Simulation Standard

Connecting TCAD To Tapeout

A Journal for Circuit Simulation and SPICE Modeling Engineers

BSIM4 Model Verilog-A Implementation

BSIM4 Model

The Verilog-A hardware description language opens many areas to SPICE users in the field of compact models, allowing manufacturers and universities to study or customize the existing models.

Berkeley BSIM4 model is developed to explicitly address many issues in modeling sub-0.13 microns CMOS technologies and RF high-speed CMOS circuit simulation. Due to its forefront use, BSIM4 was a good candidate for Verilog-A porting.

```
module mosfet(drain, gate, source, bulk);
inout drain, gate, source, bulk;
...
parameter MOBMOD=0;  // Mobility model selector
parameter RDSMOD=0;  // Bias-dependent S/D resistance model selector
parameter IGCMOD=0;  // Gate-to-channel tunneling current model selector
parameter IGBMOD=0;  // Gate-to-substrate tunneling current model selector
parameter CAPMOD=2;  // Capacitance model selector
parameter RGATEMOD=2;// Gate resistance model selector
parameter RBODYMOD=0;// Substrate resistance network model selector
parameter DIOMOD=1;  // Source/drain junction diode IV model selctor
parameter TEMPMOD=0; // Temperature mode selector
parameter GEOMOD=0;  // Geometry-dependent parasitics model selector
parameter RGEOMOD=0; // S/D diffusion resistance and contact model selector
parameter PERMOD=1;  // Source/Drain perimeter model selector
```

Figure 1.

The Silvaco Verilog-A porting is based on the BSIM4 version 3.0 released on May, 9th 2003. The version 2.6.0.R of ***SmartSpice*** Verilog-A interface has been used.

Verilog-A Porting

Silvaco BSIM4 Verilog-A implementation includes all the major physical effects and associated parameters of the original Berkeley version 4.3.0: short/narrow channel effects on threshold voltage, non-uniform doping effects, mobility reduction due to vertical field. All the equations and related parameters have been implemented in a Verilog-A module. The result is a 4,400 lines Verilog-A module.

As in Berkeley code, physical effects model selectors are accessible in the Verilog-A module with the parameters shown in Figure 1.

These model parameters are accessible in the ***SmartSpice*** netlist in the Verilog-A module model card:

```
.MODEL MOSN VLG MODULE = mosfet TYPE = 1 TNOM = 27
+ MOBMOD = 0
+ RDSMOD = 0
+ IGCMOD = 0
+ IGBMOD = 0
...
```

The link with the Verilog-A module is done with `MODULE = mosfet` parameter assignment. The devices are instanciated in the netlist, for example :

```
YVLGm1 node1 node2 0 0 MOSN W=5U L=1.3U
```

The parameters set on this line become instance parameters.

Additional model features like drain/source inversion, N/P MOS type, GMIN convergence improvement and bulk diodes have been implemented. Bulk diodes model can be selected through the model parameter `DIOMOD` shown in Figure 2.

SPICE simulator GMIN option has also been added as a model parameter to improve the convergence properties. The GMIN conductance is taken into account in bulk-drain and bulk-source currents.

Continued on page 2 ...

INSIDE

```
case(DIOMOD)
  0:
    begin
      ...
      cbd = isbd * (evbd + xexpbvd - t1 - 1.0) + GMIN * vbd_jct;
    end
  1:
    begin
      ...
      if (t2 < -`EXP_THRESHOLD)
          cbd = isbd * (`MIN_EXP - 1.0) + GMIN * vbd_jct;
      else if (vbd_jct <= vjdmfwd)
          cbd = isbd * (evbd - 1.0) + GMIN * vbd_jct;
      else
          cbd = ivjdmfwd - isbd + t0 * (vbd_jct - vjdmfwd) + GMIN * vbd_jct;
    end
  2:
    ...
endcase
```

Figure 2.

Bulk diode currents contribution is added with <+ operator :

```
I(drainb, drainp) <+ TYPE * cbd;
```

N/P MOS type is accounted for with TYPE model parameter.

Model Convergence

Limitation functionalities have been enabled to help the model to converge with large circuits simulation. maxdelta is a voltage nature attribute which has been added recently in the ***SmartSpice*** Verilog-A interface. This new feature allows to limit the per-iteration voltage change and is disabled by default.

For using voltage limitation, one must set in Verilog-A file:

```
`define  VOLTAGE_MAXDELTA <value>
`include "discipline.h"
```

<value> is the greatest voltage change allowed between 2 consecutive Newton-Raphson iterations. The preprocessor directive `define VOLTAGE_MAXDELTA <value> is a shortcut: it overrides the maxdelta default voltage nature attribute and must be set before including discipline.h file.

BSIM4 ***SmartSpice*** internal model has been successfully replaced by the Verilog-A model in the simulation of a 72-devices two-bit MOSFET adder, as shown in the output display Figure 3.

The results fit the simulation with ***SmartSpice*** internal model level=14. In AC analysis, the model has been validated using an operational amplifier benchmark file supplied by Berkeley University team.

Conclusion

Berkeley BSIM4.3.0 model has been successfully implemented in Verilog-A HDL at SILVACO. The voltage limitation feature recently implemented in Verilog-A interface improves convergence significantly when simulating large circuits, with an acceptable simulation time. The Verilog-A model is freely available on SILVACO website **(http://www.silvaco.com).**

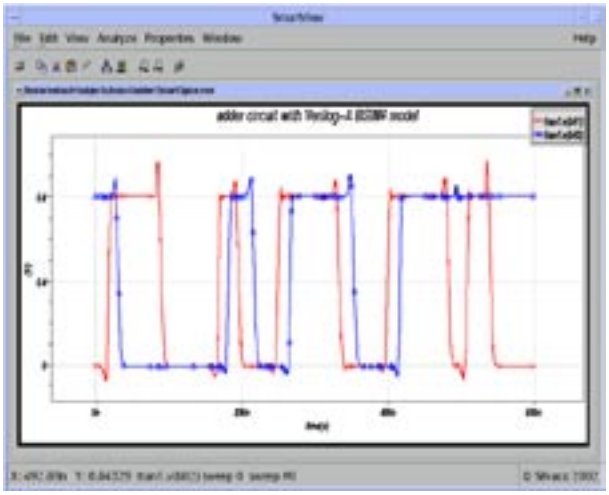

Figure 3. Verilog-A BSIM4.3.0 model based two-bit MOSFET adder.

