Lab 04: Two resistors that disagree

In this lab you'll watch the same resistor change value at every process corner while a ratio of two resistors stays fixed. Then you'll run a Monte Carlo mismatch simulation that anyone can reproduce, check Pelgrom's law against your own data and find where it stops working, and learn to tell **accuracy** (the nominal value is wrong) apart from **precision** (the spread is wide). The lab ends with a divider that you have to make accurate, precise and small at the same time.

Package: `labs/lab-04-two-resistors-that-disagree/`

Before you start

- Finish Lab 01; you'll need its 378.2448 Ω of end resistance. Lab 02 and Lab 03 help, but you don't need them.
- Read Process corners and Matching and Pelgrom's law.
- Open a terminal in the workbench and go to the lab folder:

```
cd /foss/designs/modules/ad102/labs/lab-04-two-resistors-that-disagree
```

  This lab needs no environment setup.

Parts 1 to 6 are required. Everything under Going further, at the end, is optional.

Time: about 2 hours, not counting the optional part. The Monte Carlo runs print nothing while they work, so don't mistake the quiet for a hang. A full `make` takes between about 30 seconds and two minutes, depending on your machine (the Makefile records 106 s and 128 s on the pinned image; see Warnings and delays).

Labs 01–03 assumed that a device is worth what its geometry says: you sized a resistor to 2200 Ω and the simulator returned 2200.006 Ω. That settles the design, but the factory still has to build it. The strip you drew is 1 µm wide on the mask, and the finished silicon is 1 µm wide plus or minus whatever the lithography, etch, implant dose and anneal did that day, which is slightly different for every device on the die. So a resistor's value is a random variable, and so is every other device parameter on a chip. You can't design that out, because it comes from building a million things at once, and much of analog design is about arranging for the randomness not to matter.

Part 1: Process corners

What moves between process corners, and what stays put?

The corners deck in `labs/passives-decks` is the one the two guide pages quote. Its circuit has two resistors, R_A drawn 1 × 10 µm (`r_a`) and R_B drawn 1 × 40 µm (`r_b`), a divider built from the same two shapes, and an RC made of a 1 × 10 µm resistor and a 10 × 10 µm MIM capacitor. The deck builds that circuit three times, once per corner (`ll`, `tt`, `hh`), and the only thing it changes between them is one word on the `.lib` line. `ratio` is R_B/R_A, which is drawn as 4 to 1.

1. Go to the `passives-decks` package and run the three corners:

```
cd ../passives-decks
make corners
```

2. Read the three `absolute (moves)` blocks. Divide the `ll` and `hh` values of `r_a` by the `tt` value.
3. Read the three `RC corner frequency` lines. Before you compare them, work out what the `hh` corner should do to the corner frequency: R goes up by 1.125 and the MIM capacitance (the `MIM capacitance` lines) by 1.13207, so f should scale by $1/(1.125 \times 1.13207)$.
4. Read the three `ratio and divider (do not move)` blocks.

5. Go back to the lab folder:

```
cd ../lab-04-two-resistors-that-disagree
```

You should see: `r_a` at 3106.637, 3550.443 and 3994.248 Ω, so `ll` and `hh` are 0.875000 and 1.125000 of typical. The corner frequencies are 282.935, 215.410 and 169.052 MHz, and your prediction for `hh`, $1/1.2736 = 0.7852$, compares with a simulated 0.7848. In all three corners you should also see:

```
ratio = 3.680396e+00
divider = 3.845828e-01
```

The absolute values moved by ±12.5 % and the RC corner frequency over a 1.67 : 1 range, yet the ratio and the divider didn't change in any printed digit. Why not?

Answer: A1, in Answers at the end.

Part 2: Run the Monte Carlo

Can you run a Monte Carlo that someone else can reproduce exactly?

A corner shifts every device together, whereas mismatch moves devices apart from each other, so it needs a different switch. `spice/mismatch.spice` turns the SKY130 mismatch models on with one line:

```
.param mc_mm_switch=1
```

With it on, every device instance draws its own `AGAUSS()` sample when the netlist is evaluated. The control block repeats two steps two hundred times: it calls `mc_source`, which re-evaluates the netlist (in effect, it fabricates a different die), and then it runs one `op`. After that, `alterparam mc_mm_switch=0` followed by one more `mc_source` gives the nominal die, with no mismatch at all. `setseed 1` fixes the random sequence, so your two hundred dies are the same two hundred dies as on this page, in the same order.

