EE-429
Fundamentals of VLSI Design

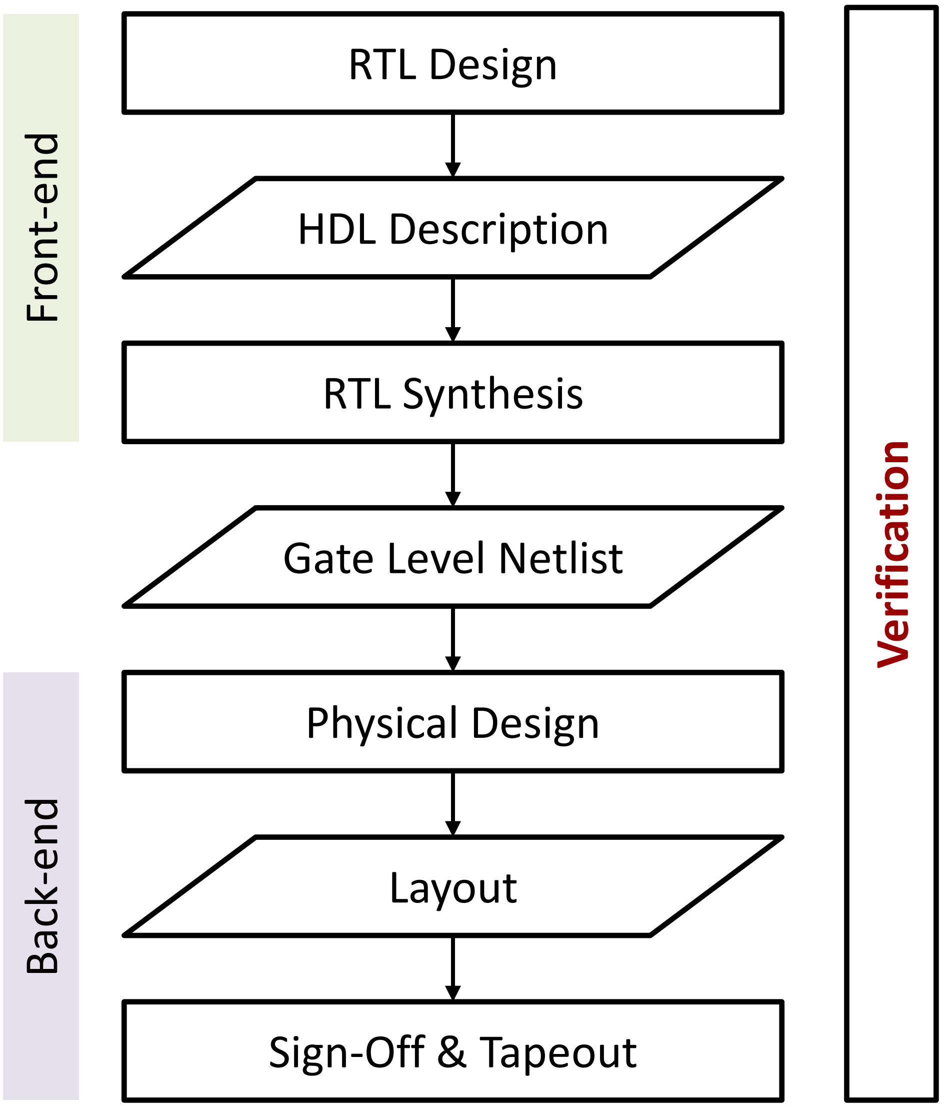
The Semi-Custom Backend

Andreas Burg

with material from the lectures of Prof. Adam Teman

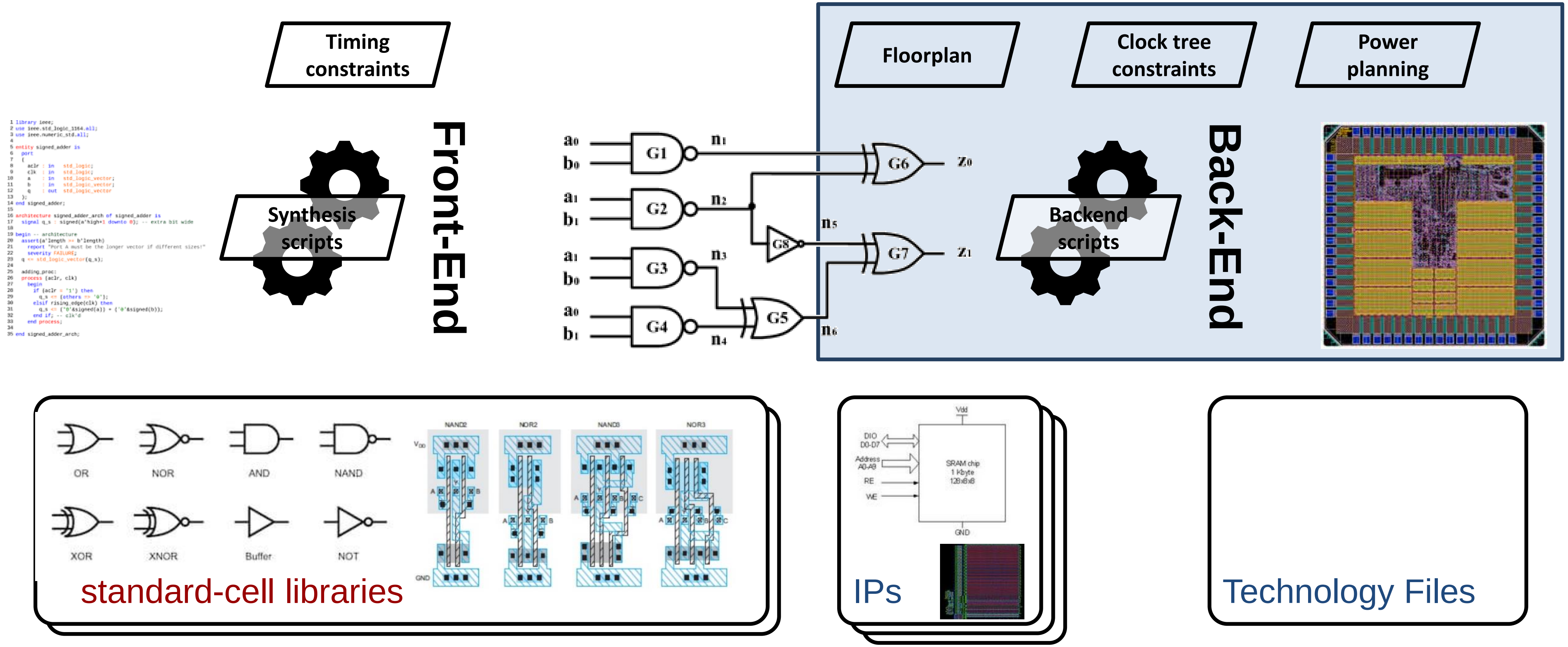
Semi-Custom (Digital) ASIC Design Flow

- **Semi-custom design flow:**
  - Starts from a Register Transfer Level description in a hardware description language (HDL)
  - **Front-end flow**: handles the transition from RTL to the gate level
  - Back-end flow: handles the transition from a netlist to physical design data
- **Each step is** always **accompanied** by **verification**
  - Check functionality, timing, and physical constraints



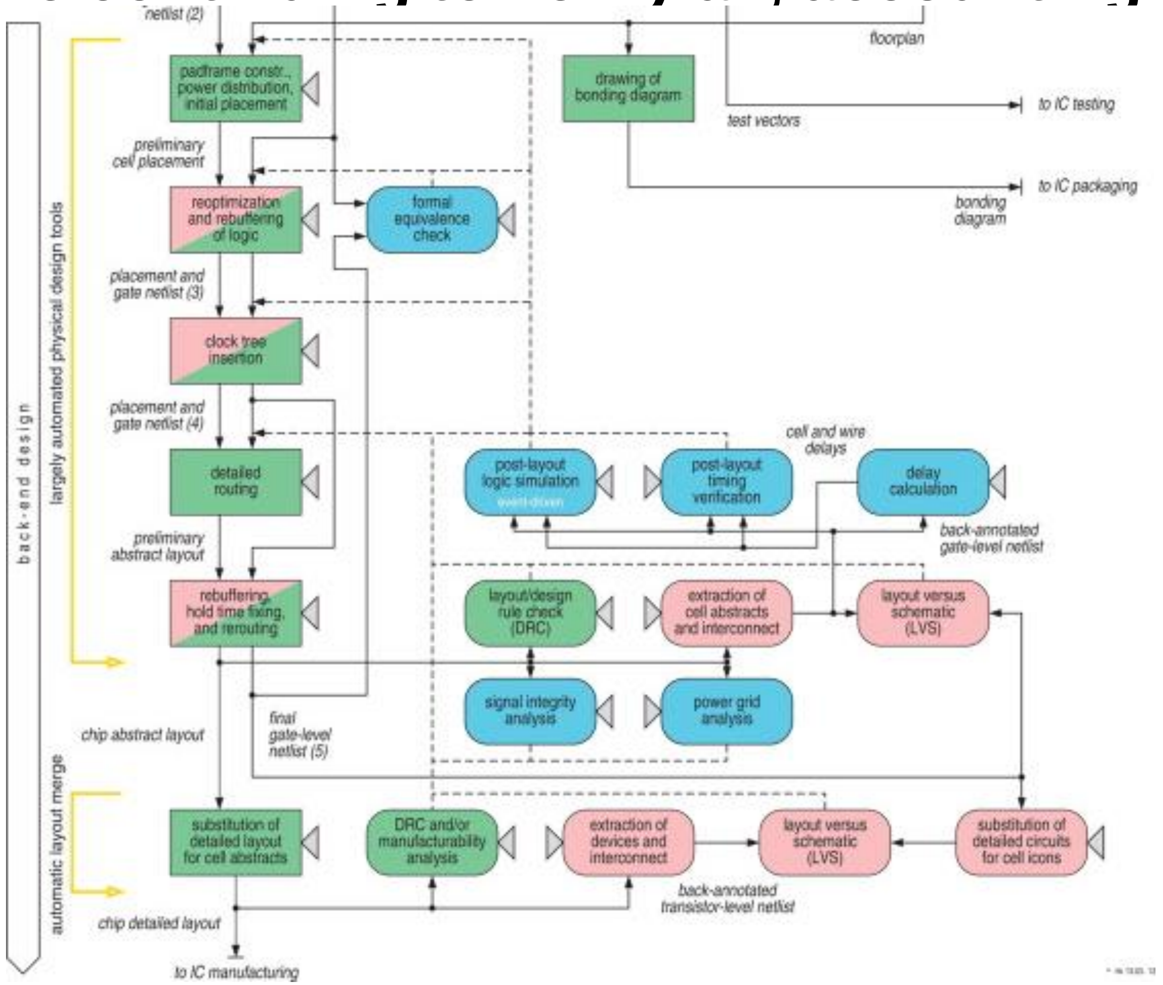
Chips are Built from IPs and Standard Cells

- **RTL design** **builds** **complex** **designs from a set of pre-defined gates (standard-cells)** **and** available **macros**



Back-End Design Flow Details

- **Move to the physical world, defining the layout** and everything we skipped in the frontend**, while continuing to verify all, accounting for parasitics**



The Physical Design Flow

- **For the backend we move from tools with a logical approach to tools with a physical approach**
  - The **floorplan** is the foundation of the physical layout
    - Defines the basic properties of the design and the location of large IPs
    - Serves as a guidance for the remaining steps
  - **Placement** defines the location of the millions of standard cells considering congestion (routability) and timing
  - **Clock tree insertion** takes care of distributing clocks and other very-high fanout signals
  - **Routing** wires up the signals between standard-cells and IPs accounting for design rules, timing, and other physical aspects
  - Finally, the design is exported for a last, more detailed verification step (timing, DRC, LVS, DFM, …)

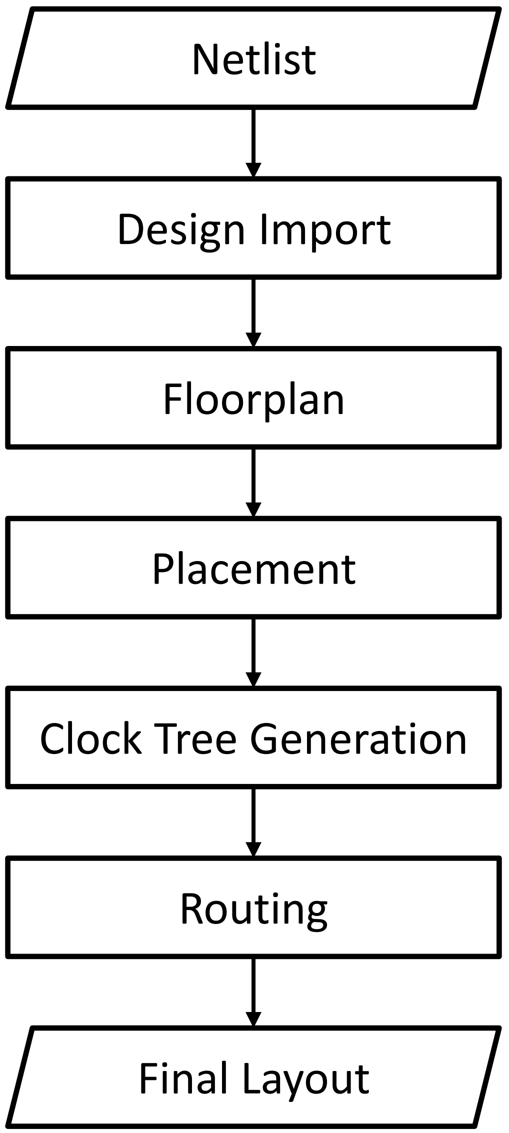
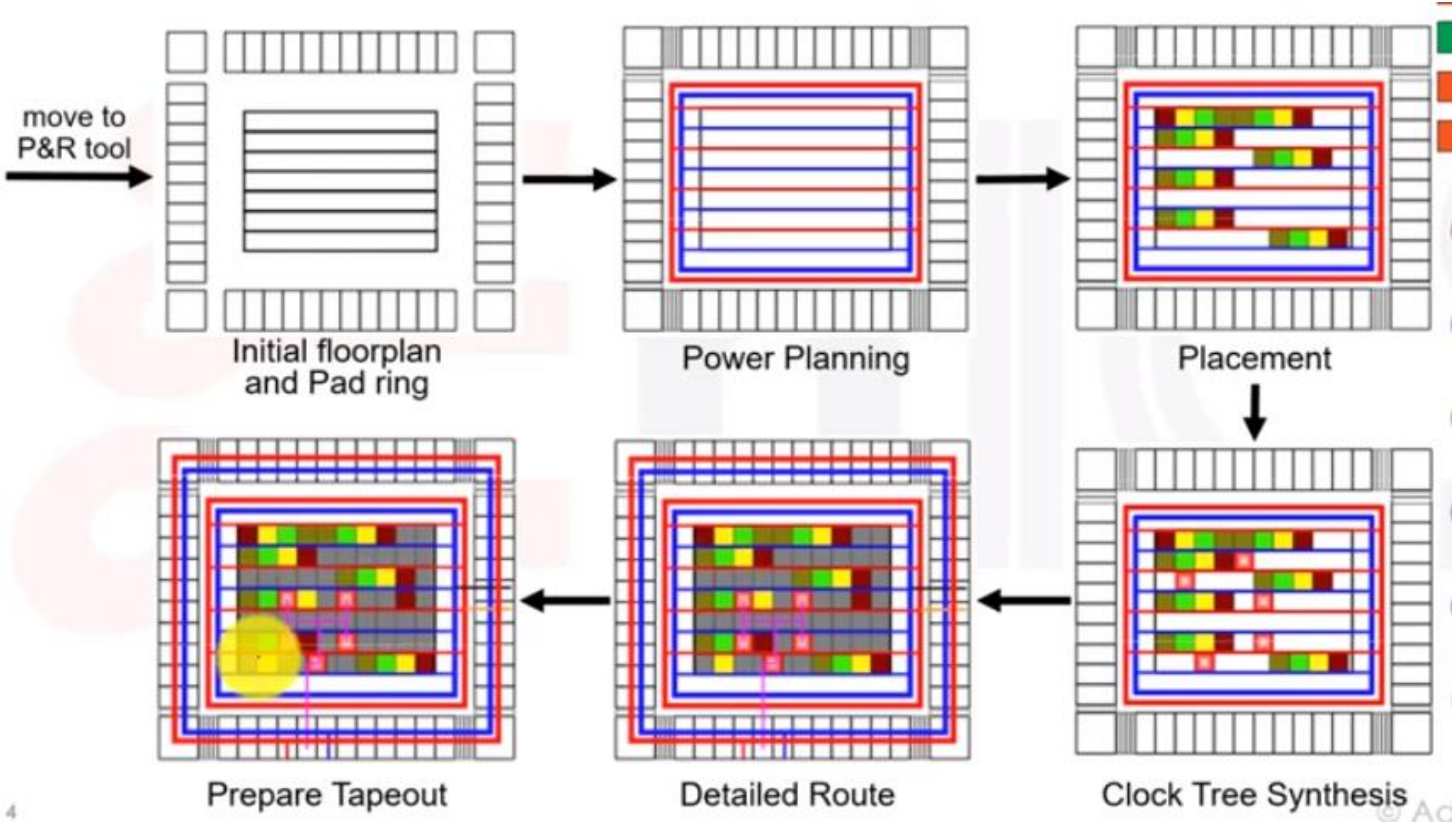
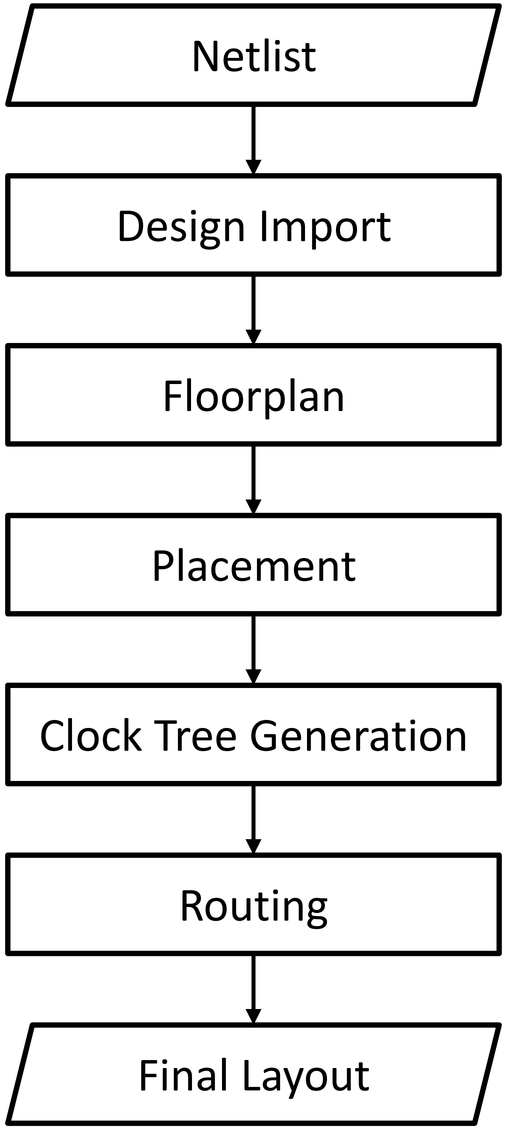


