ICLAB 2024 Autumn
LEC 11 : Cell-Based APR Design Flow

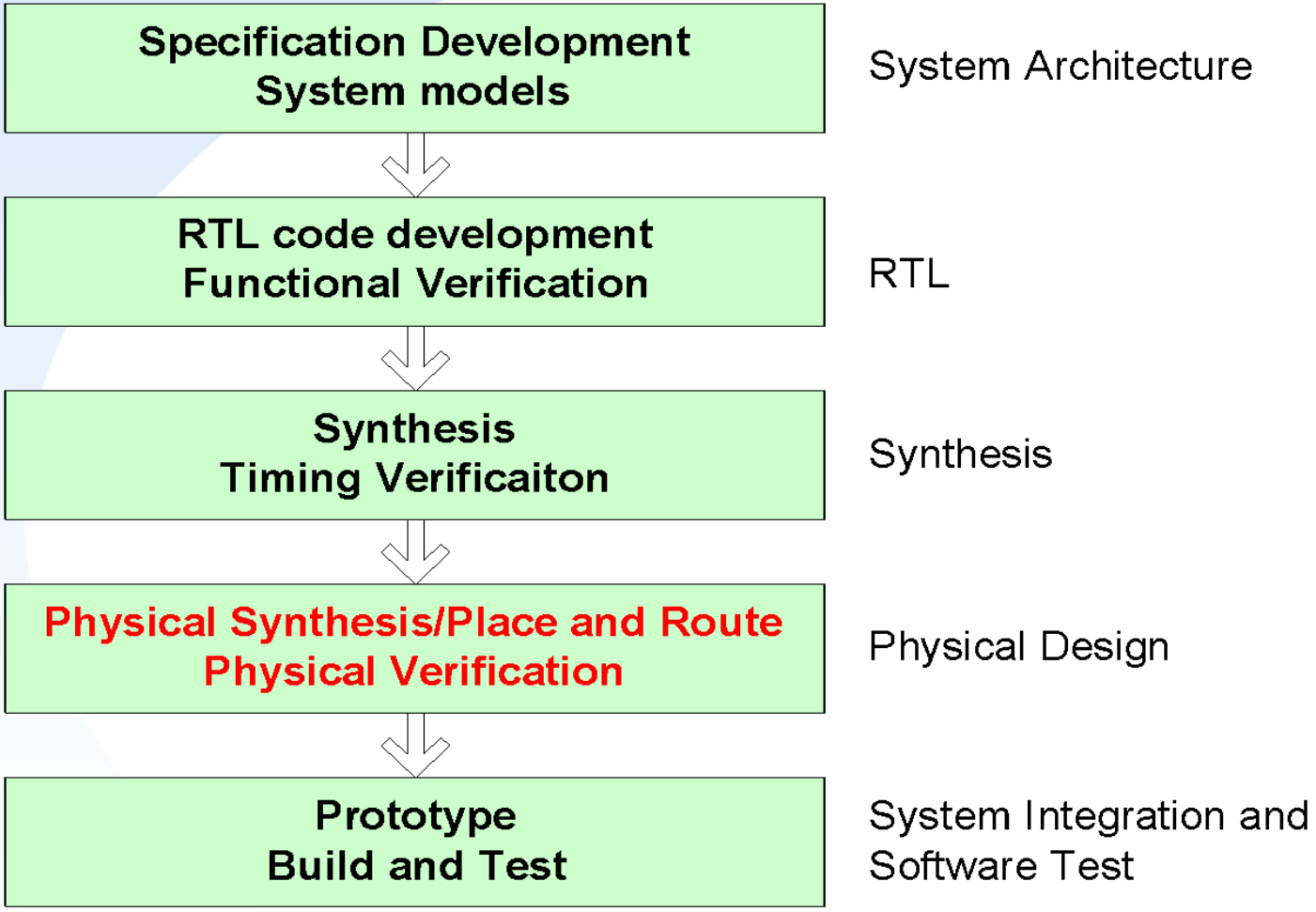
Lecturer: Bang-Yuan Xiao

Outline

- Section 1- What's APR
- Section 2- SOP of APR

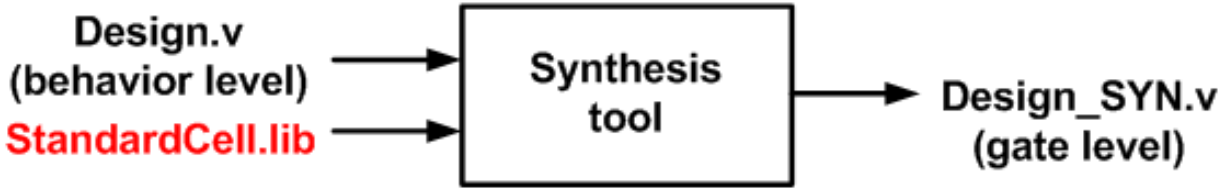
Introduction

Cell-based Design Flow

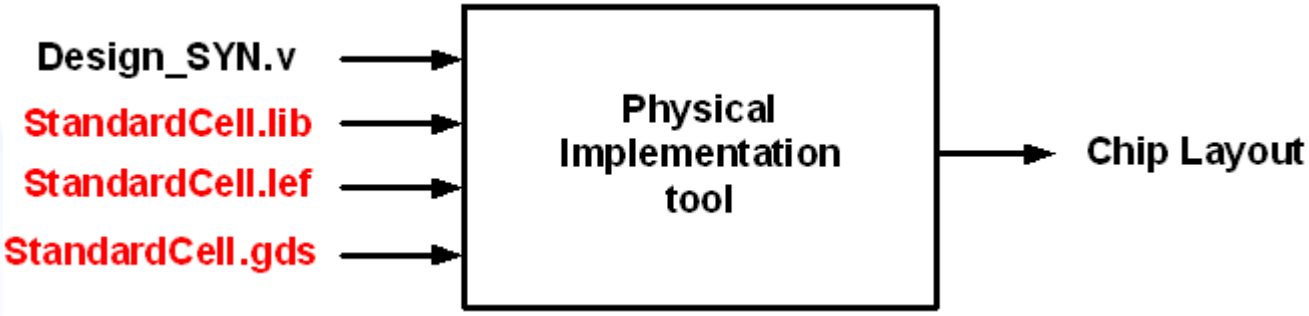


Review Cell-based Design Flow

- RTL coding
- Synthesis



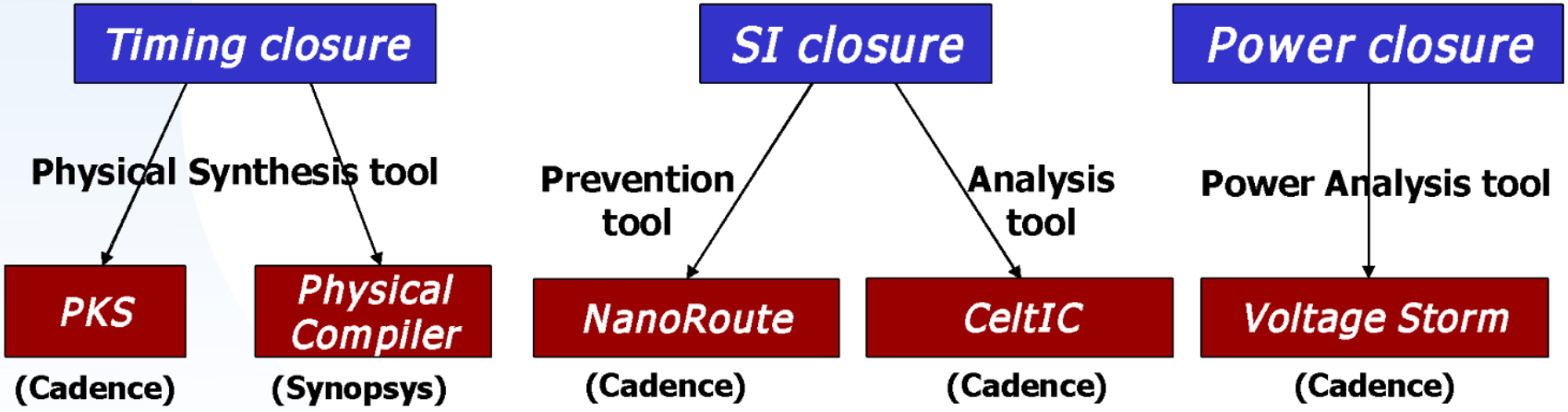
- Physical Implementation



IC Compiler · Innovus

Wiring Problem

- **New problems due to large wire resistance and capacitance**
  - Timing closure
  - Signal Integrity(SI) closure(crosstalk, …)
  - Power closure (IR drop, …)

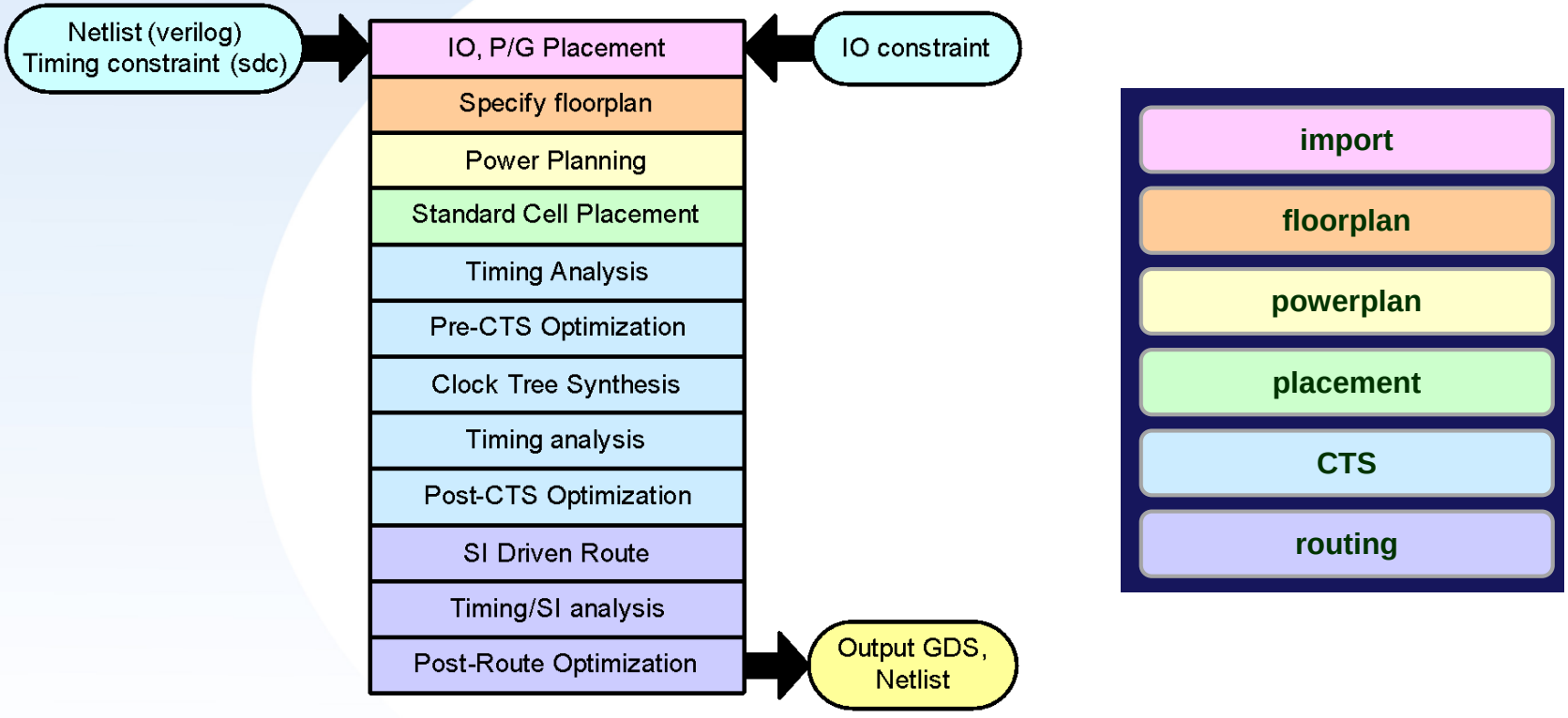


Outline

- Section 1- What's APR
- **Section 2- SOP of APR**

APR Flow

Innovus Design Flow



1. Data Preparation

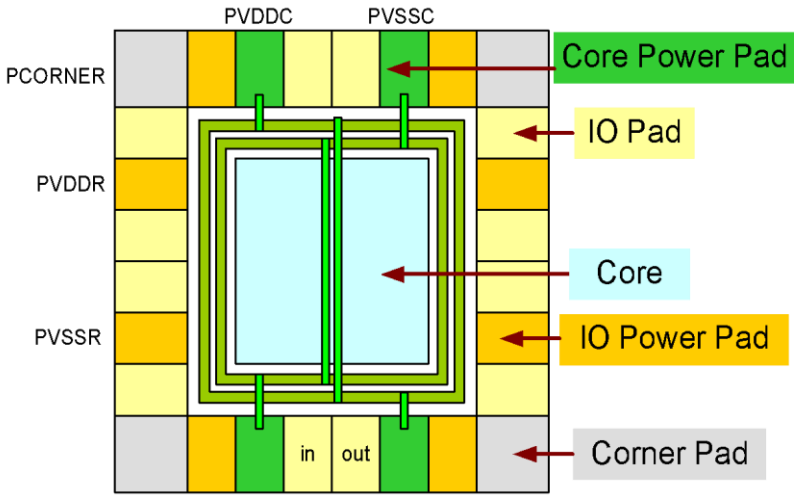
- **Library files**
  - Timing libraries (LIB)
    - slow.lib
    - fast.lib
    - umc18io3v5v_slow.lib
    - umc18io3v5v_fast.lib
    - HardMacro_slow.lib
    - HardMacro_fast.lib
  - Physical libraries (LEF)
    - umc18_6lm.lef
    - umc18io3v5v_6lm.lef
    - umc18_6lm_antenna.lef
    - HardMacro.vclef
  - RC extraction
    - umc18_1p6m.captbl
    - RCGen.tch
  - CeltIC libraries
    - slow.cdb
    - fast.cdb
  - GDSII layout
    - umc18.gds2
    - umc18io3v5v_61m.gds2
    - HardMacro.gds2
- **Timing constraint files (User Data)**
  - Generate timing constraint files from Design Complier
    - dc_shell-t> write_sdc CHIP.sdc
- **Design netlist files (User Data)**
  - Synthesized design netlist
    - core_syn.v
  - Chip design netlist (combine core_syn.v&chip_shell.v)
    - CHIP_SYN.v
  - IO pad location file
    - CHIP.io

1. Data Preparation – Library Files

- **LIB Library : timing information**
  - Calculate cell delay according to different combinations of input slew rate and output loading capacitance
  - Using SignalStorm Library Characterizer
- **LEF Library : process and cell information**
  - Metal layer information, routing pitch, default direction
  - Cell size, pin position, pin metal layer
- **RC Extraction**
  - RC extraction for further power analysis, simulation etc
- **CeltIC Library**
  - cdB model (noise model for each cell which will be use in SI optimization phase)
- **GDSII**
  - planar geometric shapes and other information about the layout in hierarchical form

1. Data Preparation - Design Netlist

- **Prepare chip design netlist: CHIP_SYN.v**
  - Add pad netlist to synthesized design netlist
  - Combine *designName_SYN.v* and *CHIP_SHELL.v*
    - *cat CHIP_SHELL.v CORE_SYN.v > CHIP_SYN.v*
  - ***CHIP_SHELL.v*** contains I/O, I/O power, and core power
    - Note: Refer to pad cell library datasheet
      - ***evince ~iclabta01/umc018/Doc/umc18io3v5v.pdf &***



1. Data Preparation - Design Netlist

- **Core/IO power pad selection (UMC 0.18)**
  - Pad cell categories

Special Cells

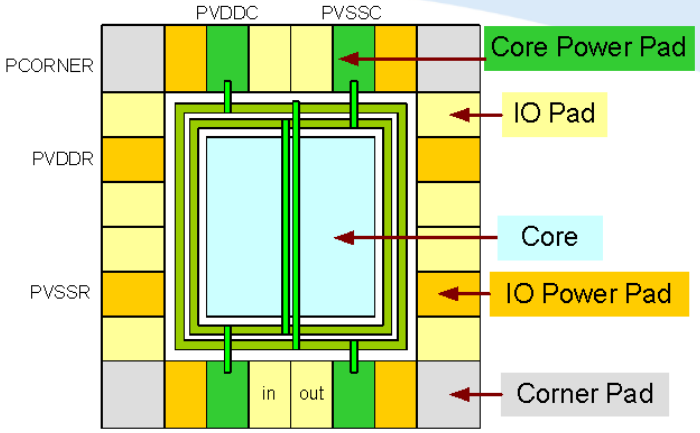
Core power pads: PVDDC, PVSSC

IO power pads: PVDDR, PVSSR

I/O filler pads: PFILL, …

I/O Cells

P2A	P4A	P8A	P16A	P24A
		P8B	P16B	P24B
P2C	P4C	P8C	P16C	P24C




```
PVDDC VDDC0 ();
PVSSC GNDC0 ();
P8C I_CLK ( .Y(C_clk),     .P(clk),    .A(1'b0), .ODEN(1'b0),
| .OCEN(1'b0), .PU(1'b1), .PD(1'b0), .CEN(1'b0), .CSEN(1'b1) );
```

