8 File Formats—Design Verification

This chapter describes the file formats recognized (as input) and produced (as output) by L-Edit and LVS.

The default filename extensions are listed in the tables below.

Input formats:

Format name	***Filename extension***
Element description file	**.elm**
Prematch file	**.pre**
SPICE file	**.sp**, **.spc**

Output formats:

Format name	***Filename extension***
Node and element list	**.lst**
Layout vs. schematic output	**.lvs**

Element Description File Format

The element description file contains a list of statements defining all non-SPICE devices present in the netlists to be compared by LVS. It is only needed when the netlist files being compared contain special devices.

Syntax

Element description statements have the following syntax:

```
|| d name pin [pin ...] [(perm[,perm ...])];
```

ll d	An element description statement always begins with these characters.
name	The name of the element being defined. The element name must match the subcircuit name used in the SPICE file.
pin	A list of the pins for the element. There must be at least one pin.

perm	An optional list of pin permutability statements, enclosed in parentheses. If more than one permutability statement is included, then they are separated with commas. No spaces are allowed within the parentheses.
;	An element description statement always ends with a semicolon.

Permutability Statements

Pin permutability statements are used to indicate pins whose order can be swapped. For example, if you were defining a transistor, it might not be possible to distinguish between the source and drain pins, so you would declare them permutable. The names of the pins used in these statements must *exactly* match the pin names used elsewhere in the statement, including capitalization. Pin permutability statements can have one of two forms:

```
pin==
```

or

```
pin1=pin2[=pin3[=...]]
```

The first example states that all pins with the given name (***pin***) are permutable. The second statement states that all pins in the list (***pin1***, ***pin2***, ***pin3***, …) are permutable with each other. No spaces are allowed in either form.

Element Description Examples

```
|| d C POS NEG (POS=NEG) ;
```

The example above defines a non-polarized capacitor.

```
|| d DEV G1 G2 G3 DS DS (G1=G2=G3,DS==) ;
```

The example above defines a custom element, which will be called in the netlist file through a subcircuit element statement.

Extract Definition File Format

The extract definition file contains a list of comments, connection statements, and device statements. For a complete description of this file format, see Extract Definition File Format on page 3-123.

LVS Output File Format

An LVS file is a text file that contains the command line used to initiate the verification run, verification options, progress statements written during the verification run, and the results of the verification itself.

In the following example, this command line is used to initiate the verification run:

```
Command line:
lvs layout_resonator.spc schematic_resonator.spc -o C:\test\resonator.lvs -l
layout_resonator.lst -e \\Petervpc\lvs v3.0 rel\Examples\resonator.elm -nrcl -y2 -
vfpar
```

Beneath the command line, LVS displays the options chosen in the setup window:

```
Engine configuration report:
Consider Bulk nodes......................................................ON
Consider Resistors as polarized elements.............................OFF
Consider Capacitors as polarized elements............................OFF
Consider Inductors as polarized elements.............................OFF
Optimize shorted & parallel R, C, MOSFETs; series R and C..............OFF
Replace series MOSFETs...................................................OFF
Fast Iteration............................................................OFF
```

Next, the verification window displays the progress statements written during the verification run:

```
Reading element definitions from file \\Petervpc\lvs v3.0 rel\Examples\resonator.el
Parsing file layout_resonator.spc...
Not resolving subckt fspring
Not resolving subckt comb
Not resolving subckt fspring
Not resolving subckt comb
Not resolving subckt plate4
Parsing file schematic_resonator.spc...
```

Verification results can vary widely according to problems encountered during the verification run. For further information on LVS verification output, see the LVS Output Tutorial on page 3-214.

Prematch File Format

A prematch file is a text file that specifies pairs of nodes and elements that are to be considered equal. LVS uses this information to resolve automorph classes.

Syntax

Each line of a prematch file names two nodes or elements that are to be considered equal:

```
member1a member2a
member1b member2b
member1c member3c
. . .
```

member1 ...	The entry on the *left* in each line belongs to the **Schematic netlist** specified in the **Run LVS** dialog—see Setup Window—File on page 3-158.
member2 ...	The entry on the *right* in each line belongs to the **Layout netlist** specified in the **Run LVS** dialog—see Setup Window—File on page 3-158.

