DC, AC Small-Signal and Transient Analysis of Level1 N-Channel MOSFET with Modelica

A. Urquia and S. Dormido
Dep. Informática y Automática, Facultad de Ciencias, U.N.E.D.
Avda. Senda del Rey 9, 28040 Madrid, Spain.
E-mail: aurquia@dia.uned.es, sdormido@dia.uned.es

ABSTRACT

The “translation” of the SPICE capabilities into Modelica language would allow combining the best of each tool: the SPICE expertise at circuit analysis and the Modelica/Dymola expertise at object-oriented modelling and simulation of hybrid systems. This contribution intends to be a first step to achieve this goal. A reduced group of SPICE device models are translated into Modelica language for OP, AC and TRAN analyses. It includes passive components (resistor and capacitor), independent voltage and current sources, and the SPICE2 level1 n-channel MOSFET.

1. INTRODUCTION

The simulator SPICE is an essential computer-aid for circuit design. Originally, SPICE2 was conceived as a stand-alone, general purpose, analog circuit simulator. However, since the development of SPICE2 at the University of California in 1975, many commercial and freeware SPICE-compatible simulators have been developed for a variety of systems (UNIX, PC, etc). Most of these tools

- run in connection with other simulation programs used in the circuit design flow,
- support analog, digital and mixed analog/digital simulation, and
- include improved device models, additional analyses and device model libraries.

They provide some support to the multi-domain system simulation facilitating the *analog behavioral modelling* (ABM). Behavioral parts allow defining a circuit segment as a mathematical expression or a lookup table. PSpice (OrCAD, 1999) is a commercial, PC-version, SPICE-compatible simulator. PSpice ABM library includes math functions, limiters, Chebyshev filters, integrators, differentiators, etc. However, the SPICE-based simulators impose a hard restriction to ABM: the function continuity (OrCAD, 1999; Kielkowski, 1998).

Device equations built into SPICE are continuous. For instance, voltage- or current-controlled switches are not ideal: they have a finite (very small) “on” resistance and (very large) “off” resistance. The switch resistance changes smoothly between the two as its control voltage or current changes. Equally, the functions available for ABM are also continuous (for instance, the *int* function can not be implemented). The reason behind this requirement is the heavy use that SPICE numerical algorithms make of continuity (OrCAD, 1999; Kielkowski, 1998). In consequence, SPICE-based simulators are not suited for the simulation of hybrid models (i.e., combined continuous/discrete models) due to its inability to handle discrete events.

On the contrary, general-purpose modelling languages are intended for the simulation of multi-domain hybrid models. To this respect, the object-oriented modelling language Modelica (Modelica, 2000) is intended to serve as a standard format so that models arising in different domains can be exchanged between tools and users (Aström, Elmqvist and Mattsson, 1998). The “translation” of the SPICE capabilities (device models and analysis modes) into Modelica language is one of the Modelica library improvements that have been suggested (Clauss et al., 2000). It would allow combining the best of each tool: the SPICE expertise at circuit analysis and the Modelica/Dymola (Elmqvist et al., 2000) expertise at object-oriented modelling and simulation of hybrid systems. This contribution intends to be a first step to achieve this goal.

An important feature of SPICE device models is their *variable-structure* nature. A model is said to have a variable structure when its mathematical description changes during the simulation run. A different device model is formulated for each analysis mode:

- static model (DC analysis),
- AC small-signal model (AC analysis), and
- large-signal model (transient analysis).

The transitions among these three device formulations are carried out in simulation time. A DC analysis (Massobrio and Antognetti, 1993)

- can be performed prior to a transient analysis to determine the transient initial conditions, and
- it is automatically performed prior to an AC small-signal analysis to determine the linearized, small-signal models for the non-linear devices.

In addition, some DC analysis algorithms require the combined use of the three device formulations.

In this contribution three analysis modes are considered:

- bias point (OP),
- AC sweep (AC), and
- transient analysis (TRAN),

for three analog device types:

- *Passive devices*: linear resistor and capacitor.
- *Independent voltage and current sources*.
- *Semiconductor device*: SPICE2 level1 n-channel MOSFET. It is composed of linear resistors, voltage-dependent capacitors and voltage-controlled current sources.

In addition, IC1 and IC2 pseudo-components are modelled for setting initial conditions.

Model structuring into libraries and the interaction between models are discussed in Section 2. The way of using the model libraries to analyse the user-defined circuits is also outlined. Initial condition setting is described Section 3. Two procedures are supported: IC symbols and the capacitor IC property. The translation into Modelica language of the passive device and source models is addressed in Section 4. Device models have a variable structure and signals are defined to control the model structure transitions. Each analysis mode consists on an ordered sequence of elementary operations implying changes in the device model structure. Analysis models set the control signals in order to accomplish the required device-model structure changes. Analysis models are discussed in Section 5. Bias point calculation is the most problematic step from the numerical point of view. Four alternative bias point calculation algorithms are implemented. Finally, level1 NMOS model is outlined in Section 6.

For the sake of simplicity, neither parameter dependence with temperature nor TEMP analysis have been considered in the present library release. Temperature is considered a constant variable intervening in some device constitutive relations (for instance, the MOSFET source-substrate pn-junction model).

2. ARCHITECTURE

A two-level architecture is proposed (see Fig. 1):

- Upper (controller) level is composed of the analysis models.
- Lower (controlled) level is composed of the device models.
- Unidirectional *control signals* (arrow in Fig. 1) and *global variables* transmit the information from analysis models to device models. In addition, global parameters sets properties common to both analysis and device models.

Controller level: analysis models

ANALYSES package contains the *OP*, *TRAN* and *AC* models. Bias point calculation is a part of OP and AC analyses and it is an option of TRAN analysis. Therefore, the bias point calculation algorithms are programmed in a separate partial model, called *BiasPointCalculation*, inherited by the analysis models (see Fig. 1). Control signals (see Table 1) and global variables (see Table 2) are evaluated in the analysis models.