1. Run the lab:

   ```
   make
   ```

   It ends in `FAIL`, which you'll fix in Part 6. Leave the output in your terminal.

2. Before you trust any statistic, check the count at the top of the output. If it says 2 instead of 201, the loop stopped after one die, and Warnings and delays explains why.

You should see: the count, then the nominal die. Section A of the deck forces 1 µA through four single resistors, so dividing `v(s1)` to `v(s4)` by 1 µA gives their nominal resistances: 695.4646, 1647.124, 1468.119 and 1370.854 Ω. `v(ta)` to `v(td)` are the divider taps you'll read in Parts 4 and 5.

```
operating points (200 dies + 1 nominal) : 201
--- NOMINAL (mismatch off) ---
v(s1) = 6.954646e-04
v(s2) = 1.647124e-03
v(s3) = 1.468119e-03
v(s4) = 1.370854e-03
v(ta) = 9.000000e-01
v(tb) = 9.000000e-01
```

```
v(tc) = 1.309327e+00
v(td) = 1.350000e+00
```

A Monte Carlo result that nobody can reproduce is an anecdote rather than a measurement. Seeding the run costs one word, and it turns "about 5.6 mV" (a spread you'll meet in Part 5) into a number you can put in a table and have someone else check.

Part 3: Pelgrom's law

Does σ halve when the area quadruples?

Section A of the deck puts 1 µA through four single resistors with areas of 1, 4, 16 and 64 µm².

1. Read the mismatch coefficients off the model card:

   ```
   grep                  pelgrom                  /foss/pdks/sky130A/libs.ref/sky130_fd_pr/
   spice/sky130_fd_pr__res_high_po.model.spice
   ```

   ```
   + body_pelgrom = 0.03552
   + head_pelgrom = 0.0761
   + rbody_match = {body_pelgrom/sqrt(w*l*mult)*MC_MM_SWITCH*AGAUSS(0,1.0,1)}
   +                                      rend_match                                          =
   {head_pelgrom/sqrt((w+0.525)*num_con_row*mult)*MC_MM_SWITCH*AGAUSS(0,1.0,1)}
   + res_match = {(body_pelgrom/sqrt(w*l*mult))*MC_MM_SWITCH*AGAUSS(0,1.0,1)}
   ```

   The first two lines are the coefficients, and the last three are the model that uses them. There are two coefficients because a resistor has two parts that mismatch independently of each other: the body and the contact head.

2. Predict all four spreads from `body_pelgrom / sqrt(W*L)`.

3. Find table A in the checker's output (the checker's tables use the deck's section letters).

You should see:

```
A. one resistor, four sizes -- does sigma halve when area quadruples?
  geometry          area    sigma/R   0.03552/sqrt(A)   ratio to previous
  W=1  L=1             1     2.5852%           3.5520%                  --
  W=1  L=4             4     1.6046%           1.7760%               1.611
  W=2  L=8            16     0.8391%           0.8880%               1.912
  W=4  L=16           64     0.4007%           0.4440%               2.094
```

The last column tests the law, and for the three larger devices it holds: 1.912 and 2.094 against a predicted 2.000, from 200 samples that can only pin σ down to ±5 % anyway.

The 1 µm² device doesn't follow the law. It should have scattered by 3.55 %, but it scattered by 2.59 %, which is 27 % less than predicted, so it does *better* than the law allows. Why? (Look back at Lab 01, Part 6.)

Answer: A2, in Answers at the end.

Part 4: The same divider at 48 times the area

What does 48 times the area buy you in precision?

Section B builds a 1:1 divider twice, once from two 0.84 µm² devices and once from two 40 µm² ones. The schematic and the ratio are the same in both, and only the size changes.

1. From area alone, predict how much more precise the big pair should be: $\sqrt{40/0.84}$.

2. Find the two `1:1` lines in the checker's `B and C` table.

You should see:

```
1:1  two devices W=0.42 L=2    0.84 um^2 each   0.900000 V  25.0787 mV
1:1  two devices W=2 L=20        40 um^2 each   0.900000 V   3.2321 mV
```

Both nominals are exactly 0.900000 V, so both dividers are perfectly accurate, but one is eight times less precise than the other. 25 mV of scatter on a 900 mV node is 2.8 %, which is roughly five bits' worth of reference error in an ADC, and it comes from a circuit whose schematic is correct. The fix is the least clever thing in engineering: draw it bigger.

