Extraction

A layout in Magic is a set of painted rectangles. Extraction is Magic reading those rectangles back and working out **what circuit they actually form**, which is not necessarily the circuit you meant to draw. That's the reason to do it at all, because an extractor that reported your intentions would tell you nothing you didn't already know.

The commands

```
extract do local          ;# extract this cell's own geometry
extract all               ;# ...and every subcell in it
ext2spice lvs             ;# format the result for netgen
ext2spice -o my_inv.spice
```

`extract` writes a `.ext` file in Magic's own format, which isn't very readable, and `ext2spice` turns that file into a SPICE netlist. The `lvs` argument is a preset that formats the netlist the way netgen wants it: no parasitic resistors, the subcircuits kept hierarchical, and no `.include` of the models.

Reading the netlist

Here is the extracted netlist of the reference inverter for Lab 03. Yours will have the same devices, though its ports may be listed in a different order, depending on the order in which you created the labels:

```
.subckt my_inv A Y VPWR VGND
X0 Y A VGND VGND sky130_fd_pr__nfet_01v8     ad=0.715 pd=3.5 as=0.6175 ps=3.2 w=0.65 l=0.15
X1 Y A VPWR VPWR sky130_fd_pr__pfet_01v8_hvt ad=1.1   pd=4.2 as=0.95   ps=3.9 w=1    l=0.15
.ends
```

Every field in it comes from something you drew:

Field	Where it came from
`.subckt my_inv A Y VPWR VGND`	the four labels you placed, promoted by `port makeall`
`X0 Y A VGND VGND`	drain, gate, source and bulk, worked out from which shapes touch which
`w=0.65`	the **height** of the n-diffusion under the poly
`l=0.15`	the **width** of the poly stripe where it crosses
`ad=0.715`	the area of the drain diffusion, in μm²
`pd=3.5`	the perimeter of the drain diffusion, in μm
`sky130_fd_pr__pfet_01v8_hvt`	the device flavour, decided by which implant layer you painted

`ad` and `pd` are easy to skim past, and they matter a great deal. Every square micron of drain diffusion is a reverse-biased junction to the body (the p-substrate under the NMOS, the n-well under the PMOS), and a reverse-biased junction is a capacitor, so the more drain area you draw, the more capacitance sits on the drain. Your inverter's drain area is **4.2 times** the foundry's, and Lab 05 measures what that costs.

Node names you did not choose

Extracting a NAND gate gives you a node with a name like `a_300_260#`:

```
X0 a_300_260# A VGND VGND sky130_fd_pr__nfet_01v8 w=0.65 l=0.15
X3 Y B a_300_260# VGND     sky130_fd_pr__nfet_01v8 w=0.65 l=0.15
```

That node is the diffusion shared between the two series NMOS transistors, the piece of silicon between them with no contact on it. You never labelled it, so Magic named it after its coordinates and added a `#` to mark it as internal. Seeing that name confirms that the two devices share one strip of diffusion, which is what you drew them to do.

So a `#` in a node name is nothing to worry about in itself. It is a problem when a node *you* labelled shows up with a `#`, because that means the label did not land on the geometry you thought it did.

PMOS flavours

The two PMOS models below are not interchangeable:

```
sky130_fd_pr__pfet_01v8        # regular threshold
sky130_fd_pr__pfet_01v8_hvt    # high threshold
```

Which one you get depends on whether the `pfethvt` implant layer covers the gate region. The foundry's standard cells all use the **hvt** PMOS. If you draw a PMOS and forget the implant, you get the regular one instead, and:

- **DRC never mentions it**, because both are legal devices.
- The layout looks identical to a correct one apart from a single hatched rectangle.
- LVS fails with a device-class mismatch. That's the outcome you want, because it means the mistake has been caught.
- Without LVS you would ship a chip with the wrong threshold voltages everywhere.

Lab 03's checker looks for this mistake and names it when it finds it.

Two kinds of extraction

You can extract the same geometry in two ways and get two different netlists:

```
ext2spice lvs             ;# devices only. This is what LVS wants.
ext2spice cthresh 0       ;# ...and every parasitic capacitance as well
```

`cthresh` is a threshold in femtofarads, and Magic throws away any capacitance smaller than it. The default drops the small ones, and since most of them are small, most of them go. If you set it to `0`, Magic keeps every capacitance it found:

```
.subckt my_inv A Y VPWR VGND
X0 Y A VGND VGND sky130_fd_pr__nfet_01v8     ...
X1 Y A VPWR VPWR sky130_fd_pr__pfet_01v8_hvt ...
C0 A Y      0.02936f
C1 Y VPWR   0.06727f
C2 A VPWR   0.10808f
C3 Y VGND   0.20685f
C4 A VGND   0.27253f
C5 VPWR VGND 1.33952f
.ends
```

That netlist has six capacitors that nobody drew. They're there because any two conductors near each other form a capacitor, and a layout is full of conductors near each other.

Use the `lvs` netlist for LVS and the `cthresh 0` one for simulation; Labs 03 and 04 and the capstone write both. If you compare a parasitic netlist against a schematic in netgen, the capacitors have nothing in the schematic to match, so the comparison produces only noise.

What extraction does not know

- **It does not know what you meant.** Checking the layout against what you meant is LVS's job, and that's the subject of the next page.
- **It does not model resistance by default.** `ext2spice` can extract resistors too, but that makes the flow much slower, and this course doesn't need it. Do keep in mind that li is about 12.8 Ω per square while metal1 is 0.125, so a long li wire is a real resistor, and the numbers in this course leave it out.
- **It does not model the third dimension beyond the layer stack**, which is why Magic is called a 2.5-D extractor. Its answers are good enough to be useful, but not good enough for sign-off.