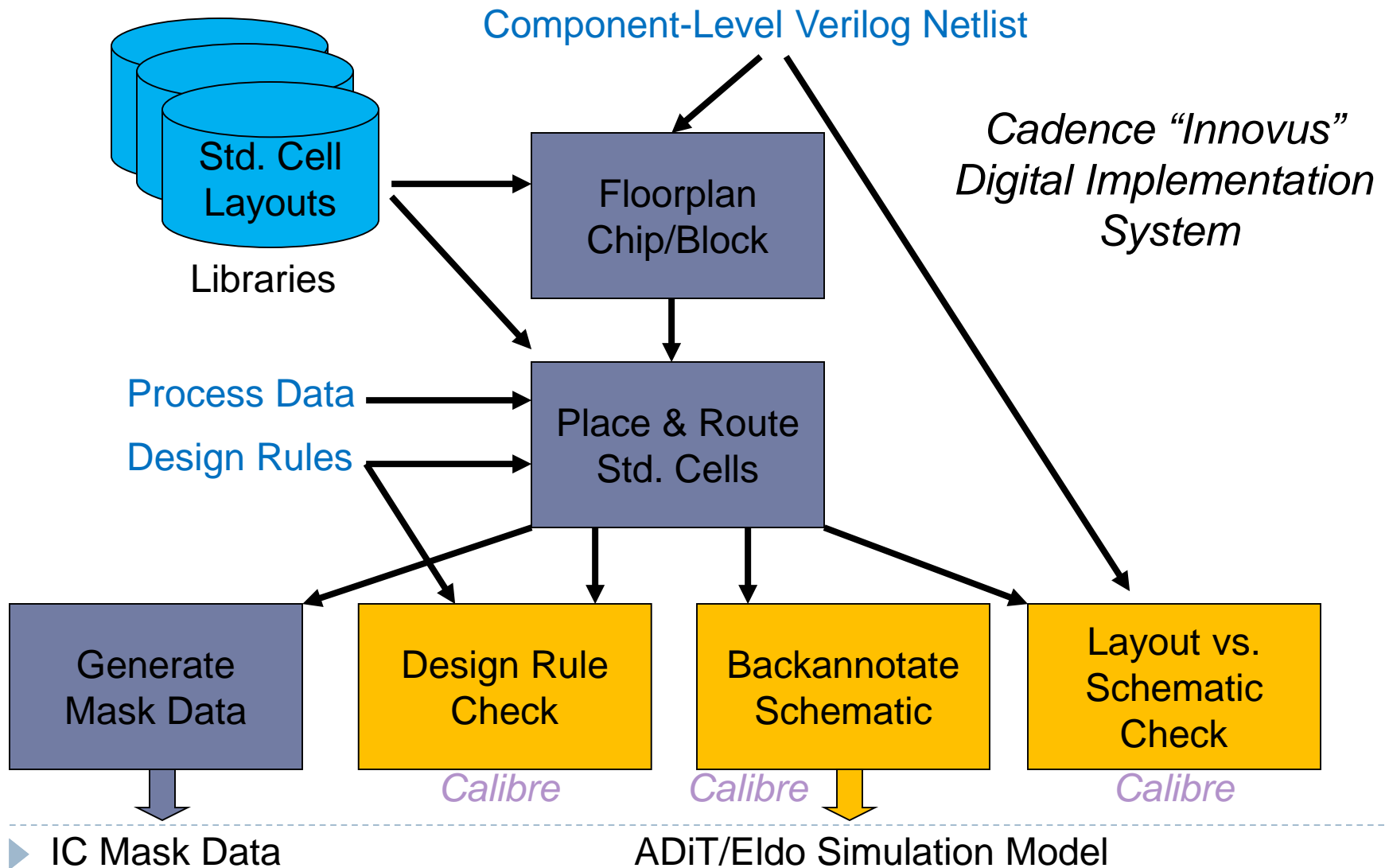


# ASIC Physical Design Standard-Cell Design Flow

Using the Cadence Innovus Digital Implementation System

# ASIC Physical Design (Standard Cell)

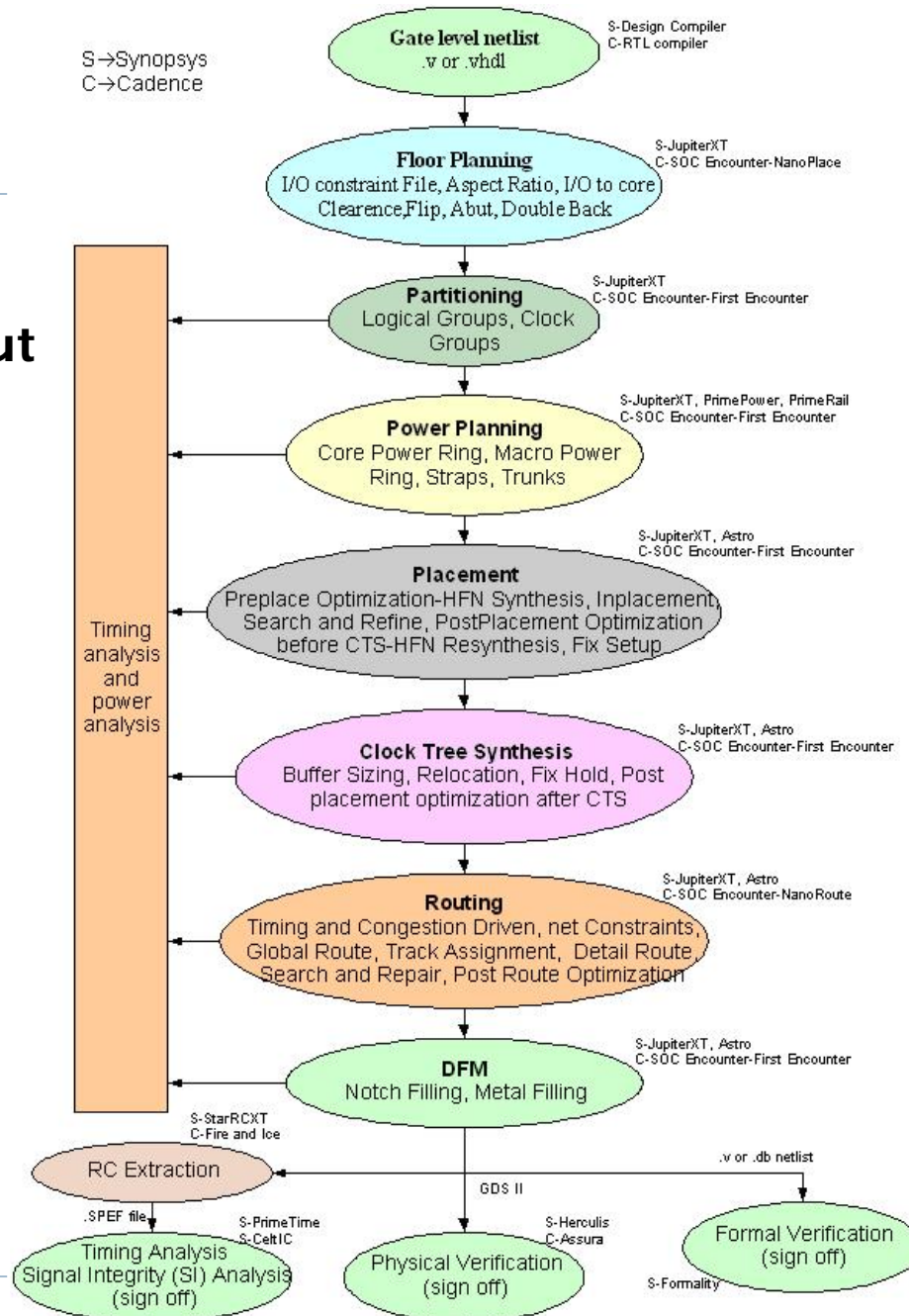
(can also do full custom layout)



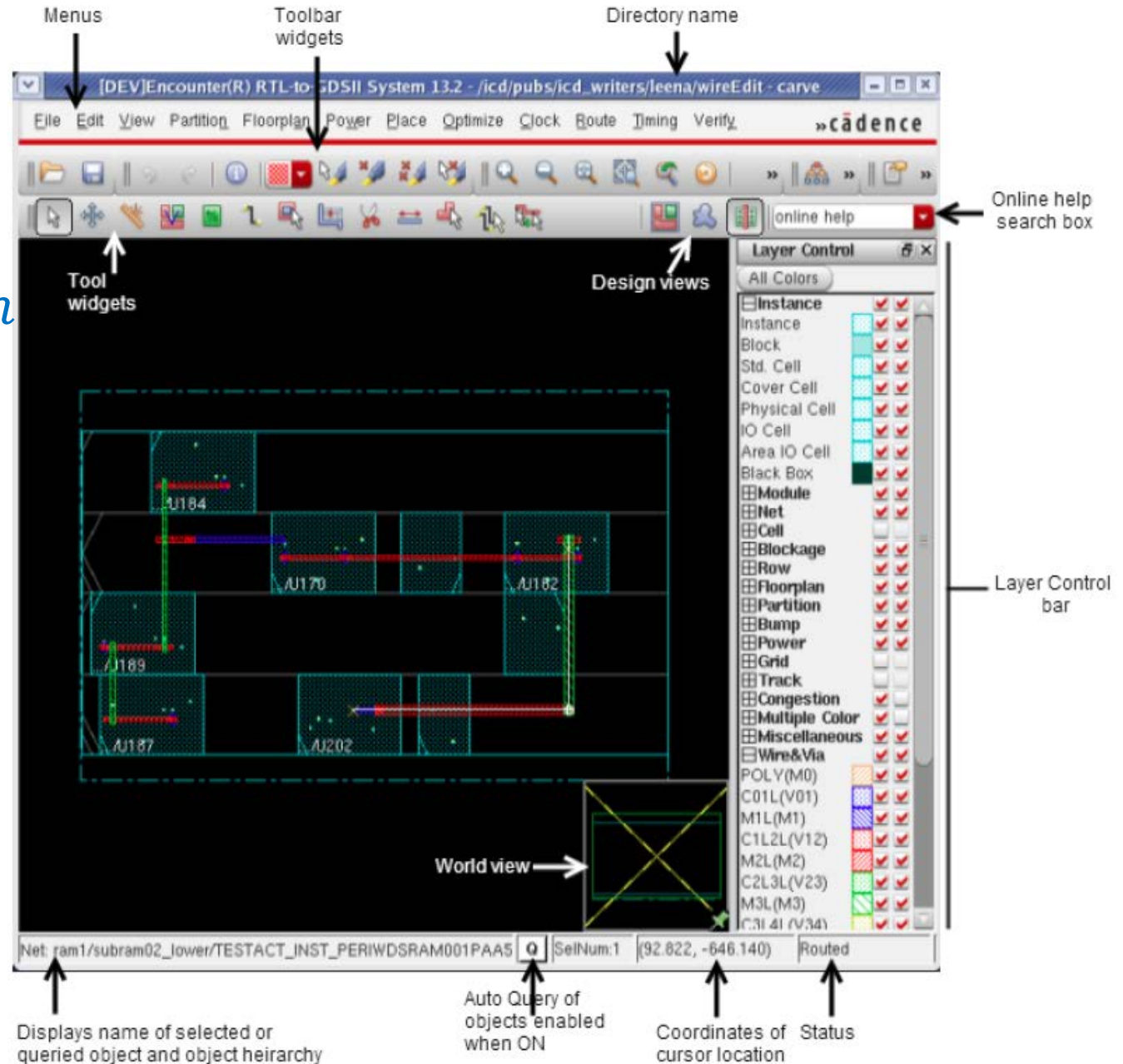
# Netlist-to-layout design flow

Synopsys  
"JupiterXT"

Cadence  
"SOC Innovus"

