Intel14 Pads

IP Pads

Official IP from Intel:

- Information about GPIO:
- Download newest version from:

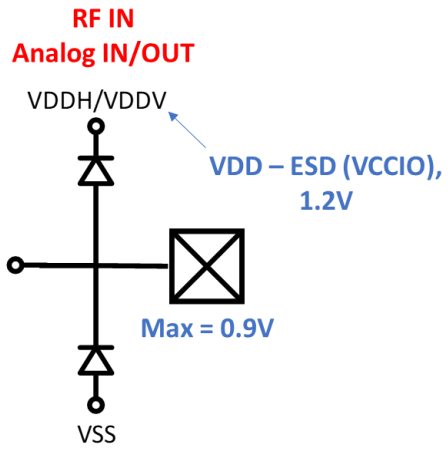
Constructed GPIO Ring:

- cell: /afs/eecs.umich.edu/wics/projects/intel_f23/Cadence/GPIO_TOP/GPIO_quad
    - Note: used sub-cells are from taped out lib top_integration_import, do not make any changes
- CDL netlist for LVS: /afs/eecs.umich.edu/wics/projects/intel_f23/Cadence/cdl_files

Customized Pads

Cell library: /afs/eecs.umich.edu/wics/projects/intel_f23/Cadence/GPIO_TOP
Types of custom pads:

- Pad_rf:
    - RF or analog IO pads. Only has ESD diodes. VDD and VSS are connected to pad supply (VDDV, VDDH = 1.2V)
    - Simulated input capacitance with PEX = 48.7fF
    - Each pad_RF includes:
        - 6 diodes in parallel for p and n diode respectively
        - 6x b88xesddjnu6ogdxnxcnx
        - 6x b88xesddjpu6ogdxnxcnx
    - Place customized pads overlapped with gmb bumps. Latch-up errors might occur.



- Pad_dc:
    - DC IO pad. Same diodes as pad_rf but with more metal interconnect mesh.

Design in Intel14

Launch:

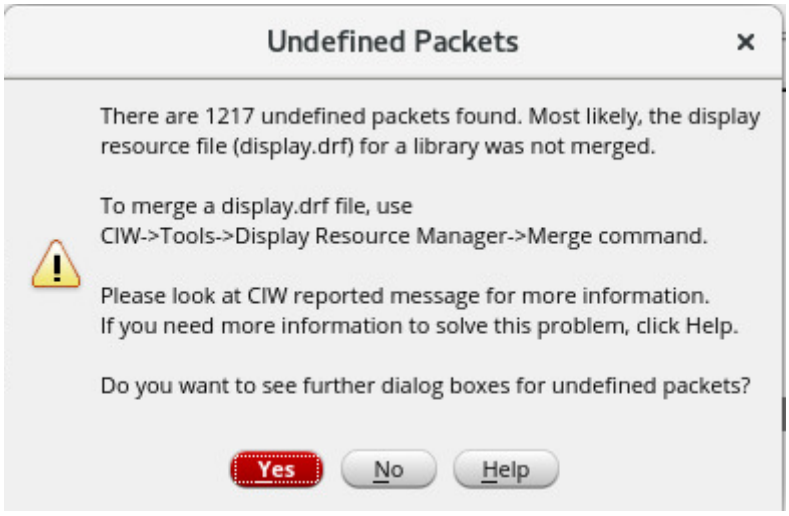
- In local PDK directory:

>tcsh
>source .cshrc_yaswanth
>virtuoso &
Notes: this source file uses 2022 version PDK

- To update to new PDK (2023):

>source .cshrc_lexzww

- Some old pcells are gone or modified. Need to replace them, then DRC and LVS should still pass with **old** cdl netlist
    - Intel22custom lib is changed. Better to copy used pcells (pre-made resistor) into design lib to avoid LVS issues or re-netlist CDL schematic netlist
    - For this, I copied all pre-made TFR cells from old intel22custom lib to my own lib
    - Inductor in new pdk is modeled as resistors in LVS. Replace inductor component to R8 net14 net16 L=1e-09 W=3e-06 $[rgmb] $X=-164355 $Y=-1111680 $D=200
- Need to use new drc rule file
- Don't touch syn/apr Digital standard cells since currently they are generated using old PDK standard cells
- Cadence has a warning on Undefined Packets. Need to be resolved but functionality is not affected.

