Buletinul Ştiinţific al Universităţii "Politehnica" din Timişoara

Seria ELECTRONICĂ şi TELECOMUNICAŢII
TRANSACTIONS on ELECTRONICS and COMMUNICATIONS

Tom 49(63), Fascicola 1, 2004

On the use of Modelica modelling language in modelling and simulation of electronic circuits

Dorel Aiordăchioaie[1]

Abstract - The work is related to the use of Modelica modelling language, object oriented and physical decomposition based, as a neutral modelling language in the modelling of the electronic circuits. Such a language is suitable for hybrid circuits, with various transformations of energy, e.g. from mechanical to electrical one or vice versa. The following circuits are considered for modelling and simulation: a passive circuit (RLC) to put in the light the basic principles of the modelling language, and an oscillator, to present the flexibility and the portability of the models.

Keywords: Electronic Circuits, Models, Modelling and Simulation, Modelling languages, Object-Oriented Technology, Physical decomposition.

I. INTRODUCTION

Working and continuously looking to new modelling languages and simulation environments could be strange for an electronic engineering world, where the offer is quite various and structural complete, i.e. covering both analogous and digital circuits. There must be a reason for such an interest in new modelling languages and simulation environments. The reason is on the reality that no modelling language can cover the diversity of electronic circuits and applications. Obviously, each language has an optimum concerning the set if suitable problems and aims to cover. A very nice, efficient and practical state-of-the-art in the field of modelling and simulation must include some sound references like [6], [8] and [11].

Modelica is a declarative modelling language useful for hybrid circuits and especially for multi-domain systems, as is robotics. Modelica is independent of the solver and from here a strong feature is obtained: the portability of the model among various simulation platform.

The aim of the work is to present the main features of the Modelica modelling language using two simple electric circuits and to sketch some similarities with other existing and intensively used modelling languages.

In section II the main features of the modelling language are presented . The next two sections are for examples, with the goal of modelling and simulating simple scenarios.

One of the best paper regarding the use of Modelica modelling language in conjunction with SPICE is [7]. It is the reason also of the present incursion and exploration. Also, in the field of electric circuits, some results could be referred in the modelling and simulating of MOSFETs, [19], or DC-DC converters, [16].

The effort of the research community involved in modelling and simulating, Modelica related, is oriented to the design of specialized libraries, i.e. domain oriented, and into design of automated translation of models in/to Modelica from other intensively used modelling and simulation environments, like SPICE, [2], and Matlab-Simulink, [19].

II. ABOUT MODELICA

Modelica is a declarative modelling language for continuous, discrete and mixed systems based on declarative models and description of the knowledge. The main attraction for such a language is coming from the fact that it follows some natural modelling principles, as physical decomposition of the considered system or object based technology. It is interesting that process modelling under Modelica is no longer a challenge but a pleasure.

A strong feature is that it is continuously improving based on collaborative work of people coming from industry, research centres and universities. More details and explanation could found in, e.g. [3], [4], [9], [12], and [15].
Based on these sound principles a lot of work was done to study and develop formalism to represent some classical formalism of modelling as bondgraph, [1], or Petri Nets, [13]. A lot of effort is also made to develop domain specific libraries as in [18] for a

[1] "Dunarea de Jos" Galati University
Electrical Engineering Faculty, Domneasca-47, Galati-800008
email: Dorel.Aiordachioaie@ugal.ro

BUPT

thermo-hydraulic processes, [14] and [15] in [illegible] ulti-b y mechanical arms and hybrid domains.

The reason for a such high interest in Modelica is based on the natural and easy understanding of the modelling principles. Also, Modelica could serve as a [illegible] and possible distributed simulation environments.

Fig.1 presents a partial metastructure of any Modelica model. A model is an aggregation of components and devices, connected with properly connection. The connection are developed by using ports. A port has a set of variables, which describes the interaction among sub-models.

