Chapter 10
SystemC AMS Power Electronic Modelling with Ideal Instantaneous Switches

Leandro Gil and Martin Radetzki

Abstract Ideal instantaneous switches are a useful behaviour abstraction technique for modelling semiconductor components in power system development. This behaviour abstraction allows fast and robust simulations of sophisticated power systems. In this paper, we present a SystemC AMS extension that supports ideal switches modelling and simulation. Using this extension, large externally and internally controlled electrical linear networks can be integrated into system level models for design and verification purposes. To validate our implementation, we modelled and simulated a complex high voltage power converter for medical applications. The results demonstrate the robustness and accuracy of our SystemC AMS extension.

10.1 Introduction

System level design and verification of analogue and mixed-signal hardware and software requires a hierarchical approach that uses different levels of abstraction. SystemC is a system level design language (SLDL) what focuses on system architecture design for large systems. It provides an open environment for consistent and efficient modelling and simulation of complex heterogeneous systems.

Based on a set of C++ classes and methods, the structure and the behaviour of hardware and software systems can be described from abstract specifications to register transfer level (RTL).

Analogue circuit modelling at higher levels of abstraction is gaining importance in facilitating high-performance system-level simulations. In order to support complex System-on-Chip (SoC) design SystemC was extended for analogue and mixed signal aspects. The current SystemC AMS standard includes models of computation (MoCs) for continuous and discrete time data flow modelling as well as conservative behaviour descriptions for electrical network modelling. These

L. Gil (✉) • M. Radetzki
Embedded Systems Department, University of Stuttgart, Pfaffenwaldring 5b,
70569 Stuttgart, Germany
e-mail: leandro.gil@informatik.uni-stuttgart.de; martin.radetzki@informatik.uni-stuttgart.de

F. Oppenheimer, J.L. Medina Pasaje (eds.), *Languages, Design Methods, and Tools for Electronic System Design*, Lecture Notes in Electrical Engineering 361,
DOI 10.1007/978-3-319-24457-0_10

abstraction methodologies provide enough facilities to support system description and simulation for a wide range of applications, especially for communication systems.

For applications requiring large signal behaviour and switching mode operation, such as driver stages with pulse width modulation, the existing MoCs do not achieve the necessary accuracy [18] or simulation performance [10, 17]. To overcome these limitations, we propose a SystemC AMS extension for power electronic modelling that relies on ideal instantaneous switches. This extension provides primitives for modelling internally and externally controlled switched networks and enables fast, robust and accurate simulations.

The presented approach implements new primitives for semiconductor modelling and reduces the size of system equations by exploiting the properties of ideal switched electrical networks. Additionally, a set of libraries are provided for validating the power circuit functionality in Simulink [8]. Electrical circuits described using SystemC AMS syntax can then be embedded in user code and thus integrated smoothly into Simulink models. The presented methodology does not intend to replace specialized analogue simulators such as Simulink/PLECS or SPICE. These programs are advantageous because they provide a set of specialized libraries and development tools to support several applications and they are normally faster or more accurate when systems with one large analogue part are simulated.

In Sect. 10.2, we outline previously implemented SystemC extensions supporting electrical network modelling and investigate similar modelling approaches for continuous time simulators. Section 10.3 introduces the modelling of switched networks. Using a simple example, some limitations of the current SystemC AMS standard for power electronic modelling are then illustrated in Sect. 10.4. The proposed model of computation and its computational implementation are described in the Sects. 10.5 and 10.6. Finally, we demonstrate the practical applicability of the proposed extension using a sophisticated industrial example in Sect. 10.7.

10.2 Related Work

10.2.1 SystemC Extensions Supporting Electrical Networks

Al-Junaid and Kazmierski presented a SystemC extension named SEAMS using a general-purpose analogue solver [1–3]. In order to provide modelling capabilities for general, mixed-mode systems with digital and non-linear analogue behaviour, a variety of abstraction levels, from system level to circuit level were proposed in this work. The described language constructs support analogue system variables, analogue components and user defined ordinary differential and algebraic equations. C++ classes for electrical nodes and primitive analogue components such as resistor, capacitor, inductor, diode and various types of sources have been implemented. An electrical circuit can be constructed by declaring system variables of type node and analogue components. At the matrix build time, the build functions of all components in the *net list* are invoked.

The analogue equation set formulation relies on the modified nodal analysis (MNA). The analogue kernel invokes the build function once before simulation start. For synchronization between SystemC ports and their corresponding values in the analogue solver, interfacing components are provided. The SystemC kernel was extended to invoke and synchronize the analogue solver in each simulation cycle by applying a lock-step method. A boost power converter was chosen as example of non-trivial analogue and digital interaction. Any numerical difficulty was reported. In a second implementation, named SystemC-A, the handling of extremely small and zero time steps was incorporated to deal with simulation issues.

Vachoux, Grimm and Einwich presented the core elements of SystemC AMS in [19] and [11]. To fill the gap in heterogeneous SoC modelling and simulation, the SystemC extension for analogue and mixed-signal systems was focused on signal processing, RF/wireless and automotive applications. It includes features for modelling linear dynamic continuous time systems and linear networks. In order to support true object oriented model refinement, from abstract specifications to detailed implementations, generalized signals and channels were defined. Using a static dataflow scheduler (with fixed time steps in the first release) synchronization between continuous-time and discrete event model parts was achieved. In a first approach continuous-time descriptions were embedded into discrete-event modules as a cluster of dataflow components.

SystemC AMS is structured using a layered approach. New continuous time models of computation can be added utilizing the descriptive methods provided by the user view layer. Different solver implementations are possible at the solver layer. Finally, the synchronization layer implements a generic mechanism to interface continuous-time solvers and discrete event parts.

To enable the modelling of analogue, non-linear parts of cyber-physical systems at higher levels of abstraction, Uhle and Einwich proposed in [18] a new model of computation for SystemC AMS with similar features like in VHDL-AMS or VerilogAMS. It allows the modelling and simulation of nonlinear networks. The proposed extension integrates smoothly into the existing SystemC AMS language architecture.

The previous three approaches implement models of computation for a wide range of applications. Although electrical network modelling is supported by all theses extensions, they don't provide the necessary accuracy or simulation performance for the modelling of power electronic systems at high abstraction levels. SystemC AMS allows rapid electrical network simulations, but it does not offer primitives for the modelling of internally controlled switches. The applied numerical integration method limits the type of circuit that can be simulated. The proposed extension for nonlinear continuous behaviour leads to sophisticated models and slow simulations. It restricts the practical application to small power circuits.

In [13], Grimm et al. presented a novel approach to enable fast simulations of analogue power drivers. The C++ behavioural models can be easily integrated in SystemC. The underlying method utilizes pre-solved parameterized differential equations. The dominant cycle of the network is determined using Dijkstra's

algorithm. Because, in this approach, analogue models cannot cause discrete events, only a single topological change is permitted at each switching instant. It is not possible to simulate internally controlled switches such as diodes.

