**OpenFPGA**

Consider [fpga\_verilog](https://github.com/lnis-uofu/OpenFPGA/tree/master/openfpga_flow/tasks/fpga_verilog) tasks from git repo (https://github.com/lnis-uofu/OpenFPGA/)--

/home/ubuntu/Desktop/OpenFPGA/openfpga\_flow/tasks/fpga\_verilog/verilog\_netlist\_formats/synthesizable\_verilog

My repo for this thesis - <https://github.com/baruaeee/FPGA-OpenFPGA>

Copy ‘synthesizable\_verilog’ to ‘FPGA-OpenFPGA/Fabric’

All command need to be run from the OpenFPGA root folder

Run task command—

run-task /home/ubuntu/Desktop/FPGA-OpenFPGA/Fabric/tasks/synthesizable\_verilog --debug --show\_thread\_logs

the task should be successful. But as I have to change the openfpga architecture file a lot, lets copy the architecture file from “/home/ubuntu/Desktop/FPGA-OpenFPGA/Fabric/tasks/synthesizable\_verilog/latest/k6\_frac\_N10\_tileable\_40nm/and2/MIN\_ROUTE\_CHAN\_WIDTH/arch” to ‘/home/ubuntu/Desktop/FPGA-OpenFPGA/Fabric/tasks/synthesizable\_verilog/’’

cp -r /home/ubuntu/Desktop/FPGA-OpenFPGA/Fabric/tasks/synthesizable\_verilog/latest/k6\_frac\_N10\_tileable\_40nm/and2/MIN\_ROUTE\_CHAN\_WIDTH/arch /home/ubuntu/Desktop/FPGA-OpenFPGA/Fabric/tasks/synthesizable\_verilog/

make a copy of “k6\_frac\_N10\_stdcell\_mux\_40nm\_openfpga\_synthesizable.xml” as “k6\_frac\_N10\_stdcell\_mux\_sky130\_openfpga\_synthesizable.xml”

as the architecture location have changed update it in the task file

line 20/21: openfpga\_arch\_file=${PATH:OPENFPGA\_PATH}/openfpga\_flow/openfpga\_arch/k6\_frac\_N10\_stdcell\_mux\_40nm\_openfpga\_synthesizable.xml

replace with

openfpga\_arch\_file= openfpga\_arch\_file=${PATH:TASK\_DIR}/arch/k6\_frac\_N10\_stdcell\_mux\_sky130\_openfpga\_synthesizable.xml

line 28/29:

arch0=${PATH:OPENFPGA\_PATH}/openfpga\_flow/vpr\_arch/k6\_frac\_N10\_tileable\_40nm.xml

replace with

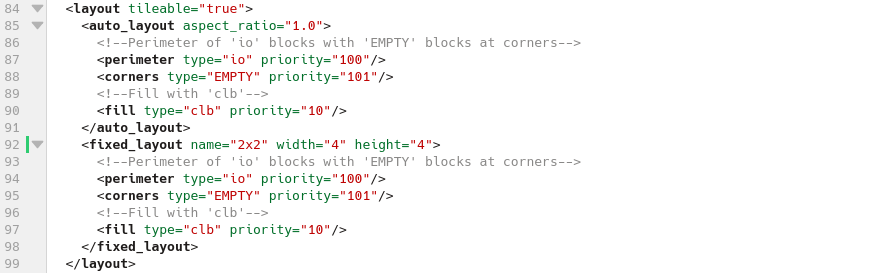
arch0=${PATH:TASK\_DIR}/arch/k6\_frac\_N10\_tileable\_40nm.xml

Run task command again to chack the file location update is corrects—

run-task /home/ubuntu/Desktop/FPGA-OpenFPGA/Fabric/tasks/synthesizable\_verilog --debug --show\_thread\_logs

to get larger grid size the vpr architecture file (k6\_frac\_N10\_tileable\_40nm.xml) need to be edited. Lets open the architecture file—

the layout definition is like the following—



There are 2 layout defined, 1 is auto and another is fixed (2x2). Lets define another fixed layout—

<fixed\_layout name="8x8" width="10" height="10">

<!--Perimeter of 'io' blocks with 'EMPTY' blocks at corners-->

<perimeter type="io" priority="100"/>

<corners type="EMPTY" priority="101"/>

<!--Fill with 'clb'-->

<fill type="clb" priority="10"/>

</fixed\_layout>

Keep width and length moderate, otherwise the compilation will take very long time.

Make a copy of “/home/ubuntu/Desktop/FPGA-OpenFPGA/Fabric/tasks/synthesizable\_verilog/arch/k6\_frac\_N10\_stdcell\_mux\_40nm\_openfpga\_synthesizable.xml”

cp /home/ubuntu/Desktop/FPGA-OpenFPGA/Fabric/tasks/synthesizable\_verilog/arch/k6\_frac\_N10\_stdcell\_mux\_40nm\_openfpga\_synthesizable.xml /home/ubuntu/Desktop/FPGA-OpenFPGA/Fabric/tasks/synthesizable\_verilog/arch/k6\_frac\_N10\_stdcell\_mux\_sky130\_openfpga\_synthesizable.xml

lets open and edit the new copied file—

all standarded cell should be replace with the desired technology library (in this case Sky130)

the library can be found from https://github.com/google/skywater-pdk.git

from OpenFPGA root folder run—

git clone https://github.com/google/skywater-pdk.git

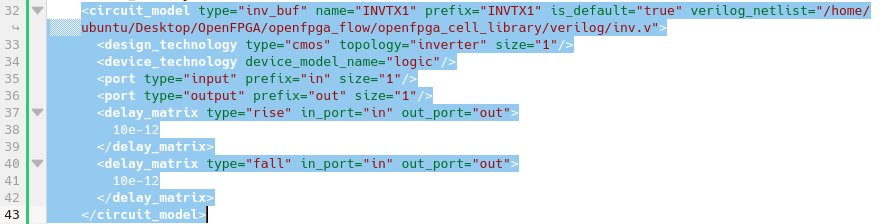
cd skywater-pdk

SUBMODULE\_VERSION=latest make submodules -j3 || make submodules -j1

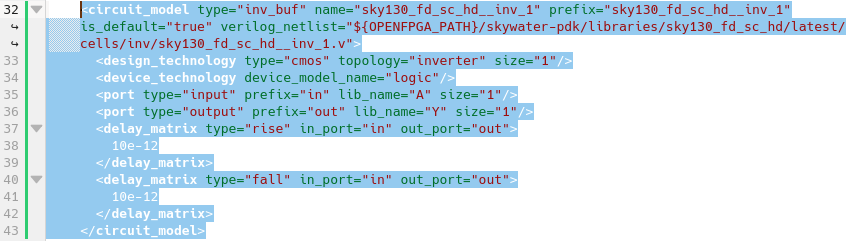
The library location should be ” /home/ubuntu/Desktop/OpenFPGA/skywater-pdk/libraries/”

Replace e.g.

INVTX1:

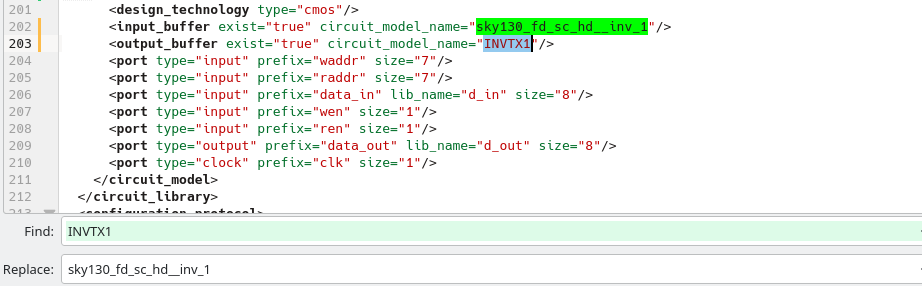


After modification—



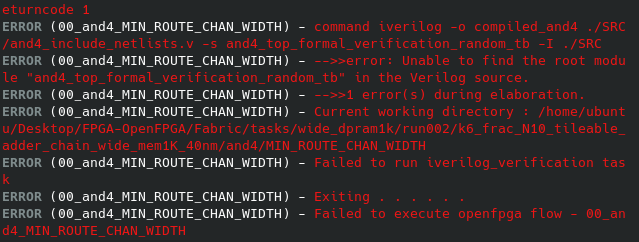
In line 35, 36 the ‘lib name’ should be corresponding to the definition in “sky130\_fd\_sc\_hd\_\_inv\_1.v”

Now look for INVTX1 in the architecture file to replace the remaining definition related to ‘INVTX1’ and rplace it with “sky130\_fd\_sc\_hd\_\_inv\_1”



Now run the task again to check if everything ok.

It should throw some error related to iverilog.



To fix the error navigate to the directory “/home/ubuntu/Desktop/FPGA-OpenFPGA/Fabric/tasks/synthesizable\_verilog/latest/k6\_frac\_N10\_tileable\_40nm/and2/MIN\_ROUTE\_CHAN\_WIDTH/” and open “iverilog\_output.txt”

The line with “…… ……. -I ./SRC” need to be modified as it cannot find the relevant files from sky130 library. The libray location already mentioned in the next line—

“/home/ubuntu/Desktop/OpenFPGA/skywater-pdk/libraries/sky130\_fd\_sc\_hd/latest/cells/inv/sky130\_fd\_sc\_hd\_\_inv\_1.v:34: Include file sky130\_fd\_sc\_hd\_\_inv.v not found

error: Unable to find the root module "and4\_top\_formal\_verification\_random\_tb" in the Verilog source.

: Perhaps ``-s and4\_top\_formal\_verification\_random\_tb'' is incorrect?

1 error(s) during elaboration.

1”

Replace everything with

“iverilog -o compiled\_and2 ./SRC/and2\_include\_netlists.v -s and2\_autocheck\_top\_tb \

-I ${OPENFPGA\_PATH}/skywater-pdk/libraries/sky130\_fd\_sc\_hd/latest/cells/inv

Now run open a terminal in the same folder and run—

source source iverilog\_output.txt

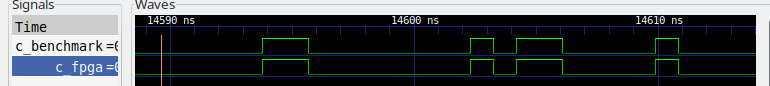
it should run success and then run—

vvp compiled\_and2

the compiled file name can be different depending on the benchmark name. it should work and then run—

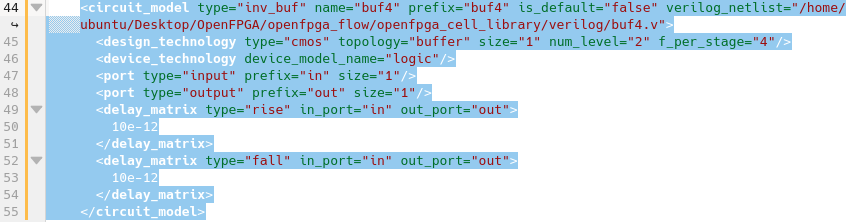
gtkwave and2\_formal.vcd

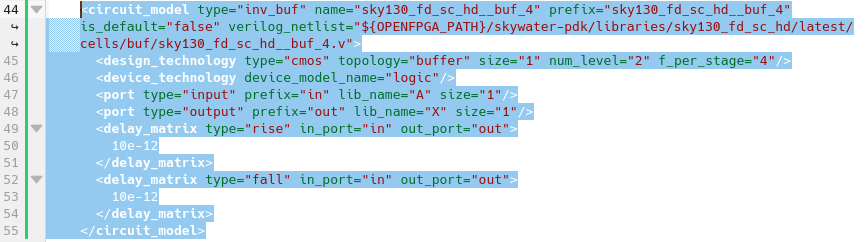
it should open the GTKWave windows where the fpga output can be checked and compared with the bench output.