Control signal	T	W	R
Ctrl_AC	B	*	S
Ctrl_CBREAK_resetTran	B	BPC	C
Ctrl_CBREAK_Tran2DC	B	*	C
Ctrl_CBREAK_Tran2IC	B	*	C
Ctrl_DC	B	BPC	S
Ctrl_IC_clampDC	B	BPC	C, IC
Ctrl_IC_clampTran	B	BPC	C, IC
Ctrl_IC_mode	I	BPC	C, IC
Ctrl_IS_inhibit	B	BPC	S
Ctrl_IS_TranOP	B	BPC	S
Ctrl_log_AC	B	*	S, R
Ctrl_log_DC	B	BPC	S, R
Ctrl_OP_mode	I	BPC	S
Ctrl_OP_value	I	*	S
Ctrl_RBREAK_Tran2DC	B	BPC	R
Ctrl_Tran	B	*	S

Table 1. Control signals.
T: Variable type. (B): Boolean. (I): Integer (0,1)
W: Control signal written during the…
(BPC): bias point calculation. (*): other steps of the analyses.
R: Control signal read by …
(S): source. (C): capacitor. (R): resistor. (IC):IC symbols

scaleGMIN	Scale factor of the “GMIN stepping” algorithm for bias point calculation.
Freq	AC small-signal frequency.
Temp	Analysis temperature.

Table 2. Global variables.

Controlled level: device models

Device models are grouped in three packages:

- BREAKOUT,
- SOURCE, and
- SPECIAL.

The models of BREAKOUT and SOURCE packages allow the composition of user-defined circuits, while the SPECIAL’s provide one way to specify the simulation initial conditions. In addition, a fourth package containing the device model interfaces has been defined: INTERFACE.

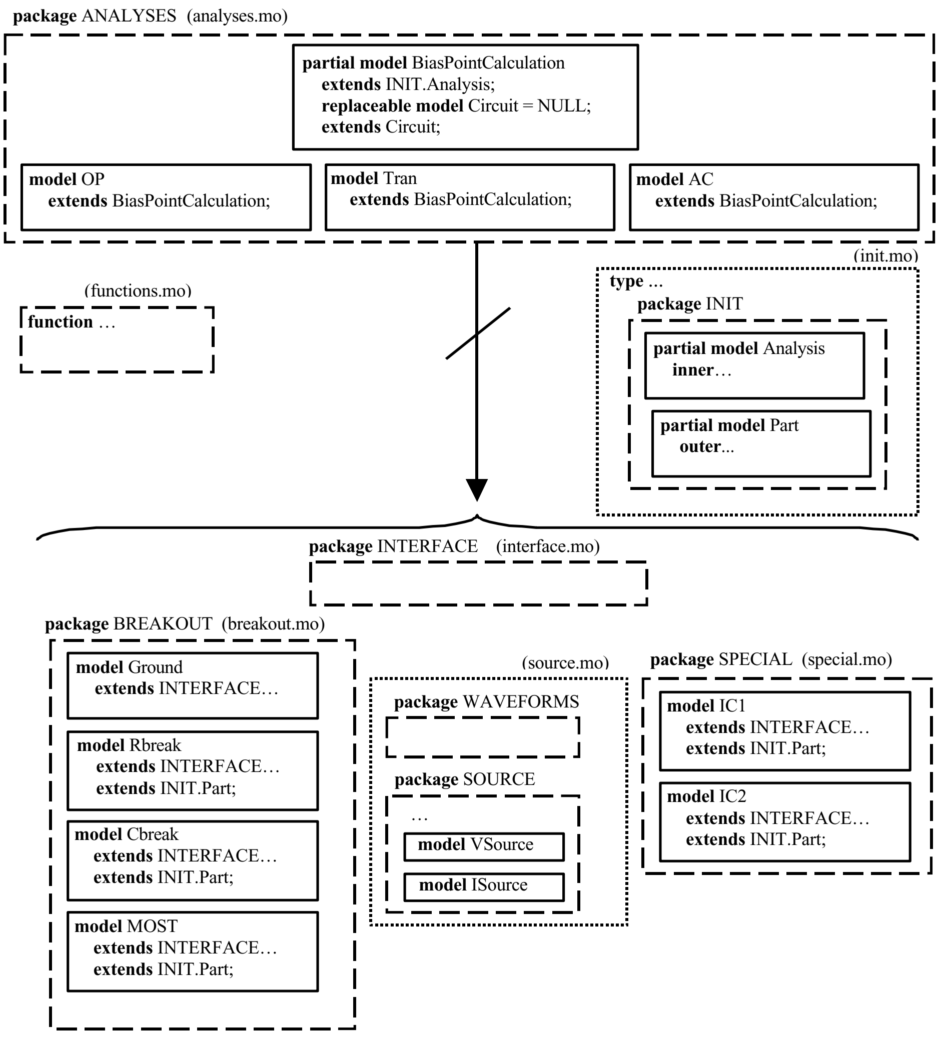


Figure 1. Two-level architecture.

Initialisation file

The initialisation file, *init.mo*, contains:

- Type definitions. Types conform to the Modelica *SIunits* package. However, they are redefined for the sake of conciseness when used. For instance:
```
        type Voltage   =
              Modelica.SIunits.Voltage;
```
- INIT package. The control signals, the global variables and the global parameters are defined in the INIT package. It contains two partial models (see Fig. 1):
  - *Analysis*, inherited by the analysis models.
  - *Part*, inherited by the device models.

  The same set of control signals, variables and parameters is defined in both partial models: *Analysis* model variables are **inner** ones, while *Part* variables are **outer** ones.

Global parameters

Two global parameters have been defined (see Table 3). `TIME_SCALE` is used for setting the length of the source-ramping processes of some bias point calculation algorithms. In addition, it is used for establishing the time elapsed between consecutive control signal transitions (conceptually similar to the system clock period). To this end, the integer parameter `TIME_SLOT` is defined in the analysis models. It represents a percentage (1 to 100). The time between consecutive events, `CLOCK`, is defined as follows:

```
    CLOCK = TIME_SLOT * TIME_SCALE / 100
```

TIME_SCALE	It is intended for providing an (rough) approximate value of the circuit time-constant.
LOG_RESULTS	It determines the amount of information to be logged during the bias point calculation and the AC small-signal analysis.

Table 3. Global parameters

`TIME_SCALE` parameter plays another important role (not implemented in the current release of the libraries): redefine the units of the `time` variable in order to allow the adequate numerical solution of the system. Circuit simulation for microelectronics applications requires very small time values in comparison with the by-default time-related DAE-solver parameters. For this reason, it is best to include a scale factor between the circuit time and the DAE-solver time (i.e., the `time` variable).