  - Core power pad
    - One set core power pad (PVDDC along with PVSSC) can provide 40~50mA current
  - IO power pad
    - One set IO power pad (PVDDR along with PVSSR) can provide the power for
      - 3~4 output pads/PG pad set
      - 6~8 input pads pads/PG pad set

1. Data Preparation – I/O Files

- **Prepare IO pad location file: CHIP.io**
  - Edit IO pad location file
    - CHIP.io
  - Syntax
    - Pad: pad_instance_name direction [pad_type]
  - Categories

Field	Usage	
pad_instance_name	Corresponding instance name of pad in the chip design netlist	
direction	Specify pad location in the floorplan	S (southern pads)
		E (eastern pads)
		W (western pads)
		N (northern pads)
pad_type	Specify pad type (only for pad filler and corner pad)	PFILL (pad fillers)
		PCORNER (corner pads)

1. Data Preparation – I/O Files

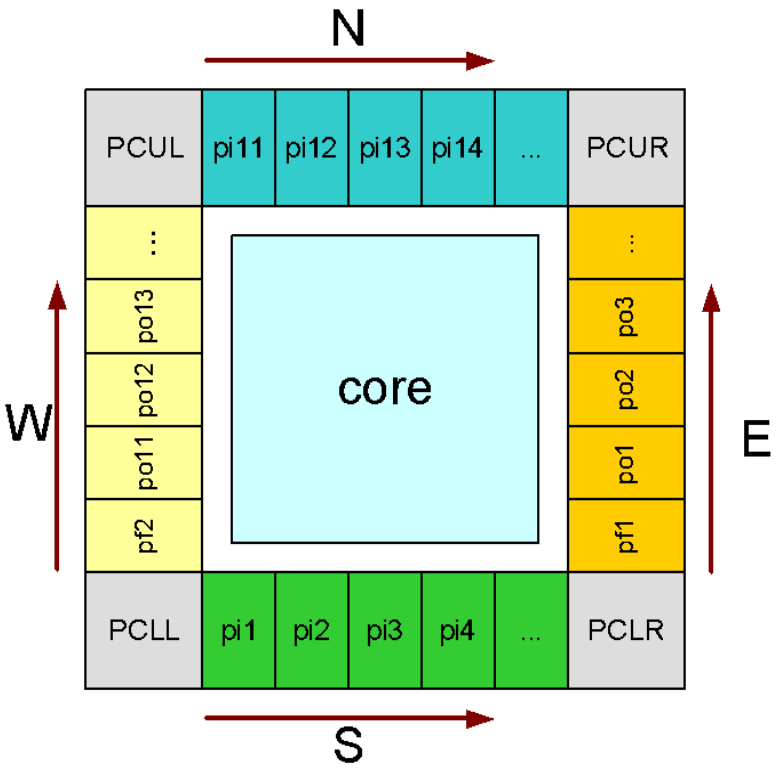
Example of IO pad location file

CHIP_SYN.v

```
PVDDR pi11 ();
PVSSR pi12 ();
```

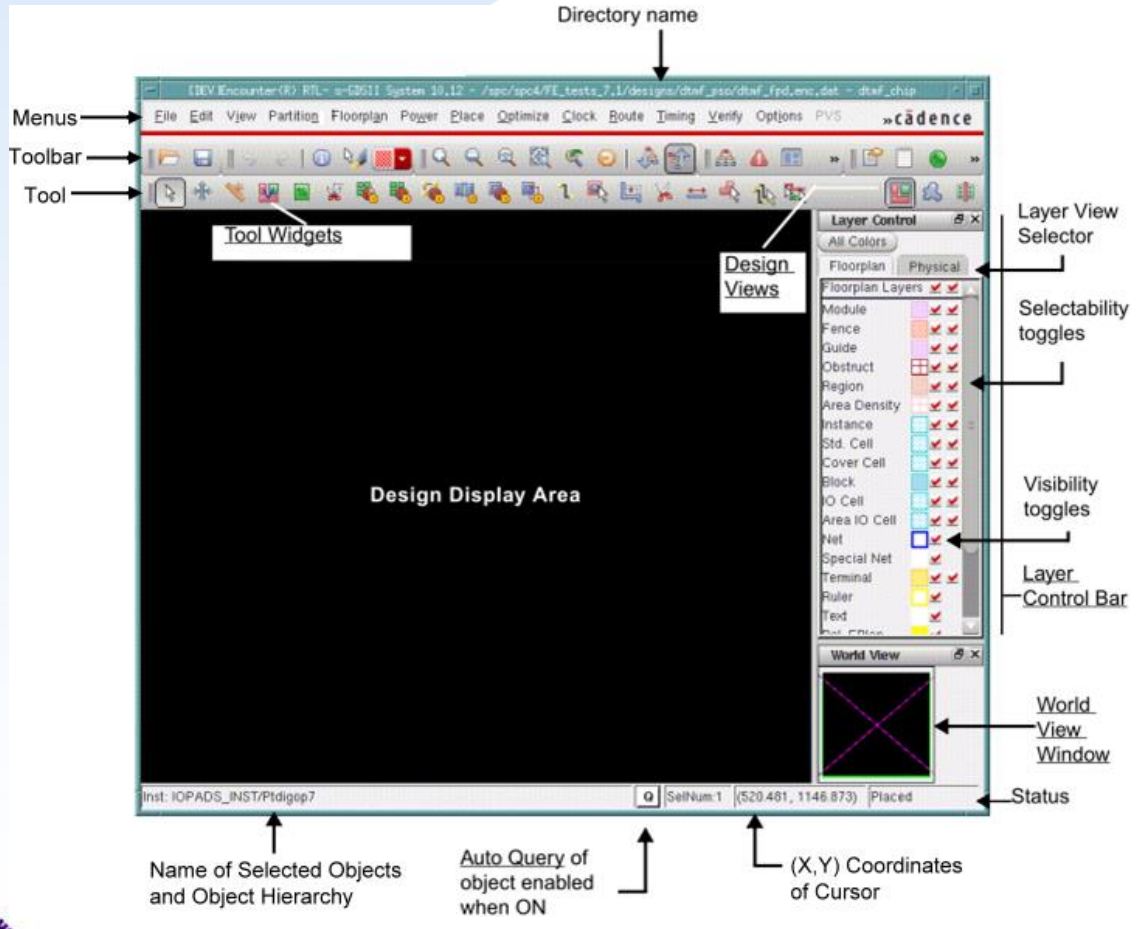
CHIP.io

```
Version: 2

Pad: pi1        S
Pad: pi2        S
Pad: pi3        S
Pad: pi4        S
……

Pad: pf1        E       PFILL
Pad: po1        E
Pad: po2        E
Pad: po3        E
……

Pad: pi11       N
Pad: pi12       N
Pad: pi13       N
Pad: pi14       N
……

Pad: pf2        W       PFILL
Pad: po11       W
Pad: po12       W
Pad: po13       W
……

Pad: PCLL       SW      PCORNER
Pad: PCLR       SE      PCORNER
Pad: PCUL       NW      PCORNER
Pad: PCUR       NE      PCORNER
```



2. Importing Design – Invoke Innovous

- **Command to start an Innovus session**
  - UNIX% innovus



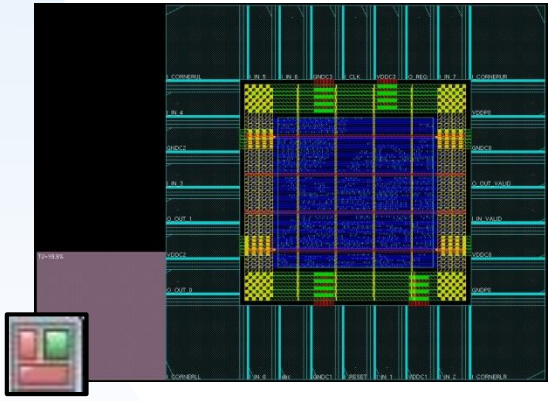
Note:

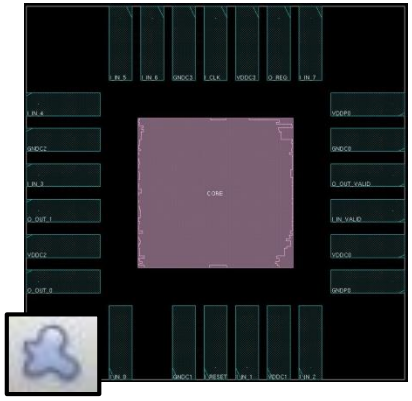
1. **Do not use background execution!!!**
   (The terminal become the interface of command input while running soc innovus)
2. **Log file name:**
   innovus.log
   innovus.log1
   innovus.log2
   …
3. **Command file name:**
   innovus.cmd
   innovus.cmd1
   innovus.cmd2
   …

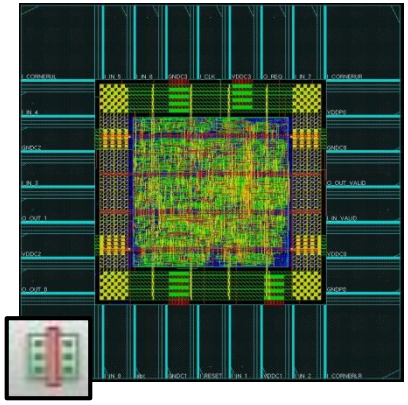
2. Importing Design – Innovus Views

Design Views

-  Floorplan View: displays the hierarchical module and block guides, connection flight lines and floorplan objects
-  Amoeba View: display the outline of modules after placement
-  Placement View: display the detailed placements of cells.





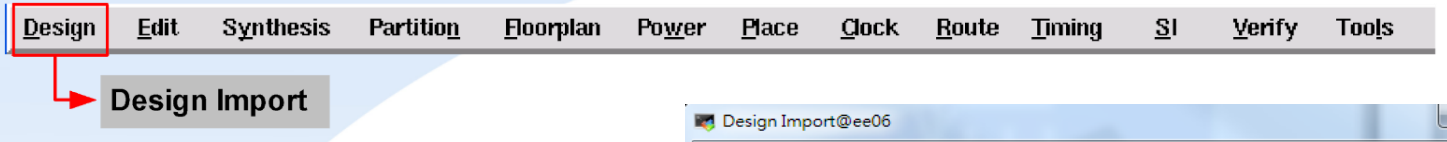


Set design view

- innovus #> setDrawMode fplan

2. Importing Design

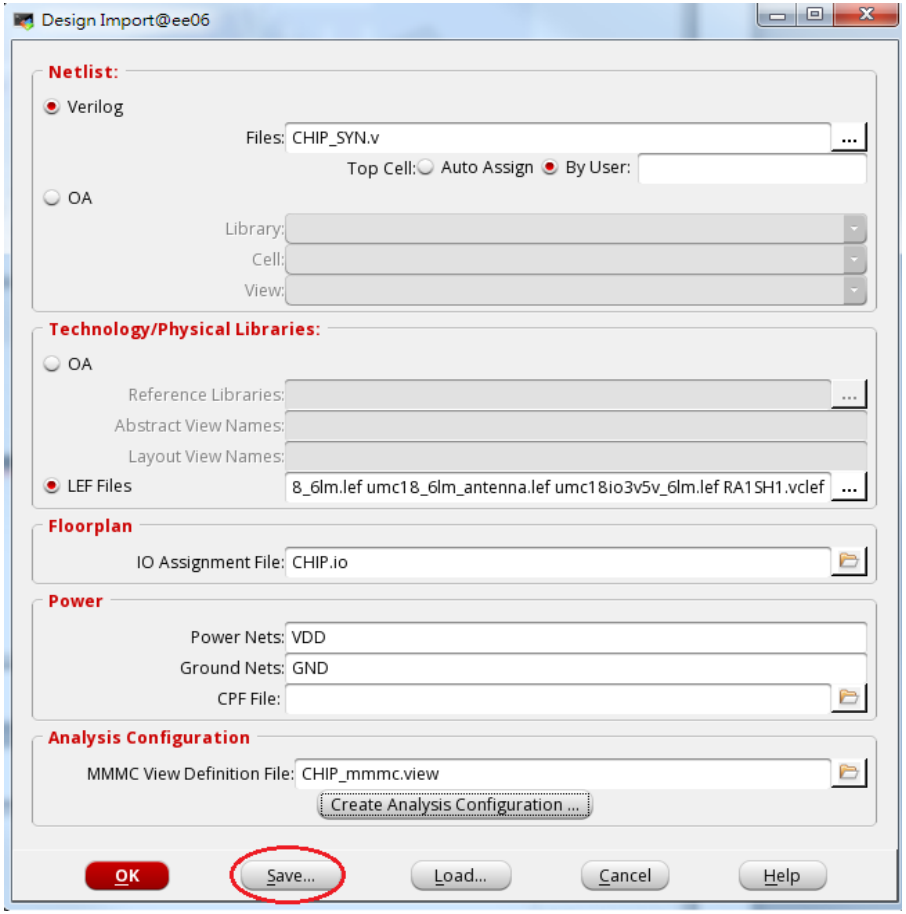
Import design files into Innovus environment



• Netlist:
By Verilog, and specifies the top cell by user

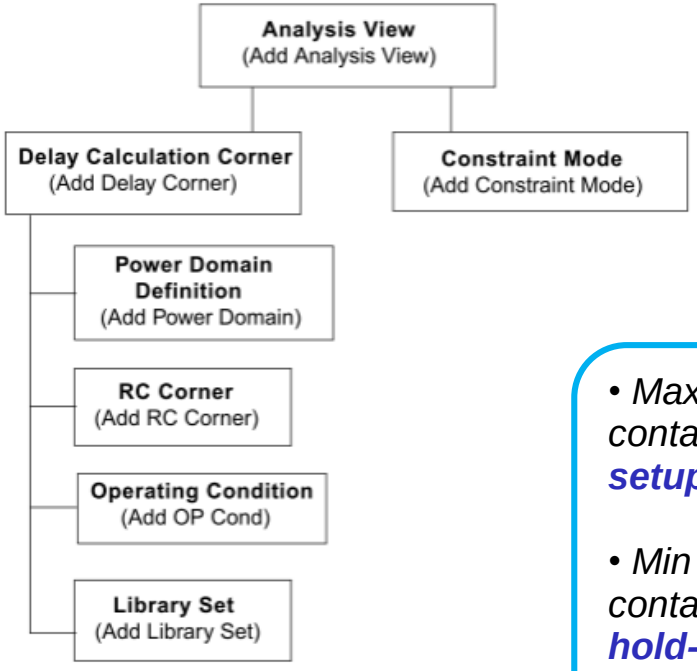
• Technology/Physical Libraries:
You must specify the technology LEF file first, then specify the standard cell LEF and block LEF. (Use LEF flow here)

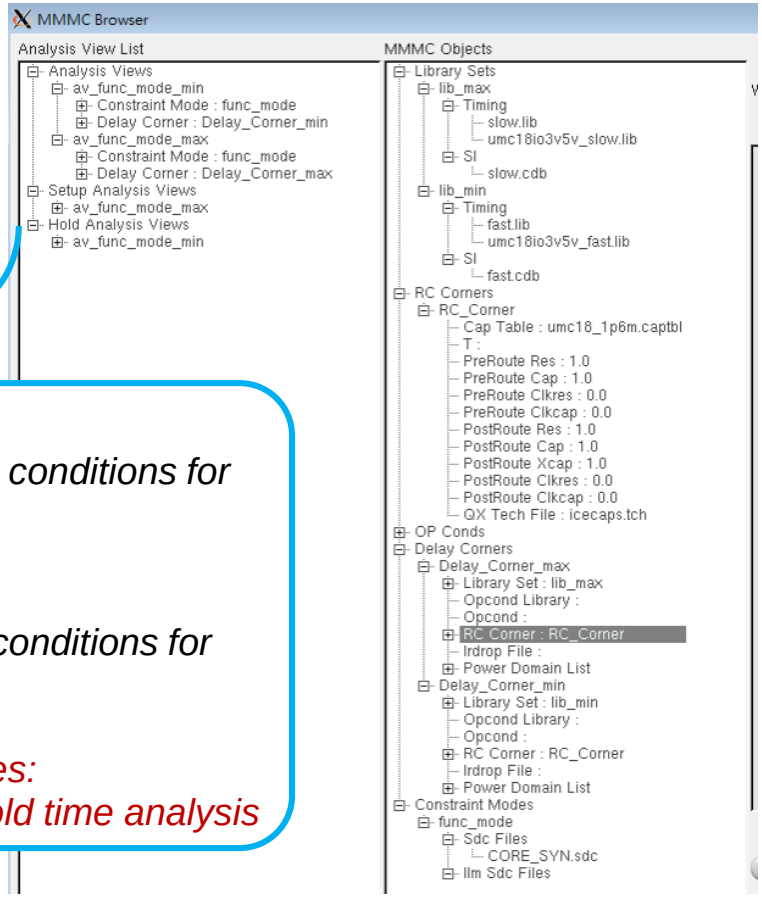
• Floorplan
Specifies the I/O files to import.



