Birzeit University

Department of Electrical & Computer Engineering

First Semester 2024/2025

HW Design Lab

Project#1 – Physical Design

Name : Anas Naji

ID : 1200231

Section: #2

Instructor : Dr Abdullatif abuissa

Teaching assistant: Eng. Rania Shahwan

Contents

Objective

The goal of this project was to gain practical experience in digital and physical design by modifying an existing 8-bit processor design to support 32-bit operations. The project involved updating the data paths, control units, and memory interfaces, as well as performing synthesis, floorplanning, placement, routing, and clock tree synthesis (CTS) to meet timing, power, and area constraints.

Initial Analysis and Planning

The selected section outlines three main components of the provided 8-bit processor design :

1. ALU (Arithmetic Logic Unit)
   - Performs the core arithmetic and logical operations.
   - Supported operations include:
     1. **Addition (add):** Adds two numbers.
     2. **Subtraction (sub):** Subtracts one number from another.
     3. **Logical AND (and):** Performs a bitwise AND operation.
     4. **Logical OR (or):** Performs a bitwise OR operation.
     5. **Move (mov):** Transfers a value from one register to another.
     6. **Load Immediate (loadi):** Loads a constant value directly into a register.

2. Register File
   - A set of eight 8-bit registers used for storing intermediate values and results of computations.
   - These registers provide inputs to and store outputs from the ALU.

3. Instruction Set
   - Defines the operations the processor can execute, such as:
     1. Arithmetic operations: add, sub.
     2. Logical operations: and, or.
     3. Data transfer operations: mov, loadi.
     4. Control flow instructions: jump (j), branch if equal (beq).
   - Instructions follow a structured format, ensuring consistency in decoding and execution.

Design Specification

Key changes for the 32-bit processor involved increasing the data width from 8-bit to 32-bit, adjusting the instruction format to accommodate 32-bit operations, and expanding the control unit to decode 32-bit instructions while managing larger registers and buses.

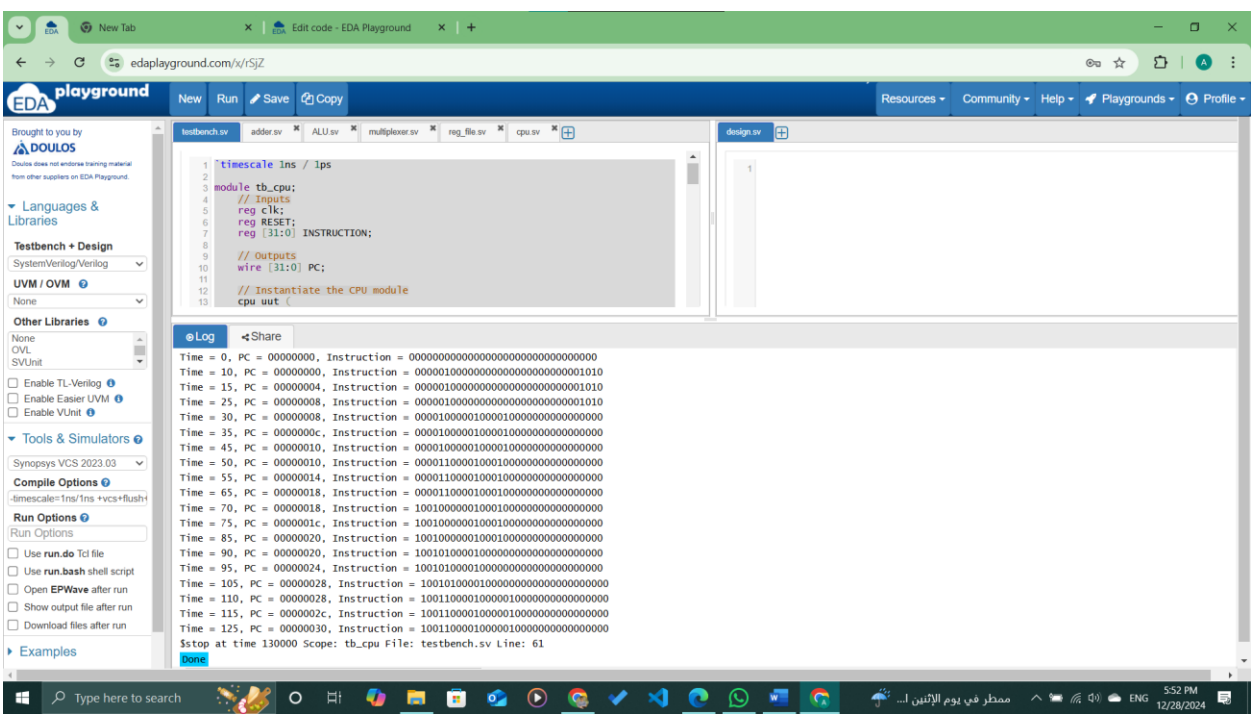
Modification and Expansion of Code

The data path was expanded by updating all data signals, registers, and buses to 32 bits, allowing for operations on larger data widths. The ALU was adjusted to handle 32-bit inputs effectively, and the instruction decoder was expanded to support the 32-bit instruction format. Additionally, new opcodes were implemented to facilitate 32-bit operations, and control signals were modified to ensure compatibility with the wider data paths and updated instruction set.

Testing and Debugging

A testbench was written to validate the design's functionality using simulation. The testbench applied a series of 32-bit instructions, including loadi, add, and, or, mov, and sub. Each instruction was executed and verified against expected outcomes. Key highlights from the simulation results include:

- The program counter (PC) updated correctly after each instruction.
- The ALU performed the specified operations, including addition, bitwise AND/OR, and subtraction, without errors.
- The zero flag was set correctly for ALU operations that resulted in a zero output.



Simulation Results

- Instructions executed correctly, as observed in the EDA Playground logs.
- All tested instructions produced the expected results, confirming the correctness of the design.

Physical Design Steps

Synthesis

To do this part we used these commands :

*commands.txt - Notepad

File Edit Format View Help

```
1. mkdir -p./1200231/EXP2
2. cd ./1200231/EXP2
3. cp -R /home/iccTA/all_labs/dc_EXP3_env/* .
-> Then we use this command : "gedit MY_DESIGN.v" to enter to the file and put our code
4. dcnxt_shell
5. source rm_dc_scripts/dc.tcl
6. read_ddc results/MY_DESIGN.elab.ddc
7.start_gui
```

Figure 1 : Synthesis codes

And the result observed is that :

