MANIPAL
ACADEMY of HIGHER EDUCATION
(Institution of Eminence Deemed to be University)

``

MANIPAL SCHOOL OF INFORMATION SCIENCES

(A Constituent unit of MAHE, Manipal)

UVM based Verification & Physical Design of APB bridge

Reg. Number	Name	Branch
221038018	Vaishnavi Holla	VLSI Design
221038023	Abhilash N	VLSI Design
221038024	Nithin S	VLSI Design

Under the guidance of

Dr. Madhushankara M
Associate Professor,
Manipal School of Information Sciences,
MAHE, MANIPAL

08/05/2023

Table of Contents

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS

APB	Advance Peripheral Bus
CPU	Central Processing Unit
ARM	Advanced RISC machine
SoC	System-on-Chip
DSP	Digital Signal Processing
AHB	Advanced High-performance Bus
AXI	Advanced eXtensible Interface
RTL	Register Transfer Logic
PADDR	Peripheral Address
PSEL	Peripheral Select
PSLVERR	Peripheral Slave Error
PRDATA	Peripheral Read Data
PWDATA	Peripheral Write Data
GUI	Graphical User Interface
CTS	Clock Tree Synthesis
QoR	Quality of Results

1. Introduction

APB (Advanced Peripheral Bus) is a protocol used for connecting peripheral devices to a central processing unit (CPU) in embedded systems. It was developed by ARM Holdings, a leading technology company that specializes in producing microprocessors and embedded systems. The APB protocol provides a simple and efficient way to interface various peripherals with the CPU, and is widely used in many embedded systems, including microcontrollers, system-on-chip (SoC) designs, and digital signal processors (DSPs).

The APB protocol is a bus-based architecture that uses a shared set of wires to transmit data, commands, and control signals between the CPU and the peripheral devices. The protocol defines a set of rules and procedures for controlling the bus and ensuring reliable data transfer. It also specifies the data format and timing requirements for each transfer, including the addressing scheme, data size, and transfer mode.

One of the key advantages of the APB protocol is its simplicity and ease of implementation. The protocol has a minimalistic design that allows it to be easily integrated into various system designs, while still providing efficient data transfer and control. It also supports multiple devices on a single bus, enabling the integration of multiple peripherals with the CPU.

The APB protocol is also highly configurable, allowing designers to adjust the protocol parameters to optimize performance and power consumption for their specific system requirements. These parameters include clock frequency, data transfer size, and addressing scheme, among others.

Another advantage of the APB protocol is its compatibility with other bus-based protocols, such as AHB (Advanced High-performance Bus) and AXI (Advanced eXtensible Interface). This allows designers to create complex SoC designs with multiple bus protocols and peripheral devices.

In this project report, we will explore the features and advantages of the APB protocol, as well as its use cases and limitations in various applications.

2. Objectives

1. To verify RTL code associated with APB protocol using Universal Verification Methodology.
2. To obtain Physical Design of APB protocol

3. Specifications

Some of the key specifications of the APB protocol:

- Bus Width: The APB protocol supports data transfers of 8, 16, or 32 bits wide.
- Addressing Scheme: The APB protocol uses a simple addressing scheme, where peripherals are assigned a unique address on the bus.
- Clocking Scheme: The APB protocol is asynchronous, meaning that it does not require a clock signal to transmit data. This results in lower power consumption compared to synchronous bus protocols.
- Read/Write Cycle: The APB protocol follows a standard read/write cycle, where the CPU sends a command to the peripheral device to initiate a data transfer.
- Burst Mode: The APB protocol supports burst transfers, where multiple data transfers can be initiated with a single command.
- Error Handling: The APB protocol defines error handling mechanisms, including parity checking and error reporting, to ensure reliable data transfer.
- Configurability: The APB protocol is highly configurable, allowing designers to adjust the protocol parameters to optimize performance and power consumption for their specific system requirements.

4. Architecture/Design/Block or Schematic diagram/Flowchart/Proposed modules

4.1 AMBA BUS architecture

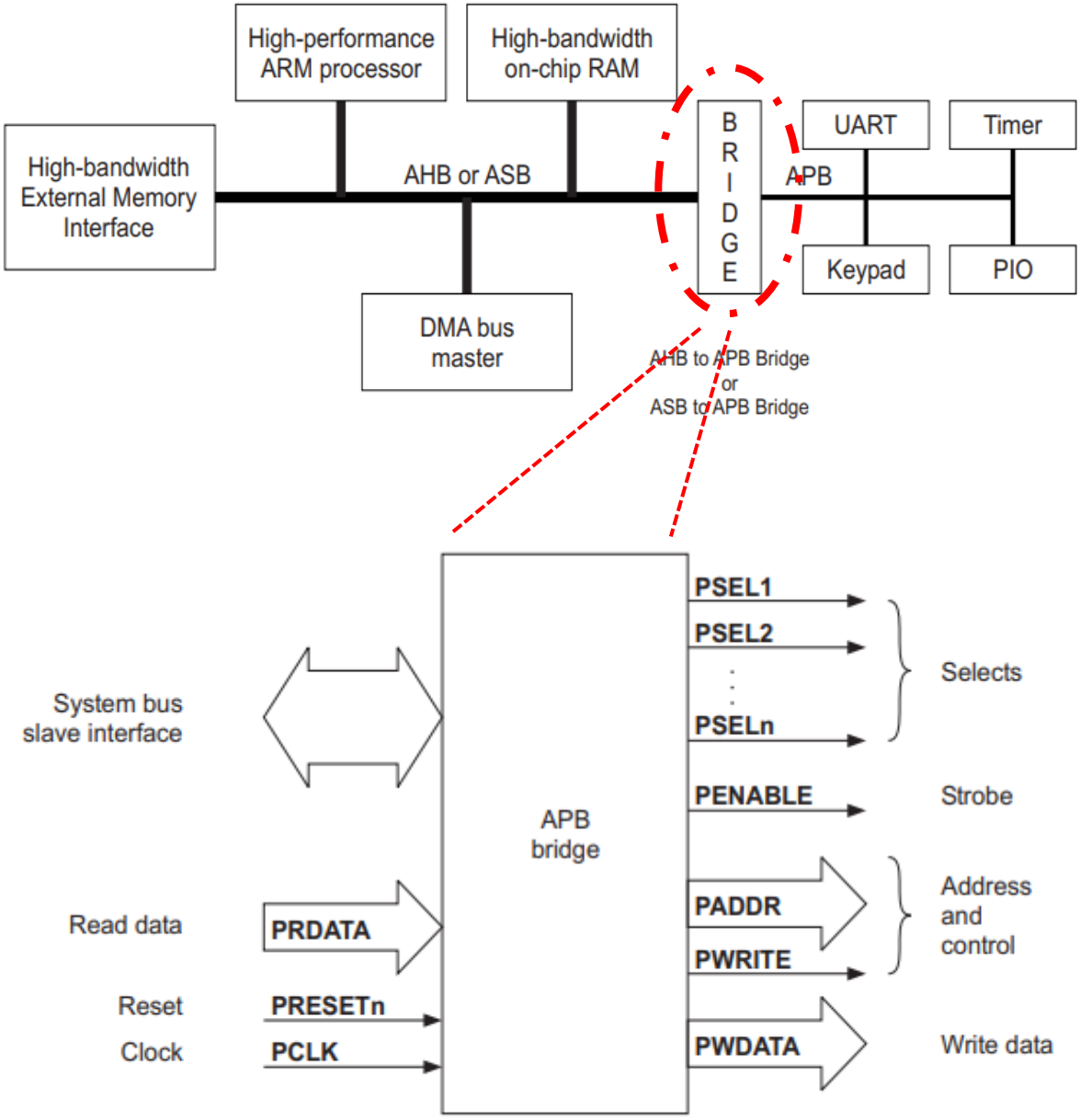


Fig 4.1: AMBA bus architecture

The block diagram of a AMBA bus architecture provides an overview of the available pins and the pin description below provides a brief understanding of the abbreviations and the basic functionalities that each pin performs.

Signal	Source	Description	Width (bit)
TRANSFER	System Bus	APB enable signal. If high AAPB is activated else APB is disabled.	1
PCLK	Clock Source	All APB functionality occurs at rising edge	1
PRESETn	System Bus	An active low signal	1
PADDR	APB bridge	The APB address bus can be up to 32 bits.	8
PSEL1	APB bridge	There is a PSEL for each slave. It's an active high signal	1
PENABLE	APB bridge	It indicates the 2nd cycle of data transfer. It's an active high signal.	1
PWRITE	APB bridge	Indicates the data transfer direction. PWRITE = 1 indicates APB write access (master to slave) PWRITE = 0 indicates APB read access (slave to master)	1
PREADY	Slave interface	This is an input from slave. It is used to enter access state.	1
PSLVERR	Slave interface	This indicates a transfer failure by the slave.	1
PRDATA	Slave interface	Read Data. The selected slave drives this bus during read operation.	8
PWDATA	Slave interface	Write data. This bus is driven by the peripheral bus bridge unit during write cycles when PWRITE is high	8

Table 4.1: Pin description of AMBA APB bridge

4.2 Interface diagram

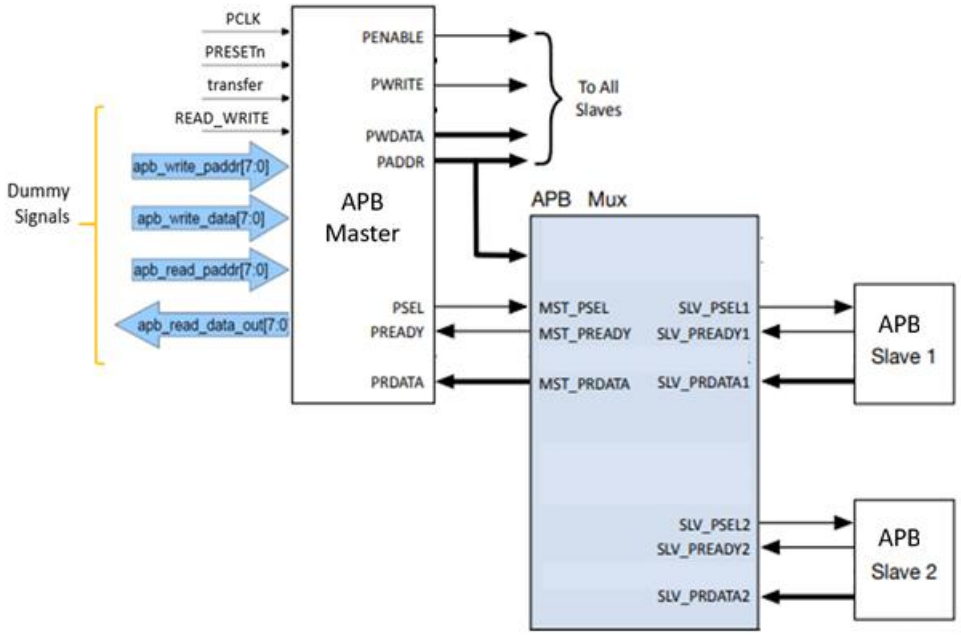


Fig 4.2 APB Interface diagram

The design of the RTL code utilized contains a master and two slaves following the APB protocol. There are additional signals in the designed interface which controls the communication between master and slave. These signals are apb_write_paddr[7:0] (eight bit address of particular memory location in the slave ram for write operation), apb_read_paddr[7:0] (eight bit address of particular memory location in the slave ram for read operation), apb_write_data[7:0] (eight bit data need to be written at apb_write_paddr[7:0]), apb_read_data_out[7:0] (output eight bit data read from apb_read_paddr[7:0] of selected slave ram),READ_WRITE (signal that specifies whether the operation to be performed is a read from slave when it is logic high or an operation to write on to slave when the signal is logic low).

4.3 Finite State diagram

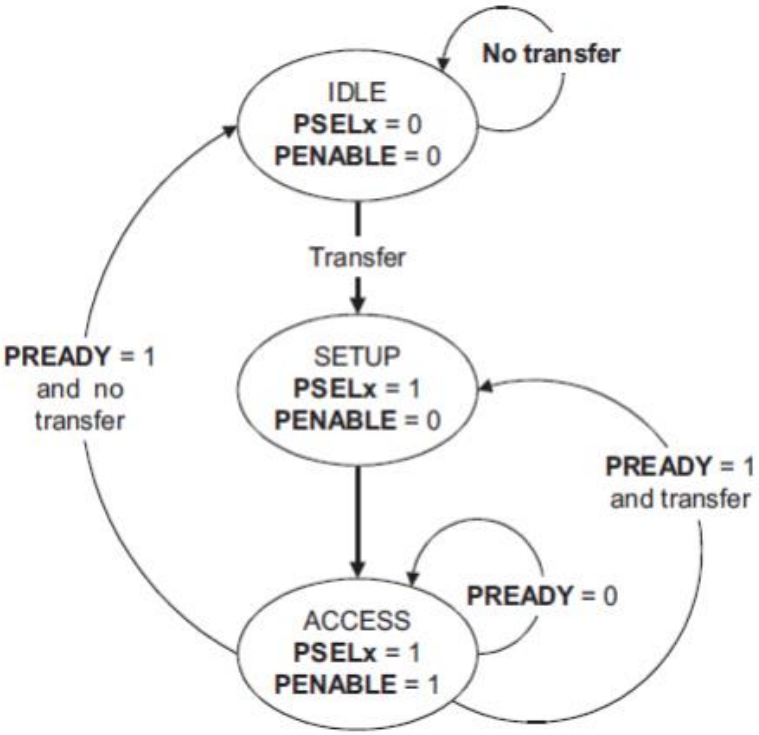


Fig 4.3: Finite state diagram

FSM states

1. Idle state

This state indicates that no operation is being performed. All signals - PSEL, PENANBLE, PADDR are initially unknown (zero). If no transfer is required then automatically idle state will be asserted. Idle state is also the default state.

2. Setup state

Setup state is active when the transfer is required. In the setup phase, PSEL signal will become active. This indicates beginning of transfer and the presence of PADDR & PDATA. The bus will enter setup phase only when the transfer is needed, otherwise bus will remain in idle phase.

PWRITE, PADDR, PWDATA are provided in this phase.

The bus will remain in setup phase for one clock cycle and in the next positive edge of clock or rising edge of clock, the bus will automatically move to ACCESS state.

3. Access state

This state is mainly used to tell that the transaction is completed. At the starting of access state PENABLE will become high (1) and the remaining signals like (PADDR, PWRITE,

PSEL, PWDATA) all remain stable as in setup phase, without changes in this signal. In access state PENABLE and PREADY signal will become high (1) and this change of both PENABLE and PREADY indicates completion of one transaction. If further transfer needed, then bus will move to SETUP phase otherwise moves to the default state (IDLE STATE).

5. Work done/Results

5.1 Physical Design

5.1.1 Design import

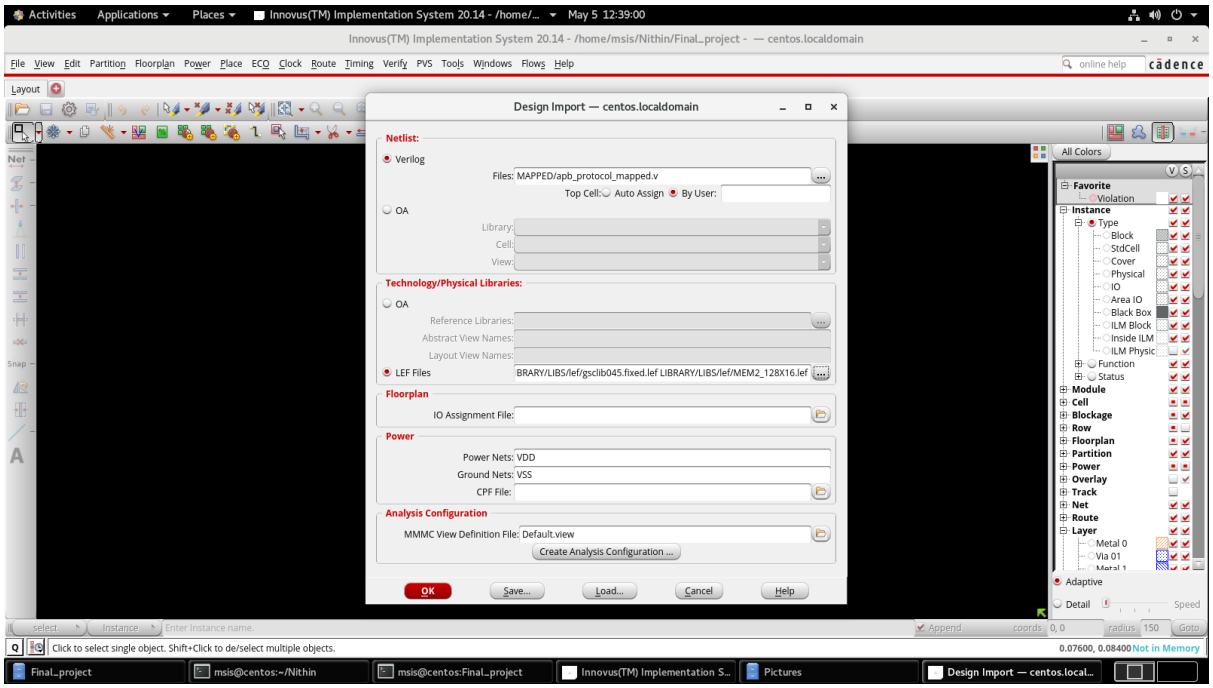


Fig 5.1: Importing design

Importing the design, with the Verilog and LEF files. The Power and Grounds nets are specified as VDD & VSS respectively.

