# Implementing the MTM-RISCV chip.

## TABLE OF CONTENTS

Document history

17.05.2021 – v1.00 – RS - initial version
12.08.2021 – v1.01 – RS – corrected git repo address
12.05.2022 – v1.02 – RS – updated for 2022

# 1 INTRODUCTION

Note: ***All your actions regarding "TODO" items in the instruction and scripts should be documented in the project report.***

## 1.1 APPLICATION NOTES

The application notes are the most important documents to read before starting the implementation.

The application notes can be found in the folder:

*/cad/dk/PDK_CRN45GS_DGO_11_25/digital/Documentation/application_notes*

TODO: Browse the application notes to see, what they are about. Do not dig into the details at the moment. The important references will be given in the following sections.

## 1.2 FLOORPLAN

The floorplan is a map of the layout regions where physical cells (standard cells, I/O cells, bond pads, blocks) are placed. The floorplan consists of the following elements, shown also on the picture below.

### 1.2.1 Core

The core is the region where standard cells (digital gates) are placed. The core is built of rows. Each row designates the area for a set of standard cells connected by abutment. The height of the row is equal to the height of the standard cell.

Standard cell documentation is in the folder:

*/cad/dk/PDK_CRN45GS_DGO_11_25/digital/Documentation/documents/tcbn40lpbwp_200a*

Note that for each timing corner a separate file is generated.

### 1.2.2 I/O rows

Each I/O row is a region, where the I/O cells are placed. The I/O cells include drivers and receivers of the signals from the out-of-the-chip world. They are powered with a dedicated power supply, called post-driver-voltage, (VDDPST). Another type of I/O cell provides power connections.

The documentation of the I/O cells is in the folder:

*/cad/dk/PDK_CRN45GS_DGO_11_25/digital/Documentation/documents/tpfn40lpgv2od3_120a*

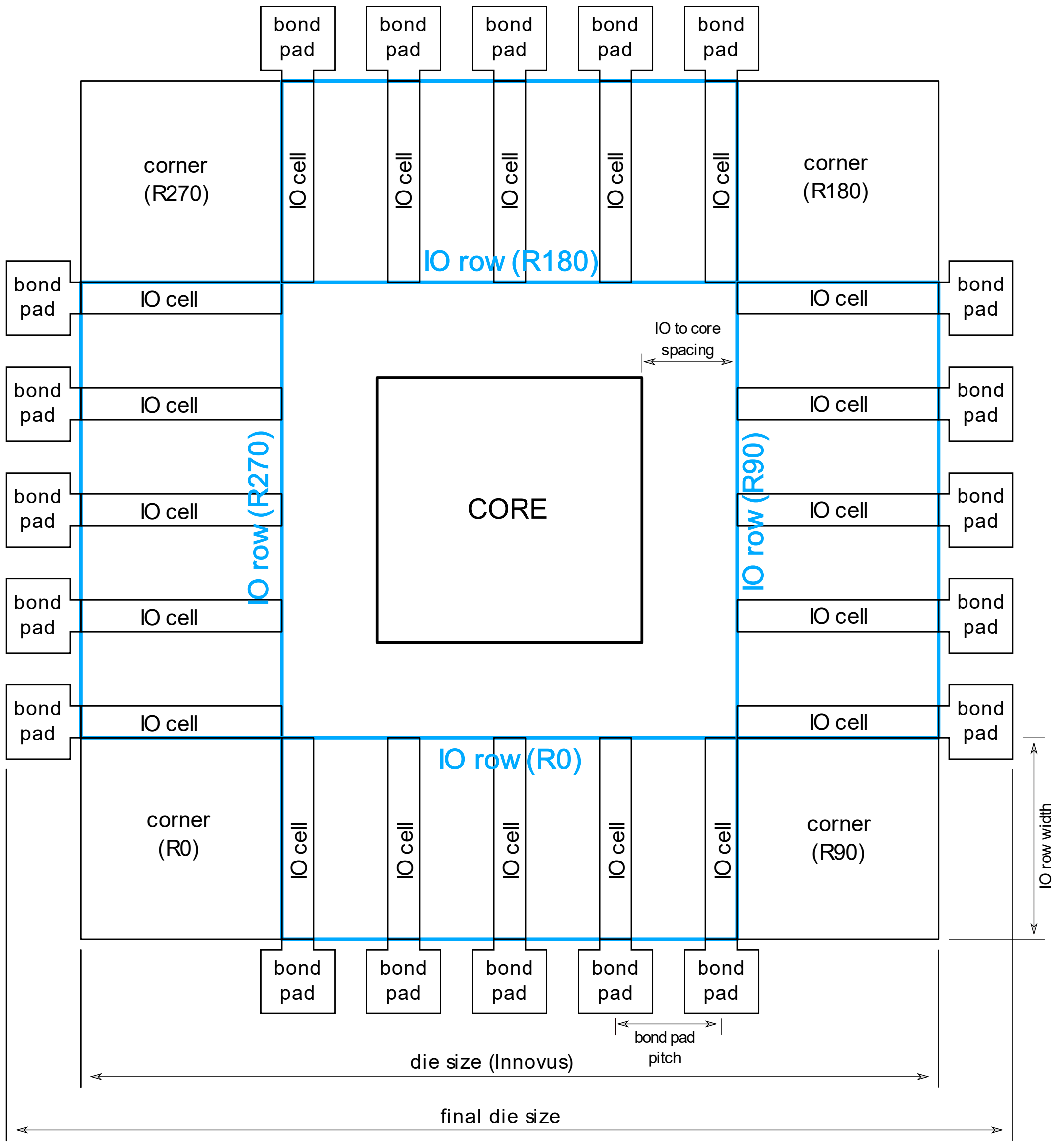
Note that there are many possible configurations of the VDDPST and VDD (core) voltages. In our case the VDDPST = 2.5V, and the relevant documents are *1.lib, (*tc1.lib, *lt1.lib, etc.).

