23 April

Initial Design of Delbruck's Bump

Brief

The Simple Bump circuit is used for computing the similarity of analog voltages. It consists of two sub-circuits, a current correlator ($M_{p1} - M_{p4}$) and a simple differential pair as shown in Figure 1.
To avoid unnecessary added complexion, we will guarantee that all transistors operate exclusively in the sub-threshold region. Therefore, the currents for the PMOS and NMOS devices are, respectively:

$$I_{pmos} = I_{op} e^{\kappa_p (V_w - V_G)/V_T} (e^{(V_S - V_w)/V_T} - e^{(V_D - V_w)/V_T}) \tag{1}$$

$$I_{nmos} = I_{on} e^{\kappa_n (V_G - V_w)/V_T} (e^{(V_w - V_S)/V_T} - e^{(V_w - V_D)/V_T}) \tag{2}$$

Where the characteristic currents I_o can be calculated using:

$$I_{o(n,p)} = \frac{1}{2} \mu_{(n,p)} C_{ox} \left(\frac{W}{L}\right) \left(\frac{kT}{q}\right)^2 \tag{3}$$

Our goal is to tune the parameters of our circuit to ensure $I_{out} = I_{bias}$ when $V_{in} = V_M$. The output current is given by:

$$I_{\text{out}} = \frac{I_{\text{bias}} S}{4 \cosh^2 \left(\frac{\kappa V_{\text{in}} - V_{\text{mean}}}{2}\right)} \tag{4}$$

where I_{bias} and V_{mean} control the height and the mean value, respectively. The quantity S is given as such:

$$S = \frac{(W/L)_{3,4}}{(W/L)_{1,2}} \tag{5}$$

where κ is the slope factor and V_{in} is the input voltage.
The height and the mean value are independently tuned via the circuit's parameters, whereas the deviation is altered by the effective W/L ratio (via transistor's dimensions).

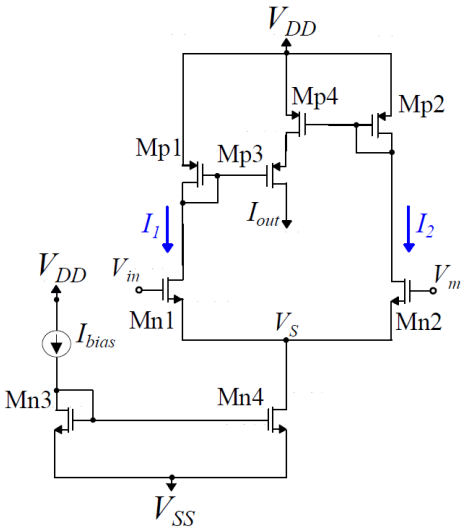


Figure 1: Transistor level implementation of Delbruck's Simple Bump

Simulations

For our simulations, we will use the testbench shown in Figure 2. With:

- $I_{bias} = 5nA$
- $V_{DD} = 0.3V$
- $V_{SS} = -0.3V$

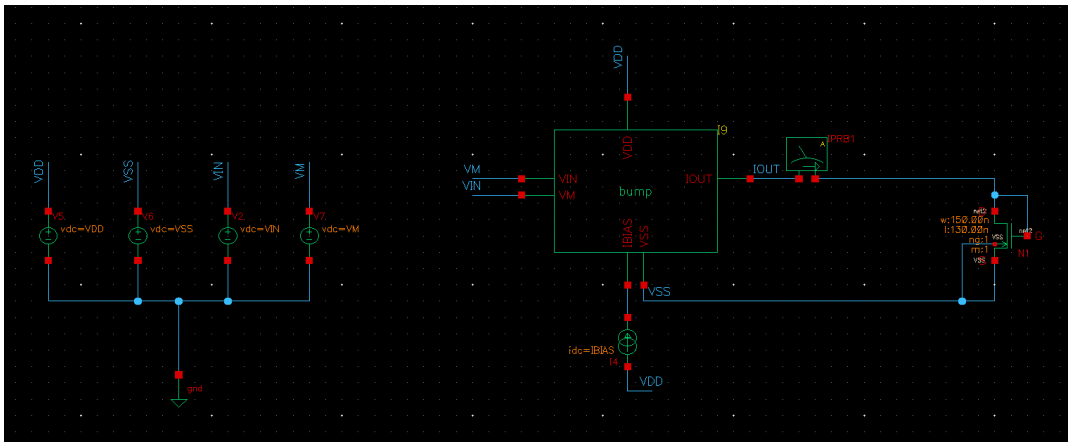


Figure 2: Delbruck Bump testbench in Cadence

Our maestro set-up (variables and parameters) is shown in Figure 3. At this stage, we have matched everything (multipliers, W, L, number of gates). For our simulation we run a dc analysis Voltage Sweep on VIN in the range of our power supply and an xf analyses as shown in Figure 4.

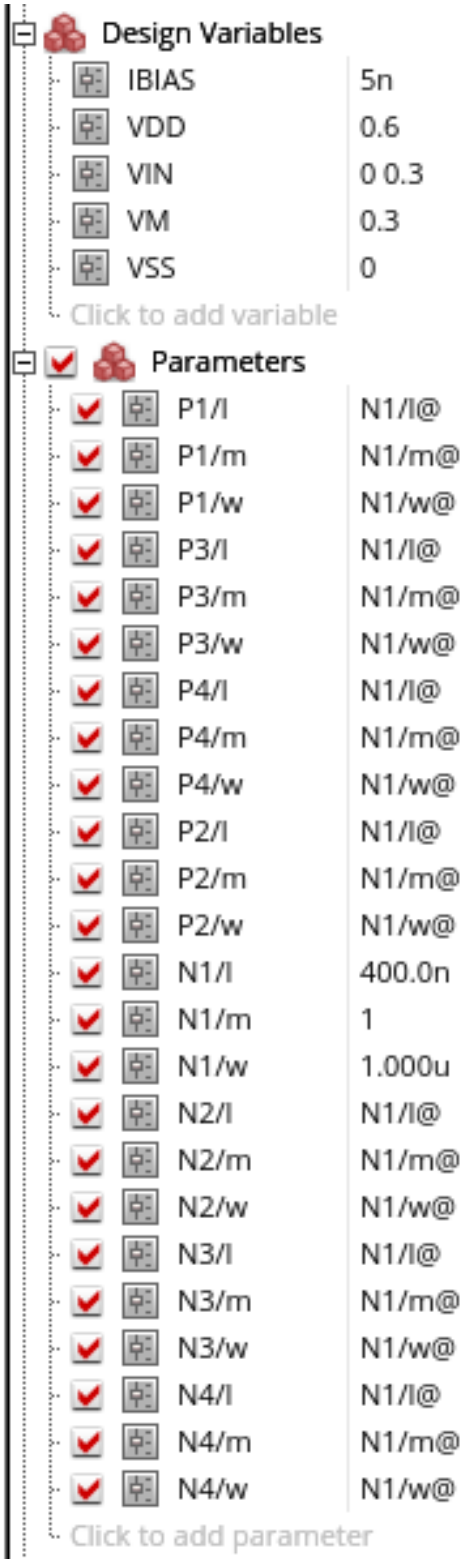
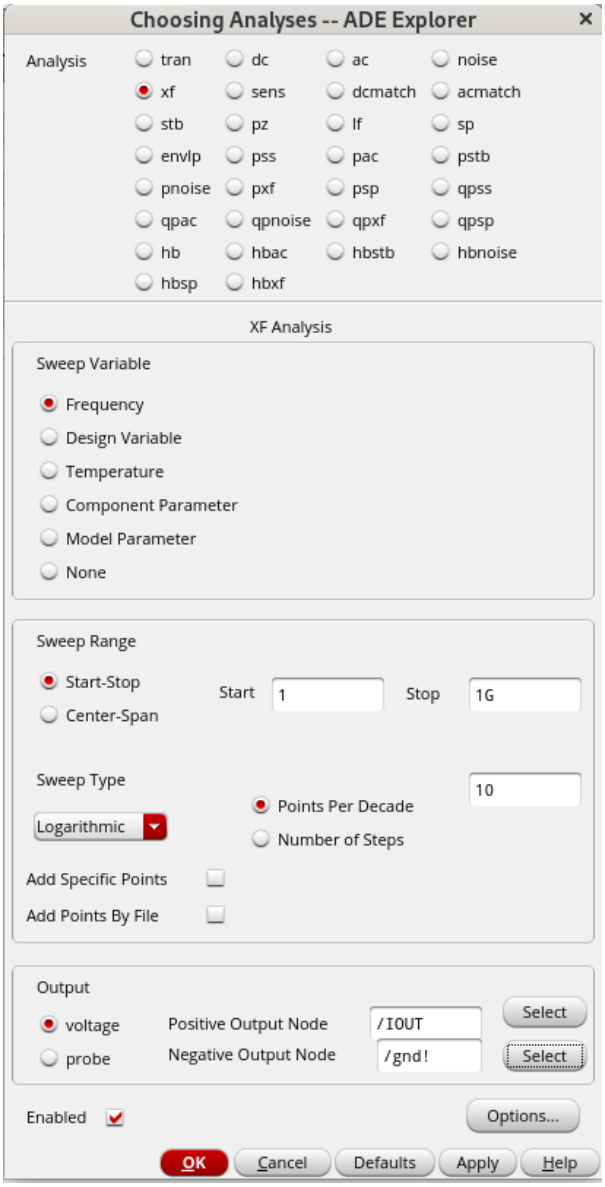
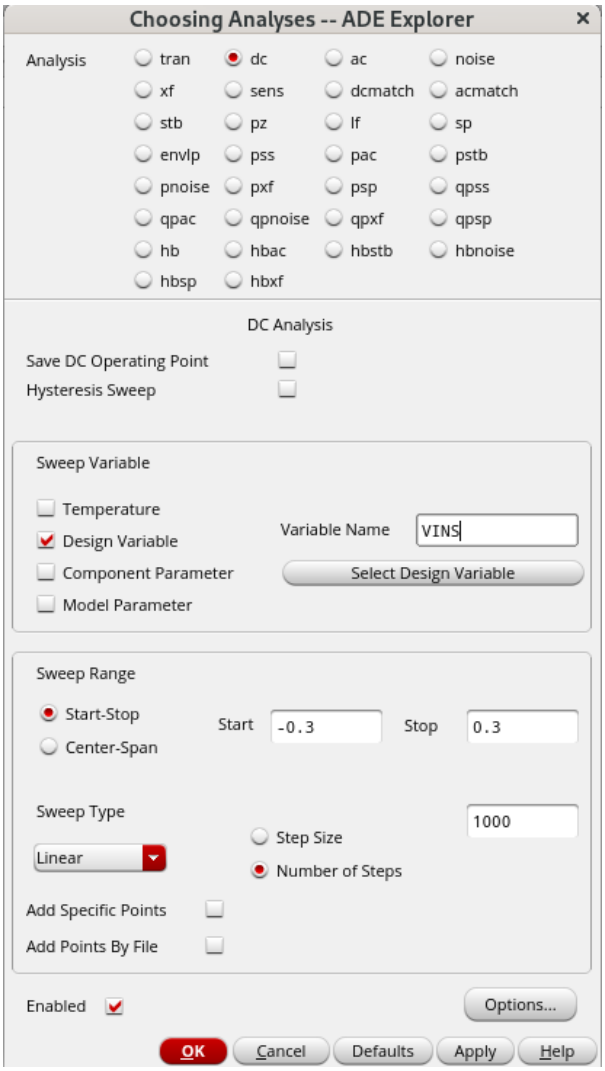


Figure 3: Caption



(a) Output current



(b) Output voltage

Figure 4: DC and XF analyses set-up

For our waveforms we used the expressions:

- **Iout**: i("/IPRB1PLUS" ?result "dc")
- **Vs_out**: VS("/IOUT")
- **IORAMP**: IS("/V5/PLUS")
- **POWER**: (IORAMP * VDC("/VDD"))
- **AVDD**: getData("/V5" ?result "xf")
- **AVSS**: getData("/V6" ?result "xf")

- **AV**: getData(”/V2” ?result ”xf”)
- **PSRR+**: dB20((AV / AVDD))
- **PSRR-**: db20((AV / AVSS))
- **Noise**: db(getData(”out” ?result ”noise”))

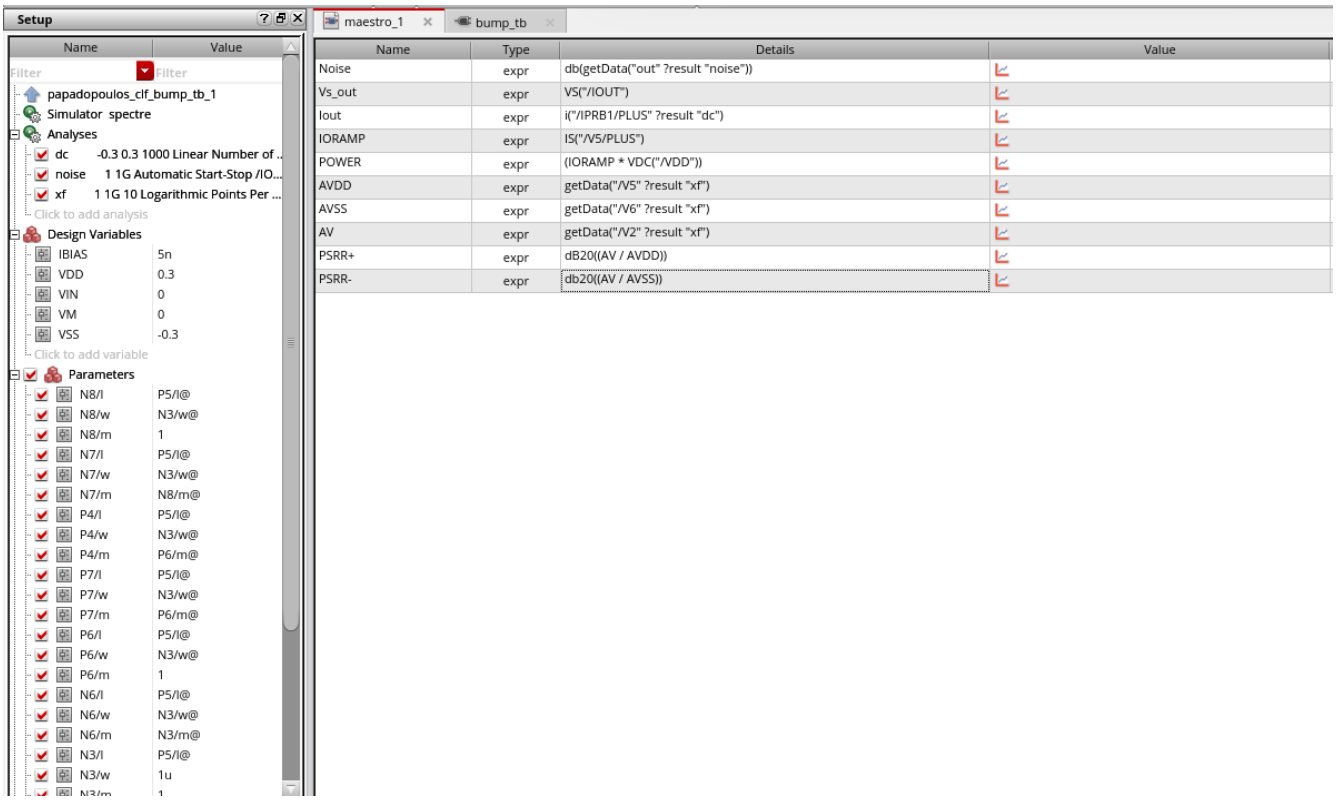


Figure 5: An overall view of our maestro tesbench for simulations

Our initial simulation used the dimensions shown in Table 1, with the corresponding results presented in Figure 5. It should be noted that, at this stage, we simply selected some dimensions to ensure that the setup was working correctly. We have not yet begun our actual analysis. As is evident, the current parameters clearly do not meet our intended objectives.

Table 1: MOS Transistors Dimensions: Simulation 1

Block	**W/L (µm/µm)**	**Current Correlator**	**W/L (µm/µm)**
M_{n1}, M_{n2}	1.0/0.4	M_{p1}, M_{p3}	1.0/0.4
M_{n3}, M_{n4}	1.0/0.4	M_{p2}, M_{p4}	1.0/0.4

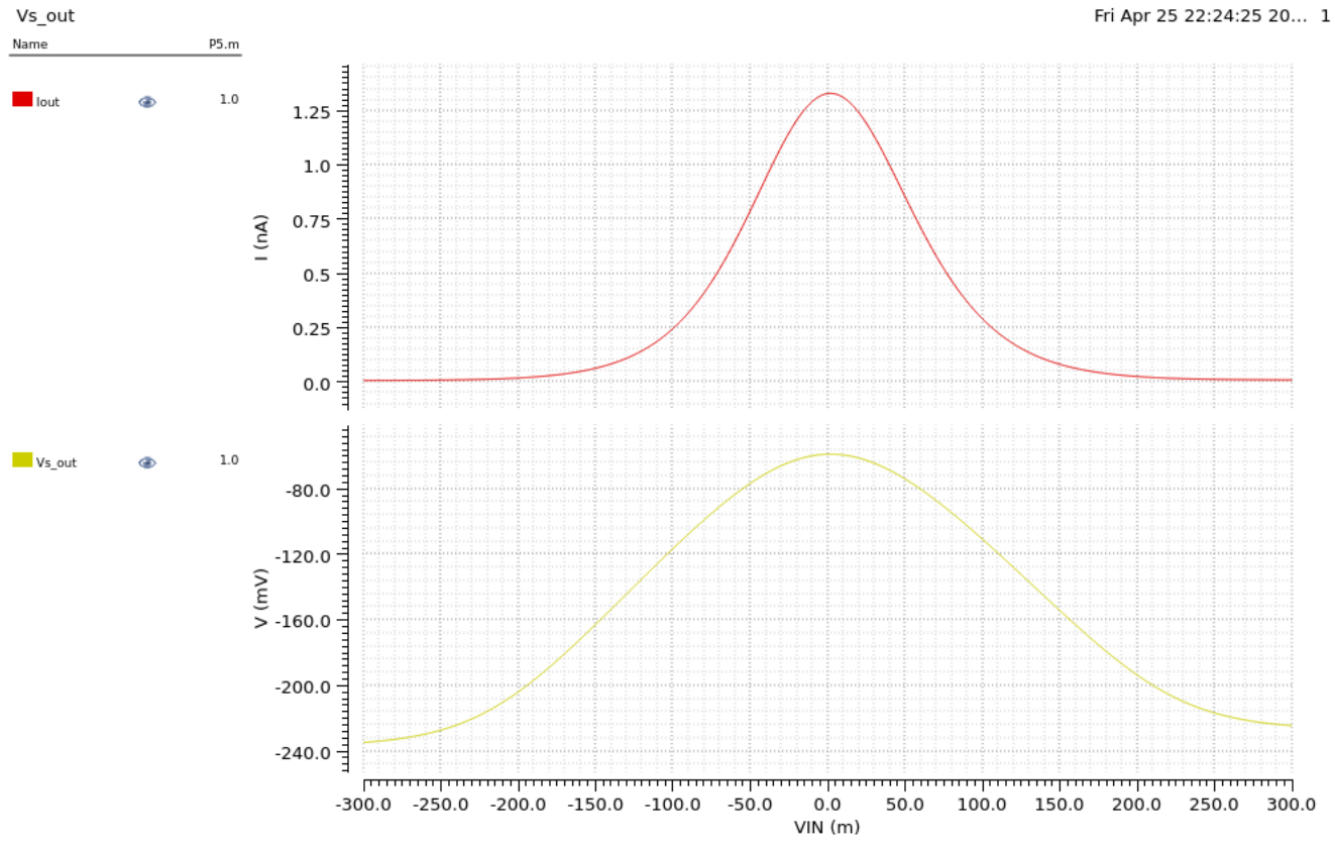


Figure 6: Iout and Vout of sim1

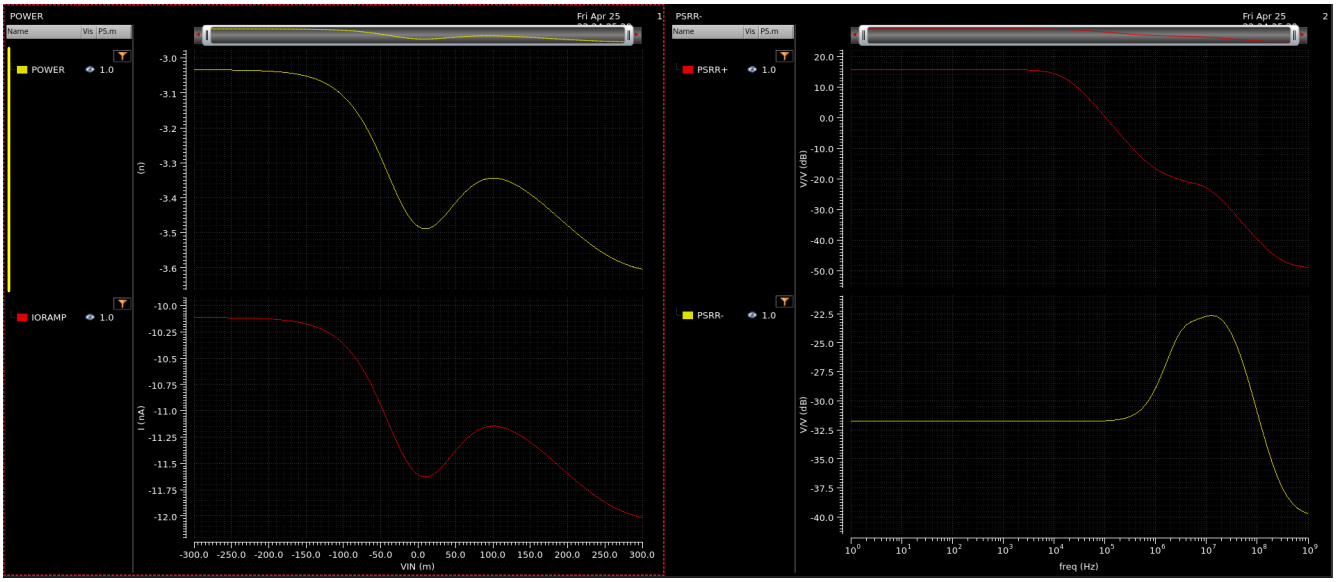


Figure 7: Power, Ioramp, PSRR+ and PSRR- of sim1

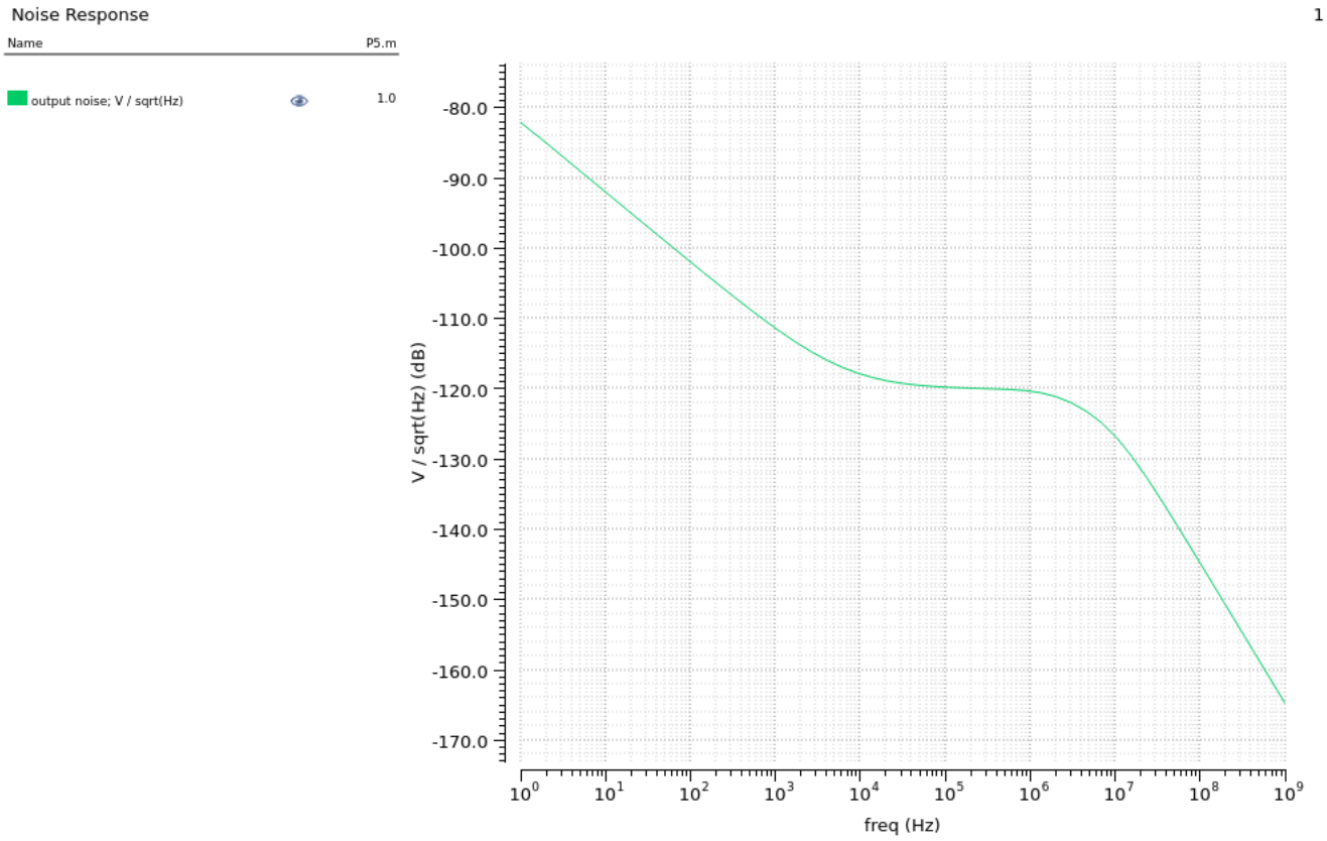


Figure 8: Noise of sim1

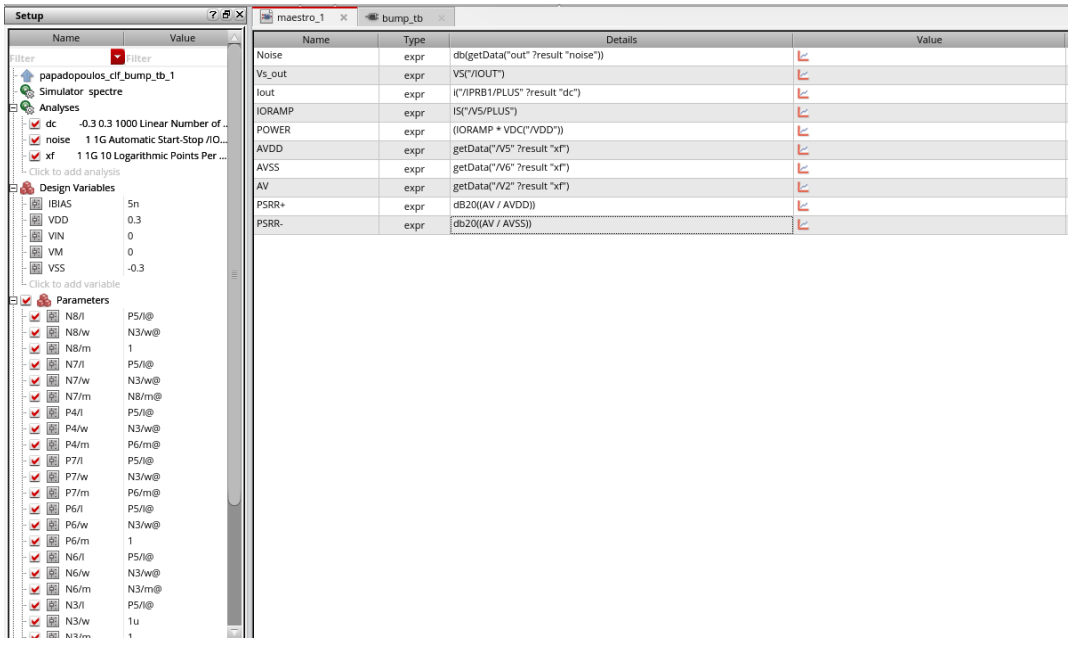


Figure 9: An overall view of our maestro tesbench for simulations

25 April

Simulations

Choice Analysis

Based on Equation (4) from the previous logbook entry, it becomes evident that what we are truly aiming for is:

$$\frac{S}{4\cosh^2\left(\frac{kV_{\text{in}}-V_{\text{mean}}}{2}\right)} = 1 \Rightarrow S = 4 \Rightarrow \frac{(W/L)_{3,4}}{(W/L)_{1,2}} = 4$$

For proper circuit operation and symmetry, careful transistor matching is required. Within the PMOS current correlator, Mp_1 should be matched to Mp_2, and Mp_3 should be matched to Mp_4. Furthermore, it is essential to match the transistors within the NMOS differential pair ($Mn_1 - Mn_2$) to minimize the input offset voltage and ensure that the output peak aligns accurately with $V_{in} = V_r$. Matching within the current source circuitry ($Mn_3 - Mn_4$) providing I_{bias} is also important for accurate biasing.

So, our next simulation is executed with the values in Table 1. Obviously, even with just one parameter changed, we already have a much better result as shown by the I_{out} plot in Figure 1 (the rest of the outputs are omitted at this point).