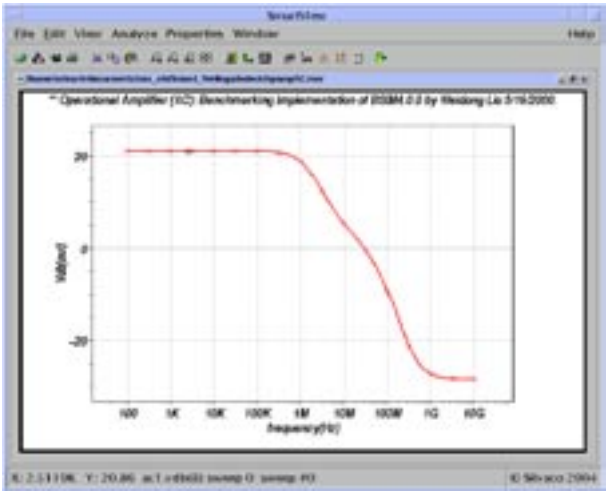

Figure 4. AC analysis of an operational amplifier with BSIM4 Verilog-A model.

New Philips MOS20 LDMOS Model in SmartSpice

Introduction

Philips MOS20 was released in January 2004. Its purpose is to provide a high-voltage compact model to describe both operation of the channel region and drift region under the thin gate oxide. It can be used as a replacement for the macro-model composed of MOS9 and MOS30 to describe Lateral or Vertical Double-diffused MOS (LDMOS or VDMOS) or Extended-Drain MOS devices (EPMOS).

A Compact Model to Replace Two Macro-Models

Before MOS20 was released, simulating LDMOS devices required the use of macro-models. To do this, ***SmartSpice*** provided the following models: MOS9, MOS30 and MOS40.

One was used to achieve channel region simulation (MOS9) and another was required to account for the drift region (MOS30 or MOS40). Both were assembled at the netlist level, for example in a subcircuit.

MOS20 accounts for both regions, and above all internally computes the voltage at the transition between channel and drift regions. This improvement is very important with regard to convergence. It does not require any external node to link two models together, and this particular voltage is available to model equations since it is internally computed.

The only case when associating MOS20 and MOS40 is still necessary is when very high voltages devices are simulated. In this case, a macro model will help accounting for the drift region under the thick field oxide.

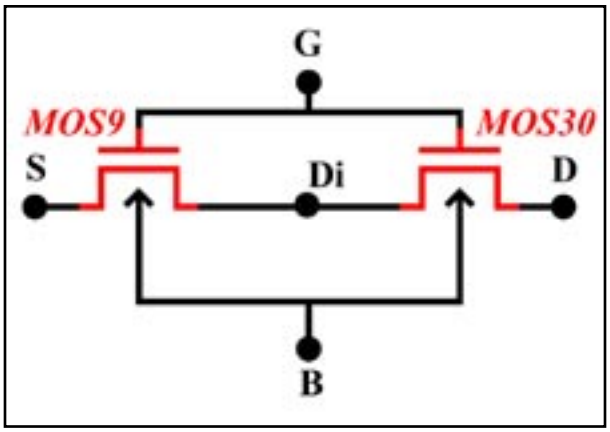

Figure 1. Macro-model using MOS9 and MOS30.

MOS20 Takes Advantages of all Philips Models

The model is based on all the best achievements in compact models. Model core has been derived from the SOI-LDMOS model from University of Southampton.

MOS20 equations are based on surface potentials. With this technique, it provides an accurate description of all operating regimes. Only one equation is computed for all regimes, ensuring that no discontinuity appears, and no smoothing function is used. The equations implemented are based on MOS11 equations. Surface potentials are calculated using an approximation of Poisson equation, in order to get explicit expression of surface potential with regard to node voltages.

This way, MOS20 benefits from all the improvements and experience acquired with those models.

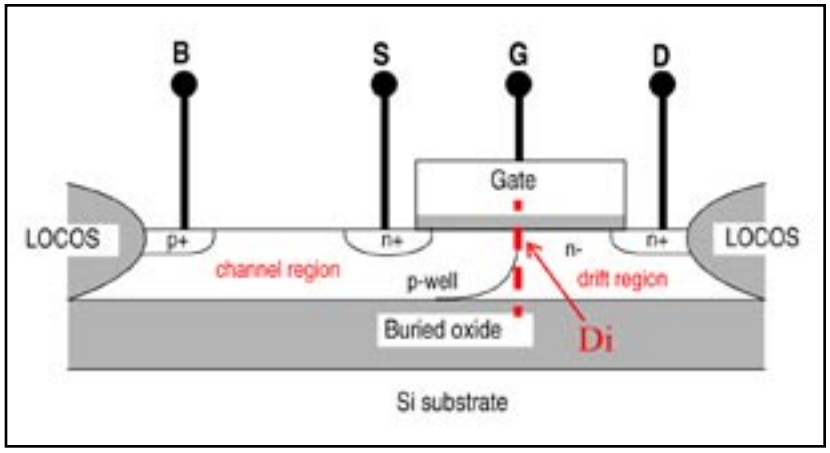

Figure 2. The region under the thin gate oxide of LDMOS device (n-channel).

