

3 Extracting Layout

- [Configuring the Extractor](#) 3-86
- [SPICE OUTPUT Properties](#) 3-94
- [Running the Extractor](#) 3-97
- [Subcircuit Recognition](#) 3-109
- [Extract Definition File Format](#) 3-123

Extraction is a method of verifying a layout. The extraction process produces a *netlist* that describes the circuit represented by the layout in terms of device and connectivity information.

The L-Edit general device extractor:

- Recognizes active devices (BJTs, diodes, GaAsFETs, JFETs, and MOSFETs), passive devices (capacitors, inductors, and resistors), and non-standard or compound devices (by means of *subcircuit recognition*).
- Maintains process independence by means of an *extract definition file*, which describes how layers interact electrically.
- Handles large regions of layout in a memory-efficient manner by *binning*.
- Uses device definitions that can be specified using *generated layers*, for a greatly expanded set of possible definitions. Derived layers are generated and disposed of automatically.
- Works with the most common device parameters, including resistance, capacitance, and device length, width, and area. These parameters provide useful information when verifying drive, fanout, and other circuit performance characteristics.
- Creates a netlist file in Berkeley 2G6 SPICE format, usable with any tool that reads a SPICE netlist. This netlist is ideal for use with the Tanner T-Spice™ circuit simulator (to verify device sizes, drive capabilities, and other circuit performance factors) or the LVS netlist comparator (to check the equivalence of netlists generated from different sources).

Configuring the Extractor

The extraction process is defined by making associations between patterns of layout geometry and the circuit components they represent. These associations are defined in the extract definition file (**.ext**). For additional information on the format of this file, see [Extract Definition File Format on page 3-123](#).

See [Configuration Example on page 3-90](#) for an illustration of the concepts described in this section.

Devices and Connections

The first step is to determine the specific classes of devices and connections that are to be extracted.

- A *device* is any circuit element (transistor, resistor, capacitor, diode, etc.).
- A *connection* is any electrical connectivity between two process layers, such as between the Poly and Metal1 layers when a contact is present on the Poly Contact layer.

Only relevant devices and connections need be defined. For example, every design contains resistors, because no process layer is a perfect conductor. But if the design to be extracted does not contain any wire long enough for its inherent

resistance to affect the circuit's performance, then the wires do not have to be defined and extracted as resistors.

Generated Layers

When you use generated layers in an extract definition, L-Edit automatically generates objects on those layers before proceeding with netlist extraction. Following netlist extraction, L-Edit automatically deletes the objects it created during that Extract run. It only deletes objects generated during that Extract run, however—previously generated objects remain. For further information about generated layers, see [Generating Layers on page 1-435](#).

To extract resistors and capacitors, you must also enter the following three constants for each involved layer. Use [General Layer Parameters \(page 1-158\)](#) to enter these values.

- An *area capacitance* (in attofarads per square micron)
- A *fringe capacitance* (in femtofarads per micron)
- A *resistivity* (in ohms per unit area)

Capacitance is the sum of two products: that of the area of the capacitor and the area capacitance, and that of the perimeter of the capacitor and the fringe capacitance. (Capacitors are polygons on the recognition layer.) *Resistance* is the product of the resistivity and the length of the resistor, divided by the width.

The extractor only operates on boxes and on 45° and 90° polygons and wires; it does not extract circles or all-angle polygons and wires.

Note:

Generation of layers containing 45° objects can produce off-grid vertices due to off-grid intersections of 45° polygons or conversion of 45° wires to polygons. For further information on this topic, see [Working with 45° Objects on page 1-476](#).

Extract Definition File

The *extract definition file* contains a list of the connections and devices to be extracted. This file can be used to define:

- Connections between two different process layers
- Devices in terms of their type, component layers, pins, and model names

The directory **L-Edit Pro\tech\mosis** contains a set of extract definition files that correspond to various technology processes. You can modify these files as necessary to define additional connections and devices for extraction.

For a detailed reference on the syntax of the extract definition file, see [Extract Definition File Format on page 3-123](#).

Node Names

Extract can write out nodes as internally generated numbers (using the option **Integers** in **Extract—Output**) or as descriptive strings (using the option **Names** in **Extract—Output**). For further information on this dialog, see **Extract—Output** (page 3-101).

To label a node or element for extract to a netlist, you must add a port to the layer of that node or element, within an object (box, polygon or wire) on that layer. When **Names** is selected, L-Edit derives node names from the names of ports found on the same layer as the node. It derives port names from ports found on the device-specific recognition layer that are completely enclosed by that device.

If you want to use the same ports for a design rule check that uses an assigned dummy layer, you can change the layers for your ports by:

- ☒ hiding all objects except for ports
- ☒ hiding all layers except the node layer
- ☒ selecting all objects, which will be just the ports on the node layer
- ☒ using **Edit > Edit Object** to change the layer to the layer of your choice.

The strings produced by the extractor are the hierarchical names; each instance involved in a node is mentioned and separated from the others by a slash (/), with

the port name at the end. (Instances that are unnamed in the layout are named automatically by the extractor.) For example, the node name **U1/alpha/in** describes a port **in** contained by an instance **alpha**, which in turn is contained by an instance **U1**.

Configuration Example

The following example illustrates how to configure the extractor to recognize a transistor in a CMOS *n*-well process. It shows how transistors may be clearly and uniquely identified by generated layer definitions and device statements in the extractor definition file. Other SPICE devices may be identified in similar fashion.

An NMOS transistor in an *n*-well CMOS process consists of:

- The channel
- A source pin of *n*-doped diffusion material touching the channel
- A gate pin of polysilicon over the channel
- A drain pin of *n*-doped diffusion material touching the channel
- A bulk pin to the substrate

When the extractor finds a configuration of polygons in the layout corresponding to this definition, it should write an NMOS transistor statement into the output file (netlist).

Device Definition

The following statement causes a MOSFET to be generated in the output.

```
# NMOS transistor
device = MOSFET(
    RLAYER=ntran;
    Drain=ndiff, WIDTH;
    Gate=Poly;
    Source=ndiff, WIDTH;
    Bulk=subs;
    MODEL=NMOS;
)
```

The recognition layer is defined as **ntran**, and the pin layers are defined as **ndiff**, **Poly**, and **subs**. This causes the extractor to recognize a MOSFET wherever it sees **ntran** geometry, touched by geometry on **ndiff**, **Poly**, and **subs**.

However, MOSFETs are not typically created by drawing geometry on **ntran**, **ndiff**, or **subs**. They are created by drawing **Poly** geometry over **Active** geometry inside **NSelect** geometry.

To generate correct geometry on the **ntran**, **ndiff**, and **subs** layers from user-drawn geometry on **Active**, **Poly**, and **N Select** layers, use generated layers (see [Generating Layers on page 1-435](#)).

