TFE4188 - Lecture 1

Analog design fundamentals

Week	Book	Monday	Project plan
2	CJM 1-6	Course intro, what I expect you to know, project, analog design fundamentals	Specification
3	Slides	ESD and IC Input/Output	Specification
4	CJM 7,8	Reference and bias	M1. Specification review
5	CJM 12	Analog Front-end	Design
6	CJM 11-14	Switched capacitor circuits	Design
7	JSSC, CJM 18	State-of-the-art ADCs	Design
8	Slides	Low power radio recievers	M2. Design review
9	Slides	Communication standards from circuit perspective	Layout
10	CJM 7.4, CFAS,+DC/DC	Voltage regulation	Layout
11	CJM 19, CFAS	Clock generation	M3. Layout DRC/LVS clean
12	Paper	Energy sources	Layout Parasitic Extracted simulation
13	Slides	Chip infrastructure	Layout Parasitic Extracted simulation
14		Tapeout review	M4. Tapeout review
15		Easter	
16		Easter	
17		Exam repetition	

Goal for today

Choosing transistor sizes is **complicated**

How to make **state-of-the-art** designs

Recommendations for transistor sizes

Complicated

Analog Design Process

- Define the problem, what are you trying to solve?
- Find a circuit that can solve the problem (papers, books)
- **Find right transistor sizes. What transistors should be weak inversion, strong inversion, or don't care?**
- Check operating region of transistors (.op)
- Check key parameters (.dc, .ac, .tran)
- Check function. Exercise all inputs. Check all control signals
- Check key parameters in all corners. Check mismatch (Monte-Carlo simulation)
- Do layout, and check it's error free. Run design rule checks (DRC). Check layout versus schematic (LVS)
- Extract parasitics from layout. Resistance, capacitance, and inductance if necessary.
- On extracted parasitic netlist, check key parameters in all corners and mismatch (if possible).
- If everything works, then your done.

On failure, go back as far as necessary

Assume active ($V_{ds} > V_{eff}$ in strong inversion, or $V_{ds} > 3V_T$ in weak inversion)

For diode connected transistors, this is always true

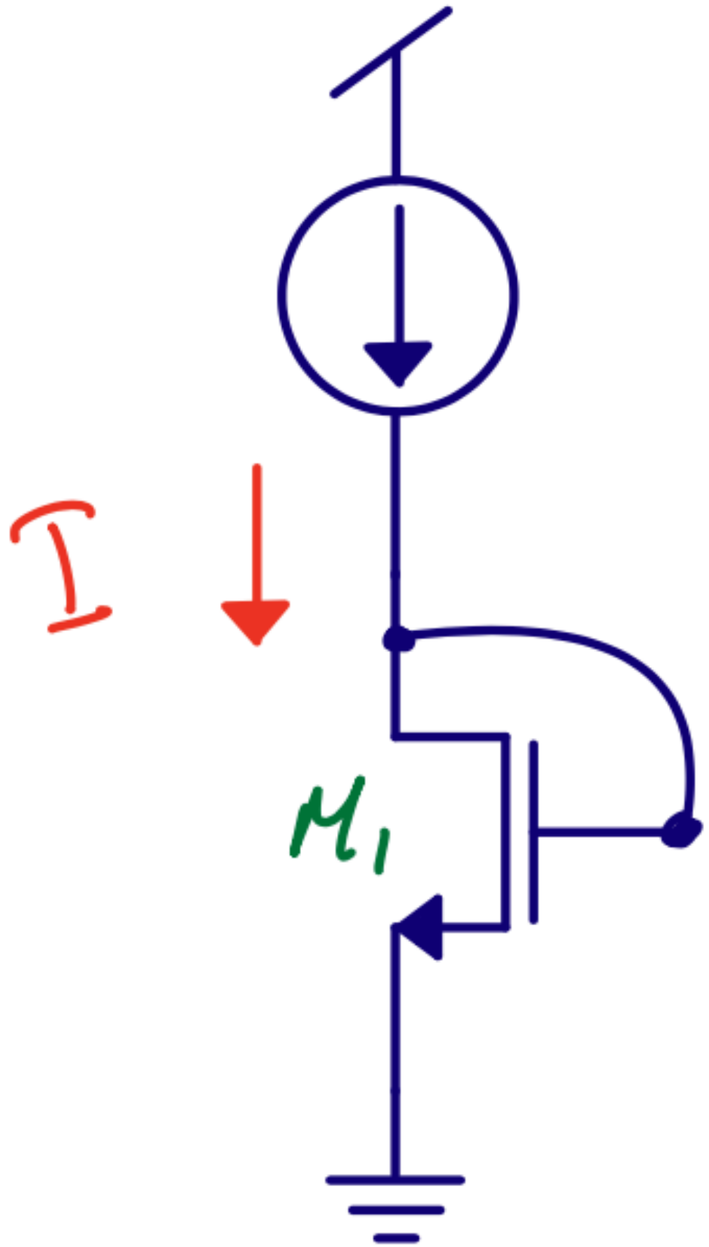
Weak inversion

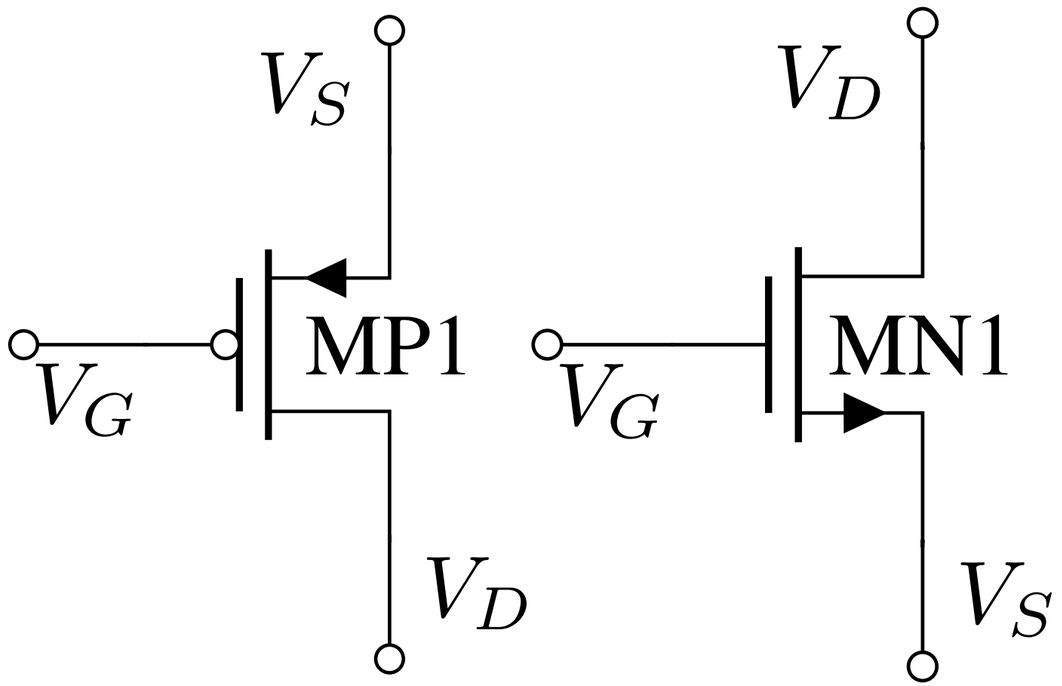
$$I_D = I_{D0}\frac{W}{L}e^{V_eff/nV_T}, V_{eff} \propto \ln I_D$$

Strong inversion

$$I_D = \frac{1}{2}\mu_n C_{ox}\frac{W}{L}V_{eff}^2, V_{eff} \propto \sqrt{I_D}$$

Operating region for a diode connected transistor only depends on the current





Flavors

1.5 V MOS (standard Vt, native Vt)

Parameter	Min	Max	Unit
L	0.13	100	um
W	0.15	50	um

Assume 5n quantization. Permutations per transistor (P)

$$P = 2 \times 19974 \times 9970 = 398\ \mathrm{M}$$

Typical ADC ~ 20 k transistors

$$P_{ADC} = 20\ \mathrm{k} \times 2 \times 398\ \mathrm{M} = 16\ \mathrm{P}$$

Too large solution space for exhaustive search

Picking transistor size

Not possible with exhaustive search

Simplify as much as possible

Use brain

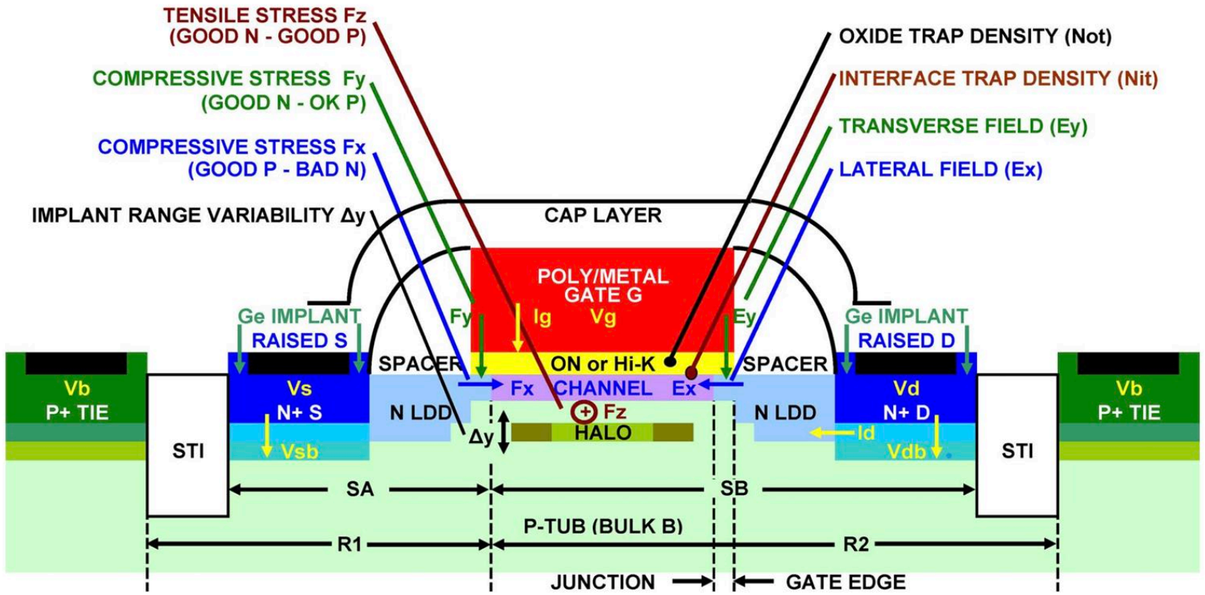


Fig. 2. *NMOS cross-section. In addition to stress from cap layers and Ge raised source-drain (S-D) implants, device dimensions such as distance from source-channel boundary to nearby STI (SA and SB), proximity and regularity of overlying metal patterns, and short distances to other device patterns within the local* ($< 2\ \mu m$) *stress field induce transverse* (F_y) *and lateral* ($F_{\underline{x}}$ *and* $F_{\underline{z}}$) *stress components, which affect threshold and mobility. Increasing the distance to P+ ties increases local tub (bulk) resistance components R1 and R2, which isolate the device MOS model substrate node from the device subcircuit symbol* V_b *node and degrade HF performance. Hot carrier reliability stress is dependent on the sum of transverse and lateral fields* E_Y *and* E_x*. These fields are increased near the drain by increasing source to bulk* (V_{sb}) *and drain* (V_d) *to gate* (V_g) *or source* (V_s) *voltages in various combinations. As hot carrier stress increases, damage to channel from interface trap density* (N_{it}) *affects threshold and mobility, while gate oxynitirde (ON) or high-dielectric-constant (Hi-K) insulator trap density* (N_{ot}) *affects threshold and gate leakage.*