In addition to these core equations, MOS20 also accounts for :

- Mobility reduction
- Velocity saturation
- Drain-Induced Barrier Lowering (DIBL)
- Static feedback
- Channel length modulation
- Weak avalanche current

Silvaco Implementation

MOS20 implementation in ***SmartSpice*** takes advantages of both Philips' original equations, and SmartSpice common MOSFET features such as :

- Transient Noise analysis using Philips detailed noise model
- RF analyses with ***SmartSpiceRF***
- Speed improvements : VZERO and BYPASS options, parallel architectures
- Convergence control and user-friendly hints
- Out-of-bound guards, both for parameters and internal values
- Extrinsic elements : bulk diodes, Source/Drain resistances

All these features allow the user to make the most of MOS20 model.

Examples

The plot shown in Figure 3 is a simple Id=f(Vd, Vg) characteristic. The uprising part of the curve is due to the weak-avalanche current (or impact ionization). This current variation is important because it has an influence on dissipated power, that results in temperature elevation. Since this is a power device, temperature is critical.

Another interesting plot (Figure 4) is the potential at the transition between channel region and drift region. These curves help to investigate how the model is computed, giving an inside view of the specific LDMOS structure. This curves shows how V_{Di} potential varies and reflects where the transition lies, between V_D and V_S potentials.

MOS20 also provides an accurate charge description. For Gate and Bulk charges, the nodal charges are simply decomposed into drift and channel contribution. The Drain charge

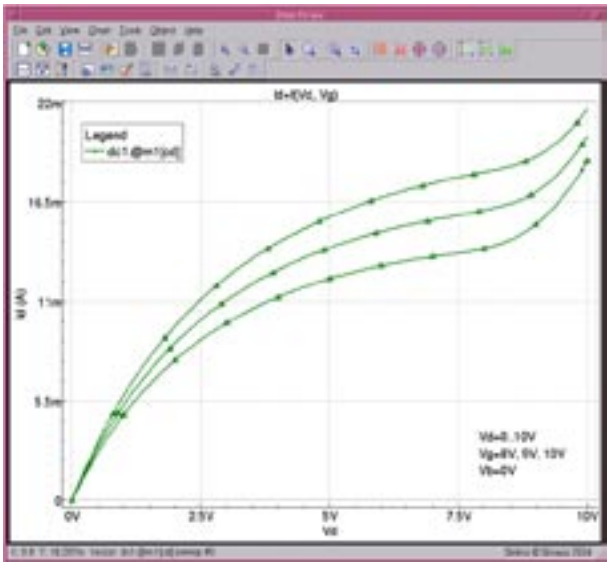

Figure 3. Id=f(Vd,Vg) characteristics, showing avalanche current.

is computed a different way, distinguishing two cases : well above threshold, and below threshold. The following example is a plot of C_{GG} capacitance, with regard to V_D and V_G (Figure 5).

Conclusion

The MOS20 model is relevant for its ability to account for high-voltage devices without using macro-models. Furthermore, it is based on all the latest modeling techniques such as surface potentials. For these reasons, users can rely on an accurate model with good convergence properties.

References

[1] Detailed model parameters and equations can be found in SmartSpice modeling manual vol 1.

[2] Philips' website also contains all the documentation and literature about MOS 20 :

http://www.semiconductors.philips.com/Philips_Models

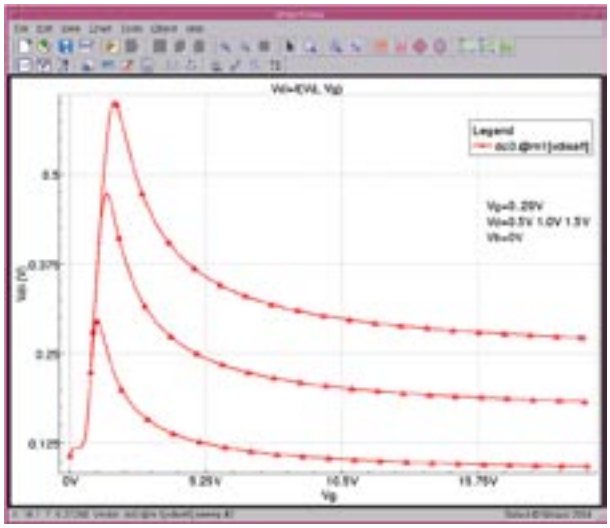

Figure 4. Potential at transition between channel and drift regions vs Vg, Vd.

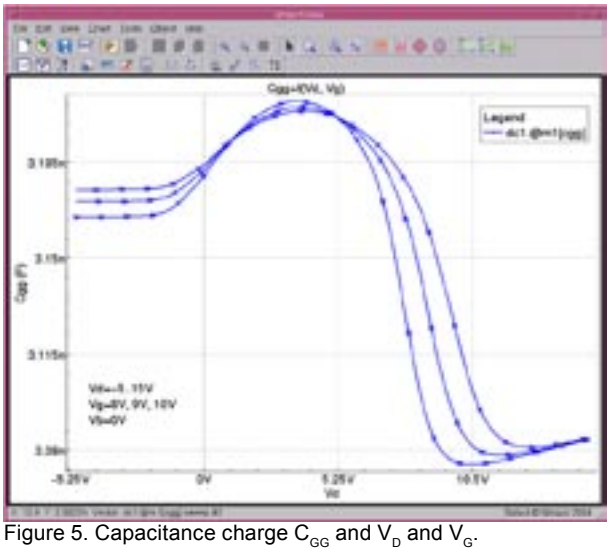

Figure 5. Capacitance charge C_{GG} and V_D and V_G.

Stress Effect Model in BSIM3v3 Model

Stress effect models are now implemented in major models such as BSIM4 or HiSIM. The need for evermore accurate models with a strong relation to technology is accute. Since BSIM3v3 is still a widely-used model and has not been totally replaced by its successor, an improvement was made to the model in ***SmartSpice*** to fulfill customers need for stress effect equations.