Similar considerations will be made when discussing the pn-junction model. The use of the international system of units for the current is inadequate, because it leads to numerical problems. Large differences in the order of magnitude of the variables (for instance, the current and the voltage) makes impossible to set adequate values for the numerical algorithm tolerances, the Dymola *eveps* parameter for event detection (Elmqvist, Cellier and Otter, 1993), etc. This fact is taken into account by re-formulating the model constitutive relations. In order to keep the compatibility with Modelica standard libraries, the international system of units is used for all the model terminal variables.

Performing circuit analyses

Two pieces of information are needed to perform a circuit analysis: the analysis model and the circuit model. The analysis models inherit (as a *replaceable model*, called *Circuit*) the circuit model (see *BiasPointCalculation* in Fig 1). The analysis model instantiations have to contain the redeclaration of the *Circuit* model. Consider the following example:

(File: *my_circuit.mo*)
```
model my_circuit
…
end my_circuit;

model circuitAnalysis_OP =
ANALYSES.OP ( redeclare model Circuit =
              my_circuit);

model circuitAnalysis_Tran =
ANALYSES.Tran ( redeclare model Circuit =
                my_circuit);

model circuitAnalysis_AC =
ANALYSES.AC ( redeclare model Circuit =
              my_circuit);
```

The analysis to perform (only one per run) is selected in the script file. For instance, AC analysis:

(File: *my_circuit.mos*)
```
openModel("pspice.mo");
openModel("my_circuit.mo");
checkModel(problem="circuitAnalysis_AC");
translateModel(problem=
              "circuitAnalysis_AC");
```

The file *pspice.mo*:

- imports the library files (see Table 4), and
- defines the graphic windows containing the model icons.

File	Package
analyses.mo	ANALYSES
breakout.mo	BREAKOUT
functions.mo	
init.mo	INIT
interface.mo	INTERFACE
pspice.mo	PSPICE
source.mo	WAVEFORMS SOURCE
special.mo	SPECIAL

Table 4. Complete list of files and packages.

3. SETTING INITIAL CONDITIONS

Adopting the PSpice methodology (OrCAD, 1999), two equivalent procedures are provided to specify the analysis initial conditions:

- Setpoint pseudo-components: IC1 and IC2 (see Fig. 1, SPECIAL package). IC1 is a one-pin symbol that allows setting the initial voltage on a node. IC2 is a two-pin symbol that allows setting the initial voltage between two nodes.
- The IC property of capacitors (inductor model is not included in this library release).

IC property allows associating the initial condition with a device, while the IC symbols allow the association to be with a node or a node pair. Note that these ways of specifying the simulation initial condition substitute the Dymola standard procedures to set the initial value of the state variables.

Two operations require the static model solution:

- *bias point* calculation (during OP and AC), and
- *transient initial condition calculation*.

When the transient initial condition calculation is skipped (a Boolean parameter controls this option), the devices with the IC property defined start with the specified value. However, all other such devices have an initial state of zero. IC symbols are ignored.

IC symbols clamp the voltage for the entire bias point calculation. PSpice attaches a voltage source with a 0.0002 ohm series resistance (`R_EPS`) at each net to which an IC symbol is connected. This is the set-up of the IC-symbol Modelica model. The model of the capacitor IC-property depends on whether the bias point is calculated or the calculation is skipped:

- During the bias point calculation, the capacitor IC property is implemented using an IC2 symbol in parallel with the capacitor. The capacitor model contains this voltage-clamp circuit.
- When the initial transient solution is skipped, the capacitor voltage is initialised to its IC value using a “when clause”.

Control signals have been defined to set the state (open/close) of the IC symbols switches, initialise the capacitor voltage drop, etc.

4. DEVICE MODELS

Resistor, capacitor and independent source models are discussed.

4.1. Interface

Device models are composed of three formulations: static, AC small-signal and large-signal. Each model formulation is described by its own set of equations and variables. Pin model is conceived to allow the simultaneous connection of the three formulation terminal variables. AC small-signal currents and voltages (complex numbers) are represented in rectangular coordinates (i.e., real and imaginary). The current is positive when flows into the pin.

The interface of the two-pin devices is composed of two *Pin* connectors. PSpice sign criterion for current is adopted: *positive current flows from the (+) node through the device to the (-) node*.

(File: *interface.mo*)
```
connector Pin
  Voltage      vDC    "Static model”;
  Voltage      vTran  "Large-signal model";
  Voltage      vAC_Re "AC small-signal";
  Voltage      vAC_Im "AC small-signal”;
  flow Current iDC    "Static model”;
  flow Current iTran  "Large signal”;
  flow Current iAC_Re "AC small-signal";
  flow Current iAC_Im "AC small-signal";
…
end Pin;
```

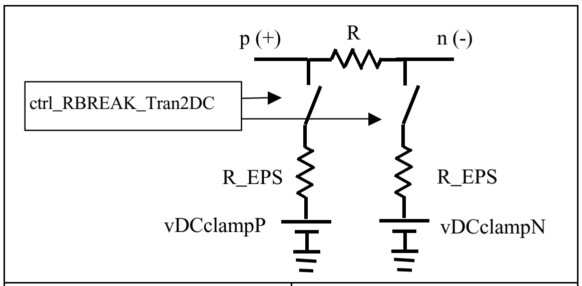
```
partial model TwoPin
   Pin      p   "(+) node";
   Pin      n   "(-) node";
…
```

4.2. Linear resistor

Resistor static model is shown in Fig 2. The purpose of the IC1-like circuits (switches, `R_EPS` resistors and voltage sources) is clamping the DC-formulation voltage at the pins. The bias point calculation algorithm "dynamic model ramping" requires the following operation: clamping the DC-formulation voltage to the instantaneous value of the large-signal formulation. The `ctrl_RBREAK_Tran2DC` signal controls this information transfer between formulations. When `ctrl_RBREAK_Tran2DC` becomes true:

- The source voltages (`vDCclampP` and `vDCclampN`) are set to the instantaneous value of the transient voltage at the correspondent pin. Then source voltages are held constant.
- The switches are closed. They remain closed only while the signal is true.

The large-signal and AC small-signal models do not include these IC1-like circuits.



When	Then
`ctrl_RBREAK_Tran2DC`	`vDCclampP = p.vTran;` `vDCclampN = n.vTran;`

Figure 2. Resistor static model.

