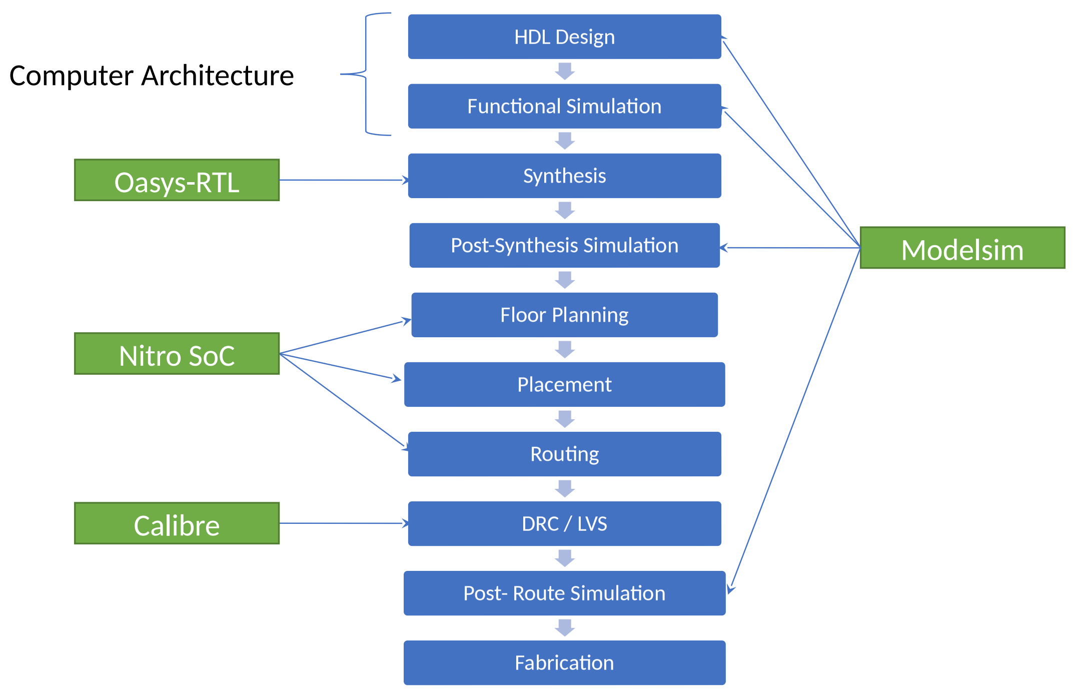
CMP3020 VLSI

Lab3-4-5:
FloorPlanning
Place & Route

Agenda

1. <u>Review</u> on Last Lab	Oasys-RTL + Modelsim
2. FloorPlanning	Nitro-SoC
3. Placement	Nitro-SoC
4. Routing	Nitro-SoC
5. Post-Routing Simulation	Modelsim (Questasim)
• Export Verilog & SDF	Nitro-SoC

1.Review: ASIC Flow



1.Review: Oasys-RTL Flow Required Files

- RTL (Verilog or VHDL) design by **YOU**
- Librety Files (.lib) →

    describe electrical constraints and **timing** for each cell (Capacitance, resistances, ..etc)
- Physical libraries (.lef) →

    describe the cell as shapes (width, height and antennas effect as well, vias places and metals, …etc)
- Timing constraints (.sdc)
- Floorplans (.def) → floor planning defination
- Power intent (.upf) → power information
- Process technology (.ptf) → interconnect information
- Design for Testability DFT (.ctl)
- Switching activity Interchange Format (SAIF)

1.Review: Oasys-RTL Flow Used **Scripts**

- 0_init_design.tcl → **Make sure to set All the parameters in this file correctly**
- 1_read_design.tcl
- 2_synthesize_optimize.tcl
- 3_export_design.tcl
- 4_clear_designs.tcl
- run.tcl → it calls script from 0 till 3

2.Post-synthesize Simulation

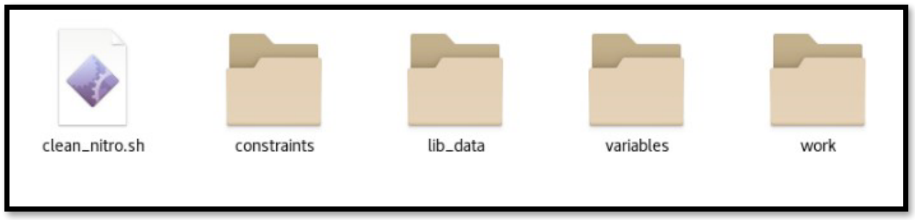
This step is to guarantee that the output netlist does the same functionality as the input, with no timing consideration yet.