State-of-the-art

How to make state-of-the-art designs

Know what is **known**

Find a good **problem** to solve

Find an **architecture** that could work

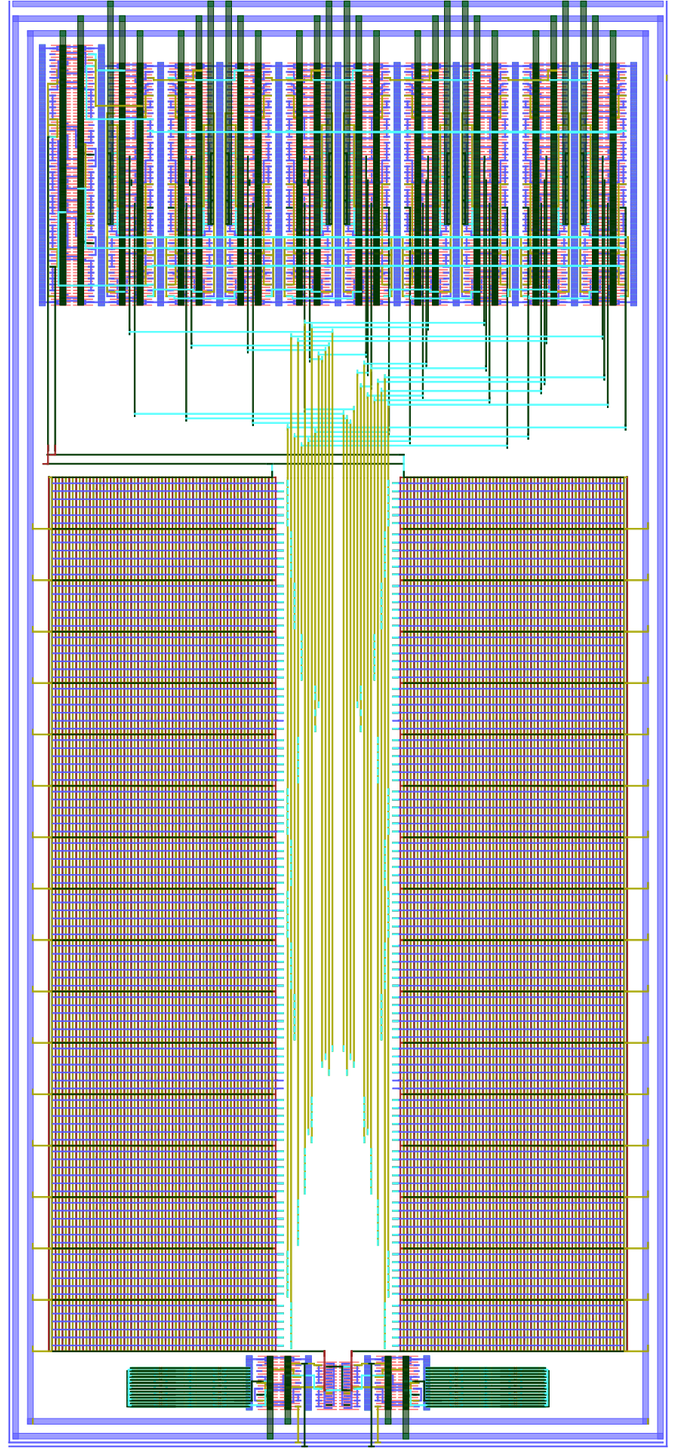
Work through all **important** details

If publishing, have some **luck**

There is no magic in state-of-the-art designs

However, a fast brain might get there faster. A slow brain may never reach the end.

My only published state-of-the-art design



Trigger

CONTRIBUTED PAPER

Analog Circuit Design in Nanoscale CMOS Technologies

Classic analog designs are being replaced by digital methods, using nanoscale digital devices, for calibrating circuits, overcoming device mismatches, and reducing bias and temperature dependence.

By Lanny L. Lewyn, *Life Senior Member IEEE*, Trond Ytterdal, *Senior Member IEEE*, Carsten Wulff, *Member IEEE*, and Kenneth Martin, *Fellow IEEE*

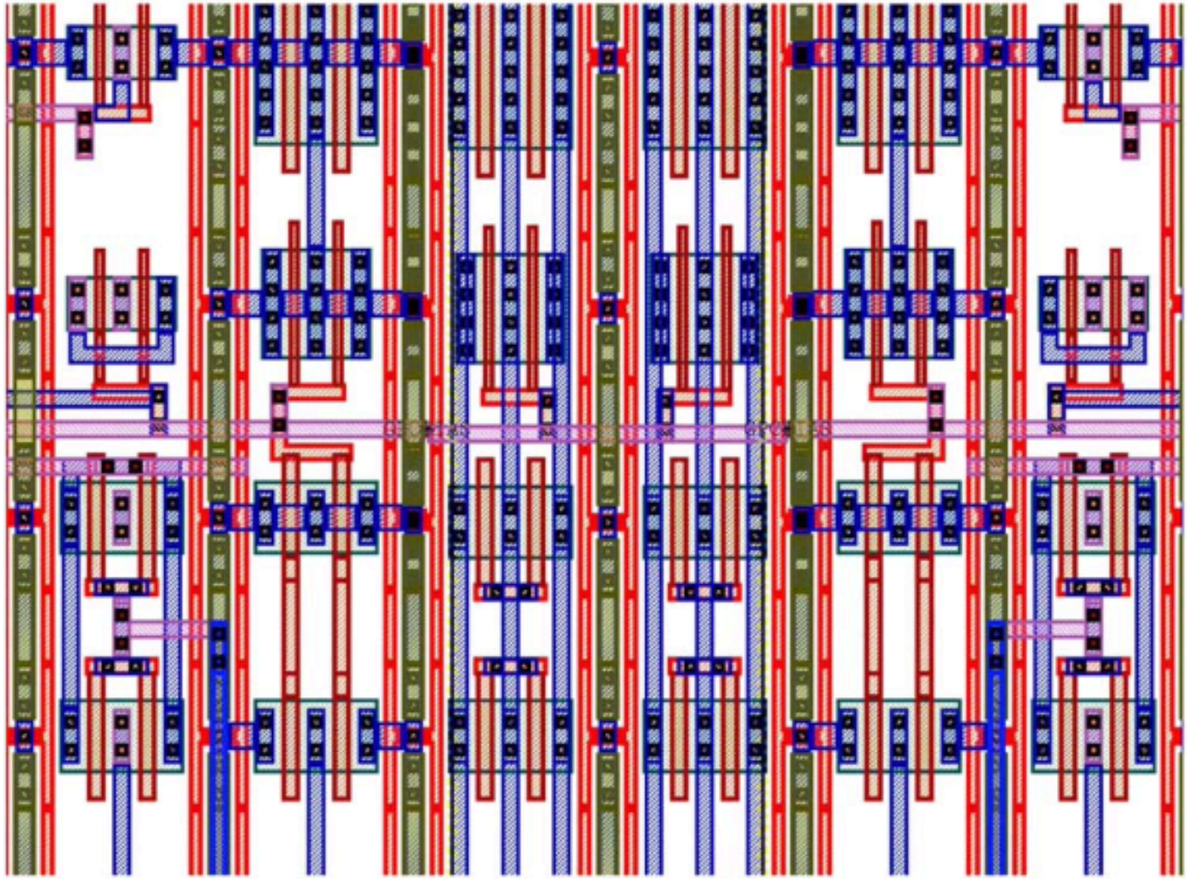
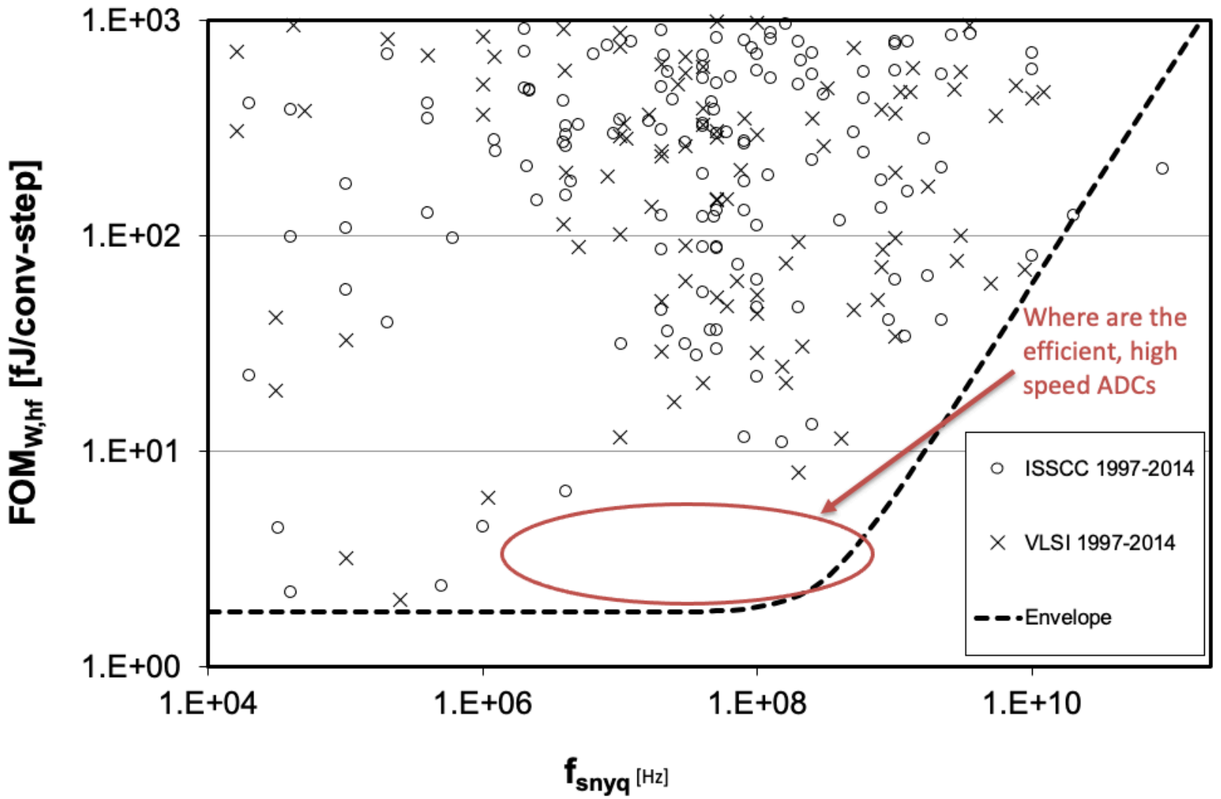
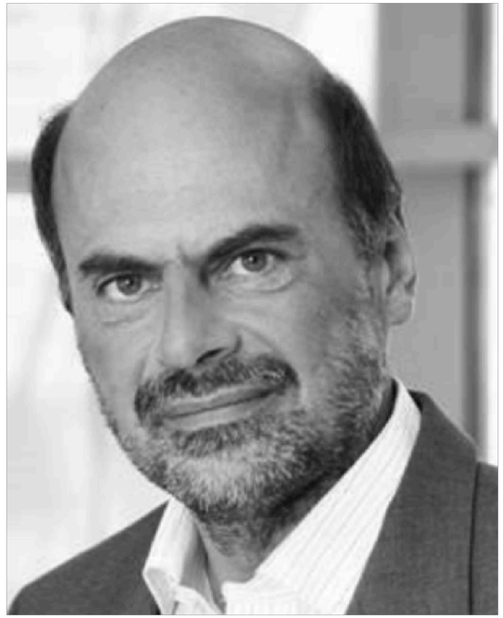


Fig. 6. *A portion of an amplifier cell with regular device pitch in both X and Y directions (upper metal layers removed for clarity). For best HF performance, all devices' substrate ties are placed on either side of two-finger gate patterns. Grounded stripes of poly are interposed between device active area and all substrate ties to minimize the need for reticle compensation (OPC) and also reduce poly etch loading to achieve good CD accuracy.*

Problem


1.E+03
1.E+02
1.E+01
1.E+00
1.E+04
1.E+06
1.E+08
1.E+10
FOM$_{W,hf}$ [fJ/conv-step]
f_{snyq} [Hz]
Where are the efficient, high speed ADCs
ISSCC 1997-2014
VLSI 1997-2014
Envelope

Architecture



14.7 A 6b 600MHz 10mW ADC Array in Digital 90nm CMOS

Dieter Draxelmayr

Infineon Design Centers, Villach, Austria

Low power A/D conversion is the key to many applications, especially in portable equipment. Therefore much effort has been put into the creation of low-power architectures and low-power designs [4,5]. In the last few years, we also saw the advent of parallel ADC structures designed to achieve higher speed than possible before. However, using parallelism we also can exploit the power efficiency of simple ADC structures in order to get high performance A/D conversion at low power. In this paper, eight successive approximation ADCs have been put in parallel to get high throughput at low power. We have achieved a sampling rate of 600MHz at a power consumption of 10mW, which to our knowledge is the lowest power reported for high speed ADCs.

which are the buffer RAM. The nine smaller blocks next to them are the nine converters. In principle the converter array consists of eight converters, but there is also a nineth converter for evaluation purposes.

The chip has been fabricated in a "digital" 90nm CMOS process with 6 metal layers. The capacitors are formed with regular metallization layers, so no MIM-cap has been used. Due to the low analog requirements of the charge-redistribution architecture, only regular threshold transistors have been used. The chip operates at a supply from 1 to 1.2V. Current consumption is 8.5mA in the analog portion, as predicted.

