OBJECT-ORIENTED SWITCHING POWER CONVERTER MODELING USING DYMOLA WITH EVENT-HANDLING

John S. Glaser, François E. Cellier, and Arthur F. Witulski
Department of Electrical and Computer Engineering
University of Arizona, Tucson AZ 85721
Email: glaser@hermes.ece.arizona.edu, cellier@ece.arizona.edu, witulski@ece.arizona.edu

ABSTRACT

The Dymola modeling language has been shown to be useful for modeling electrical networks in an object-oriented manner. The ability of Dymola to handle time and state events in an objected-oriented manner also makes it particularly suitable for modeling switching power converters, whose models may generate thousands of events over a simulation run.

The object-oriented nature of Dymola is well suited for modeling complex power supply systems. Circuit designers may develop and test their converter models; systems designers may use these as building blocks in larger systems without needing detailed knowledge of the converter models. This is demonstrated by means of an example wherein a complete converter system is developed beginning from the component level.

Keywords: object-oriented, simulation, Dymola, switching power converter, power supply.

I. INTRODUCTION

The ability of the modeling software Dymola to model events in an object-oriented manner makes it a useful tool for the modeling of switching power converters and power systems. Switching events can be modeled at the electrical component level, along with standard electrical component models. The circuit designer can "wire" these component models together according to the design of his or her particular power converter, in order to develop a converter model. A power systems designer can then connect the circuit designer's converter model with other system components in the development of a complete power system. At each level, the modeling is completely object-oriented. The circuit designer need not know the details of event-handling nor component modeling. Likewise, the power system designer does not care about the detailed workings of the converter models that s/he employs; s/he only cares that the models work. In this paper, the object-oriented modeling of a complete power supply system, from the component to system level, will be demonstrated by means of a practical example.

Simulation of electronic switching power converters is a problematic task for a number of reasons. Such converters operate by using switches to change the configuration of an energy-storage (inductor-capacitor) network at frequencies up to several megahertz, and the time constants of the network are usually one or more orders of magnitude larger than the switching period. Each switching event is, in an ideal sense, a discontinuity. Typical power converter simulations can run for hundreds or thousands of switching cycles, which can create problems for simulation software that does not have explicit mechanism for handling discontinuities (events). Among these problems are long simulation times due to decreased integration time steps in the region of the discontinuity, lack of convergence, and erroneous simulation results.

The standard software for circuit simulation, SPICE (and its derivatives), is not well suited to power converter simulation due to its lack of explicit event handling (Quarles *et al.* 1993). While there exists other simulation software, such as SIMPLIS, which is designed specifically for power converter simulation, this software is unsuitable for more general circuit problems (Lee and Wilson 1992). This paper proposes the use of the general purpose modeling language Dymola in conjunction with the standard simulation language ACSL to simulate switching power converters (Elmqvist 1993). ACSL has both state and time event handling, which makes it well suited for switching converter simulation; however, it is unsuitable for circuit designers since it requires an unfamiliar state-equation model of the converter in question (Mitchell and Gauthier 1991). Furthermore, ACSL is not object-oriented, rendering the development of complete power supply system models difficult. Dymola solves both problems: First, a set of electrical component models can be developed. These components include the standard linear components such as resistors, inductors and capacitors. In addition, simple switch and diode component models are developed that contain the event descriptions. Second, the circuit designer can connect the components together in a netlist format similar to SPICE, and need not worry about simulation events or discontinuities. Such a circuit model may in turn be used as a component in a larger system, in keeping with the object-oriented nature of Dymola. Dymola generates an ACSL program from the model, which can then be compiled and run under ACSL.

II. SWITCH-MODE POWER CONVERTERS

Switch-mode power converters find use in nearly every system that requires electrical power due to their high efficiency and high power density. Some examples are dc-dc converters, ac-dc rectifiers, and dc-ac inverters. This section gives a brief overview of switching power converters. Operation, modeling, and simulation are discussed.