4.3. Capacitor

Linear and voltage-dependent capacitors have to be modelled. The partial model *Capacitor* describes all the capacitor behavior except its large-signal and AC small-signal capacitance. *Cbreak* model (linear capacitor) and MOS1 capacitors extend *Capacitor*.

Capacitor static-formulation is shown in Fig. 3. The implementation of the IC property requires the IC2-like circuit (switch, `R_EPS` resistor and `vClampDC` source). Large-signal formulation is shown in Fig. 4. IC2-like circuit is also included because the "dynamic model ramping" algorithm uses the large-signal formulation during the bias point calculation. The Boolean signals

- `ctrl_IC_clampDC`, and
- `ctrl_IC_clampTran`.

controls the static and large-signal model switches respectively.

The capacitor parameter `IC_ENABLED` enables or disables the IC property. It allows distinguishing between the cases when IC is intentionally set to zero and those cases when the IC property is not enabled (and its by-default value is also zero).

The signal `ctrl_IC_mode` controls `vClampDC` and `vClampTran` voltages. Some bias point calculation algorithms need the independent sources ramping from zero up to their nominal initial values. When implementing these algorithms, the voltage clamping sources of the IC symbols and the capacitor IC property need also be ramped from zero to their respective IC values. Two cases are distinguished:

- `ctrl_IC_mode==0`, the clamping voltage (`vClampDC or vClampTran`) is constant and equal to the IC value.
- `ctrl_IC_mode==1`, the clamping voltage is ramped from zero up to its IC value.

In addition, control signals trigger instantaneous changes in the capacitor large-signal voltage drop (see Fig. 4).

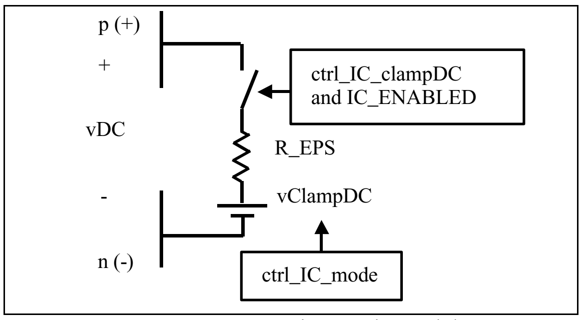
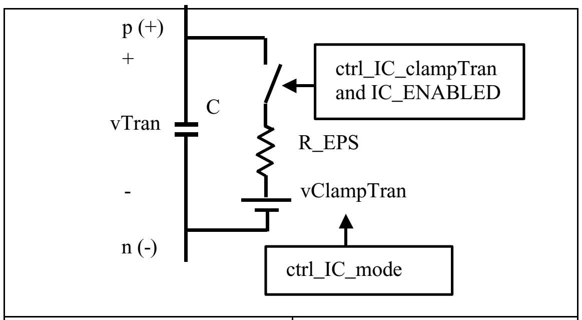


Figure 3. Capacitor static model.



When	then
`ctrl_CBREAK_Tran2IC and IC_ENABLED`	`reinit(vTran, IC);`
`ctrl_CBREAK_Tran2DC`	`reinit(vTran, vDC);`
`ctrl_CBREAK_resetTran`	`reinit(vTran, 0);`

Figure 4. Capacitor large-signal model.

4.4. Independent sources

There are a lot of similarities between the models of the voltage and the current independent sources:

- the interface,
- the DC and transient analysis signals, etc.

The elements in common are defined in the partial model *Stimulus* (SOURCE package) and the source models (*VSource* and *ISource*, see Fig. 1) inherit it.

Source model parameters allow defining the DC and AC characteristics of the source:

- DC analysis: `DC_VALUE`.
- AC analysis: `AC_MAG` and `AC_PHASE`.

Time-dependent waveforms used in the transient analyses are defined in the WAVEFORMS package (see Fig. 1): EXP, PULSE and PWL. PSpice standard has been adopted for waveform parameter names. The *Stimulus* model inherits the waveform model as a *replaceable model*. Therefore, the waveform model can be declared when instantiating the source model (no waveform is selected by default). Some examples are provided in Table 5.

DC and AC specifications:
```
SOURCE.VSource  V1(
   DC_VALUE=3, AC_MAG=10, AC_PHASE=45 );
```
EXP waveform:
```
SOURCE.VSource  V1(
   DC_VALUE=3, AC_MAG=10, AC_PHASE=45,
   redeclare model
    TransientSpecification =
    WAVEFORMS.EXP( S1=1,S2=2,TD1=1,TC1=1,
                   TD2=3,TC2=1 ));
```
PULSE waveform:
```
SOURCE.VSource  V1(
   DC_VALUE=3, AC_MAG=10,
   redeclare model
    TransientSpecification =
    WAVEFORMS.PULSE( S1=1,S2=2, TD=1,TR=1,
                     PW=3,TF=1, PER=8 ));
```
PWL waveform:
```
SOURCE.VSource  V1(
   DC_VALUE=3, AC_MAG=10, AC_PHASE=30,
   redeclare model
    TransientSpecification =
    WAVEFORMS.PWL(
      signalCorners = { 1, 2, 4, 8, 16 },
      timeCorners   = { 0, 1, 2, 3, 4 } ));
```
Table 5. Examples of source instantiations.

DC analysis

The control signal `ctrl_DC` enables or disables the DC model:

- While `ctrl_DC==false`, the DC value of all the independent sources of the circuit is zero.
- While `ctrl_DC==true`, the DC value of the sources is determined by the integer parameters:
  - `ctrl_OP_mode`, and
  - `ctrl_OP_value`.

In order to set the source value when calculating the initial transient condition, a parameter is associated to each waveform model: `TRANS_INITIAL`. This parameter coincides with the waveform initial value.

The parameter `ctrl_OP_value` determines the source value during the static model solution:

- `ctrl_OP_value==0`: source value is `DC_VALUE`.
- `ctrl_OP_value==1`: value is `TRANS_INITIAL`.

The parameter `ctrl_OP_mode` determines the mode of reaching the previous value:

- `ctrl_OP_mode==0`: the source is hold constant to the value.
- `ctrl_OP_mode==1`: the source value is increased linearly from zero with a slope equal to the value divided by `TIME_SCALE`.

The "dynamic model ramping" algorithm requires the cancellation of the independent sources. The control signal `ctrl_IS_inhibit` allows this operation. While it is true:

- voltage independent sources are substituted by opens (current=0), and
- current independent sources by shorts (voltage=0).

