Simulation Standard

Connecting TCAD To Tapeout

A Journal for Circuit Simulation and SPICE Modeling Engineers

IBIS Models in SmartSpice

1. Introduction

The Input/Output Buffer Information Specification (IBIS) is a standard for electronic behavioral models based on I/V and V/T curve data. It is being developed by the IBIS Open Forum, which is affiliated with the Electronics Industry Alliance (EIA). These models are suitable for high-speed designs of digital systems to evaluate Signal Integrity issues (deformation of electronic signals, cross-talk, power/ground bounce, transmission lines...) on printed circuit boards (PCBs).

The IBIS standard offers a way to provide fast and accurate models of I/O buffers without divulgating any proprietary technology process. As it protects IP, it is now widely used by semiconductor vendors as a replacement for SPICE netlists. The IBIS standard specifies only what kind of information is provided, how this information is presented in ASCII files and how some data are derived from measurements or simulations. How these data are used and processed by a simulator is not part of the standard. The purpose of this documentation is to present the IBIS model support in ***SmartSpice.***

The reader who is not familiar with the IBIS standard or would like to learn more about IBIS may refer to the Web site of the IBIS Open Forum at *"http://www.eigroup.org/ibis"* where numerous documents are available for download, including introductions, slide shows, articles and complete specifications (from the initial v1.0 to the latest v4.1 of January 2004).

2. IBIS Buffer Equivalent Circuit

A buffer is implemented as a new element in ***SmartSpice.*** Even though different types of buffers are available to cover a wide range of functions and technologies, they are all base on the same equivalent circuit, which is shown on Figure 1.

Several elements and terminals are optional depending on the buffer type: Input (node IN) and Enable (node EN) high-impedance inputs, Output (node OUT) voltage source and conductance, Pullup/Pulldown (nodes PU/PD) Voltage-Controlled Voltage Sources (VCVS). Only Input/Output buffers have all elements and all terminals available. The number of terminals and the description of the equivalent circuit are given in Table 1 for all buffers currently supported in ***SmartSpice.***

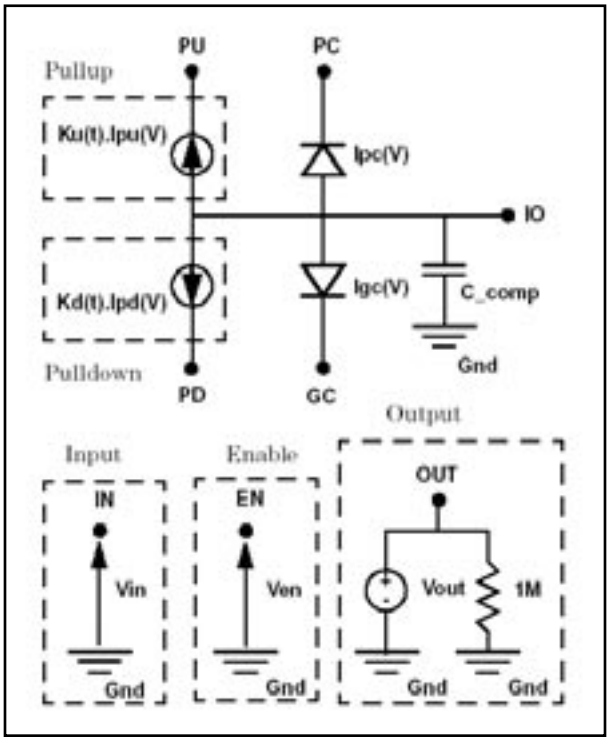

Figure 1. IBIS Buffer General Circuit Diagram.

Continued on page 2 ...

INSIDE

SILVACO INTERNATIONAL

Type/Number	Terminals (min/max)	Input	Enable	Output	Pullup	Pulldown
input/1	4	no	no	yes	no	no
output/2	4/6	yes	no	no	yes	yes
input_output/3	6/8	yes	yes	yes	yes	yes
three_state/4	5/7	yes	yes	no	yes	yes
open_drain/5	4/6	yes	no	no	no	yes
io_open_drain/6	6/8	yes	yes	yes	no	yes
open_sink/7	4/6	yes	no	no	no	yes
io_open_sink/8	6/8	yes	yes	yes	no	yes
open_source/9	4/6	yes	no	no	yes	no
io_open_source/10	6/8	yes	yes	yes	yes	no
input_ecl/11	4	no	no	yes	no	no
output_ecl/12	3/5	yes	no	no	yes	yes
io_ecl/13	5/7	yes	yes	yes	yes	yes
three_state_ecl/14	4/6	yes	yes	no	yes	yes

Table 1. Description of the equivalent circuit for all buffer types.

The Gnd node corresponds to the SPICE ground node, also called node 0. All connections to Gnd are internal (C_comp, Vout...) and so this node is not available as a terminal. The die capacitance C_comp specified in IBIS models is usually connected between IO node and ground. However, if die capacitances C_comp_pc, C_comp_gc, C_comp_pu and C_comp_pd are specified in the IBIS model instead of C_comp, these four capacitances are connected between IO and PC, GC, PU and PD nodes, respectively.