IIA. Switching Power Converter Operation

Switch-mode power converters operate by the periodic storage and release of electrical energy in capacitors and inductors. The flow of energy is controlled by means of switches. These switches are divided into two classes, time-event driven and state-event driven. Time-event driven switches are controlled by some periodic waveform external to the converter network, and are usually implemented with transistors. State-event driven switches are controlled by the state variables of the converter network, i.e., the inductor currents and capacitor voltages; such switches are usually implemented with diodes.

Figure 1 illustrates the common buck-boost converter (a) and its simplest operating mode (b). The converter is controlled by modulating the duty cycle D of a constant-frequency control signal applied to the switch S_1, hence this converter is classified as a pulse-width modulated (PWM) converter. Switch D_1 is a diode, and is controlled by voltage and current waveforms internal to the converter network. The values of L and C are chosen such that the

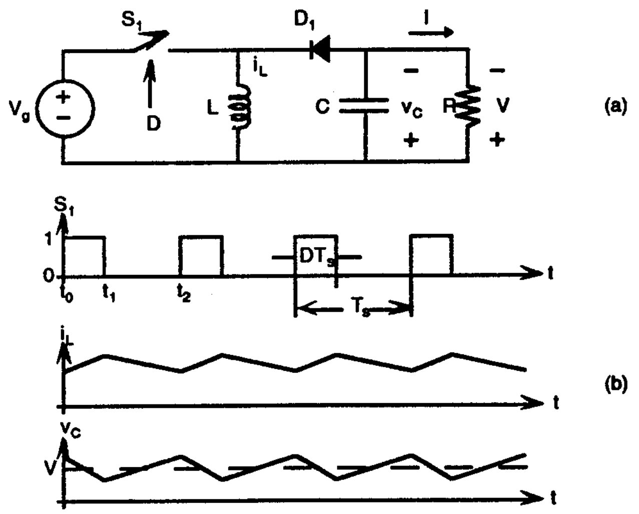

Figure 1. Buck-boost converter (a) and major waveforms (b).

circuit's time constants are large compared to the period of the switching signal applied to S_1. This will cause an average dc output voltage $V = DV_g/(1 - D)$ which will have a small variation superimposed due to the switching behavior of the circuit. The detail behavior of switching power converters can be found in many texts and references, including (Cuk 1983).

IIB. Switching Converter Modeling

Switch-mode converter models can be divided into two classes: instantaneous and averaged. Non-averaged models are just the circuit model of the converter and as such are time-varying and discontinuous due to their periodically switched nature (Fig. 1 is a instantaneous model). Such models accurately model component waveforms and large-signal behavior. They are useful for verification of operation, determination of operating points, transient phenomena, stress analysis, operating mode changes, etc. However, these models are of limited utility for the study of frequency response, stability analysis of closed-loop systems, etc. Furthermore, these models have the simulation difficulties mentioned in the introduction.

Averaged models are circuit or equation models in which the time-variations due to the switching behavior have been averaged, resulting in a continuous time-invariant model. Averaged models are much more amenable to mathematical analysis, frequency response analysis, closed-loop stability studies, and so forth. Such models are limited to frequencies much less than the switching frequency. Furthermore, averaged models cannot normally model more than one operating mode of a converter (Middlebrook and Cuk 1976).

IIC. Utility Of Simulation Events in Converter Modeling

The use of events can improve both the speed and accuracy of simulation (Cellier 1995). Suppose we are using a simulation program without explicit event handling. At each discontinuity, integration step size decreases markedly, since the simulator "perceives" the discontinuity as a sudden increase in the apparent magnitude of the system's eigenvalues. After the event is over, the step size usually increases slowly due to the conservative nature of typical integration algorithms. In the actual system, the eigenvalues may change slightly or not at all during each event, allowing the same step size as before the event, but the integration algorithm has no way of knowing this. If events occur with sufficient frequency, the integration algorithm step size remains much smaller than the dynamic behavior of the system justifies, thus lengthening the simulation run unnecessarily. Furthermore, it has been shown that lack of event handling can result in erroneous simulation results without any indication of error (Cellier 1995). These errors are due to the failure of most integration error estimation methods at extremely small step sizes, and they can result in convergence to the wrong solution without any warning.

