Lab 3 report

組別:2
組員: 劉祐瑋、陳昇達、劉佩雯
Lab: 3
HackMD link: https://hackmd.io/KXAS8ZMZSdGc5_2W3l1inw?view
(https://hackmd.io/KXAS8ZMZSdGc5_2W3l1inw?view) (歡迎利用此閱讀)
Github link: 本次實驗無

目錄

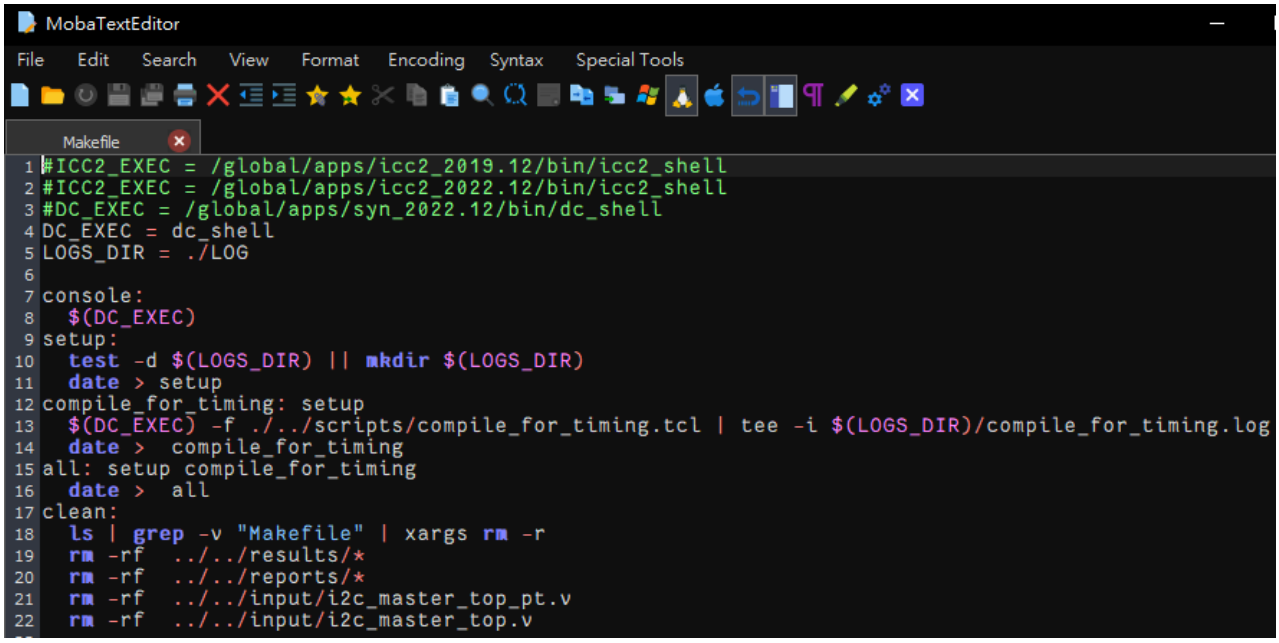
Script_explain

Lab Synthesis

Script discreption

- Makefile:
  Makefile主要執行 dc_shell ./../scripts/compile_for_timing.tcl

使用Design Compile進行logic synthesis，將Verilog或VHDL編寫的設計描述轉換成optimized gate-level netlist，並map到specific logic libary。當synthsis的設計滿足功能性、時序、功率和其他設計目標時，再將設計給icc2進行floor plan、placement and route optimization。



- Main TCL:

compile_for_timing.tcl

```tcl
source ../../common/common.tcl
set link_library          "$Std_cell_lib $Ram_lib"
set target_library        "$Std_cell_lib $Ram_lib"
set dc_allow_rtl_pg   true
source $analyze_script
elaborate ${DESIGN_NAME} -architecture verilog -library WORK
current_design ${DESIGN_NAME}
link
source $Constraints_file
set_fix_multiple_port_nets -outputs -feedthroughs
source $Warning_file
source ../scripts/set_dont_use.tcl
check_design
link
compile -exact_map -map_effort high -area_effort medium -power_effort none
report_timing > ../../reports/timing_${DESIGN_NAME}_timing_reports.log
report_qor > ../../reports/timing_${DESIGN_NAME}_qor_reports.log
report_area -hierarchy  > ../../reports/timing_${DESIGN_NAME}_area_reports.log
report_power -hierarchy > ../../reports/timing_${DESIGN_NAME}_power_reports.log
change_names -rules verilog
write_file -format verilog -hierarchy -pg -output ../../input/${DESIGN_NAME}.v
quit
```

- 逐行解釋 Main TCL:

```
第01行:先做基本設定:連結tool path,design library file,constraints file,tech file,
第02行:為了使設計完整,Design Compiler將all cell instances與 library components和設
第03行:使用者指定standard library,讓Design Compiler mapping設計中的標準單元,像是com
第04行:在RTL生成PG pin
第05行:檢查design以及報告錯誤,並且產生中繼檔
第06行:透過第5行產生的中繼檔,將design轉換成technology-independent design(GTECH)
第07行:設定目前design
第08行:連結目前design
第09行:對速度做優化約束
第10行:設定防止multiple-port connections
第11行:去除不重要的警告
第12行:指定在優化過程中排除一些在target library的特定cell
第13行:檢查當前設計的內部表示的一致性,並根據需要發出錯誤和警告訊息
第14行:連結目前design
第15行:編譯順序單元應按照HDL中的指示正確映射來產生gate-level description
第16行:報告timing information
第17行:報告QoR information
第18行:報告area information
第19行:報告power information
第20行:確保每個檔名match
第21行:用verilog format及hierarchy形式,將desgin寫入到檔案
第22行:離開Design Compiler
```

- 補充Main TCL:

Main TCL 根據圖1的Design flow 與圖2的Synthesis flow

圖1顯示Design Compiler in the Design flow:

Figure 1 shows an overview of how Design Compiler fits into the design flow.

Figure 1 Design Compiler in the Design Flow

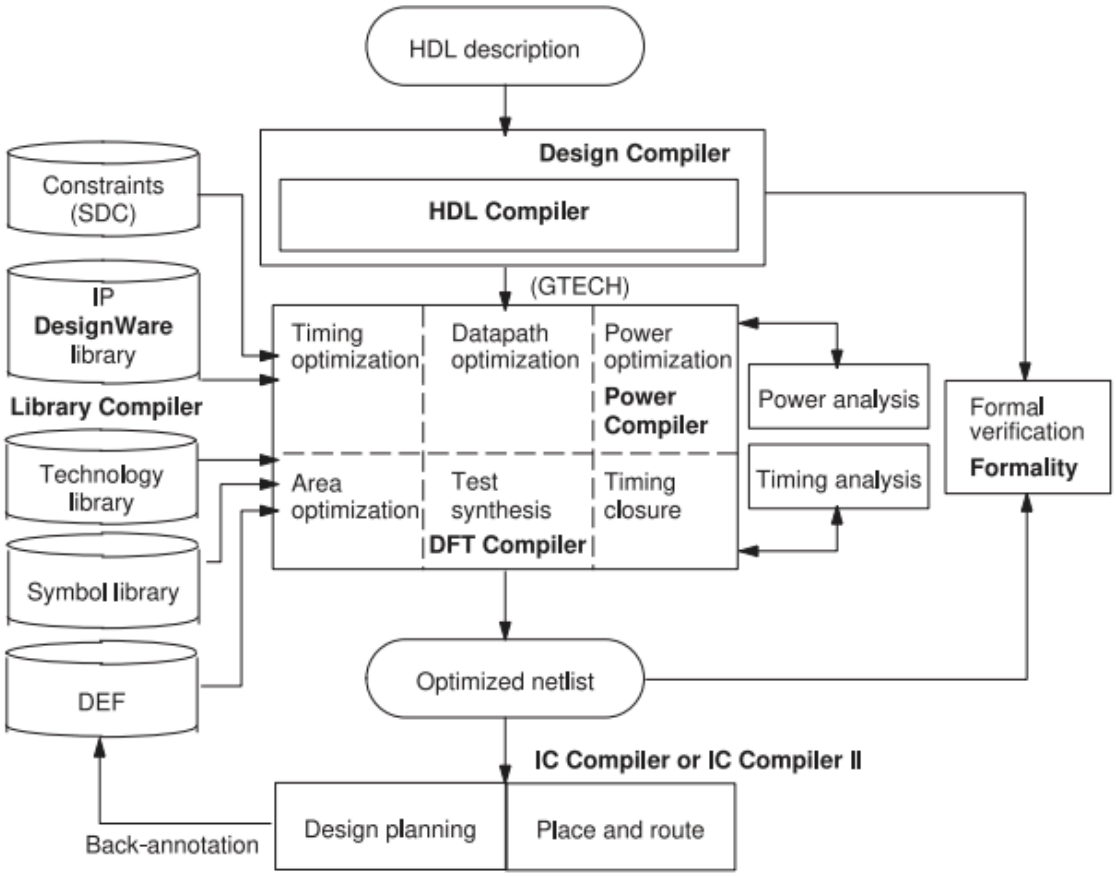
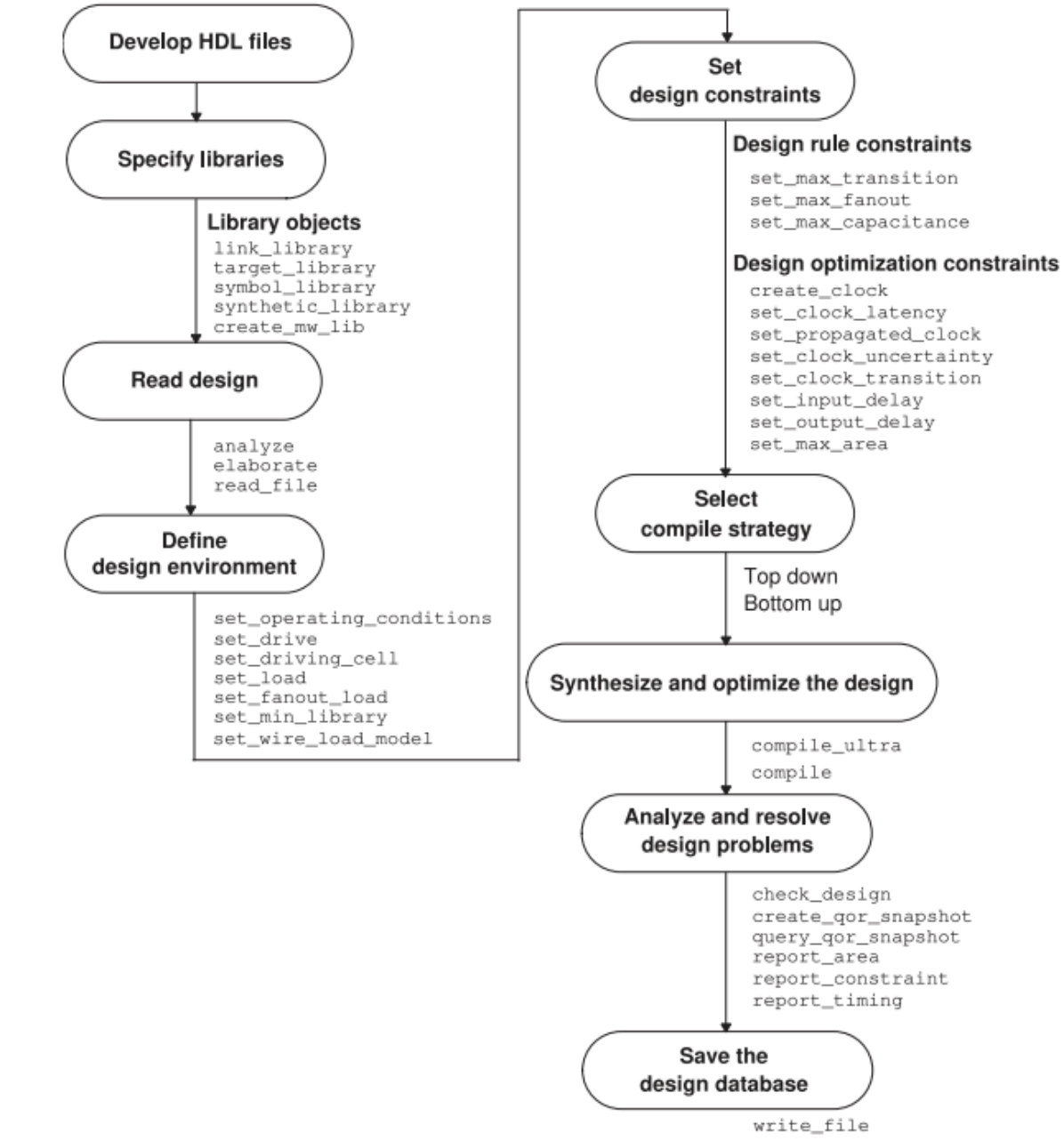


圖2顯示Design Compiler in the Synthesis flow:

Figure 6 Synthesis Flow



第02行:set link_library "$Std_cell_lib $Ram_lib"

dcug.pdf,page112

Link Libraries

For a design to be complete, all cell instances in the design must be linked to the library components and designs that are referenced. This process is called linking the design or resolving references. To resolve references, Design Compiler uses the link libraries set by the following variables and attribute:

- The `link_library` application variable lists the libraries and design files that Design Compiler uses to resolve references.

  Design Compiler searches the files listed in the `link_library` variable from left to right, and it stops searching when it finds a reference. Specifying an asterisk in the `link_library` variable means that Design Compiler searches loaded libraries in memory for the reference. For example, if you set the `link_library` variable to {"*" lsi_10k.db}, Design Compiler searches for the reference in memory first and then in the lsi_10k library.

- The `local_link_library` attribute lists the design files and libraries added to the beginning of the `link_library` variable during the link process. Design Compiler searches files in the `local_link_library` attribute first when it resolves references. You can set this attribute by using the `set_local_link_library` command.

- The `search_path` variable specifies a list of directory paths that the tool uses to find logic libraries and other files when you specify a plain file name without a path. It also sets the paths where Design Compiler can continue the search for unresolved references after it searches the link libraries.

  If Design Compiler does not find the reference in the link libraries, it searches in the directories specified by the `search_path` variable, as described in Specifying a Library Search Path.

第03行:set target_library "$Std_cell_lib $Ram_lib"

dcug.pdf,page111

Target Libraries

Design Compiler selects functionally correct gates from the target libraries to build a circuit during mapping. It also calculates the timing of the circuit by using the vendor-supplied timing data for these gates.

To specify the target libraries, use the `target_library` variable. Your should specify only the standard cell libraries that you want Design Compiler to use for mapping the standard cells in your design, such as combinational logic and registers. You should not specify any DesignWare libraries or macro libraries, such as pads or memories.

Specifying Logic Libraries

Table 4 lists the variables that control library reading for each library type and the typical file name. You use these variables to specify logic libraries and DesignWare libraries.

Table 4 Library Variables

Library type	Variable	Default	File extension
Target library	`target_library`	`{your_library.db}`	.db
Link library	`link_library`	`{* your_library.db}`	.db
Symbol library	`symbol_library`	`{your_library.sdb}`	.sdb
DesignWare library	`synthetic_library`	""	.sldb

第04行:set dc_allow_rtl_pg true

dcug.pdf,page104-105

Instantiating RTL PG Pins

Design Compiler can accept RTL designs containing a small number of PG pin connections on macros. The tool does not support a full PG netlist for a block. For example, the tool only supports designs that contain a small number of analog macros that have PG pins.

To instantiate PG pins in your RTL design, set the `dc_allow_rtl_pg` variable to `true`. The default is `false`. To preserve the PG connections in a Verilog output, execute the

`write_file -pg -format verilog` command. To preserve the PG connections in a .ddc format output, execute the `write_file -format ddc` command. Note that when saving the design in .ddc format, you do not need to use the `-pg` option. When reading the .ddc file back into Design Compiler, make sure that the `dc_allow_rtl_pg` variable is set to `true`; otherwise, the tool issues a DDC-21 error:

```
Error: The feature used to generate this DDC file is not supported by
this tool or is not enabled in the current session. (DDC-21)
Information: This .ddc file contains RTL PG data. Set the dc_allow_rtl_pg
variable to true before reading the file back into Design Compiler.
```

To preserve the PG connections in a Milkyway database, use the `write_milkyway` command. To pass the design netlist to IC Compiler, PrimeTime, or Formality, you can use a Verilog output, .ddc file, or Milkyway database.

第05行:analyze -library WORK -format sverilog "

dcug.pdf,page151

The `analyze` command performs the following tasks:

- Reads an HDL source file
- Checks for errors without building generic logic for the design
- Creates HDL library objects in an HDL-independent intermediate format
- Stores the intermediate files in a location you define

If the `analyze` command reports errors, fix them in the HDL source file and then run the `analyze` command again. After a design is analyzed, you must reanalyze it only when you change it.

第06行:elaborate ${DESIGN_NAME} -architecture verilog -library WORK

dcug.pdf,page152

The `elaborate` command performs the following tasks:

- Translates the design into a technology-independent design (GTECH) from the intermediate files produced during analysis
- Allows changing of parameter values defined in the source code

- Allows VHDL architecture selection
- Replaces the HDL arithmetic operators in the code with DesignWare components
- Automatically executes the `link` command, which resolves design references

To use this method, analyze the top-level design and all subdesigns in bottom-up order and then elaborate the top-level design and any subdesigns that require parameters to be assigned or overwritten:

```
prompt> analyze -format verilog -library -work RISCTYPES.v
prompt> analyze -format verilog -lib -work {ALU.v STACK_TOP.v\
                STACK_MEM.v ...}
prompt> elaborate RISC_CORE -architecture STRUCT -library WORK -update
```

第07行:current_design ${DESIGN_NAME}

dcug.pdf,page154

Setting the Current Design

You can set the current design (the design you are working on) in the following ways:

- With the `read_file` command

  When the `read_file` command successfully finishes processing, it sets the current design to the design that was read in.

  ```
  prompt> read_file -format ddc MY_DESIGN.ddc
  Reading ddc file '/designs/ex/MY_DESIGN.ddc'
  Current design is 'MY_DESIGN'
  ```

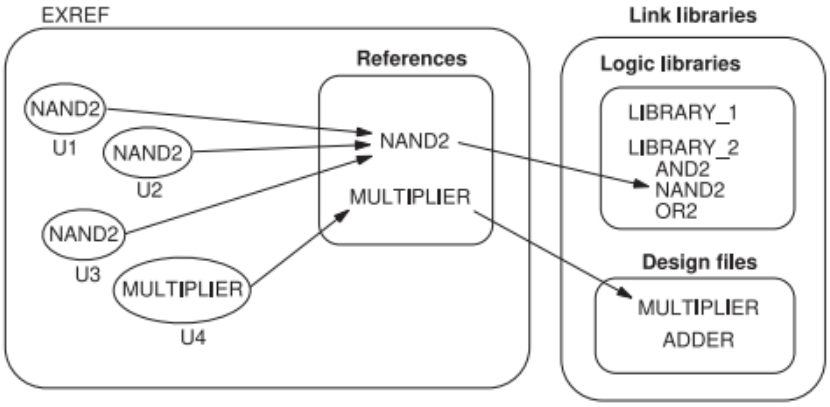
- With the `elaborate` command
- With the `current_design` command

  Use this command to set any design in dc_shell memory as the current design.

第08行:link

dcug.pdf,page155

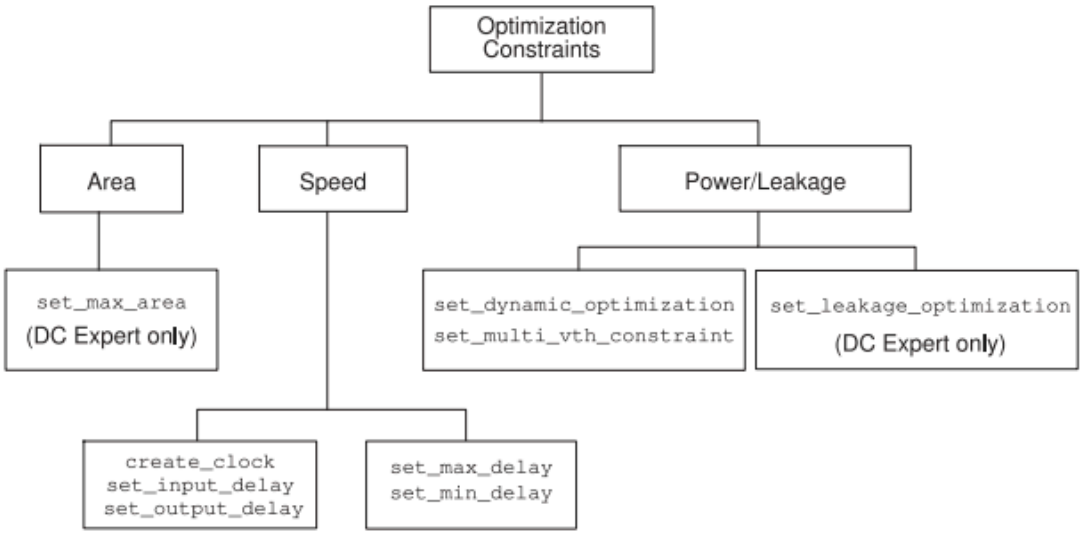
Figure 20 Resolving References



第09行:source ../../common/i2c_master_top.sdc

dcug.pdf,page106,page213,page226

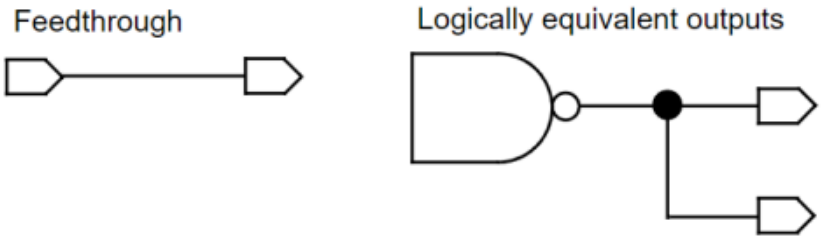
Figure 28 Major Design Optimization Constraints



第10行:set_fix_multiple_port_nets -outputs -feedthroughs

dcug.pdf,page408

Figure 90 Multiple-Port Connection Types



To represent such nets, Design Compiler uses `assign` statements in the gate-level netlist. Back-end tools could potentially have problems with `assign` statements in the netlist.

To prevent multiple-port connections, use the `set_fix_multiple_port_nets` command to set the `fix_multiple_port_nets` attribute on a design.

Use the `set_fix_multiple_port_nets` command to control

- Feedthroughs

  Specify the `-feedthroughs` option to insert buffers so that input ports are isolated from output ports. A feedthrough net occurs when an input port and output port are connected directly with no intervening logic.

- Multiple output ports

  Specify the `-outputs` option to insert buffers so that no cell driver pin drives more than one output port.

- Constants

  Specify the `-constants` option to duplicate constant logic so that no constant drives more than one output port. Alternatively, you can use the `-buffer_constants` option to buffer logic constants instead of duplicating them.

To exclude clock network nets from buffering during multiple-port nets fixing, use the `-exclude_clock_network` option.

You can also fix multiple-port nets, feedthroughs, and constant-driven ports by rewiring the connections elsewhere in the hierarchy first. Any remaining feedthroughs, nets with logically equivalent outputs, and constants that could not be fixed by rewiring are fixed by buffering. To enable this feature, set the `compile_advanced_fix_multiple_port_nets` variable to `true`. It is set to `false` by default.

Note:

Design Compiler does not rewire across the hierarchy if boundary optimization is disabled.