Table 1: Transistor Parameter Values

Device Name	Length (L)	Width (W)	Multiplier (m)
N1, N2	300 nm	1.0 µm	1
N3, N4	300 nm	1.0 µm	1
P1, P2	300 nm	1.0 µm	1
P3, P4	300 nm	1.0 µm	4

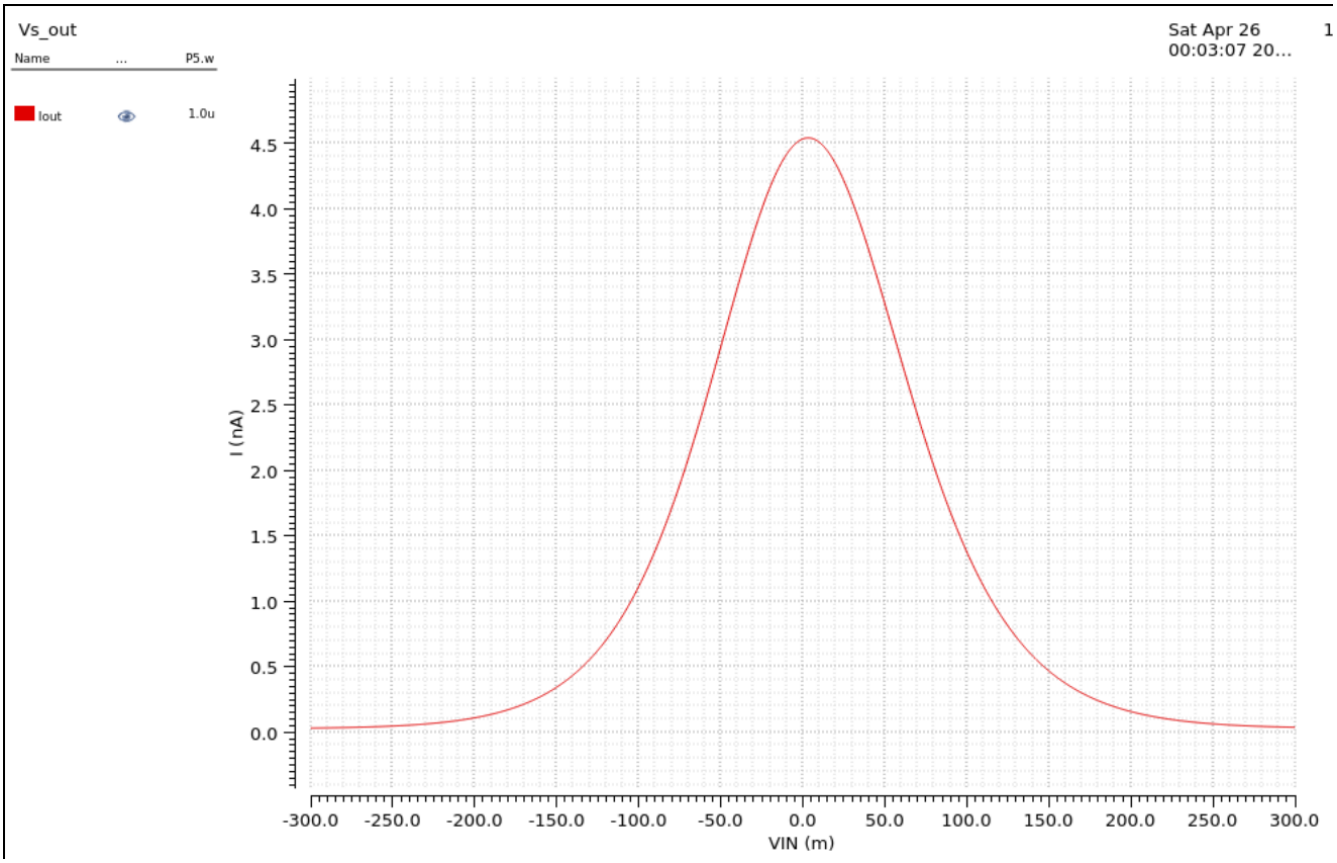


Figure 1: Sim2 Iout

However, it is still not perfect as it is not exactly equal to I_{bias} and there are also some asymmetry issues.

From this point on, we will run parametric simulations to tune our circuit.
Since $I_{\text{out}} < I_{\text{bias}}$, it follows that our scaling factor S is smaller than required. Therefore, our first run is:

```
P4/m:  {From/To}LinearStepCount:4:10:5{From/To}
```

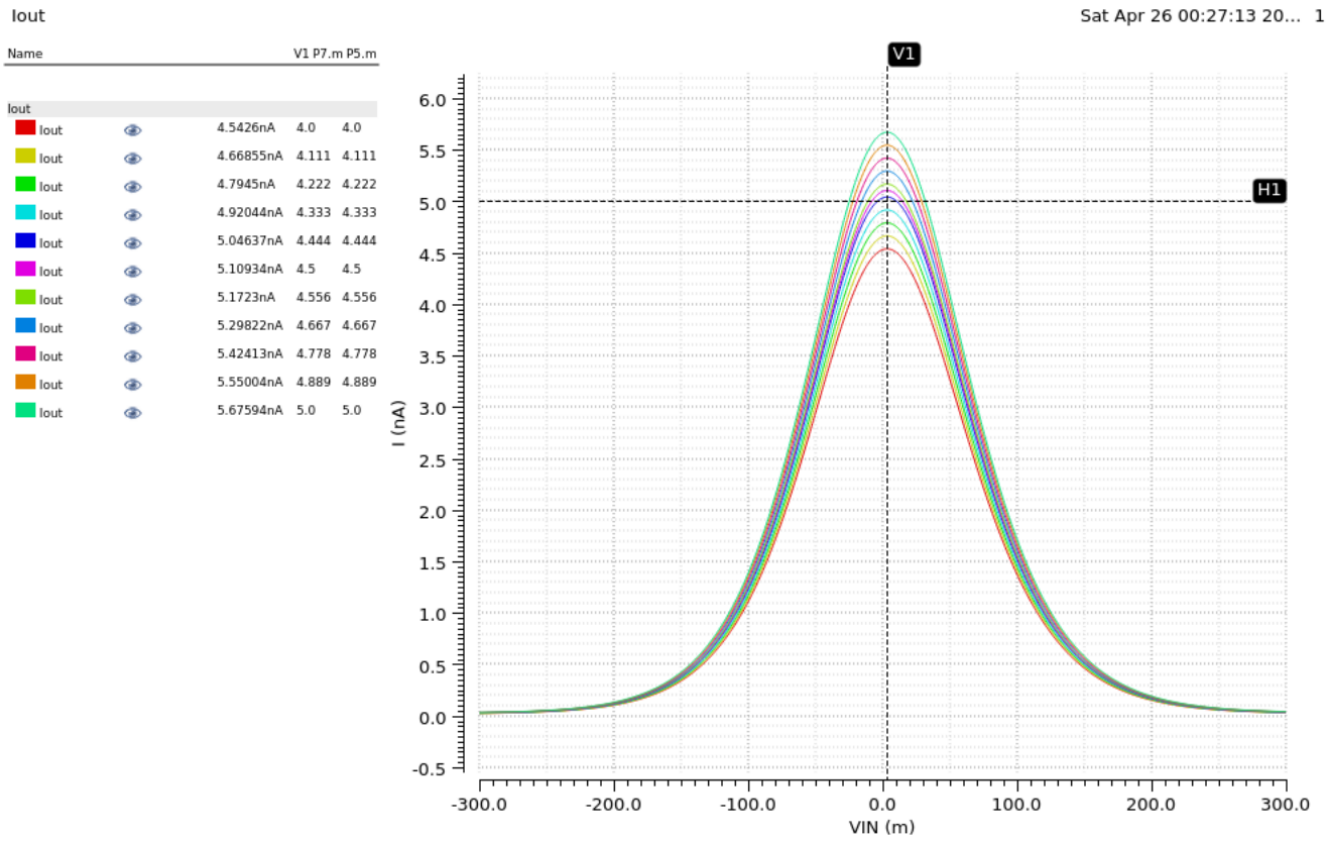


Figure 2: Iout for sweeping m

The optimal values appear to lie between 4.2 and 4.5, with only a small offset remaining. The exact choice will ultimately depend on the available chip area and whether it permits achieving the desired ratio. Other considerations include power consumption, noise, and the deviation from the target I_{bias}.

1 May

Parametric Simulations

Today we will be running multiple parametric simlations in order to find the optimal values for the parameters of the Delbruck bump.

First of all, we will run a simulation to for the Length. As far as matching goes:

- M_{p1} is matched to M_{p3} as parts of a current mirror
- M_{p2} is matched to M_{p4} as parts of a current mirror
- M_{p3} is matched to M_{p4} as shown in equation 5 of Logbook - 23 April
- M_{p1} is matched to M_{p2} as shown in equation 5 of Logbook - 23 April
- M_{n1} is matched to M_{n2} as parts of a differential pair
- M_{n3} is matched to M_{n4} as parts of a current mirror

First we sweep the length of the PMOS transistors. The values are:

From/ToLinearStepCount:100n:10:1uFrom/To

With the results presented in figure 1:

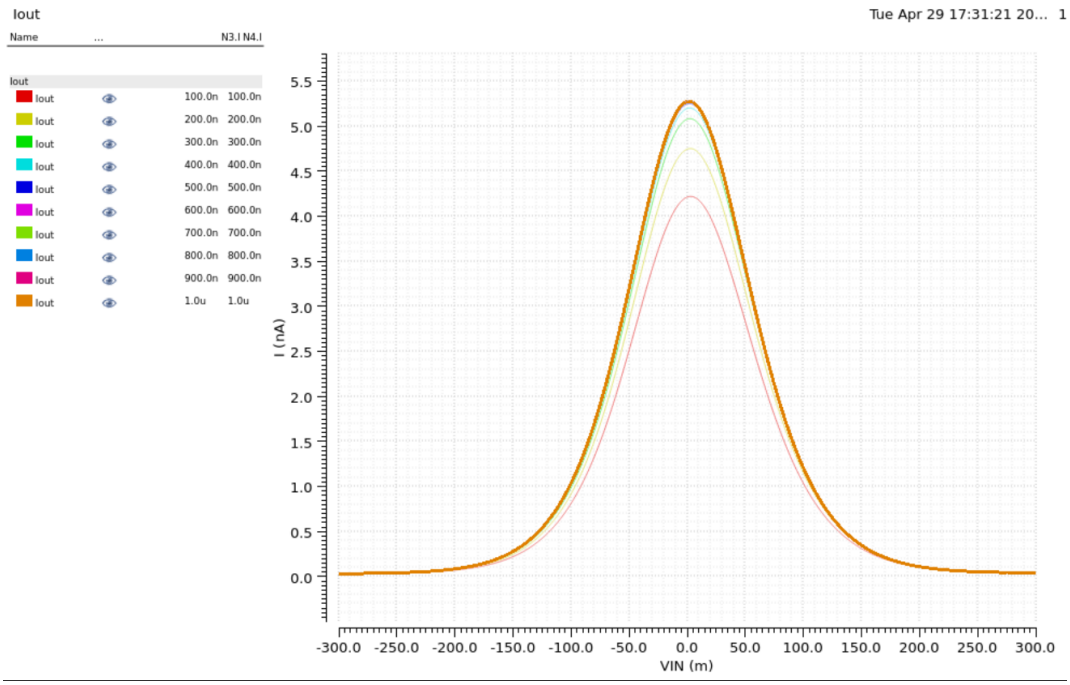


Figure 1: PMOS Length Sweep

It is clear that the optimal length lies somewhere close to 300nm. For lengths smaller than that we have a very low output current, while for lengths larger we have diminishing returns.

Now we will sweep the length of all transistors. The values are: 250n and 350n both for the PMOS and the NMOS transistors.

The results are presented in figure 2:

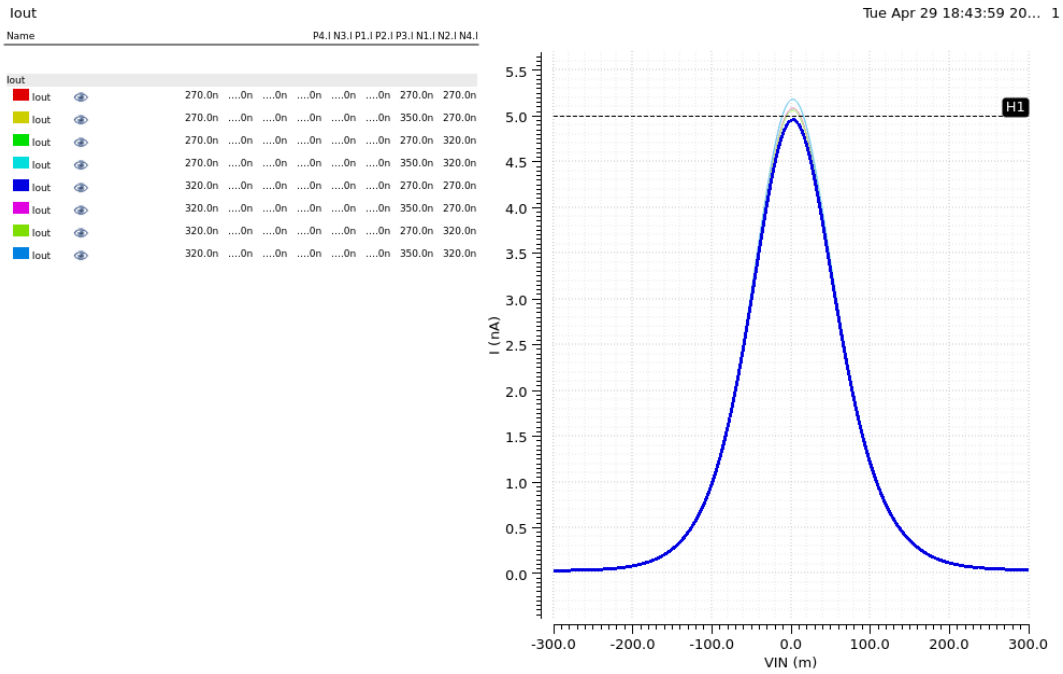


Figure 2: PMOS and NMOS Length Sweep

By sweeping once more for 300nm for all transistors we get the following output:

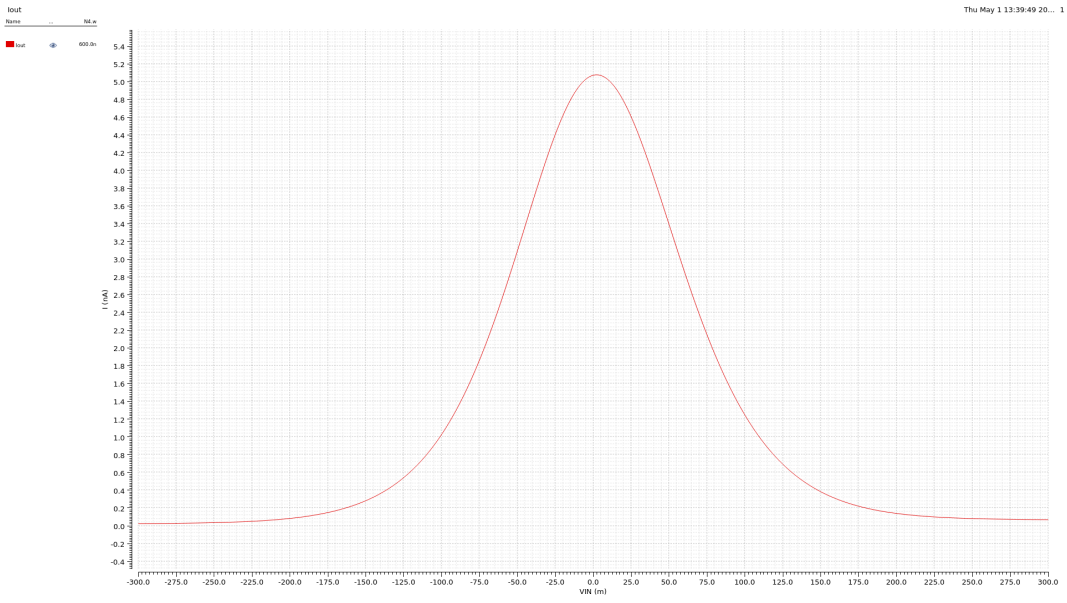


Figure 3: PMOS and NMOS Length Sweep 300nm

Therefore, we match all transistors to 300nm, which will also help with layout and process variation.

By sweeping the width of the correlator and the differential block for the values: 0.2u 0.4u 0.6u (for both PMOS and NMOS) we get the following results:

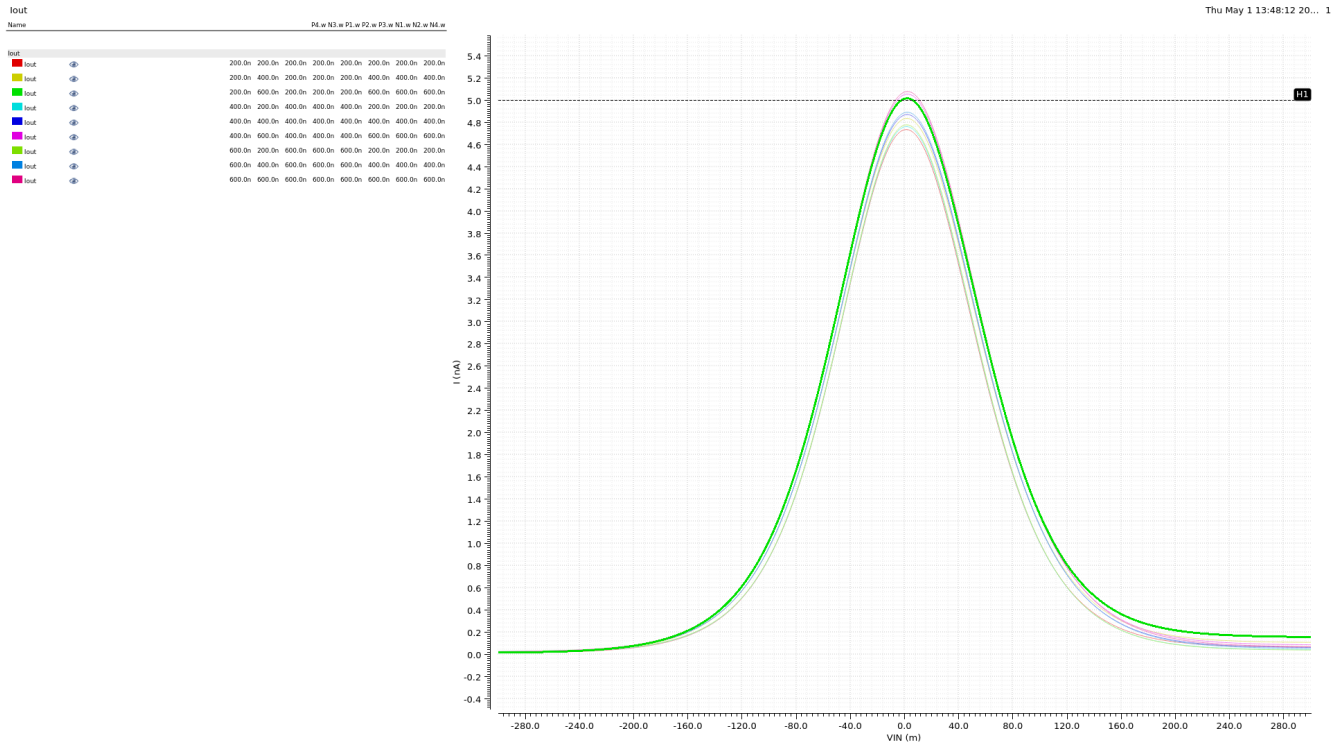


Figure 4: Length Sweep

Therefore, the values we choose are presented in Table 1.

Block	**W/L (um/um)**	**Current Correlator**	**W/L (um/um)**
$M_{n1} - M_{n4}$	0.6/0.3	$M_{p1} - M_{p4}$	0.2/0.3

Table 1: Transistor Dimensions

Now we present the results of our current design: Our default values, unless being swept, are:

Parameter	**Default Value**
I_{bias}	5 nA
V_{mean}	0 V

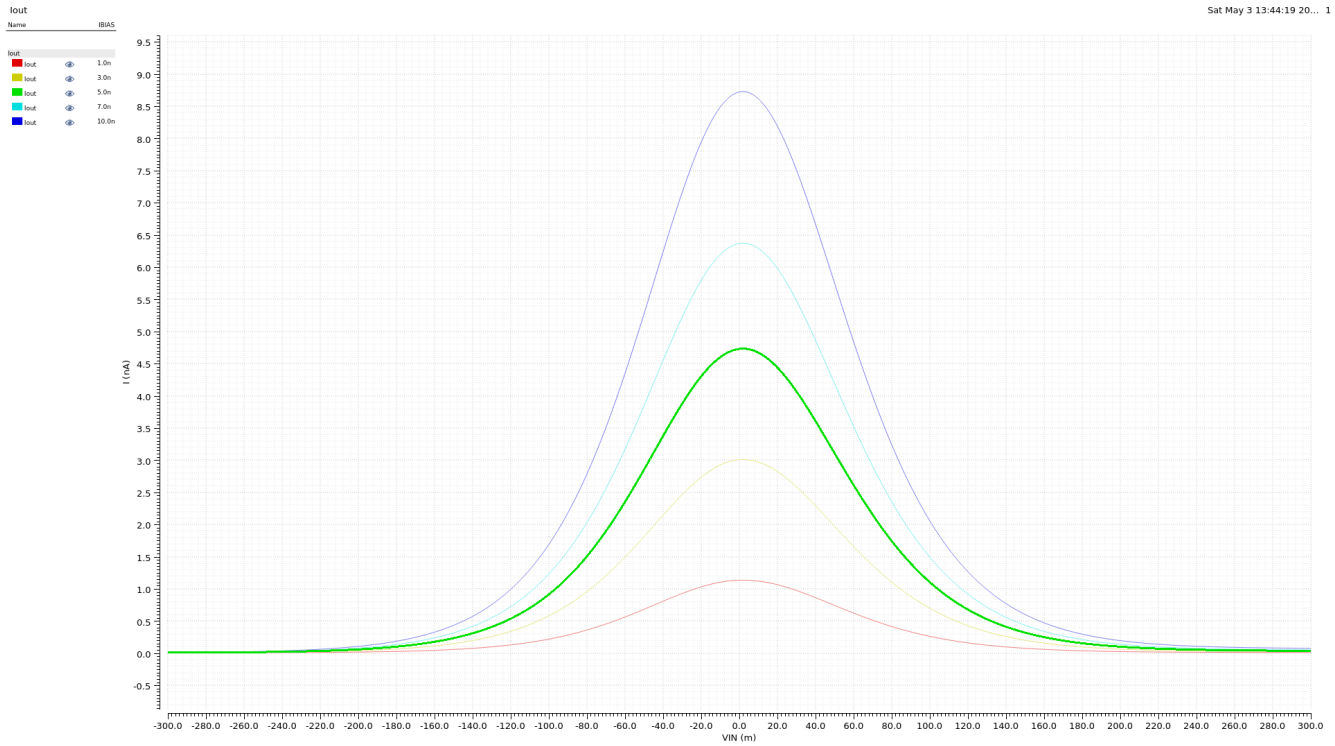


Figure 5: Ibias Sweep

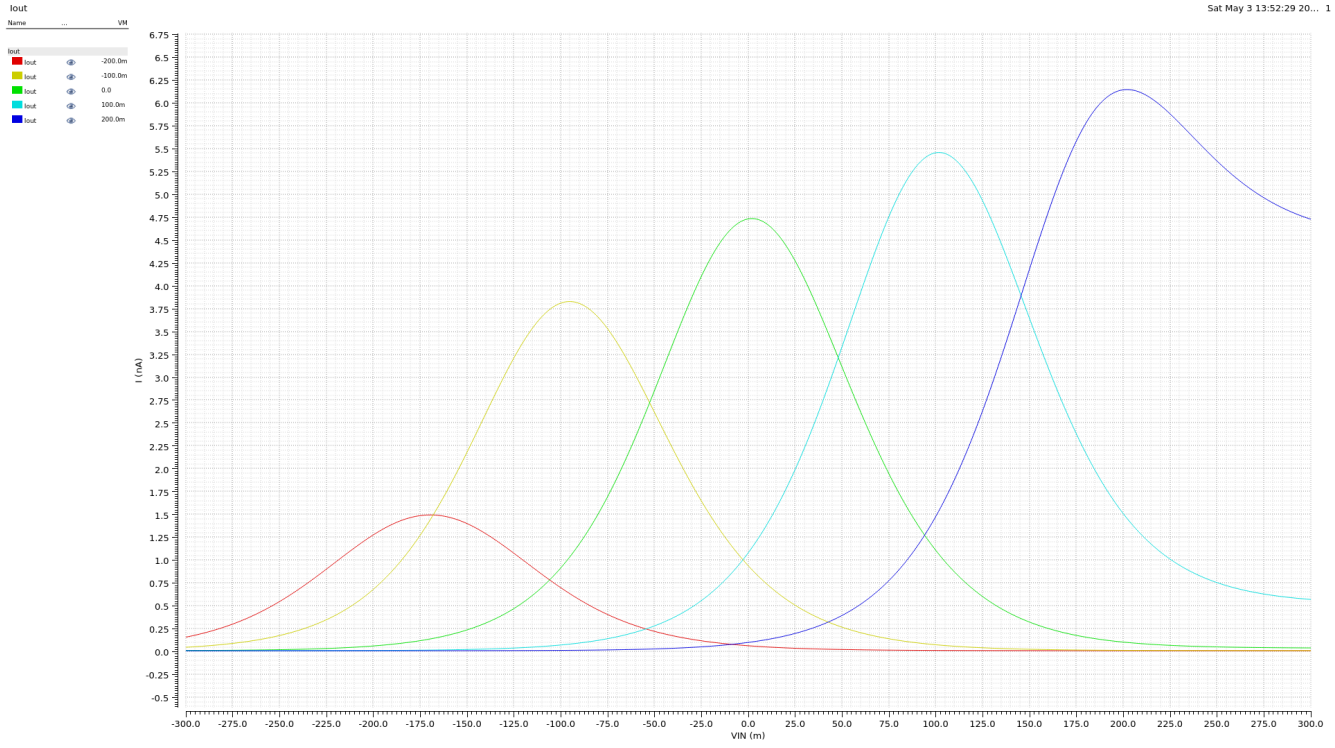


Figure 6: Mean Value Sweep

And these are the results for noise, power and PSRR.

