Elec 422: VLSI Design I
Fall 1998

November 3, 1998

Additional Information on Spice

1 Spice

Examples of spice usage are located on Owlnet in /home/cavallar/demo/1998/spice.

```
great-gray% cd /home/cavallar/demo/1998/spice
great-gray% ls
README              buffer3.mag         buffer3.spiceallps  invert1.sim
Template.kota       buffer3.model       buffer3.spiceps     invert2.ext
Template.new        buffer3.names       buffer3.vec         invert2.mag
buf_n09a.ps         buffer3.nodes       buffer3_n09a.spcin  invert4.ext
buf_n09aall.ps      buffer3.ps          buffer3f.ir.ps      invert4.mag
buffer3.al          buffer3.pspplot     buffer3r.ir.ps      irsim.results
buffer3.cif         buffer3.sim         crystal.results     spice.results
buffer3.cmd         buffer3.spcin       invert1.al          test2.ps
buffer3.cryvec      buffer3.spcout      invert1.ext
buffer3.exp         buffer3.spcx        invert1.mag
buffer3.ext         buffer3.spice       invert1.nodes
great-gray%
```

1.1 Primary spice example files

- Template.new: updated version of a spice experiment template for Elec422
- buffer3.mag: magic file for chain of inverters. 5 inverters total.
- buffer3.exp: spice experiment file to be used by pspicetool.
- buffer3.spcin: output of pspicetool and input to spice
- README: notes on usage and example interactive commands
- irsim.results, crystal.results: timing info from irsim and crystal

1.2 Spice model files

The following model files are located in /home/cavallar/lib/spice:

- models.n09a: spice parameters from the 1990 9a 2μ n-well MOSIS process.
- models.n13l: spice parameters from the 1991 3l 2μ n-well MOSIS process.

- models.n81x: spice parameters from the 1998 1x 1.2μ n-well MOSIS process.
- defs: information for sim2spice to identify substrate nodes.

Note: the n81x models are level 3 models and are causing difficulty with the present version of spice.

1.3 Pspicetool

There is also the pspicetool shell script mentioned in class.
It is located in /home/cavallar/bin/pspicetool.
Please note that this has not been installed in /usr/site/cad/bin and your default path may not find it. You may want to add /home/cavallar/bin to your path. The contents of pspicetool is:

```
#
# usage for prepspice.  11/88
# updated 11/91
#
#
# Uses default "definition" file for CMOSP CMOSN substrates.
# Uses 1991 MOSIS 2 micron models.
#
# IF your .sim file is called test.sim, then the
# experiment file should be test.exp
#
if ( "$1" == '' ) then
        echo "Usage:    $0 simfilebasename"
        exit (1)
endif
~cavallar/bin/prepspice -d ~cavallar/lib/spice/defs \
                        -m ~cavallar/lib/spice/models.n13l \
                        -e $1.exp  $1
```

The pspicetool shell script takes a single argument which is the .sim file basename. It uses the above defs and models.n13l files. Pspicetool expects that the experiment file based on Template.new will have a .exp filename extension. For example, to build a spice input file from project.sim, create an experiment and place it in project.exp

1.4 Experiment Template.new

You should copy Template.new to your directory and modify it for each spice experiment. Look at /home/cavallar/demo/1997/Template.new for a rough experiment file for spice. You will need this along with your sim file for pspicetool.

Sample session:

```
>> pspicetool buffer3
>> sim2spice -d /home/cavallar/lib/spice/defs buffer3.sim
>> Running spcpp
>> Normal termination from spcpp.
>> Done
```

The file buffer3.mag has 5 inverters in 3 increasing sizes. buffer3.exp defines the experiment, which is the input signal to a chain of inverters.

buffer3.spcin is the final input file to spice.

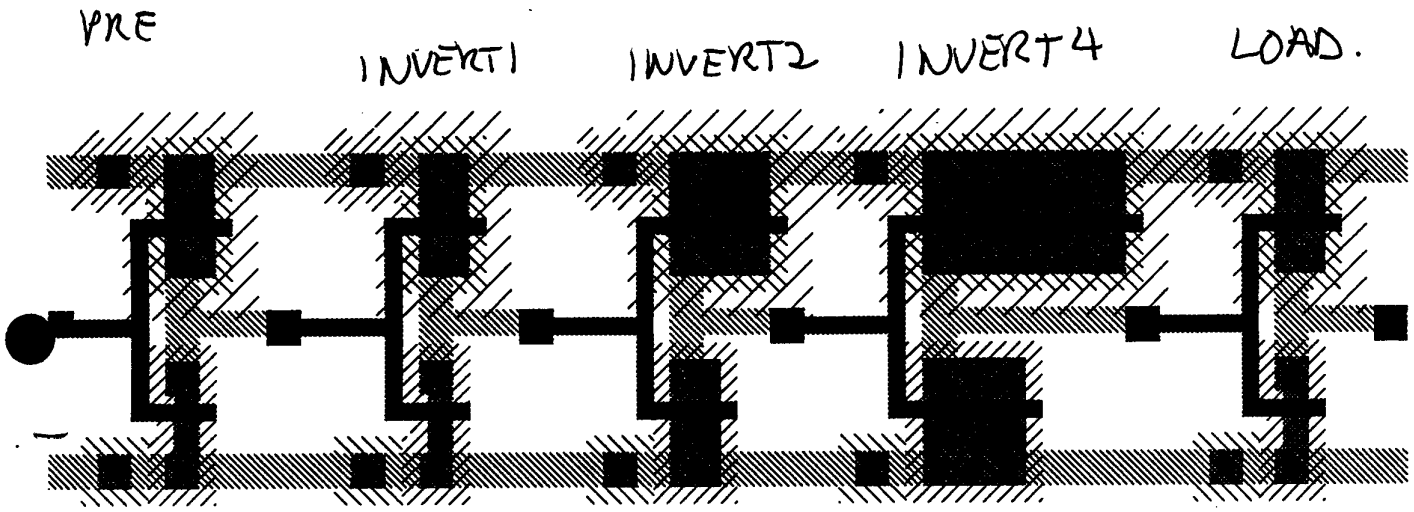
Here are some initial suggestions on running spice from a sample session:

```
>> flammulated% spice
>> Program: Spice, version: 3e1
>> Date built: Tue Aug  6 10:58:27 CDT 1991
>>
>> ** WELCOME TO SPICE-3e1 **
>> (Type "help" for more information, "quit" to leave.)
>> Spice 1 -> source buffer3.spcin
>>
>> Circuit: *** SPICE DECK created from buffer3.sim, tech=scmos
>>
>> Spice 2 -> run
>> Spice 3 -> set hcopydevtype=postscript
>> Spice 4 -> hardcopy buffer3.spiceps v(9) v(5)
>> The file "buffer3.spiceps" may be printed on a postscript printer.
>> Spice 5 -> plot v(9) v(8) v(7) v(6) v(4) v(5)
>> Spice 6 -> hardcopy buffer3.spiceallps  v(9) v(8) v(7) v(6) v(4) v(5)
>> The file "buffer3.spiceallps" may be printed on a postscript printer.
>> Spice 7 -> quit
>> Warning: the following plot hasn't been saved:
>> tran2   *** SPICE DECK created from buffer3.sim, tech=scmos, transient
>>
>> Are you sure you want to quit (yes)?
>> Spice-3e1 done
>> flammulated%
>>
>>
```

Note: spice just reads in the circuit in 1 above. run (in 2) does the analysis. Hardcopy in 4 and 6 just plot to a file and create a Postscript file since I've set the device in 3. The plot command in 5 plots to the X display.

