Compact Models User Guide

Version N-2017.09, September 2017

SYNOPSYS®

Synopsys, Inc.
690 E. Middlefield Road
Mountain View, CA 94043
www.synopsys.com

Contents

About This Guide

This user guide is to be used in conjunction with the *Sentaurus™ Device User Guide* and *Sentaurus™ Interconnect User Guide*. It provides details about the different types of compact model that are available:

- SPICE models based on the Berkeley SPICE model version 3F5 [1][2][3][4]. The BSIM3v3.2, BSIM4.1.0, and BSIMPD2.2 MOS models are also available.
- Frequently used Synopsys HSPICE® models.
- Built-in models, which are available only in Sentaurus Device. They work in a similar way to SPICE models. However, they provide additional functionality not found in SPICE models.
- User-defined models, which are available only in Sentaurus Device. They can be implemented using a compact model interface. The model code must be implemented in C++ and is linked to Sentaurus Device dynamically at runtime. No access to the source code of Sentaurus Device is necessary. The speed of user-defined models is comparable to that of built-in models in Sentaurus Device.

Related Publications

For additional information, see:

- The TCAD Sentaurus release notes, available on the Synopsys SolvNet® support site (see Accessing SolvNet on page viii).
- Documentation available on SolvNet at https://solvnet.synopsys.com/DocsOnWeb.

Conventions

The following conventions are used in Synopsys documentation.

Convention	Description
Blue text	Identifies a cross-reference (only on the screen).
Bold text	Identifies a selectable icon, button, menu, or tab. It also indicates the name of a field or option.

Convention	Description
`Courier font`	Identifies text that is displayed on the screen or that the user must type. It identifies the names of files, directories, paths, parameters, keywords, and variables.
Italicized text	Used for emphasis, the titles of books and journals, and non-English words. It also identifies components of an equation or a formula, a placeholder, or an identifier.

Customer Support

Customer support is available through the Synopsys SolvNet customer support website and by contacting the Synopsys support center.

Accessing SolvNet

The SolvNet support site includes an electronic knowledge base of technical articles and answers to frequently asked questions about Synopsys tools. The site also gives you access to a wide range of Synopsys online services, which include downloading software, viewing documentation, and entering a call to the Support Center.

To access the SolvNet site:

1. Go to the web page at https://solvnet.synopsys.com.
2. If prompted, enter your user name and password. (If you do not have a Synopsys user name and password, follow the instructions to register.)

If you need help using the site, click **Help** on the menu bar.

Contacting Synopsys Support

If you have problems, questions, or suggestions, you can contact Synopsys support in the following ways:

- Go to the Synopsys Global Support Centers site on synopsys.com. There you can find e-mail addresses and telephone numbers for Synopsys support centers throughout the world.
- Go to either the Synopsys SolvNet site or the Synopsys Global Support Centers site and open a case online (Synopsys user name and password required).

Contacting Your Local TCAD Support Team Directly

Send an e-mail message to:

- support-tcad-us@synopsys.com from within North America and South America.
- support-tcad-eu@synopsys.com from within Europe.
- support-tcad-ap@synopsys.com from within Asia Pacific (China, Taiwan, Singapore, Malaysia, India, Australia).
- support-tcad-kr@synopsys.com from Korea.
- support-tcad-jp@synopsys.com from Japan.

Acknowledgments

The SPICE circuit simulator was developed by the Department of Electrical Engineering and Computer Science (EECS), University of California, Berkeley, and is copyrighted by the University of California.

BSIM3 was developed by the BSIM Research Group in the Department of Electrical Engineering and Computer Science (EECS), University of California, Berkeley, and is copyrighted by the University of California (http://bsim.berkeley.edu/).

References

[1] R. M. Kielkowski, *Inside SPICE*, New York: McGraw-Hill, 2nd ed., 1998.

[2] K. S. Kundert, *The Designer's Guide to SPICE and SPECTRE*, Boston: Kluwer Academic Publishers, 1995.

[3] G. Massobrio and P. Antognetti, *Semiconductor Device Modeling With SPICE,* New York: McGraw-Hill, 2nd ed., 1993.

[4] T. Quarles *et al.*, *SPICE 3 Version 3F5 User's Manual*, Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, CA, USA, March 1994.

CHAPTER 1 SPICE Models

This chapter describes the available SPICE models.

Available SPICE Models

The following models are available:

- Elementary Devices on page 3
- Voltage Sources and Current Sources on page 8
- MOSFET Models (NMOS and PMOS) on page 19
- Non-MOSFET Transistors and Diodes on page 100

NOTE The compact model for an arbitrary source (ASRC) and the models for transmission lines (LTRA, Tranline, URC) are not available.

MOSFET Models

The following MOSFET models are implemented:

- Mos1 is described by a square-law I–V characteristic.
- Mos2 [1] is an analytic model.
- Mos3 [1] is a semiempirical model.
- Mos6 [2] is a simple analytic model accurate in the short-channel region.
- BSIM1 [3][4][5] and BSIM2 [6] are the Berkeley short-channel IGFET models.
- BSIM3 is a physics-based, accurate, and robust MOSFET SPICE model [7][8] for circuit simulation and CMOS technology development.
- BSIM4 is the fourth version of the Berkeley IGFET model for SPICE.
- BSIMPD2.2 is a partially depleted silicon-on-insulator MOSFET model.

The Mos2, Mos3, and BSIM1 models include second-order effects such as channel-length modulation, subthreshold conduction, scattering-limited velocity saturation, small-size effects, and charge-controlled capacitances.

The BSIM3 model has extensive built-in dependencies of important dimensional and processing parameters, allowing you to model MOSFET behavior accurately over a wide range of channel lengths and channel widths. The BSIM3 model includes compact analytic expressions for the following physical phenomena:

- Short-channel and narrow-channel effects on threshold voltage
- Nonuniform doping effect (in both lateral and vertical directions)
- Mobility reduction due to vertical field
- Bulk charge effect
- Carrier velocity saturation
- Drain-induced barrier lowering (DIBL)
- Channel-length modulation (CLM)
- Substrate current-induced body effect (SCBE)
- Subthreshold conduction
- Source and drain parasitic resistances

The BSIM3v3.2 model also is included, which has the following enhancements and improvements relative to BSIM3v3.1:

- An original and accurate charge thickness capacitance model that considers the finite charge layer thickness (quantum effects). This model is smooth, continuous, and very accurate through all regions of operation.
- Improved modeling of C–V characteristics at the weak to strong inversion transition.
- Addition of T_{ox} dependency in the threshold voltage (V_{th}) model.
- Addition of flat-band voltage (V_{fb}) as a new model parameter.
- Improved substrate current scalability with channel length.
- Restructured non-quasistatic (NQS) model, addition of NQS into the pole-zero analysis, and fixed bugs in NQS codes.
- Addition of temperature dependency into the diode junction capacitance.
- DC diode model supports a resistance-free diode and current-limiting feature.
- Option of using the inversion charge of capMod 0, 1, 2, or 3 to evaluate BSIM3 thermal noise.
- Elimination of the small negative capacitance of Cgs and Cgd in the accumulation–depletion regions.
- A separate set of channel-width and channel-length dependency parameters (`llc`, `lwc`, `lwlc`, `wlc`, `wwc`, and `wwlc`) to calculate W_{eff} and L_{eff} for the C–V model for a better fit of the capacitance data.
- Addition of parameter checking to avoid inappropriate values for certain parameters.

Temperature Dependencies

The SPICE models assume that input data has been measured at a nominal temperature of $27°C$. This value can be overridden for the parameter sets that provide a `tnom` parameter.

Similarly, the default operating temperature of all SPICE instances is $27°C$ (300.15 K). This default can be changed for those instances that provide a `temp` parameter.

For details of the BSIM temperature adjustments, refer to the literature [9][10].

Elementary Devices

The elementary device models discussed in this section include:

- Simple Linear Resistor
- Capacitor
- Inductor
- Coupled (Mutual) Inductors
- Voltage-Controlled Switch
- Current-Controlled Switch

Simple Linear Resistor

Resistors are specified by giving the value of the resistance [Ω]. This value can be positive or negative, but not zero.

A more general form of the resistor allows for modeling temperature effects and calculating the actual resistance value from strictly geometric information and specifications of the process.

The sheet resistance is used, with the narrowing parameter and the length and width of the device, to determine the nominal resistance by the formula:

$$r = rsh\frac{l - narrow}{w - narrow} \tag{1}$$

`defw` is used to supply a default value for w if none is specified for the device. If either rsh or l is not specified, the standard default resistance $1\ \text{k}\Omega$ is used. After the nominal resistance is calculated, it is adjusted for temperature by the formula:

$$r(temp) = r(tnom) \cdot (1 + tc_1 \cdot (temp - tnom)) + tc_2 \cdot (temp - tnom)^2) \tag{2}$$

Device name:	`Resistor`
Default parameter set name:	`Resistor_pset`
Electrodes:	`R+, R-`
Internal variables:	None

Table 1 Resistor model parameters

Name	Description	Type	Default	Unit
`defw`	Default device width	double	1e-05	m
`narrow`	Narrowing of resistor	double	0	m
`rsh`	Sheet resistance	double	0	Ω /sq
`tc1`	First-order temperature coefficient	double	0	$°C^{-1}$
`tc2`	Second-order temperature coefficient	double	0	$°C^{-2}$
`tnom`	Parameter measurement temperature	double	27	°C

Table 2 Resistor instance parameters

Name	Description	Type	Default	Unit
`resistance`	Resistance	double	1000	Ω
`temp`	Instance operating temperature	double	27	°C
`l`	Length	double	0	m
`w`	Width	double	1e-05	m

Capacitor

If the value of capacitance is not given, it can be computed from strictly geometric information and the specifications of the process as follows:

$$\text{capacitance} = cj \cdot (l - narrow) \cdot (w - narrow) + 2 \cdot cjsw \cdot (l + w - 2 \cdot narrow) \tag{3}$$

Device name:	`Capacitor`
Default parameter set name:	`Capacitor_pset`
Electrodes:	`C+, C-`
Internal variables:	None

Table 3 Capacitor model parameters

Name	**Description**	**Type**	**Default**	**Unit**
`cj`	Bottom capacitance per area	double	0	F/m^2
`cjsw`	Sidewall capacitance per meter	double	0	F/m
`defw`	Default width	double	1e-05	m
`narrow`	Width correction factor	double	0	m

Table 4 Capacitor instance parameters

Name	**Description**	**Type**	**Default**	**Unit**
`capacitance`	Device capacitance	double	0	F
`ic`	Initial capacitor voltage	double	0	V
`l`	Device length	double	0	m
`w`	Device width	double	1e-05	m

Inductor

Device name:	`Inductor`
Default parameter set name:	`Inductor_pset`
Electrodes:	`L+, L-`
Internal variables:	`branch` (current through inductor)

NOTE There are no parameters for this parameter set.

Table 5 Inductor instance parameters

Name	**Description**	**Type**	**Default**	**Unit**
`ic`	Initial current through inductor	double	0	A
`inductance`	Inductance of inductor	double	0	H

Coupled (Mutual) Inductors

Coupled inductors are specified by introducing a coupling `k` between two existing inductors. The inductors `inductor1` and `inductor2` must have been previously specified. Example: Implementing Coupled Inductances on page 156 discusses the implementation of coupled inductances using the compact model interface.

Device name:	`mutual`
Default parameter set name:	`mutual_pset`
Electrodes:	None
Internal variables:	None

NOTE There are no parameters for this parameter set.

Table 6 Coupled inductors instance parameters

Name	Description	Type	Default	Unit
`coefficient`	(redundant parameter)	double	0	–
`inductor1`	First coupled inductor	string	“ ”	–
`inductor2`	Second coupled inductor	string	“ ”	–
`k`	Mutual inductance	double	0	–

Voltage-Controlled Switch

The electrodes `S+` and `S-` are the nodes between which the switch terminals are connected. The electrodes `SC+` and `SC-` are the positive and negative controlling nodes, respectively. The switch is not ideal because it must have a finite positive resistance in the off-state. However, the value can be chosen such that it is effectively infinite compared to the other circuit elements.

The switch is switched on if the controlling voltage is greater than `vt` + `vh`. It is switched off if the controlling voltage is smaller than `vt` – `vh`.

NOTE A voltage-controlled switch must be used only for transient simulations. It will not switch on or off during a quasistationary simulation.

Device name:	`Switch`
Default parameter set name:	`Switch_pset`
Electrodes:	`S+, S-, SC+, SC-`
Internal variables:	None

Table 7 Voltage-controlled switch model parameters

Name	Description	Type	Default	Unit
`roff`	Resistance when open	double	1e+12	Ω
`ron`	Resistance when closed	double	1	Ω
`vh`	Hysteresis voltage	double	0	V
`vt`	Threshold voltage	double	0	V

Table 8 Voltage-controlled switch instance parameters

Name	Description	Type	Default	Unit
`off`	Switch initially open	integer	–	–
`on`	Switch initially closed	integer	–	–

Current-Controlled Switch

The electrodes `W+` and `W-` are the nodes between which the switch terminals are connected. The switch is controlled by the current that flows through the voltage source given by the parameter `control`. The direction of a positive controlling current flow is from the positive node, through the source, to the negative node.

NOTE This voltage source must be specified before the switch.

The switch is not ideal because it must have a finite positive resistance in the off-state. However, the value can always be chosen such that it is effectively infinite compared to the other circuit elements. The switch is switched on if the controlling current is greater than `it` + `ih`. It is switched off if the controlling current is smaller than `it` – `ih`.

NOTE A current-controlled switch must be used only for transient simulations. It will not switch on or off during a quasistationary simulation.

Device name:	`CSwitch`
Default parameter set name:	`CSwitch_pset`
Electrodes:	`W+, W-`
Internal variables:	None

Table 9 Current-controlled switch model parameters

Name	Description	Type	Default	Unit
ih	Hysteresis current	double	0	A
it	Threshold current	double	0	A
roff	Open resistance	double	1e+12	Ω
ron	Closed resistance	double	1	Ω

Table 10 Current-controlled switch instance parameters

Name	Description	Type	Default	Unit
control	Name of controlling source	string	" "	–
off	Initially open	integer	–	–
on	Initially closed	integer	–	–

Voltage Sources and Current Sources

The voltage source and the current source models discussed in this section include:

- Values of Independent Sources
- Independent Voltage Source
- Independent Current Source
- Voltage-Controlled Current Source
- Voltage-Controlled Voltage Source
- Current-Controlled Current Source
- Current-Controlled Voltage Source

Values of Independent Sources

The independent voltage sources and current sources have the same parameters.

DC Source

The parameter `dc` specifies the DC value of the source. For example, `dc = 10` defines a DC voltage/current source of 10 V/10 A.

Pulse Source

The `pulse` parameter must be a vector of length 7. Its entries define a transient pulse as shown in Table 11.

Table 11 Pulse source instance parameters

Parameter	Description	Unit
`v1 = pulse [0]`	Initial value	V or A
`v2 = pulse [1]`	Pulsed value	V or A
`td = pulse [2]`	Delay time	s
`tr = pulse [3]`	Rise time	s
`tf = pulse [4]`	Fall time	s
`pw = pulse [5]`	Pulse width	s
`per = pulse [6]`	Period	s

Such a pulse produces the values in Table 12 (see Figure 1 on page 10).

Table 12 Pulse source values

Time	Value
`0`	v1
`td`	v1
`td+tr`	v2
`td+tr+pw`	v2
`td+tr+pw+tf`	v1
`per+td`	v1
`per+td+tr`	v2

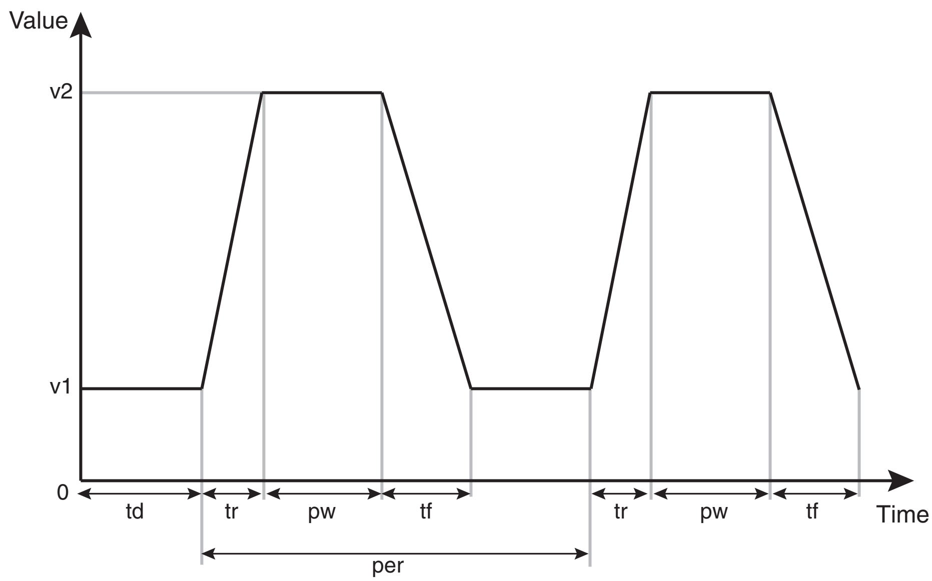


Figure 1 Pulse source parameters

Intermediate values are determined by linear interpolation. For example, the following specification produces the pulse shown in Figure 2:

```
pulse = (-0.2  1  0.8  0.2  2.0  1  5)
```

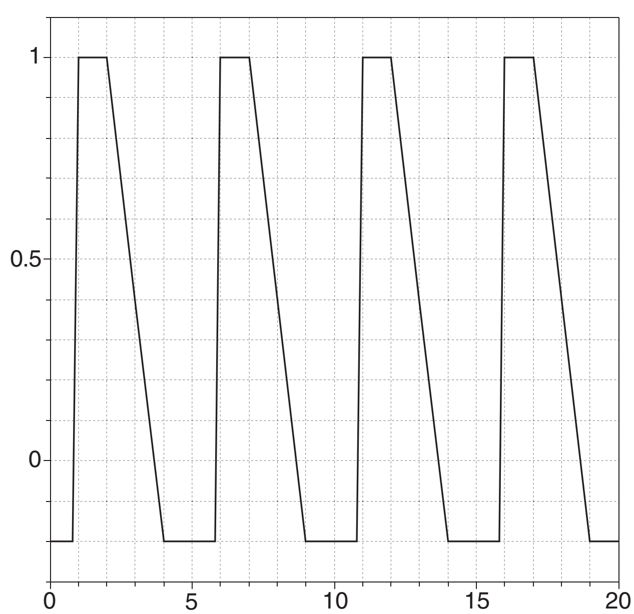


Figure 2 Pulse source

Sinusoidal Source

The `sine` parameter must be a vector of length 5. Its entries are listed in Table 13.

Table 13 Sinusoidal source instance parameters

Parameter	Description	Unit
`vo = sine [0]`	Offset	V or A
`va = sine [1]`	Amplitude	V or A
`freq = sine [2]`	Frequency	Hz
`td = sine [3]`	Delay	s
`theta = sine [4]`	Damping factor	s^{-1}

A sinusoidal source produces the values shown in Table 14.

Table 14 Sinusoidal source values

Time	Value
$t \le td$	vo
$t > td$	$vo + va \cdot e^{-(t-td) \cdot theta} \cdot \sin(2 \cdot \pi \cdot freq \cdot (t - td))$

For example, the following specification produces the sine wave shown in Figure 3:

```
sine = (0.2  0.4  0.25  4  0.01)
```

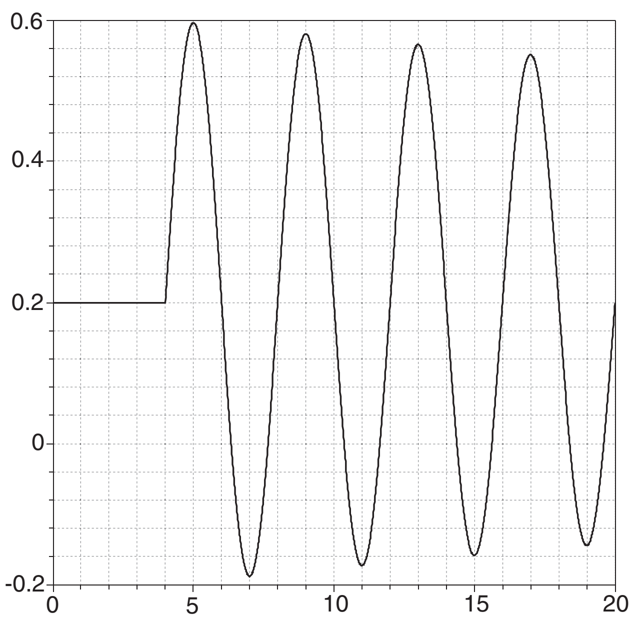


Figure 3 Sine source

Exponential Source

The `exp` parameter must be a vector of length 6. Its entries are listed in Table 15.

Table 15 Exponential source instance parameters

Parameter	**Description**	**Unit**
`v1 = exp [0]`	Initial value	V or A
`v2 = exp [1]`	Pulsed value	V or A
`td1 = exp [2]`	Rise delay time	s
`tau1 = exp [3]`	Rise time constant	s
`td2 = exp [4]`	Fall delay time	s
`tau2 = exp [5]`	Fall time constant	s

The shape of the waveform is described by Table 16.

Table 16 Exponential source values

Time	**Value**
$t \le td1$	$v1$
$td1 < t \le td2$	$v1 + (v2 - v1)\left(1 - e^{-\frac{t - td1}{tau1}}\right)$
$t > td2$	$v1 + (v2 - v1)\left(1 - e^{-\frac{t - td1}{tau1}}\right) + (v1 - v2)\left(1 - e^{-\frac{t - td2}{tau2}}\right)$

For example, the following specification produces the shark fin shown in Figure 4 on page 13:

```
exp = (0.2  0.4  2  5  10  3)
```

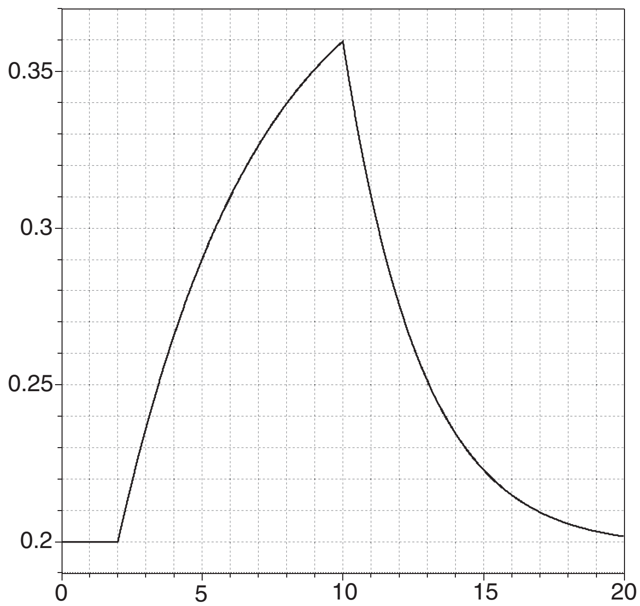


Figure 4 Exponential source

Piecewise Linear Source

The parameter `pwl` must be a vector of even size. It consists of pairs (t_k, v_k) that specify the value v_k [V or A] at the time $t = t_k$. The value of the source at intermediate values of time is determined using linear interpolation on the input values.

For example, the following specification produces the curve shown in Figure 5 on page 14:

```
pwl = (0 0  2 2  3 -1  6 6  10 5  12 -1  16 4  19 2  20 5)
```

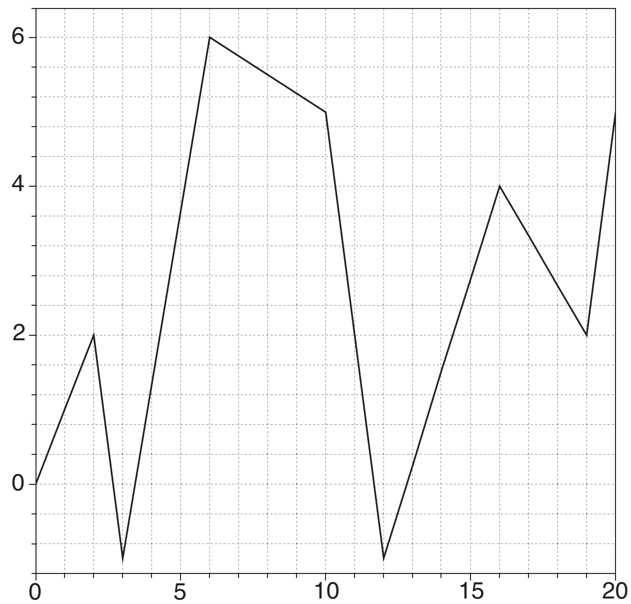


Figure 5 Piecewise linear source

Single-Frequency FM Source

The parameter `sffm` must be a vector of size 5. Its entries are listed in Table 17.

Table 17 Single-frequency FM source instance parameters

Parameter	Description	Unit
`vo = sffm [0]`	Offset	V or A
`va = sffm [1]`	Amplitude	V or A
`fc = sffm [2]`	Carrier frequency	Hz
`mdi = sffm [3]`	Modulation index	–
`fs = sffm [4]`	Signal frequency	Hz

The shape of the waveform is described by:

$$v(t) = vo + va \cdot \sin(2 \cdot \pi \cdot fc \cdot t + mdi \cdot \sin(2 \cdot \pi \cdot fs \cdot t)) \tag{4}$$

For example, the following specification produces the signal shown in Figure 6 on page 15:

```
sffm = (1  2  1  3  0.25)
```

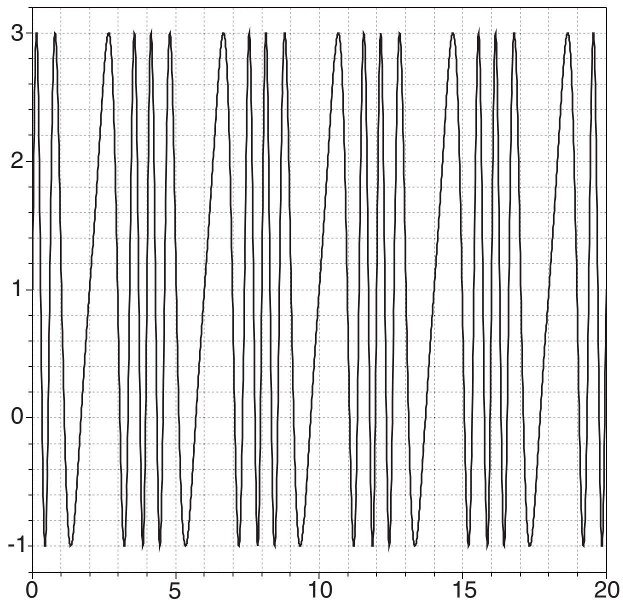


Figure 6 Single-frequency FM source

Independent Voltage Source

A SPICE voltage source can be used as an *ammeter* in a circuit, that is, a zero-valued voltage source can be inserted into the circuit to measure the current. Voltage sources are referenced by the `control` parameter in current-controlled current sources (CCCS), current-controlled voltage sources (CCVS), and current-controlled switches (CSwitch).

Only one of the parameters `dc`, `pulse`, `sine` or `sin`, `exp`, `pwl`, or `sffm` must be specified. For DC simulations, the value of the source for the time t = 0 is used.

Device name:	`Vsource`
Default parameter set name:	`Vsource_pset`
Electrodes:	`V+, V-`
Internal variables:	`branch` (current through voltage source)

NOTE There are no parameters for this parameter set.

Table 18 Independent voltage source instance parameters

Name	Description	Type	Default	Unit
dc	DC source value	double	0	V
pulse	Pulse description	double[7]	–	–
sine	Sinusoidal source description	double[5]	–	–
sin[a]	Sinusoidal source description	double[5]	–	–
exp	Exponential source description	double[6]	–	–
pwl	Piecewise linear description	double[][b]	–	–
sffm	Single-frequency FM description	double[5]	–	–

a. Equivalent to the sine parameter.
b. Vector of even size.

Independent Current Source

A current source of positive value forces the current to flow from the I+ node, through the source, to the I- node.

Only one of the parameters dc, pulse, sine or sin, exp, pwl, or sffm must be specified. For DC simulations, the value of the source for the time t = 0 is used.

Device name:	Isource
Default parameter set name:	Isource_pset
Electrodes:	I+, I-
Internal variables:	None

NOTE There are no parameters for this parameter set.

Table 19 Independent current source instance parameters

Name	Description	Type	Default	Unit
dc	DC value of source	double	0	A
pulse	Pulse description	double[7]	–	–
sine	Sinusoidal source description	double[5]	–	–
sin[a]	Sinusoidal source description	double[5]	–	–
exp	Exponential source description	double[6]	–	–

Table 19 Independent current source instance parameters (Continued)

Name	Description	Type	Default	Unit
`pwl`	Piecewise linear description	double[][b]	–	–
`sffm`	Single-frequency FM description	double[5]	–	–

a. Equivalent to the `sine` parameter.
b. Vector of even size.

Voltage-Controlled Current Source

`V+` and `V-` are the positive and negative nodes, respectively. The current flows from the positive node, through the source, to the negative node. `VC+` and `VC-` are the positive and negative controlling nodes, respectively. The value of the current is given by:

$$i = gain \cdot (v(VC+) - v(VC-)) \tag{5}$$

Device name:	`VCCS`
Default parameter set name:	`VCCS_pset`
Electrodes:	`V+, V-, VC+, VC-`
Internal variables:	None

NOTE There are no parameters for this parameter set.

Table 20 Voltage-controlled current source instance parameter

Name	Description	Type	Default	Unit
`gain`	Transconductance of source (gain)	double	0	Ω^{-1}

Voltage-Controlled Voltage Source

The positive and negative nodes are `V+` and `V-`, respectively. The positive and negative controlling nodes are `VC+` and `VC-`, respectively. The value of the voltage is given by:

$$v = gain \cdot (v(VC+) - v(VC-)) \tag{6}$$

Device name:	`VCVS`
Default parameter set name:	`VCVS_pset`
Electrodes:	`V+, V-, VC+, VC-`
Internal variables:	`branch` (current through voltage source)

NOTE There are no parameters for this parameter set.

Table 21 Voltage-controlled voltage source instance parameter

Name	Description	Type	Default	Unit
gain	Voltage gain	double	0	–

Current-Controlled Current Source

The positive and negative nodes are `F+` and `F-`, respectively. The current flows from the positive node, through the source, to the negative node. The parameter `control` identifies the controlling voltage source, which must have been previously declared. The direction of the positive controlling current flow is from the positive node, through the voltage source `control`, to the negative node. The value of the current is given by:

$$i = gain \cdot i(control) \tag{7}$$

Device name:	`CCCS`
Default parameter set name:	`CCCS_pset`
Electrodes:	`F+, F-`
Internal variables:	None

NOTE There are no parameters for this parameter set.

Table 22 Current-controlled current source instance parameters

Name	Description	Type	Default	Unit
control	Name of controlling source	string	“ ”	–
gain	Current gain	double	0	–

Current-Controlled Voltage Source

`H+` and `H–` are the positive and negative nodes, respectively. The parameter `control` identifies the controlling voltage source, which must have been previously declared. The direction of the positive controlling current flow is from the positive node, through the voltage source `control`, to the negative node. The value of the current is given by:

$$v = gain \cdot i(control) \tag{8}$$

Device name:	`CCVS`
Default parameter set name:	`CCVS_pset`
Electrodes:	`H+, H-`
Internal variables:	`branch` (current through voltage source)

NOTE There are no parameters for this parameter set.

Table 23 Current-controlled voltage source instance parameters

Name	Description	Type	Default	Unit
`control`	Controlling voltage source	string	“ ”	–
`gain`	Transresistance (gain)	double	0	Ω

MOSFET Models (NMOS and PMOS)

Different SPICE MOSFET models are available: Mos1, Mos2, Mos3, Mos6, BSIM1, BSIM2, BSIM3, BSIM4, and BSIMPD2.2.

The DC characteristics are defined by the device parameters `vto`, `kp`, `lambda`, `phi`, and `gamma`. These parameters are computed by SPICE if process parameters (`nsub`, `tox`, ...) are given, but user-specified values always override. `vto` is positive (negative) for enhancement mode and negative (positive) for depletion mode n-channel (p-channel) devices.

Charge storage is modeled by the constant capacitors `cgso`, `cgdo`, and `cgbo`, which represent overlap capacitances by the nonlinear thin-oxide capacitance that is distributed among the gate, source, drain, and bulk regions, and by the nonlinear depletion-layer capacitances for both substrate junctions divided into the bottom and the periphery, which vary as the `mj` and `mjsw` power of junction voltage, respectively, and are determined by the parameters `cbd`, `cbs`, `cj`, `cjsw`, `mj`, `mjsw`, and `pb`. Charge storage effects are modeled by the piecewise, linear, voltage-dependent capacitance model proposed by Meyer. The thin-oxide charge-storage effects are treated differently for the Mos1 model. These voltage-dependent capacitances are included only if `tox` is specified in the input description. These capacitances are represented using the Meyer formulation.

There is some overlap among the parameters that describe the junctions, for example, the reverse current can be input as either `is` [A] or `js` $[\mathrm{A/m^2}]$. Whereas, the first is an absolute value, the second is multiplied by `ad` and `as` to give the reverse current of the drain and source junctions, respectively. The same idea also applies to the zero-bias junction capacitances `cbd` and `cbs` [F] on one hand, and `cj` $[\mathrm{F/m^2}]$ on the other hand. The parasitic drain and source series resistance can be expressed as either `rd` and `rs` $[\Omega]$ or `rhs` $[\Omega/\mathrm{sq}]$, the latter is multiplied by the number of squares `nrd` and `nrs`.

The BSIM1, BSIM2, and BSIM3 parameters are all values obtained from process characterization. Various parameters also have corresponding parameters with length and width dependencies. For example, consider the parameter `vfb` (flat-band voltage) [V]. It is accompanied by the parameters `lvfb` and `wvfb` $[\mathrm{V/\mu m}]$. The effective flat-band voltage is then computed by:

$$vfb_{eff} = vfb + 10^{-6} \cdot \left(\frac{lvfb}{l_{eff}} + \frac{wvfb}{w_{eff}}\right) \tag{9}$$

where the effective lengths and widths are given by:

$$\begin{aligned} l_{eff} &= l - dl \cdot 10^{-6} \\ w_{eff} &= w - dw \cdot 10^{-6} \end{aligned} \tag{10}$$

Level 1 MOSFET Model and Meyer Capacitance Model

The Mos1 model is described by a square-law I–V characteristic. `l` and `w` are the channel length and width. `ad` and `as` are the areas of the drain and source diffusions. `pd` and `ps` are the perimeters of the drain and source junctions. `nrd` and `nrs` designate the equivalent number of squares of the drain and source diffusions. These values multiply the sheet resistance `rsh` for an accurate representation of the parasitic series drain and source resistance of each transistor. The `temp` value is the temperature at which the device will operate.