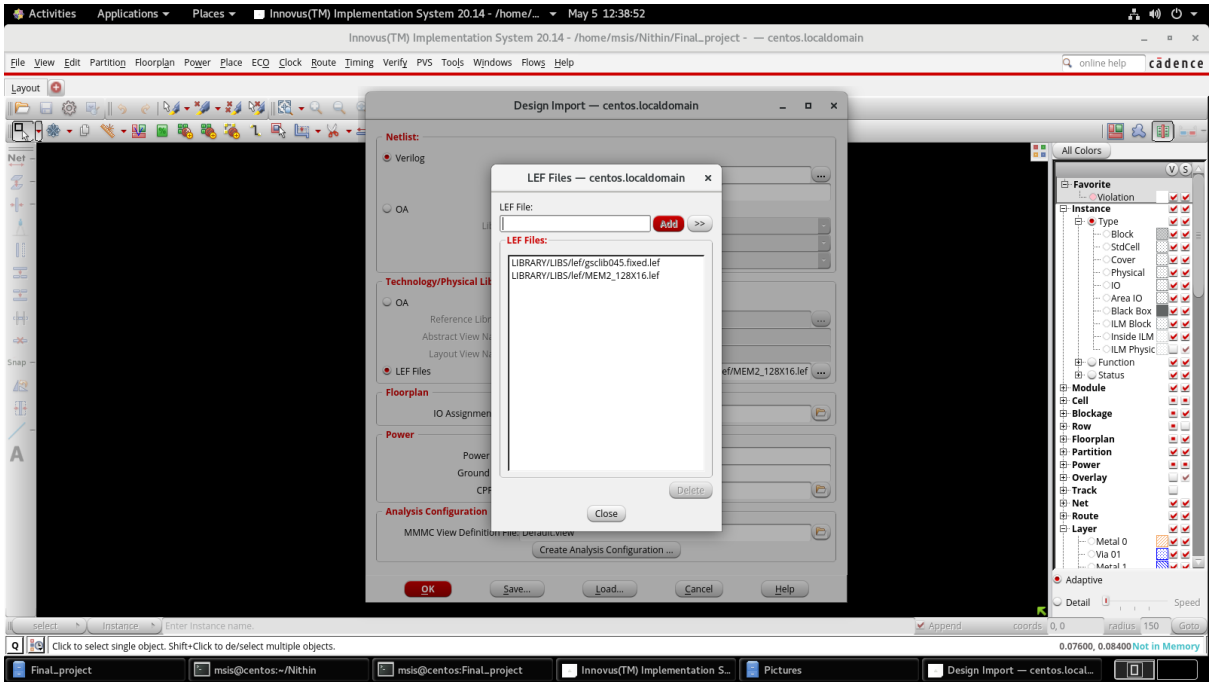


Fig 5.2 LEF files added

Two LEF files added are “gsclib045.fixed.lef” and “MEM2_128x16.lef”

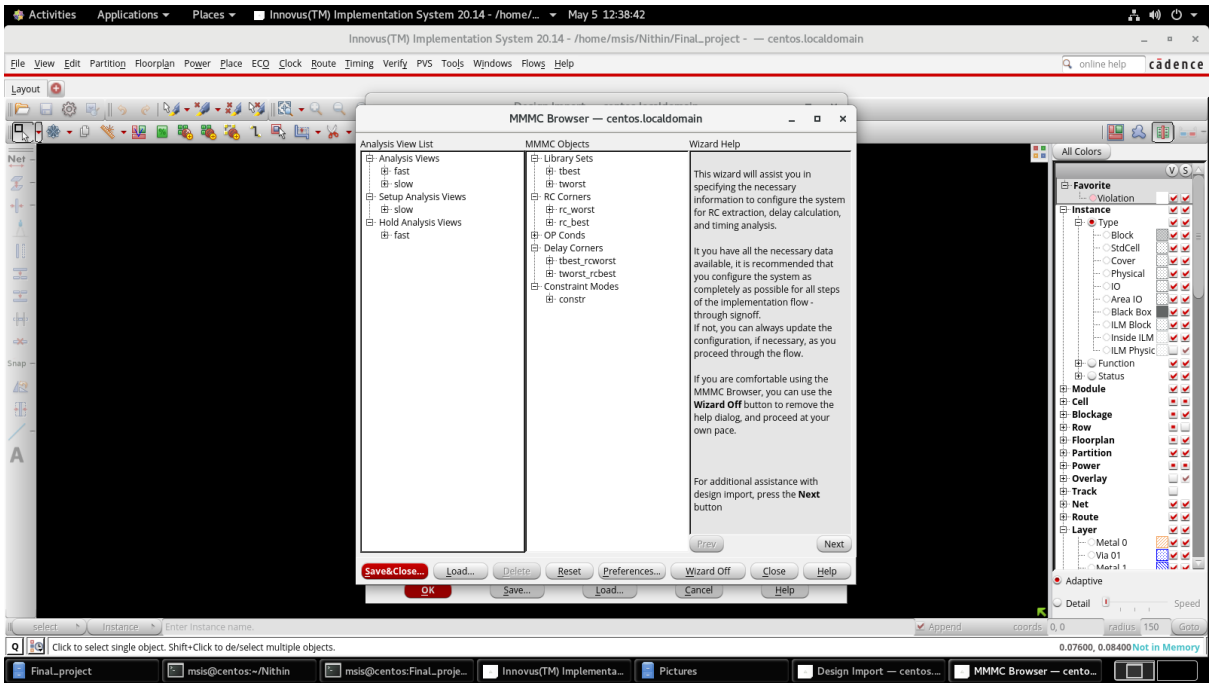


Fig 5.3: MMMC Browser

Added library sets, RC corners, delay corners, constraints. Created analysis views for both setup and hold.

5.1.2 Floor planning

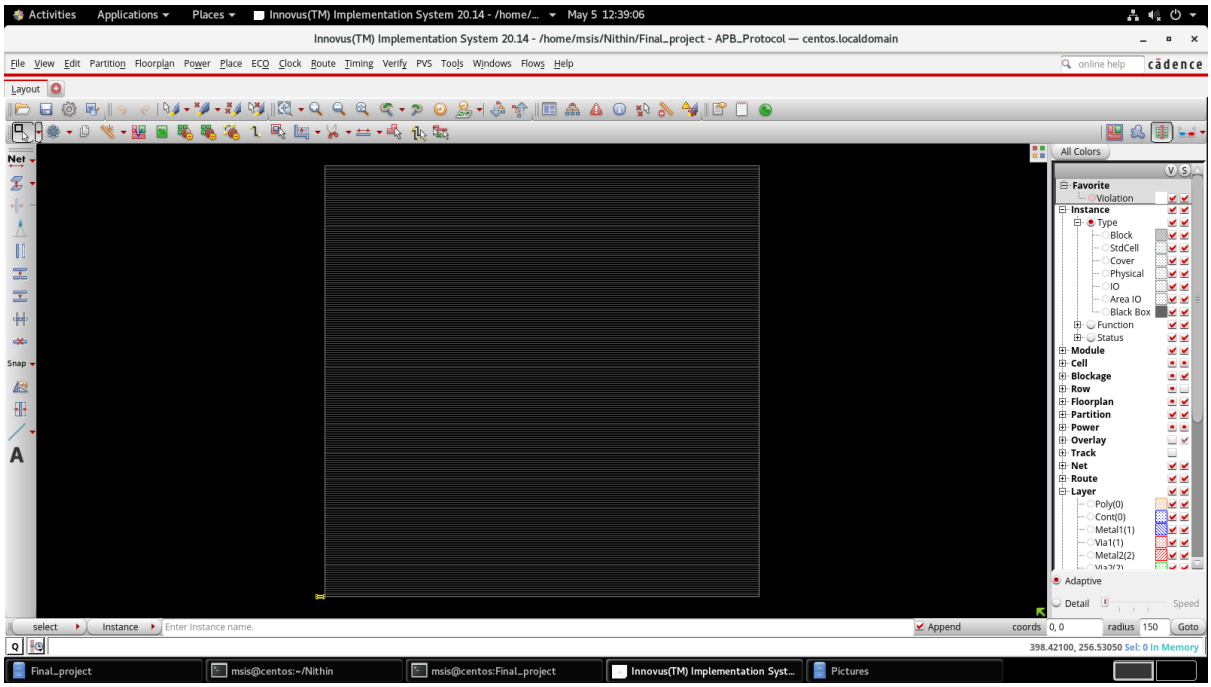


Fig 5.4: Initial floorplan

Initial floorplan after completing design import.

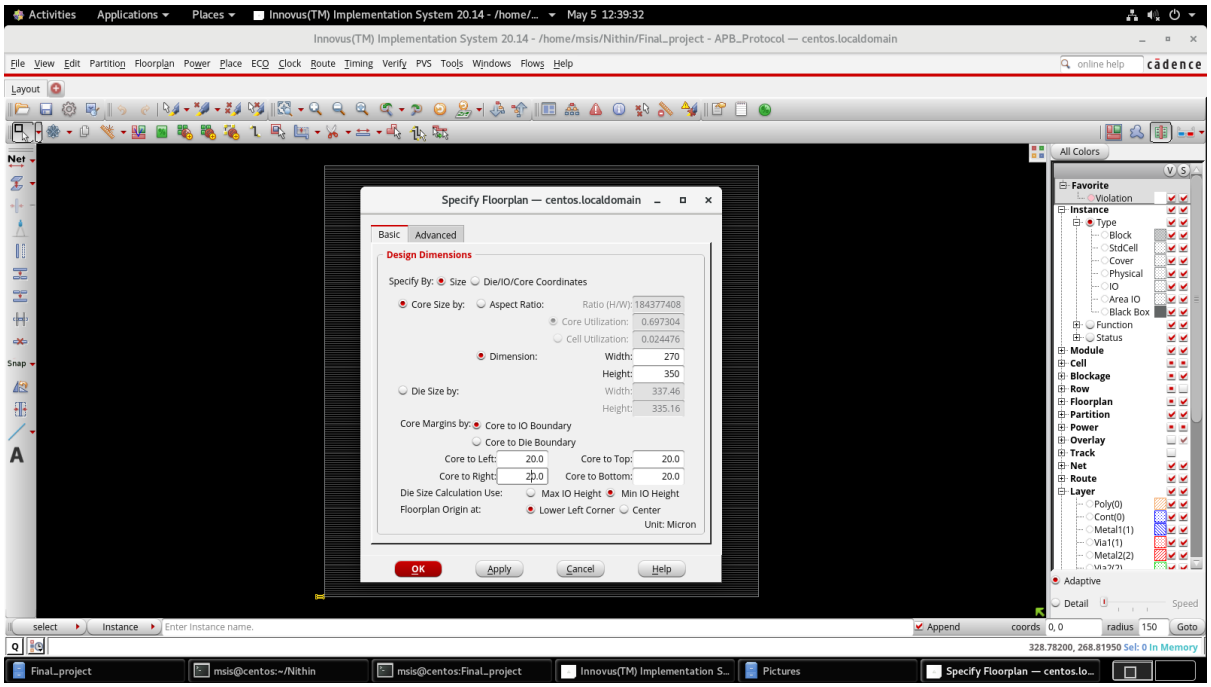


Fig 5.5: Floorplan specifications

Dimensions has been set Width: 270, Height: 350 to accommodate sufficient space for macros and standard cells. Core to IO boundary has been equally spaced 20units from all directions.

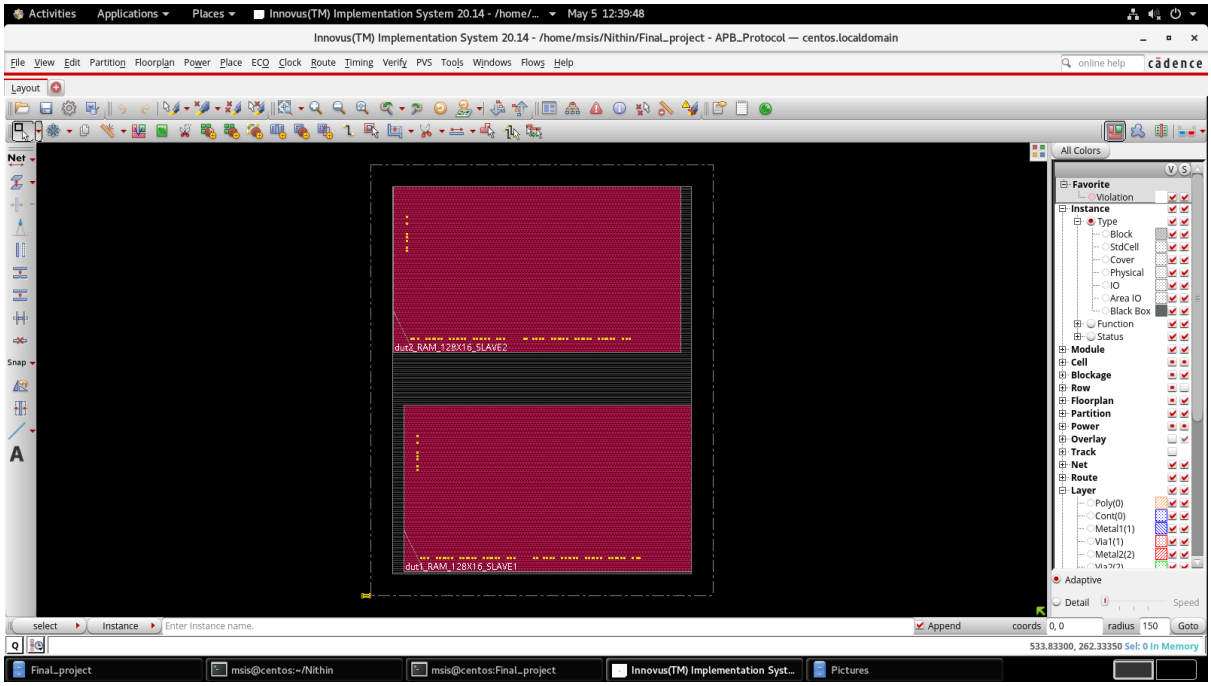


Fig 5.6: Floorplan after macros placement

The APB protocol code consists of one master and two slaves, out of which the two slaves have been redesigned to macros and placed.

5.1.3 Power Planning

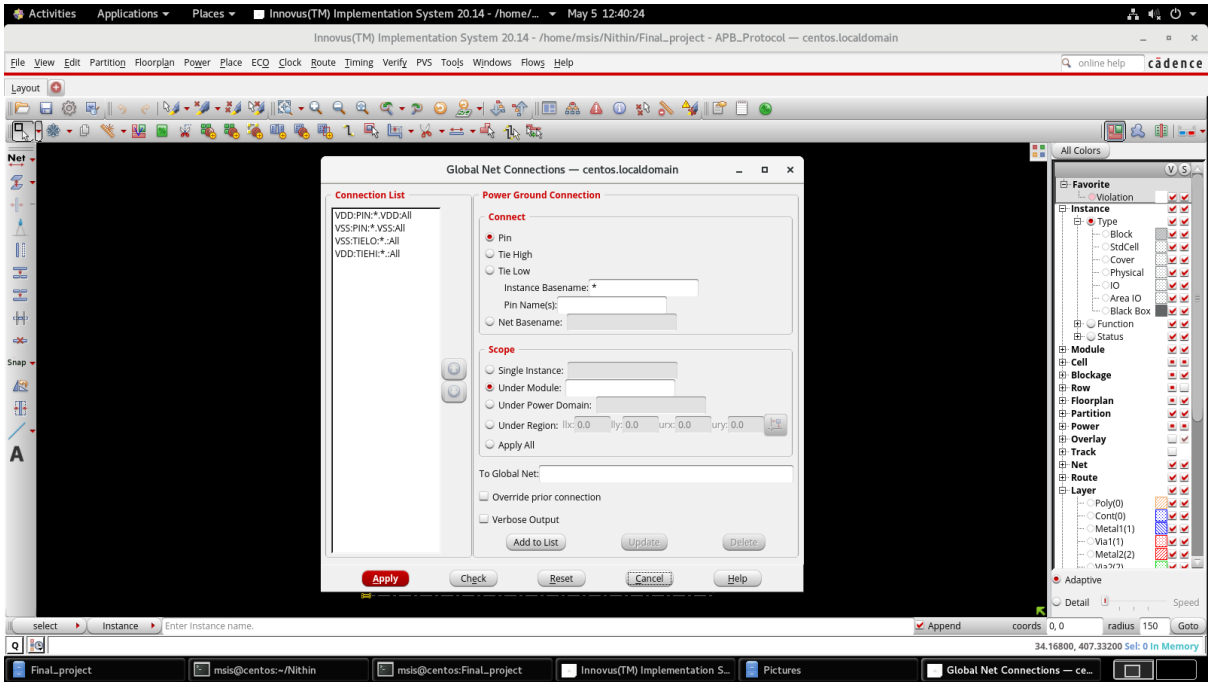


Fig 5.7: Global Net Connections

Creating Global Net Connections for VDD & VSS

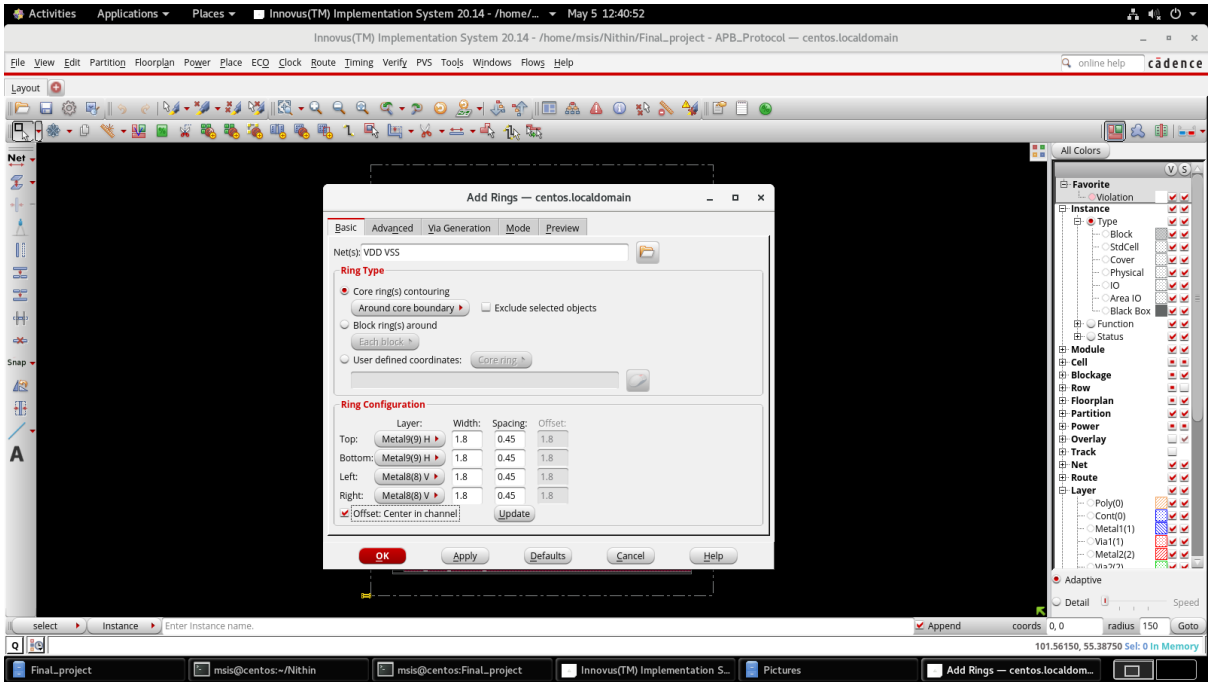


Fig 5.8: Adding Rings

Adding VDD & VSS rings around the core boundary using “Metal 9” for Top and Bottom, “Metal 8” for Left & Right.