Background

The stress effect became important for simulation because of more and more shrinking processes. The smaller devices now require efficient isolation techniques. One of them is Shallow Trench Isolation (STI), mainly used with strain channel materials.

Shallow Trench Isolation is used to replace LOCOS, as shown on the schematic in Figure 1.

This particular process induces a mechanical stress on the device structure. Because of this behavior, device performance is related to the dimension of the active area, as well as the location of the device.

It has been shown that :

- Stress has an influence on mobility
- Saturation velocity is also modified
- Dopant diffusion during processing is modified, leading to different doping profiles. This implies a threshold voltage shift as well as changes in second-order effect such as Drain-Induced Barrier Lowering (DIBL) and body effect

Berkeley University considered that the effect of stress is due to two main mechanisms: mobility variation induced by changes in the band structure, and influence on threshold voltage because of different doping profiles as explained above.

Both of these mechanisms have the same dependence on $1/L_{OD}$ (invert of the Length of Oxide Definition), but show a different trend with regard to width and length of the device.

Implementation

When the effect is enabled, the following model elements are modified :

- Mobility
- Saturation velocity
- Threshold voltage
- Drain-Induced Barrier Lowering (DIBL)
- Body Effect

The approach used is to tune model parameters values to account for the Stress Effect : it is a phenomenological model.

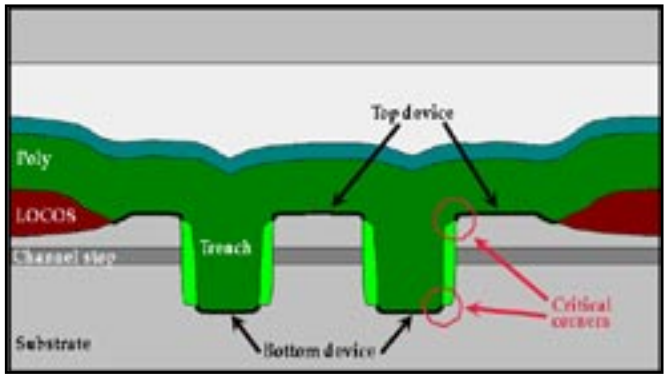

Figure 1. Process cross-section showing showing stress regions.

The Stress Effect was initially developed by Berkeley University in the BSIM4 model. It was improved with new equations along the official releases of the model. Silvaco implementation (both in BSIM3v3 and BSIM4) provides all of these implementations, using a dedicated selector: *STIMOD*.

In ***SmartSpice***, the user can select among four equations sets:

- *STIMOD = 0*: No stress effect
- *STIMOD = 1*: Berkeley model, -version
- *STIMOD = 2*: TSMC model for irregular devices
- *STIMOD* = 3: Berkeley model for multi-finger devices

Enabling the Stress Effect model does not imply a longer simulation time, since expressions do not depend on voltages. It is computed only once before running the simulation.

Berkeley β-version model (STIMOD=1)

Since the implementation was greatly changed between BSIM4v3.0-β and BSIM4v3.0, Silvaco chose to keep both implementations. The mobility *U0(T)*and *VSAT(T)* carrier velocity are computed using the following equations:

$$U0'(T) = \frac{U0(T)}{1+\rho} \quad \text{and} \quad VSAT'(T) = \frac{VSAT(T)}{1+K\cdot\rho}$$

where

$$\rho = SK0\cdot\left[\rho'\cdot\frac{\sinh\left(\frac{NF}{2\cdot SL}\cdot(SD+L)\right)}{NF\cdot\sinh\left(\frac{SD+L}{2\cdot SL}\right)}+\frac{SK2}{W}\cdot\exp\left(-\frac{W}{SW}\right)\right]$$

$$\rho' = \left(1+\frac{SK1}{LOD}\right)\cdot\left(\exp\left(-\frac{SA+D}{SL}\right)+\exp\left(-\frac{SB+D}{SL}\right)\right)\cdot\frac{\sinh\left(\frac{L_{eff}}{2\cdot SL}\right)}{\frac{L_{eff}}{2\cdot SL}}$$

$$D = 0.5\cdot(NF\cdot L+(NF-1)\cdot SD)$$

$$LOD = SA+SB+2\cdot D$$

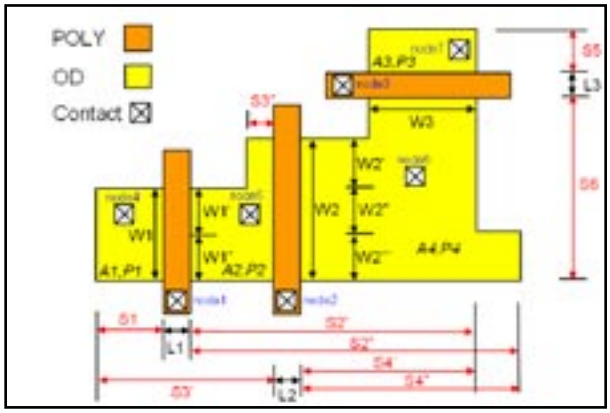

Figure 2. Layout dimensions for stress calculation STIMOD=2.

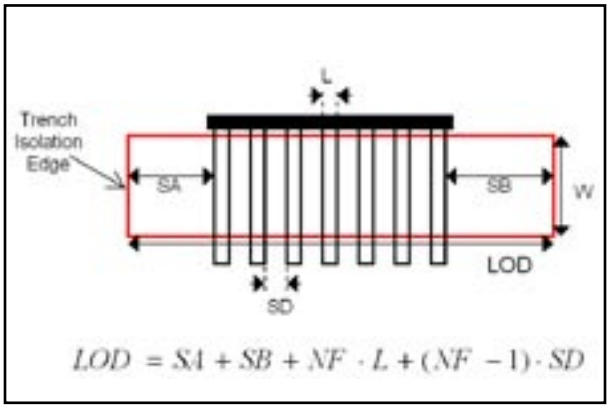

Figure 3. Berkley model diagram.

