Agenda

DAY 1

- i — Introduction & Overview
- 1 — Data Setup & Basic Flow
- 2 — Design Planning

Unit Objectives

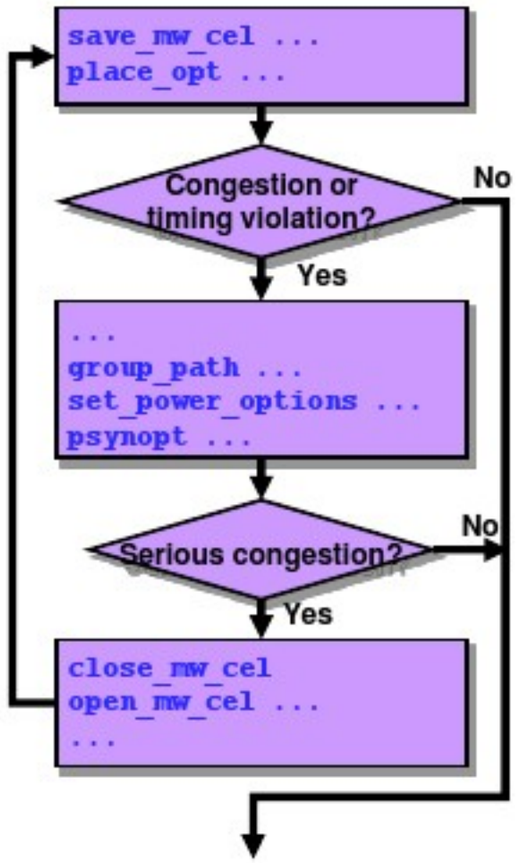
After completing this unit, you should be able to:

- **Perform data setup to create an initial *design cell* which is ready for design planning:**
  - Load necessary synthesis data: logical libraries, constraints, netlist
  - Load necessary physical design data: physical libraries, technology file, RC parasitic model files
  - Create a Milkyway *design library* and initial *design cell*
  - Apply timing and optimization controls
  - Perform checks on libraries, RC parasitic models, constraints and timing
- **Execute a basic flow which includes loading a floorplan and performing placement, CTS and routing**

A Word of Caution About Scripts and Flows

This workshop contains many scripts and flow diagrams showing specific commands executed in a specific order

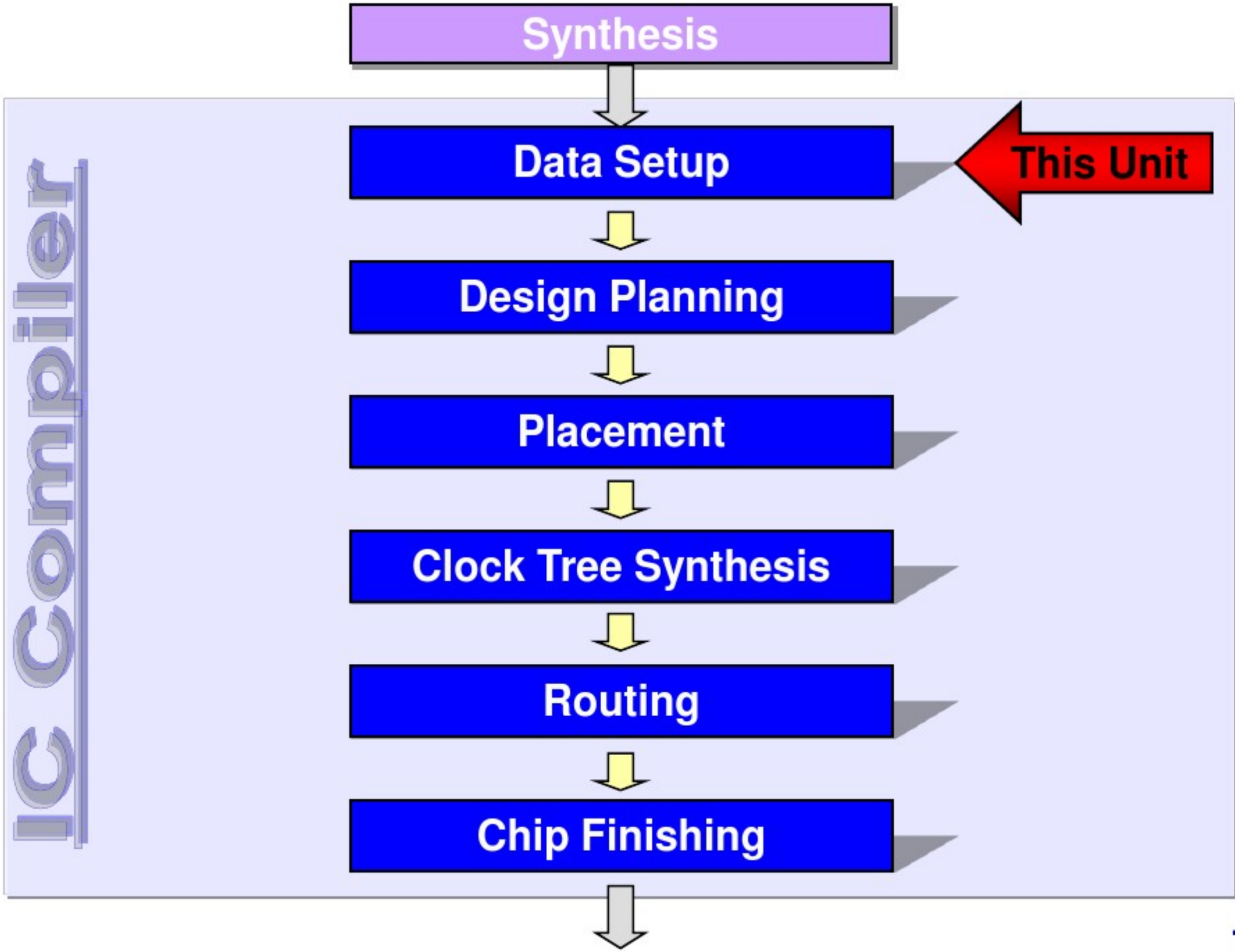
- **These flows DO NOT represent "the recommended flow"**
  - Each flow is just one example of many possible flows
  - They help to better organize and present the material



- **The specific commands and order of execution required to achieve the best results is completely design dependent**

There is no "golden script" for physical design

General IC Compiler Flow



Data Setup

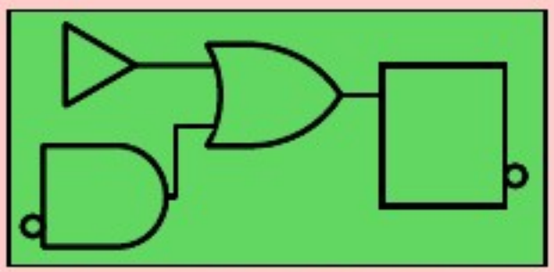
db Standard cells

db Macro cells

db Pad cells

Logic/timing Library Files

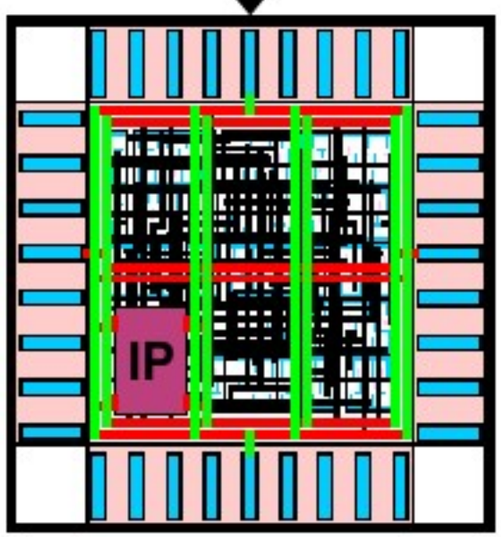
Constraints File
orca.sdc



Gate-Level Netlist *orca.v*

Synthesis Data

IC Compiler



Placed, Routed & Optimized Layout with Clock Trees

sc

macros

io

MW Standard cells

MW Macro cells

MW Pad cells

Physical Library Directories

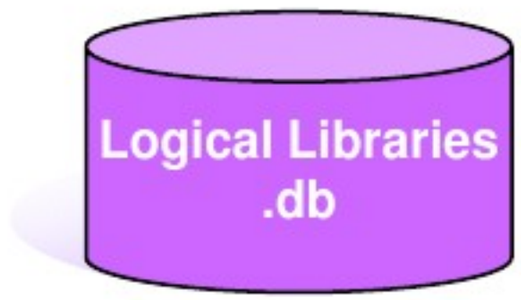
Technology File
abc_6m.tf

RC Model Files
(*TLU+*)

Physical Data

Logical Libraries

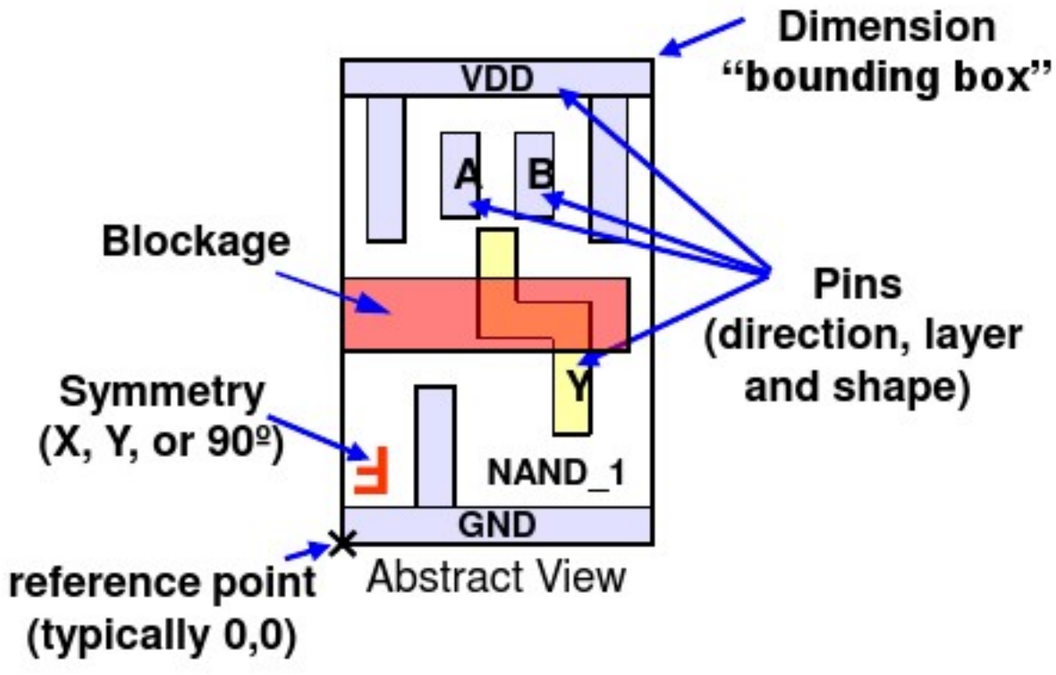
- **Provide timing and functionality information for all standard cells (`and`, `or`, `flipflop`, ...)**
- **Provide timing information for hard macros (IP, ROM, RAM, ...)**
- **Define drive/load design rules:**
  - Max fanout
  - Max transition
  - Max/Min capacitance



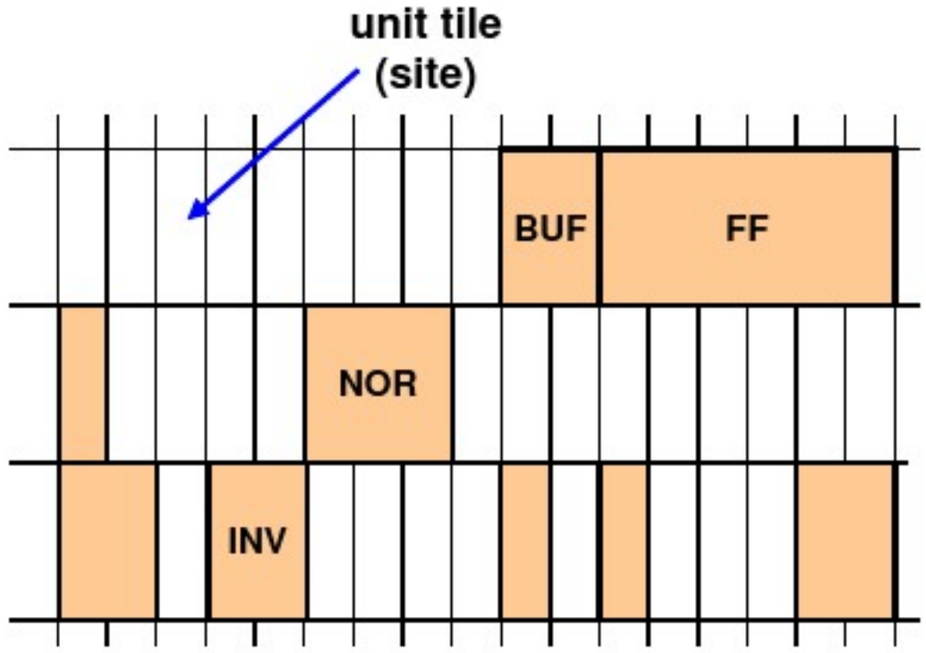
- **Are usually the same ones used by Design Compiler during synthesis**
- **Are specified with variables:**
  - `target_library`
  - `link_library`

Physical *Reference Libraries*

- **Contain physical information of *standard*, *macro* and *pad* cells, necessary for placement and routing**



- **Define placement *unit tile***
  - Height of placement rows
  - Minimum width resolution
  - Preferred routing directions
  - Pitch of routing tracks
  - ...

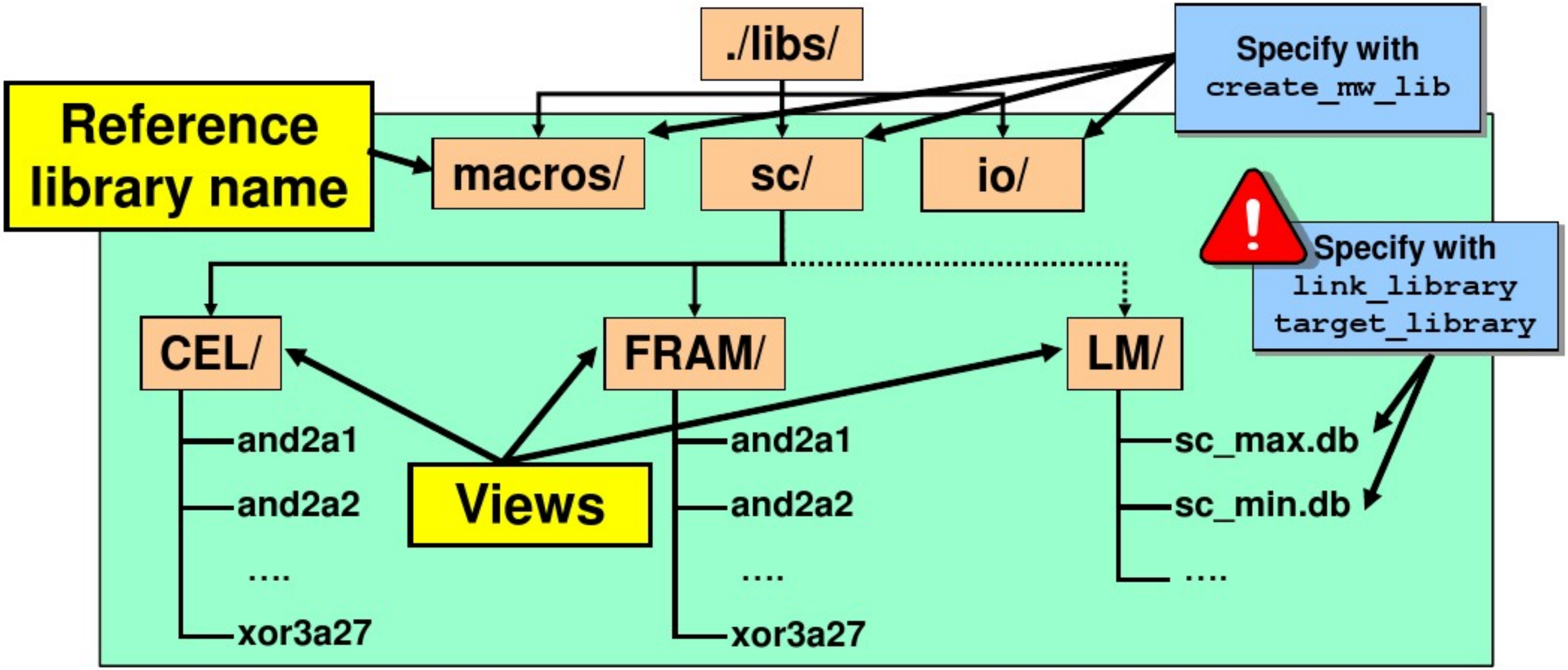


- **Are specified with the command:**
  - `create_mw_lib -mw_reference_library ...`

