

# Analog & Custom Digital Design: The Big Problems

**Jim Solomon**

*Ciranova, Gemini, AWR, Silicon Navigator, Pyxis, Nascentric*

*Founder SDA / Cadence, Xulu Entertainment*

# Designing Consumer SOCs: Today's Two Biggest Issues\*

- Low power digital
- Analog design

\* *Ken Liou, Director IP and Design Support Division, UMC*

# The Analog Problems: Design Time, IP Reuse, Fab Migration, PDKs

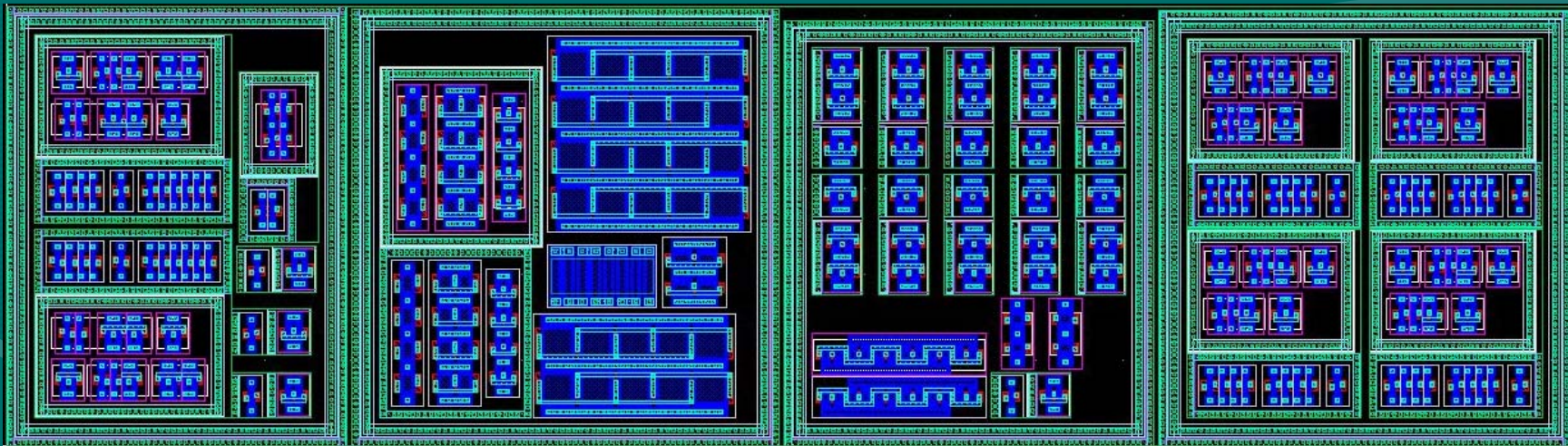
- Analog Design Time
  - Layout done by hand
  - Cannot simulate large MS blocks, especially post-layout
- IP Reuse
  - Largely manual, takes too much time
- Fab Migration
  - Manual brute force is only option
- PDK's
  - Need continual redesign as fabs evolve, all manual

# Improved Analog Design Automation: Is there any hope??

*The industry has been trying to develop better analog design tools for over 20 years, ...*