2. Importing Design – MMMC

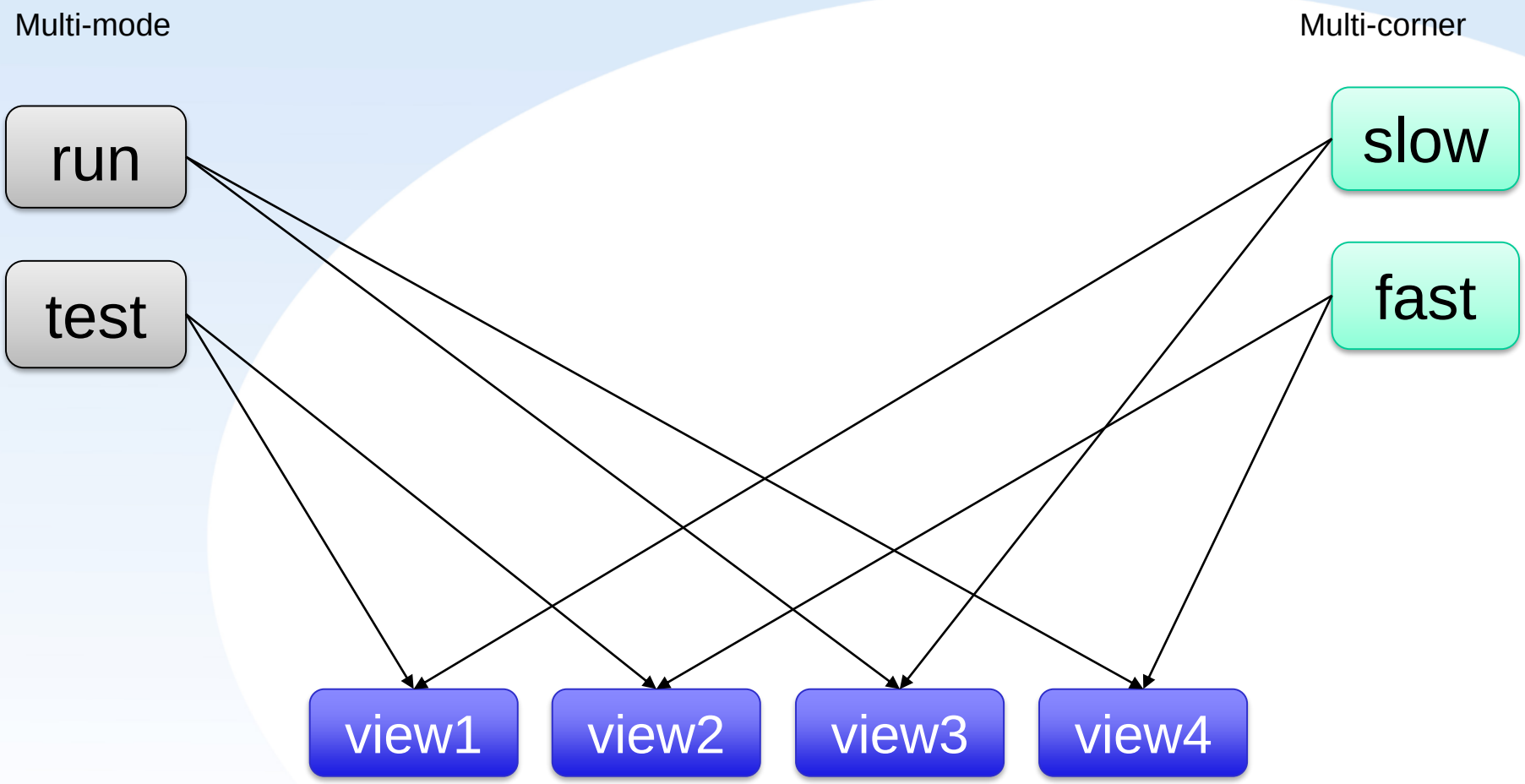
- **Multi-mode multi-corner (MMMC) timing analysis and optimization**
  - Press “Create Analysis Configuration” tab
  - In MMMC environment you must specify the lower-level information first.



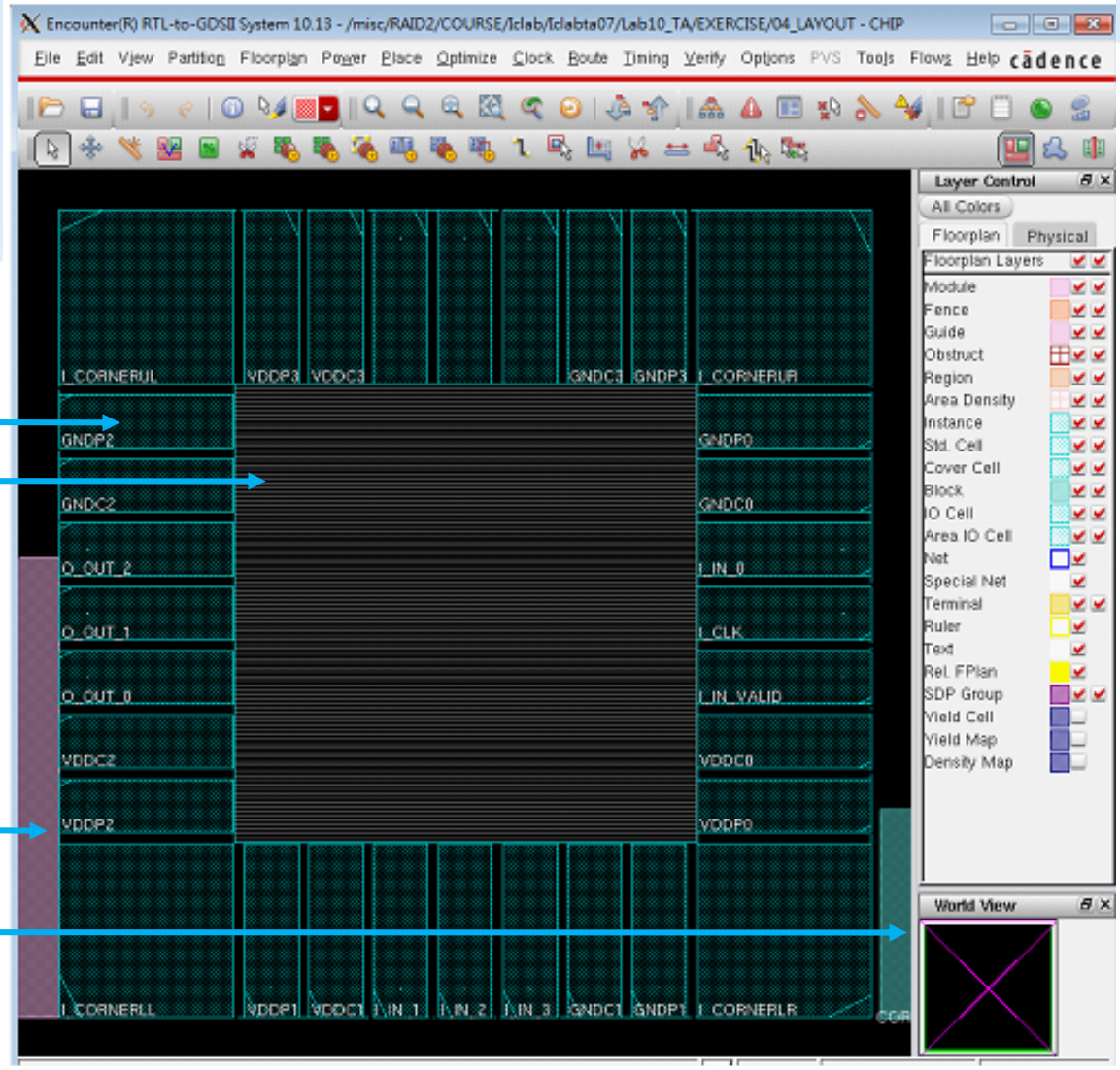


- *Max Timing Libraries:*
  containing the worst-case conditions for ***setup-time*** *analysis*
- *Min Timing Libraries:*
  containing the best-case conditions for ***hold-time*** *analysis*
- *Common Timing Libraries:*
  used in both setup and hold time analysis

2. Importing Design – MMMC



2. Importing Design - Result



3. Floor Planning – Intro.

Calculating core size, width and height

- When calculating core size of standard cells, the core utilization must be decided first. Usually the core utilization is higher than **70%**
- The core size can be calculated as follows

$$\text{Core Size of Standard Cell} = \frac{\text{standard cell area}}{\text{core utilization}}$$

- The recommended core shape is a square, i.e. Core Aspect Ratio = 1. Hence the width and height can be calculated as

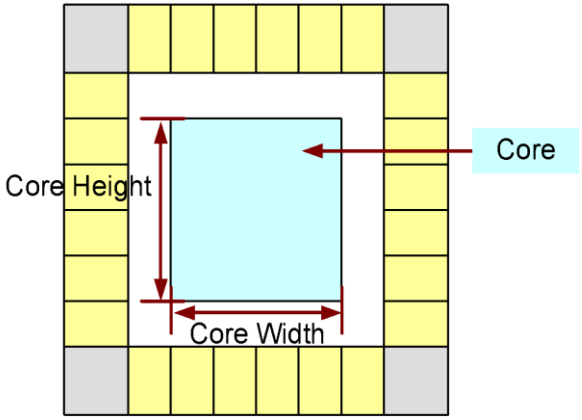
$$\text{Width} = \text{Height} = \sqrt{\text{Core Size of Standard Cell}}$$

  - Note: because stripes and macros will be added, width and height are usually set larger than the value calculated above

- E.g.
  - Standard cell area = 2,000,000
  - Core utilization demanded = 85%
  - No macros

$$\text{Core Size of Standard Cell} = \frac{2{,}000{,}000}{0.85} = 2{,}352{,}941$$

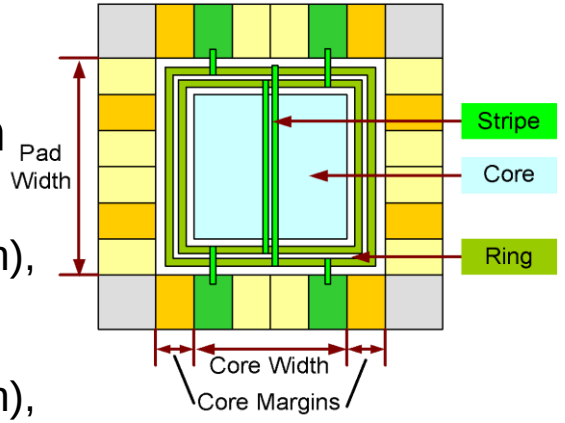
$$\text{Width} = \text{Height} = \sqrt{2{,}352{,}941} = 1534$$



3. Floor Planning

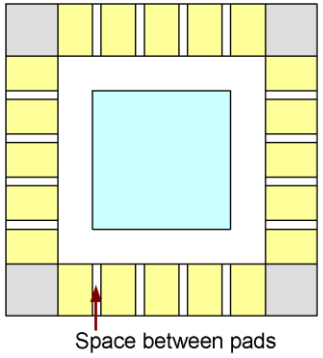
Core margins

- When setup the floorplan, remember to leave enough space for Power/Ground (P/G) ring
  - Note: the width needed for P/G ring will be discussed in power planning



Core limited design or Pad limited design

- Die size determination
  - When pad width > (core width + core margin), die size is decided by pads.
    And it is called pad limited design
  - When pad width < (core width + core margin), die size is decided by core.
    And it is called core limited design

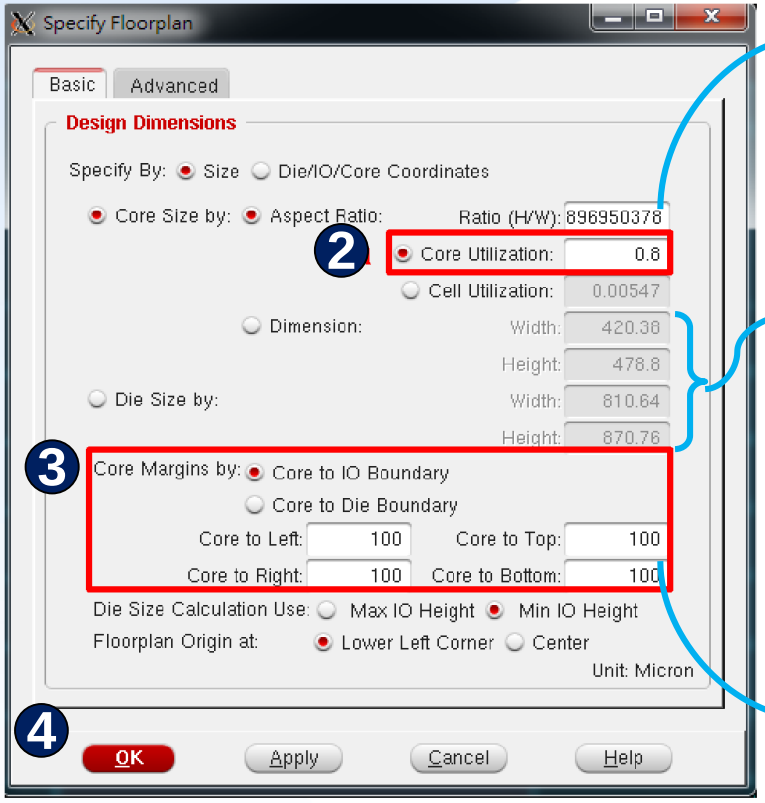


Core limited design

- Adding pad filler
  - Provide power to pad
  - There should be no spacing between pads.
    Pad filler is necessary for core limited design

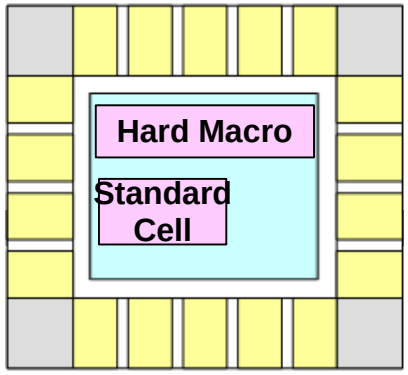
3. Floor Planning

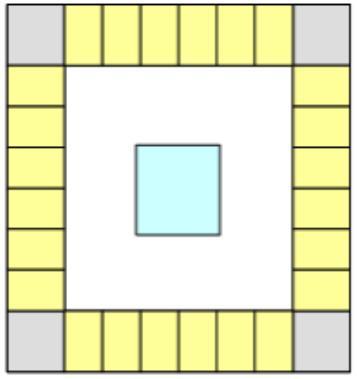
1 → Specify Floorplan



Some space are remained for routing.

Set a demanded value of width and height, when Hard Macro takes most of the place.





• **Core Limited Design** • **Pad Limited Design**

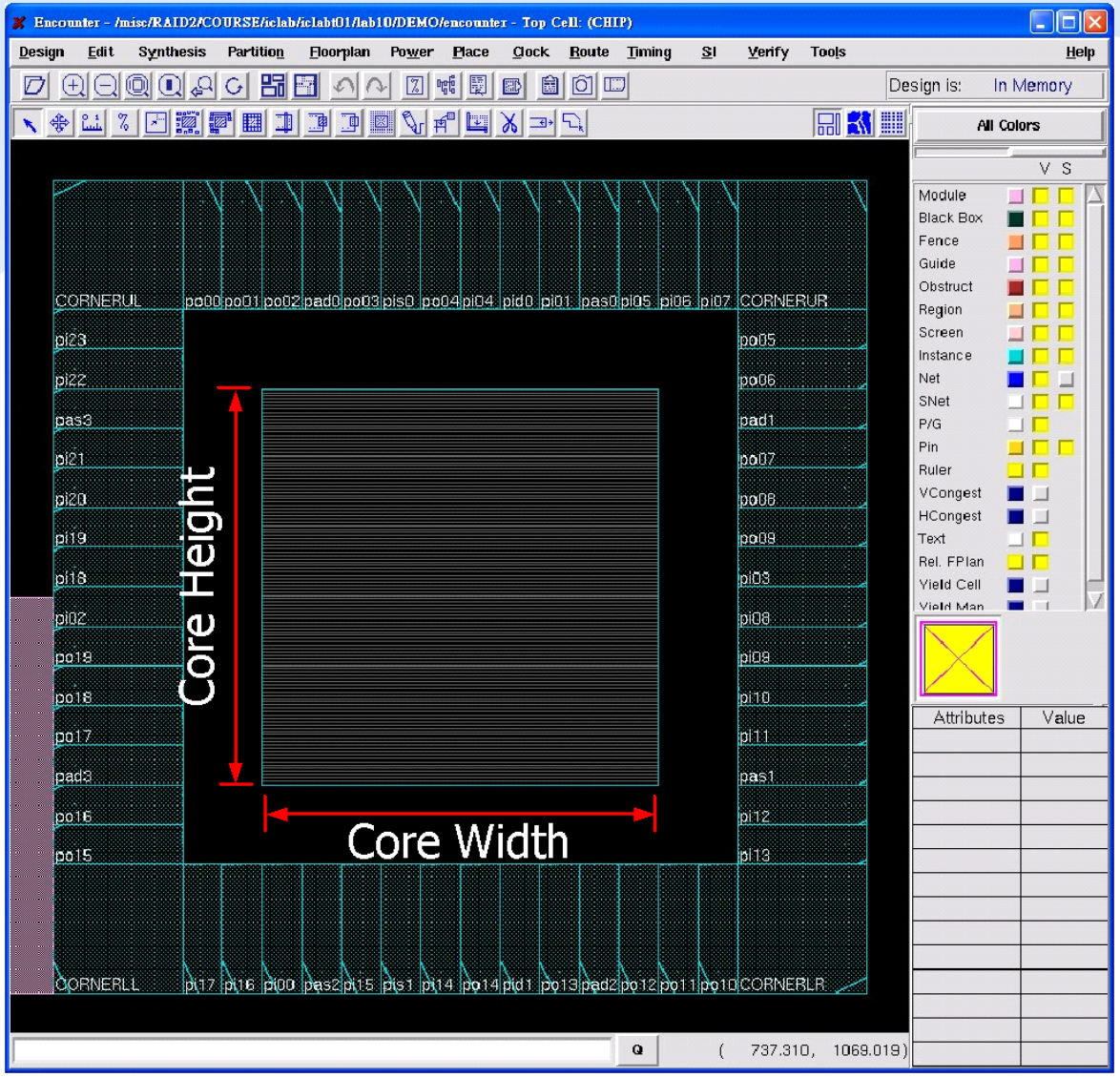
Some space are remained for power rings. (design dependent)

