MADVLSI Final Project: A/D/A Bi-Directional Converter

James Jagielski, Kenta Burpee, and Jiayuan Liu

December 15, 2023

GitHub: https://github.com/James-Jagielski/bi-directional-A-D-A-converter

1 Overview of how the system works

The data converter we have created combines the functions of algorithmic digital-to-analog conversion and successive-approximation analog-to-digital conversion in a single circuit that provides bits serially in time, most-significant bit first. The overall architecture uses a comparator, two operational transconductance amplifiers and storage capacitors. The system utilizes an enable pin to switch between digital to analog (DAC) and analog to digital (ADC) conversion.

The data conversion scheme and basic schematic of the system is based on the design described in the following paper: G. Cauwenberghs, "A Micropower CMOS Algorithmic A/D/A Converter," IEEE Transactions on Circuits and Systems I, vol. 42, no. 11, November 1995, pp. 913-919.

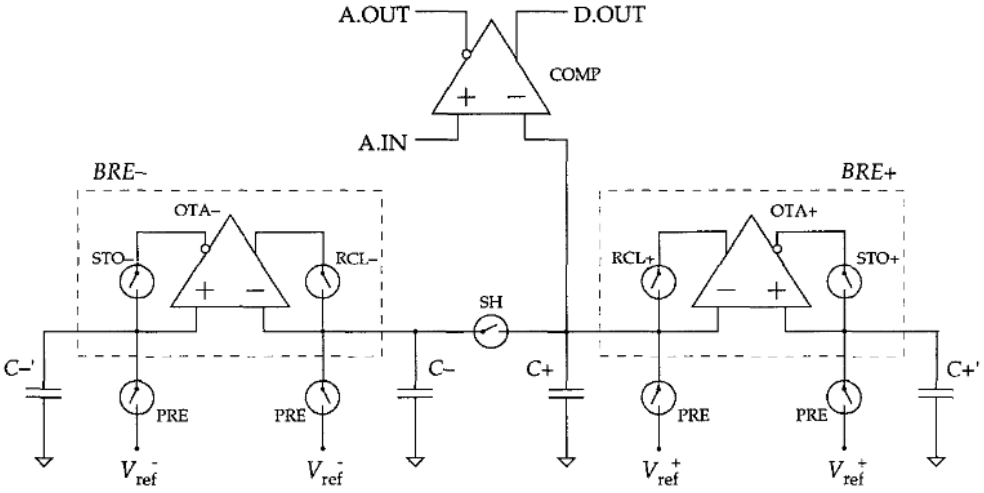


Figure 1: Bi-directional A/D/A converter block diagram (Cauwenberghs 1995).

The analog to digital process is as follows: Ain is the analog input to the system and is fed into the input of the comparator. The other input of the comparator is the voltage on the share capacitors. The OTAs will control the voltage on the share capacitors node checking each bit from the most significant to the least significant. Thus, for each bit the comparator simply measures if the analog voltage is higher or lower than the current bit voltage giving a 1 or a 0. The OTAs control the voltage on the capacitors by using store and recall capacitors to either set a new state or recall a previous state to share between the positive and negative sides.

The digital to analog process is as follows: We preset the voltage on the share capacitors and have a range from V_{ref-} to V_{ref+}. Each bit iteration we receive will determine if we will average with the negative or positive reference voltage. The loop will look like a tree diagram as each iteration will average up with a "1" and down for a "0". An example is shown below.

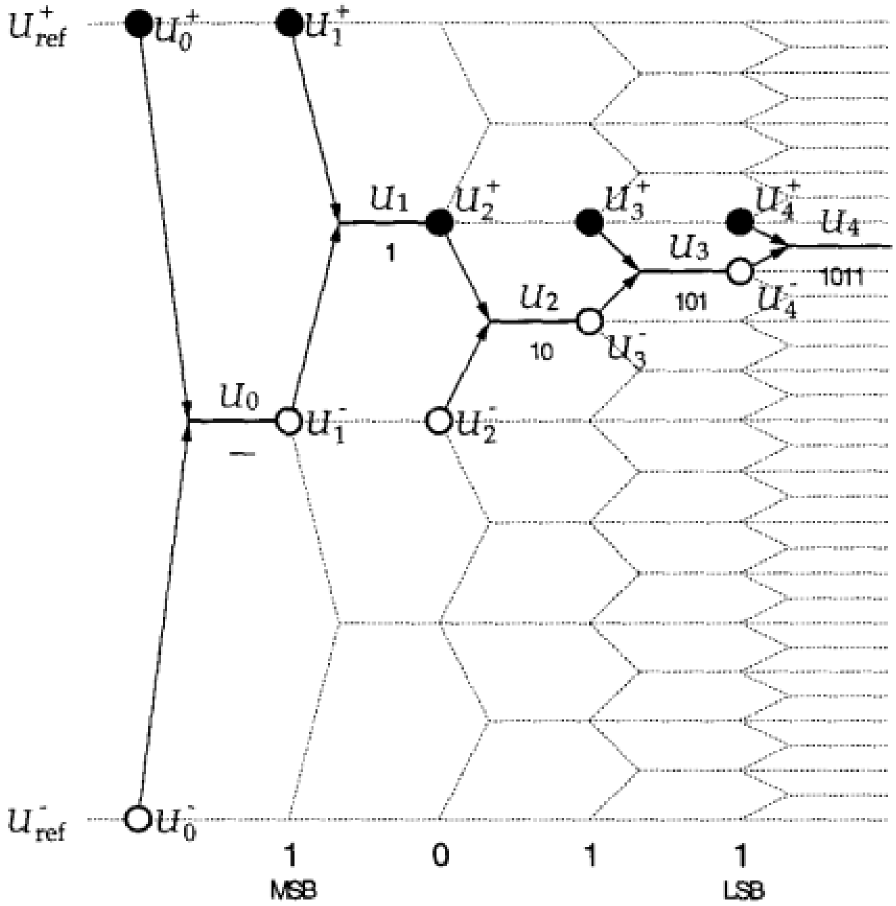
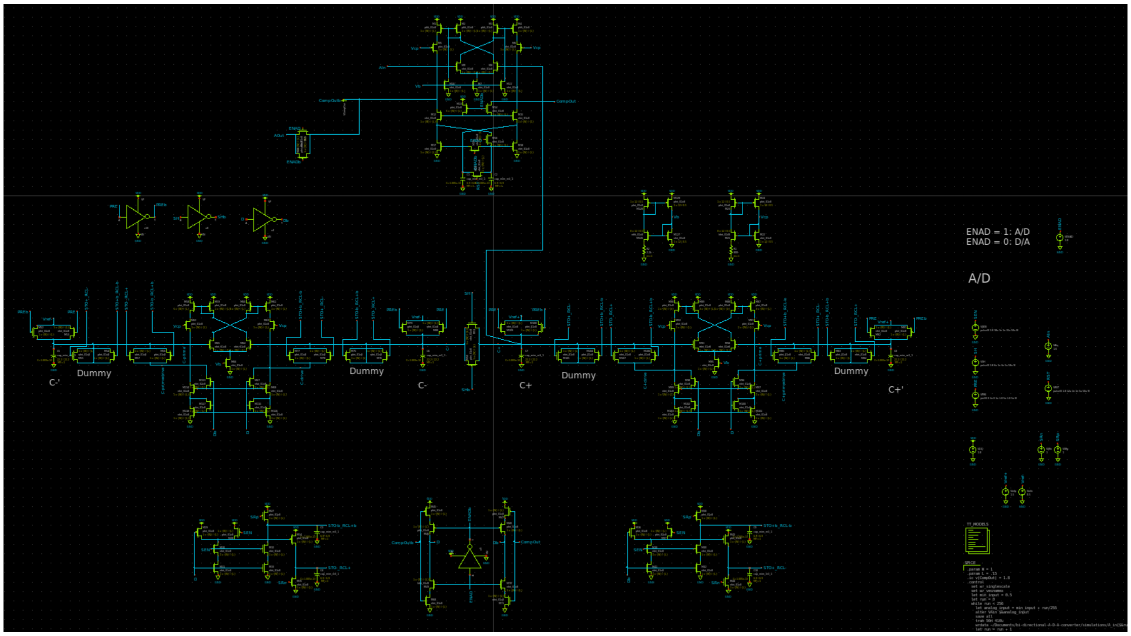


Figure 2: D/A example conversion (Cauwenberghs 1995).

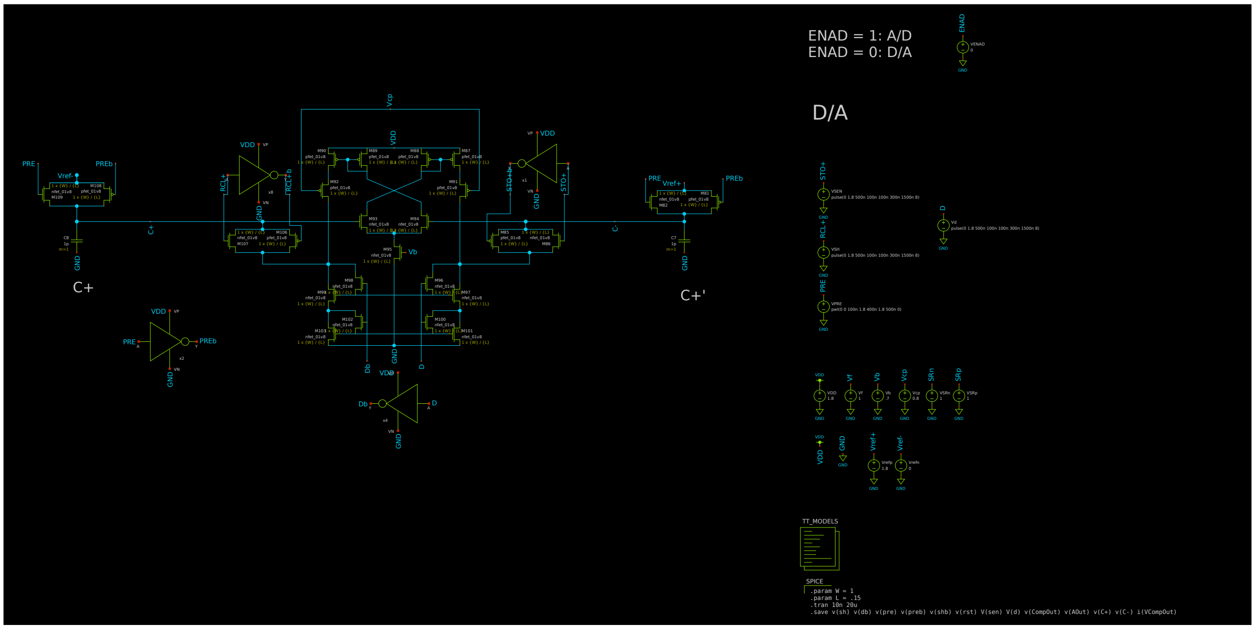
2 Schematics



```
.param W = 1
.param L = .15
.ic v(CompOut) = 1.8
.control
  set wr_singlescale
  set wr_vecnames
  let min_input = 0.5
  let run = 0
  while run < 256
    let analog_input = min_input + run/255
    alter VAin $&analog_input
    save all
    tran 50n 410u
    wrdata ~/Documents/bi-directional-A-D-A-converter/simulations/A_in{$&ru
    let run = run + 1
```

Figure 3: Full human readable schematic of the bi-directional A/D/A converter.