10.2.2 Electrical Networks with Ideal Switches

Bedrosian and Vlach presented in [6, 7] a model of computation for time domain analysis of networks with internally controlled ideal switches. In this work the network equations for each topology are generated with a two-graph MNA technique. In order to determine the correct topology after switching, impulsive voltages and currents are considered at the switching instants, in conjunction with the initial conditions. An accurate handling of voltage and current impulses at the switching instants is carried out by splitting the circuit response into a non-impulsive and an impulsive component. The impulsive response part is calculated using a computational method based on inverse Laplace transformation. To find a valid topology after switching several topological changes are allowed. This general analysis method is suitable for any internally controlled switched network (see Sect. 10.3). The computational approach was implemented in a circuit simulator named SWANN.

Massarini, Reggiani and Kazimierczuk proposed a method for large-signal time-domain analysis of switched networks based on a state variable approach in [12–14]. In their work, the network equations are represented with a reduced tableau matrix. An efficient algorithm for the systematic formulation of state equations and output equations for linear active networks was developed. Every switching element is also modelled as an ideal switch. The evolution of the network is represented by a sequence of linear circuit topologies. For each topology, the state equations are systematically obtained through a simple interchange of columns. In order to find the correct topology after switching, a logical representation of impulsive quantities is introduced in the analysis. Using the presented state space description a method to predict possibly impulsive transitions was presented. The impulse analysis is performed only for a limited number of transitions. Switched networks consisting of linear elements and both externally and internally controlled switches were simulated.

Based on the previous approaches Allmeling and Hammer developed a toolbox, PLECS, for simulation of power electronic circuits under Simulink (See [4, 5, 15, 16]). It exploits the features offered in the Simulink environment, allowing the simulation of large systems containing both electrical circuits and sophisticated controllers. A state-space circuit formulation is also applied in this work. The independent mesh and node equations are, however, obtained using MNA. The automatically generated equation system, describing the circuit, is then reduced by elimination of dependent variables. In order to derive the state matrices for a specific topology, the system variables are ordered into: state derivatives, output variables, undetermined switch variables, state variables and input variables. Using Gauss

Jordan elimination with partial pivoting, the generic equation system is transformed into an upper triangular matrix. The state-space equations are then embedded in Simulink by means of user code in C. By analysing the system outputs and the control inputs a switch manager determines the circuit topology. In addition to the non-impulsive part of the system output, an output impulse-multiplier, which is implicitly associated with a voltage or current impulse, is determined after switching. In order to hit the exact switching point in time, Simulink's zero crossing detector is utilized.

Unfortunately, PLECS does not provide a toolbox for discrete event simulators such as Verilog or SystemC. In order to carry out power circuit simulations in SystemC, we exploit the generic architecture of SystemC AMS to develop a model of computation supporting internally and externally controlled ideal switches. A special solver enabling instantaneous switching and accurate results for discrete time simulation is presented in this work.

10.3 Power Electronic Modelling

10.3.1 Abstraction of Power Electronic Circuits

Depending on design and verification goals, different abstraction levels are commonly used to model power electronic circuits:

1. Transfer function:
   For controller design a mathematical description of the system in form of a transfer function is commonly used. The electric circuit is considered as linear causal system. It is only valid for small signal behaviour and no switching response can be analysed (no harmonics).
2. Electrical network with ideal components:
   For circuit design and controller verification tasks a time domain mathematical description of the system is required. The electric circuit is represented using linear components and switches. It is valid for large signal behaviour and applied to evaluate the overall system performance. Voltage and current waveforms of different system parts are analysed.
3. Electrical network with detailed components:
   At the last design stages, a detailed mathematical model of circuit components, including manufacturer specific characteristics, is required for the choice of components. Parasitic effects, switching transitions and component stress are considered during the analysis. Usually, non-linear component behaviour needs to be considered. SystemC AMS provides MoCs to describe analogue systems as transfer function or electrical network. Circuit specific characteristics cannot be adequately analysed. A specialized circuit simulator such as SPICE or SABER is required for such tasks.

10.3.2 Classes of Switched Networks

Taking into consideration the network components and topology, according to [7] switched networks can be classified into:

1. Externally controlled switched networks:
   In this type of electrical networks the state of the switches does not depend on the network response. All switches are controlled by external signals. It leads to "forced commutations".
2. Internally controlled switched networks:
   In this type of electrical networks one or more switches are controlled by network voltages and/or currents. The switching time between different topologies depends on the state of network components. It leads to "natural (and forced) commutations".

SystemC AMS does not include primitives for internally controlled switches. They can be modelled by implementing their control logic as separated module. Alternatively, user primitives can be developed from basic SystemC AMS classes. Additionally, the handling of multiple simultaneous switching requires an appropriated extension, as will be explained in the next chapter.

10.4 Limitations of Modelling and Simulating Power Electronics in SystemC AMS

Power electronic circuits commonly consist of linear components and one or more semiconductor switches. These elements are already included in the ELN primitives. The current computational approach of this MoC provides very good results for the applications considered in the SystemC AMS specification, namely signal processing, RF/wireless and automotive [19].

As mentioned in [9], there are some limitations regarding the simulation of electrical linear networks. Some networks can be described using ELN primitives but the resulting differential equation system cannot be solved.

The Buck converter circuit shown in Fig. 10.1 wasutilized in [4, 5] to explain ideal switching modelling. It is useful to illustrate current SystemC AMS limitations for modelling and simulation of power electronic models. Figure 10.2 shows the ELN code of the Buck converter circuit.

Figure 10.3 shows the ELN code of the diode. It is modelled using a resistive switch. Additional primitives for voltage and current monitoring (V_m and A_m) are required to represent the diode characteristics. The diode control logic is implemented using a simple discrete event module (Fig. 10.4).

When the presented model is simulated, an error message is displayed in the system console (see Fig. 10.5). The SystemC AMS solver cannot initialize the underlying equation system.

