DRONACHARYA

Group of Institutions

ELECTRICAL SIMULATION

LABORATORY MANUAL

(EEE-453)

DEPARTMENT OF ELECTRICAL & ELECTRONICS ENGINEERING

27, Knowledge Park-III, Greater Noida, (U.P.)
Phone : 0120-2323854-58
website :- www.dronacharya.info

CONTENTS

SYLLABUS

(PSPICE based)

L	T	P
0	0	2

1. Study of various commands of PSPICE.
2. To determine node voltages and branch currents in a resistive network.
3. To obtain Thevenin's equivalent circuit of a resistive network.
4. To obtain transient response of a series R-L-C circuit for step voltage input.
5. To obtain transient response of a parallel R-L-C circuit for step current input.
6. To obtain transient response of a series R-L-C circuit for alternating square voltage waveform.
7. To obtain frequency response of a series R-L-C circuit for sinusoidal voltage input.
8. To determine line and load currents in a three phase delta circuit connected to a 3-phase balanced ac supply.
9. To plot magnitude, phase and step response of a network function.
10. To determine z,y,g,h and transmission parameters of a two part network.
11. To obtain transient response of output voltage in a single phase half wave rectifier circuit using capacitance filter.
12. To obtain output characteristics of CE NPN transistor.
13. To obtain frequency response of a R-C coupled CE amplifier.
14. To obtain frequency response of an op-Amp integrator circuit.
15. To verify truth tables of NOT, AND or OR gates implemented by NAND gates by plotting their digital input and output signals.

STUDY AND EVALUATION SCHEME

SESSIONAL EVALUATION:-

CLASS TEST	**:**	**10 MARKS**	
TEACHER'S ASSESMENT	**:**	**10 MARKS**	
EXTERNAL EXAM	**:**	**30 MARKS**	
TOTAL	**:**	**50**	**MARKS**

LIST OF EXPERIMENTS

1. Study of various commands of PSPICE.
2. To determine node voltages and branch currents in a resistive network.
3. To obtain Thevenin's equivalent circuit of a resistive network.
4. To obtain transient response of a series R-L-C circuit for step voltage input.
5. To obtain transient response of a parallel R-L-C circuit for step current input.
6. To obtain transient response of a series R-L-C circuit for alternating square voltage waveform.
7. To obtain frequency response of a series R-L-C circuit for sinusoidal voltage input.
8. To determine line and load currents in a three phase delta circuit connected to a 3-phase balanced ac supply.
9. To plot magnitude, phase and step response of a network function.
10. To determine z,y,g,h and transmission parameters of a two part network.
11. To obtain transient response of output voltage in a single phase half wave rectifier circuit using capacitance filter.
12. To obtain output characteristics of CE NPN transistor.
13. To obtain frequency response of a R-C coupled CE amplifier.
14. To obtain frequency response of an op-Amp integrator circuit.
15. To verify truth tables of NOT, AND or OR gates implemented by NAND gates by plotting their digital input and output signals.

INDEX

S.NO.	NAME OF EXPERIMENT	DATE OF EVALUATION	GRADE

EXPERIMENT NO. 1

1) **OBJECTIVE: Study of various commands of PSPICE**

2) **SOFTWARE REQUIRED:**
MicroSim PSpice, version 8

3) **THEORY:**
The following information concerns the text-edited version of MicroSim PSpice, version 8. It is offered here solely for the purpose of helping undergraduate students complete their classroom assignments in computer-aided circuit analysis at the University of Texas at Arlington.
The output file always generated by PSpice is a text (ASCII) file that has the file type "OUT." I.e., if you submit a data file to PSpice named "MYCIRKUT.CIR," it will create an output file named "MYCIRKUT.OUT." This output file is created even if your run is unsuccessful due to input errors. The cause for failure is reported in the *.OUT file, so this is a good place to start looking when you need to debug your simulation model. You examine the *.OUT file with the TextEdit or Notepad programs. When everything works properly, you will find the output results in this file if you are running a DC analysis. If you are running a transient analysis or a frequency sweep analysis, there will be too much data for the *.OUT file. In these cases, we add a command to the *.CIR file that tells PSpice to save the numerical data in a *.DAT file.
The aforementioned *.DAT file is by default a binary (i.e., non-ASCII) file that requires a MicroSim application called PROBE for you to see the data. PROBE is installed with PSpice from the CD-ROM. If you want, you can change the default storage format to ASCII. This is not recommended because it requires more disk space to store the data in ASCII code. Later, we will describe the procedure for invoking PROBE and creating the *.DAT file. A companion file to the *.DAT file is the *.PRB file which holds initializing information for the PROBE program. Another common method used by experienced PSpice users is the use of *.INC (include) files. These enable us to store frequently used subcircuits that have not yet been added to a library. Then we access these *.INC files with a single command line in the *.CIR file. Very convenient. Other files used with PSpice are *.LIB files where the details of complex parts are saved; we may discuss this later, but it is unlikely that we will engage in LIB file alterations until you are taking advanced courses.
When we begin using the schematic capture program that is bundled with PSpice, we will encounter some additional file types. These are the *.SCH (the schematic data, itself), *.ALS (alias files) and *.NET (network connection files).

Node Designations in PSpice

The original SPICE program developed decades ago at U. C. Berkeley, accepted data only on BCD punch cards. That's why it was not case sensitive; developers have preserved this lack of case sensitivity for backward compatibility. In the original SPICE program, users were expected to designate nodes by number.
Most users used small integers, and the numbers did not have to be contiguous. Today, most spice programs accept ordinary text for node designations. If you want to declare a node as "Pbus," you can. The only restriction seems to be that you can't embed spaces in a node name. Use the underscore ("_") character to simulate spaces. Out of habit, most users of PSpice still use small integers as node designations. This often improves the readability of a PSpice source file or output file. In general, you should avoid extremely long textual names for node designations. Naming a node "Arlington_Junior_Chamber_of_Commerce" makes your files look choppy and hard to read. Also, you will sometimes have to *type* that long

cumbersome name when you are performing analysis on the output data file. My suggestion is to use small integers as node designators for most cases. However, use short descriptive names for nodes whenever clarity is improved. "T1_col," when used to designate the collector node of transistor, T1, carries a lot more meaning than "37."

