Design for Leading-Edged Mixed-Signal ICs

Rob A. Rutenbar
Carnegie Mellon University
Pittsburgh, PA, USA
rutenbar@ece.cmu.edu
http://www.ece.cmu.edu/~rutenbar
Be Honest: I Say “Analog”... You Think This
Modern Systems Have Analog Interfaces

- Telecom
- Automotive
- Consumer
- Medical

Analog Interface

Digital Computing Core

© R.A. Rutenbar 2001
Lots of Digital “Support” Functions Are Analog

- Some obvious, some not

Wireless connectivity is analog

Modem frontend also analog

Clock synch is an analog problem

IO pads use analog to control signal shape

Physical LAN layer (Ether, Firewire, ..) is all analog

RF Front-end

MODEM

Clock PLL

LAN Interface
Lots of “Digital” Signals—Aren’t

■ Ex: What the bits really look like read off a magnetic disk

The bits are the bumps on these sine waves, by the way…

1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1
Result: An Increasing Design Problem

Commercial Mixed Signal ASIC

% Effort

Digital

Analog
Why This Happens

Analog Methodology
⇒ CAD tools
⇒ Abstraction
⇒ Reuse & IP

Digital Methodology
✓ CAD tools
✓ Abstraction
✓ Reuse & IP

© R.A. Rutenbar 2001
Why This Matters

![Bar chart showing SoC Design Starts from 1997 to 2001. The chart compares SoC Designs and Mixed Signal categories.]

Source: BT Alex Brown Research
Outline

- Quick tour of mixed-signal System-on-Chip (SoC) design
- Design problems & strategies for analog building blocks
- Design problems & strategies for mixed-signal chips
- Talk emphasis
  - We do all this analog design by hand, as painful full custom, today
  - That has got to change—too many opportunities, too few designers
  - What are the prospects for “buy it” or “reuse it” for analog?
  - This is the hot topic in analog today: analog intellectual property
We do all this analog design by hand, as painful full custom, today
That had got to change—too many opportunities, too few designers
What are the prospects for “buy it” or “reuse it” for analog?
This is the hot topic in analog today: analog intellectual property
CMOS Scaling: Different Impact on Analog

- Central fact of life for digital: ICs get smaller, denser, faster

- Scaling matters for analog too; but it’s **different**
Analog: The Eggshell Model

- Analog circuits don’t get a lot *bigger* with scaling
  - Analogy credited to Paul Gray of Berkeley
  - Scaling provides more *opportunities* for analog interfaces
  - 10K-20K analog devices/chip is common

![Diagram of eggshell model with digital core in the middle and more egg and shell on the outer layer.]
What “More Mixed-Signal SoCs” Means

- Larger fraction of SoCs need some analog interfaces

Yesterday

Today

Tomorrow
Example: Automotive Mixed-Signal ASIC
Example: Automotive Mixed-Signal ASIC
Example: Alcatel ISDN Chip

DSP

CPU Core

Logic

Analog Frontend

Memory

Courtesy Frank Op’t Eynde, Alcatel

© R.A. Rutenbar 2001
Example: Alcatel GSM Cellular Chipset ‘96

Courtesy Frank Op’t Eynde, Alcatel
Example: Alcatel GSM Cellular Chipset ‘98

Diagram showing the components of a GSM cellular chipset:
- RF Chip
- LNA
- PA
- VCO
- MAC Layer
- Digital Chip
- MicroProcessor
- RAM
- ROM
- Battery Manager
- Codec
Alcatel GSM Frontend Chip

Courtesy Frank Op’t Eynde, Alcatel
Alcatel GSM Power Manager Chip

[Diagram showing various components such as LNA, PA, VCO, MAC layer, DIGITAL CHIP, CODEC, MicroProcessor, RAM, ROM, Battery Manager.]

Courtesy Frank Op’t Eynde, Alcatel

© R.A. Rutenbar 2001
Example: Alcatel GSM Telecom Chipset ‘00

- Natural result of scaling is analog integration: fewer chips
Outline

- Quick tour of mixed-signal System-on-Chip (SoC) design
- Design problems & strategies for analog building blocks
- Design problems & strategies for mixed-signal chips
- Talk “spin”
  - We do all this analog design by hand, as painful full custom, today
  - That had got to change—too many opportunities, too few designers
  - What are the prospects for “buy it” or “reuse it” for analog?
  - This is the hot topic in analog today: **analog intellectual property**
Example of a Basic Building Block (or Cell)

Mixed-Signal System-on-Chip

Example:
one analog cell on analog-side of a mixed-signal ASIC

© R.A. Rutenbar 2001
Just What Is An “Analog Building Block?”