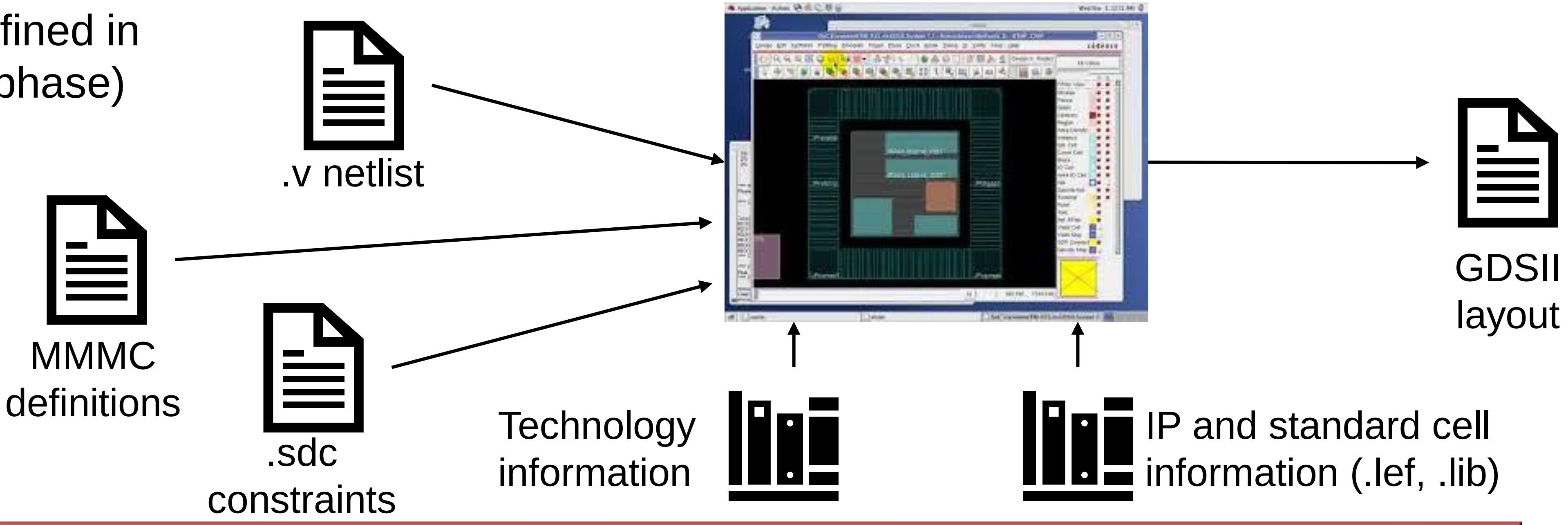
Illustration of the Physical Design





Frontend Design Information

- **For frontend design, we need the following information**
  - Netlist from the synthesis (frontend) (.v file)
  - Design timing constraints (.sdc file)
  - Design libraries (.lef and .lib files) and technology information (often also as lef files)
  - Operating corner definitions (Multi-mode-multi-corner (MMMC))
  - Floorplan (defined in the backend phase)

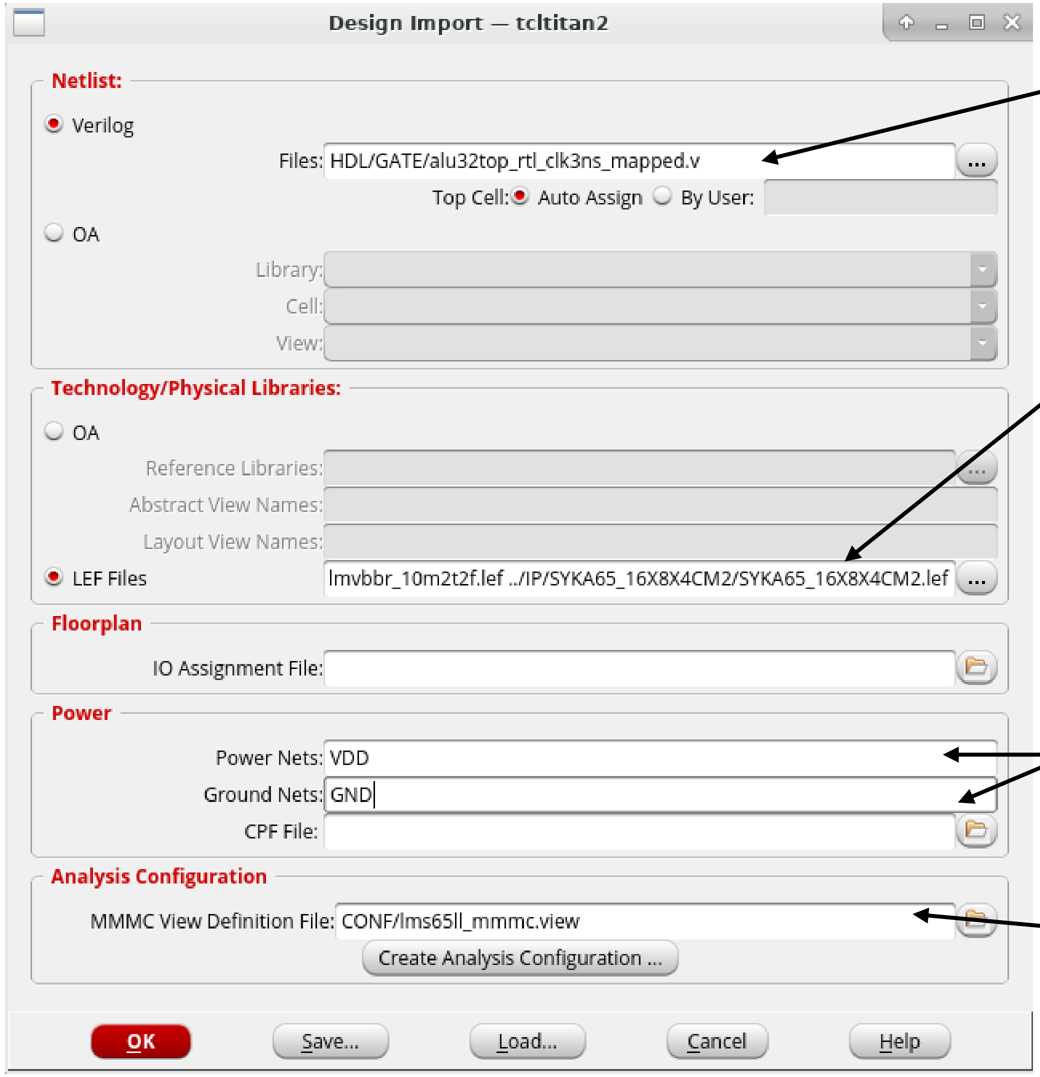


The Backend Environment

- **The backend design runs in a dedicated sub-directory and your project**
- **In this directory you find typically**
  - Setup/configuration files for the backend tool
  - Technology files from the PDK (e.g., information on routing and DRC rules, Vias, parasetics,...)
  - Pointers to source files for backend (netlist) and constraints inherited from synthesis (sdc)
  - Scripts for reading design files, for floor plan definition, and for place & route (all steps)
  - Folders for design databases
  - Folders with links to std cell library and IP macro abstracts (timing and physical)

```
CDS_IVUS
├── BIN                          --- folder for desig specific tcl scripts
├── CONF                         --- folder for tool and technology setup
├── DB                           --- folder for design data bases
├── DEX                          --- folder for gds export setup files
├── HDL -> ../HDL                --- pointer to your .v netlist
├── LOG                          --- folder for log files
├── RPT                          --- folder for reports
├── SDC                          --- folder for constraint files
├── TEC                          --- folder for technology/PEX  and IP/std-cell
├                                    abstracts (LEF)
└── TIM                          --- folder for IP/std-cell timing (lib)
```

Design Import and MMMC Setup



- Verilog netlist, exported from Synthesis

- **Technology LEF:** Technology information for P&R including simplified physical design rules and physical building blocks (e.g., VIAs)
- **Library LEF:** Abstract physical view of standard cells and IP macros

- Definition of **global power and ground nets**: Power and ground nets are not part of the Verilog netlist (contains only logical nets) and must be defined separately

- **MMMC Views**: define timing requirements and analysis conditions (corners) in the form of TCL commands

Multi-Mode Multi-Corner (MMMC)

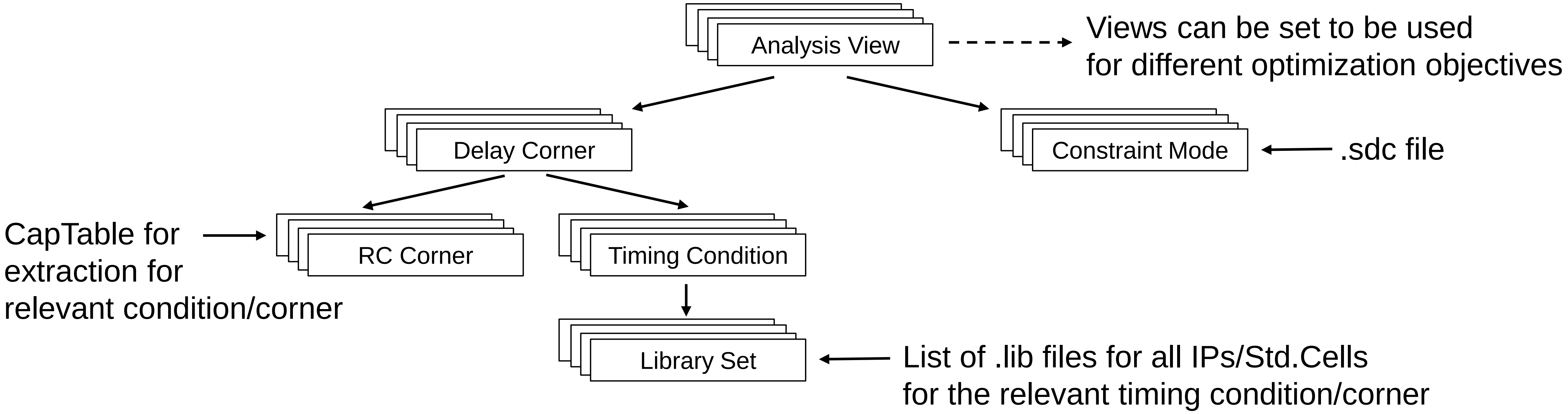
- **Timing analysis (and optimization) is complex and must consider:**

Operating conditions/corners	Operation Modes	Types of analysis/optimization
• Temperature: low/high • Process corners: FF/SS/... • Voltages: 0.6V (standby) • Voltage uncertainty +/- 10%	• High clock frequency • Low clock frequency • Standby/Sleep • Different interface timings • ...	• Setup checks • Hold check • Leakage • Power
↓	↓	↓
Different corners for cells (.lib) and even parasitics (.capTbl)	Different constraints (.sdc)	Different optimization objectives

- **Analysis and optimization must target various (many) combinations of these scenarios and objectives**
  - MMMC defines "scenarios" to be able to refer to them during analysis and optimization and to consider jointly all relevant combinations

MMMC Definition

- **MMMC Views define combinations of: “Constraint Modes” and “Delay Corners” (comprised of “RC Corners” and “Timing Conditions”)**
  - Constraint Modes: list of relevant timing constraints (in the form of SDC files) that define timing requirements and definitions (clock, input delay, output delay, false paths, ...) to be checked
  - Delay corners: timing libraries and extraction rules to determine how to calculate delays



Using MMMC

- **For each optimization objective we can set one or multiple MMMC Views**
- **Example:**

```
set_analysis_view -setup {view-1, view-2} -hold {view-1, view-2}
```

  - Typical objectives are Setup and Hold
  - Also available: leakage and dynamic

- **Timing reports refer always to the corresponding analysis views**

Example from EDA-Lab

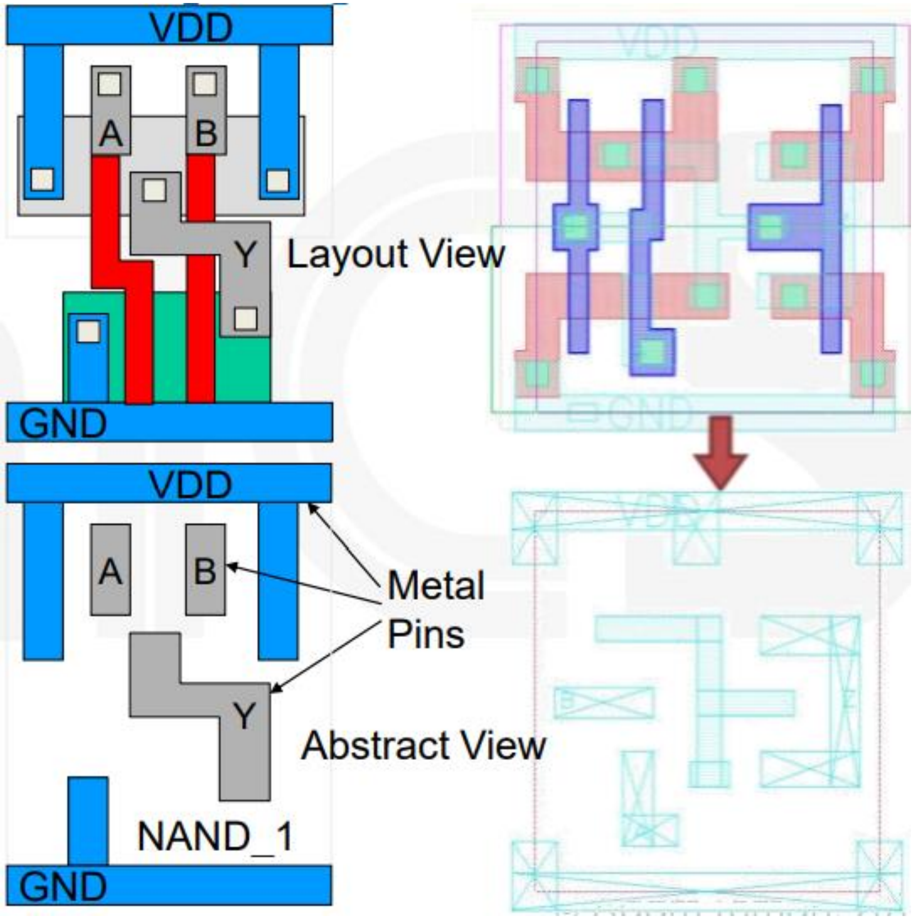
```
innovus> optDesign -preCTS
...
------------------------------------------------------------
     optDesign Final Summary
------------------------------------------------------------

Setup views included:
 analysis_setup

+--------------------+---------+---------+---------+
|     Setup mode     |   all   | reg2reg | default |
+--------------------+---------+---------+---------+
|           WNS (ns):|  0.009  |  0.009  |  0.127  |
|           TNS (ns):|  0.000  |  0.000  |  0.000  |
|    Violating Paths:|    0    |    0    |    0    |
|          All Paths:|   538   |   241   |   426   |
+--------------------+---------+---------+---------+
```

