Tutorial on the "hspc" Program

by Michael H. Perrott

November, 2002

Contents

1 Intro

The Hspice/NGspice Conversion (hspc) program provides an easy method of modifying Hspice or NGspice netlists to perform the following functions

- Add digital input stimulus to circuit nodes using binary vector notation
- Automatically add source/drain area and perimeter values to each MOSFET instantiation
- Simulate over combinations of temperature, process, and parameter values using automatically generated .alter statements
- Selectively remove lines from the SPICE netlist
- Add SPICE statements such as signal sources, voltage supply sources, etc. to the netlist
- Add SPICE statements such as .alter and .end that are explicitly placed at the end of the netlist.
- Convert transistor instantiations to subcircuits in order to perform simulations involving random mismatch.

Each of the above features will be covered in the text below.

2 Installing and Compiling the Program

The 'hspc' package is contained in the file named hspc.tar.gz. To unpack the contents of that file on a UNIX computer, type the following commands:
gunzip hspc.tar.gz
tar xvf hspc.tar
You will now have a directory named HSPC within which are 5 files:

- hspc.c
- hspc_makefile
- matrix_func.c
- matrix_func.c
- hspc.pdf

To compile the program on a UNIX computer, simply go into the HSPC directory and type:
make -f hspc_makefile
The above command will create the executable 'hspc'.

3 Running the Program

To use the hspc program, simply run it with the following parameters

hspc input_spice_file output_spice_file parameter_file

The hspc program will then alter the input_spice_file according to the information in the parameter_file, and output the resulting spice deck to the file output_spice_file. You should then run SPICE on output_spice_file.

4 Example Parameter File

To demonstrate the capabilities of the program, we present an example parameter file as follows below. An explanation of the statements used in the example will be given in the next sextion.

******* Comment lines have an asterisk as their first character

**** set parameters used to calculate source/drain areas and perimeters
**** Note: set_mode_diff specifies method of calculation with fingered transistors

```
****          options: geo, conservative, drain_smaller, source_smaller
> set_mode_diff geo
> set_hdout .54u
> set_hdin .5u
> use_four_sided_perimeter

******* Create .alter statements to simulate combinations of
****    process, temperature, and parameter corners
****    example: simulate over all combinations of temp, process, and parameter vdd
****    for the following parameter values
> skew
> process 'mos_library.l' ss ff
> temp -40 25 125
> vdd 2.37 2.5 2.625
> end_skew

******* Replace transistor instantiations with subcircuits in order to
****    simulate the impact of random mismatch
> mismatch

******* Remove all lines in the netlist that begin with the specified words
****    Example: flatten the top subcircuit
- .subckt top_level_ckt
- .ends top_level_ckt

**** Example: change the value of statements
- .TEMP
.TEMP 50

******* Add HSPICE lines that are explicitly placed at the end of the netlist
**** (Note: "> skew . . . . . > end_skew" statements are treated as "!" statements
**** therefore, place "> skew . . . . . > end_skew" BEFORE "!.end" statement
!.end

******* Include additional HSPICE statements into the netlist
.prot
.lib '/mit/perrott/cds/models/model.lib' ss
.unprot
.option post
* .tran .001u 1u
.dc Vin1 -.01 .01 .0001
.ac dec 10 100 100meg
.measure dc biaspoint find v(in) when v(iop2) = 0
.noise v(iop2) vin2 10

Vsupply g1 0 3.5v
Vgnd g0 0 0v
Iiop1 0 iop1 64u
Iiop2 0 iop2 192u
```

```
Vin1 in dc_in dc 0mv ac 1
Vin2 _in dc_in dc −0.1mv

V_dc_in dc_in 0 1.8v
Cout iop2 0 15p
Rout iop2 0 100k
                                                                        60
******************** Add digital stimulus lines *****************
**** timing statements: timing delay transition_time time_inc vlow vhigh
**** input statements: input [signal_name(s)] [signal_waveform(s)]

** Example: create a 3V, 100 MHz clock with .3ns transition times
> timing 0n .3n 5n 0 3
> input clk [set 1 0 R]

** Example: create a periodic signal waveform A with .3n transition times
**          whose transitions occur .4n after rising transitions of clk   70
> timing .4n .3n 10n 0 3
> input A [set 1 0 0 1 0 1 1 1 0 R]
** Note: a convenient way of expressing the above statement is
**    to use (symbol_value number_of_symbols) as shown below
** > input A [set 1 (0 2) 1 0 (1 3) 0 R]

** Example: create nonperiodic signal waveforms b and c as identical waveforms
**          that transition a few times and then remain at value one
** (Use the same timing as already set above in the last 'timing' statement)
> input [b c] [set 1 1 0 1 0 0 1 0 1]                                   80

** Example: load data signals x, y, and z from file "signals.dat" whose first
**          column (column 0) corresponds to x, second column to y, and
**          third column to z
** Note: notation is [file file_name first_column] as shown below
** (Use the same timing as already set above in the last 'timing' statement)
> input [x y z] [file signals.dat 0]

** Example: load data signals u and v from file "signals2.dat" whose fourth
**          column (column 3) corresponds to u and fifth column to v      90
> input [u v] [file signals2.dat 3]

******* Add digital stimulus lines with varying phase
** Example: repeat the sequence 1 0 1 1 0 0 for a total period
**          of 100 cycles (each cycle period is set by previous "> timing"
**          statement), and then output 5 such periods where all
**          the cycles in each period are adjusted in phase by
**          .00, .05, .10, .05, .00 UI
> phase data 100 [phase 0 5 10 5 0] [data 1 0 1 1 0 0]
```

