CAD Techniques to Automate Analog Cell Design

Rob A. Rutenbar
Carnegie Mellon University
Pittsburgh, PA, USA
rutenbar@ece.cmu.edu
http://www.ece.cmu.edu/~rutenbar
Talk’s Emphasis

- Analog cells
- CAD & methodology issues
- Synthesis, reuse, IP options

Example: one cell on analog-side of a mixed-signal ASIC
Critical design tasks
- Circuit design: topology, sizing, centering
- Circuit layout: devices, placement, routing

About analog cells
- Why analog cells != digital cells
- Different design and reuse scenarios
- Different intellectual property (IP) issues

CAD & methodology
- Current methodologies: today's industrial coping strategies
- Evolving techniques: leading-edge strategies, universities, startups

Conclusions
Historically—Why is this so *Hard*?

- Mediocre analog point tools
- Ad hoc, incomplete capture of design intent
- Too much art, not enough science
Outline

- Critical design tasks
  - Circuit design: topology, sizing, centering
  - Circuit layout: devices, placement, routing
- About analog cells
- CAD & methodology
- Conclusions
Just What Is An “Analog Building Block?”

- **Typical analog cell**
  - ~5-75 devices (if bigger, usually use some hierarchy)
  - Active devices (FET, BJT, etc) and passives (R, L, C)
  - Often requires precision devices/passives for performance
  - Often requires sensitive device placement, wiring

![Circuit diagram](image)

**Specification**

- Gain 60dB
- UGF 111MHz
- Phase 60deg
- Slew 2V/µs
- CMRR: 60dB
- PSRR: 70dB
- THD: 1%

**Circuit topology & sizing**

**Physical layout**

Need all 3 of these to have a “complete” cell
Analog Cells: Common Examples

- Common cells
  - OpAmp
  - Comparator
  - Bandgap Voltage Ref
  - Analog Switch
  - Oscillator
  - LNA
  - Mixer
  - Etc...

- Common subsystems composed from basic cells
  - Filter
  - PLL
  - General A/D & D/A
  - Audio ΔΣ A/D
  - Regulator
  - CODEC
  - I/O Line Drivers
  - Etc...
Analog Cell Design: Critical Tasks

- No matter *how* you do it, you have to do these tasks
  - Basic *device-level circuit design*

---

Gain 60dB  
UGF 111MHz  
Phase 60deg  
Slew 2V/μs  
CMRR: 60dB  
PSRR: 70dB  
THD: 1%  
...

Generate proper specs  
Choose proper circuit topology  
Design proper device sizing/biasing  
Optimize for centering, yield
Analog Cell Design: Critical Tasks

- No matter how you do it, you have to do these tasks
  - Basic *device-level layout design*

From sized schematic  ➔ Choose proper cell footprint  ➔ Design individual device geometries  ➔ Place/route devices, optimize area, coupling, etc.
Outline

- Critical design tasks
- About analog cells
  - Why analog cells != digital cells
  - Different design and reuse scenarios
  - Different IP issues
- CAD & methodology
- Conclusions
Why Is This Actually Difficult…?

- **Common misperceptions here**
  - Based mostly on familiarity with digital cells, digital libraries, and with digital design scenarios

- **Myth of “limited size”**
  - “Hey--only 50 transistors? How hard can that be to design?”
  - “I don’t see people obsessing over NAND gate design!”

- **Myth of “limited libraries”**
  - “There’s not much analog on chip, and it’s mostly understood functions like A/D and D/A, so why not just design all the required cells once, put them in a library, reuse them?”
Reminder: Cell-Based Digital Design

- Digital ASIC design
  - Often **starts** from assumed library of cells (maybe some cores too)
  - Supports changes in cell-library; assumed part of methodology
  - Cell libraries heavily **reused** across different designs

![Diagram of Digital HDL, Logic Synthesis, Tech Mapping, Physical Design, and Gate-Level Cell Library](image)
Cell-Based Design Strategies: Digital

Where do digital cells come from?

- **Foundries:**
  - Optimized for this fab

- **3rd Party IP:**
  - Emphasize portability, quick use

- **Migration Tools:**
  - Old cells -> new cells

- **Manual, Custom Design:**
  - Proprietary or custom library
Cell-Based Design Strategies: Analog

Where do analog cells come from?
- Mainly manual design
- Often, manual redesign
- Not much device-level reuse
- Significant design effort here
- (Some IP is emerging…)

Why is this?
Analog Cells: Strong Fab Dependence

- No digital abstraction to “hide” process
  - No logic levels, noise margins, etc, on analog cells

- Exploits physics of fab process, instead of avoiding it
  - Individual devices designed to achieve precise behaviors
  - Especially true with precision passive devices, which might require separate process steps (e.g., double poly for capacitors)
  - Circuits sensitive to all aspects of device/interconnect behavior, even modest changes due to simple dimensional shrinks

Can’t hide behind nice 1s and 0s...
Analog Cells in Digital Processes

- For SoC designs, want analog in standard digital process
- Common problems
  - Low supply voltages preclude some circuit topologies
  - Precision structures may be hard/impossible to build if special layers are unavailable (e.g., poly-poly capacitor)
  - Digital processes do not characterize devices for analog uses, e.g., models do not capture subthreshold ops, matching, etc

4-high gate stack works fine in 2µm, fails in deep submicron due to lack of $\Delta V_{GS}$
Analog Cell Myths Revisited

- **Cell design difficulty, libraries**
  - OK, so, maybe it’s hard to design an analog cell.
  - So, why not just **design it once**, add to lib, reuse it?

