Lab #3

Due: *Wednesday, May 7th, 11:59 PM (Design Review)*
Due: *Tuesday, May 20th, 5:00 PM (Lab Report)*

Your third project at Triton Industries, Inc. is to lay out the 9-bit capacitive DAC considering matching, synthesize the digital logic, lay out a non-overlapping clock generator and switches, and then integrate these blocks with your comparator. The schematic is shown in **Figure 1** and provided in the starter files ($PUBLIC/Lab_3). The SAR inputs should be on the left side of the block, and the digital inputs/outputs on the right side. All pins must be labeled and on metal 6. You are strongly encouraged, although not required, to label all internal nets. Your custom capacitors in the DAC should use metal shields, as appropriate. A sample capacitor layout is provided for you; you may use this as-is or modify it to your liking. The digital synthesis code (which you only need to modify to fit your design) is in the starter files. Your layout must be <690 µm × <350 µm! No exceptions!!!

You must work on this in teams of two. Each team must submit one project report as specified below. You are encouraged to discuss the design problem with other teams, but your implementation *must be unique*. Under **NO** circumstances should you exchange computer files with other teams; this would violate the student honor code and be submitted to the academic integrity office.

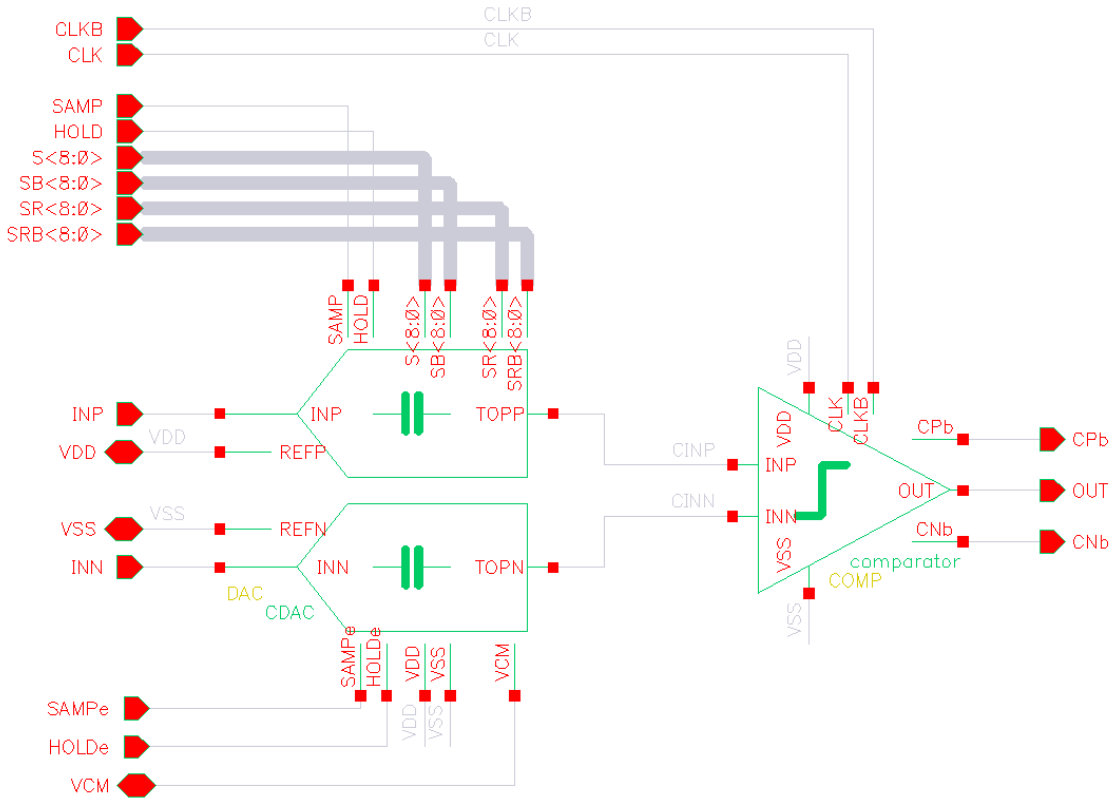


Figure 1 – SAR ADC Core (Schematic: SAR_Core)

Keep in mind that this project involves a great deal of plain old labor; it takes a significant amount of time to learn the tools, floorplan, wire up the design, run the necessary design checks, and clearly document your work. Don't delay getting started!