Converter arrays are known to suffer from several mismatch effects. In principle, any kind of mismatch between the converters may generate additional error components. Figure 14.7.4 shows a spectrum of the array taken at 600MHz clock and 329MHz input frequency. We see several spurious tones which take the SINAD down to 24.6dB. The dominant tones come from the individual offset values. We can remove the offsets by sub-

Question

Designing a SAR ADC from scratch takes times!

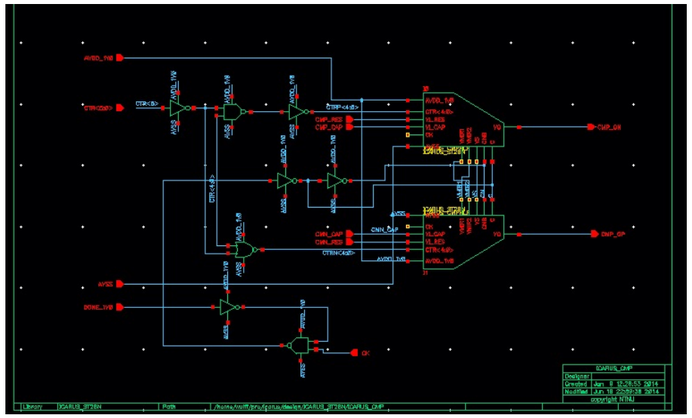
Could I design "once and for all" a process independent SAR architecture that is tolerant towards supply, temperature, process, mismatch and process technology?

Plan

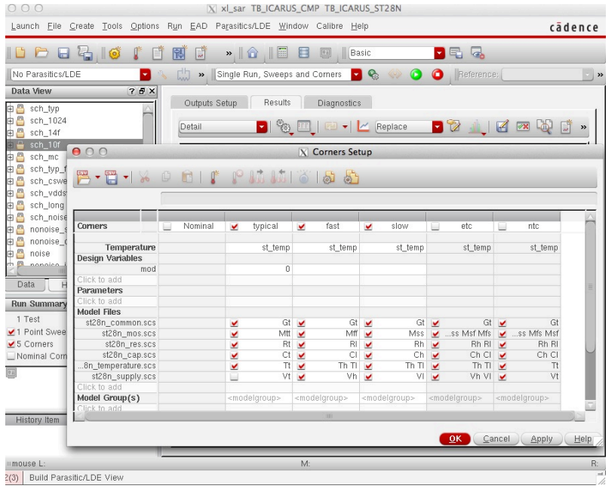
9-bit SAR ADC with 28 nm FDSOI transistors

9-bit SAR ADC with IO voltage (180 nm) FDSOI transistors

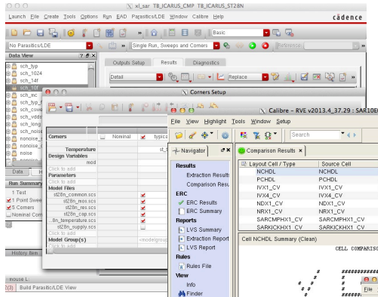
Schematic



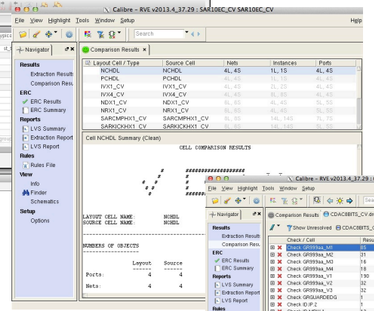
Simulation



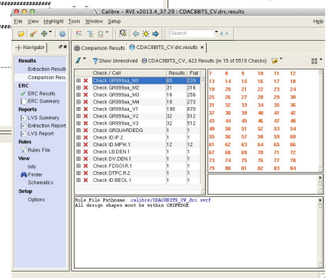
Simulation



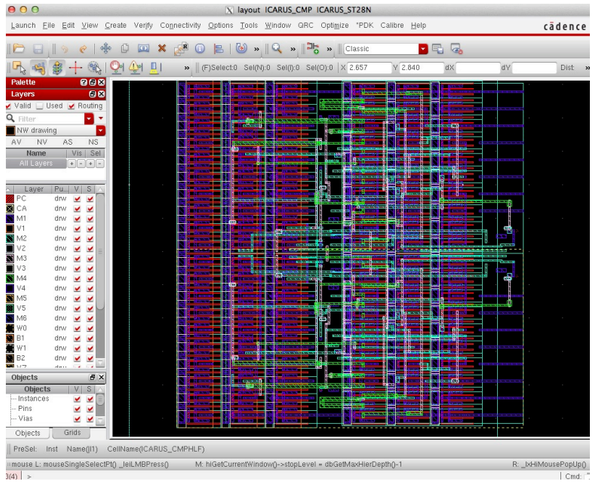
LVS



DRC



Layout

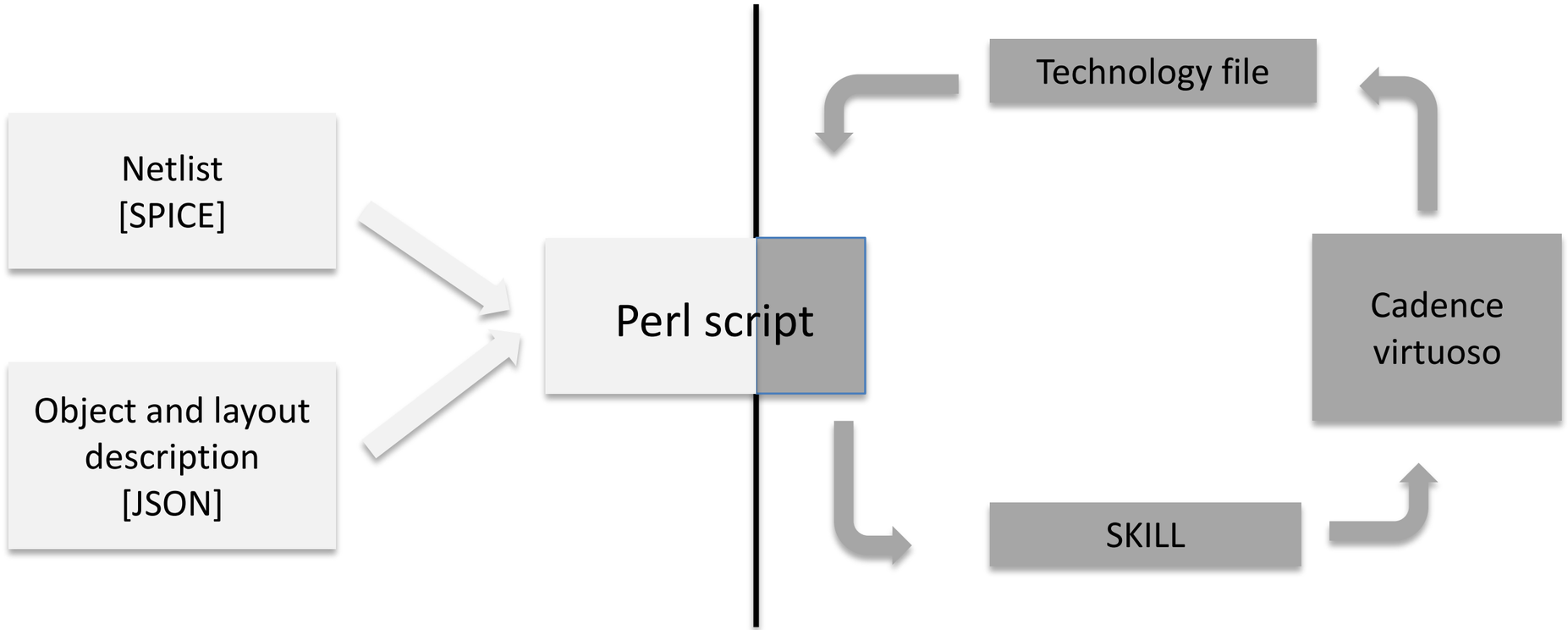


How to make multiple SAR ADCs with limited time?

Spend 50% of time for 6 months to **develop** a tool to make SAR ADCs

Spend 50% of time for 6 months to **make** the SAR ADCs

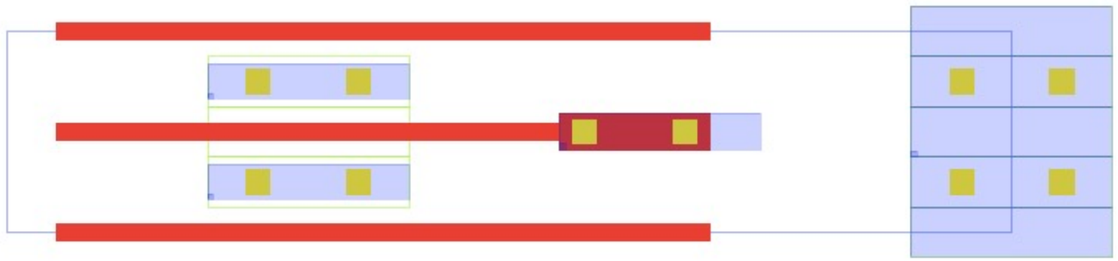
```
#include "core/layoutcell.h"

typedef QMap<QString,QList<cIcSpice::SubcktInstance*>>  SARgroup;

namespace cIcCells{

    class SAR : public cIcCore::LayoutCell
    {
        Q_OBJECT

    public:
        virtual void place();

        virtual void route();

        int getCellWidth(SARgroup groups,QString group);

        cIcCore::Instance* placeAlternateMirror(SARgroup groups,QString group,
            int i, int x ,int y, int xoffset);

        int addSarRouting(int y,int msw,int mw);

        static bool sortGraph(cIcCore::Graph* a, cIcCore:: Graph *b);

    private:
        Rect* sarn;
        Rect* sarp;
    };
}
```



16 k Perl lines. Ported to C++ for speed ⇒ ciccreator

```
{ "name" : "DMOS" ,
  "class" : "Gds::GdsPatternTransistor",
  "yoffset": -0.5,
  "widthoffset" : -1,
  "fillCoordinatesFromStrings" : [
              [   "OD",
                  "-----------------xxxx",
                  "----xCxC----------xCxC",
                  "----xxxx----------xxxx",
                  "----xCxC----------xCxC",
                  "-----------------xxxx"
              ],
              [   "PO",
                  "-mmmmmmmmmmmmm--------",
                  "----------------------",
                  "-mmmmmmmmmmcxc--------",
                  "----------------------",
                  "-mmmmmmmmmmmmm--------"
              ],
              [   "M1",
                  "-----------------xxxx",
                  "----wDww----------xxxx",
                  "-----------wGww---xBxx",
                  "----wSww----------xxxx",
                  "-----------------xxxx"
              ]
        ]
}
```

- Structure and any other property is described in JSON (JavaScript Object Notation)
- “name” is the name of the cell
- “class” defines which object to use
- All other classes in the JSON object refer to object methods (there are some special functions, but more on that later)
- Convert a text string into a layout drawing
  - c = contact
  - C = center contact on rectangle left edge
  - x = fill rectangle
  - m = use minimum length poly
  - w = use “width” from techfile
  - DGSB = add ports



```
{ "name": "IVX1_CV" ,
  "symbol" : "inv",
  "class" : "Layout::LayoutDigitalCell",
  "spice" : [
      ".subckt IVX1_CV A Y AVDD AVSS" ,
      "MN0 Y A AVSS AVSS NCHDL",
      "MP0 Y A AVDD AVSS PCHDL",
      ".ends IVX1_CV”],
  "addSchematicCoordinates" : {
      "MN0" : [ 0.25, 0, "R0"],
      "MP0" : [0.25, 0.5, "R0”]
  },
  "beforeRoute" : {
      "addDirectedRoutes" : [ ["M1","Y","MN:D-|--MP:D"], ["PO","A","MN:G-MP:G"] ]
  },
  "afterRoute"  : {
      "addPortOnRects" : [  ["A","M1", "MN0:G"] , ["Y", "M1", "MN0:D"]]
  }
}
```

What symbol to use, defaults to templates/skill/<name>.il

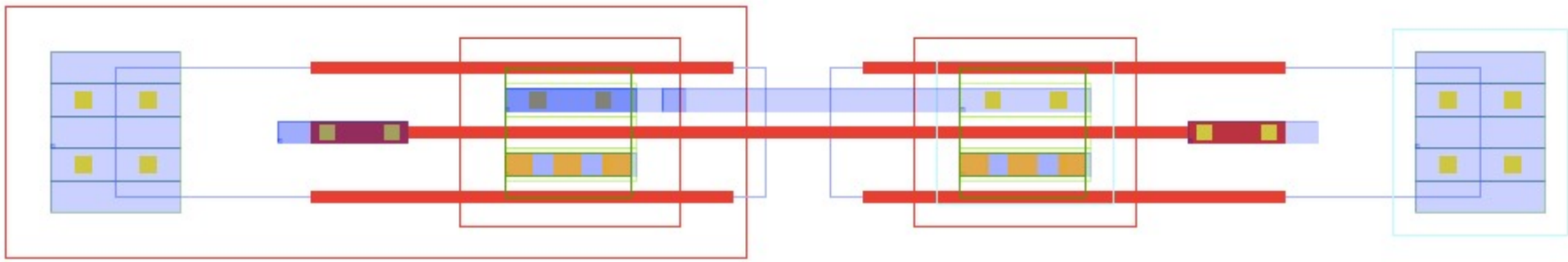
LayoutDigitalCell has extra functions for digital cells, and will add power rails.

