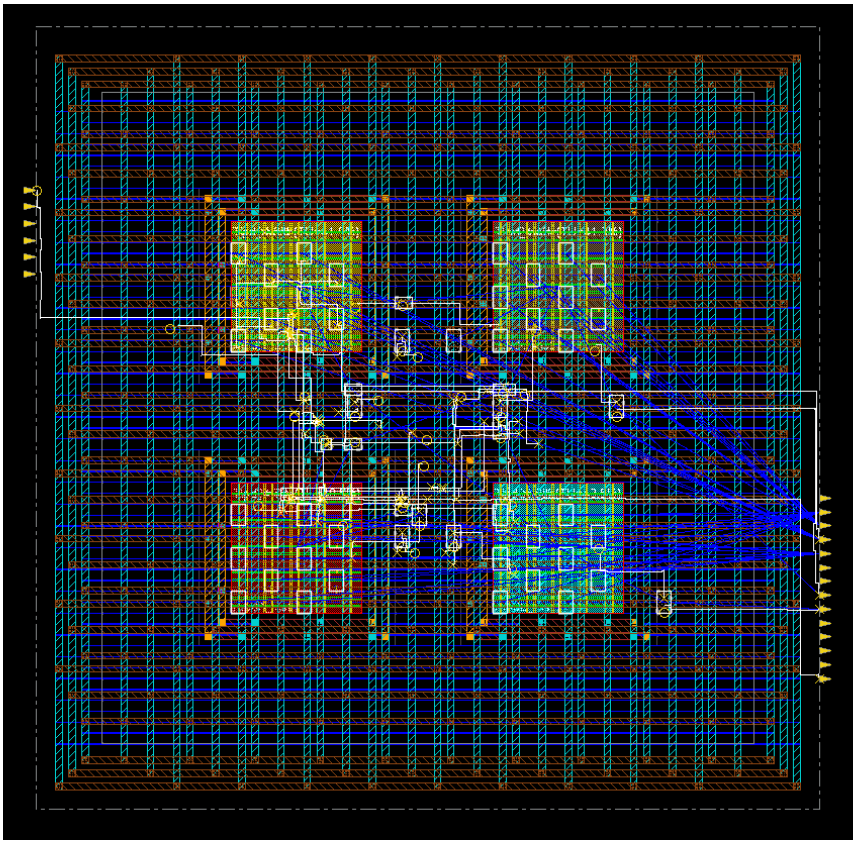
Low-Power UPF Physical implementation

Alexander Maso, Jie Sun
Graduate Students, Department of Electrical and Computer Engineering, Portland State University
Portland, Oregon, USA
amaso@pdx.edu, sun5@pdx.edu



1. Introduction

1.1 Overview of the Low-Power UPF Physical implementation

UPF, which stands for Unified Power Format, serves as a standardized methodology and format employed in electronic design automation (EDA) to describe and manage the power intent of a digital design. In the context of this project, our focus will be on comprehending the Physical Design components of a Multi-Voltage design. We utilized Synopsys Design Compiler and Cadence Genus to perform RTL synthesis and generate preliminary metrics. Subsequently, Cadence Innovus was employed to carry out the Physical Implementation aspects of the Multi-Voltage design. For this project specifically, the physical implementation encompassed activities such as floorplanning, power domain implementation, integration of power management cells (such as power-switches, level shifters, and isolation), power routing (using rings and stripes), placement, and routing.

1.2 The significance of power management in digital design.

Power management and low power design helps with energy efficiency, enhances performance, improves reliability, reduces costs, addresses thermal challenges, and promotes environmental sustainability.

Incorporating effective power management techniques is crucial for achieving optimal power consumption, ensuring longevity, and meeting the requirements of modern digital systems.

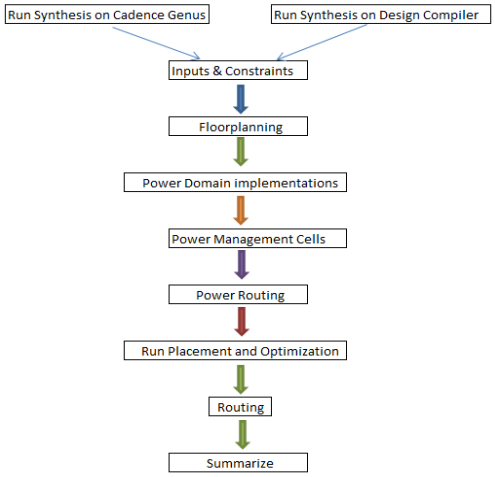


Figure 1: Work Flow

2. General Flow of Process

2.1 Progress flow

The team implemented a work structure that encompassed individual work on specific projects and collaborative work on others. Jie began by executing synthesis on the provided RTL, UPF, and SDC using Synopsys Compiler, employing specific commands to produce the necessary reports in Synopsys Design Compiler. Alex started with synthesis on the given RTL, UPF, and SDC utilizing Cadence Genus, creating a schematic of the entire design, and highlighting the power management cells within the design. Following this, the team collaborated on the remaining tasks. As the team worked through each task required and wrote out the tcl script, Alex used the script to run the results from Genus and Jie used the script to run the results from Design Compiler. Ultimately the synthesis from Genus was used for the physical implementation that was presented in the final presentation.