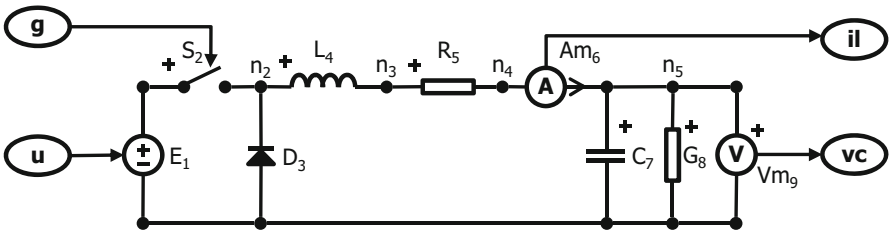

Fig. 10.1 Buck converter schematic

```
namespace sca_eln_mdl
{
    using namespace sca_eln;
    SC_MODULE(Circuit)
    {
        sc_core::sc_in<bool> g;
        sc_core::sc_in<double> u;
        sc_core::sc_out<double> i_l;
        sc_core::sc_out<double> v_c;

        sca_r R;    sca_l L;    sca_c C;    sca_r G;
        sca_de::sca_vsource E;  sca_de::sca_isink Am; sca_de::sca_vsink Vm;
        sca_de::sca_switch S; sca_de::sca_diode D;

        Circuit(sc_core::sc_module_name name):
            g("g"), u("u"), i_l("i_l"),
            R("R", 0.05), L("L",0.1), C("C",0.1), G("G",0.1),
            E("E",1), Am("Am"),Vm("Vm"),
            S("S"), D("D"),
            gnd("gnd"),n1("n1"),n2("n2"),n3("n3"),n4("n4"),n5("n5")
        {
            sc_core::sc_time time_step = sc_core::sc_time(1,sc_core::SC_US);
            E.p(n1); E.n(gnd); E.inp(u);        E.set_timestep(time_step);
            S.p(n1); S.n(n2); S.ctrl(g);        S.set_timestep(time_step);
            D.p(gnd); D.n(n2);                  D.set_timestep(time_step);
            L.p(n2); L.n(n3);R.p(n3); R.n(n4);
            Am.p(n4); Am.n(n5); Am.outp(i_l);   Am.set_timestep(time_step);
            C.p(n5); C.n(gnd); G.p(n5); G.n(gnd);
            Vm.p(n5); Vm.n(gnd); Vm.outp(v_c);  Vm.set_timestep(time_step);
        }
        private:
        sca_node_ref gnd;
        sca_node     n1, n2, n3, n4, n5;
    };
}
```

Fig. 10.2 SystemC AMS ELN Buck converter implementation

In order to keep the size of resulting system matrix constant, switches are described in SystemC AMS by a variable resistance. Very small and very large resistance values are respectively utilized to represent the ON and OFF switch states. This allows the exploitation of the fast and generic linear solver developed for continuous time modelling. No solver extension is required to compute the linear network solution after a topology change.

The circuit response usually does not change considerably if switches are modelled in this way; however, it often leads to non-solvable equation systems.

```
namespace sca_eln
{
    namespace sca_de
    {
        using namespace sca_eln::sca_de;
        SC_MODULE(sca_diode)
        {
            sca_eln::sca_terminal p;
            sca_eln::sca_terminal n;

            private:
            bool state; double r_on; double r_off;
            sca_rswitch D;  sca_isink Am; sca_vsink Vm;
            sca_diode_control ctrl;

            public:
            sca_diode(sc_core::sc_module_name,double _r_on = 0.0,
            double _r_off = sca_util::SCA_INFINITY):
            p("p"),n("n"),r_on(_r_on),r_off(_r_off), D("D",_r_on,_r_off),
            Am("Am"),Vm("Vm"), ctrl("ctrl"), n1("n1")
            {
                Am.p(p); Am.n(n1); Am.outp(i_d);        // Current measure
                D.p(n1); D.n(n); D.ctrl(s_d);           // Resistive switch
                Vm.p(p); Vm.n(n); Vm.outp(v_d);         // Voltage measure
                ctrl.v(v_d); ctrl.i(i_d); ctrl.s(s_d);  // Diode control module
            }

            void set_timestep(int time_step_value, sc_core::sc_time_unit time_step_unit)
            {
                set_timestep(sc_core::sc_time(time_step_value,time_step_unit));
            }

            void set_timestep(sc_core::sc_time _time_step)
            {
                time_step = _time_step;
                Am.set_timestep(time_step);
                D.set_timestep(time_step);
                Vm.set_timestep(time_step);
                ctrl.set_timestep(time_step);
            }

            void set_ron (double r_value){D.ron  = r_value;}
            void set_roff(double r_value){D.roff = r_value;}

            private:
            sc_signal<bool>   s_d;
            sc_signal<double> v_d, i_d;
            sca_eln::sca_node n1;
            sc_core::sc_time time_step;
        };
    }   // namespace sca_de
}       // namespace sca_eln
```

Fig. 10.3 SystemC diode implementation

By setting the diode switch off state resistance to a smaller value (1e9), the solvability problem can be eliminated.

The power circuit controller is also implemented using a discrete event module as shown in Fig. 10.6. The inductor current is maintained inside a defined range by open and closing the switch S. When the maximal inductor current is achieved, the controller opens the switch and a new solvability error occurs during simulation, as shown in Fig. 10.7. At the switching instant, the inductor current is interrupted causing a voltage impulse. This turns the diode ON. Due to the very small diode ON resistance, the resulting equation system cannot be solved. Changing diode ON resistance to an appropriated value (1e-6), the system equations can be numerically solved.

```
namespace sca_eln
{
    namespace sca_de
    {
        SC_MODULE(sca_diode_control)
        {
            public:
            sc_core::sc_in<double> v;
            sc_core::sc_in<double> i;
            sc_core::sc_out<bool> s;

            void event_control(void)
            {
                double voltage = v.read();
                double current = i.read();

                if((state==true)&&(current <= 0.0))
                {
                    state = false; s.write(false);
                }
                if((state==false)&&(voltage > 0.0))
                {
                    state = true; s.write(true);
                }
            }
            SC_HAS_PROCESS(sca_diode_control);
            sca_diode_control(sc_core::sc_module_name): v("v"),i("i")
            {
                state = false;
                SC_METHOD(event_control); sensitive << v << i;
            }
            private:
            bool state;
        };
    }   // namespace sca_de
}       // namespace sca_eln
```

Fig. 10.4 SystemC diode control logic implementation

```
Error: SystemC-AMS: Initialization equation system failed in
sca_linear_solver_0: 1 the error is in the following net (max. 50):
circuit.R
circuit.L
circuit.S
circuit.E
circuit.D.S
circuit.D.Am
circuit.D.Vm
circuit.Am
circuit.C
circuit.G
circuit.Vm

The error is may  be near:
circuit.L  and
circuit.n2
```

Fig. 10.5 SystemC AMS error message during initialization

```
namespace sca_eln_mdl
{
    SC_MODULE(Control)
    {
        public:
        sc_core::sc_in<double> i_l;
        sc_core::sc_out<bool> g;

        void set_reference(double _reference){reference=_reference;}
        void set_p_threshold(double _threshold){p_threshold=_threshold;}
        void set_n_threshold(double _threshold){n_threshold=_threshold;}

        void event_control(void)
        {
            double current = i_l.read();

            if(((current - reference) >= p_threshold)&&(state==true))
            {
                state = false;  g.write(false);
            }
            if(((current - reference) <= n_threshold)&&(state==false))
            {
                state = true;  g.write(true);
            }
        }
        SC_HAS_PROCESS(Control);

        Control(sc_core::sc_module_name, double _reference = 1.0,
        double _p_threshold = 0.5, double _n_threshold = -0.5):
        i_l("i_l"),g("g"), reference(_reference),
        p_threshold(_p_threshold), n_threshold(_n_threshold)
        {
            state = true;
            g.initialize(true);
            SC_METHOD(event_control); sensitive << i_l;
        }

        private:
        bool    state;          // Relay state
        double  reference;      // Reference value for control
        double  p_threshold;    // Upper threshold
        double  n_threshold;    // Lower threshold
    };
}   // namespace sca_eln_mdl
```

