Design Databases and More

Peter Kinget

Columbia University

Outline

- Schematic Databases
  - What do they represent?
  - Example: From transistors to application
  - Guidelines for good schematics
- CMOS Process
- Tracking Progress
- Documenting Simulations
- GitHub Design Database Repo Template

Keys to a Successful Tape-Out

<u>In arbitrary order:</u>

- Tracking Schedule
- Understanding the physical reality
- Work deliberate
  - First time right
    - → One compile in IC design (=fab) & no `bug patches’
  - Be organized
    - → Lots of small details that matter
  - Document your work
  - Track issues
- Teamwork

Recommendations

Learn the Linux terminal & shell and command-line interfaces

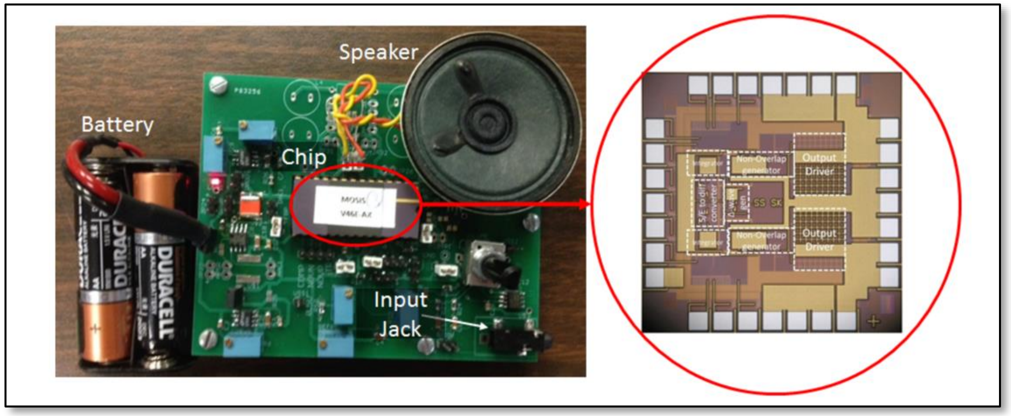
Learn Git and GitHub repositories

Take a peek under the hood of EDA tools; e.g. netlists for `ngspice`

Schematic Databases

What we want to represent in the CAD tools

Example: Class-D Amplifier



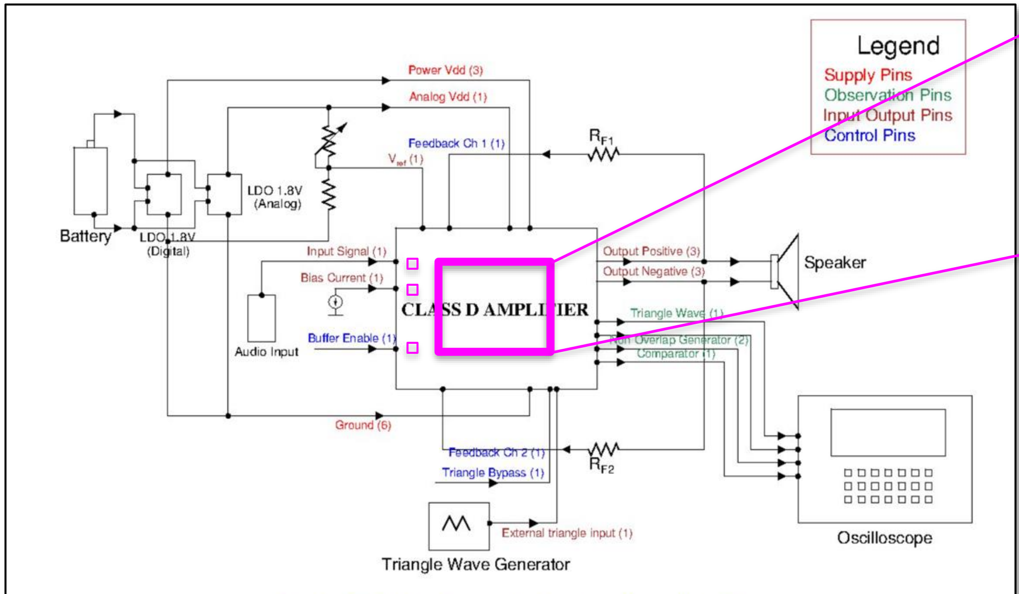
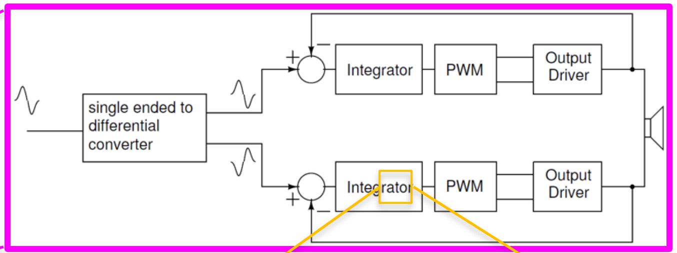
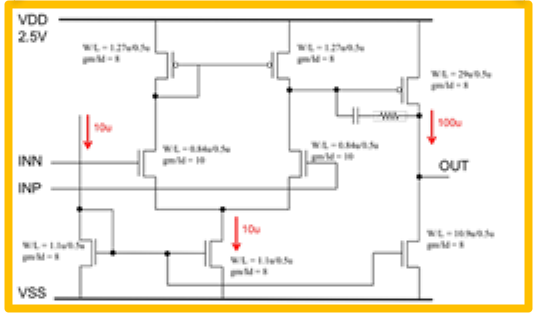


Fig. 1: PCB block diagram for the class-D amplifier IC





From: https://www.ee.columbia.edu/~kinget/EE6350_S14/ClassD_SKSS/pcbDesign.html

How To Construct A Schematic Database

- Database needs to reflect the physical reality:
  - Application:
    - PCB testboard
  - Packaged Chip:
    - Package + Die
  - Die:
    - Pads + ESD + Core Circuits
  - Circuits:
    - From top level to block to basic circuit level
- A similar hierarchy in the schematic and layout database simplifies LVS verification greatly
- Schematic database needs to capture everything:
  - Dummy devices, …
- Notes and documentation on the database are KEY for future reference
  - Like comments in computer code

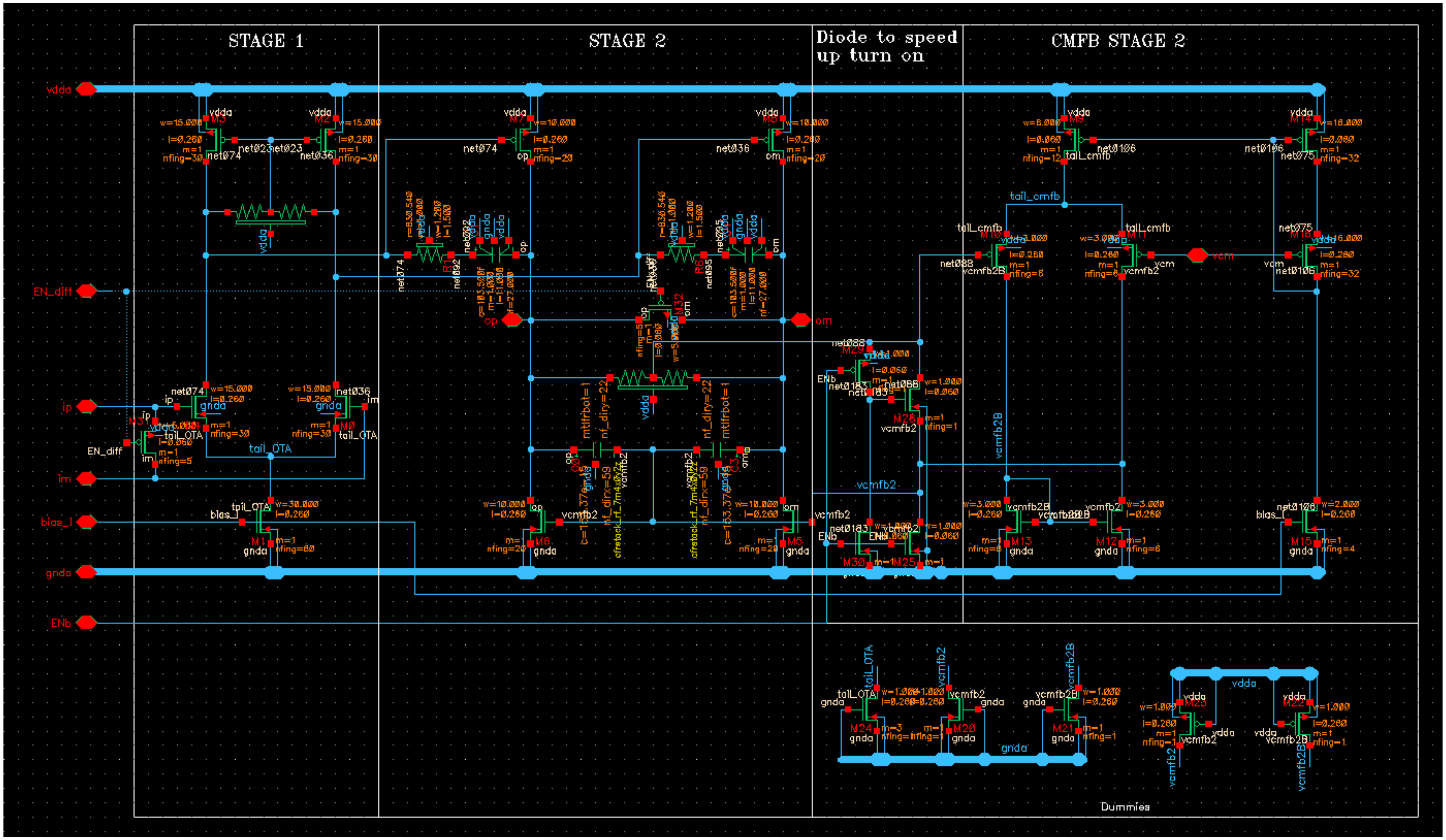
Collaboration "Discipline"

- Shared Libraries
  - Carefully choose your block names to avoid overlap among designers
  - One practice is to name your blocks always starting with your initials, e.g.,
    - PK_OTA
    - PK_BGR
- Naming and Versioning control
  - Make sure to carefully name your blocks
    - TEST1, TEST2, TEST3, … really does not say much
    - twostageOTA, foldedcascodeOTA, … is more informative
  - Separate your blocks from your testbenches
    - twostageOTA and tb_twostageOTA
- Housekeeping: spend time keeping things organized
  - Within a few days, weeks you will not remember what is what …
  - Don't necessarily delete things, but move them to "old" folders
  - Consider making finished blocks read-only

Example: From Transistor Level to Application Level

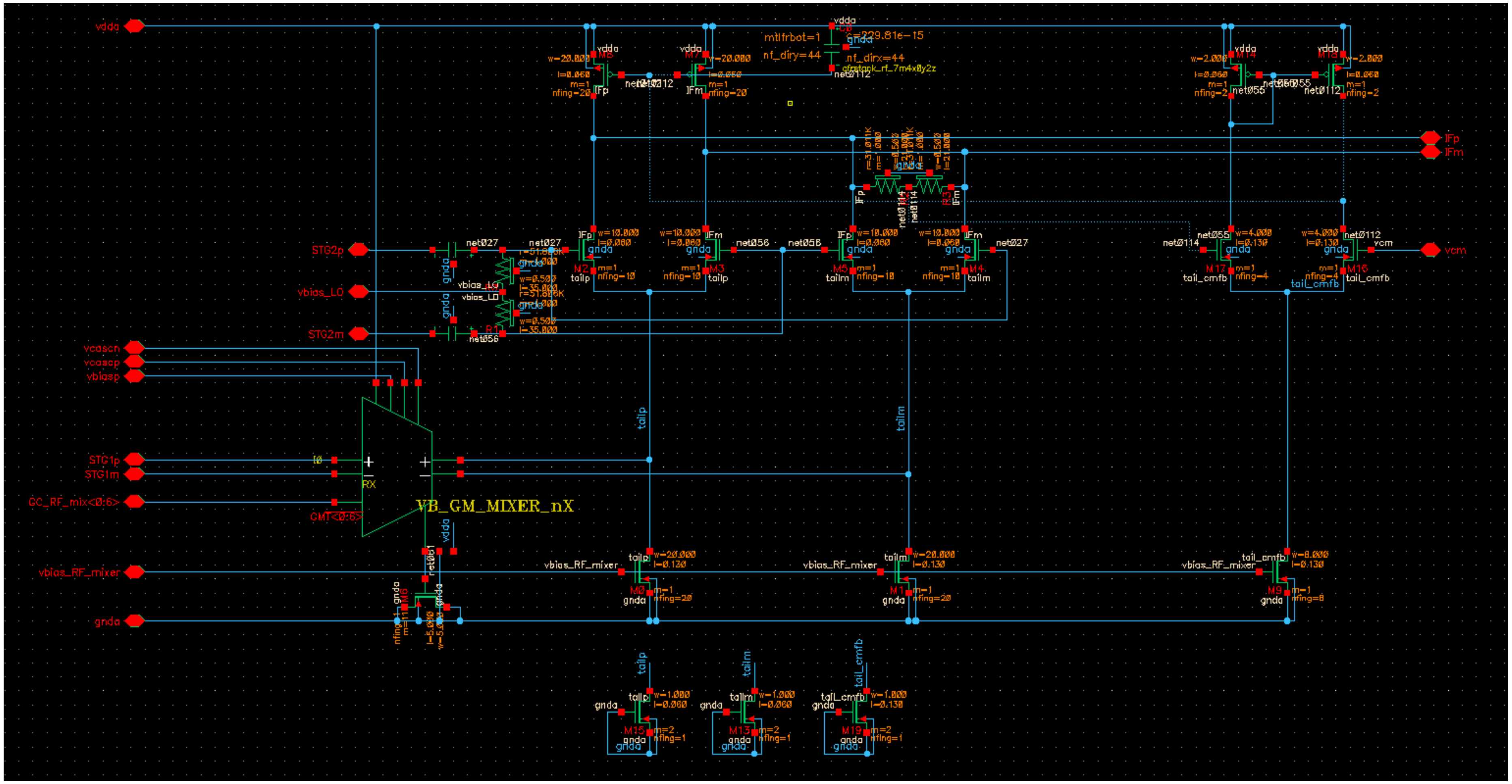
B. Vigraham and P. R. Kinget, “A Self-Duty-Cycled and Synchronized UWB Pulse-Radio Receiver SoC With Automatic Threshold-Recovery Based Demodulation,” *IEEE Journal of Solid-State Circuits*, vol. 49, no. 3, pp. 581–594, Mar. 2014, doi: 10.1109/JSSC.2014.2303804.