Project Report: Your team is required to prepare a single report. Reports must be submitted online through Canvas. *We will not grant any extensions.* Reports are in the form of PowerPoint slides and must follow the following structure:
Slide 0: Cover page. Indicate the names of the team members, including PID and email.
Slide 1: Schematic annotated with dummy devices. (White background only)
Slide 2: DAC layout placement strategy.
Slide 3: Picture(s) of your DAC layout. Annotate the devices/blocks and the dimensions of the block (using rulers). Make sure to have at least one top-level block and clearly show the unit cell.
Slide 4: Summary table of parasitic coupling caps between B0-B9 to Top/Bottom.
Slides 5 and 6: DAC DRC and LVS result summary.
Slide 7: Picture(s) of your switch layout.
Slide 8: Picture(s) of your synthesized digital logic. Annotate the pin locations and the dimensions of the block (using rulers).
Slide 9: Picture(s) of your non-overlapping clock generator layout. Annotate the pin locations and the dimensions of the block (using rulers).
Slides 10 and 11: Digital DRC and LVS result summaries.
Slide 12: Picture(s) of your assembled SAR ADC. Annotate the devices/blocks and the dimensions of the block (using rulers).
Slides 13 and 14: SAR DRC and LVS result summary.
Slide 15-18: Simulation results from the extracted layout. Show transient, transient noise, and the spectrum. Simulate power consumption using the synthesized logic (only need to run a few cycles).
Slide 19-21: Summarize your strategy, floorplan (metal routing), and approach. Explain the techniques you employed to achieve the performance, using figures as necessary. Convey any issues you encountered with the lab. Comment on places where you would improve the layout if you had more time.

Grading Rubric: 40% design (DAC, switches, digital blocks), 10% DRC clean, 10% LVS clean, 10% simulations, 10% summary, and 20% design review. Points will be deducted for non-professional presentations at my discretion.

Frequently Asked Questions

What needs to be matched in the CDAC?

The top plate parasitic (comparator input) attenuates the signal but does not lead to non-linearity. It is only minorly important. The bottom plate parasitic (switch output) does not matter! It is driven by a low-impedance source! It costs "reference power" but does not impact linearity. The capacitance between the TOP and BOTTOM plate is the only cap that matters! (Hint: This is the CC cap in your extracted result.)

Layout Net: TOP Source Net: TOP Coupled Capacitors: CC (77 CC, C=2.50048E-12)

No.	To Layout Net	Value (F)	To Source Net
31	BOT<8>	1.23779E-12	BOT<8>
32	BOT<7>	6.20788E-13	BOT<7>
33	BOT<6>	3.10226E-13	BOT<6>
34	BOT<5>	1.56074E-13	BOT<5>
35	BOT<4>	7.78106E-14	BOT<4>
36	BOT<3>	3.94243E-14	BOT<3>
37	BOT<2>	1.96240E-14	BOT<2>
38	BOT<1>	9.92493E-15	BOT<1>
39	BOT<0>	4.85428E-15	BOT<0>

How well does the cap need to match?

You have a 9b ADC. The caps need to match to $1/2^9$, ~0.2%. So, the ratio should be within 1.996 for each of the bits.

Can you connect names in the layout without physically connecting them at that hierarchy level?

Yes! You can use virtual pins where the net name is followed by a :

I am running out of space running long simulations, extracting parasitics, and/or running LVS/DRC. How can I fix this?

Change the folder for temporary files to /tmp. Files in /tmp are not counted against your quota. For setting simulation files, check this out: https://www.youtube.com/watch?v=fbzNHNKtjVc&ab_channel=JeffreyWalling

How can I change the size of the digital block?

