Simulation Standard

Connecting TCAD To Tapeout

A Journal for Circuit Simulation and SPICE Modeling Engineers

BSIM3SOI
Level=25 Model Released in SmartSpice

Introduction

The Berkeley BSIM3SOI model, released in December 1997, is now available within **SmartSpice** as the MOSFET level=25 model. This model incorporates three separate implementations: the original Berkeley model implementation is invoked with the selector Berk=2; the Silvaco implementation is invoked with Berk=-2. The older Silvaco implementation, of October 1997, is also supported, under the selector Berk=-1.

All implementations produce virtually identical results when commonly accepted model parameter sets are used. However, the Silvaco Berk=-2 implementation supports a number of additional parameters and options, and provides certain improvements compared to what is supported in the Berk=2 (Berkeley) and Berk =-1 (older Silvaco) Level=25 models.

Physical Effects

Major Features

The major features of the BSIM3SOI model, as excerpted from the BSIM3SOIv1.3 Manual (copyright 1998 UC Berkeley), are as follows:

- Dynamic depletion selector (ddMod) to suit different requirements for SOI technologies
- Dynamic depletion approach is applied on both I-V and C-V. Charge and Drain currents are scalable with Tbox and Tsi continuously
- Single I-V expression as in BSIM3v3.1 to guarantee continuities of Ids, gds and gm and their derivatives for all bias conditions
- Supports external body bias and backgate bias; a total of 6 nodes
- Real floating body simulation in both I-V and C-V. Body potential is properly bounded by diode and C-V formulation
- Self heating implementation improved over the alpha version
- An improved impact ionization current model

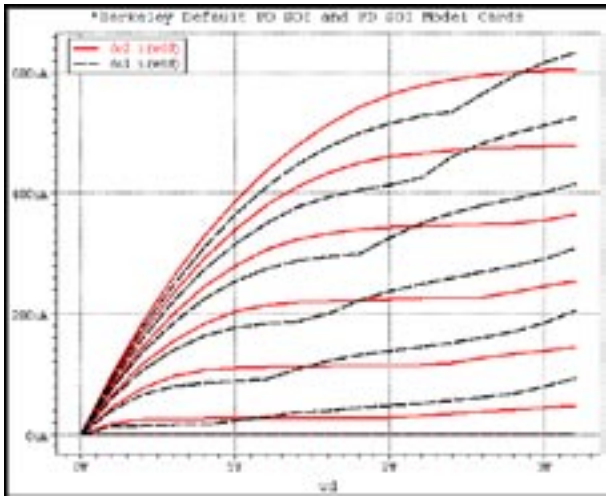

Figure 1. BSIM3SOI Level=25 partially and fully depleted models with floating body: dc1.i represents PD model and dc2.i represent FD model.

- Various diode leakage components and parasitic bipolar current included
- New depletion charge model (EBCIO) introduced for better accuracy in capacitive coupling prediction. An improved BSIM3v3 based model is added as well

Impact Ionization Current Partitioning

Continued on page 2....

INSIDE

SILVACO INTERNATIONAL

Simulation results produced by S-Pisces show that the partitioning of the impact ionization current between source and drain significantly depends on the channel lengths.

In long channel devices 100% of the impact ionization current flows into the bulk. In short channel devices the entire impact ionization current flows into the source.

The IIRAT model parameter, implemented in **SmartSpice,** can be used to direct a certain portion of the substrate current to the source. If IIRAT=0 then the entire impact ionization current will be directed to the bulk of the device (for long channel devices). If IIRAT=1 then this current will be directed to the source, as in Berkeley's default model (for short channel devices). The binning parameters LIIRAT, WIIRAT and PIIRAT can be used to adjust the actual IIRAT value depending on the geometry of a particular device.

The IIRAT parameter was implemented as follows: when the IIRAT parameter is explicitly specified in the .MODEL card the values of the model parameters ALPHA0, ALPHA1 and BETA0 will be ignored. These values will instead be calculated automatically to provide a desired level of the substrate current partitioning between the bulk and source of the device.

Silvaco Improvements

Impact Ionization Current

In the Berkeley SOI model implementation the impact ionization current Isub depends on the model parameters ALPHA0 and BETA0. The current Isub represents a current directed from the drain to the bulk.

Simulation results produced by the device simulator Atlas show that the partitioning of the impact ionization current between bulk and source significantly depends on the channel lengths.

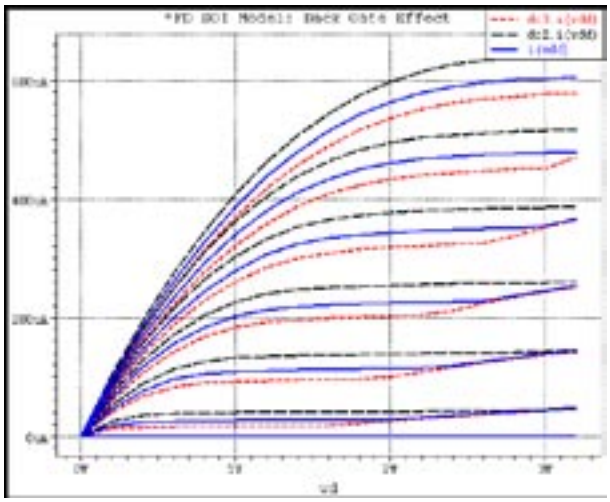

Figure 2. Back-Gate effects for FD model with floating body: dc1.i is Ids current for Vbg=0, dc2.i is Ids current for Vbg=3V and dc3.i is Ids current for Vbg=-3V.

In long channel devices 100% of the impact ionization current flows into the bulk. In short channel devices the entire impact ionization current flows into the source.

In the Silvaco Berk=-2 model implementation the impact ionization current directed from the drain to the source is modeled as follows:

```
Iscbe = Idsa * PSCBE2*diffVd/Leff *
exp(-PSCBE1* Litl/diffVd)
```

where

Idsa is the total drain current, including the CLM and DIBL effect currents;

```
diffVd = Vds - Vdsat;
```

PSCBE2 and PSCBE1 are the model parameters;

Litl is a bias independent parameter computed as a function of TOX and XJ.

3.2 Intrinsic Capacitance Model CAPMOD=0

The Berkeley SOI model implementation supports the intrinsic capacitance models CAPMOD=2 and 3. The Silvaco Berk=-2 model implementation also supports the CAPMOD=0 intrinsic capacitance model. It provides the best convergence and performance of the three. The most complicated model is CAPMOD=3; this Berkeley model has the worst convergence and performance of the three. The Berkeley CAPMOD=2 capacitance model has better convergence and performance than the CAPMOD=3 model, but these are still much worse than those obtained with the CAPMOD=0 model.

The Vfbcv model selector can be used with the CAPMOD=0 capacitance model: the recommended value is INTCAP=1. If this value is specified, then the flat band voltage will be calculated in the same manner as for CAPMOD=2. The INTCAP selector will be ignored if the VFBCV parameter is explicitly specified in the model card.

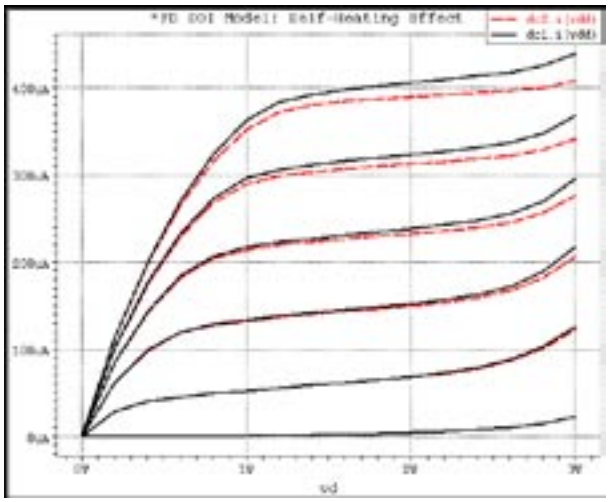

Figure 3. Self-Heating effects: dc1.i represents curves without self-heating and dc2.i represents curves with self-heating turned on.