The top third of the schematic is the comparator, the middle third are the OTAs, and the bottom third is the digital circuitry. The comparator acts as the interface between the input/output signals for the converter. The OTAs control the share capacitors voltages. The digital circuitry controls the store and recall on the OTAs based on the data bits. On the right side of the schematic are the clocks and the simulation code.



```
.param W = 1
.param L = .15
.tran 10n 20u
.save v(sh) v(db) v(pre) v(preb) v(shb) v(rst) V(sen) V(d) v(CompOut) v(AOut) v(C+) v(C-) i(VCompOut)
```

Figure 4: Zoomed in single OTA for testing the store and recall system on the capacitors.

We were running into some issues when we were initially simulating the circuit, so we broke the system down to smaller pieces to test. We tested the comparator and the OTAs individually.

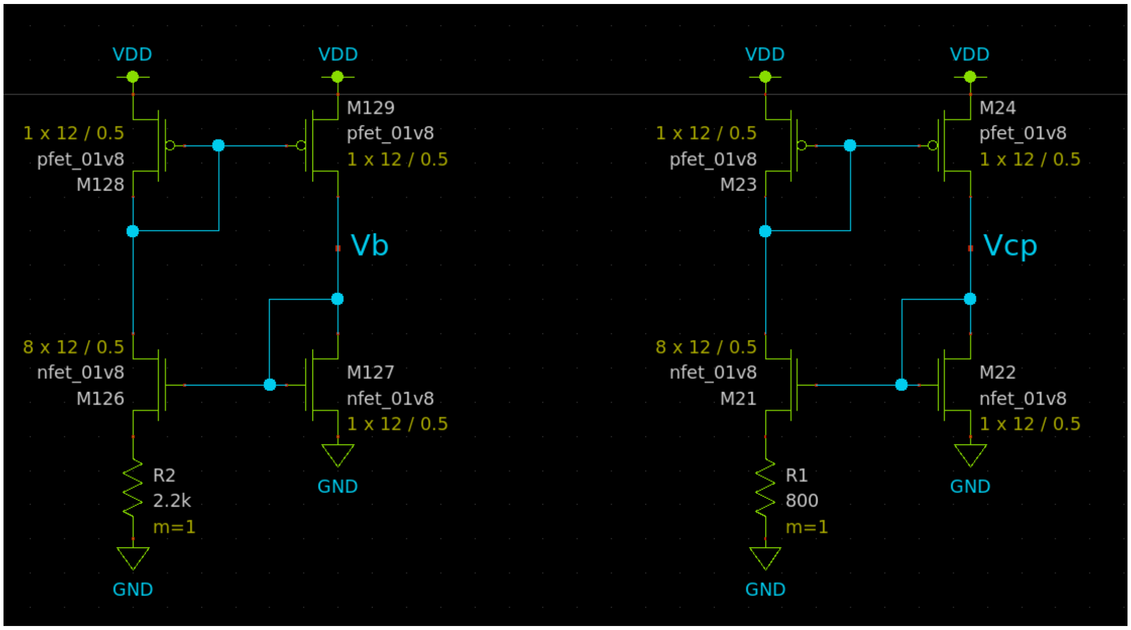


Figure 5: Bias voltage generators for Vb and Vcp.

We initially started with the voltage bias generators we used during class, but the monte carlo simulations we screwing with the voltages which threw the entire results off. We found the current bias generator created the most stable voltages and we changed the bias values by changing the resistor sizes in the circuit.

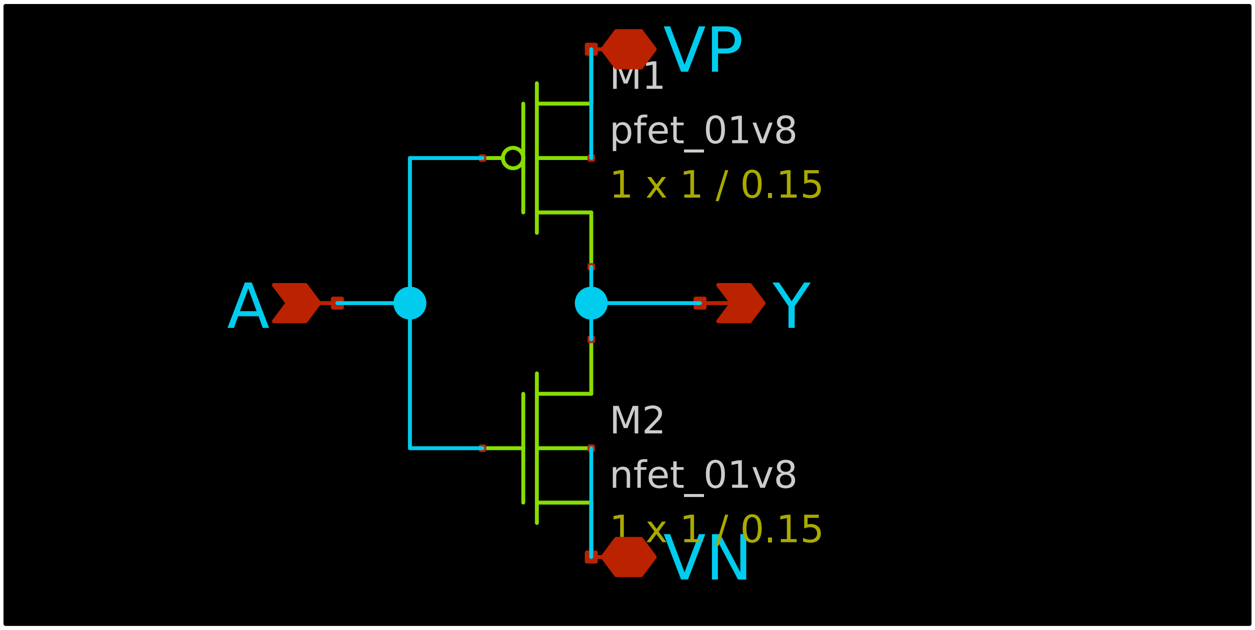


Figure 6: Inverter schematic.

2.1 Layout Driven Schematics

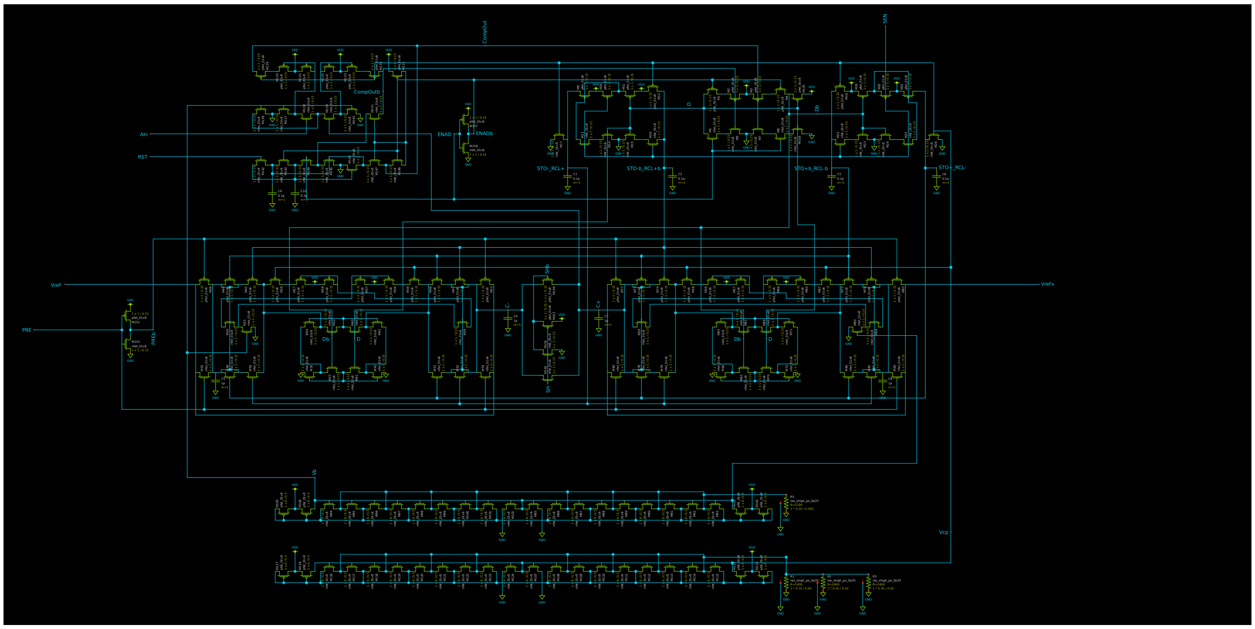


Figure 7: Layout Driven Schematic of the full system.