Large and Small Numbers in PSpice

PSpice is a computer program used mostly by engineers and scientists. Accordingly, it was created with the ability to recognize the typical metric units for numbers. Unfortunately, PSpice cannot recognize Greek fonts or even upper vs. lower case. Thus our usual understanding and use of the standard metric prefixes has to be modified. For example, in everyday usage, "M" indicates "mega" (106) and "m" stands for milli (10-3). Clearly, this would be ambiguous in PSpice, since it is not case sensitive. Thus, in PSpice, a factor of 106 is indicated by "MEG" or "meg." "M" or "m" is reserved for 10-3. Another quirk of PSpice is the designation for 10-6. In most publications, the Greek letter, μ, is used for this multiple. Since there can be no Greek fonts (or any other special font designations) in PSpice, the early developers of PSpice borrowed a trick from those who used typewriters. Before the IBM Selectric typewriter was introduced, most writers of technical papers had to improvise for Greek letters. Since the Latin letter "u" (at least in lower case) sort of resembled the lower case Greek μ, it was widely used as a substitute for μ. Hence, either "U" or "u" stands for 10-6 in PSpice. Without further background explanations, these are the metric prefix designations used in PSpice:

Ideal Independent Voltage Sources

We begin with the DC version of the ideal independent voltage source. This is the default form of this class of part. The beginning letter of the part name for all versions of the ideal independent voltage source is "*V*."

This is the character that must be placed in column 1 of the line in the text file that is used to enter this part.

The name is followed by the positive node designation, then the negative node designation, then an optional tag: "DC" followed by the value of the voltage. The tag "DC" (or "dc" if you prefer) is optional because it is the default. Later, when we begin modeling AC circuits and voltage sources that produce pulses and other interesting waveforms, we will be required to designate the type of source or it will default back to DC.

One of the interesting uses of ideal independent voltage sources is that of an *ammeter*. We can take advantage of the fact that PSpice saves and reports the value of current entering the positive terminal of an independent voltage source. If we do not actually require a voltage source to be in the branch where we want to measure the current, we simply set the voltage source to a zero value. It still calculates the current in the branch. In fact, we *require* an ind pendent voltage source in a branch where that branch's current is the controlling current for a current-controlled dependent source.

Examples:

```
*name +node -node type value comment
Va 4 2 DC 16.0V; "V" after "16.0" is optional
vs qe qc dc 24m ; "QE" is +node & "qc" is -node
VWX 23 14 18k ; "dc" not really needed
vwx 14 23 DC -1.8E4 ; same as above
Vdep 15 27 DC 0V ; V-source used as ammeter
```

Resistors

Although PSpice allows for sophisticated temperature-dependent resistor models, we will begin with the simple, constant-value resistor. The first letter of the name for a resistor must be

"R." The name is followed by the positive node, then the negative node and then the value in ohms or some multiple of ohms. The value of resistance will normally be positive. Negative values are allowed in order to permit an alternative model of an energy source. A value of zero, however, will produce an error. Later, we will introduce special resistor models that will permit additional analysis methods to be used. The resistor is not an active device, so the polarity of its connection has no effect on the values of the voltages and currents reported in the solution. However, the current through a resistor is reported as that which flows from the node on the left to the node on the right in the source code line in which the resistor is entered. Thus .PRINT statements and PROBE queries that report resistor current may show negative values of current depending on the order in which you list the resistor's two nodes in the *.CIR file. If you want to see the resistor's current as a positive value, reverse the order of the nodes on the resistor's line in the *.CIR file and re-run the analysis. Nothing else will be affected and both solutions can be correct.

Examples:

```
*name +node -node value comment
Rabc 31 0 14k ; reported current from 31 to 0
Rabc 0 31 14k ; reported current changes sign
rshnt 12 15 99m ; 0.099 ohm resistor
Rbig 19 41 10MEG ; 10 meg-ohm resistor
```

Ideal Independent Current Sources

The name of an ideal independent current source begins with the letter "I" in column 1 of the data file. As with the independent voltage source, we begin by introducing only the DC form of this part, but several other forms exist. Since the current source, is an active element, it matters greatly how it is connected. Designated current flows into the node written on the left, through the current source, out the node written on the right.

As with the independent voltage source, the default type is DC. Remember that the so-called +node on a current source may have a negative voltage with respect to the so-called -node. This is due to the fact that the circuit external to the current source determines its voltage.

Examples:

```
*name +node -node type value comment
Icap 11 0 DC 35m ; 35mA flows from node 11 to 0
ix 79 24 1.7 ; "DC" not needed
I12 43 29 DC 1.5E-4 ;
I12 29 43 dc -150uA ; same as above
```

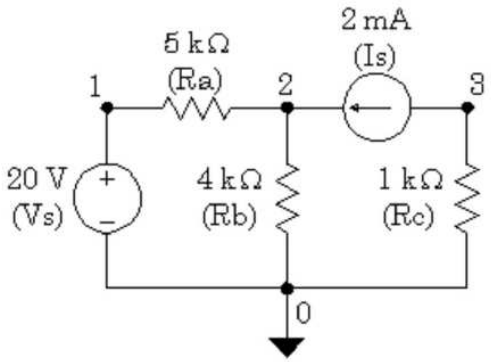
4) <u>RESULTS & DISCUSSION</u>:

EXPERIMENT NO. 2

1) **<u>OBJECTIVE</u>: To determine node voltages and branch currents in a resistive network**

2) **<u>SOFTWARE REQUIRED</u>:**
   PSPICE MANUAL VERSION 8

3) **<u>CIRCUIT & THEORY</u>:**




```
Example_1 EXMPL01.CIR
Vs 1 0 DC 20.0V ; note the node placements
Ra 1 2 5.0k
Rb 2 0 4.0k
Rc 3 0 1.0k
Is 3 2 DC 2.0mA ; note the node placements
.END
```

The output file EXMPL01.OUT is below. This has been edited to remove extra lines.

```
Example_1 EXMPL01.CIR                          <== Title Line
Vs 1 0 20.0V ; note the node placements
Ra 1 2 5.0k
Rb 2 0 4.0k
Rc 3 0 1.0k
Is 3 2 2.0mA ; note the node placements
Example_1 EXMPL01.CIR                          <== Title Line
NODE VOLTAGE NODE VOLTAGE NODE VOLTAGE NODE VOLTAGE
(1) 20.0000  (2) 13.3330  (3) -2.0000        <==Results

VOLTAGE SOURCE CURRENTS
NAME CURRENT
Vs -1.333E-03                <== Current entering node 1 of Vs

TOTAL POWER DISSIPATION 2.67E-02 WATTS
JOB CONCLUDED
TOTAL JOB TIME .26
```

This was the bare-bones minimum problem we could ask of PSpice. Note that we obtained the node voltages which are sufficient information to calculate the resistor currents. However, there is another command that we can use to get even that done by PSpice.