第12行:source ../scripts/set_dont_use.tcl

dcug.pdf,page121

Excluding Cells From the Target Libraries

When Design Compiler maps a design to a logic library, it selects library cells from this library. To specify cells in the target library to be excluded during optimization, use the `set_dont_use` command. For example, to prevent Design Compiler from using the INV_HD high-drive inverter, enter

```
prompt> set_dont_use MY_LIB/INV_HD
```

This command affects only the copy of the library that is currently loaded in memory and has no effect on the version that exists on disk. However, if you save the library, the exclusions are saved, and the cells are permanently excluded.

> 第13行:check_design
> ***dcug.pdf,page796,page849***

> 第15行:compile -exact_map -map_effort high -area_effort medium ->
> power_effort none
> ***dcug.pdf,page468***

> 第16-19行: report_xxx
> ***dcug.pdf,page393***

> 第20行:change_names -rules verilog
> ***dcug.pdf,page186***

> 第21行:write_file -format verilog -hierarchy -pg -output
> ../../input/${DESIGN_NAME}.v
> ***dcug.pdf,page72***

- 參考
  https://drive.google.com/drive/folders/1DS7P8PmUP2mCEkGKJ4PQ0MUdv5HzQNYh (https://drive.google.com/drive/folders/1DS7P8PmUP2mCEkGKJ4PQ0MUdv5HzQNYh)

Lab1 Floorplan

Script discreption

- Makefile:
  Makefile主要執行3個步驟:

1. icc2_shell ./../scripts/step1_data_setup.tcl
2. icc2_shell ./../scripts/step2_floorplan.tcl
3. icc2_shell ./../scripts/step3_powerplan.tcl

ICC2的Design Planning提供了以下功能，以開發hierarchical and flat designs，例如：
-多層Physical Hierarchy Floorplanning
-高級抽象管理，用於處理大型設計
-圖形任務管理器guided floorplanning
-數據流分析，用於 macro management and placement assistance
-快速放置 macros and standard cells
-Pattern-based power planning
-使用 UPF 的Power network描述
-虛擬就地時序優化
-針腳放置和匯流排bundling
-時序預算
這些功能有助於提高設計的效率和質量，並確保最終產品能夠滿足性能和功耗的要求。
這些工具的使用可以顯著減少設計時間，並提高大型複雜設計的可管理性。

Makefile

```
#ICC2_EXEC = /global/apps/icc2_2019.12/bin/icc2_shell
#ICC2_EXEC = /global/apps/icc2_2022.12-SP5/bin/icc2_shell
ICC2_EXEC = icc2_shell
LOGS_DIR = ./LOG

console:
  $(ICC2_EXEC)
setup:

  test -d $(LOGS_DIR) || mkdir $(LOGS_DIR)
  date > setup
step1_data_setup: setup
  test -e $(LOGS_DIR)/../../results/i2c_master_top || rm -rf ../../results/i2c_master_top
  $(ICC2_EXEC) -f ./../scripts/step1_data_setup.tcl | tee -i $(LOGS_DIR)/step1_data_setup.log
  date >  step1_data_setup
step2_floorplan: setup step1_data_setup
  $(ICC2_EXEC) -f ./../scripts/step2_floorplan.tcl | tee -i $(LOGS_DIR)/step2_floorplan.log
  date > step2_floorplan
step3_powerplan: setup step2_floorplan
  $(ICC2_EXEC) -f ./../scripts/step3_powerplan.tcl | tee -i $(LOGS_DIR)/step3_powerplan.log
  date >  step3_powerplan
all: setup step3_powerplan
  date > all
clean:
  ls | grep -v "Makefile" | xargs rm -r
  rm -rf ../../results/i2c_master_top
```

- Main TCL:

1. step1_data_setup.tcl

step1_data_setup.tcl

```
source ../../common/common.tcl
#########################################################################Set_Library
set link_library          "$Std_cell_lib $Ram_lib"
set target_library        "$Std_cell_lib $Ram_lib"
#########################################################################Create_Lib
create_lib $ARC_TOP \
        -technology $Tech_file  \
        -ref_libs $REFERENCE_LIB
#########################################################################Set_Tluplus_Files
read_parasitic_tech -tlup "$Tlup_max_file $Tlup_min_file" \
                    -layermap $Map_file
#########################################################################Creat_Block
read_verilog "$Core_compile"
current_design ${DESIGN_NAME}
source $Constraints_file
save_block -as ${DESIGN_NAME}_1_data_setup
save_lib
close_block
close_lib
exit
```

2. step2_floorplan.tcl

step2_floorplan.tcl | step3_powerplan.tcl

```
open_lib $ARC_TOP
copy_block -from_block ${DESIGN_NAME}_1_data_setup -to temp_data_setup
open_block temp_data_setup
################################################################################################Reports
report_clocks -skew -attributes
report_exceptions
report_disable_timing
################################################################################################Set_Power/Ground_Nets_And_Pins
set power                           "VDD"
set ground                          "VSS"
set powerPort                       "VDD"
set groundPort                      "VSS"
set ndm_logic0_net                  "VSS"
set ndm_logic1_net                  "VDD"
################################################################################################Set_Options
set_app_option -name time.disable_recovery_removal_checks -value false
set_app_option -name time.disable_case_analysis -value false
group_path -name INPUT  -from [all_inputs]
group_path -name OUTPUT -to   [all_outputs]
group_path -name COMBO -from [all_inputs] -to [all_outputs]
################################################################################################Save_Block
save_block -as temp_floorplan_init
################################################################################################Defining_Prefered_Routing_Directions
set_ignored_layers -min_routing_layer ${route_min_layer} -max_routing_layer ${pns_hlayer}
set_attribute [get_layers M1] routing_direction vertical
set_attribute [get_layers M2] routing_direction horizontal
set_attribute [get_layers M3] routing_direction vertical
set_attribute [get_layers M4] routing_direction horizontal
set_attribute [get_layers M5] routing_direction vertical
set_attribute [get_layers M6] routing_direction horizontal
set_attribute [get_layers M7] routing_direction vertical
set_attribute [get_layers {M1}] track_offset 0.037
################################################################################################Create_Floorplan
initialize_floorplan -core_utilization 0.2 \
            -core_offset {10 10 10 10}
################################################################################################Ports_Placement
place_pins -ports [get_ports *]
################################################################################################Defining_Power/Ground_Nets_And_Pins
set_attribute -objects [get_nets VDD] -name net_type -value power
set_attribute -objects [get_nets VSS] -name net_type -value ground
check_mv_design
################################################################################################Save_Block
save_block -as temp_floorplane
################################################################################################Create_Floorplane_Placement
create_placement -floorplan -effort high -timing_driven
legalize_placement
route_global -congestion_map_only true -effort high
report_placement
################################################################################################Save_Block
save_block -as temp_floorplane_placed
save_block -as ${DESIGN_NAME}_2_floorplan_ends
save_lib
close_block
close_lib
exit
```

3. step3_powerplan.tcl

step3_powerplan.tcl

```tcl
source ../../common/common.tcl
################################################################################Open_Lib
open_lib $ARC_TOP
copy_block -from_block ${DESIGN_NAME}_2_floorplan_ends -to temp_floorplan_ends
open_block temp_floorplan_ends
################################################################################Power_Planing
set_attribute -objects [get_nets VDD] -name net_type -value power
set_attribute -objects [get_nets VSS] -name net_type -value ground
connect_pg_net -net VDD [get_pins -physical_context */VDD]
connect_pg_net -net VSS [get_pins -physical_context */VSS]
remove_pg_via_master_rules -all
remove_pg_patterns -all
remove_pg_strategies -all
remove_pg_strategy_via_rules -all
################################################################################Create_STD_Cells_Rail
create_pg_std_cell_conn_pattern M1_rail -layers {M1} -rail_width {@wtop @wbottom} -parameters {wtop wbottom}
set_pg_strategy M1_rail_strategy_pwr -core -pattern {{name: M1_rail} {nets: VDD} {parameters: {0.094 0.094}}}
set_pg_strategy M1_rail_strategy_gnd -core -pattern {{name: M1_rail} {nets: VSS} {parameters: {0.094 0.094}}}
compile_pg -strategies M1_rail_strategy_pwr -ignore_drc
compile_pg -strategies M1_rail_strategy_gnd -ignore_drc
################################################################################Create_Top_Vertical_Mesh
create_pg_mesh_pattern TOP_MESH_VERTICAL \
    -layers " \
        { {vertical_layer: M6}   {width: 0.3} {spacing: interleaving} {pitch: 4} {offset: 0.5}  {trim : true} } \
        "
set_pg_strategy VDDVSS_TOP_MESH_VERTICAL \
    -core \
    -pattern   { {name: TOP_MESH_VERTICAL} {nets:{VSS VDD}} } \
    -extension { {{stop:design_boundary_and_generate_pin}} }
compile_pg -strategies {VDDVSS_TOP_MESH_VERTICAL}
################################################################################Create_Top_Horizontal_Mesh
create_pg_mesh_pattern TOP_MESH_HORIZONTAL \
    -layers " \
        { {horizontal_layer: M7}   {width: 0.3} {spacing: interleaving} {pitch: 4} {offset: 0.5}  {trim : true} }
        "
set_pg_strategy VDDVSS_TOP_MESH_HORIZONTAL \
    -core \
    -pattern   { {name: TOP_MESH_HORIZONTAL} {nets:{VSS VDD}} } \
    -extension { {{stop:design_boundary_and_generate_pin}} }
compile_pg -strategies {VDDVSS_TOP_MESH_HORIZONTAL}
################################################################################Create_Rectangular_Rings
create_pg_ring_pattern \
                ring_pattern \
                -horizontal_layer M7  -vertical_layer M6 \
                -horizontal_width 1 -vertical_width 1 \
                -horizontal_spacing 3 -vertical_spacing 3

set_pg_strategy RING -core -pattern {{ name: ring_pattern} { nets: "VDD VSS" }}
compile_pg -strategies RING
check_pg_connectivity -nets "VDD VSS"
################################################################################Save_Block
save_block -as ${DESIGN_NAME}_3_powerplan_ends
save_lib
close_block
close_lib
exit
```

- 逐行解釋 Main TCL:

1. step1_data_setup.tcl

第01行:先做基本設定:連結tool path,design library file,constraints file等等...
#設定Ref. Library
第03行:指定library作為目前設計的reference library
第04行:在優化過程中，icc2可以從target_library中選擇任何library cell
#建立Library
第06行:建立新的design library
#設定Tlupuls file
第10行:將TLUPlus models讀入library工作區
#建立Block
第13行:讀取design netlist,如同新建立block
第14行:指定在logic netlist中的hierarchical module name當作目前的block
第15行:Clock Tree Synthesis的前置作業
第16行:儲存block到disk
第17行:儲存design library到disk
第18行:關閉block,將block從memory移除
第19行:將design library從memory移除
第20行:離開icc2_shell

2. step2_floorplan.tcl

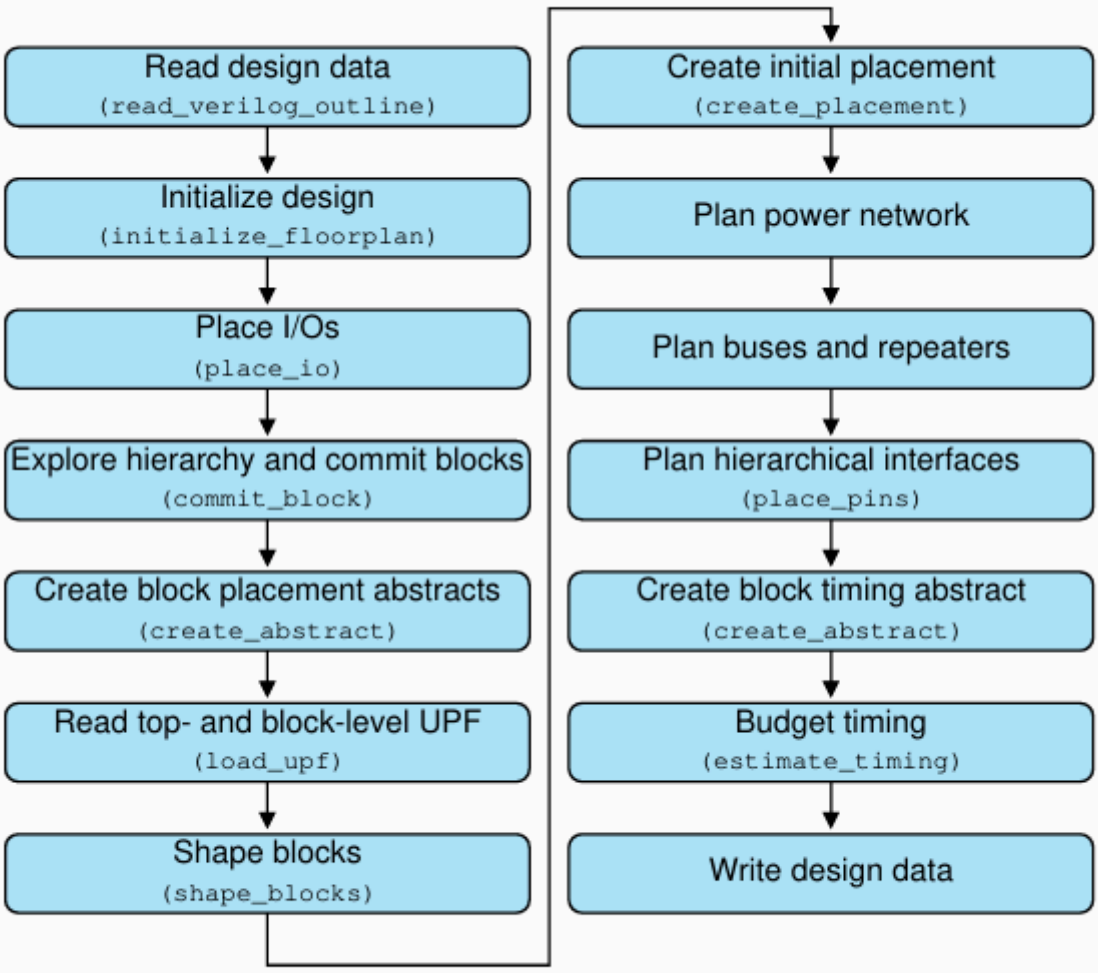
第01行:先做基本設定:連結tool path,design library file,constraints file等等...
#設定Ref. Library
第03行:指定library作為目前設計的reference library
第04行:在優化過程中，icc2可以從target_library中選擇任何library cell
#開啟 Library
第06行:開啟已儲存的library來看或編輯
第07行:在同樣的library複製block到新的block
第08行:開啟已儲存的block來看或編輯
#Report
第10行:報告clocks information
第11行:報告exceptions information
第12行:報告disable_timing information
#設定Pins/節點/Port
第14-19行:power設定為"VDD";ground設定為"VSS"
#設定Option
第21-22行:使用者自行需求設定應用選項
第23-25行:優化路徑
#儲存Block
第27行:將編輯好的block存到disk
#設定Routing方向
第29行:設定可忽略的layers
第30-37行:奇數層是垂直方向;偶數層是水平方向
#建立FloorPlan
第39行:在design planning初始化floorplan
#Ports放置
第42行:在design planning展現pin placement
#設定Pins/節點/Port
第44-45行:設定VDD,VSS
第46行:檢查design的電路是否有接好
#儲存Block
第48行:將編輯好的block存到disk
#建立FloorPlan and Placement
第50行:建立一個粗略的布局
第51行:合法的布局
第52行:展現global繞線
第53行:報告placement QoR information
#儲存Block
第55-56行:將編輯好的block存到disk
第57行:儲存design library到disk
第58行:關閉block,將block從memory移除
第59行:將design library從memory移除
第60行:離開icc2_shell

3. step3_powerplan.tcl

第01行:先做基本設定:連結tool path,design library file,constraints file等等...
#開啟 Library
第03行:開啟已儲存的library來看或編輯
第04行:在同樣的library複製block到新的block
第05行:開啟已儲存的block來看或編輯
#佈局Power
第07行:設定節點VDD值為power
第08行:設定節點VSS值為ground
第09行:連接VDD pins到PG net
第10行:連接VSS pins到PG net
第11行:移除PG via規則
第12行:移除PG pattern
第13行:移除PG strategies
第14行:移除PG strategy via rules
#建立Standard cell Rail
第16行:設定M1 rail連接模式
第17行:設定M1 rail power network PG佈局策略
第18行:設定M1 rail ground network PG佈局策略
第19行:建立M1_rail power network
第20行:建立M1_rail ground network
#建立Top垂直Mesh
第22行:建立TOP_MESH_VERTICAL一個PG mesh pattern
第23行:對VDDVSS_TOP_MESH_VERTICAL power ground network設定PG策略
第30行:建立VDDVSS_TOP_MESH_VERTICAL power ground network
#建立Top水平Mesh
第22行:建立TOP_MESH_HORIZONTAL一個PG mesh pattern
第23行:對VDDVSS_TOP_MESH_HORIZONTAL power ground network設定PG策略
第30行:建立VDDVSS_TOP_MESH_HORIZONTAL power ground network
#建立Rectangular Rings
第22行:建立ring_pattern一個PG ring pattern
第23行:對RING power ground network設定PG策略
第30行:建立RING power ground network
第31行:檢查PG network連接
#儲存Block
第52行:將編輯好的block存到disk
第53行:儲存design library到disk
第54行:關閉block,將block從memory移除
第55行:將design library從memory移除
第56行:離開icc2_shell

- 補充Main TCL:

Figure 1 Hierarchical Design Planning Flow



1. step1_data_setup.tcl

Data setup主要是建立library與block兩個操作架構組合而成

圖1建立Design Library

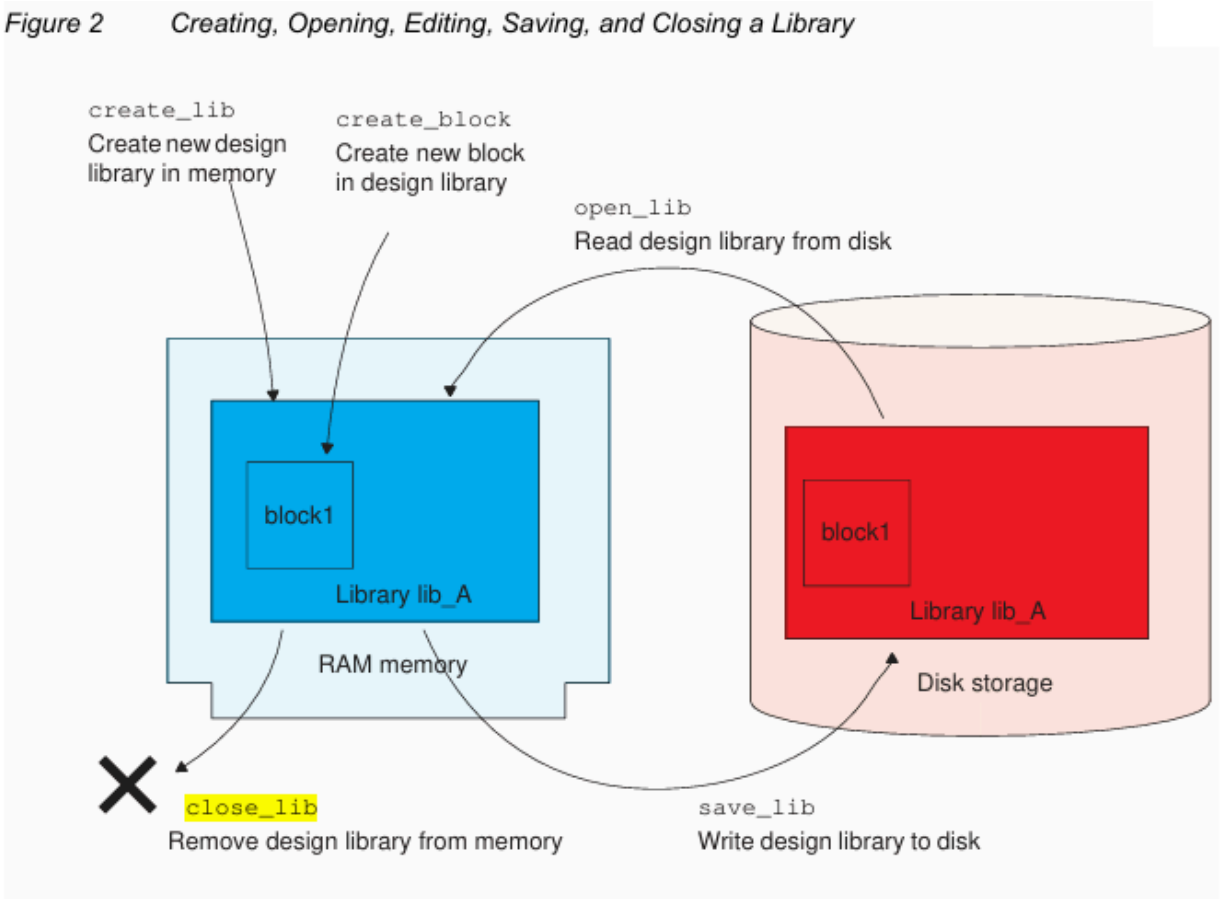


圖2建立Block

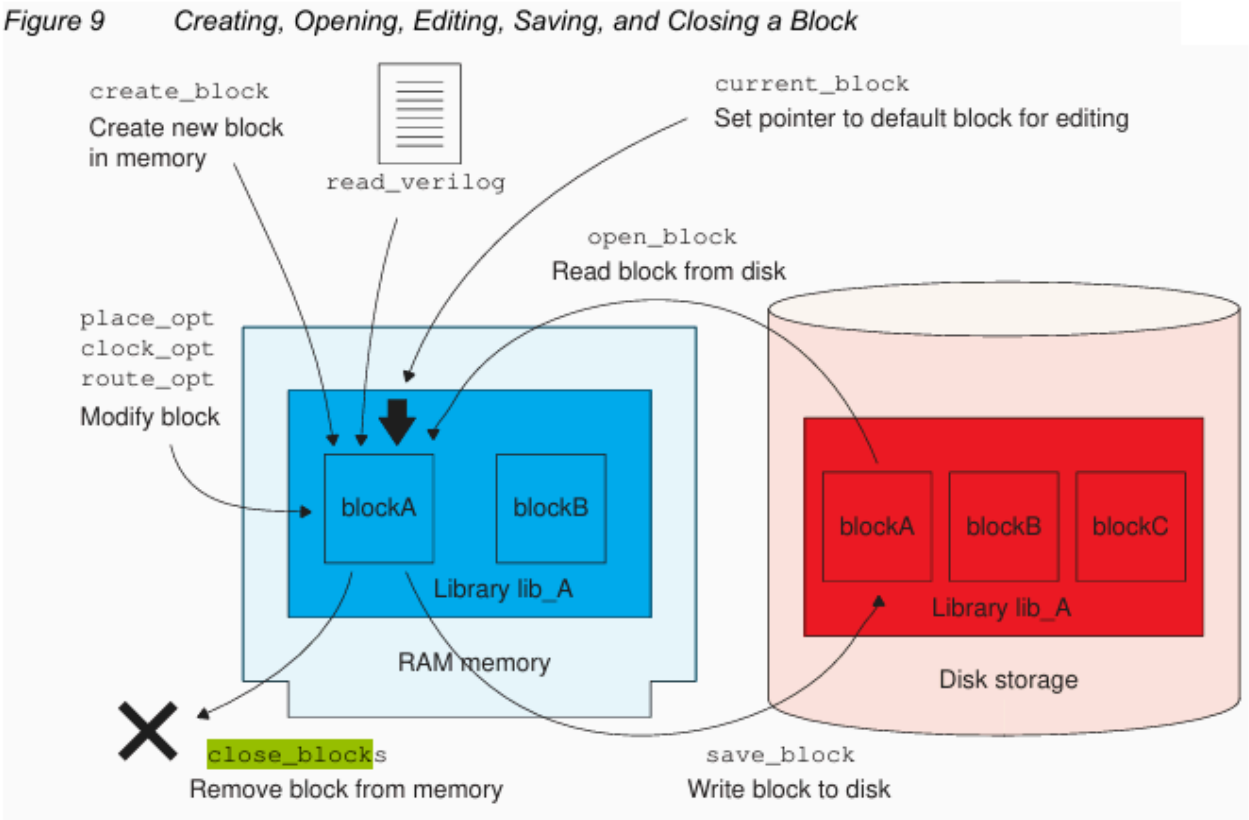


表1列出Design Library Operation

Table 1 Commands That Operate on Design Libraries

Command	Description
close_lib	Closes a library, removing it from memory
copy_lib	Copies a design library on disk
create_lib	Creates a design library in memory
current_lib	Sets or gets the current library
move_lib	Moves a library on disk
open_lib	Opens a saved library for viewing or editing
report_lib	Reports library information
save_lib	Saves a design library to disk
get_libs	Creates a collection of the libraries loaded in memory
report_ref_libs	Reports the reference libraries of a library loaded in memory
set_ref_libs	Sets the reference libraries for a library loaded in memory

表2列出Block Operation

Table 2 Commands That Operate on Blocks

Command	Description
close_blocks	Closes a block, removing it from memory

Table 2 Commands That Operate on Blocks (Continued)

Command	Description
copy_block	Copies a block to a new block in the same or different design library
create_block	Creates a new block in memory
current_block	Sets or gets the current block
get_blocks	Creates a collection of the blocks loaded in memory
link_block	Resolves the references in a block
list_blocks	Lists the blocks (stored on disk) in the specified design libraries
move_block	Moves a block to a new block, design library, or view
open_block	Opens a saved block for viewing or editing
rebind_block	Rebinds the references in a block
remove_blocks	Removes a block from memory and disk
reopen_block	Changes the open mode (read-only or edit) of an opened block
save_block	Saves a block to disk

第03行:link_library

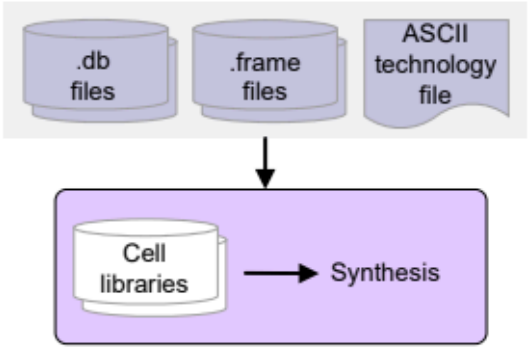
icc2ug.pdf,page69

Library Configuration

Library configuration allows you to specify which vendor libraries to use as reference libraries for the current design. You specify the technology file, physical libraries, and logic libraries by using the `search_path` and `link_library` variables, and then you use the `create_lib` or `set_ref_libs` command to assemble the cell libraries.