Another disadvantage of programs without explicit event handling is that they may generate huge amounts of superfluous data, due to the very small step sizes in the region of each discontinuity. If the events are few and far between, this may not be a problem, but if they are numerous and closely spaced, as with switching converters, a program without event handling may generate a volume of data an order of magnitude larger than one that has explicit event handling. While one may attempt to alleviate this problem by specifying a larger output communication interval, by doing so one may miss event times, causing the output plots to be inaccurate (and look very strange) from aliasing of the data. A simulation language with explicit event handling will always catch each event exactly, and generate only one extra point per event. Such a program will always record data at the event times, so one's plots will be accurate.

Non-averaged models make good use of simulation events. A instantaneous model uses events to describe each switch opening and closing. Since event-handling simulation software recognizes discontinuities, it can store all state variables at the time of an event, and restart the integration with the same step size after the discontinuity is over. Furthermore, the simulation is no longer susceptible to erroneous convergence due to error estimation failure.

Although not covered in this paper, simulation events are also useful for constructing multi-mode averaged models. One of the limitations of averaged models is their inability to operate across mode boundaries. For simulation purposes, one may include more than one model description and switch between them according to operating point.

III. Dymola Modeling of Electrical Networks

It has been shown in (Cellier 1991) that Dymola may be used for topological modeling of linear electrical networks. The topological description of the circuit can be given in a netlist format similar to that used by SPICE, a necessary property for its use by circuit designers already familiar with SPICE. Dymola solves the circuit description and computes a set of state equations, from which it can generate a model in a number of simulation languages. ACSL is preferred due to its event-handling capability.

The Dymola language uses an object-oriented (hierarchical) modeling method that is well suited to circuit modeling. Each electrical component can be developed into a model, which can in turn be used as part of a larger model, similar to subcircuits in SPICE. In addition, model parameters, even those at the bottom of the hierarchy, are always easily accessible; however, they need not be carried up the hierarchy, which would result in increasingly unwieldy model descriptions. Of course, Dymola is a general purpose modeling language not limited to electrical circuits.

Dymola has been extended to handle event descriptions in a high-level object-oriented manner (Elmqvist, Cellier and Otter 1993). This makes Dymola extremely useful for modeling switching power electronic circuits. The switching event descriptions are simply included in switch models, which can then be used like any other circuit components. The following sections describe models of electrical components.

IIIA. Linear Circuit Modeling

Many switching converters can be accurately modeled with linear components such as resistors, capacitors, inductors, transformers, etc. We first define a class that describes terminal behavior for various components. The component descriptions then call this description and inherit its terminal definitions. For example, the following listing describes terminal behavior of all two-terminal (one-port) devices:

```
model class OnePort
  cut WireA(Va/i), WireB(Vb/-i)
  main path AB <WireA - WireB>
  terminal p
  local v
    v = Va - Vb
    p = v*i
end
```

Dymola keywords appear in bold type. The **model class** command defines an object class of type *OnePort*. The **cut** command defines the terminal connections of an electrical component, and the **path** command created a directed path from one cut to another, i.e. a port. The **local** variable v defines the voltage drop across the port, while p computes the power into the port.

Components are now easily described. For example, a capacitor is given by:

```
model class (OnePort) Capacitor
  parameter C=1.0
    C*der(v) = i
end
```

This model inherits the terminal description given by model class *OnePort*. The capacitance is given by the **parameter** C, with a default value of 1.0. The capacitor behavior is described by the differential equation. Other components (resistors, inductors, sources, ground, etc.) are similarly described. They are collected into a library file which can be called by a Dymola model.

IIIB. Event-Based Switch & Diode Models

An ideal switch can be considered to be a controlled resistor that has either zero or infinite resistance depending on whether the switch is on (closed) or off (open), respectively (Elmqvist, Cellier and Otter 1993). A Dymola model of an ideal switch is given by:

```
model class (OnePort) Switch0
  terminal On
    0 = if On then v else i
end
```

The **terminal** statement declares a variable *On* that may be connected to another model; this allows control of the switch. The **if** statement will generate an appropriate event description in the target simulation language, thus allowing the simulation to properly handle the switching event.