Fig. 10.6 SystemC Buck converter controller implementation

The obtained simulation results are however not correct. As shown in Fig. 10.8, inductor current becomes zero after switching. Thereby, the diode does not remain turned ON and the controller closes the switch again to reach the desired inductor current. The reason for this behaviour is that SystemC AMS solver allows only one switching event at a given point in time. This restriction limits the type of circuits that may be simulated using SystemC AMS. The simulations results using multiple instantaneous switching are showed in Sect. 10.6 (Fig. 10.10).

```
Error: SystemC-AMS: Reinitialization failed in sca_linear_solver_0: 1
during reinitialization equation system at 21217 us (dt = 1e-06).
the error is in the following net (max. 50):
circuit.R
circuit.L
circuit.S
circuit.E
circuit.D.S
circuit.D.Am
circuit.D.Vm
circuit.Am
circuit.C
circuit.G
circuit.Vm

Parameters of the following modules changed for the current time step:
circuit.S changed to 0
circuit.D.S changed to 0

The error is may  be near:
circuit.D.Am
```

Fig. 10.7 SystemC AMS error message during simulation

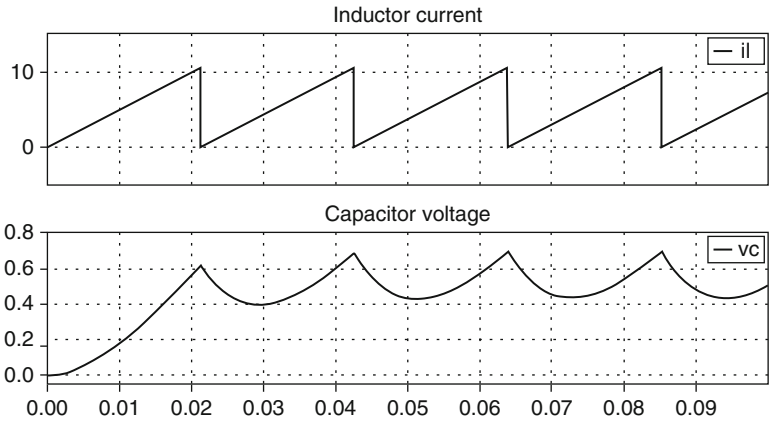

Fig. 10.8 Buck converter simulation results with SystemC AMS ELN

10.5 Modelling and Simulating Power Electronics Using Ideal Switches

The use of ideal instantaneous switches is often an appropriate behaviour abstraction to model power electronic semiconductors because the processes during switching are not important when power electronic circuits are simulated for system performance evaluation. An ideal or perfect switch has zero resistance when ON, zero admittance when OFF, and switches between both states in zero time. Representing

switches in this way leads to more robust and faster simulations. On the other hand, circuit state inconsistencies after topology change need to be properly handled during simulation.

10.5.1 Computational Advantages of Ideal Switching Modelling

1. Simple numerical integration algorithm:
   The numerical integration algorithm is applied during simulation to compute the electrical circuit behaviour after a circuit topology was defined. The method and parameters required to solve the associated differential equations depend mainly on current circuit characteristics. They may change after each switching event. Using linear electrical elements, an accurate representation of power electronic circuits is possible. Linear differential algebraic equations can be numerically solved using very simple integration techniques such as forward and backward Euler or the trapezoidal rule [20].
2. Short time switching process:
   Because ideal switches change their state in zero time, only two integration steps are necessary to handle the discontinuities at each switching event. The numerical integration algorithm is applied again to compute the system state after switching.
3. Reduced equation system:
   Nodes connecting ideal switches can be contracted into a single node when ideal switches are ON. Thus, the size of the network matrices and hence the computation time can be reduced.
4. Less solvability problems:
   Because ideal switches do not have very small or very large parameter values, the resulting circuit equation system is normally non-stiff which leads to stable simulations. Additional passive components (snubbers) to make the simulation converge are rarely necessary and therefore no parameter tuning is required.
5. More accurate models:
   Due to the large signal behaviour of power circuit models, avoiding additional resistances for solvability increases considerably the simulation accuracy. Furthermore, the additional parameters of non-ideal switches must be chosen according to the circuit being analysed. It requires a good understanding of the circuit's operation. Ideal switches eliminate the trade-off between accuracy and stability.
6. Faster simulations:
   Ideal switches do not affect system response. In opposite, the additional components of non-ideal switches introduced for simulation stability may considerably increase the simulation time, in particular, if their parameters are not properly chosen.

10.5.2 Computational Requirements for Ideal Switching Modelling

An ideal switch represents a short circuit when it is closed and an open circuit when it is open. Depending on the resulting topology after commutation, different circuit or state inconsistencies can be caused by:

- floating nodes
- short-circuited voltage sources
- open-circuited current sources
- short-circuited capacitors
- open-circuited inductors
- different initial conditions of circuit elements

If one or more inconsistencies are present in the circuit topology after switching, the resulting differential and algebraic equation system cannot be numerically solved. Using circuit analysis such network inconsistencies must then be found and removed from the system description matrices.

Additionally, the network response may be discontinuous at the switching instants, if inconsistencies are present. It means that voltage or current impulses may be generated by the state change. They can be also produced by inconsistent initial conditions [21]. Thus, impulsive currents or voltages need to be accurately recognized and appropriately handled for determining the correct topology after switching. Internally controlled switches may change their state if a current or voltage impulse is applied to them. As described in the related work, there are several methods to cope with voltage and current impulses in switched networks.

10.6 Electrical Piece-Wise-Linear Networks (EPN)

As presented in [19], the language architecture of SystemC AMS standard is defined in an extensible way. New models of computation can be defined and seamlessly integrated by using base classes for signals, ports and modules.

10.6.1 Syntax and Primitives

The proposed extension of SystemC AMS for modelling electrical circuits with ideal switches follows the same syntax as the currently available MoC ELN (Electrical Linear Networks). Thus, existing SystemC AMS models can be compiled and simulated with only minimal code modifications.

In order to support the specific features of ideal switching modelling with natural commutation, primitives for electrical elements were implemented using the

attributes and methods of the new solver class. We named our model of computation "Electrical Piece-wise-linear Networks" and assigned the namespace sca_epn to the new types and classes. All primitives are derived from the base class sca_module.