PC/PU and GC/PD terminals are usually supposed to connect to power and ground rails, respectively. By default they are connected to internal voltage sources (not shown on the circuit diagram) and should not be connected to any other elements in the netlist (especially voltage sources). However the instance parameter **power** may be used to allow connections to external elements or power supplies.

3. IBIS Buffer Device Line

Using buffers in ***SmartSpice*** is identical to using other elements like passive or semiconductor devices. The general syntax of a buffer statement is given by:

Bname term1 term2 term3 [term4 [term5 [term6 [term7 [term8]]]]]

+ file = 'filename' model = 'modelname'

+ [typ = {typ|min|max|fast|slow}]

+ [power = {on|off}]

+ [interpol = {1|2}]

+ [buffer = {number|type}]

+ [ramp_rwf] = {0|1|2}

+ [ramp_fwf] = {0|1|2}

+ [fwf_tune = value] [rwf_tune = value]

+ [c_comp_pc = value]

+ [c_comp_gc = value]

+ [c_comp_pu = value]

+ [c_comp_pd = value]

Device naming convention

The buffer element name must begin with B followed by optional alphanumeric characters.

Terminals

The number and the order of terminals specified on the device line are type-dependent:

B_input PC GC IO OUT (Input and Input_ECL)

B_output PU PD IO IN [PC [GC]] (Ouput, Open_drain, Open_sink, Open_source)

B_three_state PU PD IO IN EN [PC [GC]] (Three_state)

B_input_output PU PD IO IN EN OUT [PC [GC]] (Input_output, IO_open_drain, IO_open_sink and IO_open_source)

B_output_ecl PU IO IN [PC [GC]] (Output_ecl)

B_io_ecl PU IO IN EN OUT [PC [GC]] (IO_ecl)

B_three_state PU IO IN EN [PC [GC]] (Three_state_ecl)

Open_drain, IO_open_drain, Open_sink and IO_open_sink buffers have no pullup circuitry but PU terminal must be specified even though not connected to internal elements. Open_source and IO_open_source buffers have no pulldown circuitry but PD terminal must be specified even though not connected to internal elements.Ouput_ecl, Three_state_ecl and IO_ecl buffers have pullup and pulldown circuitry but no PD terminal because this latter node is internally connected to PU node.

Required Parameters

file and **model** are required parameters to define the location of the .ibs file containing the IBIS model for this buffer. In SmartSpice these parameters are also used to decide if a B statement corresponds to a MESFET device or to an IBIS buffer. See Backward Compatibility paragraph below for further details.

- 'filename' is case-sensitive and must correspond either to the absolute path to the .ibs file or the relative path in respect to the directory where SmartSpice is run or to the directories specified by the option 'd_ibis'. See Options paragraph below for a description of this new option
- 'modelname' is case-sensitive and must match one of the models in the .ibs file

Optional Parameters

typ must be set to select what column of all IBIS data will be used during the simulation: TYP (default), MIN, MAX, SLOW or FAST. If FAST or SLOW are specified, the column MIN or MAX is selected depending on the IBIS parameter as defined in table 2. This is especially useful for best case / worst case analysis. If min or max values are not available in the IBIS model for a given parameter, typ values are used.

IBIS Parameter/Data	Fast	Slow
C_comp	min	max
C_comp_pc	min	max
C_comp_gc	min	max
C_comp_pu	min	max
C_comp_pd	min	max
Voltage_range	max	min
Pullup_reference	max	min
Pulldown_reference	min	max
Power_clamp_reference	max	min
Gnd_clamp_reference	min	max
Pulldown	max	min
Pullup	max	min
Gnd_clamp	max	min
Power_clamp	max	min
Ramp	max	min
Rising_waveform	max	min
Falling_waveform	max	min
V_fixture	max	min

Table 2. Min/Max combinations for Slow/Fast conditions.

power is used to select how the buffer is powered via PC, GC, PU and PD nodes (if these latter nodes exist for the given buffer type).

- If **power** is set to 'on' (default), these nodes are internally connected to voltage sources whose values are taken from the IBIS parameters: [POWER Clamp Reference], [GND Clamp Reference], [Pullup Reference], [Pulldown Reference] (or [Voltage Range] if preceding parameters are missing). For this case, terminal names specified on the element card may be usefull to print out the voltage values if needed
- If **power** is set to 'off', internal voltage sources are not created and PC, GC, PU and PD nodes must connect to external voltage sources either directly or through passive devices like RLC networks or transmission lines.

interpol is the interpolation method selector.

- If **interpol** is set to 1 (default), I/V curves are interpolated using linear interpolation
- If **interpol** is set to 2, quadratic bi-spline interpolation is used. This latter method is usually not recommended and useless anyway if IBIS data are accurate

buffer is used to specify the type of the buffer. This value overrides the corresponding IBIS parameter Model_type. It is usually not recommended to specify a different value. Integer values are allowed to select a buffer. The correspondence with literal names is given in table 1.

ramp_fwf and **ramp_rwf** selectors allow the user to choose the calculation method of multipliers Ku(t) and Kd(t). These parameters are totally independent and may have different values.