### 1.2.3 Chip corners

The corners are physical cells placed at the corners of the chip to connect the I/O rows. Usually, a single cell is used with a different rotation.

### 1.2.4 Bond pads

Bond pads are metal structures with no logic function. They will be added to the design at the last stage.



## 2 Requirements for the Floorplan

#1. The floorplan should be square.
#2. The number of bond pads at each side should be the same (with a single exception of Power-On-Control cell).
#3. The minimum bond pad pitch is **100 µm** (90 µm after scaling) – this is the bonding requirement.

#4. The output digital I/O cells should be able to drive **30 pF** load with **50 MHz** frequency at **worst-case timing**. The maximum rise/fall time should not exceed **5ns**.

#5. The chip will have a single power domain - core **VDD = 1.1V** for all the standard cells and blocks, and I/O power **VDDPST = 2.5V** for all I/O cells. Common **VSS** will be used for both core and I/O cells.

#6. The number of power/ground pads should be selected accordingly to the IO cell requirements. Assume **5.2 nH** bond wire inductance. Assume that all the outputs can switch simultaneously.

#7. The metal stack is **8M_5x2z**.

# 3 TOP-LEVEL NETLIST

TODO: synchronize to GIT repo, ***master*** branch from the repo:

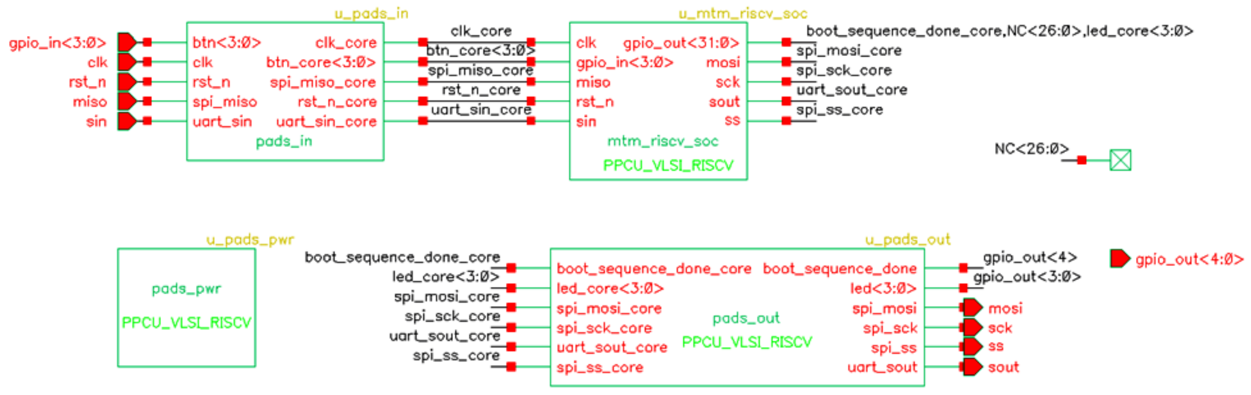
```
git clone –recursive git@github.com:agh-riscv/mtm_ppcu_vlsi_riscv.git
```

Comparing to FPGA, the ASIC implementation needs additional components – the I/O cells. They provide buffering the input signals and driving the chip output with the proper drive strength (loads in the range of pF) with the required voltage, which is usually much higher than the core power supply (e.g. 2.5V vs 1.1V). Therefore one more hierarchy level is added in the design.

The top-level module is *asic/rtl/mtm_riscv_chip.v*

## 3.1 TOP-LEVEL SCHEMATIC

The top-level chip consists of the core (module: *mtm_riscv_soc*), and three submodules containing input (module: *pads_in*), output (module: *pads_out*) and power (module *pads_pwr*) I/O cells.



