Assignment 3

ASIC flow with Synopsys Tools

{System}Verilog for ASIC/FPGA Design & Simulation

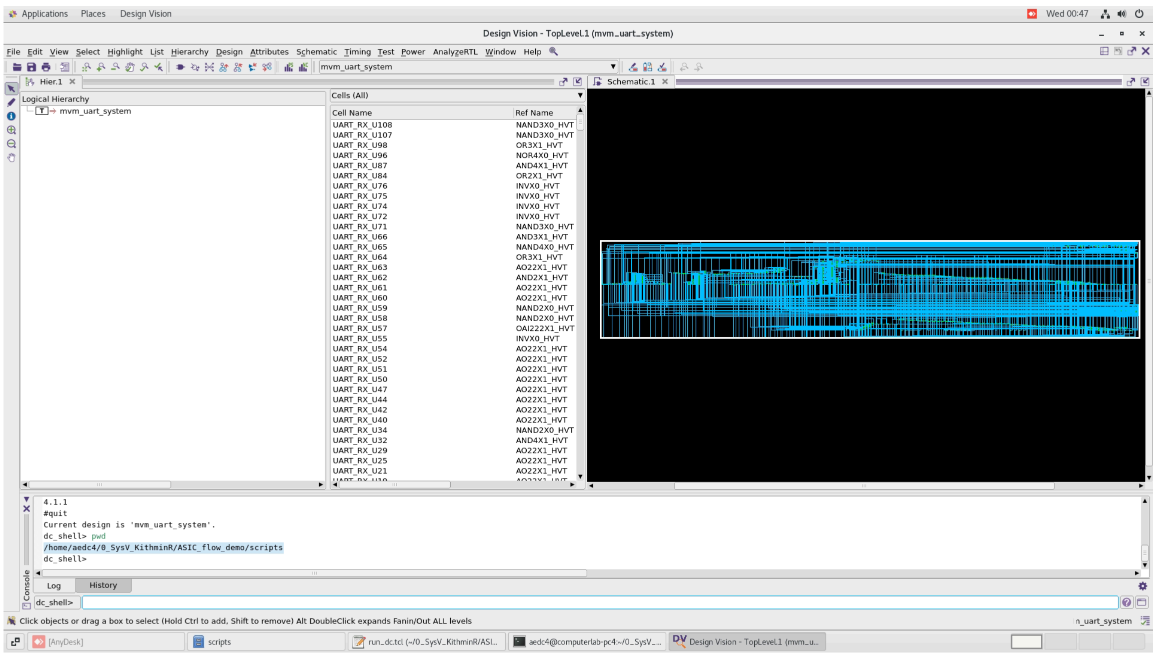
In this Assignment, you have to take the Matrix Vector Multiplier UART System final verilog code (https://github.com/SkillSurf/systemverilog/tree/master/rtl), and synthesize the RTL using the SAED 32nm EDK and Synopsys Design Compiler. Next, you must get the final layout after doing PnR using the Synopsys IC Compiler II. You are free to vary the 4 parameters R, C, W_X, W_K, but include the final selection of parameters in the submission (suggested to use {2,2,2,2}). Finally, you have to export the final .gds file and visualize it using kLayout (https://www.klayout.de/build.html). The klayout step is optional.

Please navigate into the folder that you created with your name during the demo and make a new folder called "assignment". Please make sure to run your tools from that location for the assignment.

Deliverables

For successful completion of this assignment, you should submit a pdf report with screenshots of the different sections in the ASIC flow that you followed. The below mentioned parts must be included in the report.

1. The synthesized RTL design schematic visualized in Design Vision of DC (example shown below) - use the 'pwd' command on the terminal to show the directory you are inside, which should have **the folder with your name**.


Applications
Places
Design Vision
Wed 00:47
Design Vision - TopLevel.1 (mvm_uart_system)
File Edit View Select Highlight List Hierarchy Design Attributes Schematic Timing Test Power AnalyzeRTL Window Help
mvm_uart_system
Hier.1
Schematic.1
Logical Hierarchy
mvm_uart_system
Cells (All)
Log
History
Console
Click objects or drag a box to select (Hold Ctrl to add, Shift to remove) Alt DoubleClick expands Fanin/Out ALL levels
[AnyDesk]
scripts
run_dc.tcl (~/0_SysV_KithminR/ASI...
aedc4@computerlab-pc4:~/0_SysV_...
Design Vision - TopLevel.1 (mvm_u...