The Abstract Cell View for Backend

- **Abstracts are a simplified view of the layout to be used by automatic P&R tools**
  - Abstracts contain information about
    - The frame (size and shape) of the macro.
    - Place and connecting layers of the pins.
    - Place and connecting layers to power rails/rings.
    - Blockage (including DRC rules) to used interconnect layers.
  - But contain no information about
    - Poly layer (not used for top-level routing)
    - Transistors and diffusion layers



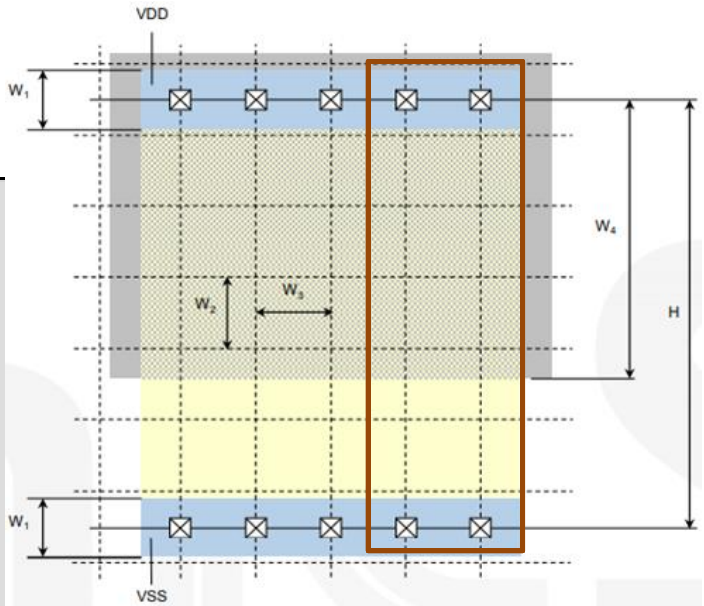
The Library Exchange Format (LEF)

- **LEF files contain basic (abstracted) layout information for the P&R tool**
  - Human readable ASCII format

- **There are two types of LEF files:**
  - The **technology LEF** defines basic geometry rules of the technology, basic layout geometries, and basic layout and routing structures and guidelines
    - **The technology LEF file is always read first**
  - The **library/cell LEFs** define specific structures for one or multiple cells including cell-type, placement constraints, pins, blockages, and cell antenna information
    - Larger macros have their own LEF file
    - Libraries use a single LEF for all cells and there may be many libraries (e.g., IO, standard-cell, …)

Technology LEF

- **Technology LEF Files contain (simplified) information about the technology for use by the placer and router:**
  - **Layers**
    - Layer names, such as M1, M2, etc.
    - Layer type, such as routing, cut (via)
    - Electrical properties (R, C)
    - Design Rules
    - Preferred routing direction
  - **Sites** (x and y grid of the library cells)
    - CORE sites are minimum standard cell size
    - Can have site for double height cells!
    - IOs have special SITE.
  - **Grids** for layout and routing
  - **Via definitions**
  - **Others**: Units, Version, …




```
SITE CORE
  CLASS CORE;
  SIZE 0.2 X 12.0;
END CORE

LAYER MET1
  TYPE ROUTING ;
  PITCH 3.5 ;
  WIDTH 1.2 ;
  SPACING 1.4 ;
  DIRECTION HORIZONTAL ;
  RESISTANCE RPERSQ .7E-01 ;
  CAPACITANCE CPERSQDIST .46E-04 ;
END MET1

LAYER VIA
  TYPE CUT ;
END VIA
```

Library/Cell LEF

- **A Library/Cell LEF contains one or multiple cells**

Basic cell information

```
MACRO AN2
    CLASS CORE ;
    FOREIGN AN2 0.000 0.000 ;
    ORIGIN 0.000 0.000 ;
    SIZE 2.480 BY 5.040 ;
    SYMMETRY x y   ;
    SITE core_5040 ;
    PIN O
        DIRECTION OUTPUT ;
        PORT
        LAYER metal1 ;
        RECT  2.000 2.790 2.310 3.190 ;
        RECT  2.030 1.180 2.310 3.300 ;
        RECT  2.000 1.380 2.310 1.780 ;
        END
    END O
```

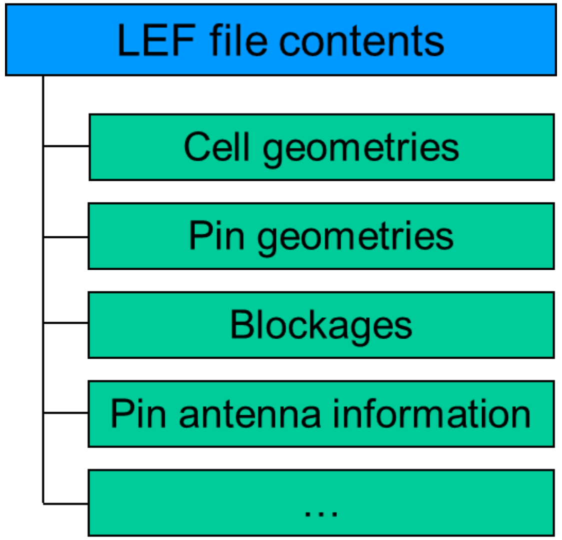
Signal pin information

Power and GND pins in M1

```
PIN GND
    DIRECTION INOUT ;
    USE ground ;
    SHAPE ABUTMENT ;
    PORT
    CLASS CORE ;
    LAYER metal1 ;
    RECT  0.000 -0.380 2.480 0.380 ;
    RECT  1.340 -0.380 1.740 0.560 ;
    END
END GND
PIN VCC
    DIRECTION INOUT ;
    USE power ;
    SHAPE ABUTMENT ;
    PORT
    CLASS CORE ;
    LAYER metal1 ;
    RECT  1.410 4.090 1.810 5.420 ;
    RECT  0.000 4.660 2.480 5.420 ;
    RECT  0.470 4.130 0.870 5.420 ;
    END
END VCC
```

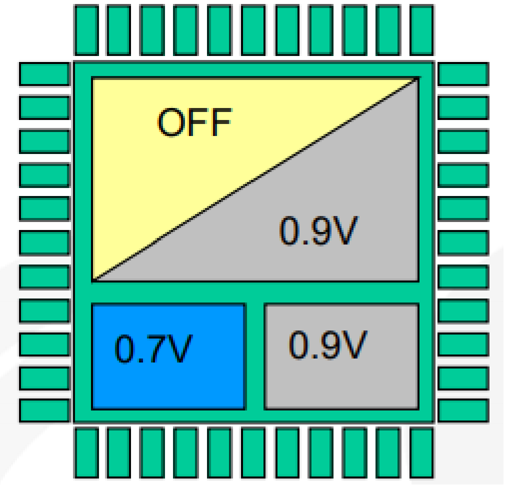
Cell-internal routing: **OBS**truction protects from routing over it

```
OBS
    LAYER metal1 ;
    RECT  0.470 3.410 1.760 3.650 ;
    RECT  1.520 1.080 1.760 3.650 ;
    RECT  1.520 2.250 1.790 2.650 ;
    RECT  0.160 1.080 1.760 1.320 ;
END
END AN2
```



What we have not dealt with in the Frontend

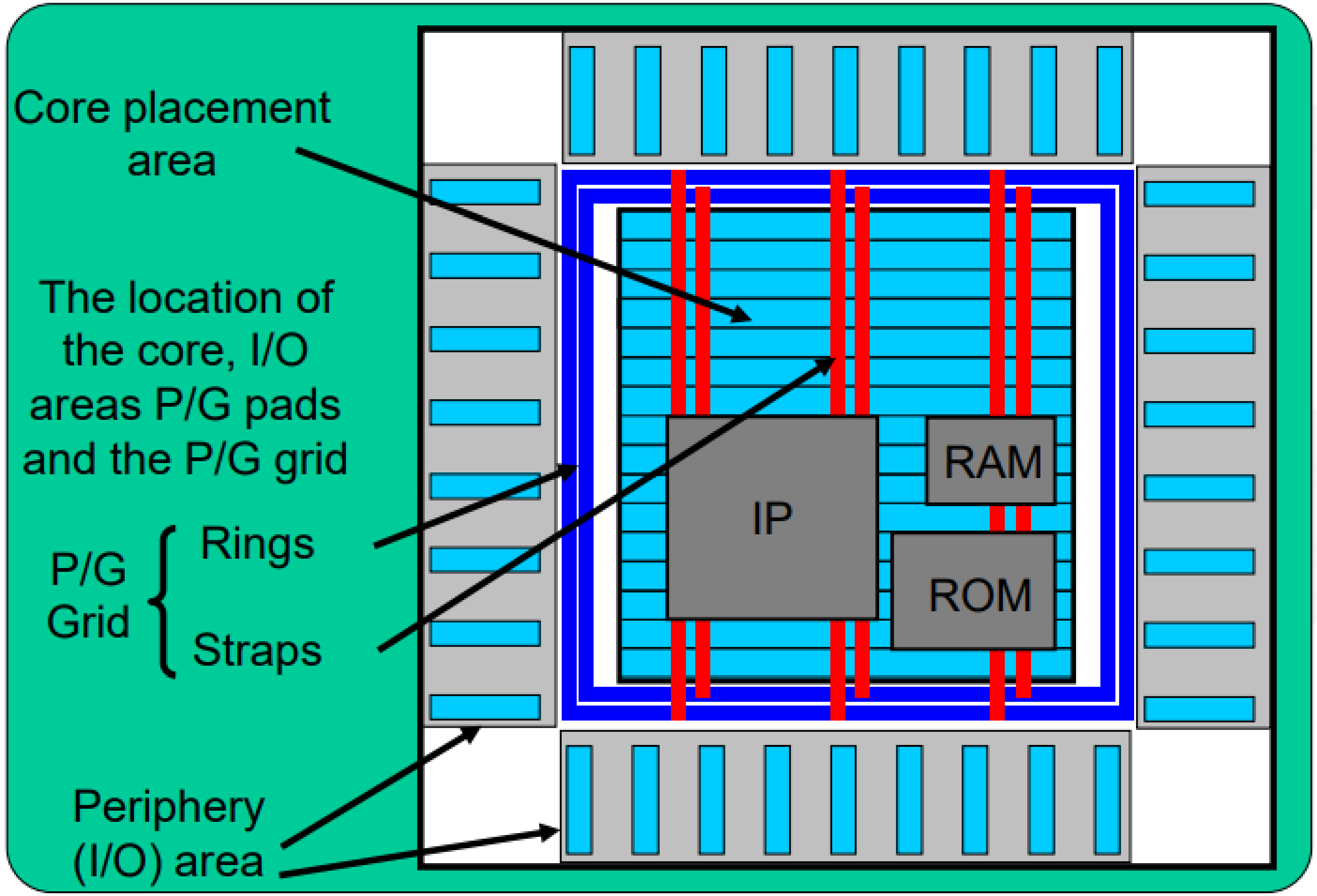
- **Frontend design ignores many important aspects**
  - Power supply:
    - During frontend design, cells do not have power supply connections
    - Routing and connection of clean power supplies was ignored
  - Physical layout: no information on cell location
  - Physical connections: pins were not physically connected with wires
  - Clock distribution: clocks were assumed ideal



- **Physical design needs to take care of these:**
  - Define and connect power supply nets (including voltage domains)
  - Provide physical cell and technology information
  - Take care of nets that were considered ideal (e.g., clocks) and check again timing
  - Add physical cells which are not required for logic functionality:
    - Tie cells (where to get ‘0’ and ‘1’), P/G pads, Filler and DeCap cells, Well tap cells, …

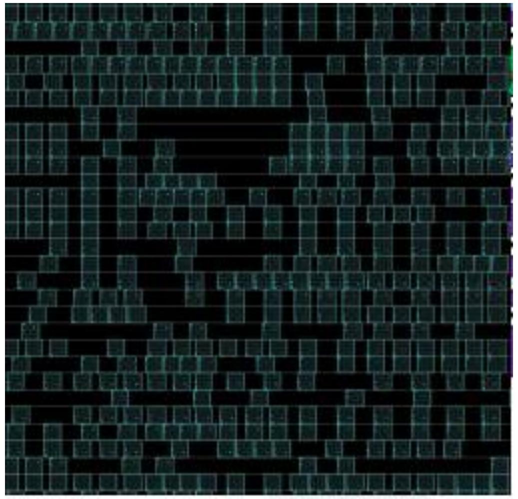
Defining a Floorplan

- **Information required**
  - Design list (logic information)
  - Area requirements: more detailed than just “as small as possible”
  - Timing constraints
  - Physical partitioning information
  - IO information
  - Information on power requirements
- **Defined during floorplanning**
  - Area and shape of the die/block
  - Location of the I/O pads/connections
  - Placement of the large macros
  - Placement regions for standard cells
  - Power distribution and power grid

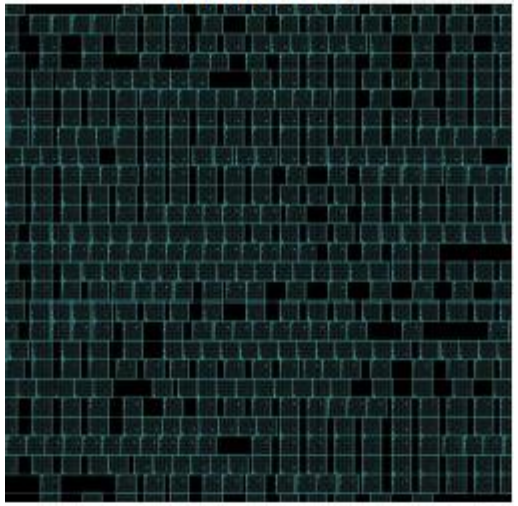


Core Area

- **Core area is determined by the component area and by the utilization**
- **Utilization refers to the percentage of core area taken up by standard cells**
  - High utilization can make it difficult to close a design
    - Routing congestion
    - Negative impact during optimization legalization stages
  - Low utilization affects the area, however
    - It can not always guarantee that there is no congestion (e.g., local congestion)
    - Too low density may lead to long wires and higher power
  - A typical starting utilization might be 70-80%
    - This can vary a lot depending on the design
    - Utilization changes during the post-layout optimization
    - Finding good utilization requires some few iterations



Low standard-cell utilization

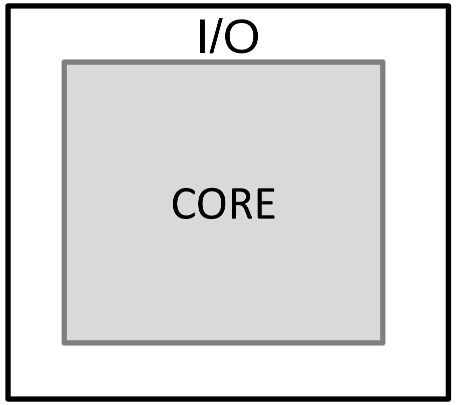


High standard-cell utilization

IO Planning

- **The IO-planning sets the starting point for the floorplan**

- **IO Planning is driven by a few very important constraints and considerations**
  - I/Os do not tend to scale with Moore's Law and therefore, they are very expensive (in terms of area).
  - I/Os have specific constraints to be easy to connect to the package
  - I/Os have special requirements in terms of power and their connection
  - Defines the outline of the circuit and its size

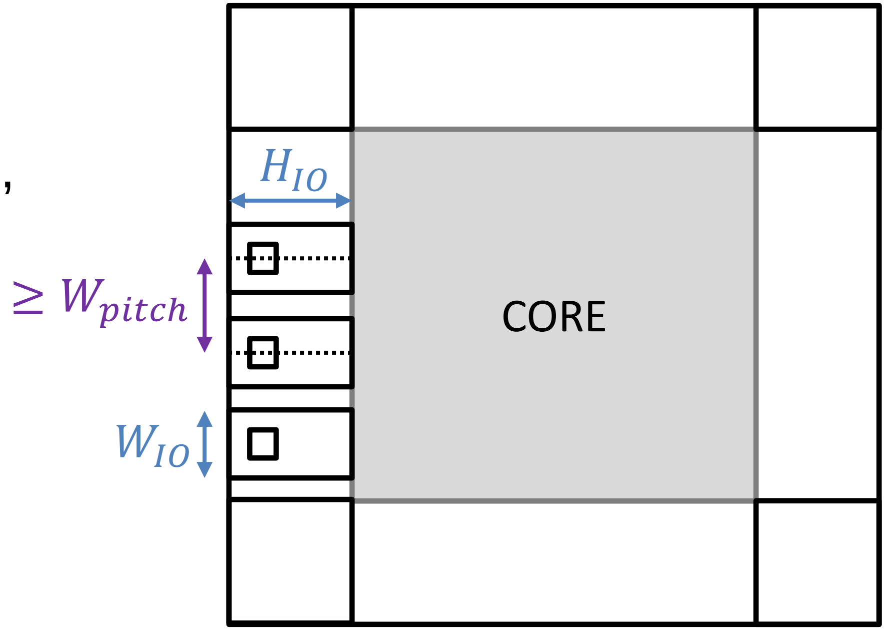


- **I/O drivers are** typically placed around the core and form **the outline of a chip**
  - For blocks, I/Os are also placed on the periphery to be easily accessible

IO Planning: Key Parameters and Questions

- **I/Os (Pads) consume a significant amount of the chip area**
- **Potential questions that will arise during IO planning**
  - Given a fixed die size (as often in MPW projects), how much core area do I have available?
  - Given a core area, how large will my chip be once I add the pads?
  - Given a number of pads, how large must my chip be to fit all pads?

