Digital VLSI Design

Lecture 5: Moving to the Physical Domain

Semester A, 2016-17

Lecturer: Dr. Adam Teman

12/12/2016

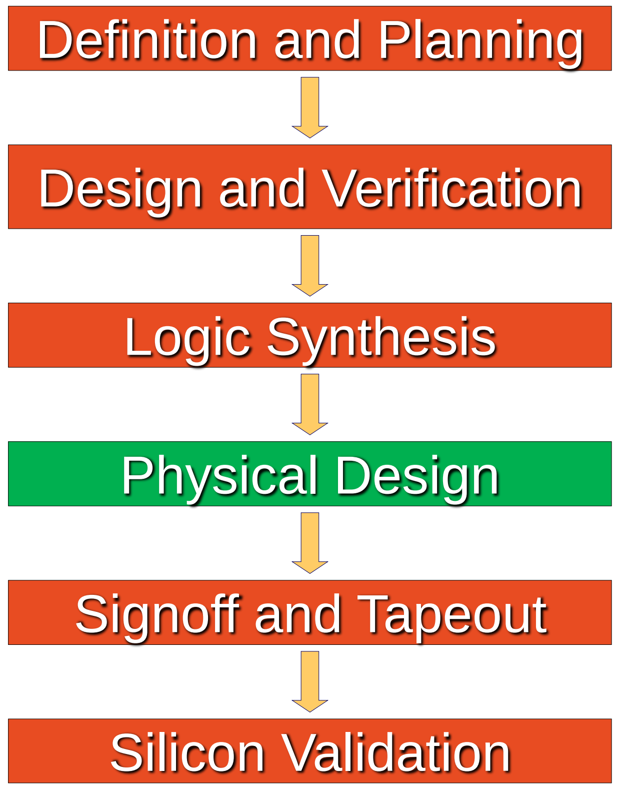
EnICS
Emerging Nanoscaled Integrated Circuits and Systems Labs

Bar-Ilan University
אוניברסיטת בר-אילן

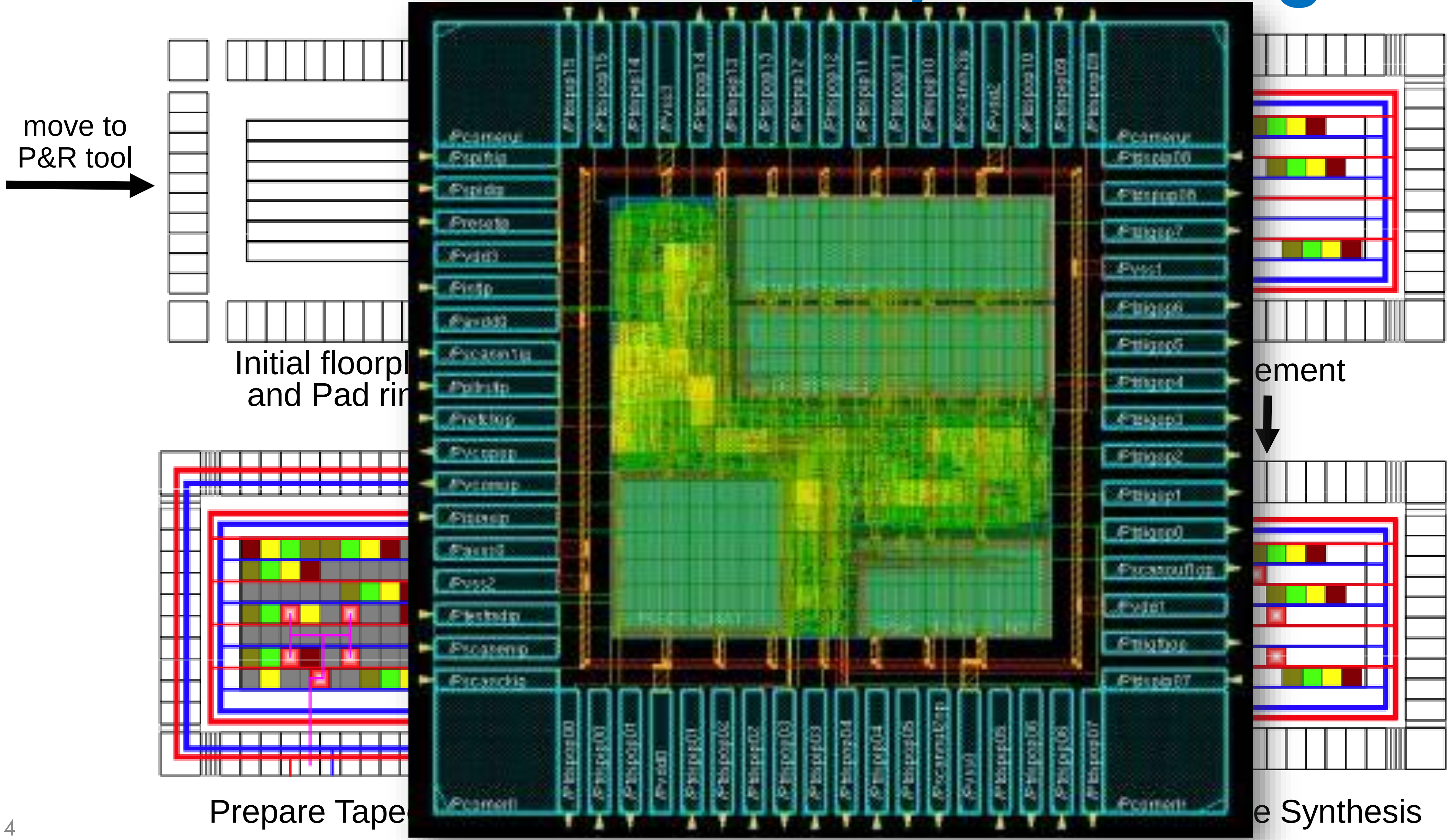
Moving to Physical Design

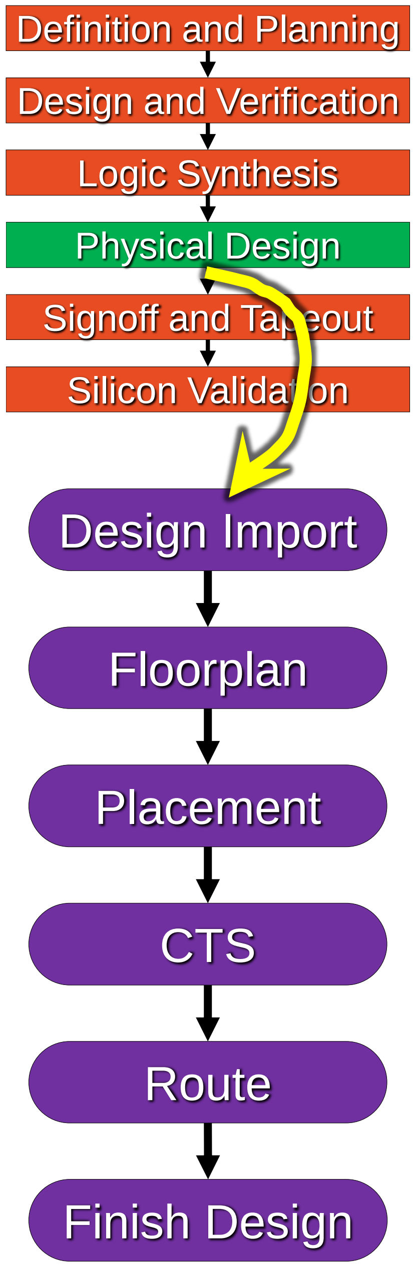
So, what’s next?

- **We’ve basically finished the *Front-End* of the design process and we will now start the *Back-End*:**
  - To start, we will move between tools with a *logical* approach to ones with a *physical* approach to design implementation.
  - Then, we will make a physical foundation for our design by drawing up a *floorplan*.
    - This will include making decisions where “big” or “important” pieces will sit, such as IPs, I/Os, Power grids, special routes, etc.
  - After that, we can *place* our gates taking into account *congestion* and *timing.*
  - With our flip-flops in place, we can go about designing a *clock-tree*.
  - And finally, we can *route* all our nets, according to *DRCs*, *timing*, *noise*, etc.
  - Before *tapeout*, we will clean things up, verify, etc.



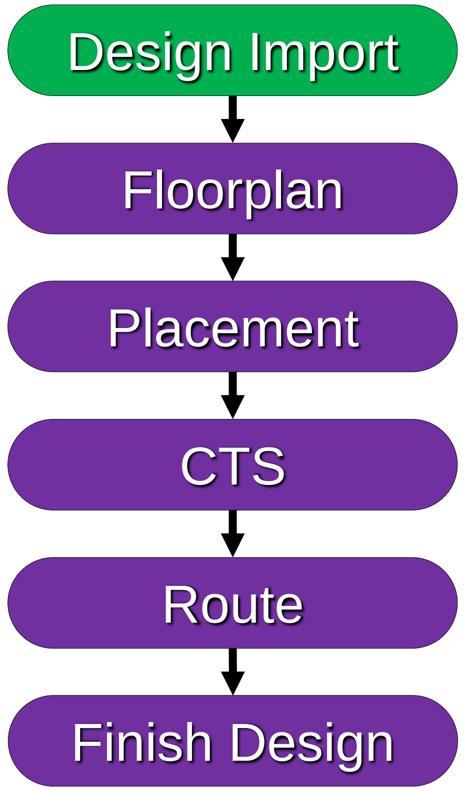
An illustrative view of Physical Design

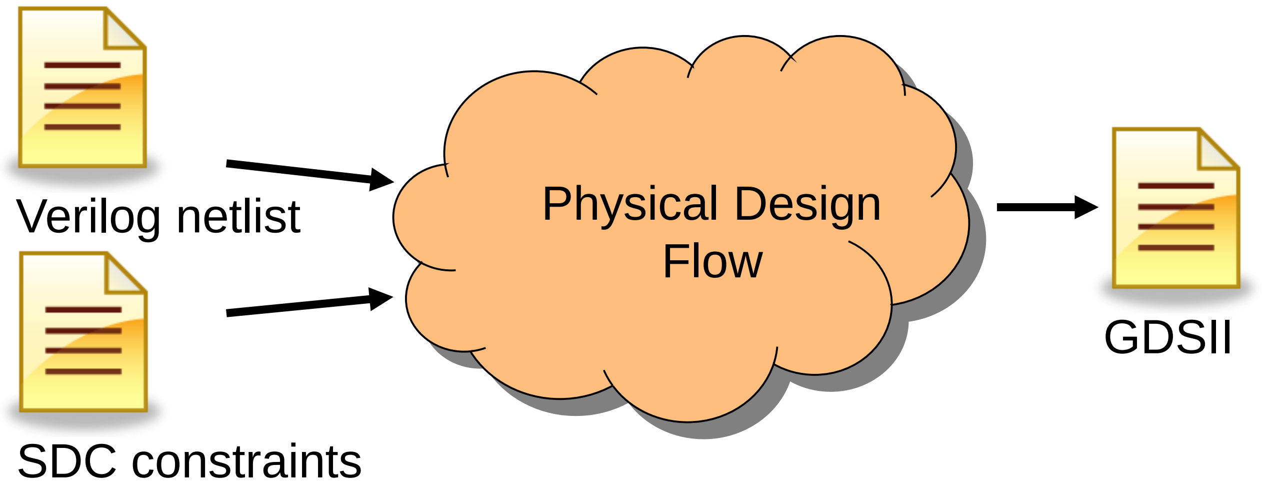




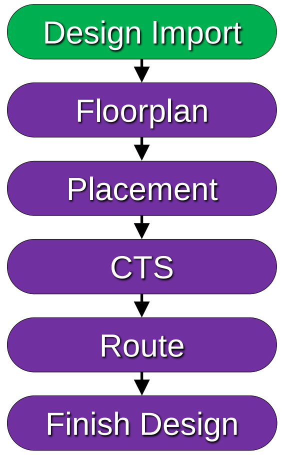
Moving from Logical to Physical

- Define design (.v)
- Define design constraints/targets (.sdc)
- Define operating conditions/modes (MMMC)
- Define technology and libraries (.lef)
- Define physical properties (Floorplan)

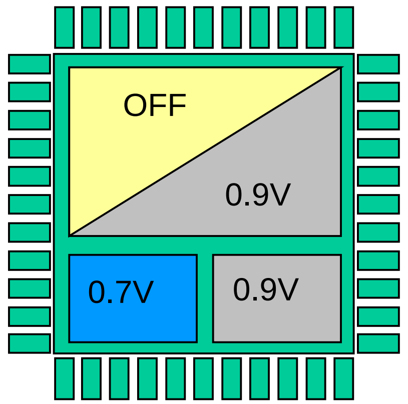




Moving from Logical to Physical

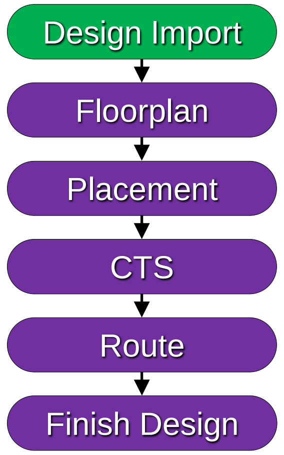


- **During synthesis, our world view was a bit idealistic.**
  - We didn't care about power supplies.
  - We didn't care about physical connections/entities.
  - We didn't care about clock non-idealities.
- **Therefore, in order to start physical implementation:**
  - Define "global nets" and how they connect to physical instances.
  - Provide technology rules and cell abstracts (.lef files)
  - Provide physical cells, unnecessary for logical functionality:
    - Tie cells, P/G Pads, DeCaps, Filler cells, etc.
  - Define hold constraints and all operating modes and conditions (MMMC)
    - Hold was "easy to meet" with an ideal clock, so we didn't really check it…
  - Set up "low power" definitions, such as voltage domains, power gates, body taps, etc.



MV with power gating (shut down)

More than one operating mode

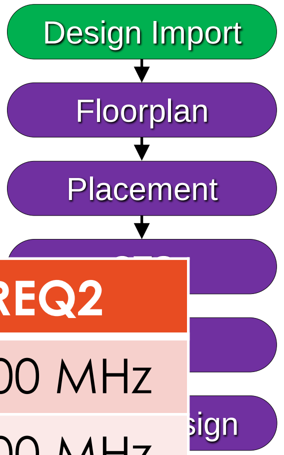


- **During synthesis, we (usually) target timing for a worst-case scenario.**
  - But, what is "worst-case"?
  - Intuitively, that would be a slow corner, (i.e., SS, VDD-10%, 125C)
  - No need for hold checking, since clock is ideal (No skew = No hold)
- **But, what if there is an additional operating mode?**
  - For example, a test (scan) mode.
  - Do we have to close timing at the same (high) clock speed?

Mode	TEST_MODE	FREQ
Functional	0	1 GHz
Test	1	10 MHz

- **No problem, we'll just deal with both modes separately**
  - Prepare an additional SDC and rerun STA/optimization.

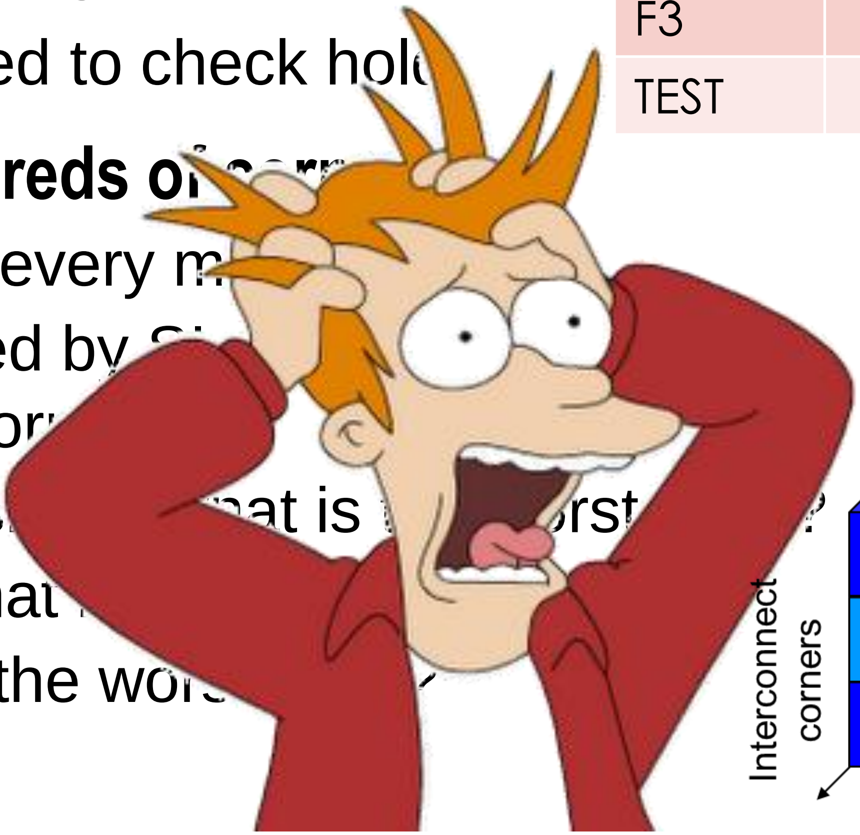
```
set_case_analysis 1 [get_ports TEST_MODE]
create_clock -period [expr TCLK/100] -name TEST_CLK [get_ports TEST_CLK]
```

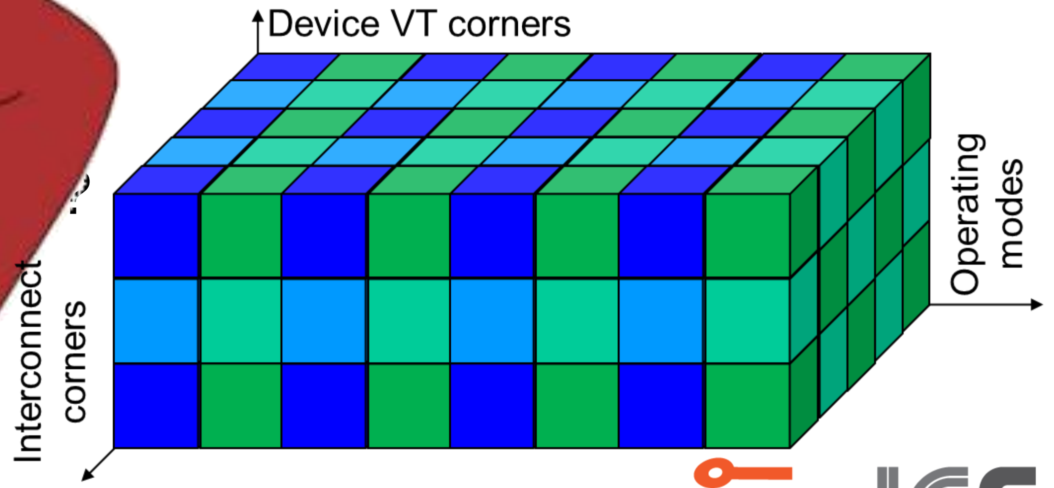


Many, many, corners…

- **But real SoCs are much more complex:**
  - Many operating modes.
  - Many voltage domains.
  - With real clock, need to check hol
- **We easily get to hundreds of corn**
  - Setup and hold for every m
  - Hold can be affected by Si check hold for all cor
  - Temperature invers at is rst ?
  - RC Extraction – what
  - Leakage – what is the wor ?
- **Aaaaarrrrrgggghhhh!**