In order to utilize the built-in binding mechanism of SystemC, terminals are instances of a type derived from sca_port and nodes are instances of a type derived from sca_interface. Circuit elements are also interconnected by binding terminals to nodes. Primitives for modelling externally and internally controlled ideal switches (epn_switch and epn_diode) are provided to enable the novel language capabilities.

10.6.2 Network Equations Formulation

Similar to the current SystemC AMS implementation, we obtain circuit equations by applying the rules of the MNA. The nodal equation set-up is carried out according to the following expressions:

$$[Y_n B_b]\,[V_n] = [J_n] \tag{10.1}$$

$$[B_n Z_b]\,[I_b] = [E_b] \tag{10.2}$$

where Y_n is the nodal admittance matrix, Z_b is the branch impedance matrix, B_b is the branch incidence matrix, B_n is the nodal loopset matrix, J_n represents the equivalent nodal current sources, E_b represents the equivalent branch voltage sources, V_n is the nodal voltage vector and I_b is the branch current vector.

Matrix stamps for this equation partition are formulated as described in [20]. This leads to less equations than the current SystemC AMS implementation and is more appropriate for power circuit simulation consisting of linear elements.

During the SystemC AMS initialization phase, the matrix_stamp function is called for each instantiated circuit element. The previously defined matrices, Y_n, B_b, B_n and Z_b, are then created and used to compute the system matrix by applying the numerical integration algorithm as described in [20]. Linear multistep methods (LMS) are currently provided.

The value of the system matrix coefficients depends on the state of the internally and externally controlled switches. The resulting circuit topology is then analysed considering the criteria specified in the previous chapter.

10.6.3 Topology Analysis of Switched Electrical Networks

To carry out an efficient topology analysis of switched electrical networks we extended graph theory representation methods for the detection of network inconsistencies. A network graph considers the network elements as terminal components. It describes the connection between network elements, capturing the properties of

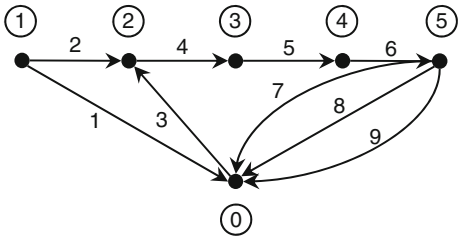

Fig. 10.9 Graph representation of the Buck converter circuit

Table 10.1 Buck converter representation using a table

Edge	1	2	3	4	5	6	7	8	9
Type	E	S	D	L	R	A_m	C	G	V_m
N_p	1	1	0	2	3	4	5	5	5
N_n	0	2	2	3	4	5	0	0	0
Z_i	0	1	2	0	0	0	0	0	0

the network in a natural way. Network graphs can be efficiently implemented on the computer in form of a simple table.

1. Network representation using a directed graph
   For the construction of the directed graph network representation, each two-terminal element of the network is numbered and replaced in the circuit by a line called edge. An orientation corresponding with the assigned current flow direction is associated with each edge in the graph. For passive elements, the node from which the current flows is the positive terminal. For current sources, the direction of the current is defined by its symbol. For voltage sources, the direction of the current flow is from the positive to the negative terminal. The vertex of the graph represents the nodes of the network. They are numbered assigning zero to the ground. The node numbers in the graph are placed within circles to distinguish them from the edge numbers. Figure 10.9 shows the graph representation of the Buck converter circuit.
2. Network representation using a table
   For the construction of table representing an electrical network, each two-terminal element of the network adds a new column to the table. The edge, the type, the positive and the negative node of the each network element are described in the corresponding table row. For switched network analysis we add a row containing the state variable number of switching elements.

Table 10.1 buck converter circuit. N_p, N_n and Z_i represent the positive node number, the negative node number and the discrete state variable number respectively.

In order to handle two ports elements by a single graph, the input and the output terminals are represented as separated edges (columns in the table representation form).

Table 10.2 Reduced Buck converter representation using a table

Edge	1	2	3	4-5-6	7-8-9
Type	E	S	D	L-R-A_m	C-G-V_m
N_p	1	1	0	2	5
N_n	0	2	2	5	0
Z_i	0	1	2	0	0

10.6.3.1 Topology Analysis Using a Network Graph

The graph representation of a switched electrical network can be utilized to reduce the number of equations and predict inconsistent equations. The following information about the network nodes and branches can be obtained:

- connected elements which can be reduced to a single element
- floating nodes and its switching dependencies
- floating network elements
- complementary network branches (i.e. diode bridge)

The following steps are carried out during the graph based topology analysis of a switched network:

1. Group serial connected impedances (R-L-A_m elements which may be represented as single impedance or current branch).
2. Group parallel connected admittances (G-C-V_m elements which may be represented as single admittance).
3. Identify floating nodes if all switches are open.
4. Identify floating branches if all switches are open.

A floating node is a node to which only one element is connected. The current through a branch connected to a floating node is zero. A floating branch is a branch terminated with floating nodes. It is not possible to compute the voltage across its nodes.

Applying the steps 1–2, a reduced circuit table is obtained (Table 10.2) for the Buck converter circuit. The elements L, R and A_m have the same current and can be grouped into a single edge. The parallel admittance branches G-C have the same voltage applied to its nodes and can also be grouped into a single edge. The output voltage V_m is the same as the voltage across G-C.

If all switches are open (step 3), there two floating nodes in the circuit (node 1 and node 2). Because all circuit branches are connected to ground when all switches are open, does not exist any floating branch which need to be eliminated from the network equation system (step 4).

10.6.3.2 Voltage and Current Graphs Analysis

An edge in an electrical network graph simultaneously represents the current through the element and the voltage across the network element. For some elements of the network, one of the constitutive variables may be zero.

For example, the current through an ideal switch or the voltage across its terminal is zero when it is open or closed respectively (complementary behaviour). For some other elements, one of the constitutive variables is not necessary for the solution of the resulting equation system and it is also not of interest, such as the current through a voltage source and the voltage across the terminal of a current source. Using separated graphs to represent the network voltages and currents, this redundancy can be eliminated (see [20] for more details).

Extended current and voltage graphs can be utilized for the prediction of network topologies producing impulses.

1. Topology analysis using the current graph (I-graph)
   The current graph can be utilized to get the following information about the switched electrical network:
   - open circuited branches which may generate voltage impulses (branches containing current sources and inductors)
   - switching dependencies for such open circuited branches
   - voltage impulses acting on internally controlled switches
2. Topology analysis using the voltage graph (V-graph)

The voltage graph can be utilized to get the following information about the switched electrical network:

- short circuited elements branches which may generate current impulses (voltage sources and capacitors)
- switching dependencies for such short circuited elements
- current impulses acting on internally controlled switches

10.6.3.3 Current Graph Topology Analysis Steps

The following steps are carried out during the current graph based topology analysis of a switched network:

1. Collapse the nodes of the network elements that are not of interest for network equation formulation (voltage sources E and sinks V_m).
2. Collapse the nodes of the network elements that are not of interest for impulse analysis (only switches and branches containing inductors remain)
3. Identify open circuited branches containing current sources and inductors (branches which generate voltage impulses).
4. Identify the switching state dependencies for branches generating voltage impulses.
5. Identify voltage impulses acting on internally controlled switches and their polarity.