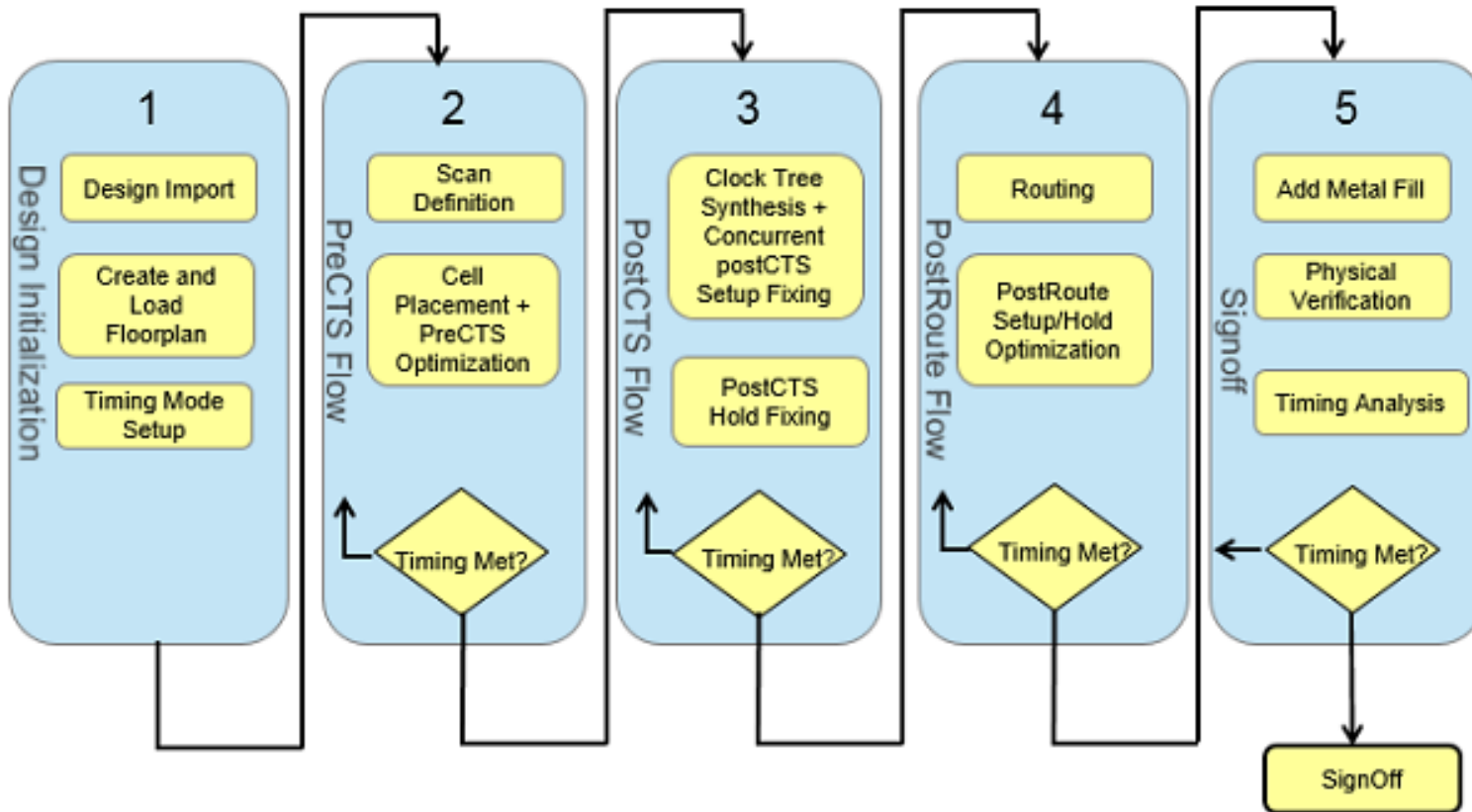


*Innovus  
Digital  
Implementation  
(EDI)  
System GUI*



# EDI design flow

## Floorplan ("flat") through implementation



# Design Import (specify input files)

**File > Design Import**

Gate-level netlist  
Verilog file(s)

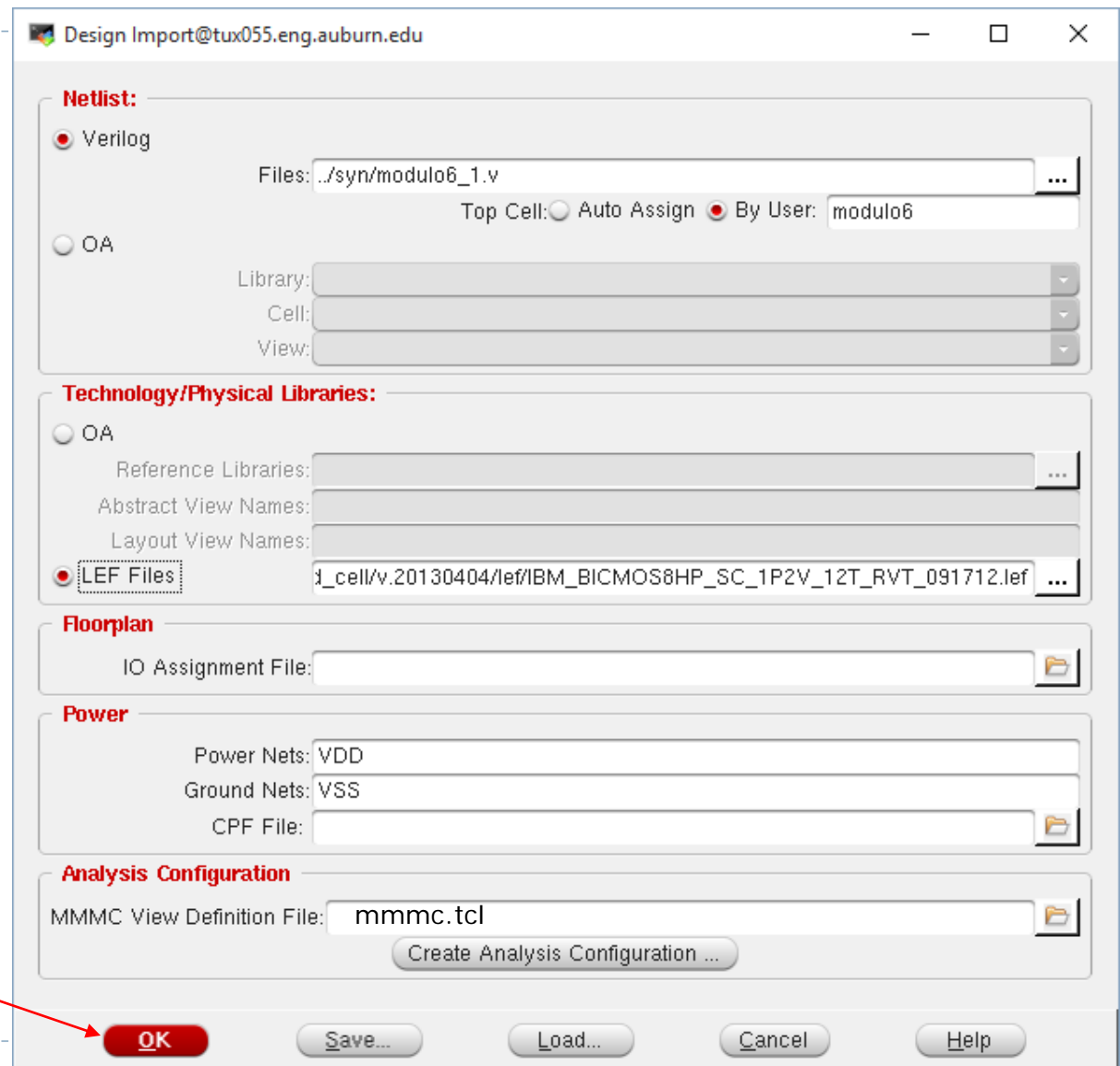
Physical data (LEF)  
Technology Cells

IO pin planning

Power planning

Timing data and  
constraints

Executes `init_design`  
command to load data.



# Netlist files

---

- ▶ Verilog gate-level netlist(s)

- ▶ Gates from the standard cell library
- ▶ Design can be hierarchical or flat
- ▶ Tcl commands:

```
set design_netlisttype verilog
```

```
set init_verilog [list file1.v file2.v]
```

```
set init_design_set_top 1 ← 0 to auto-assign top cell
```

```
set init_top_cell "top" ← specify if above = 1
```

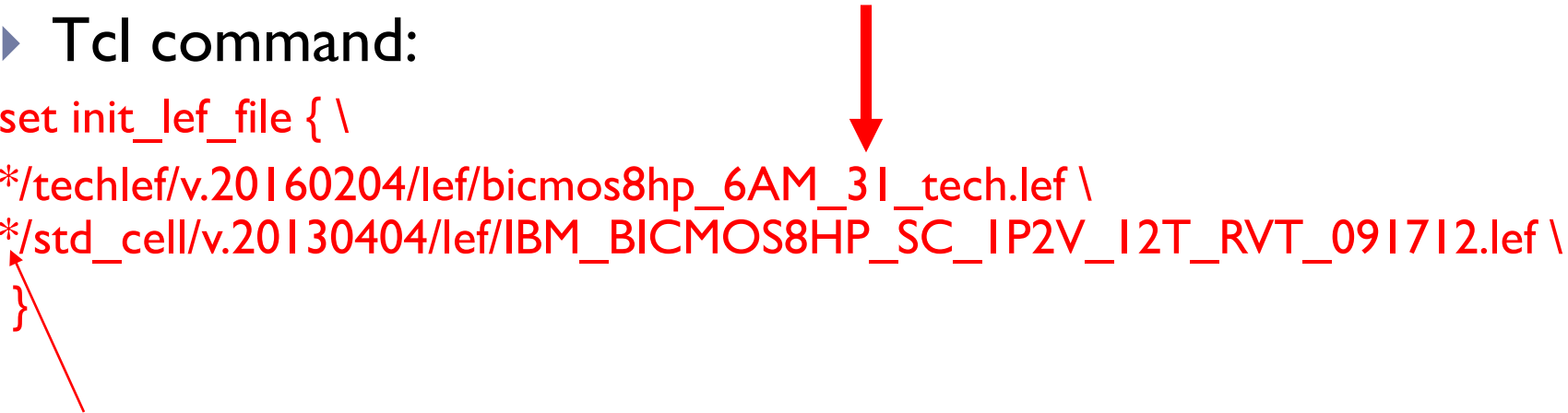


# Physical/Technology Library

---

- ▶ Libraries in LEF (Library Exchange Format)
- ▶ Technology Library
  - ▶ Technology-specific characterizations of metal layers, vias, etc.
- ▶ Standard Cell Library
  - ▶ Abstract view of each cell (box, pins, obstructions)
  - ▶ Includes metal layers for pins (**read tech. library first!**)
- ▶ Tcl command:

```
set init_lef_file { \  
*/techlef/v.20160204/lef/bicmos8hp_6AM_3I_tech.lef \  
*/std_cell/v.20130404/lef/IBM_BICMOS8HP_SC_IP2V_I2T_RVT_091712.lef \  
}
```



For \* insert /class/ELEC6250/cmos8hp

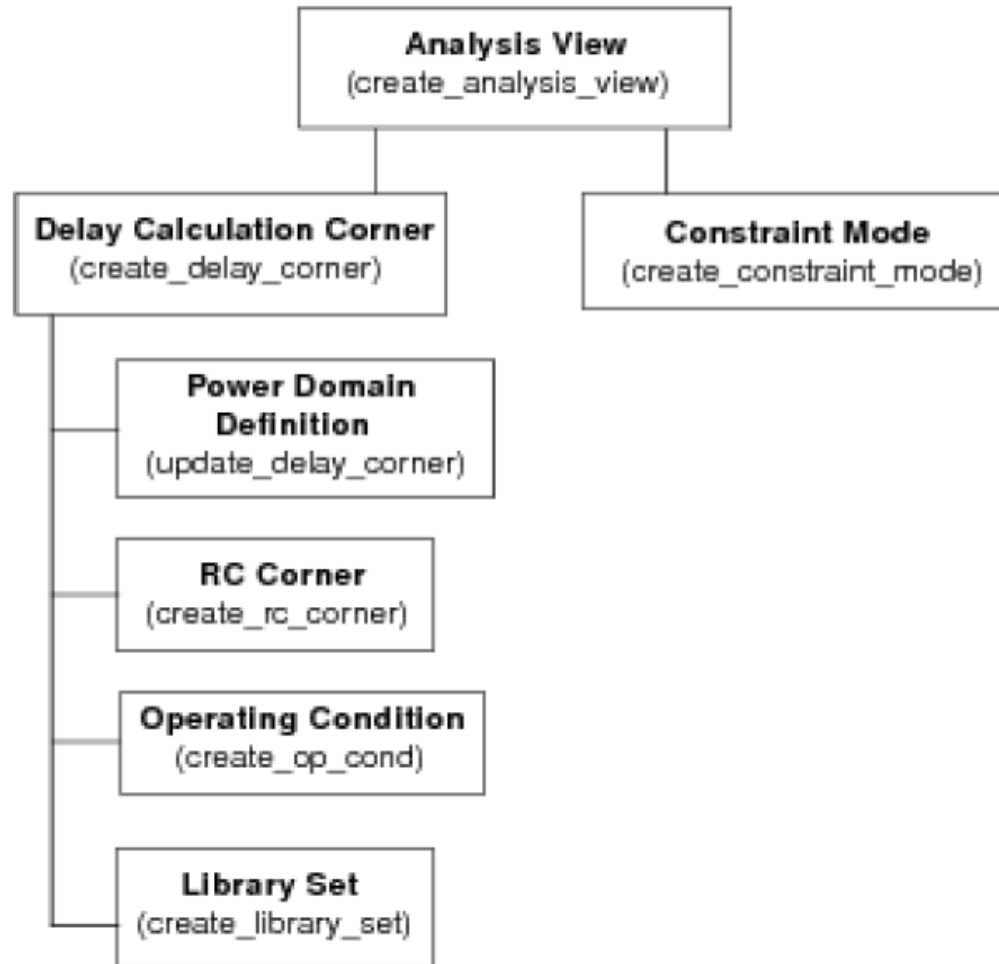
---





# Setting up MMMC analysis

---



## # Multi-Mode/Multi-Corner (MMMM) Analysis Setup

### # Configure 1-corner single-model MMMC

---

# Timing constraints file from synthesis

```
create_constraint_mode -name CONSTRAINTS -sdc_files { ../syn/modulo6_1.sdc}
```