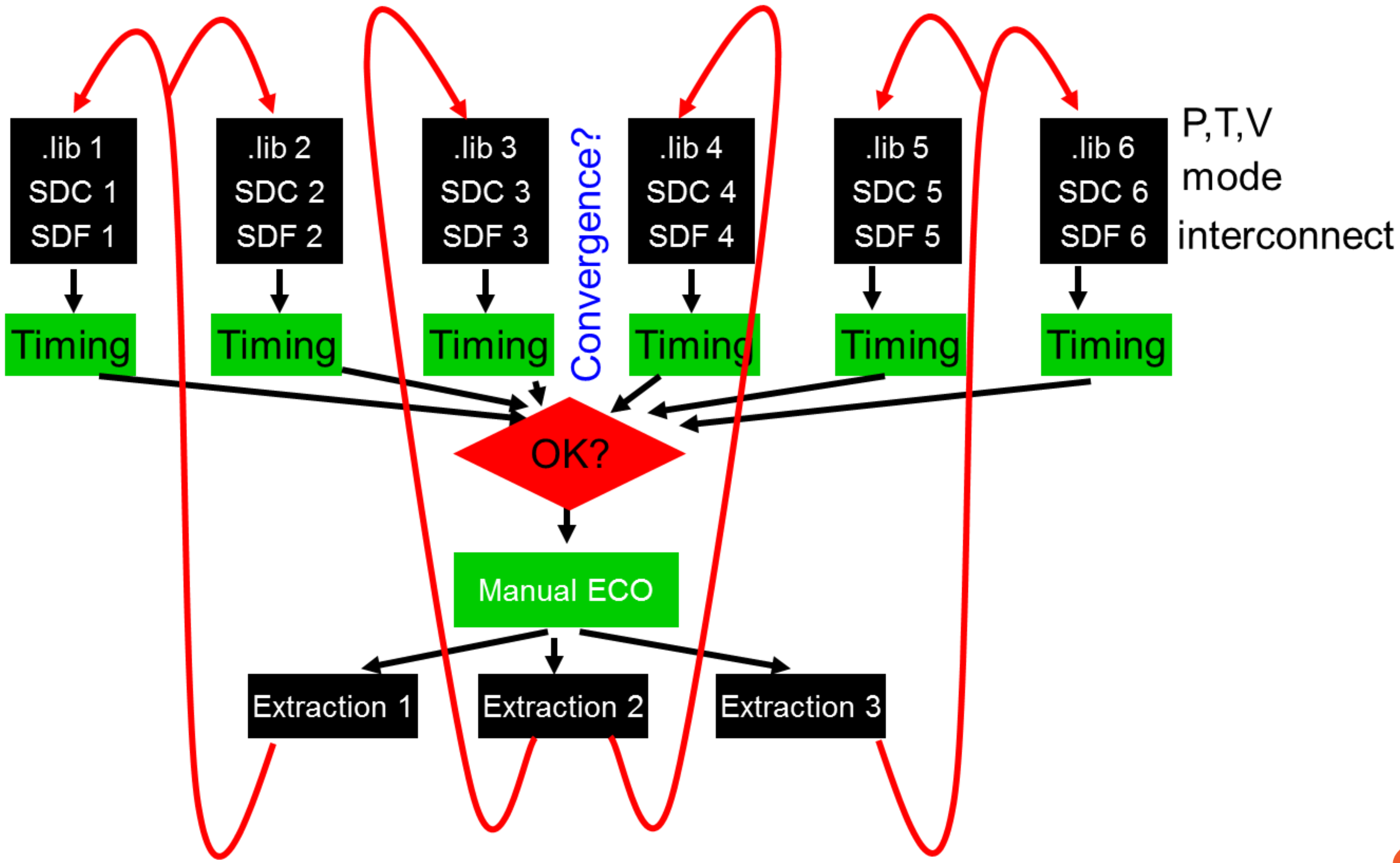
Mode	VDD1	FREQ1	VDD2	FREQ2
F1	1.2 V	2 GHz	0.8 V	500 MHz
F2	0.8 V	400 MHz	0.8 V	400 MHz
F3	Off	Off	0.5 V	50 MHz
TEST	1.2 V	10 MHz	1.2 V	10 MHz

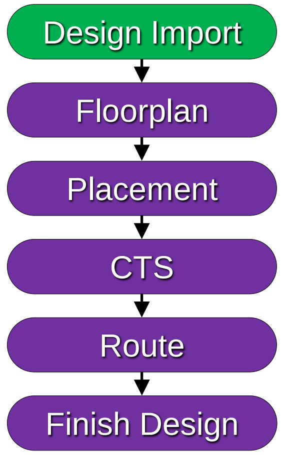




The corner crisis

- Traditional approach not feasible



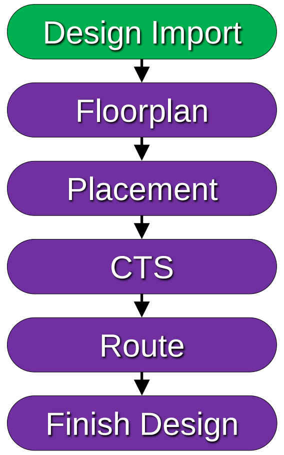


1 Moving to Physical Design | **2 MMMC** | 3 MSV Design | 4 Floorplanning | 5 Hierarchical Design | 6 Power Planning

Multi-Mode Multi-Corner

Or how to deal with the corner crisis!

Multi-Mode, Multi-Corner



- **MMMC to the rescue!**
  - It's implemented in a slightly (!) confusing way, but it really simplifies things.
- **The basic concept is that we create *analysis views* that can then be selected for setup and hold (max and min) constraints.**

Analysis View = "Turbo"

VDD=1.2V
Freq=2 GHz
Corner=SS
Temp=125 C
OpMode="Turbo"

Analysis View = "Low Power"

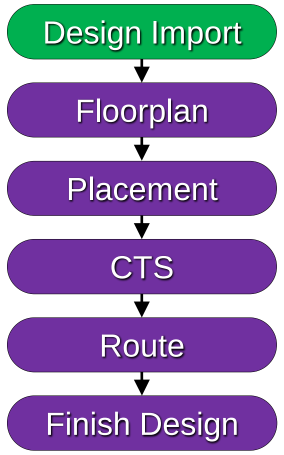
VDD=0.5V
Freq=50 MHz
Corner=SS
Temp=125 C
OpMode="LP"

Analysis View = "Turbo - Hold"

VDD=1.3V
Freq=2 GHz
Corner=FF
Temp= -40 C
OpMode="Turbo"

Setup checks: "Turbo" and "Sleep" Modes
Hold checks: "Turbo – Hold" Mode

Multi-Mode, Multi-Corner

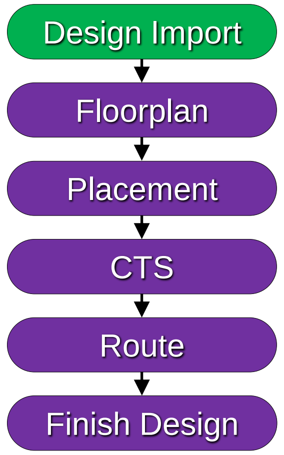


- **Defining Analysis Views is done in hierarchical fashion.**
  - An *analysis view* is constructed from a *delay corner* and a *constraint mode*.

```
create_analysis_view -name turbo \
       -constraint_mode turbo_mode -delay_corner slow_corner_vdd12
create_analysis_view -name low_power \
       -constraint_mode low_power_mode -delay_corner slow_corner_vdd05
create_analysis_view -name turbo_hold \
       -constraint_mode turbo_mode -delay_corner fast_corner_vdd13

set_analysis_view -setup {turbo low_power} -hold {turbo_hold}
```

  - A *delay corner* tells the tool how the delays are supposed to be calculated. Therefore it contains timing libraries and extraction rules.
  - A *constraint mode* is basically the relevant SDC commands/conditions for the particular operating mode.

Multi-Mode, Multi-Corner



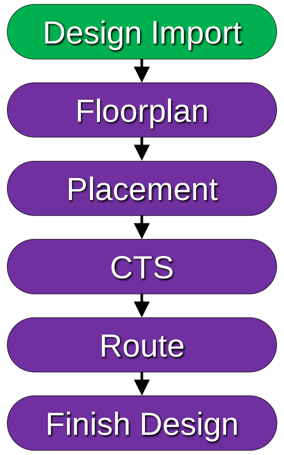
- **So now, let's define the lower levels of the MMMC hierarchy.**
  - A *constraint mode* is simply a list of relevant SDC files. When you move between analysis views, the STA tool will automatically apply the relevant constraints to the design.

```
create_constraint_mode -name turbo_mode -sdc_files {turbo.sdc}
create_constraint_mode -name low_power_mode -sdc_files {low_power.sdc}
```

  - A *delay corner* is a bit more complex. It comprises a *library set* an *RC corner* and a few other things that we won't discuss right now.

```
create_delay_corner -name slow_corner_vdd12 \
       -rc_corner {RCmax} -library_set {ss_1p2V_125C}
create_delay_corner -name slow_corner_vdd05 \
       -rc_corner {RCmax} -library_set {ss_0p5V_125C}
create_delay_corner -name fast_corner_vdd13 \
       -rc_corner {RCmin} -library_set {ff_1p3V_m40C}
```

Multi-Mode, Multi-Corner



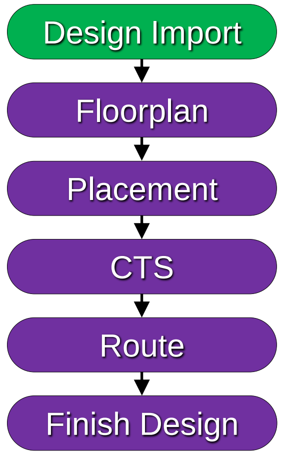
- **Confused yet? Well, we still have more to go.**
  - A *library set* is a collection of the .lib characterizations that should be used for timing the relevant gates. This includes the standard cells and other macros, such as RAMs and I/Os. There also may be special “SI” characterizations for noise.

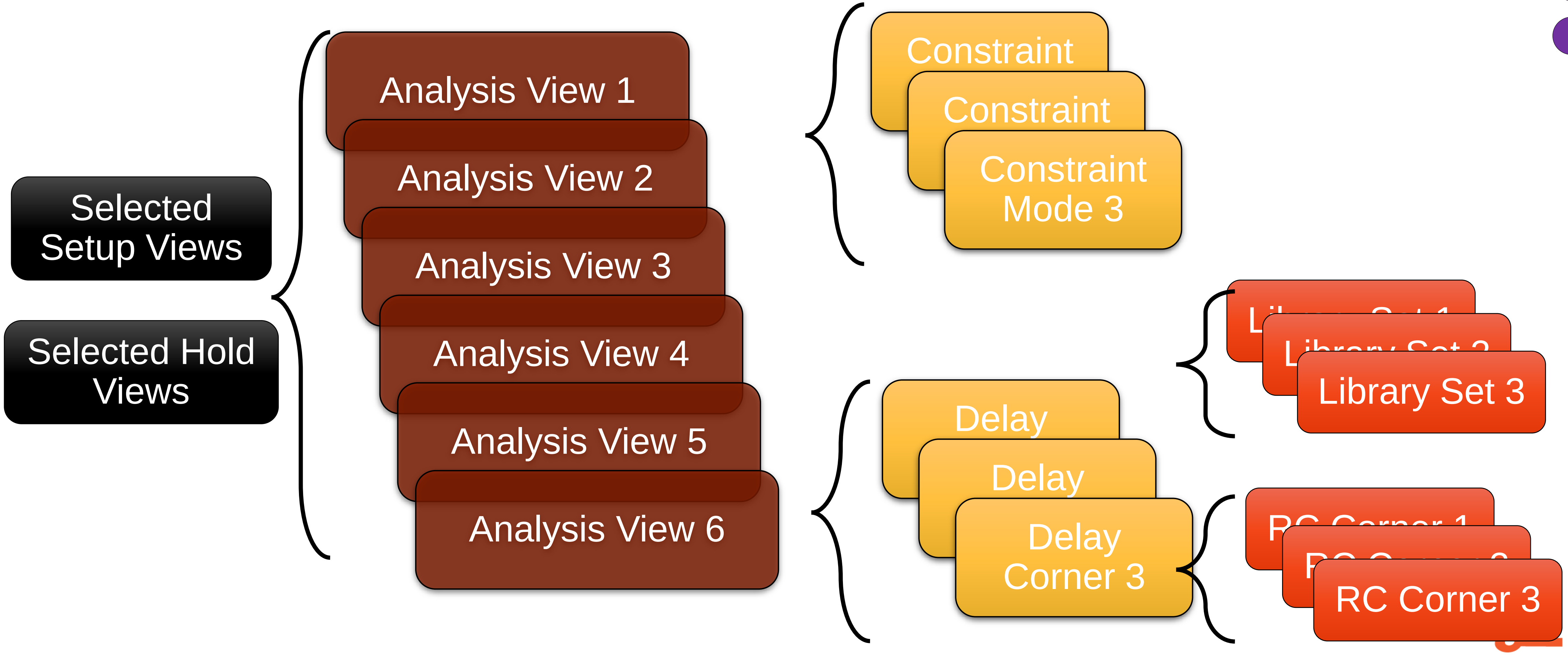
```
create_library_set -name ss_1p2V_125C \
       -timing [list ${sc_libs}/ss_1p2V_125.lib ${mem_libs}/ss_1p2V_125.lib \
        ${io_libs}/ss_1p8V_125.lib] –si ${sc_libs}/ss_1p2V_125.si
```

  - And finally, an *RC corner* is a collection of the rules for RC extraction. There may be a “capacitance table” for quick extraction and a “QRC rulefile” for accurate extraction. The temperature is also defined in the RC corner, but it is taken into account in the .lib file, as well.

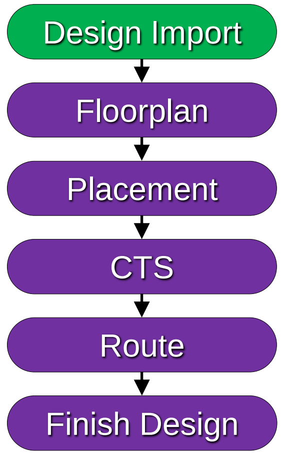
```
create_rc_corner -name RCmax -cap_table ${tech}/RCmax.CapTbl} -T {125} \
       -qx_tech_file ${tech}/RCmax.qrctech
```

Multi-Mode, Multi-Corner - Summary

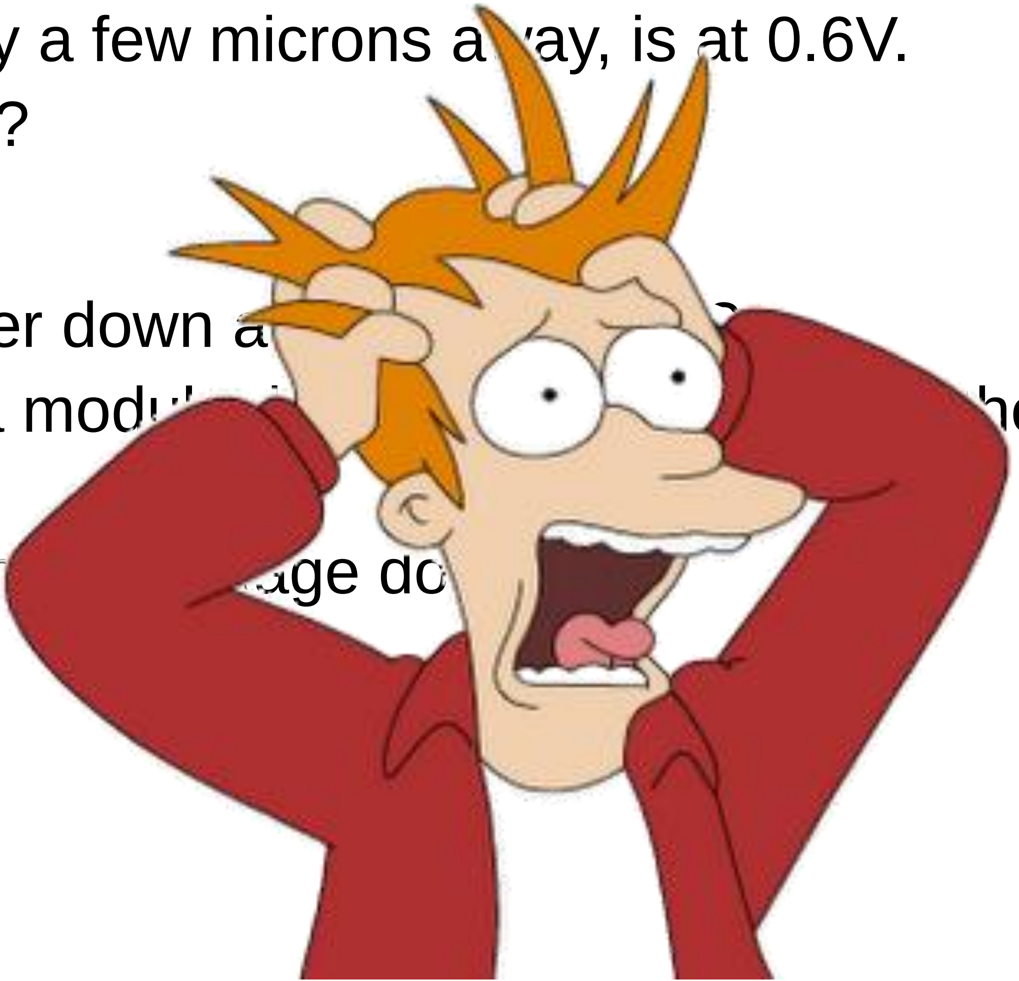




...So you think that was complicated?