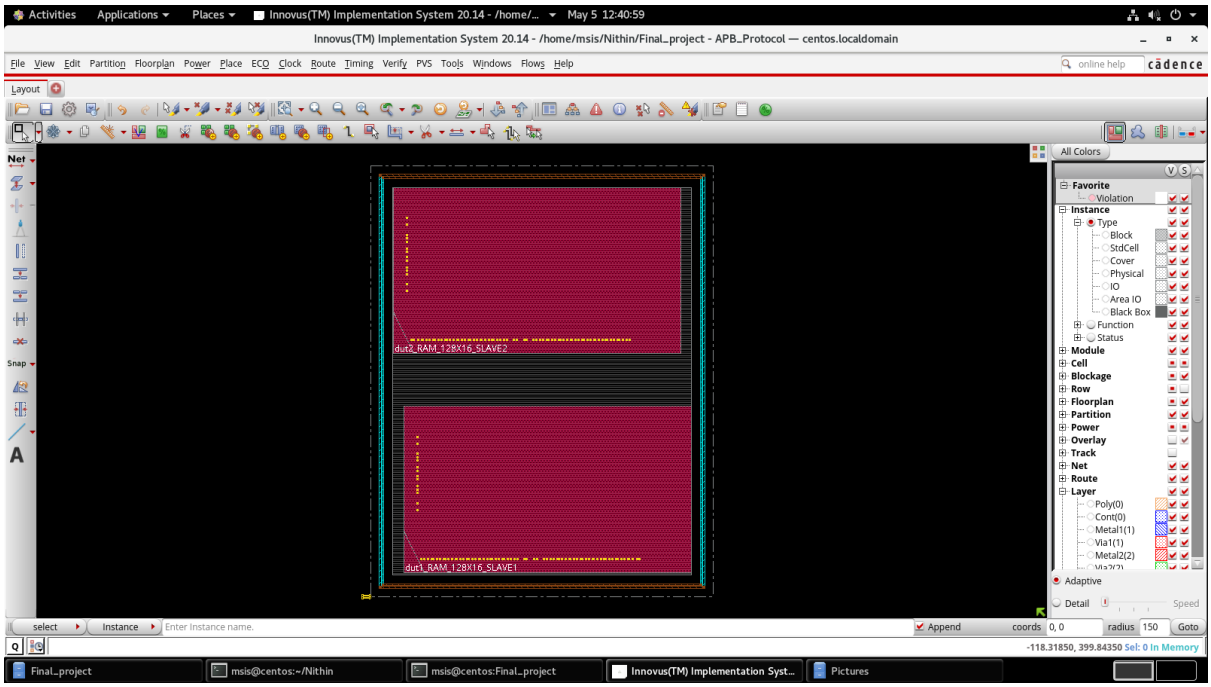


Fig 5.9: Ring placement around core boundary

The VDD & VSS rings have been placed around the core IO boundary.

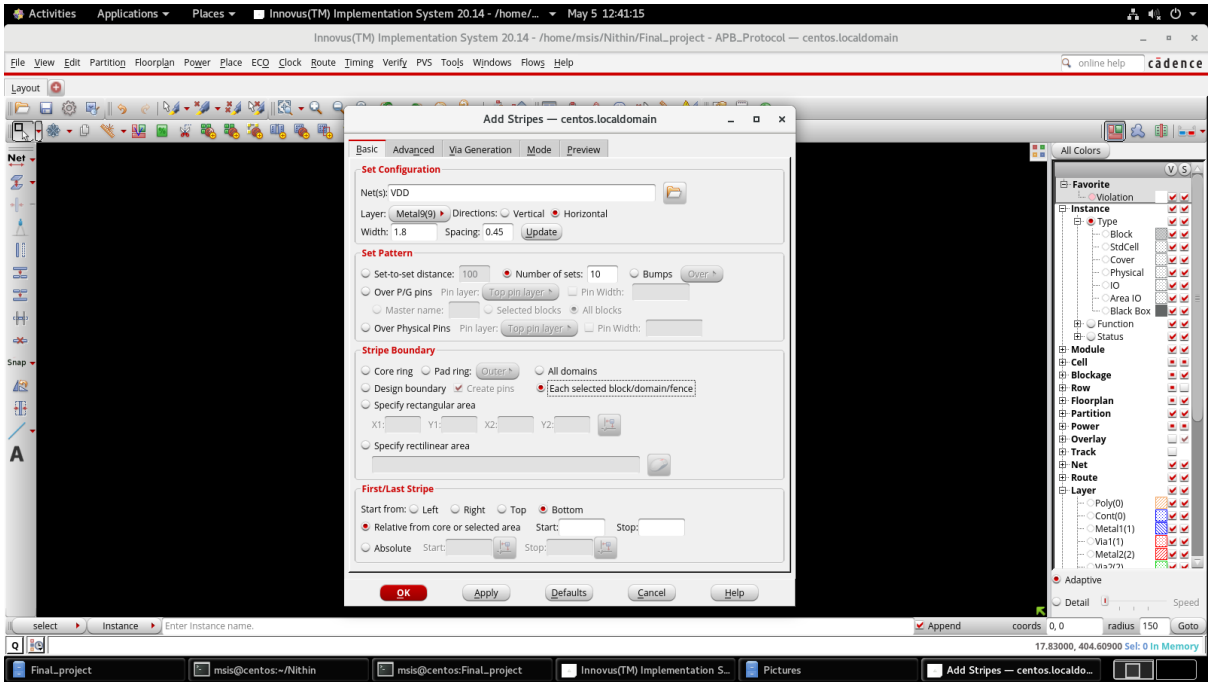


Fig 5.10: Adding VDD stripes

VDD stripes using “Metal 9” of count 10 have been placed.

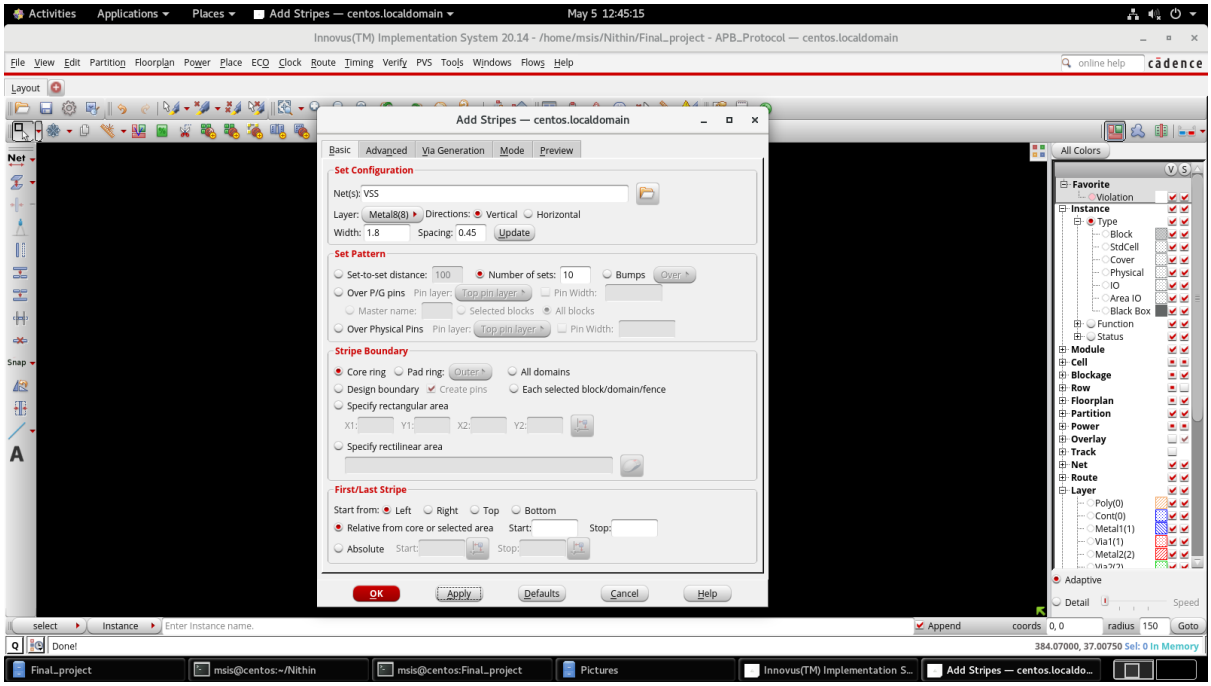


Fig 5.11: Adding VSS stripes

VSS stripes using “Metal 8” of count 10 have been placed.

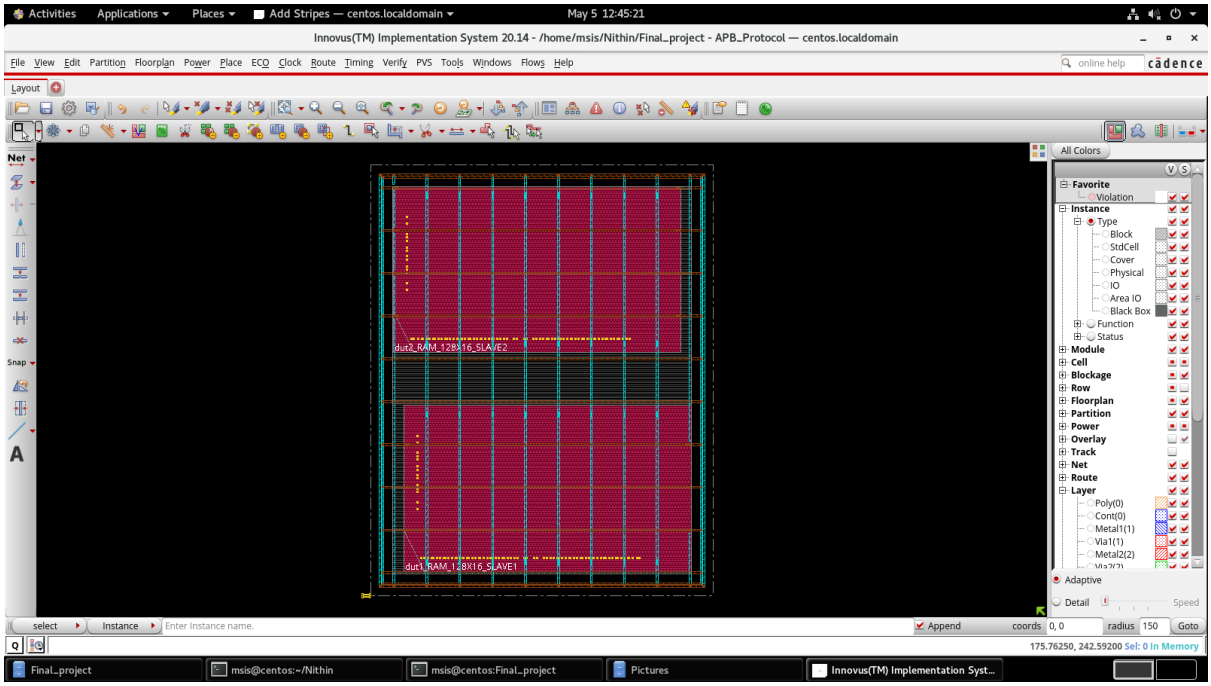


Fig 5.12: Stripes placement

The VDD & VSS stripes have been placed.

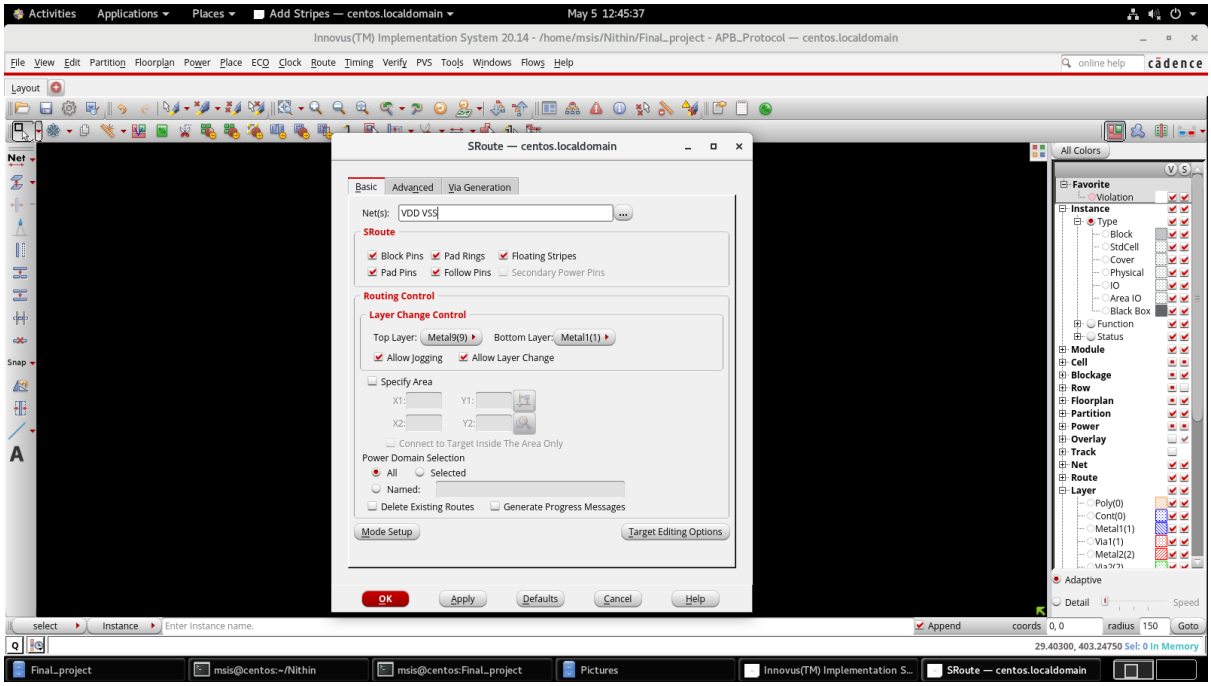


Fig 5.13: Adding VDD & VSS railings

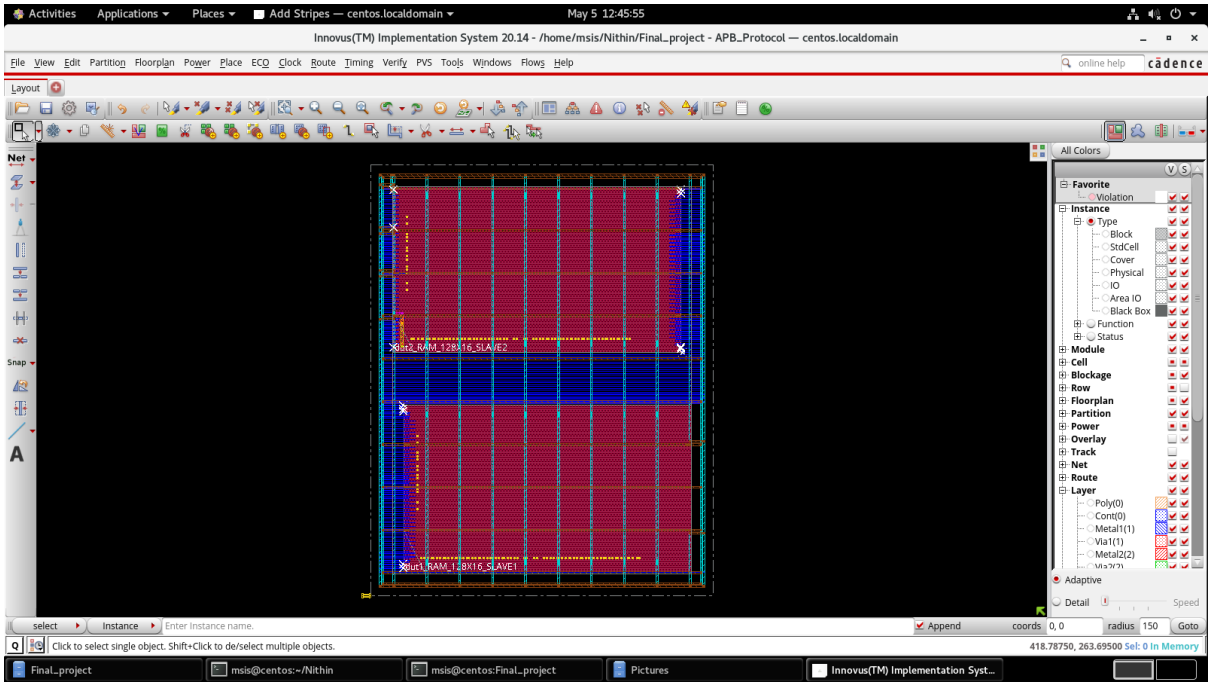


Fig 5.14: VDD & VSS railings

VDD & VSS railings are placed to power supply for macros and standard cells.

