Teaching Mixed-Mode Full-Custom VLSI Design with gaf, SpiceOpus and Glade

S. Sutula[1], F. Vila[1], J. Pallarès[1], K. Sabine[2], L. Terés[1,3] and F. Serra-Graells[1,3]

jofre.pallares@imb-cnm.csic.es

[1]Instituto de Microelectrónica de Barcelona, IMB-CNM(CSIC)

[2]Peardrop Design Systems Ltd

[3]Dept. of Microelectronics and Electronic Systems
Universitat Autònoma de Barcelona

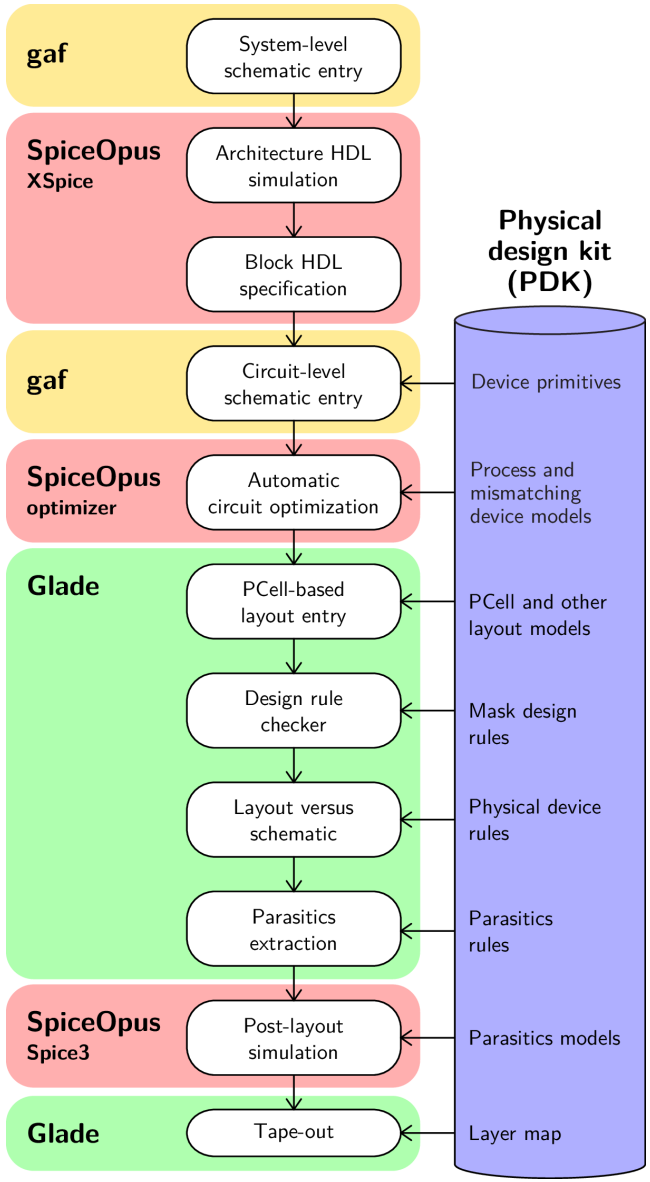
May 2014

Motivation and Objectives

- **Problems** teaching VLSI deign at lab:
  - Professional EDA tools requirements (licenses, hardware, administration)
  - Technology confidentiality
  - Limited lab session time

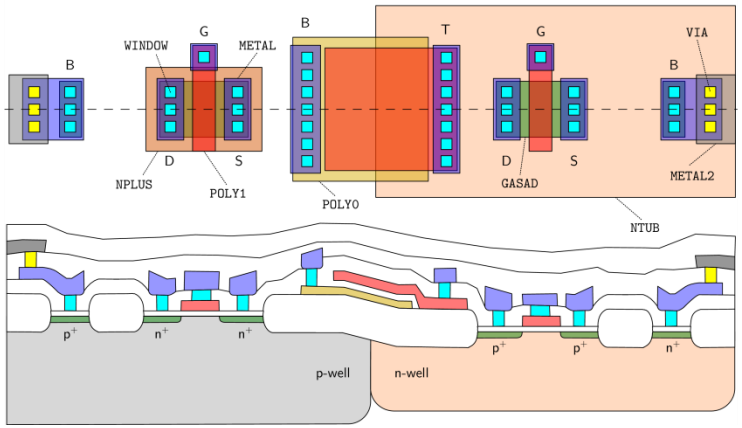
- **EDA** environment proposal for mixed-mode full-custom ASIC design:
  - **gaf** (gschem and friends) for schematic edition and netlisting
  - **SpiceOpus** (SPICE with integrated optimization utilities) for mixed-mode HDL-electrical simulation
  - **Glade** (GDS, LEF and DEF editor) for layout design and verification

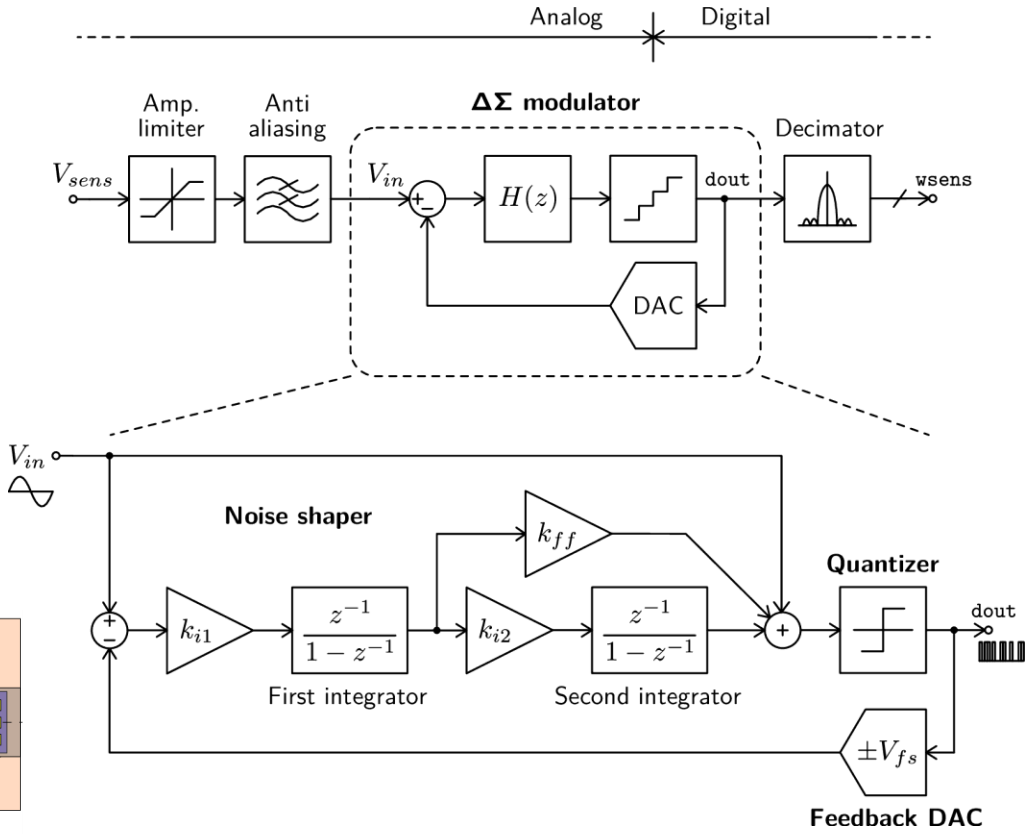
- ▲ **Freeware**, available for
- ▼ **PDK** development



