AMS Circuit Generation,
It’s in the
BAG

John Crossley, Alberto Puggelli, Hanh-Phuc Le,
Lingkai Kong, Rachel Nancollas, Bonjern Yang, Elad Alon
Plastic Bag Ban in 14 Alameda County Cities

Bags will cost you a dime apiece

By Lisa Fernandez | Tuesday, Jan 1, 2013 | Updated 7:35 AM PST
BAG motivation similar to bag ban motivation
**Analog Circuit Design Reuse**

- **Apple A6**
- **Personal Example - Digital PLL**

- **Need for reuse in industry and academia:** TTM vs. TTG

- **Reuse is hard due to highly manual analog design flows**
  - New process/specs → need to re-design/re-verify

- **What we need for reuse:** Scriptable design flow
  - Old idea that didn’t catch on\(^1\) but complexity is higher now

---

Outline

• BAG basics:
  – What is BAG?
  – How is layout handled?

• BAG details:
  – “Standard” layout styles
  – Case Study: DCDC Regulator

• Conclusions
Berkeley Analog Generator (BAG)

• **Goal:** codify analog designer’s methodology
  - Designer’s output is a generator rather than an instance.
• **Python-based, hierarchical framework**

<table>
<thead>
<tr>
<th>«Interface»</th>
<th>Analog Generator</th>
</tr>
</thead>
<tbody>
<tr>
<td>ReadSpecifications()</td>
<td></td>
</tr>
<tr>
<td>DesignSchematic()</td>
<td></td>
</tr>
<tr>
<td>DesignLayout()</td>
<td></td>
</tr>
<tr>
<td>WritePerformances()</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>«Abstract Class»</th>
</tr>
</thead>
<tbody>
<tr>
<td>BAG Module</td>
</tr>
<tr>
<td>specs: Spec[*]</td>
</tr>
<tr>
<td>perfs: Perf[*]</td>
</tr>
<tr>
<td>ReadTechFile()</td>
</tr>
<tr>
<td>RunOptimizer()</td>
</tr>
<tr>
<td>LaunchSimulations()</td>
</tr>
<tr>
<td>RunDRCandLVS()</td>
</tr>
<tr>
<td>IO_OpenAccess()</td>
</tr>
<tr>
<td>● ● ●</td>
</tr>
</tbody>
</table>

Berkeley	
  Analog	
  Generator	
  (BAG)
BAG Structure Diagram

BAG Developer

«Abstract Class»
BAG Module
specs: Spec[*]
perfs: Perf[*]
ReadTechFile()
RunOptimizer()
LaunchSimulations()
RunDRCandLVS()
IO_OpenAccess()

«Abstract Class»
BAG Custom Digital

«Abstract Class»
BAG Array

Generator Writer

«Class»
BAG SC-DCDC Converter
«Class»
BAG NOR
«Class»
BAG Comparator
«Class»
BAG SRAM
«Class»
BAG Current-DAC
Hierarchical Example: DCDC Regulator

Hierarchical Example:

DCDC Regulator

«Class» BAG Switched-Capacitor DCDC Converter

«Class» BAG Converter Interleaved Phase

«Class» BAG Controller

«Class» BAG Switch

«Class» BAG Capacitor
BAG: Typical Generator Writer’s Flow

Python Code

```
In [3]: execfile('setup.py')
In [2]: cfg = BAG.BagCfg('BAG.cfg')
    log file for virtual run is located here: /tmp/fileks8EiZ
In [3]: from reg.vco import reg_vco
In [4]: rvco = reg.vco(cfg)
```

Sized Schematic

Sim. Results

### Schematic Entry
- Creation of Parametric Schematic
- Creation of Parametric Testbench

### BAG framework
- Template Generation
- Interactive Schematic Exploration
- `<class>`.DesignSchematic() Codification
- Layout Exploration
- `<class>`.DesignLayout() Codification
Outline

• Review:
  – What is BAG?
  – How is layout handled?