- **Important numbers for IO planning**
  - **IO width W_{IO} and height H_{IO}**: physical dimensions of the IO, **INCLUDING both the driver and the bond pad**
  - The minimum **bonding pitch** W_{pitch}: determined by the capabilities of the packaging house (often significantly larger than the io width)



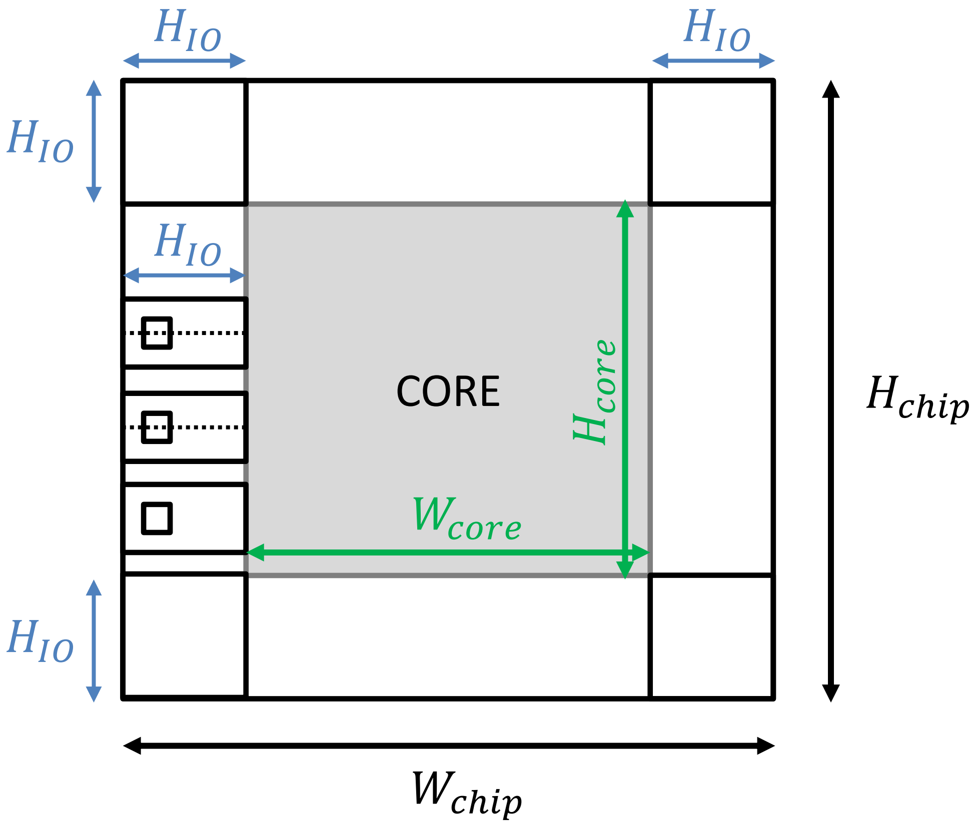
IO Planning: Calculation Examples

- **How much core area do I have?**
  - Given is the chip area $A_{chip} = W_{chip} \cdot H_{chip}$ (e.g., Mini@SICs with a fixed area of 1x1 mm)
  - Given the pad and pitch characteristics: $H_{IO}, W_{IO}, W_{pitch}$

$$W_{core} = W_{chip} - 2 \cdot H_{IO}$$
$$H_{core} = H_{chip} - 2 \cdot H_{IO}$$
$$A_{core} = W_{chip} H_{chip} - 2H_{IO}\left(W_{chip} + H_{chip} - 2H_{IO}\right)$$

  - Example: $W_{chip} = H_{chip} = 1\text{mm},\ H_{IO} = 0.12\text{mm}$

    $\boldsymbol{A_{core} = 0.577\text{mm}^2}$

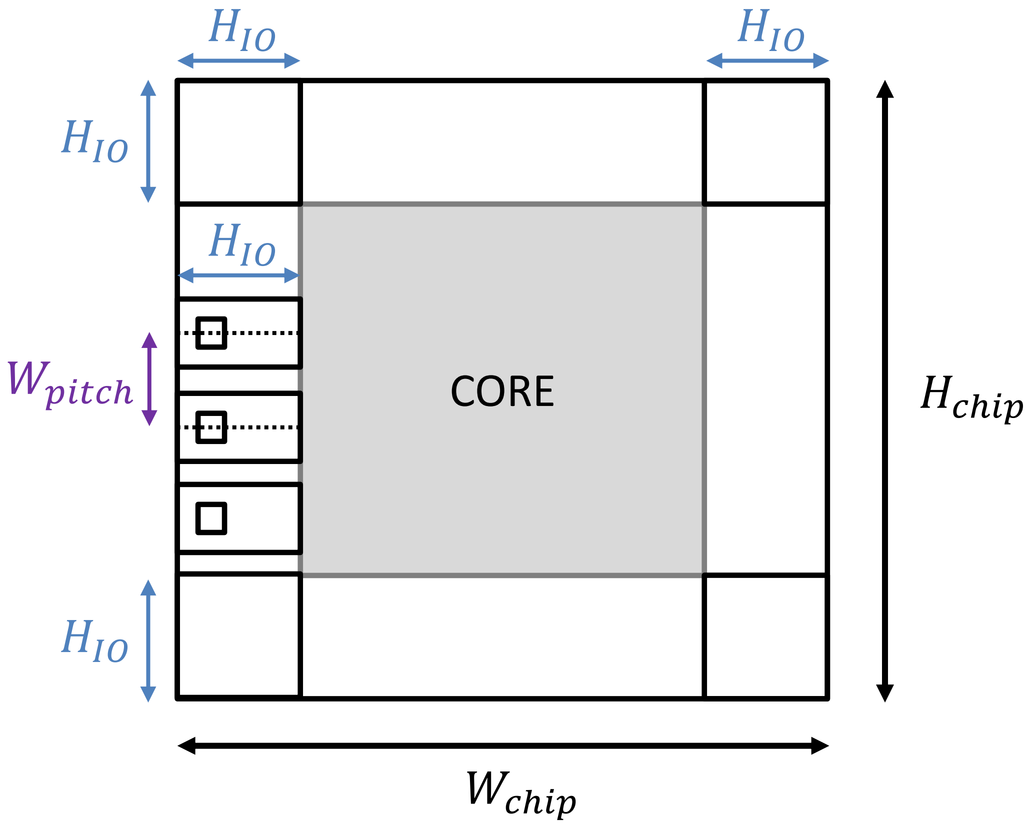


IO Planning: Calculation Examples

- **How many pads can I fit around my chip?**
  - Given is the chip area $A_{chip} = W_{chip} \cdot H_{chip}$
  - Given the pad and pitch characteristics: $H_{IO}, W_{IO}, W_{pitch}$

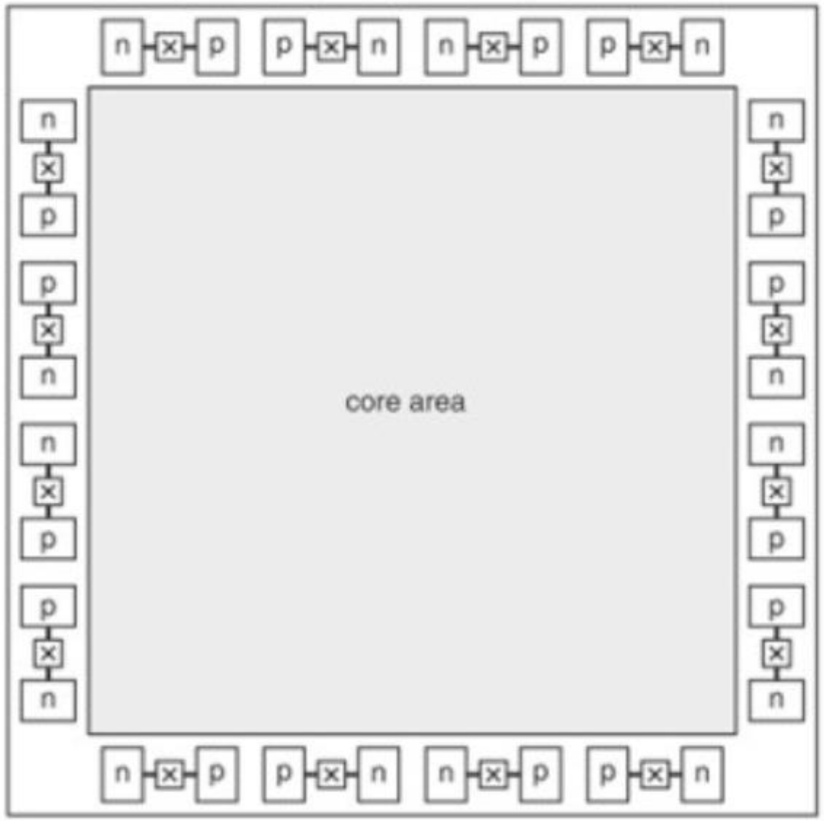
$$\#Pads = 2 \cdot \left(\left\lfloor \frac{(W_{chip} - 2 \cdot H_{IO} - 2W_{IO})}{W_{pitch}} \right\rfloor + 1 \right)$$
$$+2 \cdot \left(\left\lfloor \frac{(H_{chip} - 2 \cdot H_{IO} - 2W_{IO})}{W_{pitch}} \right\rfloor + 1 \right)$$

  - Example: $W_{chip} = H_{chip} = 1\text{mm},$
    $H_{IO} = 0.12\text{mm}, W_{IO} = 0.05\text{mm}$
    $W_{pitch} = 0.07\text{mm}$
    $\#Pads = \mathbf{44}$



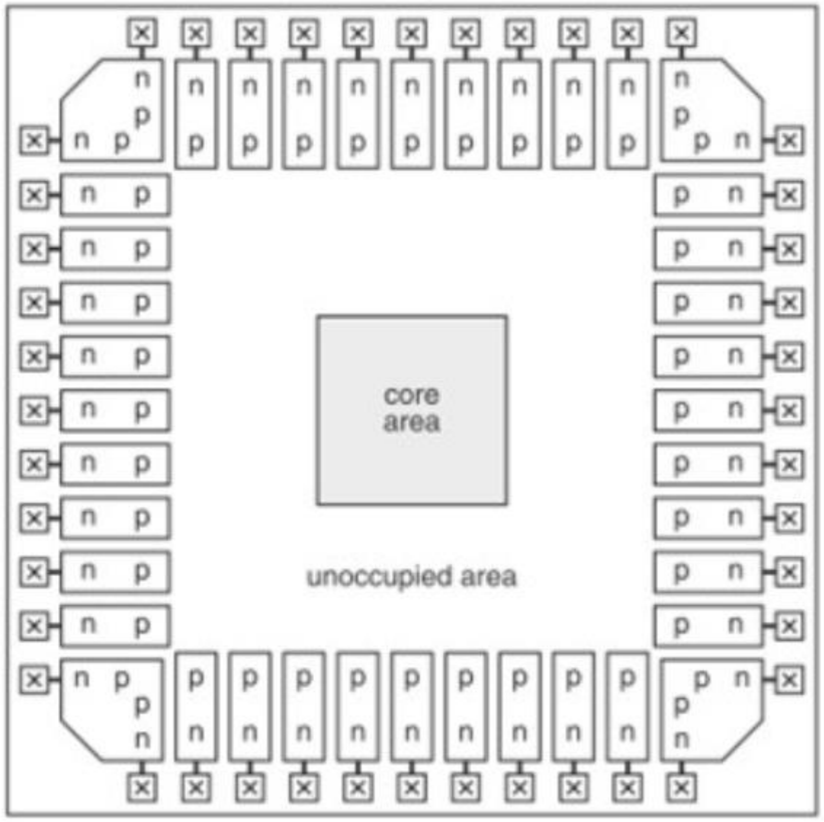
Pad vs. Core Limited Chips

- **Chip size is either defined by the pads or by the size of the core**
  - Determines the type and aspect ratio of the pads to minimize chip area



Core Limited

b)



Pad Limited

IO Planning: Power

- **I/Os are not only needed for connecting signals to the outside world, but also to provide power to the chip.**

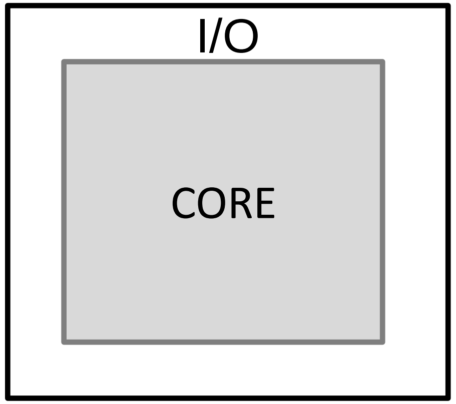
- **Every chip need to have at least the following supply voltages**

  CORE supply

  - VDD: supply for the core logic (typically 0.6 – 1.2 V depending on process)
  - GND: ground for the core logic

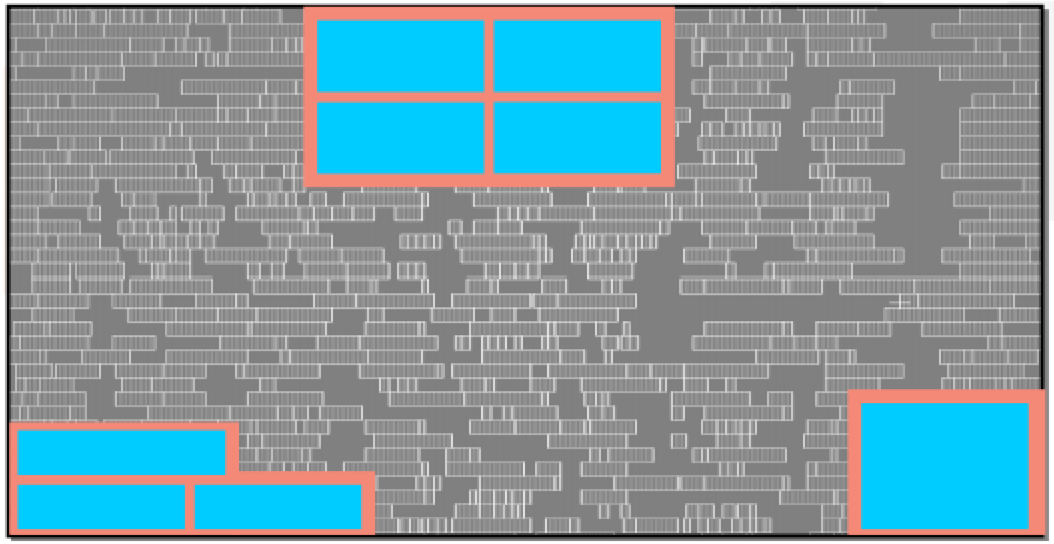
  IO supply:

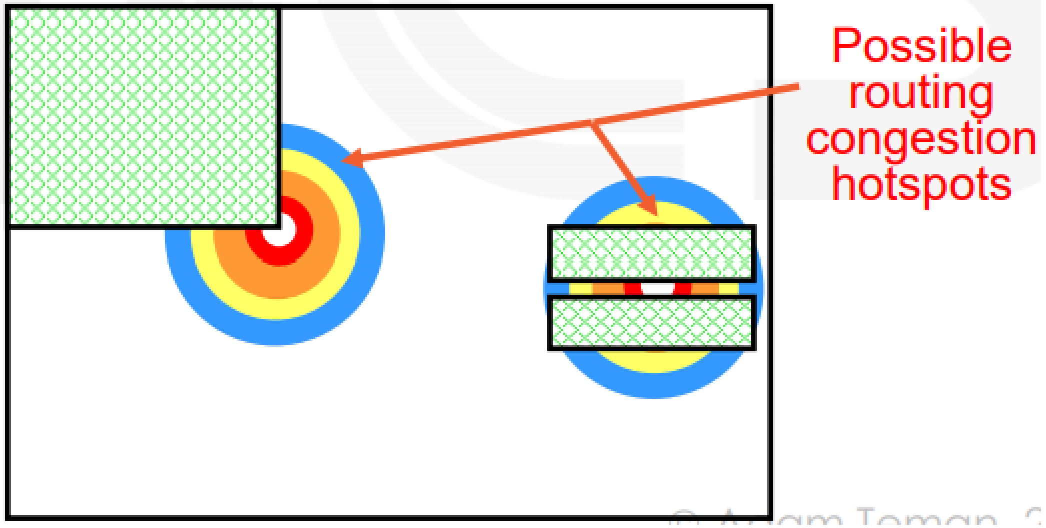
  - VDD IO supply for the IO (off-chip) drivers (typically 1.8 – 3.3 V depending on IO standard)
  - GND IO ground for the IO (off-chip) drivers