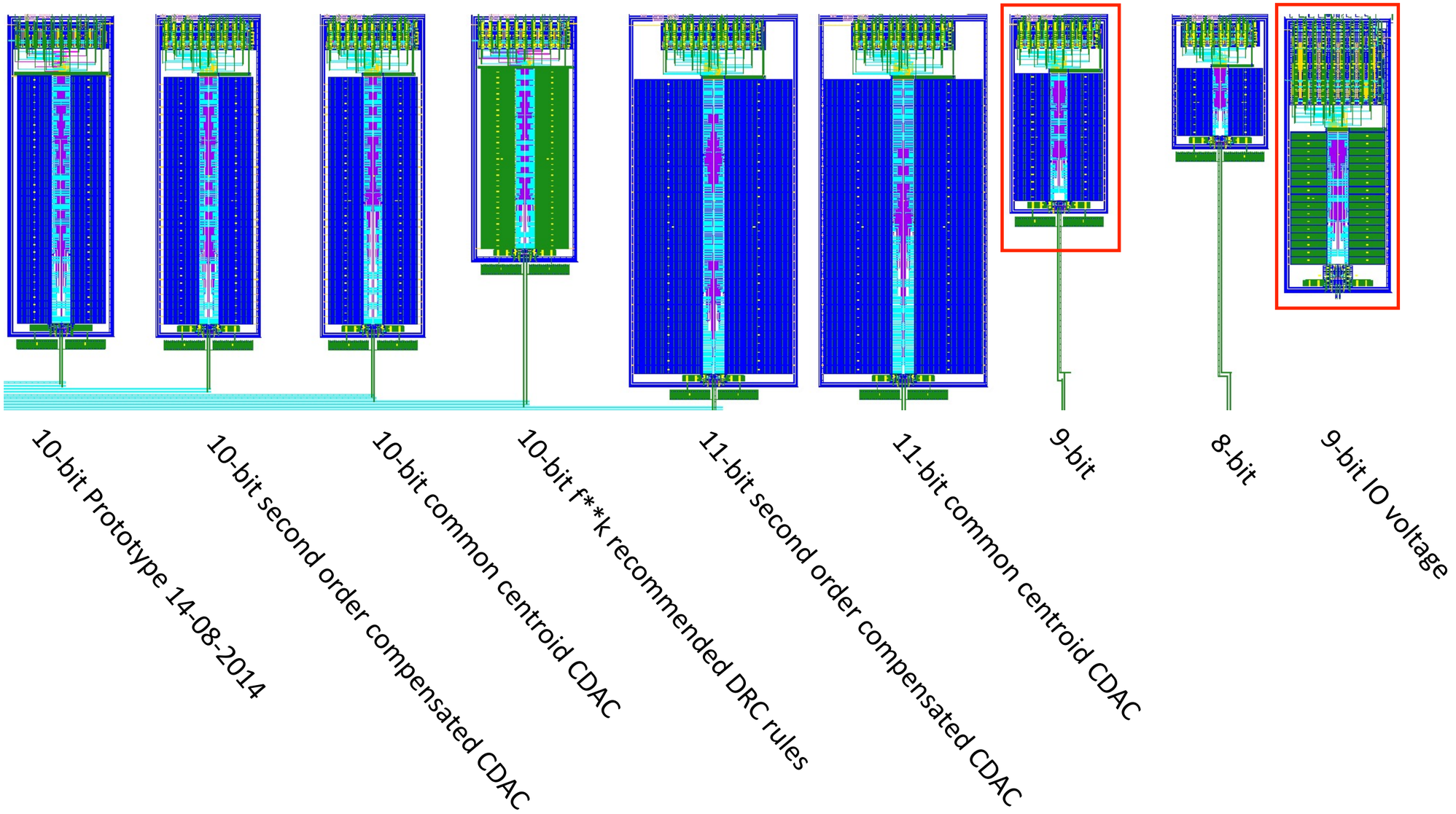
Connectivity defined by SPICE. SPICE subcircuit can be read from a separate file

Help the schematic generator to place transistors so it’s easier to read schematics

Find rectangle on device MN:D, and route in M1 to rectangle MP:D using a left, up or down, left pattern.

Add port for A on the gate of MN0




10-bit Prototype 14-08-2014
10-bit second order compensated CDAC
10-bit common centroid CDAC
10-bit f**k recommended DRC rules
11-bit second order compensated CDAC
11-bit common centroid CDAC
9-bit
8-bit
9-bit IO voltage

A Compiled 9-bit 20-MS/s 3.5-fJ/conv.step SAR ADC in 28-nm FDSOI for Bluetooth Low Energy Receivers

	Weaver [5]	Harpe [9]	Patil [10]	Liu [11]	This work	
Technology (nm)	90	90	28 FDSOI	28	28 FDSOI	
Fsample (MS/s)	21	2	No sampling	100	2	20
Core area (mm^2)	0.18	0.047	0.0032	0.0047	0.00312	
SNDR (dB)	34.61	57.79	40	64.43	46.43	48.84
SFDR (dBc)	40.81	72.33	30	75.42	61.72	63.11
ENOB (bits)	5.45	6.7 - 9.4	6.35	10.41	7.42	7.82
Supply (V)	0.7	0.7	0.65	0.9	0.47	0.69
Pwr (μW)	1110	1.64 -3.56	24	350	0.94	15.87
Compiled	Yes	No	No	No	Yes	
FoM (fJ/c.step)	838	2.8 - 6.6	3.7	2.6	2.7	3.5

14-2 (7055)

IEEE Asian Solid-State Circuits Conference
November 4 – 6, 2019
The Parisian Macao, Macao SAR, China

A 68 dB SNDR Compiled Noise-Shaping SAR ADC With On-Chip CDAC Calibration

Harald Garvik*, Carsten Wulff† and Trond Ytterdal*
Email: harald.garvik@ntnu.no
*Dept. of Electronic Systems, Norwegian University of Science and Technology (NTNU), Trondheim, Norway
†Nordic Semiconductor, Trondheim, Norway

***Abstract*—This paper[1] presents a noise-shaping SAR ADC with an on-chip, foreground capacitive DAC (CDAC) calibration system. At start-up, the ADC uses the LSBs in the CDAC to measure and digitize the errors of the MSBs. A synthesized digital module accumulates the noise-shaped measurements, computes calibration coefficients, and corrects ADC codes at run-time. The loop filter implements two optimal zeros and two poles, and achieves 27.8 dB in-band attenuation at an oversampling rate (OSR) of 4. The prototype is implemented in 28 nm FDSOI, and achieves 68.2 dB SNDR at 5 MHz bandwidth, while consuming 108.7 µW from a 0.8 V supply. The Walden FOM is 5.2 fJ/conv.-step. The layout of the ADC is compiled from a netlist, a rule file, and an object definition file.**

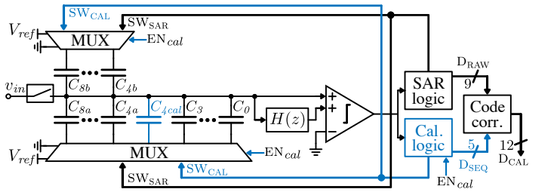
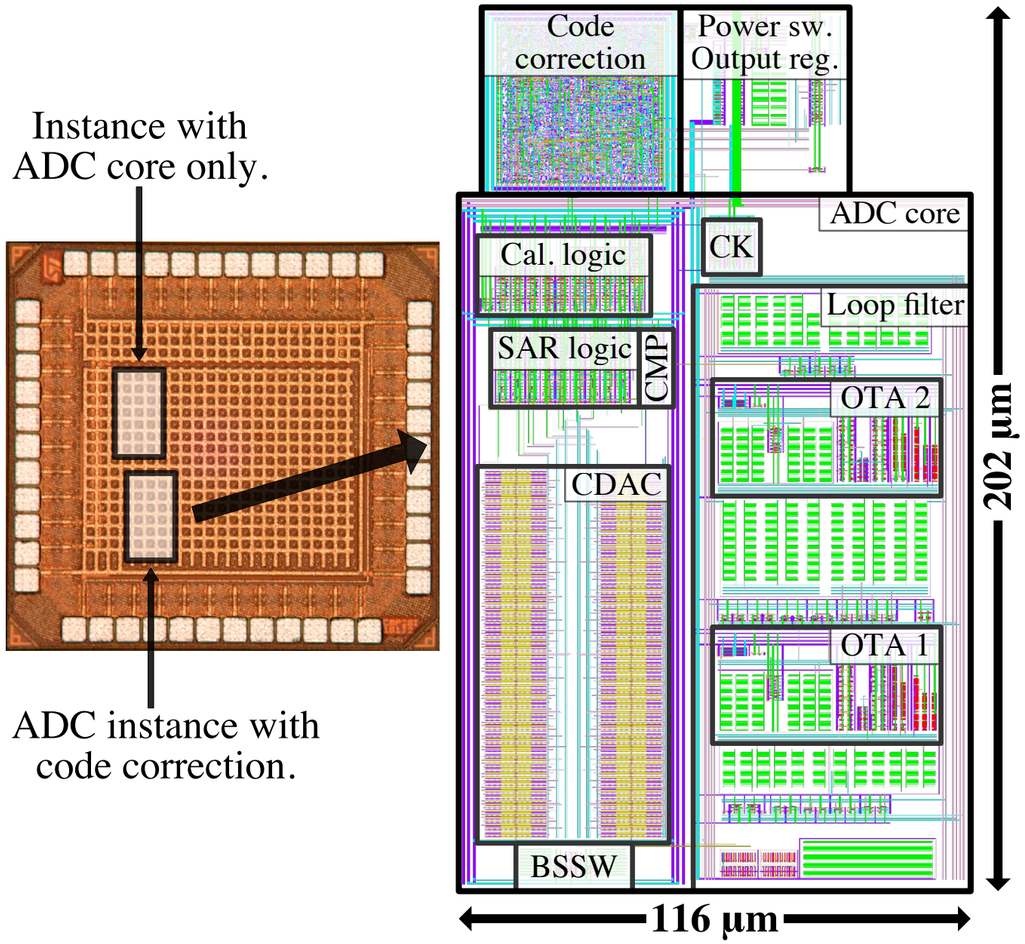
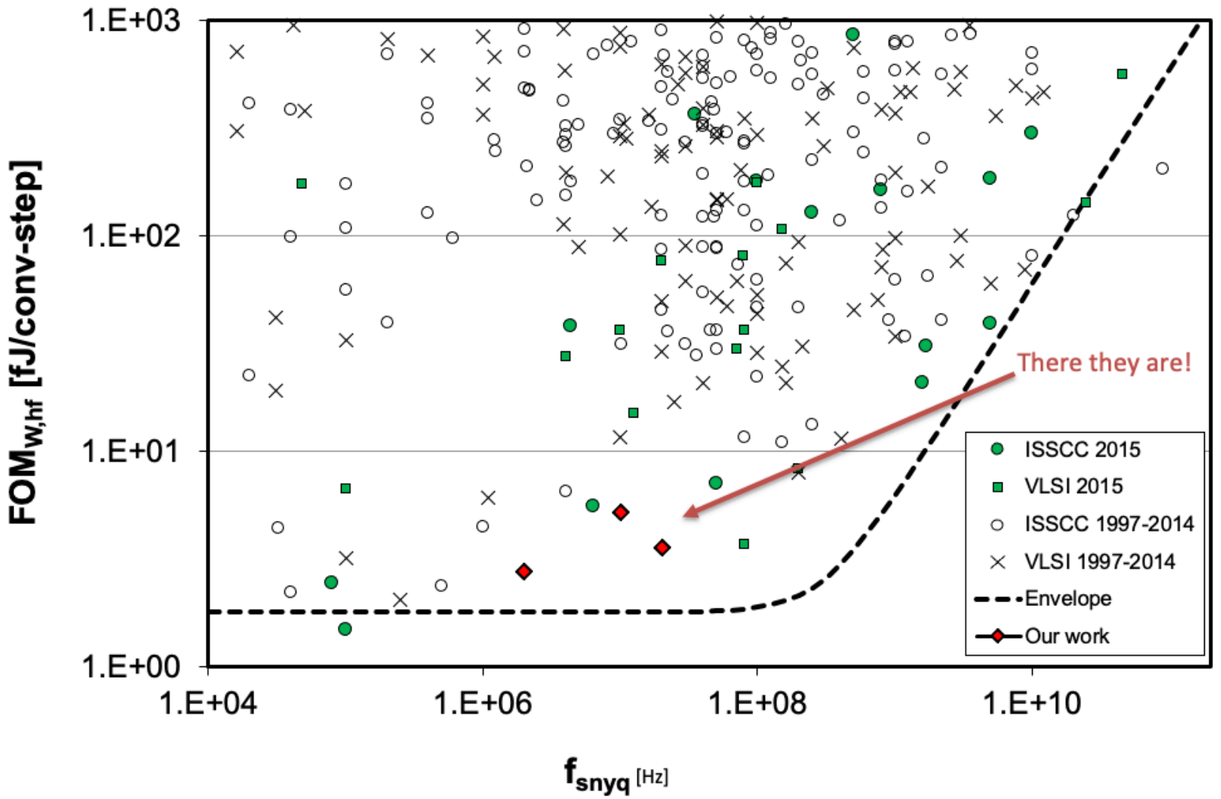


Fig. 1. Proposed noise-shaping SAR architecture. Blue blocks and paths are only active in calibration mode.



