SPICE
Cheatsheet

*SPICE simulates circuits using an iterative algorithm to approximate circuit voltages & currents under a wide number of stimuli. The general pattern involves creating a "**netlist**", followed by the invocation of one or more "**analyses**." There exist several ways to **extract** data. Interactive or batch "**control**" affords more granular simulator manipulation, while "**models**" allow granular setup of individual circuit elements. "**XSPICE**" enables mixed analog/digital simulations, and an **API** enables remote control by other applications. For example:*

```
.TITLE PNP Power Switch      $ title
* Simulating a BJT switch    $ line comment
.GLOBAL gnd vcc              $ global elements
.INCLUDE /my/file            $ import file
.LIB /my/file mod1           $ import module
.temp 27                     $ set temperature
.param pvcc=5                $ variables
.param vmax={1 + pvcc}       $ expressions
.func foo(a,b) = a + b       $ function
RF2=1K $ Gain set by RF2     $ in-line cmnt
.END                         $ end SPICE
```

*You can invoke from within **KiCad**, which will invoke an instance of ngspice in an external terminal window using the netlist that KiCad exports. See here. For more **tutorials**, see here. For simulating **PSpice** and LTSpice within ngspice, see here.*

1 Syntax

Scale Factors

Scale factors $\in [T|G|M|K|m|u|n|p|f]$ *are suffixed directly following a number.*

Built-in Functions

Expressions can include many of the normal functions: trig (`sin`, `cos`, `tan`, `acos`, `asin`, `atan`*), hyperbolic (*`sinh`, `cosh`, `acosh`, `asinh`, `atanh`*), exponential (*`exp`, `ln`, `log`, `log10`*), misscellaneous (*`abs`, `sqrt`, `u`, `u2`, `uramp`, `floor`, `ceil`, `i`*).*

Options

Gen/Elem	DC/OP	AC/Tran
[NO]ACCT	ABSTOL	AUTOSTOP
NOINIT	GMIN	CHGTOL
LIST	ITIL[1|2]	CONVSTEP
NOMOD	NOOPITER	CONVABSTEP
NOPAGE	PIVREL	INTERP
NODE	PIVTOL	ITIL[3-6]
OPTS	RELTOL	MAXEVITER
SEED	RSHUNT	MAXOPALTER
TEMP	VNTOL	MAXORD
TNOM		METHOD
WARN		NOOPALTER
BADMOS3		RAMPTIME
DEFA[D|S]		SRCSTEPS
DEF[L|W]		TRTOL
SCALE		XMU

Conditionals

```
.if(m0==1) .model N1 NMOS lvl=49 Ver=3.1
.elseif(m1==1) .model N1 NMOS lvl=49 Ver=3
.else .model N1 NMOS lvl=49 Ver=3.3.0
.endif
```

Looping

```
while <cond> ... end              # while
repeat <num> ... end              # repeat
dowhile <cond> ... end            # dowhile
foreach <var> <val> ... end       # foreach
```

Common ancillary looping aids include: `label`, `goto`, `continue`, *and* `break`.

2 Netlists

Each entry creates an "instance" of a device element, indicating the element type (first letter), its nodal connections, & any element-specific param's:

```
<device> <nd>+ <val> <model>? <param>*
```

Common to most element types are options `m` *(multiple) and* `scale` *(for sizing main parameter).*

Resistors

```
Rxxx n+ n- [resistance|r=]value
+ [ac=val] [temp=val] [dtemp=val]
+ [tc1=val] [tc2=val] [noisy=0|1]
R2 5 7 1K ac=2K              $ 1k resistor
```

Capacitors

```
Cxxx n+ n- [value] [mname] [scale=val]
+ [temp=val] [dtemp=val] [tc1=val]
+ [tc2=val] [ic=init_cond.]
Cosc 17 23 10U IC=3V         $ 10μ cap.
```

Inductors

```
Lyyy n+ n- [value] [mname] [nt=val]
+ [temp=val] [dtemp=val] [tc1=val]
+ [tc2=val] [ic=init_cond.]
LSHUNT 3 9 10U IC=15.1MA     $ inductor
```

Switches

```
Sxxx n+ n- nc+ nc- model [on|off]
s1 1 2 3 4 switch1 ON        $ switch

Wyyy n+ n- vnam model [on][off]
wreset 5 6 vclck lossyswitch OFF  $ I-ctrl'd
```

