Digital IC-Project and Verification

Place and Route

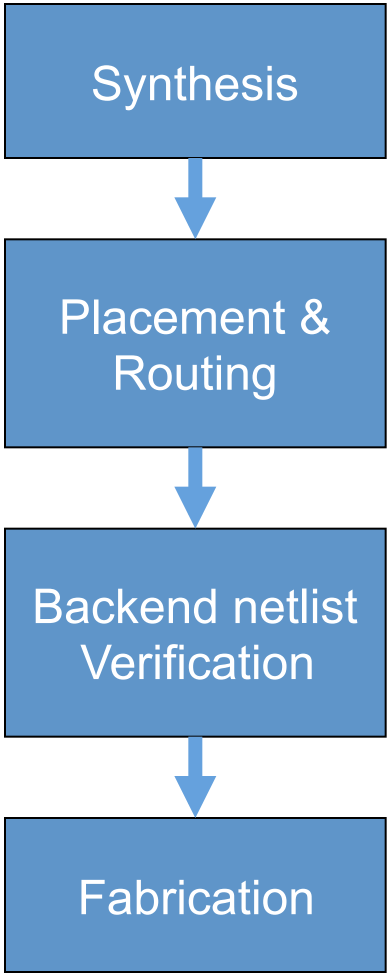
Oskar Andersson

Outline

- Backend ASIC Design flow (Physical Design)
  - General steps

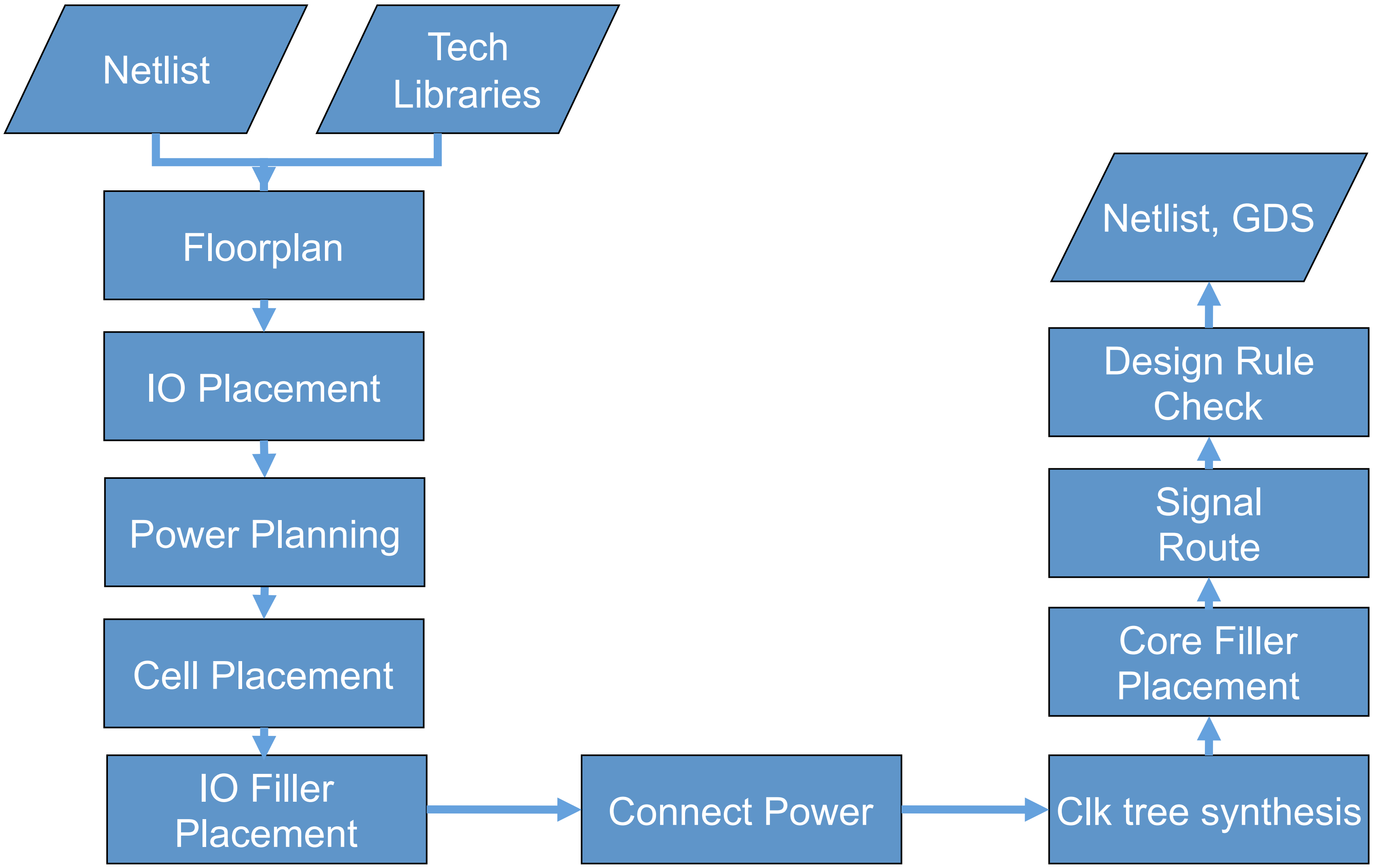
- Input files
- Floorplanning
- Placement
- ClockTree-synthesis
- Routing

Typical Backend Design Flow

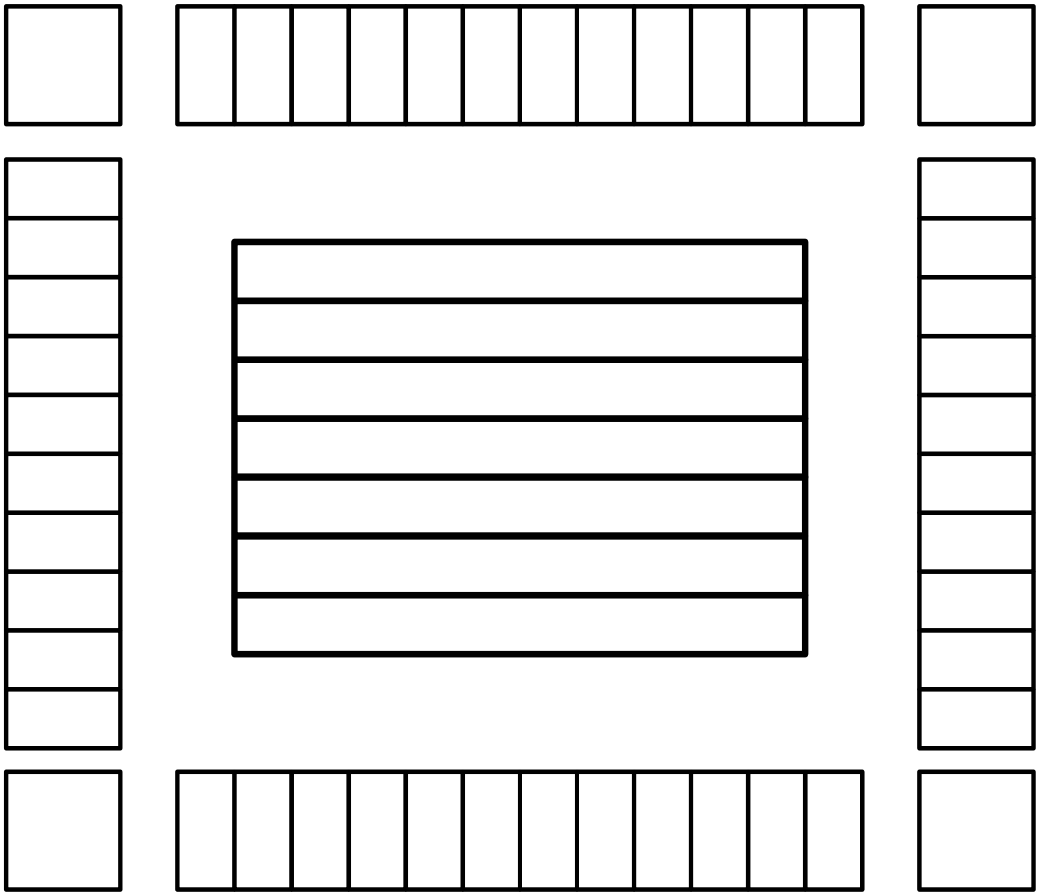


- Synthesis:
  - Synopsys Design Compiler (Design Vision)
- Placement:
  - Encounter Digital Implementation
- Backend netlist Verification
  - Modelsim
- Fabrication
  - ASIC vendor, e.g. UMC, ST, TSMC

