PD Submission Doc

Name: Dylan Benzi (dbenzi3)

Discord: benziboy

Part 1

1.1a

```
****************************************
Report : area
Design : chip_top
Version: X-2025.06-SP3
Date   : Sun Oct 11 14:24:51 2026
****************************************

Library(s) Used:

    sky130_tt_1.8_25 (File: /nethome/dbenzi3/physical-design-onboarding-F26/resources/synopsys_libs/sky130_tt_1.8_25.db)
    CF_SRAM_1024x32_tt_180V_25C (File: /nethome/dbenzi3/physical-design-onboarding-F26/inputs/CF_SRAM_1024x32/db/CF_SRAM_1024x32_tt_180V_25C.db)

Number of ports:                         5678
Number of nets:                         13459
Number of cells:                         7825
Number of combinational cells:           6707
Number of sequential cells:              1029
Number of macros/black boxes:               2
Number of buf/inv:                        922
Number of references:                       9

Combinational area:              69871.797988
Buf/Inv area:                     6558.699284
Noncombinational area:           35564.478043
Macro/Black Box area:           235293.578125
Net Interconnect area:      undefined  (No wire load specified)

Total cell area:                340729.854156
Total area:                 undefined
1
```

1.1b

chip_top.gate.v seems to be the complete verilog implementation of every logic gate declared by the RTL

Part 2

2.1a

Halo area per SRAM = (387.870+2(21))*(306.775+2(21)) – (387.870*306.775)

= 30928.343

Halo area for both SRAMs = 30928.343 * 2

= 61856.686

Total cell area w/ halos = total cell area + halo area for both SRAMs

= 402586.541

~50% utilization total area = 805173.082

2:1 dimensions = 1268.9941 x 634.4971

2.2a

```
## Chip Dimension
### TODO adjust to more appropriate values
### heads up: these variables aren't magic!
### make sure you pass them to the appropriate command.
set x_dim 1268.9941
set y_dim 634.4971

## Core Offsets
### Sets an offset on where standard cells can be placed.
### This prevents the tools from placing cells under
### the design's power rings on the edges of the floorplan
set left_offset     9.6
set right_offset    9.6
set top_offset      9.6
set bottom_offset   9.6

# FloorPlan Cmd.
#floorPlan -site CoreSite \
# -s $x_dim $y_dim \
# $left_offset $right_offset $bottom_offset $top_offset
## TODO is this using the right variables?
floorPlan -site CoreSite -adjustToSite \
-s x_dim y_dim \
$left_offset $bottom_offset $right_offset $top_offset
```

2.2b

```
### ===== Place Blocks ===== #
# Edit halo around every block
## reference chip_top.sv
set macro_name_1 data_memory/u_sram
set macro_name_2 instruction_memory/u_sram

set macro_w 387.870
set macro_h 306.775
set halo_buffer 21

set sram_x1 [expr {($x_dim - 2*$macro_w) / 2 + $left_offset}]
set sram_y1 [expr {($y_dim - $macro_h) / 2 + $bottom_offset}]

# add block halos
addHaloToBlock [list $halo_buffer $halo_buffer $halo_buffer $halo_buffer] $macro_name_1
addHaloToBlock [list $halo_buffer $halo_buffer $halo_buffer $halo_buffer] $macro_name_2

placeInstance $macro_name_1 $sram_x1 $sram_y1 -fixed
placeInstance $macro_name_2 [expr {$sram_x1 + $macro_w}] $sram_y1 -fixed
```

2.3a

The macros like SRAM have a large localized power draw. Having a “dedicated” power source close to the macro, like a power ring, reduces parasitics around that portion of silicon. I believe this leads to considerably better stability, which is important especially for a RAM block.

2.3c

```
addRing \
    -around cluster \
    -jog_distance 0.4 \
    -nets {VSS VDD} \
    -type block_rings \
    -layer {met4 met5} \
    -width 6.2 \
    -spacing 3.2 \
    -offset 5
```

2.4a

checkFPlan -reportUtil -outFile util.txt
###
Generated by: Cadence Innovus 25.11-s102_1
OS: Linux x86_64(Host ID ece-rhelsrv)
Generated on: Sun Oct 11 20:27:29 2026
Design: chip_top
Command: checkFPlan -reportUtil -outFile util.txt
###
WARNING (IMPFP-7235): DIE's corner: (1289.0000000000 , 654.0000000000) is NOT on PlacementGrid, Please use command #CUI#getFPlanMode#get_db# to check current Grid settings, use command #CUI#setFPlanMode#set_db# to change which grid to snap to. And use command #CUI#get_snap_grid_info#get_db floorplan_*_grid# to get grids' offset and pitch. Command #CUI#floorPlan#create_floorplan# can be used to fix this issue.
WARNING (IMPFP-10013): Halo should be created around block data_memory/u_sram and snapping rules are not checked since it's width is not multiple of default tech site's width.
WARNING (IMPFP-10013): Halo should be created around block data_memory/u_sram and snapping rules are not checked since it's height is not multiple of default tech site's height.
WARNING (IMPFP-10013): Halo should be created around block instruction_memory/u_sram and snapping rules are not checked since it's width is not multiple of default tech site's width.
WARNING (IMPFP-10013): Halo should be created around block instruction_memory/u_sram and snapping rules are not checked since it's height is not multiple of default tech site's height.
**INFO: No. of regular pre-routes not on tracks : 0