Independent Sources

Independent source input can be defined by a function $\in$*:* `pulse`, `sin`, `exp`, `pwl`, `sffm`, `am`, `trnoise`, `trrandom`, `external`.

```
[I|V]xxx n+ n- [[dc] dc/tran value]
+[ac [acmag [acphase]]]
+[distof1 [f1mag [f1phase]]]
+[distof2 [f2mag [f2phase]]]
Vin 13 2 0.001 AC 1 SIN(0 1 1M)        $ V-src
Iin 8 9 AC .33 45 SFFM(0 1 10K 5 1K)   $ I-src
```

Dependent Sources

$\exists$ *4 types of dependent sources: VCCS (G), VCVS (E), CCCS (F), CCVS (H). One can create source non-linearity (not shown) by additionally specifying arbitrary controlling expressions or polynomials.*

```
[E|G]xxx N+ N- NC+ NC- VALUE
E1 2 3 14 1 2.0              $ VCVS

[F|H]xxx N+ N- VNAM VALUE
F1 13 5 VSENS 5 m=2          $ CCCS
```

Diodes

```
Dxxx n+ n- mname [area=val] [pj=val]
+ [off] [ic=vd] [temp=val] [dtemp=val]
DBRIDGE 2 10 DIODE1          $ Diode
```

BJTs

```
Qxxx nc nb ne [ns] [tj] mname
+ [area=val] [areac=val] [areab=val]
+ [off] [ic=vbe,vce] [temp=val]
+ [dtemp=val]
Q23 10 24 13 QMOD IC=0.6, 5.0   $ BJT
```

MOSFETs

```
Mxxx 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]
M31 2 17 6 10 MOSN L=5U W=2U$ MOSFET
```

Subcircuits

Afford hierarchically nestable, and reusable subcircuits. These become flattened upon compilation. Subcircuits may expose an arbitrary # of nodes external to themselves. For example, a voltage divider:

```
.subckt vdivide 1 2 3        $ 3 ports
r1 1 2 10K                   $ 10K resistor
r2 2 3 5K                    $ 5K resistor
.ends vdivide                $ exclude to end all
```

Can parameterize subckts, here with rval, cval:

```
.subckt myfilter in out rval=100k cval=100nF
```

3 Analyses

AC

Undergo variation on AC frequency, progressing in decade, octave, or linear fashion for indicated numb of pts:

```
.ac [dec|oct|lin] n fstart fstop
.ac dec 10 1K 100MEG
```
\# $f \in [1e3, 1e8]$

DC

Sweep a source, resistor, or temperature:

```
.dc srcnam vstart vstop vincr
.dc VIN 0.25 5.0 0.25
.dc TEMP -15 75 5            # temp sweep
```
\# sweep $V_{IN} \in [.25, 5]$ (for `.dc VIN 0.25 5.0 0.25`)

Distortion

```
.disto dec nd fstart fstop
.disto dec 10 1kHz 100MEG    #
```

Noise

```
.noise v(output) src [dec|lin|oct]
+ pts fstart fstop
.noise v(5) VIN dec 10 1kHz 100MEG   #
```

Operating Point

```
.op                          #
```

Pole-Zero

```
.pz nd1 nd2 nd3 nd4
+ [vol|cur] [pol|zer|pz]
.pz 1 0 3 0 cur pol          # pole-only for currents
.pz 1 0 3 0 vol pz           # poles & zeros
```

Sensitivity

```
.sens outvar
+ [dec|oct|lin] n fstart fstop
.sens V(1,OUT)               #
.sens V(OUT) ac dec 10 100 100k
```

Transfer Function

```
.tf outvar insrc
.tf v(5, 3) VIN
.tf i(VLOAD) VIN             #
```
\# ratio $\frac{V_5}{V_{IN}}$ (for `.tf v(5, 3) VIN`)

Transient

Study temporal transient, optionally using initial conditions:

```
.tran tstep tstop
+ [tstart [tmax]] [uic]
.tran 1ns 100ns              # 100ns transient
.tran 1ns 1us 500ns          # start later
```

Periodic Steady State

[experimental] Calculate indicated harmonics:

```
.pss gfreq tstab oscnob psspoints
+ harms sciter steadycoeff [uic]
.pss 150 200e-3 2 1024 11 50 5e-3 uic
.pss 624e6 500n bout 1024 10 100 5e-3 uic
```