Steps:

- Synthesize
- Get output synthesized Verilog File from Oasys
- Get Library Verilog File (NangateOpenCellLibrary.v)
- Compile them together
- Simulate

3. **Nitro**: Setup…Folder Structure

- Folder Structure



  - Constraints & lib_data → are same as Lab2
  - Work → empty folder to start our Nitro from
  - Clean_nitro.sh → our cleanup script
  - variables → Contains the main script files that we will change with our data.

3. **Nitro**: Setup… variables Folder

- variables Folder contains the following files:
  - import_variables.tcl → You need to modify with your source, library and constraints paths.
  - floorplan_variables.tcl → parameters need to be set for floorplanning.
    → set power planning to false.
  - flow_variables.tcl → variables required for the rest of the flow.

3. Nitro: Run …Floorplan & Placement

- Add your synthesized verilog and constraints files in place
- Update scripts in variables folder
- Clean and setup work environments

```
source clean_nitro.sh
```

```
cd work
```

3. **Nitro**: Run …Floorplan & Placement

- Go to work Directory, Open Nitro

```
nitro -log LOGs/nitro.log  -journal LOGs/nitro.jou
```

- Run setup

```
setup_nrf
```

- Run Import Script

```
source flow_scripts/0_import.tcl
```

3. Nitro: Run …GUI…Check & Save

- Open GUI

```
start
```

- Check_design

```
check_design   (press F2 to open results in separate window)
```

- Save design

```
write_db -file ../db/import.db
```

3. **Nitro**: Run …GUI…Power Planning

- Create Power/Ground Rails

  Floorplan Menu → Power → Create P/G rails

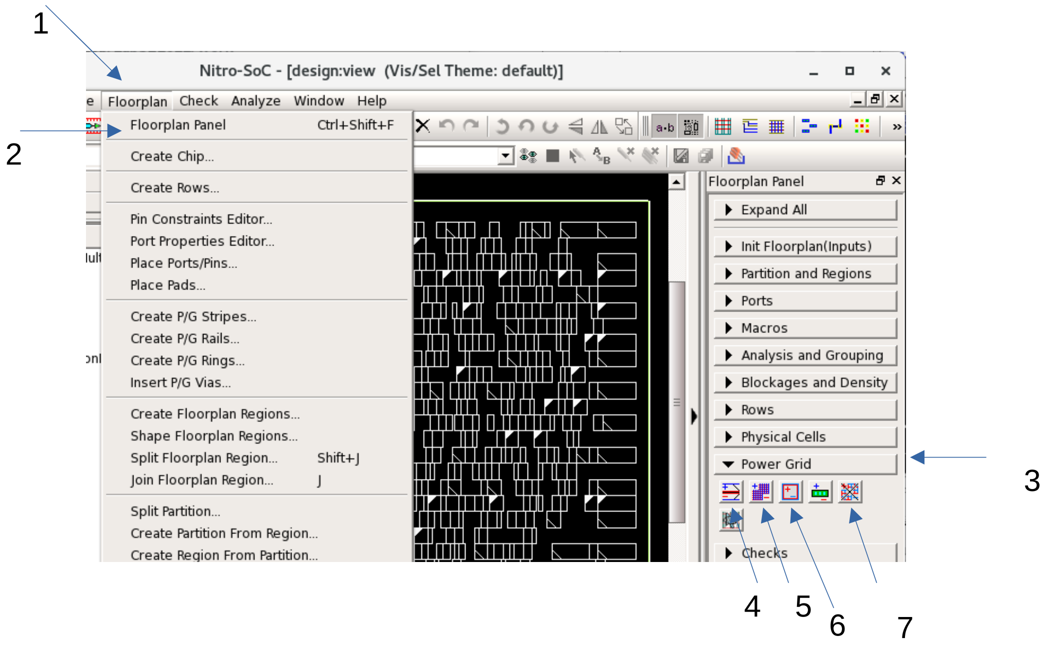
- Create Power/Ground strips

  Floorplan Menu → Create P/G strips

- Create Power/Ground rings

  Floorplan Menu → Create P/G rings

1. open menu
2. choose floorplan Panel
3. open power Grid
4. Create power rails
5. Create power strips
6. Create power rings
7. remove power elements


