

AI디지털집적회로

Assignment #4

전기전자공학부

202110410

조민우

서론

과제 2에서는 과제 1에서 합성한 RCA 기반 8-bit multiplier를 하위 모듈(submodule)로 포함하는 MAC을 대상으로 hierarchical synthesis를 진행하고 Static Timing Analysis(STA)를 수행하였었다. 또한, 과제 3에서는 과제 2에서 만든 mac8 unit을 하위 모듈로 사용해 systolic array(sa4x4) 모듈을 hierarchical synthesis하고, STA(Static Timing Analysis)를 수행하여 SDF 파일을 생성하고 QuestaSim 시뮬레이터에 back-annotation하여, 합성된 회로의 delay가 반영된 동적 타이밍 시뮬레이션을 진행하였다.

과제 4에서는 과제 3에서 합성한 sa4x4 모듈을 가져와 ICC2 툴을 사용하여 PNR(Placement & Routing) 과정을 진행하려 한다. 이 과정에서 ICC2 툴의 commands와 options들에 익숙해지고, 물리적 layout을 생성하여 Backend 설계 흐름을 익히고자 한다. 또한, 이 PNR 결과를 바탕으로 PPA를 평가 및 파악하고 최적의 설계점을 찾고자 한다.

본격적인 분석에 앞서 마지막 과제인 만큼, 지금까지 진행된 과제들의 흐름을 정리하고 이번 과제가 갖는 의미를 고찰하고자 한다.

구분	Assn1	Assn2	Assn3	Assn4
타겟 모듈	mult8	mac8	sa4x4	sa4x4(from Assn3)
사용 Tool	Design Compiler	Design Compiler, PrimeTime	Design Compiler, PrimeTime, QusetaSim	ICC2
분석 방법	Timing Report Analysis	STA	Dynamic Timing Verification	Floorplanning, Placement, CTS, Routing
검증 방법	Behavior Simulation	STA	Dynamic Gate-level Simulation	DRC, LVS, Post-layout STA, IR drop Analysis, Connectivity Check
Corner 고려	Typical	Slow/Fast/OCV	Slow/Fast/OCV + SDF max delay	Single Corner Only, MCMM (이번 과제는 제외)
설계 방식	Frontend	Frontend	Frontend	Backend

<Fig 1.1 Comparison between Assignments>

과제 1부터 3까지는 Frontend 설계 단계로, RTL 코드의 logic으로 올바른 gate-level netlist로 변환하는 과정에 집중했다. 이 과정에서는 Design Compiler를 이용한 logic synthesis를 통해 standard cell들의 조합으로 회로를 구성하지만, 각 cell들이 실제 chip에서 어디에 배치될지, 어떻게 연결될지는 결정되지 않는다. Timing 분석 역시 wire load model이나 RC estimation 같은 추정치에 의존하며, PrimeTime의 STA도 이론적인 worst-case 분석에 그친다.

Partitioning	Top-level에서 기능별 모듈들을 논리적으로 분할
Chip planning	Floorplanning 이라고도 함, user가 직접 모듈 위치/크기를 결정
Placement	Standard cell들을 chip 영역에 물리적으로 배치
Clock Tree Synthesis (CTS):	Clock distribution network 생성, Clock skew 최소화가 목표
Signal Routing	Standard cell 간 metal wire로 연결
Timing Closure	Setup/Hold time violation 해결

<Fig 1.2 Physical Design Flow>

반면 과제 4는 Backend 설계 단계로, ICC2를 사용하여 netlist를 실제 제조 가능한 물리적 layout으로 구현하는 Place-and-Route 과정을 다룬다. 이 단계에서는 floorplanning을 통해 die size와 I/O 위치를 결정하고, power planning으로 power delivery network를 생성하며, placement로 standard cell들의 물리적 좌표를 할당하고, routing으로 실제 metal layer에 배선을 완성한다. 가장 중요한 차이는 parasitic extraction을 통해 실제 배선의 RC 값을 추출하여 정확한 timing을 분석할 수 있다는 점이며, DRC/LVS 같은 물리적 검증을 통과해야만 실제 fabrication이 가능한 GDSII 파일을 생성할 수 있다.

또한, Frontend에서는 timing이 만족되더라도 Backend에서는 실제 배선 delay와 IR drop, crosstalk 등의 물리적 효과로 인해 추가적인 timing violation이 발생할 수 있어, 이를 해결하기 위한 반복적인 optimization이 필수적이다.

본론

* 이번 과제에서 새롭게 추가된 icc2 directory File들을 우선 분석해보려한다.

2.1.1 icc2/config/

-icc2_common_setup.tcl

해당 파일은 PNR 공정 시작 전에 환경, 라이브러리, input file등의 설정을 담당하는 초기 세팅용 파일이다. Design 이름, include할 SUBMODULE, netlist 파일 경로 등을 설정하고, floorplanning, UPF, MCMM 등의 환경을 구축한다. 아래는 script중 일부 중요 부분을 발췌한 figure들이다.

```

36
37 #set REFERENCE_LIBRARY [list] ;# A list of reference libraries for the design library.
38 set REFERENCE_LIBRARY [list \
39     $STD_CELL_RVT_NDM_DIR/cmos28hpc_1p9m_6x1z1u_13b_rvt_frame.ndm \
40     $STD_CELL_RVT_NDM_DIR/cmos28hpc_1p9m_6x1z1u_13b_rvt_physical_only.ndm \
41     $STD_CELL_RVT_NDM_DIR/tcbs28hpcplusbwp30p140tt0p9v25c_ccs.ndm \
42 ]
43 ;# A list of reference libraries for the design library.
44 ;# for hierarchical designs using bottom-up flows: include sub
45 ;# for hierarchical designs using ETMs: include the ET

```

사용할 NDM(standard cell physical library)들을 선언하여 PNR에서 참조할 수 있도록 설정한다. (이번 과제에서는 RVT 공정의 라이브러리를 사용한다.)

```
54 set VERILOG_NETLIST_FILES [list]
55 foreach m $DESIGN_INCLUDED {
56     lappend VERILOG_NETLIST_FILES "${IMPORT}/${m}/results/${m}.mapped.v"
57 }
58 lappend VERILOG_NETLIST_FILES "${IMPORT}/${DESIGN_NAME}/results/${DESIGN_NAME}.mapped.v" ;# Verilog netlist files;
59 #set VERILOG_NETLIST_FILES "${IMPORT}/${DESIGN_NAME}/results/${DESIGN_NAME}.mapped.v" ;# Verilog netlist files;
60 ;# for DP: required
61 ;# for PNR: required if INIT_DESIGN_INPUT is ASCII in icc2_pnr_setup.tcl; not re
62 or DP_RM_NDM
```