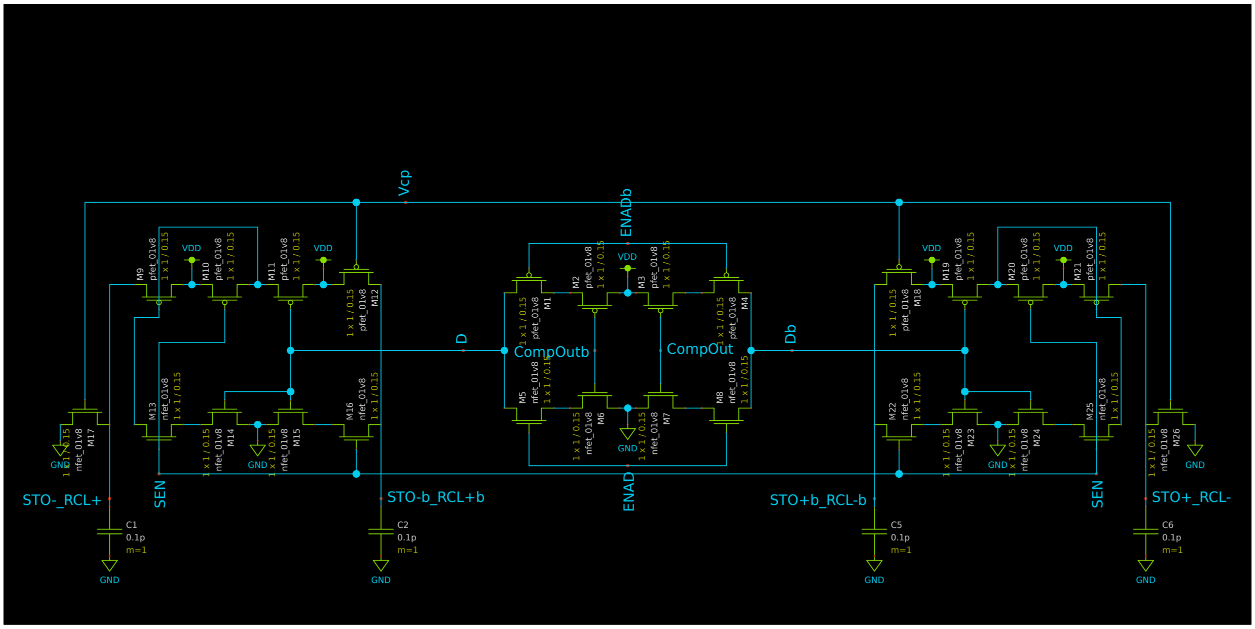


Figure 8: Layout Driven Schematic of the digital circuitry.

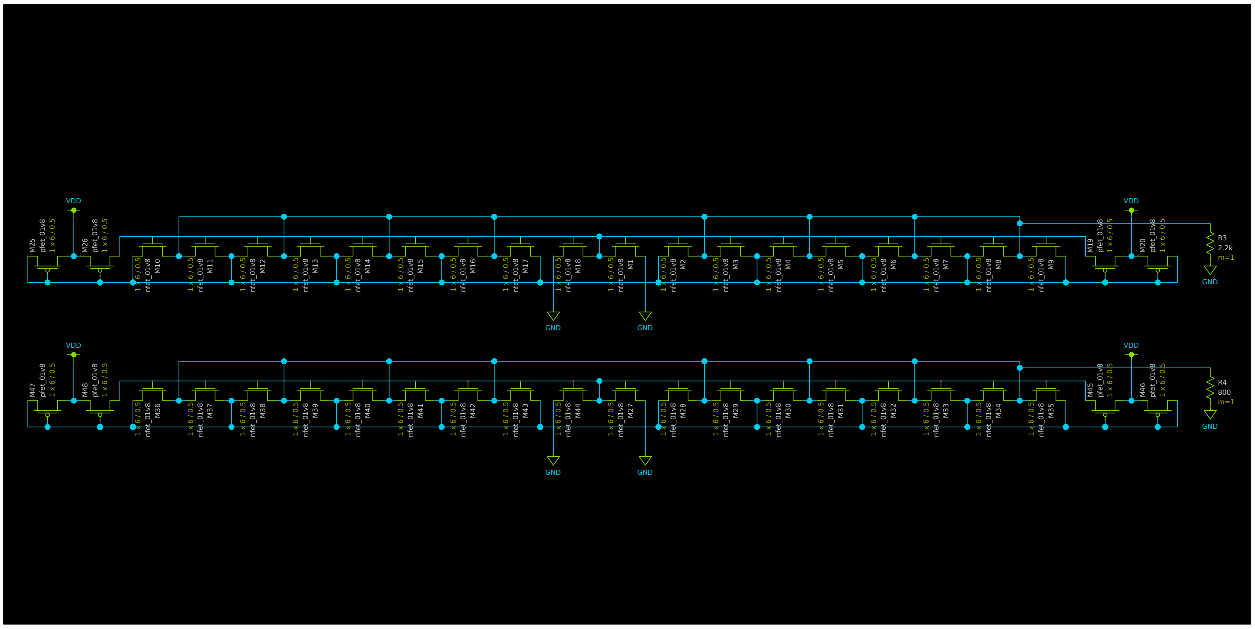


Figure 9: Layout Driven Schematic of the bias generators.

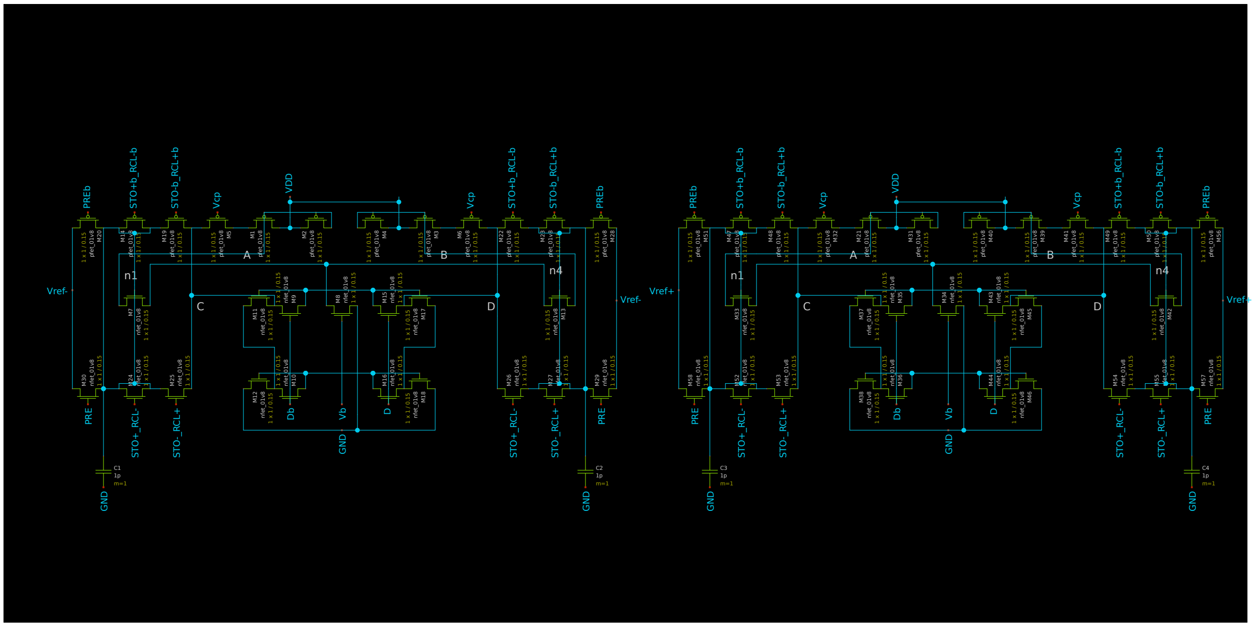


Figure 10: Layout Driven Schematic of the two operational transconductance amplifiers.

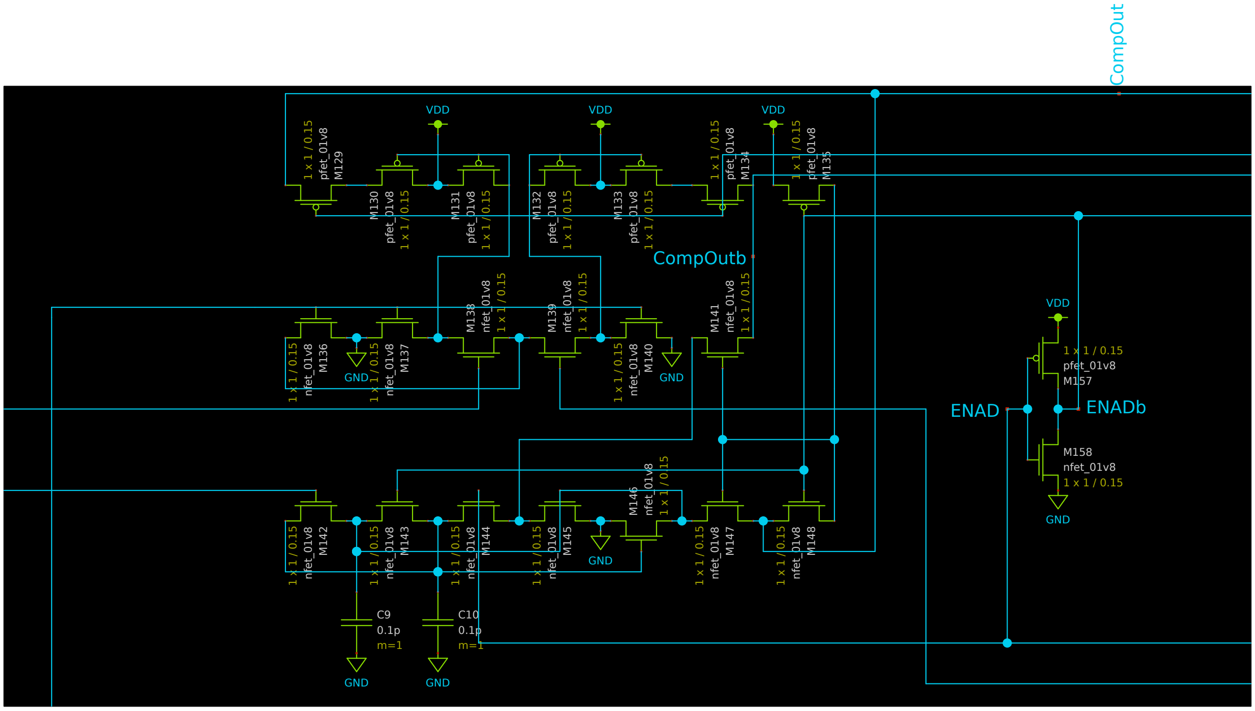


Figure 11: Layout Driven Schematic of the comparator.