- **What if I have multiple voltage domains?**
  - Now, for example, in a certain operating mode, one inverter is operated at 1.2V, while another one, only a few microns a ay, is at 0.6V.
  - How do I define *that* library set?
- **Even worse…**
  - What happens if I want to power down a ?
  - What if I want to power down a modu he value stored in the flip flops)?
  - How do I transfer data between age do
- **Arrrrrggghhhh!**

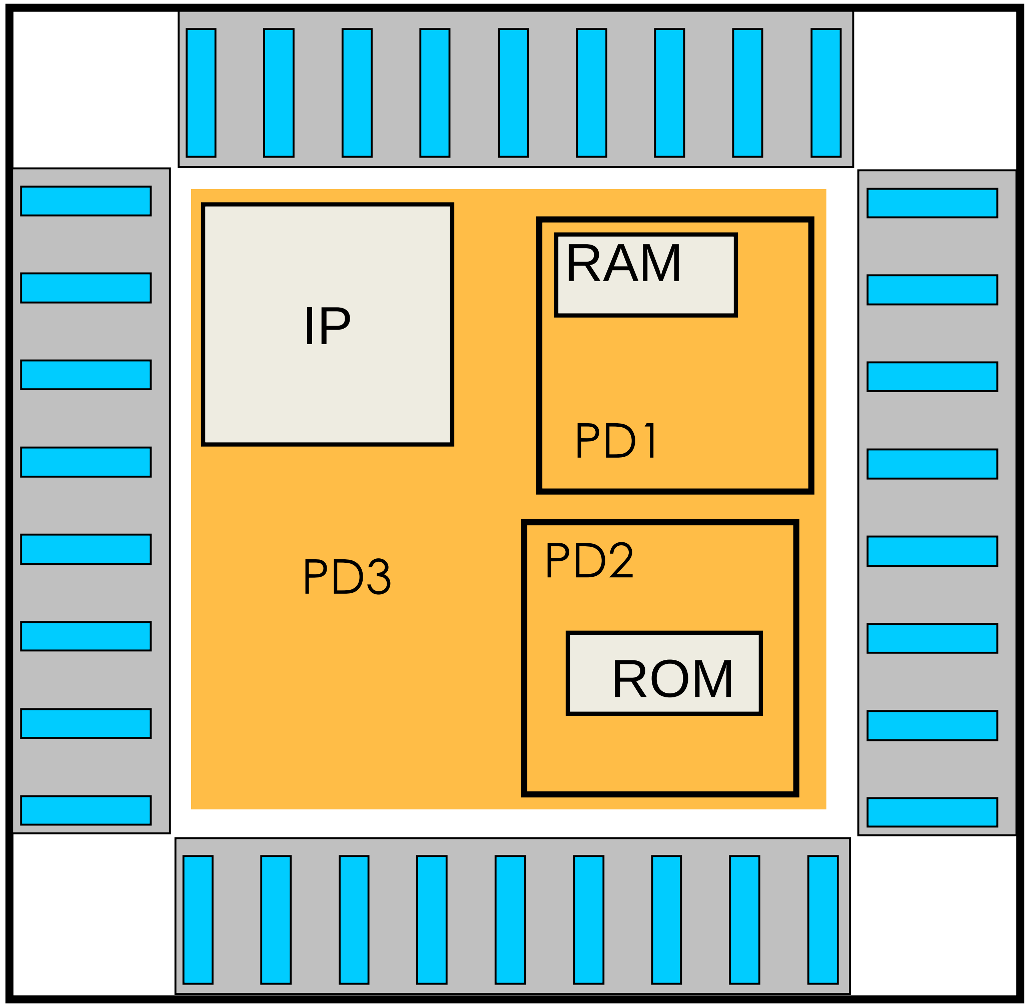


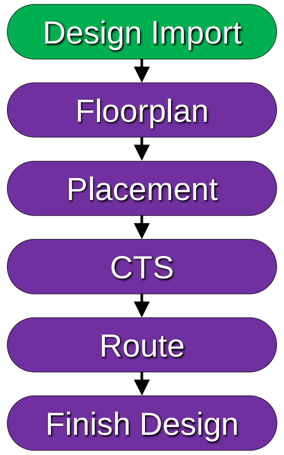
A bit about Multiple Voltage Domains

Often referred to as "Low Power Design" Methodologies

Multiple Domain Design

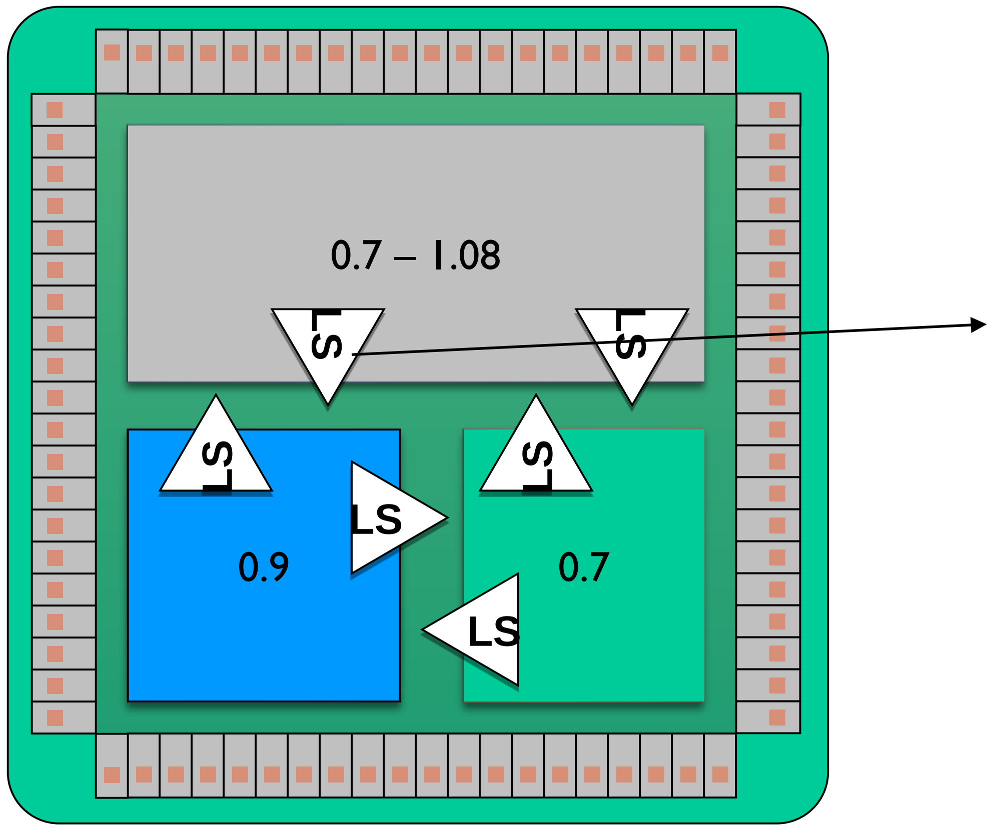
- **Define power domains**
  - create power domain names
  - list of cells connected to VDD1, VDD2,GND1,…
  - draw the power domains
- **Place macros**
  - Take into account:
    - routing congestion
    - orientation
  - Manual usually better then auto
- **Place switches**
  - For the power down domains

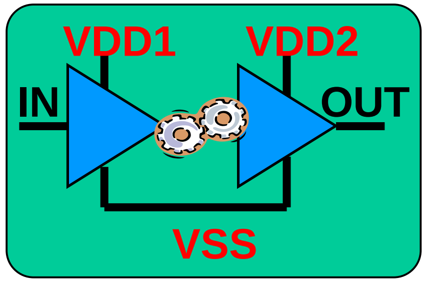




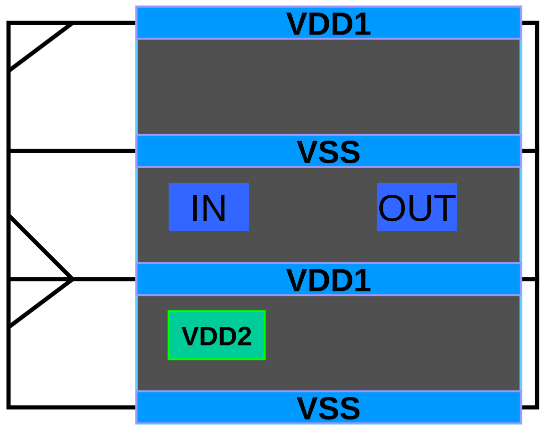
Design Import
Floorplan
Placement
CTS
Route
Finish Design

Multiple Domain Design – Level Shifters



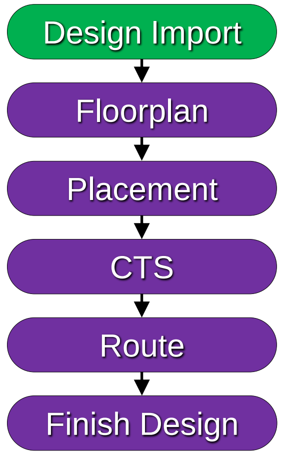


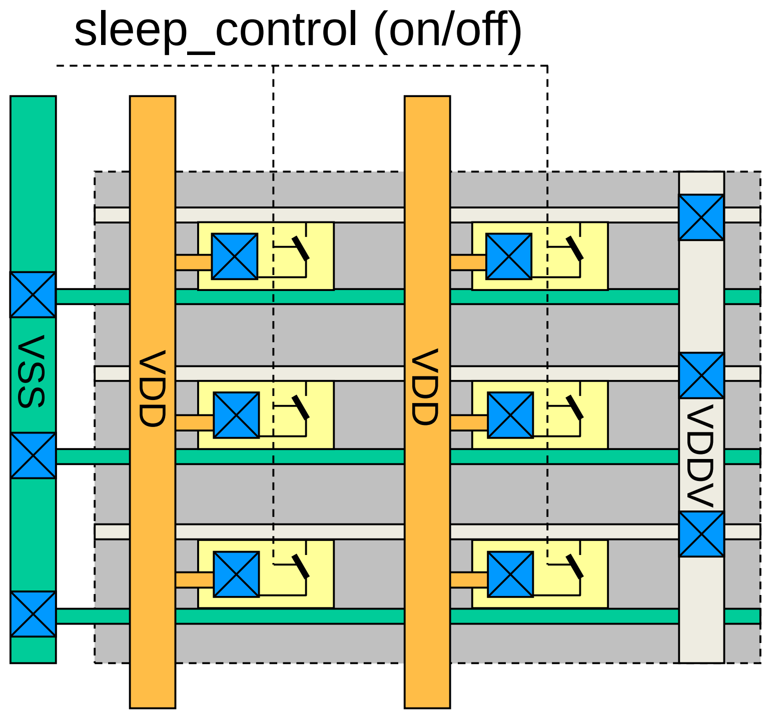
logic model

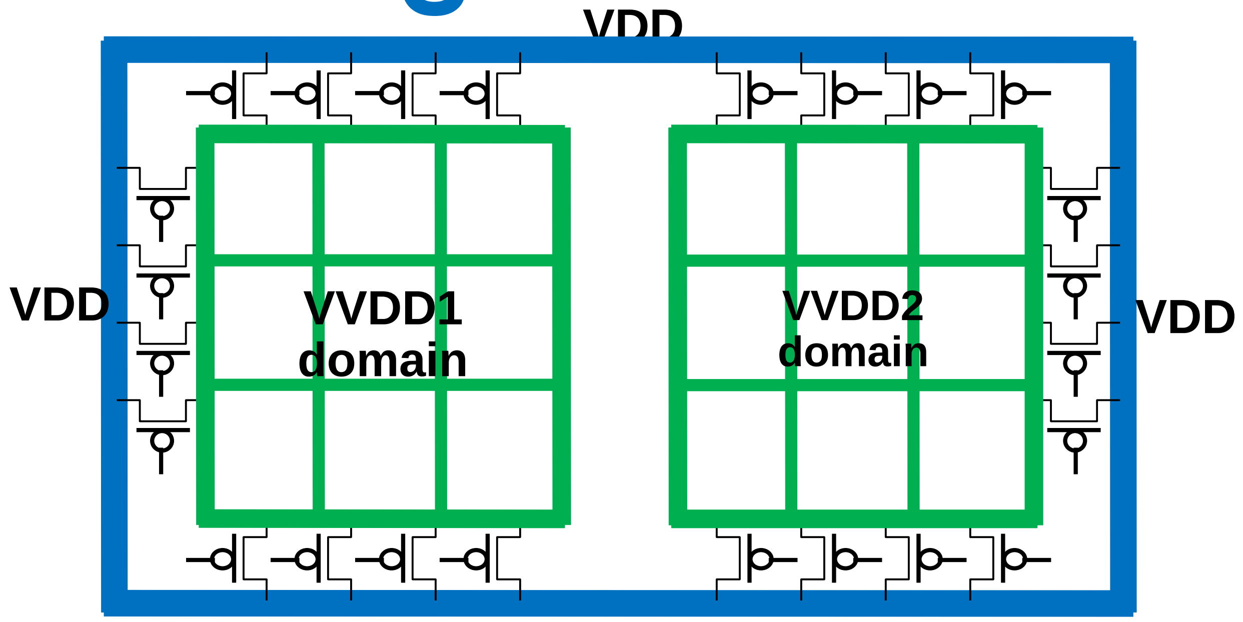


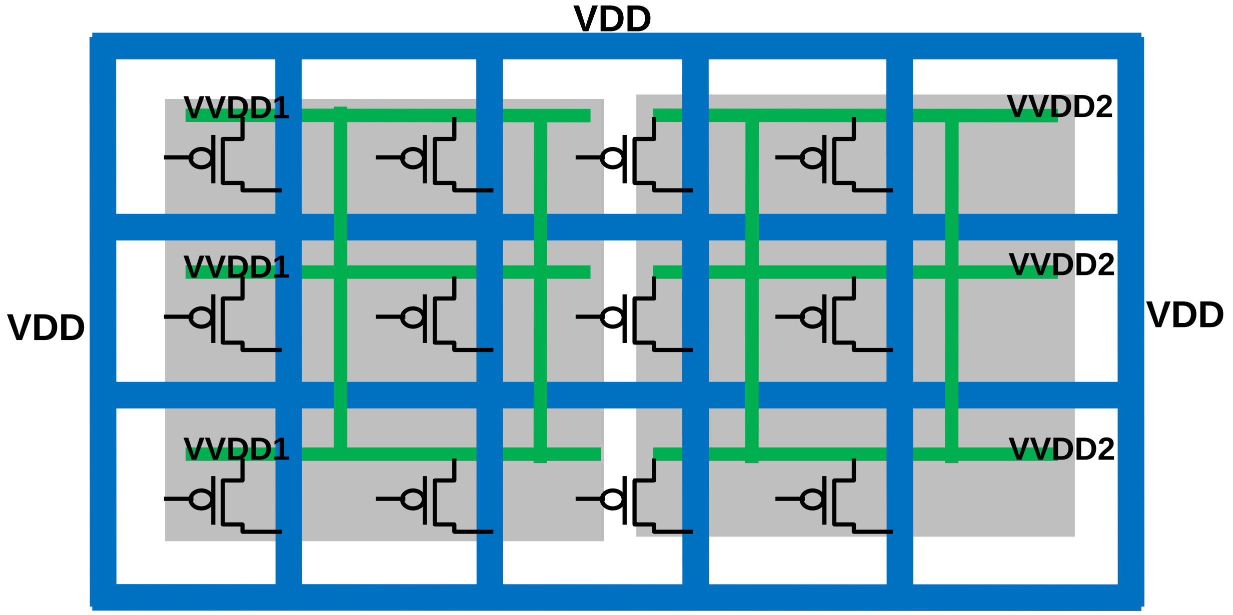
a possible FP view

Multiple Domain Design – Power Gating

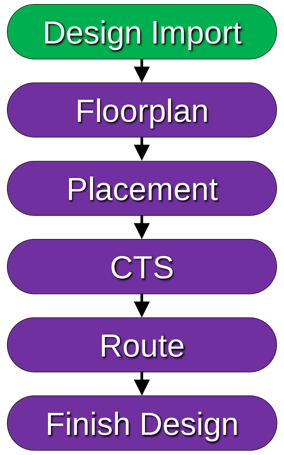








How do we define this?

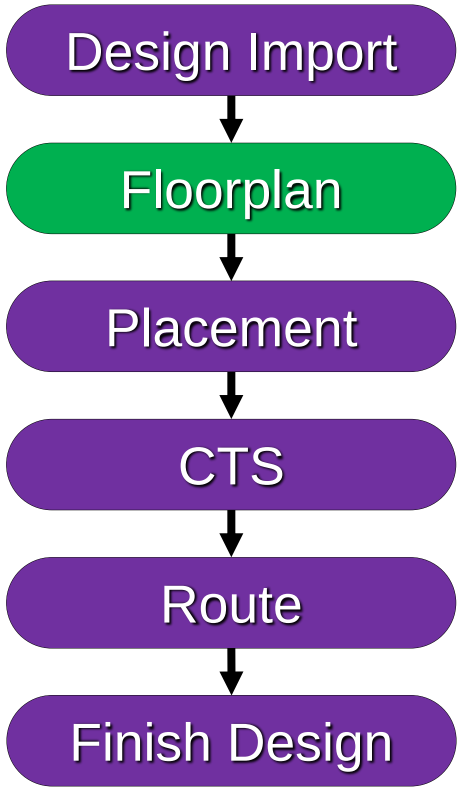


- **Well, we probably will leave that to an advanced course or your MSc research…**
- **But, in general, there is a command format for this.**
  - Well, actually two.
  - Cadence calls theirs **CPF** (Common Power Format), and it’s surprisingly (…confusingly) similar to MMMC.
  - Synopsys calls theirs **UPF** (Unified Power Format), and it’s surprisingly similar to SDC.
  - I guess neither is very *common* or *unified*…
- **Luckily for you, we will not talk any more about this right now ☺.**
- **Instead, we’ll start with the basics of Floorplanning.**