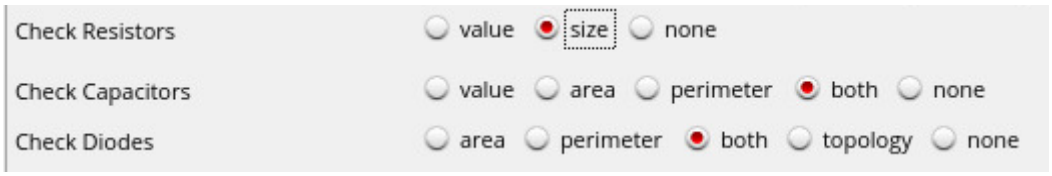


LVS with components from intel22custom lib:

1. Export schematic CDL netlist :
    a. CIW window - File - Export - CDL
    b. Select cellview, CDL netlist (output) file

c. Include intel22custom lib CDL file under "include file" from file path: /afs/eecs.umich.edu/kits/Intel/P1222.4/2022.08/pdk224_r0.9HP3/libraries/custom/cdl/common/intel22custom.cdl
d. Note: new PDK has a similar intel22custom lib but does not include some of the cells in 2022 version. If used pcells from 2022 PDK, need to export with old PDK cdl. /afs/eecs.umich.edu/kits/Intel/P1222.4/2023.09/pdk224_r1.0.3HP1/libraries/custom/cdl/m11_1x_3xa_1xb_1xc_2yb_2ga_mim2_1gb__bumpp/intel22custom.cdl

2. Check the following:



3. When running LVS, uncheck - export from schematic view in Input setting, and run LVS.

Same for everything with standard cells, for example: /afs/eecs.umich.edu/kits/Intel/P1222.4/2023.09/lib224_b0m_6t_108pp_pdk103_r4v5p0_fv/base_hp/cdl/lib224_b0m_6t_108pp_base_hp.cdl

DRC with Synopsys flow on Block Level (Without Density):

1. In any preferred directory,
   >mkdir [blockname_version]
2. From old DRC running directory, /afs/eecs.umich.edu/wics/projects/intel_f23/DRC_Block_Check,
   >cp ./Makefile_103 ./[your directory]/Makefile
   >cp ./.cshrc_103 ./[your directory]/.cshrc_103
3. Put your layout OASIS file to [your directory]
   >cp [your .oas file] ./[your library]
   ***Note: MAKE SURE FILE EXTENSION IS ".oasis"**
4. Back to [your directory] to run DRC
   >cd [your directory]
5. Use text editor of your choice to make the following updates:
   a. Update .cshrc_103 lines 22 & 23 with your block/dir
      ***Note: If something goes wrong later... check .cshrc lines with ###### at end**
   b. Check makefile lines 19 & 37
   c. Might need to change Makefile GATEDIR_VERT to 0
6. Set up environment:
   >source .cshrc_103; modc22s;]
   ***NOTE: IF CHANGE DESIGN, NEED TO SOURCE AGAIN**
7. Run DRC:
   >make drcd

***Note: If you are experiencing crashes, try setting "-host_init 16" in Makefile to smaller number (1 worked).**

- check prompt after command and make sure it's running on the correct layout file

8. Check results in /drcd/[design].LAYOUT_ERRORS
   - To see errors with GUI you can use alias "icvwb" and load the DRC file.