3 Simulations

3.1 DAC

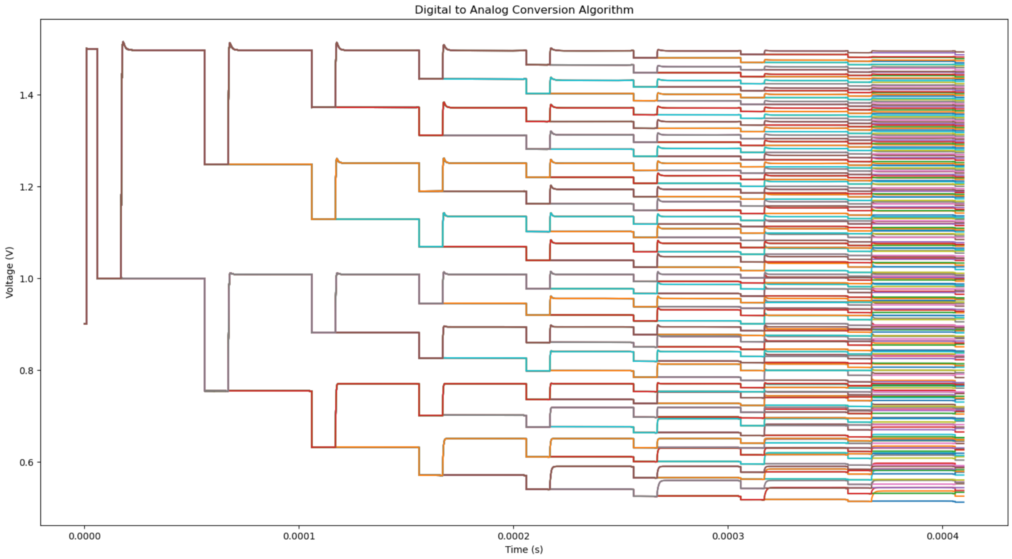


Figure 12: 8-bit digital to analog conversion for all 256 possible digital inputs.

3.1.1 Standard Operation

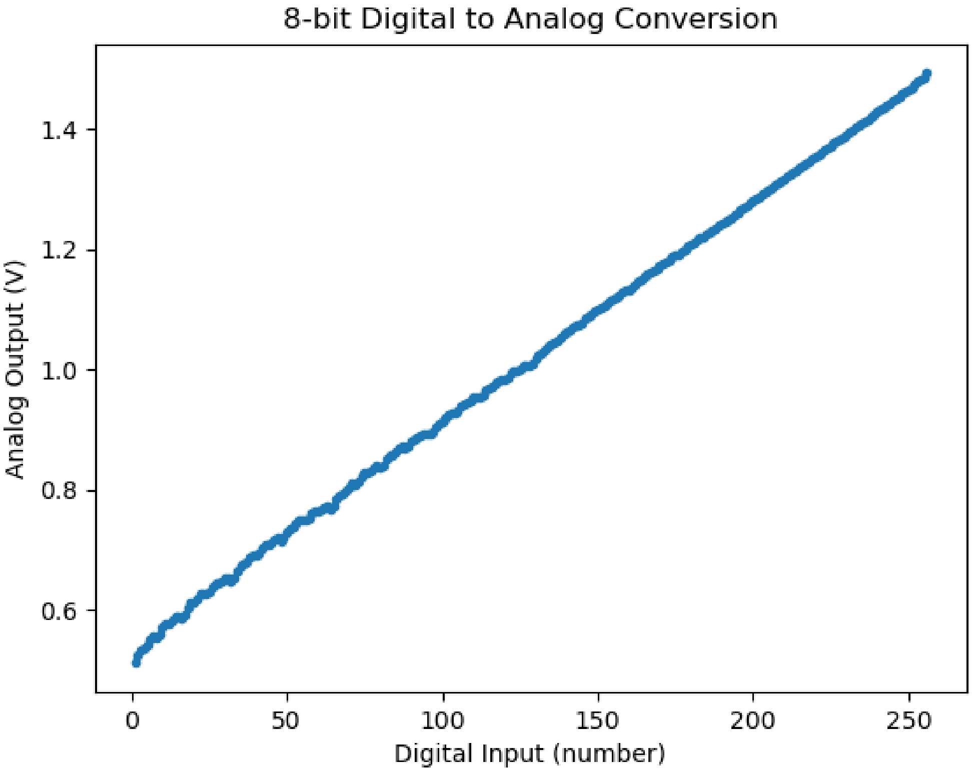


Figure 13: Digital to Analog conversion.

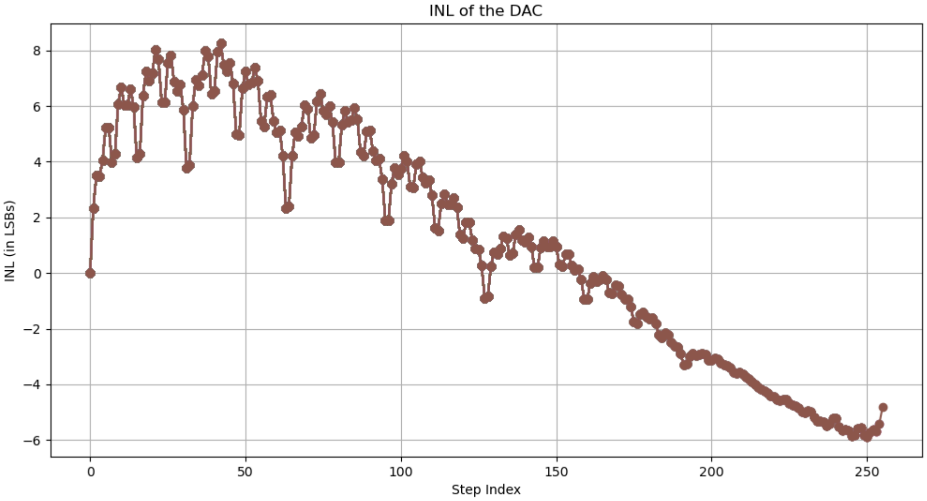


Figure 14: Digital to Analog integral nonlinearity.

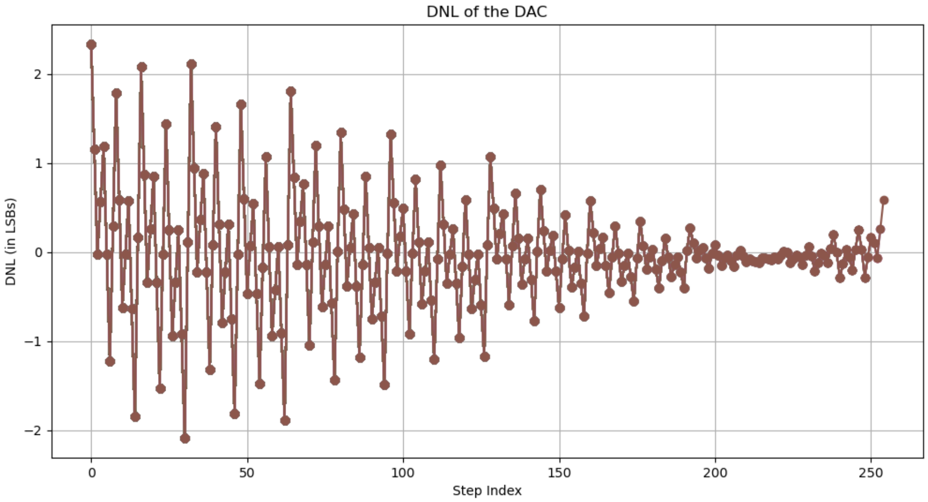


Figure 15: Digital to Analog differential nonlinearity.

3.1.2 Monte Carlo Simulations

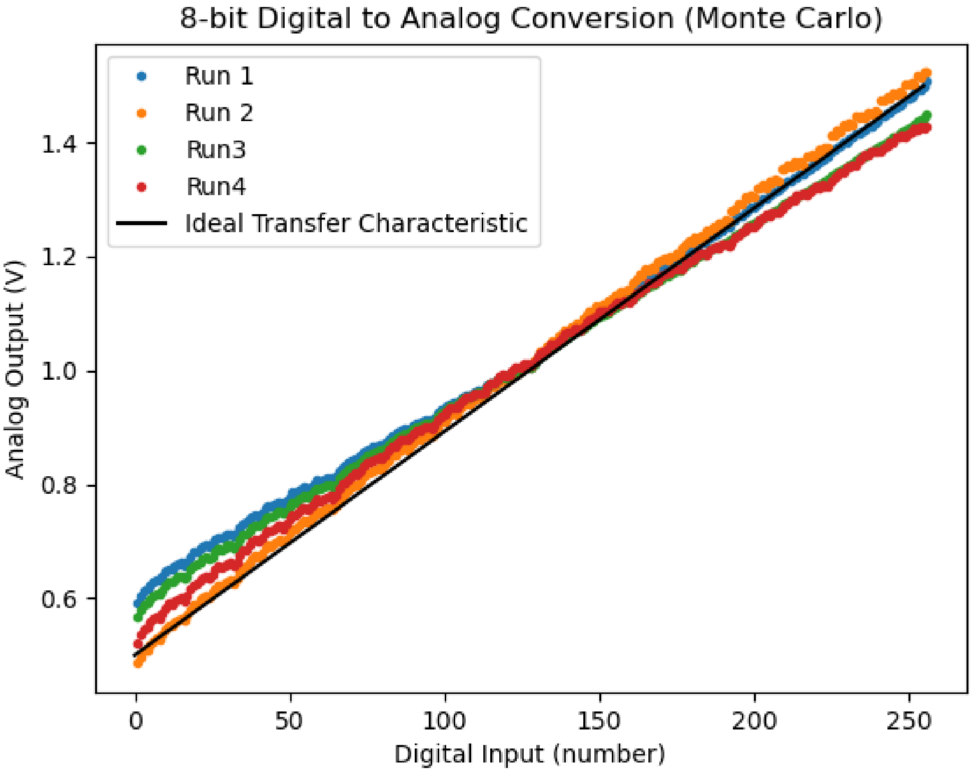


Figure 16: Digital to Analog conversion monte carlo.