Floorplanning

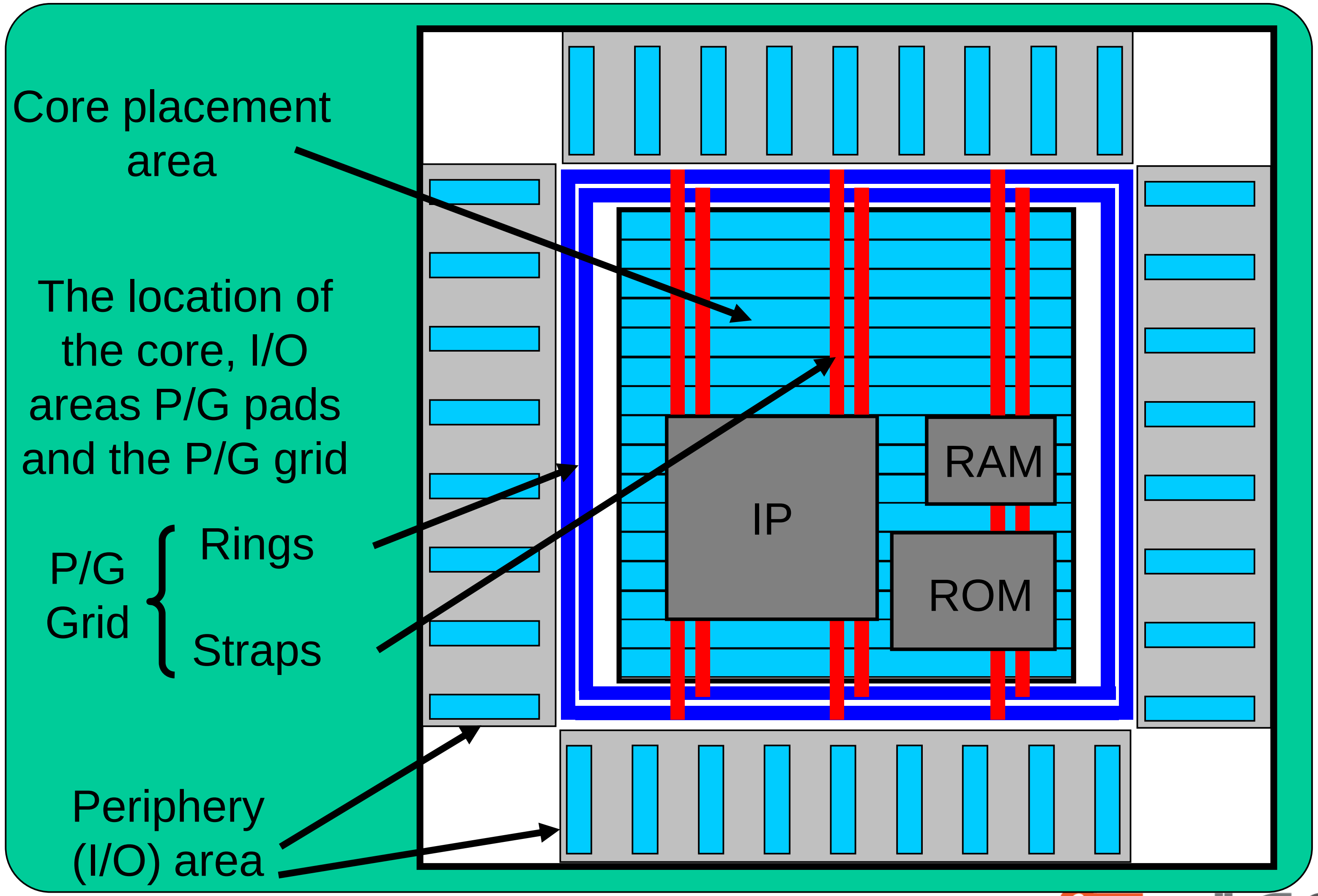
Floorplanning Goals and Objectives

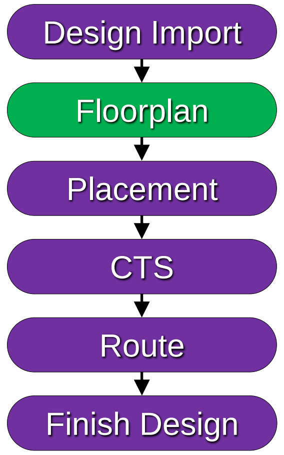
- **Floorplanning is a mapping between the logical description (the netlist) and the physical description (the floorplan).**
- **Goals of floorplanning:**
  - Arrange the blocks on a chip,
  - Decide the location of the I/O pads,
  - Decide the location and number of the power pads,
  - Decide the type of power distribution
  - Decide the location and type of clock distribution.
- **Objectives of floorplanning are:**
  - Minimize the chip area
  - Minimize delay
  - Minimize routing congestion



Fullchip Design Overview

- Chip size
- Number of Gates
- Number of Metal layers
- Interface to the outside
- Hard IPs/Macros
- Power Delivery
- Multiple Voltages
- Clocking Scheme
- Flat or Hierarchical?

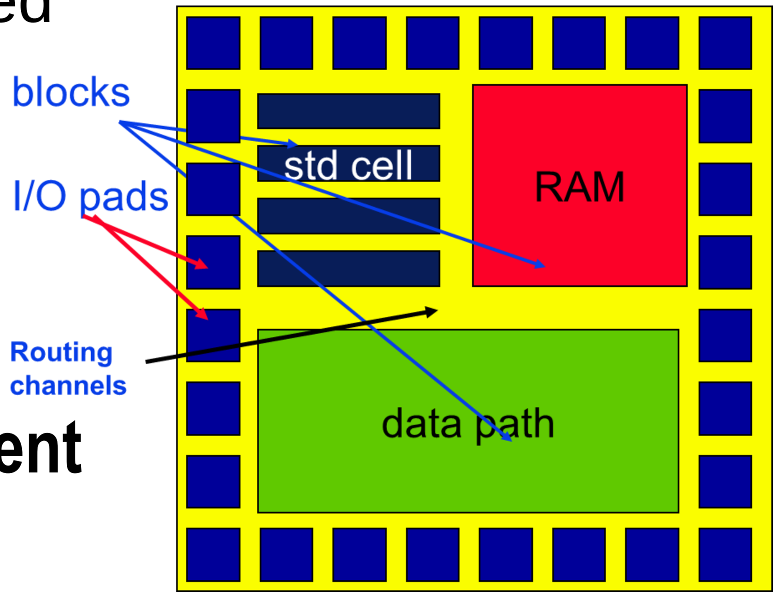




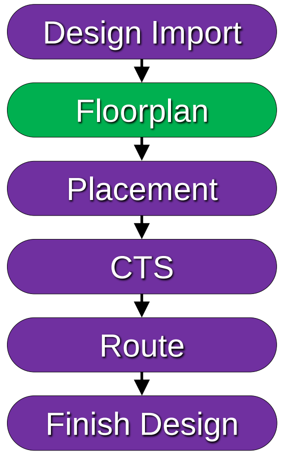
Floorplanning Inputs and Outputs

- **Inputs**
  - Design netlist (required)
  - Area requirements (required)
  - Power requirements (required)
  - Timing constraints (required)
  - Physical partitioning information (required)
  - Die size vs. performance vs. schedule trade-off (required)
  - I/O placement (optional)
  - Macro placement information (optional)

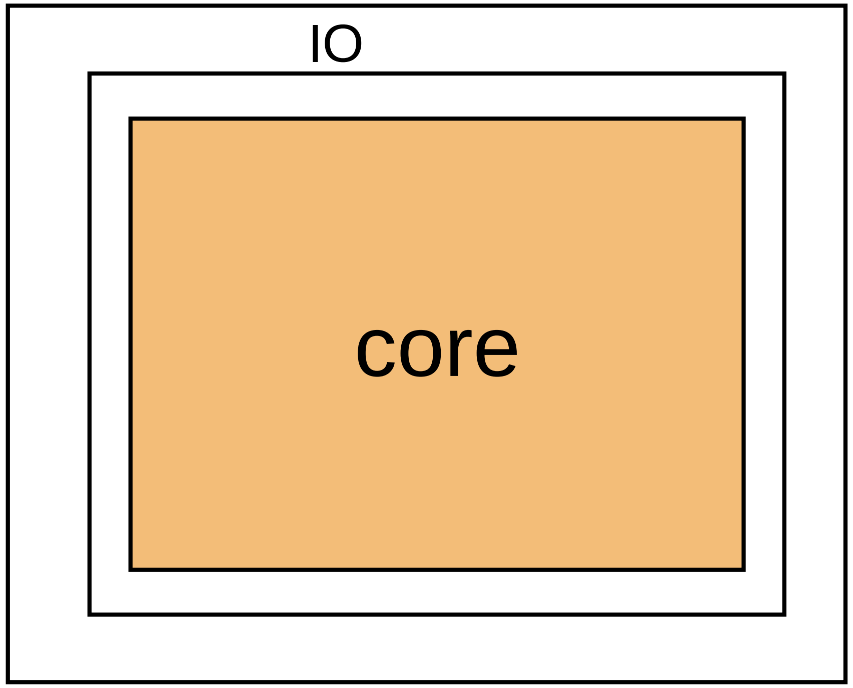
- **Outputs**
  - Die/block area
  - I/Os placed
  - Macros placed
  - Power grid designed
  - Power pre-routing
  - Standard cell placement areas
- **Design ready for standard cell placement**



IO Ring

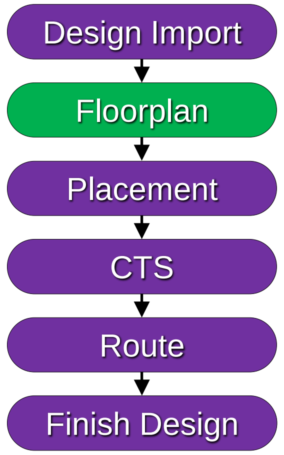


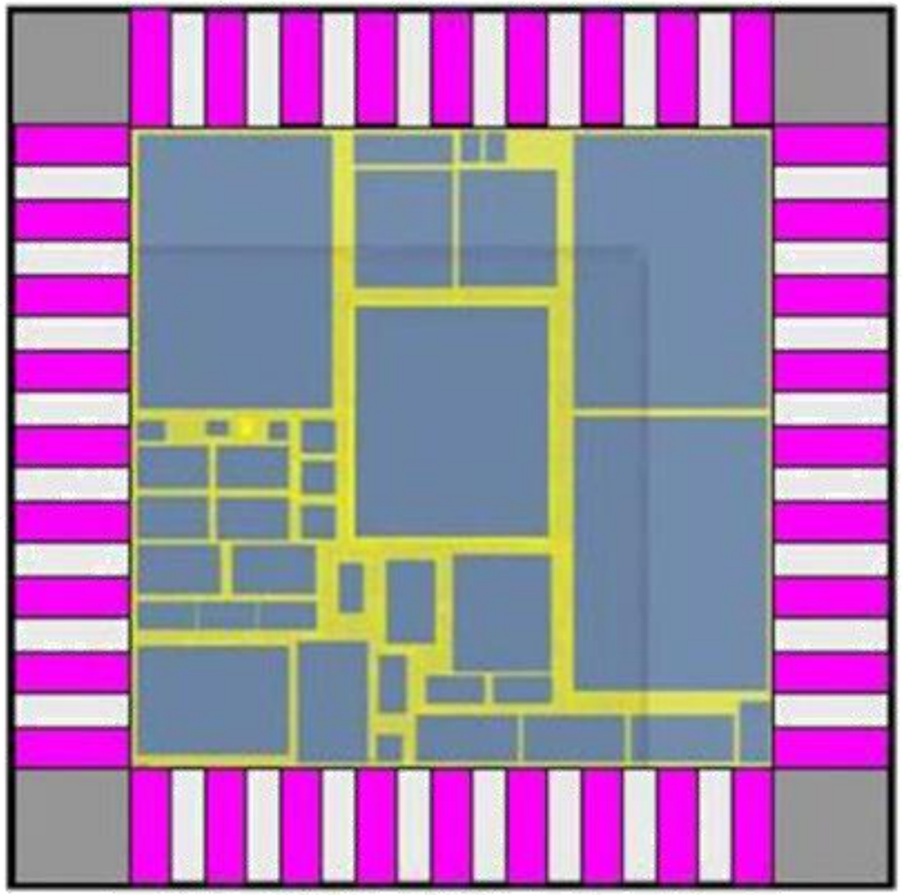
- **The pinout is often decided by front-end designers, with input from physical design and packaging engineers.**
  - I/Os do not tend to scale with Moore's Law and therefore, they are very expensive (in terms of area).
  - I/Os are not only needed for connecting signals to the outside world, but also to provide power to the chip.
  - Therefore, I/O planning is a critical and very central stage in Floorplanning the chip.

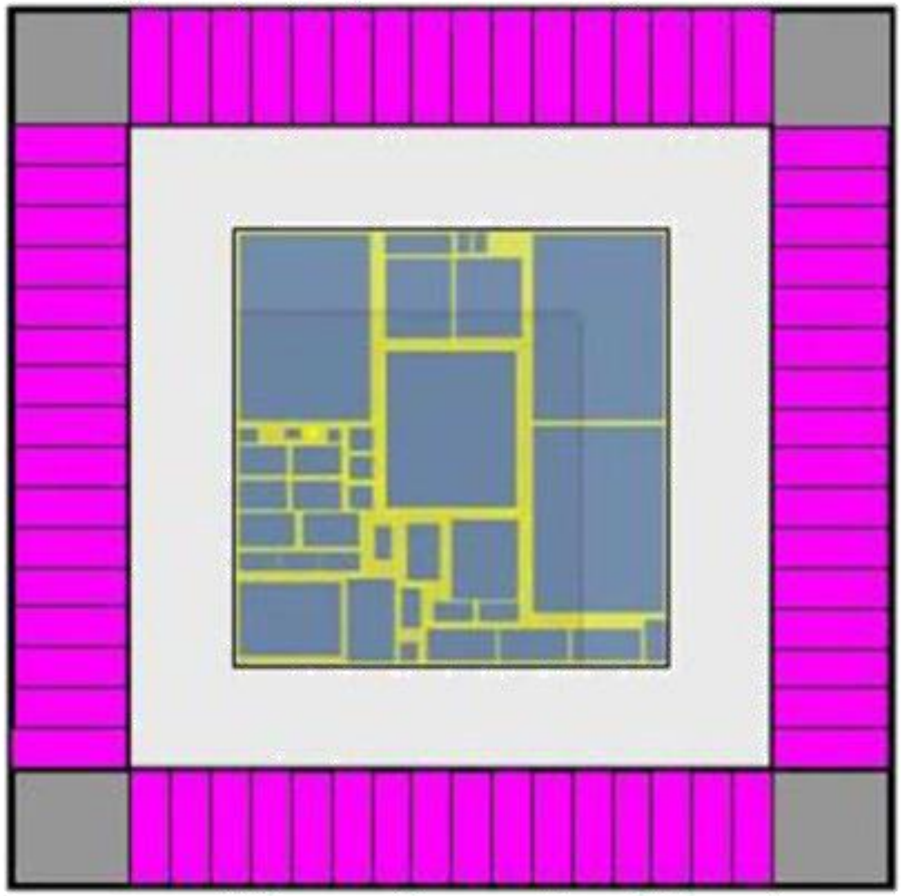


- **Let's leave it at that for a bit, and revisit the I/Os a little later…**

How do we choose our chip size?

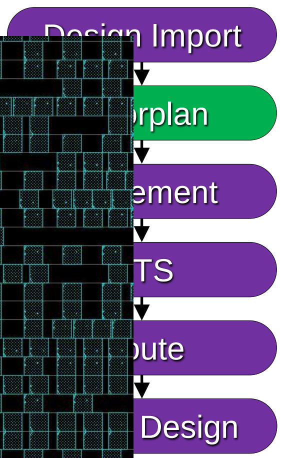


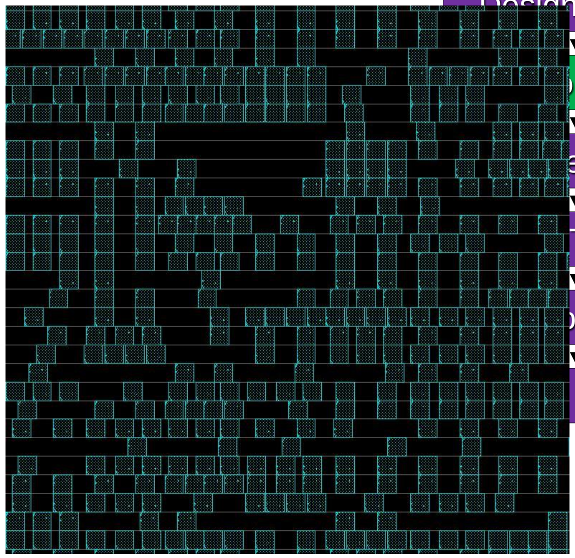

Core Limited

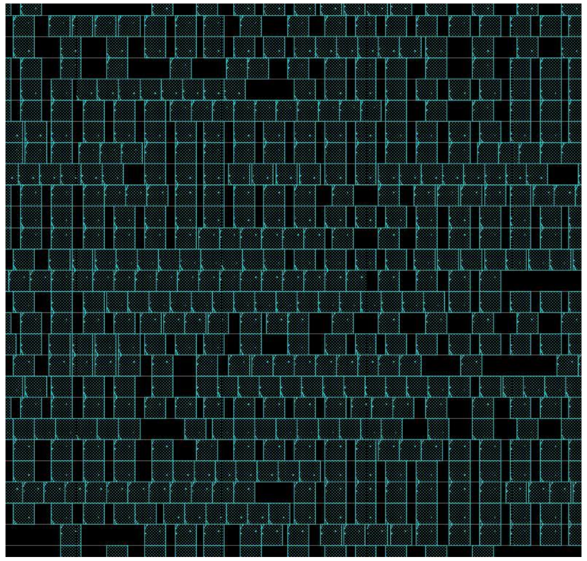

Pad Limited

Utilization

- **Utilization refers to the percentage of core area that is taken up by standard cells.**
  - A typical starting utilization might be 70%
  - This can very a lot depending on the design
- **High utilization can make it difficult to close a design**
  - Routing congestion,
  - Negative impact during optimization legalization stages.
- **Local congestion**
  - Can occur with pin dense cells like multiplexers, so utilization is not completely sufficient for determining die size.
  - Run a quick trial route to check for routing congestion
  - Refine the synthesis or increase routing resources



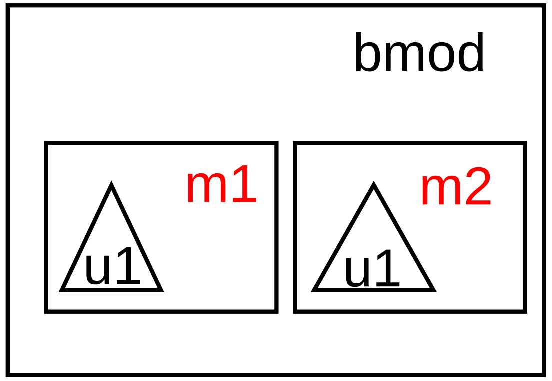

Low standard-cell utilization


High standard-cell utilization

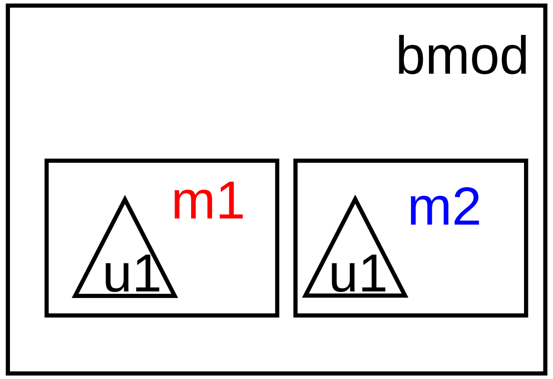
Uniquifying the Netlist

- **When moving to the physical domain, the netlist must be unique**
  - A unique netlist, means that each sub-module is only referenced once.
  - In the example, the non-unique netlist cannot optimize instance m1/u1 without changing instance m2/u1

```
module amod ();
 BUFFD1 u1 ();
endmodule

module bmod ();
 amod m1 ;
 amod m2 ;
endmodule
```