3. Overall Design Goals

For this particular project, since the RTL and UPF files have already been provided, our main objective is to delve into the Physical Design aspects of the MV design implementation. The focus here is on gaining knowledge regarding RTL synthesis and generating metrics using Synopsys Design Compiler and Cadence Genus. However, the integration of power management cells at the physical level will be exclusively performed using Cadence Innovus. As novice learners, we are striving to leverage and master the following commands in order to successfully carry out the tasks. We will refer to the Innovus shell's manual to obtain detailed information and usage (refer to the command list below).

init_design
read_power_intent
commit_power_intent

floorplan
editPin
modifyPowerDomainAttr
planDesign

addPowerSwitch
addRing
addStripe
sroute
place_design
place_opt_design
optDesign
routeDesign

4. Project Design Progress and Result

4.1 Tas1k 1 Run Synthesis in Genus and Design Compiler

4.1.1 Task 1-ab - Synthesis in Design Compiler

Step 1: change the dc-mv_lp_top.tcl file first line to the command : source -echo -verbose ../../$top_design.design_config.tcl

Step 2: we run Synthesis in Design Compiler, we generated UPF diagram, thenThrough the Generated UPF diagram help a UPF diagram function, the team get the UPF diagram view as below:

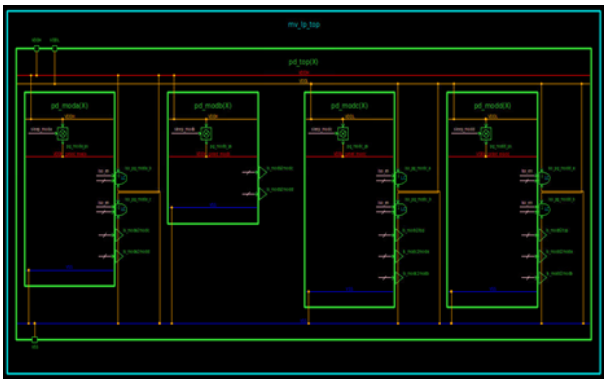


Figure 2:

Step 3: according the spec, we input the command as below:
report_power_domain –nosplit

To get the report as below:

power_domin				
	Power-VDDH	Ground-VSS	Input	Output
pd_top	1.16	0	X	X
pd_moda	1.16	0	1.16	1.16
pd_modb	1.16	0	0.85	0.85
pd_modc	1.16	0	0.85	0.85

Figure 3:

Step 4: input the command as below:
report_level_shifter –nosplit

To get the report as below:

level_shifter –nosplit					
Level Shifter	Reference	Input P/G (VDDH)	Output P/G (VSS)	Main P/G (VDDH)	Violation
D2B_UPF_LS	LSUPX1_HVT	0.85	0.00	1.16	0.00
C2B_UPF_LS	LSUPX1_HVT	0.85	0.00	1.16	0.00
C2A_UPF_LS	LSUPX1_HVT	0.85	0.00	1.16	0.00
D2A_UPF_LS	LSUPX1_HVT	0.85	0.00	1.16	0.00

Figure 4:

Step 5: input the command as below:
report_power_switch -verbose

To get the report as below:

Power_switch					
Power_switch	Inport net (VDDH)	VDDH_gated_moda	Control port	On States	Off States
pg_moda_ps	1.16	1.16	Sleep_moda	ON, sleep_moda	OFF, !sleep_moda
pg_modb_ps	1.16	1.16	Sleep_modb	ON, sleep_modb	OFF, !sleep_modb
pg_modc_ps	0.85	0.85	Sleep_modc	ON, sleep_modc	OFF, !sleep_modc
pg_modd_ps	0.85	0.85	Sleep_modd	ON, sleep_modd	OFF, !sleep_modd

Figure 5:

Step 6: input the command as below:
report_isolation_cell -verbose

To get the report as below:

report_isolation_cell --Power domain: pd_moda					
Port	Port Dir	ISO lib cell	ISO Strategy	ISP Powet net	ISO Ground net
moda_inst/A2B	Out	ISOLANDX1_HVT	iso_pg_moda_b	VDDH	VSS
moda_inst/A2C	Out	ISOLANDX1_HVT	iso_pg_moda_c	VDDH	VSS
report_isolation_cell --Power domain: pd_modb					
Port	Port Dir	ISO lib cell	ISO Strategy	ISP Powet net	ISO Ground net
moda_inst/B2C	Out	ISOLANDX1_HVT	iso_pg_moda_c	VDDH	VSS
moda_inst/B2D	Out	ISOLANDX1_HVT	iso_pg_moda_c	VDDH	VSS
report_isolation_cell --Power domain: pd_modc					
Port	Port Dir	ISO lib cell	ISO Strategy	ISP Powet net	ISO Ground net
moda_inst/C2B	Out	ISOLANDX1_HVT	iso_pg_moda_b	VDDL	VSS
moda_inst/C2A	Out	ISOLANDX1_HVT	iso_pg_moda_a	VDDL	VSS
report_isolation_cell --Power domain: pd_modd					
Port	Port Dir	ISO lib cell	ISO Strategy	ISP Powet net	ISO Ground net
moda_inst/D2B	Out	ISOLANDX1_HVT	iso_pg_moda_b	VDDL	VSS
moda_inst/D2A	Out	ISOLANDX1_HVT	iso_pg_moda_c	VDDL	VSS

Figure 6:

Step 7: input the command as below:
report_supply_net

To get the report as below:

Supply net	Opt voltage-max	Opt voltage-min
VDDH	1.16	1.16
VDDL	0.85	0.85
VSS	0	0
VDDH_gated_moda	1.16	1.16
VDDH_gated_modb	1.16	1.16
VDDH_gated_modc	0.85	0.85
VDDH_gated_modd	0.85	0.85

Figure 7:

Step 8: input the command as below:
report_pst

To get the report as below:

report_pst	
user PST name	mv_gated
total Power states	3

Figure 8:

	VDDH	VDDL	VSS	pg_moda_ps/VDDH_gated_moda	pg_moda_ps/VDDH_gated_modb	pg_moda_ps/VDDH_gated_modc	pg_moda_ps/VDDH_gated_modd
s0:	power on	power on	gnd	power on	power on	power on	power on
s1:	power on	power on	gnd	power off	power on	power on	power off
s2:	power on	power on	gnd	power on	power on	power off	power off

Figure 9:

Step 19: input the command as below:
report_qor

To get the report as below:

	in2out	in2reg	reg2out	upf_clk
Level of logic	4	4	0	3
critical Path Length	0.35	0.62	0.11	0.59
Critical Path Slack	-0.08	-0.07	0.40	0.12
Critical Path clk period	0.80	0.80	0.80	0.80
Total negative slack	-0.27	-0.50	0.00	0.00
No. of Violating Paths	5.00	12.00	0.00	0.00
Worst Hold Violation	0.00	0.00	0.00	0.00
Total Hold Violation	0.00	0.00	0.00	0.00
No. of Hold Violations	0.00	0.00	0.00	0.00

Figure 10:

Design Rules	
total Number of Nets	157
Nets with Violations	0
Max Trans Violations	0
Max Cap Violations	0

Figure 11:

4.1.2 Task 1-cd - Synthesis in Genus

After running the provided RTL and UPF files through synthesis using Genus, we were asked to display the entire design with the schematic view (Figure 12). In addition, we were asked to highlight all the power management cells present in the design. This became one of the very first challenges. After identifying and selecting the power management cells in the GUI, Alex searched through the different tools available through the drop-down menus and even the help menu itself, looking for a way to highlight all the power management cells at once. After trying a number of things without success, Alex reached out for some help on slack. Professor Brian recommended that we try using the command line and provided this command *select_obj [get_cell -hier -filter "ref_name=~cell_name"]* and suggested we try something similar but replace *cell_name* with wildcards surrounding HVT, ISO etc. While, ultimately this was not a solution to highlighting all the cells at once, the attempts to do so helped to familiarize Alex with the different cells present in our design. Eventually, it became clear that this was an exercise in futility as no group was able to select them all at once. Instead, Alex highlighted power management cells individually (Figure 13, Figure 14).

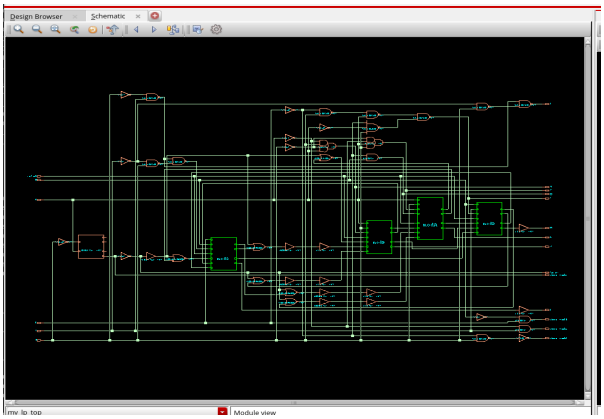


Figure 12:

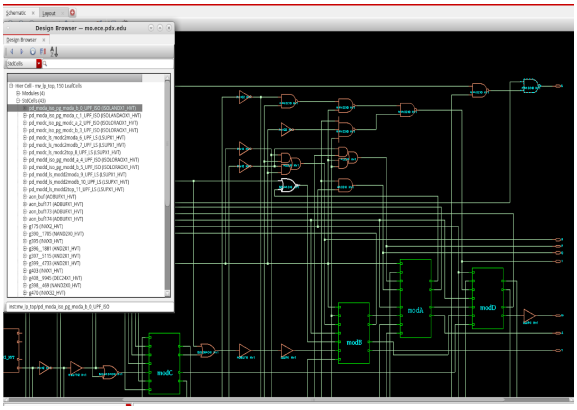


Figure 13:

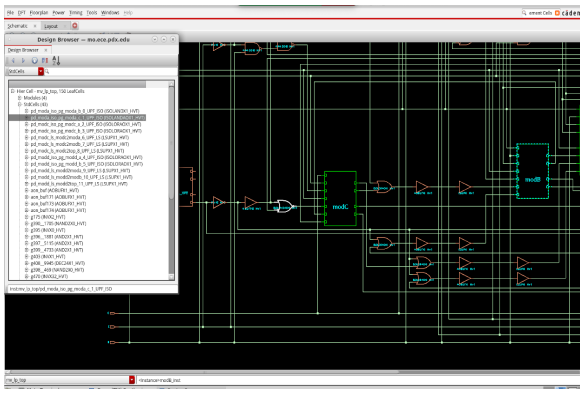


Figure 14:

After running synthesis individually, Alex and Jie compared timing results from Genus with the results from Design Compiler.

	Critical Path Slack	Total Negative Slack	Number of Violating Paths	Critical Path Slack	Total Negative Slack	Number of Violating Paths
in2out	-0.398	-1.276	6	-0.08	-0.27	5
in2reg	-11.547	-71.677	30	-0.07	-0.5	12
reg2out	-6.083	-13.442	4	0.4	0	0
upf_clk	-14.125	-127.841	49	0.12	0	0

Figure 15: Compare Timing from Genus and from Design Compiler

4.2 Task 2 – Inputs & Constraints

When we run Script the PnR Flow: first, we grep for aliases and created our own for mv_lp_top. Then we use floorplan-macros-innovus.tcl as template. After that, we referred to UPF and SV files for pwr_net & other differences.

Below are the codes:

```
alias fs set top_design fifo1_sram
alias f set top_design fifo1
alias o set top_design ORCA_TOP
alias e set top_design ExampleRocket
alias lp set top_design mv_lp_top
```

Figure 16: enc.tcl

```
source -echo -verbose ../../$top_design.design_config.tcl

set designs [get_db designs * ]
if { $designs != "" } {
  delete_obj $designs
}

set_global _enable_mmmc_by_default_flow      $CTE::mmmc_default

source ../scripts/innovus-get-timlibslefs.tcl
source ../../constraints/${top_design}.mmmc.sdc

set init_design_netlisttype Verilog
set init_verilog ../../syn/outputs/${top_design}.genus.vg
set init_top_cell $top_design
set init_pwr_net VDD
set init_gnd_net VSS


init_design

set_interactive_constraint_modes [all_constraint_modes -active]
```

Figure 17: floorplan-macros-innovus.tc

```
#############################################################
# Example PnR Script for UPF MV implementation project
# ECE-530 PDIC
# Author: Venkatesh Patil
# This is skeloton script to be udpated by students
# as needed for the project and add good comments
# to showcase your thinking and work
#
# This script is for INNOVUS
#############################################################
# Currently this is impletented as one-script, but can be
# Modified to source multiple scripts based on functionality
#
#############################################################

# Project design specific config file
source -echo -verbose ../../${top_design}.design_config.tcl


set_global _enable_mmmc_by_default_flow      $CTE::mmmc_default
source ../scripts/innovus-get-timlibslefs.tcl


###NOTE You need to fill in the correct data < > per your design

# Design specified Variables
set init_design_netlisttype Verilog
set init_verilog ../../syn/outputs/${top_design}.genus.vg
set init_top_cell ${top_design}
set init_pwr_net {VDDH VDDL}
set init_gnd_net VSS
set power_intent_file "../../syn/rtl/${top_design}.upf"
```

Figure 18: mv_lp_top.tc

4.3 Task 3 – Floorplanning

This task primarily focused on learning how to use the *floorplan* and the *editPin* commands in Innovus. As with the previous tasks, we referred to the ORCA_TOP design and the scripts that we sourced in the lab assignments as a starting point. From the scripts we were able to figure out which parameters we might need to use and were able to research them using the man pages as