PEX Extraction

1. generate .oas:
    1. Virtuoso command window > File > Export > OASIS
    2. make sure that output file has extension .oasis
2. generate hspice netlist:
    1. create a schematic for the symbol only, and name top level nets
        1. all lowercase names
        2. special symbols (+,-) does not translate well
        3. do not need to include model libraries in this
        4. for elements in intel22custom lib, include cdl file and add auCdl in the Setup - Environment - Switch View List
    2. export top level cell (just the symbol) as .sp file
        1. modify the netlist to leave only the subckts., aka remove last block section
        2. delete TEMP, OPTIONS section
3. setup ICV and starrc and hspice environment:
    1. in ICV directory
        1. tcsh first, run: >> source .cshrc_adf
        2. >> modc22s
4. Modify ICV/Makefile, x3 lines
    1. input_oas, input_sp, DR_DESIGN_NAME
    2. Vim tip for replacing strings:
       >:%s/[old string]/[new string]/g
5. in ICV directory, run LVS:
    1. run >> make pex_lvs
    2. check result at /ICV/pex_lvs_run/xxx.RESULTS: LVE compare results: PASS; ABORT means ICV didn't run correctly
    3. check at pex_lvs_run/milkyway/XTR to verify if design result is here: [design name]:1
6. Modify extraction/star_ccp.custom.cmd file, x4 lines
    1. BLOCK, NETLIST_FILE, NETLIST_IDEAL_FILE, SPICE_SUBCKT_FILE
7. in extraction/
    1. run: >> StarXtract star_ccp.custom.cmd >& starrxt.log
    2. check at starrxt.log for results
        1. check "Run completed successfully" -- PEX is done
        2. [design].tttt_25.spf.typ_nom is the PEX file

<u>After this point, either run manually written script or include NAME.tttt_25.spf.typ_nom in ADE as parasitic file</u>

- Use parasitic file in ADE:
    1. Select Setup - Simulation Files - Parasitic Files (DSPF) - include [design name].tttt_25.spf.typ_nom (.spf file)

2. in simulation log window, should display "input.scs is replaced with [given .spf file]"

Subckt `TIA_diff_3' defined in the file `input.scs' will be replaced with subckt `TIA_diff_3' specified in the DSPF file `/n/brevort/z/lexzww/proj/INTEL16/ICV/extraction/TIA_diff_3.tttt_25.spf.typ_nom'.

3. deselect for pre-PEX simulation

***Note: THIS MIGHT NOT BE ACCURATE ！ ！ ！**

- **Simulate with hspice script:**

1. Build testbench, test run in ADE, and generate hspice netlist
    1. remove all subcircuits that are included in the extracted cell, only leaving the last block
    2. Add line in testbench .sp file:
       .INCLUDE "./NAME.sp"
       .OPTION BA_FILE = "./NAME.tttt_25.spf.typ_nom"
            note: the spf file is the extracted parasitic file
2. mkdir in /tmp/ directory for temporal simulation storage
3. copy files to /tmp/directory
    1. NMAE.sp
    2. /extraction/NAME.tttt_25.spf.typ_nom
    3. tb hspice netlist
4. run hspice simulation:
    1. How to use /tmp/
        1. mkdir in /tmp/
        2. cd there, and source any bash or tcsh file, or load module hspice/2020.12-SP2-1
        3. run: >> hspice -mp 4 -i tb_single_stage_ro_layout.sp
    2. printed results
        1. result saved as: xxx.mt0
        2. in the same directory. >>module load custom-explorer/2015.06, run >> wv & to view result waveforms
        3. printed result can be seen at NAME.mt0
5. To compare with pre-PEX result, comment out .OPTION BA_FILE line

Generating Fills on Block Level

***NOTE: recommended to test run fill generation on design block level, especially for manually made designs, to avoid DRC error between design blocks and fills**