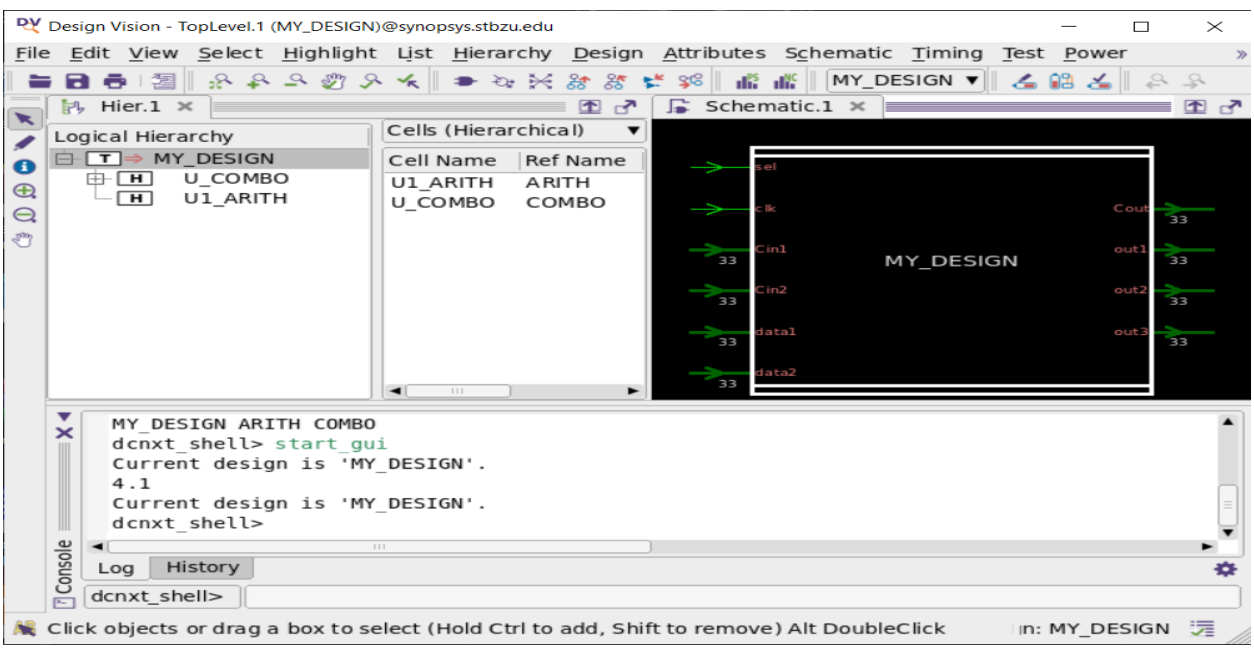


Figure 2 : Shown the circui

Floorplanning and Power Planning

To do this part we used these commands :

```
commands.txt - Notepad
File  Edit  Format  View  Help
1. mkdir -p./1200231/EXP3
2. cd ./1200231/EXP3
3. cp -R /home/iccTA/all_labs/dc_EXP4_env/* .
4. ln -s /home/BZU6/1200231/EXP2/results/MY_DESIGN.mapped.v inputs/
5. make init_design
6. icc2_shell
7. open_lib MY_DESIGN.nlib
8. list_blocks
9. open_block MY_DESIGN/init_design.design
10. start_gui
-> then after we worked in the gui window we enter this command : save_block ->
->to Save the block with the power connections
```

Figure 3 : Floorplanning and Power Planning codes

And this is the result that observed after we finish:

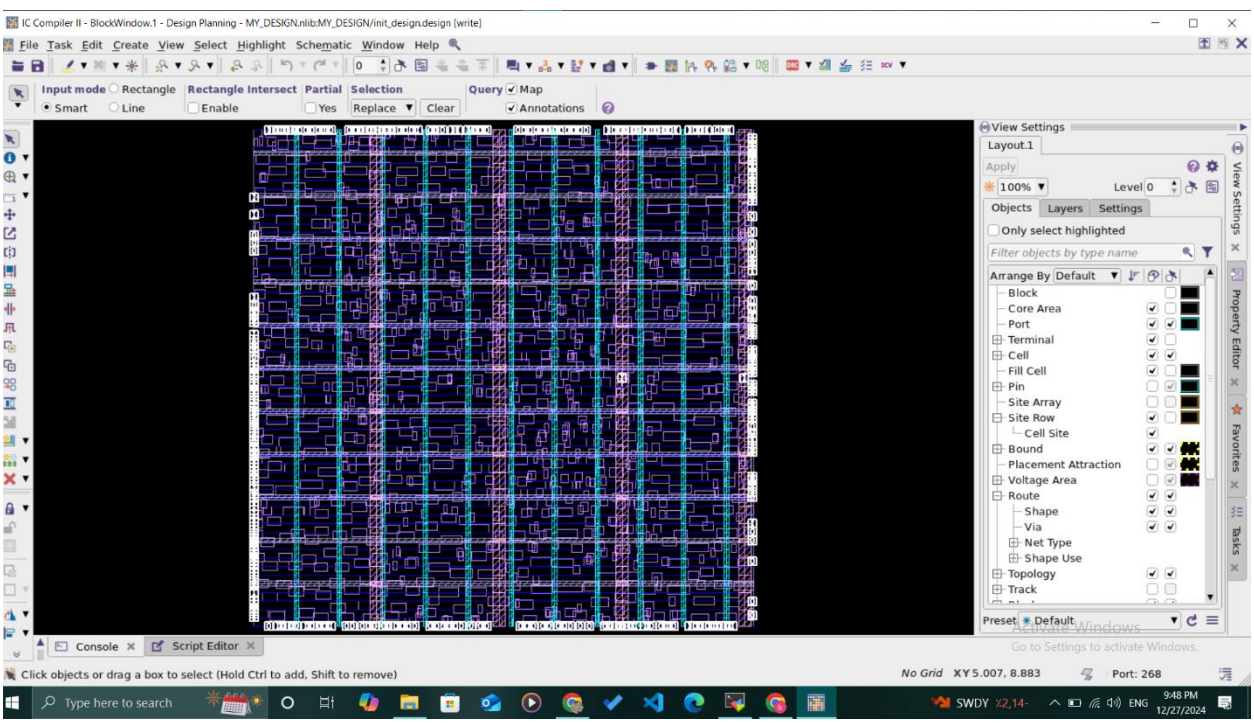


Figure 4 : Design layers

And here is assigned ports zoomed in :

