APPENDIX
A

A Brief Introduction to SPICE

J. Chia and R. Saleh

CHAPTER OUTLINE

A.1 Introduction

The simple design equations employed throughout the text are useful for "back-of-the-envelope" calculations. Typically, hand calculations are anywhere between 30–50% from the results obtained after chip fabrication. For more accurate analysis, more complex equations can be used with parameters that incorporate process and environmental variations. Unfortunately, calculations with complex equations and a large number of parameters can be time-consuming and error-prone. Instead, CAD tools for the simulation and analysis of integrated circuits are used to obtain detailed solutions. The SPICE program is the most popular tool in industry today. Since it is used throughout this book, a brief tutorial is provided here for those unfamiliar with SPICE. This treatment is only intended to introduce the reader to the SPICE tool. It contains many useful features that are not covered in this appendix. There are many good books and reference manuals that provide a comprehensive study of SPICE. The interested reader should consult these references (see Chapter 3) for more details.

This appendix is most useful after reading Chapters 1, 2, and 3 of the textbook. It describes the input syntax for SPICE so that the simulations described in Chapters 3, 4, 5, and 6 may be better understood. The two most useful types of SPICE analyses for digital circuits are the dc and transient analyses. The dc analysis can be used to generate voltage transfer characteristics, current-voltage characteristics, etc. Transient analysis is used to produce time-domain plots of current or voltage versus time. These are the two types of analyses that are described in this appendix. After describing the syntax of a SPICE input file, two examples are presented to illustrate the use of dc and transient analysis.

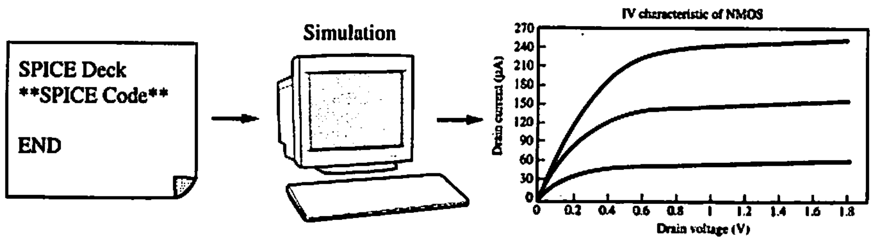


Figure A.1
Typical SPICE design flow.

A.2 Design Flow

Typical SPICE analysis consists of three steps:

1. The circuit is described in SPICE-readable format in a text file known as a SPICE "deck." The deck contains a list of the sources, active elements and passive elements in the circuit, and their respective connections. The deck is also the place where the desired types of analyses (dc, transient) are specified. Today, there are schematic editors that allow you to specify the devices and their connections graphically and then automatically generate the SPICE file from the schematic. This type of input specification should be used if available at your site. We assume that a text file is used as input.
2. The input file is compiled and simulated using the SPICE program. At this point SPICE reads and checks the deck for syntax errors. This is also the stage where the requested simulations are performed. There are many different commercial version of SPICE available. Some run on workstations while others run on personal computers. For the purposes of this textbook, they are all roughly equivalent. We assume a workstation version of SPICE running under the UNIX operating system.
3. Once the deck has been successfully simulated, the results may be analyzed using graphical display programs. SPICE can print out results in text format, but a graphical view of the waveforms is preferred. This form also allows rapid measurement of propagation delays, permits superposition of multiple waveforms on one plot, etc.

Figure A.1 illustrates this design flow.

A.3 Syntax

A sample SPICE deck for an inverter circuit is shown in Figure A.2. Each line will be described briefly and then detailed in the sections to follow.

```
Inverter Circuit
.param Supply=1.8 * Set value of Vdd
.lib 'bsim3v3.cmos.18um' * Set 0.18um library
.opt scale=0.1u * Set lambda (here lambda=0.10)
Vdd   1     0     'Supply'
Vin   2     0     pulse      0     'Supply'   1ns  0ns  0ns  2ns  4ns
Mp 3 2 1 1 pmos L=2 W=8 as=40 ps=4 ad=40 pd=8
Mn 3 2 0 0 nmos L=2 W=4 as=20 ps=4 ad=20 pd=4
Cload      3     0     10fF
* Transient analysis
.tran 50ps 5ns
* Plotting requests
.plot tran V(3)
.end
```

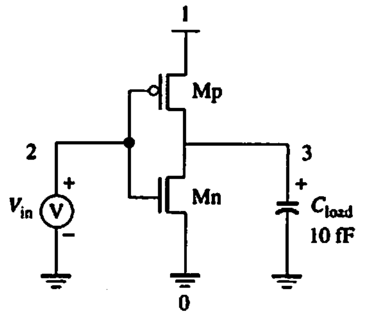


Figure A.2
An example inverter circuit.