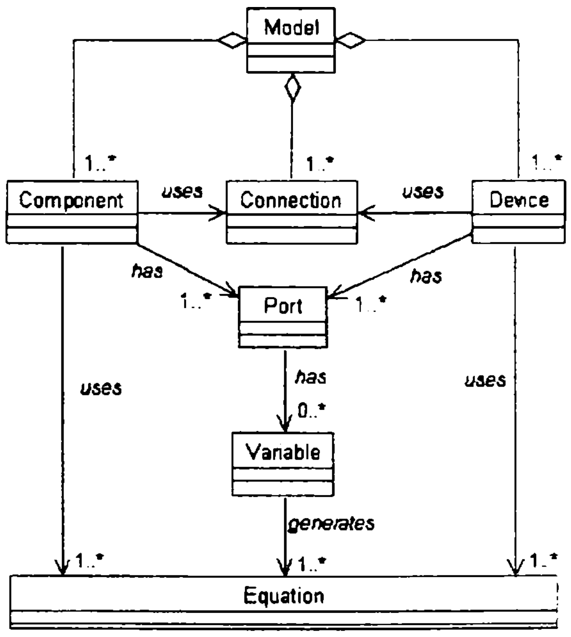

Fig.1. The *metastructure* of the Modelica model

Be putting all the variables, wich describes the behaviour of the models at the ports, and the constitutive equation, which describes the behaviour of sub-models, an equation based model is obtained and used in simulations.

III. A SIMPLE EXAMPLE

The first considered circuit is an RLC circuit as it is represented in Fig.2. It is composed of a voltage source, a capacitor, an inductor and a resistive load. The problem is to study the frequency behaviour of the voltage on resistive load, using a signal generator with variable frequency.

First, an interface must be defined for each component as the place where the interaction with other components or devices is developed.

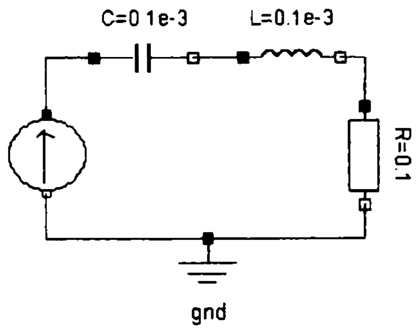

Fig.2. RLC circuit

A simple interface (or port), called *conector* in Modelica, may have two variables to describe the current and the voltage at the considered interface. A common definition is

```
connector Pin
    Voltage v;
    flow Current i;
end Pin;
```

where Voltage and Current are defined as

```
type Voltage = Real (unit="V");
type Current = Real (unit="A");
```

Typically, a component must have at least two connectors to properly describe the electrical behaviour:

```
partial model TwoPins
    Pin p, n;
    Voltage v "Drop Voltage";
    Current i;
equation
    v = [illegible].v - n.v;
    p.i + n.i = 0;
    i = p.i;
end TwoPins;
```

From now on all of the components' description is based on the model TwoPins. The model of the ideal inductor is

```
model Inductor "Ideal electrical inductance"
    extends TwoPins;
    parameter Real L=0.1e-3 "[Henry]";
    equation
        L * der(i) = p.v - n.v;
end Inductor;
```

The model of the ideal resistor is

```
model Resistor "Ideal electrical resistor"
    extends TwoPins;
    parameter Real R=0.1 "[Ohm]";
    equation
        R*i = p.v - n.v;
end Resistor;
```

The model of the ideal capacitor is

BUPT

```
    model Capacitor "Ideal electrical capacitor"
        extends TwoPins;
        parameter Real C=0.1e-3  [Farad s] ;
        equation
            C*der(v) = i;
    end Capacitor;
```

Finally, the voltage reference must be defined as

```
    model Ground "The reference"
        Pin p;
        equation
            p.v = 0;
    end Ground;
```

The model of the signal voltage source is quite [illegible] fr[illegible]cy i[illegible]d:

```
    model SourceAC
      extends TwoPins;
      parameter Real A(unit="V") = 1"Ampl.";
      parameter Real Fmin(unit="Hz") = 50"Freq":
      parameter Real max(unit="Hz")=2000"Freq";
      Real freq(start=50);
      Real increment(start=0);
       algorithm
            freq := F_min + increment;
       equation
            v = A*sin(2*3.14*freq*time);
            der(increment) = (Fmax - Fmin);
    end SourceAC;
```

Finally, the model of the circuit describes the way in which the components are connected:

```
            model RLC_circuit
                SourceAC S;
                Resistor R;
                Inductor L;
                Capacitor C;
                Ground gnd;
            equation
                connect(S.p, C.p);
                connect(S.n, gnd.p);
                connect(C.n, L.p);
                connect(L.n, R.p);
                connect(R.n, gnd.p);
            end RLC_circuit;
```

The resulted graphical model is presented in Fig.3, where each component has a model. As it can be recognized from the presented models, a model has mainly two parts: one for parameters and another one for equations. By considering all the equations from the considered models and adding the equation generated, when a connection is made between components, an equation based model in obtained, as it is presented in Annex 1. Such a model is portable in the sense that could be solved by any general simulation platform. In Fig.4 the behaviour of the load voltage is presented on a time horizon of 1 second.