Cell Name	Ref Name
UART_RX_U108	NAND3X0_HVT
UART_RX_U107	NAND3X0_HVT
UART_RX_U98	OR3X1_HVT
UART_RX_U96	NOR4X0_HVT
UART_RX_U87	AND4X1_HVT
UART_RX_U84	OR2X1_HVT
UART_RX_U76	INVX0_HVT
UART_RX_U75	INVX0_HVT
UART_RX_U74	INVX0_HVT
UART_RX_U72	INVX0_HVT
UART_RX_U71	NAND3X0_HVT
UART_RX_U66	AND3X1_HVT
UART_RX_U65	NAND4X0_HVT
UART_RX_U64	OR3X1_HVT
UART_RX_U63	AO22X1_HVT
UART_RX_U62	AND2X1_HVT
UART_RX_U61	AO22X1_HVT
UART_RX_U60	AO22X1_HVT
UART_RX_U59	NAND2X0_HVT
UART_RX_U58	NAND2X0_HVT
UART_RX_U57	OAI222X1_HVT
UART_RX_U55	INVX0_HVT
UART_RX_U54	AO22X1_HVT
UART_RX_U52	AO22X1_HVT
UART_RX_U51	AO22X1_HVT
UART_RX_U50	AO22X1_HVT
UART_RX_U47	AO22X1_HVT
UART_RX_U44	AO22X1_HVT
UART_RX_U42	AO22X1_HVT
UART_RX_U40	AO22X1_HVT
UART_RX_U34	NAND2X0_HVT
UART_RX_U32	AND4X1_HVT
UART_RX_U29	AO22X1_HVT
UART_RX_U25	AO22X1_HVT
UART_RX_U21	AO22X1_HVT

```
4.1.1
#quit
Current design is 'mvm_uart_system'.
dc_shell> pwd
/home/aedc4/0_SysV_KithminR/ASIC_flow_demo/scripts
dc_shell>
```

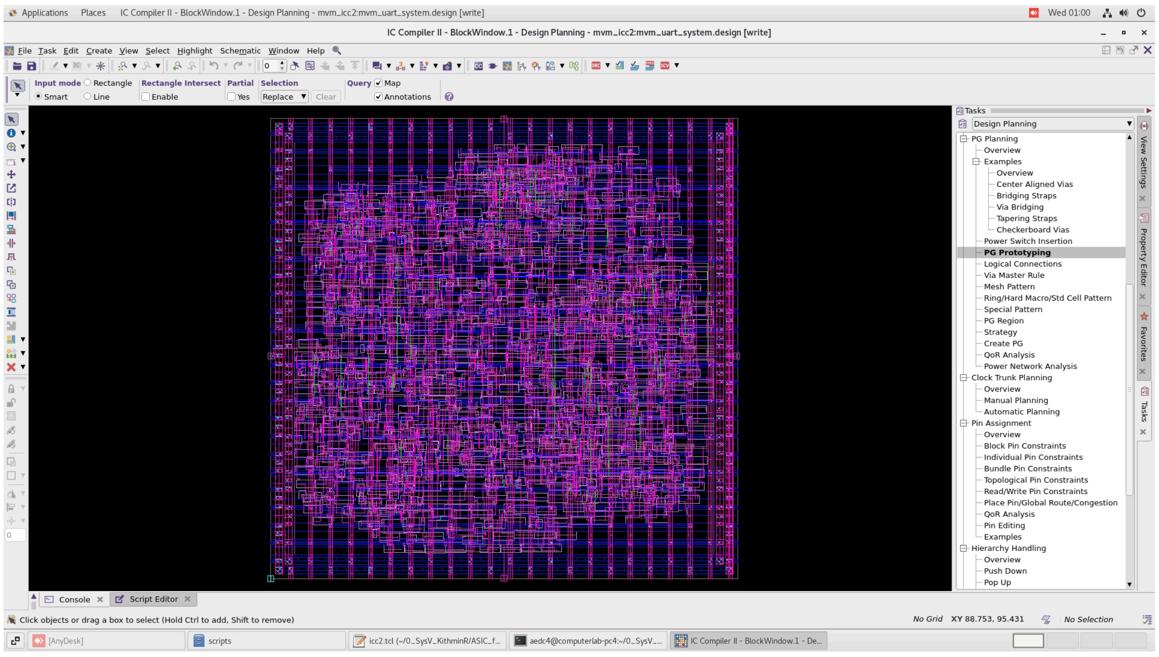
2. Go inside the **report**/ folder and find the different reports generated during the synthesis process of the mvm_uart_system. Each of the reports contain information about
    a. Ports
    b. Area