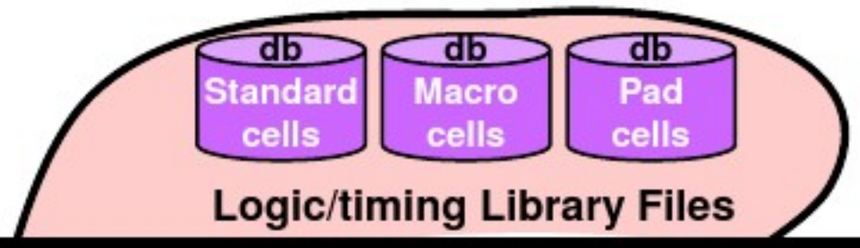
Milkyway Structure of Physical Libraries

Each physical or reference library is a UNIX directory under which information is stored in sub-directories called *views*

- FRAM: Abstract view - Used during P&R
- LM: (Optional) Logic model view - Contains *db* logical libraries[1]



1. Specify the Logical Libraries



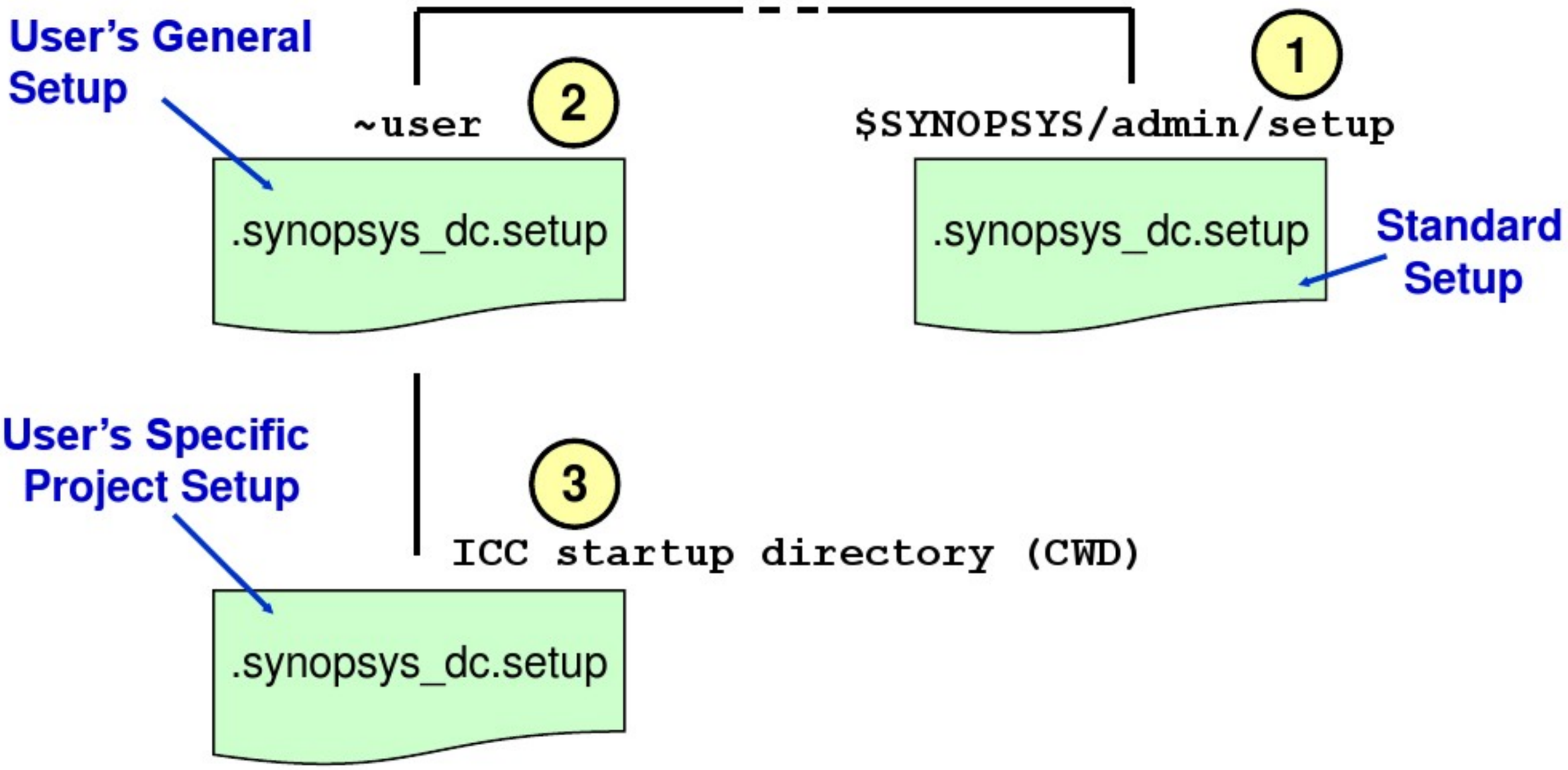
.synopsys_dc.setup

```
lappend search_path [glob ./libs/*/LM]
set_app_var target_library "sc_max.db"
set_app_var link_library "* sc_max.db io_max.db \
                                        macros_max.db"
set_min_library sc_max.db -min_version sc_min.db
set_min_library io_max.db -min_version io_min.db
set_min_library macros_max.db -min_version macros_min.db
set_app_var symbol_library "sc.sdb io.sdb macros.sdb"
```

These settings can be re-applied in each new IC Compiler session, or more conveniently, entered once in the `.synopsys_dc.setup` file, which is automatically read by the tool when ICC is invoked

TCL: `glob` returns files/directories that match the specified pattern

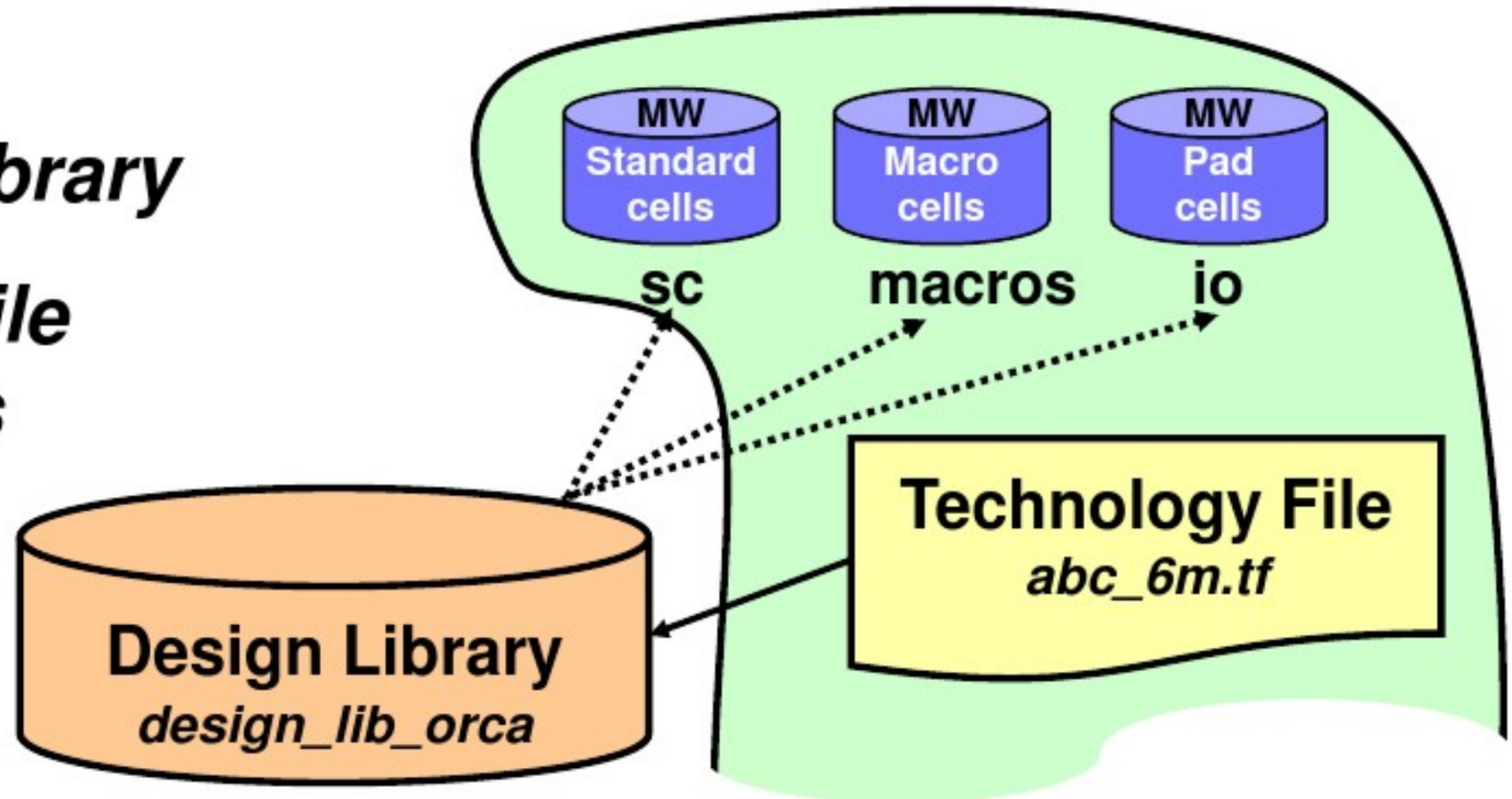
IC Compiler Initialization Files



Commands in `.synopsys_dc.setup` are executed upon tool startup, in the order shown above.

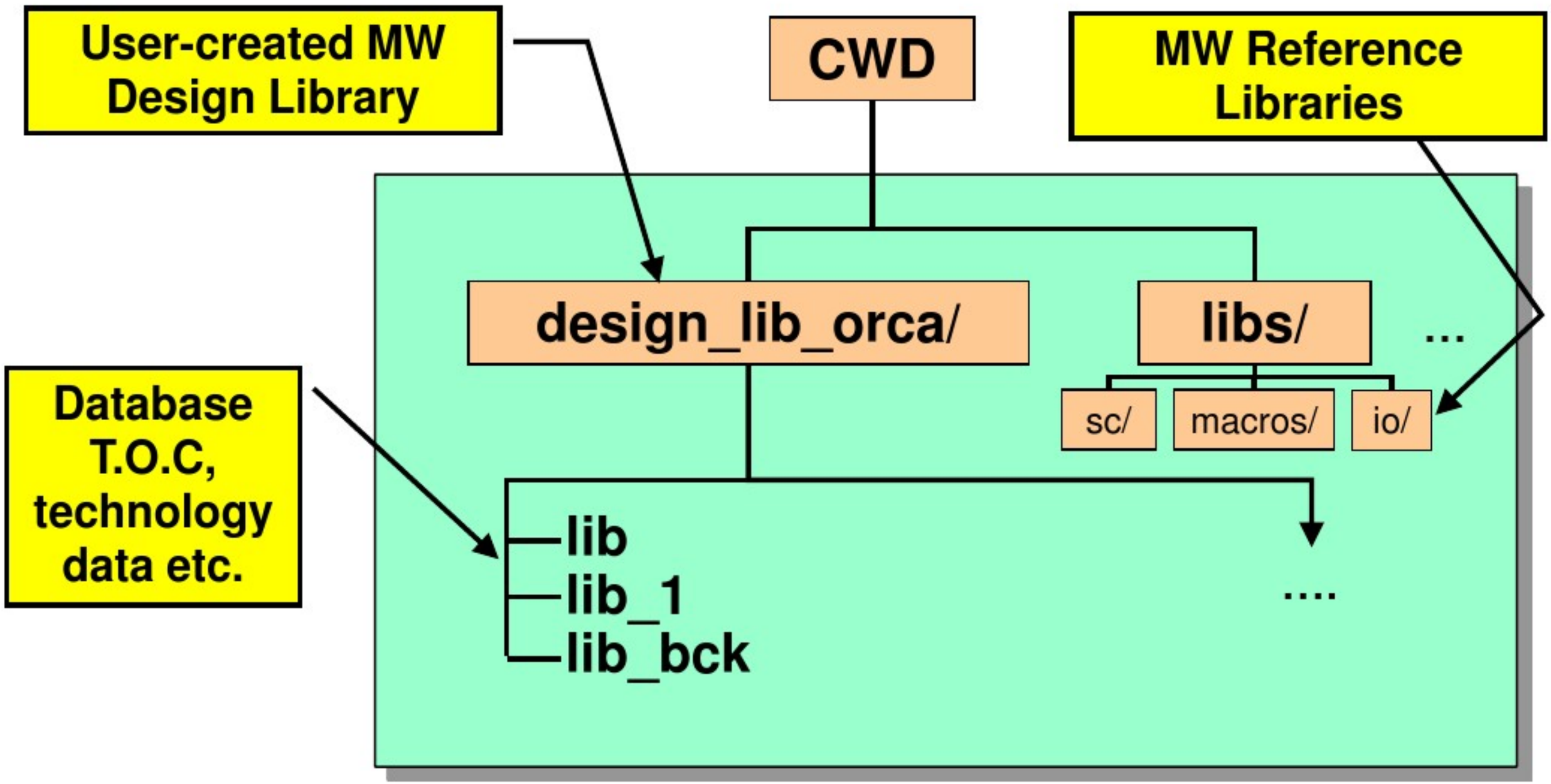
2. Create a "Container": The *Design Library*

- Create a *design library*
- Specify the *tech file* and *reference libs*



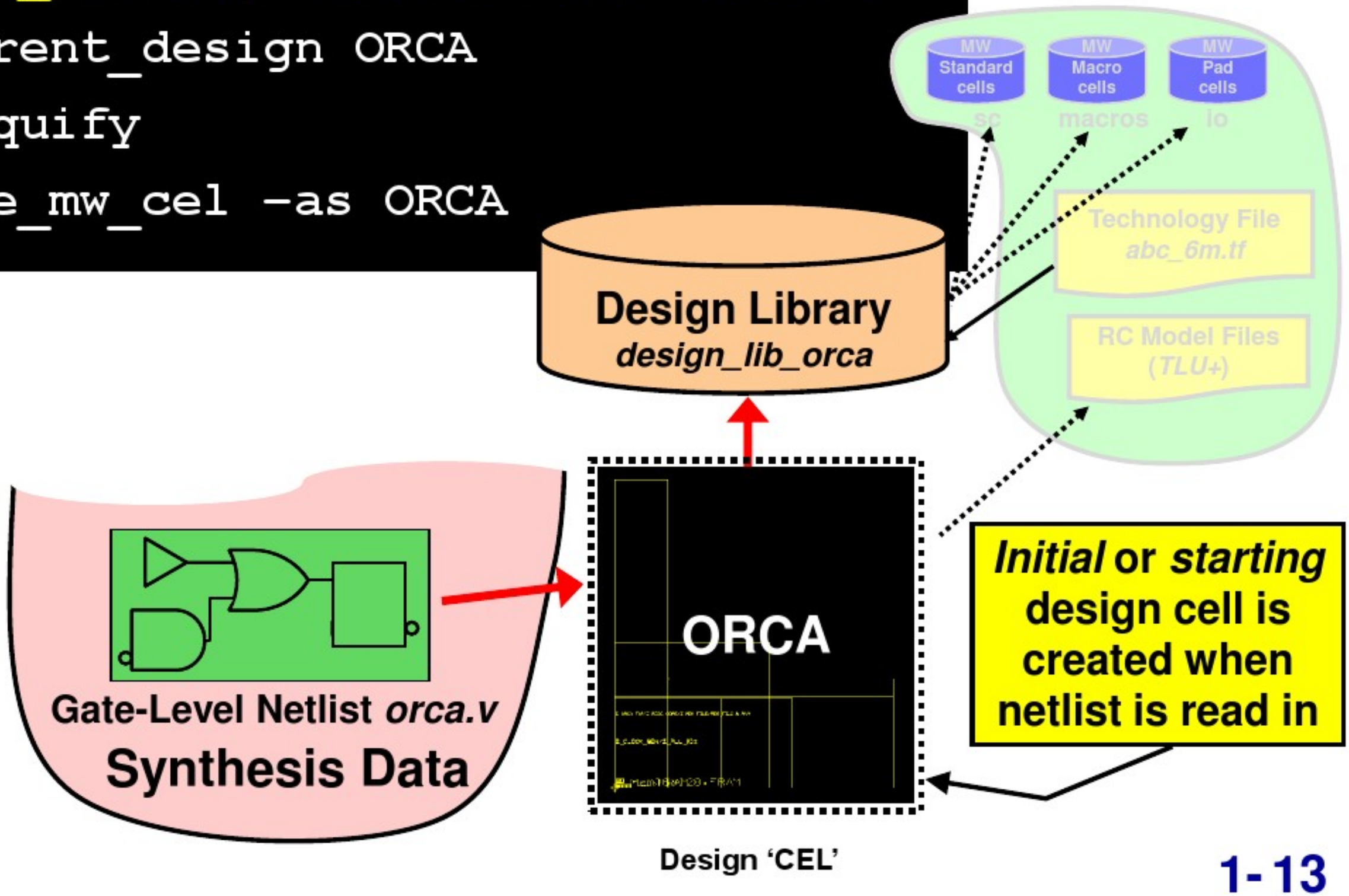

```
create_mw_lib  design_lib_orca -open \
      -technology ./libs/abc_6m.tf     \
      -mw_reference_library            \
"./libs/sc  ./libs/macros ./libs/io"
```