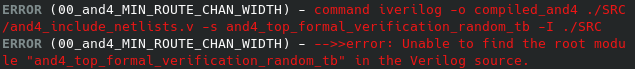
If the both outputs are same, then the std cell binding is working fine.

buf4(inv\_buf):



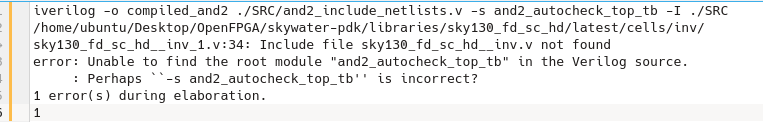


Find and replace “buf4”,” sky130\_fd\_sc\_hd\_\_buf\_4”



Navigate to—

“/home/ubuntu/Desktop/FPGA-OpenFPGA/Fabric/tasks/synthesizable\_verilog/latest/k6\_frac\_N10\_tileable\_40nm/and2/MIN\_ROUTE\_CHAN\_WIDTH/”

è

“iverilog -o compiled\_and2 ./SRC/and2\_include\_netlists.v -s and2\_autocheck\_top\_tb \

-I ./SRC \

-I ${OPENFPGA\_PATH}/skywater-pdk/libraries/sky130\_fd\_sc\_hd/latest/cells/inv \

-I ${OPENFPGA\_PATH}/skywater-pdk/libraries/sky130\_fd\_sc\_hd/latest/cells/buf”

è

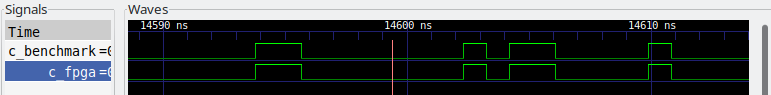
source iverilog\_output.txt

è

vvp compiled\_and2

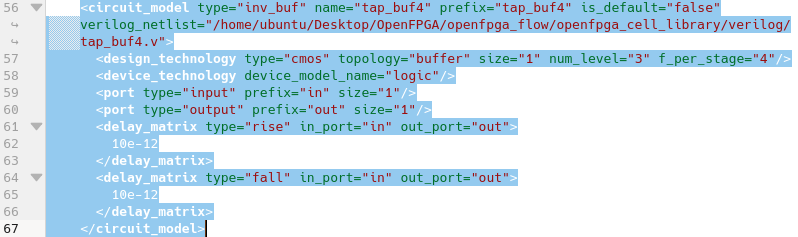
è

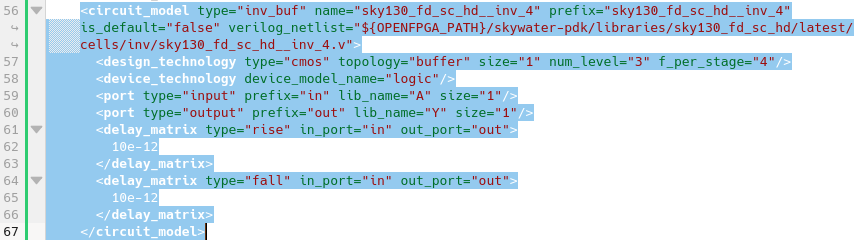
gtkwave and2\_formal.vcd



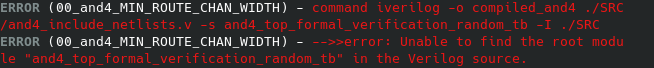
tap\_buf4:

**tap\_buf4** can be interpreted as an inverter with a fanout of 4

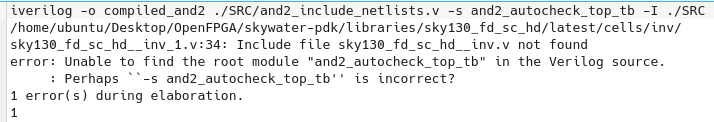




Find and replace “tap\_buf4”, ”sky130\_fd\_sc\_hd\_\_inv\_4”



Navigate to “/home/ubuntu/Desktop/FPGA-OpenFPGA/Fabric/tasks/synthesizable\_verilog/latest/k6\_frac\_N10\_tileable\_40nm/and2/MIN\_ROUTE\_CHAN\_WIDTH/”



è

iverilog -o compiled\_and2 ./SRC/and2\_include\_netlists.v -s and2\_autocheck\_top\_tb \

-I ./SRC \

-I ${OPENFPGA\_PATH}/skywater-pdk/libraries/sky130\_fd\_sc\_hd/latest/cells/inv \

-I ${OPENFPGA\_PATH}/skywater-pdk/libraries/sky130\_fd\_sc\_hd/latest/cells/buf

è

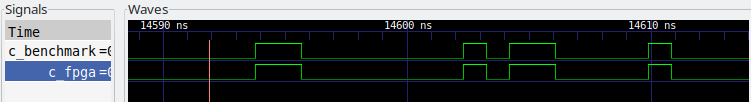
source iverilog\_output.txt

è

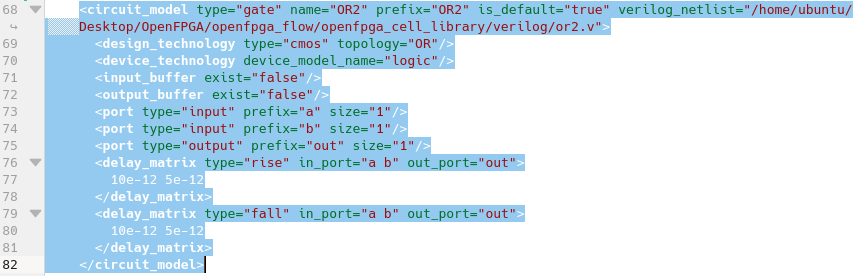
vvp compiled\_and2

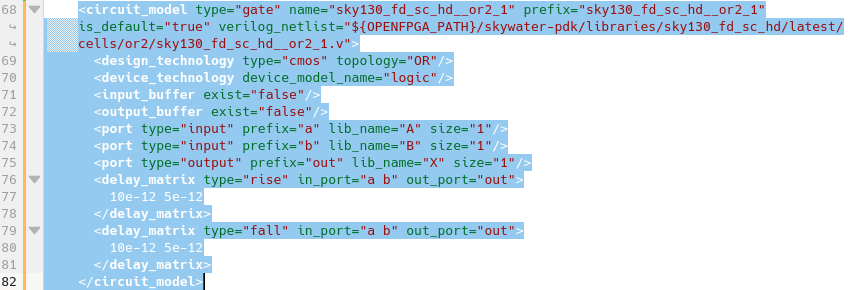
è

gtkwave and2\_formal.vcd



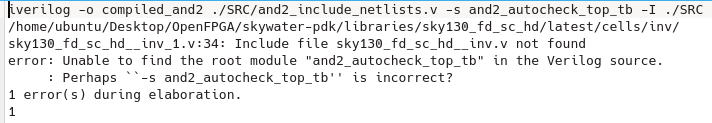
OR2:





Replace “OR2” with " sky130\_fd\_sc\_hd\_\_or2\_1"

Navigate to “/home/ubuntu/Desktop/FPGA-OpenFPGA/Fabric/tasks/synthesizable\_verilog/latest/k6\_frac\_N10\_tileable\_40nm/and2/MIN\_ROUTE\_CHAN\_WIDTH/” and open “iverilog\_output.txt”



è

iverilog -o compiled\_and2 ./SRC/and2\_include\_netlists.v -s and2\_autocheck\_top\_tb \

-I ./SRC \

-I ${OPENFPGA\_PATH}/skywater-pdk/libraries/sky130\_fd\_sc\_hd/latest/cells/inv \

-I ${OPENFPGA\_PATH}/skywater-pdk/libraries/sky130\_fd\_sc\_hd/latest/cells/buf \

-I ${OPENFPGA\_PATH}/skywater-pdk/libraries/sky130\_fd\_sc\_hd/latest/cells/or2

è

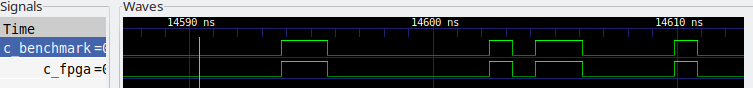
source iverilog\_output.txt

è

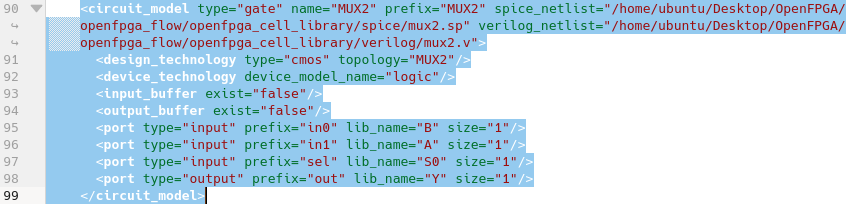
vvp compiled\_and2

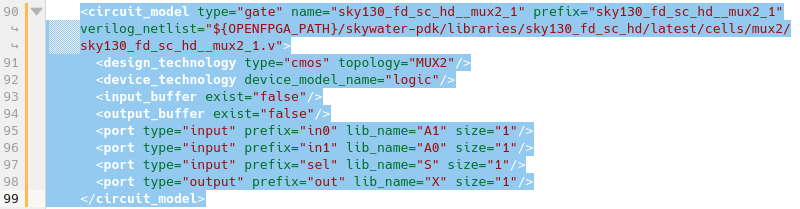
è

gtkwave and2\_formal.vcd



MUX2 (gate):

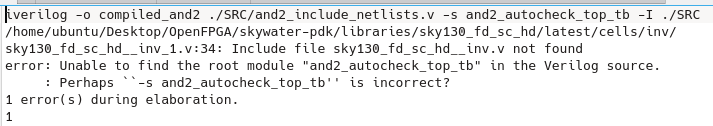




replace-"MUX2"==>"sky130\_fd\_sc\_hd\_\_mux2\_1"

as the topology is also “MUX2”, don’t replace it.

Navigate to “/home/ubuntu/Desktop/FPGA-OpenFPGA/Fabric/tasks/synthesizable\_verilog/latest/k6\_frac\_N10\_tileable\_40nm/and2/MIN\_ROUTE\_CHAN\_WIDTH/” and open “iverilog\_output.txt”



è

iverilog -o compiled\_and2 ./SRC/and2\_include\_netlists.v -s and2\_autocheck\_top\_tb \

-I ./SRC \

-I ${OPENFPGA\_PATH}/skywater-pdk/libraries/sky130\_fd\_sc\_hd/latest/cells/inv \

-I ${OPENFPGA\_PATH}/skywater-pdk/libraries/sky130\_fd\_sc\_hd/latest/cells/buf \

-I ${OPENFPGA\_PATH}/skywater-pdk/libraries/sky130\_fd\_sc\_hd/latest/cells/or2 \

-I ${OPENFPGA\_PATH}/skywater-pdk/libraries/sky130\_fd\_sc\_hd/latest/cells/mux2

è

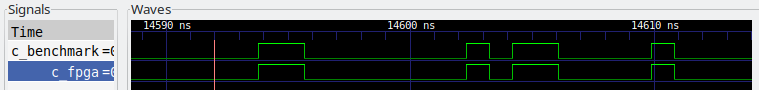
source iverilog\_output.txt

è

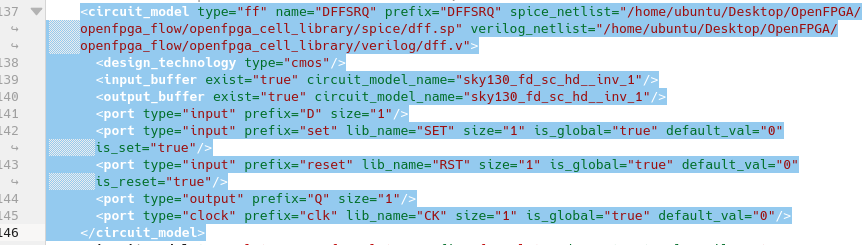
vvp compiled\_and2

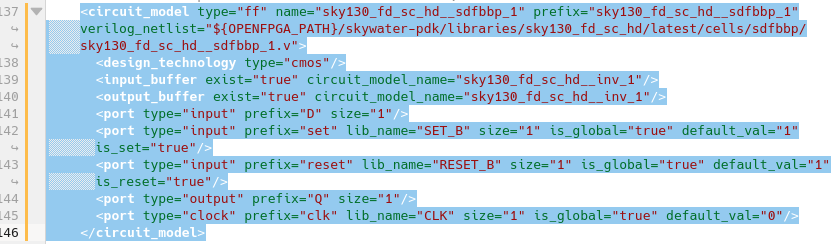
è

gtkwave and2\_formal.vcd



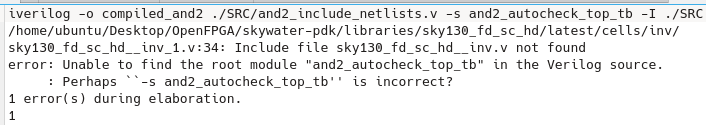
DFFSRQ:





replace-DFFSRQ==>sky130\_fd\_sc\_hd\_\_sdfbbp\_1

Navigate to “/home/ubuntu/Desktop/FPGA-OpenFPGA/Fabric/tasks/synthesizable\_verilog/latest/k6\_frac\_N10\_tileable\_40nm/and2/MIN\_ROUTE\_CHAN\_WIDTH/” and open “iverilog\_output.txt”



è

iverilog -o compiled\_and2 ./SRC/and2\_include\_netlists.v -s and2\_autocheck\_top\_tb \

-I ./SRC \

-I ${OPENFPGA\_PATH}/skywater-pdk/libraries/sky130\_fd\_sc\_hd/latest/cells/inv \

-I ${OPENFPGA\_PATH}/skywater-pdk/libraries/sky130\_fd\_sc\_hd/latest/cells/buf \

-I ${OPENFPGA\_PATH}/skywater-pdk/libraries/sky130\_fd\_sc\_hd/latest/cells/or2 \

-I ${OPENFPGA\_PATH}/skywater-pdk/libraries/sky130\_fd\_sc\_hd/latest/cells/mux2 \

-I ${OPENFPGA\_PATH}/skywater-pdk/libraries/sky130\_fd\_sc\_hd/latest/cells/sdfbbp