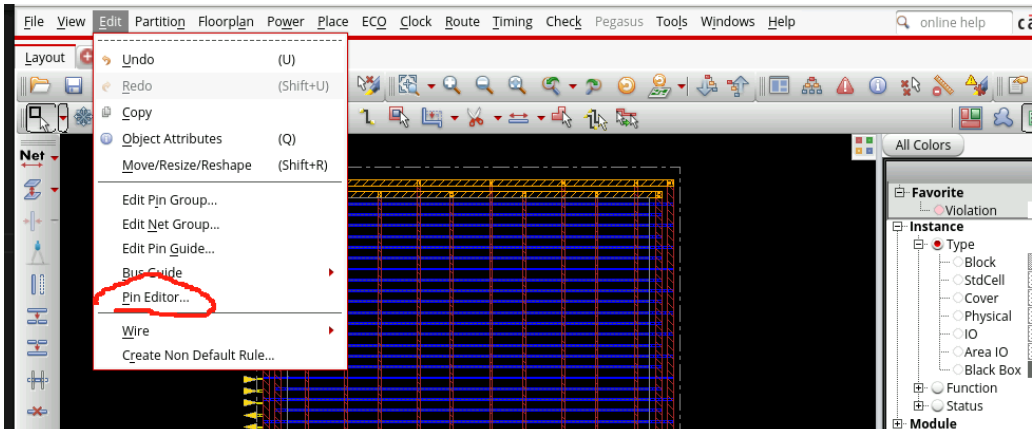
Change the Width and Height values in Design/Config/DesignConfig.cfg. If you make it too small, it won't be able to synthesize, so verify that it is correct! If it is too large, it will just be filled with decap cells. It is worth modifying these values to fit the digital logic to your design.

How can I change the pin location in the digital block?

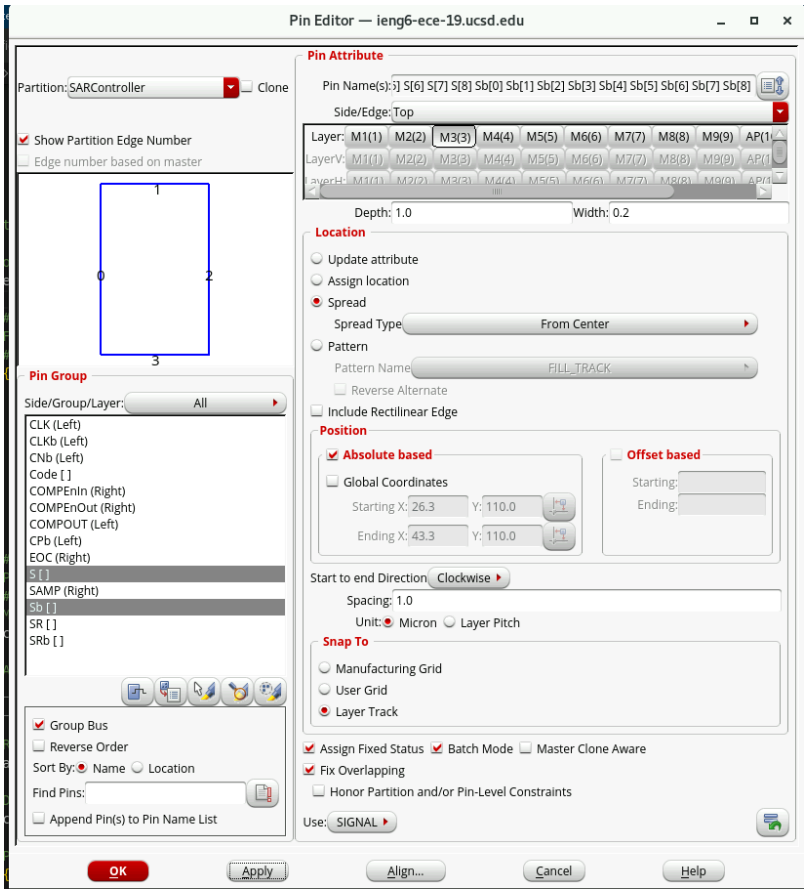
You can directly change the pin location in SARController.io, but normally, you can't determine the right coordinates for your pin initially, so it takes several tries to find the best pin placement. Alternatively, you can use the Innovus graphical user interface (GUI). To do this, we need the code to stop. The easiest way is to add a "bug" in our code! Create a "bug" in the PnR TCL script $4.0_PlaceNRoute/do_file.tcl to make it stall; then, you can see the innocuous GUI. (You can write anything, we just need it to call the Innovus GUI.) Then, rerun the shell script DigitalSynthesis/RUNME.sh. For example,

```
161
162     route_special  \
163         -nets                [list ${PowerNet} ${GroundNet}]
164 }
165
166 # It can be anything you want, just to make it stall
167 # I strongly recomend you to add it after powerplan
168 # so you can see the floorplan and pin place of the original desigh
169 make a stall
170
171 ##############################################################################
172 ## Floorplan database generation
173 ##############################################################################
174 if { $PlacementMode == 1 } {
175     ## Generate hand over data for GENUS
176     write_def -floorplan -no_std_cells -io_row ${_OUTPUTS_PATH}/${DesignName}.def
177     file delete -force $DEF
178     file copy   -force ${_OUTPUTS_PATH}/${DesignName}.def $DEF
179
180     # Quit
181     exit
182 }
183
184 ##############################################################################
185 ## Placement
186 ##############################################################################
```