During library configuration,

- The IC Compiler II tool automatically calls the Library Manager tool without user intervention to generate cell libraries, as shown in the following figure:



- The tool saves the generated cell libraries to disk and adds them to the reference library list of the design library.
- These cell libraries are the same as when the cell libraries are created during library preparation in the Library Manager tool.

第06行:target_library

icc2ug.pdf,page66

Working With Design Libraries

You can create, open, query, save, or close a design library, using an absolute path, a relative path, or no path, by using the following commands:

- `create_lib`

  This command creates the library in memory and sets it as the current library. When you run this command to create a new design library, you must specify the library

name. Slash (/) and colon (:) characters are not allowed in library names. The following command creates the my_libA library using a relative path:

```
icc2_shell> create_lib ../my_lib_dir/my_libA
{my_libA}
```

Setting Up Reference Libraries

You can specify a reference library list for a design library when you create the design library by using the `-ref_libs` option of the `create_lib` command. You can also change the reference library list at any time by using the `set_ref_libs` command.

Use the following commands to specify, rebind, and report reference libraries:

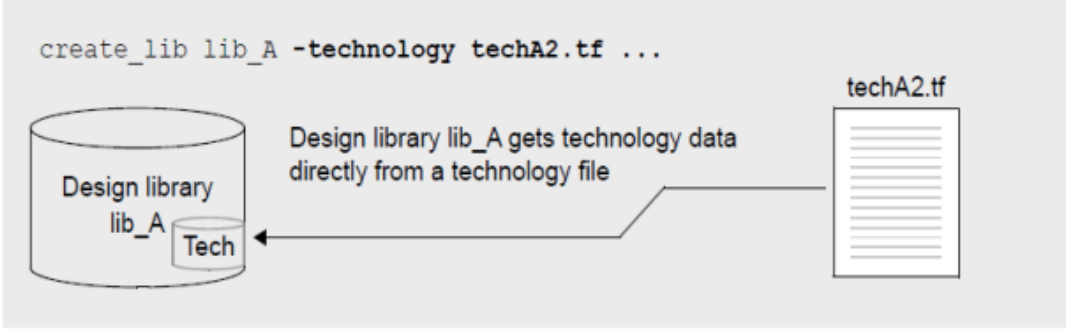
- `create_lib -ref_libs`

  You can specify a relationship between a new design library and its lower-level reference libraries by using the `create_lib` command. For example,

```
icc2_shell> create_lib lib_B \
   -ref_libs {../LIBS/lib_c ../STND/stdhvt.ndm} ...
{lib_B}
```

icc2dm.pdf,page52

Figure 7 Direct Method of Specifying a Library's Technology Data



第10行:read_parasitic_tech

lmug.pdf,page41

Loading Parasitic Parameters

The RC coefficients for a specific technology are represented by TLUPlus models, which are stored in a TLUPlus file or a common technology file. The TLUPlus models enable accurate RC extraction results by including the effects of width, space, density, and temperature on the resistance and capacitance coefficients.

To read the TLUPlus models into the library workspace, use the `read_parasitic_tech` command. You must specify the files that contain the TLUPlus models by using the `-tlup` option. If you specify the files with a relative path or with no path, the library manager uses the search path defined with the `search_path` variable to locate the files.

If the layer names in the TLUPlus or common technology file do not match the layer names in the technology file, you must use the `-layermap` option to specify the layer mapping file. The layer mapping file uses the following format:

```
conducting_layers
tf_metal_layer_name1      ITF_metal_layer_name1
...
tf_metal_layer_namen      ITF_metal_layer_namen

via_layers
tf_via_layer_name1      ITF_via_layer_name1
...
tf_via_layer_namen      ITF_via_layer_namen
```

第13行:read_verilog

icc2dm.pdf,page56,

Blocks

A *block* is a container for physical and functional design data. A typical project consists of the following steps to create and edit each block, at each hierarchical level:

1. Read the design netlist using the `read_verilog` command. This implicitly creates a new block, like using the `create_block` command.
2. Read other design information and constraints by using commands such as `load_upf`, `read_sdc`, and `read_def`.
3. Perform physical implementation by using the `place_opt`, `clock_opt`, and `route_opt` commands.
4. Save the block by using the `save_block` command.

You create, query, and edit blocks in memory. To save a new or edited block to disk, use the `save_block` command; or to read a block from disk into memory, use the `open_block` command, as shown in Figure 9.

第14行:current_design

icc2dm.pdf,page19,page211,

current_design

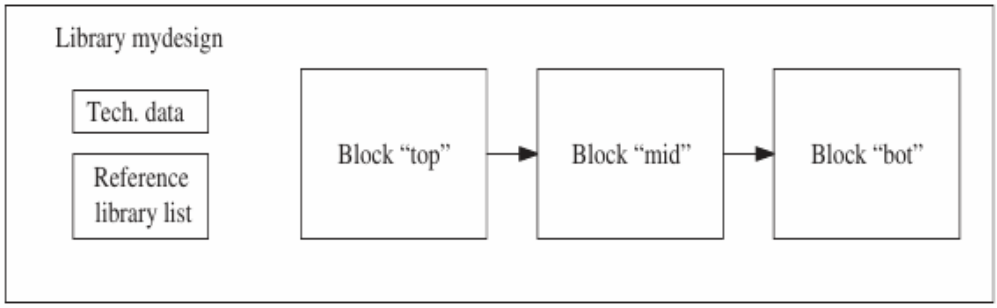
The `current_design` command takes a design object as an argument, such as an object returned by the `get_designs` command. In this context, a *design* is a hierarchical module in the logical netlist. When the `current_design` command is used with an argument, it sets the current block to the block containing the specified design. When used without an argument, it returns the current block, just like the `current_block` command.

```
icc2_shell> get_designs
{design3mp mul32_infer0 design3s design3s_2 design3s_2 design3X}

icc2_shell> current_design design3X
{mylibB:design3X.design}

icc2_shell> current_design
{mylibB:design3X.design}
```

Figure 3 Hierarchical Data Stored in a Single Design Library



第15行:source ../../common/i2c_master_top.sdc

icc2ug.pdf,page225,plot1-9

Prerequisites for Clock Tree Synthesis

This topic details the prerequisites required before running the clock tree synthesis.

Before you run clock tree synthesis on a block, it should meet the following requirements:

- The clock sources are identified with the `create_clock` or `create_generated_clock` commands.
- The block is placed and optimized.

  Use the `check_legality -verbose` command to verify that the placement is legal. Running clock tree synthesis on a block that does not have a legal placement might result in long runtimes and reduced QoR.

  The estimated QoR for the block should meet your requirements before you start clock tree synthesis. This includes acceptable results for

  - Congestion

    If congestion issues are not resolved before clock tree synthesis, the addition of clock trees can increase congestion. If the block is congested, you can rerun the

第16行:save_block

icc2dm.pdf,page61,plot1-6,plot1-10

To save a block on disk using a new or different label, use the `save_block` command with the `-label` option. You can save different versions of a block while maintaining the current block setting in the tool:

```
current_block blkZ
save_block -label v1
save_block -label v2
current_block
```

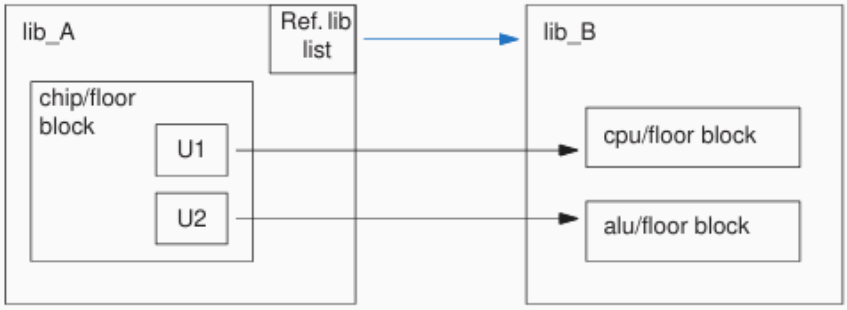
See Also

- Block Views
- Block Naming Conventions

Using Labels in a Hierarchical Design

Suppose that you have a top-level design named chip/floor in a design library named lib_A. The block contains instances U1 and U2, which represent lower-level blocks named cpu/floor and alu/floor in reference library lib_B, as shown in the following figure. All the blocks are at the floorplanned stage.

Figure 10 Hierarchical Design Using Block Names With Labels



第17行:save_lib

icc2dm.pdf,page24

Saving a Design Library

When you create a design library or edit its contents, the changes are stored only in memory. To save a design library to disk, use the `save_lib` command.

```
icc2_shell> current_lib
{lib_A}
icc2_shell> save_lib
Saving library 'lib_A'
1
icc2_shell> save_lib lib_B
Saving library 'lib_B'
1
icc2_shell> save_lib -all
Saving all libraries...
5
```

The `save_lib` command saves all blocks in the design library that have been modified and not yet saved. Be sure to save a new or edited library before you close it.

第18行:close_block

icc2dm.pdf,page75

Closing a Block

When you no longer need access to a block, you can close it with the close_blocks command:

```
icc2_shell> close_blocks MUX2
Closing block 'lib_A:MUX2.design'
1
```

To automatically save a block before closing it:

```
icc2_shell> close_blocks -save MUX2
...
```

Alternatively, you can first save the block by using the save_block command.

To close a block and deliberately discard recent changes to the block:

```
icc2_shell> close_blocks -force MUX2
```

If the block *open count* is 1 or more after you execute the close_lib command, the block remains open.

To remove unwanted blocks from disk, use the remove_blocks command. Note that removed blocks cannot be recovered.

Do not attempt to add, move, or delete block files directly using operating system commands, as doing so can corrupt the database.

第19行:close_lib

icc2dm.pdf,page30

Closing a Design Library

When you no longer need access to data in a library, you can close it. Be sure to save the changes in the library before you close it.

To close a library, use the close_lib command.

```
icc2_shell> current_lib
{lib_A}
icc2_shell> close_lib
Closing library 'lib_A'
1
icc2_shell> close_lib lib_B
Closing library 'lib_B'
1
icc2_shell> close_lib -all
Closing all libraries...
1
```

To deliberately close an edited library and discard the changes:

```
icc2_shell> close_lib -force
Closing library 'lib_A'
1
```

2. step2_floorplan.tcl

第06行:open_lib

icc2dm.pdf,page20,plot2-1

Opening a Design Library

To open an existing design library saved on disk, use the open_lib command:

```
icc2_shell> open_lib my_libA
Information: Loading library file '/usr/lib/StdCells.ndm' (FILE-007)
Information: Loading library file '/usr/lib/RAMs.ndm' (FILE-007)
Information: Loading library file '/usr/lib/PhysicalOnly.ndm' (FILE-007)
{my_libA}
```

The tool opens the specified design library, makes that library the current library, and opens all of its associated reference libraries. Opening a design library means loading it into memory and making its blocks accessible. To reduce memory usage, you can restrict the process, voltage, and temperature combinations loaded into memory from the cell

第07行:copy_block

icc2dm.pdf,page74,plot2-2

Copying a Block

To copy a block to a new block in the same library and view or a different library and view, use the copy_block command. If you do not explicitly specify a view for the source block and the destination block, all available views of the source block are copied to the destination block, and the design view of the destination block is returned. The block's source library does not need to be open. The tool opens the library if it is not already open.

By default, copied blocks are stored in memory only. To list the blocks stored in memory, run the list_blocks command. To save the copied block to disk, set the design.on_disk_operation application option to true before running the copy_block command. The default is false.

第08行:open_block

icc2dm.pdf,page68,page76,plot2-3

Opening a Block

To open an existing saved block for viewing or editing, use the open_block command:

```
icc2_shell> open_block my_lib:MUX2
Opening block 'my_lib:MUX2.design'
{my_lib:MUX2.design}
```

The tool opens the specified block and sets it as the current block. If you specify the library name with the block name (for example, my_lib:MUX2), the tool also opens that library, if it is not already open, and sets it as the current library.

To open a block in read-only mode:

```
icc2_shell> open_block -read my_lib:MUX2
```

第10行:report_clocks
icc2dm.pdf,page183,plot2-4

Table 19 Commands to Query Some Common Design Objects (Continued)

Object	Command	Description
Bounds	`get_bounds`	Returns a collection of placement bounds.
	`get_bound_shapes`	Returns a collection of shapes associated with placement bounds.
	`report_bounds`	Displays information about the placement bounds.
Cells	`get_cells`	Returns a collection of cell instances. By default, the tool uses the logic hierarchy to resolve cell names. To use the physical hierarchy to resolve cell names, use the `-physical_context` option.
	`report_cells`	Displays information about the cell instances.
Clocks	`get_clocks`	Returns a collection of clocks.
	`get_clock_groups`	Returns a collection of clock groups.
	`get_generated_clocks`	Returns a collection of generated clocks.
	`report_clocks`	Reports information about clocks.

第21-22行:set_app_option
icc2dm.pdf,page179

User Default for Application Options

You can optionally set a *user default* for an application option. The user default has higher priority than the *system default* but lower priority than an explicit setting for the option. The user default applies only to the current session and is not saved.

For example, the system default for the `place.coarse.max_density` application option is 0.0, so that value applies to all blocks by default. To set the user default to a different value, use the `set_app_options` command with the `-as_user_default` option:

```
icc2_shell> set_app_options -as_user_default \
   -name place.coarse.max_density -value 0.65
place.coarse.max_density 0.65

icc2_shell> report_app_options place.coarse.max_density
...
Name                      Type   Value  User-default  System-default  ...
------------------------- ------ ------ ------------- --------------- ---
place.coarse.max_density  float  --     0.65          0               ...
------------------------- ------ ------ ------------- --------------- ---
```

The new default, 0.65, applies to all blocks in the current session that do not have an explicit value set.

第23-25行:group_path

dcug.pdf,page399

Creating Path Groups

By default, Design Compiler groups paths based on the clock controlling the endpoint (all paths not associated with a clock are in the default path group). If your design has complex clocking, complex timing requirements, or complex constraints, you can create path groups to focus Design Compiler on specific critical paths in your design.

Use the `group_path` command to create path groups. The `group_path` command allows you to

- Control the optimization of your design
- Optimize near-critical paths
- Optimize all paths

第46行:check_mv_design
icc2mv.pdf,page37

第52行:route_global
icc2ug.pdf,page409

Global Routing

Before you run global routing,

- Define the common routing application options

  For information about the common routing application options, see the `route.common_options` man page.

- Define the global routing application options

  For information about the global routing application options, see the `route.global_options` man page.

To perform standalone global routing, use the `route_global` command. By default, global routing is not timing-driven. For information about enabling timing-driven global routing, see Timing-Driven Global Routing.

The global router divides a block into global routing cells. By default, the width of a global routing cell is the same as the height of a standard cell and is aligned with the standard cell rows.

For each global routing cell, the routing capacity is calculated according to the blockages, pins, and routing tracks inside the cell. Although the nets are not assigned to the actual wire tracks during global routing, the number of nets assigned to each global routing cell is noted. The tool calculates the demand for wire tracks in each global routing cell and reports the overflows, which are the number of wire tracks that are still needed after the tool assigns nets to the available wire tracks in a global routing cell.

For advanced node designs, via density can be a concern. To enable via density modeling, set the `route.global.via_cut_modeling` application option to `true`. The tool then calculates the number of vias in each global routing cell and reports via overflows.

Global routing is done in two phases:

- The initial routing phase (phase 0), in which the tool routes the unconnected nets and calculates the overflow for each global routing cell
- The rerouting phases, in which the tool tries to reduce congestion by ripping up and rerouting nets around global routing cells with overflows

  The tool might perform several rerouting phases. At the end of each rerouting phase, the tool recalculates the overflows. You should see a reduction in the total number of global routing cells with overflow and in the total overflow numbers. The global router stops and exits from the rerouting phase when the congestion is solved or cannot be solved further or after the maximum number of phases has occurred, as defined by the `-effort_level` option. You can force the global router to perform the maximum number of phases based on the specified effort level by setting the `route.global.force_full_effort` application option to `true`. By default, the tool

第53行:report_placement

icc2ug.pdf,page203

Reporting the Placement QoR

The `report_placement` command displays information about the placement QoR for the current block. By default, the command reports the total half-perimeter boundary box wire length for all nets in the block, as well as any placement violations, as shown in the following example:

```
icc2_shell> report_placement
...
...

  Wire length report (all)
  ==================
  wire length in design leon3mp: 12386597.894 microns.
```

3. step3_powerplan.tcl

第07行:set_attribute

icc2/icc2dm.pdf,page188

Settable Attributes

You can set a “settable” attribute for an object by using the `set_attribute` command. For example,

```
icc2_shell> set_attribute -objects [get_nets net16] \
   -name dont_touch -value true
{{net16}}
icc2_shell> get_attribute -objects [get_nets net16] \
   -name dont_touch
true
```

An attribute that is not “settable” is read-only; you cannot change it directly by using the `set_attribute` command. For example, the `bbox` (bounding box) attribute of a net is not settable. However, when you edit or optimize a net, the tool automatically updates the `bbox` attribute to reflect the net’s bounding box.

第19-20行:compile_pg
icc2/icc2mv.pdf,page207

Hierarchical Secondary PG Placement Constraints

You can use secondary PG constraints during design planning to define where secondary supplies are available within the subblocks of a design. The general flow is as follows:

1. Use the `split_constraints` and `commit_block` commands to create the subblocks.
2. Create voltage area shapes by using the `shape_blocks` command.
3. Create PG straps at the top level, then use the `characterize_block_pg` command to create the PG strategies for the subblocks.
4. Create the PG straps at the block level by using the `compile_pg` command.
5. Create secondary PG constraints for each block and commit the constraints.
6. Create secondary PG constraints at the top level and commit the constraints.

When you commit the constraints at the top level of the design, use the `commit_secondary_pg_placement_constraints -commit_subblocks` command. The `-commit_subblocks` option checks all block-level secondary PG constraints and commits any uncommitted constraints. The tool then creates voltage area shapes based on the secondary PG constraints and keeps track of supplies available in or excluded from each voltage area shape.

To list the constraints, use the `report_secondary_pg_placement_constraints -all_blocks` command. To check for consistency and error conditions in specific blocks, use the `check_secondary_pg_placement_constraints -blocks` command with a list of blocks.

For more information, see the *IC Compiler II Design Planning User Guide*.