SoC Encounter Flow

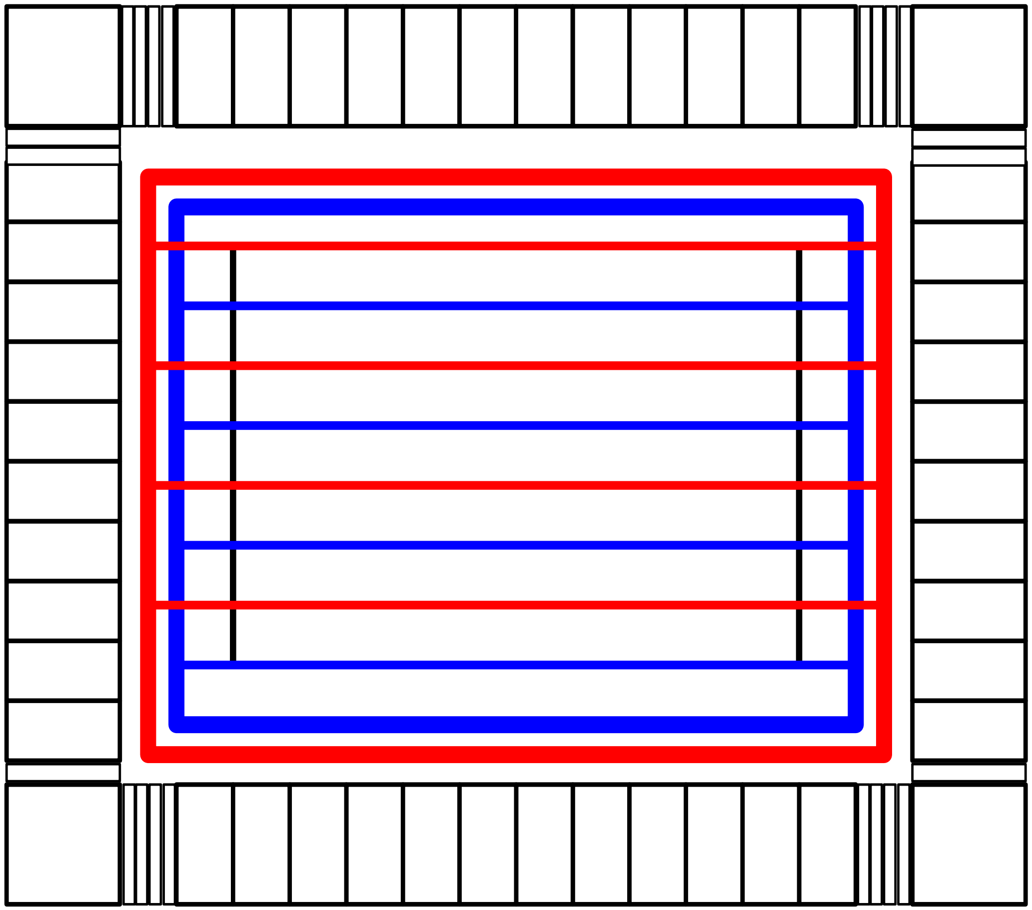


SoC Encounter Flow



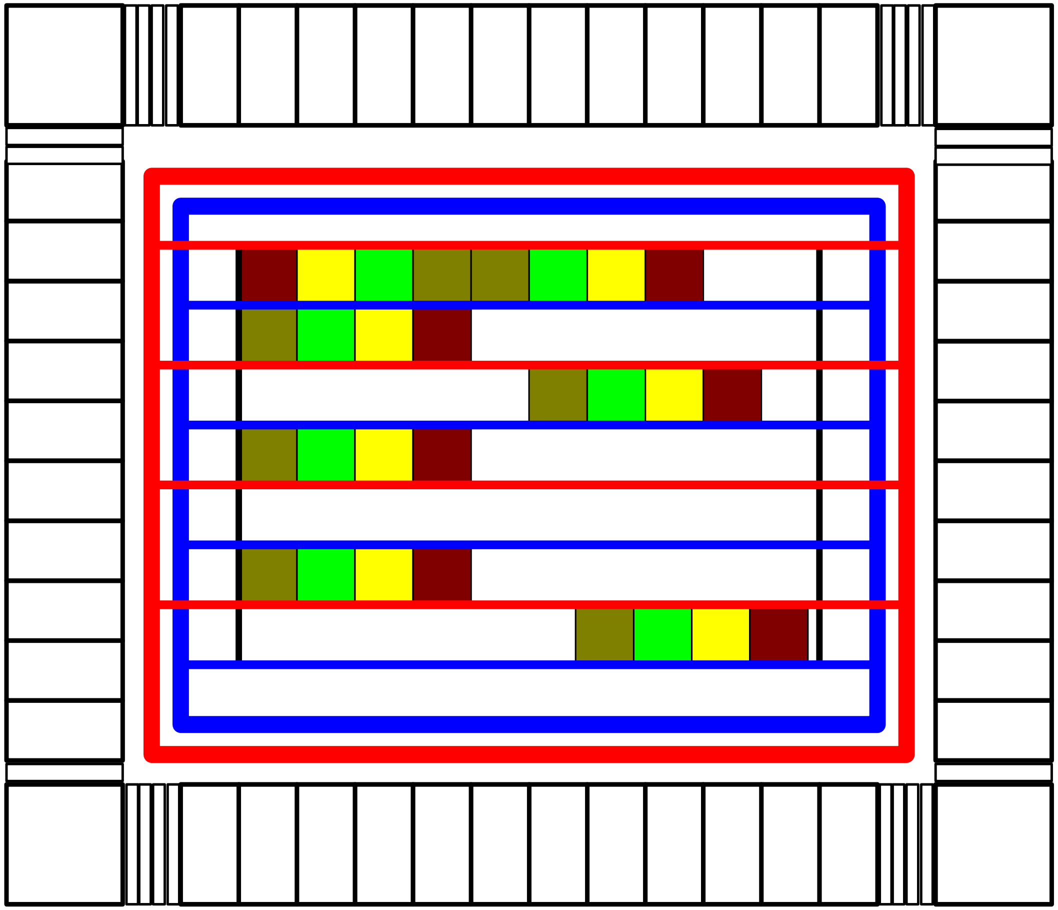
- Floorplan:
  - Placement area
  - IOs
  - RAM/ROM

SoC Encounter Flow



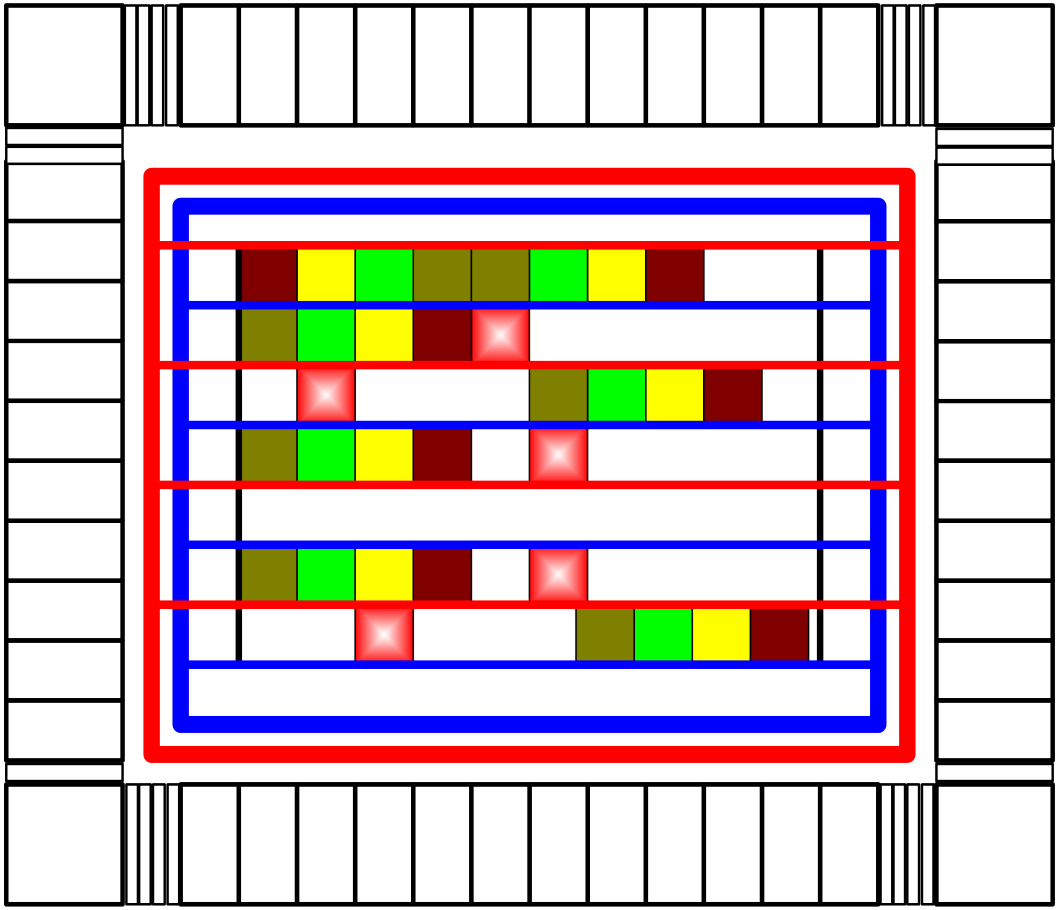
- Power Planning
  - Design a power ring
  - Add horizontal and vertical power stripes

SoC Encounter Flow



- Place Cells:
  - Place all the standard cells into the rows

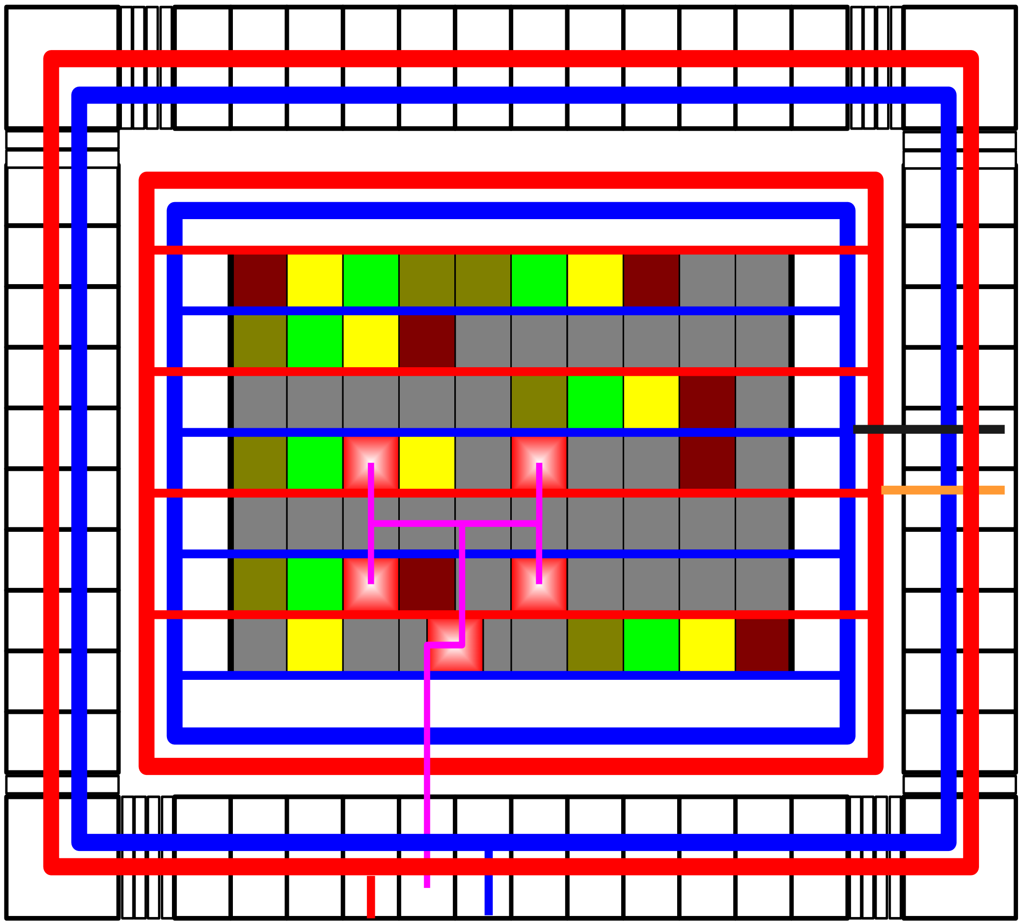
SoC Encounter Flow



Clock Tree Synthesis:

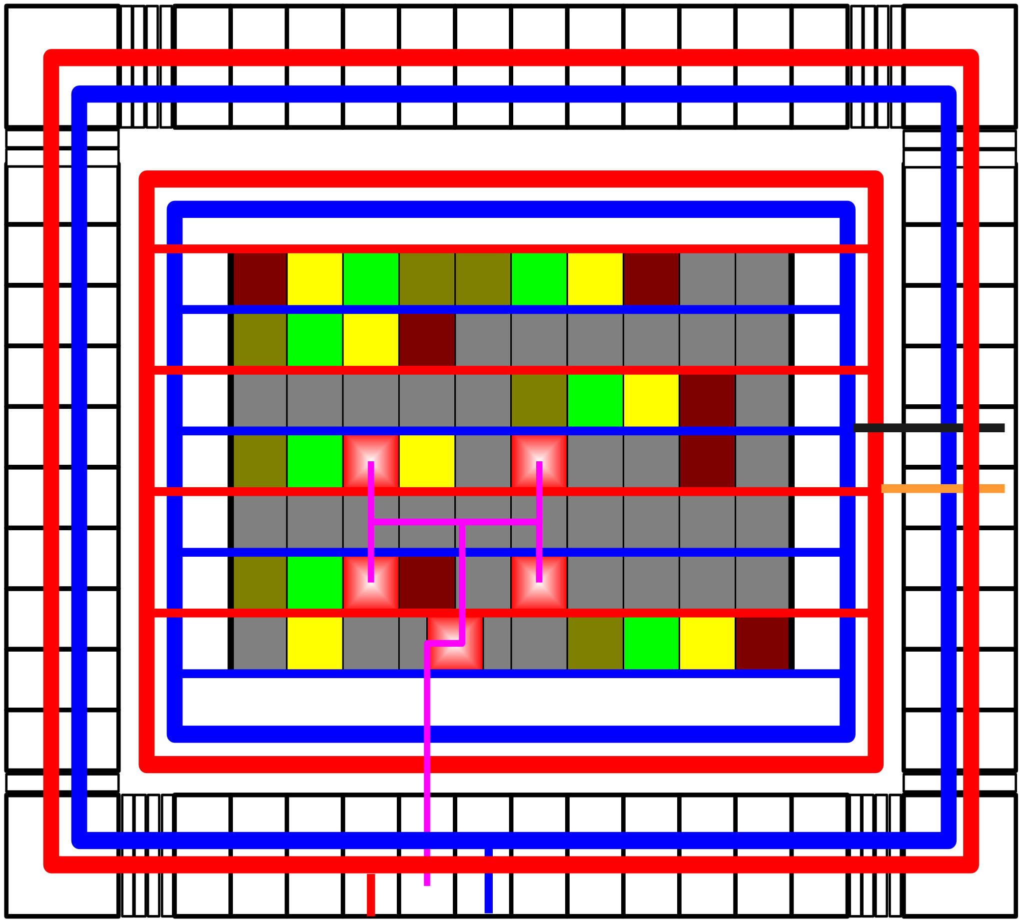
- Places clock buffers
- Timing constraints
  - Skew etc

SoC Encounter Flow



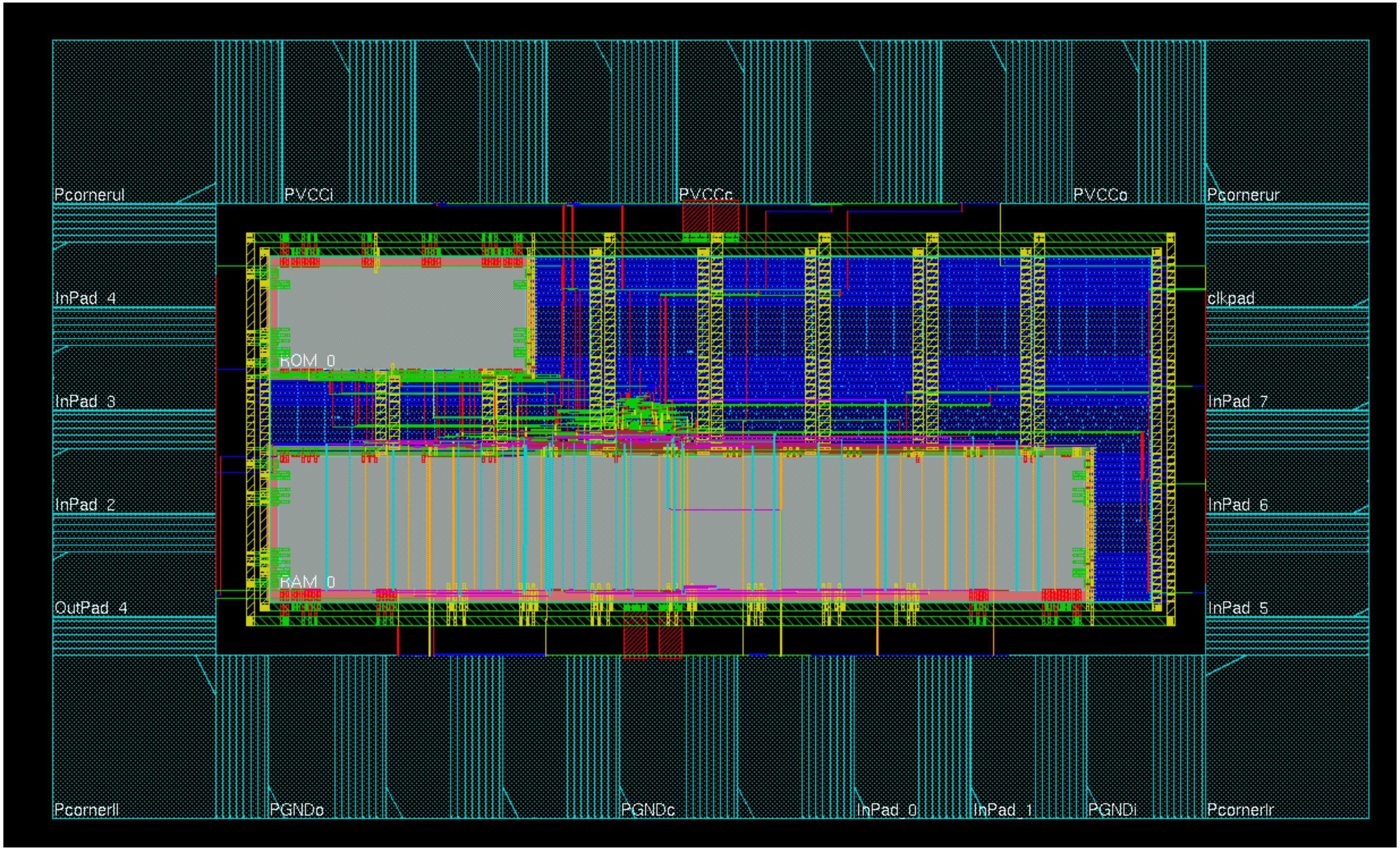
- Connect Power Supply:
  - Core Power
  - Pad Power
- Add FILLER cells
  - core filler cells
  - IO filler cells

SoC Encounter Flow



- Route Clock tree:
  - Finds an “optimal” way
  - Reduces skew
- Route signal nets
  - Final step

Demo Layout


Pcornerul
PVCCi
PVCCc
PVCCo
Pcornerur
InPad_4
clkpad
ROM_0
InPad_3
InPad_7
InPad_2
InPad_6
RAM_0
OutPad_4
InPad_5
Pcornerll
PGNDo
PGNDc
InPad_0
InPad_1
PGNDi
Pcornerlr

Technology Description Files

LEF: Library Exchange Format

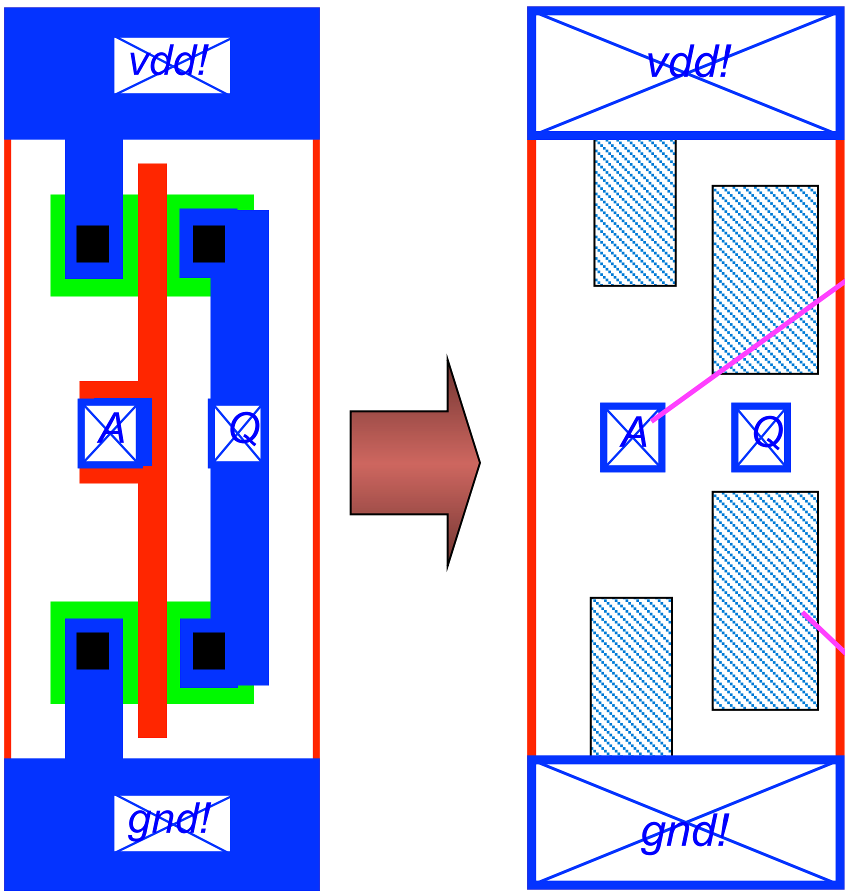
- Technology: Design rules, Capacitance, Resistance, Antenna factor, Vias
  - header.lef

- Cells & pads: Size, Class, Placement, Pin Information, Obstructions.
  - Standard_cell.lef
  - IO.lef

LEF-Example: Inverter

Layout

Abstract



Physical cell size

Terminals with physical placement

Obstructions

LEF

```
MACRO IV
    CLASS CORE ;
    FOREIGN IV 0.000 0.000 ;
    ORIGIN 0.00 0.00 ;
    SIZE 3.00 BY 12.00 ;
    SYMMETRY x y  ;
    SITE CORE ;
    PIN A
        DIRECTION INPUT ;
        ANTENNASIZE 1.4 ;
        PORT
        LAYER metal1 ;
        RECT  0.50 5.00 1.00
    5.50 ;
        END
    END A
            ⋮
    OBS
        LAYER metal1 ;
        RECT  1.90 6.50 2.60
    7.20 ;
        RECT  0.40 4.90 1.00
    5.60 ;
```

Design Description Files

Enc: Encounter Format

- Netlist, Layout

DEF: Design Exchange Format (not used in our flow.