Transient analysis

The control signal `ctrl_Tran` determines:

- whether the transient analysis is enabled, and the source signal is calculated of its associated waveform (`ctrl_Tran==true`),
- or the static bias point calculation is enabled (`ctrl_Tran==false`). The algorithm "dynamic model ramping" requires the circuit large-signal model simulation in order to calculate a "good" initial value for static model iteration.

While `ctrl_Tran==false`, the source value is determined by the parameter `ctrl_IS_TranOP`:

- While `ctrl_IS_TranOP==false`, the value is zero.
- While `ctrl_IS_TranOP==true`, the value depends on the parameters `ctrl_OP_mode`, and `ctrl_OP_value`. The response associated to these parameters is the same than the previously discussed for the static formulation.

AC small-signal analysis

While the control signal `ctrl_AC` is true, the AC small-signal value of the source is set according to the source parameters `AC_MAG` and `AC_PHASE`. Otherwise, the value is zero.

Model of the disabled formulations

It is important to notice that while a model formulation is not enabled, the correspondent values of the independent sources are zero. In this situation, the circuit node voltages are trivially calculated and the simulation computational effort is not unnecessarily increased. The control signals that enable each of the three formulations are:

- `ctrl_DC`,
- `ctrl_Tran`, and
- `ctrl_AC`.

Total power dissipation

The bias point calculation includes the evaluation of the total power dissipation. It is calculated adding the contribution of all the independent voltage sources:

$$W_{DC} = \sum_{\substack{\text{all indep.} \\ \text{V sources}}} v_{DC}\left(-i_{DC}\right)$$

The calculation is implemented thanks to the Modelica capability of describing "physical fields" (see Table 6). The `PowerDisipation` connector is defined. The model of the voltage source contains:

- an instantiation of this connector,
- the declaration of an outer connector of this type,
- the connection between them.

The "environment" (inner) connector is defined in the *BiasPointCalculation* model.

4.5. Log of analysis results

The analysis results are logged to the *dslog.txt* file using the Dymola's *LogVariable* function. Two parameters control this information log:

- `LOG_RESULTS` (global parameter). It allows specifying the required detail level at logging results (see Table 7).
- `HIDDEN_COMPONENT`. This device-dependent parameter classifies the circuit devices into two types: those whose variables have to be logged always (`HIDDEN_COMPONENT==false`), and those whose variables have to be logged only in special cases (`HIDDEN_COMPONENT==true`).

The complex AC small-signal voltages and currents are logged in Cartesian and polar coordinates. In addition, the polar magnitude is also expressed in decibels (defined as 20log10( )).

(File: *interface.mo*)
```
connector PowerDisipation
   flow Power disipatedPower;
…
```
(File: *source.mo*)
```
model VSource
…
 outer INTERFACE.PowerDisipation
          TotalPowerDisipation;
 INTERFACE.PowerDisipation powerDisipation;
…
equation
 when ctrl_log_DC then
  powerDisipation.disipatedPower =
                             vDC*(-iDC);
 end when;
 connect ( powerDisipation,
          TotalPowerDisipation );
…
```
(File: *analyses.mo*)
```
partial model BiasPointCalculation
   inner INTERFACE.PowerDisipation
            TotalPowerDisipation;
…
```
Table 6. Total power dissipation calculation.

	`HIDDEN_COMPONENT`	
	`False`	`true`
Voltage at resistor pins	0, 1, 2	2
Current through independent voltage sources	0, 1, 2	2
Total power dissipation	0, 1, 2	2
Voltage drop at resistors	1, 2	2
Current through resistors	1, 2	2
Power dissipation of each independent voltage source	1, 2	2

Table 7. `LOG_RESULTS` values producing the variable log as a function of `HIDDEN_COMPONENT` value.

5. ANALYSES

PSpice OP, AC and TRAN analyses are translated into Modelica language. Note that analysis models force the simulation end when they have completed their operations (*terminate* function is used). Large simulation times should be selected in the Dymola program window to avoid interfering with analysis execution.

5.1. Bias point calculation

PSpice provides three alternative algorithms for solving the circuit static model (OrCAD, 1999):

- static model iteration,
- static model ramping, and
- GMIN stepping.

PSpice first tries to solve the static model of the circuit using the Newton-Raphson algorithm. If a solution is not found and "GMIN stepping" is enabled (using .OPTION STEPGMIN) then GMIN algorithm is applied. If it also fails or it is not enabled then "static model ramping" is applied. In addition to these three algorithms, a fourth one is programmed in the *BiasPointCalculation* model: the "dynamic model ramping" algorithm, proposed in (Cellier, 1991). The `SOLVE_STATIC` parameter determines which of the four algorithms to use.

Two control signals, internal to the analysis models, are defined to synchronize the bias point calculation with other analysis operations:

- `biasPoint`. Its transition from false to true indicates that the static-model solution must start.
- `biasPointCalculated`. When the static-model solution is just finished, it becomes true.

The *BiasPointCalculation* model reads the value of `biasPoint` signal and writes `biasPointCalculated`.

Next, the four algorithms are briefly discussed. The control signal transitions required for algorithm completion are shown, but for the sake of clarity, their cause-effect relationships are omitted. Two additional comments:

- `ctrl_OP_value` signal is not written by the bias point calculation algorithms.
- Control signals evaluated at bias point calculation (see Table 1) and hold to false during the whole algorithm, are omitted.

Static model iteration (`SOLVE_STATIC:=0`)

The solution of the static problem is left in hands of the modelling language. PSpice has two symbols to provide an initial guess for Newton-Raphson algorithm: NODESET1 and NODESET2 (OrCAD, 1999). These symbols have not been translated into Modelica language because they do not represent any advantage compared to Dymola Initial Calculation methods (Elmqvist et al., 2001). The Modelica implementation of the algorithm is shown in Fig. 5.

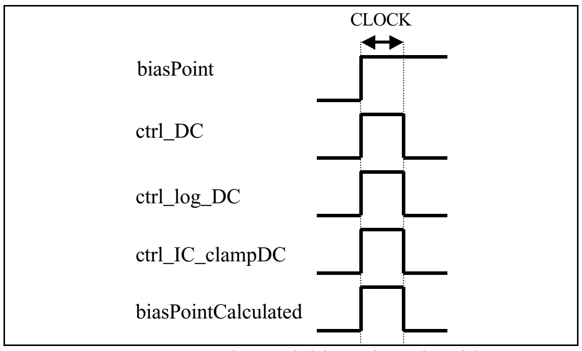


Figure 5. Static model iteration algorithm.