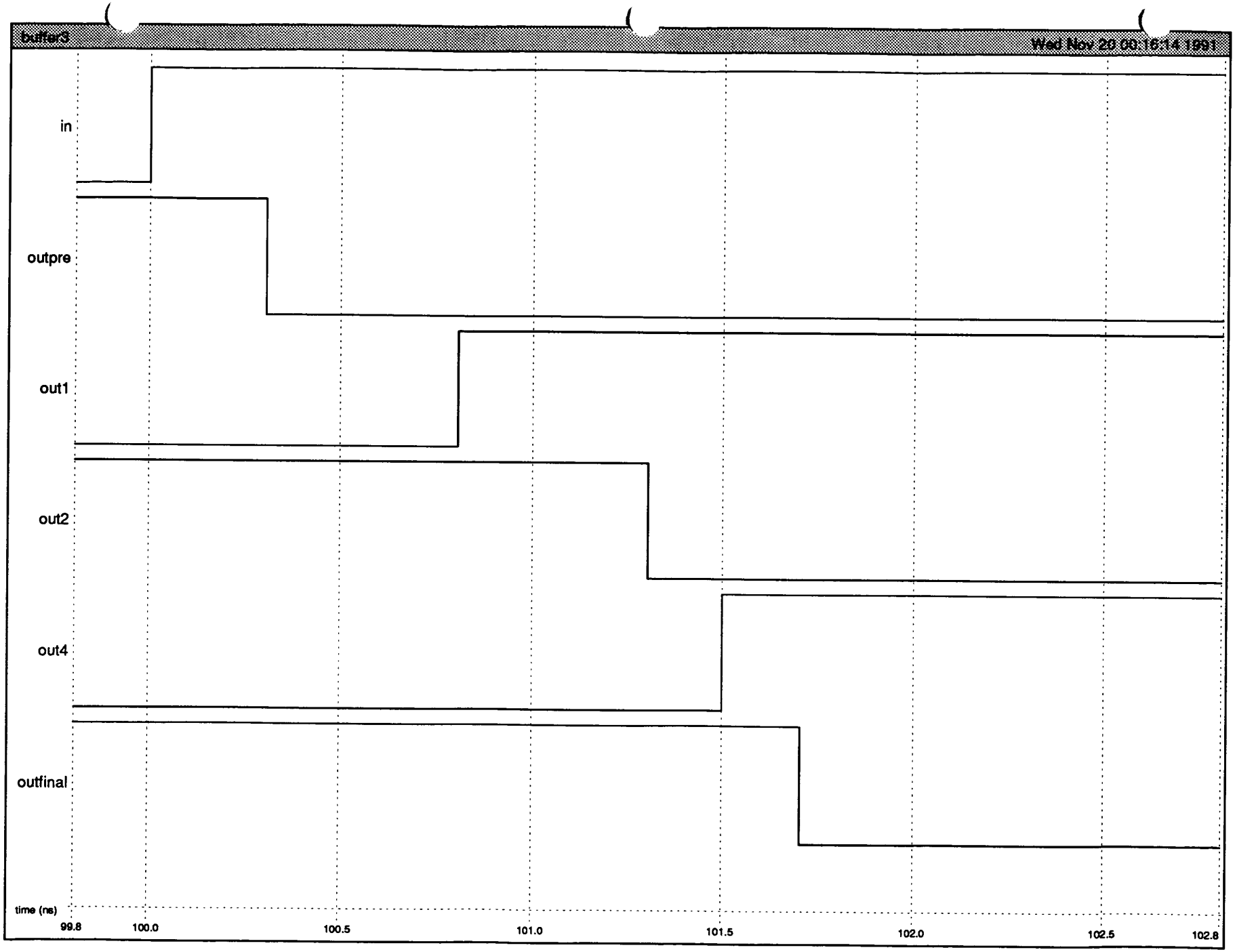
Note too: The "hardcopy" button in the X window plot doesn't seem to work. You need to type the explicit hardcopy command.

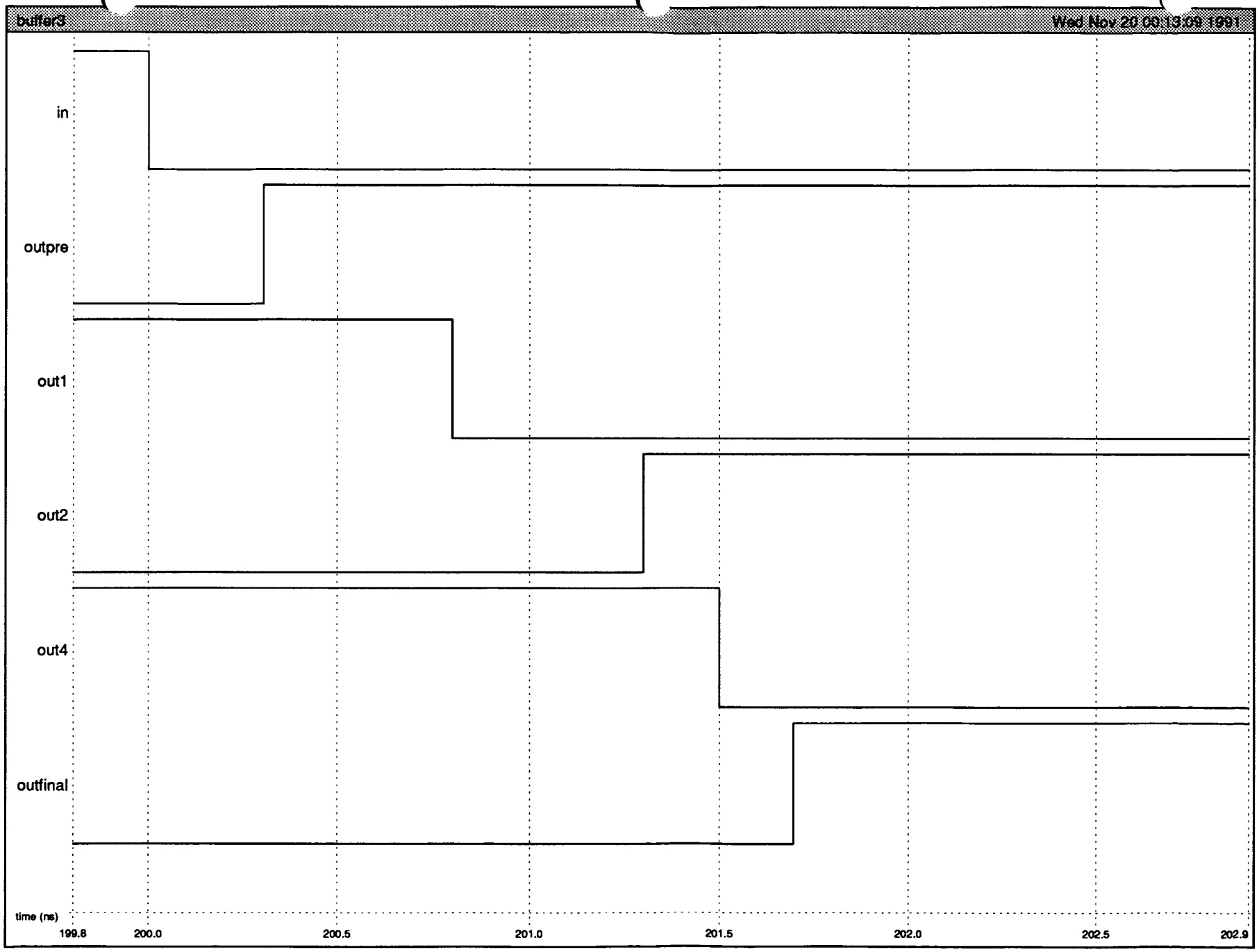
BUFFER 3 CIRCUIT
BUFFERS OF 3 SIZES.