5 Documentation on Parameter File Statements

We now explain the various statements and their options used in a parameter file. We will reference the example file given in the last section throughout much of this discussion.

There are 4 special characters that specify the purpose of a line when they are used as the first non-space character in that line. These characters are as follows

- >: Specifies parameters used to modify transistor statements in input_spice_file or to auto-generate .alter statements or digital input sources.
- −: Specifies the removal of lines in input_spice_file that begin with the specified words,
- !: Specifies SPICE statements that should be added to the end of output_spice_file.
- ∗: Specifies that the line is a comment, and should be ignored in producing output_spice_file,

All lines in the parameter file that do not begin with any of the above characters will be considered as SPICE statements that are added to output_spice_file in addition to the SPICE statements in input_spice_file.

We will now go into more detail in regards to commands used in conjunction with the above characters

5.1 Documentation on '>' Commands

Parameter file lines starting with '>' are intended to specify modification of SPICE lines within input_spice_file in the following ways

- Specification of source/drain area and perimeter values for the transistors specified in input_spice_file,
- Auto-generation of .alter statements to simulate combinations of temperature, process, and parameter variations,
- Replacement of transistor instantiations with subcircuits in order to simulate the impact of random mismatch,
- Auto-generation of pwl signals that correspond to digital signals specified with a binary sequence.

Beginning with the first item, there are four statements used in specifying the manner in which the source/drain area and perimeter values are calculated:

- set_mode_diff: used to specify the manner in the source/drain area and perimeter dimensions are calculated. The four possible values of set_mode_diff are conservative, drain_smaller, and source_smaller, and geo.
  - conservative (geo=0): neither the source or drain of ALL transistors are shared.
  - drain_smaller (geo=1): the drain of ALL transistors are assumed to be shared with other transistors or other stripes of the same transistor.
  - source_smaller (geo=2): the source of ALL transistors are assumed to be shared with other transistors or other stripes of the same transistor.
  - geo: the geo parameter of each transistor is used to determine whether their source and/or drain is shared.
- set_hdout: specifies the source/drain extension when it is NOT shared with another transistor.
- set_hdin: specifies the source/drain extension when it is shared with another transistor.
- use_four_sided_perimeter: when calculating perimeter capacitance, include the sidewall adjacent to the gate.

Figure 1 displays the corresponding transistor configurations for each value of geo.