Case Study

- 14-bit 8kHz 2V_{dp} **A/D ΔΣM**
  - Mixed signal domains
  - HDL modeling required due to oversampling
- 2P2M 2.5µm CMOS (**CNM25**) target technology
  - Reduced DRC rule set
  - Simple device modeling
  - Easy PDK development





Schematic Entry

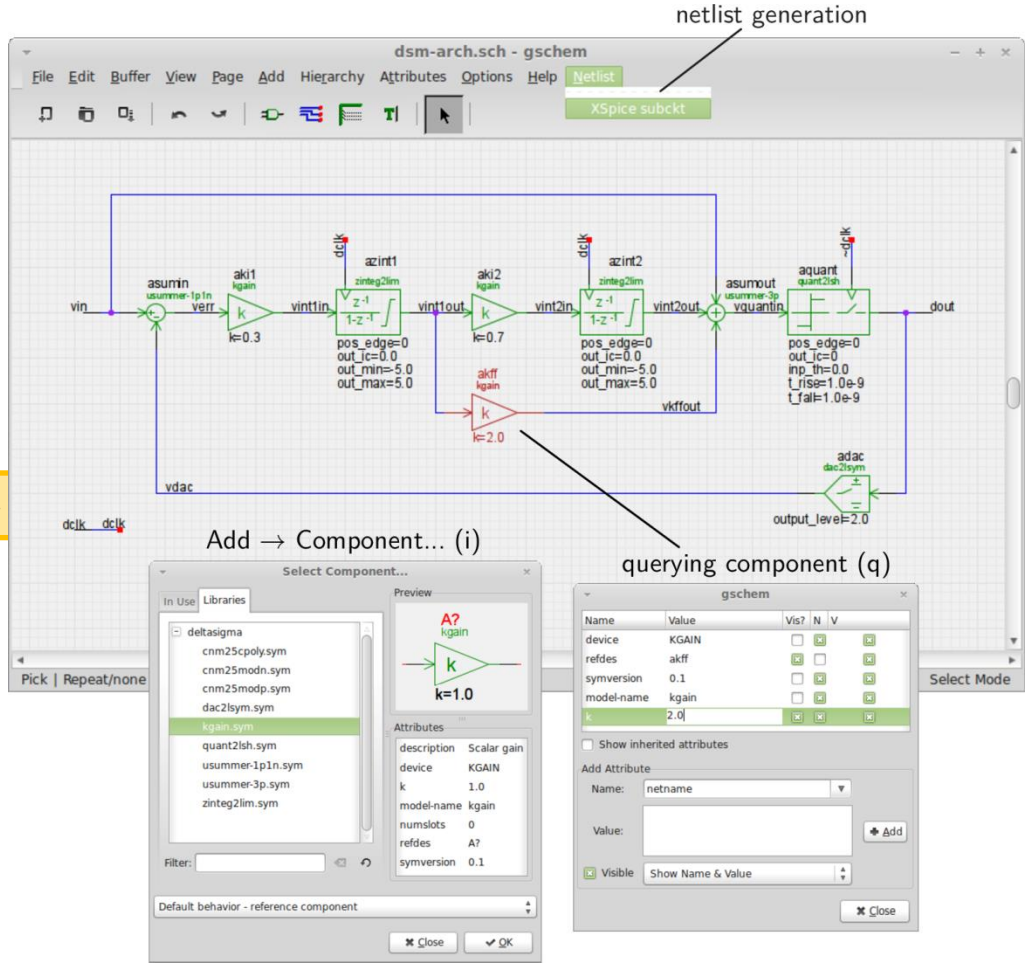
- Fully customizable **symbols**
- Programmable **netlisting rules** for standard SPICE devices and custom XSpice models

e.g. ΔΣM architecture

gnetlist

dsm-arch.sub

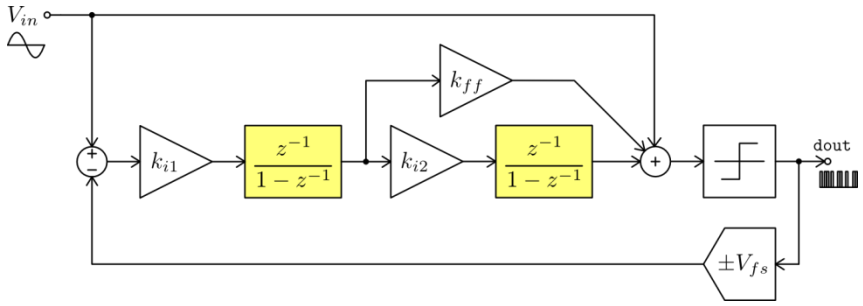
```
.subckt dsm_arch vin dclk dout
asumin [%v(vin) %v(vdac)] %v(verr) msumin
.model msumin usummer(sign=[1.0 -1.0])
aki1 %v(verr) %v(vint1in) mki1
.model mki1 kgain(k=0.3)
azint1 %v(vint1in) %d(dclk) %v(vint1out) mzint1
.model mzint1 zinteg2lim(pos_edge=0 out_ic=0.0
 + out_min=-5.0 out_max=5.0)
aki2 %v(vint1out) %v(vint2in) mki2
.model mki2 kgain(k=0.7)
azint2 %v(vint2in) %d(dclk) %v(vint2out) mzint2
.model mzint2 zinteg2lim(pos_edge=0 out_ic=0.0
 + out_min=-5.0 out_max=5.0)
akff %v(vint1out) %v(vkffout) mkff
.model mkff kgain(k=2.0)
asumout [%v(vint2out) %v(vkffout) %v(vin)]
 + %v(vquantin) msumout
.model msumout usummer(sign=[1.0 1.0 1.0])
aquant %v(vquantin) %d(~dclk) %d(dout) mquant
.model mquant quant2lsh(inp_th=0.0 out_ic=0 pos_edge=0
 + t_rise=1e-9 t_fall=1e-9)
adac %d(dout) %v(vdac) mdac
.model mdac dac2lsym(out_level=2.0)
.ends
```

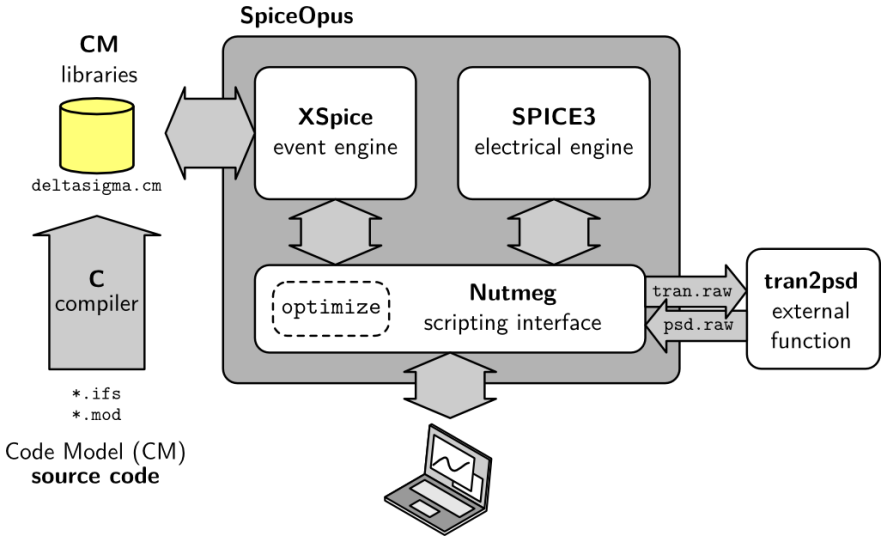


- Library browsing, net and pin labeling, **hierarchical** navigation, instance annotation and automatic **rewiring**...