5.1.4 Placement

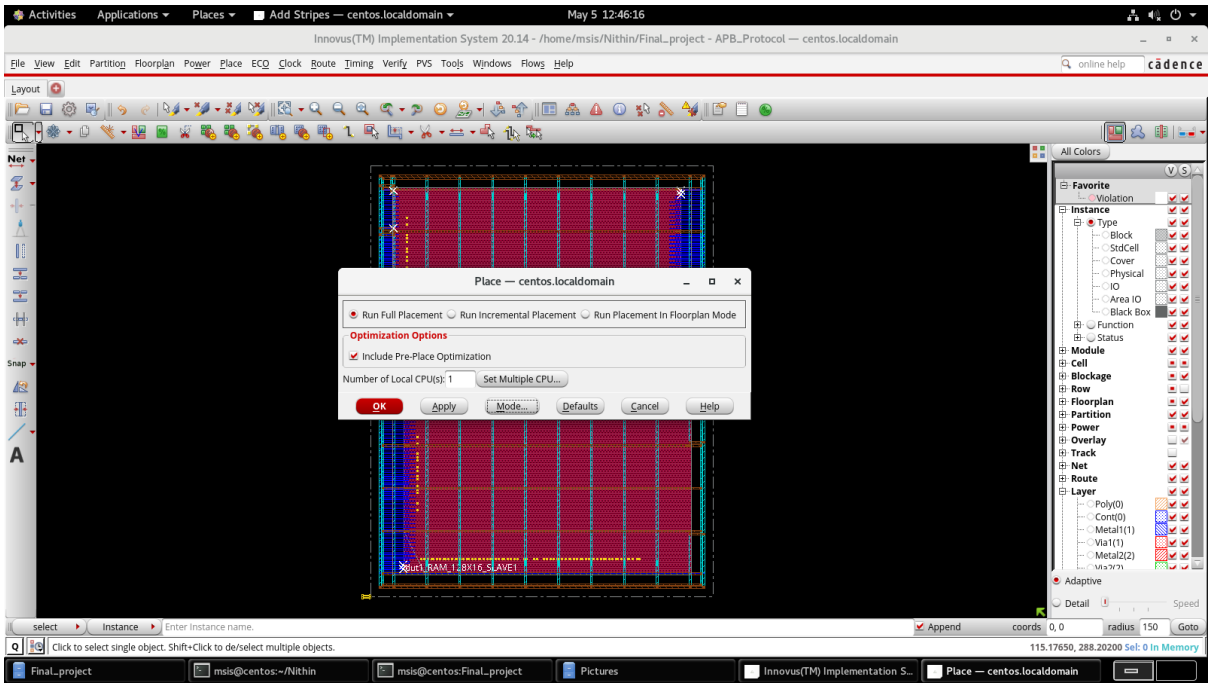


Fig 5.15: Adding standard cells

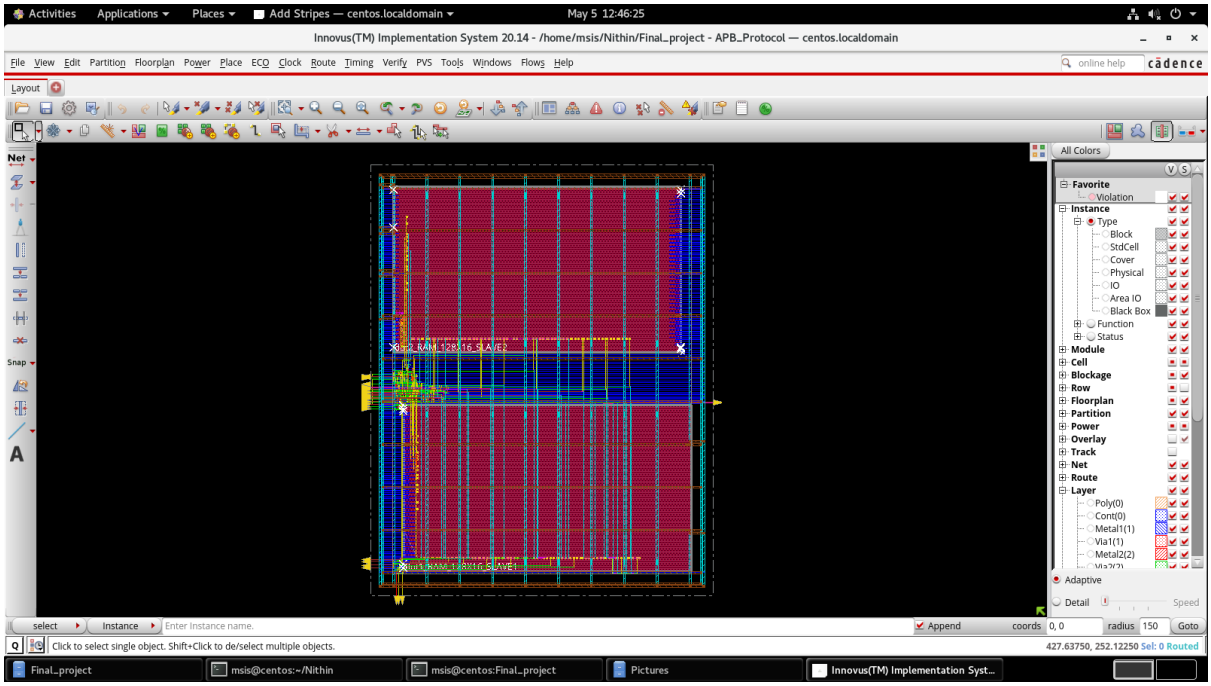


Fig 5.16: Standard cells and IO's placed

Standard cells and IO's have been placed and optimized for best placement.

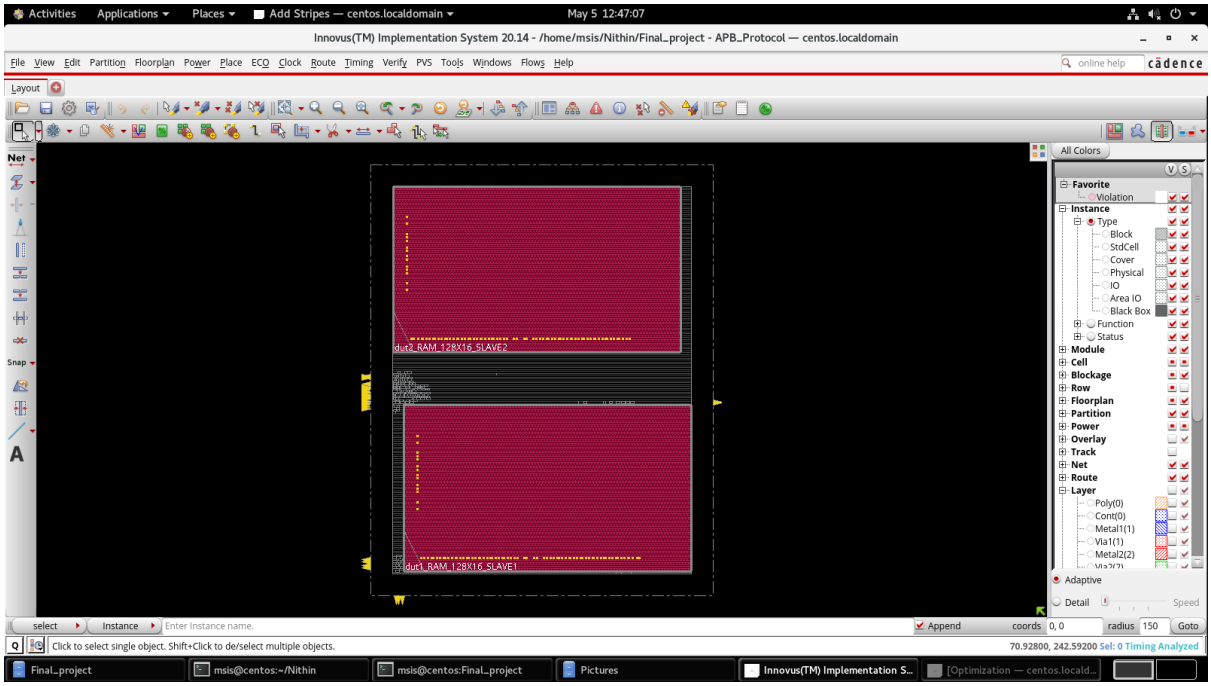


Fig 5.17: Standard cells placement without wiring

A view of the standard cells placed without any wiring connections.

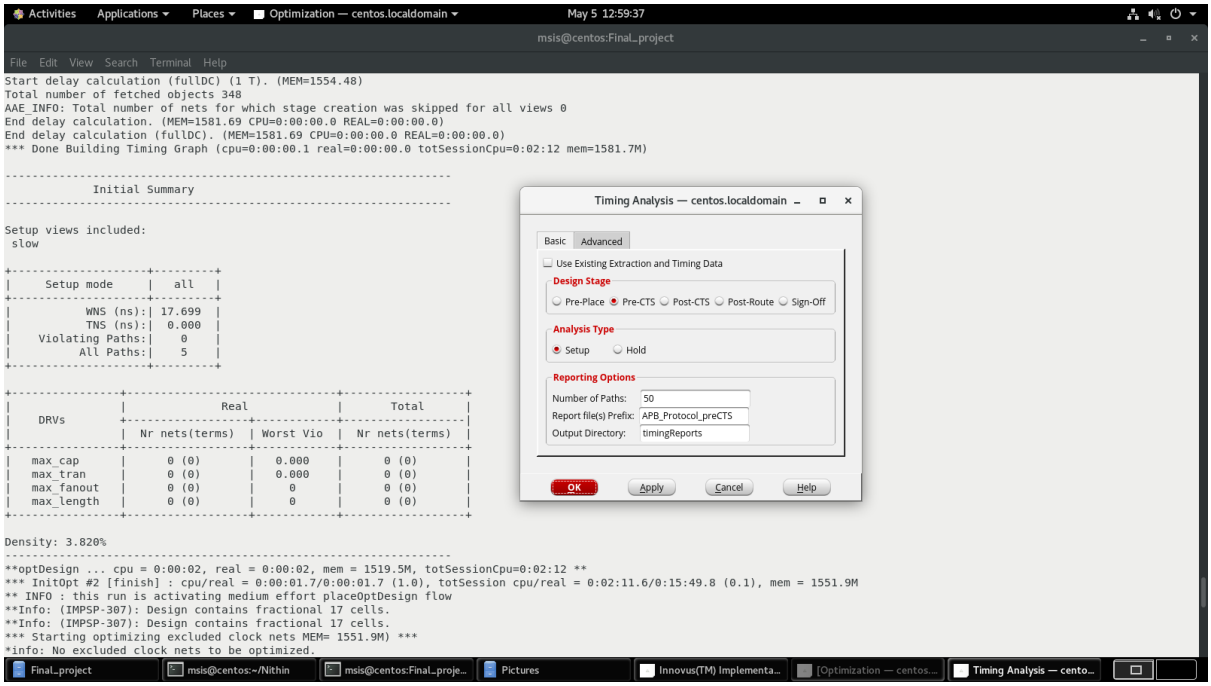
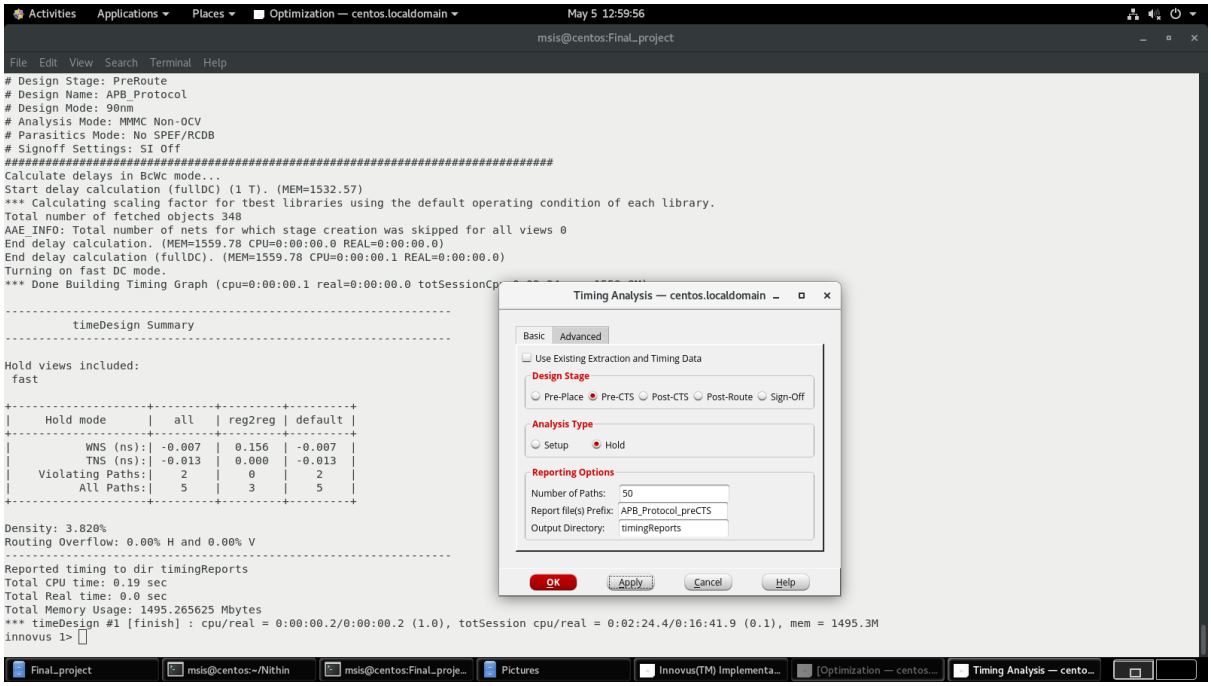


Fig 5.18: Pre-CTS setup timing report

Pre-CTS setup timing report generated shows a positive slack of 17.699ns



.

Fig 5.19: Pre-CTS hold timing report

Pre-CTS hold timing report generated shows a negative slack of 0.007ns.

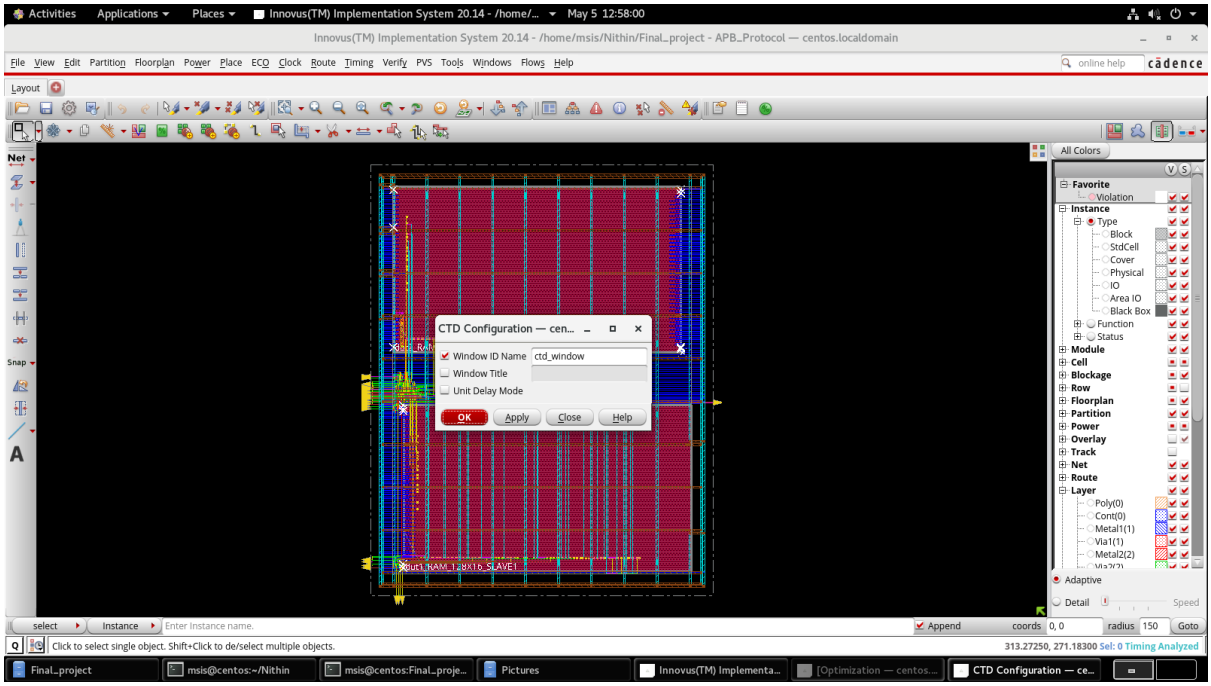


Fig 5.20: Initiating CTS

Clock Tree Synthesis (CTS) has been initiated.

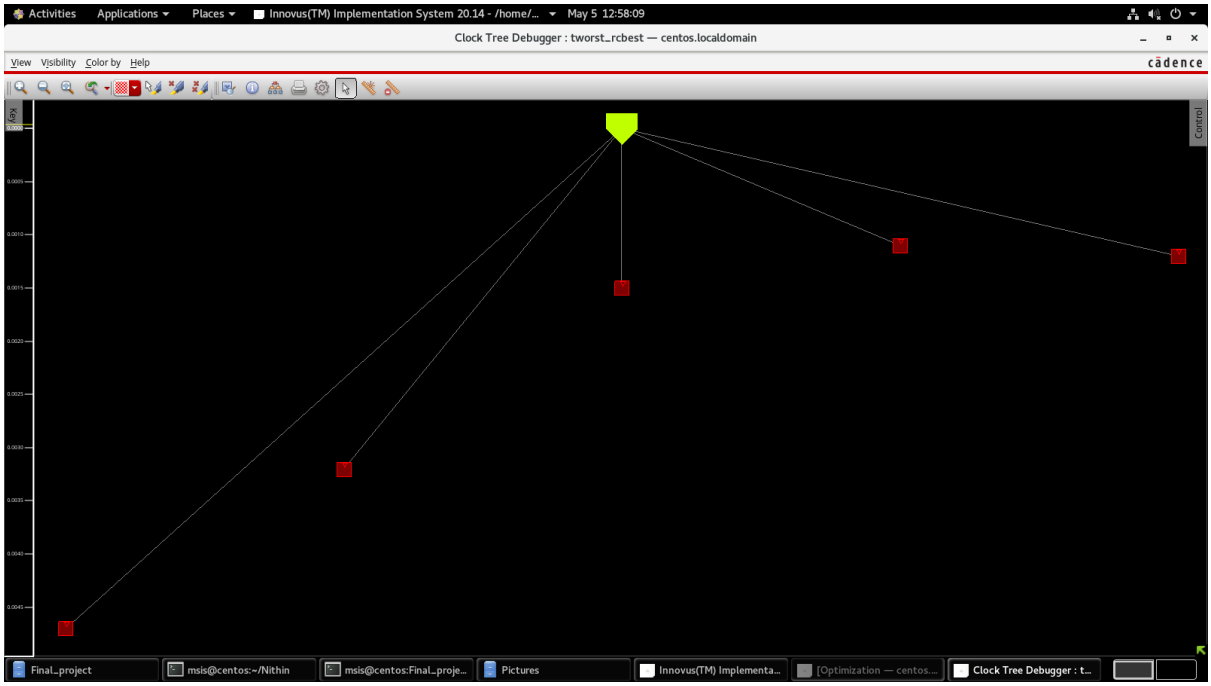


Fig 5.21: Clock Tree Synthesis generated

CTS generated successfully.