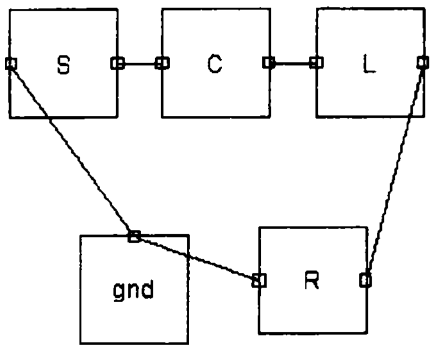

Fig.3. A graphical model of the RLC circuit

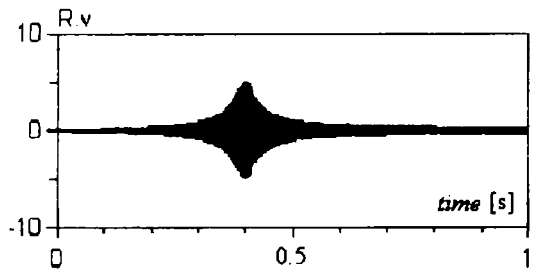

Fig.4. The load voltage

IV. SECOND EXAMPLE

The second example is related to a more complex circuit, an electronic RC oscillator with an operational amplifier. The electric circuit is presented in Fig. 5, it is a basic Wien oscillator.

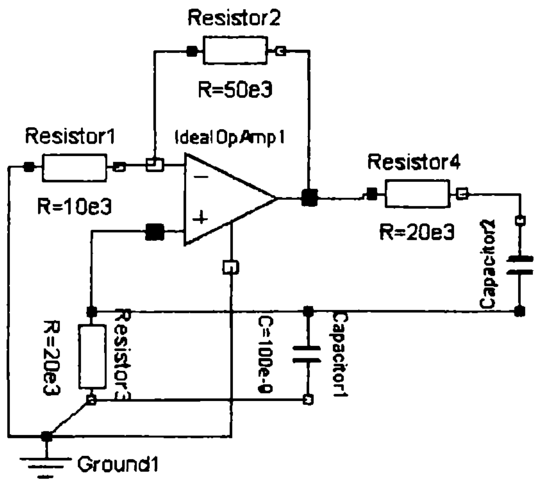

Fig.5. A model of the Wien oscillator

The problem could be defined by requiring the bevaviour of the signals on all nodes of the circuits. The modelling problem is mainly attracted by the model of the operational amplifier. The first model takes into account the general parameters, such as the finite input resistance and capacitance, the finite open loop voltage transfer coefficient. It is a partial model and it is based on the statements:

BUPT

```
partial model AO_simple
    Pin in_m, in_p, out;
    Resistor Rin;
    Capacitor Cin;
    parameter Real a=1e6;
    parameter Real Vmax=10;
    parameter Real Vmin=-10;
    parameter Voltage Voffset=0.1;
    Voltage V,  Vdif, Vout, Vtr;
algorithm
    Vdif := in_p.v - in_m.v;
    V := a*Vdif;
    Vout := V + Voffset + Vtr;
equa i n
    connect(in_p, Cin.p) ;
    connect(Cin.n, in_m) ;
    connect(in_p, Rin.p) ;
    connect(in_m, Rin.n) ;
    out.v = if Vout > Vmax then Vmax
        else if Vout <  Vmin then Vmin
        else Vout;
end AO_simple;
```

Such a model could be used in some applications as amplifier or comparator. If it will be used in the model of Fig.5, some wrong results are obtained, in the sense that no signals on any node will be. The reason is that the model does not contains knowledge of the transient regime, as description of the behaviour of the circuit when it is supplied with energy.