Architecture HDL Simulation

- Students can code in C their own mixed-mode XSpice **HDL models**





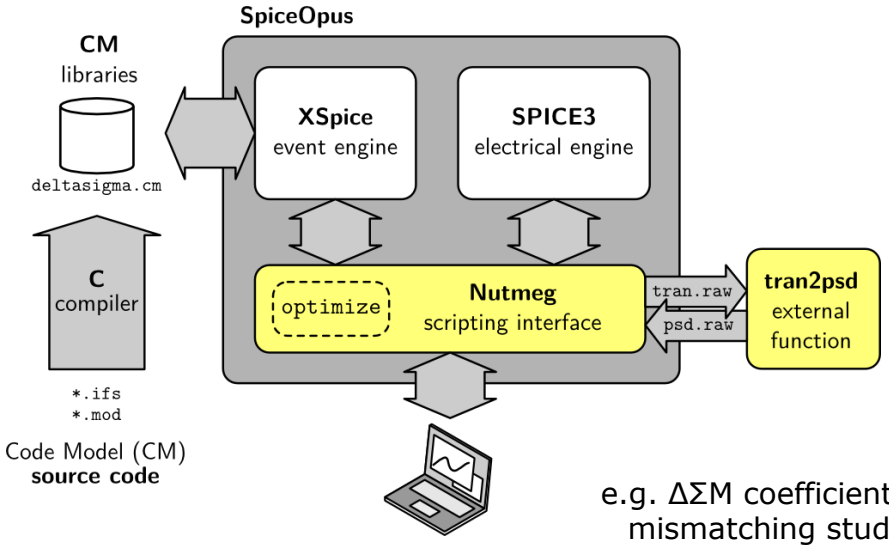
zinteg2lim.mod

```
void cm_zinteg2lim(ARGS) {
 inp = INPUT(inp);            /* Retriving input values */
 clk = INPUT_STATE(clk);
 pos_edge = PARAM(pos_edge);  /* Retrieving parameters */
 out_max = PARAM(out_ax);
 ...
 switch (ANALYSIS) {
  case TRANSIENT:
   if ((*clk_mem==ONE)&&(clk==ZERO)) {/* Neg. clk edge */
    if (pos_edge==FALSE)
     action = SAMPLING_INTEGRATION;
   } else {
    if ((*clk_mem==ZERO)&&(clk==ONE)) {/* Pos. clk edge*/
     if (pos_edge==TRUE)
      action = SAMPLING_INTEGRATION;
    } else {                          /* No clock edge */
     action = HOLDING;
   }
 ...
   switch (action) {
    case SAMPLING_INTEGRATION:
     *inp_mem = inp;
     out = *out_mem+*inp_mem;
     if (out<out_min) { out = out_min; }    /* Limiter */
     if (out>out_max) { out = out_max; }
     *out_mem = out;
     break;
    case HOLDING:
     out = *out_mem;
}}}
```

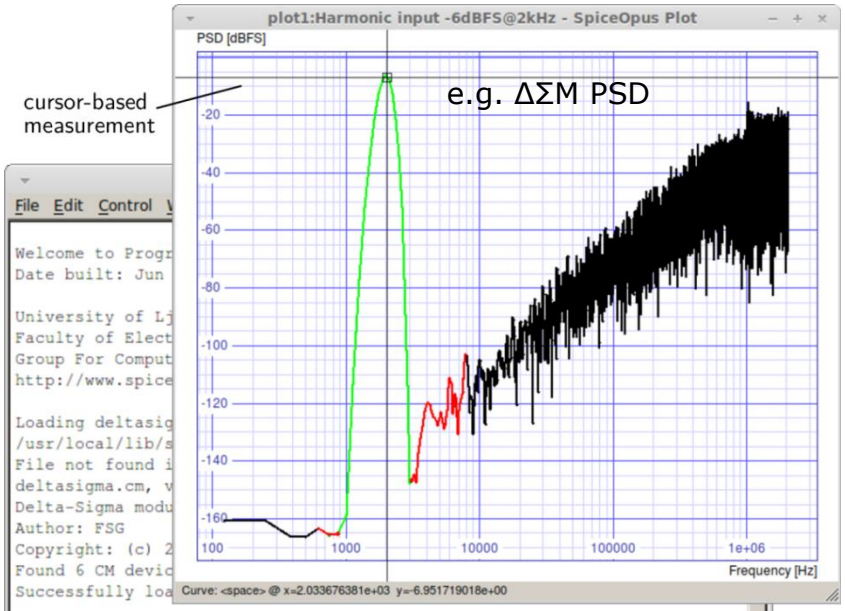
e.g. Z-domain integrator with built-in limiter XSpice code model

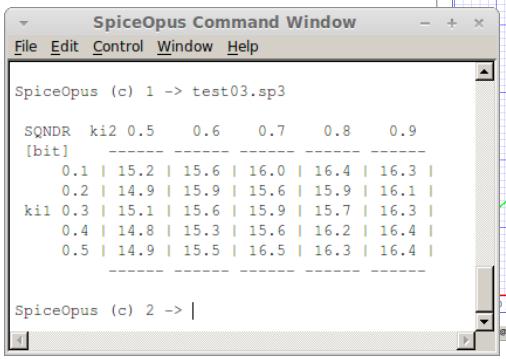
Architecture HDL Simulation

- Students can code in C their own mixed-mode XSpice **HDL models**
- **Nutmeg** scripting allows to manage several circuits and analysis at the same time
- Auxiliary **external programs** may be also combined