Recognition Layers

A transistor gate is formed on the chip when **Poly** geometry and **Active** geometry intersect on the layout. The generated layer

```
gate = ( Poly ) AND ( Active )
```

is used to define a generic transistor gate.

However, a CMOS process will have both NMOS transistors (in the substrate) and PMOS transistors. The **gate** layer definition does not differentiate between the two.

L-Edit uses a default CMOS setup that assumes a *p*-substrate, has a nongenerated layer (**N Well**) for defining the *n*-well, but does not have a layer for defining the substrate surface. The generated layer

```
subs = NOT ( N Well )
```

is used to define the substrate surface.

Now, two generated layers can uniquely identify NMOS and PMOS transistor channels:

```
ntran = ( gate ) AND ( subs )
ptran = ( gate ) AND ( N Well )
```

Pin Layers

When the extractor identifies a transistor to be written to the output netlist, it looks for the pins that should be touching the recognition layer if the device is properly constructed.

A MOSFET has four pins attached to it: drain, gate, source, and bulk. The gate is defined to be the **Poly** geometry that touches the transistor. The bulk in a PMOS device is the **N Well**, and in an NMOS device is the substrate (**subs**). For these pins, the proper layers are already defined.

In the layout, a single polygon on the **Active** layer stretches across the whole transistor, but in a fabricated chip, the diffusion material will not exist under the gate. The generated layer

```
Field Active = ( Active ) AND ( NOT ( Poly ) )
```

creates geometry on either side of, but not underneath, a transistor gate.

Finally, an NMOS transistor has source and drain pins made up of *n*-doped material, and a PMOS transistor has source and drain pins made of *p*-doped material. The doping type is controlled by drawing geometry on the **N Select** and **P Select** layers, so two generated layers can uniquely identify both pin layers:

```
ndiff = ( Field Active ) AND ( N Select )
pdiff = ( Field Active ) AND ( P Select )
```

SPICE OUTPUT Properties

SPICE OUTPUT properties are a subset of the general purpose L-Edit properties described in [Properties on page 1-70](#). In subcircuit extraction, **SPICE OUTPUT** properties determine the device name, connectivity, and device parameters of the subcircuit.

You can use properties to format the output of subcircuit information written to a SPICE netlist. You can attach properties to either a parent cell or an instance. L-Edit/Extract searches for instance properties first. If it does not find any, it searches for properties on the parent cell.

SPICE OUTPUT properties are only processed as part of subcircuit extraction. See [Subcircuit Recognition on page 3-109](#) for more information.

Property Tokens

String properties can include tokens, which are references to other values and variables. The following tokens are expanded during subcircuit extraction. All other text is parsed without expansion.

<i>Token</i>	<i>Expansion</i>
#	An incremented integer that counts the instances of the cell. (This token is only expanded during extract.)

<i>Token</i>	<i>Expansion</i>
<code>\${property}</code>	The value of the named <i>property</i> (as a string). <i>property</i> can be expressed using the full path—e.g., Mechanical.Length . If no path is designated, EXTRACT is assumed.
<code>%{port}</code>	The name of the node to which the pin associated with the named <i>port</i> is attached.
<code>\n \t</code>	New line (<code>\n</code>) or tab (<code>\t</code>) characters.
<code>\\ \# \\$ \% \{ \} \[\]</code>	The character after the initial backslash (instead of being interpreted as part of a token).

For example, the value of the following **EXTRACT.SPICE OUTPUT** property:

```
XPlate %{right} %{left} platemodel W=${W} L=${L}
```

would result in the following netlist output:

```
XPlate 5 3 platemodel W=5e-6 L=2e-6
```

where **EXTRACT.W**=5e-6 and **EXTRACT.L**=2e-6.

Application Example

In the following example, the **SPICE OUTPUT** property allows you to specify multiple energy domain connections such as electrical and mechanical connections—for example, the **SPICE OUTPUT** property of the following MEMS plate:

```
X${instance} %{PL_Left}_m %{PL_Right}_m %{PL_Bottom}_m
              %{PL_Top}_m %{PL_Left}_e %{PL_Right}_e %{PL_Bottom}_e
              %{PL_Top}_e mass4_geo W=${W} L=${L}
```

will result in the following netlist output:

```
Xu5      3_m      4_m      6_m      2_m
          3_e      4_e      6_m      2_m
          mass4_geo W=3e-3 L=2e-4
```


where **EXTRACT.W**=3e-3 and **EXTRACT.L**=2e-4.

Notice that all ports are referenced twice in this string using property tokens: one to specify the mechanical connection and the other to specify the electrical connection.


Running the Extractor

Tools > Extract performs netlist extraction from the active cell.

The options in this dialog are categorized under three tabs:

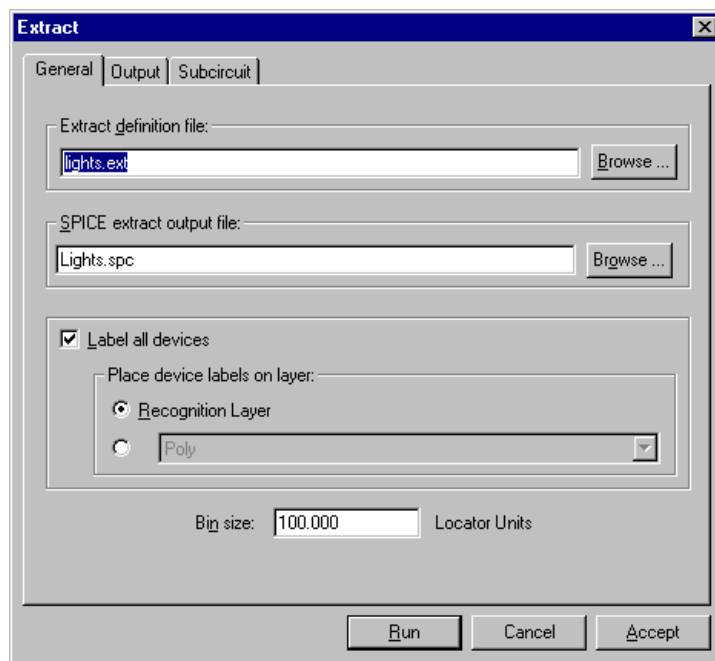


General	Used to specify input and output file names and the bin size.
Output	Used to specify the way in which the extracted circuit is written to the output netlist.
Subcircuit	Used to specify parameters for subcircuit extraction .



Two buttons are general to all tabs.

Run	Begins the extraction process on the active cell. Any instances are temporarily “flattened” (except for those marked as subcircuits).
Accept	Saves the current settings without running Extract .

Extract-General

Extract definition file

Name of the input file containing the extractor device and interconnection definitions. You can choose from available files and directories with the **Browse** button.