c. Power
d. Timing
e. No. of Cells

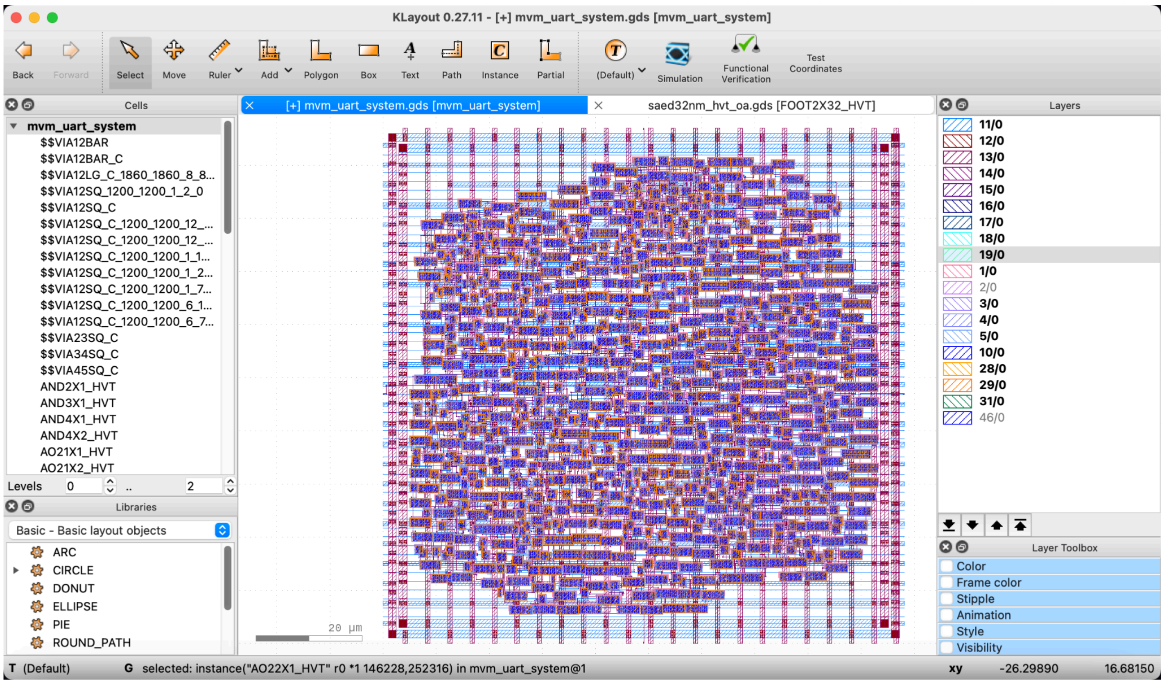
Include the information from these reports in your report as well.

3. Perform the synthesis step for several combinations of R, C, W_X, W_K parameter values in the top module and then report the area and power in each case in a tabular format.

4. The completed PnR layout visualized inside ICC2 (example shown below) - use the ‘pwd’ command on the terminal to show the directory you are inside, which should have which should have **the folder with your name**.



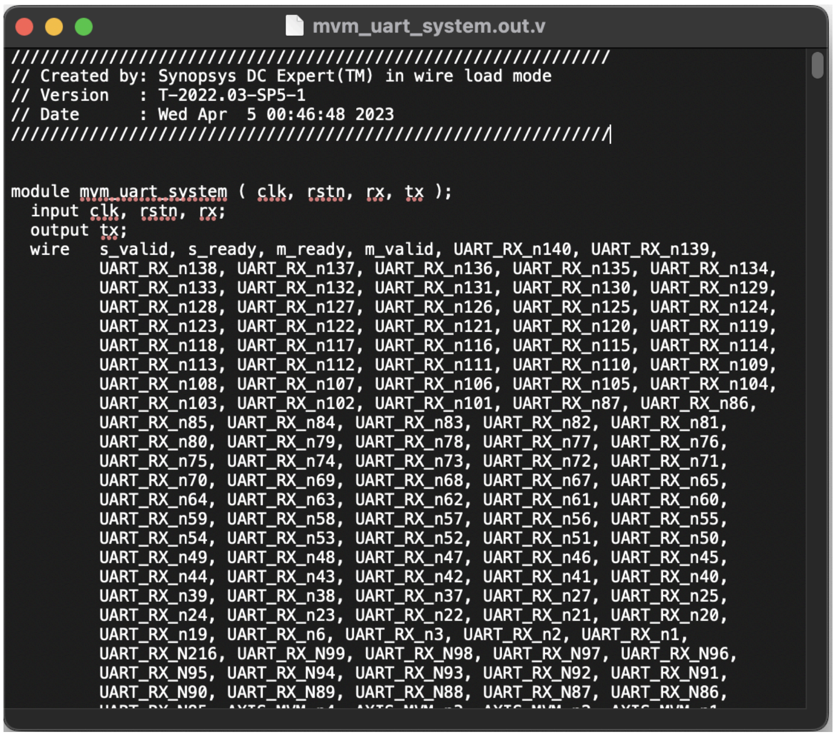
5. The final exported GDSII file visualized using kLayout (example shown below) - this is an **optional** part of the assignment.