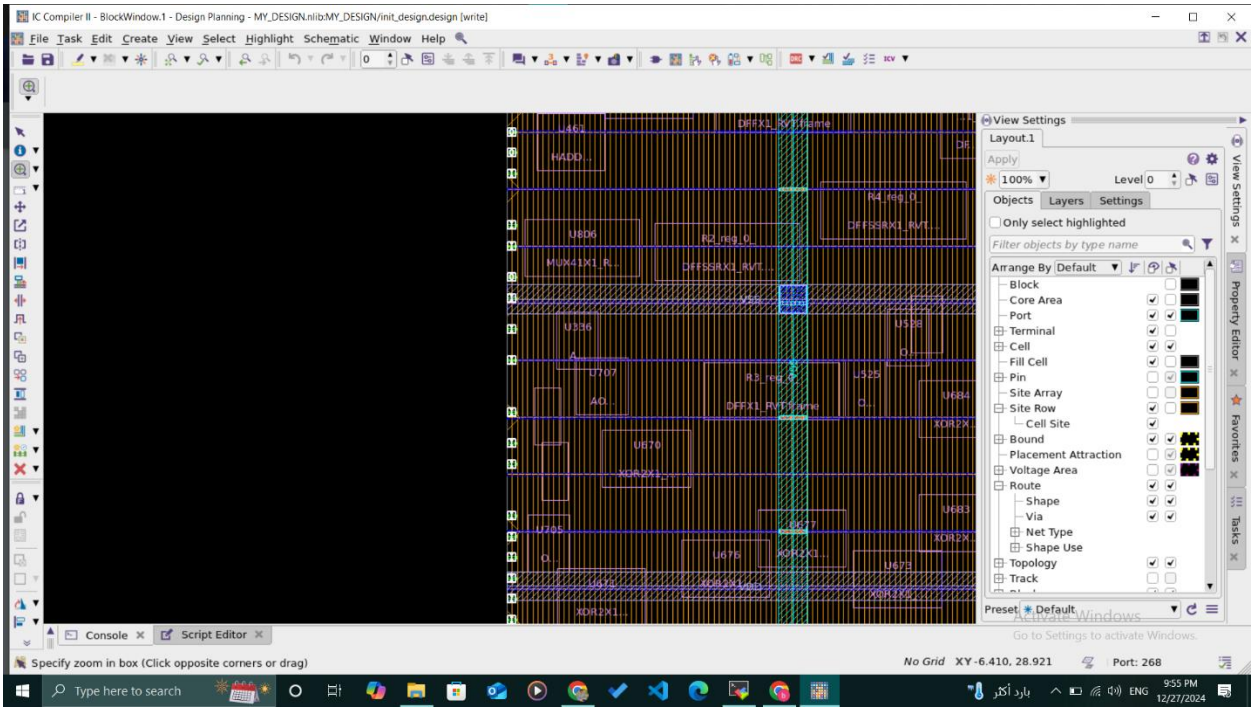


Figure 5 : Ports zoomed in

Placement

To do this part we used these commands :

*commands.txt - Notepad

File Edit Format View Help

```
1. mkdir -p./1200231/EXP4
2. cd ./1200231/EXP4
3. cp -R /home/iccTA/all_labs/dc_EXP5_env/* .
4. ln -s /home/BZU6/1200231/EXP2/results/MY_DESIGN.mapped.v inputs/
5. cp -RPi /home/BZU6/1200231/EXP3/MY_DESIGN.nlib .
6. make place_opt
7. icc2_shell
8. open_lib MY_DESIGN.nlib
9. list_blocks
10. open_block MY_DESIGN/init_design.design
11. start gui
```