A diode can be considered a special kind of switch whose state is controlled by its terminal variables. An ideal diode has two possible states: (1) $v \le 0 \Rightarrow$ *On = false*, or (2) $i > 0 \Rightarrow$ *On = true*. Figure 2 describes the characteristic of an ideal diode. A Dymola model is given by:

```
model class (Switch0) Diode0
    new(On) = v > 0 or i > 0
end
```

This model inherits the *Switch0* behavior. The state of the switch is controlled by the switch's internal variables. On a zero-crossing of either i or v, an event is generated. The **new** statement resets the boolean variable *On* according to the boolean value of the right-hand side of the diode equation. The value of *On* changes only at the time of the event, so that between events, *On* can be considered a constant.

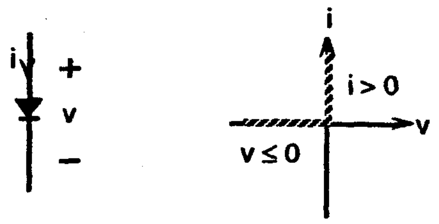

Figure 2. Ideal diode behavior.

Note that in the *Switch0* models, the computational causality of the switch differs according the value of v and i, and is thus indicated by the state of the *On* variable, so that the switch can take on either possible causality. Hence, both causalities must be compatible with the circuit. In practical terms, this means that an ideal switch cannot in general occur in a cutset whose only other elements are inductors and/or current sources, nor in a loop with only capacitors and/or voltage sources. This makes physical sense, as we cannot interrupt the current in an inductor nor instantaneously discharge a capacitor.

Unfortunately, this creates problems. Consider the operation of the buckboost converter of Fig. 1. For $t \in (t_0, t_1)$, S_1 is closed, and the inductor current ramps up with slope $(V_g - V)/L$. The voltage across the diode D_1 during this time is V_g, so D_1 is open. At $t = t_1$, S_1 is opened. The inductor current must go somewhere, so it flows through D_1, turning it on. However, the simulation program will not see it this way. It views D_1 as a switch, thus we have a cutset consisting of two switches (S_1 and D_1) and an inductor. The simulation runs until S_1 opens, at which point the simulation dies due to a division by zero.

There are a number of possible solutions to this problem. The first, and most obvious, is to make the diode into a non-ideal switch by giving it finite non-zero "on" and "off" resistances. Below is a Dymola model for such a non-ideal switch, *Switch1*. We then create a *Diode1* model identical to *Diode0*, except that it calls *Switch1*.

```
model class (OnePort) Switch1
  terminal On
  parameter Ron=1.0E-4, Roff=1.0E4
    0 = if On then v-i*Ron else v-i*Roff
end
```

This solves the problem, but not without drawbacks. For instance, in the case of the buck-boost converter of Fig. 1, there exists a mode of operation known as "discontinuous conduction mode" where, during each switching cycle, the inductor current becomes zero while S_1 is off. This results in D_1 turning off as well. The large off resistance of D_1 in series with L results in a stiff system, which may create other problems, including long simulation times. This is seen in Section IV of this paper. Another solution has been proposed in (Cellier 1995), but this is not yet formulated in an object-oriented manner.

IIIC. Modulator Models

There are two common switch control waveforms for switching converters. The most common today are pulse-width control and frequency control, achieved by means of pulse-width modulators (PWMs) and voltage-controlled oscillators (VCOs). Both waveforms can be generated by the same system, shown in Fig. 3.

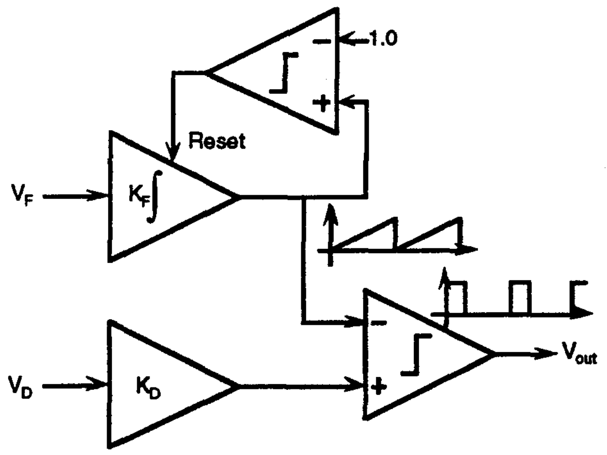

Figure 3. General purpose switching modulator. V_F *controls the switching frequency of* V_{out}, V_D *the duty cycle.*

