ILLINOIS INSTITUTE OF TECHNOLOGY

ECE 429 Final Project Report

CASE STUDY FOR 32-BIT PIPELINED CPU DESIGN WITH NEW ALU ARCHITECTURE

MUNEESHA KATHULA

A20538777

FALL-2023

PROFESSOR: KEN CHOI

TA: VICTORIA KIM

Date :12-03-2023

Table of Contents:

S.no	Content	Page No
1	Introduction	1
1.1	Circuit Description	1
1.2	Memory File	2
1.3	Arithmetic Logic Unit(ALU)	2
1.4	Synchronization Theory and Implementation	3
	Case Study -1	
2	**Carry Ripple Adder**	4
2.1	RTL Simulation	4
2.2	Logical Synthesis and Post-Synthesis Simulation	6
2.3	Place & Route and Post-P&R Simulation	9
2.4	Maximum Clock Frequency	12
2.5	New Test Bench	13
3	**Carry Look Ahead Adder**	17
3.1	RTL Simulation	17
3.2	Logical Synthesis and Post-Synthesis Simulation	19
3.3	Place & Route and Post-P&R Simulation	23
3.4	Maximum Clock Frequency	27
3.5	New Test Bench	28
4	**Carry Skip Adder**	32
4.1	RTL Simulation	32
4.2	Logical Synthesis and Post-Synthesis Simulation	34
4.3	Place & Route and Post-P&R Simulation	37

4.4	Maximum Clock Frquency	42
4.5	New Test Bench	43
5	**Carry Select Adder**	48
5.1	RTL Simulation	48
5.2	Logical Synthesis and Post-Synthesis Simulation	50
5.3	Place &Route and Post-P&R Simulation	53
5.4	Maximum Clock Frequency	57
5.5	New Test Bench	58

Case Study -2

6	Comparator	64
6.1	RTL Simulation	64
6.2	Logical Synthesis and Post-Synthesis Simulation	66
6.3	Place & Route and Post-P&R Simulation	69
6.4	Maximum Clock Frequency	72

Source Codes

7.1	The Source code for new test bench for Case-1	74
7.2	The Source code for the test bench of Comparator	89
	Conclusion	106

1) Introduction:

In this final project we will understand a 32-bit Pipelined Central Processing Unit (CPU). Here the word length of the data used in the circuits is 32bits. More than one instruction is executed simultaneously because the circuit is pipelined. By using the external clock signal the synchronization is done.

1.1 Circuit Description:

The overview of the circuit is given below which has the primary building blocks (memory file and ALU) and signals for the CPU. The external clock signal synchronizes the capture and release the data within the memory file block. The pipelining of the circuit is accomplished in two clock cycles.

<u>First Clock Cycle:</u> In this two decoders are used to decode the external address selection signals and in each clock cycle multiplexer blocks are used to select the operands for ALU.

<u>Second Clock Cycle:</u> In this cycle, through the buffer state, the ALU can read data from outside of CPU.

Finally, the results of ALU can be written back in memory file which is specified by the Address B.

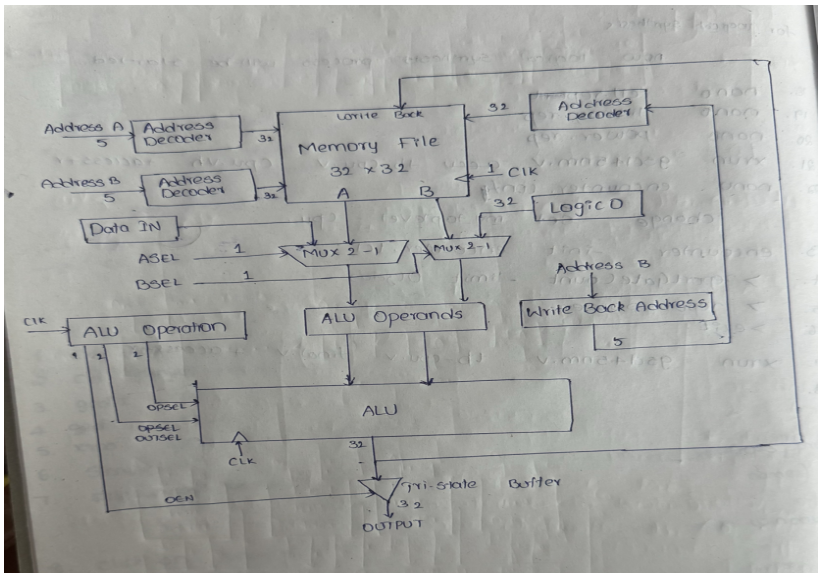


Fig. 1 :Overview of the primary blocks and signals

1.2 Memory File:

In this Project, the memory file stores 32 32-bit words in which we have two read ports and one write port. The address A and Address B are used to specified to read the words. Here, the storing element is D-register and output of each is connected to two output ports of memory , this can be done through tri-state buffers

The value of Address B specifies the word address in the memory file.

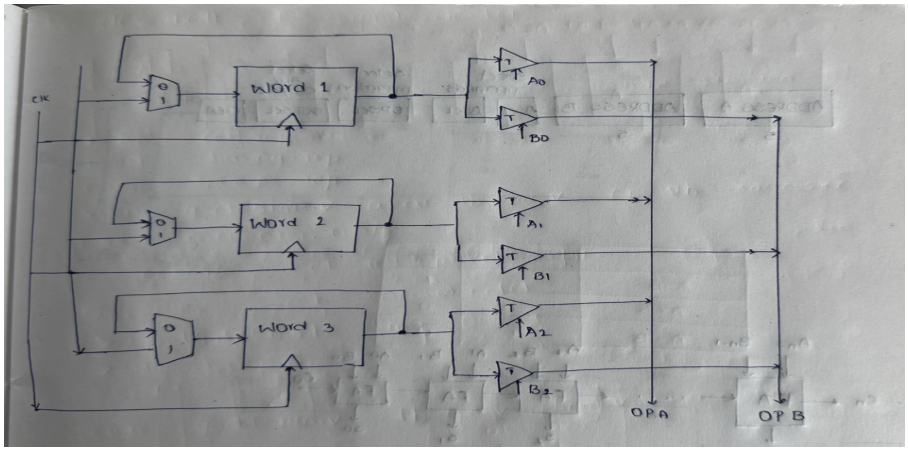


Fig.2 : Memory File Configuration

1.3 Arithmetic Logic Unit(ALU):

For this ALU Circuit we have two operands A and B and eight operations are performed.

Let A and B are two operands and ALU performes the following operations like :

- A*B : Multiplication
- A+B : Addition
- A-B :Subtraction
- B-A: Subtraction
- A or B : Logical OR Function
- A and B : Logical AND Function
- A xor B : Logical XOR Function
- A xnor B : Logical XNOR Function.

The Operand of CPU can be selected as

- ➔ Operand A can be selected between the read port A and the data in which is defined externally. This selection is done by external signal ASEL.
- ➔ Operation B can be selected between the read port B and the logic zero value. This selection is done by external signal BSEL.

The ALU has three primary operation blocks they are multiplier, adder and the logic function block. The multiplication required 3 clock cycles, it is possible by pipelinig the multiplier unit.

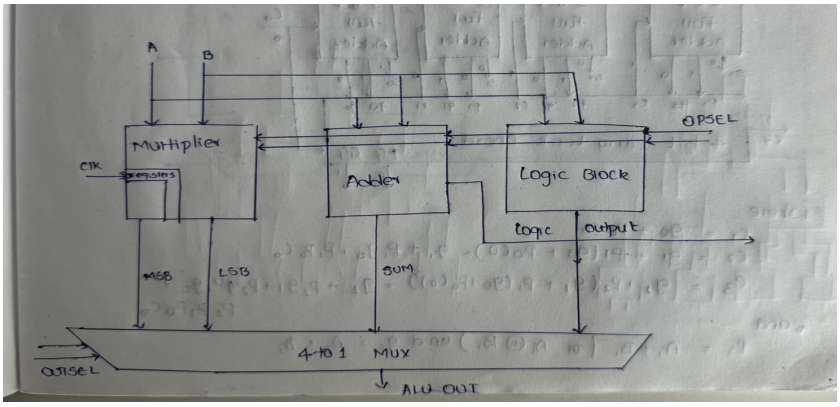


Fig.3 :Block Diagram of the ALU Circuit.

1.4 Synchronization:

An Instruction which must be executed by CPU will be determined by the external control signals. The contents of memory file and operands of ALU is specified by the control signals. In the second cycle, these control signals determines the operation of the ALU and returns the ALU Output. The ALU results are available only if the output enable signal is set and the control signals are set after a positive edge of the clock.

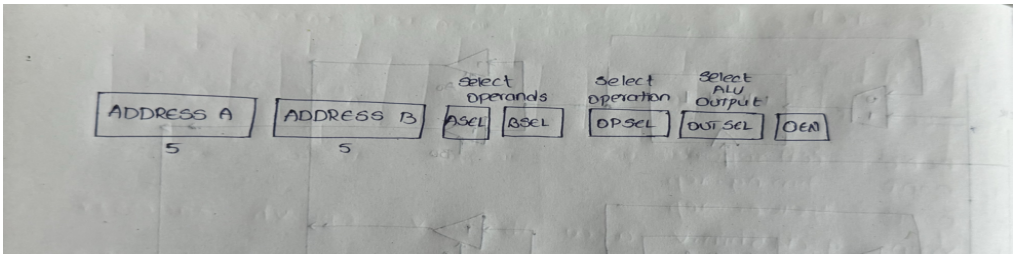


Fig.4 :Instruction word Contents

2) Theory and Implementation

Case Study – 1

32-bit CPU Design with Different Adders – Carry Ripple Adder, Carry Lookahead Adder, Carry Skip Adder, and Carry Select Adder.

We will study the design od 32-bit Design with Different Adders.

2. Carry Ripple Adder:

The Carry Ripple Adder is formed by Concatenating Full Adder in cascade. For n-bit CRA we concatenate n FA's . In this Carry Ripple Adder, the output of one Full Adder is connected to to next Full Adder.

The block Diagram of Carry Ripple Adder is as follows

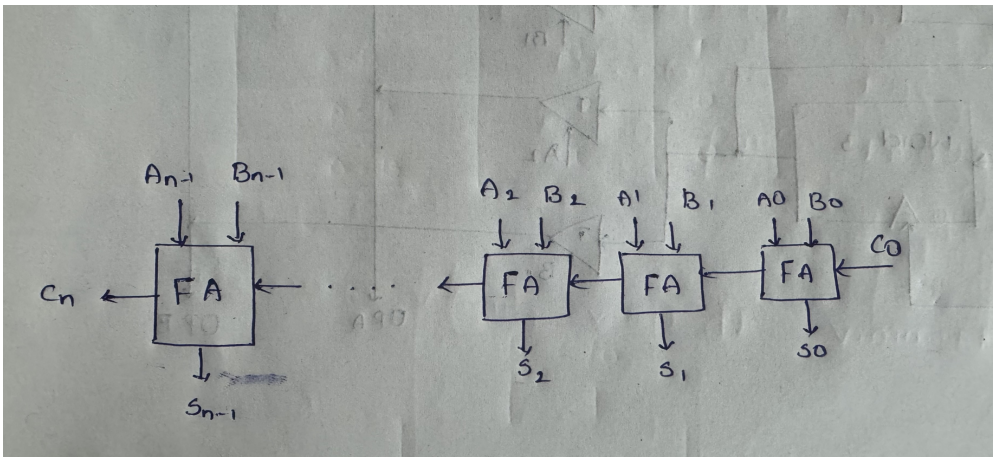


Fig.5 :Carry Ripple Adder

2.1 RTL Simulation:

In this RTL simulation we will verify the logic without delays.

The RTL Simulation is done by using the cpu_cra.v and tb_cpu.v Verilog files.

It can be run by "xrun tb_cpu.v cpu_cra.v +access+r".

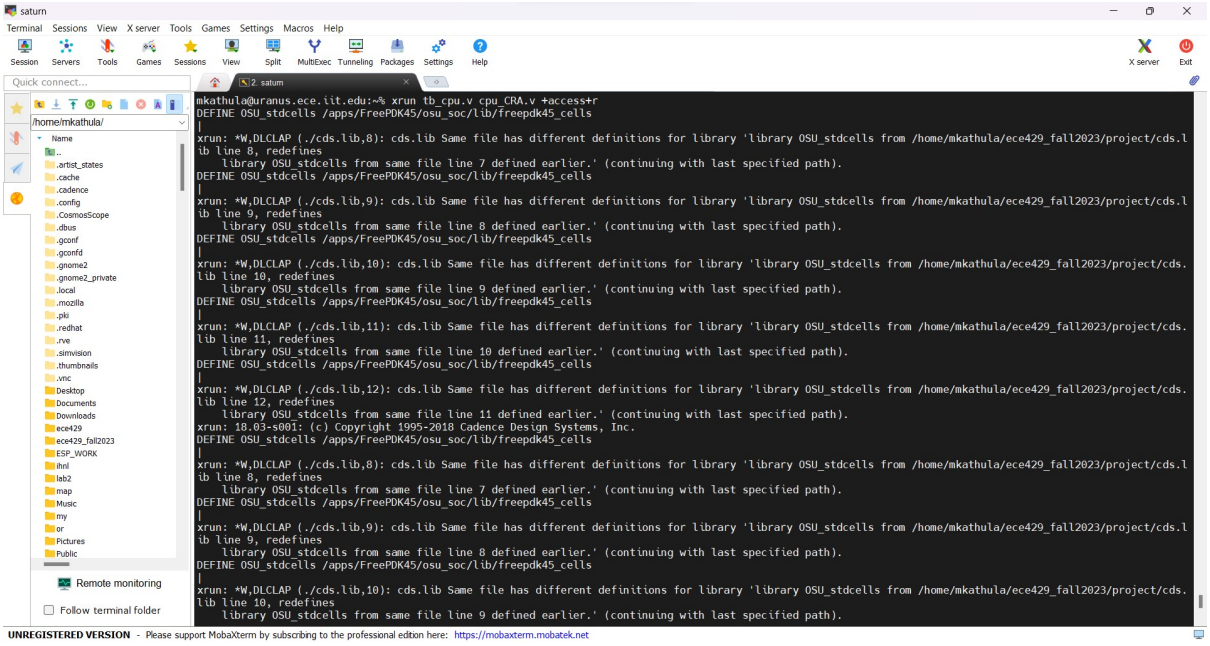


Fig.6 : CRA RTL Simulation

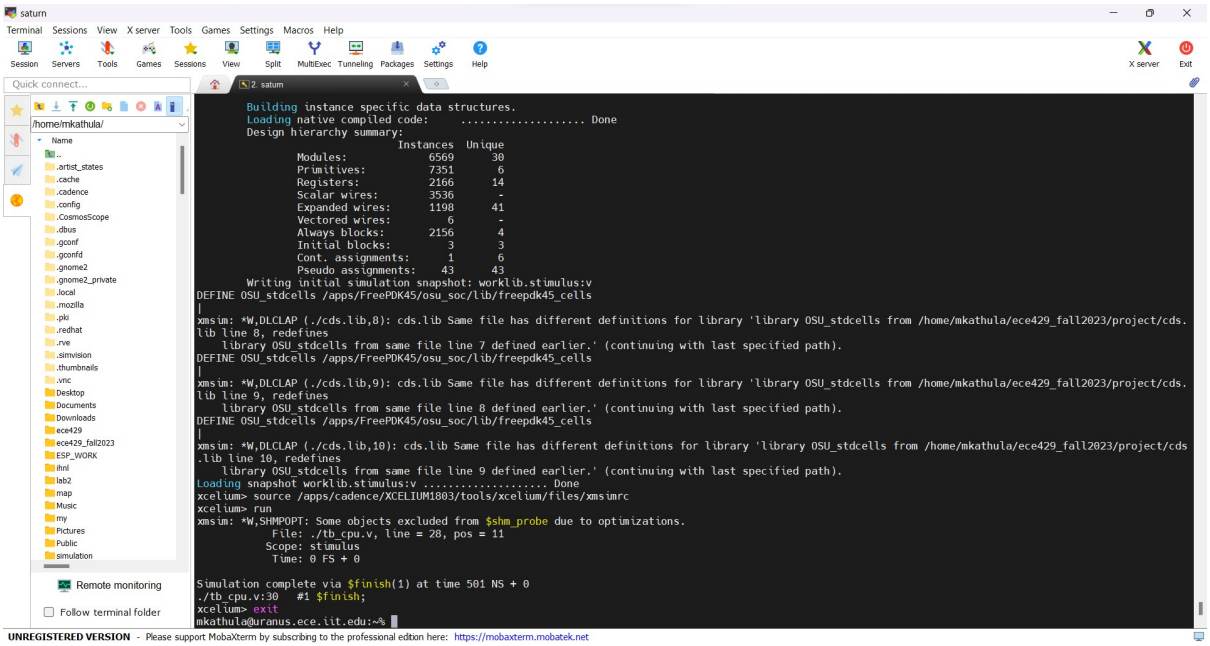


Fig.7 : CRA RTL Simulation

There are few warnings but there were no errors associated with the Verilog files.

And the wave forms can be seen by using “simvision”.

When the simvison is opened

- Open database
- Click on shm.db
- Right click on shm.db and send to waveform window.

The wave form can be as follows.

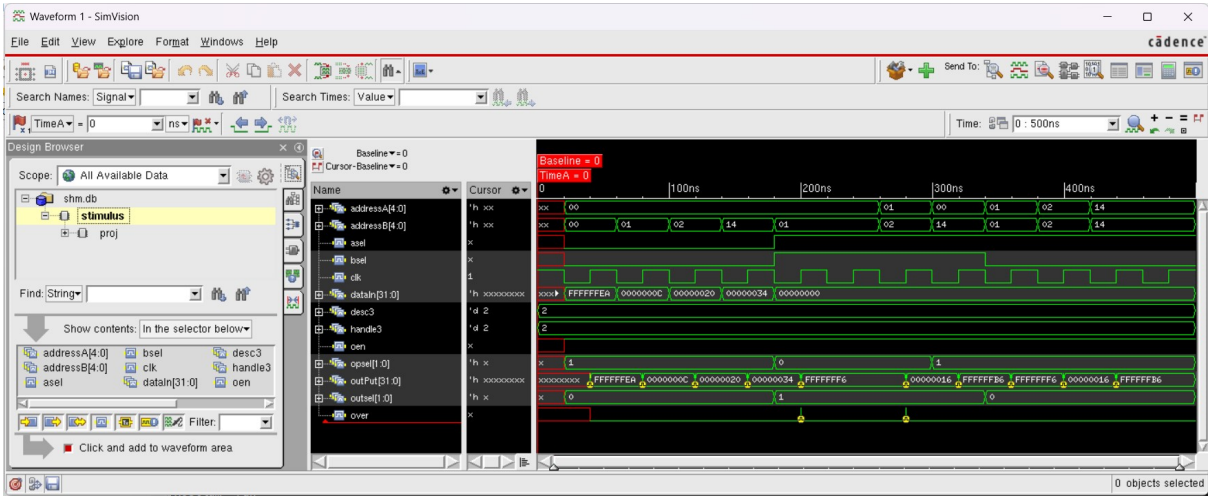


Fig.8 : Simvision Wavefrom

2.2 Logical Synthesis and Post-Synthesis Simulation:

For this we import the cadence library as

cd /import/scripts/FreePDK45_2011/FreePDK45/osu_soc/flow_ece429/* .

By this we will get the compiling files.

Here we need to change the file name and set my_toplevel to the respective filemename and module name as “[list cpu_CRA.v]” and “cpu”. The frequency must be 30MHz.

For the compile report we will run

“dc_shell -f compile_dc.tcl”

The cell report can be obtained by “nano cell.report”

It is as follows

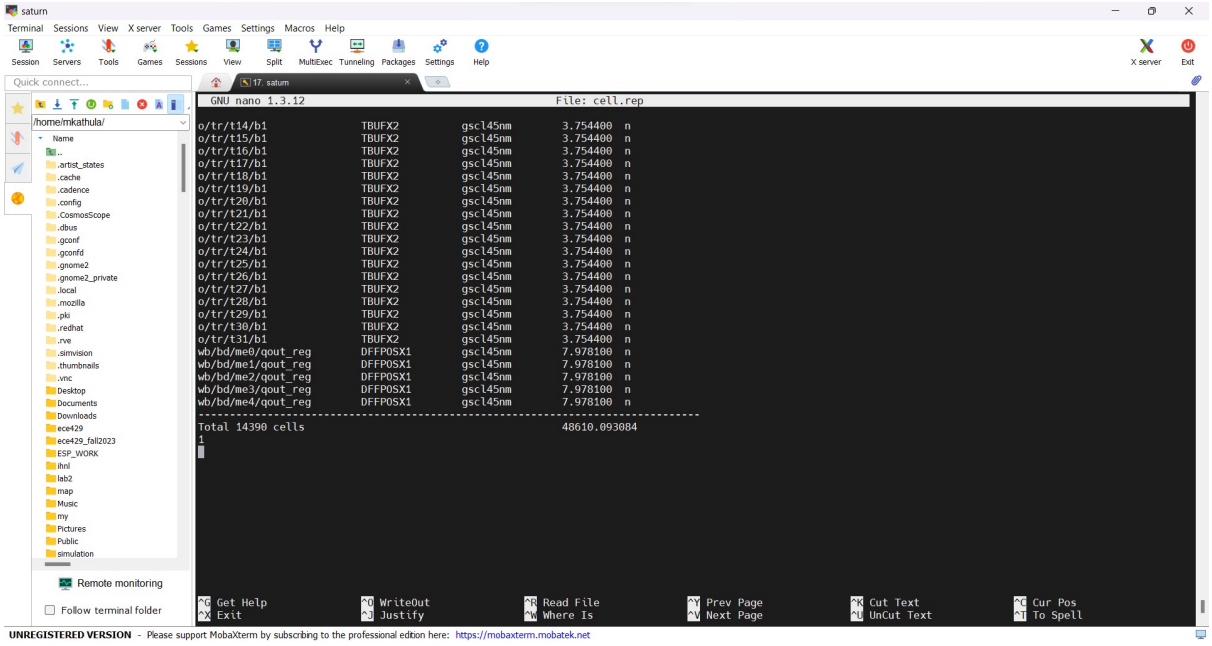


Fig 9 : Cell Report

The timing report can be obtained by “nano timing.rep”
And it is as follows

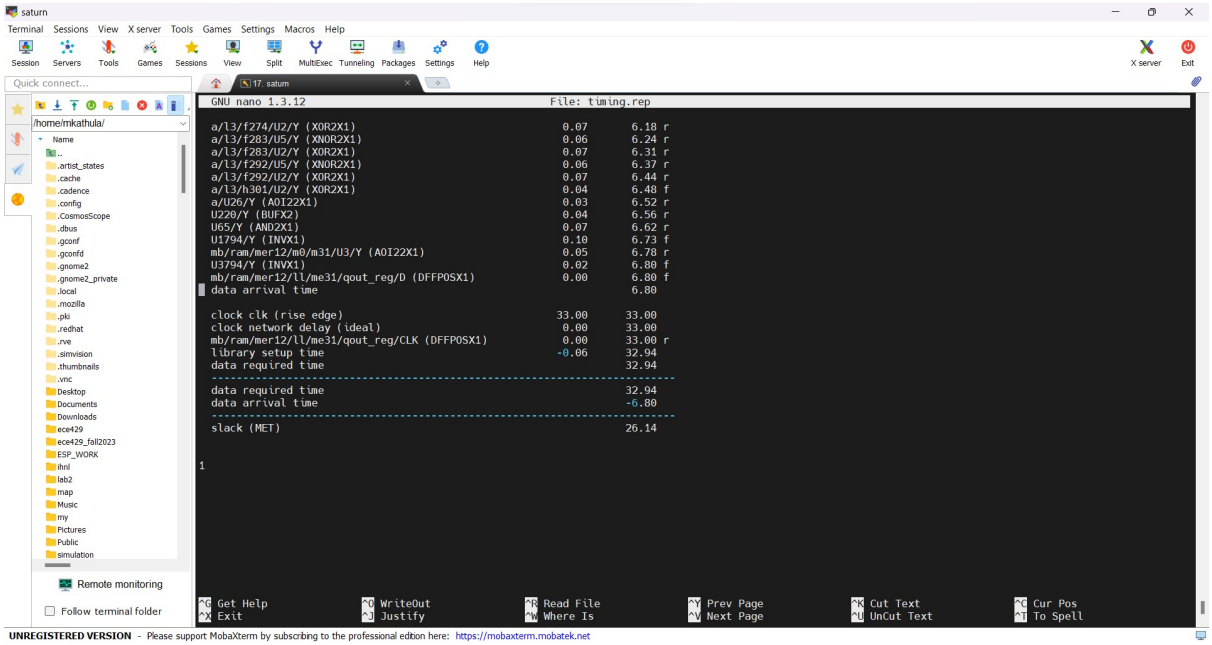


Fig.10 : Timing Report

The power report can be obtained by “nano power.rep”

It is as follows

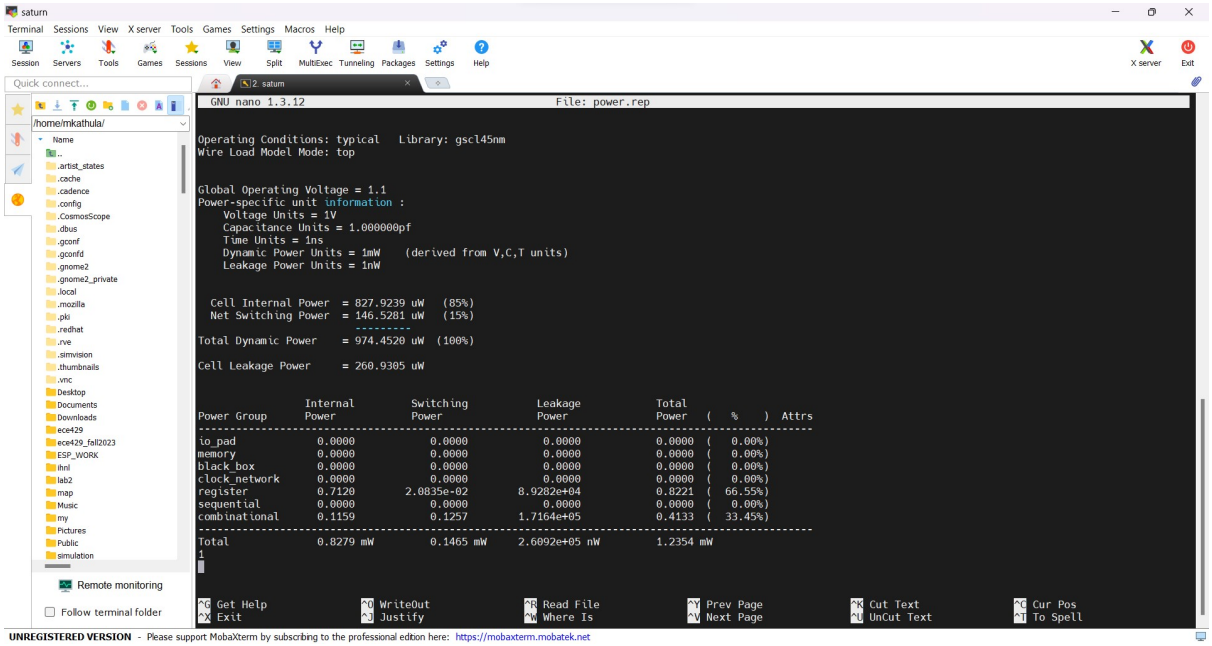


Fig.11 : Power Report

For the synthesis we will simulate with Verilog models of standard cells,i.e. gscl45nm.v, and we compare the result with rtl simulation.

This can be done by “xrun gscl45nm.v tb_cpu.v cpu.vh +access+r”

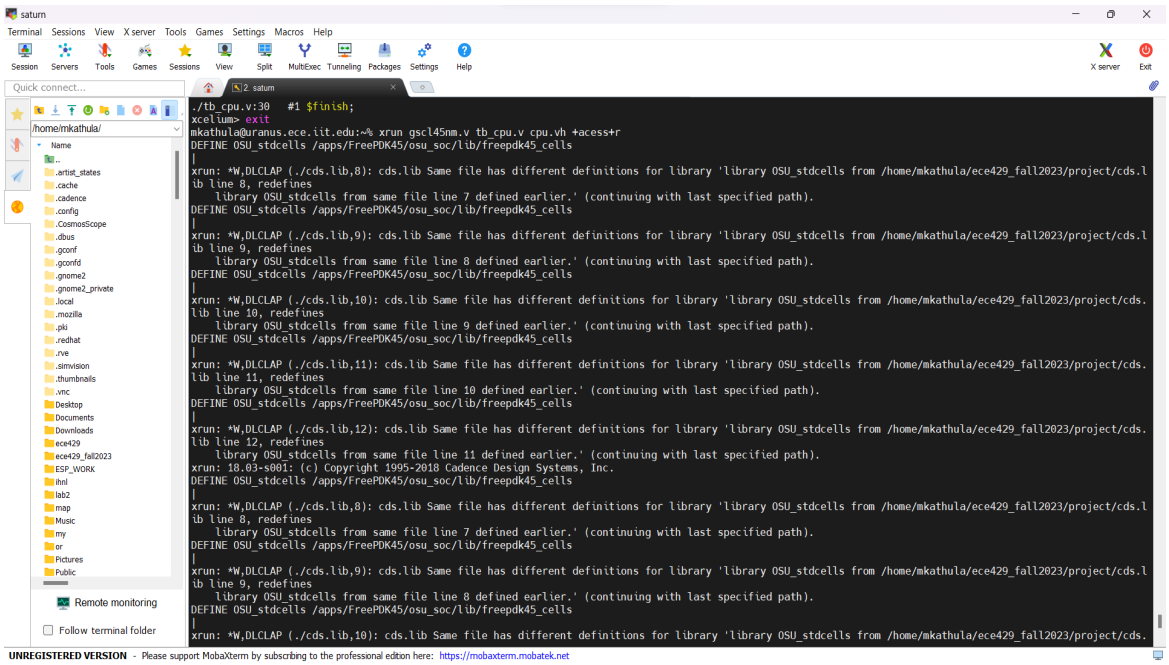


Fig. 12 :Logical and Post synthesis simulation

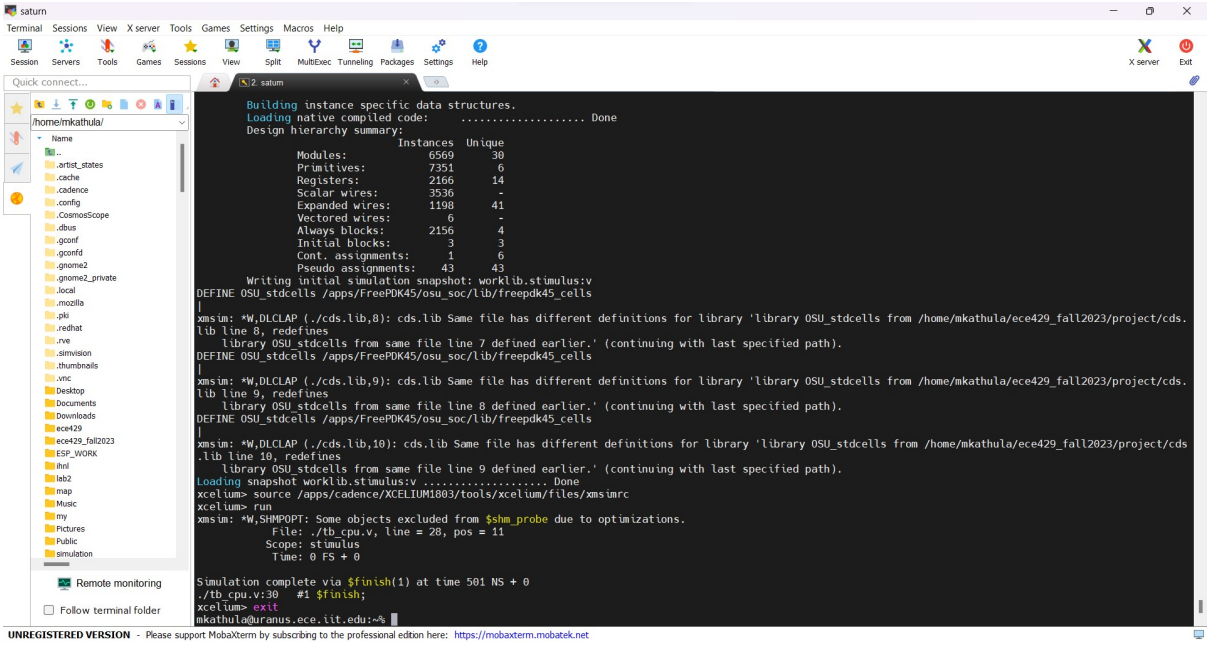


Fig. 13 :Logical and Post synthesis simulation

It shows same as the RTL Simulation.

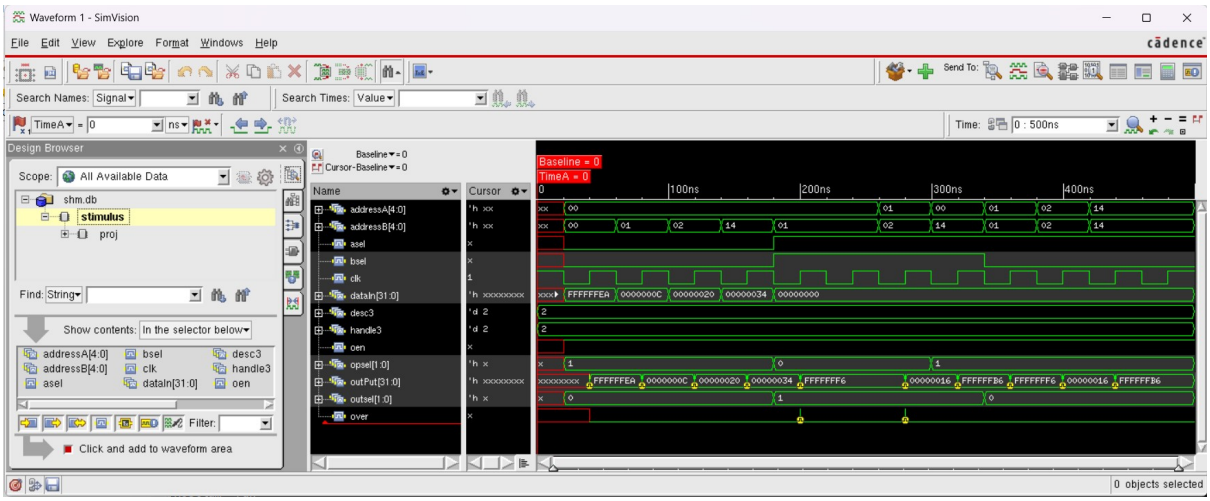


Fig14. :Simvision for Logical and post synthesis Stimulation

2.3 Place & Route and Post-P&R Simulation:

After the synthesis we will change the set my_toplevel to “cpu” in “encounter.conf”.

Now for the encounter results we run the

“encounter -init encounter.tcl”

For Gate Area we will run “reportGateCount -limit 0”.

It can be as follows

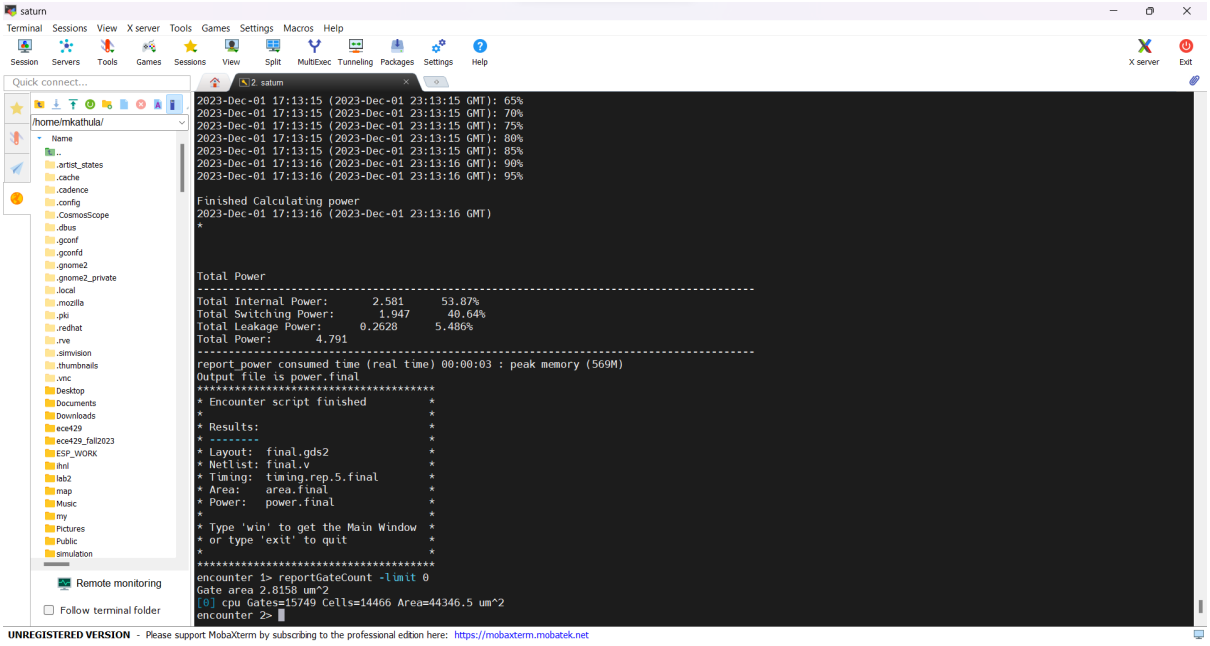


Fig.15 : Gate Area

The power can be obtained by “report_power”.

It can be as follows

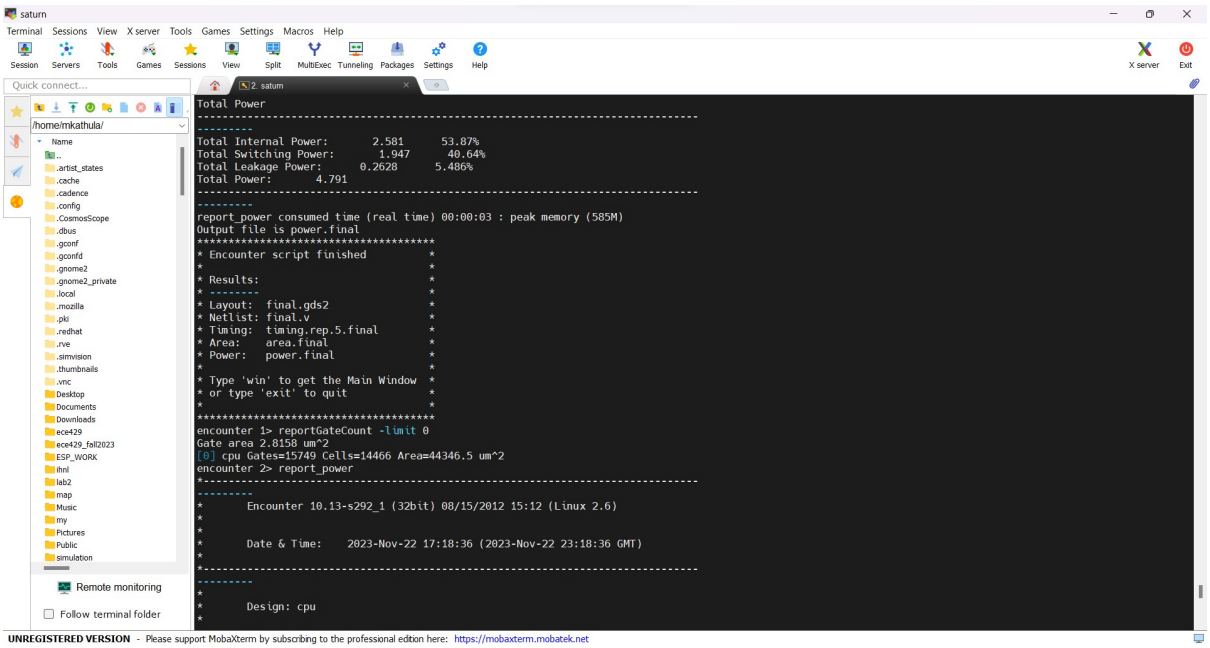


Fig16. : Report_Power

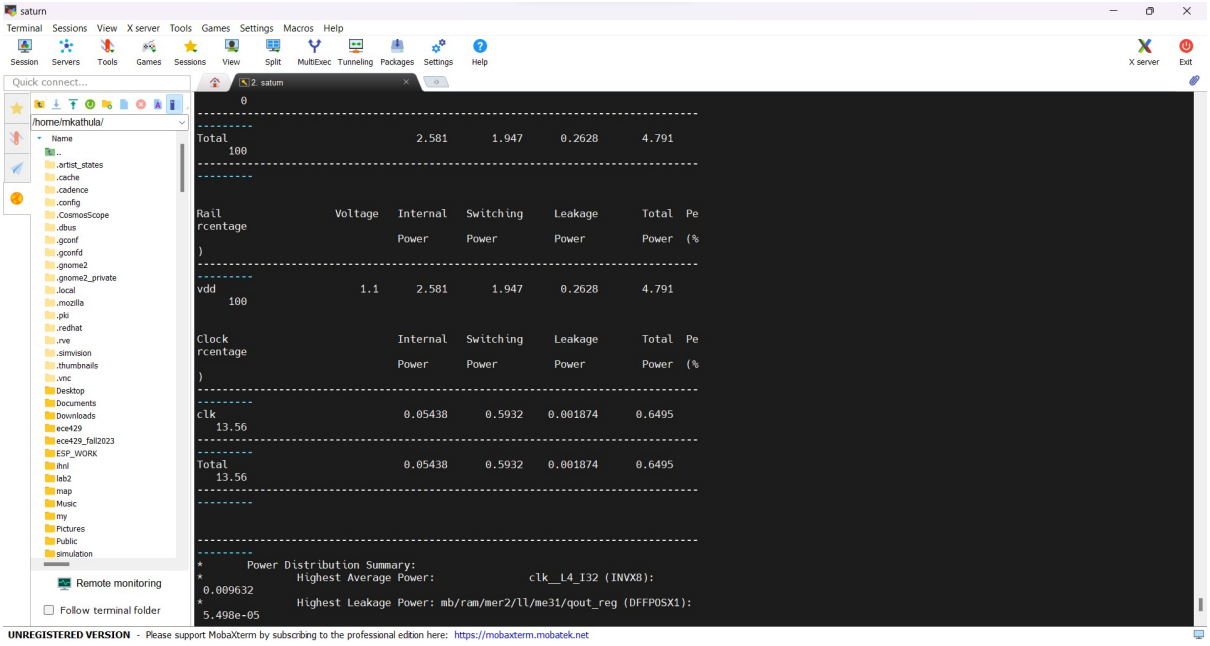


Fig.17 : Report_Power

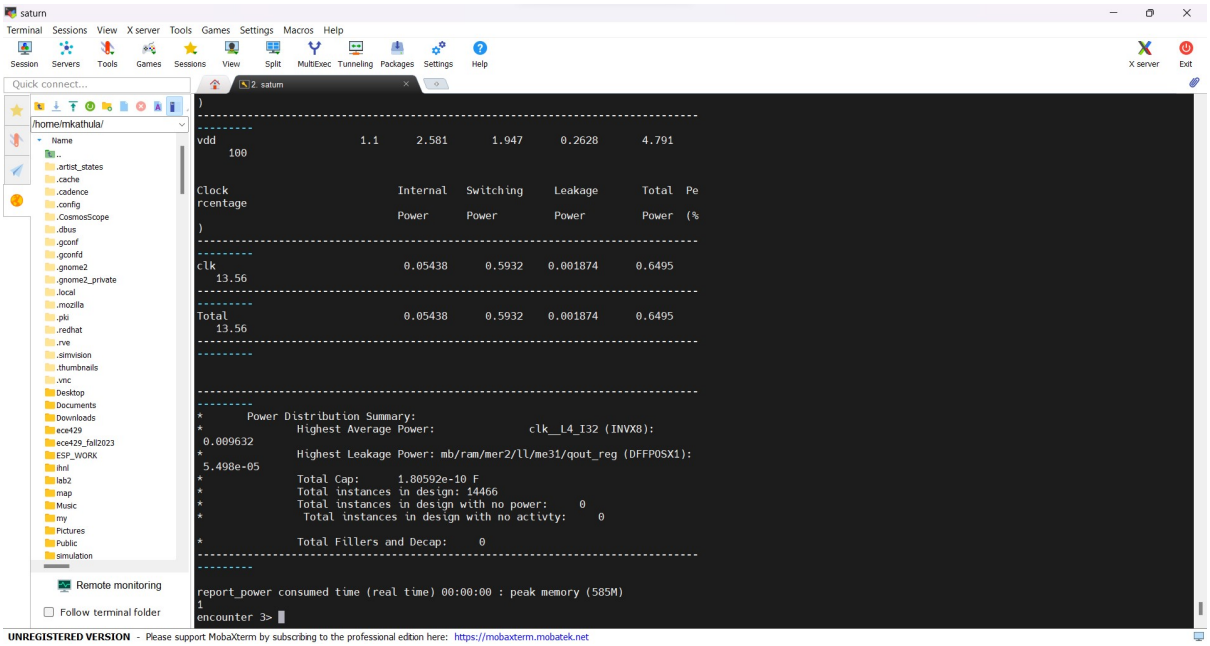


Fig.18 : Report_Power

For Place and Route and Post-P&R Simulation we run “xrun gscl45nm.v tb_cpu.v final.v +access+r”

It is as follows:

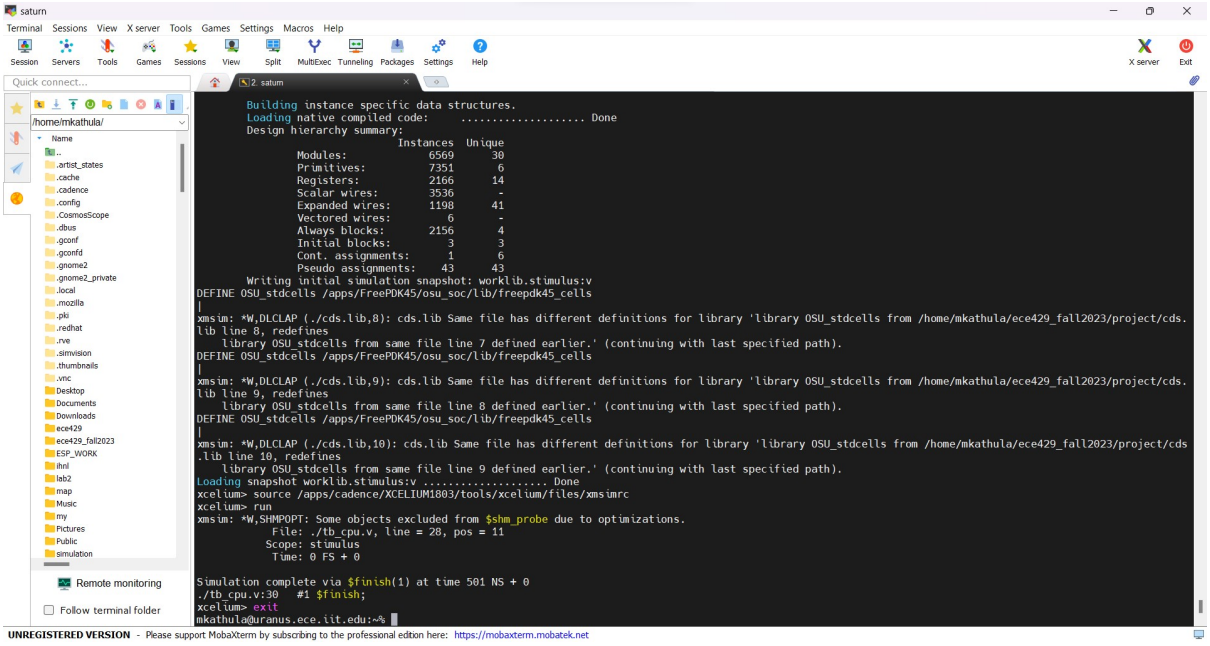


Fig.19 : Place and Route and Post-P&R Simulation

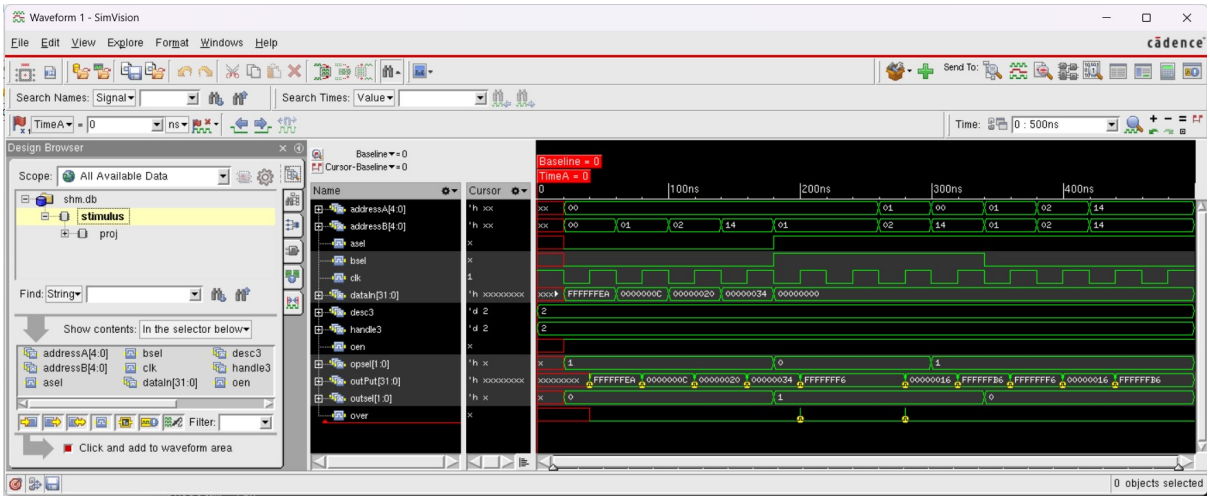


Fig.20 :Simvision Graph for Place & Route and Post-P&R Simulation

2.4 Maximum Clock Frequency:

This can be obtained in timing.rep.5.final.