IRSIM.

```
irsim ~cad/lib/scmos100.prm buffer3.sim
*** IRSIM version 8.6 ***
8 nodes; transistors: n-channel=5 p-channel=5
parallel txtors:none
irsim> @ buffer3.cmd
time = 100.0ns
time = 100.0ns
time = 200.0ns
time = 200.0ns
time = 300.0ns
time = 300.0ns
irsim> path outfinal
critical path for last transition of outfinal:
  in -> 0 @ 200.0ns , node was an input
  outpre -> 1 @ 200.3ns     (0.3ns)
  out1 -> 0 @ 200.8ns     (0.5ns)
  out2 -> 1 @ 201.3ns     (0.5ns)
  out4 -> 0 @ 201.5ns     (0.2ns)
  outfinal -> 1 @ 201.7ns     (0.2ns)
irsim>
```


buffer3
Wed Nov 20 00:16:14 1991
in
outpre
out1
out2
out4
outfinal
time (ns)
99.8
100.0
100.5
101.0
101.5
102.0
102.5
102.8


buffer3
Wed Nov 20 00:13:09 1991
in
outpre
out1
out2
out4
outfinal
time (ns)
199.8
200.0
200.5
201.0
201.5
202.0
202.5
202.9

```
flammulated% pwd
/tmp_mnt/net/barn/cavallar/demo/1991
flammulated% crystal buffer3.sim
Crystal, v.2
: build buffer3.sim
[0:00.0u 0:00.0s 15k]
: source buffer3.cryvec
: ! Crystal Test Vectors for buffer3
: input in
[0:00.0u 0:00.0s 24k]
: output outfinal
[0:00.0u 0:00.0s 24k]
: ! Input in falls at 0.00ns
: delay in -1 0
Marking transistor flow...
Setting Vdd to 1...
Setting GND to 0...
(6 stages examined.)
[0:00.0u 0:00.0s 25k]
: critical
Node outfinal is driven high at 3.14ns
      ...through fet at (58, 15) to Vdd after
    out4 is driven low at 2.70ns
      ...through fet at (37, 4) to GND after
    out2 is driven high at 2.34ns
      ...through fet at (22, 15) to Vdd after
    out1 is driven low at 1.63ns
      ...through fet at (8, 4) to GND after
    outpre is driven high at 0.75ns
      ...through fet at (-8, 15) to Vdd after
    in is driven low at 0.00ns
[0:00.0u 0:00.0s 25k]
: ! Input in rises at 0.00ns
: delay in 0 -1
(6 stages examined.)
[0:00.0u 0:00.0s 25k]
: critical
Node outfinal is driven low at 3.14ns
      ...through fet at (58, 4) to GND after
    out4 is driven high at 2.70ns
      ...through fet at (37, 15) to Vdd after
    out2 is driven low at 2.34ns
      ...through fet at (22, 4) to GND after
    out1 is driven high at 1.63ns
      ...through fet at (7, 15) to Vdd after
    outpre is driven low at 0.75ns
      ...through fet at (-8, 4) to GND after
    in is driven high at 0.00ns
[0:00.0u 0:00.0s 25k]
[0:00.0u 0:00.0s 25k]
: quit
[0:00.0u 0:00.1s 25k] Crystal done.
```

CRYSTAL

```
*** SPICE DECK created from buffer3.sim, tech=scmos
M1 1 4 5 3 CMOSP L=2.0U W=6.0U
M2 5 4 0 2 CMOSN L=2.0U W=3.0U
M3 1 6 4 3 CMOSP L=2.0U W=24.0U
M4 4 6 0 2 CMOSN L=2.0U W=12.0U
M5 1 7 6 3 CMOSP L=2.0U W=12.0U
M6 6 7 0 2 CMOSN L=2.0U W=6.0U
M7 1 8 7 3 CMOSP L=2.0U W=6.0U
M8 7 8 0 2 CMOSN L=2.0U W=3.0U
M9 1 9 8 3 CMOSP L=2.0U W=6.0U
M10 8 9 0 2 CMOSN L=2.0U W=3.0U
C11 9 0 12.0F
C12 8 0 42.0F
C13 7 0 41.0F
C14 6 0 54.0F
C15 5 0 31.0F
C16 4 0 87.0F
C17 1 0 180.0F
*  GND  0
*  Vdd  1
*  NMOS 2
*  PMOS 3
*  CMOSN        2
*  CMOSP        3
*  out4 4
*  outfinal     5
*  out2 6
*  out1 7
*  outpre       8
*  in   9
.MODEL CMOSN NMOS LEVEL=2 LD=0.240703U TOX=407.000000E-10
+ NSUB=3.133604E+15 VTO=0.711043 KP=4.879000E-05 GAMMA=0.3801
+ PHI=0.6 UO=575.027 UEXP=0.129885 UCRIT=24726.9
+ DELTA=0.82016 VMAX=63588.7 XJ=0.250000U LAMBDA=2.474695E-02
+ NFS=6.455307E+12 NEFF=1 NSS=1.000000E+10 TPG=1.000000
+ RSH=29.040000 CGDO=3.063325E-10 CGSO=3.063325E-10 CGBO=5.135289E-10
+ CJ=1.130300E-04 MJ=0.699680 CJSW=5.235300E-10 MJSW=0.263078 PB=0.800000
* Weff = Wdrawn - Delta_W
* The suggested Delta_W is -0.60 um
.MODEL CMOSP PMOS LEVEL=2 LD=0.250000U TOX=407.000000E-10
+ NSUB=6.222000E+15 VTO=-0.80612 KP=2.113000E-05 GAMMA=0.5357
+ PHI=0.6 UO=249.519 UEXP=0.226251 UCRIT=18368.3
+ DELTA=1.75689 VMAX=40684.7 XJ=0.250000U LAMBDA=5.307119E-02
+ NFS=3.574015E+11 NEFF=1.001 NSS=1.000000E+10 TPG=-1.000000
+ RSH=87.160000 CGDO=3.181644E-10 CGSO=3.181644E-10 CGBO=6.682147E-10
+ CJ=2.493100E-04 MJ=0.551790 CJSW=2.815500E-10 MJSW=0.311229 PB=0.800000
* Weff = Wdrawn - Delta_W
* The suggested Delta_W is 0.04 um
*
*  Elec560 SPICE EXPERIMENT TEMPLATE;
*  J. Cavallaro 11/88, updated 11/89,
*  updated N.D.Hemkumar & Kota Kishore 11/90, 11/91
*  ------------------------------------

*  Set BASIC VOLTAGE levels
*vdd <Vdd>  <GND>  dc 5
vdd 1      0      dc 5
*  set substrate voltages : P-sub = Vdd;  N-sub = GND
*vs1 <CMOSP> <Vdd>  dc 0
vs1 3       1      dc 0
*vs2 <CMOSN> <GND>  dc 0
vs2 2       0      dc 0

*  ------------------------------------

*  Set other constant circuit inputs, for example b, cin:
```

BUFFER3.SPCIN.

