



**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**



**MANIPAL SCHOOL OF INFORMATION SCIENCES**  
MANIPAL  
*(A constituent unit of MAHE, Manipal)*

## Table of Contents

<b>1. Introduction .....</b>	1
<b>2. Objectives .....</b>	2
<b>3. Specifications.....</b>	2
<b>4. Architecture/Design/Block or Schematic diagram/Flowchart/Proposed modules ....</b>	3
<b>4.1 AMBA BUS architecture.....</b>	3
<b>4.2 Interface diagram.....</b>	5
<b>4.3 Finite State diagram .....</b>	6
<b>5. Work done/Results .....</b>	7
<b>5.1 Physical Design.....</b>	7
<b>  5.1.1 Design import .....</b>	7
<b>  5.1.2 Floor planning .....</b>	9
<b>  5.1.3 Power Planning .....</b>	10
<b>  5.1.4 Placement.....</b>	14
<b>  5.1.5 Routing.....</b>	19
<b>  5.1.6 DRC check .....</b>	23
<b>  5.1.7 Reports generated &amp; constraints used .....</b>	23
<b>5.2 Verification of APB bridge using UVM .....</b>	27
<b>6. Conclusions.....</b>	33
<b>7. References.....</b>	33

## LIST OF FIGURES

Figure 4.1: AMBA Bus architecture.....	3
Figure 4.2: APB Interface diagram.....	5
Figure 4.3: Finite State Diagram.....	6
Figure 5.1: Importing design.....	7
Figure 5.2: LEF files added.....	8
Figure 5.3: MMMC browser.....	8
Figure 5.4: Initial floorplan.....	9
Figure 5.5: Floorplan specifications.....	9
Figure 5.6: Floorplan with macros placement.....	10
Figure 5.7: Global net connections.....	10
Figure 5.8: Adding rings.....	11
Figure 5.9: Ring placement around core boundary .....	11
Figure 5.10: Adding VDD stripes.....	12
Figure 5.11: Adding VSS stripes.....	12
Figure 5.12: Stripes placement.....	13
Figure 5.13: Adding VDD & VSS railings.....	13
Figure 5.14: VDD & VSS railings.....	14
Figure 5.15: Adding standard cells.....	14
Figure 5.16: Standard cells and IO's placed.....	15
Figure 5.17: Standard cells placement without wiring.....	15
Figure 5.18: Pre-CTS setup timing report.....	16
Figure 5.19: Pre-CTS hold timing report.....	16
Figure 5.20: Initiating CTS.....	17
Figure 5.21: CTS generated.....	17
Figure 5.22: Post-CTS design optimization.....	18
Figure 5.23: Post-CTS setup timing report.....	18
Figure 5.24: Post-CTS hold timing report.....	19
Figure 5.25: Routing specifications.....	19
Figure 5.26: Post routing design.....	20
Figure 5.27: Post routing setup timing report.....	20
Figure 5.28: Post routing hold timing report.....	21
Figure 5.29: Adding filers.....	21

Figure 5.30: After adding fillers.....	22
Figure 5.31: Final Physical Design.....	22
Figure 5.32: DRC check result.....	23
Figure 5.33: Area report.....	23
Figure 5.34: Timing report.....	24
Figure 5.35: Power report.....	25
Figure 5.36: Constraints defined.....	25
Figure 5.37: Quality of Results(QoR) report.....	26
Figure 5.38: Simulation results.....	27
Figure 5.39: Write to slave 1.....	28
Figure 5.40: Write to slave 2.....	28
Figure 5.41: Driving randomized values.....	29
Figure 5.42: Read from slave 1.....	29
Figure 5.43: Read from slave 2.....	30
Figure 5.44: Invalid write data.....	30
Figure 5.45: Invalid write address.....	31
Figure 5.46: Invalid read address.....	31
Figure 5.47: Verification metrics.....	32
Figure 5.48: FSM coverage.....	32

## **LIST OF TABLES**

Table 4.1: Pin description of AMBA APB bridge.....	4
Table 5.1: Test Cases for Verification .....	27

## **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

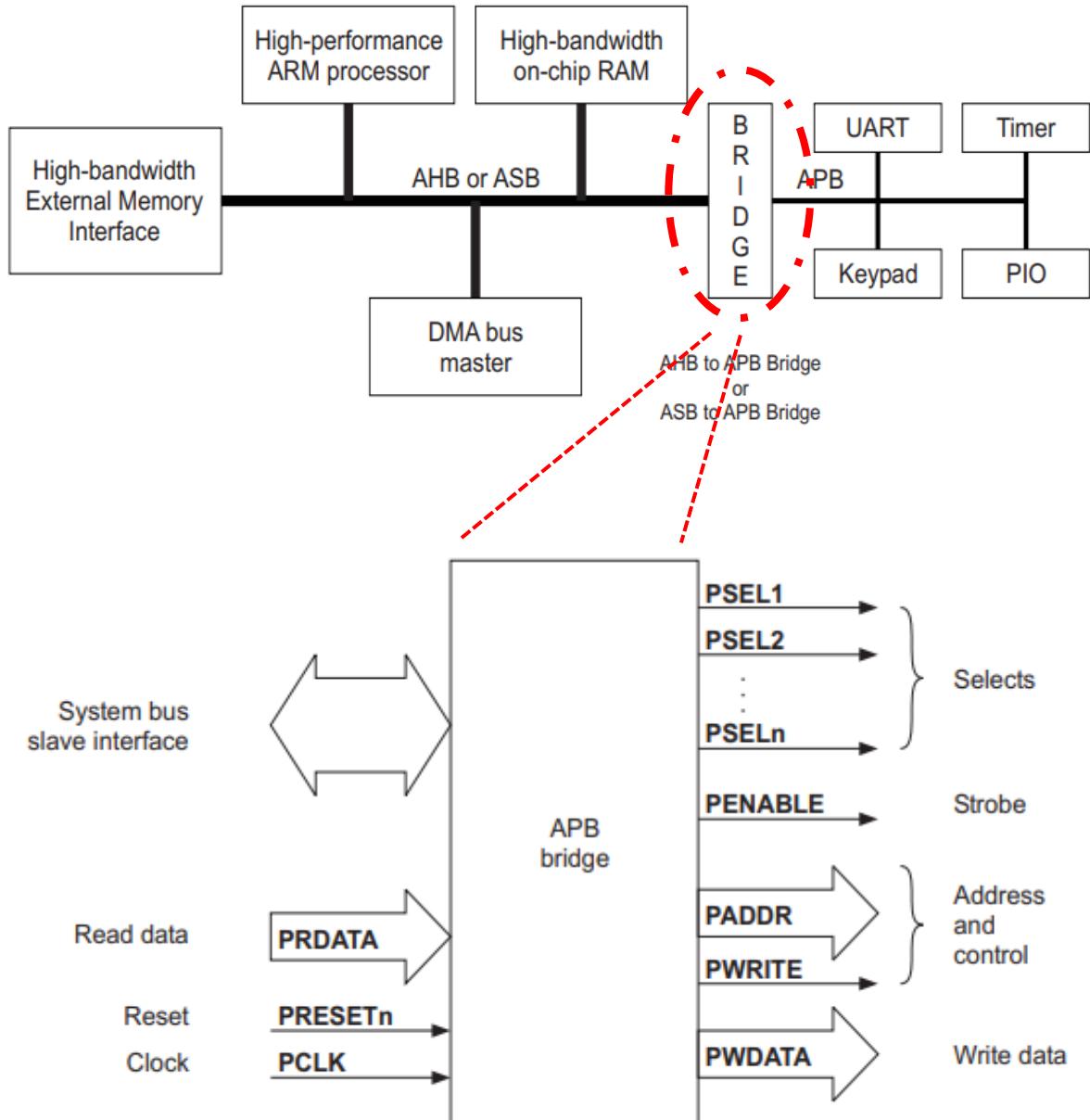


Fig 4.1: AMBA bus architecture

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.

<b>Signal</b>	<b>Source</b>	<b>Description</b>	<b>Width (bit)</b>
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 2 <sup>nd</sup> 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