A basic SPICE file will contain the following:

1. A title.

   ```
   Inverter Circuit
   ```

2. Settings of various global parameters such as λ, V_{DD}, and the MOS device models to be used. The models to be used are specified by including a library file.

   ```
   .param Supply=1.8 * Set value of Vdd
   .lib 'bsim3v3.cmos.18um' * Set 0.18um library
   .opt scale=0.1u * Set lambda (here lambda=0.10)
   ```

3. A listing of sources, active elements, and passive elements.

```
Vdd 1 0 'Supply'
Mp 3 2 1 1 pmos L=2 W=8 as=40 ps=4 ad=40 pd=8
Mn 3 2 0 0 nmos L=2 W=4 as=20 ps=4 ad=20 pd=4
Cload  3    0    10fF
```

4. Analysis statements (DC, Transient).

```
.tran 50ps 5ns
```

5. Plotting and printing statements.

```
.plot tran V(3)
```

6. Comments.

```
* Transient analysis
```

7. An .end statement.

```
.end
```

A.3.1 Title

The first line of any SPICE file is treated by the compiler as the title of the deck regardless of what the line contains. A descriptive title is best used here with dates, versions, name, etc.

A.3.2 Settings of Various Global Parameters

The three main types of global parameters that can be set are:

1. Parameters, which are floating-point number variables.

```
.param Supply=1.8 * Set value of Vdd
```

2. Libraries, which contain the various parameters used in the MOSFET model.

```
.lib 'bsim3v3.cmos.18um' * Set 0.18um library
```

3. Lambda, the length of the λ unit. Since a 0.18 μm technology is being used, λ is set to 100 nm (for convenience).

```
.opt scale=0.1u * Set lambda (here lambda=0.10)
```

A.3.2.1 Parameters

A parameter in SPICE is a declaration of a global variable that can be assigned to values such as the voltage of a voltage source, the resistance of a resistor or in .dc and .tran commands. The syntax for a parameter declaration is:

```
.param      paramname=realnumber
```

param	All parameter declarations must begin with this.
Paramname	The name of the parameter (e.g.. Supply, Length, Delay).
realnumber	The value of the parameter (e.g., 1.2, 0.1u, 10n).

The example below declares a parameter called "Supply" with a value of 1.8. Note that no units have been assigned. If Supply is assigned to a voltage source, then that voltage source has a value of 1.8 V, if Supply is assigned to an inductor, then the inductor has a value of 1.8 H.

```
.param      Supply=1.8
```

A.3.2.2 Libraries

Libraries contain all the parameter values of the MOS model for a particular technology to be used when designing your circuits. In the past, users could set the model parameters such as oxide thickness, junction capacitances, and doping levels. However, there are so many parameters associated with a model that are tied closely to the foundry service that they are now stored in library files. Users should not adjust these values but should check out the models by generating typical *I-V* curves to ensure that the model libraries are consistent.

Libraries are declared using the .lib command. The syntax for a library call is

```
.lib        '<filepath> filename'
```

`.lib`	All library calls must begin with this.
`filepath`	This is the location of the library file. You should enter this path if the library file is not in the same folder as the deck being compiled. You can use "../" if the library file is in one of the parent directories. You must enclose the path and name in single quotes as shown above.
`filename`	This is the name of the library file. You can use "../" if the library file is in one of the parent directories. You must enclose the path and name in single quotes as shown above.

The example below includes a library called bsim3v3.cmos.18um that is held in the parent directory of the current deck.

```
    .lib '../bsim3v3.cmos.18um'
```

A.3.2.3 Specifying Device Sizes in Lambda Units

SPICE provides an .opt scale command that will allow you to more conveniently express the physical dimensions of MOS transistors in units of lambda. The dimensions will be scaled by this value.

```
.opt scale=length
```

`.opt scale`	The lambda declaration must begin with this.
`length`	Value of the unit of lambda in meters.