- Netlist, Layout

Verilog

- Netlist, generated from synthesis tool

Required Data for PnR (Faraday 130nm)

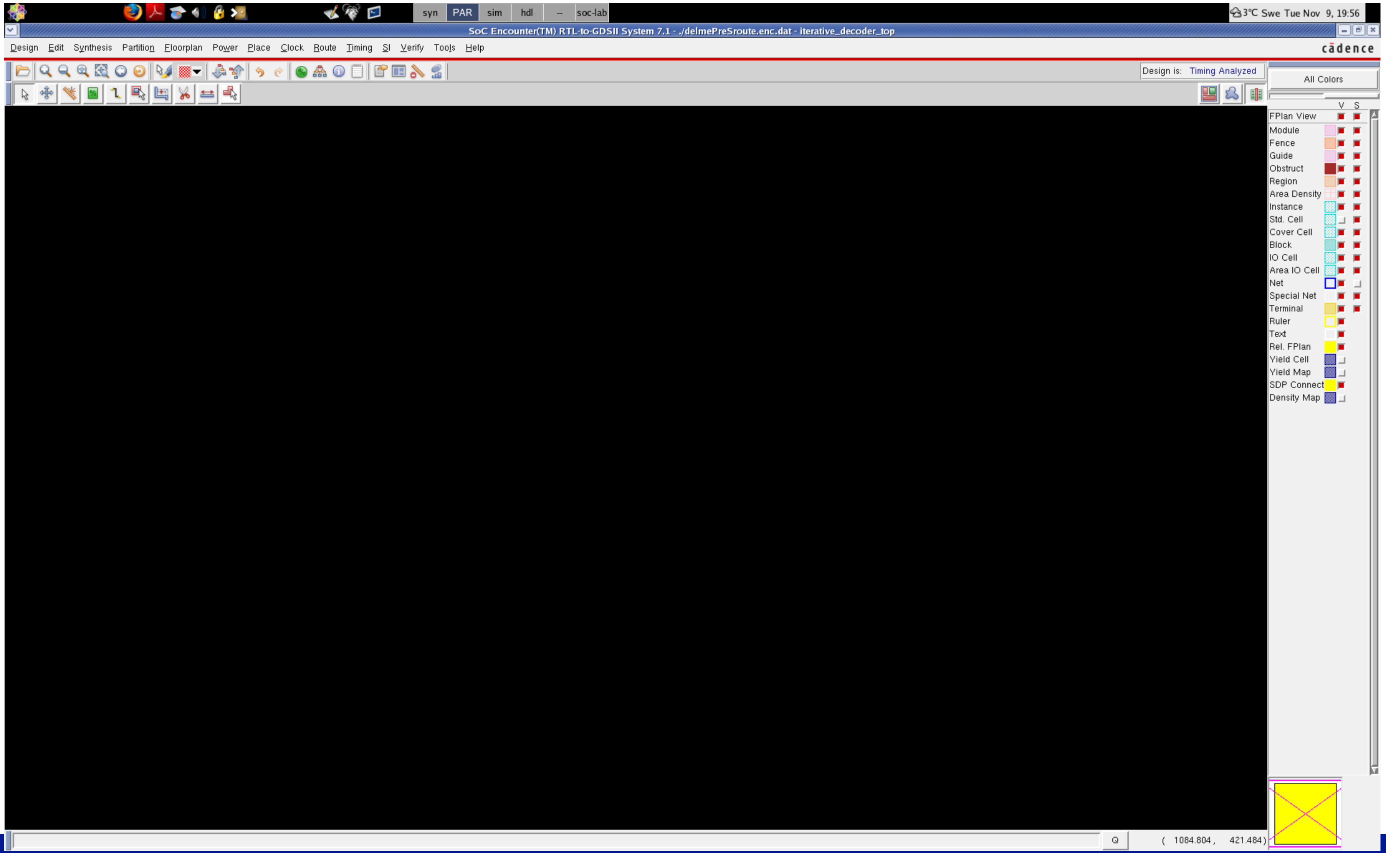
- LEF: Library Exchange Format
  - header.lef
  - standardCell.lef : Cell Library
  - IO.lef : Pad Library
  - memory.lef : custom
- lib/tlf: libraries that contain timing information
- sdc: Synopsys Design Constraint (generated during synthesis). Optional
- Memory: memory.lib
- Design (netlist): your_design.v

Starting the SoC Encounter

inittde dicp13

velocity

Remember to maintain the directory hierarchy.