Use `nmos=1` to specify an NMOS transistor or `pmos=1` to specify a PMOS transistor.

Device name:	`Mos1`
Default parameter set name:	`Mos1_pset`
Electrodes:	`Drain, Gate, Source, Bulk`
Internal variables:	`drain` (internal drain voltage, only available if `rd` $\neq 0$ or `rsh` $\neq 0$ and `nrd` $\neq 0$) `source` (internal source voltage, only available if `rs` $\neq 0$ or `rsh` $\neq 0$ and `nrs` $\neq 0$)

Table 24 Mos1 model parameters

Name	Description	Type	Default	Unit
`vto`	Threshold voltage	double	0	V
`vt0`	(redundant parameter)	double	0	V
`kp`	Transconductance parameter	double	2e-05	A/V^2
`gamma`	Bulk threshold parameter	double	0	$V^{1/2}$

Table 24 Mos1 model parameters (Continued)

Name	Description	Type	Default	Unit
phi	Surface potential	double	0.6	V
lambda	Channel length modulation	double	0	V^{-1}
rd	Drain Ohmic resistance	double	0	Ω
rs	Source Ohmic resistance	double	0	Ω
cbd	Base–drain junction capacitance	double	0	F
cbs	Base–source junction capacitance	double	0	F
is	Bulk junction saturation current	double	1e-14	A
pb	Bulk junction potential	double	0.8	V
cgso	Gate–source overlap capacitance	double	0	F/m
cgdo	Gate–drain overlap capacitance	double	0	F/m
cgbo	Gate–bulk overlap capacitance	double	0	F/m
rsh	Sheet resistance	double	0	Ω /sq
cj	Bottom junction capacitance per area	double	0	F/m^2
mj	Bottom grading coefficient	double	0.5	–
cjsw	Side junction capacitance per area	double	0	F/m
mjsw	Side grading coefficient	double	0.5	–
js	Bulk junction saturation current density	double	0	A/m^2
tox	Oxide thickness	double	0	m
ld	Lateral diffusion	double	0	m
u0	Surface mobility	double	0	$cm^2/V/s$
uo	(redundant parameter)	double	0	$cm^2/V/s$
fc	Forward bias junction fit parameter	double	0.5	–
nmos	N-type MOSFET model	integer	1	–
pmos	P-type MOSFET model	integer	0	–
nsub	Substrate doping	double	0	cm^{-3}
tpg	Gate type	integer	0[a]	–
nss	Surface state density	double	0	cm^{-2}
tnom	Parameter measurement temperature	double	27	°C

Table 24 Mos1 model parameters (Continued)

Name	Description	Type	Default	Unit
kf	Flicker noise coefficient	double	0	–
af	Flicker noise exponent	double	1	–

a. 1: opposite to substrate; –1: same as substrate; 0: Al gate.

Table 25 Mos1 instance parameters

Name	Description	Type	Default	Unit
l	Length	double	0.0001	m
w	Width	double	0.0001	m
ad	Drain area	double	0	m^2
as	Source area	double	0	m^2
pd	Drain perimeter	double	0	m
ps	Source perimeter	double	0	m
nrd	Drain squares	double	1	–
nrs	Source squares	double	1	–
off	Device initially off	integer	–	–
icvds	Initial drain–source voltage	double	0	V
icvgs	Initial gate–source voltage	double	0	V
icvbs	Initial base–source voltage	double	0	V
temp	Instance temperature	double	27	°C
ic	Vector of D–S, G–S, B–S voltages	double[3]	–	V

Level 2 MOSFET Model and Meyer Capacitance Model

The Mos2 model is an analytic model [1]. l and w are the channel length and width. ad and as are the areas of the drain and source diffusions. pd and ps are the perimeters of the drain and source junctions. nrd and nrs designate the equivalent number of squares of the drain and source diffusions. These values multiply the sheet resistance rsh for an accurate representation of the parasitic series drain and source resistance of each transistor. The temp value is the temperature at which the device will operate.

Use nmos=1 to specify an NMOS transistor or pmos=1 to specify a PMOS transistor.

Device name:	`Mos2`
Default parameter set name:	`Mos2_pset`
Electrodes:	`Drain, Gate, Source, Bulk`
Internal variables:	`internal#drain` (internal drain voltage, only available if `rd` $\neq 0$ or `rsh` $\neq 0$ and `nrd` $\neq 0$) `internal#source` (internal source voltage, only available if `rs` $\neq 0$ or `rsh` $\neq 0$ and `nrs` $\neq 0$)

Table 26 Mos2 model parameters

Name	**Description**	**Type**	**Default**	**Unit**
`vto`	Threshold voltage	double	0	V
`vt0`	(redundant parameter)	double	0	V
`kp`	Transconductance parameter	double	2.07189e-05	A/V^2
`gamma`	Bulk threshold parameter	double	0	$V^{1/2}$
`phi`	Surface potential	double	0.6	V
`lambda`	Channel length modulation	double	0	V^{-1}
`rd`	Drain Ohmic resistance	double	0	Ω
`rs`	Source Ohmic resistance	double	0	Ω
`cbd`	Base–drain junction capacitance	double	0	F
`cbs`	Base–source junction capacitance	double	0	F
`is`	Bulk junction saturation current	double	1e-14	A
`pb`	Bulk junction potential	double	0.8	V
`cgso`	Gate–source overlap capacitance	double	0	F/m
`cgdo`	Gate–drain overlap capacitance	double	0	F/m
`cgbo`	Gate–bulk overlap capacitance	double	0	F/m
`rsh`	Sheet resistance	double	0	Ω /sq
`cj`	Bottom junction capacitance per area	double	0	F/m^2
`mj`	Bottom grading coefficient	double	0.5	–
`cjsw`	Side junction capacitance per area	double	0	F/m
`mjsw`	Side grading coefficient	double	0.33	–
`js`	Bulk junction saturation current density	double	0	A/m^2
`tox`	Oxide thickness	double	1e-07	m

Table 26 Mos2 model parameters (Continued)

Name	Description	Type	Default	Unit
ld	Lateral diffusion	double	0	m
u0	Surface mobility	double	600	$cm^2/V/s$
uo	(redundant parameter)	double	600	$cm^2/V/s$
fc	Forward bias junction fit parameter	double	0.5	–
nmos	N-type MOSFET model	integer	1	–
pmos	P-type MOSFET model	integer	0	–
nsub	Substrate doping	double	0	cm^{-3}
tpg	Gate type	integer	0[a]	–
nss	Surface state density	double	0	cm^{-2}
delta	Width effect on threshold	double	0	–
uexp	Critical field exp. for mobility degradation	double	0	–
ucrit	Critical field for mobility degradation	double	10000	V/cm
vmax	Maximum carrier drift velocity	double	0	m/s
xj	Junction depth	double	0	m
neff	Total channel charge coefficient	double	1	–
nfs	Fast surface state density	double	0	cm^{-2}
tnom	Parameter measurement temperature	double	27	°C
kf	Flicker noise coefficient	double	0	–
af	Flicker noise exponent	double	1	–

a. 1: opposite to substrate; –1: same as substrate; 0: Al gate.

Table 27 Mos2 instance parameters

Name	Description	Type	Default	Unit
l	Length	double	0.0001	m
w	Width	double	0.0001	m
ad	Drain area	double	0	m^2
as	Source area	double	0	m^2
pd	Drain perimeter	double	0	m
ps	Source perimeter	double	0	m
nrd	Drain squares	double	1	–

Table 27 Mos2 instance parameters (Continued)

Name	Description	Type	Default	Unit
`nrs`	Source squares	double	1	–
`off`	Device initially off	integer	–	–
`icvds`	Initial drain–source voltage	double	0	V
`icvgs`	Initial gate–source voltage	double	0	V
`icvbs`	Initial base–source voltage	double	0	V
`temp`	Instance operating temperature	double	27	°C
`ic`	Vector of D–S, G–S, B–S voltages	double[3]	–	V

Level 3 MOSFET Model and Meyer Capacitance Model

The Mos3 model is a semiempirical model [1]. `l` and `w` are the channel length and width. `ad` and `as` are the areas of the drain and source diffusions. `pd` and `ps` are the perimeters of the drain and source junctions. `nrd` and `nrs` designate the equivalent number of squares of the drain and source diffusions. These values multiply the sheet resistance `rsh` for an accurate representation of the parasitic series drain and source resistance of each transistor. The `temp` value is the temperature at which the device will operate.

Use `nmos=1` to specify an NMOS transistor or `pmos=1` to specify a PMOS transistor.

Device name:	`Mos3`
Default parameter set name:	`Mos3_pset`
Electrodes:	`Drain, Gate, Source, Bulk`
Internal variables:	`internal#drain` (internal drain voltage, only available if $\mathtt{rd} \neq 0$ or $\mathtt{rsh} \neq 0$ and $\mathtt{nrd} \neq 0$) `internal#source` (internal source voltage, only available if $\mathtt{rs} \neq 0$ or $\mathtt{rsh} \neq 0$ and $\mathtt{nrs} \neq 0$)

Table 28 Mos3 model parameters

Name	Description	Type	Default	Unit
`nmos`	N-type MOSFET model	integer	1	–
`pmos`	P-type MOSFET model	integer	0	–
`vto`	Threshold voltage	double	0	V
`vt0`	(redundant parameter)	double	0	V
`kp`	Transconductance parameter	double	2.07189e-05	A/V^2

Table 28 Mos3 model parameters (Continued)

Name	Description	Type	Default	Unit
gamma	Bulk threshold parameter	double	0	$V^{1/2}$
phi	Surface potential	double	0.6	V
rd	Drain Ohmic resistance	double	0	Ω
rs	Source Ohmic resistance	double	0	Ω
cbd	Base–drain junction capacitance	double	0	F
cbs	Base–source junction capacitance	double	0	F
is	Bulk junction saturation current	double	1e-14	A
pb	Bulk junction potential	double	0.8	V
cgso	Gate–source overlap capacitance	double	0	F/m
cgdo	Gate–drain overlap capacitance	double	0	F/m
cgbo	Gate–bulk overlap capacitance	double	0	F/m
rsh	Sheet resistance	double	0	Ω /sq
cj	Bottom junction capacitance per area	double	0	F/m^2
mj	Bottom grading coefficient	double	0.5	–
cjsw	Side junction capacitance per area	double	0	F/m
mjsw	Side grading coefficient	double	0.33	–
js	Bulk junction saturation current density	double	0	A/m^2
tox	Oxide thickness	double	1e-07	m
ld	Lateral diffusion	double	0	m
u0	Surface mobility	double	600	$cm^2/V/s$
uo	(redundant parameter)	double	600	$cm^2/V/s$
fc	Forward bias junction fit parameter	double	0.5	–
nsub	Substrate doping	double	0	cm^{-3}
tpg	Gate type	integer	0[a]	–
nss	Surface state density	double	0	cm^{-2}
vmax	Maximum carrier drift velocity	double	0	m/s
xj	Junction depth	double	0	m
nfs	Fast surface state density	double	0	cm^{-2}
xd	Depletion layer width	double	0	–

Table 28 Mos3 model parameters (Continued)

Name	Description	Type	Default	Unit
alpha	Alpha	double	0	–
eta	V_{ds} dependence of threshold voltage	double	0	–
delta	Width effect on threshold	double	0	–
input_delta	(redundant parameter)	double	0	–
theta	V_{gs} dependence on mobility	double	0	V^{-1}
kappa	Kappa	double	0.2	–
tnom	Parameter measurement temperature	double	27	°C
kf	Flicker noise coefficient	double	0	–
af	Flicker noise exponent	double	1	–

a. 1: opposite to substrate; –1: same as substrate; 0: Al gate.

Table 29 Mos3 instance parameters

Name	Description	Type	Default	Unit
l	Length	double	0.0001	m
w	Width	double	0.0001	m
ad	Drain area	double	0	m^2
as	Source area	double	0	m^2
pd	Drain perimeter	double	0	m
ps	Source perimeter	double	0	m
nrd	Drain squares	double	1	–
nrs	Source squares	double	1	–
off	Device initially off	integer	–	–
icvds	Initial drain–source voltage	double	0	V
icvgs	Initial gate–source voltage	double	0	V
icvbs	Initial base–source voltage	double	0	V
ic	Vector of D–S, G–S, B–S voltages	double[3]	–	V
temp	Instance operating temperature	double	27	°C

Level 6 MOSFET Model and Meyer Capacitance Model

The Mos6 model [2] is a simple analytic model that is accurate in the short-channel region. `l` and `w` are the channel length and width. `ad` and `as` are the areas of the drain and source diffusions. `pd` and `ps` are the perimeters of the drain and source junctions. `nrd` and `nrs` designate the equivalent number of squares of the drain and source diffusions. These values multiply the sheet resistance `rsh` for an accurate representation of the parasitic series drain and source resistance of each transistor. The `temp` value is the temperature at which the device will operate.

The parameter `ps` in the parameter set was renamed `ps1` to avoid ambiguity with the instance parameter of the same name.

Use `nmos=1` to specify an NMOS transistor or `pmos=1` to specify a PMOS transistor.

Device name:	`Mos6`
Default parameter set name:	`Mos6_pset`
Electrodes:	`Drain, Gate, Source, Bulk`
Internal variables:	`drain` (internal drain voltage, only available if $\texttt{rd} \neq 0$ or $\texttt{rsh} \neq 0$ and $\texttt{nrd} \neq 0$) `source` (internal source voltage, only available if $\texttt{rs} \neq 0$ or $\texttt{rsh} \neq 0$ and $\texttt{nrs} \neq 0$)

Table 30 Mos6 model parameters

Name	Description	Type	Default	Unit
`vto`	Threshold voltage	double	0	V
`vt0`	(redundant parameter)	double	0	V
`kv`	Saturation voltage factor	double	2	–
`nv`	Saturation voltage coefficient	double	0.5	–
`kc`	Saturation current factor	double	5e-05	–
`nc`	Saturation current coefficient	double	1	–
`nvth`	Threshold voltage coefficient	double	0.5	–
`ps1`[a]	Saturation current modification parameter	double	0	–
`gamma`	Bulk threshold parameter	double	0	$V^{1/2}$
`gamma1`	Bulk threshold parameter 1	double	0	–
`sigma`	Static feedback effect parameter	double	0	–

Table 30 Mos6 model parameters (Continued)

Name	**Description**	**Type**	**Default**	**Unit**
`phi`	Surface potential	double	0.6	V
`lambda`	Channel length modulation parameter	double	0	–
`lambda0`	Channel length modulation parameter 0	double	0	–
`lambda1`	Channel length modulation parameter 1	double	0	–
`rd`	Drain Ohmic resistance	double	0	Ω
`rs`	Source Ohmic resistance	double	0	Ω
`cbd`	Base–drain junction capacitance	double	0	F
`cbs`	Base–source junction capacitance	double	0	F
`is`	Bulk junction saturation current	double	1e-14	A
`pb`	Bulk junction potential	double	0.8	V
`cgso`	Gate–source overlap capacitance	double	0	F/m
`cgdo`	Gate–drain overlap capacitance	double	0	F/m
`cgbo`	Gate–bulk overlap capacitance	double	0	F/m
`rsh`	Sheet resistance	double	0	Ω /sq
`cj`	Bottom junction capacitance per area	double	0	F/m^2
`mj`	Bottom grading coefficient	double	0.5	–
`cjsw`	Side junction capacitance per area	double	0	F/m
`mjsw`	Side grading coefficient	double	0.5	–
`js`	Bulk junction saturation current density	double	0	A/m^2
`ld`	Lateral diffusion	double	0	m
`tox`	Oxide thickness	double	0	m
`u0`	Surface mobility	double	0	$cm^2/V/s$
`uo`	(redundant parameter)	double	0	$cm^2/V/s$
`fc`	Forward bias junction fit parameter	double	0.5	–
`nmos`	N-type MOSFET model	integer	1	–
`pmos`	P-type MOSFET model	integer	0	–
`tpg`	Gate type	integer	0[b]	–
`nsub`	Substrate doping	double	0	cm^{-3}

Table 30 Mos6 model parameters (Continued)

Name	Description	Type	Default	Unit
nss	Surface state density	double	0	cm^{-2}
tnom	Parameter measurement temperature	double	27	°C

a. Original SPICE name: ps.
b. 1: opposite to substrate; –1: same as substrate; 0: Al gate.

Table 31 Mos6 instance parameters

Name	Description	Type	Default	Unit
l	Length	double	0.0001	m
w	Width	double	0.0001	m
ad	Drain area	double	0	m^2
as	Source area	double	0	m^2
pd	Drain perimeter	double	0	m
ps	Source perimeter	double	0	m
nrd	Drain squares	double	0	–
nrs	Source squares	double	0	–
off	Device initially off	integer	–	–
icvds	Initial drain–source voltage	double	0	V
icvgs	Initial gate–source voltage	double	0	V
icvbs	Initial base–source voltage	double	0	V
temp	Instance temperature	double	27	°C
ic	Vector of D–S, G–S, B–S voltages	double[3]	–	V

Berkeley Short-Channel IGFET Model (BSIM1)

The BSIM1 model [3][4][5] is a Berkeley short-channel IGFET model. In SPICE, this model is sometimes called a level 4 MOSFET model. `l` and `w` are the channel length and width. `ad` and `as` are the areas of the drain and source diffusions. `pd` and `ps` are the perimeters of the drain and source junctions. `nrd` and `nrs` designate the equivalent number of squares of the drain and source diffusions. These values multiply the sheet resistance `rsh` for an accurate representation of the parasitic series drain and source resistance of each transistor.

Use `nmos=1` to specify an NMOS transistor or `pmos=1` to specify a PMOS transistor.

Device name:	`BSIM1`
Default parameter set name:	`BSIM1_pset`
Electrodes:	`Drain, Gate, Source, Bulk`
Internal variables:	`drain` (internal drain voltage, only available if `rsh` $\neq 0$ and `nrd` $\neq 0$) `source` (internal source voltage, only available if `rsh` $\neq 0$ and `nrs` $\neq 0$)

Table 32 BSIM1 model parameters

Name	Description	Type	Default	Unit
vfb	Flat-band voltage	double	0	V
lvfb	Length dependence of vfb	double	0	V µm
wvfb	Width dependence of vfb	double	0	V µm
phi	Strong inversion surface potential	double	0	V
lphi	Length dependence of phi	double	0	V µm
wphi	Width dependence of phi	double	0	V µm
k1	Bulk effect coefficient 1	double	0	$V^{1/2}$
lk1	Length dependence of k1	double	0	$V^{1/2}$ µm
wk1	Width dependence of k1	double	0	$V^{1/2}$ µm
k2	Bulk effect coefficient 2	double	0	–
lk2	Length dependence of k2	double	0	µm
wk2	Width dependence of k2	double	0	µm
eta	V_{ds} dependence of threshold voltage	double	0	–
leta	Length dependence of eta	double	0	µm
weta	Width dependence of eta	double	0	µm
x2e	V_{bs} dependence of eta	double	0	V^{-1}
lx2e	Length dependence of x2e	double	0	V^{-1} µm
wx2e	Width dependence of x2e	double	0	V^{-1} µm
x3e	V_{ds} dependence of eta	double	0	V^{-1}
lx3e	Length dependence of x3e	double	0	V^{-1} µm
wx3e	Width dependence of x3e	double	0	V^{-1} µm

Table 32 BSIM1 model parameters (Continued)

Name	Description	Type	Default	Unit
dl	Channel length reduction	double	0	μm
dw	Channel width reduction	double	0	μm
muz	Zero field mobility at VDS=0 VGS=VTH	double	0	$cm^2/V/s$
x2mz	V_{bs} dependence of muz	double	0	$cm^2/V^2/s$
lx2mz	Length dependence of x2mz	double	0	$cm^2/V^2/s$ μm
wx2mz	Width dependence of x2mz	double	0	$cm^2/V^2/s$ μm
mus	Mobility at VDS=VDD VGS=VTH, channel length modulation	double	0	$cm^2/V^2/s$
lmus	Length dependence of mus	double	0	$cm^2/V^2/s$ μm
wmus	Width dependence of mus	double	0	$cm^2/V^2/s$ μm
x2ms	V_{bs} dependence of mus	double	0	$cm^2/V^2/s$
lx2ms	Length dependence of x2ms	double	0	$cm^2/V^2/s$ μm
wx2ms	Width dependence of x2ms	double	0	$cm^2/V^2/s$ μm
x3ms	V_{ds} dependence of mus	double	0	$cm^2/V^2/s$
lx3ms	Length dependence of x3ms	double	0	$cm^2/V^2/s$ μm
wx3ms	Width dependence of x3ms	double	0	$cm^2/V^2/s$ μm
u0	V_{gs} dependence of mobility	double	0	V^{-1}
lu0	Length dependence of u0	double	0	V^{-1} μm
wu0	Width dependence of u0	double	0	V^{-1} μm
x2u0	V_{bs} dependence of u0	double	0	V^{-2}
lx2u0	Length dependence of x2u0	double	0	V^{-2} μm
wx2u0	Width dependence of x2u0	double	0	V^{-2} μm
u1	V_{ds} dependence of mobility, velocity saturation	double	0	μm /V
lu1	Length dependence of u1	double	0	μm /V μm
wu1	Width dependence of u1	double	0	μm /V μm
x2u1	V_{bs} dependence of u1	double	0	μm V^{-2}
lx2u1	Length dependence of x2u1	double	0	μm V^{-2} μm
wx2u1	Width dependence of x2u1	double	0	μm V^{-2} μm

Table 32 BSIM1 model parameters (Continued)

Name	Description	Type	Default	Unit
x3u1	V_{ds} dependence of u1	double	0	$\mu m\ V^{-2}$
lx3u1	Length dependence of x3u1	double	0	$\mu m\ V^{-2}\ \mu m$
wx3u1	Width dependence of x3u1	double	0	$\mu m\ V^{-2}\ \mu m$
n0	Subthreshold slope	double	0	–
ln0	Length dependence of n0	double	0	μm
wn0	Width dependence of n0	double	0	μm
nb	V_{bs} dependence of subthreshold slope	double	0	–
lnb	Length dependence of nb	double	0	μm
wnb	Width dependence of nb	double	0	μm
nd	V_{ds} dependence of subthreshold slope	double	0	–
lnd	Length dependence of nd	double	0	μm
wnd	Width dependence of nd	double	0	μm
tox	Gate oxide thickness	double	0	μm
temp	Temperature	double	0	°C
vdd	Supply voltage to specify mus	double	0	V
cgso	Gate–source overlap capacitance per unit channel width [m]	double	0	F/m
cgdo	Gate–drain overlap capacitance per unit channel width [m]	double	0	F/m
cgbo	Gate–bulk overlap capacitance per unit channel length [m]	double	0	F/m
xpart	Flag for channel charge partitioning	integer	0[a]	–
rsh	Source–drain diffusion sheet resistance	double	0	Ω /sq
js	Source–drain junction saturation current per unit area	double	0	A/m^2
pb	Source–drain junction built-in potential	double	0.1	V
mj	Source–drain bottom junction capacitance grading coefficient	double	0	–
pbsw	Source–drain side junction capacitance built-in potential	double	0.1	V
mjsw	Source–drain side junction capacitance grading coefficient	double	0	–

Table 32 BSIM1 model parameters (Continued)

Name	Description	Type	Default	Unit
cj	Source–drain bottom junction capacitance per unit area	double	0	F/m^2
cjsw	Source–drain side junction capacitance per unit area	double	0	F/m^2
wdf	Default width of source–drain diffusion in µm	double	0	m
dell	Length reduction of source–drain diffusion	double	0	m
nmos	Flag to indicate NMOS	integer	1	–
pmos	Flag to indicate PMOS	integer	0	–

a. The parameter `xpart` describes the channel charge partitioning. `xpart=0` selects a 40/60 drain/source charge partition in saturation, while `xpart=1` selects a 0/100 drain/source charge partition.

Table 33 BSIM1 instance parameters

Name	Description	Type	Default	Unit
l	Length	double	5e-06	m
w	Width	double	5e-06	m
ad	Drain area	double	0	m^2
as	Source area	double	0	m^2
pd	Drain perimeter	double	0	m
ps	Source perimeter	double	0	m
nrd	Number of squares in drain	double	1	–
nrs	Number of squares in source	double	1	–
off	Device is initially off	integer	0	–
vds	Initial drain–source voltage	double	0	V
vgs	Initial gate–source voltage	double	0	V
vbs	Initial base–source voltage	double	0	V
ic	Vector of D–S, G–S, B–S initial voltages	double[3]	–	V

Berkeley Short-Channel IGFET Model (BSIM2)

The BSIM2 model [6] is a Berkeley short-channel IGFET model. In SPICE, this model is sometimes called a level 5 MOSFET model. `l` and `w` are the channel length and width. `ad` and `as` are the areas of the drain and source diffusions. `pd` and `ps` are the perimeters of the drain and source junctions. `nrd` and `nrs` designate the equivalent number of squares of the drain and source diffusions. These values multiply the sheet resistance `rsh` for an accurate representation of the parasitic series drain and source resistance of each transistor.

Use `nmos=1` to specify an NMOS transistor or `pmos=1` to specify a PMOS transistor.

Device name:	`BSIM2`
Default parameter set name:	`BSIM2_pset`
Electrodes:	`Drain, Gate, Source, Bulk`
Internal variables:	`drain` (internal drain voltage, only available if `rsh` $\neq 0$ and `nrd` $\neq 0$) `source` (internal source voltage, only available if `rsh` $\neq 0$ and `nrs` $\neq 0$)

Table 34 BSIM2 model parameters

Name	Description	Type	Default	Unit
`vfb`	Flat-band voltage	double	–1	V
`lvfb`	Length dependence of `vfb`	double	0	V μm
`wvfb`	Width dependence of `vfb`	double	0	V μm
`phi`	Strong inversion surface potential	double	0.75	V
`lphi`	Length dependence of `phi`	double	0	V μm
`wphi`	Width dependence of `phi`	double	0	V μm
`k1`	Bulk effect coefficient 1	double	0.8	$V^{1/2}$
`lk1`	Length dependence of `k1`	double	0	$V^{1/2}$ μm
`wk1`	Width dependence of `k1`	double	0	$V^{1/2}$ μm
`k2`	Bulk effect coefficient 2	double	0	–
`lk2`	Length dependence of `k2`	double	0	μm
`wk2`	Width dependence of `k2`	double	0	μm
`eta0`	V_{ds} dependence of threshold voltage at VDD=0	double	0	–
`leta0`	Length dependence of `eta0`	double	0	μm

Table 34 BSIM2 model parameters (Continued)

Name	Description	Type	Default	Unit
weta0	Width dependence of eta0	double	0	µm
etab	V_{bs} dependence of eta	double	0	–
letab	Length dependence of etab	double	0	–
wetab	Width dependence of etab	double	0	–
dl	Channel length reduction	double	0	µm
dw	Channel width reduction	double	0	µm
mu0	Low-field mobility, at VDS=0 VGS=VTH	double	400	$cm^2/V^2/s$
mu0b	V_{bs} dependence of low-field mobility	double	0	–
lmu0b	Length dependence of mu0b	double	0	–
wmu0b	Width dependence of mu0b	double	0	–
mus0	Mobility at VDS=VDD VGS=VTH	double	500	$cm^2/V^2/s$
lmus0	Length dependence of mus0	double	0	–
wmus0	Width dependence of mus	double	0	–
musb	V_{bs} dependence of mus	double	0	–
lmusb	Length dependence of musb	double	0	–
wmusb	Width dependence of musb	double	0	–
mu20	V_{ds} dependence of mu in tanh term	double	1.5	–
lmu20	Length dependence of mu20	double	0	–
wmu20	Width dependence of mu20	double	0	–
mu2b	V_{bs} dependence of mu2	double	0	–
lmu2b	Length dependence of mu2b	double	0	–
wmu2b	Width dependence of mu2b	double	0	–
mu2g	V_{gs} dependence of mu2	double	0	–
lmu2g	Length dependence of mu2g	double	0	–
wmu2g	Width dependence of mu2g	double	0	–
mu30	V_{ds} dependence of mu in linear term	double	10	–
lmu30	Length dependence of mu30	double	0	–
wmu30	Width dependence of mu30	double	0	–
mu3b	V_{bs} dependence of mu3	double	0	–

Table 34 BSIM2 model parameters (Continued)

Name	Description	Type	Default	Unit
lmu3b	Length dependence of mu3b	double	0	–
wmu3b	Width dependence of mu3b	double	0	–
mu3g	V_{gs} dependence of mu3	double	0	–
lmu3g	Length dependence of mu3g	double	0	–
wmu3g	Width dependence of mu3g	double	0	–
mu40	V_{ds} dependence of mu in linear term	double	0	–
lmu40	Length dependence of mu40	double	0	–
wmu40	Width dependence of mu40	double	0	–
mu4b	V_{bs} dependence of mu4	double	0	–
lmu4b	Length dependence of mu4b	double	0	–
wmu4b	Width dependence of mu4b	double	0	–
mu4g	V_{gs} dependence of mu4	double	0	–
lmu4g	Length dependence of mu4g	double	0	–
wmu4g	Width dependence of mu4g	double	0	–
ua0	Linear V_{gs} dependence of mobility	double	0.2	–
lua0	Length dependence of ua0	double	0	–
wua0	Width dependence of ua0	double	0	–
uab	V_{bs} dependence of ua	double	0	–
luab	Length dependence of uab	double	0	–
wuab	Width dependence of uab	double	0	–
ub0	Quadratic V_{gs} dependence of mobility	double	0	–
lub0	Length dependence of ub0	double	0	–
wub0	Width dependence of ub0	double	0	–
ubb	V_{bs} dependence of ub	double	0	–
lubb	Length dependence of ubb	double	0	–
wubb	Width dependence of ubb	double	0	–
u10	V_{ds} dependence of mobility	double	0.1	–
lu10	Length dependence of u10	double	0	–
wu10	Width dependence of u10	double	0	–

Table 34 BSIM2 model parameters (Continued)

Name	Description	Type	Default	Unit
u1b	V_{bs} dependence of u1	double	0	–
lu1b	Length dependence of u1b	double	0	–
wu1b	Width dependence of u1b	double	0	–
u1d	V_{ds} dependence of u1	double	0	–
lu1d	Length dependence of u1d	double	0	–
wu1d	Width dependence of u1d	double	0	–
n0	Subthreshold slope at VDS=0 VBS=0	double	1.4	–
ln0	Length dependence of n0	double	0	–
wn0	Width dependence of n0	double	0	–
nb	V_{bs} dependence of n	double	0.5	–
lnb	Length dependence of nb	double	0	–
wnb	Width dependence of nb	double	0	–
nd	V_{ds} dependence of n	double	0	–
lnd	Length dependence of nd	double	0	–
wnd	Width dependence of nd	double	0	–
vof0	Threshold voltage offset at VDS=0 VBS=0	double	1.8	–
lvof0	Length dependence of vof0	double	0	–
wvof0	Width dependence of vof0	double	0	–
vofb	V_{bs} dependence of vof	double	0	–
lvofb	Length dependence of vofb	double	0	–
wvofb	Width dependence of vofb	double	0	–
vofd	V_{ds} dependence of vof	double	0	–
lvofd	Length dependence of vofd	double	0	–
wvofd	Width dependence of vofd	double	0	–
ai0	Prefactor of hot-electron effect	double	0	–
lai0	Length dependence of ai0	double	0	–
wai0	Width dependence of ai0	double	0	–
aib	V_{bs} dependence of ai	double	0	–
laib	Length dependence of aib	double	0	–

Table 34 BSIM2 model parameters (Continued)

Name	Description	Type	Default	Unit
waib	Width dependence of aib	double	0	–
bi0	Exponential factor of hot-electron effect	double	0	–
lbi0	Length dependence of bi0	double	0	–
wbi0	Width dependence of bi0	double	0	–
bib	V_{bs} dependence of bi	double	0	–
lbib	Length dependence of bib	double	0	–
wbib	Width dependence of bib	double	0	–
vghigh	Upper bound of cubic spline function	double	0.2	–
lvghigh	Length dependence of vghigh	double	0	–
wvghigh	Width dependence of vghigh	double	0	–
vglow	Lower bound of cubic spline function	double	–0.15	–
lvglow	Length dependence of vglow	double	0	–
wvglow	Width dependence of vglow	double	0	–
tox	Gate oxide thickness	double	0.03	µm
temp	Temperature	double	27	°C
vdd	Maximum V_{ds}	double	5	V
vgg	Maximum V_{gs}	double	5	V
vbb	Maximum V_{bs}	double	5	V
cgso	Gate–source overlap capacitance per unit channel width [m]	double	0	F/m
cgdo	Gate–drain overlap capacitance per unit channel width [m]	double	0	F/m
cgbo	Gate–bulk overlap capacitance per unit channel length [m]	double	0	F/m
xpart	Flag for channel charge partitioning	integer	0[a]	–
rsh	Source–drain diffusion sheet resistance	double	0	Ω /sq
js	Source–drain junction saturation current per unit area	double	0	A/m^2
pb	Source–drain junction built-in potential	double	0.1	V
mj	Source–drain bottom junction capacitance grading coefficient	double	0	–

Table 34 BSIM2 model parameters (Continued)

Name	Description	Type	Default	Unit
pbsw	Source–drain side junction capacitance built-in potential	double	0.1	V
mjsw	Source–drain side junction capacitance grading coefficient	double	0	–
cj	Source–drain bottom junction capacitance per unit area	double	0	F/m^2
cjsw	Source–drain side junction capacitance per unit area	double	0	F/m
wdf	Default width of source–drain diffusion	double	10	µm
dell	Length reduction of source–drain diffusion	double	0	m
nmos	Flag to indicate NMOS	integer	1	–
pmos	Flag to indicate PMOS	integer	0	–

a. The parameter `xpart` describes the channel charge partitioning. `xpart=0` selects a 40/60 drain/source charge partition in saturation, while `xpart=1` selects a 0/100 drain/source charge partition.

Table 35 BSIM2 instance parameters

Name	Description	Type	Default	Unit
l	Length	double	5e-06	m
w	Width	double	5e-06	m
ad	Drain area	double	0	m^2
as	Source area	double	0	m^2
pd	Drain perimeter	double	0	m
ps	Source perimeter	double	0	m
nrd	Number of squares in drain	double	1	–
nrs	Number of squares in source	double	1	–
off	Device is initially off	integer	0	–
vds	Initial drain–source voltage	double	0	V
vgs	Initial gate–source voltage	double	0	V
vbs	Initial base–source voltage	double	0	V
ic	Vector of D–S, G–S, B–S initial voltages	double[3]	–	V

Berkeley Short-Channel IGFET Model Version 3 (BSIM3)

The BSIM3 model is the latest physics-based MOSFET SPICE model for circuit simulation and CMOS technology development. In SPICE, this model is also called a level 8 MOSFET model.

Use `nmos=1` to specify an NMOS transistor or `pmos=1` to specify a PMOS transistor.