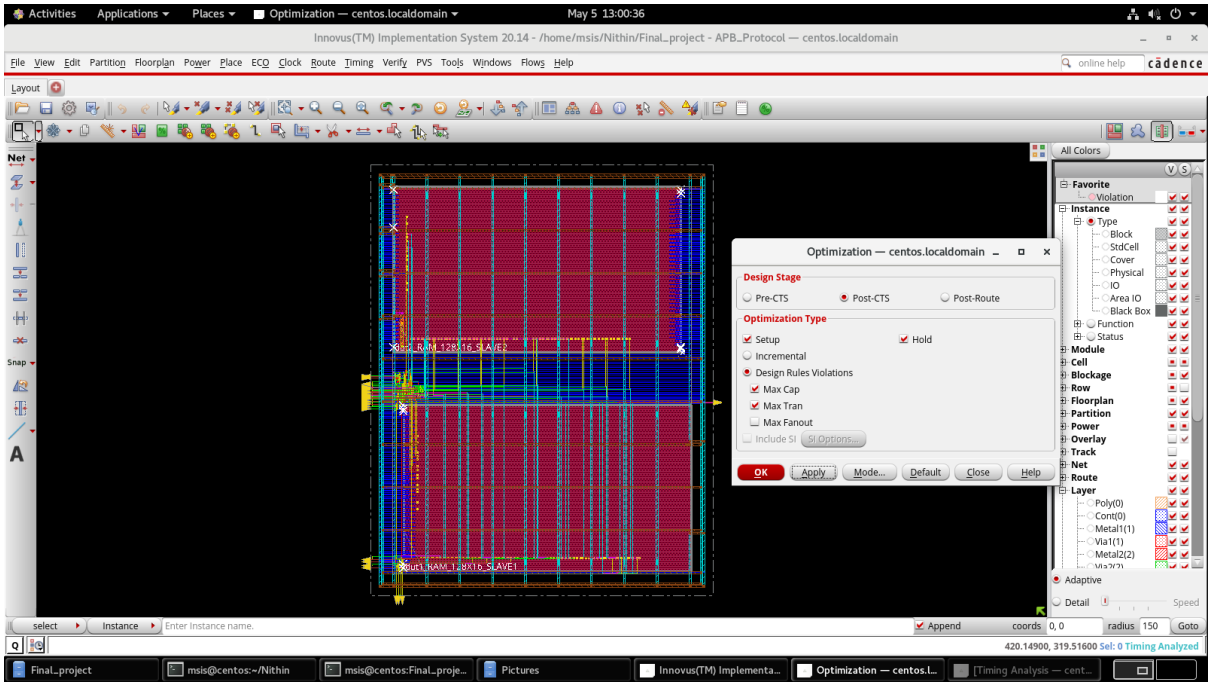


Fig 5.22: Post CTS design optimization

The design has been optimized Post-CTS for best possible results.

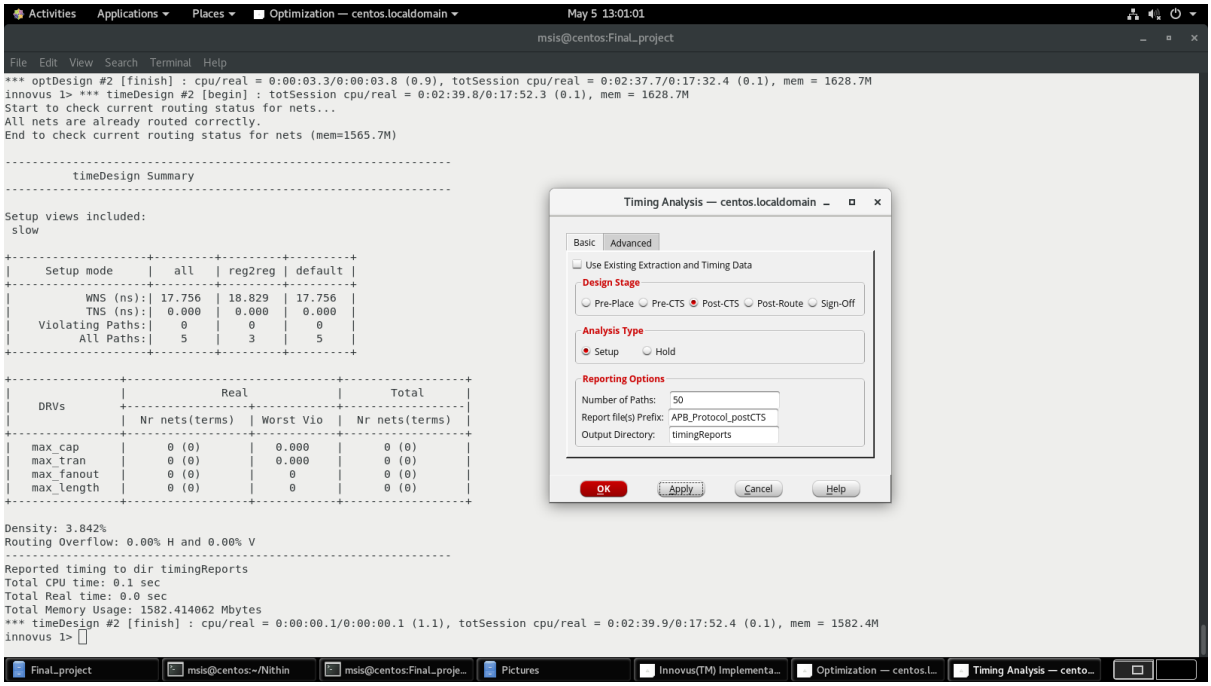


Fig 5.23: Post-CTS setup timing report

Post-CTS setup timing report has been generated and a positive slack of 17.756ns can be observed.

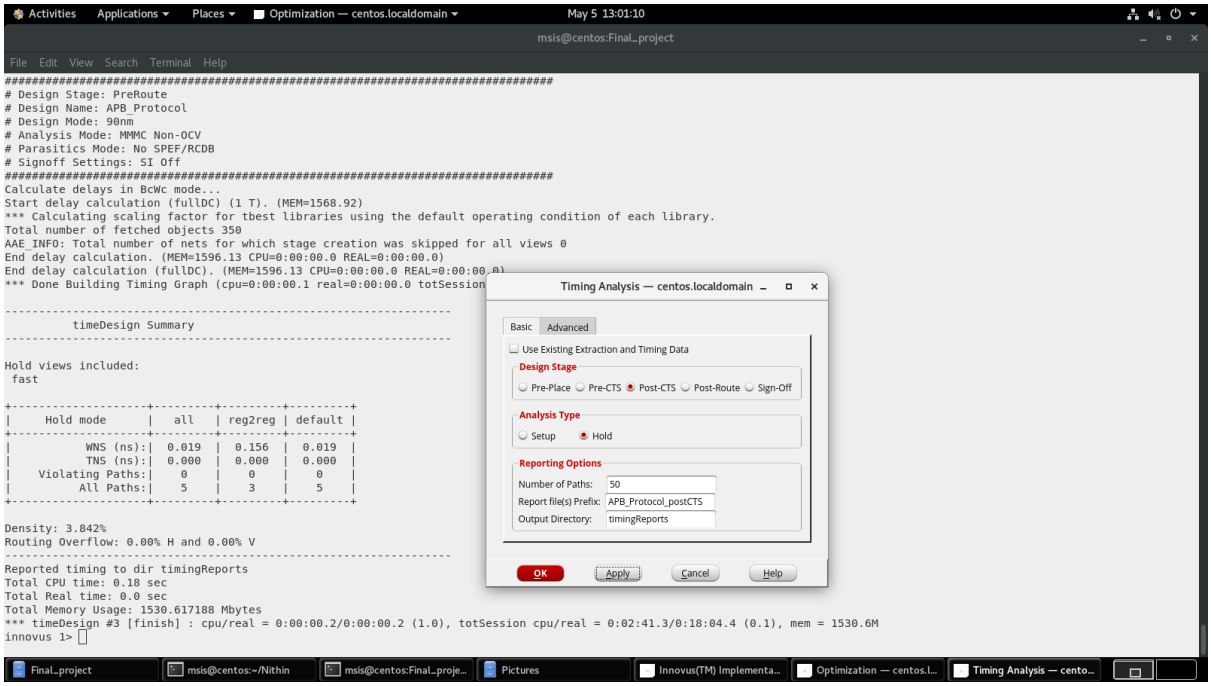


Fig 5.24: Post-CTS hold timing report

Post-CTS hold timing report has been generated and a positive slack of 0.019ns slack has been generated.

5.1.5 Routing

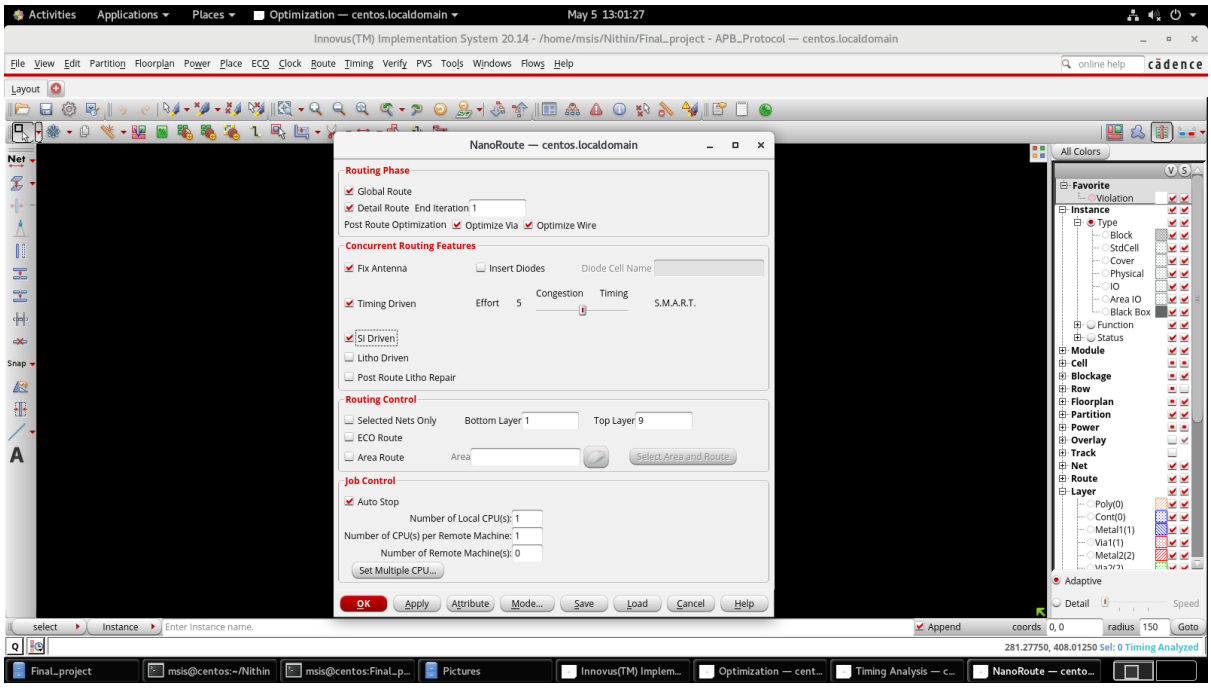


Fig 5.25: Routing specifications

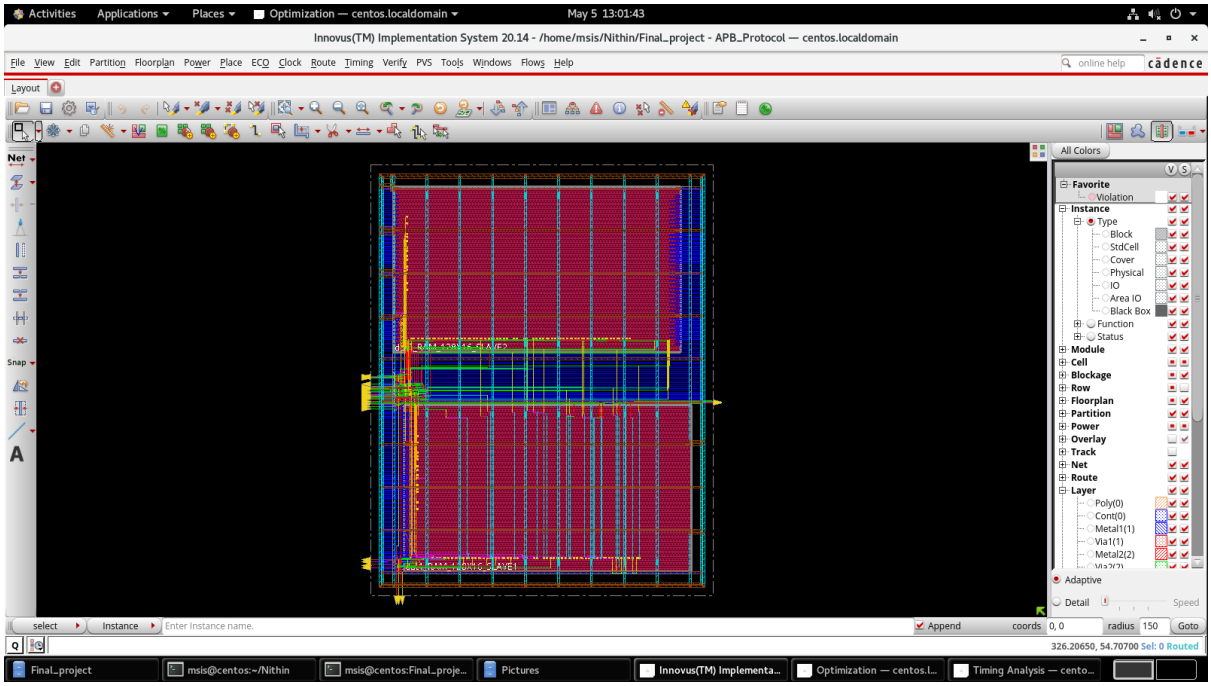


Fig 5.26: Post Routing design

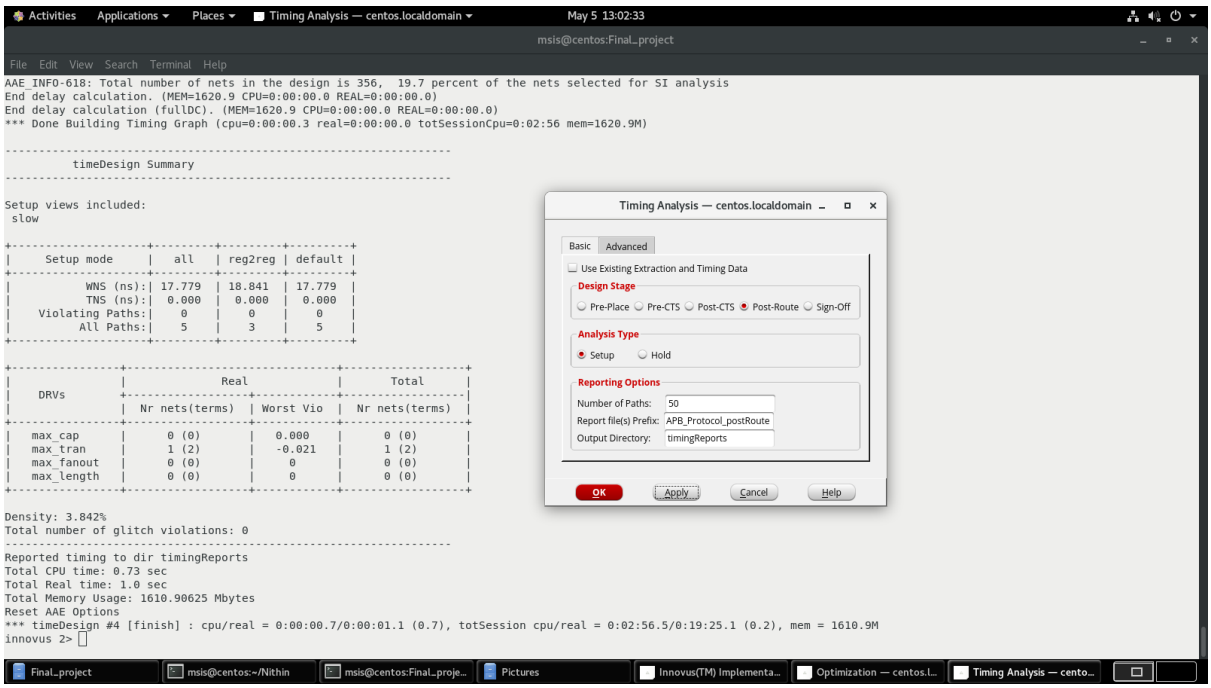


Fig 5.27: Post routing setup timing report

Post routing setup timing report has been generated and a positive slack of 17.779ns can be observed.

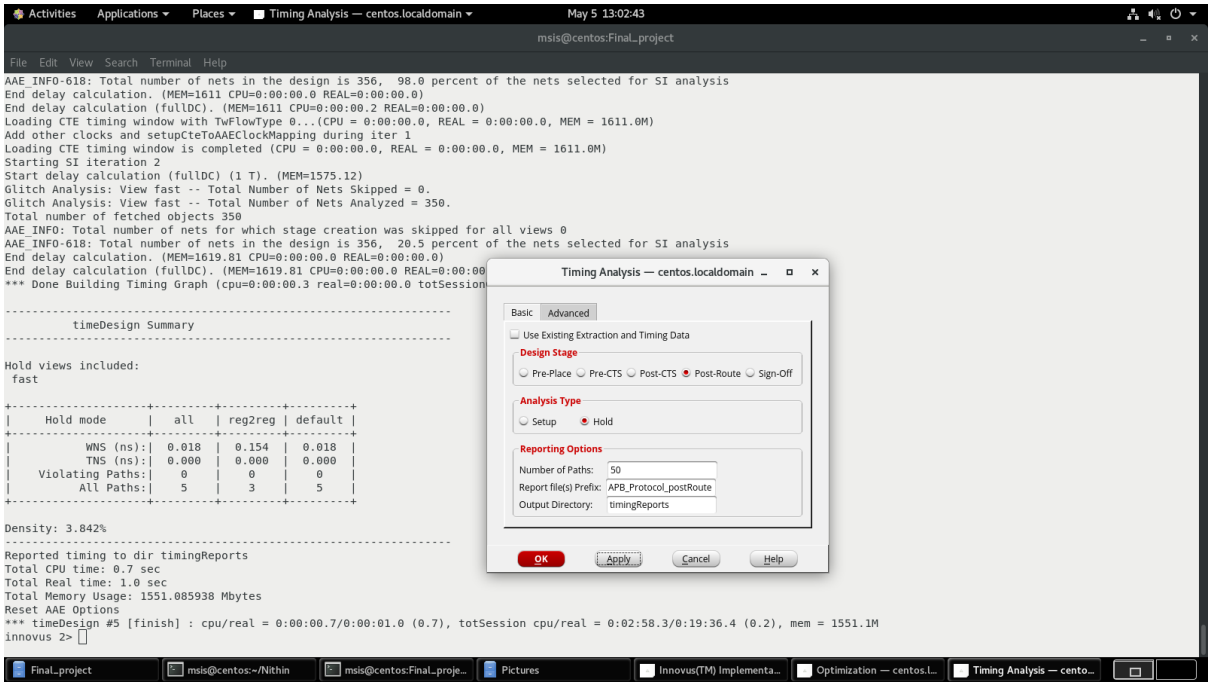


Fig 5.28: Post routing hold timing report

Post routing hold timing report has been generated and a positive slack of 0.018ns can be observed.