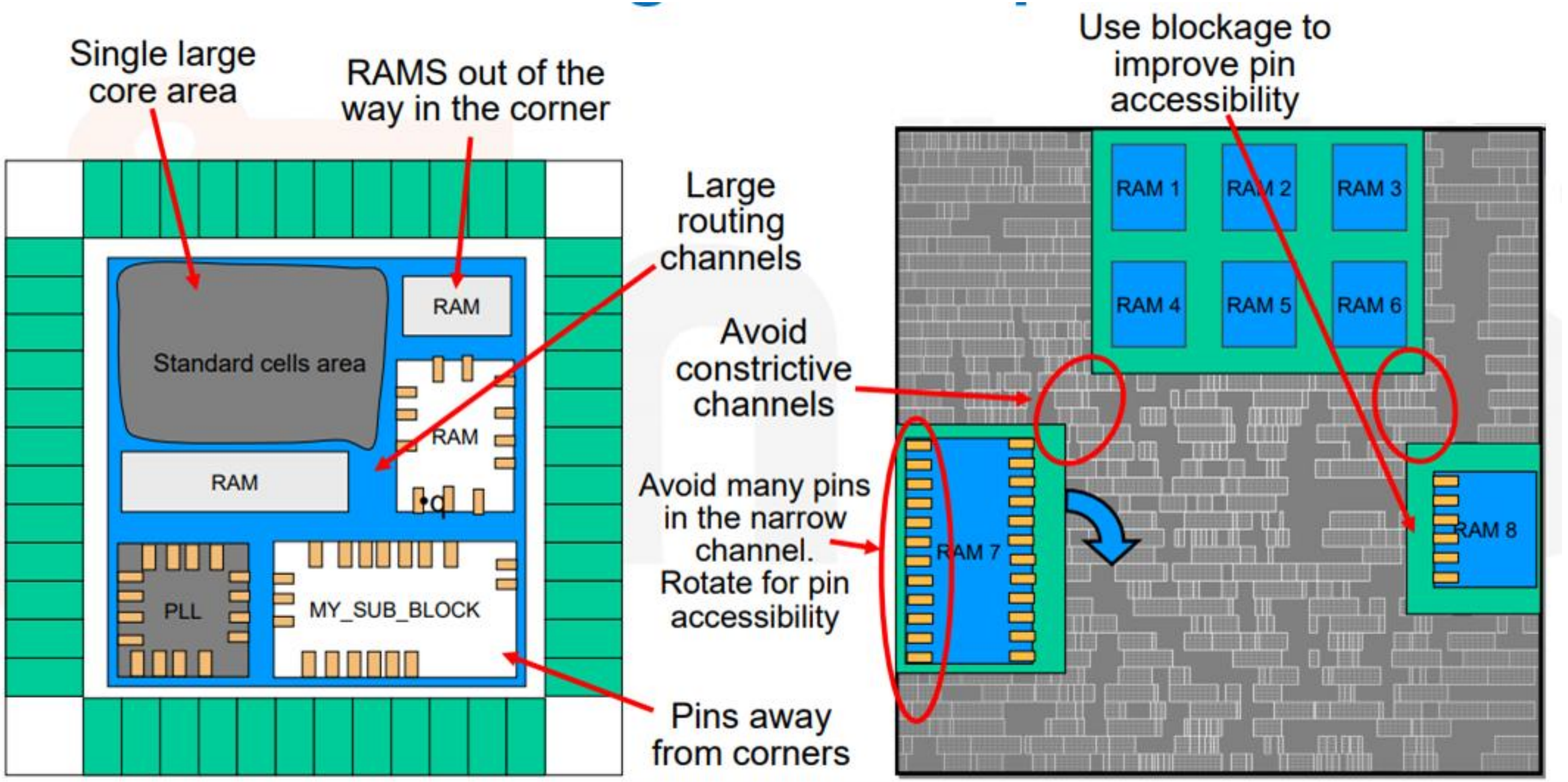
Macro Placement

- **Core floorplan design starts with placing the large macros**
  - Macros are typically placed manually (block placement tools exist, but do not perform well)
- **Macro placement must consider the impact on routing, timing and power**
  - Macros are often placed on the sides of the floorplan
    - Standard-cell placement performs better with a single large rectangular area
    - Power hungry macros are better kept close to the power supply pins (on the periphery)
    - **Macros often** have many pins, while blocking many layers which often **lead to congestion**



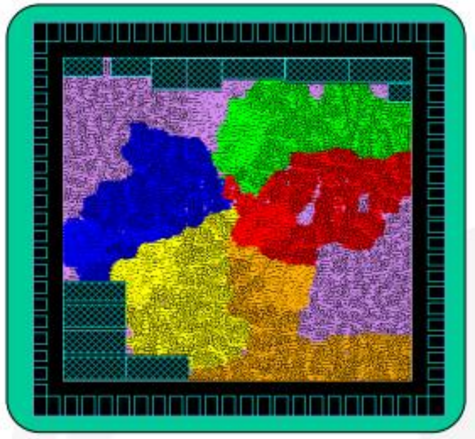


Some Guidelines for Macro Placement

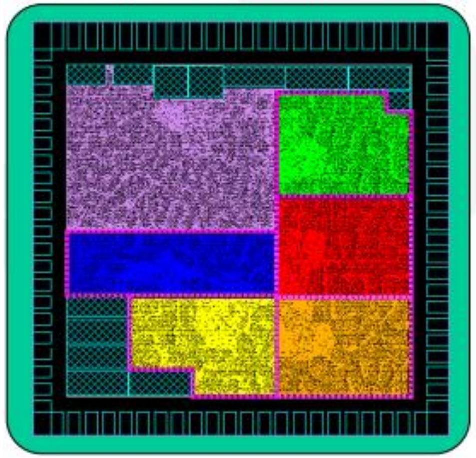


Standard Cell Placement Guidance

- **Standard cells are placed by automatic tools in few large dedicated areas**
  - Manual placement is not an option and strong constraints limit the freedom of the tool
  - Nevertheless some guidance is often desirable



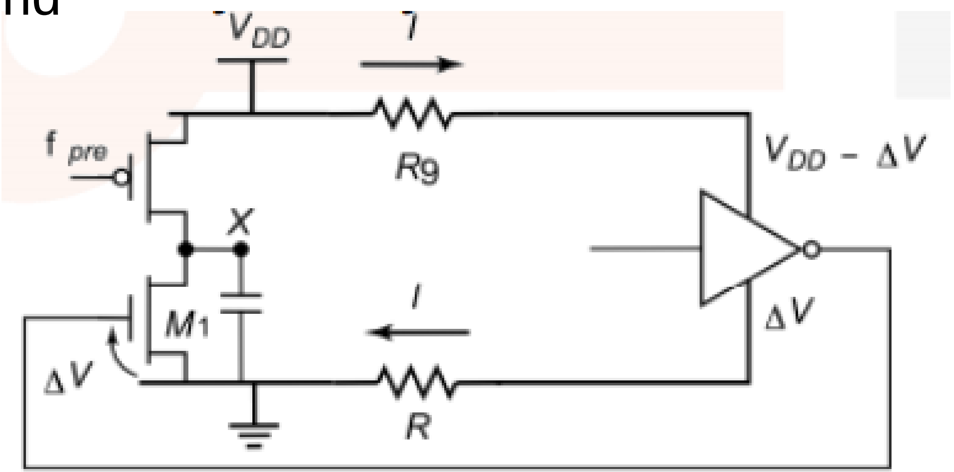
- **Place and Route tools define several types of placement regions:**
  - Soft guide – try to cluster these cells together without a defined area.
  - Guide – try to place the cells in the defined area.
  - Region – must place the cells in the defined area, but other cells may also be placed there.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

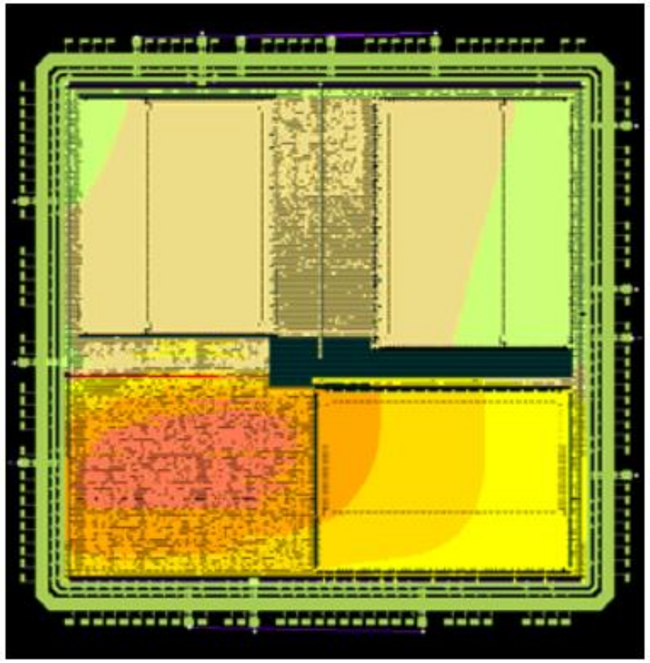


Power Planning

- **Standard cells and IP blocks must be connected to power and ground**
  - VDD and GND are usually defined as global nets
  - Complex designs can have multiple supplies (VDD1, VDD2, …)

- **Power is delivered through the pads (often from the side of the chip)**
  - Need to distribute current to all circuit components on the chip
  - IPs and standard cells behave as distributed current sources
  - Resistance of the supply grid leads to IR voltage drop

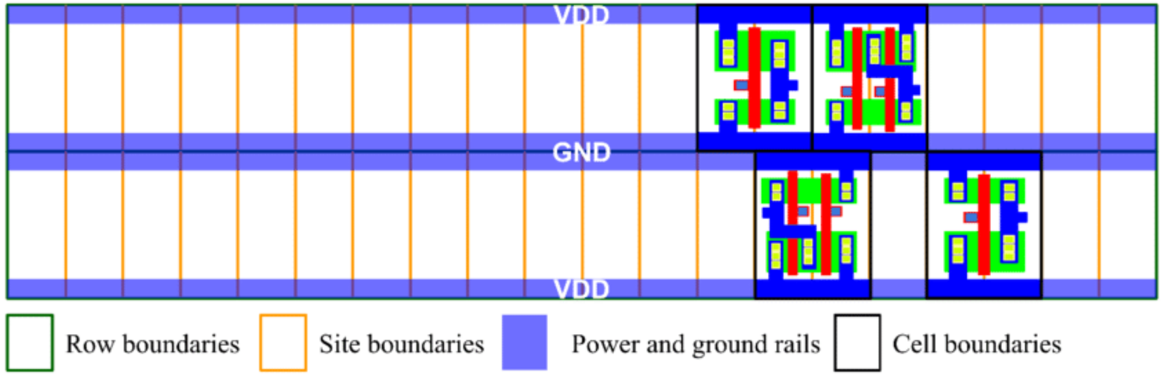


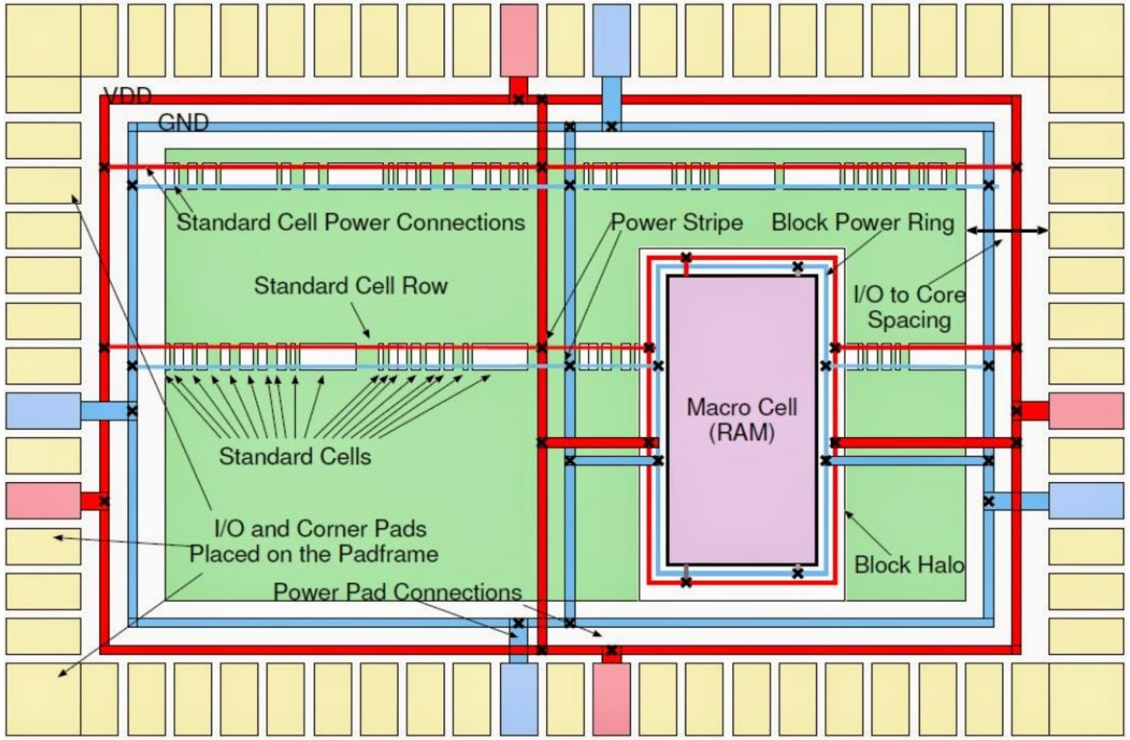


Power Planning

- **Power planning involved two steps:**
  - **Definition** and connection **of the global power grid (distribution network)**
  - **Connection** of the power and ground pins of standard cells and IP blocks **to the global grid**

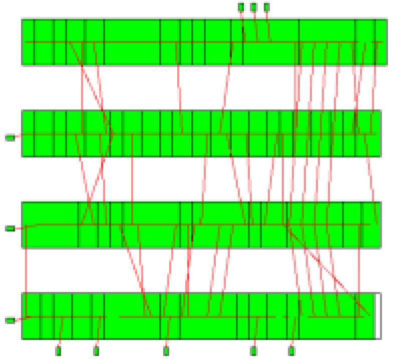
- **Macros have dedicated individual power pins**
- **Standard cells are connected with a local grid to their neighbours**



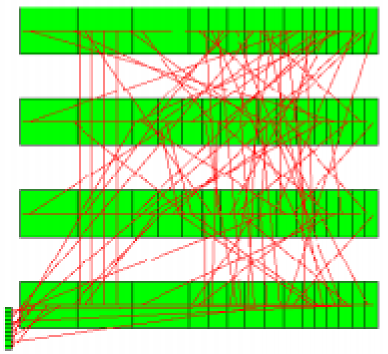


Automatic Placement

- **Standard cells are placed automatically with the objective to**
  - **Minimize routing congestion** (congestion-driven placement)
  - Optimize the timing (timing-driven placement)