Initial Structure of a *Milkyway* Design Library



3a. Read the Netlist and Create a Design *CEL*

```
read_verilog -netlist ./rtl/orca.v
current_design ORCA
uniquify
save_mw_cel -as ORCA
```

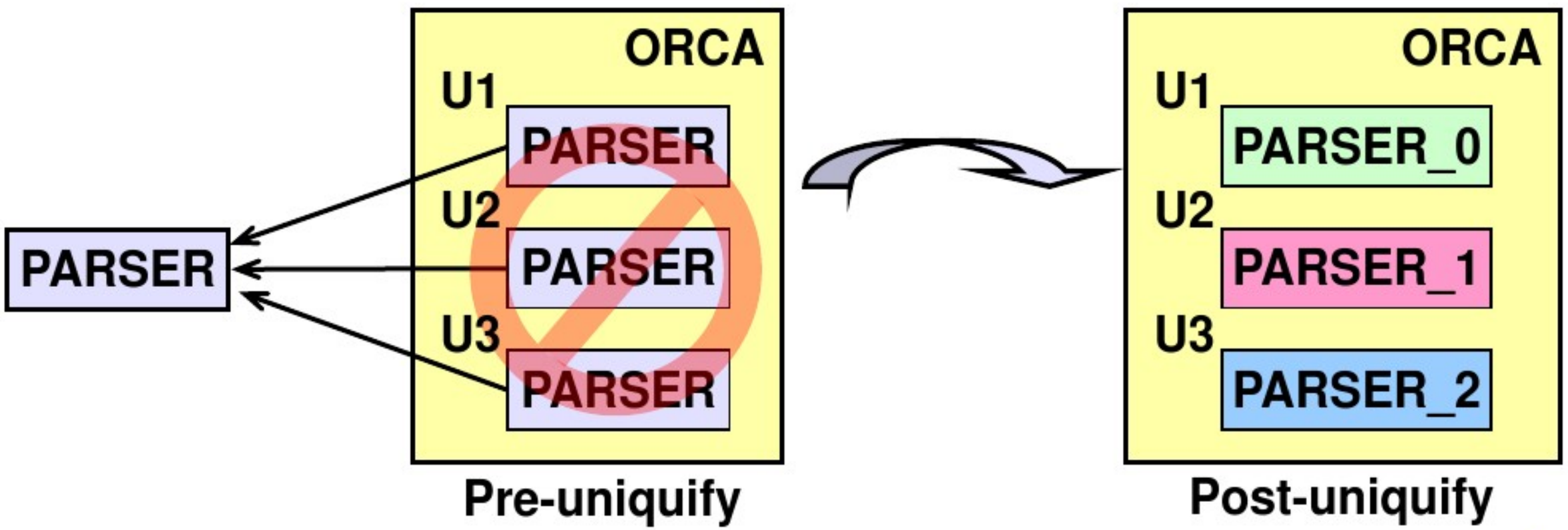


Design 'CEL'

Must *Uniquify* Multiply Instantiated Designs

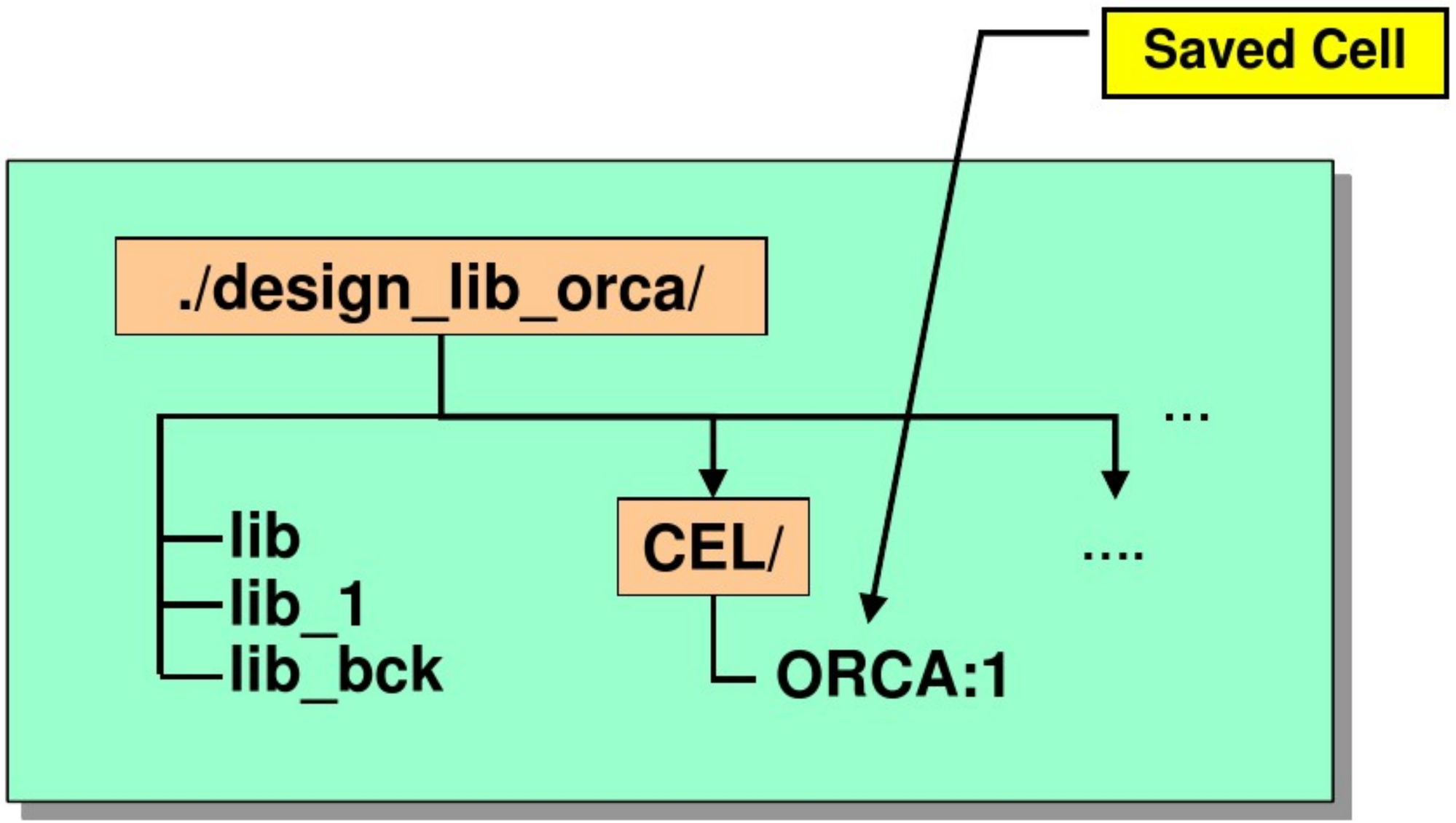
- IC Compiler does not support non-uniquified designs, i.e. designs with multiple instantiations!
- If incoming netlist is not *uniquified*, do so first!

```
current_design ORCA
uniquify
```



Milkyway Design Library with Design Cell

The `save_mw_cel` command creates a new *CEL* view



3b. Shortcut: Import the Netlist

```
import_designs ./rtl/orca.v
      -format verilog  \
      -top ORCA
```

Format can be *verilog, db, ddc*

Replaces:

```
read_verilog -netlist ./rtl/orca.v
current_design ORCA
uniquify
save_mw_cel -as ORCA
```

The Technology File (.tf file)

- **The *technology file* is unique to each technology**
- **Contains metal layer technology parameters:**
  - Number and name designations for each layer/via
  - Physical and electrical characteristics of each layer/via
  - Design rules for each layer/Via (Minimum wire widths and wire-to-wire spacing, etc.)
  - Units and precision for electrical units
  - Colors and patterns of layers for display
  - …

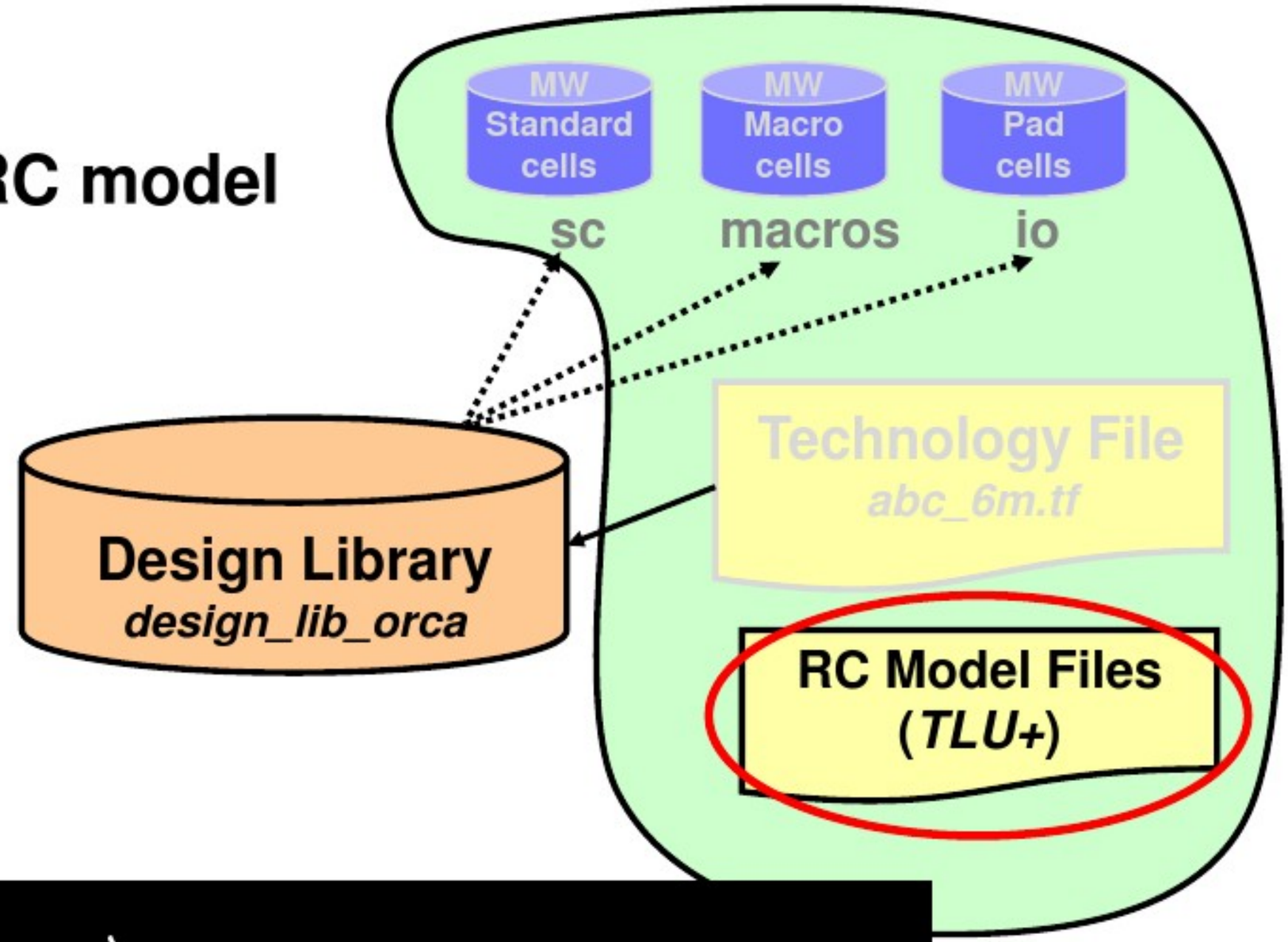
Example of a Technology File

abc_6m.tf

```
Technology  {
          unitTimeName                = "ns"
          timePrecision               = 1000
          unitLengthName              = "micron"
          lengthPrecision             = 1000
          gridResolution              = 5
          unitVoltageName             = "v"
          }
...
Layer   "m1" {
           layerNumber                = 16
           maskName                   = "metal1"
           pitch                      = 0.56
           defaultWidth               = 0.23
           minWidth                   = 0.23
           minSpacing                 = 0.23
           ...
```

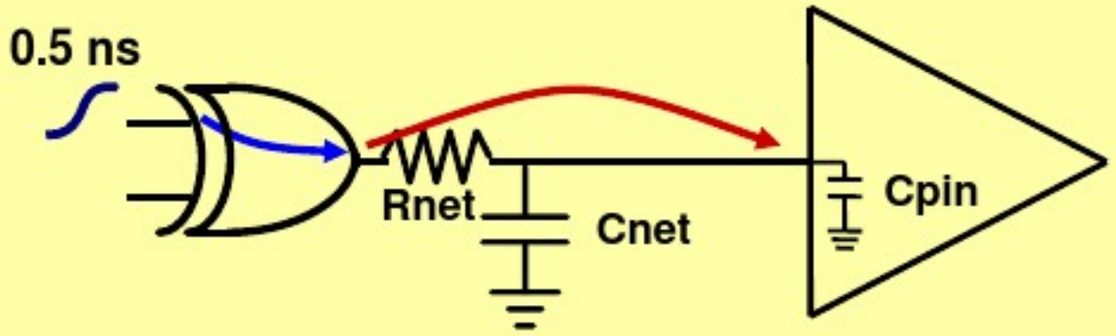
4. Specify *TLU+* Parasitic RC Model Files

- **Specify the *TLU+* RC model files to be used**
- **Must be re-applied after re-loading a saved *un-placed* design in a new IC Compiler session**




```
set_tlu_plus_files \
      -max_tluplus ./libs/abc_max.tlup \
      -min_tluplus ./libs/abc_min.tlup \
      -tech2itf_map  ./libs/abc.map
```

Timing is Based on Cell and Net Delays



Cell Delay = f(Input Transition Time, $C_{net} + C_{pin}$)

Net Delay = f(R_{net}, $C_{net} + C_{pin}$)