Result

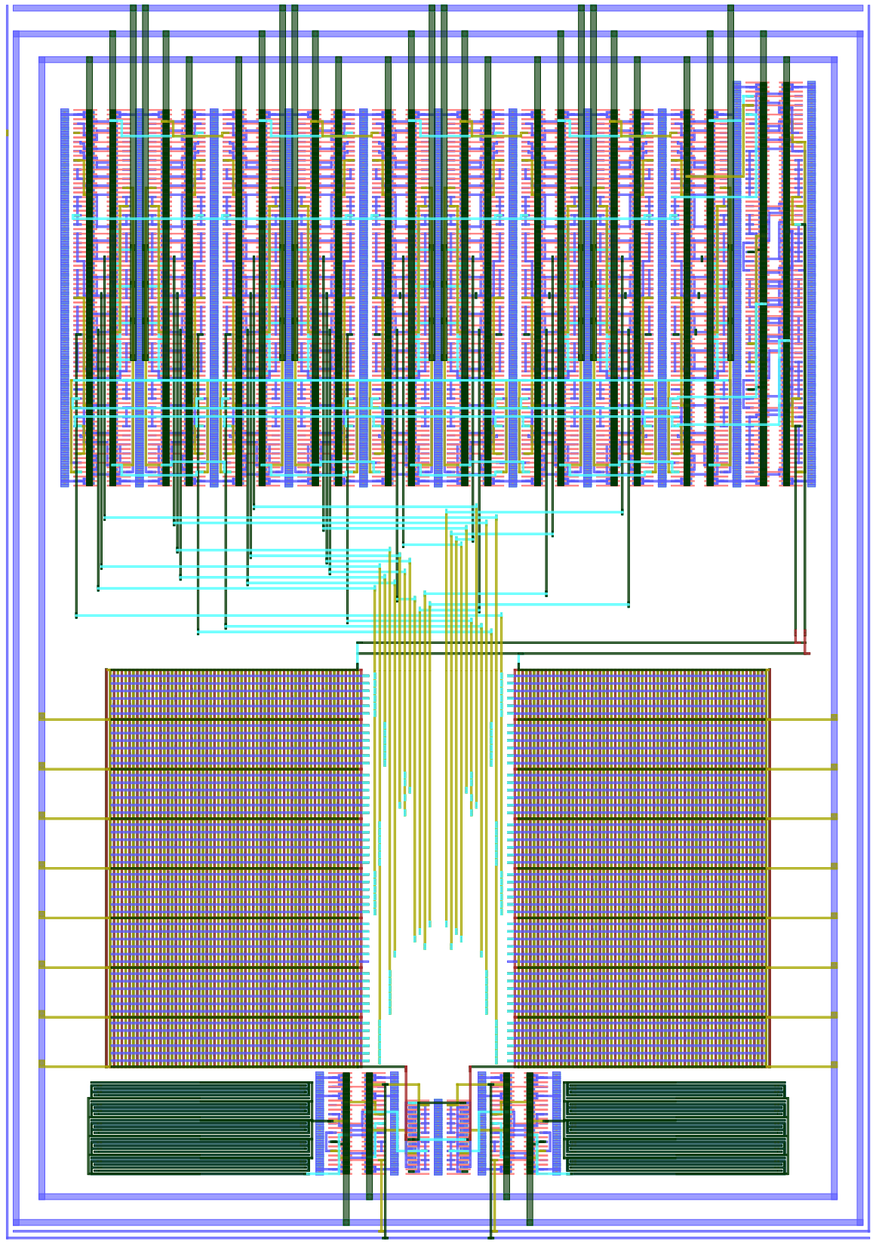

1.E+03
1.E+02
1.E+01
1.E+00
1.E+04
1.E+06
1.E+08
1.E+10
FOM$_{W,hf}$ [fJ/conv-step]
f_{snyq} [Hz]
There they are!
ISSCC 2015
VLSI 2015
ISSCC 1997-2014
VLSI 1997-2014
Envelope
Our work

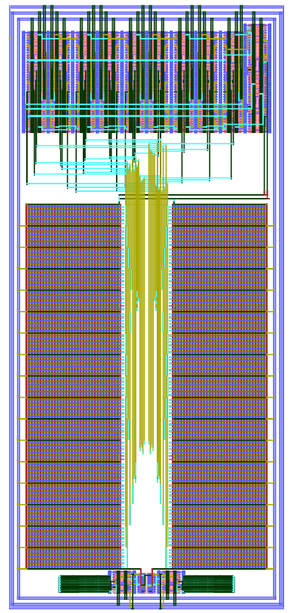
SUN_SAR9B_GF130N

```
├── SAR9B_CV.json
├── SAR9B_CV.spi
├── capacitor.json
├── dmos_gf130nm_core.json
└── gf_130bcdlite.tech
```

9-bit is not the energy optimum for GF130N. The power consumption of the digital is too high. The energy optimum is more like 13-bit, however, the CDAC matching limits performance to about 10/11-bit

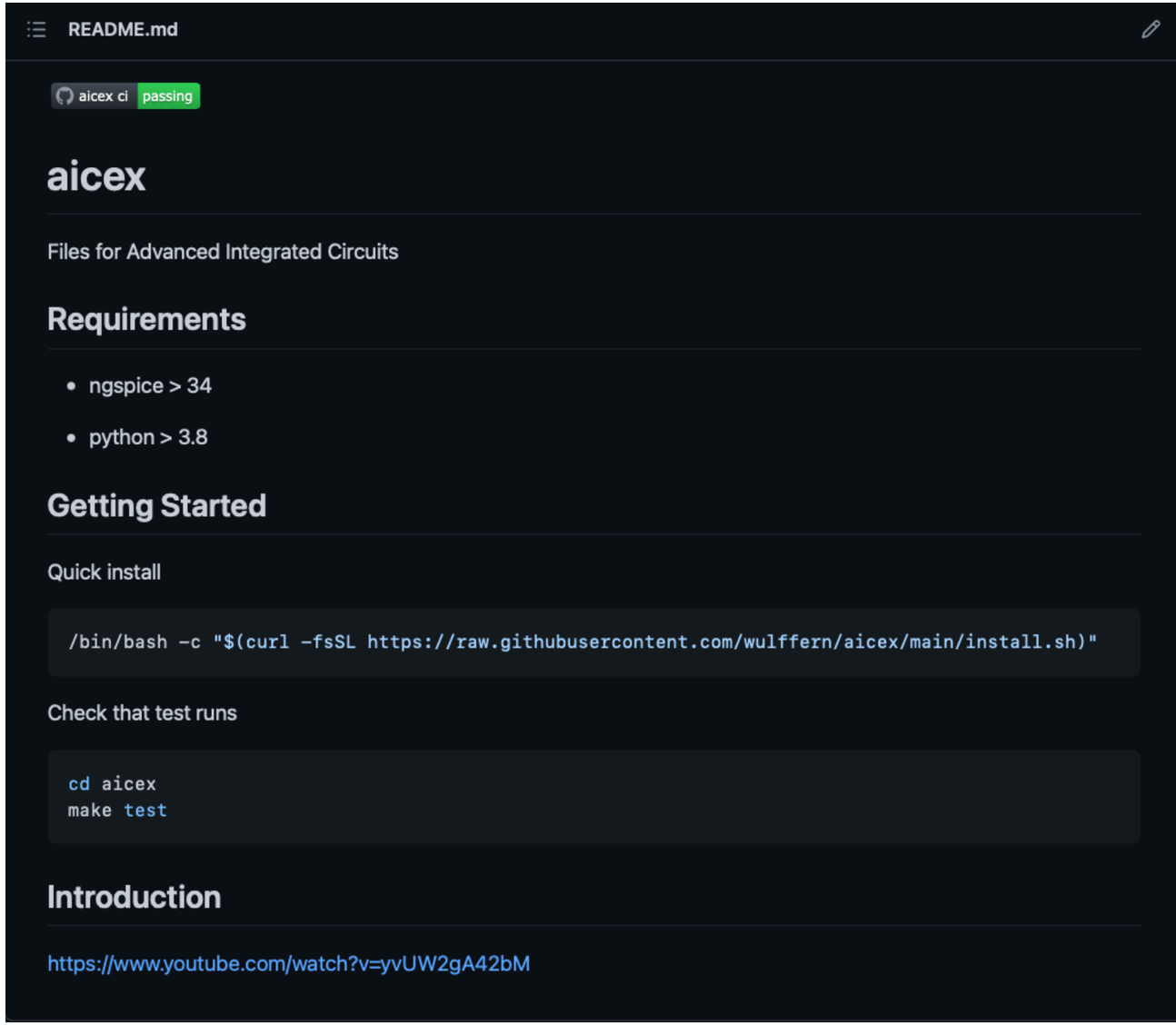
$$\log 2[(W_{130n}L_{130n})/(W_{28n}L_{28n})] = \log 2[(0.95 \times 0.14)/(0.258 \times 0.03)] = \text{4-bit extra}$$