è

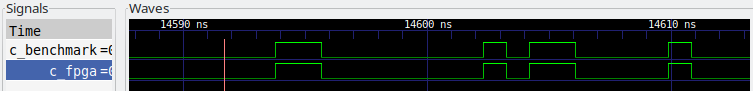
source iverilog\_output.txt

è

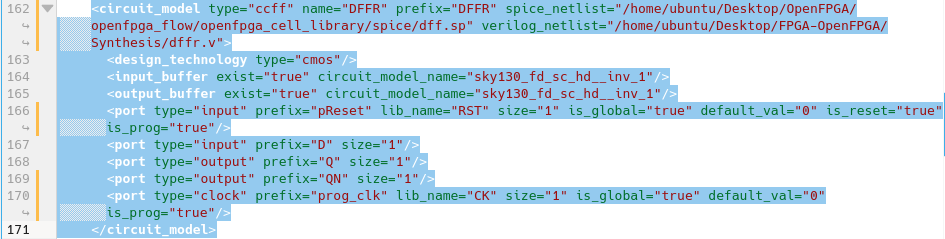
vvp compiled\_and2

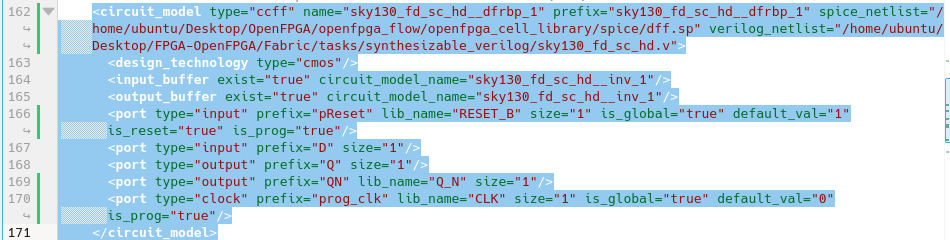
è

gtkwave and2\_formal.vcd



DFFR:





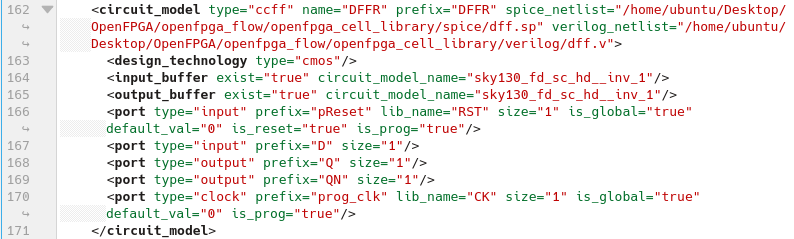
replace-DFFR==>sky130\_fd\_sc\_hd\_\_dfrbp\_1

this part cannot be compiled with iverilog because of limitation of iverilog (probably it cann’t handle delayed d input and throw some internal error that cannot be identified from OpenFPGA iverilog wrapped)

to confirm it I have to run the dff.v (from OpenFPGA/openfpga\_flow/openfpga\_cell\_library/verilog/) with a testbench with a separately synthesized netlist.

**Synthesis**

As the synthesis already done within the OpenFPGA process no synthesis is necessary. But during binding the OpenFPGA architecture file with Sky130 I had to leave the “DFFR” scan chain flip-flop as there is no suitable Standard cell in the cell library. The nearest equivalent was with inverted reset did’nt passed the compilation.



Now I have to synthesize (with sky130 librery) the ‘dff.v’ only to include in the fabric\_netlists.v file. As I am going to use this for test purpose, time constraints are not considered and running through interactive mode in Cadence Genus. For the lib files follow the steps in <https://github.com/baruaeee/ASIC_ADC/blob/main/MyProject/Installation_Details.docx>

genus -gui

read\_libs sky130\_fd\_sc\_hd\_tt\_025C\_1v80.lib

read\_hdl dff.v

elaborate DFFR

set\_db syn\_generic\_effort medium

set\_db syn\_map\_effort medium

set\_db syn\_opt\_effort medium

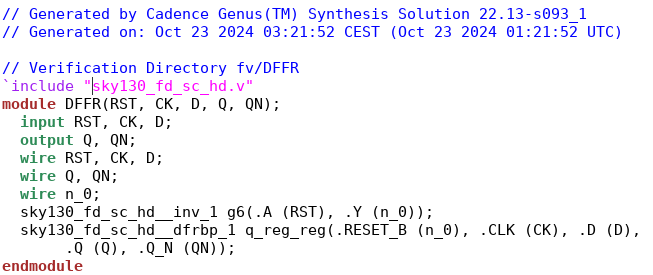
syn\_generic

syn\_map

syn\_opt

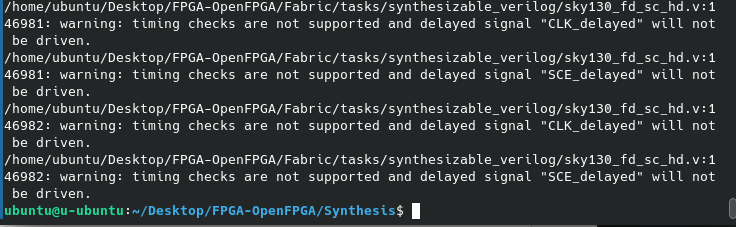
write\_hdl > dffr.v

the content of the synthesized file--



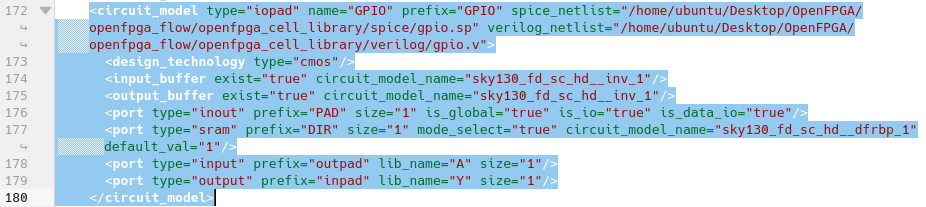
The testbench can be found at <https://github.com/baruaeee/FPGA-OpenFPGA/blob/main/Synthesis/dffr_tb.v>

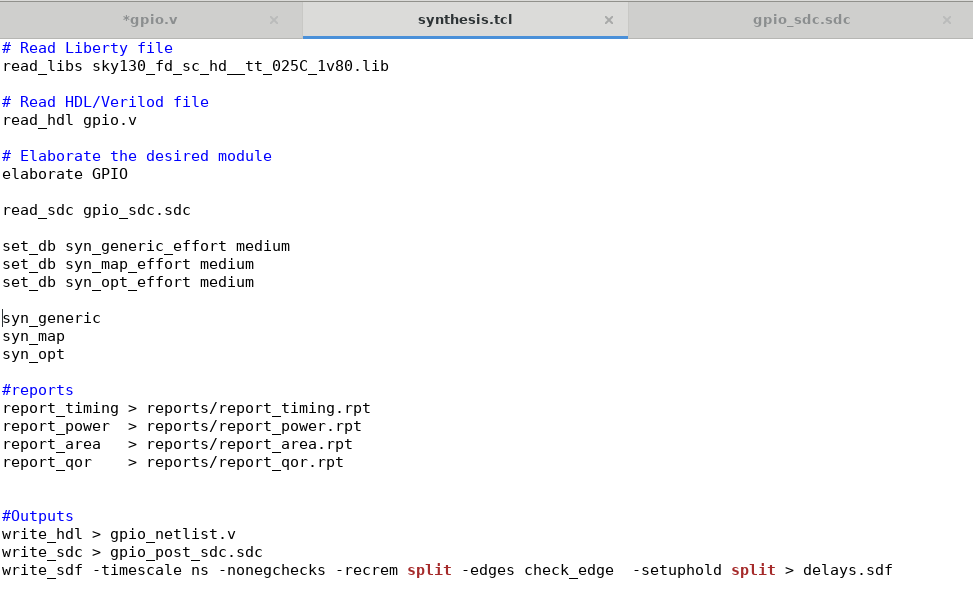
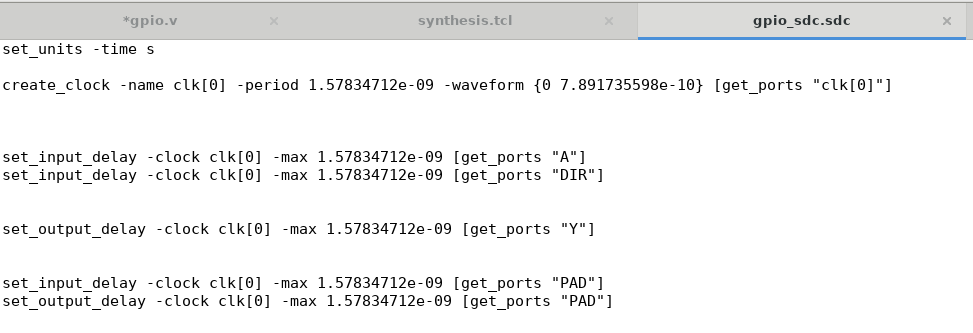
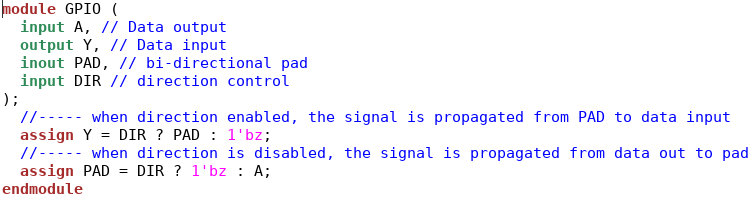
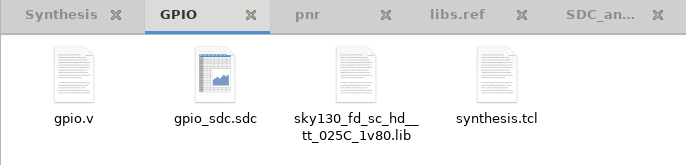
If I run the testbench with iverilog it throws bunch of errors which can’t be seen during compilation from OpenFPGA processes.

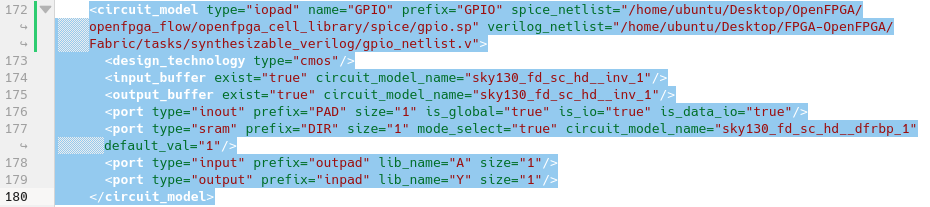
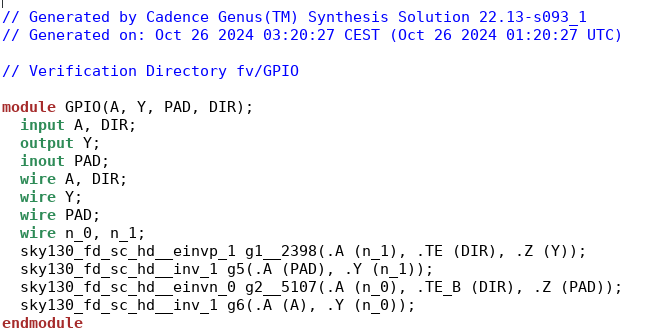


But the fabric netlist generated by OpenFPGA is correct.

**GPIO:**

The gpio.v need to be synthesized to properly bind with sky130.

the sdc file is not useful, as the verilog design has no clock defined.



However the above is a generic GPIO. As Sky130 has a well defined GPIO library, let try to bind it with the fabric netlist. The procedure will be the same. I am just explaining the custom verilog code to integrate it.

*//-----------------------------------------------------*

*// Design Name : General Purpose I/Os*

*// File Name : gpio\_sky130\_v\_1.v*

*// Coder : Sajib Barua*

*//-----------------------------------------------------*

*//-----------------------------------------------------*

*// Function : A typical general purpose I/O*

*//-----------------------------------------------------*

*module GPIO (*

*input A, // Data output => write to external/PAD || direction A to PAD*

*output Y, // Data input => read from external/PAD || direction PAD to Y*

*inout PAD, // bi-directional pad*

*input DIR //----- when direction enabled(1), the signal is propagated from PAD to data input(Y)*

*);*

*wire tie\_hi\_esd, tie\_lo\_esd;*

*sky130\_fd\_io\_\_top\_gpio\_ovtv2 gpio(*

*// Direction PAD to IN => input buffer enable*

*// REF. assign IN = (x\_on\_in\_lv ===1 || pwr\_good\_inpbuff\_lv===0) ? 1'bx : (disable\_inp\_buff\_lv===1 ? 0 : (^PAD===1'bx ? 1'bx : PAD));*

*.IN(Y),*

*.INP\_DIS(~DIR), // when DIR==1(INP\_DIS==0), the signal is propagated from PAD to Y(IN)||input buffer enbled (active low)*

*.ENABLE\_INP\_H(tie\_hi\_esd), // enable input buffer powr on/off. hard-tie to either tie\_hi\_esd or tie\_lo\_esd and should not be register controlled.*

*.TIE\_HI\_ESD(tie\_hi\_esd), // connected to VDDIO/Logic High*

*.TIE\_LO\_ESD(tie\_lo\_esd), // connected to VSSIO/Logic Low*

*.VTRIP\_SEL(tie\_lo\_esd), // define input buffer voltage 30%/70% of Vddio.*

*.HYS\_TRIM(tie\_lo\_esd), // input buffer ref. signaling voltage < 2.2v selected*

*.IB\_MODE\_SEL(2'b00), // input buffer mode CMOS selected*

*// .VINREF // only used when IB\_MODE\_SEL(10/11) selected*

*// Direction OUT to PAD => Output buffer enable*

*// REF. out\_final*  *<= (^out\_buf === 1'bx || !pwr\_good\_hold\_ovr\_mode) ? 1'bx : out\_buf;*

*// if (^ENABLE\_H===1'bx || !pwr\_good\_hold\_mode || (ENABLE\_H===1 && (^hld\_h\_n\_buf===1'bx ||(hld\_h\_n\_buf===0 && hld\_ovr\_final===1'bx)||(hld\_h\_n\_buf===1 && hld\_ovr\_final===1'bx)))) ------- this conditions restrict output buffer flow*