The example below sets the value of lambda to 0.1 μm.

```
.opt scale=0.1um
```

Later in the description, if the following line is used,

```
Mn 3 2 0 0 nmos L=2 W=4 as=20 ps=4 ad=20 pd=4
```

it is interpreted as a transistor with $L = 2\lambda$, $W = 4\lambda$, $AS = 20\lambda^2$, $PS = 4\lambda$, $AD = 20\lambda^2$ and $PD = 4\lambda$. For scale $= 0.1\ \mu$m, it would have the same effect as using the following line:

```
Mn 3 2 0 0 nmos L=0.2u W=0.4u as=0.2p ps=0.4u ad=0.2p pd=0.4u
```

However, the first description is much easier to read and debug. It also allows you to change the technology dimensions without modifying the circuit description.

A.3.3 Listing of Sources and Active and Passive Elements

The circuit can be constructed by typing a list of sources (voltage or current sources), active elements (diodes and transistors), passive elements (resistors and capacitors), and their associated connection nodes. Each type of source and element is specified on a separate line. Sources and elements are added to the list by specifying:

1. A unique instance name.
2. The nodes to which they are connected.
3. Specific values associated with the element (such as 10 kΩ for a resistor.)
4. A model, if an active element is being specified (such as the BSIM model for MOS transistors).
5. Element specific parameters (such as the width and length of a MOSFET).

A.3.3.1 Instance Name

The instance name of an element is a unique designation that may contain up to 8 alphanumeric characters. The first character of the name must correspond to the type of element that it is being invoked because this is how SPICE will know the type of element you want to describe. For example, the first letter of a resistor instance must begin with the letter R. Any unique combination of characters may follow after.

Table A.1 lists the element type and name convention. 'xxx' can be any combination of up to 7 alphanumeric characters.(This is the syntax for all instance names in SPICE.)

A.3.3.2 Nodes

To specify the connectivity information for the circuit, the nodes to which the element is connected are specified. Most elements require two nodes, while other elements like BJTs and FETs require three or more. In the past, nodes were specified as positive integer values. For a more descriptive input, alphanumeric node names should be used whenever possible.

Table A.1
Syntax of element names

Element Type	Naming Convention	Instance Examples
Voltage Source	Vxxx	V1, VDD, Vsource
Current Source	Ixxx	I2, Isource, ID
Resistor	Rxxx	R5, Rin, Rout
Capacitor	Cxxx	Cg, Cout, Cbig
Inductor	Lxxx	L3, Lself, L1
MOSFET	Mxxx	M1, Mpmos23, Mpu
Diode	Dxxx	D1, Dpn, Dclamp

Note: Node "0" is reserved for the ground node. For some two terminal elements, the "positive" node must be entered before the "negative" node. "Positive" and "negative" do not necessarily refer to the voltage level of the nodes with respect to each other, rather they refer to the designated "+" and "−" side of the element. Voltage sources and diodes are examples of devices that alter their circuit topology if their nodes are specified in different orders. The nodes on the resistors, capacitors, and inductors, on the other hand, may be specified in different order without any effect on the way the circuit behaves. For these elements, it is customary to enter the node that will likely be at the higher voltage as the positive side.

Figure A.3 shows a 10 kΩ resistor named R5 connected between node "5" and ground.

```
R5      5      0      10k
```

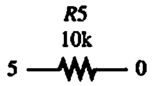


Figure A.3
A resistor passive element.

A.3.3.3 Values

Element values are specified using floating-point numbers. Values that are extremely large or small (e.g., > 1000 or < 0.001 may be specified in exponential format or with engineering prefixes. Table A.2 shows multiplying factors with their associated power-of-ten suffix letter, metric prefixes, and exponential format. The last column shows two equivalent values expressed in engineering and exponential format.

Note: Units (ohms, farads, volts) are not required since SPICE will infer the units based on the type of element (resistors, capacitors, voltage sources). However, they should be specified wherever possible to improve readability. Prefix and exponential formats *cannot* be mixed (e.g., 2E-16P).