T[illegible] i[illegible] ith k[illegible] l d[illegible] b[illegible] the transient regime:

```
model AO_real_with_transients
    extends AO_simple;
    parameter Real startTime=0.0;
    parameter Real endTime=0.001;
equation
    Vtr = if time < startTime then 0.0
          else if time > endTime
          then 0.0 else Vmax ;
end AO_real_with_transients;
```

Now a model of the circuit will looks like

```
model wien
    AO_real_with_transients AO1 ;
    Resistor R1(R=10e3) ;
    Resistor R2(R=50e3) ;
    Resistor R3(R=20e3) ;
    Resistor R4(R=20e3) ;
    Capacitor C1(C=100e-9);
    Capacitor C2(C=100e-9);
    Ground Gnd ;
equation
    connect(R4.p, Gnd.p) ;
    connect(Gnd.p, C1.p) ;
    connect(AO1.in_p, R4.n) ;
    connect(AO1.in_p, C1.n) ;
    connect(R1.n, AO1.in_m) ;
    connect(R1.n, R2.p) ;
    connect(R2.n, AO1.out) ;
    connect(AO1.out, R3.p) ;
    connect(R3.n, C2.n) ;
    connect(C2.p, C1.n) ;
    connect(R1.p, Gnd.p) ;
end wien;
```

Using the last model the simulation results are satisfactory, in the sense of the right qualitative behaviour, as it is presented in Fig. 6.

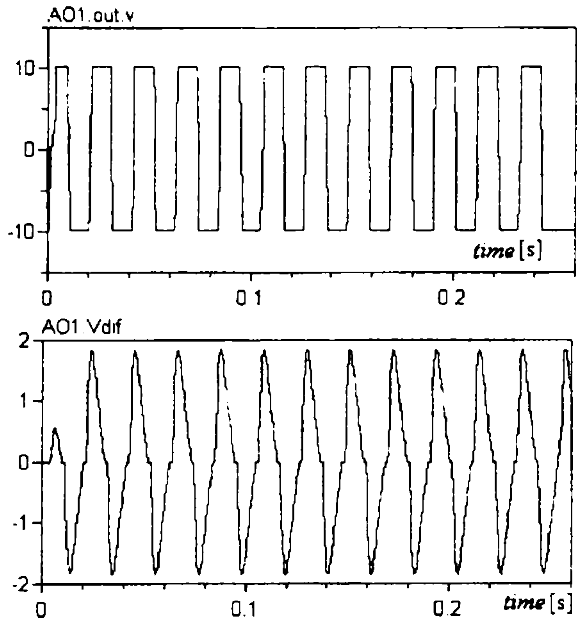

Fig.6. Oscillator's waveforms: output and the differential voltage of the operational amplifier

VI. CONCLUSIONS

The purpose of the paper was to investigate the main features of a new modelling language called Modelica. Two simple examples were considered and discussed: a passive circuit and an active one. In both cases the right results were obtained.

Using Modelica without libraries seems difficult, especially for complex circuits. For simple cases, like those considered here, a library of models is not a big drawback.

Such a language is strongly recommended and used for modelling in neutral format, i.e. solvers independent, and for hybrid and multi-domain circuits, i.e. circuits with physical variables from different domains like electrical, mechanical, chemical etc. To exploit the neutral format representation, many tools are continuously developed to translate models to/from Modelica in other classical representation formalisms. Hybrid circuits, not considered here, for the sake of simplicity, are described in a more natural way by using a physical

BUPT

decomposition and by using variables with physical meanings.

An interesting study, just started, is the similarity in principles with VHDL. Also research is conducted regarding the opportunity to write tools to import and to export models under VHDL to and from Modelica format.

REFERENCES

[1] Broenink J.F., Bond-Graph Modeling In Modelica, *The European Simulation Symp.*, 1997, Passau Germany, Oct. 19-22.

[2] Dempsey Mike, Automatic translation of Simulink models into Modelica using Simelica and the AdvancedBlocks Library, *Proceedings of the 3rd International Modelica Conference*, Linköping, Sweden, November 3-4, 2003, Peter Fritzson (editor), pp.115-124.

[3] Elmqvist H., S.E. Mattsson, and M. Otter,"Modelica - An International Effort to Design an Object-Oriented Modeling Language", *Summer Computer Simulation Conference -98*, Reno, Nevada, USA, July 19-22, 1998.