Lab2 Placement & Route

Script discreption

- Makefile:
  Makefile主要執行3個步驟:

1. icc2_shell ./../scripts/step4_place.tcl
2. icc2_shell ./../scripts/step5_clock_tree_syntesis.tcl
3. icc2_shell ./../scripts/step6_route.tcl

ICC2會先對設計做初始化動作,接著會對各部分做優化步驟:
第一步佈局優化,第二步Clock Tree合成與優化,第三步Clock Route與 post-clock優化,第四步繞線及postroute優化,
以上步驟均完成後,及設計完成可以製成程序

Makefile

```
#ICC2_EXEC = /global/apps/icc2_2019.12/bin/icc2_shell
#ICC2_EXEC = /global/apps/icc2_2022.12-SP5/bin/icc2_shell
ICC2_EXEC = icc2_shell
LOGS_DIR = ./LOG

console:
  $(ICC2_EXEC)
setup:
  test -d $(LOGS_DIR) || mkdir $(LOGS_DIR)
  date > setup
step4_place: setup
  $(ICC2_EXEC) -f ./../scripts/step4_place.tcl | tee -i $(LOGS_DIR)/step4_place.log
  date > step4_place
step5_clock_tree_syntesis: step4_place
  $(ICC2_EXEC) -f ./../scripts/step5_clock_tree_syntesis.tcl | tee -i $(LOGS_DIR)/
/step5_clock_tree_syntesis.log
  date > step5_clock_tree_syntesis
step6_route: step5_clock_tree_syntesis
  $(ICC2_EXEC) -f ./../scripts/step6_route.tcl | tee -i $(LOGS_DIR)/step6_route.log
  date > step6_route
all: step6_route
  date > all

clean:
  ls | grep -v "Makefile" | xargs rm -r
```

- Main TCL:

1. step4_place.tcl

step4_place.tcl | Lab2-1_Script.txt

```
source ../../common/common.tcl
################################################################################Set_Library
set link_library          "$Std_cell_lib $Ram_lib"
set target_library        "$Std_cell_lib $Ram_lib"
#############################################################################
open_lib $ARC_TOP
copy_block -from_block  ${DESIGN_NAME}_3_powerplan_ends -to temp_powerplan_ends
open_block  temp_powerplan_ends
################################################################################General Optimization
set_app_options -name time.disable_recovery_removal_checks -value false
set_app_options -name time.disable_case_analysis -value false
set_app_options -name place.coarse.continue_on_missing_scandef -value true
set_app_options -name opt.common.user_instance_name_prefix -value place
set_app_options -name  place.coarse.congestion_driven_max_util -value 0.6
set_app_options -name  place.coarse.max_density -value 0.2
source ./../scripts/mcmm.tcl
source ./../scripts/set_dont_use.tcl
place_opt
legalize_placement
################################################################################Reports
set_app_option -name time.snapshot_storage_location -value "./"
create_qor_snapshot -name place_qor_snp -significant_digits 4
report_qor_snapshot -name place_qor_snp > ../../reports/place.qor_snapshot.rpt
report_qor > ../../reports/place.qor
report_constraints -all_violators > ../../reports/place.con
report_timing -capacitance -transition_time -input_pins -nets -delay_type max > ../../reports/place.max.tim
report_timing -capacitance -transition_time -input_pins -nets -delay_type min > ../../reports/place.min.tim
################################################################################Save_Block
save_block -as ${DESIGN_NAME}_4_place_ends
save_lib
close_block
close_lib
exit
```

2. step5_clock_tree_syntesis.tcl

step5_clock_tree_syntesis.tcl

```
source ../../common/common.tcl
################################################################################Set_Library
set link_library           "$Std_cell_lib $Ram_lib"
set target_library         "$Std_cell_lib $Ram_lib"
################################################################################Open_Lib
open_lib $ARC_TOP
copy_block -from_block  ${DESIGN_NAME}_4_place_ends -to temp_place_ends
open_block temp_place_ends
################################################################################Set_Clock_Tree_Options
check_legality  -verbose > ../../reports/place_report.rpt
set_ignored_layers -min_routing_layer ${route_min_layer} -max_routing_layer ${route_max_layer}
set_app_options -name cts.compile.enable_cell_relocation -value all
set_app_options -name cts.compile.size_pre_existing_cell_to_cts_references -value true
set_clock_tree_options    -clocks [all_clocks] \
                          -target_skew 0.1
################################################################################Set_Clock_Tree_References
set_lib_cell_purpose -include cts {*/SAEDRVT14_INV_1 */SAEDRVT14_INV_2 */SAEDRVT14_INV_4 */SAEDRVT14_INV_8 */SAEDRVT14_INV_16 */SAEDRVT14_INV_20 \
                    */SAEDRVT14_BUF_2 */SAEDRVT14_BUF_4 */SAEDRVT14_BUF_6 */SAEDRVT14_BUF_8 */SAEDRVT14_BUF_16 */SAEDRVT14_BUF_20 }
################################################################################Set_Clock_Tree_Options
set_clock_uncertainty 0.1 [all_clocks]
create_routing_rule CLK_SPACING -spacings {M2 0.3 M3 0.5 M4 0.7}
set_clock_routing_rules -rules CLK_SPACING -min_routing_layer M2 -max_routing_layer M4
report_clock_settings
set_app_options -name opt.common.user_instance_name_prefix -value clock
source ./../scripts/mcmm.tcl
clock_opt -from build_clock -to build_clock
################################################################################Save_Cell
set_app_option -name time.snapshot_storage_location -value "./"
create_qor_snapshot -name clock_pre_route -significant_digits 4
set_app_option -name cts.compile.enable_global_route -value true
################################################################################Reports
report_qor_snapshot -name clock_pre_route > ../../reports/clock_pre_route.qor_snapshot.rpt
report_qor > ../../reports/clock_pre_route.qor
report_constraints -all_violators > ../../reports/clock_pre_route.con
report_timing -capacitance -transition_time -input_pins -nets -delay_type max > ../../reports/clock_pre_route.max.tim
report_timing -capacitance -transition_time -input_pins -nets -delay_type min > ../../reports/clock_pre_route.min.tim
################################################################################Clock_optimiaztion
set_app_options -name opt.common.user_instance_name_prefix -value clock
clock_opt -from route_clock -to final_opto
################################################################################Reports
report_clock_qor > ../../reports/clock_tree.rpt
report_clock_timing -type skew > ../../reports/clock_timing.rpt
create_qor_snapshot -name clock -significant_digits 4
report_qor_snapshot -name clock > ../../reports/clock.qor_snapshot.rpt
report_qor > ../../reports/clock.qor
report_constraints -all_violators > ../../reports/clock_route.con
report_timing -capacitance -transition_time -input_pins -nets -delay_type max > ../../reports/clock.max.tim
report_timing -capacitance -transition_time -input_pins -nets -delay_type min > ../../reports/clock.min.tim
################################################################################Connecting_power/Ground_Nets_And_Pins
connect_pg_net -net VDD [get_pins -physical_context */VDD]
connect_pg_net -net VSS [get_pins -physical_context */VSS]
################################################################################Save_Block
save_block -as ${DESIGN_NAME}_5_clock_ends
save_lib
close_block
close_lib
exit
```

3. step6_route.tcl

step6_route.tcl

```
source ../../common/common.tcl
################################################################################Set_Library
set link_library          "$Std_cell_lib $Ram_lib"
set target_library        "$Std_cell_lib $Ram_lib"
################################################################################Open_Lib
open_lib $ARC_TOP
copy_block -from_block  ${DESIGN_NAME}_5_clock_ends -to temp_clock_ends
open_block temp_clock_ends
################################################################################Checking
set_ignored_layers -min_routing_layer  ${route_min_layer} -max_routing_layer ${route_max_layer}
report_ignored_layers
################################################################################Route_Optimization
source ./../scripts/mcmm.tcl
set_app_option -name route.detail.ignore_drc -value { }
route_auto
route_opt
################################################################################Connecting_power/Ground_Nets_And_Pins
connect_pg_net -net VDD [get_pins -physical_context */VDD]
connect_pg_net -net VSS [get_pins -physical_context */VSS]
optimize_routes -max_detail_route_iterations 5
check_lvs -max_errors 2000
################################################################################Reports
set_app_option -name time.snapshot_storage_location -value "./"
create_qor_snapshot -name route -significant_digits 4
report_congestion
write_verilog -include {pg_netlist} "$Top_design_pt"
report_qor_snapshot > ../../reports/route.qor_snapshot.rpt
report_qor > ../../reports/route.qor
report_constraints -all_violators > ../../reports/route.con
report_timing -capacitance -transition_time -input_pins -nets -delay_type max > ../../reports/route.max.tim
report_timing -capacitance -transition_time -input_pins -nets -delay_type min > ../../reports/route.min.tim
################################################################################Save_Cell
save_block -as ${DESIGN_NAME}_6_complete
save_lib
################################################################################GDS_OUT (for PT pre-check)
write_verilog -exclude {physical_only_cells} ../../results/${DESIGN_NAME}.out.wo_filler.v
write_sdc -output ../../results/${DESIGN_NAME}.out.wo_filler.sdc
write_def ../../results/${DESIGN_NAME}.out.wo_filler.def
write_parasitics -format SPEF -output ../../results/${DESIGN_NAME}.out.wo_filler.spef
write_gds -design ${DESIGN_NAME}_6_complete \
      -layer_map $Gds_map_file \
      -keep_data_type \
      -fill include \
      -output_pin all \
      -merge_files $Std_cell_gds \
      -long_names \
      ../../results/${DESIGN_NAME}.out.wo_filler.gds
close_block
close_lib
exit
```

- 逐行解釋 Main TCL:

1. step4_place.tcl

第01行:先做基本設定:連結tool path,design library file,constraints file等等...
#設定Ref. Library
第03行:指定library作為目前設計的reference library
第04行:在優化過程中，icc2可以從target_library中選擇任何library cell
#開啟 Library
第06行:開啟已儲存的library來看或編輯
第07行:在同樣的library複製block到新的block
第08行:開啟已儲存的block來看或編輯
#設定優化
第10-11行:設定time相關訊息優化
第12-15行:設定佈局相關設定優化
第16行:設定corner;設定parasitic;產生scenario;讀取constraints information
第17行:指定在優化過程中排除一些在target library的特定cell
第18行:佈局優化
第19行:驗證佈局是否合法,確保放置位置符合所有設計規則及製造限制
#Reports
第21行:優化時間報告儲存位置
第22行:用HTML format產生一個分類的佈局QoR分析報告
第23行:報告佈局QoR information
第24行:報告目前block的QoR information以及統計數據
第25行:報告目前block的logical DRC violations
第26行:報告目前block的最大時序路徑
第26行:報告目前block的最小時序路徑
#儲存Block
第29行:將編輯好的block存到disk
第30行:儲存design library到disk
第31行:關閉block,將block從memory移除
第32行:關閉lib,將design library從memory移除
第33行:離開icc2_shell

2. step5_clock_tree_syntesis.tcl

第01行:先做基本設定:連結tool path,design library file,constraints file等等...
#設定Ref. Library
第03行:指定library作為目前設計的reference library
第04行:在優化過程中，icc2可以從target_library中選擇任何library cell
#開啟 Library
第06行:開啟已儲存的library來看或編輯
第07行:在同樣的library複製block到新的block
第08行:開啟已儲存的block來看或編輯
#設定Clock Tree Options
第10行:驗證佈局是否合法,如果佈局不合法會導致Clock Tree合成時間過長並且減少QoR
第11行:設定忽略最小繞線層與最大繞線層
第12-13行:設定相關Clock Tree Synthesis選項
第14行:使用者調整Clock Tree Synthesis skew量以及latency量,並且用在所有的clock
#設定Clock Tree Reference
第17行:設定library來啟動Clock Tree Synthesis
#設定Clock Tree Options
第20行:預測clock skew量對於所有的clock,來模擬佈局對於clock tree effect
第21行:產生 M2~M4的CLK_space routing rule
第22行:設定M2的CLK_SPACING是最小繞線層,M4的CLK_SPACING是最大繞線層
第23行:報告clock設定
第24行:設定clock優化
第25行:設定corner;設定parasitic;產生scenario;讀取constraints information
第26行:時序優化 從 build_block 到 build_block
#儲存單元
第28行:優化時間報告儲存位置
第29行:用HTML format產生一個分類的時序QoR分析報告
第30行:設定Clock Tree Synthesis compile enable global routing
#報告
第32行:報告在routing之前的佈局QoR information
第33行:報告目前block在routing之前的QoR information以及統計數據
第34行:報告目前block在routing之前的logical DRC violations
第35行:報告目前block在routing之前的最大時序路徑
第36行:報告目前block在routing之前的最小時序路徑
#設定時序優化
第37行:設定時序優化
第38行:時序優化從 route_clock 到 final_opto
#報告
第41行:產生Clock Tree Summary QoR報告等等
第42行:產生single-clock local skew報告
第43行:用HTML format產生一個分類的時序QoR分析報告
第44行:報告在routing之後的佈局QoR information
第45行:報告目前block在routing之後的QoR information以及統計數據
第46行:報告目前block在routing之後的logical DRC violations
第47行:報告目前block在routing之後的最大時序路徑
第48行:報告目前block在routing之後的最小時序路徑
#連接Power/Gnd的Pins & Nets
第50行:連接VDD pins到PG net
第51行:連接VSS pins到PG net
#儲存Block
第53行:將編輯好的block存到disk
第54行:儲存design library到disk
第55行:關閉block,將block從memory移除

第56行:關閉lib,將design library從memory移除
第57行:離開icc2_shell

3. step6_route.tcl

第01行:先做基本設定:連結tool path,design library file,constraints file等等...
#設定Ref. Library
第03行:指定library作為目前設計的reference library
第04行:在優化過程中，icc2可以從target_library中選擇任何library cell
#開啟 Library
第06行:開啟已儲存的library來看或編輯
第07行:在同樣的library複製block到新的block
第08行:開啟已儲存的block來看或編輯
#檢查
第10行:設定忽略最小繞線層與最大繞線層
第11行:報告忽略層
#繞線優化
第13行:設定corner;設定parasitic;產生scenario;讀取constraints information
第14行:設定繞線忽略DRC選項
第15行:自動繞線
第16行:繞線優化
#連接Power/Gnd的Pins & Nets
第18行:連接VDD pins到PG net
第19行:連接VSS pins到PG net
第20行:優化接線跟Via
第21行:檢查LVS
#報告
第23行:優化時間報告儲存位置
第24行:用HTML format產生一個分類的繞線QoR分析報告
第25行:報告congestion情況
第26行:從目前設計寫verilog pg_netlist
第27行:報告繞線QoR information
第28行:報告目前block的繞線QoR information以及統計數據
第29行:報告目前block的繞線的logical DRC violations
第30行:報告目前block的最大繞線路徑時間
第31行:報告目前block的最小繞線路徑時間
#儲存Block
第33行:將編輯好的block存到disk
第34行:儲存design library到disk
#輸出GDS
第36行:從目前設計寫verilog i2c_master_top netlist
第37行:寫入i2c_master_top sdc檔案
第38行:寫入i2c_master_top DEF檔案
第39行:寫入i2c_master_top block level寄生spf檔案
第40行:寫入i2c_master_top GDS檔案
第48行:關閉block,將block從memory移除
第49行:關閉lib,將design library從memory移除
第50行:離開icc2_shell

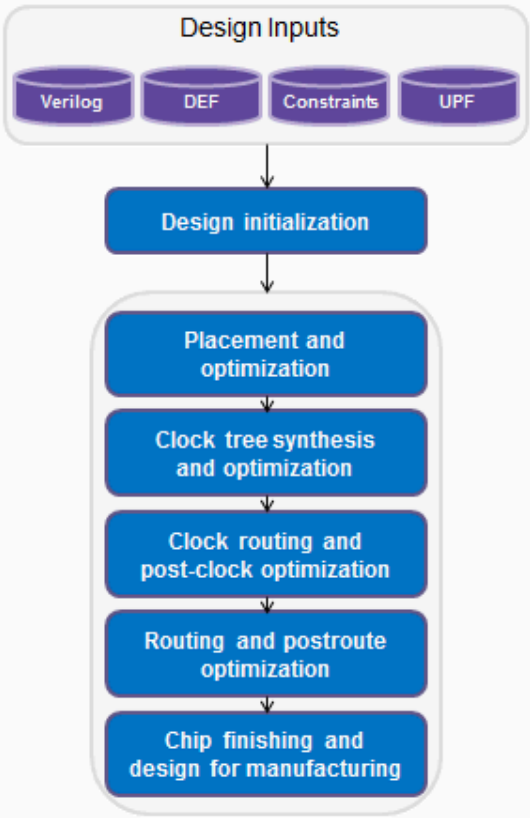
- 補充Main TCL:

下圖顯示ICC2 Placement and Route Overview

Place and Route Design Flow Overview

Figure 1 shows the basic place and route design flow using the IC Compiler II tool.

Figure 1 IC Compiler II Place and Route Flow



1. step4_place.tcl

第10行:set_app_options

icc2dm.pdf,page167

Application Options

You set the tool's *application options* by using the `set_app_options` command. These options control various aspects of tool behavior. For example, to set the maximum allowed coarse placement density for the current block:

```
icc2_shell> set_app_options -name place.coarse.max_density /
   -value 0.6
place.coarse.max_density 0.6
```

第19行:legalize_placement

icc2ug.pdf,page133

Specifying Legalization Settings

The tool performs legalization at various points during the design flow. The following topics describe settings for controlling legalization:

- Minimizing Large Displacements During Legalization
- Optimizing Pin Access During Legalization
- Enabling Advanced PG Net Checks
- Enabling Advanced Legalization Algorithms

第24-27行:report_xxx

icc2ug.pdf,page207

You can use the following commands to generate timing reports for a block:

- `report_timing`

  The `report_timing` command reports the worst-case timing paths for the current block.

- `report_qor`

  The `report_qor` command displays QoR information and statistics for the current block.

- `report_constraints`

  The `report_constraints` command reports the logical DRC violations for the current block.

For more information about generating these reports, see the *Generating Reports* topic in the *IC Compiler II Timing Analysis User Guide*.

2. step5_clock_tree_syntesis.tcl

第10行:check_legality

icc2ug.pdf,page225

Prerequisites for Clock Tree Synthesis

This topic details the prerequisites required before running the clock tree synthesis.

Before you run clock tree synthesis on a block, it should meet the following requirements:

- The clock sources are identified with the `create_clock` or `create_generated_clock` commands.
- The block is placed and optimized.

  Use the `check_legality -verbose` command to verify that the placement is legal. Running clock tree synthesis on a block that does not have a legal placement might result in long runtimes and reduced QoR.

  The estimated QoR for the block should meet your requirements before you start clock tree synthesis. This includes acceptable results for

  - Congestion

    If congestion issues are not resolved before clock tree synthesis, the addition of clock trees can increase congestion. If the block is congested, you can rerun the

第14行:set_clock_tree_options

icc2ug.pdf,page246

Setting Skew and Latency Targets

By default, clock tree synthesis tries to achieve the best skew and latency for all clocks. However, this can lead to area, power, and runtime overhead for low frequency clocks, which have relaxed skew and latency targets.

- To specify a skew target, use the `-target_skew` option with the `set_clock_tree_options` command.
- To specify a latency target, use the `-target_latency` option with the `set_clock_tree_options` command.

By default, when you define skew and latency targets, they apply to all clocks in all corners. To define targets for specific clocks, use the `-clocks` option. To define targets for specific corners, use the `-corners` option.

To report the user-defined skew and latency targets, use the `report_clock_tree_options` command.

To remove user-defined skew or latency targets, use the `remove_clock_tree_options` command. To remove all skew and latency targets, use the `-all` option; otherwise, use the appropriate `-target_skew`, `-target_latency`, `-clocks`, and `-corners` options to remove the specific targets.

第20行:set_clock_uncertainty

icc2ug.pdf,page143

Marking the Clock Networks

If a block does not contain clock trees, you should perform optimization using ideal clocks.

To mark all clock networks as ideal, use the following command:

```
foreach_in_collection mode [all_modes] {
 current_mode $mode
 set_ideal_network [all_fanout -flat -clock_tree]
}
```

To model the clock tree effects for placement, you should also define the uncertainty, latency, and transition constraints for each clock by using the `set_clock_uncertainty`, `set_clock_latency`, and `set_clock_transition` commands.

Before performing clock tree synthesis, you must use the `remove_ideal_network` command to remove the ideal setting on the fanout of the clock trees.

第22行:set_clock_routing_rules
icc2ug.pdf,page171

Specifying Clock-Tree Layer Constraints

To specify clock-tree layer constraints, use the `set_clock_routing_rules` command. To specify the minimum and maximum routing layers, use the `-min_routing_layer` and

`-max_routing_layer` options. Specify the routing layers by using the layer names from the technology file.

By default, the `set_clock_routing_rules` command assigns the specified layer constraints to all clock trees. Table 14 shows the options used to restrict the layer constraints.

Table 14 Restricting Clock-Tree Layer Constraints

To assign a layer constraint to	Use this option
Specific clock trees	`-clocks clocks`
Nets connected to the clock root[1]	`-net_type root`
Nets connected to one or more clock sinks[1]	`-net_type sink`
Internal nets in the clock tree (all nets except the root and sink nets)[1]	`-net_type internal`
Specific clock nets	`-nets nets`

第41行:report_clock_qor
icc2ug.pdf,page316

Generating Clock Tree QoR Reports

To generate clock tree QoR reports, use the `report_clock_qor` command.

The `report_clock_qor` command can generate the following types of reports:

- Summary

  By default, this command reports a summary of the clock tree QoR, which includes the latency, skew, DRC violations, area, and buffer count.

- Latency

To report the longest and shortest path for each clock, use the `-type latency` option.

- DRC violators

  To report the maximum transition and capacitance constraint violators, use the `-type drc_violators` option.

- Robustness

  To report the robustness of each sink, use the `-type robustness` option. The robustness of a sink is the ratio between its latency for the corner for which the report is being generated and its latency for the corner specified by the `-robustness_corner` option. The mode use for both latency values is the mode for which the report is being generated. When you use the`-type robustness` option, you must specify a corresponding robustness corner by using the `-robustness_corner` option.

第42行:report_clock_timing
icc2ug.pdf,page320

Analyzing Clock Timing

The timing characteristics of the clock network are important in any high-performance design. To obtain detailed information about the clock networks in the current block, use the `report_clock_timing` command.

You must use the `-type` option to specify the type of report to generate. The `report_clock_timing` command can generate the following types of reports:

- Single-clock local skew

  To generate a single-clock local skew report, use the `-type skew` option.

- Interclock skew

  To generate an interclock skew report, use the `-type interclock_skew` option.

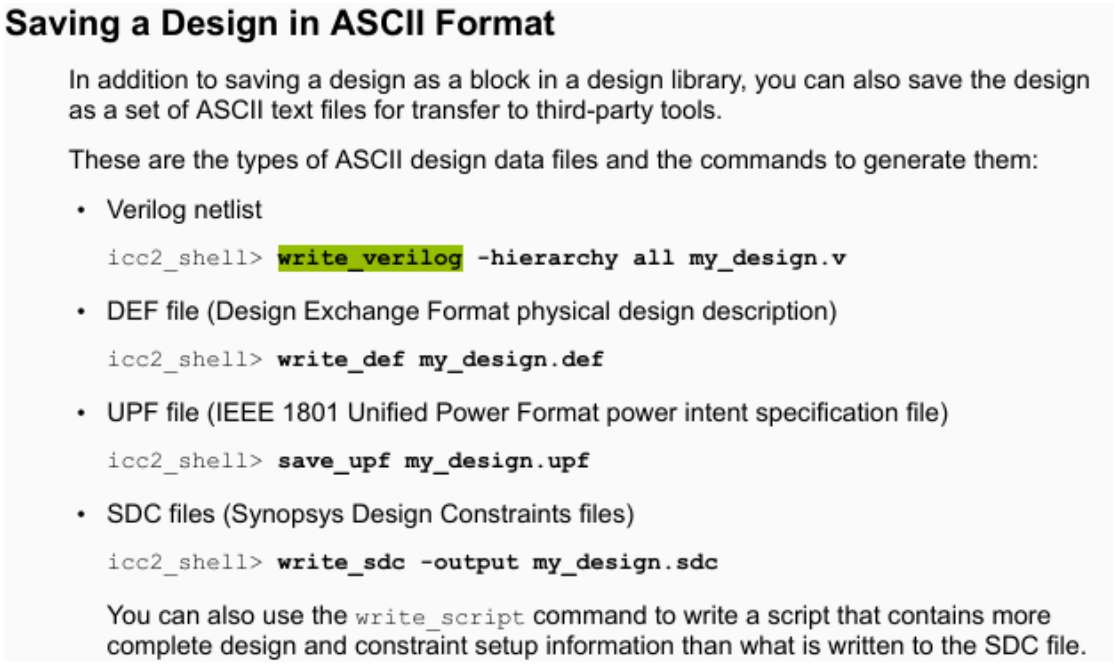
- Latency

  To generate a latency report, use the `-type latency` option.

3. step6_route.tcl

第36-38行:write_xxx

icc2dm.pdf,page134



Saving a Design in ASCII Format

In addition to saving a design as a block in a design library, you can also save the design as a set of ASCII text files for transfer to third-party tools.

These are the types of ASCII design data files and the commands to generate them:

- Verilog netlist

```
icc2_shell> write_verilog -hierarchy all my_design.v
```

- DEF file (Design Exchange Format physical design description)

```
icc2_shell> write_def my_design.def
```

- UPF file (IEEE 1801 Unified Power Format power intent specification file)

```
icc2_shell> save_upf my_design.upf
```

- SDC files (Synopsys Design Constraints files)

```
icc2_shell> write_sdc -output my_design.sdc
```

You can also use the `write_script` command to write a script that contains more complete design and constraint setup information than what is written to the SDC file.