*.OUT(A),*

*.OE\_N(DIR), //when DIR==0, the signal is propagated from OUT to pad|| outpu buffer enabled (active low)*

*.HLD\_H\_N(tie\_hi\_esd), // When hld\_h\_n=1, HLD\_OVR signal is ignored (active low)*

*.ENABLE\_H(tie\_hi\_esd), // The enable\_h signal takes priority over the hld\_h\_n signal|| PAD I/O enable*

*.ENABLE\_VDDA\_H(tie\_lo\_esd), // The master enable signal to the I/O-cell’s analog section (1'b0 =>disabled)*

*.ENABLE\_VDDIO(tie\_hi\_esd), // I/O pwr supply enabled*

*.ENABLE\_VSWITCH\_H(tie\_lo\_esd), // vswitch disabled*

*.SLOW(tie\_lo\_esd), // output slew rate default selected*

*// .SLEW\_CTL, // needed only for I2C config*

*.HLD\_OVR(tie\_lo\_esd), // // No hold override (normal)*

*.ANALOG\_EN(tie\_lo\_esd), // Analog mode disabled*

*.ANALOG\_SEL(1'bx), // Analog mode disabled*

*.ANALOG\_POL(1'bx), // Analog mode disabled*

*.DM(3'b110), // Strong pull-up/down mode*

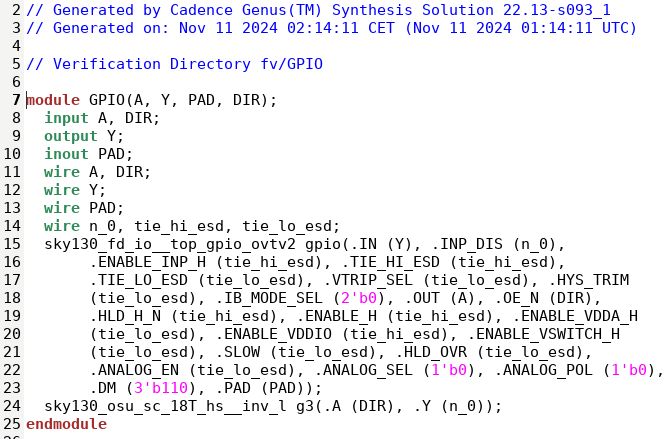
*.PAD(PAD)*

*);*

*endmodule*

The explanation of the port mapping are provided in the comments.

After synthesis--



The compilation it a bit tricky, because of the complexity of Sky130 GPIO. Without proper reference of the pdk verilog library (both functional and behavioral) the result could be confusing.

Lets try the compilation in Cadence XCELIUM.

**Compile in Cadence XCELIUM:**

This is the functional netlist verification. If the synthesis could be done with Genus, a Logical Equivalent Check (LEC) can be done using CONFRML. As synthesis has been already done inside the OpenFPGA process and we have data for functional verification, for now skip the LEC.

The XCELIUM tools (e.g. xmvlog) do not work properly if the ‘include’ files are out of the working directory. For sky130 the complete PDK directory is about 12GB. To avoid copying all library files I will only copy the necessary files/folders to the working directory.

Here are the view of the task directory.



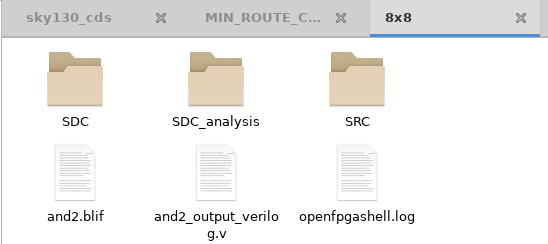
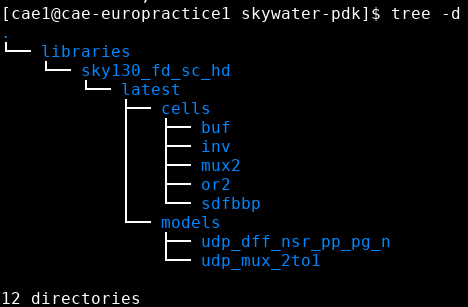
Lets make ‘8x8’ directory for compilation. I will delete all unnecessary files for simplicity.

Without modification, “/home/cae1/Desktop/FPGA-OpenFPGA/Fabric/tasks/synthesizable\_verilog/8x8/k6\_frac\_N10\_tileable\_40nm/and2/MIN\_ROUTE\_CHAN\_WIDTH/” is the working directory. Lets move everything to ‘8x8’ directory and delete everything.

è

mv /home/cae1/Desktop/FPGA-OpenFPGA/Fabric/tasks/synthesizable\_verilog/8x8/k6\_frac\_N10\_tileable\_40nm/and2/MIN\_ROUTE\_CHAN\_WIDTH/\* .

after modifying it should look like following—



Copy ‘skywater-pdk’ from ‘/home/cae1/Desktop/FPGA-OpenFPGA/Fabric/tasks/synthesizable\_verilog/8x8/skywater-pdk/libraries/sky130\_fd\_sc\_hd/latest/cells/’. It should look like following—

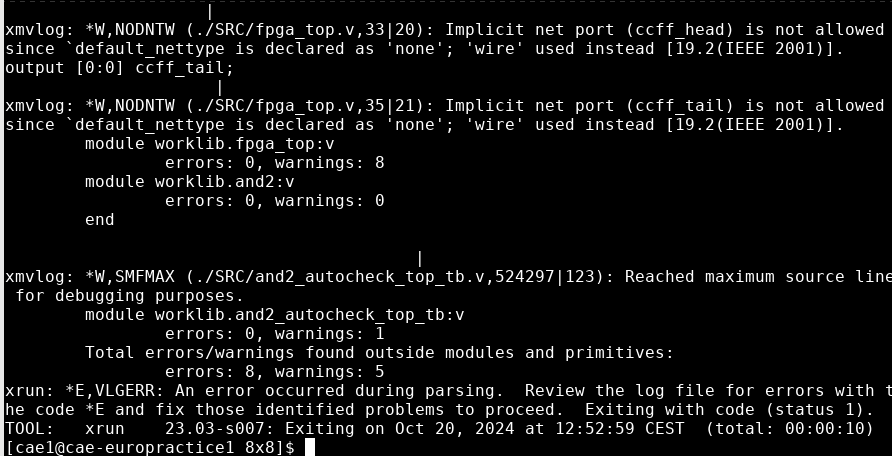


Open terminal from ‘8x8’. Let run the ‘xrun’ without any modification of the verilog files.

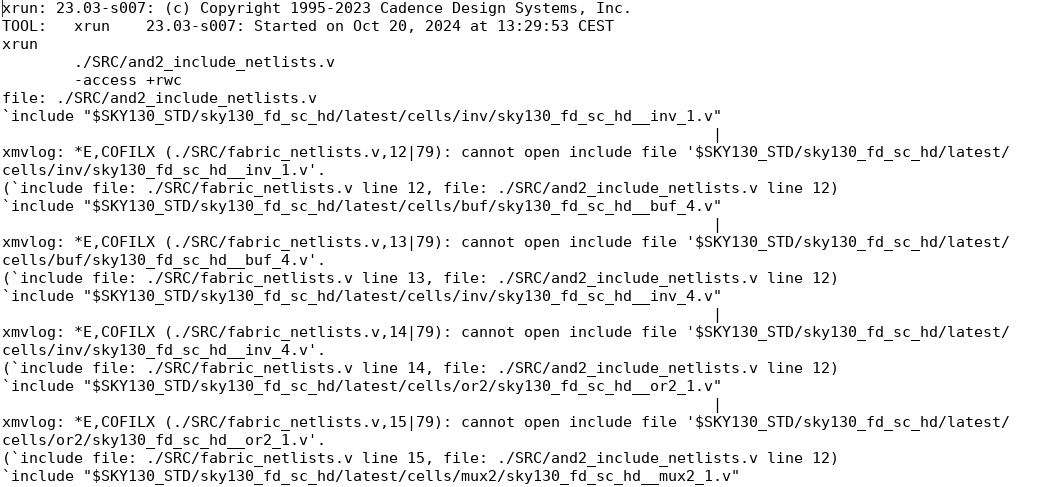
è

xrun ./SRC/and2\_include\_netlists.v -access +rwc

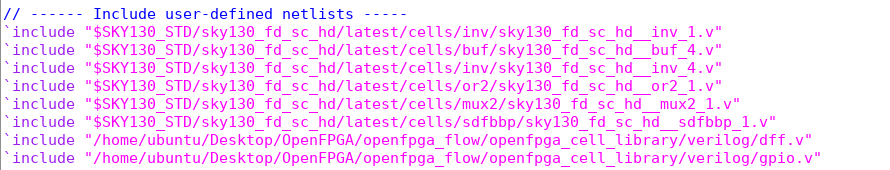
as the ‘and2\_include\_netlists.v’ file has include reference to sky130-pdk, it will throw a lots of errors.



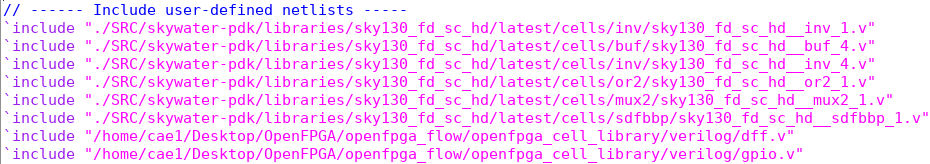
The details on these errors can be found in “/home/cae1/Desktop/FPGA-OpenFPGA/Fabric/tasks/synthesizable\_verilog/8x8/xrun.log”



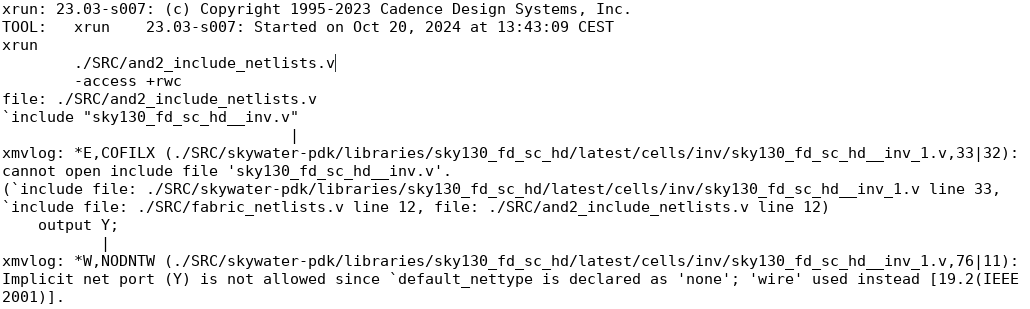
Now I have to fix all the errors and run the ‘xrun’ command on and on.



Even though the files are in the denoted location, still ‘xrun/xmvlog’ can’t detect. Let’s modify these lines—



After running the ‘xrun’ command—



It cannot find the include verilog files annoted in the ‘sky130\_fd\_sc\_hd\_\_inv\_1.v’ and similarly other corresponding files. To fix this add use ‘-include’ in the xrun command to include the relevant directories. Without 64bit mode the compilation throws an error related to virtual memory.