- If **ramp_fwf** (or **ramp_rwf**) is set to 0 (default), only the ramp data is used to derive multipliers for the falling (or rising) transition
- If **ramp_fwf** (or **ramp_rwf**) is set to 1, the first falling (or rising) waveform table available in IBIS model is used to derive corresponding multipliers
- If **ramp_fwf** (or **ramp_rwf**) is set to 2, the first two falling (or rising) waveform tables available in IBIS model are used to derive corresponding multipliers

These latter option is highly recommended to get accurate results in transient analysis. However, if the required data are not available in IBIS model, the value of **ramp_fwf** (or **ramp_rwf**) is decremented and a warning message is issued. For example, if **ramp_fwf** = 2 and only one waveform table is given, then **ramp_fwf** is set to 1, if **ramp_fwf =1** and only ramp data are given, then **ramp_fwf** is set to 0.

fwf_tune and **rwf_tune** factors are control parameters for **ramp_fwf** =0, 1 and **ramp_rwf** = 0, 1 algorithms, respectively. When only ramp data or one waveform is available, it is necessary to impose an additional condition to compute multipliers. Usually it is assumed that Ku(t)+K(t)=1, which was demonstrated to be not realistic because the circuitry that goes from ON to OFF undergoes this transition faster than the circuitry that goes from OFF to ON.

By setting **fwf_tune** or **rwf_tune** to a value between 0 and 1 (default 0.1), it is possible to get more accurate transitions by using the following assumption: if deltaT is the duration of a complete transition, the multiplier K(t) corresponding to the circuitry that goes from ON to OFF decreases linearly from 1 to 0 between t=0 and **t=fwf_tune** * deltaT (or **t=rwf_tune** * deltaT depending on the transition). Thus, the other multiplier is uniquely determined from an IBIS ramp or one IBIS waveform. The multiplier computation methods are described in articles (1, 2).

C_comp_pc, C_comp_gc, C_comp_pu and **C_comp_pd** are dimensionless die capacitance partionioning factors. They do not override the IBIS parameters with the same names, which correspond to actual die capacitances. If these latter capacitances are specified in the IBIS model, the dimensionless factors are useless and ignored if given on the element card. If only C_comp is available in the IBIS model, it may be desirable to split it into several parts for simulating power/ground bounce. This is achieved by specifying the fractions of C_comp connected between IO node and PC, GC, PU, PD nodes. If given, the values of instance parameters **C_comp_pc**, **C_comp_gc**, **C_comp_pu** and **C_comp_pd** should be between 0 (default) and 1. It is also expected that their sum equals 1.

4. Buffer Logical State

The logical state of a buffer is controlled by the voltage of IO, IN and/or EN nodes relative to ground and noted Vio, Vin and Ven, respectively.

For buffers with no controlling signals (no IN or EN nodes), the state is a function of Vio, the IBIS parameters Vin_l, Vin_h (thresholds), Polarity and the previous state if any.

If Polarity=Non-Inverting

- Initially (t=0 in transient analysis or first computed point of a DC sweep), state is set to LOW if Vio<(Vin_h+Vin_l)/2 or to HIGH in the opposite case
- If state=HIGH then it goes to LOW only if Vio<Vin_l
- If state=LOW then it goes to HIGH only if Vio>Vin_h

else Polarity=Inverting

- Initially (t=0 in transient analysis or first computed point of a DC sweep), state is set to LOW if Vio>(Vin_h+Vin_l)/2 or to HIGH in the opposite case
- If state=HIGH then it goes to LOW only if Vio>Vin_h
- If state=LOW then it goes to HIGH only if Vio<Vin_l

For buffers with only one controlling signal (IN node), the state is a function of Vin, the IBIS parameter: Polarity and the previous state if any. Here thresholds are constant built-in parameters

If Polarity=Non-Inverting

- Initially (t=0 in transient analysis or first computed point of a DC sweep), state is set to HIGH if Vin>0.5 or to LOW in the opposite case
- If state=HIGH then it goes to LOW only if Vin<0.2
- If state=LOW then it goes to HIGH only if Vin>0.8

else Polarity=Inverting

- Initially (t=0 in transient analysis or first computed point of a DC sweep), state is set to HIGH if Vin<0.5 or to LOW in the opposite case
- If state=HIGH then it goes to LOW only if Vin>0.8
- If state=LOW then it goes to HIGH only if Vin<0.2

For buffers with two controlling signals (IN and EN nodes), the state is a function of Ven, Vin, Vio, the IBIS parameters: Vin_l, Vin_h (thresholds), Polarity, Enable and the previous state if any. The enable signal Ven supersedes the input signal Vin and is used to determine whether the buffer is in ENABLE or DISABLE state:

If Enable=Active-High

- Initially (t=0 in transient analysis or first computed point of a DC sweep), buffer is ENABLE if Ven>0.5 or DISABLE in the opposite case
- If buffer=ENABLE then it goes to DISABLE only if Ven<0.2
- If buffer=DISABLE then it goes to ENABLE only if Ven>0.8

else Enable=Active-Low

- Initially (t=0 in transient analysis or first computed point of a DC sweep), buffer is ENABLE if Ven<0.5 or DISABLE in the opposite case
- If buffer=ENABLE then it goes to DISABLE only if Ven>0.8
- If buffer=DISABLE then it goes to ENABLE only if Ven<0.2

If a buffer is ENABLE, the state is controlled by Vin according to the rules defined above for buffers with only one controlling signal (IN node).