✔ Text Command (Specify core width, height and core margin)

– innovus #> floorPlan –s coreW coreH coreToLeft coreToBottomcoreToRight coreToTop

3. Floor Planning – Result

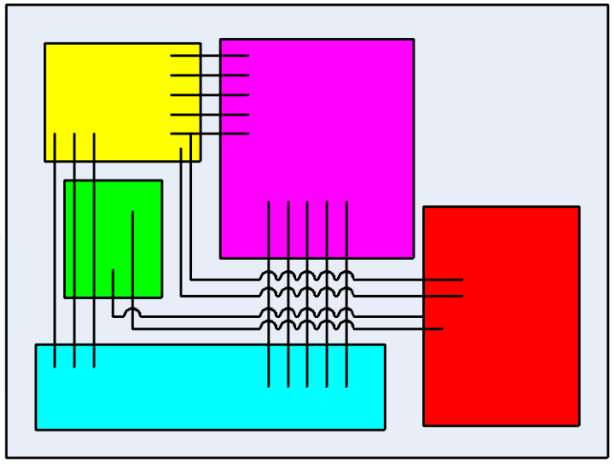
- ✔ **Result**



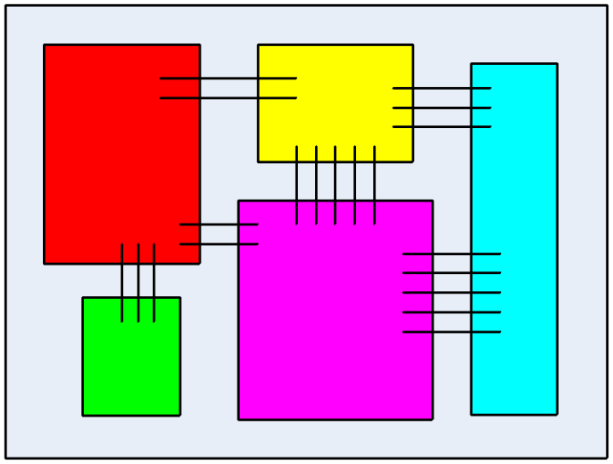
3. Floor Planning – Intro.

Floorplan Purposes

- Develop early physical layout to ensure design objective can be achieved
  - Minimum area for low cost
  - Minimum congestion for design routable
  - Estimate parasitic for delay calculation
  - Analysis power for reliability
- Gain early visibility into implementation issues
- Different floorplan, different performance



Bad floorplan

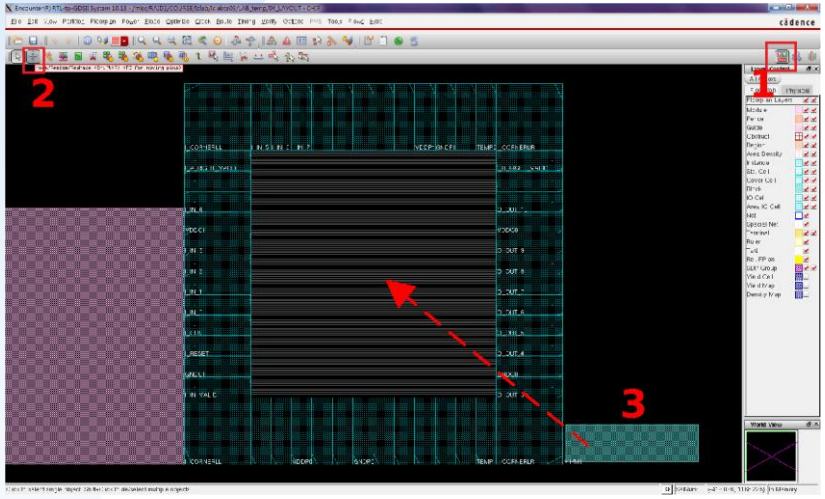


Good floorplan

3. Floor Planning with Hard Macro

Macro Floorplanning

- Move macro blocks to proper position
- Place macro around the corner to improve routability



Block place issue:

- ***power issue***
- ***noise issue***
- ***route issue***

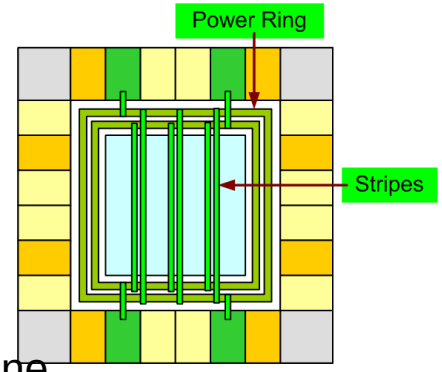
In practice, if there are many hard macro, try to use automatic floorplan and the following tools

- Floorplan → Relative Floorplan → Define array constraint
- Floorplan → Placement toolbox

4. Power Planning – Intro.

Power issue

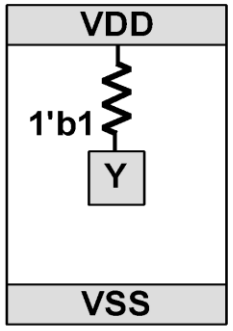
- Metal migration (also known as electro-migration)
  - Under **high currents**, electron collisions with metal grains cause the metal to move. The metal wire may be **open circuit** or **short circuit**.
  - Prevention: sizing power supply lines to ensure that the chip does not fail
  - Experience: make current density of power ring < **1mA/μm**
- IR drop
  - IR drop is the problem of voltage drop of the power and ground due to high current flowing through the power-ground resistive network
  - When there are excessive voltage drops in the power network or voltage rises in the ground network, the device will run at **slower speed**
  - IR drop can cause the chip to fail due to
    - Performance (circuit running slower than specification)
    - Functionality problem (setup or hold violations)
    - Unreliable operation (less noise margin)
    - Power consumption (leakage power)
    - Latch up
  - Prevention: adding stripes to avoid IR drop on cell's power line



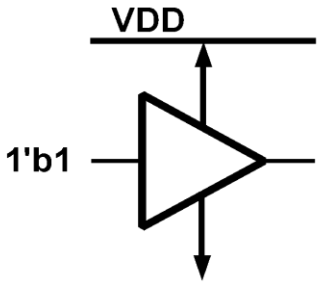
4.1. Power Planning - Connect/Define Global Nets

4.1. Connect/Define Global Nets

- Complete the form
  - Add power nets and pins connection list
  - Add ground nets and pins connection list
  - Tie High: Connect all the 1'b1 to VDD
  - Tie Low: Connect all the 1'b0 to GND

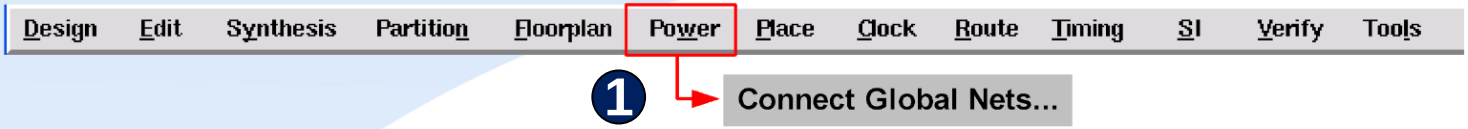


Tie High Cell



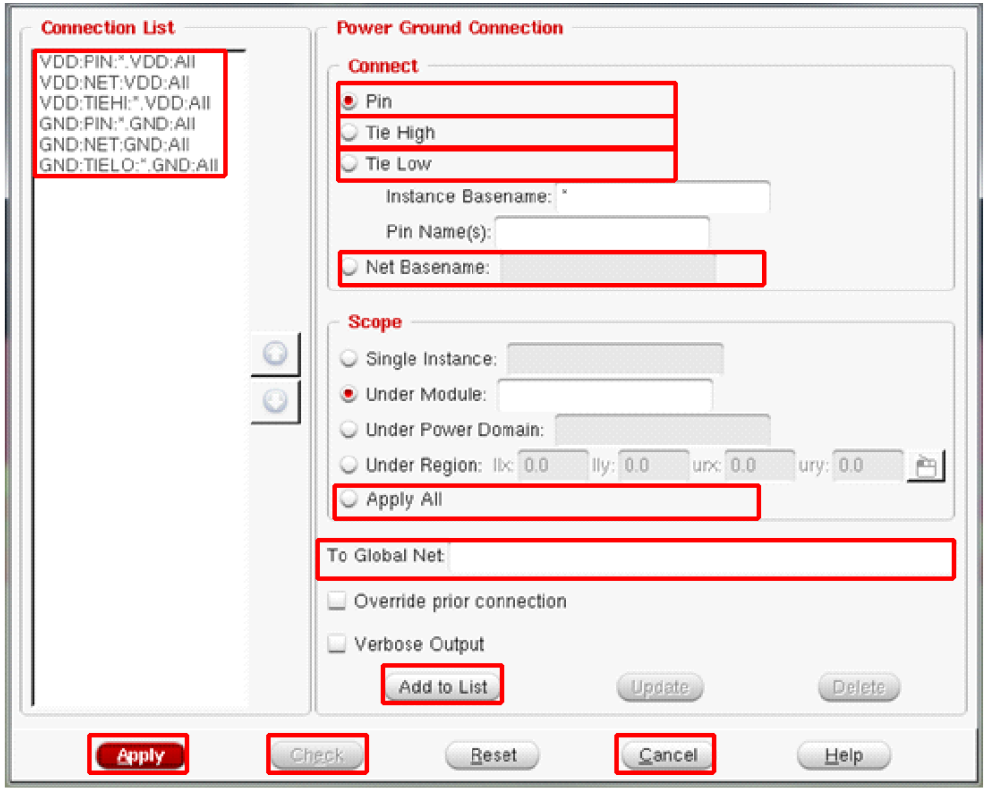
- After Applying, Only warnings of IO pads and padfiller are allowed
- Text Command
  - innovus #> clearGlobalNets
  - innovus #> globalNetConnect VDD -type pgpin -pin VDD -inst *
  - innovus #> globalNetConnect VDD -type net -net VDD
  - innovus #> globalNetConnect VDD -type tiehi -inst *
  - innovus #> globalNetConnect GND -type pgpin -pin GND -inst *
  - innovus #> globalNetConnect GND -type net -net GND
  - innovus #> globalNetConnect GND -type tielo -inst *
  - innovus #> globalNetConnect GND -type pgpin -pin VSS -inst *

4.1. Power Planning - Connect/Define Global Nets

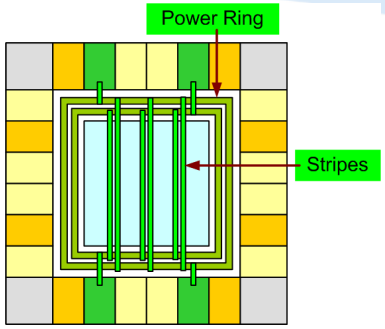


Create 6 global nets connection list

2. Connect ◆ Pins: *VDD*
   To Global Net: *VDD*
3. Connect ◆ Net Basename: *VDD*
   To Global Net: *VDD*
4. Connect ◆ Tie High
   To Global Net: *VDD*
5. Connect ◆ Pins: *GND*
   To Global Net: *GND*
6. Connect ◆ Net Basename: *GND*
   To Global Net: *GND*
7. Connect ◆ Tie Low
   To Global Net: *GND*



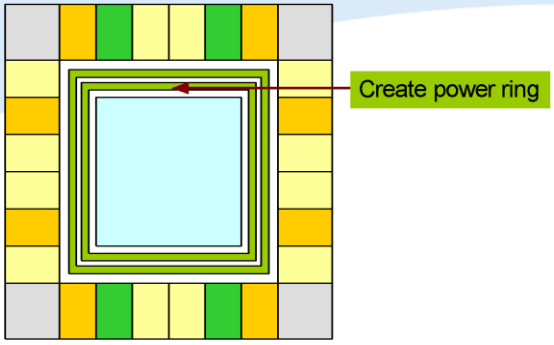
4. Power Planning – RING



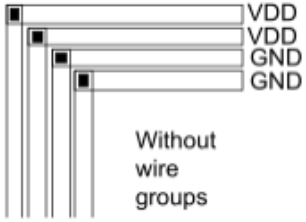
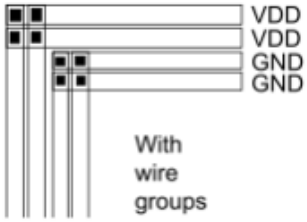
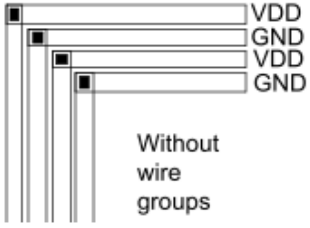
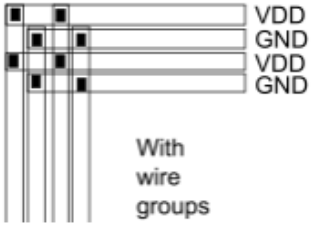
- **Calculate power/ground ring width**
  - Experience
    - Gate count = 70 k
    - 4000 Flip-Flops
    - 80% FF with dynamic gated clock
    - Current needed = 0.2mA/MHz
      - Note: the value should multiply with 1.8~2 for no gated design
  - Example:
    - Gate count = 200 k
    - No gated clock
    - Clock frequency = 20 MHz
    - Current needed = (200k/70k) * 0.2 mA/MHz * 20MHz * 2 = 22.86 mA
    - Current density < 1mA/μm
    - The Width of P/G Ring > 22.86 μm
    - In order to avoid the slot rule of wide metal, the largest width is 20 μm (process dependent)
    - Use two set P/G ring for this case

4.2. Power Planning - Core Power Ring

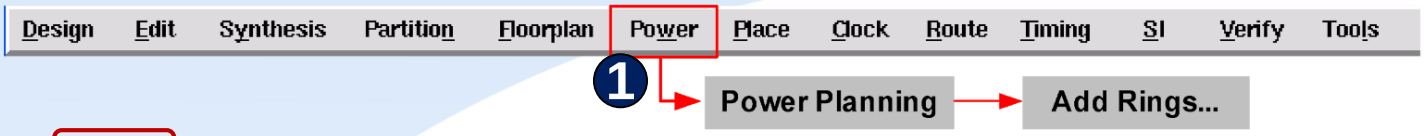
4.2. Core Power Ring



– Interleaving and Wire group

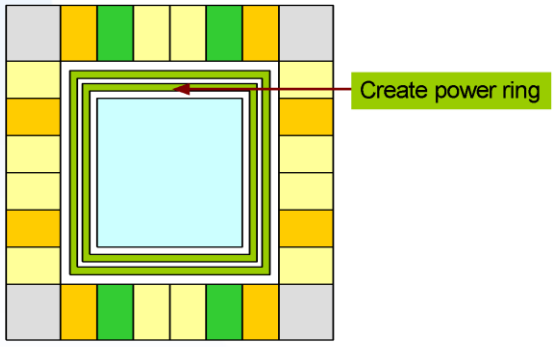
	Without Wire group	With Wire group
Without Interleaving		
Interleaving		

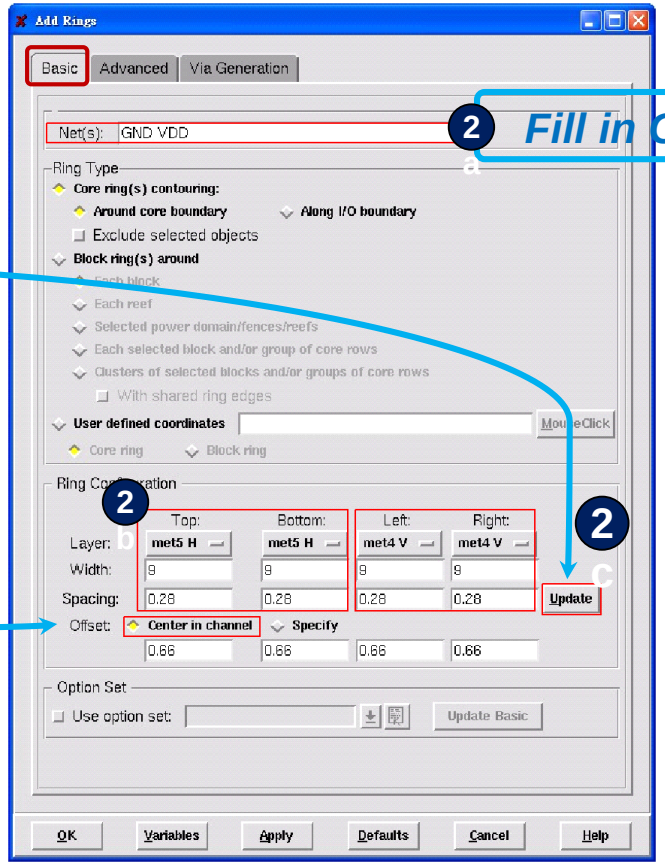
4.2. Power Planning - Core Power Ring



– In Basic tab

Field	Fill In
Top and Bottom Layer	Metal 3
Left and Right Layer	Metal 4
Width	All 9 (or the demanded value)
	Click Update
Offset	◆ Center in channel