Non-unique

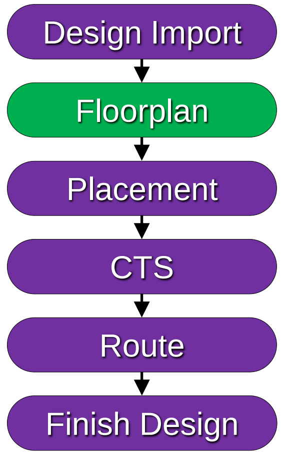


Unique

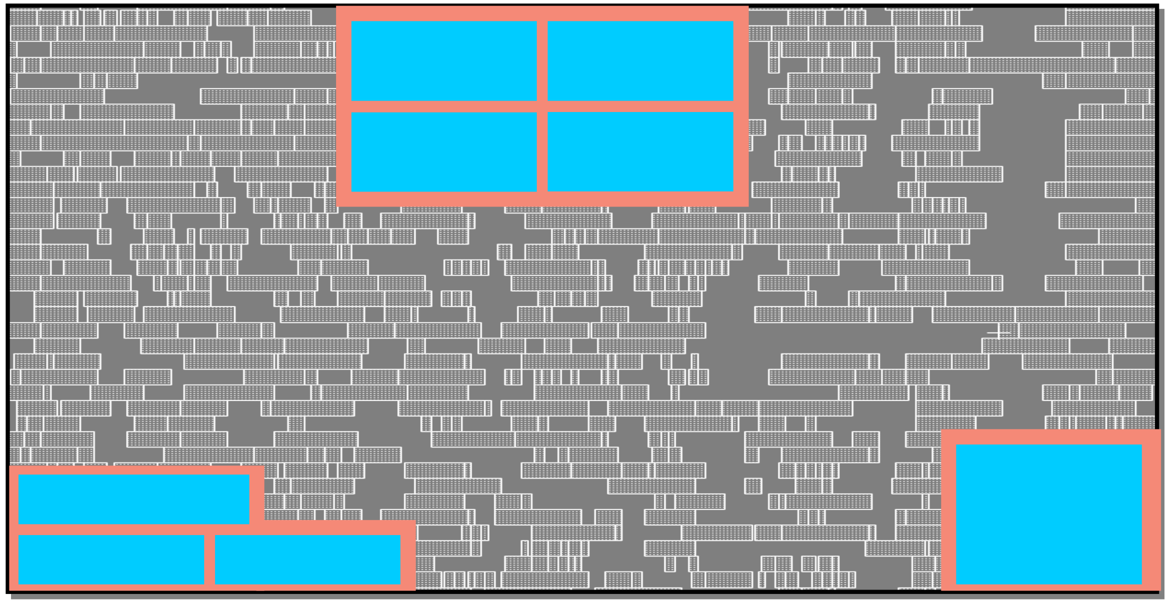
```
module amod1 ();
BUFFD1 u1 ();
endmodule

module amod2 ();
BUFFD1 u1 ();
endmodule

module bmod ();
amod1 m1 ;
amod2 m2 ;
endmodule
```

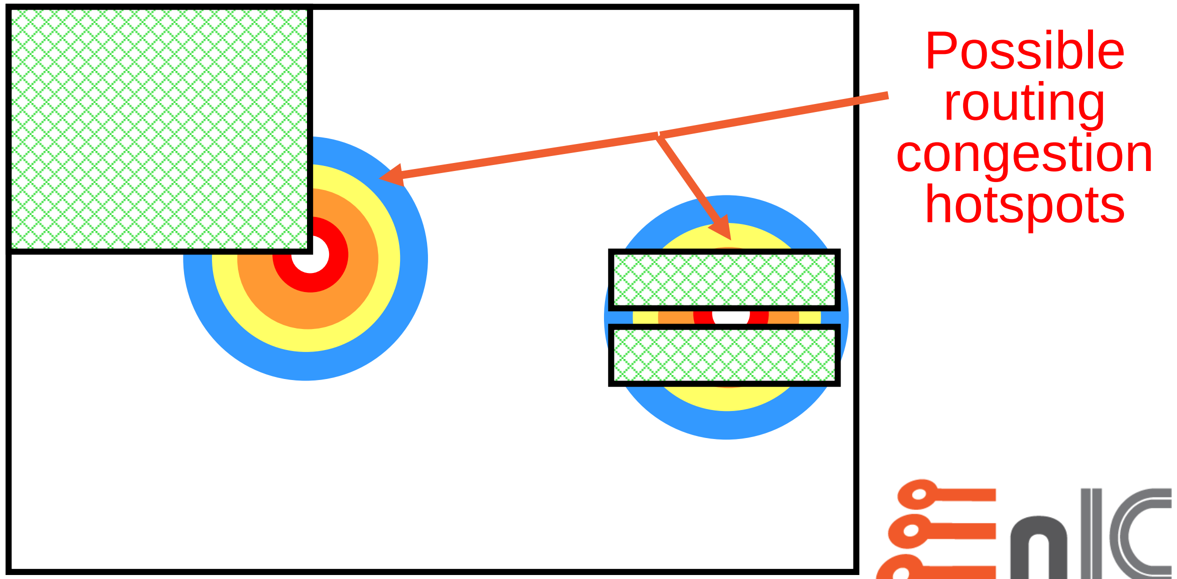
  - A synthesized netlist must be uniquified before placement can begin. This can be done either by the synthesizer or during design import.

Hard Macro Placement



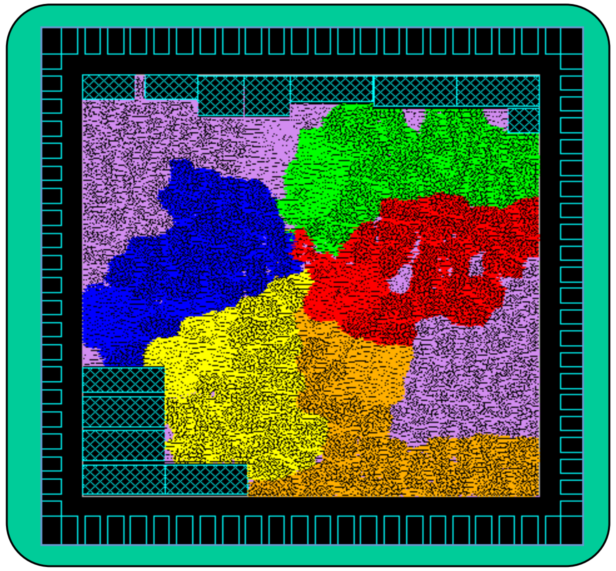
- **When placing large macros we must consider impacts on routing, timing and power. Usually push them to the sides of the floorplan.**
  - Placement algorithms generally perform better with a single large rectangular placement area.
  - For wire-bond place power hungry macros away from the chip center.
- **After placing hard macros, mark them as FIXED.**

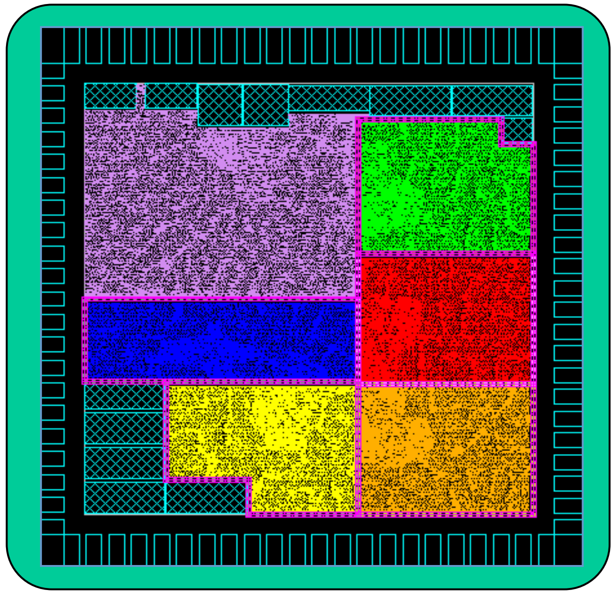


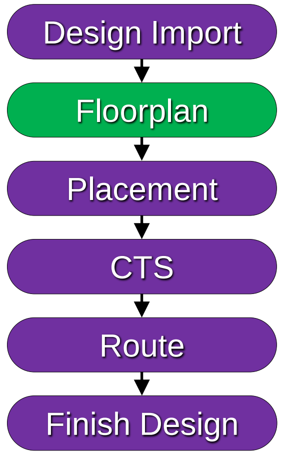


Placement Regions

- **Sometimes, we want to "help" the tool put certain logic in certain regions or cluster them together.**
- **Place and Route tools define several types of placement regions:**
  - Soft guide – *try* to cluster these cells together without a defined area.
  - Guide – *try* to place the cells in the defined area.
  - Region – must place the cells in the defined area, but other cells may also be placed there.
  - Fence – must place the cells in the defined area and keep out all other cells.

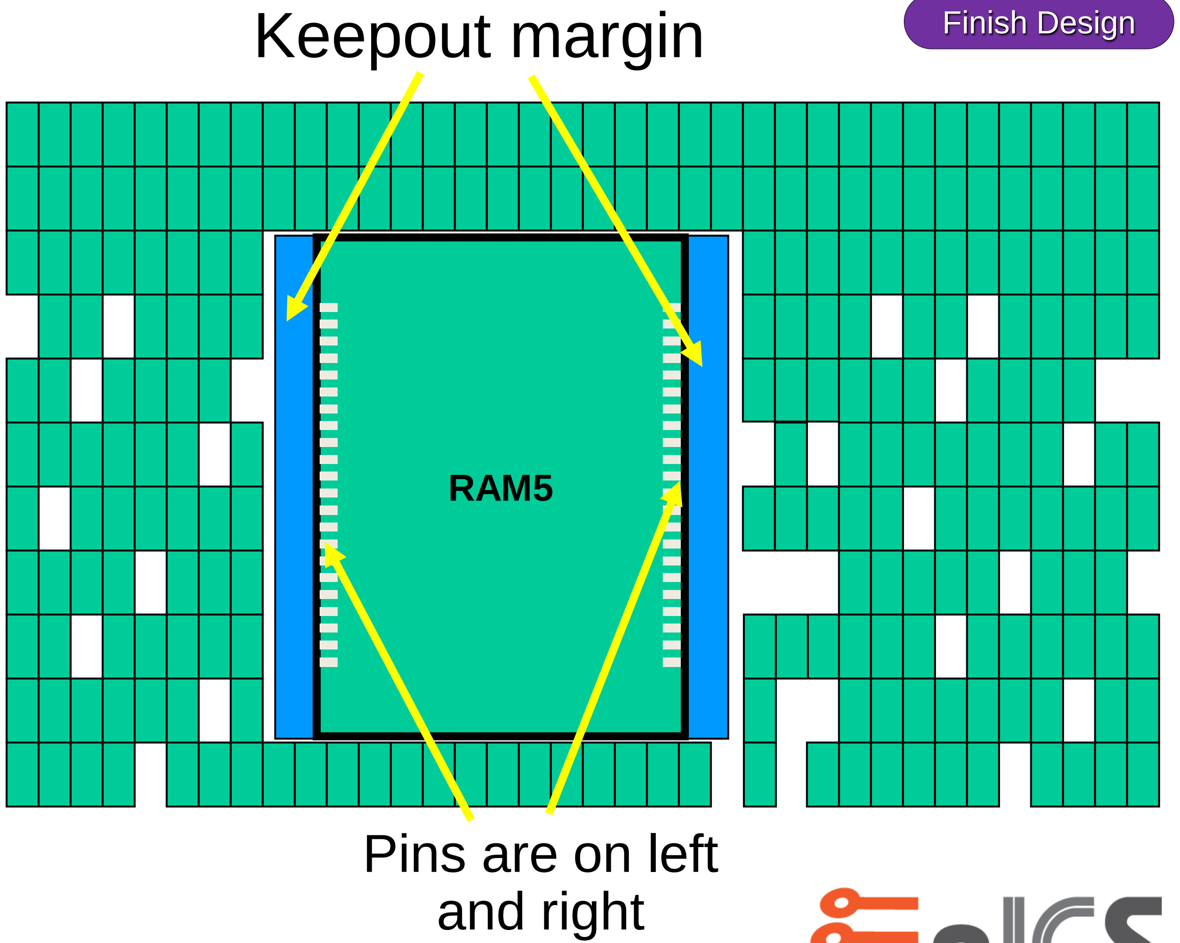




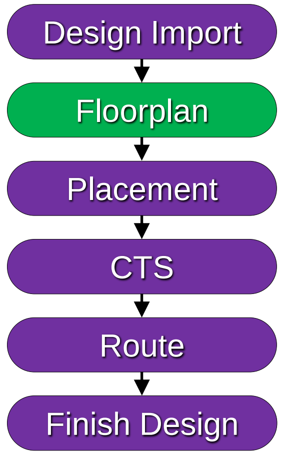


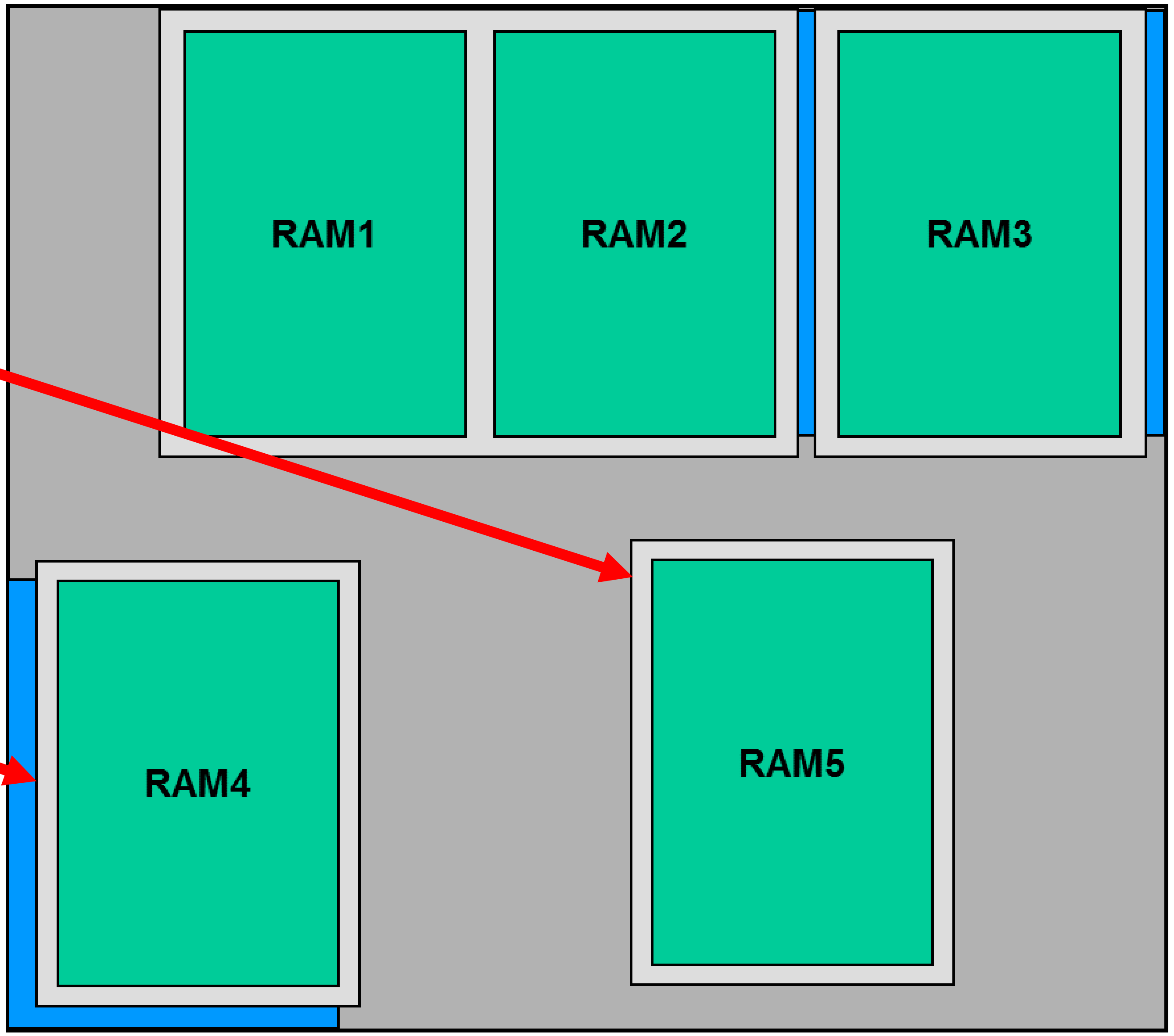
Placement Blockages and Halos

- **Placement blockages halos are areas that the tools *should not* place any cells.**
- **These, too, have several types:**
  - Hard Blockage – no cells can be placed inside.
  - Soft Blockage – cannot be used during placement, but may be used during optimization.
  - Partial Blockage – an area with lower utilization.
  - Halo (padding) – an area outside a macro that should be kept clear of standard cells.