Note that the core has 32-bit GPIO output, while only 5 signals are used at the chip top.

## 3.2 I/O CELLS SELECTION

Note that in the documentation the name “I/O cell” and “I/O pad” are used interchangeably.

### 3.2.1 Input I/O cells

TODO: Find in the documentation a simple input buffer (with no control input, pull-up, pull-down, etc.), with hysteresis.

TODO: Replace the input pad cell names in the netlist file: *asic/rtl/pads_in.v*

### 3.2.2 Output I/O cells

For the output, also simple I/O cells will be used. The names used in the library are PDO*, they have different drive strengths.

TODO: Select the proper output cells to fulfill requirement #4. Use the data in the **worst case** (WC1) timing library files (*.lib) to verify required rise/fall time. Provide calculations, if necessary.

TODO: Replace the output pad cell names in the netlist file: *asic/rtl/pads_out.v*

## 3.3 Power pads

### 3.3.1 Power pads for the core

Use two power pads PVDD1DGZ and two ground pads PVSS3DGZ for the core.

### 3.3.2 Power pads for I/O cells (post-driver-voltage).

Use the proper number of PVDD2DGZ pads and PVDSS3DGZ pads for the i/o cells.

TODO: Calculate the required number I/O power/ground cells to fulfill requirement #6.

The method to calculate the correct number of pads is described in the *TSMC Universal Standard I/O Library General Application Note*, page 74. Note that for calculation of the number of power pads you are supposed to use the data for the **best case** (the fastest) library. Include the calculations in the report.

TODO: Add the correct number of the power pads in the file: *asic/rtl/pads_pwr.v*

### 3.3.3 Power-on control

One (and only one) power-on control cell is required. One PVDD2POC cell is placed in the netlist for this purpose. See chapter 3 “Power On Control System” in the application note for more information.

### 3.3.4 Other requirements.

To keep the floorplan regular (square), it is recommended to have the same number of pads on each side of the chip. Therefore the number of pads should be multiple of 4.

TODO: If necessary, add more power pads, so that the total number of pads is a multiple of 4.

## 3.4 Netlist verification

The netlist will be verified by simulation. The simulations are started from the *sim* directory. The main script to run the simulations is *run.sh.* The default simulation generates the binary pattern on the *led[3:0]* outputs. The pattern is compared to the one stored in the *led_wave_correct.txt* file.

The simulation can be run in FPGA and ASIC modes. The FPGA mode does the simulation without any pads, and with the RAM models appropriate for FPGA implementation. The ASIC mode includes the pads and uses TSMC RAM models.

Run the script without any arguments to see the syntax.

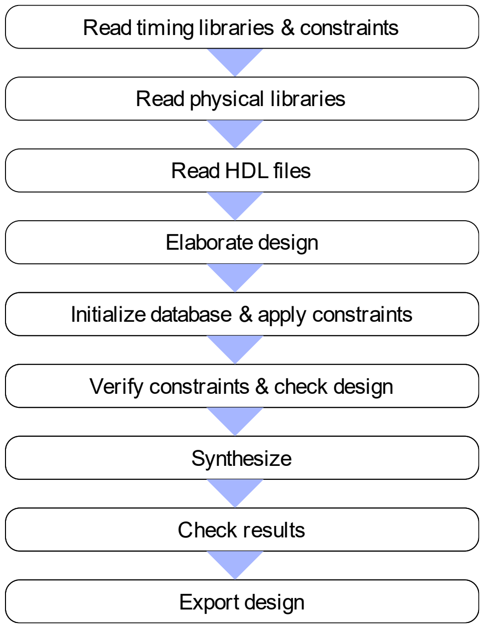
TODO: Run the simulation in FPGA and ASIC mode to verify the correctness of the netlist.

# 4 LOGIC SYNTHESIS

The logic synthesis is run in the *asic/synth* directory.

To open the help application, run the command: *./run_help_genus &*

## 4.1 THE FLOW


Read timing libraries & constraints
Read physical libraries
Read HDL files
Elaborate design
Initialize database & apply constraints
Verify constraints & check design
Synthesize
Check results
Export design

## 4.2 NETLIST PREPARATION

To generate the netlist for the synthesis, change to *asic/synth/rtl* directory and run *./get_netlist.sh* command. The command puts all the files used for simulation into a single *netlist.sv* file and verifies the contents running *xrun* elaboration. Alternatively, you can use *make* command.

## 4.3 SYNOPSYS® DESIGN CONSTRAINTS

The timing requirements for the ASIC are defined in SDC (Synopsys Design Constraint) file. The file to be modified is *asic/synth/constraints/design.sdc*. The lines to be modified are marked with the "TODO" keyword.

TODO: Complete the SDC file according to the description inside the file. Use help to find the correct syntax.