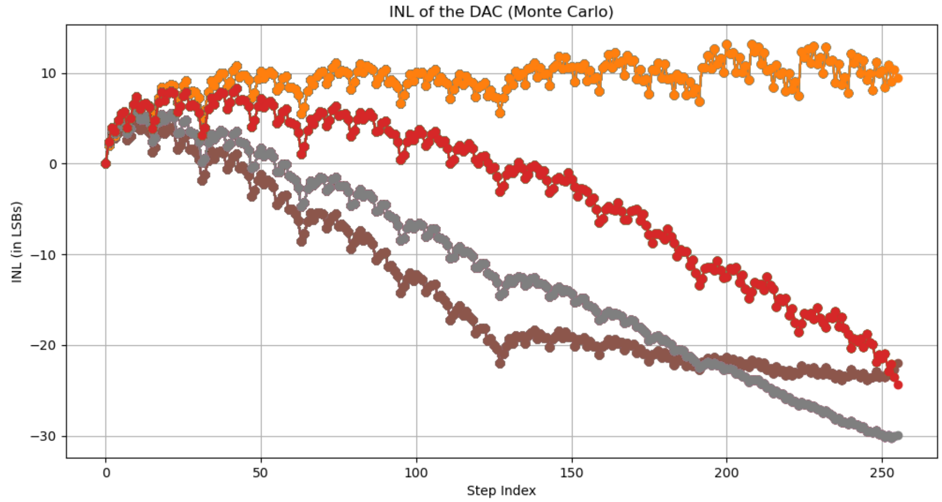


Figure 17: Digital to Analog integral nonlinearity.

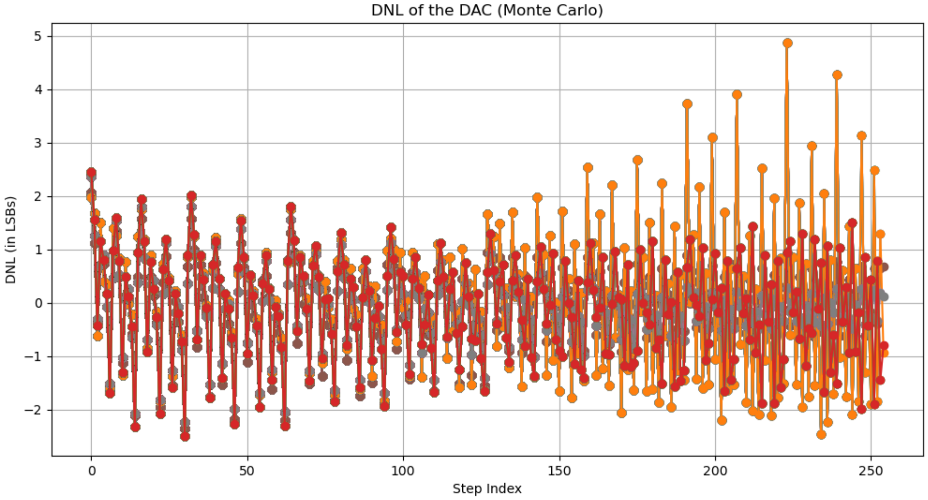


Figure 18: Digital to Analog differential nonlinearity.

3.2 ADC

3.2.1 Standard Operation

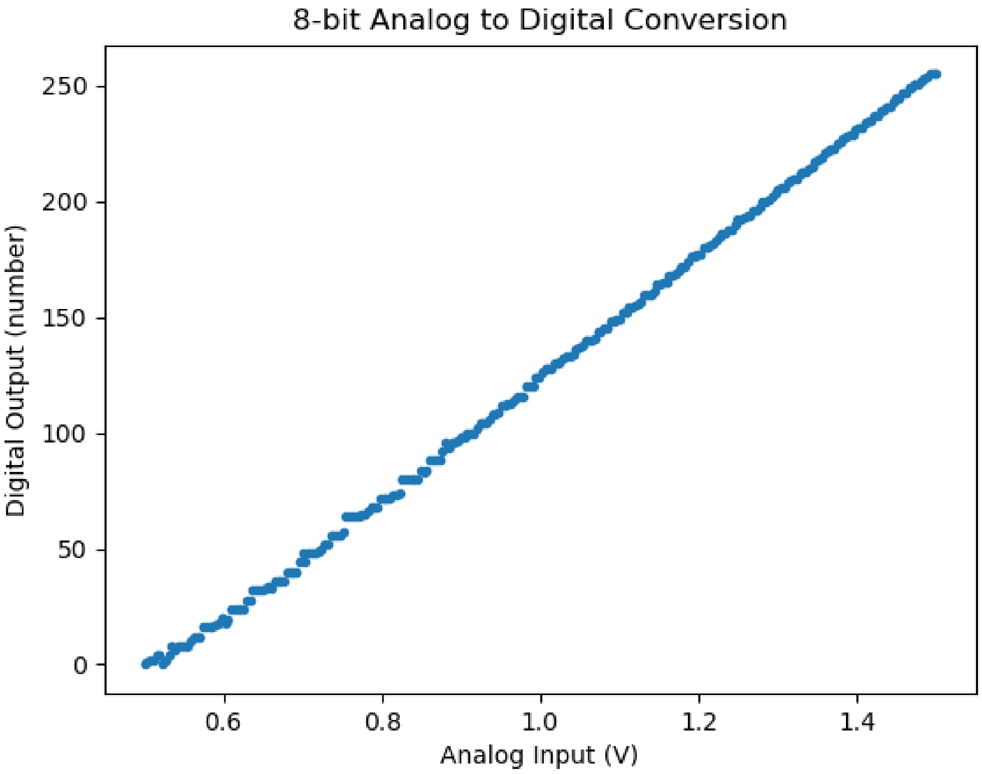


Figure 19: Analog to digital conversion.

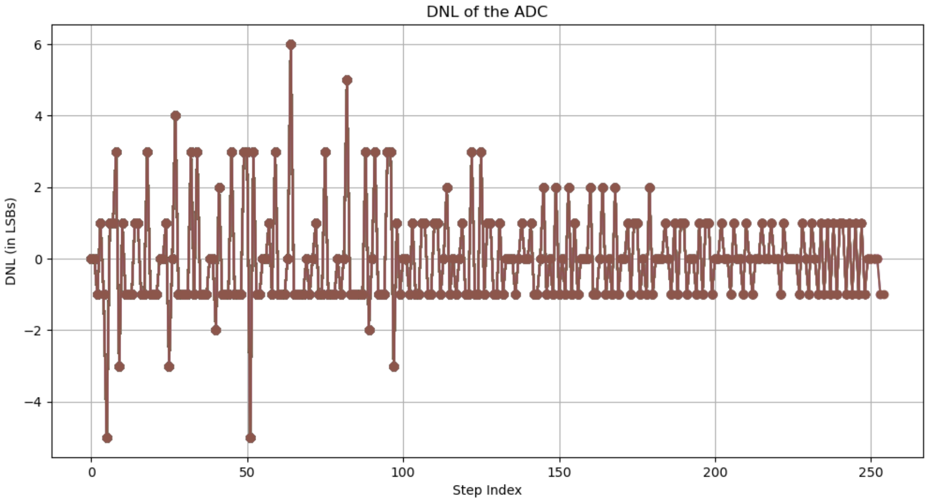


Figure 20: Analog to digital differential nonlinearity.

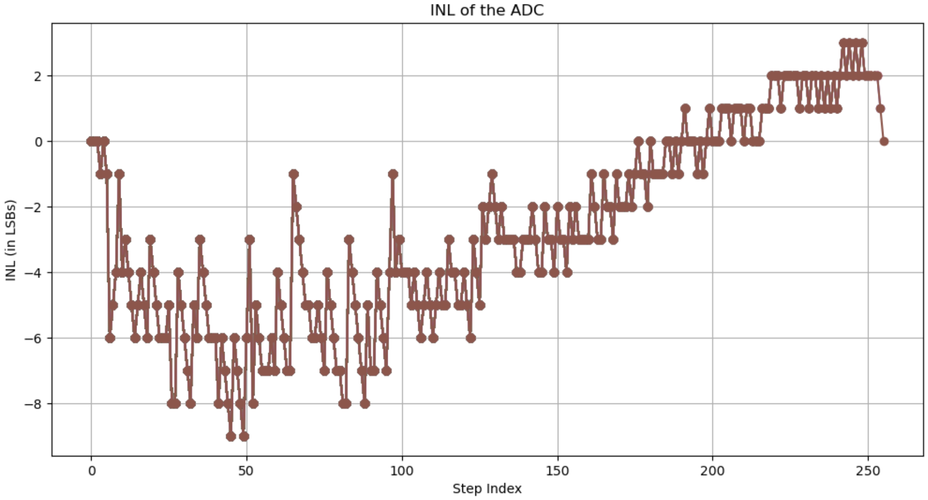


Figure 21: Analog to digital integral nonlinearity.

3.2.2 Monte Carlo Simulations

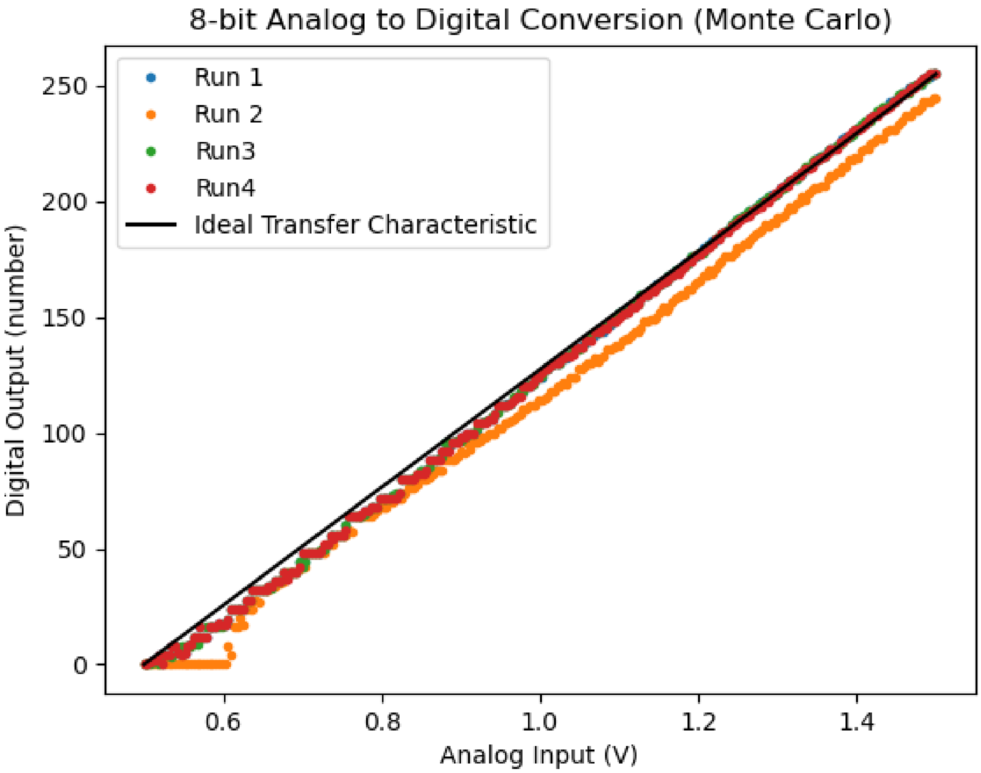


Figure 22: Analog to digital conversion monte carlo.

