Circuit simulation with NGSpice

Models for MOS simulation

• BSIM: Berkeley Short-Channel IGFET Model
• The models (mathematical equations) are internal to NGSpice
• Libraries recall a model and configure its parameters

BSIM models are:

- BSIM1 level xx: empirical
- BSIM2 level xx: piecewise analytical
- BSIM3 level xx: analytical with smoothing between the operating regions
- BSIM4 level xx: analytical with introduction of the physical effects of technologies <100nm

BSIM4 model parameters (NMOS. PMOS is analogous)

```
* PTM 45nm Metal Gate / High-K

.model nmos nmos level = 54

+ version = 4.0 binunit = 1 paramchk = 1 mobmod = 0
+ capmod = 2 igcmod = 1 igbmod = geomod = 1
+ diomod = 1 rdsmod = 0 rbodymod = 1 rgatemod = 1
+ permod = 1 acnqsmod = 0 trnqsmod = 0
+ tnom = 27 toxe = 9e-010 toxp = 6.5e-010 toxm = 9e-010
+ dtox = 2.5e-010 epsrox = 3.9 wint = 5e-009 lint = 2.7e-009
+ ll = 0 wl = 0 lln = 1 wln = 1
+ lw = 0 ww = 0 lwn = 1 wwn = 1
+ lwl = 0 wwl = 0 xpart = 0
+ toxref = 9e-010 xl = -20e-9
+ dlcig = 2.7e-009
+ cgso = 1e-010 cgdo = 1e-010 cgbo = 0 cgdl = 7.5e-013
+ cgsl = 7.5e-013 clc = 1e-007 cle = 0.6 cf = 1.1e-010
```

Netlist Spice example (CMOS inverter)

```
*      NOT_ord_1f.net

.INCLUDE ../model/45nm_MGK.pm

.TRAN 0.1p 1800p
.PRINT tran V (nodeIN) V (nodeZ)

.meas tran delay_LH trig v (nodeIN) val = 0.5 fall = 1
targ v (nodeZ) val = 0.5 rise = 1
.meas tran delay_HL trig v (nodeIN) val = 0.5 rise = 1
targ v (nodeZ) val = 0.5 fall = 1

.PARAM Lmin = 45n
.PARAM Wmin = 45n .PARAM Coutp = 1f

*      DGSB
Mp1 nodeZ nodeIN node1 node1 pmos W = {2 * Wmin} L =
{Lmin}
Mn1 nodeZ nodeIN 0 0 nmos W =
{Wmin} L = {Lmin}
Cout nodeZ 0 {Coutp}

Vdd node1 0 1
Vin nodeIN 0 PWL (0 0 600p 0 601p 1 1200p 1 1201p 0
1800p 0)

.end
```

SPICE netlist commands

.includes

General form:

```
.INCLUDE filename
```

Examples:

```
.INCLUDE /users/spice/common/wattmeter.cir
```

Frequently, portions of circuit descriptions will be reused in several input files, particularly with common models and subcircuits. In any ngspice input file, the `.INCLUDE` line may be used to copy some other file as if that second file appeared in place of the `.INCLUDE` line in the original file.

There is no restriction on the file name imposed by ngspice beyond those imposed by the local operating system.

.print

General form:

```
.print prtype ov1 <ov2 ... ov8>
```

Examples:

```
.print tran v(4) i(vin)
.print dc v(2) i(vsrc) v(23, 17)
.print ac vm(4, 2) vr(7) vp(8, 3)
```

The `.print` line defines the contents of a tabular listing of one to eight output variables. `prtype` is the type of the analysis (DC, AC, TRAN, NOISE, or DISTO) for which the specified outputs are desired.

.tran

15.3.9 .TRAN: Transient Analysis

General form:

```
.tran tstep tstop <tstart <tmax>> <uic>
```

Examples:

```
.tran 1ns 100ns
.tran 1ns 1000ns 500ns
.tran 10ns 1us
```

`tstep` is the printing or plotting increment for lineprinter output. For use with the post-processor, `tstep` is the suggested computing increment. `tstop` is the final time, and `tstart` is the initial time. If `tstart` is omitted, it is assumed to be zero. The transient analysis always begins at time zero. In the interval <zero, `tstart`>, the circuit is analyzed (to reach a steady state), but no outputs are stored. In the interval <`tstart`, `tstop`>, the circuit is analyzed and outputs are stored. `tmax` is the maximum stepsize that ngspice uses; for default, the program chooses either `tstep` or (`tstop-tstart`)/50.0, whichever is smaller. `tmax` is useful when one wishes to guarantee a computing interval which is smaller than the printer increment, `tstep`.