- 參考: https://drive.google.com/drive/folders/1Rszjde4knKzEiAIJcq2k4bWaV6affNwA (https://drive.google.com/drive/folders/1Rszjde4knKzEiAIJcq2k4bWaV6affNwA)

Lab StarRC

Script discreption

首先我們先從Makefile來討論。

```
#ICC2_EXEC = /global/apps/icc2_2019.12/bin/icc2_shell
#ICC2_EXEC = /global/apps/icc2_2022.12/bin/icc2_shell
#DC_EXEC = /global/apps/syn_2022.12/bin/dc_shell
#ICV_EXEC = /global/apps/icv_2022.12-SP5/bin/LINUX.64/icv
LOGS_DIR = ./LOG
TECH_DIR = ../../../SAED14_EDK_LAB/tech/
#STARXTRACT_EXEC = /global/apps/starrc_2022.12-SP5/bin/StarXtract
STARXTRACT_EXEC = StarXtract

################################################################
#standalone StarRC EXTRACTIN setting
################################################################

console:
  $(ICV_EXEC)
setup:
  test -d $(LOGS_DIR) || mkdir $(LOGS_DIR)
  test -d $(RUN_DIR) || mkdir $(RUN_DIR)
  date > setup
gen_StarRC_smc: setup
  rm -rf ./../script/StarRC.smc
  ./../script/gen_StarRC_smc.tcl
  date >  gen_StarRC_smc
gen_StarRC_cmd: setup
  rm -rf ./../script/star_cmd_gate
  ./../script/gen_star_cmd_gate_DEFLEF.tcl
  date >  gen_StarRC_cmd
run_StarRC_cmd: setup
  $(STARXTRACT_EXEC) ./../script/star_cmd_gate
  date >  run_StarRC_cmd
all: setup gen_StarRC_smc gen_StarRC_cmd run_StarRC_cmd
  date >  all
clean:
  ls | grep -v "Makefile" | xargs rm -r
```

圖:Makefile內容

我們會先利用gen_StarRC_smc.tcl和gen_star_cmd_gate_DEFLEF.tcl產生兩個檔案StarRC.smc(裡面設定tempature 、corner的資訊)和Star_cmd_gate(用來跑StarXtract的tcl)

```
CORNER_NAME: fast
OPERATING_TEMPERATURE: 25
TCAD_GRD_FILE: //home/course/u110020015/lab_snps_flow/lab_starRC/script/saed14nm_1p9m_Cmin.nxtgrd

CORNER_NAME: slow
OPERATING_TEMPERATURE: 25
TCAD_GRD_FILE: //home/course/u110020015/lab_snps_flow/lab_starRC/script/saed14nm_1p9m_Cmax.nxtgrd
```

圖:StarRC.smc內容

```
********** SETUP     **********
* Specify block name for parasitic extraction
BLOCK: i2c_master_top
TOP_DEF_FILE: ../../results/i2c_master_top.out.wo_filler.def
* Provide the mapping file in which design layers mapped to process layers
MAPPING_FILE:  /home/course/ee5252/lab_snps_flow/SAED14_EDK_LAB/SAED14_EDK_LAB/tech/star_rc/saed14nm_tf_itf_tluplus.map
LEF_FILE: /home/course/ee5252/lab_snps_flow/SAED14_EDK_LAB/SAED14_EDK_LAB/lib/stdcell_rvt/lef/saed14nm_rvt_1p9m.lef
* Reduction setting fro STA Analysis
REDUCTION: NO_EXTRA_LOOPS
NETS: *
EXTRACTION: RC
DPT: YES
**********     FLOW SELECTION       **********
NUM_CORES: 4
STARRC_DP_STRING:
CORNERS_FILE: ../script/StarRC.smc
DENSITY_BASED_THICKNESS: YES
SELECTED_CORNERS: fast slow
SIMULTANEOUS_MULTI_CORNER: YES
**********     SKIPPING ALL CELLS **********
SKIP_CELLS: !*
**********     PARASITIC OUTPUT (SPEF)       **********
COUPLE_TO_GROUND: NO
COUPLING_ABS_THRESHOLD: 3e-15
COUPLING_REL_THRESHOLD: 0.03
REDUCTION_MAX_DELAY_ERROR: 1e-14
NETLIST_FORMAT: SPEF
* Provide the name of a file to which output parasitic netlist is written
NETLIST_FILE: ./../../results/i2c_master_top.star.wo_filler.spef
* Provide the name of a summary file to which runtime and memory usage is written
SUMMARY_FILE: ./../../results/i2c_master_top.star_sum
* Provide the working directory name to which StarRC internal information is written in binary
STAR_DIRECTORY: ./star_directory
**********     PARASITIC OUTPUT (SPF)       **********
```

圖:Star_cmd_gate內容

接著利用StarXtract來執行Star_cmd_gate就完成這部分lab。

StarXtract 是一款由 Synopsys 公司開發的寄生抽取軟件，屬於 StarRC（寄生抵抗和電容計算）工具的一部分。它專門用於從半導體設計的物理佈局中抽取寄生參數，如電阻、電容和電感，這對於確定電路在實際製造和運行中的表現至關重要。

主要功能和用途：

- 寄生參數抽取：
  StarXtract 能夠從集成電路設計的佈局數據中抽取寄生電阻和電容值。這些值是通過分析導體間的幾何關係以及導體與其周圍環境的相互作用來計算得出的。
- 多層次寄生模型支持：
  它支援多層次的寄生模型，能夠分析從單個元件到完整子系統或整個晶片層面的寄生效應，並將其整合入更大的系統級模型中。
- 高精度和高效率：
  StarXtract 設計用於提供高精度的抽取結果，這對於高性能和高頻率的設計尤其重要，同時它也針對大型設計進行優化，以提高抽取過程的效率。
- 設計驗證和優化：
  抽取得到的寄生參數可用於後續的設計驗證和性能優化，如信號完整性分析、功率整合分析和電磁干擾分析等。

下面是Star_cmd_gate中的指令說明:

- BLOCK: 指定要進行寄生效應抽取的區塊名稱，這裡是 i2c_master_top。

BLOCK

Defines the layout block name that is to be extracted.

Syntax

`BLOCK: block_name`

Arguments

Argument	Description
block_name	The layout name of the block to be extracted Default: none

Description

The usage of the `BLOCK` option depends on the input design database, as follows:

- NDM format Fusion Compiler or IC Compiler II designs

  The `BLOCK` option is mandatory. You can optionally specify a label name, if one exists, to identify the specific block to be extracted, in the format `block_name/label_name`.

- Milkyway designs

  The `BLOCK` option is mandatory. Valid arguments are top-level blocks or fully placed and routed core blocks that exist in the Milkyway library.

- LEF/DEF designs

  The `BLOCK` option is invalid. The block to be extracted is determined by the DESIGN keyword in the DEF file specified by the `TOP_DEF_FILE` command.

- Calibre gate-level and transistor-level flows

  The `BLOCK` option is mandatory. Valid arguments depend on the settings of the `XREF` and `CELL_TYPE` commands. With `CELL_TYPE: LAYOUT`, which is the default, valid arguments are any cell that exists in the annotated GDSII file and layout netlist file (.nl). With `CELL_TYPE:SCHEMATIC` and `XREF: YES`, valid arguments are any valid HCELL from the Calibre compare file (.ixf file).

- Hercules gate-level and transistor-level flows

  The `BLOCK` option is mandatory. Valid arguments depend on the settings of the `XREF` and `CELL_TYPE` commands. With `CELL_TYPE:LAYOUT`, which is the default, valid arguments are any cell that exists in the `WRITE_EXTRACT_VIEW` library from Hercules. With `CELL_TYPE: SCHEMATIC` and `XREF:YES` or `XREF:COMPLETE`, valid arguments are any valid equivalence point from Hercules compare.

圖:StarRCTM User Guide and Command Reference 518頁

- TOP_DEF_FILE: 定義文件的路徑，用於指定抽取寄生效應的頂層設計文件。

TOP_DEF_FILE

Specifies the top-level block design file in DEF format.

Syntax

`TOP_DEF_FILE: def_file`

Arguments

Argument	Description
def_file	The top block design file in DEF format Default: none

Description

This command defines the top-level block for extraction and is mandatory for LEF/DEF flows.

This DEF file can reference macros defined in separate files with the `MACRO_DEF_FILE` command. The standard cell and routing layer definitions should be defined in the accompanying LEF file. Macro blocks appearing in the DEF file specified by the `TOP_DEF_FILE` command are skipped by default.

You can specify gzip files with the `TOP_DEF_FILE` command.

圖:StarRCTM User Guide and Command Reference 1084頁

- MAPPING_FILE: 提供設計層到製程層的映射文件路徑。

MAPPING_FILE

Specifies the file containing physical layer mapping information between the input database and the specified nxtgrd file.

Syntax

`MAPPING_FILE: file_name`

Arguments

Argument	Description
file_name	The mapping file name Default: none

Description

This command is required for all flows. Every connected layer must be mapped.

You can elect to override sheet resistance values specified in the nxtgrd file by specifying new values in the mapping file specified by the `MAPPING_FILE` command.

For simultaneous multicorner flows, you can specify a mapping file for each corner by including mapping file commands in the corner definitions in the corners file. If a corner definition does not include a `MAPPING_FILE` command, a global mapping file must be defined in the top-level command file. If every corner includes a mapping file, a global mapping file is not necessary; if one exists, it is ignored.

All mapping files for simultaneous multicorner flows must be consistent in terms of the number of database and ITF file layers and their mapping relationships. Each mapping file category, such as conductors or vias, should also be consistent. The only allowed variations are the RPSQ value for conductors and the RPV and AREA values for vias.

For more information about mapping files, see Chapter 16, Mapping Files.

圖:StarRCTM User Guide and Command Reference 756頁

- LEF_FILE: 指定 LEF 文件的路徑，這通常包含有關標準元件的信息。

LEF_FILE

Creates a white-space-delimited list of LEF format files containing complete library cell descriptions.

Syntax

```
LEF_FILE: technology_lef library_lef
LEF_FILE: library_lef macro_lef
LEF_FILE: macro_lef custom_lef
```

Arguments

Argument	Description
technology_lef	LEF file that contains the technology information
library_lef	LEF file that contains the cell library information
macro_lef	LEF file that contains the core cell information
custom_lef	LEF file that contains the custom cell and block information

Description

This command, which is mandatory for LEF/DEF flows, can be specified multiple times in a single command file. The order in which the LEF files are specified is very important.

There are three types of LEF files: the technology LEF file, the standard cell LEF file, and the macro LEF file. The technology LEF file must be listed first in the command file. Every layer defined in the technology LEF file must be mapped to an nxtgrd file layer by a mapping file entry. Undefined layers that in a LEF file cause the StarRC tool to issue an error and exit.

If subsequently specified LEF files contain the following technology information, the StarRC tool reads this information and uses it to override previously read values:

- VIA constructs
- WIDTH, DIRECTION, and WIREEXTENSION attributes under a LAYER construct

The tool ignores all other technology information in LEF files that are read after the first technology LEF file.

The standard cell LEF file and the macro LEF file can be listed in any order after the technology LEF file. You do not need to provide LEF descriptions for DEF macros specified in a `MACRO_DEF_FILE` command. Either a LEF or DEF description is required for every member of the list specified by the `SKIP_CELLS` command.

圖:StarRCTM User Guide and Command Reference 742頁

- REDUCTION: 抽取時是否進行減少，例如不添加額外迴路。

REDUCTION

Specifies parasitic netlist reduction options for signal nets.

Syntax

```
REDUCTION:HIGH | YES | NO | LAYER | NO_EXTRA_LOOPS | LAYER_NO_EXTRA_LOOPS
```

Arguments

Argument	Description
HIGH	Performs maximum reduction; the netlist is smaller than with the YES setting
YES (default)	Performs standard reduction
NO	Performs minimal reduction, including removing shorting resistors (if possible) and reducing parallel resistances
LAYER	Similar to YES, but performs reduction on the same layer only, which prevents reducing via nodes
NO_EXTRA_LOOPS	Performs standard reduction and ensures that no resistive loops are introduced, for use with delay calculators that cannot interpret resistive loops. The netlist is 10 to 20 percent larger than with the YES setting. Loops in the parasitic netlist that result from the layout topology are not removed. For example, overlapping metals connected by parallel vias can produce meshes in the parasitic netlist even with this option.
LAYER_NO_EXTRA_LOOPS	Similar to NO_EXTRA_LOOPS, but performs reduction on the same layer only

Description

This command enables the reduction of extracted parasitic devices for signal nets. Providing a reduced netlist might improve the runtime of downstream simulation tools. The degree of reduction is controlled by the REDUCTION_MAX_DELAY_ERROR command. The reduction algorithm preserves point-to-point resistance and total net capacitance.

To specify reduction for power nets, you must enable extraction of power nets by using the POWER_EXTRACT command and specify the type of reduction by using the POWER_REDUCTION command.

If you use the PrimeTime tool for timing analysis, use the NO_EXTRA_LOOPS setting to simplify the resistor network.

To prevent specific nets from being reduced, use the INDESIGN_OPEN_NETS command.

The StarRC tool also provides a standalone reducer that operates on SPEF and DSPF files. For more information, see Standalone Reducer.

圖:StarRCTM User Guide and Command Reference 996頁

- NETS: 指定哪些網路需要抽取，這裡使用 * 表示所有網路。

NETS

Specifies a list of nets to extract.

Syntax

`NETS: net1 net2 ...`

Arguments

Argument	Description
net1 net2 ...	Nets to be extracted, separated by spaces Default: all nets (*)

Description

By default, the StarRC tool extracts all nets, which is the equivalent of the `NETS:*` command. This is the behavior when the `NETS` command is not used in the StarRC command file.

However, if you use the `NETS` command even one time, the tool extracts only the nets that are specified in a `NETS` command. If the `NETS` command appears multiple times, the list of nets to extract is a cumulative list based on the contents of all of the `NETS` commands.

You can also use the `NETS_FILE` command to specify a file that contains a list of `NETS` commands. The effect of the `NETS` command is the same whether it appears directly in the command file or in a file specified by the `NETS_FILE` command.

The following usage notes apply:

- Wildcards are supported: asterisk (*) for all values, question mark (?) for a single character, and exclamation mark (!) for negation.
- If a `NETS:*` command appears anywhere within the command file or in a file specified with the `NETS_FILE` command, the tool extracts all nets.
- A `NETS` command that does not specify at least one net for extraction returns an error for gate-level extraction. An example is the !* argument when used alone. However, if the same `NETS` command specifies at least one net for extraction, that net specification overrides the !* argument.
- Net names that originate from a hierarchical netlist must be fully flattened with the hierarchical separator defined by the `HIERARCHICAL_SEPARATOR` command. Additionally, any reserved character from the input database must be included in this list to allow the use of special characters such as the `BUS_BIT` delimiter.
- Names must be case-sensitive in accordance with the `CASE_SENSITIVE` command.

The StarRC tool does not alter the net information in the input database except to flatten names as required. The case of names in the input database is always preserved in the output netlist. It is important to understand how the place and route tool handles names. If possible, extract names directly from the input database.

Table 78 describes the behavior of the NETS command for different combinations of input that include wildcards.

Table 78 Behavior of Wildcards in the NETS Command

Argument	NETS command behavior
*	Selects all nets (default)
* A B	Selects all nets
* !A	Selects all nets except net A
* !XY*	Selects all nets except nets with names beginning with XY
!* or !* !A or !A	Gate-level extraction: Returns an error , because there must be at least one net for extraction Transistor-level extraction: Generates output with no parasitics
!* A	Selects only net A
!* XY*	Selects only nets with names beginning with XY

For transistor-level extraction, the behavior of the NETS command when the argument list does not select any nets (including settings such as !*, !* !A, and !A) depends on the setting of the NETLIST_INSTANCE_SECTION command, as follows:

- If the NETLIST_INSTANCE_SECTION command is set to SELECTED (the default), the instance section contains instances connected to extracted nets.
- If the command is set to YES, the instance section contains all instances.
- If the command is set to NO, the instance section is not generated.

圖:StarRCTM User Guide and Command Reference 920頁

- EXTRACTION: 抽取類型，這裡是 RC（電阻-電容）。

EXTRACTION

Specifies the type of extraction and the scope of the generated netlist.

Syntax

```
EXTRACTION: RC | C | R | FSCOMPARE | NORC
```

Arguments

Argument	Description
RC (default)	Extracts both parasitic resistor and capacitor devices and merges them into the original database network to produce a consolidated RC network description of the layout in the specified format.
C	Extracts only parasitic capacitor devices and produces a merged parasitic layout network description as a SPICE file. The NETLIST_FORMAT command is ignored for capacitance-only extractions.
R	Extracts only parasitic resistor devices and produces a merged parasitic layout network description in the specified format.
FSCOMPARE	Provides a comparison report of a merged layout network description containing only parasitic capacitors, executes a field solver analysis of the layout, and produces report files that describe the accuracy in a comparison of the two results. When this option is specified, the .fscomptot and .fs_compcoup output comparison files always use the layout net names, regardless of the XREF command setting.
NORC	Retains only *|P and *|I information of the nets that are extracted or netlisted and the instance section with ideal connection for device terminals. Does not retain parasitic resistor or capacitor devices.

Description

The extraction of parasitic devices is performed only on that portion of the layout network defined by the NETS command, terminating each net at the boundary of a skip cell.

When you use the NORC option,

- The resistors and capacitors are not extracted if they are postprocessed with the NETLIST_POSTPROCESS_COMMAND command.
- The tool does not run the REDUCTION command if you use both the EXTRACTION: NORC and REDUCTION commands together. This is because the resistors and capacitors and their node information are not required in the netlist.
- The tool skips opens as they are not required with the EXTRACTION: NORC command.

圖:StarRCTM User Guide and Command Reference 644頁

- DPT: 是否開啟深層次處理。

DPT

Enables extraction for double or multiple patterning processes.

Syntax

```
DPT: YES | NO
```

Arguments

Argument	Description
YES	Uses dielectric constant changes
NO (default)	Performs regular extraction

Description

Models the misalignment effects for multiple patterning technologies. If you set the DPT command to YES, the nxtgrd file must include either the ER_VS_SI_SPACING command or the LATERAL_CAP_SCALING_VS_SPACING command.

圖:StarRCTM User Guide and Command Reference 611頁

- NUM_CORES: 指定用於計算的核心數量。

NUM_CORES

Specifies the number of cores used for distributed processing.

Syntax

`NUM_CORES: number_of_cores`

Arguments

Argument	Description
number_of_cores	The number of cores Default: 1

Description

The `NUM_CORES` command enables distributed processing for extraction and specifies the number of cores to use.

If you specify a value greater than the number of cores available, the StarRC and grdgenxo tools issue a warning and use as many cores as available.

For more information, see Distributed Processing and Using Distributed Processing With the grdgenxo Tool.

Note:

The number of worker processes launched by the StarRC and grdgenxo tools is equal to the setting of the `NUM_CORES` command. If your submission command specifies a smaller number of cores, some cores are reserved but not used. For best results, the `NUM_CORES` command and the submission command should specify the same number of cores.

圖:StarRCTM User Guide and Command Reference 929頁

- STARRC_DP_STRING: 額外的 StarRC 配置參數（未指定具體值）。

STARRC_DP_STRING

Enables automatic submission of distributed processing jobs.

Syntax

```
STARRC_DP_STRING:
      bsub lsf_arguments
    | qsub gridware_arguments
    | list [login_protocol] host1[:n1] [host2[:n2] ... hostm[:nm]]
    | list localhost:num_processes
    | nc run rtda_arguments
    | sbatch -p partition_name
```

Arguments

Argument	Description
lsf_arguments	Arguments for an LSF system
gridware_arguments	Arguments for a Gridware system
login_protocol	Login protocol. Valid values: `rsh` (default) or `ssh` Using the ssh protocol requires additional setup. For more information, see Distributed Processing.
host1, host2 ...	Name of host machine
n1, n2 ...	Number of runs to submit on corresponding host
num_processes	Number of processes on the local host
rtda_arguments	Arguments for a Runtime Design Automation (RTDA) system
`sbatch`	Executes jobs in the default partition queue In the StarRC command file, specify the `STARRC_DP_STRING: sbatch` command only. To use the `-P` option with `sbatch`, see Example 39.

Description

Distributed processing allows you to start a single StarRC run and let the tool automatically submit multiple jobs. You can specify the job submission command by setting the `STARRC_DP_STRING` environment variable or by using the `STARRC_DP_STRING` command in the StarRC command file. If both the environment variable and command are set, the `STARRC_DP_STRING` command in the command file takes precedence.

The control parameters in the submission command are site-specific; contact your system administrator for assistance.

Distributed processing is available for the following computing environments:

- A single host
- A general network with a list of machines
- LSF system
- Gridware system
- Runtime Design Automation (RTDA) system

For more information, see Distributed Processing.

Note:

The number of worker processes launched by the StarRC tool is equal to the setting of the `NUM_CORES` command. If your submission command specifies a larger number of cores, some cores are reserved but not used. For best results, the StarRC `NUM_CORES` command and the submission command should specify the same number of cores.

圖:StarRCTM User Guide and Command Reference 1068-1069頁

- CORNERS_FILE: 指定角落分析的配置文件。

CORNERS_FILE

Specifies a file that contains the definitions of corners available for extraction in the simultaneous multicorner flow.

Syntax

```
CORNERS_FILE: corner_file_name
```

Arguments

Argument	Description
corner_file_name	Name of the file containing corner definitions

Description

The CORNERS_FILE command specifies a file that contains the corner definitions of all corners available to be extracted in the simultaneous multicorner (SMC) flow. To specify which of the defined corners to extract, use the SELECTED_CORNERS command in the StarRC command file. These commands have an effect only if the SMC flow is enabled.

The CORNERS_FILE and STAR_DIRECTORY command arguments must follow these naming conventions:

- If the STAR_DIRECTORY command specifies a relative path, you can use either a relative path or an absolute path in the CORNERS_FILE command. For example:

  ```
  STAR_DIRECTORY: star_work
  CORNERS_FILE: smc_config
  ```

- If the STAR_DIRECTORY command specifies an absolute path, you must use an absolute path in the CORNERS_FILE command. For example:

  ```
  STAR_DIRECTORY: /tmp/star
  CORNERS_FILE: /remote/.../work_directory/smc_config
  ```

You can use the following commands in the corners file for all design databases:

```
CORNER_NAME: name_of_corner
TCAD_GRD_FILE: nxtgrd_path_and_file_name
OPERATING_TEMPERATURE: temperature_in_Celsius

(optional) MAPPING_FILE: map_file_name
(optional) VIA_COVERAGE_OPTION_FILE: via_file
(optional) DENSITY_OUTSIDE_BLOCK: density_value
(optional) 3D_IC_SUBCKT_FILE: filename
```

If you specify the 3D_IC_SUBCKT_FILE command in the StarRC command file and

- Specify the `3D_IC_SUBCKT_FILE` command in the corner file, the tool considers corners specified in the corner file only
- Do not specify the `3D_IC_SUBCKT_FILE` command in the corner file, the tool considers corners specified in the StarRC command file only

RC Scaling for Fusion Compiler or IC Compiler II Designs

The following commands can be used in the corners file only for NDM format designs created by the Fusion Compiler or IC Compiler II tool:

```
(optional) RES_SCALE: res_value
(optional) CAP_SCALE: cap_value
(optional) CC_SCALE: cc_value
```

The default for the scaling parameters is 1.0, which is equivalent to no scaling. The `RES_SCALE` factor applies to all resistors, excluding special resistors such as shorting resistors. The `CC_SCALE` factor applies to all coupling capacitances, excluding coupling capacitances to ground. The `CAP_SCALE` factor applies to all capacitances including total capacitance, excluding ground capacitances.