well as the Cadence support site. The *floorplan* command found in floorplan-macros-innovus.tcl indexes the elements of *$design_size* and uses *$margin*, which was already defined in the framework we were given for our tcl script. Reading through the if statement that sets the value of margin helped us to better understand how it was being used as a parameter for *floorplan* and we were able decode the remaining parameters using the man page. We used similar methods to understand the *editPin* command and how floorplan-macros-innovus.tcl uses *get_ports* and *sizeof_collection* to split the pins into two groups of equal size and place each group on a different layer. At first, we just copied this portion of floorplan-macros-innovus.tcl and updated some of the numeric values to work with our design. Once we knew *editPin* was working, we used apropos to look for a way to select the input pins separately from the output pins. This led us to *all_inputs* and *all_outputs* and with those, we were able to alter the framework we already had and select all the input pins and place them separately from all of the output pins (figure 19). Following ORCA_TOP we placed the pins on different layers initially, but after speaking with Professor Venkatesh, we moved them to the same layer. The correct edge numbers were found by trial and error since there weren't many possibilities.

```
## Copied Pin placement commands from ORCA_TOP tcl file
## Updated to conform to our design specifications

#Cadence method.  Not floating with these statements
#Isolated all input ports and then all output ports
setPinAssignMode -pinEditInBatch true
set input_ports [ all_inputs ]
set output_ports [ all_outputs ]
# Place the inputs on left edge and the outputs on the right
# Keep all pins on the same layer and space them out properly
editPin -edge 0 -pin [get_attribute $input_ports full_name ] -layer M6 -spreadDirection
counterclockwise -spreadType START -offsetStart 25 -spacing 10 -unit MICRON -fixedPin 1
editPin -edge 2 -pin [get_attribute $output_ports full_name ] -layer M6 -spreadDirection
 counterclockwise -spreadType START -offsetStart 20 -spacing 8 -unit MICRON -fixedPin 1
setPinAssignMode -pinEditInBatch false
```

Figure 19: editPin

4.4 Task 4 – Power Domains

The next task required us to create separate power domains using the command *modifyPowerDomainAttr* and then use *planDesign* to update the floorplan. We were a lot more comfortable with *modifyPowerDomainAttr* than with some of the other commands, since we had some practice with it in lab 2 and lab 3. During these labs we were asked to alter the dimensions of the provided power domain and gained some familiarity with the -box parameter. We used grep to find PD_RISC_CORE in the upf file for ORCA_TOP which led us to look through our own upf file for the names of the different power domains. After sketching out a rough graph of our floorplan and labeling it with some coordinates, it was simple enough to place the four power domains in 20 x 20 boxes, evenly dispersed across the floor plan. The last step for creating the power domains was to make a halo around each. In the lecture slides we saw a parameter called gapEdge but when we tried to use it, the tool informed us that was not valid. Another quick apropos and we found the minGaps parameter and created a 5 micron halo around our power domains (figure 20). The man page for *planDesign* initially indicated the need for a constraint file but the tool threw errors when we tried to pass it the sdc file for this parameter and (since the sdc file was sourced earlier in the script) we ultimately used it without any parameters.

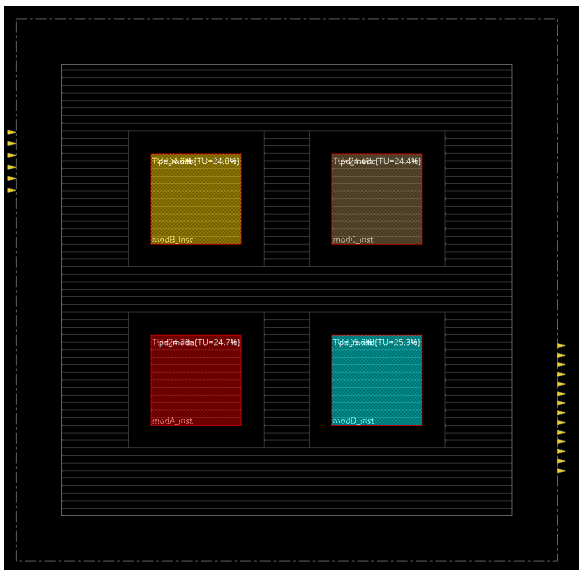


Figure 20: Halos around PD

4.5 Task 5 – Power Management Cells

Task 5 was to implement the power management cells by inserting power switches into each of the four separate domains. This was accomplished using one *addPowerSwitch* command for each domain. The header cell was provided to us in the tcl script itself but we had to work out the rest of the parameters ourselves. Using grep again, we found the header cell used as a parameter for the *create_power_swtich* command in the provided upf file. From the upf file we were able to determine the name of each power switch itself and try to use it from the command line. The tool itself indicated that we needed to include the *horizontalPitch* parameter and the man page made it apparent that this was the distance between the individual switches placed. Through trial and error we discovered that if this value was too low (or too high) then the tool wouldn't place any switches at all and ultimately, a value of 5 was juuust right. (figure 21) Our team had trouble with what net to use for the *enableNetIn* parameter but we were able to get some help from the T.A. and from Professor Venkatesh. Initially we thought the enable net was iso_en and had some difficulty understanding why the enable net would be sleep but after reviewing the upf file and the lecture slides a bit more, we came to the understanding that the power switches toggle the domain from full power (either VDDH or VDDL depending) to the gated power. The enable net is named sleep here because the power switch essentially puts a portion of the power supply to sleep.