e.g. ΔΣM coefficients mismatching study





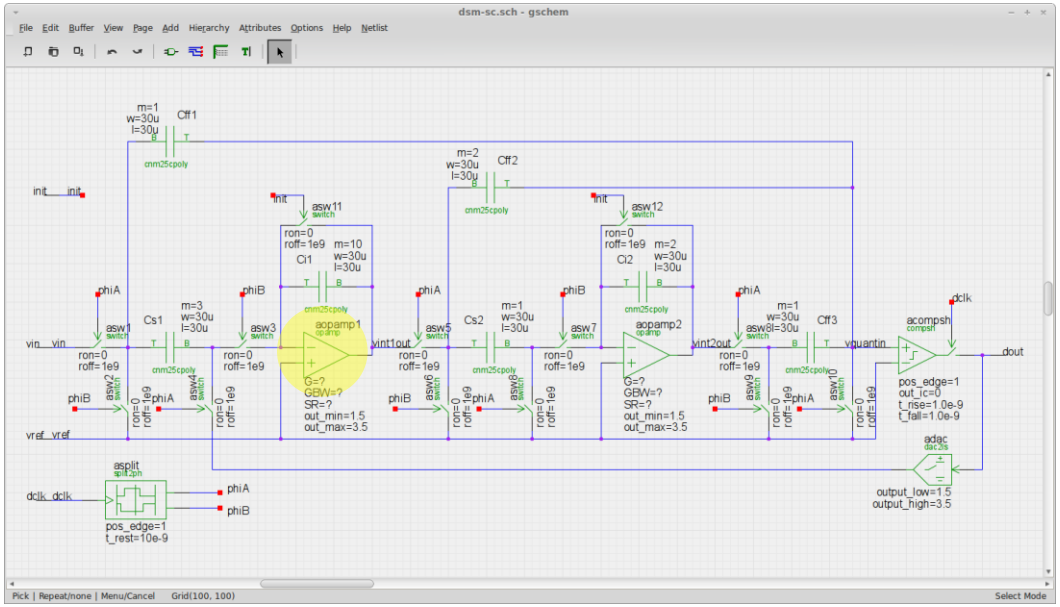
Block HDL Specification

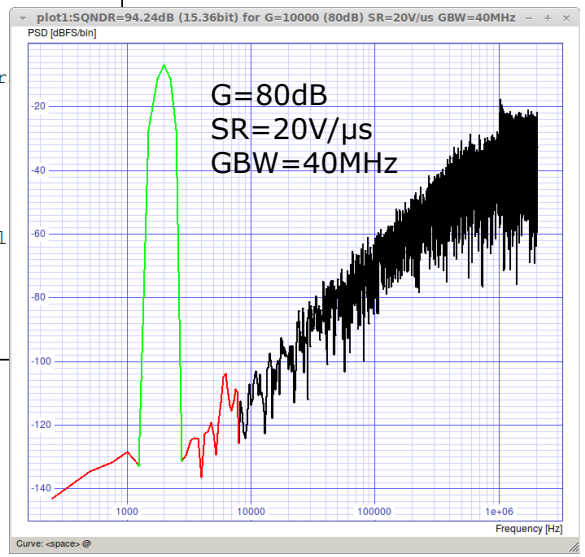
- Mixed **HDL-electrical** simulation to define circuit block requirements

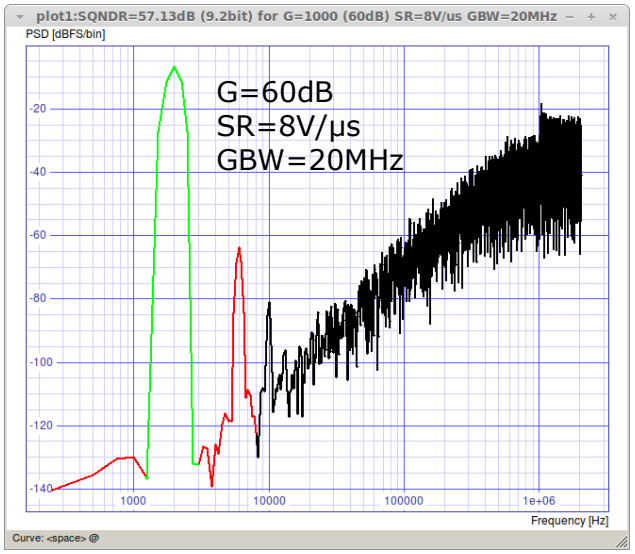
opamp.ifs

```
NAME_TABLE:
Spice_Model_Name:  opamp
C_Function_Name:   cm_opamp
Description:       "OpAmp macro"

PORT_TABLE:
Port_Name:         inp           inn           out
Description:       "pos. input"  "neg. input"  "output"
Direction:         in            in            out
Default_Type:      v             v             v
...
PARAMETER_TABLE:
Parameter_Name:    G             GBW           SR
Description:       "DC OL gain"  "GBW"         "slew-r
Data_Type:         real          real          real
Default_Value:     1000          1e6           1e6
Limits:            [0 -]         [0 -]         [0 -]
...
PARAMETER_TABLE:
Parameter_Name:    out_min              out_max
Description:       "lower out limit"    "upper out l
Data_Type:         real                 real
Default_Value:     0                    5.0
Limits:            [0 5.0]              [0 5.0]
...
```



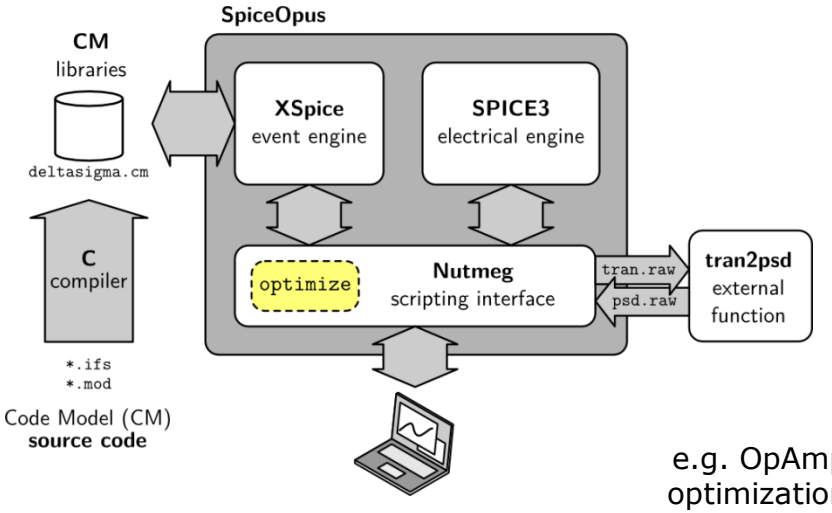




e.g. OpAmp specification study

Automatic Circuit Optimization

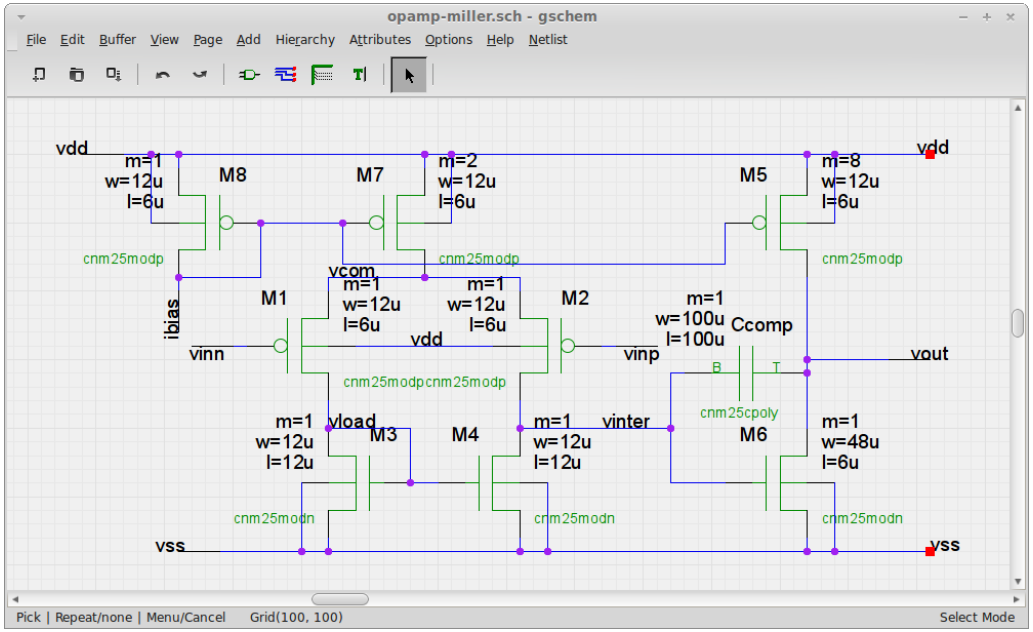
- Routine scripting:
  - Design **parameters**
  - Figures-of-merit (**FoMs**)
  - Implicit **rules** for discarding solutions
  - **Cost function** to score candidates