• Update:
  – “Standard” layout styles
  – Case Study: DCDC Regulator

• Conclusions
## Layout Styles

- Many cells can be mapped to a handful of “styles”
- Complete (or nearly complete) parameterized layouts generated automatically

### Amplifier Style

<table>
<thead>
<tr>
<th></th>
<th>Output</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>D</td>
<td></td>
<td>D</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Input</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>D</td>
<td></td>
<td>D</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Bias</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>D</td>
<td></td>
<td>D</td>
</tr>
</tbody>
</table>

### Array Style

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>D</td>
<td>D</td>
</tr>
<tr>
<td>D</td>
<td>3</td>
</tr>
<tr>
<td>D</td>
<td>2</td>
</tr>
<tr>
<td>D</td>
<td>3</td>
</tr>
<tr>
<td>D</td>
<td>D</td>
</tr>
</tbody>
</table>

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>D</td>
<td>D</td>
</tr>
<tr>
<td>D</td>
<td>3</td>
</tr>
<tr>
<td>D</td>
<td>1</td>
</tr>
<tr>
<td>D</td>
<td>3</td>
</tr>
<tr>
<td>D</td>
<td>D</td>
</tr>
</tbody>
</table>

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>D</td>
<td>D</td>
</tr>
<tr>
<td>D</td>
<td>0</td>
</tr>
<tr>
<td>D</td>
<td>1</td>
</tr>
<tr>
<td>D</td>
<td>3</td>
</tr>
<tr>
<td>D</td>
<td>D</td>
</tr>
</tbody>
</table>

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>D</td>
<td>D</td>
</tr>
<tr>
<td>D</td>
<td>2</td>
</tr>
<tr>
<td>D</td>
<td>2</td>
</tr>
<tr>
<td>D</td>
<td>3</td>
</tr>
<tr>
<td>D</td>
<td>D</td>
</tr>
</tbody>
</table>

### “Standard” Style

![Diagram of “Standard” Style]
Pycell Based Layout

- Pycells provide more control over implementation
- Assisted pycell generation using layout styles
Outline

• Review:
  – What is BAG?
  – How is layout handled?

• Update:
  – “Standard” family of layout styles
    – Case Study: DCDC Regulator

• Conclusions
- Nearly automated
- Number of fingers and finger width adjusted automatically based on vertical pitch
Standard Cell Style – Analog Cells

- Standard cell style based on digital BUT
- Also useful for analog/mixed circuits
1. Rail width
2. Dummies
3. Channel length
4. Balance PMOS/NMOS
5. Vertical pitch
Standard Row Style

- Combine std. cells using vertical routing channels
- Vertical pitch and pmos/nmos ratio passed to lower level standard cells
Standard Block Style

- Combine groups of standard cells and rows
- Routing channels shared within each row
Standard Block Example:
Control for DCDC Regulator

Subcircuits

Asynchronous comparator  Decoupling capacitors
Flip flops                  Level shifters
Skewed logic gates         Phase splitter
Outline

• Review:
  – What is BAG?
  – How is layout handled?

• Update:
  – “Standard” family of layout styles
  – Case Study: DCDC Regulator

• Conclusions
Switched Cap DCDC Regulator
Top Level Floorplan

- *Manual pycell* used for upper levels of hierarchy
Unit Cell Floorplan

Flying Capacitor
Switches
Control
Flying Capacitor

Flying Capacitor
Switches
Control
Flying Capacitor
Unit Cell Floorplan

- Flying Capacitor
- Control
- Switches
Power Switch Layout

M1 + Driver
M2 + Driver
M3 + Driver
M4 + Driver
M5 Driver
M6 + Driver
M7 + Driver
M8 + Driver
M9 + Driver

C1

C2

Pmos Switch  Nmos Switch  Driver
Conclusion

- BAG framework extended to enable automated layout
  - Pycells are powerful enough for full system layout
  - Standard layout styles ease creation of layout

- AMS circuit generation is in the bag!
  - We have real generators that handle sizing and layout
  - DCDC regulator taping out this month
  - Graduate analog circuit design class currently using BAG
Acknowledgements

• UC Discovery Grant ele07-10283 under the IMPACT program
• NSF Infrastructure Grant No. 0403427.
• BWRC students, faculty and sponsors
• TSMC for the iPDK
• Ciranova/Synopsys for Helix License
• Matt Spencer for terrible jokes