Typical analog cell
- ~5-100 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

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

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 $\Delta \Sigma$ A/D
  - Regulator
  - CODEC
  - I/O Line Drivers
  - Etc...
No matter how you do it, you have to do these tasks

- Basic device-level circuit design

- 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.
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?”
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

Digital HDL → Logic Synthesis → Tech Mapping → Physical Design

Gate-Level Cell Library
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*
  - Almost *no* reuse

- 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
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 (eg, poly-poly capacitor)
  - Digital processes do not characterize devices for analog uses, eg, models do not capture subthreshold ops, matching, etc

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

Precision structures may be hard/impossible to build if special layers are unavailable (eg, poly-poly capacitor)

Digital processes do not characterize devices for analog uses, eg, models do not capture subthreshold ops, matching, etc
Analog cells manipulate precise electrical quantities
- Depend on precise physical parameters, precise device geometry
- Scale or migrate: process changes, so must redo circuit and layout
- Retarget circuit function: specs change (even a little), must redo ckt
Note: Feature Size Is Scaling…

Min Wire Widths

0.5um  0.35u  0.25u  0.18u
Note: ... Electrical Interface Specs May Not

Example: currents in critical wires affects min allowed width

<table>
<thead>
<tr>
<th>Min Wire Widths</th>
<th>0.5um</th>
<th>0.35u</th>
<th>0.25u</th>
<th>0.18u</th>
</tr>
</thead>
<tbody>
<tr>
<td>Min Wire Width to Carry Current ~ 2.5mA</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

~1 um / mA of current, independent of feature size
Major Impact: Analog is Less Library-able

- **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

```
Logic functions

X

Fanin & fanout variants

X

Timing, latch/FF, scan variants

X

Drive strength (1X, 2X, 4X, 8X) variants

= ~1k-2k cells
```
Analog Cell Libraries: Dimensionality

Problem: many continuous specs for analog cells

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
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?

Design: focuses at 3 levels
- Devices, cells, cores

IP/reuse: focuses on 3 strategies
- Hard, firm, soft IP strategies
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 assembly</td>
</tr>
</tbody>
</table>

Focus is on layout reuse
Focus is on reusable circuit & layout templates
Focus is on synthesis, from spec to ckt to layout
What are people *most commonly* doing right now?

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

### IP/REUSE

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

Digital FET

Analog FET
Device-Level IP: Limited Porosity

Also, precision devices almost never allow wires over the top, to minimize potential coupling.

This whole object is *blocked for upper metals.*
Large Can Mean Very Large, Too

A small CPU core

A few capacitors

1 FET
Example: Analog Precision Tricks for Devices

- Consider a resistor which uses a resistive poly layer

Low-precision R, poly snake resistor

Resistive material

Metal-strapped pins

High-precision R, poly bars with all-metal interconnect

High-precision R, add dummy bars at ends, well and guard ring

Interdigitated pair of precise-ratio 2:1 resistors
Industrial Ex: Precision Interdig Resistor Array

Courtesy Neolinear
Next, Look at Hard Analog IP

Question: how much can you reuse complete layouts?

<table>
<thead>
<tr>
<th>IP/REUSE</th>
<th>hard</th>
<th>firm</th>
<th>soft</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>device</strong></td>
<td>Libraries of difficult, exotic device layouts</td>
<td>Parametric device layout generators</td>
<td>--</td>
</tr>
<tr>
<td><strong>cell</strong></td>
<td><strong>Libs of generic cell layouts for specific fab</strong></td>
<td>Parametric templates for schematic, layout</td>
<td>Analog ckt synthesis and layout synthesis</td>
</tr>
<tr>
<td><strong>core</strong></td>
<td><strong>Libs of useful block layouts for specific fab</strong></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 will 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
Focus Now on Design & Synthesis