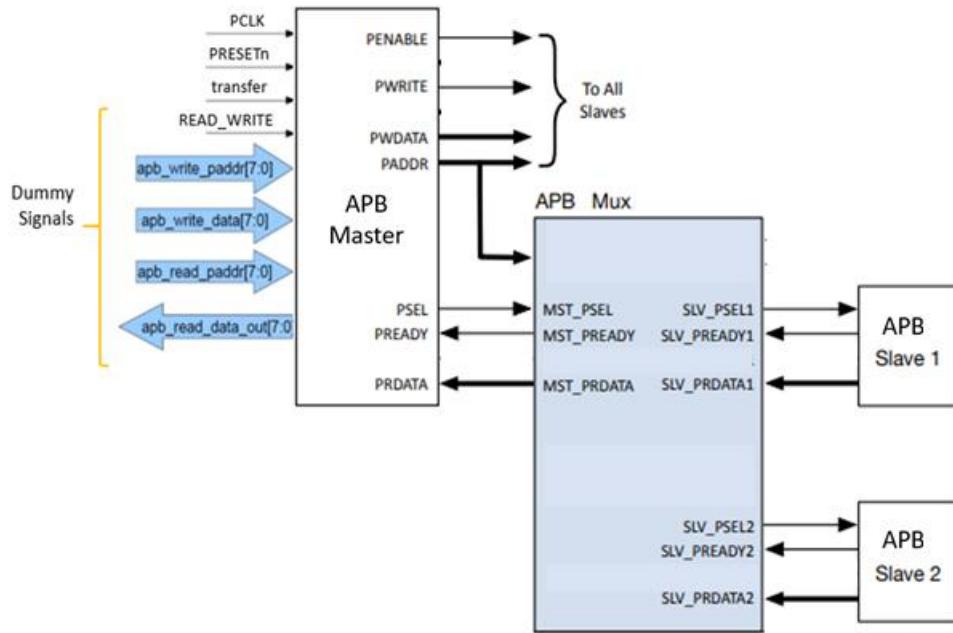


Fig 4.2 APB Interface diagram

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

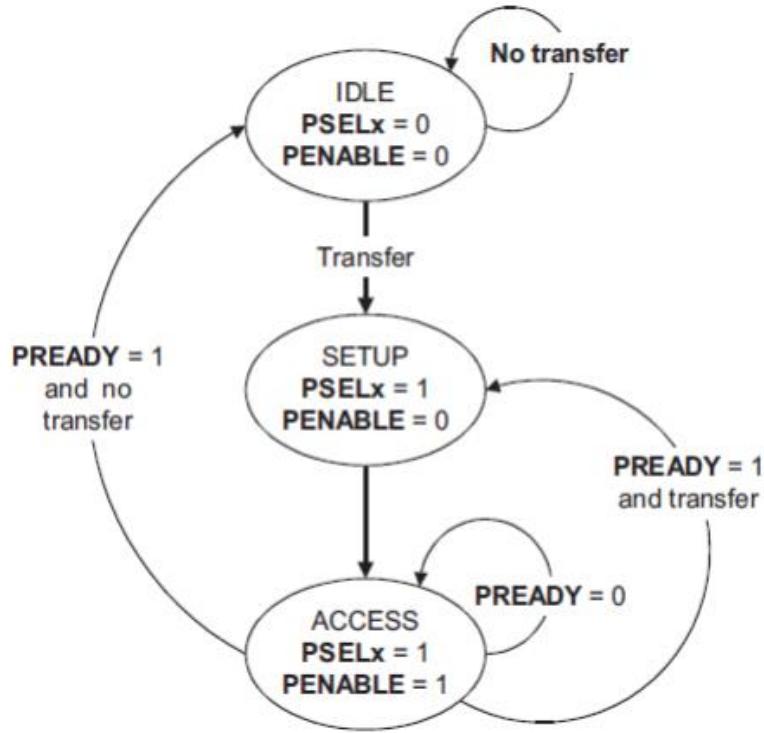


Fig 4.3: Finite state diagram

### FSM states

#### 1. Idle state

This state indicates that no operation is being performed. All signals - PSEL, PENABLE, 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, PWpdata 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

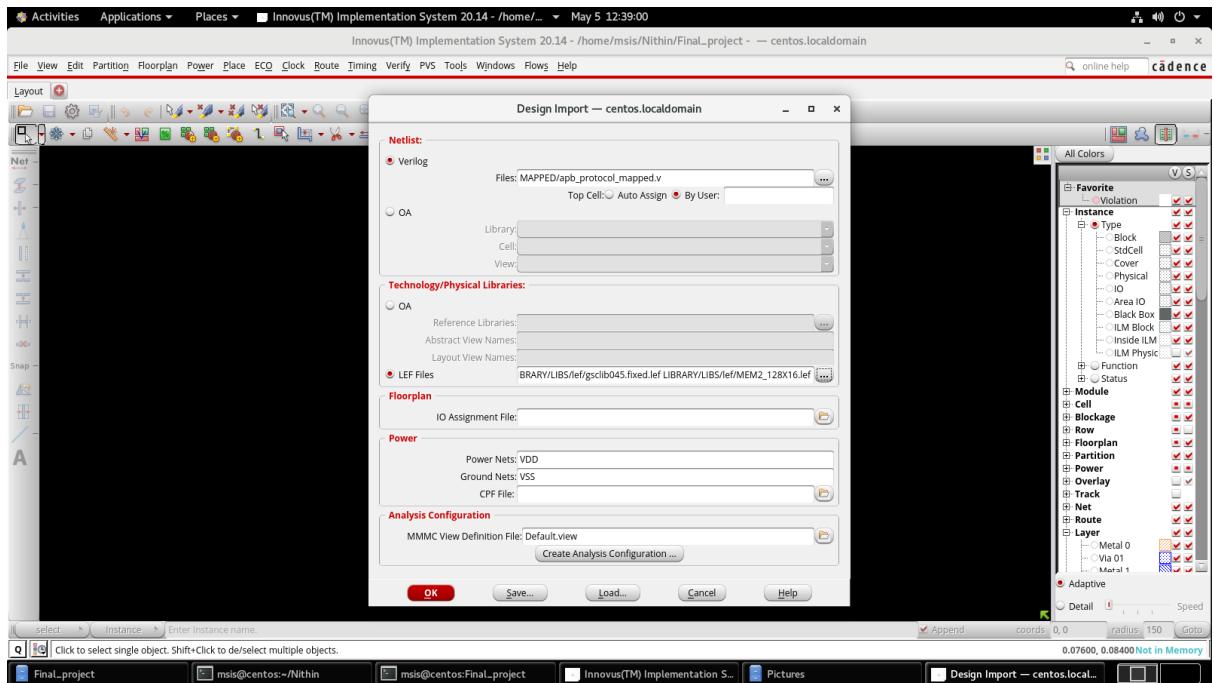


Fig 5.1: Importing design

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

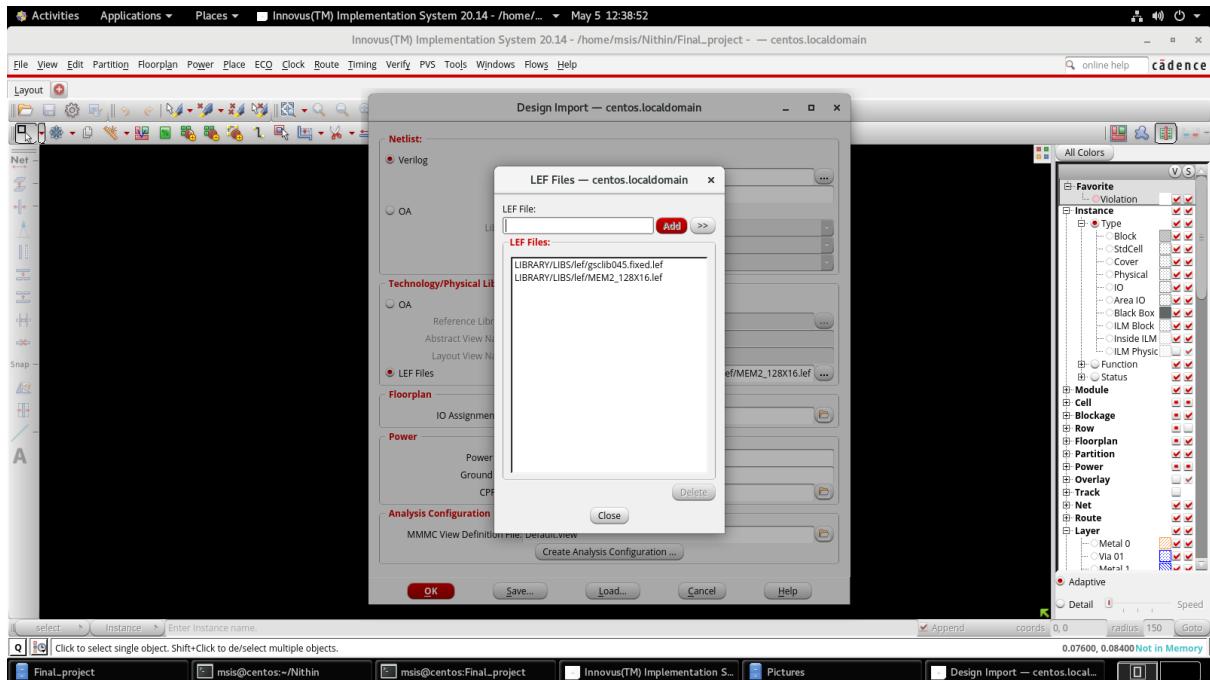


Fig 5.2 LEF files added

Two LEF files added are “gsclib045.fixed.lef” and “MEM2\_128x16.lef”