xrun ./SRC/and2\_include\_netlists.v -access +rwc \

-incdir ./SRC/sky130/cells/inv \

-incdir ./SRC/sky130/cells/buf \

-incdir ./SRC/sky130/cells/or2 \

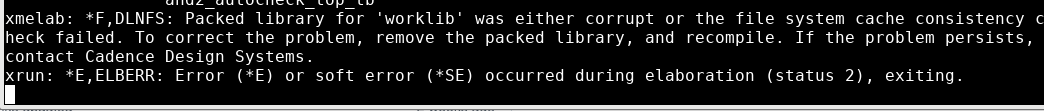
-incdir ./SRC/sky130/cells/mux2 \

-incdir ./SRC/sky130/cells/sdfbbp \

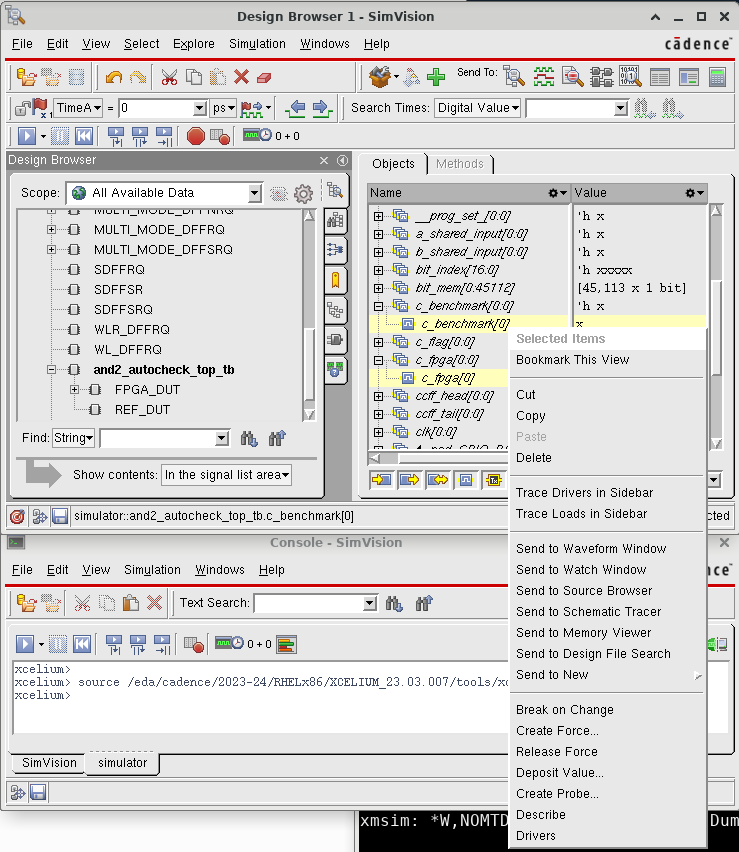
-64bit \

-gui &

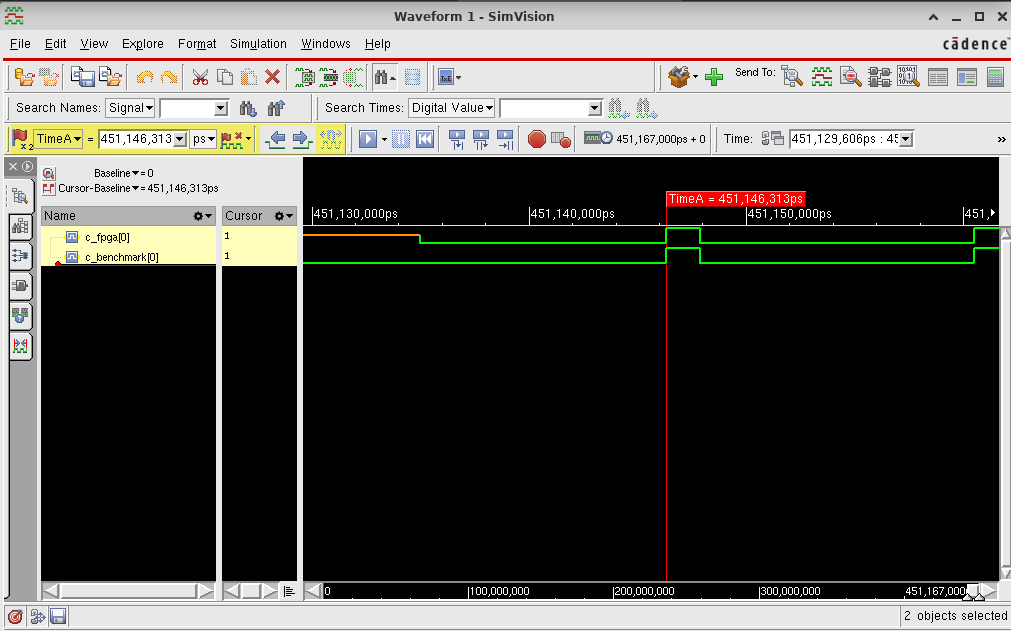
It throwing some error without details specifics.



If similar procedure done in ‘6x6’ è 8x8 grid fabric it works.



Use ‘Send to Waveform Window’—



**Synthesis**

Before importing the design to Innovus I have done a Resynthesis with genus to condition the SDC files. As the SDC files generated by OpenFPGA are not perfectly readable in Genus, I have written a python script to combine and edit the SDC files automatically and quickly. It will normalize a lots of errors described in the next part (Place and Toute). The synthesis is done with a simple tcl script.

**Place and Route (Physical Design)**

Input/output files—

Synthesized gate level netlist (\*.v file, from OpenFPGA or Genus)

Physical library (\*.lef file for Innovus)

IO pad assignment file (\*.io -- optional)

Timing library (\*.lib file)

Constraint file (\*.sdc file)

Innovus either can be run with TCL script or in gui.

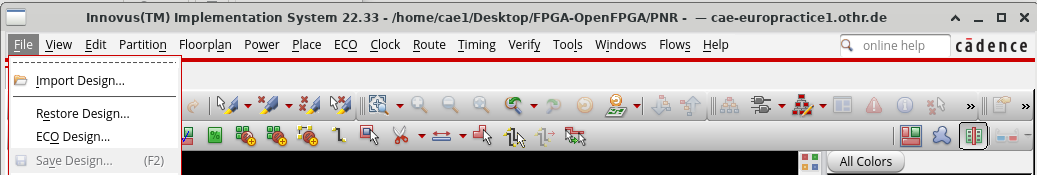
Design import—

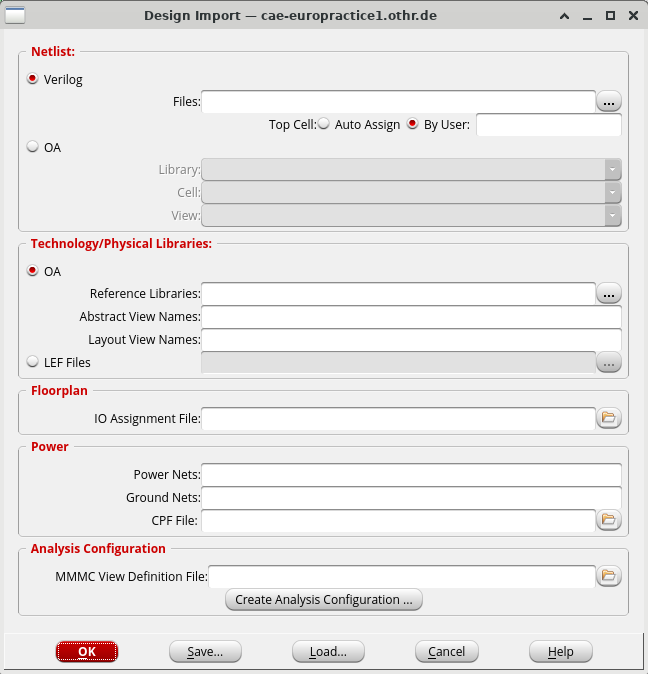
1. Netlist (from synthesis)
2. Physical library (\*.lef)
   1. Technology lef – metals/via (e.g. sky130\_ef\_sc\_hd.lef)
   2. Standard cell lef -- (e.g. sky130\_fd\_sc\_hd.lef)
3. Floorplan –
4. Power—VDD, VSS
5. Analysis configuration—MMMC setup

==>

Innovus

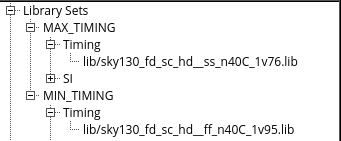
==>

Import design:

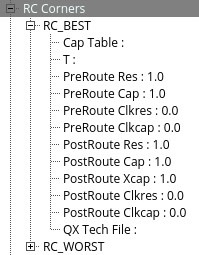


Select 

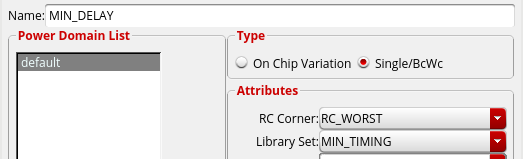
Create 2 library set, 1 for max and aonther for min. 

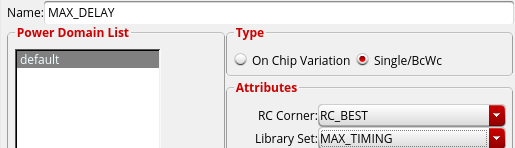


Create 2 RC Corner, 1 for best and another for worst. 

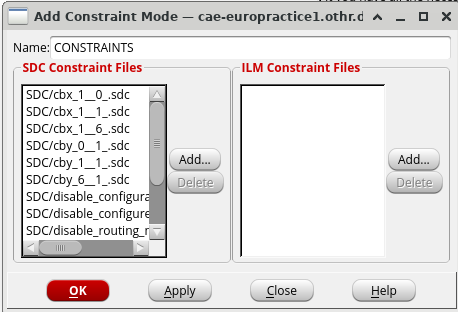
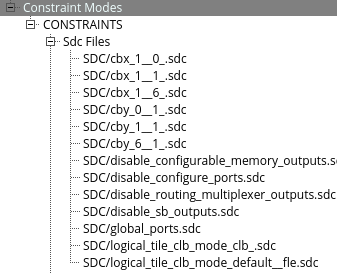


Create 2 Delay Corners 1 for max delay and another min dilay. 

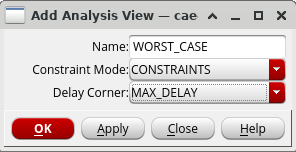
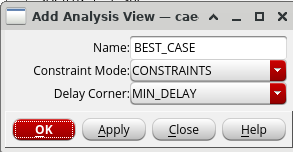




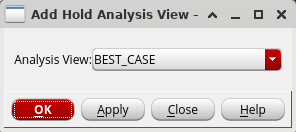
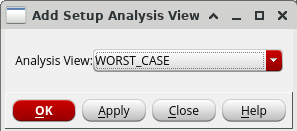
Create Constraint Mode. 



Create Analysis Views, 1 for best case and another for worst case. 



Create Setup and Analysis Views. 



Save & close.

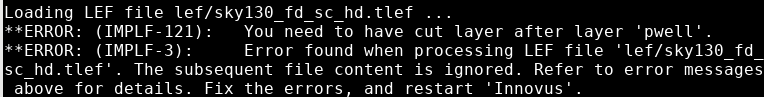
After including every files--

The working directory and its contents can be found <https://github.com/baruaeee/FPGA-OpenFPGA/tree/main/PNR>

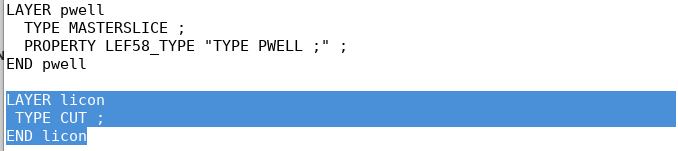
For innouvus the sky130\_fd\_sc\_hd.v file is not necessary. Including this in the fabric netlist can invite unwanted syntax errors. So it can be removed or commented in the fabric\_netlists.v file.

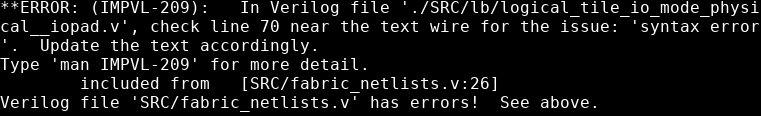


It can throw some error relater to ‘pwell’--

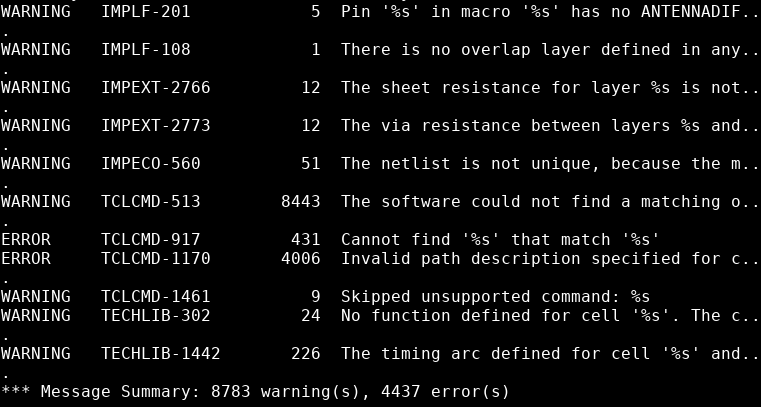
To fix the error open the ’sky130\_fd\_sc\_hd.tlef’ file and add the folloing after pwell layer dfinition.

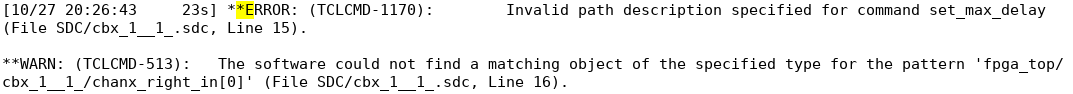
LAYER licon  
 TYPE CUT ;  
END licon

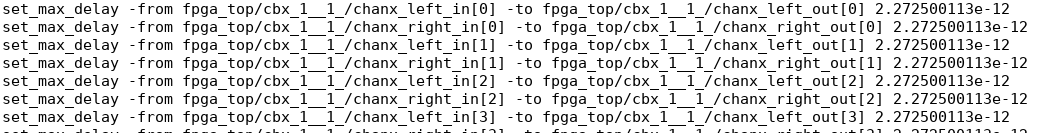
Next it can throw some syntax erro due to ”`default\_nettype wire” in the structural verilog file.

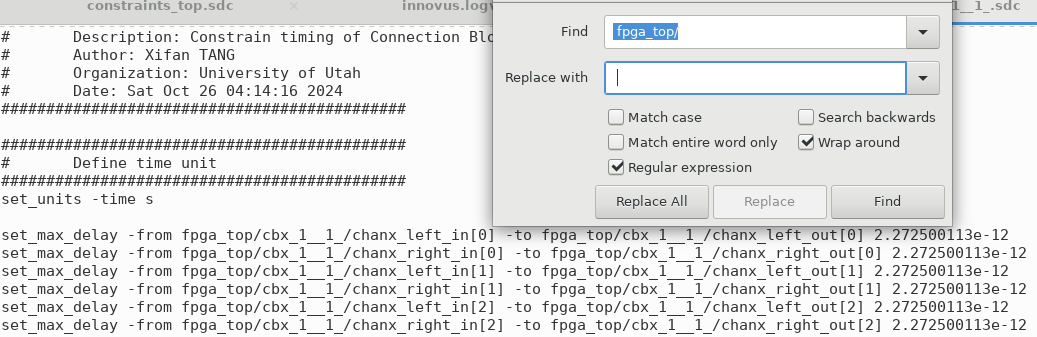
Lets remove or comment this line in all verilog netlist files where it is not necessary (at the end of module decription).