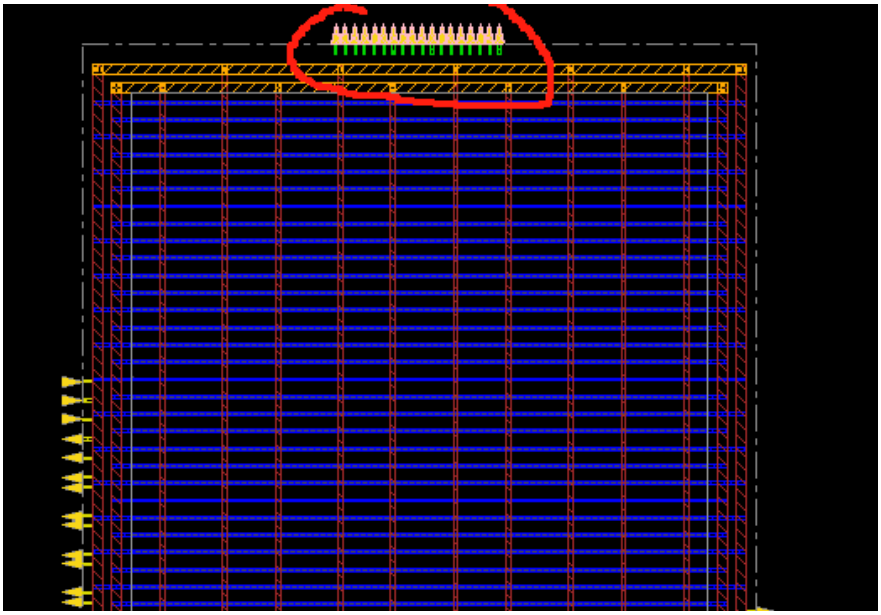
You should see the Innovus GUI with your original floorplan. Click the pin editor, then change the pin location:



For example, if you want S[ ] and Sb[ ] to be located in the middle of the top edge, you can config like this:



Then, you'll see the pins changed in your layout:



Next, find the *edit_pin* related commands in Innovus.cmd (every operation in Innovus will generate a corresponding command), and copy them into your do_file.tcl. Remember to delete the code you used to cause a stall, and now, every time, it will use your pin placement! Remember to check if you have the *-fixed_pin 1* option. Otherwise, Innovus might change your pin location after doing PnR optimization. You can find them in Innovus.cmd and copy:

```
gui_select -point {0.04150 0.02450}
set_db assign_pins_edit_in_batch true
edit_pin -pin_width 0.2 -pin_depth 1.0 -fixed_pin 1 -fix_overlap 1 -unit micron -spread_direction clockwise -side Top -layer 3 -spread_type center -spacing 1 -pin {{S[0]} {S[1]} {S[2]} {S[3]} {S[4]}
set_db assign_pins_edit_in_batch false
```

Then paste them in your TCL file:

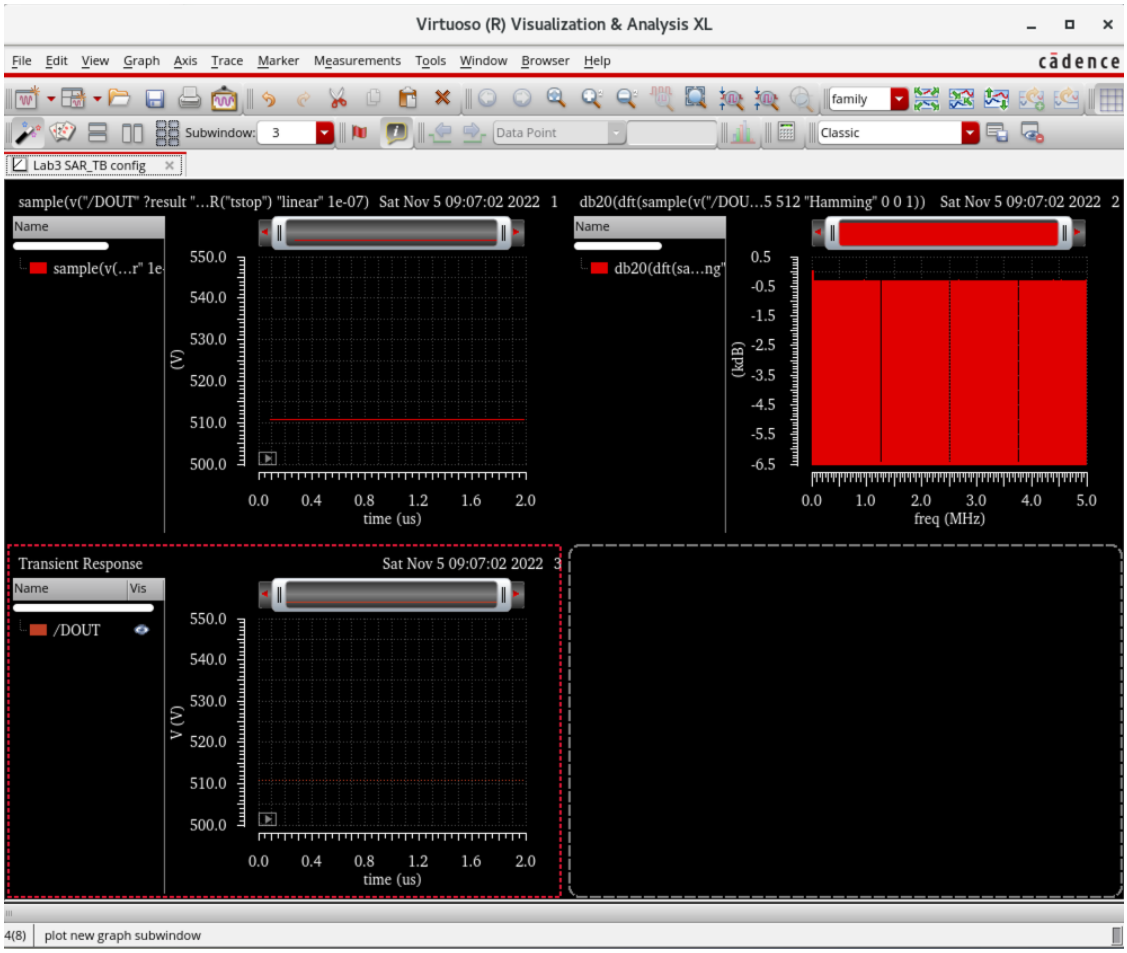
```
161
162     route_special  \
163         -nets                [list ${PowerNet} ${GroundNet}]
164 }
165
166 # It can be anything you want, just to make it stall
167 # I strongly recomend you to add it after powerplan
168 # so you can see the floorplan and pin place of the original desigh
169 # make a stall
170
171 ##########################################################################
172 ## Pinplacement
173 ##########################################################################
174 set_db assign_pins_edit_in_batch true
175 edit_pin -pin_width 0.2 -pin_depth 1.0 -fixed_pin 1 -fix_overlap 1 -unit micron -spread_direction clockwise
176 set_db assign_pins_edit_in_batch false
177
178
179 ##########################################################################
180 ## Floorplan database generation
181 ##########################################################################
182 if { $PlacementMode == 1 } {
183     ## Generate hand over data for GENUS
184     write_def -floorplan -no_std_cells -io_row ${_OUTPUTS_PATH}/${DesignName}.def
185     file delete -force $DEF
186     file copy   -force ${_OUTPUTS_PATH}/${DesignName}.def $DEF
187
188     # Quit
189     exit
190 }
```

Note that the scrips run PnR twice (Step 4 and 6), so paste the same commands in both $4.0_PlaceNRoute/do_file.tcl and $6.0_PlaceNRoute/do_file.tcl. Finally, exit Innovus and the terminal. You can run the DigitalSynthesis/RUNME.SH again, it will be the new pin placement.