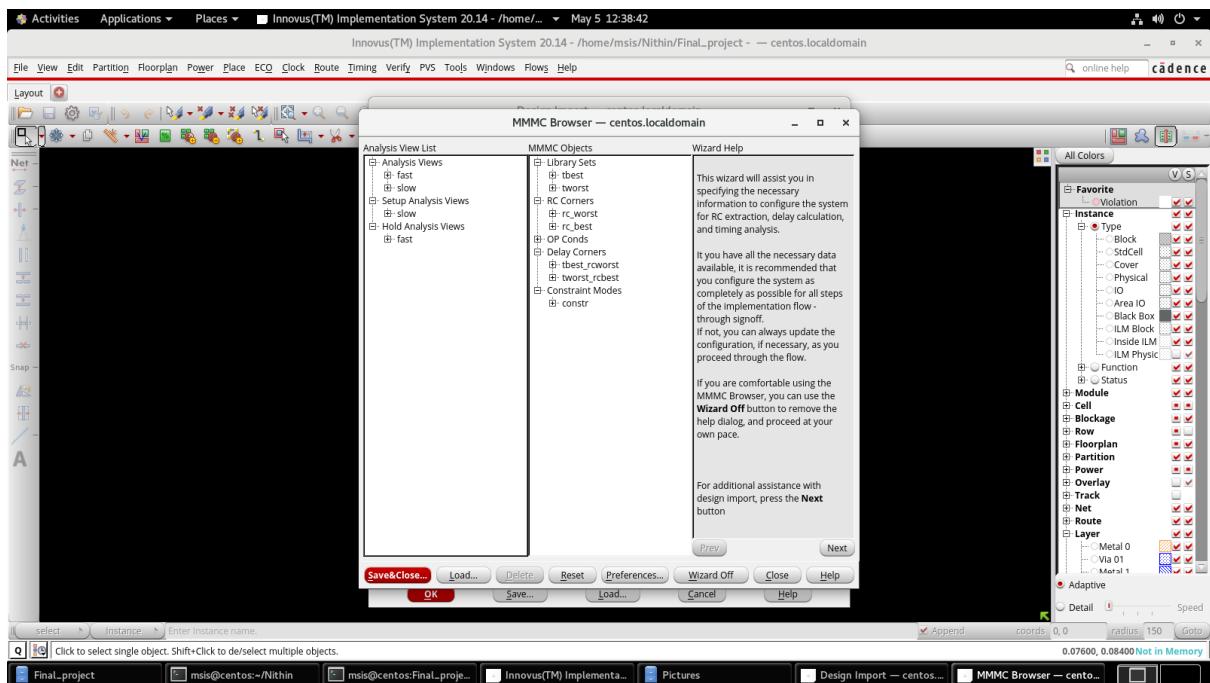


Fig 5.3: MMMC Browser

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

## 5.1.2 Floor planning

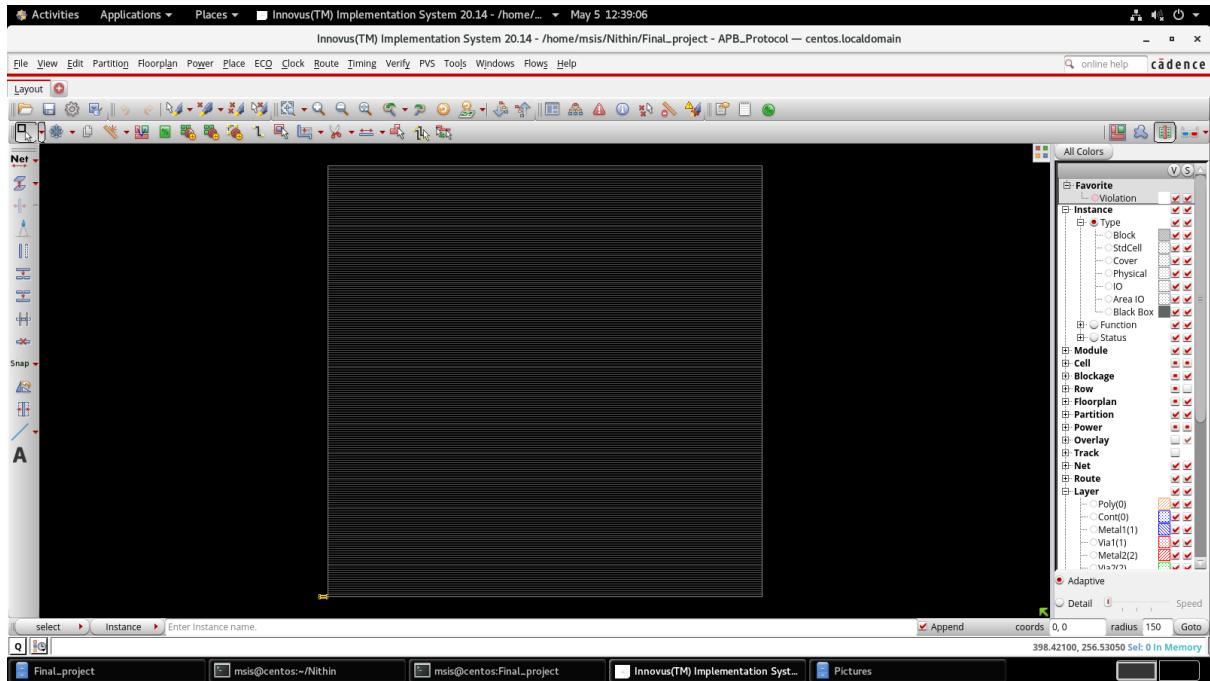


Fig 5.4: Initial floorplan

Initial floorplan after completing design import.

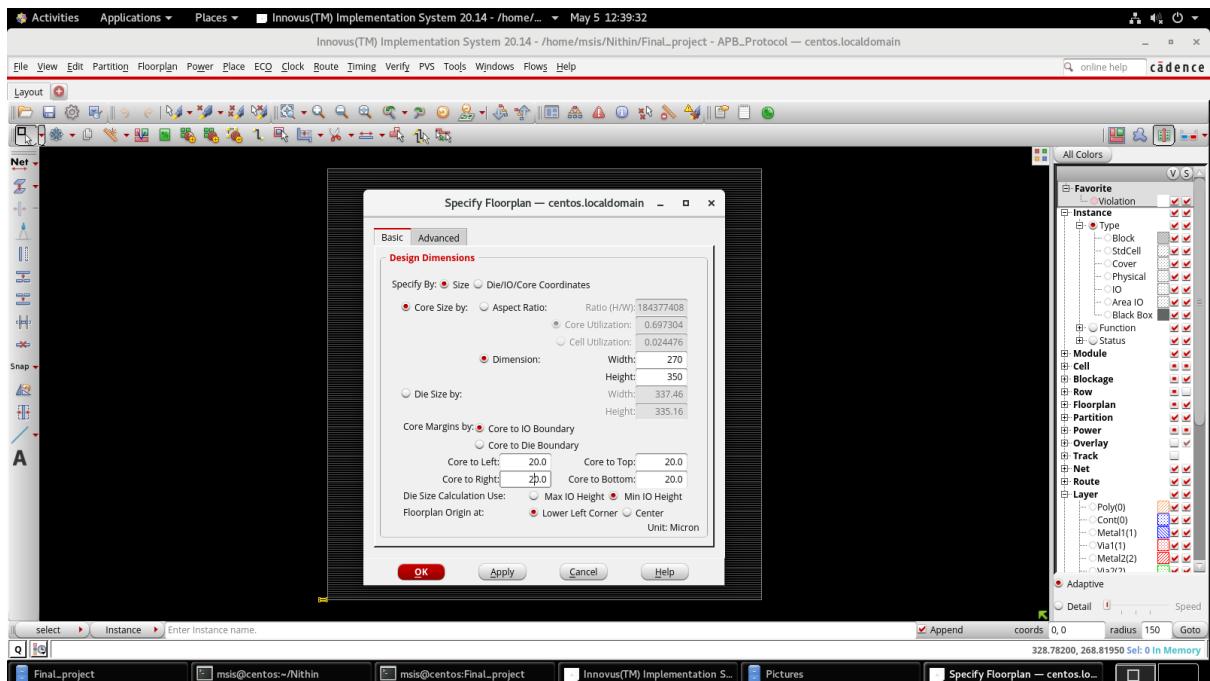


Fig 5.5: Floorplan specifications

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.