Now innovus the files provided, but with a lots of errors.

The details of these errors can be found in the innovus.log file in the working directory. Lets resolve one by one of these problems.

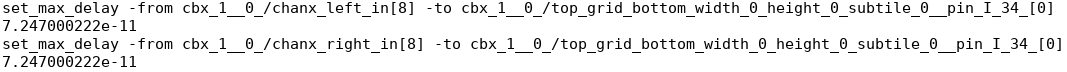
The sdc file cbx\_1\_\_1\_.sdc looks like--

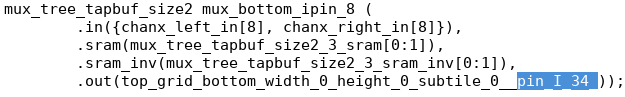
Remove fpga\_top/ --

after editing (same for other similar files)--

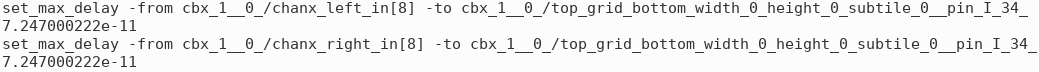
Some ports are mentioned with [0] at the end even though in the verilog files it defined differently.

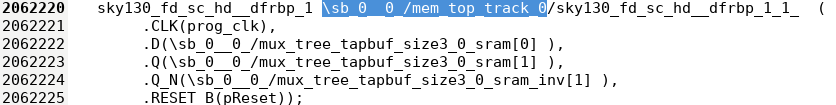
In the sdc file--

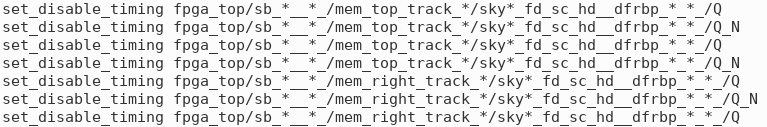
In the verilog file--



These “[0]” need to be removed to avoid errors. After editing--

Some instantiation has long names that should be fixed in sdc file--

This instance referenced in the sdc generated from OpenFPGA as below--

After correction--

As the sdc generated for hierarchical netlist and innovus is flattening the netlist the the ports/pins name become different. As a result innovus can’t identify the path description often and it difficult to figure it out the proper fix. To make the fix easier lets save the flatten netlist from innovus consol.

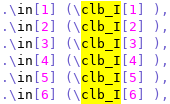
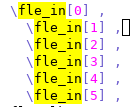
to get the netlists run the following command in the innovus consol.

==>

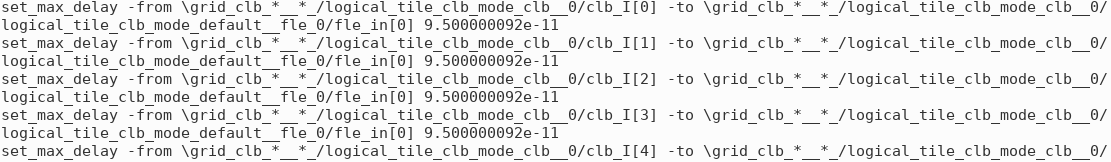
saveNetlist -flat fabric\_flat\_innovus.v

saveNetlist -flattenBus fabric\_innovus.v

==>

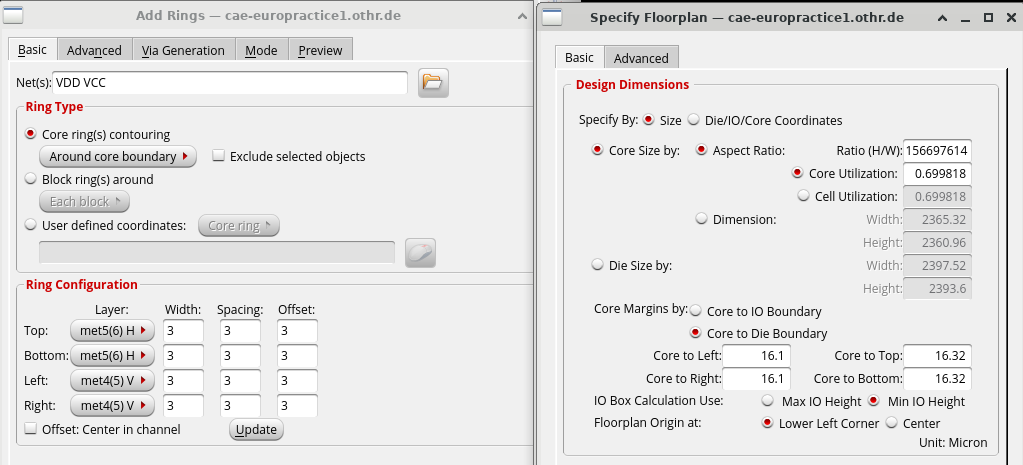
From the flattenBus netlist the path look like-- 

After correction--

**Floorplan**

Before planning IO pad, to get a rough overview I want make discreate design.

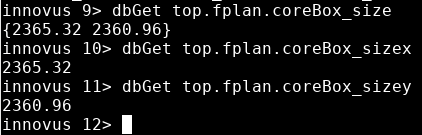
The Specify Floorplan and Add Rings are as below--

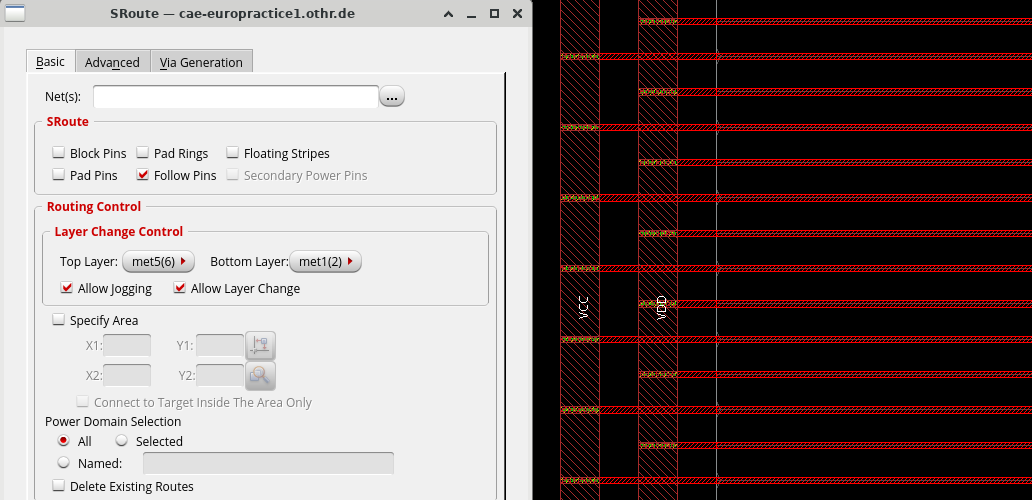
To get the size of the core area run the following--

==>

dbGet top.fPlan.coreBox\_size

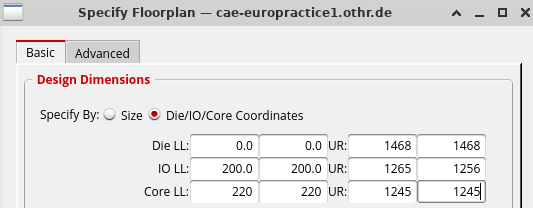
dbGet top.fPlan.coreBox\_sizex

dbGet top.fPlan.coreBox\_sizey



Corner cell size: 200 x 203.665

PAD cell size: 75 x 200



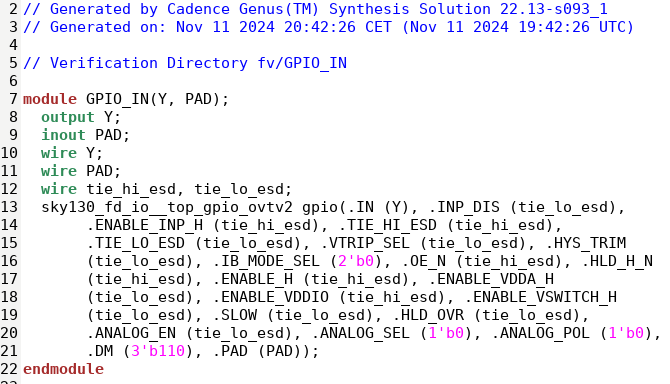
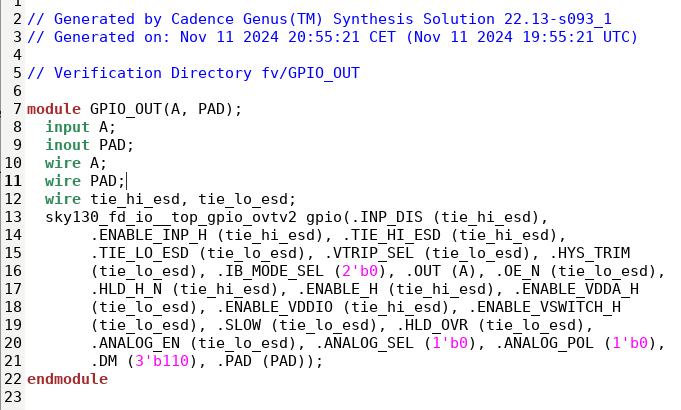
**Sky130 IO pad integration:**

Till now I have the GPIO cell (sky130\_fd\_io\_\_top\_gpio\_ovtv2). Now I need to integrate 4 corner cell and additional power pad cells.

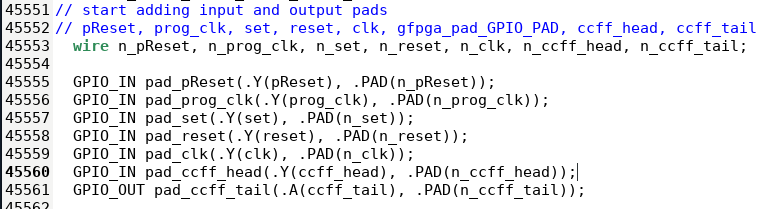
The design has 39 pins:

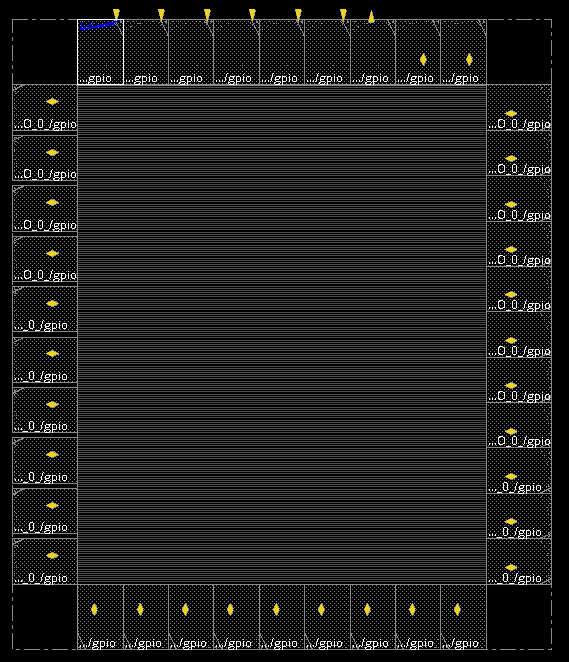
32 gfpga\_pad\_GPIO\_PAD, pReset, prog\_clk, set, reset, clk, ccff\_head and ccff\_tail.

Till now I have 32 GPIO integrated with the fabric netlist. But due to some limitations I can’t bind IO pads in the OpenFPGA architecture. That's why for the rest of the pins I have to tweak the GPIO verilog code previously for Input (disable output buffer) and Output (disable input buffer). After synthesis and checking it on testbanch--



Add this two modules in design netlist file and instantiate in the top module (fpga\_top). After necessary change made it should look like the following--

Without inserting IO configuration file in INNOVUS--



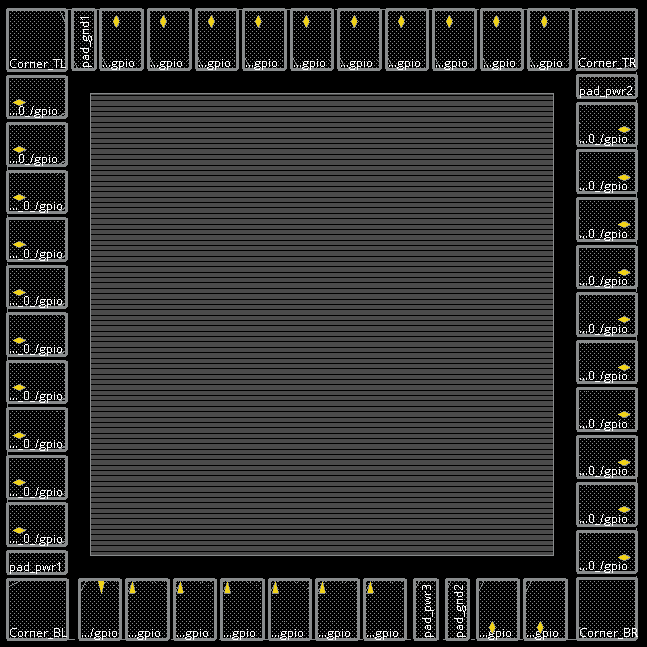
Here are 39 pins imported from the design. Also I need at least 2 additional pads for VDD and VSS. In addition there will be 4 corner cells.

|  |  |  |
| --- | --- | --- |
| Corner cell (4) | sky130\_fd\_io\_\_corner\_bus\_overlay | SIZE 200.000 BY 203.665 |
| Power cell (1?) | sky130\_fd\_io\_\_top\_power\_hvc\_wpadv2 | SIZE 75.000 BY 200.000 |
| Ground cell (1) | sky130\_fd\_io\_\_top\_ground\_hvc\_wpad | SIZE 75.000 BY 200.000 |
| GPIO cell (39) | sky130\_fd\_io\_\_top\_gpio\_ovtv2 | SIZE 140.000 BY 200.000 |

To balance the gaps and also spread the power evenly I am using 2 ground cell and 3 power cell. In total 44 pads.