Table A.2
SPICE prefixes and exponents

Multiplying Factor	Metric Prefix	Power-of-Ten Suffix Letter	Exponential	SPICE Examples
10^{9}	Giga	G, g	E9	4.2GHz 4.2E9
10^{6}	Mega	Meg, meg	E6	.2Meg .2E6
10^{3}	Kilo	K, k	E3	27kohms 27E3
10^{-3}	Milli	M, m	E-3	273mA 273E-3
10^{-6}	Micro	U, u	E-6	.02us .02E-6
10^{-9}	Nano	N, n	E-9	4ns 4E-9
10^{-12}	Pico	P, p	E-12	95pF 95E-12
10^{-15}	Femto	F, f	E-15	3fF 3E-15

A.3.3.4 Models

For active elements like diodes and transistors, instead of specifying a single value (like resistance or capacitance) a model is specified. The model is added after the nodes have been specified. Then, a list of device dimensions are provided.

Figure A.4 shows a MOS transistor that uses the "nmos" model.

```
Mn    2    3    0    0    nmos l=2 w=4 as=20 ad=20 ps=4 pd=4
```

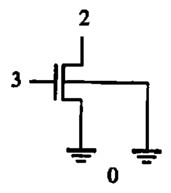


Figure A.4
A MOS transistor.

Ask your system administrator for the file path and name for the model library.

A.3.3.5 Element Specific Parameters

Elements like MOSFETs have additional parameters like their length and width. To specify these values, provide the parameter to be specified, followed by an equal "=" sign and the value of the parameter as a real number. The syntax of the parameter is the same as that specified in the Values section above. The example to follow specifies an NMOS device named "Mn" connected between nodes 2 and 3, using the model "nmos," with a length of 2λ and a width of 4λ. The areas of the source and drain are typically specified by multiplying the width by 5λ (source/drain extension). The perimeters of the source and drain should be specified as the width of the device in deep submicron technologies.

```
Mn   2   3   0   0   nmos l=2 w=4 as=20 ad=20 ps=4 pd=4
```

Sometimes, there may not be enough space on one line to specify all the parameters. Additional parameters may be placed on subsequent lines by starting the line with the plus "+" sign and then entering the rest of the parameters. The example below specifies the same NMOS device but with the dimensional parameters on a separate line.

```
Mn   2     3     0       0      nmos
+    l=2   w=4   as=20   ad=20   ps=4  pd=4
```

A.3.3.6 Element Syntax

The syntax for the elements are:

A.3.3.6.1 Voltage and Current Sources

```
Vxxx   n+   n-   voltageval
Ixxx   n+   n-   currentval
```

Vxxx Ixxx	The name of the voltage or current source.
n+	The positive node.
n-	The negative node.
voltageval currentval	Voltage or current value of the source. This may be a number or a constant previously declared with a .param command.

Figure A.5 shows a voltage source named "Vsupply" connected between nodes 5 and ground with a value of 15 volts and a current supply named "Isupply" connected between nodes 2 and 3 with a value of 'Icc' amps that was declared with a .param command (not shown).

```
Vsupply   5   0   15
Isupply   2   3   'Icc'
```

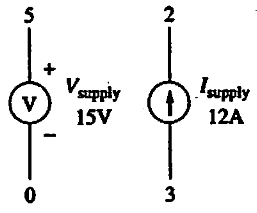


Figure A.5
A voltage and current source.

Sometimes, it is useful to observe the behavior of elements as they are being excited by a trapezoidal waveform. In this case, the PULSE option could be used to specify the parameters of the wave. The syntax for a voltage or current pulse is

```
Vxxx   n+   n-   PULSE   <v1 v2 td tr tf pw per>
Ixxx   n+   n-   PULSE   <v1 v2 td tr tf pw per>
```

Vxxx Ixxx	The name of the voltage or current source.
n+	The positive node.
n-	The negative node.
PULSE	This keyword replaces "voltageval" and signifies that a pulse is being specified.
v1	The initial value of the voltage or current in volts or amps.
v2	The final value of the voltage or current in volts or amps.
td	The delay time in seconds. This is the time of the beginning of the first transition between v1 and v2.
tr	This is the rise time of the pulse, in other words, how long it takes to go from v1 to v2.
tf	This is the fall time of the pulse, in other words, how long it takes to go from v2 to v1.
pw	This is the pulse width of the waveform which is defined as the time the wave remains at v2.
per	This is the period of the waveform.