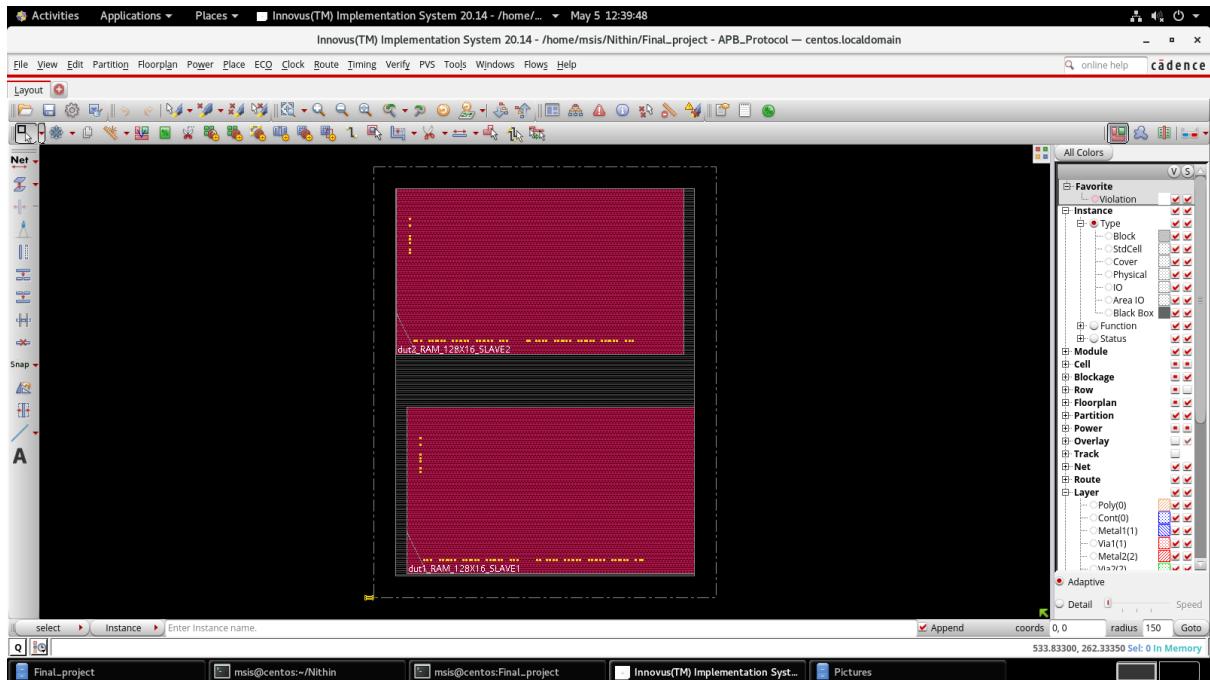


Fig 5.6: Floorplan after macros placement

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

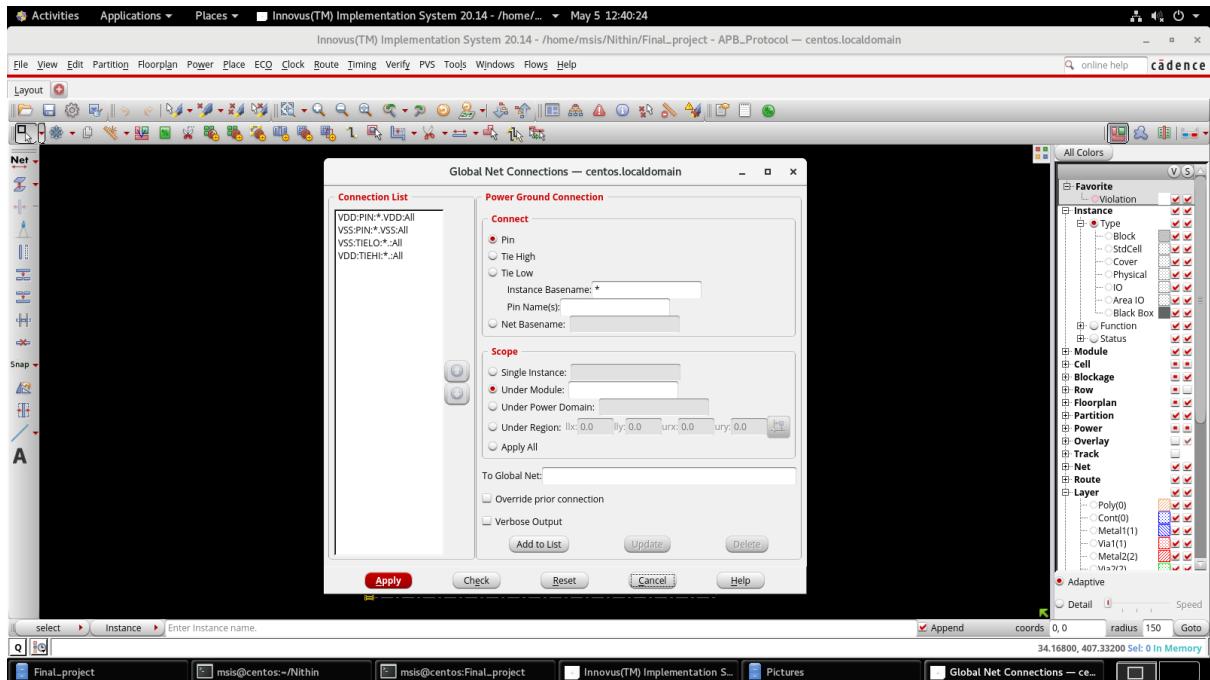


Fig 5.7: Global Net Connections

Creating Global Net Connections for VDD & VSS

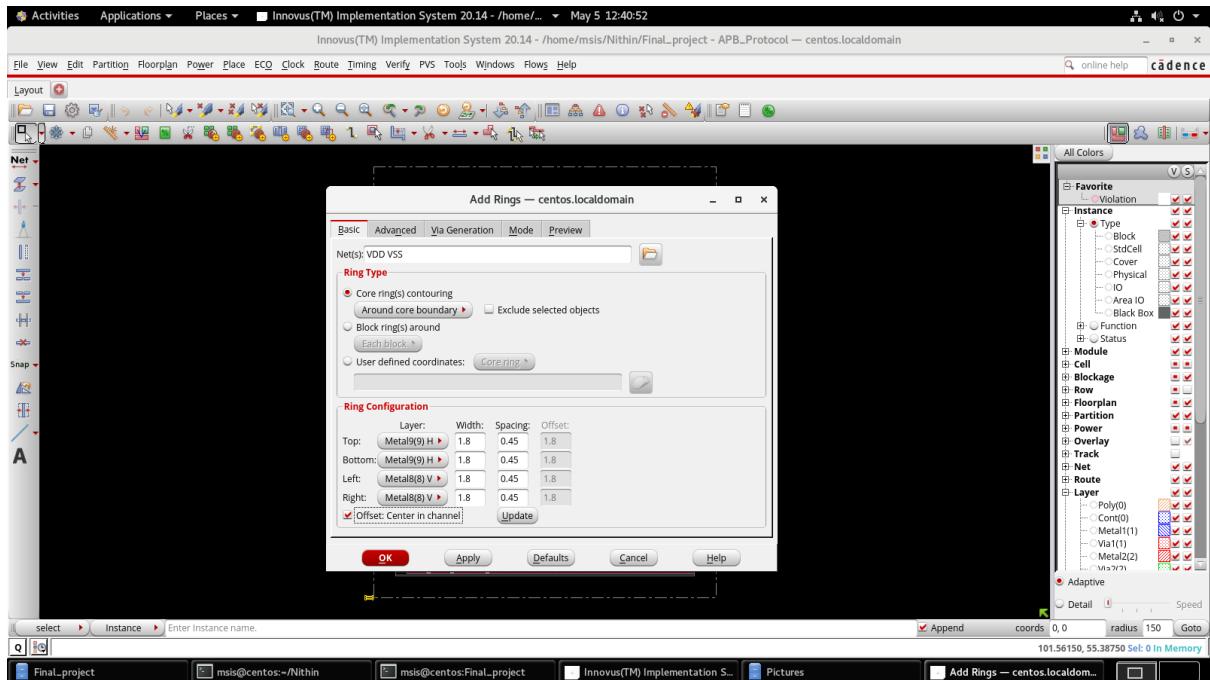


Fig 5.8: Adding Rings

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

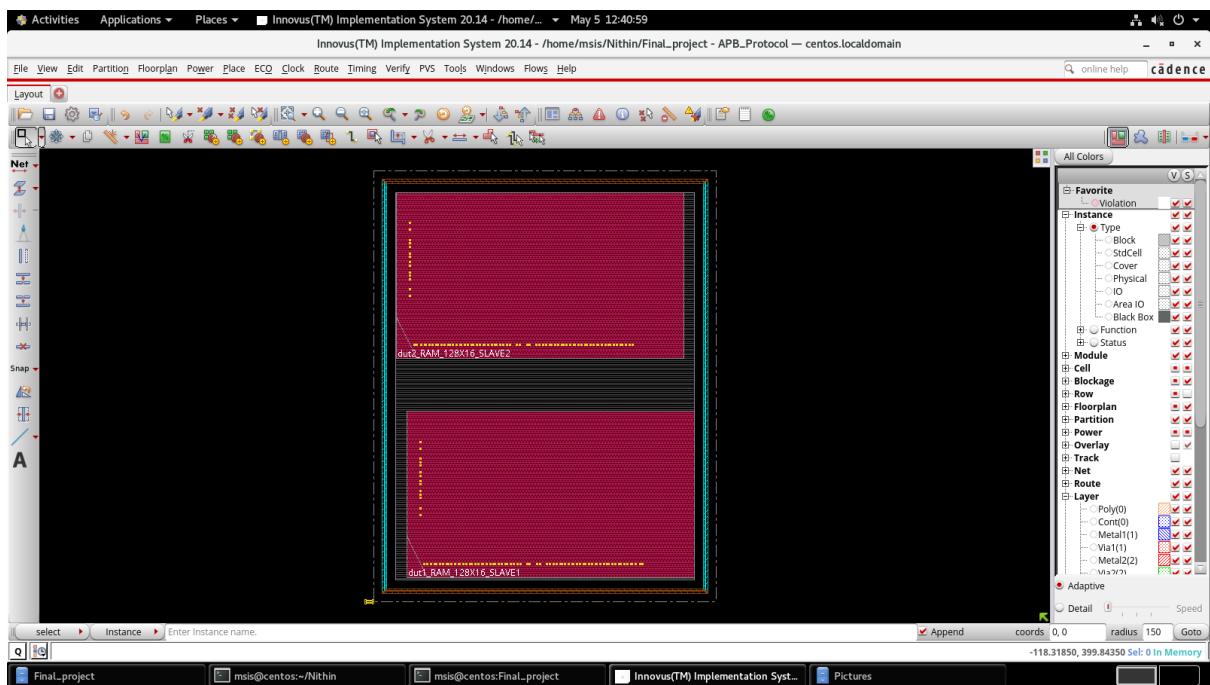