4.2. Power Planning - Core Power Ring

- In Advanced tab
  - You can create one more set of power ring

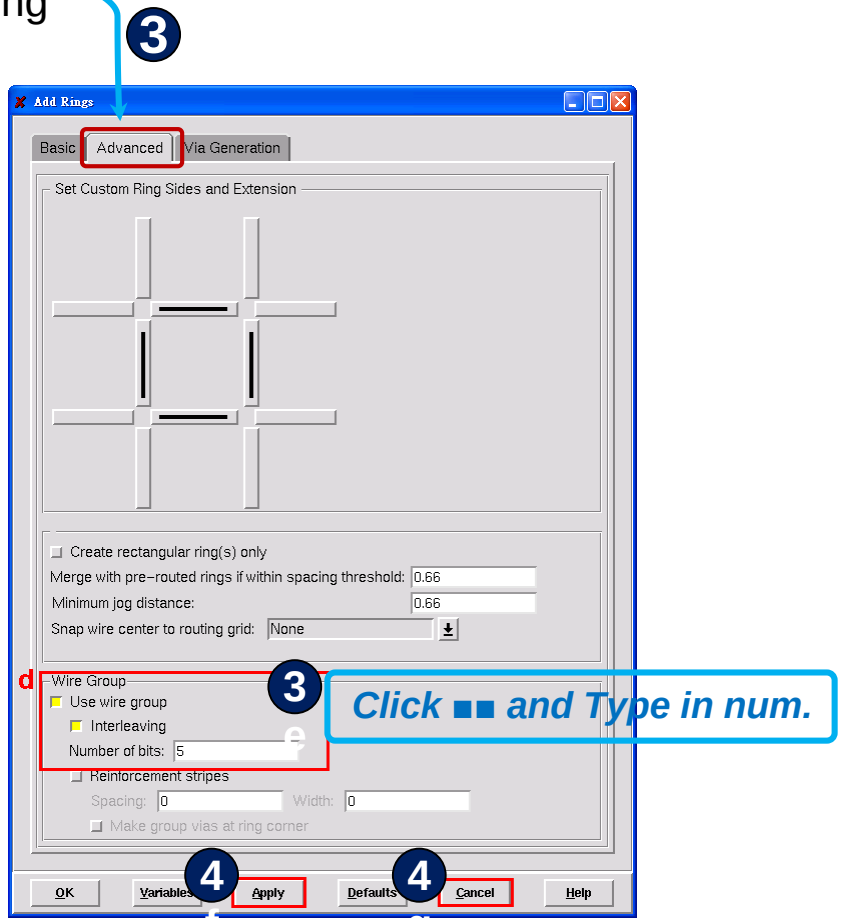
Field	Fill In
Wire Group	◆ Use wire group ◆ Use wire group
Number of bits	The demanded value

Note:

you can choose the option “Interleaving” to observe the difference in power ring created

  - Click Apply

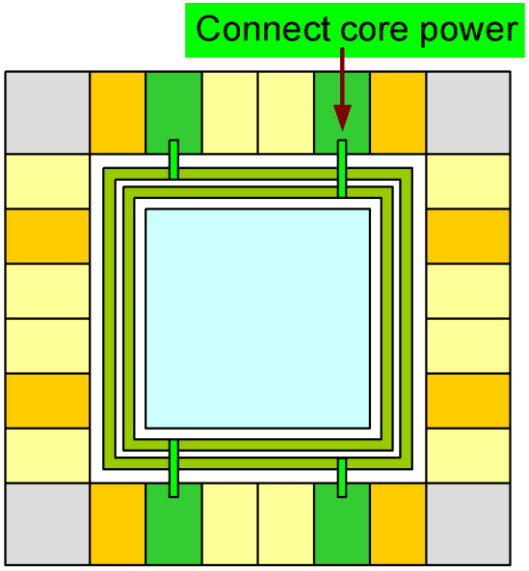
- Tips
  - Click Apply then you can undo this step
  - Click OK then undo is not allowed



4.3. Power Planning - Core Power Pin

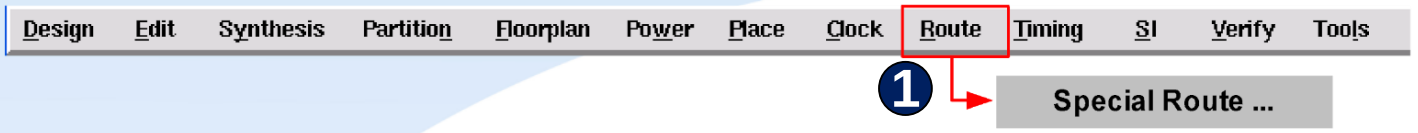
4.3. Core Power Pin

- Connect from core power pads to power rings



- Note:
  If global nets aren't correctly connected, core power pins will have trouble in generating.

4.3. Power Planning - Core Power Pin

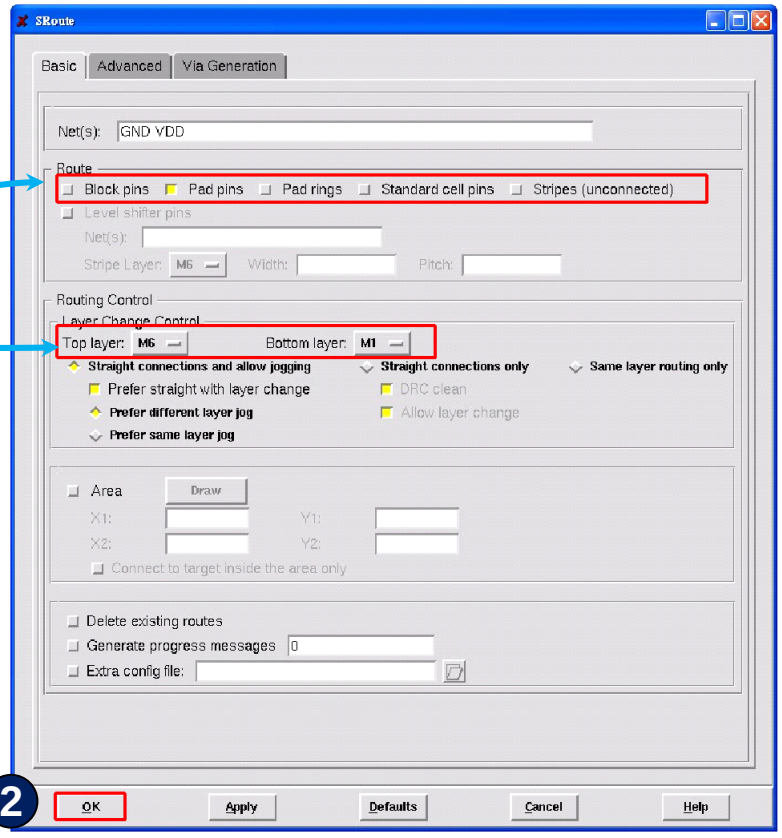


✔ Connect core power pin

- Complete the form and click Apply

Field	Fill In
Block pins	◇ (off)
Pad pins	◆ (on)
Pad rings	◇ (off)
Standard cell pins	◇ (off)
Stripes (unconnected)	◇ (off)

2



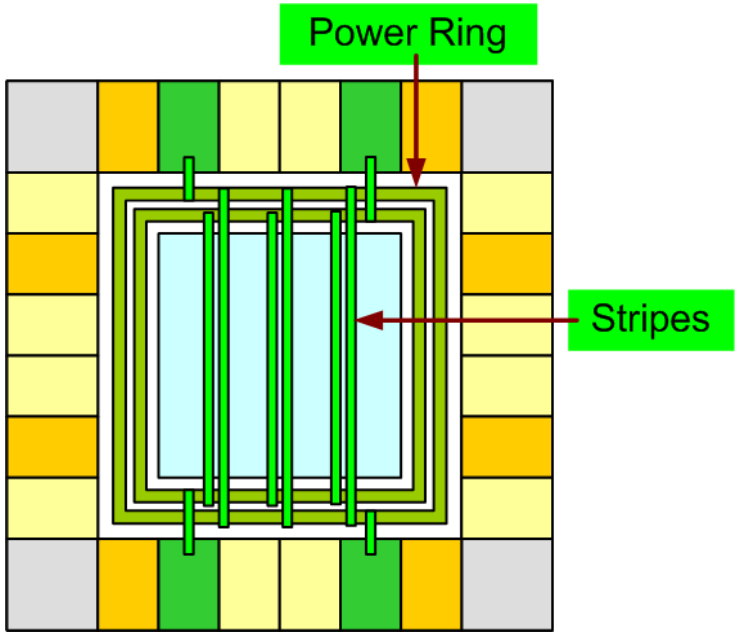
Choose M2 for Bottom layer (Suggestion)

Note:
Make sure that core power pads are connected to the power ring.
Adjust the top and bottom layer if needed.

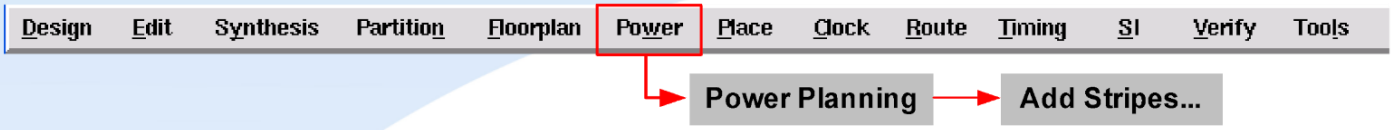
2

4. Power Planning – Stripes

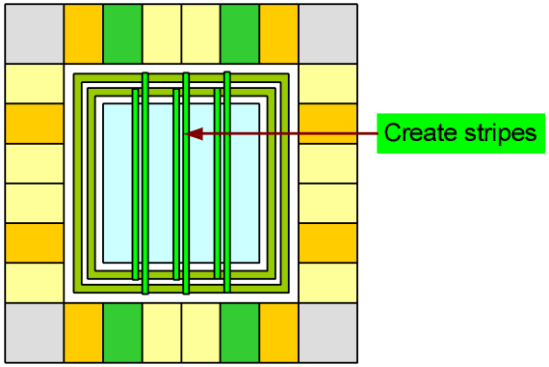
- **Calculate stripe set**
  - Experience
    - Add one stripe set per 100 μm
  - Example
    - Core width = height = 1600
    - Stripe set added = 15

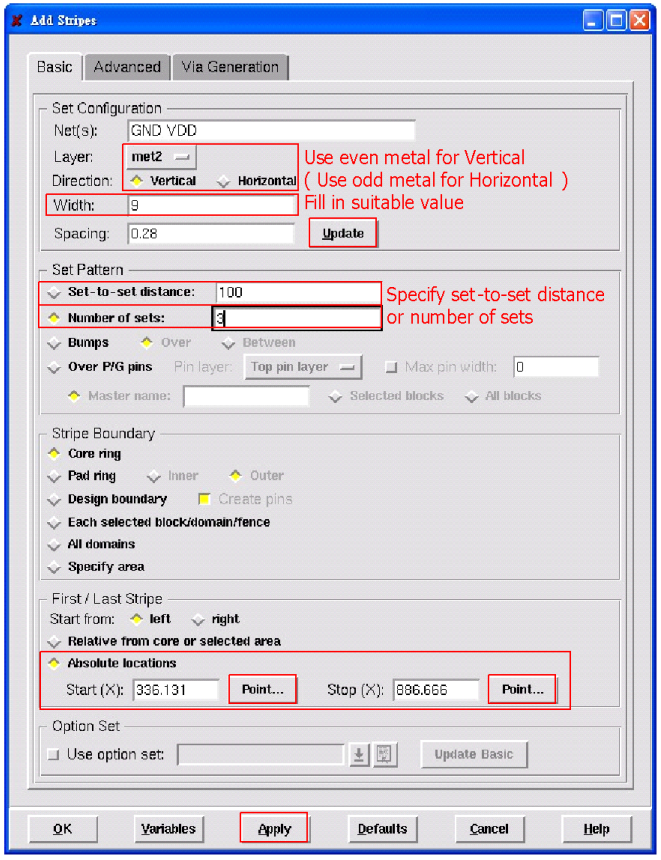


4.4. Power Planning - Stripes



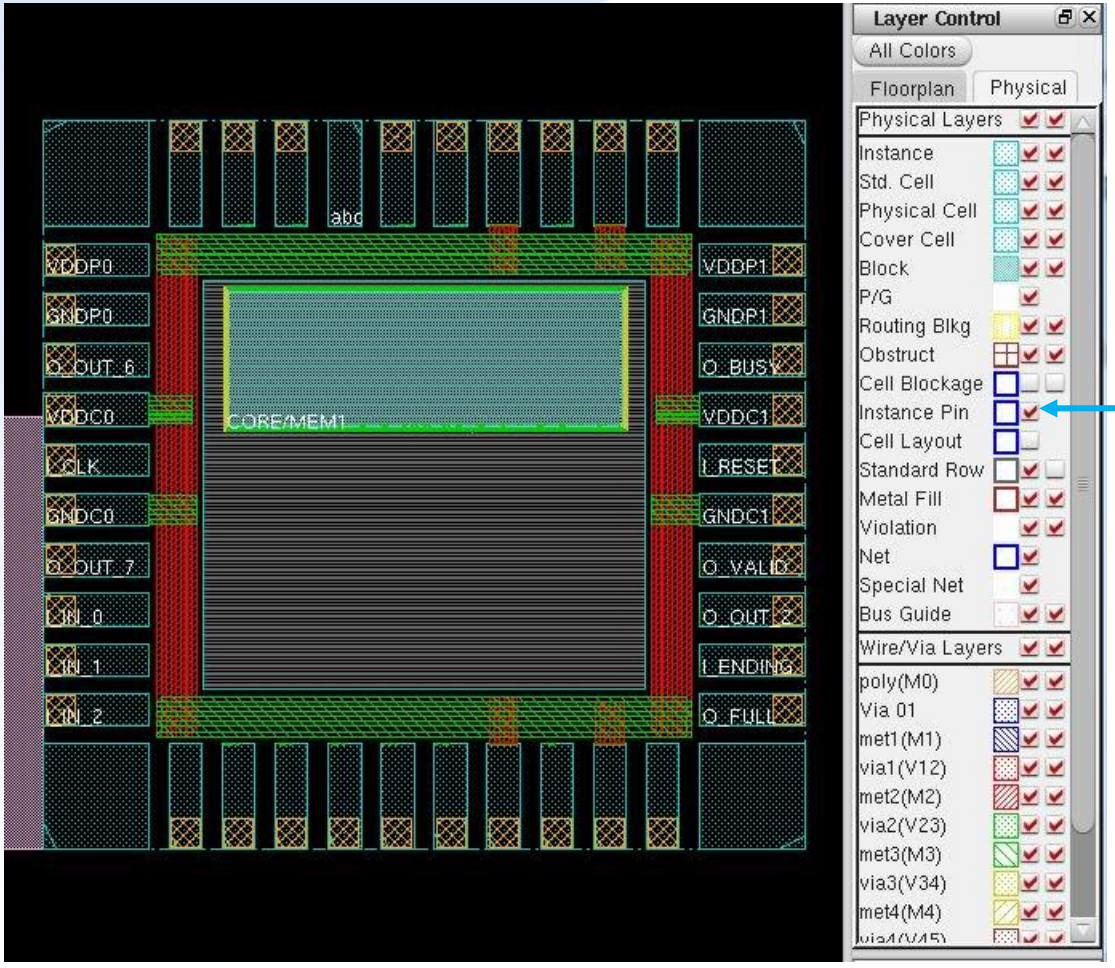
– Complete the form in Basic tab and click Apply





Note: after click point
Start (X): click the location of first stripe in design display area
Stop (X): click the location of last stripe in design display area
Note: Specify Start (Y) and Stop (Y) for horizontal stripes

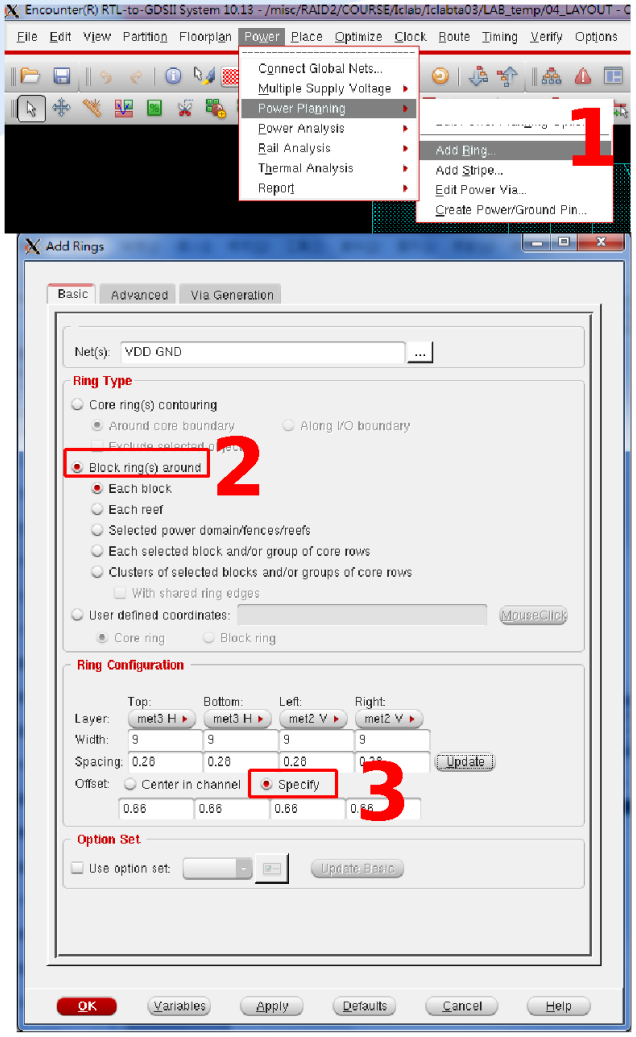
4. Power Planning – Block Ring for Hard Macro