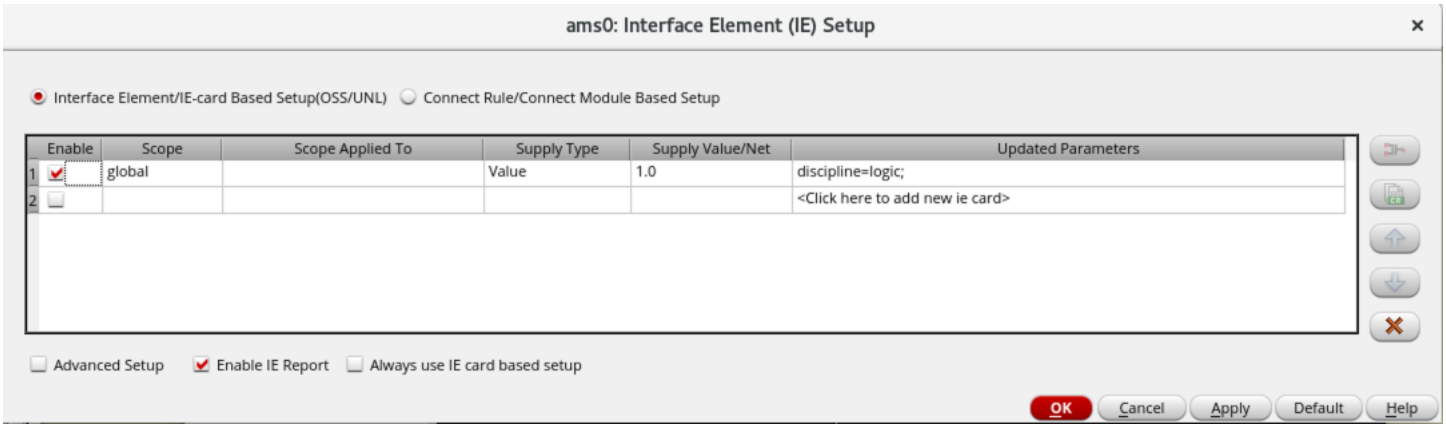
What type of matching is preferable for a capacitor array?
Generally, common centroid is preferred. However, good matching and routing parasitics are also considered, so a compromise between matching and routability is often made. Look at some examples in published research papers and think about why they did it that way.

Is it ok to connect both ends of the dummy capacitor to the TOP pin?
The top plate parasitic is not ideal. It will attenuate your input full-scale range. You need to see how much extra capacitance the dummies add.

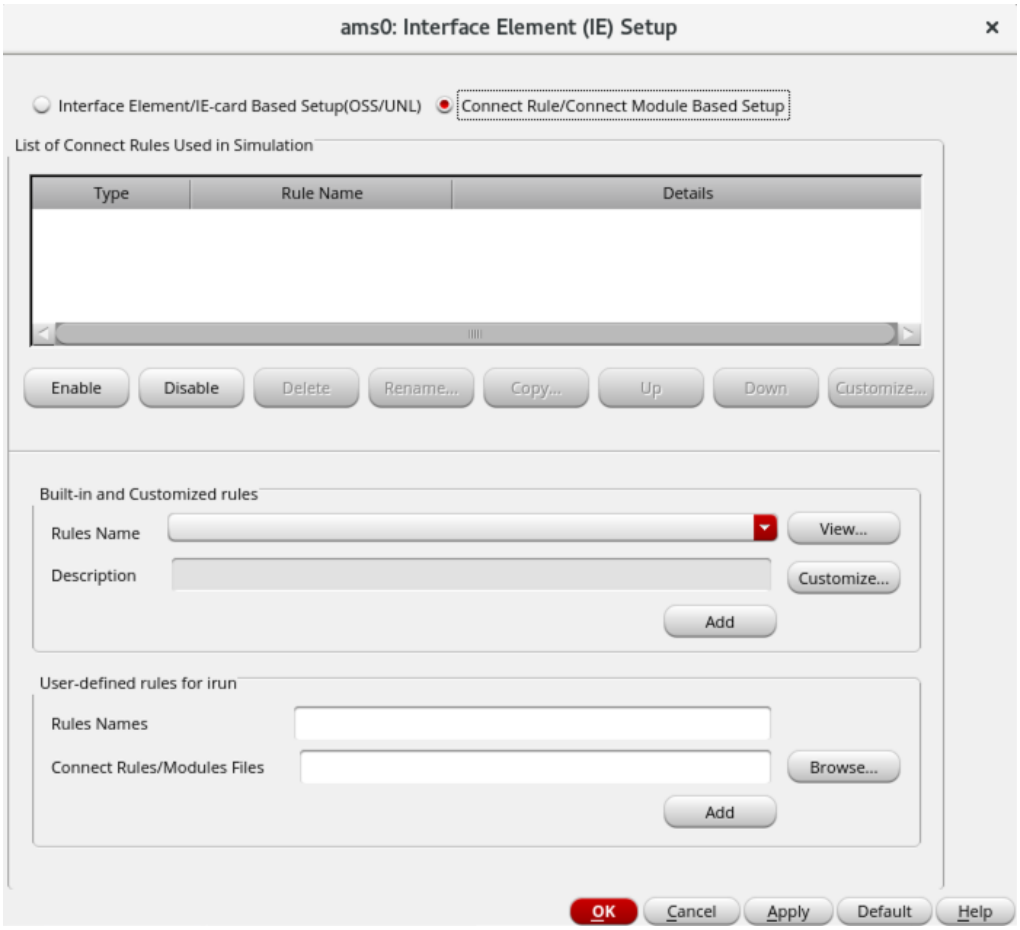
The transient simulations are coming up with flat lines on Dout. What is wrong?
The most common problem is that the Connect Rules/Interface Element is not properly configured. We are running “mixed-signal” simulations. That is, if you use the Verilog code, we need to pass off signals from the digital engine to spectre (the analog simulator). If your simulation results look like the following, we must fix the “connect rules.”