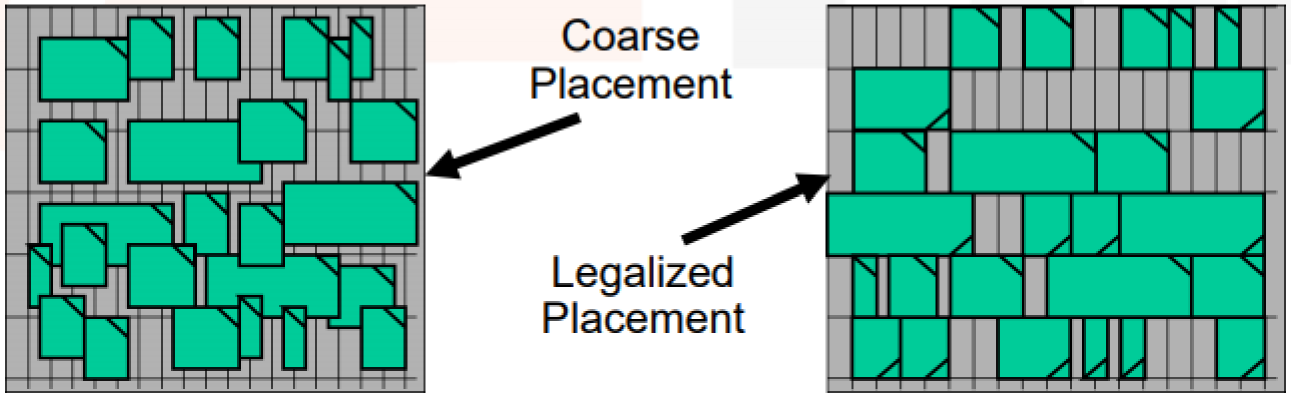


good placement



bad placement

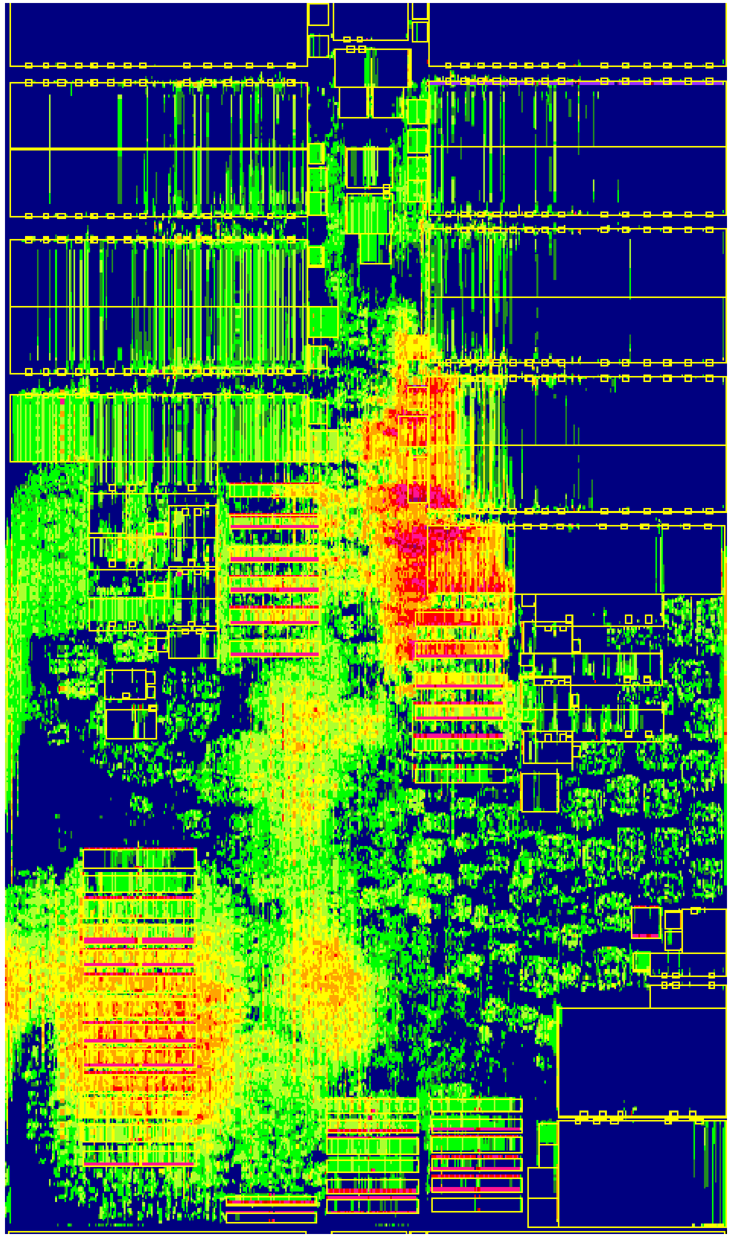
- **Placement is often performed in two steps:**
  - Initial placement cares only about congestic
    but does not check for legal cell locations
  - Once a good initial placement is found, cell locations are “legalized” and optimized in iterations



Congestion Maps

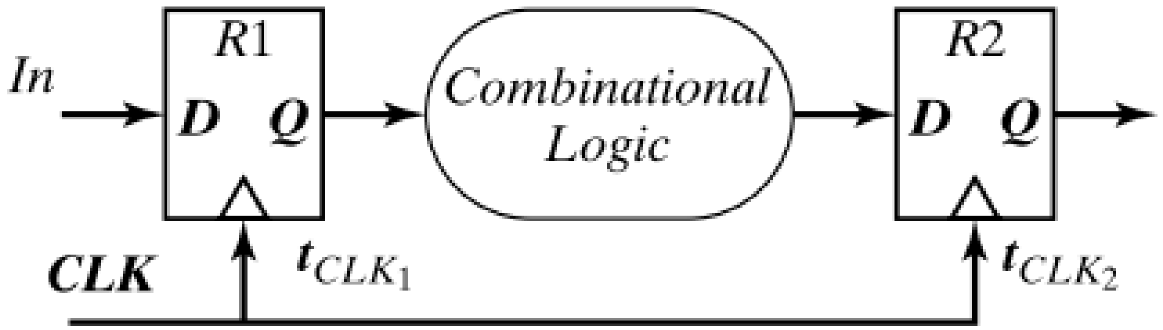
- **During placement, a rapid global routing is conducted to estimate routing congestion**
  - The expected routing density is shown in a **congestion map**

- **Reducing congestion is achieved by**
  - **Rearranging** the location of **macro blocks** that cause congestion
  - Providing **routing channels** (e.g., with halos to facilitates access to macro pins)
  - **Reducing** placement **density (utilization)**
  - Defining **placement constraints**
    - Often overestimated and can do more bad then good

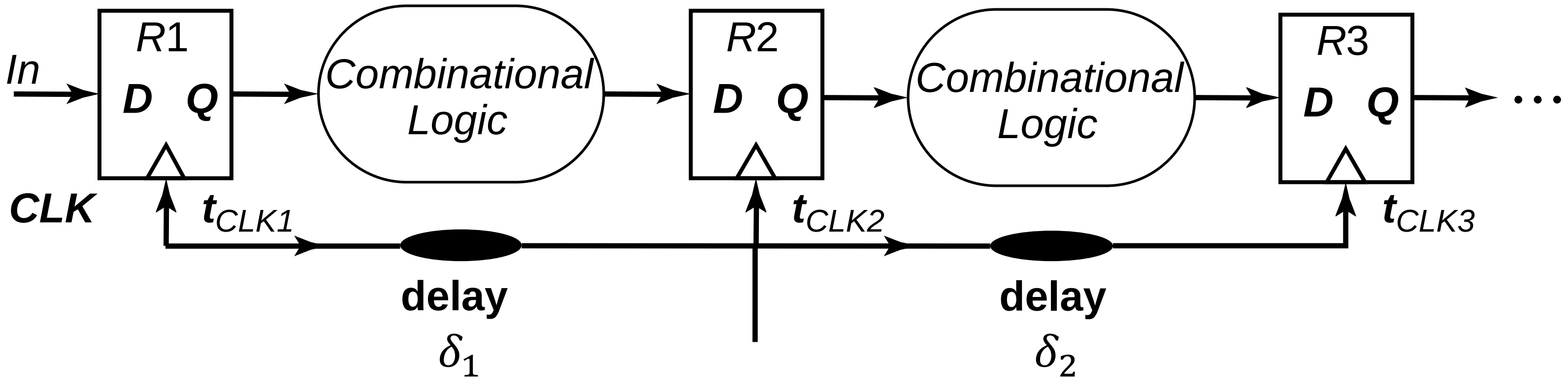


Clock Distribution

- **Reminder: synchronous design assumes that all flip-flops are clocked at (exactly) the same time**
  - During frontend design, clock signals were considered ideal

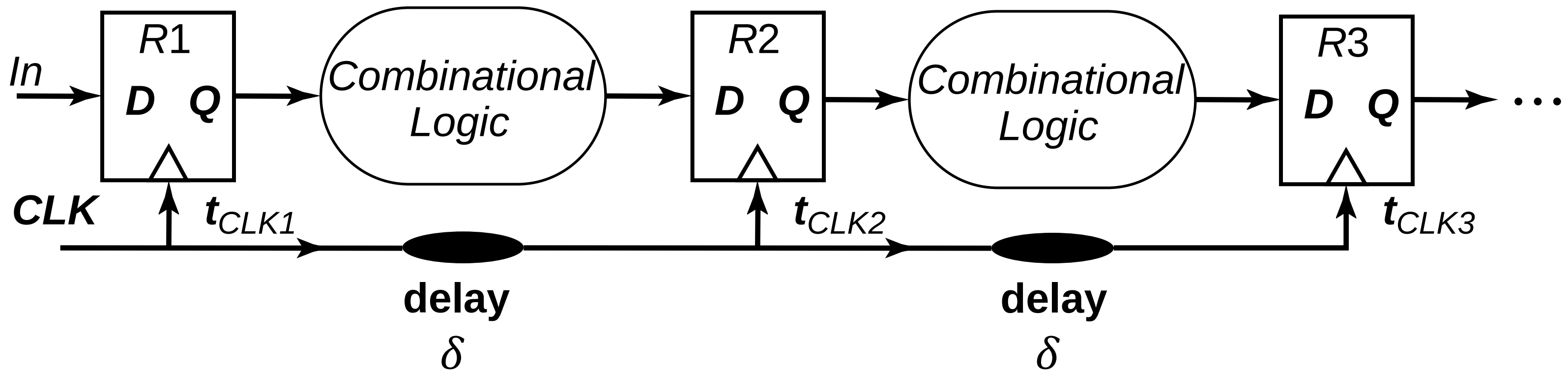


- **Inequalities in the clock distribution leads to CLOCK SKEW**



Impact of Non-Ideal Clock Distribution

- **Clock skew can be positive or negative.**
  - In a typical design we will mostly experience both positive and negative skew

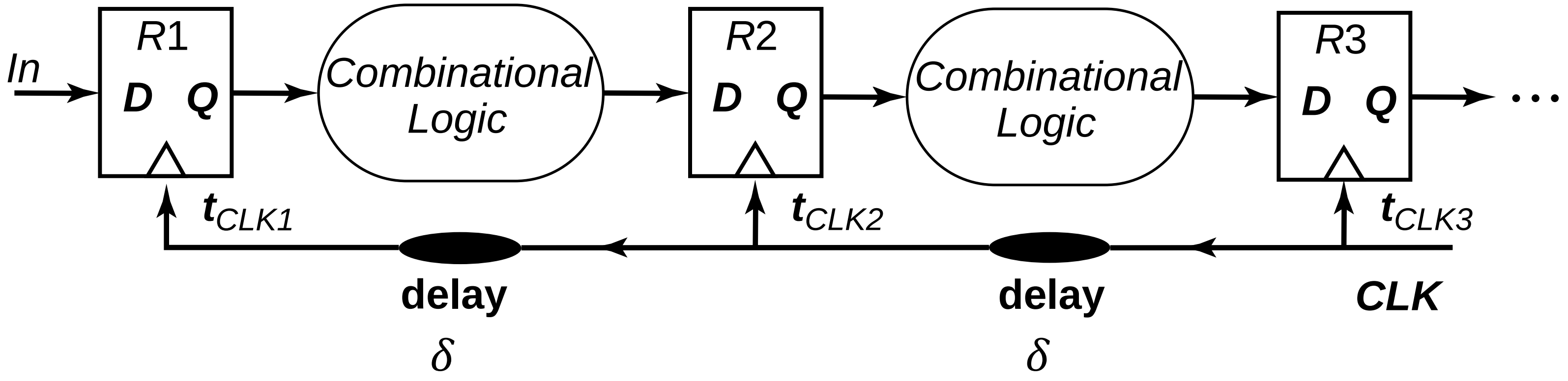


Positive skew

	Setup check	Hold check
No Skew	$T_{PD} < T_{CLK} - T_{setup}$	$T_{PD} > T_{hold}$
Skew	$T_{PD} < T_{CLK} + \delta - T_{setup}$ ✓	$T_{PD} > T_{hold} + \delta$ ✗

Impact of Non-Ideal Clock Distribution

- **Clock skew can be positive or negative.**
  - In a typical design we will mostly experience both positive and negative skew



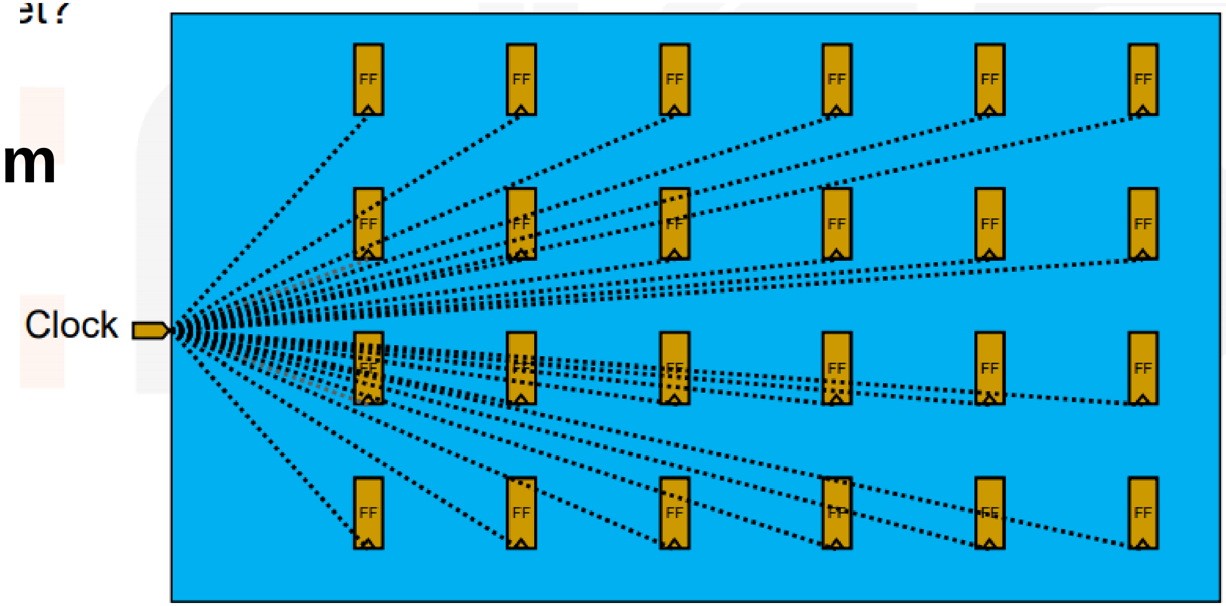
Negative skew

	Setup check	Hold check
No Skew	$T_{PD} < T_{CLK} - T_{setup}$	$T_{PD} > T_{hold}$
Skew	$T_{PD} < T_{CLK} - \delta - T_{setup}$ ✘	$T_{PD} > T_{hold} - \delta$ ✓

Clock Distribution

- **Clock skew leads to serious timing issues and must be minimized**
- **During physical design, the clock must be distributed to all sequential elements with the objective to**
  - Avoid arrival time differences between the individual flip-flops
  - Maintain a clock with sharp transitions at the clock pins of the sequential elements
  - Minimize delay from clock input to the flops

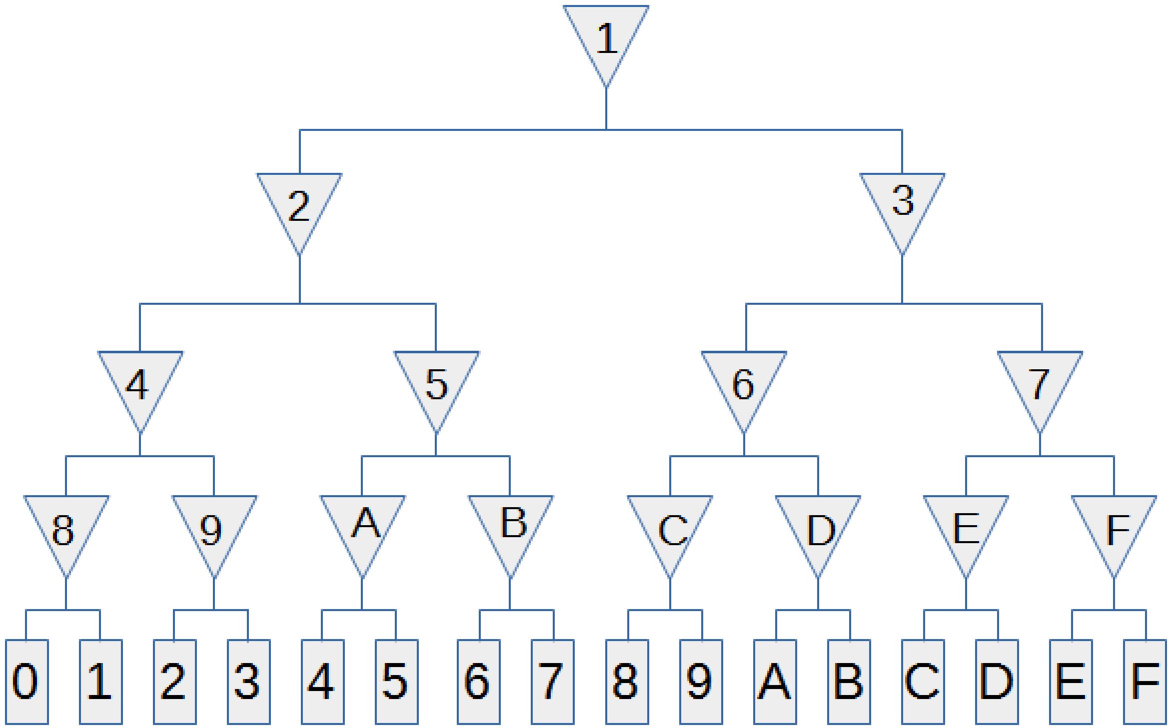
- **Straightforward clock distribution from a single (even strong) driver leads to large skew and uncertainty**
  - Differences in wire length lead to differences in RC delays
  - Signal quality with long wires is difficult or even impossible to control



Clock Tree Synthesis

- **How to distribute a clock signal well to millions of flip-flops?**
  - Need to manage a very large load
  - Wires need to be relatively short to avoid strong delay variations due to different wire lengths