Here the slack time has decreased to 17.458, Based on the required time of 28.398, max clock frequency is 35.21Mhz(1/28.398).

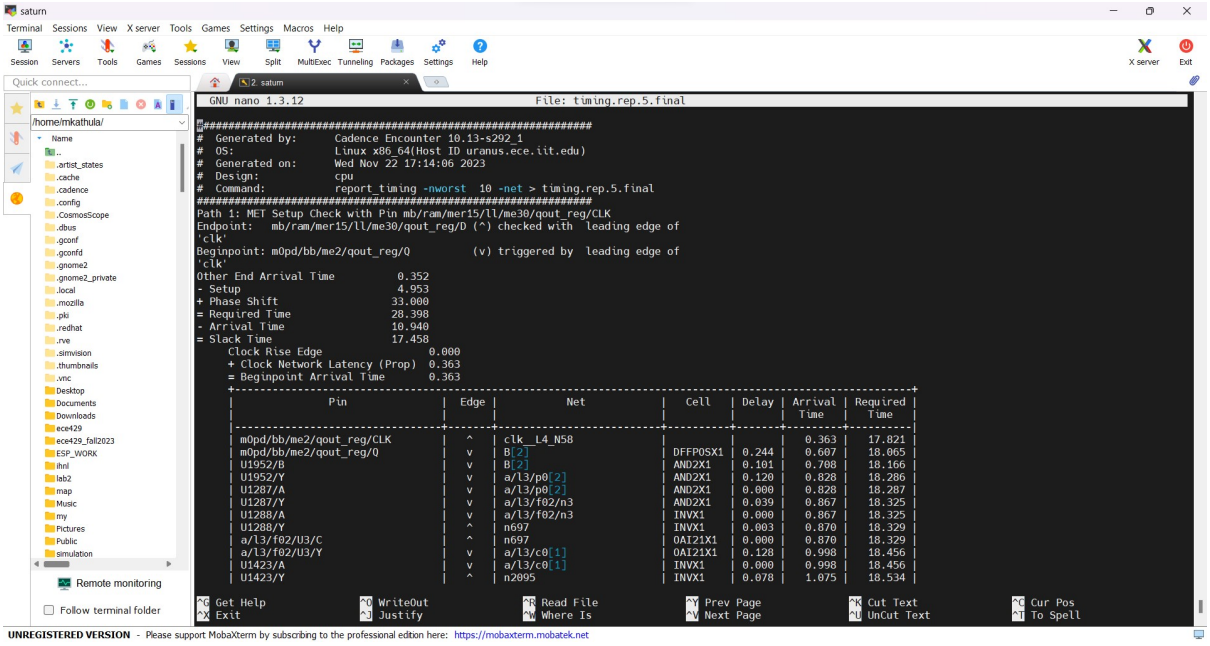


Fig. 21 : Timing.rep.5.final file for Maximum Clock Frquency

2.5 New Test Bench:

The source code of the test bench is given in the appendix. By the tb_test_cpu.v we verify using RTL simulation.

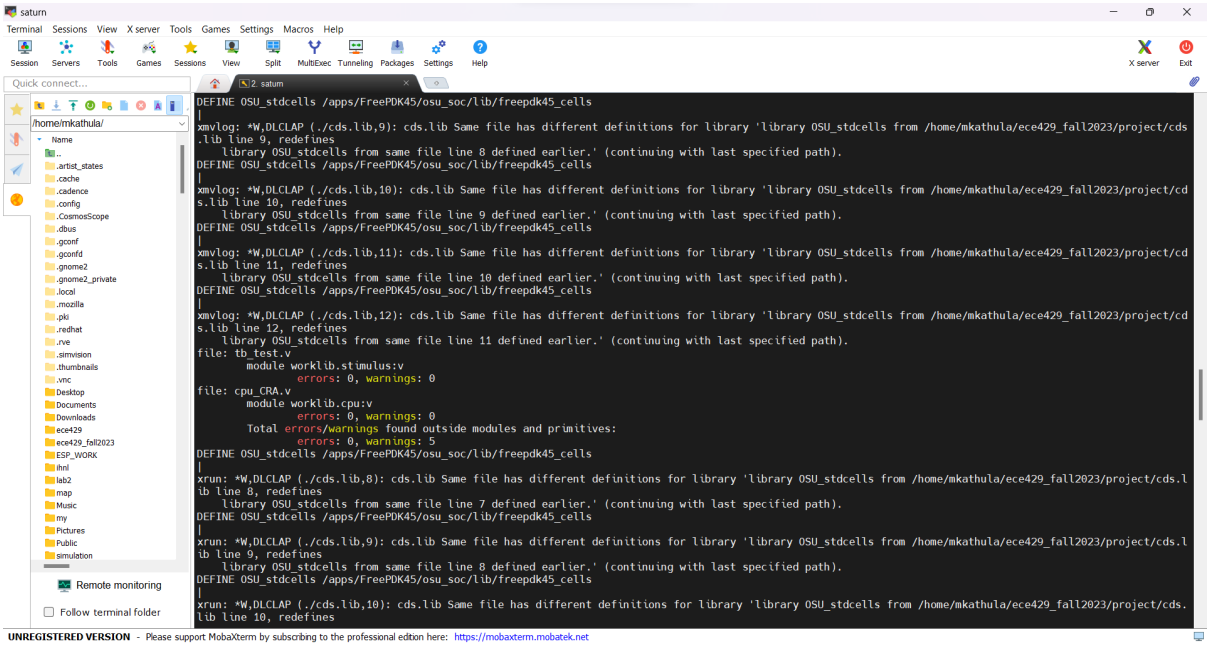


Fig. 22 : RTL Simulation for New Test Bench

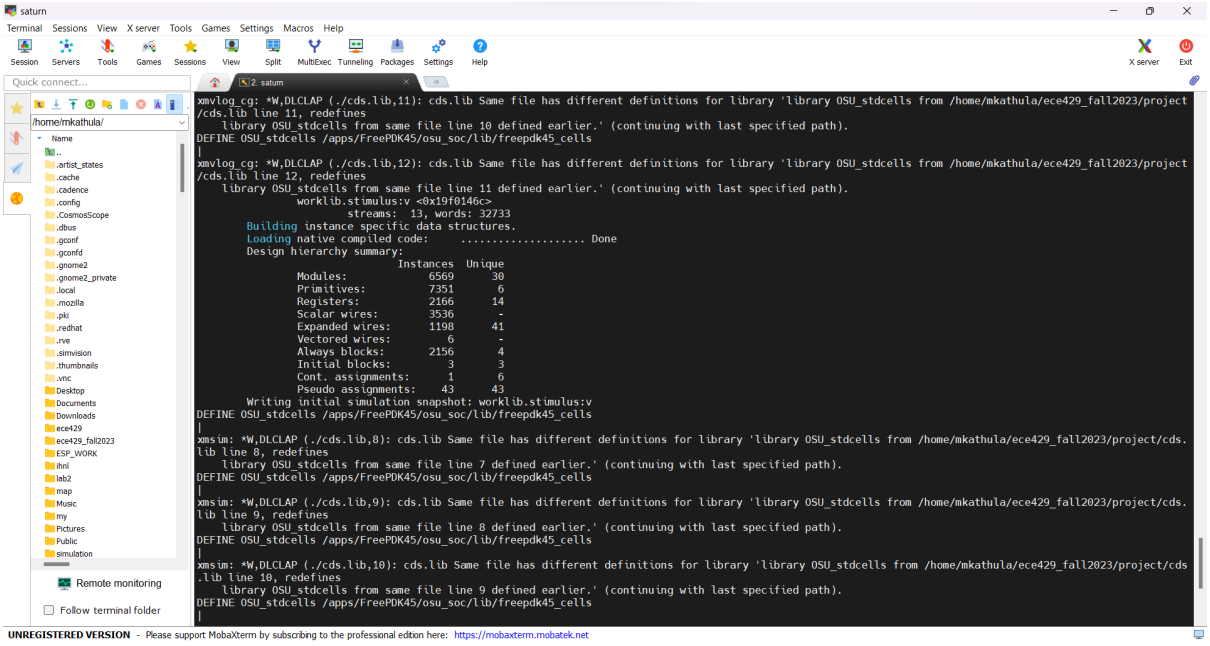


Fig.23 : RTL Simulation for New Test Bench

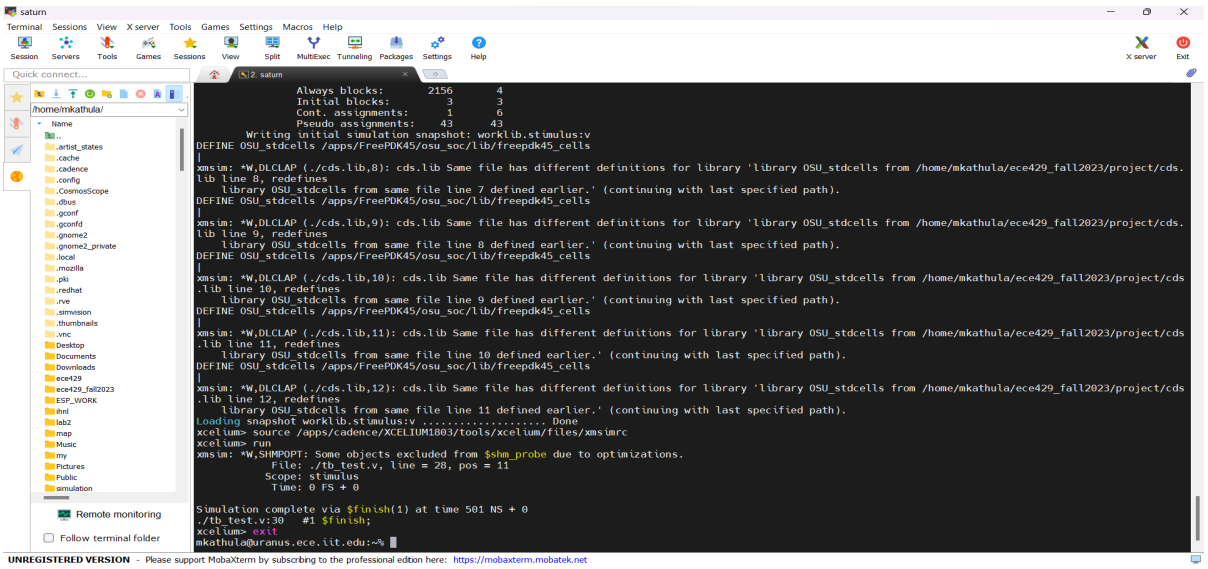


Fig. 24 : RTL Simulation for New Test Bench

By using simvision we find the waveforms.

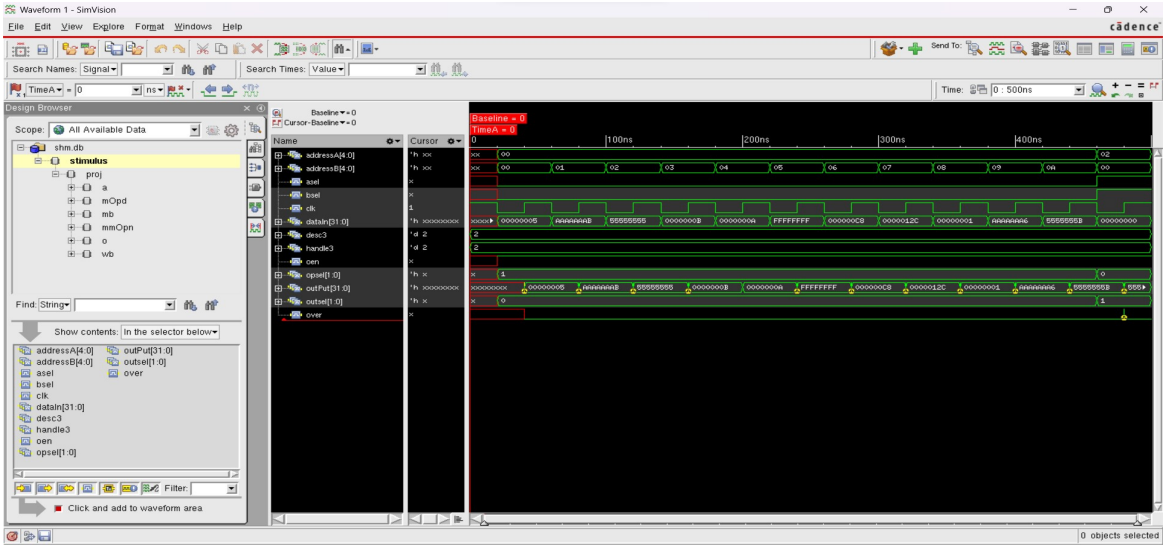


Fig. 25 : Simvision Waveform for New Test Bench

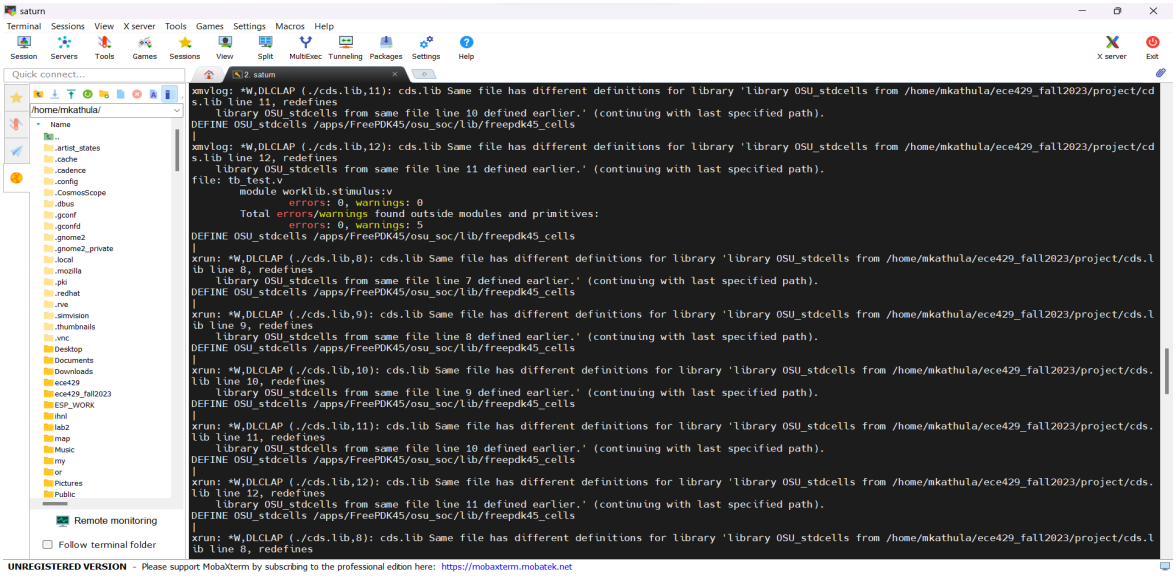


Fig. 26 :Logical and Post Synthesis Simulation

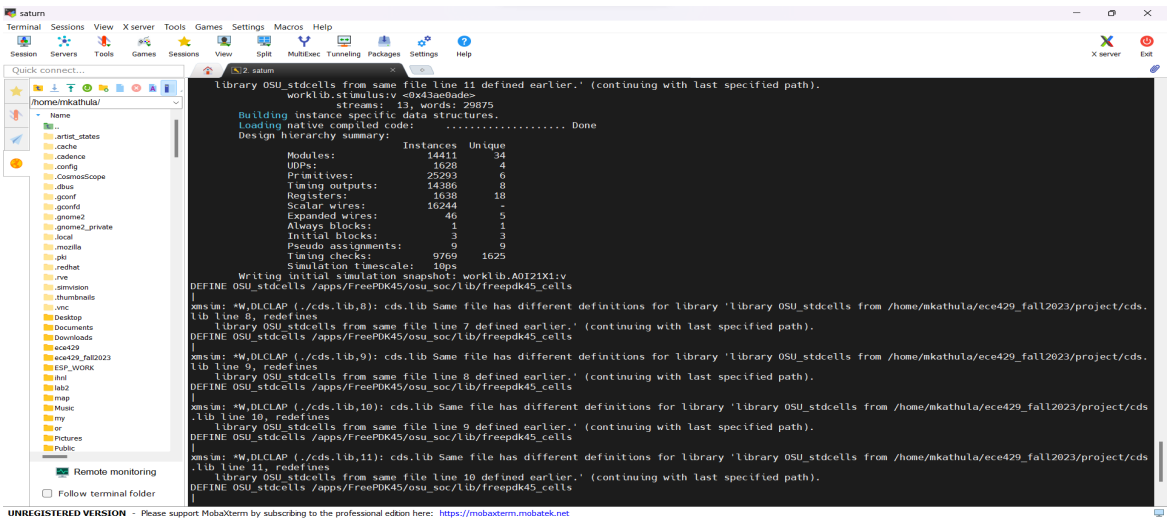


Fig. 27 :Logical and Post Synthesis Simulation

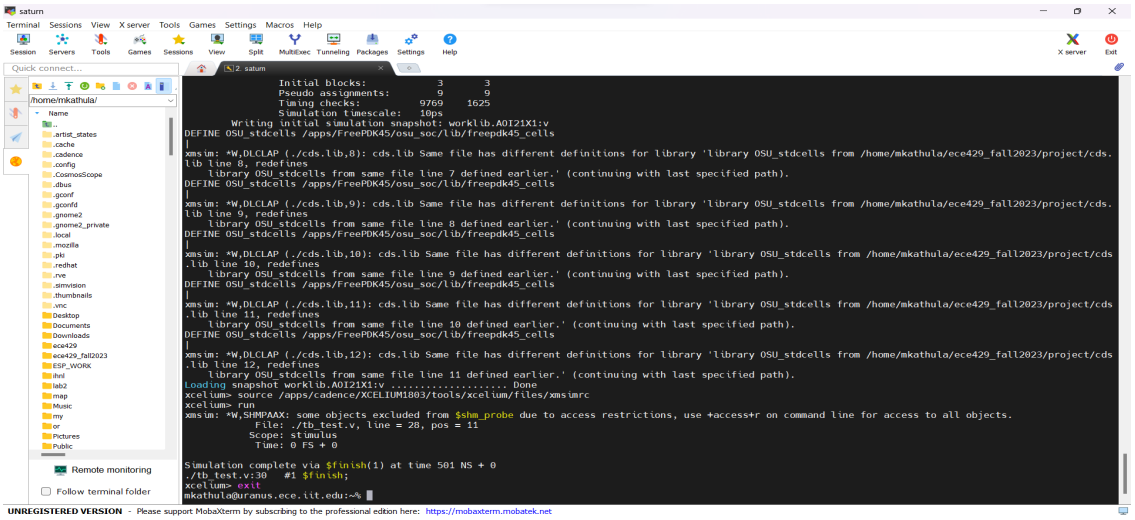


Fig. 28 :Logical and Post Synthesis Simulation

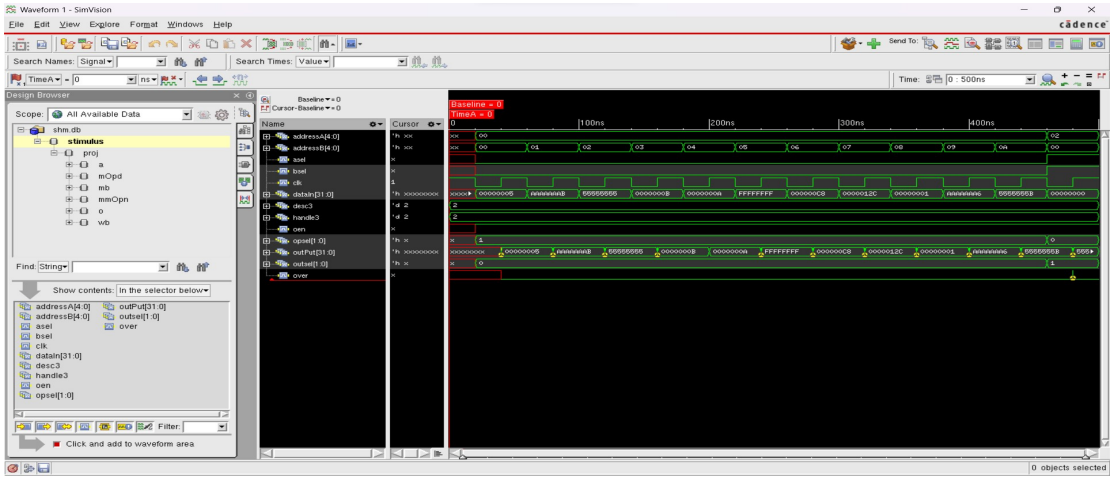


Fig. 29:Simvison waveform for Logical and Post Synthesis Simulation

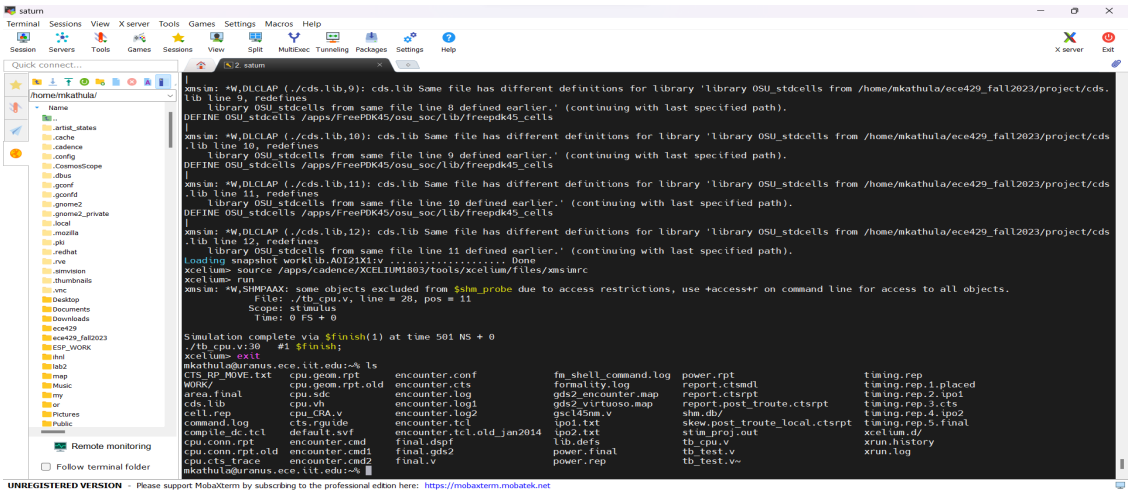


Fig. 30:Files created for CRA.

3. Carry Lookahead Adder:

For the more complex circuits, the addition operation will be difficult. For this Carry Look Ahead Adder is used, it dramatically reduces the addition performance time. The Carry Look Ahead Adder is obtained from the transformation of ripple carry adder.

The block Diagram of Carry Look Ahead Adder is as follows

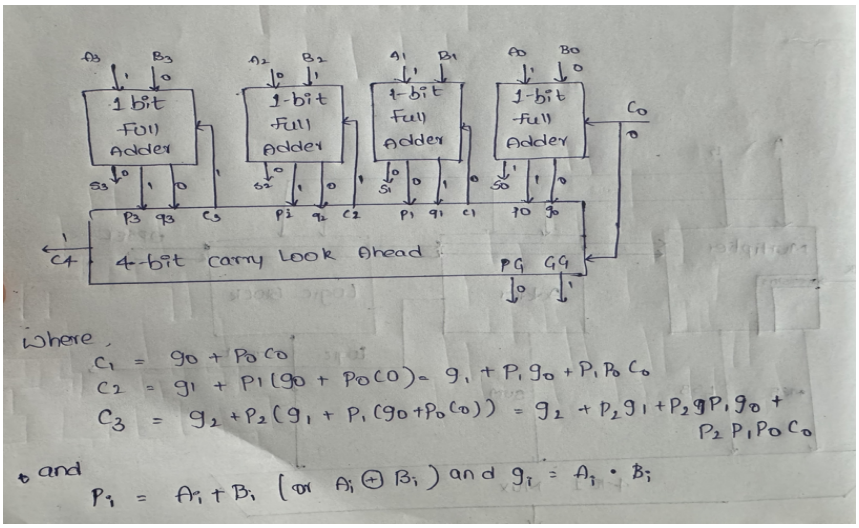


Fig. 31 : 4-bit Carry Look head Adder

3.1 RTL Simulation:

In this RTL simulation we will verify the logic without delays.

The RTL Simulation is done by using the cpu_cla.v and tb_cpu.v Verilog files.

It can be run by “xrun tb_cpu.v cpu_cla.v +access+r”.

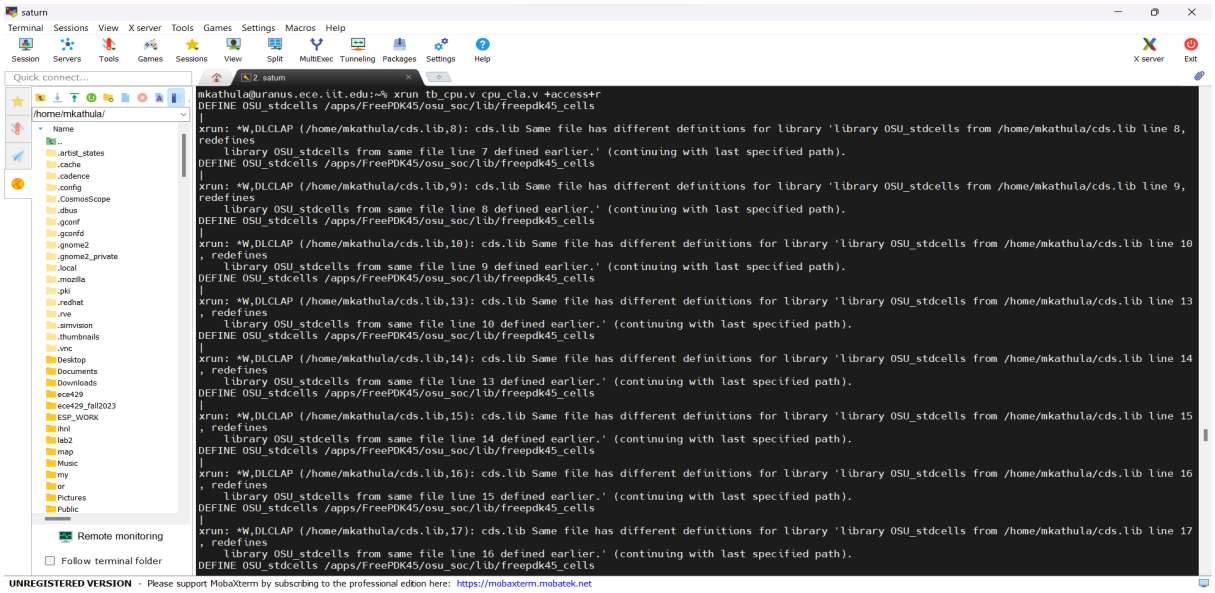


Fig. 32: CLA RTL Stimulation

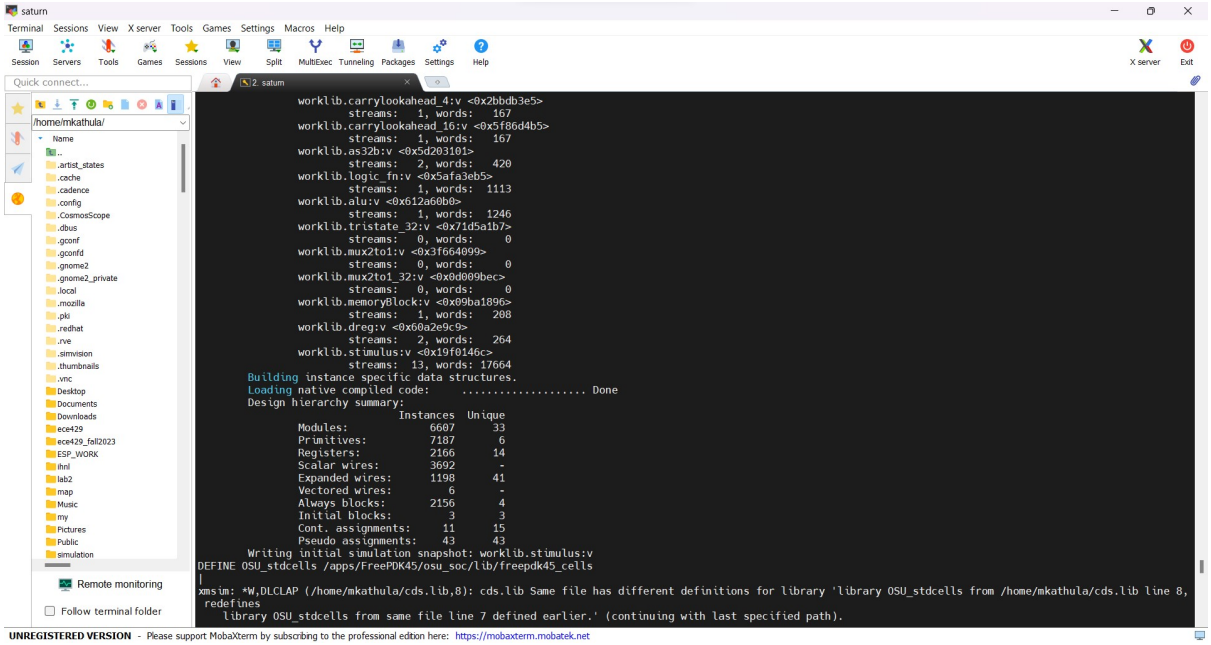


Fig. 33: CLA RTL Simulation

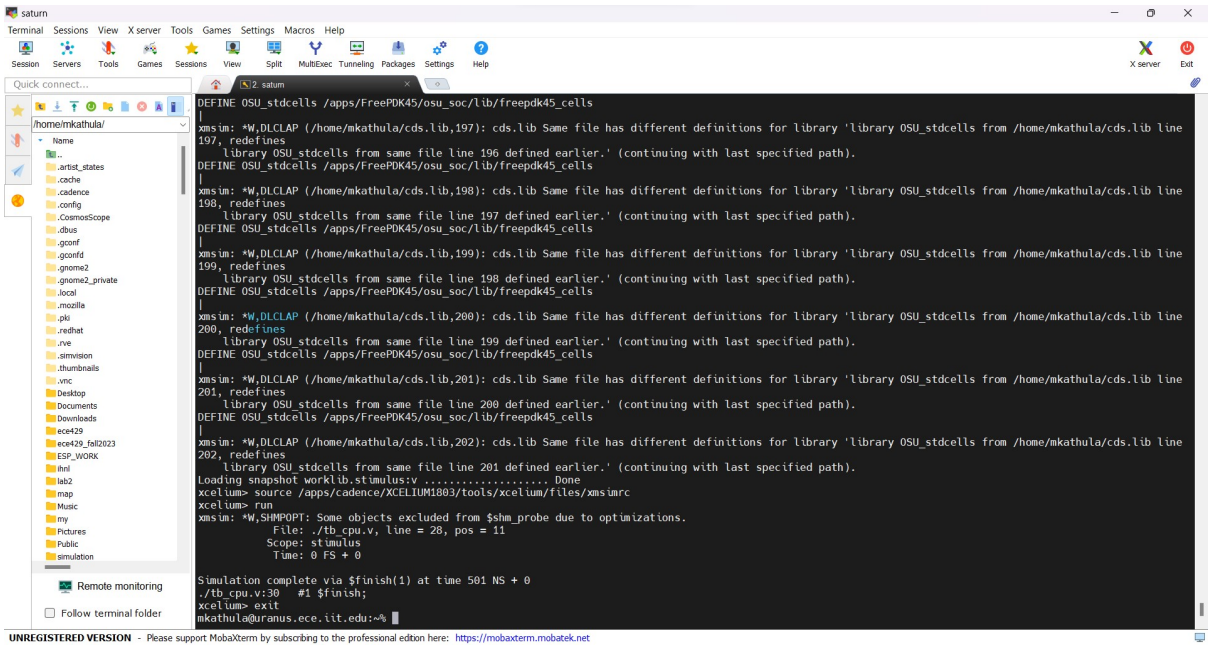


Fig. 34 : CLA RTL Simulation

There are few warnings but there were no errors associated with the Verilog files.

And the wave forms can be seen by using “simvision”.

When the simvison is opened

- Open database

- Click on shm.db
- Right click on shm.db and send to waveform window.
  The wave form can be as follows.

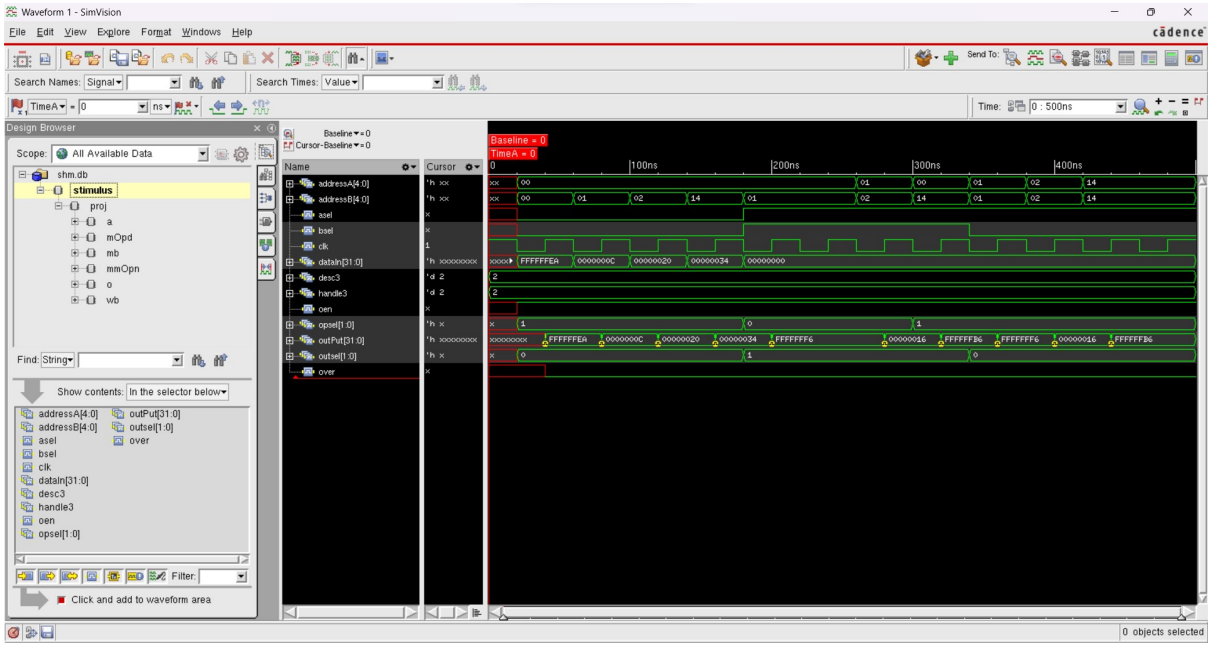


Fig. 35:Simvision Waveform

3.2 Logical Synthesis and Post-Synthesis Simulation:

For this we import the cadence library as
cd /import/scripts/FreePDK45_2011/FreePDK45/osu_soc/flow_ece429/* .

By this we will get the compiling files.
Here we need to change the file name and set my_toplevel to the respective filemename and module name as “[list cpu_CRA.v]” and “cpu”. The frequency must be 30MHz.
For the compile report we will run
“dc_shell -f compile_dc.tcl”
The cell report can be obtained by “nano cell.report”
It is as follows

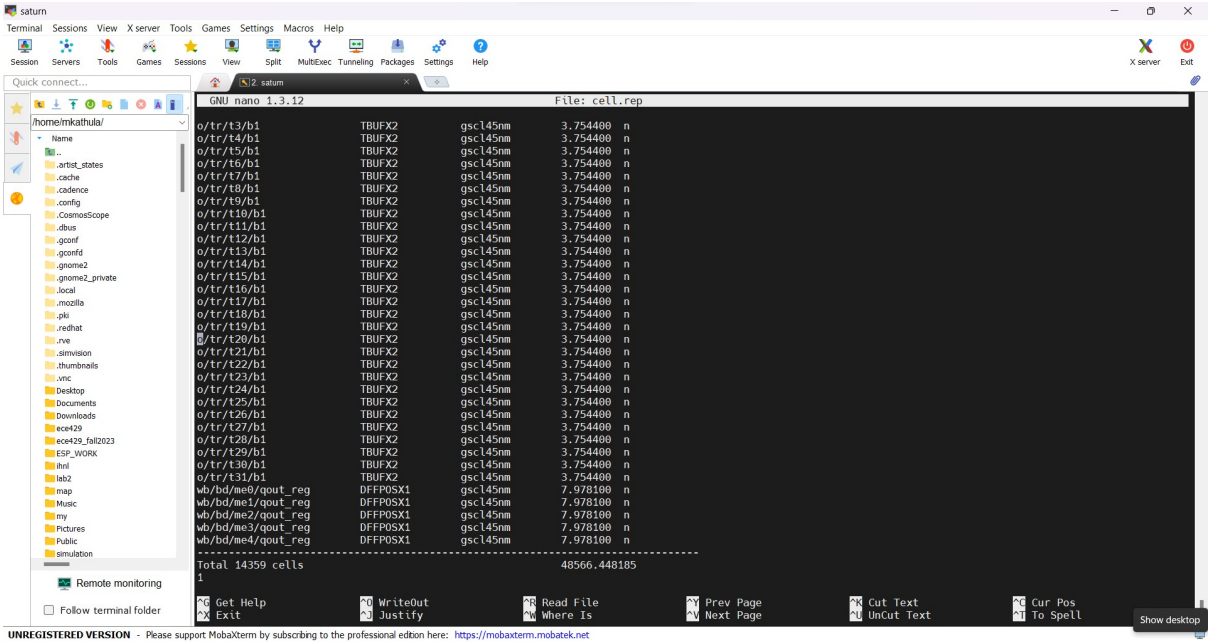


Fig. 36: Cell Report

The timing report can be obtained by “nano timing.rep”
And it is as follows

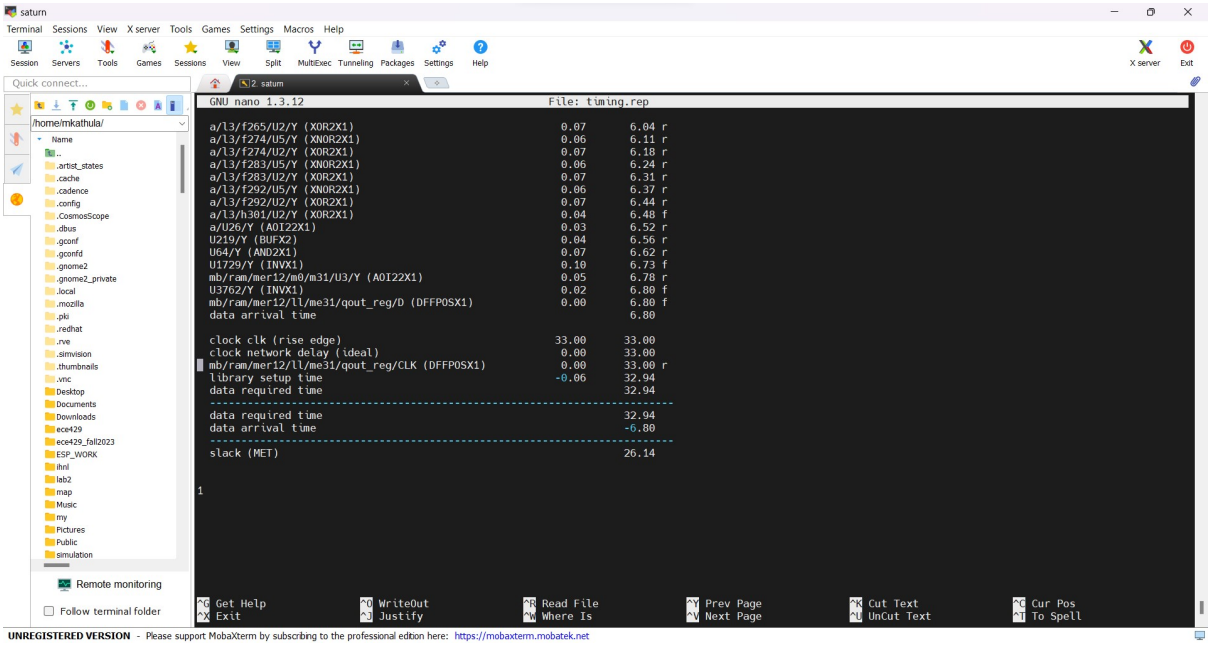


Fig. 37: Timing Report

The power report can be obtained by “nano power.rep”
It is as follows

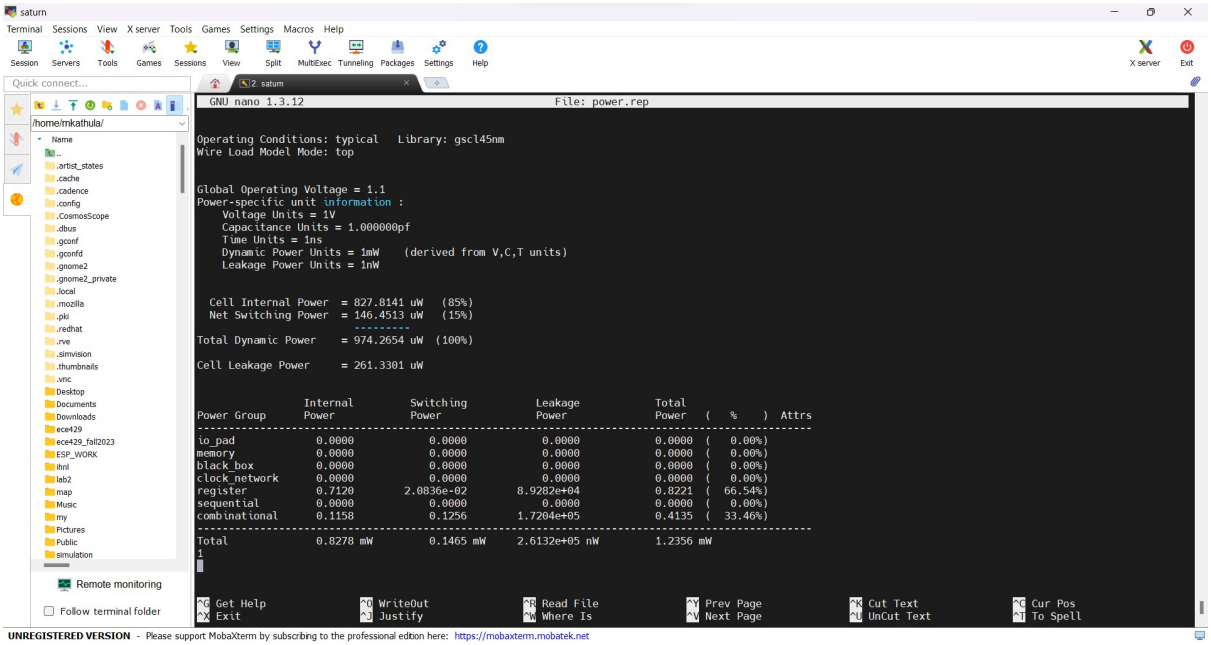


Fig. 38 : Power_Report

For the synthesis we will simulate with Verilog models of standard cells,i.e. gscl45nm.v, and we compare the result with rtl simulation.