Applying the steps 1–2, a reduced circuit table is obtained (Table 10.3) for the Buck converter circuit. If all switches are open, the branch 4-5-6 is not closed.

Table 10.3 Reduced Buck converter representation table for current analysis

Edge	2	3	4-5-6
Type	S	D	L-R-A_m
N_p	0	0	2
N_n	2	2	0
Z_i	1	2	0

Table 10.4 Reduced Buck converter representation table for voltage analysis

Edge	1	2	3	4-5	7-8-9
Type	E	S	D	L-R	C-G-V_m
N_p	0	0	0	0	5
N_n	0	0	0	5	0
Z_i	0	1	2	0	0

Using the reduced table, it is not difficult to find the states producing voltage impulses. In this case, if the switch S (state z_1) and the diode D (state z_2) are open, a voltage may impulse occur (the inductor current must be greater than zero). Additionally, we can find out that the produced voltage impulse, when the switches are open, is applied to the diode D with positive polarity.

10.6.3.4 Voltage Graph Topology Analysis Steps

The following steps are carried out during the voltage graph based topology analysis of a switched network:

1. Collapse the nodes of the network elements that are not of interest (current sources and sinks).
2. Identify short circuited voltage sources and capacitors if all switches are closed.
3. Identify the switching state dependencies for branches generating current impulses.
4. Identify current impulses acting on internally controlled switches and their polarity.

Applying the step 1, a reduced circuit table is obtained (Table 10.4). The voltage through A_m is zero, and its nodes can be collapsed. After collapsing all switch nodes, the current impulses generated by short circuited voltage sources and capacitors can be founded (step 2). In our example, the battery E is short circuited when the switch S and the diode D are simultaneously closed (step 3). The current impulse is then applied to the diode D and the corresponding current flow direction is determined by the polarity of the battery (step 4).

10.6.4 Solver Implementation for Switches Networks

The topology analysis described in the previous section is carried out by the solver before the simulation starts. It provides the necessary information for the computation of the current topology equations. If circuit inconsistencies are detected during the equation setup; the rows and columns associated with non-valid branches and floating nodes are shifted to the outside of the resulting system matrix. The solver variables indicating the number of system nodes and branches are respectively modified.

The system matrix creation and analysis is repeated during circuit simulation, when switching conditions are reached or circuit equations modified. In order to improve simulation performance, the system matrix is factorized using LU decomposition. Additionally, the computed topologies can be stored. The number of stored topologies can be limited by the user.

As shown in Fig. 10.10, the solver algorithm calls a list of pre-solve methods at the start of the integration step. They are dynamically registered by the circuit elements and managed by the solver. This SystemC AMS functionality is utilized by our ideal switches to read SystemC ports and update circuit topology if necessary. If state changes are reported, the system matrix is computed again. The pre-solve methods are called again if inconsistencies were detected. Forward and backward substitution is applied at each integration step to obtain the system response.

After one integration step is carried out, the solver calls the post-solve methods. The diode class exploits this mechanism to evaluate switching conditions. It sets a solver flag if their threshold values are reached. The solver creates the system matrix and repeats the integration step, if a natural switch condition was reported. Because ideal diodes modify their state instantaneously, this loop needs to be executed until no more switching conditions are detected. If during the switching process a topology occurs twice, the system is not stable and the simulation is aborted.

10.6.5 Time Step Control

In order to keep the errors caused by the numerical integration small as well as to synchronize continuous and discrete time parts, analogue simulators carry out step size control during simulation. The SystemC kernel employs an entirely different approach to control time advance. It uses events that trigger processes. The solver time control is implemented in SystemC AMS using a spawn process.

Due to the large signal behaviour of power electronic circuits, a very short integration step is often required to minimize numerical integration errors. This can notably decrease the simulation performance. As shown in Fig. 10.10, a step refinement loop was incorporated into the solver algorithm to improve the simulation time and obtain accurate results. A solver variable controlling the loop can be modified by the primitive pre- and post-solve methods. In a first approach this