Coupling capacitances other than coupling capacitances to ground are subject to both the `CC_SCALE` and `CAP_SCALE` factors. Coupling capacitance to ground is adjusted to preserve the total capacitance of the node scaled by the `CAP_SCALE` value. If the capacitance to ground becomes negative after scaling, it is set to zero.

The scaling parameters are used in the following ways:

- During In-Design extraction in the Fusion Compiler or IC Compiler II tool

  The Fusion Compiler or IC Compiler II tool inserts the scaling commands into the corners file based on settings in the Fusion Compiler or IC Compiler II flow.

- During standalone StarRC extraction

  You can use the scaling commands in a corners file to compare standalone StarRC extraction to Fusion Compiler or IC Compiler II In-Design extraction.

Note:

You can use RC scaling in a standalone StarRC extraction run for comparison with IC Compiler II In-Design extraction runs. However, you should not use RC scaling for final signoff extraction runs.

圖:StarRCTM User Guide and Command Reference 558-559頁

- DENSITY_BASED_THICKNESS: 是否基於密度來設定厚度。

DENSITY_BASED_THICKNESS

Enables the calculation of density and thickness variation during extraction.

Syntax

```
DENSITY_BASED_THICKNESS: YES | NO
```

Arguments

Argument	Description
`YES` (default)	Considers density-based thickness variation options as specified in the ITF file
`NO`	Does not consider density-based thickness options

Description

This command enables the calculation of density and thickness variation using the `THICKNESS_VS_DENSITY` or the `POLYNOMIAL_BASED_THICKNESS_VARIATION` commands in the ITF file.

If this command is not specified or is set to `YES`, the StarRC tool issues a warning if thickness variation commands are not specified in the ITF file. The tool does not issue a warning if you set the `DENSITY_BASED_THICKNESS` command to `NO`.

Examples

```
DENSITY_BASED_THICKNESS: YES
```

圖:StarRCTM User Guide and Command Reference 595頁

- SELECTED_CORNERS: 選定的角落，這裡是 fast 和 slow。

SELECTED_CORNERS

Specifies the corners to be extracted in the simultaneous multicorner flow.

Syntax

`SELECTED_CORNERS: name_list`

Arguments

Argument	Description
name_list	The corners to be extracted, separated by spaces

Description

This command specifies the corners to be extracted in the simultaneous multicorner (SMC) flow. The command has an effect only if the SMC flow is enabled.

Every corner listed in the `SELECTED_CORNERS` command must be defined in the corners file specified by the `CORNERS_FILE` command.

To create a SPEF netlist containing more than one corner, use a colon (:) between corner names. The first corner listed is considered the primary corner for output netlist reduction. This feature is valid only for transistor-level flows.

The name of the netlist file for a given corner is the base file name specified in the `NETLIST_FILE` command, with the corner name appended to it.

Examples

Consider the following definition of selected corners:

```
SELECTED_CORNERS: C1 C3 C4
```

In this case, three netlists are created. The first netlist contains parasitic data from C1, the second netlist contains data from C3, and the third netlist contains data from C4.

圖:StarRCTM User Guide and Command Reference 1023頁

- SIMULTANEOUS_MULTI_CORNER: 是否同時進行多角落分析。

SIMULTANEOUS_MULTI_CORNER

Enables the simultaneous multicorner (SMC) flow.

Syntax

```
SIMULTANEOUS_MULTI_CORNER: YES | NO
```

Arguments

Argument	Description
YES (default)	Enables the simultaneous multicorner flow
NO	Uses the single-corner extraction flow

Description

Simultaneous multicorner (SMC) extraction optimizes the efficient extraction of multiple process and temperature corners for a single design. The SMC flow is enabled by default for both gate-level and transistor-level extraction.

To use simultaneous multicorner extraction, the CORNERS_FILE and SELECTED_CORNERS commands must also be set. Table 83 summarizes the effect of StarRC command settings on the extraction mode.

Table 83 Effect of Command File Settings on SMC Extraction

SIMULTANEOUS_MULTI_CORNER command	CORNERS_FILE and SELECTED_CORNERS commands	Extraction
YES	present	SMC
not present	present	SMC
NO	not present	single-corner
not present	not present	single-corner
YES	not present (either or both)	error
NO	present (either or both)	error

For each corner, the corners file must define the corner name, the process corner (through the nxtgrd file), and the operating temperature. During SMC extraction,

- The StarRC tool uses the nxtgrd files specified in the corners file and ignores any other TCAD_GRD_FILE commands in the command file
- The tool uses the operating temperatures specified in the corners file and ignores any other OPERATING_TEMPERATURE commands in the command file

Using the Field Solver Flows With Simultaneous Multicorner Extraction

If you use the SMC flow with the EXTRACTION: FSCOMPARE command, the StarRC tool generates the total and coupling capacitance report files for each corner with a unique nxtgrd file specified by the SELECTED_CORNERS command. If multiple corners with the same nxtgrd file are selected, the tool generates a report only for the first corner because capacitance does not depend on the operating temperature.

You can use the FS_EXTRACT_NETS command in the simultaneous multicorner flow. If you specify nets with the SELECTED_CORNERS and FS_EXTRACT_NETS commands, the field solver generates netlist files for the different corners.

圖:StarRCTM User Guide and Command Reference 1033-1034頁

- SKIP_CELLS: 設置忽略的元件，!* 表示忽略所有元件。

SKIP_CELLS

Creates a white-space-delimited list of cells to skip during extraction.

Syntax

`SKIP_CELLS: cell1 cell2 … cellN`

Arguments

Argument	Description
cell1 cell2 … cellN	A list of cells to be skipped during extraction. All instances of these cells are skipped. Default: *

Description

The SKIP_CELLS command creates a white-space-delimited list of cells for to skip during extraction. This command can be specified multiple times in a single command file. The asterisk (*), exclamation mark (!), and question mark (?) wildcard characters are acceptable. Case sensitivity for selection purposes is determined by the value of the CASE_SENSITIVE command, but the netlist always retains the case of the original input database.

Skip cells are typically cells with their own timing models, which can later be applied, along with the StarRC parasitic netlist, to perform a timing analysis. Skip cells should contain labeled or otherwise annotated pin shapes.

The default for all extraction flows is to skip all lower-level blocks in the input database, so any macro blocks without timing models must be negated from the list.

To skip only specific instances of a cell, use the SKIP_INSTANCES command.

The translate.sum file, located in the summary directory, contains the names of cells that are skipped during extraction because they appear in a SKIP_CELLS or SKIP_PCELLS command. Skipped instances are not reported in this file.

Note:

When you use the CALIBRE_PDBA_FILE command, the SKIP_CELLS command might fail because the DFM properties annotation can promote the instance hierarchy.

圖:StarRCTM User Guide and Command Reference 1043頁

- COUPLE_TO_GROUND: 是否將耦合效應接地。

COUPLE_TO_GROUND

Specifies whether coupling capacitances are retained or lumped to ground.

Syntax

```
COUPLE_TO_GROUND: YES | NO | YES RETAIN_GATE_CONTACT_COUPLING
                | YES RETAIN_GATE_CONTACT_AND_DIFFUSION_COUPLING
```

Arguments

Argument	Description
YES (default)	Grounds all coupling capacitors and applies scaling factor
NO	Retains coupling capacitors if they meet threshold settings
YES RETAIN_GATE_CONTACT_COUPLING	Retains coupling capacitances between the gate and the via or trench contact. Gate, trench contact, and diffusion layers are identified by the LAYER_TYPE declaration in the ITF file. You must also set the EXTRACT_VIA_CAPS command to YES. Coupling capacitance thresholds do not apply. Valid for IC Validator, Hercules, and Calibre flows.
YES RETAIN_GATE_CONTACT_AND_DIFFUSION_COUPLING	Retains coupling capacitances between the gate and the diffusion layer in addition to the coupling capacitances between the gate and the via or trench contact. Gate, trench contact, and diffusion layers are identified by the LAYER_TYPE declaration in the ITF file. You must also set the EXTRACT_VIA_CAPS command to YES and the IGNORE_CAPACITANCE command to NONE, DIFF, or ALL RETAIN_GATE_DIFFUSION_COUPLING. Coupling capacitance thresholds do not apply. Valid for IC Validator, Hercules, and Calibre flows.

Description

By default, or if you set the COUPLE_TO_GROUND command to YES, the StarRC tool grounds all coupling capacitors.

If you set the COUPLE_TO_GROUND command to NO, the tool evaluates the coupling capacitances based on the settings of the COUPLING_ABS_THRESHOLD and COUPLING_REL_THRESHOLD commands. As a result, coupling capacitances might be retained or grounded, depending on the capacitance value.

The tool scales all grounded capacitances by the factor specified by the COUPLING_MULTIPLIER command, which approximates the crosstalk effects that are lost by grounding the coupling capacitors.

You can retain gate-to-contact capacitance by using either of the following methods:

- Set the COUPLE_TO_GROUND command to YES RETAIN_GATE_CONTACT_COUPLING (or to YES RETAIN_GATE_CONTACT_AND_DIFFUSION_COUPLING) and the EXTRACT_VIA_CAPS command to YES. In this case, the gate-to-contact capacitance is retained, regardless of the settings of the COUPLING_ABS_THRESHOLD and COUPLING_REL_THRESHOLD commands.
- Set the RETAIN_GATE_CONTACT_COUPLING command to YES and the EXTRACT_VIA_CAPS command to YES.
  - If the COUPLE_TO_GROUND command is set to NO, the gate-to-contact capacitance is retained only if the capacitance meets the thresholds specified by the COUPLING_ABS_THRESHOLD and COUPLING_REL_THRESHOLD commands.
  - If the COUPLE_TO_GROUND command is set to YES, the gate-to-contact capacitance is always retained.

The RETAIN_GATE_CONTACT_COUPLING command takes precedence. If you set the RETAIN_GATE_CONTACT_COUPLING command to NO and the COUPLE_TO_GROUND command to YES RETAIN_GATE_CONTACT_COUPLING, the COUPLE_TO_GROUND setting is reset to YES and the gate-to-contact capacitance is not retained.

By default, the StarRC tool does not retain coupling capacitances for power nets. To retain the gate-to-contact capacitance for power nets, use one of these command settings:

- Set the POWER_EXTRACT command to YES to retain the gate-to-contact capacitances for all signal and power nets.
- Set the POWER_EXTRACT command to DEVICE_LAYERS to retain the gate-to-contact capacitances for all signal nets and all power nets whose layers are specified with the device_layers keyword in a conducting_layers statement in the mapping file.

Note:

If the device SPICE models already contain gate-to-contact capacitances, you would typically avoid capacitance double counting by setting the EXTRACT_VIA_CAPS command to YES IGNORE_GATE_CONTACT_COUPLING. In this case, the tool does not extract the gate-to-contact capacitances; therefore, the commands discussed in this section have no effect on these capacitances.

圖:StarRCTM User Guide and Command Reference 565-566頁

- COUPLING_ABS_THRESHOLD 和 COUPLING_REL_THRESHOLD: 設定耦合的絕對和相對閾值。

COUPLING_ABS_THRESHOLD

Specifies an absolute threshold for grounding coupling capacitors.

Syntax

COUPLING_ABS_THRESHOLD: *threshold*

Arguments

Argument	Description
threshold	Absolute threshold for grounding coupling capacitors Units: farads (F) Default: 3e-15

Description

Specifies an absolute threshold for grounding coupling capacitors.

By default, coupling capacitors are grounded if both of the following conditions are met:

- The ratio of coupling capacitance to each individual net's total capacitance is less than the value specified by the COUPLING_REL_THRESHOLD command.
- The coupling capacitance is less than the value specified by the COUPLING_ABS_THRESHOLD command.

However, if you set the COUPLING_THRESHOLD_OPERATION command to OR, the coupling capacitance is grounded if either of the two conditions is satisfied.

圖:StarRCTM User Guide and Command Reference 573頁

COUPLING_REL_THRESHOLD

Specifies the ratio of coupling capacitance to total capacitance used to determine whether to ground coupling capacitors.

Syntax

`COUPLING_REL_THRESHOLD: threshold`

Arguments

Argument	Description
threshold	Floating-point number between 0 and 1 Default: 0.03

Description

Specifies the ratio of coupling capacitance to total capacitance used as a threshold for grounding coupling capacitors.

Coupling capacitors are grounded if both of the following conditions are met:

- The ratio of coupling capacitance to each individual net's total capacitance is less than the value specified by the `COUPLING_REL_THRESHOLD` command.
- The coupling capacitance is less than the value specified by the `COUPLING_ABS_THRESHOLD` command.

However, if you set the `COUPLING_THRESHOLD_OPERATION` command to `OR`, the coupling capacitance is grounded if either of the two conditions is satisfied.

圖:StarRCTM User Guide and Command Reference 575頁

- REDUCTION_MAX_DELAY_ERROR: 設定最大延遲誤差的閾值。

REDUCTION_MAX_DELAY_ERROR

Specifies the acceptable net delay error tolerance during reduction.

Syntax

`REDUCTION_MAX_DELAY_ERROR: threshold`

Arguments

Argument	Description
threshold	The absolute delay error Units: seconds Default: 1.0 e-12

Description

The `REDUCTION_MAX_DELAY_ERROR` command controls the parasitic reduction operation by specifying the maximum amount of delay error allowed.

The difference in the absolute delay between the original and reduced netlists cannot be greater than the specified threshold.

圖:StarRCTM User Guide and Command Reference 998頁

- NETLIST_FORMAT: 輸出網表的格式，這裡是 SPEF。

NETLIST_FORMAT

Defines the format of the output parasitic netlist.

Syntax

```
NETLIST_FORMAT: SPF | SPEF | OA | NETNAME | STAR | PARAMETERIZED_SPICE
```

Arguments

Argument	Description
SPF	Standard Parasitic Format. Supports only EXTRACTION: RC with COUPLE_TO_GROUND: YES | NO. Supports coupling capacitors.
SPEF	Flexible and compact. All names are mapped internally, reducing netlist size. SPEF prints the D_NET (detailed parasitics) net type in the netlist.
OA	Transistor-level extraction only. Creates an OpenAccess view.
NETNAME	Transistor-level extraction only. Formats internal node names as *netname*:1, *netname*:2, and so on .
STAR	Transistor-level extraction only. A compact format that uses SPICE-like subnode naming conventions.
PARAMETERIZED_SPICE	A parameterized SPICE netlist that enables Monte Carlo analysis by downstream simulators.

圖:StarRCTM User Guide and Command Reference 852頁

- NETLIST_FILE: 指定輸出寄生網表的文件名。
- SUMMARY_FILE: 輸出運行時間和記憶體使用的摘要文件。

SUMMARY_FILE

Specifies the name of the summary file.

Syntax

```
SUMMARY_FILE: file_name
```

Arguments

Argument	Description
file_name	The name of the summary file Default: *block_name*.star_sum

Description

The SUMMARY_FILE command specifies the name of the summary file.

By default, the summary file is located in the run directory and has the name *block_name*.star_sum, where *block_name* is the block specified by the BLOCK command.

You can use the SUMMARY_FILE command to change the name and location of the summary file. This command accepts a path relative to the run directory. However, an absolute path is not permitted because of the possibility of a change in the network file system.

Examples

To create a summary file my_summary.log in the results subdirectory, use the following syntax in the StarRC command file:

```
SUMMARY_FILE: ./results/my_summary.log
```

圖:StarRCTM User Guide and Command Reference 1075頁

- STAR_DIRECTORY: 指定 StarRC 內部資訊的工作目錄。

STAR_DIRECTORY

Sets the star directory name.

Syntax

`STAR_DIRECTORY: directory`

Arguments

Argument	Description
directory	A valid directory name Default: star

Description

The `STAR_DIRECTORY` command specifies a directory that the StarRC tool creates for reports and intermediate files. The command defaults to the string "star." As a result, StarRC documentation often uses the term "star directory" as a generic term. However, you can use any string that follows valid directory naming conventions.

The `STAR_DIRECTORY` command accepts both absolute and relative paths. However, you can specify a relative path only if the directory is below the working directory (the working directory is the directory from which you run the `StarXtract` command). For example:

```
% star_dir/working_dir/other_dir     (incorrect)
% working_dir/other_dir/star_dir     (correct)
```

For use with simultaneous multicorner extraction, the arguments in the `CORNERS_FILE` and `STAR_DIRECTORY` commands must follow these naming conventions:

- If the `STAR_DIRECTORY` command argument is a relative path, you can use either a relative path or an absolute path in the `CORNERS_FILE` command. For example:

  ```
  STAR_DIRECTORY: star
  CORNERS_FILE: smc_config
  ```

- If the `STAR_DIRECTORY` command argument is an absolute path, you must use an absolute path in the `CORNERS_FILE` command. For example:

  ```
  STAR_DIRECTORY: /tmp/star
  CORNERS_FILE: /remote/.../work_directory/smc_config
  ```

圖:StarRCTM User Guide and Command Reference 1065頁

如需更詳細內容設定可以參考 synopsys提供的 StarRC User Guide and Command Reference (https://drive.google.com/drive/folders/1hGrDiKlNjs57P9MnZ5D2zseKvDKL2mAm)

Lab PrimeTime

Script discreption

首先我們先從Makefile來討論。

```
#ICC2_EXEC = /global/apps/icc2_2019.12/bin/icc2_shell
#ICC2_EXEC = /global/apps/icc2_2022.12/bin/icc2_shell
#DC_EXEC = /global/apps/syn_2022.12/bin/dc_shell
#PT_EXEC = /global/apps/pt_2022.12-SP5/bin/pt_shell
PT_EXEC = pt_shell
LOGS_DIR = ./LOG
TECH_DIR = ../../../SAED14_EDK_LAB/tech/

PT_SELECTED_SCENARIO = fast slow

console:
  $(PT_EXEC)
setup:
  test -d pt_workspace | mkdir pt_workspace
  for SCENARIO in ${PT_SELECTED_SCENARIO}; do test -d ./pt_workspace/$$SCENARIO || mkdir ./pt_workspace/$$SCENARIO ; done
  date > setup
gen_pt_cmd: setup
  tclsh ./../script/gen_pt_cmd.tcl
  date > gen_pt_cmd
set_pt_cmd: setup gen_pt_cmd
  for SCENARIO in ${PT_SELECTED_SCENARIO}; do chmod 777 -R ./pt_workspace/$$SCENARIO ; done
  date > set_pt_cmd
run_pt: setup gen_pt_cmd set_pt_cmd
  for SCENARIO in ${PT_SELECTED_SCENARIO}; do $(PT_EXEC) -file ./pt_workspace/$$SCENARIO/run_pt_cmd.$$SCENARIO.tcl -output_log_file ./pt_workspace/$$SCENARIO/run_pt.log.$$SCENARIO  ; done
  date > run_pt
all: setup run_pt
  date > all
clean:
  ls | grep -v "Makefile" | xargs rm -r


```

圖:Makefile

gen_pt_cmd:用來產生run_pt_cmd.fast.tcl 和 run_pt_cmd.slow.tcl，這兩個檔案將會用來跑pt_shell 的tcl。

```
1  #!/usr/bin/tclsh
2  source ../../common/common.tcl
3
4  foreach scenario $PT_SELECTED_SCENARIO  {
5    set fp [open "./pt_workspace/${scenario}/run_pt_cmd.${scenario}.tcl" w+]
6    puts $fp "##########################################################################"
7    puts $fp "# PrimeTime Reference Methodology Script"
8    puts $fp "# Script: pt.tcl"
9    puts $fp "# Version: U-2022.12-SP4 (June 30, 2023)"
10   puts $fp "# Copyright (C) 2007-2023 Synopsys, Inc. All rights reserved."
11   puts $fp "##########################################################################"
12   puts $fp "# Please do not modify the sdir variable."
13   puts $fp "# Doing so may cause script to fail."
14   puts $fp "set sdir \".\" "
15   puts $fp "\n"
16   puts $fp "##########################################################################"
17   puts $fp "#    Source common and pt_setup.tcl File"
18   puts $fp "##########################################################################"
19   puts $fp "source ../../common/common.tcl"
20   puts $fp "# make PT_REPORTS_DIR"
21   puts $fp "file mkdir ./pt_workspace/${scenario}/\$PT_REPORTS_DIR"
22   puts $fp "# make RESULTS_DIR"
23   puts $fp "file mkdir ./pt_workspace/${scenario}/\$PT_RESULTS_DIR"
24   puts $fp "\n"
```

圖:gen_pt_cmd.tcl中部分內容，可以看到第6行創立run_pt_cmd.fast.tcl 和 run_pt_cmd.slow.tcl下面是撰寫內容。

```
1  ##########################################################################
2  # PrimeTime Reference Methodology Script
3  # Script: pt.tcl
4  # Version: U-2022.12-SP4 (June 30, 2023)
5  # Copyright (C) 2007-2023 Synopsys, Inc. All rights reserved.
6  ##########################################################################
7  # Please do not modify the sdir variable.
8  # Doing so may cause script to fail.
9  set sdir "."
10
11
12 ##########################################################################
13 #    Source common and pt_setup.tcl File                                 #
14 ##########################################################################
15 source ../../common/common.tcl
16 # make PT_REPORTS_DIR
17 file mkdir ./pt_workspace/fast/$PT_REPORTS_DIR
18 # make RESULTS_DIR
19 file mkdir ./pt_workspace/fast/$PT_RESULTS_DIR
20
21
22 ##########################################################################
23 #    Search Path, Library and Operating Condition Section                 #
24 ##########################################################################
25 set report_default_significant_digits 3
26 set sh_source_uses_search_path true
27 set search_path ". $search_path "
28
29
30 ##########################################################################
31 #    Netlist Reading Section                                             #
32 ##########################################################################
33 set link_path "* $PT_link_path"
34 read_verilog $PT_NETLIST_FILES
35 current_design $DESIGN_NAME
36 link
```

圖:run_pt_cmd.fast.tcl 1-36行

```tcl
##########################################################################
#    Back Annotation Section                                             #
##########################################################################
if { [info exists PT_PARASITIC_PATHS] && [info exists PT_PARASITIC_FILES] } {
  foreach para_path $PT_PARASITIC_PATHS para_file $PT_PARASITIC_FILES {
    if {[string compare $para_path $DESIGN_NAME] == 0} {
      read_parasitics $para_file
    } else {
      read_parasitics -path $para_path $para_file
    }
  }
}


##########################################################################
#    Reading Constraints Section                                         #
##########################################################################
if  {[info exists PT_CONSTRAINT_FILES]} {
        foreach constraint_file $PT_CONSTRAINT_FILES {
                if {[file extension $constraint_file] eq ".sdc"} {
                        read_sdc -echo $constraint_file
                } else {
                        source -echo $constraint_file
                }
        }
}


##########################################################################
#    Fix ECO Variable Setup                                              #
##########################################################################
set timing_save_pin_arrival_and_slack true