Fig 5.9: Ring placement around core boundary

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

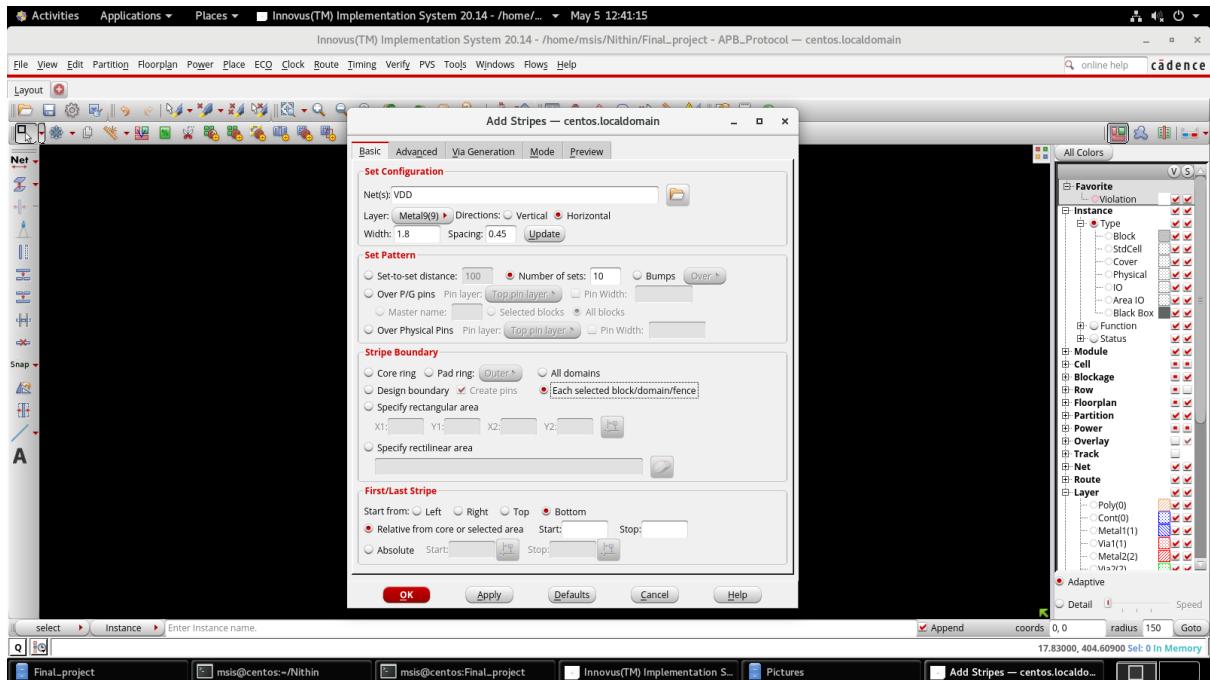


Fig 5.10: Adding VDD stripes

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

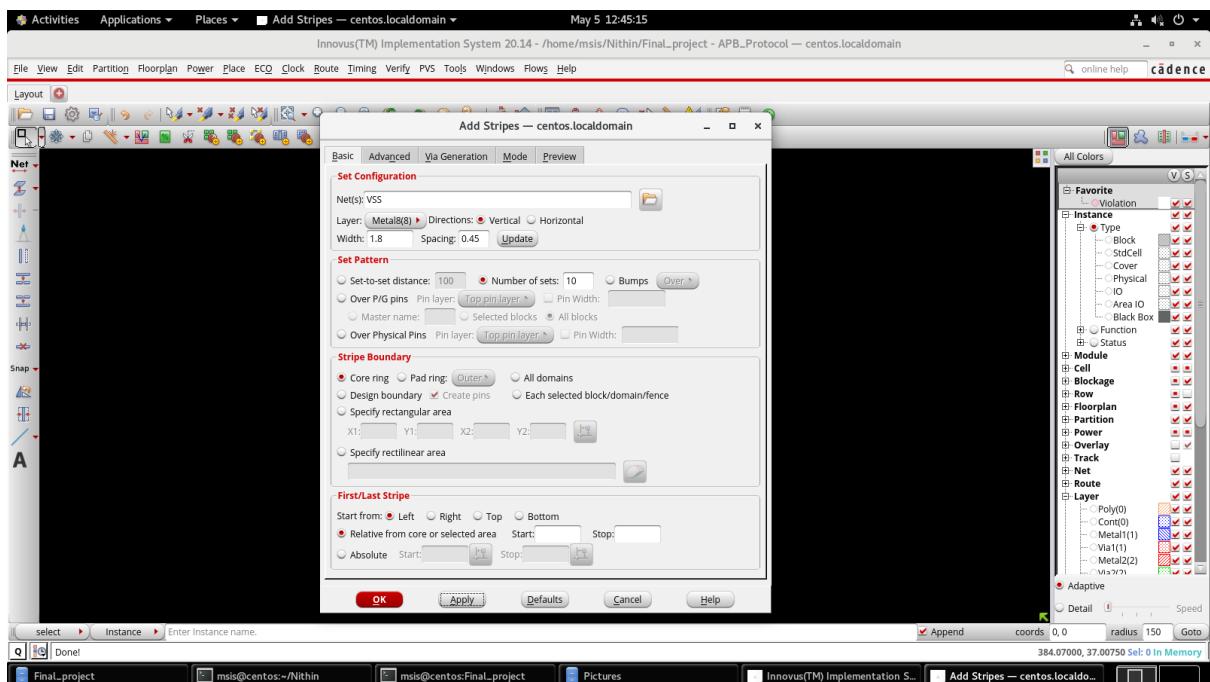


Fig 5.11: Adding VSS stripes

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

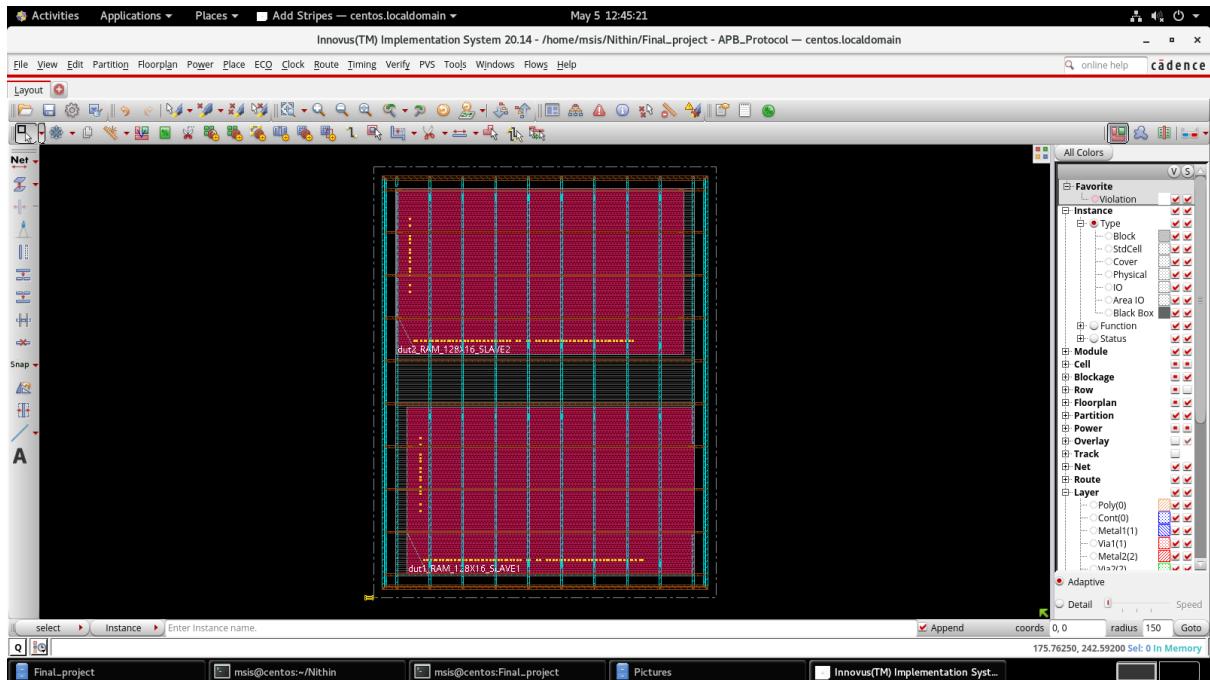


Fig 5.12: Stripes placement

The VDD & VSS stripes have been placed.