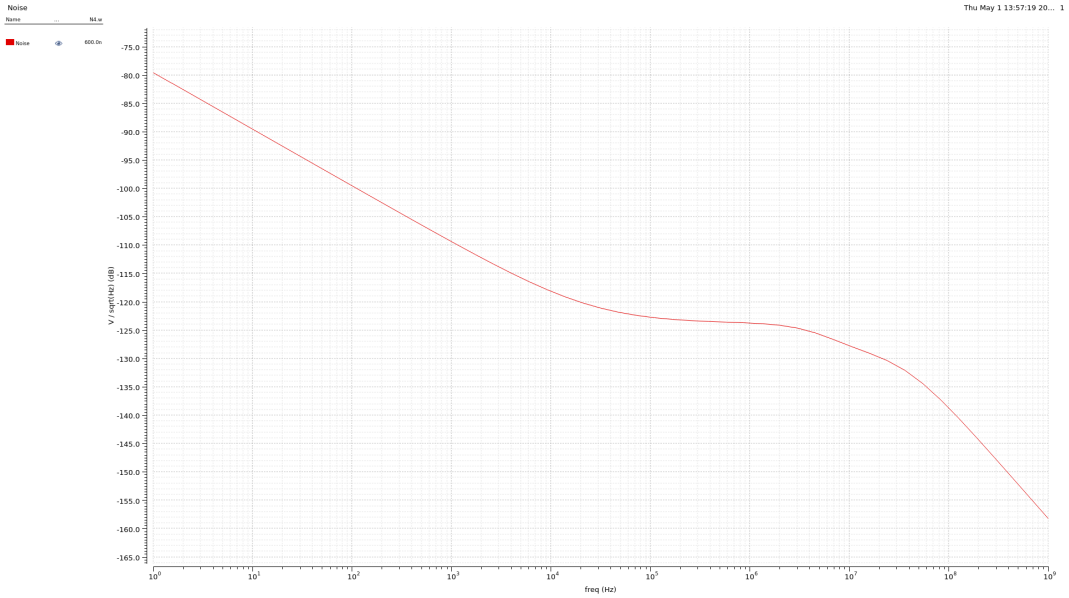


Figure 7: Noise

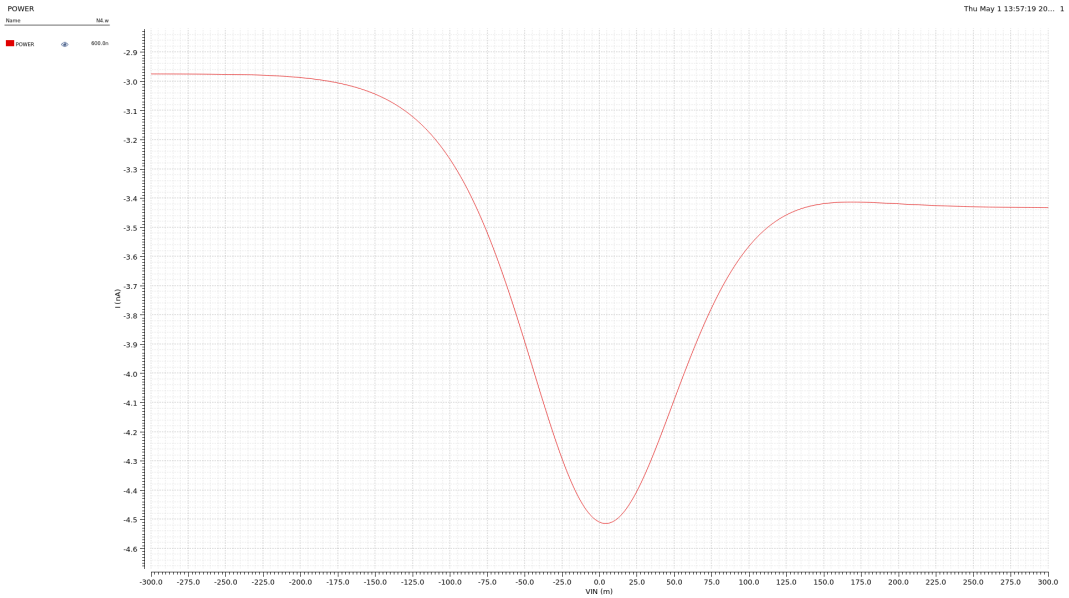


Figure 8: Power

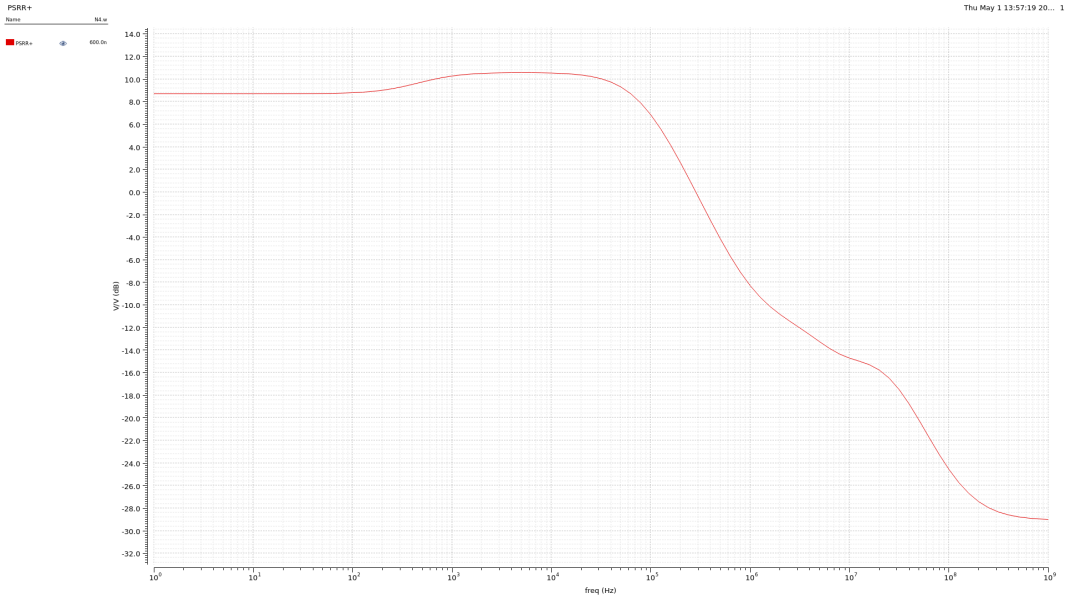


Figure 9: PSRR+

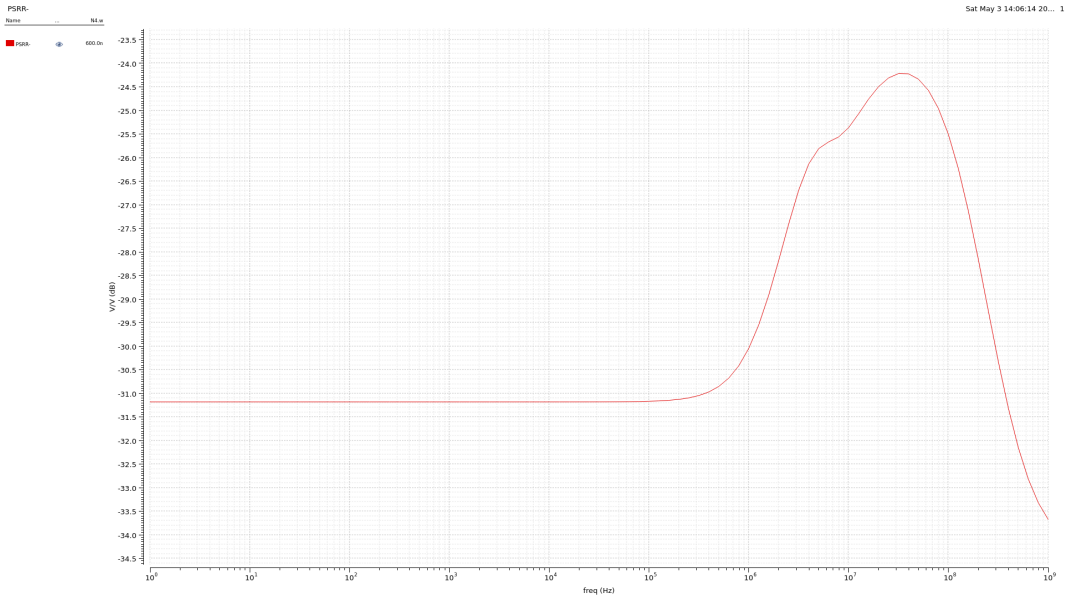


Figure 10: PSRR-

We modify our cirtcuit and our testbench in order to include a body voltage variable V_c for the differential pair.

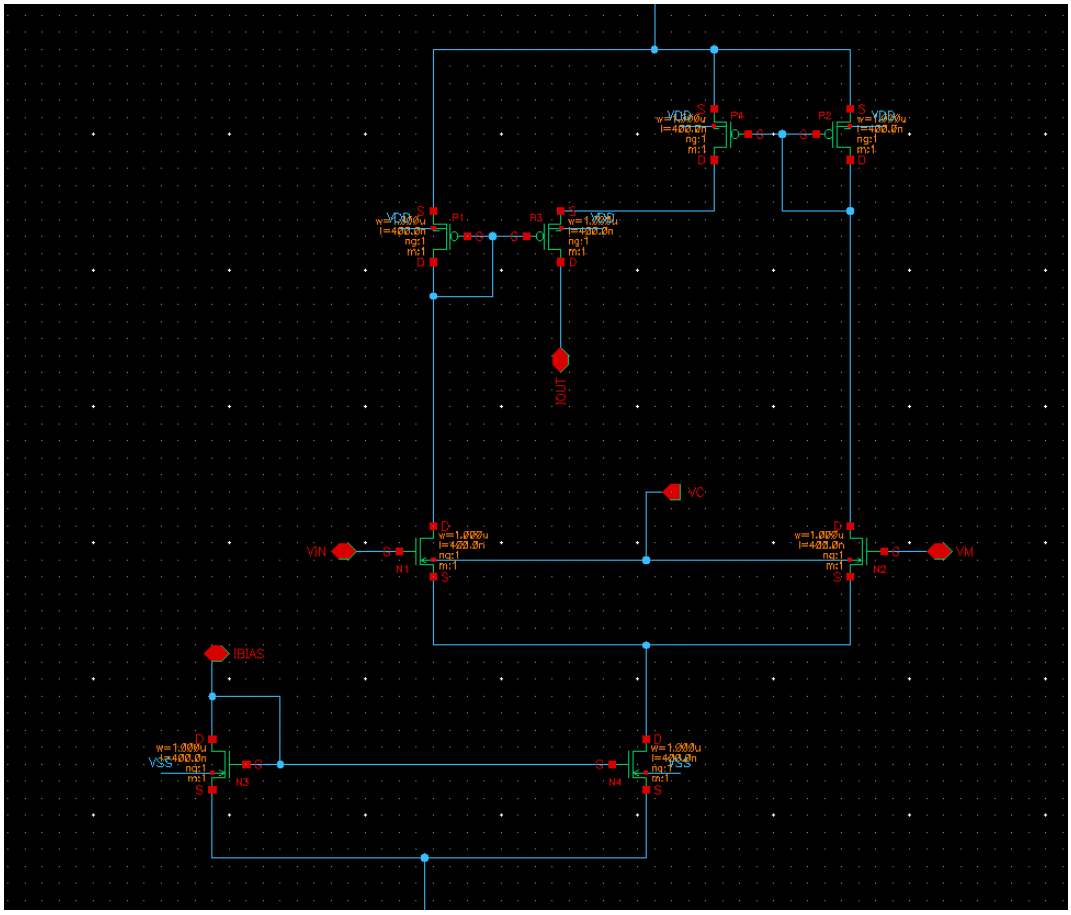


Figure 11: Transistor Schematic

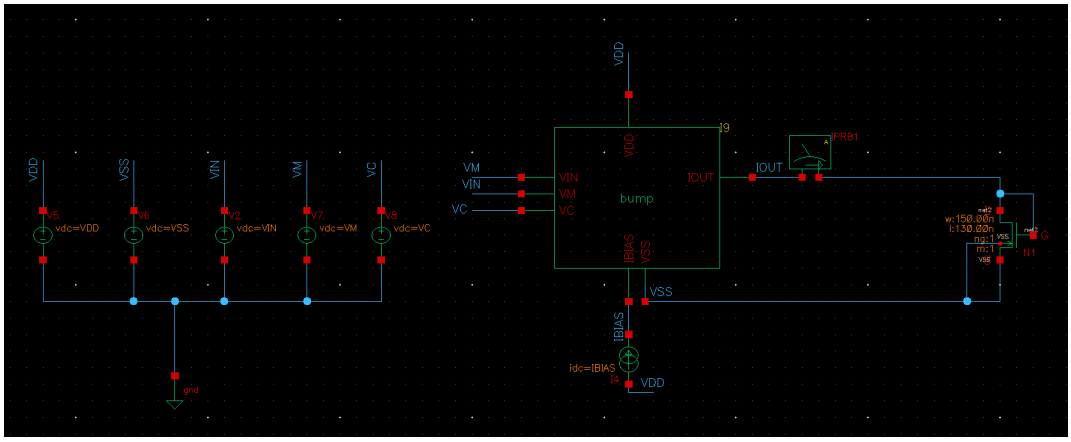


Figure 12: Testbench

We then sweep the body voltage for the values: -0.3V, -0.15V, 0V, 0.15V, 0.3V

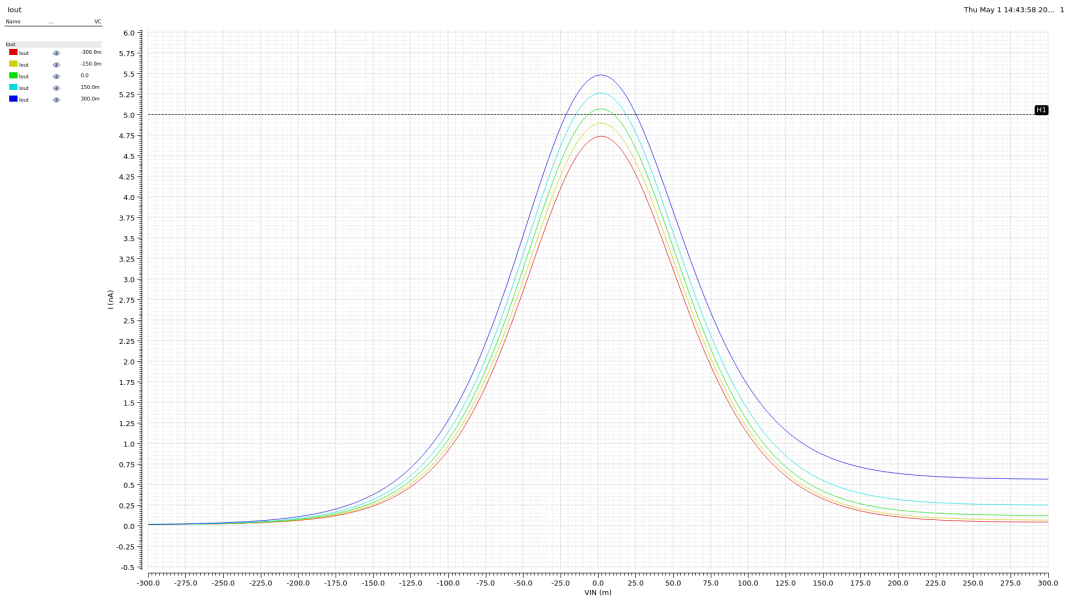


Figure 13: Body Voltage Sweep

We notice that we can actually tune the width of the bell curve by changing the body voltage.

7 May

Entry

In our previous entry, we noticed that the circuit practically stopped working when lowering the mean voltage (V_{mean}) below 0V. As shown in the figure below, the output current was significantly reduced compared to larger V_{mean} values. Therefore, in this entry we aim to desensitize the circuit to V_{mean}, in order to achieve a, relatively, uniform output current peak across a range of V_{mean} values.

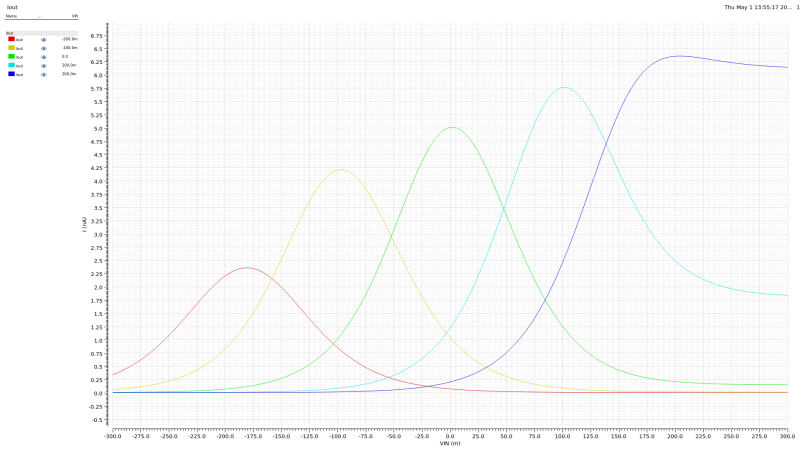


Figure 1: Output current vs V_{mean}

Our first approach is to investigate the effect of the bulk voltage (V_C) on the output current. The previous figures were obtained with $V_C = V_{SS} = -0.3V$ By sweeping V_C from -0.3V to 0.3V, we get the following results:

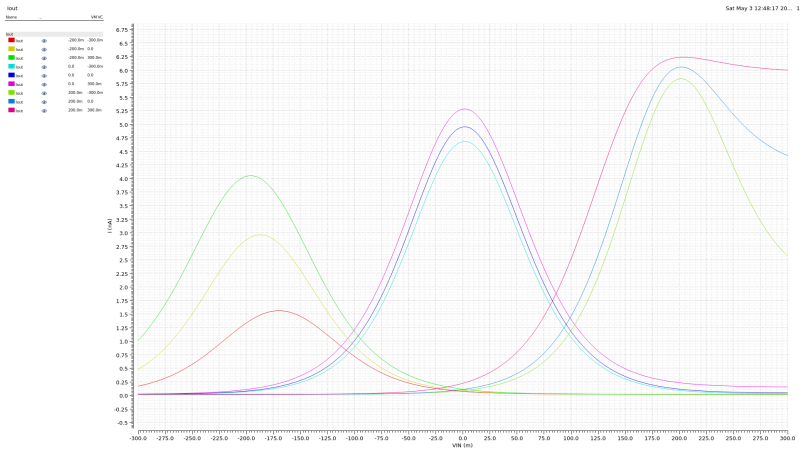


Figure 2: Output current vs V_C

It is clear that we get significantly improved results by increacing the value of V_C. The cause of this is the body effect, which lowers the threshold voltage of the NMOS transistors. However, our goal is to have three independent parameters to control the output current. Therefore, it is not ideal to have V_C and V_{mean} coupled.

Moving on, we can try modifying the size of the transistors. Sweeping the differential pair:

- Length from 0.3um to 2um
- Width from 0.2um to 2um

Sweeping the length we find:

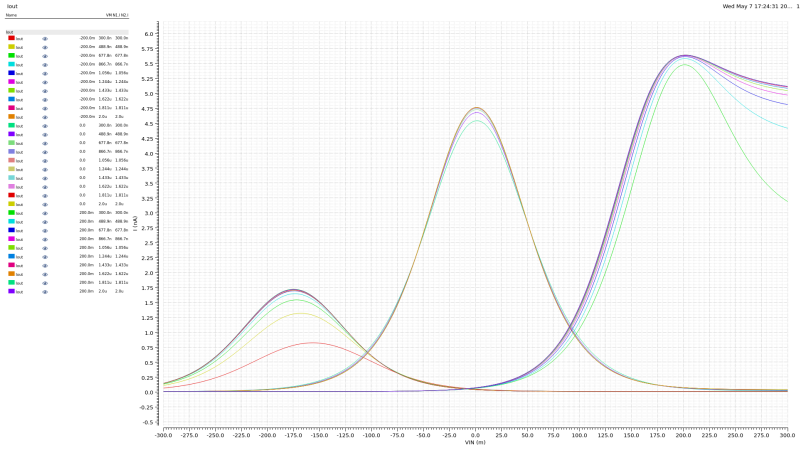


Figure 3: Output current vs transistor length

Clearly, not much is happening here. Sweeping the width we find:

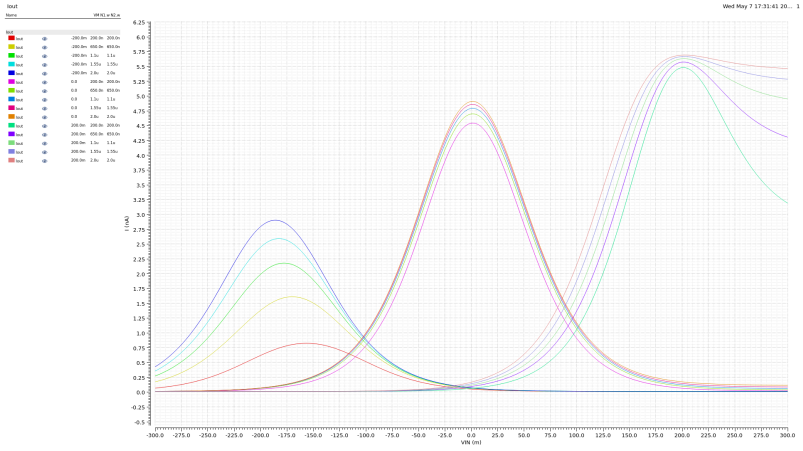


Figure 4: Output current vs transistor width

We notice a somewhat inverse relationship between the current at low V_{mean} (which improves its peak

significantly with increasing the widht) and the current at high V_{mean} (which gets more assymetric with increasing the width). We choose a width of 1um and further sweep the multipliers of the width.

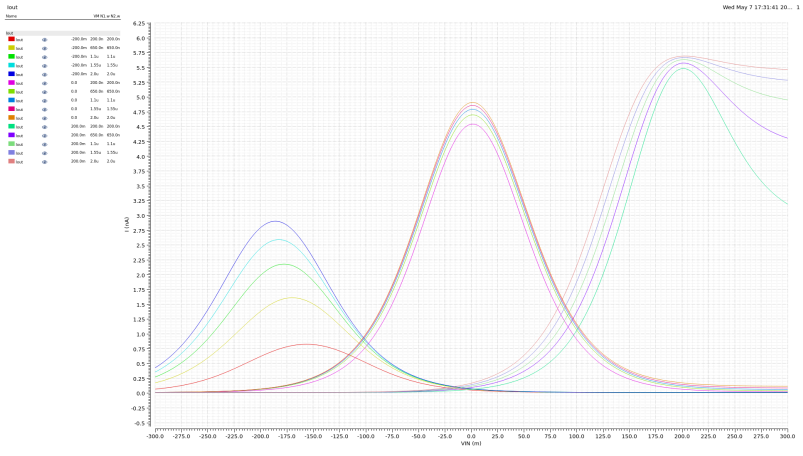


Figure 5: Output current vs transistor width multiplier

We can see that the current at low V_{mean} needs a multiplier of at least 20 to even be close to the peak we desire, at which point the current at high V_{mean} is completely derailed.

At this poitn we can start modifying our design. For starters, we can replace the current mirror with an ideal current source, by connecting the Ibias pin directly to the source of the differential pair (obviously we reverse the polarity of the bias current as well).

Figure 6: Ideal current source

A significant improvement is observed, but the current at high V_{mean} is still not ideal.

For the values: length = 0.2um width = 0.2um

Figure 7: Output current

we get very good output current throughout the range of V_{mean}.

12 July

Improving Current Source

In the previous experiments, we saw that having a reliable current source was crucial for consistent results. The output of the current mirror connects to the common source node of the input differential pair. The voltage at this node is not static; it fluctuates in response to the common-mode level of the input signals (V_{in} and V_{mean}). These voltage fluctuations at the mirror's output cause the mirrored current, I_{bias}, to vary due to the channel length modulation effect. An unstable tail current leads directly to an unstable and signal-dependent amplitude of the Gaussian output curve. Therefore, we aim to improve the current source used so far.

Our first step is to replace the basic current mirror with a **cascode current mirror**, as shown in Figure 1. The cascoded configuration provides a significantly higher output resistance, which is a fundamental characteristic of an ideal current source.

The output resistances of the two configurations are:

- **Basic Current Mirror output resistance:**

$$r_{\text{out}} \approx r_{ds}$$

- **Cascoded Current Mirror output resistance:**

$$r_{\text{out}} \approx g_m \cdot r_{ds}^2$$

This shows that the cascoded current mirror achieves a much higher output resistance by effectively multiplying the transistor parameters, resulting in a more stable and accurate current source.

The most effective method to increase r_{ds} is to increase the transistor's channel length, as r_{ds} is roughly proportional to L. Therefore, the channel length of all transistors comprising the cascode current mirror should be made significantly larger than the minimum feature size of the technology. Additionally, since the currents flowing through the transistors are very small, we will choose a small channel width (W).

We match all parameters of all the transistors.

We perform a sweep

- **W**: From `0.1` μm to `2` μm, auto step `10`
- **L**: From `0.5` μm to `4` μm, auto step `10`

and we choose these parameters:

- $\mathbf{W/L} = 0.2\,\mu\text{m}/2\,\mu\text{m}$

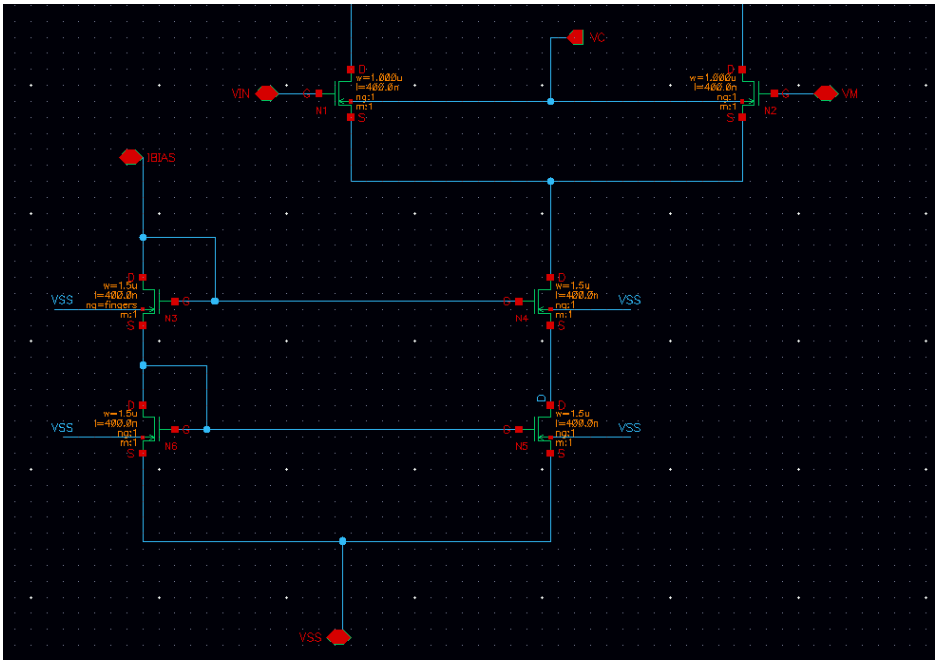


Figure 1: Cascode Current Mirror

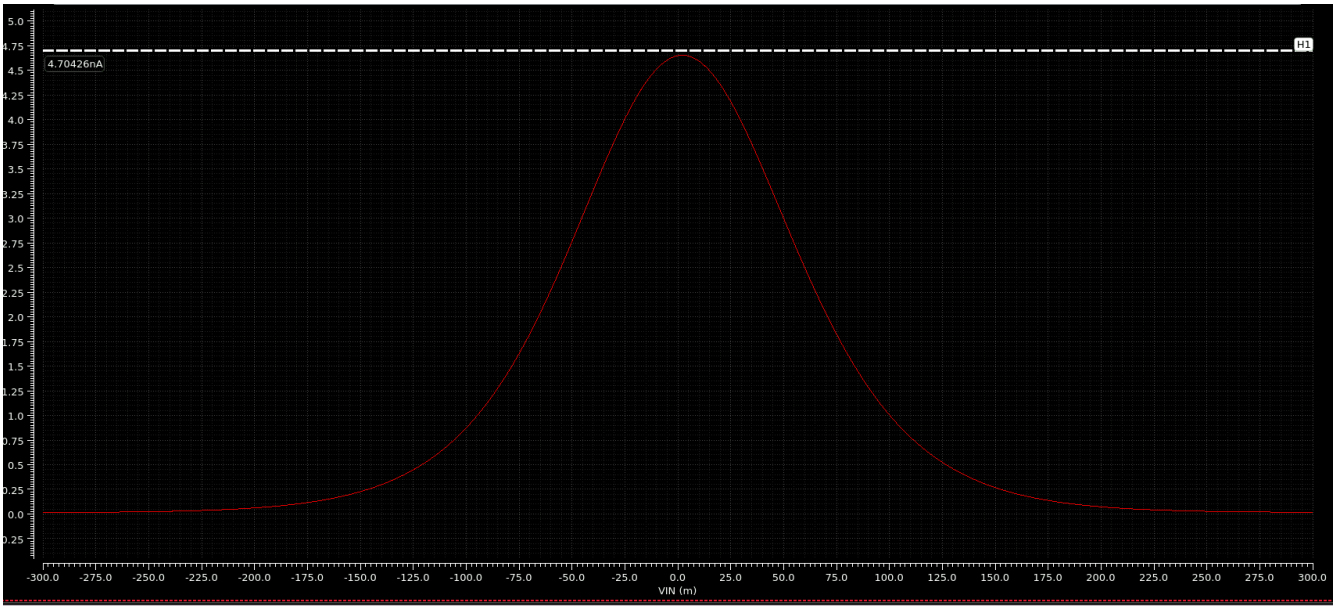


Figure 2: Cascode Current Mirror IOUT

Next, we will test the effect of multipliers and fingers. Virtuoso doesn't allow to pass the amount of gates neither as a parameter, nor as a design variable. Therefore, we will manually change it in the schmematic, performing a multiplier sweep each time: From/ToAuto:1:10:20From/To.