Circuit Schematic: OTA



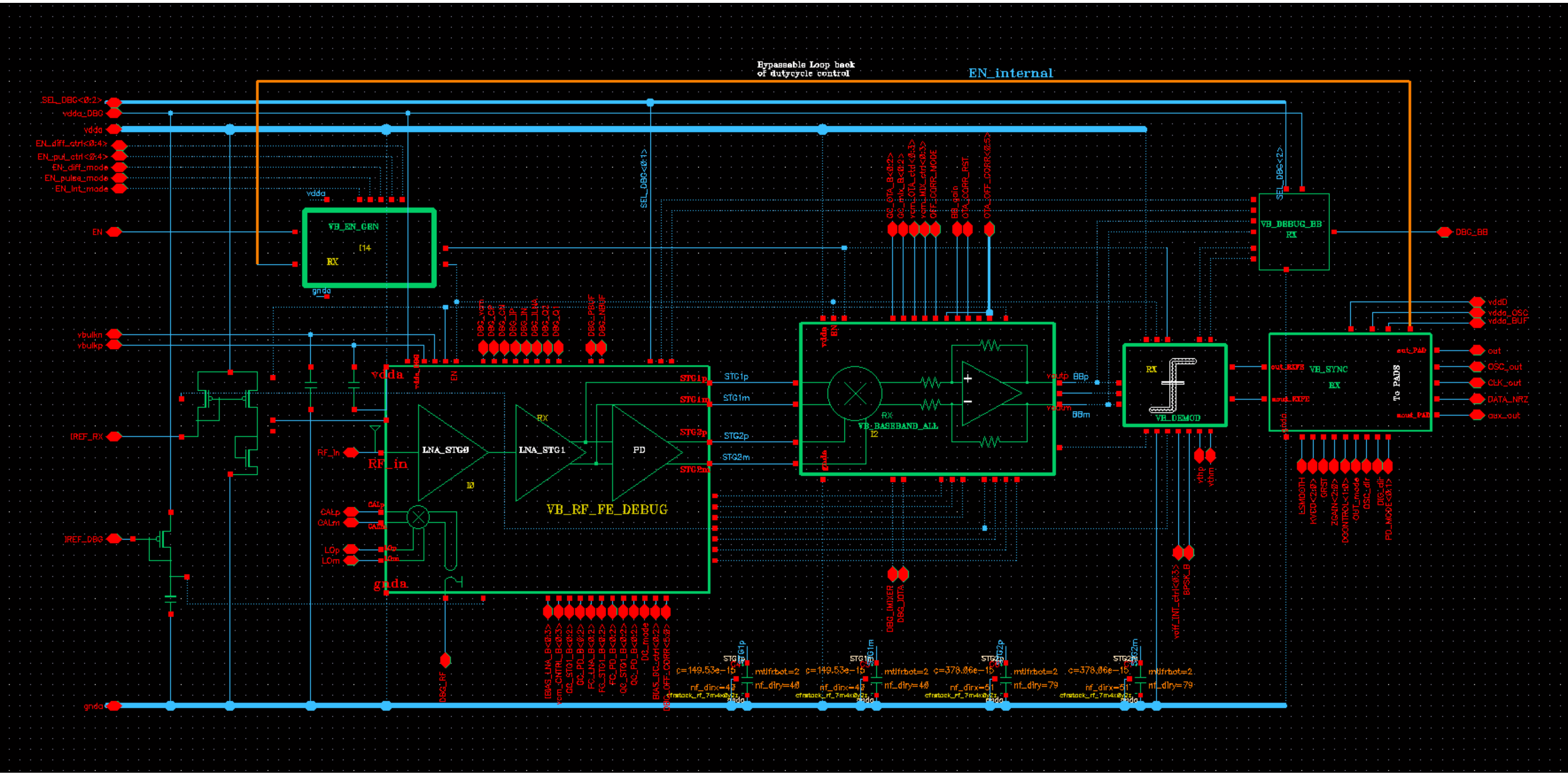
- Transistor level

Circuit Schematic: Self-Mixer



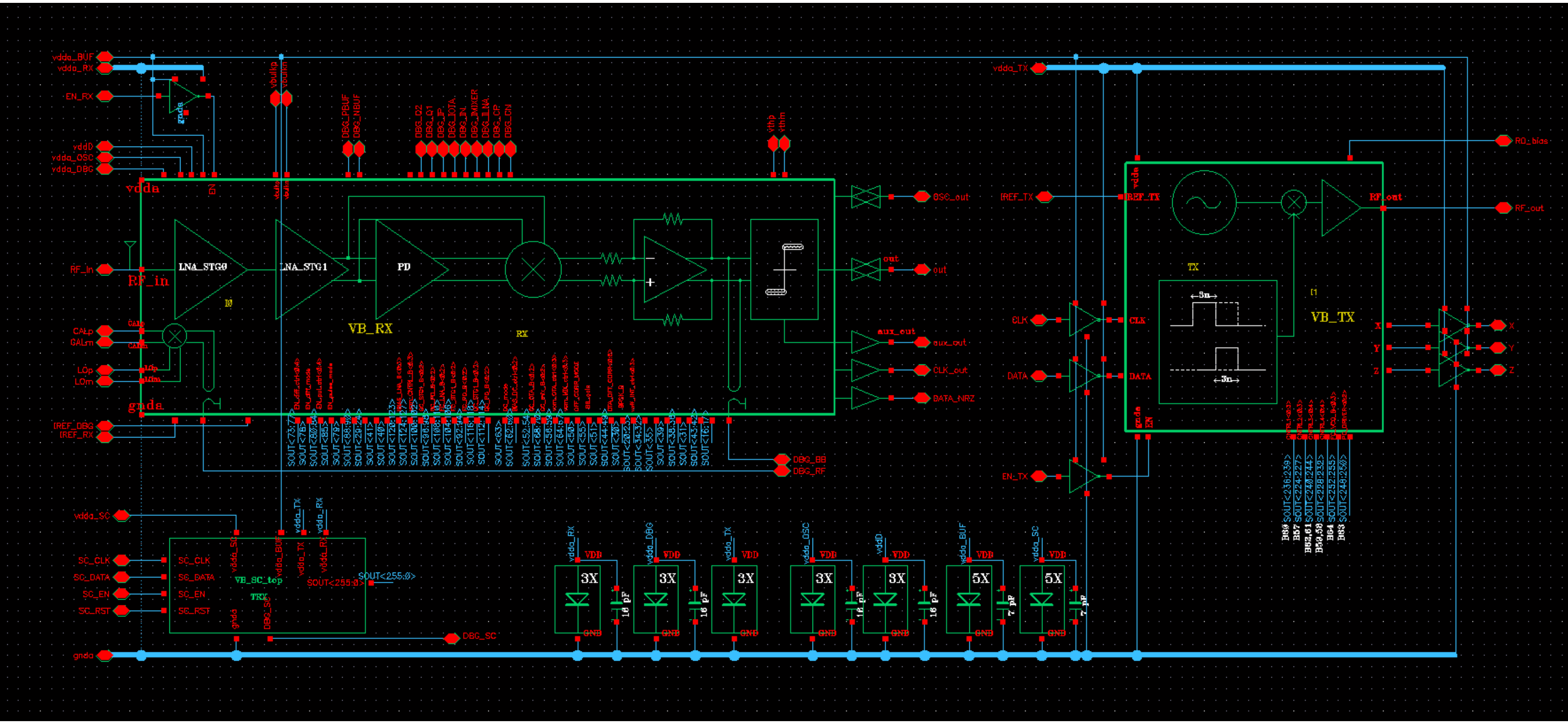
- Transistor level

Toplevel



- Receiver Top Level

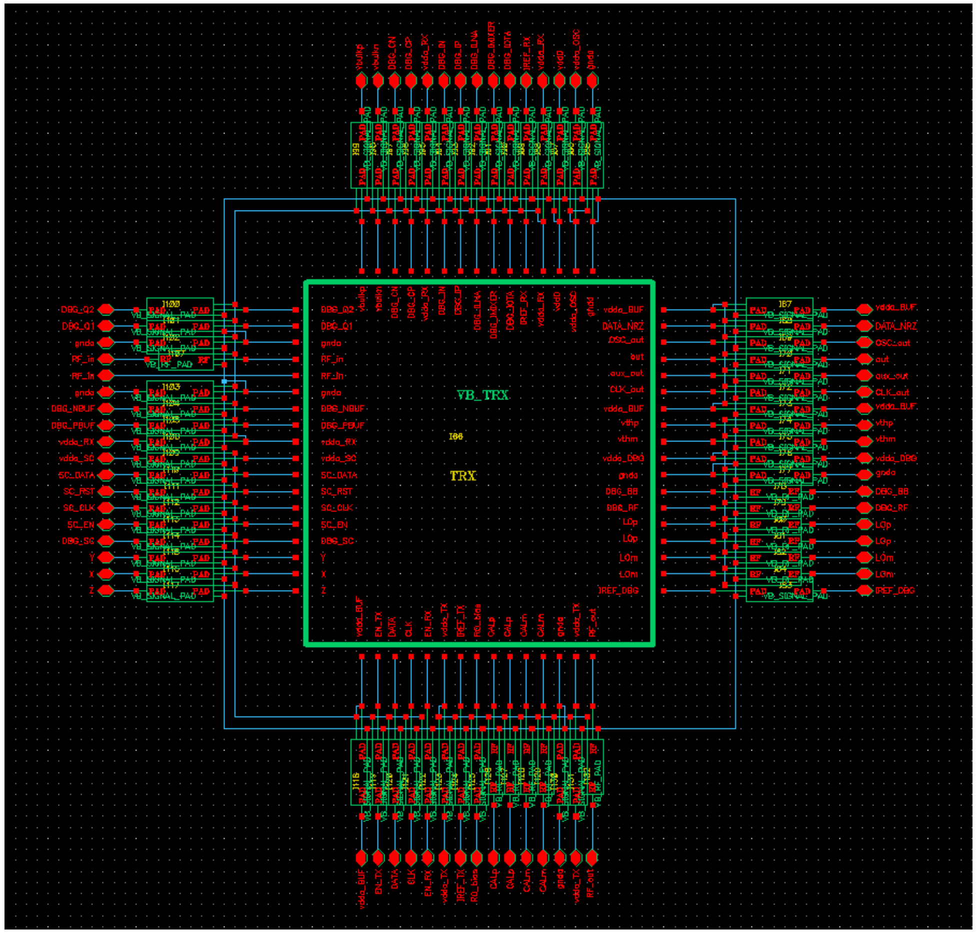
Toplevel



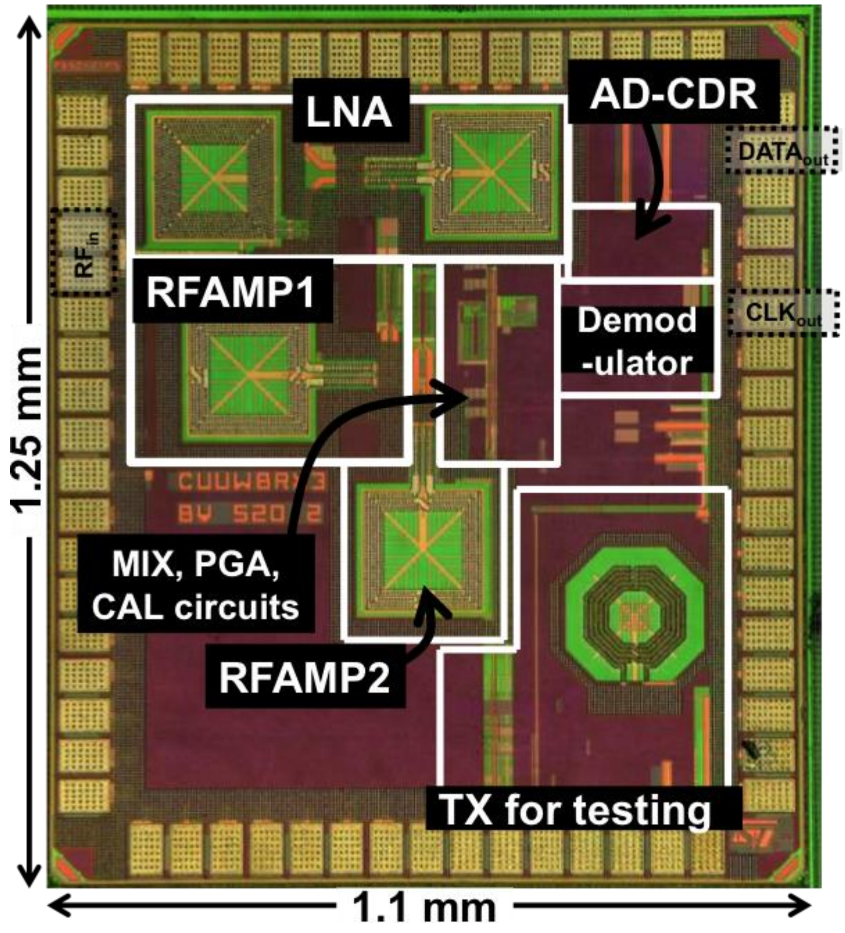
- Chip Top Level: RX + TX

Chip Die: PADS-ESD

Schematic View

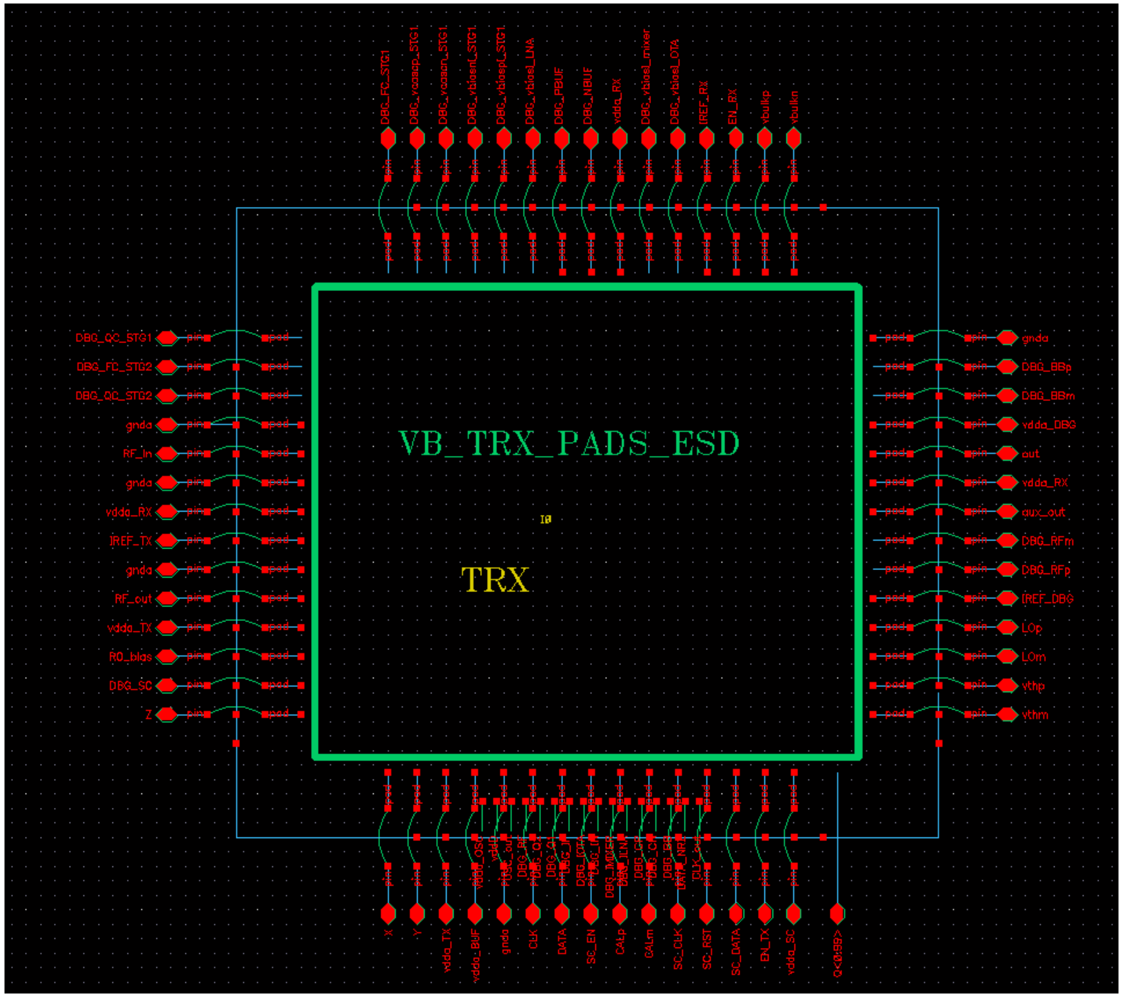


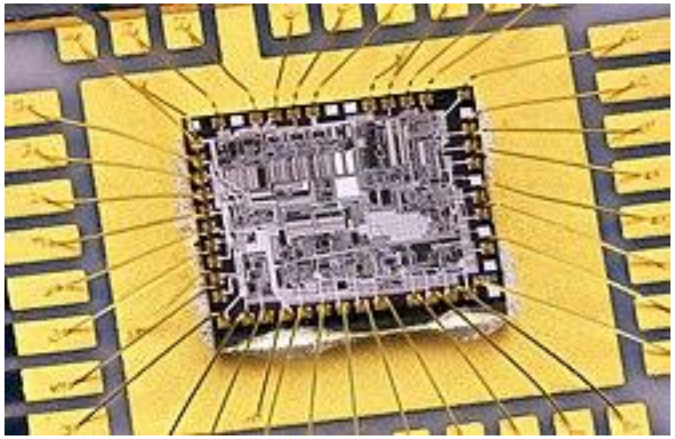
Die Photo



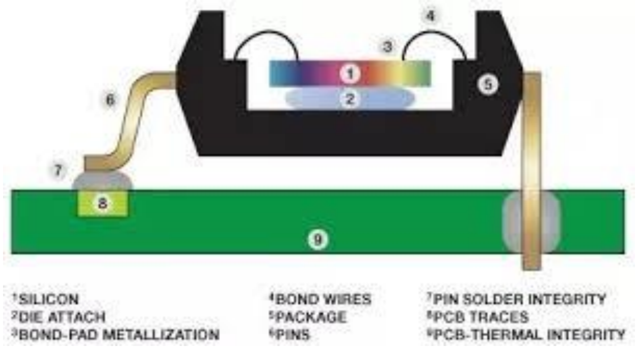
- Top Level Schematic + PADS + ESD Ring

Packaged Chip





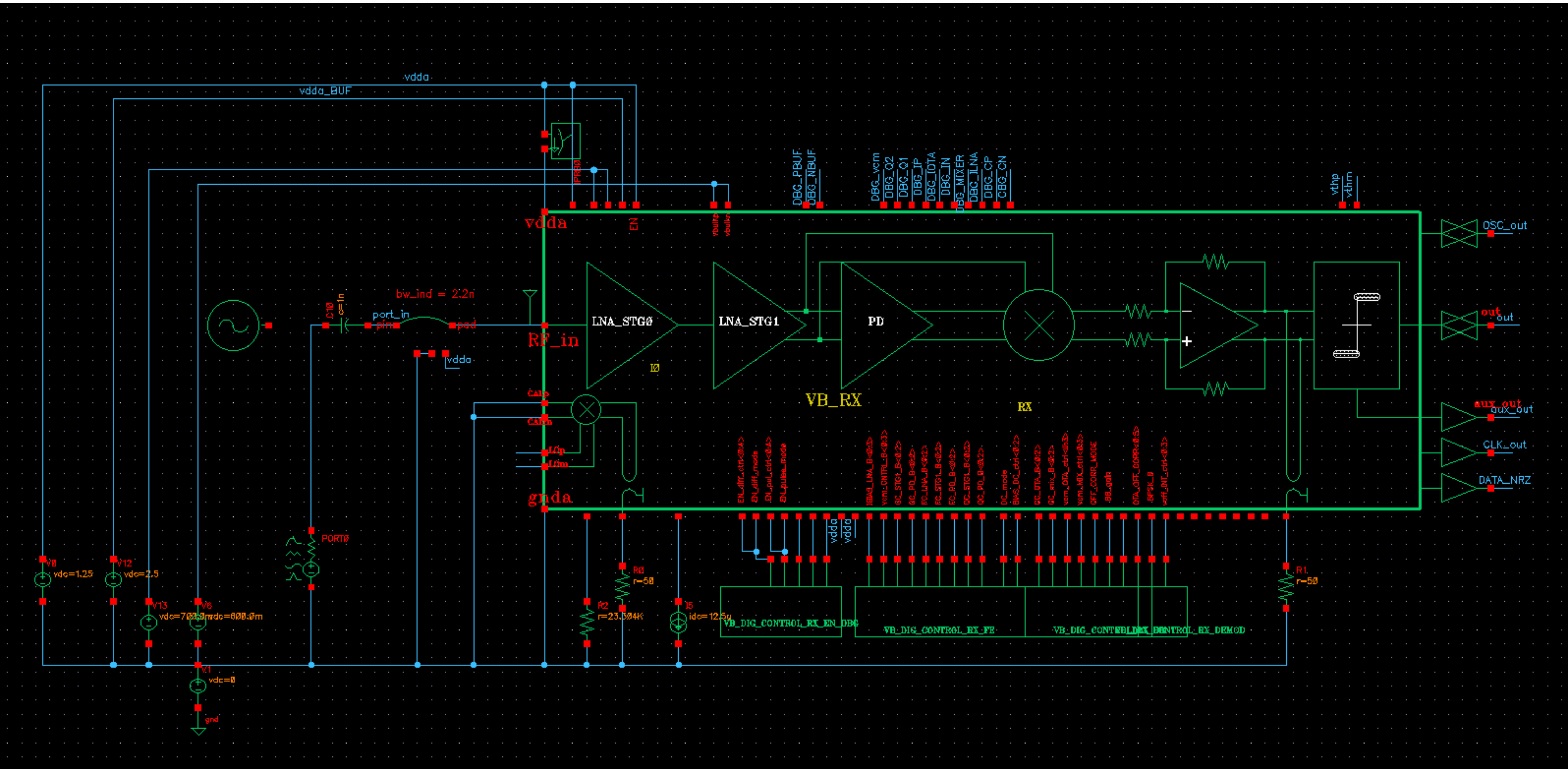
https://en.wikipedia.org/wiki/Wire_bonding