앞선 과제에서 Design Compiler 툴을 이용해 생성한 netlist(sa4x4.mapped.v, mac8.mapped.v, mult8.mapped.v)를 가져와 추가한다.

```
74 #set TCL_MCMC_SETUP_FILE "" ;# Specify a Tcl script to create your corners, r
75 set TCL_MCMC_SETUP_FILE "./config/${DESIGN_NAME}/${DESIGN_NAME}.mcm.top.tcl" ;# Speci
enarios and load respective constraints;
```

MCMM 설정 파일을 지정한다. 이 mcm.top.tcl file 내부에는 전압, 온도 등의 변화에 따른 시나리오와 칩의 기능이 정의되어 있다. 이를 통해, setup과 hold time 위반 없이 설계를 최적화할 수 있다.

```
{M1 horizontal} {M2 vertical} {M3 horizontal} {M4 vertical} {M5 horizontal} {M6 vertical} {M7 horizontal}
} {M8 vertical} {M9 horizontal} {AP vertical}"
;# Specify the routing layers as well as their direction and offset in a list of space delimited pairs:
```

Metal Layer별 배선 방향을 정의한다. M1, M3 같은 홀수 층은 horizontal, M2, M4 같은 짝수 층은 vertical로 교차 배선하였다.

-icc2_pnr_setup.tcl

해당 파일 역시 PNR 설정을 위한 다양한 변수들을 정의해둔 초기 설정 파일이다. 입력 파일 경로, 라이브러리 규칙, 공정 제약 조건 등 핵심 변수들이 정의되어 있는데 이 중 가장 주요한 몇 가지만 추려서 정리해보려 한다.

```
518 #set WRITE_GDS_LAYER_MAP_FILE "" ;# A layer map file which provides a mapping between the tool and GDS layers
519 set WRITE_GDS_LAYER_MAP_FILE "$REFLIB_PATH/RF/APR_Tech/Synopsys/tn28clpr002s1_1_9_1a/PRTF_ICC_28nm_Syn_V19_1a/PR_tech/Synopsys/GdsOut
ap/gdsout_6x1210.map" ;# A layer map file which provides a mapping between the tool and GDS layers
520 set WRITE_OASIS_LAYER_MAP_FILE "" ;# A layer map file which provides a mapping between the tool and OASIS layers
```

Design tool의 layer와 공정 mask layer를 매칭한다.

```
set TCL_FLOORPLAN_FILE "" ;# Tcl floorplan file written by the write_floorplan command; for example, floorplan/floorplan.tcl
;# TCL_FLOORPLAN_FILE and DEF_FLOORPLAN_FILE are mutually exclusive; please specify only one of them
;# Not effective if INIT_DESIGN_INPUT = DC ASCII or DP_RM_NDM.
;# The write_floorplan command writes a floorplan.tcl Tcl script and a floorplan.def DEF file;
;# reading floorplan.tcl alone can restore the entire floorplan - refer to write_floorplan man for mo
```

Chip size, I/O pin의 위치, macro의 위치 등을 정의하는 floorplan file이다. (우리는 우리가 정의한 floorplanning 파일을 사용할 예정이므로 공란이다.)

```
set TCL_PG_CREATION_FILE "./scripts/pns.tcl" ;# a Tcl scr
;# such as create_pg*, set_pg st
```

추후 다룰 pns.tcl 파일을 이용해 Vdd, Vss를 정의하고 연결한다.

```
set CTS_LIB_CELL_PATTERN_LIST "*/CKBD* */CKND* "
;# =7 =====
```

CTS에 사용할 균형잡힌 library cell pattern들의 list를 선언한다.

```
set CTS_NDR_RULE_NAME "icc2rm_2w2s_shield_default" ;#
a of the PM predefined rules
```

Clk 신호에 사용할 선을 지정하는 NDR을 선언한다.

```
_NAME
set CTS_NDR_MIN_ROUTING_LAYER "M4" ;# M
ied to.
set CTS_NDR_MAX_ROUTING_LAYER "M5" ;# M
ied to.
```

Clk 신호를 배선할 layer를 설정한다. M1~M3는 저항이 높아 속도가 느리므로, 저항이 낮은 M4, M5의 중간 layer에 clk을 배선한다.

-sa4x4/sa4x4.mcmm.top.tcl

```
15 ## We are working on typical-only corner (not MCMM)
16 ## So, unify late and early are all tt
17 set_parasitic_parameters \
18 -corners {tt0p9v25c} \
19 -late_spec tt0p9v25c -late_temperature 25 \
20 -early_spec tt0p9v25c -early_temperature 25 \
21
22
23 set_operating_conditions tt0p9v25c \
24 -analysis_type on_chip_variation \
25 -library /tech/TSMC_HPC28/Standard_Cell/core_cell_library/gate_length_30nm/9-tt
    _power_noise/CCS/tcbs28hpcplusbwp30p140_180a/tcbs28hpcplusbwp30p140tt0p9v25c_ccs
26
27 source ../dc/${DESIGN_NAME}/results/${DESIGN_NAME}.sdc
28
29 set_scenario_status -none -setup true -hold false tt0p9v25c
30 if [string length [get_defined_attributes -class scenario {dynamic_power}]] {
31     set_scenario_status -dynamic_power true -leakage_power true tt0p9v25c;
32 } else {
33     set_scenario_status -power [expr true || true] tt0p9v25c;
34 }
```

해당 파일은 MCMM(multi-corner, multi-mode)를 가정하여 다양한 scenario에서의 분석을 가능하게 한다. 그러나, 우리 과제에서는 (tt/0.9V/25°C)의 single scenario를 가정하므로 크게 수정할 것은 없다. 우리가 설계한 칩은 0.9V, 25도, typical 공정 조건에서 동작하며 이 조건에 맞추어 timing, power를 최적화한다.

-sa4x4/sa4x4.tdf

```
17
18 set_individual_pin_constraints -ports {A[0]} -allowed_layers {M4} -side 4 -width 0.05 -length 0.05 -offset 14
19 set_individual_pin_constraints -ports {A[1]} -allowed_layers {M4} -side 4 -width 0.05 -length 0.05 -offset 16
20 set_individual_pin_constraints -ports {A[2]} -allowed_layers {M4} -side 4 -width 0.05 -length 0.05 -offset 18
21 set_individual_pin_constraints -ports {A[3]} -allowed_layers {M4} -side 4 -width 0.05 -length 0.05 -offset 20
22 set_individual_pin_constraints -ports {A[4]} -allowed_layers {M4} -side 4 -width 0.05 -length 0.05 -offset 22
23 set_individual_pin_constraints -ports {A[5]} -allowed_layers {M4} -side 4 -width 0.05 -length 0.05 -offset 24
24 set_individual_pin_constraints -ports {A[6]} -allowed_layers {M4} -side 4 -width 0.05 -length 0.05 -offset 26
25 set_individual_pin_constraints -ports {A[7]} -allowed_layers {M4} -side 4 -width 0.05 -length 0.05 -offset 28
```

해당 파일은 sa4x4의 I/O pin의 위치를 수동으로 지정한다. 홀수 층의 metal은 horizontal, 짝수 층의 metal은 vertical로 설계된다. 또한, tdf I/O pin의 위치 뿐만 아니라 metal layer, width, length, offset 등을 설정한다. 또 다른 module을 만든다면, design에 맞는 configuration을 위해 해당 파일을 수정할 필요가 있다.