This can be done by “xrun gscl45nm.v tb_cpu.v cpu.vh +access+r”

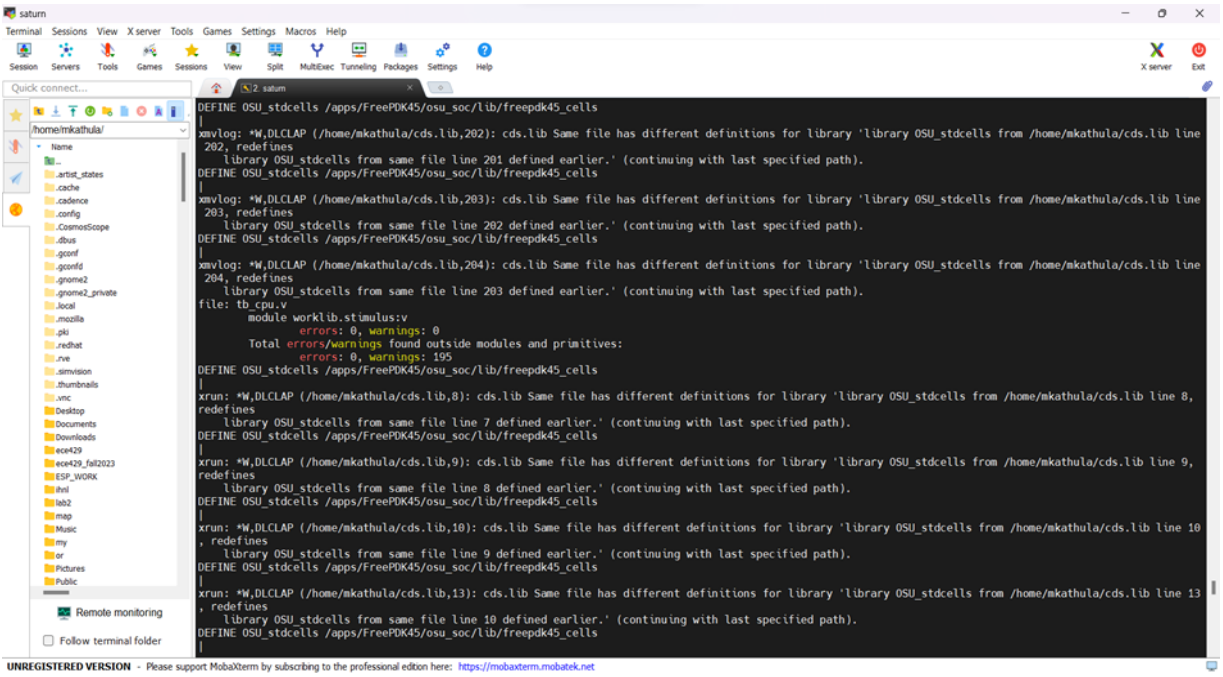
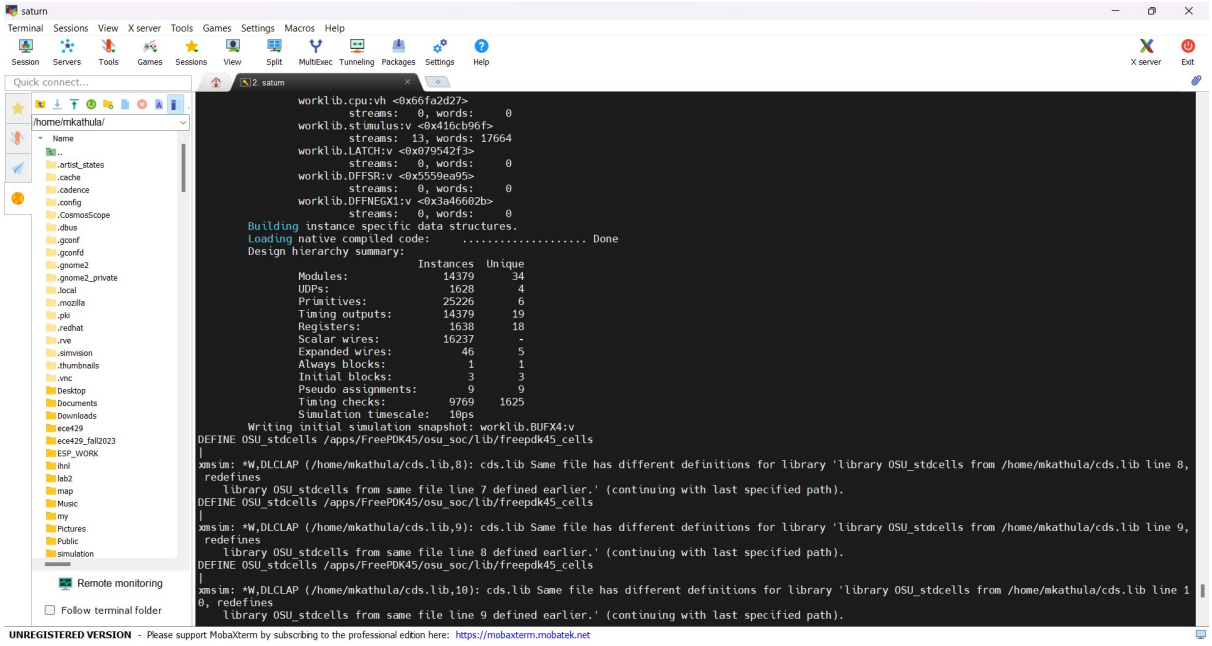
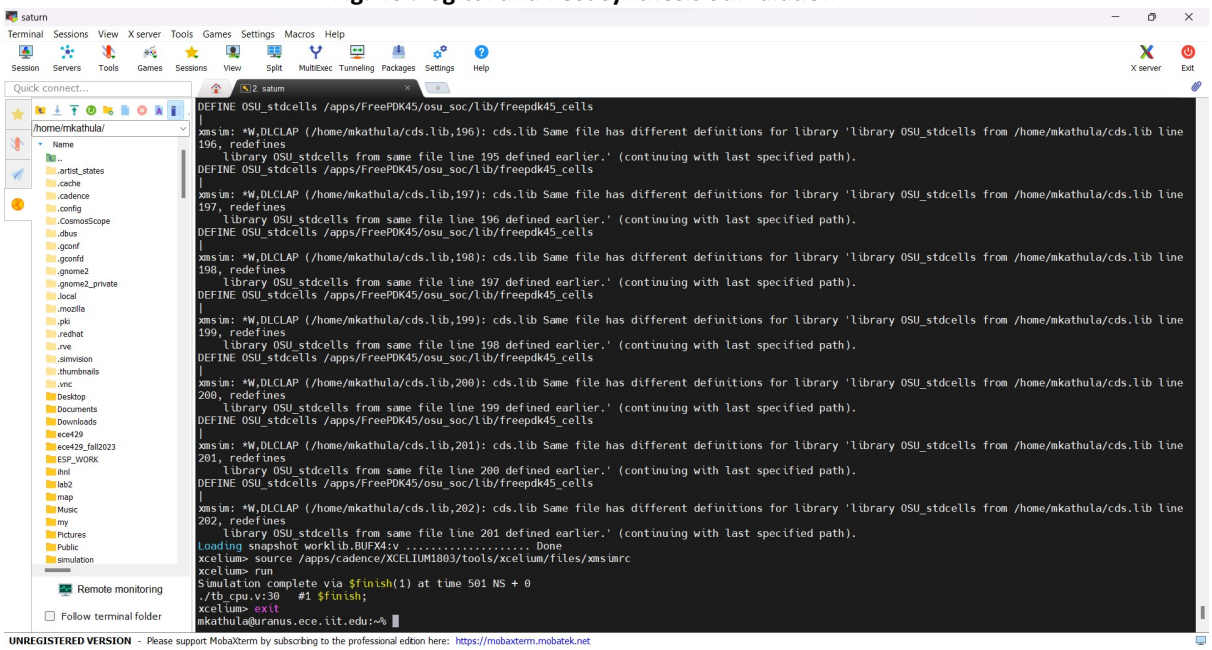


Fig. 39 :Logical and Post synthesis Stimulation



```
saturn
Terminal Sessions View X server Tools Games Settings Macros Help
Session Servers Tools Games Sessions View Split MultiExec Tunneling Packages Settings Help X server Exit
Quick connect...
2 saturn
/home/mkathula/
Name
.artist_states
.cache
.cadence
.config
.CosmosScope
.dbus
.gconf
.gconfd
.gnome2
.gnome2_private
.local
.mozilla
.pki
.redhat
.rve
.simvision
.thumbnails
.vnc
Desktop
Documents
Downloads
ece429
ece429_fall2023
ESP_WORK
ihnl
lab2
map
Music
my
Pictures
Public
simulation
Remote monitoring
Follow terminal folder

                worklib.cpu:vh <0x66fa2d27>
                        streams:   0, words:     0
                worklib.stimulus:v <0x416cb96f>
                        streams:  13, words: 17664
                worklib.LATCH:v <0x079542f3>
                        streams:   0, words:     0
                worklib.DFFSR:v <0x5559ea95>
                        streams:   0, words:     0
                worklib.DFFNEGX1:v <0x3a46602b>
                        streams:   0, words:     0
        Building instance specific data structures.
        Loading native compiled code:     .................... Done
        Design hierarchy summary:
                                  Instances  Unique
                Modules:              14379      34
                UDPs:                  1628       4
                Primitives:           25226       6
                Timing outputs:       14379      19
                Registers:             1638      18
                Scalar wires:         16237       -
                Expanded wires:          46       5
                Always blocks:            1       1
                Initial blocks:           3       3
                Pseudo assignments:       9       9
                Timing checks:         9769    1625
                Simulation timescale:   10ps
        Writing initial simulation snapshot: worklib.BUFX4:v
DEFINE OSU_stdcells /apps/FreePDK45/osu_soc/lib/freepdk45_cells
|
xmsim: *W,DLCLAP (/home/mkathula/cds.lib,8): cds.lib Same file has different definitions for library 'library OSU_stdcells from /home/mkathula/cds.lib line 8,
 redefines
    library OSU_stdcells from same file line 7 defined earlier.' (continuing with last specified path).
DEFINE OSU_stdcells /apps/FreePDK45/osu_soc/lib/freepdk45_cells
|
xmsim: *W,DLCLAP (/home/mkathula/cds.lib,9): cds.lib Same file has different definitions for library 'library OSU_stdcells from /home/mkathula/cds.lib line 9,
 redefines
    library OSU_stdcells from same file line 8 defined earlier.' (continuing with last specified path).
DEFINE OSU_stdcells /apps/FreePDK45/osu_soc/lib/freepdk45_cells
|
xmsim: *W,DLCLAP (/home/mkathula/cds.lib,10): cds.lib Same file has different definitions for library 'library OSU_stdcells from /home/mkathula/cds.lib line 1
0, redefines
    library OSU_stdcells from same file line 9 defined earlier.' (continuing with last specified path).
UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: https://mobaxterm.mobatek.net
```

Fig. 40 :Logical and Post synthesis Stimulation



```
saturn
Terminal Sessions View X server Tools Games Settings Macros Help
Session Servers Tools Games Sessions View Split MultiExec Tunneling Packages Settings Help X server Exit
Quick connect...
2 saturn
/home/mkathula/
Name
.artist_states
.cache
.cadence
.config
.CosmosScope
.dbus
.gconf
.gconfd
.gnome2
.gnome2_private
.local
.mozilla
.pki
.redhat
.rve
.simvision
.thumbnails
.vnc
Desktop
Documents
Downloads
ece429
ece429_fall2023
ESP_WORK
ihnl
lab2
map
Music
my
Pictures
Public
simulation
Remote monitoring
Follow terminal folder

DEFINE OSU_stdcells /apps/FreePDK45/osu_soc/lib/freepdk45_cells
|
xmsim: *W,DLCLAP (/home/mkathula/cds.lib,196): cds.lib Same file has different definitions for library 'library OSU_stdcells from /home/mkathula/cds.lib line
196, redefines
    library OSU_stdcells from same file line 195 defined earlier.' (continuing with last specified path).
DEFINE OSU_stdcells /apps/FreePDK45/osu_soc/lib/freepdk45_cells
|
xmsim: *W,DLCLAP (/home/mkathula/cds.lib,197): cds.lib Same file has different definitions for library 'library OSU_stdcells from /home/mkathula/cds.lib line
197, redefines
    library OSU_stdcells from same file line 196 defined earlier.' (continuing with last specified path).
DEFINE OSU_stdcells /apps/FreePDK45/osu_soc/lib/freepdk45_cells
|
xmsim: *W,DLCLAP (/home/mkathula/cds.lib,198): cds.lib Same file has different definitions for library 'library OSU_stdcells from /home/mkathula/cds.lib line
198, redefines
    library OSU_stdcells from same file line 197 defined earlier.' (continuing with last specified path).
DEFINE OSU_stdcells /apps/FreePDK45/osu_soc/lib/freepdk45_cells
|
xmsim: *W,DLCLAP (/home/mkathula/cds.lib,199): cds.lib Same file has different definitions for library 'library OSU_stdcells from /home/mkathula/cds.lib line
199, redefines
    library OSU_stdcells from same file line 198 defined earlier.' (continuing with last specified path).
DEFINE OSU_stdcells /apps/FreePDK45/osu_soc/lib/freepdk45_cells
|
xmsim: *W,DLCLAP (/home/mkathula/cds.lib,200): cds.lib Same file has different definitions for library 'library OSU_stdcells from /home/mkathula/cds.lib line
200, redefines
    library OSU_stdcells from same file line 199 defined earlier.' (continuing with last specified path).
DEFINE OSU_stdcells /apps/FreePDK45/osu_soc/lib/freepdk45_cells
|
xmsim: *W,DLCLAP (/home/mkathula/cds.lib,201): cds.lib Same file has different definitions for library 'library OSU_stdcells from /home/mkathula/cds.lib line
201, redefines
    library OSU_stdcells from same file line 200 defined earlier.' (continuing with last specified path).
DEFINE OSU_stdcells /apps/FreePDK45/osu_soc/lib/freepdk45_cells
|
xmsim: *W,DLCLAP (/home/mkathula/cds.lib,202): cds.lib Same file has different definitions for library 'library OSU_stdcells from /home/mkathula/cds.lib line
202, redefines
    library OSU_stdcells from same file line 201 defined earlier.' (continuing with last specified path).
Loading snapshot worklib.BUFX4:v ..................... Done
xcelium> source /apps/cadence/XCELIUM1803/tools/xcelium/files/xmsimrc
xcelium> run
Simulation complete via $finish(1) at time 501 NS + 0
./tb_cpu.v:30   #1 $finish;
xcelium> exit
mkathula@uranus.ece.iit.edu:~%
UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: https://mobaxterm.mobatek.net
```

Fig. 41 :Logical and Post synthesis Stimulation

It shows same as the RTL Simulation.

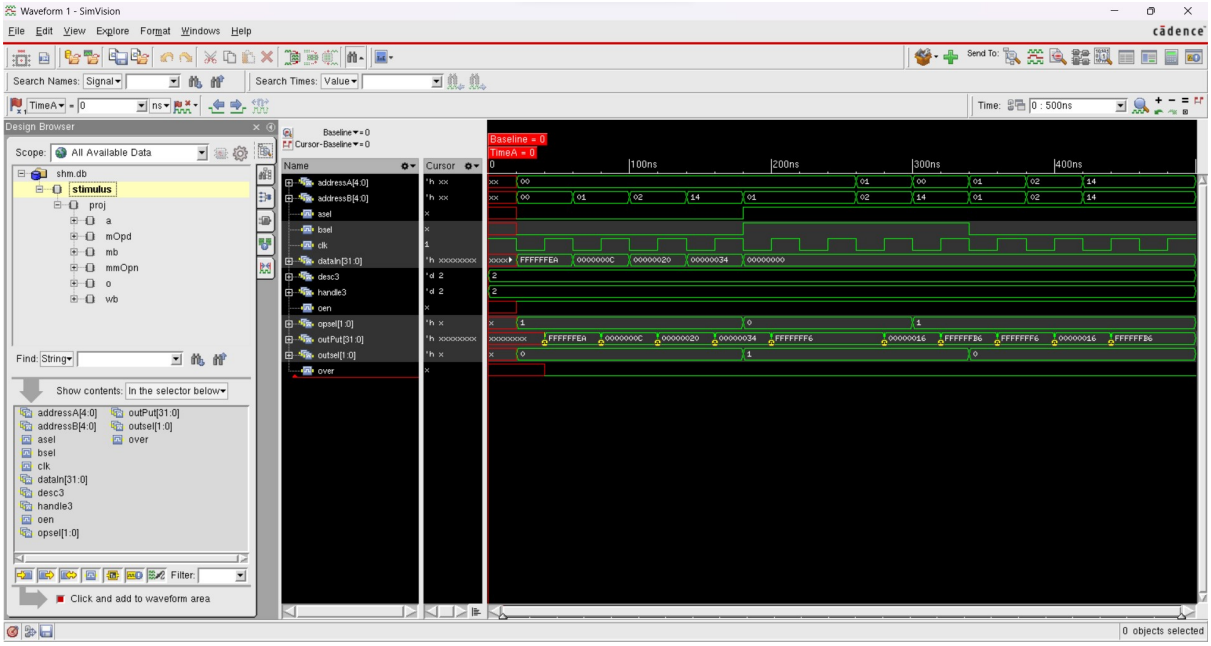


Fig. 42: Simvision for Logical and Post Synthesis Stimulation

3.3 Place & Route and Post-P&R Simulation:

After the synthesis we will change the set my_toplevel to “cpu” in “encounter.conf”.

Now for the encounter results we run the

“encounter -init encounter.tcl”

For Gate Area we will run “reportGateCount -limit 0”.

It can be as follows

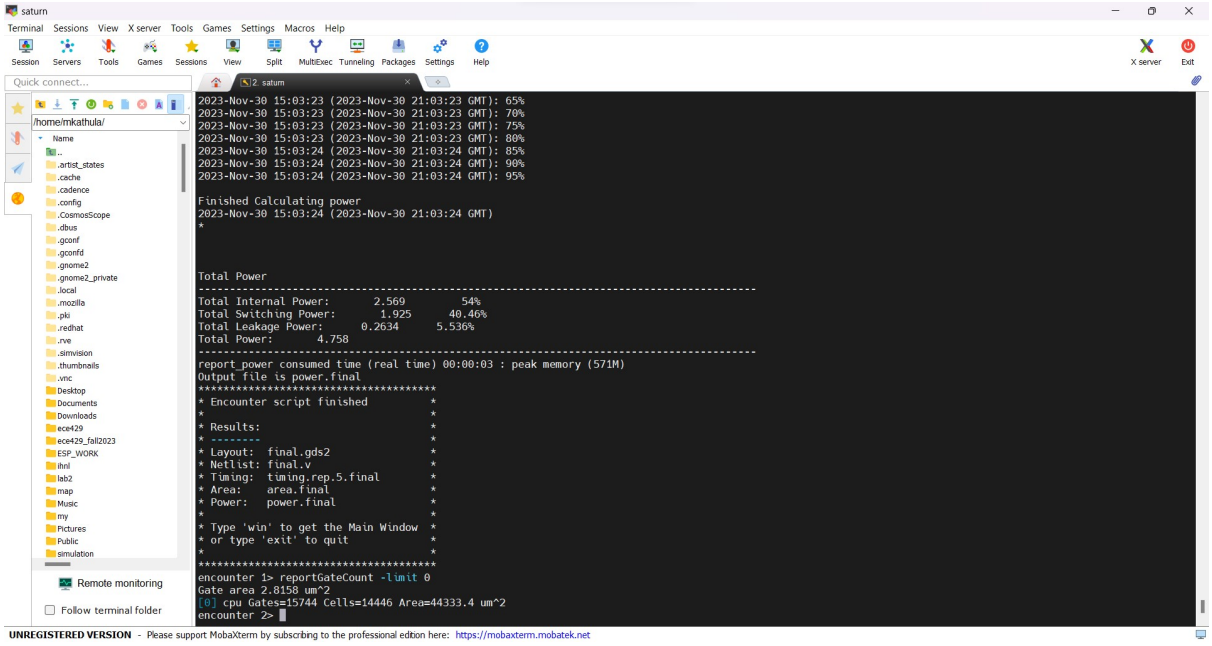


Fig. 43 : Gate Area

The power can be obtained by “report_power”.

It can be as follows

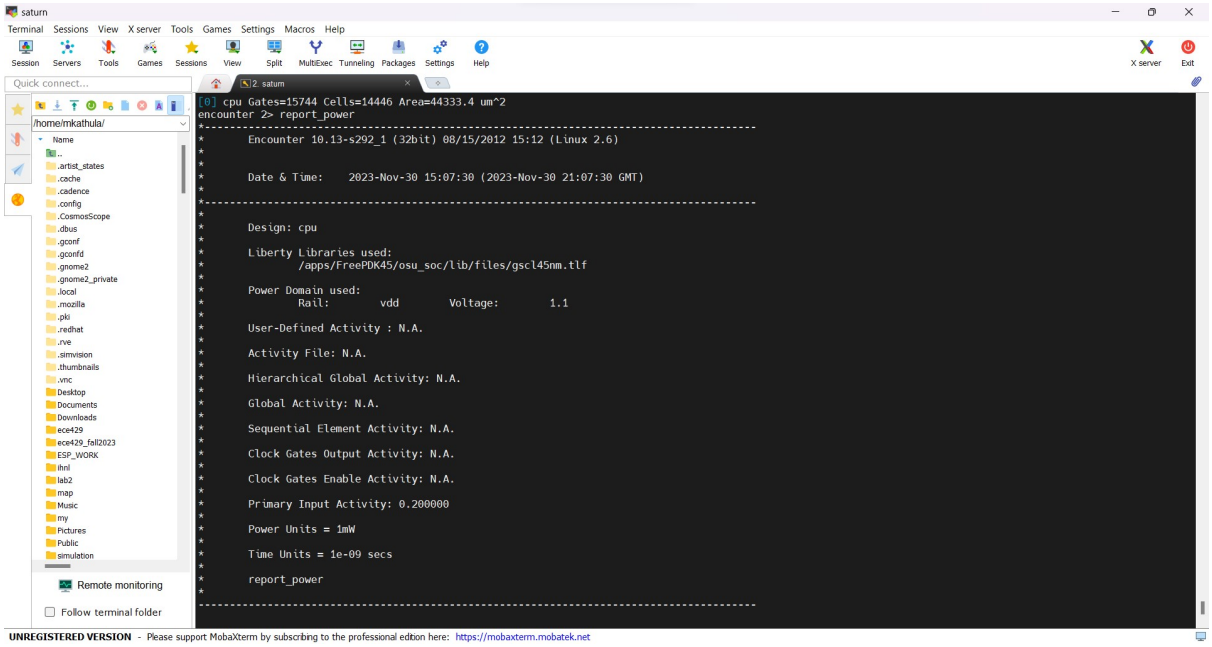


Fig. 44 : Report_Power

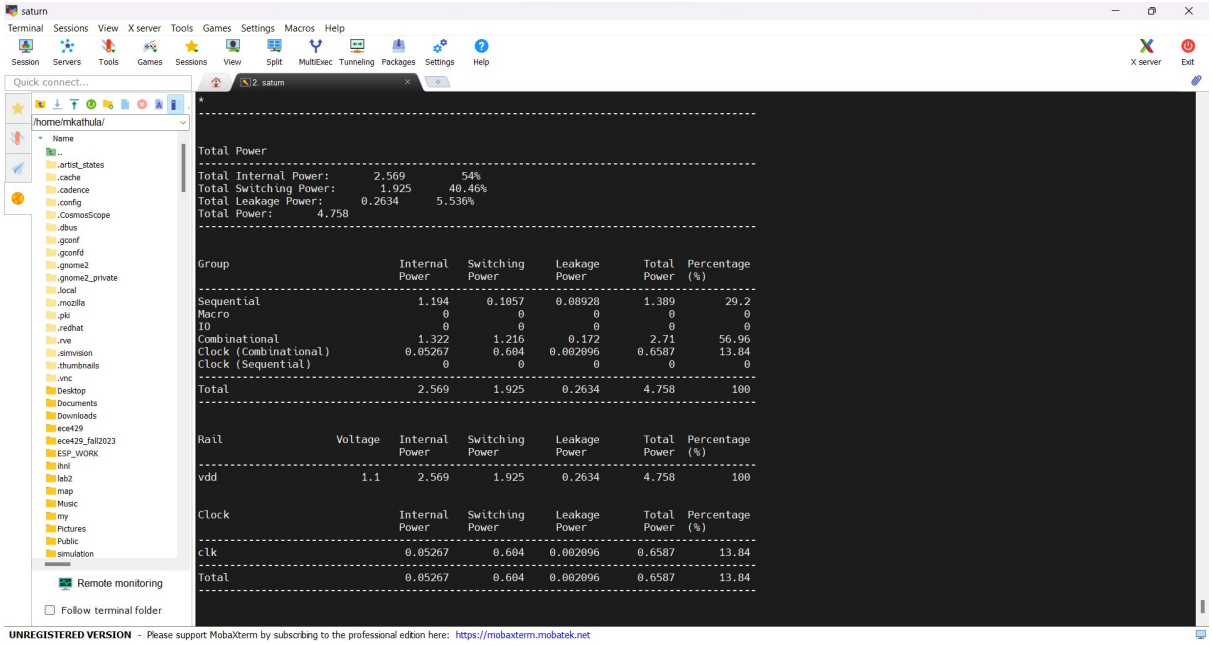


Fig.45 : Report_Power

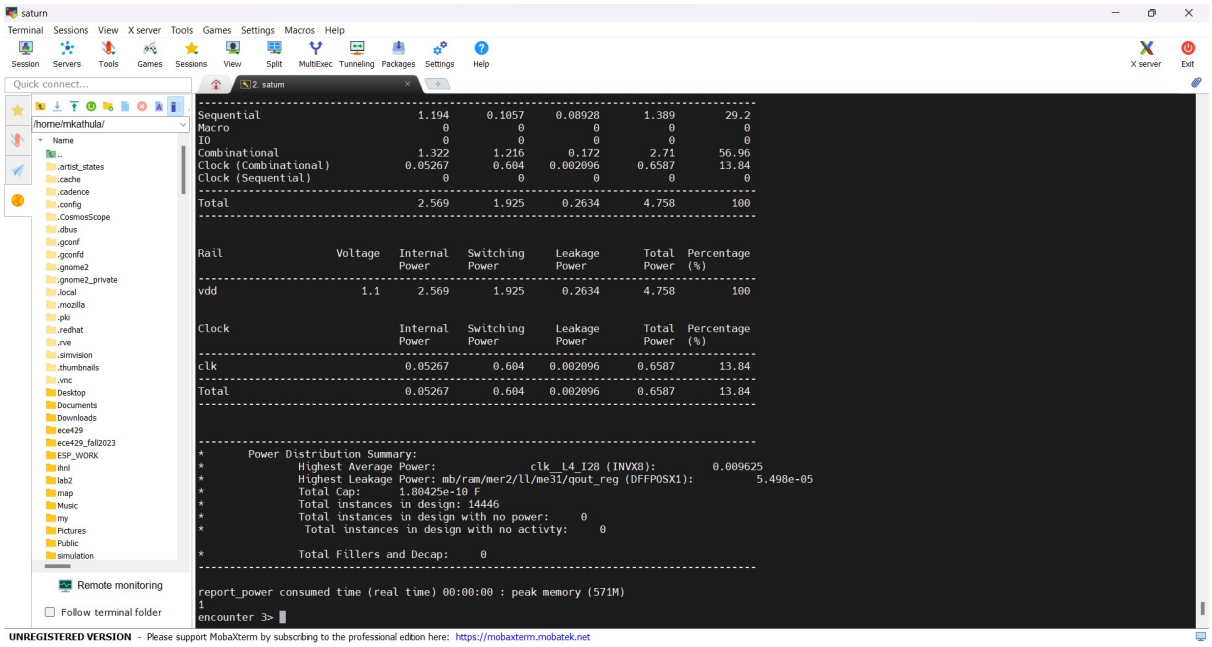


Fig.46 : Report_Power

For Place and Route and Post-P&R Simulation we run “xrun gscl45nm.v tb_cpu.v final.v +access+r”

It is as follows:

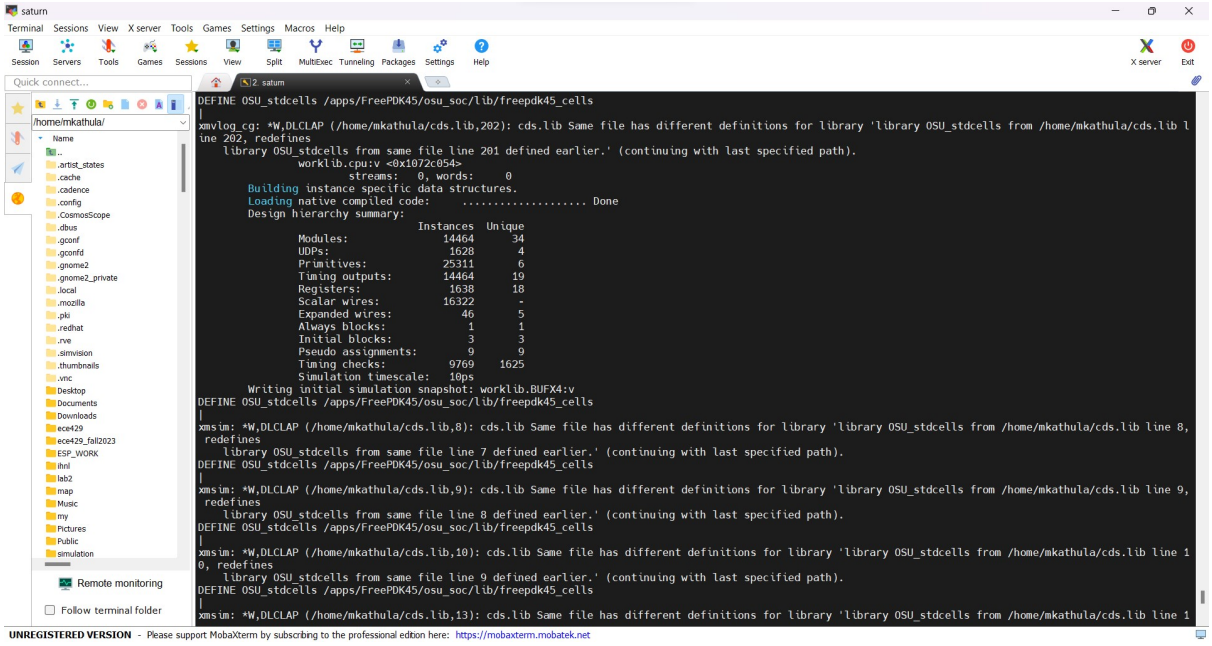


Fig. 47 : Place & Route and Post-P&R Simulation

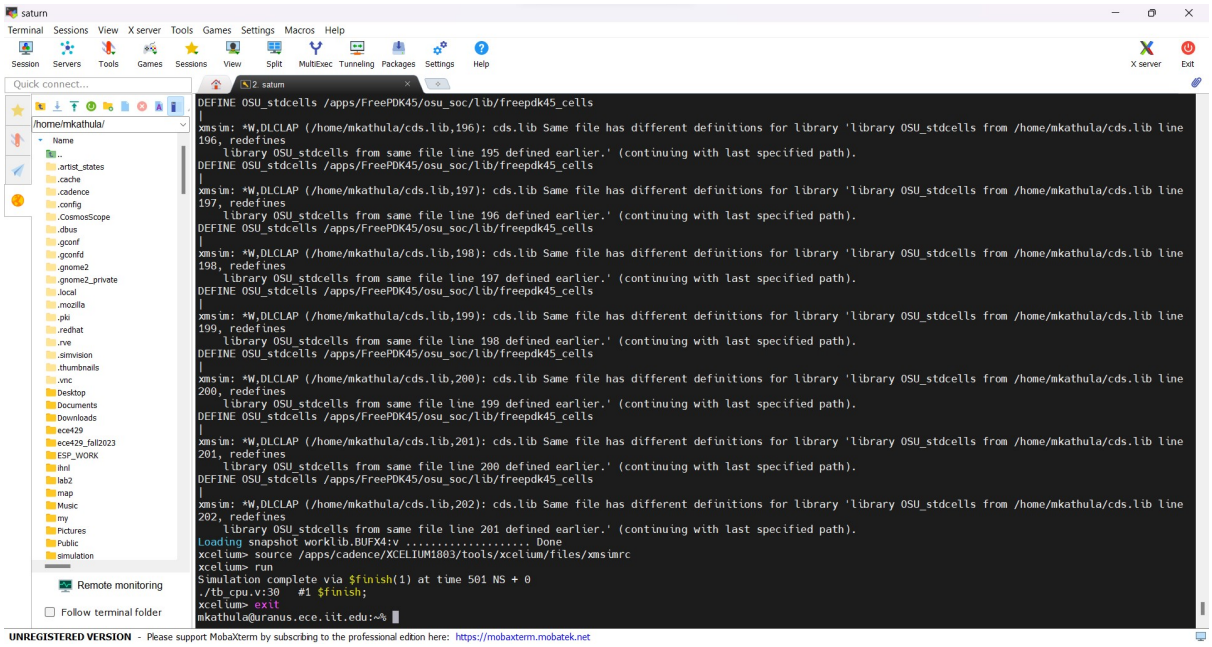


Fig. 48 : Place & Route and Post-P&R Simulation

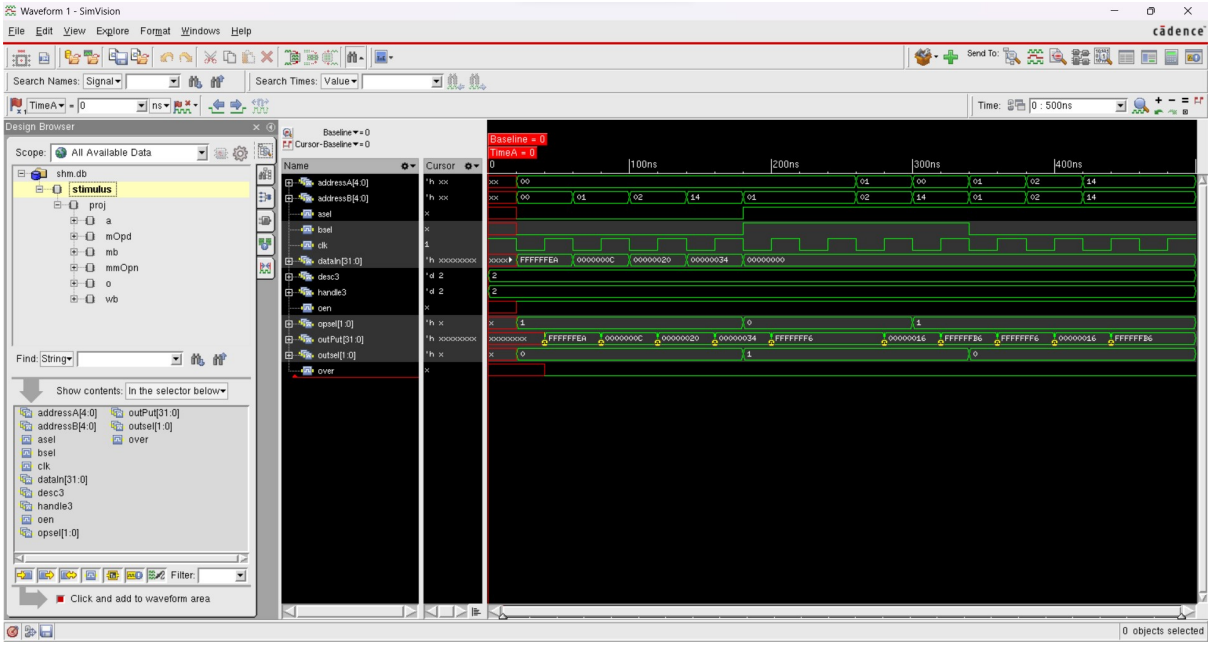


Fig. 49 :Simvision for Place & Route and Post-P&R Simulation

3.4 Maximum Clock Frequency:

This can be obtained in timing.rep.5.final.

Here the slack time has decreased to 17.921, Based on the required time of 28.846, max clock frequency is 34.67Mhz(1/28.864).

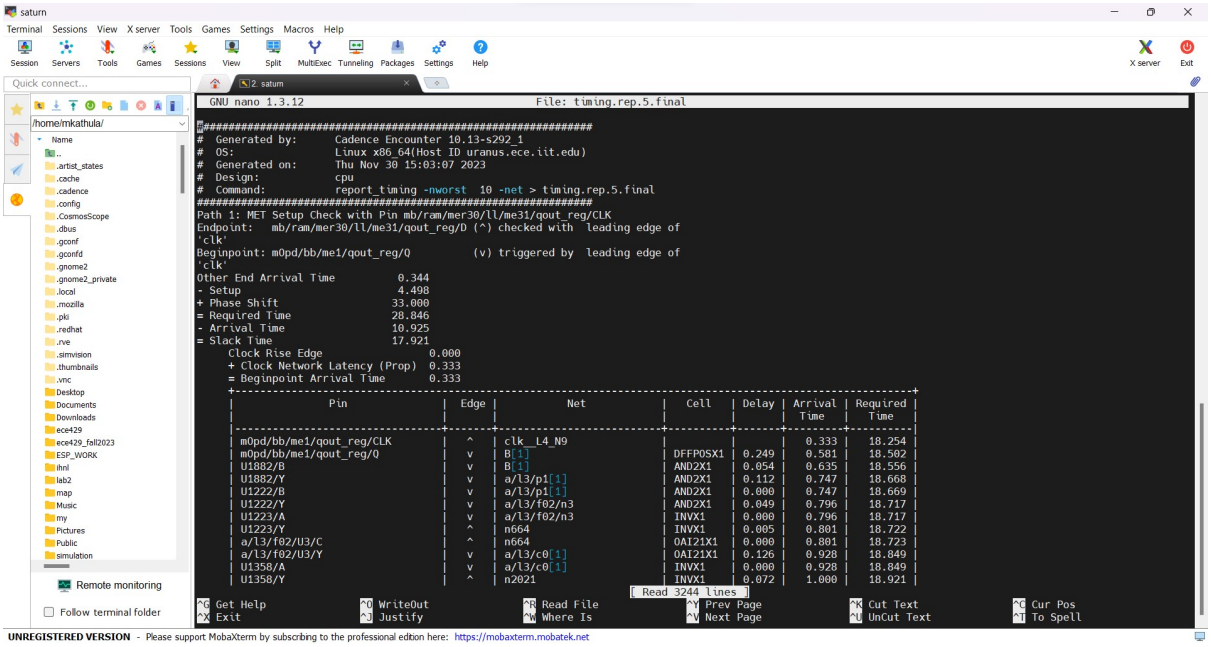


Fig. 50 : Timing.rep.5.final file for Maximum Clock Frquency

3.5 New Test Bench:

The source code of the test bench is given in the appendix. By the tb_test_cpu.v we verify using RTL simulation.

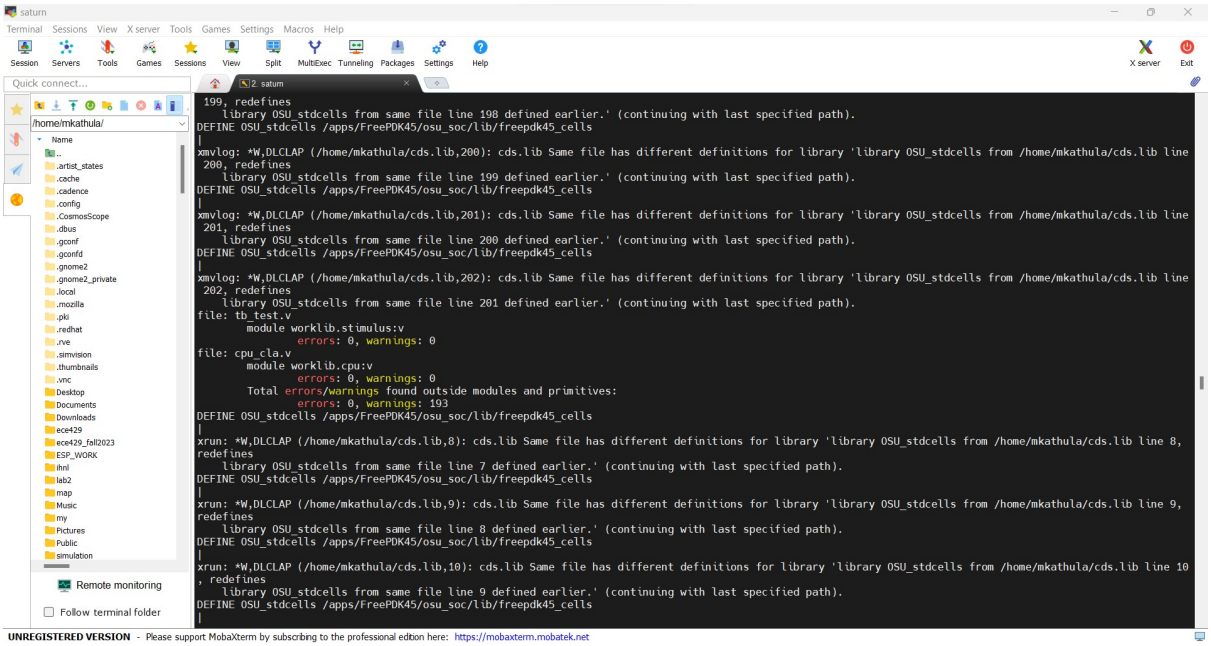


Fig. 51: RTL Simulation for New Test Bench

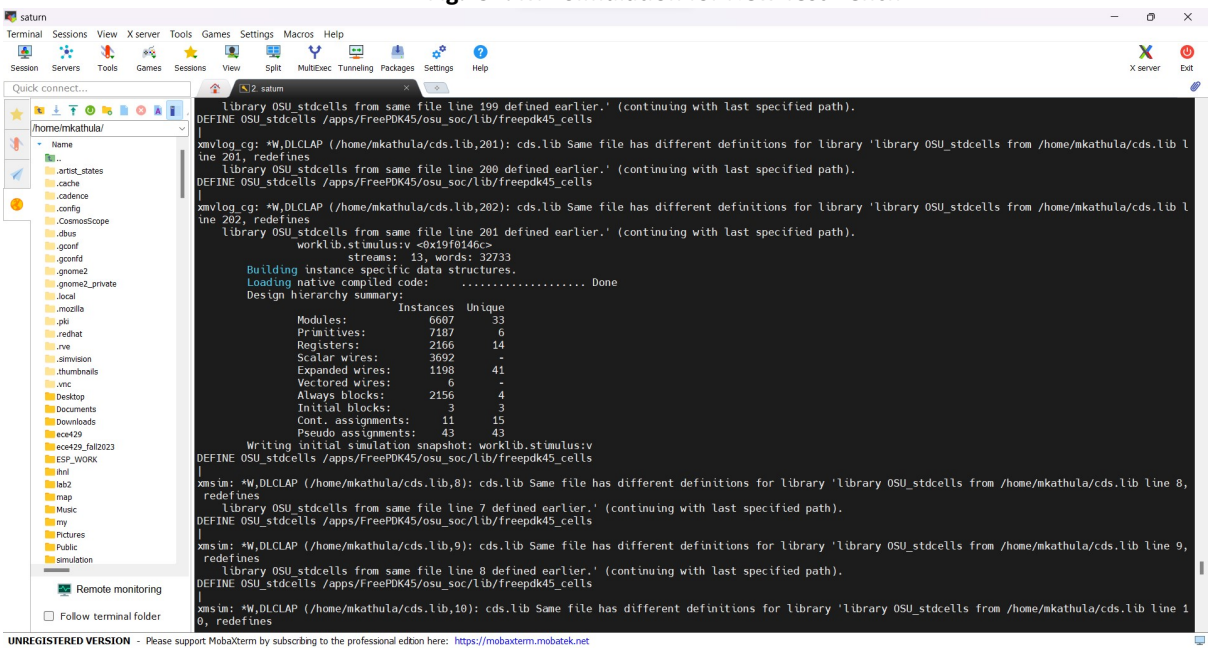


Fig.52 : RTL Simulation for New Test Bench

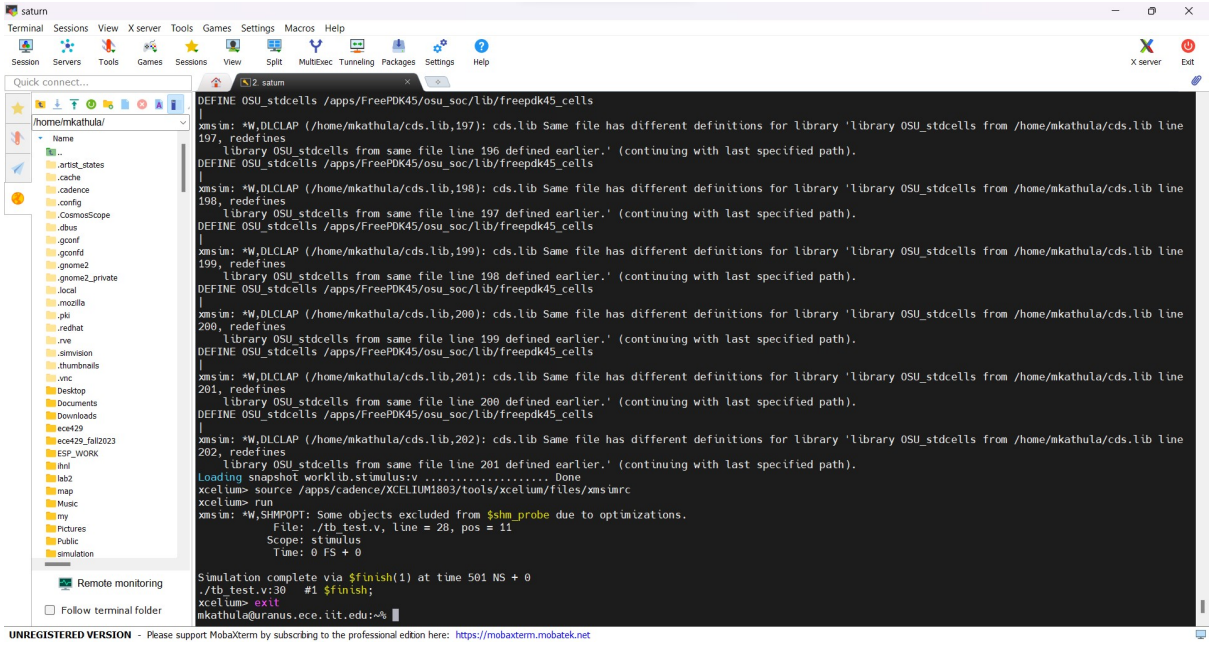


Fig. 53 : RTL Simulation for New Test Bench

By using the simvision we can get the waveforms of the new test bench.