https://en.wikipedia.org/wiki/List_of_integrated_circuit_packaging_types

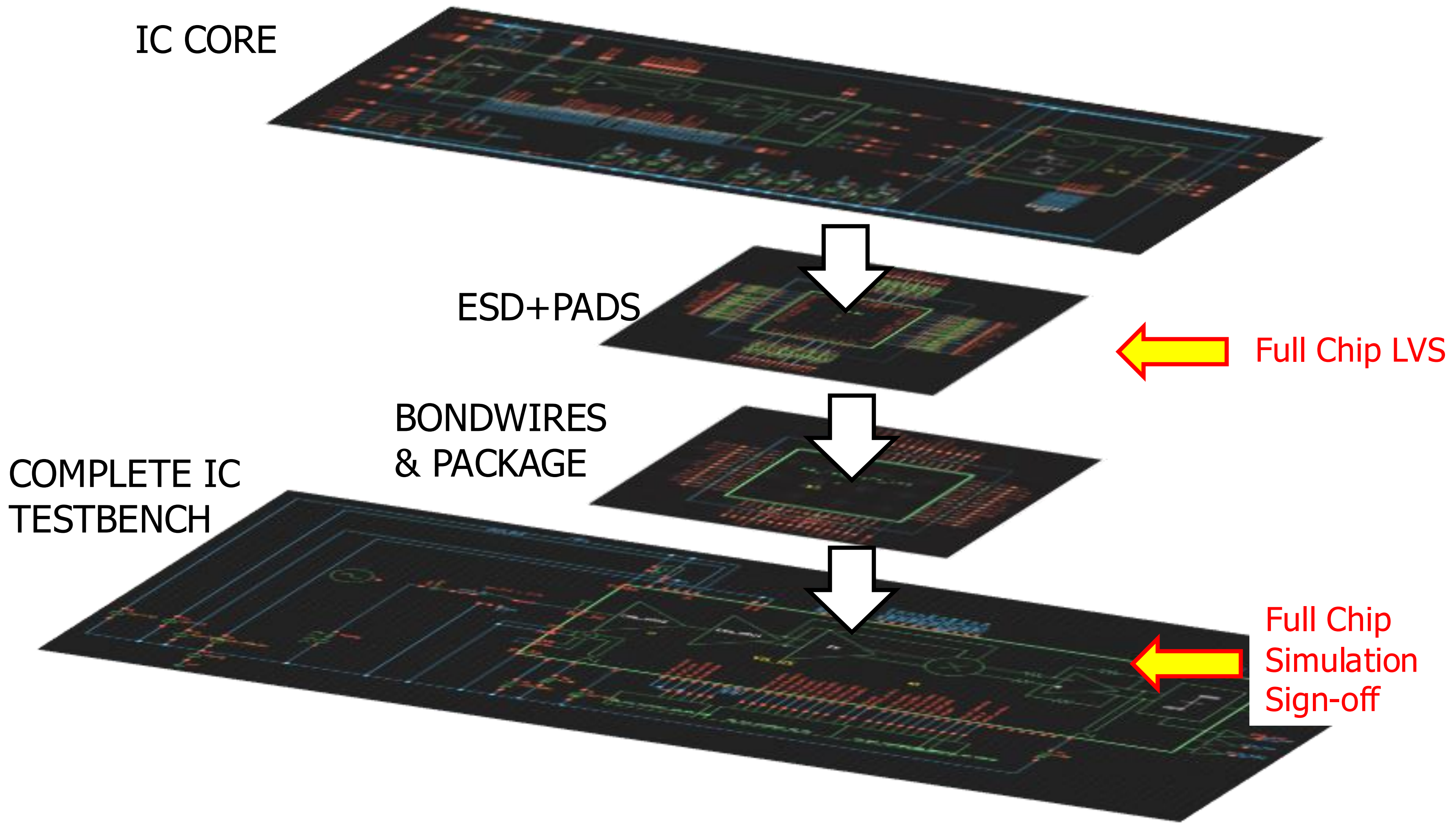
- Die schematic + Bondwires + Package Parasitics

Application Testbench – PCB



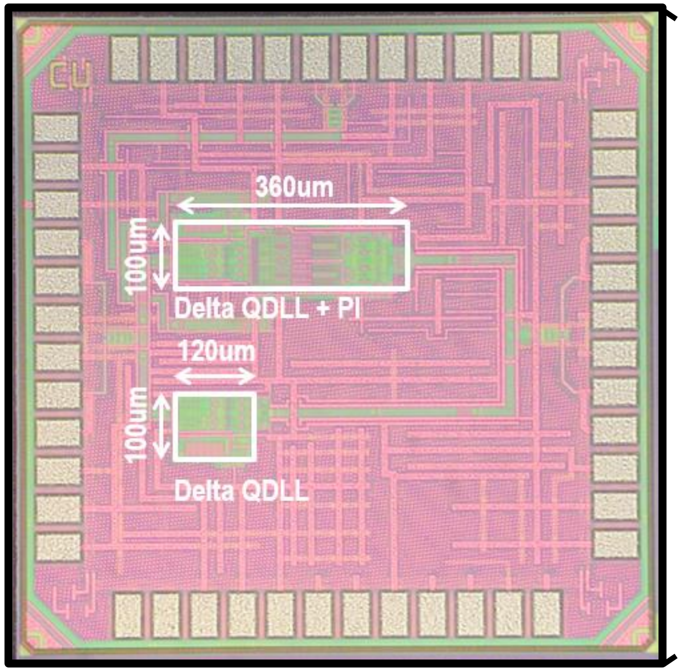
- Application: Packaged chip on PCB

Schematic Database Hierarchy

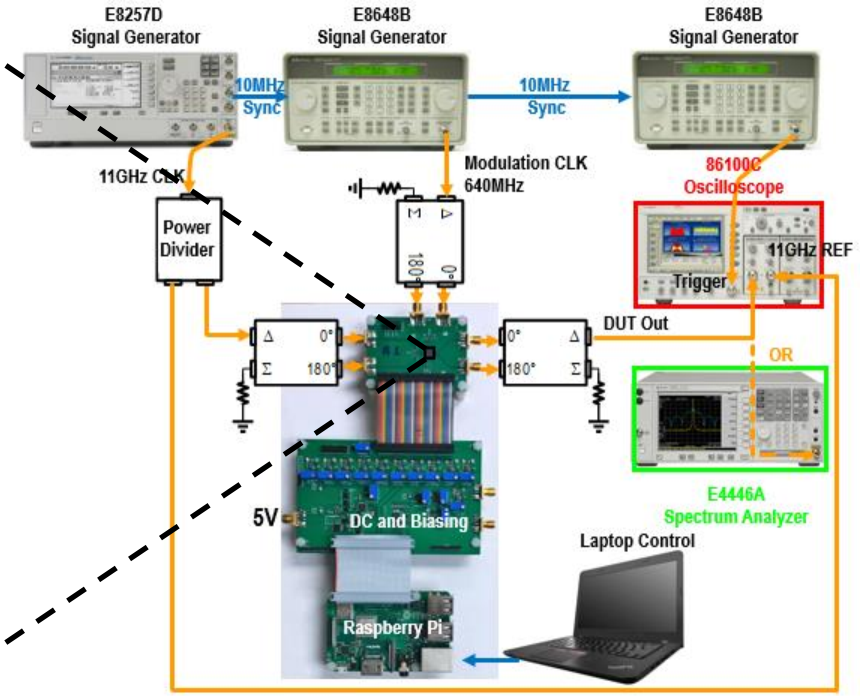


Example: Chip Photo and Measurement Bench

65nm Chip Photo



Measurement Bench



Z. Wang, Y. Zhang, Y. Onizuka, and P. R. Kinget, “Multi-Phase Clock Generation for Phase Interpolation With a Multi-Phase, Injection-Locked Ring Oscillator and a Quadrature DLL,” *IEEE Journal of Solid-State Circuits*, pp. 1–1, 2022, doi: 10.1109/JSSC.2021.3124486.