Placement Blockages and Halos



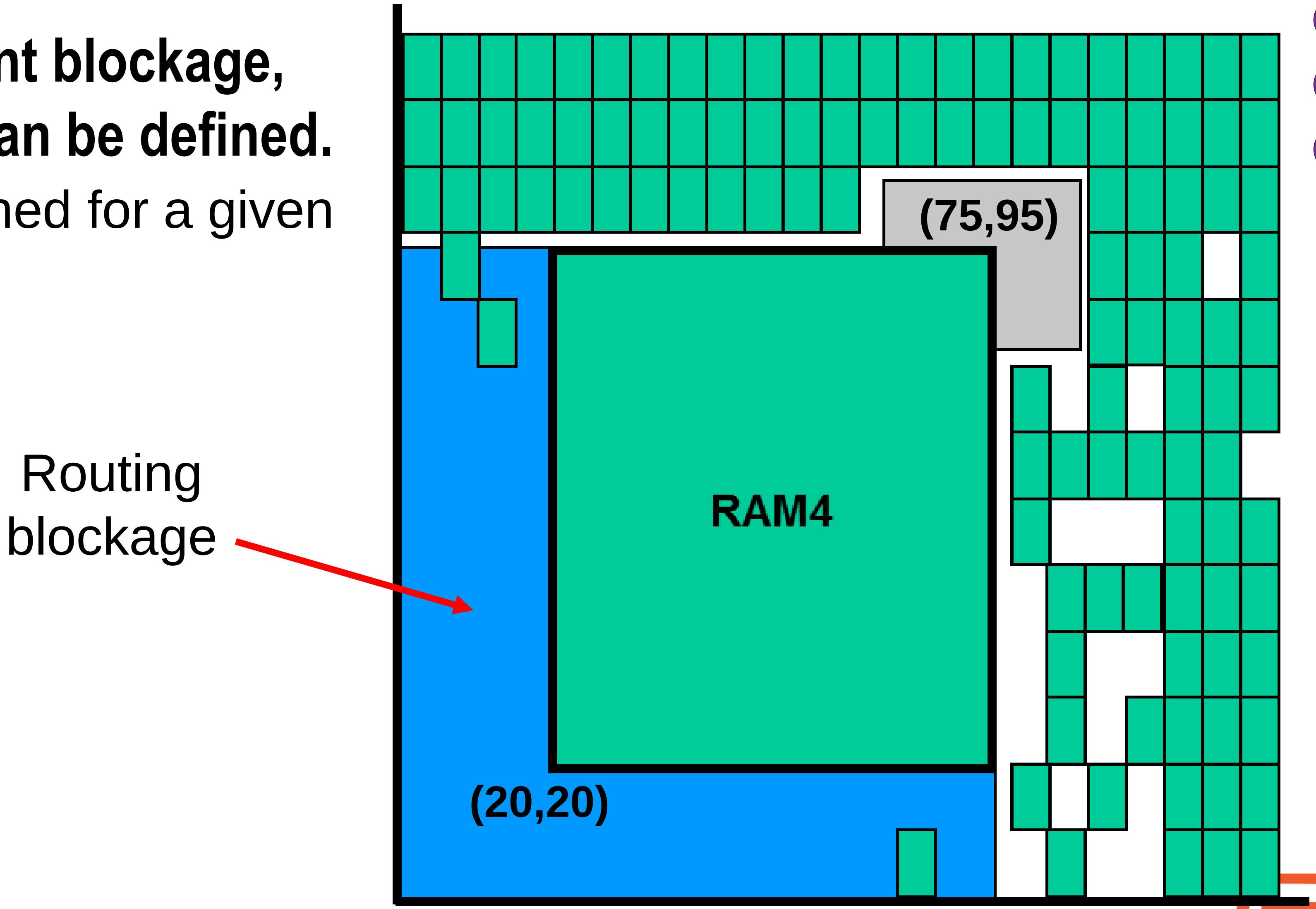


Hard blockage always created on all four sides

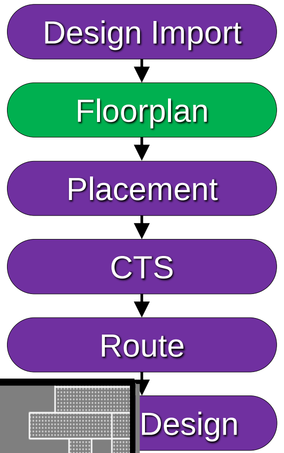
Soft blockage created only for the channels between the macros or between the macro and the core boundary

Routing Blockage

- **Similar to placement blockage, routing blockage can be defined.**
  - Blockage is defined for a given layer.



Guidelines for a good floorplan



Single large core area

RAMS out of the way in the corner

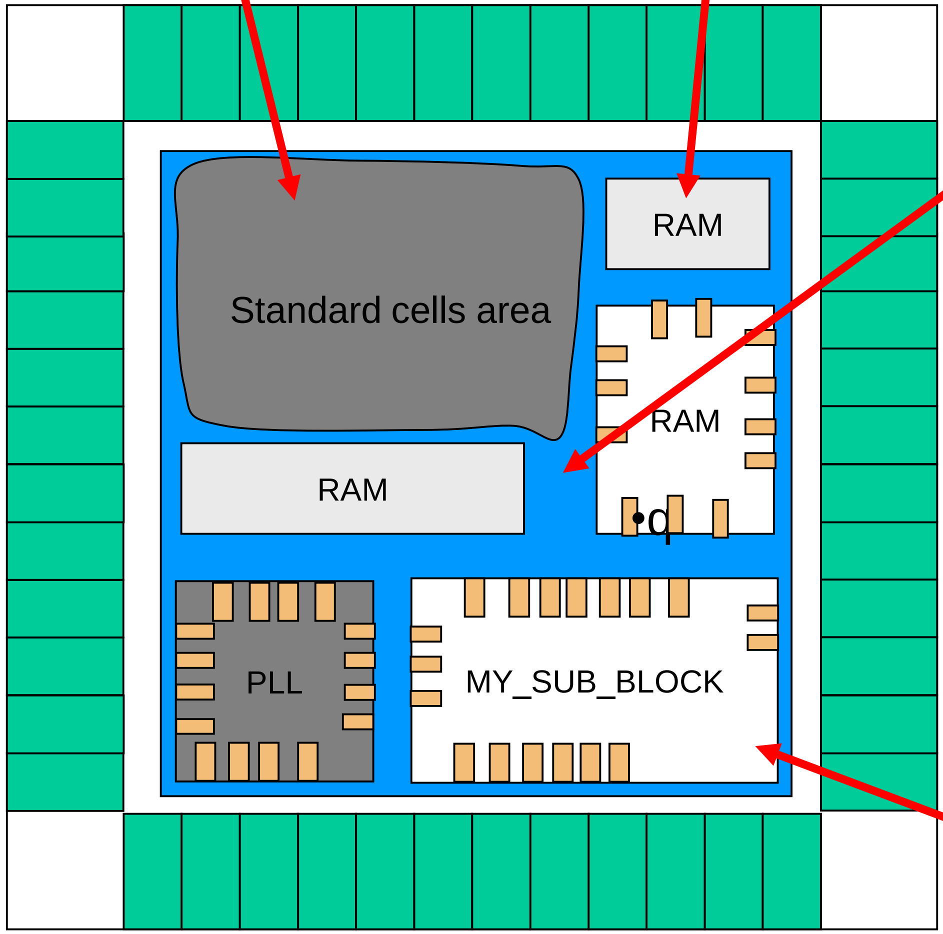
Large routing channels

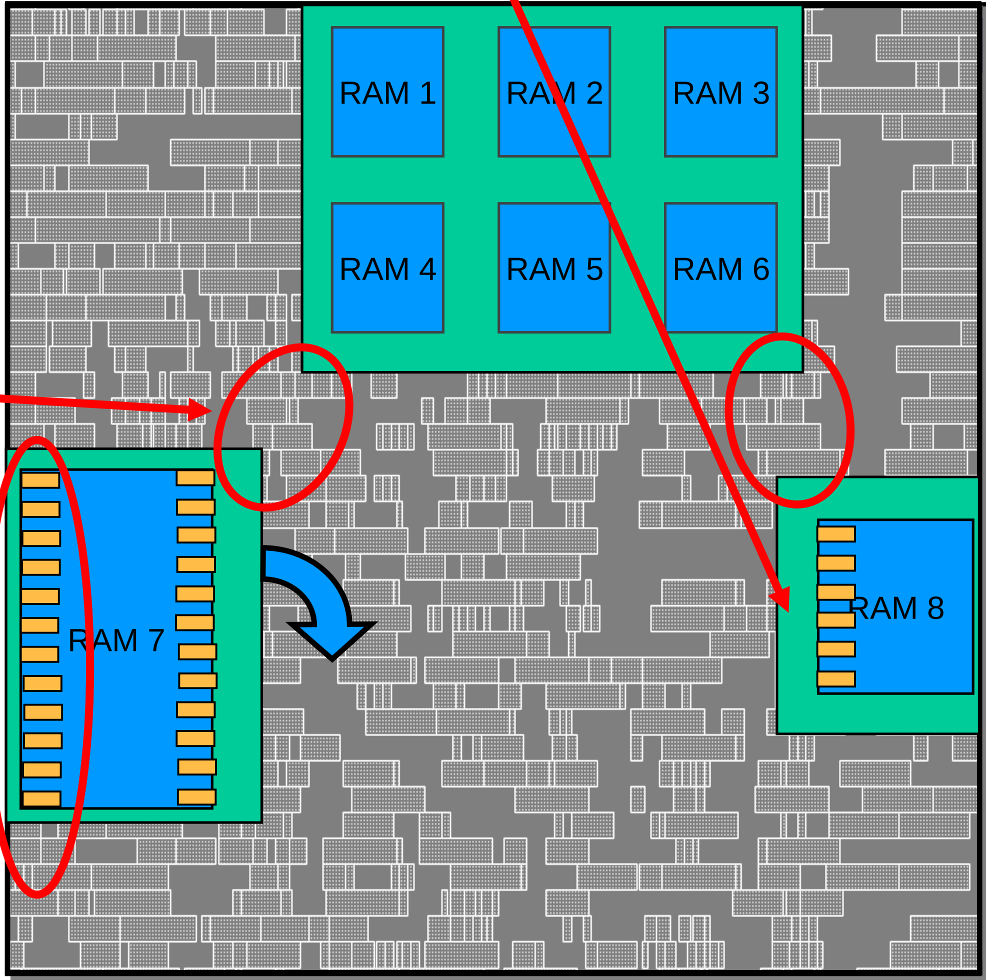
Avoid constrictive channels

Avoid many pins in the narrow channel. Rotate for pin accessibility

Pins away from corners

Use blockage to improve pin accessibility



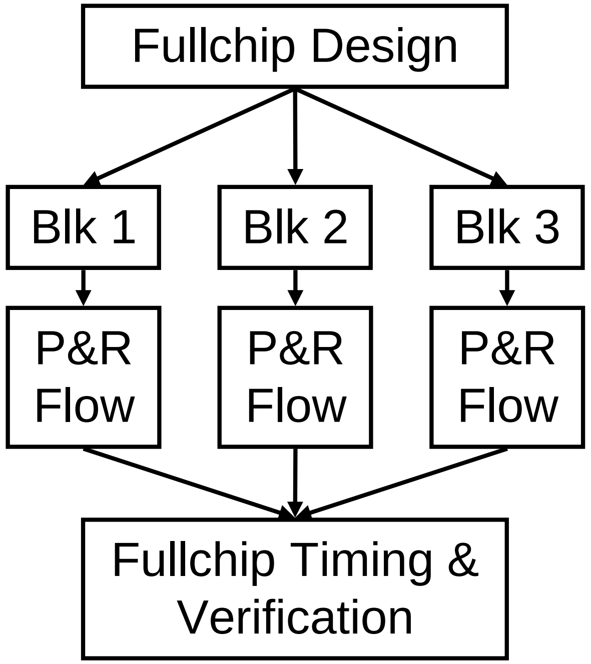


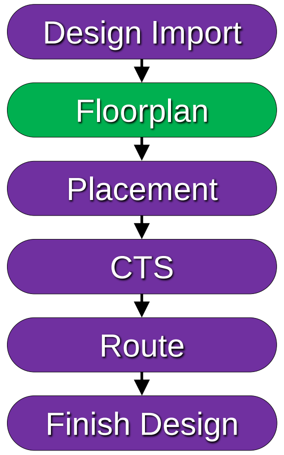
A bit about Hierarchical Design

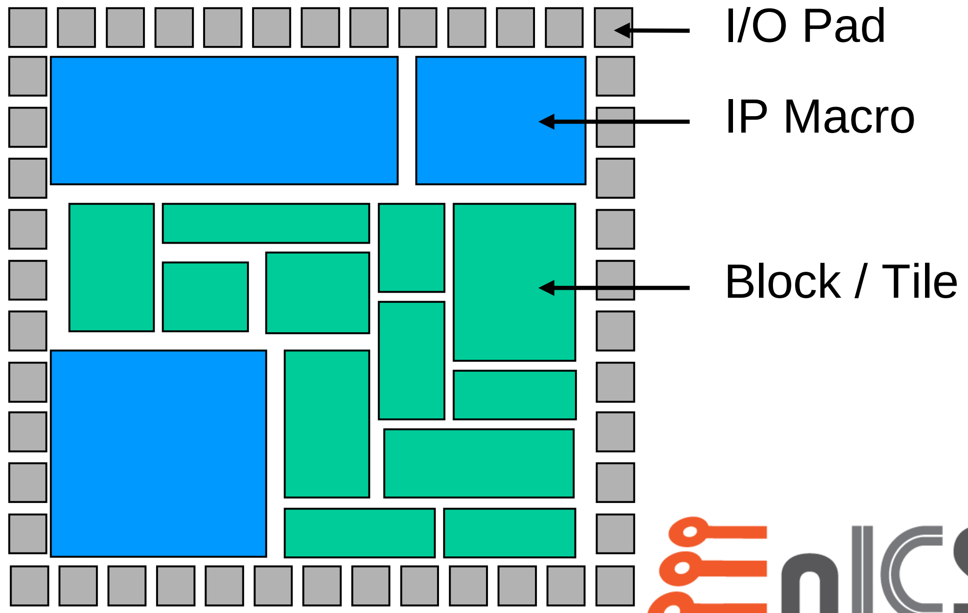
Or how do you deal with a really big chip

Flat vs. Hierarchical Design

- **If the design is too big, partition it into hierarchies**
  - Advantages
    - Faster runtime, less memory needed for EDA tools
    - Faster eco turn-around time
    - Ability to do design re-use
  - Disadvantages
    - Much more difficult for fullchip timing closure (ILMs)
    - More intensive design planning needed, feedthrough generation, repeater insertion, timing constraint budgeting.



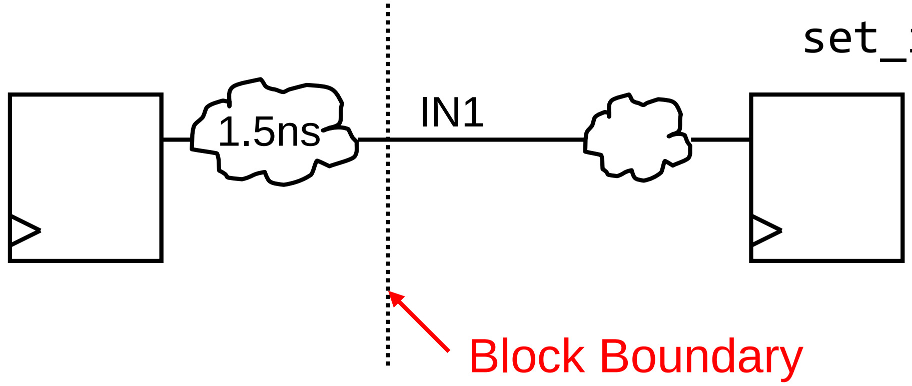




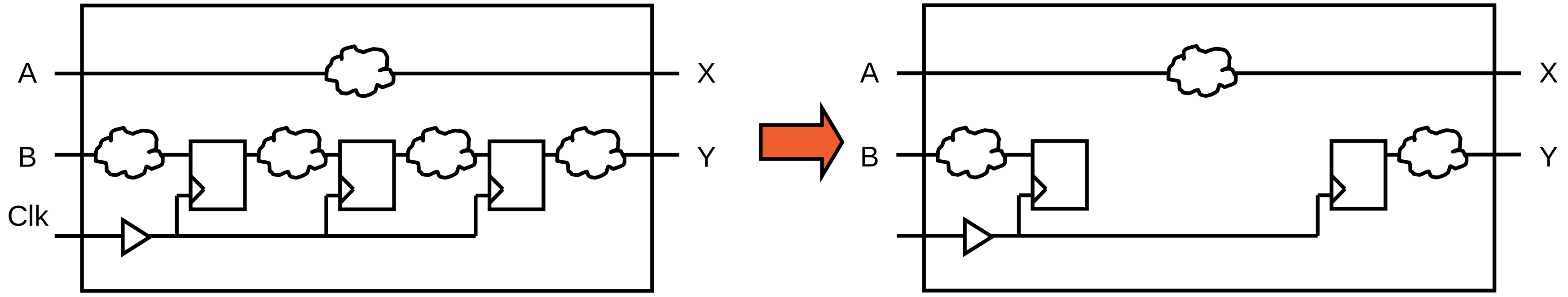
Hierarchical Design – Time Budgeting

- **Chip level constraints must be mapped correctly to block level constraints as I/O constraints**

```
set_input_delay 1.5 [get_port IN1]
```



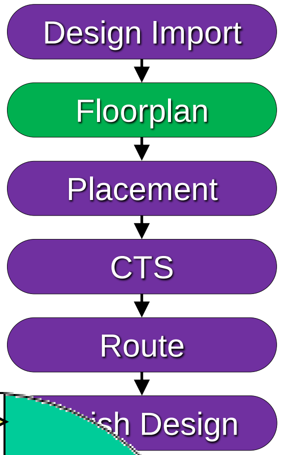
- **Interface Logic Models (ILMs) help simplify and speed-up design**



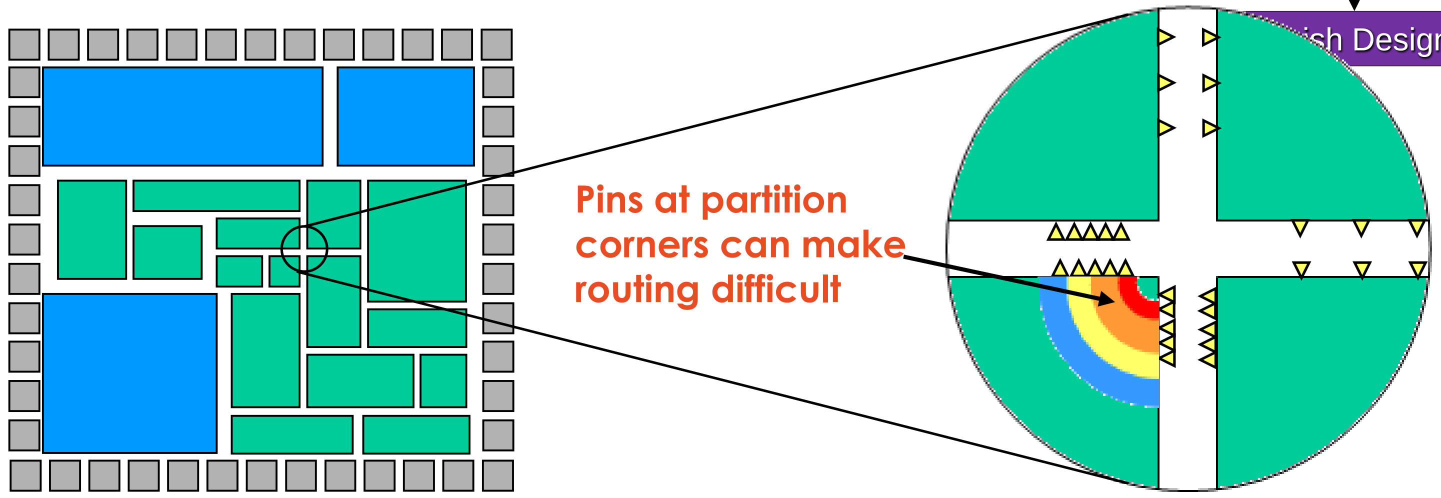
Original Netlist

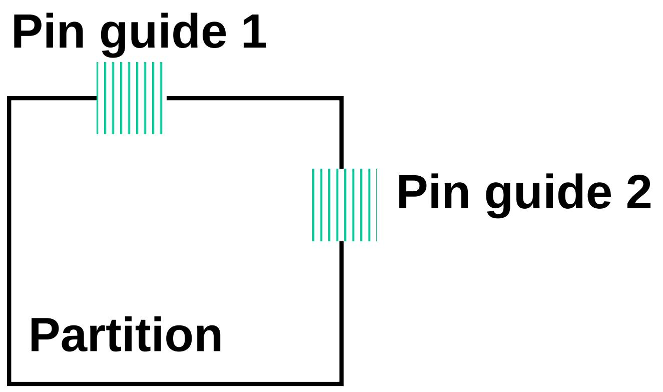
Interface Logic Model (ILM)