This is represented by the following Dymola model. The **terminal** variable *D* is a boolean variable used to turn a switch on or off. The **when-endwhen** construct senses when the integrator output exceeds unity and generates a corresponding event description in the target simulation language. The integrator state *x* is reset at the event time with the **init** statement. Note that this model has two voltage inputs, *v1* and *v2*, which modulate frequency and duty-cycle respectively. The terminal structure is inherited from the model class *TwoPort*, much like the model class *OnePort* previously defined. Note that a SPICE model of this modulator requires at least two comparators along with several dependent sources and other components.

```
model class (TwoPort) Modulator
  parameter KFreq=1.0, KD=1.0, F0=0.0
  parameter RinFreq=1.0E6, RinD=1.0E6
  terminal D
  local x=0.0, xref, Frequency
    v1 = RinFreq*i1
    v2 = RinD*i2
    xref = KD*v2
    Frequency = F0 + KFreq*v1
    D = x < xref
    der(x) = Frequency
    when not x<1 then
      init(x) = 0
    endwhen
end
```

A pulse-width modulated signal can also be constructed directly, without requiring an integrator (state variable). The simpler Dymola model below accomplishes the task. Note that this model includes a start time *Tstart* before which the output is *false*. This variable can be used to offset the phase of converters in a multiple-converter system, a commonly used method to reduce switching noise.

```
model class (OnePort) PWM1
  parameter K=1.0, Ts, Tstart=0.0, Rin=1.0E6
  terminal D
  local x, xref, Nextime, Start=true
    v = i*Rin
    xref = K*v
    D = x < xref and not Time<Tstart
    x = if Time < Tstart then 0 ->
          else (Time - Nextime)/Ts + 1
    when (not Time < Nextime) or Start then
      new(Nextime) = if ->
        Start and (Tstart<0 or Tstart>0) ->
        then Tstart else Nextime+Ts
      new(Start) = false
    endwhen
end
```

There are simpler ways to model this system even in Dymola, but this model accurately represents commercially available PWM integrated circuits.

A completely equivalent SPICE model is several times more complex, since it must be specified as a circuit.

IV. Dymola Models Of Converter Systems

The following section develops a converter model by connecting the various components defined in the previous section. This model is then used as part of a larger system.

The example given is the flyback converter, shown in Fig. 4. The flyback converter is functionally equivalent to the buck-boost converter, but is more commonly used due to the isolation and turns ratio provided by the transformer.

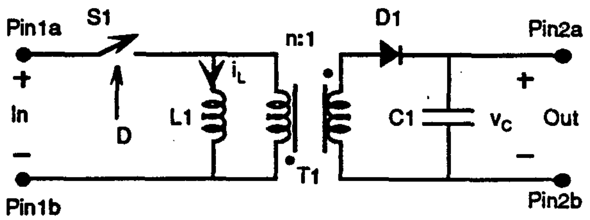

Figure 4. Flyback converter.

First, we define a terminal structure compatible with all single-input-single-output converters. This model class is similar to that for *TwoPort*, but reflects conventions used in power converter design:

```
model class Converter
  cut Pin1a(V1a/i1a), Pin1b(V1b/i1b)
  cut Pin2a(V2a/i2a), Pin2b(V2b/i2b)
  cut Port1[Pin1a, Pin1b], Port2[Pin2a, Pin2b]
  path In<Pin1a - Pin1b>, Out<Pin2a - Pin2b>
  main path P <Port1 - Port2>
    local v1, v2
      v1 = V1a - V1b
      v2 = V2a - V2b
end
```