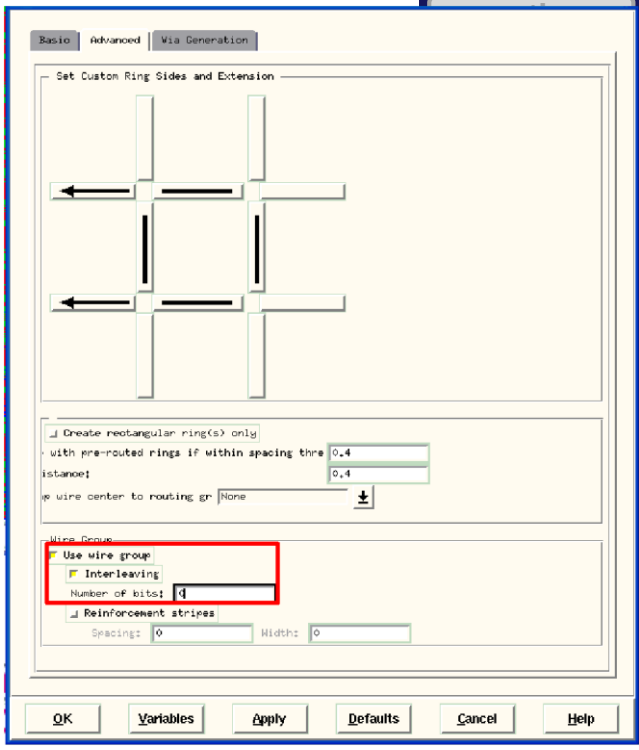


To visualize the block ring of SRAM

4. Power Planning – Block Ring for Hard Macro

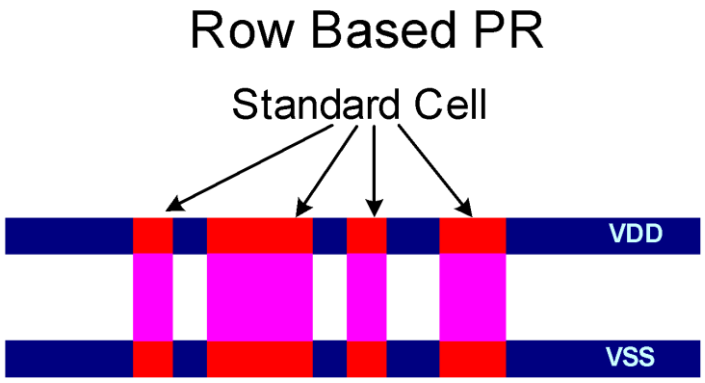
- **Add block ring**
  - Specify nets
  - Block ring around
  - Specify metals
  - Width, offset
  - Extension
  - Wiregroups





4.5. Standard Cell Power Connection

- **Connect standard cell power line**

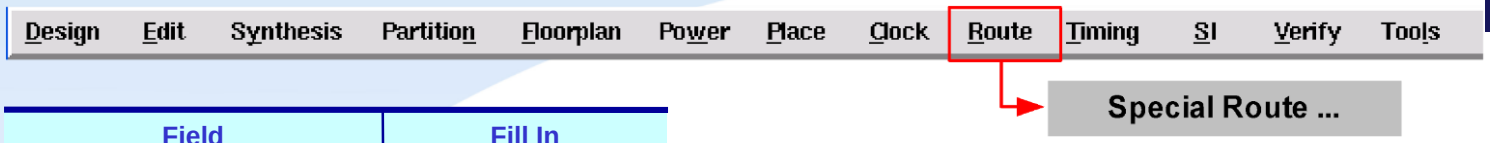


– Text Command

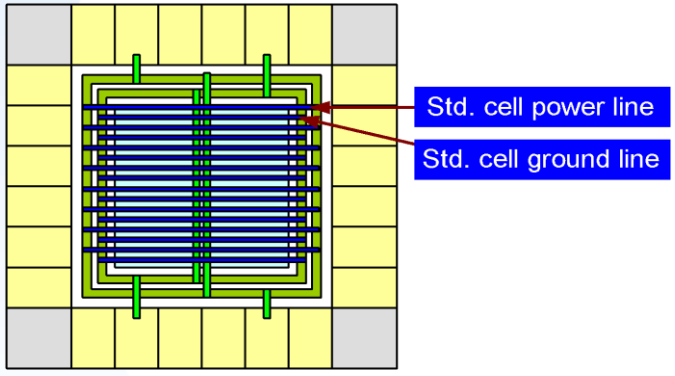
innovus #> sroute –noBlockPins –noPadRings –noPadPins –noStripes –jogControl
{ preferWithChanges differentLayer }

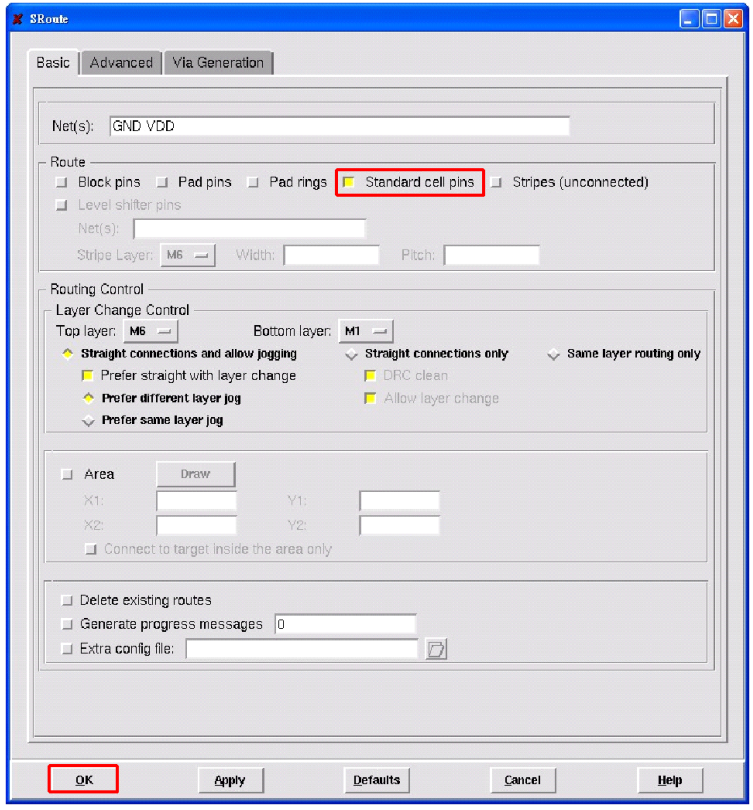
4.5. Standard Cell Power Connection

Connect standard cell power line



Field	Fill In
Block pins	◇ (off)
Pad pins	◇ (off)
Pad rings	◇ (off)
Standard cell pins	◆ (on)
Stripes (unconnected)	◇ (off)



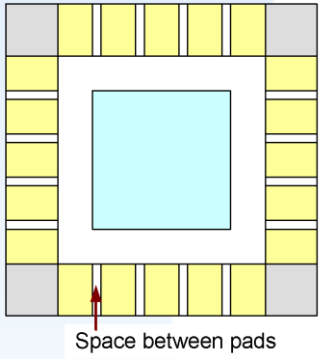


– Text Command

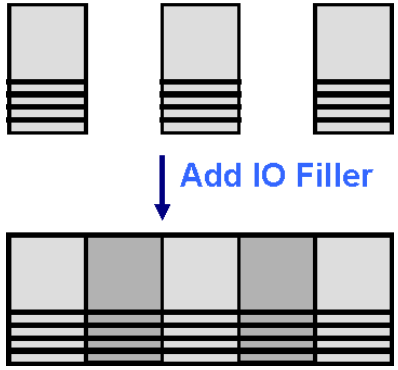
innovus #> sroute –noBlockPins –noPadRings –noPadPins –noStripes –jogControl
{ preferWithChanges differentLayer }

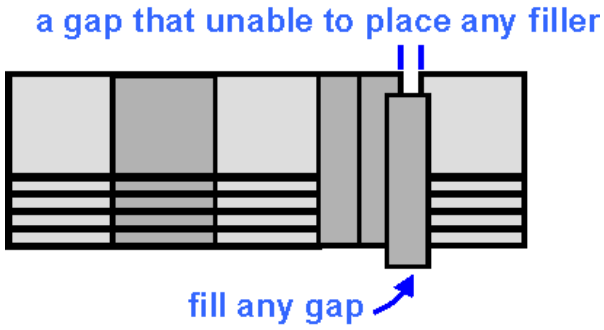
4.6 Add PAD Filler

- **Adding pad filler**
  - There should be no spacing between pads.
    Therefore adding pad filler is necessary for core limited design
  - PAD filler must be added before detail route, or there may have some DRC/LVS violations after PAD filler insertion



Core limited design





  - Text Command

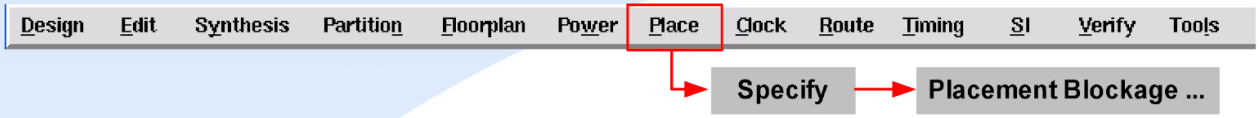
From wider fillers to narrower ones

- innovus #> addIoFiller -cell PFILL -prefix IOFILLER
- innovus #> addIoFiller -cell PFILL_9 -prefix IOFILLER
- innovus #> addIoFiller -cell PFILL_1 -prefix IOFILLER
- innovus #> addIoFiller -cell PFILL_01 -prefix IOFILLER –fillAnyGap

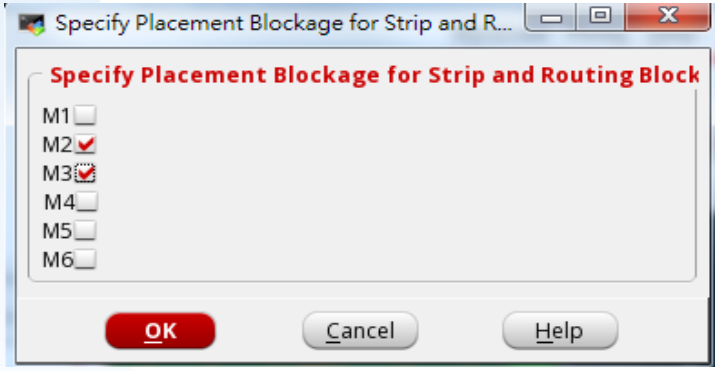
Only the smallest filler can use –fillAnyGap option

5. Standard Cell Placement

Specify placement blockage



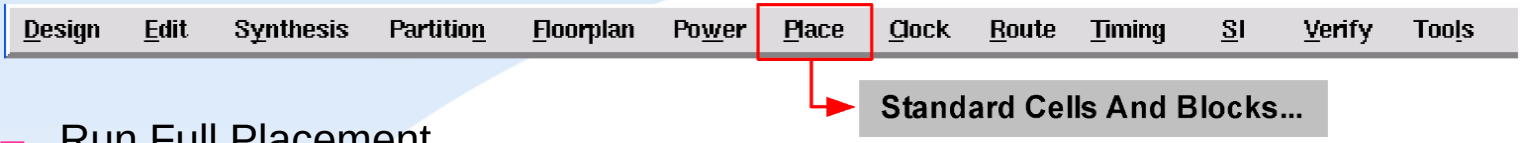
- Choose the metal layers of stripe.
  Then there will be no cell placed under stripe
- Text Command
  innovus #> setPrerouteAsObs {2 3}



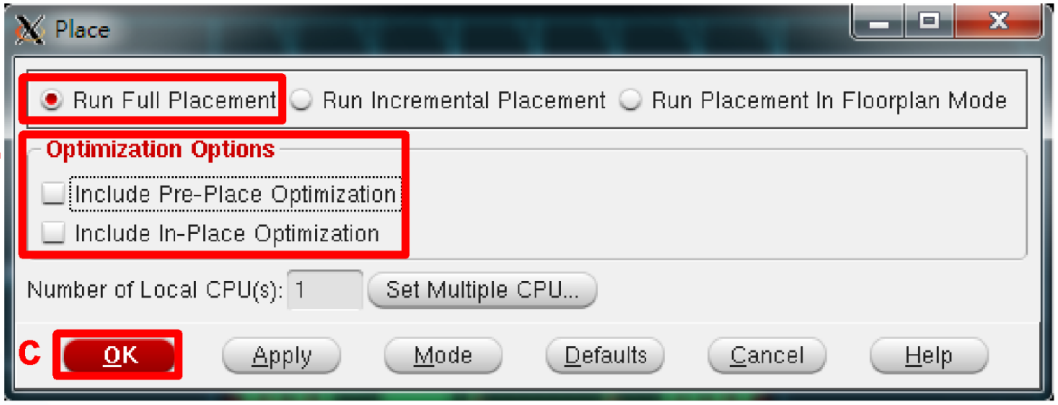
- Add placement blockage for macros
- Text Command
  innovus #> createObstruct *llx lly urx ury*

5. Standard Cell Placement

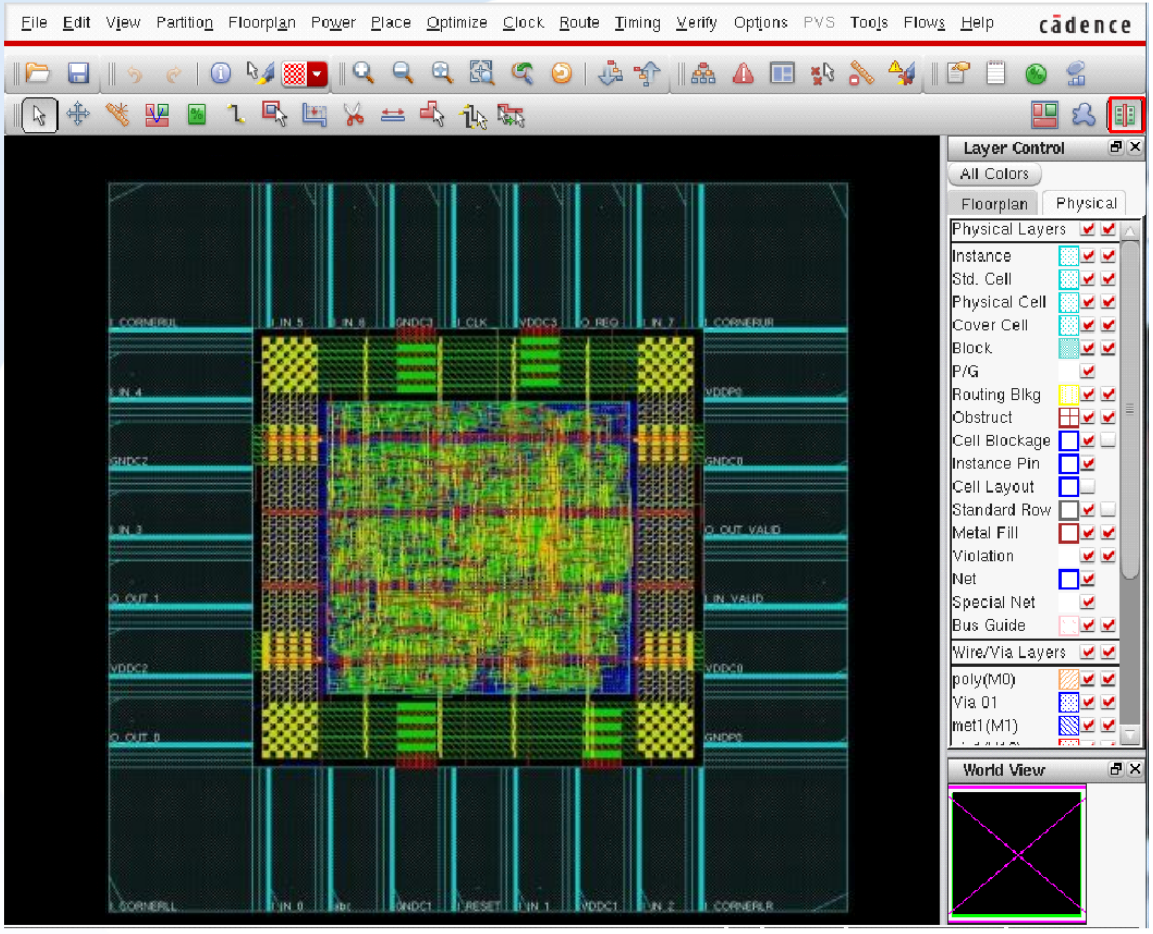
Place standard cells



- Run Full Placement
- Optimization Options
  - ☐ Include Pre-Place Optimization
  - ☐ Include In-Place Optimization
- Click OK button