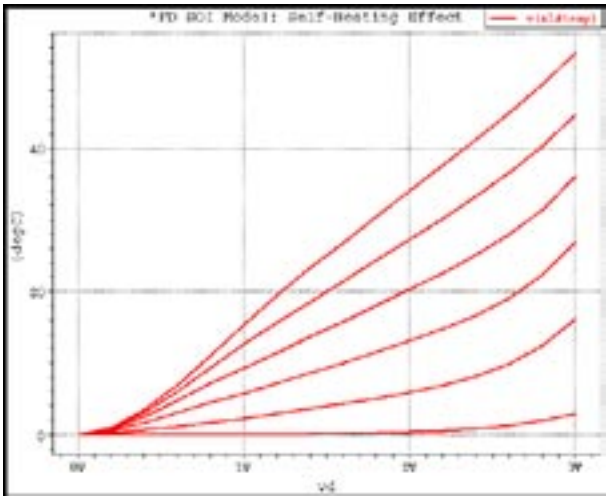

Figure 4. Self-Heating effects with v(m1#temp) represents device temperature.

Summary of Silvaco's Unique Improvements

- Added ACM equations for parasitics
- Improved impact ionization current model
- Improved self-heating model
- Added CAPMOD=0 to improve convergence and physical modeling of intrinsic capacitances
- Fully compatible to Berkeley
- Convergence properties
  - dramatic improvement
- Model performance in CPU time:
  - dramatic (10 to 20 times for large circuits)
- **EXPERT** mode - unique for model problem diagnostics
- Binning is available (for L, W and WL dependances)

Binning Parameters

A number of binning parameters were re-implemented in the Level=25 Berk=-2 model. The binning procedure is supported for the following model parameters:

VHT0, RDSW, U0, KETA, K2, UA, PSCBE1 and PSCBE2.

and is identical to that used in the BSIM3 v3.1 MOSFET Level=8 model.

Geometry Parameter Calculation

In the Berkeley implementation, the geometry parameters NRD, NRS, AD,PD, AS, PS can only be specified as device parameters. In the Silvaco Berk=-2 model implementation, the geometry parameters NRD, NRS, AD,PD, AS, PS can also be calculated as functions of Weff and the following model parameters:

LD	0.0	"m"	"Length difference"
LDIF	0.0	"m"	"Lateral diffusion"
HDIF	0.0	"m"	"Heavy doped region length"
RS	0.0	"Ohm"	"Source resistance"
RD	0.0	"Ohm"	"Drain resistance"
RSC	0.0	"Ohm"	"Contact source resistance"
RDC	0.0	"Ohm"	"Contact drain resistance"

References

[1] BSIM3SOI v1.0 Manual, Department of Electrical Engineering and Computer Sciences, University of California, Berkeley

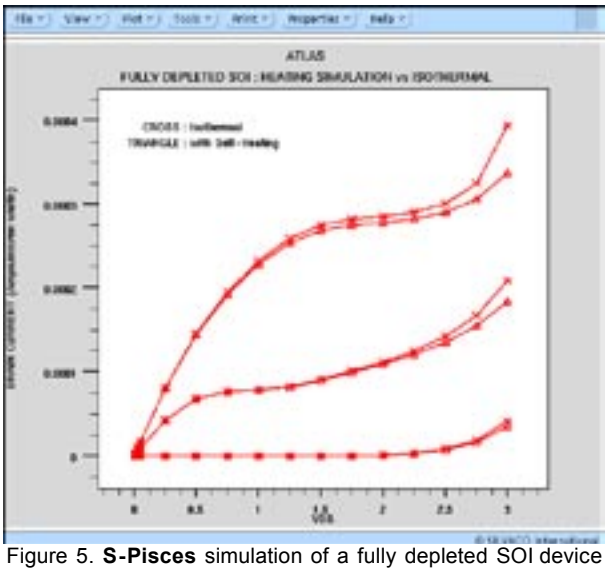

Figure 5. **S-Pisces** simulation of a fully depleted SOI device that demonstrates heating effects confirming the physical nature of the SOI model.

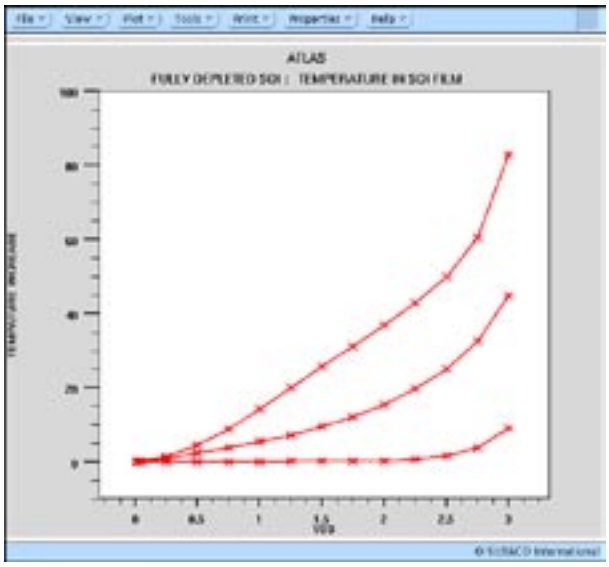

Figure 6. **S-Pisces** simulation confirming the thermal dependencies of an SOI device.

Release of an Upgraded SmartSpice Interface to Cadence

The ***SmartSpice*** Interface to Cadence integrates the Analog Artist and Composer elements of the Cadence Design Framework II (DFII) with ***SmartSpice*** **.** This integration is accomplished, in versions 4.4.0 and later of DFII, through the Cadence Spice Socket (cdsSpice) and the OASIS interface in the Analog Artist and Composer components of DFII. Versions of DFII prior to 4.4.0 are also supported by SmartSpice, but these solutions rely on the older HSPICE Socket, and necessarily offer substantially less functionality than is provided by the current interface. This article will therefore concentrate on the OASIS/ Spice Socket version of the ***SmartSpice*** interface, first summarizing the development done to date, before going on to describe some significant enhancements which will be incorporated into the next release of the interface.

The ***SmartSpice*** interface is distributed as the Silvaco product "ssi_cds", with the version currently standing at 1.0.9.R. The interface can be installed into one or more existing platform-dependent installations of DFII with the command:

```
smartspice -oasis -install
```

The most obvious result of performing this installation will be seen when the Cadence Command Interpreter Window (CIW) is subsequently started up, as it will now contain (among other things) the line "Loading SmartSpice.cxt". The "Simulator" menu of Analog Artist's Simulator/Directory/Host control screen will now include "**SmartSpice**" among its available simulator modes; selecting this mode will enable the ***SmartSpice*** interface for that session of Artist. This means that Artist will now generate netlists in SmartSpice format, run ***SmartSpice*** transparently when required, and obtain the appropriate PSF output from ***SmartSpice*** necessary to enable all of the standard Artist/ Composer back-annotation and cross-probing functionality.

In order to take advantage of the ***SmartSpice*** -specific features now enabled, each user of Analog Artist will need to reference the Silvaco-supplied versions of the Cadence 'basic' and 'analogLib' component libraries. This is easily achieved by running the command:

```
smartspice -artist -configure
```

which will create a file called 'cds.lib.1.0.9.R' in the user's current working directory (assuming version 1.0.9.R of ssi_cds), the contents of which should be included in the user's 'cds.lib' file.

In the January 1998 edition of the *Simulation Standard*, the main capabilities of the current release of the

Figure 1. The new dc analysis dialog box.

