Teaching Mixed-Mode Full-Custom VLSI Design

S. Sutula¹, F. Vila¹, J. Pallarès¹, K. Sabine², L. Terés¹,³ and F. Serra-Graells¹,³
jofre.pallares@imb-cnm.csic.es

¹Instituto de Microelectrónica de Barcelona, IMB-CNM(CSIC)
²Peardrop Design Systems Ltd
³Dept. of Microelectronics and Electronic Systems
Universitat Autònoma de Barcelona

May 2014
1. Introduction
2. Schematic Entry
3. Mixed-Mode HDL Simulation
4. Automatic Circuit Optimization
5. Full-Custom Layout Design and Verification
6. Conclusions
Motivation and Objectives

- **Problems** teaching VLSI design 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
1 Introduction

2 Schematic Entry

3 Mixed-Mode HDL Simulation

4 Automatic Circuit Optimization

5 Full-Custom Layout Design and Verification

6 Conclusions
Schematic Entry

▲ Fully customizable **symbols**

▲ Programmable **netlisting rules** for standard SPICE devices and custom XSpice models

e.g. ΔΣM architecture

```plaintext
.subckt dsm_arch vin dc1k dout
asumin [%v(vin) %v(vdac)] %v(verr) masumin
.model msumin usummer(sign=[1.0 -1.0])
  ak1 %v(verr) %v(vint1) mk1
  .model mk1 kgain(k=0.3)
azint1 %v(vint1) %d(dclk) %v(vint1out) mzint1
  .model mzint1 zinteg2lim(pos_edge=0 out_ic=0.0
  + out_min=-5.0 out_max=5.0
ak12 %v(vint1out) %v(vint2) mk12
  .model mk12 kgain(k=0.7)
aazint  %v(vint2) %d(dclk) %v(vint2out) mzint2
  .model mzint2 zinteg2lim(pos_edge=0 out_ic=0.0
  + out_min=-5.0 out_max=5.0
akf %v(vint1out) %v(vkffout) mkff
  .model mkff kgain(k=2.0)
asumout [%v(vint2out) %v(vkfout) %v(vin)]
  + %v(vquantin) masumout
  .model msumout usummer(sign=[1.0 1.0 1.0])
aquant %v(vquantin) %d(~dclk) %d(dout) nquant
  .model nquant quant2lim(inp_th=0.0 out_ic=0 pos_edge=0
  + t_rise=1e-9 t_fall=1e-9)
adc %d(dout) %v(vdac) mdac
  .model mdac dac2sym(out_level=2.0)
.ends
```

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

2 Schematic Entry

3 Mixed-Mode HDL Simulation

4 Automatic Circuit Optimization

5 Full-Custom Layout Design and Verification

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

```
v0 = INPUT(inp);  /* Retrieving input values */
clk = INPUT_STATE(clk);
```

```
if (!clk) { /* Neg. clk edge */
    action = SAMPLING_INTEGRATION;
} else {
    if (!clk_state) { /* Pos. clk edge */
        if (clk_state) action = SAMPLING_INTEGRATION;
    } else { /* No clock edge */
        action = HOLDING;
    }
}
```

```
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

---

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.
Block HDL Specification

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

```plaintext
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-rate"
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 limit"
Data_Type: real  real
Default_Value: 0    5.0
Limits: [0 5.0] [0 5.0]
```

e.g. OpAmp specification study

- G=80dB
  - SR=20V/µs
  - GBW=40MHz

- G=60dB
  - SR=8V/µs
  - GBW=20MHz
1 Introduction

2 Schematic Entry

3 Mixed-Mode HDL Simulation

4 Automatic Circuit Optimization

5 Full-Custom Layout Design and Verification

6 Conclusions
Automatic Circuit Optimization

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