- **Number of Gates:** 1
  - **IOUT:**

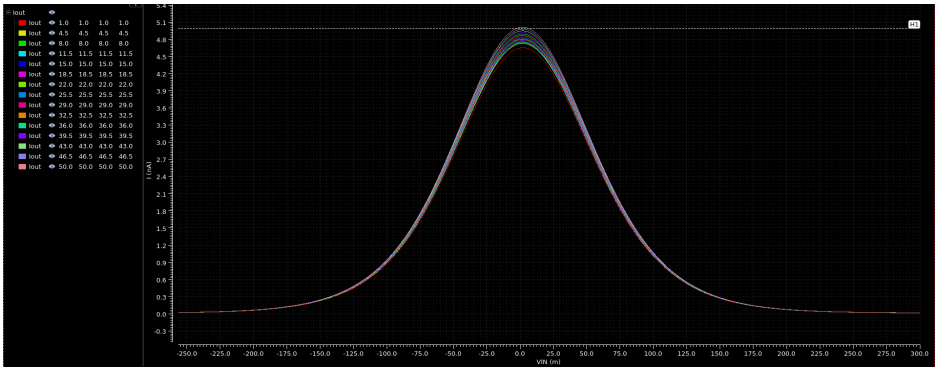


Figure 3: Cascode Current Mirror IOUT with 1 Gate

  - **Noise:**

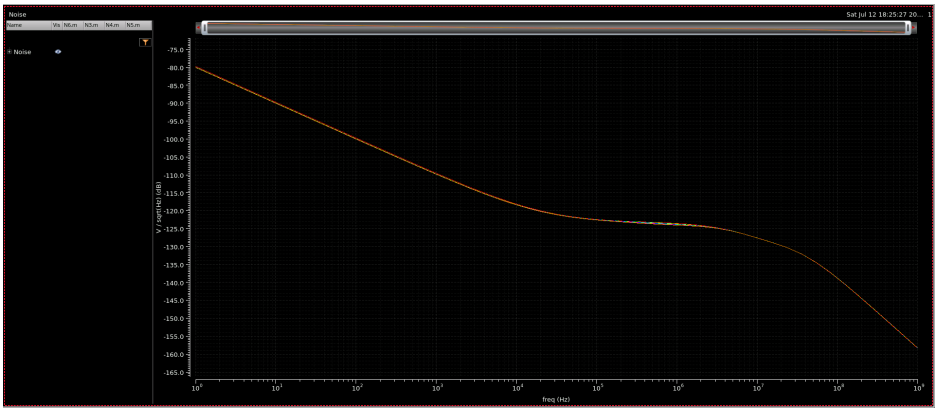


Figure 4: Cascode Current Mirror Noise with 1 Gate

  - **PSRR-:**

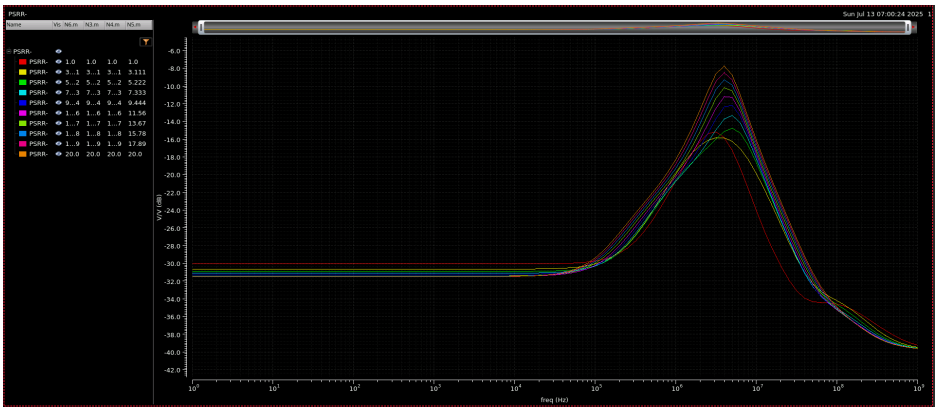


Figure 5: Cascode Current Mirror PSRR- with 1 Gate

- **Number of Gates:** 2
  - **IOUT:**

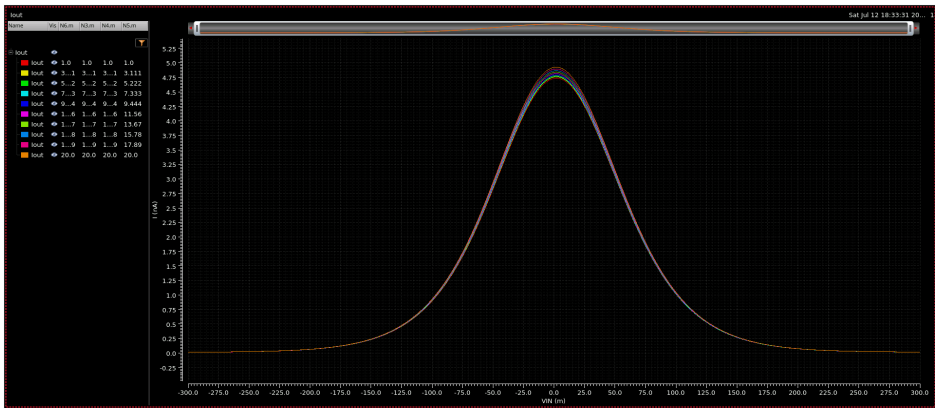


Figure 6: Cascode Current Mirror IOUT with 2 Gates

  - **Noise:**

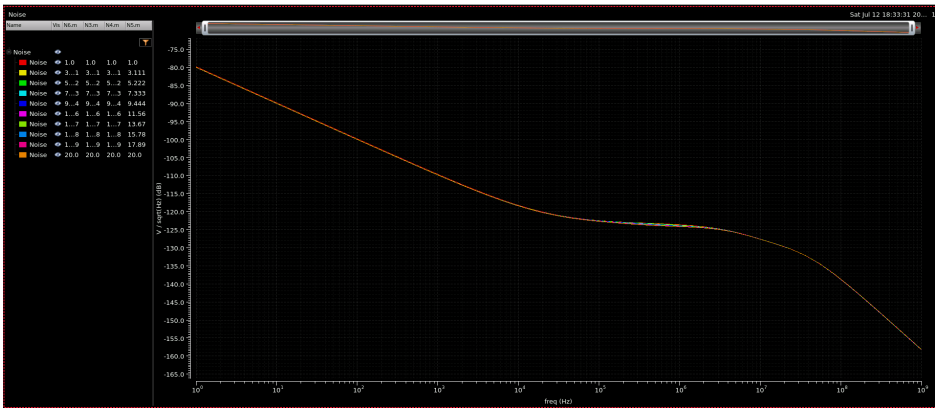


Figure 7: Cascode Current Mirror Noise with 2 Gates

  - **PSRR-:**

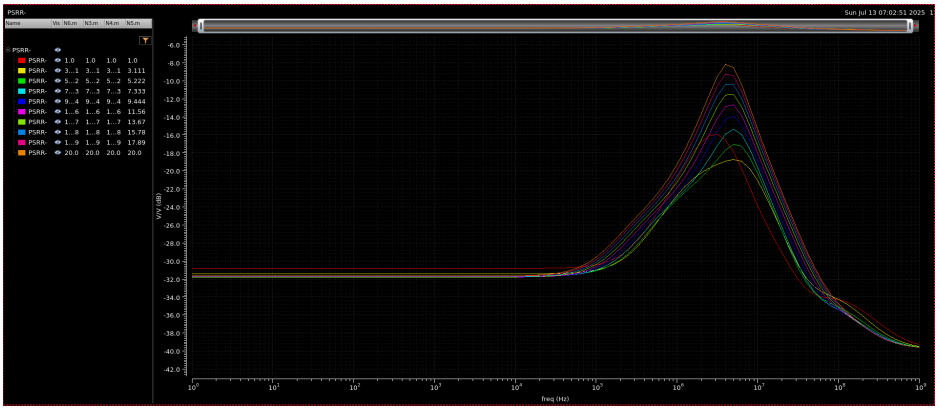


Figure 8: Cascode Current Mirror PSRR- with 2 Gates

- **Number of Gates:** 4
  - **IOUT:**

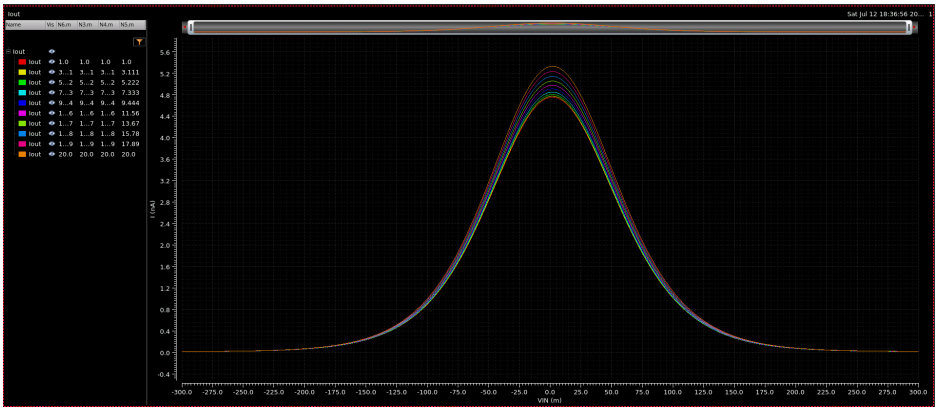


Figure 9: Cascode Current Mirror IOUT with 4 Gates

  - **Noise:**

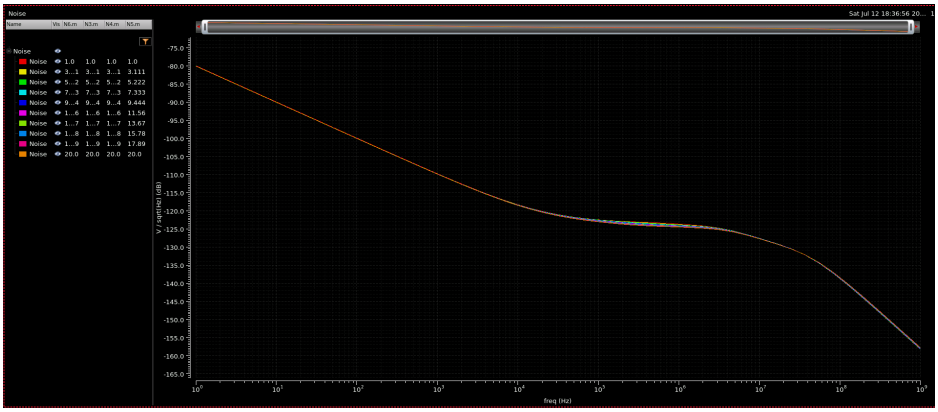


Figure 10: Cascode Current Mirror Noise with 4 Gates

  - **PSRR-:**

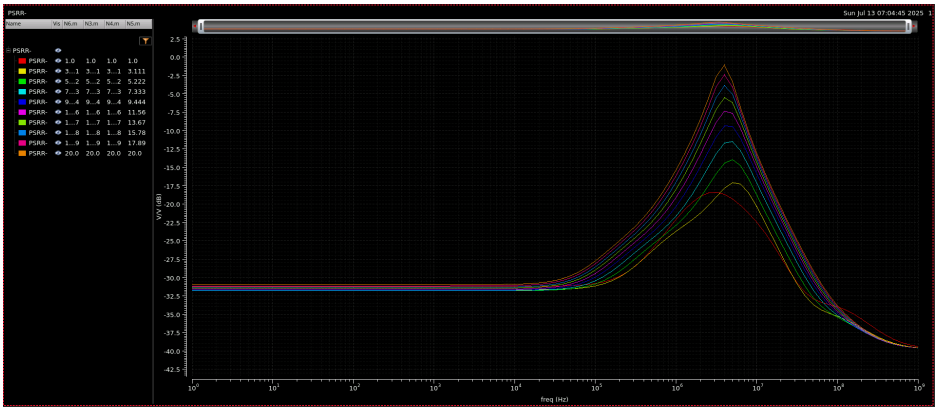


Figure 11: Cascode Current Mirror PSRR- with 4 Gates

- **Number of Gates:** 8
  - **IOUT:**

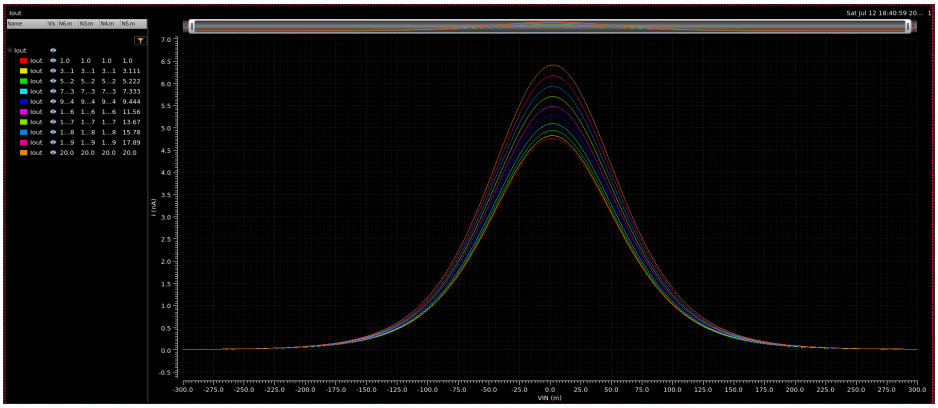


Figure 12: Cascode Current Mirror IOUT with 8 Gates

  - **Noise:**

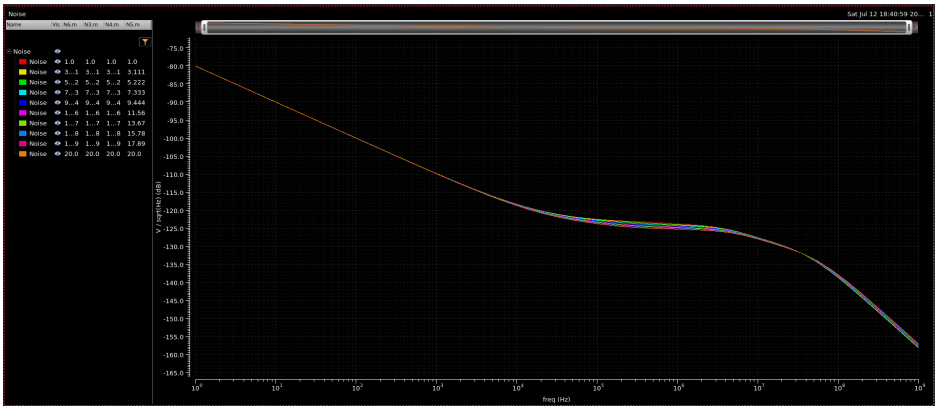


Figure 13: Cascode Current Mirror Noise with 8 Gates

  - **PSRR-:**

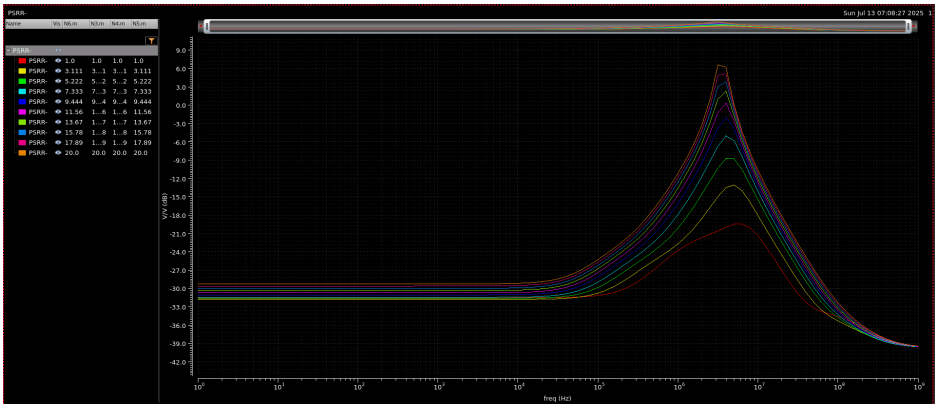


Figure 14: Cascode Current Mirror PSRR- with 8 Gates

We notice that, besides the output current, the noise and PSRR- (also Power which is not shown here) remain basically unchanged. Therefore, we choose **multipliers = 6 fingers = 8** for all transistors in the cascode current mirror.
An alternative current source could be the low-voltage self-biased current mirror, shown below.

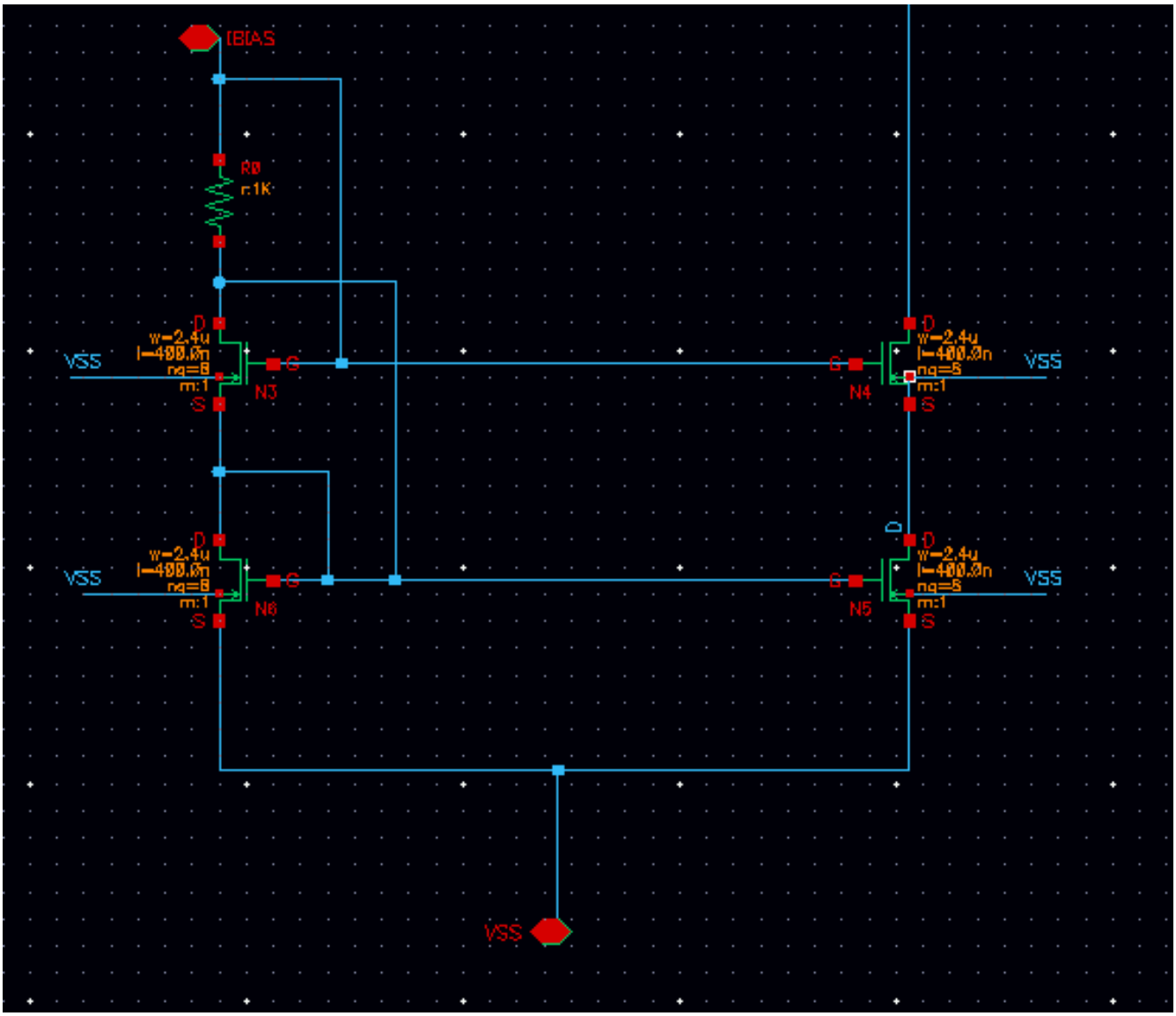


Figure 15: Self-biased Current Mirror

However, after multiple parametric sweeps, no noticeable improvement was observed compared to the cascode current mirror.

Another alternative configuration is the one shown below, which utilizes feedback with an ideal opamp in order to achieve enhanced output resistance.

The output resistance of this configuration is:

$$r_{\text{out}} \approx g_{m17} \cdot r_{ds17} \cdot r_{ds5} \cdot (A + 1)$$

where A is the gain of the operational amplifier.

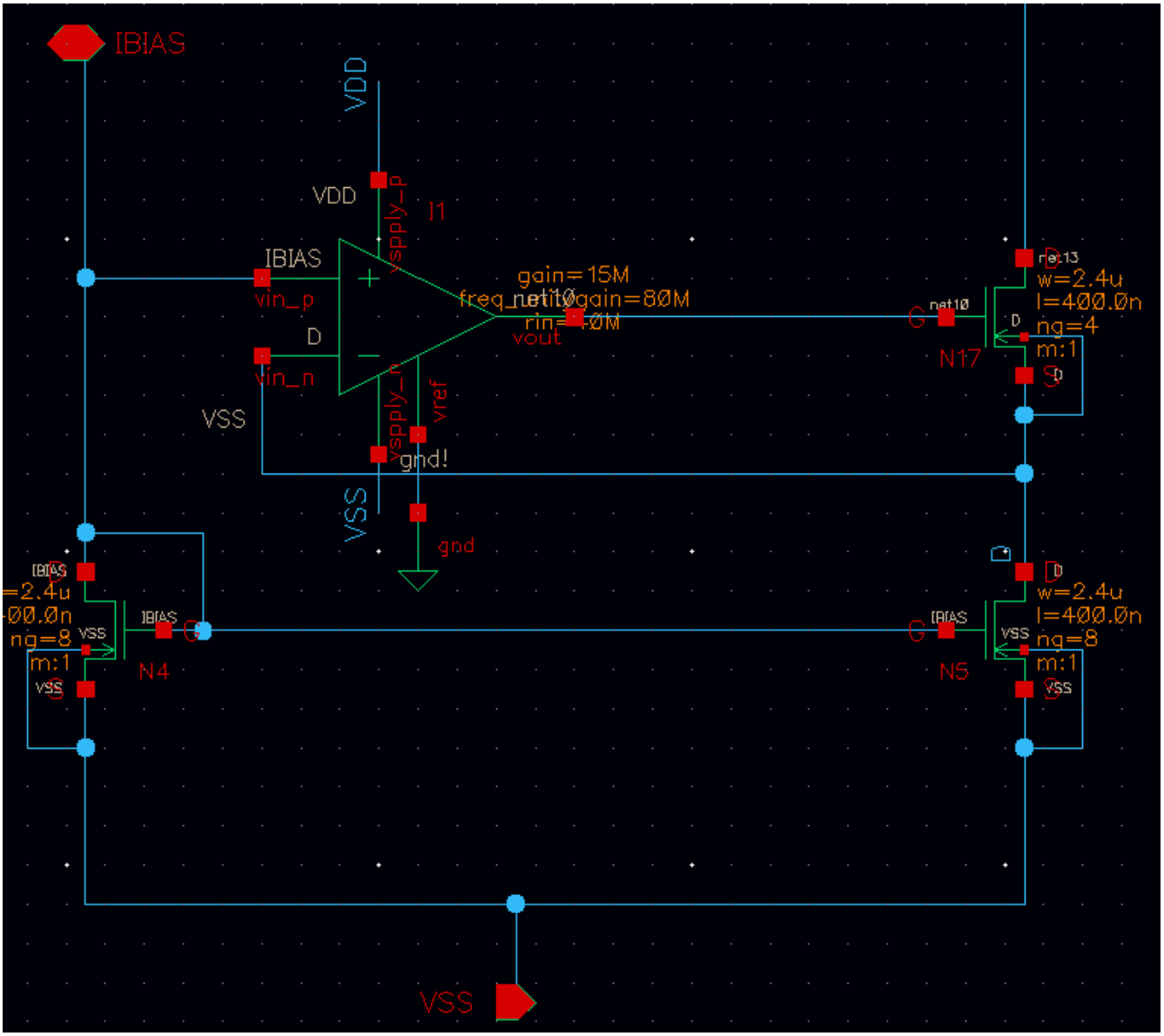


Figure 16: Feedback Current Mirror

With the ahdLib opamp with the following parameters:

Parameter	**Value**
Gain (A)	15M
Unity-Gain Bandwidth ($f_{\text{unitygain}}$)	80M
Input Resistance (r_{in})	40M
Input Offset Voltage ($v_{in,\text{offset}}$)	10µ
Bias Current (i_{bias})	1n
Slew Rate	80M
Output Resistance (r_{out})	70

Table 1: Opamp parameters used in the enhanced current mirror configuration.

Which, theoretically, should provide us with infinite output resistance. Throught multiple parametric sweeps, we keep these parameters:

Device(s)	**W/L (μm/μm)**
P_1–P_4	2.0/0.3
N_1–N_2	1.0/1.0
N_3–N_6	1.5/2.5
N_{17}	2.5/2.5

Table 2: Transistor Dimensions after Parameter Substitution.

As far as fingers are concerned, all transistors are set to 8, except for N_{17} which is set to 4.
We get the following result for a V_{mean} sweep:

Figure 17: Feedback Current Mirror Output Current

16 July

Further Improvements

We move on from the previous correlator block and replace it with a new symmetric one.

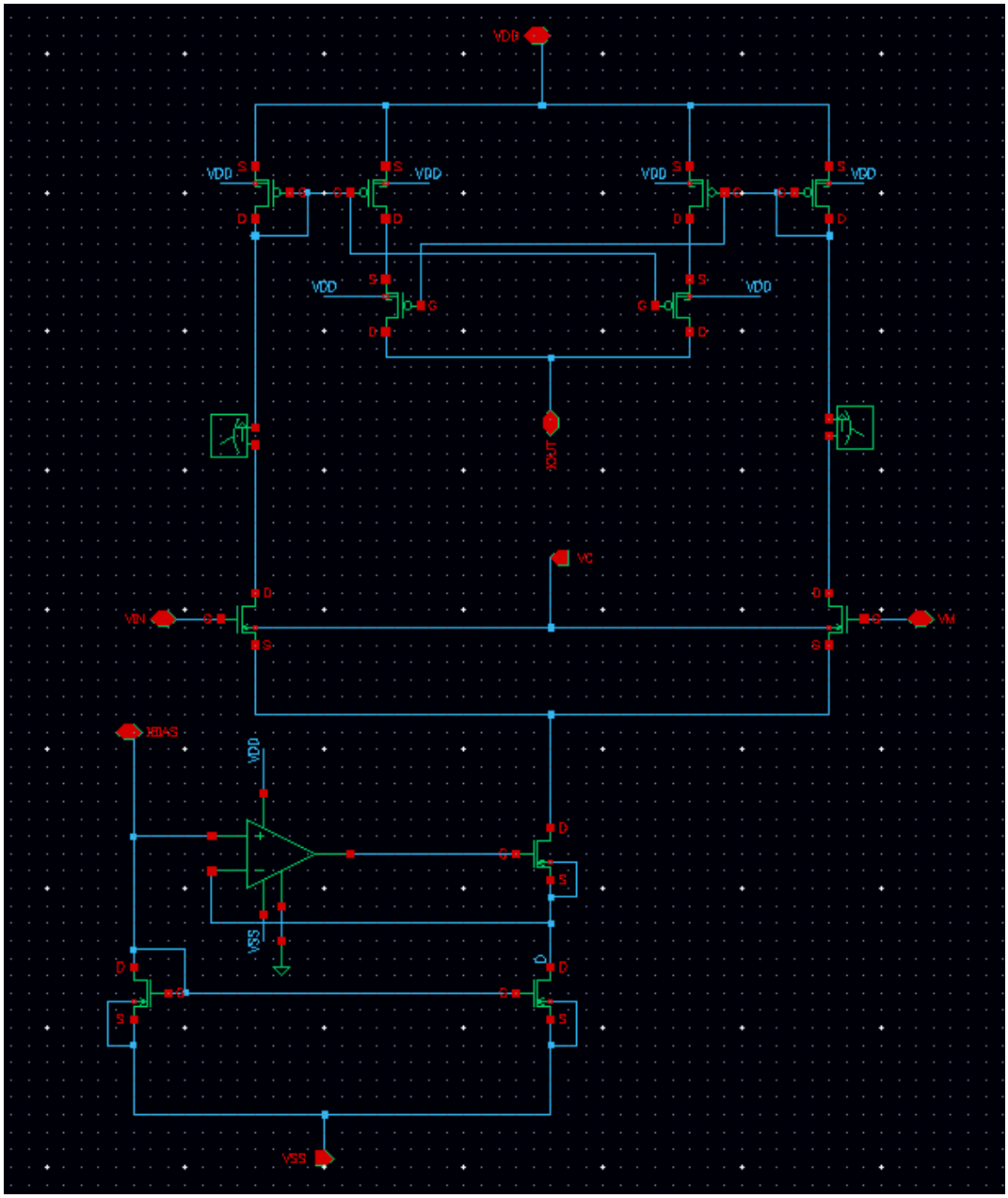


Figure 1: Symmetric Current Correlator

Symmetric Current Correlator Analysis

The original "Simple Bump" circuit utilized a four-transistor, non-symmetric current correlator. However, this topology is structurally asymmetric and, as noted in the literature, suffers from "inherent asymmetries in the output current". These asymmetries arise from the unavoidable mismatches in transistor characteristics (such as threshold voltage and current factor). Furthermore, if any of the transistors operate above the sub-threshold region, the correlation function is no longer commutative (i.e., $f(I_1, I_2) \neq f(I_2, I_1)$), which further exacerbates the asymmetry and complicates the mathematical model.

The symmetric current correlator is composed of two classic current correlator sub-circuits: one formed by transistors M_{p1}, M_{p2}, M_{p3}, and M_{p7}, and the other by transistors M_{p1}, M_{p2}, M_{p4}, and M_{p8}. These sub-circuits generate two equal currents, satisfying $I_{M_{p7}} = I_{M_{p8}}$. The drain current of transistor M_{p7}, which operates in the saturation region, is expressed as follows:

$$I_{out} = I_{M_{p7}} = I_{op} e^{((\kappa_p - 1)V_{DD} - \kappa_p V_{D_{Mp2}} + V_{D_{Mp3}})/V_T} \tag{1}$$

where $V_{D_{Mp2}}$ and $V_{D_{Mp3}}$ are the drain voltages of transistors M_{p1} and M_{p3} respectively. Transistor's M_{p1} current is given by:

$$I_1 = I_{M_{p1}} = I_{op} e^{\kappa_p (V_{DD} - V_{D_{Mp1}})/V_T} \tag{2}$$

where $V_{D_{Mp1}}$ is the drain voltage of transistor M_{p1}. Transistor's M_{p2} current is given by:

$$I_2 = I_{M_{p2}} = I_{op} e^{\kappa_p (V_{DD} - V_{D_{Mp2}})/V_T} \tag{3}$$

Both transistors M_{p3} and M_{p7} have the same current. Transistor M_{p3} operates in triode region. The drain current for transistor M_{p3} is given by:

$$I_{M_{p3}} = I_{op} e^{\kappa_p (V_{DD} - V_{D_{Mp1}})/V_T} \left(1 - e^{(V_{D_{Mp3}} - V_{DD})/V_T}\right) \tag{4}$$

Combining (1)–(4) and using $I_{M_{p7}} = I_{M_{p8}}$ (symmetric current correlator implementation) we conclude:

$$I_{out} = 2\frac{I_1 I_2}{I_1 + I_2} \tag{5}$$

We run the following parametric simulation:

- P_1/m: 5, 10, 15
- N_1/l: 0.2μ, 0.5μ, 1μ
- N_1/m: 5, 10, 15
- N_1/w: 0.2μ, 0.5μ, 1μ

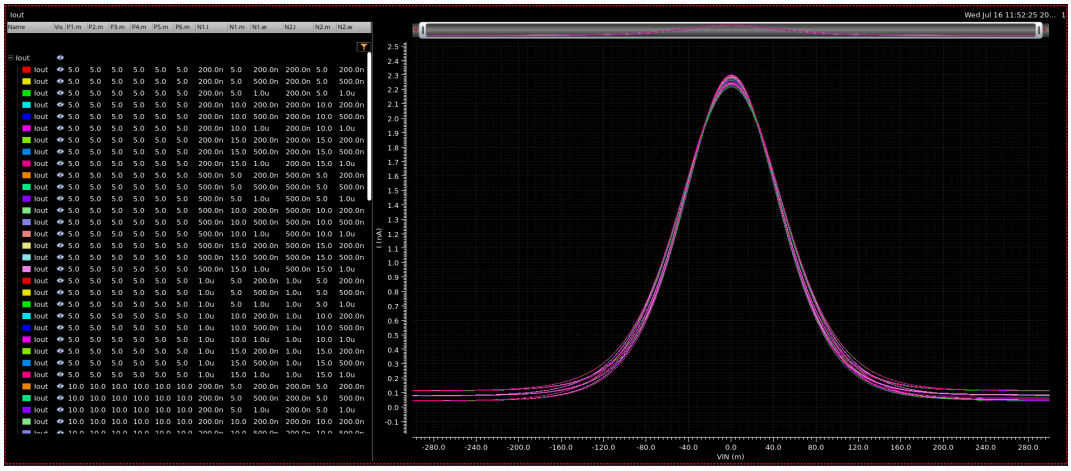


Figure 2: Symmetric Current Correlator Parametric Simulation

From the graph we can see that no matter the parameters, th output current's peak is practically the same. Although some combinations achieve a slightly higher peak, they have an offset at their lowest point, rendering the difference negligible.