5. Standard Cell Placement – Result



6. Pre-CTS Timing Analysis

- **To check timing**
  - innovus #> timeDesign –preCTS

- **Operations**
  - Trial route
  - RC extraction
  - Timing analysis
  - Generates reports are saved in ./timingReports/ , including
    - _preCTS.cap
    - _preCTS.fanout
    - _preCTS.tran
    - _preCTS_all.tarpt

- **Reports**
  - Congestion distribution
  - Timing Summary
  - If the timing is met, pre-CTS optimization can be skipped

Congestion distribution:

Remain	cntH		cntV	
-2:	5	0.02%	0	0.00%
-1:	10	0.03%	0	0.00%
1:	10	0.03%	21	0.06%
2:	10	0.03%	58	0.17%
3:	0	0.00%	88	0.25%
4:	33	0.10%	379	1.09%
5:	0	0.00%	4227	12.18%

Be careful of negative congestion

Setup mode	all	reg2reg	in2reg	reg2out	in2out	clkgate
WNS (ns):	-0.744	N/A	-0.744	N/A	N/A	N/A
TNS (ns):	-7.291	N/A	-7.291	N/A	N/A	N/A
Violating Paths:	16	N/A	16	N/A	N/A	N/A
All Paths:	40	N/A	40	N/A	N/A	N/A

Negative slack should be fixed

Density: 19.855%
Routing Overflow: 0.00% H and 0.00% V
Real DRV (fanout, cap, tran): (0, 1, 0)
Total DRV (fanout, cap, tran): (0, 1, 0)

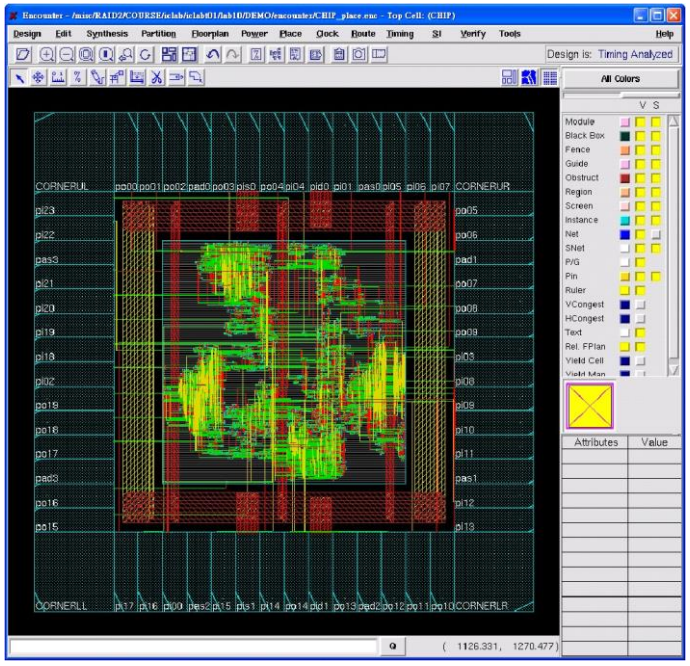
Total DRV = real DRV + clock network DRV

6. Pre-CTS Optimization

- **To optimize timing placed design for the first time with ideal clocks**
  - innovus #> optDesign –preCTS
- **To further optimize a design after above command execution**
  - innovus #> optDesign –preCTS –incr
- **Operation**
  - Repair Setup slack, Setup times
  - Repair Design rule violations (DRVs) and remaining DRVs

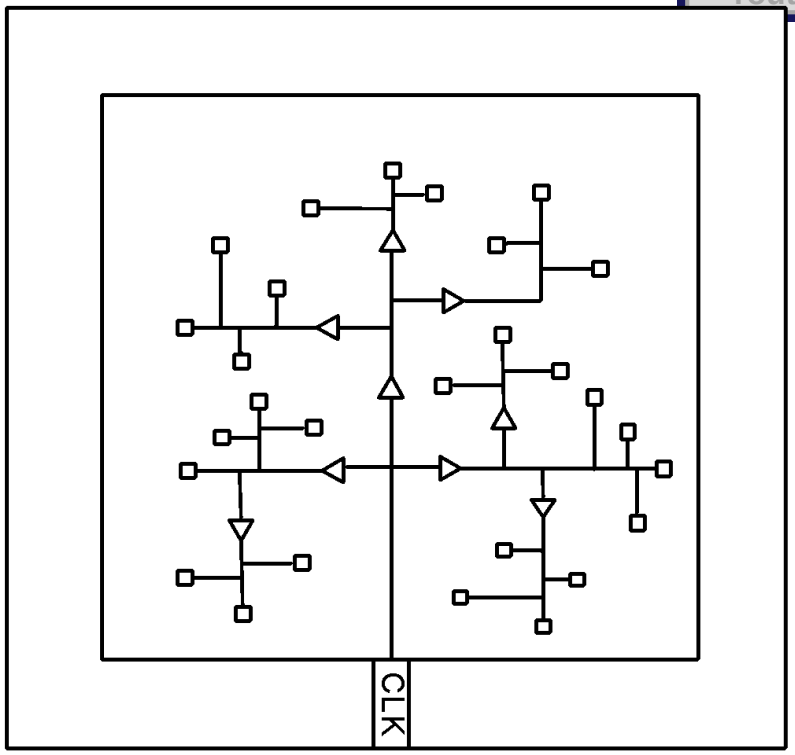
Setup mode	all	reg2reg	in2reg	reg2out	in2out	clkgate
WNS (ns):	0.102	N/A	0.102	N/A	N/A	N/A
TNS (ns):	0.000	N/A	0.000	N/A	N/A	N/A
Violating Paths:	0	N/A	0	N/A	N/A	N/A
All Paths:	40	N/A	40	N/A	N/A	N/A

Density: 20.928%
Routing Overflow: 0.00% H and 0.00% V
Real DRV (fanout, cap, tran): (0, 0, 0)
Total DRV (fanout, cap, tran): (0, 0, 0)



7. Clock Tree Synthesis - Clock Problem

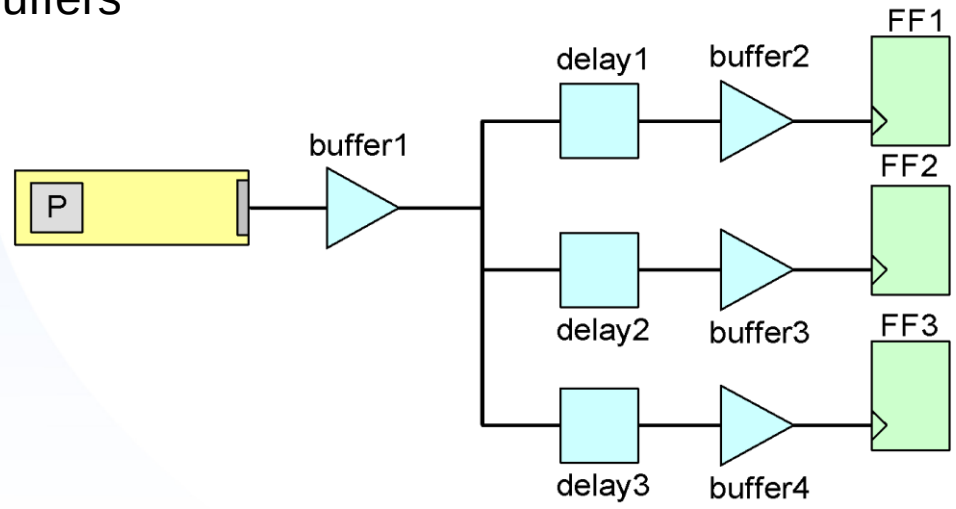
- **Clock problem**
  - Heavy clock net loading
  - Long clock insertion delay
  - Clock skew
  - Skew across clocks
  - Clock to signal coupling effect
  - Clock is power hungry
  - Electromigration on clock net



- **Clock is one of the most important treasure in a chip, do not take it as other use.**

7. Clock Tree Synthesis

- **The goal of clock tree synthesis includes**
  - Creating clock tree spec file
  - Building a buffer distribution network
  - Routing clock nets using CTS-NanoRoute
- **In automatic CTS mode, Innovus will do the following things**
  - Build the clock buffer tree according to the clock tree specification file
  - Balance the clock phase delay with appropriately sized, inserted clock buffers



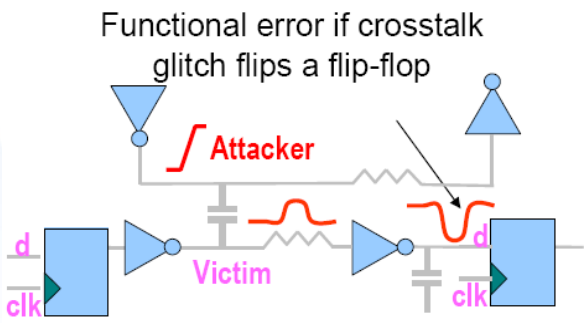
8. Post-CTS Timing Analysis - Text Command

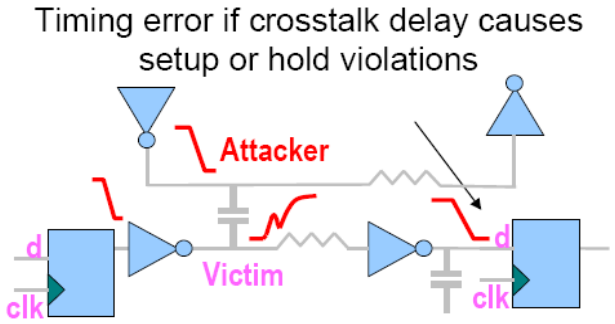
- **To check setup time**
  - innovus #> timeDesign –postCTS
- **To check hold time**
  - innovus #> timeDesign –postCTS –hold
- **Reports**
  - Generated timing reports are saved in ./timingReports/, including _postCTS.cap , _postCTS.fanout , _postCTS.tran , _postCTS_all.tarpt”
- **To correct setup violations and design rule violation**
  - innovus #> optDesign –postCTS
- **To correct hold violations**
  - innovus #> optDesign –postCTS –hold
- **Operation**
  - Repair Setup slack, Setup times
  - Repair Design rule violations (DRVs)

10. NanoRoute – Intro.

Signal Integrity (SI) Issue

- Crosstalk
- Charge sharing
- Supply noise
- Leakage
- Propagated noise
- Overshoot
- Under shoot



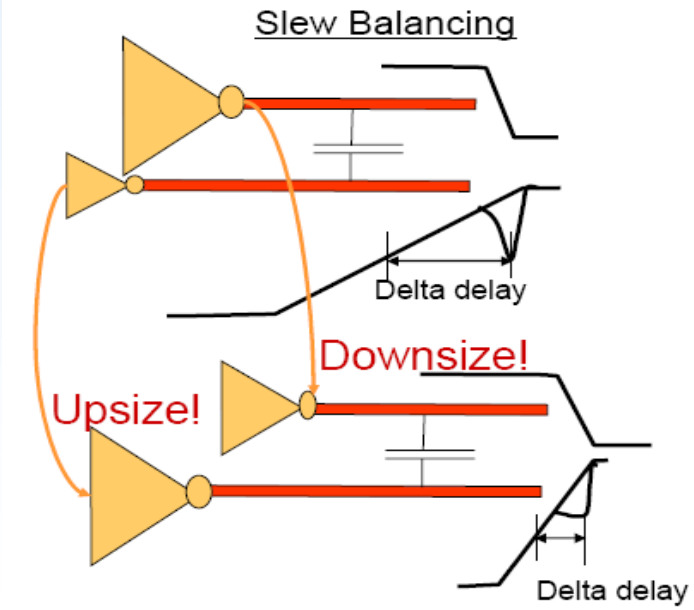


SI closure

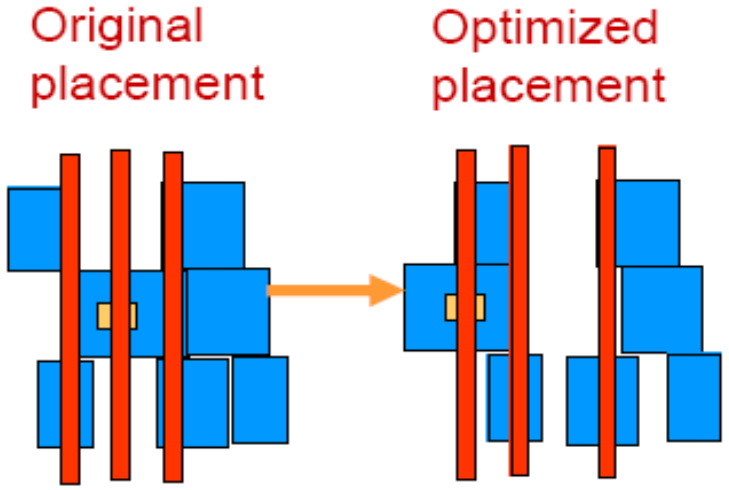
- Occur when a design is free from SI induced functional glitch and timing failure

Appendix: Crosstalk

- **Crosstalk prevention – Placement solution**
  - Insert buffer in lines
  - Upsize driver
  - Congestion optimization
- **Crosstalk prevention – Routing solution**
  - Limit length of parallel nets
  - Wider routing grid
  - Shield special nets



• Reduce coupling capacitance

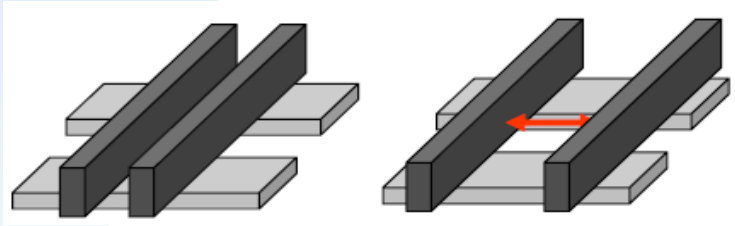


10. NanoRoute – Intro.

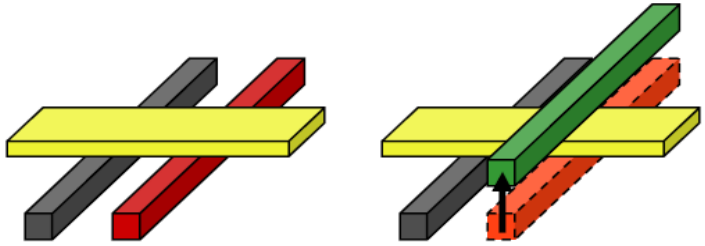
SI prevention (cont.)

- Routing-based SI prevention

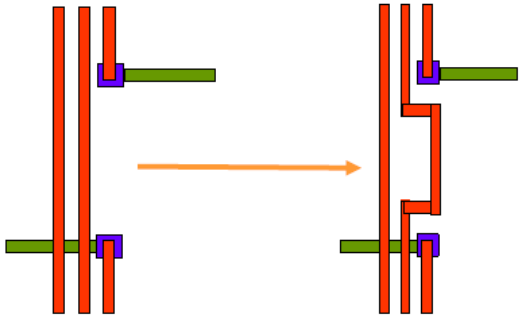
• Wiring Spacing



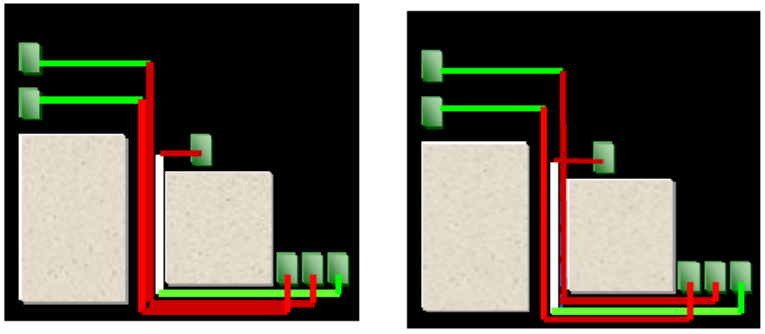
• Layer Switching



• Parallel Wires Reducing



• Net Re-ordering

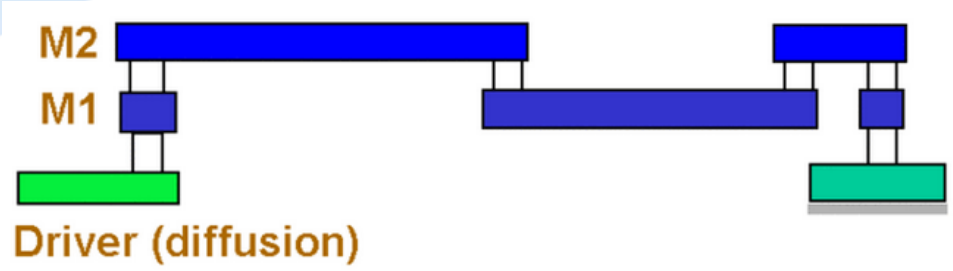


10. NanoRoute – Process Antenna Effect

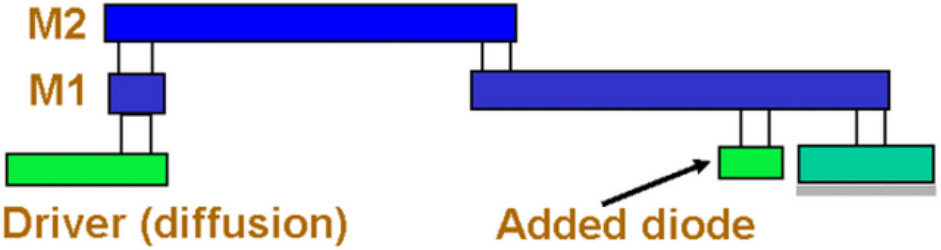
- **Antenna Effect**
  - In a chip manufacturing process, metal is initially deposited so it covers the entire chip
  - Then, the unneeded portions of the metal are removed by etching, typically in plasma (charged particles).
  - The exposed metal collect charge from plasma and form voltage potential.
  - If the voltage potential across the gate oxide becomes large enough, the current can damage the gate oxide.
- **Antenna Ratio =** $\dfrac{\text{Area of process antennas on a node}}{\text{Area of gates to the node}}$
- **Problem Repair**
  - Add jumper (changer metal layer)
  - Add antenna cell (diode)
  - Add dummy transistor