Hints

1. In, **run_dc.tcl**, change name of top module.

2. Open the mvm_uart_system.out.v file inside "***output/***" folder to make sure the file was synthesized correctly. You can check this on the DC Design Vision gui as well.

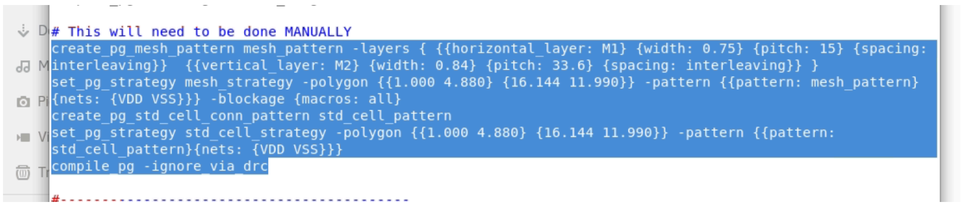


```
/////////////////////////////////////////////////////////////
// Created by: Synopsys DC Expert(TM) in wire load mode
// Version   : T-2022.03-SP5-1
// Date      : Wed Apr  5 00:46:48 2023
/////////////////////////////////////////////////////////////


module mvm_uart_system ( clk, rstn, rx, tx );
  input clk, rstn, rx;
  output tx;
  wire   s_valid, s_ready, m_ready, m_valid, UART_RX_n140, UART_RX_n139,
         UART_RX_n138, UART_RX_n137, UART_RX_n136, UART_RX_n135, UART_RX_n134,
         UART_RX_n133, UART_RX_n132, UART_RX_n131, UART_RX_n130, UART_RX_n129,
         UART_RX_n128, UART_RX_n127, UART_RX_n126, UART_RX_n125, UART_RX_n124,
         UART_RX_n123, UART_RX_n122, UART_RX_n121, UART_RX_n120, UART_RX_n119,
         UART_RX_n118, UART_RX_n117, UART_RX_n116, UART_RX_n115, UART_RX_n114,
         UART_RX_n113, UART_RX_n112, UART_RX_n111, UART_RX_n110, UART_RX_n109,
         UART_RX_n108, UART_RX_n107, UART_RX_n106, UART_RX_n105, UART_RX_n104,
         UART_RX_n103, UART_RX_n102, UART_RX_n101, UART_RX_n87, UART_RX_n86,
         UART_RX_n85, UART_RX_n84, UART_RX_n83, UART_RX_n82, UART_RX_n81,
         UART_RX_n80, UART_RX_n79, UART_RX_n78, UART_RX_n77, UART_RX_n76,
         UART_RX_n75, UART_RX_n74, UART_RX_n73, UART_RX_n72, UART_RX_n71,
         UART_RX_n70, UART_RX_n69, UART_RX_n68, UART_RX_n67, UART_RX_n65,
         UART_RX_n64, UART_RX_n63, UART_RX_n62, UART_RX_n61, UART_RX_n60,
         UART_RX_n59, UART_RX_n58, UART_RX_n57, UART_RX_n56, UART_RX_n55,
         UART_RX_n54, UART_RX_n53, UART_RX_n52, UART_RX_n51, UART_RX_n50,
         UART_RX_n49, UART_RX_n48, UART_RX_n47, UART_RX_n46, UART_RX_n45,
         UART_RX_n44, UART_RX_n43, UART_RX_n42, UART_RX_n41, UART_RX_n40,
         UART_RX_n39, UART_RX_n38, UART_RX_n37, UART_RX_n27, UART_RX_n25,
         UART_RX_n24, UART_RX_n23, UART_RX_n22, UART_RX_n21, UART_RX_n20,
         UART_RX_n19, UART_RX_n6, UART_RX_n3, UART_RX_n2, UART_RX_n1,
         UART_RX_N216, UART_RX_N99, UART_RX_N98, UART_RX_N97, UART_RX_N96,
         UART_RX_N95, UART_RX_N94, UART_RX_N93, UART_RX_N92, UART_RX_N91,
         UART_RX_N90, UART_RX_N89, UART_RX_N88, UART_RX_N87, UART_RX_N86,
```