## 4.4 RUN SYNTHESIS

The Cadence synthesis tool (Genus) is run in the *asic/synth* directory, the starting script is *run_synth.sh*. The script is configured to run in the batch mode, but it can be modified to run in the interactive mode (see the script content).

The main synthesis script is *asic/synth/scripts/synthesize.tcl.* It is not complete, the necessary actions are marked with "TODO" keyword.

TODO: Complete the synthesis script according to the instructions inside the file, and run the synthesis. Make sure there are no errors before you go to the place & route stage.

# 5 Place and Route

To open the help application, run *./run_help_innovus &* command in the *asic/pr* directory.

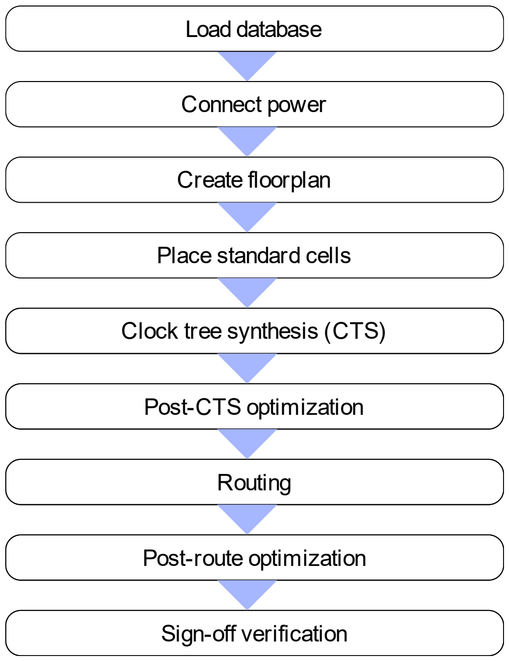
The Cadence place & route tool (Innovus) is run in the *asic/pr* directory, the starting script is *run_pr.sh*. The script is configured to run in the interactive mode, but it can be modified to run in the batch mode (see the file). You will need the interactive to mode to floorplan the chip.

Note that Innovus is started in the common interface mode with the -*stylus* command option. If this option is missing the tool is started in so called *legacy* mode.

The main Innovus script for the whole place & route is *scripts/run_pr.tcl* . It calls a list of scripts for the consecutive design stages. Before running each script check it's contents for further instructions. The necessary actions are marked with the "TODO" keyword.

## 5.1 The Flow

The figure below shows the main steps in the physical design process (place &route).



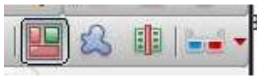
## 5.2 Design initialization

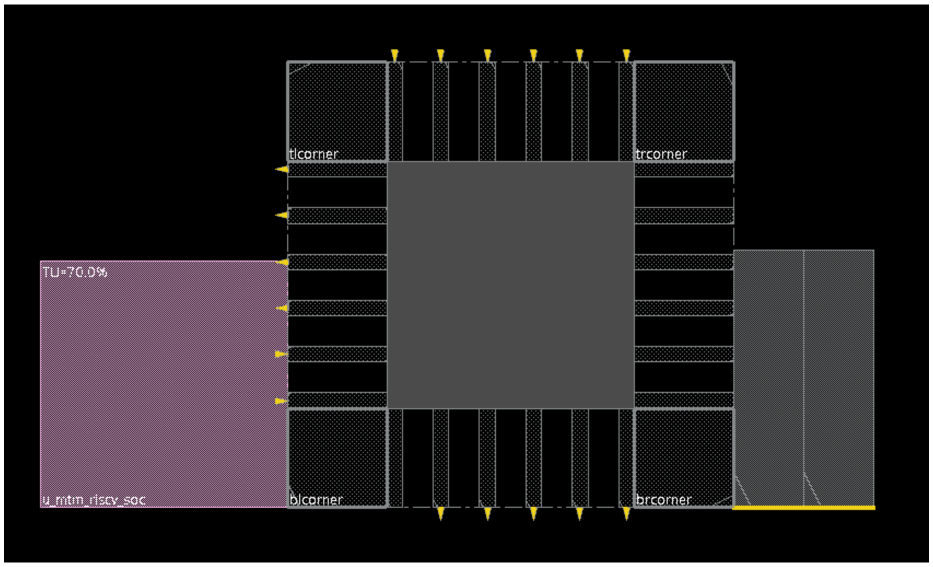
To load the design, execute the following Innovus command. TODO:

```
source scripts/01_initialize_design.tcl
```

The script does the following:

- defines the power signal names to be used in the design,
- loads the design which was save with Genus
- add physical cells for the corners used to close the I/O cell ring. The cells are defined in the *corners.io* file. Have a look into the file to see the instance names of the corner cells.