The example below describes a voltage source called "Vpulse" connected between nodes 3 and 0 that switches between −1 and 1 V. It begins to rise after a delay of 2 ns, with a rise time of 4 ns, a fall time of 3 ns, a pulse width of 20 ns, and a period of 50 ns. A graph of this wave is shown in Figure A.6.

```
Vpulse  3  0  PULSE  -1  1  2ns  4ns  3ns  20ns  50ns
```

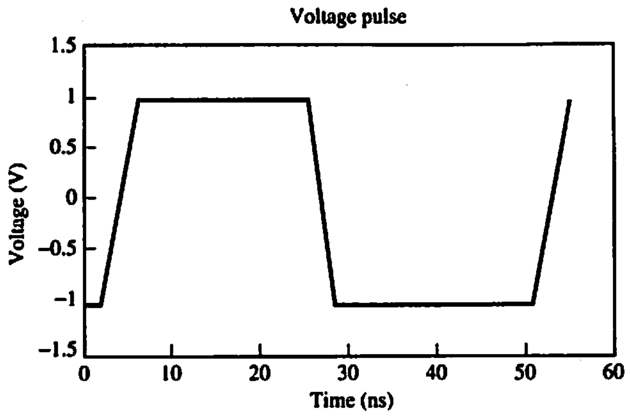


Figure A.6
A pulse waveform.

Another useful source is the piecewise linear source or PWL. The waveform is specified as (time, value) pairs. For the same graph in Figure A.5, the PWL would be specified as follows:

```
Vpulse    3    0   PWL 0 -1 2ns -1 6ns 1 26ns 1 29ns -1
+ 52ns -1 56ns 1
```

A.3.3.6.2 Resistors, Capacitors, and Inductors

```
Rxxx    n+    n-    resistanceval
Cxxx    n+    n-    capacitanceval
Lxxx    n+    n-    inductanceval
```

Rxxx Cxxx Lxxx	The name of the resistor, capacitor, or inductor.
n+	The positive node.
n-	The negative node.
resistanceval capacitanceval inductanceval	Resistance, capacitance, or inductance value of the element.

Figure A.7 shows a 17 MΩ resistor, a 1.5 fF capacitor, and 0.7 mH inductor connected in series.

```
R7      1    2    17Megohms
Cin     2    3    1.5fF
Lbig    3    4    .7mH
```

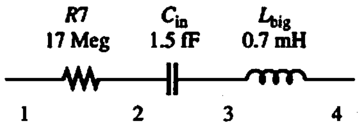


Figure A.7
An *RLC* circuit.

A.3.3.6.3 MOSFET

```
Mxxx  nd   ng   ns    nb mname L=length   W=width
+     AD=drain_area   PD=drain_perimeter
+     AS=source_area  PS=source_perimeter
```

Mxxx	The name of the MOSFET.
nd	The drain node.
ng	The gate node.
ns	The source node.
nb	The bulk node.
mname	The name of the model to be used. This is also where you distinguish between a PMOS and NMOS FET.
length	The length of the MOSFET in units of *m*.
width	The width of the MOSFET in units of *m*
drain_area	The area of the drain diffusion of the MOSFET in units of m^2.
drain_perimeter	The drain perimeter of the MOSFET in units of *m*.
source_area	The area of the source diffusion of the MOSFET in units of m^2.
source_perimeter	The source perimeter of the MOSFET in units of *m*.

Figure A.8 shows a PMOS and an NMOS connected in a transmission gate configuration. Both MOS devices have a $20\lambda^2$ diffusion area and a 5λ diffusion extension. They are both 4λ by 2λ in size.

```
Mp   3     0     2     1     PMOS
+    L=2         W=4
+    AS=20       PS=4
+    AD=20       PD=4
Mn   2     1     3     0     NMOS
+    L=2         W=4
+    AS=20       PS=4
+    AD=20       PD=4
```

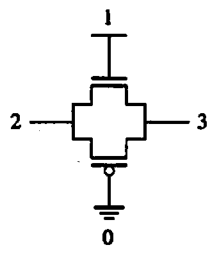


Figure A.8
A transmission gate. For simplicity, the bulk nodes are not shown.

A.3.4 Analysis Statements

In order to instruct SPICE to perform various simulations on a given circuit, SPICE provides analysis commands. The two analyses of particular interest in this book are:

1. transient
2. dc

You may invoke multiple transient and dc requests in your SPICE file.