# Create operating condition (P-V-T) for the timing library

```
create_op_cond -name OPcondition \  
-library_file
```

```
{/class/ELEC6250/cmos8hp/std_cell/v.20130404/synopsys/typ_v120_t025/PnomV1p20T025  
_STD_CELL_8HP_12T.lib} \  
-P {1} -V {1.2} -T {25}
```

# Use typical timing library file for this design

```
create_library_set -name TYPLib \  
-timing
```

```
{/class/ELEC6250/cmos8hp/std_cell/v.20130404/synopsys/typ_v120_t025/PnomV1p20T025  
_STD_CELL_8HP_12T.lib}
```

# Create RC corner from capacitance table(s)

```
create_rc_corner -name RCcorner \  
-cap_table
```

```
/class/ELEC6250/IBM_PDK/BiCMOS8HP_Fire_Ice/bicmos8hp_cadence_20160215/cadence/v.  
20160215/captable/bicmos8hp_6AM_31_nm.CapTbl \  
-T {25}
```



## # Multi-Mode/Multi-Corner (MMMC) Analysis Setup

### # Configure 1-corner single-model MMMC

---

# Delay corner = timing library plus rc corner

# Worst-case corner = max delay/affects **setup** times

# Best-case corner = min delay/affects **hold** times

# For 1-corner use typical values for both

```
create_delay_corner -name DELAYcorner \
                    -library_set TYPLib \
                    -rc_corner RCcorner
```

# Analysis view = delay corner matched to constraints

```
create_analysis_view -name TYPview \
                    -delay_corner {DELAYcorner} \
                    -constraint_mode {CONSTRAINTS}
```

# Set analysis view to above for both setup and hold

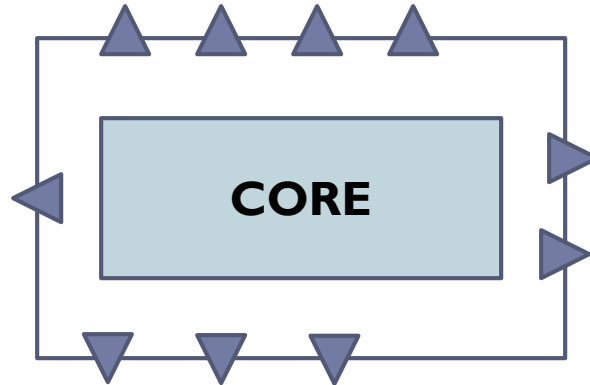
```
set_analysis_view -setup {TYPview} \
                    -hold {TYPview}
```



# Floorplan I/O assignment file

---

- ▶ Specify placement of I/O pins on the “IO box”



- ▶ Read pin placement from file via Tcl command:  
`set init_io_file {modulo6.io}`
- ▶ Placement can be adjusted via **Pin Place** tool or `editPin` command
- ▶ **File format on next slide**



# IO assignment file format

---

(globals

```
version = 3
io_order = clockwise
total_edge = 4
space = 2
```

place pins in this order  
4 edges on the IO box  
global spacing of 2um between pins

)

(iopin

start pin definitions  
pins on left side

```
(left
)
(top
```

pins on top side

```
(pin      name = "I[0]"
          layer = 3
          width = 0.5
          depth = 0.6
          skip = 2
          place_status = fixed
```

pin name  
metal layer for connecting wire  
pin dimensions

```
)
```

skip 2 positions to get away from corner

```
(pin      name = "I[1]"
          layer = 3
          width = 0.5
          depth = 0.6
          place_status = fixed
```

```
)
```

Continue for other pins, including right and bottom sides

---



# Power planning

---

- ▶ Specify power/ground net name(s)

- ▶ Tcl commands

`set init_pwr_net {VDD}`

VDD net name(s)

`set init_gnd_net {VSS}`

GND net name(s)

- ▶ CPF (Common Power Format) file is *optional*

- ▶ Can be used for low-power design and timing

- ▶ Useful for multiple power domains required

- ▶ TCL command:

`set init_cpf_file {modulo6.cpf}`



# Analysis Configuration

---

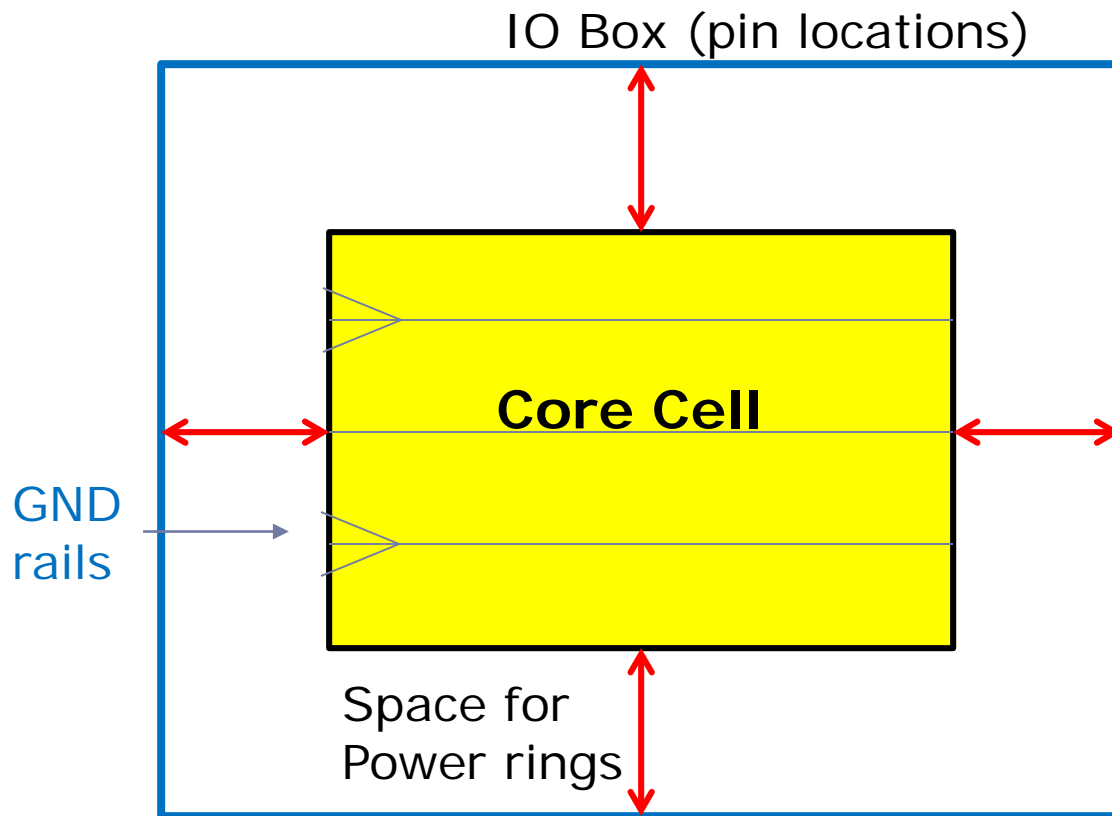
- ▶ **MMMC View Definition File**
  - ▶ Multi-Mode/Multi-Corner analysis
  - ▶ Specify timing libraries for process “corners”
    - ▶ Worst case and best case timing (min/max delays, etc.)
  - ▶ Used to meet timing constraints and calculate delays
- ▶ If MMMC info not provided, physical design only
- ▶ Tcl command:  

```
set init_mmmc_file {modulo6.tcl}
```
- ▶ MMMC to be discussed later



# Floorplanning a standard cell block

(assume no hand-placed blocks)

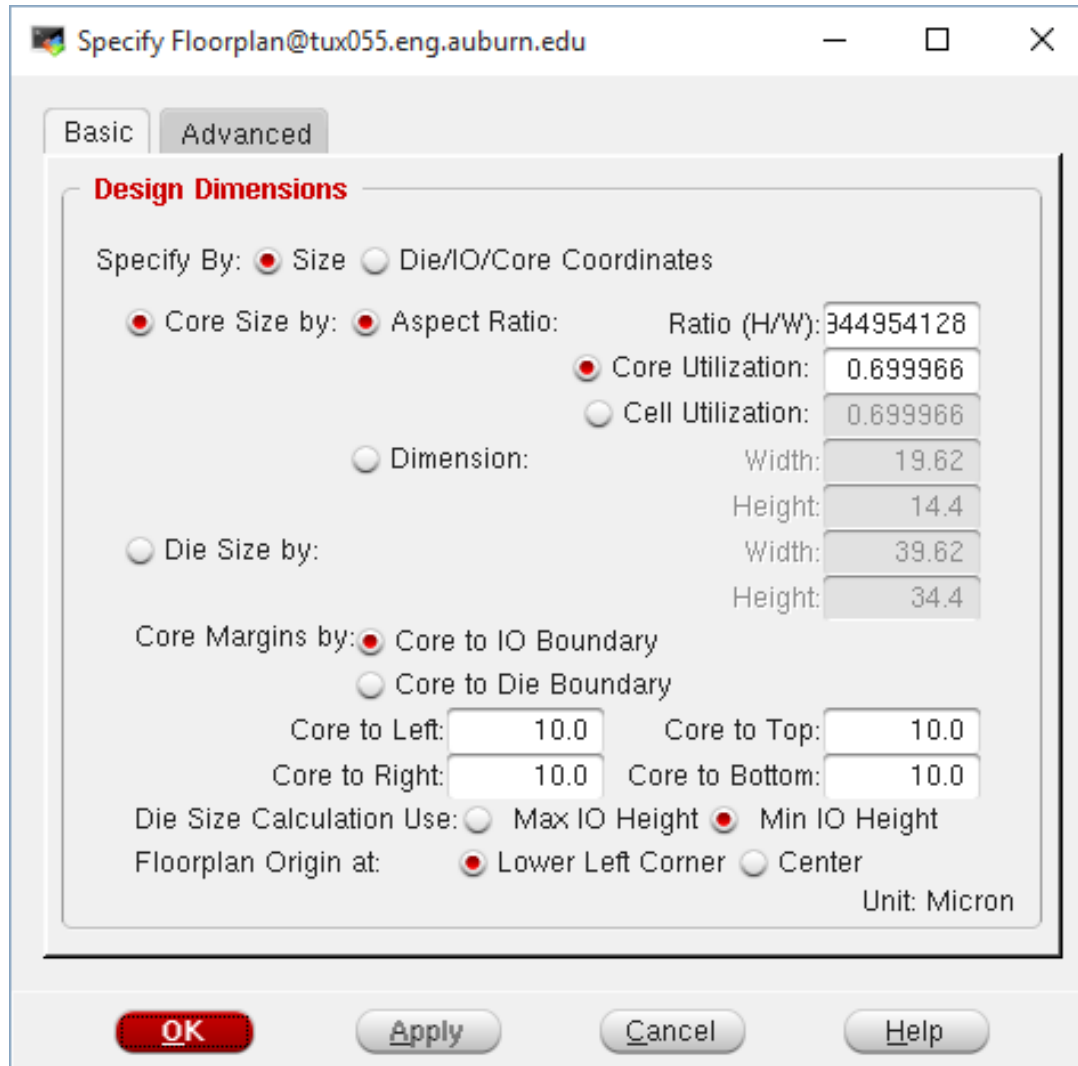


Chip floorplan has modules and I/O pads



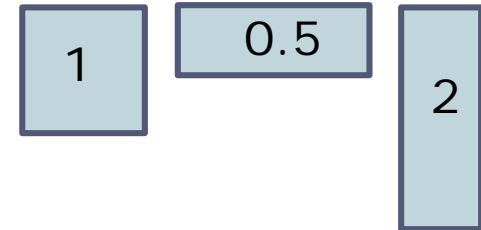


# Specify floorplan



Specify by size  
or by coordinates

Core size "aspect ratio"



Core utilization %  
leaves space for routing

Core to IO boundary  
leaves space for  
power rings

# Floorplan Tcl Command

---

- Initiate floorplanning and generate tracks

```
setDrawView fplan          - display floorplan view
setFPlanRowSpacingAndType $rowgap 1  — 1 every row
floorplan -r 0.8 0.7 20 20 20 20  — 2 every other row
                                left bottom right top
                                Core-to-IO spacing
```