```

圖:run_pt_cmd.fast.tcl 39-72行

```tcl
##############################################################
#    Update_timing and check_timing Section                  #
##############################################################
update_timing -full
check_timing -verbose > ./pt_workspace/fast/$PT_REPORTS_DIR/${DESIGN_NAME}_check_timing.report


##############################################################
#    Report_timing Section                                   #
##############################################################
report_global_timing > ./pt_workspace/fast/$PT_REPORTS_DIR/${DESIGN_NAME}_report_global_timing.report
report_clock -skew -attribute > ./pt_workspace/fast/$PT_REPORTS_DIR/${DESIGN_NAME}_report_clock.report
report_analysis_coverage > ./pt_workspace/fast/$PT_REPORTS_DIR/${DESIGN_NAME}_report_analysis_coverage.report
report_timing -slack_lesser_than 10 -delay min_max -nosplit -input -net  > ./pt_workspace/fast/$PT_REPORTS_DIR/${DESIGN_NAME}_report_timing.report
report_timing -slack_lesser_than 0.0 -delay min_max -nosplit -input -net  > ./pt_workspace/fast/$PT_REPORTS_DIR/${DESIGN_NAME}_report_timing_violation.report


##############################################################
#    Fix ECO Power Cell Downsize Section                     #
##############################################################
fix_eco_power -pba_mode none -verbose


##############################################################
#   write SDF for Post-sim                                   #
##############################################################
write_sdf ./../../results/i2c_master_top.fast.sdf


##############################################################
#    Save PT Session                                         #
##############################################################
save_session ./pt_workspace/fast/$PT_SESSION_DIR

```

圖:run_pt_cmd.fast.tcl 73-106行

```
108  ###################################################################
109  #    Copy the parasitics_command.log and pt_shell_command.log     #
110  ###################################################################
111  exec /bin/cp -r parasitics_command.log ./pt_workspace/fast/parasitics_command.log.fast
112  exit
113
```

圖:run_pt_cmd.fast.tcl 108-113行

set_pt_cmd:用於設定檔案系統權限的 Shell 腳本

- for SCENARIO in $(PT_SELECTED_SCENARIO); do ... done：
  這是一個 for 循環，它會遍歷 PT_SELECTED_SCENARIO 變量中列出的每個場景。PT_SELECTED_SCENARIO 被設定為 fast slow，意味著此循環會先對 fast 場景進行操作，然後是 slow 場景。

- chmod 777 -R ./pt_workspace/$$SCENARIO：
  - chmod 是改變檔案或目錄權限的命令。
  - 777 是權限設定，其中每個數字代表不同的權限級別。7 表示讀、寫和執行權限。這裡的 777 表示對檔案或目錄的所有者、所屬組和其他使用者都賦予讀、寫和執行權限。
  - -R 表示遞迴地應用權限變更，也就是不僅對指定的目錄，而且對其內所有子目錄和檔案都設置相同的權限。
  - ./pt_workspace/$$SCENARIO 是目標目錄的路徑

接者利用pt_shell來執行run_pt_cmd.fast.tcl 和 run_pt_cmd.slow.tcl就完成這部分lab。

pt_shell 是指 PrimeTime Shell，它是由 Synopsys 公司開發的一種軟件工具，用於集成電路（IC）設計的時序分析。PrimeTime 是業界廣泛使用的靜態時序分析工具，它可以精確地分析和驗證數字集成電路設計在不同製程、電壓和溫度條件下的時序。

主要功能

- 時序分析：
  PrimeTime 可以計算和驗證設計的時序，確保設計在特定的製程條件下達到預期的時鐘頻率，並且滿足設定的時序要求。
- 報告生成：
  它可以生成詳細的時序報告，這些報告包含了關鍵路徑、時序違規等重要信息，這對於時序優化和問題診斷至關重要。
- 時鐘樹分析：
  軟件提供時鐘樹分析工具，幫助設計師優化時鐘信號在整個芯片上的分配，以減少時鐘偏斜和提高整體性能。
- 功耗分析：
  除了時序分析，PrimeTime 也支援對設計功耗的計算和分析，這對於功耗敏感的應用非常重要。

使用方式
pt_shell 通常在命令行界面中運行，並可通過腳本（通常是 Tcl 腳本）自動化復雜的時序分析流程，例如本次lab利用run_pt_cmd.fast.tcl 和 run_pt_cmd.slow.tcl，以下是run_pt_cmd.fast.tcl(slow類似就不特別說明)中的指令說明:

- set sdir "." :
  設定一個名為 sdir 的變量，其值為當前目錄（.）。
- source ../../common/common.tcl：
  載入上層目錄中 common 子目錄下的 common.tcl 檔案。
- file mkdir ./pt_workspace/fast/\(PT_REPORTS_DIR 和 file mkdir ./pt_workspace/fast/\)PT_RESULTS_DIR：
  創建目錄，以存放報告和結果檔案。使用的路徑依賴於環境變數。
- set report_default_significant_digits 3：
  設定報告中顯示的數字精度為三位有效數字。
- set sh_source_uses_search_path true：
  允許搜索路徑功能，使得 PrimeTime 在尋找檔案時會遵循設定的搜索路徑。
- set search_path ". $search_path " ：
  更新搜索路徑，將當前目錄添加到現有的搜索路徑中。
- read_verilog $PT_NETLIST_FILES：
  讀取先前lab產生Verilog 檔案，這些檔案通常包含netlist資訊。

```
84  set PT_NETLIST_FILES "${ROOT_DIR}/results/${DESIGN_NAME}.out.wo_filler.v"
```

圖:common.tcl 84行

- current_design $DESIGN_NAME：
  設定當前的設計名稱，以便於後續的分析。
- link：
  執行連結過程，通常是將讀取的netlist與設計庫中的元件進行對應。
- read_parasitics：
  讀取先前產生的寄生參數檔案，這些檔案包含電路元件之間的寄生效應信息。

```
85  set PT_PARASITIC_FILES   "${ROOT_DIR}/results/${DESIGN_NAME}.star.wo_filler.spef.fast"
86  set PT_PARASITIC_PATHS   "i2c_master_top"
```

圖:common.tcl 85-86行

Threaded Multicore Parasitics Reading

The `read_parasitics` command runs using a separate process launched within PrimeTime. The parasitic reading is performed in parallel with other commands, such as the `read_sdc` command. This parasitic reading process is transparent and requires no explicit control to be enabled. However, you can turn it off by using the `set_host_options -max_cores 1` command. To get the maximum benefit out of the separate parasitic reading process, use the `read_parasitics` command immediately after linking the design, which is the recommended usage for this command rather than using it later in the flow.

圖:Prime time User Guide 75頁

- read_sdc -echo：
  讀取並應用時序約束檔案（SDC），這是控制設計時序要求的主要方式。

```
################################################################################
#
# Design name:  i2c_master_top_7_finished
#
# Created by icc2 write_sdc on Thu Jan  4 19:48:34 2024
#
################################################################################

set sdc_version 2.1
set_units -time ns -resistance MOhm -capacitance fF -voltage V -current uA

################################################################################
#
# Units
# time_unit              : 1e-09
# resistance_unit        : 1000000
# capacitive_load_unit   : 1e-15
# voltage_unit           : 1
# current_unit           : 1e-06
# power_unit             : 1e-12
################################################################################


# Mode: func
# Corner: fast
# Scenario: func_fast

# /remote/testcases/TC102/042022/4196128/LIB_AJ/ICC2/FlowTraining_BE/TEST_AJ/University_LAB/lab_formal_release/common/i2c_master_top.sdc, \
#   line 8
create_clock -name wb_clk_i -period 2 -waveform {0 1} [get_ports {wb_clk_i}]
# Warning: Libcell power domain derates are skipped!

set_clock_uncertainty -setup 0.3 [get_clocks {wb_clk_i}]
set_clock_transition 0.2 [get_clocks {wb_clk_i}]
```

圖:i2c_master_top.sdc file

Reporting Exceptions Source File and Line Number Information

For some constraints, PrimeTime can track and report the source file and line number information for the constraints. Source files are read into the PrimeTime shell using the source or `read_sdc` commands or the `-f` command line option.

To enable the source file name and line number information for the current design constraints, set the `sdc_save_source_file_information` variable to `true`. By default, this variable is set to `false`.

Note:

You can modify the value of this variable only if you have not yet input exceptions.

This implies that you can set the variable to `true` in the setup file or inside pt_shell before applying a timing exception. If at least one exception command has already been successfully input when you try to set this variable, you receive an error and the value remains unchanged. For example:

```
pt_shell> echo $sdc_save_source_file_information
false
pt_shell> set_max_delay -to port1 0.7
pt_shell> set sdc_save_source_file_information true
Error: can't set "sdc_save_source_file_information":
        Use error_info for more info. (CMD-013)
pt_shell> echo $sdc_save_source_file_information
false
```

圖:Prime time User Guide 262頁

- update_timing -full 和 check_timing -verbose：
  更新和檢查時序。這些命令確保時序數據是最新的，並對時序進行詳細檢查。

  Note:

  The `update_timing -full` command causes a complete timing update and overrides the fast timing updates previously described. Avoid invoking `update_timing -full` unless it is necessary.

  圖:Prime time User Guide 669頁

  Run the `check_timing -verbose` command. Ensure that all warnings and errors are understood, verified, and acceptable for timing analysis and model extraction to proceed. For example, you must resolve unconstrained paths, unclocked registers, and mismatching rail voltages.

  圖:Prime time User Guide 925頁

- report_*：
  生成各種時序分析報告，包括全局時序、時鐘偏移、分析覆蓋率和具體的時序報告。

  ### Multivoltage Reporting and Checking

  Several commands are available for reporting the power supply network and checking the network for errors or unusual conditions, including `get_*` commands, `report_*` commands, and `check_timing`.

  圖:Prime time User Guide 456頁

- fix_eco_power -pba_mode none -verbose：
  執行 ECO以優化功耗，這裡設定為不使用 PBA(Path-Based Analysis)模式(可以選none、full)，且verbose 模式時，程式會輸出更為詳細的資訊。
  除此之外，還可以利用此功能來減少漏電流。

  ### Reducing Leakage Power by Cell Swapping

  With the `fix_eco_power` command, you can reduce leakage power with these operations:

  - Cell Swapping Based on Library Cell Names – When the library cells follow the recommended naming convention.
  - Cell Swapping Based on a User-Defined Attribute – When the library cells do not follow the recommended naming convention.

  圖:Prime time User Guide 871頁

- write_sdf ./../../results/i2c_master_top.fast.sdf：
  寫出 SDF 檔案，用於後續的模擬。
  SDF檔案:在進行後仿真（Post-Simulation）時，SDF文件用於將實際的timing信息反向註釋到模擬中，以確保仿真結果能準確地反映實際的電路行為。

- save_session ./pt_workspace/fast/$PT_SESSION_DIR：
  保存當前的 PrimeTime 會話，以便將來恢復或查閱。

  ### Saving a PrimeTime Session

  Before ending a PrimeTime working session, you might want to save the data of the current session. If you need to examine the analysis results later, save the session by using the `save_session` command. To later restore the session, use the `restore_session` command, which takes you to the same point in the analysis.

  Saving and restoring the session is useful in the following situations:

  - You use a script to run a PrimeTime session overnight. The script uses `save_session` after the final `update_timing` command. Later, you can restore the session and examine the results using `gui_start`, `report_delay_calculation`, `report_timing`, and so on.
  - You save the current state of the analysis as a checkpoint, which you can restore later in case of an error or unexpected change in the analysis environment.
  - You save the current session and then restore it multiple times to apply different chip operating modes. This is an alternative to saving and applying the timing data in SDF format.

  圖:Prime time User Guide 66頁

- exec /bin/cp -r parasitics_command.log ./pt_workspace/fast/parasitics_command.log.fast：
  複製日誌文件到指定目錄。

- exit：
  結束腳本執行

如需更詳細內容設定可以參考 synopsys提供的 Prime time User Guide (https://picture.iczhiku.com/resource/eetop/SykdfdRIrLPEQBBX.pdf (https://picture.iczhiku.com/resource/eetop/SykdfdRIrLPEQBBX.pdf))

Lab4 Chip Finishing

Script discreption

首先我們先從Makefile來討論。

```
#ICC2_EXEC = /global/apps/icc2_2019.12/bin/icc2_shell
#ICC2_EXEC = /global/apps/icc2_2022.12/bin/icc2_shell
ICC2_EXEC = icc2_shell
LOGS_DIR = ./LOG

console:
  $(ICC2_EXEC)
setup:
  test -d $(LOGS_DIR) || mkdir $(LOGS_DIR)
  date > setup
step7_finishing: setup
  $(ICC2_EXEC) -f ./../scripts/step7_finishing.tcl | tee -i $(LOGS_DIR)/step7_finishing.log
  date >  step7_finishing
all: setup step7_finishing
  date > all
clean:
  ls | grep -v "Makefile" | xargs rm -r
```

圖:Makefile

接者利用icc2_shell來執行step7_finishing.tcl 就完成這部分lab。
icc2_shell這是由 Synopsys 公司開發的集成電路布局和布線工具的命令行界面。IC Compiler II 是一個高性能的電子設計自動化（EDA）工具，主要用於完成數字集成電路的物理設計階段，包括布局、布線、優化和分析。

以下是step7_finishing.tcl中的指令說明(特別說明與icc2有關指令，terminal 端就不特別說明，像是set ...等):

- open_lib:
  打開頂層庫

  Working With Designs

  In most cases, when you work with a design in the IC Compiler II tool, you use the design view of one of its versions (blocks) from the design library. In this manual, the term *block* is used to refer to a specific design version in the design library, whether it is a top-level design or a subdesign, and the term *design* is used generically to refer to the design being processed. In both cases, unless otherwise specified, the term refers to the design view.

  The IC Compiler II tool reads designs in Verilog format. The Verilog netlist file is a structural or gate-level design in one file. To read a design into the IC Compiler II tool,

  1. Create or open the design library associated with the design.
     - If the design library does not yet exist, use the `create_lib` command to create it.
     - If the design library already exists, use the `open_lib` command to open it.

  圖:IC CompilerTM II Implementation User Guide 74頁

- copy_block、open_block:
  從一個現有塊中複製資料到一個臨時塊，用於進一步的處理、打開指定的塊以進行操作。

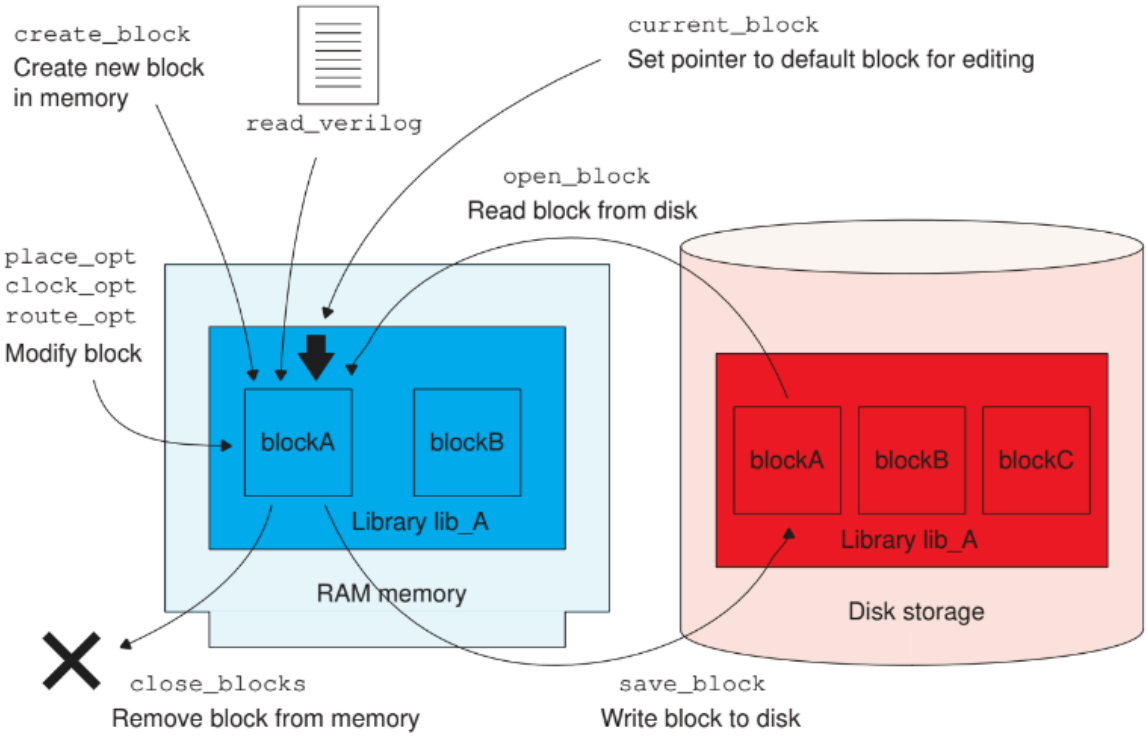


圖:IC CompilerTM II Data Model User Guide 57頁中Figure 9 Creating, Opening, Editing, Saving, and Closing a Block

Command	Description
close_blocks	Closes a block, removing it from memory

Command	Description
copy_block	Copies a block to a new block in the same or different design library
create_block	Creates a new block in memory
current_block	Sets or gets the current block
get_blocks	Creates a collection of the blocks loaded in memory
link_block	Resolves the references in a block
list_blocks	Lists the blocks (stored on disk) in the specified design libraries
move_block	Moves a block to a new block, design library, or view
open_block	Opens a saved block for viewing or editing
rebind_block	Rebinds the references in a block
remove_blocks	Removes a block from memory and disk
reopen_block	Changes the open mode (read-only or edit) of an opened block
save_block	Saves a block to disk

圖:IC CompilerTM II Data Model User Guide 58頁

Copying a Block

To copy a block to a new block in the same library and view or a different library and view, use the `copy_block` command. If you do not explicitly specify a view for the source block and the destination block, all available views of the source block are copied to the destination block, and the design view of the destination block is returned. The block's source library does not need to be open. The tool opens the library if it is not already open.

By default, copied blocks are stored in memory only. To list the blocks stored in memory, run the `list_blocks` command. To save the copied block to disk, set the `design.on_disk_operation` application option to `true` before running the `copy_block` command. The default is `false`.

圖:IC CompilerTM II Data Model User Guide 74頁

Opening a Block

To open an existing saved block for viewing or editing, use the `open_block` command:

```
icc2_shell> open_block my_lib:MUX2
Opening block 'my_lib:MUX2.design'
{my_lib:MUX2.design}
```

The tool opens the specified block and sets it as the current block. If you specify the library name with the block name (for example, `my_lib:MUX2`), the tool also opens that library, if it is not already open, and sets it as the current library.

To open a block in read-only mode:

```
icc2_shell> open_block -read my_lib:MUX2
```

圖:IC CompilerTM II Data Model User Guide 68頁

- add_redundant_vias:
  添加冗餘的通孔以增強連接的可靠性。

Postroute Redundant Via Insertion

To perform postroute redundant via insertion, use the `add_redundant_vias` command.

This command can replace single-cut vias with multiple-cut via arrays, single-cut vias with other single-cut vias that have a different contact code, and multiple-cut via arrays

with different multiple-cut via arrays. During redundant via insertion, the detail router also checks the design rules within the neighboring partition to minimize DRC violations.

By default, the `add_redundant_vias` command inserts redundant vias on all nets. To insert redundant vias only on specific nets, use the `-nets` option to specify the nets. Using net-specific redundant via insertion allows you to further improve the optimized via rate without causing large-scale routing and timing changes.

After the vias are checked and replaced, the detail router rechecks for DRC violations and corrects any violations.

If the percentage of redundant vias is not high enough, you can increase the effort level by using the `-effort` option to get a better redundant via rate. Increasing the effort level to high can increase the redundant via rate by about 3 to 5 percent by shifting the vias to make room for additional vias. However, because high-effort redundant via insertion moves the vias more, it can result in a less lithography-friendly pattern at the 45-nm technology node and below. In this case, you should use concurrent soft-rule-based redundant via insertion to improve the redundant via rate.

You can also try to increase the postroute redundant via rate by setting the `route.detail.optimize_wire_via_effort_level` application option to `high`, which reduces the number of single vias and makes more room for redundant vias by reducing wire length.

After you perform the initial postroute redundant via insertion, set the `route.common.post_detail_route_redundant_via_insertion` application option to enable automatic insertion of redundant vias after subsequent detail routing or ECO routing. This helps to maintain the redundant via rate in your block.

圖:IC CompilerTM II Implementation User Guide 508-509頁

- create_stdcell_fillers:
  創建標準單元填充物，通常用於填補空間以保持版圖的均勻性和改善製造過程。

  Standard Filler Cell Insertion

  The standard filler cell insertion flow uses the `create_stdcell_fillers` command to insert filler cells in the block and the `remove_stdcell_fillers_with_violation` command to perform design rule checking on the filler cells and remove filler cells with violations.

  During filler cell insertion, the `create_stdcell_fillers` command uses the placement legalizer to ensure that the inserted filler cells honor advanced node placement rules and physical constraints such as placement blockages and keepout margins. When the `place.legalize.enable_advanced_legalizer` application option is set to `true`, the command uses the advanced legalization algorithms for 2D rule checking and cell interaction, which can reduce filler cell insertion runtime.

  圖:IC CompilerTM II Implementation User Guide 519頁

  By default, the create_stdcell_fillers command fills all empty space in the horizontal standard-cell rows of the entire block by inserting instances of the filler library cells specified by the -lib_cells option.(You can control the another aspects of the filler cell insertion 參考 IC CompilerTM II Implementation User Guide 521-524頁)
- set_app_options :
  設定應用選項
- lappend create_metal_fill_cmd -select_layers $METAL_FILL_SELECT_LAYER：
  lappend 命令用來將元素添加到一個列表的末尾。這裡，它被用來將選擇層的選項（-select_layers $METAL_FILL_SELECT_LAYER）添加到已存在的命令

create_metal_fill_cmd 中。這樣做的目的是為了擴展原始的命令，使其包含一個額外的選項，指定了哪些層應該被選中進行填充。

- set_attribute and connect_pg_net:
  為電源（VDD）和地（VSS）網設置屬性並連接，這對於整個設計的votage穩定性非常關鍵。

Creating Logical Power and Ground Connections

After you read in the design, you must ensure that there are logical connections between the power and ground nets and the power, ground, and tie-off pins on the cells in your design. If your design does not already have these connections, use the `connect_pg_net` command to create them. This command creates the logical power and ground connections for leaf cells, hierarchical cells, and physical-only cells in both single-voltage and multivoltage designs.

圖:IC CompilerTM II Implementation User Guide 85頁

The `connect_pg_net` command operates in two modes:

- Automatic

  In automatic mode, the command derives all power and ground nets, power and ground pins, tie-off pins, and connections from the UPF specification. If the supply nets do not exist, the tool creates them.

  To create the logical power and ground connections in automatic mode, use the `-automatic` option with the `connect_pg_net` command.

- Manual

  In manual mode, the command makes the connections that you specify. If a specified pin or port has an existing connection, the tool removes the existing connection and then creates the specified connection. The tool verifies that the connections match the power intent. If it finds a mismatch, the tool issues a warning but still creates the connection.

  To create logical power and ground connections in manual mode, use the `-net` option to specify the power or ground net and specify the pins and ports to be connected to that net as an argument to the command. For example, to connect the VDD power net to all pins named vdd and the VSS ground net to all pins named vss, use the following commands:

  ```
  icc2_shell> connect_pg_net -net VDD [get_pins */vdd]
  icc2_shell> connect_pg_net -net VSS [get_pins */vss]
  ```

  To connect PG nets to power and ground pins only, use the `-pg` option. To connect PG nets to tie-off pins only, use the `-tie` option. By default, if neither option is specified, the tool makes connections to all power, ground, and tie-off pins. These options cannot be used with an object list or with the `-create_nets_only` and `-net` options.

  Regardless of the command options used, the tool always creates connections required to build a complete PG netlist structure, such as top-level PG ports and intermediate hierarchical PG pins.

圖:IC CompilerTM II Implementation User Guide 86頁

- check_mv_design:
  檢查多電壓設計的配置是否正確。

After creating the voltage areas, run the `check_mv_design` command to verify that the design does not have any multivoltage violations.