SmartSpice interface were described in some detail; these will now be repeated here in summary:

- Ability to specify ***SmartSpice*** as the default simulator in the Setup->Simulator/Directory/Host control screen of Analog Artist.
- Ability to include model files in ***SmartSpice*** or cdsSpice format in the Setup->Environment control screen of Analog Artist.
- Ability to generate PSF output from ***SmartSpice*** **,** implemented through automatic generation of the "psf=2" option.
- Annotation of node voltages to the Composer schematic editor, selected via the Results->Annotate->DC Node Voltages menu item in Analog Artist.
- Annotation of device operating points (for example, device currents, gds, etc.) to the Composer schematic editor, selected via the Results->Annotate->DC Operating Points menu item in Analog Artist, and controlled via the opPointLabelSet field of the Interpreted Labels Information section of the CDF properties in the Silvaco-supplied analogLib library, accessed via the Tools->CDF->Edit control screen in the Cadence CIW.
- Support for marching waveforms, implemented via the ***SmartSpice*** waveform viewer through automatic generation of the ".IPLOT" statement.
- Direct plot of waveforms in the Cadence Waveform Window, implemented via the Results->Direct Plot menu in Analog Artist, combined with user selection in the Composer schematic editor.

- Hierarchical netlisting capability, implemented through **SmartSpice**/Spice Socket name mapping routines in the OASIS interface.
- New Silvaco-supplied versions of the Cadence basic and analogLib libraries, incorporating **SmartSpice** views of all appropriate devices.
- Compatibility with ac, dc, transient and noise analyses, via the Choosing Analyses control screen in Analog Artist.
- Ability to save bias points in dc and transient analyses, via the Choosing Analysis control screen in Analog Artist.
- Ability to configure the following ***SmartSpice*** options from within the Simulator Options control screen in Analog Artist:

  ABSTOL, ACCT, ACCURATE, ACM, AUTOSTOP, BYPASS, CAPDC, CAPMOD, CAPTAB, CHGTOL, COEF1, CONV, DCGMCHK, DCGMIN, DCGMSTEPS, DCIAP, DCPATH, DEFAD, DEFAS, DEFL, DEFPD, DEFPS, DEFNRD, DEFNRS, DEFW, DISTRIBUTION, EXPERT, FORMAT, GMIN, GMINSTEPS, HDIF, ICG, INTEGR, INTERP, ITL1, ITL2, ITL4, ITL41, ITL5, LD, LDIF, LIMPTS, LIST, LOGIC, METHOD, NODE, NOMOD, NOPAGE, NUMDGT, OPTS, PIVREL, PIVTOL, RAWPTS, RELTOL, SCALE, SCALM, SRCSTEPS, TEMP, TNOM, TRTOL, TRYTOCOMPACT, TTICK, VNTOL, VSTA, VZERO, WIDTH

Since the release of version 1.0.9.R, some more substantial enhancements have been made to the ***SmartSpice*** interface. The most important, and immediately noticeable improvement has taken place in the Analog Artist user interface. In order to provide enhanced control over **SmartSpice** simulations, the analysis dialogs within Analog Artist have been redesigned to incorporate many more useful and requested ***SmartSpice*** features than are available in the current release.

The transient analysis dialog has been extended to support the following features:

- Specification of either single time point syntax (via the "TSTEP", "TSTOP" and "TMAX" keywords) or multiple time point syntax.
- Addition of several methods of operating point calculation, enabling the specification of initial conditions for circuits with multiple stable operating points (via the "UIC" option), the restoring and saving of operating points (the "CALLV"/"SAVEV" options) and support for simplifying the calculation of operating points for circuits with no path to ground (the "TRANOP" option).
- Support for the "STORE" and "RAWPTS" options.

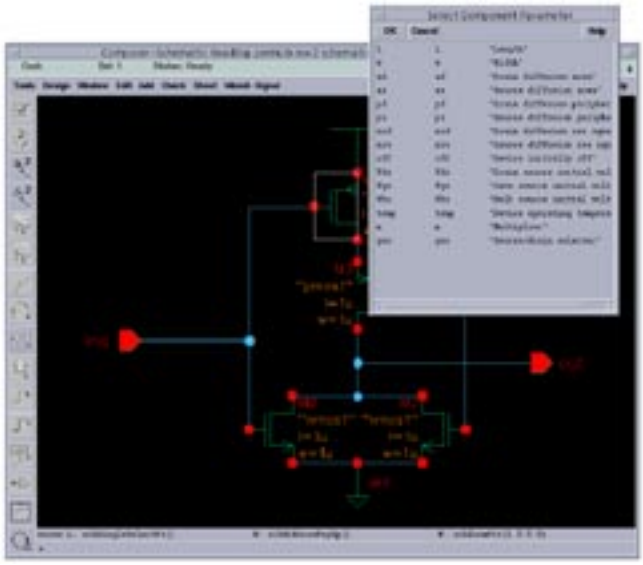

Figure 2. Source selection schematic.

The AC and DC analysis dialogs have also received new attention:

- Addition of the "UIC" and "CALLV"/"SAVEV" operating point options.
- Addition of nested sweeps, providing the ability to sweep sources, component parameters, model parameters or temperature beneath the primary analysis, and allowing for selections to made directly on the schematic.

The new DC analysis dialog is illustrated in figure 1. Both new analyses take advantage of two useful features of the OASIS interface: all parameter values are automatically validated before being incorporated into input decks, and certain types of parameter (eg. sources, component parameters, etc.) can be selected directly from the schematic (see Figure 2).

Another important aspect of the forthcoming release of the ***SmartSpice*** interface is entirely hidden from the end user, but will result in a reduction in the development cycle for future releases. In the current version the input deck for ***SmartSpice*** is actually generated by the Cadence Spice Socket (cdsSpice), under control from Analog Artist. Any changes made to Analog Artist (through the OASIS interface) must also be reflected in cdsSpice. Therefore installation of a new version of the ***SmartSpice*** interface also requires a new version of cdsSpice. We are now able to take advantage of a recent enhancement to the OASIS interface allowing most of the input deck generation to be moved from cdsSpice to Analog Artist, thus minimizing the need for new versions of cdsSpice in the future.

This new version of the ***SmartSpice*** interface will be labeled 1.0.10.R, and will be available shortly.

Advanced Cell Characterization Using SmartSpice Scripting Features

Introduction

In a previous article [1], the efficient use of the ***SmartSpice*** `.MODIF` statement for cell characterization was discussed. This article will focus on using advanced features of the ***SmartSpice*** scripting language to solve this problem in a more flexible manner.

Unlike most commercial simulators, ***SmartSpice*** provides the user with a powerful command interface which can be utilized in input decks in the form of command scripts [2]. These commands can be used to pre-process the input deck, altering parameters and running multiple simulations, and they can also be used to perform post-processing in the form of measurements, plotting, etc.

Variables

Variables can be manipulated in a script as in most other scripting languages. A variable can be created in a number of ways, but the most frequently used method is via the *set* command. For example, to create a new variable called foo, the following command could be used.

```
% set foo = 1.0
```

The value assigned to a variable can be numeric, string, boolean or a list. To access the value stored in a variable, the standard *$var* notation is used. It is possible to determine if a variable has been created using the special *$?var* notation. If the variable exists, then this evaluates to true, otherwise it evaluates to false.