TSMC Model for Irregular Devices (STIMOD=2)

This model is suitable when there is only one finger. Parameters are used to describe square elements of the oxide layer geometry (Figure 2).

The equations used to compute the stress effect are the same as the ones when STIMOD=3, but with different intermediate geometry definitions:

$$\text{lnv\_sa} = \sum_{i=1}^{\#0} \frac{SW_i}{W} \cdot \frac{1}{SA_i + 0.5 \cdot L} \text{ and } \text{lnv\_sb} = \sum_{i=1}^{\#0} \frac{SW_i}{W} \cdot \frac{1}{SB_i + 0.5 \cdot L}$$

Berkeley Model for Multi-Finger Devices

This model is the one released by Berkeley in the final BSIM4v3.0 version. Parameters are used to compute the intermediate length LOD (Length of Oxide Definition).

Then equations shown in Figure 4 are used.

Conclusion

The Stress Effect model was first tested and improved in the BSIM4 model. Now that it is fully validated, it is also available in BSIM3v3 with the same selectable implementations. This will open new opportunities to customers that still use BSIM3v3 as standard.

$$U0'(T) = U0(T) \cdot \frac{1+\rho}{1+\rho_{ref}}$$

$$VSAT'(T) = VSAT(T) \cdot \frac{1+\text{KVSAT}\cdot\rho}{1+\text{KVSAT}\cdot\rho_{ref}}$$

$$VTH0'(T) = VTH0(T) + \frac{\text{KVTH0}}{\text{Kstress\_vth0}} \cdot (\text{lnv\_sa} + \text{lnv\_sb} - \text{lnv\_sa\_ref} - \text{lnv\_sb\_ref})$$

$$K2'(T) = K2(T) + \frac{\text{STK2}}{\text{Kstress\_vth0}^{LODK2}} \cdot (\text{lnv\_sa} + \text{lnv\_sb} - \text{lnv\_sa\_ref} - \text{lnv\_sb\_ref})$$

$$ETA0'(T) = ETA0(T) + \frac{\text{STETA0}}{\text{Kstress\_vth0}^{LODETA0}} \cdot (\text{lnv\_sa} + \text{lnv\_sb} - \text{lnv\_sa\_ref} - \text{lnv\_sb\_ref})$$

where

$$\rho = \frac{\text{KU0}}{\text{Kstress\_u0}} \cdot (\text{lnv\_sa} + \text{lnv\_sb})$$

$$\text{Kstress\_u0} = \left(1 + \frac{\text{LKU0}}{(L+XL)^{\text{LLODKU0}}} + \frac{WKU0}{(W+XW+WLOD)^{\text{WLODKU0}}} + \frac{PKU0}{(L+XL)^{\text{LLODKU0}} \cdot (W+XW+WLOD)^{\text{WLODKU0}}}\right) \cdot (1 + TKU0 \cdot (T_{ratio} - 1))$$

$$\text{lnv\_sa} = \frac{1}{NF} \cdot \sum_{i=1}^{NF-1} \frac{1}{SA + 0.5 \cdot L + i \cdot (SD + L)} \text{ and } \text{lnv\_sb} = \frac{1}{NF} \cdot \sum_{i=1}^{NF-1} \frac{1}{SB + 0.5 \cdot L + i \cdot (SD + L)}$$

$$\rho_{ref} = \frac{\text{KU0}}{\text{Kstress\_u0}} \cdot (\text{lnv\_sa\_ref} + \text{lnv\_sb\_ref})$$

$$\text{lnv\_sa\_ref} = \frac{1}{\text{SAREF} + 0.5 \cdot L} \text{ and } \text{lnv\_sb\_ref} = \frac{1}{\text{SBREF} + 0.5 \cdot L}$$

$$\text{Kstress\_vth0} = 1 + \frac{\text{LKVTH0}}{(L+XL)^{\text{LLODKVTH}}} + \frac{WKVTH0}{(W+XW+WLOD)^{\text{WLODKVTH}}}$$

Figure 4. Parameter value modification due to stress effects.

New SmartLIb Library of Models

Overview

In all previous versions of ***SmartSpice*** the model code (BSIM, diode etc.) was included in the one executable (***SmartSpice***). This means any updates to the model code would take a while to reach the customer because of the full SPICE functionality checks required before releasing a new ***SmartSpice*** version. We have therefore separated the ***SmartSpice*** core from the modeling code to eliminate this delay and dependency. Now the new model release time has been reduced by having a separate library that the customer can download from the web and include into the ***SmartSpice*** program through the use of these described new functions. All previous ***SmartSpice*** functionality is maintained as before. This new configuration is only of interest to customer who wish to explore model changes in more depth.

The -slinstall Option

The "-slinstall" option moves libraries from a download directory to the installation directory.

Example 1

```
smartspice -slinstall mydir 0.2.0.R
```

where `mydir` is the directory of the libraries that have to be installed, and 0.2.0.R is the version number of the ***SmartLib*** library it belongs to.

This command will inspect the files that lay in the mydir directory. ***SmartSpice*** moves each file to the installation directory if it can be used as a library, and there is no file with the same name. This insures that the file installed will work correctly, and that an installed file cannot be lost while adding a new one.

To use this option, you must have the rights to modify the installation. Ask to your system Administrator if you are allowed such rights.

How to Use the -slinstall Option

Example 1

The situation: I've downloaded the Solaris library `libSGP105R.so.tar` which is in a PUB directory from my home.

At this point, you need to untar the file. Move it to the PUB directory:

```
cd ~/PUB
tar -xvf libSGP105R.so.tar
```

Now you must have the `libSGP105R.so` in the directory:

```
smartspice -slinstall . 1.0.0.R
```

SmartSpice indicates that `libSGP105R.so.tar` cannot be loaded so it won't be installed. It also indicates that `libSGP105R.so` has been copied, and that the installation has been successful.