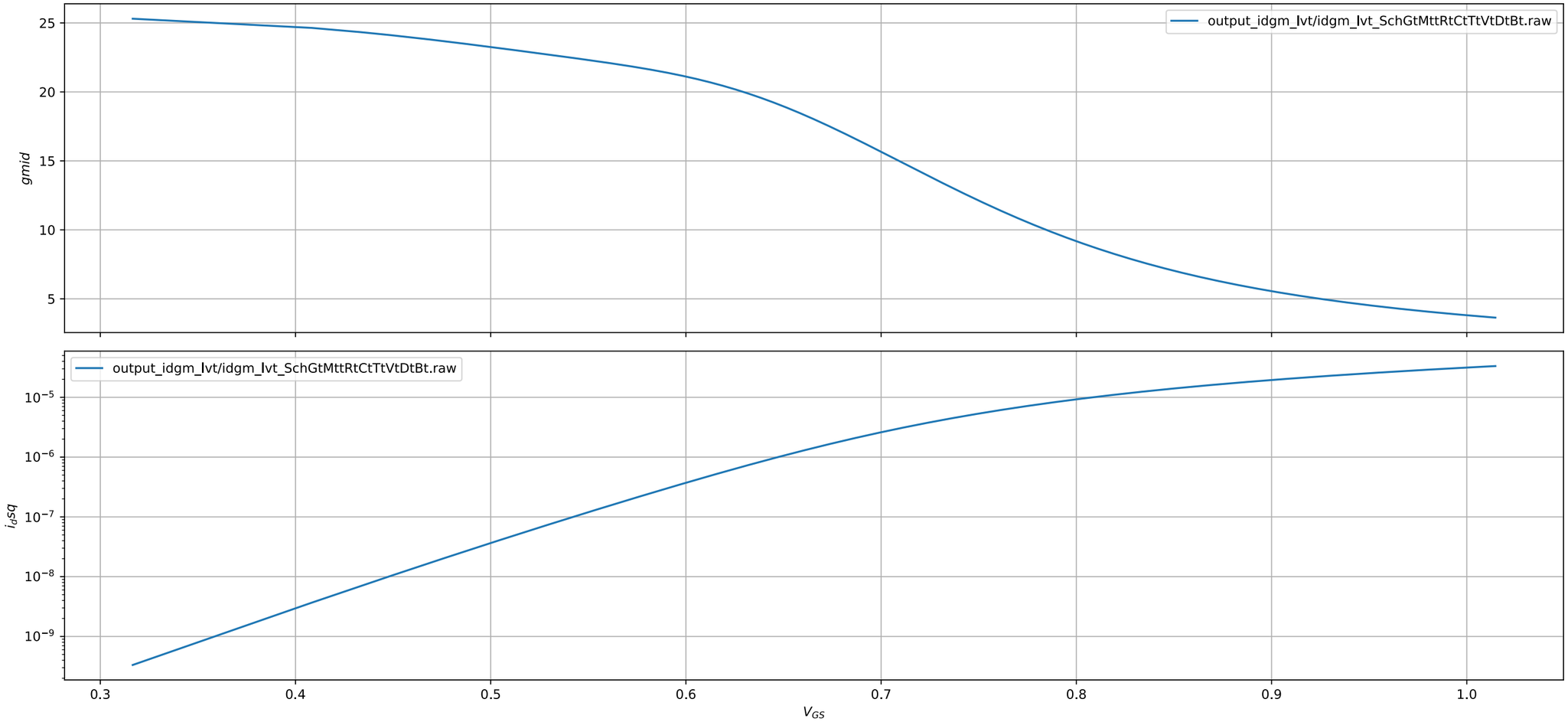


Recommendations

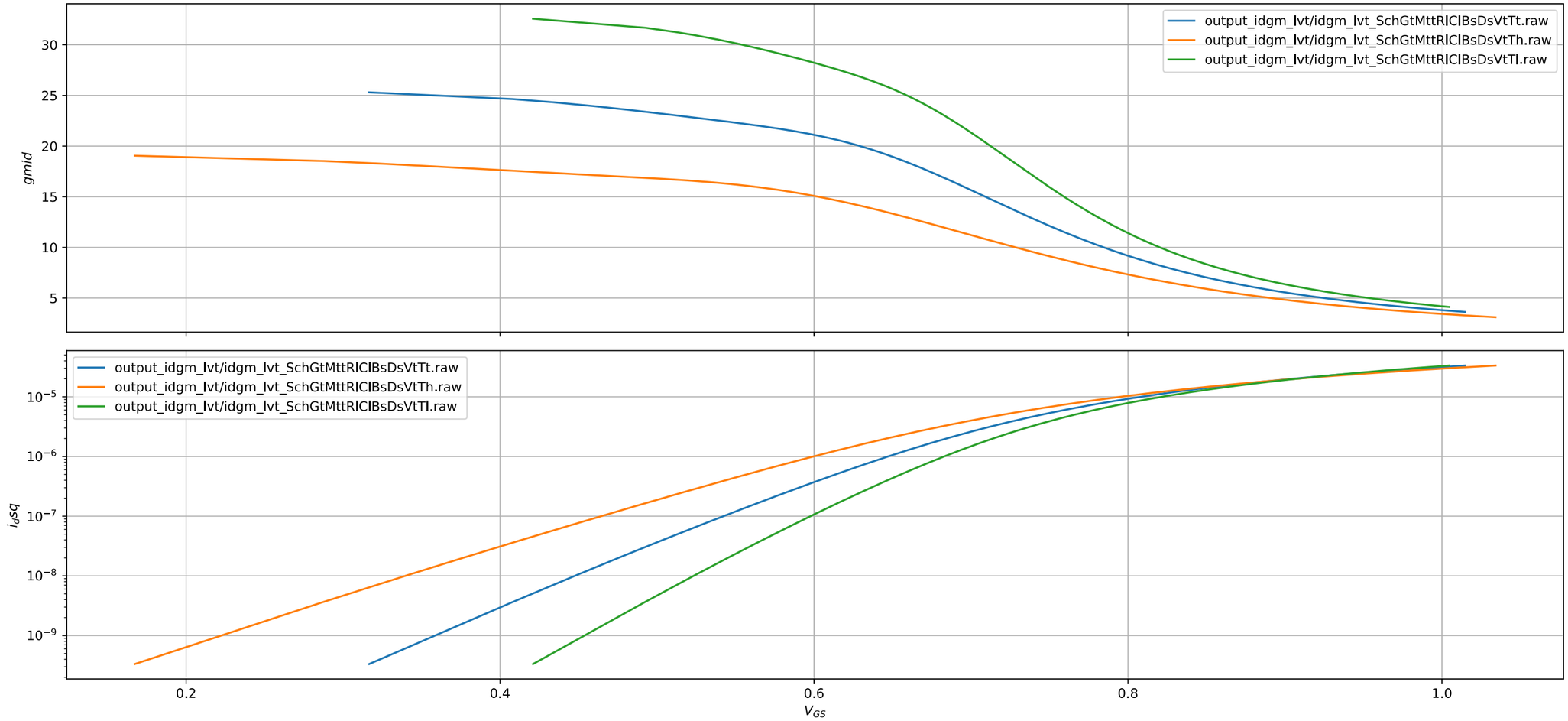
https://github.com/wulffern/aicex



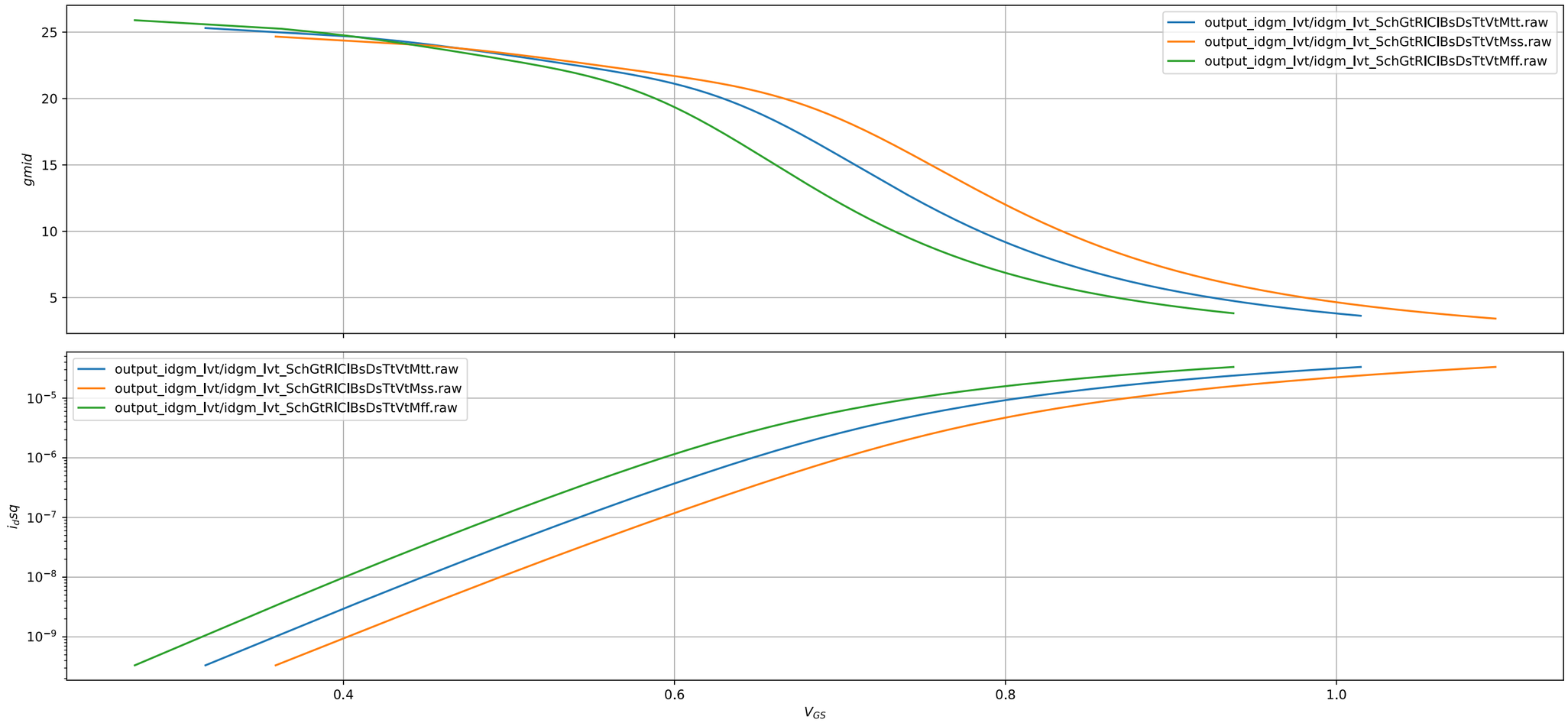
Typical


output_idgm_Ivt/idgm_Ivt_SchGtMttRtCtTtVtDtBt.raw
gmid
25
20
15
10
5
output_idgm_Ivt/idgm_Ivt_SchGtMttRtCtTtVtDtBt.raw
i_dsq
10^{-5}
10^{-6}
10^{-7}
10^{-8}
10^{-9}
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1.0
V_{GS}