2.1.2 icc2/scripts/

-init_design.tcl

해당 파일은 Physical Design의 초기 setup을 담당하는 핵심 script이다. 현재 설정되어 있는 다양한 변수들을 sweep하면서 Place & Route를 다양하게 시도할 수 있다.

Floorplanning

```
300
301 # modified by smkcow
302 ## Perform initialize_floorplan if neither DEF_FLOORPLAN_FILES nor TCL_FLOORPLAN_FILE is specified
303 } else {
304     puts "RM-info: creating floorplan using initialize_floorplan"
305     #initialize_floorplan -side_length {40 80} -core_offset {5} -use_site_row
306     initialize_floorplan -side_length {320 100} -core_offset {5}
307
308 # move_objects [get_cell Ucmos10lpsrvv_ra1_hd_256x17m8] -to {10 36.5150}
309
310 # create_keepout_margin -type hard -outer { 8.0000 2.0000 2.0000 2.0000 } {Ucmos10lpsrvv_ra1_hd_256x17m8}
311 # If you want to remove, command :::::: remove_keepout_margins [get_keepout_margins]
312
313 # create_placement_blockage -type hard -boundary {{10 114} {213.5 114} {213.5 35} {10 35}}
314 # If you want to remove, command :::::: remove_placement_blockage
315
316 source "./config/${DESIGN_NAME}/${DESIGN_NAME}.tdf"
317 place_pins -self ;# to place unplaced pins if needed
318 }
```

306: core size를 설정한다. (now 320 μm (X) x 100 μm (Y))

316: 앞서 I/O pin configuration을 선언한 tdf file을 source하여 pin들을 place한다.

* cores_offset 5: chip boundary로부터 5 μm 의 margin을 유지하여 power ring 배치 or standard cell 배치(margin 안쪽)에 활용된다.

* icc2_shell console에서 "man create_floorplan"을 typing하면 command description과 arguments 등의 다양한 option을 확인할 수 있고, 이를 이용해 design을 최적화할 수 있다.

Power planning hook

```
546 # below lines are executed :: smkcow
547 #####
548 ## Power and ground network creation
549 #####
550 ## A Tcl script placeholder for your power ground network creation commands, such as create_pg*,
551 # set_pg_strategy, and compile_pg, etc.
552
553 #set TCL_PG_CREATION_FILE "/home/smkcow/QnA/digital/example_smkcow_DC_ICC2/fifo/PNR/SS/pnr/rm_icc2
554
555 ## Create standard cell PG rail
556 # Example : rm_icc2_pnr_scripts/init_design_std_cell_rail_example.tcl
557
558 # smkcow : pns.tcl includes STD Cell PG rail
559
560 if {[file exists [which $TCL_PG_CREATION_FILE]]} {
561     puts "RM-info: Sourcing [which $TCL_PG_CREATION_FILE]"
562     source -echo $TCL_PG_CREATION_FILE
563 } elseif {$TCL_PG_CREATION_FILE != ""} {
564     puts "RM-error: TCL_PG_CREATION_FILE($TCL_PG_CREATION_FILE) is invalid. Pls correct it."
565 }
566 }
```

564: pwr/gnd network script를 call한다.

* 실제 PG creation commands는 pns.tcl에 적혀있기에, 단순히 source 해준다.

End-cap cells & Tap cells

```
582 #####
583 ## Boundary cells
584 #####
585
586 ## Boundary cells: to be added around the boundaries of objects, such as voltage areas, macros, blockages, and the core area
587 #   set_boundary_cell_rules ...
588 #   report_boundary_cell_rules
589 #   compile_boundary_cells
590 #   check_boundary_cells
591
592 # add End cap cell
593 set_boundary_cell_rules -left_boundary_cell tcbn28hpcplusbwp30p140tt0p9v25c_ccs/TAPCELLBWP30P140 -right_boundary_cell tcbn28hpcp
v25c_ccs/TAPCELLBWP30P140
594
595
596 #####
597 # Fix all shaped blocks and macros
598 #####
599 #set_fixed_objects [get_flat_cells -filter "is_hard_macro"]
600
601 if [sizeof collection [get_cells -hier -filter is_hard_macro==true -quiet]] {
602     set_attribute -quiet [get_cells -hierarchical -filter is_hard_macro==true] status fixed
603 }
604 #####
605 ## Tap cells
606 #####
607 # Example : create_tap_cells -lib_cell myLib/Cell1 -distance 30 -pattern every_row
608 # Add tap cell array
609 create_tap_cells -lib_cell tcbn28hpcplusbwp30p140tt0p9v25c_ccs/TAPCELLBWP30P140 -distance 42 -pattern every_row -skip_fixed_cell
610 compile_boundary_cells
```

593(End-cap cells): standard cell들이 n-well을 공유하며 배치되는데, row가 끝난 시점에서도 n-well이 끊기지 않고 마감될 수 있도록 end-cap을 달아준다. (end cap이 없다면, DRC 관련 issue가 생긴다)

609(Tap cells): latch-up 현상을 방지하기 위해 Vdd, Gnd를 substrate와 well에 공급하는 tap cell을 42um 간격으로 모든 row에 배치한다. (macro는 제외)

610: 앞선 설정들을 실제로 배치한다.

-pns.tcl

해당 파일은 PNS(Power Network Synthesis)를 위한 파일로 chip의 Vdd, Gnd network를 구축한다. 해당 script를 실행하면 PG Ring, standard cell rail 삽입, PG Strap 생성 등의 task를 수행한다.