*Analog is now one of the biggest SOC bottlenecks!!*

# Foundational technology key to solving analog problems: Automated Analog Placement



Phase Detector

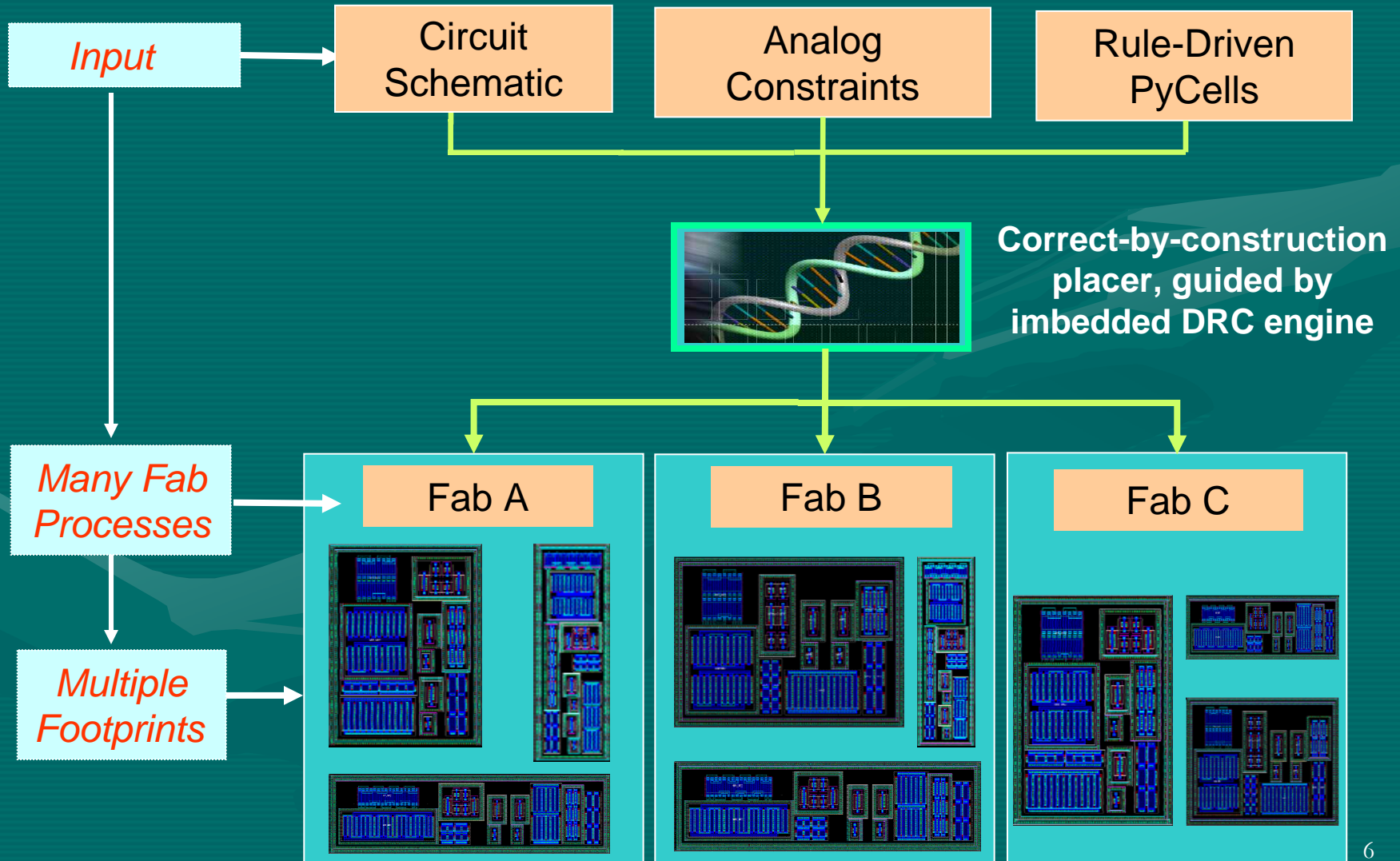