Hierarchical Design – Pin Assignment



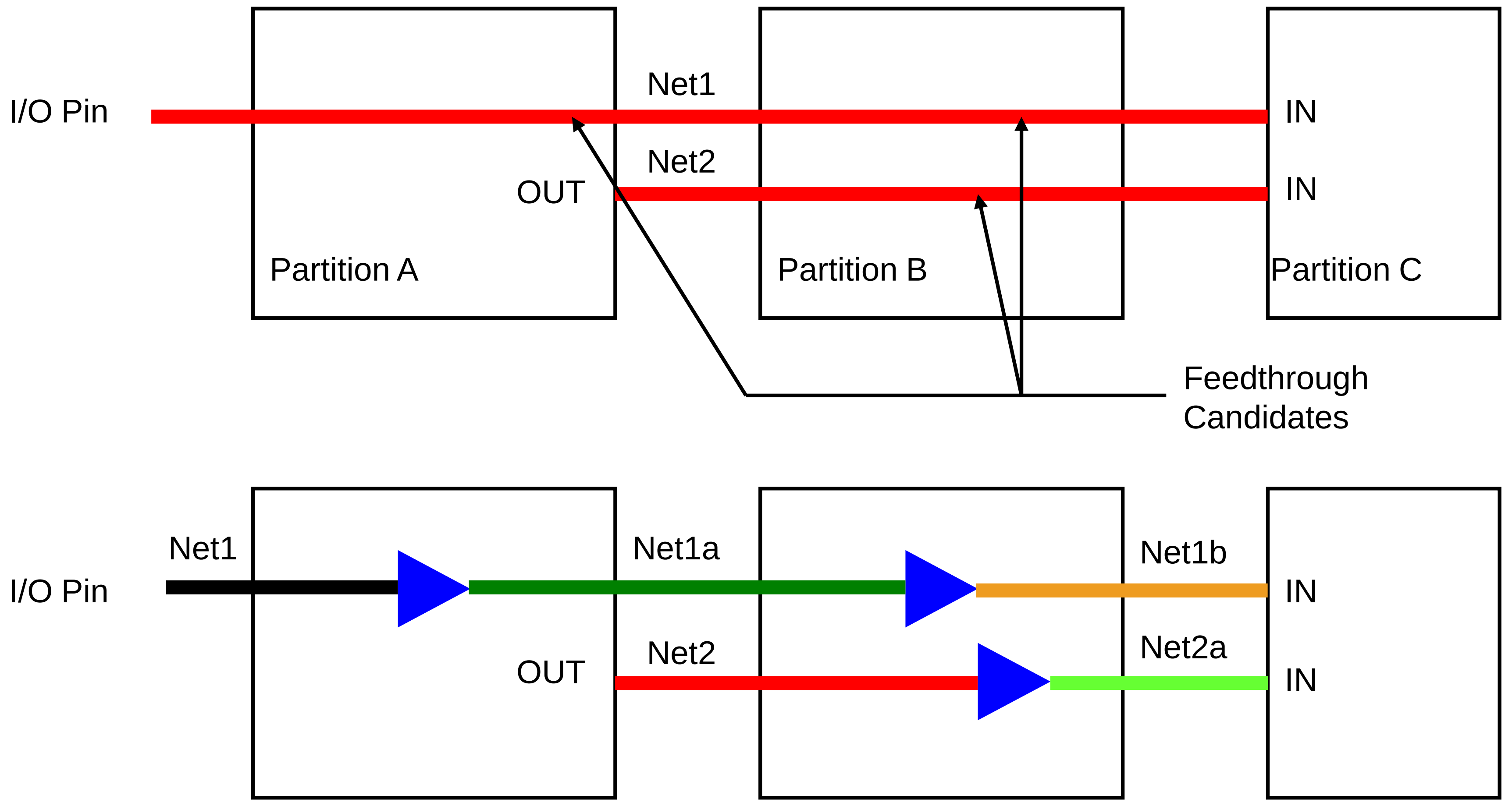
- **Pin constraints include parameters, such as:**
  - Layers, spacing, size, overlap
  - Net groups, pin guides
- **Pins can be assigned:**
  - Placement-based (flightlines)
  - Route-based (trial route, boundary crossings).
- **Pin guides**
  - Can be used to influence automatic pin placement of particular net groups





Hierarchical Design - Feedthrough

- For channel-less designs or designs with limited channel resources



Power Planning

Power Consumption and Reliability

Dynamic Power

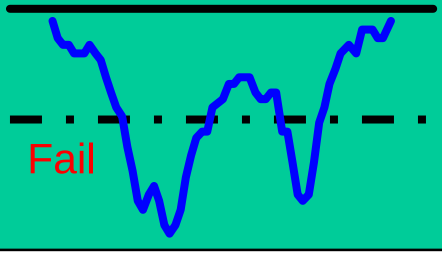
Static Power
(Leakage Power)

Floorplan
+
Design of the grid

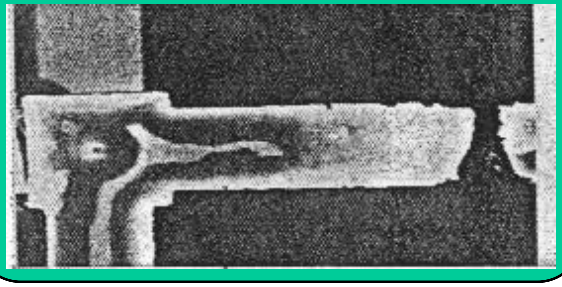
Average Power problem

Power density problem in the Long run

IR-Drop /
Voltage Drop

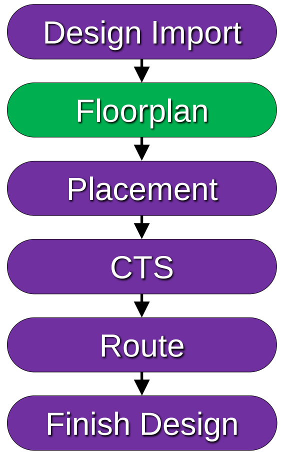


Electromigration
(EM)

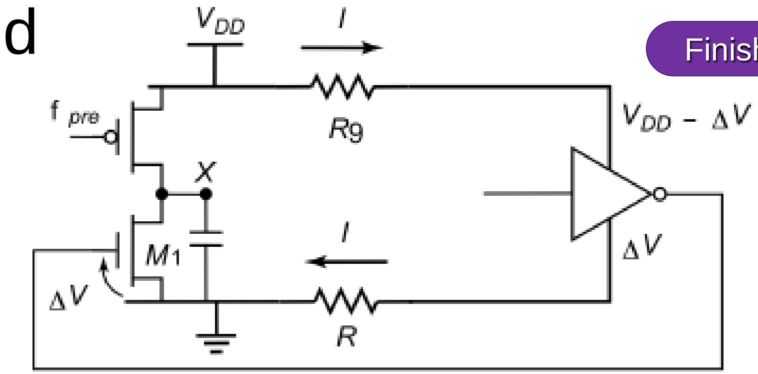


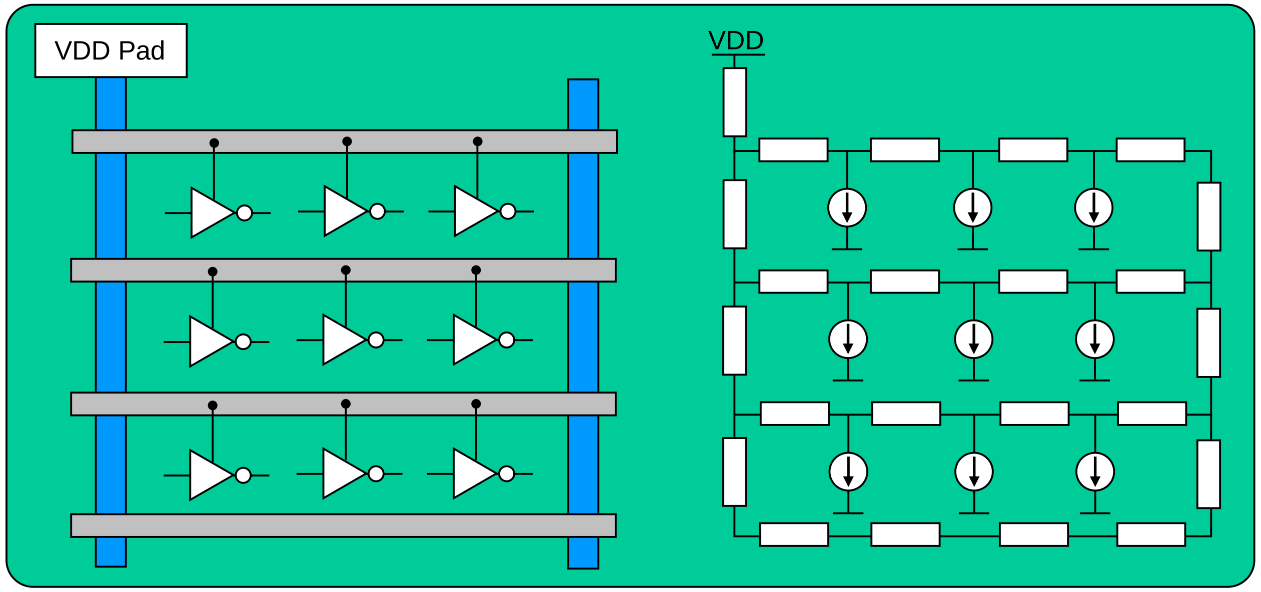
1 out of 5 chips fail due to excessive power consumption

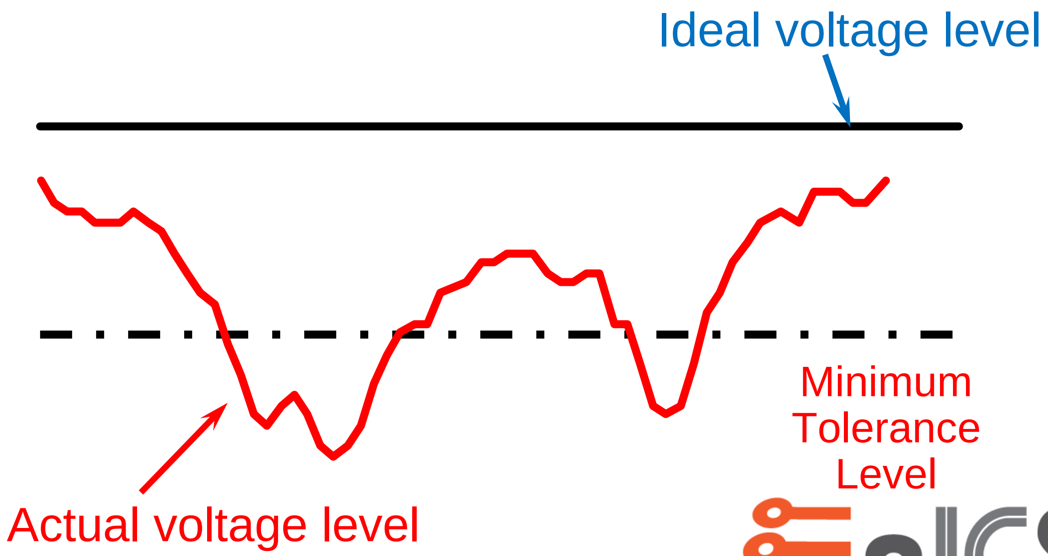
IR Drop



- **The drop in supply voltage over the length of the supply line**
  - A resistance matrix of the power grid is constructed
  - The average current of each gate is considered
  - The matrix is solved for the current at each node, to determine the IR-drop.

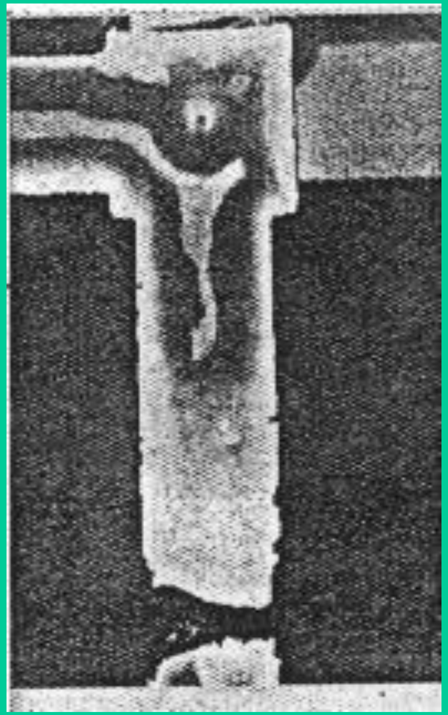






Electromigration (EM)

- **Electromigration refers to the gradual displacement of the metal atoms of a conductor as a result of the current flowing through that conductor.**
  - Transfer of electron momentum
- **Can result in catastrophic failure do to either**
  - Open : void on a single wire
  - Short : bridging between to wires
- **Even without open or short, EM can cause performance degradation**
  - Increase/decrease in wire RC



Power Distribution

- **Power Distribution Network functions**
  - Carry current from pads to transistors on chip
  - Maintain stable voltage with low noise
  - Provide average and peak power demands
  - Provide current return paths for signals
  - Avoid electromigration & self-heating wearout
  - Consume little chip area and wire
  - Easy to lay out

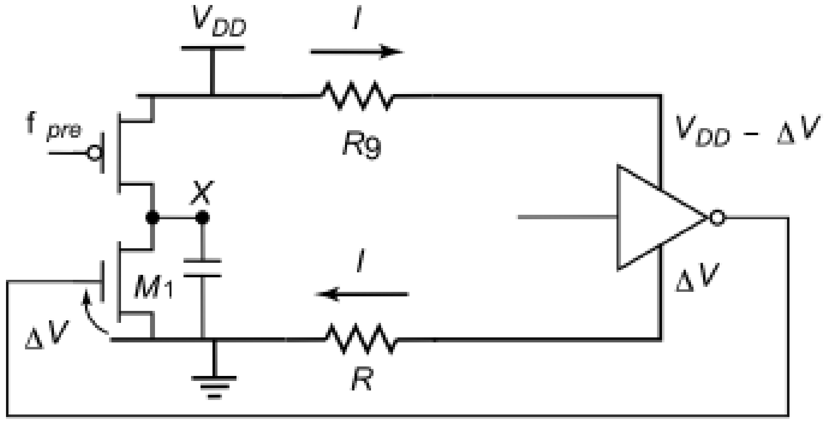
<u>More (Wider) Power Lines</u>:

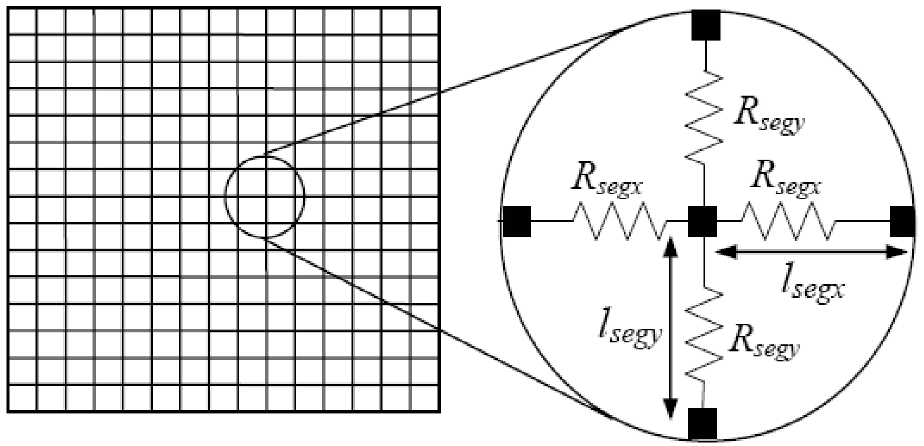
- Less Static (IR) drop
- Less Dynamic (dI/dt) drop
- Less Electromigration

BUT

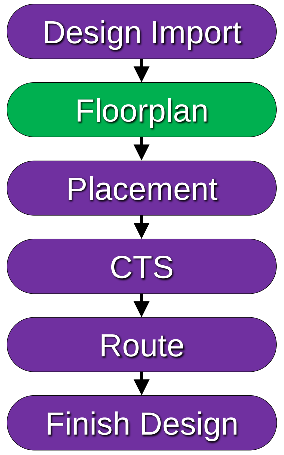
<u>More (Wider) Power Lines</u>:

- Fewer (signal) routing resources
  (i.e., higher congestion)





Power Distribution Challenge



- **Assume we have a 1mm long power rail in M1.**
  - Square resistance is given to be 0.1 ohm/square
  - If we make a 100nm wide rail, what is the resistance of the wire?

$$R = R_{\square} L/W = 0.1\Omega/_{\square} \cdot \frac{10^{-3}m}{100 \cdot 10^{-9}m} = 1000\Omega$$

  - Now, given a max current of 1mA/1um, due to Electromigration, what is the IR drop when conducting such a current through this wire?

$$I_{\max} = \frac{1\text{mA}}{1\mu\text{m}} \cdot 100\text{nm} = 0.1\text{mA}$$

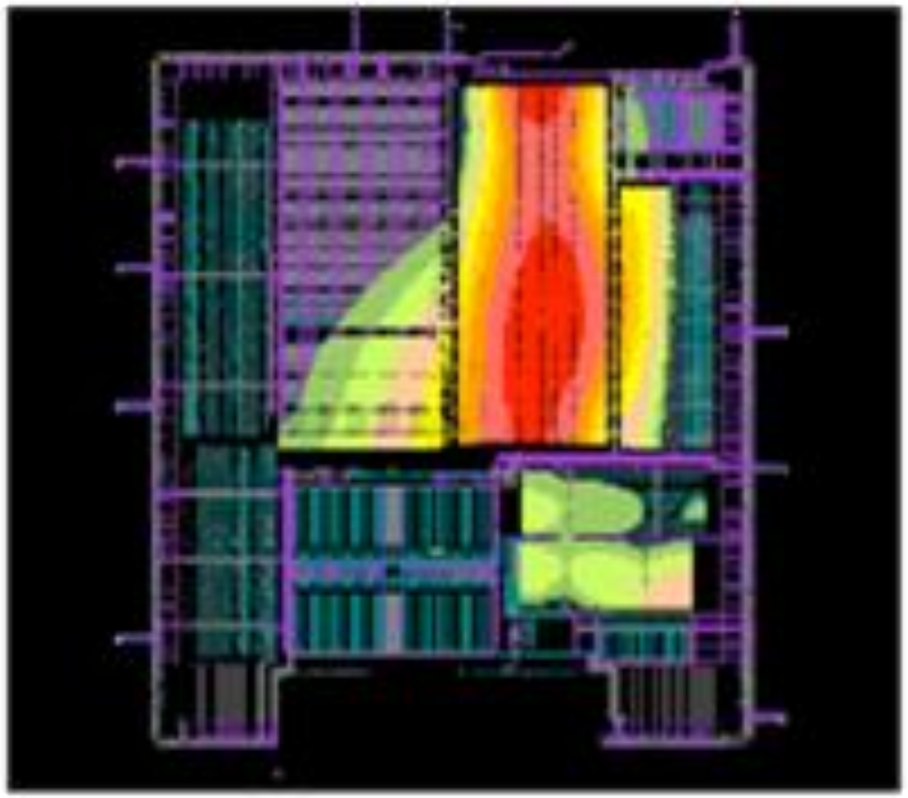
$$IR_{drop} = I_{\max} \cdot R_{wire} = 10^{-4} \cdot 10^{3} = 100\text{mV}$$