Guidelines for Analog/Mixed-Signal Schematics

http://www.ee.columbia.edu/~kinget/TOOLS/schematics.html

These are not 'absolute' rules and guidelines might differ, but stick to good practices when making your schematics

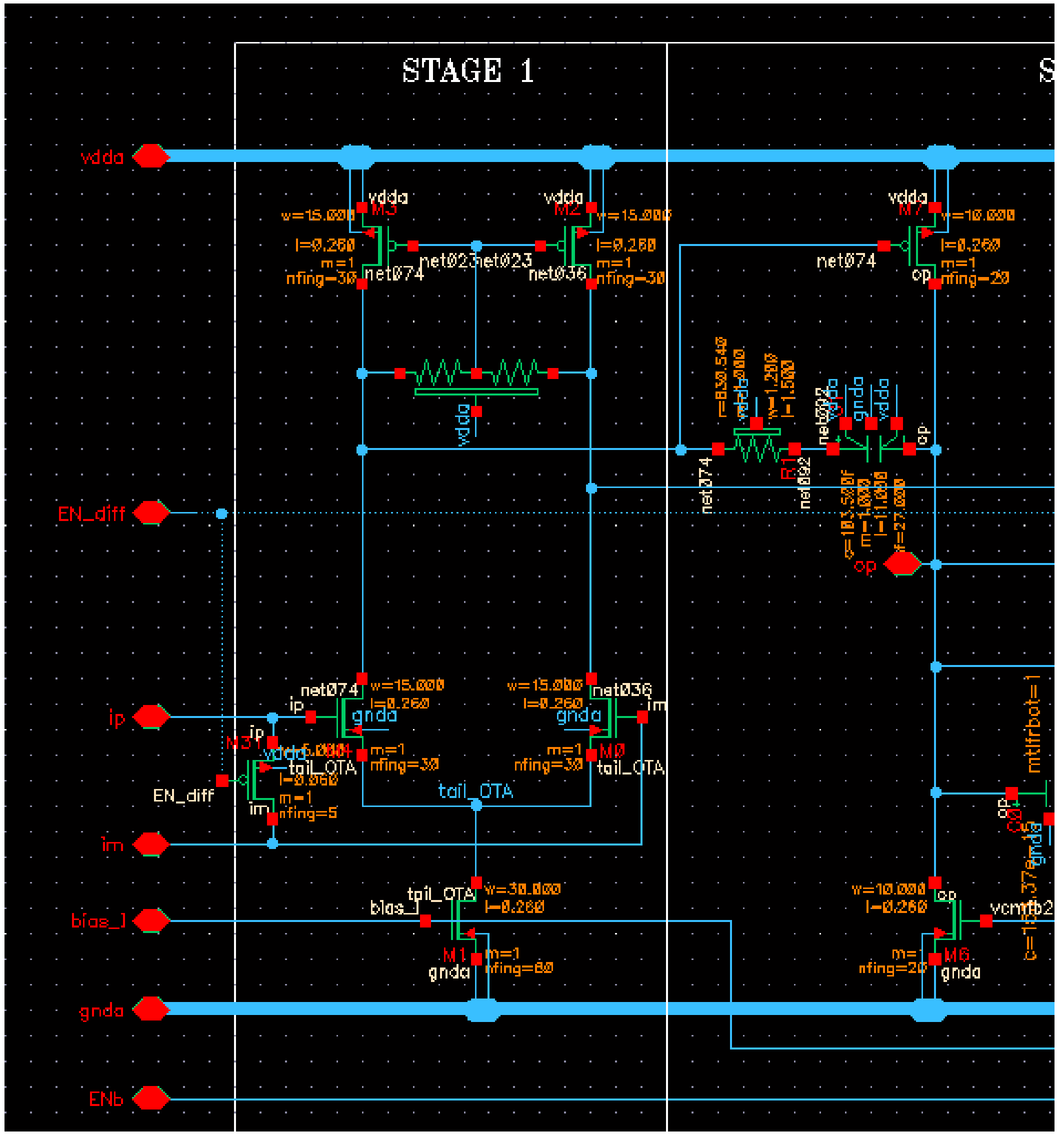
Device Symbols

- Transistor Symbols
  - Only use 4-terminal MOS symbols.
  - This forces you to think where the body should be connected.
  - For pMOS transistors and nMOS transistors in a separate well the option exist to connect the body to the source.
- Capacitor Symbols
  - Typically have a substrate connection – think carefully where to connect it; verify with technology manual
- Resistor Symbols
  - Typically have a substrate connection – think carefully where to connect it; verify with technology manual
- Also, if you want to do (manual) substrate noise simulations you need access to the body terminal.

Power Supply and Ground

- VDD & GND symbols
  - Do not use the special VDD symbols anywhere.
  - Do not use the special GND symbols inside your cell schematics.
- Use only one GND symbol in your toplevel (testbench) schematic so the simulator has a reference node.
- Subcells
  - Each subcell needs to have a VDD and VSS I/O pin.
- Note:
  - Power supplies and grounds are circuit nodes like any other nodes. The only thing that sets them apart is that a lot of components are connected to them compared to the number of components connected to signal nodes.
  - Using special symbols inside subcells results in a lot of implicit connections that cannot be undone and are hard to trace.
  - Moreover, you cannot do any power supply or ground noise simulations on individual blocks or subsections of your circuit.
  - In the layout the supplies and ground are actual nodes. To model the physical reality, you often need to include series resistance and inductance or decoupling capacitance at different locations. This is not possible if everything is globally connected through special symbols.

Circuit Schematic: OTA



- Vdda on top
- Gnda on bottom
- Inputs left
- Outputs right (not shown)
- 4T-transistor symbols
- Resistor & Capacitor devices have 3rd term.

- Notes as comments
- Internal nodes have names
- Devices have names

Naming and Connections by Naming

- Avoid global names,
  - all interfacing needs to be done through pins on the symbols.
- Avoid connections by naming.
  - Use wires to connect nodes and elements. (You will have to draw these wires in layout also …)
- Note:
  - Global nodes have the same problems as outlined above for power supplies and ground.
  - Global nodes do not physically exist.
  - Also connections in schematics made by giving the nets the same name are very difficult to trace.
  - Changes to the schematic might overlook these connections.
  - Connections by naming can simplify the look of a schematic compared to a jungle of wires, but that is deception. If the circuit requires complicated routing, that should be obvious from the schematic.

Naming – Use descriptive names

Analog Signals	
<u>Don't use</u>	<u>Use</u>
Ib1	Ibias_100u_p 100uA current going into pMOS mirror
Ib2	Ibias_10n_n 10nA current going into nMOS mirror
inp, inn	inp_OTA, inn_OTA
bias	vbias_casc_OTA
Vdd	Vdd_1V8, Vdd_3V3
Digital Signals	
make sure to indicate active low vs active high	Enable_B or ENb (active low)
	Enable or EN (active high)

Name Everything!

- Add names on everything.
  - Every single transistor,
  - every single inverter,
  - every single instance,
  - every single net.
- The debugging becomes much simpler.
  - For example, you run a simulation and get a comment on voltage level for I0.I13.I14.I19.M12.D connected to net /I0/I13/I14/I19/net3.
  - In my world it is much easier to read: Ifilter.Ibias.Istartup.Iota.Mtail.D connected to net /Ifilter/Ibias/Istartup/Iota/vCommon.

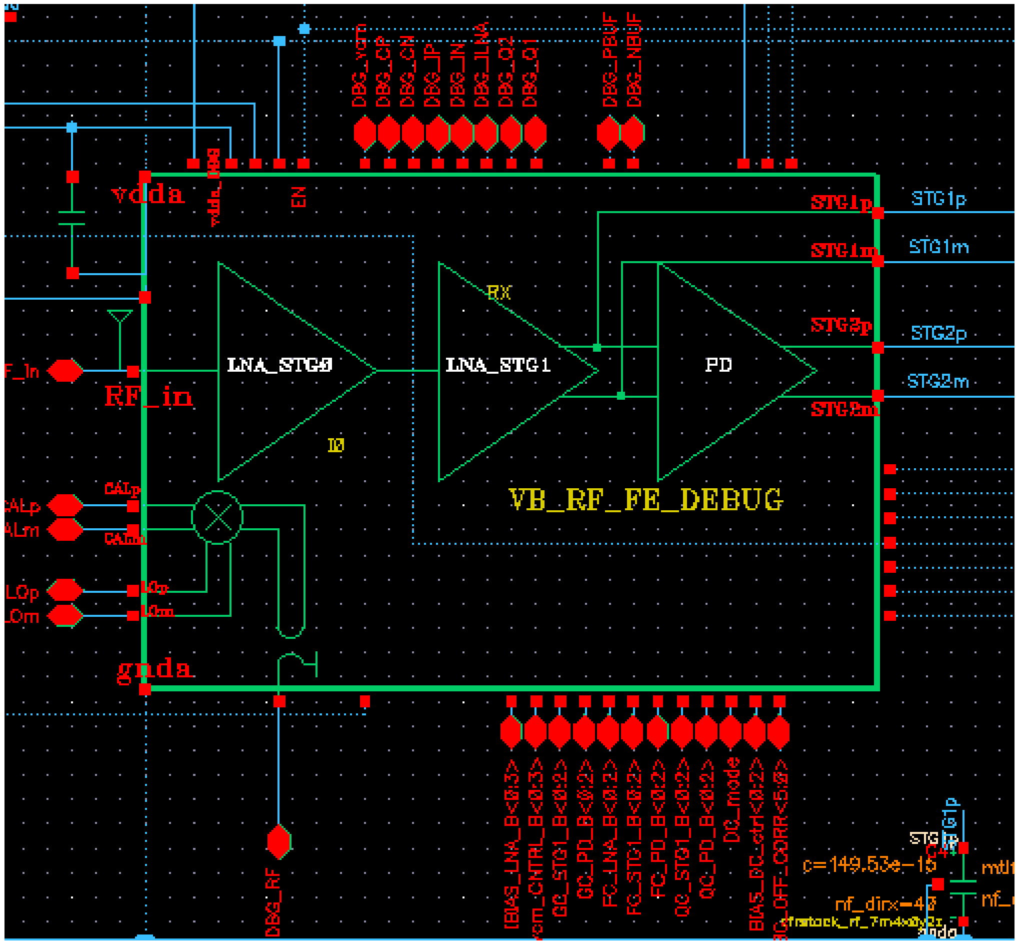
From: http://mixedsignal.wordpress.com/2011/03/13/top-ten-tips-on-schematic-entry-in-cadence/

Hierarchy and Symbols

- All interactions between the sub cells and the higher up cells needs to be through I/O pins.
  - Again, do not make connections through global nodes.
- Pin placement on Symbols:
  - Use symbols with the inputs at the (top) left side,
  - the outputs at the right side,
  - bias and control at the (bottom) left side,
  - power supplies at the top and ground at the bottom.
- Try to maintain the same hierarchy in schematics as in the layout. This simplifies layout-versus-schematic checks.

Symbol

VDD & Control



Inputs

Outputs

VSS & Control

Pin Placement In Schematics

- Keep the positive supply rail at the top of the schematic.
- Keep the negative supply rail at the bottom of the schematic.
- Keep all input, control and bias pins lined up at the left side.
- Keep all output pins lined up at the right.
- Do not put any I/O pins in the inside of the schematic

Signal Flow

- Keep the signals going from the left to the right in your schematic.

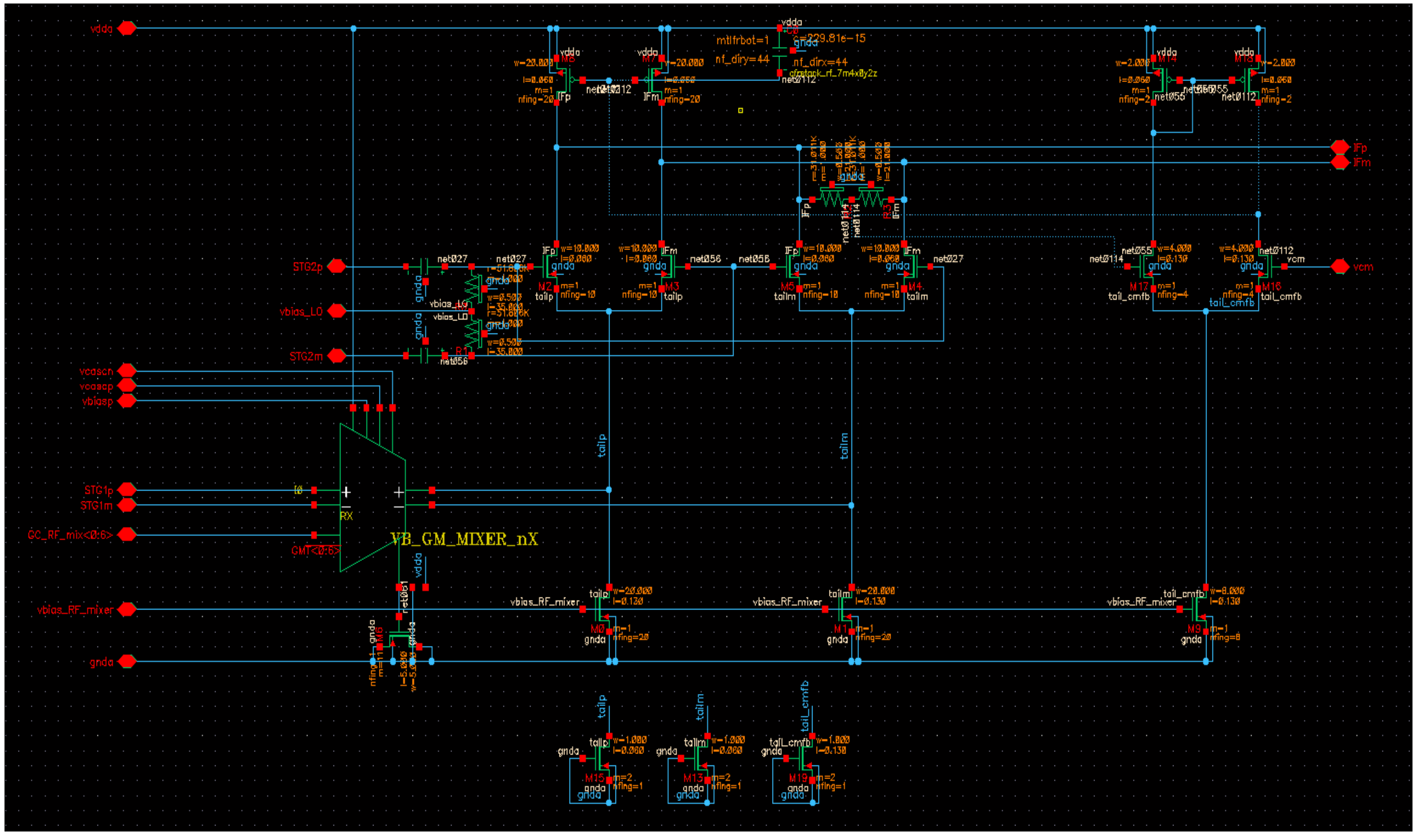
Circuit Schematic: Self-Mixer

VDD

Outputs

Inputs

Bias Control



VSS

Note: These guidelines matter even more in *open-source designs* where you want others to *reuse* your design.