A.3.4.1 Transient

Transient analysis is used to observe the value of a variable as it changes with time. For example, one may want to view the voltage of the capacitor in an RC network as a step input is applied, or the current through a MOS device as it is switching. For this type of analysis, the user must specify the interval over which the simulation will be performed, and the time step to be used for plotting purposes. SPICE internally solves the differential equations for the circuit numerically and produces very accurate results.

An example of a transient analysis for an RC circuit is shown in Figure A.9.

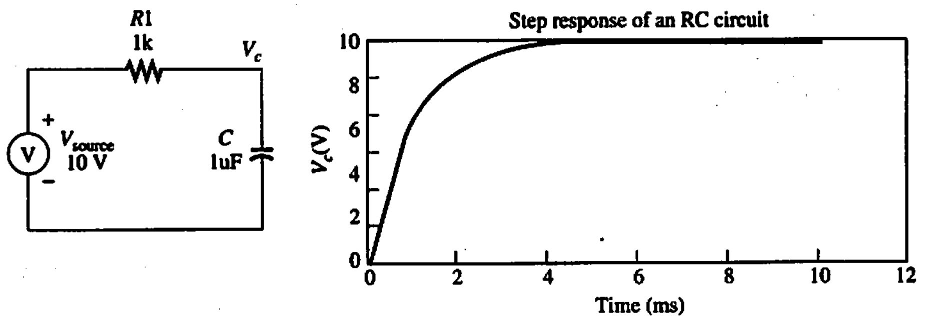


Figure A.9
An RC circuit and a graph of V_c while V_{source} is swept from 0 V to 10 V.

The syntax for the .tran analysis statement is:

```
.tran tstep   tstop
```

`.tran`	All transient statements must begin with this directive.
`tstep`	This value represents the suggested computing increment for SPICE. If this value is 1 second, then spice will calculate the variables of the circuit (voltage, current) every 1 second starting from t = 0. If the increment unit is extremely large or small, prefixes may be used, (e.g., one millisecond would be 1ms, and 200 nanoseconds would be 200 ns).
`tstop`	This value represents the time at which transient analysis will stop. If tstep = 1 and tstop = 10, then SPICE will calculate the variables of the circuit (voltage, current) every 1 second starting from t = 0 to t = 10. If the increment unit is extremely large or small, prefixes may be used, (e.g., one millisecond would be 1ms, and 200 nanoseconds would be 200 ns).

For the RC circuit example above, the syntax for the command would appear as follows:

```
.tran  1m  10m
```

A.3.4.2 DC

DC analysis is used to observe the value of one variable while another is changing or "sweeping" from one value to another. This is very useful when generating the voltage transfer characteristics of an inverter, or plotting the *I*-*V* characteristics of the MOS transistor. In this type of analysis, the user specifies the output variable to be plotted and the variable to be swept from a start value to a stop value, with a given step size. Internally, SPICE carries out the dc solution at each intermediate point between the start and stop values and displays the results.

For example, in the voltage divider circuit in Figure A.10, the voltage V_r between R1 and R2 is observed while V_{source} is swept from 0 V to 10 V.

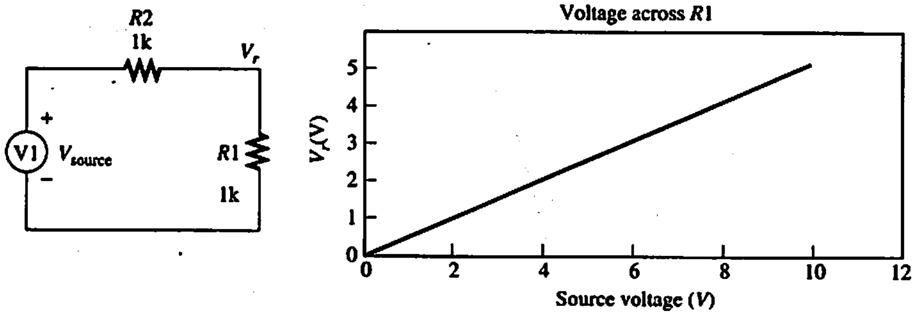
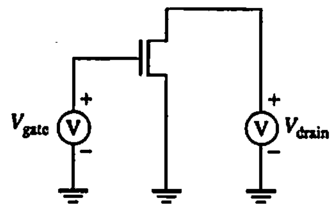


Figure A.10
A graph of V_r while V_{source} is swept from 0 V to 10 V.