At this stage the floorplan view will show something similar to: 



## 5.3 Connecting the power

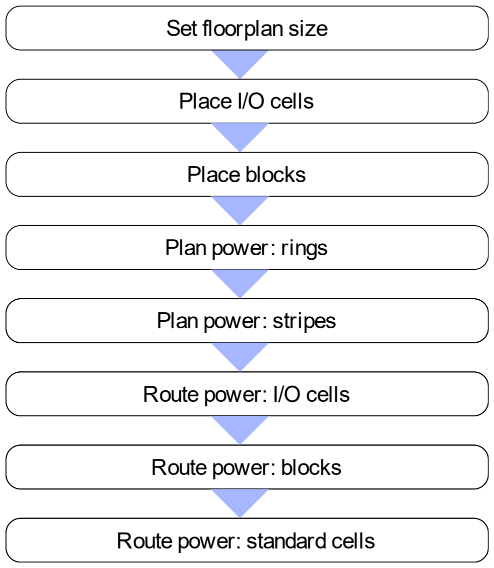
The netlist used so far does not include the power connections. The power nets have already been defined in the previous stage.

TODO: Use the file: *scripts/02_connect_power_to_gates.tcl*

## 5.4 Floorplan

Floorplanning is probably one of the most time-consuming part of the digital design. At this stage you define the design size, place I/O cells and blocks, and create the power network.

### 5.4.1 The flow



### 5.4.2 Initialize the floorplan

TODO: Calculate the size of the chip bearing in mind the requirements from section 2. Reserve 100 µm between the core and the I/O cells for the power ring. The I/O ring width is 190 µm. Include the calculations in the project report.

Use the following routing directions for all the power connections:

- horizontal: M7, M5
- vertical: M8, M6

TODO: Follow the description in the *scripts/03_create_floorplan.tcl* file create the floorplan.

### 5.4.3 IO placement

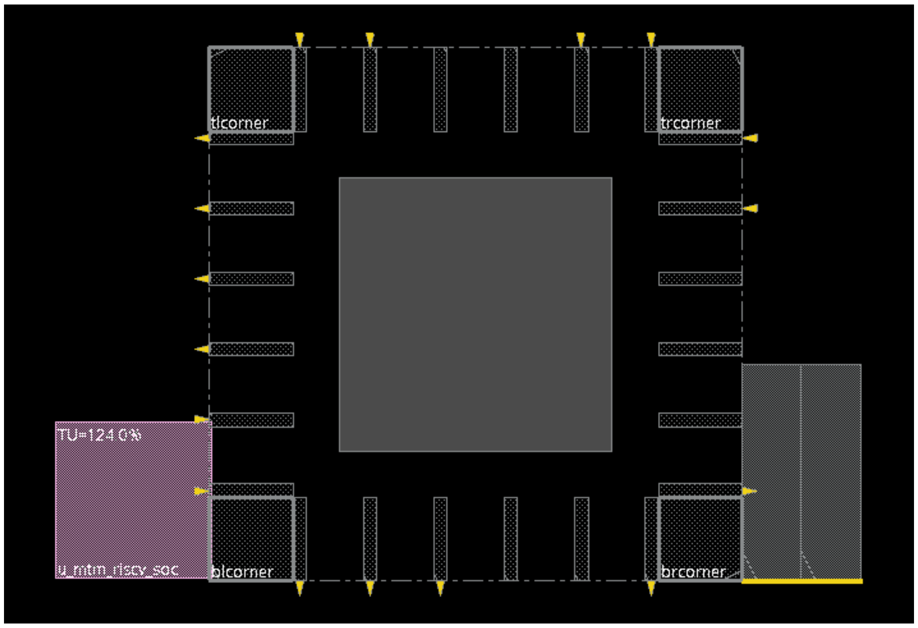
TODO: Draw a simple picture showing the locations of the I/O cells in the floorplan bearing in mind the requirements from section 2 .

Do not place the core power supplies (VSS, VDD) at the corner cells (this will result in routing problems).

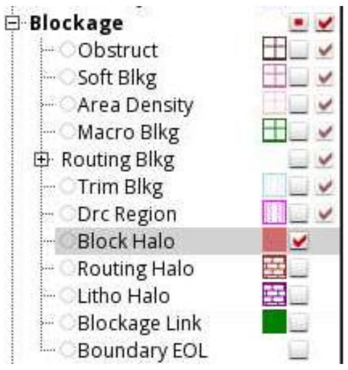
Try to interleave the power and ground cells, to keep the power distribution uniform.

TODO: follow the instructions in the *tcl* file to configure the IO.

After this stage you should have floorplan with the correct size, similar to the figure below.



### 5.4.4 Placing blocks



"Halo" is the area around the block with specific constraints, e.g. placement, or routing not allowed. We will put the power rings around the RAM cells and we do not want the standard cells below this rings. Make block halo visible, as shown in figure on the right.