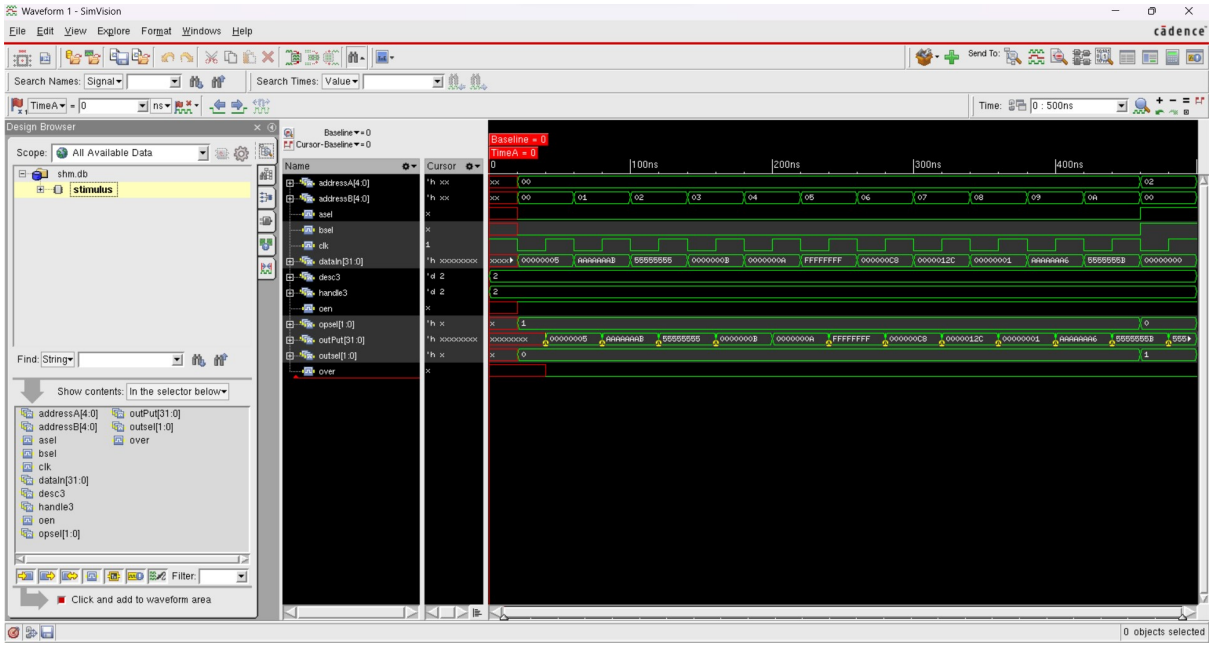


Fig. 54 : Simvision Waveform for New Test Bench

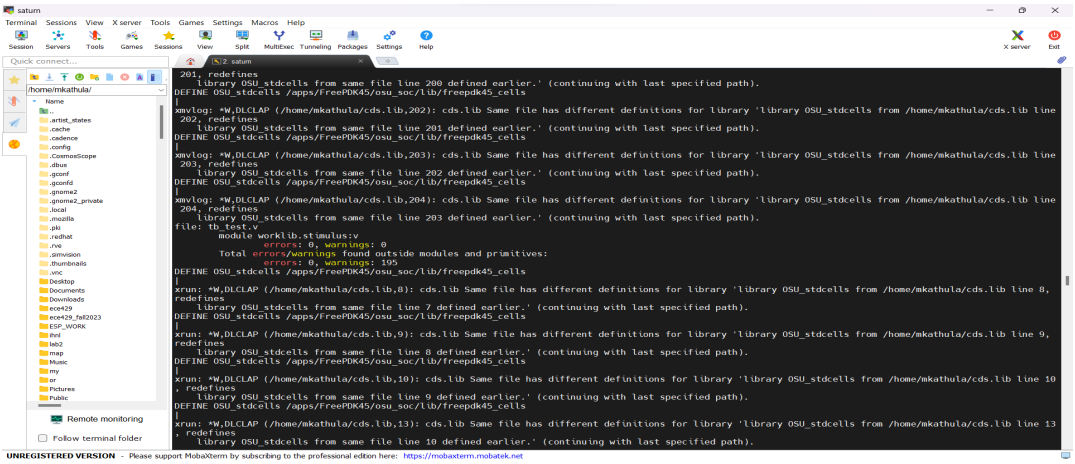


Fig. 55: Logical and Post Synthesis Simulation for new test bench

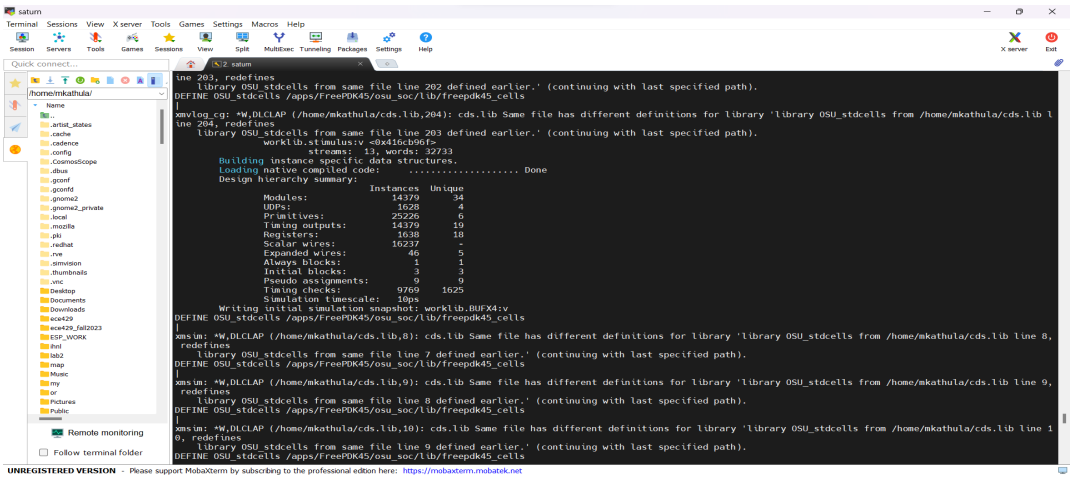


Fig.56:Logical and Post Synthesis Simulation for new test bench

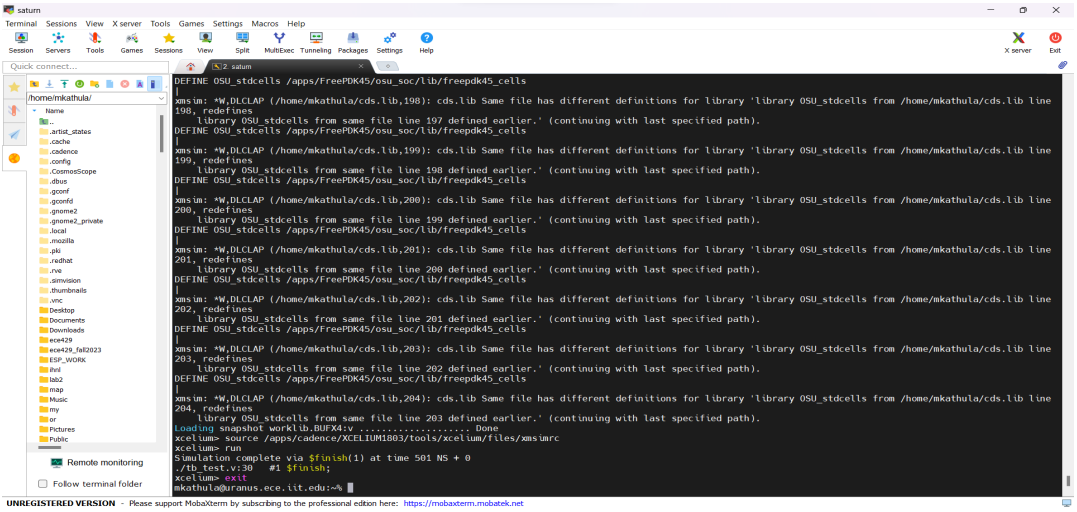


Fig.57:Logical and Post Synthesis Simulation for new test bench

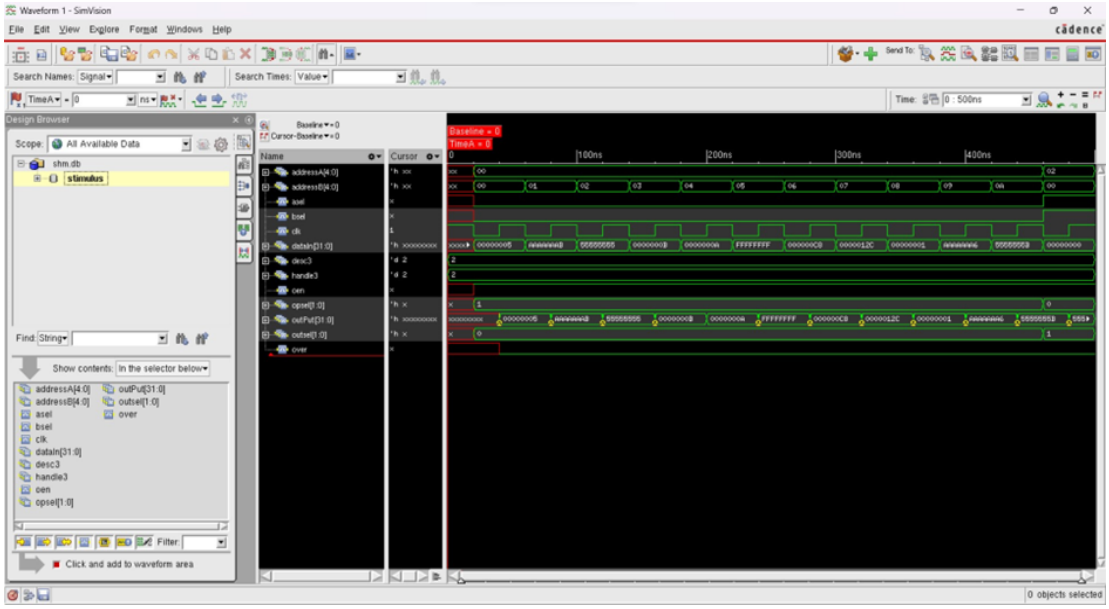


Fig. 58: Simvision waveform for new test bench(Logical and Post Synthesis Simulation)

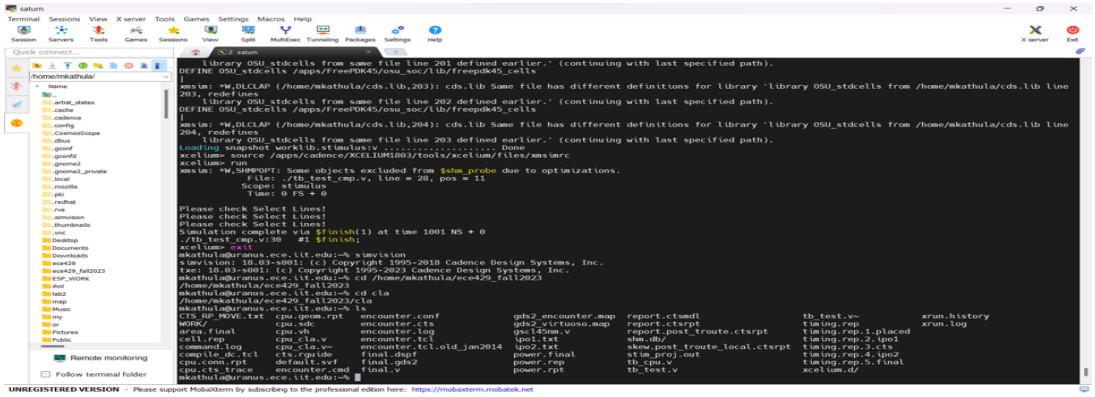


Fig.59 :Files Created for CLA

4. Carry Skip Adder:

The Carry Skip Adder is one of the adder implementation which improves the delay of the ripple carry adder. It decreases the Delay of the Ripple Carry Adder with simple effort as compared to other adders.

The block diagram of Carry Skip Adder is as follows

Here, the operands of CSA is divided into r bit blocks. In which, the each block a ripple carry adder is utilized to produce the sum bits and carryout bits for the block

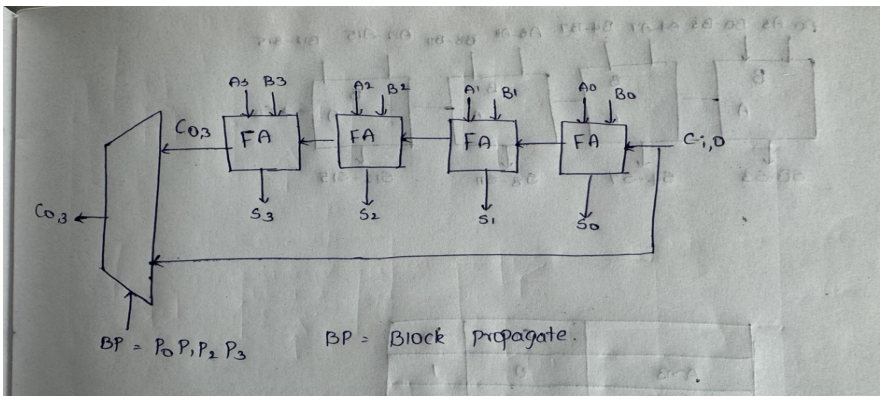


Fig.60 : 4-bit Carry Skip Adder

4.1 RTL Simulation:

In this RTL simulation we will verify the logic without delays.

The RTL Simulation is done by using the cpu_csa.v and tb_cpu.v Verilog files.

It can be run by “xrun tb_cpu.v cpu_csa.v +access+r”.

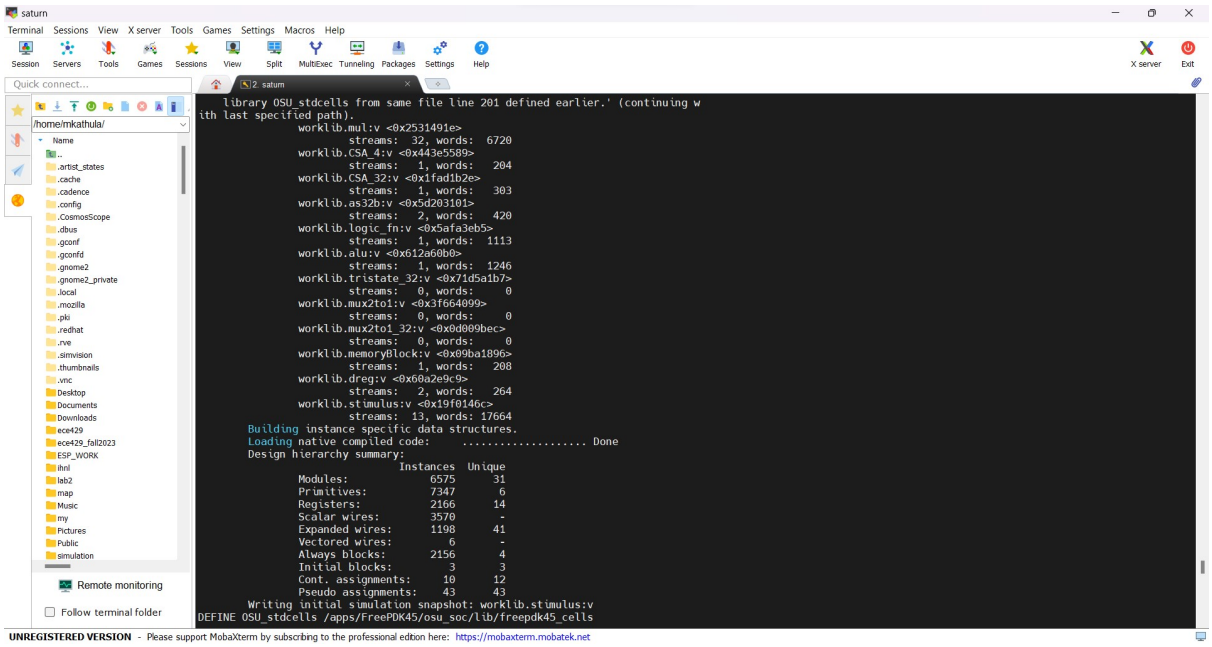


Fig. 61 : CSA RTL Simulation

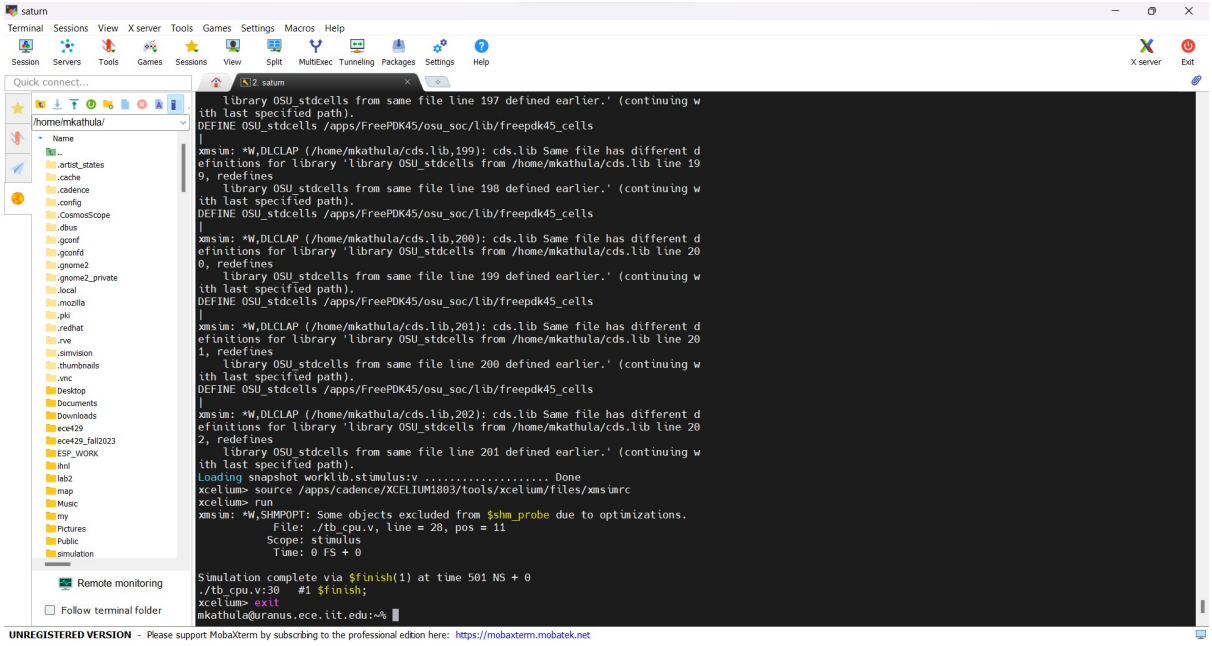


Fig. 62 : CSA RTL Simulation

There are few warnings but there were no errors associated with the Verilog files.

And the wave forms can be seen by using “simvision”.

When the simvison is opened

- Open database
- Click on shm.db
- Right click on shm.db and send to waveform window.

The wave form can be as follows.

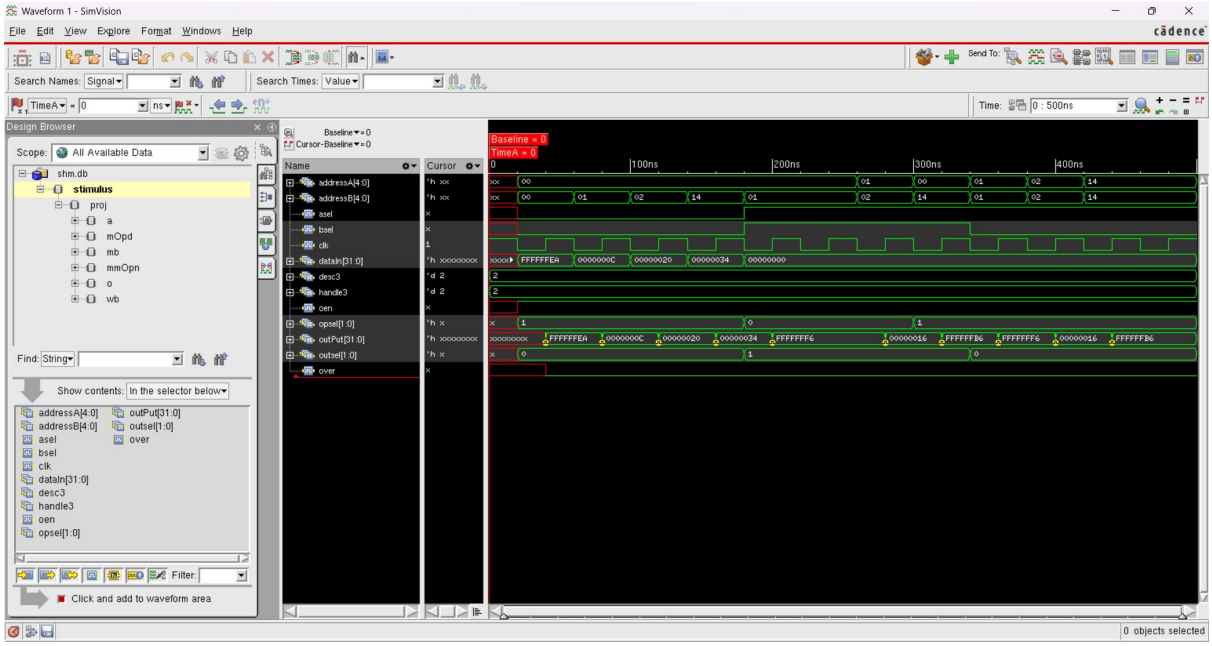


Fig. 63 : Sim Vision Waveform

4.2 Logical Synthesis and Post-Synthesis Simulation:

For this we import the cadence library as

cd /import/scripts/FreePDK45_2011/FreePDK45/osu_soc/flow_ece429/* .

By this we will get the compiling files.

Here we need to change the file name and set my_toplevel to the respective filemename and module name as “[list cpu_CRA.v]” and “cpu”. The frequency must be 30MHz.

For the compile report we will run

“dc_shell -f compile_dc.tcl”

The cell report can be obtained by “nano cell.report”

It is as follows

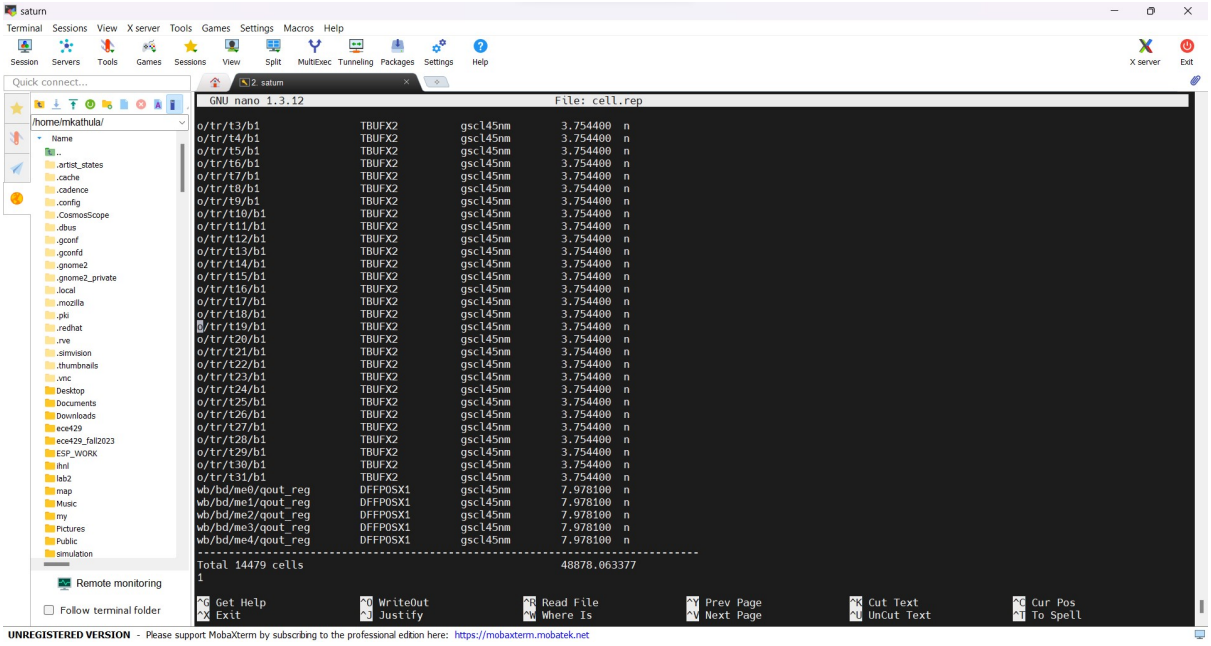


Fig. 64 : Cell Report

The timing report can be obtained by “nano timing.rep”

And it is as follows

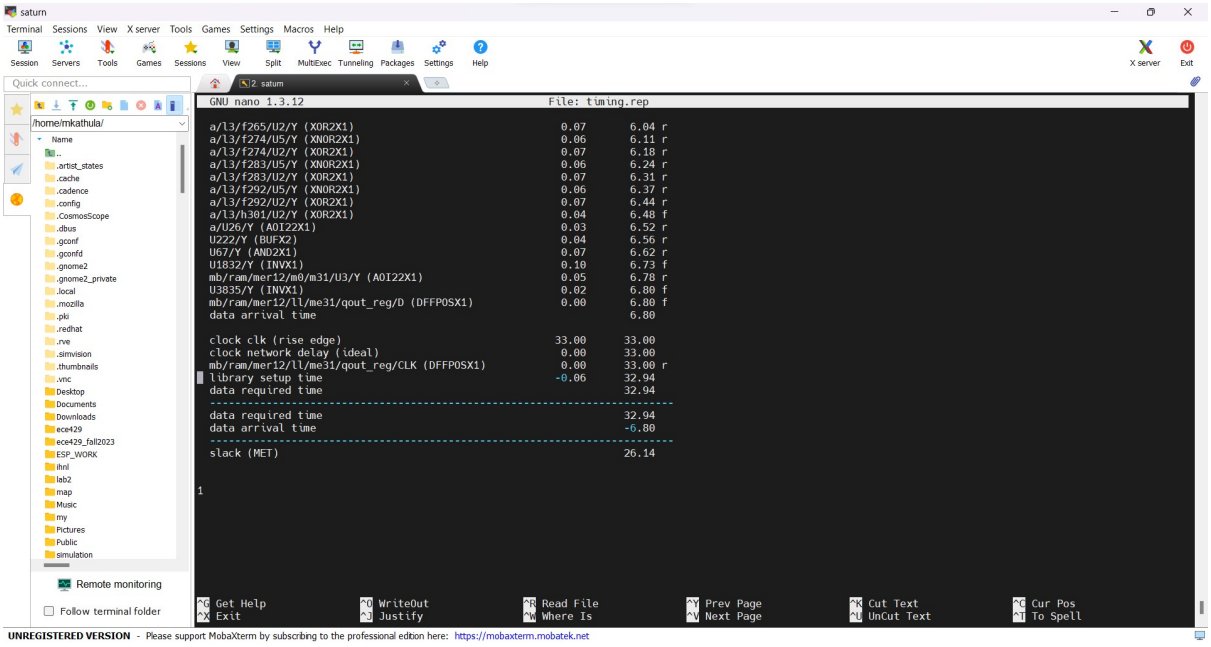


Fig. 65 : Timing Report

The power report can be obtained by “nano power.rep”

It is as follows

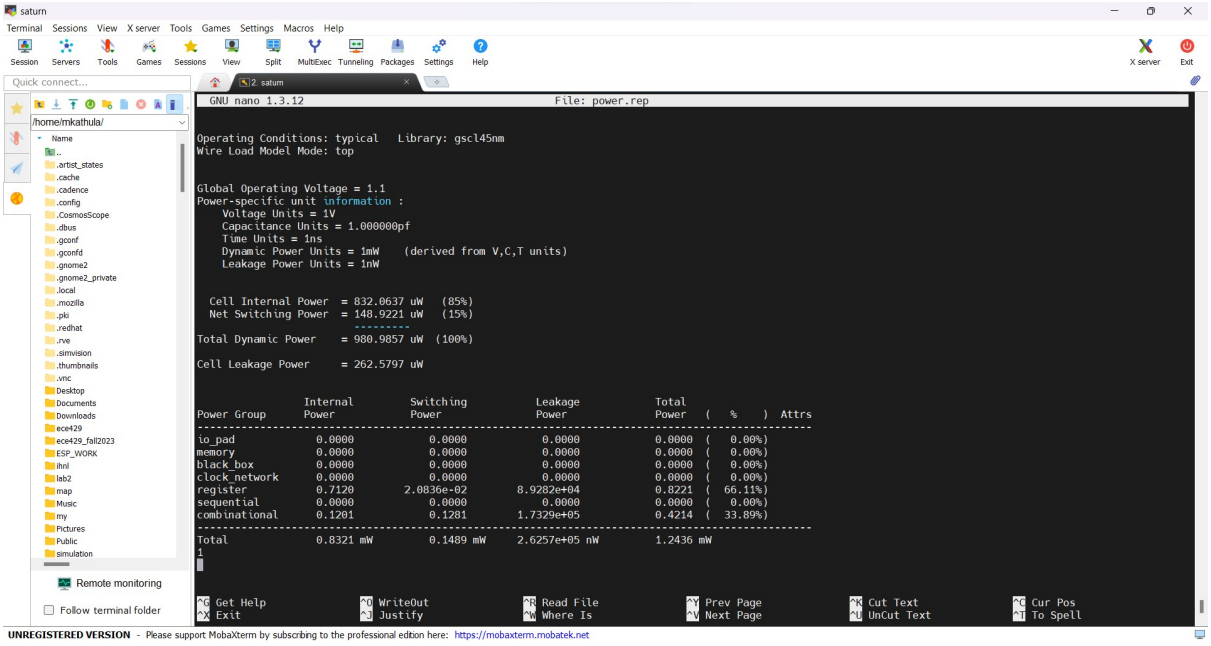


Fig. 66 : Power_Report

For the synthesis we will simulate with Verilog models of standard cells,i.e. gscl45nm.v, and we compare the result with rtl simulation.

This can be done by “xrun gscl45nm.v tb_cpu.v cpu.vh +access+r”

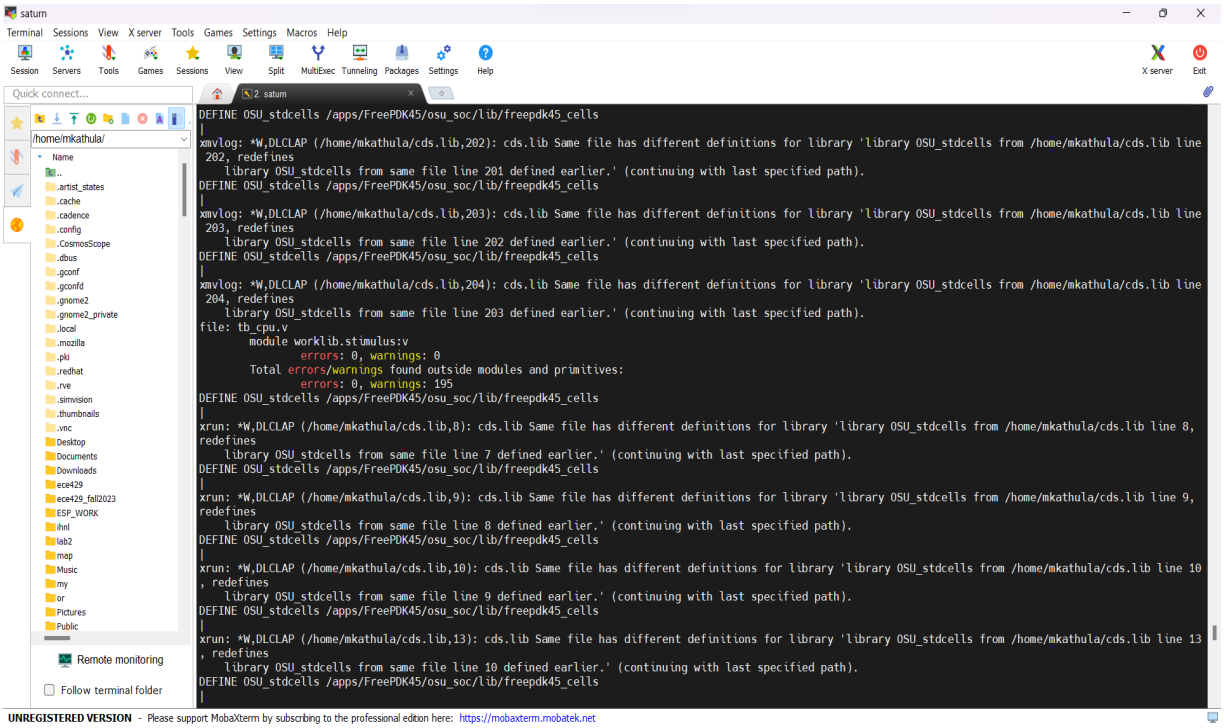


Fig.67 : Logical & post Synthesis Simulation

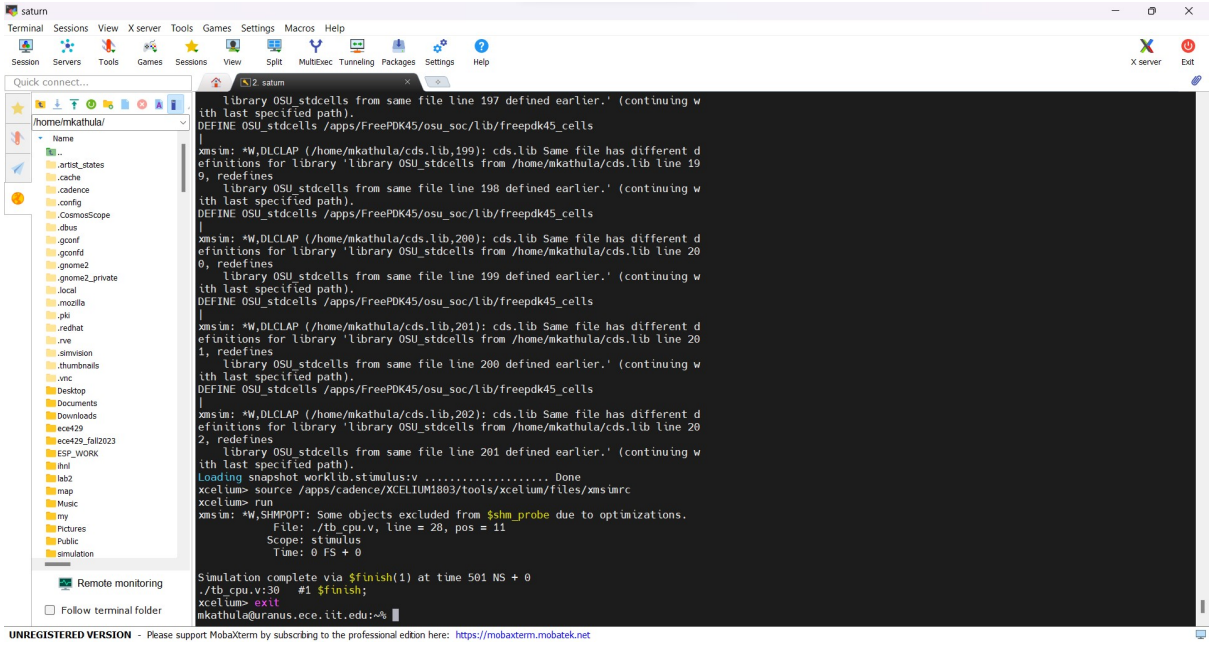


Fig. 68 : Logical & post Synthesis Simulation

It shows same as the RTL Simulation.

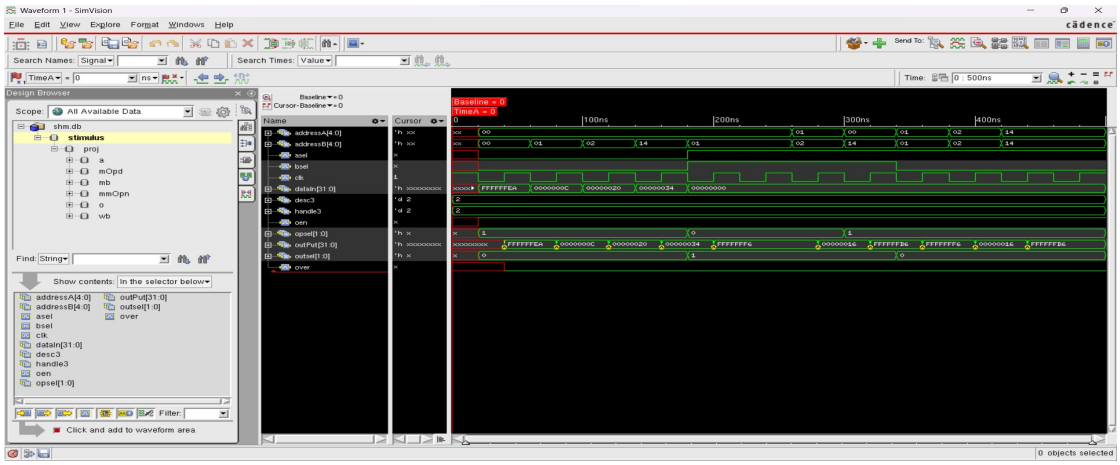


Fig.69 :Simvision for Logical and Post Synthesis Stimulation

4.3 Place & Route and Post-P&R Simulation:

After the synthesis we will change the set my_toplevel to “cpu” in “encounter.conf”.

Now for the encounter results we run the

“encounter -init encounter.tcl”

For Gate Area we will run “reportGateCount -limit 0”.

It can be as follows

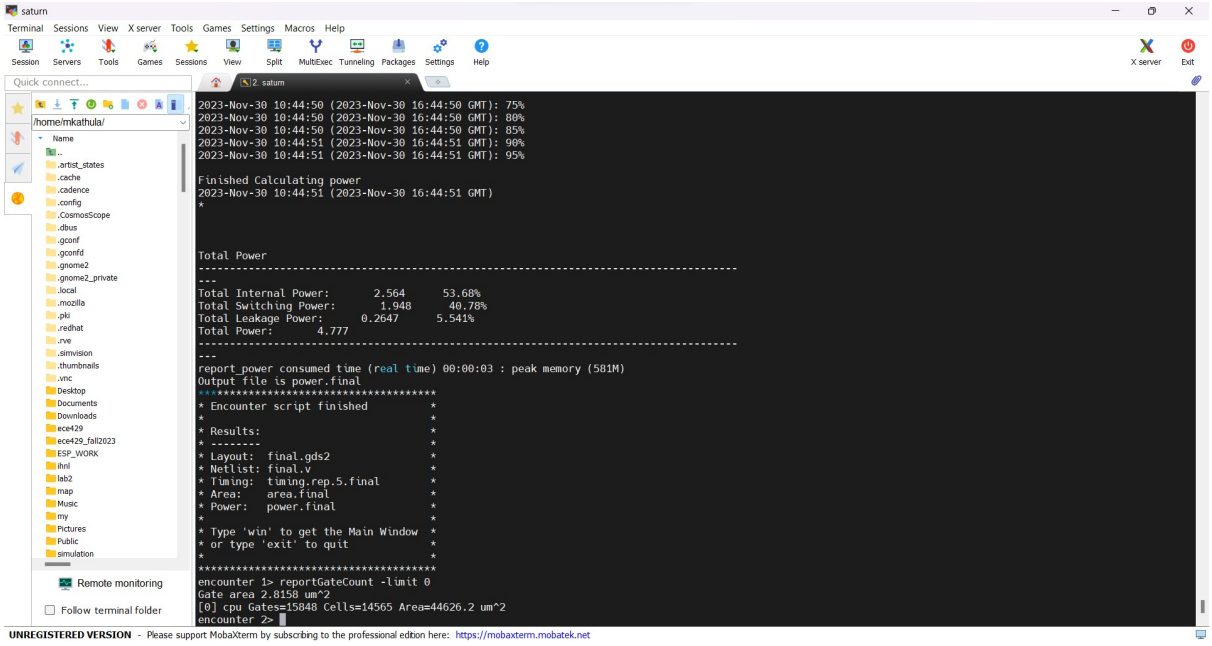


Fig. 70 : Gate Area

The power can be obtained by “report_power”.

It can be as follows

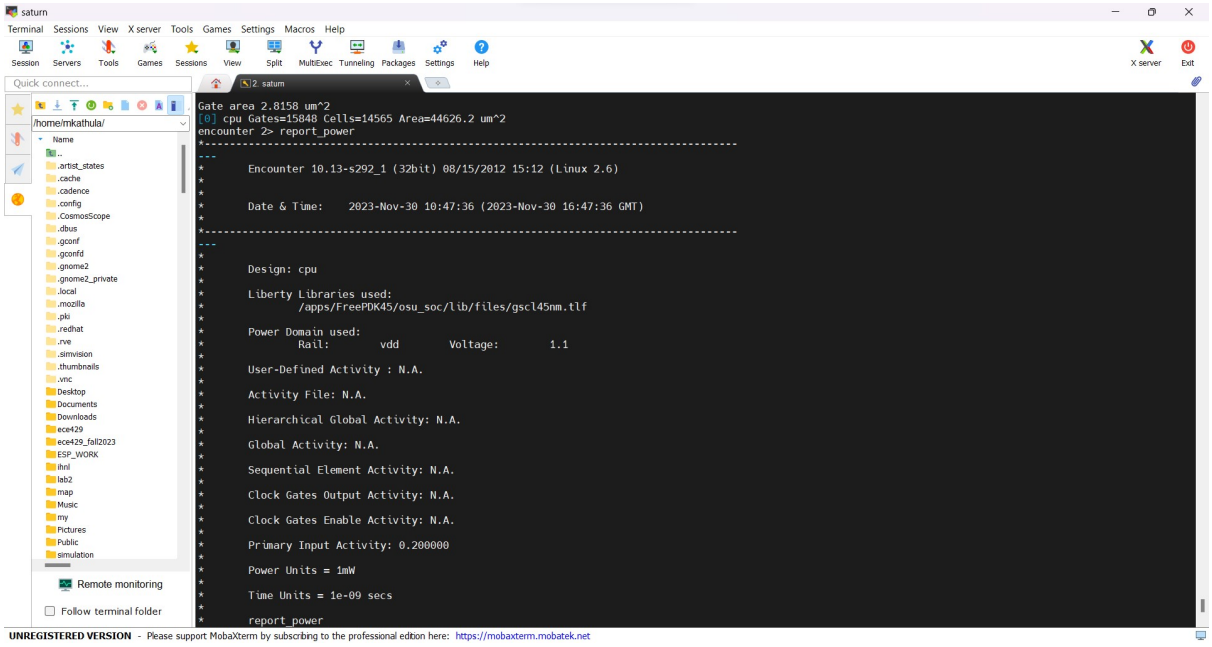
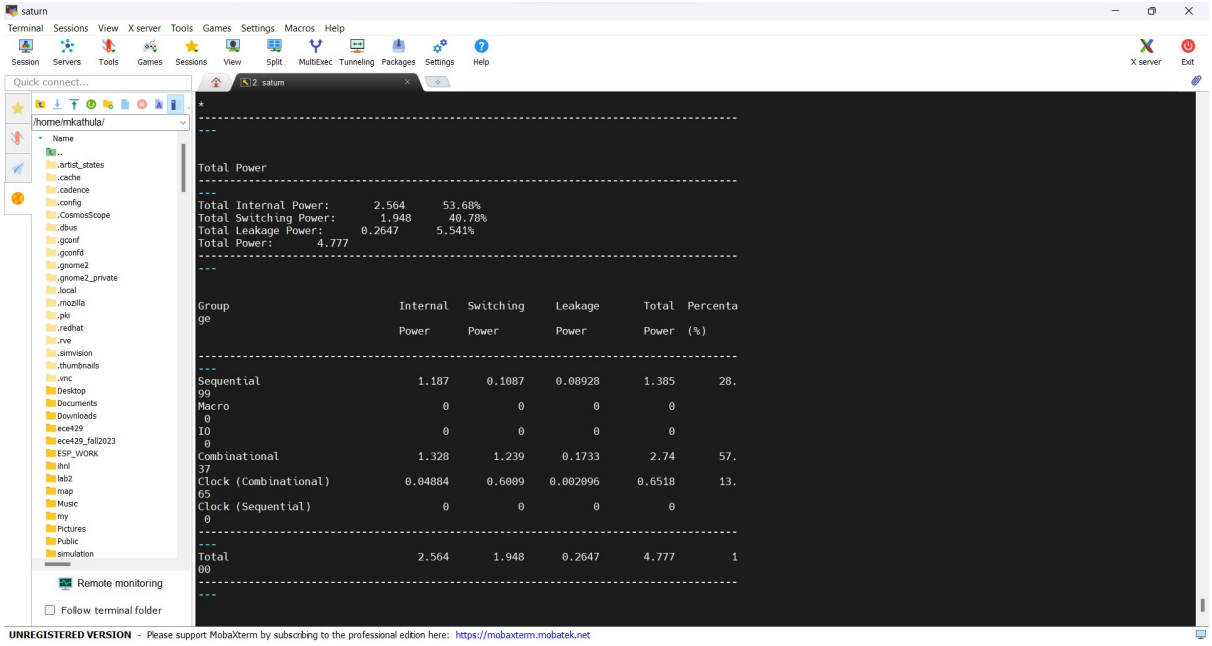
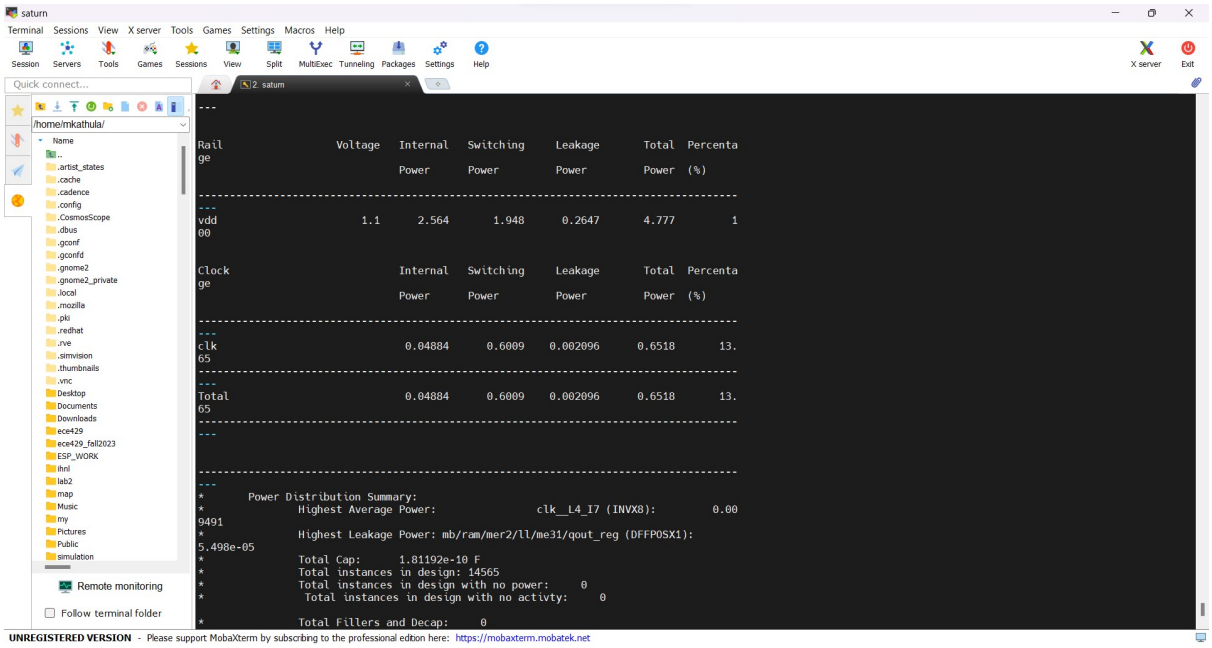