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>
<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>
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
How do people put big ASICs together today?
- In big pieces, compiling & synthesizing the chunks as needed

I need 75,000 gates of random logic:  
*Use logic synthesis followed by physical synthesis*

I need an Embedded SRAM:  
*Use a RAM generator tool*

I need a Regular Datapath:  
*Use a Datapath compiler*

I need a Register File:  
*Use a RegFile compiler*
We want the same sort of functionality

- Synthesis: for the very custom cells that determine analog performance
- Templates: for the less custom, more regular stuff left over

**Mixed-SignalASIC**

- I need a custom Voltage Reference
  *Use analog circuit & physical synthesis*

- I need a custom Video Amplifier
  *Use analog circuit & physical synthesis*

- I need a set of custom High-Precision Passives
  *Use a Device generator*

- I need a custom A/D Converter
  *Use a mix of template compilers and custom analog synthesis*
Template Example: CMOS Analog Cells

- 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
Template Example: RF Components

- 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
  - Auto template-based generation of VCO layout

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Low resistive sub CMOS 0.35 μm</th>
<th>High resistive sub BiCMOS 0.65 μm</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 Ω</td>
<td>5.2 Ω</td>
</tr>
<tr>
<td>Rad, W, Turns</td>
<td>109 μm, 40 μm, 2</td>
<td>141 μm, 5 μm, 2</td>
</tr>
<tr>
<td>Power</td>
<td>32 mW</td>
<td>8.2 mW</td>
</tr>
</tbody>
</table>

Courtesy Georges Gielen, K. U. Leuven

© R.A. Rutenbar 2001
Template Example: RF Components

- RF mixer, circuit & layout optimized together, [Gielen ICCAD01]

One quarter of the mixer core

Complete mixer floorplan, with quarter-piece above highlighted

An optimized layout

One quarter of the mixer core

Complete mixer floorplan, with quarter-piece above highlighted

Third order intermodulation

Output Amplitude (dBm)

Input Amplitude (dBm)

1IP3 extraction

Courtesy Georges Gielen, K. U. Leuven
More General Attack: Analog Synthesis

- **Basic idea**
  - **Circuit synthesis:** transform cell spec into sized/biased schematic
  - **Layout synthesis:** 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
Most approaches have this overall structure.

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
20 Years of Synthesis Distilled Onto 1 Slide…

(1) Scripting
Optimization Engine
You
Evaluation Engine
Eqns you write:
\[ I = \frac{K'}{2} \frac{W}{L} (V_{gs}-V_{t})^2 \]

(2) Equation-Based
Optimization Engine
Numerical optimizer
Evaluation Engine
Eqns you write:
\[ I = \frac{K'}{2} \frac{W}{L} (V_{gs}-V_{t})^2 \]

(3) Symbolic analysis
Optimization Engine
Numerical optimizer
Evaluation Engine
Auto-derived eqns
\[ I = \frac{K'}{2} \frac{W}{L} (V_{gs}-V_{t})^2 \]

(4) Simulation-Based
Optimization Engine
Global optimizer
Evaluation Engine
Industrial simulator
Eqns you write:
\[ I = \frac{K'}{2} \frac{W}{L} (V_{gs}-V_{t})^2 \]
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>Specification</th>
<th>Spec</th>
<th>GP</th>
<th>SPICE</th>
<th>Measured</th>
</tr>
</thead>
<tbody>
<tr>
<td>Power (μW)</td>
<td>≤ 2</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>
<td>71</td>
</tr>
<tr>
<td>UGBW (MHz)</td>
<td>Max.</td>
<td>19</td>
<td>19</td>
<td>17</td>
</tr>
<tr>
<td>Phase margin (°)</td>
<td>60</td>
<td>63</td>
<td>65</td>
<td>58</td>
</tr>
<tr>
<td>Slew rate (V/μs)</td>
<td>≤ 30</td>
<td>35</td>
<td>33</td>
<td>33</td>
</tr>
<tr>
<td>Noise, 1kHz ($\frac{V}{\sqrt{Hz}}$)</td>
<td>≤ 400</td>
<td>393</td>
<td>390</td>
<td>-</td>
</tr>
<tr>
<td>Area (μm²)</td>
<td>≤ 10k</td>
<td>4.8k</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Optimal trade-off curves