PG Ring Creation

```

33 #####
34 #-----
35 # P G   R I N G   C R E A T I O N
36 #-----
37 #####
38 #create_pg_ring_pattern ring_pattern -parameters {h_w h_s v_w v_s}\
39   -horizontal_layer M5 -horizontal_width {@h_w} -horizontal_spacing {@h_s} \
40   -vertical_layer M4 -vertical_width {@v_w} -vertical_spacing {@v_s} \
41   -via_rule { {intersection: all} {via_master: M4_M5_via_rule} } \
42   -corner_bridge false
43
44 create_pg_ring_pattern ring_pattern -parameters {h_w h_s v_w v_s}\
45   -horizontal_layer M5 -horizontal_width {@h_w} -horizontal_spacing {@h_s} \
46   -vertical_layer M4 -vertical_width {@v_w} -vertical_spacing {@v_s} \
47   -corner_bridge false
48
49 set_pg_strategy core_ring -core \
50   -pattern {{pattern: ring_pattern}{nets: {VDD VSS}}{offset: {2 2}} {parameters:1 0.5 1 0.5}} \
51   -extension {{stop: design_boundary_and_generate_pin}}
52
53 #set_pg_strategy macro_ring -macros Ucmos10lpsrv_rai_hd_256x17m8 \
54 #-pattern {{name : ring_pattern}{nets : {VDD VSS}}{offset : {1 1}}{skip_sides : 1 2}} \
55 #-extension {stop:innermost_ring}
56

```

Chip boundary 주변에 power ring을 생성한다. (M5 layer – horizontal, M4 layer – vertical, width = 1um, spacing = 0.5)

* -extension stop: design_boundary_and_generate_pin option ensures that PG pins are generated at the boundary

Standard Cell Rail Insertion

```

157 #-----
158 # S T A N D A R D   C E L L   R A I L   I N S E R T I O N
159 #-----
160 #####
161 create_pg_std_cell_conn_pattern \
162   std_cell_rail \
163   -layers {M1}
164 #   -rail_width {0.14 0.14}
165 #   -rail_mask {follow_pin}
166
167 set_pg_strategy rail_strat -core \
168   -pattern { {name : std_cell_rail} \
169     {nets: VDD VSS} } \
170   -extension {stop: innermost_ring}
171

```

standard cell row마다 M1 레이어에 Vdd, Vss rail을 삽입한다. 이 rail pattern은 각 cell row에 전원을 공급하는 수평 rail을 자동으로 생성하며, 가장 안쪽의 power ring까지 연결하여 standard cell들에 일관되게 전원을 공급한다.

PG Straps

```

207
208 #-----
209 # S T R A P
210 #-----
211 #####
212 #create_pg_strap -layer IB -direction vertical -width 5.0000 -net VDD -start 30 -stop 60 -pitch 30 -extend_low innermost_r
ing -extend_high innermost_ring \
213 -via_rule { \
214 {{existing: ring}}{via_master: IA_IB_via_rule}} \
215 {{intersection: undefined}}{via_master: default}} \
216 }
217 #create_pg_strap -layer IB -direction vertical -width 5.0000 -net VSS -start 38 -stop 68 -pitch 30 -extend_low innermost_r
ing -extend_high innermost_ring \
218 -via_rule { \
219 {{existing: ring}}{via_master: IA_IB_via_rule}} \
220 {{intersection: undefined}}{via_master: default}} \
221 }
222
223 create_pg_strap -layer M4 -direction vertical -width 0.5 -net VDD -start 15 -stop 300 -pitch 10 -extend_low innermost_ring
-extend_high innermost_ring
224 create_pg_strap -layer M4 -direction vertical -width 0.5 -net VSS -start 16 -stop 301 -pitch 10 -extend_low innermost_ring
-extend_high innermost_ring
225 create_pg_strap -layer M5 -direction horizontal -width 0.5 -net VDD -start 15 -stop 90 -pitch 10 -extend_low innermost_rin
g -extend_high innermost_ring
226 create_pg_strap -layer M5 -direction horizontal -width 0.5 -net VSS -start 16 -stop 91 -pitch 10 -extend_low innermost_rin
g -extend_high innermost_ring
227

```

M4 vertical strap과 M5 horizontal strap을 chip에 주기적으로 삽입한다. 이 strap들은 standard cell rail과 power ring 사이를 연결하여 낮은 저항의 power delivery network를 형성한다. 현재 설정에서는 0.5 μ m 너비로 10 μ m pitch마다 strap을 배치하는데, 이 값들은 IR drop 감소와 routing congestion 사이의 균형을 맞추기 위해 조정 가능하다.

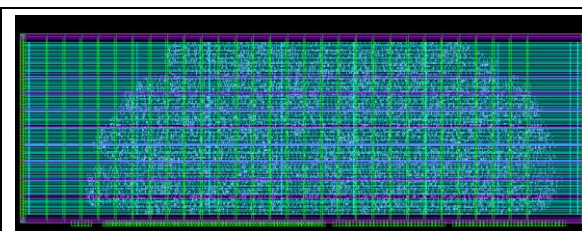
* Strap의 width가 너무 작으면 IR drop이 증가하고, 너무 크면 routing 공간이 부족하다. 마찬가지로 pitch가 너무 넓으면 IR drop이 증가하고, 너무 좁으면 congestion이 증가한다.

* check_pg_connectivity, report_pg_connectivity, report_power_plan commands를 이용해 다양한 task를 진행해 확인해 보자.

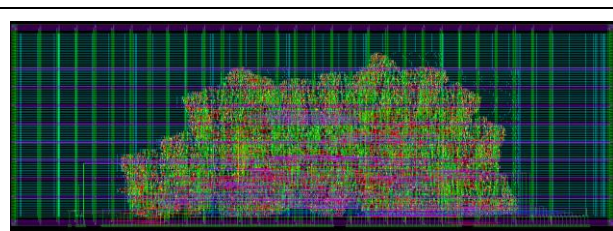
2.1.3 icc2/

Floorplan / Place & Route

이제 icc2 home directory에서 make clean clean_all init_design 명령어를 입력하면 이전 결과를 지우고 configuration files에 근거해 다시 floorplanning하게 된다. 그리고, start_gui command를 icc2_shell에 입력하면 아래 그림과 같은 floorplan의 gui가 나오게 된다. GUI를 확인하면서, Floorplanned 된 chip의 core boundary, power-ring offset, pin placement, end-cap/tap cells등이 문제 없이 적용되었는지 검증한다. 그리고, icc2 shell에서 exit하고 make all을 입력하여 floorplan, PNR 모두를 진행 후 GUI를 확인할 수 있다.



After init_design (floorplanning)



After make all(Complete Place-and-Route)

이후, sa4x4 directory가 생성되고 result directory에서 다양한 report를 확인할 수 있다.

2.2.1 Explore to find PPA

우선, 과제3에서 optimal frequency는 2.2GHz, Max frequency는 2.4GHz(mac8_2.6GHz, mult8_2.4GHz)였다. 두 모듈을 이용해 default 설정으로 PNR을 진행해보았다. 2.2GHz는 1.2GHz까지, 2.4GHz는 1.3GHz까지 timing 조건을 만족했다.