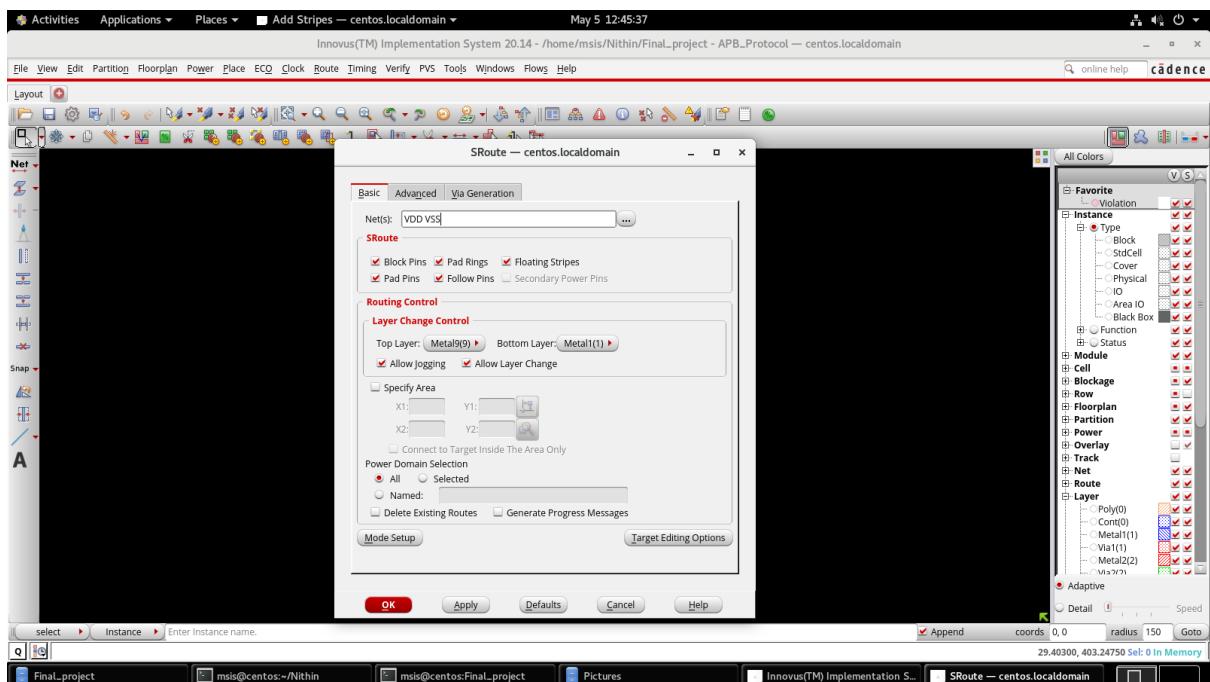


Fig 5.13: Adding VDD & VSS railings

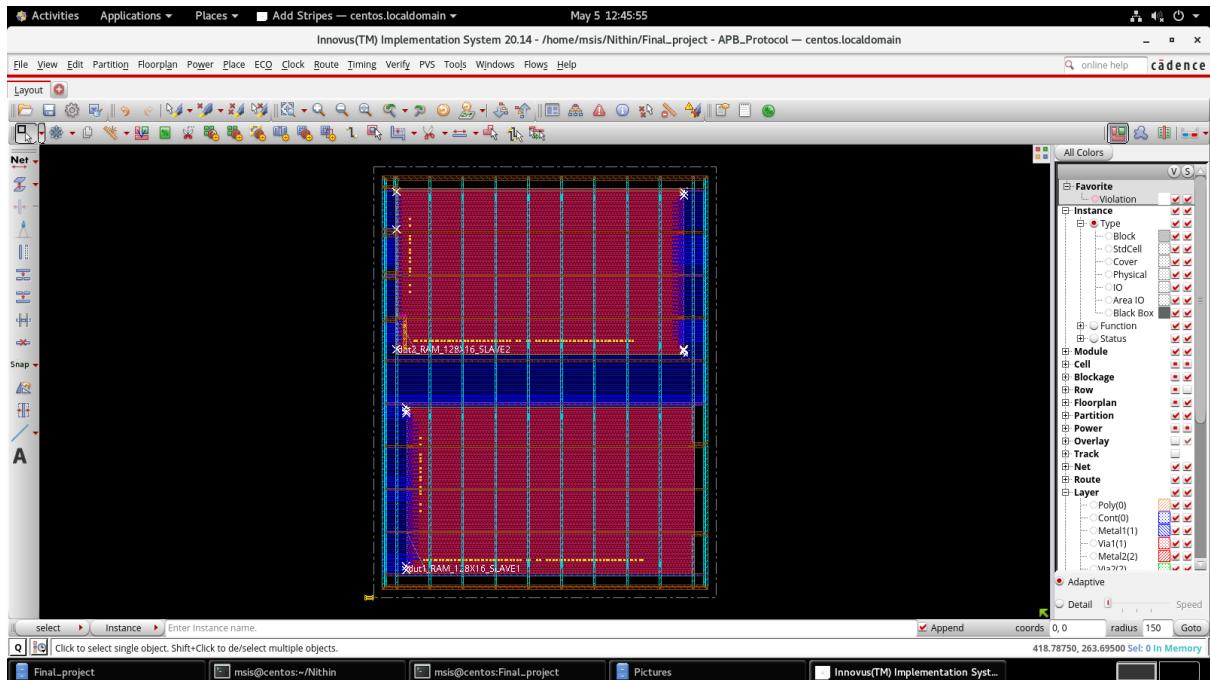


Fig 5.14: VDD & VSS railings

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

### 5.1.4 Placement

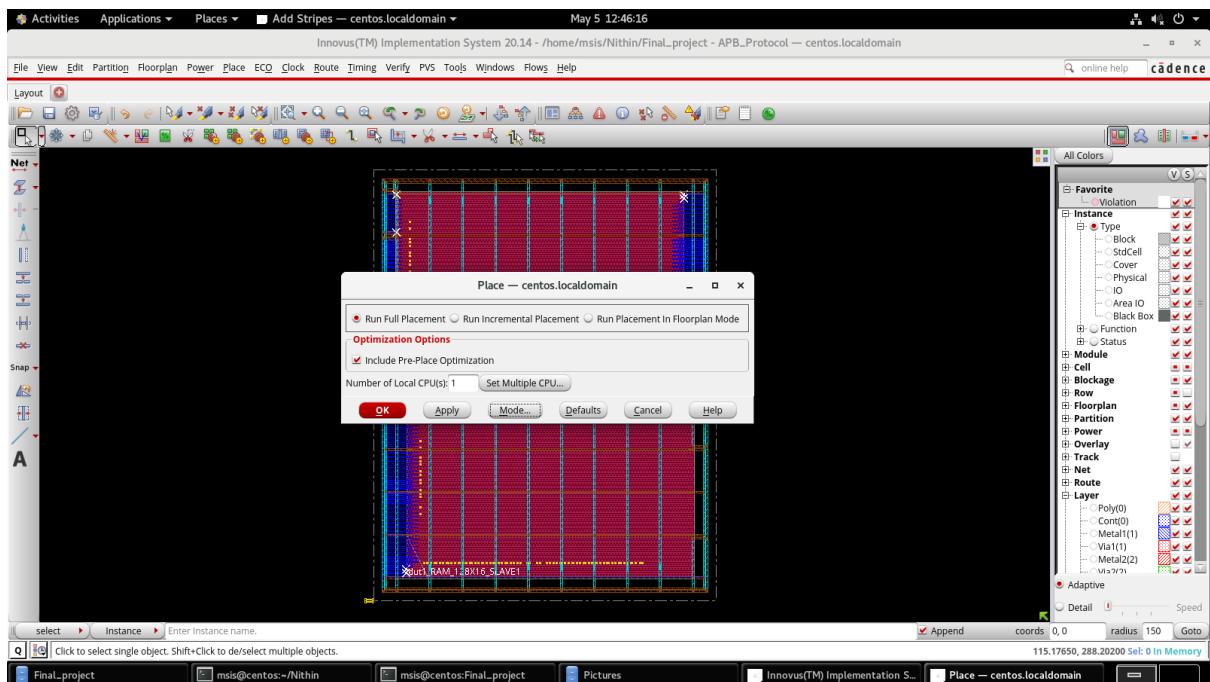


Fig 5.15: Adding standard cells

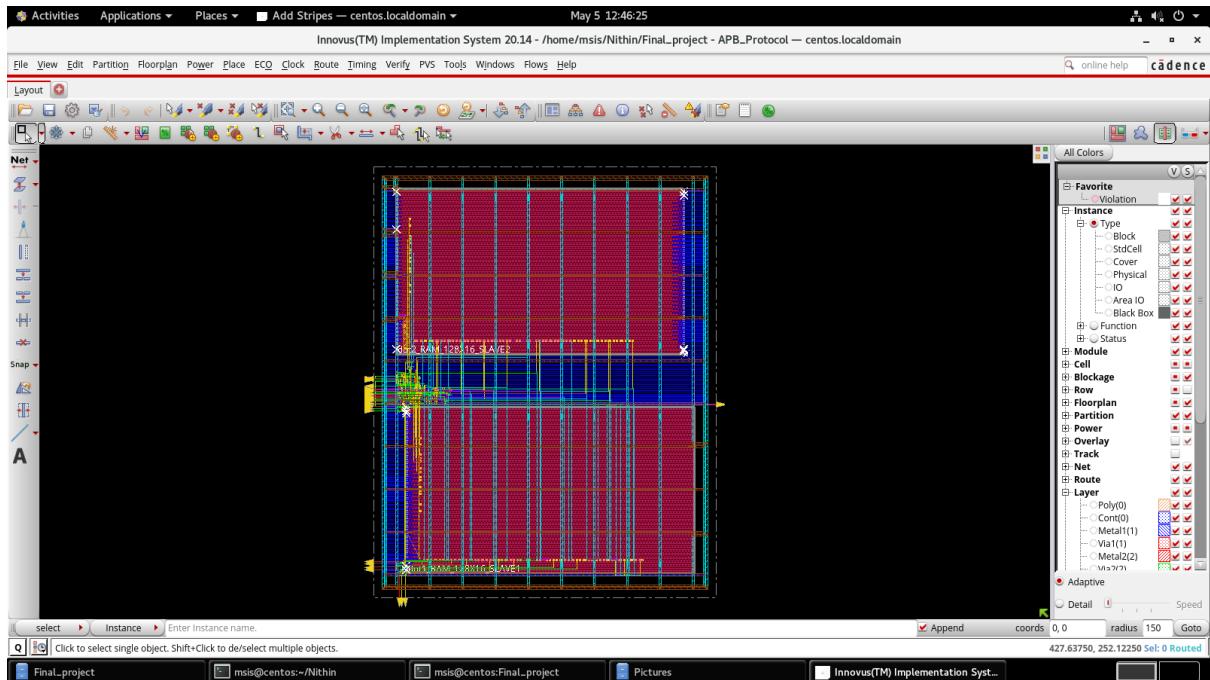


Fig 5.16: Standard cells and IO's placed

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

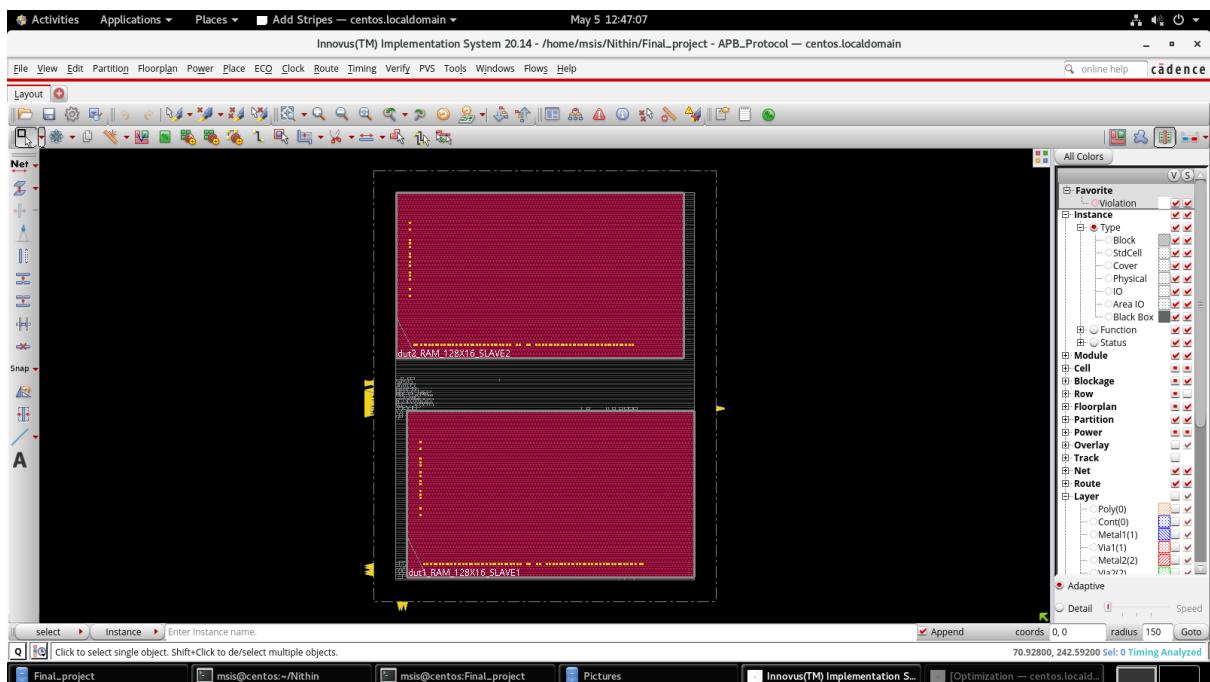


Fig 5.17: Standard cells placement without wiring