4 Control

Batch (or "Control") Mode

Use this mode for scripting the build and execution of a SPICE simulation. Batch mode commands are generally prefaced with a . (dot), unless they appear in an explicit `.control` *section:*

```
.nodeset v(12)=4.5 v(4)=2.2      # iter. start guess
.ic v(11)=5 v(4)=-5 v(2)=2.2     # set init. cond's
.control                         # enter
...                              # control section
.endc                            # leave
```

Interactive

This is like a REPL except that the simulation requires an explicit directive in order to commence. Side-effect having commands issued interactively include:

ac
alias
alter
altermod
alt'rparam
asciiplot
aspice
bug
cd
cdump
circbyline
codemodel
compose
dc
define
deftype
delete
destroy
devhelp
diff
display
echo
edit
edisplay

eprint
eprvcd
fft
fourier
getcwd
gnuplot
hardcopy
help
history
inventory
iplot
jobs
let
linearize
listing
load
mc_source
meas
m[r]dump
noise
op
option
plot

pre_<cmd>
print
psd
quit
rehash
remcirc
remzerov'c
reset
reshape
resume
rspice
rusage
save
sense
set
setcs
setcirc
setplot
setscale
setseed
settype
shell
shift

show
showmod
snload
snsave
source
spec
status
step
stop
strcmp
sysinfo
tf
trace
tran
transpose
unalias
undefine
unlet
unset
version
where
wrdata
write
wrs2p

Interactive mode is governed by setting general options in addition to the "internal variables" described below:

appendwrite
askquit
batchmode
colorN
controlswait
cpdebug
curplot
curplotdate
curplotname
curplottitle
debug
device
diff_abstol
diff_reltol
diff_vntol
echo
editor
filetype
fourgridsize
gridsize
gridstyle
hcopydev
hcopyfont
hcopydevtype
hcopyfontsize
hcopyscale
hcopywidth
hcopyheight
hcopypscolor
hcopypstxcolor
height

history
inputdir
interactive
lprplot5
lprps
modelcard
moremode
nfreqs
ngbehavior
no_auto_gnd
nobjthack
nobreak
noasciiplotvalue
noclobber
noglob
nolegend
nomatch
noparse
noprintscale
nosavecurrents
nosort
nostepsizelimit
nosubckt
notrnoise
numdgt
num_threads
oscompiled
plotstyle
pointchars
polysteps
program

prompt
rawfile
remote_shell
renumber
rndseed
rhost
rprogram
shared_mode
sim_status
sourcepath
specwindow
specwindoworder
spicepath
sqrnoise
strict_err'handl'g
subend
subinvoke
substart
term
ticchar
ticmarks
ticlist
units
unixcom
wfont
wfont_size
width
win_console
x11lineararcs
xbrushwidth
wgridwidth
xfont
xtrtol

5 Extracting Data

$\exists$ several means of getting data out: `.meas` *(interactive-mode only),* `.plot`, `.print`, `.four`, `.probe`, *and* `.save`. `.meas` *is used in interactive mode, whereas* `.plot`, `.print`, `.four`, *in batch, and* `.probe` *and* `.save` *in both modes.*

Save

```
.save [vector]+
```

```
.save i(vin) node1 v(node2)    # save 3 nodes
.save m1[id] vsource#branch    # internal device data
```

Print

Print output variable "vectors":

```
.print prtype [ov]+
```

```
.print tran v(4) i(vin)
.print dc v(2) i(vsrc) v(2, 7)    # volt-Δ, v(2) − v(7)
.print ac vm(4, 2)                # volt-Δ magnitude
```

Plot

```
.plot pltype [ov1 [(plo1, phi1)]?]+
```

```
.plot dc v(4) v(5) v(1)       # no "plo", "phi"
.plot dc emit base out        # using node names
.plot db(coll) ph(coll)       # Bode-style plot
.plot tran v(17, 5) (2, 5)
```

Four[ier]

```
.four freq [ov]+
```

```
.four 100K v(5)          # fundamental freq 100k
```

Probe

```
.probe [vector]+
```

```
.probe i(vin) input output
```

Meas[ure]