Reporting Utilizations....

Core Utilization = 50.387511
Effective Utilizations
Average macro density = 0.000.

Average module density = 0.205.
Density for the design = 0.205.
      = stdcell_area 55554 sites (105797 um^2) / alloc_area 270912 sites (515925 um^2).
Pin Density = 0.06699.
           = total # of pins 28277 / total area 422127.

Part 3

3.1a

Instance: instruction_memory/u_sram

Arc: CLKin ^ -> DO[17] v

Cell: CF_SRAM_1024x32

Delay: 2.340

Arrival Time: 4.863

Required Time: 1.761

Slack: (1.761-4.863) = -3.102

The SRAM macro likely takes an enormous amount of time compared to other simpler combinational logic. This is because of the sheer amount of gates contained in the macro needed to store the relatively large amount of data. Because of this, we need to adjust the timing path to be at least as slow as the SRAM.

3.2a

```
proc sj_get_tns {} {
  set path [report_timing -collection -max_paths 1]
  get_db $path .slack
}

proc sj_get_wns {} {
  set path [report_timing -collection -max_paths 1]
  get_db $path .worst_delay
}
```

3.2b

TNS: -3.1012

WNS: 2.3401

Part 4

4.1a

Add -gate_clock to the compile_ultra commands in scripts/synthesis/steps/step_compile. Command found in dc_shell-t: man compile_ultra