47.6 times the area bought 7.76 times the precision, but area alone predicts $\sqrt{47.6} = 6.90$. Why is the small pair about 12 % worse than its area explains?

Answer: A3, in Answers at the end.

Precision is paid for in square micrometres, and that is the main reason analog blocks look enormous next to digital ones.

Part 5: One ratio drawn two ways

Does building a ratio from identical units help its accuracy, its precision, or both?

Section C builds a 3:1 divider twice, each time from exactly 24 µm² of poly, with the tap meant to sit at three quarters of the 1.8 V supply:

- `tc`: one device of $L = 6$ from `vdd` to the tap and one of $L = 18$ from the tap to ground. This is the way most people would draw it.
- `td`: four identical $L = 6$ units in series, tapped after the first one down from `vdd`.

On paper the two have the same ratio, and they use the same area and the same material.

1. Predict the nominal and the spread for each, and decide which will be better and why.

2. Find the two `3:1` lines and the summary below them.

You should see:

```
3:1  one L=6 and one L=18        24 um^2 total  1.309327 V   5.6433 mV
3:1  four identical L=6 units    24 um^2 total  1.350000 V   5.7290 mV

the 3:1 ratio you asked for is 1.350000 V.
  two devices  : 1.309327 V  (-3.01 %)
  four units   : 1.350000 V  (+0.00 %)
and the spread is 5.6433 mV against 5.7290 mV -- a 1.5 % difference,
where 200 samples only pin sigma down to +/- 5.0 %. Those are the same number.
```

The nominals differ by 3.01 %, and the two-device version is wrong in the same direction, by the same amount, on every die. Why?

Answer: A4, in Answers at the end.

The spreads, 5.6433 against 5.7290 mV, are 1.5 % apart. Did the unit cells buy any precision?

Answer: A5, in Answers at the end.

So “matching” usually means two different things, and you pay for each of them differently:

	what fixes it	what it costs
accuracy: the nominal is wrong	identical unit devices	nothing
precision: the spread is wide	area	area, at 4× per factor of 2

Build every ratio you care about from identical unit cells. It costs nothing, and it removes a whole class of error, including errors that nobody has told you about yet.

> **What this simulation cannot show.** Every analog textbook tells you to lay matched devices out as interleaved unit cells in a common centroid, and yet you have just measured that unit cells do nothing for σ. Both are true, because this simulation contains only random, per-device mismatch. Real wafers also have **gradients**: oxide thickness, implant dose and temperature vary smoothly across a die, so two devices ten micrometres apart are more alike than two a millimetre apart, and the left member of a pair sits in a systematically different environment from the right one. Interleaved unit cells in a common centroid cancel the linear part of that gradient. It’s a large, real effect, and it’s the main reason the technique exists.
>
> `AGAUSS` knows nothing about where your devices are, because your netlist doesn’t say where they are. Position enters a design at layout, and a SPICE netlist has no coordinates. If you build the interleaved version and run ten thousand dies, you will measure zero benefit, because what you’re measuring is the model rather than the silicon. Before you use a simulation to check a design decision, check that the simulator models the effect the decision is about.
>
> Post-layout extraction, which AD104 covers, doesn’t fix this either, because it adds parasitic R and C from your actual geometry and says nothing about process gradients. Gradient matching is checked with test chips and layout review instead, and this is one of the real limits of simulation.

Part 6: Design a 0.45 V divider

Can you make one divider accurate, precise and small at the same time?

`spice/my_divider.spice` has to put 0.4500 V on `tap` from a 1.8 V supply, using `sky130_fd_pr__res_high_po`, and meet all three of these:

1. nominal within 0.2 % of 0.4500 V,
2. σ at most 3.0 mV over 200 dies,
3. total drawn area under 150 µm².

You can use as many devices as you like. Keep the node called `tap` and the supply called `vdd`, because the checker reads your geometry back out of the file.

The file ships with the divider everybody draws first: one resistor on top three times as long as the one underneath (`L=18` and `L=6`, both `W=1`). That is Part 5’s `tc` turned upside down, but it runs in a separate deck with its own 200 dies, which is why its σ differs from `tc`’s. When you ran `make` in Part 2, it failed two of the three checks:

```
  1. nominal tap   0.490673 V    target 0.450000 V ( +9.04 %)      FAIL
  2. sigma           5.9285 mV   budget 3.0000 mV                    FAIL
  3. drawn area       24.00 um^2 budget    150 um^2                 PASS
```

1. Decide how to fix the nominal (Part 5) and how to fix σ (Parts 3 and 4).
2. Edit `spice/my_divider.spice`. Write `W` and `L` as plain microns, with no `u` suffix.