An *element* from one file cannot be equated with a *node* from the other file.

Node and Element List Format

The node and element list, an optionally generated text file, names matching and unresolved nodes and elements, by iteration, in the two compared netlists.

Syntax

For each iteration, nodes and elements are listed in the order:

- Resolved nodes
- Other (unresolved) nodes
- Resolved elements
- Other (unresolved) elements

Resolved nodes and elements are listed as follows:

```
member1a <=> member2a
member1b <=> member2b
...
```

member1 …	The entry on the *left* in each line belongs to the **Schematic netlist** specified in the **Run LVS** dialog.
member2 …	The entry on the *right* in each line belongs to the **Layout netlist** specified in the **Run LVS** dialog.

Other (unresolved) nodes and elements are listed in single-column format.

SPICE File Format

By using the correct commands, an entire circuit and all contained devices can be described in SPICE format.

The maximum number of characters per line is 80. Statements extending over multiple lines must have the **+** continuation character in the first column of each line after the first.

The end-of-line character depends on the operating system; these differences must be accounted for when moving SPICE files between operating systems.

Filenames are dependent on the operating system in use; the filename specified in an **.include** command must meet operating system requirements, including maximum length and special character requirements.

Passive Elements

Passive SPICE elements include capacitors, resistors, inductors, and transmission lines.

Capacitors, resistors, and inductor statements have parallel syntax:

```
Cname node1 node2 [model] [C=]c [M=m]
Rname node1 node2 [model] [R=]r [M=m]
```

```
Lname node1 node2 [model] [L=]l [M=m]
```

For Extract, the syntax for an inductor is:

```
Lname node1 node2 [model] [L=]
```

C, **R**, **L**	The SPICE key letter identifying the element type: **C** (capacitor), **R** (resistor), **L** (inductor).
name	A unique element name.
node1 node2	Node names.
model	An optional model name. If included, the specified model must be elsewhere defined with a **.model** command.
C=*c*, **R=*r***, **L=*l***	Electrical parameters: ***c*** (capacitance), ***r*** (resistance), ***l*** (inductance).
M=*m*	Multiplicity: the number of devices in parallel (the default is 1).

The syntax for parasitic capacitance is:

```
Cparxxx Node 0 [C=] CapValue
```

For example, **Cpar1** is a capacitance that represents the parasitic capacitance on node **Out**. It goes from node **Out** to ground (**0**) and has a value of 23.5 femtofarads.

```
Cpar1 Out 0 23.459362f
```

The capacitance value is calculated for each piece of geometry comprising the node as

$$CapValue = Area * Ca + Perimeter * Cf$$

where *Ca* is the capacitance per unit area (area capacitance) and *Cf* is the fringe capacitance. Both of these constants are set in General Layer Parameters on page 1-158.

Transmission line statements have slightly different syntax:

```
Tname n1a n1b n2a n2b Z0=z [TD=d] [F=f] [NL=n]
```

T	The SPICE key letter specifying a transmission line.
name	A unique element name.
n1a n2a n1b n2b	Node names.
Z0=*z*	Impedance.

TD=*d*	Transmission delay. The delay may instead be specified indirectly from ***f*** and ***n***.
F=*f*	Frequency.
NL=*n*	Normalized length (in number of wavelengths). The transmission delay is ***n* / *f***.

Parameters may appear in any order, with one exception: if an optional parameter name (such as the **c=** for a capacitor) is not specified, then that parameter must be the first parameter listed.

Other unlisted parameters may appear, but are not used for netlist comparison; they are ignored.

For example:

```
c120 n1997 set1 c=120pf
```

defines a 120pF capacitor with the unique name **C120**. One pin is connected to node **N1997**, and the other pin is connected to node **set1**.

Active Elements

Active elements (semiconductor devices) include diodes, BJTs, MESFETs, JFETs, and MOSFETs.