Fig. 71 : Report_Power



```
*
-----------------------------------------------------------------------------
---


Total Power
-----------------------------------------------------------------------------
---
Total Internal Power:       2.564      53.68%
Total Switching Power:      1.948      40.78%
Total Leakage Power:       0.2647      5.541%
Total Power:       4.777
-----------------------------------------------------------------------------
---


Group                          Internal   Switching     Leakage       Total  Percenta
ge
                               Power      Power         Power         Power  (%)

-----------------------------------------------------------------------------
---
Sequential                       1.187      0.1087     0.08928       1.385       28.
99
Macro                                0          0           0           0
 0
IO                                   0          0           0           0
 0
Combinational                    1.328      1.239      0.1733        2.74       57.
37
Clock (Combinational)          0.04884     0.6009    0.002096      0.6518       13.
65
Clock (Sequential)                   0          0           0           0
 0
-----------------------------------------------------------------------------
---
Total                            2.564      1.948      0.2647       4.777        1
00
-----------------------------------------------------------------------------
---
```

Fig.72 : Report_Power



```
---


Rail                  Voltage   Internal   Switching     Leakage       Total  Percenta
ge
                                Power      Power         Power         Power  (%)

-----------------------------------------------------------------------------
---
vdd                       1.1      2.564      1.948      0.2647       4.777        1
00


Clock                           Internal   Switching     Leakage       Total  Percenta
ge
                                Power      Power         Power         Power  (%)

-----------------------------------------------------------------------------
---
clk                            0.04884     0.6009    0.002096      0.6518       13.
65
-----------------------------------------------------------------------------
---
Total                          0.04884     0.6009    0.002096      0.6518       13.
65
-----------------------------------------------------------------------------
---


-----------------------------------------------------------------------------
---
*       Power Distribution Summary:
*               Highest Average Power:                clk__L4_I7 (INVX8):        0.00
9491
*               Highest Leakage Power: mb/ram/mer2/ll/me31/qout_reg (DFFPOSX1):
5.498e-05
*               Total Cap:      1.81192e-10 F
*               Total instances in design: 14565
*               Total instances in design with no power:     0
*                Total instances in design with no activty:     0

*               Total Fillers and Decap:     0
```

Fig.73 : Report_Power

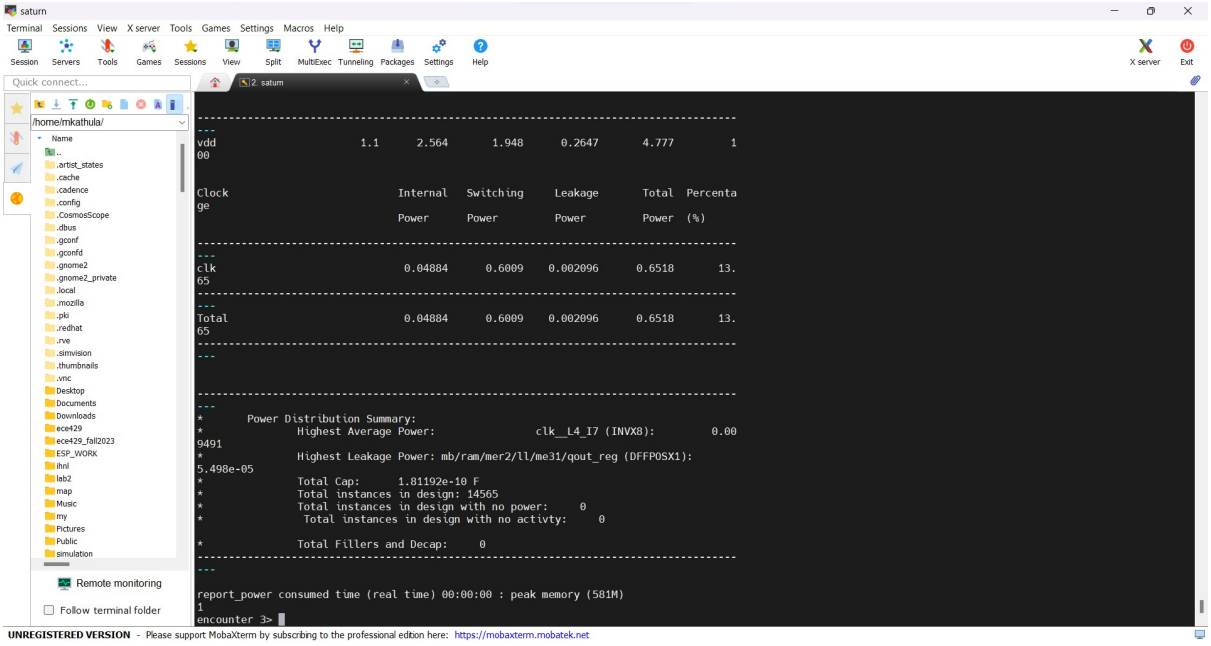


Fig. 74 : Report_Power

For Place and Route and Post-P&R Simulation we run “xrun gscl45nm.v tb_cpu.v final.v +access+r”

It is as follows:

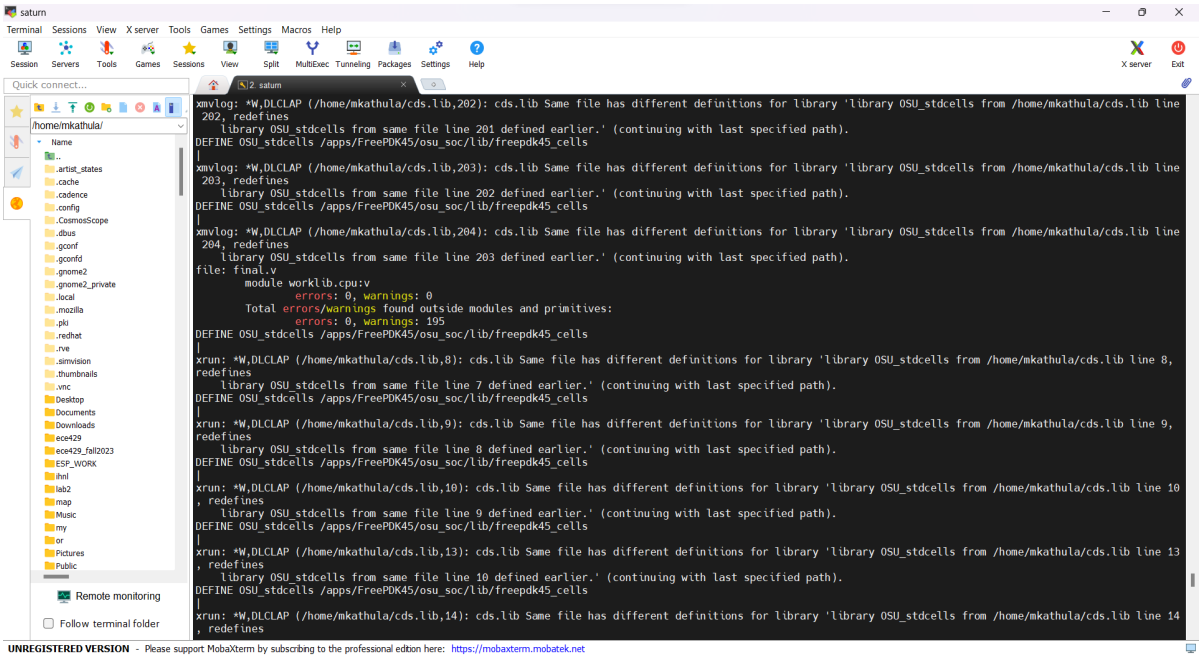


Fig. 75 :Place & Route and Post-P&R Simulation

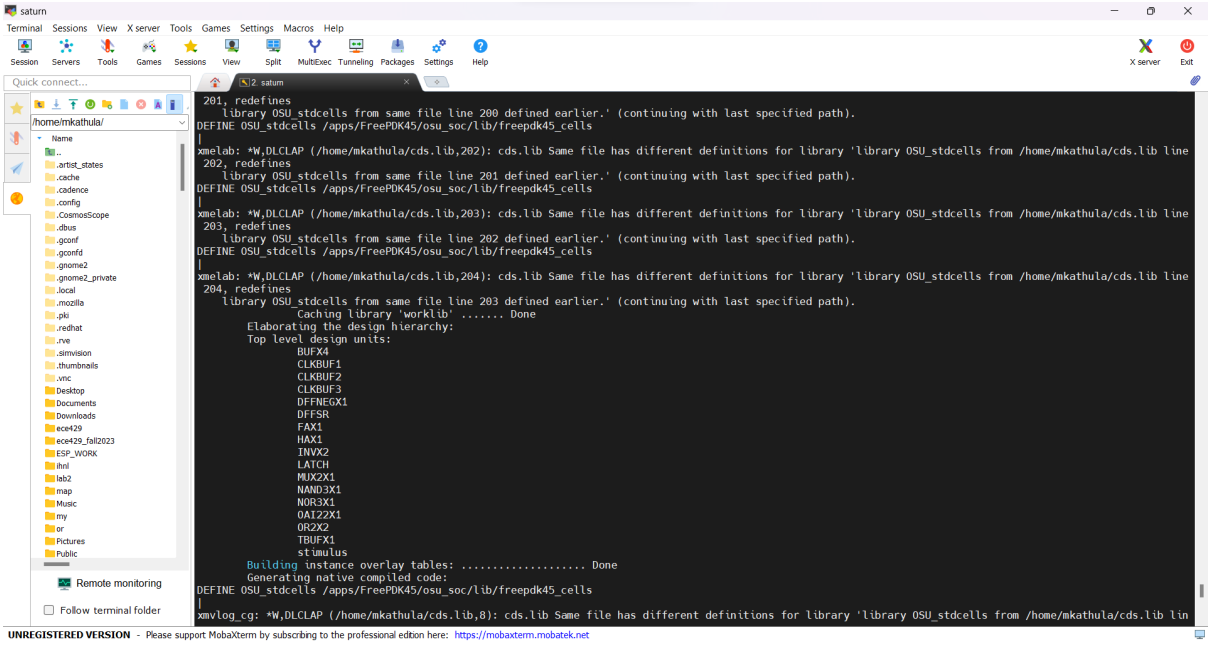


Fig. 76 :Place & Route and Post-P&R Simulation

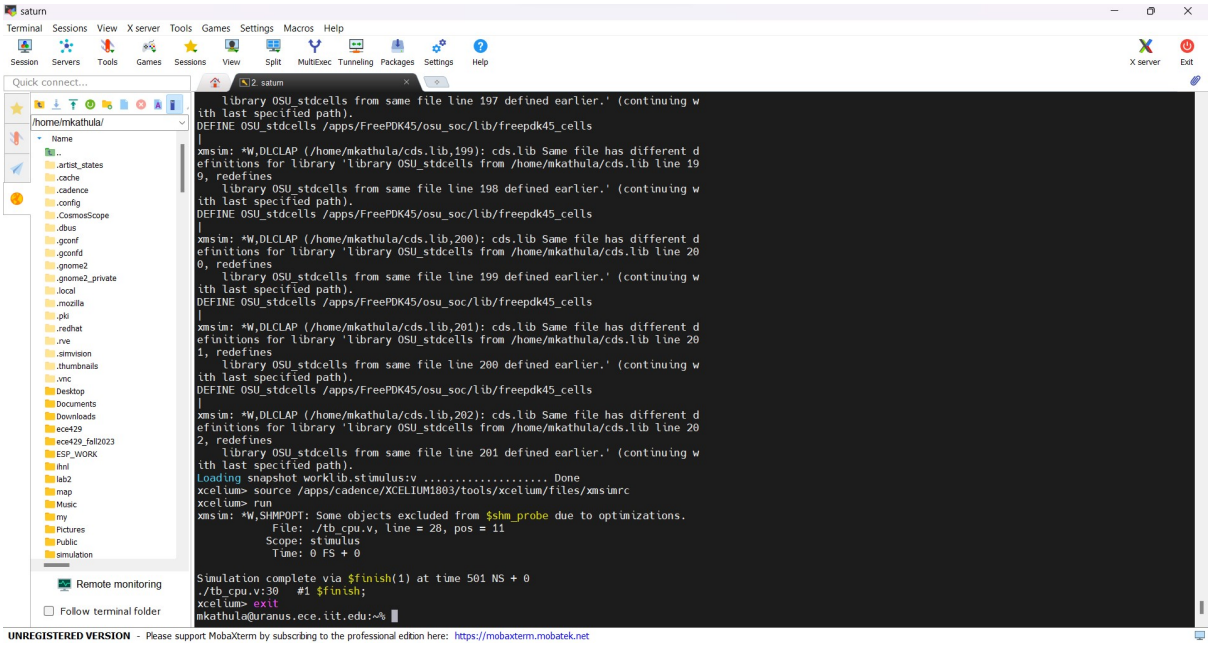


Fig. 77 :Place & Route and Post-P&R Simulation

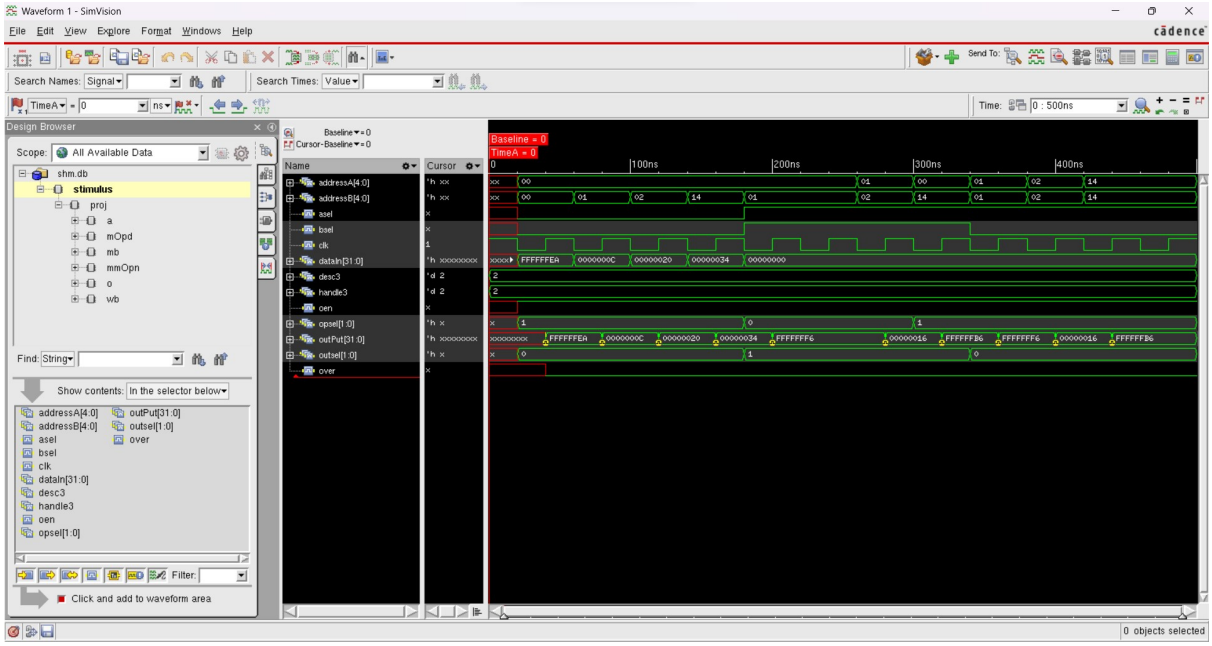


Fig.78 : SimVision for Place & Route and Post-P&R Simulation

4.4 Maximum Clock Frequency:

This can be obtained in timing.rep.5.final.

Here the slack time has decreased to 18.318, Based on the required time of 29.364, max clock frequency is 34.05Mhz(1/29.364).

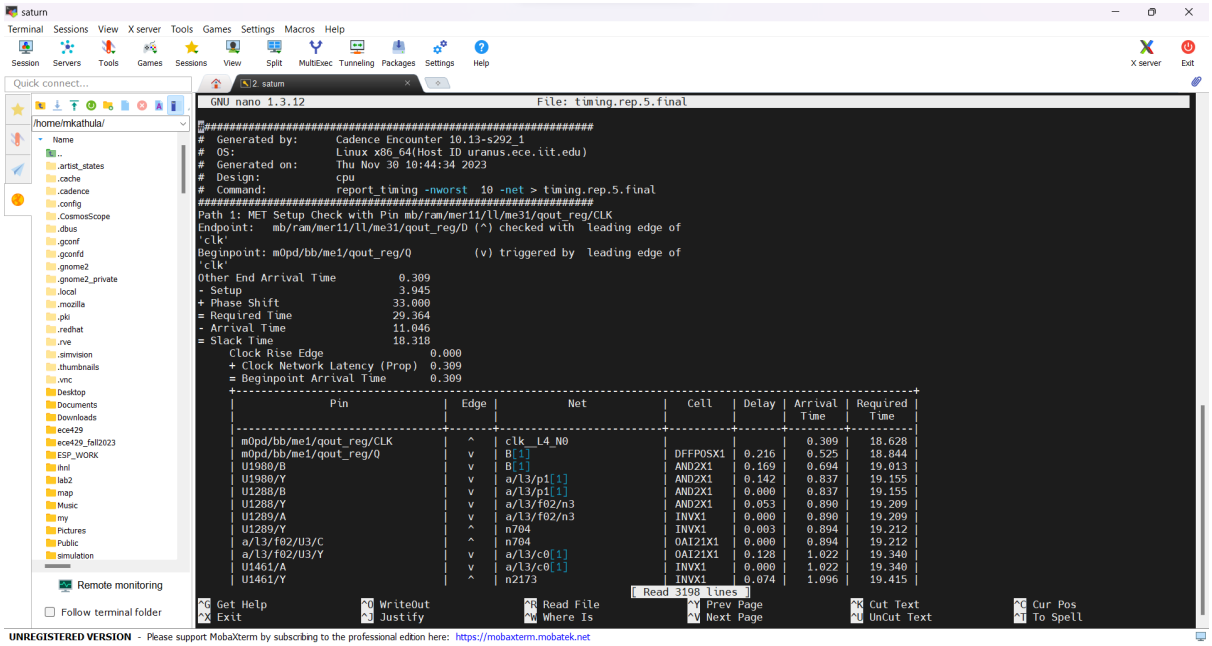


Fig. 79 : Timing.rep.5.final file for Maximum Clock Frquency

4.5 New Test Bench:

The source code of the test bench is given in the appendix. By the tb_test_cpu.v we verify using RTL simulation.

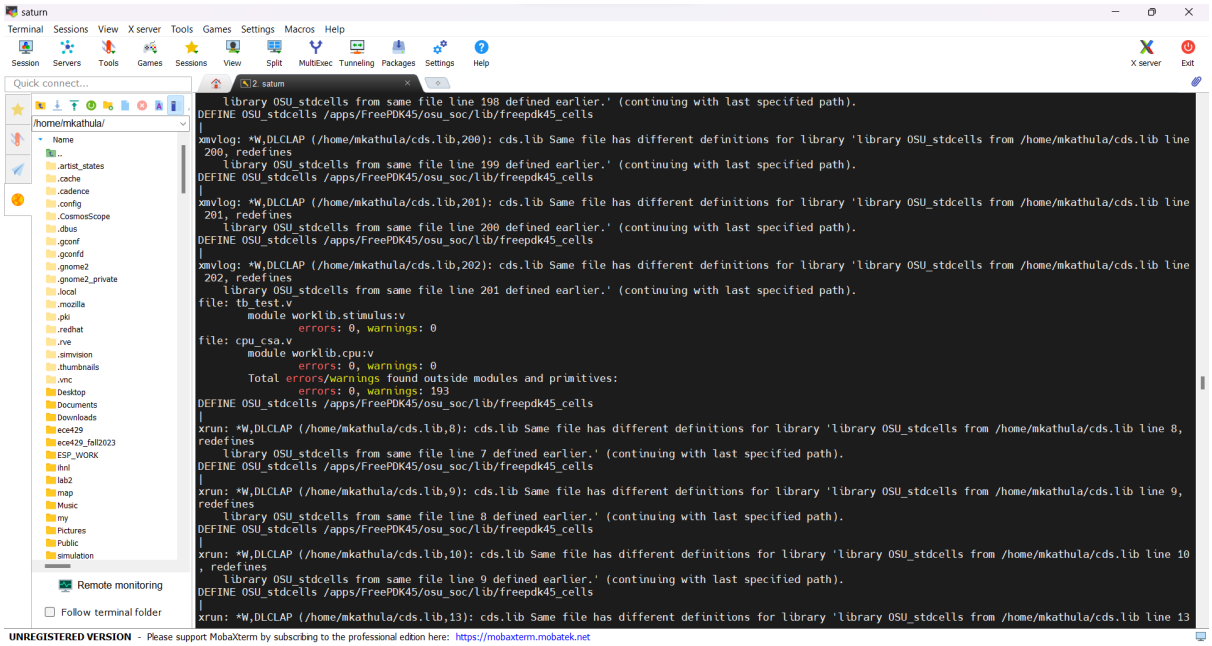


Fig.80 :RTL Simulation for New Test Bench

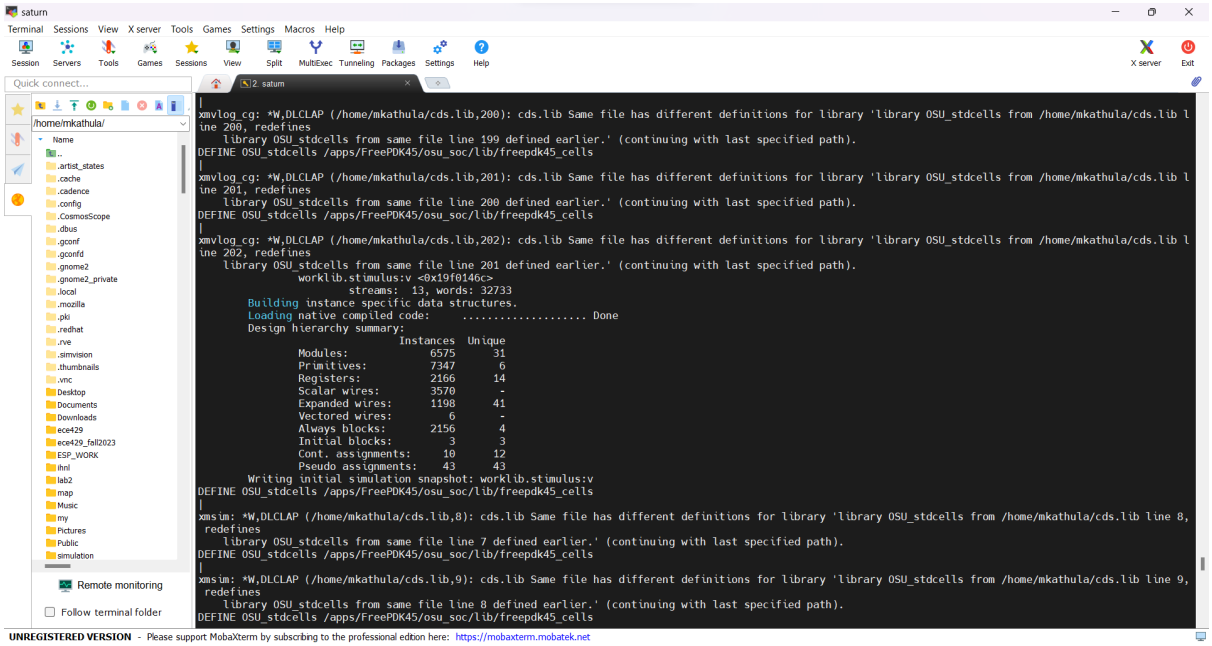


Fig.81 :RTL Simulation for New Test Bench

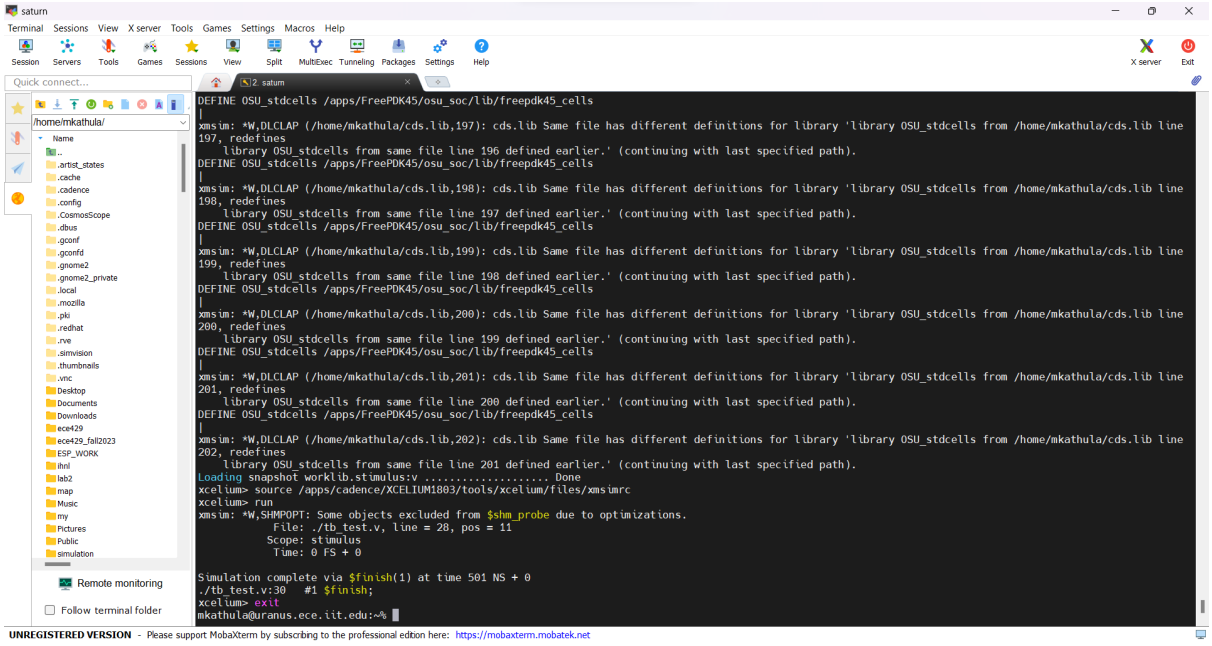


Fig. 82 :RTL Simulation for New Test Bench

By Using the simvision we can get the waveforms of the new test bench

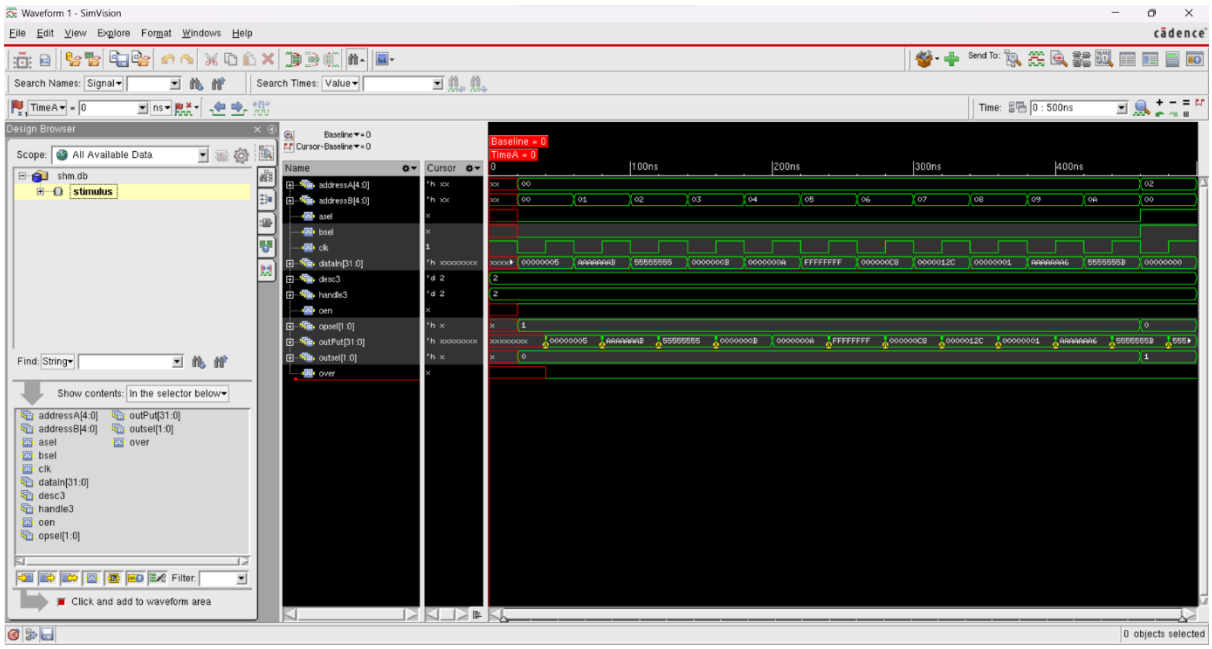


Fig.83 :Simvision for New Test Bench

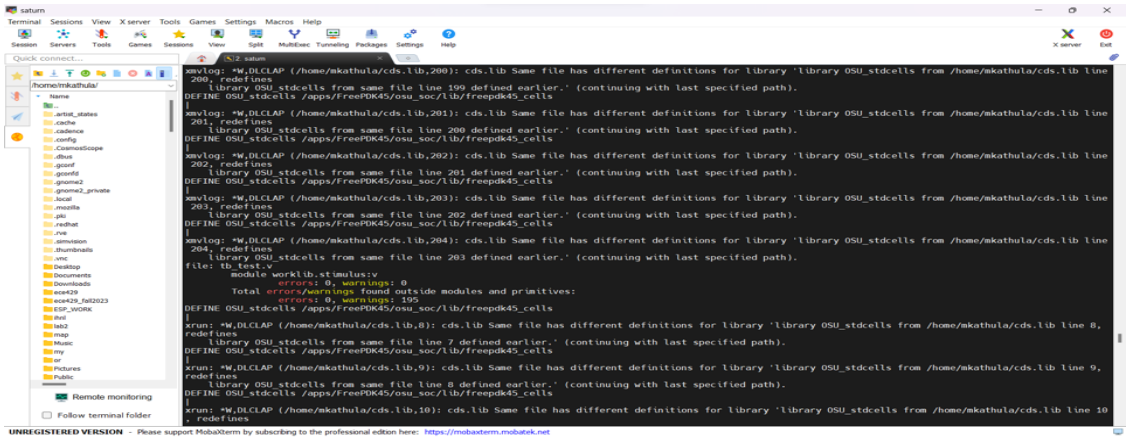


Fig. 84 :Logical and Post Synthesis Simulation for New Test Bench

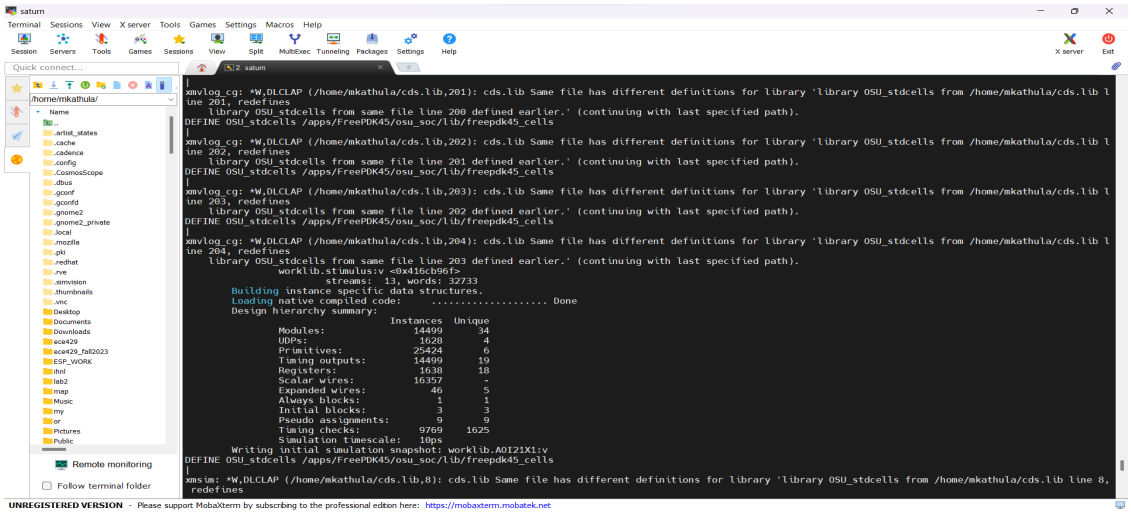


Fig. 85 :Logical and Post Synthesis Simulation for New Test Bench

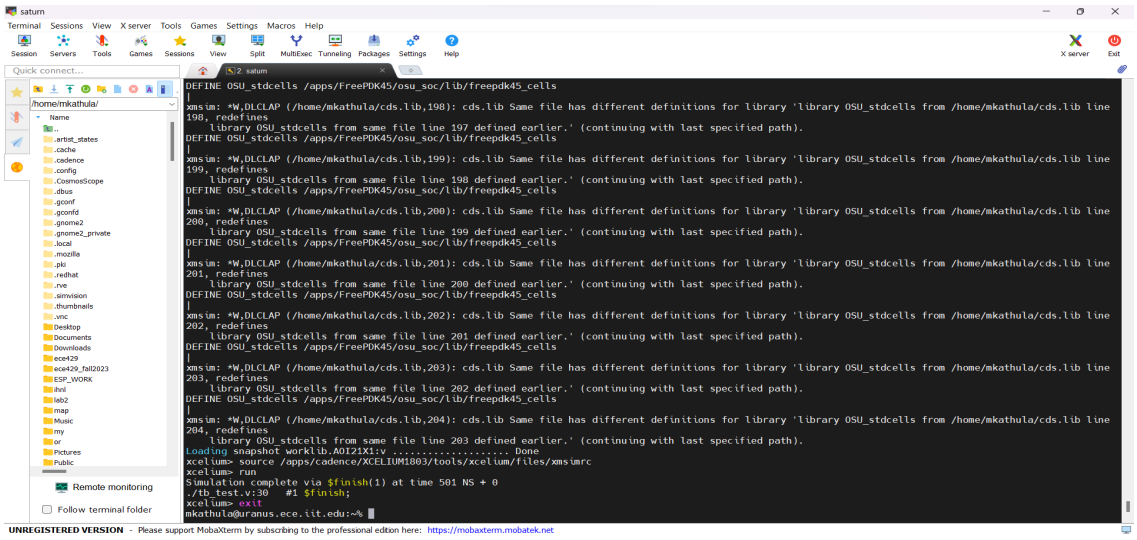


Fig.86 :Logical and Post Synthesis Simulation for New Test Bench

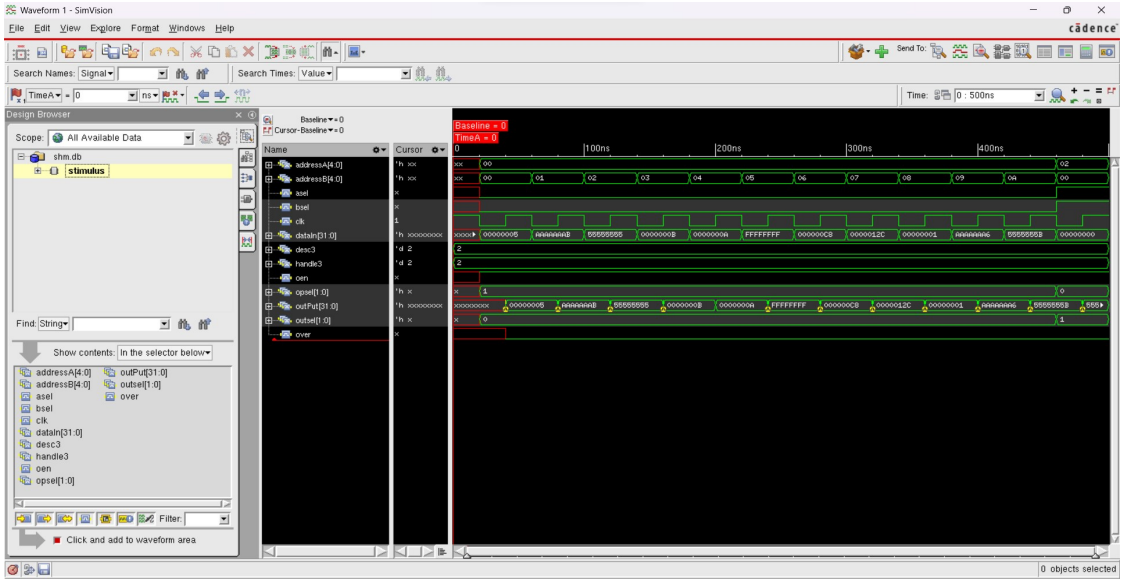


Fig. 87 : Simvision wavefrom for new test bench(Logical and Post synthesis simulation)

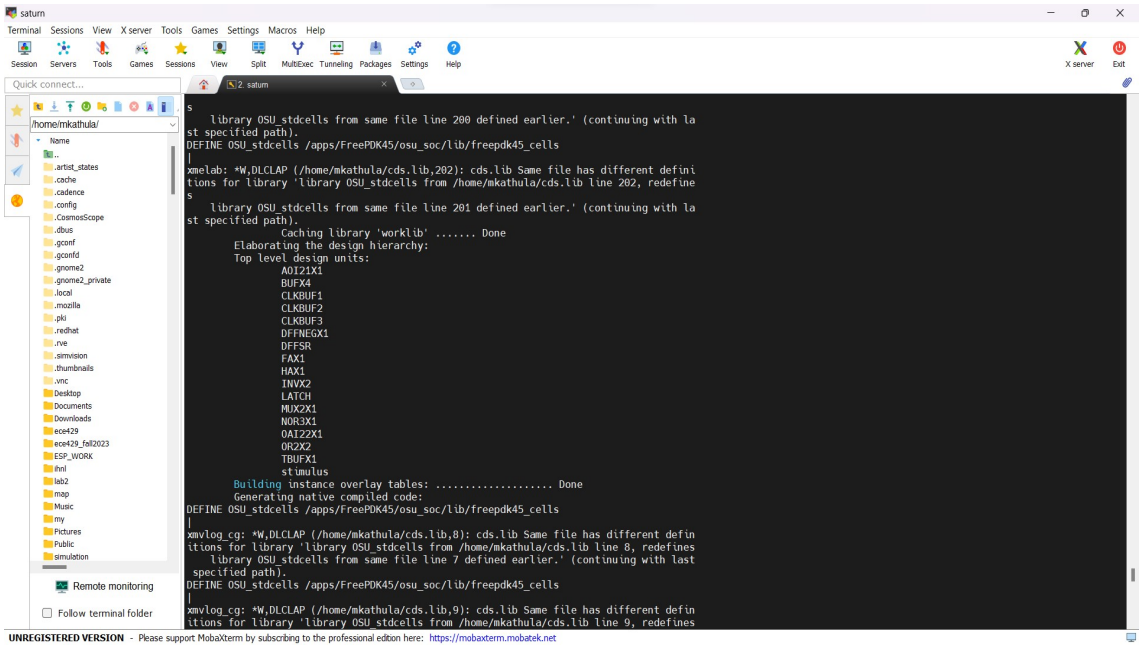


Fig. 88 : Logical and Post synthesis Simulation for new Test Bench

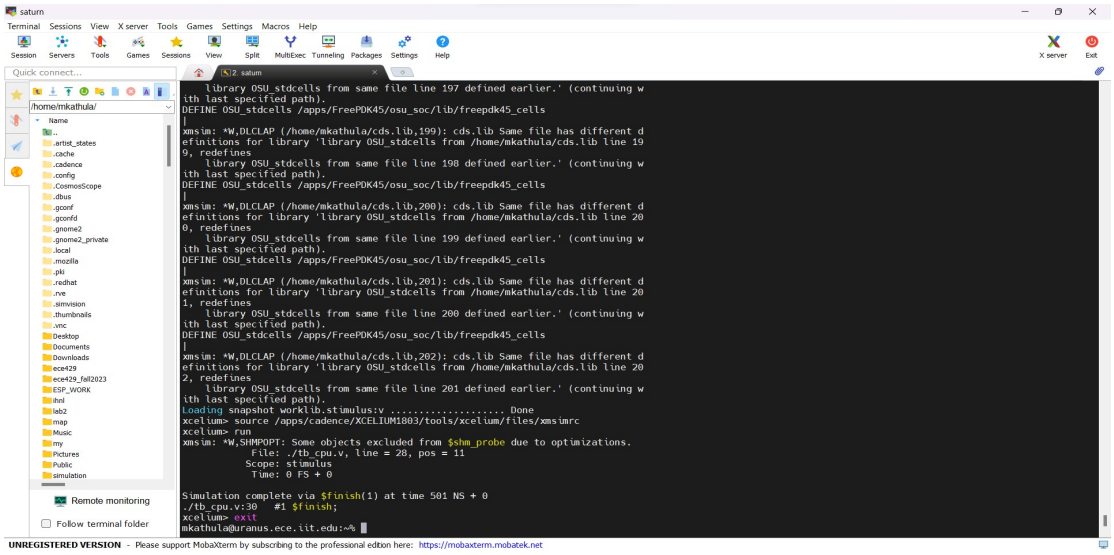


Fig. 89 : Logical and Post synthesis Simulation for new Test Bench

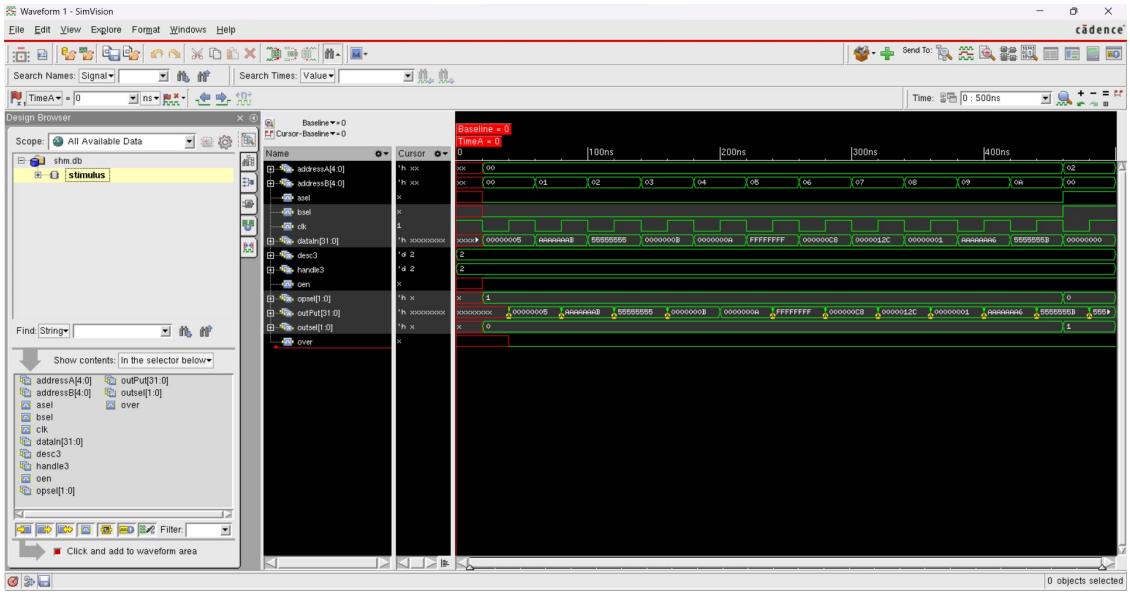


Fig. 90 :Simvision Waveform for new Test Bench(Logical and Post Synthesis simulation)

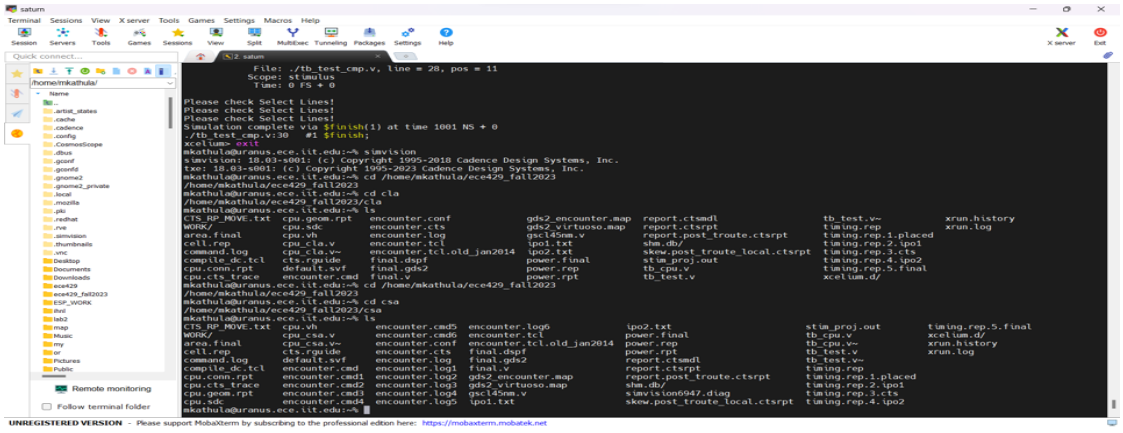


Fig. 91 :Files Created for CSA.

5. Carry Select Adder:

The Carry Select Adder is similar to Carry Skip Adder. Here the operands are divided into r bit blocks same as the carry skip adder. Here, two r-bit Ripple Carry Adder operates in parallel to form 2 sets of sum bits and carry out signals.