圖:IC CompilerTM II Implementation User Guide 89頁

- check_lvs:
  執行佈局與原理圖（LVS）檢查，確保佈局的準確性。

Performing Layout-Versus-Schematic Checking

To perform layout-versus-schematic (LVS) checking, which checks for inconsistencies in the physical layout, use the `check_lvs` command.

By default, the `check_lvs` command performs the following checks for all signal, clock, and PG nets:

- Shorted nets

  A shorted net occurs when a net shapes from different nets touch or intersect.

  By default, the command

  - Checks for shorts between net shapes in the top-level design, including shapes in top-level blockages

    To disable checking for shapes in top-level blockages, use the `-check_top_level_blockages false` option.

  - Does not check for shorts between net shapes in the top-level design and net-shapes in child cells

    To enable this checking, use the `-check_child_cells true` option. To exclude certain types of child cells from checking, use the `-exclude_child_cell_types`

option to specify one or more of the following cell types: `abstract`, `analog`, `black_box`, `corner`, `cover`, `diode`, `end_cap`, `fill`, `filler`, `flip_chip_driver`, `flip_chip_pad`, `lib_cell`, `macro`, `module`, `pad`, `pad_spacer`, `physical_only`, and `well_tap`.

- Does not check for shorts with zero-spacing blockages

  To enable this checking, use the `-check_zero_spacing_blockages true` option.

Note:

The `check_lvs` command supports only default routing blockages that apply to all net types and have a blockage group ID of 0. If the blockage applies only to specific net types or has a nonzero blockage group ID, the command ignores the blockage. In addition, the command ignores corridor routing blockages (which are created when you use the `-reserve_for_top_level_routing` option with the `create_routing_blockage` command) and boundary routing blockages (which are created when you use the `-boundary_internal` or `-boundary_external` options with the `create_routing_blockage` command).

- Open nets

  An open net occurs when the pins of a net are not connected by its net shapes.

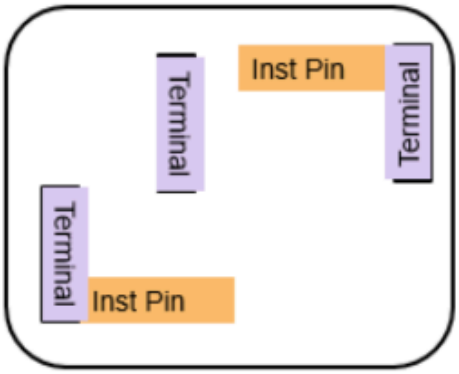
  By default, the command

  - Treats power and ground terminals as unconnected voltage sources

    To treat power and ground terminals as connected, use the `-treat_terminal_as_voltage_source true` option.

    For example, assume your block contains the layout shown in Figure 83. By default, the `check_lvs` command reports two opens for this layout. If you use the `-treat_terminal_as_voltage_source true` option, no opens are reported.

Figure 83 Layout With Power and Ground Terminals



  - Reports open nets as the bounding box of the open net and does not report floating pins

    To report detailed open locations, use the `-open_reporting detailed` option. Note that using this option might increase the runtime. To report the floating pins, use the `-report_floating_pins true` option.

- Floating net shapes

  A floating net shape occurs when a net shape is not physically connecting to a pin of its net.

To perform a subset of these checks, use the `-checks` option to specify one or more of the following checks: `short` (shorted nets), `open` (open nets), and `floating_routes` (floating net shapes). To check only specific nets, use the `-nets` option to specify the nets of interest.

By default, the `check_lvs` command reports a maximum of 20 violations for each type of error. To change this limit, use the `-max_errors` option. To report all violations, specify the maximum number of violations as zero (`-max_errors 0`). You can view the violations reported by the `check_lvs` command in the GUI by using the error browser. For information about using the error browser, see the *IC Compiler II Graphical User Interface User Guide*.

To reduce the runtime required by the `check_lvs` command, enable multithreading by using the `set_host_options` command, as described in Enabling Multicore Processing.

圖:IC CompilerTM II Implementation User Guide 456-458頁

- report_design and report_timing:
  生成設計總結和時序報告，提供關於設計的詳細信息以及時序性能。

Analyzing Timing

The IC Compiler II tool automatically updates timing and delay estimation results when needed. However, you can explicitly update the timing by using the `update_timing` command.

You can use the following commands to generate timing reports for a block:

- `report_timing`

  The `report_timing` command reports the worst-case timing paths for the current block.

- `report_qor`

  The `report_qor` command displays QoR information and statistics for the current block.

- `report_constraints`

  The `report_constraints` command reports the logical DRC violations for the current block.

For more information about generating these reports, see the *Generating Reports* topic in the *IC Compiler II Timing Analysis User Guide*.

圖:IC CompilerTM II Implementation User Guide 207頁

- write_verilog、write_sdc, write_parasitics, write_def, and write_gds:
  輸出Verilog檔案、設計約束（SDC）、寄生參數（SPEF）、設計交換格式文件（DEF）和圖形資料系統格式文件（GDS），這些都是下線前所需的關鍵文件。

Saving a Design in ASCII Format

In addition to saving a design as a block in a design library, you can also save the design as a set of ASCII text files for transfer to third-party tools.

These are the types of ASCII design data files and the commands to generate them:

- Verilog netlist

  ```
  icc2_shell> write_verilog -hierarchy all my_design.v
  ```

- DEF file (Design Exchange Format physical design description)

  ```
  icc2_shell> write_def my_design.def
  ```

- UPF file (IEEE 1801 Unified Power Format power intent specification file)

  ```
  icc2_shell> save_upf my_design.upf
  ```

- SDC files (Synopsys Design Constraints files)

  ```
  icc2_shell> write_sdc -output my_design.sdc
  ```

  You can also use the `write_script` command to write a script that contains more complete design and constraint setup information than what is written to the SDC file.

圖:IC CompilerTM II Data Model User Guide 134頁

Writing Out the Extracted Parasitics

To write out the extracted parasitics information, use the `write_parasitics` command.

Note:

If you are using Fusion Signoff Extraction, the `write_parasitics` command requires a StarRC license.

When you use the `write_parasitics` command, you must specify an output file name by using the `-output` option.

By default, the `write_parasitics` command:

- Generates a SPEF file for each corner.

  The names of these SPEF files are derived based on the output file name you specify and the technology name, temperature, and the scaling factors specified by the `set_extraction_options` command. The tool also generates a file named XX.spef_scenario that lists the scenarios associated with each of the output SPEF files.

  To generate parasitic information:

  - For a specific corner, use the `-corner` option
  - In the Galaxy Parasitic Database (GPD) format, which is a binary file, use the `-format gpd` option

    GPD files can be used in the PrimeTime tool for timing analysis.

- Maps the net names to numbers and uses these numbers when writing the parasitic information for the nets.

  This reduces the file size. To override the mapping of net names, use the `-no_name_mapping` option.

- Generates a SPEF file for the top-level design.

  To generate separate SPEF files for each lower-level block, use the `-hierarchical` option.

When you run the `write_parasitics` command, if the parasitic information is out-of-date, the tool performs extraction. However, if you use the `-hierarchical` option, the tool does not check if extraction has been performed for the lower-level blocks. Therefore, run the `update_timing` command before you use the `write_parasitics` command with the `-hierarchical` option.

圖:IC CompilerTM II Timing Analysis User Guide 148頁

- save_block and save_lib:
  保存當前的塊和庫，以確保所有變更都被正確保存。
- close_block and close_lib:
  關閉打開的塊和庫，以結束當前的設計階段。

如需更詳細內容設定可以參考 synopsys提供的 User Guide:
IC CompilerTM II Multivoltage User Guide
IC CompilerTM II Graphical User Interface User Guide
IC CompilerTM II Data Model User Guide
IC CompilerTM II Timing Analysis User Guide
IC CompilerTM II Implementation User Guide
IC CompilerTM II Design Planning User Guide

(https://drive.google.com/drive/folders/1Rszjde4knKzEiAIJcq2k4bWaV6affNwA
(https://drive.google.com/drive/folders/1Rszjde4knKzEiAIJcq2k4bWaV6affNwA))

Lab IC Validator - DRC

Script discreption

```
#############################################################
#standalone icv drc setting (GDS)
#############################################################
TOP_CELL = i2c_master_top
FORMATE = GDSII
LAYOUT_FILE = $(INPUT_LIBRARY_PATH)/i2c_master_top.gds
LAYER_MAPPING_FILE = $(TECH_DIR)/milkyway/saed14nm_1p9m_gdsout_mw.map
INPUT_LIBRARY_PATH = ../../results
RUN_DIR = ./signoff_drc_run
RUNSET_FILE = $(TECH_DIR)/icv_drc/saed14nm_1p9m_drc_rules.rs
#NUMBER_HOST = 4



console:
  $(ICV_EXEC)
setup:
  test -d $(LOGS_DIR) || mkdir $(LOGS_DIR)
  test -d $(RUN_DIR) || mkdir $(RUN_DIR)
  date > setup
run_icv_DRC: setup
  $(ICV_EXEC) -vue -c $(TOP_CELL) -f $(FORMATE) -lf $(LAYER_MAPPING_FILE) -p $(INPUT_LIBRARY_PATH) -i $(LAYOUT_FILE) -I $(RUN_DIR) $(RUNSET_FILE) | tee -i $(LOGS_DIR)/run_icv_DRC.log
  date >  run_icv_DRC
all: setup run_icv_DRC
  date >  all
clean:
  ls | grep -v "Makefile" | xargs rm -r
```

這個 makefile 主要用於自動化設置環境和執行 ICV 工具進行 Design rule check。

- TOP_CELL：用於指定驗證過程中考慮的主要設計元件。
    - `read_fill_view_from`. Optional. Lists the fill cells that are included on the specified layer when the fill view is read. You can include the top fill cells only, all of the fill cells except the top cells, or all of the fill cells. By default, the IC Validator tool reads all of the fill cells when the fill view is read.
        - `TOP_CELL`. Includes the top fill cells.
- FORMATE：用於指定文件格式。
- LAYOUT_FILE：指定產生出的.gds檔案的路徑。
- LAYER_MAPPING_FILE：用於定義不同製程產生出的不同layer在驗證軟體之間的對應關係。
- INPUT_LIBRARY_PATH：定義包含驗證所需的標準文件庫資料的路徑，這些標準文件庫包含了設計中使用的元件特性。
- RUN_DIR：用於存放驗證過程中生成的產生的文件和中間文件的位置。
- RUNSET_FILE：由台積提供，指定用於 DRC 驗證的運行設定文件（Runset File），包含了進行 DRC 驗證所需要的所有規則和參數設定。

Lab IC Validator – LVS

Script discreption

```
#############################################################
#standalone icv lvs setting (GDS)
#############################################################
GDSII_TOPCELL_NAME = i2c_master_top
GDSII_FILE = ../../results/$(GDSII_TOPCELL_NAME).gds
SPICE_FILE = ./netlist2spice.sp
NETLIST_TOPCELL_NAME = i2c_master_top
LVS_RUNSET_FILE = ../script/saed14nm_1p9m_lvs_runset.rs.tim

console:
  $(ICV_EXEC)
setup:
  test -d $(LOGS_DIR) || mkdir $(LOGS_DIR)
  test -d $(RUN_DIR) || mkdir $(RUN_DIR)
  date > setup
run_icv_nettran: setup
  $(ICV_NETTRAN_EXEC) -verilog ../../results/i2c_master_top.lvs.v.gz -sp ../script/rev_saed14nm_rvt_2.cdl ../script/rev_test.sp -outType SPICE -outName netlist2spice.sp
  date > run_icv_nettran
run_icv_LVS: setup run_icv_nettran
  $(ICV_EXEC) -vue -vueshortALL -create_lvs_short_outputALL -i $(GDSII_FILE) -c $(GDSII_TOPCELL_NAME) -s $(SPICE_FILE) -stc $(NETLIST_TOPCELL_NAME) -sf SPICE $(LVS_RUNSET_FILE)
  date >  run_icv_LVS
all: setup run_icv_nettran run_icv_LVS
  date >  all
clean:
  ls | grep -v "Makefile" | xargs rm -r
```

這個 makefile 主要用於自動化設置環境和執行 ICV 工具進行 LVS check。

- GDSII_TOPCELL_NAME：指定GDSII文件中的 top cell 的名稱。在 LVS 驗證過程中，需要明確指出哪一個 cell 是設計的頂層，以確保驗證工具正確地處理整個 chip 或

miodule。

- GDSII_FILE：指定 GDSII 格式的 layout file 的位置和名稱。
- SPICE_FILE：指定 SPICE 格式的 netlist 檔案的位置和名稱。
- NETLIST_TOPCELL_NAME：指定 SPICE netlist 檔中的 top cell 名稱。進行LVS驗證時，對照 GDSII layout file 的頂層單元。
- LVS_RUNSET_FILE：由台積提供，指定用於 LVS 驗證的運行設定文件（Runset File），包含了進行LVS驗證所需要的所有規則和參數設定。

Lab formality

Script discreption

```
#ICC2_EXEC = /global/apps/icc2_2019.12/bin/icc2_shell
#ICC2_EXEC = /global/apps/icc2_2022.12/bin/icc2_shell
#DC_EXEC = /global/apps/syn_2022.12/bin/dc_shell
#ICV_EXEC = /global/apps/icv_2022.12-SP5/bin/LINUX.64/icv
#FM_EXEC = /global/apps/fm_2022.12-SP6/bin/fm_shell
FM_EXEC = /usr/cad/synopsys/formality/2021.06/bin/fm_shell
LOGS_DIR = ./LOG
TECH_DIR = ../../../SAED14_EDK_LAB/tech/


console:
  $(FM_EXEC)
setup:
  test -d $(LOGS_DIR) || mkdir $(LOGS_DIR)
  date > setup
gen_formality_cmd: setup
  tclsh ./../script/gen_formality_cmd.tcl
  date > gen_formality_cmd
run_formality_cmd: gen_formality_cmd setup
  $(FM_EXEC) -file ../script/run_formality_cmd.tcl | tee -i $(LOGS_DIR)/run_formality.log
  date > run_formality_cmd
all: setup run_formality_cmd
  date >  all
clean:
  ls | grep -v "Makefile" | xargs rm -r
```

這個 makefile 主要用於 gate level 和做完 placement & routing 做完的檔案做比較。Reference design 是 design compiler 所寫出的 netlist 檔，而 Implement design 是 P&R 完產生的 netlist

- read_db "$Std_cell_lib $Ram_lib"
  讀取標準元件和 RAM 的 library。

Argument Lists

When you supply more than one argument for a given Formality command, adhere to Tcl rules. Most publications about Tcl contain extensive discussions about specifying lists of arguments with commands. This section highlights some important concepts.

- Because command arguments and results are represented as strings, lists are also represented as strings, but with a specific structure.
- Lists are typically entered by enclosing a string in braces, as shown in the following example:

```
{file_1 file_2 file_3 file_4}
```

In this example, however, the string inside the braces is equivalent to the following list:

```
[list file_1 file_2 file_3 file_4]
```

Note:
Do not use commas to separate list items.

If you are attempting to perform command or variable substitution, the form with braces does not work. For example, the following command reads a single file that contains designs in the Synopsys internal .db format. The file is located in a directory defined by the `DESIGNS` variable.

```
fm_shell (setup)> read_db $DESIGN/my_file.db
Loading db file '/u/project/designs/my_file.db'
No target library specified, default is WORK
1
fm_shell (setup)>
```

- set hdlin_interface_only ""
  這個指令可以設定特定的 design 作為 black box

Load Design Interfaces

If you know you want an object to be a black box, specify the `hdlin_interface_only` environment variable rather than loading nothing into Formality. Formality benefits from having the pin names and directions supplied by this variable.

Note:
Specify the `hdlin_interface_only` variable before reading in your designs.

To load only the pin names and directions for designs, do one of the following

fm_shell	GUI
Specify: `set hdlin_interface_only "designs"`	1. Click Reference or Implementation. 2. Click Options. 3. Click the Variables tab. 4. In the "Read interface only for these designs" box, enter list of designs. 5. Click OK.

The `hdlin_interface_only` environment variable allows you to load the specified designs as black boxes, even when their models exist. This capability is useful for loading in RAM, intellectual property (IP), and other special models. When you specify `report_black_boxes`, these designs are attributed with an "I" (interface only) to indicate that you specified this variable.

- read_verilog -r /path/to/reference.v -work_library WORK
  從指定的文件讀取 reference design，-r 代表 reference，-work_library 指定使用哪個資料庫

Load the Reference Library

As with the design verification process described in Chapter 4, "Tutorial," you must specify the reference library prior to the implementation library.

To specify the reference library, use one of the following read commands, depending on the library format:

```
fm_shell (library_setup)> read_db -r file_list
fm_shell (library_setup)> read_verilog -r [-tech] file_list
```

As described in the man pages, the `read_db` and `read_verilog` commands have several options that do not apply to library verification. Use the `read_verilog -tech` option if you have a UDP file.

Formality loads the reference library into the `r` container. You cannot rename this container.

In the Formality shell, you represent the design hierarchy by using a `designID` argument. The `designID` argument is a path name whose elements indicate the container (r or i), library, and design name.

Unlike with the design verification process, you do not specify the `set_top` command because multiple top cells are available.

- set_top -auto
  設定 reference 的 top-level design，-auto 代表自動設定

Set Top-Level Design

To set the top-level design for the reference design, do the following

fm_shell

Specify:

```
set_top
[-vhdl_arch name ]
[moduleName | designID | -auto ]
[-parameter value ]
```

If you are elaborating VHDL and you have more than one architecture, use the `-vhdl_architecture` option.

- read_verilog -i /path/to/implementation.v
  從指定的文件讀取 implement design，-i 代表 implement

Load the Implementation Library

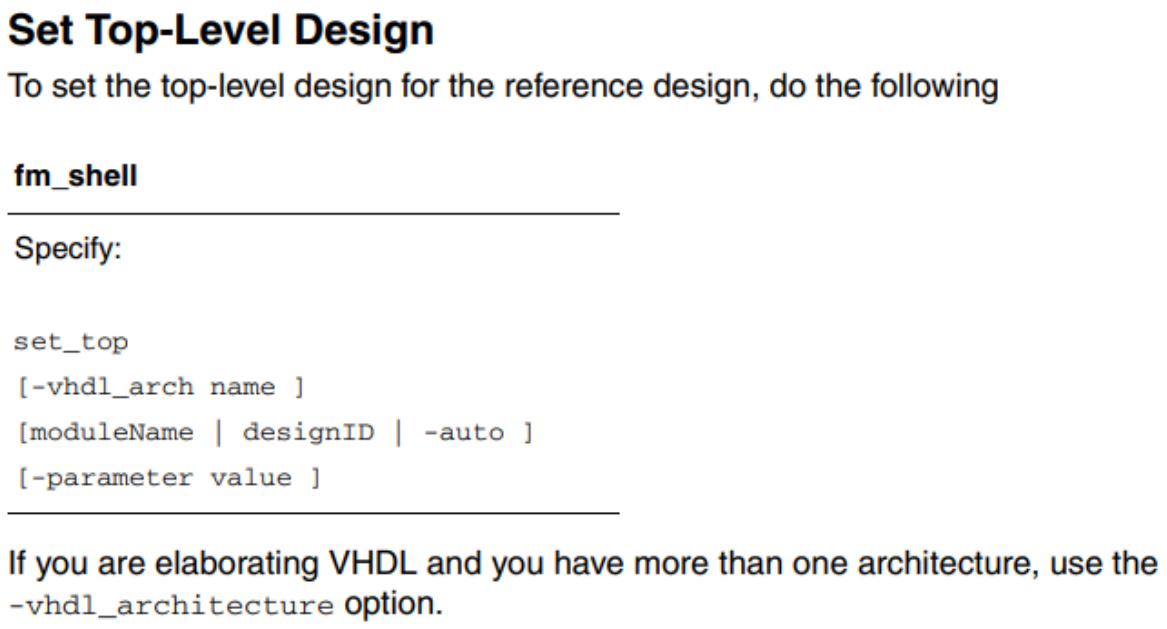
Specify the implementation library in the same manner as described in the previous section, with the exception of the `-r` argument. Instead, use the `-i` argument as follows:

```
fm_shell (library_setup)> read_db -i file_list
fm_shell (library_setup)> read_verilog -i [-tech] file_list
```

Formality loads the reference library into the `i` container. You cannot rename this container.

After you read in the implementation library, Formality performs cell matching to generate the list of cells that are verified. Cells and ports must match by name. The cell list consists of single cell names, and each cell on it is expected to be found in the reference library. If not, it is a nonmatching cell and remains unverified.

- set_top i:WORK/i2c_master_top
  設定 implement 的 top-level design



Set Top-Level Design

To set the top-level design for the reference design, do the following

fm_shell

Specify:

```
set_top
[-vhdl_arch name ]
[moduleName | designID | -auto ]
[-parameter value ]
```

If you are elaborating VHDL and you have more than one architecture, use the `-vhdl_architecture` option.

discussion and observation

IC Design Flow Overview

1. Specification Definition:
Before any design begins, a detailed specification of the IC must be created. This includes defining the functionality, performance targets, power consumption limits, physical size constraints, and environmental conditions under which the IC must operate.

2. Architectural Design:
In this stage, high-level decisions are made about the structure and interconnections of the major components of the IC. This could involve selecting processor cores, memory architectures, and the integration of various IP blocks.

3. RTL Design:
The detailed functionality of the IC is described in a Hardware Description Language (HDL), such as Verilog or VHDL. This description is at the Register Transfer Level (RTL), focusing on how data moves between registers and how it is processed.

Lab Synthesis:
The RTL description is transformed into a gate-level netlist using synthesis tools like Synopsys Design Compiler. Synthesis involves translating RTL code into a form that maps onto the available technology library's logic gates and flip-flops. Optimizations for speed, area, and power are conducted at this stage to meet the initial specifications.

Lab1 Floorplan:
Initial placement of the major components, I/O pads, and defining the power and ground rails. The floorplan must optimize the use of space and provide a framework for efficient routing while considering thermal and manufacturability constraints.

Lab2 Placement & Route:
After floorplanning, the next step is the detailed placement of all the standard cells and the routing of the interconnections between them. Tools like Synopsys IC Compiler are used. The aim is to minimize wire length and congestion, which impacts performance and power consumption. This stage may iterate with synthesis to refine the netlist for better placement and routing results.

Lab StarRC:
Post-placement, the parasitic extraction tool StarRC is used to model the resistance and capacitance of the interconnects. This information is crucial for accurate simulation and timing analysis, as it significantly impacts the IC's performance, particularly at higher frequencies.

Lab PrimeTime:
Performs Static Timing Analysis (STA) to ensure all timing requirements are met, including setup and hold times, and conducts power analysis to verify that power consumption remains within specifications. This analysis helps identify critical timing paths and potential power issues that could affect the chip's reliability and function.

Lab4 Chip Finishing:
The final steps in the design include adding filler cells to ensure density requirements, making adjustments for manufacturability, and producing the GDSII file, which is the final output used for manufacturing the IC.

Lab IC Validator - DRC and LVS:
Design Rule Checking (DRC) ensures that the IC layout adheres to all foundry-specific manufacturing rules, preventing physical defects. Layout Versus Schematic (LVS) checks confirm that the layout accurately reflects the original schematic and the netlist generated post-synthesis. Errors here can lead to functional failures.

Lab Formality:
Formality is used to ensure logical equivalence between the RTL and the synthesized netlist, verifying that transformations during synthesis have not introduced errors.

Finally, note that in this experiment, LVS and DRC errors are likely to occur due to issues with the process files. If the correct files are used, these problems would not occur.