If a buffer is DISABLE, there are two possible behaviors depending on the type:

- For buffers without output circuitry (no OUT node), the state is just locked till the buffer returns to ENABLE. Three-state buffers belong to this family
- For buffers with output circuitry (OUT node), the state is controlled by Vio according to the rules defined above for the buffers with no controlling signals. Input-output buffers belong to this family

The logical state can be printed out if the output circuitry (OUT node) is available:

If state=HIGH then Vout=1.0V. If state=LOW then Vout=0.0V.

OUT node can also connect to external elements, especially IN or EN nodes of other buffers. This nodes offer a simple way to create complex digital blocks in SPICE netlists.

5. Output Variables

The variables listed in table 3 can be printed out using the SmartSpice syntax @B_name[variable_name].

6. IBIS-related Options

GMIN/DCGMIN conductances are connected in parallel with PC and GC diodes and with PU and PD Voltage-Controlled Voltage Sources (if they exist for the buffer type) to ensure better convergence of buffer devices in particular situations.

A new option d_ibis has been added to specify the location of .ibs files. Several paths can be specified. A .ibs file will be searched in all specified paths if the filename given on B statements is not an absolute path and is not found in the directory from which ***SmartSpice*** runs. This option is case-sensitive. For example:

```
.option d_ibis='/home/mylogin/myIbisModels'
```

Variable name	Definition
ku	Pullup transient current multiplier
kd	Pulldown transient current multiplier
cio	Input/Output terminal current
cpc	Power Clamp terminal current
cgc	Ground Clamp terminal current
cpu	PullUp terminal current
cpd	PullDown terminal current
cin	Input terminal current
cen	Enable terminal current
cout	Output terminal current

Table 3. Buffer internal variables.

d_ib is also available as a variable and can be set in SmartSpice .ini files. For example:

```
set d_ibis = ( . /home/mylogin/myibismodels )
```

7. Backward Compatibility

In previous releases of ***SmartSpice***, B statements were only used to define instances of MESFET models (as an alias of Z). From now on, they may also be used to define IBIS buffers. When a B statement is encountered in a netlist, ***SmartSpice*** first checks whether the IBIS-specific parameters file and model are specified in the element card. As these parameters are required to create an IBIS buffer, ***SmartSpice*** creates a MESFET device if they are missing, so that backward compatibility is maintained.

8. Limitations

- Only DC, Transient and AC analysis are supported for IBIS buffers
- The ***SmartSpice*** VZERO=2 option is not supported for IBIS buffers
- Unlike other simulators, the IBIS Golden Parser is not incorporated into ***SmartSpice*** yet. As a consequence SmartSpice does not check the syntax of .ibs files and just issues a generic 'parse error' message if the syntax of an .ibs file is not in compliance with IBIS v3.2 specifications. To avoid such problems all .ibs files should be systematically verified with the Golden Parser, freely available as an executable on the IBIS Open Forum web site. If the Golden Parser reports warnings and errors, the .ibs file can probably not be used in SmartSpice netlists
- ***SmartSpice*** buffers correspond to IBIS [Model] descriptions in .ibs files and so do not account for packages, which are defined in [Component] descriptions. However it is possible to add manually equivalent networks in the netlist (made of passive devices R_pkg, C_pkg and L_pkg whose values are taken from .ibs files)
- The series, series switch and terminator buffers are not supported in this beta-release. They are currently being implemented
- The capability for ***SmartSpice*** to use [Component] descriptions is also under development

9. References

[1] Peivand F. Tchrani, Yuzhe Chen, Jiayuan Fang, "Extraction of transient behavioral model of digital I/O buffers from IBIS", *46th IEEE Electronic Components&Technology Conference*, Orlando, May 28-31, 1996, pp 1009-1015

[2] Ying Wang, Han Ngee Tan, "The development of analog SPICE behavioral model based on IBIS model", *Ninth Great Lakes Symposium on VLSI*, pp.101-104, 1999

New Features in SmartSpice 2.3.4.C

Stop-Continue Feature

A powerful new feature "Stop-Continue" has been added to ***SmartSpice*** to allow the user to suspend a transient simulation and investigate the output before resuming the simulation run from the suspended state. This allows generated data checks during the simulation run and therefore ensuring relevant simulation data is generated. This feature allows a user to check intermediate simulation results and/or save (if necessary) on the fly. The user can allocate CPU priority to the most important simulation job with a unix system command such as "nice" if multiple simulation jobs are running on the same machine and if a user wants to finish one job as soon as possible.

The Stop-Continue algorithm is activated by adding the ".OPTOINS STOPCONT" line in the input deck to be activated and works in a command mode(smartspice -c) or a command line on New ***SmartSpice*** GUI with some SmartSpice commands:

1. pause: pauses a current simulation.
2. cont: resumes a current paused simulation
3. cancel: stops current simulation.

Note: The command to continue a suspended process is not "continue" but "cont".

The keyword "continue" is reserved for other purpose.

The following message is displayed on the ***SmartSpice*** main window after sourcing an input deck and running it:

--> run

STOP-CONTINUE MECHANISM PROGRESSES...