Regarding the currents of each branch, they are as follows:

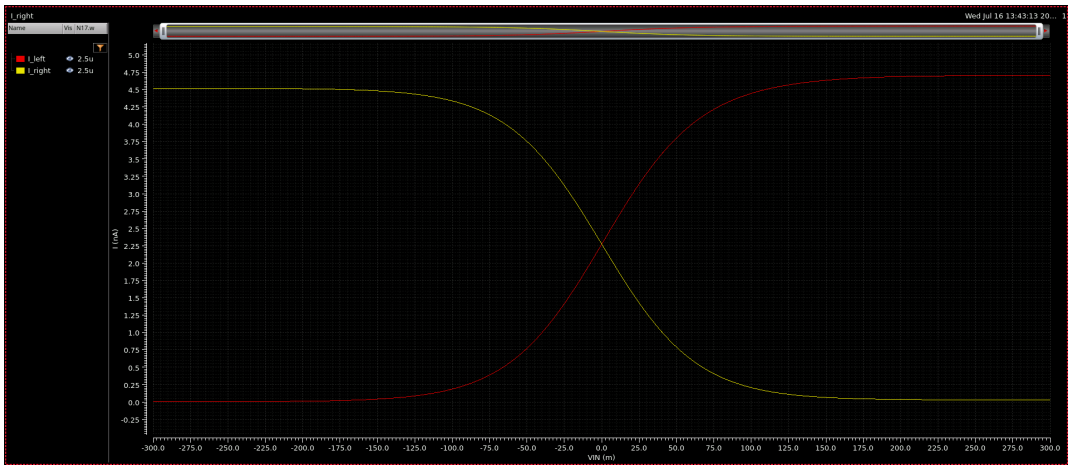


Figure 3: Symmetric Current Correlator Branch Currents

Differential Pair and Current Source	**W/L (μm/μm)**	**Current Correlator**	**W/L (μm/μm)**
M_{n1}, M_{n2}	1.6/0.8	M_{p1}-M_{p6}	2.5/1
M_{n3}, M_{n4}, M_{n17}	2.5/2.5	-	-

Table 1: Transistor Dimensions after Parameter Substitution.

We get the following result for a V_{mean} and V_C sweep:

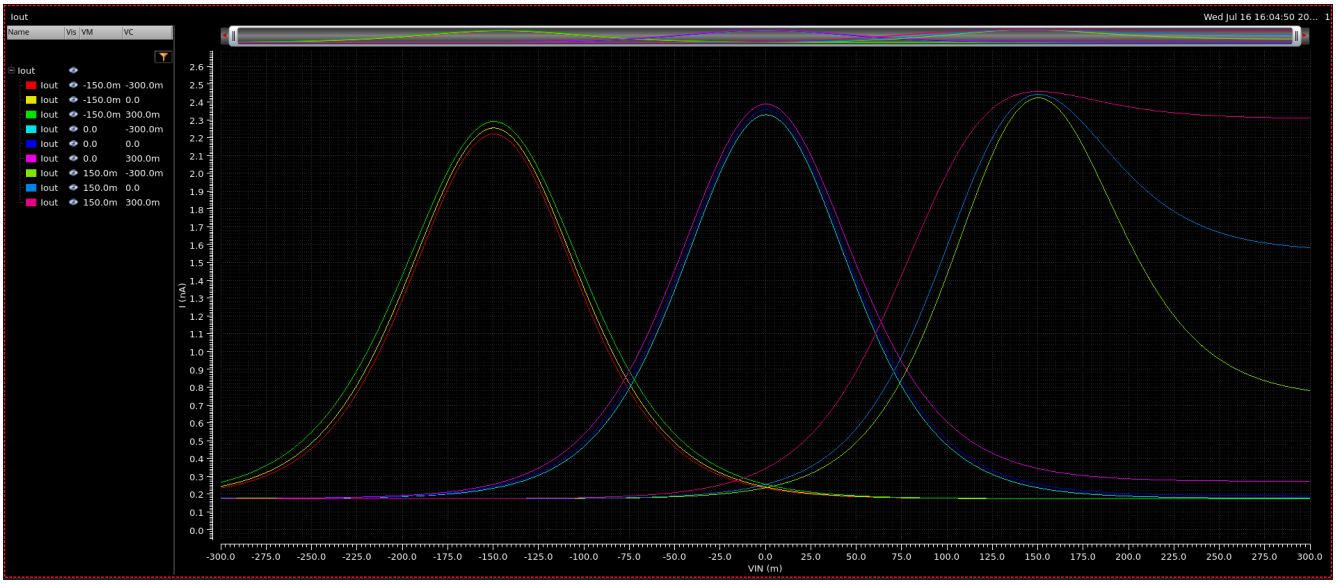


Figure 4: Symmetric Current Correlator Output Current

Moving on, we replace the simple differential pair with a more complex one. Specifically, we use a differential difference pair, as shown in Figure 5.

We also add a simple current mirror connected to the output of the opamp enhanced current source, in order to properly bias the differential pair.

Differential Difference Pair Analysis

In this work two transistors, M_{n1} and M_{n4}, of the differential difference pair are bulk-controlled and their bulks are connected to the parameter voltage V_c. All four transistors M_{n1} to M_{n4} operate in saturation region with voltages $V_D \gg V_S$. More specifically for transistors M_{n1} and M_{n4} their saturation currents are given by the following expressions (using the appropriate aspect ratio W/L):

$$I_{M_{n1}} = 2I_{0n}e^{(\kappa_n V_{in} - V_{S1} + (1-\kappa_n)V_c)/V_T} \tag{1}$$

$$I_{M_{n4}} = 2I_{0n}e^{(\kappa_n V_{in} - V_{S2} + (1-\kappa_n)V_c)/V_T} \tag{2}$$

Transistors M_{n2} and M_{n3} have their bulks connected to V_{SS}. Their saturation currents are given by the following expressions (using the appropriate aspect ratio W/L):

$$I_{M_{n2}} = I_{0n}e^{(\kappa_n V_i - V_{S1} + (1-\kappa_n)V_{SS})/V_T} \tag{4}$$

$$I_{M_{n3}} = I_{0n}e^{(\kappa_n V_{in} - V_{S2} + (1-\kappa_n)V_{SS})/V_T} \tag{5}$$

Transistors $M_{n5} - M_{n7}$ operate in saturation as current mirrors. According to their relative W/L values, the bias current I_{bias} is equal to:

$$I_{\text{bias}} = 2\frac{I_{M_{n1}} + I_{M_{n2}}}{3} \tag{6}$$

$$I_{\text{bias}} = 2\frac{I_{M_{n3}} + I_{M_{n4}}}{3} \tag{7}$$

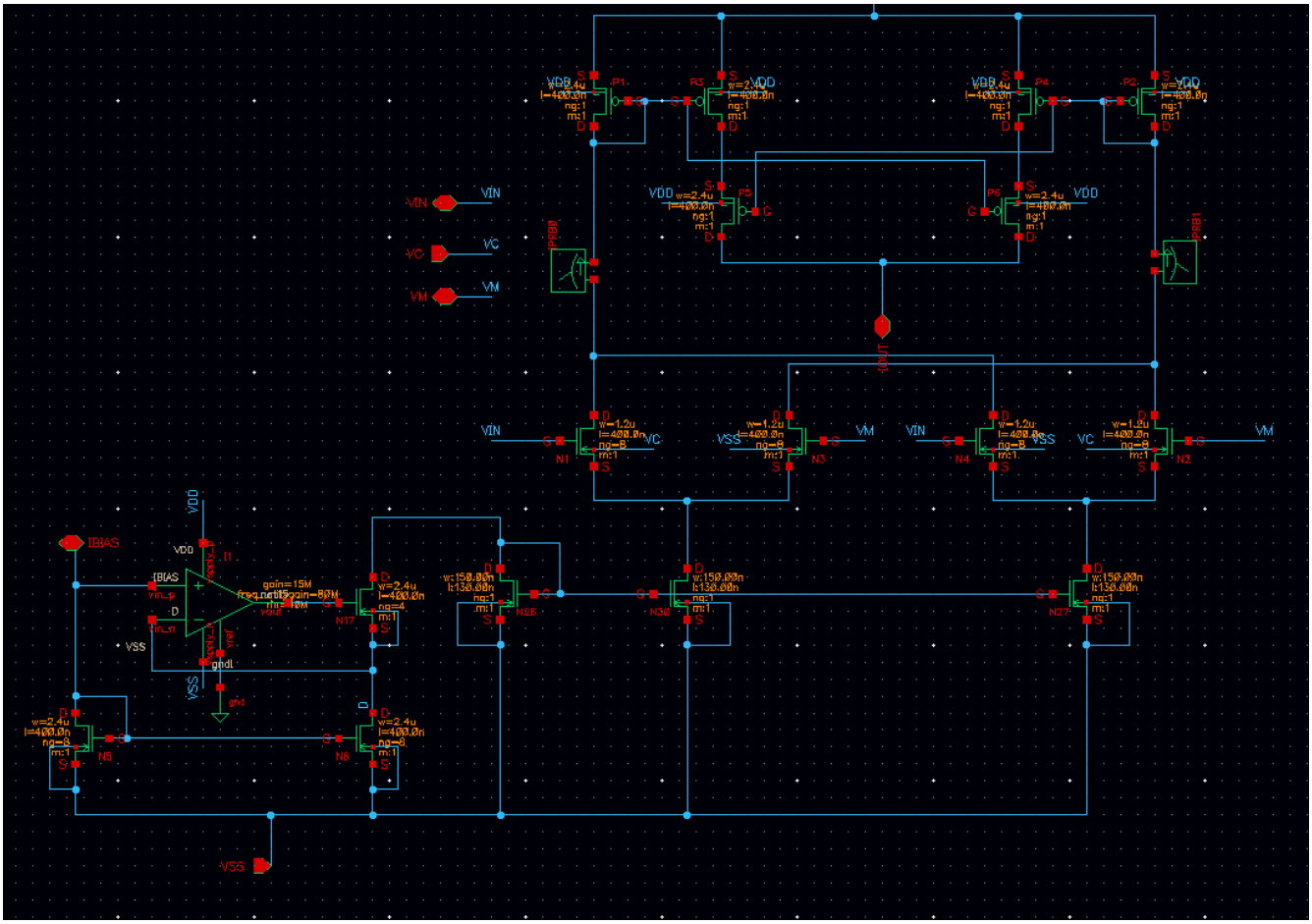


Figure 5: Differential Difference Pair

for the two differential pairs. Combining (1),(4) and (6) we conclude:

$$I_{M_{n1}} = \frac{3I_{\text{bias}}}{2 + e^{(\kappa_n(V_i - V_{in}) + (1-\kappa_n)(V_c - V_{SS}))/V_T}} \tag{8}$$

In order to simplify the expression, we set $\Delta V = V_i - V_{in}$ and $V_{c1} = V_c - V_{SS}$ and the previous equation is transformed in the following way:

$$I_{M_{n1}} = \frac{3I_{\text{bias}}}{2 + e^{(\kappa_n \Delta V + (\kappa_n - 1)V_{c1})/V_T}} \tag{9}$$

In this step we combine (4),(5) and (7). The drain current of M_{n3} is given by:

$$I_{M_{n3}} = \frac{3I_{\text{bias}}}{2 + 4e^{(\kappa_n \Delta V + (1-\kappa_n)V_{c1})/V_T}} \tag{10}$$

The current I_1 is equal to the sum of $I_{M_{n1}}$ and $I_{M_{n3}}$, as shown in Fig. 3.2. The total expression of I_1 is:

$$I_1 = \frac{3I_{\text{bias}}}{2 + e^{(\kappa_n \Delta V + (\kappa_n - 1)V_{c1})/V_T}} + \frac{3I_{\text{bias}}}{2 + 4e^{(\kappa_n \Delta V + (1-\kappa_n)V_{c1})/V_T}} \tag{11}$$

The current I_2 is equal to the sum of $I_{M_{n2}}$ and $I_{M_{n4}}$. By using the same methodology we can calculate the total expression of I_2 which is given by:

$$I_2 = \frac{3I_{\text{bias}}}{2 + e^{(-\kappa_n \Delta V + (\kappa_n - 1)V_{c1})/V_T}} + \frac{3I_{\text{bias}}}{2 + 4e^{(-\kappa_n \Delta V + (1-\kappa_n)V_{c1})/V_T}} \tag{12}$$

After multiple parametric sweeps we end up with the following results:

Differential Pair and Current Source	**W/L (μm/μm)**	**Current Correlator**	**W/L (μm/μm)**
$M_{n1} - M_{n4}$	1.6/0.4	M_{p1}-M_{p6}	2.5/0.4
M_{n5}, M_{n6}	2.5/0.2	-	-
M_{n17}	0.8u/0.2	-	-
$M_{n26}, M_{n27}, M_{n30}$	1.6/0.2	-	-

Table 2: Transistor Dimensions after Parameter Substitution.

And the following graphs:

- **Iout** for V_{mean} and V_C sweep:

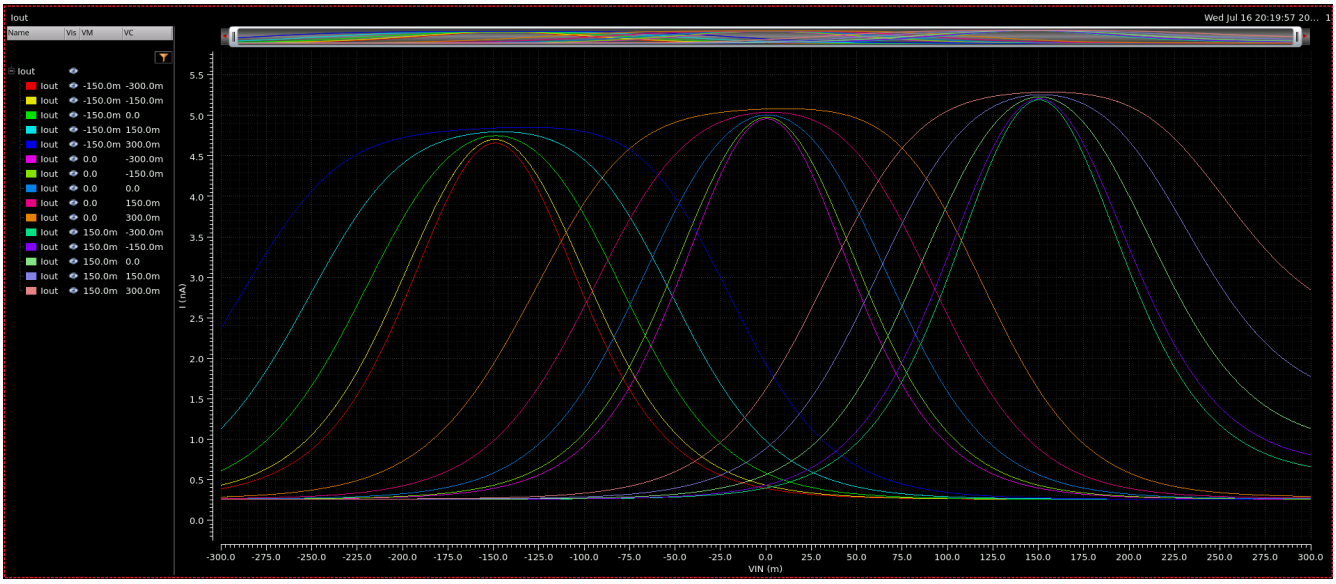


Figure 6: Differential Difference Pair Output Current

- **PSRR-**:

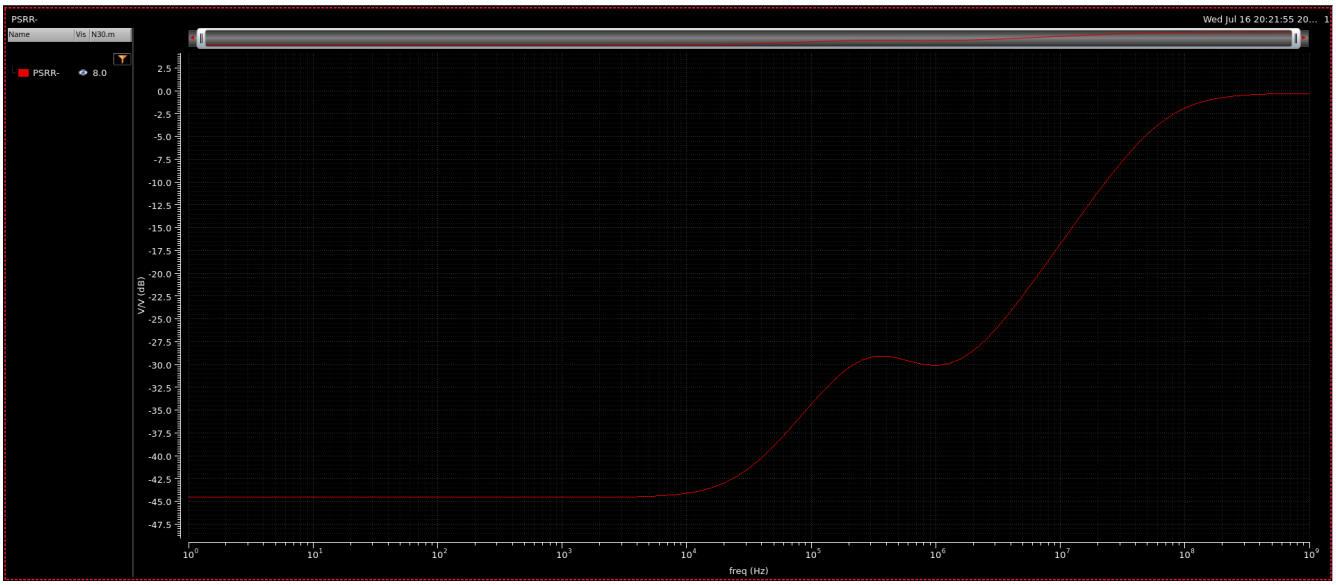


Figure 7: PSRR-

- **PSRR+**:

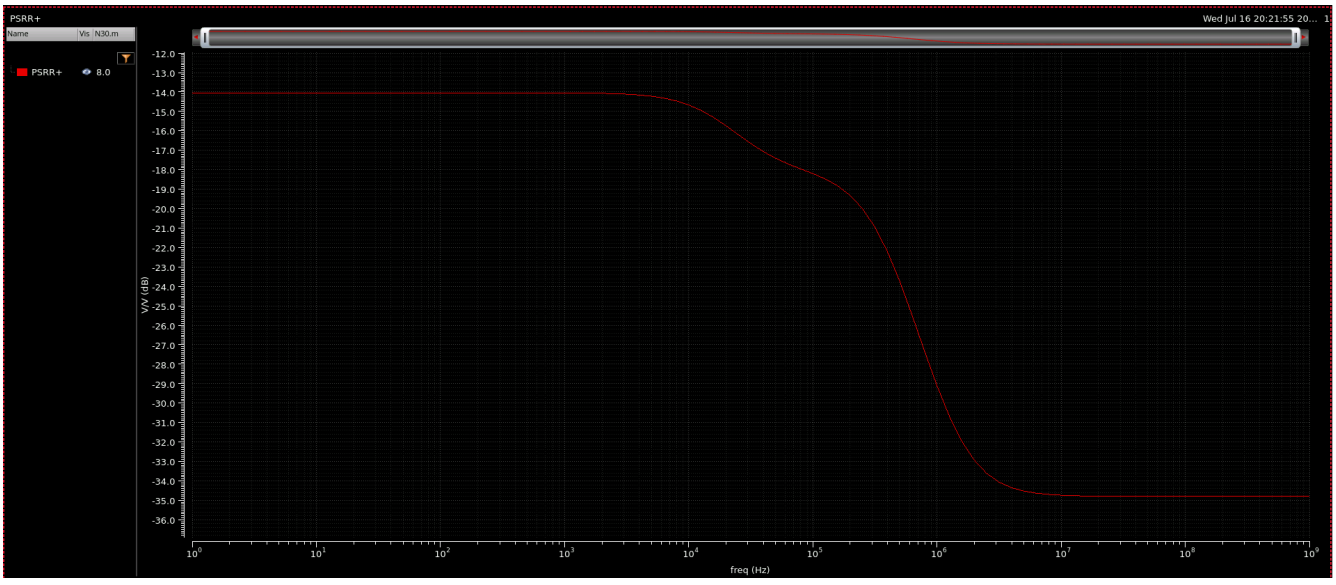


Figure 8: PSRR+

- **Power**:

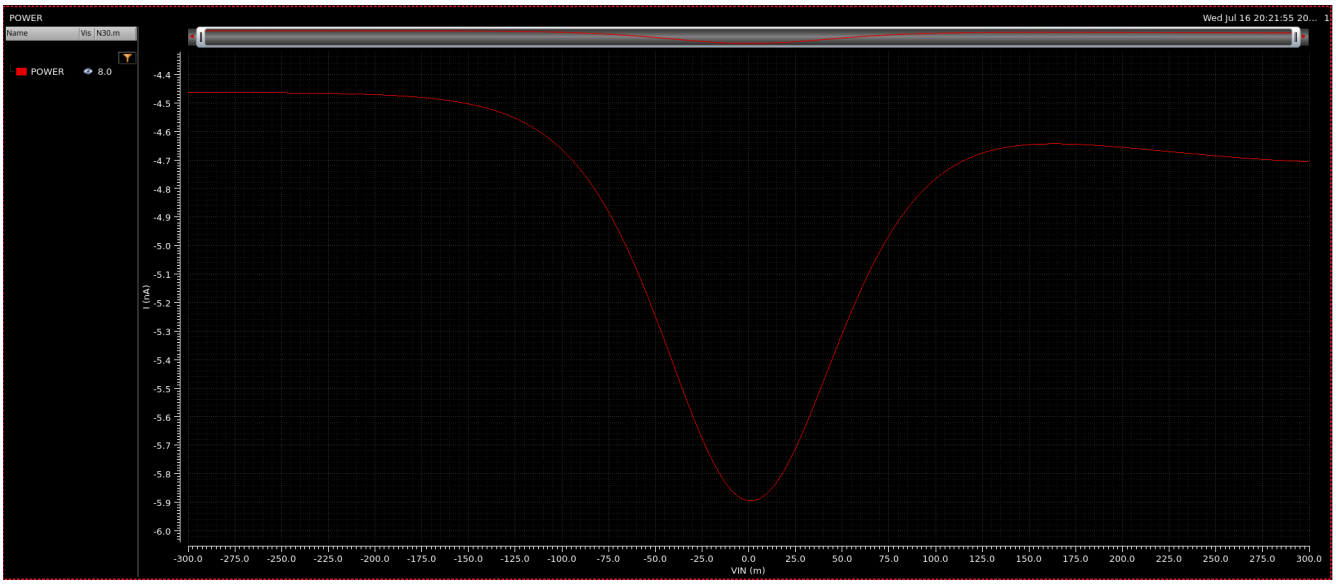


Figure 9: Power Consumption

- **Noise**:

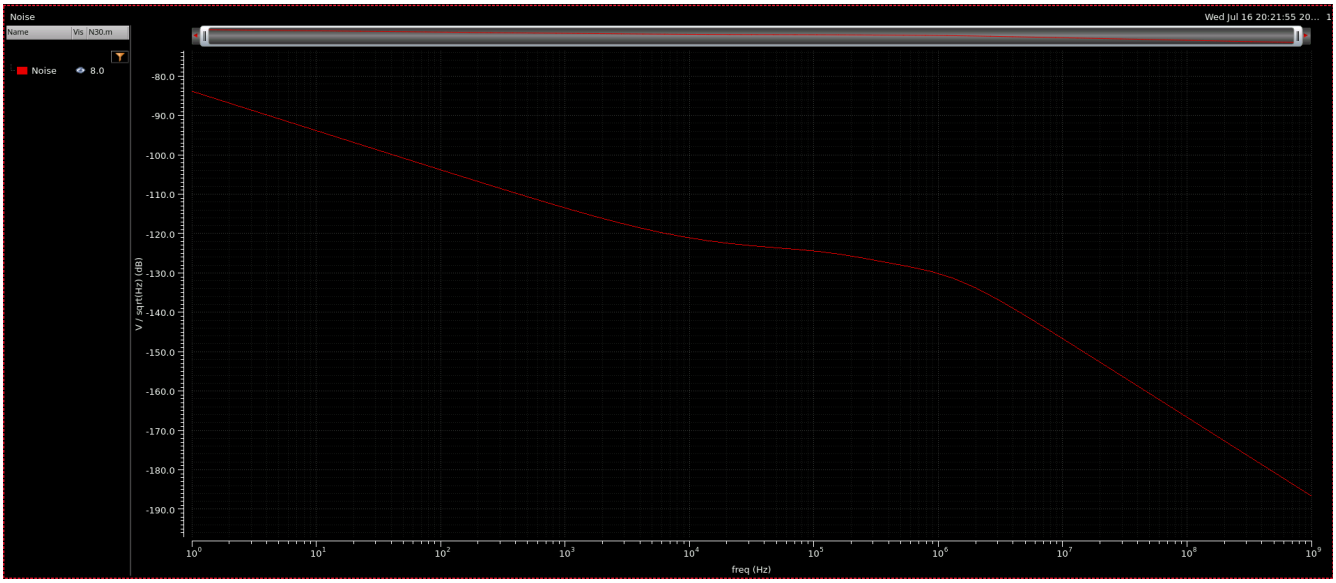


Figure 10: Noise Analysis

- **Ibias** sweep:

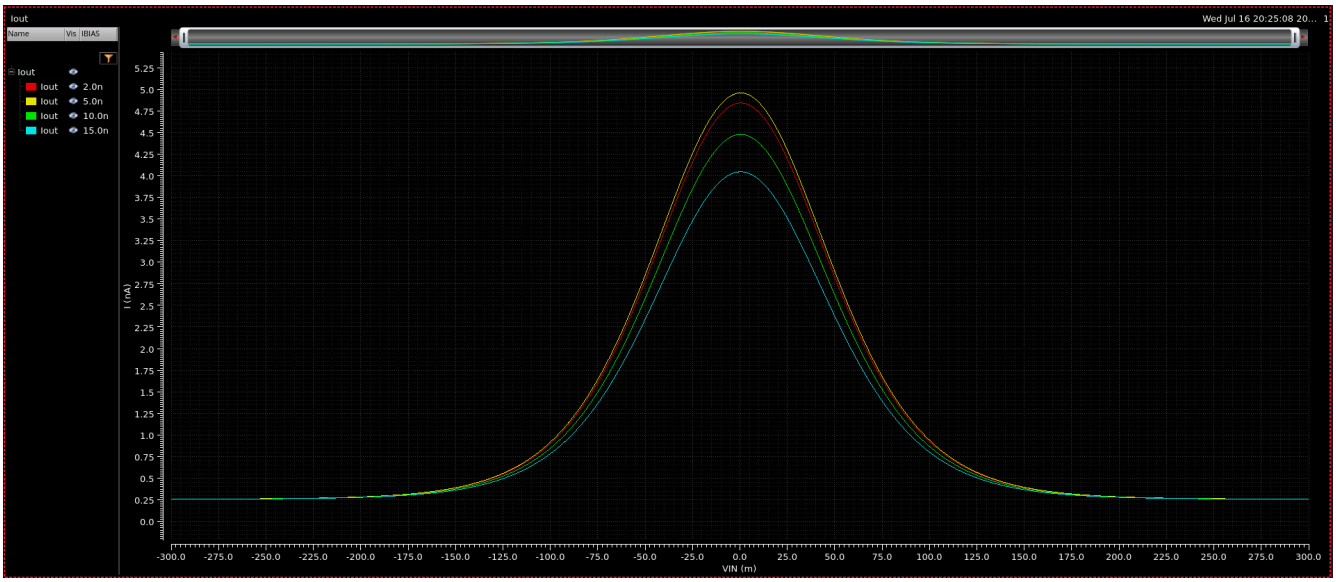


Figure 11: Ibias Sweep

We observe that the circuit is tunable with respect to V_{mean} and V_c, but it is not at all tunable with respect to I_{bias}. Specifically, increasing I_{bias} decreases I_{out}!