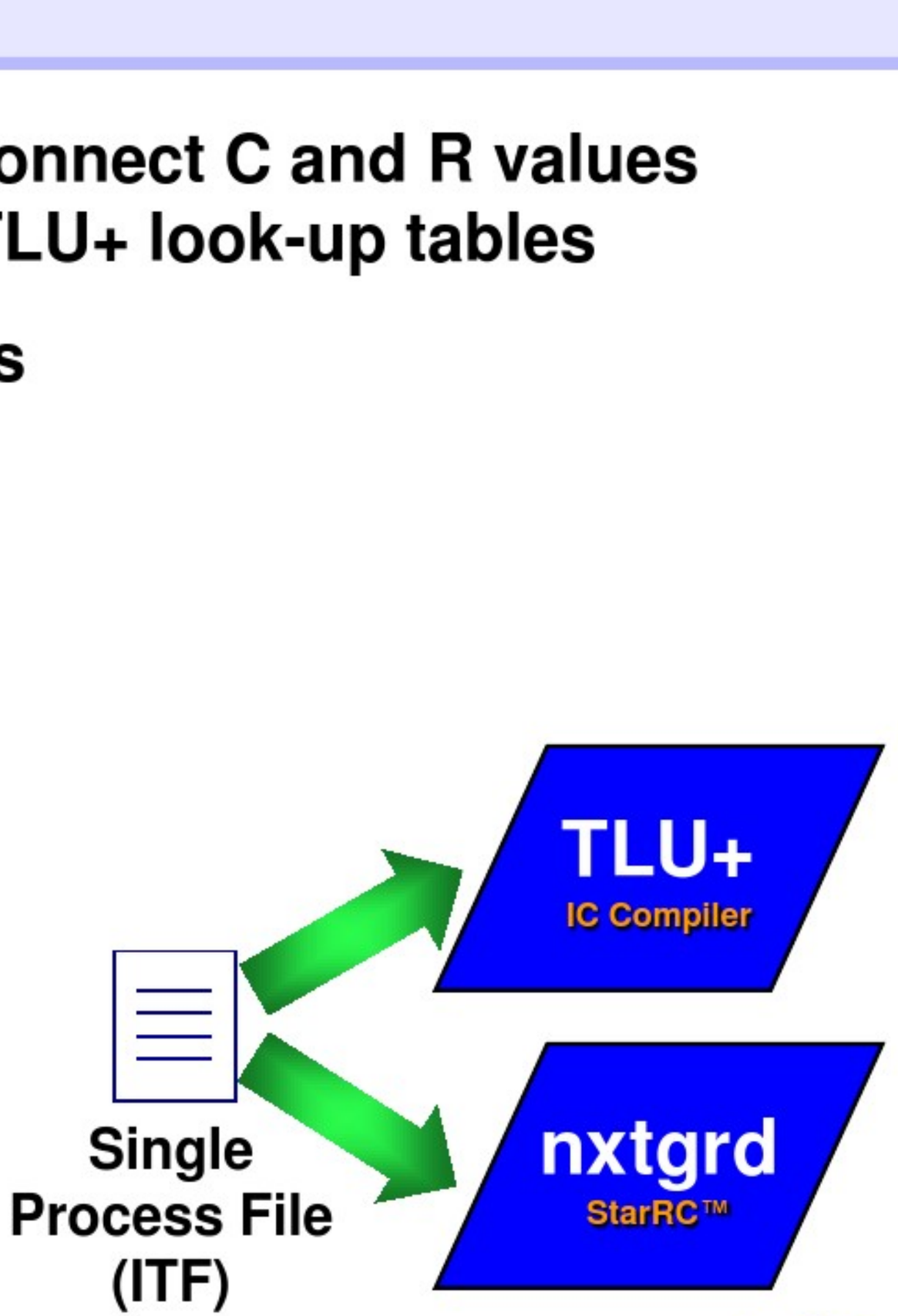
- ICC calculates delay for every cell and every net
- To calculate delays, ICC needs to know each net's parasitic Rs and Cs

TLU+ Models

- IC Compiler calculates interconnect C and R values using net geometry and the TLU+ look-up tables
- Models UDSM process effects

UDSM Process Effects
- Conformal Dielectric
- Metal Fill
- Shallow Trench Isolation
- Copper Dishing:
  - Density Analysis
  - Width/Spacing
- Trapezoid Conductor

- Some vendors provide only an *ITF* process file
- User must then generate *TLU+* from *ITF* (see below)



Mapping file

The *Mapping File* maps the technology file (.tf) layer/via names to StarRC (.itf) layer/via names.

abc.tf

```
Layer  "METAL" {
                layerNumber = 14
                maskName = "metal1"
                …
```

abc.itf

```
DIELECTRIC cm_extra3 { THICKNESS=0.06 ER=4.2 }
CONDUCTOR cm { THICKNESS=0.26 WMIN=0.16 …}
DIELECTRIC diel1d { THICKNESS=0.435 ER=4.2 }
…
```

abc.map

```
conducting_layers
  poly                poly
  metal1              cm
  metal2              cm2

  …
```

5a. Check the Libraries

- **The `check_library` command reports library inconsistencies, for example:**
  - Between logic (`link_library`) and physical libraries:
    - Missing cells
    - Missing or mismatched pins
  - Within physical libraries:
    - Missing CEL (layout) or FRAM (abstract) view cells
    - Duplicate cell name in multiple reference libraries
- **The `check_tlu_plus_files` command performs a sanity check on the TLU+ files and settings**

```
set_check_library_options -all
check_library
check_tlu_plus_files
```

5b. Verify Logical Libraries Are Loaded

Ensure that all the required logical libraries (specified by `set_app_var link_library`) have been loaded

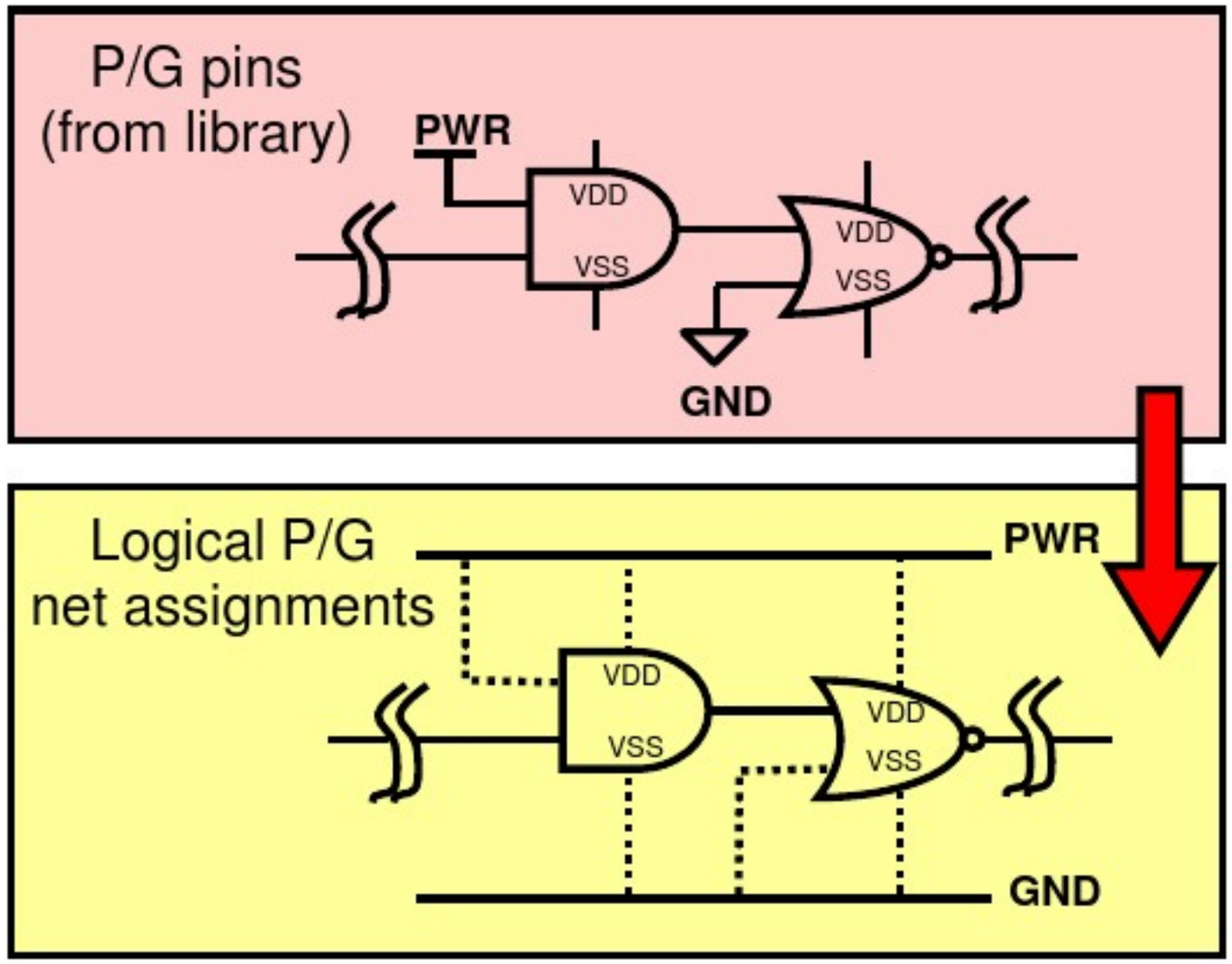
```
list_libs
```

```
icc_shell> list_libs
Logical Libraries:
--------------------------------------------------------------------------
Library          File              Path
-------          ----              ----
M std_cells_max  sc_max.db         /projects/XYZ_design/libs/sc/LM
m std_cells_min  sc_min.db         /projects/XYZ_design/libs/sc/LM
M io_pads_max    io_max.db         /projects/XYZ_design/libs/io/LM
m io_pads_min    io_min.db         /projects/XYZ_design/libs/io/LM
M macros_max     macros_max.db     /projects/XYZ_design/libs/macros/LM
m macros_min     macros_max.db     /projects/XYZ_design/libs/macros/LM
  gtech          gtech.db          /global/apps3/icc_2010.03-SP2/libraries/syn
  standard.sldb  standard.sldb     /global/apps3/icc_2010.03-SP2/libraries/syn
```

The *gtech* and *standard* libraries are generic libraries that are loaded by default – used during synthesis

6. Define Logical Power/Ground Connections

- **Define P/G net names and create "logical connections" between P/G pins and P/G nets**
- **Create connections between tie-high/low inputs and P/G nets**




```
derive_pg_connection -power_net PWR -power_pin VDD \
                     -ground_net GND -ground_pin VSS
derive_pg_connection -power_net PWR -ground_net GND \
                     -tie
check_mv_design -power_nets
```

`derive_pg_connection` will need to be re-applied during the physical design flow – see guidelines below

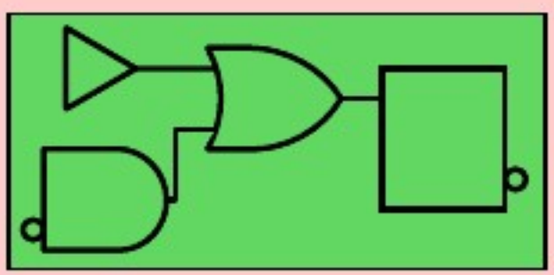
7. Apply and Check Timing Constraints

```
read_sdc ./cons/orca.sdc
check_timing
report_timing_requirements
report_disable_timing
report_case_analysis
```

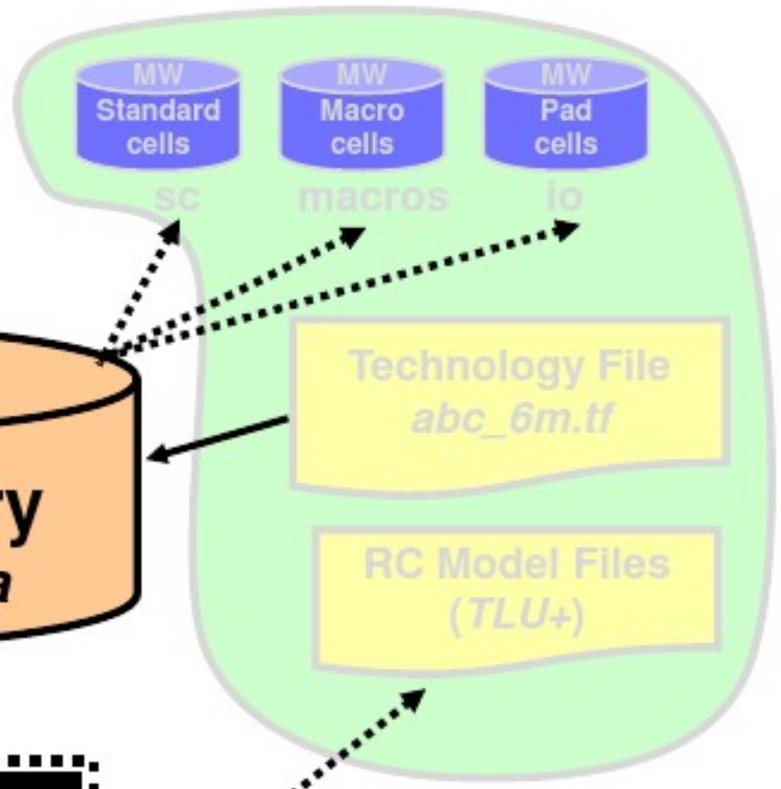
Only needed if reading in a *ASCII* netlist

Constraints File ***orca.sdc***

```
create_clock -period 10 ……
set_input_delay -max 1.2 ….
set_output_delay -max 2.5 ….
```



Gate-Level Netlist ***orca.v***



Design Library
design_lib_orca

Design 'CEL'

Timing Constraints

- **"Timing Constraints" are required to communicate the design's timing intentions to IC Compiler**
- **Already included if netlist is in *ddc* or *db* format**
- **If the netlist is *Verilog* must read in the constraint file**
  - Should be the same constraints used for synthesis with Design Compiler (preferably in SDC format)

```
create_clock -period 10 [get_ports clk]
set_input_delay 4 -clock clk \
        [get_ports sd_DQ[1]]
set_output_delay 5 -clock clk
        [get_ports sd_LD]
set_load 0.2 [get_ports DOUT]
set_driving_cell -lib_cell buf5 \
        [get_ports pdevsel_n]
...
```

SDC = Synopsys Design Constraints

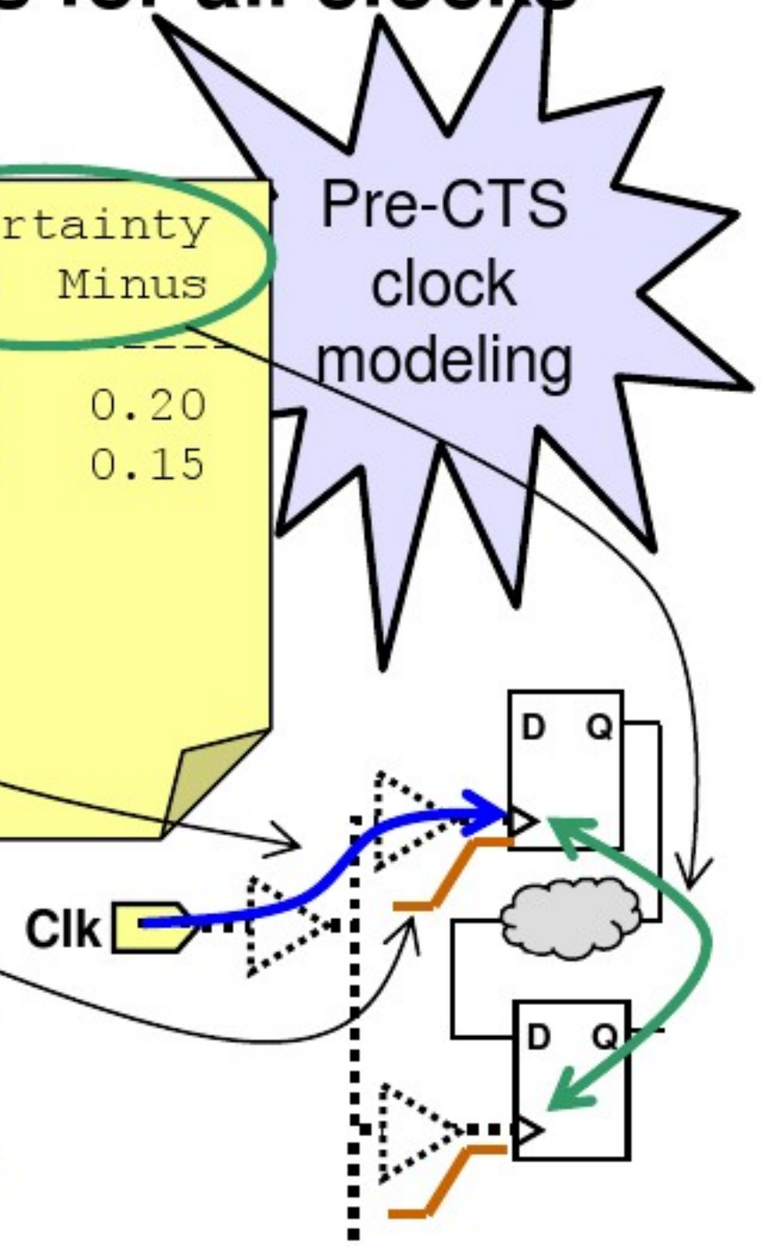
8. Ensure Proper Modeling of Clock Tree

- **Ensure your SDC constraints to model estimates of clock skew, latency and transition times for all clocks**

`report_clock -skew`

```
                  Rise    Fall  Min Rise  Min Fall  Uncertainty
Object            Delay   Delay  Delay     Delay    Plus  Minus
-------------------------------------------------------------------
SYS_2x_CLK         0.80    0.80    0.40      0.40    0.10   0.20
SDRAM_CLK             -       -       -         -    0.10   0.15

                  Max Transition   Min Transition
Object            Rise      Fall   Rise      Fall
-------------------------------------------------------------------
SYS_2x_CLK        0.07      0.07      -         -
SDRAM_CLK         0.07      0.07      -         -
```

Pre-CTS clock modeling



- **Ensure no clocks are defined as "propagated" clocks**

`report_clock`

```
Clock         Period   Waveform   Attrs   Sources
-------------------------------------------------
SDRAM_CLK       7.50   {0 3.75}     p     {sdram_clk}
SYS_2x_CLK      4.00   {0 2}              {sys_2x_clk}
```

9. Apply Timing and Optimization Controls

- Timing and optimization in IC Compiler is controlled by many variables and commands, for example:

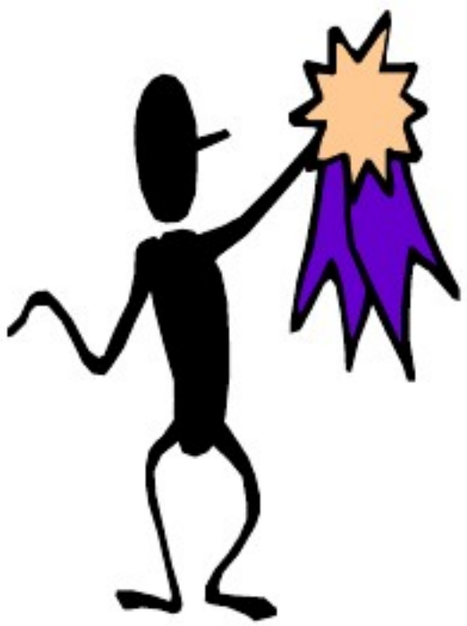
```
set_app_var timing_enable_multiple_clocks_per_reg true
set_fix_multiple_port_nets -all -buffer_constants
group_path -name INPUTS -from [all_inputs]
```

- These variables and commands can affect design planning, placement, CTS and routing
- Therefore, they should be applied prior to design planning (*and re-applied after re-starting IC Compiler*)[1]

```
source tim_opt_ctrl.tcl
```

- Learning all the available variables and commands can be a challenge – the GUI provides help!