Printing DC Voltages

In addition to printing the node voltages in which you type the letter "V" with the node number in parentheses, you can print the voltage between any pair of nodes; ergo, V(m,n) prints the voltage from node
"m" to node "n."

```
.PRINT DC V(1) V(2) V(3) ; prints the node voltages
.PRINT DC V(1,2) ; prints the voltage across Ra
.PRINT DC V(3,2) ; prints the voltage across Is
```

Printing DC Currents

To print currents, you type the letter "I" with the element name in parentheses. Note that the reported current
is that which flows into the element from the node listed on the left in the *.CIR file, through the element, and
out the node listed on the right in the *.CIR file. If you want to change the sign of the reported current in a
resistor, then swap the two nodes for that resistor.

```
.PRINT DC I(Ra) ; prints the currents from + to - of Ra
.PRINT DC I(Rb) I(Rc) ; prints the currents through Rb and Rc
```

Print Commands can be Combined

```
.PRINT DC V(1,2) I(Ra) ; voltage and current for Ra
.PRINT DC V(2,0) I(Rb) ; V(2,0) same as V(2)
.PRINT DC V(3,0) I(Rc) ; V(3,0) same as V(3)
```

4) RESULTS & DISCUSSION:

EXPERIMENT NO. 3

1) <u>OBJECTIVE</u>: To obtain Thevenin's equivalent circuit of a resistive network.

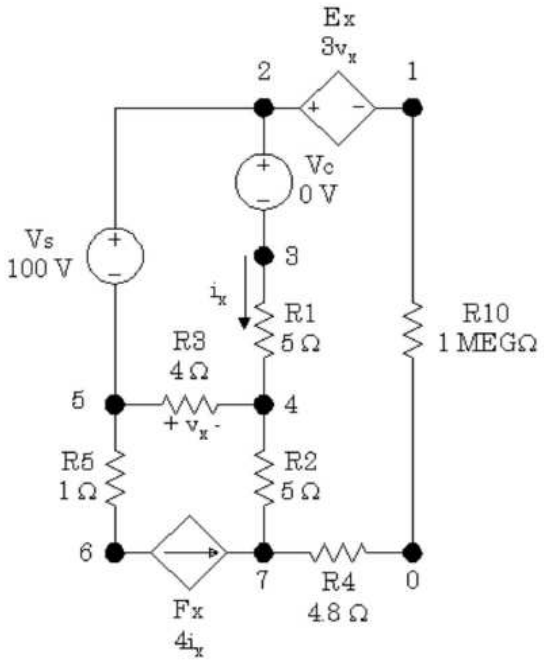
2) <u>SOFTWARE REQUIRED</u>:
PSPICE SOFTWARE VERSION 8

3) <u>CIRCUIT & PROGRAM</u>:

PSPICE can be used to determine the Thévenin resistance and open circuit voltage of a circuit. This can be of great advantage if the circuit is complex, with several dependent sources, or if the circuit cannot be reduced by successive source transformations. The PSpice "dot command" that makes this easy, is ".TF," where "TF" indicates "transfer function." The tran fer function is intended to find the ratio between a source voltage or current, and a resulting voltage difference or branch current. This is useful in characterizing circuits. In addition to reporting the calculated transfer function ratio and input resistance at the source, PSpice reports the *output resistance* at the terminal pair of interest. The voltage across the terminal pair of interest is the Thévenin voltage and the output resistance is the Thévenin resistance. At this point we will ignore the transfer function ratio and the input resistance at the source. In fact, we do not care which source is chosen as long as we only want the Thévenin equivalent circuit parameters. An example of the syntax for the .TF command is shown below.

```
*command output_variable input_source
.TF V(4) Vs
```

The above command will report the ratio between source Vs and node voltage V(4). If we wanted the Thévenin circuit from nodes 4 to 0, the output resistance reported would be our Thévenin resistance, and the voltage V(4) would be the Thévenin (open circuit) voltage. The input source can be a voltage or a current source, and the output variable can be a node voltage, branch voltage or a device current. Now we examine a specific example.



In this example, we want the Thévenin equivalent circuit from nodes 1 to 0. The 1 Megohm resistor is placed in the circuit because PSpice requires at least two connections to each node. This resistor is large enough that it will not have an effect on the calculations. Note the use of voltage source Vc which has the purpose of monitoring the control current, i_x, used for the current-controlled dependent current source, Fx. The input lines in the *.CIR file are shown below.

```
Thevenin Example No. 1
Vs 2 5 DC 100V
Vc 2 3 DC 0V; controls Fx
Fx 6 7 Vc 4.0; gain = 4
* n+ n- NC+ NC gain
Ex 2 1 5 4 3.0; gain = 3
R1 3 4 5.0
R2 4 7 5.0
R3 5 4 4.0
R4 7 0 4.8
R5 5 6 1.0
R10 1 0 1MEG; satisfies PSpice
* out_var input_source
.TF
```

Portions of the output file produced by this case will now be listed.

```
Thevenin Example No. 1
**** CIRCUIT DESCRIPTION
Vs 2 5 DC 100V
Vc 2 3 DC 0V; controls Fx
Fx 6 7 Vc 4.0; gain = 4.0
Ex 2 1 5 4 3.0; gain = 3.0
R1 3 4 5.0
R2 4 7 5.0
R3 5 4 4.0
R4 7 0 4.8
R5 5 6 1.0
Rab 1 0 1MEG
.TF V(1,0) Vs

Thevenin Example No. 1
**** SMALL SIGNAL BIAS SOLUTION TEMPERATURE = 27.000 DEG C
NODE VOLTAGE NODE VOLTAGE NODE VOLTAGE NODE VOLTAGE
( 1) 180.0000 ( 2) -60.0010 ( 3) -60.0010 ( 4) -80.0010
( 5) -160.0000 ( 6) -176.0000 ( 7)-864.0E-06
VOLTAGE SOURCE CURRENTS
NAME CURRENT
Vs -4.000E+00
Vc 4.000E+00
TOTAL POWER DISSIPATION 4.00E+02 WATTS
**** SMALL-SIGNAL CHARACTERISTICS
V(1,0)/Vs = 1.800E+00 <== Transfer function
INPUT RESISTANCE AT Vs = 2.500E+01
OUTPUT RESISTANCE AT V(1,0) = 5.000E+00 <== Thévenin resis-
tance
JOB CONCLUDED
TOTAL JOB TIME .01
```

We conclude that the Thévenin resistance is 5 ohms and the open circuit voltage is 180 volts. Use of the .TF function allows us to get the answers in a single job. The alternative to using the .TF function would be to run one case with a large resistor across the terminal pair of i nterest (if necessary) to get the open circuit voltage; and then run a second case with a zero-valued voltage source across the terminal pair to get the short circuit current. Then divide the short circuit current into the open circuit voltage to get the Thévenin resistance. We prefer the ".TF" method for obtaining Thévenin equivalent circuits.