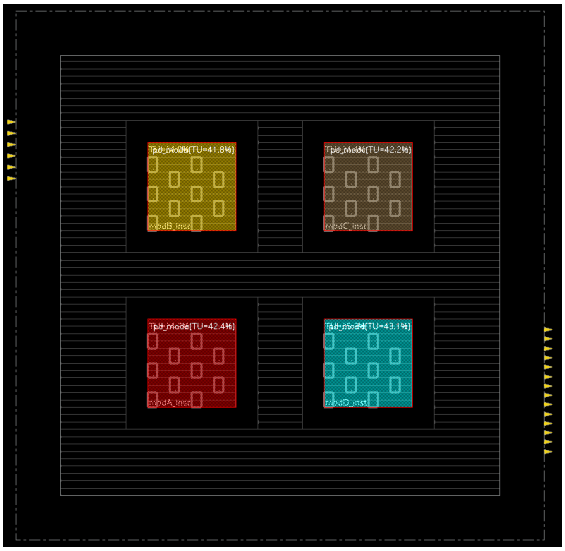


Figure 21: Power Switches

4.6 Task 6 – Power Routing, Ring, stripes and Routing

4.6.1 Top Power Domain

Task 6 was far and away the most challenging task. This task required us to add power rings and stripes for the top power domain as well as the four separate modules. The first part of this task was to create rings for VDDH, VDDL and VSS that encircled the entire design (figure 22) and create stripes that ran across it. The skeleton tcl script that we were provided indicated that these rings and stripes needed to be placed on layers M8 and M9. The lectures

on low power design and power routing helped us to determine what parameters we needed to include for these commands and many of the values for spacing, offset etc. were worked out through trial and error by running the commands one at a time on the command line.

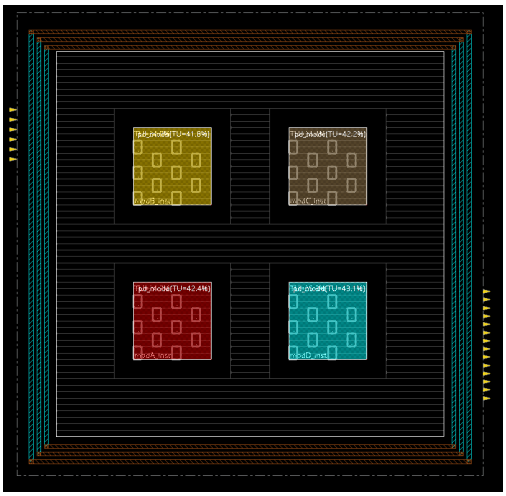


Figure 22: Rings for the Top PD

4.6.2 Modules A - D

After completing the power routing commands for the top power domain, the same needed to be done for each of the four modules. Each module in our design was its own power domain with both a primary net and a secondary, gated net. Modules A and B used the high voltage supply (VDDH) as their primary net and Modules C and D used the low voltage supply (VDDL). The metal layers to use for these rings and stripes were provided in the skeleton tcl file (M6 & M5 for the rings and M4 & M3 for the stripes). We created rings around each of the four domains and added stripes through them for both the primary and gated nets. (figure 23) The comments provided in the initial tcl script indicated that an *sroute* command was required to route each of the four separate power domains. The *sroute* command and the parameters that were required for this command proved to be one of the most difficult to figure out. Initially, we used the man page to look up each of the parameters shown in the low power design lecture slide. The values for some of these parameters were included in the slide (ie. *-connect*) but others were not and required us to do some research. The most challenging of all was trying to find the correct net to use for the *-secondaryPinNet* parameter. Initially we thought we should use the gated net for each domain but that proved unsuccessful at this stage. The cadence support page indicated that the secondaryPinNet was for level shifters and so we tried that next but still no success. Next we tried VDDH and VDDL as the secondary nets as well as the primary. As we went through and tried all of these various nets, the GUI never seemed to show many differences between them. This began to make us suspect that the issue was elsewhere.