- **Problem: leverage not same for analog libraries**
  - How big is a digital library? Big enough to get all necessary logic functions, IO variants, timing variants, drive strengths, to first order

\[
\text{Logic functions} \times \text{Fanin \\& fanout variants} \times \text{Timing, latch/FF, scan variants} \times \text{Drive strength (1X, 2X, 4X, 8X) variants} = \sim 1\text{k-2k cells}
\]
Analog Cell Libraries: Dimensionality

Problem: many continuous specs for analog cells

\[ \text{10 independent performance specifications} \]

\[ \text{Spec=LOW} \quad \text{Spec=HIGH} \]

\[ \times \quad \text{for ALL combinations} \]

\[ \approx 1000 \text{ variants for just this cell} \]

Can’t just build a practical-size, universal analog library
Analog Cell Libraries: Dimensionality

- Dimensionality: Reality check
  - OK, do you really need all 1000 of those variants?
  - Can’t we make do with just a few—like we do for digital gates?
- Maybe: depends on your application

At modest levels of performance, you may be able to survive with limited variants, specs.

But not out here, on high-performance apps, where every spec matters, most are interdependent, and there is little slack on meeting design goals.
Analog Cells: Design & Reuse Strategies

- 2 major issues
  - How do I make it easier to design this cell in the first place?
  - How do I avoid designing it again? Can I reuse it, wrap/buy it as IP?
  - Actually, **interdependent** set of technical responses here

- Design: focuses at 3 levels
  - Device-level design
  - Cell-level design
  - Core-level design (this is mostly ongoing research)

- IP/reuse: focuses on 3 strategies
  - Hard
  - Firm
  - Soft
Analog Cells: Design & Reuse Strategies

- Simple taxonomy

<table>
<thead>
<tr>
<th>IP/REUSE</th>
<th>hard</th>
<th>firm</th>
<th>soft</th>
</tr>
</thead>
<tbody>
<tr>
<td>device</td>
<td>Libraries of difficult, exotic device layouts</td>
<td>Parametric device layout generators</td>
<td>--</td>
</tr>
<tr>
<td>cell</td>
<td>Libs of generic cell layouts for specific fab</td>
<td>Parametric templates for schematic, layout</td>
<td>Analog ckt synthesis and layout synthesis</td>
</tr>
<tr>
<td>core</td>
<td>Libs of useful block layouts for specific fab</td>
<td>Parametric templates for useful cores</td>
<td>Mixed-signal system synthesis</td>
</tr>
</tbody>
</table>

Focus is on:
- layout reuse
- reusable circuit & layout templates
- synthesis, from spec to ckt to layout
Outline

- Critical design tasks
- About analog cells
- **CAD & methodology**
  - Current methodologies: today's industrial coping strategies
  - Evolving techniques: leading-edge strategies, universities, startups
- Conclusions
**What are people *most commonly* doing right now?**

(Actually, they’re mostly designing *by hand*, one device at a time…)

<table>
<thead>
<tr>
<th>IP/REUSE</th>
<th>hard</th>
<th>firm</th>
<th>soft</th>
</tr>
</thead>
<tbody>
<tr>
<td>device</td>
<td>Libraries of difficult, exotic device layouts</td>
<td>Parametric device layout generators</td>
<td>--</td>
</tr>
<tr>
<td>cell</td>
<td>Libs of generic cell layouts for specific fab</td>
<td>Parametric templates for schematic, layout</td>
<td>Analog ckt synthesis and layout synthesis</td>
</tr>
<tr>
<td>core</td>
<td>Libs of useful block layouts for specific fab</td>
<td>Parametric templates for useful cores</td>
<td>Mixed-signal system synthesis</td>
</tr>
</tbody>
</table>
First, Look at Device-Level Issues

- Question: why the emphasis on *individual* devices…?