Now, we construct the converter by connecting the various electrical components together. Each type of component used in the model is called by the **submodel** statement. Each submodel statement declares the components used in the model and their parameters. In this example, the parameters for the components *L1*, *C1*, and transformer *T1* are also parameters of the model class *Flyback0*. The terminal *D* is used to turn switch *Sw1* on or off. The **node** statement is used to connect the cuts of the various components together; note that nodes and cuts are structural equivalents. The flyback converter model is:

```
model class (Converter) Flyback0
  submodel (Inductor)   L1(L=L)
  submodel (Capacitor) C1(C=C)
  submodel (Transformer0)        T1(n=n)
  submodel (Switch0) Sw1
  submodel (Diode1)     D1(Vd=Vd)
  parameter L, C, n, Vd
  terminal D
  node n1, n2
    connect L1 from Pin1a to n1
    connect Sw1 from n1 to Pin1b
    connect <In> T1 from Pin1a to n1
    connect <Out> T1 from Pin2b to n2
    connect D1 from n2 to Pin2a
    connect C1 from Pin2a to Pin2b
    Sw1.On = D
end
```

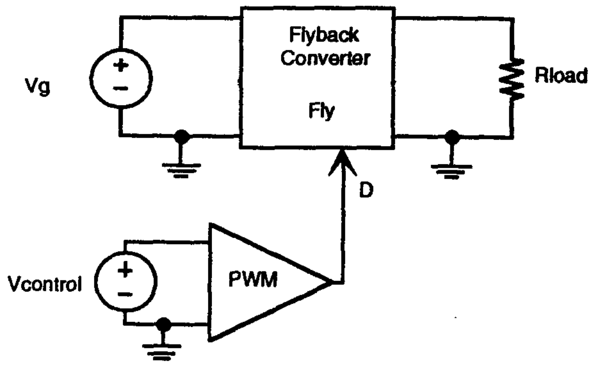

Figure 5. Flyback converter system.

The *Flyback0* model can now be used in a larger system. The simplest example of this is to connect a voltage source to the input and a resistive load to the output, shown in Fig 5. A PWM is needed to supply the switching signal to the converter.

The following Dymola program represents the system of Fig. 5:

```
model circuit
  submodel (VSource)   Vg, Vcontrol
  submodel (Resistor)  RLoad(R=5)
  submodel (Flyback0)  Fly(L=200E-6, C=22E-6, ->
                       n=1, Vd=0.7)
  submodel (PWM1)  PWM(Ts=1.67E-5)
  submodel Common
    node n0, n1, n2, n3
    output  iLoad, vLoad
      connect Common at n0
      connect Vg from n1 to n0
      connect RLoad from n2 to n0
      connect <In> Fly from n1 to n0
      connect <Out> Fly from n2 to n0
      connect PWM from n3 to n0
      connect Vcontrol from n3 to n0
      Vg.V = 40
      Vcontrol.V = 1/3
      Fly.D = PWM.D
      vLoad = RLoad.v
      iLoad = RLoad.i
end
```

This model is compiled by Dymola into an ACSL model, which is then simulated. The results of this simulation and a SPICE3 simulation of this system are plotted in Fig. 6. The two results are virtually identical. Unfortunately, a fair speed comparison was not possible, as the available version of ACSL was run on a DEC VAX/VMS system, and SPICE3 was only available on UNIX systems. For the record, the simulation CPU times were 19.12 seconds on the VAX and 80.52 seconds on a color NeXTStation (25MHz 68040). In addition, the ACSL run generated 1280 data points, and SPICE generated over 20,000! Finally, note that in order to obtain an accurate SPICE simulation, the maximum allowed time-step was 0.1 microseconds. Larger time-steps resulted in convergence to erroneous results, but they did not cause a failure to converge, i.e. there was no indication that something had gone awry. This is a much more serious problem than slow simulation speed, for obvious reasons.

