Kirchhoff laws for voltage and current deliver the following differential equations:

$$dx1/dt = (-x2 + VDC)/L1 \quad (1)$$

$$dx2/dt = (x1 - x2/R(t) - x3)/C2 \quad (2)$$

$$dx3/dt = (x2 - RL*x3 - x4)/L3 \quad (3)$$

$$dx4/dt = x3/C4 \quad (4)$$

The aim was to implement these equations into MOSILAB structure. Therefore two different ways have been chosen to model these equations.

The first solution is using textual Modelica notation. Designing the model is relativly easy in Modelica by using the exact equations above in the equation section and declaring all variables in the beginning section. The time dependent resistor $R(t)$ is modelled using an algorithm section, the whole modelling code is listed below:

```
 1 model C3Mosilab_taskc
 2 constant Real L1= 79.9E-6; C2= 17.9E-9;
 3 ...
 4 Real x1(start=0.26144);
 5 ...
 6 Real Rt; Real t_red; Real k;
 7 Real IRT; Real VRL; Real derx3;
 8 equation
 9     t_red= mod(time, 10E-6);
10     k=((5e+6)-(5e-2))/TRF;
11 algorithm
12    if (0<=t_red) and (t_red<TRF) then
13       Rt:= (5e-2) + k*t_red;
14    elseif(TRF<=t_red) and (t_red<(5e-6)) then
15       Rt:= 5e+6;
16    elseif ((5e-6)<=t_red)
                and (t_red<((5e-6)+TRF)) then
17       Rt:= (5e+6) - k*(t_red - (5e-6));
18    elseif ((5e-6)+TRF<=t_red)
                and (t_red<(10e-6)) then
19       Rt:= 5e-2;
20    else
21       Rt:= -5;
22    end if;
23 equation
24    L1 * der(x1)= -x2 + VDC;
25    C2 * der(x2)= x1 -(x2/Rt) - x3;
26    ...
27 end C3Mosilab_taskc;
```

Listing 1: Mosilab Code in Modelica standard notation for the solution of Task C of the ARGESIM Benchmark 3

For the second solution the MOSILA language extension for statechart modelling is used, dividing the system in separated model parts, depending on the state of the time dependent resistor $R(t)$. It switches between the state $OFF(s1)$ and the state $ON(s2)$. Before simulation, $s1$ is set up as initial state. Thus, the model will change to $s2$ when the time dependent resistor $R(t)$ reaches value $50m\Omega$ and the model will again change to $s1$ when $R(t)$ reaches $5M\Omega$. By using the same code in an algorithm section as definied in the previous solution for $R(t)$, the value of the time dependent resistor is implemented. Declaring all variables in the beginning section and adding statechart code is realized as follows:

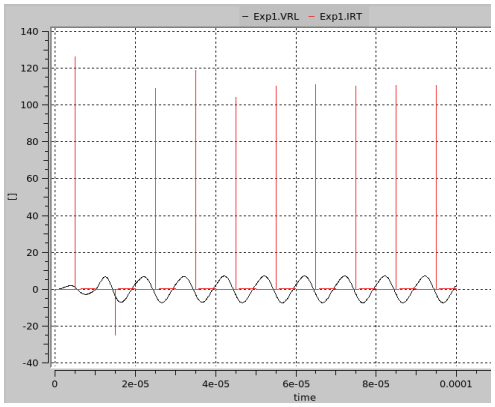

Figure 1: Time Curve $IR(t)$ and VRL

```
1  equation
2     s1 = if Rt>=5e+6 then true else false;
3     s2 = if Rt<=5e-2 then true else false;
4  statechart
5  state C3MosilabStateSC extends State;
6    State State1; State State2;
7    State Initial(isInitial=true);
8    transition Initial->State1
       action Rs:=5e+6;
9    end transition;
10   transition State1->State2 event s2
11     action Rs:= 5e-2;
12   end transition;
13   transition State2->State1 event s1
14     action Rs:= 5e+6;
15   end transition;
16 end C3MosilabStateSC;
```

Listing 2. Statechart code

The program codes of all solutions are done in the modelling section of Mosilab. Compiling the code, setting up the simulation parameter and simulation process is performed in the simulation section of Mosilab. The result of simulation is shown in the post-processing section.

A-Task: Calculating the eigenvalues of the system when $R(t)$ is $ON(50m\Omega)$ and when $R(t)$ is $OFF(5M\Omega)$ cannot be realized using Mosilab, because it does not have a `eigenValues` function in the Modelica library.

B-Task: Simulation of this task is done by setting `Dassl` as integration solver, $1e-10$ as min. stepsize, $1e-07$ as ma.x stepsize and $0\ldots1e-5$ sec as simulation time interval. Choosing the initial value zero for $x1, x2, x3$ and $x4$, the results for current $IR(t)$ and output voltage VRL are presented in fig. 1.

C-Task: Simulation of this task was performed using the same settings as in task b apart from

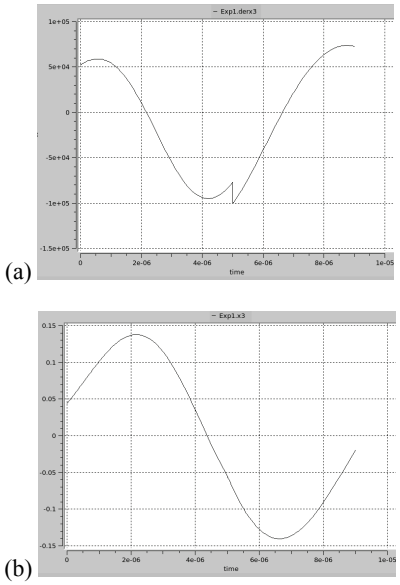

Figure 2: Time Curve (a) $VL3$ and (b) $IL3$

the simulation time interval which was chosen as $0\ldots9e-6$ sec. Reinitialization of the start parameters can be done using a script file or manually. Figure 2 shows the time curves for the voltage at coil $L3$ and the current at $L3$. Plotting this task is performed by using only a time curve of $VL3$ and $IL3$, because of lack of plot options for phase plane curves.

For all calculations and simulations, Mosilab 3.1 was used.

Resumé: Mosilab is a powerful simulation tool that makes it quite simple to model physical system, because it is based on Modelica, an object- and equation-oriented model description language. The unique part of Mosilab is that models can be implemented by using a statechart approach. This extension of the Modelica standard can be used to implement state events of any kind. A feature which can not be solved by every Modelica simulator or other standard simulation software. Mosilab has until now the disadvantage that it cannot calculate eigenvalues

Mosilab offers explicit as well as implicite algorithms for numerical integration and is therefore a adequate simulator.

Corresponding author: Günther Zauner,
dieDrahtwarenhandlung Simulation Services
Neustiftgasse 57-59, 1070 Vienna, Austria
guenther.zauner@drahtwarenhandlung.at

Received: October 6, 2007
Revised: October 20, 2007
Accepted: November 3, 2007