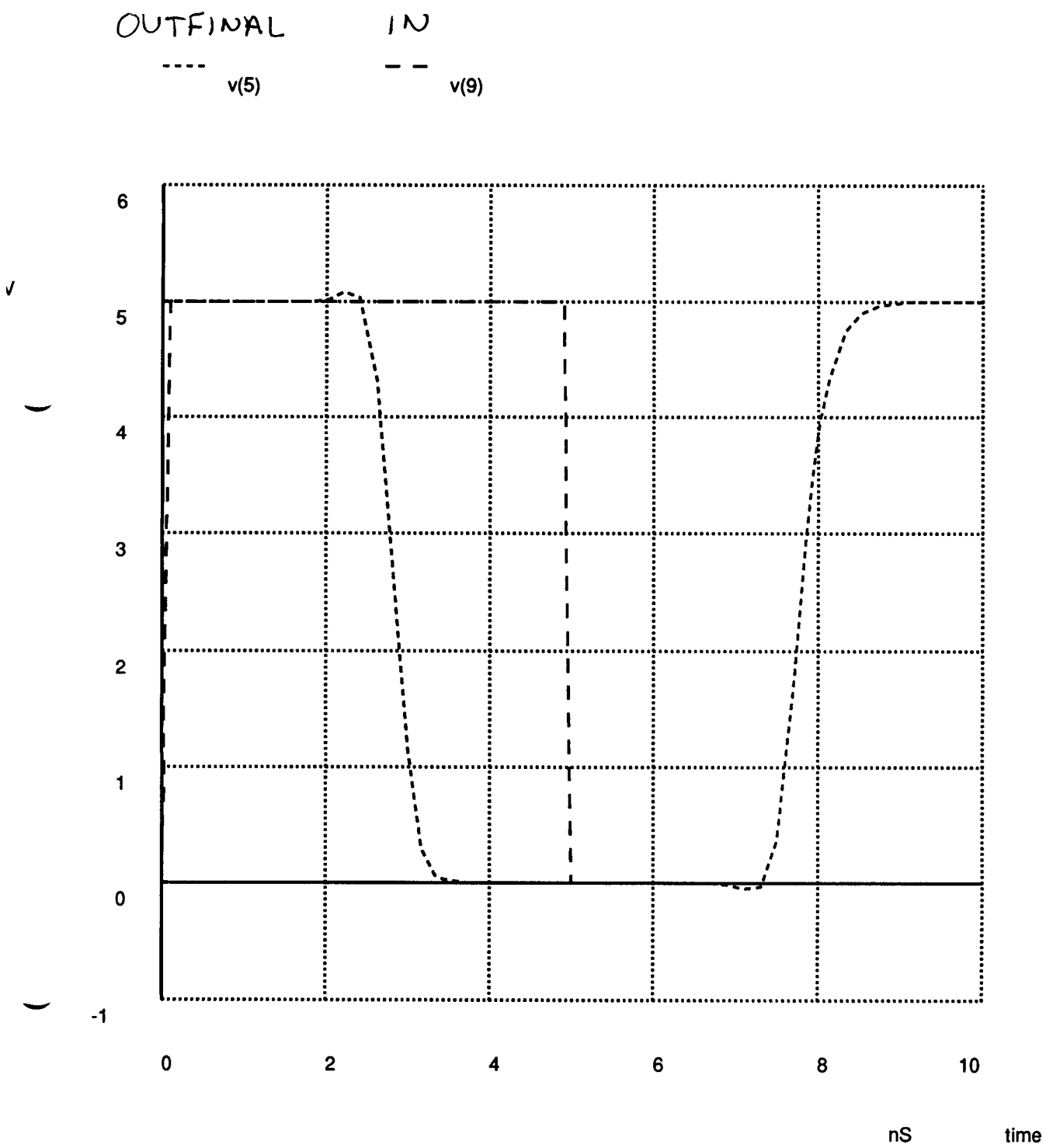
```
*  None here.

*  ------------------------------------

*  Set Circuit Input which will change, for example a:
*  input pulse between node and GND (initially 0 ) of:
*  pulse (init_value pulse_value delay rise_time fall_time pulse_width period)
*vin <in> <GND> 0 pulse(0 5 0ns 0.1ns 0.1ns 4.8ns 10ns)
vin 9    0     0 pulse(0 5 0ns 0.1ns 0.1ns 4.8ns 10ns)

*  ------------------------------------



*  Do analysis: give increments and total time for analysis.
.tran .1ns 10ns
*
*
*  Plot Voltages
*.plot tran v(<in>) v(<outfinal>)
.plot tran v(9)    v(5)

*  set hcopydevtype=postscript


.end
```

```
pspicetool buffer3
sim2spice -d /barn/cavallar/lib/spice/defs buffer3.sim
Running spcpp
Normal termination from spcpp.
Done


flammulated% spice
Program: Spice, version: 3e1
Date built: Tue Aug  6 10:58:27 CDT 1991

** WELCOME TO SPICE-3e1 **
(Type "help" for more information, "quit" to leave.)
Spice 1 -> source buffer3.spcin

Circuit: *** SPICE DECK created from buffer3.sim, tech=scmos

Spice 2 -> run
Spice 3 -> set hcopydevtype=postscript
Spice 4 -> hardcopy buffer3.spiceps v(9) v(5)
The file "buffer3.spiceps" may be printed on a postscript printer.
Spice 5 -> plot v(9) v(8) v(7) v(6) v(4) v(5)
Spice 6 -> hardcopy buffer3.spiceallps  v(9) v(8) v(7) v(6) v(4) v(5)
The file "buffer3.spiceallps" may be printed on a postscript printer.
Spice 7 -> display
Here are the vectors currently active:

Title: *** SPICE DECK created from buffer3.sim, tech=scmos
Name: tran2 (transient)
Date: Wed Nov 20 02:15:22  1991

    V(1)                : voltage, real, 82 long
    V(2)                : voltage, real, 82 long
    V(3)                : voltage, real, 82 long
    V(4)                : voltage, real, 82 long
    V(5)                : voltage, real, 82 long
    V(6)                : voltage, real, 82 long
    V(7)                : voltage, real, 82 long
    V(8)                : voltage, real, 82 long
    V(9)                : voltage, real, 82 long
    m1#drain            : voltage, real, 82 long
    m1#source           : voltage, real, 82 long
    m10#drain           : voltage, real, 82 long
    m10#source          : voltage, real, 82 long
    m2#drain            : voltage, real, 82 long
    m2#source           : voltage, real, 82 long
    m3#drain            : voltage, real, 82 long
        -- hit return for more, ? for help --

    m3#source           : voltage, real, 82 long
    m4#drain            : voltage, real, 82 long
    m4#source           : voltage, real, 82 long
    m5#drain            : voltage, real, 82 long
    m5#source           : voltage, real, 82 long
    m6#drain            : voltage, real, 82 long
    m6#source           : voltage, real, 82 long
    m7#drain            : voltage, real, 82 long
    m7#source           : voltage, real, 82 long
    m8#drain            : voltage, real, 82 long
    m8#source           : voltage, real, 82 long
    m9#drain            : voltage, real, 82 long
    m9#source           : voltage, real, 82 long
    time                : time, real, 82 long [default scale]
    vdd#branch          : current, real, 82 long
    vin#branch          : current, real, 82 long
```

```
    vs1#branch          : current, real, 82 long
    vs2#branch          : current, real, 82 long
Spice 8 -> quit
Warning: the following plot hasn't been saved:
tran2    *** SPICE DECK created from buffer3.sim, tech=scmos, transient

Are you sure you want to quit (yes)?
Spice-3e1 done
flammulated%
```