The above simulation runs very fast, and the use of the nonideal switches does not cause any problems due to stiffness. The reason for the latter is that for the given parameters, the converter runs in "continuous conduction mode (CCM)" meaning one of the two switches is always on. As a result, the system never becomes stiff. However, for some applications, the so-called "discontinuous conduction mode (DCM)" is the operating mode of

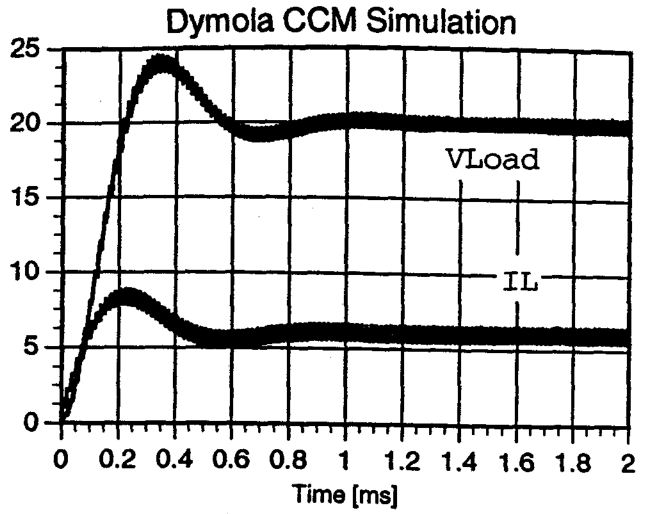

Figure 6. Dymola and SPICE3 simulation results for the system of Fig. 5, operating in CCM (continuous conduction mode).

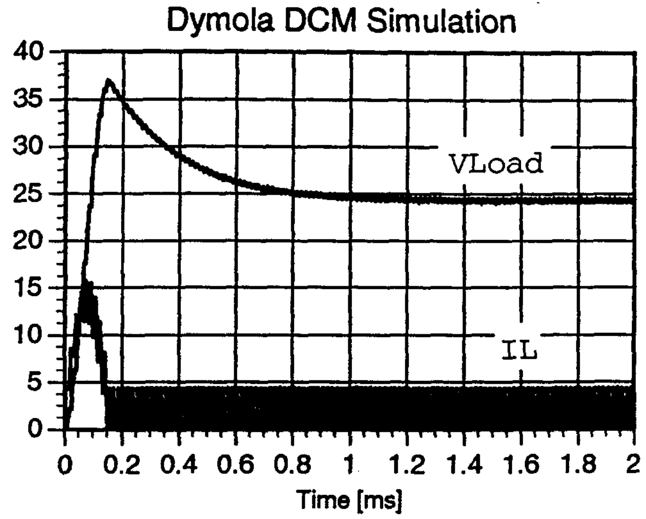

Figure 7. Dymola and SPICE3 simulation results for the system of Fig. 5, operating in DCM (discontinuous conduction mode).

choice. In this mode, all switches are off for an interval during each switching cycle. If we reduce the value of *L* (the inductor) in the submodel *Fly* to 50μH and increase the load resistance *Rload* to 20Ω, the converter will operate in DCM (simulation results for a 1 millisecond run are shown in Fig. 7). The simulation time for a 2 millisecond run has increased to 94.69 seconds for ACSL on the VAX and 92.26 for SPICE3 on the NeXTStation (ACSL generated about 1500 data points, much less than the nearly 22,000 generated by SPICE). In this case, the system becomes stiff for a fraction of each switching period, and the ACSL simulation slows drastically. What is worse is that the ACSL implementation of the Gear algorithm is unsuitable for simulation of switching converters, as it handles the discontinuities poorly (or not at all). Hence, we are stuck with using the 5th order variable-step Runge-Kutta algorithm. This is not a good algorithm for stiff systems, contributing to the slow simulation speed.

The use of non-ideal diode models can cause another problem due to introduction of artificial stiffness. In the Dymola model for the system of Fig. 5, note that the ideal *Switch0* model was used. We can get away with this since non-ideal diode *D1* is actually modeled as a variable resistance, thus *Sw1* is always in an algebraic loop. If, however, we model *Sw1* with the non-ideal switch model *Switch1*, and run the converter in DCM, the inductor current exhibits extremely small, rapid oscillation about zero. This oscillation will cause the diode to turn on and off dozens of times each switching period, slowing the simulation to a crawl (although the results remain accurate).