CMOS Process

CMOS Technology & Analog Design Tutorials

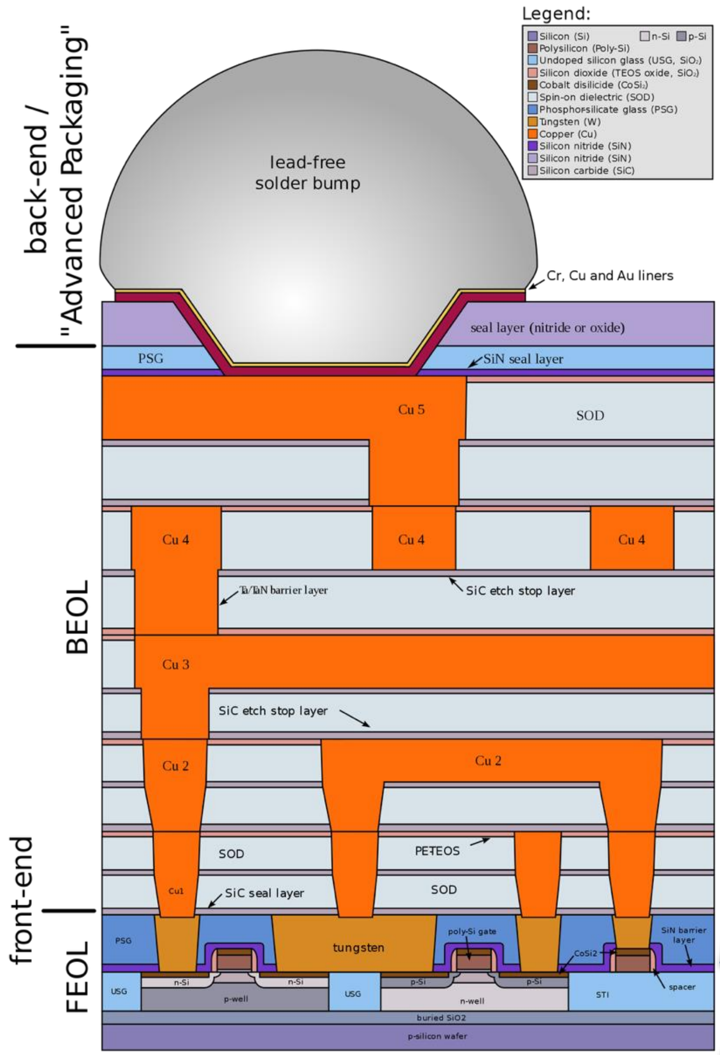
- Excellent tutorials available from https://aicdesign.org/2016-short-course-notes-2/
  - Recommended Reading:
    - Lecture 3: CMOS Technologies
    - Lecture 4: Ultra-Deep Submicron CMOS (BiCMOS is 'of interest')
    - Lecture 5: PN Junctions and CMOS Transistors
    - Lecture 6: Capacitors (focus on conductor-insulator-conductor caps)
    - Lecture 7: Resistors & Inductors (focus on resistors)
  - Of interest:
    - Lecture 1: Introduction on Analog Design
    - Lecture 2: CMOS Processing Steps
  - Note: some of the material is not fully up to date, but the slides still provide a valuable overview

Some Cross Sections

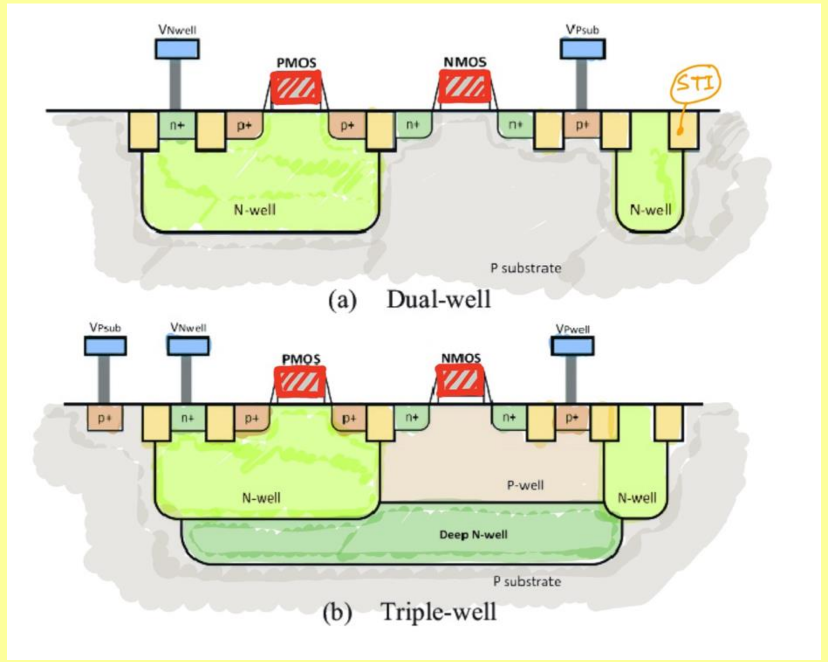
- These are some **public-domain images** and are not for the specific technology we are using; they give a good initial idea though.

See also: https://en.wikipedia.org/wiki/Semiconductor_device_fabrication

Note, in our process:
- there are more metals;
- there is an AP layer;
- there is no buried Si02.



[From Wikipedia]()

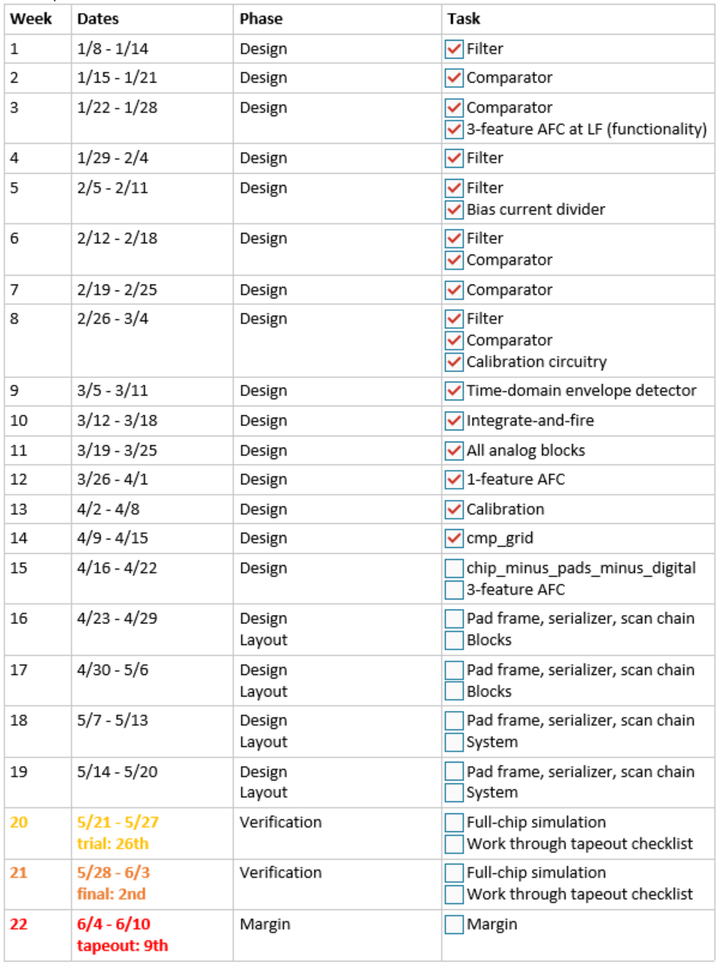


[From ResearchGate]() (coloring by PK)

Tracking Progress

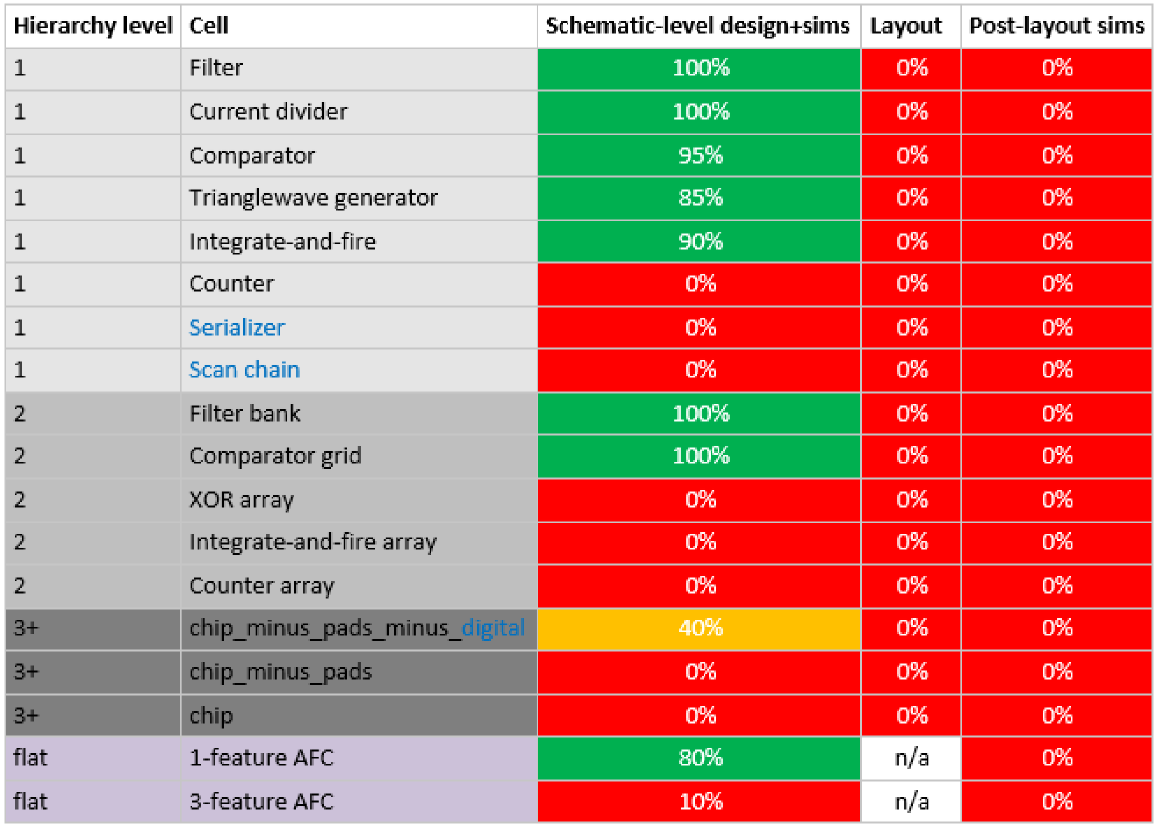
Tracking Progress

Week-by-week Schedule



Week	Dates	Phase	Task
1	1/8 - 1/14	Design	✓ Filter
2	1/15 - 1/21	Design	✓ Comparator
3	1/22 - 1/28	Design	✓ Comparator ✓ 3-feature AFC at LF (functionality)
4	1/29 - 2/4	Design	✓ Filter
5	2/5 - 2/11	Design	✓ Filter ✓ Bias current divider
6	2/12 - 2/18	Design	✓ Filter ✓ Comparator
7	2/19 - 2/25	Design	✓ Comparator
8	2/26 - 3/4	Design	✓ Filter ✓ Comparator ✓ Calibration circuitry
9	3/5 - 3/11	Design	✓ Time-domain envelope detector
10	3/12 - 3/18	Design	✓ Integrate-and-fire
11	3/19 - 3/25	Design	✓ All analog blocks
12	3/26 - 4/1	Design	✓ 1-feature AFC
13	4/2 - 4/8	Design	✓ Calibration
14	4/9 - 4/15	Design	✓ cmp_grid
15	4/16 - 4/22	Design	☐ chip_minus_pads_minus_digital ☐ 3-feature AFC
16	4/23 - 4/29	Design Layout	☐ Pad frame, serializer, scan chain ☐ Blocks
17	4/30 - 5/6	Design Layout	☐ Pad frame, serializer, scan chain ☐ Blocks
18	5/7 - 5/13	Design Layout	☐ Pad frame, serializer, scan chain ☐ System
19	5/14 - 5/20	Design Layout	☐ Pad frame, serializer, scan chain ☐ System
20	5/21 - 5/27 trial: 26th	Verification	☐ Full-chip simulation ☐ Work through tapeout checklist
21	5/28 - 6/3 final: 2nd	Verification	☐ Full-chip simulation ☐ Work through tapeout checklist
22	6/4 - 6/10 tapeout: 9th	Margin	☐ Margin

Progress Summary



Hierarchy level	Cell	Schematic-level design+sims	Layout	Post-layout sims
1	Filter	100%	0%	0%
1	Current divider	100%	0%	0%
1	Comparator	95%	0%	0%
1	Trianglewave generator	85%	0%	0%
1	Integrate-and-fire	90%	0%	0%
1	Counter	0%	0%	0%
1	Serializer	0%	0%	0%
1	Scan chain	0%	0%	0%
2	Filter bank	100%	0%	0%
2	Comparator grid	100%	0%	0%
2	XOR array	0%	0%	0%
2	Integrate-and-fire array	0%	0%	0%
2	Counter array	0%	0%	0%
3+	chip_minus_pads_minus_digital	40%	0%	0%
3+	chip_minus_pads	0%	0%	0%
3+	chip	0%	0%	0%
flat	1-feature AFC	80%	n/a	0%
flat	3-feature AFC	10%	n/a	0%

S. Ray and P. R. Kinget, "Ultra-Low-Power and Compact-Area Analog Audio Feature Extraction Based on Time-Mode Analog Filterbank Interpolation and Time-Mode Analog Rectification," *IEEE Journal of Solid-State Circuits*, vol. 58, no. 4, pp. 1025–1036, Apr. 2023, doi: 10.1109/JSSC.2022.3227246.

Review and update at least once a week

Simulations

Simulations

- Document and save your results

- A simulation you do not document *never happened*.