1
2
3
4
5
6
7
Nitro-SoC - [design:view (Vis/Sel Theme: default)]
Floorplan Check Analyze Window Help
Floorplan Panel Ctrl+Shift+F
Create Chip...
Create Rows...
Pin Constraints Editor...
Port Properties Editor...
Place Ports/Pins...
Place Pads...
Create P/G Stripes...
Create P/G Rails...
Create P/G Rings...
Insert P/G Vias...
Create Floorplan Regions...
Shape Floorplan Regions...
Split Floorplan Region... Shift+J
Join Floorplan Region... J
Split Partition...
Create Partition From Region...
Create Region From Partition...
Floorplan Panel
Expand All
Init Floorplan(Inputs)
Partition and Regions
Ports
Macros
Analysis and Grouping
Blockages and Density
Rows
Physical Cells
Power Grid
Checks

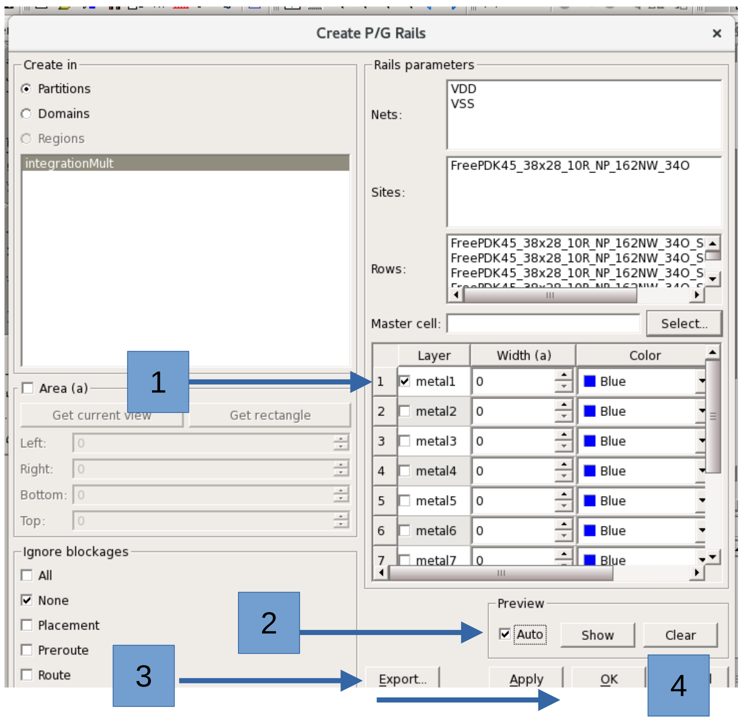
Create Power Rails

1. choose metal layer you want (usually metal1 or metal2)

2. choose auto to preview the placement on the chip

3. export to save tcl command to script (to reuse it later)

4. Apply or OK to apply (not both)



Create P/G Rails

Create in
- [✓ Partitions]
- Domains
- Regions

integrationMult

Area (a)

Get current view | Get rectangle

Left: 0
Right: 0
Bottom: 0
Top: 0

Ignore blockages
- All
- [✓ None]
- Placement
- Preroute
- Route

Rails parameters

Nets: VDD, VSS

Sites: FreePDK45_38x28_10R_NP_162NW_34O

Rows:
FreePDK45_38x28_10R_NP_162NW_34O_S
FreePDK45_38x28_10R_NP_162NW_34O_S
FreePDK45_38x28_10R_NP_162NW_34O_S

Master cell:

	Layer	Width (a)	Color
1	✓ metal1	0	🟦 Blue
2	metal2	0	🟦 Blue
3	metal3	0	🟦 Blue
4	metal4	0	🟦 Blue
5	metal5	0	🟦 Blue
6	metal6	0	🟦 Blue
7	metal7	0	🟦 Blue

Preview
- [✓ Auto]

Create Power Strips

1. choose metal layer you want (usually a “vertical high metal” like 6)

2. choose auto to preview the placement on the chip

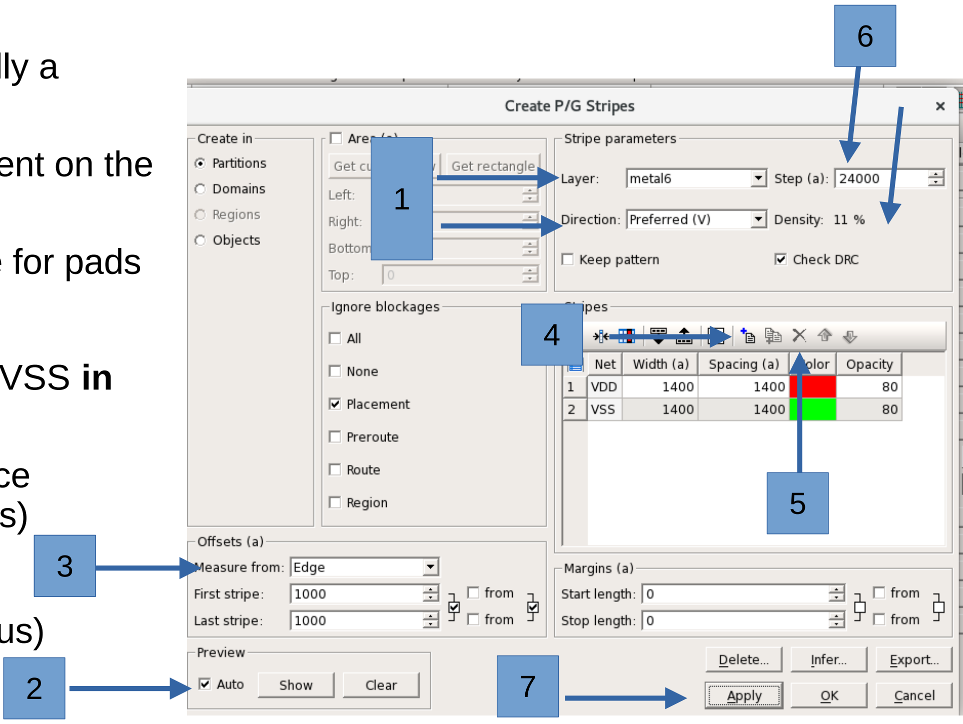
3. set offset from sides to leave space for pads

4. add both VDD and VSS strips

5. use this to remove the added VDD/VSS **in case of mistakes**

6. change step to introduce more space between strips (watch density changes)

7. export/ Apply / OK (same as previous)



Create in: [✓ Partitions] Domains Regions Objects

Area (a): Left: Right: Bottom: Top: 0

Stripe parameters
Layer: metal6 Step (a): 24000
Direction: Preferred (V) Density: 11 %
☐ Keep pattern ☑ Check DRC

Ignore blockages: ☐ All ☐ None ☑ Placement ☐ Preroute ☐ Route ☐ Region

Stripes

	Net	Width (a)	Spacing (a)	Color	Opacity
1	VDD	1400	1400	🟥	80
2	VSS	1400	1400	🟩	80

Offsets (a)
Measure from: Edge
First stripe: 1000 ☐ from
Last stripe: 1000 ☐ from

Margins (a)
Start length: 0 ☐ from
Stop length: 0 ☐ from

Preview: ☑ Auto

Create Power Rings

1. choose metal layer you want (usually HIGHER than strip)