4) RESULTS & DISCUSSION:

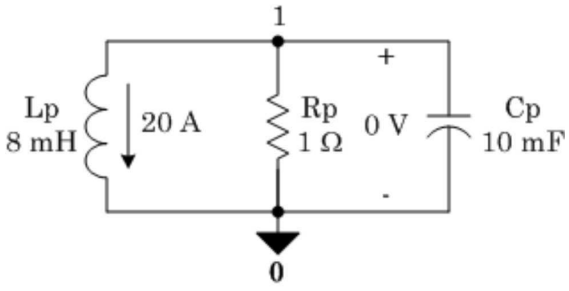
EXPERIMENT NO. 4

1) **<u>OBJECTIVE</u>: To obtain transient response of a parallel R-L-C circuit for step current input.**
2) **<u>SOFTWARE REQUIRED</u>:**

PSPICE VERSION 8

3) **<u>CIRCUIT & PROGRAM</u>:**

One of the most interesting aspects of circuit analysis is the study of natural and step responses of circuits and the responses of circuits to time-varying sources. To perform these analyses we introduce another group of "dot" commands



Use of the .TRAN command

This is the command that passes the user's parameters for performing the transient analysis on a circuit to the PSpice program. There are four time parameters and an instruction to use the initial conditions rather than calculated bias point values for starting conditions. First, we show a sample .TRAN statement and then we will describe its parameters.

```
* prt_stp t_max prt_dly max_stp
.TRAN 20us 20ms 8ms 10us UIC
```

In the above statement, the "20us" value labeled "prt_stp" *(print step)* is the frequency with which data is saved. In this case, the system variables are stored each 20μs of simulation time. The actual time steps used by PSpice may be different from this. The second parameter, "20ms," labeled as "t_max" *(final time)* is the value of time at which the simulation will be ended. Since PSpice starts at t = 0, there will be a total of 20ms time span of simulation for the circuit. The third parameter, "8ms," labeled as "prt_dly" *(print delay)* is the print delay time. In some cases, we do not want to store the data for the entire time span of the simulation. In our sample statement shown above, we ignore the data from the first 8ms of simulation and then store the data for the last 12ms. Most of the time, this parameter is set to zero or not used. The fourth parameter, "10us," labeled as "max_stp" *(max step)* is the maximum time step size PSpice is allowed to take during the simulation. Since PSpice automatically adjusts its time step size during the simulation, it may increase the step size to a value greater than desirable for displaying the data. When the variables are changing rapidly, PSpice shortens the step size, and when the variables change more slowly, it increases the step size. Use of this parameter is optional. The last parameter in our list is "UIC." It is an acronym for "Use Initial Conditions." Unless you include this parameter, PSpice will ignore the initial conditions you set for your inductors and capacitors and will use its own calculated bias point information instead. Note that the use of the letter "s" after the numbers in the .TRAN state-

ment is optional. PSpice assumes these values are seconds and actually ignores the "s." However, it is recommended that you use units until you are extremely familiar with all of these commands and definitions.
Now, we will examine some more .TRAN examples.

```
.TRAN 10ns 500us
```

In the above example, PSpice will save data at each 10ns interval of the simulation starting at t = 0 until the final time of 500μs. I.e., there is no print delay and the user has given full control of the calculation step size to PSpice. In addition, PSpice will calculate its own initial conditions for any inductors and capacitors, ignoring any initial conditions set by the user.

```
.TRAN 50m 2.5 0 10m UIC
```

In the above statement, PSpice collects the data at each 50ms time interval starting from zero up to 2.5s. A zero was required as as a placeholder for the print delay parameter since the maximum step size of 10ms was specified. PSpice will use the designated initial conditions of capacitor voltage and inductor current. Notice that the units were left off the numbers in this statement. Only the prefixes which size the values are needed.

Use of the .PROBE command
In addition to specifying the time parameters for a transient solution of a circuit problem, we need to specify how the data is to be saved. In most cases, this simply means that we include a line in the *.CIR file consisting of ".PROBE." This instructs PSpice to create a data file and store the data it calculates. If we create a circuit listing named "CIRCUIT1.CIR" containing a ".TRAN" statement and a ".PROBE" statement, PSpice will create a file named "CIRCUIT1.DAT" holding the data as well as the usual "CIRCUIT1.OUT" file with basic information about the circuit. By default, the data file created by PSpice is a binary data file; i.e., you can't read it with a text editor. This is the most efficient way of saving the data. However, there is an optional parameter (/CSDF) for the .PROBE statement that causes PSpice to save the data in a Common Simulation Data Format which is a text format that allows you to look at the raw data with a text editor.
However, it will take up more space and PROBE doesn't load it for graphing. You will need to make a second run without the /CSDF parameter if you want to plot the data.
Also by default, .PROBE causes *all* the circuit variables to be saved, including all the variables inside each instance of each subcircuit. In some cases, this can amount to a lot of data. If you simulate a large complex circuit with many parts and need to save data at short time intervals over a very long time span, you can easily create gigabyte-size "DAT" files. To avoid this, you can specify the values you want to save. If the
".PROBE" command is issued without any parameters, everything is saved. If you specify the quantities you want saved, *only* those quantities will be saved. We will now examine some .PROBE statements.

```
.PROBE
```

All the above statement does is the enable PSpice to save everything in a binary DAT file.

```
.PROBE/CSDF
```

The above statement enables PSpice to save everything in a CSDF file that can be opened (and edited) with a text editor. You can

```
.PROBE V(5,23) I(Rx) I(L4)
```

The above statement tells PSpice to save only the voltage drop between nodes 5 and 23, the current through resistor, Rx, and the current through inductor, L4, all in binary format. No other data will be saved.

4) RESULTS & DISCUSSION:

EXPERIMENT NO. 5

1) **OBJECTIVE: To obtain transient response of a series R-L-C circuit for step voltage input.**

2) **SOFTWARE REQUIRED:**
   PSPICE SOFTWARE VERSION 8

3) **CIRCUIT & PROGRAM:**

4) **RESULTS & DISCUSSION:**

EXPERIMENT NO. 6

1) **OBJECTIVE: To obtain transient response of a series R-L-C circuit for alternating square voltage waveform.**

2) **SOFTWARE REQUIRED:**
   PSPICE SOFTWARE VERSION 8

3) **CIRCUIT & PROGRAM:**

4) **RESULTS & DISCUSSION:**

EXPERIMENT NO. 7

1) **<u>OBJECTIVE</u>: To obtain frequency response of a series R-L-C circuit for sinusoidal voltage input.**

2) **<u>SOFTWARE REQUIRED</u>:**
   PSPICE SOFTWARE VERSION 8

3) **<u>CIRCUIT & PROGRAM</u>:**

4) **<u>RESULTS & DISCUSSION</u>:**

EXPERIMENT NO. 8

1) **<u>OBJECTIVE</u>: To determine line and load currents in a three phase delta circuit connected to a 3-phase balanced ac supply.**