The block diagram of Carry Select Adder is as follows

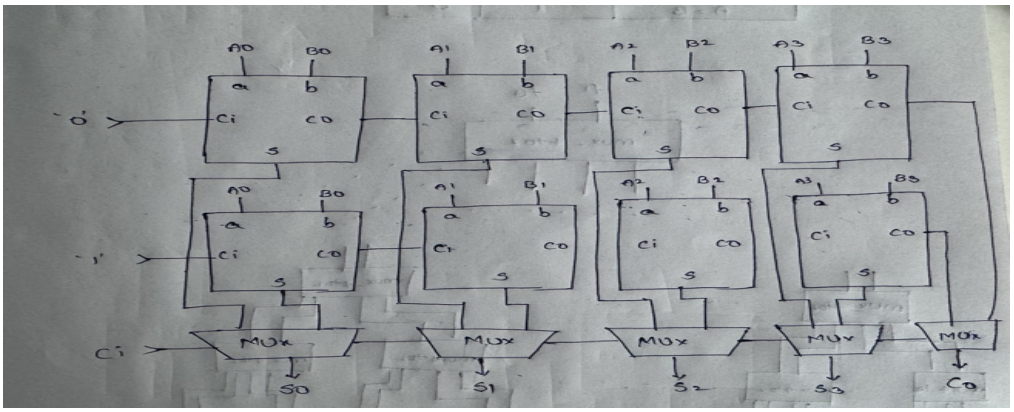


Fig.92 :4-bit Carry Select Adder

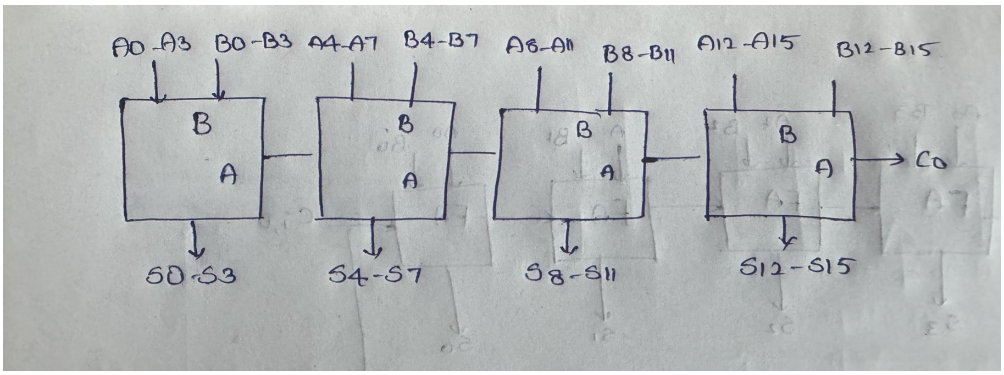


Fig.93 :16-bit Carry Select Adder

5.1 RTL Simulation:

In this RTL simulation we will verify the logic without delays.

The RTL Simulation is done by using the cpu_csea.v and tb_cpu.v Verilog files.

It can be run by “xrun tb_cpu.v cpu_csea.v +access+r”.

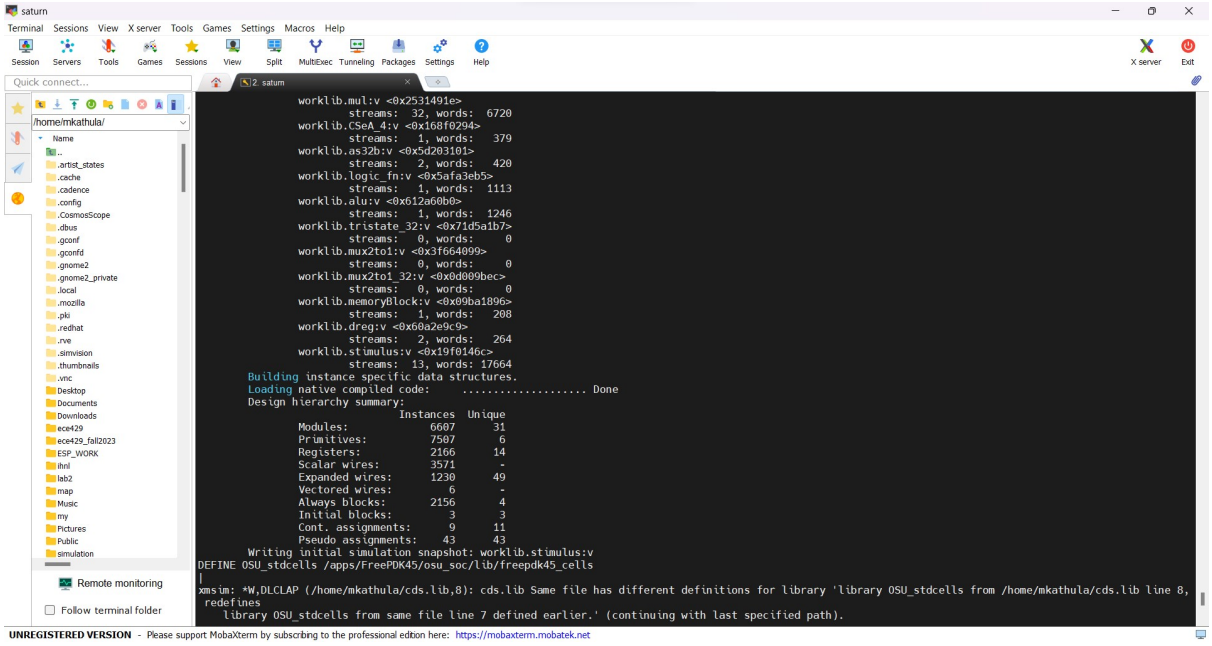


Fig.94 : CSEA RTL Stimulation

There are few warnings but there were no errors associated with the Verilog files.

And the wave forms can be seen by using “simvision”.

When the simvison is opened

- Open database
- Click on shm.db
- Right click on shm.db and send to waveform window.

The wave form can be as follows.

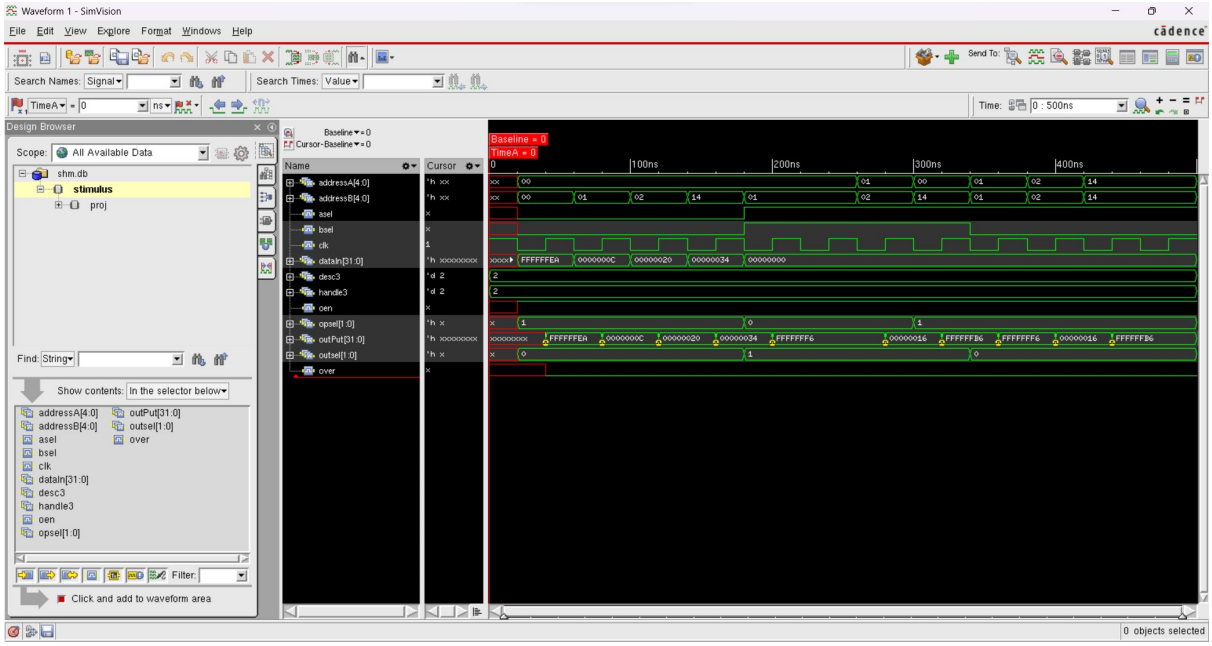


Fig. 95 : SimVision Waveform

5.2 Logical Synthesis and Post-Synthesis Simulation:

For this we import the cadence library as

cd /import/scripts/FreePDK45_2011/FreePDK45/osu_soc/flow_ece429/* .

By this we will get the compiling files.

Here we need to change the file name and set my_toplevel to the respective filemename and module name as “[list cpu_CRA.v]” and “cpu”. The frequency must be 30MHz.

For the compile report we will run

“dc_shell -f compile_dc.tcl”

The cell report can be obtained by “nano cell.report”

It is as follows

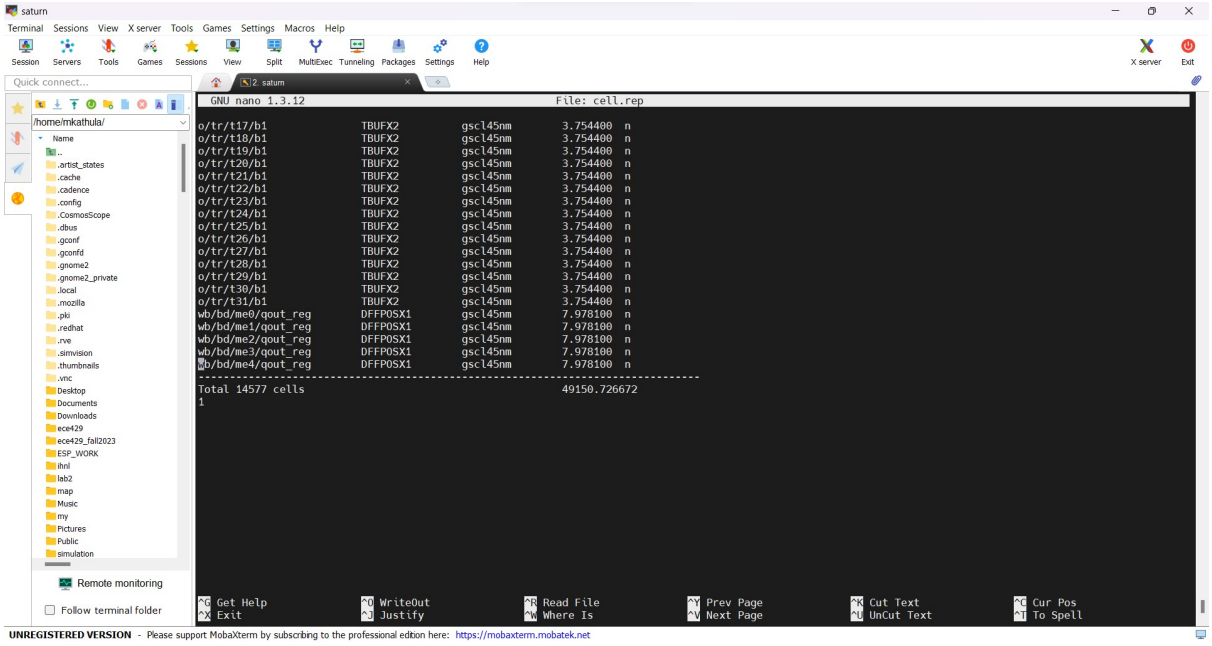


Fig.96 :Cell Report

The timing report can be obtained by “nano timing.rep”

And it is as follows

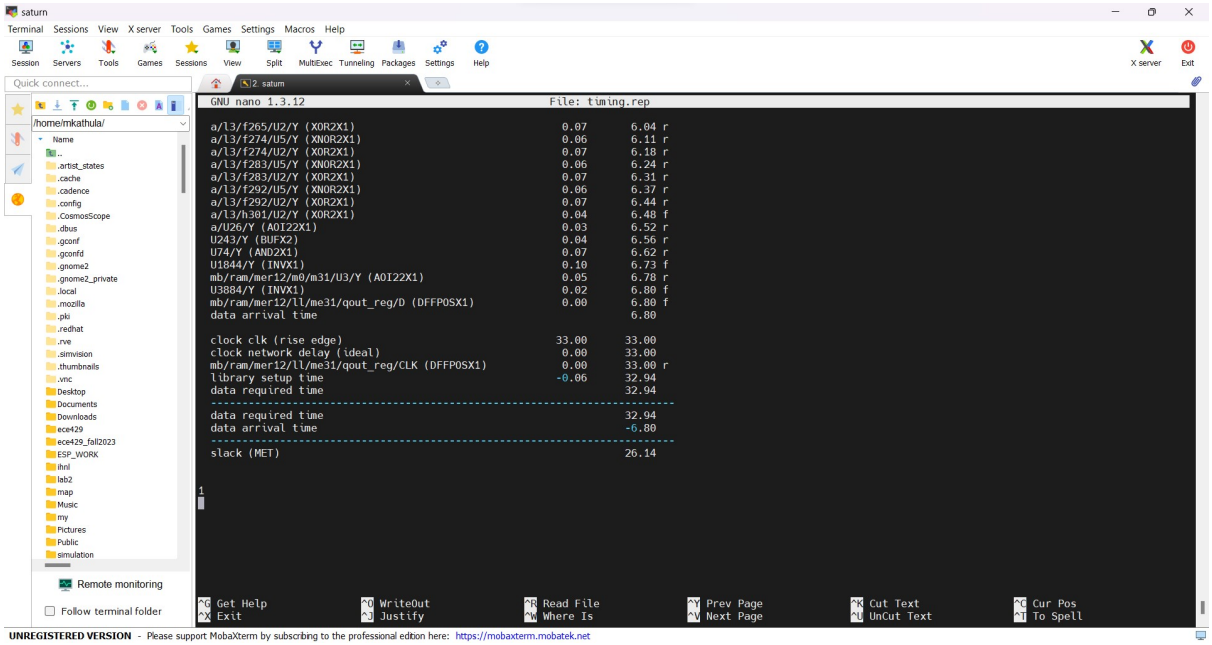


Fig. 97 : Timing Report

The power report can be obtained by “nano power.rep”

It is as follows

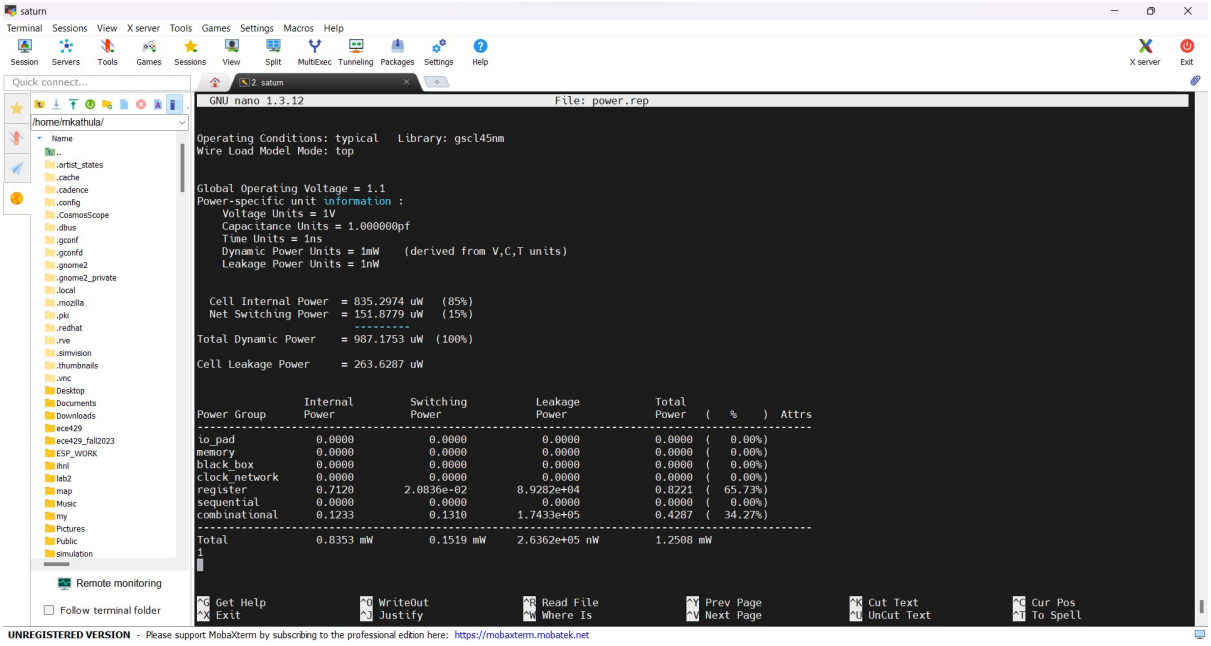


Fig. 98 : Power_Report

For the synthesis we will simulate with Verilog models of standard cells,i.e. gscl45nm.v, and we compare the result with rtl simulation.

This can be done by “xrun gscl45nm.v tb_cpu.v cpu.vh +access+r”

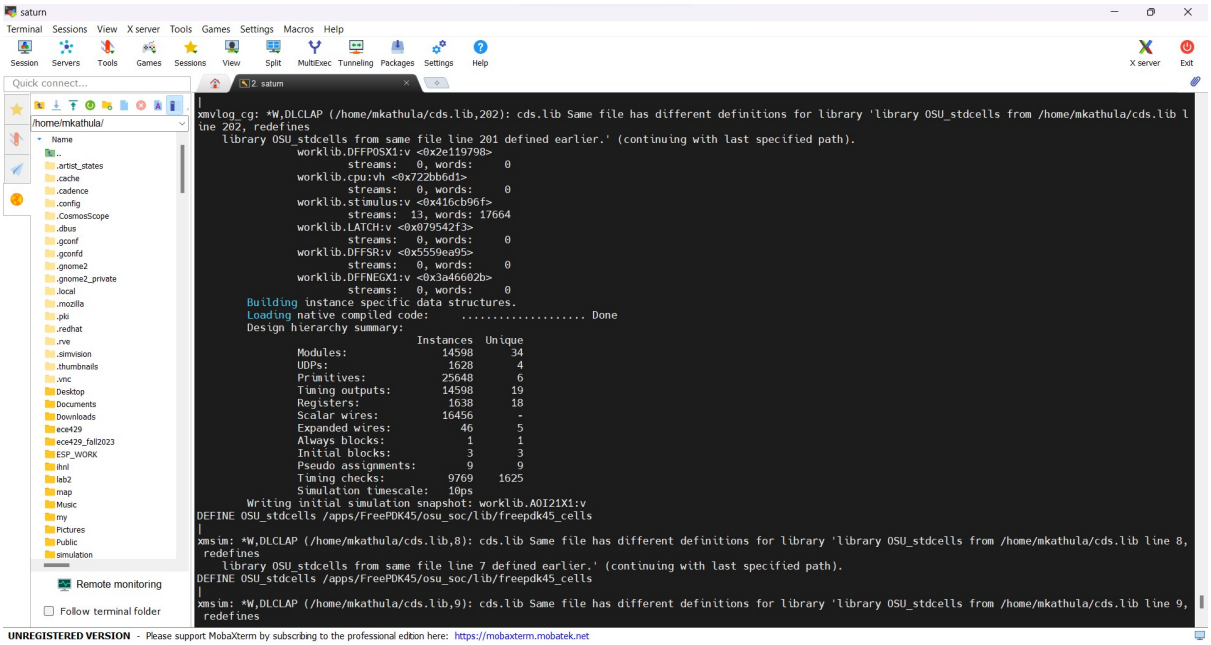


Fig.99 : logical and Post Synthesis Stimulation

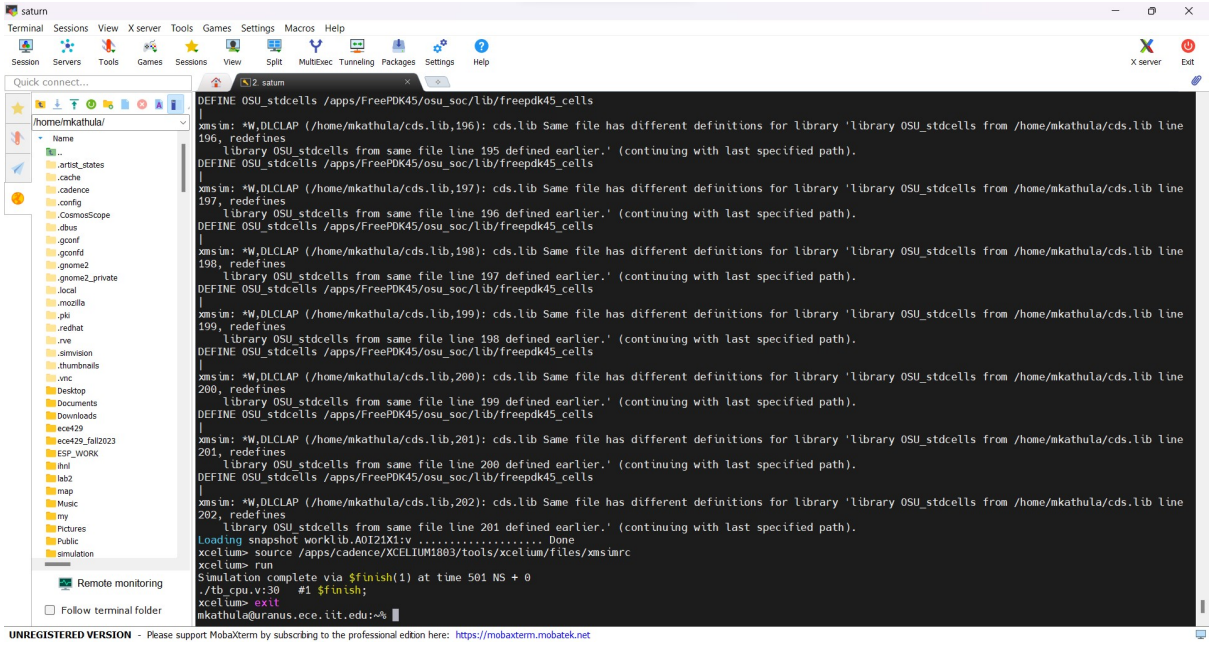


Fig. 100 : logical and Post Synthesis Stimulation

It shows same as the RTL Simulation.

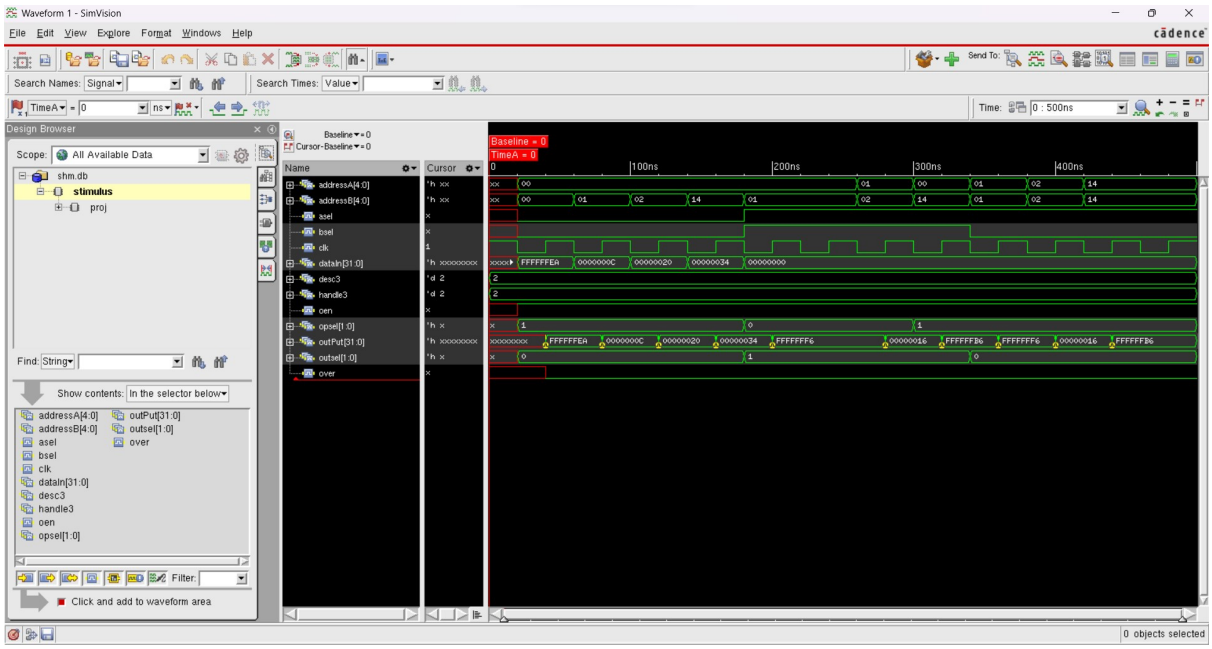


Fig. 101 :Simvision for Logical and post Logical Synthesis Stimulation

5.3 Place & Route and Post-P&R Simulation:

After the synthesis we will change the set my_toplevel to “cpu” in “encounter.conf”.

Now for the encounter results we run the

“encounter -init encounter.tcl”

For Gate Area we will run “reportGateCount -limit 0”.

It can be as follows

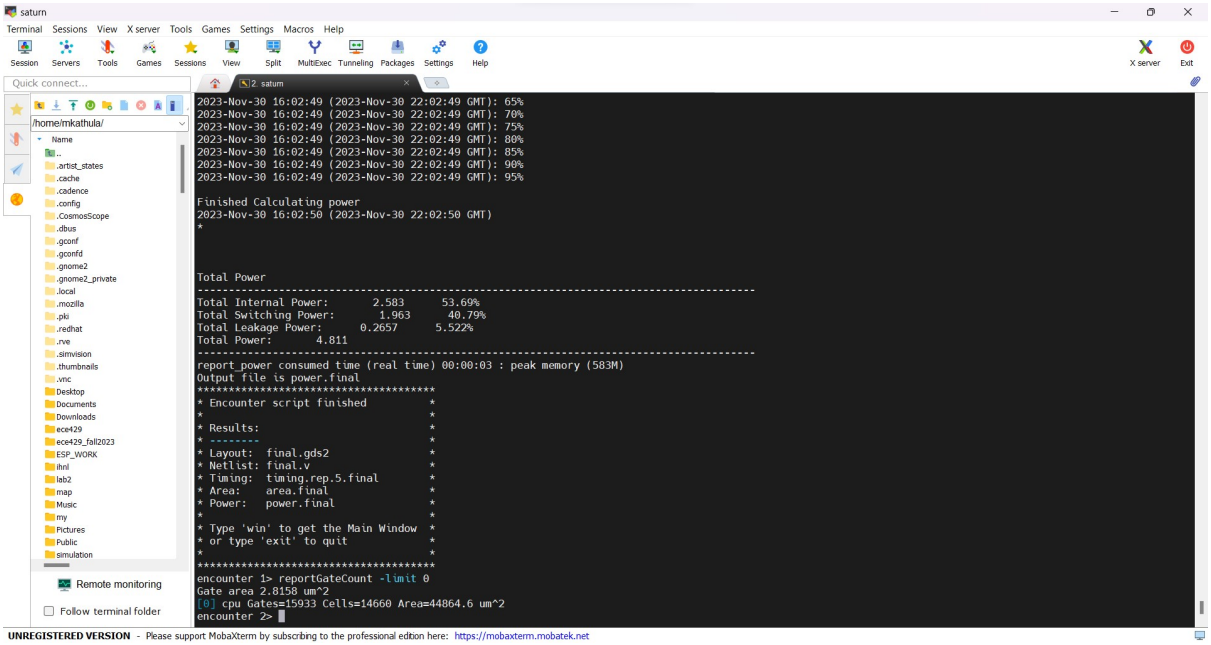


Fig. 102 : Gate Area

The power can be obtained by “report_power”.

It can be as follows

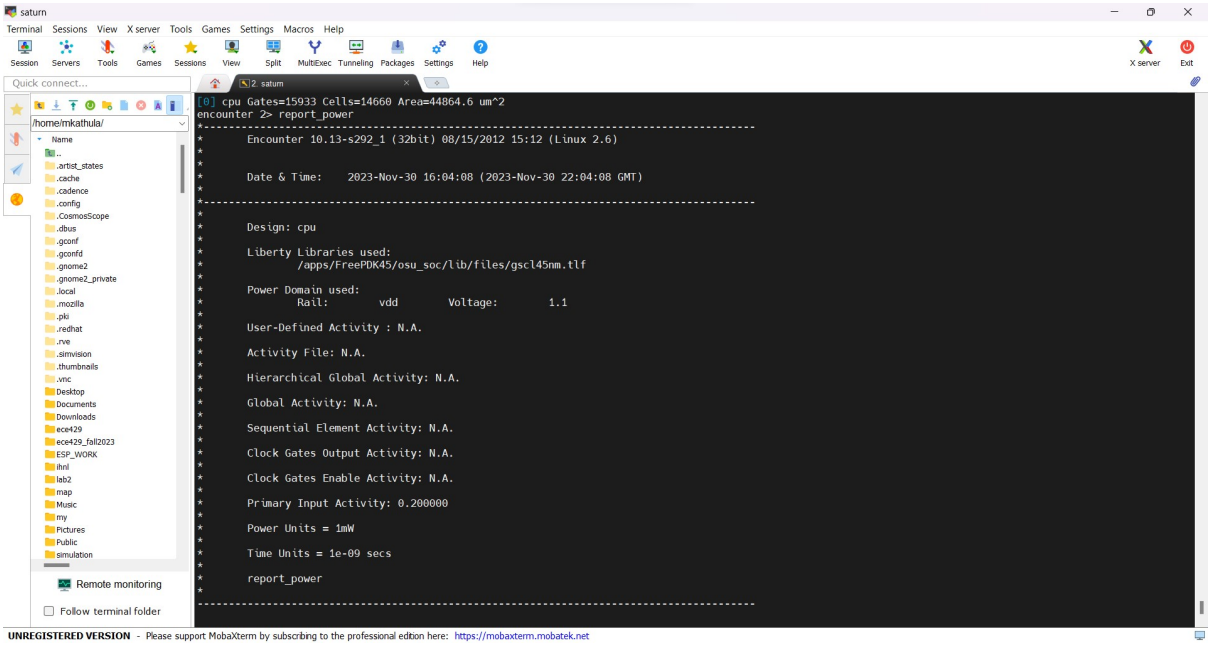
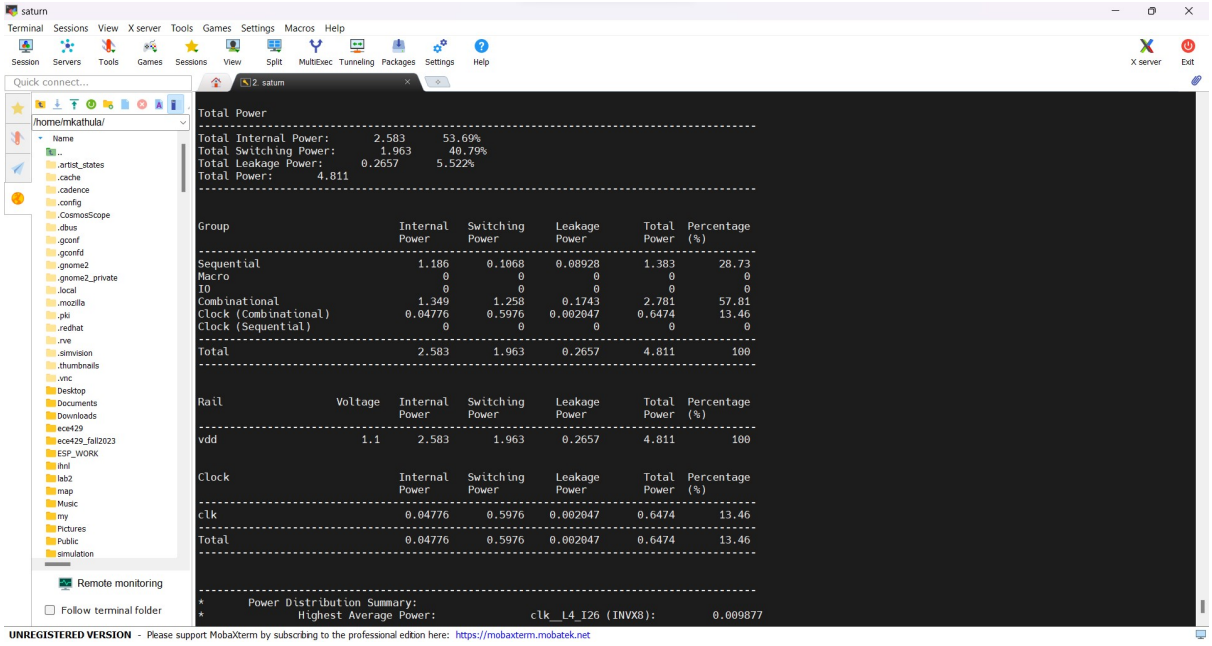
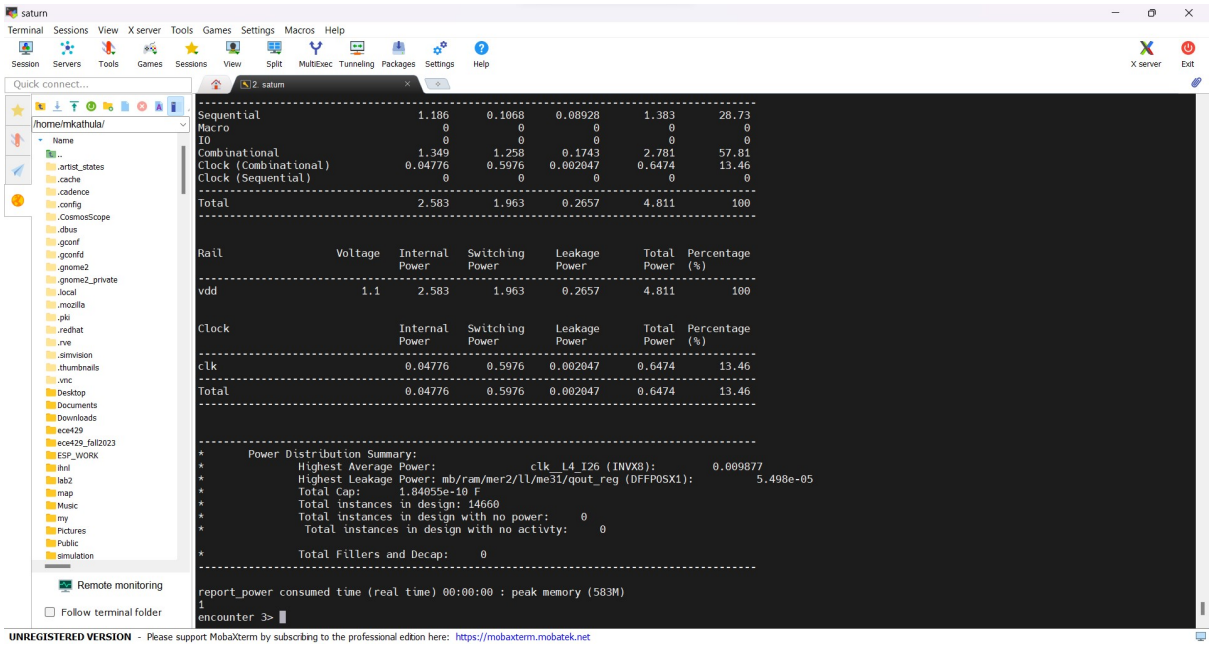


Fig. 103 : Report_Power



```
Total Power
-----------------------------------------------------------------------------------------
Total Internal Power:       2.583      53.69%
Total Switching Power:      1.963      40.79%
Total Leakage Power:      0.2657      5.522%
Total Power:       4.811
-----------------------------------------------------------------------------------------


Group                          Internal   Switching     Leakage       Total  Percentage
                               Power      Power         Power         Power  (%)
-----------------------------------------------------------------------------------------
Sequential                       1.186      0.1068     0.08928       1.383       28.73
Macro                                0           0           0           0           0
IO                                   0           0           0           0           0
Combinational                    1.349       1.258      0.1743       2.781       57.81
Clock (Combinational)          0.04776      0.5976    0.002047      0.6474       13.46
Clock (Sequential)                   0           0           0           0           0
-----------------------------------------------------------------------------------------
Total                            2.583       1.963      0.2657       4.811         100
-----------------------------------------------------------------------------------------


Rail                  Voltage  Internal   Switching     Leakage       Total  Percentage
                               Power      Power         Power         Power  (%)
-----------------------------------------------------------------------------------------
vdd                     1.1      2.583       1.963      0.2657       4.811         100


Clock                          Internal   Switching     Leakage       Total  Percentage
                               Power      Power         Power         Power  (%)
-----------------------------------------------------------------------------------------
clk                            0.04776      0.5976    0.002047      0.6474       13.46
-----------------------------------------------------------------------------------------
Total                          0.04776      0.5976    0.002047      0.6474       13.46
-----------------------------------------------------------------------------------------


-----------------------------------------------------------------------------------------
*       Power Distribution Summary:
*              Highest Average Power:              clk__L4_I26 (INVX8):        0.009877
```

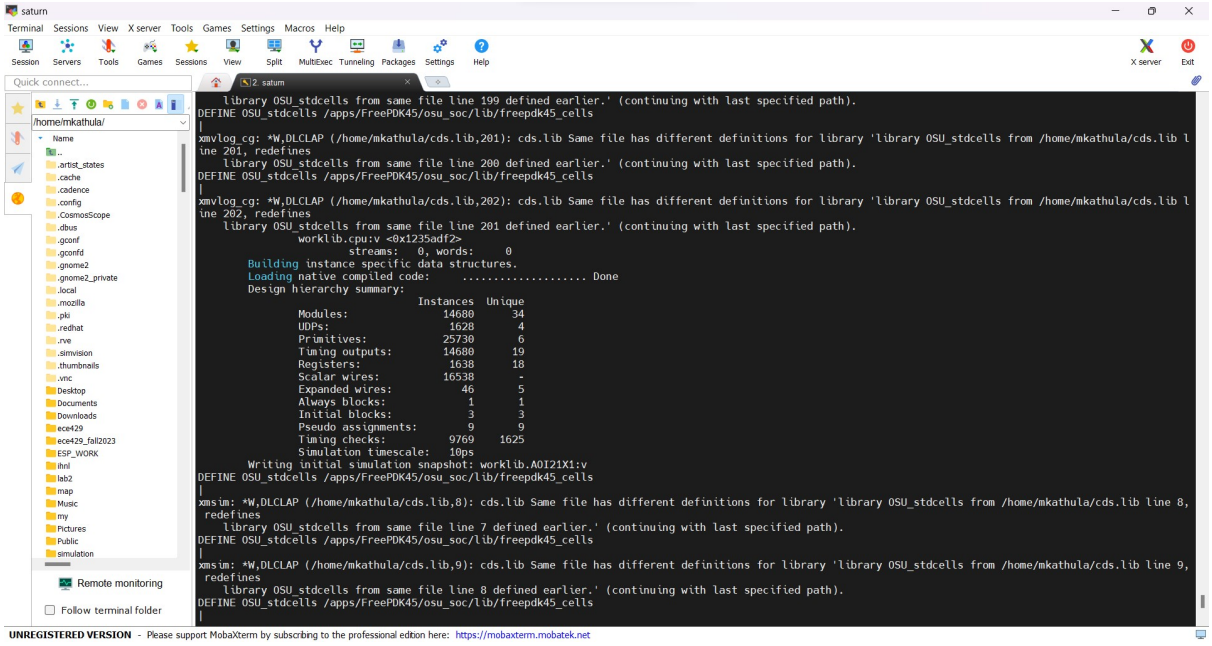
Fig. 104 : Report_Power



```
-----------------------------------------------------------------------------------------
Sequential                       1.186      0.1068     0.08928       1.383       28.73
Macro                                0           0           0           0           0
IO                                   0           0           0           0           0
Combinational                    1.349       1.258      0.1743       2.781       57.81
Clock (Combinational)          0.04776      0.5976    0.002047      0.6474       13.46
Clock (Sequential)                   0           0           0           0           0
-----------------------------------------------------------------------------------------
Total                            2.583       1.963      0.2657       4.811         100
-----------------------------------------------------------------------------------------


Rail                  Voltage  Internal   Switching     Leakage       Total  Percentage
                               Power      Power         Power         Power  (%)
-----------------------------------------------------------------------------------------
vdd                     1.1      2.583       1.963      0.2657       4.811         100


Clock                          Internal   Switching     Leakage       Total  Percentage
                               Power      Power         Power         Power  (%)
-----------------------------------------------------------------------------------------
clk                            0.04776      0.5976    0.002047      0.6474       13.46
-----------------------------------------------------------------------------------------
Total                          0.04776      0.5976    0.002047      0.6474       13.46
-----------------------------------------------------------------------------------------


-----------------------------------------------------------------------------------------
*       Power Distribution Summary:
*              Highest Average Power:              clk__L4_I26 (INVX8):        0.009877
*              Highest Leakage Power: mb/ram/mer2/ll/me31/qout_reg (DFFPOSX1):         5.498e-05
*              Total Cap:      1.84055e-10 F
*              Total instances in design: 14660
*              Total instances in design with no power:     0
*               Total instances in design with no activty:     0

*              Total Fillers and Decap:     0
-----------------------------------------------------------------------------------------

report_power consumed time (real time) 00:00:00 : peak memory (583M)
1
encounter 3>
```

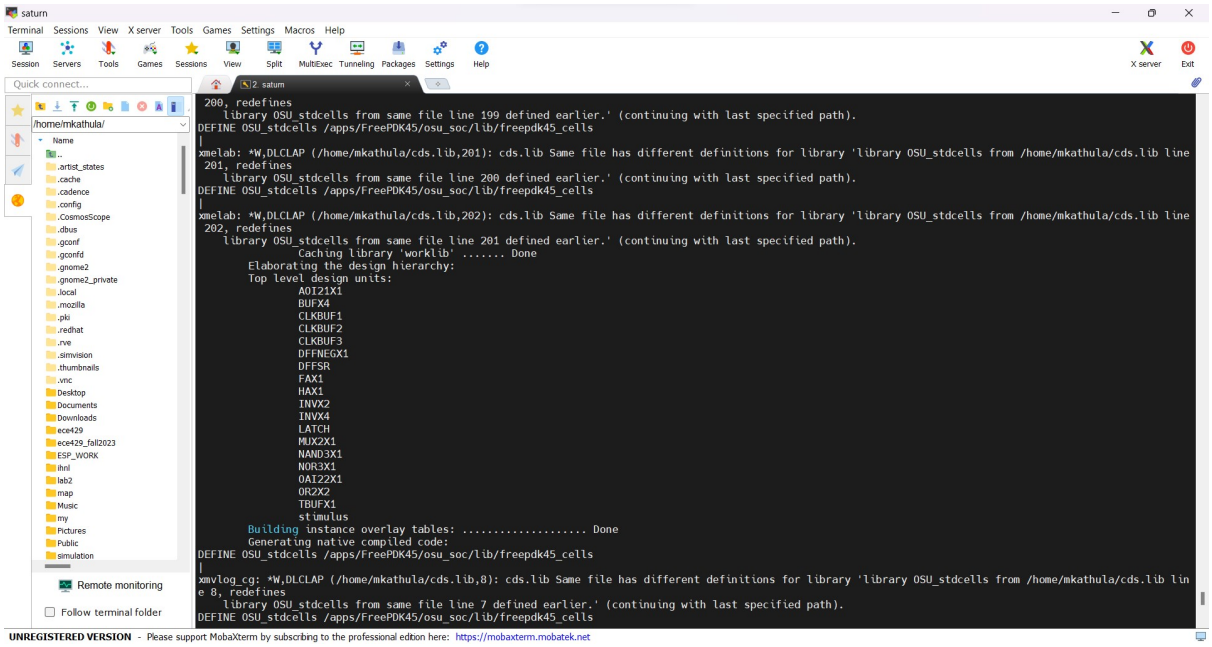
Fig. 105: Report_Power

For Place and Route and Post-P&R Simulation we run “xrun gscl45nm.v tb_cpu.v final.v +access+r”



```
    library OSU_stdcells from same file line 199 defined earlier.' (continuing with last specified path).
DEFINE OSU_stdcells /apps/FreePDK45/osu_soc/lib/freepdk45_cells
|
xmvlog_cg: *W,DLCLAP (/home/mkathula/cds.lib,201): cds.lib Same file has different definitions for library 'library OSU_stdcells from /home/mkathula/cds.lib l
ine 201, redefines
    library OSU_stdcells from same file line 200 defined earlier.' (continuing with last specified path).
DEFINE OSU_stdcells /apps/FreePDK45/osu_soc/lib/freepdk45_cells
|
xmvlog_cg: *W,DLCLAP (/home/mkathula/cds.lib,202): cds.lib Same file has different definitions for library 'library OSU_stdcells from /home/mkathula/cds.lib l
ine 202, redefines
    library OSU_stdcells from same file line 201 defined earlier.' (continuing with last specified path).
                worklib.cpu:v <0x1235adf2>
                        streams:   0, words:     0
        Building instance specific data structures.
        Loading native compiled code:     ..................... Done
        Design hierarchy summary:
                                  Instances  Unique
                Modules:              14680      34
                UDPs:                  1628       4
                Primitives:           25730       6
                Timing outputs:       14680      19
                Registers:             1638      18
                Scalar wires:         16538       -
                Expanded wires:          46       5
                Always blocks:            1       1
                Initial blocks:           3       3
                Pseudo assignments:       9       9
                Timing checks:         9769    1625
                Simulation timescale:   10ps
        Writing initial simulation snapshot: worklib.AOI21X1:v
DEFINE OSU_stdcells /apps/FreePDK45/osu_soc/lib/freepdk45_cells
|
xmsim: *W,DLCLAP (/home/mkathula/cds.lib,8): cds.lib Same file has different definitions for library 'library OSU_stdcells from /home/mkathula/cds.lib line 8,
 redefines
    library OSU_stdcells from same file line 7 defined earlier.' (continuing with last specified path).
DEFINE OSU_stdcells /apps/FreePDK45/osu_soc/lib/freepdk45_cells
|
xmsim: *W,DLCLAP (/home/mkathula/cds.lib,9): cds.lib Same file has different definitions for library 'library OSU_stdcells from /home/mkathula/cds.lib line 9,
 redefines
    library OSU_stdcells from same file line 8 defined earlier.' (continuing with last specified path).
DEFINE OSU_stdcells /apps/FreePDK45/osu_soc/lib/freepdk45_cells
|
```

Fig. 106 :Place & Route and Post-P&R Simulation



```
 200, redefines
    library OSU_stdcells from same file line 199 defined earlier.' (continuing with last specified path).
DEFINE OSU_stdcells /apps/FreePDK45/osu_soc/lib/freepdk45_cells
|
xmelab: *W,DLCLAP (/home/mkathula/cds.lib,201): cds.lib Same file has different definitions for library 'library OSU_stdcells from /home/mkathula/cds.lib line
 201, redefines
    library OSU_stdcells from same file line 200 defined earlier.' (continuing with last specified path).
DEFINE OSU_stdcells /apps/FreePDK45/osu_soc/lib/freepdk45_cells
|
xmelab: *W,DLCLAP (/home/mkathula/cds.lib,202): cds.lib Same file has different definitions for library 'library OSU_stdcells from /home/mkathula/cds.lib line
 202, redefines
    library OSU_stdcells from same file line 201 defined earlier.' (continuing with last specified path).
                Caching library 'worklib' ....... Done
        Elaborating the design hierarchy:
        Top level design units:
                AOI21X1
                BUFX4
                CLKBUF1
                CLKBUF2
                CLKBUF3
                DFFNEGX1
                DFFSR
                FAX1
                HAX1
                INVX2
                INVX4
                LATCH
                MUX2X1
                NAND3X1
                NOR3X1
                OAI22X1
                OR2X2
                TBUFX1
                stimulus
        Building instance overlay tables: ..................... Done
        Generating native compiled code:
DEFINE OSU_stdcells /apps/FreePDK45/osu_soc/lib/freepdk45_cells
|
xmvlog_cg: *W,DLCLAP (/home/mkathula/cds.lib,8): cds.lib Same file has different definitions for library 'library OSU_stdcells from /home/mkathula/cds.lib lin
e 8, redefines
    library OSU_stdcells from same file line 7 defined earlier.' (continuing with last specified path).
DEFINE OSU_stdcells /apps/FreePDK45/osu_soc/lib/freepdk45_cells
```

Fig.107 :Place & Route and Post-P&R Simulation

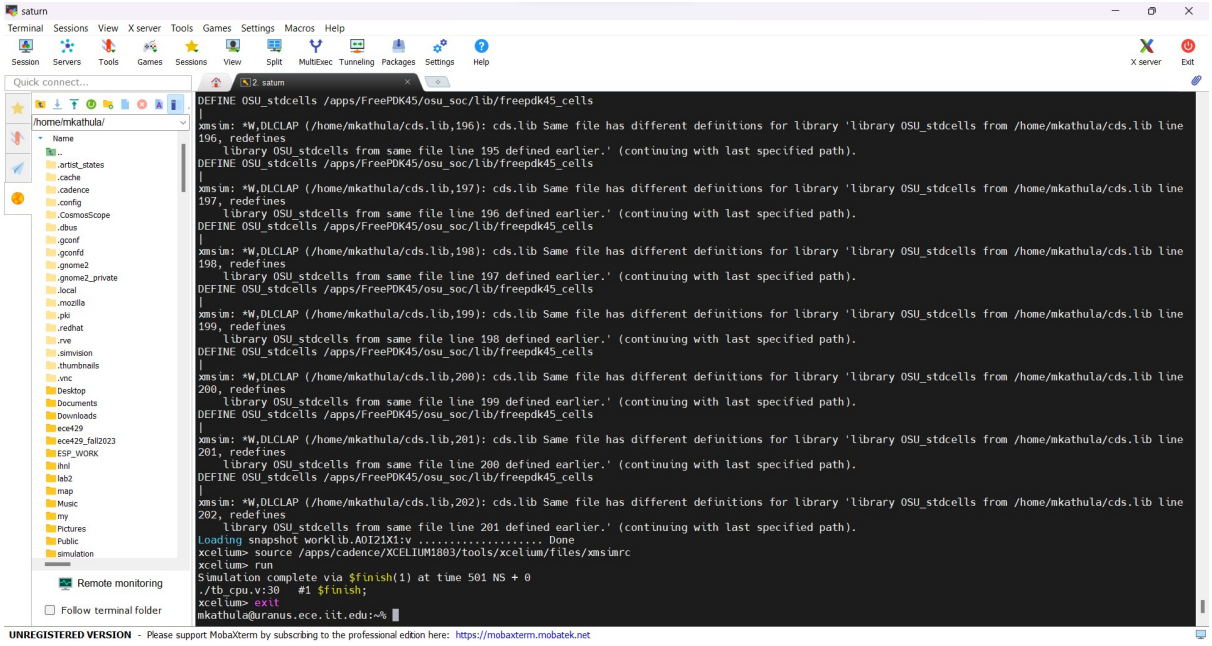


Fig.108 :Place & Route and Post-P&R Simulation

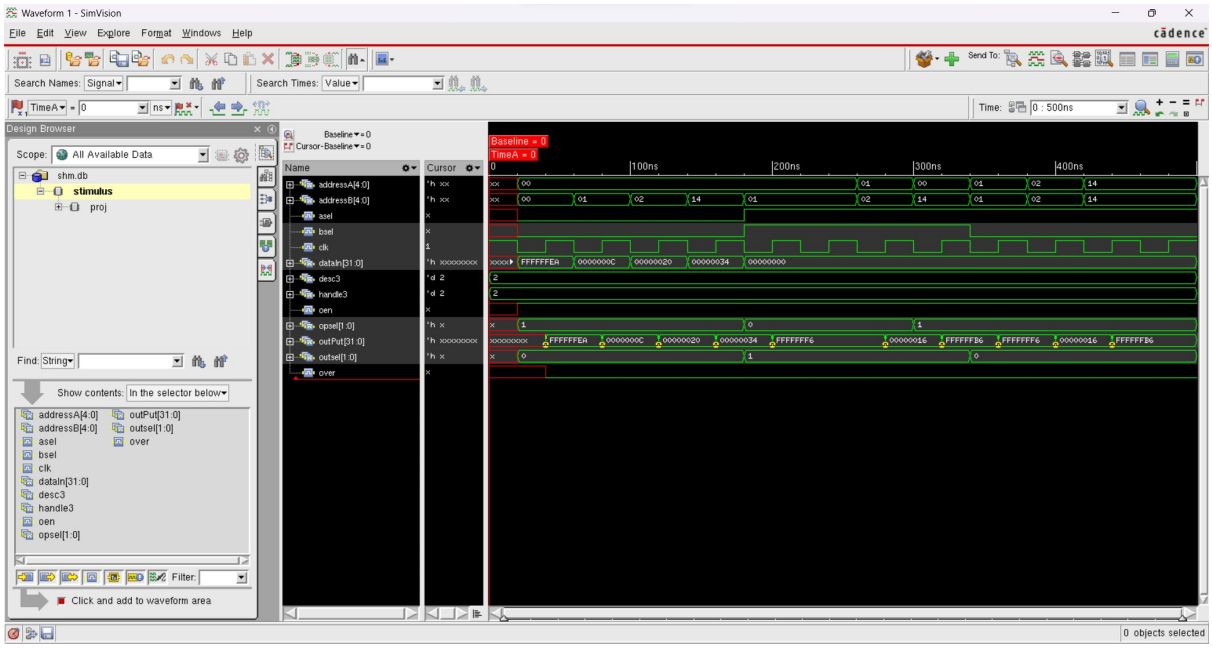


Fig. 109 :Simvision for Place &Route and Post-P&R Simulation

5.4 Maximum Clock Frequency:

This can be obtained in timing.rep.5.final.