e.g. OpAmp optimization

opamp_optimize.sp3

```
optimize
   parameter 0 @m1:xopamp[w] low 6u high 120u initial 32u
 " parameter 1 @m6:xopamp[m] low 1 high 10 initial 8
 " parameter 3 @ccomp:xopamp[w] low 25u high 250u
...
 " analysis 25 ac dec 50 10 10e6
 " analysis 26 let gmag=20*log10(mag(v(vout)))
 " analysis 27 let gph=phase(v(vout))
 " analysis 28 cursor c right gmag 0
 " analysis 29 let gbw=abs(frequency[%c])/1e6
 " analysis 30 let pm=180+gph[%c]
...
 " analysis 46 tran 1n 5u
 " analysis 47 cursor c right vout 2.1
 " analysis 48 let t1=time[%c]
 " analysis 49 cursor c right vout 2.9
 " analysis 50 let t2=time[%c]
 " analysis 51 let srpos=0.8/(t2-t1)*1e-6
...
 " implicit 0 op2.pd lt 1.5
 " implicit 1 op2.area lt 0.025
 " implicit 4 ac2.pm gt 60
 " implicit 5 tran2.srpos gt 12
 ...
 " cost 1/tran2.srneg+1/tran2.srpos+abs(60-ac2.pm)
 " method genetic elitism yes maxgen 1000
```



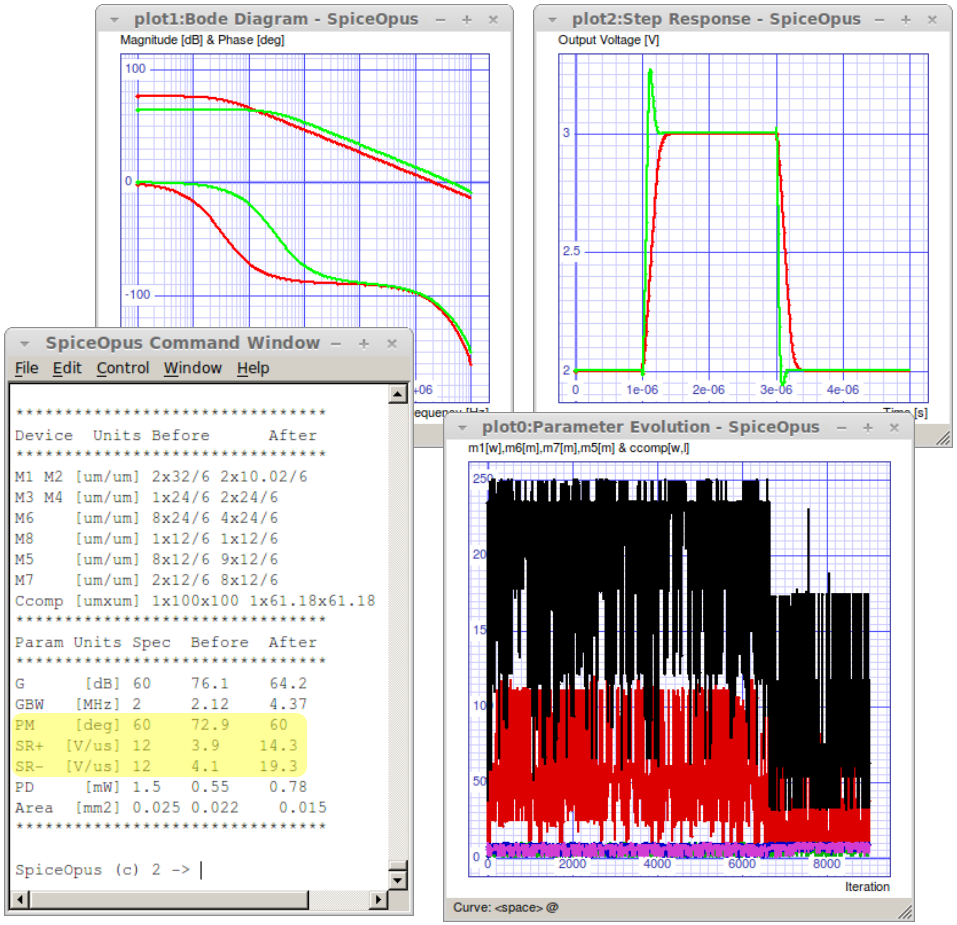
Automatic Circuit Optimization

- Routine scripting:
  - Design **parameters**
  - Figures-of-merit (**FoMs**)
  - Implicit **rules** for discarding solutions
  - **Cost function** to score candidates