Available Timing and Optimization Controls

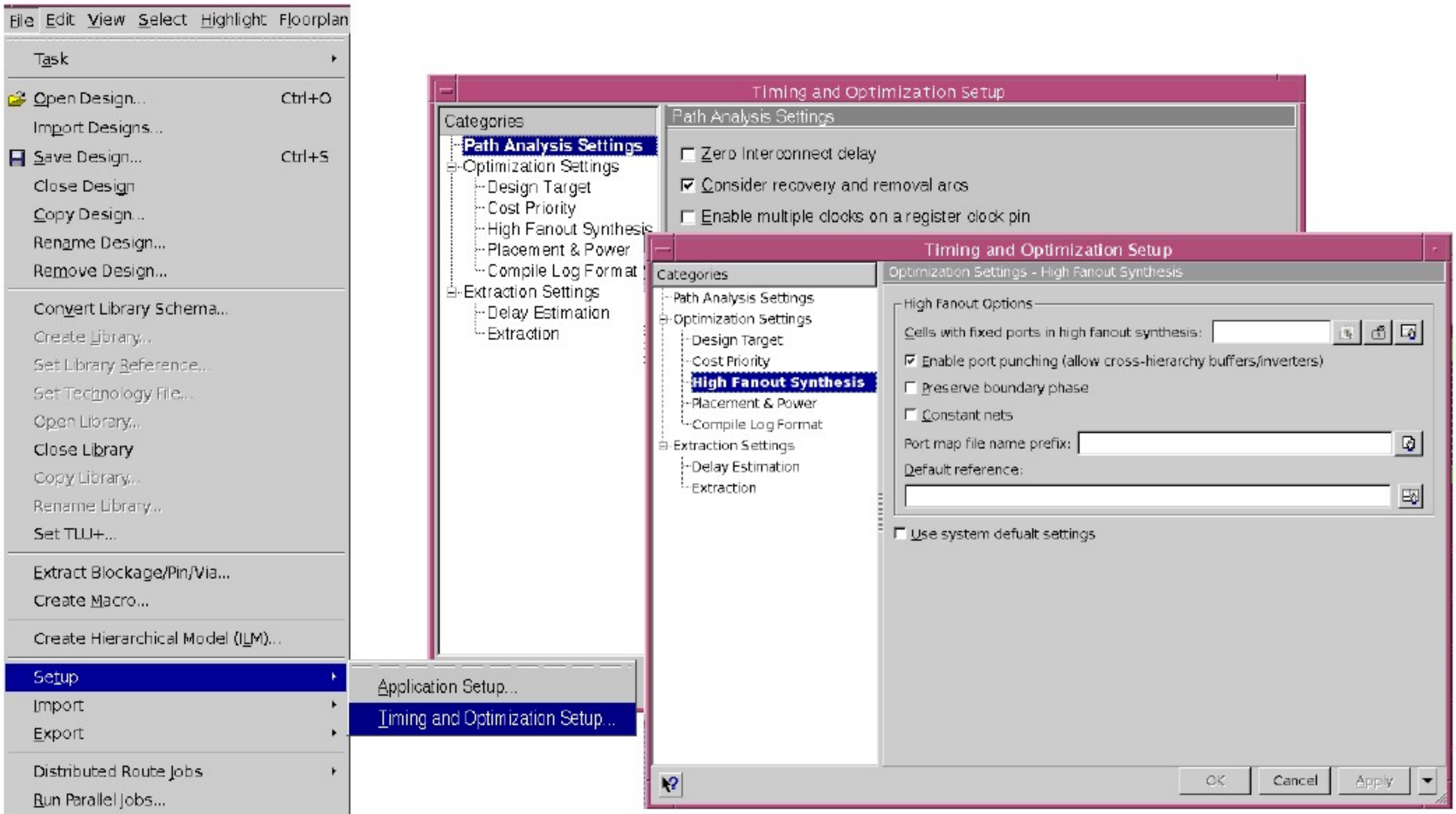


Use the GUI to find out what timing and optimization settings are available

Use the *man* pages to learn more

Use the GUI to perform your initial setup, then copy the variables/commands into a *control setup file* for subsequent uses

Timing and Optimization Setup Example



See *Appendix A* for examples of some common control settings. Others are discussed in later units.

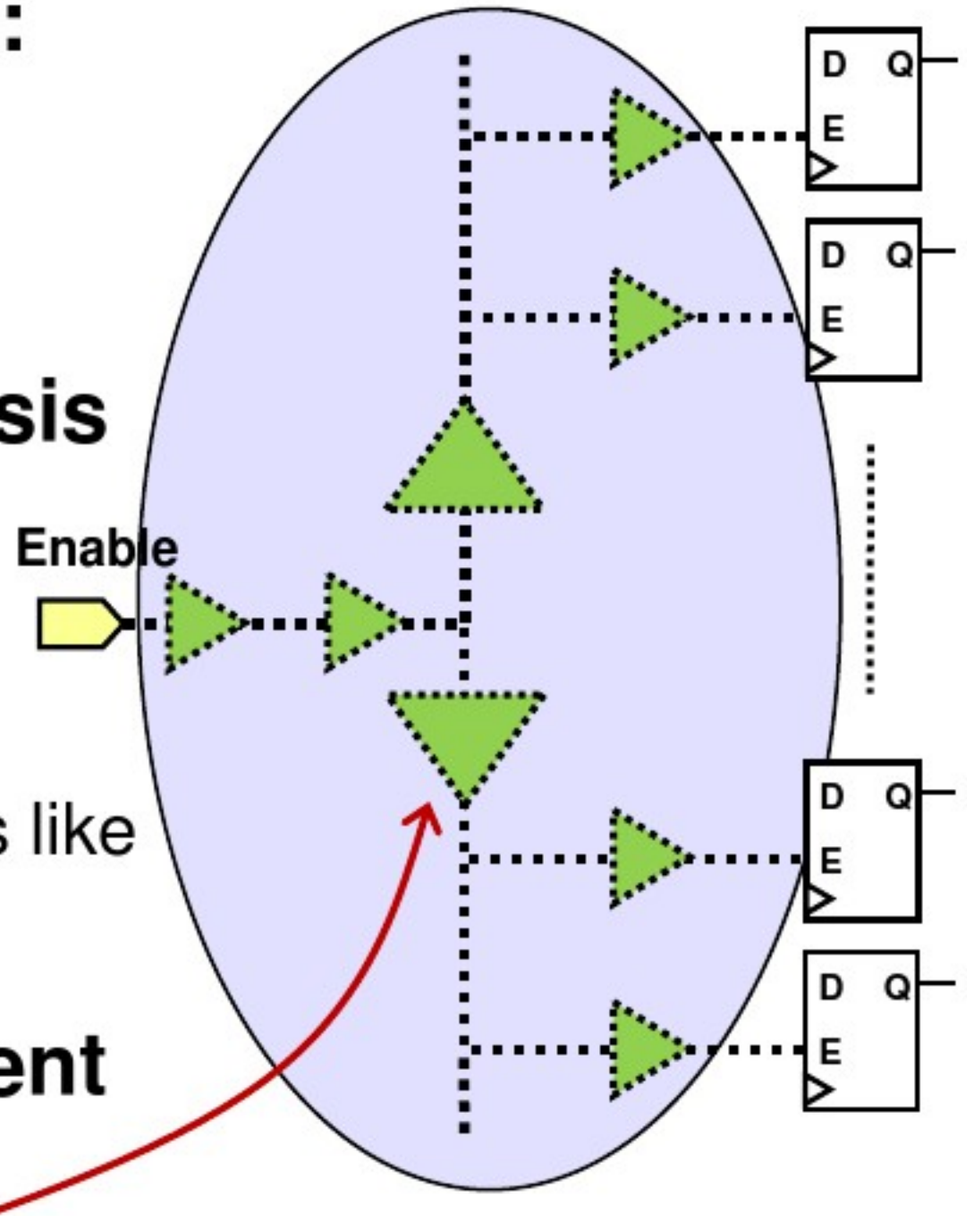
10. Perform a 'Timing Sanity Check'

- **Before starting placement it is important to ensure that the design is not over-constrained**
  - Constraints should match the design's specification
- **Report 'ZIC' timing before placement**
  - Check for unrealistic or incorrect constraints
  - Investigate large zero-interconnect timing violations

```
set_zero_interconnect_delay_mode true
Warning: Timer is in zero interconnect delay mode. (TIM-177)
report_constraint -all
report_timing
set_zero_interconnect_delay_mode false
Information: Timer is not in zero interconnect delay mode. (TIM-176)
```

11. Remove Unwanted "Ideal Net/Networks"

- **Your SDC constraints may contain either of the following commands:**
  - `set_ideal_network` **(preferred)**
  - `set_ideal_net` **(obsolete)**
- **These commands prevent synthesis (Design Compiler) from building buffer trees on specified signals, which is deferred to the physical design phase** (typically high fanout nets like *set/reset*, *enable*, *select*, etc.)



- **To allow buffering during placement remove the constraints:**

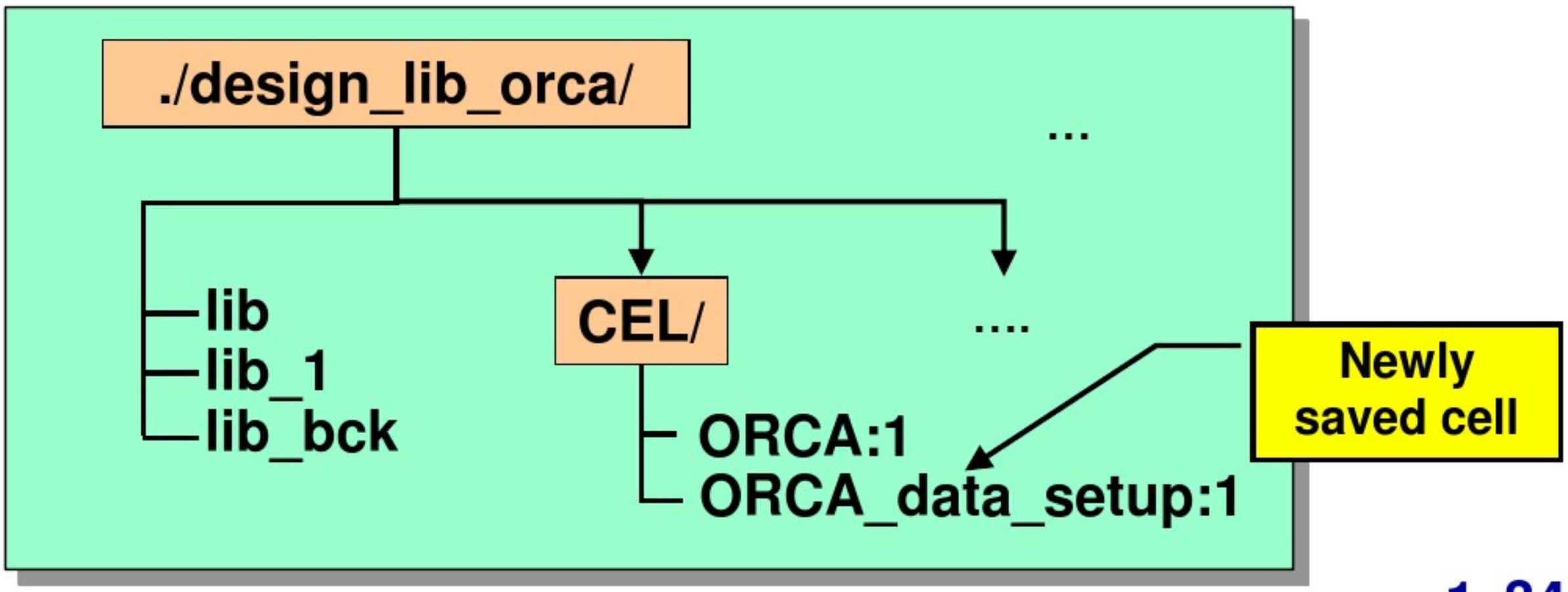
```
remove_ideal_network [get_ports "Enable Select Reset"]
```

12. Save the Design

It's good practice to save the design after each key design phase, for example: data setup, design planning, placement, CTS and routing:

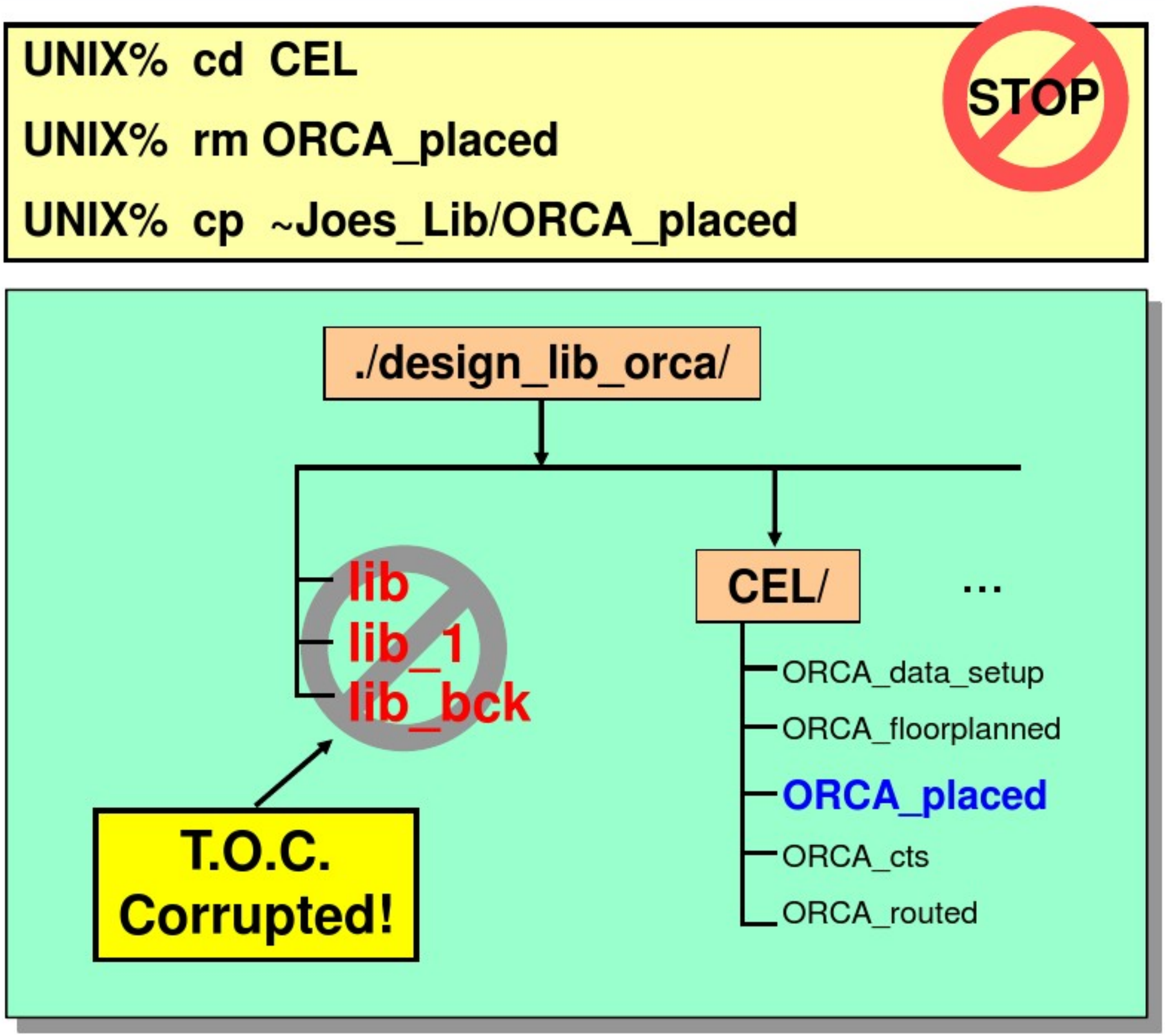
```
save_mw_cel -as ORCA_data_setup
```

- Note: The open cell is still the original *ORCA* cell !!