Here the slack time has decreased to 18.144, Based on the required time of 28.259 clock frequency is 35.17Mhz(1/28.259).

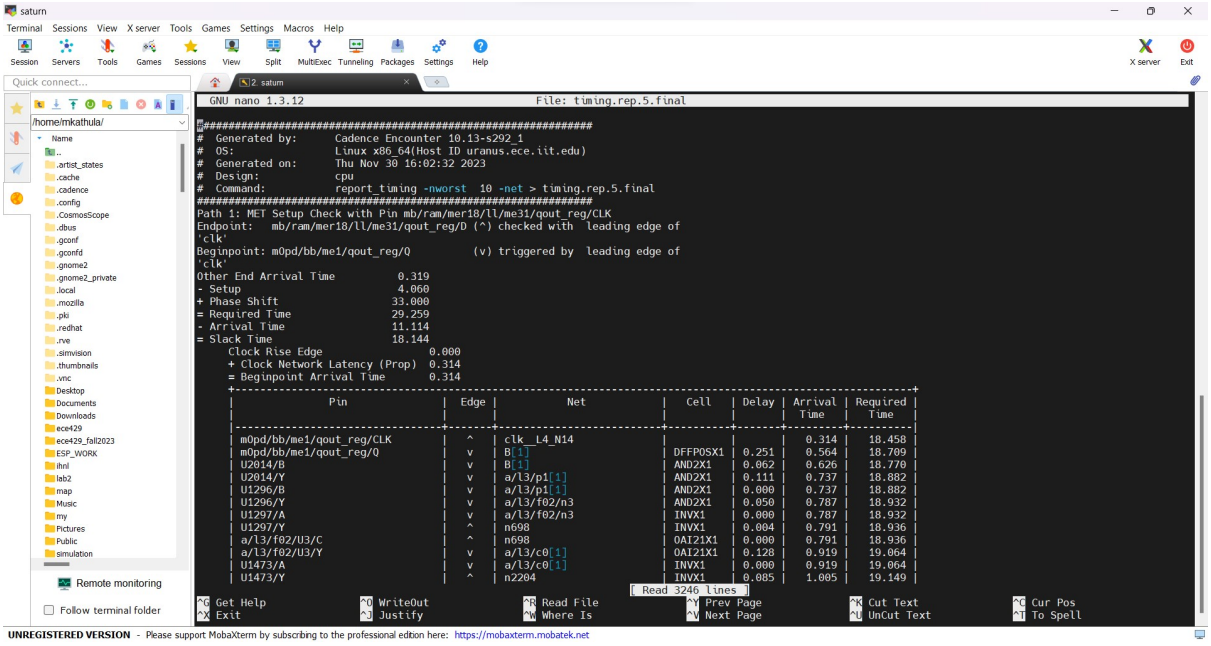


Fig. 110 : Timing.rep.5.final file for Maximum Clock Frquency

5.5 New Test Bench:

The source code of the test bench is given in the appendix. By the tb_test_cpu.v we verify using RTL simulation.

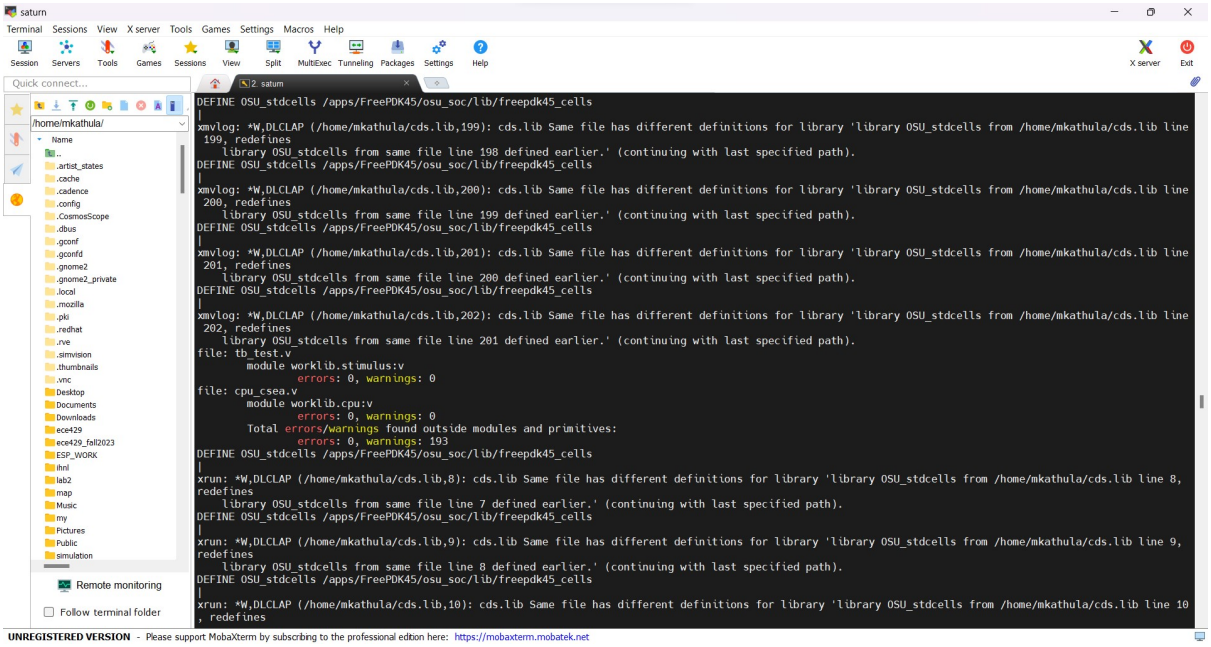


Fig.111 : RTL Simulation for New Test Bench

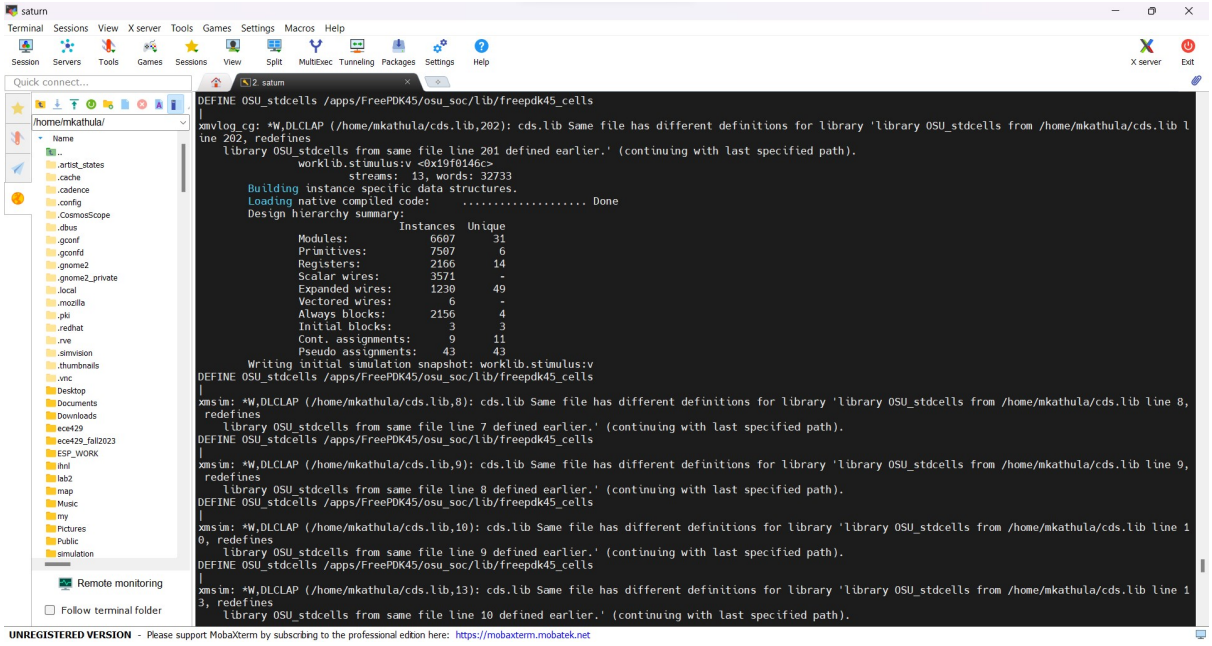


Fig.112 : RTL Simulation for New Test Bench

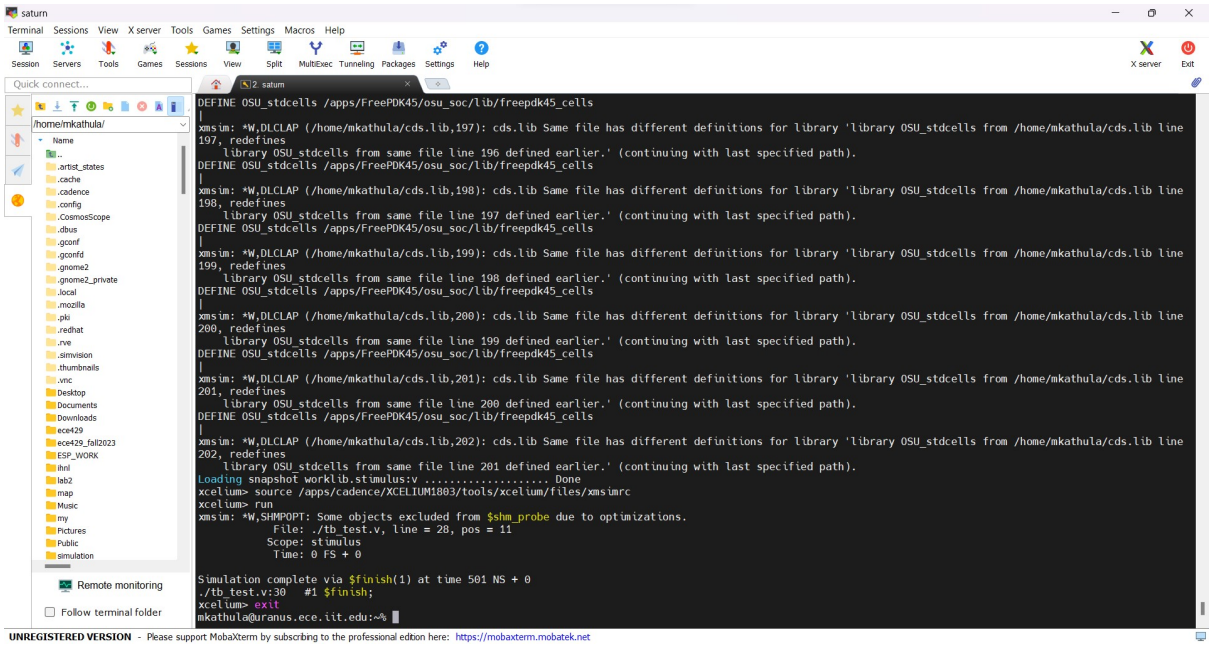


Fig.113 : RTL Simulation for New Test Bench

By using the simvision we will get the waveforms of the new test bench .

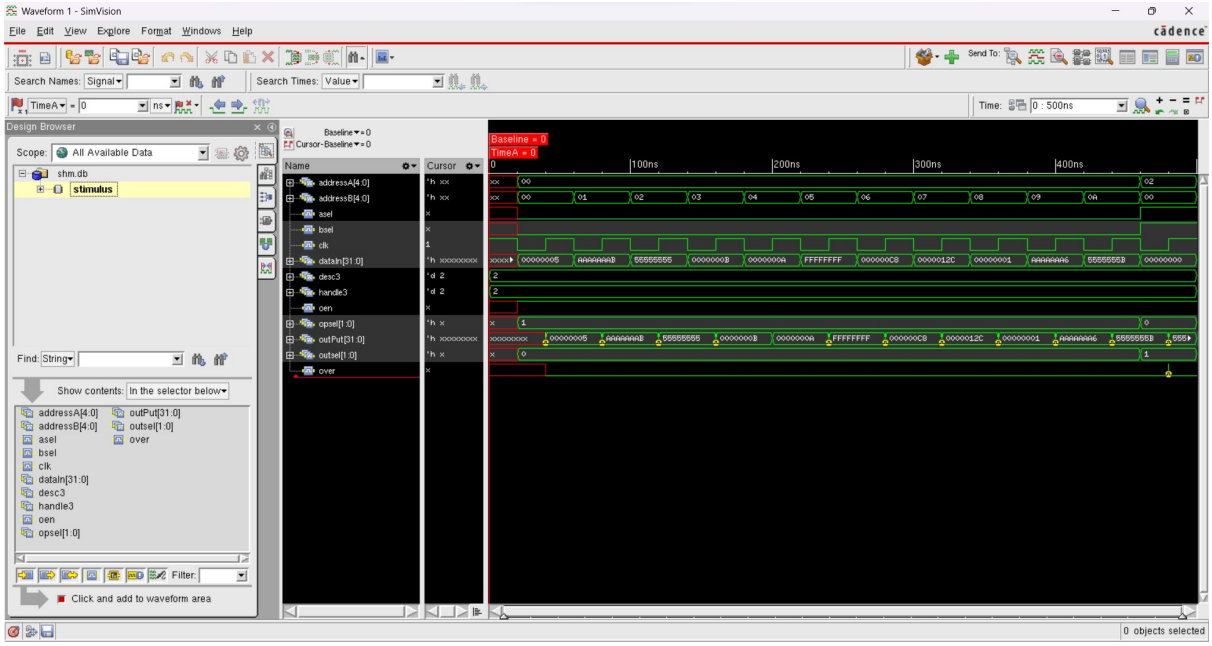


Fig. 114: Simvision Wavefrorm for New Test Bench

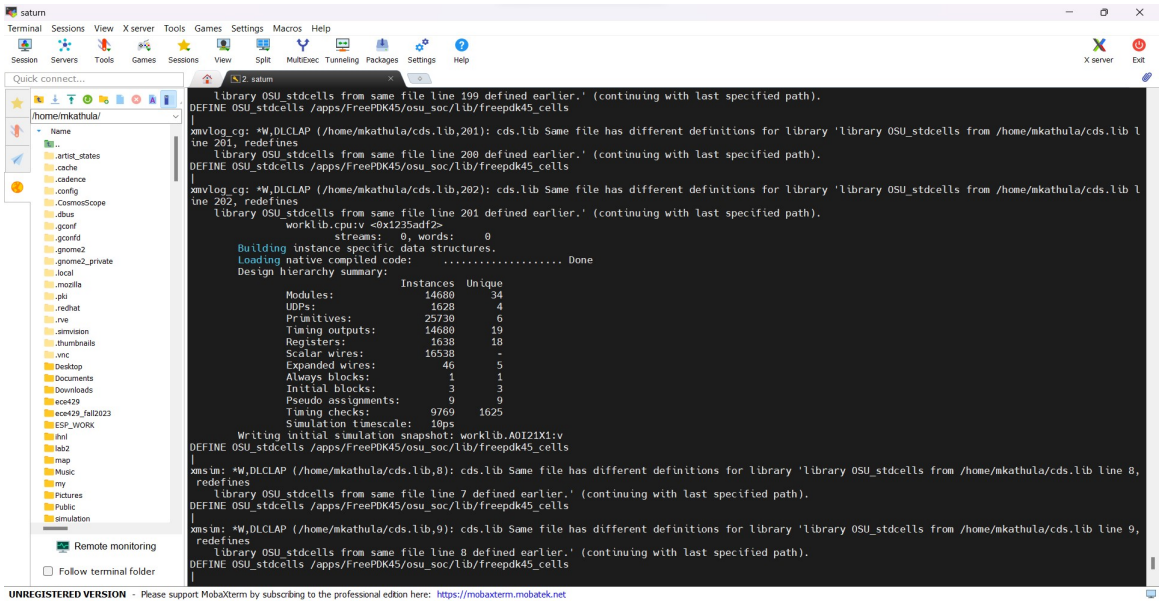


Fig. 115 :Logical and post Synthesis Simulation for new Test Bench

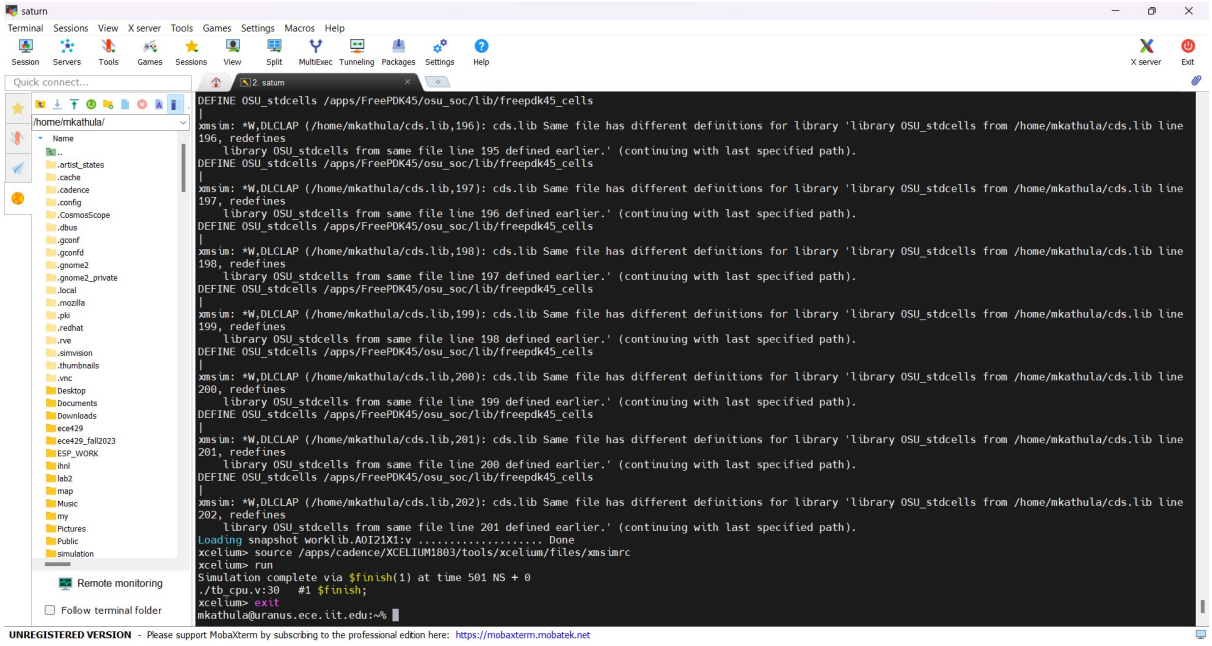


Fig. 116 :Logical and post Synthesis Simulation for new Test Bench

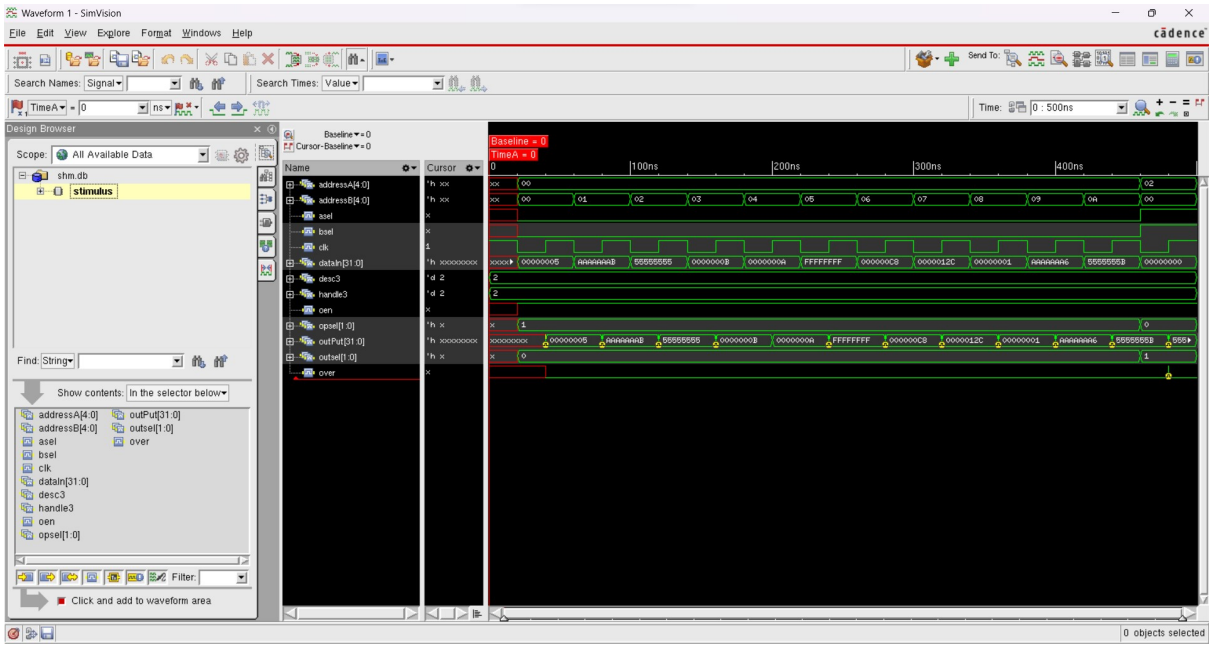


Fig. 117 :Simvision wave form for new test bench(logical and post Synthesis Simulation)

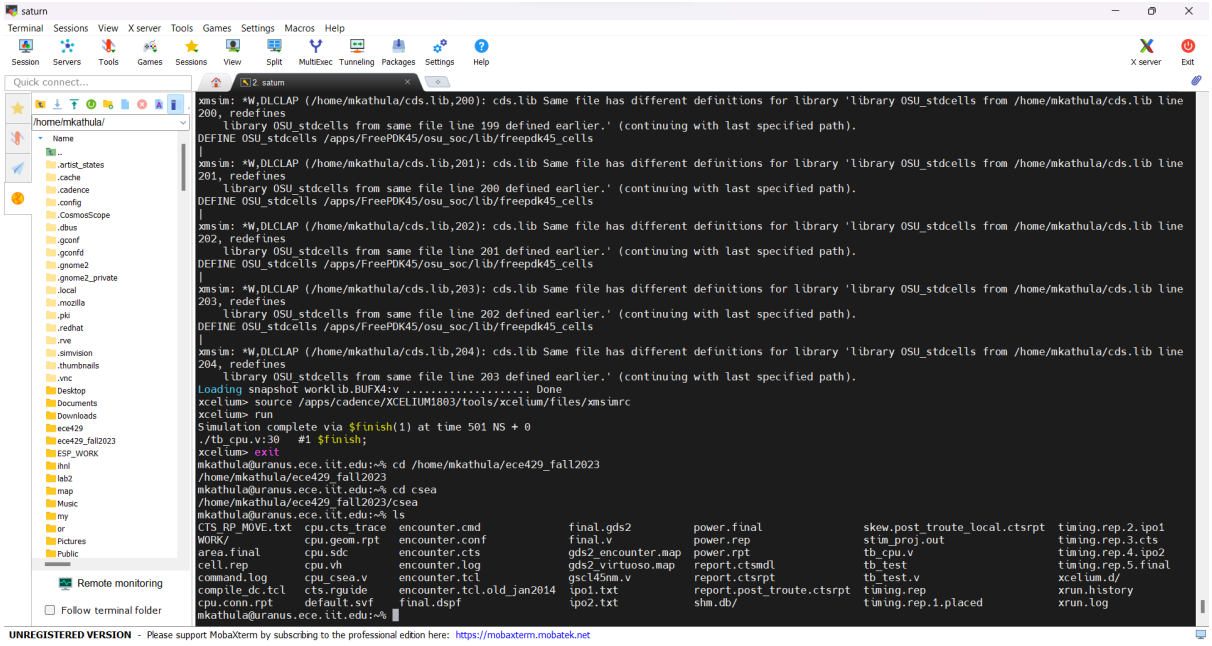


Fig. 118 : Files Created for csea

	Instruction	Stored Data	CRA	CLA	CSA	CSeA
Calculate the path Delay for Each Operation (Post-Synthesis Gate-Level Delay	**ADD[2][0]**	5555_5555+0000_0005	2.12	2.9	2.17	1.21
	ADD[1][2]	AAAA_AAAB+5555_5555	2.88	2.84	2.34	1.63
	ADD[6][7]	0000_00C8+0000_012C	2.74	2.44	2.10	1.68
	ADD[5][8]	FFFF_FFFF+0000_0001	2.54	2.65	2.15	1.69
	SUB[2][3]	5555_5555+0000_000B	2.28	2.64	2.23	2.12
	ADD[3][10]	0000_000B+5555_555B	2.32	2.53	2.28	1.83
	SUB[4][7]	0000_000A+0000_012C	2.35	2.12	1.85	1.62
	ADD[0][8]	0000_0005+0000_0001	2.38	2.40	1.81	1.66

Table 1: Path Delay for Each Operation

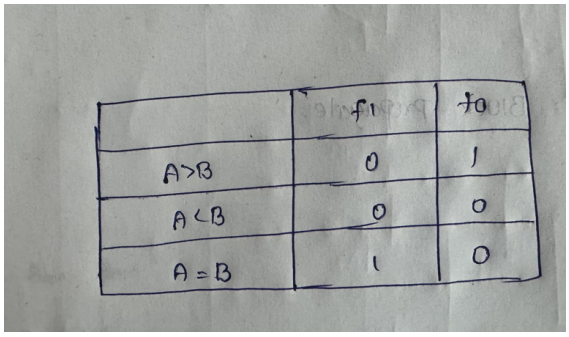
- CSeA take minimum path delay as it has less data to handle and CLA always takes less time to perform operation.

Case Study-2

32-bit CPU Design with New ALU Architecture

6. 32-bit Comparator

The comparator compare the 2 bits operands as follows



	f1	f0
A>B	0	1
A<B	0	0
A=B	1	0

Fig. 119 :Comparator Operation

For the 32-bit comparator we will design using the 4-bit comparator.

It is designed in a tree structure model. In the bottom level which is in the level 2 there are 4 one bit comparators. In which each of them is used to compare the corresponding bit A and B.

The structure of 32-bit comparator is as follows

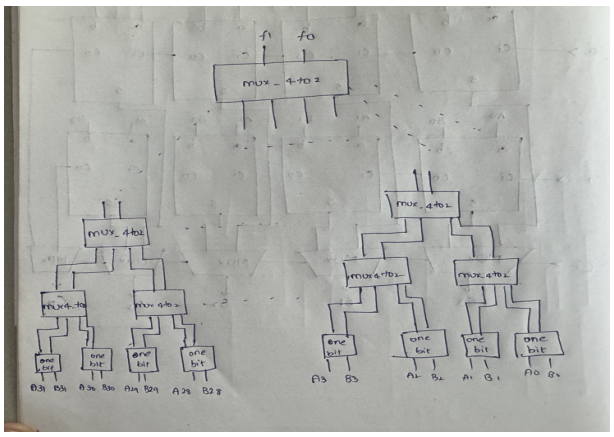


Fig. 120 :Structure of 32-bit comparator

6.1 RTL Simulation:

In this RTL simulation we will verify the logic without delays.

The RTL Simulation is done by using the cpu_cmp.v and tb_test_cmp.v Verilog files.

It can be run by “xrun tb_test_cmp.v cpu_cmp.v +access+r”.

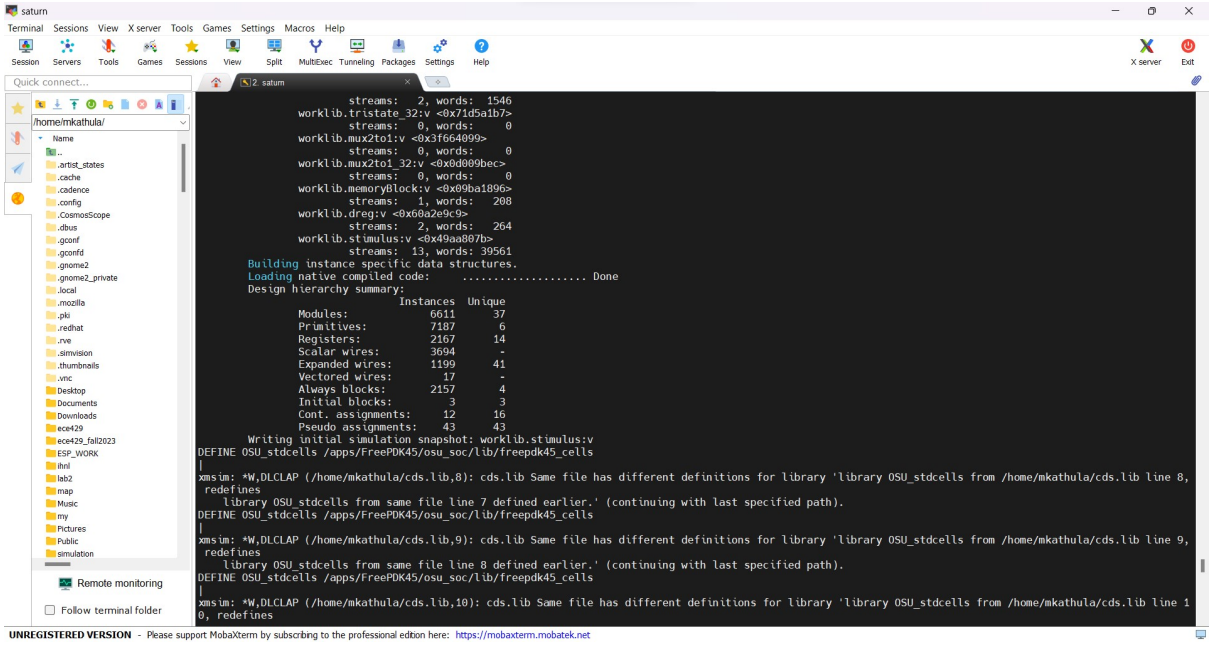


Fig. 121 :RTL Stimulation for Comparator

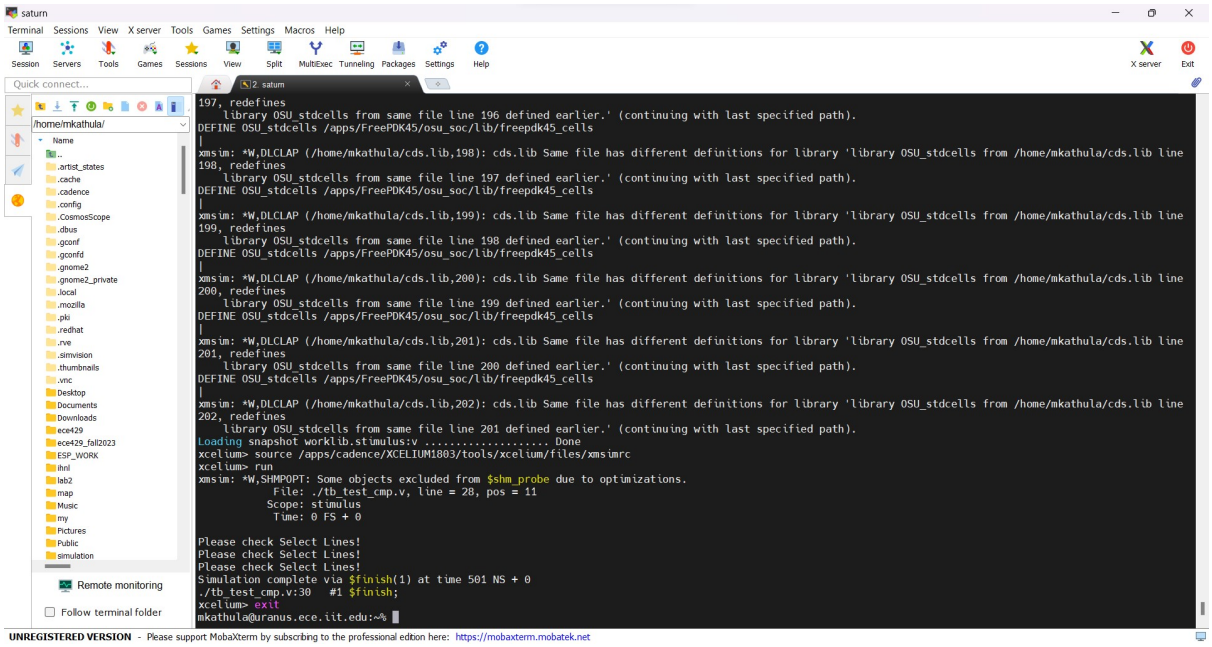


Fig. 122 :RTL Stimulation for Comparator

There are few warnings but there were no errors associated with the Verilog files.

And the wave forms can be seen by using “simvision”.

When the simvison is opened

- Open database
- Click on shm.db
- Right click on shm.db and send to waveform window.

The wave form can be as follows.

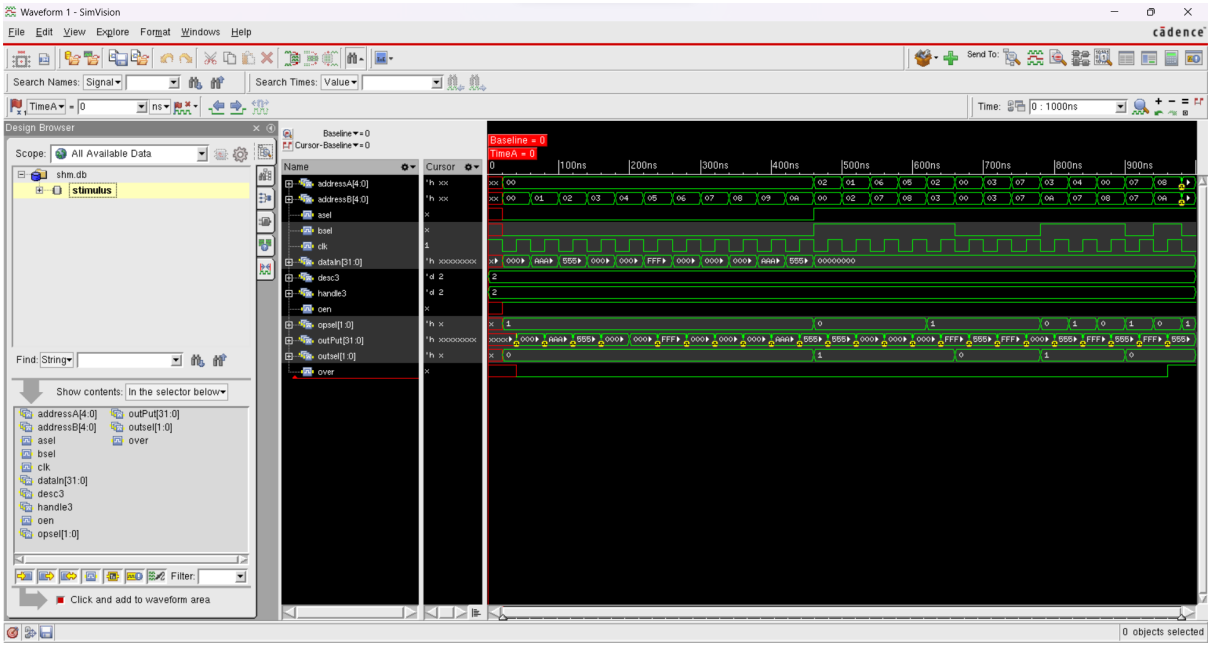


Fig. 123 :Simvision Graph

6.2 Logical Synthesis and Post-Synthesis Simulation:

For this we import the cadence library as

cd /import/scripts/FreePDK45_2011/FreePDK45/osu_soc/flow_ece429/* .

By this we will get the compiling files.

Here we need to change the file name and set my_toplevel to the respective filemename and module name as “[list cpu_CRA.v]” and “cpu”. The frequency must be 30MHz.

For the compile report we will run

“dc_shell -f compile_dc.tcl”

The cell report can be obtained by “nano cell.report”

It is as follows

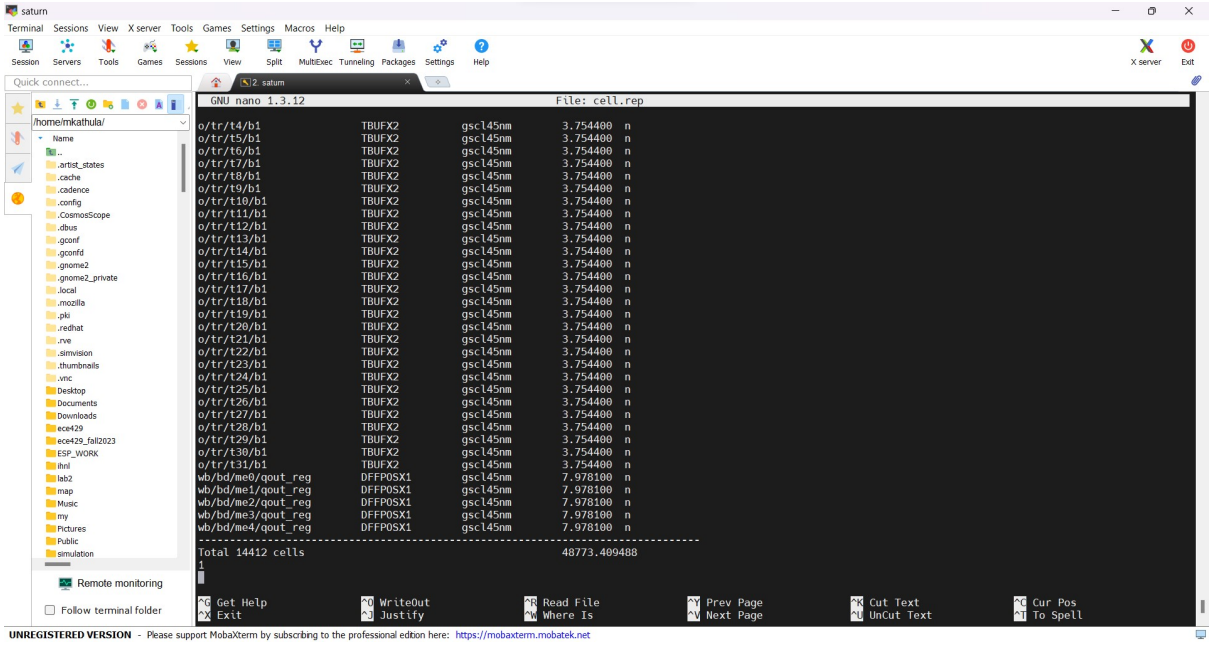


Fig. 124 :Cell Report

The timing report can be obtained by “nano timing.rep”

And it is as follows

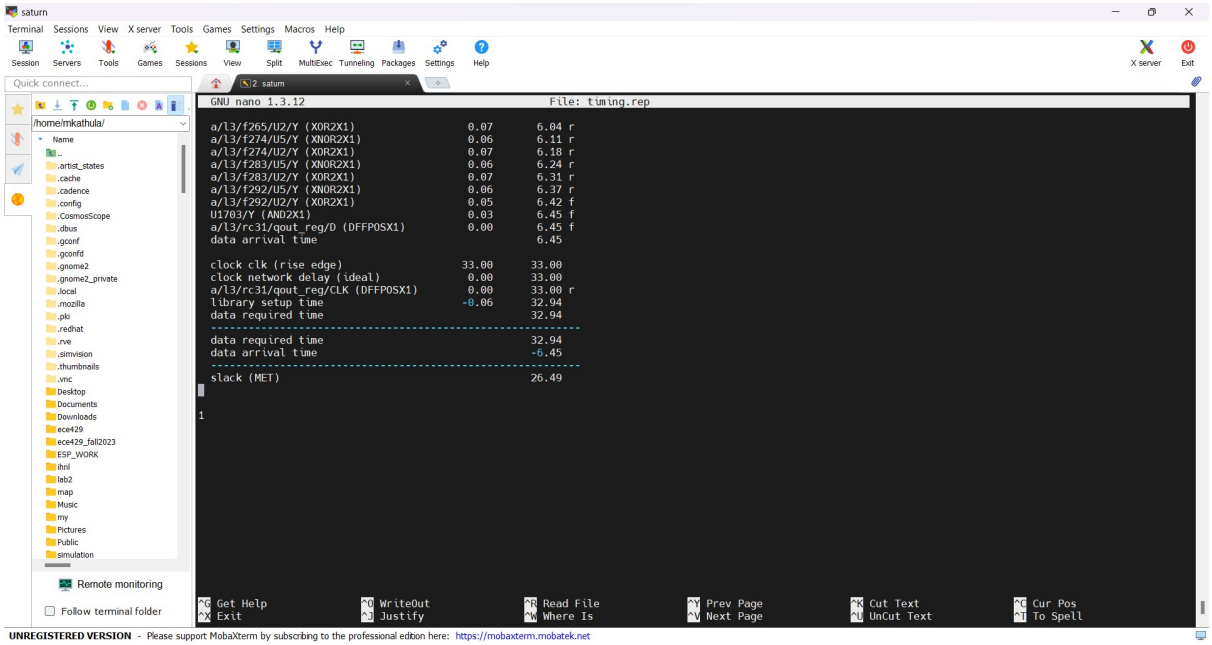


Fig. 125 :Timing Report

The power report can be obtained by “nano power.rep”

It is as follows

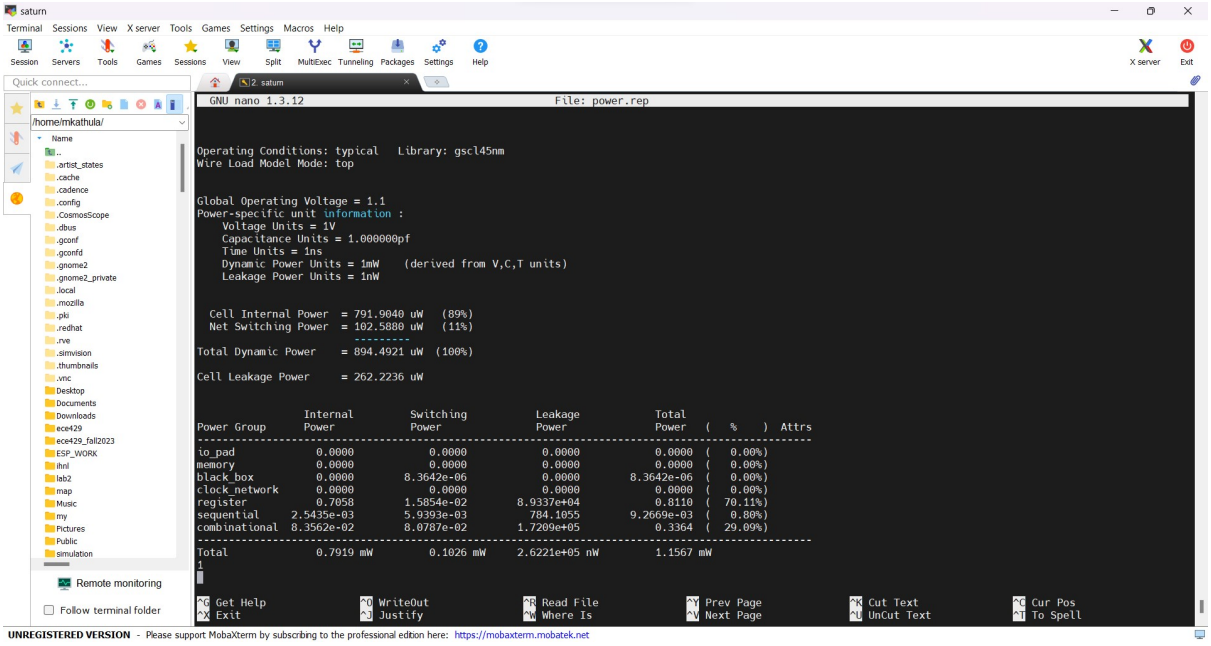


Fig. 126 :Power Report

For the synthesis we will simulate with Verilog models of standard cells,i.e. gscl45nm.v, and we compare the result with rtl simulation.