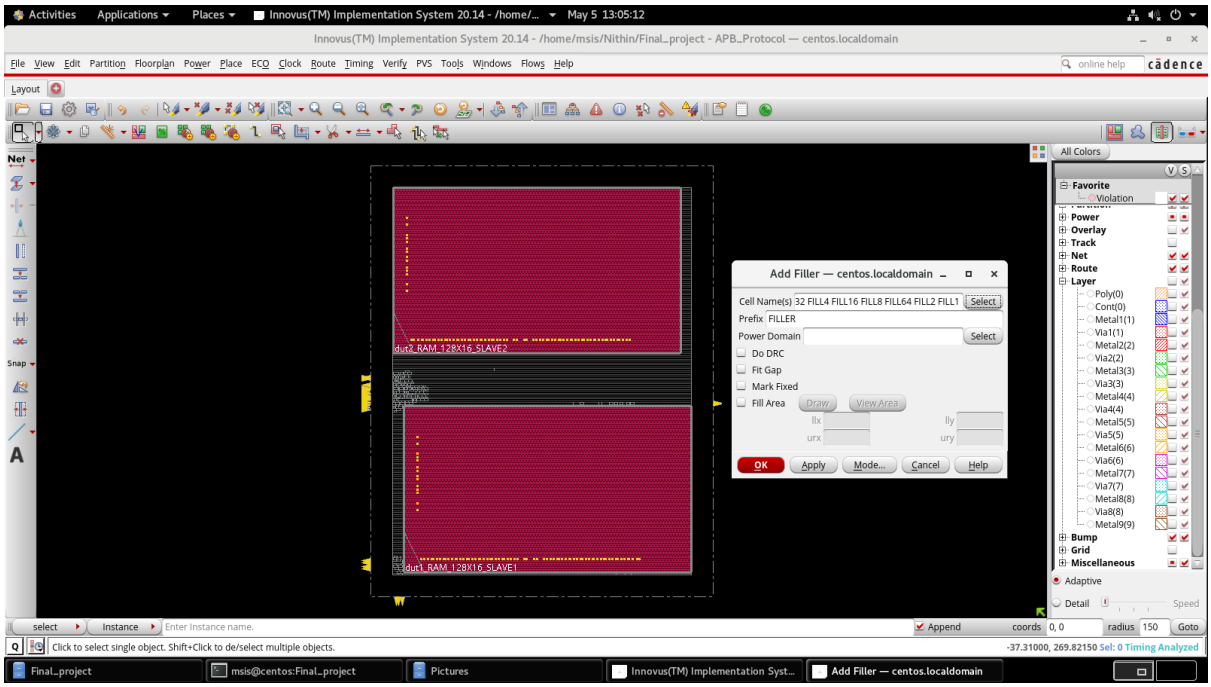


Fig 5.29: Adding fillers

Adding fillers to the design.

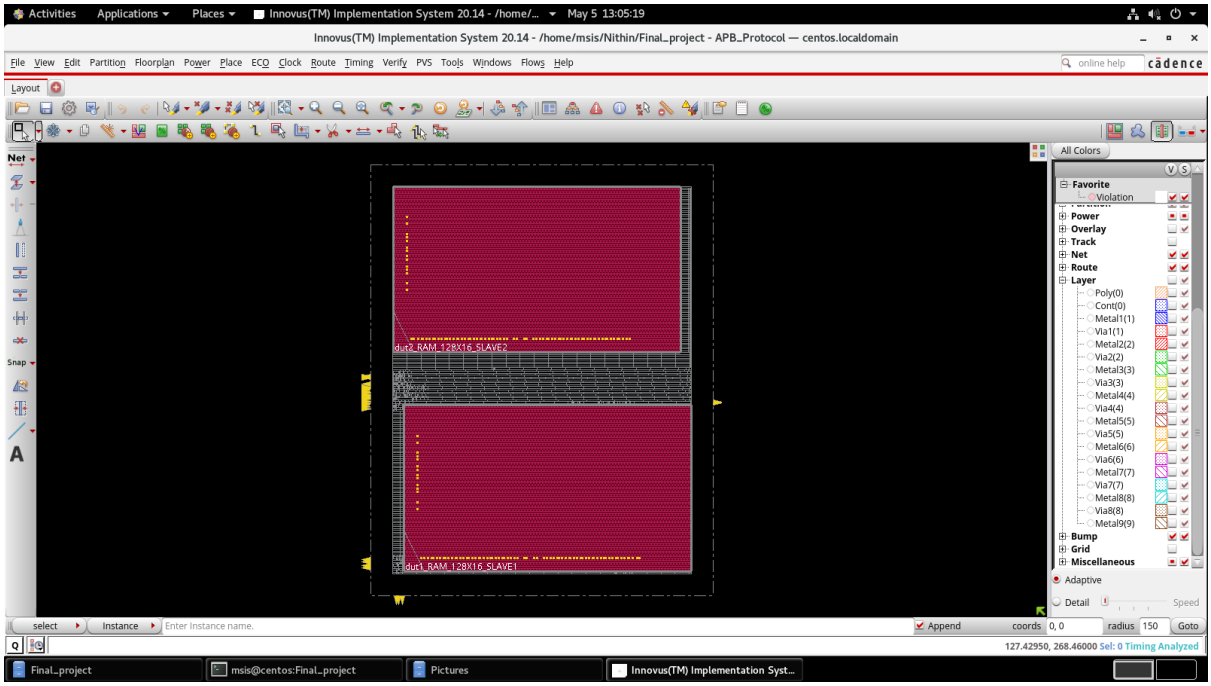


Fig 5.30: After adding fillers

Design has been added with fillers.

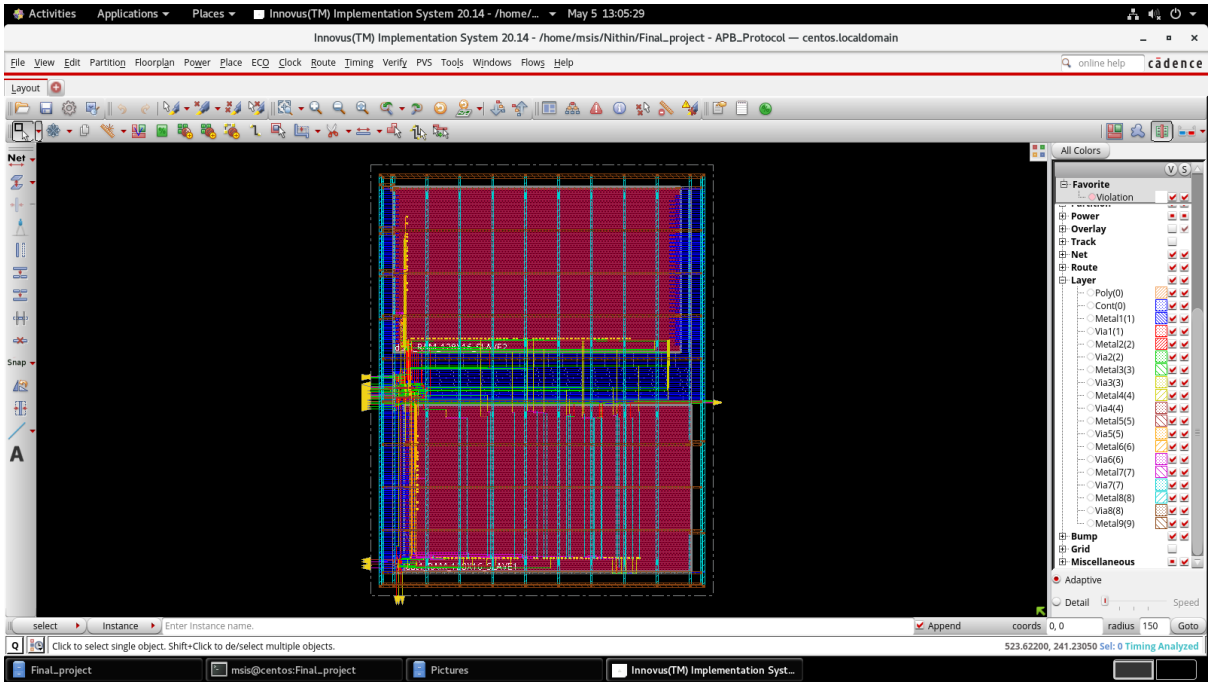


Fig 5.31: Final Physical Design

The completed Physical Design for APB Protocol is as seen above.

5.1.6 DRC check

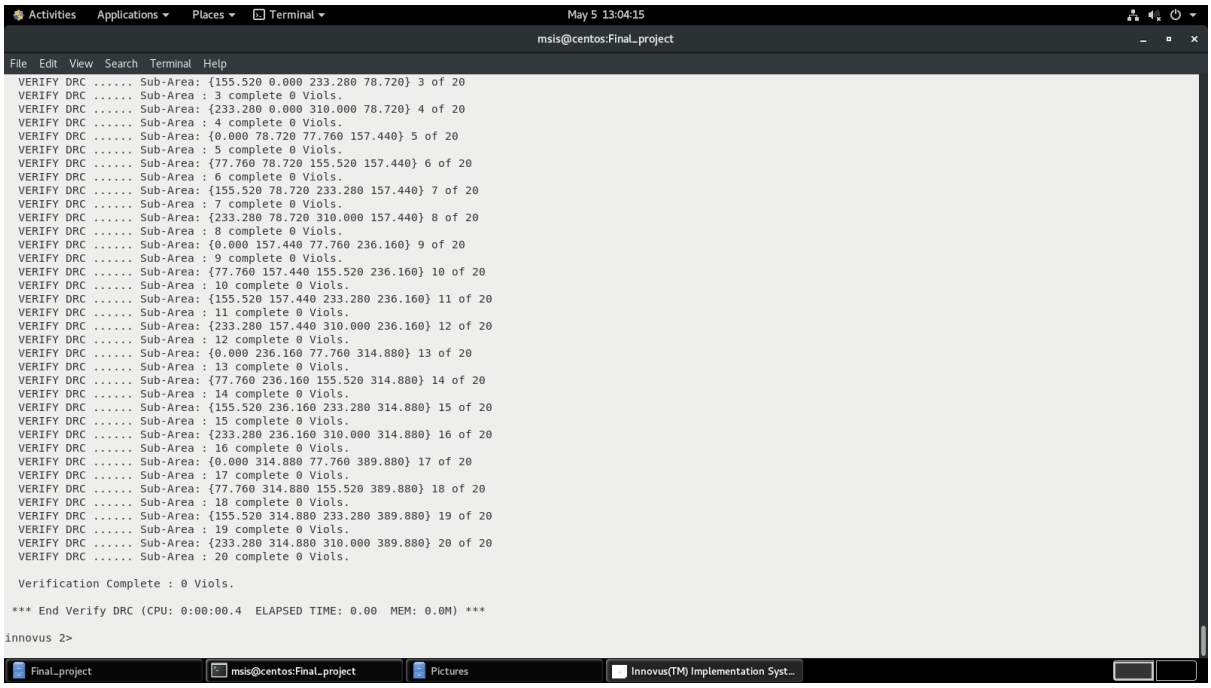


Fig 5.32: DRC check result

DRC check has been performed and zero violations can be observed.

5.1.7 Reports generated & constraints used

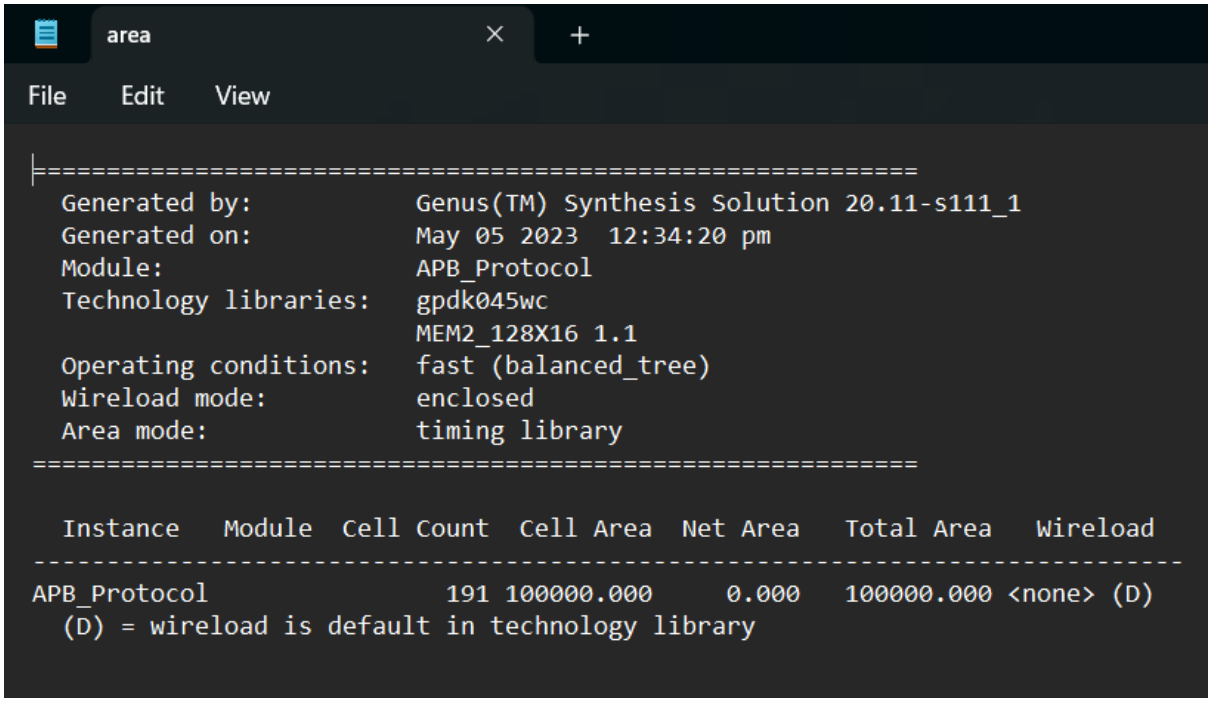


Fig 5.33: Area report

The design is observed to have a total area of 1,00,000 and a total cell count of 191.

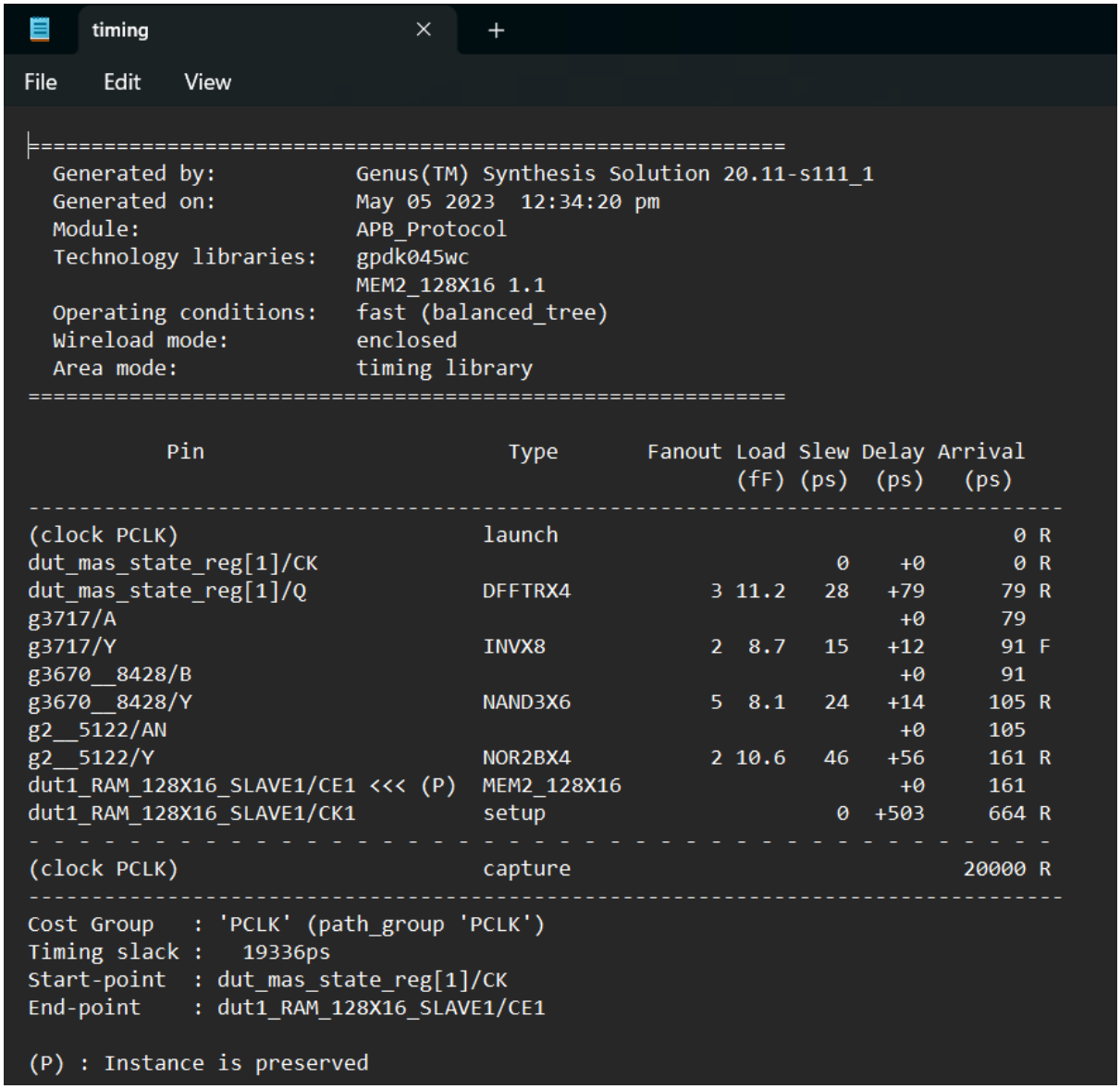


Fig 5.34: Timing report

The timing report generated post synthesis shows a positive slack of 19336ps.

timing power

File Edit View

```
Instance: /APB_Protocol
Power Unit: W
PDB Frames: /stim#0/frame#0
  ------------------------------------------------------------------------
    Category         Leakage     Internal    Switching        Total    Row%
  ------------------------------------------------------------------------
      memory     2.00000e-01  8.09800e-03  3.91910e-08  2.08098e-01  99.98%
    register     4.41935e-09  2.16706e-06  2.78128e-07  2.44960e-06   0.00%
       latch     8.58629e-08  9.85068e-06  2.15626e-06  1.20928e-05   0.01%
       logic     5.42117e-08  5.23674e-06  2.30096e-06  7.59191e-06   0.00%
        bbox     0.00000e+00  0.00000e+00  0.00000e+00  0.00000e+00   0.00%
       clock     0.00000e+00  0.00000e+00  2.63102e-05  2.63102e-05   0.01%
         pad     0.00000e+00  0.00000e+00  0.00000e+00  0.00000e+00   0.00%
          pm     0.00000e+00  0.00000e+00  0.00000e+00  0.00000e+00   0.00%
  ------------------------------------------------------------------------
    Subtotal     2.00000e-01  8.11525e-03  3.10848e-05  2.08146e-01 100.00%
  Percentage          96.09%        3.90%        0.01%      100.00% 100.00%
  ------------------------------------------------------------------------
```