Charge Pump

VCO

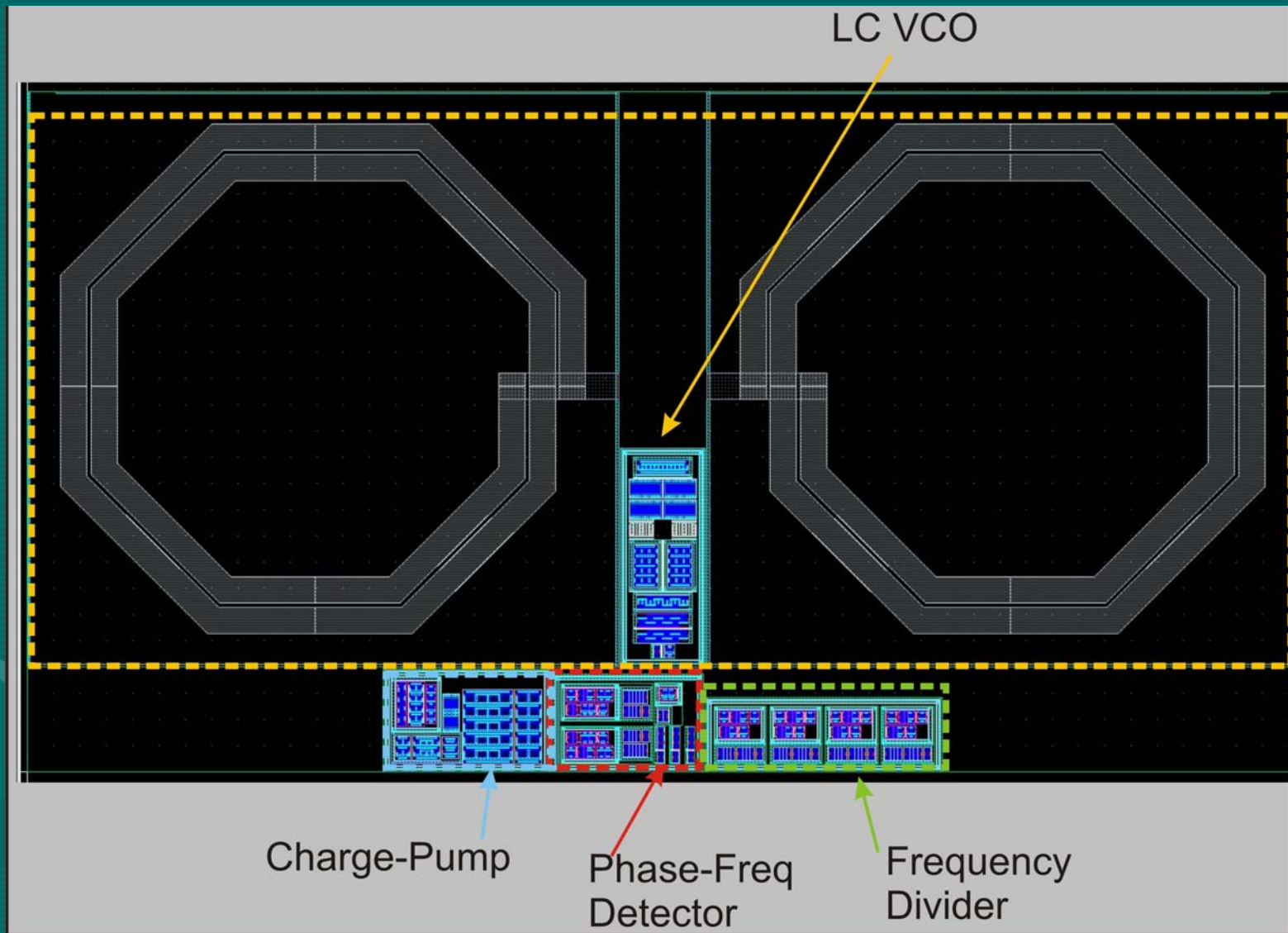
Frequency Divider

*200 device PLL placed in 12 minutes using Ciranova's Helix (4 core PC)*

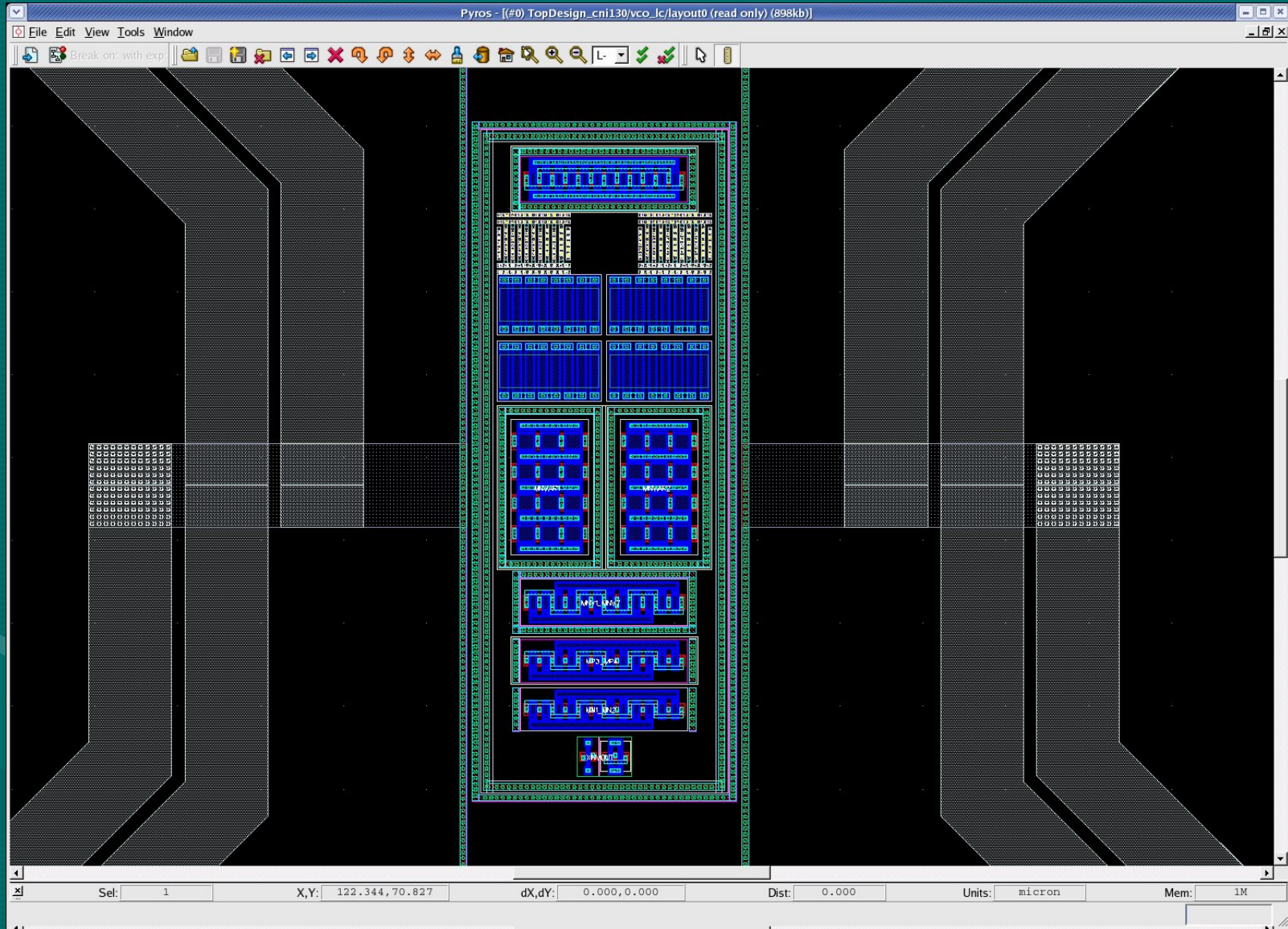