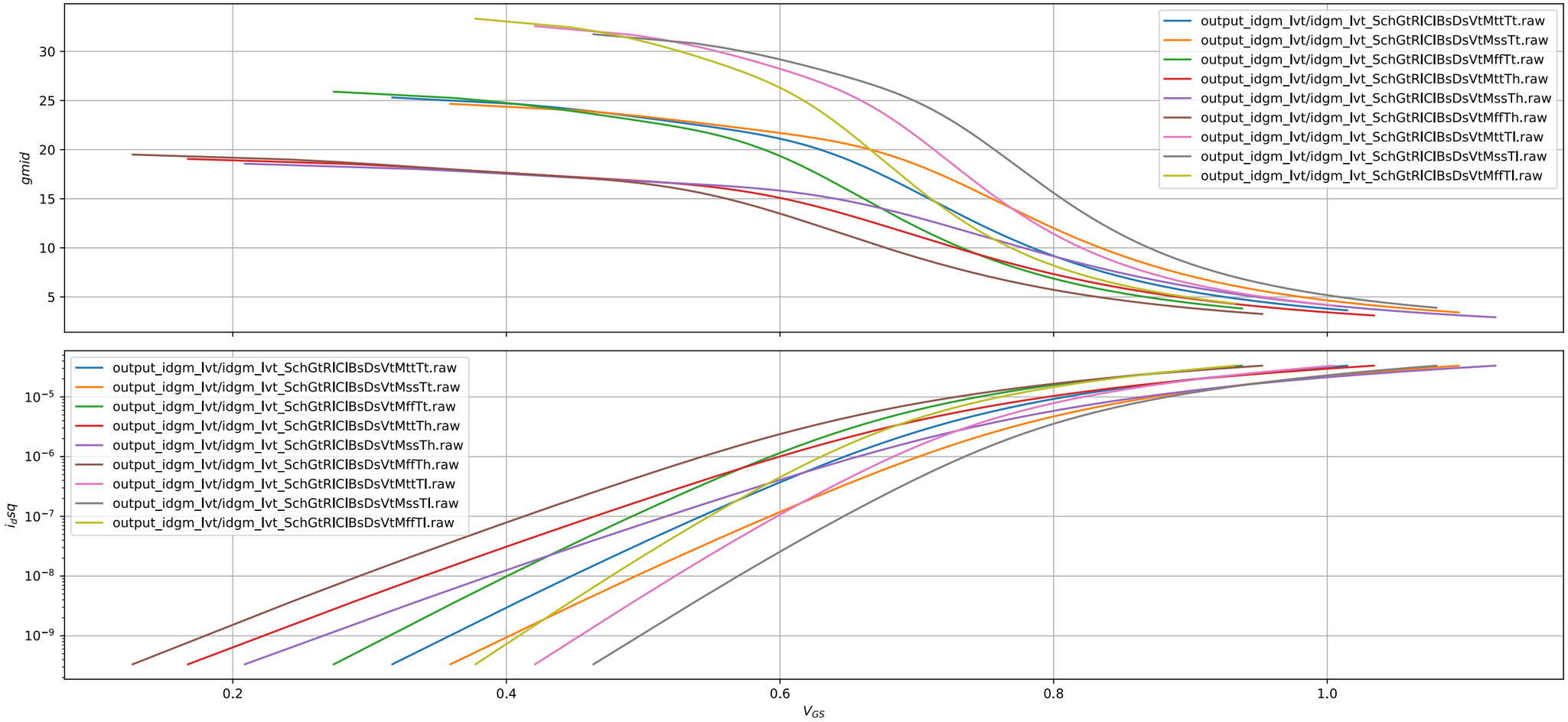
Temperature



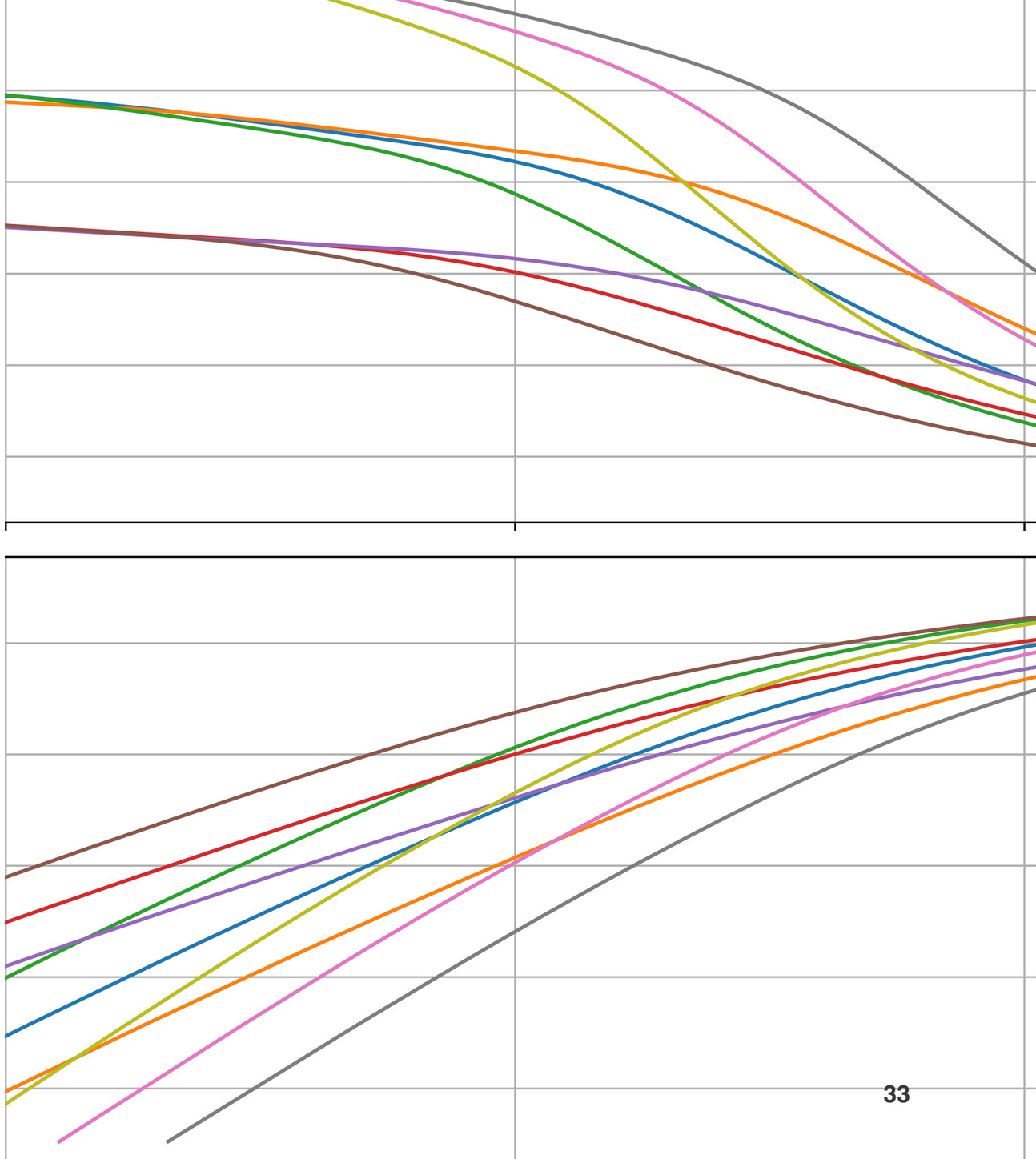
Process


output_idgm_lvt/idgm_lvt_SchGtRlClBsDsTtVtMtt.raw
output_idgm_lvt/idgm_lvt_SchGtRlClBsDsTtVtMss.raw
output_idgm_lvt/idgm_lvt_SchGtRlClBsDsTtVtMff.raw
gmid
25
20
15
10
5
output_idgm_lvt/idgm_lvt_SchGtRlClBsDsTtVtMtt.raw
output_idgm_lvt/idgm_lvt_SchGtRlClBsDsTtVtMss.raw
output_idgm_lvt/idgm_lvt_SchGtRlClBsDsTtVtMff.raw
i_dsq
10^{-5}
10^{-6}
10^{-7}
10^{-8}
10^{-9}
0.4
0.6
0.8
1.0
V_{GS}

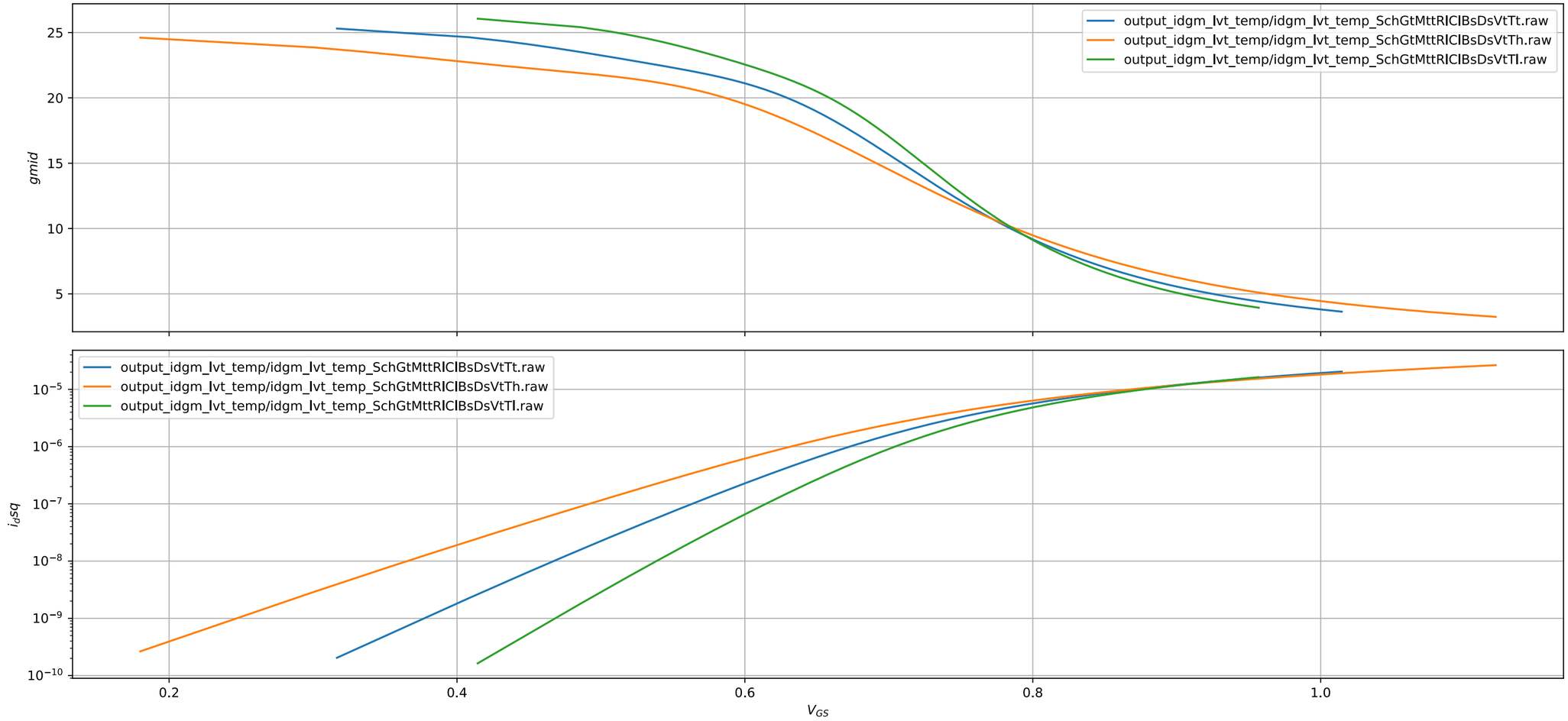
Process and temperature



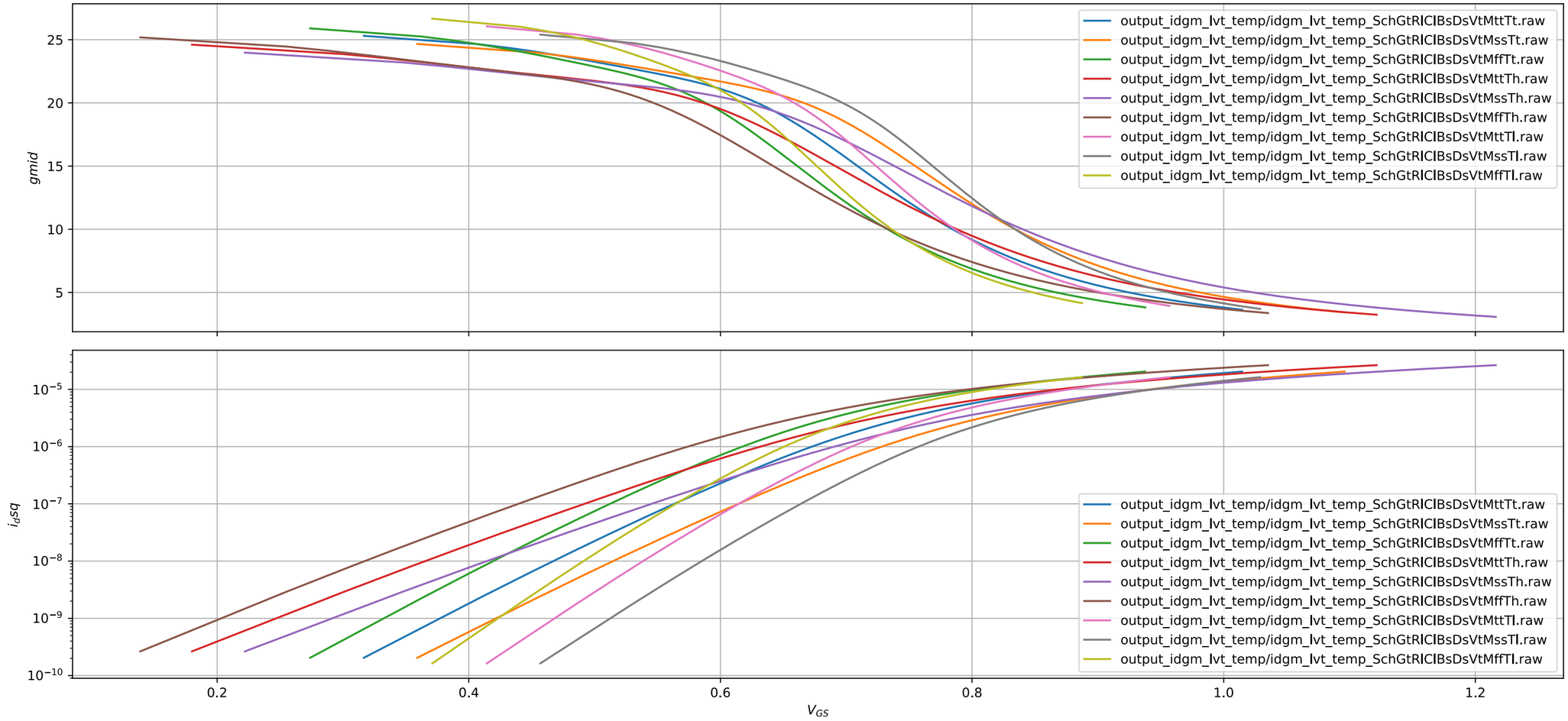
What gm/I_d would you pick?