<table>
<thead>
<tr>
<th>IP/REUSE</th>
<th>hard</th>
<th>firm</th>
<th>soft</th>
</tr>
</thead>
<tbody>
<tr>
<td>device</td>
<td>Libraries of difficult, exotic device layouts</td>
<td>Parametric device layout generators</td>
<td>--</td>
</tr>
<tr>
<td>cell</td>
<td>Libs of generic cell layouts for specific fab</td>
<td>Parametric templates for schematic, layout</td>
<td>Analog ckt synthesis and layout synthesis</td>
</tr>
<tr>
<td>core</td>
<td>Libs of useful block layouts for specific fab</td>
<td>Parametric templates for useful cores</td>
<td>Mixed-signal system synthesis</td>
</tr>
</tbody>
</table>
Analog Device IP

- **Basic idea**
  - Analog cells require “difficult” device structures
  - May need large devices, aggressive matching, unusual precision
  - Can save device layouts in a library, or more commonly...
  - ... write **layout generators**; may be provided by your foundry
  - Implementations vary: can use commercial frameworks (Mentor GDT, Cadence PCELL), or write your own (C++, JAVA, etc)
Device-Level Design Issues

- **Focus is often on precision**
  - May want precise electrical characteristics, or matching among several devices, or precise ratios among devices

- **Central issues**
  - Analog devices are often large; e.g., a 40000/4 FET is not unusual
  - Analog devices are often designed and laid out as a careful connection of many small, well-matched unit-size devices
  - Guard-ring(s) common for electrical isolation

- **Result**
  - Even one device may end up with a complex, large geometric layout
Example of Digital vs Analog Size Disparity

Digital FET

Analog FET
Consider a resistor which uses a resistive poly layer

- **Low-precision R, poly snake resistor**
- **High-precision R, add dummy bars at ends, well and guard ring**
- **Interdigitated pair of precise-ratio 2:1 resistors**

- **Resistive material**
- **Metal-strapped pins**
- **Higher-precision R, poly bars with all-metal interconnect**
Industrial Example: Large Resistor Array

Courtesy Neolinear
Analog Device IP: Analysis

**PRO**
- Easier to get complex devices, device groups, laid out correctly
- Easier to get careful precision structures laid out correctly
- Insulates users from some of the nastier low-level foundry rules

**CON**
- Easy as a concept, hard in practice to build good generators
- Like any library (hard or generator), maintenance is an issue
- Does not help in sizing the circuit in the first place
- Does not remove requirement to place/route these devices into a functioning cell, with its own precision/performance subtleties
Next, Look at Hard Analog IP

Question: how much can you reuse complete layouts?

<table>
<thead>
<tr>
<th>IP/REUSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>hard</td>
</tr>
<tr>
<td>firm</td>
</tr>
<tr>
<td>soft</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DESIGN</th>
<th>hard</th>
<th>firm</th>
<th>soft</th>
</tr>
</thead>
<tbody>
<tr>
<td>device</td>
<td>Libraries of difficult, exotic device layouts</td>
<td>Parametric device layout generators</td>
<td>--</td>
</tr>
<tr>
<td>cell</td>
<td>Libs of generic cell layouts for specific fab</td>
<td>Parametric templates for schematic, layout</td>
<td>Analog ckt synthesis and layout synthesis</td>
</tr>
<tr>
<td>core</td>
<td>Libs of useful block layouts for specific fab</td>
<td>Parametric templates for useful cores</td>
<td>Mixed-signal system synthesis</td>
</tr>
</tbody>
</table>
Hard Analog Cell IP

- Basic idea
  - Hard IP (layouts) for common, generic cell functions
  - Performance ranges estimated to target common application areas (e.g., audio, video, LAN, IO driver, etc)
  - Available from some foundries; also some 3rd party IP shops who design for standard digital fabs

Tend to stay away from **maximally aggressive** performance specs; target common mid-range performance.
Hard Analog Cell IP: Analysis

- **PRO**
  - Again, makes it easy to do some simple functions

- **CON**
  - Unlike digital libraries, *unlikely* that 100% of needed cells available
  - And, cell portfolio may differ significantly from vendor to vendor

Your mixed signal ASIC

Vendor 1 Coverage

Vendor 2 Coverage

Vendor 3 Coverage

Sorry, this requires *custom* analog--more design effort, impact on design risk
Recent commercial idea

- Don’t focus on basic cells, focus on **bigger mixed-signal cores**
- Industry standards **fix** many specs; target big ASIC foundries
- Interesting technical (& business) issues here

**MixSig Core**

- PLL
- A/D, D/A
- Filter
- Codec
- Ethernet IO
- Firewire IO, ….

Hide low-level analog; basic cells hand-crafted to exploit foundry process
Analog Cores: Design Issues

- Not necessarily all hard (fixed layout) approaches here
  - Can do modest parameterization on cells--if they don’t vary much
  - Can relax foundry rules to create “subset” rules that work across several similar processes (e.g., foundry 0.25\(\mu\)m); lose some density and performance, gain some reuse
  - Can design some of the circuits themselves to be programmable, eg, a programmable bandgap voltage reference, programmable gain stage etc. Again, trade some density/performance for reuse.