To fix this, go to ADE L -> Setup -> Connect Rules/IE Setup,



Then click on the Interface Element radio button in the upper left,



Change the supply Value/Net from 1.8 to 1.0. Hit Apply and re-run the simulation. This is how Verilog turns its digital representation of a 0 and 1 into a voltage. (Note, I spent a long time figuring out why this doesn't always save -- I give up! It appears to be random...)

What are the different simulations in the SAR_TB?

I've set up the test bench and stored the state in the cell.

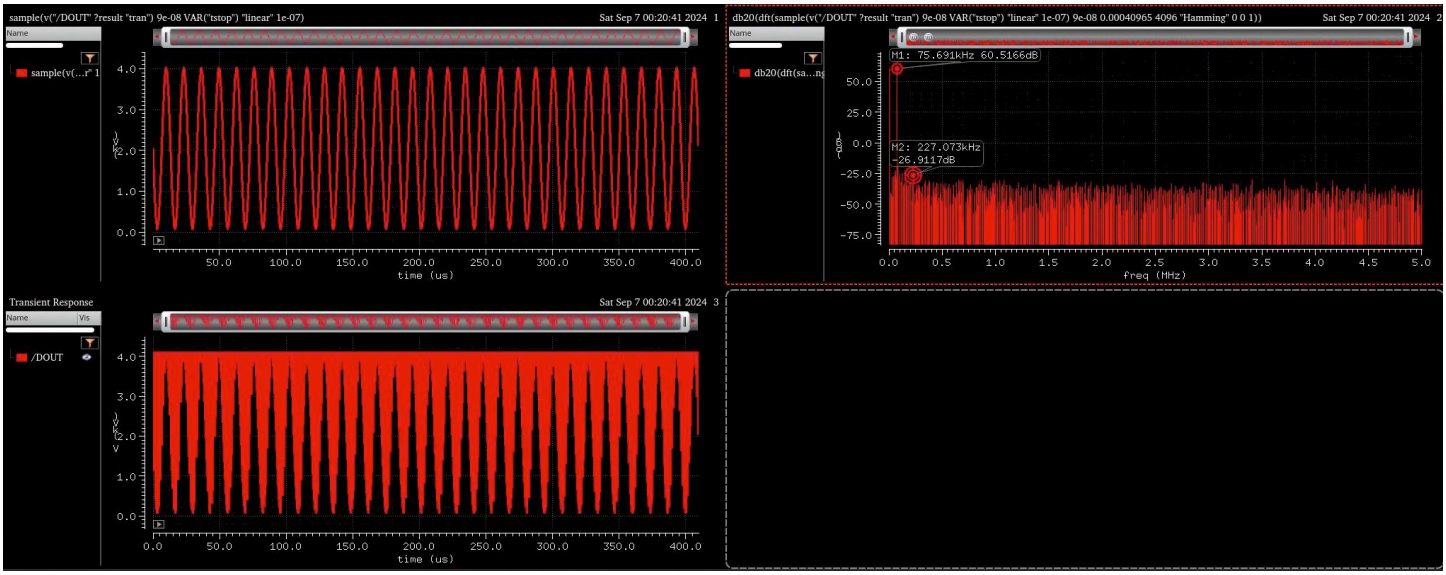
- **Tran Short** is to make sure that everything is working before you fire off one of the long simulations. This simulation takes only 2 minutes. This will not tell you the SNR of the ADC -- it doesn't run long enough to do that! I still recommend running this first to ensure things are not grossly wrong.
- **Tran_Long** is to simulate the SNR. On the ieng6 servers, this should take around 33 minutes to run (with schematics and Verilog code).
- **Tran_Long_Spectre** after you have imported your synthesized Verilog code into the design, this state switches the simulator from AMS (mixed-signal) to Spectre, which is much faster and has more control over the accuracy of the simulation. With imported Verilog code (see the digital synthesis document), this simulation should take around 30 minutes.
- **TranNoise_Long** is to simulate the design with transient noise. Depending on the server load, this will take 2-3 hours.