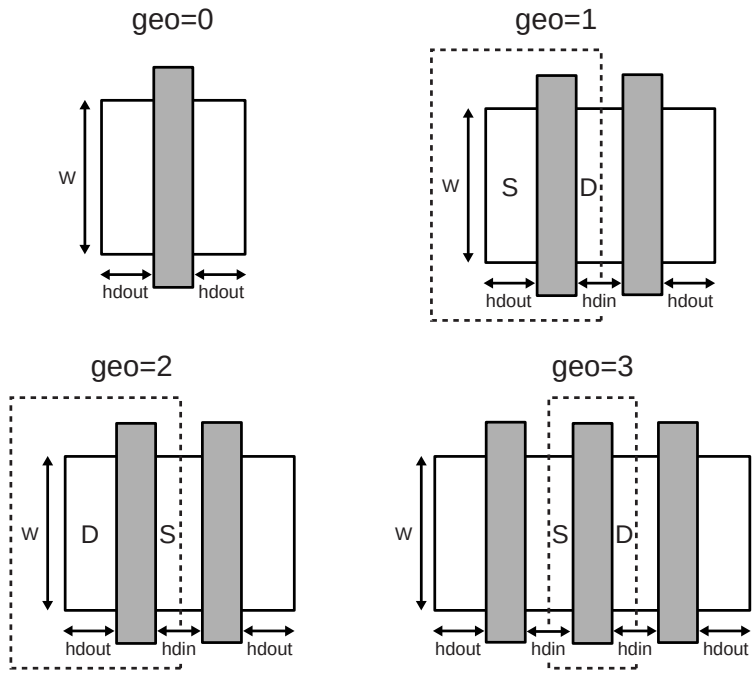

Figure 1 Diffusion parameters for different values of geo.

The simulation of a spice deck over temperature, process, and parameter variations can be performed by using the "> skew" command. As an example, the statements

```
> skew
> process 'mos_library.l' ss ff
> temp −40 125
> vdd 2.37 2.625
> end_skew
```

add the following .alter statements to the Spice netlist:

```
.alter (proc,temp,vdd)=(ss,−40,2.37)
.lib 'mos_libraryl' ss
.temp −40
.param vdd = 2.37

.alter (proc,temp,vdd)=(ss,−40,2.625)
.lib 'mos_libraryl' ss
.temp −40
.param vdd = 2.625
```



```
.alter (proc,temp,vdd)=(ss,125,2.37)
.lib 'mos_libraryl' ss
.temp 125
.param vdd = 2.37

.alter (proc,temp,vdd)=(ss,125,2.625)
```

```
.lib 'mos_libraryl' ss
.temp 125
.param vdd = 2.625

.alter (proc,temp,vdd)=(ff,-40,2.37)
.lib 'mos_libraryl' ff
.temp -40
.param vdd = 2.37

.alter (proc,temp,vdd)=(ff,-40,2.625)
.lib 'mos_libraryl' ff
.temp -40
.param vdd = 2.625

.alter (proc,temp,vdd)=(ff,125,2.37)
.lib 'mos_libraryl' ff
.temp 125
.param vdd = 2.37

.alter (proc,temp,vdd)=(ff,125,2.625)
.lib 'mos_libraryl' ff
.temp 125
.param vdd = 2.625
```

When using the skew function, you must specify at least one temperature and process parameter (and the library for the process). You may also include up to four parameters associated with .param statements.

Transistor instantiations can be replaced with subcircuits in order to simulate the impact of random mismatch. As an example, by including the "> mismatch" command in the parameter file, the transistor instantiation:

```
M18 n0 db net38 gnds nch w='nwid' l='nlen'
```

is changed to

```
XM18 n0 db net38 gnds
+  nch len='nlen' wid='nwid' mul=1
+  adrain='hdout*(nwid)' asource='hdout*(nwid)'
+  pdrain='2*hdout+2*(nwid)' psource='2*hdout*2*(nwid)'
```