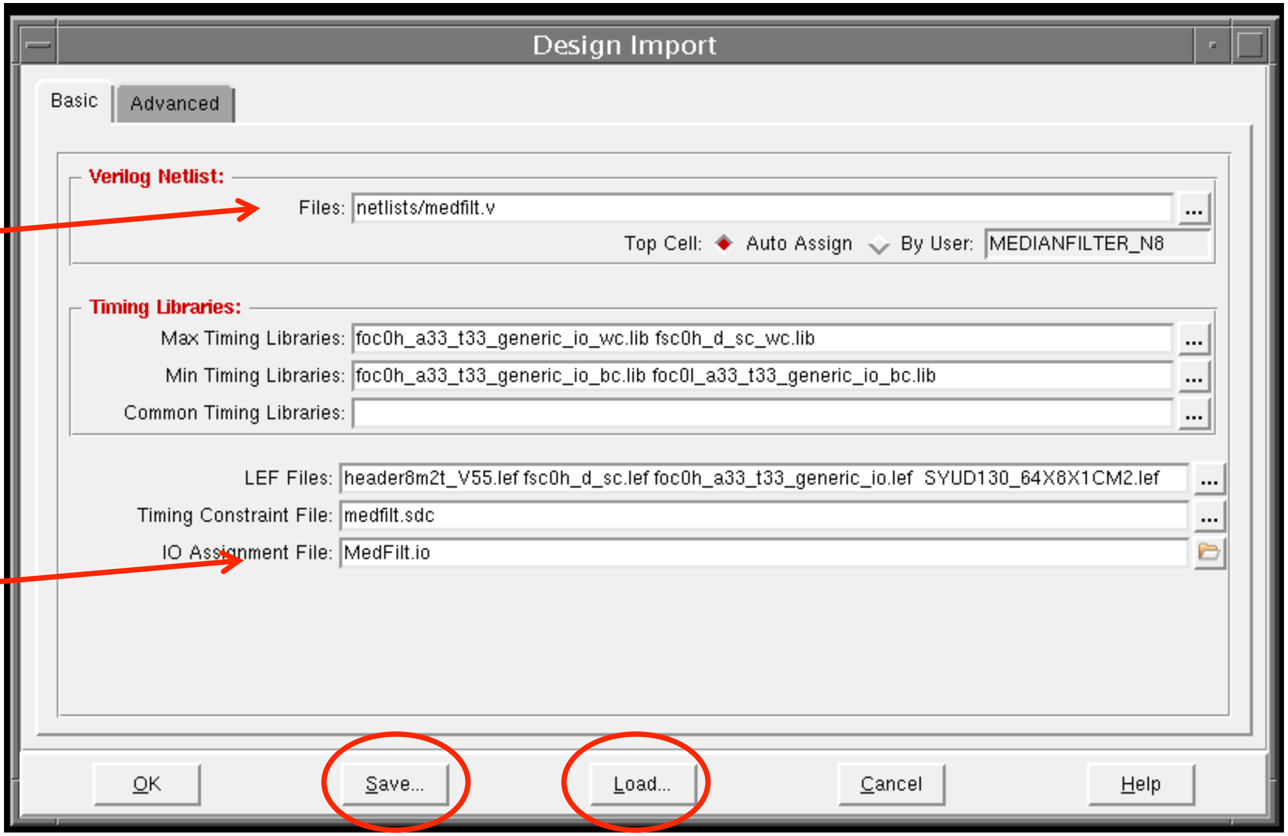
Design Import

Design -> Import Design

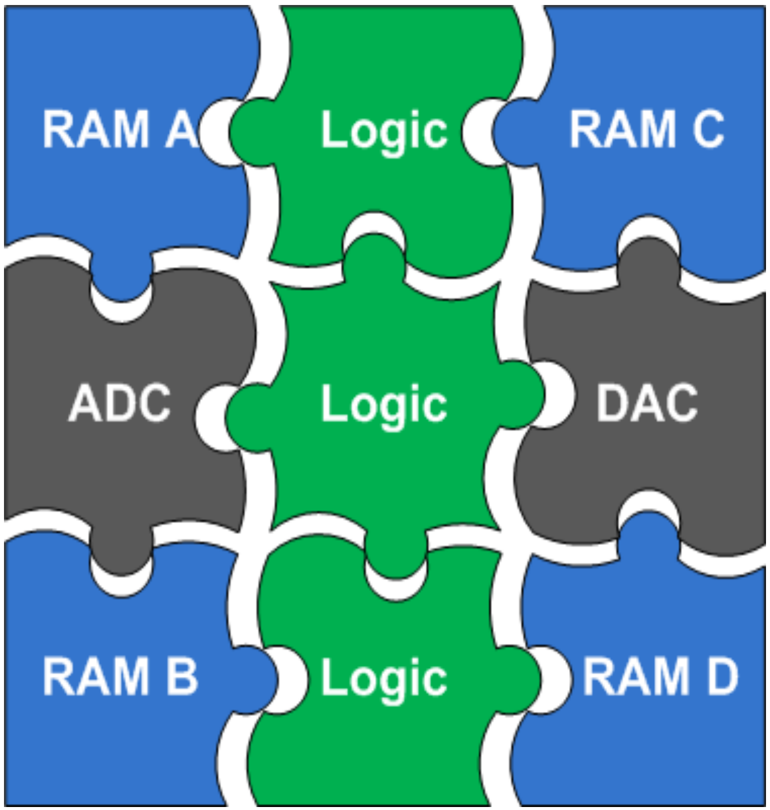
/usr/local-eit/cad2/far130/syn2010/

Needs to be specified

Will be provided

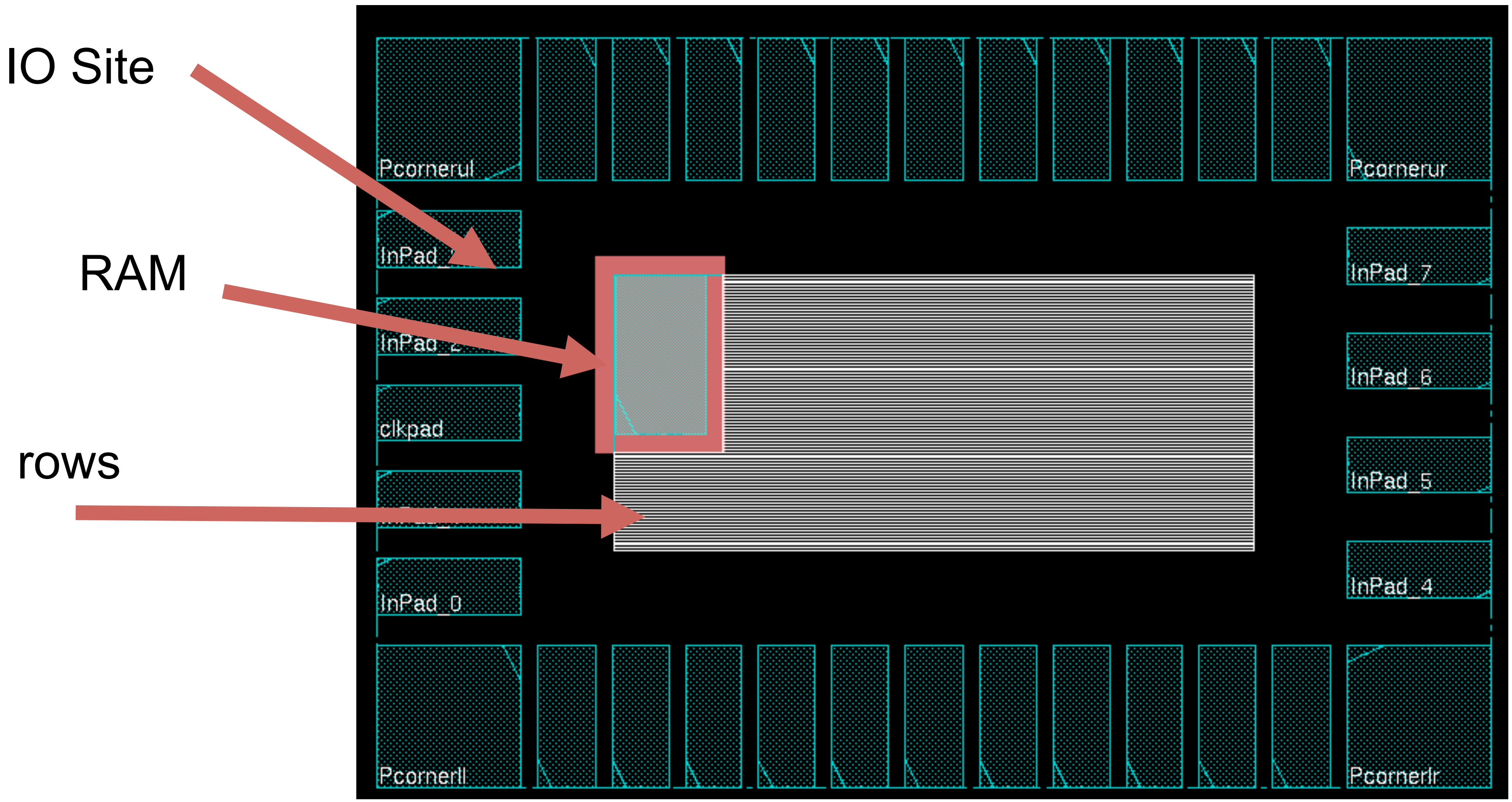


Floorplan

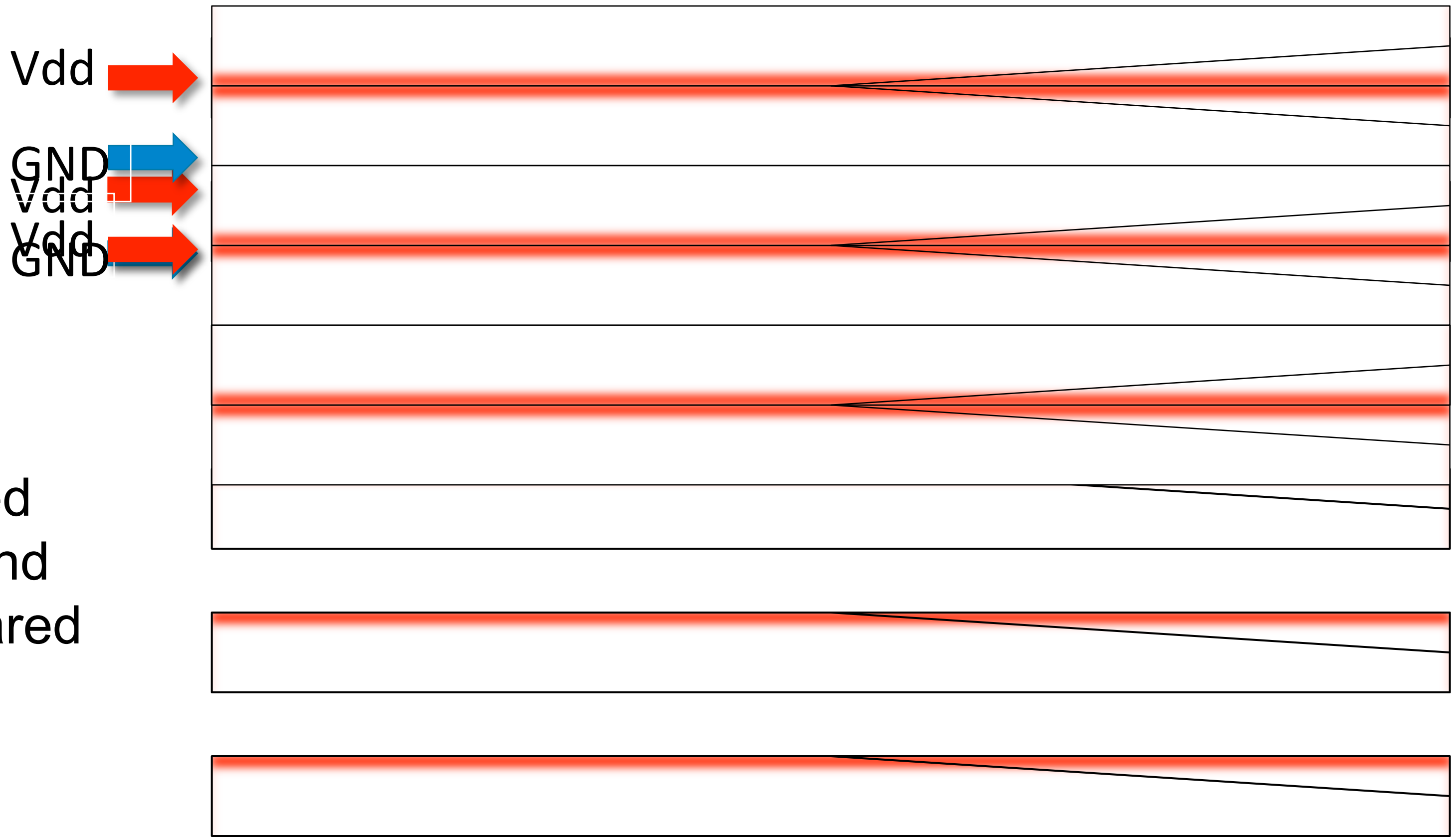


- A starting floorplan is created (required area is estimated by the tool)
- Global and detailed routing grids are created
- The core rows are created
- Sites for IOs are created
  - IO and block to core distance is defined by the user

Floorplanning



Core Rows



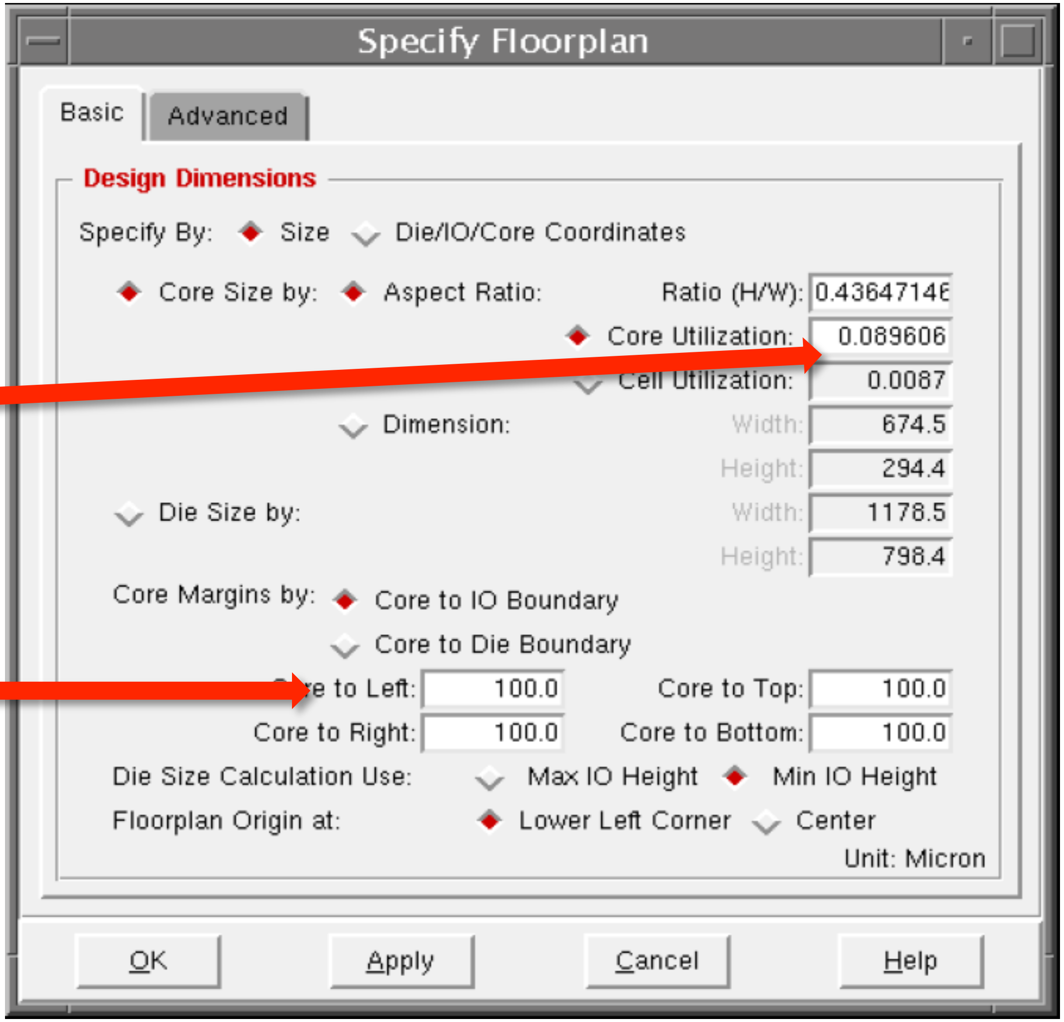
If rows are flipped and abut VDD and GND can be shared by 2 rows.
Default setting!