Measurements and Variables

The `.MEASURE` statement or *measure* command will both create variables to store the value of the measurement, as a side-effect. The variable will only store the last valid measurement. The name of the variable is the same as that of the measurement. For example,

```
.MEASURE tran rise_time
+ trig v(en) val='pvdd/2' rise=1
+ targ v(q) val='pvdd/2' rise=1
```

will create a variable called *rise_time* when the measurement is executed during simulation. This variable can then be accessed via the command-line or a script using the standard notation.

```
echo "The rise time =" $rise_time
```

It is possible for a measurement to fail, especially during cell characterization. When this happens the corresponding variable will not be created and the command script can check for the existence of the variable. The return value of this operation could then be used in conditional constructs as in the following example.

```
unset rise_time
run
if ~$?rise_time
    echo "Failed to measure rise time"
else
    echo "The rise time =" $rise_time
end
```

The ~ operator is used as a logical not in the previous example script. The *unset* command can be used to ensure that a variable does not exist prior to a run. This can occur if multiple simulations are being executed from within the script.

Variable Expressions

In common with most scripting languages, ***SmartSpice*** does not explicitly support variable expressions. This inconvenience can be bypassed using the measure command. For instance, given a variable *foo,* a new variable *bar* can be created as a function of *foo* using the following *measure* command.

```
measure bar param='foo*2.5+3'
echo "foo =" $foo "bar =" $bar
```

This is the only situation in which a variable is not referenced using the *$var* notation.

Iterative Commands

During cell characterization, many simulations can be run, with parameters being varied between each simulation. Typically this iterative process is controlled using nested sweeps, or the `.MODIF` statement in ***SmartSpice.*** If a command script is being used to control this iterative process in ***SmartSpice,*** then a number of different looping constructs can be used.

The simplest of these statements is the *repeat* construct. This will execute the body of the loop a specified number of times. For instance, the following `.MODIF` statement could be used in an input deck to repeat a simulation 31 times, incrementing the parameter label *tin* by 0.1ns each iteration.

```
.MODIF LOOP=31
+ tin += 0.1n
```

This could be replicated using the following *repeat* block.

```
repeat 31
    modif loop=1 tin += 0.1n
end
```

The *modif* command must be used within the repeat loop to run the simulation, since this is the only method available to communicate changes in parameters to the simulation engine from the script. Otherwise a standard *run* command would be used.

While this example demonstrates no obvious advantage over the .MODIF implementation, other features of the scripting language can then be used within the loop to provide more flexibility as required. For instance, it is possible to exit from the repeat block using the *break* command, or perform true nested sweeps of multiple variables, as shown in the following examples.

```
repeat 31
   unset max_q
   modif loop=1 tin += 0.1n
   if $max_q < 1.6
     break
   end
end
```

This example will repeat the simulation for a maximum of 31 iterations, but will break out of the loop when the value of the measurement, *max_q*, is less than 1.6.

```
foreach ttime 0.5n 1n 2n 5n
  set fout = 1
  repeat 10
     modif loop=1 sr={$ttime} sf={$ttime}
\
        fan_out={$fout}
    measure fout param='fout+1'
  end
end
```

This example demonstrates the use of a nested set of loops using a *repeat* block nested within the foreach block. The *foreach* loop is useful for looping over lists of irregularly spaced values. A total of 40 simulations will be executed by this section of code. The *measure* command is used to increment the value of the *fout* variable within the repeat block. This is necessary since *fout* is used to vary the circuit parameter label *fan_out* in the inner loop, while the outer loop varies the rise and fall times of the input pulses. As with the previous examples, further conditional statements can also be used to reduce the number of simulations, exit loops, change increments, etc as required.

Setup Time Calculation

In [1], an example to calculate the setup time of a latch using a .MODIF statement with conditional stops was given. This example used a binary search type algorithm to minimize the number of iterations required to compute the setup time. The .MODIF code necessary to accomplish this task is given below.

```
.MODIF LOOP=4 STOP max_q LE 1.6
+ tin += (7n) 1n
+MODIF LOOP=2 STOP 1.6 LE max_q
+ tin -= 0.5n
+MODIF LOOP=2 STOP max_q LE 1.6
+ tin += 0.25n
+MODIF LOOP=3 STOP 1.6 LE max_q
+ tin -= 0.1n
```

This algorithm can also be coded using the scripting language, with greater flexibility. A script that performs an equivalent characterization to that shown above, is as follows.

```
foreach increment 1n -0.5n 0.25n -0.1n
  repeat
    modif loop=1 tin += $increment \
       print_measures = 0 iterations +=
1
     if ( $increment > 0 and $max_q < 1.6
) \
        or ( $increment < 0 and $max_q >
1.6 )
      break
   end
  end
 end

  echo "    setup time =" $setup >
setup.tim
```

As can be seen two nested loops are used in this example. The *foreach* loop controls the size of the increment, while the *repeat* loop will execute until the *if* command's conditions are met. At the end of the *foreach* loop, the last simulation will have calculated the setup time to the required accuracy (0.1ns) and this value is then printed to the file "setup.tim".

The advantage which this approach can have over ***SmartSpice's*** standard .MODIF statement, is that the criteria used to determine the failure of the latch are not confined to functions of the current simulation. For instance in this example, the maximum output voltage is used as the criteria for success/failure. Within the script, this can easily be extended to be a function of prior measurements as can sometimes be required.

Conclusion

This article has focussed on the use of the **SmartSpice** scripting language and its flexibility when applied to the problem of efficient cell characterization. The language itself provides the user with the ability to encapsulate within each input deck functionality which in other simulators can only be achieved through the use of external shell like scripting languages such as Perl, sh, awk and sed. In conjunction with unique ***SmartSpice*** statements for parametric, worst-case and monte carlo type analyses, this combination provides the user with unparalleled flexibility, functionality and speed.

References

[1] "Cell Characterization using the .MODIF Statement in SmartSpice", Simulation Standard, Vol. 9, No. 1, pp 12-13, January 1998.

[2] "Advanced SmartSpice Command Functionality", TCAD Driven CAD, Vol. 9, No. 7, pp 5-6, July 1997.

UTMOST Log File Conversion for Tonyplot

A new feature has recently been added to ***UTMOST***: the ability to convert ***UTMOST*** log files into a format suitable for viewing in ***TonyPlot***. This new feature is controlled from the ***UTMOST*** Output Log File screen, and is compatible with all ***UTMOST*** plot types. The conversion process will also automatically produce the derivative data types associated with certain ***UTMOST*** routines (such as the mosfet module's ALL_DC, or the bipolar module's BF and BR).

The Output Log File screen is illustrated in Figure 1. To perform a log file conversion, the user types the name of the ***UTMOST*** logfile (eg. 'logfile') into the input field 'Source File 1', and then presses the '**TonyPlot**' button. This will result in the creation of two new files with additional suffixes '.sff' and '.set'. These files should be supplied to ***TonyPlot*** with a command line of the form:

```
tonyplot logfile.ssf -set logfile.set
```

An example of applying this procedure to data generated from a single device with the mosfet ALL_DC routine is shown in Figure 2. In this figure we can see both the 'IDvsVDS' and 'IDvsVGS' curves;

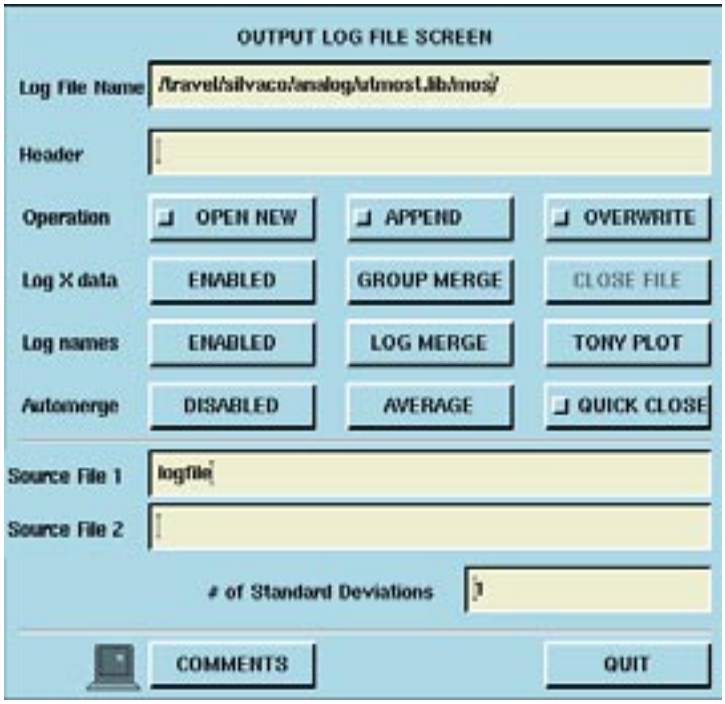

Figure 1. ***UTMOST*** log file screen.