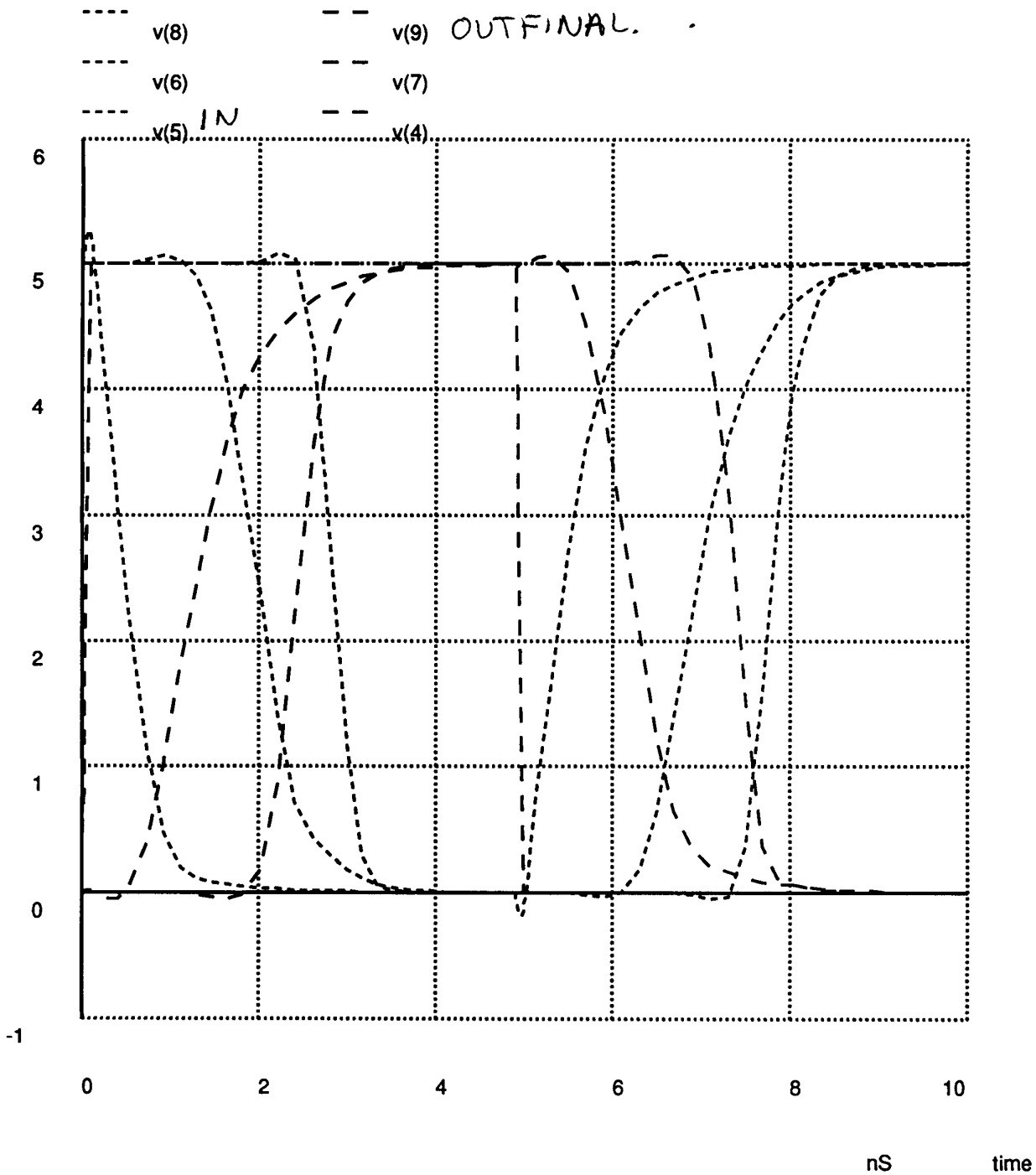
SPICE

N13L
PARAMETERS



SPICE

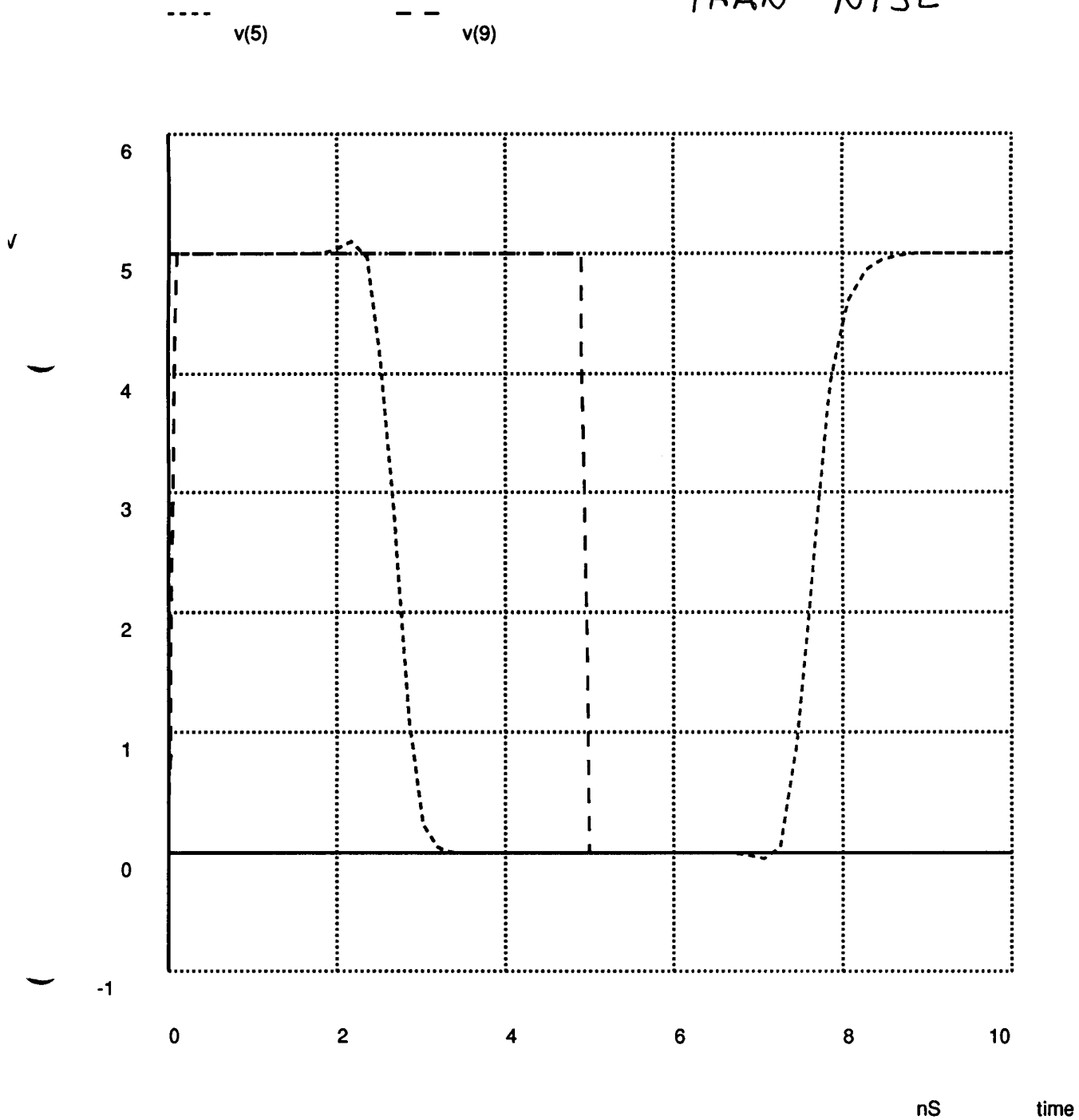
N13L
PARAMETERS



SPICE

N09A
PARAMETERS
COMPARISON

$\approx$ 0.2 ns FASTER
THAN N13L



NAME

pspice – prepare an input file for the Spice circuit simulator

SYNOPSIS

pspice [**–rm**] [**–nos2s**] [**–d** *defsfile*] [**–c** *configfile*] [**–e** *expfile*] *basename*