What are the design variables?

Fs is the sampling frequency (in Hz), Nfft is the number of points in the DFT, tstop is the time to stop the simulation, bin is a prime number to calculate the input frequency such that it is not at the boundary of a bin, and Fin is the input frequency. Unless you are doing something different with the project, you should not modify these!

What should the SAR output look like?

Here is an example. The upper left shows the sampled input signal, and the upper right shows the spectrum of the signal. Note it has been annotated here to show the highest spur to measure the spurious free dynamic range (SFDR). This plot is in dB on a linear frequency axis. Finally, the bottom left plot shows the digital output (converted to a "voltage" where the gain is 1V/code).



I am doing something different, can you explain the calculations?

The performance of the ADC is calculated using two equations:

db20(dft(sample(v("/DOUT" ?result "tran") 9e-08 VAR("tstop") "linear" 1e-07) 9e-08 0.0004097 4096 "Hamming" 0 0 1))

Let me breakdown this equation:

- sample(v("/DOUT" ?result "tran") 9e-08 VAR("tstop") "linear" 1e-07): This samples DOUT linearly for every 100ns starting from 90ns until VAR("tstop")
- 9e-08 0.0004097: calculates the DFT of the sample signal, starting from 90ns until 409.7us
- 4096: performs a 4096-point DFT
- "Hamming": apply a Hamming window to reduce spectral leakage in DFT
- 0 0 1: first 0 is the start frequency of the DFT; second 0 means that the DFT would compute up until Nyquist frequency (half the sampling frequency), 1 means normalization

spectrumMeasurement(sample(v("/DOUT" ?result "tran") 9e-08 VAR("tstop") "linear" 1e-07) t 9e-08 0.0004097 4096 75691 300000.0 3 "Hamming" 0 0 1 "snr")

- sample(v("/DOUT" ?result "tran") 9e-08 VAR("tstop") "linear" 1e-07): This samples DOUT linearly for every 100ns starting from 90ns until VAR("tstop")
- t: indicates we are working with a time-domain signal
- 9e-08 0.0004097: calculates spectrum measurement of the sample signal, starting from 90ns until 409.7us
- 4096: performs a 4096-point DFT
- 75691: fundamental frequency for the analysis, should be set to the same as the input frequency
- 300000.0: maximum frequency over which the spectrum measurement is considered, set this number to be at least 3 times the fundamental frequency to consider noise up till 3 times the harmonic
- "Hamming" 0 0 1: same as before
- "snr": SNR is to be calculated from the spectrum

LVS will not pass for my custom MOM capacitor. What is wrong?
A few things must be set properly for the LVS to pass with a custom MOM capacitor.

- Calibre → Database → Additional Spice files: You need to add $PUBLIC/Calibre/lvs/source.added.
- Calibre → Inputs → HCells: Check "Use H-Cells file" and add $PUBLIC/Calibre/lvs/xcell
- Calibre → Options → Check "Include Rule Statements" and type LVS FILTER C(CP) OPEN SOURCE

These changes should get you your check mark and green happy face! There will still be warnings about the power nets -- these are fine; there are no power nets! At higher levels of hierarchy, you can fix these by going to Calibre → LVS Options → Supply: Enter the names of the power nets (VDD and VSS)

Why won't LVS will not pass for the synthesized logic?
You must add a port on VDD and VSS in the imported layout. Once you do this, it will pass with Calibre. (In the weeds a bit, but the synthesis script itself runs LVS, and it passes! Which made this a rather interesting problem to debug. It appears to be a nuance with a newer version of Calibre that we are using now.)