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

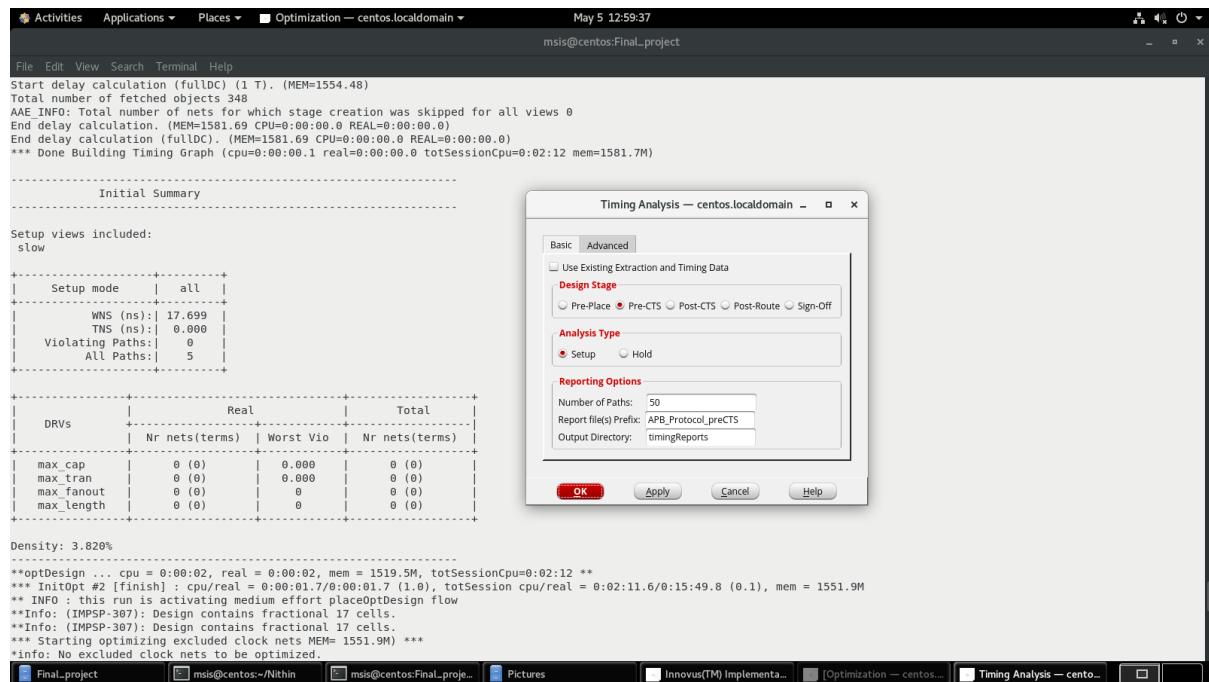


Fig 5.18: Pre-CTS setup timing report

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

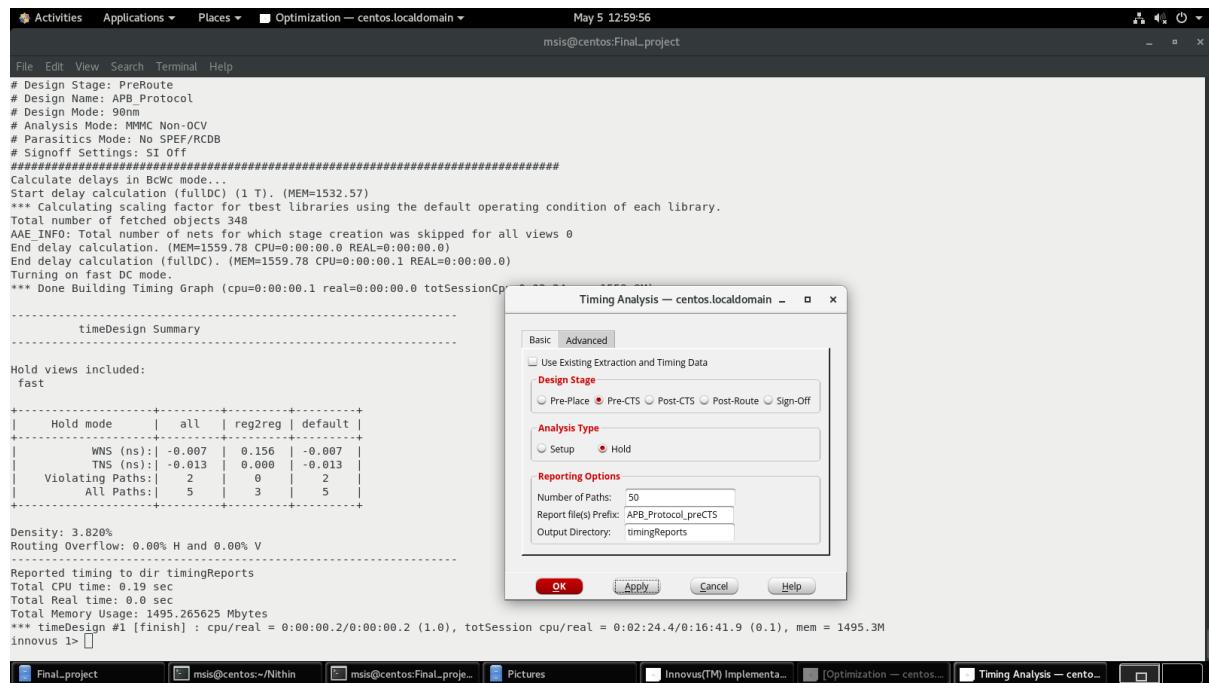


Fig 5.19: Pre-CTS hold timing report

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

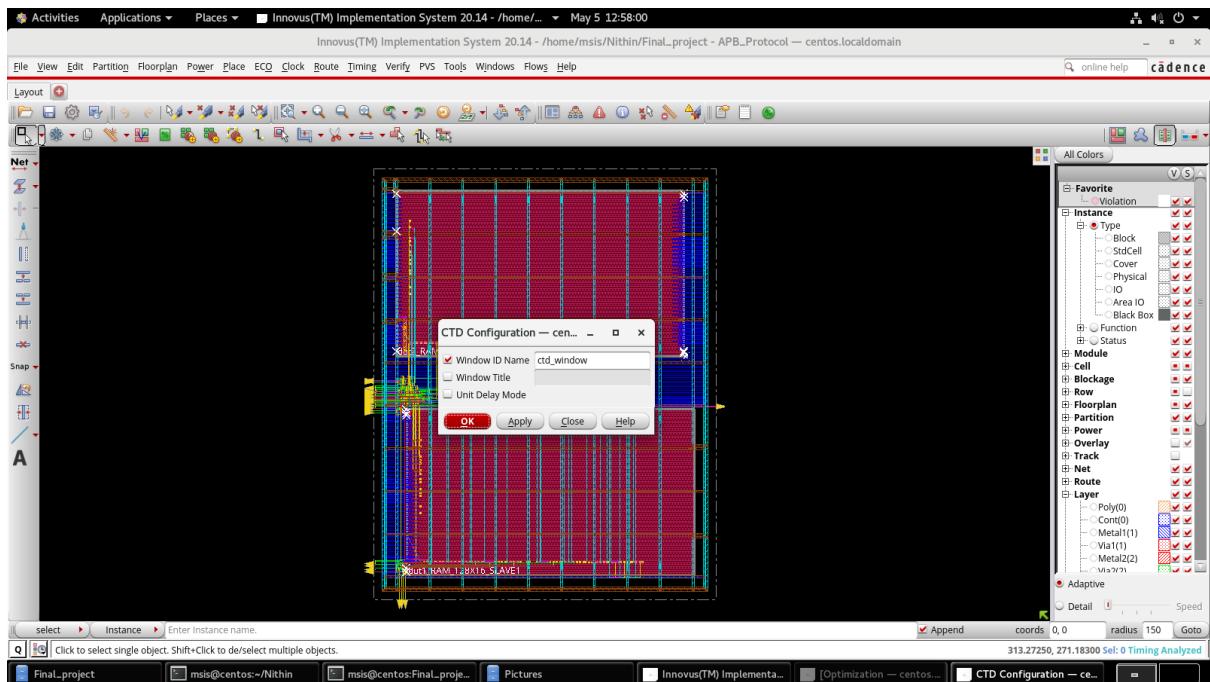


Fig 5.20: Initiating CTS

Clock Tree Synthesis (CTS) has been initiated.

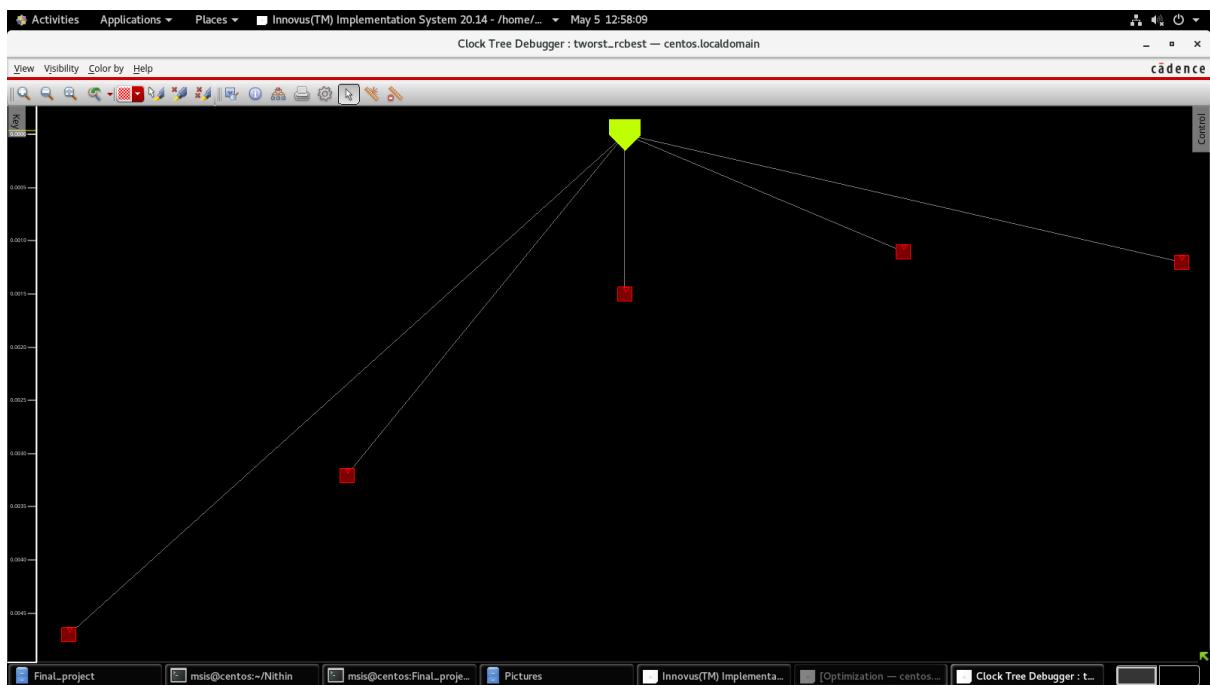


Fig 5.21: Clock Tree Synthesis generated

CTS generated successfully.