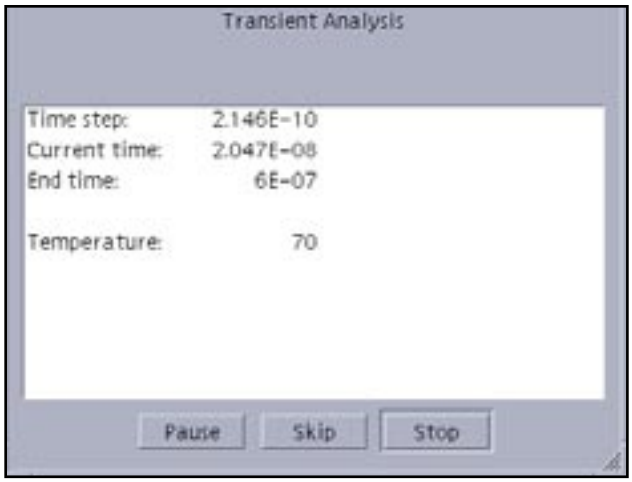

Figure 1. Run time screen with Pause button.

Once the feature is turned on, a user can interactively operate pause/continue/stop using the above three SmartSpice commands, or Pause and Stop buttons on the SmartSpice run time screen(Fig.2). Clicking on Pause and Stop buttons behave as pause and cancel commands, respectively.

In the case of Stop, a ***SmartSpice*** simulation job is immediately terminated and cannot be resumed with the Pause button (it's just an interactive stop operation in ***SmartSpice*** GUI).

The run time window disappears when the Pause button is pressed and the interactive ***SmartSpice*** window is released for a user to do various kinds of post processing operations as if the simulation was completed. A user can now check or save an intermediate simulation result and after the check is done, to resume the simulation, type a command 'cont' (Figure 2) and the run time dialog will be displayed on the screen again.

Note : Not all ***SmartSpice*** commands are available. Some commands cannot be issued in the Stop-Continue algorithm including in conjunction with a multi-thread mode as described below, and such commands can be checked by issuing a special help command "mt_notsafe_com".

The option ".OPTIONS STOPCONT" also enables ***SmartSpice*** to split a transient simulation job into two modes - foreground and background operations. This means that a simulation job is running in a background mode and post-processing, such as .print or .measure processes in a foreground. This foreground operation is also available for a command line (Figure 2) or interactive ***SmartSpice*** GUI operations without a Pause command as described above.

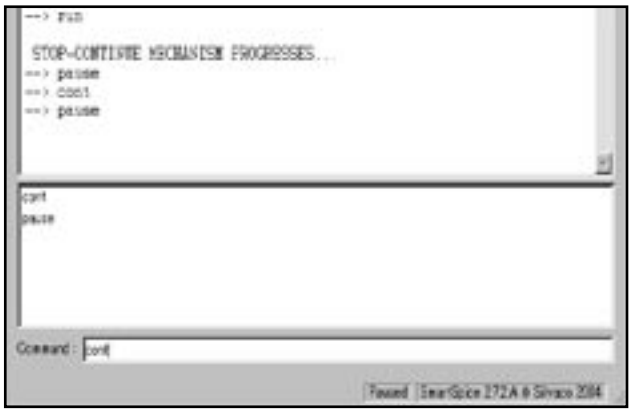

Figure 2. SmartSpice command line Windows.

The option enables not only the Stop-Continue feature but also the foreground operation on the fly but we recommend a user to pause a simulation when he or she wants to operate the post-processing. The reason is that the CPU sometimes might be loaded for such post-processing operations and the simulation itself might result in taking more time than the user expects.

This Stop-Continue feature can also function in a multi-thread mode, but due to the same reason, one extra processor is recommended to allocate post-processing, e.g. 4 CPU machine, for "smartspice -P 2 or 3" and one remaining processor can be used for the post-processing in this case.

A New SmartSpice Variable for Safe Mode

Nowadays, high performance computing systems are available at a very low price, and even a server class PC environment is recently replacing conventional and expensive EWSs by a high cost performance PC systems equipped with a huge hard drive and physical memory, such as Windows and Linux platforms.

However, we could also say that some application software consume a lot of system resources at the same time, and such software might sometimes cause the system to crash when they are simultaneously executed on a same machine.

SmartSpice supports a new feature "Safe Mode" to guard your simulation job from such unexpected risk.

The features is turned on by adding one ***SmartSpice*** variable "set safemode=true" into your .SmartSpice.ini and it monitors system resources during a simulation. If either memory or hard disk space is getting exhausted(the limit is set to 50MB), ***SmartSpice*** notifies a user through a pop-up window(fig.3) and outputs the current status of system resources in order for ***SmartSpice*** to safely continue a current simulation job on the fly.

The following warning messages illustrate when .options RAWPTS=N is specified, which stores data at into a raw file every specified transient data points N and reached the disk space limit 50MB(below 50MB) during a simulation:

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

Insufficient disk space.

The job has been terminated.

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

Total physical memory size:	267304960 bytes
Available physical memory size	22835200 bytes
Total virtual memory size:	1058312192 bytes
Available virtual memory size:	598319104 bytes
Available disk size:	17571840 bytes

Figure 3. Pop-up window for system resources warning

Thanks to the new feature, it's still time to save the current simulation. A user now has to free up disk space by deleting some unnecessary files on the HDD and continue the simulation answering "Yes" after securing enough disk space. Likewise, memory resources are also being monitored during the simulation.