DESCRIPTION

Pspice is a shell script for preparing Spice input from information from several sources. *Pspice* runs *sim2spice* to convert from a *basename*.**sim** format circuit description to a Spice-compatible description and modifies the *sim2spice* node label translation table to be acceptable Spice comments. It then runs *spcpp* to translate a pseudo-Spice formatted file that contains symbolic node labels to a Spice-acceptable file. Finally, *pspice* concatenates the circuit description file, the translation table, a file of untranslated Spice input, and the translated Spice input into a single file named *basename*.**spcin**. This file is usually an acceptable Spice input file. The optional parameters can be used to cause parts of this process to be skipped.

The options and parameters are:

–nos2s Suppresses the execution of the *sim2spice* step.

–rm Indicates that the files created in intermediate steps are to be deleted.

–d *defsfile* Specifies a file to be used as a *sim2spice* definitions file.

–c *configfile* Specifies a *configfile* that contains SPICE .MODEL cards that are to be included (untranslated) in the final output. In addition, if a *configfile* is specified, *sim2spice* is called with *-c* option. If the *-c configfile* command line option is not used, the technology parameters specified by a *.cadrc* file are used. Otherwise the *configfile* is used to specify the technology parameters. In this case, it is not necessary to provide a *defsfile*, since these statements can be included in the *configfile*. If **sim2spice** is unable to open the *configfile* specified on the command line, the directory $UW_VLSI_TOOLS/lib/technology is searched for *configfile*. If *sim2spice* still can not open it, then it aborts. In case **-c** option is specified without a *configfile* name, a typical config file from the directory $UW_VLSI_TOOLS/lib/technology is read.

–e *expfile* Specifies a file that contains pseudo-input for Spice. *Spcpp* will interpret strings in *expfile* that are bracketed by '<' and '>' as node names to be translated into *spice* node numbers using the translation table (*basename*.**names**) created by *sim2spice*. Lines containing bracketed tokens are converted into Spice comments. It is intended that *expfile* contain Spice commands that describe the experiment to be simulated on the circuit. The ability to use mnemonic node names makes the preparation of Spice input much easier and it means that the description of the experiment need only be specified once, even if the circuit is modified and reextracted. If *expfile* is not specified then spcpp is not executed.

basename Specifies the base name for the files describing the circuit. If *sim2spice* is run then a file named *basename*.**sim** must exist. If *sim2spice* is not run then the files *basename*.**names** and *basename*.**spice** must exist.

FILES

basename.sim	circuit description input to *sim2spice*
defsfile	optional *sim2spice* defs input
basename.names	modified *sim2spice* translation table output. This is read by *spcpp* (*)
basename.spice	*sim2spice* output Spice format circuit element definitions (*)
configfile	file "*basename*.model" is created which contains optional Spice .MODEL commands from *configfile*, to be included in *basename*.**spcin**
expfile	input to *spcpp* containing pseudo-*spice* commands describing the experiment to be simulated
basename.spcx	translated output from *spcpp* (*)

basename.spcin The Spice input deck created by concatenating *basename*.**spice**, *basename*.**names**, *basename*.**model**, and *basename*.**spcx**

Note: Files marked (*) are deleted by the **-rm** option.

SEE ALSO

chconfig(1.vlsi)
sim2spice(1.vlsi), spcpp(1.vlsi)
spice(1.vlsi)
mextra(1.vlsi), cifplot(CAD1)

AUTHOR

Robert Fowler (Northwest LIS, University of Washington)

MODIFICATIONS

Akhilesh Tyagi (Northwest LIS, University of Washington) -- added -c option. deleted -m option.

DIAGNOSTICS

The error messages are intended to be self explanatory. Note that *sim2spice* and *spcpp* produce their own error messages.

BUGS

The command line is long enough to tempt a user to call *pspice* from yet another shell script. A better way to do this is to set up an alias for *pspice* with the commonly used options already set.

NAME

spcpp – Spice (circuit simulator) input pre-processor

SYNOPSIS

spcpp [**–c**] [**–s** *n*] [**–d** *lr*] [**–t** *tname*] [**–o** *oname*] *iname*

DESCRIPTION

Spcpp is a program that translates bracketed text tokens in an input file into other text strings. It is intended to allow users of *spice* to prepare their simulation input using mnemonic node names rather than the numeric node numbers required by Spice commands. The program has two major modes of operation. If the user does not specify a file that contains a translation table, then *spcpp* builds a translation table itself numbering the tokens from zero as it encounters them. Alternatively, the user can specify the name of a file containing a translation table to be used. In particular, the **.names** file created by *sim2spice* is usable as a translation table file.

The options and parameters are:

–c Indicates that the first non-whitespace word of each line of the translation table file should be skipped over. This is useful if your translation table has an asterisk ('*') in column 1 of each line to allow it to be read by *spice* as comments.

–s *n* Indicates that *n* lines at the beginning of the translation table file should be skipped over. If no number is specified then only the first line of the file is skipped.

–d *lr* Redefines the token delimiters to be '*l*' and '*r*' respectively. The default delimiters are '<' and '>'.

–t *tname* Specifies a file that contains a translation table (default is to build a translation table as described above). Each line of this file should have at least two non-whitespace words on it. If the **–c** option is specified then the first word on each line is ignored. The next word is interpreted as a string to be translated and following one is interpreted as the target string into which it is translated. Any subsequent words on the line are ignored. For Spice input preparation the target string should be a numeral. The –s option allows the file to be prefaced by one or more lines that *spcpp* will ignore.

–o *oname* Specifies a file into which the output is to be written. If this option is not used then the output is written to *iroot*.**spcx** where *iroot* is obtained by stripping away any tags from *iname*.

iname Specifies the name of the file to process.

A bracketed token is defined to be a left delimiter character, zero or more spaces, a word (the token) not containing either right or left delimiters, zero or more spaces, and a right delimiter character. Unmatched delimiter characters are not allowed in any context. Bracketed tokens are not allowed to span lines. Tokens and the strings that they translate into are limited to be at most 40 characters each.

Any line that contains no bracketed tokens is simply copied from the input to the output. If a line does contain a bracketed token then the input line is written into the output a Spice comment line. An output line follows immediately. If the line is valid, then the output line has the untranslated parts immediately below the corresponding parts of the commented input line with the target strings substituted for the bracketed tokens. If an error is detected, then the output line has a caret ('^') immediately below the point at which the first error is detected. An error message line then follows. Since the scanning of the line is abandoned there may be subsequent undetected errors in the remaining part of the line.