SPICE extract output file

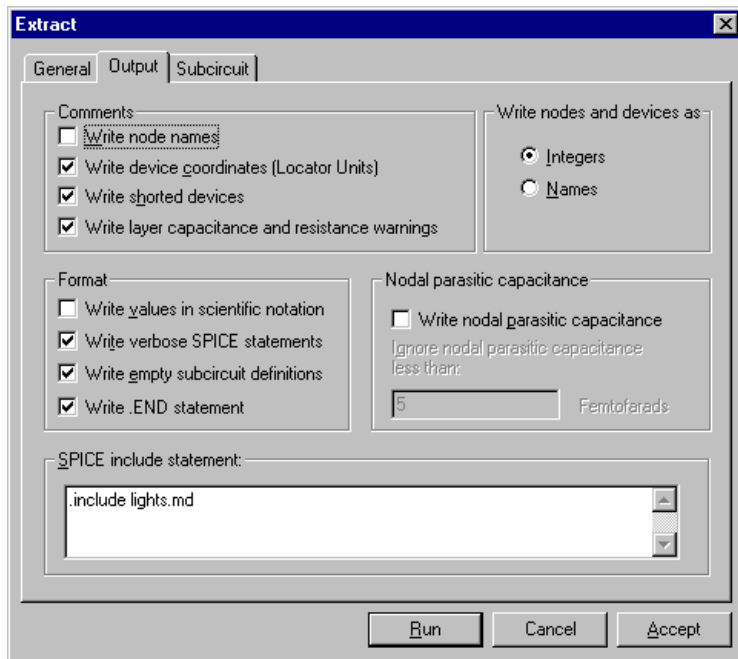
Name of the output file containing the extracted netlist. Enter the name (or use the default). You can choose from available files and directories with the **Browse** button.

Bin size

Length of one side of a bin, in locator units. To improve performance, L-Edit divides the layout into a grid of square bins and extracts each bin individually. Devices that cross bin boundaries are extracted properly. Binning is not used if the **Recognize subcircuit instances** box in the **Subcircuit** tab is checked.

Label all devices

For each unnamed device, creates a two-dimensional port at the location of the device. The text of the port is the text of the element name for the device. Device labels will not be generated for devices with user-placed labels. The group **Place device labels on layer** contains options for writing the device labels on the device-specific **Recognition Layer** or a user-specified layer.

Extract-Output

Write node names

Includes node names with each device statement in the netlist. Node names are written as comment lines in the section **NODE NAME ALIASES**. See

**Write device coordinates
(Locator Units)**

Writes a comment line following each device statement in the netlist. The comment includes the device name, pin names, and coordinates of the lower left and upper right corners of the device.

Write shorted devices

If IGNORE_SHORTS is set in the extract definition file, writes shorted devices into the netlist as comments; otherwise, shorted devices are ignored. If IGNORE_SHORTS is not set, a shorted device is written to the SPICE file as a regular device.

**Write layer capacitance
and resistance warnings**

Writes warnings on layer capacitance and resistance errors to the specified SPICE file.

Write nodes and devices as

Controls whether nodes are written as internally generated numbers (**Integers**) or as descriptive strings (**Names**). Ports in the layout can be used as node or element names in the netlist. For further information, see [Node Names on page 3-89](#).

Write values in scientific notation

When checked, this option writes numerical values in scientific notation instead of engineering units.

Write verbose SPICE statements

Writes resistors, inductors, and capacitors to the netlist file with the device value preceeded with a **R=**, **L=**, or **C=**. For example, a capacitor would have the following format: **Cxxx n1 n2 modelName C=cValue**.

Write empty subcircuit definition

Writes an empty subcircuit definition block at the top of the netlist file. Use only with **Recognize subcircuit instances** on **Extract-Subcircuit** (page 3-105).

Write .END statement

Writes a **.END statement** at the end of the netlist

Write nodal parasitic capacitance

Computes the capacitance with respect to the substrate of each node in the circuit using the area and fringe capacitance constants specified with **Layer Setup** (page 1-155). The node to substrate capacitance of **N** is written to the netlist as a capacitor between **N** and the substrate/ground (**0**). The form of this notation is Cpar1, Cpar2, etc. For further information, see

Ignore nodal parasitic capacitance less than

Write nodal parasitic capacitance is not generally turned on when a netlist is extracted for LVS since the other netlist (typically derived from a schematic) will not contain parasitic capacitors associated with nodes.

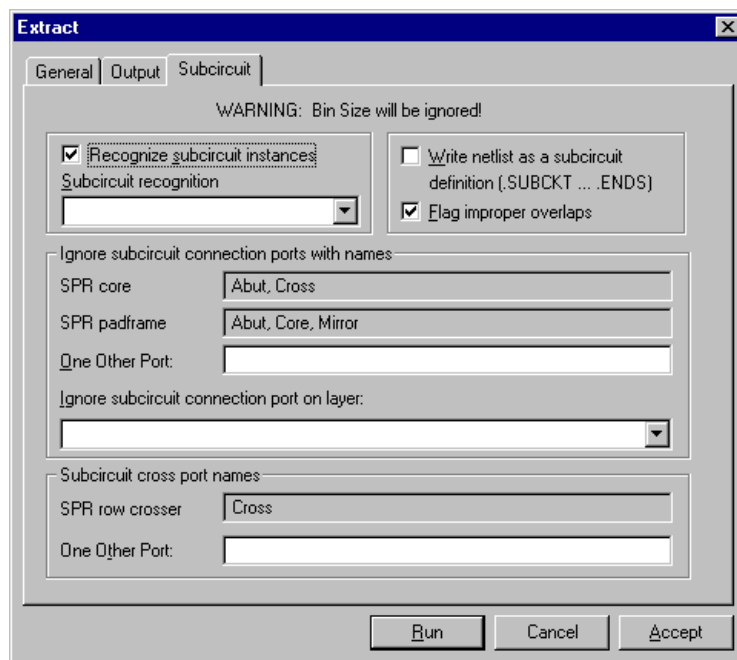
Specifies a limit, in femtofarads, below which the nodal parasitic capacitance will not be written to the netlist. This field is disabled when the **Write nodal parasitic capacitance** box is unchecked.

SPICE include statement

Specifies text that is written unaltered as the second line of the output netlist. Typically, an **.include file** command is entered, where **file** represents a model or subcircuit file name.

Note:

L-Edit cannot determine if other nodes in the circuit are ground nodes. If other nodes are to represent ground, then they must be renamed **0**—or any of its equivalents—in the netlist.

Extract-Subcircuit

Recognize subcircuit instances

Activates the subcircuit recognition feature.

Subcircuit recognition layer

Name of the subcircuit recognition layer (SRL). This mandatory layer should not contain electrically significant geometry.

Write netlist as a subcircuit definition (.SUBCKTENDS)

When checked, this option writes the entire netlist in subcircuit format. A **.subckt** command appears before the first device, and an **.ends** command appears after the last device. When this option is used, there must be a subcircuit recognition polygon at the top level defining the subcircuit in order for Extract to proceed with subcircuit extraction.