Core to IO      Aspect Ratio (H/W)      Density

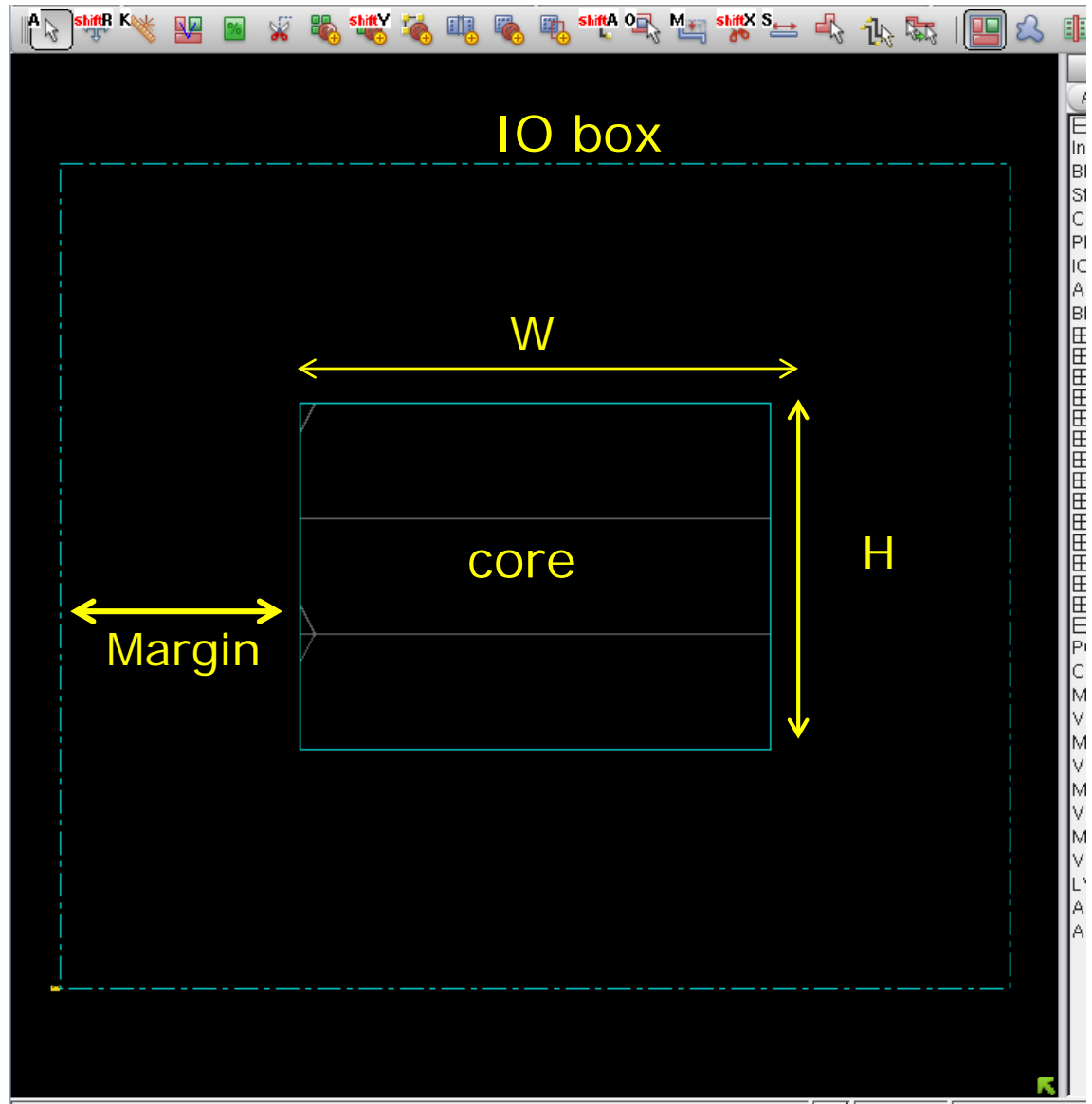
- Can also specify core and/or die & IO pad dimensions
- Defaults: IO pins vs Pads,  
1<sup>st</sup> cell row flip from bottom up



# Floorplan for Modulo6

Aspect = 1  
(3 cell rows)

Core-to-IO  
margins = 20

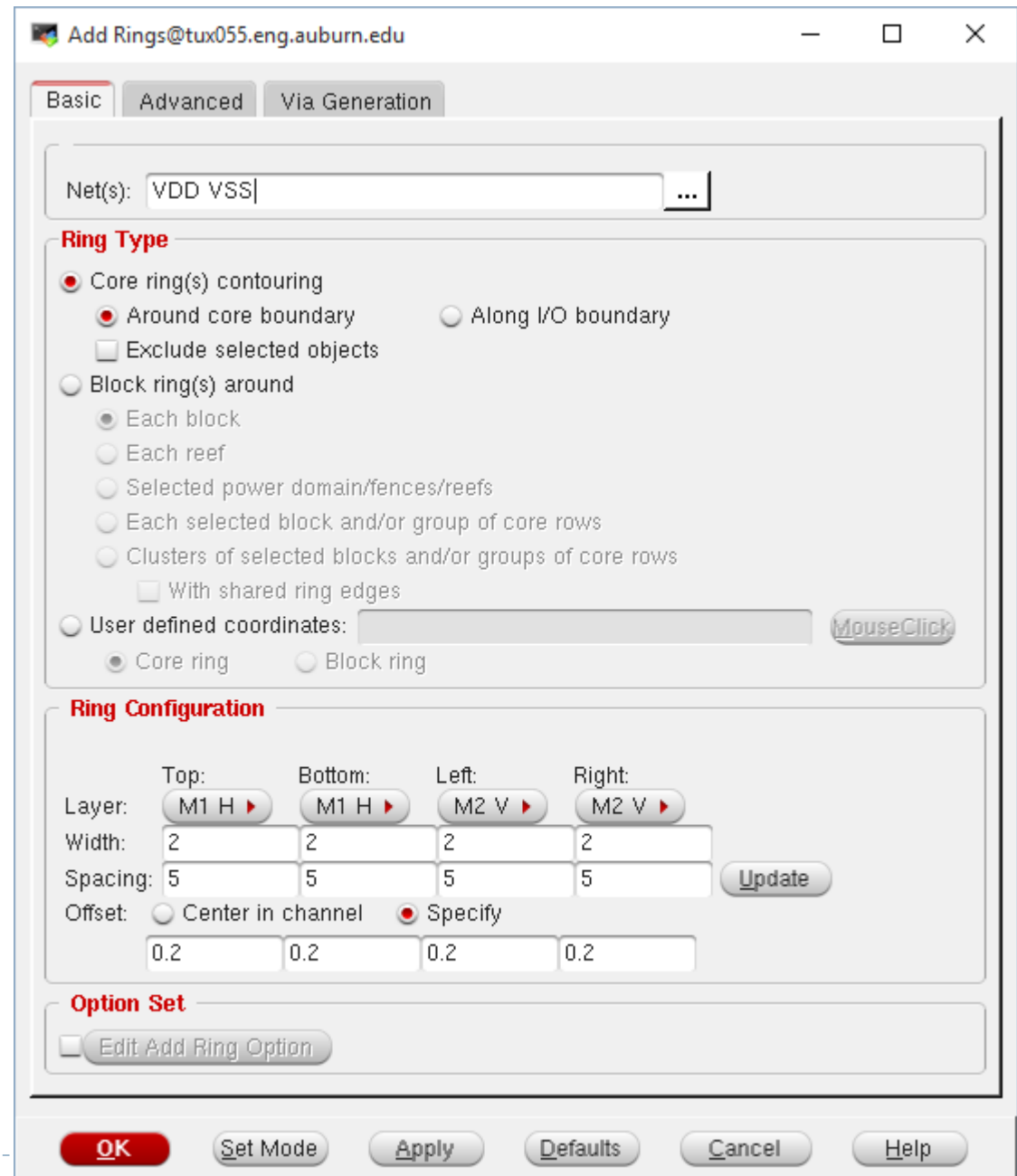


# Power Planning: Add Power Rings

Around core  
or I/O box

For each side:

- Metal layer
- Metal width
- Spacing between wires
- Offset from boundary or center in channel



# Power Planning

---

- ▶ Specify configuration of power rings

```
setAddRingMode -stacked_via_top_layer M3
```

```
-stacked_via_bottom_layer M1
```

```
addRing -nets {VDD VSS} \
```

```
-type core_rings \
```

```
-around user_defined \
```

```
-center 0 \
```

```
-spacing $pspace \
```

```
-width $pwidth \
```

```
-offset $poffset \
```

```
-threshold auto \
```

```
-layer {bottom M1 top M1 right M2 left M2 }
```

Metal wire layers

} Around core boundary  
— 1 to center rings in channel



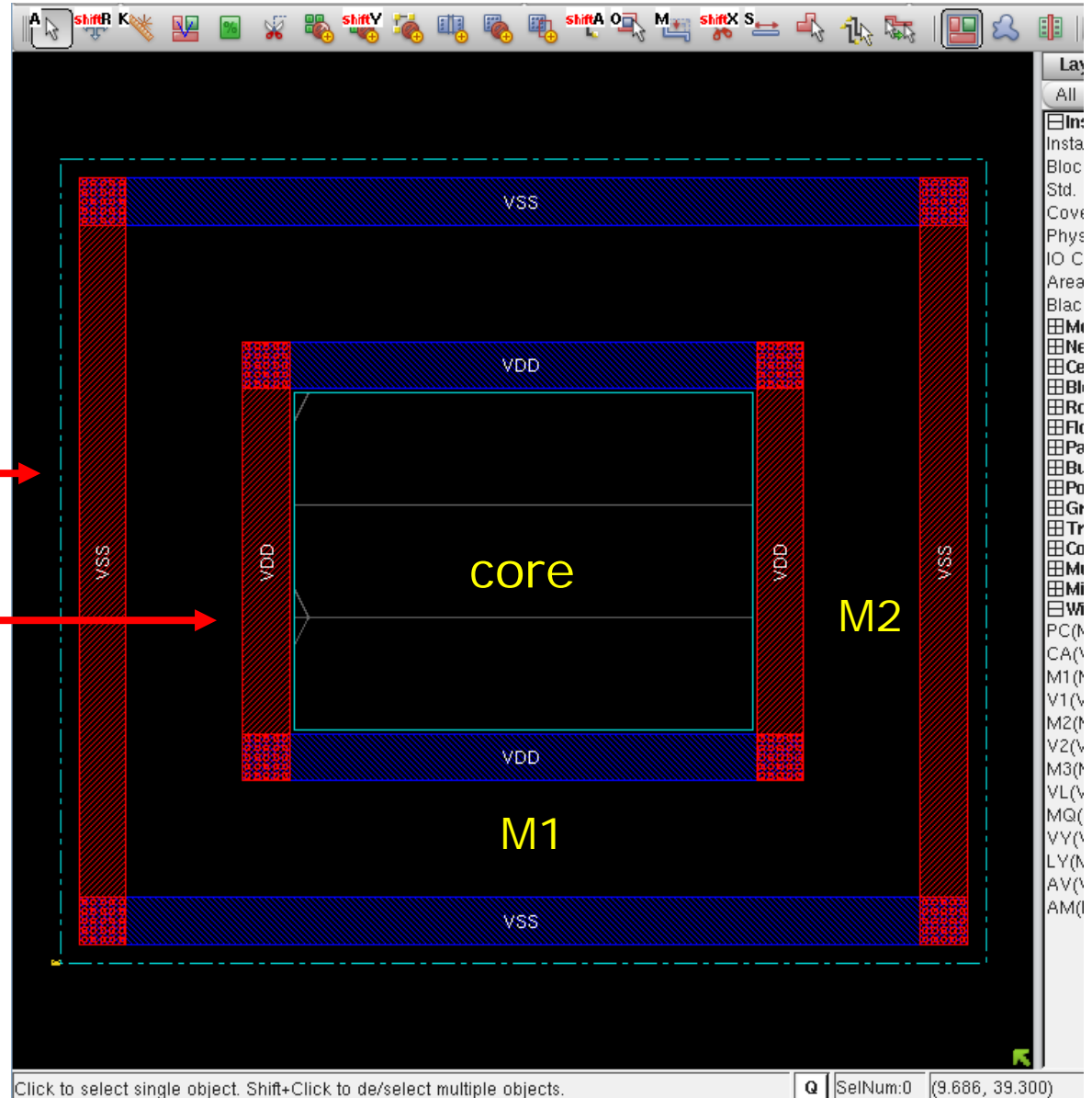
# Power Rings

## modulo6

Ground



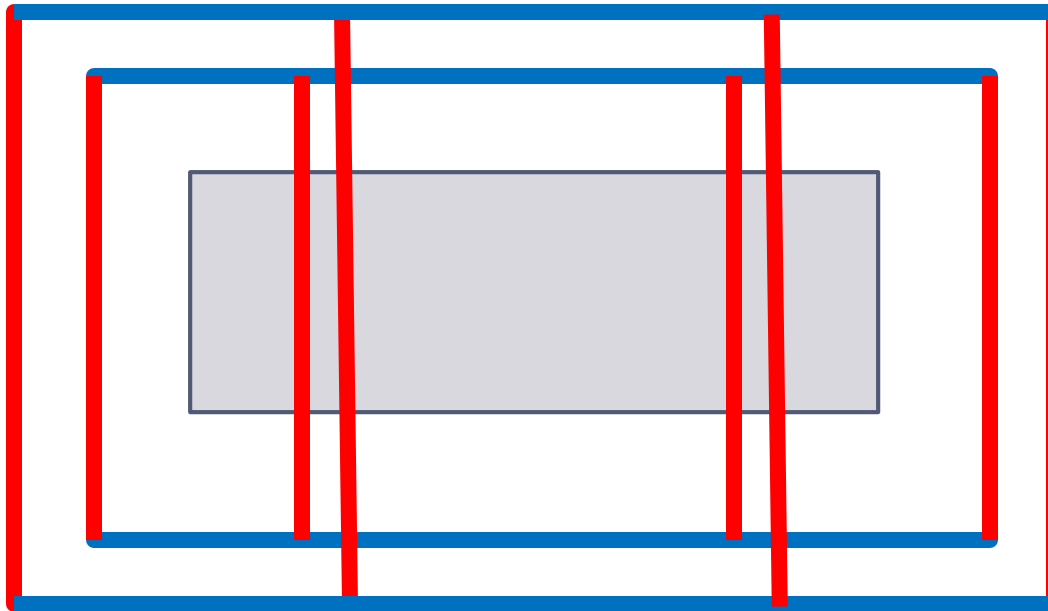
Power



# Power stripes

---

Optional: Additional connections from power rings to power/ground rails in the core.