- Of course...
  - The people who actually design these cells still have all the problems of anybody who has to design custom analog
  - You get lucky if you can buy it from them...
Hard Analog Core IP: Analysis

- **PRO**
  - Good idea--when it works technically, and as a business
  - Scene evolving quite rapidly here
  - Lots of common IO interfaces require analog; productivity benefit to be able to buy this functionality

- **CON**
  - Functionality, versatility still limited
  - **Obtaining** an analog core != **integrating** an analog core; noise, coupling issues still difficult for big mixed signal ICs
  - No guarantees to be able to find function, speed, power, etc. you need, in the fab process you use today…or tomorrow
  - If you can’t buy it…you still have to design it yourself
OK, suppose you can’t just buy the analog cells you need; what can you do to help design them faster, better?

<table>
<thead>
<tr>
<th>IP/REUSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>hard</td>
</tr>
<tr>
<td>device</td>
</tr>
<tr>
<td>cell</td>
</tr>
<tr>
<td>core</td>
</tr>
</tbody>
</table>
Cell-Level Strategies

- Aside from doing everything manually, are there options?
- **Template-based design**
  - If you keep designing the same cells, for similar ranges of performance, try to capture central characteristics as a template
  - Parameters fill in the template, change resulting design

- **Analog synthesis**
  - For more general case, specify critical performance constraints (electrical, geometric, etc)
  - Synthesis tool uses numerical/geometric search to create circuit to match your design goals

- Actually, these are variants on same technical theme...
Analog Cell Synthesis

- **Basic idea**
  - **Circuit synthesis:** transform cell spec into sized/biased schematic
  - **Circuit layout:** transform device-level netlist into laid-out cell

- Mimics ideas from digital logic/layout synthesis
- But, focus is transistor-level synthesis
- A few alternative approaches
Central idea is \textit{not} to start from scratch on each new design.

Difference here is \textit{who} does most of the work...

\textbf{Parametric templates:}
- Designer has initiative, makes effort
- Identifies commonalities among designs
- Extracts & encodes in reusable way
  \textit{More designer effort, less CPU time}

\textbf{Circuit/layout synthesis:}
- Designer specifies specs, constraints
- New discipline: need complete specs
- Tools do numerical, geometric search
  \textit{More CPU time, less designer effort}
Cell-Level Analog Circuit Synthesis

- **Basic task**

- **Major strategies**
  - Procedural scripting
  - Equation-based search-- flat and hierarchical
  - Symbolic analysis
  - Simulation-based optimization

Gain 60dB
UGF 111MHz
Phase 60deg
Slew 2V/us
CMRR: 60dB
PSRR: 70dB
THD: 1%

Design topology
Design sizing/biasing
Center *(maybe)*

Most approaches have this overall structure:

- **Optimization Engine**: proposes candidate circuit solutions
- **Evaluation Engine**: evaluates quality of each candidate
- **Cost-based search**: cost metric represents “goodness” of design

Uses heuristic or numerical search:

- **Optimization engine**: proposes candidate circuit solutions
- **Evaluation engine**: evaluates quality of each candidate
- **Cost-based search**: cost metric represents “goodness” of design
Basic idea
- Capture equations, models, calculations you keep re-solving in sensible, solvable order
- Write a program -- a script -- that does it
- Analogy: a spreadsheet

Issues
- OK for simple circuits, if you have good models, require modest parameter changes
- Hard (impossible) to write for complex circuits
- Can’t get good analytical model for all specs
- Often problems with accuracy (vs. simulation models), robustness

Examples:
[DeGrauwe, JSSC'87]
[Harvey, TCAD'92]
Procedural Scripting: Mirror Example

Function

Fixed Topology

Controlling Node

Output Node

IN

OUT

M1

M2

Design Vars:

Device Model:

Design Vars:

Device Model:

Input Specs:

Heuristic Design Script

\[ W_1 = \frac{2}{\mu_n C_{OX} V_{c-min}^2} \cdot \frac{I_{in} L_1}{\mu_n C_{OX} V_{c-min}^2} \]

\[ L_1 = L_2 \]

\[ W_2 = M W_1 \]

\[ L_2 = \lambda \cdot I_{out} r_{o-min} L_{min} \]
Synthesis: Equation-Based Optimization

- **Basic idea**
  - Capture equations, models, etc.
  - Can’t script everything analytically; use numerical search
  - Styles vary: gradient search, annealing, geometric (convex) programming, ILP, ...