`uic` (use initial conditions) is an optional keyword which indicates that the user does not want ngspice to solve for the quiescent operating point before beginning the transient analysis. If this keyword is specified, ngspice uses the values specified using IC=... on the various elements as the initial transient condition and proceeds with the analysis. If the `.ic` control line has been specified, then the node voltages on the `.ic` line are used to compute the initial conditions for the devices. Look at the description on the `.ic` control line for its interpretation when `uic` is not specified.

.meas

The .meas or .measure statement is used to analyse the output data of a tran, ac, or dc simulation. The command is executed immediately after the simulation has finished. .meas analysis may not be used in batch mode (`-b` command line option), if an output file (rawfile) is given at the same time (`-r rawfile` command line option). In this batch mode ngspice will write its simulation output data directly to the output file. The data is not kept in memory, thus is no longer available for further analysis. This is made to allow a very large output stream with only a relatively small memory usage. For .meas to be active you need to run the batch mode with a `.plot` or `.print` command. A better alternative may be to start ngspice in interactive mode, e.g. by running the command

```
ngspice inputfile.
```

.param

General form:

```
.param <ident> = <expr> ; <ident> = <expr> ....
```

Examples:

```
.param pippo=5
```

This line assigns numerical values to identifiers. More than one assignment per line is possible using the ';' separator. The `.param` lines inside subcircuits are copied per call, like any other line. All assignments are executed sequentially through the expanded circuit. Before its first use, a name must have been assigned a value.

Components: MOSFET

General form:

```
MXXXXXXX nd ng ns nb mname <m=val> <l=val> <w=val>
+ <ad=val> <as=val> <pd=val> <ps=val> <nrd=val>
+ <nrs=val> <off> <ic=vds, vgs, vbs> <temp=t>
```

Examples:

```
M1 24 2 0 20 TYPE1
M31 2 17 6 10 MODM L=5U W=2U
M1 2 9 3 0 MOD1 L=10U W=5U AD=100P AS=100P PD=40U PS=40U
```

Note the suffixes in the example: the suffix "u" specifies microns (1e-6 m) and "p" sq-microns (1e-12 m^2).

In the instance card, `nd`, `ng`, `ns`, and `nb` are the drain, gate, source, and bulk (substrate) nodes, respectively. `mname` is the model name and `m` is the multiplicity parameter, which simulates "m" paralleled devices. All MOS models support the "m" parameter. Instance parameters `l` and `w`, channel length and width respectively, are expressed in meters. The areas of drain and source diffusions: `ad` and `as`, in squared mters (m^2).

Components: Capacitor

General form:

```
CXXXXXXX n+ n- <value> <mname> <m=val> <scale=val> <temp=val>
+ <dtemp=val> <ic=init_condition>
```

Examples:

```
CBYP 13 0 1UF
COSC 17 23 10U IC=3V
```

Ngspice provides a detailed model for capacitors. Capacitors in the netlist can be specified giving their capacitance or their geometrical and physical characteristics. Following the original spice3 "convention", capacitors specified by their geometrical or physical characteristics are called "semiconductor capacitors" and are described in the next section.

In this first form `n+` and `n-` are the positive and negative element nodes, respectively and `value` is the capacitance in Farads.

Capacitance can be specified in the instance line as in the examples above or in a `.model` line, as in the example below:

```
@C1 15 5 cstd
C2 2 7 cstd
.model cstd C cap=3n
```

Components: Voltage Generator

General form:

```
VXXXXXXX N+ N- <<DC> DC/TRAN VALUE> <AC <ACMAG <ACPHASE>>>
+ <DISTOF1 <F1MAG <F1PHASE>>> <DISTOF2 <F2MAG <F2PHASE>>>
IYYYYYYY N+ N- <<DC> DC/TRAN VALUE> <AC <ACMAG <ACPHASE>>>
+ <DISTOF1 <F1MAG <F1PHASE>>> <DISTOF2 <F2MAG <F2PHASE>>>
```

Examples:

```
VCC 10 0 DC 6
VIN 13 2 0.001 AC 1 SIN(0 1 1MEG)
ISRC 23 21 AC 0.333 45.0 SFFM(0 1 10K 5 1K)
VMEAS 12 9
VCARRIER 1 0 DISTOF1 0.1 -90.0
VMODULATOR 2 0 DISTOF2 0.01
IIN1 1 5 AC 1 DISTOF1 DISTOF2 0.001
```

`n+` and `n-` are the positive and negative nodes, respectively. Note that voltage sources need not be grounded. Positive current is assumed to flow from the positive node, through the source, to the negative node. A current source of positive value forces current to flow out of the `n+` node, through the source, and into the `n-` node. Voltage sources, in addition to being used for circuit excitation, are the "ammeters" for ngspice, that is, zero valued voltage sources may be inserted into the circuit for the purpose of measuring current. They of course have no effect on circuit operation since they represent short-circuits.