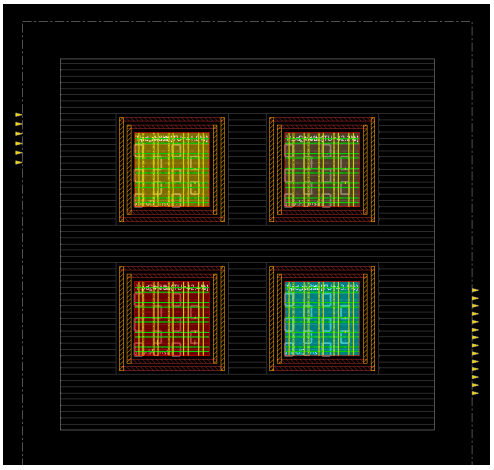


Figure 23: Rings for each Module

4.6.3 globalNetConnect

It turned out that there was a command that we were missing which needed to come before the power routing commands. This command was not included in the lecture slides on low power design or mentioned in the comments of our tcl script and so we did

not know that we were missing it until we got some help from Professor Venkatesh. The professor helped us by providing the *globalNetConnect* commands needed for the top power domain as well as for the module a (figure 24) and we were able to work through the commands for the other modules from there.

```
globalNetConnect VSS -type pgpin -pin VSS -all
globalNetConnect VDDH -type pgpin -pin VDDH -all
globalNetConnect VDDL -type pgpin -pin VDDL -all
globalNetConnect VDDH_gated_moda -type pgpin -pin VDD -powerDomain pd_moda
globalNetConnect VDDH -type pgpin -pin VDD -instanceBasename *moda*HEAD*
globalNetConnect VDDH_gated_moda -type pgpin -pin VDDG -instanceBasename
*moda*HEAD*
```

Figure 24: globalNetConnect

Unfortunately, after testing out many different combinations of parameters for *addRing*, *addStripe*, and *sroute*, we had strayed from our initial setup and had accidentally mixed up the metal layers for the top power domain. Adithya Rajesh was incredibly helpful in explaining this, as well as another issue we were having with the spacing, and with his help we were able to get power routing finished. (figure 25)

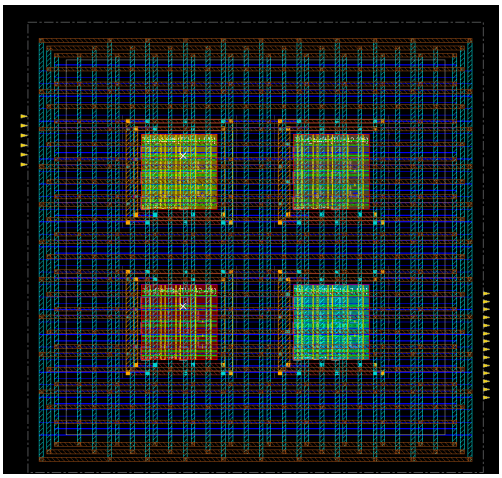


Figure 25: Power Routing Complete

4.7 Tasks 7 & 8 – Place and Route

These tasks proved less challenging than the last one but we did run into some difficulty when trying to look up *placeDesign*. At first we thought the command was not correct since there was no man page associated with it. We used apropos again and found a very similar command *place_design* and decided to use that. While researching *place_design* on the cadence support site we found *place_opt_design* which indicated that it was specifically intended for pre-CTS placement. Using this command as well as *routeDesign* we were able to place the level shifter, isolation cells, etc. and route the entire design. (figure 26)

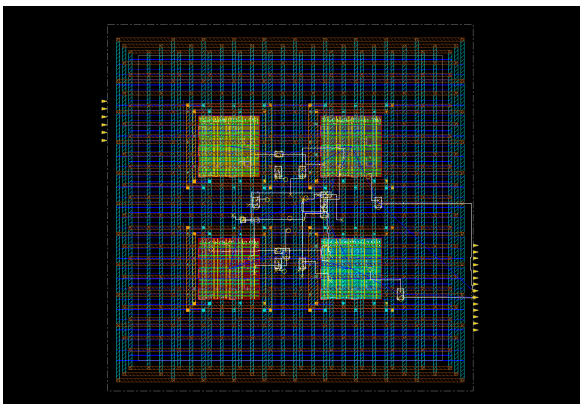


Figure 26: Placement and Route

4.8 Summarize

We iterated through tasks 4.6 and the different power routing command a number of times in an attempt to minimize the number of DRC violations we were seeing. We started with a lot of violations but, ultimately, we were able to reduce it down to only a handful of remaining DRC violations before *place_opt_design* and *routeDesign*. (figure 27) After running *place_opt_design* the tool was able to eliminate the remaining DRC violations and get us down to none.

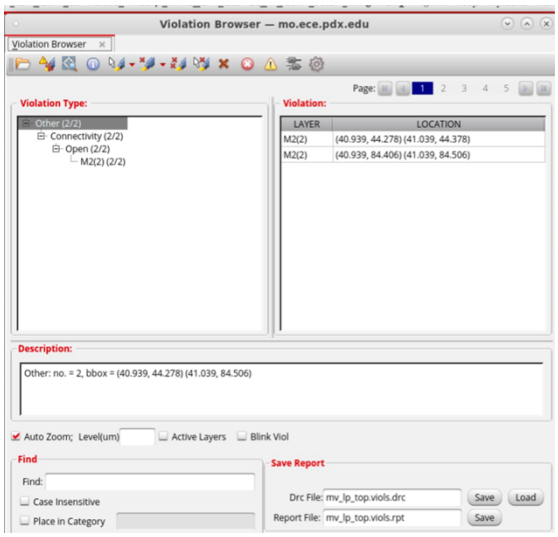


Figure 27: DRC Violations

5. Final Report Teamwork Break Down

Within this project, given our limited team size of two members, we are investing over 40 hours collectively in synchronous Zoom meetings to ensure the project's successful culmination. Our joint efforts are focused on accomplishing both the preliminary draft and the ultimate version of the project report.