Decending Sub-Circuit Paths in *SmartSpice*

With parasitic elements becoming more important in circuit performance and characterization it is important that the ***SmartSpice*** deck is constructed in the right way. This then is a short guide to what needs to be specified.

Input Deck Structure

The SPICE input deck must contain circuit heirachy and this must be evaluated first to provide a structure reference on which the path name is mapped (i.e. like a look-up table reference for the X call mapping). For example if you use "." as the sub-circuit delimiter you can have say a capacitor connected between nodes "x1.xnext.xlast" and ground (statement line in deck c1 x1.xnext.xlast 0 10pf).

If this occours at the start of the input deck then there is no reference heirachy and the node name at the top level is the string "x1.xnext.xlast". ***SmartSpice*** allows a large muixture of characters to be used as a node name for customer convienience but for clarity and debugging it is best to reserve some syntax for heirachical traversing only.

So you can specify the path to the connections of a lumped circuit element but in extracting the parasitics it may be necessary to break the element like a transmition line up into a distributed representation. In this case ***Hipex*** will create a sequence of internal nodes of the form "~1", "~2" …….. "~N" to allow the internal breakup of the element.

SPICE has a reference "#1" say to this node but the user is not allowed to connect to these internal device nodes of a distributed element.

The default sub-circuit delimiter is a period "." But this can be changed by the use of the variable of a variable definition (subckt_delimiter=":")

SPECTRE Replacement in the Cadence Flow

Spectre is the Cadence SPICE simulator that can be used in the OASIS design environment. The Cadence environment has been available on Solaris for a number of years. Currently under shifting market pressures to standardize engineering desktop computer on the Linux OS, Cadence has introduced their design environment on Linux. Silvaco has always supported the Solaris based Cadence design environment for seemless ***SmartSpice*** integration. To accommodate new customer demands to support the new Linux environment, Silvaco has developed a ***SmartSpice*** interface for it. This new interface software allows users to replace SPECTRE with ***SmartSpice*** as the analog simulation engine without disrupting the design flow. This benefits analog circuit designers as they can access the latest high quality SPICE models (such as BSIM4.40 CMOS model, TFT, SOI, etc.) without changing the setup. From the user's point of view nothing has changed. The user see's the same Cadence schematic and the same viewer, and is oblivious to which SPICE engine (SPECTRE or ***SmartSpice***) is used. The power of this interface is that it can provide both Analog and mixed Analog/Digital simulation under the Solaris operating system at present. The mixed signal capability will be available shortly for Linux. New functionality has been included to dynamically change the menus so multi-level Monte-Carlo simulation will be possible as well as use of the optimizer functions inside ***SmartSpice.***