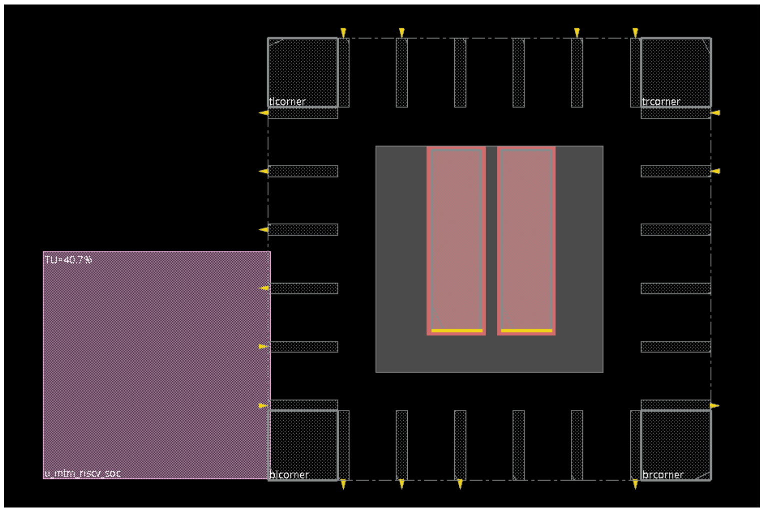
TODO: Create halos around the blocks as described in the *tcl* file.

To move the RAMs switch to the *Move/Resize/Reshape* mode:

Use mouse to place the RAM blocks in the desireg position.

Open block properties (double-click in the select mode, or press Q) and write down the block coordinates.

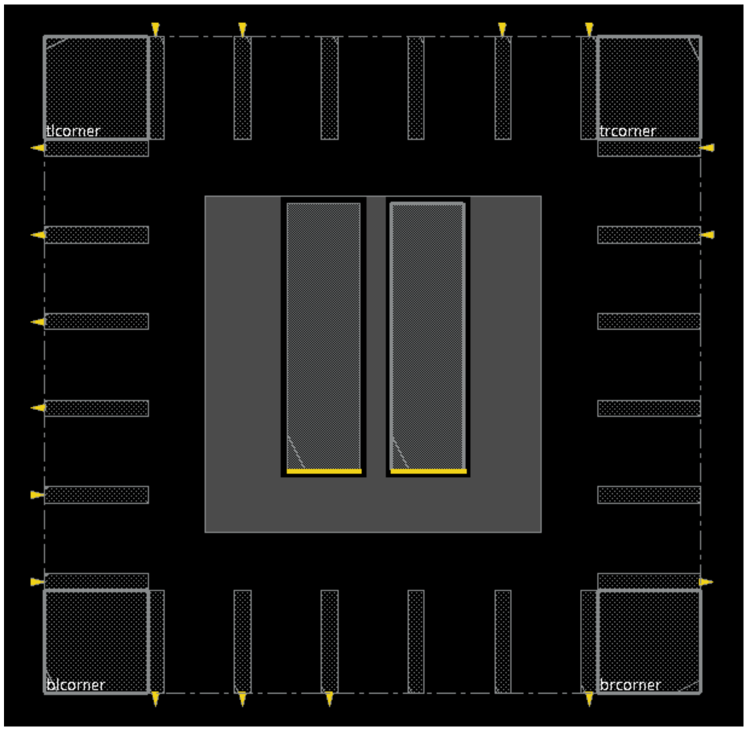
TODO: Put the coordinates in the *tcl* script.



As you  do not want the standard cells to be placed over the RAM blocks and within the halo, you will remove the core rows in this area.

TODO: Follow the instruction in the *tcl* script to cut the rows.

After disabling the display of the block halo, you will see the following picture.



### 5.4.5 Power planning

#### *5.4.5.1 Core rings*

The first power distribution is a set of rings around the core area (or between the core and I/O cells). You have reserved the 100 µm spacing here.

The ring will be done using M6, M7 and M8 layers (remember the required directions).

Use the area in the optimum way, following the DRC rules:

/cad/dk/PDK_CRN45GS_DGO_11_25/PDK/PDK_doc/DesignRules/T-N45-CL-DR-001_2_4.pdf

Have in mind: maximum wire width and maximum metal density. Use the same width for both VDD and VSS.

Make M6+M7 ring first. Than add M8 covering M6.

TODO: put in the *tcl* script the commands for core ring creation.

#### *5.4.5.2 Block rings*

TODO: Add the rings around the RAM blocks. Follow the instructions in the *tcl* file.