Example 2

The situation: I've downloaded the full set of Windows libraries for ***SmartSpice*** 1.1.0.R, and they are in the C:\tmp folder.

Use a Zip de-compactor to get the dll files. If you can delete the zip files it will make the following simpler.

In the Start Menu click on Run. When the window appears, type the following command:

```
smartspice -slinstall c:\tmp 1.1.0.R
```

A window opens and indicates that the folder has been created, and that each dll is being copied in the installation folder.

The -sllist Option

The list option lets you have a look at the installed libraries for a given ***SmartLib*** version.

Example 1

```
smartspice -sllist 0.2.0.R
```

where 0.2.0.R is the version number of the ***SmartLib*** you want to inspect.

This command indicates the interface version number of each file in the installation directory, if it is a compatible library, and if the lib can be used or even loaded. It also can be used to confirm if the installation procedure succeeded.

The -slremove Option

The remove option removes the older version of the ***SmartLib*** .

Example 1

```
smartspice -slremove 0.2.0.R
```

where 0.2.0.R is the ***SmartLib*** you want to erase.

To use this option, you must have the rights to modify the installation. Ask to your system Administrator if you are allowed such rights.

Warning: Once a ***SmartLib*** has been removed it cannot be used again.

The -slsmartlibconf Option

This option updates the configuration file, so that ***SmartSpice*** will use the newer libraries.

Example 1

```
smartspice -slsmartlibconf 0.2.0.R
```

where 0.2.0.R is the version number of the ***SmartLib*** you want to use.

SmartSpice inspects the files in the ***SmartLib*** installation directory. It chooses the valid files with the higher version number to update the configuration file.

How to Use the -slsmartlibconf Option

Example 1
I've just installed my Solaris library libSGP_1_0_5_R.so, and I want ***SmartSpice*** to use this library.

Just type:

```
smartspice -slsmartlibconf 1.0.0.R
```

and insure you have a `.SmartSpice.conf` file in your home directory:

```
mv ~/.SmartSpice.conf SmartSpice.conf.old
```

Example 2
I've just downloaded and installed the full set of libraries from ***SmartLib*** 1.1.0.R. How do I make ***SmartSpice*** use it?

In the Start Menu click **Run**, and then type:

```
smartspice -slsmartlibconf 1.1.0.R
```

How to Install a Downloaded Library

- Create a download directory in your home directory:
  ```
  cd $HOME
  mkdir download
  ```
- Using your web browser, go to the Silvaco Resource Centre Web site and download the library into the folder you have just created.
- Prepare the library to be installed:
  ```
  cd $HOME/download
  tar -xvf *.tar
  ```
- Install the library:
  ```
  cd $HOME
  smartspice -slinstall download 1.0.0.R
  ```
- Verify that the library has been installed and is in the list:
  ```
  smartspice -sllist 1.0.0.R
  ```
- Make the Library Active:
  ```
  smartspice -slsmartlibconf 1.0.0.R
  ```

Typegroup	Technology	Internal name	Info	Level	Lib name
npn	BJT	BJT	Bipolar Junction Transistor	1, 2	libSGP
pnp		VBIC	VBIC Bipolar Junction Transistor	5	libVBIC
lpnp		HICUM	HICUM Bipolar Junction Transistor	6	libHICUM
		PBJT	Mextram BJT (Philips)	503	libMEXTRAM
		MODELLA	Philips TPL500 Bipolar Transistor	500	linMODELLA
		HBT	Hetero-Junction Bipolar Transistor	20	libHBT
nmos	SOI	BSIM31SOI	Berkeley SOI MOSFET model version 1 (level 25)	25	libBSIM3SOIv1
pmos		BSIM3SOI2DD	Berkeley SOI MOSFET model version 2 (level 27)	27	libBSIM3SOIv2_DD
ntft		BSIM3SOI2FD	Berkeley SOI MOSFET model version 2 (level 26)	26	libBSIM3SOIv2_FD
ptft		BSIM3SOI2PD	Berkeley SOI MOSFET model version 2 (level 29)	29	libBSIM3SOIv2_PD
		BSIM3SOI3	Berkeley SOI MOSFET model version 3 (level 33)	33	libBSIM3SOIv3
		UFS	University of Florida SOI Model (level 21)	21	libUFS
		LETISOI	CEA/LETI SOI MOSFET model	32	libLETISOI
	TFT	TFT	MOS field-effect transistor	15	libLeroux
		PTFT	PolySi TFT model	16	libBerkeleyTFT
		MOS15	MOS15 TFT Model	35	libRPIaSi
		MOS16	RPI Poly-Si TFT Model	36	libRPIpolySi
	MOSFET	MOS123	MOS field-effect transistor	1, 2, 3	libMOSlevel123
		BSIM1	Berkeley Short Channel IGFET Model	4, 13	libBSIM1
		BSIM3	Berkeley Short Channel IGFET Model) Version-3 (level 81)	81	libBSIM3
		BSIM3v3	Berkeley Short Channel IGFET Model Version-3 (level 8, 49, 53)	8, 49, 53	libBSIM3v3
		BSIM3M	Modified Berkeley Short Channel IGFET Model Version 3 (level 7,10,47)	7, 10, 47	libBSIM3v3
		BSIM4S	Berkeley Short Channel IGFET Model-4 (level 14, 54)	14, 54	libBSIM4
		MOS11	Philips MOS11 model	11,63	libMOS11
		MOS31	MOS31 MOSFET Model	30, 31, 40	libMOS31
		MOS20	Philips MOS20 LDMOS model	20	libMOS20
		EKV	EKV MOSFET Model	44	libEKV
		BSIM3H	High-Voltage MOSFET Model (level 88)	88	libHVMOS
		HISIM	Hiroshima University STARC IGFET Model (level 111)	111	libHiSIM
njf	JFET/	JFET	Junction/Schottky contact	1, 2, 3, 4, 5, 6	libJFETMESFET
pjf	MESFET		field-effect transistor		
nmf					
pmf					
d	Diode	DIO	Junction Diode	1, 3	libDiodeL13
		DIO2	Fowler-Nordheim Diode	2	libDiodeL2
		DIO500	Diode Level 500	500	libDiodeL500
		JCAP	Junction Capacitor	9	libJuncap
		LAS1	VCSEL model	4	libVCSEL
c	Capacitance	FCAP	Ramtron Ferroelectric Capacitance Model	5	libFCAP
cap		FRMC	Ramtron Ferroelectric Capacitance Model	6	libFRMC