MOSbius Switch Matrix Library

!! Work in Progress !!

https://github.com/mosbiuschip/switch_matrix_gf180mcu_9t5v0

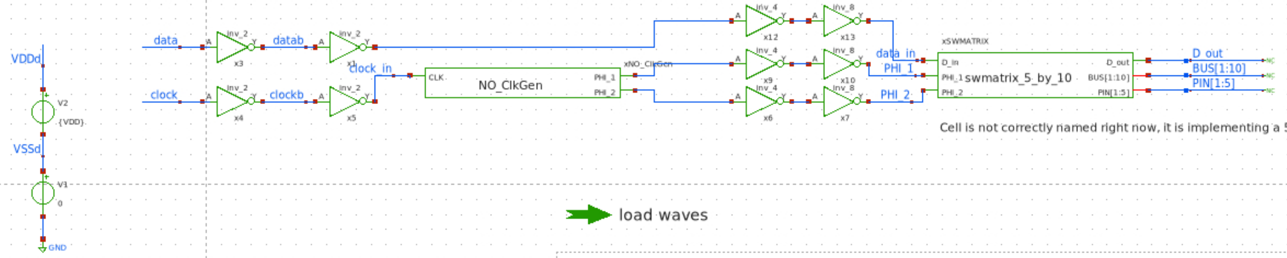
cells

```
switch_matrix_gf180mcu_9t5v0/
├── DFF_2phase_1
│   ├── DFF_2phase_1.sch
│   └── DFF_2phase_1.sym
├── NO_ClkGen
│   ├── NO_ClkGen.sch
│   └── NO_ClkGen.sym
├── ShiftReg_row_10
│   ├── ShiftReg_row_10.sch
│   └── ShiftReg_row_10.sym
├── ShiftReg_row_10_2
│   ├── ShiftReg_row_10_2.sch
│   └── ShiftReg_row_10_2.sym
├── docs
│   ├── saved_simulations
│   │   ├── data_swmatrix5_10.csv
│   │   ├── random.csv
│   │   ├── tb_swmatrix_5_10_20250720b.raw.save
│   │   └── tb_swmatrix_random_20250720b.raw.save
│   ├── simulation_analysis.ipynb
│   ├── simulation_analysis_5_10.html
│   └── simulation_analysis_random.html
├── swmatrix_5_by_10
│   ├── swmatrix_5_by_10.sch
│   └── swmatrix_5_by_10.sym
├── swmatrix_Tgate
│   ├── swmatrix_Tgate.sch
│   └── swmatrix_Tgate.sym
├── swmatrix_row_10
│   ├── swmatrix_row_10.sch
│   └── swmatrix_row_10.sym
└── testbenches
    ├── data_source
    │   ├── data_swmatrix5_10.csv
    │   ├── data_swmatrix5_10.txt
    │   ├── data_swmatrix5_10_clk.txt
    │   ├── generate_data_clock_files.py
    │   ├── random.csv
    │   ├── random.txt
    │   └── random_clk.txt
    ├── tb_ShiftReg_row_10.sch
    └── tb_swmatrix.sch
```

documentation

testbenches

tb_swmatrix.sch



The buffers on the PHI_1 and PHI_2 will need further optimizing, esp. if the matrix is increased in size; these are global signals with no internal buffering in the current (simple) implementation

The data_in does not need buffers but they have been inserted to approx. match the delay (out of an abundance of caution)

Cell is not correctly named right now, it is implementing a 5 by 10

load waves

Models

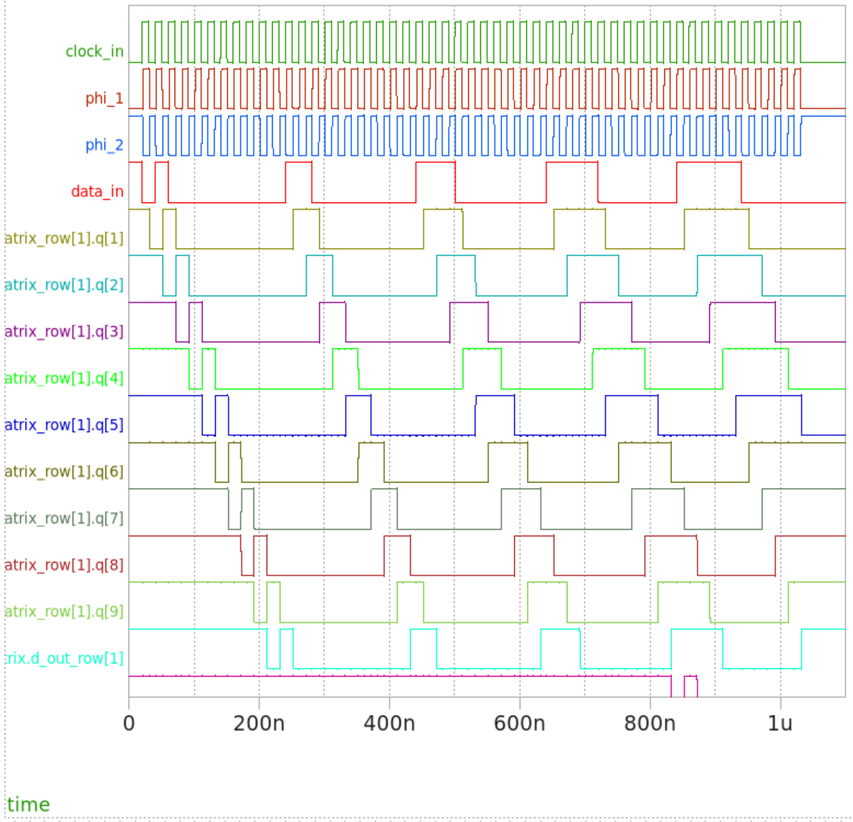
```
.include /foss/pdks/gf180mcuD/libs.ref/gf180mcu_fd_sc_mcu9t5v0/spice/gf180mcu_fd_sc_mcu9t5v0.spice
.include $::180MCU_MODELS/design.ngspice
.lib $::180MCU_MODELS/sm141064.ngspice typical
```

NETLIST

```
.param VDD = 3.3

.global VDDd VSSd

* clock
abit [ bit_node ]  input_vector
.model input_vector d_source(input_file="/foss/designs/libs/switch_matrix_gf180mcu_9t5v0/testbenches/data_source/data_swmatrix5_10.txt")
* data
aclock [ clock_node ] clock_vector
.model clock_vector d_source(input_file="/foss/designs/libs/switch_matrix_gf180mcu_9t5v0/testbenches/data_source/data_swmatrix5_10_clk.txt")
* convert digital signals to analog
aconvert [ bit_node clock_node ] [ data clock ] dac_in
.model dac_in dac_bridge (out_low=0V out_high=3.3V t_rise=0.2ns t_fall=0.2ns)
```

Simulation

```
.control

    save all
    TRAN 0.2n 1100n
    write tb_swmatrix.raw

.endc
```

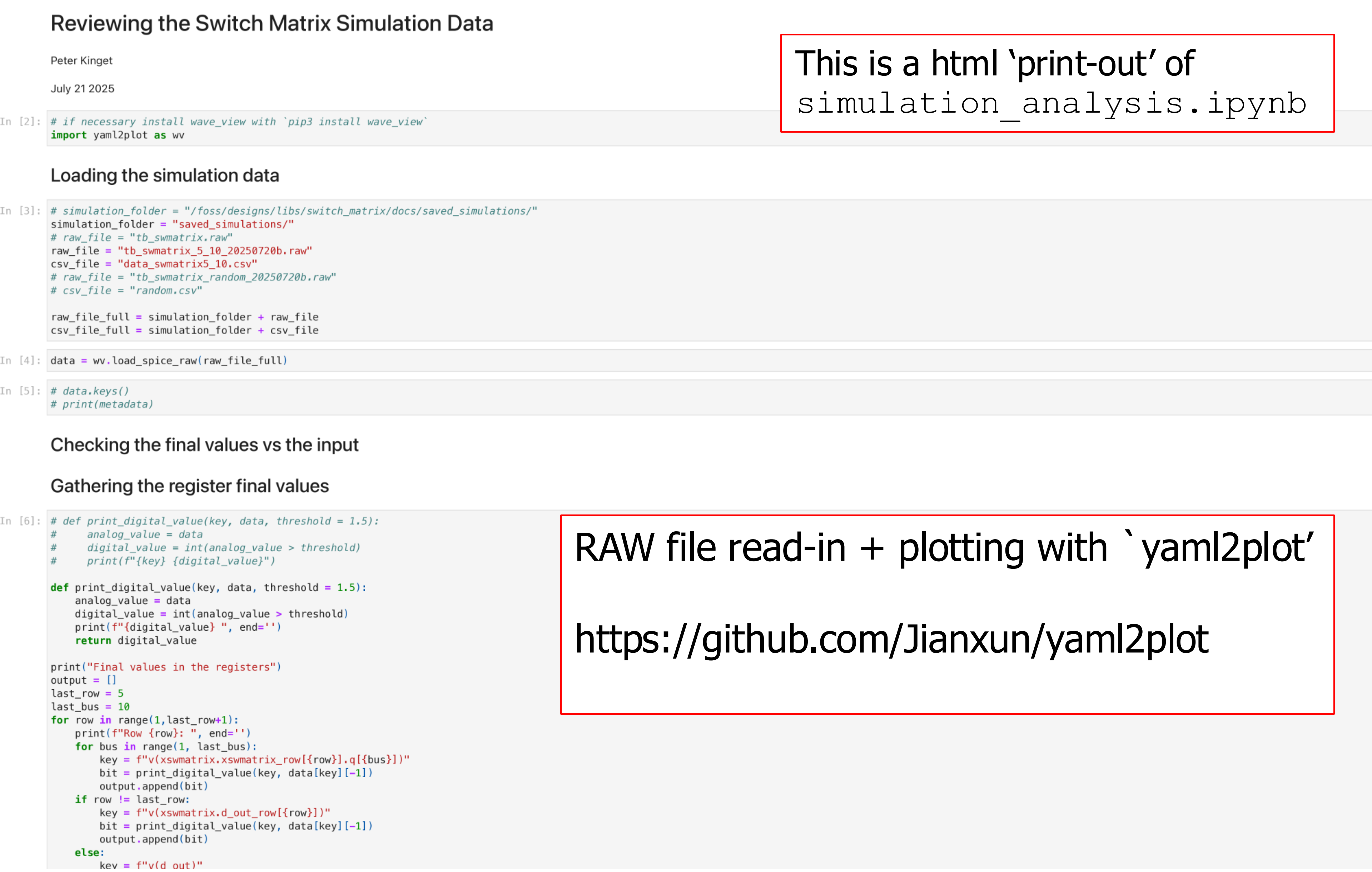


Testing a Switch Matrix with 5 by 10 registers
- 5 pins (rows) and 10 buses (columns) [can be expanded later]
- clock and data read from a file (see code block)
- for the clock and data interfaces, Schmitt Triggers need to be added
- typically we drive in a leading 0 followed by the 50 data bits
- the leading 0 makes sure all registers are first set to 0
- the last data bit appears at xsr.q[1] and the first at the 50th register which is D_out
- there is a python script in the data_source folder to generate source and clock txt files
- the transmission gates are empty cells for now

XSCHEM

Peter Kinget 2025-07-24 02:05:21
testbenches/tb_swmatrix.sch

simulation_analysis_5_10.html (1)



Reviewing the Switch Matrix Simulation Data

Peter Kinget

July 21 2025

```
In [2]: # if necessary install wave_view with `pip3 install wave_view`
        import yaml2plot as wv
```

Loading the simulation data

```
In [3]: # simulation_folder = "/foss/designs/libs/switch_matrix/docs/saved_simulations/"
        simulation_folder = "saved_simulations/"
        # raw_file = "tb_swmatrix.raw"
        raw_file = "tb_swmatrix_5_10_20250720b.raw"
        csv_file = "data_swmatrix5_10.csv"
        # raw_file = "tb_swmatrix_random_20250720b.raw"
        # csv_file = "random.csv"

        raw_file_full = simulation_folder + raw_file
        csv_file_full = simulation_folder + csv_file
```

```
In [4]: data = wv.load_spice_raw(raw_file_full)
```

```
In [5]: # data.keys()
        # print(metadata)
```

Checking the final values vs the input

Gathering the register final values

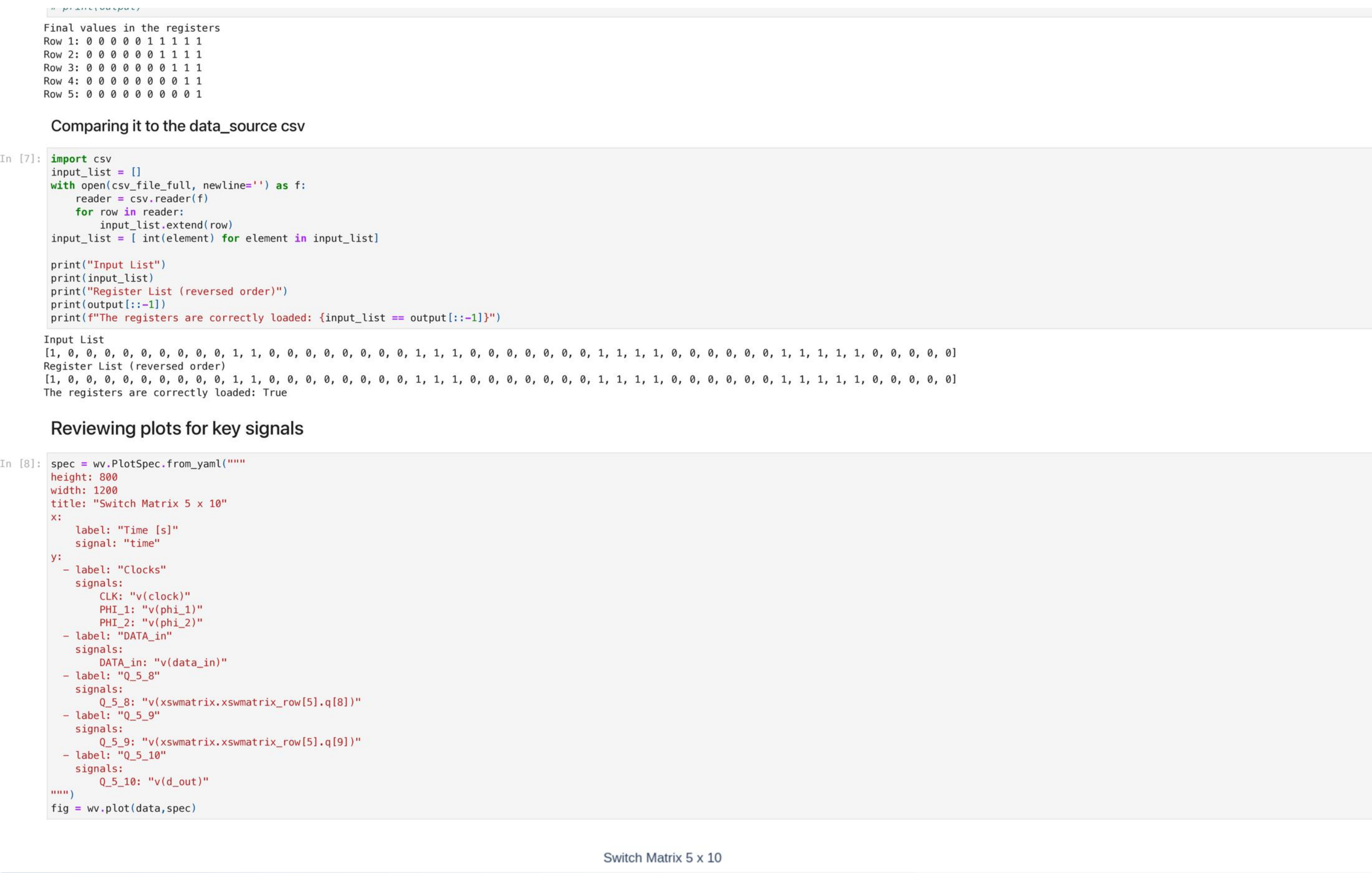
```
In [6]: # def print_digital_value(key, data, threshold = 1.5):
        #     analog_value = data
        #     digital_value = int(analog_value > threshold)
        #     print(f"{key} {digital_value}")

        def print_digital_value(key, data, threshold = 1.5):
            analog_value = data
            digital_value = int(analog_value > threshold)
            print(f"{digital_value} ", end='')
            return digital_value

        print("Final values in the registers")
        output = []
        last_row = 5
        last_bus = 10
        for row in range(1,last_row+1):
            print(f"Row {row}: ", end='')
            for bus in range(1, last_bus):
                key = f"v(xswmatrix.xswmatrix_row[{row}].q[{bus}])"
                bit = print_digital_value(key, data[key][-1])
                output.append(bit)
            if row != last_row:
                key = f"v(xswmatrix.d_out_row[{row}])"
                bit = print_digital_value(key, data[key][-1])
                output.append(bit)
            else:
                key = f"v(d_out)"
```