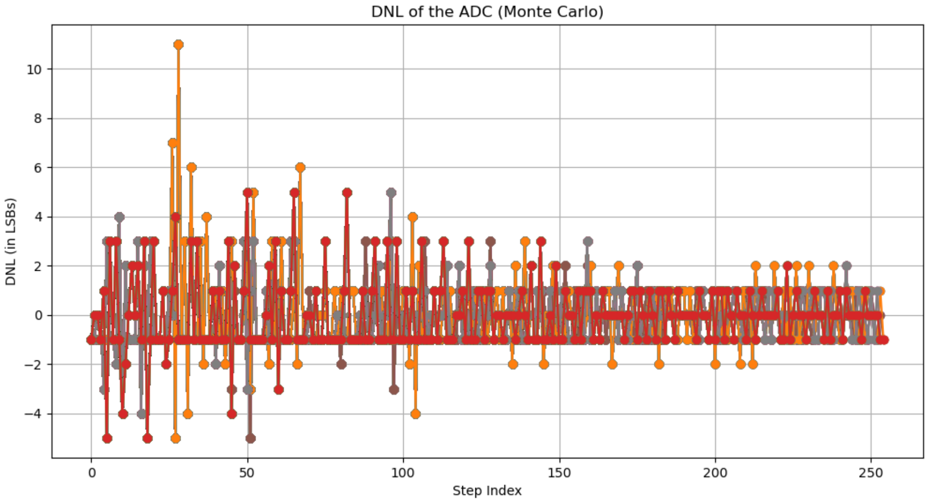


Figure 23: Analog to digital differential nonlinearity.

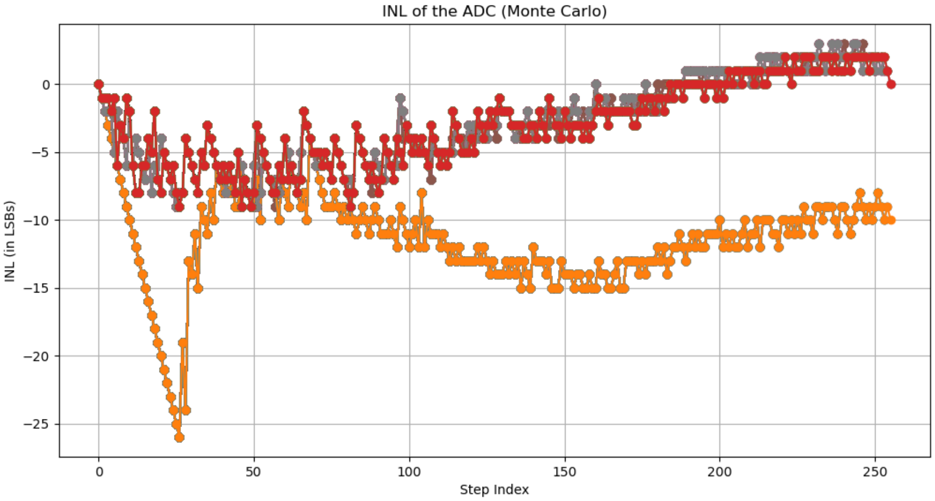


Figure 24: Analog to digital integral nonlinearity.

Layout

Due to the design of the converter typical analog layout best practices like same surround, best matching, and common centroid don't need to be considered because the circuit handles the mismatch. When we were sizing the transistors we found the minimum size of $1\mu m \times 0.15\mu m$ yielded the best results when it came to INL and DNL simulations.

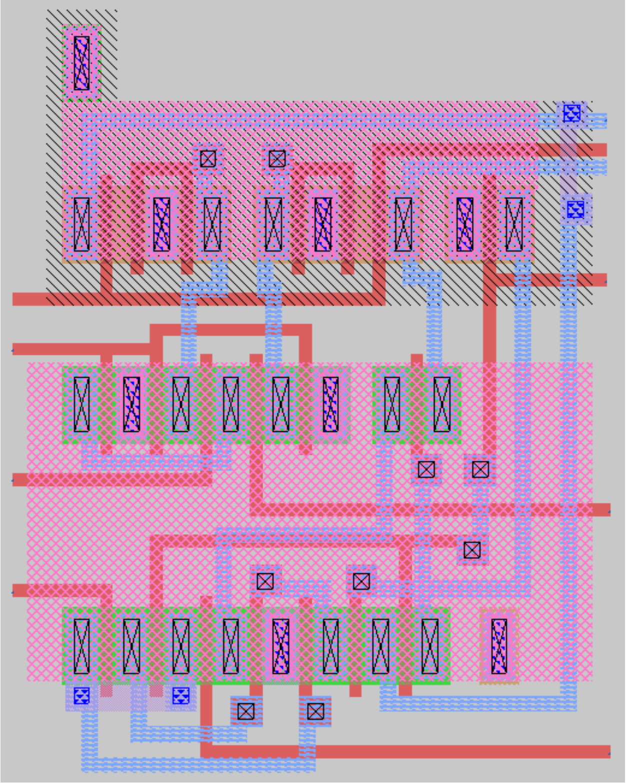


Figure 25: Layout of the comparator.

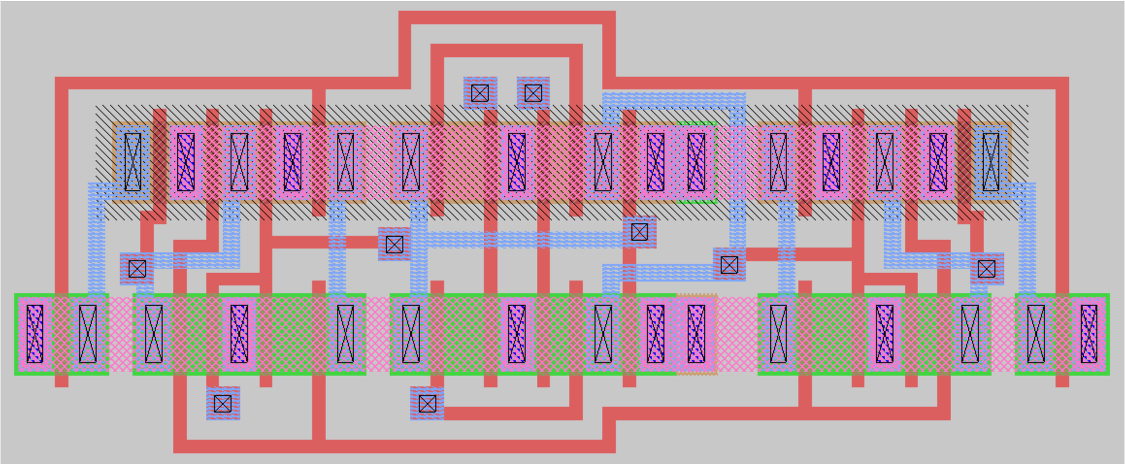


Figure 26: Layout of the digital circuitry.

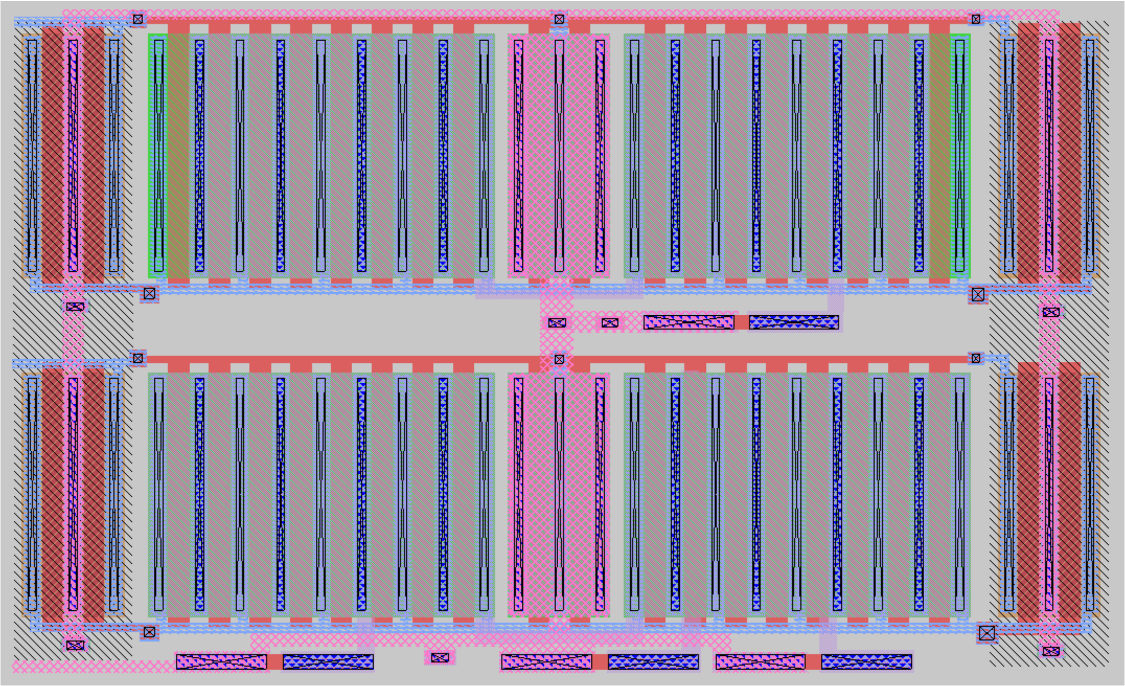


Figure 27: Layout of the bias voltage generators.