Components: Piece-wise linear voltage generator

General Form:

```
PWL(T1 V1 <T2 V2 T3 V3 T4 V4 ...>) <r=value> <td=value>
```

Examples:

```
VCLOCK 7 5 PWL(0 -7 10NS -7 11NS -3 17NS -3 18NS -7 50NS -7) r=0 td=15NS
```

Each pair of values (T_i, V_i) specifies that the value of the source is V_i (in Volts or Amps) at time = T_i. The value of the source at intermediate values of time is determined by using linear interpolation on the input values. The parameter r determines a repaet time point. If r is not given, the whole sequence of values (T_i, V_i) is issued once, then the output stays at its final value. If $r = 0$, the whole sequence from $time = 0$ to $time = Tn$ is repeated forever. If $r = 10ns$, the sequence between 10ns and 50ns is repeated forever. the r value has to be one of the time points T1 to Tn of the PWL sequence. If td is given, the whole PWL sequence is delayed by a delay time $time = td$. The current source still needs to be patched, td and r are not yet available.

gschem

- It is possible to draw a circuit with the gschem program
- The output is a graphical .sch file
- The gnetlist program converts the .sch file into a spice netlist

Command line:

```
gnetlist -g spice-sdb -o file_output.net file_input.sch
```

Interactive use of NGSpice

Launch ngspice:

`ngspice`

From the ngspice prompt

- Load the circuit:

`source file_netlist.net`

- Launch the simulation:

`run`

For the graphical diagram of a variable:

`plot node_name`

Commands from NGSpice prompt: display

General Form:

```
display [varname ...]
```

Prints a summary of currently defined vectors, or of the names specified. The vectors are sorted by name unless the variable nosort is set. The information given is the name of the vector, the length, the type of the vector, and whether it is real or complex data. Additionally, one vector is labelled [scale]. When a command such as plot is given without a vs argument, this scale is used for the X-axis. It is always the first vector in a rawfile, or the first vector defined in a new plot. If you undefine the scale (i.e, let TIME = []), one of the remaining vectors becomes the new scale (which is undetermined).

Commands NGSpice prompt: setcirc

General Form:

```
setcirc [circuit name]
```

The current circuit is the one that is used for the simulation commands below. When a circuit is loaded with the source command (see below) it becomes the current circuit.

Commands from NGSpice prompt: setplot

General Form:

```
setplot [plotname]
```

Set the current plot to the plot with the given name, or if no name is given, prompt the user with a menu. (Note that the plots are named as they are loaded, with names like `tran1` or `op2`. These names are shown by the setplot and display commands and are used by diff, below.) If the "`New plot`" item is selected, the current plot becomes one with no vectors defined.

Note that here the word "plot" refers to a group of vectors that are the result of one ngspice run. When more than one file is loaded in, or more than one plot is present in one file, ngspice keeps them separate and only shows you the vectors in the current plot.

Commands from NGSpice prompt: listing

General Form:

```
listing [logical] [physical] [deck] [expand] [param]
```

If the **logical** argument is given, the listing is with all continuation lines collapsed into one line, and if the **physical** argument is given the lines are printed out as they were found in the file. The default is logical. A **deck** listing is just like the physical listing, except without the line numbers it recreates the input file verbatim (except that it does not preserve case). If the word **expand** is present, the circuit is printed with all subcircuits expanded. The option **param** allows to print all parameters and their actual values.

Commands from NGSpice prompt: plot

General Form:

```
plot exprs [ylimit ylo yhi] [xlimit xlo xhi] [xindices xilo xihi]
[xcompress comp] [xdelta xdel] [ydelta ydel] [xlog] [ylog] [loglog]
[vs xname] [xlabel word] [ylabel word] [title word] [samep]
[linear]
```

Plot the given exprs on the screen (if you are on a graphics terminal). The xlimit and ylimit arguments determine the high and low x- and y-limits of the axes, respectively. The xindices arguments determine what range of points are to be plotted - everything between the xilo'th point and the xihi'th point is plotted. The xcompress argument specifies that only one out of every comp points should be plotted. If an xdelta or a ydelta parameter is present, it specifies the spacing between grid lines on the X- and Y-axis. These parameter names may be abbreviated to xl, yl, xind, xcomp, xdel, and ydel respectively.