Floorplan Setup

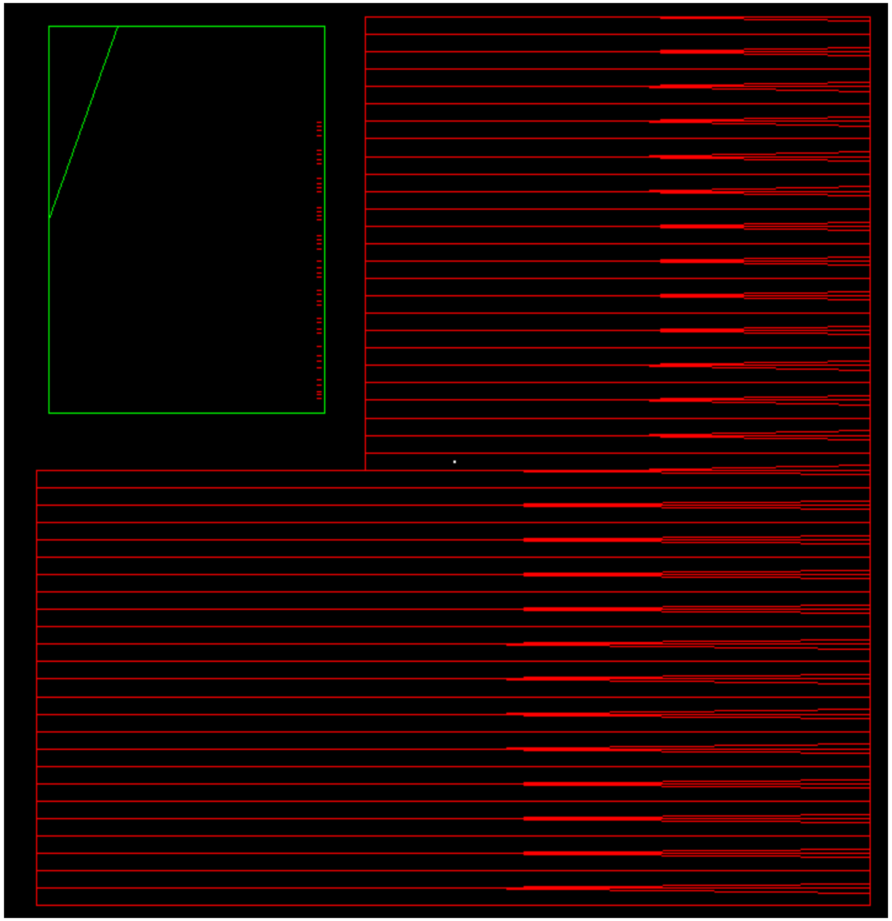
Floorplan -> Specify Floorplan

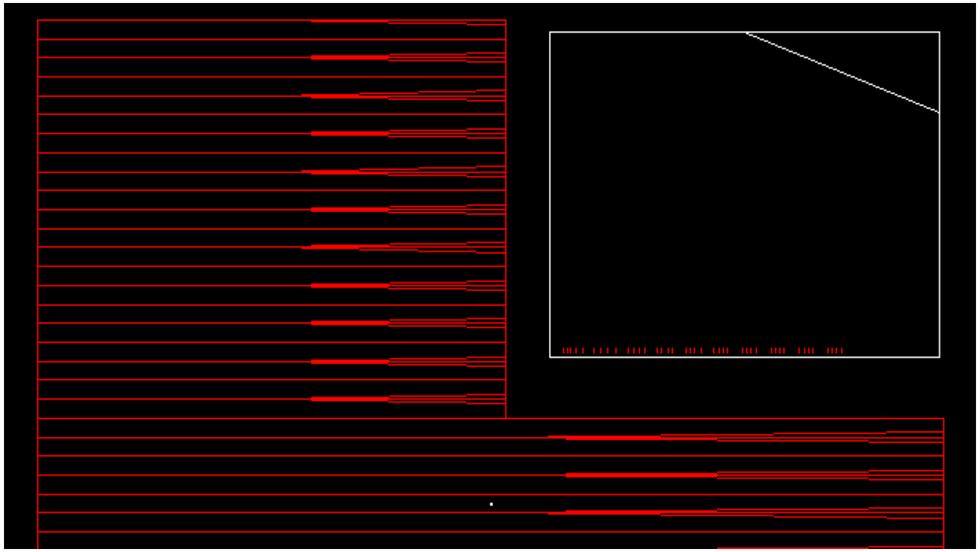
Core utilization

IO to core distance



Block Placement





Flight lines will indicate location of the pins

Block: Circuitry that is pre-routed, e.g., RAM.

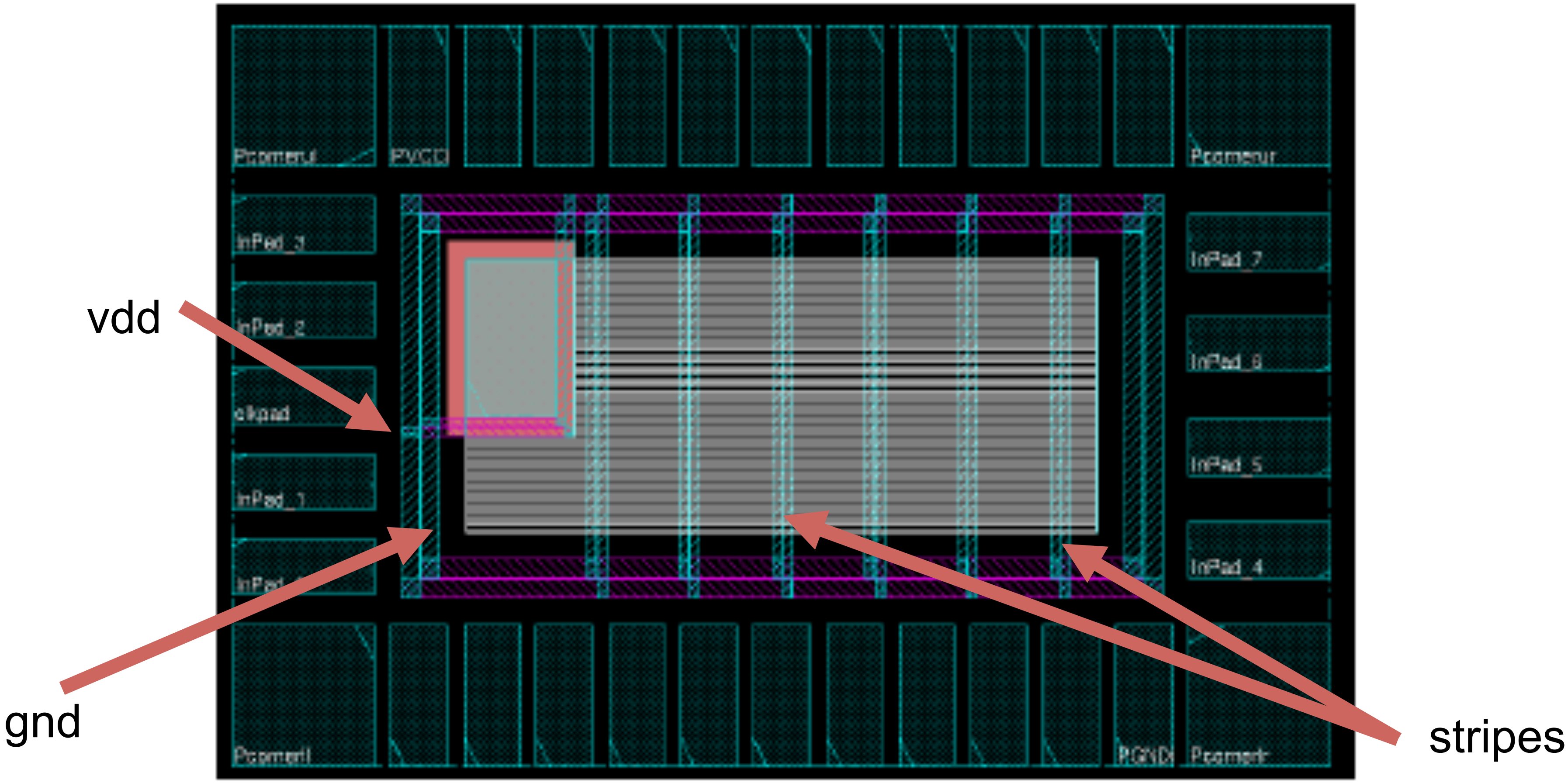
IO Placement

- Specify location/orientation of pads
  - Input, output
  - core-power, pad-power
- Recommendation:
  - Put core power supply on *top or botton*
  - Use gaps in the pad frame for additional power supply.
  - !No CORE power supply at the corners!
  - The more supplies the better

Power Rings

- Power paths are planned and modified before routing
- Creation of power rings that surround all blocks and core.
- Creation of stripes over rows
- Connects rings, stripes and pads