Active elements have the following syntax:

```
Dname node1 node2 model [[AREA=]a ] [M=m]
Qname col bas emt [sub] model [[AREA=]a] [M=m]
Zname drn gat src [blk] model [[AREA=]a] [M=m]
Jname drn gat src [blk] model [[AREA=]a] [M=m]
Mname drn gat src [blk] model [L=l] [W=w] [AD=ad]
+ [AS=as] [PD=pd] [PS=ps] [NRD=nrd]
+ [NRS=nrs] [NRG=nrg] [NRB=nrb] [M=m]
```

For Extract, the syntax for active elements varies as follows:

```
Dname node1 node2 model [AREA= {rLayerArea | pinArea } /
    areaVal]
Qname col bas emt [sub] model [AREA= {rLayerArea | pinArea }
    /areaVal]
Jname drn gat src [blk] model [AREA= {rLayerArea | pinArea }
    /areaVal]
Mname drn gat src [blk] model L=lengthValue W=widthValue
+      [AD=areaValue] [PD=perimeterValue] [AS=areaValue]
```

Extract also includes the following syntax for MESFETs:

```
Zname drn gat src [blk] model [AREA= {rLayerArea | pinArea }
    /areaVal]
```

```
Zname drn gat src [blk] model L=length W=width
```

D, **Q**, **Z**, **J**, **M**	SPICE key letter identifying the element type: ▪ **D** (diode) ▪ **Q** (BJT) ▪ **Z** (GaAsFET/MESFET) ▪ **J** (JFET) ▪ **M** (MOSFET)
name	A unique device name.
col, ***bas***, ***emt***, ***sub***, ***drn***, ***gat***, ***src***, ***blk***	Node names corresponding to terminals: ***col*** (collector), ***bas*** (base), ***emt*** (emitter), ***sub*** (substrate), ***drn*** (drain), ***gat*** (gate), ***src*** (source), ***blk*** (bulk).
model	The model name of the device. The model name is required for semiconductor devices, and must be defined with a **.model** command.
AREA=*a*	Device area.
M=*m*	Multiplicity.
L=*l*, **W=*w***	Length and width.

AD=*ad*, AS=*as*	Drain and source areas.
PD=*pd*, PS=*ps*	Drain and source perimeters.
NRD=*nrd*, NRS=*nrs*, NRG=*nrg*, NRB=*nrb*	Number of resistance squares for drain, source, gate, and bulk.

Parameters may appear in any order. For example:

```
M12 17 19 21 21 PMOS L=2U W=28U
```

defines a PMOS transistor with the unique name of **M12**. The drain node is **17**, the gate node is **19**, and the source node and bulk nodes are both **21**. The transistor has a length of 2 microns and a width of 28 microns.

A question mark (**?**) indicates that a pin specified in the EXT file was not found in the layout. For example:

```
M1 M1_Drain G1 ? 1 PMOS L=0.8u W=0.5u AS=1p PS=4u
* M1 DRAIN GATE SOURCE BULK (23.5 -15 28.5 -13)
```

defines the PMOS transistor **M1**. The drain and gate pins are identified by their ports (**M1_Drain** and **G1**, respectively), but the source pin is shown only as **?**, indicating it was not found on the layout. This output can result from an error in the device definition or in the layout. Examine your design file and extract definition file to find the cause of the error.

Subcircuits

A *subcircuit* consists of an arrangement of elements treated as a unit. Subcircuits can be instantiated repeatedly at different places in a circuit.

Subcircuit Instances

Subcircuit instance statements have the following syntax:

T-Spice / H-SPICE:

```
Xiname node1 [node2 ...] cname [par1=p1 par2=p2 ...]
```

P-SPICE:

```
Xiname node1 [node2 ...] cname
PARAMS:[par1=p1 par2=p2 ...]
```

For Extract, the syntax varies as follows:

```
Xiname node1 [node2 ...] cname [AREA=rLayerArea/areaVal]
[AREA_pinName=pin1Area/areaVal]
[AREA_pinName=pin2Area/areaVal] ...
```

iname	A unique instance name.

node1 node2	Node names. There must be as many node names listed as there are in the subcircuit definition.
par1 par2	The list of available parameters is determined by the subcircuit definition. Parameter assignments are optional on a subcircuit instance statement; parameters may be listed in any order. If a parameter's value is not specified in the instance statement, its value is taken from the default assigned in the subcircuit definition statement. The multiplicity parameter (**M=*m***) is implicitly defined and can also be listed.
cname	The subcircuit name (from the definition statement).