There are at least two solutions to these problems. First, since Dymola can formulate models as a differential-algebraeic equation (DAE), one may use a simulator that handles DAE systems. This does away with the switch causality problem, but one may pay a penalty in decreased simulation efficiency versus an ODE formulation, particularly if the system is not normally stiff. Second, one may use an extension of the Pantelides algorithm (Cellier 1995). This allows one to use ideal switches and will result in an ODE formulation of the problem. However, Dymola does not yet implement this in an automated, object-oriented manner.

VI. Conclusion

As demonstrated in the previous sections, the Dymola language allows the modeling of power converter systems in a completely objected-oriented manner. One can develop electrical component models with which one can build power converter (and other) circuits. A topological circuit description similar to that used by SPICE can be used to develop such converter models, much as the *Flyback0* model was developed in this paper. Behavioral models of more complex components can also be developed, as demonstrated by the *Modulator* and *PWM1* models given in Section IIIC. All these models can be used together to model a complete system. Dymola will translate the models and their connections into a set of differential equations and generate from the latter a complete simulation model in a choice of several simulation languages.

The use of simulation events aids the simulation of switching power converters in several ways. It can help increase simulation speed and accuracy, and it can substantially reduce the size of the data set without loss of information. Unfortunately, most simulation languages that handle events, e.g. ACSL, require a formulation of the system in terms of differential equations; this formulation is unwieldy for all but the simplest of circuits. In addition, most circuit designers are unfamiliar with such a description. The problem is worsened by the fact that most simulation languages are not object oriented, making the modeling and simulation of practical systems difficult; this is especially true when one must include event scheduling. Dymola solves both these problems at once: First, it allows event descriptions to be handled in a high-level manner by placing them inside a component model. Second, Dymola is inherently object-oriented, and the event-description-containing models are no exception. They can be employed in a system just like any other model.

Some of the problems encountered with the use of ideal switch models are discussed. In an ODE formulation of a problem, ideal switches can in general only be used when placed in an algebraic loop. Unfortunately, this creates problems in nearly all switching converters. The simplest solution, making the switches into variable resistances, can in some circumstances introduce artificial stiffness into the system, slowing the simulation speed. Other solutions include the use of a simulation language with a DAE solver, or perhaps an extension of the Pantelides algorithm to generate a ODE system where all switches end up in algebraic loops.

A practical example is given by means of the commonly used flyback converter. A converter model is constructed from common circuit components in the Dymola modeling language. This model is then used as a component in a larger system consisting of a PWM modulator model, along with a load and source. This model is compiled and simulated in ACSL. The results are consistent with those obtained by SPICE.

References

Cellier, F. 1991. *Continuous System Modeling*. Springer-Verlag New York Inc., New York, NY.

Cellier, F. to be published 1995. *Continuous System Simulation.* Springer-Verlag New York Inc., New York, NY.

Cuk, S. 1983. *Advances in Switched-Mode Power Conversion, Vol. 2.* TESLAco, Pasadena, CA.

Elmqvist, H. 1993. *Dymola - User's Manual.* DynaSim AB, Research Park Ideon, Lund, Sweden,.

Elmqvist, H.; F. Cellier; and M. Otter. 1993. "Object-Oriented Modeling Of Hybrid Systems." Proceedings of the 1993 European Simulation Symposium (ESS'93), pp. 31-41.

Lee, E. S. and T. G. Wilson. 1992. "Electrical Design Inspection: A Methodology for Using Circuit Simulation in the Design and Development of Electronic Power Supplies." In 1992 IEEE Power Electronics Specialists Conference Record, pp. 34-45.

Middlebrook, R.D. and S. Cuk. 1976. "A General Unified Approach to Modelling Switching-Converter Power Stages." 1976 IEEE Power Electronics Specialists Conference Record, pp. 73-89.

Mitchell & Gauthier Associates, Inc. 1991. *Advanced Continuous Simulation Language (ACSL) Reference Manual.* Concord, MA.

Quarles, T.; A.R. Newton, D.O. Pederson, A. Sangiovanni-Vincentelli. 1993. SPICE3 Version 3f3 User's Manual. Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, CA.