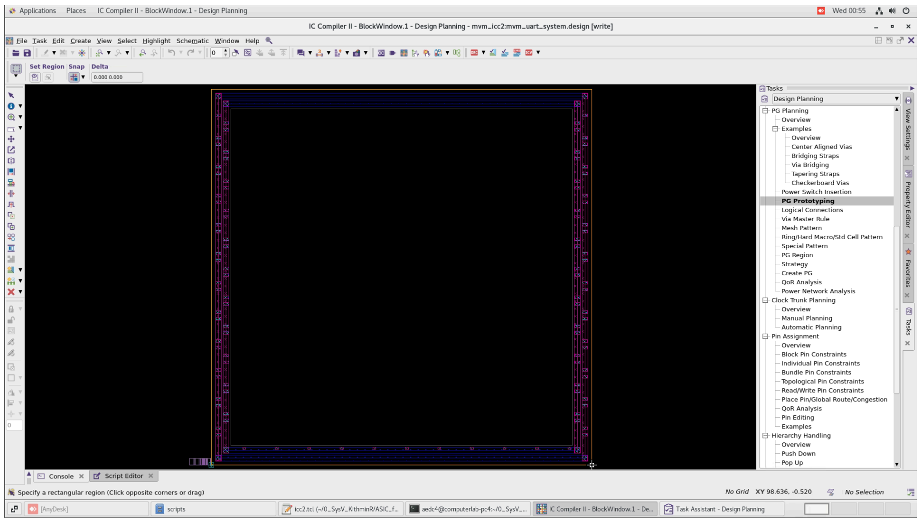
3. In **icc2.tcl**, change name of top module.

4. In **icc2.tcl**, this section of the script needs to be done manually

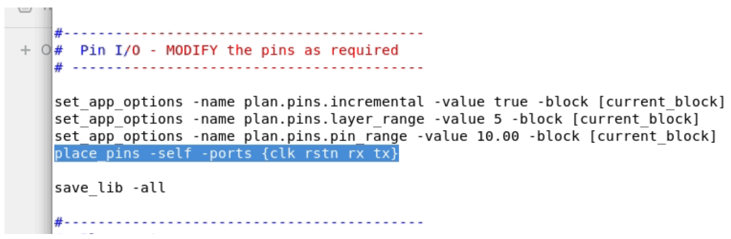


```
# This will need to be done MANUALLY
create_pg_mesh_pattern mesh_pattern -layers { {{horizontal_layer: M1} {width: 0.75} {pitch: 15} {spacing:
interleaving}}  {{vertical_layer: M2} {width: 0.84} {pitch: 33.6} {spacing: interleaving}} }
set_pg_strategy mesh_strategy -polygon {{1.000 4.880} {16.144 11.990}} -pattern {{pattern: mesh_pattern}
{nets: {VDD VSS}}} -blockage {macros: all}
create_pg_std_cell_conn_pattern std_cell_pattern
set_pg_strategy std_cell_strategy -polygon {{1.000 4.880} {16.144 11.990}} -pattern {{pattern:
std_cell_pattern}{nets: {VDD VSS}}}
compile_pg -ignore_via_drc
#-----------------------------------------
```

- Set the two PG tracks (M1, M2) to 5%, and chose the entire layout area for the polygon size



5. In **icc2.tcl**, adjust the pin names during PnR



```
#-----------------------------------------
#  Pin I/O - MODIFY the pins as required
# -----------------------------------------

set_app_options -name plan.pins.incremental -value true -block [current_block]
set_app_options -name plan.pins.layer_range -value 5 -block [current_block]
set_app_options -name plan.pins.pin_range -value 10.00 -block [current_block]
place_pins -self -ports {clk rstn rx tx}

save_lib -all

#-----------------------------------------
```