This is a html 'print-out' of `simulation_analysis.ipynb`

RAW file read-in + plotting with `yaml2plot'

https://github.com/Jianxun/yaml2plot

simulation_analysis_5_10.html (2)



```
# print(output)
```

```
Final values in the registers
Row 1: 0 0 0 0 0 1 1 1 1 1
Row 2: 0 0 0 0 0 0 1 1 1 1
Row 3: 0 0 0 0 0 0 0 1 1 1
Row 4: 0 0 0 0 0 0 0 0 1 1
Row 5: 0 0 0 0 0 0 0 0 0 1
```

Comparing it to the data_source csv

In [7]:
```
import csv
input_list = []
with open(csv_file_full, newline='') as f:
    reader = csv.reader(f)
    for row in reader:
        input_list.extend(row)
input_list = [ int(element) for element in input_list]

print("Input List")
print(input_list)
print("Register List (reversed order)")
print(output[::-1])
print(f"The registers are correctly loaded: {input_list == output[::-1]}")
```

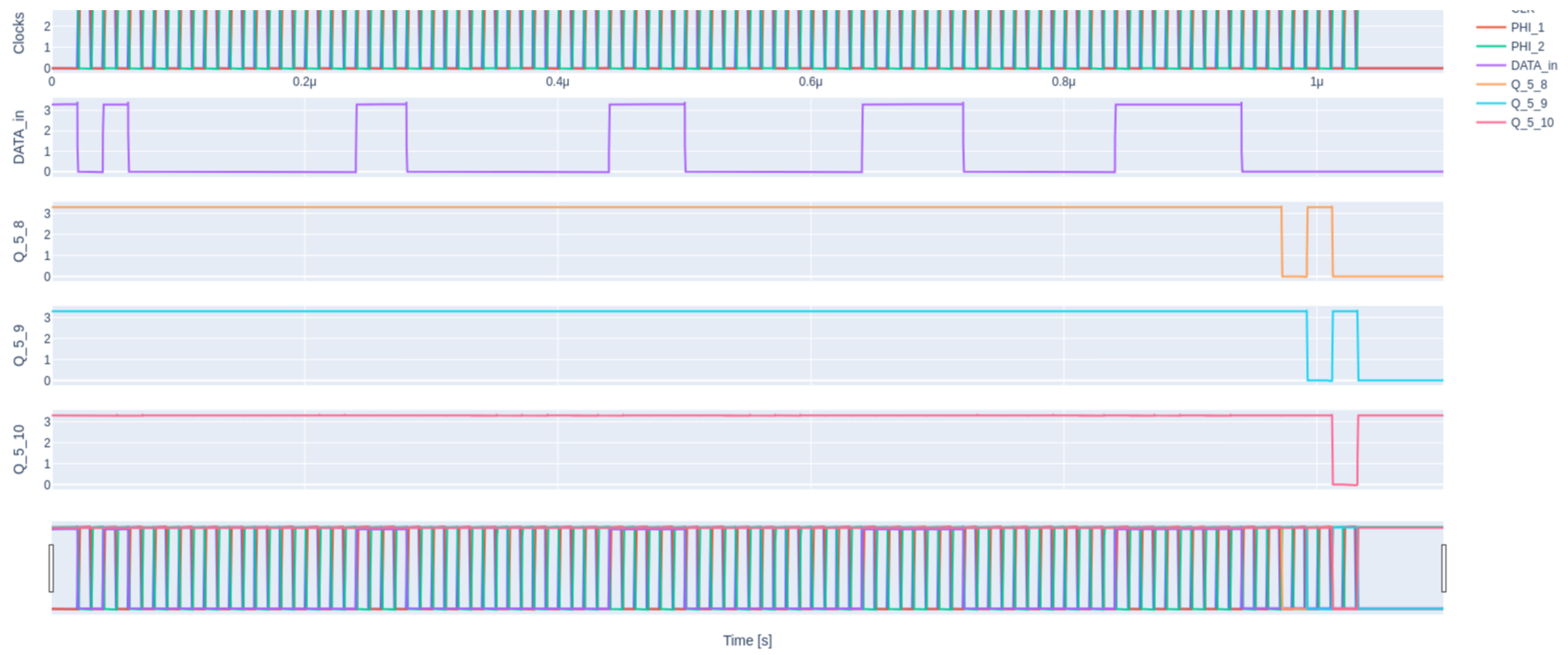
```
Input List
[1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0]
Register List (reversed order)
[1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0]
The registers are correctly loaded: True
```

Reviewing plots for key signals

In [8]:
```
spec = wv.PlotSpec.from_yaml("""
height: 800
width: 1200
title: "Switch Matrix 5 x 10"
x:
    label: "Time [s]"
    signal: "time"
y:
  - label: "Clocks"
    signals:
        CLK: "v(clock)"
        PHI_1: "v(phi_1)"
        PHI_2: "v(phi_2)"
  - label: "DATA_in"
    signals:
        DATA_in: "v(data_in)"
  - label: "Q_5_8"
    signals:
        Q_5_8: "v(xswmatrix.xswmatrix_row[5].q[8])"
  - label: "Q_5_9"
    signals:
        Q_5_9: "v(xswmatrix.xswmatrix_row[5].q[9])"
  - label: "Q_5_10"
    signals:
        Q_5_10: "v(d_out)"
""")
fig = wv.plot(data,spec)
```

Switch Matrix 5 x 10

simulation_analysis_5_10.html (3)

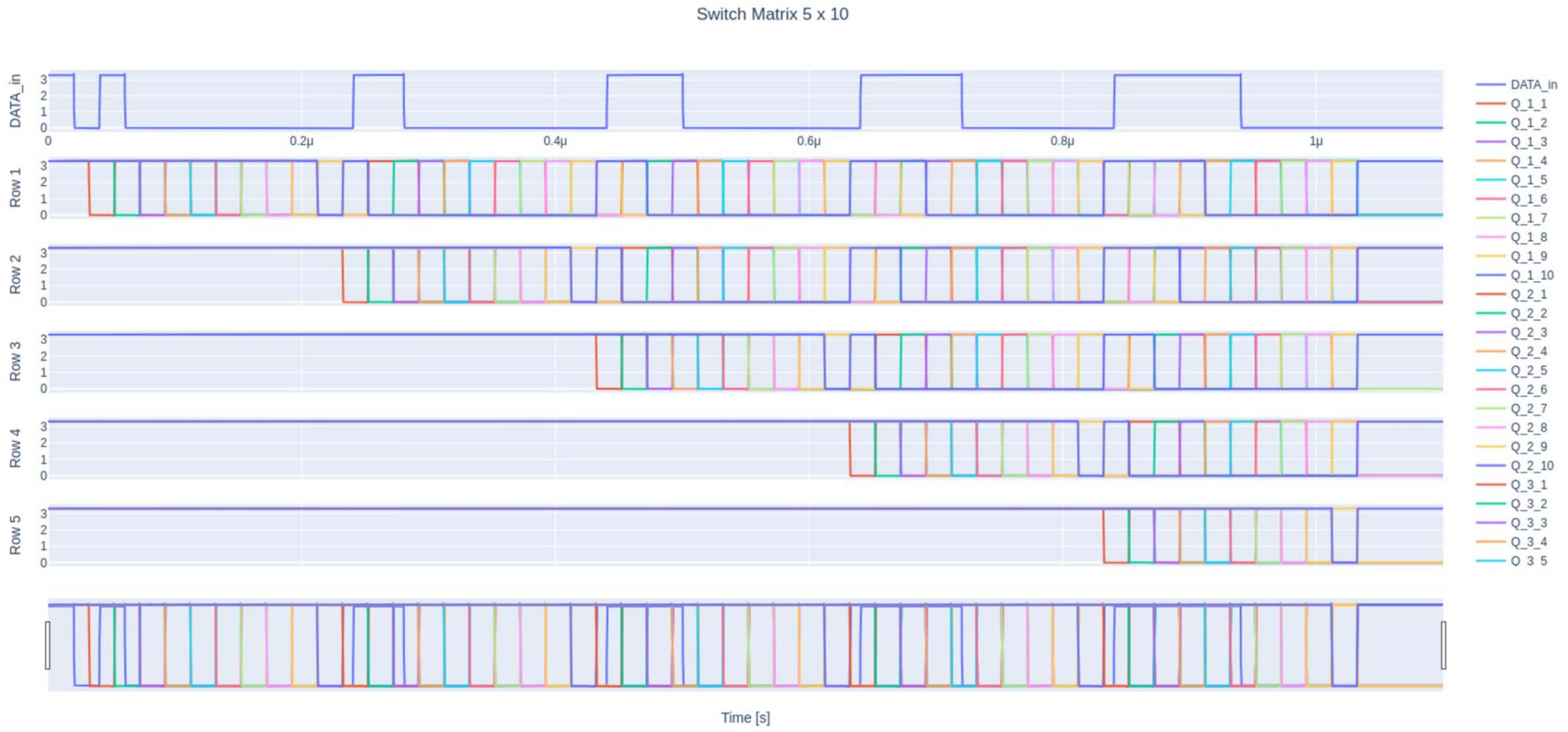


Plotting all Register outputs

```
In [9]: def add_row_label(plot_spec, row):
            plot_spec = plot_spec + f'  - label: "Row {row}"\n    signals:\n'
            return plot_spec

        def add_signal(plot_spec, key, name):
            plot_spec = plot_spec + f'        {name}: "{key}"\n'
            return plot_spec

        plot_spec = """
        height: 800
        width: 1200
        title: "Switch Matrix 5 x 10"
        x:
            label: "Time [s]"
            signal: "time"
        y:
          - label: "DATA_in"
            signals:
                DATA_in: "v(data_in)"
```

simulation_analysis_5_10.html (4)

```python
for row in range(1,last_row+1):
    plot_spec = add_row_label(plot_spec,row)
    for bus in range(1, last_bus):
        key = f"v(xswmatrix.xswmatrix_row[{row}].q[{bus}])"
        plot_spec = add_signal(plot_spec, key, f"Q_{row}_{bus}")
    if row != last_row:
        key = f"v(xswmatrix.d_out_row[{row}])"
        plot_spec = add_signal(plot_spec, key, f"Q_{row}_{last_bus}")
    else:
        key = f"v(d_out)"
        plot_spec = add_signal(plot_spec, key, f"Q_{row}_{last_bus}")

# print(plot_spec)
fig = wv.plot(data,wv.PlotSpec.from_yaml(plot_spec))
```

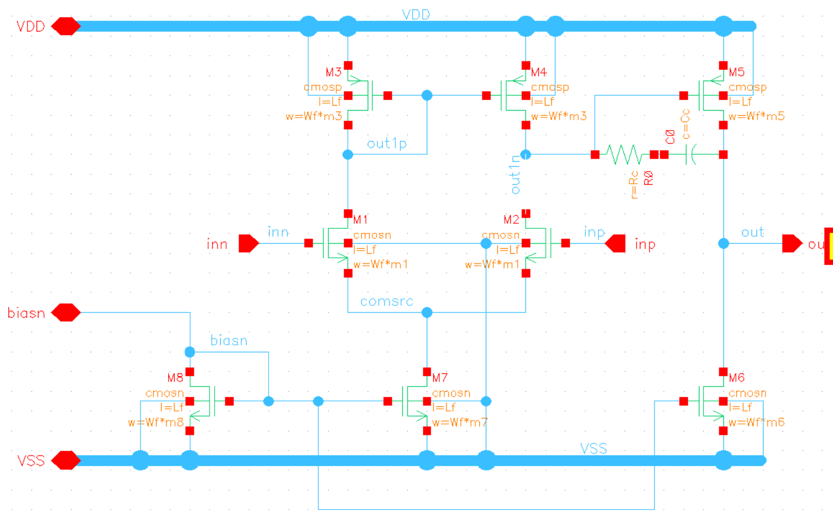


In [ ]:

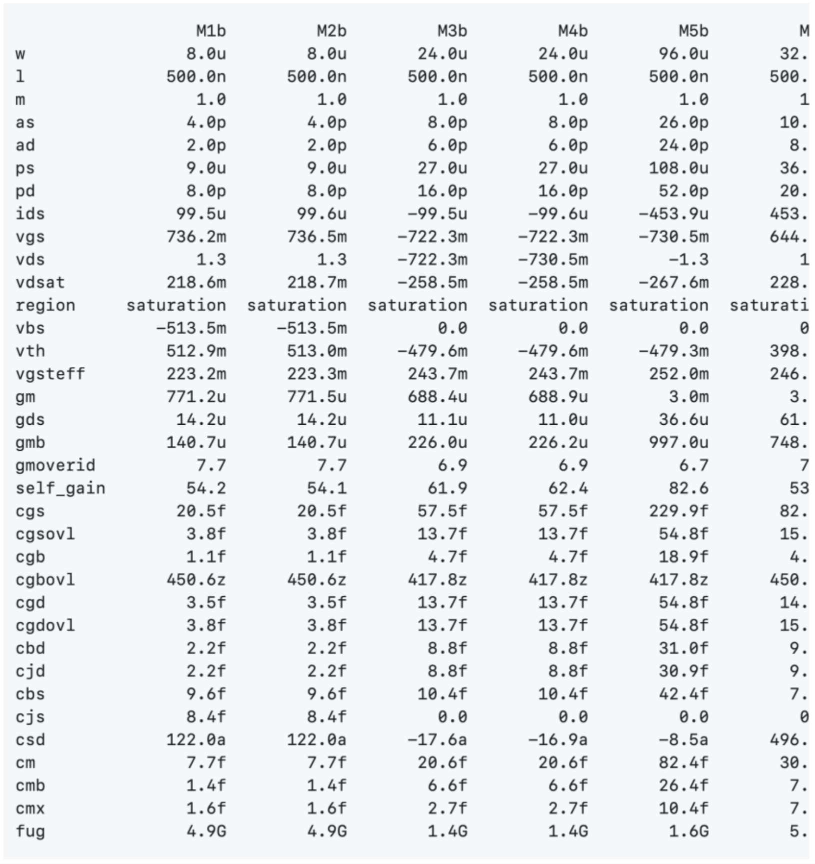
Analog Design

- Check your operating point first
  - You have heard it **twice** now, last week Prof. Murmann stressed this as well.