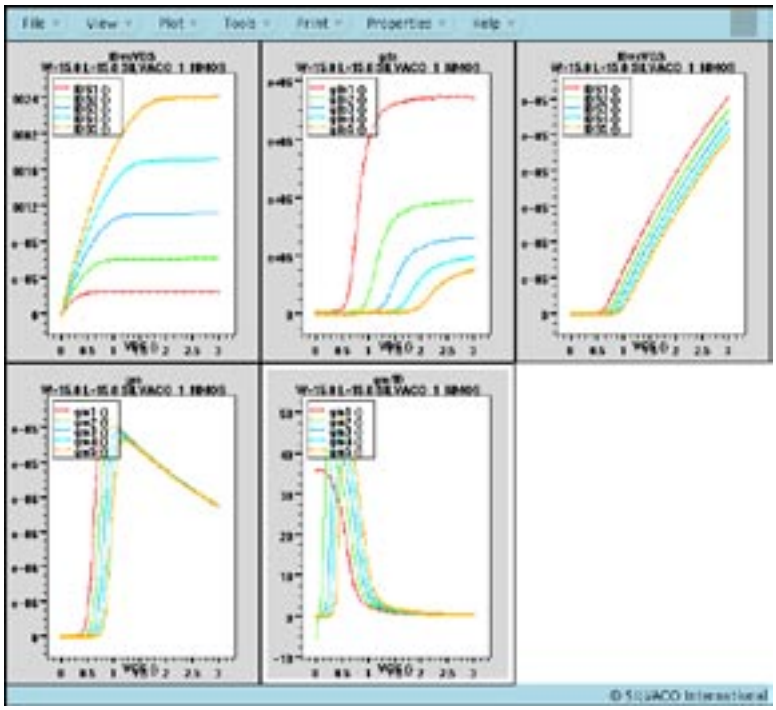

Figure 2. Example of converted log file viewed in ***TonyPlot***.

we also see the 'gds', 'gm' and 'gm/ID' curves, all of which are generated automatically during the conversion process. (In fact, whether the convertor produces a 'gds' or an 'rds' derivative depends in this case on the value of the ***UTMOST*** DC Measurement screen's 'gds/rds' flag.)

This new feature will be available in ***UTMOST*** version 12.3.14.R, and all ***UTMOST*** distributions will now include the latest version of ***TonyPlot***. Once inside ***TonyPlot*** the plots obtained in this manner can be further manipulated at will, to take advantage of the full functionality of **TonyPlot**, enabling the user to select individual curves, zoom in or out, alternate between linear and log scales or simply transform the data through user-defined functions.

Release of RPI Amorphous Silicon and Polysilicon TFT Models in *SmartSpice* and *UTMOST*

Introduction

Thin film transistors (TFTs) have an important application in the manufacture of active matrix LCD displays. As this technology has become more mature, a number of different models of both amorphous silicon (a-Si) and polysilicon TFTs have been proposed. Recently two new models developed by the Rensselaer Polytechnic Institute (RPI) have been implemented in the ***SmartSpice*** circuit simulator. These models are also now available in the **TFT** module of ***UTMOST III*** and this article will discuss the different model characteristics, and their use in both ***SmartSpice*** and ***UTMOST III.***

Description

Due to the lack of commercial quality **TFT** models, designers typically approximate the behaviour of the **TFT** device using standard MOSFET models. With decreasing device geometries this approach is no longer suitable. While **TFT** devices can essentially be thought of as MOSFETs without a substrate contact, i.e. a three terminal device, the most significant physical difference occurs in the band structure of the thin film.

In crystalline silicon the structure has well defined conductance and valence bands. In both the a-Si and polysilicon lattices, the bandgap is populated with localized trap states. These trap states are due to the non-periodic nature of the lattice in a-Si and in the case of polysilicon due to trap states produced at the boundaries of each crystalline grain. Some of the physical effects affected by the presence of localized trap states are the field-effect mobility, subthreshold and leakage currents and the frequency dispersion of capacitance.

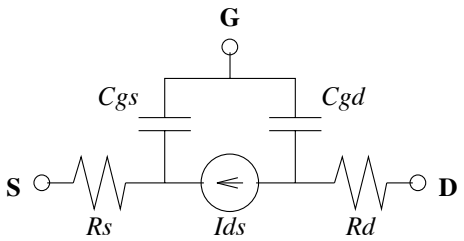

Figure 1. Equivalent circuit for the a-Si TFT model

Amorphous TFT Model

The a-Si **TFT** models is available in ***SmartSpice*** as a LEVEL=35 MOSFET model. Each device has three terminals, drain, gate and source. The bulk node is not present. The model parameters that are specific to the a-Si TFT model are given in Table 1.

The equivalent circuit for this model is given in Figure 1. As can be seen the circuit is similar to that of a basic MOSFET, with the bulk node and its associated elements removed. The Ids current has a number of regions, leakage, subthreshold and above threshold. The leakage current is modeled empirically at large negative biases and is generally quite small.

In the subthreshold regime, most of the electron carriers are trapped in energy states, and as a consequence, the sheet electron concentration can be related to material parameters and the density of these states. This relationship is then used to derive the subthreshold current.

Above threshold, the sheet electron concentration is given by the modified charge control model. At threshold, carriers induced in the thin film are still trapped in the bandgap traps and the current flowing in the device

Parameter	Symbol	Description	Parameter	Symbol	Description
VTO [V]	V_{T0}	Threshold Voltage	VFB [V]	V_{FB}	Flat-band Voltage
GAMMA	γ	Power Law Mobility Parameter	V0 [V]	V_0	Characteristic Voltage of Deep States (Optimized)
VAA [V]	V_{AA}	Characteristic Voltage for μ_{FET}	IOL [A]	I_{OL}	Zero Bias Leakage Current
ALPHASAT	α_{SAT}	Saturation Parameter	VDSL [V]	V_{DSL}	Vds Leakage Dependence
LAMBDA [V^{-1}]	λ	CLM Parameter	VGL [V]	V_{GL}	Vds Leakage Dependence
MSAT	m_{SAT}	Knee Shape Parameter	SIGMA0 [A]	σ_0	Minimum Current
MUBAND	μ_n	Band Mobility: 0.001m^2 /V/s	EPSI	ε_i	Relative Permittivity of Gate Insulator: 7.4
G0	g_0	Midgap DOS: 1.0 x 10^{23} m^{-3} eV^{-1}	EPS	ε_s	Relative Permittivity of Amorphous Silicon: 11
DEF0	dE_{F0}	Dark Fermi Level Position: 0.6 eV	NC	N_C	Effective Conduction Band Band DOS: 3x10^{25} m^{-3}

Table 1 : Parameters specific to the a-Si TFT model.

will still be relatively small. As the free charge increases with increasing Vgs, the device will eventually turn on, but this Von will be greater than the threshold voltage Vt. This is not the case with standard MOSFET models. This effect results in a gradual transition between the exponential and linear regions. This effect is taken into account in the above threshold current model, by manipulating the field effect mobility.

Polysilicon TFT Model

The polysilicon TFT model is available in **SmartSpice** as a LEVEL=36 MOSFET model. Each device has three terminals, drain, gate and source. The bulk node is not present. The model parameters that are specific to the polysilicon **TFT** model are given in Table 2.