18 July

We swap out the opamp based current source and replace it with a cascode current mirror.

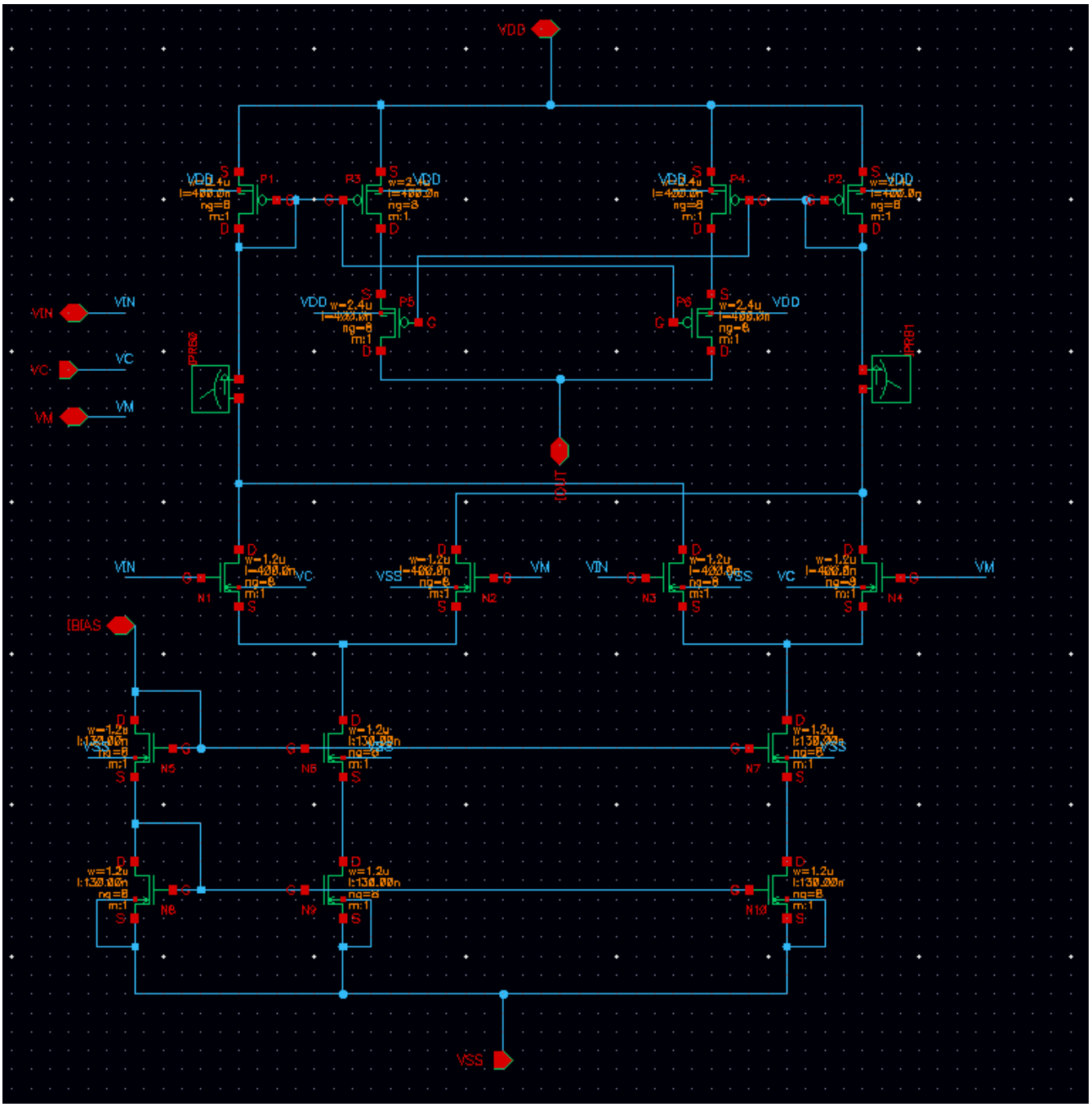


Figure 1: Cascode Current Mirror Configuration

We will now begin sizing the bump circuit.

Bump Circuit Sizing

Cascode Current Mirror

As previously discussed, the output resistance of the cascode current mirror is given by:

$$R_{out} = g_m r_{ds}^2$$

With r_{ds} being analogous to transistor length L, it becomes apparent that we should use long channel devices for the cascode current mirror. The effect of a bad current source is that the source voltage of the differential fluctuates, therefore we have a varying V_{DS}, leading to the extremely low magnitudes we observed when shifting V_{mean} towards the lower end.

Therefore, our simulations will focus on $V_{mean} = -150mV$.

We shall also investigate the effect of fingers on the circuit.

- Fingers = 1

We match transistors $M_{n5} - M_{n10}$ on all their parameters and we perform the following sweep:

- W = 200n, 400n, 1u, 1.5u
- L = 1u, 1.5u, 2u, 2.5u, 3u

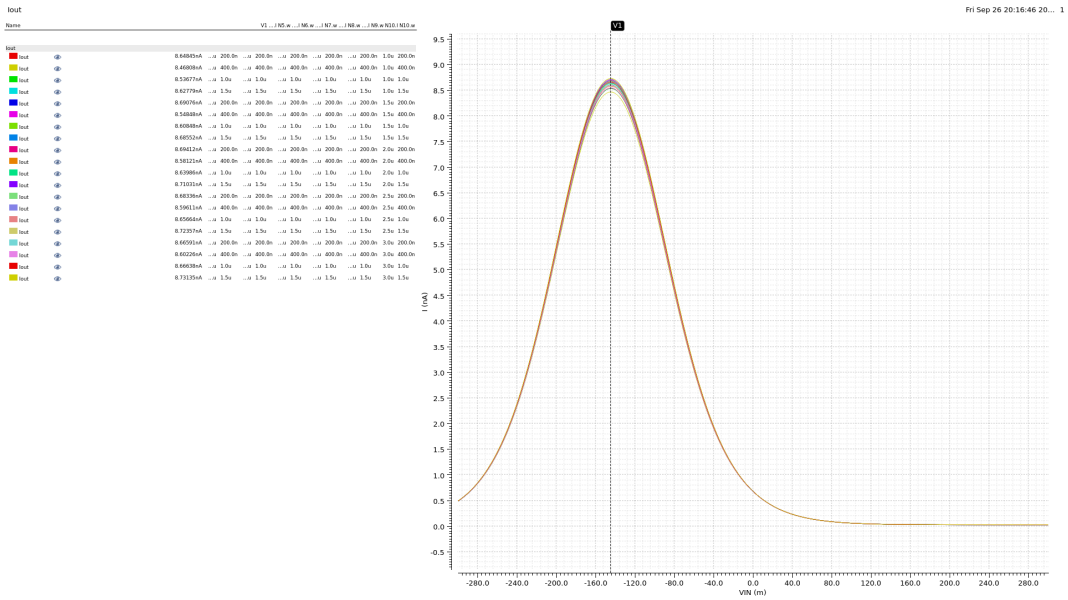


Figure 2: Cascode Current Mirror Sweep, Fingers = 1

We choose $W = 200n$ and $L = 2u$ and we sweep the multipliers:

- M = 1, 2, 4, 6

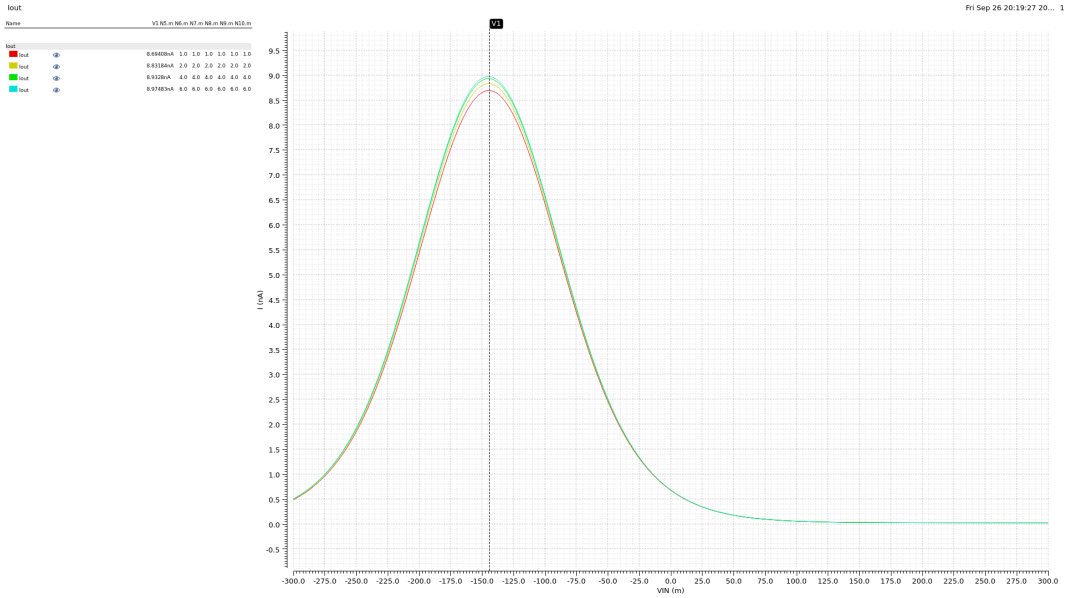


Figure 3: Cascode Current Mirror Multiplier Sweep, Fingers = 1

We observe that the output resistance increases with increasing multiplier.

- Fingers = 2

We perform the same sweeps as before.

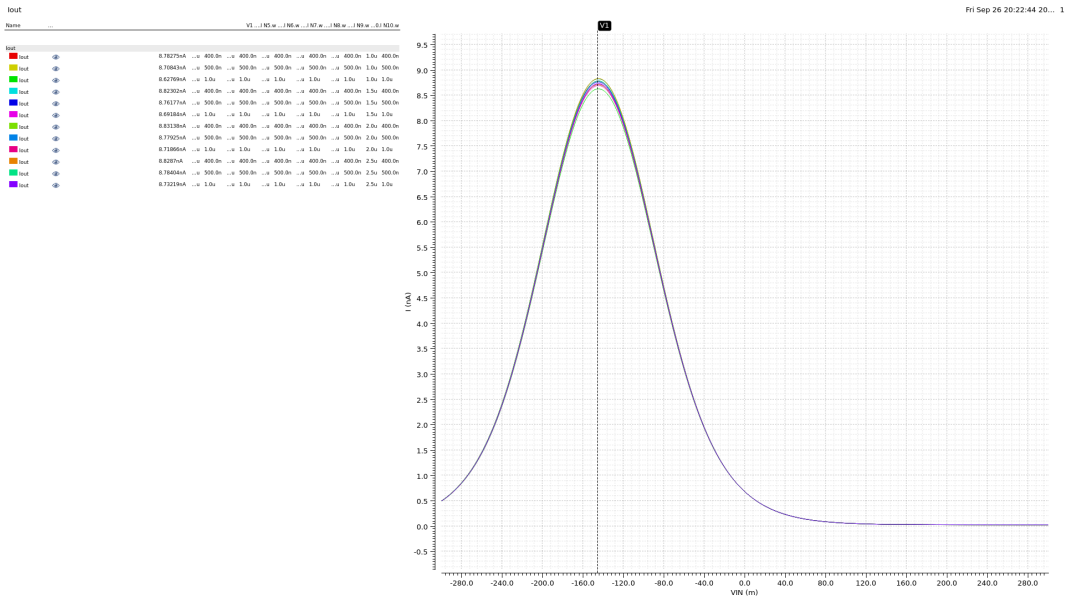


Figure 4: Cascode Current Mirror Sweep, Fingers = 2

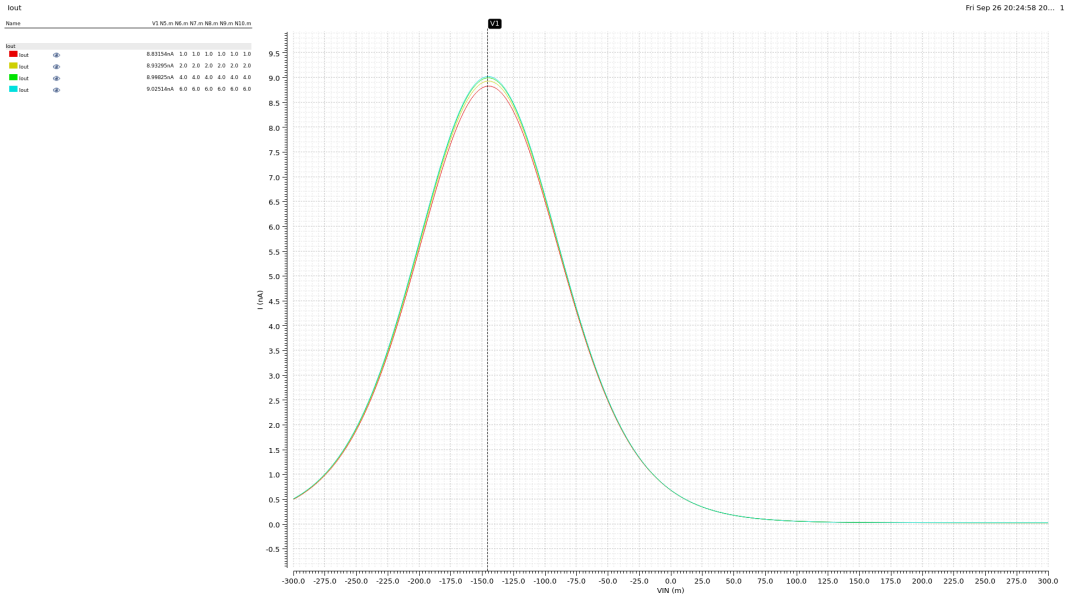


Figure 5: Cascode Current Mirror Multiplier Sweep, Fingers = 2

A clear improvement is observed.

- Fingers = 4

We perform the same sweeps as before.

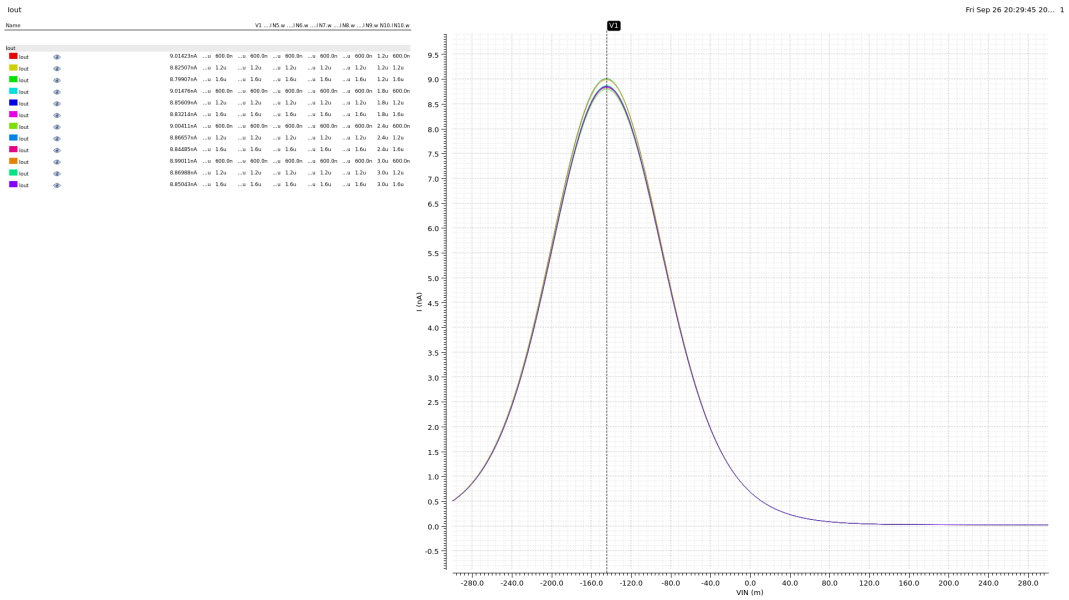


Figure 6: Cascode Current Mirror Sweep, Fingers = 4

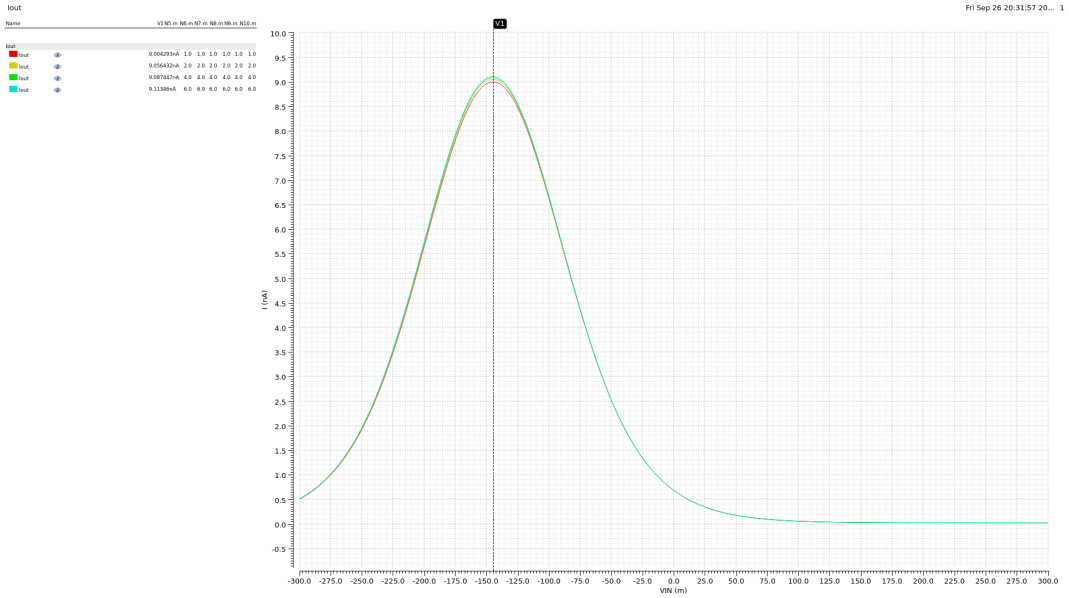


Figure 7: Cascode Current Mirror Multiplier Sweep, Fingers = 4

We are now observing diminishing returns, but the improvement is still clear.

- Fingers = 8

We perform the same sweeps as before.

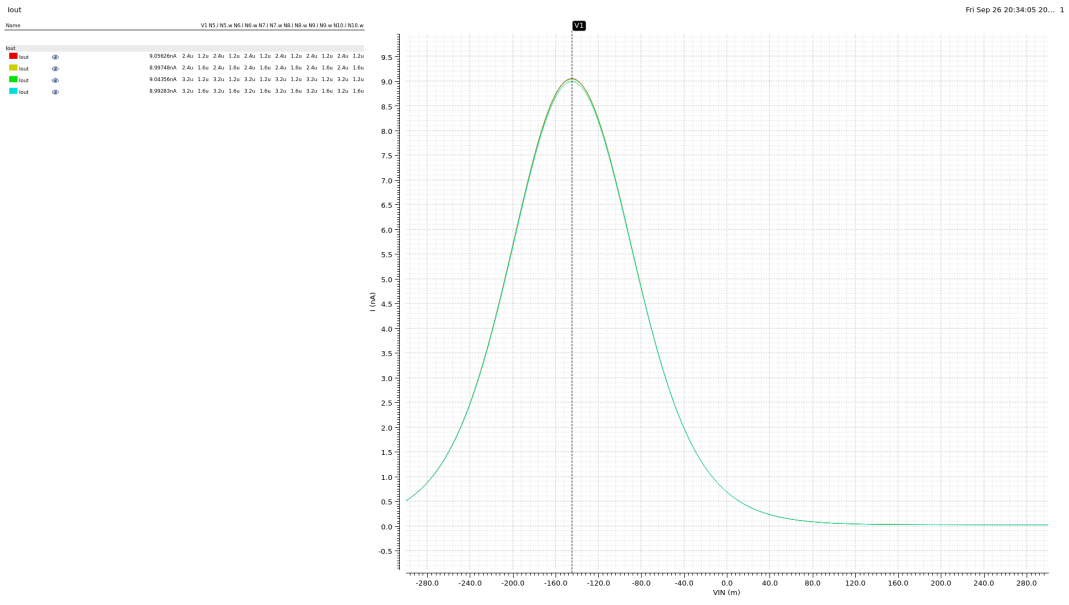


Figure 8: Cascode Current Mirror Sweep, Fingers = 8

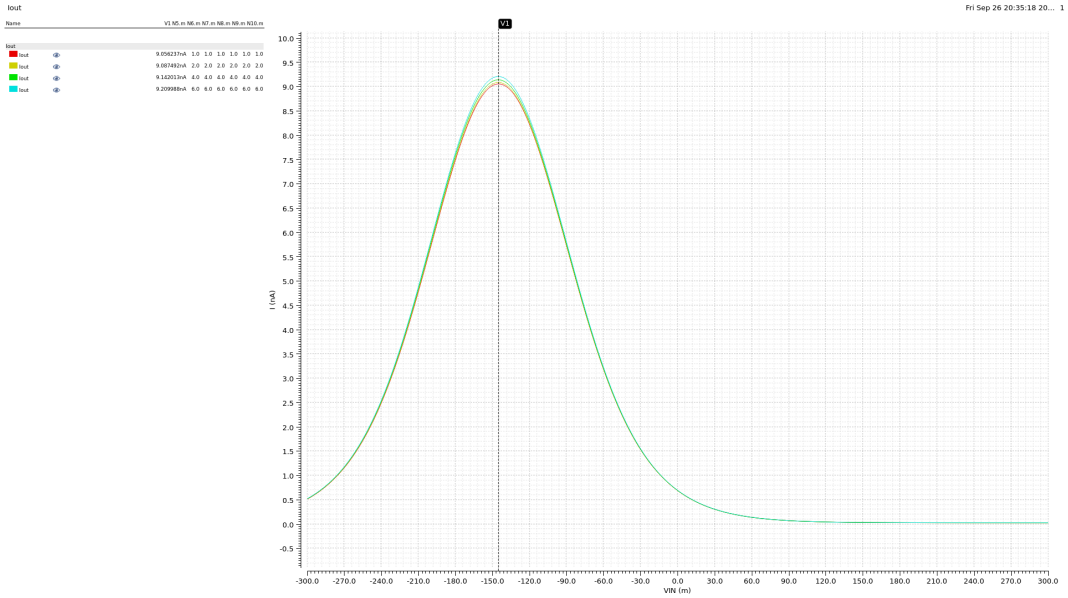


Figure 9: Cascode Current Mirror Multiplier Sweep, Fingers = 8

We are now finished with the simulations. We don't choose the strictly best performing configuration, as it would take up too much area and increase consumption disanageously to its gains. We choose the following configuration:

- Fingers = 4, $W/L = 600n/2.4u$, m = 4

and get the following results:

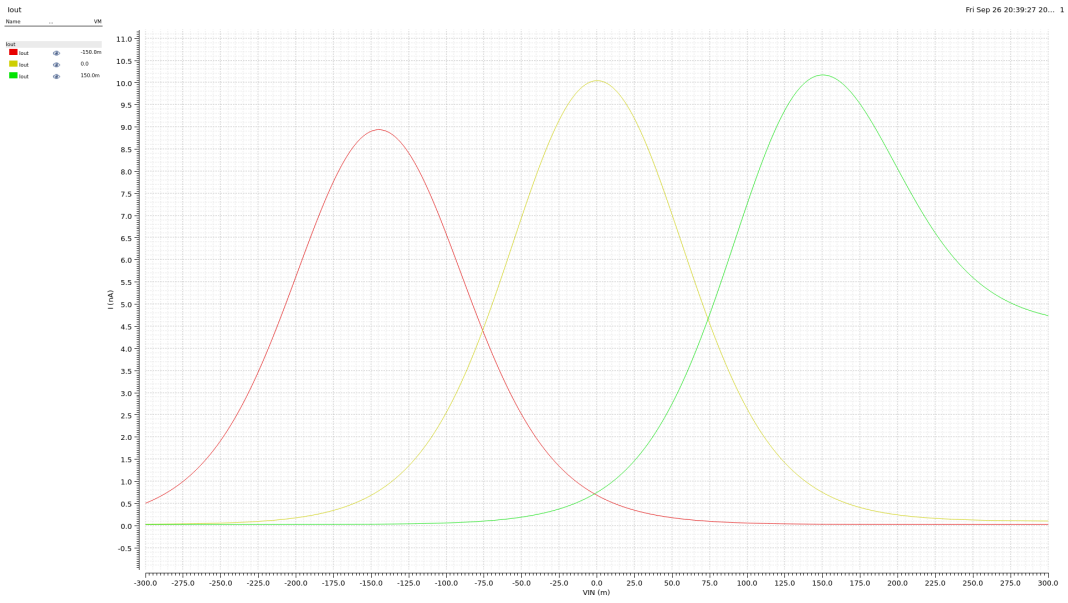


Figure 10: Cascode Current Mirror Chosen Configuration, Fingers = 4, $W/L = 600n/2.4u$, m = 4

We will revisit this decision after we size the rest of the circuit.

Differential Difference Pair

We will now size the differential difference pair. Again, our main focus will be $V_{mean} = -150mV$, since the sizing of the Current Correlator will be aimed at the effects observed at higher voltages. Initially, we will match all transistors on all parameters and perform the following sweeps:

- Fingers = 1
- W = 1u, 1.6u, 2.4u
- L = 200n, 400n, 800n, 1.6u

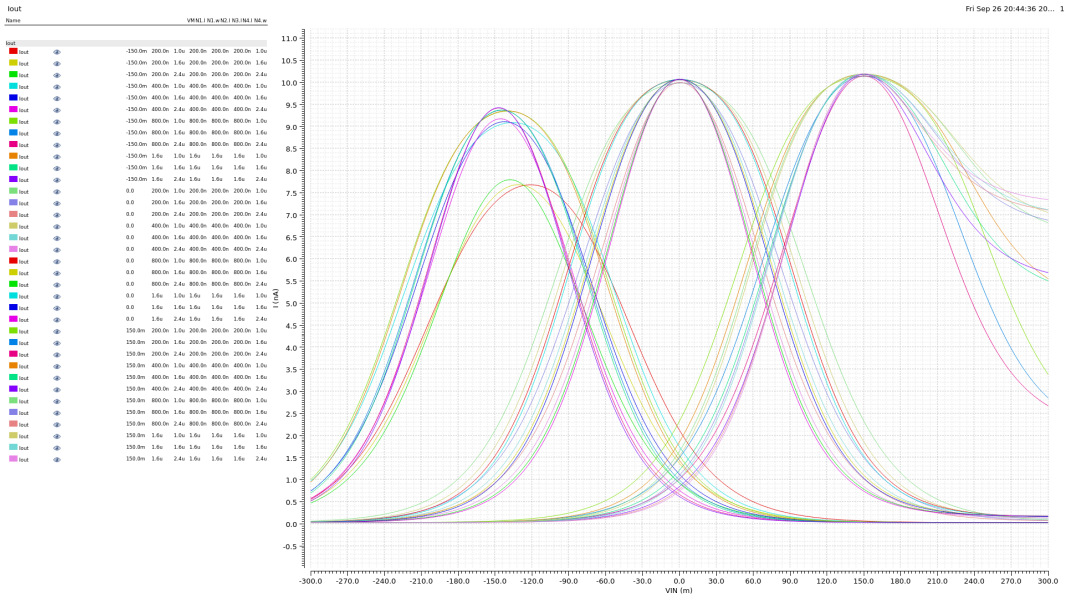


Figure 11: Diff Diff Pair Sweep, Fingers = 1

Clearly, the results are all over the place, with multiples levels of performance. The only configurations that seem to perform consistently well are the ones with larger transistor sizes and fairly small W/L ratios. Also, we notice a slight offset in the mean of the Gaussian output. We also notice that the sizing of the DDP affects the variance of the Gaussian output, similarly to how the parameter V_C does.

We choose $W = 2.4u$ and $L = 1.6u$ and we sweep the multipliers:

- M = 1, 2, 4, 6

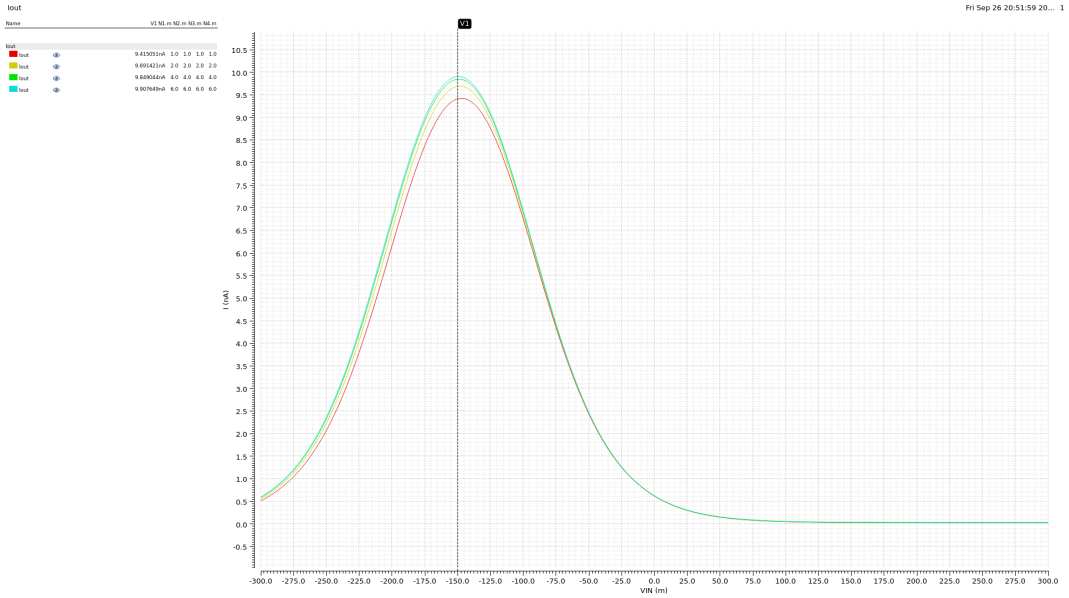


Figure 12: Diff Diff Pair Multiplier Sweep, Fingers = 1

- Fingers = 2

We perform the same sweeps as before.