Device name:	`BSIM3`
Default parameter set name:	`BSIM3_pset`
Electrodes:	`Drain, Gate, Source, Bulk`
Internal variables:	`drain` (internal drain voltage, only available if `rsh` > 0 and `nrd` > 0) `source` (internal source voltage, only available if `rsh` > 0 and `nrs` > 0) `charge` (internal charge node, only available if `nqsmod` $\neq 0$)

Table 36 BSIM3 model parameters

Name	Description	Type	Default	Unit
`capmod`	Capacitance model selector	integer	3	–
`mobmod`	Mobility model selector	integer	1	–
`noimod`	Noise model selector	integer	1	–
`paramchk`	Model parameter checking selector	integer	0	–
`binunit`	Bin unit selector	integer	1	–
`version`	Parameter for model version	double	3.2	–
`tox`	Gate oxide thickness	double	1.5e-08	m
`toxm`	Gate oxide thickness used in extraction	double	1.5e-08	m
`cdsc`	Drain–source and channel coupling capacitance	double	0.00024	F/m^2
`cdscb`	Body-bias dependence of `cdsc`	double	0	$F/V/m^2$
`cdscd`	Drain-bias dependence of `cdsc`	double	0	$F/V/m^2$
`cit`	Interface state capacitance	double	0	F/m^2
`nfactor`	Subthreshold swing coefficient	double	1	–
`xj`	Junction depth	double	1.5e-07	m

Table 36 BSIM3 model parameters (Continued)

Name	Description	Type	Default	Unit
`vsat`	Saturation velocity at `tnom`	double	80000	m/s
`at`	Temperature coefficient of `vsat`	double	33000	m/s
`a0`	Nonuniform depletion width effect coefficient	double	1	–
`ags`	Gate bias coefficient of Abulk	double	0	V^{-1}
`a1`	Nonsaturation effect coefficient	double	0	V^{-1}
`a2`	Nonsaturation effect coefficient	double	1	–
`keta`	Body-bias coefficient of nonuniform depletion width effect	double	–0.047	V^{-1}
`nsub`	Substrate doping concentration	double	6e+16	cm^{-3}
`nch`	Channel doping concentration	double	1.7e+17	cm^{-3}
`ngate`	Poly-gate doping concentration	double	0	cm^{-3}
`gamma1`	Vth body coefficient	double	0	$V^{1/2}$
`gamma2`	Vth body coefficient	double	0	$V^{1/2}$
`vbx`	Vth transition body voltage	double	0	V
`vbm`	Maximum body voltage	double	–3	V
`xt`	Doping depth	double	1.55e-07	m
`k1`	Bulk effect coefficient 1	double	0	$V^{1/2}$
`kt1`	Temperature coefficient of Vth	double	–0.11	V
`kt1l`	Temperature coefficient of Vth	double	0	Vm
`kt2`	Body coefficient of `kt1`	double	0.022	–
`k2`	Bulk effect coefficient 2	double	0	–
`k3`	Narrow width effect coefficient	double	80	–
`k3b`	Body effect coefficient of `k3`	double	0	V^{-1}
`w0`	Narrow width effect parameter	double	2.5e-06	m
`nlx`	Lateral nonuniform doping effect	double	1.74e-07	m
`dvt0`	Short-channel effect coefficient 0	double	2.2	–
`dvt1`	Short-channel effect coefficient 1	double	0.53	–
`dvt2`	Short-channel effect coefficient 2	double	–0.032	V^{-1}
`dvt0w`	Narrow width coefficient 0	double	0	m^{-1}

Table 36 BSIM3 model parameters (Continued)

Name	Description	Type	Default	Unit
dvt1w	Narrow width effect coefficient 1	double	5.3e+06	m^{-1}
dvt2w	Narrow width effect coefficient 2	double	–0.032	V^{-1}
drout	DIBL coefficient of output resistance	double	0.56	–
dsub	DIBL coefficient in subthreshold region	double	drout	–
vth0	Threshold voltage	double	0.7[a]	V
vtho	(redundant parameter)	double	0.7	V
ua	Linear gate dependence of mobility	double	2.25e-09	m/V
ua1	Temperature coefficient of ua	double	4.31e-09	m/V
ub	Quadratic gate dependence of mobility	double	5.87e-19	$(m/V)^2$
ub1	Temperature coefficient of ub	double	–7.61e-18	$(m/V)^2$
uc	Body-bias dependence of mobility	double	–4.65e-11[b]	m/V^2
uc1	Temperature coefficient of uc	double	–5.6e-11[c]	m/V^2
u0	Low-field mobility at tnom	double	0.067[d]	$cm^2/V/s$
ute	Temperature coefficient of mobility	double	–1.5	–
voff	Threshold voltage offset	double	–0.08	V
tnom	Parameter measurement temperature	double	27	°C
cgso	Gate–source overlap capacitance per width	double	2.07188e-10	F/m
cgdo	Gate–drain overlap capacitance per width	double	2.07188e-10	F/m
cgbo	Gate–bulk overlap capacitance per length	double	0	F/m
xpart	Channel charge partitioning	double	0[e]	–
elm	Non-quasistatic Elmore constant parameter	double	5	–
delta	Effective V_{ds} parameter	double	0.01	V
rsh	Source–drain sheet resistance	double	0	Ω/sq
rdsw	Source–drain resistance per width	double	0	$\Omega/\mu m^{wr}$
prwg	Gate-bias effect on parasitic resistance	double	0	V^{-1}
prwb	Body effect on parasitic resistance	double	0	$V^{-1/2}$
prt	Temperature coefficient of parasitic resistance	double	0	$\Omega/\mu m$
eta0	Subthreshold region DIBL coefficient	double	0.08	–
etab	Subthreshold region DIBL coefficient	double	–0.07	V^{-1}

Table 36 BSIM3 model parameters (Continued)

Name	Description	Type	Default	Unit
pclm	Channel-length modulation coefficient	double	1.3	–
pdiblc1	Drain-induced barrier-lowering coefficient	double	0.39	–
pdiblc2	Drain-induced barrier-lowering coefficient	double	0.0086	–
pdiblcb	Body effect on drain-induced barrier lowering	double	0	V^{-1}
pscbe1	Substrate current body-effect coefficient	double	4.24e+08	V/m
pscbe2	Substrate current body-effect coefficient	double	1e-05	m/V
pvag	Gate dependence of output resistance parameter	double	0	–
js	Source–drain junction reverse saturation current density	double	0.0001	A/m^2
jsw	Sidewall junction reverse saturation current density	double	0	A/m
pb	Source–drain junction built-in potential	double	1	V
nj	Source–drain junction emission coefficient	double	1	–
xti	Junction current temperature exponent	double	3	–
mj	Source–drain bottom junction capacitance grading coefficient	double	0.5	–
pbsw	Source–drain sidewall junction capacitance built-in potential	double	1	V
mjsw	Source–drain sidewall junction capacitance grading coefficient	double	0.33	–
pbswg	Source–drain (gate side) sidewall junction capacitance built-in potential	double	pbsw	V
mjswg	Source–drain (gate side) sidewall junction capacitance grading coefficient	double	mjsw	–
cj	Source–drain bottom junction capacitance per unit area	double	0.0005	F/m^2
vfbcv	Flat-band voltage parameter for capmod=0 only	double	–1	–
vfb	Flat-band voltage	double	0	V
cjsw	Source–drain sidewall junction capacitance per unit periphery	double	5e-10	F/m
cjswg	Source–drain (gate side) sidewall junction capacitance per unit width	double	cjsw	F/m
tpb	Temperature coefficient of pb	double	0	–

Table 36 BSIM3 model parameters (Continued)

Name	Description	Type	Default	Unit
tcj	Temperature coefficient of cj	double	0	–
tpbsw	Temperature coefficient of pbsw	double	0	–
tcjsw	Temperature coefficient of cjsw	double	0	–
tpbswg	Temperature coefficient of pbswg	double	0	–
tcjswg	Temperature coefficient of cjswg	double	0	–
acde	Exponential coefficient for finite charge thickness	double	1	–
moin	Coefficient for gate bias–dependent surface potential	double	15	–
noff	C–V switch on and off parameter	double	1	–
voffcv	C–V lateral-shift parameter	double	0	–
lint	Length reduction parameter	double	0	m
ll	Length reduction parameter	double	0	m^{lln}
llc	Length reduction parameter for C–V	double	0	–
lln	Length reduction parameter	double	1	–
lw	Length reduction parameter	double	0	m^{lwn}
lwc	Length reduction parameter for C–V	double	0	–
lwn	Length reduction parameter	double	1	–
lwl	Length reduction parameter	double	0	$m^{lwn+lln}$
lwlc	Length reduction parameter for C–V	double	0	–
lmin	Minimum length of model	double	0	m
lmax	Maximum length of model	double	1	m
wr	Width dependence of rds	double	1	–
wint	Width reduction parameter	double	0	m
dwg	Width reduction parameter	double	0	m/V
dwb	Width reduction parameter	double	0	$m/V^{1/2}$
wl	Width reduction parameter	double	0	m^{wln}
wlc	Width reduction parameter for C–V	double	0	–
wln	Width reduction parameter	double	1	–
ww	Width reduction parameter	double	0	m^{wwn}

Table 36 BSIM3 model parameters (Continued)

Name	Description	Type	Default	Unit
`wwc`	Width reduction parameter for C–V	double	0	–
`wwn`	Width reduction parameter	double	1	–
`wwl`	Width reduction parameter	double	0	$m^{wwn+wln}$
`wwlc`	Width reduction parameter for C–V	double	0	–
`wmin`	Minimum width of model	double	0	m
`wmax`	Maximum width of model	double	1	m
`b0`	Abulk narrow width parameter	double	0	m
`b1`	Abulk narrow width parameter	double	0	m
`cgsl`	New C–V model parameter	double	0	F/m
`cgdl`	New C–V model parameter	double	0	F/m
`ckappa`	New C–V model parameter	double	0.6	–
`cf`	Fringe capacitance parameter	double	7.29897e-11	F/m
`clc`	V_{dsat} parameter for C–V model	double	1e-07	m
`cle`	Vdsat parameter for C–V model	double	0.6	–
`dwc`	Delta W for C–V model	double	wint	m
`dlc`	Delta L for C–V model	double	lint	m
`alpha0`	Substrate current model parameter	double	0	m/V
`alpha1`	Substrate current model parameter	double	0	–
`beta0`	Substrate current model parameter	double	30	V
`ijth`	Diode-limiting current	double	0.1	–
`lcdsc`	Length dependence of `cdsc`	double	0	–
`lcdscb`	Length dependence of `cdscb`	double	0	–
`lcdscd`	Length dependence of `cdscd`	double	0	–
`lcit`	Length dependence of `cit`	double	0	–
`lnfactor`	Length dependence of `nfactor`	double	0	–
`lxj`	Length dependence of `xj`	double	0	–
`lvsat`	Length dependence of `vsat`	double	0	–
`lat`	Length dependence of `at`	double	0	–
`la0`	Length dependence of `a0`	double	0	–

Table 36 BSIM3 model parameters (Continued)

Name	Description	Type	Default	Unit
lags	Length dependence of ags	double	0	–
la1	Length dependence of a1	double	0	–
la2	Length dependence of a2	double	0	–
lketa	Length dependence of keta	double	0	–
lnsub	Length dependence of nsub	double	0	–
lnch	Length dependence of nch	double	0	–
lngate	Length dependence of ngate	double	0	–
lgamma1	Length dependence of gamma1	double	0	–
lgamma2	Length dependence of gamma2	double	0	–
lvbx	Length dependence of vbx	double	0	–
lvbm	Length dependence of vbm	double	0	–
lxt	Length dependence of xt	double	0	–
lk1	Length dependence of k1	double	0	–
lkt1	Length dependence of kt1	double	0	–
lkt1l	Length dependence of kt1l	double	0	–
lkt2	Length dependence of kt2	double	0	–
lk2	Length dependence of k2	double	0	–
lk3	Length dependence of k3	double	0	–
lk3b	Length dependence of k3b	double	0	–
lw0	Length dependence of w0	double	0	–
lnlx	Length dependence of nlx	double	0	–
ldvt0	Length dependence of dvt0	double	0	–
ldvt1	Length dependence of dvt1	double	0	–
ldvt2	Length dependence of dvt2	double	0	–
ldvt0w	Length dependence of dvt0w	double	0	–
ldvt1w	Length dependence of dvt1w	double	0	–
ldvt2w	Length dependence of dvt2w	double	0	–
ldrout	Length dependence of drout	double	0	–
ldsub	Length dependence of dsub	double	0	–

Table 36 BSIM3 model parameters (Continued)

Name	Description	Type	Default	Unit
lvth0	Length dependence of vto	double	0	–
lvtho	Length dependence of vto	double	0	–
lua	Length dependence of ua	double	0	–
lua1	Length dependence of ua1	double	0	–
lub	Length dependence of ub	double	0	–
lub1	Length dependence of ub1	double	0	–
luc	Length dependence of uc	double	0	–
luc1	Length dependence of uc1	double	0	–
lu0	Length dependence of u0	double	0	–
lute	Length dependence of ute	double	0	–
lvoff	Length dependence of voff	double	0	–
lelm	Length dependence of elm	double	0	–
ldelta	Length dependence of delta	double	0	–
lrdsw	Length dependence of rdsw	double	0	–
lprwg	Length dependence of prwg	double	0	–
lprwb	Length dependence of prwb	double	0	–
lprt	Length dependence of prt	double	0	–
leta0	Length dependence of eta0	double	0	–
letab	Length dependence of etab	double	0	–
lpclm	Length dependence of pclm	double	0	–
lpdiblc1	Length dependence of pdiblc1	double	0	–
lpdiblc2	Length dependence of pdiblc2	double	0	–
lpdiblcb	Length dependence of pdiblcb	double	0	–
lpscbe1	Length dependence of pscbe1	double	0	–
lpscbe2	Length dependence of pscbe2	double	0	–
lpvag	Length dependence of pvag	double	0	–
lwr	Length dependence of wr	double	0	–
ldwg	Length dependence of dwg	double	0	–
ldwb	Length dependence of dwb	double	0	–

Table 36 BSIM3 model parameters (Continued)

Name	Description	Type	Default	Unit
lb0	Length dependence of b0	double	0	–
lb1	Length dependence of b1	double	0	–
lcgsl	Length dependence of cgsl	double	0	–
lcgdl	Length dependence of cgdl	double	0	–
lckappa	Length dependence of ckappa	double	0	–
lcf	Length dependence of cf	double	0	–
lclc	Length dependence of clc	double	0	–
lcle	Length dependence of cle	double	0	–
lalpha0	Length dependence of alpha0	double	0	–
lalpha1	Length dependence of alpha1	double	0	–
lbeta0	Length dependence of beta0	double	0	–
lvfbcv	Length dependence of vfbcv	double	0	–
lvfb	Length dependence of vfb	double	0	–
lacde	Length dependence of acde	double	0	–
lmoin	Length dependence of moin	double	0	–
lnoff	Length dependence of noff	double	0	–
lvoffcv	Length dependence of voffcv	double	0	–
wcdsc	Width dependence of cdsc	double	0	–
wcdscb	Width dependence of cdscb	double	0	–
wcdscd	Width dependence of cdscd	double	0	–
wcit	Width dependence of cit	double	0	–
wnfactor	Width dependence of nfactor	double	0	–
wxj	Width dependence of xj	double	0	–
wvsat	Width dependence of vsat	double	0	–
wat	Width dependence of at	double	0	–
wa0	Width dependence of a0	double	0	–
wags	Width dependence of ags	double	0	–
wa1	Width dependence of a1	double	0	–
wa2	Width dependence of a2	double	0	–

Table 36 BSIM3 model parameters (Continued)

Name	Description	Type	Default	Unit
wketa	Width dependence of keta	double	0	–
wnsub	Width dependence of nsub	double	0	–
wnch	Width dependence of nch	double	0	–
wngate	Width dependence of ngate	double	0	–
wgamma1	Width dependence of gamma1	double	0	–
wgamma2	Width dependence of gamma2	double	0	–
wvbx	Width dependence of vbx	double	0	–
wvbm	Width dependence of vbm	double	0	–
wxt	Width dependence of xt	double	0	–
wk1	Width dependence of k1	double	0	–
wkt1	Width dependence of kt1	double	0	–
wkt1l	Width dependence of kt1l	double	0	–
wkt2	Width dependence of kt2	double	0	–
wk2	Width dependence of k2	double	0	–
wk3	Width dependence of k3	double	0	–
wk3b	Width dependence of k3b	double	0	–
ww0	Width dependence of w0	double	0	–
wnlx	Width dependence of nlx	double	0	–
wdvt0	Width dependence of dvt0	double	0	–
wdvt1	Width dependence of dvt1	double	0	–
wdvt2	Width dependence of dvt2	double	0	–
wdvt0w	Width dependence of dvt0w	double	0	–
wdvt1w	Width dependence of dvt1w	double	0	–
wdvt2w	Width dependence of dvt2w	double	0	–
wdrout	Width dependence of drout	double	0	–
wdsub	Width dependence of dsub	double	0	–
wvth0	Width dependence of vto	double	0	–
wvtho	Width dependence of vto	double	0	–
wua	Width dependence of ua	double	0	–

Table 36 BSIM3 model parameters (Continued)

Name	Description	Type	Default	Unit
wua1	Width dependence of ua1	double	0	–
wub	Width dependence of ub	double	0	–
wub1	Width dependence of ub1	double	0	–
wuc	Width dependence of uc	double	0	–
wuc1	Width dependence of uc1	double	0	–
wu0	Width dependence of u0	double	0	–
wute	Width dependence of ute	double	0	–
wvoff	Width dependence of voff	double	0	–
welm	Width dependence of elm	double	0	–
wdelta	Width dependence of delta	double	0	–
wrdsw	Width dependence of rdsw	double	0	–
wprwg	Width dependence of prwg	double	0	–
wprwb	Width dependence of prwb	double	0	–
wprt	Width dependence of prt	double	0	–
weta0	Width dependence of eta0	double	0	–
wetab	Width dependence of etab	double	0	–
wpclm	Width dependence of pclm	double	0	–
wpdiblc1	Width dependence of pdiblc1	double	0	–
wpdiblc2	Width dependence of pdiblc2	double	0	–
wpdiblcb	Width dependence of pdiblcb	double	0	–
wpscbe1	Width dependence of pscbe1	double	0	–
wpscbe2	Width dependence of pscbe2	double	0	–
wpvag	Width dependence of pvag	double	0	–
wwr	Width dependence of wr	double	0	–
wdwg	Width dependence of dwg	double	0	–
wdwb	Width dependence of dwb	double	0	–
wb0	Width dependence of b0	double	0	–
wb1	Width dependence of b1	double	0	–
wcgsl	Width dependence of cgsl	double	0	–

Table 36 BSIM3 model parameters (Continued)

Name	Description	Type	Default	Unit
wcgdl	Width dependence of cgdl	double	0	–
wckappa	Width dependence of ckappa	double	0	–
wcf	Width dependence of cf	double	0	–
wclc	Width dependence of clc	double	0	–
wcle	Width dependence of cle	double	0	–
walpha0	Width dependence of alpha0	double	0	–
walpha1	Width dependence of alpha1	double	0	–
wbeta0	Width dependence of beta0	double	0	–
wvfbcv	Width dependence of vfbcv	double	0	–
wvfb	Width dependence of vfb	double	0	–
wacde	Width dependence of acde	double	0	–
wmoin	Width dependence of moin	double	0	–
wnoff	Width dependence of noff	double	0	–
wvoffcv	Width dependence of voffcv	double	0	–
pcdsc	Cross-term dependence of cdsc	double	0	–
pcdscb	Cross-term dependence of cdscb	double	0	–
pcdscd	Cross-term dependence of cdscd	double	0	–
pcit	Cross-term dependence of cit	double	0	–
pnfactor	Cross-term dependence of nfactor	double	0	–
pxj	Cross-term dependence of xj	double	0	–
pvsat	Cross-term dependence of vsat	double	0	–
pat	Cross-term dependence of at	double	0	–
pa0	Cross-term dependence of a0	double	0	–
pags	Cross-term dependence of ags	double	0	–
pa1	Cross-term dependence of a1	double	0	–
pa2	Cross-term dependence of a2	double	0	–
pketa	Cross-term dependence of keta	double	0	–
pnsub	Cross-term dependence of nsub	double	0	–
pnch	Cross-term dependence of nch	double	0	–

Table 36 BSIM3 model parameters (Continued)

Name	Description	Type	Default	Unit
pngate	Cross-term dependence of ngate	double	0	–
pgamma1	Cross-term dependence of gamma1	double	0	–
pgamma2	Cross-term dependence of gamma2	double	0	–
pvbx	Cross-term dependence of vbx	double	0	–
pvbm	Cross-term dependence of vbm	double	0	–
pxt	Cross-term dependence of xt	double	0	–
pk1	Cross-term dependence of k1	double	0	–
pkt1	Cross-term dependence of kt1	double	0	–
pkt1l	Cross-term dependence of kt1l	double	0	–
pkt2	Cross-term dependence of kt2	double	0	–
pk2	Cross-term dependence of k2	double	0	–
pk3	Cross-term dependence of k3	double	0	–
pk3b	Cross-term dependence of k3b	double	0	–
pw0	Cross-term dependence of w0	double	0	–
pnlx	Cross-term dependence of nlx	double	0	–
pdvt0	Cross-term dependence of dvt0	double	0	–
pdvt1	Cross-term dependence of dvt1	double	0	–
pdvt2	Cross-term dependence of dvt2	double	0	–
pdvt0w	Cross-term dependence of dvt0w	double	0	–
pdvt1w	Cross-term dependence of dvt1w	double	0	–
pdvt2w	Cross-term dependence of dvt2w	double	0	–
pdrout	Cross-term dependence of drout	double	0	–
pdsub	Cross-term dependence of dsub	double	0	–
pvth0	Cross-term dependence of vto	double	0	–
pvtho	Cross-term dependence of vto	double	0	–
pua	Cross-term dependence of ua	double	0	–
pua1	Cross-term dependence of ua1	double	0	–
pub	Cross-term dependence of ub	double	0	–
pub1	Cross-term dependence of ub1	double	0	–

Table 36 BSIM3 model parameters (Continued)

Name	Description	Type	Default	Unit
puc	Cross-term dependence of uc	double	0	–
puc1	Cross-term dependence of uc1	double	0	–
pu0	Cross-term dependence of u0	double	0	–
pute	Cross-term dependence of ute	double	0	–
pvoff	Cross-term dependence of voff	double	0	–
pelm	Cross-term dependence of elm	double	0	–
pdelta	Cross-term dependence of delta	double	0	–
prdsw	Cross-term dependence of rdsw	double	0	–
pprwg	Cross-term dependence of prwg	double	0	–
pprwb	Cross-term dependence of prwb	double	0	–
pprt	Cross-term dependence of prt	double	0	–
peta0	Cross-term dependence of eta0	double	0	–
petab	Cross-term dependence of etab	double	0	–
ppclm	Cross-term dependence of pclm	double	0	–
ppdiblc1	Cross-term dependence of pdiblc1	double	0	–
ppdiblc2	Cross-term dependence of pdiblc2	double	0	–
ppdiblcb	Cross-term dependence of pdiblcb	double	0	–
ppscbe1	Cross-term dependence of pscbe1	double	0	–
ppscbe2	Cross-term dependence of pscbe2	double	0	–
ppvag	Cross-term dependence of pvag	double	0	–
pwr	Cross-term dependence of wr	double	0	–
pdwg	Cross-term dependence of dwg	double	0	–
pdwb	Cross-term dependence of dwb	double	0	–
pb0	Cross-term dependence of b0	double	0	–
pb1	Cross-term dependence of b1	double	0	–
pcgsl	Cross-term dependence of cgsl	double	0	–
pcgdl	Cross-term dependence of cgdl	double	0	–
pckappa	Cross-term dependence of ckappa	double	0	–
pcf	Cross-term dependence of cf	double	0	–

Table 36 BSIM3 model parameters (Continued)

Name	Description	Type	Default	Unit
pclc	Cross-term dependence of clc	double	0	–
pcle	Cross-term dependence of cle	double	0	–
palpha0	Cross-term dependence of alpha0	double	0	–
palpha1	Cross-term dependence of alpha1	double	0	–
pbeta0	Cross-term dependence of beta0	double	0	–
pvfbcv	Cross-term dependence of vfbcv	double	0	–
pvfb	Cross-term dependence of vfb	double	0	–
pacde	Cross-term dependence of acde	double	0	–
pmoin	Cross-term dependence of moin	double	0	–
pnoff	Cross-term dependence of noff	double	0	–
pvoffcv	Cross-term dependence of voffcv	double	0	–
noia	Flicker noise parameter	double	1e+20[f]	–
noib	Flicker noise parameter	double	50000[g]	–
noic	Flicker noise parameter	double	–1.4e-12[h]	–
em	Flicker noise parameter	double	4.1e+07	V/m
ef	Flicker noise frequency exponent	double	1	–
af	Flicker noise exponent	double	1	–
kf	Flicker noise coefficient	double	0	–
nmos	Flag to indicate NMOS	integer	1	–
pmos	Flag to indicate PMOS	integer	0	–

a. NMOS: 0.7; PMOS: –0.7.
b. mobmod=1,2: –4.65e-11; mobmod=3: –0.0465.
c. mobmod=1,2: –5.6e-11; mobmod=3: –0.056.
d. NMOS: 0.067; PMOS: 0.025.
e. The parameter xpart describes the channel charge partitioning. xpart=0 selects a 40/60 drain/source charge partition in saturation, while xpart=1 selects a 0/100 drain/source charge partition.
f. NMOS: 1e20; PMOS: 9.9e18.
g. NMOS: 5e4; PMOS: 2.4e3.
h. NMOS: –1.4e-12; PMOS: 1.4e-12.

Table 37 BSIM3 instance parameters

Name	Description	Type	Default	Unit
`l`	Length	double	5e-06	m
`w`	Width	double	5e-06	m
`ad`	Drain area	double	0	m^2
`as`	Source area	double	0	m^2
`pd`	Drain perimeter	double	0	m
`ps`	Source perimeter	double	0	m
`nrd`	Number of squares in drain	double	1	–
`nrs`	Number of squares in source	double	1	–
`off`	Device is initially off	integer	0	–
`nqsmod`	Non-quasistatic model selector	integer	0	–
`ic`	Vector of D–S, G–S, B–S initial voltages	double[3]	–	V

Berkeley Short-Channel IGFET Model Version 4 (BSIM4)

The BSIM4 model is the fourth version of the Berkeley IGFET model for SPICE.

Device name:	`BSIM4`
Default parameter set name:	`BSIM4_pset`
Electrodes:	`Drain, Gate, Source, Bulk`
Internal variables:	`drain, source, charge`

Table 38 BSIM4 model parameters

Name	Description	Type	Default	Unit
`a0`	Nonuniform depletion width effect coefficient	double	1	–
`a1`	Nonsaturation effect coefficient	double	0	V^{-1}
`a2`	Nonsaturation effect coefficient	double	1	–
`acde`	Exponential coefficient for finite charge thickness	double	1	mV^{-1}
`acnqsmod`	AC NQS model selector	integer	0	–
`af`	Flicker noise exponent	double	1	–
`agidl`	Pre-exponential constant for GIDL	double	0	Ω^{-1}

Table 38 BSIM4 model parameters (Continued)

Name	Description	Type	Default	Unit
ags	Gate bias coefficient of Abulk	double	0	V^{-1}
aigbacc	Parameter for I_{gb}	double	0.43	$F^{0.5}sg^{-0.5}m^{-1}$
aigbinv	Parameter for I_{gb}	double	0.35	$F^{0.5}sg^{-0.5}m^{-1}$
aigc	Parameter for I_{gc}	double	0.43 for NMOS; 0.31 for PMOS	$F^{0.5}sg^{-0.5}m^{-1}$
aigsd	Parameter for $I_{gs,d}$	double	0.43 for NMOS; 0.31 for PMOS	$F^{0.5}sg^{-0.5}m^{-1}$
alpha0	Substrate current model parameter	double	0	Am/V
alpha1	Substrate current model parameter	double	0	A/V
at	Temperature coefficient of `vsat`	double	33000	m/s
b0	Abulk narrow width parameter	double	0	m
b1	Abulk narrow width parameter	double	0	m
beta0	Substrate current model parameter	double	30	V
bgidl	Exponential constant for GIDL	double	2.3e+09	V/m
bigbacc	Parameter for I_{gb}	double	0.054	$F^{0.5}sg^{-0.5}m^{-1}V^{-1}$
bigbinv	Parameter for I_{gb}	double	0.03	$F^{0.5}sg^{-0.5}m^{-1}V^{-1}$
bigc	Parameter for I_{gc}	double	0.054 for NMOS; 0.024 for PMOS	$F^{0.5}sg^{-0.5}m^{-1}V^{-1}$
bigsd	Parameter for $I_{gs,d}$	double	0.054 for NMOS; 0.024 for PMOS	$F^{0.5}sg^{-0.5}m^{-1}V^{-1}$
binunit	Bin unit selector	integer	1	–
bvd	Drain diode breakdown voltage	double	10	V
bvs	Source diode breakdown voltage	double	10	V
capmod	Capacitance model selector	integer	2	–
cdsc	Drain–source and channel coupling capacitance	double	0.00024	Fm^{-2}
cdscb	Body-bias dependence of `cdsc`	double	0	$FV^{-1}m^{-2}$

Table 38 BSIM4 model parameters (Continued)

Name	Description	Type	Default	Unit
cdscd	Drain-bias dependence of cdsc	double	0	$FV^{-1}m^{-2}$
cf	Fringe capacitance parameter	double	1.07725e-10	F/m
cgbo	Gate–bulk overlap capacitance per length	double	0	F/m
cgdl	New C–V model parameter	double	0	F/m
cgdo	Gate–drain overlap capacitance per width	double	1.03594e-09	F/m
cgidl	Parameter for body-bias dependence of GIDL	double	0.5	v^3
cgsl	New C–V model parameter	double	0	F/m
cgso	Gate–source overlap capacitance per width	double	1.03594e-09	F/m
cigbacc	Parameter for I_{gb}	double	0.075	V^{-1}
cigbinv	Parameter for I_{gb}	double	0.006	V^{-1}
cigc	Parameter for I_{gc}	double	0.075 for NMOS; 0.03 for PMOS	V^{-1}
cigsd	Parameter for $I_{gs,d}$	double	0.075 for NMOS; 0.03 for PMOS	V^{-1}
cit	Interface state capacitance	double	0	Fm^{-2}
cjd	Drain bottom junction capacitance per unit area	double	0.0005	Fm^{-2}
cjs	Source bottom junction capacitance per unit area	double	0.0005	Fm^{-2}
cjswd	Drain sidewall junction capacitance per unit periphery	double	5e-10	F/m
cjswgd	Drain (gate side) sidewall junction capacitance per unit width	double	5e-10	F/m
cjswgs	Source (gate side) sidewall junction capacitance per unit width	double	5e-10	F/m
cjsws	Source sidewall junction capacitance per unit periphery	double	5e-10	F/m
ckappad	Drain–gate overlap C–V parameter	double	0.6	V
ckappas	Source–gate overlap C–V parameter	double	0.6	V
clc	V_{dsat} parameter for C–V model	double	1e-07	m

Table 38 BSIM4 model parameters (Continued)

Name	Description	Type	Default	Unit
cle	V_{dsat} parameter for C–V model	double	0.6	–
delta	Effective V_{ds} parameter	double	0.01	V
diomod	Diode IV model selector	integer	1	–
dlc	Delta L for C–V model	double	0	m
dlcig	Delta L for Ig model	double	0	m
dmcg	Distance of mid-contact to gate edge	double	0	m
dmcgt	Distance of mid-contact to gate edge in test structures	double	0	m
dmci	Distance of mid-contact to isolation	double	0	m
dmdg	Distance of mid-diffusion to gate edge	double	0	m
drout	DIBL coefficient of output resistance	double	0.56	–
dsub	DIBL coefficient in subthreshold region	double	0.56	–
dtox	Defined as (toxe - toxp)	double	0	m
dvt0	Short-channel effect coefficient 0	double	2.2	–
dvt0w	Narrow width coefficient 0	double	0	–
dvt1	Short-channel effect coefficient 1	double	0.53	–
dvt1w	Narrow width effect coefficient 1	double	5.3e+06	m^{-1}
dvt2	Short-channel effect coefficient 2	double	–0.032	V^{-1}
dvt2w	Narrow width effect coefficient 2	double	–0.032	V^{-1}
dvtp0	First parameter for Vth shift due to pocket	double	0	m
dvtp1	Second parameter for Vth shift due to pocket	double	0	V^{-1}
dwb	Width reduction parameter	double	0	–
dwc	Delta W for C–V model	double	0	m
dwg	Width reduction parameter	double	0	m/V
dwj	Delta W for source–drain junctions	double	0	–
ef	Flicker noise frequency exponent	double	1	–
egidl	Fitting parameter for band bending	double	0.8	V
eigbinv	Parameter for Si band gap for Igbinv	double	1.1	V
em	Flicker noise parameter	double	4.1e+07	–

Table 38 BSIM4 model parameters (Continued)

Name	Description	Type	Default	Unit
epsrox	Dielectric constant of gate oxide relative to vacuum	double	3.9	–
eta0	Subthreshold region DIBL coefficient	double	0.08	–
etab	Subthreshold region DIBL coefficient	double	–0.07	V^{-1}
eu	Mobility exponent	double	1.67 for NMOS; 1.0 for PMOS	–
fnoimod	Flicker noise model selector	integer	1	–
fprout	Rout degradation coefficient for pocket devices	double	0	$Vm^{-0.5}$
gamma1	Vth body coefficient	double	0	–
gamma2	Vth body coefficient	double	0	–
gbmin	Minimum body conductance	double	1e-12	Ω^{-1}
geomod	Geometry-dependent parasitics model selector	integer	0	–
igbmod	Gate-to-body Ig model selector	integer	0	–
igcmod	Gate-to-channel Ig model selector	integer	0	–
ijthdfwd	Forward drain diode forward-limiting current	double	0.1	A
ijthdrev	Reverse drain diode forward-limiting current	double	0.1	A
ijthsfwd	Forward source diode forward-limiting current	double	0.1	A
ijthsrev	Reverse source diode forward-limiting current	double	0.1	A
jsd	Bottom drain junction reverse saturation current density	double	0.0001	Am^{-2}
jss	Bottom source junction reverse saturation current density	double	0.0001	Am^{-2}
jswd	Isolation edge sidewall drain junction reverse saturation current density	double	0	–
jswgd	Gate edge drain junction reverse saturation current density	double	0	A/m
jswgs	Gate edge source junction reverse saturation current density	double	0	A/m

Table 38 BSIM4 model parameters (Continued)