UNIX Manipulation of a Milkyway Database

```
UNIX%  cd  CEL
UNIX%  rm ORCA_placed
UNIX%  cp  ~Joes_Lib/ORCA_placed
```



Loading an Existing Cell After Exiting ICC

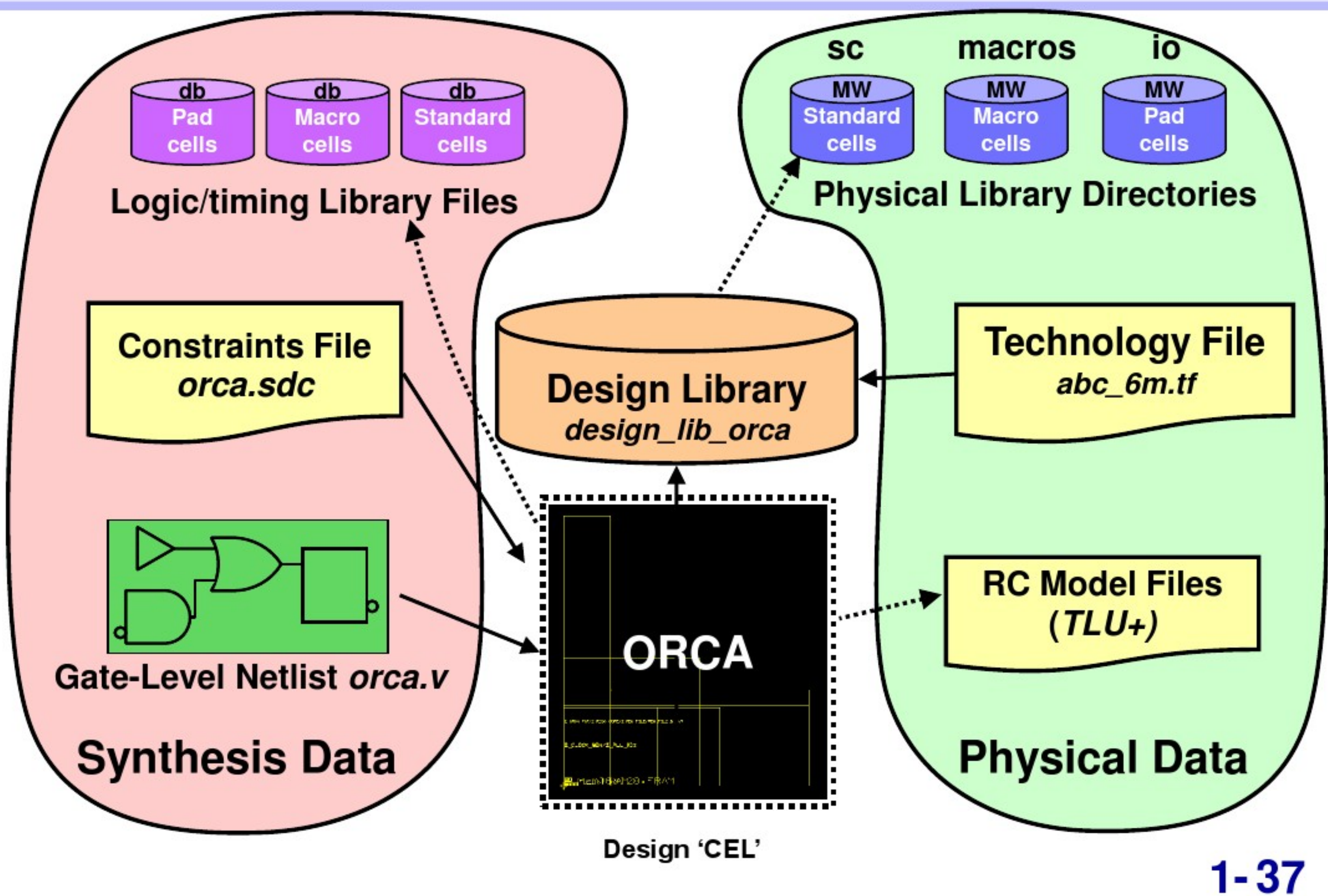
TLU+ must be re-applied if the loaded cell is un-placed

```
UNIX% icc_shell -gui

icc_shell> open_mw_lib design_lib_orca

icc_shell> open_mw_cel ORCA_setup

icc_shell> set_tlu_plus_files \
           -max_tluplus ./libs/abc_max.tlup \
           -min_tluplus ./libs/abc_min.tlup \
           -tech2itf_map ./libs/abc.map
```

Must be re-applied if the control file contains *variable* settings

```
icc_shell> source tim_opt_ctrl.tcl
```

Data Setup Summary



Example .synopsys_dc.setup

.synopsys_dc.setup

```
lappend search_path [glob ./libs/*/LM]
set_app_var target_library "sc_max.db"
set_app_var link_library \
            "* sc_max.db io_max.db macros_max.db"
set_min_library sc_max.db -min_version sc_min.db
set_min_library io_max.db -min_version io_min.db
set_min_library macros_max.db -min_version macros_min.db
set_app_var symbol_library "* sc.sdb io.sdb macros.sdb"
```

Example *Timing & Optimization Control*

tim_opt_ctrl.tcl

```
set_app_var timing_enable_multiple_clocks_per_reg true
set_false_path -from [get_clocks C1] -to [get_clocks C2]
set_false_path -from [get_clocks C2] -to [get_clocks C1]
set_app_var case_analysis_with_logic_constants true
set_fix_multiple_port_nets -all -buffer_constants
set_auto_disable_drc_nets -constant false
set_dont_use <off_limit_cells>
set_prefer -min <hold_fixing_cells>
set_app_var physopt_delete_unloaded_cells false
set_ideal_network [all_fanout -flat -clock_tree]
set_cost_priority {max_transition max_delay}
set_app_var enable_recovery_removal_arcs true
set_max_area 0
set_app_var physopt_power_critical_range <t>
set_app_var physopt_area_critical_range <t>
```

See *Appendix A*

Example *Data Setup* Run Script (1 of 2)

data_setup.tcl

```
create_mw_lib design_lib_orca -open \
    -technology libs/techfile.tf \
    -mw_reference_library "libs/sc libs/io libs/macros"
import_designs rtl/design.v -format verilog
                            -top ORCA; # See below
set_tlu_plus_files -max_tluplus libs/abc_max.tlup \
                   -min_tluplus libs/abc_min.tlup \
                   -tech2itf_map libs/abc.map
set_check_library_options -all
check_library
check_tlu_plus_files
list_libs
derive_pg_connection -power_net PWR -power_pin VDD \
                     -ground_net GND -ground_pin VSS
derive_pg_connection -power_net PWR -ground_net GND -tie
check_mv_design -power_nets
.....
```

Data Setup Run Script cont'd (2 of 2)

data_setup.tcl

```
....
# Omit if importing constrained ddc
read_sdc constraints.sdc

check_timing
report_timing_requirements
report_disable_timing
report_case_analysis

report_clock
report_clock -skew

# Timing and optimization controls
source tim_opt_ctrl.tcl

set_zero_interconnect_delay_mode true
report_constraint -all
report_timing
set_zero_interconnect_delay_mode false

remove_ideal_network [get_ports "Enable Select Reset"]
save_mw_cel -as ORCA_data_setup
```

Test for Understanding (1 of 2)

1. **Why does IC Compiler require logical libraries in addition to physical libraries?**
2. **What is the difference between a Milkyway design library and reference library? What do they have in common?**
3. **The initial or starting *design cell* is created when the netlist is imported. *True / False***
4. **When is it not necessary to read in a constraints file?**
5. **Clock uncertainty**
   a. Is used to model clock skew pre-CTS
   b. Is used during CTS as the maximum skew constraint
   c. Is used to model clock skew post-CTS
   d. A and B

Test for Understanding (2 of 2)

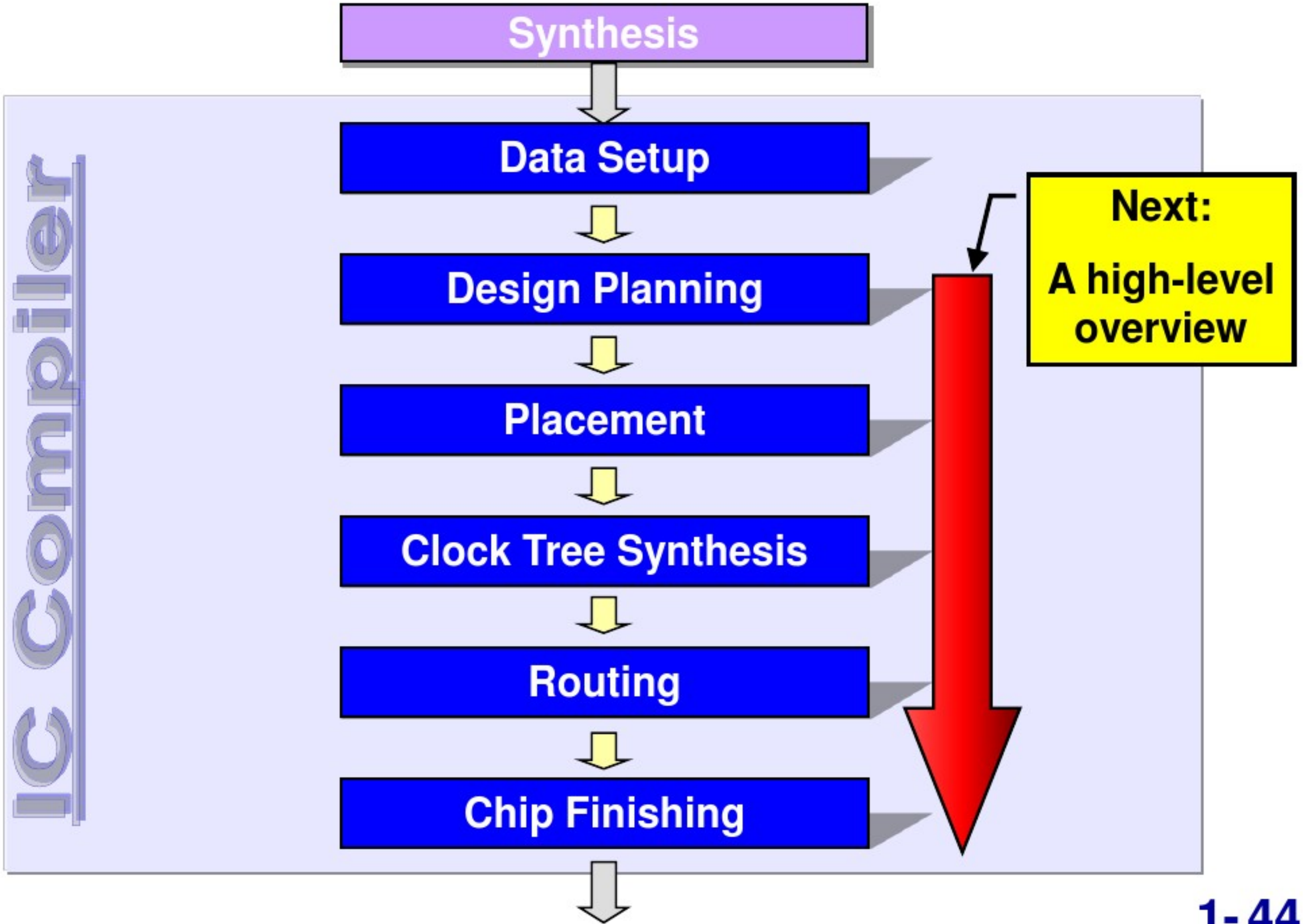
6. What is the purpose of performing a *timing sanity check* ("ZIC" timing) and what constitutes a "passing" result?

7. Why is it recommended to remove the "ideal network" property from high fanout nets like *enable* and *reset* <u>after</u> performing a "zero-interconnect" timing sanity check, and <u>not before</u>?

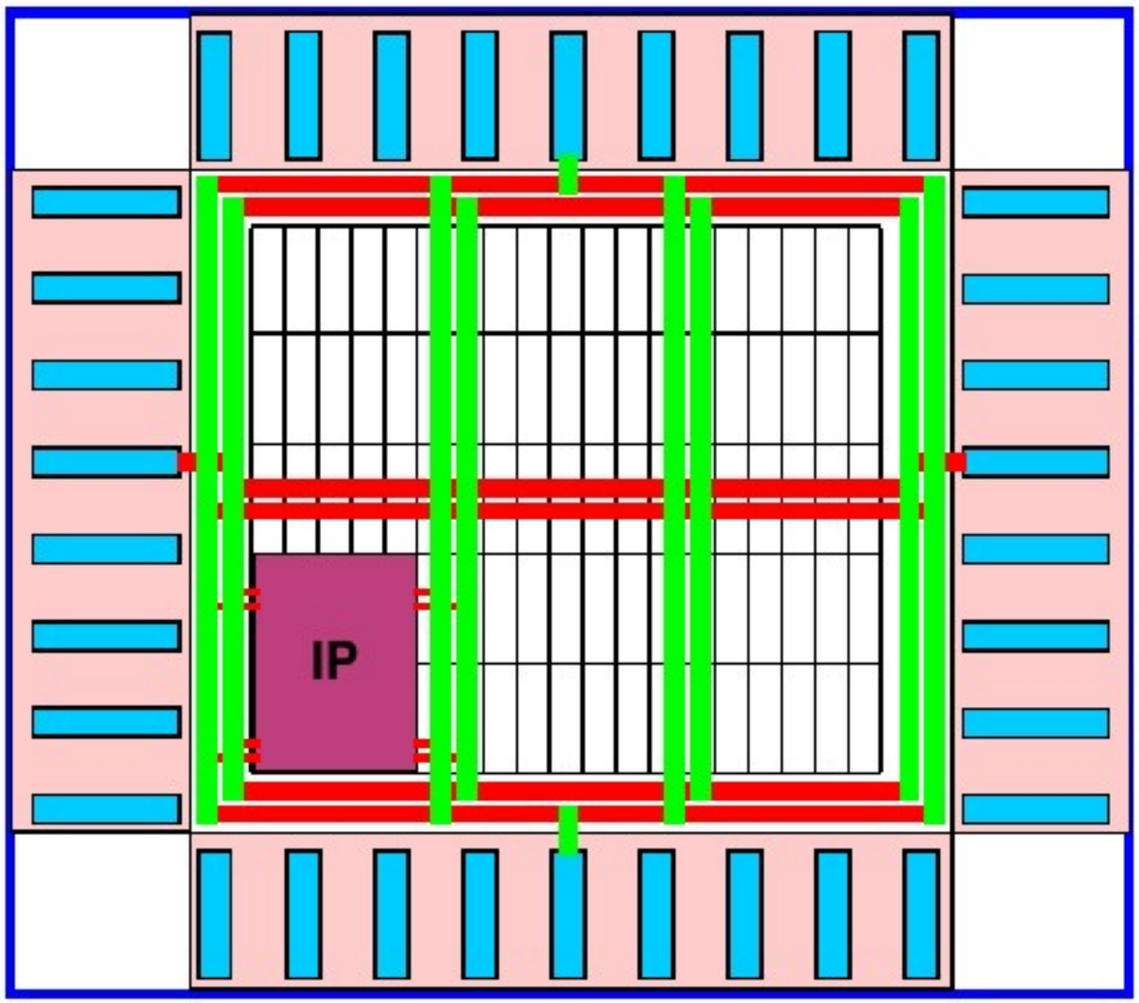
8. Is it acceptable to rename a Milkyway library with the Unix `mv` command?

