Extracting Layout

- Introduction
- Configuring the Extractor
- Running the Extractor
- Subcircuit Recognition

Introduction

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.

Tools > Extract runs a general device extractor. Its features include the following.

- 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*. Devices that cross bin boundaries are extracted properly.
- 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 by Tanner Research's T-Spice circuit simulator (to verify device sizes, drive capabilities, and other circuit performance factors) or LVS netlist comparator (to check the equivalence of netlists generated from different sources) — or other tools that read SPICE netlists.

Configuring the Extractor

Configuration Concepts

The extraction process is set up by making associations between *patterns of layout geometry* and *the circuit components they represent*.

(See Configuration Example, for an illustration of the concepts described in this section.)

Devices and Connections

The first step is determining 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

The extractor recognizes patterns of geometric objects as devices. This is accomplished by specifying generated layers that uniquely define the devices and their pins.

For resistors and capacitors, the extractor also requires three constants to be entered for each involved layer (with **Setup > Layers** under the **General** tab):

- 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.

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.

The coordinates of off-grid vertices are rounded to the nearest Internal Unit while still preserving angles of edges. If the dimensions of the source objects (measured in Internal Units) are small, then the resulting polygons may be distorted.

To prevent distortion of generated objects due to rounding, a *subgrid* must be maintained. The subgrid is the smallest possible non-zero value, in Internal Units, of any edge length, wire width, distance between any two objects, or Grow distance.

The best way to maintain a subgrid is to set the mouse snap grid to prevent objects or spacings smaller than a specified value from being created. A snap grid equivalent to at least 100 Internal Units is recommended for designs containing 45° layout.

Extract Definition File

The *extract definition file* contains a list of the connections and devices to be extracted.

Configuration Example

This 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 of 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).

Several material layers have explicit names in the example file: the channel layer is *ntran*; the *n*-doped diffusion material is *ndiff*; the polysilicon layer is *Poly*; and the substrate layer is *subs*.

Device Definition

The following statement in the extract definition file 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 *N Select* 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.

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's default CMOS setup, which assumes a *p*-substrate, has a non-generated 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: source, drain, gate, 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 )
```

Running the Extractor

Tools > Extract performs netlist extraction from the active cell.

The options in this dialog are categorized under three tabs:

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

Subcircuit Recognition

Overview

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

Tools > 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 **x*instance 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 under the **Subcircuit** tab in the **Tools > Extract** dialog.

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.

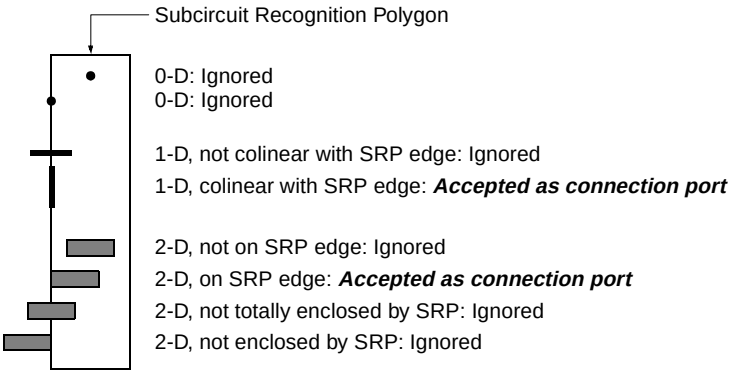
- Geometry over cross port channels.

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 **Tools > Extract** dialog, under the **Subcircuit** tab, in the **Ignore subcircuit connection ports with names** section:

Ignored ports	*How specified*
SPR core ports	Setup > SPR (under **Core Setup**)
SPR padframe ports	Setup > SPR (under **Padframe Setup**)
Ports on the Icon layer	**Setup > Special Layers**
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.

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.

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.

Certain named ports can be recognized as cross ports. These are shown in the **Tools > Extract** dialog, under the **Subcircuit** tab, in the **Subcircuit cross port names** section:

Cross ports	***How specified***
Row crosser ports	Setup> SPR (under **Core Setup**)
Ports matching a single additional name	**Other** text field