`.meas` *defaults to extracting time, use* `FIND` *for another variable. Store measurement in* `result` *parameter, and transform it with specified FUNCtion $\in$ [AVG|MAX[_AT]|MIN[_AT]|PP|RMS|DERIV|INTEG]. Snapshot data at specific time with* `WHEN`*, or bookend a range of time (generally, the abscissa) with triggers using TRIGger .. TARGet, or at specified times using* `FROM .. TO`*.*

```
.MEAS [DC|AC|TRAN|SP] result [FUNC]?
+ [[TRIG|TARG] trigvar VAL=val [TD=td]?
+ [CROSS=num|LAST]? [RISE=num|LAST]?
+ [FALL=num|LAST]? [AT=time]?
```

```
.measure tran tdiff
+ TRIG v(1) VAL=0.5 RISE=1
+ TARG v(1) VAL=0.5 RISE=2
.measure tran teval
+ WHEN v(2)=0.7 CROSS=LAST
.measure tran teval
+ WHEN v(2)=v(1) RISE=LAST
.measure tran yeval FIND v(2)
+ WHEN v(1)=-0.4 FALL=LAST
.meas tran out_slew trig v(out)
+ val='0.2*vp' rise=2
+ targ v(out) val='0.8*vp' rise=2
```

6 Modelling Elements

General format and example:

```
.model [NAME] [TYPE] [PARAM]*
```

```
.model MOD1 npn (bf=50 is=1e-13 vbf=50)
```

Resistor	Lossy TRAnsm.	MOSFET
Capacitor	Diode	P-channel
Inductor (L)	NPN Transistor	MOSFET
V-Controlled	PNP Transistor	N-channel
SWitch	N-channel JFET	MESFET
Cur-Controlled	P-channel JFET	P-channel
SWitch	N-channel	MESFET
Unif. RC Mdl		Powr VD MOS

There are many, many model parameters. For example, you can choose from amongst over 20 distinct MOSFET models, each of which is controlled by a dozen or more parameters. In all, $\exists$ over 100 MOSFET parameters alone. Conversely, inductors are much simpler: $\exists$ only 1 model controlled by only 8 parameters.

7 XSPICE

Package comes with numerous pre-built models. These afford the simulation of digital and analog components that would be computationally-prohibitive if built atop only analog foundational components.

Digital Models

Buffer	Tristate	D Latch
Inverter	Pullup	Set-Reset Latch
And	Pulldown	State Mach'n
Nand	D Flip Flop	Freq. Divid'r
Or	JK FF	RAM
Nor	Toggle FF	Digital Source
Xor	Set-Reset FF	LUT
Xnor		General LUT

Analog Models

Gain	Current Limiter	Ctlrd Trng Osc
Summer	Hysteresis Blk	Ctlrd Sqr Osc
Multiplier	Differentiator	Ctlrd 1-shot
Divider	Integrator	Cap'tance Mtr
Limiter	S-Domn Xfer F	Inductance Mtr
Contrl'd Lmtr	Slew-rate blk	Memristor
PWL Ctrl'd Src	Induct. Coupl'g	2d table
Filesource	Magnetic Core	3d table
Analog Switch	Ctlrd Sine Osc	Simpl Diod Mdl
Zener Diode		

Analog-Digital Bridge Models

D-to-A Node Bridge	Node Bridge, D-to-Real
A-to-D Node Bridge	Node Bridge, Real-to-A
Ctrld Digital Osc	Gain Blk, Event Data
	Z**-1 Blk

User-Created

For new models, create the "Interface Specification File", and "Model Definition File", the first which specifies the interface, the latter which implements it. For new "User Defined Nodes", create a UDN file. The new files must be referenced via the `ngspice/src/xspice/icm/<type>/modpath.lst` *file and then ngspice recompiled. ISFs comprise a table indicating metadata about the model's ports, parameters, and storage variables. MDFs implement the functionality ISFs specify in C code, making use of a number of "accessor macros" (circuit, parameter, and input and output data, etc), and helper functions (for data smoothing, simulator convergence control, ℂ-math, etc). Review XSPICE sourcecode to base new code off of other implementations.*

8 API

Include `sharedspice.h` *in client code. Compile ngspice with --with-ngshared to yield ngshared.lib, which you link to from client code.*

Callbacks

SendChar	SendInitData
SendStat	BgThreadRunning
ControlledExit	SendEvtData
SendData	SendInitEvtData

Commands

API commands prefaced with `ngSpice_`*, suffixed with:*

Init	AllPlots
Init_Sync	AllVecs
running	SetBkpt
Command	Init_Evt
Get_Vec_Info	Get_Evt_NodeInfo
CurPlot	All_Evt_Nodes

Data

Data structs to retrieve simulation data $\in$ `vecinfo`, `vector_info`, `vecinfoall`*.*