9. What two key steps should you always perform after re-loading a design in a new IC Compiler session?

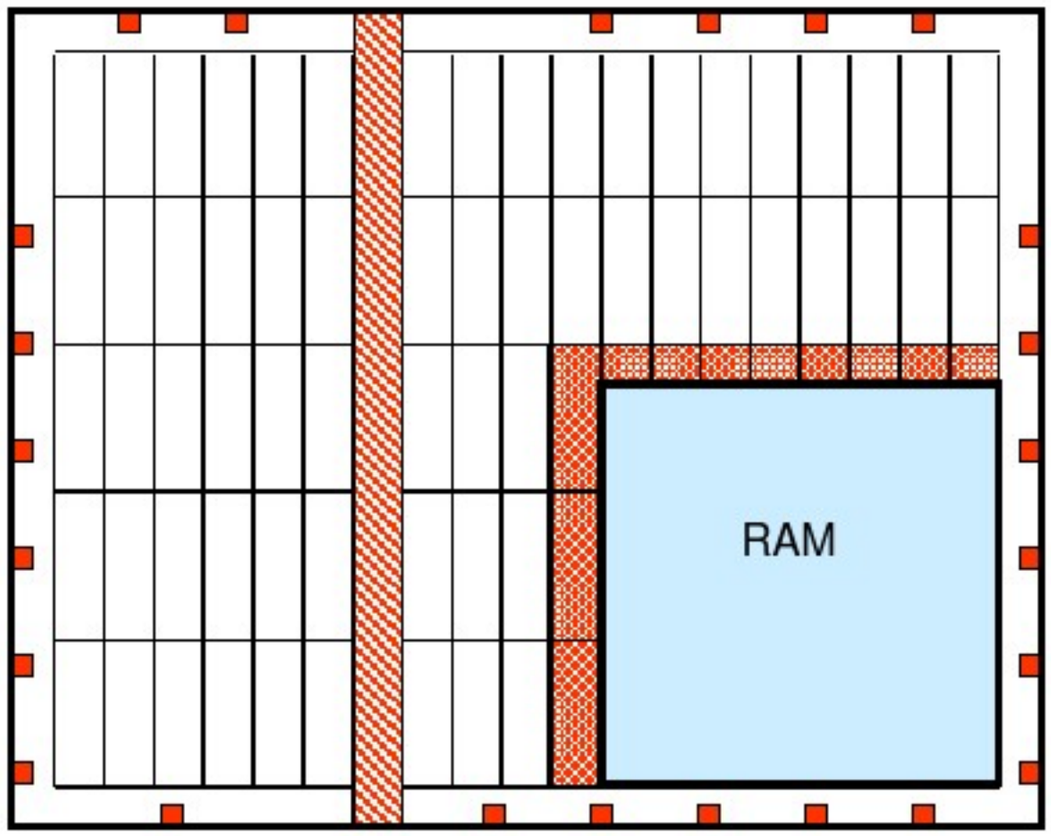
General IC Compiler Flow



Design Planning



Chip-level Floorplan



Block-level Floorplan

"Design planning" is the iterative design phase of defining a "floorplan"

Load an Existing Floorplan

- Read in as a *DEF file* from ICC or 3rd party tool

```
read_def DESIGN.def
```

- Open an already floorplanned *Milkyway cell*

```
open_mw_lib my_design_lib
open_mw_cel DESIGN_floorplanned
```

Creating a new floorplan with IC Compiler will be discussed in Unit 2

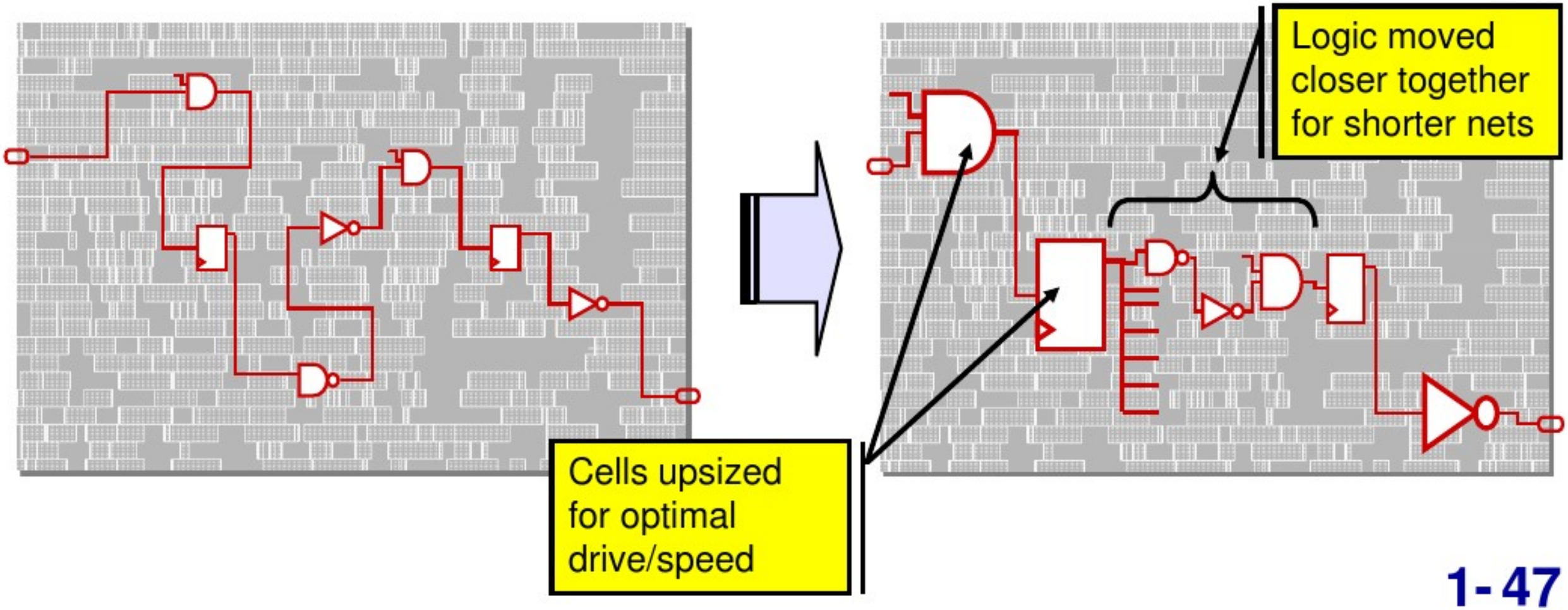
See Appendix A in Unit 2 if floorplanning with a 3rd party tool

Placement and Related Optimizations

Data Setup | Design Planning | **Placement** | Clock Tree Synthesis | Routi

1. **Set placement options**
2. **Run** `place_opt`

- Performs iterative placement and logic optimization
- Objective: Fix timing violations and congestion



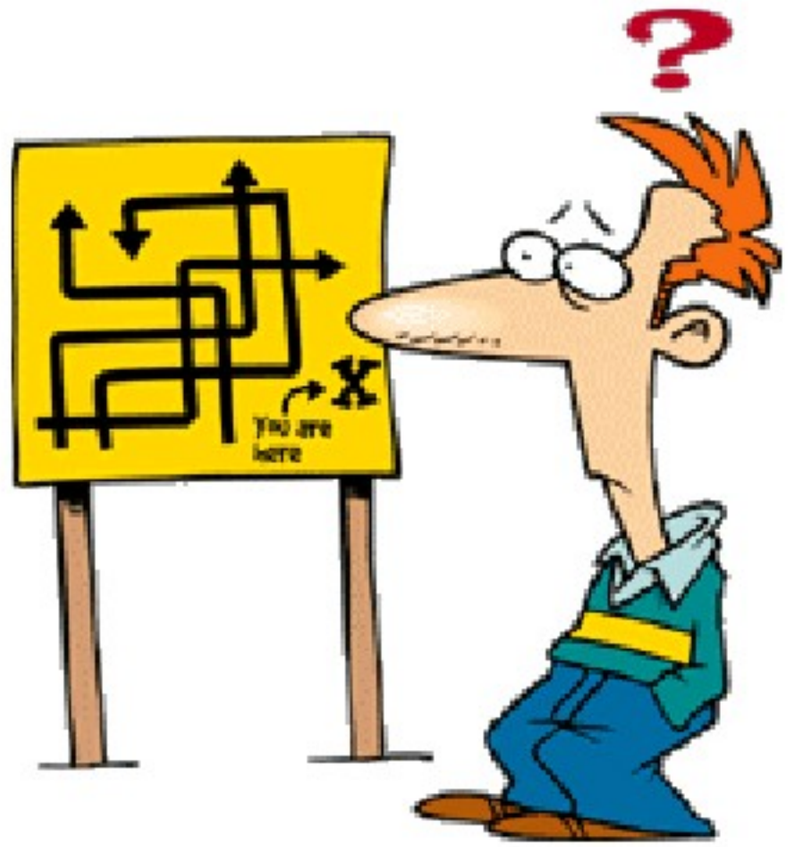
Clock Tree Synthesis

1. **Set clock tree options/exceptions**
2. **Run** `clock_opt`

- Builds the clock trees
- Performs incremental logic and placement optimizations
- Runs clock tree optimizations
- Routes the clock nets
- Can fix hold time violations
- Can perform inter-clock balancing

Routing

1. **Set routing options/exceptions**
2. **Run** `route_opt`



Performs

- Global Route
- Track Assignment
- Detailed Route

Concurrently performs

- Logic, placement and routing optimizations
- Objective: Complete routing and meet timing

Chip Finishing

- Also known as 'Design for Manufacturing'
- Entails:
  - Antenna Fixing
  - Wire Spreading
  - Double Via Insertion
  - Filler Cell Insertion
  - Metal Fill Insertion

Discussed in the Routing and Chip Finishing units

Analyzing the Results (1/2)

After each placement, CTS and routing step you should:

- **Examine the log output for design summaries:**
  - Utilization
  - Worst Negative Slack (WNS)
  - Total Negative Slack (TNS)
  - Legality of cell placement
  - Cell count and area
  - Design rule violations
- **Use `report_qor` to see:**
  - WNS/TNS per path group (clock group)
  - Other statistics

Analyzing the Results (2/2)

- **Generate more detailed reports**
  - Show all violating path end points

```
report_constraint -all_violators
```

  - Show details of the worst violating setup path

```
report_timing
```

  - Report physical design statistics (e.g. utilization)

```
report_design -physical
```

  - Analyze the congestion
    - Congestion map (GUI)

```
report_congestion
```

Example “run” Script

```
open_mw_cel ORCA_data_setup
set_tlu_plus_files -max_tluplus libs/abc_max.tlup \
                   -min_tluplus libs/abc_min.tlup \
                   -tech2itf_map libs/abc.map
read_def ORCA.def
save_mw_cel -as ORCA_floorplanned

place_opt
save_mw_cel -as ORCA_placed
report_constraint -all

remove_clock_uncertainty [all_clocks]
clock_opt
save_mw_cel -as ORCA_cts
report_constraint -all

route_opt
save_mw_cel -as ORCA_routed
report_constraint -all
report_timing

close_mw_lib
exit
```

run.tcl

Commands can be run interactively, or in “batch” mode, shown here

```
UNIX$ icc_shell -f run.tcl | tee myrun.log
```

Basic Flow Summary

You should now be able to:

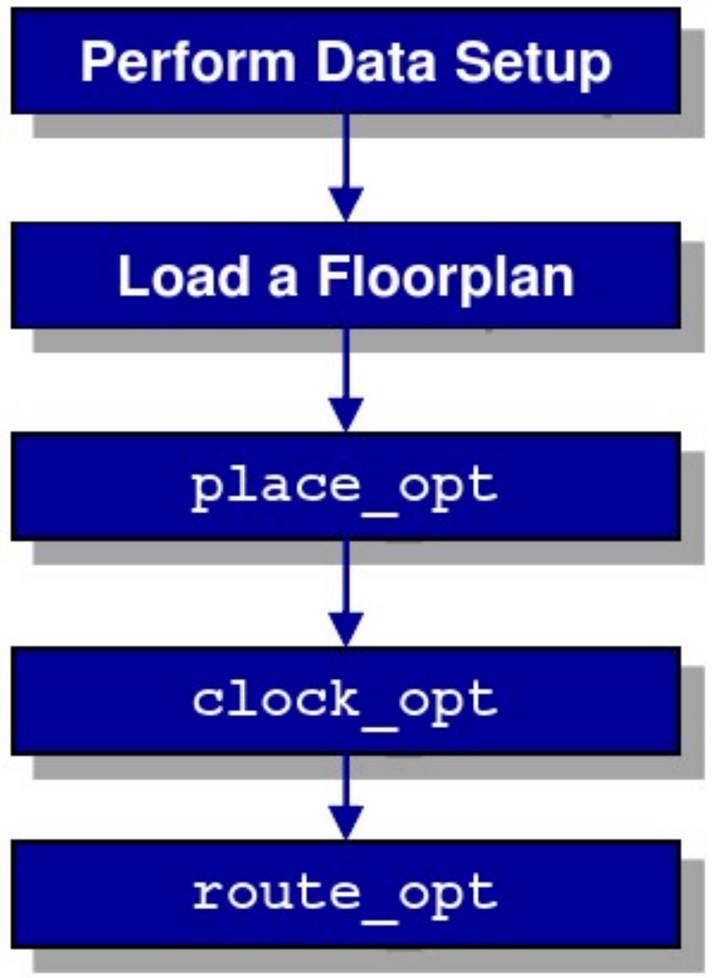
- **Perform data setup to create an initial *design cell* which is ready for design planning:**
  - Load necessary synthesis data: logical libraries, constraints, netlist
  - Load necessary physical design data: physical libraries, technology file, RC parasitic model files
  - Create a Milkyway *design library* and initial *design cell*
  - Apply timing and optimization controls
  - Perform checks on libraries, RC parasitic models, constraints and timing
- **Execute a basic flow which includes loading a floorplan and performing placement, CTS and routing**

Lab 1: Design Setup and Basic Flow

60 minutes

Goals:

- Perform data setup
- Load a floorplan
- Perform placement, CTS and routing



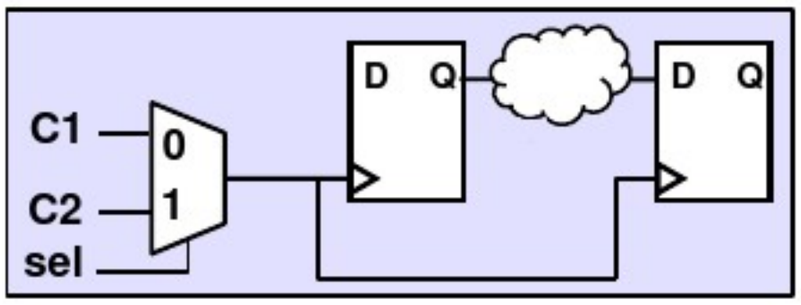
Appendix A

Some common *timing and optimization control* command and variable examples

Enable Multiple Clocks per Register

- **By default timing analysis (TA) will only consider one clock per register**
  - May pick optimistic or incorrect launching and capturing clocks
- **Can enable multiple clocks per register**
  - Set *false paths* as needed
  - TA will now consider the correct clock timing

```
set_app_var timing_enable_multiple_clocks_per_reg true
set_false_path -from [get_clocks C1] -to [get_clocks C2]
set_false_path -from [get_clocks C2] -to [get_clocks C1]
```



Default: TA picks one of four possible clock combinations (see notes)

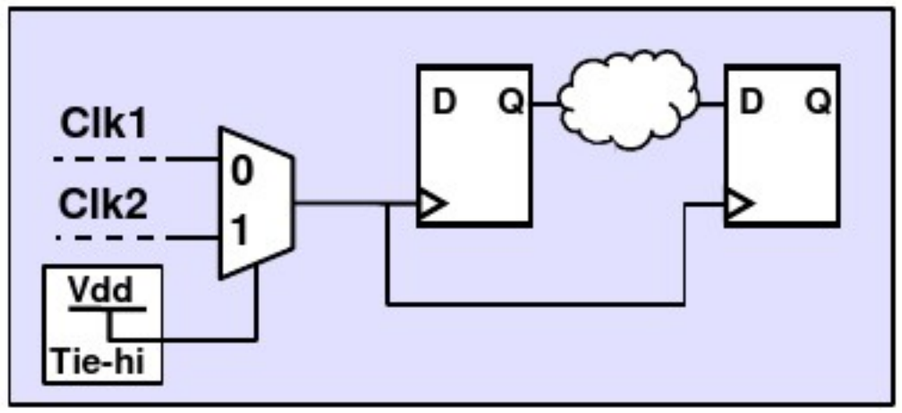
With the above settings: TA considers C1→C1 and C2→C2

Enable Constant Propagation

- **By default IC Compiler will not propagate constants**
  - Control and other signals tied high/low are not considered during timing analysis (TA) – may result in incorrect TA
- **Can enable constant propagation**

```
set_app_var case_analysis_with_logic_constants true
```