Example:

If the following lines are contained in the translation table file:

```
Vdd 1
```

```
Input 55
Output 107
foo  23
bar  45
```

then *spcpp* will, upon seeing the lines:

```
.plot trans  v(<Input>) v(<Output>), i(<Vdd>)
+ v(<foo>), v(<bar>)
```

will output the lines:

```
* .plot trans  v(<Input>) v(<Output>) v(<Vdd>)
  .plot trans  v(55) v(107) v(1)
* + v(<foo>), v(<bar>)
  + v(23), v(45)
```

Note that *spcpp* correctly handles Spice continuation cards.

Note also that the substitution process is not recursive. That is, once a token has been translated, the translated string is not rescanned.

The usefulness of *spcpp* for simulating a circuit extracted from a layout depends upon the user being able to ensure that his mnemonic node labels will be retained through the extraction process. The *mextra* and *sim2spice* manual entries will help with this.

Pspice is a shell script that runs *sim2spice* and *spcpp* and concatenates several files is useful for preparing Spice inputs from **.sim** files.

FILES

iname
iroot.spcx
oname
tname

SEE ALSO

mextra(1.vlsi), pspice(1.vlsi), sim2spice(1.vlsi), simtools(1.vlsi), spice(1.vlsi),

SPICE User's Guide, **VLSI Design Tools Reference Manual,** Northwest LIS, University of Washington, (*SPICE Version 2G6 User's Guide*, A. Vladimirescu *et al.*, 15 October 1980)

AUTHOR

Robert Fowler (Northwest LIS, University of Washington)

DIAGNOSTICS

The error messages are intended to be self explanatory. If *spcpp* encounters a syntax error on a line then it suspends processing on that line and writes it as a Spice comment to the output file. It then writes a line containing a caret ('^') under the character at which scanning failed and finally, a line containing an error message. It then goes on to process the remaining lines of the file. If errors have been encountered then at the end of the output file *spcpp* writes messages to the effect that errors have been encountered and exits with status 1. The error messages written to the output file begin with dollar signs. In addition, some number of messages are directed towards the standard error output.

BUGS

The target strings are not checked to see whether they are valid numerals or not. This can be regarded as either a bug or a feature.

The target string must fit into the space from the left to right token delimiter inclusive. This is normally not a problem since most node numbers will be small integers and the available space will be at least three characters. This was done so that the input lines and the translated outputs would line up vertically.

```
*** SPICE DECK created from buffer3.sim, tech=scmos
M1 1 4 5 3 CMOSP L=1.2U W=3.6U
M2 5 4 0 2 CMOSN L=1.2U W=1.8U
M3 1 6 4 3 CMOSP L=1.2U W=14.4U
M4 4 6 0 2 CMOSN L=1.2U W=7.2U
M5 1 7 6 3 CMOSP L=1.2U W=7.2U
M6 6 7 0 2 CMOSN L=1.2U W=3.6U
M7 1 8 7 3 CMOSP L=1.2U W=3.6U
M8 7 8 0 2 CMOSN L=1.2U W=1.8U
M9 1 9 8 3 CMOSP L=1.2U W=3.6U
M10 8 9 0 2 CMOSN L=1.2U W=1.8U
C11 8 0 21.0F
C12 7 0 21.0F
C13 6 0 29.0F
C14 5 0 16.0F
C15 4 0 47.0F
C16 1 0 112.0F
*  GND  0
*  Vdd  1
*  NMOS 2
*  PMOS 3
*  CMOSN        2
*  CMOSP        3
*  out4 4
*  outfinal     5
*  out2 6
*  out1 7
*  outpre       8
*  in   9
.MODEL CMOSN NMOS LEVEL=2 PHI=0.700000 TOX=2.9700E-08 XJ=0.200000U TPG=1
+ VTO=0.5597 DELTA=9.3220E-01 LD=6.5780E-08 KP=7.8748E-05
+ UO=677.3 RSH=2.9330E+01 GAMMA=0.5743
+ NSUB=1.3430E+16 NFS=7.1500E+11 VMAX=1.8650E+05
+ CGDO=1.1472E-10 CGSO=1.1472E-10
+ CGBO=3.3855E-10 CJ=2.7107E-04 MJ=5.2656E-01 CJSW=1.5072E-10
+ MJSW=1.0000E-01 PB=9.1954E-01
* Weff = Wdrawn - Delta_W
* The suggested Delta_W is 8.7580E-07
.MODEL CMOSP PMOS LEVEL=2 PHI=0.700000 TOX=2.9700E-08 XJ=0.200000U TPG=-1
+ VTO=-0.7595 DELTA=2.2970E+00 LD=1.1000E-09 KP=2.3440E-05
+ UO=201.6 RSH=8.4110E-01 GAMMA=0.3238
+ NSUB=4.2700E+15 NFS=6.4990E+11 VMAX=1.9020E+05
+ CGDO=5.0000E-11 CGSO=5.0000E-11
+ CGBO=3.3325E-10 CJ=2.9032E-04 MJ=4.5540E-01 CJSW=1.8518E-10
+ MJSW=1.0904E-01 PB=8.0729E-01
* Weff = Wdrawn - Delta_W
* The suggested Delta_W is 8.5120E-07
*
* N81X SPICE LEVEL3 PARAMETERS --- AMI 1.2micron Run Feb. 1998
*
*
*  Elec560 SPICE EXPERIMENT TEMPLATE;
*  J. Cavallaro 11/88, updated 11/89,
*  updated N.D.Hemkumar & Kota Kishore 11/90, 11/91
*  ------------------------------------

*  Set BASIC VOLTAGE levels
*vdd <Vdd>  <GND>  dc 5
vdd 1      0      dc 5
*  set substrate voltages : P-sub = Vdd;  N-sub = GND
*vs1 <CMOSP> <Vdd>  dc 0
vs1 3       1      dc 0
*vs2 <CMOSN> <GND>  dc 0
vs2 2       0      dc 0

*  ------------------------------------
```