Name	Description	Type	Default	Unit
jsws	Isolation edge sidewall source junction reverse saturation current density	double	0	–
k1	Bulk effect coefficient 1	double	0	$V^{0.5}$
k2	Bulk effect coefficient 2	double	0	–
k3	Narrow width effect coefficient	double	80	–
k3b	Body effect coefficient of k3	double	0	V^{-1}
keta	Body-bias coefficient of nonuniform depletion width effect	double	–0.047	V^{-1}
kf	Flicker noise coefficient	double	0	–
kt1	Temperature coefficient of Vth	double	–0.11	V
kt1l	Temperature coefficient of Vth	double	0	m
kt2	Body coefficient of kt1	double	0.022	–
la0	Length dependence of a0	double	0	–
la1	Length dependence of a1	double	0	–
la2	Length dependence of a2	double	0	–
lacde	Length dependence of acde	double	0	–
lagidl	Length dependence of agidl	double	0	–
lags	Length dependence of ags	double	0	–
laigbacc	Length dependence of aigbacc	double	0	–
laigbinv	Length dependence of aigbinv	double	0	–
laigc	Length dependence of aigc	double	0	–
laigsd	Length dependence of aigsd	double	0	–
lalpha0	Length dependence of alpha0	double	0	–
lalpha1	Length dependence of alpha1	double	0	–
lat	Length dependence of at	double	0	–
lb0	Length dependence of b0	double	0	–
lb1	Length dependence of b1	double	0	–
lbeta0	Length dependence of beta0	double	0	–
lbgidl	Length dependence of bgidl	double	0	–

Table 38 BSIM4 model parameters (Continued)

Name	Description	Type	Default	Unit
lbigbacc	Length dependence of bigbacc	double	0	–
lbigbinv	Length dependence of bigbinv	double	0	–
lbigc	Length dependence of bigc	double	0	–
lbigsd	Length dependence of bigsd	double	0	–
lcdsc	Length dependence of cdsc	double	0	–
lcdscb	Length dependence of cdscb	double	0	–
lcdscd	Length dependence of cdscd	double	0	–
lcf	Length dependence of cf	double	0	–
lcgdl	Length dependence of cgdl	double	0	–
lcgidl	Length dependence of cgidl	double	0	–
lcgsl	Length dependence of cgsl	double	0	–
lcigbacc	Length dependence of cigbacc	double	0	–
lcigbinv	Length dependence of cigbinv	double	0	–
lcigc	Length dependence of cigc	double	0	–
lcigsd	Length dependence of cigsd	double	0	–
lcit	Length dependence of cit	double	0	–
lckappad	Length dependence of ckappad	double	0	–
lckappas	Length dependence of ckappas	double	0	–
lclc	Length dependence of clc	double	0	–
lcle	Length dependence of cle	double	0	–
ldelta	Length dependence of delta	double	0	–
ldrout	Length dependence of drout	double	0	–
ldsub	Length dependence of dsub	double	0	–
ldvt0	Length dependence of dvt0	double	0	–
ldvt0w	Length dependence of dvt0w	double	0	–
ldvt1	Length dependence of dvt1	double	0	–
ldvt1w	Length dependence of dvt1w	double	0	–
ldvt2	Length dependence of dvt2	double	0	–
ldvt2w	Length dependence of dvt2w	double	0	–

Table 38 BSIM4 model parameters (Continued)

Name	Description	Type	Default	Unit
ldvtp0	Length dependence of dvtp0	double	0	–
ldvtp1	Length dependence of dvtp1	double	0	–
ldwb	Length dependence of dwb	double	0	–
ldwg	Length dependence of dwg	double	0	–
legidl	Length dependence of egidl	double	0	–
leigbinv	Length dependence for eigbinv	double	0	–
leta0	Length dependence of eta0	double	0	–
letab	Length dependence of etab	double	0	–
leu	Length dependence of eu	double	0	–
lfprout	Length dependence of pdiblcb	double	0	–
lgamma1	Length dependence of gamma1	double	0	–
lgamma2	Length dependence of gamma2	double	0	–
lint	Length reduction parameter	double	0	m
lk1	Length dependence of k1	double	0	–
lk2	Length dependence of k2	double	0	–
lk3	Length dependence of k3	double	0	–
lk3b	Length dependence of k3b	double	0	–
lketa	Length dependence of keta	double	0	–
lkt1	Length dependence of kt1	double	0	–
lkt1l	Length dependence of kt1l	double	0	–
lkt2	Length dependence of kt2	double	0	–
ll	Length reduction parameter	double	0	m^{lln}
llc	Length reduction parameter for CV	double	0	–
lln	Length reduction parameter	double	1	–
llpe0	Length dependence of lpe0	double	0	–
llpeb	Length dependence of lpeb	double	0	–
lmax	Maximum length of model	double	1	m
lmin	Minimum length of model	double	0	–
lminv	Length dependence of minv	double	0	m

Table 38 BSIM4 model parameters (Continued)

Name	Description	Type	Default	Unit
lmoin	Length dependence of moin	double	0	–
lndep	Length dependence of ndep	double	0	–
lnfactor	Length dependence of nfactor	double	0	–
lngate	Length dependence of ngate	double	0	–
lnigbacc	Length dependence of nigbacc	double	0	–
lnigbinv	Length dependence of nigbinv	double	0	–
lnigc	Length dependence of nigc	double	0	–
lnoff	Length dependence of noff	double	0	–
lnsd	Length dependence of nsd	double	0	–
lnsub	Length dependence of nsub	double	0	–
lntox	Length dependence of ntox	double	0	–
lpclm	Length dependence of pclm	double	0	–
lpdiblc1	Length dependence of pdiblc1	double	0	–
lpdiblc2	Length dependence of pdiblc2	double	0	–
lpdiblcb	Length dependence of pdiblcb	double	0	–
lpdits	Length dependence of pdits	double	0	–
lpditsd	Length dependence of pditsd	double	0	–
lpe0	Equivalent length of pocket region at zero bias	double	1.74e-07	m
lpeb	Equivalent length of pocket region accounting for body bias	double	0	m
lphin	Length dependence of phin	double	0	–
lpigcd	Length dependence for pigcd	double	0	–
lpoxedge	Length dependence for poxedge	double	0	–
lprt	Length dependence of prt	double	0	–
lprwb	Length dependence of prwb	double	0	–
lprwg	Length dependence of prwg	double	0	–
lpscbe1	Length dependence of pscbe1	double	0	–
lpscbe2	Length dependence of pscbe2	double	0	–

Table 38 BSIM4 model parameters (Continued)

Name	Description	Type	Default	Unit
lpvag	Length dependence of pvag	double	0	–
lrdsw	Length dependence of rdsw	double	0	–
lrdw	Length dependence of rdw	double	0	–
lrsw	Length dependence of rsw	double	0	–
lu0	Length dependence of u0	double	0	–
lua	Length dependence of ua	double	0	–
lua1	Length dependence of ua1	double	0	–
lub	Length dependence of ub	double	0	–
lub1	Length dependence of ub1	double	0	–
luc	Length dependence of uc	double	0	–
luc1	Length dependence of uc1	double	0	–
lute	Length dependence of ute	double	0	–
lvbm	Length dependence of vbm	double	0	–
lvbx	Length dependence of vbx	double	0	–
lvfb	Length dependence of vfb	double	0	–
lvfbcv	Length dependence of vfbcv	double	0	–
lvoff	Length dependence of voff	double	0	–
lvoffcv	Length dependence of voffcv	double	0	–
lvsat	Length dependence of vsat	double	0	–
lvth0	Length dependence of vto	double	0	–
lvtho	Length dependence of vto	double	0	–
lw	Length reduction parameter	double	0	m^{lwn}
lw0	Length dependence of w0	double	0	–
lwc	Length reduction parameter for CV	double	0	–
lwl	Length reduction parameter	double	0	$m^{lln+lwn}$
lwlc	Length reduction parameter for CV	double	0	–
lwn	Length reduction parameter	double	1	–
lwr	Length dependence of wr	double	0	–
lxj	Length dependence of xj	double	0	–

Table 38 BSIM4 model parameters (Continued)

Name	Description	Type	Default	Unit
lxrcrg1	Length dependence of xrcrg1	double	0	–
lxrcrg2	Length dependence of xrcrg2	double	0	–
lxt	Length dependence of xt	double	0	–
minv	Fitting parameter for moderate inversion in Vgsteff	double	0	–
mjd	Drain bottom junction capacitance grading coefficient	double	0.5	–
mjs	Source bottom junction capacitance grading coefficient	double	0.5	–
mjswd	Drain sidewall junction capacitance grading coefficient	double	0.33	–
mjswgd	Drain (gate side) sidewall junction capacitance grading coefficient	double	0.33	–
mjswgs	Source (gate side) sidewall junction capacitance grading coefficient	double	0.33	–
mjsws	Source sidewall junction capacitance grading coefficient	double	0.33	–
mobmod	Mobility model selector	integer	0	–
moin	Coefficient for gate bias–dependent surface potential	double	15	–
ndep	Channel doping concentration at depletion edge	double	1.7e+17	cm^{-3}
nfactor	Subthreshold swing coefficient	double	1	–
ngate	Poly-gate doping concentration	double	0	cm^{-3}
ngcon	Number of gate contacts	double	1	–
nigbacc	Parameter for Igbacc slope	double	1	–
nigbinv	Parameter for Igbinv slope	double	3	–
nigc	Parameter for Igc slope	double	1	–
njd	Drain junction emission coefficient	double	1	–
njs	Source junction emission coefficient	double	1	–
noff	C–V switch on or off parameter	double	1	–

Table 38 BSIM4 model parameters (Continued)

Name	Description	Type	Default	Unit
noia	Flicker noise parameter	double	6.25e+41 NMOS 6.188e+40 PMOS	$s^{1-EF}eV^{-1}m^{-3}$
noib	Flicker noise parameter	double	3.125e+26 NMOS 1.5e+25 PMOS	$s^{1-EF}eV^{-1}m^{-3}$
noic	Flicker noise parameter	double	8.75e+09	$s^{1-EF}F$
nsd	Source–drain doping concentration	double	1e+20	cm^{-3}
nsu	Substrate doping concentration	double	6e+16	cm^{-3}
ntnoi	Thermal noise parameter	double	1	–
ntox	Exponent for Tox ratio	double	1	–
pa0	Cross-term dependence of a0	double	0	–
pa1	Cross-term dependence of a1	double	0	–
pa2	Cross-term dependence of a2	double	0	–
pacde	Cross-term dependence of acde	double	0	–
pagidl	Cross-term dependence of agidl	double	0	–
pags	Cross-term dependence of ags	double	0	–
paigbacc	Cross-term dependence of aigbacc	double	0	–
paigbinv	Cross-term dependence of aigbinv	double	0	–
paigc	Cross-term dependence of aigc	double	0	–
paigsd	Cross-term dependence of aigsd	double	0	–
palpha0	Cross-term dependence of alpha0	double	0	–
palpha1	Cross-term dependence of alpha1	double	0	–
paramchk	Model parameter checking selector	integer	1	–
pat	Cross-term dependence of at	double	0	–
pb0	Cross-term dependence of b0	double	0	–
pb1	Cross-term dependence of b1	double	0	–
pbd	Drain junction built-in potential	double	1	V
pbeta0	Cross-term dependence of beta0	double	0	–

Table 38 BSIM4 model parameters (Continued)

Name	Description	Type	Default	Unit
pbgidl	Cross-term dependence of bgidl	double	0	–
pbigbacc	Cross-term dependence of bigbacc	double	0	–
pbigbinv	Cross-term dependence of bigbinv	double	0	–
pbigc	Cross-term dependence of bigc	double	0	–
pbigsd	Cross-term dependence of bigsd	double	0	–
pbs	Source junction built-in potential	double	1	V
pbswd	Drain sidewall junction capacitance built-in potential	double	1	V
pbswgd	Drain (gate side) sidewall junction capacitance built-in potential	double	1	V
pbswgs	Source (gate side) sidewall junction capacitance built-in potential	double	1	V
pbsws	Source sidewall junction capacitance built-in potential	double	1	V
pcdsc	Cross-term dependence of cdsc	double	0	–
pcdscb	Cross-term dependence of cdscb	double	0	–
pcdscd	Cross-term dependence of cdscd	double	0	–
pcf	Cross-term dependence of cf	double	0	–
pcgdl	Cross-term dependence of cgdl	double	0	–
pcgidl	Cross-term dependence of cgidl	double	0	–
pcgsl	Cross-term dependence of cgsl	double	0	–
pcigbacc	Cross-term dependence of cigbacc	double	0	–
pcigbinv	Cross-term dependence of cigbinv	double	0	–
pcigc	Cross-term dependence of cigc	double	0	–
pcigsd	Cross-term dependence of cigsd	double	0	–
pcit	Cross-term dependence of cit	double	0	–
pckappad	Cross-term dependence of ckappad	double	0	–
pckappas	Cross-term dependence of ckappas	double	0	–
pclc	Cross-term dependence of clc	double	0	–
pcle	Cross-term dependence of cle	double	0	–

Table 38 BSIM4 model parameters (Continued)

Name	Description	Type	Default	Unit
pclm	Channel-length modulation coefficient	double	1.3	–
pdelta	Cross-term dependence of delta	double	0	–
pdiblc1	Drain-induced barrier-lowering coefficient	double	0.39	–
pdiblc2	Drain-induced barrier-lowering coefficient	double	0.0086	–
pdiblcb	Body effect on drain-induced barrier lowering	double	0	V^{-1}
pdits	Coefficient for drain-induced Vth shifts	double	0	V^{-1}
pditsd	Vds dependence of drain-induced Vth shifts	double	0	V^{-1}
pditsl	Length dependence of drain-induced Vth shifts	double	0	m^{-1}
pdrout	Cross-term dependence of drout	double	0	–
pdsub	Cross-term dependence of dsub	double	0	–
pdvt0	Cross-term dependence of dvt0	double	0	–
pdvt0w	Cross-term dependence of dvt0w	double	0	–
pdvt1	Cross-term dependence of dvt1	double	0	–
pdvt1w	Cross-term dependence of dvt1w	double	0	–
pdvt2	Cross-term dependence of dvt2	double	0	–
pdvt2w	Cross-term dependence of dvt2w	double	0	–
pdvtp0	Cross-term dependence of dvtp0	double	0	–
pdvtp1	Cross-term dependence of dvtp1	double	0	–
pdwb	Cross-term dependence of dwb	double	0	–
pdwg	Cross-term dependence of dwg	double	0	–
pegidl	Cross-term dependence of egidl	double	0	–
peigbinv	Cross-term dependence for eigbinv	double	0	–
permod	Pd and Ps model selector	integer	1	–
peta0	Cross-term dependence of eta0	double	0	–
petab	Cross-term dependence of etab	double	0	–
peu	Cross-term dependence of eu	double	0	–
pfprout	Cross-term dependence of pdiblcb	double	0	–

Table 38 BSIM4 model parameters (Continued)

Name	Description	Type	Default	Unit
pgamma1	Cross-term dependence of gamma1	double	0	–
pgamma2	Cross-term dependence of gamma2	double	0	–
phin	Adjusting parameter for surface potential due to nonuniform vertical doping	double	0	V
pigcd	Parameter for Igc partition	double	1	–
pk1	Cross-term dependence of k1	double	0	–
pk2	Cross-term dependence of k2	double	0	–
pk3	Cross-term dependence of k3	double	0	–
pk3b	Cross-term dependence of k3b	double	0	–
pketa	Cross-term dependence of keta	double	0	–
pkt1	Cross-term dependence of kt1	double	0	–
pkt1l	Cross-term dependence of kt1l	double	0	–
pkt2	Cross-term dependence of kt2	double	0	–
plpe0	Cross-term dependence of lpe0	double	0	–
plpeb	Cross-term dependence of lpeb	double	0	–
pminv	Cross-term dependence of minv	double	0	–
pmoin	Cross-term dependence of moin	double	0	–
pndep	Cross-term dependence of ndep	double	0	–
pnfactor	Cross-term dependence of nfactor	double	0	–
pngate	Cross-term dependence of ngate	double	0	–
pnigbacc	Cross-term dependence of nigbacc	double	0	–
pnigbinv	Cross-term dependence of nigbinv	double	0	–
pnigc	Cross-term dependence of nigc	double	0	–
pnoff	Cross-term dependence of noff	double	0	–
pnsd	Cross-term dependence of nsd	double	0	–
pnsub	Cross-term dependence of nsub	double	0	–
pntox	Cross-term dependence of ntox	double	0	–
poxedge	Factor for the gate edge Tox	double	1	–
ppclm	Cross-term dependence of pclm	double	0	–

Table 38 BSIM4 model parameters (Continued)

Name	Description	Type	Default	Unit
ppdiblc1	Cross-term dependence of pdiblc1	double	0	–
ppdiblc2	Cross-term dependence of pdiblc2	double	0	–
ppdiblcb	Cross-term dependence of pdiblcb	double	0	–
ppdits	Cross-term dependence of pdits	double	0	–
ppditsd	Cross-term dependence of pditsd	double	0	–
pphin	Cross-term dependence of phin	double	0	–
ppigcd	Cross-term dependence for pigcd	double	0	–
ppoxedge	Cross-term dependence for poxedge	double	0	–
pprt	Cross-term dependence of prt	double	0	–
pprwb	Cross-term dependence of prwb	double	0	–
pprwg	Cross-term dependence of prwg	double	0	–
ppscbe1	Cross-term dependence of pscbe1	double	0	–
ppscbe2	Cross-term dependence of pscbe2	double	0	–
ppvag	Cross-term dependence of pvag	double	0	–
prdsw	Cross-term dependence of rdsw	double	0	–
prdw	Cross-term dependence of rdw	double	0	–
prsw	Cross-term dependence of rsw	double	0	–
prt	Temperature coefficient of parasitic resistance	double	0	Ω*m
prwb	Body effect on parasitic resistance	double	0	$V^{-0.5}$
prwg	Gate-bias effect on parasitic resistance	double	1	V^{-1}
pscbe1	Substrate current body-effect coefficient	double	4.24e+08	V/m
pscbe2	Substrate current body-effect coefficient	double	1e-05	m/V
pu0	Cross-term dependence of u0	double	0	–
pua	Cross-term dependence of ua	double	0	–
pua1	Cross-term dependence of ua1	double	0	–
pub	Cross-term dependence of ub	double	0	–
pub1	Cross-term dependence of ub1	double	0	–
puc	Cross-term dependence of uc	double	0	–

Table 38 BSIM4 model parameters (Continued)

Name	Description	Type	Default	Unit
puc1	Cross-term dependence of uc1	double	0	–
pute	Cross-term dependence of ute	double	0	–
pvag	Gate dependence of output resistance parameter	double	0	–
pvbm	Cross-term dependence of vbm	double	0	–
pvbx	Cross-term dependence of vbx	double	0	–
pvfb	Cross-term dependence of vfb	double	0	–
pvfbcv	Cross-term dependence of vfbcv	double	0	–
pvoff	Cross-term dependence of voff	double	0	–
pvoffcv	Cross-term dependence of voffcv	double	0	–
pvsat	Cross-term dependence of vsat	double	0	–
pvth0	Cross-term dependence of vto	double	0	–
pvtho	Cross-term dependence of vto	double	0	–
pw0	Cross-term dependence of w0	double	0	–
pwr	Cross-term dependence of wr	double	0	–
pxj	Cross-term dependence of xj	double	0	–
pxrcrg1	Cross-term dependence of xrcrg1	double	0	–
pxrcrg	Cross-term dependence of xrcrg2	double	0	–
pxt	Cross-term dependence of xt	double	0	–
rbdb	Resistance between bNode and dbNode	double	50	Ω
rbodymod	Distributed body R model selector	integer	0	–
rbpb	Resistance between bNodePrime and bNode	double	50	Ω
rbpd	Resistance between bNodePrime and bNode	double	50	Ω
rbps	Resistance between bNodePrime and sbNode	double	50	Ω
rbsb	Resistance between bNode and sbNode	double	50	Ω
rdsmod	Bias-dependent source–drain resistance model selector	integer	0	–
rdsw	Source–drain resistance per width	double	200	$\Omega\mu m^{WR}$
rdswmin	Source–drain resistance per width at high Vg	double	0	$\Omega\mu m^{WR}$

Table 38 BSIM4 model parameters (Continued)

Name	Description	Type	Default	Unit
rdw	Drain resistance per width	double	100	$\Omega\mu m^{WR}$
rdwmin	Drain resistance per width at high Vg	double	0	$\Omega\mu m^{WR}$
rgatemo	Gate R model selector	integer	0	–
rsh	Source–drain sheet resistance	double	0	Ω /sq
rshg	Gate sheet resistance	double	0.1	Ω /sq
rsw	Source resistance per width	double	100	–
rswmin	Source resistance per width at high Vg	double	0	–
tcj	Temperature coefficient of cj	double	0	K^{-1}
tcjsw	Temperature coefficient of cjsw	double	0	K^{-1}
tcjswg	Temperature coefficient of cjswg	double	0	K^{-1}
tnoia	Thermal noise parameter	double	1.5	–
tnoib	Thermal noise parameter	double	3.5	–
tnoimod	Thermal noise model selector	integer	0	–
tnom	Parameter measurement temperature	double	27	°C
toxe	Electrical gate oxide thickness	double	3e-09	m
toxm	Gate oxide thickness at which parameters are extracted	double	3e-09	m
toxp	Physical gate oxide thickness	double	3e-09	m
toxref	Target tox value	double	3e-09	m
tpb	Temperature coefficient of pb	double	0	V/K
tpbsw	Temperature coefficient of pbsw	double	0	V/K
tpbswg	Temperature coefficient of pbswg	double	0	V/K
trnqsmod	Transient NQS model selector	integer	0	–
u0	Low-field mobility at tnom	double	0.067 for NMOS 0.025 for PMOS	$m^2(Vs)^{-1}$
ua	Linear gate dependence of mobility	double	1e-09 MOBMOD=0,1 1.0e-15 MOBMOD=2	m/V
ua1	Temperature coefficient of ua	double	1e-09	m/V

Table 38 BSIM4 model parameters (Continued)

Name	Description	Type	Default	Unit
ub	Quadratic gate dependence of mobility	double	1e-19	m^2/V^{-2}
ub1	Temperature coefficient of ub	double	–1e-18	m^2/V^{-2}
uc	Body-bias dependence of mobility	double	–4.65e-11 MOBMOD=0,2 –0.0465 MOBMOD=1	mV^{-2} V^{-1}
uc1	Temperature coefficient of uc	double	–5.6e-11 V^{-1} for MOBMOD 1 –0.056 for MOBMOD 0,2	–
ute	Temperature coefficient of mobility	double	–1.5	–
vbm	Maximum body voltage	double	–3	V
vbx	Vth transition body voltage	double	0V	–
version	Parameter for model version	string	4.1.0	–
vfb	Flat-band voltage	double	0V	–
vfbcv	Flat-band voltage parameter for capmod=0 only	double	–1	V
voff	Threshold voltage offset	double	–0.08	V
voffcv	C–V lateral shift parameter	double	0	V
voffl	Length dependence parameter for Vth offset	double	0	mV
vsat	Saturation velocity at tnom	double	80000	m/S
vth0	Threshold voltage	double	0.7 NMOS 0.7 PMOS	V
vtho	Threshold voltage	double	0.7	–
w0	Narrow width effect parameter	double	2.5e-06	m
wa0	Width dependence of a0	double	0	–
wa1	Width dependence of a1	double	0	–
wa2	Width dependence of a2	double	0	–
wacde	Width dependence of acde	double	0	–
wagidl	Width dependence of agidl	double	0	–
wags	Width dependence of ags	double	0	–

Table 38 BSIM4 model parameters (Continued)

Name	Description	Type	Default	Unit
waigbacc	Width dependence of aigbacc	double	0	–
waigbinv	Width dependence of aigbinv	double	0	–
waigc	Width dependence of aigc	double	0	–
waigsd	Width dependence of aigsd	double	0	–
walpha0	Width dependence of alpha0	double	0	–
walpha1	Width dependence of alpha1	double	0	–
wat	Width dependence of at	double	0	–
wb0	Width dependence of b0	double	0	–
wb1	Width dependence of b1	double	0	–
wbeta0	Width dependence of beta0	double	0	–
wbgidl	Width dependence of bgidl	double	0	–
wbigbacc	Width dependence of bigbacc	double	0	–
wbigbinv	Width dependence of bigbinv	double	0	–
wbigc	Width dependence of bigc	double	0	–
wbigsd	Width dependence of bigsd	double	0	–
wcdsc	Width dependence of cdsc	double	0	–
wcdscb	Width dependence of cdscb	double	0	–
wcdscd	Width dependence of cdscd	double	0	–
wcf	Width dependence of cf	double	0	–
wcgdl	Width dependence of cgdl	double	0	–
wcgidl	Width dependence of cgidl	double	0	–
wcgsl	Width dependence of cgsl	double	0	–
wcigbacc	Width dependence of cigbacc	double	0	–
wcigbinv	Width dependence of cigbinv	double	0	–
wcigc	Width dependence of cigc	double	0	–
wcigsd	Width dependence of cigsd	double	0	–
wcit	Width dependence of cit	double	0	–
wckappad	Width dependence of ckappad	double	0	–
wckappas	Width dependence of ckappas	double	0	–

Table 38 BSIM4 model parameters (Continued)

Name	Description	Type	Default	Unit
wclc	Width dependence of clc	double	0	–
wcle	Width dependence of cle	double	0	–
wdelta	Width dependence of delta	double	0	–
wdrout	Width dependence of drout	double	0	–
wdsub	Width dependence of dsub	double	0	–
wdvt0	Width dependence of dvt0	double	0	–
wdvt0w	Width dependence of dvt0w	double	0	–
wdvt1	Width dependence of dvt1	double	0	–
wdvt1w	Width dependence of dvt1w	double	0	–
wdvt2	Width dependence of dvt2	double	0	–
wdvt2w	Width dependence of dvt2w	double	0	–
wdvtp0	Width dependence of dvtp0	double	0	–
wdvtp1	Width dependence of dvtp1	double	0	–
wdwb	Width dependence of dwb	double	0	–
wdwg	Width dependence of dwg	double	0	–
wegidl	Width dependence of egidl	double	0	–
weigbinv	Width dependence of eigbinv	double	0	–
weta0	Width dependence of eta0	double	0	–
wetab	Width dependence of etab	double	0	–
weu	Width dependence of eu	double	0	–
wfprout	Width dependence of pdiblcb	double	0	–
wgamma1	Width dependence of gamma1	double	0	–
wgamma2	Width dependence of gamma2	double	0	–
wint	Width reduction parameter	double	0	m
wk1	Width dependence of k1	double	0	–
wk2	Width dependence of k2	double	0	–
wk3	Width dependence of k3	double	0	–
wk3b	Width dependence of k3b	double	0	–
wketa	Width dependence of keta	double	0	–

Table 38 BSIM4 model parameters (Continued)

Name	Description	Type	Default	Unit
wkt1	Width dependence of kt1	double	0	–
wkt1l	Width dependence of kt1l	double	0	–
wkt2	Width dependence of kt2	double	0	–
wl	Width reduction parameter	double	0	m^{wln}
wlc	Width reduction parameter for CV	double	0	–
wln	Width reduction parameter	double	1	–
wlpe0	Width dependence of lpe0	double	0	–
wlpeb	Width dependence of lpeb	double	0	–
wmax	Maximum width of model	double	1	m
wmin	Minimum width of model	double	0	m
wminv	Width dependence of minv	double	0	–
wmoin	Width dependence of moin	double	0	–
wndep	Width dependence of ndep	double	0	–
wnfactor	Width dependence of nfactor	double	0	–
wngate	Width dependence of ngate	double	0	–
wnigbacc	Width dependence of nigbacc	double	0	–
wnigbinv	Width dependence of nigbinv	double	0	–
wnigc	Width dependence of nigc	double	0	–
wnoff	Width dependence of noff	double	0	–
wnsd	Width dependence of nsd	double	0	–
wnsub	Width dependence of nsub	double	0	–
wntox	Width dependence of ntox	double	0	–
wpclm	Width dependence of pclm	double	0	–
wpdiblc1	Width dependence of pdiblc1	double	0	–
wpdiblc2	Width dependence of pdiblc2	double	0	–
wpdiblcb	Width dependence of pdiblcb	double	0	–
wpdits	Width dependence of pdits	double	0	–
wpditsd	Width dependence of pditsd	double	0	–
wphin	Width dependence of phin	double	0	–

Table 38 BSIM4 model parameters (Continued)

Name	Description	Type	Default	Unit
wpigcd	Width dependence of pigcd	double	0	–
wpoxedge	Width dependence of poxedge	double	0	–
wprt	Width dependence of prt	double	0	–
wprwb	Width dependence of prwb	double	0	–
wprwg	Width dependence of prwg	double	0	–
wpscbe1	Width dependence of pscbe1	double	0	–
wpscbe2	Width dependence of pscbe2	double	0	–
wpvag	Width dependence of pvag	double	0	–
wr	Width dependence of rds	double	1	–
wrdsw	Width dependence of rdsw	double	0	–
wrdw	Width dependence of rdw	double	0	–
wrsw	Width dependence of rsw	double	0	–
wu0	Width dependence of u0	double	0	–
wua	Width dependence of ua	double	0	–
wua1	Width dependence of ua1	double	0	–
wub	Width dependence of ub	double	0	–
wub1	Width dependence of ub1	double	0	–
wuc	Width dependence of uc	double	0	–
wuc1	Width dependence of uc1	double	0	–
wute	Width dependence of ute	double	0	–
wvbm	Width dependence of vbm	double	0	–
wvbx	Width dependence of vbx	double	0	–
wvfb	Width dependence of vfb	double	0	–
wvfbcv	Width dependence of vfbcv	double	0	–
wvoff	Width dependence of voff	double	0	–
wvoffcv	Width dependence of voffcv	double	0	–
wvsat	Width dependence of vsat	double	0	–
wvth0	Width dependence of vto	double	0	–
wvtho	Width dependence of vto	double	0	–

Table 38 BSIM4 model parameters (Continued)

Name	Description	Type	Default	Unit
ww	Width reduction parameter	double	0	m^{wwn}
ww0	Width dependence of w0	double	0	–
wwc	Width reduction parameter for CV	double	0	–
wwl	Width reduction parameter	double	0	$m^{wwn + wln}$
wwlc	Width reduction parameter for CV	double	0	–
wwn	Width reduction parameter	double	1	–
wwr	Width dependence of wr	double	0	–
wxj	Width dependence of xj	double	0	–
wxrcrg1	Width dependence of xrcrg1	double	0	–
wxrcrg2	Width dependence of xrcrg2	double	0	–
wxt	Width dependence of xt	double	0	–
xgl	Variation in Ldrawn	double	0	m
xgw	Distance from gate contact center to device edge	double	0	m
xj	Junction depth	double	1.5e-07	m
xjbvd	Fitting parameter for drain diode breakdown current	double	1	–
xjbvs	Fitting parameter for source diode breakdown current	double	1	–
xpart	Channel charge partitioning	double	0	–
xrcrg1	First fitting parameter the bias-dependent Rg	double	12	–
xrcrg2	Second fitting parameter the bias-dependent Rg	double	1	–
xt	Doping depth	double	1.55e-07	m
xtid	Drain junction current temperature exponent	double	3	–
xtis	Source junction current temperature exponent	double	3	–

Table 39 BSIM4 instance parameters

Name	Description	Type	Default	Unit
l	Length	double	5e-06	m
w	Width	double	5e-06	m
nf	Number of fingers	double	1	–
min	Minimize either D or S	integer	0	–
ad	Drain area	double	0	m^2
as	Source area	double	0	m^2
pd	Drain perimeter	double	0	m
ps	Source perimeter	double	0	m
nrd	Number of squares in drain	double	1	–
nrs	Number of squares in source	double	1	–
off	Device is initially off	integer	false	–
rbdb_ins	Body resistance	double	50	Ω
rbsb_ins	Body resistance	double	50	Ω
rbpb_ins	Body resistance	double	50	Ω
rbps_ins	Body resistance	double	50	Ω
rbpd_ins	Body resistance	double	50	Ω
trnqsmod_ins	Transient NQS model selector	integer	0	–
acnqsmod_ins	AC NQS model selector	integer	0	–
rbodymod_ins	Distributed body R model selector	integer	0	–
rgatemod_ins	Gate resistance model selector	integer	0	–
geomod_ins	Geometry-dependent parasitics model selector	integer	0	–
rgeomod	Source–drain resistance and contact model selector	integer	0	–
ic[]	Vector of D–S, G–S, B–S initial voltages	double	–	V

BSIMPD2.2 MOSFET Model

The BSIMPD2.2 model is a partially depleted silicon-on-insulator MOSFET model.