```
optimize
  parameter 0 @ml:xopamp[w] low 6u high 120u initial 32u
  parameter 1 @m6:xopamp[m] low 1 high 10 initial 8
  parameter 3 @ecomp: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[%])/1e6
  analysis 30 let pm=180+gph[%]
...
  analysis 46 tran 1n 5u
  analysis 47 cursor c right vout 2.1
  analysis 48 let t1=time[%]
  analysis 49 cursor c right vout 2.9
  analysis 50 let t2=time[%]
  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

```plaintext
optimize
  parameter 0 @m1:opamp[w] low 6u high 120u initial 32u
  parameter 1 @m6:opamp[m] low 1 high 10 initial 8
  parameter 3 @ccomp:opamp[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
  analysis 29 let gbw=abs(frequency[Hz])/1e6
  analysis 30 let pm=180+gph[°]
  ...
  analysis 46 tran 1n 5u
  analysis 47 cursor c right vout 2.1
  analysis 48 let t1=time[%]
  analysis 49 cursor c right vout 2.9
  analysis 50 let t2=time[%]
  analysis 51 let srpos=0.8/(t2-t1)*1e-6
  ...
  implicit 0 op2.pm 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-ae2.pm)
  method genetic elitism yes maxgen 1000
```
1 Introduction

2 Schematic Entry

3 Mixed-Mode HDL Simulation

4 Automatic Circuit Optimization

5 Full-Custom Layout Design and Verification

6 Conclusions
PCell-Based Layout Design

- Fully featured full-custom layout editor

Library browser (Tools→Library Browser)

Layer selection window (Tools→LSW)

Querying object properties (q)

Merge (shift+m)

Create path (p)

Stretch (s)

Ruler (k)

Clear rulers (shift+k)

Chop (shift+c)

Create rectangle (r)

Command options (F3)

Copy (c)

Move (m)

Python console and message window (Tools→Message Window)

Selected

Non selectable

Non visible

Full/partial selection (F4)
PCell-Based Layout Design

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

Example PCells:
- 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)

```python
... active = geomGetShapes("GASAD", "drawing")
polygate = geomGetShapes("POLY1", "drawing")
polycap = geomGetShapes("POLY0", "drawing")
gate = geomAnd(polygate, active)
cpoly = geomAnd(polygate, polycap)
gemOffGrid(polygate, 0.25, 1, "0.0. Design grid is ...
gemWidth(gate, 3, "4.1.a. Poly1 width inside GASAD => ...
gemSpace(gate, 3, diffnet, "4.2. Poly1 spacing...
gemNotch(gate, 3, "4.2. Poly1 notch >= 3um")
gemExtension(polygate, active, 2.5, "4.4. Poly1 ext...
gemEnclose(polycap, cpoly, 3, "4.6. Poly0 enclosure...
```

- e.g. CNM25 DRC script

![Image of Design Rule Checker interface and Python script example]
LVS and Parasitics Extraction

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

E.g. CNM25 extraction script

```python
... 
cnm25xtr.py
...
geomLabel(polygate, "POLY1", "pin", 1)
geomLabel(polygate, "POLY1", "net", 0)
geomConnect([cont, ndiff, pdiff, polygate, polycap, metall1],
[vial1, metall1, metall2]... ])
eextractMOS("cnm25modn", ngate, polygate, ndiff, pwell)
eextractParasitic3(pdiff, metal2, cmetal2diffeff, 0,
[metall1, polygate, polycap])
...
```

- Report on total extracted device count and on any short circuit error...
- Querying net properties (q)
- 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**

```
.SUBCKT opamp vinn vinp vout vdd vss ibias
MM0 vdd ibias vdd vdd cmm25modp w=1.2e-05 l=6e-06 as=-...
MM1 vdd ibias vout vdd cmm25modp w=1.2e-05 l=6e-06 as=-...
Co0 vinter vout cmm25poly w=6.42928e-05 l=0.000156207
MM0 vout ibias vdd vdd cmm25modp w=1.2e-05 l=6e-06 as=-...
... CP1 vinter vss C=3.8592e-13
CP2 vout ibias C=3.33692e-15
CP3 vinp vss C=1.85938e-15
CP4 vout vcomm C=2.09180e-15
... END
```

**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
<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Introduction</td>
</tr>
<tr>
<td>2</td>
<td>Schematic Entry</td>
</tr>
<tr>
<td>3</td>
<td>Mixed-Mode HDL Simulation</td>
</tr>
<tr>
<td>4</td>
<td>Automatic Circuit Optimization</td>
</tr>
<tr>
<td>5</td>
<td>Full-Custom Layout Design and Verification</td>
</tr>
<tr>
<td>6</td>
<td>Conclusions</td>
</tr>
</tbody>
</table>
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

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