2) **<u>SOFTWARE REQUIRED</u>:**
PSPICE SOFTWARE VERSION 8

3) **<u>CIRCUIT & PROGRAM</u>:**

4) **<u>RESULTS & DISCUSSION</u>:**

EXPERIMENT NO. 9

1) **OBJECTIVE: To plot magnitude, phase and step response of a network function.**

2) **SOFTWARE REQUIRED:**
   PSPICE SOFTWARE VERSION 8

3) **CIRCUIT & PROGRAM:**

4) **RESULTS & DISCUSSION:**

EXPERIMENT NO. 10

1) **<u>OBJECTIVE</u>: To determine z, y, g, h and transmission parameters of a two part network.**

2) **<u>SOFTWARE REQUIRED</u>:**
   PSPICE SOFTWARE VERSION 8

3) **<u>CIRCUIT & PROGRAM</u>:**

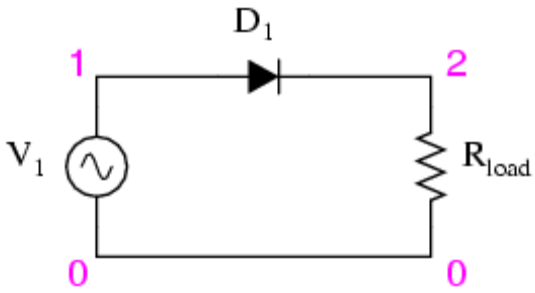
4) **<u>RESULTS & DISCUSSION</u>:**

EXPERIMENT NO. 11

1) **<u>OBJECTIVE</u>: To obtain transient response of output voltage in a single phase half wave rectifier circuit.**

2) **<u>SOFTWARE REQUIRED</u>:**
PSPICE SOFTWARE VERSION 8

3) **<u>CIRCUIT & PROGRAM</u>:**



In half wave rectification, either the positive or negative half of the AC wave is passed, while the other half is blocked. Because only one half of the input waveform reaches the output, it is very inefficient if used for power transfer. Half-wave rectification can be achieved with a single diode in a one-phase supply, or with three diodes in a three-phase supply. Half wave rectifiers yield a unidirectional but pulsating direct current.

```
V1 1 0 SIN(0 10V 100HZ)
R 1 2 1K
DA 0 2 D1
.MODEL D1 D
.TRAN 0.01MS 20MS
.PROBE
.END
```

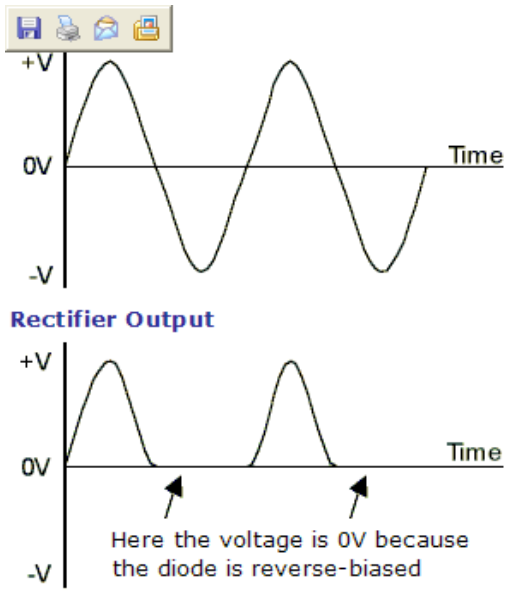
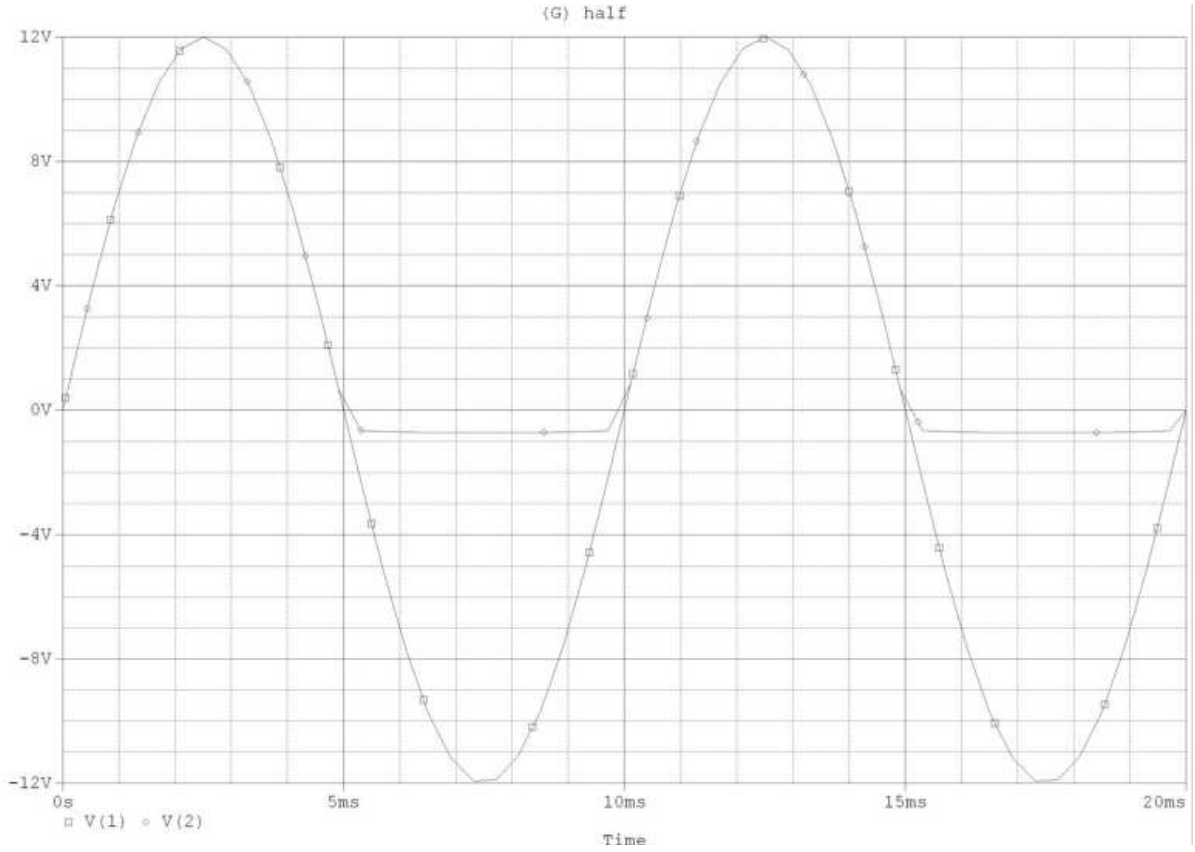


Figure : Half-wave rectification

4) RESULTS & DISCUSSION:



