Layout versus schematic (LVS)

DRC asks whether a layout *can be manufactured*. LVS asks a different question: whether the layout *is the circuit you meant*.

The two questions are independent. A layout can be perfectly manufacturable and still be a short circuit, and a layout can be exactly the right circuit and still be impossible to build. You need both answers, and they come from different tools.

netgen

The LVS tool you'll use is **netgen**, which compares two SPICE netlists as graphs. It ignores coordinates, port order, and the names of devices and internal nets. Instead it walks through the two circuits and asks whether there is a one-to-one correspondence between their devices and between their nets. The names of the cell's pins are the exception, and the first failure below is what you get when they disagree.

```
netgen -batch lvs \
  "results/my_inv.lvs.spice my_inv" \
  "spice/my_inv_sch.spice my_inv" \
  "$PDK_ROOT/sky130A/libs.tech/netgen/sky130A_setup.tcl" \
  results/lvs.report
```

The command takes four arguments: your layout netlist and the cell in it, the schematic netlist and the cell in it, the PDK's setup file, and the file to write the report to.

The setup file is easy to treat as a formality, but netgen depends on it. It tells netgen that `sky130_fd_pr__nfet_01v8` has interchangeable source and drain, which W and L to compare and with what tolerance, and which properties to ignore. Without it, netgen reports every symmetric device as a mismatch.

A match

```
Final result: Circuits match uniquely.
```

The word "uniquely" means that netgen found exactly one correspondence between the two circuits. A symmetric circuit can have several, and when it does, netgen says so.

Two failures to recognise

"Netlists match uniquely" with "failed pin matching"

This is the more confusing of the two failures, so in Lab 04 we have you produce it on purpose with `make swapped`:

```
Netlists match uniquely.

Subcircuit pins:
Circuit 1: my_nand2         |Circuit 2: my_nand2
----------------------------|----------------------------
VPWR                        |VPWR
Y                           |Y
VGND                        |VGND
B                           |A **Mismatch**
A                           |B **Mismatch**

Final result: Top level cell failed pin matching.
```

The report is making two separate statements, and it helps to read them apart:

- **The topology is right.** The layout has four devices, wired exactly as the schematic says.
- **Two names are on the wrong shapes.** To make the match work, netgen had to swap `A` and `B`.

A NAND is symmetric in its inputs, so the circuit still works, and nothing else in the flow will ever notice the swap: simulation passes and DRC passes. The timing model is wrong all the same, because in a series NMOS stack the input nearest the output is measurably faster than the one nearest ground.

To fix it, swap the two labels in the layout, not in the schematic.

Different device or net counts

```
Circuit 1 contains 4 devices, Circuit 2 contains 5 devices.
```

When the counts differ, something is missing from one of the circuits. In these labs the cause is almost always one of these:

- **A wire that doesn't land.** Your li (local interconnect) or metal wire stops 0.02 µm short of the pin it was meant to touch, so the layout has one more net than the schematic, and one input floats.
- **A missing label.** Without a label there is no port, and so nothing for netgen to compare.
- **A missing tap**, which turns the bulk terminals into invented nodes. The full DRC rule set explains this.
- **The wrong device flavour**, such as `pfet_01v8` in the layout where the schematic says `pfet_01v8_hvt`. Netgen reports this as a device class with no partner.

Whatever the cause, read the report from the bottom up. The last table before `Final result:` names the device or net that couldn't be matched, and that name points you at the problem.

Where the schematic netlist comes from

In this course the schematic netlist comes from a file. Each lab ships a small `.spice` file with the circuit written out by hand, because a hand-written netlist is unambiguous and runs in a bare container:

```
.subckt my_inv A Y VPWR VGND
XM1 Y A VGND VGND sky130_fd_pr__nfet_01v8     w=0.65 l=0.15
XM2 Y A VPWR VPWR sky130_fd_pr__pfet_01v8_hvt w=1    l=0.15
.ends
```

In a real project it comes from your schematic editor. With XSchem, which you used in AD103, the command is `xschem -n -s -q -x file.sch`. There is one trap here that's worth knowing about now, so that it doesn't cost you an afternoon later:

> If you netlist a schematic on its own, XSchem writes its devices at the top level with no `.subckt` around them, and netgen then says `Cannot find cell my_inv`. Instead, netlist a *wrapper* schematic that instantiates your symbol, so that the netlist has a subcircuit in it for netgen to compare.

Why the labs run LVS

DRC failures are loud and local: Magic hatches the spot and `drc why` names the rule, so they're cheap to fix. LVS failures are quiet, and the cause can be anywhere in the cell.

Take a wire that stops 0.02 µm short of its pin. At any zoom you would normally use, it looks perfect. It passes DRC, because a gap doesn't violate any rule, and it extracts without complaint. Netgen is the only tool in the flow that notices, because it counts the nets and finds one too many.

That's why every lab in this course except Lab 05 runs LVS, including Lab 02. In Lab 02 the point is that **erasing the shape that broke a DRC rule doesn't fix it**: the DRC count drops to zero, but LVS immediately reports that the cell is no longer an inverter.