The equivalent circuit for this model is given in Figure 2. As can be seen this model differs from the equivalent circuit for the a-Si device in that two resistors are added in series with the gate-source and gate-drain capacitances. These resistors are used to account for dispersion of capacitances with frequency, their value is a function of the channel resistance.

The polysilicon **TFT** model differs from the a-Si model in that a larger leakage current can exist. This current is a function of the thermionic field emission of carriers through grain boundary trap states. The leakage current a function of temperature, Vfb and the terminal voltages. It is independent of Vt and L. The expression for the leakage current also accounts for drain induced barrier lowering.

In the subthreshold region, the drain current can be effectively modeled using standard MOSFET theory. Above threshold, the current is subject to same effects as in the a-Si case, as discussed previously. In addition, trap states cause the kink effect. This effect is seen at high drain biases with the device biased in saturation.

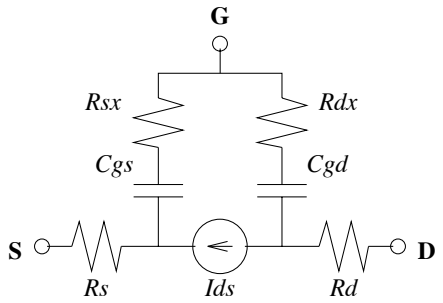

Figure 2. Equivalent circuit for the polysilicon TFT model

Device Characterization

The **TFT** module of **UTMOST III** has been modified to support both of these models. Models for both a-Si and polysilicon devices have been extracted and the operation of the models verified against the measured data. Subsequently transient simulation using **SmartSpice** can been used to verify the effectiveness and accuracy of the I-V and C-V models. Typically a very good match exists between the measured and simulated results.

Acknowledgements

References

[1] Michael S. Shur, H. C. Slade, et al., "Modeling and scaling of a-Si: H and Poly-Si Thin Film transistors", MRS Spring Meeting, San Francisco, March 31-April 4, 1997.

[2] Michael S. Shur, Mark D. Jacunski, et al., "SPICE models for amorphous silicon and polysilicon thin film transistors", Elec. Chem. Soc. Proc., Vol 96-23, pp 242-259, 1996.

Parameter	Symbol	Description	Parameter	Symbol	Description
VTO [V]	V_{TO}	Long Channel Threshold Voltage	DG [m]	d_G	Drain Electric Field Parameter*
ETAI	ηi	Subthreshold Ideality Factor	DD [m]	d_D	Gate Electric Field Parameter*
ALPHASAT	α_{SAT}	Saturation Parameter	BLK	B_{lk}	Leakage DIBL Parameter*
MUS [cm^2/V/s]	μ_S	Subthreshold Mobility	I0 [A/m]	I_0	TFE Leakage Coeff.*
MUO [cm^2/V/s]	μ_0	High Field Mobility	I00 [A/m]	I_{00}	Diode Leakage Coeff.*
MMU	m	Mobility Exponent	LKINK	L_{kink}	Kink Length Coeff.*
MUI [cm^2/V/s	μ_I	Low Field Mobility Coefficient	MKINK	M_{kink}	Feedback Exponent*
VFB [V]	V_{FB}	Flat Band Voltage*	VKINK	$V_{\cdot kink}$	Electric Field Parameter*

* Optimized Only

Table 2 . Parameters specific to the polysilicon TFT model.

SPICE Model Validation
(part II)

Introduction

The importance of the MOS device SPICE model validation and the introduction of the validation routine in ***UTMOST III*** was presented in *Simulation Standard* article issued on September 1996. The recent developments and the practical applications for the "Validate" routine will be presented in this article.

The validate routine allows users to verify the measured versus simulated parameters such as VSAT (Saturation Voltage), ISAT (Saturation Current) and VTH (Threshold Voltage) for the range of L-array or W-array devices. Different combination of the L-array devices or W-array devices or parameters can be selected using the flags in the fit variable screen of the "Validate" routine.

Data Acquisition

The Validate routine can only read measurement data from BSIM3_MG and ALL_DC routines. The Validate routine can not be used for measurement. The measurement variables in the measurement setup screen is used for simulation only.

Measurement Setup

Measurement Variables (Used with simulation only)

1	VGS_sat	Constant Gate voltage for Isat and Vsat plots.
2	VBSstart_sat	Starting value of the bulk voltage for Isat and Vsat plots.
3	VBSstep_sat	Step value of the bulk voltage for Isat and Vsat plots.
4	#of_VBS_step	Number of substrate bias points for Isat and Vsat plots.
5	VDS_VTH	Constant drain voltage for Vth vs L/W/Vbs plots.
6	VBSstart_VTH	Starting value for the bulk voltage of Vth vs L, W or Vbs plots.
7	VBSstep_VTH	Step value of the bulk voltage for Vth vs L or W plots.
8	#of_VBS_step	Defines a number of substrate bias points for Vth vs L or W plots.
9	VBS_VTD	Constant bulk voltage for Vth vs Vds plot.
10	VDSstart_VTD	Starting value of the drain voltage for Vth vs Vds plot.
11	VDSstep_VTD	Step value for the drain voltage for Vth vs Vds plot.
12	#of_VDS_step	Number of drain bias points for the Vth vs Vds plot.
13	VBSstep_VBS	Step value for the bulk voltage for Vth vs Vbs plot.
14	#of_VBS_step	Number of bulk bias points for the Vth vs Vbs plot.
15	SAT_L_group	Strategy screen Group number for Isat or Vsat vs L plots.
16	SAT_W_group	Strategy screen Group number for Isat/Vsat vs W plots.
17	VTH_L_group	Strategy screen Group number for Vth vs L plot.
18	VTH_W_group	Strategy screen Group number for Vth vs W plot.
19	VTH_VD_group	Strategy screen Group number for Vth vs Vds plot.
20	VTH_VB_group	Strategy screen Group number for Vth vs Vbs number.

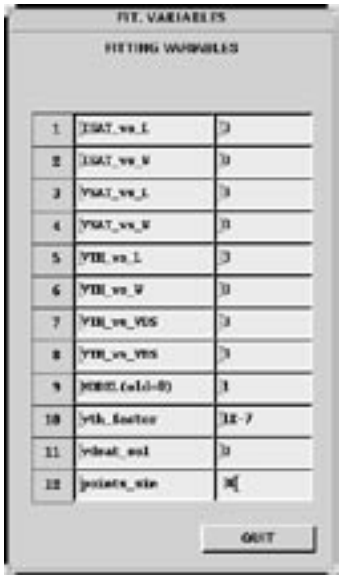

Figure 1. The Fitting Variable Screen for the Validate Routine.

Figure 2: The Validate Measurement Setup Screen

Fitting Variables

The fitting variables allow the user to select the type of data that will be displayed in the graphics screen and define the method used for Vth extraction.

The fitting variables (from 1 to 8) can take one of the 4 possible values:

0 No plot
1 Measurement data only
2 Simulation data only
3 Measurement + Simulation data

* ISAT_vs_L: Flag for Isat vs L plot.
* ISAT_vs_W: Flag for Isat vs W plot.
* VSAT_vs_L: Flag for Vsat vs L plot.
* VSAT_vs_W: Flag for Vsat vs W plot.
* VTH_vs_L: Flag for Vth vs L plot.
* VTH_vs_W: Flag for Vth vs W plot.
* VTH_vs_VDS: Flag for Vth vs Vds plot.
* VTH_vs_VBS: Flag for Vth vs Vbs plot.
* Model (Old=0): Model Selector for Vth extraction (0 - 5).
* Vth_factor: Vth factor value for threshold voltage extraction using a constant current method.
* vdsat_sel: Drain voltage used for the direct drain current extraction.
* points_sim: Number of simulation points used when"Get Raw Data from" is set to "Simulation" and all plot flags in the fit variable screen are set to 2. Used to improve the extraction accuracy.