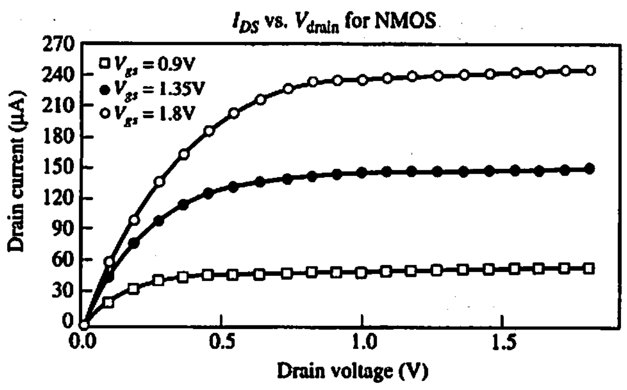


Figure A.11
I_{DS} versus V_{drain} for changing Vgate.

In SPICE, a value can be observed while *two* other values are being swept. This would producé multiple lines on the same graph. In Figure A.11, the drain current is observed while the drain voltage, V_{drain} is swept from 0 to 1.8 V and the gate voltage, V_{gate}, is swept from 0 to 1.8 V.
The syntax for the DC analysis statement is:

```
.dc var1 start stop step <var2 start2 stop2 step2>
```

where everything in <> is optional.

`.dc`	All DC statements must begin with this directive.
`var1`	Name of the variable to be swept (e.g., Vsource).
`start`	The starting value of the sweep (e.g., 0).
`stop`	The end value of the sweep (e.g., 10).
`step`	The magnitude of the increment used to sweep the variable (e.g., 1). The smaller the value, the analysis becomes more accurate but also more time-consuming.
`var2 (optional)`	Name of the second variable to be swept (e.g., R2).
`start2 (optional)`	The starting value of the second sweep (e.g., 1).
`stop2 (optional)`	The end value of the second sweep (e.g., 3).
`step2 (optional)`	The magnitude of the increment used to sweep the second variable (e.g., 1). The smaller the value, the analysis becomes more accurate but also more time-consuming.

For the first voltage divider example above, the syntax for the command would appear as follows:

```
.dc   V1   0   10   1
```

For the second example, the syntax would appear as

```
.dc Vdrain 0 'Supply' 'Supply/20' Vgate 0 'Supply' 'Supply/4'
```

This will sweep Vdrain from 0 to 'Supply' in 'Supply/20' increments, and Vgate from 0 to 'Supply' in 'Supply/4' increments. For the case of two sweep variables, the number of graphs is determined by the sweep parameters of the second variable.

A.3.4.3 Plotting and Printing Statements

Plotting and printing statements allow you to view the results of your transient or dc analysis. The .plot statement displays the results in graphical form while .print displays them in tabular. The basic syntax is shown below:

```
.plot     analysis_type   ov1    <ov2 ... ov32>
.print    analysis_type   ov1    <ov2 ... ov32>
```

where everything in < > is optional.

`.plot`	All plot statements must begin with this.
`.print`	All print statements must begin with this.
`analysis_type`	This is the type of analysis you wish to display, either tran or dc.
`ov1`	These are the output variables you wish to plot, such as the voltage of a node or the current through an element.

The first example below is for a dc plot request of the voltage at node 5 and the current through the inductor `Lself`. The second example is a transient print request for the same voltage and current:

```
.plot     dc      V(5)    I(Lself)
.print    tran    V(5)    I(Lself)
```

A.3.4.4 Comments

As in every other coding language, comments are useful to clarify the author's intention of the various lines in the file to the reader. In SPICE, the beginning of a comment is denoted by an asterix (*). Comments can exist on a line by themselves or with SPICE executable statements:

```
* This is a comment **
R5   2   0   10k    * This is also a comment.
```

A.3.4.5 .ic Statement

There are situations where the initial condition of a circuit node must be specified at time $t = 0$ for a meaningful simulation. For this purpose, the .ic statement can be used. It must also be flagged in the .tran line with a "uic" directive so that the initial condition will be used during the simulation. For example, if node 5 must be initially set to 1.8 V, and node 7 set to 0 V, the following lines would be used:

```
.ic   V(5)=1.8   V(7)=0
.tran 50ps 5ns uic
```

A.3.4.6 .end Statement

To finish off the deck the .end statement is required to tell SPICE that this is the end of the file. Any lines entered after .end are ignored.