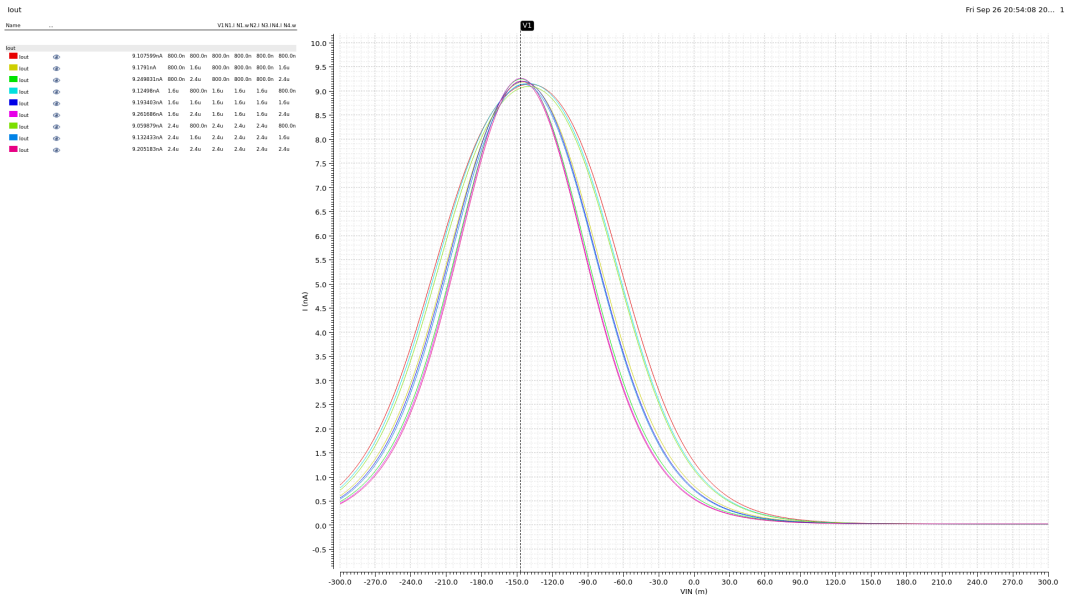


Figure 13: Diff Diff Pair Sweep, Fingers = 2

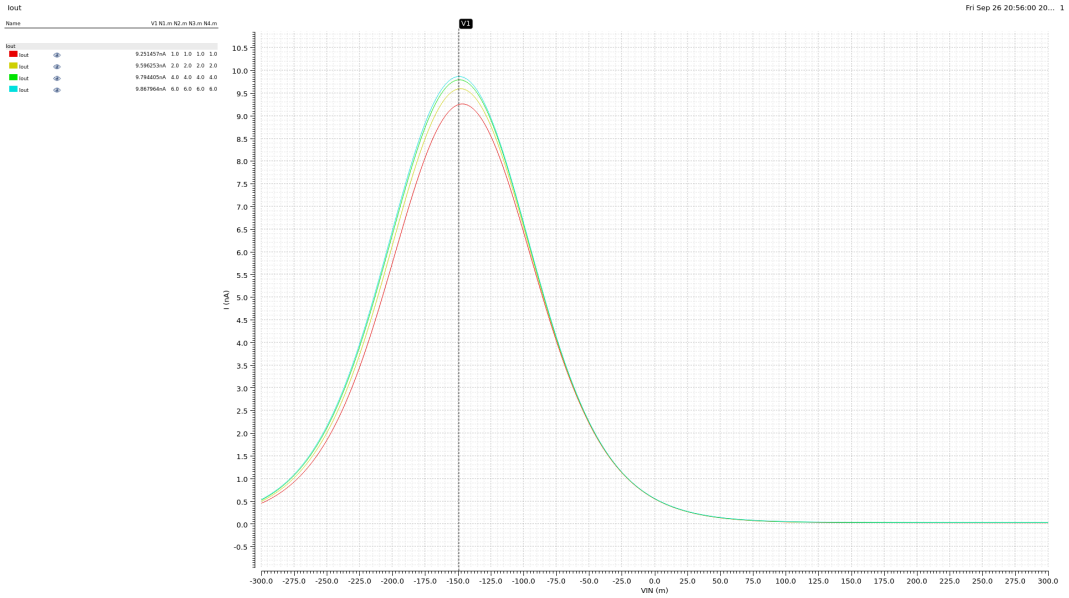


Figure 14: Diff Diff Pair Multiplier Sweep, Fingers = 2

Very good results are observed and the offset has been reduced.

- Fingers = 4

We perform the same sweeps as before.

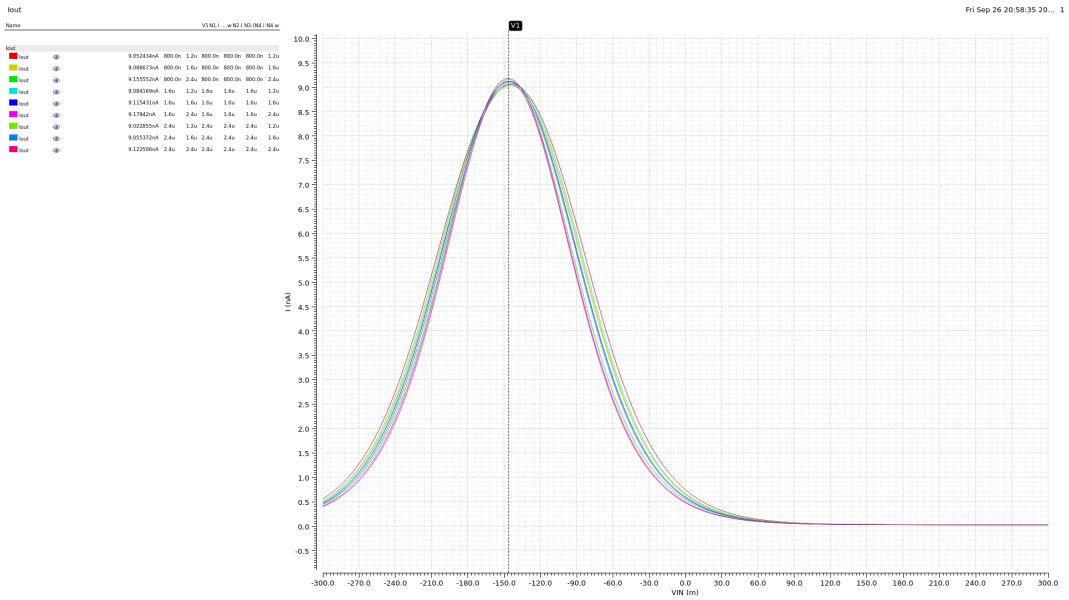


Figure 15: Diff Diff Pair Sweep, Fingers = 4

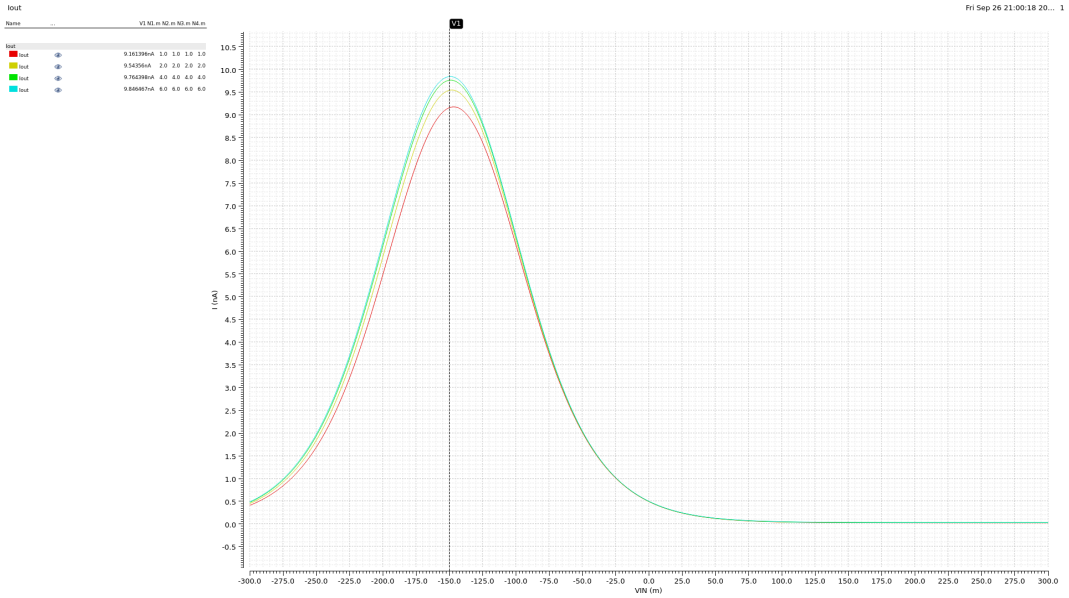


Figure 16: Diff Diff Pair Multiplier Sweep, Fingers = 4

Again, very good results are observed and the offset has practically been eliminated.

We conclude that the best performing configuration is:

- Fingers = 4, $W/L = 2.4u/1.6u$, m = 4

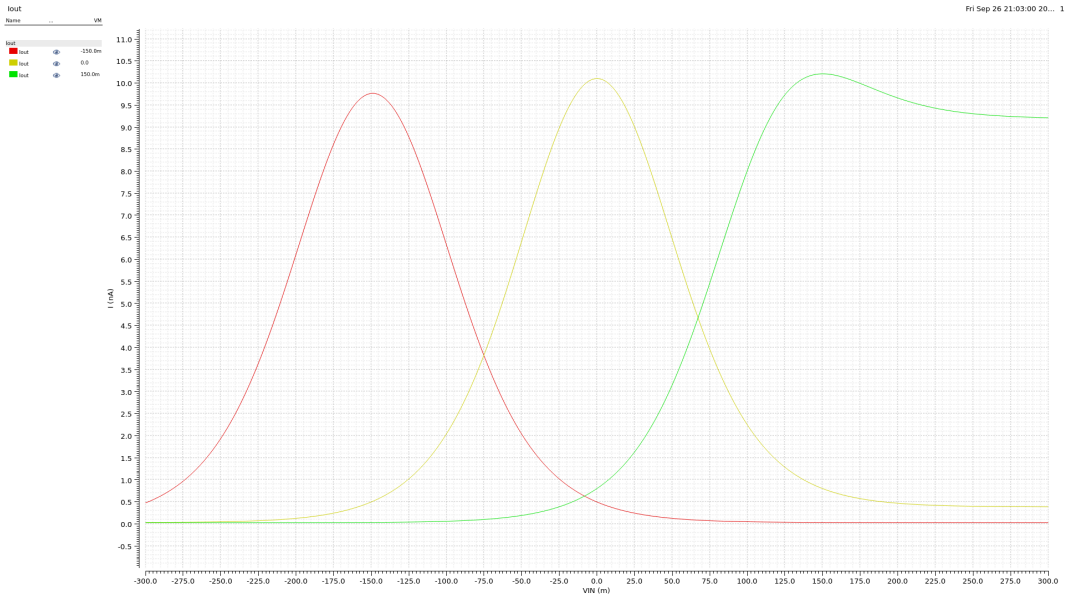


Figure 17: Diff Diff Pair Chosen Configuration, Fingers = 4, $W/L = 2.4u/1.6u$, m = 4

The areas we have targeted so far seem to be performing well. We will now move on to the current correlator.

Current Correlator

We will now size the current correlator.

In this case, we will focus on $V_{mean} = 150mV$ since this is where we observed the most issues with the previous configuration. The main issue we observed was the Gaussian output not returning to 0 at the tails, but rather settling at a higher value. This is due to the fact that there is not enough voltage headroom for the transistors to operate in the subthreshold region, therefore they operate in the triode region and conduct a small amount of current. To tackle this, we will try to minimize L and maximize W, in order to ensure the subthreshold operation of the transistors.

All transistors will be matched on all parameters and we will perform the following sweeps:

- Fingers = 1
- W = 0.8u, 1.6u, 2.4u
- L = 0.2u, 0.4u, 1u

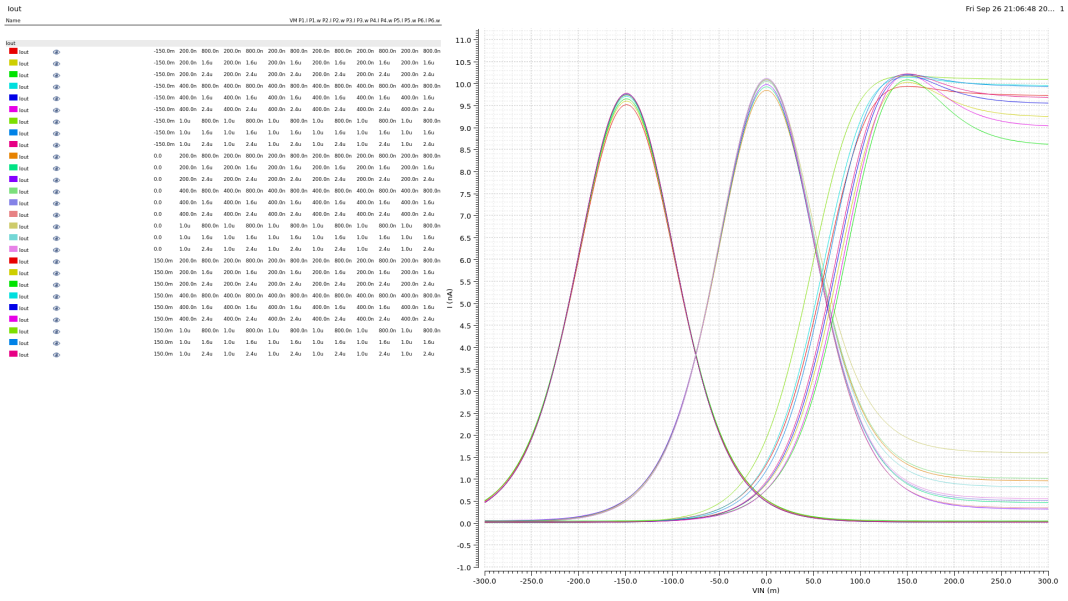


Figure 18: Current Correlator Sweep, Fingers = 1

And the multiplier sweep:

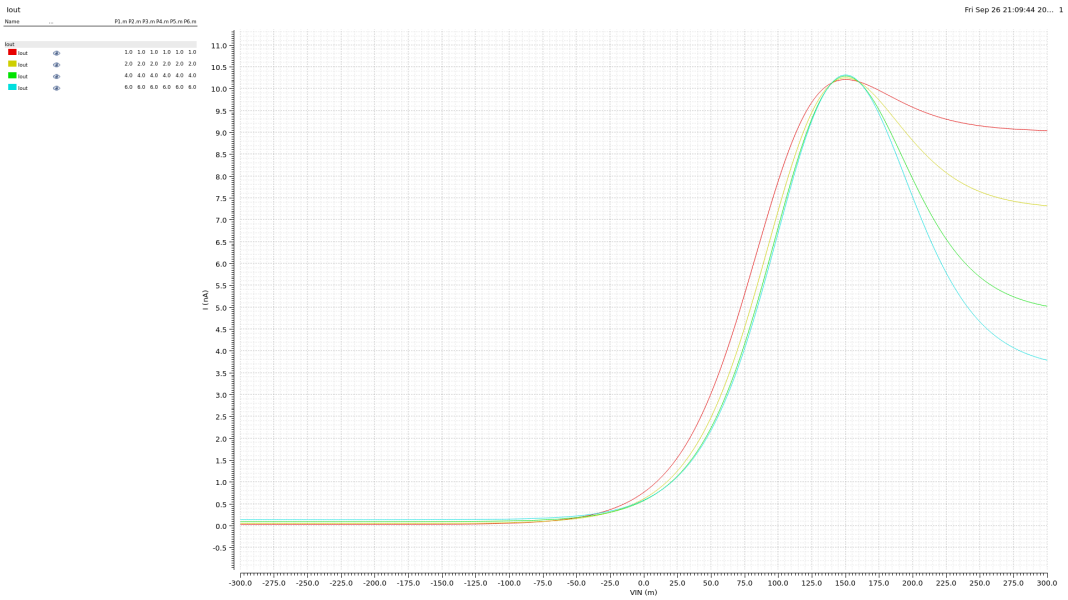


Figure 19: Current Correlator Multiplier Sweep, Fingers = 1

There is a huge improvement already, especially by increasing the multipliers.

- Fingers = 2

We perform the following sweeps:

- W = 1.6u, 2.4u, 3.2u
- L = 0.2u, 0.4u, 1u

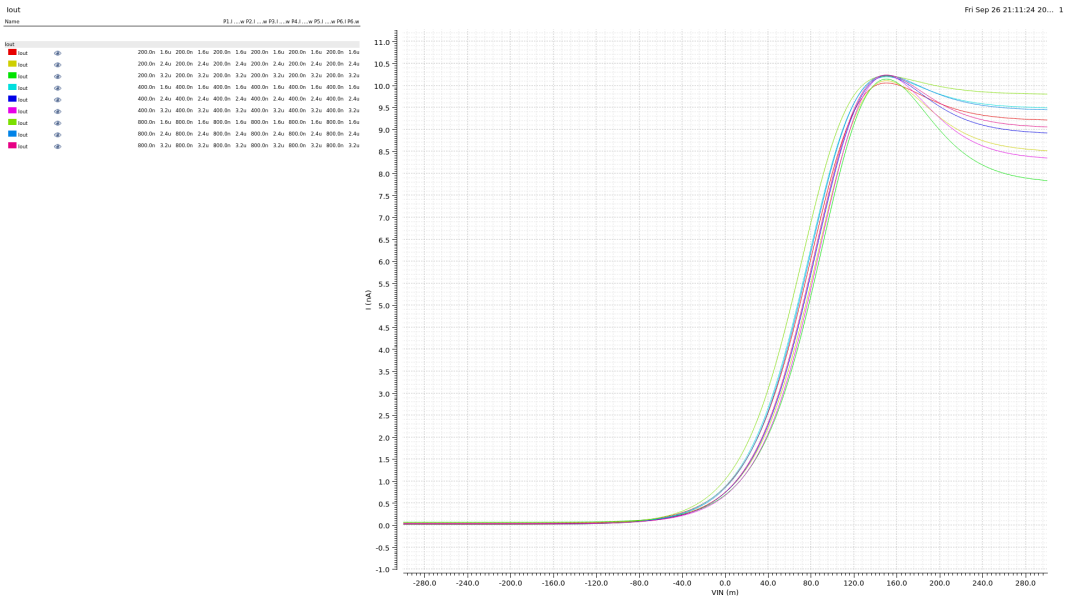


Figure 20: Current Correlator Sweep, Fingers = 2

Again, great improvement for 3.2u/0.2u.

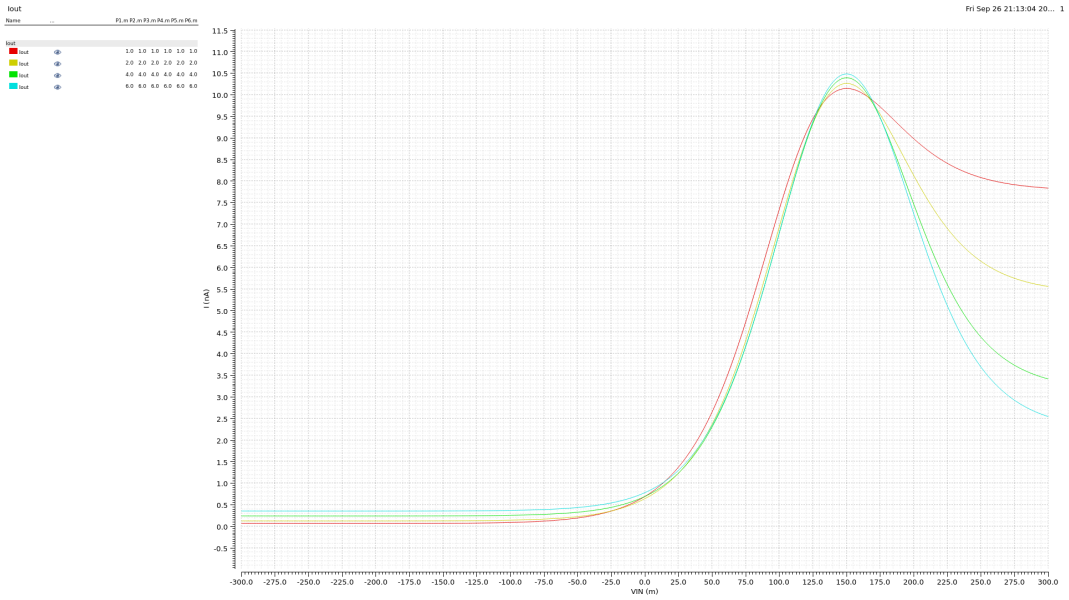


Figure 21: Current Correlator Multiplier Sweep, Fingers = 2

- Fingers = 4

We perform the same sweeps as before.

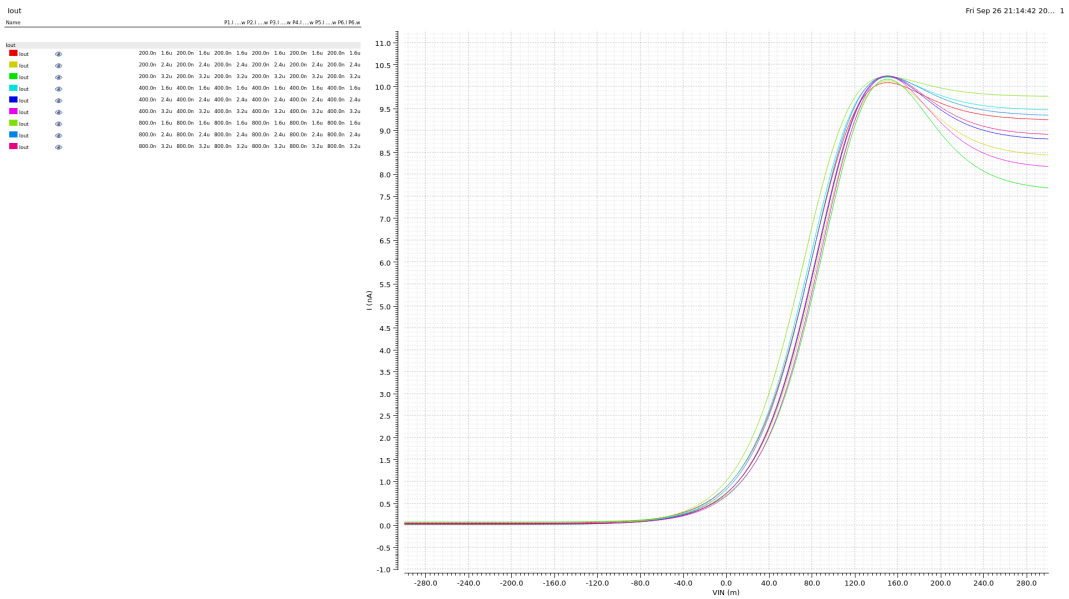


Figure 22: Current Correlator Sweep, Fingers = 4

Overall, it becomes abundantly clear that increasing the fingers and multipliers has a huge effect on the performance of the circuit, therefore after some further testing we choose the following configuration:

- Fingers = 8, $W/L = 2.4u/0.2u$, m = 10

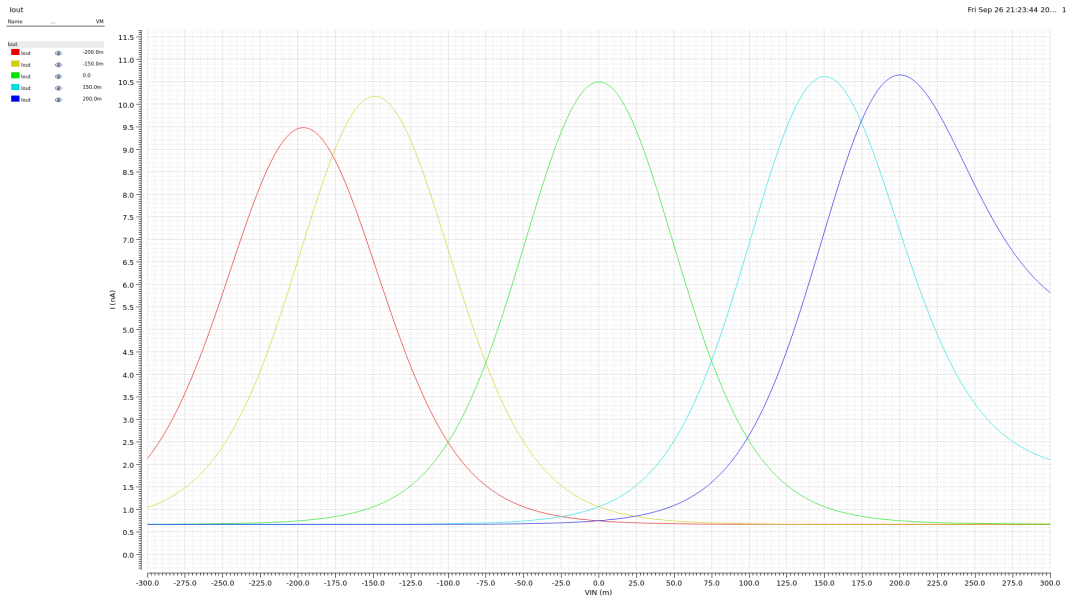


Figure 23: Current Correlator Chosen Configuration, Fingers = 8, $W/L = 2.4u/0.2u$, m = 10

We now iterate through the entire process once more.

We fix the DDP offset by setting the width of transistors $M_{n2} and M_{n3}$ to 3 times the width of transistors $M_{n1} and M_{n4}$.