Flag improper overlaps

Controls the reaction to geometry violations: under- or over-filled connection ports or geometry that overlaps the subcircuit boundary. Check to display warnings; clear to suppress warnings. Suppressing warnings can be useful when extracting autorouted standard cell designs with known over- and under-fill characteristics.

Ignore subcircuit connection ports with names

List of ports whose names are ignored in subcircuit extraction. Contains the following fields:

- **SPR core**—read-only field listing port names predefined in **SPR Core Setup—General**.
- **SPR padframe**—read-only field listing port names predefined in **SPR Padframe Setup—General**.
- **One Other Port**—use this field to specify one additional subcircuit connection port to be ignored.
- **Ignore subcircuit connection port on layer**—name of a layer on which intruding geometry and subcircuit connection ports will not be recognized. The netlist extractor also ignores any geometry on the Icon layer (often used for documentation purposes).

For more information see [Designing Subcircuit Cells on page 3-111](#).

Subcircuit cross port names

Lists ports whose names are ignored in subcircuit extraction. Contains the following fields:

- **SPR row crosser**—read-only field listing subcircuit cross ports predefined in **SPR Core Setup—General**.
- **One Other Port**—use this field to specify one additional subcircuit cross port.

For more information see [Crossing Over a Subcircuit Instance on page 3-120](#).

Subcircuit Recognition

Most physical layout designs are *hierarchical*. Hierarchical designs help manage complexity, encourage the creation and reuse of library cells, and facilitate computer-aided engineering.

Extract provides a form of hierarchical extraction to automate working with hierarchical designs and to speed up the extraction process in higher-level cells.

This is done by marking often-instanced lower-level cells as *subcircuit cells*, essentially making them “black boxes,” so that every instance will not be extracted explicitly.

When *not* set to recognize subcircuits, Extract “flattens” instances. The extracted netlist describes all devices at the same level, with no indication of hierarchy.

However, if subcircuit recognition *is* activated and there are instances of subcircuit cells, then the extracted netlist contains:

- *An empty subcircuit definition block corresponding to each subcircuit cell.* Each such block begins with the **.subckt** command and ends with the **.ends** command. Subcircuit and node names in the netlist are taken from the names of the subcircuit cells and their connection ports.
- *A SPICE subcircuit instance statement corresponding to each instance.* Each such statement has the form **xinstance pin1 ... subcircuit**, where **instance**

represents the instance name, *pin1 ...* the pin list, and *subcircuit* the subcircuit definition name. (If the instance is unnamed in the layout, **Extract** automatically assigns its name in the netlist.)

Subcircuit recognition is recursive within non-subcircuit instances. If a higher-level cell contains a non-subcircuit instance, and the instanced cell itself contains marked (subcircuit) instances, then the subcircuit instances are properly extracted as subcircuits at any level of hierarchy, and any non-subcircuit instances are flattened.

Activating Subcircuit Recognition

Subcircuit recognition is activated by checking the **Recognize subcircuit instances** option in the dialog **Extract-Subcircuit** (page 3-105).

If the **Write netlist as a subcircuit definition** option is checked, then the entire netlist is written in subcircuit format:

- A **.subckt** command appears before the first device statement, and an **.ends** command appears after the last device statement.
- Subcircuit connection ports at the *top level* (that is, not contained in instances) of the extracted cell are written as SPICE subcircuit pins in the output.

This feature can provide complete subcircuit definitions corresponding to subcircuit instance statements generated from other cells. It requires that the subcircuit recognition polygon and the proper pin ports exist at the top level.

As the extractor runs with subcircuit recognition activated, any errors are reported, and ports placed on the Error layer at their locations in the layout.

Designing Subcircuit Cells

Subcircuit Recognition Polygons

A cell is marked as a *subcircuit cell* by the presence of a *subcircuit recognition polygon* (SRP) on the *subcircuit recognition layer* (SRL).

The SRP is a box or a 90° polygon. It delimits the area of any of the subcircuit cell's instances that cannot be overlapped by geometry in the containing cell and the perimeter at which subcircuit connection ports may be placed.

There are two exceptions to the rule against overlapping an instanced cell's SRP:

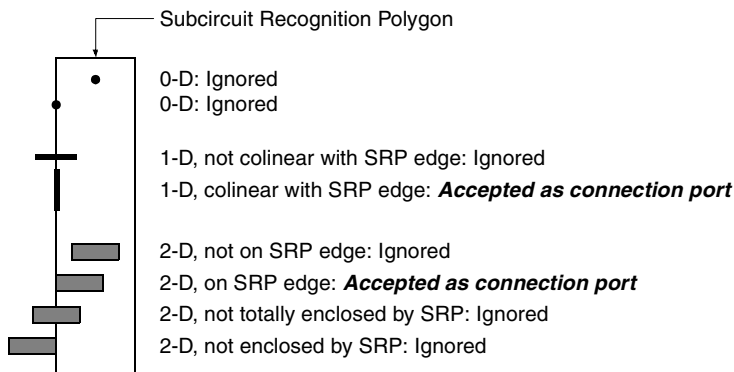
- Geometry inside subcircuit connection ports.
- Geometry over cross port channels.

There may be only one SRP per subcircuit cell. If no SRP exists in the cell, then its instances are not recognized as subcircuits; the extractor flattens them.

Any geometry in a cell that contains an SRP, including geometry outside the SRP, appears in instances of the cell but is *ignored* by the extractor.

Subcircuit Connection Ports

The pins of a subcircuit instance are formed by placing *subcircuit connection ports* inside the subcircuit cell on the particular layer on which connections will be made to the instance. A connection port must both (1) be completely contained by the SRP, and (2) share an edge with the SRP. The port may be 2-dimensional or 1-dimensional (as long as it is colinear with an SRP edge), but not 0-dimensional (a point).



The text associated with a connection port is transferred to the output netlist as the name of a signal parameter (node) on the subcircuit definition. All connection ports, on all layers, with the same name (within one subcircuit cell) are extracted as the *same* subcircuit pin. The pins of a subcircuit are written in alphabetical, then numerical, order.