The xname argument is an expression to use as the scale on the x-axis. If xlog or ylog are present then the X or Y scale, respectively, is logarithmic (loglog is the same as specifying both). The xlabel and ylabel arguments cause the specified labels to be used for the X and Y axes, respectively.

If samep is given, the values of the other parameters (other than xname) from the previous plot, hardcopy, or asciiplot command is used unless re-defined on the command line.

The title argument is used in the place of the plot name at the bottom of the graph.

The linear keyword is used to override a default logscale plot (as in the output for an AC analysis).

Finally, the keyword polar to generate a polar plot. To produce a smith plot, use the keyword smith. Note that the data is transformed, so for smith plots you will see the data transformed by the function (x-1)/(x+1). To produce a polar plot with a smith grid but without performing the smith transform, use the keyword smithgrid.

Commands from NGSpice prompt: show

General Form:

```
show devices [ : parameters ] , ...
```

The show command prints out tables summarizing the operating condition of selected devices (much like the spice2 operation point summary). If device is missing, a default set of devices are listed, if device is a single letter, devices of that type are listed; if device is a subcircuit name (beginning and ending in ":") only devices in that subcircuit are shown (end the name in a double-":" to get devices within sub-subcircuits recursively). The second and third forms may be combined ("letter:subcircuit:") or ("letter:subcircuit::") to select a specific type of device from a subcircuit. A device's full name may be specified to list only that device. Finally, devices may be selected by model by using the form "#modelname" or ":subcircuit#modelname" or "letter:subcircuit#modelname".

If no parameters are specified, the values for a standard set of parameters are listed. If the list of parameters contains a "+", the default set of parameters is listed along with any other specified parameters.

For both devices and parameters, the word "all" has the obvious meaning.

Note: there must be spaces separating the ":" that divides the device list from the parameter list.

Commands from NGSpice prompt: alter

Alter changes the value for a device or a specified parameter of a device or model.

General Form:

```
alter dev = <expression>
alter dev param = <expression>
alter @dev[param] = <expression>
```

<expression> must be real (complex isn't handled right now, integer is fine though, but no strings. For booleans, use 0/1.

Old style (pre 3f4):

```
alter device value
alter device parameter value [ parameter value ]
```

Using the old style, its first form is used by simple devices which have one principal value (resistors, capacitors, etc.) where the second form is for more complex devices (bjt's, etc.). Model parameters can be changed with the second form if the name contains a "#". For specifying vectors as values, start the vector with "[", followed by the values in the vector, and end with "]". Be sure to place a space between each of the values and before and after the "[" and "]".

Some examples are given below:

Examples (Spice3f4 style):

```
alter vd = 0.1
alter vg dc = 0.6
alter @m1[w]= 15e-06
alter  @vg[sin] [ -1 1.5 2MEG ]
alter @Vi[pwl] = [ 0 1.2 100p 0 ]
```

Commands from NGSpice prompt: set filetype

```
set filetype = ascii
```

NGSpice: write commands

General Form:

```
write [file] [exprs]
```

Writes out the expressions to `file`.

First vectors are grouped together by plots, and written out as such (i.e, if the expression list contained three vectors from one plot and two from another, then two plots are written, one with three vectors and one with two). Additionally, if the scale for a vector isn't present, it is automatically written out as well.

The default format is binary, but this can be changed to ascii with the set filetype command. The default filename is `rawspice.raw`, or the argument to the `-r` flag on the command line, if there was one, and the default expression list is all.

Parametric simulations (example):

```
.control

alter @cout[c]=1e-15
run
alter @cout[c]=2e-15
run
alter @cout[c]=3e-15
run
plot nodein tran1.nodez tran2.nodez tran3.nodez

.endc
```

Parametric simulations (example):

```
.control
let valore =0.0f
while (valore lt 11f)
   alter @cout[c]=valore
   run
   let valore = valore + 1f
end

plot nodein tran1.nodez tran2.nodez tran3.nodez
 tran4.nodez tran5.nodez tran6.nodez tran7.nodez
   tran8.nodez tran9.nodez tran10.nodez tran11.nodez

.endc
```

General Form:

```
set [word]
set [word = value] ...
```

Set the value of word to be value, if it is present. You can set any word to be any value, numeric or string. If no value is given then the value is the boolean 'true'.

The value of word may be inserted into a command by writing `$word`. If a variable is set to a list of values that are enclosed in parentheses (which must be separated from their values by white space), the value of the variable is the list.

The variables used by nutmeg are listed in the following section 17.4.46.