Table 1. ***SmartLib*** Models and Corresponding Shared libraries.

Calendar of Events

January

1	
2	
3	
4	
5	
6	
7	
8	
9	
10	
11	
12	
13	
14	
15	
16	
17	
18	
19	
20	
21	
22	
23	
24	
25	
26	
27	ASP DAC - Yokohama, Japan
28	ASP DAC - Yokohama, Japan
29	EDS Fair - Pacifico, Yokohama
30	EDS Fair - Pacifico, Yokohama
31	

February

1	
2	DesignCon - Santa Clara, CA
3	DesignCon - Santa Clara, CA
4	DesignCon - Santa Clara, CA
5	DesignCon - Santa Clara, CA
6	
7	
8	
9	
10	
11	
12	
13	
14	
15	
16	DATE - Paris, France
17	DATE - Paris, France
18	DATE - Paris, France
19	DATE - Paris, France
20	DATE - Paris, France
21	
22	
23	
24	
25	
26	
27	
28	
29	

Bulletin Board

PolarFab Delivers Silvaco Process Design Kits for PBC4 BiCMOS/DMOS Process

PolarFab, a U.S.-based pure-play semiconductor foundry, and Silvaco announced the availability of process design kits (PDKs) for PolarFab's premier process for power management applications. The PDK supports a complete analog and mixed-signal design flow with Silvaco circuit simulation and custom IC CAD tools. This process design kit includes parameterized cells for all of the fifty active and passive components. Silvaco was able to complete a comprehensive PDK in a fraction of the time it takes to develop PDKs for other design tool flows."

See Silvaco at EDSFair2004 in Yokohama Japan

Electronic Design and Solution Fair 2004 EDSFair assembles specialized information on advanced device technologies, such as EDA, ASICs, FPGA/PLDs IP re-usage, embedded software and design services. Also featured are the latest trends and targets for further development of electronics technologies

If you would like more information or to register for one of our our workshops, please check our web site at ***http://www.silvaco.com***

Hints, Tips and Solutions

Colin Shaw, Applications and Support Engineer

Remote .ALTER processing

A new parallelization method has been implemented into ***SmartSpice***. Now .ALTERs can be destributed not only over several CPUs (by using -P option), but over a network of computers as well.

Remote .ALTER processing works the following way:

When it is invoked (by using -remote command line option), ***SmartSpice*** will read the input deck and check for .ALTER statements in it. If there are no .ALTER statements in the input deck, ***SmartSpice*** will just continue simulating the given netlist in batch mode. If .ALTERs are found, ***SmartSpice*** will extract parts of the netlist which form entire circuits and write out each circuit as separate files (.ALTER files). Resulting files containing one altered circuit each are named by adding the suffix -n to the composite basical netlist file name and have no extension. Number of produced files equals to the amount of .ALTER statements in the composite netlist plus one (deck without .ALTERs).

SmartSpice will then collect hosts from ralter_hosts list variable set in SmartSpice.ini file. This data consists of host names to distribute .ALTERs onto and number of CPUs to use on each host. On the next step ***SmartSpice*** will try to launch one simulation on each host (filling all hosts if number of .ALTERs is greater than amount of hosts specified). If the simulator on certain host is launched successfully, parent ***SmartSpice*** will check for amount of available CPUs source child with according .ALTER file and start simulation. Then, if there are not processed .ALTERs remaining, the parent will launch additional simulators on this given host (up to number CPUs specified by user, but never exceeding actual number of CPUs present).

If host fails for some reason, the .ALTER file it was processing will be released, and parent ***SmartSpice*** will retry to launch the simulator on the failed host again. If the host fails for ralter_numretries times, the parent ***SmartSpice*** will stop retrying to launch the remote simulator on it. If all hosts fail, but unprocessed .ALTERs still remain, ***SmartSpice*** will exit with error an message.

When remote ***SmartSpice*** finishes simulation, it signals the parent that it became idle and is immediately sourced with another unprocessed .ALTER file. If there are no unprocessed .ALTERs left, remote ***SmartSpice*** will terminate.

Output files (raw, out, err, etc.) are created for each separate circuit file.

When all separate files are processed, parent ***SmartSpice*** removes these files and terminates. Parent does not do any simulation - it just monitors remote simulators and acts as the server manager.

Usage:

```
smartspice <input-file> -remote
```

Variables that can be set in SmartSpice.ini file:

ralter_hosts = (*hostname<=numCPUs>*
** *<<hostname<=numCPUs>> ...* >)**

This list variable specifies the list of hosts to destribute .ALTERs onto. Syntax is important. There has to be space after opening parenthesis, and before closing parenthesis. When number of CPUs to use is specified, there should not be spaces around '='. If the number of CPUs is not specified, all available CPUs on the remote host will be used.

`ralter_hosts` variable is mandatory because if it is not set, ***SmartSpice*** will not be able to destribute .ALTERs over the network, and will continue simulation localy in batch mode.