# Helix Placement Flow



# RF LC PLL with spiral PyCell Inductors

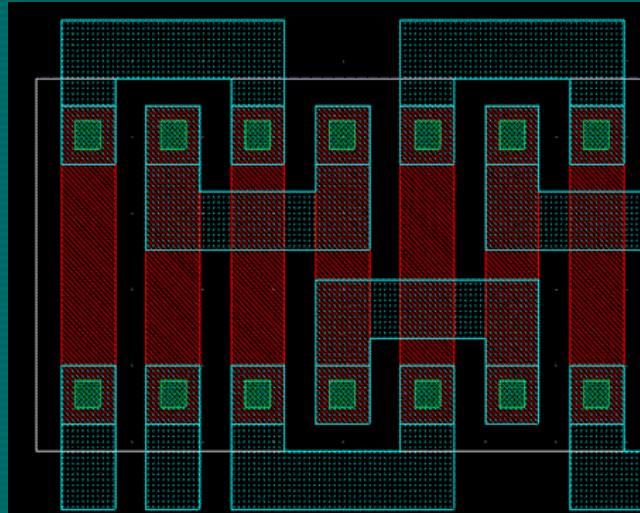


# Zoom-in on LC VCO

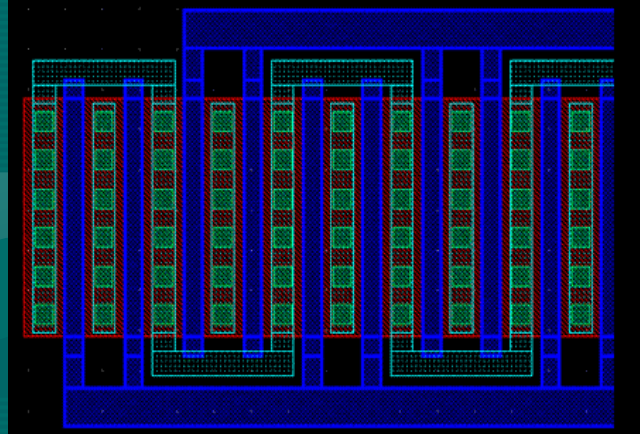
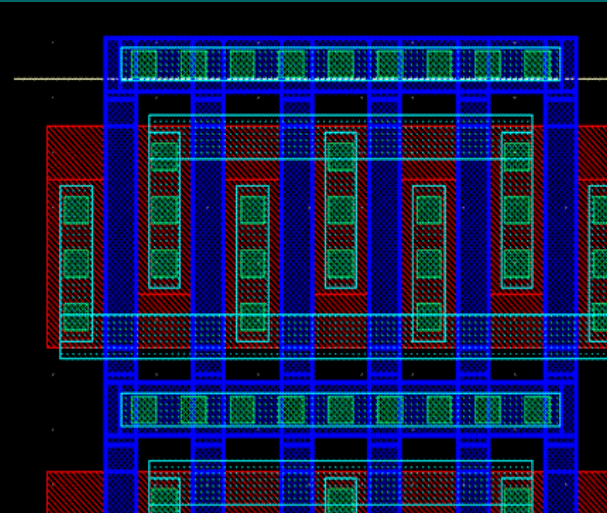