10. NanoRoute – Process Antenna Effect

- Add jumper



- Add diode



- If after fixing antenna violation, there still exist violations
  - Larger chip area
  - Place module precisely

10. NanoRoute – GUI

NanoRoute

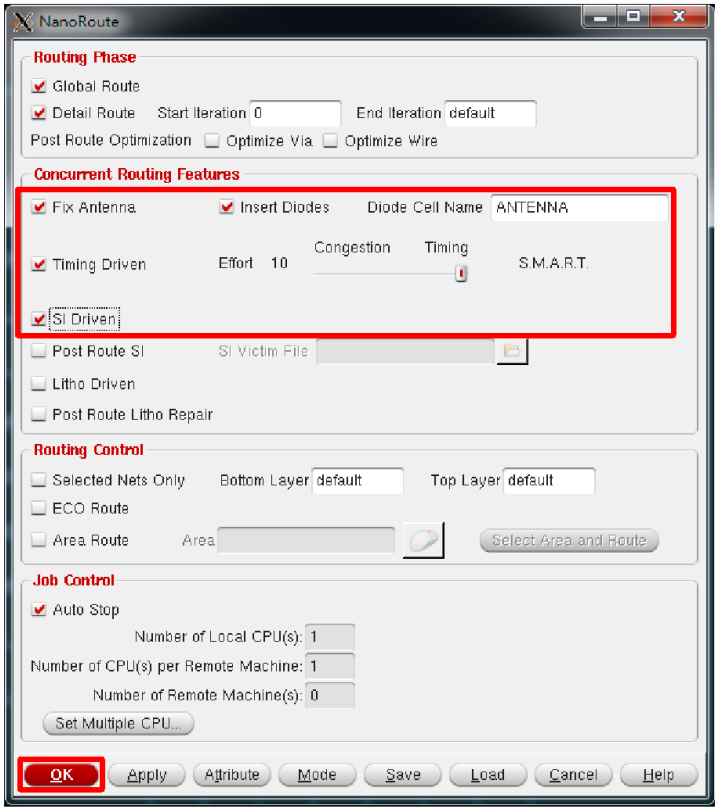
- Routing the design without creating DRC or LVS violations
- Routing the design without degrading timing or creating signal integrity violations
- Complete the form and click OK

Field	Fill In	
Concurrent Routing Features	◆ Fix Antenna	
	◆ Insert Diodes	
	Diode Cell Name	ANTENNA
	◆ Timing Driven	
	◆ SI Driven	

Add I/O Pad Filler before Routing!!

a

b

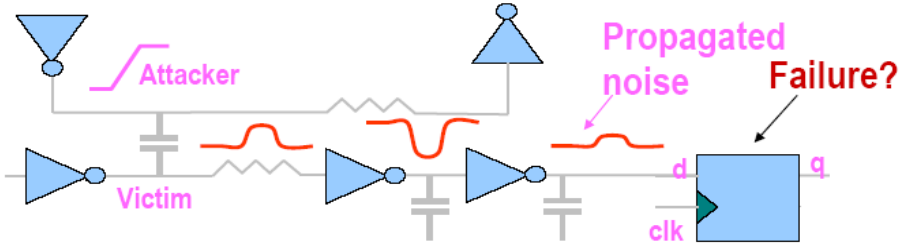


11. Post-Route Timing Analysis

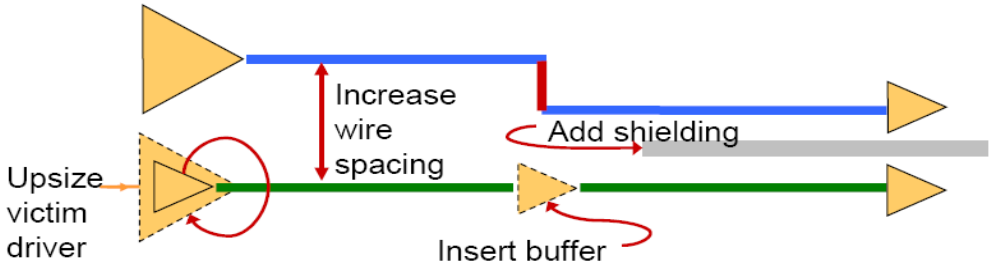
- **To check setup time**
  - innovus #> timeDesign –postRoute
- **To check hold time**
  - innovus #> timeDesign –postRoute –hold
- **Operation**
  - Repair Setup slack, Setup times
  - Repair Design rule violations (DRVs) and remaining DRVs
- **Reports**
  - Generated timing reports are saved in ./timingReports/ , including
    - postRoute.cap
    - _postRoute.fanout
    - _postRoute.tran
    - _postRoute_all.tarpt

12. Post-Route Timing Analysis (include SI)

- **Text Command**
  - To do glitch noise analysis
    - innovus #> timeDesign –postRoute –si



- **SI repair techniques**

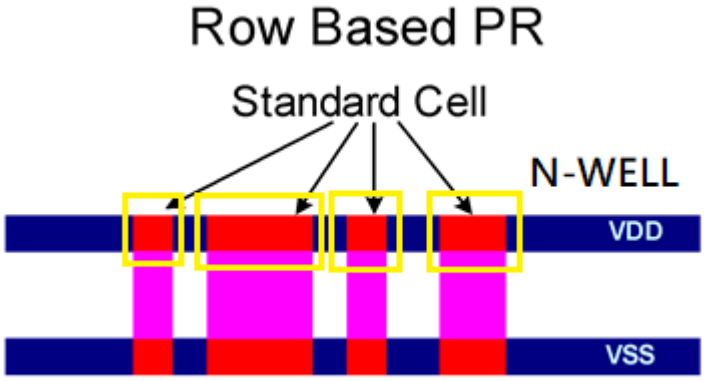


- **Text command**
  - To correct setup violations and design rule violations
    - innovus #> optDesign –postRoute –si
  - To correct hold violations
    - innovus #> optDesign –postRoute –si –hold

13. Add Filler Cells

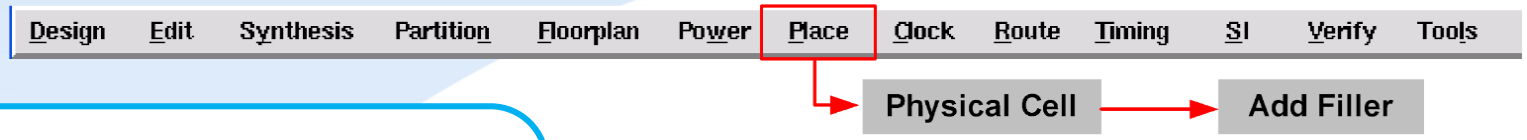
- **Purpose of adding filler cells**
  - Fill all the gaps between standard cell instances
  - Provide decoupling capacitances to complete connections in the standard cell rows

- **Metal filler inserted after routing, but before parasitic extraction and GDSII out**



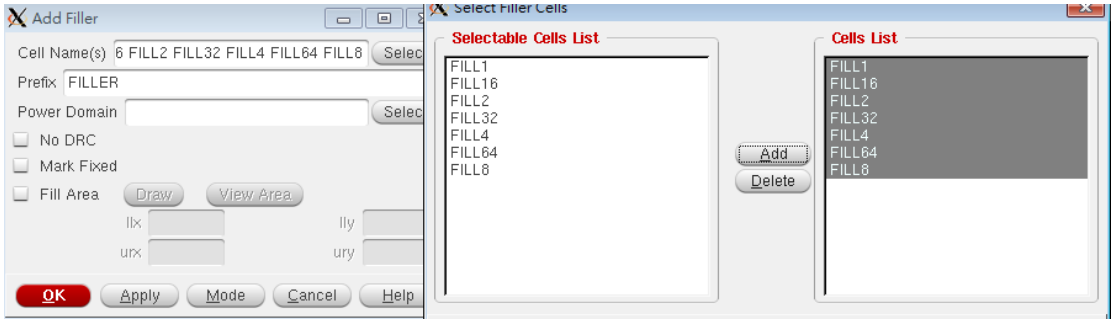
13. Add Filler Cells

- **Add filler cell into the free space of core design**



Why add dummy

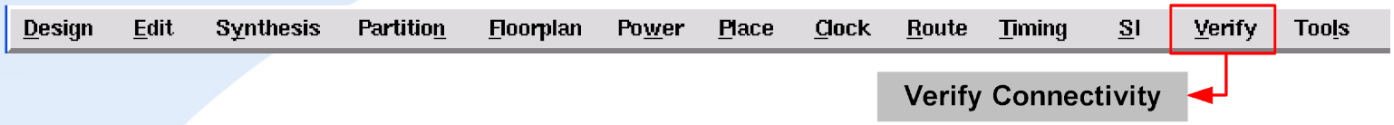
- *meet minimize metal density rule*
- *prevent over etching*
- *prevent sagging in local area*
- *improve yield*
- *reduce on chip variation*



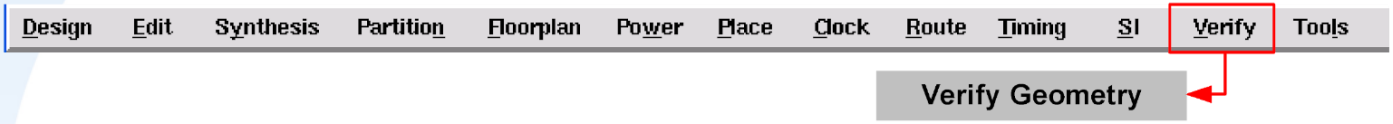
- **Text command**
  - innovus #> addFiller –cell FILL64 –prefix filler64
  - innovus #> addFiller –cell FILL32 –prefix filler32
  - innovus #> addFiller –cell FILL16 –prefix filler16
  - innovus #> addFiller –cell FILL8 –prefix filler8
  - innovus #> addFiller –cell FILL4 –prefix filler4
  - innovus #> addFiller –cell FILL2 –prefix filler2
  - innovus #> addFiller –cell FILL1 –prefix filler1

14. Before Output: LVS & DRC check

- **LVS(layout verse schemectic)**



- **DRC(design rule check)**



- **Reminder**
  - Recommended to perform the two checks after power planning
  - Check your design before streaming out

14. LVS v.s. DRC v.s. DRV

- **LVS: layout verse schematic**
  - The signals are connect correctly
- **DRC: design rule check**
  - Metal density
  - Size of poly, metals, vias
- **DRV: design rule violation**
  - Fan-out
  - Load
  - Driving capability
  - Wire length

14. Output Files

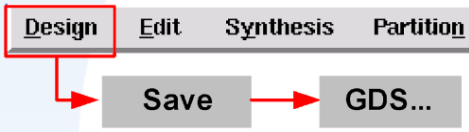
- **The following files will be generated after APR design flow**
  - CHIP.v
    - Netlist file for post-layout gate-level simulation
  - CHIP.sdf
    - Design timing file for post-layout gate-level simulation
  - CHIP.def
    - Design exchange format relevant to physical layout
  - CHIP.gds
    - GDSII stream file for Calibre-DRC (Design Rule Check)

14. Output Data

- **Export Netlist for LVS and simulation**
  - File -> Save -> Netlist …
  - innovus #> saveNetlist CHIP_LAYOUT.v
  - innovus #> saveNetlist –includePhysicalInst –excludeLeafCell CHIP_LVS.v
- **Save sdf for post layout simulation**
  - Innovus #> setAnalysisMode -checkType setup -analysisType bcwc -cppr none -clockGatingCheck 1 -timeBorrowing 1 -domain clock -useOutputPinCap 1
  - innovus #> write_sdf -edges check_edge CHIP.sdf

14. Output Data

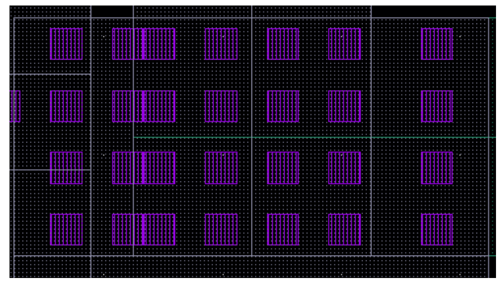
- **Export GDSII stream file for DRC, LVS, LPE (Layout Parameter Extration), and tape out**
  - CHIP.gds
- **Merging GDSII files (UMC 0.18)**
  - Merge the GDSII files of standard cells (and macros if any) into complete layout and output to a single GDSII file for hierarchical designs



Field	Fill In
Output Stream File	CHIP.gds
Map File	streamOut.map
Library Name	CHIP
◆ GDS Structure Name	CHIP
◆ Merge Stream Files	umc18.gds2 umc18io3v5v_6lm.gds2

Appendix: More on APR

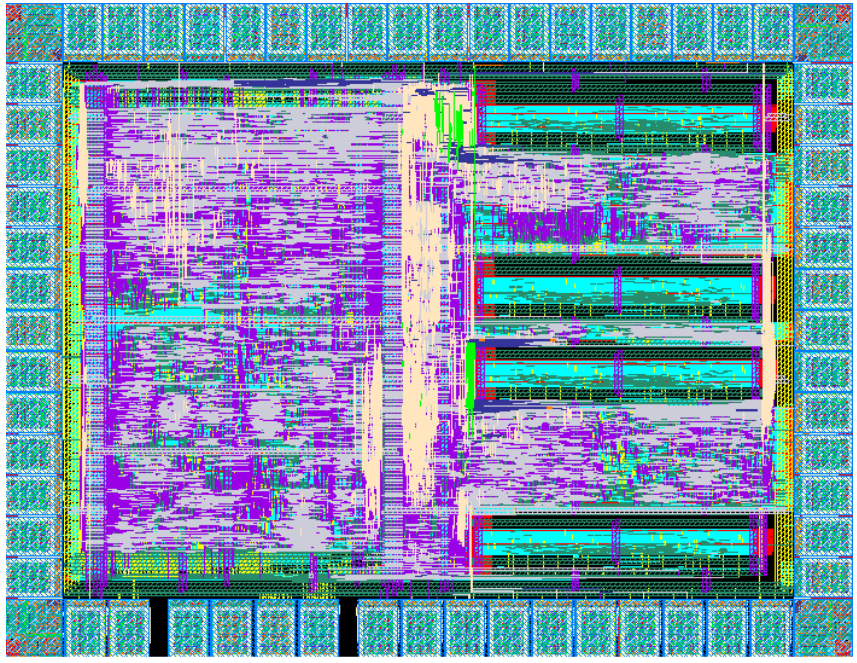
- **DRC & LVS in Virtuoso with streamout GDS**
- **DRC Result**
  - PO.R.3
  - M2.R.1
  - VIA3.S.1



```
                      OVERALL COMPARISON RESULTS


            #         ###################       _   _
           #          #                 #       *   *
      #   #           #     CORRECT     #         |
       # #            #                 #       \___/
        #             ###################

Warning:  Ambiguity points were found and resolved arbitrarily.

**************************************************************************************************************
                                     CELL  SUMMARY
**************************************************************************************************************

Result         Layout                          Source
-----------    -----------                     --------------
CORRECT        CHIP                            CHIP
```

Appendix: More on APR

- **Macro Layout**
  - Pad -> Pin
  - No Shell
- **Different Power Domain**



Appendix

- Modify the Memory.lef file
  - Change “core” to “core_5040”
  - Change “ME4” to “metal4” and so on
  - Change “VI1” to “via”
  - Change “VI2” to “via2” and so on

```
RECT 0.000 0.140 213.900 294.000 ;
LAYER VI1 ;
RECT 0.000 0.140 213.900 294.000 ;
LAYER VI2 ;
RECT 0.000 0.140 213.900 294.000 ;
LAYER VI3 ;
RECT 0.000 0.140 213.900 294.000 ;
```

```
NAMESCASESENSITIVE ON ;
MACRO TEST
CLASS BLOCK ;
FOREIGN TEST 0.000 0.000 ;
ORIGIN 0.000 0.000 ;
SIZE 213.900 BY 294.000 ;
SYMMETRY x y r90 ;
SITE core ;
PIN VCC
  DIRECTION INOUT ;
  USE POWER ;
  SHAPE ABUTMENT ;
 PORT
  LAYER ME4 ;
  RECT 212.780 282.580 213.900 285.820 ;
  LAYER ME3 ;
  RECT 212.780 282.580 213.900 285.820 ;
  LAYER ME2 ;
  RECT 212.780 282.580 213.900 285.820 ;
  LAYER ME1 ;
  RECT 212.780 282.580 213.900 285.820 ;
 END
```