Certain named ports can be ignored as candidates for connection ports. These are shown in the dialog **Extract-Subcircuit** (page 3-105), in the **Ignore subcircuit connection ports with names** section:

<i>Ignored ports</i>	<i>How specified</i>
SPR core ports	SPR Core Setup-General (page 2-45)
SPR padframe ports	SPR Padframe Setup-General (page 2-69)
Ports on the Icon layer	Special Layers (page 1-182)
Ports matching a single additional name	Other text field
Ports on a single additional layer	Ignore subcircuit connection port on layer drop-down list

Connecting to a Subcircuit Instance

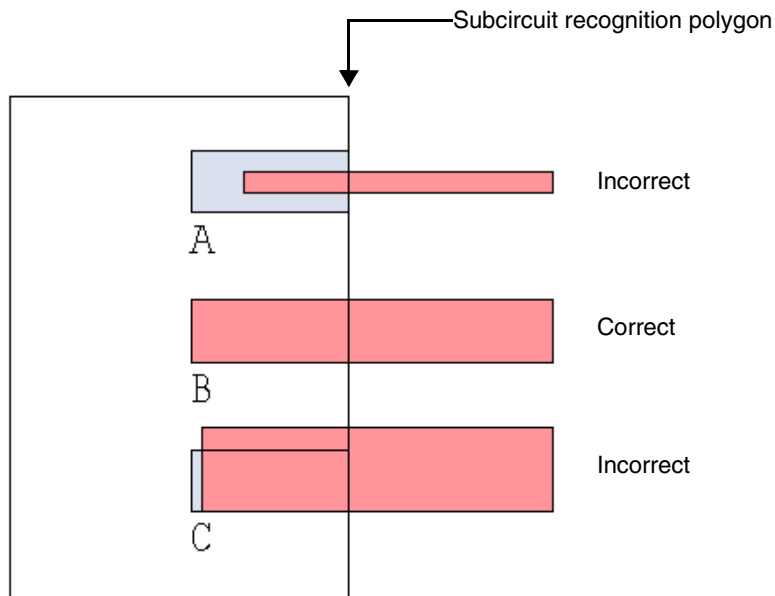
A connection to a subcircuit instance is formed by drawing an orthogonal wire, box, or polygon into a connection port, on the same layer.

- Into a *1-dimensional* port, connecting geometry should be *exactly* as wide as the port and must *exactly* abut the port without overshooting it.
- Into a *2-dimensional* port, connecting geometry should *exactly* fill the port, with neither gaps nor spillovers.

Odd-width wires with extend or round end styles should not be used.

If the connecting geometry touches the connection port but does not exactly satisfy the above criteria, then a connection is still specified in the output netlist, but a warning is generated.

The SRPs of multiple subcircuit instances may be abutted together; connections are formed between abutting 1-dimensional connection ports without additional geometry.

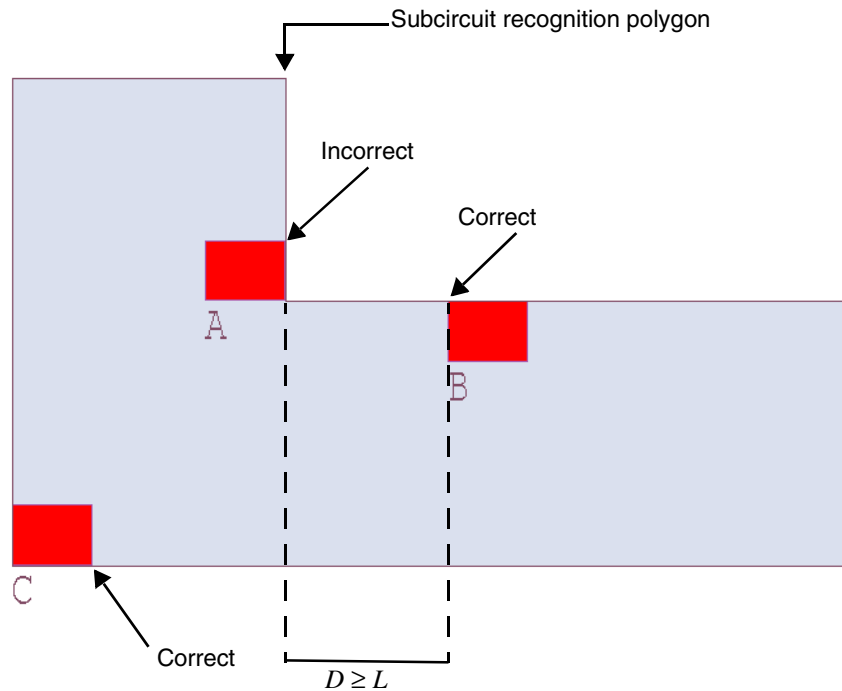


Connecting geometry should approach the SRP orthogonally for a distance at least equal to the largest DRC spacing rule specified for the connecting layer.

Non-connecting geometry should not be placed any closer to a subcircuit instance than this distance.

A connection port should not exist on an “inside” corner of an SRP, but should be separated from the corner by a distance D at least equal to the largest spacing rule value L specified for the layer. See the following illustration.





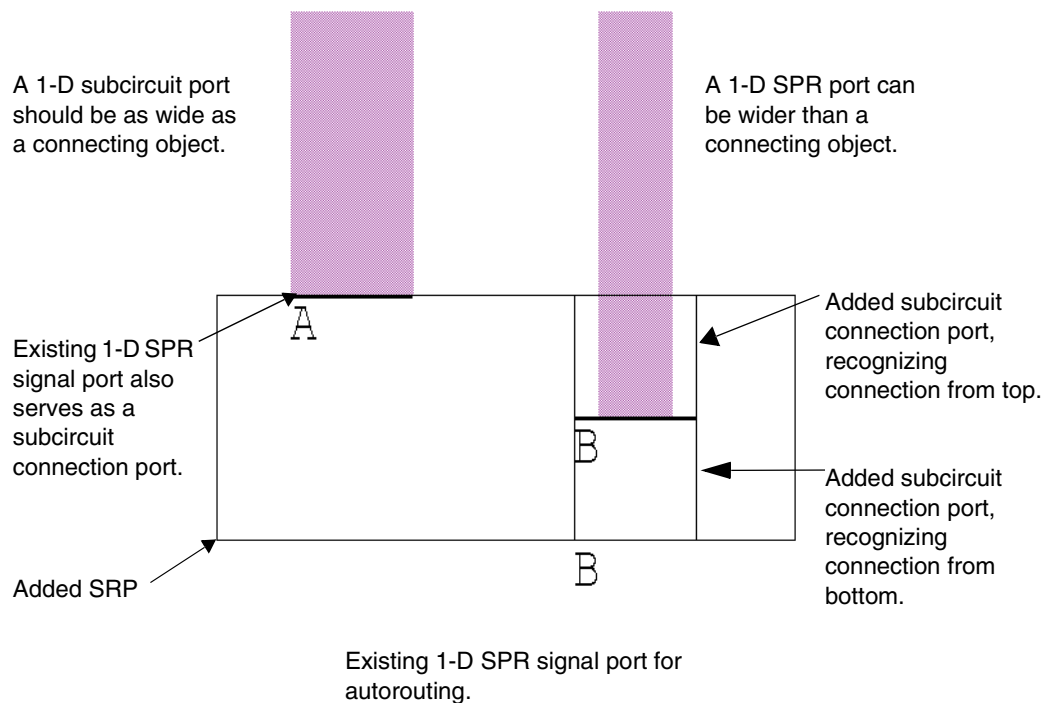
Note:

Extract does *not* check spacing rules.