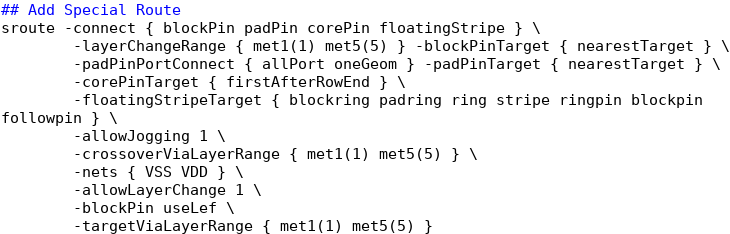
The length of the chip (left/right) = 10\*140+11\*20+75+20+200+203.665 = 2118.665

After loading the IO assignment file (<https://github.com/baruaeee/FPGA-OpenFPGA/blob/main/PNR/auto_3x3_sky_scl/IO_PAD.io>) it should look like--

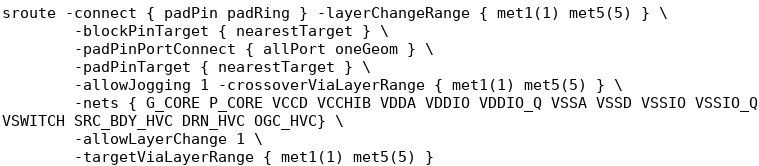
The modification of IO file is based on (page 202) <https://github.com/baruaeee/FPGA-OpenFPGA/blob/main/Innovus_user_guide_26203625.pdf>

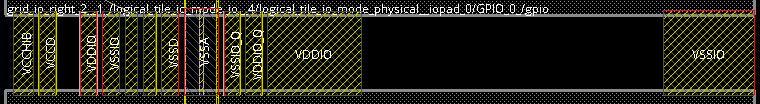
The above sample is just to get the workaround of sky130 IO pads. The online description of IO pads about sky130 is not clear to me. And also probably have some inconsistency. The corner pads define all PG pins as signal. I have to change the lef (<https://github.com/baruaeee/FPGA-OpenFPGA/blob/main/PNR/auto_3x3_sky_scl/lef/sky130_fd_io_Corner_overlay_edited.lef>) file to get around it.

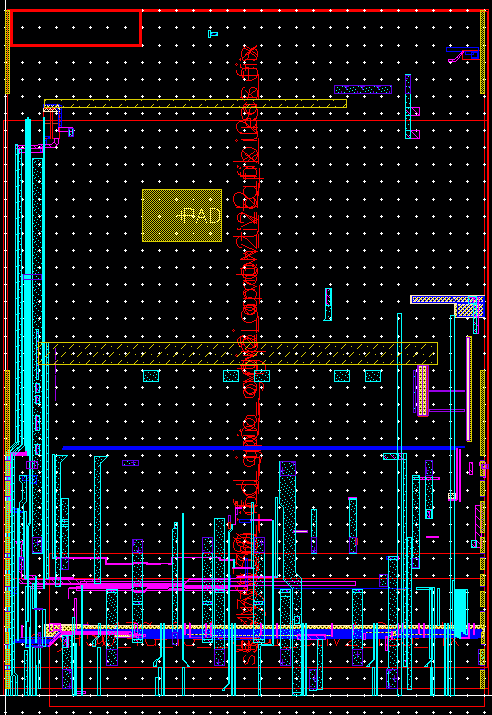
In the initial setting tcl file (<https://github.com/baruaeee/FPGA-OpenFPGA/blob/main/PNR/auto_3x3_sky_scl/initiate.tcl>) the srout done in 2 stage. 1st one will add rout inside the core.



The 2nd one will add rout in the pad ring.

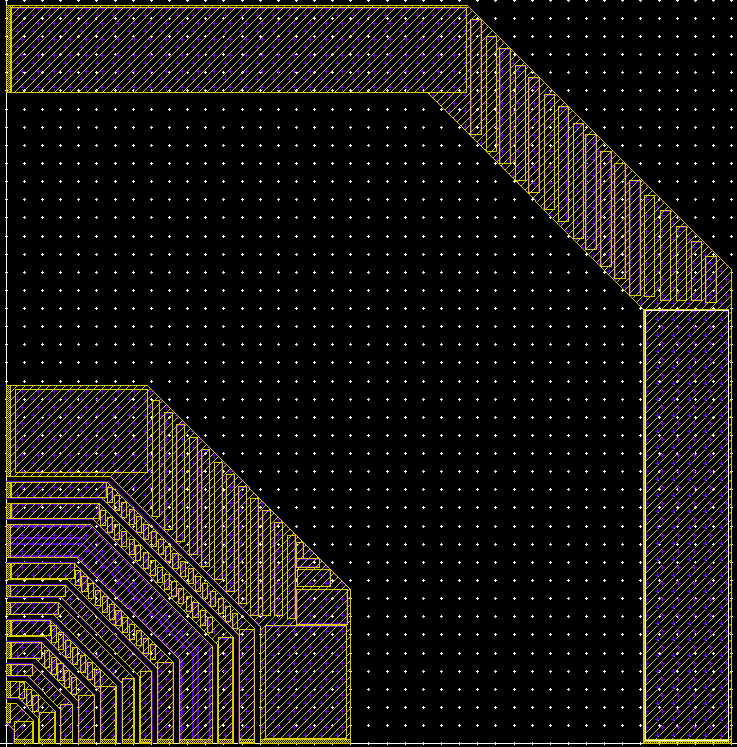


The routes are confusing. to double check lets open the gds file of sky130\_fd\_io\_\_top\_gpio\_ovtv2 in virtuoso. How to open the gds file can be found at <https://community.cadence.com/cadence_technology_forums/f/custom-ic-design/25751/import-a-gds-file-into-virtuoso>

There are 13 pins and the sequence is VSSIO, VDDIO, VDDIO\_Q, VSSIO\_Q, VSSA, VSSD, VSSA, VSWITCH, VSSIO, VDDIO, VDDA, VCCD, VCCHIB.

But the corner cell overlay has different pins.

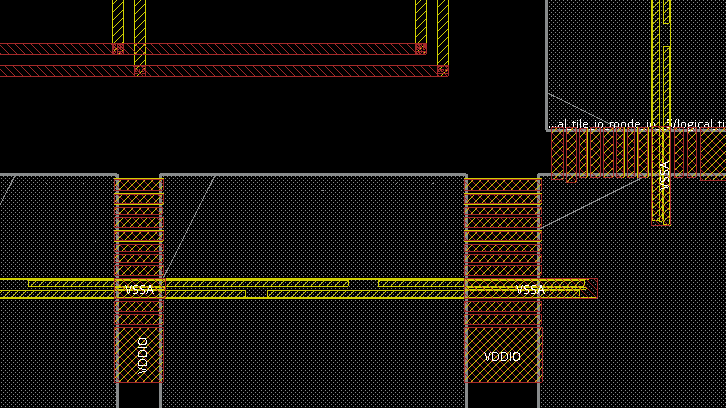


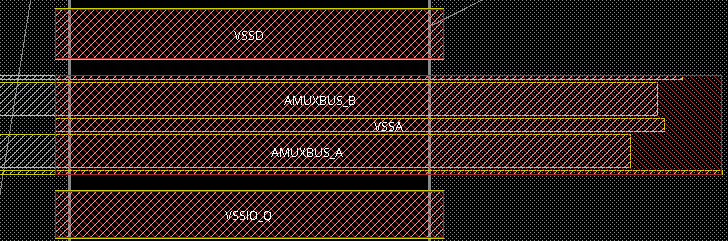
The sequence is (from right side)-- VSSIO, VDDIO, VDDIO\_Q, VSSIO\_Q, VSSA (analog shielding), AMUXBUS\_A, VSSA (analog shielding), AMUXBUS\_B, VSSA (analog shielding), VSWITCH, VSSIO, VDDIO, VDDA, VCCD, VCCHIB.

As the AMUXBUS pins are below met5 and hardly visible. These pins are for external reference voltages.

The AMAUXBUS pins (A, B) are defined as signal pins in the lef file that's why they aren't routed properly. For the sack of this issue, I am changing the lef file to define them as power pins.

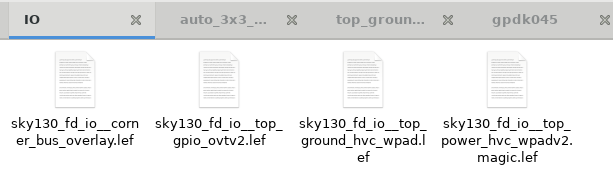
Now I have the proper sroute--





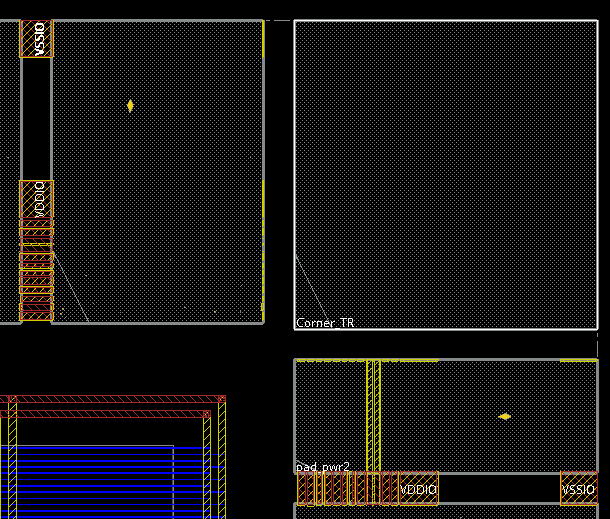
Even after this the pad power/ground pins remain open and not connected to the power/ground ring.

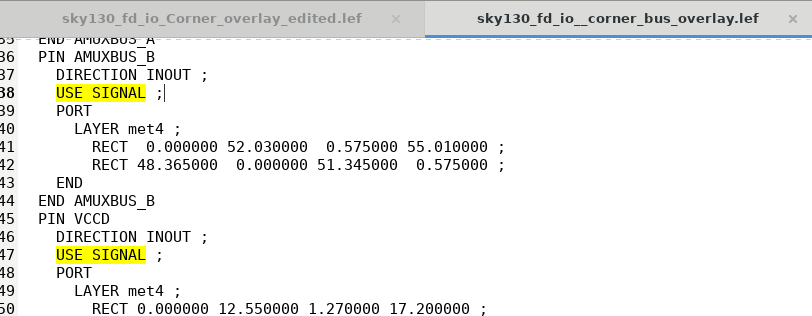
If I import the design with the combined lef file of sky130 IO library, it throw a lots of errors. To avoid that I am going to use the separate lef file for each pad I use.

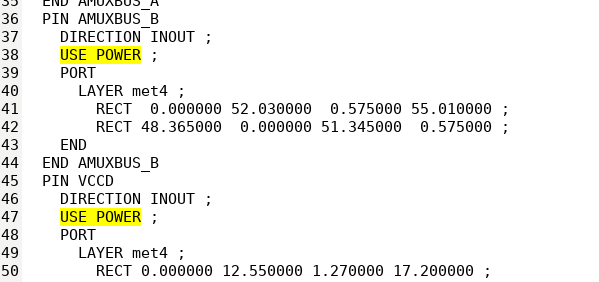
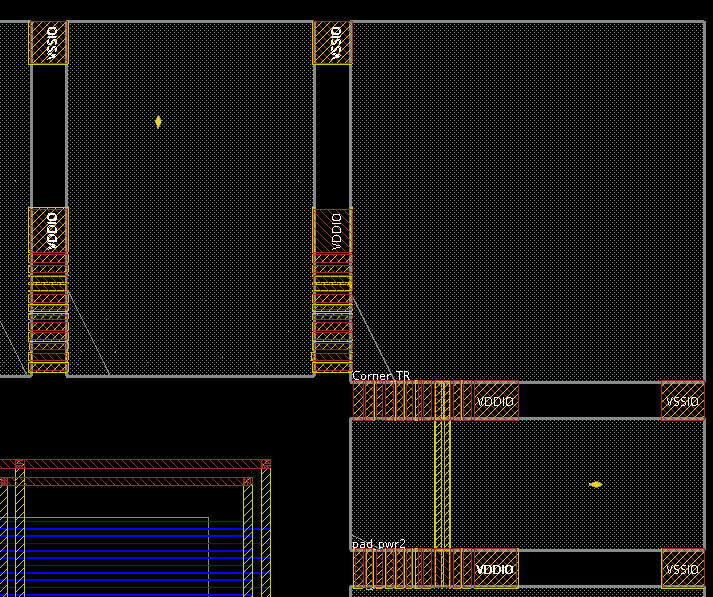


There are several corrections that need to be made to the lef files to properly work with Innovus. Lets see first what the issues are when I use them as usual.

First issue already described in the previous section about the corner cell. Due to the definition of pg pins/ports in corner cell the pad ring is incomplete.