4.2a

ICG*

I’m assuming this stands for Integrated Gate Clock or something similar

4.2b

ICGPRECX1 is instantiated 32 times in the netlist

4.3a

get_db insts -if {.base_cell.name == ICGPRECX1}

I think there is a way to find clock gating in report_timing, but I found the report_clock_gating_command more helpful

```
#################################################################
#  Generated by:      Cadence Innovus 25.11-s102_1
#  OS:                Linux x86_64(Host ID ece-rhelsrv)
#  Generated on:      Sun Oct 18 15:39:17 2026
#  Design:            chip_top
#  Command:           report_clock_gating_check
#################################################################
    +-------------------------------------------------------------------------------------------------------------------------------+
    |                                                  Clock Gating Check Report                                                    |
    |-------------------------------------------------------------------------------------------------------------------------------|
    |                  Instance                  | Enable | Clock |  Type | Level | Setup | Setup |  Hold |  Hold |       View       |
    |                                            |        |       |       |       |  Rise |  Fall |  Rise |  Fall |                  |
    |--------------------------------------------+--------+-------+-------+-------+-------+-------+-------+-------+------------------|
      cpu/u_fetch/clk_gate_pc_o_reg/latch        |   E    |  CK   |   L   |   L   | 0.083 | 0.245 |   -   |   -   | typical_analysis_view |
      cpu/decode/rf/clk_gate_registers_reg_1_/latch |   E    |  CK   |   L   |   L   | 0.095 | 0.270 |   -   |   -   | typical_analysis_view |
      cpu/decode/rf/clk_gate_registers_reg_2_/latch |   E    |  CK   |   L   |   L   | 0.114 | 0.284 |   -   |   -   | typical_analysis_view |
      cpu/decode/rf/clk_gate_registers_reg_3_/latch |   E    |  CK   |   L   |   L   | 0.095 | 0.270 |   -   |   -   | typical_analysis_view |
      cpu/decode/rf/clk_gate_registers_reg_4_/latch |   E    |  CK   |   L   |   L   | 0.098 | 0.273 |   -   |   -   | typical_analysis_view |
      cpu/decode/rf/clk_gate_registers_reg_5_/latch |   E    |  CK   |   L   |   L   | 0.096 | 0.272 |   -   |   -   | typical_analysis_view |
      cpu/decode/rf/clk_gate_registers_reg_6_/latch |   E    |  CK   |   L   |   L   | 0.096 | 0.272 |   -   |   -   | typical_analysis_view |
      cpu/decode/rf/clk_gate_registers_reg_7_/latch |   E    |  CK   |   L   |   L   | 0.107 | 0.275 |   -   |   -   | typical_analysis_view |
      cpu/decode/rf/clk_gate_registers_reg_8_/latch |   E    |  CK   |   L   |   L   | 0.111 | 0.276 |   -   |   -   | typical_analysis_view |
      cpu/decode/rf/clk_gate_registers_reg_9_/latch |   E    |  CK   |   L   |   L   | 0.096 | 0.270 |   -   |   -   | typical_analysis_view |
      cpu/decode/rf/clk_gate_registers_reg_10_/latch |   E    |  CK   |   L   |   L   | 0.102 | 0.269 |   -   |   -   | typical_analysis_view |
      cpu/decode/rf/clk_gate_registers_reg_11_/latch |   E    |  CK   |   L   |   L   | 0.093 | 0.267 |   -   |   -   | typical_analysis_view |
      cpu/decode/rf/clk_gate_registers_reg_12_/latch |   E    |  CK   |   L   |   L   | 0.085 | 0.254 |   -   |   -   | typical_analysis_view |
      cpu/decode/rf/clk_gate_registers_reg_13_/latch |   E    |  CK   |   L   |   L   | 0.089 | 0.261 |   -   |   -   | typical_analysis_view |
      cpu/decode/rf/clk_gate_registers_reg_14_/latch |   E    |  CK   |   L   |   L   | 0.104 | 0.278 |   -   |   -   | typical_analysis_view |
      cpu/decode/rf/clk_gate_registers_reg_15_/latch |   E    |  CK   |   L   |   L   | 0.095 | 0.269 |   -   |   -   | typical_analysis_view |
      cpu/decode/rf/clk_gate_registers_reg_16_/latch |   E    |  CK   |   L   |   L   | 0.110 | 0.280 |   -   |   -   | typical_analysis_view |
      cpu/decode/rf/clk_gate_registers_reg_17_/latch |   E    |  CK   |   L   |   L   | 0.096 | 0.271 |   -   |   -   | typical_analysis_view |
      cpu/decode/rf/clk_gate_registers_reg_18_/latch |   E    |  CK   |   L   |   L   | 0.104 | 0.280 |   -   |   -   | typical_analysis_view |
      cpu/decode/rf/clk_gate_registers_reg_19_/latch |   E    |  CK   |   L   |   L   | 0.133 | 0.300 |   -   |   -   | typical_analysis_view |
      cpu/decode/rf/clk_gate_registers_reg_20_/latch |   E    |  CK   |   L   |   L   | 0.105 | 0.280 |   -   |   -   | typical_analysis_view |
      cpu/decode/rf/clk_gate_registers_reg_21_/latch |   E    |  CK   |   L   |   L   | 0.090 | 0.264 |   -   |   -   | typical_analysis_view |
      cpu/decode/rf/clk_gate_registers_reg_22_/latch |   E    |  CK   |   L   |   L   | 0.091 | 0.266 |   -   |   -   | typical_analysis_view |
      cpu/decode/rf/clk_gate_registers_reg_23_/latch |   E    |  CK   |   L   |   L   | 0.109 | 0.284 |   -   |   -   | typical_analysis_view |
      cpu/decode/rf/clk_gate_registers_reg_24_/latch |   E    |  CK   |   L   |   L   | 0.097 | 0.273 |   -   |   -   | typical_analysis_view |
      cpu/decode/rf/clk_gate_registers_reg_25_/latch |   E    |  CK   |   L   |   L   | 0.097 | 0.273 |   -   |   -   | typical_analysis_view |
      cpu/decode/rf/clk_gate_registers_reg_26_/latch |   E    |  CK   |   L   |   L   | 0.118 | 0.287 |   -   |   -   | typical_analysis_view |
      cpu/decode/rf/clk_gate_registers_reg_27_/latch |   E    |  CK   |   L   |   L   | 0.104 | 0.279 |   -   |   -   | typical_analysis_view |
      cpu/decode/rf/clk_gate_registers_reg_28_/latch |   E    |  CK   |   L   |   L   | 0.093 | 0.268 |   -   |   -   | typical_analysis_view |
      cpu/decode/rf/clk_gate_registers_reg_29_/latch |   E    |  CK   |   L   |   L   | 0.104 | 0.279 |   -   |   -   | typical_analysis_view |
      cpu/decode/rf/clk_gate_registers_reg_30_/latch |   E    |  CK   |   L   |   L   | 0.093 | 0.268 |   -   |   -   | typical_analysis_view |
      cpu/decode/rf/clk_gate_registers_reg_31_/latch |   E    |  CK   |   L   |   L   | 0.089 | 0.263 |   -   |   -   | typical_analysis_view |
    +-------------------------------------------------------------------------------------------------------------------------------+
```

4.3b

TNS: -2.85901

WNS: 2.3467

Part 5

5a

A faster cell design should mean that the cell delay is smaller (hopefully by a large margin) than the current cell being used. We do need to care because if the cell is “too fast” then the rest of the logic can become the bottleneck, rather than one cell we can pinpoint. We should care about the tradeoff but only if we can prove that it is not beneficial to use such a faster cell.