Subcircuit connection ports and SPR signal connection ports have very similar functions: they mark the locations of connections from outside to inside instances. There are, however, some important differences.

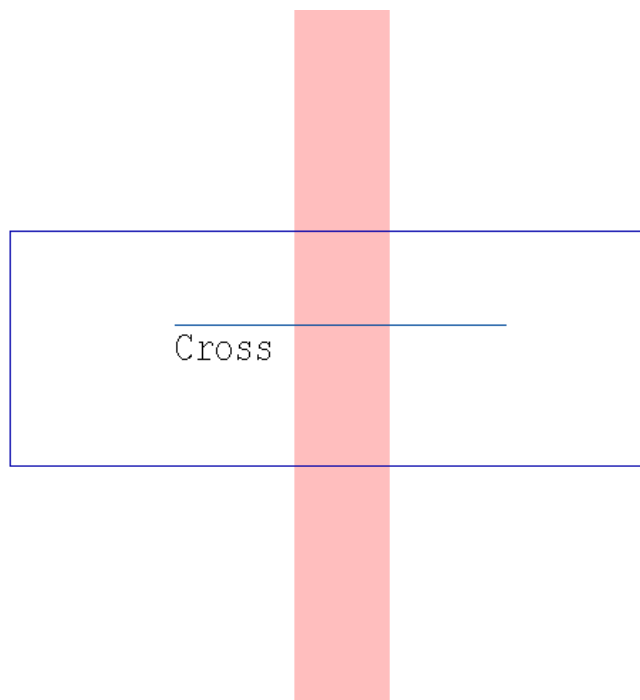
- Subcircuit connection ports may be 1- or 2-dimensional. SPR signal ports must be 1-dimensional.
- 2-dimensional subcircuit ports must be completely filled by connecting geometry, and 1-dimensional subcircuit ports require the connecting geometry to have the same width. SPR signal ports can be of a different width than the connecting geometry.
- SPR signal ports may be entirely within the interior of a cell. Subcircuit connection ports must share an edge with the subcircuit recognition polygon.

Because of these factors, the port construction shown in the following figure is used in standard cell design for use with both SPR and Extract. Moreover, running Extract with the **Flag improper overlaps** option turned off eliminates subcircuit extract warnings. For further information, see [Extract–Subcircuit \(page 3-105\)](#).



Crossing Over a Subcircuit Instance

A 1-dimensional *cross port* in a subcircuit cell defines a “channel” in the cell’s instances, over which geometry may run without causing an overlap warning. The channel runs perpendicular to the width of the cross port and extends from one end of the subcircuit recognition polygon to the other.



Extract recognizes the following as cross ports:

<i>Cross ports</i>	<i>How specified</i>
Row crosser ports	SPR Core Setup–General (page 2-45)
Ports matching a single additional name	In Extract–Subcircuit (page 3-105), in the Subcircuit cross port names section, Other text field

Extract Definition File Format

The extract definition file contains a list of comments, connection statements, and device statements. The directory **L-Edit Pro\tech\mosis** contains a set of extract definition files that correspond to various technology processes. You can modify these files as necessary to define additional connections and devices for extraction.

Extract definition files must conform to the following restrictions:

- Layer names are case-sensitive, and must match the case of layer names defined in the TDB file. The rest of the definition file is case-insensitive; upper and lower cases can be used interchangeably.
- Layer names cannot contain commas or semicolons and they cannot be longer than 40 characters.
- Layer names cannot have leading or trailing spaces.
- Pin names cannot contain commas, semicolons, or spaces, and they cannot be named **MODEL**.
- Model names cannot contain commas, semicolons, spaces, or closing parentheses.
- For compatibility with existing extract definition files, the **WIDTH** keyword is ignored for all devices except a GAASFET/MESFET.

- **IGNORE_SHORTS** indicates that if the device has all of its pins connected to the same node then it will be considered shorted and the device will be written to the extract netlist file as a comment.

Comment Statements

A comment statement begins with a pound sign (#) and continues to the end of the line:

```
# This is an extract definition file comment.
```

Connection Statements

A connection statement defines a connection between two different process layers. A connection always involves three layers: the two layers being connected and the “via” or “contact” layer which connects them. Connection statements have the following format:

```
connect(Layer1, Layer2, ThroughLayer)
```

where ***Layer1*** and ***Layer2*** are the names of the layers being connected, and ***ThroughLayer*** is the name of the connecting layer. For example:

```
# Connect Poly to Metall
CONNECT(Poly, Metall, PolyContact)
```

Substrate Node Statement

Nodal parasitic capacitances are referenced from the node to the substrate. You can designate the substrate node by indicating the substrate layer. Extract will then find the first node connected to the substrate layer and use that as the substrate node when writing the parasitic nodal capacitors. If no substrate layer is indicated, Extract will use node **0** or ground as the substrate node. The substrate node statement has the following format:

```
SUBSTRATE_NODE = SubstrateLayer;
```

where ***SubstrateLayer*** is the name of the substrate layer. For example:

```
# Use Subs as the Substrate node.  
SUBSTRATE_NODE = subs;
```

results in the following SPICE output:

```
Cpar1 5 2 10f
```

where **2** is the substrate node.

Device Statements—General Format

A device statement defines a device. Passive (capacitors, resistors, and inductors,) active (BJTs, diodes, GaAsFETs, JFETs, MOSFETs), and subcircuit devices are specified with the same general format.

For all device statements, it is necessary to identify a *recognition layer*, which is the layer Extract uses to recognize the device. You may specify multiple devices with the same recognition layer as long as they have different pin configurations. This technique is particularly useful in extracting multisource/drain transistors. The recognition layer is defined as follows:

```
RLAYER = rLayer ;
```

where **RLAYER** = is required, and *rLayer* is the name of the recognition layer.

Following the recognition layer is a list of pins of the device. The order of this list determines the order of the pins in the extracted netlist. The extractor does not require any particular order, but LVS requires that both source netlists contain pins in the same order, and SPICE simulators also have strict rules about the order in which pins appear. We recommend following the standard SPICE orders:

- BJT devices: collector—base—emitter—substrate
- MOSFET, JFET, and GaAsFet/MESFET devices: drain—gate—source—bulk

- Diodes, resistors, capacitors, and inductor devices: positiveNode—NegativeNode

If the pin names used in the EXT file are **Collector**, **Base**, **Emitter**, and **Substrate** (BJT devices), or **Drain**, **Gate**, **Source**, and **Bulk** (all other active devices), they are sorted automatically in the default SPICE order.

Pins are specified as follows:

```
pinName = pinLayer ;
```

where ***pinName*** is the name of the pin and ***pinLayer*** is the name of the associated layer.