[4] Elmqvist H., S.E. Mattsson, and M. Otter, "Modelica - A Language for Physical System Modeling, Visualization and Interaction", *The 1999 IEEE Symposium on Computer-Aided Control System Design*, CACSD'99, Hawaii, August 22-27, 1999.

[5] Engelson, V., H. Larsson, and P. Fritzon, "Design, simulation and visualization environment for object-oriented mechanical and mult-domain models in Modelica." *In Proceedings of the IEEE International Conference on Information Visualisation. IEEE Computer Society*, London, UK, 1999.

[6] Kågedal D, "*A Natural Semantics specification for the equation-based modeling language Modelica*," LiTH-IDA-Ex-98/48, Linköping University, Sweden, 1998.

[7] Martin Carla, Alfonso Urquia and Sebastian Dormido, SPICELib - Modeling and Analysis of Electric Circuits with Modelica, *Proceedings of the 3rd International Modelica Conference*, Linköping, Sweden, November 3-4, 2003, Peter Fritzson (editor), pp. 161-170.

[8] Mattsson S.E., M. Andersson, and K. J. Åström: "Object-oriented modelling and simulation". In Linkens, Ed., *CAD for Control Systems*, chapter 2, pp. 31--69. Marcel Dekker Inc, New York, 1993.

[9] Mattsson, S. E., H. Elmqvist, and M. Otter, "Physical system modeling with Modelica", *Control Engineering Practice*, **6**, pp. 501-510.

[10] Mattsson, S. E. and H. Elmqvist, "An Overview Of The Modeling Language Modelica", *Eurosim'98 Simulation Congress*, April 14-15, 1998, Helsinki, Finland.

[11] Mattsson, S. E., M. Andersson, and K. J. Åström, "*Object-oriented modelling and simulation*" In Linkens, Ed., CAD for Control Systems, chapter 2, pp. 31--69. Marcel Dekker Inc, New York, 1993.

[12] Modelica Group, "*A unified object-oriented language for physical systems modeling*". Modelica homepage: http://www.Modelica.org.

[13] Mosterman P. J., M. Otter, H. Elmqvist, "Modeling Petri Nets as Local Constraint Equations for Hybrid Systems Using Modelica", *Summer Computer Simulation Conference -98*, Reno, Nevada, USA, July 19-22, 1998.

[14] Otter M., C. Schlegel, and H. Elmqvist, "Modeling and Realtime Simulation of an Automatic Gearbox using Modelica". *ESS'97 - European Simulation Symposium*, Oct., 1997.

[15] Otter, M., H. Elmqvist, and S. E. Mattsson, "Hybrid modeling in Modelica based on the synchronous data flow principle." *In Proceedings of the 1999 IEEE Symposium on Computer-Aided Control System Design, CACSD'99*. IEEE Control Systems Society, Hawaii, USA, 1999.

[16] Torrey D.A., Selamogullari U.S., A behavioral Model for DC-DC Converter using Modelica , *Proceedings of the 2nd International Modelica Conference*, March 18-19, 2002, Oberpfaffenhofen, Germany, pp.167-172.

[18] Tummescheit, H. and J. Eborn, "Design of a thermo-hydraulic model library in Modelica." In Zobel and Moeller, Eds., *Proceedings of the 12th European Simulation Multiconference, ESM'98*, pp. 132-136. Society for Computer Simulation International, Manchester, UK, 1998.

[19] Urguia A., Dormido S., DC, AC Small-Signal and Transient Analysis of Lvel 1 N-Channel MOSFET with Modelica, *Proceedings of the 2nd International Modelica Conference*, March 18-19, 2002, Oberpfaffenhofen, Germany, pp. 99-108.