2. choose auto to preview the placement on the chip

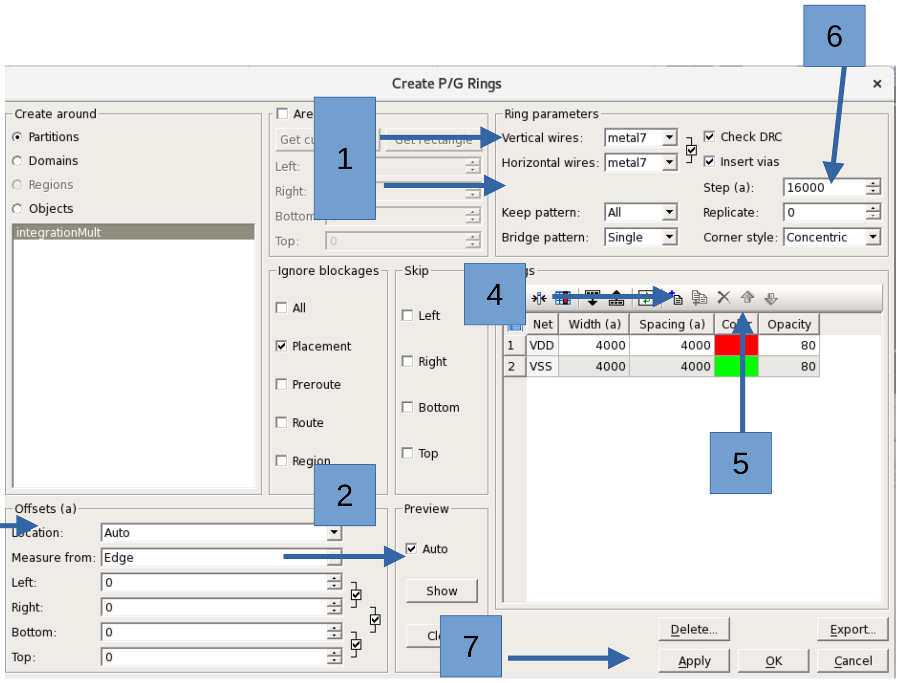
3. set offset from sides to leave space for pads

4. add both VDD and VSS strips

5. use this to remove the added VDD/VSS **in case of mistakes**

6. change step to introduce more space between strips (watch density changes)

7. export/ Apply / OK (same as previous)



Create around: [✓ Partitions] Domains Regions Objects
integrationMult

Ring parameters
Vertical wires: metal7
Horizontal wires: metal7
[✓] Check DRC
[✓] Insert vias
Step (a): 16000
Keep pattern: All
Replicate: 0
Bridge pattern: Single
Corner style: Concentric

Ignore blockages: All [✓ Placement] Preroute Route Region

Skip: Left Right Bottom Top

	Net	Width (a)	Spacing (a)	Co	Opacity
1	VDD	4000	4000	🟥	80
2	VSS	4000	4000	🟩	80

Offsets (a)
Location: Auto
Measure from: Edge
Left: 0
Right: 0
Bottom: 0
Top: 0

Preview: [✓] Auto

3. **Nitro**: Run …GUI…Check & Save

- insert P/G Vias

```
insert_pg_vias -partitions [get_top_partition]
```

- Trim P/G excess routes

```
trim_pg_route
```

- Save design

```
write_db -file db/pg.db
```

3. **Nitro**: Run …Route & Errors

- Open GUI

```
source  flow_scripts/3_route.tcl  > LOGs/route.log
```

- Run design check

```
Check → check_design  → run
```

- View errors

```
Press F2 to view errors
```

3. **Nitro**: Save..

- Save Design

```
write_db -file ../db/final.db
```

- Saving timing annotation file

```
write_sdf -file multiplier.sdf -corner corner_0_0 -mode new_mode -skip_backslash true
```

- Saving Verilog file

```
write_verilog -file multiplier.route.v
```

3. **Nitro** … Tips

- All steps made through UI are written in the transcript, you might consider saving it
- Repeating the commands doesn’t always guarantee generating same results, Saving design is essential.
- Port Editor has a bug and sometimes doesn’t show all Ports
- Always use check_design to make sure you didn’t mess up something
- Always read error and warnings, they are reported for a reason
- Some errors and warnings can be ignored safely
- Always direct the ouput of scripts to a file to help you in later tracing