Tcl command: `addStripe`

---



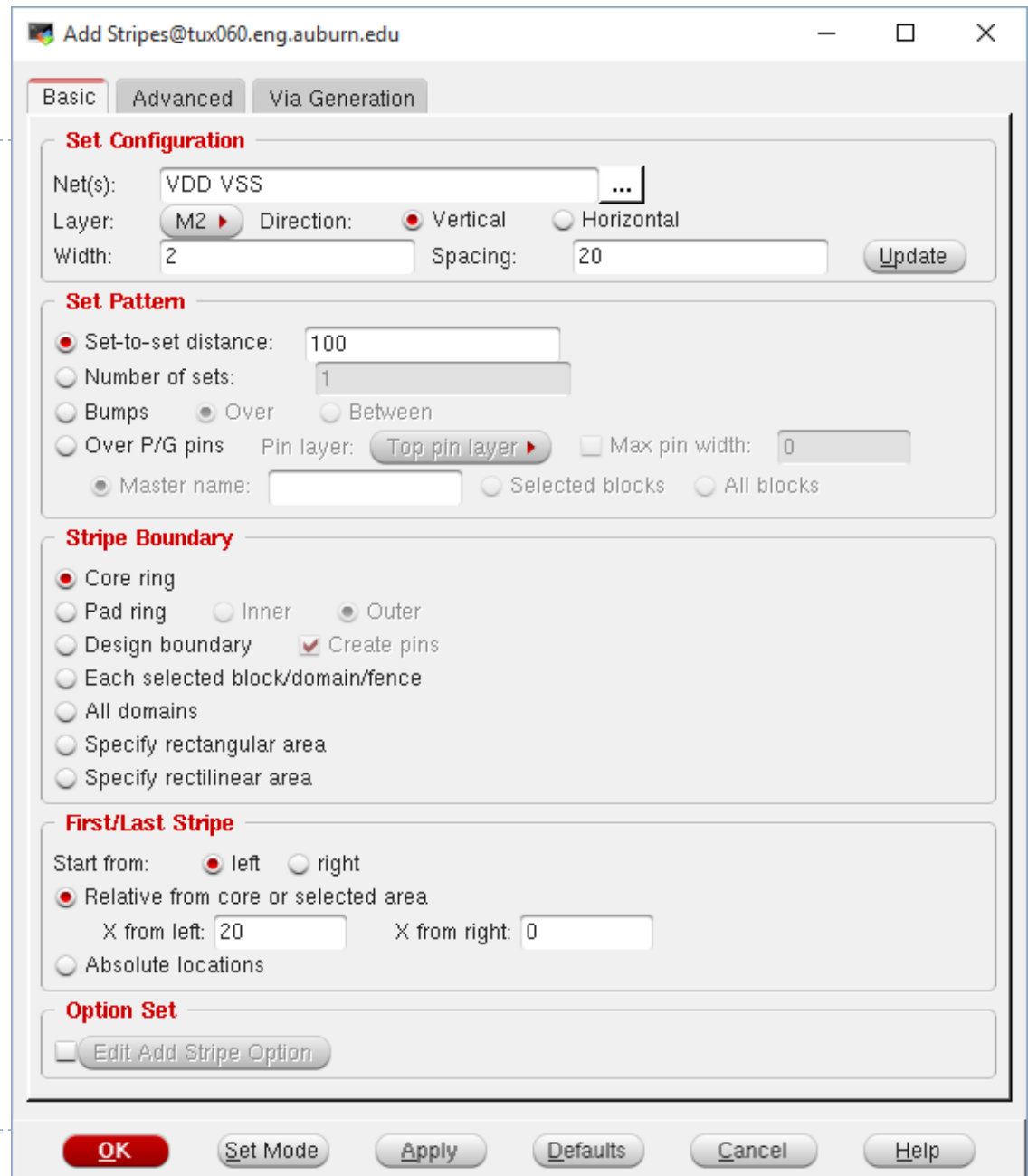
# Add Stripes Tool

Stripe wires

Between sets of stripes

Use rings around core

Space from core edges





# Add power stripes Tcl command

---

# Make Power Stripes. This step is optional.

# Check the stripe spacing (set-to-set-distance = \$sspace)

# and stripe offset (xleft-offset = \$soffset)

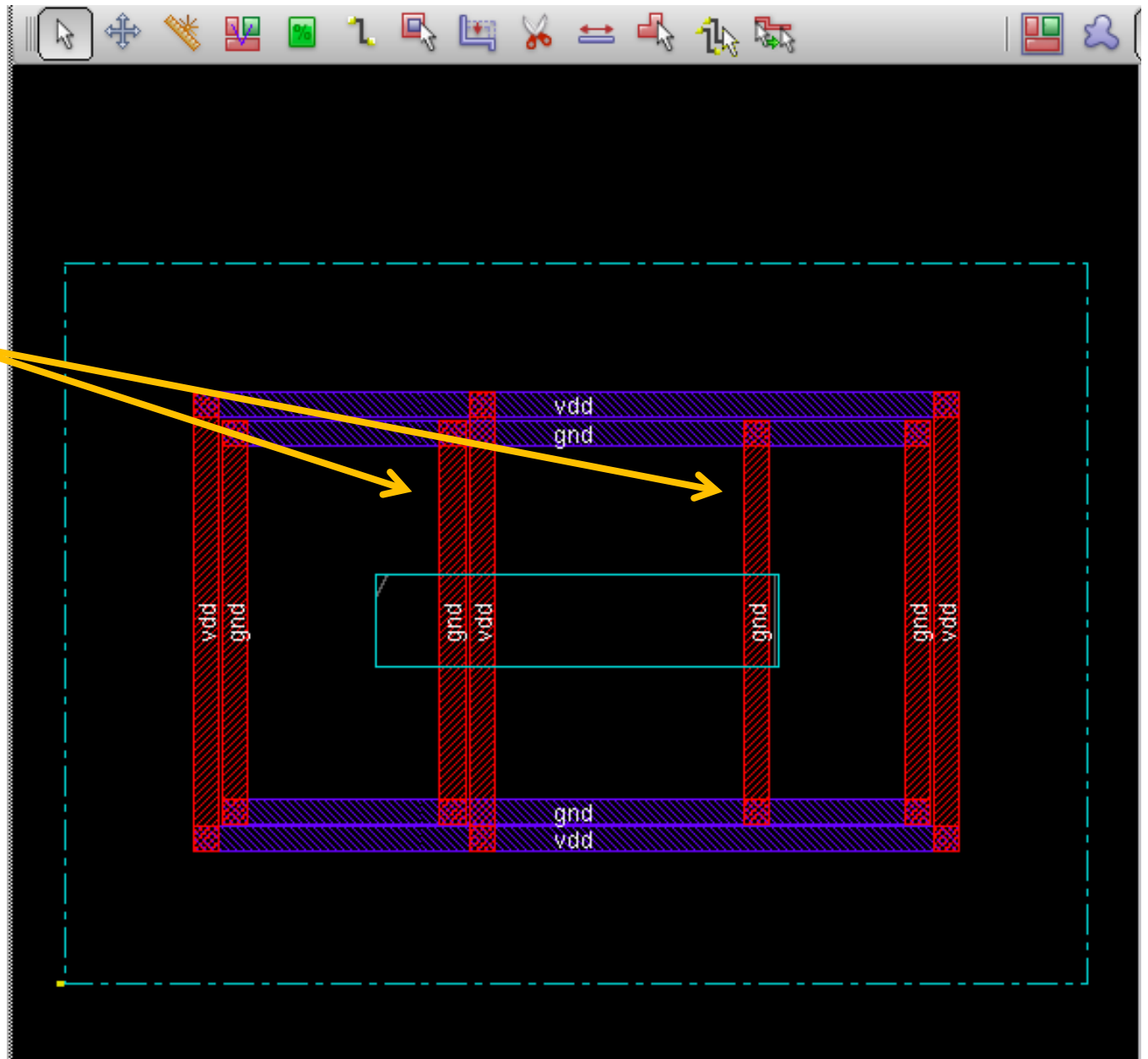
```
addStripe -nets { VSS VDD } \  
  -layer M2 \  
  -width $swidth \  
  -spacing $pspace \  
  -xleft_offset $soffset \  
  -set_to_set_distance $sspace \  
  -block_ring_top_layer_limit M3 \  
  -block_ring_bottom_layer_limit M1 \  
  -padcore_ring_bottom_layer_limit M1 \  
  -padcore_ring_top_layer_limit M3 \  
  -stacked_via_top_layer M3 \  
  -stacked_via_bottom_layer M1 \  
  -max_same_layer_jog_length 3.0 \  
  -snap_wire_center_to_grid Grid \  
  -merge_stripes_value 1.5
```