A.4 Complete SPICE Examples

Figure A.12 shows a dc analysis on the *I-V* characteristic of an NMOS by sweeping Vdrain from 0 to 1.8 V at five different values of Vgate:

```
IV Characteristic of NMOS
.param Supply=1.8          * Set value of Vdd
.lib 'bsim3v3.cmos.18um'  * Set 0.18um library
.opt scale=0.1u            * Set lambda (here lambda=0.10)
* Voltage sources
Vdrain   2   0   'Supply'
Vgate    1   0   'Supply'
* A minimum width NMOS transistor
Mn 2 1 0 0 nmos l=2 w=4 as=20 ps=4 ad=20 pd=4
* DC analysis request by sweeping the drain and gate voltage.
.dc Vdrain 0 'Supply' 'Supply/20' Vgate 0 'Supply' 'Supply/4'
* Plotting requests
.plot dc I(Mn)
.end
```

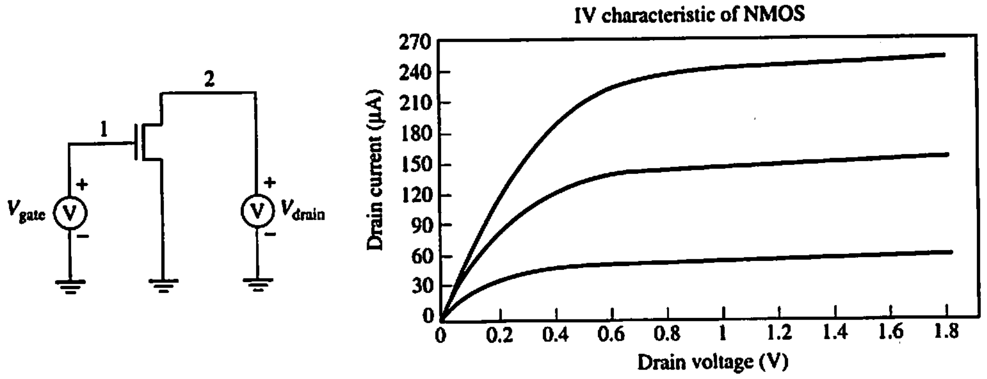
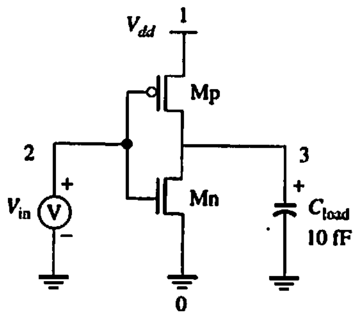


Figure A.12
IV Characteristic of NMOS.

Figure A.13 shows a transient analysis of an inverter by applying a ramp function at the input and viewing the output:

```
Transient Response of an Inverter
.param Supply=1.8            * Set value of Vdd
.lib 'bsim3v3.cmos.18um'  * Set 0.18um library
.opt scale=0.1u              * Set lambda (here lambda=0.10)
Vdd  1    0     'Supply'
Vin  2    0     pulse     0     'Supply'  1n   0n   0n   2n   4n
Mp 3  2  1  1 pmos l=2 w=8 as=40 ps=4 ad=40 pd=8
Mn 3  2  0  0 nmos l=2 w=4 as=20 ps=4 ad=20 pd=4
Cload     3     0     10fF
* Transient analysis
.tran 50ps 5ns
* Plotting requests
.plot tran V(3)
.end
```



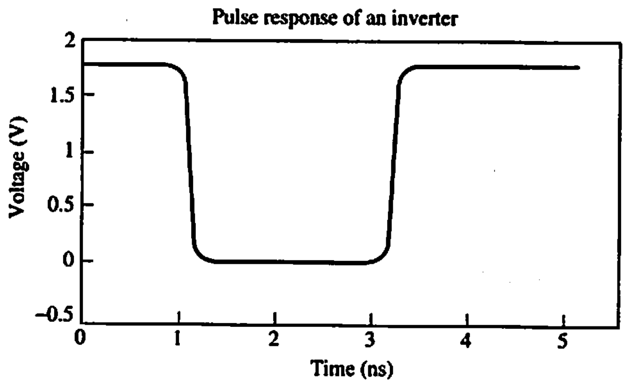


Figure A.13
Pulse response of an inverter.