Static model ramping (`SOLVE_STATIC:=1`)

PSpice cuts back the power supplies to almost zero (0.001%) so that all non-linearities are turned off. When the circuit is linear, a solution can be found (very near zero, of course). The initial condition of this first step is zero for all voltages. Then, PSpice works its way back up to 100% power supplies using a variable step size (OrCAD, 1999). The process relies heavily on the equation continuity with respect to the power supplies.

This algorithm is translated into Modelica language ramping the static-formulation value of the independent sources from zero up to their target values. The clamping voltages of the IC symbols and the capacitor IC property are also adequately ramped. The value of the parameter `TIME_SCALE` determines the length of the ramping. The algorithm is implemented by means of the signal transitions shown in Fig. 6.

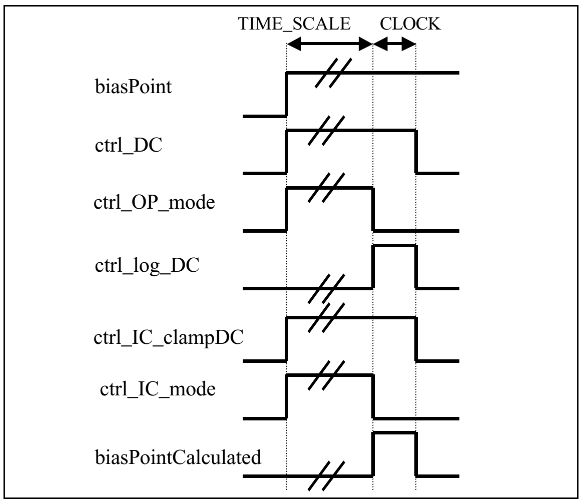


Figure 6. Static model ramping algorithm.

GMIN stepping (`SOLVE_STATIC:=2`)

GMIN stepping attempts to find a solution for the static model (with power supplies at 100%) by starting with a large value of GMIN, initially 1.0e10 times the nominal value. If a solution is found at this setting, PSpice reduces GMIN by a factor of 10 and tries again. This continues until either GMIN is back to the nominal value, or a repeating cycle fails to converge. This algorithm makes heavy use of equation continuity with respect to GMIN model parameters. The Modelica implementation of this algorithm is shown in Fig. 7.

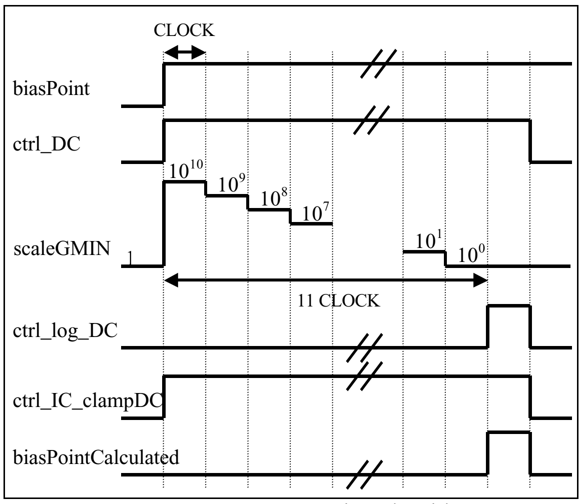


Figure 7. GMIN stepping algorithm.

Dynamic model ramping (`SOLVE_STATIC:=3`)

The initial condition to iterate the static model is obtained by simulating the large-signal model (Cellier, 1991). A transient analysis is performed: all sources are ramped up from zero to the desired initial value for the simulation and this value is held for some time to allow the circuit to stabilise. Then the large-signal formulation voltages are transferred to the static model (using `ctrl_RBREAK_Tran2DC` and `ctrl_IS_inhibit`). This static-circuit setting is held for a clock cycle. Then, the power supplies are connected to the circuit, the resistor voltage-clamping circuits are disconnected, and the static model is solved. The Modelica implementation of the algorithm is shown in Fig. 8.

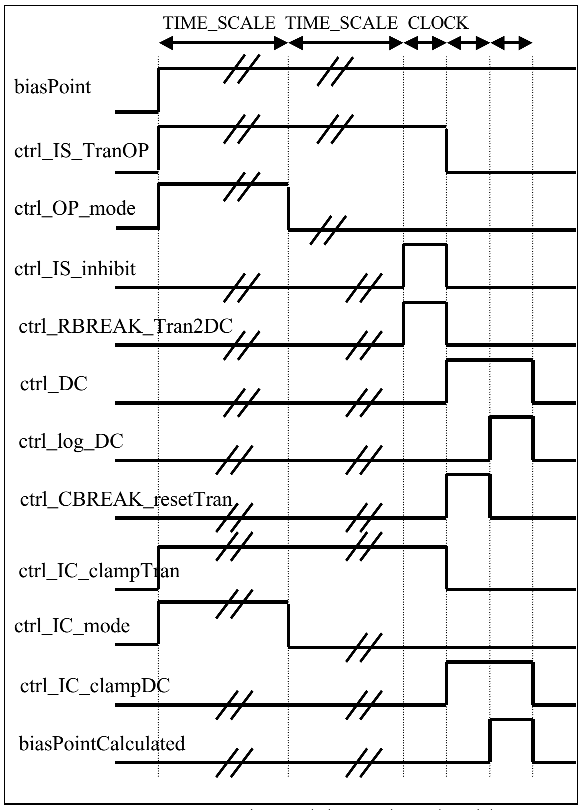


Figure 8. Dynamic model ramping algorithm.

5.2. Bias point analysis (OP)

The OP analysis (see Fig. 9):

- forces the `biasPoint` signal to become true,
- sets `ctrl_OP_value` signal to zero, and
- finish the simulation one clock cycle after the `biasPointCalculated` signal becomes true.

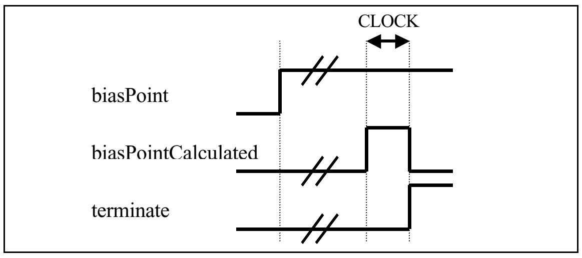


Figure 9. OP analysis signals.