# Some Example PyCells: Helix Builds on Smart PyCells

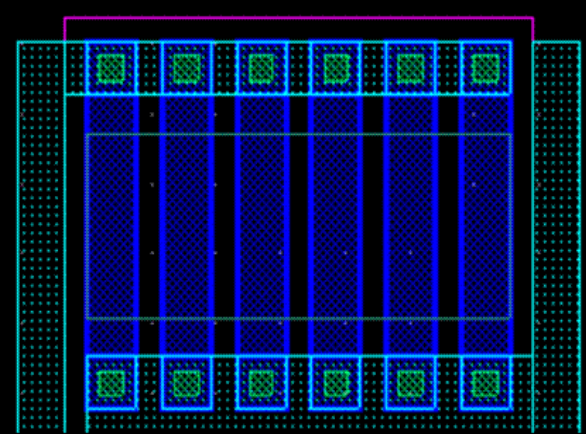
Matched Resistor Pair



Stacked FET



Matched FET Pair



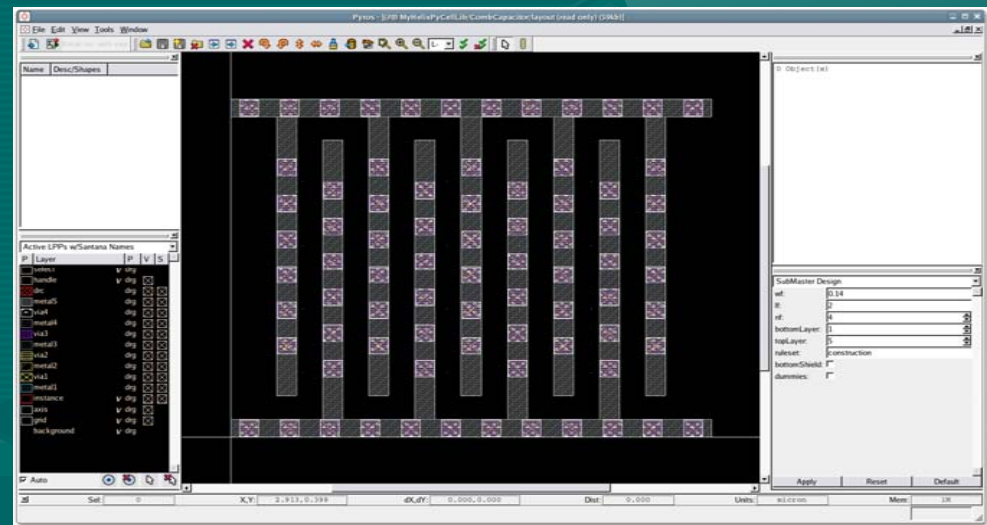
Stacked resistor

# PyCell Code for Comb Capacitor in LC PLL

## (About half of total code needed)

```
def createFingers(self):
    # generate fingers for each comb on each metal layer
    for i in range(len(self.metalLayers)):
        # ignore bottom layer, if it was used to create a shield
        if self.bottomShield and i == 0:
            continue
        layer = self.metalLayers[i]
        # generate fingers for terminal1 comb
        for j in range(self.fingers+1):
            w = self.width
            l = self.length + self.fingerSpacing
            bar1 = Bar(layer, NORTH_SOUTH, 'B1',
                    Point(0,0), Point(w,l))

            # if this is not top layer,
            # create vias
            if layer != self.metalLayers[-1]:
                contact1 = AbutContact(layer,
                    self.metalLayers[i+1],
                    routeDir1=NORTH_SOUTH,
                    routeDir2=NORTH_SOUTH,
                    abutDir=NORTH,
```



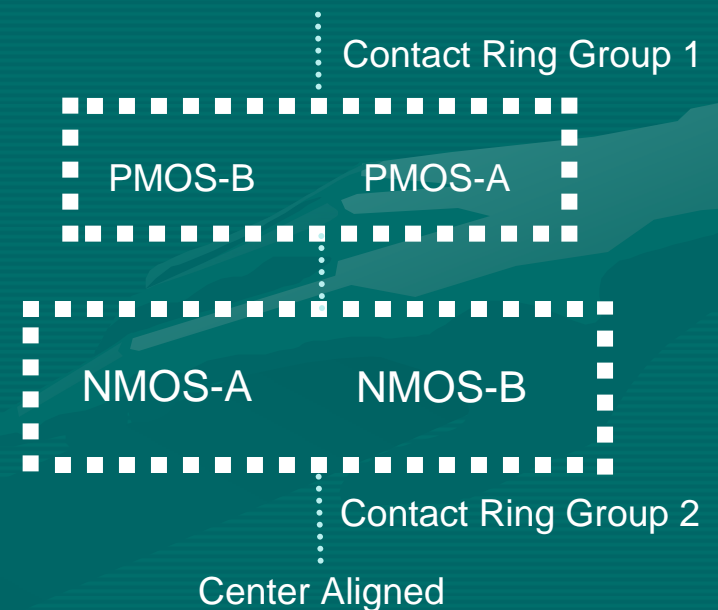
# Example Hierarchical Constraint Files

```
# Block Level Constraints
# Top-level pins and power supplies
Pin Name:vdd Type:PowerSupply Param:top
Pin Name:vddh Type:PowerSupply Param:top
Pin Name:vss Type:PowerSupply Param:bottom
```

```