One would then include a file that provides the transistor subcircuits encompassing mismatch. An example of such a file is as follows:

```
**** Get the following numbers from the fab that you are using ***
.param Avtn  = ???? $ V*um
.param Avtp  = ???? $ V*um

.param Abn  = ???? $ %*um
.param Abp  = ???? $ %*um

**** These are the subcircuit models that implement transistor mismatch ***
.subckt nch d g s b wid=0 len=0 mul=1 geom=0 adrain=0 asource=0 pdrain=0 psource=0
  .param rtarea = 'sqrt(mul*wid*len*1e12)'
  .param delvt = agauss(0.0, 'Avtn/rtarea', 1)
  .param new_w = gauss('wid', '0.01*Abn/rtarea', 1)
  mn d g s b nch w=new_w l=len delvto=delvt m=mul geo=geom ad=adrain as=asource
pd=pdrain ps=psource
.ends nch

.subckt pch d g s b wid=0 len=0 mul=1 geom=0 adrain=0 asource=0 pdrain=0 psource=0
  .param rtarea = 'sqrt(mul*wid*len*1e12)'
  .param delvt  = agauss(0.0, 'Avtp/rtarea', 1)
  .param new_w  = gauss('wid', '0.01*Abp/rtarea', 1)
  mp d g s b pch w=new_w l=len delvto=delvt m=mul geo=geom ad=adrain as=asource
pd=pdrain ps=psource
.ends pch
```

Parameter file lines starting with '>' can also be used to specify digital binary sequence signals that will be included in output_spice_file as SPICE pwl waveforms. There are two classes of statements used to specify the digital waveforms

- timing statements: specify timing and voltage swing parameters to be used for all the following input statements,
- input statements: specification of signal nodes to apply the source to, and the waveform in terms of binary vectors.

The general form of timing statements is

> timing delay transition_time time_inc vlow vhigh,

and the meaning of each of the parameters in this statement is given as

- delay: initial delay of the waveform,
- transition_time: time to complete rising or falling transitions,
- time_inc: the amount of time spanned by each binary symbol,
- vlow: the voltage value corresponding to the binary symbol 0,
- vhigh: the voltage value corresponding to the binary symbol 1.

The input statements consist of several variations that are best explained through examples

- > input clk [set 1 0 R]: creates a pwl voltage source V_clk that periodically alternates between vhigh and vlow with period 2*time_inc (as specified by the last timing statement).
- > input A [set 1 0 0 1 0 1 1 1 0 R]: creates a periodic pwl voltage source V_A that alternates between vhigh and vlow according to the binary vector given.
- > input A [set 1 (0 2) 1 0 (1 3) 0 R]: a convenient shorthand method to implement the same waveform specified in the previous statement.
- > input A [set 1 (0 2) 1 0 (1 3) 0]: implements the same waveform as specified in the previous statement, with the exception that it is not repeated. Once the specified transitions are executed, V_A remains at the last specified symbol (which yields vlow in this case).
- > input [A B] [set 1 (0 2) 1 0 (1 3) 0]: implements two identical pwl voltage sources, V_A and V_B, with the same waveform as specified in the previous statement.

- > input [x y z] [file signals.dat 0]: implements pwl voltage sources V_x, V_y, and V_z whose waveform voltage values are specified by columns in the file signals.dat and whose timing parameters are set by the most recent timing statement. The last parameter in [file ...], which is 0 in this case, specifies the column corresponding to node x. Nodes y and z then take their data from the next two columns in the file.

5.2 Documentation on '−' Commands

Parameter file lines starting with '−' specify SPICE lines in input_spice_file that should not be included in output_spice_file. The general form of this statement is

− word1 word2 ...,

in which case all lines in input_spice_file that begin with word1, word2, and any following words will not be included in output_spice_file. (*Note that the '-' symbol must be followed by a space*). Any number of words may be specified, which allows one a measure of selectivity when removing lines. For instance, the line

− .MODEL NMOS

discludes all NMOS modeling statements in input_spice_file, while

− .MODEL

discludes all modeling statements (NMOS and PMOS) in input_spice_file.

5.3 Documentation on '!' Commands

Parameter file lines starting with '!' specify SPICE lines that should be included at the end of output_spice_file. I main use this option to place a .end statement at the end of the file. Can also be useful for placing .alter statements.