opamp_optimize.sp3

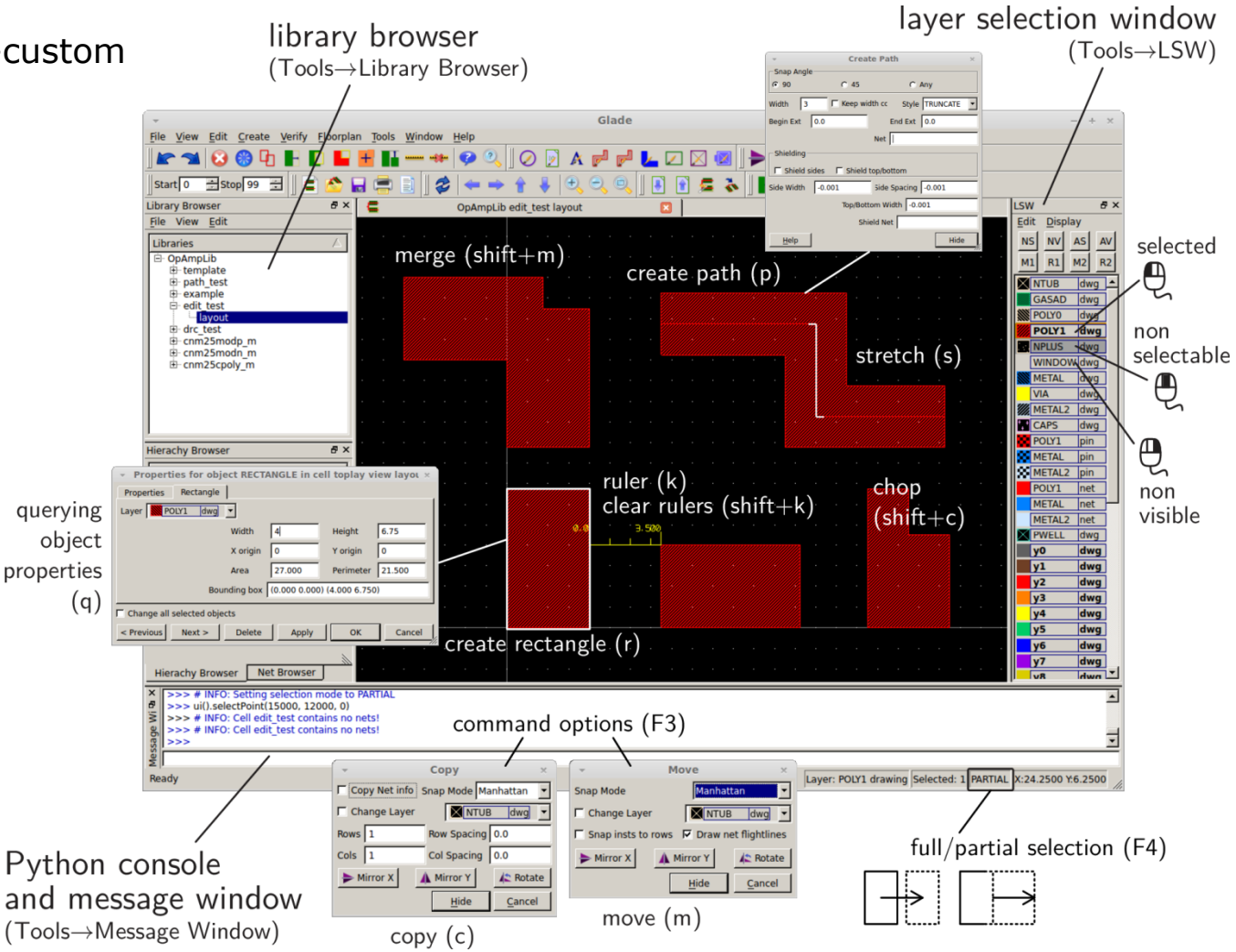
```
optimize
   parameter 0 @m1:xopamp[w] low 6u high 120u initial 32u
 " parameter 1 @m6:xopamp[m] low 1 high 10 initial 8
 " parameter 3 @ccomp:xopamp[w] low 25u high 250u
...
 " analysis 25 ac dec 50 10 10e6
 " analysis 26 let gmag=20*log10(mag(v(vout)))
 " analysis 27 let gph=phase(v(vout))
 " analysis 28 cursor c right gmag 0
 " analysis 29 let gbw=abs(frequency[%c])/1e6
 " analysis 30 let pm=180+gph[%c]
...
 " analysis 46 tran 1n 5u
 " analysis 47 cursor c right vout 2.1
 " analysis 48 let t1=time[%c]
 " analysis 49 cursor c right vout 2.9
 " analysis 50 let t2=time[%c]
 " analysis 51 let srpos=0.8/(t2-t1)*1e-6
...
 " implicit 0 op2.pd lt 1.5
 " implicit 1 op2.area lt 0.025
 " implicit 4 ac2.pm gt 60
 " implicit 5 tran2.srpos gt 12
 ...
 " cost 1/tran2.srneg+1/tran2.srpos+abs(60-ac2.pm)
 " method genetic elitism yes maxgen 1000
```

- Students can **customize** all steps and review **results**:



PCell-Based Layout Design

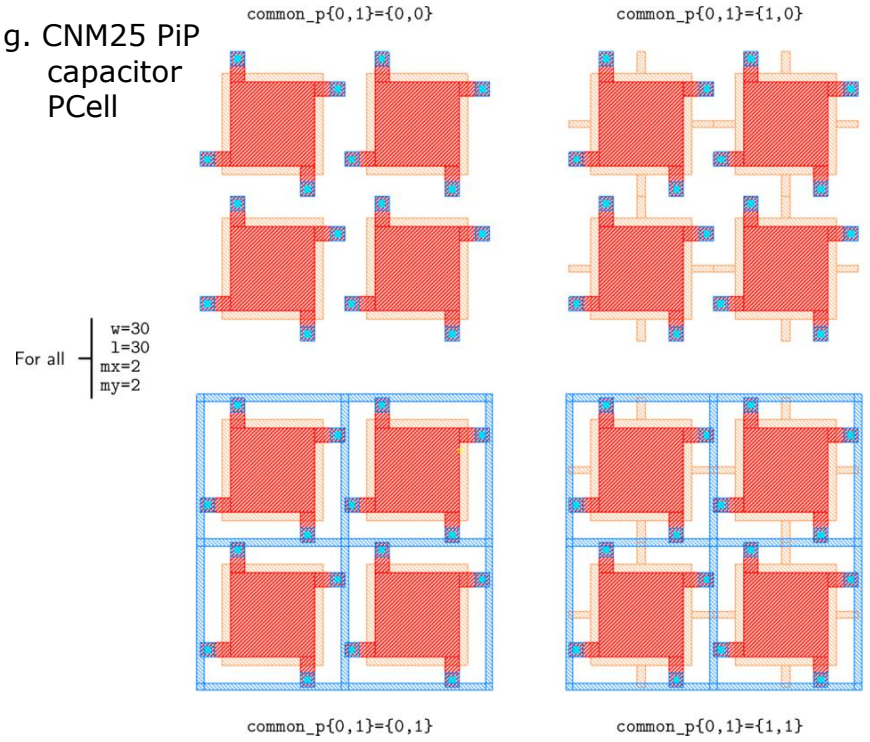
- Fully featured full-custom **layout editor**

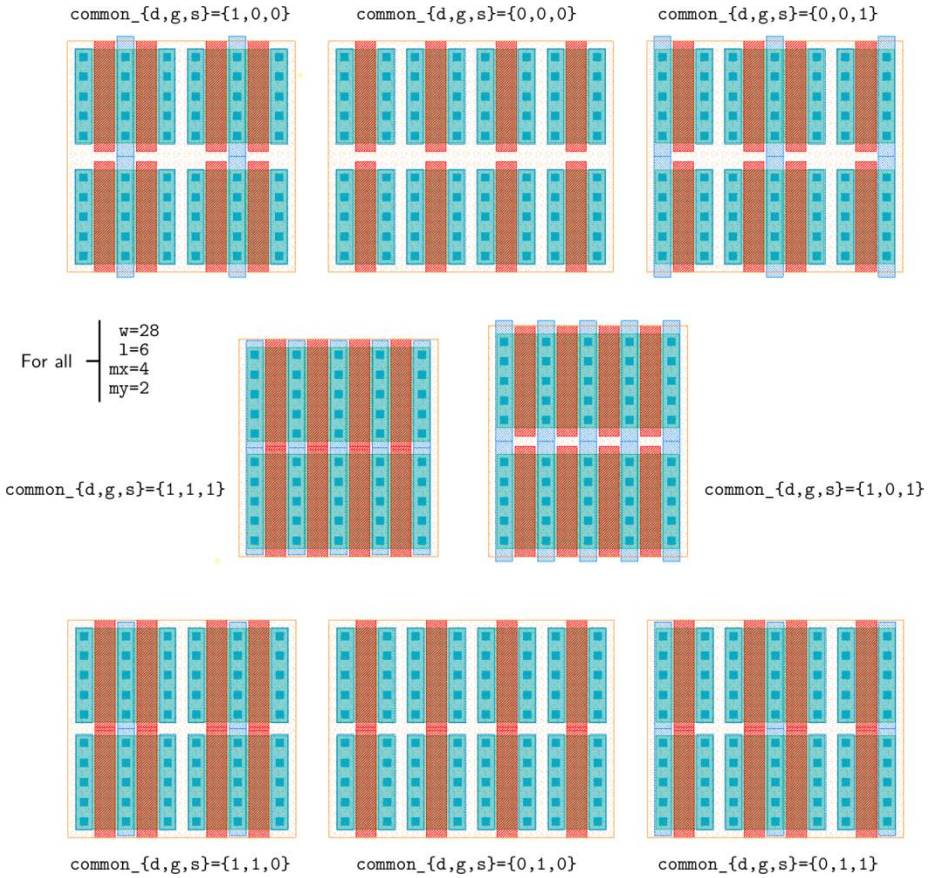


PCell-Based Layout Design

- Fully featured full-custom **layout editor**
- Parameterized cells (**PCells**) developed in Python