5b

Corners for an ASIC design correspond to different cases for timing. The corners for ASIC cells TT, SS, and FF are Typical-Typical, Slow-Slow, and Fast-Fast respectively. TT conditions are when both PMOS and CMOS transistors are nominal speed. SS conditions are when both PMOS and CMOS transistors are slower than nominal, and FF is faster. The first letter corresponds to the NMOS speed, and the second the PMOS. Typically, characteristics like temperature and voltage effect these.

5.1a

I added HS libraries by copying the existing library loading syntax, and adding HS equivalents for each TT, SS, and FF.

link_library and target_library were then modified to additionally use HS libraries as defined above.

5.1b

INVX8 U20 (.A(n921), .Y(n50));

INVX8_HS U21 (.A(n921), .Y(n52));

NAND2X1_HS U3 (.A(addr_i[8]), .B(n922), .Y(n37));

NOR3X1_HS U5 (.A(n122), .B(addr_i[6]), .C(addr_i[5]), .Y(n820));

NOR2X2 U52 (.A(n72), .B(n109), .Y(n923));

5.2a

LEF files seem to communicate the physical aspects of the device, including the layout, size, etc. This needs to be used in order to actually place and route the new high speed devices. Otherwise, there is nothing connecting the logical high speed device implementation in the synthesis portion to the silicon implementation.

5.2b

No submission necessary

5.2c

TNS: -2.66515

WNS: 2.3744

Delta TNS 3.2b: |-3.1012 - (-2.66515)| = 0.43605 faster

Delta WNS 3.2b: 2.3401 - 2.3744 = 0.0343 slower

Delta TNS 4.3b: |-2.85901 - (-2.66515)| = 0.19386 faster

Delta WND 4.3b: 2.3467 - 2.3744 = 0.0277 slower

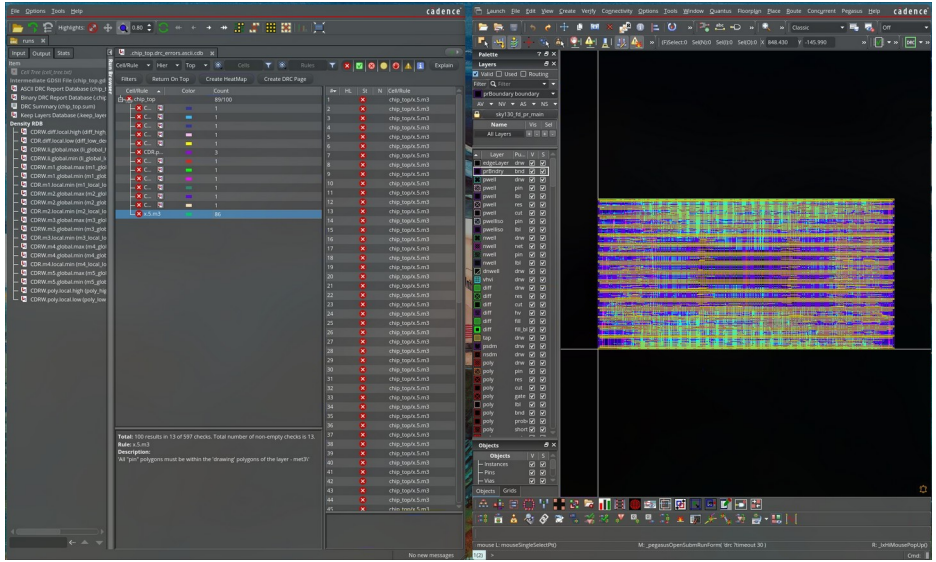
5.2d

TNS has improved while WNS has gotten worse. I am not sure why that is but I assume that is due to only adding high speed modules, while not adjusting anything else.

Part 6

6.1

The most common DRC error is that pins are being placed outside of the defined bounds. There are many other errors related to density, which I am not sure what they are correlated to. I also found errors where the diff doping for some of the transistors in the high speed buffer is too small to be fabbed. Since those were provided by sky130, I am not sure why that would fail DRC.



6.2

Unfortunately, I do have LVS errors in my design. 2 of which relate back to the SRAM, which considering we excluded in the LVS setup makes sense. 32 of which are direct shorts, which may have come from using different synthesis and P&R runs, although I am not certain where the issue came from. I do know that it is seemingly related to the clock gating lines only, so my clock gating command/addition may have been incorrect.