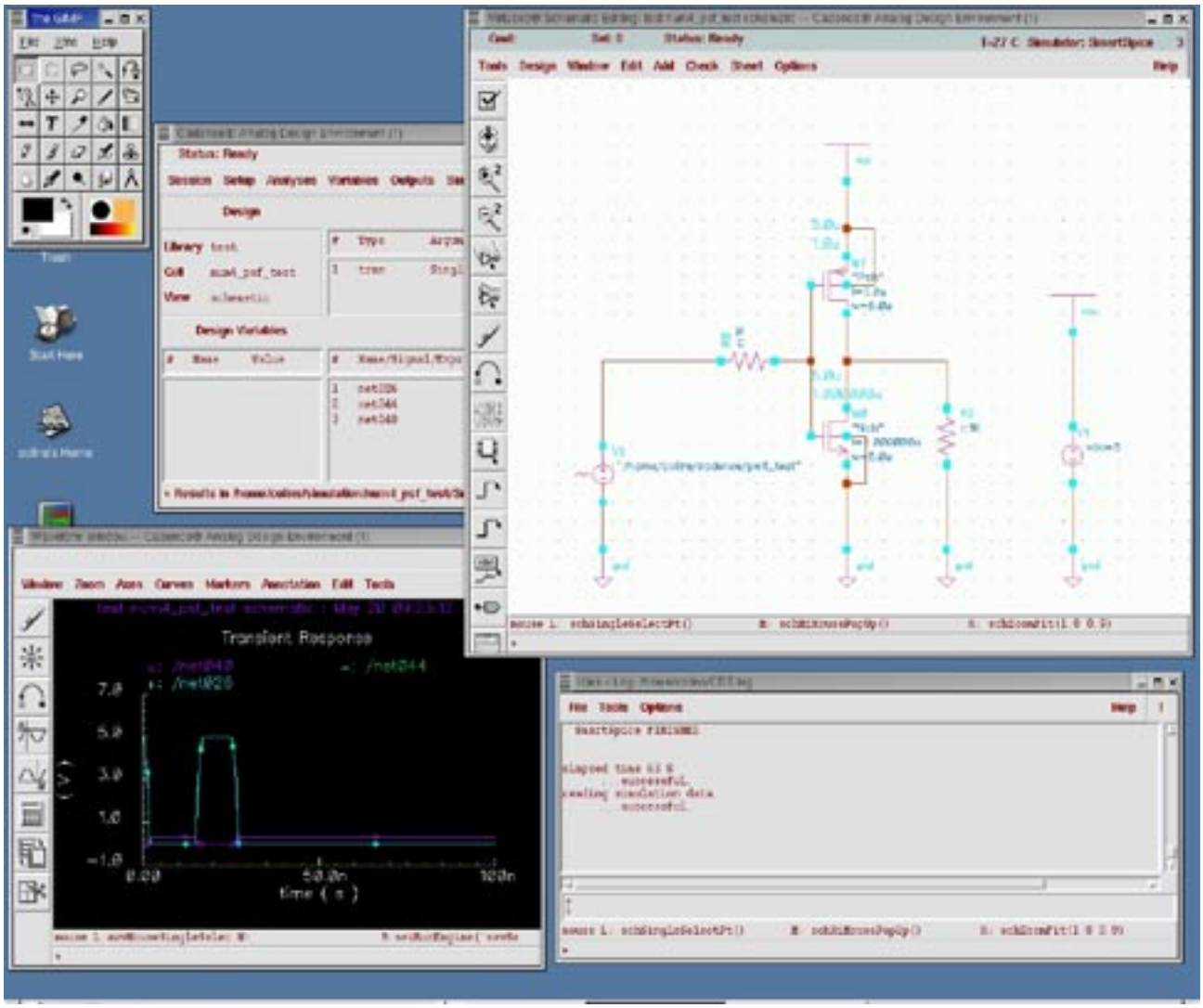

Figure 1. Linux terminal view of Cadence interface.

Calendar of Events

October

1 CMRF - Toulouse, France
IEEE SOI-Newport Beach, CA
2 IEEE SOI-Newport Beach, CA
3
4
5
6
7
8 SAME - Sophia Antipolis, France
9 SAME - Sophia Antipolis, France
FSA Suppliers Expo - San Jose,
10
11
12
13
14
15
16 MSED - Barcelona, Spain
17 MSED - Barcelona, Spain
18
19
20 Polytronic - Switzerland
21 Polytronic - Switzerland
ASICON - Beijing, China
22 Polytronic - Switzerland
ASICON - Beijing, China
23 ASICON - Beijing, China
24 ASICON - Beijing, China
25
26
27
28
29
30
31

November

1
2
3
4
5
6
7
8
9 ICCAD - San Jose, CA
10 ICCAD - San Jose, CA
11 ICCAD - San Jose, CA
CS-MAX - San Jose, CA
12 ICCAD - San Jose, CA
CS-MAX - San Jose, CA
13 ICCAD - San Jose, CA
CS-MAX - San Jose, CA
14
15
16
17
18 EDMO - Manchester, UK
19 EDMO - Manchester, UK
20
21
22
23
24
25
26
27
28
29
30

Bulletin Board

Silvaco Acquires Simucad

Silvaco has acquired the assets of Simucad Inc., a leading provider of Verilog logic and fault simulation software. Simucad was founded in 1981 and has a current user base of over 9,000 design engineers. The comprehensive SILOS Simulation Environment includes an IEEE 1364 compliant Verilog simulator, graphical waveform display, interactive debugging and analysis tools, project management software, and analog extensions.

See Silvaco at the FSA Suppliers Expo

This annual event provides attendees the opportunity to visit with more than 100 suppliers featuring state-of-the-art products and services in the semiconductor sector, as well as learn about the latest market trends from industry leaders through educational panel discussions and presentations.

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

New Functionality – BUS notation in SmartSpice

A new notation has been introduced into SmartSpice to allow a compact expression of a multiple bit wire buss to be used. From this expression the user can simply state the members of a wire bus he wants to generate vectors for or probe. This syntax can be used in conjunction with .SAVE .PROBE

Bus_name<n:m:i>

Bus_name[n – m - i]

Bus_name: an ASCII character string naming the set of wires. N, M, I : a positive integer number < > or [ ] : Indicates the expression to be expanded.

Syntax	Num. Bit's	Expanded form
B<0:2>	3	b<0>,b<1>,b<2>
B<0:2:1>	3	b<0,1,2>
B<3:0:2>	2	b<3,1>
B<0:1,2:2>	3	b<0,1,2>

DATA<2,1,0> represents DATA<2>,DATA<1>, and DATA<0>

Bus Expression Expanded to the single bit level.

DATA<0:3:2> indicates a 2-bit bus containing elements DATA<0> and DATA<2>

DATA<1:3:2> indicates a 2-bit bus containing elements DATA<1> and DATA<3>

DATA<0:3> indicates a 4-bit bus containing elements DATA<0>, DATA<1>, DATA<2> and DATA<3>

DATA<2:0> indicates a 3-bit bus containing elements DATA<2> , DATA<1> and DATA<0>

The following 2 examples show how the different forms of this abbreviation for a set of bus wires can be used.