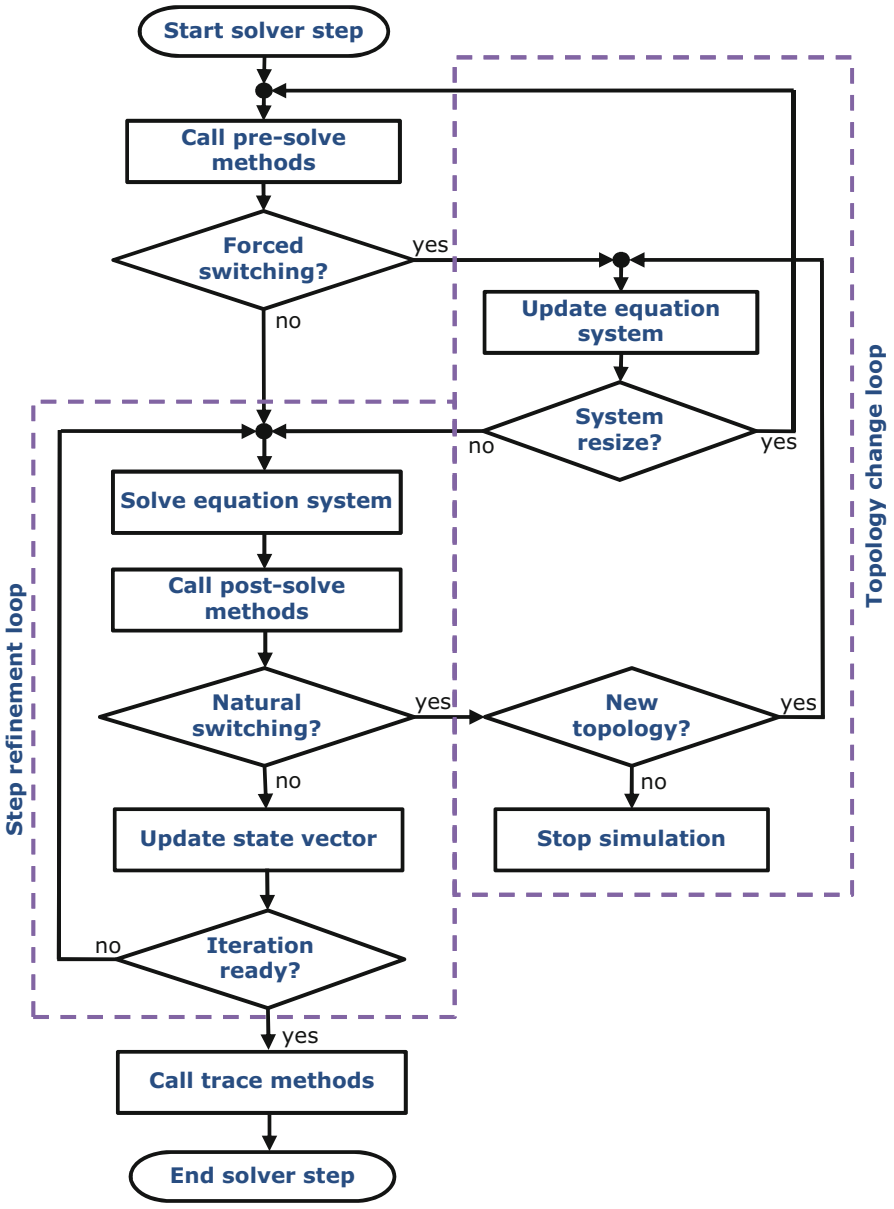

Fig. 10.10 Solver step refinement and topology change loops

simulation parameter is adjusted by the user by calling a solver interface function in the SystemC module constructor.

To split solver interface from step refinement and topology change loops, trace methods were defined and implemented for current and voltage sinks.

10.6.6 Electrical Circuit Integration in Simulink

The control algorithm of the power system is normally implemented as Simulink model. In order to validate the power circuit functionality in Simulink, a set of libraries were created. They enable the smooth integration of electrical circuits, implemented in SystemC, into Simulink models by embedding user code. The SystemC module registration and port binding process is carried out by the Simulink block frame when the model initialization function is called. During simulation input and output signals are writing to and reading from global variables assigned to the SystemC ports. The input and output port order follows SystemC declaration order.

10.7 Experimental Results

10.7.1 Buck Converter Simulation

The buck converter circuit, proposed in Sect. 10.4 to illustrate the current SystemC limitations by simulating internally controlled switched networks, was modelled and simulated using ideal instantaneous switches. As shown in Fig. 10.11, the inductor current does not present discontinuities.

This demonstrates that our approach enables multiple instantaneous switching.

At time 0.021 s, the switch S is open and the impulse generated by the inductor turns the diode ON. At time 0.091 s the switch S is closed and the negative voltage applied to the diode turn it OFF. The short circuit caused by the diode, when the switch is closed is properly handled by our algorithm.

In Table 10.5, the execution time of this model till the switch is opened at 0.021 s is compared for different approaches and data sampling frequency. The EPN MoC compute 10 ms faster, when the solver step size is increased from 1 to 10 μs and the number of step iterations set to 10. Simulink simulates faster when the model is only loaded on memory.

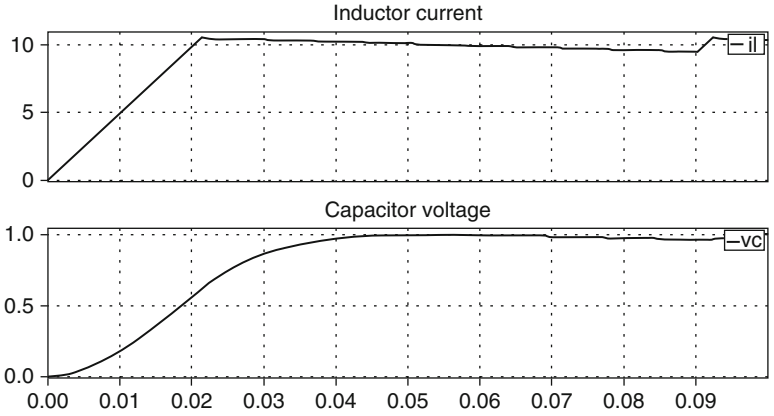

Fig. 10.11 Buck converter simulation results using ideal instantaneous switches

Table 10.5 Buck converter simulation execution time

	Simulation execution time		
Data sampling	SystemC ELN	SystemC EPN	Simulik/PLECS
10 µs	60 ms	40 ms | 50 ms	62 ms | 74 ms
100 µs	50 ms	40 ms | 50 ms	60 ms | 69 ms

10.7.2 High-Voltage Power Converter Simulation

In order to evaluate the practical application of the described SystemC AMS extension for power electronic modelling, a high-voltage power converter used in medical machines has been modelled and simulated. Figure 10.12 shows a block diagram representation of power electronic system. Due to the large number of switches (4 externally controlled and 16 internally controlled), the high frequency operation (10 ns), as well as the very large output signal range (30–120 kV) this switched electrical network provides the required complexity to validate the proposed methodology and its software implementation.

The digital controller was implemented using discrete event modules. It operates in three levels (ON, OFF and CONTROL) to achieve a rapid response. Many signals are monitored to control the output voltage. As shown in Fig. 10.13, there is a strong analogue and digital interaction.

Although circuit parameter values comprise a very large range, solvability problems were only encountered for one topology. The modelling task was considerably simplified by using ideal diode primitives. As expected, the small integration step leads to large simulation times. Simulation results were similar to those obtained with PLECS by using fixed time step.

The simulation accuracy allows an adequately analysis of signal harmonics.