Device name:	`B3SOI`
Default parameter set name:	`B3SOI_pset`
Electrodes:	`Drain` – External drain node `Gate` – Gate node `Source` – External source node `Backgate` – Substrate node `PP` – External body contact node (optional) `Body` – Internal body contact node (optional) `Temp` – Temperature node (optional)
Internal variables:	`drain` (internal drain voltage, only available if `rsh` > 0 and `nrd` > 0) `source` (internal source voltage, only available if `rsh` > 0 and `nrs` > 0)

Table 40 BSIMPD2.2 model parameters

Name	Description	Type	Default value	Unit
Model control parameters				
`binunit`	Bin unit selector	integer	1	–
`capmod`	Capacitance model selector	integer	2	–
`mobmod`	Mobility model selector	integer	1	–
`noimod`	Noise model selector	integer	1	–
`paramchk`	Model parameter checking selector	integer	0	–
`shmod`	Self-heating mode selector: 0 – no self-heating; 1 – self-heating	integer	0	–
`version`	Parameter for model version	double	2.0	–
Process parameters				
`gamma1`	Vth body coefficient	double	0.0	$V^{1/2}$
`gamma2`	Vth body coefficient	double	0.0	$V^{1/2}$
`nch`	Channel doping concentration	double	1.7e17	cm^{-3}
`ngate`	Poly-silicon gate doping concentration	double	0.0	cm^{-3}
`nsub`	Substrate doping concentration	double	6.0e16	cm^{-3}

Table 40 BSIMPD2.2 model parameters (Continued)

Name	Description	Type	Default value	Unit
tbox	Buried oxide thickness	double	3.0e-7	m
tox	Gate oxide thickness	double	1.0e-8	m
tsi	Silicon film thickness	double	1.0e-7	m
vbm	Maximum body voltage	double	0.0	V
vbx	Vth transition body voltage	double	0.0	V
xj	Source–drain junction depth	double	tsi	m
xt	Doping depth	double	1.55e-7	m
DC parameters				
vth0	Threshold voltage at V_{bs}=0 for long and wide device	double	0.7 (NMOS) –0.7 (PMOS)	V
k1	First-order body-effect coefficient	double	0.6	$V^{1/2}$
k1w1	First body-effect width-dependent parameter	double	0.0	m
k1w2	Second body-effect width-dependent parameter	double	0.0	m
k2	Second-order body-effect coefficient	double	0.0	–
k3	Narrow width coefficient	double	80.0	–
k3b	Body effect coefficient of k3	double	0.0	V^{-1}
w0	Narrow width parameter	double	2.5e-6	m
kb1	Backgate body charge coefficient	double	1	–
nlx	Lateral nonuniform doping parameter	double	1.74e-7	m
dvt0	First coefficient of short-channel effect on V_{th}	double	2.2	–
dvt1	Second coefficient of short-channel effect on V_{th}	double	0.53	–
dvt2	Body-bias coefficient of short-channel effect on V_{th}	double	–0.032	V^{-1}
dvt0w	First coefficient of narrow width effect on V_{th} for small channel length	double	0.0	–
dvt1w	Second coefficient of narrow width effect on V_{th} for small channel length	double	5.3e6	m^{-1}
dvt2w	Body-bias coefficient of narrow width effect for small channel length	double	–0.032	V^{-1}

Table 40 BSIMPD2.2 model parameters (Continued)

Name	Description	Type	Default value	Unit
u0	Low-field mobility at Temp=Tnom	double	0.067 (NMOS) 0.025 (PMOS)	$m^2/(Vs)$
ua	Coefficient of first-order mobility degradation due to vertical field	double	2.25e-9	m/V
ub	Coefficient of second-order mobility degradation due to vertical field	double	5.87e-19	m^2/V^2
uc	Coefficient of mobility degradation due to body-bias effect	double	–0.0465 mobmod=3 –4.65e-11 mobmod=1,2	V^{-1} m/V^2
vsat	Saturation velocity at Temp=Tnom	double	8.0e4	m/s
a0	Coefficient of channel-length dependence of bulk charge effect	double	1.0	–
ags	Coefficient of V_{gs} dependence of bulk charge effect	double	0.0	V^{-1}
b0	Bulk charge effect coefficient for channel width	double	0.0	m
b1	Bulk charge effect width offset	double	0.0	m
keta	Body-bias coefficient of bulk charge effect	double	–0.6	V^{-1}
ketas	Surface potential adjustment for bulk charge effect	double	0.0	V
a1	First nonsaturation effect parameter	double	0.0	V^{-1}
a2	Second nonsaturation factor	double	1.0	–
rdsw	Parasitic resistance per unit width	double	100.0	$\Omega(\mu m)^{wr}$
prwb	Body-bias dependence of rdsw	double	0.0	$V^{-1/2}$
prwg	Gate-bias dependence of rdsw	double	0.0	V^{-1}
wr	Channel-width dependence parameter of rdsw	double	1.0	–
nfactor	Subthreshold swing factor	double	1.0	–
wint	Width offset fitting parameter from I-V without bias	double	0.0	m
lint	Length offset fitting parameter from I-V without bias	double	0.0	m
dwg	Coefficient of gate bias dependence of W_{eff}	double	0.0	m/V

Table 40 BSIMPD2.2 model parameters (Continued)

Name	Description	Type	Default value	Unit
`dwb`	Coefficient of body bias dependence of W_{eff}	double	0.0	$m/V^{1/2}$
`dwbc`	Width offset for body contact isolation edge	double	0.0	m
`voff`	Offset voltage in subthreshold region for large W and L	double	–0.08	V
`eta0`	DIBL coefficient in subthreshold region	double	0.08	–
`etab`	Body-bias coefficient for the subthreshold DIBL effect	double	–0.07	V^{-1}
`dsub`	DIBL coefficient exponent in subthreshold region	double	0.56	–
`cit`	Interface trap capacitance	double	0.0	F/m^2
`cdsc`	Coupling capacitance between source–drain and channel	double	2.4e-4	F/m^2
`cdscb`	Body-bias dependence of `cdsc`	double	0.0	$F/(Vm^2)$
`cdscd`	Drain-bias dependence of `cdsc`	double	0.0	$F/(Vm^2)$
`pclm`	Channel length modulation parameter	double	1.3	–
`pdiblc1`	Parameter for DIBL effect on Rout	double	0.39	–
`pdiblc2`	Parameter for DIBL effect on Rout	double	0.0086	–
`pdiblcb`	Body-bias coefficient of DIBL effect on Rout	double	0.0	V^{-1}
`drout`	Channel-length dependence of DIBL effect on Rout	double	0.56	–
`pvag`	Gate-bias dependence of Early voltage	double	0.0	–
`delta`	Parameter for DC V_{dseff}	double	0.01	V
`alpha0`	First parameter of impact ionization current	double	0.0	m/V
`fbjtii`	Fraction of bipolar current affecting the impact ionization	double	0.0	–
`beta0`	First V_{ds}-dependent parameter of impact ionization current	double	0.0	V^{-1}
`beta1`	Second V_{ds}-dependent parameter of impact ionization current	double	0.0	–
`beta2`	Third V_{ds}-dependent parameter of impact ionization current	double	0.1	V
`vdsatii0`	Nominal drain saturation voltage at threshold for impact ionization current	double	0.9	V

Table 40 BSIMPD2.2 model parameters (Continued)

Name	Description	Type	Default value	Unit
tii	Temperature-dependent parameter for impact ionization current	double	0.0	–
lii	Channel length–dependent parameter at threshold for impact ionization current	double	0.0	m
esatii	Saturation electric field for impact ionization	double	1.0e7	V/m
sii0	First V_{gs}-dependent parameter for impact ionization current	double	0.5	V^{-1}
sii1	Second V_{gs}-dependent parameter for impact ionization current	double	0.1	V^{-1}
sii2	Third V_{gs}-dependent parameter for impact ionization current	double	0.0	V^{-1}
siid	V_{ds}-dependent parameter of drain saturation voltage for impact ionization current	double	0.0	V^{-1}
agidl	GIDL constant	double	0.0	Ω^{-1}
bgidl	GIDL exponential coefficient	double	0.0	V/m
ngidl	GIDL Vds enhancement coefficient	double	1.2	V
ntun	Reverse tunneling nonideality factor	double	10.0	–
ndiode	Diode nonideality factor	double	1.0	–
nrecf0	Recombination nonideality factor at forward bias	double	2.0	–
nrecr0	Recombination nonideality factor at reverse bias	double	10.0	–
isbjt	BJT injection saturation current	double	1.0e-6	A/m^2
isdif	Body to source–drain injection saturation current	double	0.0	A/m^2
isrec	Recombination in depletion saturation current	double	1.0e-5	A/m^2
istun	Reverse tunneling saturation current	double	0.0	A/m^2
ln	Electron–hole diffusion length	double	2.0e-6	m
vrec0	Voltage-dependent parameter for recombination current	double	0.0	V
vtun0	Voltage-dependent parameter for tunneling current	double	0.0	V
nbjt	Power coefficient of channel length–dependency for bipolar current	double	1.0	–
lbjt0	Reference channel length for bipolar current	double	2.0e-7	m

Table 40 BSIMPD2.2 model parameters (Continued)

Name	Description	Type	Default value	Unit
vabjt	Early voltage for bipolar current	double	10.0	V
aely	Channel length–dependency of Early voltage for bipolar current	double	0.0	V/m
ahli	High-level injection parameter for bipolar current	double	0.0	–
rbody	Intrinsic body contact sheet resistance	double	0.0	Ω /sq
rbsh	Extrinsic body contact sheet resistance	double	0.0	Ω /sq
rsh	Source–drain sheet resistance	double	0.0	Ω /sq
rhalo	Body halo sheet resistance	double	1.0e15	Ω /m
Gate-to-body tunneling parameters				
igmod	Gate current model selector	double	0	–
toxqm	Oxide thickness for I_{gb} calculation	double	tox	m
ntox	Power term of gate current	double	1.0	–
toxref	Target oxide thickness	double	2.5e-9	m
ebg	Effective band gap in gate current calculation	double	1.2	V
alphagb1	First V_{ox}-dependent parameter for gate current in inversion	double	0.35	V^{-1}
betagb1	Second V_{ox}-dependent parameter for gate current in inversion	double	0.03	V^{-2}
vgb1	Third V_{ox}-dependent parameter for gate current in inversion	double	300.0	V
vevb	Vaux parameter for valence band electron tunneling	double	0.075	–
alphagb2	First V_{ox}-dependent parameter for gate current in accumulation	double	0.43	V^{-1}
betagb2	Second V_{ox}-dependent parameter for gate current in accumulation	double	0.05	V^{-2}
vgb2	Third V_{ox}-dependent parameter for gate current in accumulation	double	17.0	V
vecb	Vaux parameter for conduction band electron tunneling	double	0.026	–
voxh	Limit of V_{ox} in gate current calculation	double	5.0	–
deltavox	Smoothing parameter in V_{ox} smoothing function	double	0.005	–

Table 40 BSIMPD2.2 model parameters (Continued)

Name	Description	Type	Default value	Unit
AC and capacitance parameters				
xpart	Charge partitioning rate flag	double	0.0	–
cgso	Non-LDD region source–gate overlap capacitance per channel length	double	0.0	F/m
cgdo	Non-LDD region drain–gate overlap capacitance per channel length	double	0.0	F/m
cgeo	Gate substrate overlap capacitance per unit channel length	double	0.0	F/m
cjswg	Source–drain (gate side) sidewall junction capacitance per unit width (normalized to 100 nm T_{si})	double	1.0e-10	F/m^2
pbswg	Source–drain (gate side) sidewall junction capacitance build-in potential	double	0.7	V
mjswg	Source–drain (gate side) sidewall junction capacitance grading coefficient	double	0.5	–
tt	Diffusion capacitance transit time coefficient	double	1.0e-12	s
ndif	Power coefficient of channel length–dependency for diffusion capacitance	double	1.0	–
ldif0	Channel length–dependency coefficient of diffusion capacitance	double	1.0	–
vsdfb	Source–drain bottom diffusion capacitance flat-band voltage	double	0.0	V
vsdth	Source–drain bottom diffusion capacitance threshold voltage	double	0.0	V
csdmin	Source–drain bottom diffusion minimum capacitance	double	0.0	F
asd	Source–drain bottom diffusion smoothing parameter	double	0.3	–
csdesw	Source–drain sidewall fringing capacitance per unit length	double	0.0	F/m
cgsl	Light-doped source–gate region overlap capacitance	double	0.0	F/m
cgdl	Light-doped drain–gate region overlap capacitance	double	0.0	F/m
ckappa	Coefficient of bias-dependent for light-doped region overlap capacitance	double	0.6	–

Table 40 BSIMPD2.2 model parameters (Continued)

Name	Description	Type	Default value	Unit
cf	Gate to source–drain fringing field capacitance	double	0.0	F/m
clc	Constant term for short-channel model	double	1.0e-8	m
cle	Exponential term for short-channel model	double	0.0	–
dlc	Length offset fitting parameter for gate charge	double	0.0	m
dlcb	Length offset fitting parameter for body charge	double	0.0	m
dlbg	Length offset fitting parameter for backgate charge	double	0.0	m
dwc	Width offset fitting parameter for C–V	double	0.0	m
delvt	Threshold voltage adjust for C–V	double	0.0	V
fbody	Scaling factor for body charge	double	1.0	–
acde	Exponential coefficient for charge thickness in capmod=3 for accumulation and depletion regions	double	1.0	m/V
moin	Coefficient for gate bias–dependent surface potential	double	15.0	$V^{0.5}$
Temperature parameters				
tnom	Temperature at which parameters are extracted	double	27	°C
ute	Temperature coefficient of mobility	double	–1.5	–
kt1	Temperature coefficient for V_{th}	double	–0.11	V
kt1l	Channel-length dependence of temperature coefficient for V_{th}	double	0.0	Vm
kt2	Body-bias coefficient of V_{th} temperature effect	double	0.022	–
ua1	Temperature coefficient of ua	double	4.31e-9	m/V
ub1	Temperature coefficient of ub	double	–7.61e-18	$(m/V)^2$
uc1	Temperature coefficient of uc	double	–0.056 mobmod=3 –0.056e-9 mobmod=1,2	V^{-1} m/V^2
at	Temperature coefficient of vsat	double	3.3e4	m/s
tcjswg	Temperature coefficient of cjswg	double	0.0	1/°C
tpbswg	Temperature coefficient of pbswg	double	0.0	V/°C
prt	Temperature coefficient of rdsw	double	0.0	Ω -m

Table 40 BSIMPD2.2 model parameters (Continued)

Name	Description	Type	Default value	Unit
cth0	Normalized thermal capacity	double	0.0	Ws/m°C
rth0	Normalized thermal resistance	double	0.0	m°C /W
ntrecf	Temperature coefficient for nrecf	double	0.0	–
ntrecr	Temperature coefficient for nrecr	double	0.0	–
xbjt	Power dependence of j_{bjt} on temperature	double	1.0	–
xdif	Power dependence of j_{dif} on temperature	double	1.0	–
xrec	Power dependence of j_{rec} on temperature	double	1.0	–
xtun	Power dependence of j_{tun} on temperature	double	0.0	–
wth0	Minimum width for thermal resistance calculation	double	0.0	m
dW and dL parameters				
ll	Coefficient of length dependence for length offset	double	0.0	m^{lln}
lln	Power of length dependence for length offset	double	1.0	–
lw	Coefficient of width dependence for length offset	double	0.0	m^{lwn}
lwn	Power of width dependence for length offset	double	1.0	–
lwl	Coefficient of length and width cross-term dependence for length offset	double	0.0	$m^{lwn+lln}$
wl	Coefficient of length dependence for width offset	double	0.0	m^{wln}
wln	Power of length dependence of width offset	double	1.0	–
ww	Coefficient of width dependence for width offset	double	0.0	m^{wwn}
wwn	Power of width dependence of width offset	double	1.0	–
wwl	Coefficient of length and width cross-term dependence for width offset	double	0.0	$m^{wwn+wln}$
Noise parameters				
af	Flicker noise exponent	double	1.0	–
ef	Flicker noise frequency exponent	double	1.0	–
em	Flicker noise parameter	double	4.1e7	V/m
kf	Flicker noise coefficient	double	0.0	–
noia	Flicker noise parameter	double	1e20 NMOS 9.9e18 PMOS	–

Table 40 BSIMPD2.2 model parameters (Continued)

Name	Description	Type	Default value	Unit
noib	Flicker noise parameter	double	5e4 NMOS 2.4e3 PMOS	–
noic	Flicker noise parameter	double	–1.4e-12 NMOS 1.4e-12 PMOS	–
noif	Floating body excess noise ideality factor	double	1.0	–
Length dependence				
lnch	Length dependence of nch	double	0.0	–
lnsub	Length dependence of nsub	double	0.0	–
lngate	Length dependence of ngate	double	0.0	–
lvth0	Length dependence of vto	double	0.0	–
lk1	Length dependence of k1	double	0.0	–
lk1w1	Length dependence of k1w1	double	0.0	–
lk1w2	Length dependence of k1w2	double	0.0	–
lk2	Length dependence of k2	double	0.0	–
lk3	Length dependence of k3	double	0.0	–
lk3b	Length dependence of k3b	double	0.0	–
lkb1	Length dependence of kb1	double	0.0	–
lw0	Length dependence of w0	double	0.0	–
lnlx	Length dependence of nlx	double	0.0	–
ldvt0	Length dependence of dvt0	double	0.0	–
ldvt1	Length dependence of dvt1	double	0.0	–
ldvt2	Length dependence of dvt2	double	0.0	–
ldvt0w	Length dependence of dvt0w	double	0.0	–
ldvt1w	Length dependence of dvt1w	double	0.0	–
ldvt2w	Length dependence of dvt2w	double	0.0	–
lu0	Length dependence of u0	double	0.0	–
lua	Length dependence of ua	double	0.0	–
lub	Length dependence of ub	double	0.0	–

Table 40 BSIMPD2.2 model parameters (Continued)

Name	Description	Type	Default value	Unit
luc	Length dependence of uc	double	0.0	–
lvsat	Length dependence of vsat	double	0.0	–
la0	Length dependence of a0	double	0.0	–
lags	Length dependence of ags	double	0.0	–
lb0	Length dependence of b0	double	0.0	–
lb1	Length dependence of b1	double	0.0	–
lketa	Length dependence of keta	double	0.0	–
lketas	Length dependence of ketas	double	0.0	–
la1	Length dependence of a1	double	0.0	–
la2	Length dependence of a2	double	0.0	–
lrdsw	Length dependence of rdsw	double	0.0	–
lprwb	Length dependence of prwb	double	0.0	–
lprwg	Length dependence of prwg	double	0.0	–
lwr	Length dependence of wr	double	0.0	–
lnfactor	Length dependence of nfactor	double	0.0	–
ldwg	Length dependence of dwg	double	0.0	–
ldwb	Length dependence of dwb	double	0.0	–
lvoff	Length dependence of voff	double	0.0	–
leta0	Length dependence of eta0	double	0.0	–
letab	Length dependence of etab	double	0.0	–
ldsub	Length dependence of dsub	double	0.0	–
lcit	Length dependence of cit	double	0.0	–
lcdsc	Length dependence of cdsc	double	0.0	–
lcdscb	Length dependence of cdscb	double	0.0	–
lcdscd	Length dependence of cdscd	double	0.0	–
lpclm	Length dependence of pclm	double	0.0	–
lpdiblc1	Length dependence of pdiblc1	double	0.0	–
lpdiblc2	Length dependence of pdiblc2	double	0.0	–
lpdiblcb	Length dependence of pdiblcb	double	0.0	–

Table 40 BSIMPD2.2 model parameters (Continued)

Name	Description	Type	Default value	Unit
ldrout	Length dependence of drout	double	0.0	–
lpvag	Length dependence of pvag	double	0.0	–
ldelta	Length dependence of delta	double	0.0	–
lalpha0	Length dependence of alpha0	double	0.0	–
lfbjtii	Length dependence of fbjtii	double	0.0	–
lbeta0	Length dependence of beta0	double	0.0	–
lbeta1	Length dependence of beta1	double	0.0	–
lbeta2	Length dependence of beta2	double	0.0	–
lvdsatii0	Length dependence of vdsatii0	double	0.0	–
llii	Length dependence of lii	double	0.0	–
lesatii	Length dependence of esatii	double	0.0	–
lsii0	Length dependence of sii0	double	0.0	–
lsii1	Length dependence of sii1	double	0.0	–
lsii2	Length dependence of sii2	double	0.0	–
lsiid	Length dependence of siid	double	0.0	–
lagidl	Length dependence of agidl	double	0.0	–
lbgidl	Length dependence of bgidl	double	0.0	–
lngidl	Length dependence of ngidl	double	0.0	–
lntun	Length dependence of ntun	double	0.0	–
lndiode	Length dependence of ndiode	double	0.0	–
lnrecf0	Length dependence of nrecf0	double	0.0	–
lnrecr0	Length dependence of nrecr0	double	0.0	–
lisbjt	Length dependence of isbjt	double	0.0	–
lisdif	Length dependence of isdif	double	0.0	–
lisrec	Length dependence of isrec	double	0.0	–
listun	Length dependence of istun	double	0.0	–
lvrec0	Length dependence of vrec0	double	0.0	–
lvtun0	Length dependence of vtun0	double	0.0	–
lnbjt	Length dependence of nbjt	double	0.0	–

Table 40 BSIMPD2.2 model parameters (Continued)

Name	Description	Type	Default value	Unit
llbjt0	Length dependence of lbjt0	double	0.0	–
lvabjt	Length dependence of vabjt	double	0.0	–
laely	Length dependence of aely	double	0.0	–
lahli	Length dependence of ahli	double	0.0	–
lvsdfb	Length dependence of vsdfb	double	0.0	–
lvsdth	Length dependence of vsdth	double	0.0	–
ldelvt	Length dependence of delvt	double	0.0	–
lacde	Length dependence of acde	double	0.0	–
lmoin	Length dependence of amoin	double	0.0	–
Width dependence				
wnch	Width dependence of nch	double	0.0	–
wnsub	Width dependence of nsub	double	0.0	–
wngate	Width dependence of ngate	double	0.0	–
wvth0	Width dependence of vto	double	0.0	–
wk1	Width dependence of k1	double	0.0	–
wk1w1	Width dependence of k1w1	double	0.0	–
wk1w2	Width dependence of k1w2	double	0.0	–
wk2	Width dependence of k2	double	0.0	–
wk3	Width dependence of k3	double	0.0	–
wk3b	Width dependence of k3b	double	0.0	–
wkb1	Width dependence of kb1	double	0.0	–
ww0	Width dependence of w0	double	0.0	–
wnlx	Width dependence of nlx	double	0.0	–
wdvt0	Width dependence of dvt0	double	0.0	–
wdvt1	Width dependence of dvt1	double	0.0	–
wdvt2	Width dependence of dvt2	double	0.0	–
wdvt0w	Width dependence of dvt0w	double	0.0	–
wdvt1w	Width dependence of dvt1w	double	0.0	–
wdvt2w	Width dependence of dvt2w	double	0.0	–

Table 40 BSIMPD2.2 model parameters (Continued)

Name	Description	Type	Default value	Unit
wu0	Width dependence of u0	double	0.0	–
wua	Width dependence of ua	double	0.0	–
wub	Width dependence of ub	double	0.0	–
wuc	Width dependence of uc	double	0.0	–
wvsat	Width dependence of vsat	double	0.0	–
wa0	Width dependence of a0	double	0.0	–
wags	Width dependence of ags	double	0.0	–
wb0	Width dependence of b0	double	0.0	–
wb1	Width dependence of b1	double	0.0	–
wketa	Width dependence of keta	double	0.0	–
wketas	Width dependence of ketas	double	0.0	–
wa1	Width dependence of a1	double	0.0	–
wa2	Width dependence of a2	double	0.0	–
wrdsw	Width dependence of rdsw	double	0.0	–
wprwb	Width dependence of prwb	double	0.0	–
wprwg	Width dependence of prwg	double	0.0	–
wwr	Width dependence of wr	double	0.0	–
wnfactor	Width dependence of nfactor	double	0.0	–
wdwg	Width dependence of dwg	double	0.0	–
wdwb	Width dependence of dwb	double	0.0	–
wvoff	Width dependence of voff	double	0.0	–
weta0	Width dependence of eta0	double	0.0	–
wetab	Width dependence of etab	double	0.0	–
wdsub	Width dependence of dsub	double	0.0	–
wcit	Width dependence of cit	double	0.0	–
wcdsc	Width dependence of cdsc	double	0.0	–
wcdscb	Width dependence of cdscb	double	0.0	–
wcdscd	Width dependence of cdscd	double	0.0	–
wpclm	Width dependence of pclm	double	0.0	–

Table 40 BSIMPD2.2 model parameters (Continued)

Name	Description	Type	Default value	Unit
wpdiblc1	Width dependence of pdiblc1	double	0.0	–
wpdiblc2	Width dependence of pdiblc2	double	0.0	–
wpdiblcb	Width dependence of pdiblcb	double	0.0	–
wdrout	Width dependence of drout	double	0.0	–
wpvag	Width dependence of pvag	double	0.0	–
wdelta	Width dependence of delta	double	0.0	–
walpha0	Width dependence of alpha0	double	0.0	–
wfbjtii	Width dependence of fbjtii	double	0.0	–
wbeta0	Width dependence of beta0	double	0.0	–
wbeta1	Width dependence of beta1	double	0.0	–
wbeta2	Width dependence of beta2	double	0.0	–
wvdsatii0	Width dependence of vdsatii0	double	0.0	–
wlii	Width dependence of lii	double	0.0	–
wesatii	Width dependence of esatii	double	0.0	–
wsii0	Width dependence of sii0	double	0.0	–
wsii1	Width dependence of sii1	double	0.0	–
wsii2	Width dependence of sii2	double	0.0	–
wsiid	Width dependence of siid	double	0.0	–
wagidl	Width dependence of agidl	double	0.0	–
wbgidl	Width dependence of bgidl	double	0.0	–
wngidl	Width dependence of ngidl	double	0.0	–
wntun	Width dependence of ntun	double	0.0	–
wndiode	Width dependence of ndiode	double	0.0	–
wnrecf0	Width dependence of nrecf0	double	0.0	–
wnrecr0	Width dependence of nrecr0	double	0.0	–
wisbjt	Width dependence of isbjt	double	0.0	–
wisdif	Width dependence of isdif	double	0.0	–
wisrec	Width dependence of isrec	double	0.0	–
wistun	Width dependence of istun	double	0.0	–

Table 40 BSIMPD2.2 model parameters (Continued)

Name	Description	Type	Default value	Unit
wvrec0	Width dependence of vrec0	double	0.0	–
wvtun0	Width dependence of vtun0	double	0.0	–
wnbjt	Width dependence of nbjt	double	0.0	–
wlbjt0	Width dependence of lbjt0	double	0.0	–
wvabjt	Width dependence of vabjt	double	0.0	–
waely	Width dependence of aely	double	0.0	–
wahli	Width dependence of ahli	double	0.0	–
wvsdfb	Width dependence of vsdfb	double	0.0	–
wvsdth	Width dependence of vsdth	double	0.0	–
wdelvt	Width dependence of delvt	double	0.0	–
wacde	Width dependence of acde	double	0.0	–
wmoin	Width dependence of amoin	double	0.0	–
Cross-term dependence				
pnch	Cross-term dependence of nch	double	0.0	–
pnsub	Cross-term dependence of nsub	double	0.0	–
pngate	Cross-term dependence of ngate	double	0.0	–
pvth0	Cross-term dependence of vto	double	0.0	–
pk1	Cross-term dependence of k1	double	0.0	–
pk1w1	Cross-term dependence of k1w1	double	0.0	–
pk1w2	Cross-term dependence of k1w2	double	0.0	–
pk2	Cross-term dependence of k2	double	0.0	–
pk3	Cross-term dependence of k3	double	0.0	–
pk3b	Cross-term dependence of k3b	double	0.0	–
pkb1	Cross-term dependence of kb1	double	0.0	–
pw0	Cross-term dependence of w0	double	0.0	–
pnlx	Cross-term dependence of nlx	double	0.0	–
pdvt0	Cross-term dependence of dvt0	double	0.0	–
pdvt1	Cross-term dependence of dvt1	double	0.0	–
pdvt2	Cross-term dependence of dvt2	double	0.0	–

Table 40 BSIMPD2.2 model parameters (Continued)

Name	Description	Type	Default value	Unit
pdvt0w	Cross-term dependence of dvt0w	double	0.0	–
pdvt1w	Cross-term dependence of dvt1w	double	0.0	–
pdvt2w	Cross-term dependence of dvt2w	double	0.0	–
pu0	Cross-term dependence of u0	double	0.0	–
pua	Cross-term dependence of ua	double	0.0	–
pub	Cross-term dependence of ub	double	0.0	–
puc	Cross-term dependence of uc	double	0.0	–
pvsat	Cross-term dependence of vsat	double	0.0	–
pa0	Cross-term dependence of a0	double	0.0	–
pags	Cross-term dependence of ags	double	0.0	–
pb0	Cross-term dependence of b0	double	0.0	–
pb1	Cross-term dependence of b1	double	0.0	–
pketa	Cross-term dependence of keta	double	0.0	–
pketas	Cross-term dependence of ketas	double	0.0	–
pa1	Cross-term dependence of a1	double	0.0	–
pa2	Cross-term dependence of a2	double	0.0	–
prdsw	Cross-term dependence of rdsw	double	0.0	–
pprwb	Cross-term dependence of prwb	double	0.0	–
pprwg	Cross-term dependence of prwg	double	0.0	–
pwr	Cross-term dependence of wr	double	0.0	–
pnfactor	Cross-term dependence of nfactor	double	0.0	–
pdwg	Cross-term dependence of dwg	double	0.0	–
pdwb	Cross-term dependence of dwb	double	0.0	–
pvoff	Cross-term dependence of voff	double	0.0	–
peta0	Cross-term dependence of eta0	double	0.0	–
petab	Cross-term dependence of etab	double	0.0	–
pdsub	Cross-term dependence of dsub	double	0.0	–
pcit	Cross-term dependence of cit	double	0.0	–
pcdsc	Cross-term dependence of cdsc	double	0.0	–

Table 40 BSIMPD2.2 model parameters (Continued)

Name	Description	Type	Default value	Unit
pcdscb	Cross-term dependence of cdscb	double	0.0	–
pcdscd	Cross-term dependence of cdscd	double	0.0	–
ppclm	Cross-term dependence of pclm	double	0.0	–
ppdiblc1	Cross-term dependence of pdiblc1	double	0.0	–
ppdiblc2	Cross-term dependence of pdiblc2	double	0.0	–
ppdiblcb	Cross-term dependence of pdiblcb	double	0.0	–
pdrout	Cross-term dependence of drout	double	0.0	–
ppvag	Cross-term dependence of pvag	double	0.0	–
pdelta	Cross-term dependence of delta	double	0.0	–
palpha0	Cross-term dependence of alpha0	double	0.0	–
pfbjtii	Cross-term dependence of fbjtii	double	0.0	–
pbeta0	Cross-term dependence of beta0	double	0.0	–
pbeta1	Cross-term dependence of beta1	double	0.0	–
pbeta2	Cross-term dependence of beta2	double	0.0	–
pvdsatii0	Cross-term dependence of vdsatii0	double	0.0	–
plii	Cross-term dependence of lii	double	0.0	–
pesatii	Cross-term dependence of esatii	double	0.0	–
psii0	Cross-term dependence of sii0	double	0.0	–
psii1	Cross-term dependence of sii1	double	0.0	–
psii2	Cross-term dependence of sii2	double	0.0	–
psiid	Cross-term dependence of siid	double	0.0	–
pagidl	Cross-term dependence of agidl	double	0.0	–
pbgidl	Cross-term dependence of bgidl	double	0.0	–
pngidl	Cross-term dependence of ngidl	double	0.0	–
pntun	Cross-term dependence of ntun	double	0.0	–
pndiode	Cross-term dependence of ndiode	double	0.0	–
pnrecf0	Cross-term dependence of nrecf0	double	0.0	–
pnrecr0	Cross-term dependence of nrecr0	double	0.0	–
pisbjt	Cross-term dependence of isbjt	double	0.0	–

Table 40 BSIMPD2.2 model parameters (Continued)

Name	Description	Type	Default value	Unit
pisdif	Cross-term dependence of isdif	double	0.0	–
pisrec	Cross-term dependence of isrec	double	0.0	–
pistun	Cross-term dependence of istun	double	0.0	–
pvrec0	Cross-term dependence of vrec0	double	0.0	–
pvtun0	Cross-term dependence of vtun0	double	0.0	–
pnbjt	Cross-term dependence of nbjt	double	0.0	–
plbjt0	Cross-term dependence of lbjt0	double	0.0	–
pvabjt	Cross-term dependence of vabjt	double	0.0	–
paely	Cross-term dependence of aely	double	0.0	–
pahli	Cross-term dependence of ahli	double	0.0	–
pvsdfb	Cross-term dependence of vsdfb	double	0.0	–
pvsdth	Cross-term dependence of vsdth	double	0.0	–
pdelvt	Cross-term dependence of delvt	double	0.0	–
pacde	Cross-term dependence of acde	double	0.0	–
pmoin	Cross-term dependence of amoin	double	0.0	–

Table 41 BSIMPD2.2 instance parameters

Name	Description	Type	Default	Unit
l	Length	double	5e-06	m
w	Width	double	5e-06	m
ad	Drain area	double	0	m^2
as	Source area	double	0	m^2
pd	Drain perimeter	double	0	m
ps	Source perimeter	double	0	m
nrd	Number of squares in drain	double	1	–
nrs	Number of squares in source	double	1	–
off	Device is initially off	integer	0	–
bjtoff	Switch off BJT current if equal to 1	integer	0	–

Table 41 BSIMPD2.2 instance parameters (Continued)

Name	Description	Type	Default	Unit
rth0_ins	Thermal resistance per unit width if not specified, `rth0_ins` is extracted from model card. If specified, it overrides the one in model card.	double	0	–
cth0_ins	Thermal capacitance per unit width if not specified, `cth0_ins` is extracted from model card. If specified, it overrides the one in model card.	double	0	–
nrb	Number of squares in body	double	0	–
frbody	Layout-dependent body resistance coefficient	double	1.0	–
nbc	Number of body contact isolation edge	double	0	–
nseg	Number of segments for width partitioning	double	1	–
pdbcp	Perimeter length for bc parasitics at drain side	double	0	m
psbcp	Perimeter length for bc parasitics at source side	double	0	m
agbcp	Parasitic gate-to-body overlap area for body contact	double	0	m^2
aebcp	Parasitic body-to-substrate overlap area for body contact	double	0	m^2
vbsusr	Optional initial value of V_{bs} specified by user for transient analysis	double	0	V
tnodeout	Temperature node flag indicating the use of T node	integer	0	–

Non-MOSFET Transistors and Diodes

The non-MOSFET transistor and diode models discussed in this section include:

- Diode
- Bipolar Junction Transistor
- Junction Field Effect Transistor
- GaAs MESFET

Diode

The diode model can be used for either junction diodes or Schottky barrier diodes.

The DC characteristics of the diode are determined by the parameters `is` and `n`. An Ohmic resistance, `rs`, is included. Charge storage effects are modeled by a transit time, `tt`, and a nonlinear depletion layer capacitance, which is determined by the parameters `cjo`, `vj`, and `m`. The temperature dependency of the saturation current is defined by the parameters `eg` (the energy) and `xti` (the saturation current temperature exponent). The nominal temperature at which these parameters were measured is `tnom`. Reverse breakdown is modeled by an exponential increase in the reverse diode current and is determined by the parameters `bv` and `ibv` (both are positive numbers).

Device name:	`Diode`
Default parameter set name:	`Diode_pset`
Electrodes:	`D+, D-`
Internal variables:	`internal` (internal anode voltage, only available if $rs \neq 0$)

Table 42 Diode model parameters

Name	Description	Type	Default	Unit
`af`	Flicker noise exponent	double	1	–
`bv`	Reverse breakdown voltage	double	∞	V
`cj0`	(redundant parameter)	double	0	–
`cjo`	Junction capacitance	double	0	F
`eg`	Activation energy	double	1.11	eV
`fc`	Forward bias junction fit parameter	double	0.5	–
`ibv`	Current at reverse breakdown voltage	double	0.001	A
`is`	Saturation current	double	1e-14	A
`kf`	Flicker noise coefficient	double	0	–
`m`	Grading coefficient	double	0.5	–
`n`	Emission coefficient	double	1	–
`rs`	Ohmic resistance	double	0	Ω
`tnom`	Parameter measurement temperature	double	27	°C
`tt`	Transit time	double	0	s

Table 42 Diode model parameters (Continued)

Name	Description	Type	Default	Unit
`vj`	Junction potential	double	1	V
`xti`	Saturation current temperature exponential	double	3	–

Table 43 Diode instance parameters

Name	Description	Type	Default	Unit
`area`	Area factor	double	1	–
`ic`	Initial device voltage	double	0	V
`off`	Initially off	integer	0	–
`temp`	Instance temperature	double	27	°C

Bipolar Junction Transistor

The bipolar junction transistor (BJT) model is based on the integral charge model of Gummel and Poon. However, if the Gummel–Poon parameters are not specified, the model reduces to the simpler Ebers–Moll model. In either case, charge storage effects, Ohmic resistances, and a current-dependent output conductance can be included.