A model definition follows the list of pins. This definition is not required for passive devices, where **MODEL =;** is acceptable. The model name, if present, will be written into the extracted netlist. For SPICE, model names are not required for capacitors, resistors, inductors, or diodes, but are required for all other devices.

For passive devices, model statements have the form:

```
MODEL = [model];
```

For active devices, model statements have the form:

```
MODEL = model ;
```

where **MODEL =** is required and *model* is the optional model name. The empty statement **MODEL =;** is still required if no model name is specified.

Device Statements—Specific Formats

In the following format specifications:

- Unitalicized words and characters (except the bracket characters [and]) are to be entered as shown.
- Words and characters enclosed by brackets ([]) are optional.
- Words and characters enclosed by braces and separated by a vertical pipe—{ *option1* | *option2* }—represent alternates. You can use the syntax on the left or the right side of the pipe character.
- Variables containing the string *Layer* represent layer names.
- *model* represents the SPICE model name for the device.

Capacitor

```

DEVICE= CAP (
    RLAYER = rLayer [, {AREA} | {LW}];
    Plus = Layer1 [, AREA];
    Minus = Layer2 [, AREA];
    MODEL = [modelName ];
) [IGNORE_SHORTS]
  
```

A capacitor has the following format in the SPICE output statement:

AREA *keyword*

Cxxx n1 n2 [ModelName] [**C=**]cValue

LW *keyword*

Cxxx n1 n2 [ModelName] **L**=cLength **W**=cWidth

The following rules apply to capacitors:

- The optional AREA keyword for a capacitor may be specified on only one layer (recognition or pin layer) and is used to indicate the layer for which the capacitance will be calculated.
- If no AREA keyword or LW keyword is present, the capacitance will be based on the area of the recognition layer (rLayer).
- The LW keyword cannot be used with the AREA keyword.
- The LW keyword can only be used with the recognition layer (rLayer).
- Capacitance is calculated as follows:

$$C_{\text{total}} = C_{\text{area}} + C_{\text{fringe}}$$

$$C_{\text{area}} = (\text{Area of the Layer}) * (\text{Layer's Area Capacitance})$$

$$C_{\text{fringe}} = (\text{Perimeter of the Layer}) * (\text{Layer's Fringe Capacitance})$$

- The area capacitance (aF/sq. micron) and fringe capacitance (fF/micron) are specified in the **Setup Layers** dialog for each specific layer (see [Layer Setup on page 1-155](#)).
- Capacitor average length and width are calculated as if the capacitor was a rectangle. They are calculated as follows:

$L_{\text{perimeter}} = \text{Perimeter of the Layer}$

$L_{\text{area}} = \text{Area of the Layer}$

$$C_{\text{length}} = \frac{1}{4} \bullet L_{\text{perimeter}} + \frac{1}{4} \sqrt{L_{\text{perimeter}}^2 - 16 \bullet L_{\text{area}}}$$

$$C_{\text{width}} = \frac{1}{4} \bullet L_{\text{perimeter}} - \frac{1}{4} \sqrt{L_{\text{perimeter}}^2 - 16 \bullet L_{\text{area}}}$$

Resistor

```

DEVICE=RES (
    RLAYER = rLayer [, LW];
    Plus = Layer1 ;
    Minus = Layer2 ;
    MODEL = [ModelName ];
) [IGNORE_SHORTS]
  
```

A resistor has the following format in the SPICE output statement:

AREA *keyword*

Rxxx n1 n2 [ModelName] [**R=**]rValue

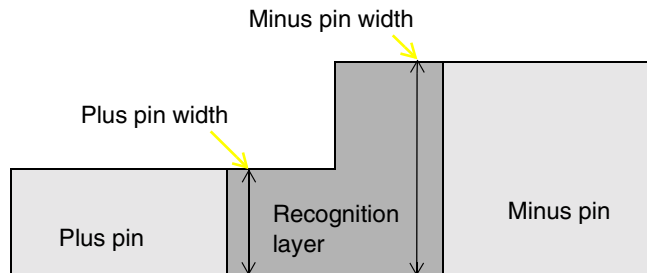
LW *keyword*

Rxxx n1 n2 [ModelName] **L=**rLength **W=**rWidth

The following rules apply to resistors:

- Resistance is calculated with the formula $R = \rho \times (l/w)$, where ρ is the sheet resistance in units of ohms/square, l is the length of the resistor, and w is the width of the resistor.
- The value of ρ is taken from the number specified with the **Setup Layers** dialog for the recognition layer (rLayer) of the resistor (see [Layer Setup on page 1-155](#)).
- The LW keyword can only be used with the recognition layer (**rLayer**).
- The values of l and w are determined from the layout. The extractor computes the area of the recognition layer and divides it by the effective width to obtain l . The width of a pin is the length of the edge that the pin

shares with the recognition layer (***rLayer***). The effective width is the average of the plus pin width and minus pin width.



Inductor

```

DEVICE=IND (
    RLAYER = rLayer ;
    Plus = Layer1 ;
    Minus = Layer2 ;
    MODEL = [model] ;
) [IGNORE_SHORTS]

```

An inductor has the following format in the SPICE output statement:

```
Lname n1 n2 model [L=]
```

Extract does not calculate inductance; users must add this value after netlist extraction.

BJT

```
DEVICE=BJT (
    RLAYER = rLayer [,AREA];
    Collector = cLayer [,AREA];
    Base = bLayer [,AREA];
    Emitter = eLayer [,AREA];
    [Substrate = [sLayer ]];
    MODEL = model ;
    [NominalArea = areaVal ;]
) [IGNORE_SHORTS]
```

A BJT device has the following format in the SPICE output statement:

```
Qname col bas emt [sub] model [AREA= {rLayerArea | pinArea }
    /areaVal]
```

The following rules apply to BJT devices:

- The optional **AREA** keyword can be specified on only one layer (the recognition layer or the pin layer). It is used to indicate the layer for which the area is to be calculated.

- Nominal area can be expressed either in decimal or scientific notation. It has units of m^2 , but no unit tag will appear after the value.
- The **NominalArea** keyword is required if the **AREA** keyword is used.
- If no **AREA** keyword is present, the area will not be written to the SPICE statement.

Diode

```

DEVICE=DIODE (
    RLAYER = rLayer [, AREA];
    Plus = Layer1 [, AREA];
    Minus = Layer2 [, AREA];
    MODEL = model ;
    [NominalArea = areaVal ;]
) [IGNORE_SHORTS]

```

A diode has the following format in the SPICE output statement:

```
Dname n1 n2 model [AREA= {rLayerArea | pinArea } /areaVal]
```

The following rules apply to diodes:

- The optional **AREA** keyword can be specified on only one layer (the recognition layer or the pin layer). It is used to indicate the layer for which the area is to be calculated.