Figure 6 codes for Placement

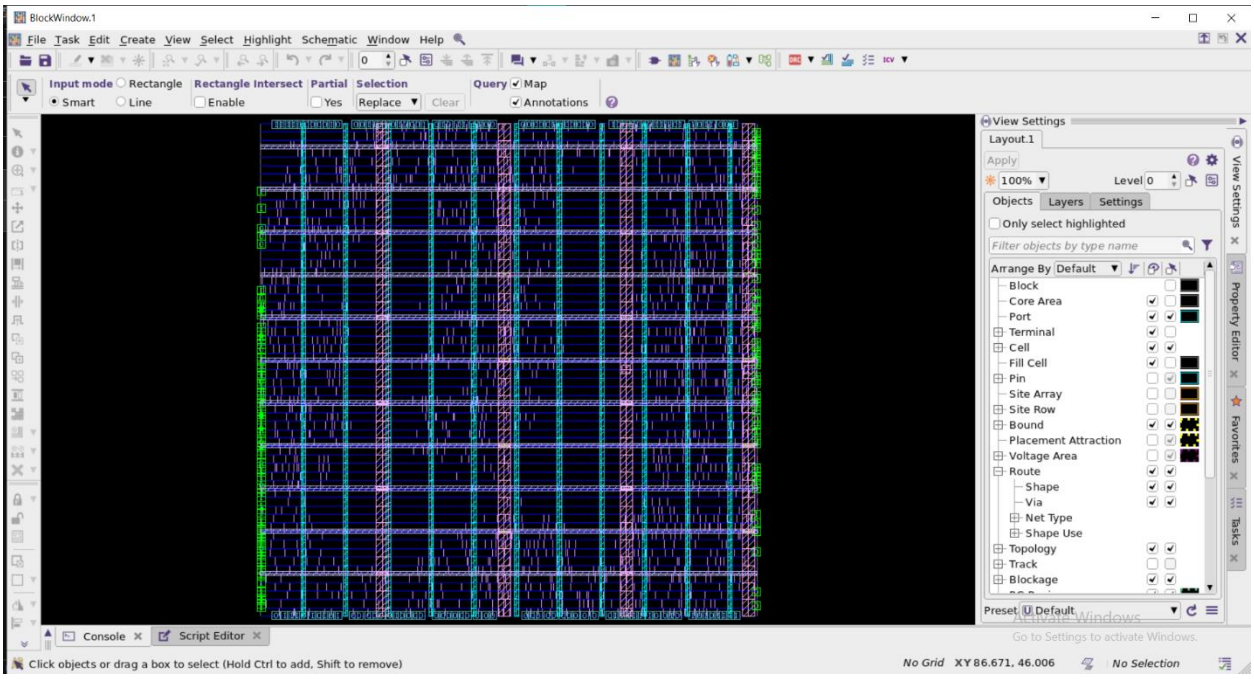


Figure 7 : Our design

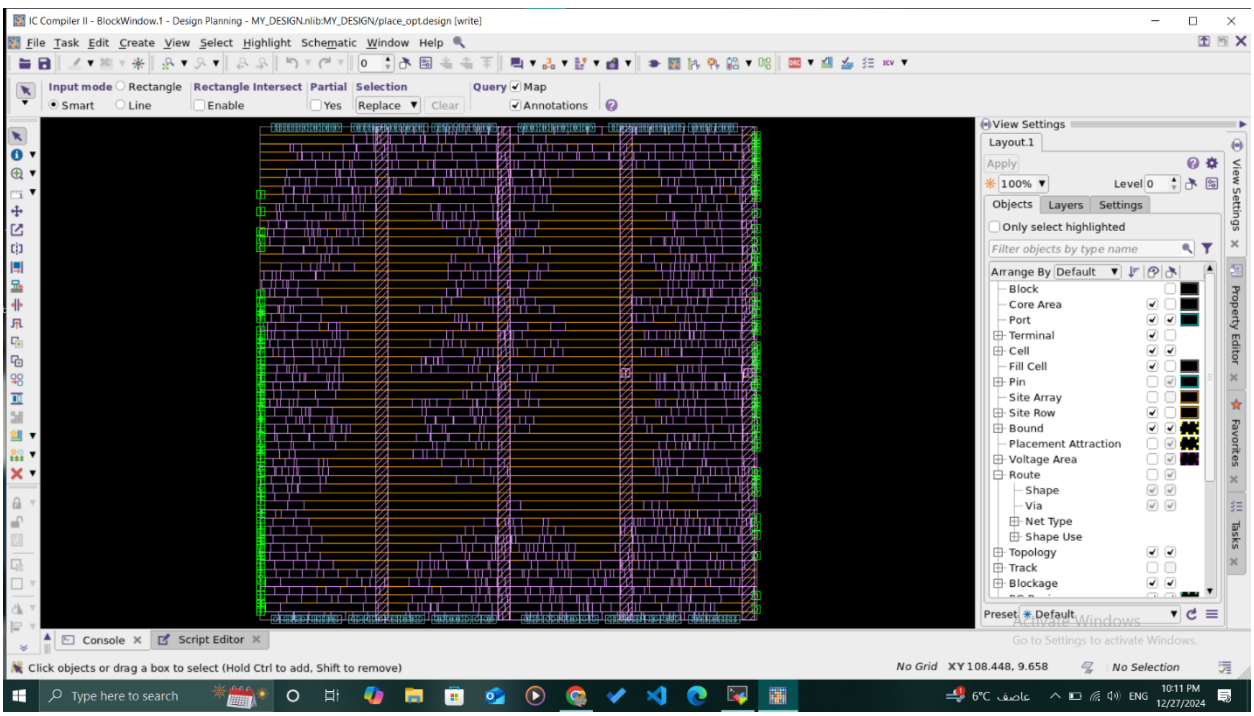


Figure 8 : : The Site Row & Cell Site on the design

And this is the result that observed after we finish:

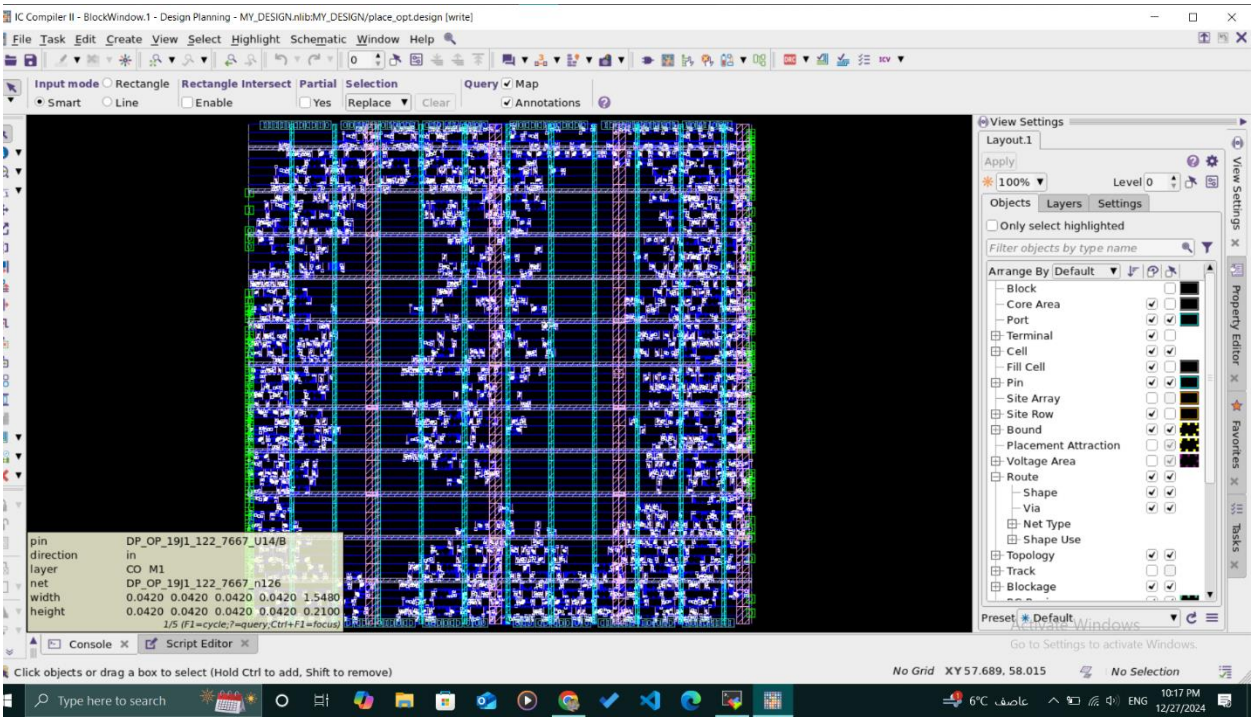


Figure 9 : final stage of Placement

confirm all the settings are read in correctly:

```
check_legality for block design MY_DESIGN succeeded!

check_legality succeeded.

**************************

1
icc2_shell>
```

Figure 10 :check_legality

```
icc2_shell> report_scenarios
****************************************
Report : scenario
Design : MY_DESIGN
Version: P-2019.03-SP2
Date   : Fri Dec 27 23:05:06 2024
****************************************

                                                   Leakage  Dynamic                          Cell  Signal
Name            Mode            Corner          Active  Setup  Hold  Power    Power    Max_tran  Max_cap  Min_cap  EM    EM
-----------------------------------------------------------------------------------------------------------------------
func::sspg      func            sspg            true    true   false true     true     true      true     false    false false

1
icc2_shell> report_modes
****************************************
Report : mode
Design : MY_DESIGN
Version: P-2019.03-SP2
Date   : Fri Dec 27 23:05:17 2024
****************************************

-------------------------------------------------------------------------------------
Mode func
-------------------------------------------------------------------------------

Current: true  Default: false  Empty: false

Scenarios associated with this mode:

                                                   Leakage   Dynamic
Scenario            Corner          Active  Setup  Hold  Power     Power     Max_tran  Max_cap  Min_cap  Cell_em  Signal_em
-----------------------------------------------------------------------------------------------------------------------------
func::sspg          sspg            true    true   false true      true      true      true     false    false    false

1
icc2_shell>
```

Activate Windows
Go to Settings to activate Windows.

ort MobaXterm by subscribing to the professional edition here: https://mobaxterm.mobatek.net

6°C عاصف ENG 10:19 PM 12/27/2024

Figure 11 : report_scenarios and report_modes

```
icc2_shell> report_clocks
****************************************
Report : clock
Design : MY_DESIGN
Mode   : func
Version: P-2019.03-SP2
Date   : Fri Dec 27 23:06:20 2024
****************************************


Attributes:
    p - Propagated clock
    G - Generated  clock
    U - Unexpanded generated clock

Clock          Period   Waveform            Attrs     Sources
-------------------------------------------------------------------------------
main_clk         1.00   {0 0.5}                       {clk}

1
icc2_shell>
```

Figure 12: report clocks

Clock Tree Synthesis (CTS)

To do this part we used these commands :

*commands.txt - Notepad

File Edit Format View Help

```
1. mkdir -p./1200231/EXP5
2. cd ./1200231/EXP5
3. cp -R /home/iccTA/all_labs/dc_EXP6_env/* .
4. ln -s /home/BZU6/1200231/EXP2/results/MY_DESIGN.mapped.v inputs/
5. cp -RPi /home/BZU6/1200231/EXP4/MY_DESIGN.nlib .
6. make clock_opt_opto
7. icc2_shell
8. open_lib MY_DESIGN.nlib
9. list_blocks
10. open_block MY_DESIGN/init_design.design
11. start_gui
```

Figure 13 : codes for Clock Tree Synthesis (CTS)