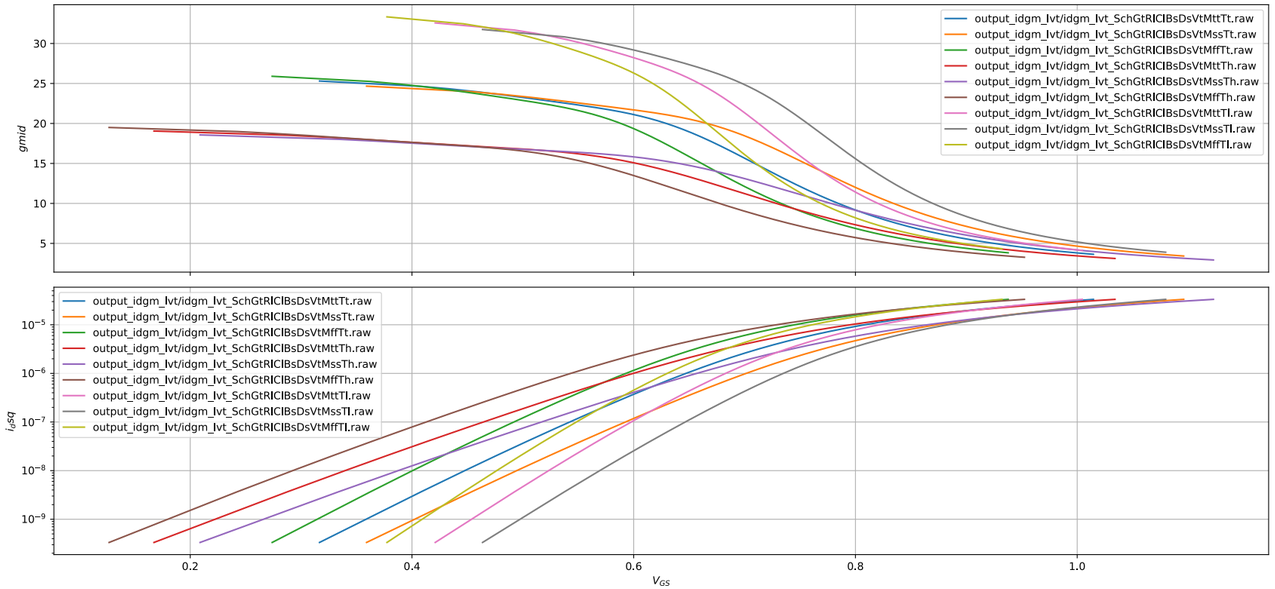


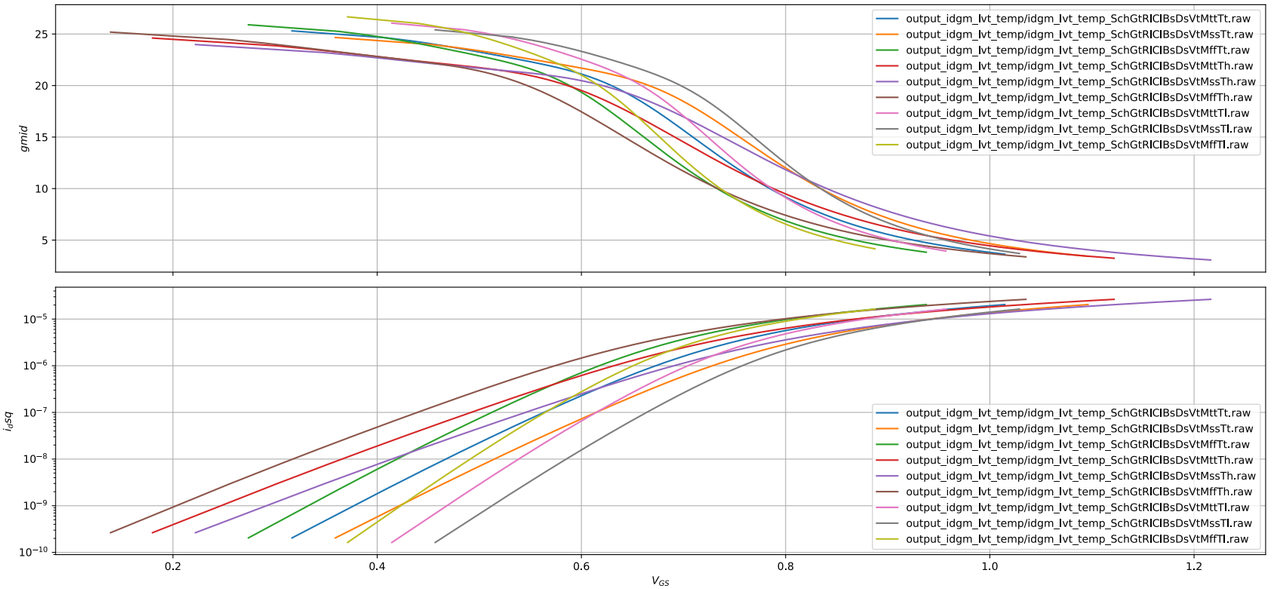
Temperature and BINN 0 N1_0 i=i(VREF) tc1={0.3/100}

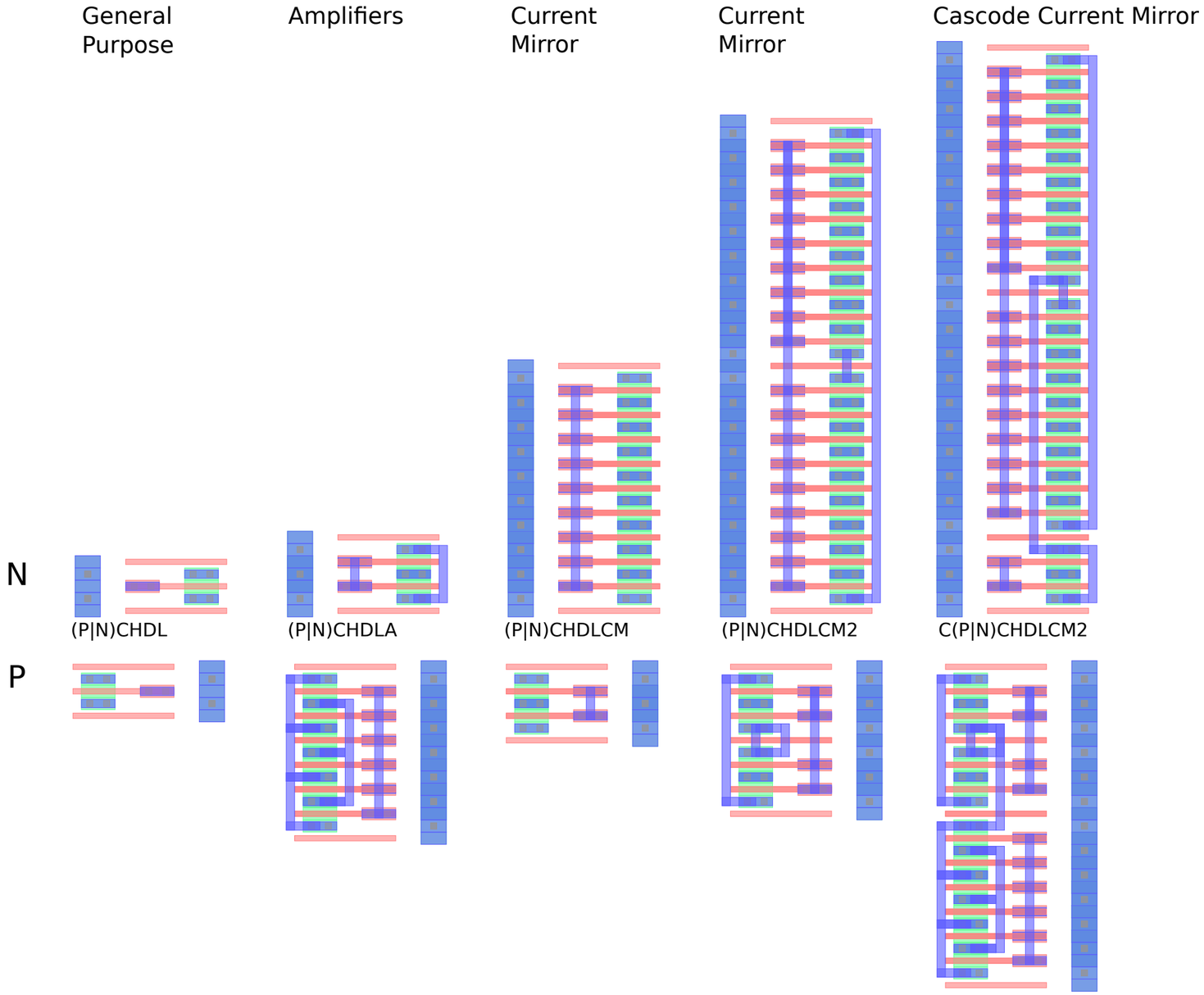


Process, Temperature and `BINN 0 N1_0 i=i(VREF) tc1={0.3/100}`




output_idgm_lvt/idgm_lvt_SchGtRIClBsDsVtMttTt.raw
output_idgm_lvt/idgm_lvt_SchGtRIClBsDsVtMssTt.raw
output_idgm_lvt/idgm_lvt_SchGtRIClBsDsVtMffTt.raw
output_idgm_lvt/idgm_lvt_SchGtRIClBsDsVtMttTh.raw
output_idgm_lvt/idgm_lvt_SchGtRIClBsDsVtMssTh.raw
output_idgm_lvt/idgm_lvt_SchGtRIClBsDsVtMffTh.raw
output_idgm_lvt/idgm_lvt_SchGtRIClBsDsVtMttTl.raw
output_idgm_lvt/idgm_lvt_SchGtRIClBsDsVtMssTl.raw
output_idgm_lvt/idgm_lvt_SchGtRIClBsDsVtMffTl.raw
gmid
i_dsq
V_{GS}


output_idgm_lvt_temp/idgm_lvt_temp_SchGtRIClBsDsVtMttTt.raw
output_idgm_lvt_temp/idgm_lvt_temp_SchGtRIClBsDsVtMssTt.raw
output_idgm_lvt_temp/idgm_lvt_temp_SchGtRIClBsDsVtMffTt.raw
output_idgm_lvt_temp/idgm_lvt_temp_SchGtRIClBsDsVtMttTh.raw
output_idgm_lvt_temp/idgm_lvt_temp_SchGtRIClBsDsVtMssTh.raw
output_idgm_lvt_temp/idgm_lvt_temp_SchGtRIClBsDsVtMffTh.raw
output_idgm_lvt_temp/idgm_lvt_temp_SchGtRIClBsDsVtMttTl.raw
output_idgm_lvt_temp/idgm_lvt_temp_SchGtRIClBsDsVtMssTl.raw
output_idgm_lvt_temp/idgm_lvt_temp_SchGtRIClBsDsVtMffTl.raw
gmid
i_dsq
V_{GS}


General Purpose
Amplifiers
Current Mirror
Current Mirror
Cascode Current Mirror
N
P
(P|N)CHDL
(P|N)CHDLA
(P|N)CHDLCM
(P|N)CHDLCM2
C(P|N)CHDLCM2

SUN_TR_GF130N

```
├── SUN_TR.json
├── components.json
├── components.spi
├── dig.json
├── dig.spi
├── dmos_gf130nm_core.json
├── gf_130bcdlite.tech
├── tr.json
└── tr.spi
```

GMID = 15

Param	Vgs (Min)	Vgs (Max)	Id (Min)	Id (Max)	Gmro (Min)	Gmro (Max)
NCHDL	455.30m	734.70m	4.14u	7.57u	20.24	23.81
NCHDLA	446.20m	723.90m	8.68u	15.61u	21.06	24.79
NCHDLCM	465.40m	765.60m	422.40n	999n	163	272.90
NCHDLCM2	465.40m	765.60m	844.70n	2u	163	272.90
CNCHDLCM2	465.60m	765.90m	772.60n	1.83u	163	272.90

Param	Vgs (Min)	Vgs (Max)	Id (Min)	Id (Max)	Gmro (Min)	Gmro (Max)
PCHDL	387.10m	739.60m	1.01u	3.53u	27.26	28.64
PCHDLA	385.70m	738.20m	5.71u	19.83u	27.32	28.69
PCHDLCM	374.30m	742.90m	383.60n	1.53u	50.22	54.14
PCHDLCM2	374.30m	742.90m	767.10n	3.06u	50.22	54.14
CPCHDLCM2	372.40m	747.80m	620.10n	2.47u	50.22	54.14

GMID = 10

Param	Vgs (Min)	Vgs (Max)	Id (Min)	Id (Max)	Gmro (Min)	Gmro (Max)
NCHDL	555.90m	848.30m	15.52u	29.15u	19.37	23.10
NCHDLA	547.20m	839m	33.38u	61.50u	20.06	24.21
NCHDLCM	582.50m	872.80m	1.75u	3.97u	134.30	196.90
NCHDLCM2	582.50m	872.80m	3.50u	7.93u	134.30	196.90
CNCHDLCM2	582.80m	873.20m	3.20u	7.27u	134.30	196.90

Param	Vgs (Min)	Vgs (Max)	Id (Min)	Id (Max)	Gmro (Min)	Gmro (Max)
PCHDL	503.40m	827.30m	4.90u	9.83u	24.16	26.74
PCHDLA	501.80m	825.60m	27.53u	55.21u	24.24	26.81
PCHDLCM	502.60m	837.60m	2.10u	4.55u	44.65	47.78
PCHDLCM2	502.60m	837.60m	4.20u	9.10u	44.65	47.78
CPCHDLCM2	500.40m	843m	3.28u	7.26u	44.65	47.78

Use a few transistors that you know well know all **important** details

Use M factor to scale current and gm bus notation

M1<9:0> on instance name in cadence

Pick the right bias current for your circuit

constant, constant gm, proportional to temperature

Thanks!