3. Re-run your divider and the checker:

```
make mine check
```

You should see: all three checks marked `PASS`, then

```
PASS -- every number matches the reference run.
```

Finishing

You're done when `make mine check`, run on the divider you designed in Part 6, prints `PASS -- every number matches the reference run.`

`setseed 1` makes the Monte Carlo reproducible, so on the pinned image (ngspice-47, `sky130A`, `tt` corner, seed 1, 200 dies) your numbers match the ones on this page.

Question: the shipped divider failed two checks. Which failure did area fix, and which did identical units fix?

Answer: A6, in Answers at the end.

After yours passes, read `solutions/README.md` for one good answer and why unit devices do nothing for σ in this simulator.

Submit: nothing for this lab.

Going further (optional)

This part isn't in the time estimate, and nothing later in the course needs it. What to do next on the Course path is on the last line of this page.

The guide page's mismatch run

The comparison of 1 × 1 µm against 10 × 10 µm on Matching and Pelgrom's law uses a different seed, and its deck is in `labs/passives-decks`. To reproduce it:

```
cd ../passives-decks
make mismatch
cd ../lab-04-two-resistors-that-disagree
```

It ends with `100x the area bought 8.38x better matching`.

Next on the Course path: read From schematic to cross-section, then do the Capstone: Defend a floor plan.

Answers

A1. Part 1: Process corners

A corner applies the same multiplier to every device made of that material, and in a ratio of two such devices the multiplier cancels, which leaves a ratio of the lengths you drew. Matching and Pelgrom's law works it through, including why the ratio is 3.680396 rather than 4.

A2. Part 3: Pelgrom's law

At $W = 1$, $L = 1$ the resistor's total is 695.46 Ω, and 299.89 Ω of that, or 43 %, is contact head rather than body. (The rest of Lab 01's 378.2448 Ω of ends is the body's extra 78 Ω, as Lab 01, Part 6 found.) `body_pelgrom` describes only the body, so the 3.55 % applies only to the body's share of the resistance. The head has its own coefficient (`head_pelgrom`, averaged over a different quantity), and on this device the head contributes about as much to σ as

the body does. The worked example gets 13.067 Ω from the head and 14.051 Ω from the body, which combine to a σ of 19.188 Ω, or 2.759 % of 695.4646 Ω. That is much closer to what you measured than 3.55 % is.

So a mismatch coefficient describes one *parameter* of a device rather than the whole device. Once your geometry stops looking like the geometry the coefficient was extracted on, the single-number version of the law no longer applies. It's the same trap as Lab 01's naive $R = R_\square L/W$, and it has the same fix: measure the thing you are actually going to build.

A3. Part 4: The same divider at 48 times the area

Each small device is 0.42 µm wide, which is below the 0.69 µm point where Lab 01, Part 4 found that the sheet resistance starts to change. It also pays head mismatch, which doesn't scale with WL at all.

A4. Part 5: One ratio drawn two ways

Lab 01 found that every resistor pays 378.2448 Ω of end resistance whatever its length, so

$$\frac{R_6}{R_6 + R_{18}} = \frac{378.24 + 6(317.22)}{8369.76} = 0.27260$$

of the supply drops across the top device instead of 0.25, and the tap sits at 1.309327 V instead of 1.350000 V. If you build both branches from the *same unit*, that overhead appears equally in both, so the ratio becomes a ratio of **counts** (one unit out of four) and the overhead divides out exactly. The measured nominal is `1.350000`.

A5. Part 5: One ratio drawn two ways

No. 200 samples can't resolve σ better than $1/\sqrt{2N} = 5$ %, so the two spreads are the same number. Splitting the resistor into unit cells bought nothing in random mismatch, and the algebra says it never will: when the errors are independent from device to device, the σ of a ratio depends on the total area and on nothing else about how you divided it up.

A6. Finishing

Identical units fixed the nominal, which was a systematic error from the end resistance (Part 5). Area fixed σ, at a cost of four times the area for each factor of two (Parts 3 and 4). The lab's `solutions/README.md` has one design that does both.