- **Issues**
  - Supports wider set of design, goals
  - Writing correct equations still *very* hard, laborious; eqns often fragile, short lifespan
  - Can’t get good analytical model for all specs
  - Accuracy problems (vs. simulation), numerical starting-point dependency

Examples:
- [Koh, TCAD’90]
- [Hershenson, ICCAD’98]
Eqn-Based Optimization: Example

- Example: posynomial-formulation [Hershenson ICCAD98]
  - If you can render all equations as posynomials (like polynomials, but real-valued exponents and only positive terms, eg $3x^2y^{2.3}z^{-2}$), can show resulting problem is convex, has one unique minimum
  - Geometric programming can solve these to optimality

Example: opamp circuit synthesized, fabbed in TSMC 0.35µm CMOS

<table>
<thead>
<tr>
<th>Spec</th>
<th>GP</th>
<th>SPICE</th>
<th>Measured</th>
</tr>
</thead>
<tbody>
<tr>
<td>Power (mW)</td>
<td>≤ 2</td>
<td>2</td>
<td>2.1</td>
</tr>
<tr>
<td>DC gain (dB)</td>
<td>≥ 70</td>
<td>73</td>
<td>76</td>
</tr>
<tr>
<td>UGBW (MHz)</td>
<td>Max.</td>
<td>19</td>
<td>19</td>
</tr>
<tr>
<td>Phase margin (°)</td>
<td>≥ 60</td>
<td>≥ 60</td>
<td>63</td>
</tr>
<tr>
<td>Slew rate ($\frac{V}{\mu s}$)</td>
<td>≥ 30</td>
<td>38</td>
<td>33</td>
</tr>
<tr>
<td>Noise, 1kHz ($\frac{V}{\sqrt{Hz}}$)</td>
<td>≤ 400</td>
<td>393</td>
<td>390</td>
</tr>
<tr>
<td>Area (µm²)</td>
<td>≤ 10k</td>
<td>4.8k</td>
<td>-</td>
</tr>
</tbody>
</table>

Optimal trade-off curves

Courtesy Mar Hershenson, Stanford
Synthesis: Hierarchical Search

- **Basic idea**
  - Equation-based search, but use hierarchical representation of circuit
  - Even small circuits have components: mirrors, references, gain stages, etc
  - Build eqns for pieces, assemble into circuit

- **Issues**
  - More easily supports search over circuit topology and circuit sizing at same time
  - Eases some of the burden of writing eqns--but still have to get eqns for components
  - Some “deep” optimizations more difficult when circuit partitioned into pieces
  - Same accuracy/robustness problems of eqns

Examples:
- [Harjani DAC’87]
- [Gielen, JCTh’95]
Hierarchical Circuit Synthesis

- **Selection** = pick an abstract design style (sub-block topology)
- **Refinement** = decompose parent performance specs for child

Op Amp Specs
- Gain
- Slew
- UGF

Style Selection

Level 0
- Style 1
- Style 2

Level 1
- Mirror Specs
  - Impedance
  - Current
  - Max voltage

1-stage (OTA)
- mirror
- mirror
- diff pair
- mirror

[Harjani DAC’87]
Aside: Gets More Interesting at System Level

- Use these ideas to explore system spec/architecture tradeoffs

- Compare and optimize power

- Analyze frontend topology

- [ORCA, FAST, FONZIE...]

- Power (W)

- SNR (dB)

- Architecture 1

- Architecture 2

© R.A. Rutenbar 2001
Synthesis: Symbolic Analysis

- **Basic idea**
  - Automatically derive eqns--when you can
  - Support powerful symbolic manipulation
  - Add designer-derived eqns for remainder
  - Use numerical optimization on *these* eqns

- **Issues**
  - Works well, but restricted to linear, weakly-nonlinear specifications, behaviors
  - Can work for continuous/discrete time (t/z)
  - Can support useful interactive modes
  - “Transient waveform” specs not well captured
  - Same accuracy/robustness problems as eqns

Examples:

- [Gielen, JSSC’90]
- [Wambacq, JSSC’95]
- [Sechen, TCAD’97]
Symbolic Analysis: Simple Example

- Basic idea: prune symbolic form
  - Symbolically manipulate determinant of admittance matrix

Toy example

\[ Z_{\text{out (full)}} = \frac{2g_{12} g_{m1} g_{m2} + g_{1} g_{2} g_{m1} + g_{2} g_{m2} g_{1} + g_{2} g_{m1} g_{m2} + g_{m2} g_{o1} g_{\pi1} + g_{1} g_{2} g_{\pi2}}{g_{2} g_{m2} g_{\pi1} + g_{1} g_{2} g_{\pi2} + g_{2} g_{m1} g_{\pi1} + g_{1} g_{2} g_{\pi2} + g_{m2} g_{o1} g_{\pi2} + g_{01} g_{\pi1} (g_{2} g_{m2} + s C_{b}) (g_{1} g_{\pi1} + g_{\pi2}) (g_{2} g_{m2} + s C_{b} g_{1})} \]

\[ Z_{\text{out (pruned)}} = \frac{g_{m1} g_{m2} (g_{2} + s C_{b})}{g_{01} g_{\pi1} (g_{2} g_{m2} + s C_{b} g_{1})} \]
Symbolic Analysis: Realistic Example