Courtesy Mar Hershenson, Stanford
Symbolic Analysis Example

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

\[ AV_0 = \frac{g_{m, M_2}}{g_{m, M_1}} \frac{g_{m, M_4}}{g_{o, M_4}} \left( \frac{g_{o, M_4} g_{o, M_5}}{g_{m, M_5} + g_{mb, M_5}} + \frac{G_a + g_{o, M_9} + g_{o, Q2}}{\beta_{Q2}} \right) \]

Courtesy Georges Gielen, KUL
Simulation Based Example: Cells from TI

- Done using CMU ANACONDA tool
- 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

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

Power (mW)

Area (1000 sq grids)

© R.A. Rutenbar 2001
Large Sim-Based Example: TI ADSL CODEC


**EQF**

- Equalizer: 1.54MHz, corner 0dB gain
- Analog Low-Pass Filter: 0-25dB/MHz gain, in 5dB/MHz steps
- Programmable Gain Amplifier: 2.5-11.5 dB gain, in 0.25dB steps
- Analog to Digital Converter: 4416KHz 14bits
- Digital Low-Pass Filter: 1.1MHz corner, 0dB gain
- Decimation: Input fc 4416KHz, Output fc 2208KHz
EQF: What It Does

- EQF = equalizer + 4th-order elliptical low-pass C-T filter
  - Programmably amplifies signal (since attenuated by copper)
  - Filters data from spectrum (avoiding phone voice band)

Spectral Mask
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
CMU 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)

© R.A. Rutenbar 2001
Synthesis Results: Spectral Mask

- ~ 2 months designed manually
- Synthesized automatically overnight
One More Messy Issue: Design Centering

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

![](chart.png)

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

- **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

$3\sigma$ process, +/-10% supply & temp. variation

Plots show low-frequency gain for manual, auto designs
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
Analog-Specific Optimizations: Place/Route

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

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

[Cohn, JSSC91]
Small Physical Synthesis Example: Close-up

- Commercial tools emerging
  - Neolinear’s NeoCell

- This example
  - CMOS
  - ~50 devices
  - Layout < 1 hr

Courtesy Neolinear
© R.A. Rutenbar 2001
Large Physical Synthesis Example

- Proprietary CMOS comparator auto-layout from NeoCell

Courtesy Neolinear
Subsystem Example: Cells + Glue Circuits
Historically—Why has this been so Hard?

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

- With new synthesis/analysis tools, improved methodologies, & improved *attitudes* about design—stage set for radical changes

© R.A. Rutenbar 2001
New Idea: Analog IP = Capture + Synthesis

Commercial example from Neolinear NeoCircuit/NeoCell flow

Unsized commercial diff-amp cell

0.6um proprietary CMOS fab

Circuit Synthesis

Physical Synthesis

TSMC 0.35um CMOS fab

Circuit Synthesis

Physical Synthesis

78% less area; 42% less power
Outline

- Quick tour of mixed-signal System-on-Chip (SoC) design
- Design problems & strategies for analog building blocks
- Design problems & strategies for mixed-signal chips

Talk emphasis
- We do all this analog design by hand, as painful full custom, today
- That has got to change—too many opportunities, too few designers
- What are the prospects for “buy it” or “reuse it” for analog?
- This is the hot topic in analog today: analog intellectual property
What’s Left to Do: Chip-Level Design

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

<table>
<thead>
<tr>
<th>IP/REUSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>hard</td>
</tr>
<tr>
<td><strong>device</strong></td>
</tr>
<tr>
<td><strong>cell</strong></td>
</tr>
<tr>
<td><strong>core</strong></td>
</tr>
</tbody>
</table>
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

<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 cell 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 assembly</td>
</tr>
</tbody>
</table>

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

*Hide* low-level analog; basic cells hand-crafted to exploit foundry process
Template-Based System Layout Example

- Analogy: just like digital datapath generators
  - Can exploit analog regularities you know; procedurally generate

14-bit 150-Ms/s 0.5µm CMOS DAC

[ISSCC’99]

Mixed-Signal SoC Revisited…

- We want block-level IP & assembly for both digital and analog
  - Synthesis: for the very custom, performance-sensitive circuits
  - Templates: for the less custom, more regular stuff left over