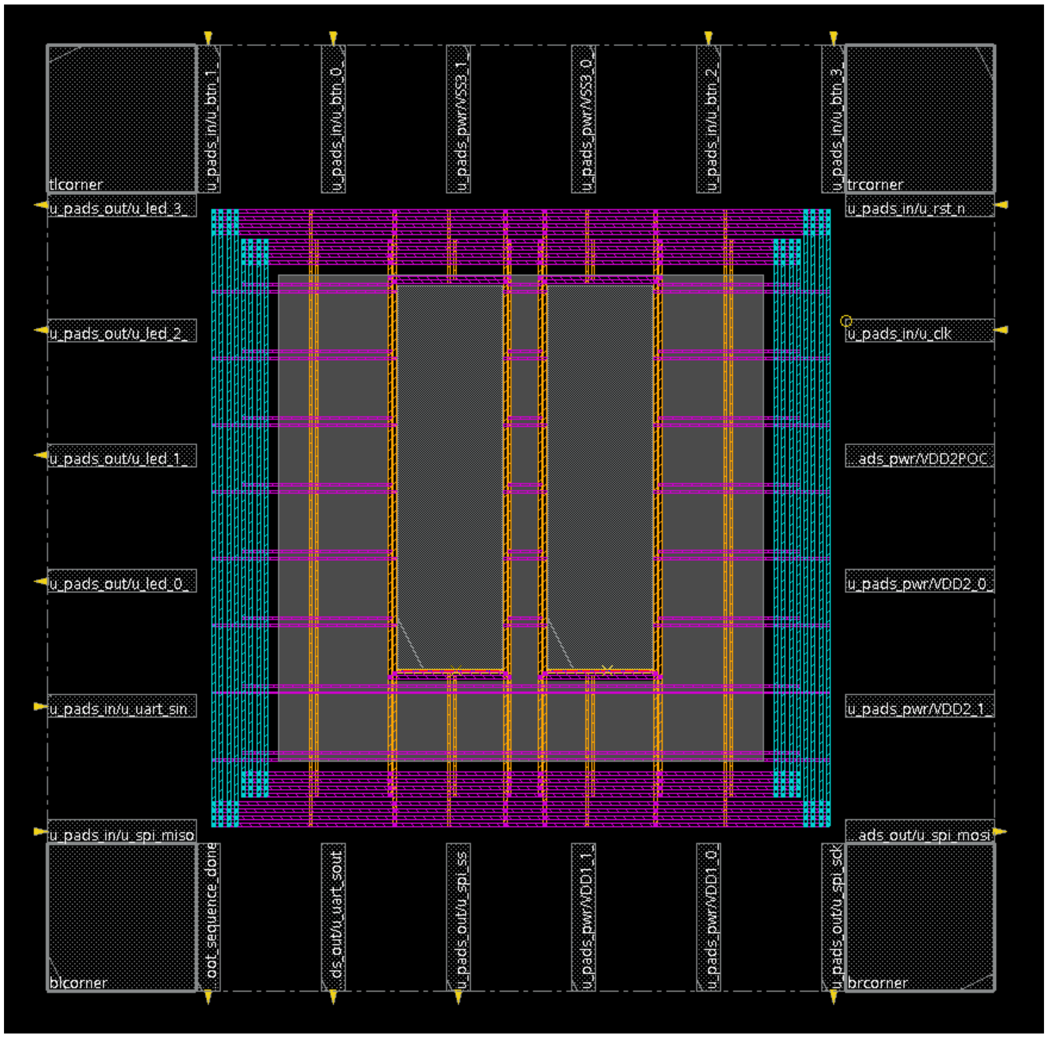
#### *5.4.5.3 Stripes*

Add vertical and horizontal power stripes.

Vertical stripes will be connected to the standard cell power rows and the core rings. They will use M6.

Horizontal stripes will be connected to vertical stripes and the core rings. They will use M7.

TODO: Follow the instruction in the *tcl* file to add the stripes.



### 5.4.6 Power routing

#### *5.4.6.1 Pads*

TODO: Follow the instructions in the *tcl* file to connect the VSS and VDD pins of the **power pads** to the **core ring**.

All the VDD/VSS pins of the pads must be connected.