For example (in T-Spice or H-SPICE):

```
X123 N125 N253 N74 myCircuit AREA=100 Q=42 E=17
```

or (in P-SPICE):

```
X123 N125 N253 N74 myCircuit PARAMS: AREA=100 Q=42 E=17
```

instantiates a previously defined subcircuit called **myCircuit**. Its unique name, to distinguish it from other instances of the same subcircuit, is **X123**. It has three pins, connected to nodes **N125**, **N253** and **N74**. It also has three parameters: **AREA**, **Q**, and **E**. The definition for this subcircuit is given as an example below.

For Extract

```
X123 N125 N253 N74 myCircuit AREA=100 AREA_Pin1=15
```

defines an instance **X123** of a subcircuit called **myCircuit**. It has three pins, connected to nodes **N125**, **N253**, and **N74**.

Subcircuit Definitions

Subcircuit definition statements have the following syntax:

T-Spice / H-SPICE:

```
.SUBCKT name pin1 [pin2 ...]
+ [par1=val1 par2=val2 ...]
<subcircuit definition>
.ENDS [name]
```

P-SPICE:

```
.SUBCKT name pin1 [pin2 ...]
PARAMS:[par1=val1 par2=val2 ...]
```

```
<subcircuit definition>
.ENDS [name]
```

name	The name, or type, of the circuit.
pin1 pin2	The pins (inputs and outputs) to the circuit.
par1 par2	An optional list defining the parameters whose values must be known when the subcircuit is instantiated. A value given for a parameter here is its default, to be assumed if the parameter is not assigned a value on the subcircuit instance statement. The multiplicity (**M=*m***) parameter may not be included here.

The last line of the subcircuit definition can optionally contain the same subcircuit name used in the first line of the definition (for example, **.ends MYCIRCUIT**).

In between the first (**.subckt**) and last (**.ends**) lines are any number of other SPICE commands and statements (except subcircuit instance and model commands).

If the subcircuit is empty, it must be defined as an element in the special element file to be used with LVS.

SPICE Statements

.INCLUDE

A SPICE file can include the contents of other SPICE files with the **.include** command.

The **.include** command has the following syntax:

```
.INCLUDE 'filename'
```

The ***filename*** can be the name of any other SPICE file, and can include drive and path information, if needed. The filename must be contained within single quotes. Care should be taken to ensure that inclusion commands do not involve logical loops (for example, **fileA** including **fileB**, which itself includes **fileA**).

.MODEL

The **.model** command defines a model to be used in device statements. It can appear anywhere in the SPICE file, even after the specified model is mentioned in an element statement.

The **.model** command has the following syntax:

```
.MODEL name type [par1=p1 par2=p2 ...]
```

name	The name of the model.
type	One of the following: **C** (capacitor); **R** (resistor); **L** (inductor); **D** (diode); **NPN** or **PNP** (BJT); **NJF** or **PJF** (JFET); **NMOS** or **PMOS** (MOSFET).
par1 par2	The parameters for the model are listed after the model type, and are specific to the model type. The set of parameters determines the SPICE behavior of the model. However, for extraction and netlist comparison, the parameters are not used and can be left off.

For example:

```
.model mydevice nmos
```

specifies an NMOS MOSFET device named **mydevice**. The model name can be used in device statements such as the following:

```
m123 42 51 7 mydevice l=2 w=28
```

This defines an NMOS transistor with the unique name of **M123**. Its drain is connected to node **42**, its gate to node **51**, and its source to node **7**. It has a length of 2 and a width of 28.

.GLOBAL

The **.global** command declares certain nodes as global throughout the SPICE file. It is used in SPICE files containing subcircuits in order to make certain signals available to the subcircuit without explicitly having to declare them in the subcircuit definition. Typical global nodes might include clocks, power, the data bus, etc.