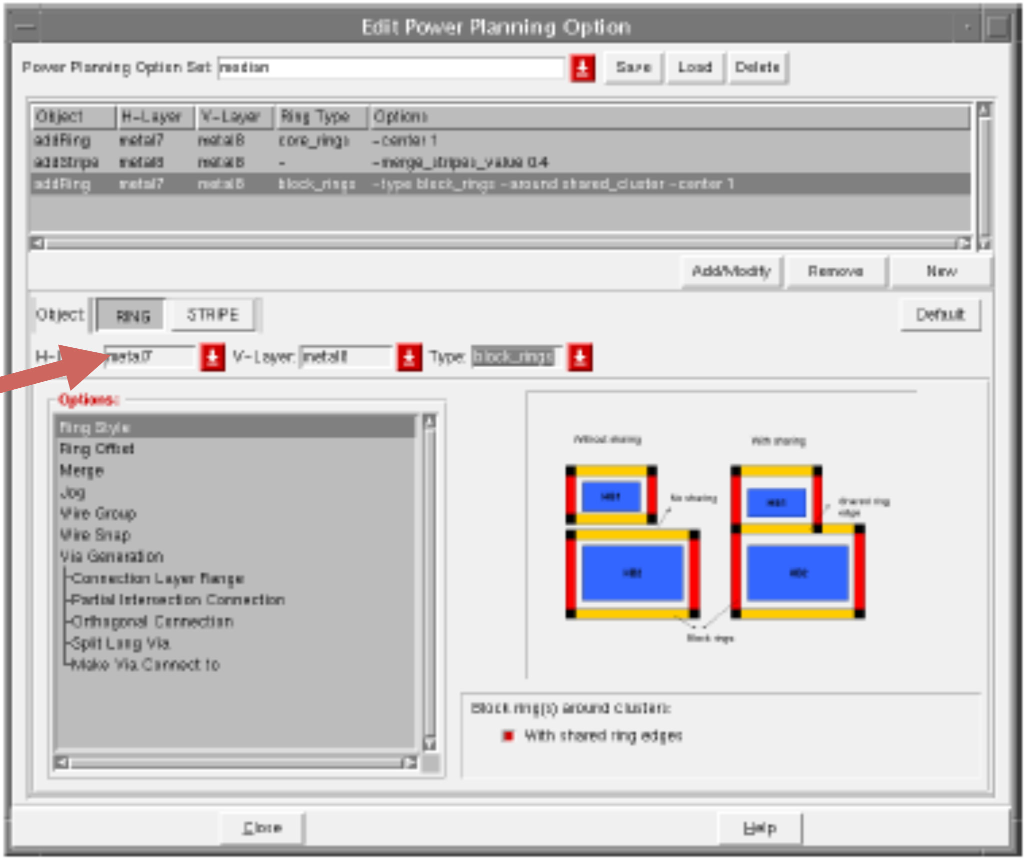
Power Rings cont'd



Power Ring Setup

Power-> Power planning -> Add Rings

Power-> Power planning -> Add Stripes

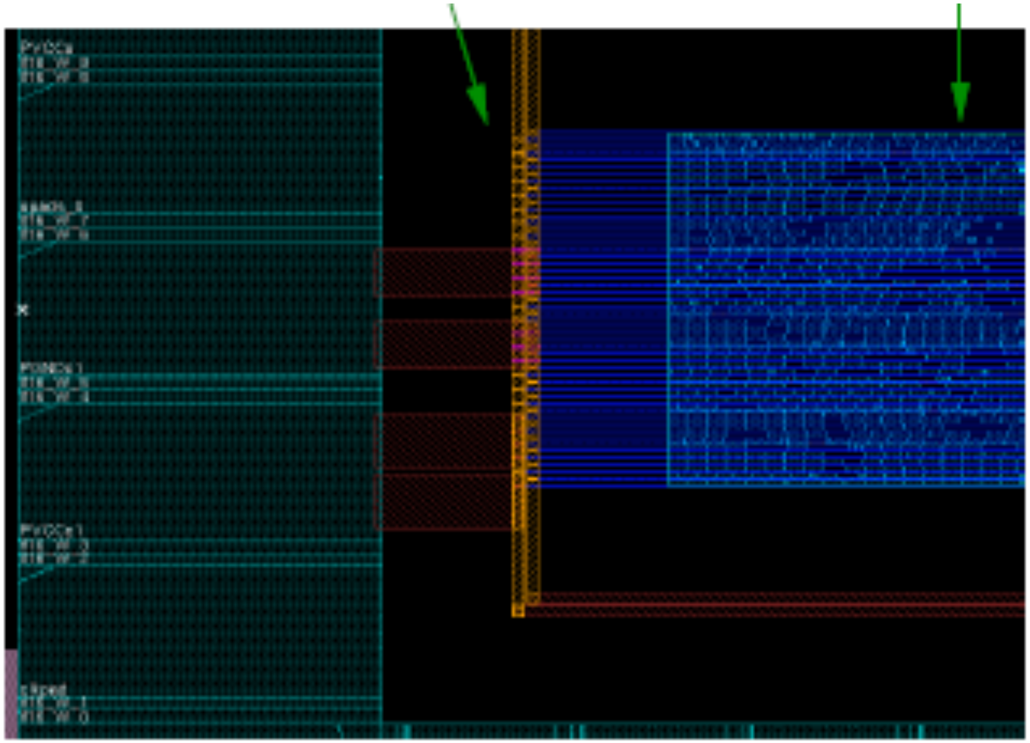


Choose upper layers (say metal 3 or 4)

Connecting Power (sRoute)

between

- IO power pins within IO rows
- CORE ring wires and the IO power pins
- stripes and core rings
- block power pins and the CORE ring wires

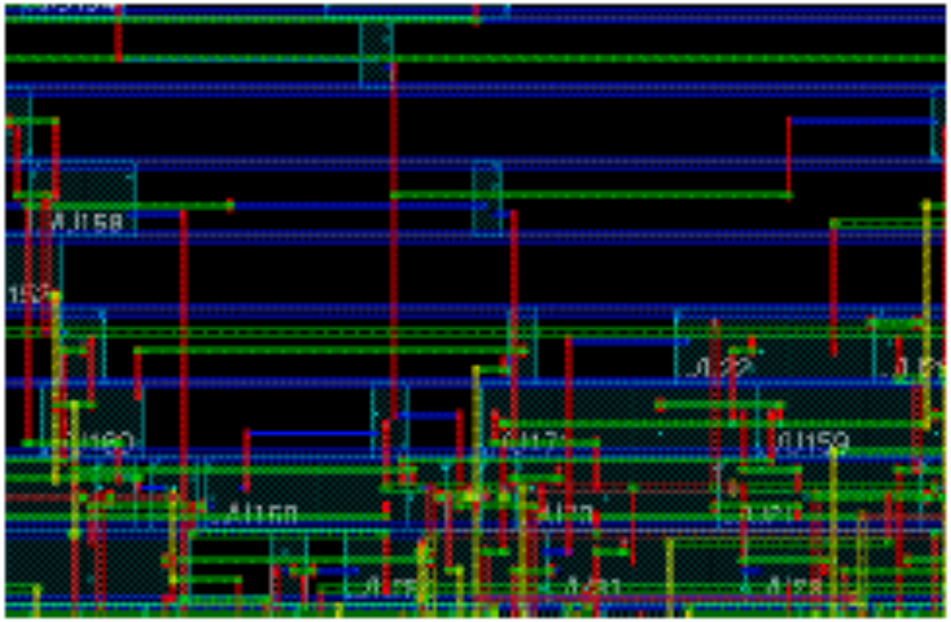


Route-> Special Route

Cell Placement

- Initial cell placement
- Moves, swaps changes orientation of cells to minimize required wire length
- Optimizes for wire length and net crossings
- A post CTS optimization may be carried out to optimize the design

Place -> Standard Cells

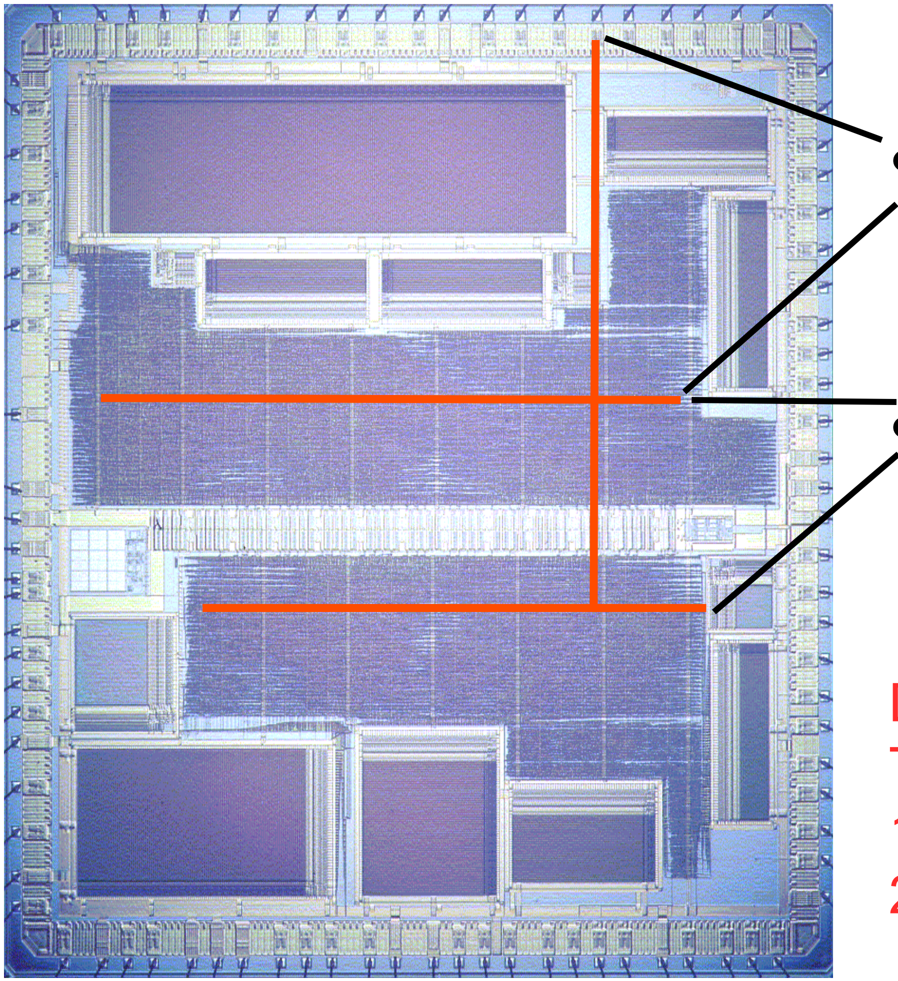


Clock Tree Synthesis

- Clockpad and output need to defined in a specification file.
  - clockpad/O

- Clock tree is synthesized and routed with highest priority to minimize clock skew.