Example

Consider the application of the OP analysis algorithms to the trivial circuit shown in Fig 10. Dymola's experiment *StopTime* variable is set to an arbitrary large value: 100. The `TIME_SLOT`, `TIME_SCALE` and `LOG_RESULTS` parameters are left to their by-default values: 10%, 1s and 0 respectively.

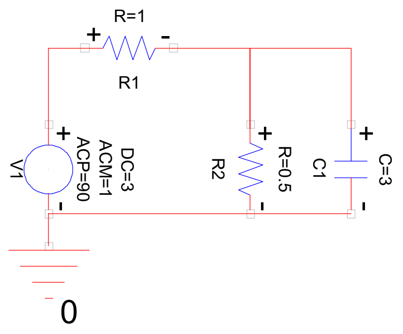


Figure 10. Simple example of a RC circuit.

- `SOLVE_STATIC:=0`. Once finished the simulation (at `T=0.1`), *dslog.txt* file contains the results:

```
V1_iDC(1e-010) = -2
V1_vDC(1e-010) = 3
R1_n_vDC(1e-010) = 1
ctrlx_0logx_0DC(1e-010) = 1
V1_powerDisipation_disipatedPower(1e-010)=6
```

- `SOLVE_STATIC:=1`. The *dslog.txt* file contains the results, logged at `T=1`. The simulation terminates at `T=1.1` (see Fig 11).
- `SOLVE_STATIC:=2`. The circuit does not contain any device with the GMIN parameter, so this algorithm is equivalent to SOLVE_STATIC:=0. Results are logged at `T=1.1` and the simulation finishes at `T=1.2` (see Fig. 12).
- `SOLVE_STATIC:=3`. Results are logged at `T=2.2` and the simulation finishes at `T=2.3`. Large-signal and static voltages at `R1.n` node are shown in Fig. 13. At `T=2.0`: large-signal to static info. transfer. At `T=2.1`: Static model solution.

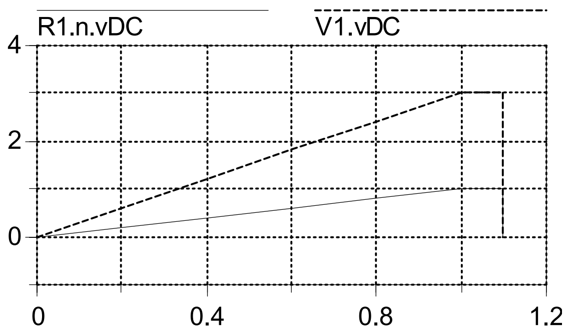


Figure 11. Voltage at circuit nodes.

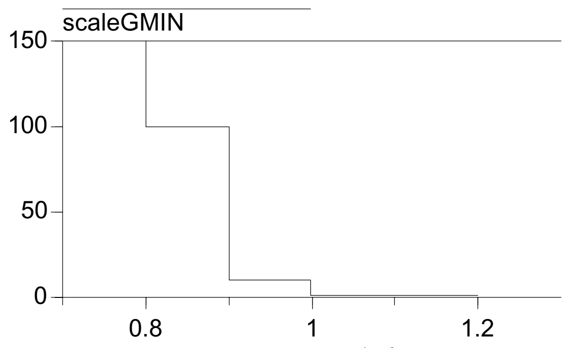


Figure 12. GMIN scale factor

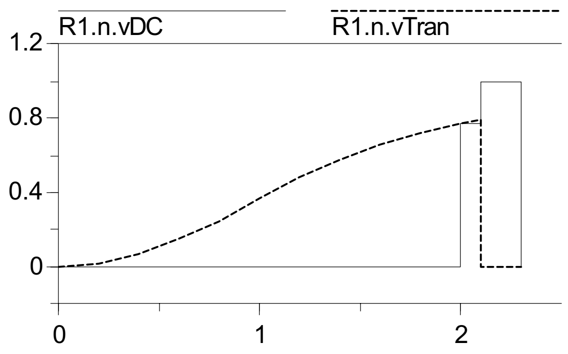


Figure 13. Static and large-signal voltages.

5.3. AC sweep analysis (AC)

The `TYPE_AC_SWEEP` parameter defines the frequency sweep type (LIN and DEC PSpice arguments):

- `TYPE_AC_SWEEP==0`: frequency linear sweep.
- `TYPE_AC_SWEEP==1`: the frequency is swept logarithmically by decades.

AC small-signal analysis (see Fig. 14):

- forces the `biasPoint` signal to become true, and
- sets `ctrl_OP_value` signal to zero.

When `biasPointCalculated` becomes true, the AC analysis:

- forces `ctrl_AC` to become true, enabling the AC model.
- Starts the frequency sweep. The frequency variation in time depends on the sweep type. In both cases, the required log frequencies are spaced at regular time-intervals of length `2*CLOCK`. Therefore, the `ctrl_log_AC` signal is a pulse train of period `2*CLOCK`.

The simulation is finished one clock cycle after the frequency reaches `END_FREQUENCY`. An AC analysis of the Fig 10 circuit is shown in Fig 15.

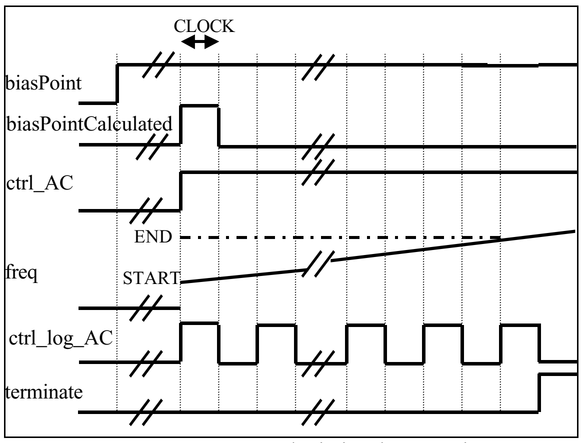


Figure 14. AC analysis implementation.