- Katholieke Univ. Leuven, ISAAC/SYMB tool [Gielen JCTh’95]

\[
A_{V0} = \frac{g_{m,M2}}{g_{m,M1}} \left( \frac{g_{o,M4}g_{o,M5}}{g_{m,M5} + g_{mb,M5}} \right) + \frac{g_{m,M4}}{\beta_{Q2}} \left( G_a + g_{o,M9} + g_{o,Q2} \right)
\]

Courtesy Georges Gielen, KUL
## Bigger Circuit Example

- **KU Leuven, AMGIE tool, [Gielen JCTH'95]**

### Circuit Diagram

- **Charge Sensitive Amplifier**
- **Semi-Gaussian pulse shaper**

### Specification Table

<table>
<thead>
<tr>
<th>Specification</th>
<th>Spec.</th>
<th>unit</th>
<th>Manual</th>
<th>Optimization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Detector capacitance</td>
<td>80</td>
<td>pF</td>
<td>80</td>
<td>80</td>
</tr>
<tr>
<td>Peaking Time</td>
<td>1.5</td>
<td>μs</td>
<td>1.1</td>
<td>1.1</td>
</tr>
<tr>
<td>Counting rate</td>
<td>200</td>
<td>kHz</td>
<td>200</td>
<td>294</td>
</tr>
<tr>
<td>Noise</td>
<td>&lt; 1000</td>
<td>e⁻ RMS</td>
<td>1000</td>
<td>905</td>
</tr>
<tr>
<td>Gain</td>
<td>20</td>
<td>mV/μC</td>
<td>20</td>
<td>21</td>
</tr>
<tr>
<td>Output Voltage range</td>
<td>2</td>
<td>V</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Power consumption</td>
<td>&lt; 40</td>
<td>mW</td>
<td>40</td>
<td>7</td>
</tr>
</tbody>
</table>

*Courtesy Georges Gielen, KUL*
Synthesis: Custom Simulator + Optimizer

- **Basic idea**
  - Build fast, custom simulator *just* for synthesis
  - Simulate *inside* numerical search loop
  - Better accuracy (avoid eqns), more CPU time

- **Issues**
  - Better accuracy, robustness
  - Usually used with stochastic search, like annealing, to avoid many local minima
  - Building a simulator is very hard
  - Usually lacks features regarded as critical in commercial simulators; may *still* need eqns
  - Requires yet more, different input deck info

Examples:

- [Medeiro, ICCAD’94]
- [Ochotta, TCAD’96]
Custom Simulator Example

- ASTRX/OBLX [Ochotta, TCAD96]

**Compilation...**

**...Solution**

<table>
<thead>
<tr>
<th>Specification</th>
<th>Spec:</th>
<th>OBLX / HSPICE</th>
</tr>
</thead>
<tbody>
<tr>
<td>dc Gain (dB)</td>
<td>maximize</td>
<td>73 / 73</td>
</tr>
<tr>
<td>UGF (MHz)</td>
<td>≥50:</td>
<td>50 / 49</td>
</tr>
<tr>
<td>Phase Margin (deg)</td>
<td>≥45:</td>
<td>45 / 45</td>
</tr>
<tr>
<td>PSRR (Vss)</td>
<td>≥40:</td>
<td>93 / 93</td>
</tr>
<tr>
<td>PSRR (Vdd)</td>
<td>≥40:</td>
<td>74 / 74</td>
</tr>
<tr>
<td>Slew Rate (V/µs)</td>
<td>≥50:</td>
<td>50 / 25 &lt;--Example of equation misprediction!</td>
</tr>
<tr>
<td>Area (X1000 sq. µ)</td>
<td>minimize</td>
<td>3132</td>
</tr>
</tbody>
</table>

© R.A. Rutenbar 2001
Synthesis: Commercial Sim + Optimizer

- **Basic idea**
  - Designers are busy people—**don’t** ask them to do extra work to do synthesis
  - Treat the circuit + SPICE deck as the **real IP**
  - Use exact same simulation/verification environment inside numerical optimization
  - Use distributed workstations for CPU cycles

- **Issues**
  - Best accuracy, robustness
  - Relies on clever, vigorous global search: annealing, genetic, pattern search
  - **No equations.** None. Zero.
  - CPU resource intensive

Examples:
- [Phelps, CICC’99]
- [Krasnicki, DAC’99]
- [Phelps, DAC’00]
- [Phelps TCAD’00]
**Example: Industrial Cell from TI**

- **CMU ANACONDA tool** [Phelps CICC99]

- **Folded cascode opamp, high-drive output stage**
  - 33 devs, 2 Rs, 2 Cs; 0.8um CMOS

- **Difficult goals**
  - High drive amplifier, 5Ω load
  - Nominal THD, 0.1%
  - 1kHz, 2.6V p-p input voltage