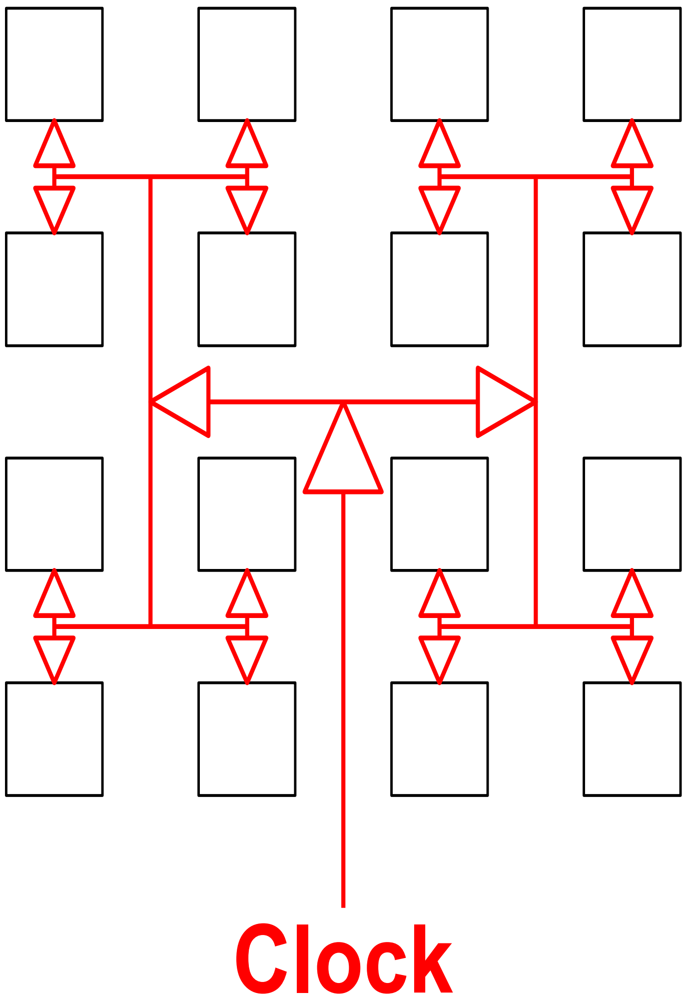
Clock Skew



- Absolute Skew
  -Delay from input to leaf cell
- Relative Skew
  -Delay difference between leaf cells

Danger!
Too much clock skew may:
1) Force you to reduce clock rate
2) Cause malfunction at any clock rate

Distributed Buffers in H-tree



Small relative skew

Absolute skew of less importance

CTS commands

- `create_clock -period` *`value`* `-name` *`clk_name`* `-add [get_ports clk]`

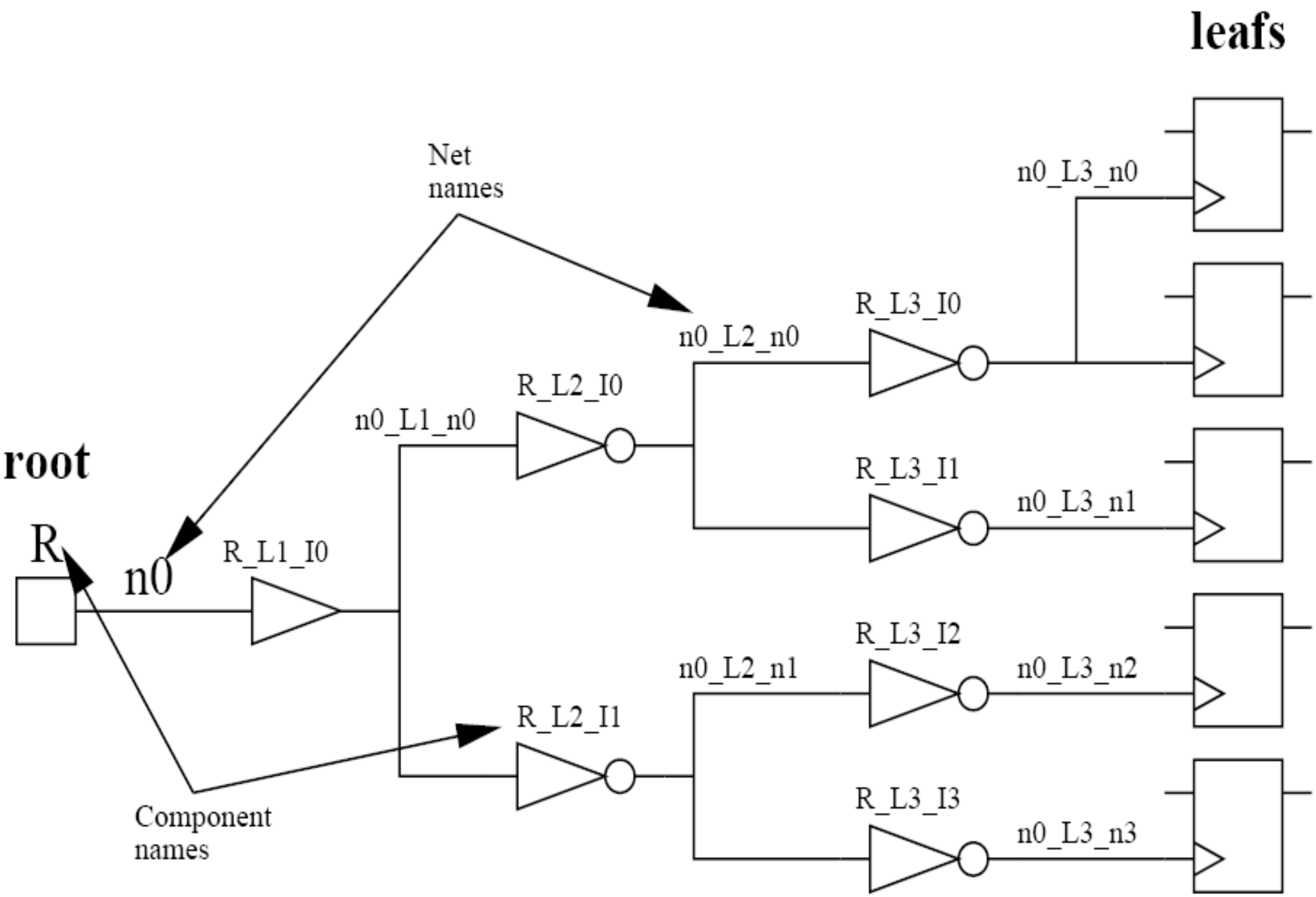
- Generate Clock tree specification

`createClockTreeSpec -output` *`file_name.ctstch`*
`-routeClknet -buffer` *`buffer_list`*

- Specify CTS file and synthesize clock tree.

`specifyClockTree -clkfile` *`file_name.ctstch`*
`clockDesign -specFile` *`file_name.ctstch`* `-clk` *`clk_name`*
`deleteTrialRoute`

Synthesized Clock tree

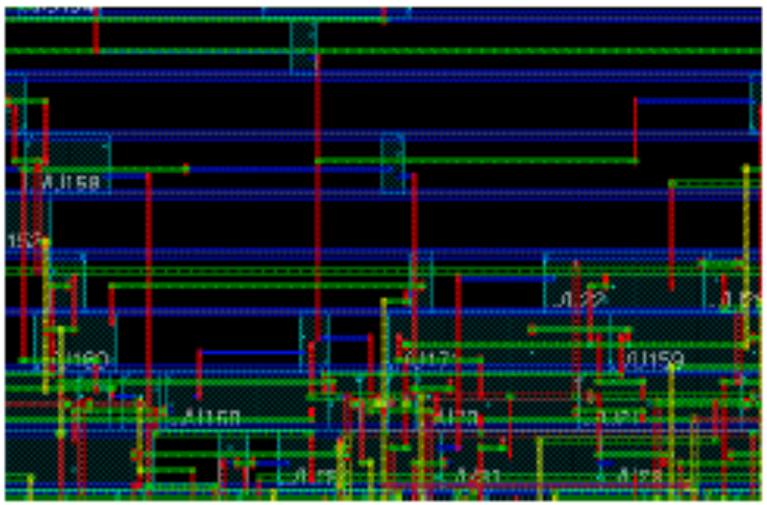


Clock buffers are placed in the core row gaps

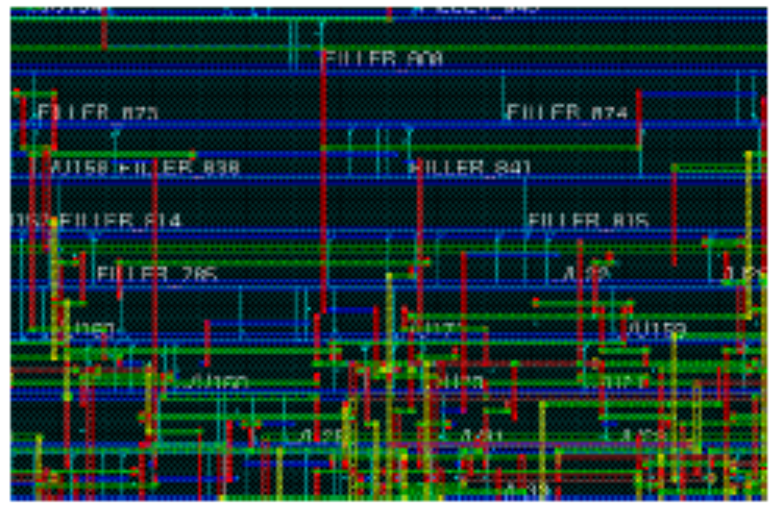
Core filler cell

Core filler cells ensure the continuity of power/ground rails and N+/P+ wells in the row.

Filler cells will close any gap it is important to perform CTS before filler cell placement.



before



after

Place -> Filler -> Add filler

Place -> Filler -> Add IO filler

Signal Routing

- Signal routing
  - Connects cells according to netlist
  - Metal wires are connected over several layers

- Routing time is strongly dependent on the design complexity

Route -> Nano Route

Verification and Tapeout

Verification (in SoCEnc)

- Connectivity, Antenna ....

Export

- Verilog (netlist) } Post-layout simulation
- sdf (timing)
- GDS II } tapeout

Verify

Routing Script

- Each command is automatically written in a script file **encounter.cmd**
- Script needs to be trimmed (remove unnecessary commands)
- Easy to change parameters
- Can be reused with modifications
- Time to do PnR iteratively is reduced
- Serves as documentation and makes it possible to repeat the flow

What's next?

- Continue in the lab with Assignment 1
  - The design needs to be taken through
    - Simulation, including post-synthesis
    - Synthesis
    - Place and Route