After multiple simulations we arrive at the following final configuration:

Block	**Transistors**	**Fingers**	**W/L**	**Multiplier**
Cascode Current Mirror	$M_{P1} - M_{P6}$	8	2.4u/0.2u	10
Differential Difference Pair	M_{n1}, M_{n4}	4	1.2u/1.2u	4
Differential Difference Pair	M_{n2}, M_{n3}	4	3.6u/1.2u	4
Current Correlator	$M_{N5} - M_{N10}$	4	0.6u/2.4u	1

Final results:

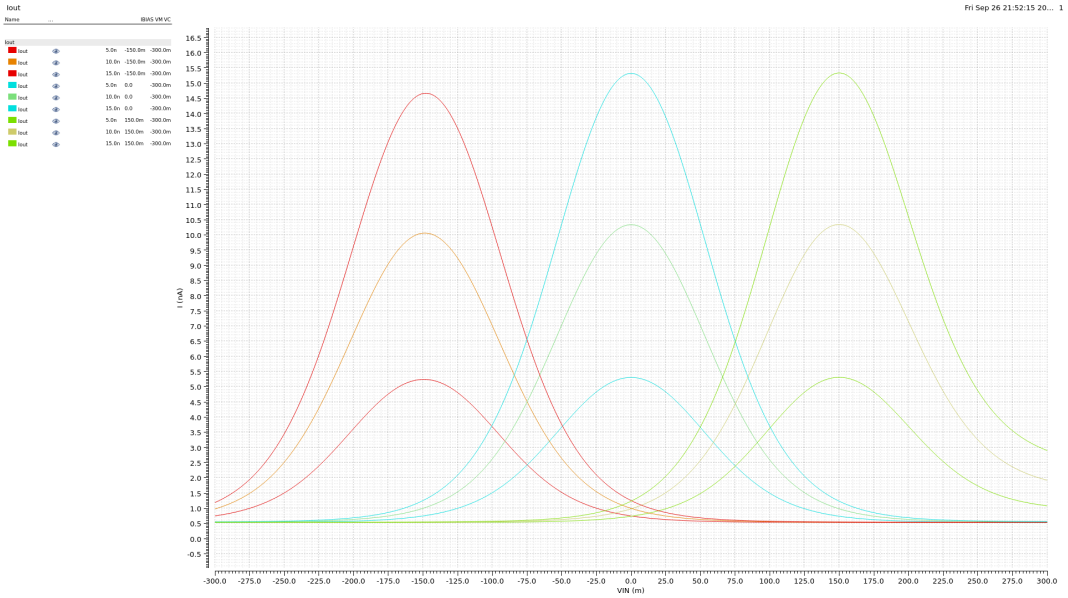


Figure 24: Final Configuration Results - Ibias and Vmean Sweep

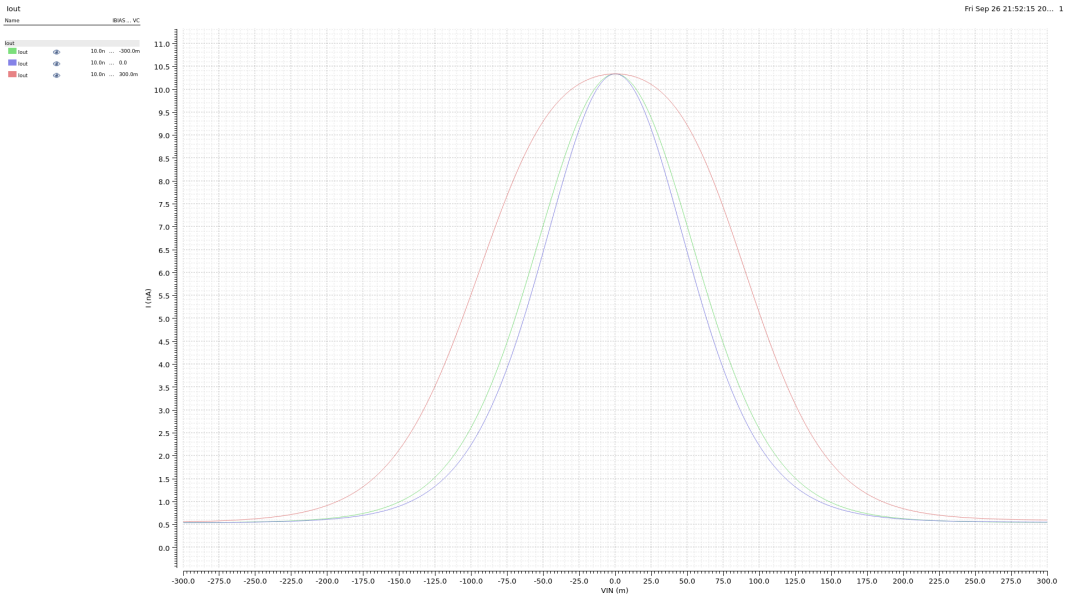


Figure 25: Final Configuration Results - Vc sweep

And here's effect of W/L of the DDP on the variance of the Gaussian output:

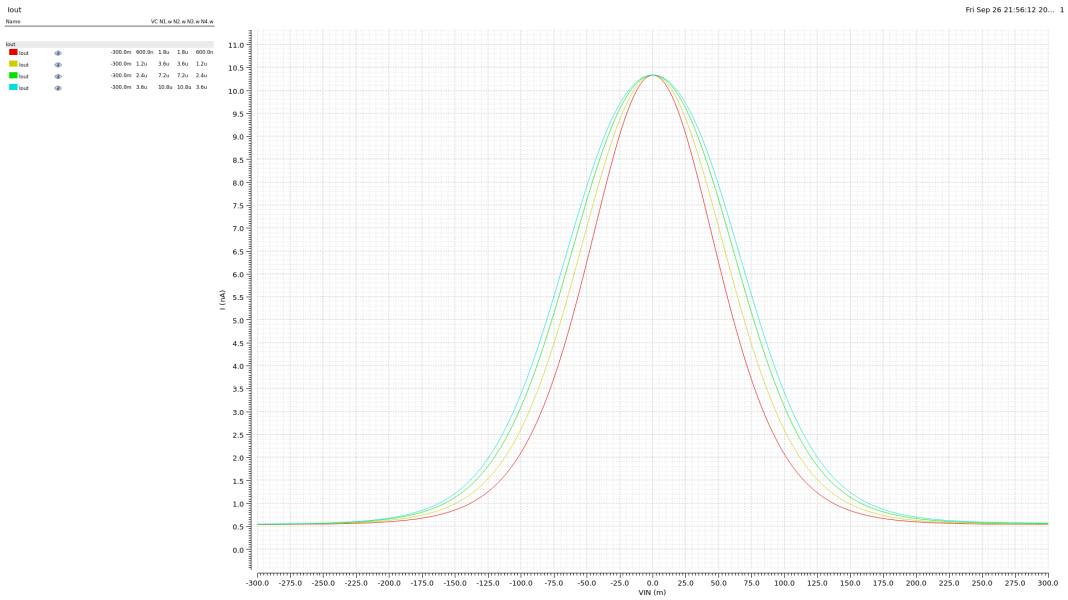


Figure 26: Effect of DDP Sizing on Variance of Gaussian Output

29 July

For the time being, we consider our bump and its sizing to be finished. We now move on to the next step, which is the design of the analog classifier.

Theoretical Background

Gaussian Mixture Model

Mixture Models are probabilistic models that assume that the data is generated from a mixture of several underlying probability distributions. Each distribution represents a different cluster or component in the data. The Gaussian Mixture Model (GMM) is a specific type of mixture model that assumes that the data is generated from a mixture of several Gaussian distributions. Each Gaussian distribution is characterized by its mean and covariance, which determine its shape and orientation in the feature space.

A GMM λ_c is uniquely defined by the number of components K, the weight factors $\{w_i^c\}_{i=1}^K$, the mean value vectors $\{M_i^c\}_{i=1}^K$, $M_i^c \in \mathbb{R}^N$ and the covariance matrices $\{\Sigma_i^c\}_{i=1}^K$, $\Sigma_i^c \in \mathbb{R}^{N\times N}$ of each Gaussian component. Let us consider an N-dimensional input vector $X \in \mathbb{R}^N$. The probability density function (PDF) of X, as approximated by λ_c, is given by:

$$p(X|\lambda_c) = \sum_{i=1}^{K} w_i^c \cdot \mathcal{N}(X|M_i^c, \Sigma_i^c) \tag{1}$$

Here it holds that $\sum_{i=1}^K w_i^c = 1$ and $0 \leq w_i^c \leq 1$ for $i = 1, 2, \ldots, K$. The i-th N-D Gaussian component of λ_c is denoted by $\mathcal{N}(X|M_i^c, \Sigma_i^c)$ and its value is given by

$$\mathcal{N}(X|M_i^c, \Sigma_i^c) = \frac{\exp\left(-\frac{1}{2}(X - M_i^c)^T(\Sigma_i^c)^{-1}(X - M_i^c)\right)}{\sqrt{(2\pi)^N|\Sigma_i^c|}} \tag{2}$$

where $|\cdot|$ denotes the Euclidean norm. For a diagonal matrix Σ_i^c, the above expression is simplified to

$$\mathcal{N}(X|M_i^c, \Sigma_i^c) = \prod_{n=1}^{N} \mathcal{N}(x_n|\mu_n^c, (\sigma_n^c)^2) \tag{3}$$

where x_n, μ_n^c and $(\sigma_n^c)^2$ are scalars taken as the n-th entry of vectors X, M_i^c and the (n,n)-th entry of the matrix Σ_i^c, respectively. The univariate Gaussian distribution for scalar inputs x_n is:

$$\mathcal{N}(x_n|\mu_n^c, (\sigma_n^c)^2) = \frac{1}{\sqrt{2\pi}\,\sigma_n^c} \exp\left(-\frac{(x_n - \mu_n^c)^2}{2(\sigma_n^c)^2}\right) \tag{4}$$

When GMMs are used in an unsupervised manner, each component captures a specific cluster of the examined dataset. This makes them suitable for clustering problems. In classification problems, such as the ones targeted by the proposed architecture, multiple GMMs are used. In this case, for each class, a

single GMM is used for data clustering irrespectively to the other classes. The number of components (clusters) is chosen based on the complexity of the dataset's distribution

The posterior probability of class C_c given input vector X is then

$$p(C_c \mid X) \;=\; \frac{p(C_c)\, p(X \mid \lambda_c)}{p(X)}, \tag{4.5}$$

where $p(C_c)$ is the prior probability of class C_c, $p(X \mid \lambda_c)$ is the likelihood of X under the class-conditional model λ_c, and $p(X)$ is the evidence probability of X.

In practice, $p(C_c)$ is estimated from the training data, $p(X \mid \lambda_c)$ is computed from the GMM of class C_c, and $p(X)$ is common across all classes. Thus, the class decision is obtained by

$$y \;=\; \arg\max_{c\in[1,N_C]} \big\{\, p(C_c \mid X) \,\big\} \;=\; \arg\max_{c\in[1,N_C]} \big\{\, p(C_c)\, p(X \mid \lambda_c) \,\big\}. \tag{4.6}$$

Accordingly, classification with Gaussian Mixture Models can be interpreted as a Bayesian decision process, in which each class C_c is represented by a separate GMM λ_c. The winning class is the one maximizing the posterior probability for the given observation X.

Bayesian Model

A Naive Bayes classifier applies Bayes' theorem assuming feature independence. Using Bayes' rule, the posterior probability of class C_k given input $\mathbf{X}$ is

$$p(C_k \mid \mathbf{X}) \;=\; \frac{p(C_k)\, p(\mathbf{X} \mid C_k)}{p(\mathbf{X})}. \tag{4.17}$$

Here, $p(C_k)$ is the class prior, $p(\mathbf{X})$ is the evidence, and $p(\mathbf{X} \mid C_k)$ is the class-conditional likelihood. Under a Gaussian PDF with a diagonal covariance (independent features), the likelihood factorizes as

$$p(\mathbf{X} \mid C_k) \;=\; \prod_{n=1}^{N} \frac{1}{\sqrt{(2\pi)\,\sigma_{kn}^2}}\, \exp\!\left(-\frac{1}{2}\,\frac{(x_n-\mu_{kn})^2}{\sigma_{kn}^2}\right), \tag{4.18}$$

where N is the number of features, and μ_{kn}, σ_{kn}^2 are the mean and variance of the n-th feature for class k.

Since $p(\mathbf{X})$ is common across classes, the decision reduces to

$$y \;=\; \arg\max_{k\in[1,K]} \big\{\, p(C_k \mid \mathbf{X}) \,\big\} \;=\; \arg\max_{k\in[1,K]} \big\{\, p(C_k)\, p(\mathbf{X} \mid C_k) \,\big\}. \tag{4.19}$$

Architecture and Implementation

The top level of the architecture is shown in the figure below.

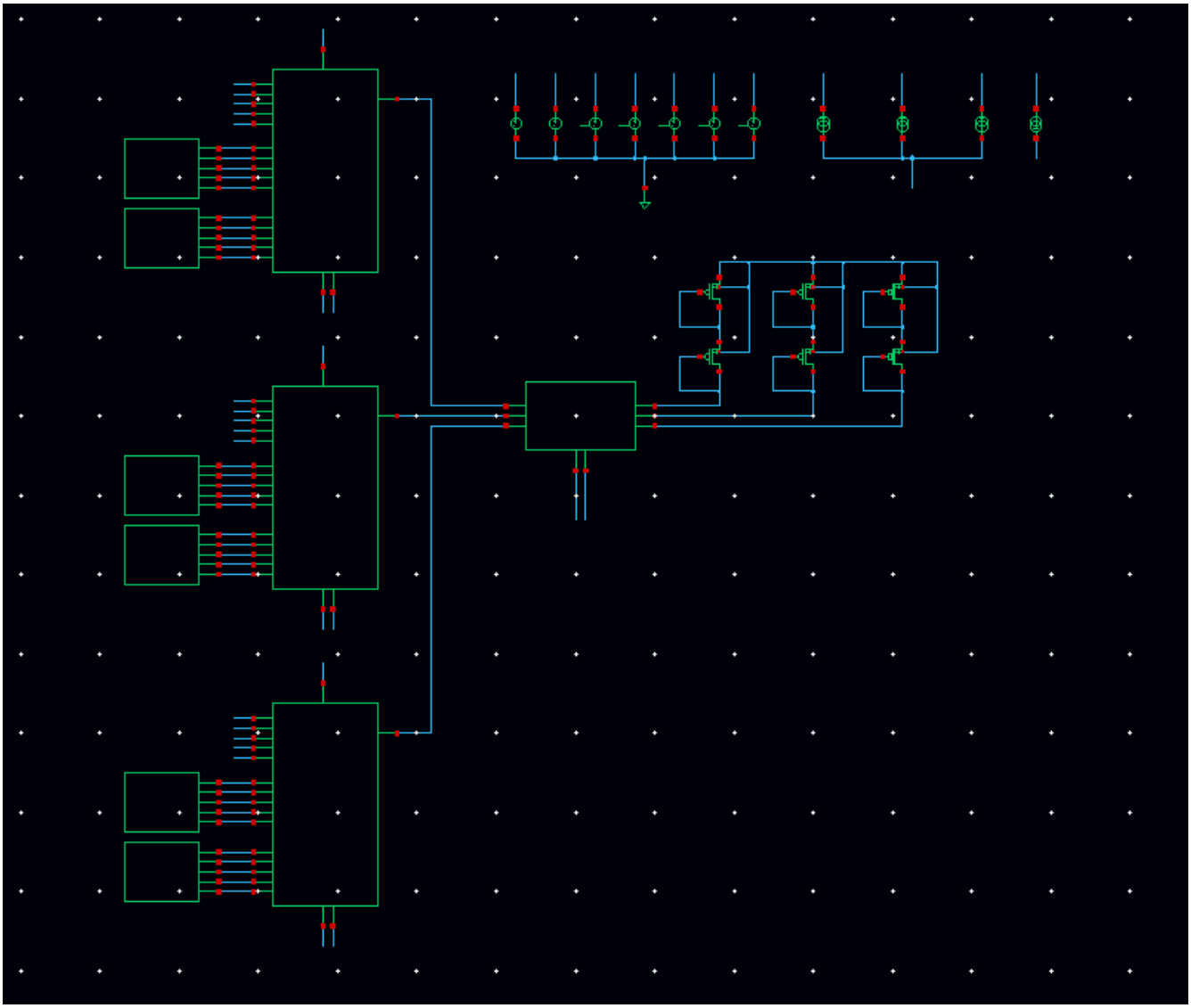


Figure 1: Top Level of the Analog Classifier Architecture

This specific classifier is designed to classify 5D input vectors into 3 classes. Specifically, it is made to work with the Thyroid dataset, which has 5 features and 3 classes. Shown below are the rest of the building blocks of the architecture.

- **Magnified View of the Top Level with visible labels:**

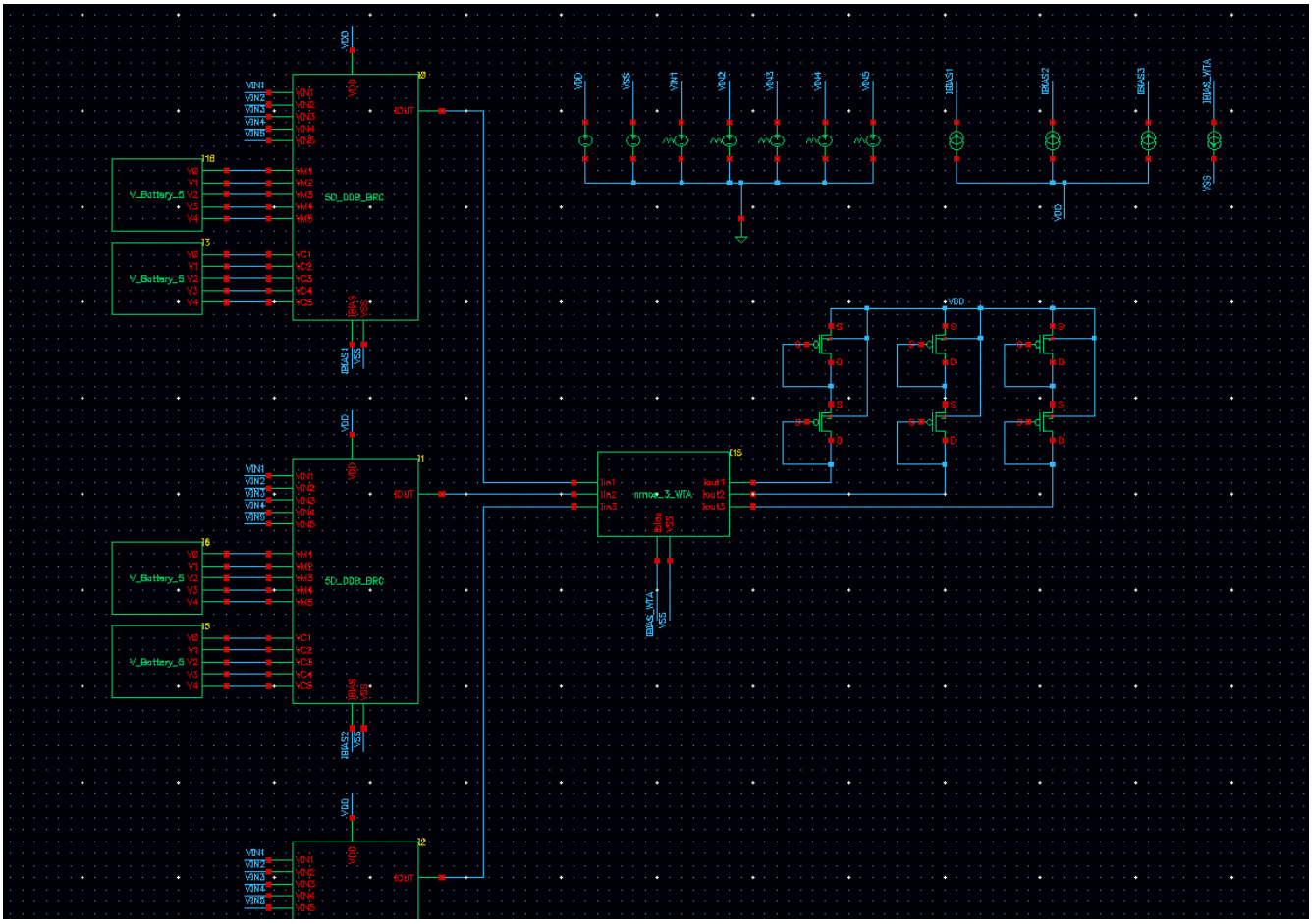


Figure 2: Top Level of the Analog Classifier Architecture with visible labels

- **5D_DDB_BRC:** This block consists of 5 Bumps connected in cascaded fashion.

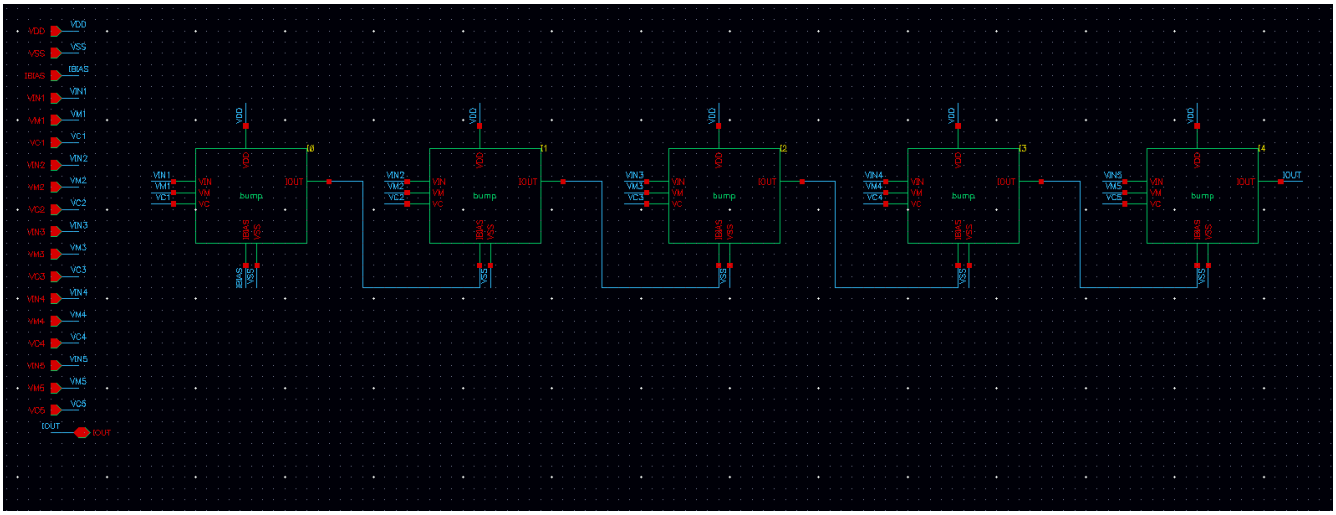


Figure 3: 5D_DDB_BRC Block

- **Bump:** This block is the bump circuit that we designed in the previous logbook entry.

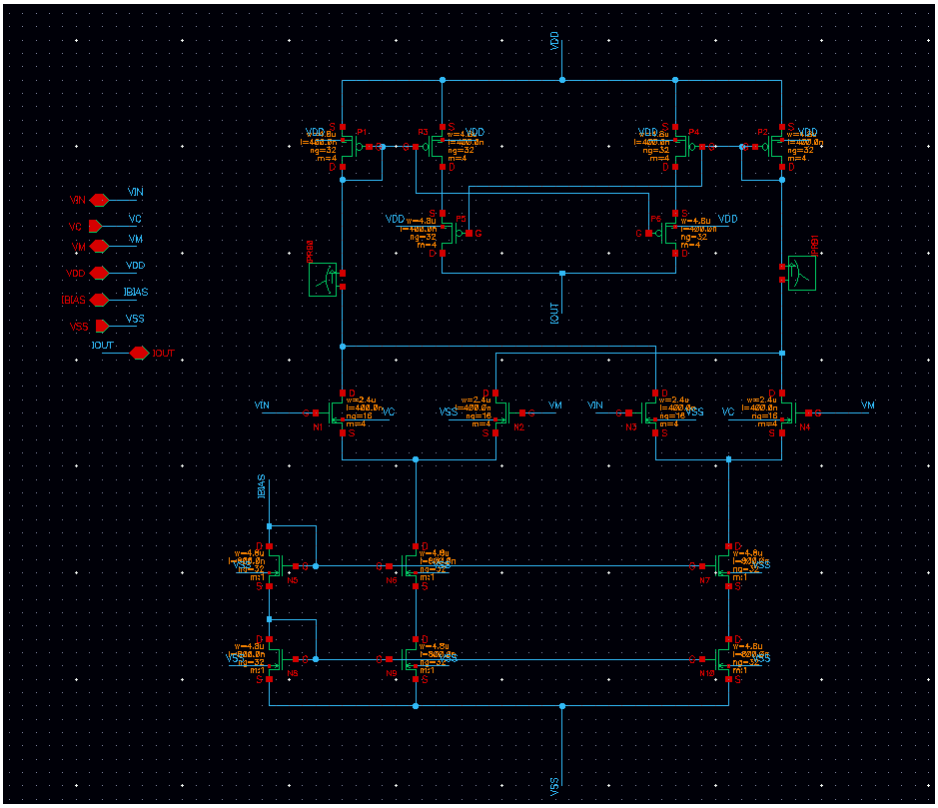


Figure 4: Bump Block

- **Lazarro WTA Circuit:** The outputs of the 3 GMM blocks are fed to a Lazarro WTA circuit, which selects the maximum output. Cascoded transistors are used as load to the WTA circuit, in order to increase the output resistance and thus the gain of the circuit.

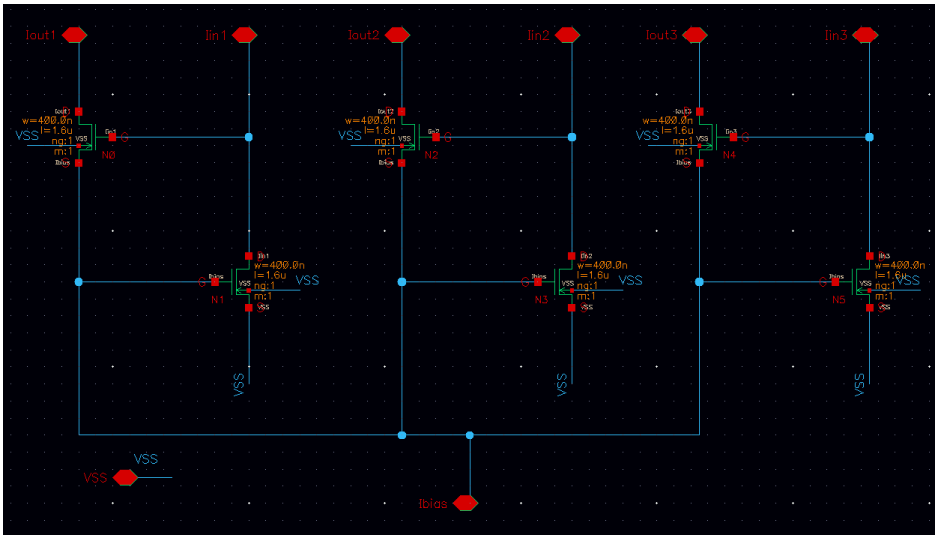


Figure 5: Lazarro WTA Block

- **'Battery'**: This block is used to simply group the voltage sources that are used as inputs to the circuit. Specifically, through training we will set the values of V_c and V_m.

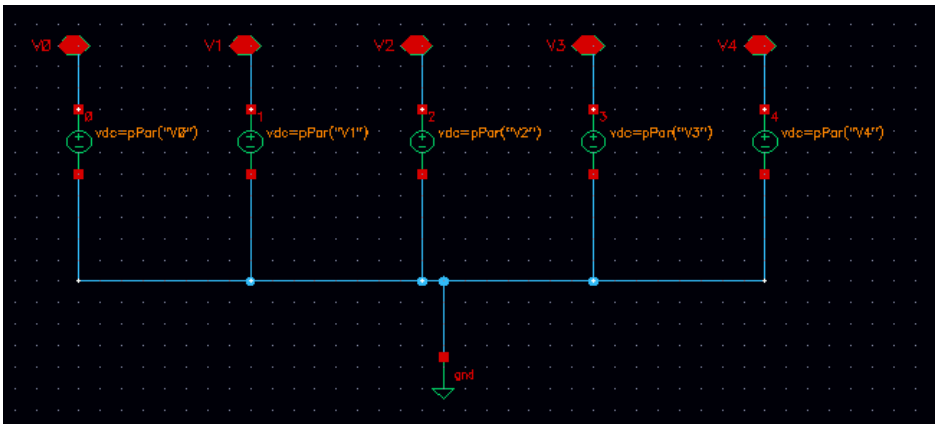


Figure 6: Battery Block

Code Implementation

Finally, we move on to the code implementation of the architecture. We follow the steps outlined in the first cell of the notebook Bayes_CLF and create a script.

Then we make the following changes to the code:

- We set the number of classes to 3 and the number of features to 5.
- We set all I_{bias} to 10nA.
- We set VDD-VSS to +0.3V and -0.3V respectively.
- We set the input voltage range to [-0.15V, 0.15V]. This value represents the V_{mean} tunability of the bump circuit we previously observed.
- We use the "VC_calc" script in order to get the polynomial coefficients that will set the V_c values.

Loading and executing the script for 10 iterations and then extracting the output currents, we get the following results:

	Iter 1	**Iter 2**	**Iter 3**	**Iter 4**	**Iter 5**	**Iter 6**	**Iter 7**	**Iter 8**	**Iter 9**	**Iter 10**
Hardware	0.6154	0.4923	0.6308	0.6000	0.4923	0.6154	0.5692	0.5538	0.5692	0.5385
Software	0.9538	0.9692	0.9538	0.9846	0.9846	0.9385	0.9538	0.9231	0.9385	0.9385

Table 1: Comparison of Hardware vs. Software values across iterations.

Clearly, the results are disappointing, as the hardware behavior is practically random. We will have to investigate the reasons behind this behavior in the next logbook entry.

6 August

After several simulations and adjustments, we figured out that the problem with the classifier was its speed. Specifically, in our code implemenation, we gave a "pulse" (meaning a new input) to the classifier every 10ns. However, because of the bump's large size we had a lot of parasitic capacitances and resistances, which made the bump slow. Thus, the classifier was not able to "catch" the input signal before it changed again, leading to completely random outputs. To solve this, we increased the time between two consecutive inputs to 50ns.

And we get the following results, shown in Table 1. We can see that the hardware results are now much

Hardware	**Software**
0.8769	0.9538
0.8923	0.9692
0.8769	0.9077
0.8923	0.9846
0.8308	0.9692
0.8923	0.9538
0.8923	0.9692
0.9538	0.9692
0.9538	0.9692
0.9231	0.9385
Average	**Average**
0.8985	0.9584

Table 1: Hardware vs. Software results with averages for the TDD dataset.

closer to the software ones and are overall satisfactory.

We then create a new classifier for the IRIS dataset. The IRIS dataset uses 4 features and has 3 classes. Therefore, all we need to do is to connect 4 bumps in cascaded fashion, instead of the 5 we had for the previous dataset. The results are shown in Table 2. Again, the hardware results are satisfactory and cover our desired specifications.

Lastly, we tried to create a classifier for the Diabetes dataset, which has 11 features and 3 classes. This time, we connected 11 bumps in cascaded fashion.

Again, the hardware results, shown in Table 3, are satisfactory and cover our desired specifications.

Hardware	**Software**
0.8000	0.9111
0.7556	0.7778
0.7778	0.9111
0.7556	0.8889
0.7778	0.8444
0.7778	0.8889
0.8667	0.8667
0.7556	0.8667
0.8222	0.9333
0.8889	0.8667
0.7978	0.8750
Average	**Average**
0.8027	0.8786

Table 2: Hardware vs. Software results with averages for the IRIS dataset.

Hardware	**Software**
0.7450	0.9250
0.7900	0.9500
0.7725	0.9375
0.7550	0.9750
0.7900	0.9375
0.7450	0.9625
0.8750	0.9500
0.7875	0.9625
0.8400	0.9500
0.8250	0.9250
Average	**Average**
0.7925	0.9475

Table 3: Hardware vs. Software results with averages for the Diabetes dataset.