ANNEX 1 – The complete equation model of the RLC circuit

```
model RLC_circuit
    parameter Real S.A = 1 "Amplitude";
    parameter Real S.Fmin = 50 "Frequency";
    parameter Real S.Fmax = 2000 "Frequency";
    parameter Real R.R = 0.1 "[Ohm]";
    parameter Real L.L = 0.0001 "[Henry]";
    parameter Real C.C = 0.0001 "[Farads]";
    Real S.p.v;
    Real S.p.i;
    Real S.n.v;
    Real S.n.i;
    Real S.v "Drop Voltage";
    Real S.i;
    Real S.freq(start = 50);
    Real S.increment(start = 0);
    Real R.p.v;
    Real R.p.i;
    Real R.n.v;
    Real R.n.i;
    Real R.v "Drop Voltage";
    Real R.i;
    Real L.p.v;
    Real L.p.i;
    Real L.n.v;
    Real L.n.i;
    Real L.v "Drop Voltage";
    Real L.i;
    Real C.p.v;
    Real C.p.i;
    Real C.n.v;
    Real C.n.i;
    Real C.v "Drop Voltage";
    Real C.i;
    Real gnd.p.v;
    Real gnd.p.i;
equation
    S.v = S.p.v-S.n.v;
    S.p.i+S.n.i = 0;
    S.i = S.p.i;
algorithm
     S.freq := S.Fmin+S.increment;
equation
    S.v = S.A*sin(6.28*S.freq*time);
    der(S.increment) = S.Fmax-S.Fmin;
    R.v = R.p.v-R.n.v;
    R.p.i+R.n.i = 0;
    R.i = R.p.i;
    R.R*R.i = R.p.v-R.n.v;
    L.v = L.p.v-L.n.v;
    L.p.i+L.n.i = 0,
    L.i = L.p.i;
    L.L*der(L.i) = L.p.v-L.n.v;
    C.v = C.p.v-C.n.v;
    C.p.i+C.n.i = 0;
    C.i = C.p.i;
    C.C*der(C.v) = C.i;
    gnd.p.v = 0;
    C.n.i+L.p.i = 0;
    L.p.v = C.n.v;
    C.p.i+S.p.i = 0;
    S.p.v = C.p.v;
    L.n.i+R.p.i = 0;
    R.p.v = L.n.v;
    R.n.i+S.n.i+gnd.p.i = 0;
    S.n.v = R.n.v;
```