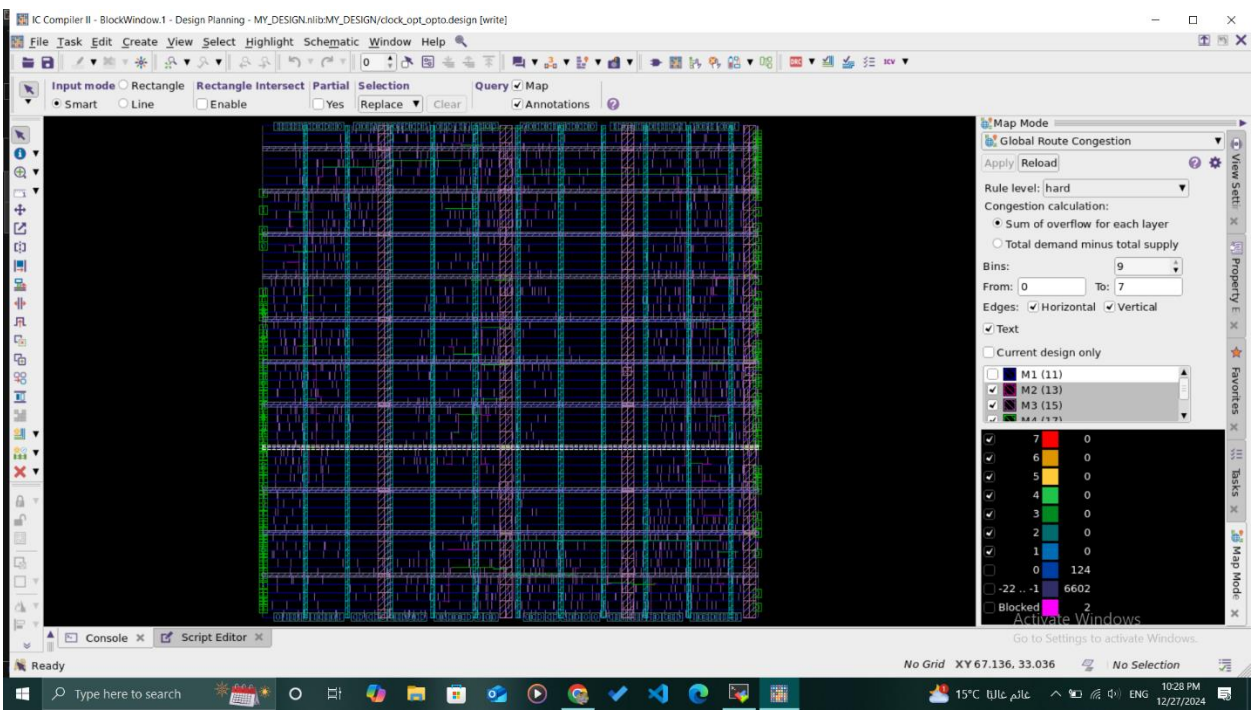


Figure 14 : our design

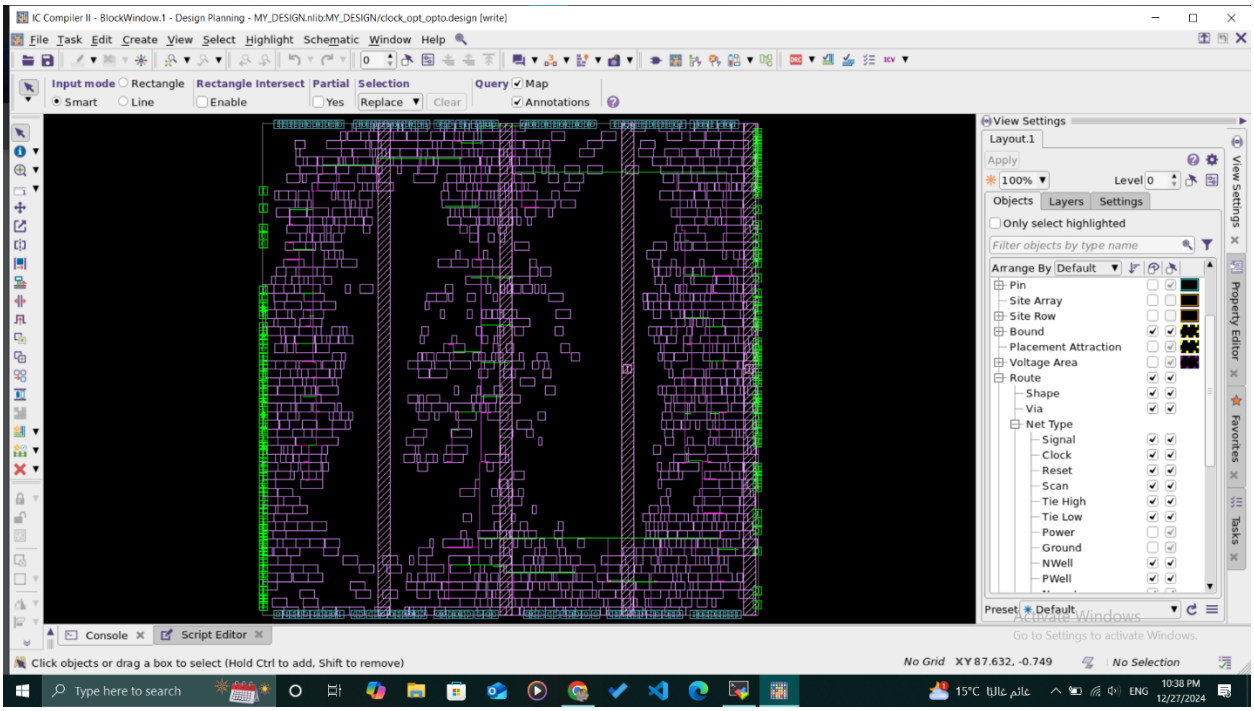


Figure 15 : Design

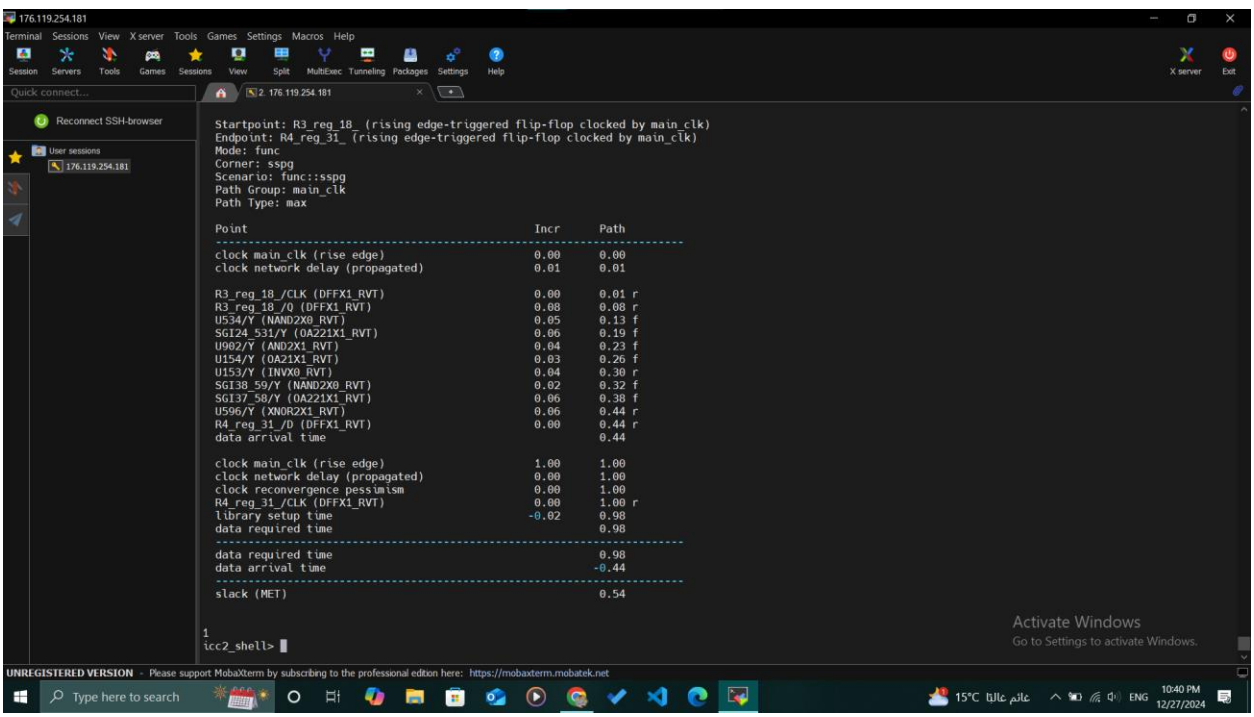


Figure 16 : Report timing

Routing

To do this part we used these commands :

commands.txt - Notepad

File Edit Format View Help

```
1. mkdir -p./1200231/EXP6
2. cd ./1200231/EXP6
3. cp -R /home/iccTA/all_labs/dc_EXP7_env/* .
4. ln -s /home/BZU6/1200231/EXP2/results/MY_DESIGN.mapped.v inputs/
5. cp -RPi /home/BZU6/1200231/EXP5/MY_DESIGN.nlib .
6. make route_opt
7. icc2_shell
8. open_lib MY_DESIGN.nlib
9. list_blocks
10. open_block MY_DESIGN/init_design.design
11. start_gui
```