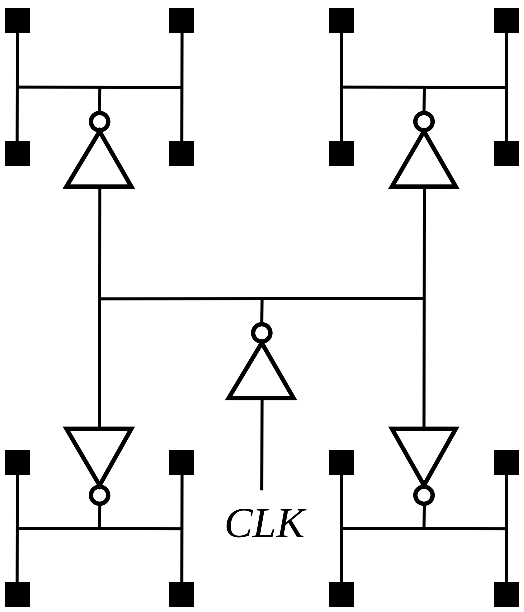
- **Buffer trees have many advantages:**
  - Short nets mean lower RC values
  - Buffers restore the signal for better slew rates
  - Lower total insertion delay (tree depth only logarithmic in the load)
  - Potential to balance the tree by sizing individual branches



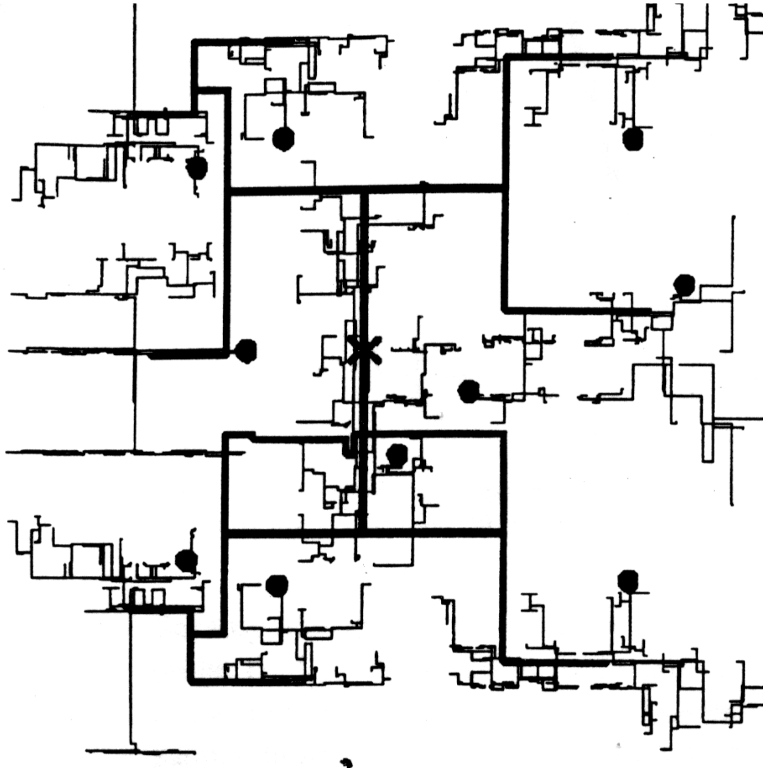
H Tree Clock Distribution

- **Clock tree must be distributed across the chip**
  - Balancing delays requires a regular structure

- **H-Trees distribute the clock tree in a regular pattern** across the chip



Ideal H-Tree



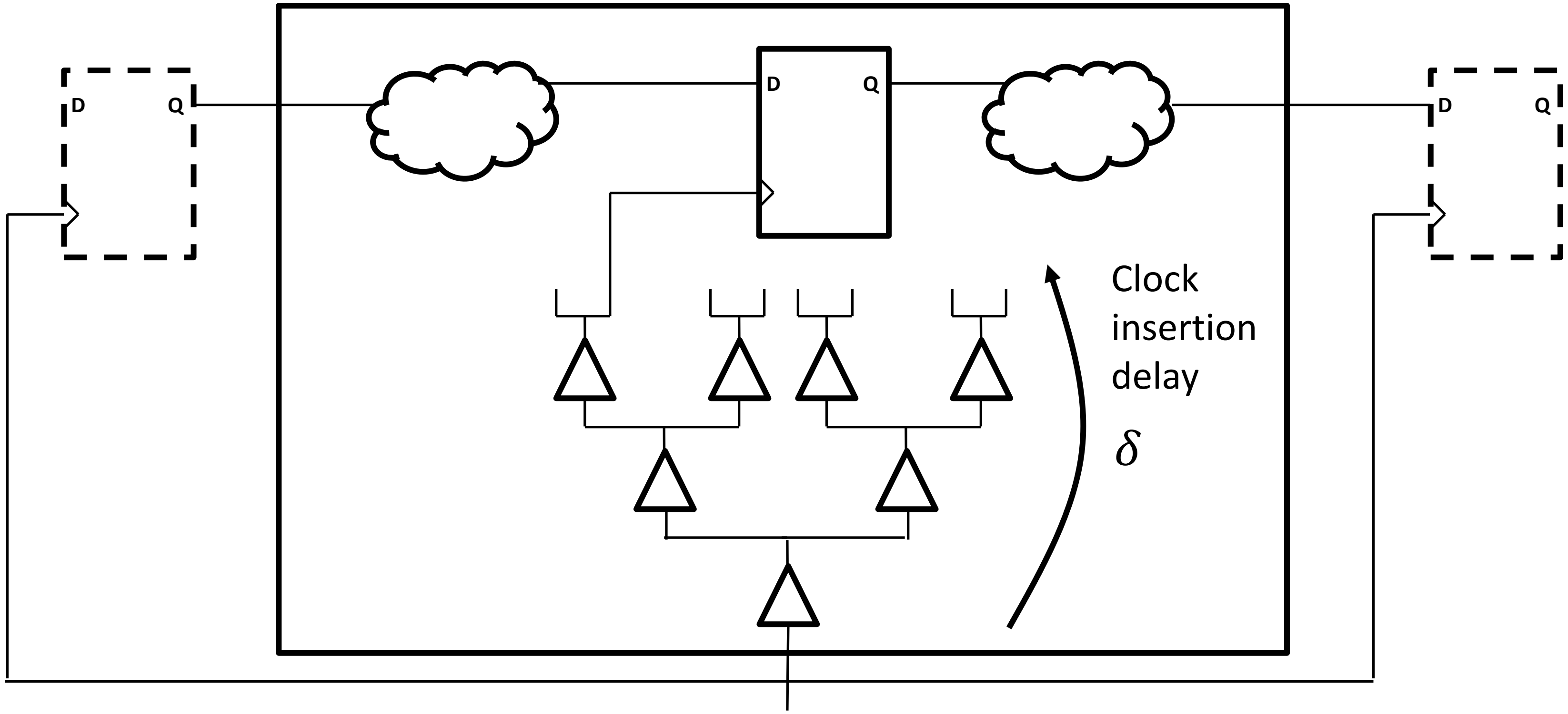
IO Timing with Clock Tree

- **Clock tree insertion delay makes IO timing more difficult:**
  - Long hold time at the input
  - Late arriving outputs

$$t_\Delta > \delta - t_{is} + t_{ho}$$

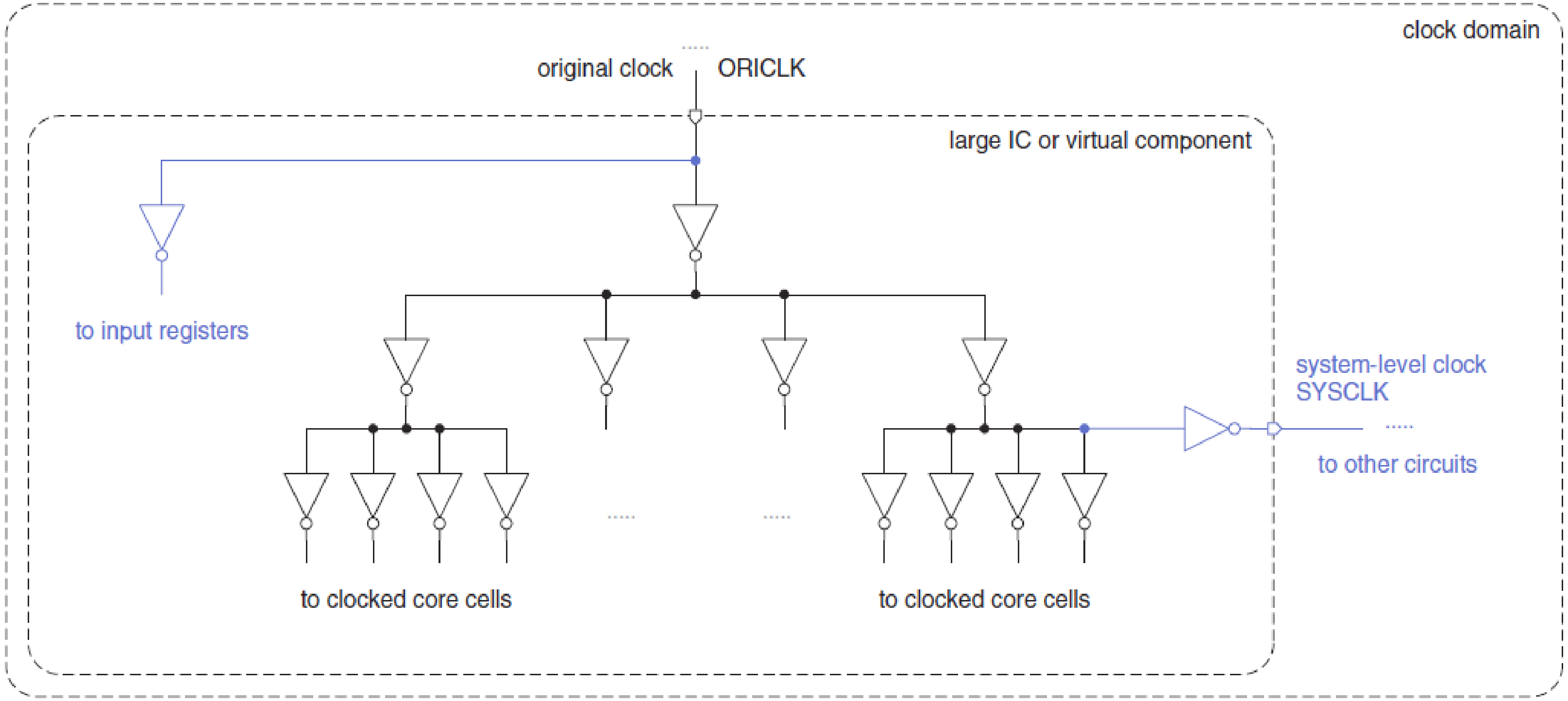
$$t_\Delta < T_{clk} + \delta - t_{su} - t_{is}$$

$$t_T > \delta + t_{so}$$



Improving IO Timing with Clock Trees

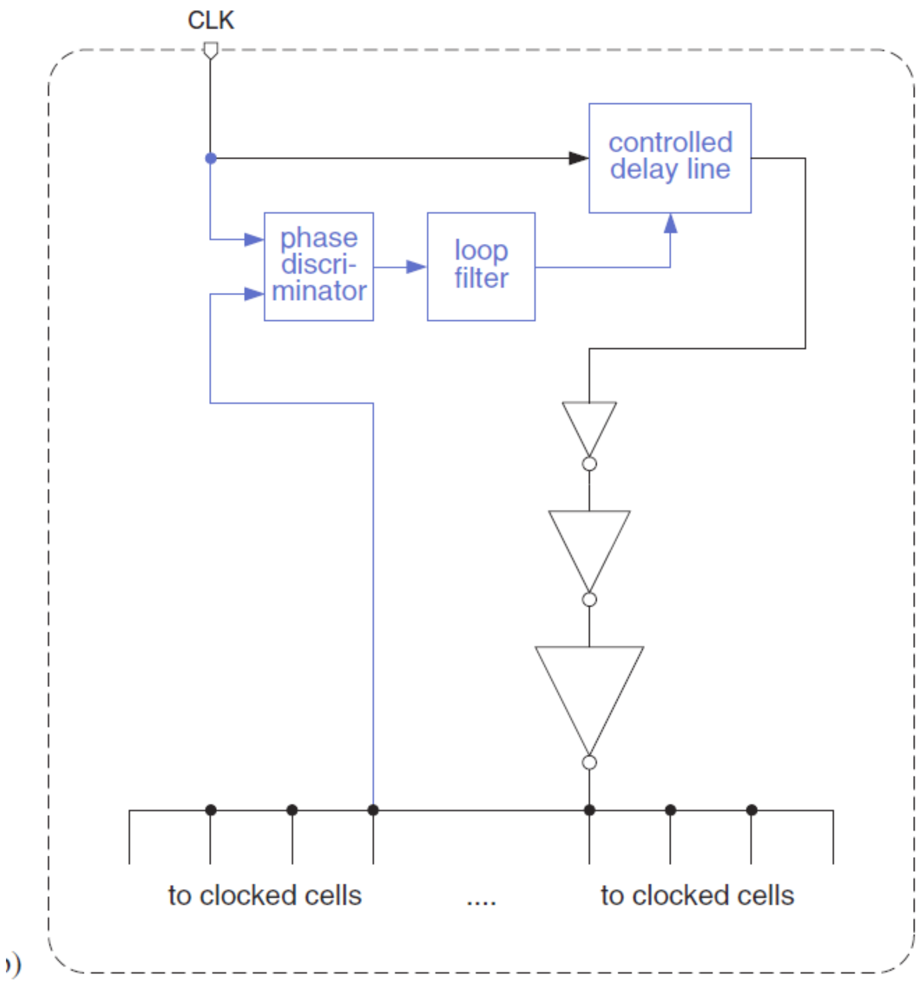
- **Chip provides a clock output that is aligned with the clock at the leaves of the FlipFlops**
  - Output of the clock output pad is declared as clock leaf
  - Delayed clock is used as a reference for rest of system



Improving IO Timing with Clock Trees

- **Delay locked loop**
  - Generates a phase shifted clock such that the reference input is phase aligned with the input clock
  - Clock reference is taken from the leaves of the clock tree

- **Internal clock at the leaves is aligned with clock input**



Other Important High-Fanout Nets: RESET

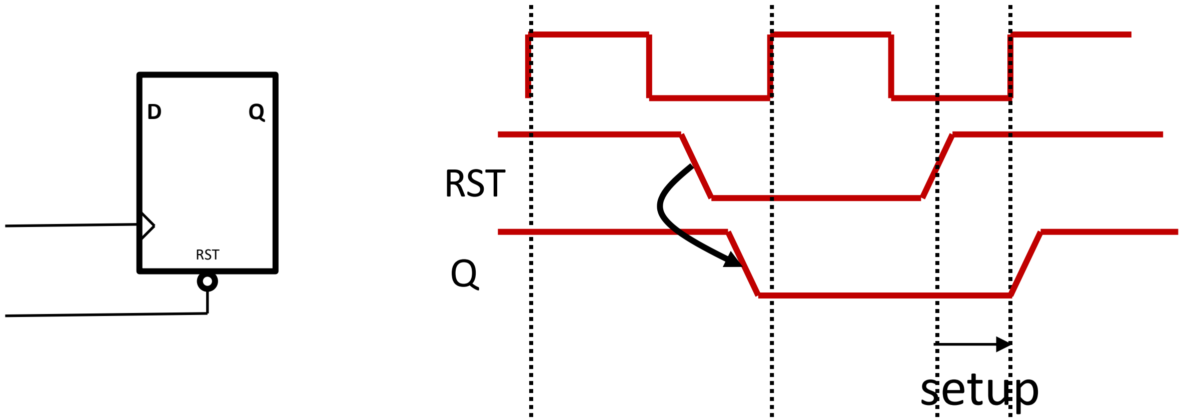
- **In addition to the clock, the global Asynchronous RESET also leads to an extremely high-fanout global net**

- **Reset net is another candidate for a well-designed buffer tree**
  - No serious constraints on skew, except that clock skew should be much less than clock period ($\delta \ll T_{clk}$)
  - Rise/fall time should be reasonable to meet electrical requirements

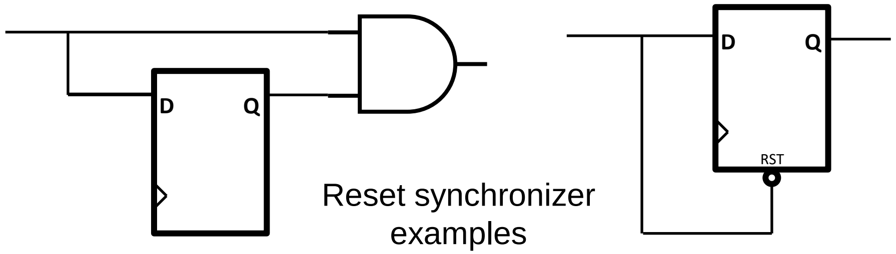
- **Sometimes clock generation tools can be used to also generate reset trees.**
  - Less often as logic buffering during placement optimization has gotten better

Asynchronous Reset Synchronizer: Best Practice

- **Asynchronous reset is applied in an "asynchronous" manner**
  - No timing constraints on reset application
- **Removal of reset signal is subject to timing requirements (setup constraint)**