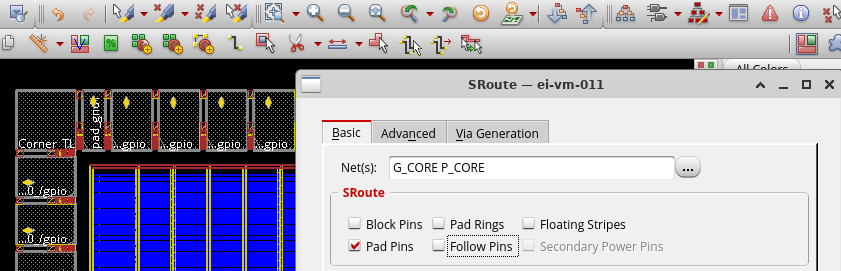


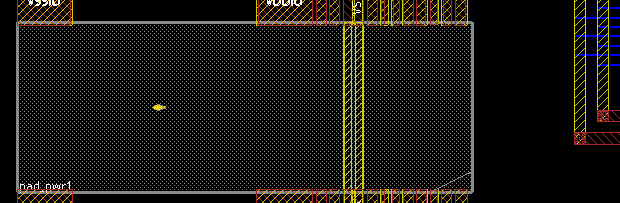
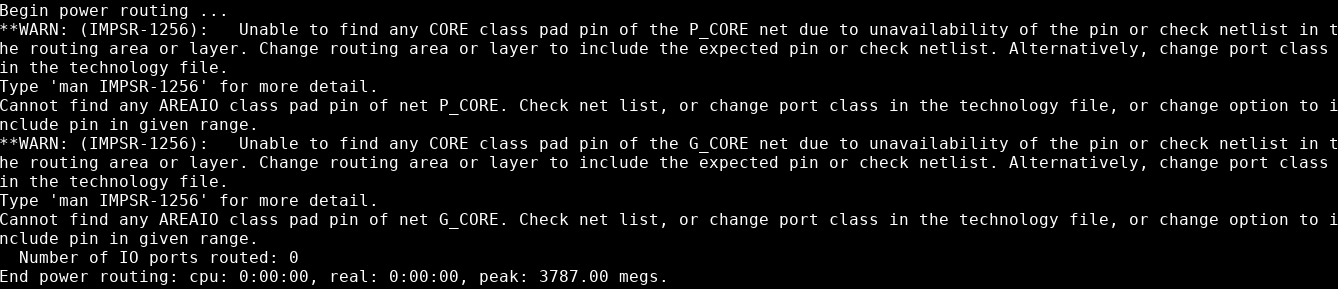
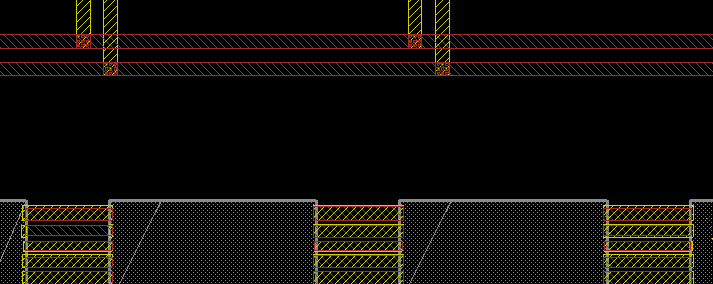
After correction--



Similarly correct other lef files where AMUXBUS defined.

Now I have to run sroute separately to establish connection between PG Pads (e.g. Power/Ground Pads).

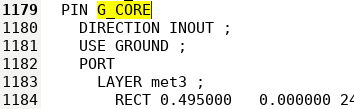
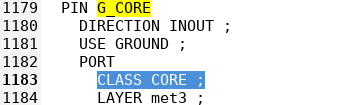
After running it throws some errors and no changes between PG pads and the power ring.

The error indicates in the interactive shell addresses the P\_CORE, G\_CORE pin need to be defined as core class in the lef file.

Now I have to edit sky130\_fd\_io\_\_top\_ground\_hvc\_wpad.lef and sky130\_fd\_io\_\_top\_power\_hvc\_wpadv2.lef to solve this problem.

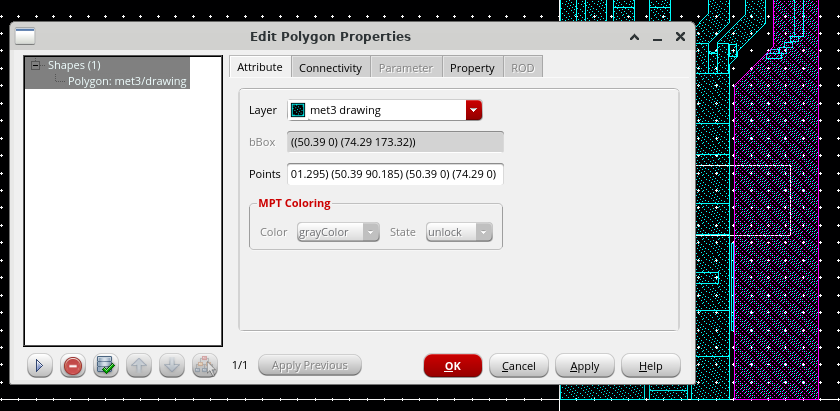
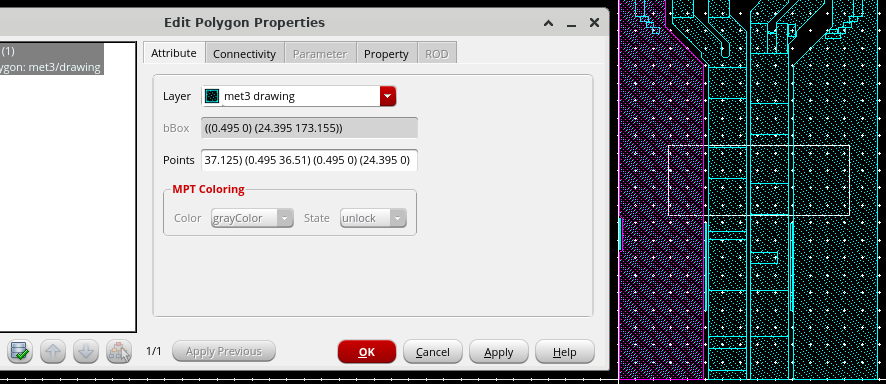
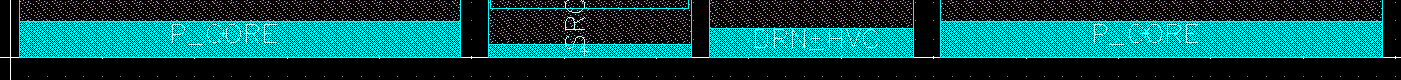
For the ground pad --

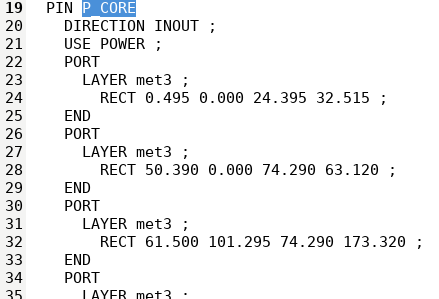
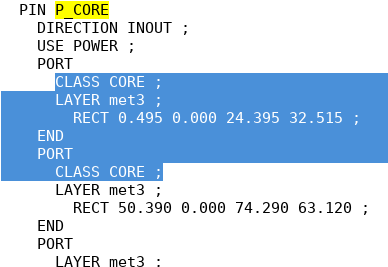
 

There are two ports under G\_CORE. These can also be confirmed by opening the gds file available in the library.

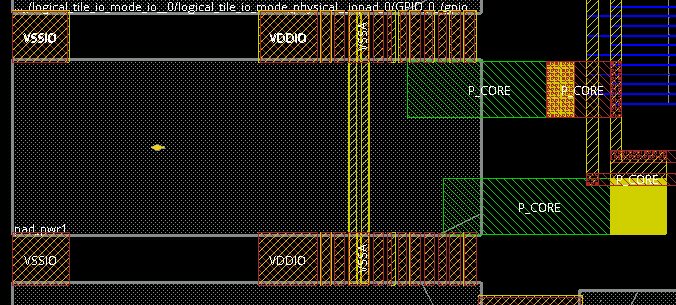
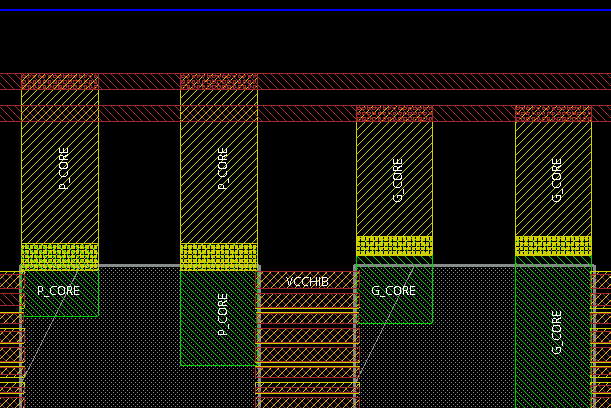
For the power pad --

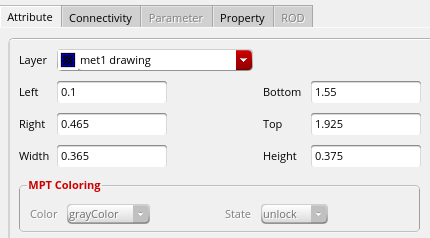
As there are many ports defined in the power pad unger G\_CORE, I have to be confirmed which port needs to be under core class. Lets see what’s in the GDS files.

Acording to the above layout, I only need to put 1st and 2nd port under core class.

After editing/correcting all --



**Modular Layout (Custom Macro)**

├── lb

│ ├── grid\_clb.v

│ ├── grid\_io\_bottom.v

│ ├── grid\_io\_left.v

│ ├── grid\_io\_right.v

│ ├── grid\_io\_top.v

│ ├── logical\_tile\_clb\_mode\_clb\_.v

│ ├── logical\_tile\_clb\_mode\_default\_\_fle\_mode\_physical\_\_fabric\_mode\_default\_\_ff.v

│ ├── logical\_tile\_clb\_mode\_default\_\_fle\_mode\_physical\_\_fabric\_mode\_default\_\_frac\_logic\_mode\_default\_\_frac\_lut6.v

│ ├── logical\_tile\_clb\_mode\_default\_\_fle\_mode\_physical\_\_fabric\_mode\_default\_\_frac\_logic.v

│ ├── logical\_tile\_clb\_mode\_default\_\_fle\_mode\_physical\_\_fabric.v

│ ├── logical\_tile\_clb\_mode\_default\_\_fle.v

│ ├── logical\_tile\_io\_mode\_io\_.v

│ └── logical\_tile\_io\_mode\_physical\_\_iopad.v

├── routing

│ ├── cbx\_1\_\_0\_.v

│ ├── cbx\_1\_\_1\_.v

│ ├── cby\_0\_\_1\_.v

│ ├── cby\_1\_\_1\_.v

│ ├── sb\_0\_\_0\_.v

│ ├── sb\_0\_\_1\_.v

│ ├── sb\_1\_\_0\_.v

│ └── sb\_1\_\_1\_.v

└── sub\_module

├── arch\_encoder.v (empty)

├── inv\_buf\_passgate.v

├── local\_encoder.v (empty)

├── luts.v

├── memories.v

├── muxes.v

├── mux\_primitives.v (empty)

├── shift\_register\_banks.v (empty)

├── user\_defined\_templates.v

└── wires.v (empty)

|  |  |  |
| --- | --- | --- |
| **Unique Modules** | **Sub-modules** | **SDC** |
| logical\_tile\_clb\_mode\_default\_\_fle\_mode\_physical\_\_fabric\_mode\_default\_\_frac\_logic\_mode\_default\_\_frac\_lut6 | frac\_lut6,  frac\_lut6\_DFFRX1\_mem | no |
| logical\_tile\_clb\_mode\_default\_\_fle\_mode\_physical\_\_fabric\_mode\_default\_\_ff | DFFSRX1 | no |
| logical\_tile\_clb\_mode\_default\_\_fle\_mode\_physical\_\_fabric\_mode\_default\_\_frac\_logic | logical\_tile\_clb\_mode\_default\_\_fle\_mode\_physical\_\_fabric\_mode\_default\_\_frac\_logic\_mode\_default\_\_frac\_lut6,  mux\_tree\_size2,  mux\_tree\_size2\_mem,  direct\_interc | yes |
| logical\_tile\_clb\_mode\_default\_\_fle\_mode\_physical\_\_fabric | logical\_tile\_clb\_mode\_default\_\_fle\_mode\_physical\_\_fabric\_mode\_default\_\_frac\_logic, logical\_tile\_clb\_mode\_default\_\_fle\_mode\_physical\_\_fabric\_mode\_default\_\_ff,  mux\_tree\_size2,  mux\_tree\_size2\_mem,  direct\_interc | yes |
| logical\_tile\_clb\_mode\_default\_\_fle | logical\_tile\_clb\_mode\_default\_\_fle\_mode\_physical\_\_fabric | yes |
| logical\_tile\_clb\_mode\_clb\_ | logical\_tile\_clb\_mode\_default\_\_fle, direct\_interc, mux\_tree\_size60, mux\_tree\_size60\_mem | yes |
| grid\_clb | logical\_tile\_clb\_mode\_clb\_ | no |

The above table is to figure out, how to plane the macro in broader scale.

Based on the above table grid\_clb is the largest block in the list and lets try to modify the design accordingly.