Fig 5.35: Power report

constraint.sdc

File Edit View

```
create_clock -name PCLK -period 20 [get_ports PCLK]
set_input_delay -clock PCLK -max 3 <apb_write_data>
set_output_delay -clock PCLK -max 3 <apb_read_data_out>
set_clock_uncertainity 1 [get_clocks PCLK]
```

Fig 5.36: Constraints defined

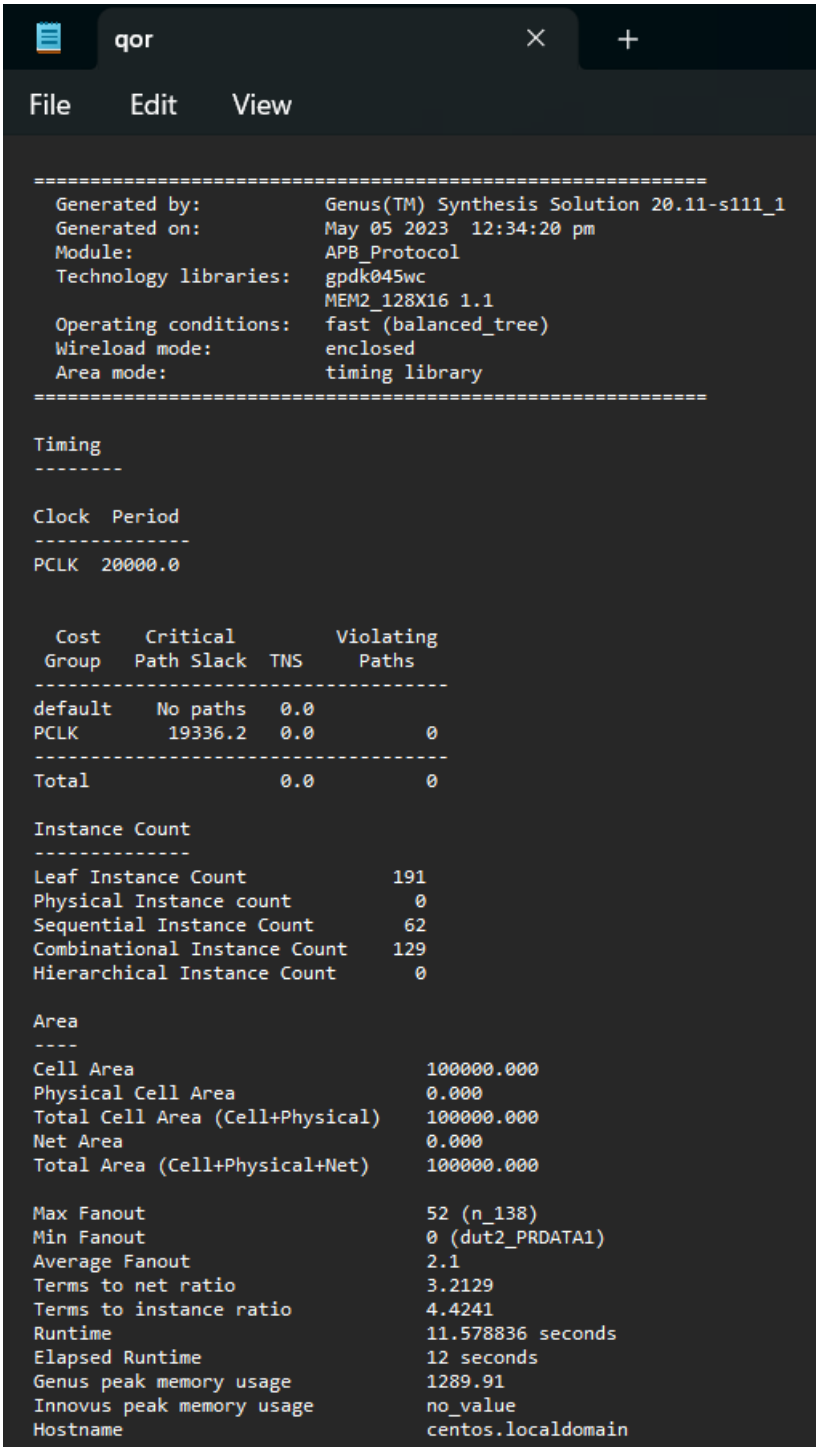


Fig 5.37: Quality of results (QoR) report

5.2 Verification of APB bridge using UVM

Sl.no	Operation type	Test case
1	Normal Operation Testing	Write on to the slave -1 (8x i.e contiguous memory)
2		Write on to the slave -2 (8x)
3		Write using generated random data
4		Read from Slave 1
5		Read from Slave 2
6	Error injection	Write transfer without write address
7		Read transfer without read address
8		Write transfer without valid write data

Table 5.1: Test Cases for Verification

The test cases are a mixture of directed and randomized values tailored in order to achieve most coverage. There are two modes of operation under test namely,

- Normal operation for reading and writing on to the slave.
- Control signals and data that would lead to Slave error.

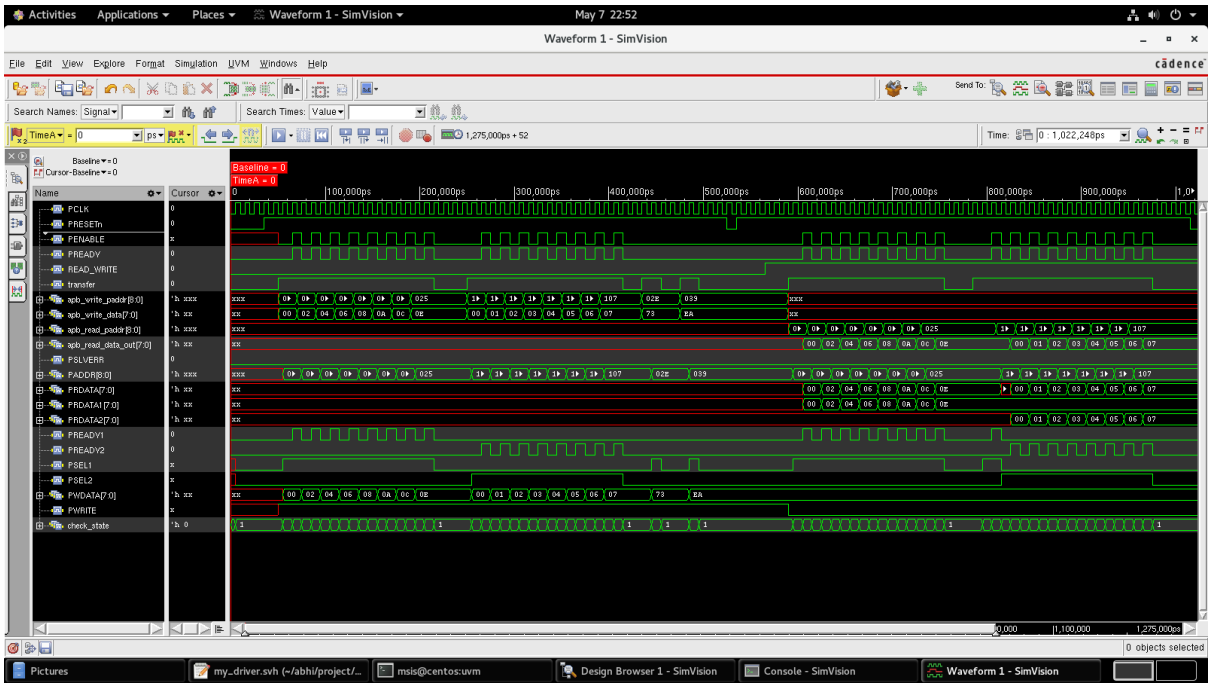


Fig 5.38: Simulation Results

Fig 5.38 is the simulation results for the test case 1-5 shown in Table 5.1. The operation is completed without any slave error.

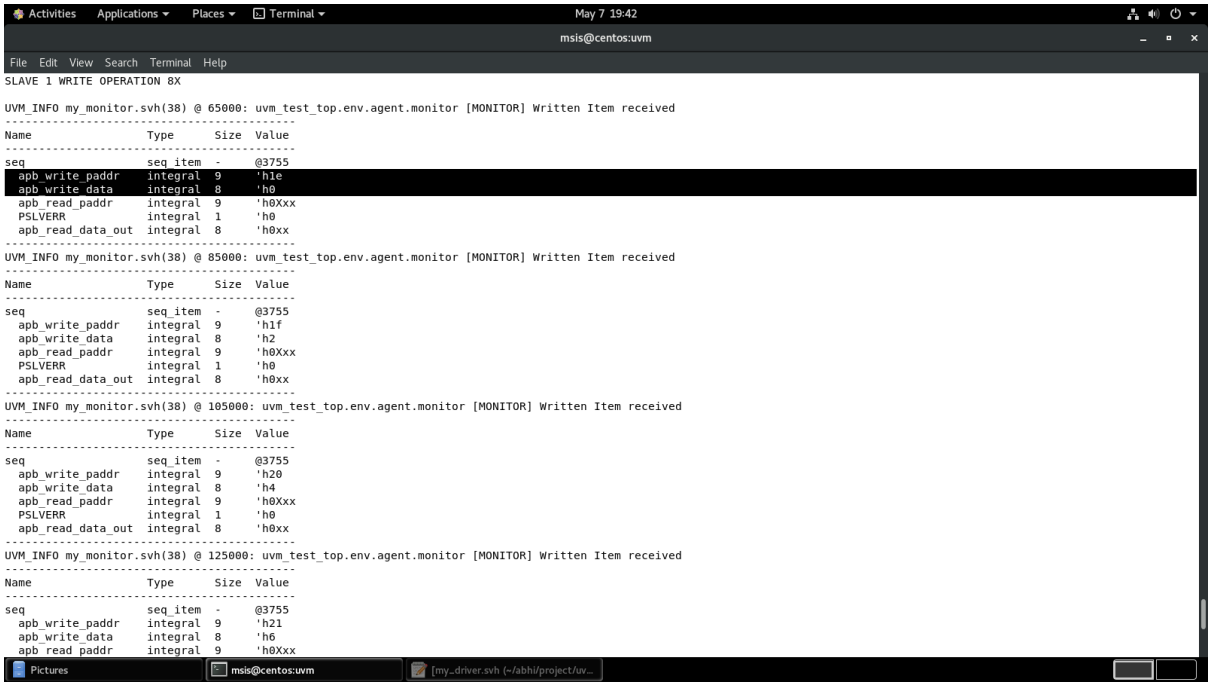


Fig 5.39: Write to Slave 1

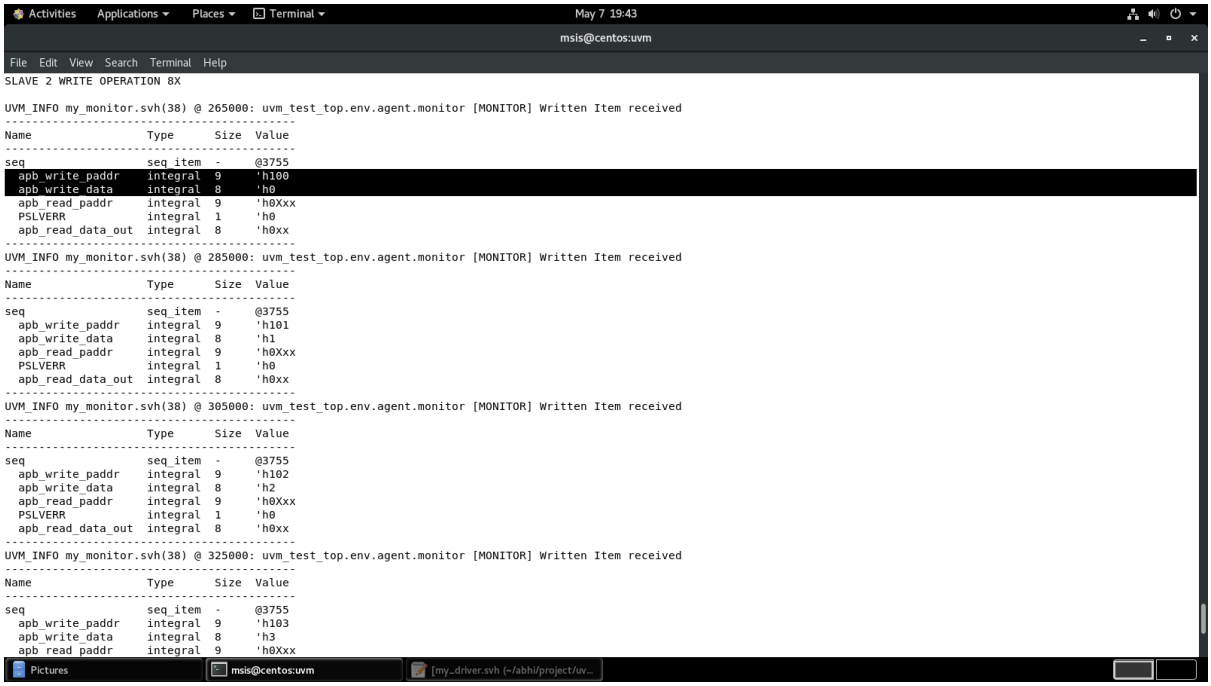


Fig 5.40: Write to Slave 2

In Write operation, data is been written in continuous memory locations for both Slave 1 and Slave2. Monitor receives the data from interface under valid condtions and displays the sequence packet as shown in Fig 5.39 and 5.40.

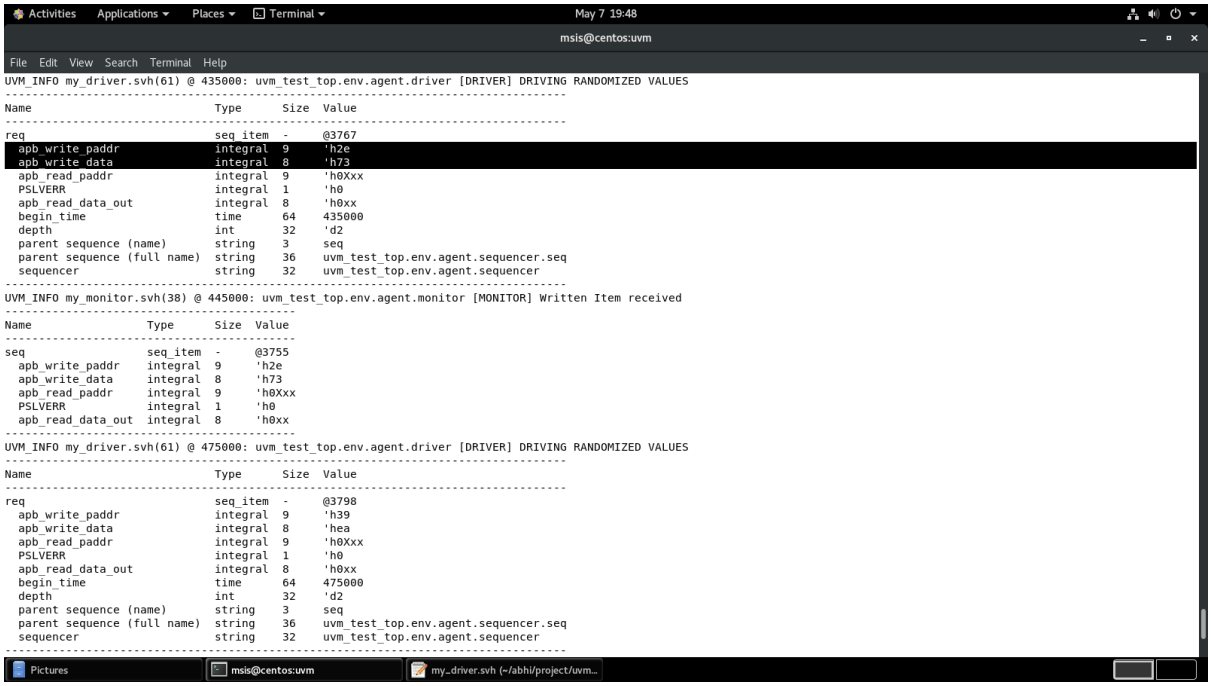


Fig 5.41: Driving randomized values

With READ_WRITE being 0 (Write mode ON), write_address and write_data have been randomized and sent from driver. Both Driver and Monitor displays the packet which can be verified.