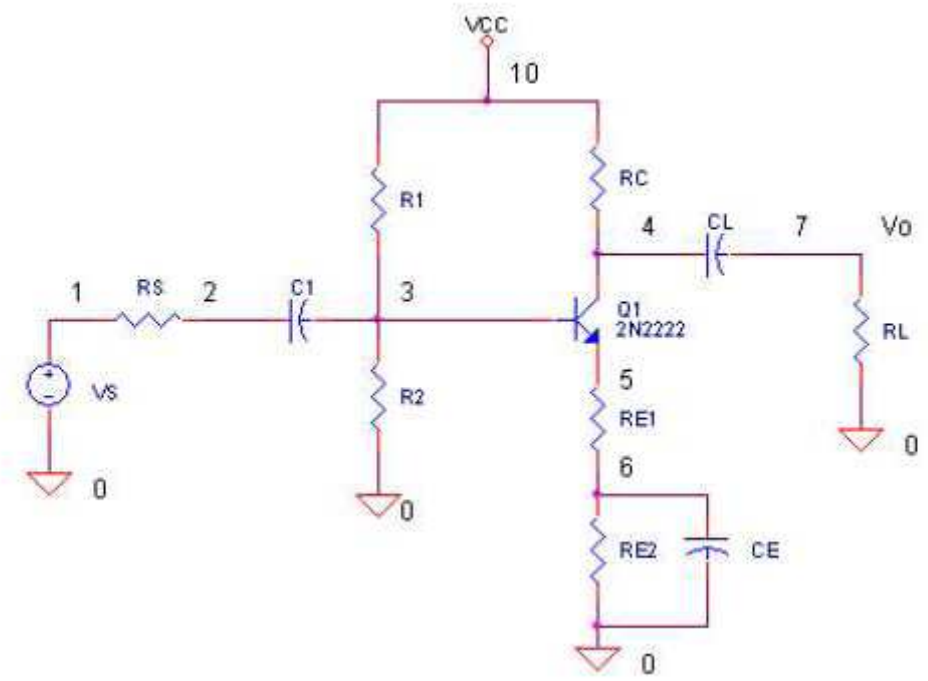
EXPERIMENT NO. 12

1) **OBJECTIVE: To obtain output characteristics of CE NPN transistor.**

2) **SOFTWARE REQUIRED:**

PSPICE SOFTWARE VERSION 8

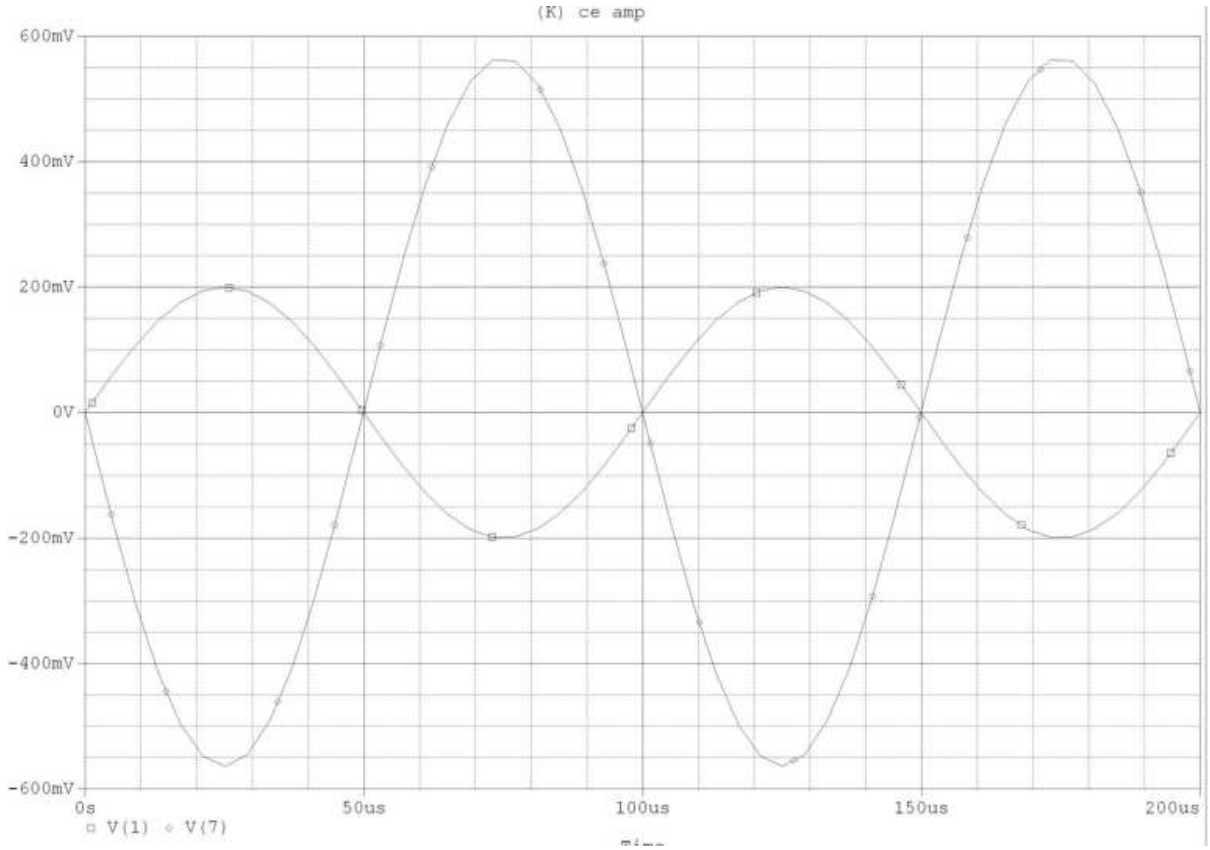
3) **CIRCUIT & PROGRAM:**



The COMMON-EMITTER CONFIGURATION (CE) is the most frequently used configuration in practical amplifier circuits, since it provides good voltage, current, and power gain. The input to the CE is applied to the base-emitter circuit and the output is taken from the collector-emitter circuit, making the emitter the element "common" to both input and output. The CE is set apart from the other configurations, because it is the only configuration that provides a phase reversal between input and output signals.

```
VIN 1 4 SIN(0 1.5V 2KHZ)
VB 4 0 2.3V
RL 3 0 15K
V1 2 0 15V
Q1 2 1 3 MOD1
.MODEL MOD1 NPN
.TRAN 0.02MS 0.78MS
.PROBE
.END
```