- https://github.com/peterkinget/MOS_OP/tree/make_installable



Needs some updates to work with ngspice; start from the LTspice code; contact me if you are willing to help


```
                M1b         M2b         M3b         M4b         M5b          M
w              8.0u        8.0u       24.0u       24.0u       96.0u        32.
l            500.0n      500.0n      500.0n      500.0n      500.0n       500.
m               1.0         1.0         1.0         1.0         1.0          1
as             4.0p        4.0p        8.0p        8.0p       26.0p        10.
ad             2.0p        2.0p        6.0p        6.0p       24.0p         8.
ps             9.0u        9.0u       27.0u       27.0u      108.0u        36.
pd             8.0p        8.0p       16.0p       16.0p       52.0p        20.
ids           99.5u       99.6u      -99.5u      -99.6u     -453.9u       453.
vgs          736.2m      736.5m     -722.3m     -722.3m     -730.5m       644.
vds             1.3         1.3     -722.3m     -730.5m        -1.3          1
vdsat        218.6m      218.7m     -258.5m     -258.5m     -267.6m       228.
region   saturation  saturation  saturation  saturation  saturation  saturati
vbs         -513.5m     -513.5m         0.0         0.0         0.0          0
vth          512.9m      513.0m     -479.6m     -479.6m     -479.3m       398.
vgsteff      223.2m      223.3m      243.7m      243.7m      252.0m       246.
gm           771.2u      771.5u      688.4u      688.9u        3.0m         3.
gds           14.2u       14.2u       11.1u       11.0u       36.6u        61.
gmb          140.7u      140.7u      226.0u      226.2u      997.0u       748.
gmoverid        7.7         7.7         6.9         6.9         6.7          7
self_gain      54.2        54.1        61.9        62.4        82.6         53
cgs           20.5f       20.5f       57.5f       57.5f      229.9f        82.
cgsovl         3.8f        3.8f       13.7f       13.7f       54.8f        15.
cgb            1.1f        1.1f        4.7f        4.7f       18.9f         4.
cgbovl       450.6z      450.6z      417.8z      417.8z      417.8z       450.
cgd            3.5f        3.5f       13.7f       13.7f       54.8f        14.
cgdovl         3.8f        3.8f       13.7f       13.7f       54.8f        15.
cbd            2.2f        2.2f        8.8f        8.8f       31.0f         9.
cjd            2.2f        2.2f        8.8f        8.8f       30.9f         9.
cbs            9.6f        9.6f       10.4f       10.4f       42.4f         7.
cjs            8.4f        8.4f         0.0         0.0         0.0          0
csd          122.0a      122.0a      -17.6a      -16.9a       -8.5a       496.
cm             7.7f        7.7f       20.6f       20.6f       82.4f        30.
cmb            1.4f        1.4f        6.6f        6.6f       26.4f         7.
cmx            1.6f        1.6f        2.7f        2.7f       10.4f         7.
fug            4.9G        4.9G        1.4G        1.4G        1.6G         5.
```

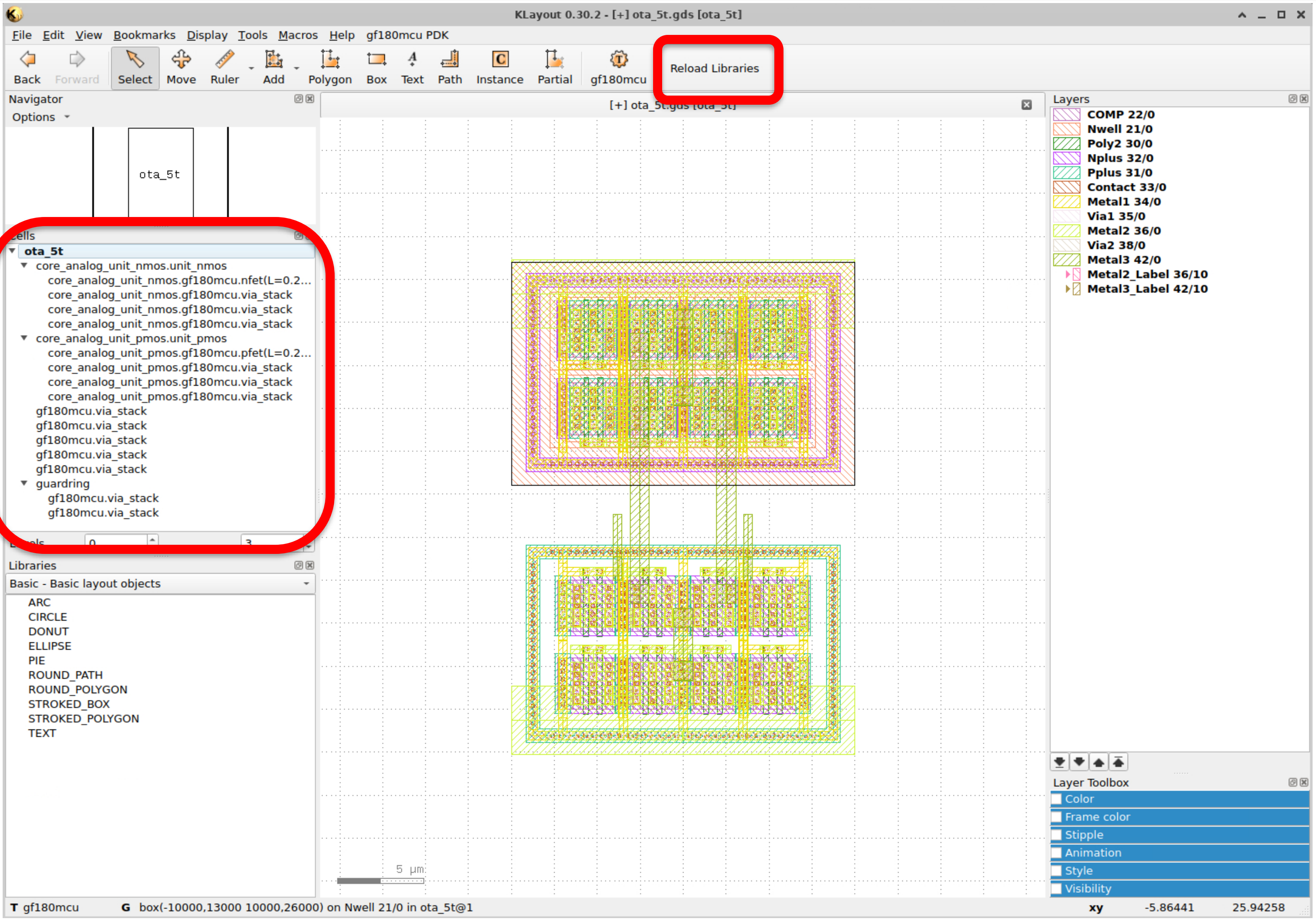
…

GitHub Repo Template

iic-osic-tools-project-template

- Jianxun Zhu and I are creating a GitHub Design Repo Template https://github.com/Jianxun/iic-osic-tools-project-template.   !!! Work in Progress !!!
- Features:
  - `start_chipathon.{sh,bat}`
    - First time use → installs IIC-OSIC-TOOLS docker image
  - `update_template.sh`
    - Pulls template updates
  - `docs/`
    - HOWTOs on schematic, layout, github workflows
  - `designs/scripts`
    - `Klayout` launch script with database manager !!! Caution, this has not been battle tested!!!
    - `GDS3D` launch script
  - `designs/libs`
    - `Ota_5t` example with LVS clean layout

Klayout with `klayout_lib_manager.py` for `ota_5t.gds`



KLayout 0.30.2 - [+] ota_5t.gds [ota_5t]

File Edit View Bookmarks Display Tools Macros Help gf180mcu PDK

Back Forward Select Move Ruler Add Polygon Box Text Path Instance Partial gf180mcu Reload Libraries

Cells

```
▼ ota_5t
  ▼ core_analog_unit_nmos.unit_nmos
      core_analog_unit_nmos.gf180mcu.nfet(L=0.2...
      core_analog_unit_nmos.gf180mcu.via_stack
      core_analog_unit_nmos.gf180mcu.via_stack
      core_analog_unit_nmos.gf180mcu.via_stack
  ▼ core_analog_unit_pmos.unit_pmos
      core_analog_unit_pmos.gf180mcu.pfet(L=0.2...
      core_analog_unit_pmos.gf180mcu.via_stack
      core_analog_unit_pmos.gf180mcu.via_stack
      core_analog_unit_pmos.gf180mcu.via_stack
    gf180mcu.via_stack
    gf180mcu.via_stack
    gf180mcu.via_stack
    gf180mcu.via_stack
    gf180mcu.via_stack
  ▼ guardring
      gf180mcu.via_stack
      gf180mcu.via_stack
```

Libraries: Basic - Basic layout objects

ARC, CIRCLE, DONUT, ELLIPSE, PIE, ROUND_PATH, ROUND_POLYGON, STROKED_BOX, STROKED_POLYGON, TEXT

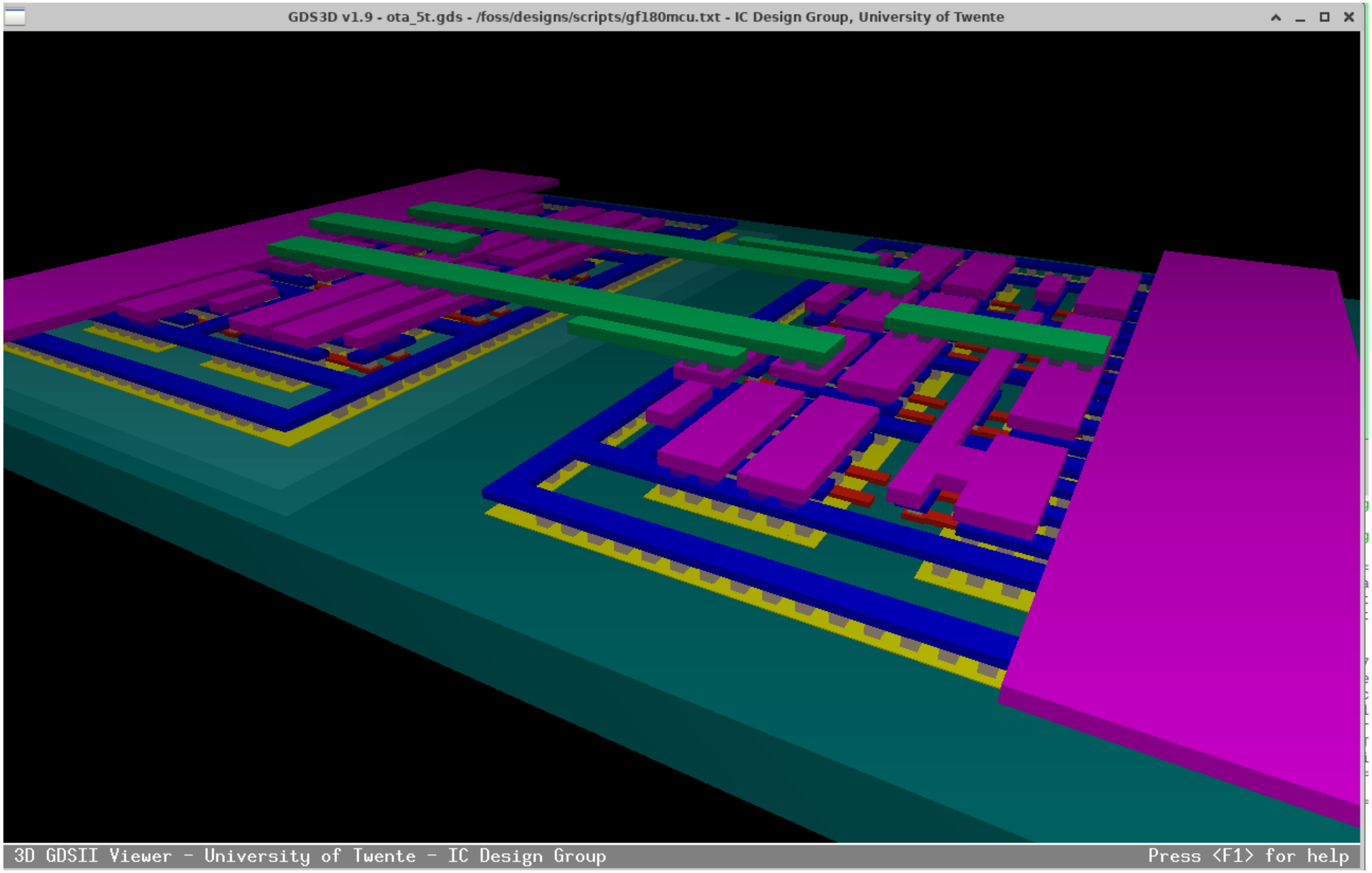
Layers

- COMP 22/0
- Nwell 21/0
- Poly2 30/0
- Nplus 32/0
- Pplus 31/0
- Contact 33/0
- Metal1 34/0
- Via1 35/0
- Metal2 36/0
- Via2 38/0
- Metal3 42/0
- Metal2_Label 36/10
- Metal3_Label 42/10

Layer Toolbox: Color, Frame color, Stipple, Animation, Style, Visibility

```
T gf180mcu    G  box(-10000,13000 10000,26000) on Nwell 21/0 in ota_5t@1        xy  -5.86441   25.94258
```

GDS3D of `ota_5t.gds`


GDS3D v1.9 - ota_5t.gds - /foss/designs/scripts/gf180mcu.txt - IC Design Group, University of Twente
3D GDSII Viewer - University of Twente - IC Design Group
Press <F1> for help

Want to learn the flow?

- Design a 5 stage ring oscillator with o/p buffer driving 1pF
- Schematic → simulation → layout → LVS → PEX → re-simulate

- Will give you a good sense of the tools + the impact of parasitcs for ‘high speed’ circuits

Ask Questions !!

- File GitHub issues on https://github.com/sscs-ose/sscs-chipathon-2025/issues
  - We are giving feedback there so it is preserved
- MOSbius Chipathon Documentation including these slides:
  - https://github.com/mosbiuschip/chipathon2025
- Ask questions on fossi-chat.org
- Let’s collaborate !!