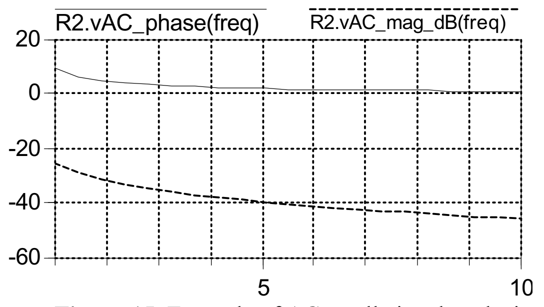


Figure 15. Example of AC small-signal analysis.

5.4. Transient analysis (TRAN)

When the transient simulation is started, the value of the `time` variable is different of zero. For this reason, a variable is defined to measure the transient simulation time: `timeTran`. The length of the transient simulation is set by the `TRAN_STOP_TIME` parameter. The transient analysis depends on the `SKIP_INITIAL_TRAN_SOLUTION` parameter.

`SKIP_INITIAL_TRAN_SOLUTION:=false`

When `biasPointCalculated` becomes true, the circuit static model contains the transient initial solution. Then (see Fig. 16):

- `ctrl_CBREAK_Tran2DC` becomes true. The large-signal circuit state is initialised to the static-circuit voltage values.
- `ctrl_Tran` becomes true. The large-signal device models are enabled.

The simulation terminates when `timeTran` reaches the value `TRAN_STOP_TIME`.

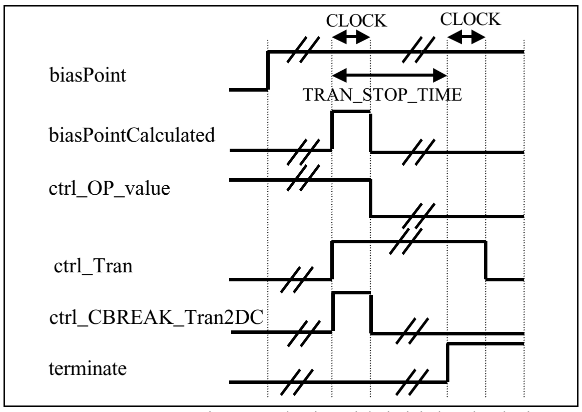


Figure 16. Transient analysis with initial calculation.

`SKIP_INITIAL_TRAN_SOLUTION:=true`

At initial time (see Fig. 17):

- `ctrl_CBREAK_Tran2IC` becomes true. The large-signal circuit state is initialised to the IC-property correspondent values.
- `ctrl_Tran` becomes true. The large-signal device models are enabled.

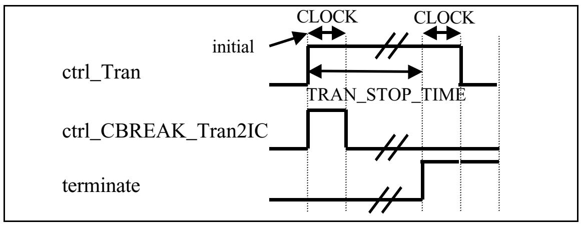


Figure 17. Transient analysis w/o initial calculation.

6. SPICE2 LEVEL 1 NMOS

The SPICE2 level1 MOS model is basically the model proposed by Shichman and Hodges (Massobrio and Antognetti, 1993). The Dymodraw diagram of the model is shown in Fig 18. Each substrate junction is modelled as a voltage-controlled current source (diode-like icon in Fig. 18) in parallel with a voltage-controlled capacitor. I_{DS} is a non-linear current source controlled by the voltages V_{DS}, V_{GS} and V_{BS}. The gate capacitance is modelled using three voltage-controlled capacitors: C_{GB}, C_{GS} and C_{GD}.

Voltage-controlled capacitors have been modelled extending the *Capacitor* model. Expressions for the large signal capacitance are provided and the small-signal capacitance is evaluated at the bias point (i.e., when `ctrl_AC` signal becomes true). Large-signal and static formulations of controlled current sources are equal (of course, each one is described by its own set of variables). Their small-signal models (conductance) are evaluated at bias point.

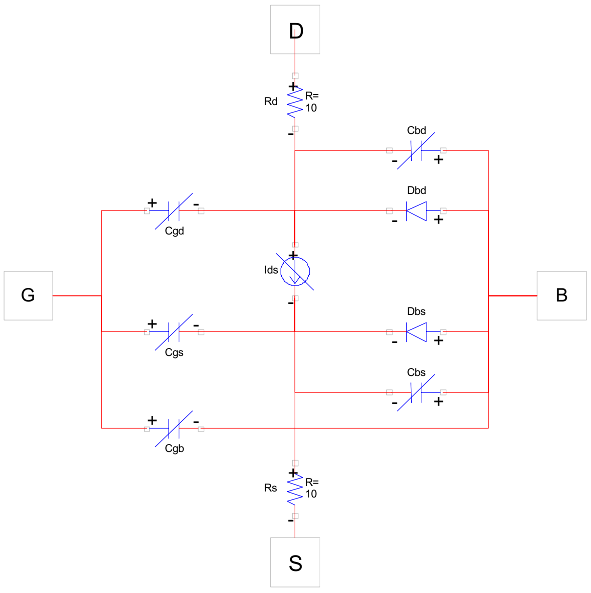


Figure 18. SPICE2 level1 NMOS

Conclusions

A reduced set of SPICE device models has been successfully translated into Modelica language for OP, AC and transient analyses.

References

Aström, K. J., H. Elmqvist and S. E. Mattsson (1998). *Evolution of Continuous-Time Modeling and Simulation.* 12th ESM, Manchester, UK.

Cellier, F. E. (1991). *Continuous System Modeling.* Springer-Verlag.

Clauss, C., et al. (2000). *Modelling of Electrical Circuits with Modelica.* Modelica Workshop 2000, Lund, Sweden.

Elmqvist, H., F. E. Cellier and M. Otter (1993). *Object-Oriented Modelling of Hybrid Systems.* ESS'93, Delft, The Netherlands.

Elmqvist, H., et al. (2001). *Dymola. Dynamic Modeling Laboratory. User Manual.* Dynasim.

Kielkowski, R.M. (1998). *Inside SPICE.* McGraw-Hill, Inc. Second Edition.

Massobrio, G and P. Antognetti (1993). *Semiconductor Device Modeling With SPICE.* McGraw-Hill, Inc.

Modelica (2000). *Modelica, Language Specification & Tutorial.* Modelica Association.

OrCAD. (1999). *OrCAD PSpice A/D. Reference Guide & User's Guide.* OrCAD, Inc.