**I need a custom Video Amplifier**
*Use analog circuit & physical synthesis*

**I need 75,000 gates of random logic:**
*Use logic synthesis followed by physical synthesis*

**I need a set of custom High-Precision Passives**
*Use a Device generator*

**I need an Embedded SRAM:**
*Use a RAM generator tool*

**I need a custom A/D Converter**
*Mix of templates and custom analog synthesis*
Example: Dual-Tone Multi-Frequency Decoder

Analog
PLL Clock
Authored with Neolinear and Cadence

RAM (256 x 16)
Authored with Artisan RAM Compiler

RAM (128 x 16)

Glue Logic

DSP Core

I/O pads

Results Converter (FFT)
Authored with Cadence

ROM (512 x 16A)
Authored with Artisan ROM Compiler

Courtesy Artisan, Cadence and Neolinear

© R.A. Rutenbar 2001
Pushing Inside the PLL

- All analog done via custom synthesis on this design

- Counter (3-bit)
  Authored with Cadence

- Voltage-Controlled Oscillator
  Authored with NeoCell

- Divider (2-bit)
  Authored with Cadence

- Phase Detector
  Authored with Cadence

- Charge Pump
  Authored with NeoCell

- Top block & support logic assembled using Cadence

Cadence ® Generic PDK
0.18 6LM Generic Process

Courtesy Cadence and Neolinear

© R.A. Rutenbar 2001
Next Problem: Mixed-Signal Chip Assembly

- …or, “When Bad Things Happen to Good Cells”
- Noise upsets on delicate/precise analog
  - From noisy digital wires nearby
  - From noisy shared substrate and 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)
Noise At Mixed-Signal Chip Level

- Coupled through supply rails and common substrate
  - Precise analog biasing easily vulnerable to voltage upset

![Diagram showing analog and digital components coupled through supply rails and common substrate]

© R.A. Rutenbar 2001
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

VDD
Substrate Gnd
5mV

Courtesy Bob Stanisic/Tim Schmerbeck, IBM
Another Example: TI High-Speed Video DAC

Texas Instruments
High-speed video DAC, ~1994

Measured chip performance at 14.4MHz
ADC codes

Volts
DAC
0
1.4
2.8
4.2
-1.4
30
90
150
210
Time (ns)
Substrate
800mV p-p

Samples
Sparkle codes, errors

Courtesy Texas Instruments
© R.A. Rutenbar 2001
**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)**

- Bus only
- +I/O cells
- +Sub cons
- +Caps

**Static IR Drop (mV)**

- Bus only
- +I/O cells
- +Sub cons
- +Caps

[Stanisic JSSC 94]

© R.A. Rutenbar 2001
Mixed-Signal Chip-Level Assembly Today

- **Embarrassingly ad hoc**
  - Lots of guessing (and lots of praying) about floorplan, global signal routing, block-level isolation structures, etc
  - Often vastly over-conservative; sometimes just plain wrong
  - Often takes a few silicon spins to iron out ("few" may mean 5-10 at RF and higher frequencies)

- **Where the action is**
  - Full-chip and package extraction and simulation for noise coupling
  - Smarter circuit design methodologies for noise immunity (think "echo cancelation", but replace "echo" with "substrate noise"…)

© R.A. Rutenbar 2001
Conclusions

- Analog circuits: here to stay
  - In an SoC world, big systems need to talk to the external world
  - The world is analog (…get used to it); analog does this communication

- Mixed-signal design realities
  - Analog cells != digital cells
  - Not as easily library-able; don’t scale; don’t migrate
  - Tightly bound to fab process, difficult precision requirements
  - Chip level assembly is nasty

- Design strategies
  - Less art, more science: better methodologies, real synthesis tools
  - Analog IP: design for migrating, retargeting is the next big thing
Where all this Analog IP Stuff is Heading

Analog folks want *IP/reuse*, too

**Analog Synthesis**

**Analog IP**

© R.A. Rutenbar 2001
Select References

**General Analog CAD Survey**


**IP Issues**

- http://www.vsia.com -- Virtual Socket Interface Alliance working on specs for interchange of analog IP
Analog Synthesis

Select References

- **Analog Synthesis, cont.**
Select References

- **Symbolic Analysis**
Select References

- **Analog Layout**
Select References

Analog Layout