Syn Freq(GHz)	Power	Area(um ²)
2.2	1.31E-02	1.12E+04
2.4	1.59E-02	1.39E+04

A_0. 초기 PNR 결과 및 문제 식별

Core Size	Synthesis result	Utilization
320μm × 100μm	1.12×10 ⁴ μm ²	28.85%

초기 설계에서는 tool의 default floorplan 설정을 그대로 사용하였으며, 이로 인해 core size가 너무 크게 설정되어 있었다. 불필요하게 큰 core size로 인해 cell 간 평균 wire length 증가하면서 RC delay가 커졌고, 이는 performance 저하로 이어진다. 또한 routing resource가 필요 이상으로 분산되어 효율이 저하된다.

이에 따라, 최종 PNR 결과 70~80%대의 utilization을 만족할 수 있도록 수정하려한다.

A_1. 이론적 Utilization 개선 (이론적 Utilization 계산)

적절한 core size를 결정하기위해 이론적으로 floorplan core size를 대략 유추하였다. Synthesis된 sa4x4의 cell area는 1.12×10⁴ μm²이다. 이에 따라 필요한 core area는 cell area에 utilization을 고려하여 약 18,480 μm²로 계산되었고, core size를 200 μm x 100 μm 로 설정하고 PNR을 진행해 보았다.

A_2. Error Message 발견 및 Debugging

```
tend_high innermost_ring
Error: Strap with center 215 and width 0.5 is outside design boundary. (PGR-029)
Error: Strap with center 225 and width 0.5 is outside design boundary. (PGR-029)
Error: Strap with center 235 and width 0.5 is outside design boundary. (PGR-029)
Error: Strap with center 245 and width 0.5 is outside design boundary. (PGR-029)
Error: Strap with center 255 and width 0.5 is outside design boundary. (PGR-029)
Error: Strap with center 265 and width 0.5 is outside design boundary. (PGR-029)
Error: Strap with center 275 and width 0.5 is outside design boundary. (PGR-029)
Error: Strap with center 285 and width 0.5 is outside design boundary. (PGR-029)
Error: Strap with center 295 and width 0.5 is outside design boundary. (PGR-029)
Number of threads: 8
```

Error message를 통해 PG (Power/Ground) strap의 위치가 core boundary를 벗어나는 것이 문제임을 확인하였다.

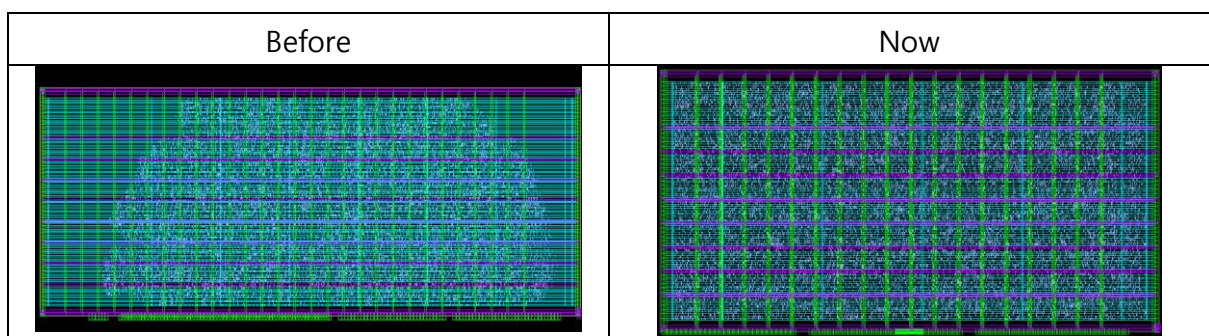
```
#It was Default strap for 320x100 design
#create_pg_strap -layer M4 -direction vertical -width 0.5 -net VDD -start 15 -stop 300 -pitch 10 -extend_low in
#create_pg_strap -layer M4 -direction vertical -width 0.5 -net VSS -start 16 -stop 301 -pitch 10 -extend_low in
#create_pg_strap -layer M5 -direction horizontal -width 0.5 -net VDD -start 15 -stop 90 -pitch 10 -extend_low i
#create_pg_strap -layer M5 -direction horizontal -width 0.5 -net VSS -start 16 -stop 91 -pitch 10 -extend_low i

#for 200x100 design
create_pg_strap -layer M4 -direction vertical -width 0.5 -net VDD -start 15 -stop 190 -pitch 10 -extend_low inn
create_pg_strap -layer M4 -direction vertical -width 0.5 -net VSS -start 16 -stop 191 -pitch 10 -extend_low inn
create_pg_strap -layer M5 -direction horizontal -width 0.5 -net VDD -start 15 -stop 90 -pitch 10 -extend_low in
create_pg_strap -layer M5 -direction horizontal -width 0.5 -net VSS -start 16 -stop 91 -pitch 10 -extend_low in
```

Pg_strap의 배치를 200x100에 맞게 다시 고쳐주고 다시 시뮬레이션을 진행하였다.

```
Utilization Ratio: 0.7009
Utilization options:
- Area calculation based on: site_row of block sa4x
- Categories of objects excluded: hard_macros macro_keep
Total Area: 19972.0080
Total Capacity Area: 19972.0080
Total Area of cells: 13998.7260
Area of excluded objects:
- hard_macros : 0.0000
- macro_keepouts : 0.0000
```

icc2_shell에서 report_utilization 명령어를 이용해 확인해보니 floorplan 단계에서 70%의 utilization을 달성하고 timing, power 모두 향상된 모습을 확인할 수 있었다. 그러나 실제 PNR 완료 후 최종 utilization은 이보다 낮아질 수 있다.



GUI도 확인해보니 코어 utilization이 한눈에 봐도 좋아진 모습이다. 우선 이 결과를 이용해 PNR 하였다.

A_3 Timing Constraints

PNR을 진행하고 report를 확인해 보았을때, Hold time violation이 발생하였다. 우선 cell 간 거리가 가까워지면서 clock skew를 조정할 수 있는 여유가 부족해졌고, 이로 인해 hold margin 확보가 어려워졌다고 생각하였다. 또한 buffer insertion을 수행하기 위한 물

리적 공간이 충분하지 않아 timing 제약을 맞추기 힘들다고 생각하였다.

이러한 문제를 해결하기 위해 Opt1~5까지 다양한 방법을 시도하였는데, 그 중 효과를 본 내용을 정리하였다.

Opt3에서는 placement 단계에서의 cell 간 거리가 좁아진 것을 원인으로 판단하고, icc2_pnr_setup.tcl에서 PREROUTE_PLACEMENT_MAX_DENSITY를 0.70으로 제한하였다. 이를 통해 placement 단계에서 cell density가 강제로 낮아지면서 routing channel과 buffer insertion을 위한 공간을 확보하였다. 그 결과, 이후 optimization 단계에서 buffer 삽입이 더욱 자유롭게 가능해졌고 hold violation을 성공적으로 해결하였다.