4. **Modelsim**: Post-synthesis simulation

- Create project, and add file to it
- Add adder.route.v and NangateOpenCellLibrary.v to project
- Add sdf file to project
- Compile Files
- run simulation using sdf file

```
vsim  <toplevelModuleName> -t ps -sdfmax <path2sdfFile>
```

```
vsim  multiplier.route.v -t ps -sdfmax /somepath/multiplier.sdf
```

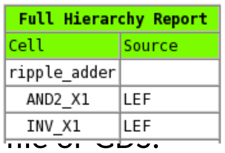
Thank You

4. **Next** Time and Additions

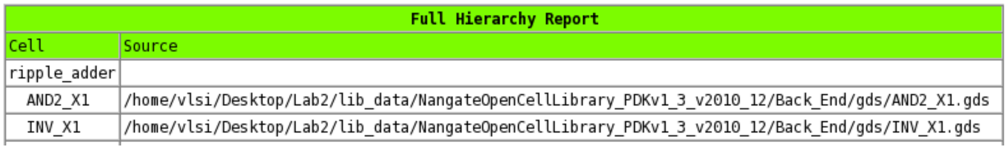
- Nitro-SoC
  - Prioritize Clock and use Clock Tree Synthesize
  - Get Statistics out of Nitro
  - Export GDSII File to perform physical verification.
- Calibre
- Extras If We have time
  - Nitro-Soc
    - Manual Floor Planning
    - Adding Ready Made components and using Def files
  - Oasys-RTL:
    - Design Space Exploration
    - Scan Chains
    - Several libraries optimization

Export GDSII

1. Configure the tool to read library GDS
   (preferably only the gds you need not the whole library)
   a. report_stream_lib
      This command will display the used cells and whether the source is just LEF
   b. config_stream_lib -common { path2file/AND2_X1.gds path2file/INV_X1.gds }
      This command will read in the gds files needed
   c. report_stream_lib



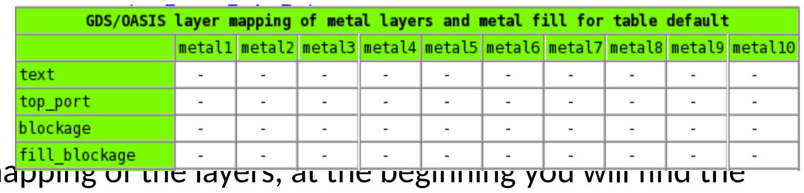
Full Hierarchy Report	
Cell	Source
ripple_adder	
AND2_X1	LEF
INV_X1	LEF



Full Hierarchy Report	
Cell	Source
ripple_adder	
AND2_X1	/home/vlsi/Desktop/Lab2/lib_data/NangateOpenCellLibrary_PDKv1_3_v2010_12/Back_End/gds/AND2_X1.gds
INV_X1	/home/vlsi/Desktop/Lab2/lib_data/NangateOpenCellLibrary_PDKv1_3_v2010_12/Back_End/gds/INV_X1.gds

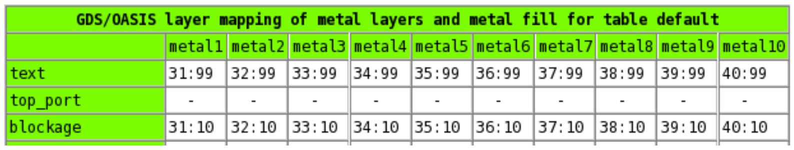
Export GDSII

2. Add Layer Mapping
   a. report_stream_layer_map
      This command will display the mapping of the layers, at the beginning you will find the whole table empty

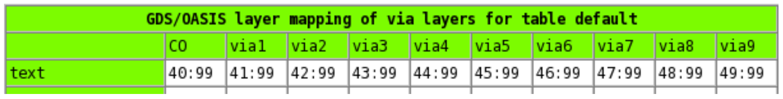


GDS/OASIS layer mapping of metal layers and metal fill for table default										
	metal1	metal2	metal3	metal4	metal5	metal6	metal7	metal8	metal9	metal10
text	-	-	-	-	-	-	-	-	-	-
top_port	-	-	-	-	-	-	-	-	-	-
blockage	-	-	-	-	-	-	-	-	-	-
fill_blockage	-	-	-	-	-	-	-	-	-	-

   b. source pathto/LayerMapping.tcl
      This command will map metals and vias to Layer numbers that GDSII can understand.
   c. report_stream_layer_map
      Now the tables have some numbers