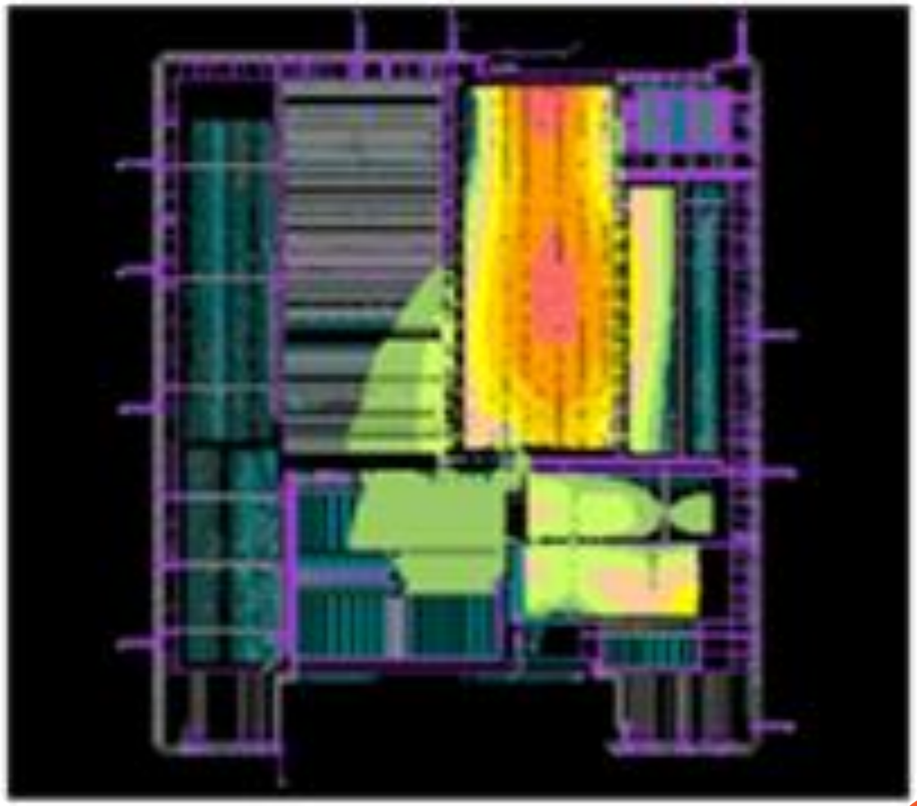
- **So what do we do?**
  - Make the power rails as wide and as thick as possible!

Hot Spots

- We generally map the IR drop of a chip using a color map to highlight “hot spots”, where the IR drop is bad.

Design Import
Floorplan
Placement
CTS
Route
Finish Design

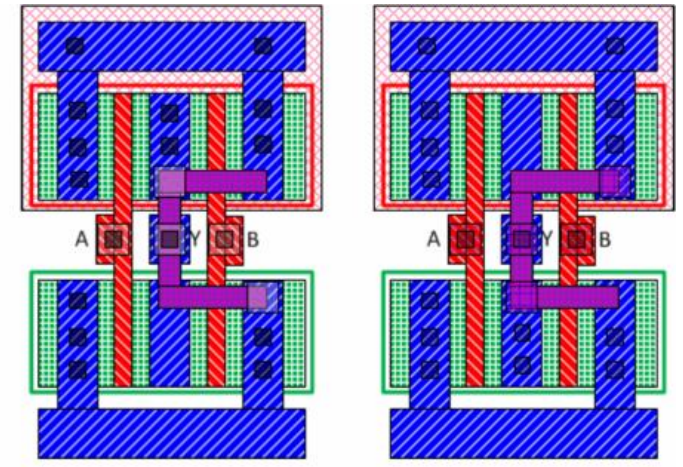

Initial IR Drop Mapping

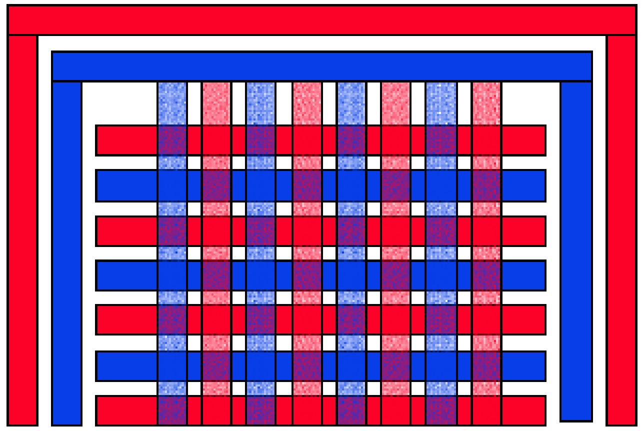

After adding a single wire!

Source: Cadence

Power and Ground Routing

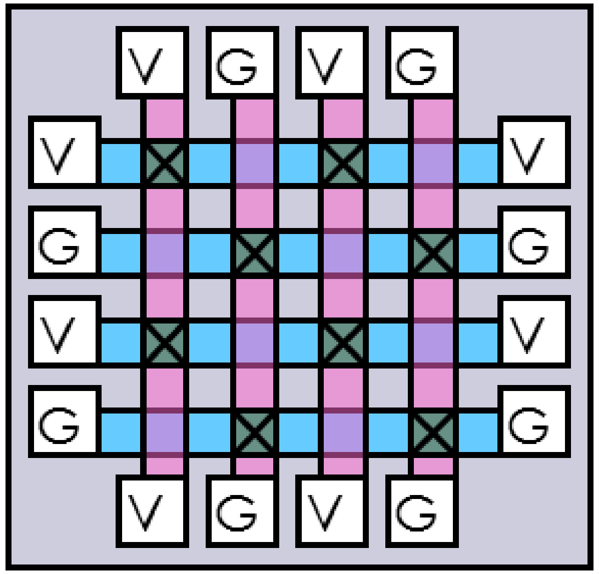
- **Each standard cell or macro has power and ground signals, i.e., VDD (power) and GND (ground)**
  - They need to be connected as well
- **Power/Ground mesh will allow multiple paths from P/G sources to destinations**
  - Less series resistance
  - Hierarchical power and ground meshes from upper metal layers to lower metal layers
  - Multiple vias between layers
- **You can imagine that they are HUGE NETWORKS!**
  - In general, P/G routings are pretty regular
  - P/G routing resources are usually reserved

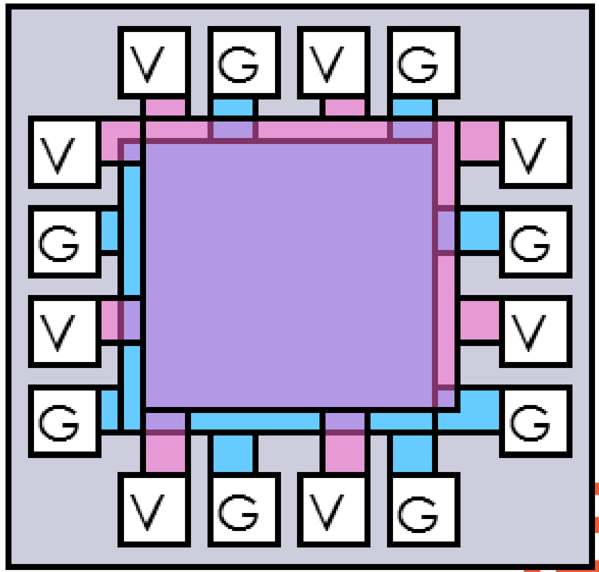


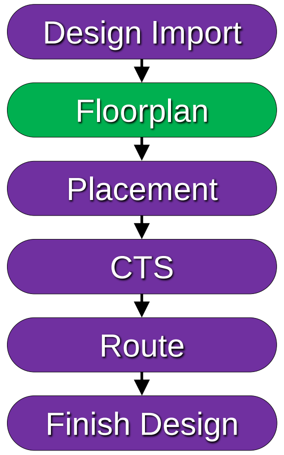


Standard Approaches to Power Routing

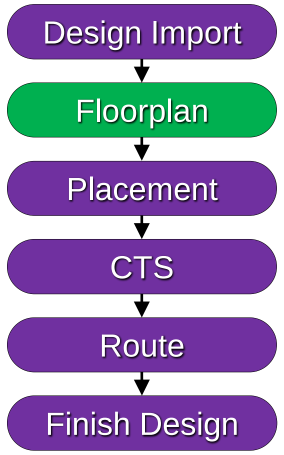
- **Power Grid**
  - Interconnected vertical and horizontal power bars.
    - Common on most high-performance designs.
    - Often well over half of total metal on upper thicker layers used for VDD/GND.
- **Dedicated VDD/GND planes.**
  - Very expensive.
    - Only used on Alpha 21264.
    - Simplified circuit analysis.
    - Dropped on subsequent Alphas.
- **Some thoughts/trends:**
  - P/G I/O pad co-optimization with classic physical design
  - Decoupling capacitors to reduce P/G related voltage drop
  - Multiple voltage/frequency islands make the P/G problem and clock distributions more challenging



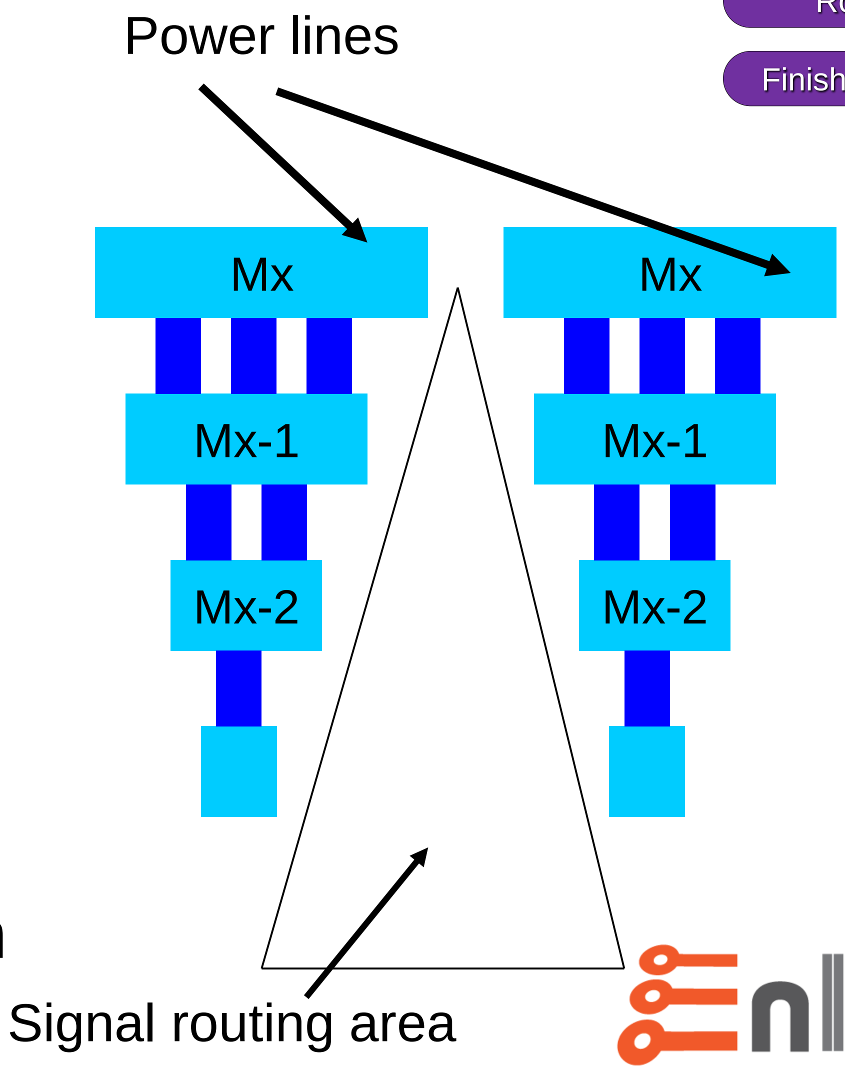




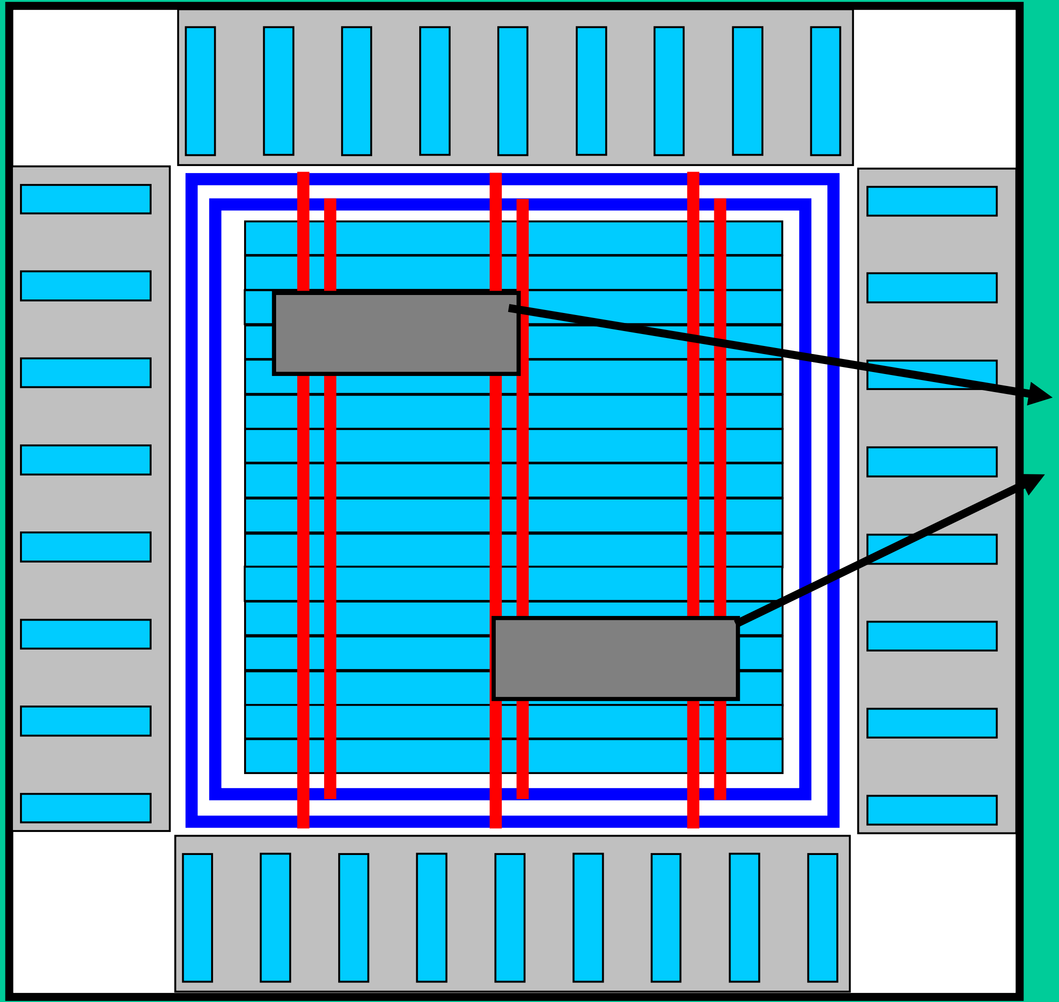
Power Grid Creation



- **Tradeoff IR drop and EM versus routing resources**
  - Require power budget
    - initial power estimation
    - average current, max current density
- **Need to determine**
  - General grid structure (gating or multi-voltage?)
  - Number and location of power pads (per voltage)
  - Metal layers to be used
  - Width and spacing of straps
  - Via stacks versus available routing tracks
  - Rings / no rings
  - Hierarchical block shielding
- **Run initial power network analysis to confirm design**



Power Grid Creation – Macro Placement

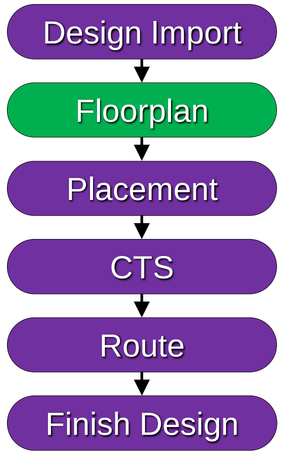


Blocks with the highest performance and highest power consumption

Close to border power pads (IR drop)

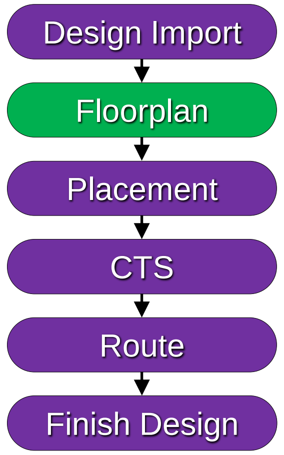
Away from each other (EM)

Summary – Floorplanning in Encounter



- **In the exercise, we will go over this, hand on, but in general, the floorplanning flow is:**
  - Initialize Design:
    - Define Verilog netlist, MMMC (timing, sdc, extraction, etc.), LEF, IO placement
  - Specify floorplan
    - Define floorplan size, aspect ratio, target utilization
  - Place hard macros
    - Absolute or relative placement
    - Define halos and blockages around macros
  - Define regions and blockages
    - If necessary, define placement regions and placement blockage
    - If necessary, define routing blockage

Summary – Floorplanning in Encounter



- Define Global nets
  - Tell the tool what the names of the global nets (VDD, GND) are and what their names are in the IPs.
- Create Power Rings
  - Often rings for VDD, GND are placed around the chip periphery, as well as around each individual hard IP.
- Build Power Grid
  - Connect standard cell ‘follow pins’
  - Build power stripes on metal layers
  - Make sure power connects to hard IPs robustly
- Assign Pins
  - If working on a block (not fullchip), assign pins to the periphery of the floorplan.

Main References

- IDESA
- Rabaey
- EPFL Tutorial
- Experience!