다만 이 과정에서 최종 utilization이 43%까지 하락하는 문제가 발생하였다. 이는 목표 utilization인 60~70%에 크게 못 미쳤다.

Opt4에서는 core size를 $150\mu\text{m} \times 80\mu\text{m}$ 로 축소하였으나, floorplan 단계에서부터 utilization이 100%를 초과하는 over-utilized error가 발생하였다.

이후 core size를 $180\mu\text{m} \times 90\mu\text{m}$ 로 설정하였다. GUI 분석 결과, cell들이 매우 조밀하게 배치된 상태임을 확인할 수 있었으며, 높은 utilization은 달성되었으나 이러한 과도한 density는 때문인지 hold violation이 다시 발생하였다.

Opt5에서는 density를 추가로 완화하여 PREROUTE_PLACEMENT_MAX_DENSITY를 0.60으로 낮추었다. 이를 통해 buffer insertion을 위한 공간이 충분히 확보되었고, 결과적으로 hold violation은 다시 한 번 해결되었다. 그러나 이 경우에도 utilization은 약 50% 수준에 머물러 area efficiency가 낮아지는 trade-off가 발생하였다.

마지막으로 시도 6에서는 core size를 $170\mu\text{m} \times 85\mu\text{m}$ 로 설정하였다. $160\mu\text{m} \times 80\mu\text{m}$ 설정에서는 utilization error가 발생하였기 때문에 이를 완화한 설정이다. 그러나 해당 조건에서도 hold violation이 재발하였으며, placement max density를 추가로 낮추는 방식은 utilization을 과도하게 희생하게 되므로 현실적인 해결책이 아님을 확인하였다. 이에 따라, 단순한 density 조절이나 core size 변경이 아닌 새로운 접근 방식이 필요하다는 결론에 도달하였다.

B_0 I/O 배치 최적화

기존 설계에서는 모든 I/O port가 die의 bottom에만 집중적으로 배치되어 있었다. 이로 인해 congestion이 물리고 clock latency가 증가하여, 이는 timing 제약으로 이어졌을 것이다. 더불어 die의 top, left, right 측면에 존재하는 routing resource가 거의 활용되지 않아, 전체 routing resource 효율이 크게 저하된 상태였다.

이러한 문제를 해결하기 위해 I/O port를 die의 네 면(top, bottom, left, right)에 분산하여 배치하였다. 특히, input port는 clock port와의 거리를 고려하여 배치하고 output port는 right, bottom에 배치하여 congestion을 최소화하였다.

이와 동시에 코어의 모양도 정사각형 모양의(110 μ m x 110 μ m)로 수정하였다. 그 결과, utilization이 76%까지 상승하여 기존 목표 범위였던 70~80%를 달성할 수 있었다.

B_1 Metal Layer 최적화

Layer	overflow		overflow (%)	# GRCs has	
	total	max		max overflow	
M1	0	0	0 (0.0000%)	0	
M2	383	3	366 (2.0383%)	2	
M3	0	0	0 (0.0000%)	0	
M4	0	0	0 (0.0000%)	0	
M5	0	0	0 (0.0000%)	0	
M6	0	0	0 (0.0000%)	0	
M7	0	0	0 (0.0000%)	0	
M8	0	0	0 (0.0000%)	0	
M9	0	0	0 (0.0000%)	0	
AP	0	0	0 (0.0000%)	0	
Both Dirs	383	3	366 (1.0192%)	2	
H routing	0	0	0 (0.0000%)	0	
V routing	383	3	366 (2.0383%)	2	

I/O port 재배치 이후에도 일부 routing layer에서 congestion이 있어, congestion report를 기반으로 metal layer별 utilization을 확인하였다. 확인 결과, M2 layer의 사용률이 과도하게 높았으며, overflow가 발생하고 있었다.

이에 따라, I/O port의 metal layer를 M7~M9에 배치하여 tool이 하위 layer에서부터 via stack을 통해 자연스럽게 상위 layer를 활용하도록 유도하였다. 이를 통해 전체 metal layer를 보다 균형 있게 활용하고자 하였다. 그러나 icc2_common_setup.tcl에서 routing layer maximum을 M7에서 M9로 확장한 결과, 예상과 달리 사용되지 않았다.

Metal layer	Hor. length	% of hor.	Ver. length	% of ver.
P0	0.0000	0.0000%	0.0000	0.0000%
M1	0.0000	0.0000%	0.0000	0.0000%
M2	1322.4900	4.9901%	10716.4850	35.6756%
M3	21393.1500	80.7221%	398.5100	1.3267%
M4	38.2600	0.1444%	17241.6050	57.3980%
M5	3286.9300	12.4025%	1.4100	0.0047%
M6	0.0000	0.0000%	1680.6800	5.5951%
M7	461.4050	1.7410%	0.0000	0.0000%
M8	0.0000	0.0000%	0.0000	0.0000%
M9	0.0000	0.0000%	0.0000	0.0000%
AP	0.0000	0.0000%	0.0000	0.0000%

생각해보니, M8과 M9는 두꺼운 layer로 RC delay가 커 setup violation을 유발할 가능성이 높아 사용이 제한되는 것으로 보였다. 이에 따라 상대적으로 M6 및 M7 layer를 중심으로 routing을 해보았으나 큰 의미가 있지는 않았다.

최종적으로는 tdf 파일에서 I/O port에 대해 하나의 metal layer를 강제하는 대신 set_individual_pin_constraints -ports {Cout} -allowed_layers {M5 M6}, 이 처럼 설정하여 복수 layer 중 하나를 선택하도록 설정하였다. 이를 통해 하위 layer인 M2에서 발생하던 congestion이 조금 줄어들기는 하였다.