ELEC 422
VLSI

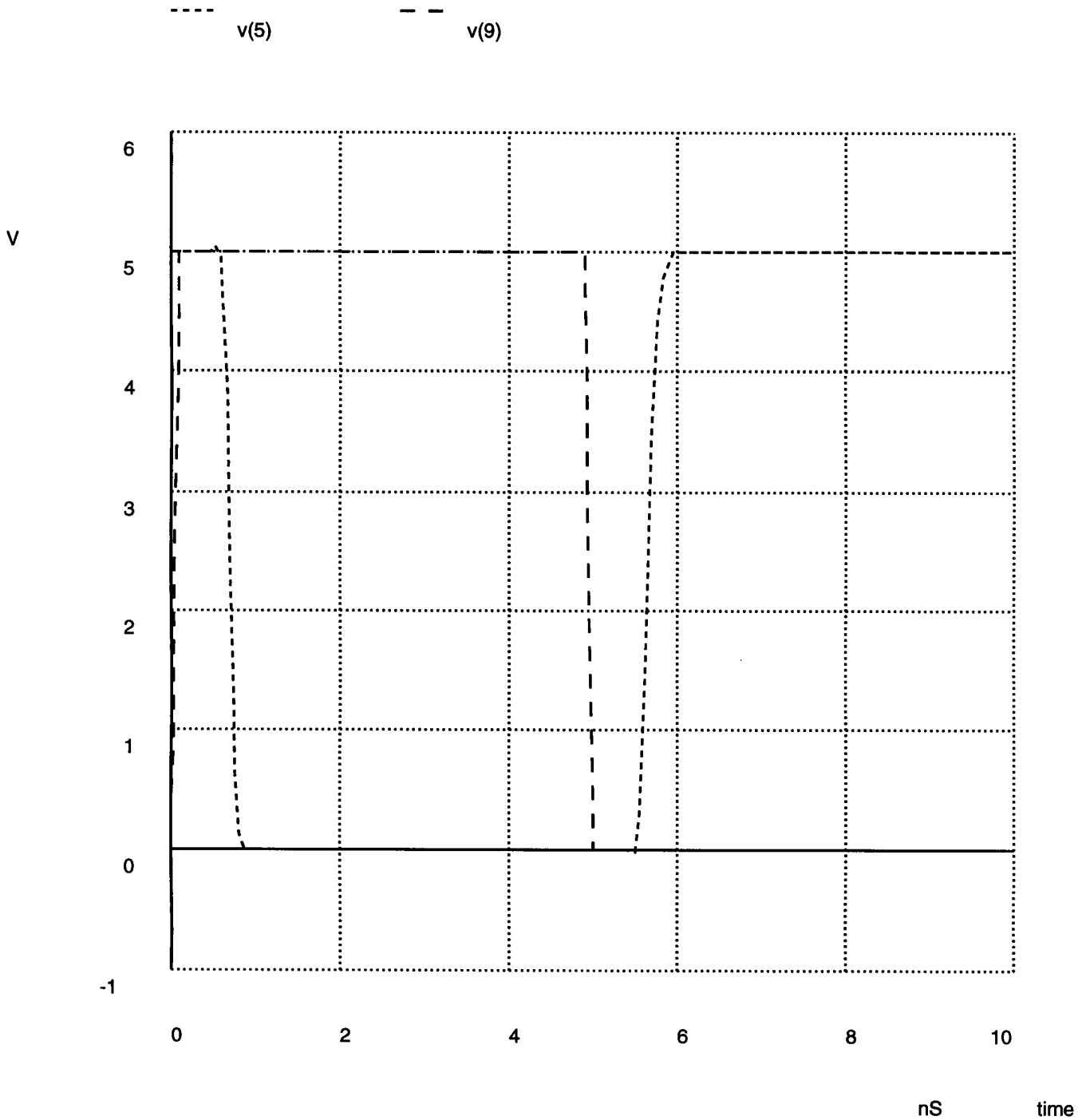
BUFFER 3

AT 1.2μ.

```
*  Set other constant circuit inputs, for example b, cin:
*  None here.

*  ------------------------------------

*  Set Circuit Input which will change, for example a:
*  input pulse between node and GND (initially 0 ) of:
*  pulse (init_value pulse_value delay rise_time fall_time pulse_width period)
*vin <in> <GND> 0 pulse(0 5 0ns 0.1ns 0.1ns 4.8ns 10ns)
vin 9    0     0 pulse(0 5 0ns 0.1ns 0.1ns 4.8ns 10ns)

*  ------------------------------------



*  Do analysis: give increments and total time for analysis.
.tran .1ns 10ns

*  If running in batch mode spice -b, then ascii plots are made
*  Plot Voltages
*.plot tran v(<in>) v(<outfinal>)
.plot tran v(9)    v(5)

*  set hcopydevtype=postscript

.end
```

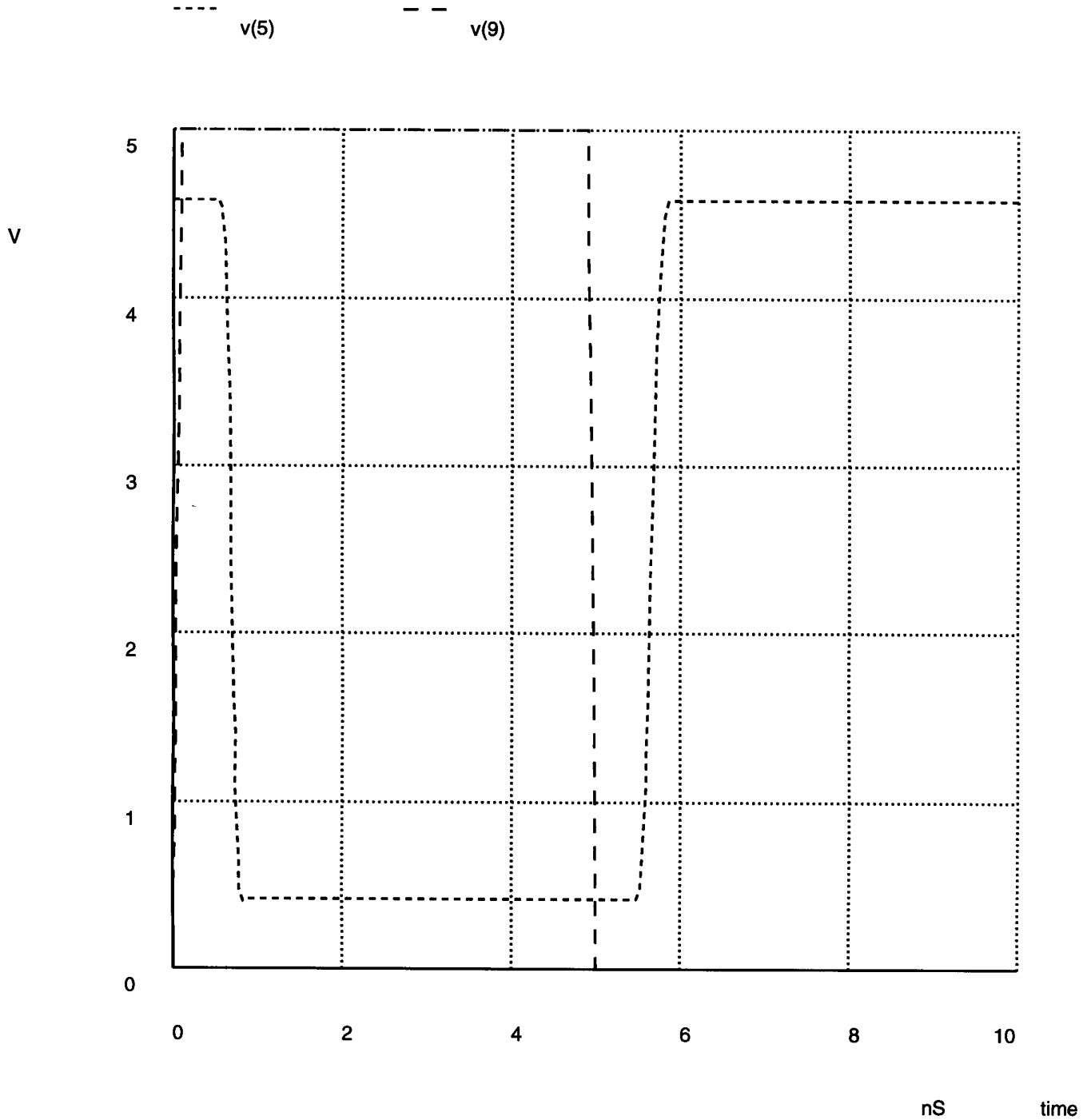
BUFFER 3.

WITH SUBSTRATE CONTACTS.



PSUB AT 1V

NWELL AT 4V



PSUB AT 2V
NWELL AT 4V