4) **RESULTS & DISCUSSION:**

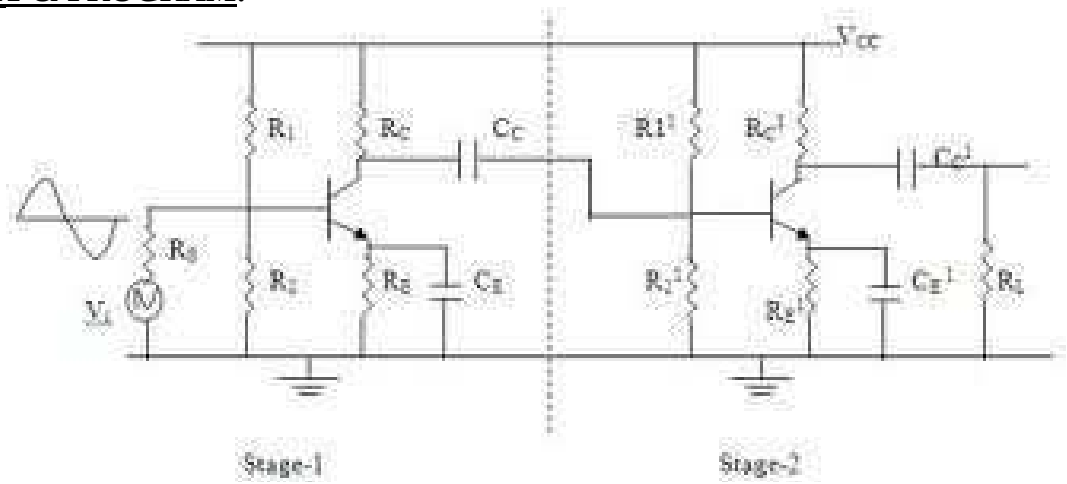

(K) ce amp
600mV
400mV
200mV
0V
-200mV
-400mV
-600mV
0s
50us
100us
150us
200us
V(1)
V(7)

EXPERIMENT NO. 13

1) **OBJECTIVE: To verify the characteristics of RC Coupled Amplifier.**

2) **SOFTWARE REQUIRED:**
PSPICE VERSION 8

3) **CIRCUIT & PROGRAM:**



When a.c. signal is applied to the base of the first transistor, it is amplified and developed across the out of the 1st stage. This amplified voltage is applied to the base of next stage through the coupling capacitor C_C where it is further amplified and reappears across the out put of the second stage. Thus the successive stages amplify the signal and the overall gain is raised to the desired level. Much higher gains can be obtained by connecting a number of amplifier stages in succession (one after the other). Resistance-capacitance (RC) coupling is most widely used to connect the output of first stage to the input (base) of the second stage and so on. It is the most popular type of coupling because it is cheap and provides a constant amplification over a wide range of frequencies. The above shows the circuit arrangement of a two stage RC coupled CE mode transistor amplifier where resistor R is used as a load and the capacitor C is used as a coupling element between the two stages of the amplifier.

4) **RESULTS & DISCUSSION:**

EXPERIMENT NO. 14

1) **OBJECTIVE: To obtain frequency response of an op-Amp integrator circuit.**

2) **SOFTWARE REQUIRED:**

   PSPICE SOFTWARE VERSION 8

3) **THEORY:**

4) **CIRCUIT DIAGRAM:**

5) **PROGRAM:**

6) **OBSERVATIONS:**

7) **CALCULATIONS:**

8) **RESULTS & DISCUSSION:**

9) **PRECAUTIONS:**

10) **PRE-EXPERIMENT QUESTIONS:**

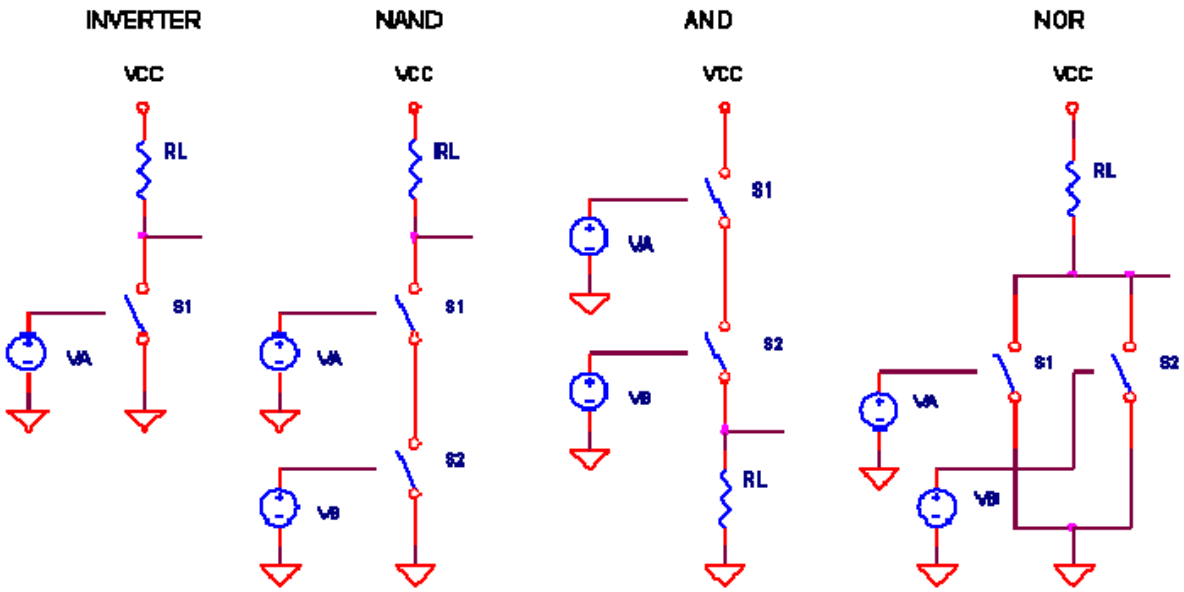
11) **POST-EXPERIMENT QUESTIONS:**

EXPERIMENT NO. 15

1) **<u>OBJECTIVE</u>: To verify truth tables of NOT, AND or OR gates implemented by NAND gates by plotting their digital input and output signals.**

2) **<u>SOFTWARE REQUIRED</u>:**

3) **<u>CIRCUIT</u> :**



You're simulating a circuit, it requires several digital gates, but you don't have a mixed-mode simulator. What to do? One solution involves creating simplified versions of the logic functions.

To do this, we look to the NMOS transistor implementation of logic gates where the transistor acts like a voltage-controlled switch. But, instead of the transistor, we'll use the SPICE switch.