#### *5.4.6.2 Blocks*

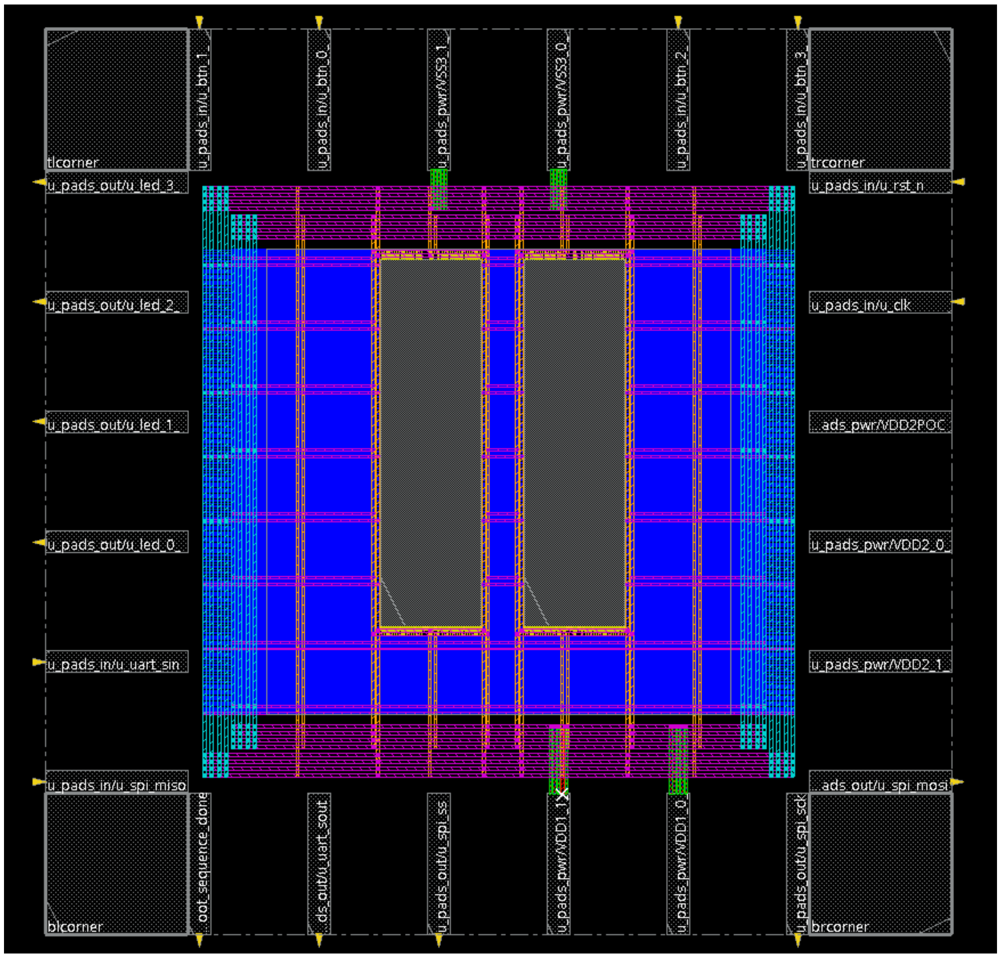
TODO: Follow the instructions in the *tcl* file to connect the VSS and VDD pins of the RAM **blocks** to the **block rings**.

All the VDD/VSS pins of the pads must be connected.

#### *5.4.6.3 Standard cell*

TODO: Follow the instructions in the *tcl* file to connect the VSS and VDD pins **standard cells** to the **stripes**, the **block rings** and the **core ring**.

The full floorplan of the chip should be similar to the figure below.



## 5.5 STANDARD CELL PLACEMENT AND OPTIMIZATION

TODO: Run the *scripts/04_*.tcl* and *scripts/05_*.tcl* scripts and write down the TNS and WNS values as defined in these files.

The first script runs the timing checks before the placement. The second one places the standard cells and runs the post-placement optimization to correct the timing.

## 5.6 Clock tree synthesis

Before clock tree synthesis is run, you need to make sure that the timing of the design is correct.

TODO: If the WNS from the previous stage is negative (less than minus few ps, eg. < -10 ps), run the pre-CTS optimization using the script *scripts/06_*.tcl* . Otherwise skip this step.

TODO: Run the clock tree synthesis using the script *scripts/07_*.tcl* . Put in the report the proper values as described in the script.

The insertion of the clock tree changes a lot of timing paths, so the consecutive optimization step is required.

TODO: Run the post-CTS optimization using the script *scripts/08_*.tcl* . Put in the report the proper values as described in the file.

## 5.7 Routing and post-route optimization

TODO: Run *scripts/09_*.tcl* to route the design.

This scripts first adds I/O and core filler cells, than runs the routing.

With existing routing it is possible to precisely estimate the timing of the design, so the optimization step is necessary.

TODO: Run *scripts/10_*.tcl* to perform the optimization. Make sure that the timing is closed and write the requested values in the report.

## 5.8 Sign-off timing analysis

Sign-off timing analysis uses precise extraction (Quantus) to get the RC values for the routing.

TODO: Runs scripts *11* and *12* to extract the RC values and analyze the timing. If the timing is not closed, run the optimization script *13*. Check the WNS/TNS/DRV values and write them into the report.

TODO: Run script 14 to check the DRC/LVS. This script will also add the bond pads first. Verify the results.

## 5.9 Writing P&R results

TODO: Run script 15 to save the results. The script will save:

- GDS file – the layout data, almost read for the production. “Almost” means: not fully verified and without filling. The missing items are post-layout simulation, power consumption analysis and filling procedure.

TODO: Provide the screenshot of your design in the report.