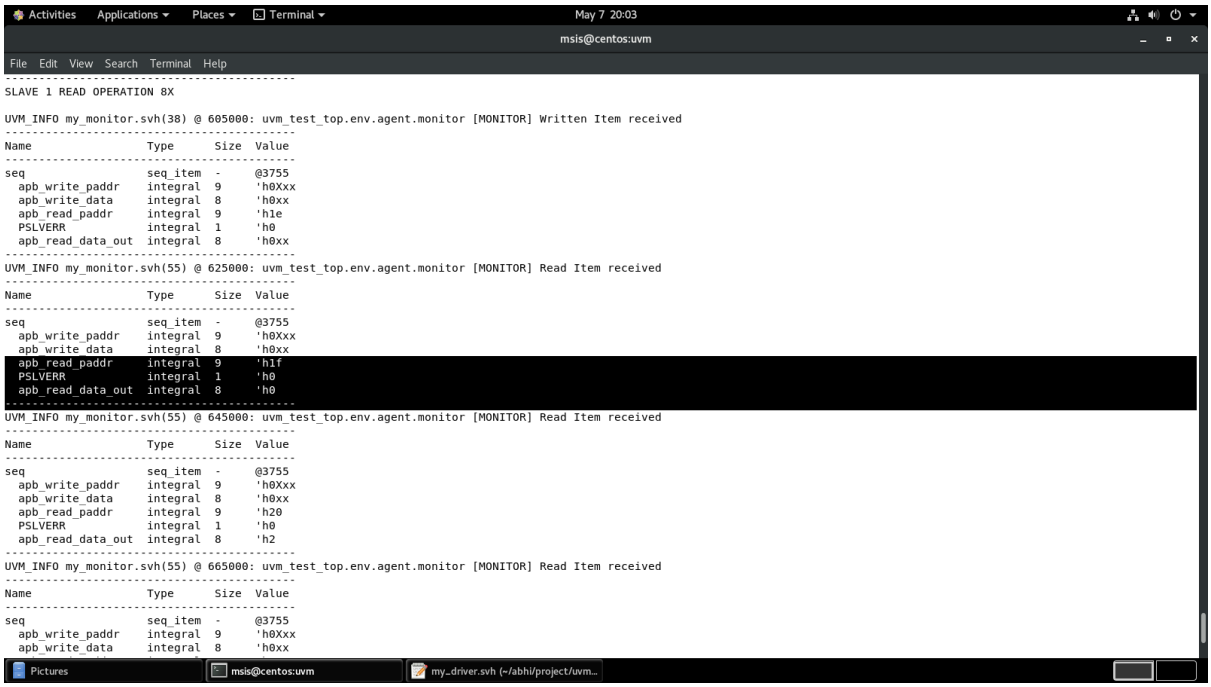


Fig 5.42: Read from Slave 1

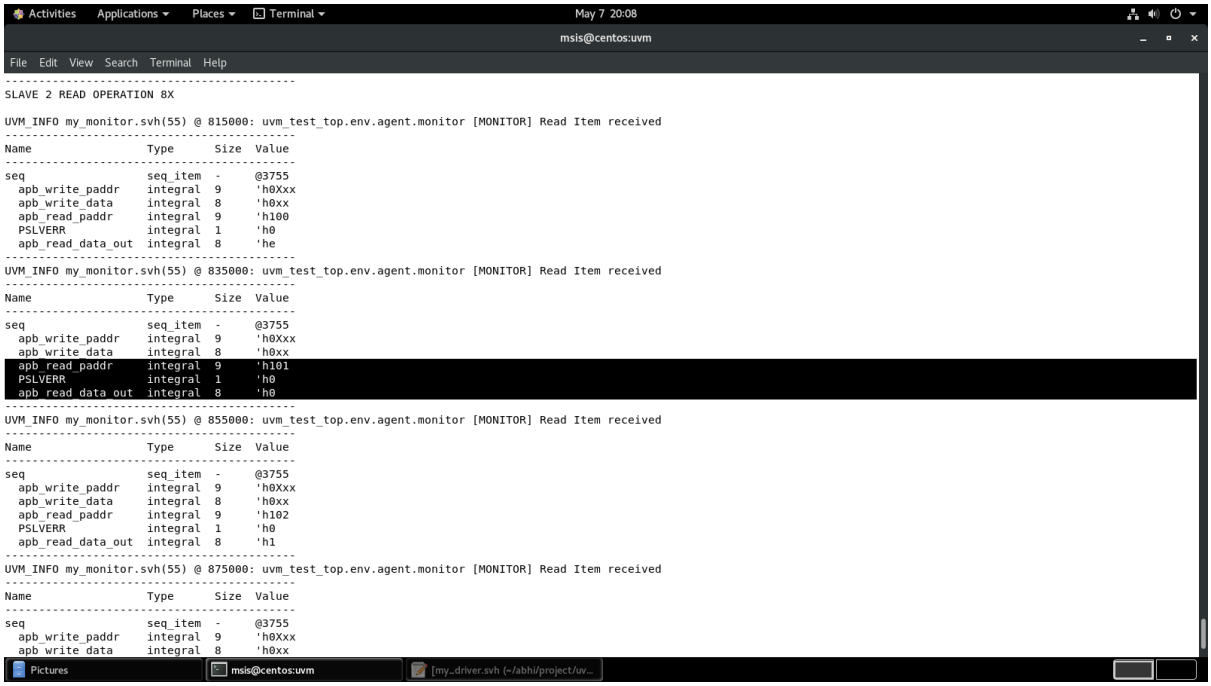


Fig 5.43: Read from Slave 2

In Read operation, data is been read in continuous memory locations from both Slave 1 and Slave2. Monitor receives the data from interface when READ_WRITE toggles and transfer signal is HIGH, it displays the sequence packet as shown in Fig 5.42 and 5.43.

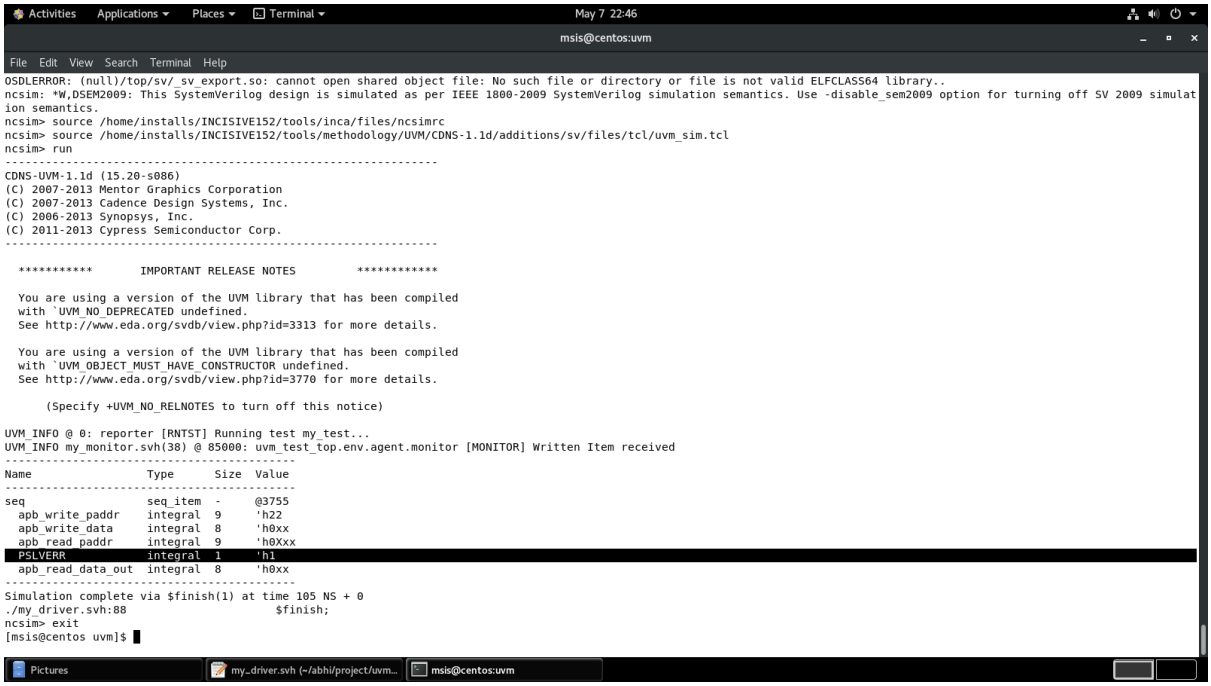


Fig 5.44: Invalid write data

With READ_WRITE being 0 (Write mode ON), when write_data provided is not valid, slave error will be HIGH indicating invalid write data.

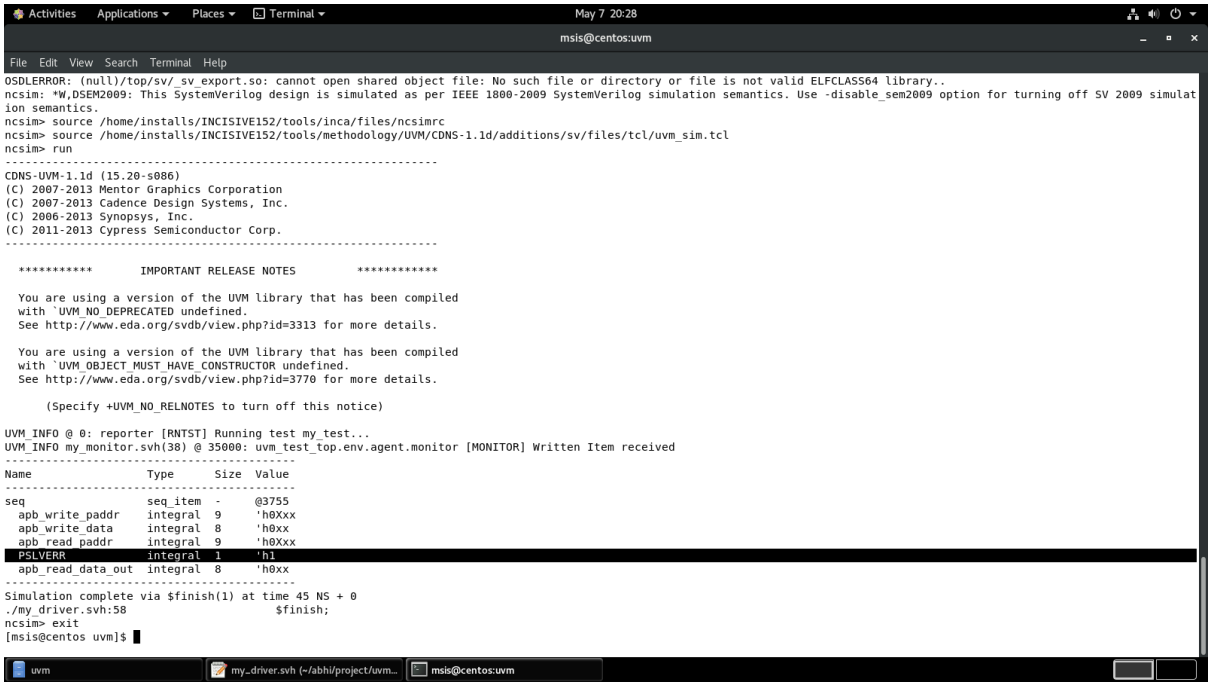


Fig 5.45: Invalid write address

With READ_WRITE being 0 (Write mode ON), when write_address provided is not valid, slave error will be HIGH indicating invalid write address.

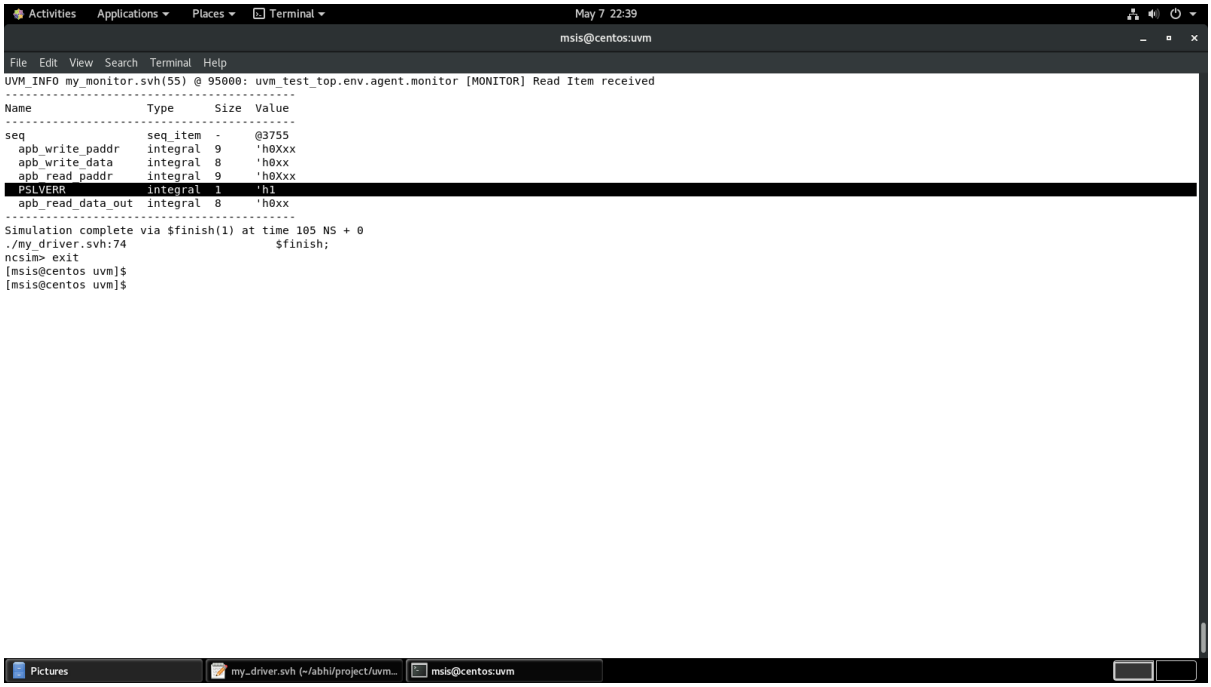


Fig 5.46: Invalid read address

With READ_WRITE being 1 (Read mode ON), when read_address provided is not valid, slave error will be HIGH indicating invalid read address.

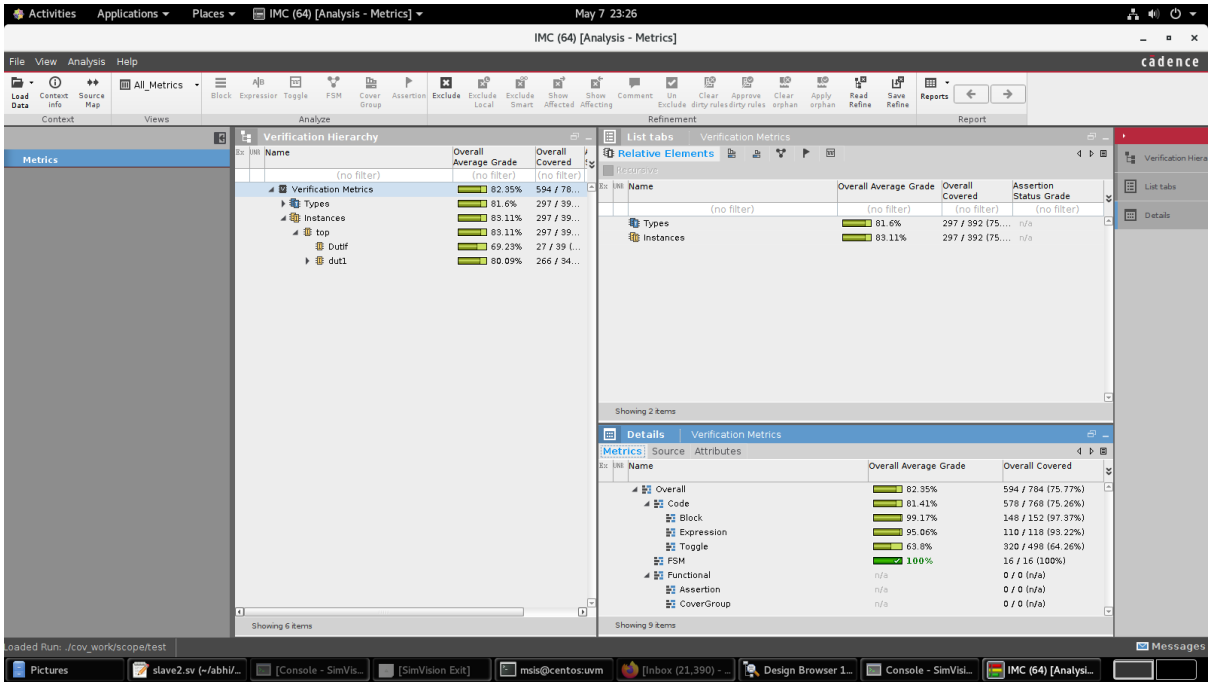


Fig 5.47: Verification Metrics

The integrated metrics centre (IMC, Cadence) has provided us with the overall Verification metrics of 82% coverage. The Toggle coverage appears to be the lowest since the randomized address and data has not covered all the memory locations.

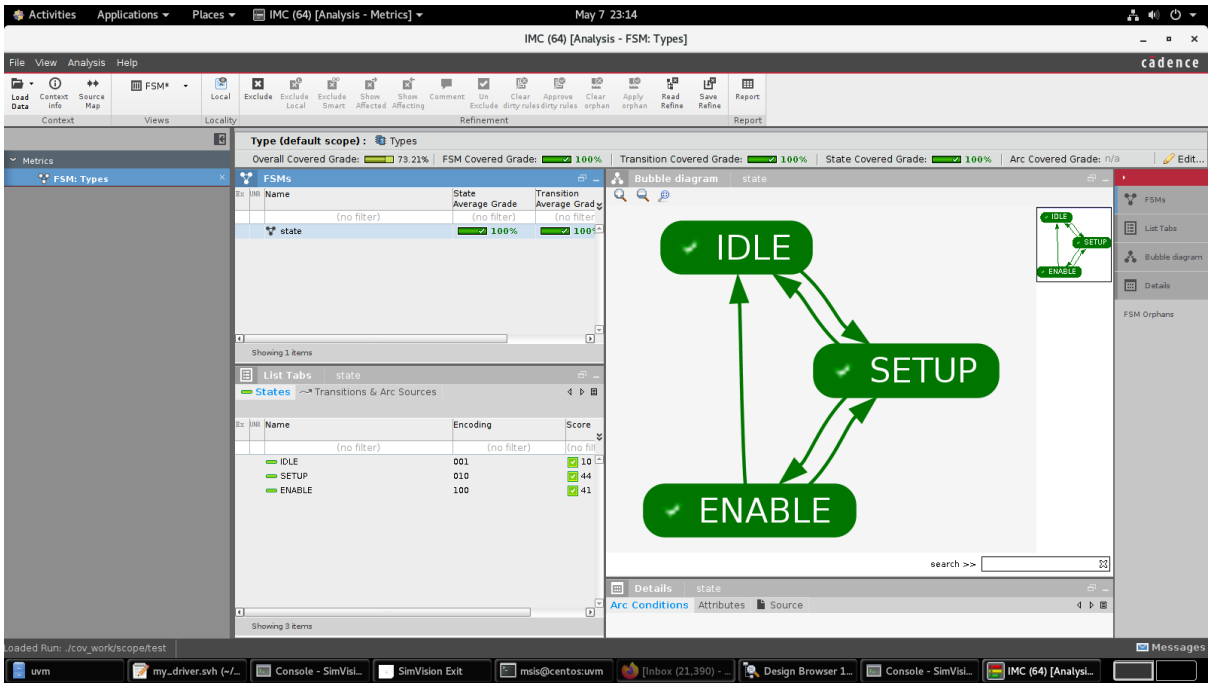


Fig 5.48: FSM Coverage

The primary goal was to achieve all the state transitions in the protocol and Fig 5.48 represents 100% FSM and transition coverage.

6. Conclusions

The Physical Design is implemented and the design is verified using Universal Verification Methodology (UVM) under different parameters and the verification metrics has been generated. The Physical Design aims to optimize the physical layout design using different methods and generate area & timing reports under different conditions like pre-CTS, post-CTS, post routing to check the design efficiency.

The UVM aims to verify the design by providing random inputs and test the design for various corner cases and check the design efficiency for several input combinations and determining the coverage.

7. References

- AMBA APB Protocol Specification – ARM
- Verification of Advanced Peripheral Bus Protocol (APB V2.0) - Meghana Jain H K1, Dr. Punith Kumar M B, Student, Professor, Dept of Electronics and Communication Engineering, P.E.S College of Engineering, Karnataka, India