Example 1

```
* Test bus notation in .COMMANDS

V1 1       0       1
R1 1       bus<1>  1
R2 bus<1>  bus<2>  1
R3 bus<2>  bus<3>  1
R4 bus<3>  bus<4>  1
R5 bus<4>  0       1
.options nomod

.tran 1n 5n

.save v(bus<1-4>)
.print v(bus<1:4>)
.print v(bus<1-4-2>)
.print v(bus<4-1-2>)
.print v(bus<1,3:4>)

.end
```

Example 2

```
* Test bus notation in shell commands

V1 1       0       1
R1 1       bus[1]  1
R2 bus[1]  bus[2]  1
R3 bus[2]  bus[3]  1
R4 bus[3]  bus[4]  1
R5 bus[4]  0       1

.options nomod

.control
save v(bus[1-4])
tran 1n 5n
print v(bus[1:4])
print v(bus[1-4-2])
print v(bus[4-1-2])
print v(bus[1,3:4])

.endc

.end
```

New Functionality – NET statement .

A new optional parameter OUTMODE in the .NET analysis statement was added

A new optional parameter OUTMODE=MODEL|DATA was added to NET (Small-Signal Multi-Terminal Network Analysis) statement. The parameter value 'MODEL' changes the format and content of the created external file, which contains S- or Y-parameter matrices. When OUTMODE=MODEL the S- or Y-parameter matrices are written as .MODEL SP statement lines, they can be included in .MODEL input statement using .INCLUDE. Default OUTMODE=DATA saves results in the external file in the previouse DATAFILE format.

Example:

```
File sdata.par with DATAFILE format
(OUTMODE=DATA):
* N=6 FSTART=1e+09 FSTOP=1e+10
* SPACING=LINEAR MATRIX=SYMMETRIC
VALTYPE=CARTESIAN
*** S - parameters DATAFILE ***
*** ======================== ***
+ 10 $$ - number of data points
*** in 1 n 1 in 2
*** RealData ImaginaryData RealData
*** --- f[0]=1.000000e+09 ----------
3.548139090830e-02 1.681117508985e-03
4.657982349517e-05 4.070485598498e-03
3.547320948755e-02 ...
3.255123897577e-05 2.361486905465e-03
4.774625207240e-05 ...
9.644822616780e-01 -7.860329805485e-03 -
2.632056547515e-05 ...
-2.504703481764e-05 -1.766573638517e-03
9.644644984382e-01 ...
-2.744819175282e-05 -1.879679982283e-03
-2.640582770474e-05 ...
*** --- f[1]=2.000000e+09 ----------
3.554542140701e-02 3.360250854423e-03
1.842940723429e-04 8.132855137570e-03
3.550913774506e-02 ...
1.293936542282e-04 4.721282832777e-03
1.845192935379e-04 ...
9.643114004049e-01 -1.571913862632e-02 -
9.926995728703e-05 ...
-9.875707611075e-05 -3.523037411001e-03
9.642434342675e-01 ...
-1.097405545596e-04 -3.754516558774e-03
-9.972001067792e-05 ...
.........
```

can be used in S-device model as:

```
S1 1 3 5 2 4 6 0 FQMODEL=testQuest
TYPE=s
.model testQuest SP
+ N=6
+ FSTART=1e9 FSTOP=1e10 NI=10
SPACING=lin
+ MATRIX=SYMMETRIC VALTYPE=CARTESIAN
+ DATAFILE=sdata.par
```

File smodel.par with MODEL line format (OUTMODE= MODEL):

```
*** S - parameters DATAFILE ***
+ N=6 FSTART=1e+09 FSTOP=1e+10
+ SPACING=LINEAR MATRIX=SYMMETRIC
VALTYPE=CARTESIAN
+ TYPE = S
+ DATA=(
+ 10 $$ - number of data points
22 of 91SILVACO International*** in
1 in 1 in 2 *** RealData Imaginary-
Data RealData *** --- f[0]=1.000000e+09
---------- + 3.548139090830e-02
1.681117508985e-03 + 4.657982349517e-05
4.070485598498e-03 3.547320948755e-02 ...
+ 3.255123897577e-05 2.361486905465e-03
4.774625207240e-05 ... + 9.644822616780e-
01 -7.860329805485e-03 -2.632056547515e-
05 ... + -2.504703481764e-05 -
1.766573638517e-03 9.644644984382e-01 ...
+ -2.744819175282e-05 -1.879679982283e-
03 -2.640582770474e-05 ... *** --
- f[1]=2.000000e+09 ---------- +
3.554542140701e-02 3.360250854423e-03
+ 1.842940723429e-04 8.132855137570e-
03 3.550913774506e-02 ... +
1.293936542282e-04 4.721282832777e-03
1.845192935379e-04 ... + 9.643114004049e-
01 -1.571913862632e-02 -9.926995728703e-
05 ... + -9.875707611075e-05 -
3.523037411001e-03 9.642434342675e-01 ...
+ -1.097405545596e-04 -3.754516558774e-03
-9.972001067792e-05 ... ......... + )
```

can be used in S-device model as

```
S1 1 3 5 2 4 6 0 FQMODEL=testQuest
.model testQuest SP .include smodel.par
```

NET Analysis

In previous versions of SMARTSPICE it was possible to specify only NET analysis specific vectors in output statements (s11, s12, ... , gamax, etc.). Now vectors that can be specified, and are not limited to NET analysis specific vectors. Expressions can be used as well.

Example:

```
.LET NET rh11='real(h11)'
rh22m='real(h22)+2' foovec='2+3'

.PRINT NET real(s11) rh11 rh22m
rh11+rh22m foovec
```

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*