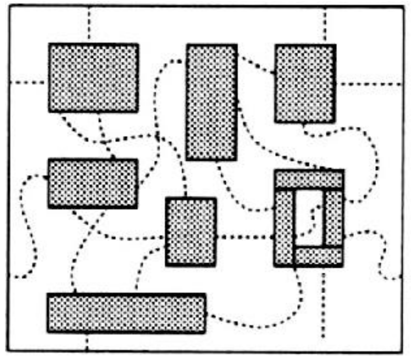
- **Reset synchronizer at chip level: avoid off-chip constraints on reset removal**
  - Pass reset assertion immediately to sequential elements
  - Ensure alignment of asynchronous reset removal with clock edge to avoid timing violations



Reset synchronizer examples

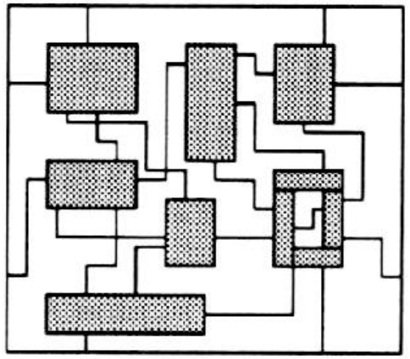
Routing

- **Given a placement, and a fixed number of metal layers, find a valid pattern of horizontal and vertical wires that connect the terminals of the nets**
  - Input: Cell locations, netlist
  - Output: Geometric layout of each net connecting various standard cells



Global Routing

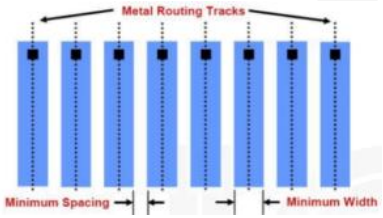
- **Routing is a two-step process**
  - **Global routing**: Rapid initial routing without committing to a precise wire location or geometry rules, with some congestion minimization
  - **Detailed routing**: Accurate final routing, guided by the initial global routing outcome

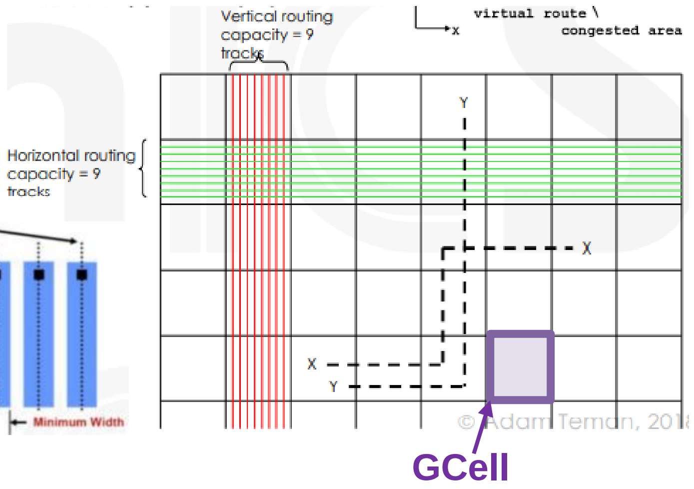


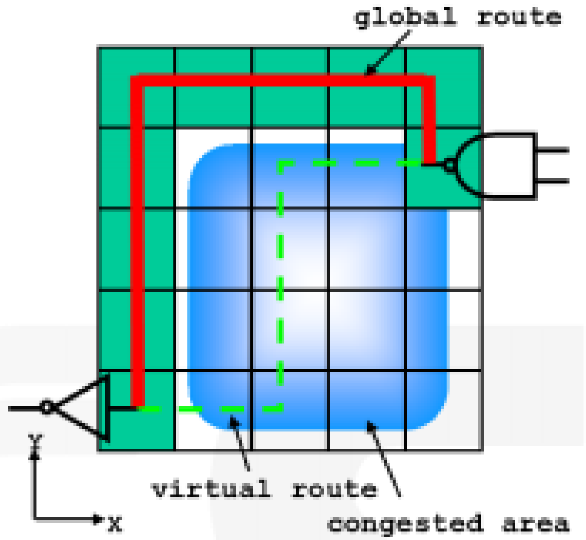
Detailed Routing

Global Routing

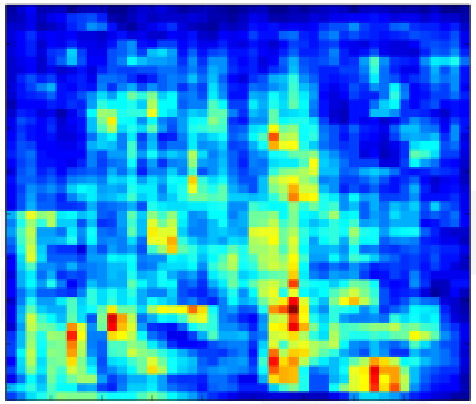
- **Divide floorplan into GCells**
  - Approximately 10 tracks per layer each
- **Perform fast grid routing:**
  - Only identifies crossed GCells
  - **Balance Congestion**
  - Minimize wire-length
  - Keep buses together
- **Outputs:**
  - Global route as basis for detailed routing
  - Congestion map







Congestion map



Congestion Report

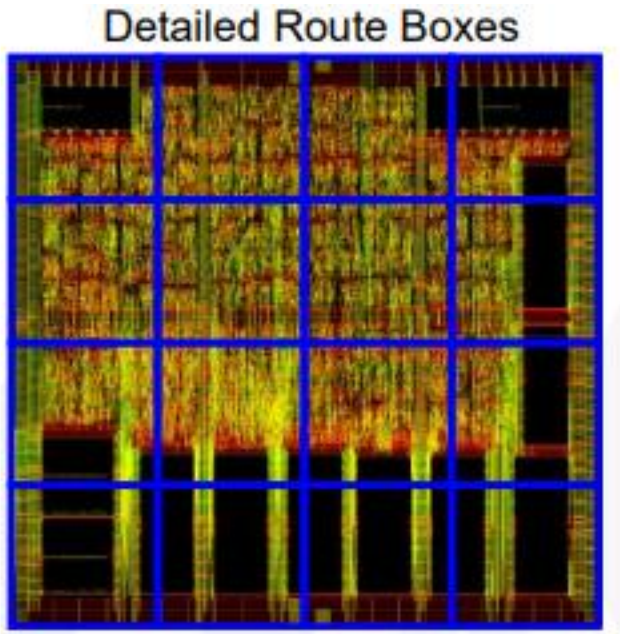
# Layer	Routing Direction	#Avail Track	#Track Blocked	#Total Gcell	%Gcell Blocked
# Metal 1	H	7607	9692	1336335	62.57%
# Metal 2	H	7507	9792	1336335	55.84%
# Metal 3	V	7636	9663	1336335	59.51%
# Metal 4	H	8609	8691	1336335	52.02%
# Metal 5	V	5747	11551	1336335	56.39%
# Metal 6	H	5400	11899	1336335	55.09%
# Metal 7	V	1831	2486	1336335	55.30%
# Metal 8	H	2415	1903	1336335	43.85%
# Total		46753	56.99%	10690680	55.07%

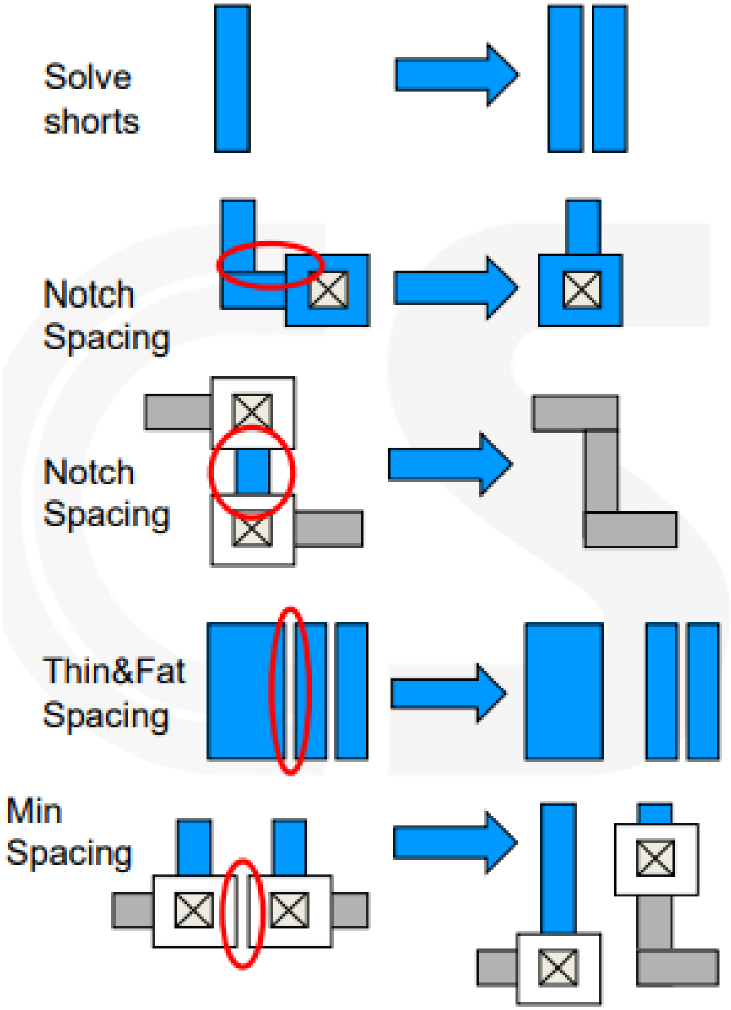
\# 589 nets (0.47%) with 1 preferred extra spacing.

- **Also used for “trial route” earlier in the flow** (e.g., during placement)

Detail Routing

- **Using global route plan, within each global route cell**
  - Assign nets to tracks
  - Lay down wires
  - Connect pins to nets
  - Solve DRC violations
  - Reduce cross couple cap
  - Apply special routing rules

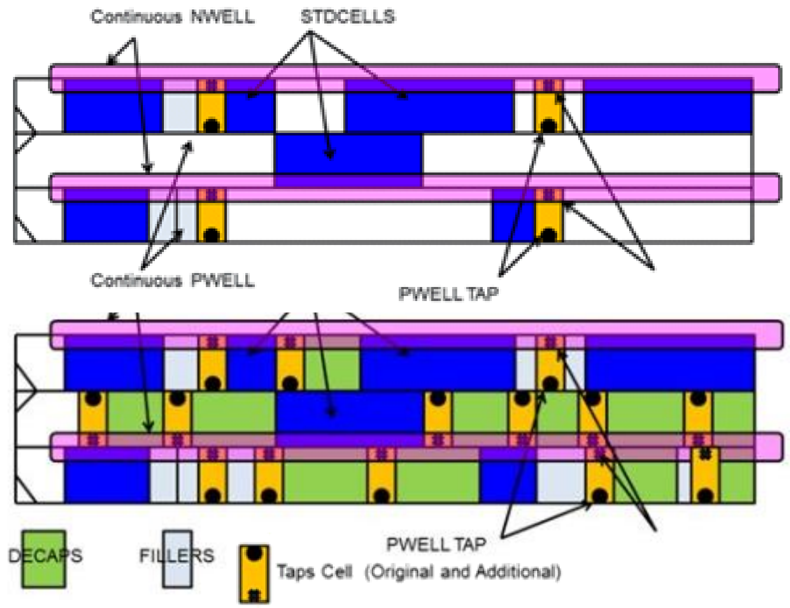




- **Flow:**
  - Track Assignment (TA)
  - DRC fixing inside a Global Routing Cell (GRC)
  - Iterate to achieve a solution (default ~20 iterations)

Filler Insertion

- **Standard Cells almost never fill all the available space** (even in areas reserved for standard cells)

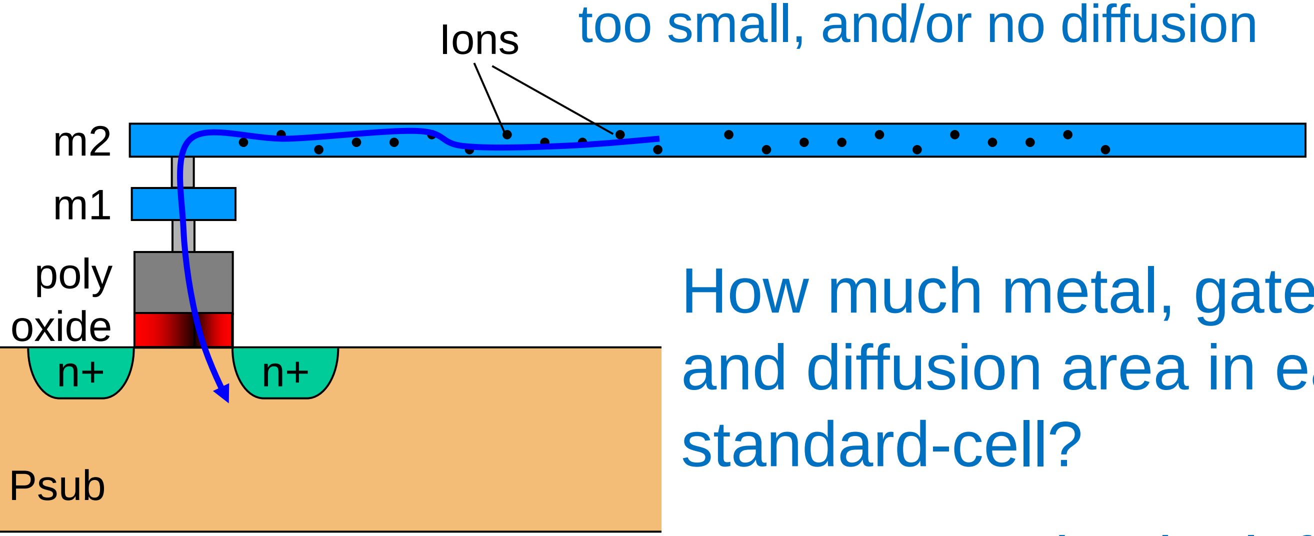


- **Gaps between standard cells are filled to**
  - Ensure continuous wells across the entire row
  - Ensure VDD/GND rails (follow pins) are fully connected
  - Ensure proper GDS layers to pass DRC
  - Ensure sufficient diffusion and poly densities
  - In scaled processes, provide regular poly/diffusion patterning

- **We can also use the gaps to add DeCap cells as fillers**
  - Provide on-chip decoupling capacitance to stabilize and support the power supply

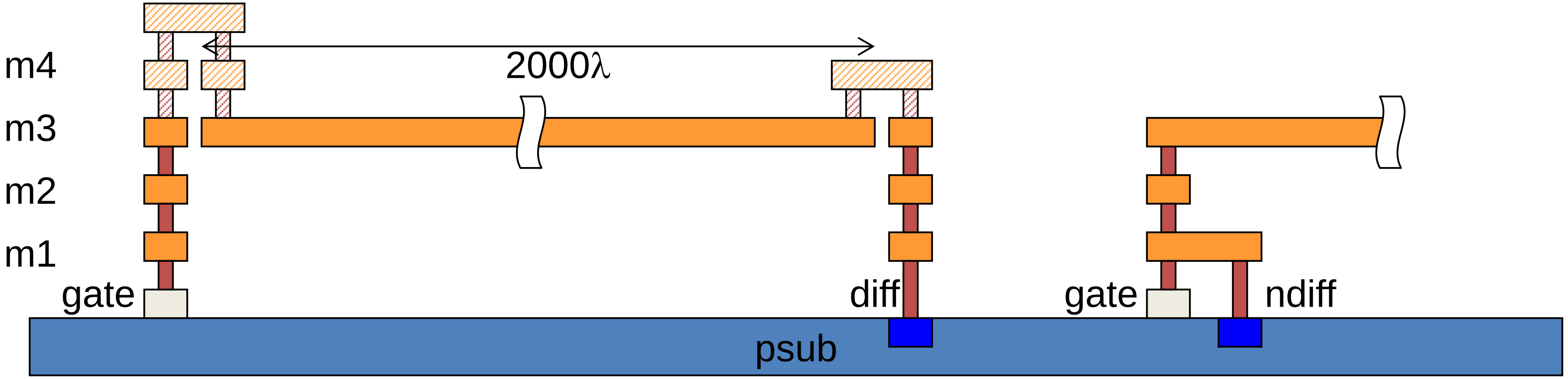
Fixing Antenna Rules

Gate can be destroyed during manufacturing if metal too long, gate too small, and/or no diffusion



How much metal, gate, and diffusion area in each standard-cell?

Antenna section in .lef



Bridging keeps gate away from long metals

Add diffusions, charge leaks away harmlessly

Antenna Information in LEF Files

- **Antenna information must be provided when the active layers are invisible**
  - LEF provides antenna information for every pin
  - Antenna information is often stored in a separate LEF file

```
MACRO AN2
  PIN I1
   AntennaPartialMetalArea          0.201400 LAYER metal1 ;
   AntennaGateArea                  0.167400 LAYER metal1 ;
   AntennaMaxAreaCAR                4.274794 LAYER metal1 ;
  END I1

  PIN I2
   AntennaPartialMetalArea          0.195200 LAYER metal1 ;
   AntennaGateArea                  0.169200 LAYER metal1 ;
   AntennaMaxAreaCAR                2.615836 LAYER metal1 ;
  END I2

  PIN O
   AntennaPartialMetalArea          0.617600 LAYER metal1 ;
   AntennaDiffArea                  1.092700 LAYER metal1 ;
  END O

END AN2
```