e.g. CNM25 PiP capacitor PCell



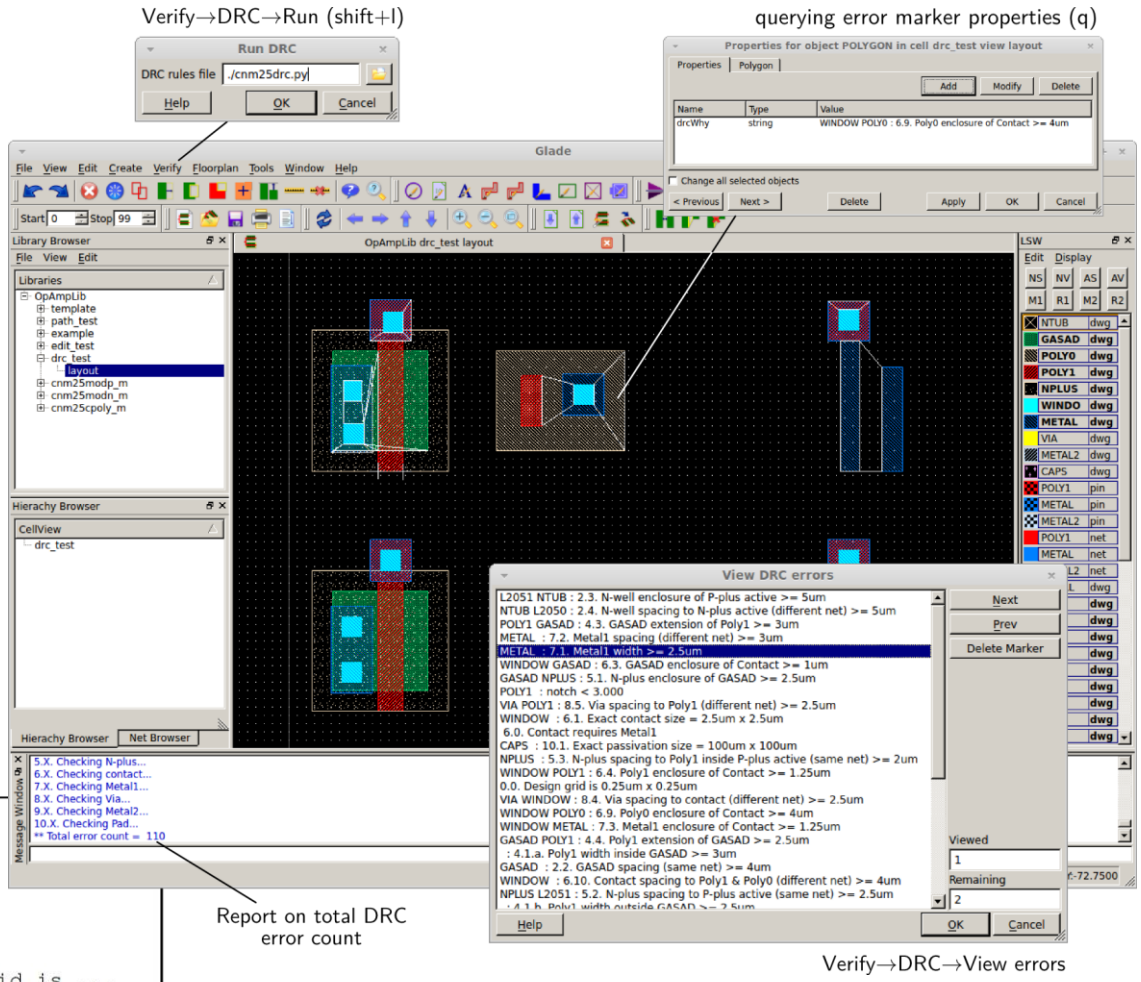


e.g. CNM25 NMOSFET PCell

- Students can design analog layout **quickly** while preserving **matching** rules

Design Rule Checker

- User **friendly interface** for debugging DRC errors
- **Design rules** set entirely scripted in Python
- Students can learn how a DRC is **programmed** (boolean operations, derived layers, geometrical concepts)



boolean operations

derived layers

cnm25drc.py

```
...
active   = geomGetShapes("GASAD", "drawing")
polygate = geomGetShapes("POLY1", "drawing")
polycap  = geomGetShapes("POLY0", "drawing")
gate     = geomAnd(polygate, active)
cpoly    = geomAnd(polygate, polycap)
geomOffGrid(polygate, 0.25, 1, "0.0. Design grid is ...
geomWidth(gate, 3, "4.1.a. Poly1 width inside GASAD >= ...
geomSpace(polygate, 3, diffnet, "4.2. Poly1 spacing...
geomNotch(polygate, 3, "4.2. Poly1 notch >= 3um")
geomExtension(polygate, active, 2.5, "4.4. Poly1 ext...
geomEnclose(polycap, cpoly, 3, "4.6. Poly0 enclosure...
...
```