The bipolar junction transistor (BJT) model is an adaptation of the integral charge control model of Gummel and Poon. This modified Gummel–Poon model extends the original model to include several effects at high bias levels. The model automatically simplifies to the Ebers–Moll model when certain parameters are not specified. The parameter names used in the modified Gummel–Poon model have been chosen because they intuitive and better reflect both physical and circuit design thinking.

The DC model is defined by the following parameters:

- `is`, `bf`, `nf`, `ise`, `ikf`, and `ne` determine the forward current gain characteristics.
- `is`, `br`, `nr`, `isc`, `kr`, and `nc` determine the reverse current gain characteristics.
- `vaf` and `var` determine the output conductance for forward and reverse regions.

Three Ohmic resistances `rb`, `rc`, and `re` are included, where `rb` can be high current–dependent. Base charge storage is modeled by forward and reverse transit times, `tf` and `tr`; the forward transit time `tf` being bias-dependent if required, and nonlinear depletion layer capacitances, which are determined by `cje`, `vje`, and `mje` for the base–emitter junction; `cjc`, `vjc`, and `mjc` for the base–collector junction; and `cjs`, `vjs`, and `mjs` for the collector–substrate junction.

The temperature dependency of the saturation current, `is`, is determined by the energy gap, `eg`, and the saturation current temperature exponent, `xti`. In addition, base current temperature dependency is modeled by the beta temperature exponent `xtb` in the new model. The values specified are assumed to have been measured at the temperature `tnom`. The `temp` value is the temperature at which the device will operate.

An npn transistor is obtained by specifying `npn=1`, and a pnp transistor is obtained by specifying `pnp=1`.

Device name:	`BJT`
Default parameter set name:	`BJT_pset`
Electrodes:	`collector, base, emitter, substrate`
Internal variables:	`collector` (internal collector voltage, only available if `rc` $\neq 0$) `base` (internal base voltage, only available if `rb` $\neq 0$) `emitter` (internal emitter voltage, only available if `re` $\neq 0$)

Table 44 BJT model parameters

Name	Description	Type	Default	Unit
`npn`	npn-type device	integer	1	–
`pnp`	pnp-type device	integer	0	–
`is`	Saturation current	double	1e-16	A
`bf`	Ideal forward beta	double	100	–
`nf`	Forward emission coefficient	double	1	–
`vaf`	Forward Early voltage	double	∞	V
`va`	(redundant parameter)	double	∞	V
`ikf`	Forward beta roll-off corner current	double	∞	A
`ik`	(redundant parameter)	double	∞	A
`ise`	Base–emitter leakage saturation current	double	0	A
`ne`	Base–emitter leakage emission coefficient	double	1.5	–
`br`	Ideal reverse beta	double	1	–
`nr`	Reverse emission coefficient	double	1	–
`var`	Reverse Early voltage	double	∞	V
`vb`	(redundant parameter)	double	∞	V
`ikr`	Reverse beta roll-off corner current	double	∞	A

Table 44 BJT model parameters (Continued)

Name	Description	Type	Default	Unit
`isc`	Base–collector leakage saturation current	double	0	A
`nc`	Base–collector leakage emission coefficient	double	2	–
`rb`	Zero bias base resistance	double	0	Ω
`irb`	Current for base resistance=(rb+rbm)/2	double	∞	A
`rbm`	Minimum base resistance	double	0	Ω
`re`	Emitter resistance	double	0	Ω
`rc`	Collector resistance	double	0	Ω
`cje`	Zero bias base–emitter depletion capacitance	double	0	F
`vje`	Base–emitter built-in potential	double	0.75	V
`pe`	(redundant parameter)	double	0.75	V
`mje`	Base–emitter junction grading coefficient	double	0.33	–
`me`	(redundant parameter)	double	0.33	–
`tf`	Ideal forward transit time	double	0	s
`xtf`	Coefficient for bias dependence of `tf`	double	0	–
`vtf`	Voltage giving VBC dependence of `tf`	double	∞	V
`itf`	High-current dependence of `tf`	double	0	A
`ptf`	Excess phase	double	0	degree
`cjc`	Zero bias base–collector depletion capacitance	double	0	F
`vjc`	Base–collector built-in potential	double	0.75	V
`pc`	(redundant parameter)	double	0.75	V
`mjc`	Base–collector junction grading coefficient	double	0.33	–
`mc`	(redundant parameter)	double	0.33	–
`xcjc`	Fraction of base–collector cap. to internal base	double	1	–
`tr`	Ideal reverse transit time	double	0	s
`cjs`	Zero bias collector–source capacitance	double	0	F
`ccs`	(redundant parameter)	double	0	F
`vjs`	Substrate junction built-in potential	double	0.75	V
`ps`	(redundant parameter)	double	0.75	V
`mjs`	Substrate junction grading coefficient	double	0	–

Table 44 BJT model parameters (Continued)

Name	Description	Type	Default	Unit
`ms`	(redundant parameter)	double	0	–
`xtb`	Forward and reverse beta temperature exponent	double	0	–
`eg`	Energy gap for IS temperature dependency	double	1.11	eV
`xti`	Temperature exponent for IS	double	3	–
`fc`	Forward bias junction fit parameter	double	0.5	–
`tnom`	Parameter measurement temperature	double	27	°C
`kf`	Flicker noise coefficient	double	0	–
`af`	Flicker noise exponent	double	1	–

Table 45 BJT instance parameters

Name	Description	Type	Default	Unit
`area`	Area factor	double	1	–
`ic`	Initial condition vector	double[2]	–	V
`icvbe`	Initial base–emitter voltage	double	0	V
`icvce`	Initial collector–emitter voltage	double	0	V
`off`	Device initially off	integer	0	–
`temp`	Instance temperature	double	27	°C

Junction Field Effect Transistor

The junction field effect transistor (JFET) model is derived from the FET model of Shichman and Hodges. The DC characteristics are defined by the following parameters:

- `vto` and `beta` determine the variation of drain current with gate voltage.
- `lambda` determines the output conductance.
- `is` is the saturation current of the two gate junctions.

Two Ohmic resistances, `rd` and `rs`, are included. Charge storage is modeled by nonlinear depletion-layer capacitances for both gate junctions, which vary as the –1/2 power of junction voltage and are defined by the parameters `cgs`, `cgd`, and `pb`.

The `temp` value is the temperature at which the device will operate. A fitting parameter `b` is also available [11].

The type of the transistor must be specified by setting either `njf=1` or `pjf=1`.

Device name:	`JFET`
Default parameter set name:	`JFET_pset`
Electrodes:	`Drain, Gate, Source`
Internal variables:	`source` (internal source voltage, only available if `rs` $\neq 0$) `drain` (internal drain voltage, only available if `rd` $\neq 0$)

Table 46 JFET model parameters

Name	Description	Type	Default	Unit
`af`	Flicker noise exponent	double	1	–
`b`	Doping tail parameter	double	1	–
`beta`	Transconductance parameter	double	0.0001	A/V^2
`cgd`	Gate–drain junction cap	double	0	F
`cgs`	Gate–source junction capacitance	double	0	F
`fc`	Forward bias junction fit parameter	double	0.5	–
`is`	Gate junction saturation current	double	1e-14	A
`kf`	Flicker noise coefficient	double	0	–
`lambda`	Channel-length modulation parameter	double	0	V^{-1}
`njf`	N-type JFET model	integer	1	–
`pb`	Gate junction potential	double	1	V
`pjf`	P-type JFET model	integer	0	–
`rd`	Drain Ohmic resistance	double	0	Ω
`rs`	Source Ohmic resistance	double	0	Ω
`tnom`	Parameter measurement temperature	double	27	°C
`vt0`	Threshold voltage	double	–2	V
`vto`	(redundant parameter)	double	–2	V

Table 47 JFET instance parameters

Name	Description	Type	Default	Unit
`area`	Area factor	double	1	–
`ic`	Initial V_{DS},V_{GS} vector	double[2]	–	V
`ic-vds`	Initial drain–source voltage	double	0	V

Table 47 JFET instance parameters (Continued)

Name	Description	Type	Default	Unit
ic-vgs	Initial gate–source voltage	double	0	V
off	Device initially off	integer	0	–
temp	Instance temperature	double	27	°C

GaAs MESFET

This model is derived from the GaAs FET model [12]. The DC characteristics are defined by the following parameters:

- vto, b, and beta determine the variation of drain current with gate voltage.
- alpha determines saturation voltage.
- lambda determines the output conductance.

The formulas are given by:

$$\begin{aligned} I_d &= \frac{\beta \cdot (V_{gs} - V_T)^2}{1 + b \cdot (V_{gs} - V_T)}\left(1 - \left(1 - \alpha\frac{V_{ds}}{3}\right)^3\right)(1 + \lambda \cdot V_{ds}) && \text{for } 0 < \mathrm{V}_{ds} < \frac{3}{\alpha} \\ I_d &= \frac{\beta \cdot (V_{gs} - V_T)^2}{1 + b \cdot (V_{gs} - V_T)} \cdot (1 + \lambda \cdot V_{ds}) && \text{for } V_{ds} > \frac{3}{\alpha} \end{aligned} \tag{11}$$

Two Ohmic resistances, rd and rs, are included. Charge storage is modeled by total gate charge as a function of gate–drain and gate–source voltages and is defined by the parameters cgs, cgd, and pb.

Use nmf=1 to specify an n-type device or pmf=1 to specify a p-type device.

Device name:	MES
Default parameter set name:	MES_pset
Electrodes:	Drain, Gate, Source
Internal variables:	source (internal source voltage, only available if rs $\neq 0$) drain (internal drain voltage, only available if rd $\neq 0$)

Table 48 MESFET model parameters

Name	Description	Type	Default	Unit
`nmf`	N-type MESFET model	integer	1	–
`pmf`	P-type MESFET model	integer	0	–
`vt0`	Pinch-off voltage	double	–2	V
`vto`	(redundant parameter)	double	–2	V
`alpha`	Saturation voltage parameter	double	2	V^{-1}
`beta`	Transconductance parameter	double	0.0025	A/V^2
`lambda`	Channel length modulation parameter	double	0	V^{-1}
`b`	Doping tail extending parameter	double	0.3	V^{-1}
`rd`	Drain Ohmic resistance	double	0	Ω
`rs`	Source Ohmic resistance	double	0	Ω
`cgs`	Gate–source junction capacitance	double	0	F
`cgd`	Gate–drain junction capacitance	double	0	F
`pb`	Gate junction potential	double	1	V
`is`	Junction saturation current	double	1e-14	–
`fc`	Forward bias junction fit parameter	double	0.5	–
`kf`	Flicker noise coefficient	double	0	–
`af`	Flicker noise exponent	double	1	–

Table 49 MESFET instance parameters

Name	Description	Type	Default	Unit
`area`	Area factor	double	1	–
`icvds`	Initial drain–source voltage	double	0	V
`icvgs`	Initial gate–source voltage	double	0	V

References

[1] A. Vladimirescu and S. Liu, *The Simulation of MOS Integrated Circuits Using SPICE2*, Memorandum UCB/ERL M80/7, Electronics Research Laboratory, University of California, Berkeley, CA, USA, February 1980.

[2] T. Sakurai and A. R. Newton, *A Simple MOSFET Model for Circuit Analysis and Its Application to CMOS Gate Delay Analysis and Series-Connected MOSFET Structure*, Memorandum UCB/ERL M90/19, Electronics Research Laboratory, University of California, Berkeley, CA, USA, March 1990.

[3] B. J. Sheu, D. L. Scharfetter, and P. K. Ko, *SPICE2 Implementation of BSIM*, Memorandum UCB/ERL M85/42, Electronics Research Laboratory, University of California, Berkeley, CA, USA, May 1985.

[4] J. R. Pierret, *A MOS Parameter Extraction Program for the BSIM Model*, Memorandum UCB/ERL M84/99, Electronics Research Laboratory, University of California, Berkeley, CA, USA, November 1984.

[5] J. R. Pierret, *A MOS Parameter Extraction Program for the BSIM Model, Appendix 5: BSIM1.0 Pascal Source Code*, Memorandum UCB/ERL M84/100, Electronics Research Laboratory, University of California, Berkeley, CA, USA, November 1984.

[6] M.-C. Jeng, *Design and Modeling of Deep-Submicrometer MOSFETs*, Memorandum UCB/ERL M90/90, Electronics Research Laboratory, University of California, Berkeley, CA, USA, October 1990.

[7] Y. Cheng *et al.*, *BSIM3v3 Manual*, Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, CA, USA, 1996.

[8] For more information about the BSIM3 model, go to http://bsim.berkeley.edu/models/bsim3/.

[9] S. Pak, *Analysis and SPICE Implementation of High Temperature Effects on MOSFETs*, Master’s thesis, University of California, Berkeley, CA, USA, 1986.

[10] C. K. Szeto, *BSIM-CAD Modeling of Thermal Effects in MOSFET Circuits*, Master’s thesis, University of California, Berkeley, CA, USA, 1988.

[11] A. E. Parker and D. J. Skellern, “An Improved FET Model for Computer Simulators,” *IEEE Transactions on Computer-Aided Design*, vol. 9, no. 5, pp. 551–553, 1990.

[12] H. Statz *et al.*, “GaAs FET Device and Circuit Simulation in SPICE,” *IEEE Transactions on Electron Devices*, vol. ED-34, no. 2, pp. 160–169, 1987.

CHAPTER 2 HSPICE Models

This chapter describes the available Synopsys HSPICE® models.

Overview of Available Models

The following HSPICE MOS models are available:

- Level 1 IDS: Shichman–Hodges model
- Level 2 IDS: Grove–Frohman model
- Level 3 IDS: Empirical model
- Level 28 modified BSIM model
- Level 49 and Level 53 BSIM3v3 MOS models
- Level 54 BSIM4 model
- Level 57 UC Berkeley BSIM3-SOI model
- Level 59 UC Berkeley BSIM3-SOI fully depleted (FD) model
- Level 61 RPI a-Si TFT model
- Level 62 RPI Poly-Si TFT model
- Level 64 STARC HiSIM model
- Level 68 STARC HiSIM2 model
- Level 69 PSP100 DFM support series model
- Level 72 BSIM-CMG multigate MOSFET model
- Level 73 STARC HiSIM-LDMOS/HiSIM-HV model
- Level 76 LETI-UTSOI MOSFET model

Refer to the *HSPICE® Reference Manual: MOSFET Models* for detailed descriptions of these models.

To use a particular HSPICE model, a parameter set must be defined in an `.scf` file.

To define an NMOS transistor, specify:

```
nmos = 1
```

To define a PMOS transistor, specify:

```
pmos = 1
```

NOTE You must define either the `nmos` or `pmos` parameter. Parameter names are case sensitive, and all names must be specified in lowercase.

The built-in defaults in Sentaurus Device and Sentaurus Interconnect might be different compared to the defaults in the HSPICE circuit simulator. Therefore, it is recommended to define all known parameters explicitly in a parameter set.

Level 1 IDS: Shichman–Hodges Model

Device name:	`HMOS_L1`
Electrodes:	`Drain, Gate, Source, Bulk`

Level 2 IDS: Grove–Frohman Model

Device name:	`HMOS_L2`
Electrodes:	`Drain, Gate, Source, Bulk`

Level 3 IDS: Empirical Model

Device name:	`HMOS_L3`
Electrodes:	`Drain, Gate, Source, Bulk`

Level 28 Modified BSIM Model

Device name:	`HMOS_L28`
Electrodes:	`Drain, Gate, Source, Bulk`

Level 49 BSIM3v3 MOS Model

Device name:	`HMOS_L49`
Electrodes:	`Drain, Gate, Source, Bulk`

Level 53 BSIM3v3 MOS Model

Device name:	`HMOS_L53`
Electrodes:	`Drain, Gate, Source, Bulk`

Level 54 BSIM4 Model

Not all model selectors are supported. The following restrictions must be observed:

- Use `rgatemod = 0`: No gate resistance model is supported.
- Use `rbodymod = 0`: No body resistance network is supported.
- Use `rdsmod = 0`: External source/drain resistances are not supported.
- Use `trnqsmod = 0`: Transient non-quasistatic (NQS) models are not supported.
- Use `acnqsmod = 0`: AC NQS models are not supported.

Device name:	`HMOS_L54`
Electrodes:	`Drain, Gate, Source, Bulk`

Level 57 UC Berkeley BSIM3-SOI Model

Device name:	`HMOS_L57`
Electrodes:	`Drain, Gate, Source, Backgate, PP, Body, Temp`

Level 59 UC Berkeley BSIM3-SOI Fully Depleted (FD) Model

Device name:	`HMOS_L59`
Electrodes:	`Drain, Gate, Source, Backgate, PP, Body, Temp`

Level 61 RPI a-Si TFT Model

The bulk node is currently not used by the model. It can simply be connected to the ground.

Device name:	`HMOS_L61`
Electrodes:	`Drain, Gate, Source, Bulk`

Level 62 RPI Poly-Si TFT Model

The bulk node is currently not used by the model. It can simply be connected to the ground.

Device name:	`HMOS_L62`
Electrodes:	`Drain, Gate, Source, Bulk, Temp`

Level 64 STARC HiSIM Model

Device name:	`HMOS_L64`
Electrodes:	`Drain, Gate, Source, Bulk`

Level 68 STARC HiSIM2 Model

Device name:	`HMOS_L68`
Electrodes:	`Drain, Gate, Source, Bulk`

Level 69 PSP100 DFM Support Series Model

Device name:	`HMOS_L69`
Electrodes:	`Drain, Gate, Source, Bulk`

Level 72 BSIM-CMG Multigate MOSFET Model

The charge segmentation model (`nqsmod=3`) is not supported.

Device name:	`HMOS_L72`
Electrodes:	`Drain, Gate, Source, Bulk, Temp`

Level 73 STARC HiSIM-LDMOS/HiSIM-HV Model

Device name:	`HMOS_L73`
Electrodes:	`Drain, Gate, Source, Bulk, Substrate, Temp`

Level 76: LETI-UTSOI MOSFET Model

Device name:	`HMOS_L76`
Electrodes:	`Drain, Gate, Source, Bulk, Temp`

CHAPTER 3 Built-in Models of Sentaurus Device

This chapter describes the available built-in models of Sentaurus Device.

Parameter Interface Model

The parameter interface is a model that acts as an interface to the parameters of other compact models. It feeds voltages or temperatures as parameters into compact models, that is, the parameter interface ensures that a parameter in a compact model always has the same value as a system variable (for example, voltage and temperature).

This interface is required to use SPICE models in an electrothermal simulation. The assumption for SPICE models is that their temperature remains constant throughout a simulation. Therefore, their operating temperature is specified as a parameter.

However, this assumption is not valid for an electrothermal simulation. A mechanism is required to update the temperature parameter in a SPICE model whenever the corresponding temperature variable changes. The parameter interface can perform this operation.

The interface has only one electrode, which must be connected to the electrode or thermode that represents the required system variable. The value of the parameter identifies the circuit element and its parameter, which must be coupled with the system variable. The value of the parameter must have the form `instance.name`. In general, the value of the parameter can be expressed as a function of the node value u:

$$value = offset + c_1 \cdot u + c_2 \cdot u^2 + c_3 \cdot u^3 \tag{12}$$

By default, $value = u$.

Device name:	`Param_Interface_Device`
Default parameter set name:	`Param_Interface`
Electrodes:	`u`
Internal variables:	None

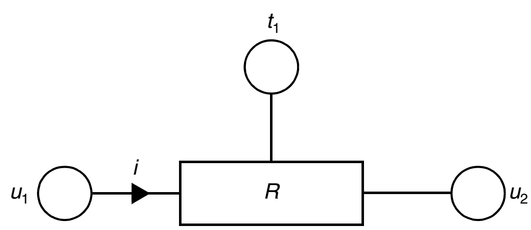
NOTE There are no parameters for this parameter set.

Table 50 Parameter interface instance parameters

Name	Description	Type	Default	Unit
`c1`	Linear coefficient	double	1	–
`c2`	Quadratic coefficient	double	0	–
`c3`	Cubic coefficient	double	0	–
`offset`	Offset value	double	0	–
`parameter`	Name of parameter	string	" "	–

Electrothermal Resistor (Ter) Model

The built-in model `Ter` simulates electrothermal resistance. This device has three contacts (two electrodes and one thermode):



The electrical behavior of the resistance is described by Ohm's law:

$$u_1 - u_2 = R \cdot i \tag{13}$$

The resistance of the device, however, depends on the thermode temperature:

$$R = r \cdot (1 + \alpha \cdot (t_1 - t_{ref}) + \beta \cdot (t_1 - t_{ref})^2) \tag{14}$$

The device also produces Joule heat, which is dissipated through the thermode:

$$P = (u_1 - u_2) \cdot i \tag{15}$$

In Example: Implementing the Electrothermal Resistor Model on page 161, the implementation of an electrothermal resistor using the compact model interface is discussed.

Device name:	`Ter`
Default parameter set name:	`Ter_pset`
Electrodes:	`u1, u2`
Thermodes:	`t1`
Internal variables:	None

NOTE There are no parameters for this parameter set.

Table 51 Electrothermal resistor instance parameters

Name	Description	Type	Default	Unit
`alpha`	Linear temperature coefficient	double	0	K^{-1}
`beta`	Quadratic temperature coefficient	double	0	K^{-2}
`r`	Resistance	double	1	Ω
`tref`	Reference temperature	double	300.15	K

MOS Harness Model

A standard MOSFET compact model uses four electrodes (drain, gate, source, bulk) to describe its electrical behavior. To use such a model in an electrothermal simulation, it is preferable to capture its power as well. The power generated by a MOSFET is given by:

$$P = i_{\mathrm{d}}u_{\mathrm{d}} + i_{\mathrm{g}}u_{\mathrm{g}} + i_{\mathrm{s}}u_{\mathrm{s}} + i_{\mathrm{b}}u_{\mathrm{b}} \tag{16}$$

where:

- i_{d}, i_{g}, i_{s}, and i_{b} denote the currents at the drain, gate, source, and bulk, respectively.
- u_{d}, u_{g}, u_{s}, and u_{b} denote the voltages at the drain, gate, source, and bulk, respectively.

A MOS harness as shown in Figure 7 on page 120 is an auxiliary compact model that has been designed to inject the power generated by the MOSFET into the thermal circuit. It acts as an interface between the MOSFET and the rest of the circuit.

The four internal electrodes $\mathrm{d_{int}}$, $\mathrm{g_{int}}$, $\mathrm{s_{int}}$, and $\mathrm{b_{int}}$ are connected to the MOSFET to monitor the voltages and currents at the four electrodes. The four external electrodes $\mathrm{d_{ext}}$, $\mathrm{g_{ext}}$, $\mathrm{s_{ext}}$, and $\mathrm{b_{ext}}$ connect the device to the rest of the circuit. Finally, the thermal contact t feeds the power generated by the MOSFET into the thermal circuit.

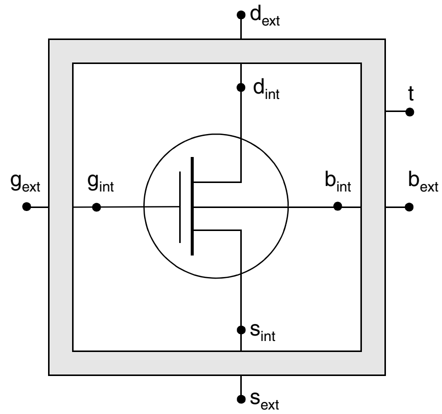


Figure 7 MOS harness

NOTE The temperature at the thermal contact t is determined only by the solution of the equations in the thermal circuit. It is not possible to provide this temperature to the MOSFET compact model.

Device name:	`MOS_harness`
Default parameter set name:	`MOS_harness_pset`
Electrodes:	`drain_ext, gate_ext, source_ext, bulk_ext,` `drain_int, gate_int, source_int, bulk_int`
Thermodes:	`t`
Internal variables:	`i_drain, i_gate, i_source, i_bulk`

NOTE There are no parameters for this device.

Example

The following Sentaurus Device simulation ramps the gate voltage of an HSPICE Level 1 NMOSFET from 0 V to 5 V. The temperature at the node `tmos` is determined by:

$$t_{\mathrm{mos}} = t_{300} + R_t P \tag{17}$$

where:

- $P = i_{\mathrm{d}} u_{\mathrm{d}}$ is the power of the NMOSFET.
- $R_t = 10\ \mathrm{K/W}$ is the value of the thermal resistance.

The command file for this simulation is:

```
File {
   Output = "MOS_harness"
   SPICEPath = "."
}

System {
   Thermal (tmos t300)
   Set (t300 = 300)

   Vsource_pset Vg (g 0) { pwl = (0 0 1 5) }
   Vsource_pset Vd (d 0) { dc = 5 }

   MOS_harness_pset harness (d g 0 0 di gi si bi tmos)
   l1_nmos mos (di gi si bi) { l=1e-6 w=1e-4 }
   Resistor_pset rt (tmos t300) { resistance = 10 }

   Plot "MOS_harness_circuit_des.plt" (
      time() v(g) i(harness d) t(tmos) h(harness tmos) h(rt t300)
   )
}

Solve {
   Transient (
      InitialTime = 0
      FinalTime = 1
      InitialStep = 0.01
      MaxStep = 0.01
   )
   { Coupled { Circuit TCircuit } }
}
```

Ferroelectric Capacitor Model

The Landau–Khalatnikov equation [1][2] is the standard equation for ferroelectric modeling and it reads:

$$\rho\frac{dP}{dt} + \nabla_{\mathrm{P}}U = 0 \tag{18}$$

where ρ is the viscosity and P is the polarization vector.

The free energy U of the ferroelectric is:

$$U = \alpha P^2 + \beta P^4 + \gamma P^6 - E \cdot P \tag{19}$$

Combining Eq. 18 and Eq. 19 returns the electric field:

$$E = 2\alpha P + 4\beta P^3 + 6\gamma P^5 + \rho\frac{dP}{dt} \tag{20}$$

The ferroelectric is treated as a capacitor connected to the gate of the FET. Figure 8 shows the equivalent circuit.

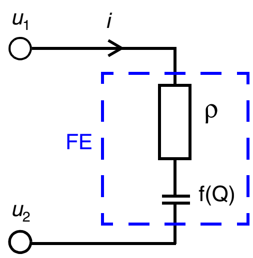


Figure 8 Equivalent circuit for the ferroelectric compact model interface

The following equation is satisfied:

$$\begin{aligned} u_1 - u_2 &= \Big(\underbrace{2\alpha Q + 4\beta Q^3 + 6\gamma Q^5}_{f(Q)} + \rho\frac{\partial Q}{\partial t}\Big)t_{\mathrm{fe}} \\ i &= \frac{\partial Q}{\partial t} \\ f'(Q) &= 2\alpha + 12\beta Q^2 + 30\gamma Q^4 \end{aligned} \tag{21}$$

Device name:	`ferroelectric`
Default parameter set name:	`ferroelectric_pset`
Electrodes:	`u1, u2`
Internal variables:	`i, Q`

Table 52 Parameters for ferroelectric capacitor

Name	**Description**	**Type**	**Default**	**Unit**
`t`	Ferroelectric thickness	double	60e-3	μm
`alpha`	α parameter	double	–4e7	m/F
`beta`	β parameter	double	–5e6	$m^5/(FC^2)$
`gamma`	γ parameter	double	5e7	$m^9/(FC^4)$
`rho`	Viscosity parameter	double	0	Ωm
`Cint`	Parameter to improve convergence	double	2.5e7	m/F

Table 52 Parameters for ferroelectric capacitor (Continued)

Name	Description	Type	Default	Unit
`A`	External MOS gate area	double	1e-8	cm^2
`alpha0`	$\alpha = \alpha_0(T - T_c)$	double	0	m/(FK)
`Tc`	Curie temperature	double	665.7	K
`T`	Temperature	double	300	K

NOTE When $\alpha_0 = 0$, the model takes parameter α. If $\alpha_0 \neq 0$ is defined in the input file, the model solves $\alpha = \alpha_0(T - T_c)$ and ignores parameter α. C_{int} helps convergence at the switching point when $f'(Q) \approx 0$. The value of C_{int} is in the same order as parameter α. A larger C_{int} can improve convergence in general, but at the risk of introducing artifacts at the switching point.

Example

The following example solves the I_d–V_g curve and the Q–V_g (P–E) curve for an NMOSFET based on the BSIM4 model. The gate of the NMOS is connected to the ferroelectric capacitor. Ferroelectric parameters are taken from [3]. The command file is:

```
File {
   Output = "NMOS"
   Current = "NMOS"
   SPICEPath = "."
}

Electrode {
   { Name="gate" voltage=0 }
   { Name="source" voltage=0 }
   { Name="drain" voltage=0 }
   { Name="substrate" voltage=0 }
}

System {
   Vsource_pset vin ("gate" 0) { pwl = (0 0 4e -2 12e-2 2 20e-2 -2) }
   Vsource_pset vdd ("drain" 0) { dc=1 }

   ferroelectric_pset ferroelectric_inst ("gate" 2)
      {t=10e-3 alpha=-1.8e9 beta=5.8e12 gamma=0 A=32e-11}
   Resistor_pset r1 ("source" 0) { resistance=1e-8 }
   NMOS_comp_pset0 M2 ("drain" 2 "source" 0) { w=1.e-6 l=32e-9 }
   Plot "NMOS_des.plt" (time () v("gate") v(2) v("drain") i(M2,"source")
      ferroelectric_inst.Q)
}
```

```
Math {
   -ExtendedPrecision
   RhsMin = 1e-10
}

Solve {
   Coupled {Circuit}
   Transient (InitialTime = 0.0 FinalTime = 4e-2 MinStep = 1e-5
              InitialStep = 1e-4 MaxStep = 4e-4) { Coupled {Circuit} }
   NewCurrentPrefix="transient_"
   Transient (InitialTime = 4e-2 FinalTime = 20e-2 MinStep = 1e-5
              InitialStep = 1e-4 MaxStep = 8e-4) { Coupled {Circuit} }
}
```

The output `ferroelectric_inst.Q` is the charge of the ferroelectric, in units of C/m^2.

NOTE Transient simulations are performed because quasistationary behavior is undefined in this case (two ideal capacitors in serial – a ferroelectric and a MOS gate oxide). The first transient simulation finds the stable operating point of the system, by sweeping the gate bias to a sufficiently large value such that the ferroelectric operates well at the stable branch. The second transient simulation generates the required results.

References

[1] G. Pahwa *et al.*, "Analysis and Compact Modeling of Negative Capacitance Transistor with High ON-Current and Negative Output Differential Resistance—Part I: Model Description," *IEEE Transactions on Electron Devices*, vol. 63, no. 12, pp. 4981–4985, 2016.

[2] G. Pahwa *et al.*, "Analysis and Compact Modeling of Negative Capacitance Transistor with High ON-Current and Negative Output Differential Resistance—Part II: Model Validation," *IEEE Transactions on Electron Devices*, vol. 63, no. 12, pp. 4986–4992, 2016.

[3] M. A. Wahab and M. A. Alam, *A Verilog-A Compact Model for Negative Capacitance FET*, Version 1.1.0, Purdue University, West Lafayette, IN, USA, April 2016.

CHAPTER 4 **Compact Model Interface in Sentaurus Device**

This chapter describes the compact model interface in Sentaurus Device.

Introduction

Sentaurus Device provides a compact model interface (CMI) for user-defined compact models. The models are implemented in C++ and are linked to Sentaurus Device at runtime. No access to the source code of Sentaurus Device is necessary.

To implement your own CMI model, you need to know the structure and the analytic model equations of your model. The structure of your CMI model will be formalized in a compact circuit (`.ccf`) file.

You will provide the executable functions for the simulator using the C++ interface in a `.C` file. The special `cmi` compiler translates this `.C` file into a dynamically linkable library (`.so`) file. Sentaurus Device will finally require the `.ccf` file and `.so` file.

Analytical Description of CMI Models

This section describes how to present your CMI model analytically and how this description is used in the simulator.

Sentaurus Device Analysis Methods

Sentaurus Device supports several analysis methods, namely, DC, transient, AC, noise, and harmonic balance (HB), which require different CMI functions.

The standard set of CMI functions (CMI-STD) supports DC, transient, AC, and noise analysis. In addition, the one-tone HB analysis mode SDFT requires only the CMI-STD function set, while the multitone HB analysis mode MDFT requires additional functionality denoted as the CMI-HB-MDFT function set.

See Sentaurus™ Device User Guide, Harmonic Balance on page 101 for a description of the different modes of harmonic balance simulations.

The CMI-STD function set is mandatory for all CMI models. The CMI-HB-MDFT set is optional, but it is required if HB-MDFT simulations are performed. Sentaurus Device automatically detects if the function set is present and uses these functions with HB-MDFT simulations. A set of provided CMI models supporting the CMI-HB-MDFT function set is described in CMI Models With Frequency-Domain Assembly on page 167.

Time-Domain Model Equations

Sentaurus Device solves differential algebraic systems of the form:

$$\frac{d}{dt}q(t, \varsigma(t)) + f(t, \varsigma(t)) = 0 \tag{22}$$

The time-dependent vector $\varsigma(t)$ consists of all the unknown system variables from all physical devices and compact models.

Each compact model works only on a subset $z(t)$ of all unknowns $\varsigma(t)$. For example, assume that the model has n_u electrodes, n_τ thermodes, and n_x internal variables.

The vector $z(t)$ of unknowns is given by:

$$z(t) = \begin{bmatrix} u_1 \\ \dots \\ u_{n_u} \\ \tau_1 \\ \dots \\ \tau_{n_\tau} \\ x_1 \\ \dots \\ x_{n_x} \end{bmatrix} \tag{23}$$

where $u_1, \dots, u_{n_u}$ are electrode voltages, $\tau_1, \dots, \tau_{n_\tau}$ are thermode temperatures, and $x_1, \dots,$ x_{n_x} are internal variables. For each unknown in the vector $z(t)$, a corresponding equation must be provided.

Sentaurus Device requires that the equations are given in the form:

$$\frac{d}{dt}\begin{bmatrix} q_{u_1}(t, z(t)) \\ \dots \\ q_{u_{n_u}}(t, z(t)) \\ q_{\tau_1}(t, z(t)) \\ \dots \\ q_{\tau_{n_\tau}}(t, z(t)) \\ q_{x_1}(t, z(t)) \\ \dots \\ q_{x_{n_x}}(t, z(t)) \end{bmatrix} + \begin{bmatrix} f_{u_1}(t, z(t)) \\ \dots \\ f_{u_{n_u}}(t, z(t)) \\ f_{\tau_1}(t, z(t)) \\ \dots \\ f_{\tau_{n_\tau}}(t, z(t)) \\ f_{x_1}(t, z(t)) \\ \dots \\ f_{x_{n_x}}(t, z(t)) \end{bmatrix} = \begin{bmatrix} \text{current from electrode 1 into the model} \\ \dots \\ \text{current from electrode } n_u \text{ into the model} \\ \text{heat flow from thermode 1 into the model} \\ \dots \\ \text{heat flow from thermode } n_\tau \text{ into the model} \\ 0 \\ \dots \\ 0 \end{bmatrix} \tag{24}$$

The first n_u components represent the currents flowing from the electrodes into the model. Similarly, the next n_τ components represent the heat flows from the thermodes. The last n_x equations are specific to the model.