This can be done by “xrun gscl45nm.v tb_cpu.v cpu.vh +access+r”

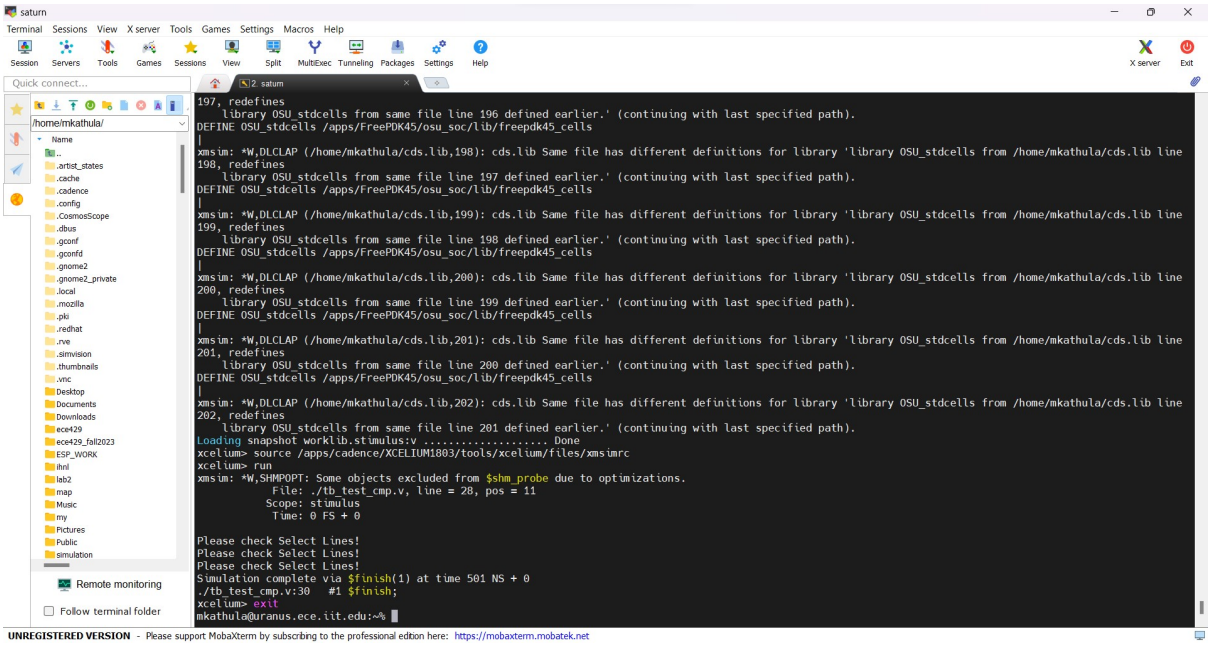


Fig. 127 : Logical and Post Synthesis Stimulation

It shows same as the RTL Simulation.

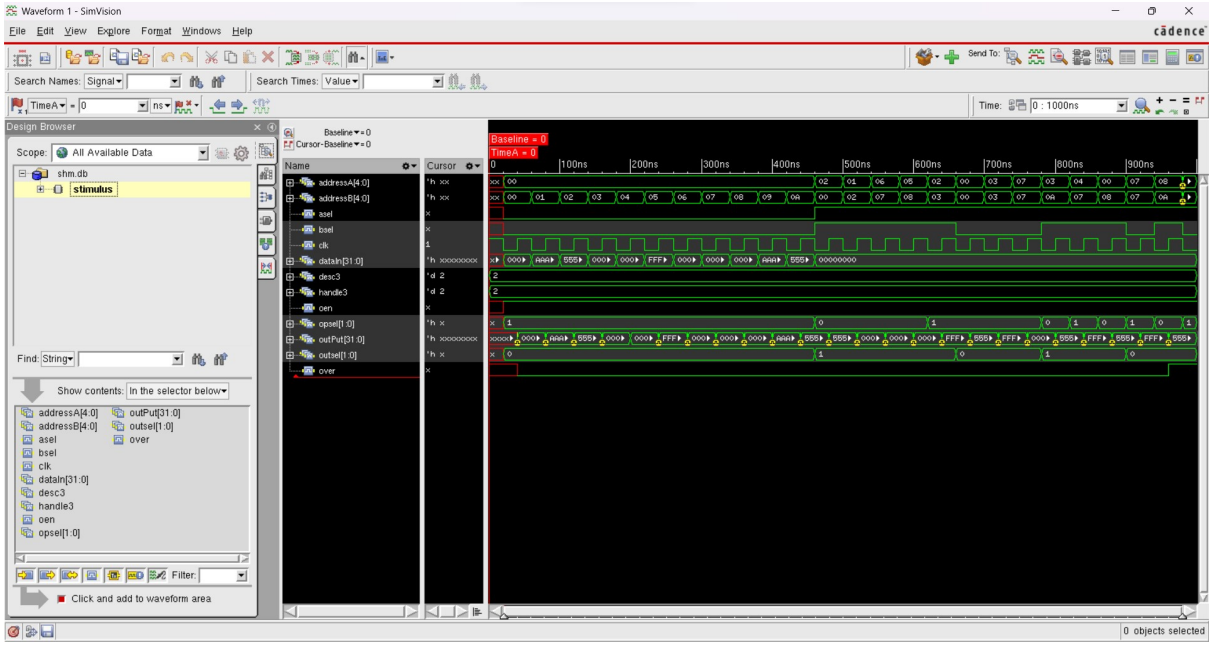


Fig. 128 :Simvision for Logical and Post Synthesis Stimulation

6.3 Place & Route and Post-P&R Simulation:

After the synthesis we will change the set my_toplevel to “cpu” in “encounter.conf”.

Now for the encounter results we run the

“encounter -init encounter.tcl”

For Gate Area we will run “reportGateCount -limit 0”.

It can be as follows

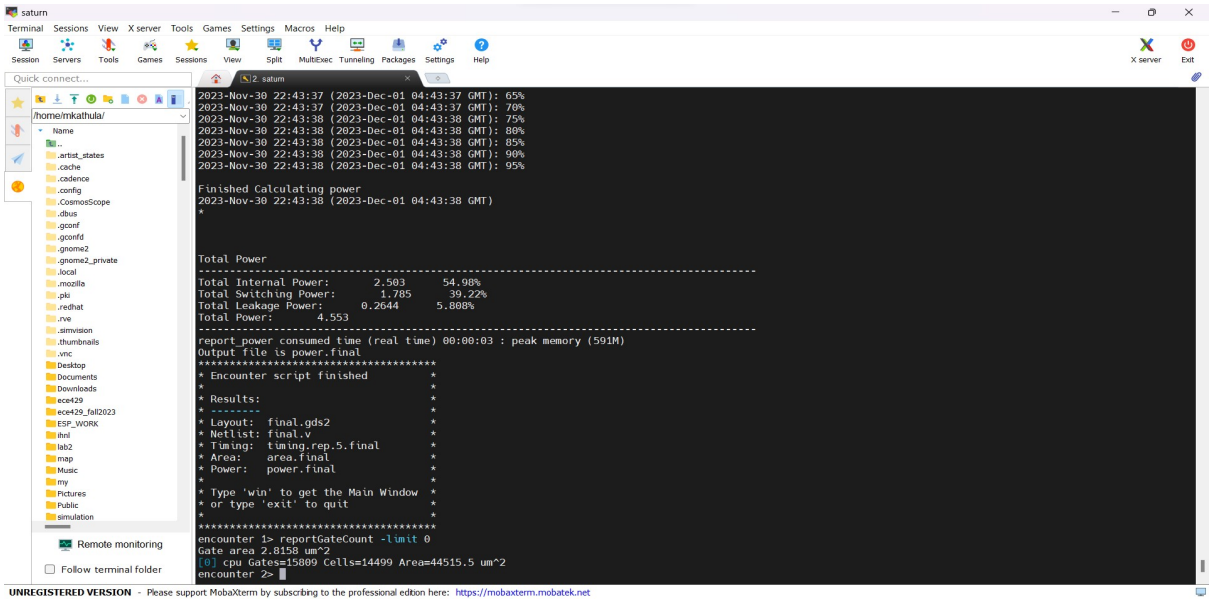


Fig. 129 :Gate Area

The power can be obtained by “report_power”.

It can be as follows

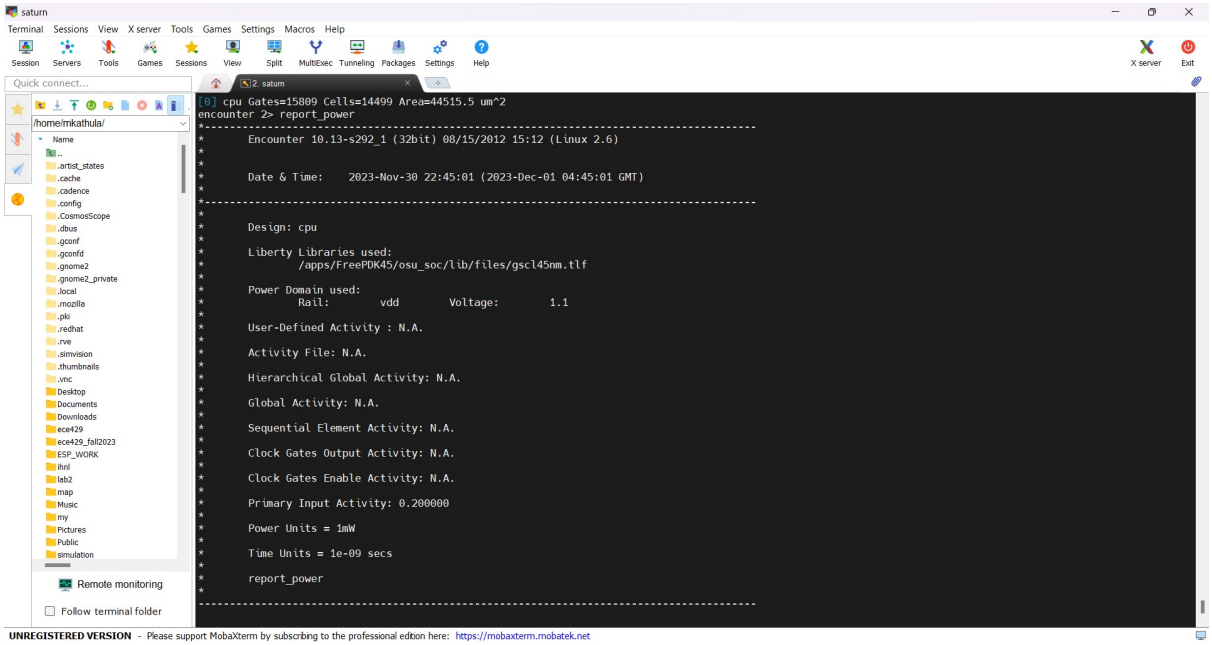


Fig. 130 :Report_Power

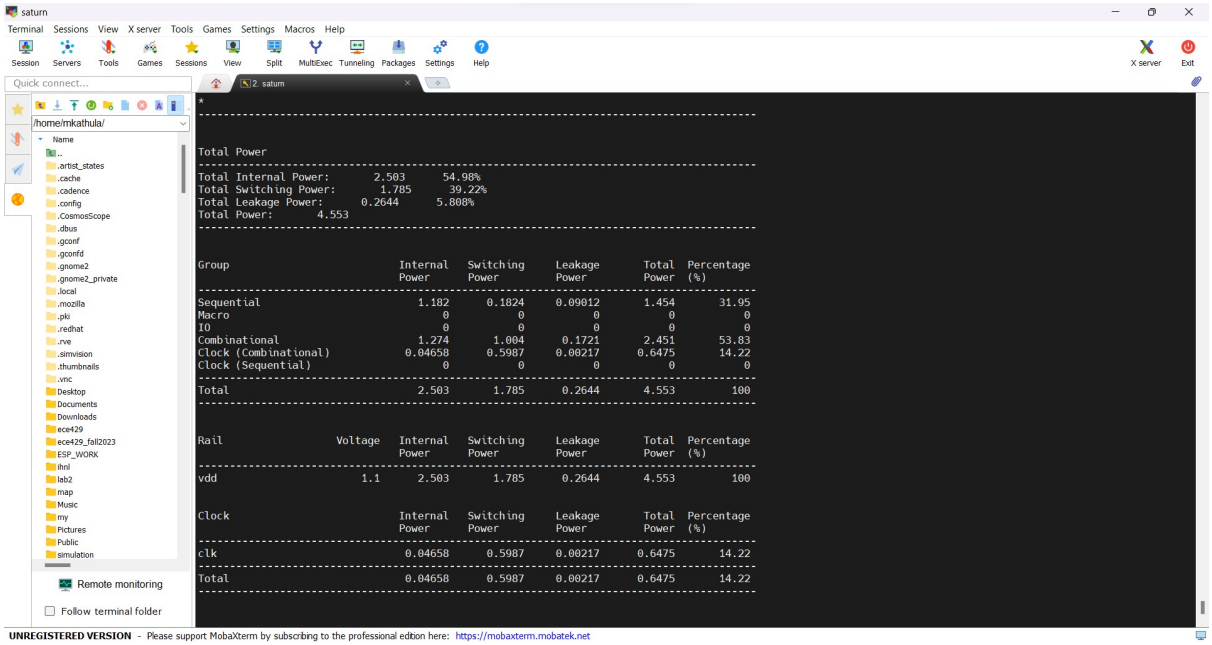


Fig.131 :Report_Power

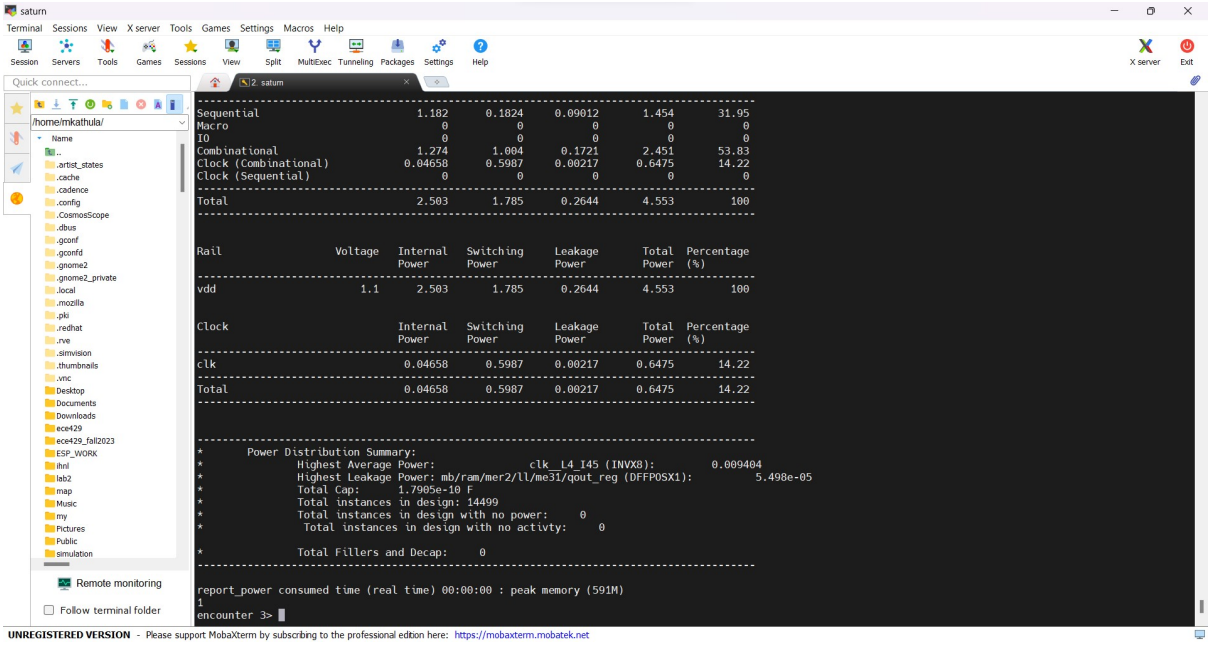


Fig. 132 : Report_Power

For Place and Route and Post-P&R Simulation we run “xrun gscl45nm.v tb_cpu.v final.v +access+r”

It is as follows:

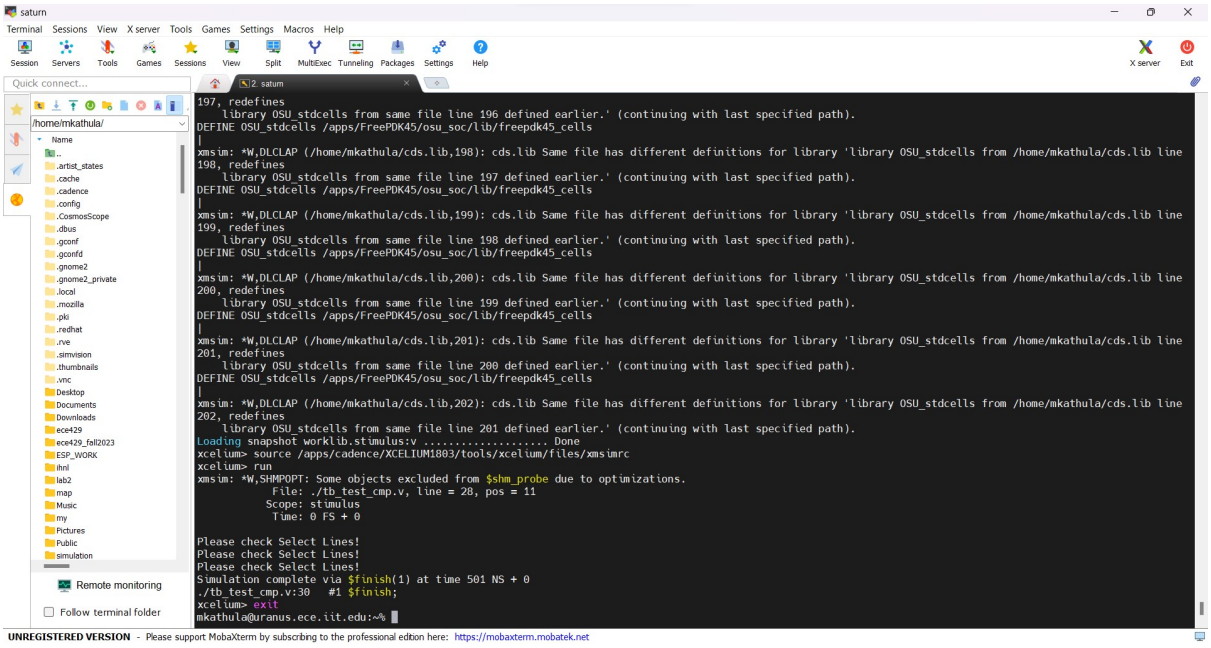


Fig. 133:Place&Route and Post-P&R Simulation

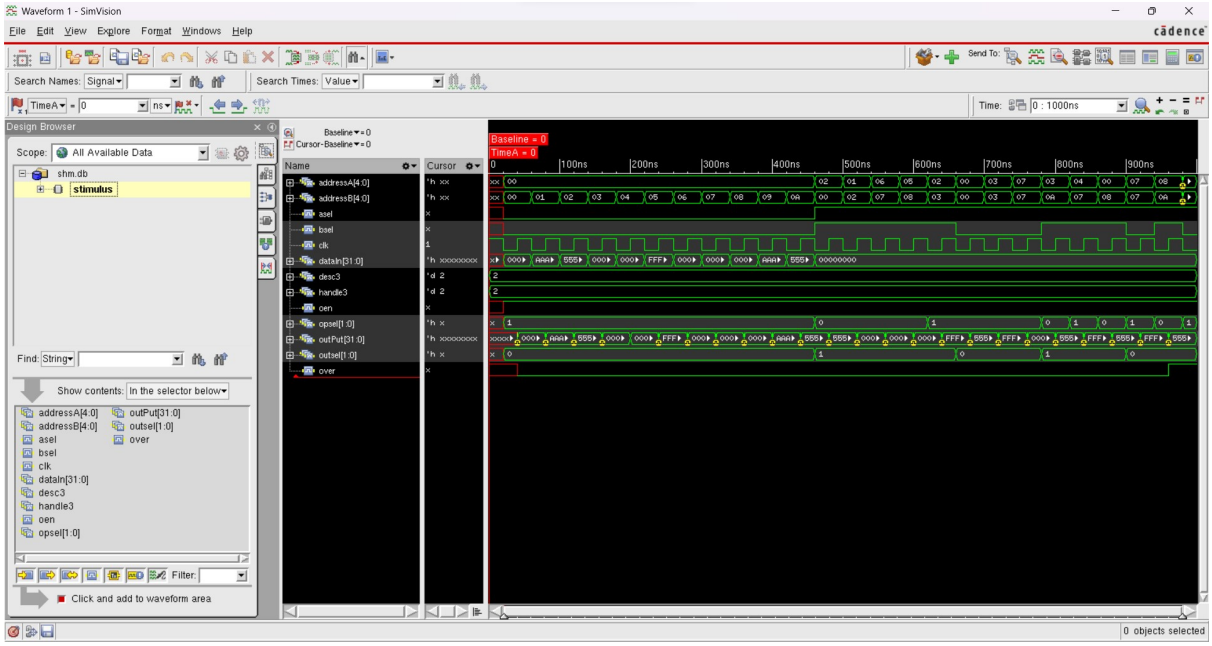


Fig.134 : Simvision for Place &Route and Post-P&R Simulation

6.4 Maximum Clock Frequency:

This can be obtained in timing.rep.5.final.

Here the slack time has decreased to 19.663, Based on the required time of 29.638, max clock frequency is 33.74Mhz(1/29.638).

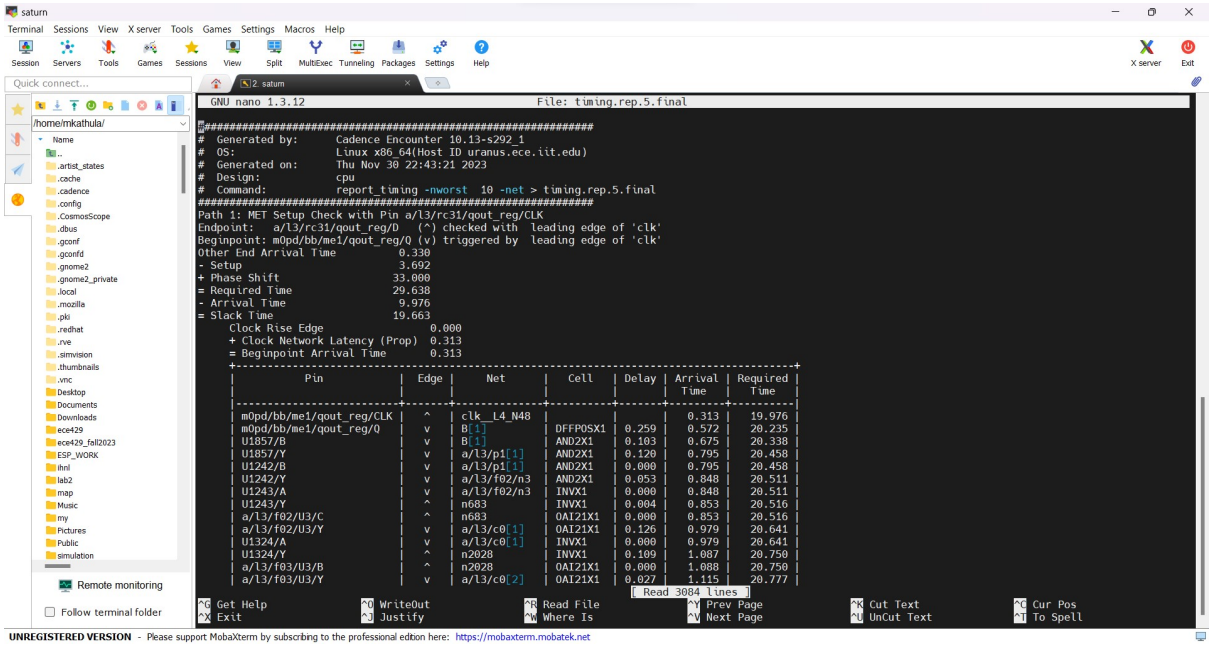


Fig.135 : Maximum Clock Frequency for timing.rep.5.final

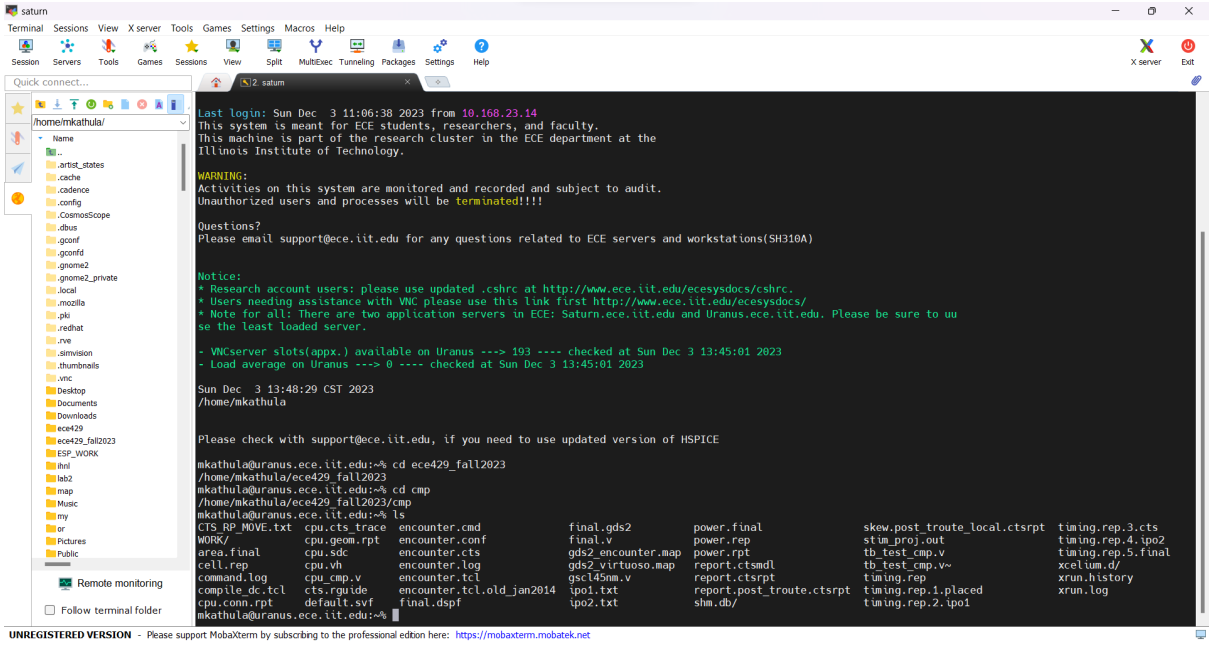


Fig. 136 :Files created for CMP

Source Codes

7.1 The Source code for new test bench of Case 1 :

```
module stimulus;

parameter T = 20;

reg [31:0] dataIn;
reg [1:0] opsel, outsel;
reg [4:0] addressA, addressB;
reg asel, bsel, oen, clk;

wire [31:0] outPut;
wire over;

integer handle3,desc3;

// Instantiate the design block counter
cpu proj(addressA, addressB, dataIn, asel,bsel, clk, opsel, outsel, oen, outPut, over);
initial
begin
clk = 1'b1;
forever #T  clk = ~clk;
end

initial
 begin
handle3 = $fopen ("stim_proj.out");
  $shm_open("shm.db",1);
  $shm_probe(stimulus,"AS");
  #500  $shm_close();
  #1 $finish;
```

```
  end

always
begin
desc3=handle3;
#1 $fdisplay(desc3, $time,"clk=%b, addressA=%d, addressB=%d, dataIn=%d, opsel=%d, outsel=%d, asel=%d,
bsel=%d, oen=%d, OUT=%d", clk, addressA, addressB, dataIn, opsel, outsel,asel, bsel, oen, outPut);

end

 // Stimulate the Input Signals
initial
 begin

// 1. store (5) in [0]

#T

addressA = 5'b00000;

addressB = 5'b00000;

dataIn = 32'h0000_005;  // 5 Hexadecimal value : 0000_0005

opsel = 2'b01;

outsel =2'b00;

asel = 0;

bsel = 0;
```

```
oen = 1;

#(T*2)

// 2. store (2863311531) in [1]

addressA=5'b00000;

addressB= 5'b00001;

dataIn = 32'hAAAA_AAAB;  //  2863311531 Hexadecimal value : AAAA_AAAB

opsel = 2'b01;

outsel = 2'b00;

asel = 0;

bsel = 0;

oen = 1;

#(T*2)

// 3. store 1431655765 in [2]

addressA = 5'b00000;

addressB = 5'b00010;

dataIn = 32'h5555_5555;  // 1431655765 Hexadecimal value :5555_5555
```

```
opsel = 2'b01;

outsel =2'b00;

asel = 0;

bsel = 0;

oen = 1;



#(T*2)

// 4. store 52 in [3]

addressA = 5'b00000;

addressB = 5'b00011; //3 in binary

dataIn = 32'h0000_000B;   //11 Hexadecimal value :0000_000B

opsel = 2'b01;

outsel =2'b00;

asel = 0;

bsel = 0;

oen = 1;
#(T*2)
```

```
// 5. store 10 in [4]

addressA = 5'b00000;

addressB = 5'b00100; //4 in binary

dataIn = 32'h0000_000A;   //10 Hexadecimal value :0000_000A

opsel = 2'b01;

outsel =2'b00;

asel = 0;

bsel = 0;

oen = 1;
#(T*2)
// 6. store 4294967295 in [5]

addressA = 5'b00000;

addressB = 5'b00101; //5 in binary

dataIn = 32'hFFFF_FFFF;   //4294967295 Hexadecimal value :FFFF_FFFF

opsel = 2'b01;

outsel =2'b00;

asel = 0;
```

```
bsel = 0;

oen = 1;
#(T*2)
// 7. store 200 in [6]

addressA = 5'b00000;

addressB = 5'b00110; //6 in binary

dataIn = 32'h0000_00C8;   //200 Hexadecimal value :0000_00C8

opsel = 2'b01;

outsel =2'b00;

asel = 0;

bsel = 0;

oen = 1;

#(T*2)
// 8. store 300 in [7]

addressA = 5'b00000;

addressB = 5'b00111; //7 in binary

dataIn = 32'h0000_012C;   //300 Hexadecimal value :0000_012C

opsel = 2'b01;
```

```
outsel =2'b00;

asel = 0;

bsel = 0;

oen = 1;

#(T*2)
// 9. store 300 in [8]

addressA = 5'b00000;

addressB = 5'b01000; //8 in binary

dataIn = 32'h0000_0001;   //1 Hexadecimal value :0000_0001

opsel = 2'b01;

outsel =2'b00;

asel = 0;

bsel = 0;

oen = 1;

#(T*2)
// 10. store 2863311526 in [9]

addressA = 5'b00000;

addressB = 5'b01001; //9 in binary
```

```
dataIn = 32'hAAAA_AAA6;   //2863311526 Hexadecimal value :AAAA_AAA6

opsel = 2'b01;

outsel =2'b00;

asel = 0;

bsel = 0;

oen = 1;

#(T*2)
// 11. store 1431655771 in [10]

addressA = 5'b00000;

addressB = 5'b01010; //10 in binary

dataIn = 32'h5555_555B;   //1431655771 Hexadecimal value :5555_555B

opsel = 2'b01;

outsel =2'b00;

asel = 0;

bsel = 0;

oen = 1;

#(T*2)
```

```
//12. ADD     [2][0] 1431655765+5=1431655770  5555_555A

addressA = 5'd2;

addressB = 5'd0;

dataIn = 32'd0;

opsel = 2'b00;

outsel =2'b01;

asel = 1;

bsel = 1;

oen = 1;

#(T*2)
//13. ADD     [1][2] 2863311531+1431655765=4294967296 7FFF_FFFF

addressA = 5'd1;

addressB = 5'd2;

dataIn = 32'd0;

opsel = 2'b00;

outsel =2'b01;

asel = 1;
```

```
bsel = 1;

oen = 1;

#(T*2)

//14. ADD      [6][7] = 200+300 =500   0000_01F4
addressA = 5'd6;

addressB = 5'd7;

dataIn = 32'd0;

opsel = 2'b00;

outsel =2'b01;

asel = 1;

bsel = 1;

oen = 1;

#(T*2)
//15. ADD      [5][8] = 4294967295+1 =4294967296   7FFF_FFFF
addressA = 5'd5;

addressB = 5'd8;

dataIn = 32'd0;

opsel = 2'b00;
```

```
outsel =2'b01;

asel = 1;

bsel = 1;

oen = 1;

#(T*2)
//16. SUB      [2][3] = 1431655765-11 =1431655754   5555_554A
addressA = 5'd2;

addressB = 5'd3;

dataIn = 32'd0;

opsel = 2'b01;

outsel =2'b01;

asel = 1;

bsel = 1;

oen = 1;

#(T*2)

//17. READ     [0] = 0000_0005
addressA = 5'd0;

addressB = 5'd0;
```

```
dataIn = 32'd0;

opsel = 2'b01;

outsel =2'b00;

asel = 1;

bsel = 0;

oen = 1;

#(T*2)

//18. READ     [3] = 0000_000B
addressA = 5'd3;

addressB = 5'd3;

dataIn = 32'd0;

opsel = 2'b01;

outsel =2'b00;

asel = 1;

bsel = 0;

oen = 1;

#(T*2)
```

```
//19. READ     [7] = 0000_012C
addressA = 5'd7;

addressB = 5'd7;

dataIn = 32'd0;

opsel = 2'b01;

outsel =2'b00;

asel = 1;

bsel = 0;

oen = 1;

#(T*2)
//20. ADD      [3][10] = 11+1431655771 =1431655782  5555_5566
addressA = 5'd3;

addressB = 5'd10;

dataIn = 32'd0;

opsel = 2'b00;

outsel =2'b01;

asel = 1;

bsel = 1;
```

```
oen = 1;

#(T*2)

//21. SUB      [4][7] = 10-300 =-290   FFFF_FEDE
addressA = 5'd4;

addressB = 5'd7;

dataIn = 32'd0;

opsel = 2'b01;

outsel =2'b01;

asel = 1;

bsel = 1;

oen = 1;

#(T*2)
//22. ADD      [0][8] = 5+1 =6  0000_0006
addressA = 5'd0;

addressB = 5'd8;

dataIn = 32'd0;

opsel = 2'b00;

outsel =2'b01;
```

```
asel = 1;

bsel = 1;

oen = 1;

#(T*2)
//23. READ     [7] = 0000_012C
addressA = 5'd7;

addressB = 5'd7;

dataIn = 32'd0;

opsel = 2'b01;

outsel =2'b00;

asel = 1;

bsel = 0;

oen = 1;

end
endmodule
```

7.2 The source for the test bench of Comparator:

```
module stimulus;

parameter T = 20;

reg [31:0] dataIn;
reg [1:0] opsel, outsel;
reg [4:0] addressA, addressB;
reg asel, bsel, oen, clk;

wire [31:0] outPut;
wire over;

integer handle3,desc3;

// Instantiate the design block counter
cpu proj(addressA, addressB, dataIn, asel,bsel, clk, opsel, outsel, oen, outPut, over);
initial
begin
clk = 1'b1;
forever #T  clk = ~clk;
end

initial
 begin
handle3 = $fopen ("stim_proj.out");
  $shm_open("shm.db",1);
  $shm_probe(stimulus,"AS");
  #1000  $shm_close();
  #1 $finish;
  end
```

```
always

begin

desc3=handle3;

#1 $fdisplay(desc3, $time,"clk=%b, addressA=%d, addressB=%d, dataIn=%d, opsel=%d, outsel=%d, asel=%d,
bsel=%d, oen=%d, OUT=%d", clk, addressA, addressB, dataIn, opsel, outsel,asel, bsel, oen, outPut);


end


 // Stimulate the Input Signals

initial

 begin


// 1. store (5) in [0]


#T


addressA = 5'b00000;


addressB = 5'b00000;


dataIn = 32'h0000_005;  // 5 Hexadecimal value : 0000_0005


opsel = 2'b01;


outsel =2'b00;


asel = 0;


bsel = 0;


oen = 1;
```

```
#(T*2)

// 2. store (2863311531) in [1]

addressA=5'b00000;

addressB= 5'b00001;

dataIn = 32'hAAAA_AAAB;  //  2863311531 Hexadecimal value : AAAA_AAAB

opsel = 2'b01;

outsel = 2'b00;

asel = 0;

bsel = 0;

oen = 1;

#(T*2)

// 3. store 1431655765 in [2]

addressA = 5'b00000;

addressB = 5'b00010;

dataIn = 32'h5555_5555;  // 1431655765 Hexadecimal value :5555_5555
```

```
opsel = 2'b01;

outsel =2'b00;

asel = 0;

bsel = 0;

oen = 1;



#(T*2)

// 4. store 52 in [3]

addressA = 5'b00000;

addressB = 5'b00011; //3 in binary

dataIn = 32'h0000_000B;   //11 Hexadecimal value :0000_000B

opsel = 2'b01;

outsel =2'b00;

asel = 0;

bsel = 0;

oen = 1;
#(T*2)
```

```
// 5. store 10 in [4]

addressA = 5'b00000;

addressB = 5'b00100; //4 in binary

dataIn = 32'h0000_000A;   //10 Hexadecimal value :0000_000A

opsel = 2'b01;

outsel =2'b00;

asel = 0;

bsel = 0;

oen = 1;
#(T*2)
// 6. store 4294967295 in [5]

addressA = 5'b00000;

addressB = 5'b00101; //5 in binary

dataIn = 32'hFFFF_FFFF;   //4294967295 Hexadecimal value :FFFF_FFFF

opsel = 2'b01;

outsel =2'b00;

asel = 0;

bsel = 0;
```

```
oen = 1;
#(T*2)
// 7. store 200 in [6]

addressA = 5'b00000;

addressB = 5'b00110; //6 in binary

dataIn = 32'h0000_00C8;   //200 Hexadecimal value :0000_00C8

opsel = 2'b01;

outsel =2'b00;

asel = 0;

bsel = 0;

oen = 1;

#(T*2)
// 8. store 300 in [7]

addressA = 5'b00000;

addressB = 5'b00111; //7 in binary

dataIn = 32'h0000_012C;   //300 Hexadecimal value :0000_012C

opsel = 2'b01;

outsel =2'b00;
```

```
asel = 0;

bsel = 0;

oen = 1;

#(T*2)
// 9. store 300 in [8]

addressA = 5'b00000;

addressB = 5'b01000; //8 in binary

dataIn = 32'h0000_0001;   //1 Hexadecimal value :0000_0001

opsel = 2'b01;

outsel =2'b00;

asel = 0;

bsel = 0;

oen = 1;

#(T*2)
// 10. store 2863311526 in [9]

addressA = 5'b00000;

addressB = 5'b01001; //9 in binary
```

```
dataIn = 32'hAAAA_AAA6;   //2863311526 Hexadecimal value :AAAA_AAA6

opsel = 2'b01;

outsel =2'b00;

asel = 0;

bsel = 0;

oen = 1;

#(T*2)
// 11. store 1431655771 in [10]

addressA = 5'b00000;

addressB = 5'b01010; //10 in binary

dataIn = 32'h5555_555B;   //1431655771 Hexadecimal value :5555_555B

opsel = 2'b01;

outsel =2'b00;

asel = 0;

bsel = 0;

oen = 1;

#(T*2)
```

```
//12. ADD     [2][0] 1431655765+5=1431655770  5555_555A

addressA = 5'd2;

addressB = 5'd0;

dataIn = 32'd0;

opsel = 2'b00;

outsel =2'b01;

asel = 1;

bsel = 1;

oen = 1;

#(T*2)
//13. ADD     [1][2] 2863311531+1431655765=4294967296 7FFF_FFFF

addressA = 5'd1;

addressB = 5'd2;

dataIn = 32'd0;

opsel = 2'b00;

outsel =2'b01;

asel = 1;
```

```
bsel = 1;

oen = 1;

#(T*2)

//14. ADD      [6][7] = 200+300 =500   0000_01F4
addressA = 5'd6;

addressB = 5'd7;

dataIn = 32'd0;

opsel = 2'b00;

outsel =2'b01;

asel = 1;

bsel = 1;

oen = 1;

#(T*2)
//15. ADD      [5][8] = 4294967295+1 =4294967296   7FFF_FFFF
addressA = 5'd5;

addressB = 5'd8;

dataIn = 32'd0;

opsel = 2'b00;
```

```
outsel =2'b01;

asel = 1;

bsel = 1;

oen = 1;

#(T*2)
//16. SUB      [2][3] = 1431655765-11 =1431655754   5555_554A
addressA = 5'd2;

addressB = 5'd3;

dataIn = 32'd0;

opsel = 2'b01;

outsel =2'b01;

asel = 1;

bsel = 1;

oen = 1;

#(T*2)

//17. READ     [0] = 0000_0005
addressA = 5'd0;

addressB = 5'd0;
```

```
dataIn = 32'd0;

opsel = 2'b01;

outsel =2'b00;

asel = 1;

bsel = 0;

oen = 1;

#(T*2)

//18. READ     [3] = 0000_000B
addressA = 5'd3;

addressB = 5'd3;

dataIn = 32'd0;

opsel = 2'b01;

outsel =2'b00;

asel = 1;

bsel = 0;

oen = 1;

#(T*2)
//19. READ     [7] = 0000_012C
```

```
addressA = 5'd7;

addressB = 5'd7;

dataIn = 32'd0;

opsel = 2'b01;

outsel =2'b00;

asel = 1;

bsel = 0;

oen = 1;

#(T*2)
//20. ADD      [3][10] = 11+1431655771 =1431655782  5555_5566
addressA = 5'd3;

addressB = 5'd10;

dataIn = 32'd0;

opsel = 2'b00;

outsel =2'b01;

asel = 1;

bsel = 1;

oen = 1;
```

```
#(T*2)

//21. SUB      [4][7] = 10-300 =-290   FFFF_FEDE
addressA = 5'd4;

addressB = 5'd7;

dataIn = 32'd0;

opsel = 2'b01;

outsel =2'b01;

asel = 1;

bsel = 1;

oen = 1;

#(T*2)
//22. ADD      [0][8] = 5+1 =6  0000_0006
addressA = 5'd0;

addressB = 5'd8;

dataIn = 32'd0;

opsel = 2'b00;

outsel =2'b01;

asel = 1;
```

```
bsel = 1;

oen = 1;

#(T*2)
//23. READ     [7] = 0000_012C
addressA = 5'd7;

addressB = 5'd7;

dataIn = 32'd0;

opsel = 2'b01;

outsel =2'b00;

asel = 1;

bsel = 0;

oen = 1;
#(T*2)
// 24. CMP [8][10]
addressA = 5'd8;
addressB = 5'd10;
dataIn = 32'd0;
opsel = 2'b00;
outsel =2'b100;
asel = 1;
bsel = 1;
oen = 1;
#(T*2)
```

```
// 25. CMP [4][2]
addressA = 5'd4;
addressB = 5'd2;
dataIn = 32'd0;
opsel = 2'b00;
outsel =2'b100;
asel = 1;
bsel = 1;
oen = 1;
// 26. CMP [8][7]
addressA = 5'd8;
addressB = 5'd7;
dataIn = 32'd0;
opsel = 2'b00;
outsel =2'b100;
asel = 1;
bsel = 1;
oen = 1;
//27. READ     [10] = 5555_555B
addressA = 5'd10;

addressB = 5'd10;

dataIn = 32'd0;

opsel = 2'b01;

outsel =2'b00;

asel = 1;

bsel = 0;
```

```
oen = 1;
#(T*2)
//28. READ     [7] = 0000_012C
addressA = 5'd7;

addressB = 5'd7;

dataIn = 32'd0;

opsel = 2'b01;

outsel =2'b00;

asel = 1;

bsel = 0;

oen = 1;
#(T*2)
//29. READ     [2] = 5555_5555
addressA = 5'd2;

addressB = 5'd2;

dataIn = 32'd0;

opsel = 2'b01;

outsel =2'b00;

asel = 1;

bsel = 0;
```

```
oen = 1;


end

endmodule
```

3) Conclusion:

From this project we have studied about the 32-bit pipelined CPU. By using different adders like CRA(Carry Ripple Adder),CLA(Carry Look Ahead Adder), CSA(Carry Skip Adder), CSeA(Carry Select Adder) we will study and we will find the delay for each of the operation using simvision.