---

- Overnight on CPU farm
  - 5 runs shown here
  - All specs met
  - All specs fully simulated

- Power (mW)

- TI’s manual design
  - Slightly overdesigned

- Area (1000 sq grids)
Larger Synthesis Example: TI ADSL CODEC

EQF Block: What It Looks Like

- 5 low-noise amps, ~100 passives, 36 program switches, 6 op-modes,
- ~400 devices, flat; ~2-3hrs to SPICE
Synthesis Results: Noise vs Area

- Full sizing/biasing ~10 hours on 20 CPUs; all TI specs met

Max Noise 25-1104KHz @25°C (nV/Hz\(^{1/2}\))

- Smaller & less noise
- Biggest & least noise

Area (1000 square grids)
Synthesis Results: Spectral Mask

Eq0 Passband

Gain (dB)

0 0.4 0.8 1.2 1.6

Freq (MHz)

0 -2 -4 -6 -8

Gain (dB)

0 2 4 6 8 10 12 14 16

Freq (MHz)

Eq0 Spectral Mask

TI Hand

CMU1
CMU2
CMU3
One More Issue: Design Centering

- Cannot ignore this *entirely* in analog synthesis flow
  - Optimization-based attacks can find “bad” corners of design space

**Phase Margin**

Input spec:
- Phase margin > 77° at Vdd = 5.0V

If ignore range / mfg variations, you only get what you *ask* for:
- Phase OK at 5V, but *not* elsewhere

- 2 broad, overall strategies
  - Use first-order heuristics in numerical synthesis, then run centering
  - Combine full statistical optimization in with numerical synthesis
  - Examples: [Mukherjee TCAD’00], [Debyser, ICCAD’98]
Example: Centering Heuristics in Synthesis

- Simple designer-derived constraints in ANACONDA synthesis
  - Require matched devices to be “big”; sensitive devices to be “far enough” into desired region of operation (e.g., 250mV above $V_T$)

Example Monte Carlo spread for a small TI opamp

3σ process, +/-10% supply & temp. variation

Plots show low-frequency gain for manual, auto designs

Hand design

Synthesized design
Cell-Level Analog Layout Synthesis

- Basic task

- Major strategies
  - Enhanced polygon-editing
  - Analog compaction & templates
  - Physical synthesis: full device-level custom place/route

From schematic + geometric constraints to physical layout
Layout: Enhanced Polygon Editing

- **Basic idea**
  - Pushing polygons is **painful**
  - Add nicer editing features to your editor
  - Examples: connectivity-maintenance, device-level layout generators, interactive routing, interactive DRC, etc.
  - Real example: Cadence VirtuosoXL

- **Issues**
  - Good, useful stuff (ie, even beyond analog)
  - Editability enhancements *always* popular in a tool you have to live with for *long* hours
  - Still, not a *radical* productivity win…still really manual layout here, just nicer
Analog Layout: Compaction

- **Basic idea**
  - Draw the layout loose, use compaction to tighten up

- **Issues**
  - Analog is not just about density--also about *precision*
  - Symmetry, align, device internals, etc, *critical*; can’t treat as digital

---

*Courtesy Enrico Malavasi, U.C. Berkeley*
Analog Layout: Templates

- Manually capture regularities as procedures for high-use cells
  - Can mix device generators, cell generators, compaction ideas, etc.
  - Still requires significant manual setup & maintenance investment

Courtesy Koen Lampaert, Conexant
Another Template Example: CYCLONE

- Optimizes LC-oscillators from specs to layout [Deranter DAC’00]
  - Simulated annealing in combination with circuit simulations and some equations
  - FEM simulations to characterize inductor coils
  - Automatic template-based generation of VCO layout

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Low resistive sub CMOS 0.35(\mu)m</th>
<th>High resistive sub BiCMOS 0.65(\mu)m</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ls</td>
<td>1.26 nH</td>
<td>2.3 nH</td>
</tr>
<tr>
<td>Rs</td>
<td>6.5 (?)</td>
<td>5.2 (?)</td>
</tr>
<tr>
<td>Rad, W, Turns</td>
<td>109 (?)m, 40(?)m, 2</td>
<td>141 (?)m, 5(?)m, 2</td>
</tr>
<tr>
<td>Power</td>
<td>32 mW</td>
<td>8.2 mW</td>
</tr>
</tbody>
</table>
Analog Layout: Physical Synthesis

- Basic tasks

1. From sized schematic
2. Design proper cell footprint
3. Design individual device geometries
4. Place/route devices, optimize area, coupling, etc.
Analog-Specific Optimizations: Place/Route

- Placement symmetric and diffusion merging
  - No symmetry
  - No merging
  - Symmetry
  - No Merging
  - Symmetry
  - Merging

- Routing: differential symmetric and coupling avoidance
  - Wiring task with Obstacle
  - Symmetry
  - No crosstalk