Example:

```
ralter_hosts = ( hostone hosttwo=4 host-
three=2 )
```

ralter_outpath = *"val"*

Path to store output files. Default: path were original composite netlist is located.

ralter_timeout = *val*

Timeout in seconds after which remote host is considered failed. Default: 30 seconds.

ralter_numretries = *val*

Number of retries to launch simulator on remote host. Default: 5

Note: TMP environmental variable must be set to location for temporary files.

All computers in network must have the same operatng system. Remote simulators are invoked by using invocation line used for parent ***SmartSpice*** , thus if parent was launched on a Linux platform, it will be unable to launch remote simulator on a Unix platform.

Remote .ALTER processing is not currently supported for Windows platforms.

`-remote` option also forces ***SmartSpice*** to run in batch mode.

If specified in command line, the startup file must include a full path. Otherwise remote ***SmartSpice*** will fail to load it.

.OVERSHOOT Statement Improved

.OVERSHOOT statement has been improved. In previous versions of ***SmartSpice*** this statement allowed the user to check all nodes on every timepoint during the simulation, and report nodes with voltages exceeding limits specified by ***vmin*** and ***vmax***. Report contained node names along with time points whenever a violation occurs.

Now .OVERSHOOT functionality has been enhanced making it more powerfull and more convinient to use. It is now possible to specify not only voltage thresholds, but also minimum spike duration to detect, and nodes list that have to be excluded from checking.

Output format has also changed. Previous report contained node names and timepoints when a spike occured, and was sorted by timepoints. Now report is sorted by nodes. If spike occured on any certain node, first there's a line stating that v(*x*) > *vmax* or v(*x*) < *vmin* and then is followed by listing of spikes detected. This list contains spike start time, spike end time, spike duration and peak voltage (maximum voltage in case of >*vmax* spike or minimum voltage in <*vmin* spike case).

It is now possible to specify more than one statement for single analysis. But like in previous versions, the only analysis supported is TRAN.

Syntax

```
.OVERSHOOT filename="val" <vmin=val>
<vmax=val> <duration=val>

+ <excludenodes=nodename <nodename
<...<nodename>>>
```

filename: The name of the file where overshoot information is saved. This file contains comment lines that identify the circuit name and type, the type and date of the simulation, specified params and report on detected spikes.

vmin=val, vmax=val: These value specify voltage limits. At least one of them has to be present.

duration=val: Specifies minimum duration of spike to detect. If omitted, ***SmartSpice*** will detect spikes of any duration.

excludenodes: List of nodes to exclude from checking. Wildcards can be used. If not specified, ***SmartSpice*** will check all nodes.

Example:

```
.OVERSHOOT filename="o.ost" vmin=-0.2 vmax=0.9
duration=1p + excludenodes="0 va* x3.*"
```

New RNOISE Parameter for Resistor

A new optional device parameter, specified resistance value for noise analysis (which can be different from DC/TRAN or AC analysis resistance values) was added for linear resistor specification:

RNOISE|NOISE=val: Resistance for noise simulation. Default is AC or R value.

Example:

```
R 3 7 R=0.001 AC=1e10 RNOISE=0.001
```

In this example resistor R5, connected between nodes 3 and 7, for DC and Transient Analysis has a resistance value R=0.001 Ohm, Noise Analysis resistance is Rnoise=0.001Ohm, and AC analysis resistance is AC=1e10 Ohm.

Multiple nested .DC, .AC, .TRAN sweep

Now, ***SmartSpice*** allows the user to use the multiple nested parametric sweep for .DC, .AC, .TRAN analyses.

Syntax

.TRAN ... **SWEEP** *VAL1* start1 stop1 step1 **SWEEP** *VAL2* start2 stop2 step2 ...

.DC ... **SWEEP** *VAL1* start1 stop1 step1 **SWEEP** *VAL2* LIST nplist val1 val2 ...

.AC ... **SWEEP** *VAL1* DEC np start stop **SWEEP** *VAL2* MODIF=VAL1 <PRTBL>...

The multiple nested parametric sweep works the following way:

At first ***SmartSpice*** saves the current amounts and makes the start initialization of the all sweep variables.

After that, ***SmartSpice*** performs the specified analysis, changing settings for the first sweep variable. When the loop of the first sweep variable is finished ***SmartSpice*** makes the start initialization for the first sweep variable, and next step for the second sweep variable. ***SmartSpice*** repeat the above mentioned actions until the loop of the second sweep variable will not be finished.

The multiple sweep uses nested loops like:

```
for (, ,)
    for (, ,)
        for(, ,)
```

Finally, ***SmartSpice*** restores the amounts of sweep variables.

The multiple nested parametric sweep is an extension for the standard sweep ideology. Sweep blocks have the same syntax and must be started with keyword "SWEEP".

Call for Questions

If you have hints, tips, solutions or questions to contribute, please contact our Applications and Support Department
Phone: (408) 567-1000 Fax: (408) 496-6080
e-mail: support@silvaco.com

Hints, Tips and Solutions Archive

Check our our Web Page to see more details of this example plus an archive of previous Hints, Tips, and Solutions
www.silvaco.com

Your Investment is Safe

20 Years and Growing

Financially Rock-Solid

Fiercely Independent

Analog/MS EDA Design Leader

We are NOT For Sale

SILVACO
INTERNATIONAL

USA Headquarters:

Silvaco International
4701 Patrick Henry Drive, Bldg. 2
Santa Clara, CA 95054 USA

Phone: 408-567-1000
Fax: 408-496-6080

sales@silvaco.com
www.silvaco.com

Contacts:

Silvaco Japan
jpsales@silvaco.com

Silvaco Korea
krsales@silvaco.com

Silvaco Taiwan
twsales@silvaco.com

Silvaco Singapore
sgsales@silvaco.com

Silvaco UK
uksales@silvaco.com

Silvaco France
frsales@silvaco.com

Silvaco Germany
desales@silvaco.com

*Products Licensed through Silvaco or e*ECAD*