The **.global** command has the following syntax:

```
.global node1 node2 ...
```

For example:

```
.global clock data1 data2 data3 data4
```

Without **.global**, the **clock** node would be considered local to the subcircuit in which it is defined and distinct from any other node called **clock** outside of the subcircuit. With **.global** however, every node called **clock** inside or outside of a subcircuit is equivalent.

If the **-pspice** option is specified on the command line, LVS recognizes PSPICE global nodes.

.OPTION

The **.option** command has the following syntax:

```
.option scale=s
```

LVS scales all geometric parameters stated in device statements by the given value ***s***. For example, a scale value of 2 doubles the length and width parameters of all MOSFETs, and squares the area parameter on all diodes, BJTs, etc.

For example:

```
.option scale=2
.option scale=100m
```

.PARAM

The **.param** command defines symbolic parameter values so that they can be used anywhere that a parameter value is called for.

The **.param** command has the following syntax:

```
.param symbol1=n1 symbol2=n2 ...
```

For example:

```
.param cap100=100pf tranlen=2u tranwid=28u
```

specifies parameter values that could then be used in statements such as:

```
c123 12 56 c=cap100
m43 24 54 300 nmos l=tranlen w=tranwid
.subckt mycircuit in out reset c=cap100
```

.END

A SPICE file is terminated with an **.end** command on the last line of the file. Anything following this command is ignored.

The **.end** command has the following syntax:

```
.end
```

Parameters

Parameter values can take many forms. Some examples are:

```
area=10      l=.001      r=3.4e-3     c=cap100
```

In the last example, **cap100** must have previously been defined by a **.param** command. None of the examples specify units, so default units are assumed (ohms for resistance, farads for capacitance, meters for length, square meters for area, and so on).

Numbers can be followed by metric abbreviations indicating order of magnitude. The base units (**s**, **v**, **a**, **f**, **h**) are implicit from the context; any characters following the metric abbreviation are ignored. For example, the following expressions can all specify a capacitance of 10.2 picofarads:

```
C=10.2pF
C=10.2P
C=10.2pxyz
```

Acceptable metric prefix abbreviations are shown in the following table.

Abbreviation	*Prefix*	*Meaning*
t or **T**	tera-	10^{12}
g or **G**	giga-	10^{9}
meg or **MEG**	mega-	10^{6}
k or **K**	kilo-	10^{3}
m or **M**	milli-	10^{-3}
u or **U**	micro-	10^{-6}

Abbreviation	*Prefix*	*Meaning*
n or **N**	nano-	10^{-9}
p or **P**	pico-	10^{-12}
f or **F**	femto-	10^{-15}

A commonly used abbreviation is the unit **mil** (or **MIL**), representing 10^{-3} inch.

Parameters listed more than once on the same statement assume the *first* assigned value. For example:

```
C12 45 123 C=10p C=20U
```

defines a 10 picofarad capacitor, not a 20 microfarad one.

Comments

Comment lines are designated by an asterisk (*****) in the first column. In-line comments are designated by a dollar sign (**$**) or a semi-colon (**;**). All text following any one of these characters up to the end of the line on which it is found is ignored.

The first line of any SPICE file is *always considered a comment*, even without comment delimiters.

Restrictions and Extensions

There are many varieties of SPICE, but only those conforming to the syntax described above are supported for the purposes of netlist comparison and layout extraction.

Unsupported parameters, if present, are ignored. Default values for parameters are not supported, except for the multiplicity parameter and subcircuit parameters. If a parameter is not specified but is needed for a computation, an error will occur and a warning message will be displayed.

Subcircuit definitions may not have model statements or other subcircuit definitions within them.

The **.bulk** command is not supported.

Only certain model types are supported (see **.MODEL** (page 3-281)); others are ignored. Parameters for model statements are not utilized. Any parameters listed in a model statement will be ignored.

For netlist comparison, the SPICE format supported by LVS contains an additional feature to enable comparison of non-standard elements. This is accomplished by defining the additional elements in an *element description file*, and then using the devices in the same manner as you would use a subcircuit element.