[Cohn, JSSC91]
Analog-Specific Optimizations: Merging

- Optimal construction of diff-merged FET groups
  - Example: merging with analog symmetry [Basaran DAC96]
Analog-Specific Optimizations: Wells

- Example: dynamic optimization of wells/latchup during place
University Layout Synthesis Example

KU Leuven LAYLA tool, [Lampaert, Kluwer99]

Courtesy Georges Gielen, K.U. Leuven
Industrial Layout Synthesis Example

Proprietary CMOS comparator auto-layout;
Neolinear NeoCell™ analog layout tool

Courtesy Neolinear
IP = Capture + Front-to-Back-Synthesis

- Commercial example from Neolinear *NeoCircuit/NeoCell* flow

Unsized commercial diff-amp cell → 0.6um proprietary CMOS fab → Circuit Synthesis → Physical Synthesis
IP = Capture + Front-to-Back-Synthesis

- Commercial example from Neolinear *NeoCircuit/NeoCell* flow

- Unsized commercial diff-amp cell
- 0.6um proprietary CMOS fab
- TSMC 0.35um CMOS fab
- Circuit Synthesis
- Physical Synthesis

78% less area; 42% less power
**Analog Cell Ckt/Layout Synthesis: Analysis**

- **PRO**
  - Good idea--getting more “real” with very recent work
  - Supports more dynamic libraries, handles flexibility and variability requirements of custom analog in more natural way
  - Removes many problems with hard IP (layout) bound to one fab
  - Trades time/quality: good designs for most common cases; same trade-offs as for ASIC-style design

- **CON**
  - Very recent, research-oriented tools and flows
  - Until recently only available from universities; in the last 24 months, some startup activity
Last Point: Different Design Discipline

- Synthesis: requires of users more *clarity of intention*
  - Digital folks have already figured this out for cell-based synthesis
  - Analog folks will need to run up the same learning curve
  - CAD tools still can’t read designers minds (yet)

Example: constraint capture/editing
Wrong...

Just like that, but better...
What’s Left to Do: System-Level Design

OK, you design/buy/synthesize all your cells...then what? **Chip-level assembly.** (...and, problems don’t get easier)

<table>
<thead>
<tr>
<th>IP/REUSE</th>
<th>hard</th>
<th>firm</th>
<th>soft</th>
</tr>
</thead>
<tbody>
<tr>
<td>device</td>
<td>Libraries of difficult, exotic device layouts</td>
<td>Parametric device layout generators</td>
<td>--</td>
</tr>
<tr>
<td>cell</td>
<td>Libs of generic cell layouts for specific fab</td>
<td>Parametric templates for schematic, layout</td>
<td>Analog ckt synthesis and layout synthesis</td>
</tr>
<tr>
<td>core</td>
<td>Libs of useful block layouts for specific fab</td>
<td>Parametric templates for useful cores</td>
<td>Mixed-signal system synthesis</td>
</tr>
</tbody>
</table>
"When Bad Things Happen to Good Cells"

- **Noise upsets on delicate/precise analog**
  - From noisy digital wires nearby
  - From noisy shared substrate
  - From noisy power grid

- **Thermal issues**
  - Large digital blocks switching, or large analog devices: heat
  - Temperature changes can affect precision analog

- **Solutions**
  - Segregate (away from digital)
  - Isolate, shield (from noise)
One Assembly Example: IBM Data Channel

- Digital switching is the source of (almost) all evil for analog

Measurements from IBM disk data channel;
Substrate noise spec 4mV -- exceeded

Courtesy Bob Stanisic/Tim Schmerbeck, IBM
CAD Solution: Power Grid Synthesis

- Auto power grid synthesis
  - Re-synthesized IBM grid
  - Power grid routed, sized
  - Power IOs assigned
  - Substrate contacts configured
  - Decoupling caps added

Dynamic Noise (mV)

Static IR Drop (mV)

[Stanisic JSSC 94]
Conclusions

- Analog cells are not like digital cells, viz CAD & methodology
  - Not as easily library-able; can't build one “complete” library
  - Tightly bound to fab process, have difficult precision requirements

- Design strategies
  - Device-level IP: many people use libraries or generators here
  - Cell-level design: templates (designer-initiative), synthesis (tool-based) are workable. Synthesis increasingly real, commercial.

- IP/Reuse strategies
  - Hard IP is often hard to use; even more true for analog
  - Emerging cores for common interface functions, targeting major foundries, hide much of the unpleasantness here; very new business
Closing Observation: What We Really Want

Practical analog synthesis / IP / reuse
Select References

### General Analog CAD Survey

### IP Issues
- http://www.vsia.com -- Virtual Socket Interface Alliance working on specs for interchange of analog IP
Select References

Analog Synthesis

Analog Synthesis, cont.

Select References

Symbolic Analysis

Select References

analog layout

Select References

Analog Layout