6. Timeline

Below is the project complete timeline: the most challenge for our group parts is task 6 Power Routing, Ring, strips and Routing (see details in section 4.6.2)

Description	Complete Date
Run Synthesis on given RTL, UPF and SDC using Synopsys Compiler	17-May
	17-May
The following reports are to be generated in Synopsys Design compiler and posted in tabular form. Don't cut/paste the reports. Extract useful information and then tabulate.	17-May
a. Generated UPF diagram	
b. report_power_domain -nosplit	17-May
c. report_level_shifter -nosplit	17-May
d. report_power_switch -verbose	17-May
e. report_isolation_cell -verbose	17-May
f. report_supply_net	17-May
g. report_pst	17-May
h. *report_qor* (cell summary)	17-May
Run Synthesis on given RTL, UPF and SDC using Cadence Genus	17-May
a. Show Schematic View of the design on Cadence Genus with screen capture.	
b. Highlight the Power management Cells	
Script PNR flow with following requirements:	20-May
a. Top level script: mv_lp_top_run.tcl should reside in apr/scripts.	
This script will be the master script that will perform following functions:	
1. Setup library (paths, points, design_name)	
2. Setup paths for input netlist, upf, sdc	
3. Load the netlist, UPF and constraints.	
4. *init_design, read_power_intent* & *commit_power_intent*	
This could be portion in the Script mv_lp_top_run.tcl or a separate script that you would source in mv_lp_top_run.tcl	23-May
5. Create floorplan for your design with appropriate dimensions using *create_floorplan*	
6. Place Input on Left and Ouput Ports on Right using editPin	
This could be portion in the Script mv_lp_top_run.tcl or a separate script that you would source in mv_lp_top_run.tcl	25-May
7. Update the Power Domains Voltage areas using *modifyPowerDomainAttr* for each domains.	
8. Update floorplan with *planDesign*	
This could be portion in the Script mv_lp_top_run.tcl or a separate script that you would source in mv_lp_top_run.tcl	28-May
9. Add power-switches in your design for each sub-modules using addPowerSwitch . Make sure you are in selected obj using select_obj pd_moda (for moda)	
This could be portion in the Script mv_lp_top_run.tcl or a separate script that you would source in mv_lp_top_run.tcl	6/3/2023 to 6/12/2023
10. Add power Ring, Strips and complete power routing using addRing, addStripe and sroute	
This could be portion in the Script mv_lp_top_run.tcl or a separate script that you would source in mv_lp_top_run.tcl	13-Jun
11. Run Placement and Optimization Pre-CTS using *placeDesign* and *optDesign*	
This could be portion in the Script mv_lp_top_run.tcl or a separate script that you would source in mv_lp_top_run.tcl	13-Jun
12. Complete the PnR flow with routing of all the nets using *routeDesign*	
13. Analyze the design and schematics to check all metal layers used, how the placement of cells and Power Management Cells are and write a brief summary report	14-Jun

Figure 28: Timeline

7. Conclusion

Within this project, we have delved into the diverse obstacles encountered during the implementation of MV design in terms of its Physical Design aspects. Our objective entails maximizing design optimization across all facets and presenting a comprehensive summary at the conclusion of each segment.

8. Challenges Faced

We faced a number of challenges over the course of this project. The bulk of our difficulties (and our learning) definitely was centered on the power routing task. This task and the many combinations of *addRing, addStripe* and *sroute* (and their various parameters) that we tried is where we spent the majority of our time during this project and is responsible for ¾ our all of our log files. With these commands we had difficulty determining the correct layers to use, what to use for the secondary Power net and figuring out that we needed to include *globalNetConnect*. The other main challenges we faced during the course of the project were the missing mmmc.sdc file that we ultimately copied from ORCA_TOP and altered to work with our design and what net to use for the enable net of our power switches.

9. References

Synopsys HDL Compiler for Verilog User Guide
/pkgs/cadence/2020-11/GENUS191/doc/genus_lowpower_legacy/genus_lowpower_legacy.pdf
/pkgs/cadence/2020-11/INNOVUS191/doc/innovusUG/Low_Power_Design.html
evince /pkgs/cadence/2020-11/INNOVUS191/doc/innovusUG/innovusUG.pdf
https://intranet.cecs.pdx.edu/software/synopsys/pdfs/icc2dp.pdf
evince /pkgs/cadence/2019-03/INNOVUS171/doc/innovusTCR/innovusTCR.pdf