Lowest layer to  
use if object  
encountered

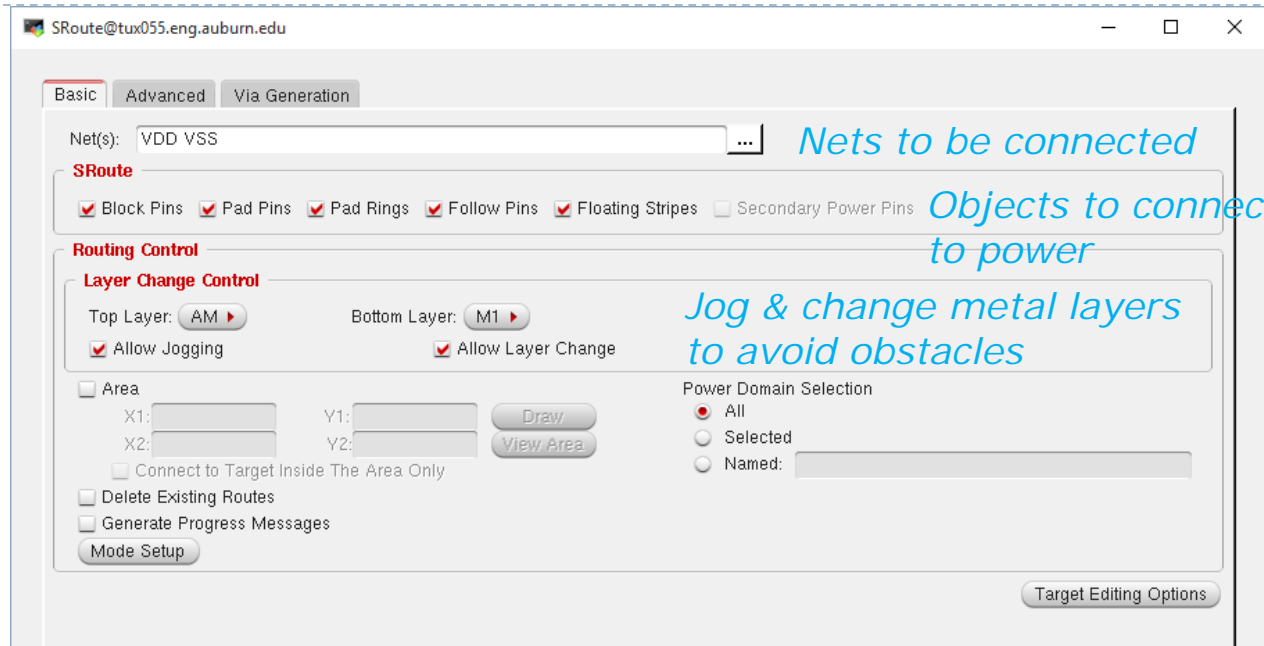
Merge with core ring  
if this close



Power stripes  
added to rings



# Special route – VDD/VSS wires between rings and core power rails



**Tcl:**

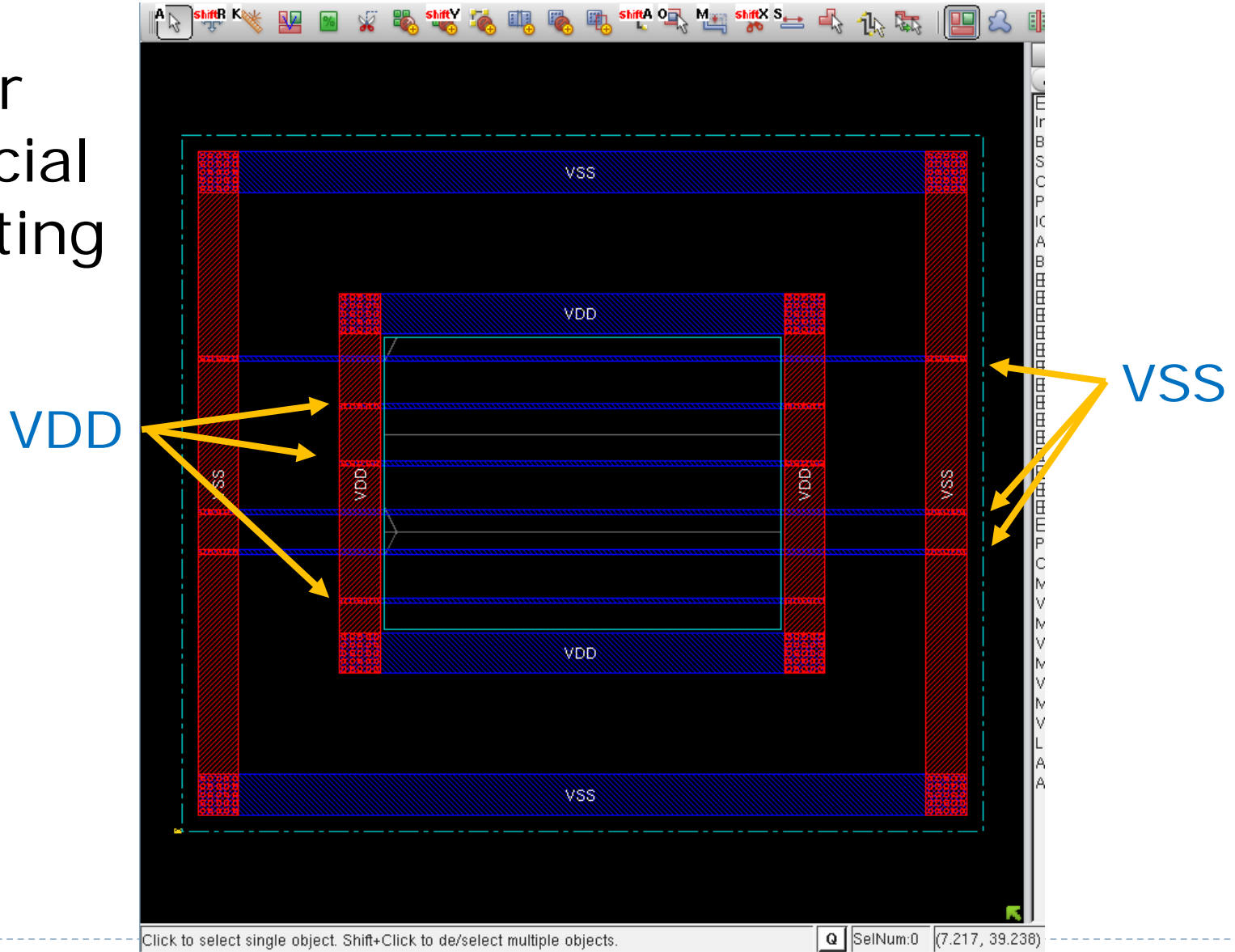
```
sroute -connect { blockPin padPin padRing corePin floatingStripe } \  
-allowJogging true \  
-allowLayerChange true \  
-blockPin useLef \  
-targetViaLayerRange { M1 AM }
```

} To avoid  
DRC errors

↑  
Objects to  
connect to  
rings/stripes



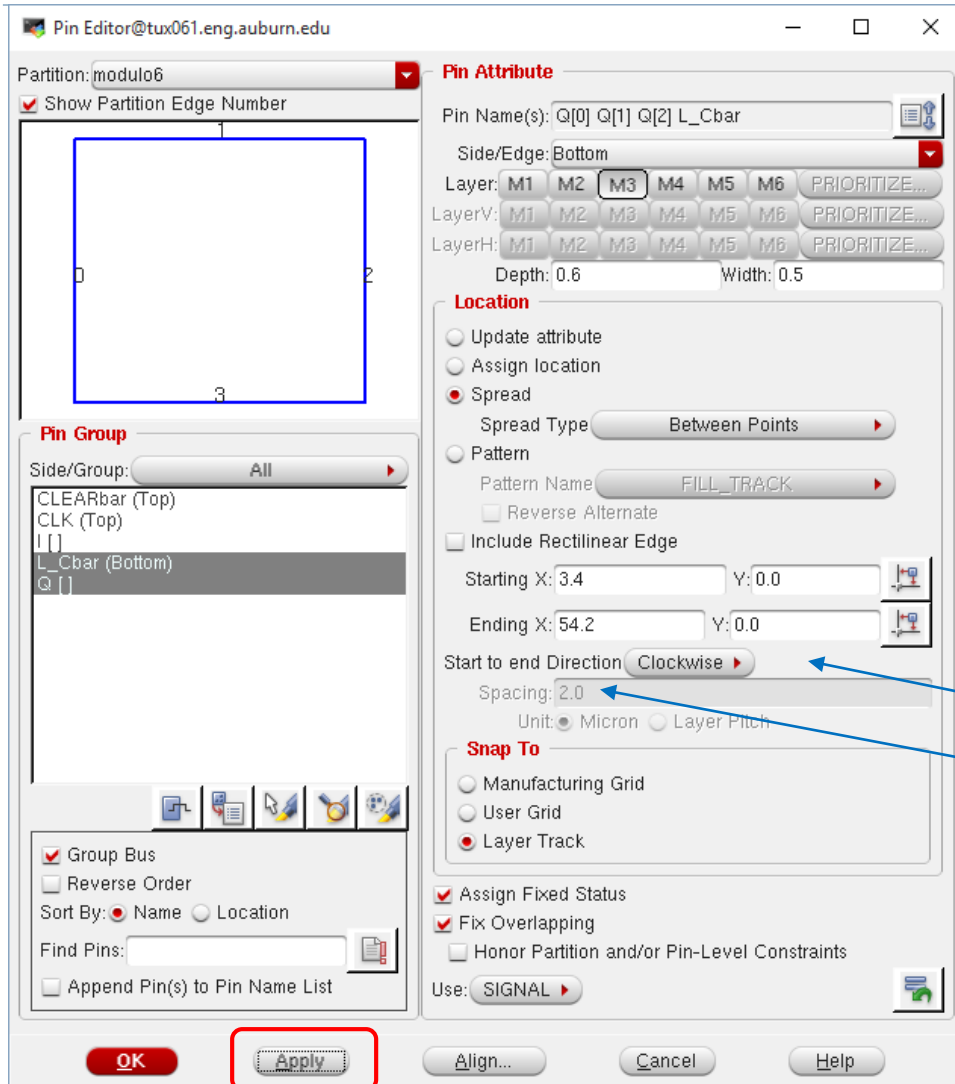
# After Special Routing



Click to select single object. Shift+Click to de/select multiple objects.

Q SelNum:0 (7,217, 39,238)

# Pin Editor Form – to adjust placement



Select pins and side

Pin layer and geometry

Pin spacing pattern:

space from Start

space from Center

spread between coord's

spread across side/edge

spacing direction

spacing amount (unless "spread")

Use "Apply" to experiment with options until satisfied.

# Pin editing Tcl command

---

# Pin placement section

```
editPin -side TOP \  
  -layer M3 \  
  -fixedPin 1 \  
  -spreadType CENTER \  
  -spacing 4 \  
  -pin { I[0] I[1] I[2] CLEARbar CLK }
```

*Space by 4,  
begin in center*

```
editPin -side BOTTOM \  
  -layer M3 \  
  -fixedPin 1 \  
  -spreadType RANGE \  
  -start { 4 0 } \  
  -end { 50 0 } \  
  -spreadDirection CounterClockwise \  
  -pin { Q[0] Q[1] Q[2] L_Cbar }
```

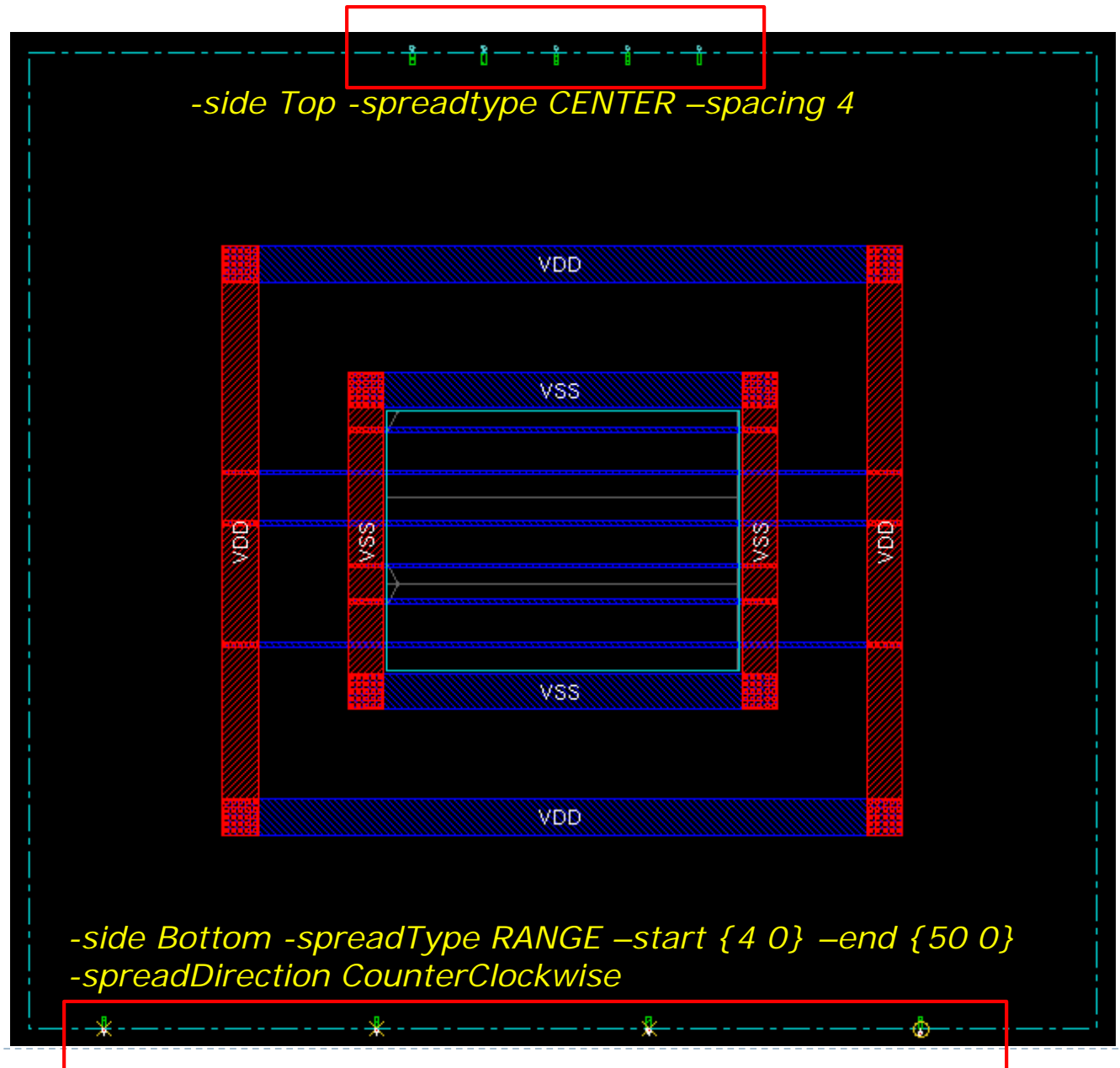
*Spread out evenly  
between end points*



# Pin editing Example

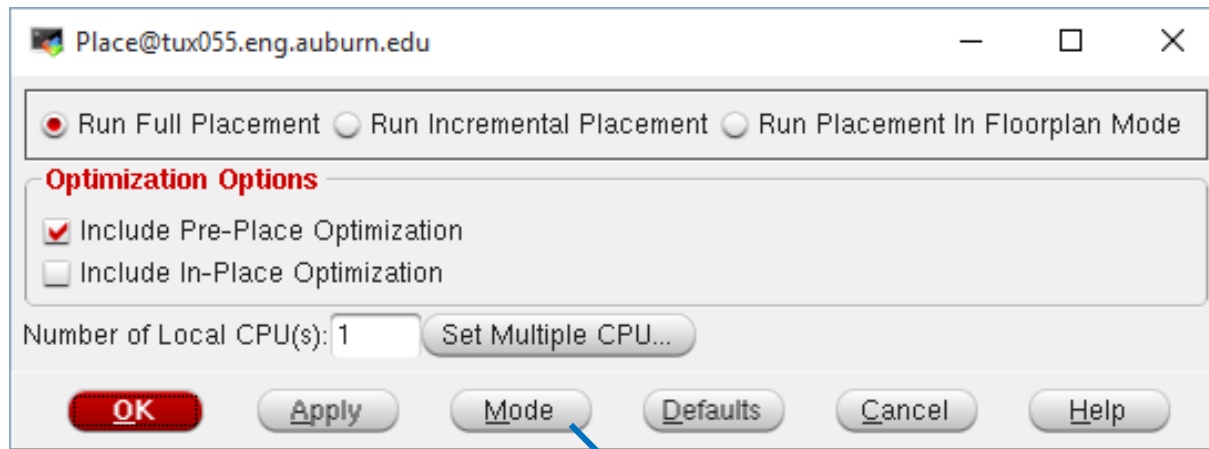
Top pins spread  
from center with  
spacing = 4