Parameter Extraction

The parameters Vsat, Isat and Vth are extracted using the methods described in Table 1.

Vsat/Isat Extraction

The saturation voltage and the saturation current (if Model is set to 0, 1 or 4) are extracted from the last sweep of the Ids/Vds - Vbs curves. These parameters are the coordinates of the intersection between the slope of the linear region and the minimum slope of the saturation region.

If Model is set to 1, 2 or 5, the saturation current is directly extracted as the drain current for the drain voltage set with the vdsat_sel fitting variable. The saturation voltage is always extracted from the intersection between the slope of the linear region and the minimum slope of the saturation region.

VTh Extraction

Three methods of threshold voltage extraction are available within the Validate routine.

If Model is set to 0 or 2, Vth is extracted from the linear regression fit with the maximum slope using the IDS versus VGS curve at zero substrate bias voltage (maximum slope method).

If Model is set to 1 or 3, Vth is the Vgs voltage when the Ids current is equal to the current defined as: vth_factor * Wdrawn/Ldrawn. The Vth_factor is set by the user in the Fit Variable Screen.

If Model is set to 4 or 5, Vth is extracted from the largest slope of the sqrt(Ids) versus Vgs curve when the transistor is in saturation mode (high Vds). If the Vds value is too low, a warning message will be displayed on the ***UTMOST*** screen.

Isat or Vsat vs Ldrawn or Wdrawn data

If the measured data is available (coming from BSIM3 or ALL_DC routines) the validate routine will sort the devices with constant W and varying L or constant L and varying W. The Isat or Vsat values for each device will be extracted based on the methodology described earlier. The Isat vs L or W data will be displayed for different VBS values. The simulation data will also use the same device selection and Isat calculation methods.

Vth vs Ldrawn or Vth vs Wdrawn data

Based on the flag selection in the fit variable screen the validate routine will extract and display the VTH versus L-array or W-array curves. The VTH extraction will be performed at different VBS bias conditions. The VTH extraction method is defined by the "Model" flag in the fit variable screen.

Model		Vth Extraction Method	Isat Extraction Method
0	Maximum Slope of Ids versus Vgs curve: GMmax.	Intercept between linear slope and the minimum saturation slope on ID/VD - VG: Lin. Sat. Inter	
1	Constant current : Vg(Id= <vth_factor> W/L) where <vth_factor> is the value of the fitting variable vth_factor.	Intercept between linear slope and the minimum saturation slope on ID/VD - VG: Lin. Sat. Inter.	
2	Maximum Slope of Ids versus Vgs curve: GMmax	Drain current for Vds = <vdsat_sel> where <vdsat_sel> is the value of the fitting variable vdsat_sel: Id(Vd)=<vdsat_sel>	
3	Constant current : Vg(Id= <vth_factor> W/L) where <vth_factor> is the value of the fitting variable vth_factor.	Drain current for Vds = <vdsat_sel> where <vdsat_sel> is the value of the fitting variable vdsat_sel : Id(Vd=<vdsat_sel>	
4	Maximum Slope of sqrt(Ids) versus Vgs curve: Sqrt(Id)	Intercept between linear slope and the minimum saturation slope on ID/VD - VG: Lin. Sat. Inter.	
5	Maximum Slope of sqrt(Ids) versus Vgs curve: Sqrt(Id)	Drain current for Vds = <vdsat_sel> where <vdsat_sel> is the value of the fitting variable vdsat_sel : Id(Vd=<vdsat_sel>	

Table 1. Vth and ISAT Extraction Methodolgies.

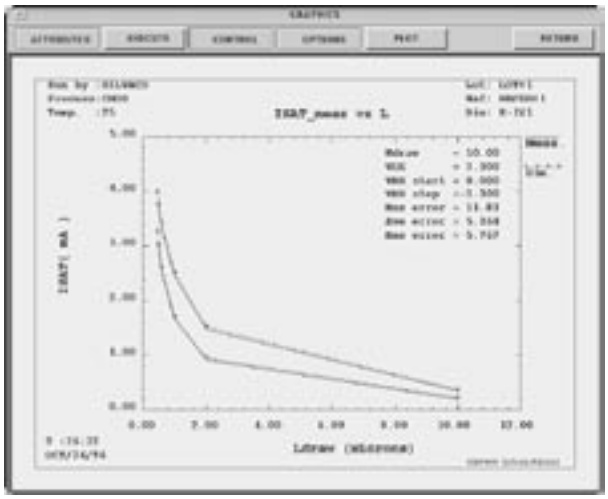

Figure 3 : Isat versus Ldrawn.

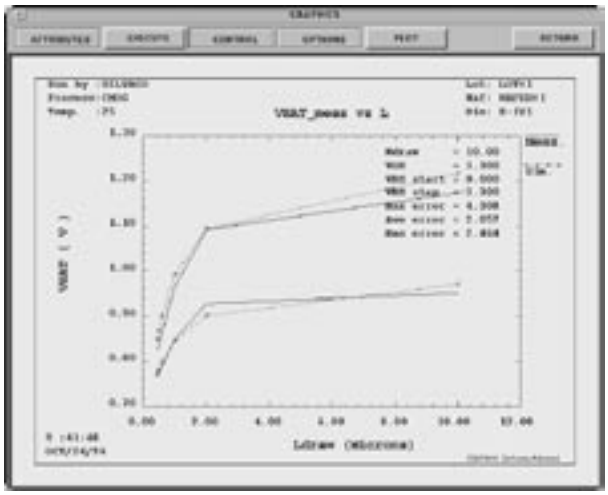

Figure 4 : Vsat versus Ldrawn

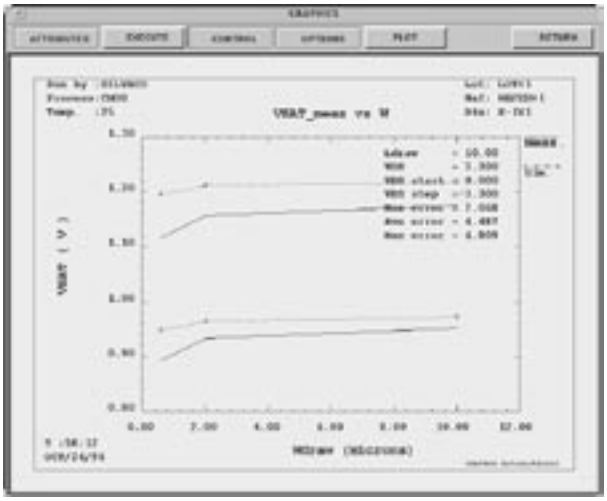

Figure 5. Vth versus Ldrawn.

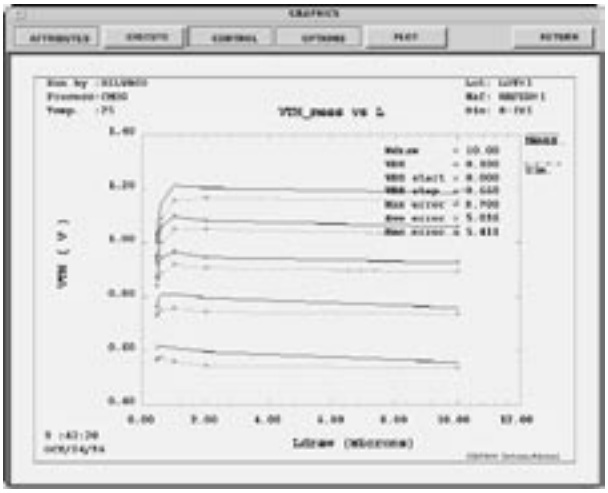

Figure 6. Vth versus VDS.