1. make a directory for fill generation /block_fill
2. export block to be filled as .oas file and copy to /block_fill directory
3. edit .cshrc_ file sourced, where FILL is defined
    1. FILL_CONFIG_INPUT_FILE_PATH ./block_fill/[oasis block to be filled]
    2. FILL_CONFIG_INPUT_FILE_TYPE OASIS
    3. FILL_REPORT_FILE ./block_fill
    4. FILL_CONFIG_OUTPUT_FILE_TYPE OASIS
    5. FILL_CONFIG_INPUT_CELL_NAME [design name]

6. FILL_IND true
   ======

1. DR_OUTPUT_FILE dr_run_output.oas
2. DR_INPUT_FILE [same as FILL_CONFIG_INPUT_FILE_PATH]
3. DR_INPUT_CELL [design].oasis
1. **re-source this file if changed**
1. cd block_fill directory and Run >calibre -drc -hier -turbo -hyper $Fill_RUNSET/p1222_fill.svrf
    1. base_fill and metal_fill are generated separately
    2. script handles merging original oas file and fill files into an output file[design name]_fill.oas
    3. if want to import back to cadence: make a new lib and import file [design name]_fill.oasis to cadence through oasis in. Cell [design name] is the main cell with fill. Cell [design name]_fill contains only metal and base fill cells
    4. DRC check in Cadence
2. Before running fil
    1. Manually add blockage layer over active regions or regions don't want to be filled
    2. add boundary layer and pass DRC

Tape Out in Intel14

Import Final Design to Cadence

- setup:
    - create a new library for the import
- Couldn't figure out how to import schematic
- Import Layout as OASIS
    - on CIW window - file - import - OASIS
    - "Stream File" select [design].fill.oas (or any correct design layout oasis file)
    - Select "Library" and "Top Cell" (leave top cell name black to avoid conflict error)
    - "Attach Tech Library" select intel22tech
    - "Layer Map" should be the correct layer map for the PDK version used:
      2022 PDK layermap example:
      /afs/eecs.umich.edu/kits/Intel/P1222.4/2022.08/pdk224_r0.9HP3/libraries/tech/pcell/m11_1x_3xa_1xb_1xc_2yb_2ga_mim2_1gb__bumpp/intel22tech/intel22tech.layermap
    - "Object Map" should be the correct layer map for the PDK version used:
      2022 PDK layermap example:
      /afs/eecs.umich.edu/kits/Intel/P1222.4/2022.08/pdk224_r0.9HP3/libraries/tech/pcell/m11_1x_3xa_1xb_1xc_2yb_2ga_mim2_1gb__bumpp/intel22tech/intel22tech.objectmap
    - Go to "More Options" - "Ref Lib File Name" - (the little pen and paper symbol next to it) - select all standard cell libs (starts with lib224) and all intel22 libs - "Save As and Exit" - "File name" ref_lib.liblist

    - Coloring mode doesn't matter for this process
    - Check "Replace [] with <>" depending on format in .sp schematic netlist
    - Translate
    - Import, and run DRC. If DRC is not clean with drcd in APR step, make sure that the errors are the same
- Run LVS in Cadence
    - For synthesized cell, under ICV/, run >make v2sp
        - This will generate a .sp SPICE schematic netlist with all the sub-cells included
        - copy this file to the local home directory where Virtuoso is run.
    - in Calibre Interactive LVS GUI, under Inputs - uncheck "Export from source viewer" - select generated [design].sp file as SPICE File.
- Notes on combined synthesized and manually design cells:
    - For DRC, DT_15/AT_15/TT_15/MT_15… Will show up errors even when ID layers are on grid. This is because sometimes the synthesized cell is pre-filled with diffCheck layers. If part of the design has a diffCheck layer and part doesn't, those errors will occur. Either delete diffCheck layers in the block (probably the easier solution) or fill the entire block to solve this error.
    - For LVS:
        - generate a symbol and dummy schematic (only include pins in the schematic) for the synthesized block
        - Construct top schematic with correct wiring in Cadence
        - export top schematic to CDL, remember to include intel22custom cdl file
        - at the beginning of exported cdl, add a line .INCLUDE [absolute path for [synthesized block design name].sp]
        - Find and delete synthesized block definition section in the netlist (bc when this block in called as a block in the top level subcircuit, the included [design].sp file will to called instead)
            - Before deleting, check if the pin order is the same in both subsections
            - If not, copy top netlist (where pin order is defined) subsection to synthesized block netlist ([design].sp)
            - After modification on [design].sp, check with synthesized block itself for LVS - should be clean
        - use this exported cdl file as input source schematic for LVS