Bottom pins  
spread evenly  
between (x y) =  
(4,0) to (50,0)



# Place standard cells setup

---



*Mode on next slide*

## Tcl Commands

```
setPlaceMode -timingDriven true \  
-congEffort auto
```

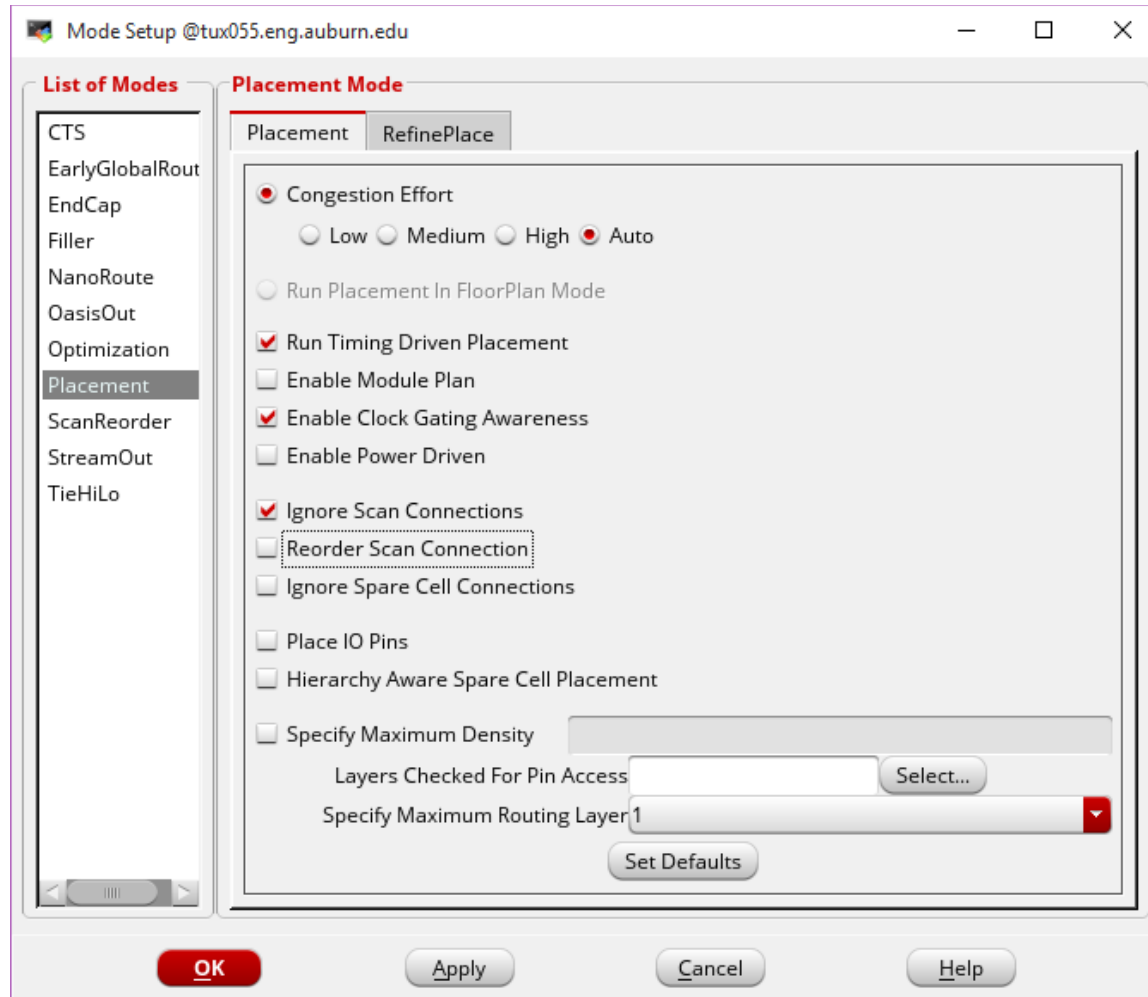
```
placeDesign  
setDrawView place  
(to view the cells)
```

*Optional placeDesign switches:  
-inPlaceOpt or -prePlaceOpt*





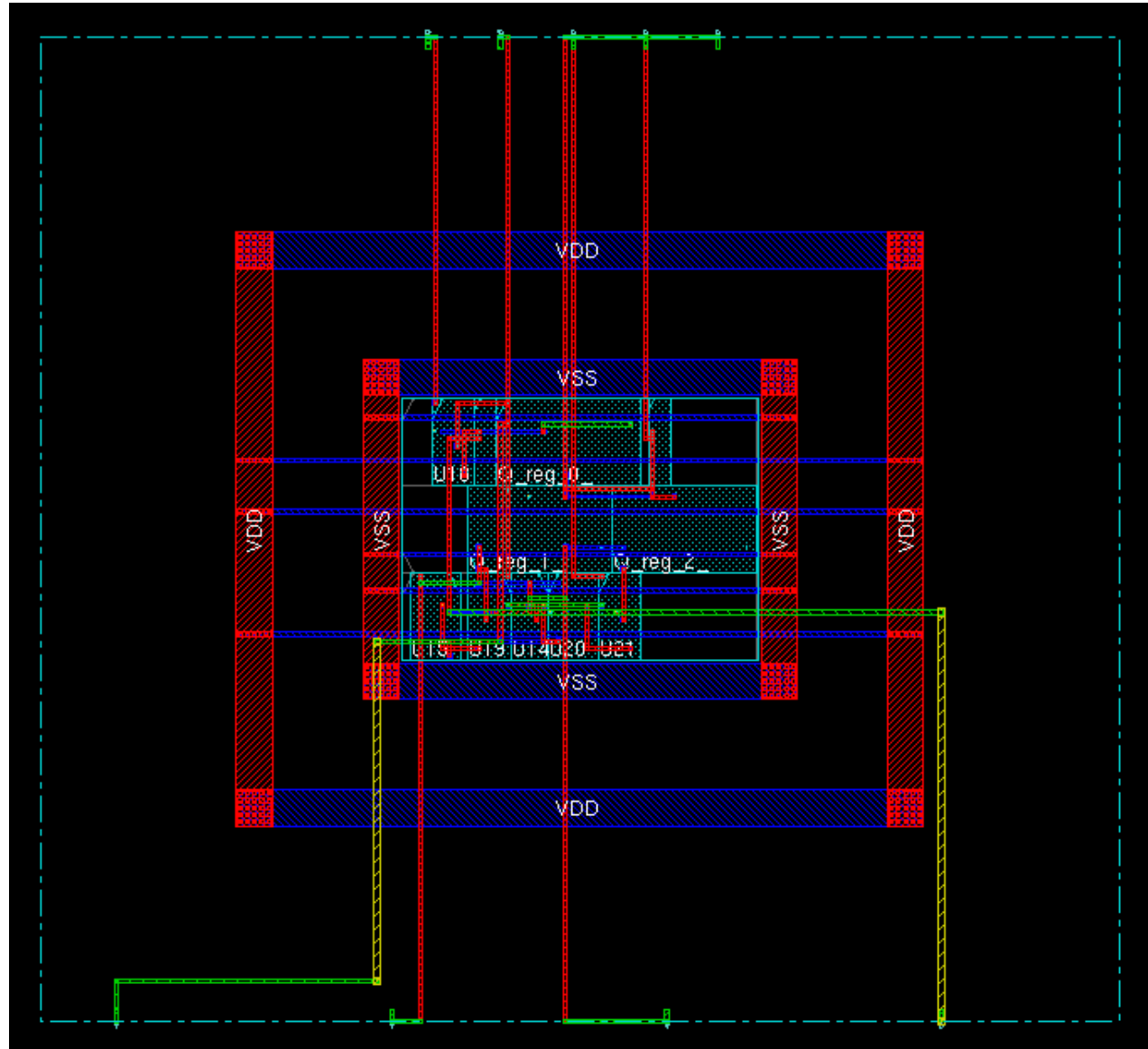
# Place Standard Cells – Mode Setup



setPlaceMode  
-congEffort auto \  
-timingDriven true \  
-ignoreScan true

After  
Placing  
Cells

Draw View  
"place"



# Timing analysis and optimization

---

- ▶ Ideally perform at three times during the design flow
  - ▶ **Pre-CTS** (clock tree synthesis) – trial route after placing cells
  - ▶ **Post-CTS** – clock tree should improve timing
  - ▶ **Post-Route** – after completed routing
- ▶ **timeDesign**: create trial route, extract delays, analyze timing, generate reports (reg2reg, in2reg, reg2out)
- ▶ **optDesign**: resize gates, restructure netlist, add/delete buffers, swap pins, move instances



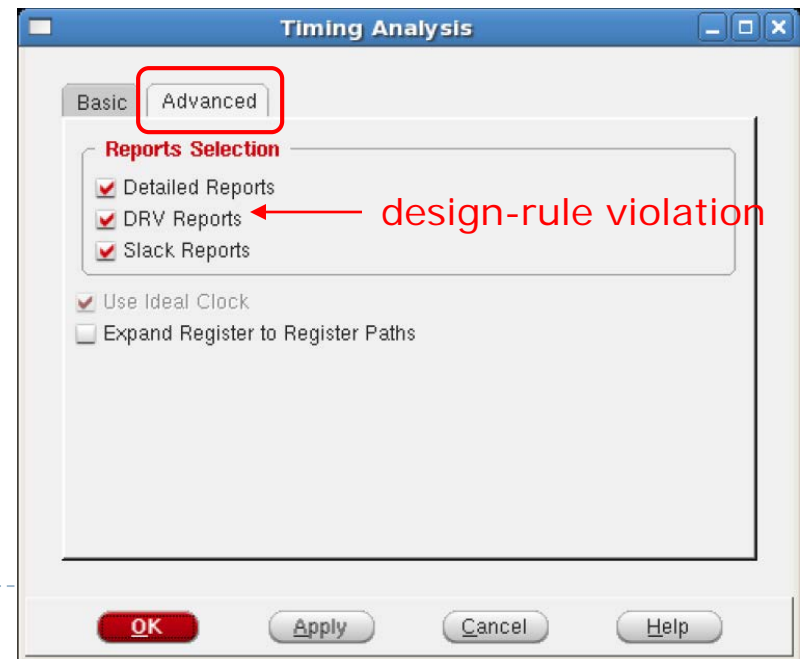
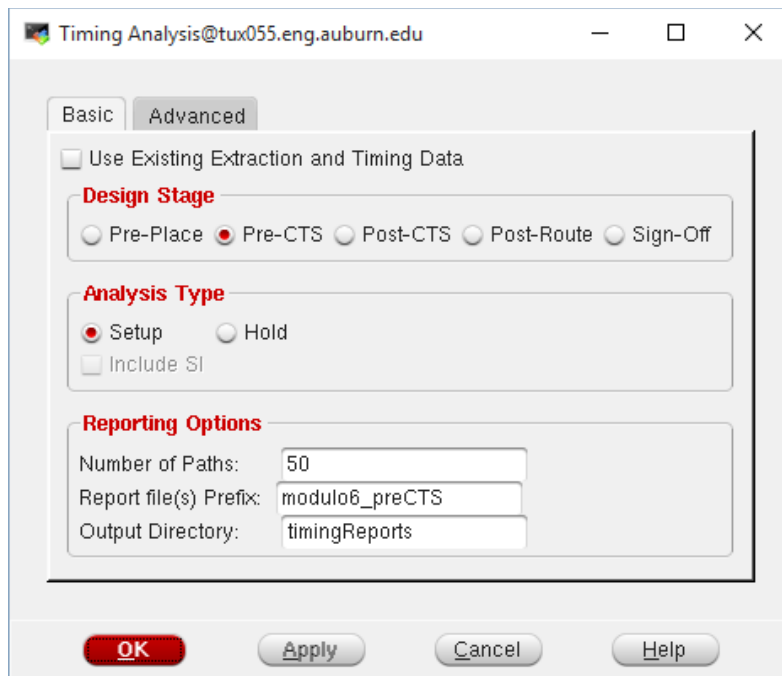
# Timing reports

Option for postRoute only

**setAnalysisMode** -analysisType onChipVariation -skew true  
-clockPropagation sdcControl

**timeDesign** -preCTS -idealClock -numPaths 50 -prefix preCTS \  
-outDir \${BASENAME}\_reports/preCTS

(or: postCTS, postRoute)