Just like the transistor, the switch is defined to turn ON when the input voltage goes HI. By placing the these switches in parallel or series, a variety of basic logic functions can come to life. Here's some helpful hints for logic circuit building:

FUNCTION OUTPUT

AND - Switches in Series

OR - Switches in Parallel

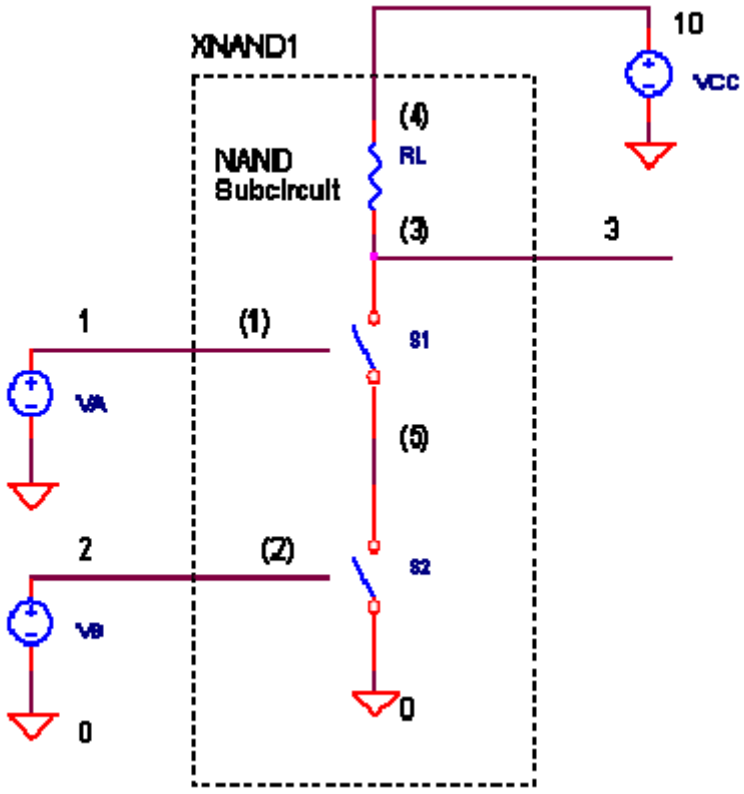
INVERTED - Pull-Up Resistor

NON-INVERTED - Pull-Down Resistor

THE NAND GATE

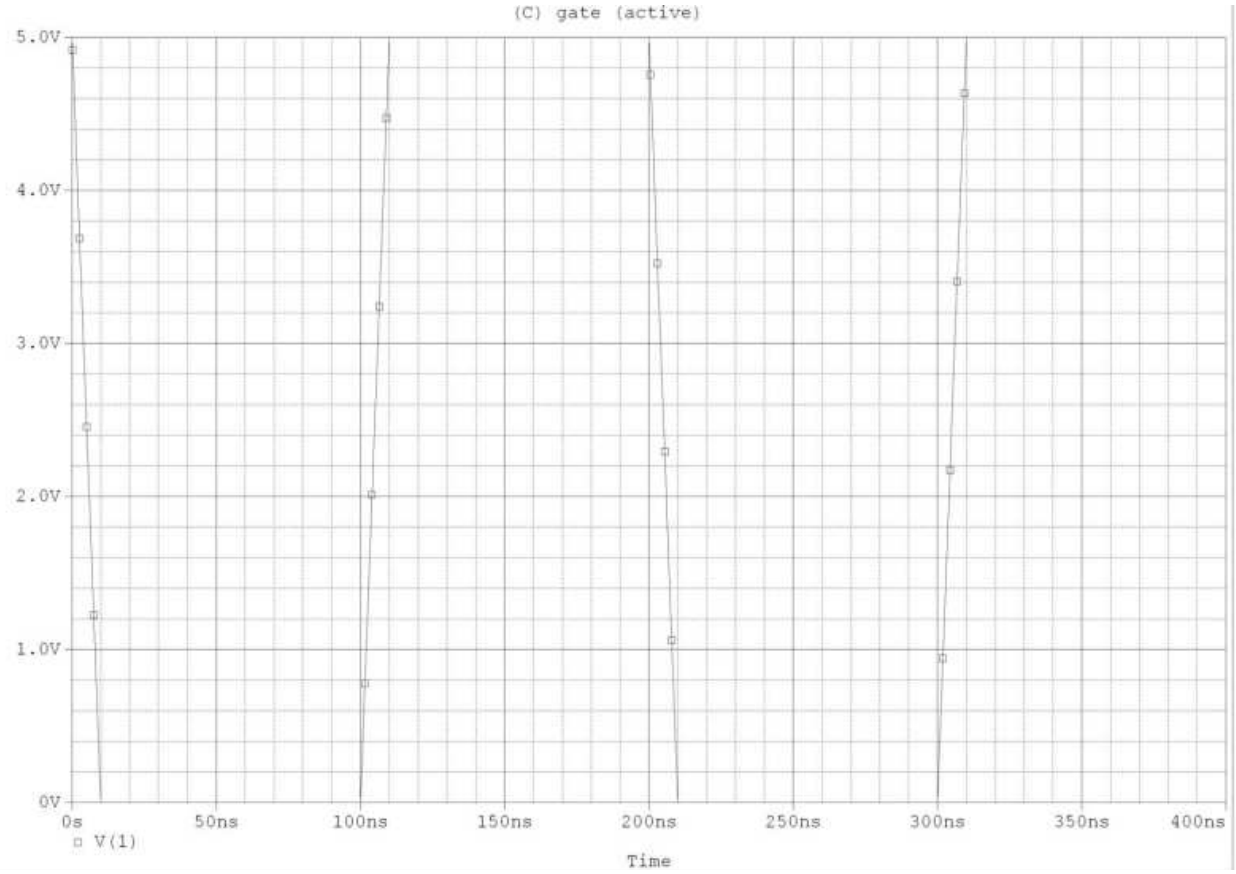
So let's have a go at simulating the NAND gate. How do you describe its function? When both A and B are HI, the output is LO. Or stated another way - it's the AND function with an inverted output. The Boolean expression looks like

The circuit appears below. S1 and S2 in series create the AND function; RL in the pull-up position inverts the output. Defining the NAND gate as a subcircuit makes it easy to insert it into a few locations if you wish. The subcircuit nodes are listed in parenthesis.



S1 and S2 are defined by RON = 10 _ and ROFF = 1 M_. Compared to the other resistances in the circuit, these should look like an ideal switches. More on the SPICE switch below. Simulate the SPICE circuit named LOGIC_SW.CIR. VA and VB create two binary signals that form the sequence 00, 01, 10 and 11. VCC = +5V supplies power to the logic gate. Plot the inputs V(1), V(2) and the output V(3). For a clearer view, you might want to plot V(3) in a separate plot window. Does the output go LO when V(1) and V(2) are HI? Note the finite rise and fall times of V(1) and V(2). You may have also noticed that the output V(3) quickly changes when the inputs pass through 2.5 V. This is the approximate logic threshold level defined for the SPICE switches.

4) RESULT:


(C) gate (active)
5.0V
4.0V
3.0V
2.0V
1.0V
0V
0s
50ns
100ns
150ns
200ns
250ns
300ns
350ns
400ns
V(1)
Time