B_2 Performance 및 Power 최적화

110 μm x 110 μm core size에서 최종적으로 76%의 utilization이 나오니 area는 여기서 만족하고, timing met을 맞추면서 performance 향상과 동시에 타이밍 위반을 막기 위해 clk을 Core의 중심부(55 μm)에 위치 시켰고 I/O Port도 standard cell과 가까운 metal layer 인 M2, M3를 사용하였다. setup violation이 나오면 앞서 설정한 PREROUTE_PLACEMENT_MAX_DENSITY 키워 cell들 간의 거리를 벌렸고, hold violation이 나오면 cell들간의 거리를 좁히는 등의 방법을 사용하였다.

```
78 #####
79 # SIDE 2 (TOP/NORTH) : Control Signals
80 # Centered for 110x110 core
81 #####
82 set_individual_pin_constraints -ports {clk} -side 2 -offset 55.0 -allowed_layers {M3}
83 set_individual_pin_constraints -ports {rstn} -side 2 -offset 57.0 -allowed_layers {M3}
84 set_individual_pin_constraints -ports {en} -side 2 -offset 59.0 -allowed_layers {M3}
85 set_individual_pin_constraints -ports {req_out} -side 2 -offset 61.0 -allowed_layers {M3}
86
```

또한, power 개선을 위해 pg strap의 width와 pitch를 조절하고 다양한 metal layer 조합을 시도하는 등의 최적화를 진행함과 동시에 timing 조건에 맞는 조합을 수차례 시도하였다.

```
36 #For 110x110 design
37 create_pg_strap -layer M4 -direction vertical -width 0.5 -net VDD -start 10 -stop 100 -pitch 25 -extend_low innermost_ring -extend_high innermost_ring
38 create_pg_strap -layer M5 -direction vertical -width 0.5 -net VSS -start 11 -stop 101 -pitch 25 -extend_low innermost_ring -extend_high innermost_ring
39 create_pg_strap -layer M4 -direction horizontal -width 0.5 -net VDD -start 10 -stop 100 -pitch 25 -extend_low innermost_ring -extend_high innermost_ring
40 create_pg_strap -layer M5 -direction horizontal -width 0.5 -net VSS -start 11 -stop 101 -pitch 25 -extend_low innermost_ring -extend_high innermost_ring
41
42
```

결론

init_design.tcl	s0	s1(Optional)	s1_	s2	s3
X,Y dim	110x110	110x110	110x110	110x110	110x110
core offset	5	5	5	5	5
pns_tcl(M4,M5)					
Ring Width	1	1	1	1	1
Ring Spacing	0.5	0.5	0.5	0.5	0.5
Strap Width	0.5	0.5	0.5	0.5	0.5
Strap Pitch	25	25	25	25	25
sa4x4.sdc					
clk Freq	1.2GHz	1.25GHz	1.2GHz	1.3GHz	1.3GHz
icc2_pnr_setup.tcl					

MAX_DENSITY	0.7	0.7	0.7		0.7
sa4x4.tdf(M2,M3)					
Routing	NSWE	NSWE	NSWE	NSWE	NSWE
Clk Signal	85um	85um	55um	85um	75um/95um
Timing	Met	Met	Not Met	Not Met	Not Met

power	8.57E+06	8.87E+06			
area	12067.06	12067.06			
performance	1.2GHz	1.25GHz			

* Design choice variations are marked in red

Summary of your team' s PNR process & how design choices influenced the results

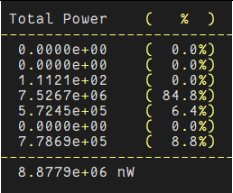
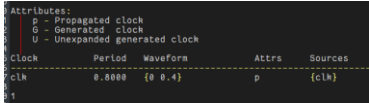
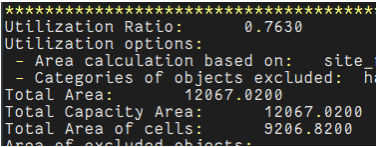
최종적으로 정리하면 이번 과제에서는 PPA 최적화를 위해 단계적으로 접근하였다. 먼저 area 측면에서 core size를 점진적으로 축소하며(200×100 → 180×90 → 110×110) 목표 utilization 70-80% 범위를 만족하는 floorplan을 확보하였다. Area가 고정된 후에는 performance 최적화 단계로 진입하여, I/O port와 clock port 위치 조정, placement density 제어 등을 통해 setup/hold timing violation을 해결하며 최대 동작 주파수를 찾았다. 마지막으로 timing이 만족된 상태에서 power 최적화를 수행하였으며, PG strap의 metal layer, width, pitch의 여러 조합을 시도하여 최적화를 수행하였다.

표를 보면 clk signal을 단순히 정중앙에 위치시키는 것이 아니라, hold/setup violation 유무에 따라 적절한 위치에 배치되어야 timing 조건을 만족하고 있음을 보여준다. 그리고, s2의 경우 setup violation이 발생하여 s3에서 clk signal의 위치를 바꾸며 timing constraint를 해결하려는 모습이다.

final PPA values (/w 2.2GHz synthesized sa4x4 netlist)

최적점 합성 결과: team_342106/assn4_mw/icc2/sa4x4_s1(Optimal)

(각자 Assn3 최적 frequency도 달랐고 많은 것을 시도하다 보니 최종 PPA value가 상이합니다.)

Power[nW]	Performance[GHz]	Area[μm^2]
 <p>8.88E+06</p>	 <p>1.25</p>	 <p>12067.06</p>

*sa4x4_s1(Optimal)/results/chip_finish.report_clocks, sa4x4_s1(Optimal)//results/chip_finish.report_power,
sa4x4_s1(Optimal)/results/chip_finish.report_utilization

Justification for Our PPA Point

1. Area

```

6 *****
7 Utilization Ratio: 0.7630
8 Utilization options:
9 - Area calculation based on: site_row of b
10 - Categories of objects excluded: hard_macr
11 Total Area: 12067.0200
12 Total Capacity Area: 12067.0200
13 Total Area of cells: 9206.8200
14 Area of excluded objects:
15 - hard_macros : 0.0000
16 - macro_keepouts : 0.0000
17 - soft_macros : 0.0000
18 - io_cells : 0.0000
19 - hard_blockages : 0.0000
20 Total Area of excluded objects: 0.0000
21 Ratio of excluded objects: 0.0000
22
23 Utilization of site-rows with:
24 - Site 'unit': 0.7630
25
26 0.7630
27

```

Utilization 76.3%는 초기 목표로 삼았던 70-80% 범위 내에 위치하며, Area efficiency가 효율적인 design을 달성했다. 또한, timing met을 위한 buffer 삽입 공간이 확보되어 있어 setup timing 관리에서 met이 가능했다. 또한, routing congestion이 적절히 관리되어 DRC violation 없이 routing이 완료되었다. 만약 utilization을 80% 이상으로 높였다면 routing congestion으로 인해 timing이 더욱 악화되어 1.25GHz 달성도 불가능했을 것이며, 반대로 70% 이하로 낮췄다면 area를 15,000 μm^2 이상으로 낭비하게 되었을 것이다.

2. Performance

1.25GHz는 timing met을 달성할 수 있는 실질적인 최대 주파수이다. setup slack과 hold slack이 각각 +11.2ps와 +0.7ps로 뽁뽁하며, 특히 critical path는 +0.7ps로 거의 한계에 도달했다. 더 높은 frequency를 시도했을 때 timing violation이 발생하여 PNR이 실패하였으므로, 1.25GHz가 현재 design과 floorplan 조건에서 달성 가능한 maximum

performance이다. 참고로, 1.3GHz로 timing met에 성공하긴 하였지만(sa4x4_2.2_110x110_1.3G_M2_clk_mid), power가 9.60E+06이 나와 실질적인 최대 주파수로 1.25GHz를 선정했다.