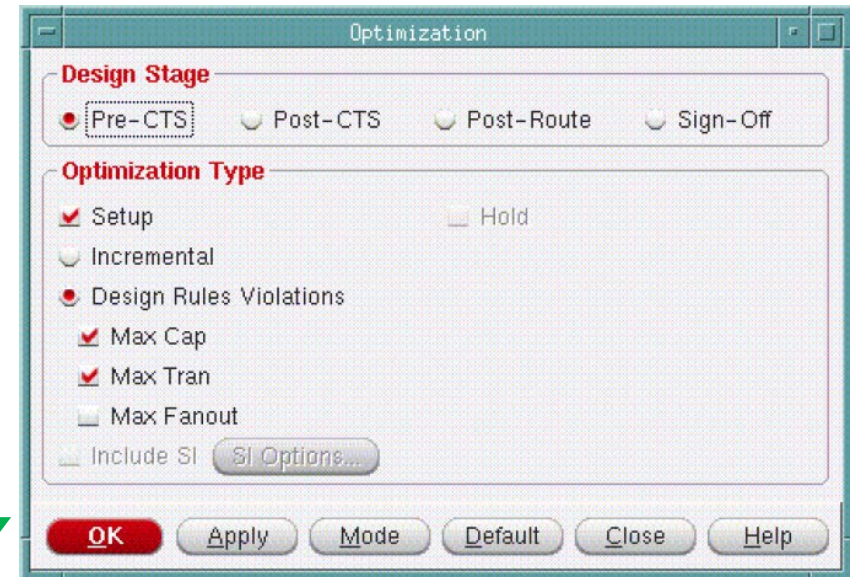


# Timing optimization

Command for postRoute only

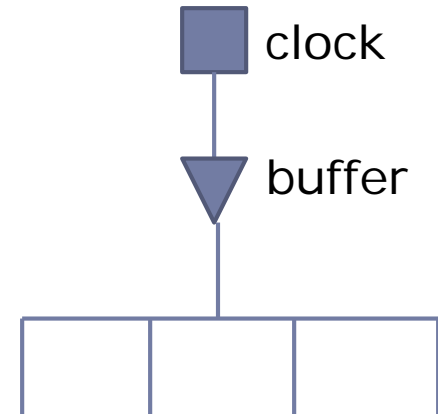
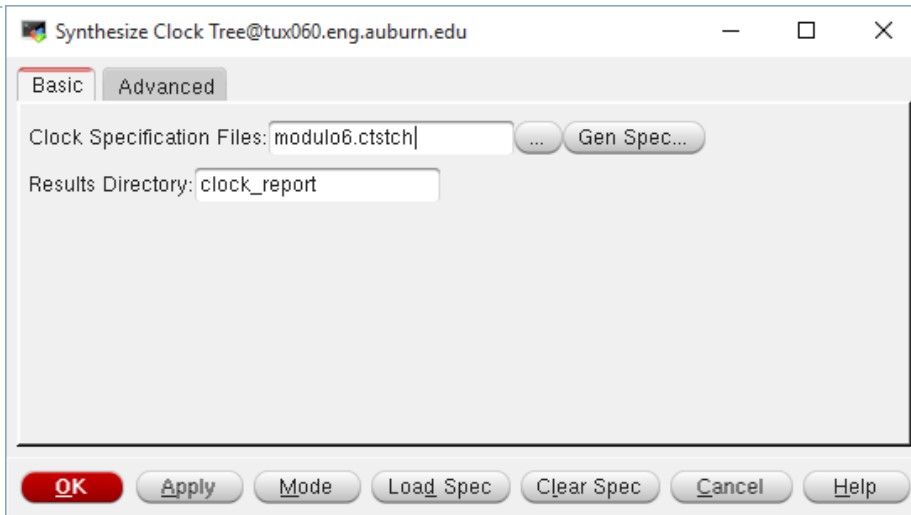
**setAnalysisMode** -analysisType onChipVariation -skew true -clockPropagation sdcControl

```
setOptMode -yieldEffort none
setOptMode -effort high
setOptMode -maxDensity 0.95
setOptMode -fixDRC true
setOptMode -fixFanoutLoad true
setOptMode -optimizeFF true
setOptMode -simplifyNetlist false
setOptMode -holdTargetSlack 0.0
setOptMode -setupTargetSlack 0.0
clearClockDomains
setClockDomains -all
setOptMode -usefulSkew false
optDesign -preCTS -drv \
    -outDir ${BASENAME}_reports/preCTSOptTiming
```



**Removed from GUI in current version.**

# Clock tree synthesis (CTS)



# Create the clock tree spec from the .sdc file (from synthesis)

```
createClockTreeSpec -output $BASENAME.ctstch
```

# Set -routeGuide to use routing guide during CTS.

```
setCTSMode -routeGuide true
```

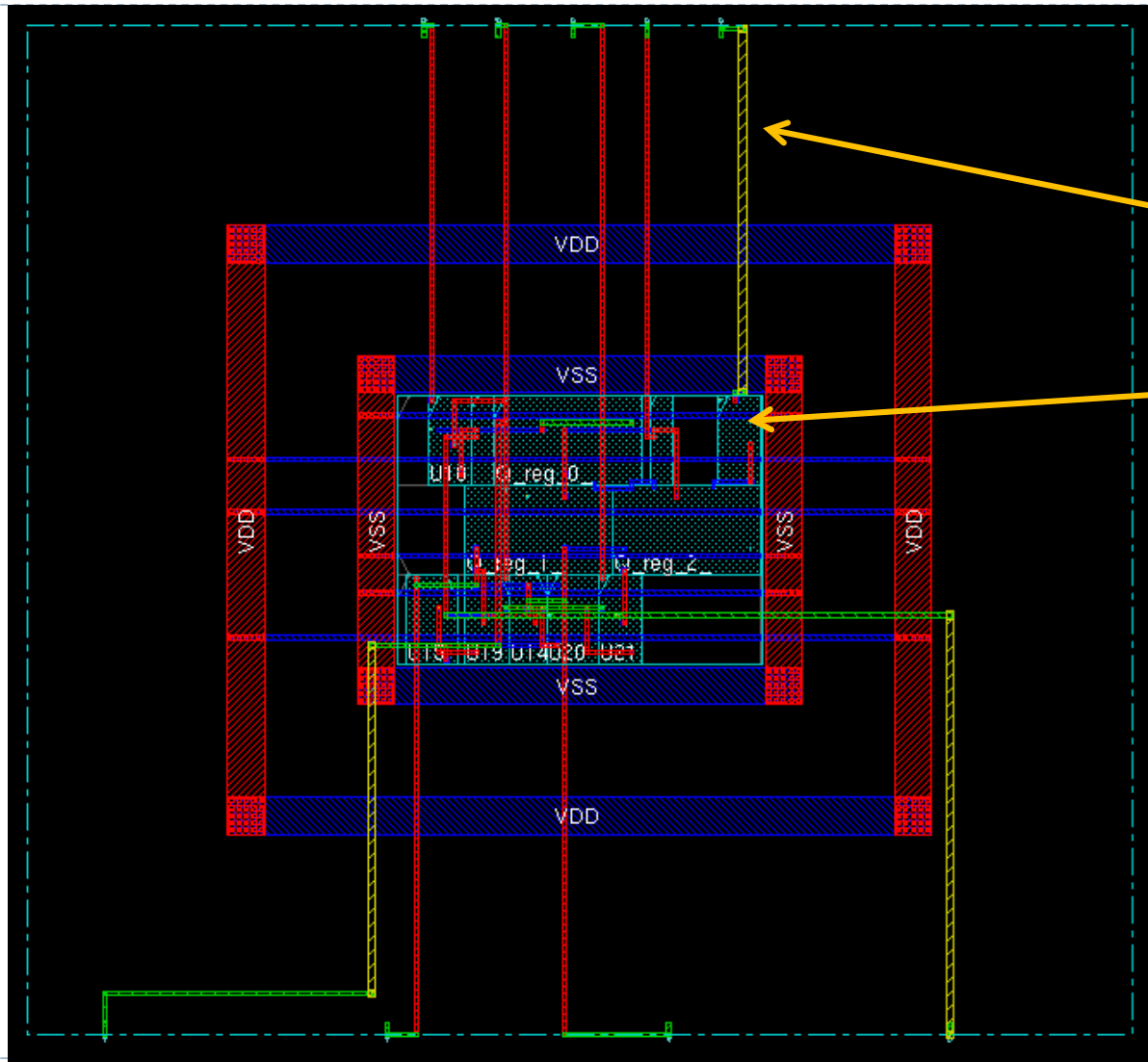
# Set routeClkNet to use NanoRoute during CTS.

```
setCTSMode -routeClkNet true
```

# Perform clock tree synthesis

```
clockDesign -outDir ${BASENAME}_clock_reports
```

# After clock tree synthesis



Clock net  
changed

Buffer cell  
added



# NanoRoute Setup

*Default options usually OK*

**NanoRoute@tux055.eng.auburn.edu**

**Routing Phase**

- Global Route
- Detail Route Start Iteration default End Iteration default
- Post Route Optimization  Optimize Via  Optimize Wire

**Concurrent Routing Features**

- Fix Antenna  Insert Diodes Diode Cell Name
- Timing Driven Effort 5 Congestion Timing S.M.A.R.T.
- SI Driven
- Post Route SI SI Victim File
- Litho Driven
- Post Route Litho Repair

**Routing Control**

- Selected Nets Only Bottom Layer default Top Layer default
- ECO Route
- Area Route Area Select Area and Route

**Job Control**

- Auto Stop
- Number of Local CPU(s): 1
- Number of CPU(s) per Remote Machine: 1
- Number of Remote Machine(s): 0
- Set Multiple CPU...

Command: **globalDetailRoute**



# Add Filler Cells

From initial Innovus script:

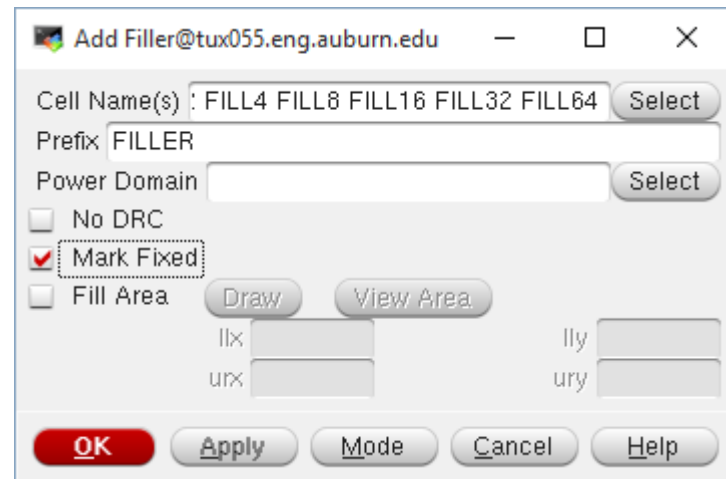
```
# Set the name(s) of the filler cell(s) in the cell libraryset  
fillerCells [list FILL1 FILL2 FILL4 FILL8 FILL16 FILL32 FILL64 ]
```

After routing complete:

```
# Add the filler cells  
setFillerMode -corePrefix ${BASENAME}_FILL -core ${fillerCells}  
addFiller -cell $fillerCells -prefix ${BASENAME}FILL -markFixed
```

Menu:

Place > Physical Cell > Add Filler





# Design verification

---

- ▶ Verify **connectivity**, looking for:
  - ▶ Antennas
  - ▶ Opens
  - ▶ Loops
  - ▶ Unconnected pins
- ▶ Verify **geometry** with data from LEF file:
  - ▶ Widths
  - ▶ Spacings
  - ▶ Internal geometries of wires/objects

*TCL:*

```
verifyConnectivity -type regular -error 50 -warning 50 -report Conn_regular.rpt  
verifyConnectivity -type special -error 50 -warning 50 -report Conn_special.rpt  
verifyGeometry -allowSameCellViols -noSameNet -noOverlap -report Geom.rpt
```



# Write results

---

```
# Export the DEF, v, spef, sdf, lef, and lib files
global dbgLefDefOutVersion
set dbgLefDefOutVersion 5.5
defOut -floorplan -netlist -routing $BASENAME.def
saveDesign ${BASENAME}_done.enc -def

puts "-----Output ${BASENAME}_soc.v file-----"
saveNetlist [format "%s_soc.v" $BASENAME]

puts "-----Save models for hierarchical flow-----"
saveModel -cts -sdf -spef -dir ${BASENAME}_hier_data

extractRC -outfile $BASENAME.cap
rcOut -spef $BASENAME.spef
#delayCal -sdf $BASENAME.sdf -idealclock
write_sdf $BASENAME.sdf

# Generate GDSII file from Innovus database
See next slide
```

---



# Generate GDSII file from Innovus database

---

# Generate mask data for layout in GDSII format

```
setStreamOutMode -snapToMGrid true
```

```
streamOut ${BASENAME}.gds2 \
```

```
  -structureName ${BASENAME} \
```

```
  -mode ALL \
```

```
  -outputMacros \
```

```
  -stripes 1 \
```

```
-mapFile /class/ELEC6250/cmos8hp/techlef/v.20160204/lef/bicmos8hp_soc2gds.map \
```

```
-merge {/class/ELEC6250/cmos8hp/std_cell/v.20130404/gds2/IBM_BICMOS8HP_SC_IP2V_I2T_RVT_091712.gds }
```

Standard cell layouts