Vth vs VDS data

The validation of the threshold voltage variations with Vds is performed on the L-array devices only. The Vth extraction is performed at Vbs=0V bias condition.

Vth vs VBS data

The validation of the threshold voltage variations with Vbs is performed on the L-array devices only. The Vth extraction is performed for single Vds bias condition.

Simulation

Simulation data can be used for model validation in two different ways.

If measured data is available, simulation data can be obtained for the same bias points used for the measured data. The measured and simulated data will be superimposed on all validation curves.

A standalone simulation option (without the measured data) can be utilized to verify the continuity of the model for different L-array or W-array devices. This option requires the user to set all fit variable flags to 2. and set the correct group numbers in the measurement setup screen. The accuracy of the actraction for the simulation only option is defined by the "points_sim" flag.

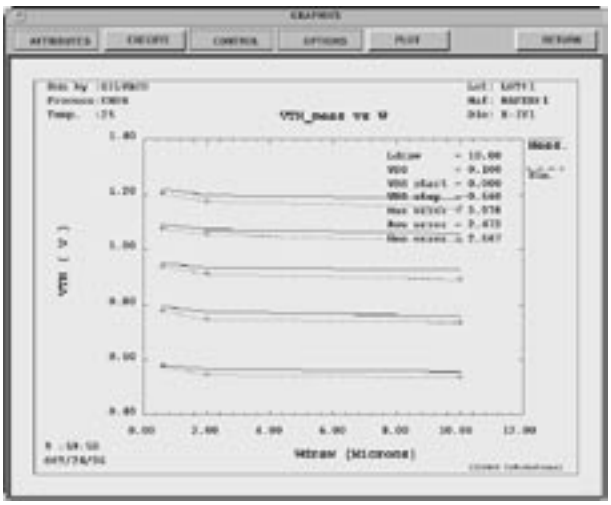

Figure 7.Vth versus VBS.

Calendar of Events

April

1
2
3
4
5
6
7 W/S - Guildford, UK
8
9
10
11
12
13
14
15
16 W/S - Munich, Germany
17
18
19
20
21 W/S - Scottsdale, AZ
22 W/S - Scottsdale, AZ
23
24
25
26
27
28 W/S - Grenoble, France
29
30

May

1
2
3
4
5
6
7
8
9
10
11
12 CICC - Santa Clara, CA
13 CICC - Santa Clara, CA
W/S - Guildford, UK
14
15
16
17
18
19
20 W/S - Guildford, UK
21
22
23
24
25
26
27 W/S - Scottsdale, AZ
W/S - Munich, Germany
28 W/S - Scottsdale, AZ
30
31

Bulletin Board

First Ever Parallel SmartSpice on NT and Linux

At the DAC '98 tradeshow, Silvaco has released parallel (multi-threaded) version of **SmartSpice** for NT computers. The speed performance improvement is the same as for Unix class of computers. This truly open a new era of parallel circuit simulations on multi-CPU computers. The demo stations set-up at the COMPAQ booth and Silvaco booth have drawn large crowds of excited circuit design engineers.

New SOI and TFT Models Released in SmartSpice

In response to a growing need for high quality circuit models for SOI and TFT technologies, Silvaco has released several new models. Level=25 is a new and improved SOI model suitable for very large circuit designs. New TFT model from RPI comes as a single, physical model suitable for simulating both amorphous and poly TFT devices.

Boston Office Expands

The East Coast economy has substantially recovered so the decision has been made to refocus on this region. Mr Micheal Ridinger is now Applications Manager for this region, Patrick Coyne is the sales manager an Joanne Miller is the office manager. The current plan is to purchase large 4 acre land in Chelmsford and build an 18,000 sq. feet office building in March of next year.

For more information on any of our workshops, please check our web site at ***http://www.silvaco.com***

Hints, Tips and Solutions

Mustafa Taner, Applications and Support Engineer

Q. What is the typical method of measuring flicker noise using S3245A Noise Amplifier?

A TYPICAL METHOD OF NOISE MEASURMENT

A. A typical noise measurement setup and the measurement equipment used for noise measurements were explained in the previous issue of the Simulation Standard.

In order to measure the flicker noise **UTMOST III** user needs to have a S3245A Noise Amplifier, DC Analyzer, A Dynamic Signal Analyzer (The list of available Dynamic Signal Analyzers were listed in the previous issue) and **UTMOST III** MOS module.

Prior to the noise measurements or modeling the DC characteristics of the MOS device should be measured. The MOS DC model should be extracted using **UTMOST III** local or global optimization techniques. The MOS model which is used for DC modeling should include the noise model which is intended for noise parameter extraction. The extracted DC model parameters should be present in the **UTMOST III** parameter screen.

After the DC model is extracted the Dynamic Signal Analyzer's measurement setup should be defined. A typical hardware settings for noise measurement is presented in Figure1.

The "Run Setup" button should be pressed to load the setup parameters diplayed in the Hardware screen into the Dynamic Signal Analyzer. In order to compensate the noise Amplifier's own noise. The "System Calibration" should be executed prior to the actual flicker noise measurements.

The System Calibrate button is located in the hardware configuration screen for the selected Dynamic Signal Analyzer. During the calibration the S3245A Noise amplifier should be switched on but all other cables should be disconnected from the amplifier. The measured noise level of the amplifier will be stored in **UTMOST III**and later on it will be subtracted automatically from the measured flicker noise data.

The S3245A Noise Amplifier, DC analyzer and Dynamic Signal Analyzer connection diagram was presented in the previous issue. After making the proper connections the DC bias conditions should be set using the "Measurement Setup" screen for the "Noise" rouitne in **UTMOST III.** The **UTMOST III** user should press the "Measure" button in the extraction screen to start the device DC biasing and subsequent flicker noise measurement. The DC analyzer will apply the specified DC voltage to the gate, drain, source and bulk terminals of the MOS device. However, there is a series resistor connected to the drain terminal of the S3245A Noise Amplifier. This series resistor will force the DC analyzer to supply higher external voltage to maintain the specified drain voltage at the device terminal. The noise rouitne in **UTMOST III** will iterate the external voltage automatically until the proper value is found. The speed of these voltage iterations is limited by the filters connected to the device terminals in the S3245A Noise Amplifier. Therefore the iteration process may take some time. However once the correct external VDS voltage is found the no_bias flag in the fit variable screen can be set to 1 to by-pass the DC voltage iteration process.

After the DC bias is established, **UTMOST III** will control the Dynamic Signal Analyzer to measure the flicker noise. The measured noise spectra will be transfered to **UTMOST III** and the data will be displayed in the graphics screen for parameter extraction.

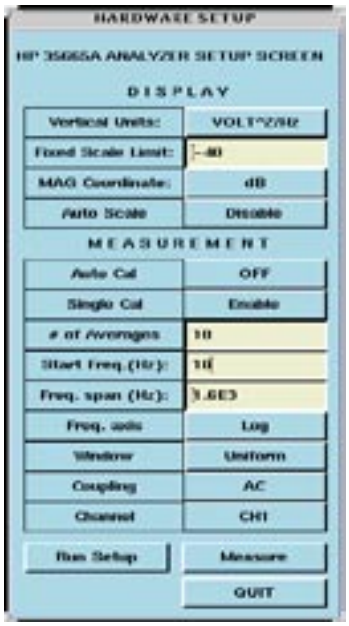

Figure1. A typical settings for a Dynamic Signal Analyzer used for flicker noise measurements.

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

http:://www.silvaco.com

Join the Winning Team!

Standardize your process/device and CAD design using Silvaco's

"TCAD Driven CAD™"

To get a demo and product description contact or visit a Silvaco office near you:

- Santa Clara
- Phoenix
- Austin
- Boston
- Guildford
- Grenoble
- Munich
- Tokyo
- Seoul
- Hsinchu

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*