3. Power

```

inf, value = 0.048409 (POW-046)
Note - message 'POW-046' limit (10) exceeded. Remainder will be suppressed.

Cell Internal Power      = 7.0009e+06 nW ( 79.3%)
Net Switching Power     = 1.8306e+06 nW ( 20.7%)
Total Dynamic Power     = 8.8315e+06 nW (100.0%)
Cell Leakage Power      = 4.6396e+04 nW

Attributes
-----
u - User defined power group
i - Includes clock pin internal power

Power Group      Internal Power      Switching Power      Leakage Power      Total Power      ( % )      Attrs
-----
io_pad           0.0000e+00      0.0000e+00      0.0000e+00      0.0000e+00      ( 0.0%)
memory          0.0000e+00      0.0000e+00      0.0000e+00      0.0000e+00      ( 0.0%)
black_box       0.0000e+00      0.0000e+00      1.1121e+02      1.1121e+02      ( 0.0%)
clock_network   6.1939e+06      1.3319e+06      9.3558e+02      7.5267e+06      ( 84.8%)      i
register        4.2813e+05      1.2927e+05      1.5042e+04      5.7245e+05      ( 6.4%)
sequential      0.0000e+00      0.0000e+00      0.0000e+00      0.0000e+00      ( 0.0%)
combinational   3.7887e+05      3.6951e+05      3.0308e+04      7.7869e+05      ( 8.8%)
-----
Total           7.0009e+06 nW      1.8306e+06 nW      4.6396e+04 nW      8.8779e+06 nW

```

Power density가 $8.88\text{mW} / 12,067\mu\text{m}^2 = 0.74 \text{ mW/mm}^2$ 로 낮은 편에 속한다. 또한, J/OPs는 $8.88\text{mW} / 1.25\text{GHz} = 7.1 \text{ pJ/cycle}$ 로 합리적인 수치이다. 또한, internal power 79.3%와 switching power 20.7%의 비율은 작은 크기의 고집적된 block에서 나타날 수 있는 typical한 분포이다.

추가적으로, 각자 진행한 PNR 과정과 결과가 상이한 만큼 조원들의 PPA와 비교하며 각자의 PPA를 netlist에 따라 비교해 보려한다.

우리 조는 각자 Assn3에서 결정된 Optimal Frequency가 달라 합성된 netlist의 특성이 다르다. Design Compiler의 최적화 전략에 따라 서로 다른 cell selection, buffering strategy 등을 거쳤으므로, PNR 단계에서도 서로 다른 최적점을 가질 수밖에 없다. 이에 따라 각 netlist는 기본적으로 다음과 같은 특성을 갖는다고 예측할 수 있다.

	1.4GHz	1.8GHz	2.2GHz
Performance	느림	균형	빠름
Power	적음	균형	높음
Area	작음	균형	큼
Timing Constraint	느슨함	균형	빡빡함

그리고 각자가 찾아낸 최종 PPA점은 아래와 같다.

	1.4GHz	1.8GHz	2.2GHz
Performance	0.9GHz	1GHz	1.25GHz
Power	5.15E+06	6.24E+06	8.88E+06
Area	8168	10979.01	12067.06

위의 표에서 synthesis frequency에 따른 아래와 같은 netlist 특성이 명확하게 최종 결과에 반영되었음을 확인할 수 있다.

1. synthesis frequency가 높을수록 최종 PNR frequency도 높아진다 (0.9GHz → 1.0GHz → 1.25GHz). 이는 높은 synthesis frequency로 합성된 netlist가 큰 cell과 많은 buffer를 포함하여 critical path delay가 짧기 때문이다.

2. synthesis frequency가 높을수록 power consumption도 비례하여 증가한다 (5.15mW → 6.24mW → 8.88mW). Large cell은 내부 capacitance가 크고 leakage current가 높으며, 많은 buffer는 switching power를 증가시키기 때문이다.

3. synthesis frequency가 높을수록 area도 증가한다 (8,168 μm^2 → 10,979 μm^2 → 12,067 μm^2). Large cell과 buffer의 물리적 크기가 크고, timing을 맞추기 위한 additional logic도 증가하기 때문이다.

조원 3명의 서로 다른 PPA point는 각각이 특정 domain에서의 optimization objective에 따라 최적의 PPA가 될 수 있다.

Additional discussion on potential methods for further PPA improvement

이번 과제에서는 utilization 목표를 70%대로 aggressive하게 잡고 최적화를 시도하니 power/performance 측면에서 다양한 시도를 해보았음에도 timing 잡기가 쉽지 않았다. 이는 placement 밀도가 높아 routing congestion과 buffer 삽입 공간이 부족하여 생긴 문제였던 것 같다.

다음에는 area를 처음부터 강하게 제한하기 보다는 상대적으로 여유 있는 core size에서 performance를 우선적으로 확보한 후, timing margin을 기반으로 area를 축소하는 방식의 접근이 더욱 효과적일 것으로 보인다. 즉, 초기 단계에서는 timing margin을 확보하고, 이후 simulation을 반복하는 과정을 통해 area를 점진적으로 조정함으로써 PPA 간의 균형을 맞추는 전략이 필요할 것으로 생각된다.

과제3에서 optimal frequency로 2.2GHz가 선정되어 해당 netlist를 기반으로 PNR을 진

행하였다. 만약 더 낮은 주파수에서 합성된 netlist를 사용하였다면, synthesis 단계에서의 timing constraint이 완화되어 gate upsizing이나 buffer 삽입이 줄어들고, 이에 따라 cell count와 routing complexity가 감소하여 hold timing/Power/Area 측면에서 이점을 가져갈 수 있었을 것으로 생각된다.

과제4에서는 sa4x4 모듈 단독 합성을 기준으로 floorplan 및 I/O 배치를 설계하였으나, 실제 설계에서는 해당 모듈이 단독으로 사용되는 경우는 거의 없으며, 최종적으로는 상위 Top module에서 통합되기 마련이다. 따라서 향후 설계에서는 top level에서의 연결 구조, 데이터 흐름, clock distribution, 그리고 주변 블록과의 관계를 사전에 고려한 설계가 필수적이다.

Assesment

한민준- 디버깅 시 안되는 것이 있으면 잘 대답해주었으며, 꼭 확인해보아야할 명령어들을 알려주었음.

주형훈- 과제4 기본 환경 세팅 및 다양한 작업들을 먼저 진행해주었으며, 그로 인해 얻은 다양한 팁들을 공유해줌.