By default: No constant propagation –
TA picks one of four clock combinations

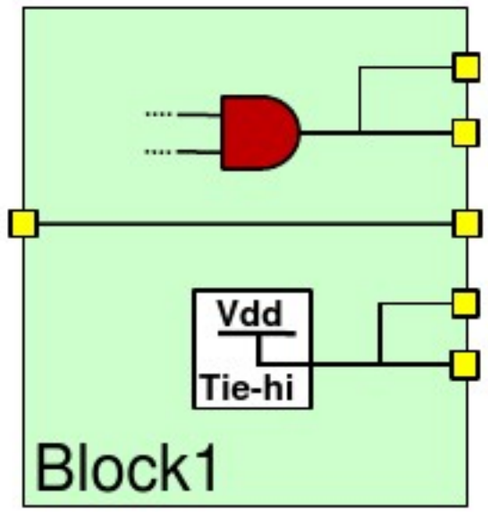


With above setting: Constant
Propagation forces TA to select Clk2→Clk2

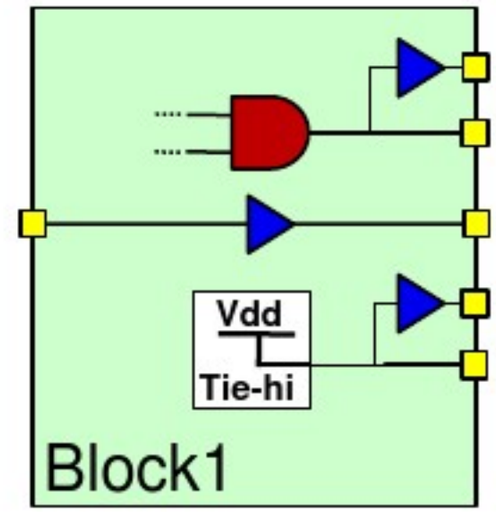
Enable Multiple Port Net Buffering

- **By default IC Compiler will not buffer nets that are connected to two or more ports**
  - Good design practice dictates that each port has a unique driver
- **Can enable multiple port net buffering**

```
set_fix_multiple_port_nets -all -buffer_constants
```



Default: Multiple port nets without buffering

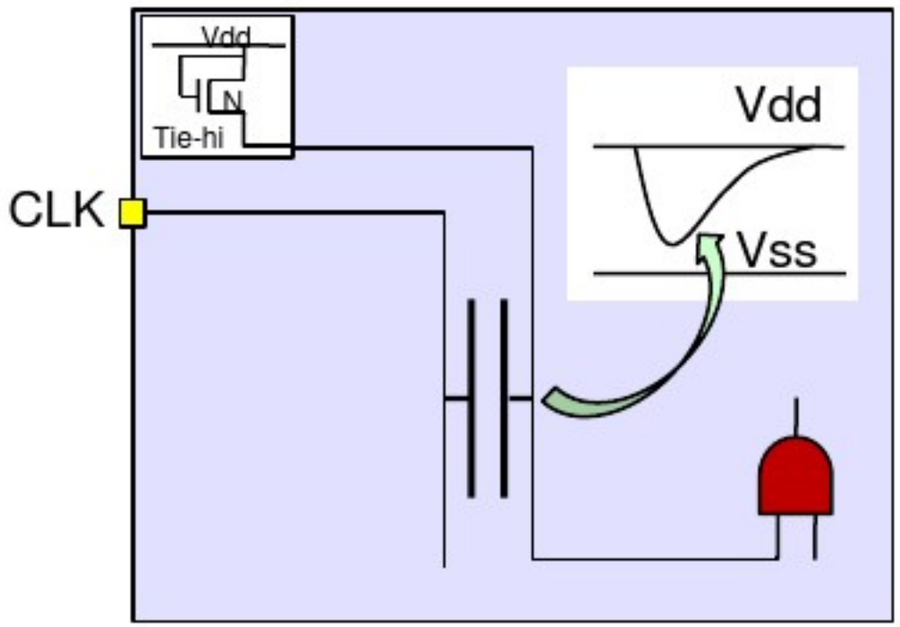


With multiple port net buffering

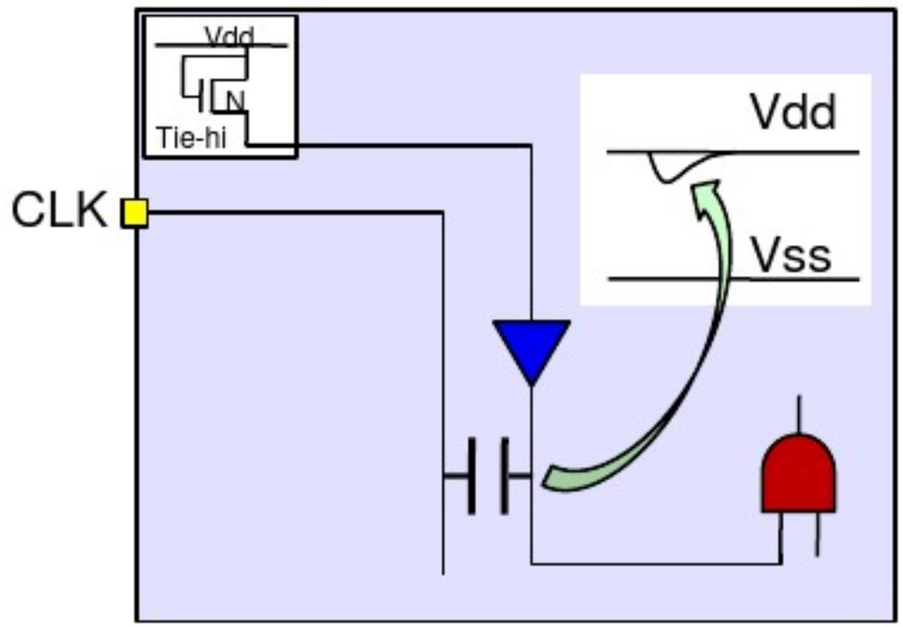
Enable Constant Net Buffering, if Needed

- **By default `place_opt` will buffer all nets except:**
  - Clock networks (done later by CTS)
  - Constant nets (tied high or low)
- **Un-buffered constant nets may be susceptible to crosstalk**
- **Can enable constant net buffering**

```
set_auto_disable_drc_nets -constant false
```



Default tie-high net



Tie-high net with buffering

Define “Don’t Use” or “Preferred” Cells

- **By default all cells available in the library can be used during buffering and optimization**
- **You may want to restrict the use of specific cells, for example:**
  - Big drivers (EM issues) and/or very weak drivers
  - Delay cells and/or clock cells

```
set_dont_use <off_limit_cells>
```

- **You may prefer the use of certain cells, when possible:**
  - Specific buffers preferred for hold time fixing

```
set_prefer -min <hold_fixing_cells>
```

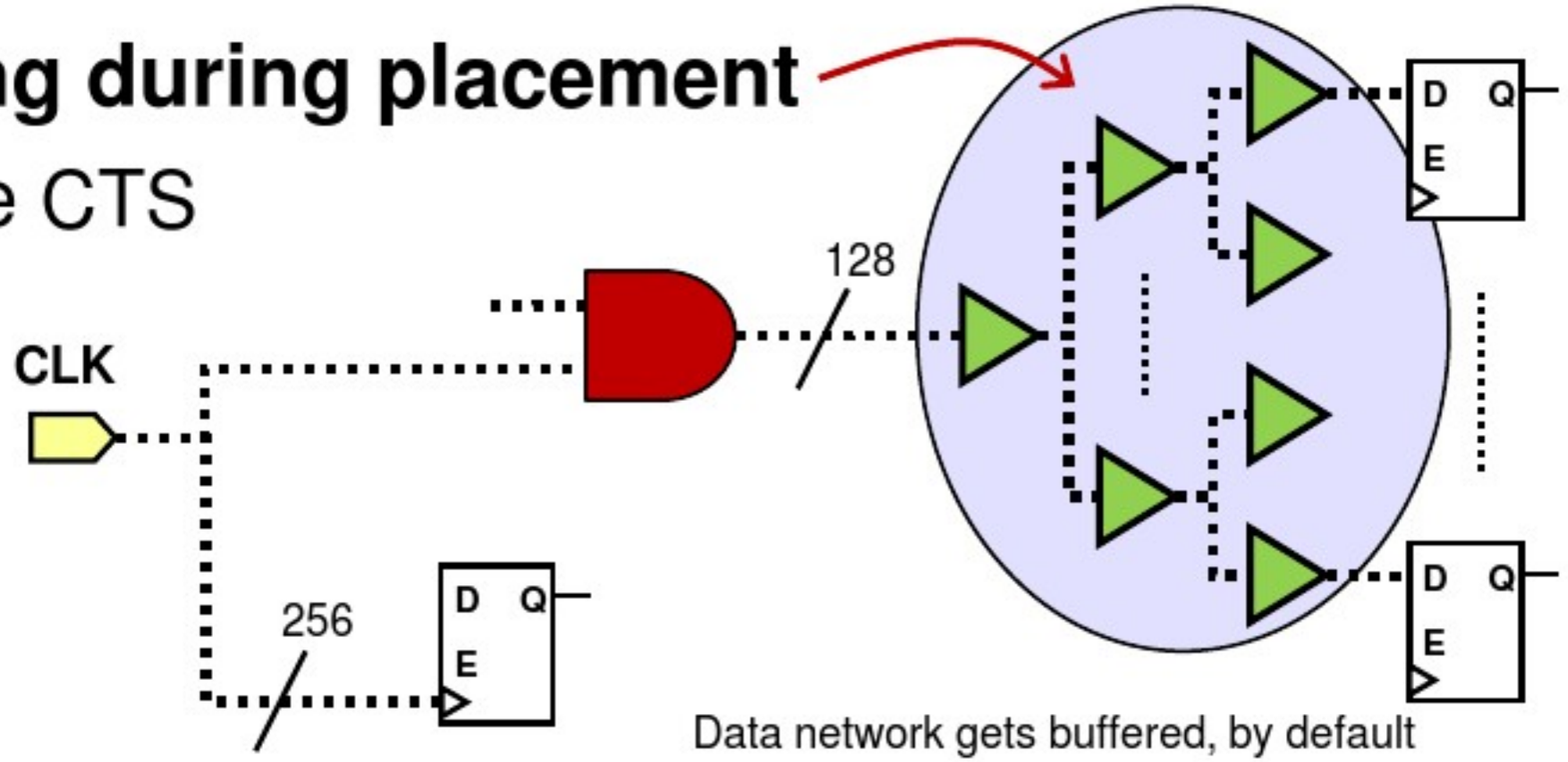
Keep Spare or Unloaded Cells

- By default any cell with unconnected output(s) will be deleted by IC Compiler
- If you need to keep such cells, *spare cells*, for example, you can turn off auto-deletion:

```
set_app_var physopt_delete_unloaded_cells false
```

Prevent Buffering of Clock-as-Data Networks

- **IC Compiler automatically treats clock signals driving clock pins as "ideal"**
  - No buffering until CTS
- **Clock signals that are used as "data" are not ideal**
  - May be buffered during placement if violating DRCs or timing
- **Prevent buffering during placement**
  - Remove before CTS



Data network gets buffered, by default

```
set_ideal_network [all_fanout -flat -clock_tree]
```

Modify Optimization Priority if Needed

- **Netlist optimization is governed by "cost priorities"**
  - If competing constraints can not all be met, higher priority constraints will be met at the expense of the lower ones
- **Default cost priorities:**

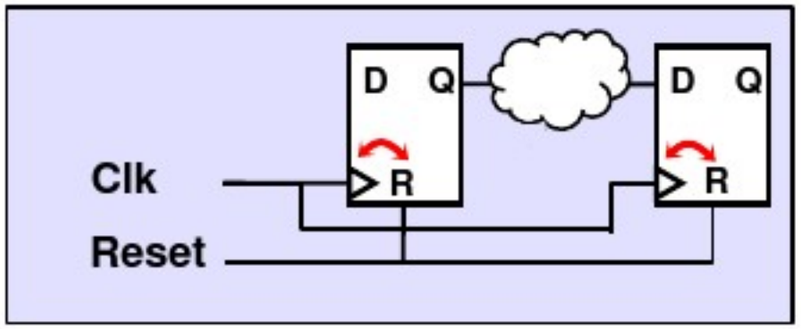
	Cost Type	Constraints
Highest	Design Rule	Max Transition; Max Fanout; Max/Min Capacitance
	Delay (max)	Clock; Max Delay
	Delay (min)	Clock; Min Delay
	Dynamic Power	Max Dynamic Power
	Leakage Power	Max Leakage Power
Lowest	Area	Max Area

- **The cost priorities can be modified, for example:**

```
set_cost_priority -delay; # See notes below
set_cost_priority {max_transition max_delay}
```

Enable Recovery and Removal Timing Arcs

- **Recovery and removal timing arcs define timing constraints between asynchronous signals of a sequential device (*set*, *reset*) and its *clock***
  - If applicable, these constraints are included in the library



- **By default IC Compiler ignores these constraints**
- **By default *Primetime* checks these constraints**
- **To enable IC Compiler to check and optimize for recovery and removal constraints, and for consistency with *Primetime*:**

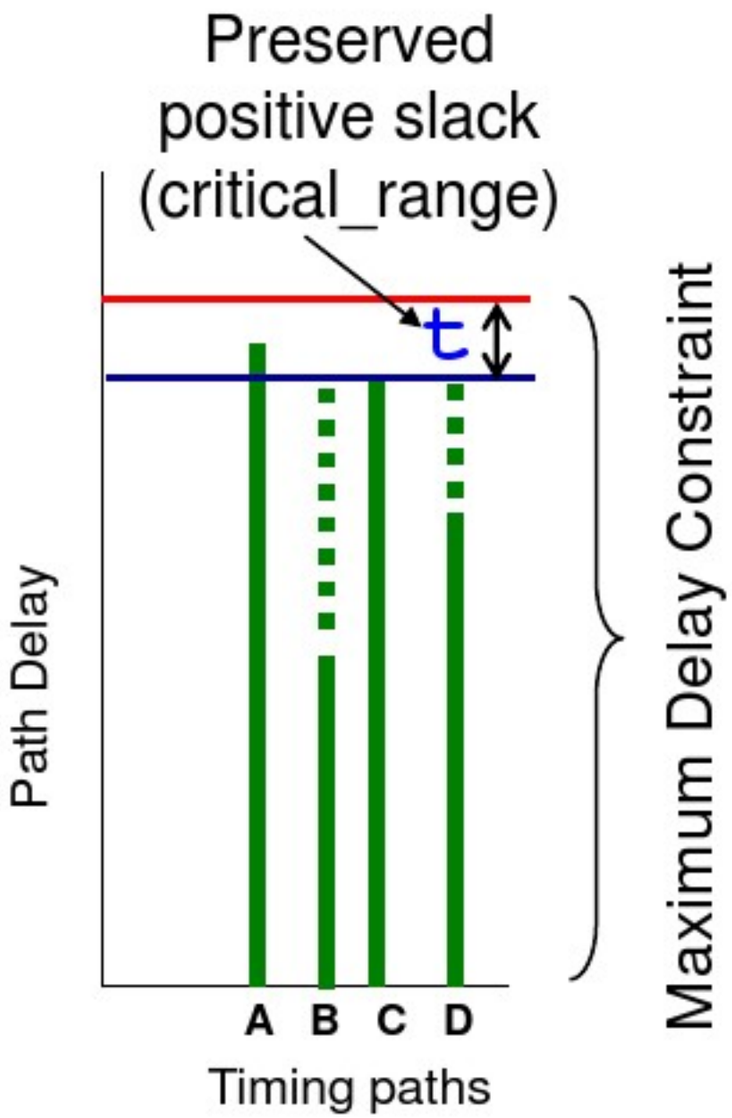
```
set_app_var enable_recovery_removal_arcs true
```

Apply Area Constraint for Area Recovery

- ***Area recovery*** **takes advantage of paths with positive timing slack to reduce or "recover" area**
  - Through cell down-sizing and buffer removal
- **Area recovery can help to reduce congestion and power consumption and is recommended during placement (`place_opt -area_recovery`)**
- **To maximize area recovery it is recommended to apply a zero *maximum area* constraint**

```
set_max_area 0
```

Apply a Power and Area Critical Range



Paths B and D can be slowed down to optimize for leakage power and/or area.

Paths A and C will be left alone.

- **Area and power optimization take advantage of paths with positive timing slack**
  - Slack can be reduced to zero, by default
  - Zero slack or margin may cause some near-critical nets to violate timing later, during CTS or routing
- **Apply a "critical range" to force optimization to preserve some positive timing slack**
  - Margin may reduce timing violations later
  - Guideline: Use ~10% of the smallest clock period

```
set_app_var physopt_power_critical_range <t>
set_app_var physopt_area_critical_range <t>
```