BUPT

```
    gnd.p.v = R.n.v;
end RLC_circuit;
```

ANNEX 2 – The complete equation model of the Wien oscillator

```
model wien
    parameter Real AO1.a = 1000000.0;
    parameter Real AO1.Vmax = 10;
    parameter Real AO1.Vmin = (-10);
    parameter Real AO1.Voffset = 0.1;
    parameter Real AO1.Rin.R = 0.1 "[Ohm]";
    parameter Real AO1.Cin.C = 1E-010 "[Farads]";
    parameter Real AO1.startTime = 0;
    parameter Real AO1.endTime = 0.05;
    parameter Real R1.R = 10000.0 "[Ohm]";
    parameter Real R2.R = 50000.0 "[Ohm]";
    parameter Real R3.R = 20000.0 "[Ohm]";
    parameter Real R4.R = 20000.0 "[Ohm]";
    parameter Real C1.C=1E-007 "[Farads]";
    parameter Real C2.C=1E-007 "[Farads]";
    Real AO1.V;
    Real AO1.Vdif;
    Real AO1.Vout;
    Real AO1.Vtr;
    Real AO1.in_m.v;
    Real AO1.in_m.i;
    Real AO1.in_p.v;
    Real AO1.in_p.i;
    Real AO1.out.v;
    Real AO1.out.i;
    Real AO1.Rin.p.v;
    Real AO1.Rin.p.i;
    Real AO1.Rin.n.v;
    Real AO1.Rin.n.i;
    Real AO1.Rin.v "Drop Voltage";
    Real AO1.Rin.i;
    Real AO1.Cin.p.v;
    Real AO1.Cin.p.i;
    Real AO1.Cin.n.v;
    Real AO1.Cin.n.i;
    Real AO1.Cin.v "Drop Voltage";
    Real AO1.Cin.i;
    Real R1.p.v;
    Real R1.p.i;
    Real R1.n.v;
    Real R1.n.i;
    Real R1.v "Drop Voltage";
    Real R1.i;
    Real R2.p.v;
    Real R2.p.i;
    Real R2.n.v;
    Real R2.n.i;
    Real R2.v "Drop Voltage";
    Real R2.i;
    Real R3.p.v;
    Real R3.p.i;
    Real R3.n.v;
    Real R3.n.i;
    Real R3.v "Drop Voltage";
    Real R3.i;
    Real R4.p.v;
    Real R4.p.i;
    Real R4.n.v;
    Real R4.n.i;
    Real R4.v "Drop Voltage";
    Real R4.i;
    Real C1.p.v;
    Real C1.p.i;
    Real C1.n.v;
    Real C1.n.i;
    Real C1.v "Drop Voltage";
    Real C1.i;
    Real C2.p.v;
    Real C2.p.i;
    Real C2.n.v;
    Real C2.n.i;
    Real C2.v "Drop Voltage";
    Real C2.i;
    Real Gnd.p.v;
    Real Gnd.p.i;
equation
    AO1.Rin.v = AO1.Rin.p.v-AO1.Rin.n.v;
    AO1.Rin.p.i+AO1.Rin.n.i = 0;
    AO1.Rin.i = AO1.Rin.p.i;
    AO1.Rin.R*AO1.Rin.i = AO1.Rin.p.v-AO1.Rin.n.v;
    AO1.Cin.v = AO1.Cin.p.v-AO1.Cin.n.v;
    AO1.Cin.p.i+AO1.Cin.n.i = 0;
    AO1.Cin.i = AO1.Cin.p.i;
    AO1.Cin.C*der(AO1.Cin.v) = AO1.Cin.i;
algorithm
     AO1.Vdif := AO1.in_p.v-AO1.in_m.v;
     AO1.V := AO1.a*AO1.Vdif;
     AO1.Vout := AO1.V+AO1.Voffset+AO1.Vtr;
equation
    AO1.out.v = if AO1.Vout > AO1.Vmax then
    AO1.Vmax else if AO1.Vout < AO1.Vmin
     then AO1.Vmin else AO1.Vout;
    AO1.Vtr = if time < AO1.startTime then 0 else if time >
    AO1.endTime then 0 else
    AO1.Vmax;
    R1.v = R1.p.v-R1.n.v;
    R1.p.i+R1.n.i = 0;
    R1.i = R1.p.i;
    R1.R*R1.i = R1.p.v-R1.n.v;
    R2.v = R2.p.v-R2.n.v;
    R2.p.i+R2.n.i = 0;
    R2.i = R2.p.i;
    R2.R*R2.i = R2.p.v-R2.n.v;
    R3.v = R3.p.v-R3.n.v;
    R3.p.i+R3.n.i = 0;
    R3.i = R3.p.i;
    R3.R*R3.i = R3.p.v-R3.n.v;
    R4.v = R4.p.v-R4.n.v;
    R4.p.i+R4.n.i = 0;
    R4.i = R4.p.i;
    R4.R*R4.i = R4.p.v-R4.n.v;
    C1.v = C1.p.v-C1.n.v;
    C1.p.i+C1.n.i = 0;
    C1.i = C1.p.i;
    C1.C*der(C1.v) = C1.i;
    C2.v = C2.p.v-C2.n.v;
    C2.p.i+C2.n.i = 0;
    C2.i = C2.p.i;
    C2.C*der(C2.v) = C2.i;
    Gnd.p.v = 0;
    AO1.in_m.i-(AO1.Cin.n.i+AO1.Rin.n.i) = 0;
    AO1.Rin.n.v = AO1.Cin.n.v;
    AO1.in_m.v = AO1.Cin.n.v;
    AO1.in_p.i-(AO1.Cin.p.i+AO1.Rin.p.i) = 0;
    AO1.Rin.p.v = AO1.Cin.p.v;
    AO1.in_p.v = AO1.Cin.p.v;
    AO1.in_m.i+R1.n.i+R2.p.i = 0;
    R1.n.v = AO1.in_m.v;
    R2.p.v = AO1.in_m.v;
    AO1.in_p.i+C1.n.i+C2.p.i+R4.n.i = 0;
    C1.n.v = AO1.in_p.v;
    C2.p.v = AO1.in_p.v;
    R4.n.v = AO1.in_p.v;
    AO1.out.i+R2.n.i+R3.p.i = 0;
    R2.n.v = AO1.out.v;
    R3.p.v = AO1.out.v;
    C1.p.i+Gnd.p.i+R1.p.i+R4.p.i = 0;
    Gnd.p.v = C1.p.v;
    R1.p.v = C1.p.v;
    R4.p.v = C1.p.v;
    C2.n.i+R3.n.i = 0;
    R3.n.v = C2.n.v;
end wien;
```

BUPT