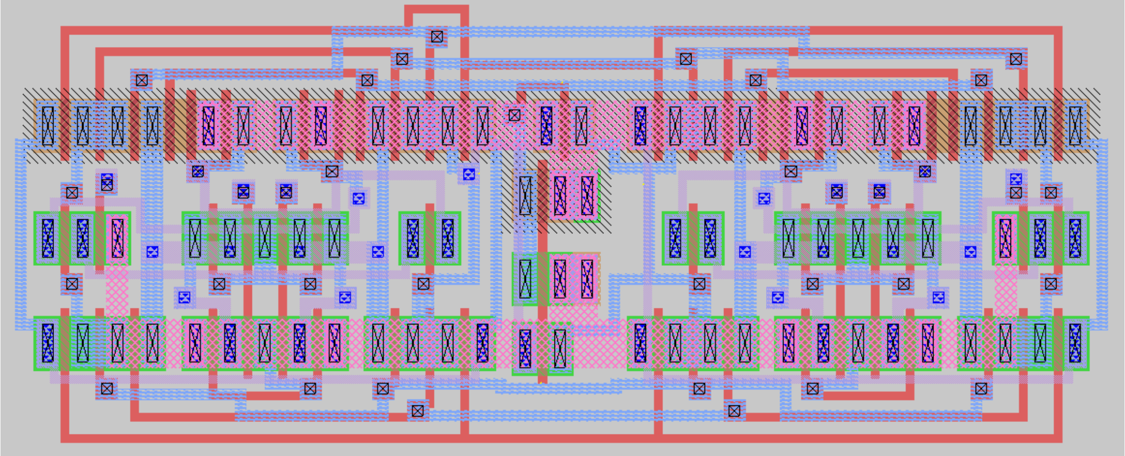


Figure 28: Layout of the OTAs.

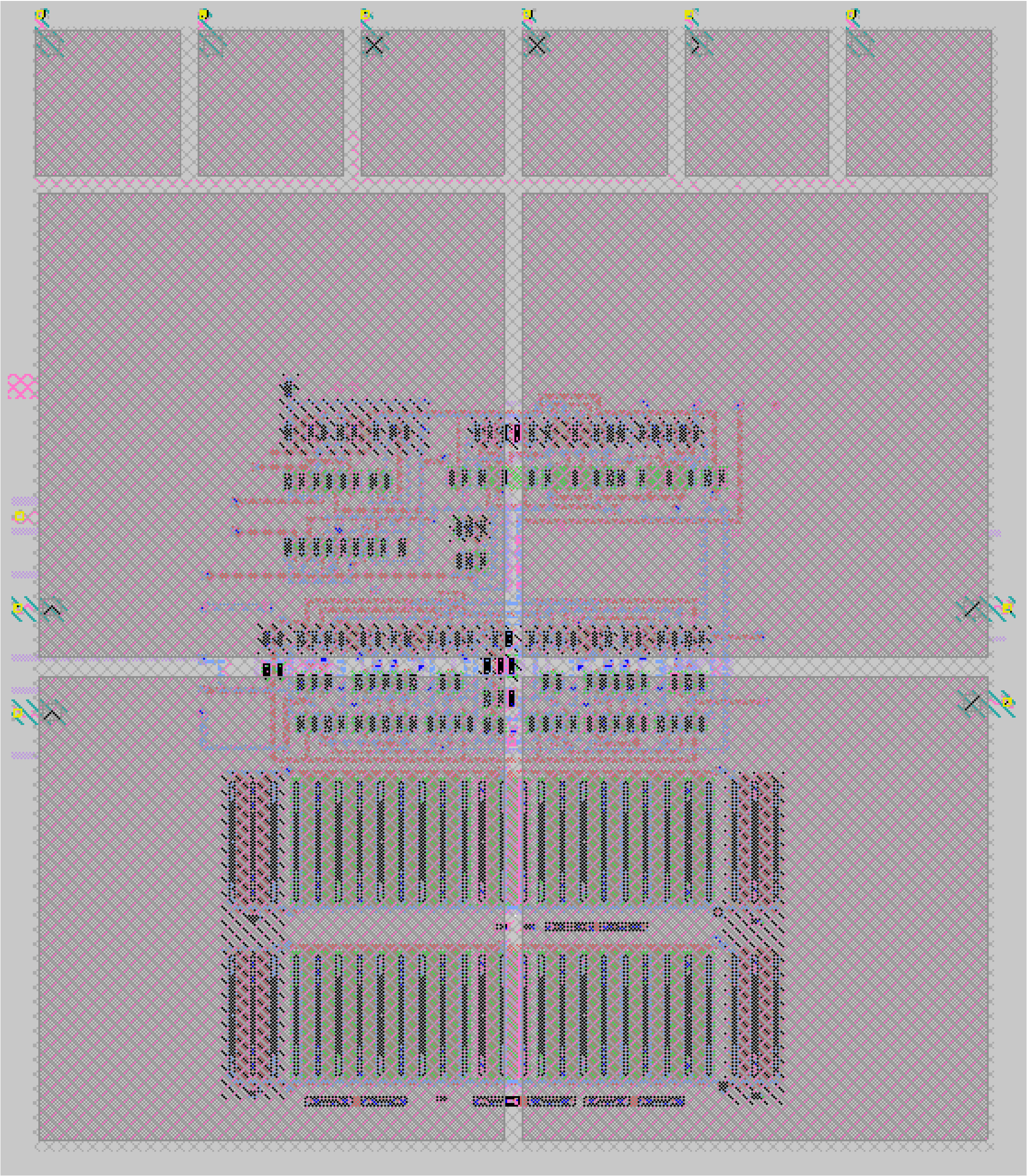
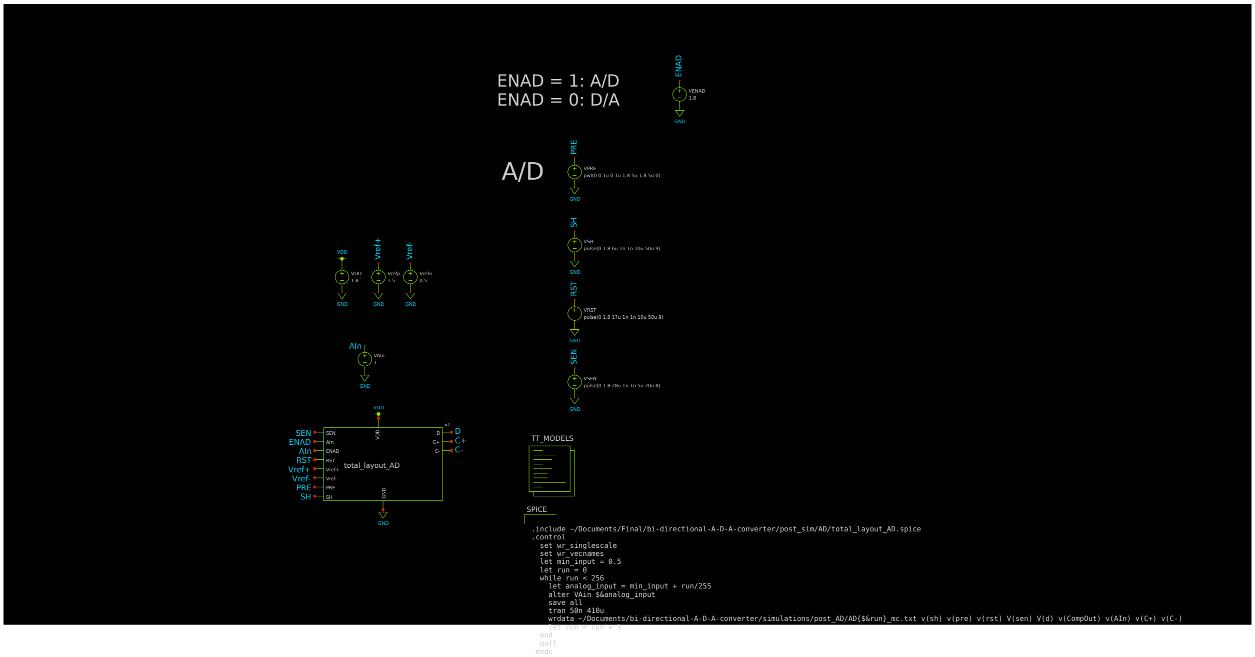


Figure 29: Full layout of the entire system.

Post-Layout Simulations

Although the LVS comaprison showed that the layout-driven schematic and layout of the entire system matched uniquely, the post-layout simulation results did not reflect the simulation results obtained from the layout-driven schematic.



```
.include ~/Documents/Final/bi-directional-A-D-A-converter/post_sim/AD/total_layout_AD.spice
.control
  set wr_singlescale
  set wr_vecnames
  let min_input = 0.5
  let run = 0
  while run < 256
    let analog_input = min_input + run/255
    alter VAin $&analog_input
    save all
    tran 50n 410u
    wrdata ~/Documents/bi-directional-A-D-A-converter/simulations/post_AD/AD{$&run}_mc.txt v(sh) v(pre) v(rst) V(sen) V(d) v(CompOut) v(AIn) v(C+) v(C-)
    let run = run + 1
  end
  quit
.endc
```

Figure 30: Analog to digital post layout simulation setup.

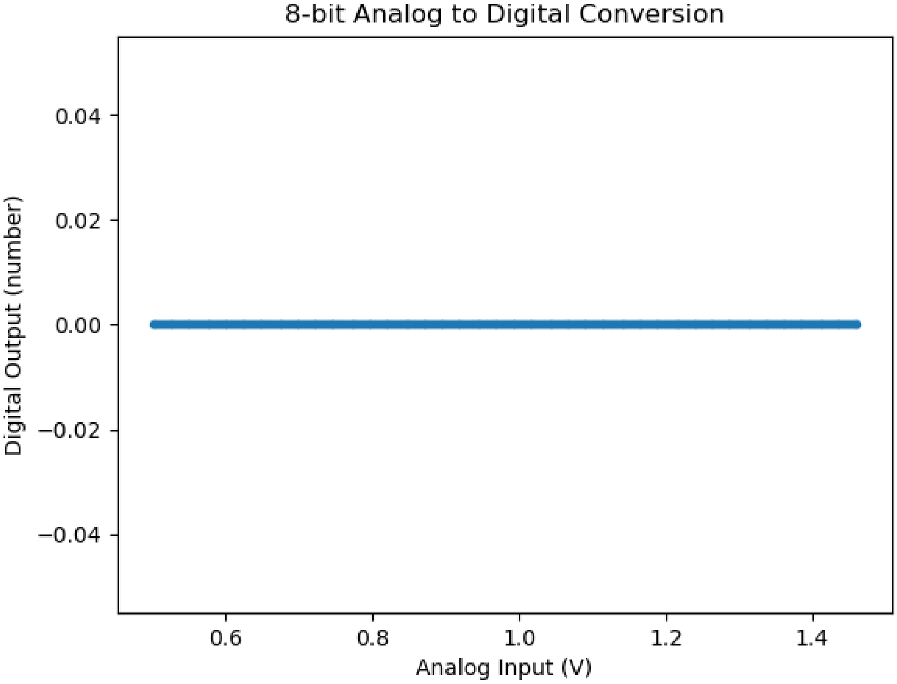


Figure 31: Full layout of the entire system.