Place DIC Cell

- Do not need to include DIC cell in schematic netlist for LVS

Top-Level Fills

1. Export top-level layout file as OASIS
    a. Do not need to add reference libs for OASIS OUT
2. Use the same setup as **Generating Fills on Block Level**
3. Now is a good time to check if any standard cells or IP cells are messed up
    a. For detail, see **Tips and Mistakes from Previous Tape Out**

Merge with Frame

Check Different Top-Level DRC Rules:

Cadence runset directory (contains rules files):
/afs/eecs.umich.edu/kits/Intel/P1222.4/2023.09/pdk224_r1.0.3HP1/runsets/calibre/svrf
Synopsys runset directory:
/afs/eecs.umich.edu/kits/Intel/P1222.4/2023.09/pdk224_r1.0.3HP1/runsets/icvtdr/PXL
Documentation is contained in main documentation:
/afs/eecs.umich.edu/kits/Intel/P1222.4/2023.09/pdk224_r1.0.3HP1/doc/Intel_P1222_4_PDK_User_Guide_r1_0_3.pdf

***NOTE: recommended to use synopsys for final top-level DRC check. LVS can be either.**

1. DRC **(includes density)**:
    a. In Cadence:
        i. Rule file: drc.svrf (new version PDK)
        ii. After loading runset, choose this rule file to run DRC
    b. In Synpsys:
        i. Same setup as **DRC with Synopsys flow**
        ii. Run >make drc
2. Collat:
    a. Check for illegal layers, template usage and grid checks
    b. In Cadence:
        i. Rule file: collat.svrf
        ii. After loading runset, choose this rule file to run DRC
    c. In Synpsys:
        i. Same setup as **DRC with Synopsys flow**
        ii. Run >make drc_COLLAT
3. LU:
    a. Check for latch-up and ESD requirements
    b. Does not work in Cadence Calibre Interactive (reasons unknown)
    c. In Synpsys:
        i. Same setup as **DRC with Synopsys flow**
        ii. Run >make drc_LU
4. Antenna:
    a. Checks for process charging in base and metal/via layers. The checks covered in antenna, including: antenna (IPall) checks and Poly jumper rule checks (IPpj).
    b. In Cadence:
        i. Rule file: antenna.svrf
        ii. After loading runset, choose this rule file to run DRC
    c. In Synpsys:
        i. Same setup as **DRC with Synopsys flow**
        ii. Copy Makefile_103_ant, which include make command for antenna check
        iii. Run >make drc_ANT

Tips and Mistakes from Previous Tape Out:

1. need to add fill blockage layer on GPIO L-ring corner
    1. no blockage layer pre-added at lower left corner in GPIO ring
    2. frame contains a fill at the corner that would overlap with fill generated in GPIO ring if blockage layer is not drawn
    3. blockage layer needed:

2. techfeatures in source file: NVM, MIM = 1 to avoid MIM cap related DRC errors
3. COLLAT - illegal layer error:
    1. root cause: need to include reference libraries (all intel22- libs) for OASIS import
    2. otherwise after OASIS OUT, standard IP cells are replaced with non-standard IP cell
    3. okay for circuit cells, but will flag COLLAT errors for GPIO IP cells
    4. if encountered, reimport OASIS for GPIO IP and replace old cells would solve this issue
        i. a quick check: export to OASIS and use calibredrv (module should be loaded in source file), and search for _0 cells in cell list to make sure none exist