- Nominal area can be expressed either in decimal or scientific notation. It has units of m^2 , but no unit tag will appear after the value.
- The **NominalArea** keyword is required if the **AREA** keyword is used.
- If no **AREA** keyword is present, the area will not be written to the SPICE statement.

GAASFET/MESFET 1

The following syntax can be used to define a GAASFET/MESFET device using its nominal area as a parameter.

```

DEVICE=GAASFET (
    RLAYER = rLayer [, AREA];
    Drain = dLayer [, AREA];
    Gate = gLayer [, AREA];
    Source = sLayer [, AREA];
    [Bulk = [bLayer];]
    MODEL = model;
    [NominalArea = areaVal;]
) [IGNORE_SHORTS]
  
```

The GAASFET/MESFET device has the following format in the SPICE output statement:

```

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

The following rules apply to GAASFET/MESFET devices:

- The optional **AREA** keyword can be specified on only one layer (the recognition layer or the pin layer). It is used to indicate the layer for which the area is to be calculated.
- Nominal area can be expressed either in decimal or scientific notation. It has units of m^2 , but no unit tag will appear after the value.
- The **NominalArea** keyword is required if the **AREA** keyword is used.
- If no **AREA** keyword is present, the area will not be written to the SPICE statement.
- The GAASFET/MESFET 1 syntax is distinguished by the presence of the **AREA**, and/or **NominalArea** keywords. L-Edit determines the appropriate output based on the presence of either of these keywords.

GAASFET/MESFET 2

The following syntax can be used to define a GAASFET/MESFET device using its width as a parameter.

```

DEVICE=GAASFET (
    RLAYER = rLayer ;
    Drain = dLayer [, WIDTH];
    Gate = gLayer ;
    Source = sLayer [, WIDTH];
    [Bulk = [bLayer ] ;]

```

```
MODEL = model ;
) [IGNORE_SHORTS]
```

A GAASFET/MESFET device has the following format in the SPICE output statement:

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

The following rules apply to GAASFET/MESFET devices:

- The length is the length of gate, and the width is the length of the edge that the indicated layer shares with the recognition layer (***rLayer***). The length and width have units of meters.
- The optional **WIDTH** keyword for a GAASFET/MESFET may be specified on only the drain or source pin but not both, and is used to indicate the layer for which width will be calculated.
- If no **WIDTH** keyword is present, the width and length will not be written to the SPICE statement.
- The GAASFET/MESFET 1 syntax is distinguished by the presence of the **WIDTH** keyword. L-Edit determines the appropriate output based on the presence of this keyword.

JFET

```
DEVICE=JFET (
```

```

        RLAYER = rLayer [, AREA];
        Drain = dLayer [, AREA];
        Gate = gLayer [, AREA];
        Source = sLayer [, AREA];
        [Bulk = [bLayer ];]
        MODEL = model ;
        [NominalArea = areaVal ;]
    ) [IGNORE_SHORTS]

```

A JFET device has the following format in the SPICE output statement:

```

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

```

The following rules apply to JFET devices:

- The optional **AREA** keyword can be specified on only one layer (the recognition layer or the pin layer). It is used to indicate the layer for which the area is to be calculated.
- Nominal area can be expressed either in decimal or scientific notation. It has units of m², but no unit tag will appear after the value.
- The **NominalArea** keyword is required if the **AREA** keyword is used.
- If no **AREA** keyword is present, the area will not be written to the SPICE statement.

MOSFET

```

DEVICE=MOSFET (
    RLAYER = rLayer ;
    Drain = dLayer {[ , AREA] [ , PERIMETER [/GATE=#]] |
    [ ,GEO]};
    Gate = gLayer ;
    Source = sLayer [ , AREA] [ , PERIMETER [/GATE=#]];
    [Bulk = [bLayer ]];
    MODEL = model ;
) [IGNORE_SHORTS]

```

A MOSFET device has the following format in the SPICE output statement:

```

Mname drn gat src [blk] model L=lengthValue W=widthValue
    {[AD=areaValue] [PD=perimeterValue] [AS=areaValue]
    [PS=perimeterValue] | [GEO=#]}

```

The following rules apply to MOSFET devices:

- The length is the length of gate, and the width is the average length of the edges that the source and drain share with the recognition layer (**rLayer**). The length and width have units of meter.
- The optional **AREA** keyword may be specified on the drain or source pin or both. It is used to indicate whether the area for that layer will be calculated and written for the **AD** (Area of the Drain) and **AS** (Area of the Source) output values.

- The optional **PERIMETER** keyword may be specified on the drain or source pin or both, and is used to indicate whether the perimeter for that layer will be calculated and written for the **PD** (Perimeter of the Drain) and **PS** (Perimeter of the Source) output values. The shared edge is not included in perimeter calculations for a source or drain.
- The optional **/GATE=#** keyword is used with the **PERIMETER** keyword. It may be specified on the drain or source pin or both, but only where the **PERIMETER** keyword has already been designated. The number is a decimal value between 0.0 and 1.0, indicating the fraction of the gate width to include in the perimeter. If the **/GATE=#** keyword is missing, the perimeter will include the gate width.
- The optional **GEO** keyword may be specified on only the drain pin. It is used to indicate that the **GEO** value will be written to the SPICE statement. The **GEO** keyword can only be used if the **AREA** and **PERIMETER** keywords are not used. The values of **GEO** written to the SPICE statement are as follows:

GEO=0—drain and source areas are not shared

GEO=1—drain is shared

GEO=2—source is shared

GEO=3—both drain and source are shared

- The **GEO** keyword is used with the area calculation method (ACM) for modeling the source and drain diodes. Users are encouraged to use the **AREA** and **PERIMETER** options because they yield more accurate approximations of the area and perimeter values than those achieved using the **GEO** option.

Subcircuit

Subcircuits can be defined explicitly for the extractor. This method of describing subcircuits is different from automatic subcircuit instance recognition (see [Subcircuit Recognition on page 3-109](#)).

```

DEVICE=SUBCKT (
    RLAYER = rLayer [, AREA];
    pin1Name = pin1Layer [, AREA];
    pin2Name = pin2Layer [, AREA];
    . . .
    MODEL = model ;
    [NominalArea = areaVal ;]
) [IGNORE_SHORTS]

```

A subcircuit has the following format in the SPICE output statement:

```

Xzzz n1 n2 n3 ... cName [AREA=rLayerArea/areaVal]
[AREA_pin1Name=pin1Area/areaVal]
[AREA_pin2Name=pin2Area/areaVal] ...

```

The following rules apply to subcircuits:

- The optional **AREA** keyword may be specified on one or more layers (recognition or pin layer). An area will be calculated for each indicated layer.

- Nominal area can be expressed either in decimal or scientific notation. It has units of m^2 , but no unit tag will appear after the value.
- The **NominalArea** keyword is required if the **AREA** keyword is used.
- If no **AREA** keyword is present, the area will not be written to the SPICE statement.