# Device Level Constraints
# Thick-gate Pmos device
# Place PMOS-B & PMOS-A side-by-side & auto abut
# if possible; Name this constraint group "tr-1"
UCell Name:tr-1 Type:RowUCell
Param:bottom,false,false Contains:PMOS-B,PMOS-A

# Wrap "tr-1" with a Contact-Ring; Call this
# object "CR-1"
UCell Name:CR-1 Type:ContactRingUCell
Param:1,diff,metall,nimp,nwell&od2,vddh,
Contains:tr-1
```

```
# Multi-device Constraints
# Stack Contact-Ring Objects 1 & 2 and center align; Call this object "UC-3"
UCell Name:UC-3 Type:ColumnUCell Param:center,false,false
Contains:CR-1,CR-2
```



# Some Features of this new Technology

- Simple hierarchical analog constraint files
  - Past efforts failed largely because constraint files were too complex
- Code Size for PyCells much smaller than for Skill
- PyCells & placement are design rule correct-by construction
- Multiple layout footprints & device placements
- Connectivity-driven/routing-aware
- Automatic well and diffusion merging & sharing

*An analog constrained router is needed next*

# Summary: Design Time, IP Reuse, Fab Migration, PDK's

- Auto placement & fast transistor simulation enable simulation of extracted post-layout designs
  - Dramatically reduce mask spins, shorten design time, cut costs
- Reuse analog IP
  - Automatically retarget IP blocks for new designs or fabs, cut design time
- Fab migration
  - Modify designs by just reading in new process rules files
- PDK automation
  - PDK's based on PyCells allow automatic regeneration for new rules/fabs

*New Standards Efforts: IPL Initiative & Analog Constraint Initiative*

END