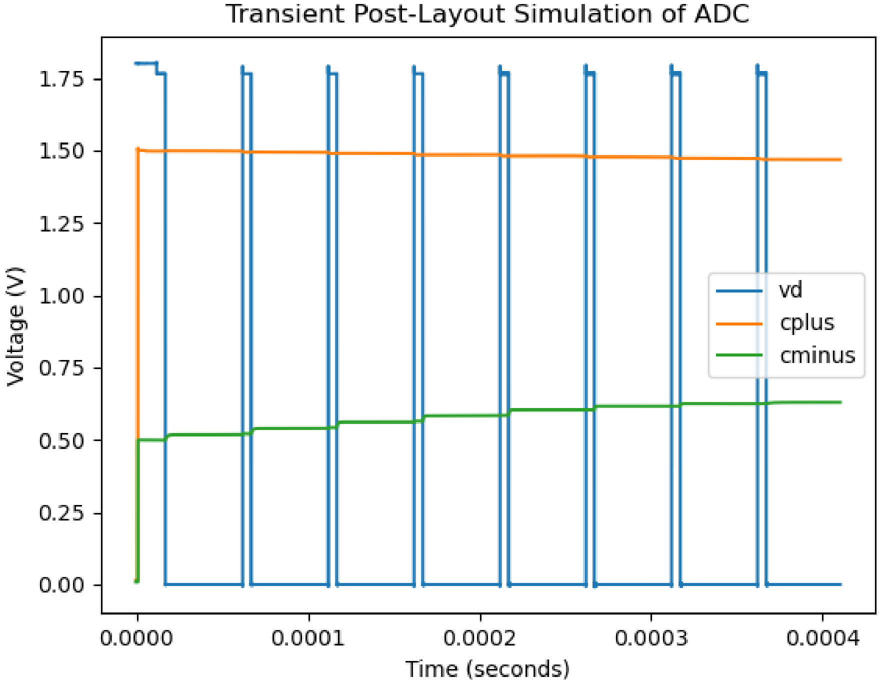


Figure 32: Full layout of the entire system.

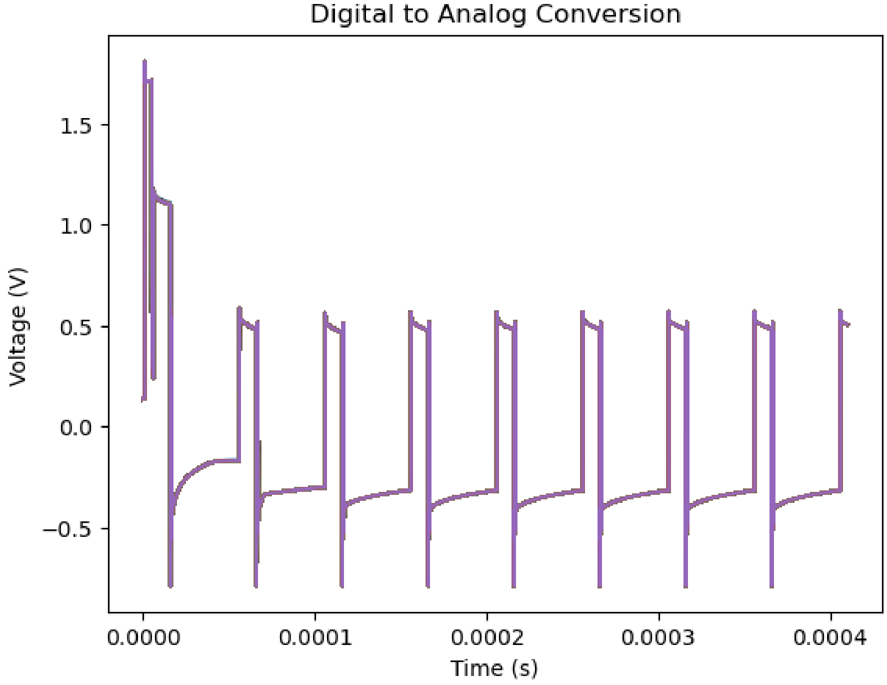


Figure 33: Full layout of the entire system.

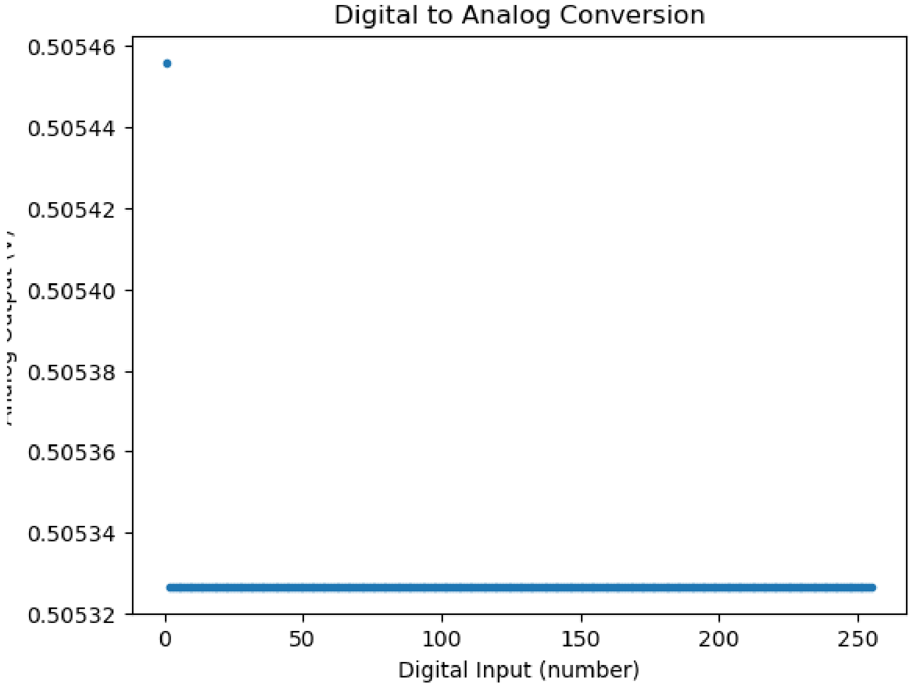


Figure 34: Full layout of the entire system.

4 LVS

The LVS results showed that the circuits matched uniquely.

```
Circuit 1 cell sky130_fd_pr__pfet_01v8 and Circuit 2 cell
 ↪  sky130_fd_pr__pfet_01v8 are black boxes.
Equate elements:  no current cell.
Device classes sky130_fd_pr__pfet_01v8 and sky130_fd_pr__pfet_01v8 are
 ↪  equivalent.

Circuit 1 cell sky130_fd_pr__nfet_01v8 and Circuit 2 cell
 ↪  sky130_fd_pr__nfet_01v8 are black boxes.
Equate elements:  no current cell.
Device classes sky130_fd_pr__nfet_01v8 and sky130_fd_pr__nfet_01v8 are
 ↪  equivalent.

Circuit 1 cell sky130_fd_pr__res_xhigh_po_0p35 and Circuit 2 cell
 ↪  sky130_fd_pr__res_xhigh_po_0p35 are black boxes.
Equate elements:  no current cell.
Device classes sky130_fd_pr__res_xhigh_po_0p35 and
 ↪  sky130_fd_pr__res_xhigh_po_0p35 are equivalent.

Circuit 1 cell sky130_fd_pr__cap_mim_m3_1 and Circuit 2 cell
 ↪  sky130_fd_pr__cap_mim_m3_1 are black boxes.
Equate elements:  no current cell.
Device classes sky130_fd_pr__cap_mim_m3_1 and sky130_fd_pr__cap_mim_m3_1 are
 ↪  equivalent.
Flattening unmatched subcell com_di in circuit total_layout.spice (1)(1
 ↪  instance)
Flattening unmatched subcell inverter in circuit total_layout.spice (1)(1
 ↪  instance)
Flattening unmatched subcell digital in circuit total_layout.spice (1)(1
 ↪  instance)
Flattening unmatched subcell comparator in circuit total_layout.spice (1)(1
 ↪  instance)
Flattening unmatched subcell middle in circuit total_layout.spice (1)(1
 ↪  instance)
Flattening unmatched subcell bias_cg in circuit total_layout.spice (1)(1
 ↪  instance)

Class lds_schematics_FINAL.spice (0):  Merged 38 parallel devices.
Class total_layout.spice (1):  Merged 38 parallel devices.
Subcircuit summary:
Circuit 1: lds_schematics_FINAL.spice      |Circuit 2: total_layout.spice
```

```
-------------------------------------------|----------------------------------
sky130_fd_pr__pfet_01v8 (56->52)           |sky130_fd_pr__pfet_01v8 (56->52)
sky130_fd_pr__nfet_01v8 (102->70)          |sky130_fd_pr__nfet_01v8
 ↪  (102->70)
sky130_fd_pr__res_xhigh_po_0p35 (4->2)     |sky130_fd_pr__res_xhigh_po_0p35
 ↪  (4->2)
sky130_fd_pr__cap_mim_m3_1 (10)            |sky130_fd_pr__cap_mim_m3_1 (10)
Number of devices: 134                     |Number of devices: 134
Number of nets: 72                         |Number of nets: 72
------------------------------------------------------------------------------
Netlists match uniquely.
Cells have no pins;  pin matching not needed.
Device classes lds_schematics_FINAL.spice and total_layout.spice are
 ↪  equivalent.

Final result: Circuits match uniquely.
.
```