e.g. CNM25 DRC script

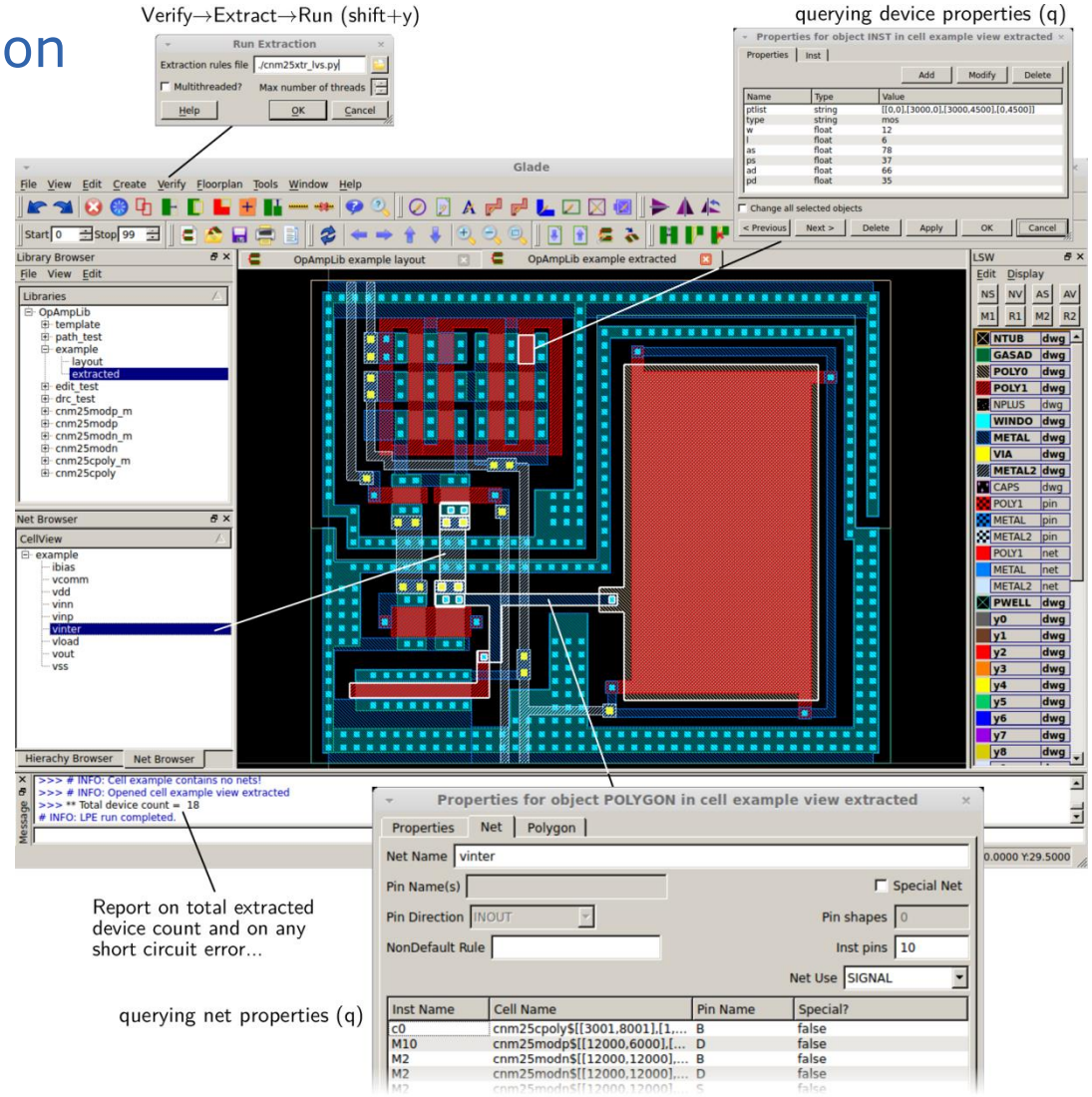
LVS and Parasitics Extraction

- User **friendly interface** for debugging ERC errors
- **Extraction rules** set entirely scripted in Python

e.g. CNM25 extraction script

cnm25xtr.py

```
...
geomLabel(polygate, "POLY1", "pin", 1)
geomLabel(polygate, "POLY1", "net", 0)
geomConnect([
  [cont, ndiff, pdiff, polygate, polycap, metal1],
  [via12, metal1, metal2]... ] )
extractMOS("cnm25modn", ngate, polygate, ndiff, pwell)
extractParasitic3(pdiff, metal2, cmetal2diff, 0,
  [metal1, polygate, polycap])
...
```



e.g. OpAmp layout extraction

LVS and Parasitics Extraction

- User **friendly interface** for debugging ERC errors
- **Extraction rules** set entirely scripted in Python
- Gemini-based layout versus schematic (**LVS**)
- Extraction of SPICE netlists with **parasitic capacitors** for post-layout simulation

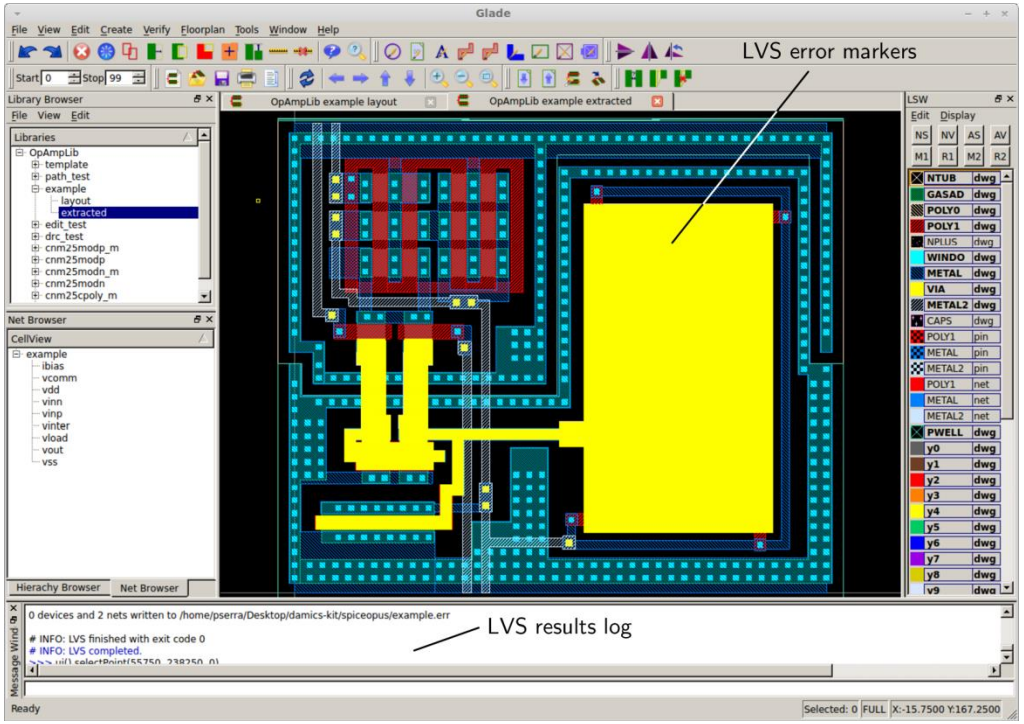
e.g. OpAmp extracted netlist

opamp_par.sub

```
.SUBCKT opamp vinn vinp vout vdd vss ibias
MM0 vdd ibias vdd vdd cnm25modp w=1.2e-05 l=6e-06 as=...
MM1 vdd ibias vout vdd cnm25modp w=1.2e-05 l=6e-06 as=...
Cc0 vinter vout cnm25cpoly w=6.42928e-05 l=0.000156207
MM8 vout ibias vdd vdd cnm25modp w=1.2e-05 l=6e-06 as=...
...
CP1 vinter vss  C=3.8582e-13
CP2 vout ibias  C=3.33692e-15
CP3 vinp vss  C=1.85938e-15
CP4 vout vcomm  C=2.0918e-15
...
.ENDS
```

parasitic caps

e.g. OpAmp with LVS errors



- Students can debug circuit connectivity or device size **matching** errors in the same environment
- Students can evaluate losses in circuit **performance** due to layout parasitics

Conclusions

- Complete **EDA** environment for teaching **mixed-mode full-custom** VLSI design
- Students can gain hands-on experience on:
  - Schematic entry
  - HDL system simulation
  - HDL block specification
  - Automatic circuit optimization
  - DRC and LVS
  - PCell-based layout
  - Parasitics extraction
- Practical A/D **ΔΣM circuit** design case in simple **CMOS** technology

INSTITUT DE MICROELECTRÒNICA DE BARCELONA CSIC

Other sources:

102726: Design of Analog and Mixed Integrated Circuits and Systems
http://www.cnm.es/~pserra/uab/damics

42838: Integrated Heterogeneous Systems Design
http://www.cnm.es/~pserra/uab/ihsd