GDS/OASIS layer mapping of metal layers and metal fill for table default										
	metal1	metal2	metal3	metal4	metal5	metal6	metal7	metal8	metal9	metal10
text	31:99	32:99	33:99	34:99	35:99	36:99	37:99	38:99	39:99	40:99
top_port	-	-	-	-	-	-	-	-	-	-
blockage	31:10	32:10	33:10	34:10	35:10	36:10	37:10	38:10	39:10	40:10



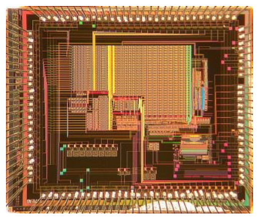
GDS/OASIS layer mapping of via layers for table default										
	CO	via1	via2	via3	via4	via5	via6	via7	via8	via9
text	40:99	41:99	42:99	43:99	44:99	45:99	46:99	47:99	48:99	49:99

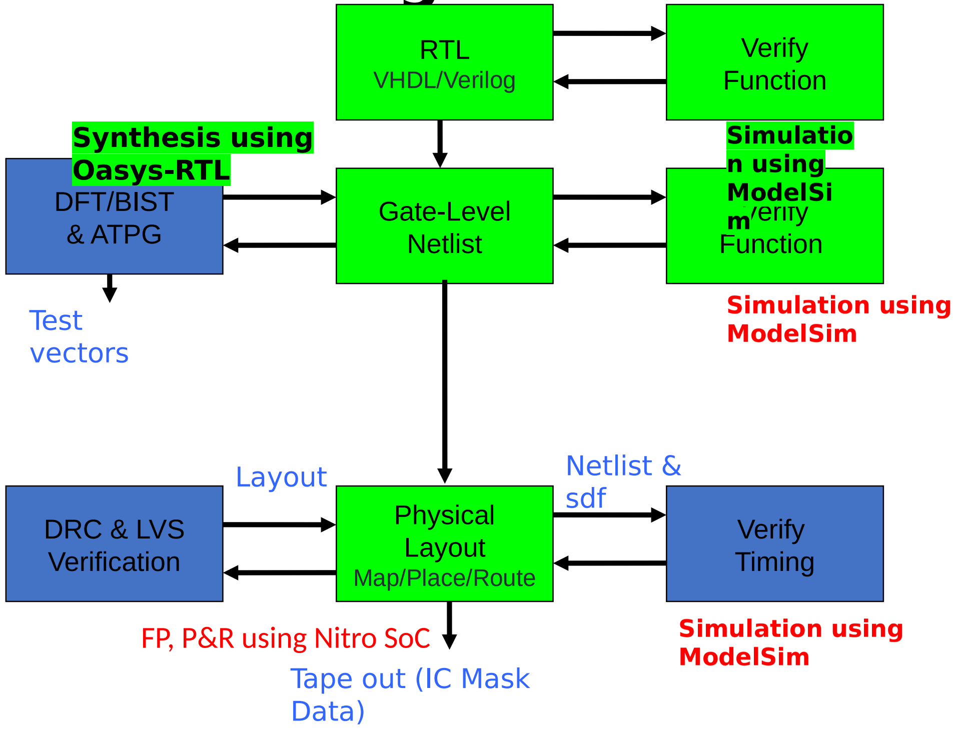
Export GDSII

2. Write GDSII file
    a. write_stream -file pathto/output.gds

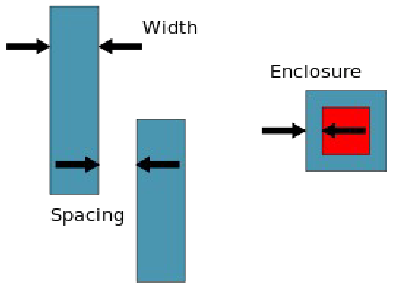
    now your design is exported as a GDS in the folder you specified into "pathto" with the name "output.gds" please do change the names for your convenience

Cell-Based ASIC Design Flow





The three basic DRC checks


Width
Spacing
Enclosure