In Sentaurus Device, the vector:

$$q(t, z(t)) = \begin{bmatrix} q_{u_1}(t, z(t)) \\ \dots \\ q_{u_{n_u}}(t, z(t)) \\ q_{\tau_1}(t, z(t)) \\ \dots \\ q_{\tau_{n_\tau}}(t, z(t)) \\ q_{x_1}(t, z(t)) \\ \dots \\ q_{x_{n_x}}(t, z(t)) \end{bmatrix} \tag{25}$$

is also called the transient right-hand side, and the vector:

$$f(t, z(t)) = \begin{bmatrix} f_{u_1}(t, z(t)) \\ \dots \\ f_{u_{n_u}}(t, z(t)) \\ f_{\tau_1}(t, z(t)) \\ \dots \\ f_{\tau_{n_\tau}}(t, z(t)) \\ f_{x_1}(t, z(t)) \\ \dots \\ f_{x_{n_x}}(t, z(t)) \end{bmatrix} \tag{26}$$

is called the DC right-hand side. The entries in the vectors q and f must be computed by user code. Sentaurus Device deals with the differentiation with respect to time and the proper insertion into the global system of equations (see Eq. 22).

As Sentaurus Device uses the Newton method to solve Eq. 22, the following Jacobians are also required:

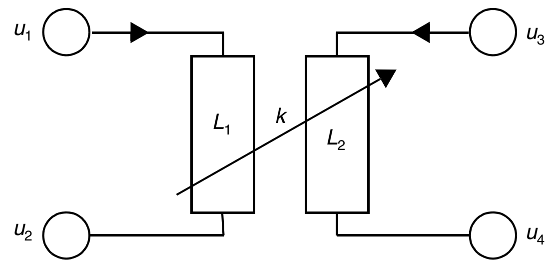
$$J_q = \frac{d}{dz}q(t, z(t)) = \begin{bmatrix} \frac{d}{du_1}q_{u_1}(t, z(t)) & \dots & \frac{d}{dx_{n_x}}q_{u_1}(t, z(t)) \\ \dots & \dots & \dots \\ \frac{d}{du_1}q_{x_{n_x}}(t, z(t)) & \dots & \frac{d}{dx_{n_x}}q_{x_{n_x}}(t, z(t)) \end{bmatrix} \tag{27}$$

and:

$$J_f = \frac{d}{dz}f(t, z(t)) = \begin{bmatrix} \frac{d}{du_1}f_{u_1}(t, z(t)) & \dots & \frac{d}{dx_{n_x}}f_{u_1}(t, z(t)) \\ \dots & \dots & \dots \\ \frac{d}{du_1}f_{x_{n_x}}(t, z(t)) & \dots & \frac{d}{dx_{n_x}}f_{x_{n_x}}(t, z(t)) \end{bmatrix} \tag{28}$$

Example: Coupled Inductance

As an example, consider a coupled inductance:



The behavior of the coupled inductance is described by the equations:

$$
\begin{aligned}
u_1 - u_2 &= L_1\frac{di_1}{dt} + m\frac{di_2}{dt} \\
u_3 - u_4 &= m\frac{di_1}{dt} + L_2\frac{di_2}{dt}
\end{aligned}
\tag{29}
$$

where:

$$
m = k\sqrt{L_1 \cdot L_2}
\tag{30}
$$

The vector of unknowns is given by:

$$
z(t) = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ i_1 \\ i_2 \end{bmatrix}
\tag{31}
$$

The equations in Eq. 29 can be transformed into the form Eq. 24 by defining the vectors q and f:

$$
q(t, z(t)) = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ -L_1 \cdot i_1 - m \cdot i_2 \\ -m \cdot i_1 - L_2 \cdot i_2 \end{bmatrix} \qquad f(t, z(t)) = \begin{bmatrix} i_1 \\ -i_1 \\ i_2 \\ -i_2 \\ u_1 - u_2 \\ u_3 - u_4 \end{bmatrix}
\tag{32}
$$

The Jacobians of q and f are given by:

$$\frac{d}{dz}q(t, z(t)) = J_q = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -L_1 & -m \\ 0 & 0 & 0 & 0 & -m & -L_2 \end{bmatrix}$$

$$\frac{d}{dz}f(t, z(t)) = J_f = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & -1 \\ 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 \end{bmatrix} \tag{33}$$

See Example: Implementing Coupled Inductances on page 156.

Frequency-Domain Model Equations

Using the time-domain model equations as a starting point, the harmonic balance analysis (HB-MDFT) is used to solve ordinary differential equations of the following form in the frequency domain:

$$\frac{d}{dt}q(\eta(t)) + f(\eta(t)) = 0 \tag{34}$$

In contrast to the time-domain model equations, both the conductive part f and the capacitive part q are not explicitly time dependent.

Fourier transformation of the equation leads to an algebraic equation of the form:

$$L(X) = 0 \tag{35}$$

where X represents the spectra of all transient solution variables, and L represents the spectra of the time-domain equation residuals.

The harmonic balance analysis uses a finite Fourier representation in the frequency domain. Let $f_1, ..., f_K$ be a finite set of (different) positive frequencies and let $\omega_k := 2\pi f_k$ be the associated circular frequencies.

The used Fourier representation of a real-valued scalar function is then given by:

$$x(t) = X_0 + \sum_{k \in K} [X_k \exp(i\omega_k t) + X_k^* \exp(-i\omega_k t)] \tag{36}$$

where the complex-valued vector $X = (X_0, X_1, \ldots, X_K)^T$ is the spectrum of the function x.

Restricting the considerations to one compact model, you obtain the following. Let $N = n_u + n_\tau + n_x$ be the total number of variables of a compact model, and denote:

$$l_m(z(t)) := \frac{d}{dt} q_m(z(t)) + f_m(z(t)) \tag{37}$$

as the m-th residual (where m represents an electrode, a thermode, or an internal equation and $1 \le m \le N$). Letting L_m denote the spectrum of the m-th residual, the following 2D complex-valued $N \times (K+1)$-array is called the *HB right-hand side* (RHS):

$$L = (L_{m,j})_{1 \le m \le N,\, 0 \le j \le K} \tag{38}$$

The four-dimensional $N \times N \times (K+1) \times (K+1)$-array of complex numbers given by the following equation is called the *HB Jacobian*:

$$J_{m,n;j,k} = \frac{\partial L_m, j}{\partial X_{n,k}} \tag{39}$$

State Variables and Parameters

Each compact model can also have internal state variables, which can be used to hold auxiliary information associated with the current operating point. However, Sentaurus Device does not provide direct support for state variables, and they must be updated by the models themselves during a simulation.

Furthermore, each compact model can have parameters (or attributes) to control its behavior. Usually, these parameters remain constant during a simulation but, occasionally, they can be modified, for example, during a quasistationary command with a goal on a model parameter.

Plotting During Transient Simulations

Special attention is needed for plotting the electrode currents of compact models during transient simulations. When Sentaurus Device plots the current flowing from an electrode into a compact model, it evaluates only the function $f(t, z(t))$ to determine this current. In particular, the function $q(t, z(t))$ is not called. More precisely, Sentaurus Device only calls the function `cmi_instance_get_rhs` with $dc \neq 0$ to evaluate the DC right-hand side (see Functions of CMI Models on page 142).

Therefore, it is recommended that the model equations are set up so that the vector f always contains the currents flowing into the compact model. This requirement is easily satisfied by introducing internal variables that represent the electrode currents. The coupled inductance previously discussed can be used as an example.

NOTE This restriction only affects the value of the electrode currents used for plotting. Sentaurus Device always works with the correct current internally during a transient simulation.

Hierarchical Description of CMI Models

This section describes the hierarchy of compact models.

Device, Parameter Set, and Instance

In Sentaurus Device, each compact model is described by a three-level hierarchy:

Device	The device describes the basic properties of a compact model. This includes the names of the model, electrodes, thermodes, and internal variables, and the names and types of the internal states and parameters.
Parameter set	Each parameter set is derived from a device. It defines default values for the parameters of a compact model. Usually, the parameters that are shared among several instances are defined in a parameter set.
Instance	Instances correspond to the elements in a Sentaurus Device circuit. Each instance is derived from a parameter set. It can override the values of its parameters if necessary.

For each user-defined compact model, the device, parameter set, and instance descriptions must appear in a compact circuit file with the extension `.ccf`. Instances can also appear in the Sentaurus Device `System` section.

Compact Circuit Files (.ccf)

For example, the model of a coupled inductance can be described by the compact circuit file:

```
DEVICE coupled
   ELECTRODES
      u1            // input voltage, high
      u2            // input voltage, low
      u3            // output voltage, high
      u4            // output voltage, low
   INTERNALS
      i1            // input current
      i2            // output current
   STATES
      double m      // mutual inductance
   PARAMETERS
      double ind1   // primary inductance
      double ind2   // secondary inductance
      double k      // coupling factor
END DEVICE

PSET coupled_pset
   DEVICE coupled
   PARAMETERS
      k = 1
END PSET

INSTANCE coupled_inst
   PSET coupled_pset
   ELECTRODES n2 n0 n3 n0
   PARAMETERS
      ind1 = 1e-3
      ind2 = 1e-3
END INSTANCE
```

Syntax of Compact Circuit (.ccf) Files on page 174 presents the complete syntax of this language in Backus–Naur form.

C++ Interface for CMI Models

This section presents the C++ interface used to integrate CMI models into the simulator and provides descriptions of C++ classes representing a device, a parameter set, and an instance of your model.

Data Structure for Device, Parameter Set, and Instance

The class `CCMBaseParam` stores the parameters of devices, parameter sets, and instances.

Table 53 Basic data types that are supported

Data type	C++ name
Character	`char`
Integer	`int`
Real	`double`
String	`char*`

NOTE The class `CCMBaseParam` manages the memory of strings. In particular, the strings are deallocated in the destructor `~CCMBaseParam` by calling `delete[]`.

The class `CCMBaseParam` can hold scalar or vector values of each basic type. The memory for the arrays is allocated and deallocated within the class itself. The method `SetDimension` resizes an array if necessary.

The classes `CCMBaseDevice`, `CCMBasePSet`, and `CCMBaseInstance` provide descriptors for devices, parameter sets, and instances, respectively. The number of electrodes, thermodes, internal variables, states, and parameters is obtained from the model description in the compact circuit file. `CCMBasePSet` provides a pointer to the device from which it was derived. Similarly, the class `CCMBaseInstance` has two pointers to the parent parameter set and the parent device.

The variables `device_desc`, `pset_desc`, and `instance_desc` are provided for use within the compact models. They represent a 'hook' to hold additional information within devices, parameter sets, and instances, which might be required by the compact models.

Header Files

The following header files are available for the implementation of compact models. These files are also in the directory `$STROOT/tcad/$STRELEASE/lib/scm/include`.

CCMBaseDevice.h

```
class CCMBaseDevice {

protected:
```

```
      char* devicename; // name of this device

      const int w_num_electrodes;    // number of electrodes
      const int w_num_thermodes;     // number of thermodes
      const int w_num_internals;     // number of internal variables
      const int w_num_variables;     // total number of variables
                                     // w_num_variables == w_num_electrodes +
                                     // w_num_thermodes +
                                     // w_num_internals
      const int w_num_states;        // number of internal states
      const int w_num_parameters;    // number of parameters

public:

      void* device_desc;             // user device descriptor

      CCMBaseDevice (const char*const name, const int num_electrodes,
                   const int num_thermodes, const int num_internals,
                   const int num_states, const int num_parameters);

      virtual ~CCMBaseDevice ();

      const char* Name () const { return devicename; };
      // return the name of the device

      int NumberElectrodes () const { return w_num_electrodes; };
      // return the number of electrodes

      int NumberThermodes () const { return w_num_thermodes; };
      // return the number of thermodes

      int NumberInternals () const { return w_num_internals; };
      // return the number of internal variables

      int NumberVariables () const { return w_num_variables; };
      // return the total number of variables

      int NumberStates () const { return w_num_states; };
      // return the number of internal states

      int NumberParameters () const { return w_num_parameters; }
      // return the number of parameters
};
```

CCMBaseInstance.h

```
class CCMBaseInstance {

protected:
```

```
      char* instancename;            // name of this instance

      const int w_num_electrodes;    // number of electrodes
      const int w_num_thermodes;     // number of thermodes
      int w_num_internals;           // number of internal variables
      int w_num_variables;           // total number of variables
                                     // w_num_variables == w_num_electrodes +
                                     // w_num_thermodes +
                                     // w_num_internals
      const int w_num_states;        // number of internal states
      const int w_num_parameters;    // number of parameters

      CCMBasePSet* CCMBasepset;         // backpointer to CCMBase parameter set
      CCMBaseDevice* CCMBasedevice;     // backpointer to CCMBase device

public:

      void* instance_desc;              // user instance descriptor

      CCMBaseInstance (const char*const name, CCMBasePSet*const pset);

      virtual ~CCMBaseInstance ();

      const char* Name () const { return instancename; };
      // return the name of the instance

      int NumberElectrodes () const { return w_num_electrodes; };
      // return the number of electrodes

      int NumberThermodes () const { return w_num_thermodes; };
      // return the number of thermodes

      int NumberInternals () const { return w_num_internals; };
      // return the number of internal variables

      int NumberVariables () const { return w_num_variables; };
      // return the total number of variables

      int NumberStates () const { return w_num_states; };
      // return the number of internal states

      int NumberParameters () const { return w_num_parameters; }
      // return the number of parameters

      CCMBasePSet* GetCCMBasePSet () const { return CCMBasepset; };
      // return the parent parameter set

      CCMBaseDevice* GetCCMBaseDevice () const { return CCMBasedevice; };
```

```
    // return the parent device
  };
```

CCMBaseParam.h

```
enum CCMType {
  CCMType_c_char,
  CCMType_c_int,
  CCMType_c_double,
  CCMType_c_string
};

union CCMData {
  // scalars
  char c;
  int i;
  double d;
  char* s;
  // vectors
  char* c_vec;
  int* i_vec;
  double* d_vec;
  char** s_vec;
};

// helper class for CCMBaseParam
class CCMBaseParam_Index_Holder_read {
  friend class CCMBaseParam_Index_Holder_read_write;
  friend class CCMBaseParam;

  const CCMBaseParam& w_param;
  int w_index;
  CCMBaseParam_Index_Holder_read (const CCMBaseParam& parameter, int index);

public:

  // read
  operator char () const;
  operator int () const;
  operator double () const;
  operator const char* () const;
};

// helper class for CCMBaseParam
class CCMBaseParam_Index_Holder_read_write {
  friend class CCMBaseParam_Index_Holder_read;
  friend class CCMBaseParam;

  CCMBaseParam& w_param;
  int w_index;
```

```
    CCMBaseParam_Index_Holder_read_write (CCMBaseParam& parameter, int index);

  public:

    // read
    operator char () const;
    operator int () const;
    operator double () const;
    operator const char* () const;

    // write
    void operator = (const char value);
    void operator = (const int value);
    void operator = (const double value);
    void operator = (const char* value);
    void operator = (const CCMBaseParam& value);
    void operator = (const CCMBaseParam_Index_Holder_read value);
    void operator = (const CCMBaseParam_Index_Holder_read_write value);
  };

  class CCMBaseParam {
    friend class CCMBaseParam_Index_Holder_read;
    friend class CCMBaseParam_Index_Holder_read_write;
    friend class CCMParam;

    void Allocate ();
    // allocate arrays if this is a vector

    void Deallocate ();
    // deallocate arrays if this is a vector

    char* name;           // name of the parameter (is used as key)
    const CCMType type;   // type of the parameter
    const int isvector;   // is this a vector?
    int dimension;        // dimension of vector, if isvector
    int defined;          // is the value of this parameter defined?
    CCMData w_value;      // value of parameter, if defined

  public:

    CCMBaseParam (const char*const paramname, const CCMType paramtype,
                  const int paramisvector, const int paramdimension);

    CCMBaseParam (const CCMBaseParam& param);

    virtual ~CCMBaseParam ();

    const char* Name () const
    { return name; }
```

```
  CCMType Type () const
  { return type; }

  int IsVector () const
  { return isvector; }

  int Dimension () const
  { return dimension; }

  void SetDimension (const int dim);
  // resize the vector
  // the old values are lost

  int Defined () const
  { return defined; }

  void Defined (const int def)
  { defined = def; }

  // conversion operators
  operator char () const;
  operator int () const;
  operator double () const;
  operator const char* () const;

  // assignment operators
  void operator = (const char value);
  void operator = (const int value);
  void operator = (const double value);
  void operator = (const char* value);
  void operator = (const CCMBaseParam& value);
  void operator = (const CCMBaseParam_Index_Holder_read value);
  void operator = (const CCMBaseParam_Index_Holder_read_write value);

  // array access
  const CCMBaseParam_Index_Holder_read operator [] (const int index) const;
  CCMBaseParam_Index_Holder_read_write operator [] (const int index);

  virtual void Print (std::ostream& stream) const;
  // print the parameter
};

std::ostream& operator << (std::ostream& stream, const CCMBaseParam* par);
```

CCMBasePSet.h

```
class CCMBasePSet {

protected:

      char* psetname;                // name of this parameter set

      const int w_num_electrodes;    // number of electrodes
      const int w_num_thermodes;     // number of thermodes
      const int w_num_internals;     // number of internal variables
      const int w_num_variables;     // total number of variables
                                     // w_num_variables == w_num_electrodes +
                                     // w_num_thermodes +
                                     // w_num_internals
      const int w_num_states;        // number of internal states
      const int w_num_parameters;    // number of parameters

      CCMBaseDevice* CCMBasedevice;  // backpointer to CCMBase device

public:

      void* pset_desc;               // user parameter set descriptor

      CCMBasePSet (const char*const name, CCMBaseDevice*const device);

      virtual ~CCMBasePSet ();

      const char* Name () const { return psetname; };
      // return the name of the parameter set

      int NumberElectrodes () const { return w_num_electrodes; };
      // return the number of electrodes

      int NumberThermodes () const { return w_num_thermodes; };
      // return the number of thermodes

      int NumberInternals () const { return w_num_internals; };
      // return the number of internal variables

      int NumberVariables () const { return w_num_variables; };
      // return the total number of variables

      int NumberStates () const { return w_num_states; };
      // return the number of internal states

      int NumberParameters () const { return w_num_parameters; }
      // return the number of parameters
```

```
        CCMBaseDevice* GetCCMBaseDevice () const { return CCMBasedevice; };
        // return the parent device
};
```

Compilation of C++ (.C) Files

Sentaurus Device tries to load the code of a compact model from an external shared object file. The file name of the shared object file must be identical to the name of the device, and the file extension must be `.so.`*`arch`*, where *`arch`* depends on the architecture listed in Table 54.

Table 54 Shared object file extensions

Architecture	Extension
AMD 64-bit	`linux64`

Special compiler flags must be used to produce a shared object file with position-independent code. The script `cmi` manages all the architecture-dependent details and is invoked by:

```
cmi options files
```

Table 55 Options of the cmi script

Option	Purpose
`-a`	Prints the version of the C++ compiler invoked by Sentaurus `cmi`.
`-c`	Suppresses linking and produce an `.o` file for each source file.
`--compiler-version`	Prints the version of the C++ compiler used by Synopsys to compile TCAD Sentaurus tools.
`-D`*`name`*`=`*`def`*	Defines a macro symbol to the preprocessor. The assignment `=`*`def`* is optional.
`-g`	Prepares the object files for debugging.
`-h`	Shows a help message.
`-I`*`pathname`*	Appends *`pathname`* to the list of directories that are searched to look for `#include` files.
`-llib`	Specifies an additional library for linking with object files.
`-L`*`pathname`*	Appends *`pathname`* to the list of directories that are searched by the linker.
`-O`	Produces optimized code.
`--openmp`	Recognizes OpenMP directives.
`-U`*`name`*	Removes any initial definition of a macro symbol.
`-v`	Displays the compiler command.

Files ending in `.C`, `.cc`, or `.cxx` are assumed to be C++ source files. Files with the extension `.o` are passed directly to the linker.

NOTE The version of the C++ compiler used for the CMI models must be identical to the version of the C++ compiler used by Synopsys to compile Sentaurus Device. Use the command `cmi -a` to verify the compiler versions.

The UNIX command `strings` can be invoked to determine the version of the header files used during the compilation of a CMI model:

```
strings -a <shared object file> | fgrep 'Sentaurus TCAD version'
```

This command results in output such as the following:

```
Synopsys Sentaurus TCAD version N-2017.09
```

Functions of CMI Models

Each user-defined compact model must implement a set of well-defined CMI functions. These functions support the creation, deletion, and initialization of devices, parameter sets, and instances. Furthermore, they support the numeric solution of compact device instances for the different analysis methods of Sentaurus Device. These functions are described in the following sections.

For reference, the header file `CMIModels.h` is provided at the end of this section, which contains the C++ declarations of these CMI functions.

NOTE All functions must conform to C linkage conventions. The prototypes of the CMI functions appear in the header file `CMIModels.h`.

cmi_device_create

```
void
   cmi_device_create (CCMBaseDevice*const device);
```

This function is called whenever a new device is created. It initializes the `device_desc` variable.

cmi_device_set_param

```
void
   cmi_device_set_param (CCMBaseDevice*const device,
                         const CCMBaseParam*const param);
```

This function assigns a new value to a device parameter. It can be called several times before `cmi_device_initialize` is invoked.

cmi_device_initialize

```
void
   cmi_device_initialize (CCMBaseDevice*const device);
```

This function is always invoked after the values of one or more parameters have changed. It can compute auxiliary variables that depend on the parameters.

cmi_device_get_param

```
void
   cmi_device_get_param (CCMBaseDevice*const device,
                         CCMBaseParam*const param);
```

This function retrieves the value of a parameter. The parameter is identified by its name, that is, by the value of `param->Name()`. The result should be stored in `*param`.

cmi_device_delete

```
void
   cmi_device_delete (CCMBaseDevice*const device);
```

This function is called whenever a device is deleted. It presents an opportunity to release the dynamic memory associated with the device.

cmi_pset_create

```
void
   cmi_pset_create (CCMBasePSet*const pset);
```

This function is called whenever a new parameter set is created. It initializes the `pset_desc` variable.

cmi_pset_set_param

```
void
   cmi_pset_set_param (CCMBasePSet*const pset,
                       const CCMBaseParam*const param);
```

This function assigns a new value to a parameter-set parameter. It can be called several times before `cmi_pset_initialize` is invoked.

cmi_pset_initialize

```
void
   cmi_pset_initialize (CCMBasePSet*const pset);
```

This function is always invoked after the values of one or more parameters have changed. It can compute auxiliary variables that depend on the parameters.

cmi_pset_get_param

```
void
   cmi_pset_get_param (CCMBasePSet*const pset,
                       CCMBaseParam*const param);
```

This function retrieves the value of a parameter. The parameter is identified by its name, that is, by the value of `param->Name()`. The result should be stored in `*param`.

cmi_pset_delete

```
void
   cmi_pset_delete (CCMBasePSet*const pset);
```

This function is called whenever a parameter set is deleted. It presents an opportunity to release the dynamic memory associated with the parameter set.

cmi_instance_create

```
void
  cmi_instance_create (CCMBaseInstance*const instance);
```

This function is called whenever a new instance is created. It initializes the `instance_desc` variable.

cmi_instance_set_param

```
void
  cmi_instance_set_param (CCMBaseInstance*const instance,
                          const CCMBaseParam*const param);
```

This function assigns a new value to an instance parameter. It can be called several times before `cmi_instance_initialize` is invoked.

cmi_instance_initialize

```
void
  cmi_instance_initialize (CCMBaseInstance*const instance);
```

This function is always invoked after the values of one or more parameters have changed. It can compute auxiliary variables that depend on the parameters.

cmi_instance_get_param

```
void
  cmi_instance_get_param (CCMBaseInstance*const instance,
                          CCMBaseParam*const param);
```

This function retrieves the value of a parameter. The parameter is identified by its name, that is, by the value of `param->Name()`. The result should be stored in `*param`.

cmi_instance_get_rhs

```
void
   cmi_instance_get_rhs (CCMBaseInstance*const instance,
                         const int dc, const double time,
                         const double*const variables,
                         double*const rhs);
```

This function computes either the DC right-hand side f (if dc $\neq 0$) or the transient right-hand side q (if dc $= 0$).

The simulation time is passed in the parameter `time`. The array `variables` contains the values of the system variables, that is, electrode voltages, thermode temperatures, and internal variables. The length of `variables` is given by `instance->NumberVariables()`. The result must be stored in `rhs`, which is also an array of size `instance->NumberVariables()`.

cmi_instance_get_jacobian

```
void
   cmi_instance_get_jacobian (CCMBaseInstance*const instance,
                              const int dc, const double time,
                              const double*const variables,
                              double*const*const jacobian);
```

This function computes either the DC Jacobian J_f (if dc $\neq 0$) or the transient Jacobian J_q (if dc $= 0$).

The simulation time is passed in the parameter `time`. The array `variables` contains the values of the system variables, that is, electrode voltages, thermode temperatures, and internal variables. The length of `variables` is given by `instance->NumberVariables()`. The result must be stored in `jacobian`, which is a square matrix of dimension `instance->NumberVariables()`.

cmi_instance_is_physical

```
int
   cmi_instance_is_physical (CCMBaseInstance*const instance,
                             const double time,
                             const double*const variables);
```

This function determines whether the given operating point lies within the valid model range. The parameter `time` represents the simulation time, and the system variables are passed in the parameter `variables`, an array of size `instance->NumberVariables()`.

This function is called whenever a simulation step has converged. If the computed solution is acceptable, `cmi_instance_is_physical` returns `1`. Otherwise, the value `0` is returned.

A model can also control the time step during a transient simulation by calling the functions described in Runtime Support on page 150.

cmi_instance_delete

```
void
   cmi_instance_delete (CCMBaseInstance*const instance);
```

This function is called whenever an instance is deleted. It presents an opportunity to release the dynamic memory associated with the instance.

cmi_instance_get_hb_rhs

```
void cmi_instance_get_hb_rhs (
      CCMBaseInstance*const instance,
      int nb_spectrum_indices,
      const CMI_MODEL_complex_type*const*const hb_variables,
      CMI_MODEL_complex_type*const*const rhs);
```

This function computes the HB RHS. It is part of the optional function set CMI-HB-MDFT.

The parameter `nb_spectrum_indices` gives the number of components in the spectrum. The parameter `hb_variables` gives the actual values of the variables in the frequency domain. It is a 2D array of size `instance->NumberVariables()` and `nb_spectrum_indices`.

The result must be stored in `rhs`, which is also a 2D array with the same sizes as `hb_variables`.

cmi_instance_get_hb_jacobian

```
void cmi_instance_get_hb_jacobian (
      CCMBaseInstance*const instance,
      int nb_spectrum_indices,
      const CMI_MODEL_complex_type*const*const hb_variables,
      CMI_MODEL_complex_type*const*const*const*const jacobian);
```

This function computes the HB Jacobian. It is part of the optional function set CMI-HB-MDFT.

The parameters `nb_spectrum_indices` and `hb_variables` are the same as in the function `cmi_instance_get_hb_rhs`. The result must be stored in `jacobian`, which is a four-dimensional array of sizes `instance->NumberVariables()`, `instance->NumberVariables()`, `nb_spectrum_indices`, and `nb_spectrum_indices`.

CMIModels.h

```
extern "C" {

// subroutines for devices
void
cmi_device_create (CCMBaseDevice*const device);

void
cmi_device_set_param (CCMBaseDevice*const device,
                      const CCMBaseParam*const param);

void
cmi_device_initialize (CCMBaseDevice*const device);

void
cmi_device_get_param (CCMBaseDevice*const device,
                      CCMBaseParam*const param);

void
cmi_device_delete (CCMBaseDevice*const device);

// subroutines for parameter sets
void
cmi_pset_create (CCMBasePSet*const pset);

void
cmi_pset_set_param (CCMBasePSet*const pset,
                    const CCMBaseParam*const param);

void
cmi_pset_initialize (CCMBasePSet*const pset);

void
```

```
cmi_pset_get_param (CCMBasePSet*const pset,
                    CCMBaseParam*const param);

void
cmi_pset_delete (CCMBasePSet*const pset);

// subroutines for instances
void
cmi_instance_create (CCMBaseInstance*const instance);

void
cmi_instance_set_param (CCMBaseInstance*const instance,
                        const CCMBaseParam*const param);

void
cmi_instance_initialize (CCMBaseInstance*const instance);

void
cmi_instance_get_param (CCMBaseInstance*const instance,
                        CCMBaseParam*const param);

void
cmi_instance_get_rhs (CCMBaseInstance*const instance,
                      const int dc, const double time,
                      const double*const variables,
                      double*const rhs);

void
cmi_instance_get_jacobian (CCMBaseInstance*const instance,
                           const int dc, const double time,
                           const double*const variables,
                           double*const*const jacobian);

void
cmi_instance_get_hb_rhs (CCMBaseInstance*const instance,
                      int nb_spectrum_indices,
                      const CMI_MODEL_complex_type*const*const hb_variables,
                      CMI_MODEL_complex_type*const*const rhs);

void
cmi_instance_get_hb_jacobian (CCMBaseInstance*const instance,
                   int nb_spectrum_indices,
                   const CMI_MODEL_complex_type*const*const hb_variables,
                   CMI_MODEL_complex_type*const*const*const*const jacobian);

int
cmi_instance_is_physical (CCMBaseInstance*const instance,
                          const double time,
                          const double*const variables);
```

```
void
cmi_instance_delete (CCMBaseInstance*const instance);

} // extern "C"
```

Runtime Support

Sentaurus Device provides functions to support the operation of compact models. The functions control the progress of an ongoing simulation.

cmi_starttime

```
double
   cmi_starttime ();
```

This function returns the start time of a transient simulation.

cmi_stoptime

```
double
   cmi_stoptime ();
```

This function returns the stop time of a transient simulation.

cmi_min_timestep

```
double
   cmi_min_timestep ();
```

This function returns the minimum step size of Sentaurus Device during a transient simulation.

cmi_max_timestep

```
double
   cmi_max_timestep ();
```

This function returns the maximum step size of Sentaurus Device during a transient simulation.

cmi_set_event

```
void
   cmi_set_event (const double time);
```

This function informs Sentaurus Device to synchronize with a given point in time during a transient simulation.

cmi_set_max_timestep

```
void
   cmi_set_max_timestep (const double stepsize);
```

This function limits the step size of Sentaurus Device during a transient simulation.

cmi_hb_spectrum_nb_basefrequencies

```
int cmi_hb_spectrum_nb_basefrequencies();
```

This function returns the number of base frequencies (‘tones’) of the spectrum in use.

cmi_hb_spectrum_index_frequency

```
double cmi_hb_spectrum_index_frequency ( int index );
```

This function returns the frequency associated with the spectrum component `index`.

cmi_hb_spectrum_index_circfrequency

```
double cmi_hb_spectrum_index_circfrequency ( int index );
```

This function returns the circular frequency associated with the spectrum component `index`.

cmi_hb_spectrum_index_multiindex

```
void cmi_hb_spectrum_index_multiindex (
   int index, int* multiindex );
```

This function returns the frequency of the spectrum component `index` (as the function `cmi_hb_spectrum_index_frequency`) and provides multi-index information for the spectrum `index`.

The function `cmi_hb_spectrum_nb_basefrequencies` must be used beforehand to allocate the array `multiindex` in the suitable size.

cmi_hb_spectrum_parameters

```
void cmi_hb_spectrum_parameters (
   double* basefrequencies, int* nb_harmonics );
```

This function returns the base frequencies and the maximum number of harmonics of the spectrum in use in the output parameters `basefrequencies` and `nb_harmonics`, respectively.

The function `cmi_hb_spectrum_nb_basefrequencies` must be used beforehand to allocate the output arrays in the suitable size.

CMISupport.h

```
double
cmi_starttime ();
// Returns the starttime of a transient simulation.

double
cmi_stoptime ();
// Returns the stoptime of a transient simulation.

double
cmi_min_timestep ();
// Returns the minimum stepsize of Sentaurus Device
// during a transient simulation.

double
cmi_max_timestep ();
// Returns the maximum stepsize of Sentaurus Device
// during a transient simulation.

void
```

```
cmi_set_event (const double time);
// This subroutine tells Sentaurus Device to synchronize with
// a given point in time during a transient simulation.

void
cmi_set_max_timestep (const double stepsize);
// This subroutine limits the stepsize of Sentaurus Device
// during a transient simulation.

int
cmi_hb_spectrum_nb_basefrequencies();
// Returns the number of base frequencies ('tones')
// of the spectrum in use.

double
cmi_hb_spectrum_index_frequency (int index);
// Returns the frequency associated with the spectrum
// component 'index' of an hb vector (e.g. hb_rhs[index]).

double
cmi_hb_spectrum_index_circfrequency (int index);
// Returns the circular frequency associated with the spectrum
// component 'index' of an hb vector (e.g. hb_rhs[index]).

void
cmi_hb_spectrum_index_multiindex (int index, int* multiindex);
// Returns the frequency of the spectrum component 'index'
// (as the function 'cmi_hb_spectrum_index_frequency')
// and provides multi index information for the spectrum index.
// The function 'cmi_hb_spectrum_nb_basefrequencies' has to be used before
// to allocate the arrays in suitable size.

void
cmi_hb_spectrum_parameters (double* basefrequencies, int* nb_harmonics);
// Returns the base frequencies and maximal number of harmonics
// of the spectrum:
//   basefrequencies[it]: basefrequency of tone 'it'
//   nb_harmonics[it]   : number of harmonics in spectrum of tone 'it'
// The function 'cmi_hb_spectrum_nb_basefrequencies' has to be used before
// to allocate the arrays in suitable size.
```

Command File of Sentaurus Device

To load external compact models into Sentaurus Device, the search path `CMIPath` must be defined in the `File` section of the command file of Sentaurus Device. The value of `CMIPath` consists of a sequence of directories, for example:

```
File {
   CMIPath = ". /home/paper/lib /home/pencil/sdevice/lib"
```

```
}
```

The given directories are searched for compact circuit files (extension `.ccf`) and the corresponding shared object files (extension `.so.arch`). Instances of compact models can also appear in the `System` section of the command file.

In this case, use the correct syntax, for example:

```
System {
   Electrical (n0 n1 n2 n3)
   coupled_pset coupled_inst (n2 n0 n3 n0) {ind1 = 1e-3 ind2 = 1e-3}
}
```

Electrothermal Models

In Sentaurus Device, circuit nodes are electrical by default. This means that their values are assumed to be voltages. However nodes can also be declared to be thermal, in which case, their values correspond to temperatures [K]. Electrical nodes are part of the electrical circuit (`Circuit`), and thermal nodes belong to the thermal circuit (`TCircuit`).