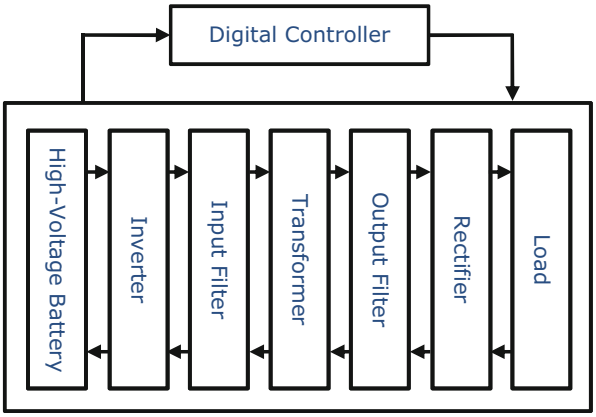

Fig. 10.12 Block diagram of the high-voltage power converter

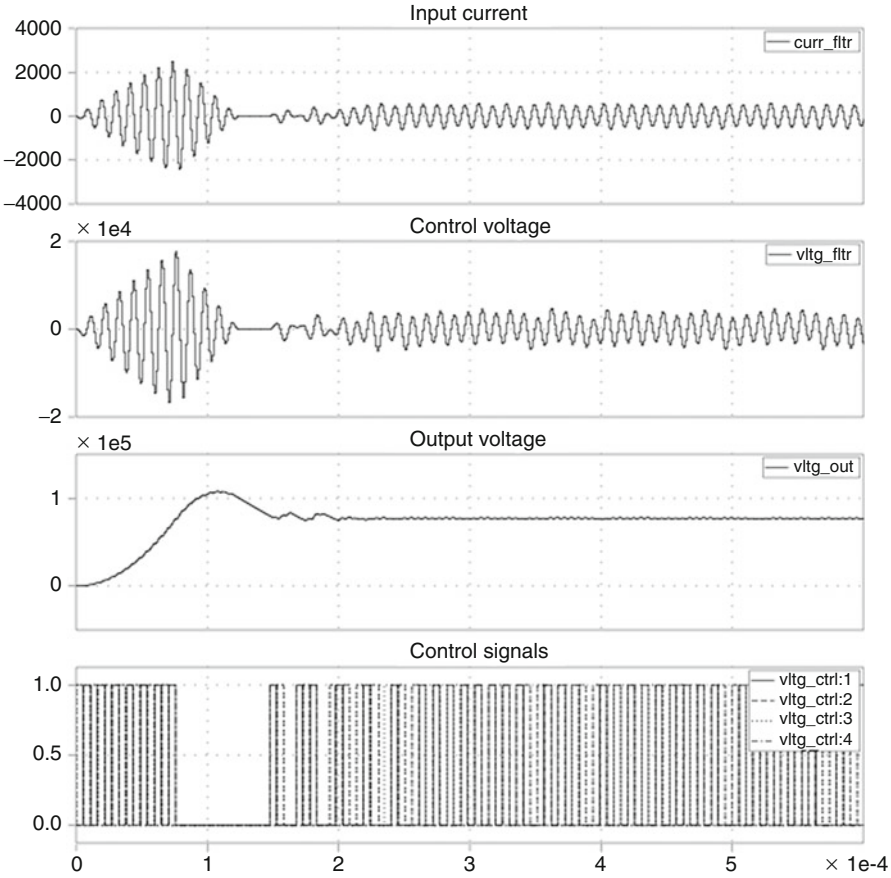

Fig. 10.13 Power converter signals

10.8 Conclusion

In this paper we proposed an extension of SystemC AMS for modelling and simulation of power electronic circuits. It was integrated into the existing language architecture. Our proposed implementation hides switching control details from the models, resulting in a simple and more efficient modelling approach of electrical piece-wise-linear networks. Experiments show that complex internally controlled electrical switched networks are robust and accurately simulated in SystemC. We are encouraged by the obtained results to continue with the development of modelling abstractions for efficient power circuit simulations in SystemC.

In the next step we will improve solver efficiency to carry out large system level simulations. In the future, we want to investigate other network abstractions taking advantage of circuit properties and the inherent repetitive operation mode of power systems.

Acknowledgment The authors would like to thank Philips Research for the contribution with power system models and the German Federal Ministry of Education and Research (BMBF) for financial support of this work within the project POWERBLOCK+ (grant number 16M3200F).

References

1. Al-Junaid, H., Kazmierski, T.: An introduction to modeling embedded analog/mixed-signal systems using SystemC AMS extensions. In: Proceedings of the International Symposium on Circuits and Systems (ISCAS'04), vol. 5, pp. 281–284, May 2004
2. Al-Junaid, H., Kazmierski, T.: An extension to SystemC to allow modelling of analogue and mixed signal systems at different abstraction levels. Conference or Workshop Item (Speech), September 2004
3. Al-Junaid, H., Kazmierski, T.: Analogue and mixed-signal extension to SystemC. In: IEE Proceedings of Circuits, Devices and Systems, pp. 682–690, December 2005
4. Allmeling, J., Hammer, W.: PLECS – Piece-wise Linear Electrical Circuit Simulation for Simulink. In: Proceedings of the IEEE International Conference on Power Electronics and Drive Systems (PEDS '99), vol. 1, pp. 355–360, July 1999
5. Allmeling, J., Hammer, W.: Simulating power electronic systems using ideal instantaneous switches. In: Proceedings of the International Conference on Power Electronics, Intelligent Motion, Power Quality (PCIM Europe '04), vol. 2 pp. 585–590, May 2004
6. Bedrosian, D., Vlach, J.: Time-domain analysis of networks with internally controlled switches. In: Proceedings of the IEEE International Symposium on Circuits and Systems, vol. 2, pp. 846–849, June 1991
7. Bedrosian, D., Vlach, J.: Time-domain analysis of networks with internally controlled switches. IEEE Trans. Circuits Syst. 1 Fundam. Theory Appl. **39**(3) (1992)
8. Bozin, A.: Electrical power systems modeling and simulation using SIMULINK. In: Proceedings of the IEE Colloquium on the Use of Systems Analysis and Modelling Tools: Experiences and Applications, pp. 10/1–10/8, March 1998
9. Einwich, K.: SystemC AMS Extensions–The Language. Tutotial, September 2010
10. Grimm, C., Meise, C., Oehler, P., Waldschmidt, K., Fey, F.: AnalogSL: A library for modeling analog power drivers with C++. In: Proceedings of the Forum on Specification and Design Languages (FDL '01), September 2001

11. Grimm, C., Barnasconi, M., Vachoux, A., Einwich, K.: An introduction to modeling embedded analog/mixed-signal systems using SystemC AMS extensions. Open SystemC Initiative, June 2008
12. Massarini, A., Reggiani, U.: Computer-aided time-domain large-signal analysis with network switches. In: Proceedings of the International Symposium on Industrial Electronics (ISIE '96), vol. 2, pp. 567–572, June 1996
13. Massarini, A., Kazmierczuk, M.K.: A new representation of Dirac impulses in time-domain computer analysis of networks with ideal switches. In: Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS '96.), vol. 1, pp. 565–568, May 1996
14. Massarini, A., Reggiani, U., Kazimierczuk, M.K.: Analysis of networks with ideal switches by state equations. IEEE Trans. Circuits Syst. I Fundam. Theory Appl. **44,** 692–697 (1997)
15. Plexim GmbH: An introduction to solvers. In: Vehicle Power and Propulsion Conference (VPPC), 2011 IEEE International, pp. 1–132, September 2011
16. Plexim GmbH: PLECS User Manual. http://www.plexim.com/files/plecsmanual.pdf
17. Reuther, C., Einwich, K.: A SystemC AMS extension for controlled modules and dynamic step sizes. In: Proceedings of the Forum on Specification and Design Languages (FDL '12), pp. 90–97, September 2012
18. Uhle T., Einwich, K.: A SystemC AMS extension for the simulation of non-linear circuits. SOC Conference (SOCC), 2010 IEEE International, pp. 193–198, September 2010
19. Vachoux, A., Grimm, C., Einwich, K.: Towards analog and mixedsignal SOC design with systemC-AMS. In: Proceedings of the International Conference on Field-Programmable Technology, 2004, pp. 97–102, January 2004.
20. Vlach, J., Singhal, K.: Computer Methods for Circuit Analysis and Design, 2nd edn. Van Nostrand Reinhold, New York (1993)
21. Vlach, J., Opal, A., Wojciechowski, J.: Simulation of networks with inconsistent initial conditions. In: Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS '93), vol. 3, pp. 1627–1630, May 1993