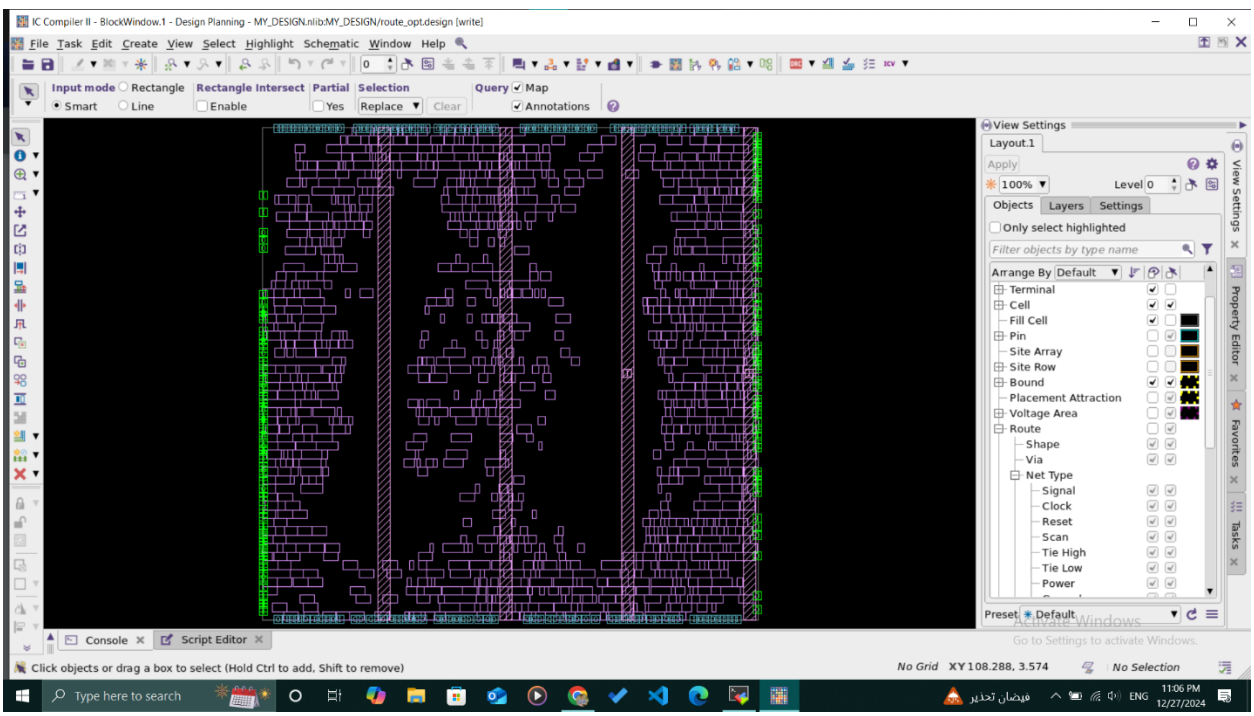


Figure 17:Design layout

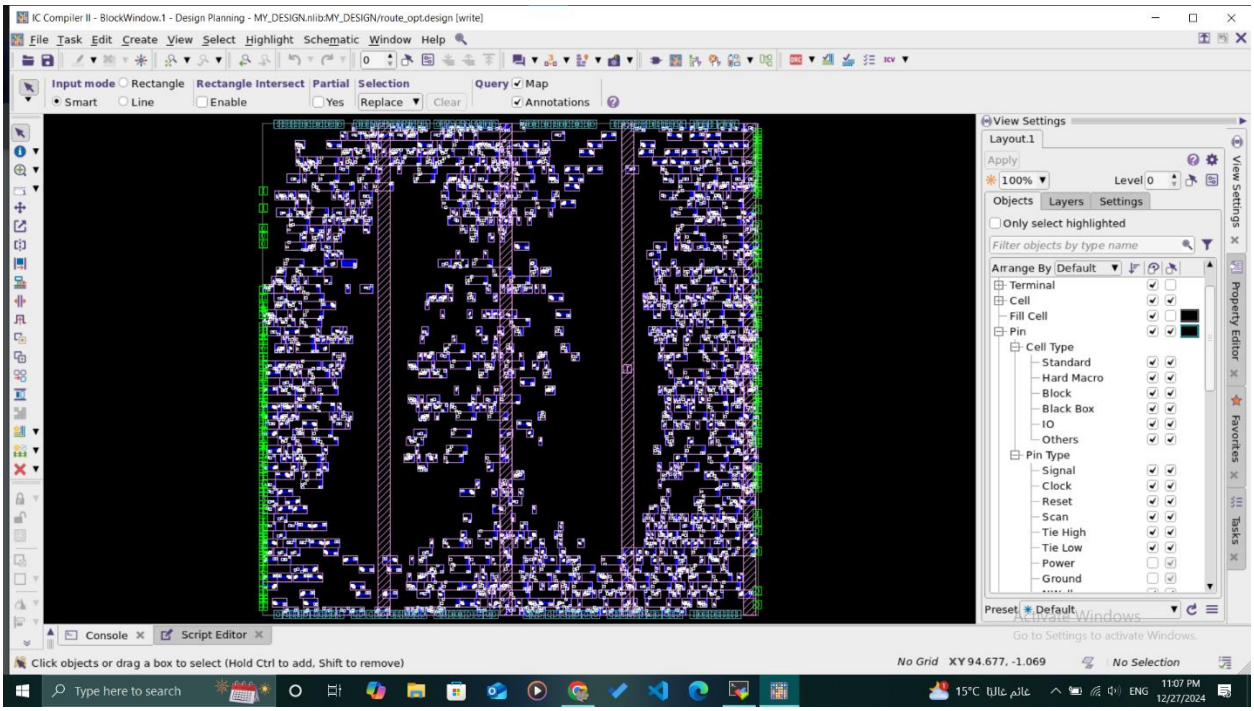


Figure 18: After disenable power pin & ground pin

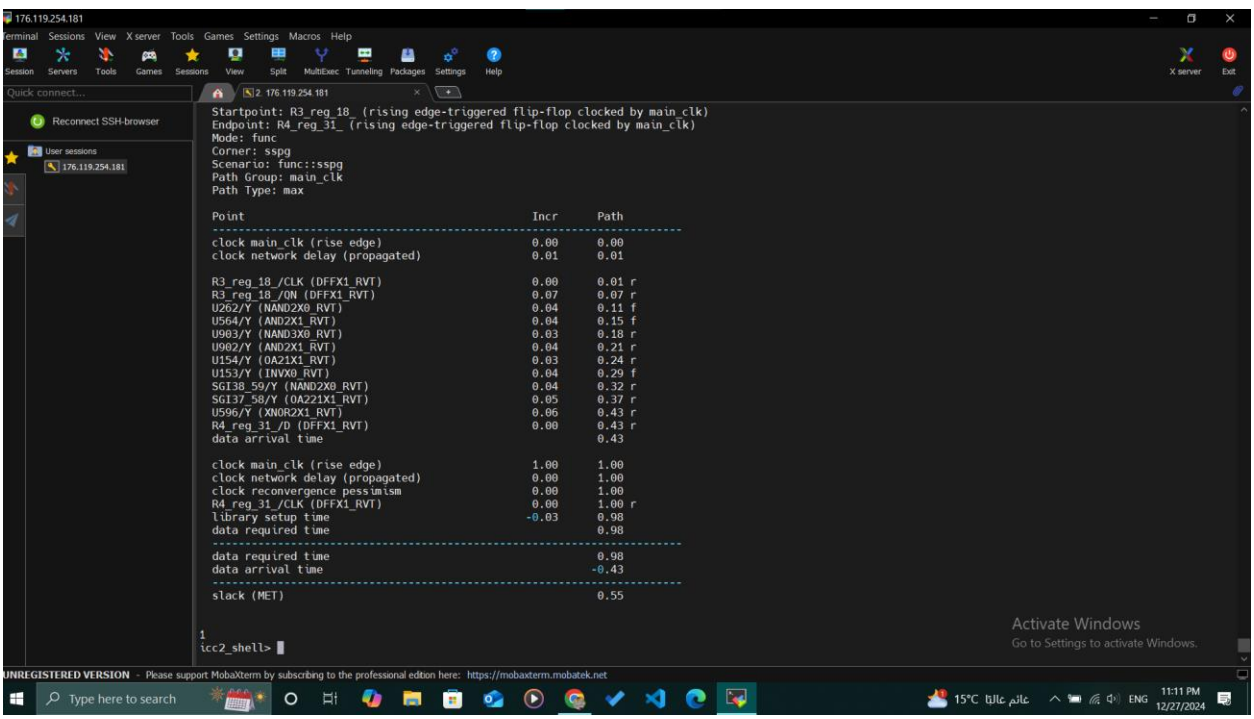


Figure 19 : Report timing

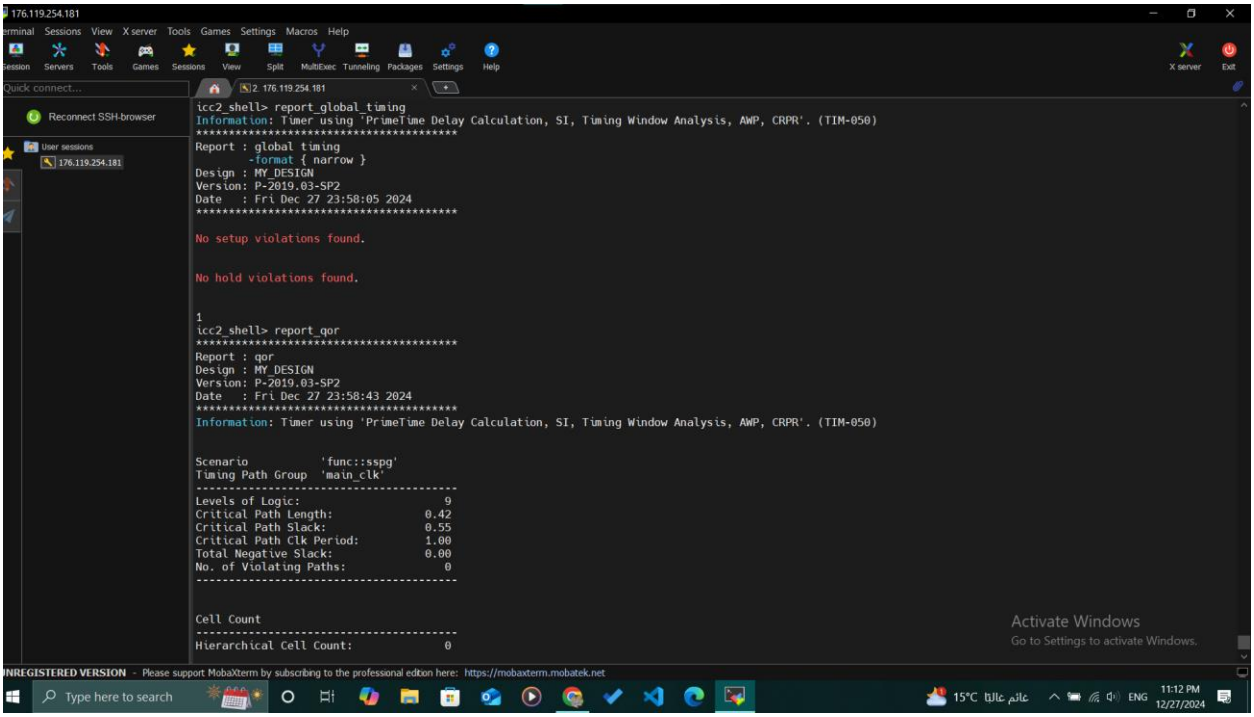


Figure 20 : Using report_global_timing & report_clock_qor

The Final Design

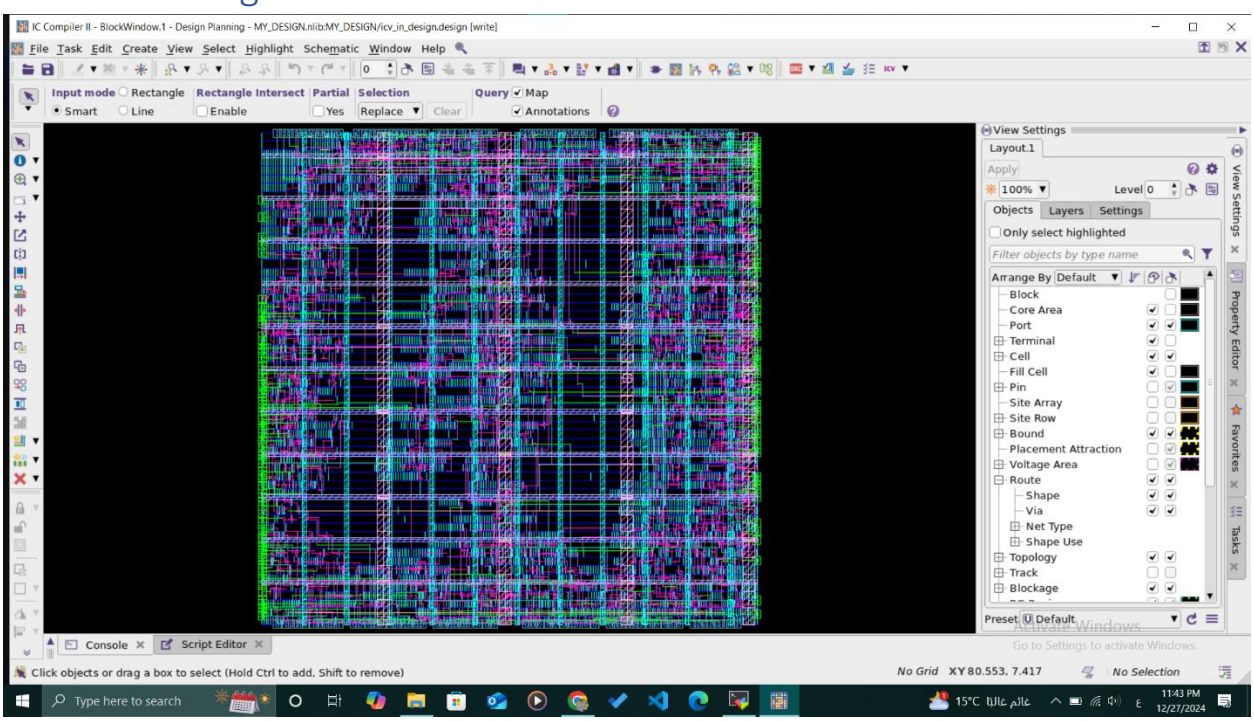


Figure 21: Design

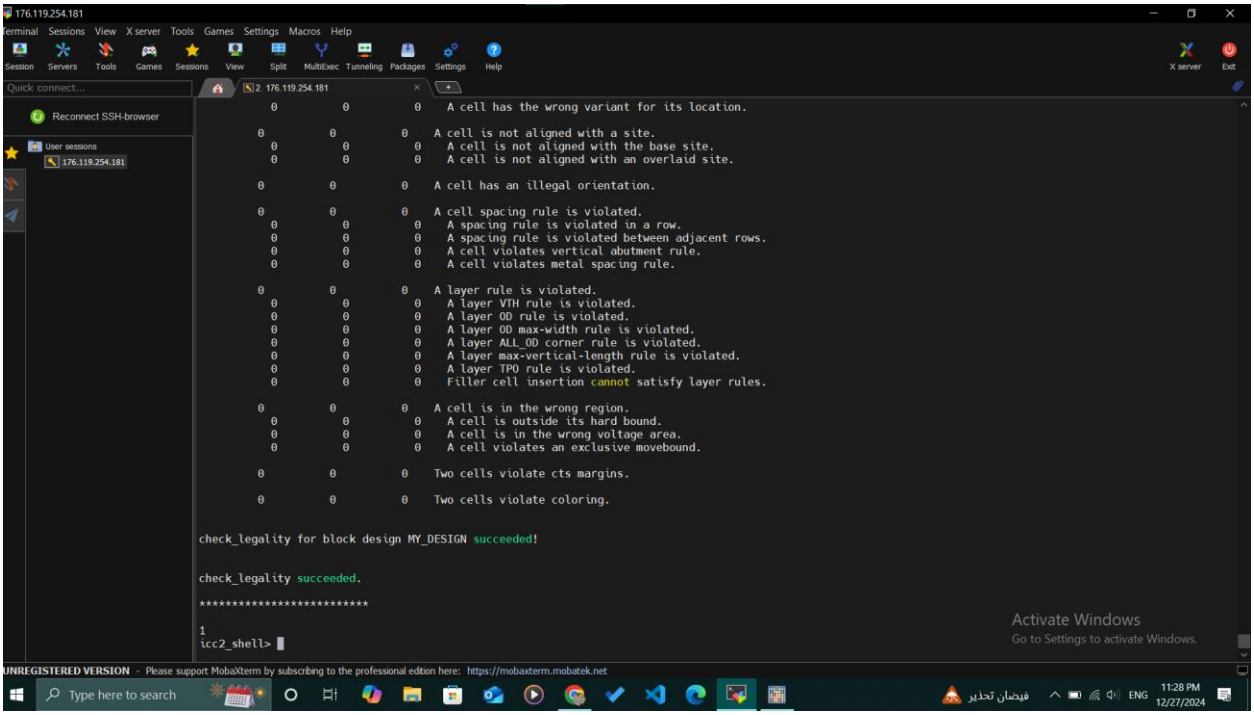


Figure 22 : cheacking legality

Verification and Validation

The verification and validation phase of the project confirmed that the design met all timing, power, and layout constraints. Static Timing Analysis (STA) showed no timing violations, ensuring the design operated correctly at the target frequency. Power analysis revealed that both leakage and dynamic power consumption were within acceptable limits, with optimizations reducing unnecessary switching activity. Design Rule Checks (DRC) and Layout Versus Schematic (LVS) validation confirmed that the layout adhered to fabrication rules and matched the intended schematic. Post-layout simulations further validated the functionality and timing of the design, with all test cases producing the expected results, demonstrating the overall correctness and robustness of the processor.

References

[1]. Synopsys | EDA Tools, Semiconductor IP and Application Security Solutions

[2] Placement | Physical Design | VLSI Back-End Adventure (vlsi-backendadventure.com)

[3] https://vlsidesignbasic.com/floorplan-basic/

[4] https://myvlsibasics.blogspot.com/2019/10/floorplan-vlsi.html

[5] https://siliconvlsi.com/power-planning/

[6] Introduction to Digital VLSI Design | by Yu | LazyYu's Blog | Medium

[7] PPT - Parallel Algorithms for VLSI Routing PowerPoint Presentation, free download - ID:1691807 (slideserve.com)

[8] https://zhuanlan.zhihu.com/p/164919161