Sentaurus Device does not require that the electrodes of a compact model are connected to electrical circuit nodes or that thermodes are connected to thermal circuit nodes. Such a restriction is undesirable because several devices can be used both in electrical and thermal circuits. An example is a resistor that can be used as an electrical resistor in `Circuit` and as a thermal resistor in `TCircuit`. Similarly, voltage sources can serve as heat sources or heat sinks in the thermal circuit.

The equations in `Circuit` and `TCircuit` can be solved independently or simultaneously, depending on the commands in the `Solve` section:

```
coupled{Circuit}
coupled{TCircuit}
coupled{Circuit TCircuit}
```

This separation of equations is useful for certain types of simulation. It might be required to determine the DC operating point of a circuit by solving `Circuit` and `TCircuit` together. For a subsequent transient analysis only, the `Circuit` equations must be solved. Of course, it is assumed that the temperatures do not change during the transient simulation (which is a reasonable assumption for many problems).

For electrothermal models, Sentaurus Device imposes a restriction. The `Circuit` and `TCircuit` equations can be solved simultaneously, but the Jacobian of the equations does not include the cross-terms or couplings between the electrical and thermal equations. This restriction only affects models that are connected to both electrical and thermal nodes. The results of the simulation will not change. However, the speed of convergence can be impaired.

If problems are encountered due to this restriction (slow convergence or nonconvergence), a simple solution is to let all of the nodes in the `System` section be electrical, that is, do not declare any thermal nodes.

The temperature equations will then also become a part of `Circuit`, and the thermal equations will be solved simultaneously with the electrical equations.

NOTE This solution applies only to compact models. If a simulation involves physical devices with electrodes and thermodes, connect the electrodes to electrical nodes and the thermodes to thermal nodes.

Summary

To add external compact models to Sentaurus Device:

1. Collect the equations as in Eq. 22, p. 126 for the compact model. Define the model parameters and their types. Define the electrodes, thermodes, and internal variables that are needed:
   - For each electrode, compute the current flowing into the device.
   - For each thermode, compute the heat flow into the device.
   - For each internal variable, provide a corresponding model equation.
2. Write a compact circuit file (with extension `.ccf`). Assign default values to the parameters (this is usually performed on the level of a device or parameter set).
3. Implement all of the CMI functions (see Functions of CMI Models on page 142).
4. Use the `cmi` script to compile the model code and to produce a shared object file with the extension `.so.`*`arch`*. Each shared object file must correspond exactly to one compact model, and the file name must be identical to the device name of the compact model.
5. Define the variable `CMIPath` in the `File` section of the Sentaurus Device command file. This defines the search path for all `.ccf` and `.so.`*`arch`* files.
6. The instances of the compact model are usually defined in the `.ccf` files. However, they can also appear in the `System` section of the Sentaurus Device command file. In this case, use the correct syntax of the Sentaurus Device command file.
7. Run Sentaurus Device to perform the simulation.

Example: Implementing Coupled Inductances

This section shows the implementation of coupled inductances as a compact model.

NOTE Coupled inductances are also available as SPICE models (see Inductor on page 5 and Coupled (Mutual) Inductors on page 6).

The source files for this example are in the directory:

```
$STROOT/tcad/$STRELEASE/lib/sdevice/src/dynamic
```

Model Equations

See Example: Coupled Inductance on page 129.

coupled.ccf

```
DEVICE coupled
   ELECTRODES
      u1            // input voltage, high
      u2            // input voltage, low
      u3            // output voltage, high
      u4            // output voltage, low
   INTERNALS
      i1            // input current
      i2            // output current
   STATES
      double m      // mutual inductance
   PARAMETERS
      double ind1   // primary inductance
      double ind2   // secondary inductance
      double k      // coupling factor
END DEVICE

PSET coupled_pset
   DEVICE coupled
   PARAMETERS
      k = 1
END PSET
```

coupled.C

```
#include <stdlib.h>
#include <string.h>
#include <iostream.h>
#include <math.h>

#include "CMIModels.h"
#include "CMISupport.h"

class Parameters {
public:
   double ind1;
   double ind2;
   double k;

   Parameters ();
   void Set (const CCMBaseParam* param);
   void Get (CCMBaseParam* param);
};

Parameters::
Parameters () :
   ind1 (0.0),
   ind2 (0.0),
   k (1.0)
{
}

void Parameters::
Set (const CCMBaseParam* param)
{ if (param->Defined ()) {
    if (strcmp (param->Name (), "ind1") == 0) {
      ind1 = *param;
    } else if (strcmp (param->Name (), "ind2") == 0) {
      ind2 = *param;
    } else if (strcmp (param->Name (), "k") == 0) {
      k = *param;
    }
  }
}

void Parameters::
Get (CCMBaseParam* param)
{ if (strcmp (param->Name (), "ind1") == 0) {
    *param = ind1;
    param->Defined (1);
  } else if (strcmp (param->Name (), "ind2") == 0) {
    *param = ind2;
    param->Defined (1);
```

```
  } else if (strcmp (param->Name (), "k") == 0) {
    *param = k;
    param->Defined (1);
  } else {
    param->Defined (0);
  }
}

class Device {
public:
   Parameters par;
};

class PSet {
public:
   Parameters par;
};

class Instance {
public:
   Parameters par;
   double m;
};

extern "C" void
cmi_device_create (CCMBaseDevice*const device)
{ device->device_desc = new Device;
}

extern "C" void
cmi_device_set_param (CCMBaseDevice*const device,
                      const CCMBaseParam*const param)
{ Device* dev = (Device*) device->device_desc;
  dev->par.Set (param);
}

extern "C" void
cmi_device_initialize (CCMBaseDevice*const device)
{
}

extern "C" void
cmi_device_get_param (CCMBaseDevice*const device,
                      CCMBaseParam*const param)
{ Device* dev = (Device*) device->device_desc;
  dev->par.Get (param);
}

extern "C" void
cmi_device_delete (CCMBaseDevice*const device)
{ delete (Device*) device->device_desc;
}
```

```
extern "C" void
cmi_pset_create (CCMBasePSet*const pset)
{ pset->pset_desc = new PSet;
}

extern "C" void
cmi_pset_set_param (CCMBasePSet*const pset,
                    const CCMBaseParam*const param)
{ PSet* ps = (PSet*) pset->pset_desc;
  ps->par.Set (param);
}

extern "C" void
cmi_pset_initialize (CCMBasePSet*const pset)
{
}

extern "C" void
cmi_pset_get_param (CCMBasePSet*const pset,
                    CCMBaseParam*const param)
{ PSet* ps = (PSet*) pset->pset_desc;
  ps->par.Get (param);
}

extern "C" void
cmi_pset_delete (CCMBasePSet*const pset)
{ delete (PSet*) pset->pset_desc;
}

extern "C" void
cmi_instance_create (CCMBaseInstance*const instance)
{ instance->instance_desc = new Instance;
}

extern "C" void
cmi_instance_set_param (CCMBaseInstance*const instance,
                        const CCMBaseParam*const param)
{ Instance* inst = (Instance*) instance->instance_desc;
  inst->par.Set (param);
}

extern "C" void
cmi_instance_initialize (CCMBaseInstance*const instance)
{ Instance* inst = (Instance*) instance->instance_desc;
  inst->m = inst->par.k * sqrt (inst->par.ind1 * inst->par.ind2);
}

extern "C" void
cmi_instance_get_param (CCMBaseInstance*const instance,
                        CCMBaseParam*const param)
{ Instance* inst = (Instance*) instance->instance_desc;
  inst->par.Get (param);
```

```
}

#define u1 (variables [0])
#define u2 (variables [1])
#define u3 (variables [2])
#define u4 (variables [3])
#define i1 (variables [4])
#define i2 (variables [5])

extern "C" void
cmi_instance_get_rhs (CCMBaseInstance*const instance,
                      const int dc, const double time,
                      const double*const variables,
                      double*const rhs)
{ Instance* inst = (Instance*) instance->instance_desc;
  if (dc) {
    rhs [0] = i1;
    rhs [1] = -i1;
    rhs [2] = i2;
    rhs [3] = -i2;
    rhs [4] = u1-u2;
    rhs [5] = u3-u4;
  } else {
    rhs [4] = -inst->par.ind1 * i1 - inst->m * i2;
    rhs [5] = -inst->par.ind2 * i2 - inst->m * i1;
  }
}

extern "C" void
cmi_instance_get_jacobian (CCMBaseInstance*const instance,
                           const int dc, const double time,
                           const double*const variables,
                           double*const*const jacobian)
{ Instance* inst = (Instance*) instance->instance_desc;
  if (dc) {
    jacobian[0][4] = 1.0;
    jacobian[1][4] = -1.0;
    jacobian[2][5] = 1.0;
    jacobian[3][5] = -1.0;
    jacobian[4][0] = 1.0;
    jacobian[4][1] = -1.0;
    jacobian[5][2] = 1.0;
    jacobian[5][3] = -1.0;
  } else {
    jacobian[4][4] = -inst->par.ind1;
    jacobian[4][5] = -inst->m;
    jacobian[5][4] = -inst->m;
    jacobian[5][5] = -inst->par.ind2;
  }
}
```

```
extern "C" int
cmi_instance_is_physical (CCMBaseInstance*const instance,
                          const double time,
                          const double*const variables)
{ return 1;
}

extern "C" void
cmi_instance_delete (CCMBaseInstance*const instance)
{ delete (Instance*) instance->instance_desc;
}
```

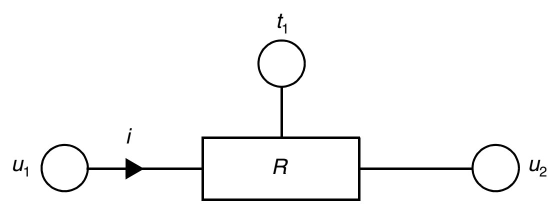
Example: Implementing the Electrothermal Resistor Model

This section presents the equations for the electrothermal resistor model.

NOTE This model is also available as a built-in model in Sentaurus Device (see Electrothermal Resistor (Ter) Model on page 118).

Model Equations

An electrothermal resistance has three contacts (two electrodes and one thermode):



The electrical behavior of the resistance is described by Ohm's law:

$$u_1 - u_2 = R \cdot i \tag{40}$$

The resistance of the device, however, depends on the thermode temperature:

$$R = r \cdot (1 + \alpha \cdot (t_1 - t_{ref}) + \beta \cdot (t_1 - t_{ref})^2) \tag{41}$$

The device also produces Joule heat, which is dissipated through the thermode:

$$P = (u_1 - u_2) \cdot i \tag{42}$$

The device can be described by the following vector of unknowns:

$$z(t) = \begin{bmatrix} u_1 \\ u_2 \\ t_1 \\ i \end{bmatrix} \tag{43}$$

The DC right-hand side is given by:

$$f_R(t, z(t)) = \begin{bmatrix} i \\ -i \\ -(u_1 - u_2) \cdot i \\ u_1 - u_2 - R \cdot i \end{bmatrix} \tag{44}$$

The first two entries of f_R correspond to the current flowing into the device through the electrodes u_1 and u_2. The third entry of f_R represents the heat flow into the device through the thermode t_1.

Eq. 40 and Eq. 42 have no derivatives with regard to time. Therefore, the transient right-hand side is zero. The Jacobian of f_R is given by:

$$\frac{d}{dz} f_R(t, z(t)) = J_{f_R} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & -1 \\ -i & i & 0 & -(u_1 - u_2) \\ 1 & -1 & -\dfrac{dR}{dt_1} \cdot i & -R \end{bmatrix} \tag{45}$$

where:

$$\frac{dR}{dt_1} = r \cdot (\alpha + 2 \cdot \beta \cdot (t_1 - t_{ref})) \tag{46}$$

tres.ccf

```
DEVICE tres
   ELECTRODES
      u1                      // input voltage 1
      u2                      // input voltage 2
   THERMODES
      t1                      // thermode for Joule heat
   INTERNALS
      i                       // current through resistor
   PARAMETERS
```

```
      double r = 1          // resistance
      double alpha = 0      // linear temperature coefficient
      double beta = 0       // quadratic temperature coefficient
      double tref = 300.15  // reference temperature
END DEVICE

PSET tres_pset
   DEVICE tres
END PSET
```

tres.C

```
#include <stdlib.h>
#include <string.h>
#include <iostream.h>
#include <math.h>
#include "CMIModels.h"
#include "CMISupport.h"

class Parameters {
public:
   double r;        // resistance
   double alpha;    // linear temperature coefficient
   double beta;     // quadratic temperature coefficient
   double tref;     // reference temperature

   Parameters ();
   void Set (const CCMBaseParam* param);
   void Get (CCMBaseParam* param);
};

Parameters::
Parameters () :
   r (1.0),
   alpha (0.0),
   beta (0.0),
   tref (300.15)
{
}

void Parameters::
Set (const CCMBaseParam* param)
{ if (param->Defined ()) {
    if (strcmp (param->Name (), "r") == 0) {
      r = *param;
    } else if (strcmp (param->Name (), "alpha") == 0) {
      alpha = *param;
    } else if (strcmp (param->Name (), "beta") == 0) {
      beta = *param;
```

```
    } else if (strcmp (param->Name (), "tref") == 0) {
      tref = *param;
    }
  }
}

void Parameters::
Get (CCMBaseParam* param)
{ if (strcmp (param->Name (), "r") == 0) {
    *param = r;
    param->Defined (1);
  } else if (strcmp (param->Name (), "alpha") == 0) {
    *param = alpha;
    param->Defined (1);
  } else if (strcmp (param->Name (), "beta") == 0) {
    *param = beta;
    param->Defined (1);
  } else if (strcmp (param->Name (), "tref") == 0) {
    *param = tref;
    param->Defined (1);
  } else {
    param->Defined (0);
  }
}
class Device {
public:
   Parameters par;
};
class PSet {
public:
   Parameters par;
};

class Instance {
public:
   Parameters par;
};

extern "C" void
cmi_device_create (CCMBaseDevice*const device)
{ device->device_desc = new Device;
}

extern "C" void
cmi_device_set_param (CCMBaseDevice*const device,
                      const CCMBaseParam*const param)
{ Device* dev = (Device*) device->device_desc;
  dev->par.Set (param);
}
```

```
extern "C" void
cmi_device_initialize (CCMBaseDevice*const device)
{
}

extern "C" void
cmi_device_get_param (CCMBaseDevice*const device,
                      CCMBaseParam*const param)
{ Device* dev = (Device*) device->device_desc;
  dev->par.Get (param);
}

extern "C" void
cmi_device_delete (CCMBaseDevice*const device)
{ delete (Device*) device->device_desc;
}

extern "C" void
cmi_pset_create (CCMBasePSet*const pset)
{ pset->pset_desc = new PSet;
}

extern "C" void
cmi_pset_set_param (CCMBasePSet*const pset,
                    const CCMBaseParam*const param)
{ PSet* ps = (PSet*) pset->pset_desc;
  ps->par.Set (param);
}

extern "C" void
cmi_pset_initialize (CCMBasePSet*const pset)
{
}

extern "C" void
cmi_pset_get_param (CCMBasePSet*const pset,
                    CCMBaseParam*const param)

{ PSet* ps = (PSet*) pset->pset_desc;
  ps->par.Get (param);
}

extern "C" void
cmi_pset_delete (CCMBasePSet*const pset)
{ delete (PSet*) pset->pset_desc;
}

extern "C" void
cmi_instance_create (CCMBaseInstance*const instance)
{ instance->instance_desc = new Instance;
}
```

```
extern "C" void
cmi_instance_set_param (CCMBaseInstance*const instance,
                        const CCMBaseParam*const param)
{ Instance* inst = (Instance*) instance->instance_desc;
  inst->par.Set (param);
}

extern "C" void
cmi_instance_initialize (CCMBaseInstance*const instance)
{ Instance* inst = (Instance*) instance->instance_desc;
}

extern "C" void
cmi_instance_get_param (CCMBaseInstance*const instance,
                        CCMBaseParam*const param)
{ Instance* inst = (Instance*) instance->instance_desc;
  inst->par.Get (param);
}

#define u1 (variables [0])
#define u2 (variables [1])
#define t1 (variables [2])
#define i  (variables [3])

extern "C" void
cmi_instance_get_rhs (CCMBaseInstance*const instance,
                      const int dc, const double time,
                      const double*const variables,
                      double*const rhs)
{ Instance* inst = (Instance*) instance->instance_desc;
  if (dc) {
    double dT = t1 - inst->par.tref;
    double R = inst->par.r *
               (1.0 + (inst->par.alpha + inst->par.beta * dT) * dT);
    rhs [0] = i;
    rhs [1] = -i;
    rhs [2] = -(u1 - u2) * i;
    rhs [3] = u1 - u2 - R * i;
  } else {
    rhs [0] = 0.0;
    rhs [1] = 0.0;
    rhs [2] = 0.0;
    rhs [3] = 0.0;
  }
}

extern "C" void
cmi_instance_get_jacobian (CCMBaseInstance*const instance,
                           const int dc, const double time,
                           const double*const variables,
```

```
                              double*const*const jacobian)
{ Instance* inst = (Instance*) instance->instance_desc;
  if (dc) {
    double dT = t1 - inst->par.tref;
    double R = inst->par.r *
               (1.0 + (inst->par.alpha + inst->par.beta * dT) * dT);
    double dRdt1 = inst->par.r *
                   (inst->par.alpha + 2.0 * inst->par.beta * dT);
    jacobian[0][3] = 1.0;
    jacobian[1][3] = -1.0;
    jacobian[2][0] = -i;
    jacobian[2][1] = i;
    jacobian[2][3] = u2 - u1;
    jacobian[3][0] = 1.0;
    jacobian[3][1] = -1.0;
    jacobian[3][2] = -i*dRdt1;
    jacobian[3][3] = -R;
  }
}

extern "C" int
cmi_instance_is_physical (CCMBaseInstance*const instance,
                          const double time,
                          const double*const variables)
{ return 1;
}

extern "C" void
cmi_instance_delete (CCMBaseInstance*const instance)
{ delete (Instance*) instance->instance_desc;
}
```

CMI Models With Frequency-Domain Assembly

This section describes the provided CMI models suitable for the HB-MDFT mode.

Notation: In general, time-domain (TD) quantities are denoted by lowercase letters, while the corresponding frequency-domain (FD) quantity is denoted by uppercase letters. See Frequency-Domain Model Equations on page 130 for the Fourier representation.

Throughout this section, the following notation is used:

- u_i is the TD voltage at electrode n_i.
- $u_{ij}:=u_i-u_j$ are the voltage differences.
- i_k is the current from electrode n_k into the device.

Admittance sd_hb_pGC

The device `sd_hb_pGC` describes the behavior of a (real-valued) conductance G in parallel with a (real-valued) capacitance C.

Device name:	`sd_hb_pGC`
Device parameter set name:	`sd_hb_pGC_pset`
Electrodes:	Two electrodes n_1 and n_2
Internal variables:	None

Table 56 Device parameters

Symbol	Description	Range	Unit	Default
G	Conductance	$(-\infty, \infty)$	S	0.
C	Capacitance	$(-\infty, \infty)$	F	0.

Transient Equation:

$$i_1(t) - [Gu_{12}(t) + C\dot{u}_{12}] = 0 \tag{47}$$

DC Equation:

$$i_1 - Gu_{12} = 0 \tag{48}$$

HB Equation:

For each frequency $f \geq 0$, you have:

$$I_1(f) - [GU_{12}(f) + \mathrm{i}2\pi f\; C\; U_{12}(f)] = 0 \tag{49}$$

Impedance sd_hb_sRL

The device `sd_hb_sRL` describes the behavior of a (real-valued) resistance R in series with a (real-valued) inductance L.

Device name:	`sd_hb_sRL`
Device parameter set name:	`sd_hb_sRL_pset`
Electrodes:	Two electrodes n_1 and n_2
Internal variables:	Current i_1 from n_1 into device

Table 57 Device parameters

Symbol	Description	Range	Unit	Default
R	Resistance	$(-\infty, \infty)$	Ω	0.
L	Inductance	$(-\infty, \infty)$	H	0.

Transient Equation:

$$[R\ i_1(t) + L\ \dot{i}_1(t)] - u_{12}(t) = 0 \tag{50}$$

DC Equation:

$$R\ i_1 - u_{12} = 0 \tag{51}$$

HB Equation:

For each frequency $f \geq 0$, you have:

$$[R\ I_1(f) + \mathrm{i}2\pi f\ L\ I_1(f)] - U_{12}(f) = 0 \tag{52}$$

Harmonic Voltage Source sd_hb_vsource

The device `sd_hb_vsource` represents a harmonic voltage source. It can operate in different modes selected by the `dcmode` parameter:

- If `dcmode=0`, the DC and HB analyses are consistent and the device behaves as a constant voltage source in transient analysis.
- If `dcmode=1`, the DC behavior is given by the transient description at time $t = 0$.

Device name:	`sd_hb_vsource`
Device parameter set name:	`sd_hb_vsource_pset`
Electrodes:	Two electrodes n_1 and n_2
Internal variables:	Current i_1 from n_1 into device

Table 58 Device parameters

Name	Description	Symbol	Domain	Unit	Default
`dcmode`	Mode of DC	–	{0,1}	1	0
`dc`	DC voltage	V_0	$(-\infty, \infty)$	V	0.
`freq`	Frequency of tone	f_1	$[0, \infty)$	Hz	0.

Table 58 Device parameters (Continued)

Name	Description	Symbol	Domain	Unit	Default
mag	Magnitude of tone	M_1	$[0,\infty)$	V	0.
phase	Phase of tone	ϕ_1^{deg}	$(-\infty, \infty)$	degree	0.

Let $\omega_1 := 2\pi f_1$, $\phi_1 := \pi/180\phi_1^{deg}$, and $V_1 = 1/2 M_1 \exp(i\phi_1)$.

Transient Equations:

$$\begin{aligned} u_{12}(t) - V_0 = 0 \quad & \text{if dcmode} = 0 \\ u_{12}(t) - [V_0 + M_1\cos(\omega_1 t + \phi_1)] = 0 \quad & \text{if dcmode} = 1 \end{aligned} \tag{53}$$

DC Equations:

$$\begin{aligned} u_{12} - V_0 = 0 \quad & \text{if dcmode} = 0 \\ u_{12} - [V_0 + M_1\cos(\phi_1)] = 0 \quad & \text{if dcmode} = 1 \end{aligned} \tag{54}$$

HB Equations:

$$\begin{aligned} U_{12}(f) - V_0 = 0 \quad & \text{for } f = 0 \\ U_{12}(f) - V_1 = 0 \quad & \text{for } f = f_1 \\ U_{12}(f) = 0 \quad & \text{else} \end{aligned} \tag{55}$$

Harmonic Current Source sd_hb_isource

The device `sd_hb_isource` represents a harmonic current source.

It is derived in complete analogy to the harmonic voltage source `sd_hb_vsource` (see Harmonic Voltage Source sd_hb_vsource on page 169).

Device name:	`sd_hb_isource`
Device parameter set name:	`sd_hb_isource_pset`
Electrodes:	Two electrodes n_1 and n_2
Internal variables:	None

Table 59 Device parameters

Name	Description	Symbol	Domain	Unit	Default
dcmode	Mode of DC	–	$\{0,1\}$	1	0
dc	DC current	I_0	$(-\infty, \infty)$	A	0.

Table 59 Device parameters (Continued)

Name	Description	Symbol	Domain	Unit	Default
`freq`	Frequency of tone	f_1	$[0,\infty)$	Hz	0.
`mag`	Magnitude of tone	M_1	$[0,\infty)$	A	0.
`phase`	Phase of tone	ϕ_1^{deg}	$(-\infty, \infty)$	degree	0.

Let $\omega_1 := 2\pi f_1$, $\phi_1 := \pi/180\phi_1^{\text{deg}}$, and $I_1 = 1/2M_1\exp(i\phi_1)$.

Transient Equations:

$$\begin{aligned} i_1(t) - I_0 = 0 &\quad \text{if dcmode} = 0 \\ i_1(t) - [I_0 + M_1\cos(\omega_1 t + \phi_1)] = 0 &\quad \text{if dcmode} = 1 \end{aligned} \tag{56}$$

DC Equations:

$$\begin{aligned} i_1 - I_0 = 0 &\quad \text{if dcmode} = 0 \\ i_1 - [I_0 + M_1\cos(\phi_1)] = 0 &\quad \text{if dcmode} = 1 \end{aligned} \tag{57}$$

HB Equations:

$$\begin{aligned} I_1(f) - I_0 = 0 &\quad \text{for } f = 0 \\ I_1(f) - I_1 = 0 &\quad \text{for } f = f_1 \\ I_1(f) = 0 &\quad \text{else} \end{aligned} \tag{58}$$

Multitone Voltage Source sd_hb_vsource2

The device `sd_hb_vsource2` represents a multitone voltage source with two tones. It is an extension of the harmonic voltage source `sd_hb_vsource` (see Harmonic Voltage Source sd_hb_vsource on page 169) to two-tone HB analysis by allowing the specification of two independent tones of oscillation.

Device name:	`sd_hb_vsource2`
Device parameter set name:	`sd_hb_vsource2_pset`
Electrodes:	Two electrodes n_1 and n_2
Internal variables:	Current i_1 from n_1 into device

Table 60 Device parameters

Name	Description	Symbol	Domain	Unit	Default
dcmode	Mode of DC	–	{0,1}	1	0
dc	DC voltage	V_0	$(-\infty, \infty)$	V	0.
freq	Frequency of first tone	f_1	$[0,\infty)$	Hz	0.
mag	Magnitude of first tone	M_1	$[0,\infty)$	V	0.
phase	Phase of first tone	ϕ_1^{deg}	$(-\infty, \infty)$	degree	0.
freq2	Frequency of second tone	f_2	$[0,\infty)$	Hz	0.
mag2	Magnitude of second tone	M_2	$[0,\infty)$	V	0.
phase2	Phase of second tone	ϕ_2^{deg}	$(-\infty, \infty)$	degree	0.

Let $k \in \{1, 2\}$, $\omega_k := 2\pi f_k$, $\phi_k := \pi/180\phi_k^{\mathrm{deg}}$, and $V_k = 1/2 M_k \exp(i\phi_k)$.

Transient Equations:

$$\begin{aligned} u_{12}(t) - V_0 &= 0 \quad \text{if dcmode} = 0 \\ u_{12}(t) - \left[V_0 + \sum_{k=1}^{2} M_k \cos(\omega_k t + \phi_k)\right] &= 0 \quad \text{if dcmode} = 1 \end{aligned} \tag{59}$$

DC Equations:

$$\begin{aligned} u_{12} - V_0 &= 0 \quad \text{if dcmode} = 0 \\ u_{12} - \left[V_0 + \sum_{k=1}^{2} M_k \cos(\phi_k)\right] &= 0 \quad \text{if dcmode} = 1 \end{aligned} \tag{60}$$

HB Equations:

$$\begin{aligned} U_{12}(f) - V_0 &= 0 \quad \text{for } f = 0 \\ U_{12}(f) - V_k &= 0 \quad \text{for } f = f_k \\ U_{12}(f) &= 0 \quad \text{else} \end{aligned} \tag{61}$$

Multitone Current Source sd_hb_isource2

The device `sd_hb_isource2` represents a multitone current source with two tones. It is derived in analogy to the multitone voltage source `sd_hb_vsource2` (see Multitone Voltage Source sd_hb_vsource2 on page 171).

Device name:	`sd_hb_isource2`
Device parameter set name:	`sd_hb_isource2_pset`
Electrodes:	Two electrodes n_1 and n_2
Internal variables:	None

Table 61 Device parameters

Name	Description	Symbol	Domain	Unit	Default
`dcmode`	Mode of DC	–	{0,1}	1	0
`dc`	DC current	I_0	$(-\infty, \infty)$	A	0.
`freq`	Frequency of first tone	f_1	$[0,\infty)$	Hz	0.
`mag`	Magnitude of first tone	M_1	$[0,\infty)$	A	0.
`phase`	Phase of first tone	ϕ_1^{deg}	$(-\infty, \infty)$	degree	0.
`freq2`	Frequency of second tone	f_2	$[0,\infty)$	Hz	0.
`mag2`	Magnitude of second tone	M_2	$[0,\infty)$	A	0.
`phase2`	Phase of second tone	ϕ_2^{deg}	$(-\infty, \infty)$	degree	0.

Let $k \in \{1, 2\}$, $\omega_k{:=}2\pi f_k$, $\phi_k{:=}\pi/180\phi_k^{\mathrm{deg}}$, and $I_k = 1/2M_k\exp(\mathrm{i}\phi_k)$.

Transient Equations:

$$\begin{aligned} i_1(t) - I_0 &= 0 \quad \text{if dcmode} = 0 \\ i_1(t) - \left[I_0 + \sum_{k=1}^{2} M_k\cos(\omega_k t + \phi_k)\right] &= 0 \quad \text{if dcmode} = 1 \end{aligned} \tag{62}$$

DC Equations:

$$\begin{aligned} i_1 - I_0 &= 0 \quad \text{if dcmode} = 0 \\ i_1 - \left[I_0 + \sum_{k=1}^{2} M_k\cos(\phi_k)\right] &= 0 \quad \text{if dcmode} = 1 \end{aligned} \tag{63}$$

HB Equations:

$$\begin{aligned} I_1(f) - I_0 &= 0 \quad \text{for } f = 0 \\ I_1(f) - I_k &= 0 \quad \text{for } f = f_k \\ I_1(f) &= 0 \quad \text{else} \end{aligned} \tag{64}$$

Syntax of Compact Circuit (.ccf) Files

The grammar of a compact circuit file can be defined in Backus–Naur form (BNF). The symbol ε denotes empty production. Terminal symbols are in Courier font. Identifiers are sequences of letters and digits. The first character must be a letter. An underscore (_) is also considered to be a letter.

CCMStrFile	⟶	ε
		| Descriptors
Descriptors	⟶	Device
		| PSet
		| Instance
		| Device Descriptors
		| PSet Descriptors
		| Instance Descriptors
Device	⟶	DeviceHeader DeviceElectrodes DeviceThermodes DeviceInternals DeviceStates DeviceParams DeviceFooter
DeviceHeader	⟶	`DEVICE` DeviceName
DeviceName	⟶	Identifier
DeviceElectrodes	⟶	ε
		| `ELECTRODES`
		| `ELECTRODES` DeviceElectrodeList
DeviceElectrodeList	⟶	DeviceElectrode
		| DeviceElectrode DeviceElectrodeList
DeviceElectrode	⟶	Identifier
DeviceThermodes	⟶	ε
		| `THERMODES`
		| `THERMODES` DeviceThermodeList
DeviceThermodeList	⟶	DeviceThermode
		| DeviceThermode DeviceThermodeList

DeviceThermode	→	Identifier
DeviceInternals	→	ε
		| `INTERNALS`
		| `INTERNALS` DeviceInternalList
DeviceInternalList	→	DeviceInternal
		| DeviceInternal DeviceInternalList
DeviceInternal	→	Identifier
DeviceStates	→	ε
		| `STATES`
		| `STATES` DeviceStateList
DeviceStateList	→	DeviceState
		| DeviceState DeviceStateList
DeviceState	→	SimpleType StateName
		| SimpleType StateName [Dimension]
StateName	→	Identifier
DeviceParams	→	ε
		| `PARAMETERS`
		| `PARAMETERS` DeviceParamList
DeviceParamList	→	DeviceParam
		| DeviceParam DeviceParamList
DeviceParam	→	SimpleType ParamName
		| SimpleType ParamName ParamDefault
		| SimpleType ParamName [Dimension]
		| SimpleType ParamName [Dimension] ParamDefault
SimpleType	→	char
		| int

		| double
		| string
Dimension	→	ε
		| Integer
ParamDefault	→	= ParamValue
DeviceFooter	→	`END DEVICE`
PSet	→	PSetHeader PSetDevice PSetParams PSetFooter
PSetHeader	→	`PSET` PSetName
PSetName	→	Identifier
PSetDevice	→	`DEVICE` DeviceName
PSetParams	→	ε
		| `PARAMETERS`
		| `PARAMETERS` PSetParamList
PSetParamList	→	PSetParam
		| PSetParam PSetParamList
PSetParam	→	ParamName = ParamValue
PSetFooter	→	`END PSET`
Instance	→	InstanceHeader InstancePSet InstanceElectrodes InstanceThermodes InstanceParams InstanceFooter
InstanceHeader	→	`INSTANCE` InstanceName
InstanceName	→	Identifier
InstancePSet	→	`PSET` PSetName
InstanceElectrodes	→	ε
		| `ELECTRODES`
		| `ELECTRODES` InstanceElectrodeList
InstanceElectrodeList	→	InstanceElectrode
		| InstanceElectrode InstanceElectrodeList
InstanceElectrode	→	Identifier

		| Integer
InstanceThermodes	⟶	ε
		| `THERMODES`
		| `THERMODES` InstanceThermodeList
InstanceThermodeList	⟶	InstanceThermode
		| InstanceThermode InstanceThermodeList
InstanceThermode	⟶	Identifier
		| Integer
InstanceParams	⟶	ε
		| `PARAMETERS`
		| `PARAMETERS` InstanceParamList
InstParamList	⟶	InstanceParam
		| InstanceParam InstParamList
InstanceParam	⟶	ParamName = ParamValue
InstanceFooter	⟶	`END INSTANCE`
ParamName	⟶	Identifier
ParamValue	⟶	VectorValue
		| ScalarValue
VectorValue	⟶	[ ]
		| [ScalarList]
ScalarList	⟶	ScalarValue
		| ScalarValue ScalarList
ScalarValue	⟶	Character
		| Integer
		| Real
		| String

In ScalarValue, Table 62 lists the values that are recognized.

Table 62 Scalar values

Type	Value	Example
Character	A character enclosed in single quotation marks.	`'c'`
Integer	An optionally signed sequence of digits.	`1, +2, -33`
Real	An integer, a fixed point value, or a floating-point value as in C++.	`1, -2.0, 3.4e-5`
String	A string consisting of a sequence of characters enclosed in double quotation marks.	`"hello world"`

Real values can also consist of fixed point numbers with a scaling factor as listed in Table 63.

Table 63 Scaling factors

Scaling factor	Value	Example	Scaling factor	Value	Example
`t (tera)`	10^{12}	`1.2t` = 1.2×10^{12}	`u (micro)`	10^{-6}	`7u` = 7×10^{-6}
`g (giga)`	10^{9}	`-2.3g` = -2.3×10^{9}	`n (nano)`	10^{-9}	`8n` = 8×10^{-9}
`meg (mega)`	10^{6}	`4meg` = 4×10^{6}	`p (pico)`	10^{-12}	`5.6p` = 5.6×10^{-12}
`k (kilo)`	10^{3}	`5k` = 5×10^{3}	`f (femto)`	10^{-15}	`0.2f` = 0.2×10^{-15}
`m (milli)`	10^{-3}	`-3.2m` = -3.2×10^{-3}	`a (atto)`	10^{-18}	`2a` = 2×10^{-18}