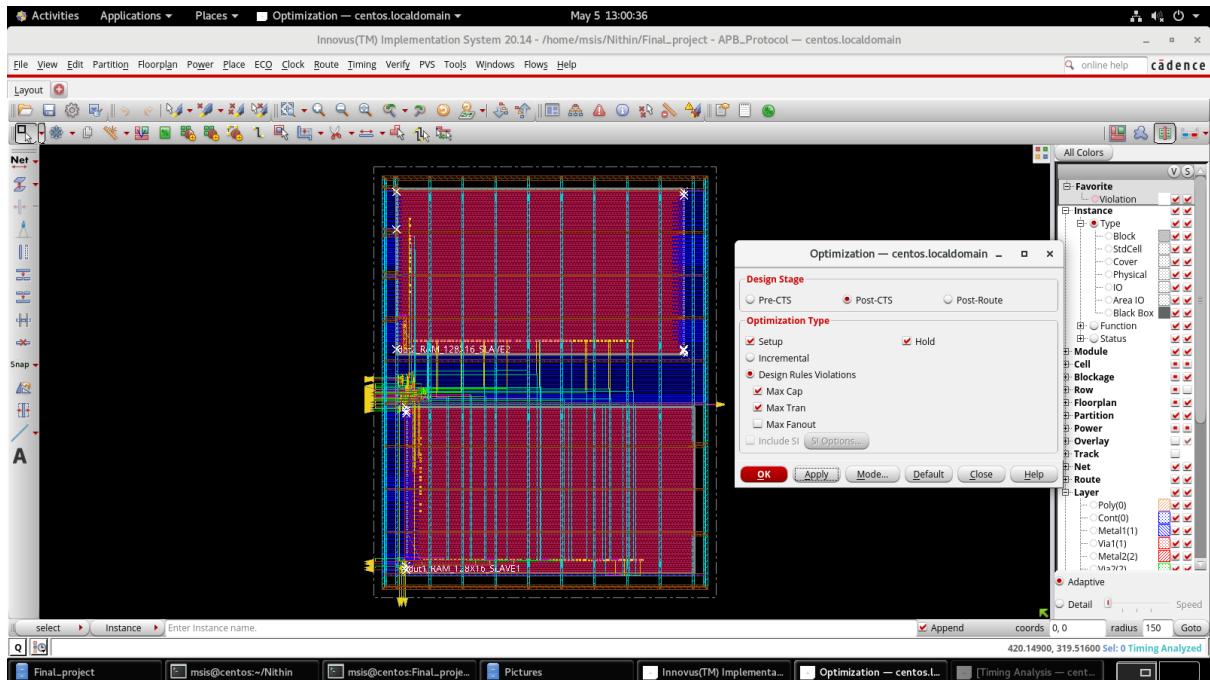


Fig 5.22: Post CTS design optimization

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

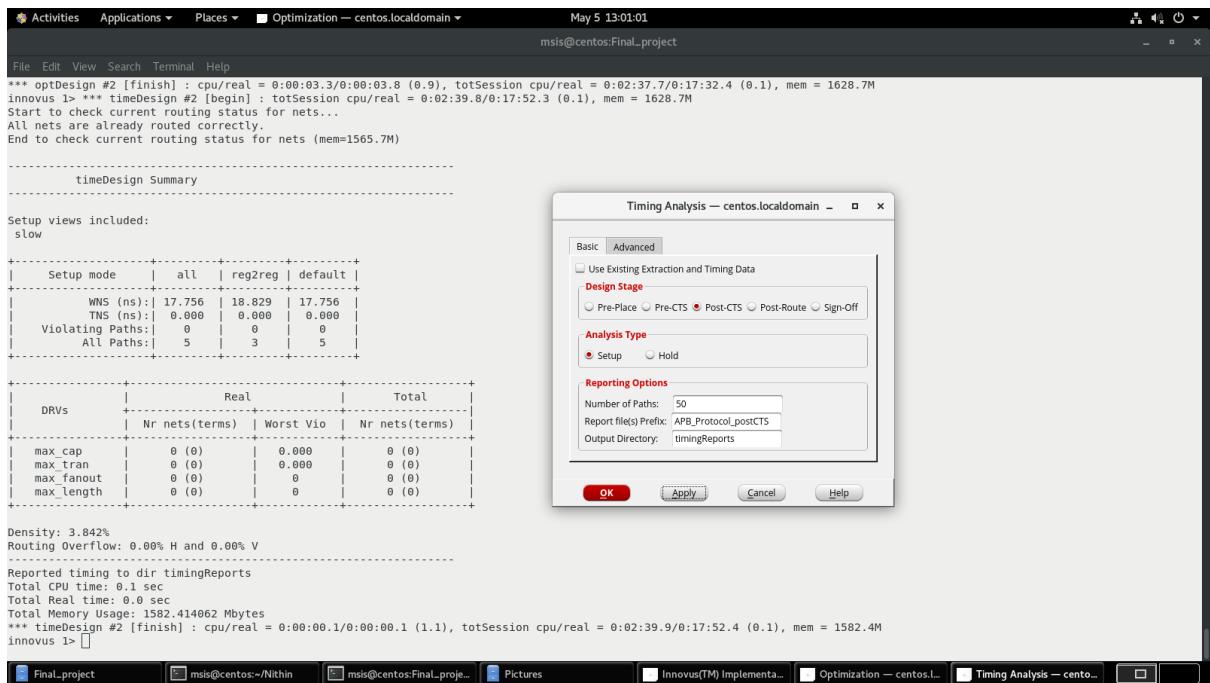


Fig 5.23: Post-CTS setup timing report

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

```

#####
# Design Stage: PreRoute
# Design Name: APB Protocol
# Design Mode: 90nm
# Analysis Mode: MMMC Non-OCV
# Parasitics Mode: No SPICE/RCDB
# Signoff Settings: SI Off
#####
Calculate delays in Bcwc mode...
Start delay calculation (fullDC) (1 T). (MEM=1568.92)
*** Calculating scaling factor for tbtest libraries using the default operating condition of each library.
Total number of fetched objects 350
AAE INFO: Total number of nets for which stage creation was skipped for all views 0
End delay calculation. (MEM=1596.13 CPU=0:00:00.0 REAL=0:00:00.0)
End delay calculation (fullDC). (MEM=1596.13 CPU=0:00:00.0 REAL=0:00:00.0)
*** Done Building Timing Graph (cpu=0:00:00.1 real=0:00:00.0 totSession
-----
timeDesign Summary
-----
Hold views included:
fast

+-----+ +-----+ +-----+
| Hold mode | all | reg2reg | default |
+-----+ +-----+ +-----+
| WNS (ns): | 0.019 | 0.156 | 0.019 |
| TNS (ns): | 0.000 | 0.000 | 0.000 |
| Violating Paths: | 0 | 0 | 0 |
| All Paths: | 5 | 3 | 5 |
+-----+ +-----+ +-----+

Density: 3.842%
Routing Overflow: 0.00% H and 0.00% V
-----
Reported timing to dir timingReports
Total CPU time: 0.18 sec
Total Real time: 0.0 sec
Total Memory Usage: 1530.617188 Mbytes
*** timeDesign #3 [finish] : cpu/real = 0:00:00.2/0:00:00.2 (1.0), totSession cpu/real = 0:02:41.3/0:18:04.4 (0.1), mem = 1530.6M
innovus 1> 
```

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

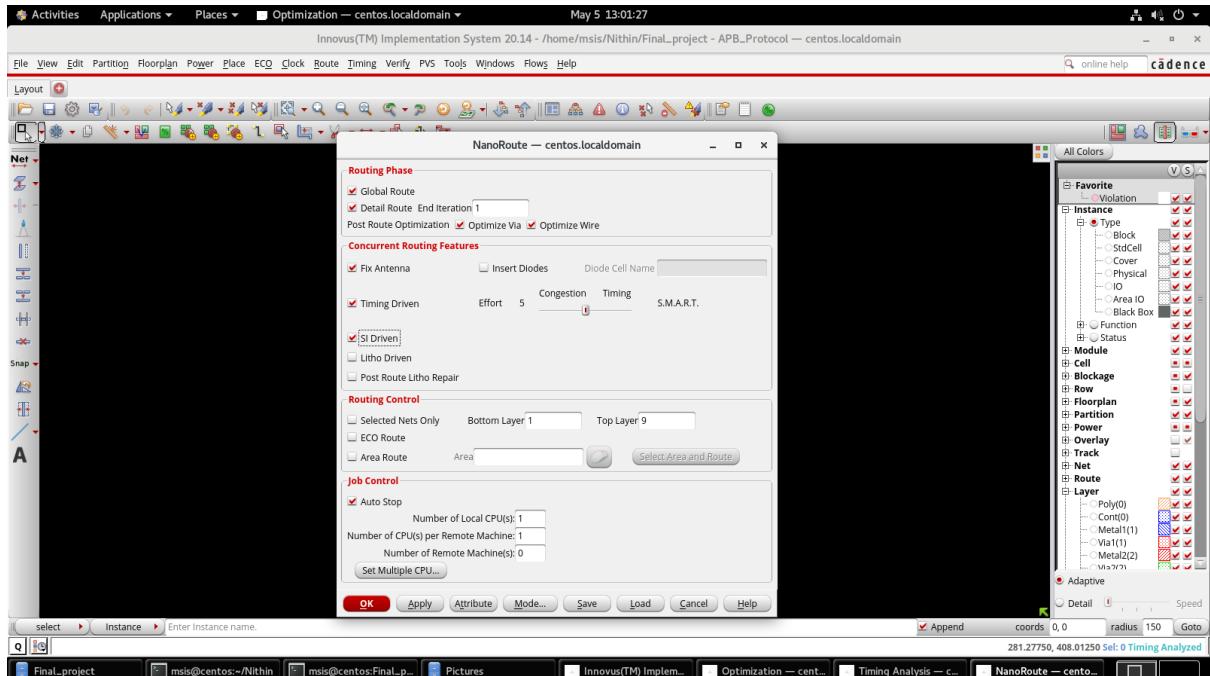


Fig 5.25: Routing specifications

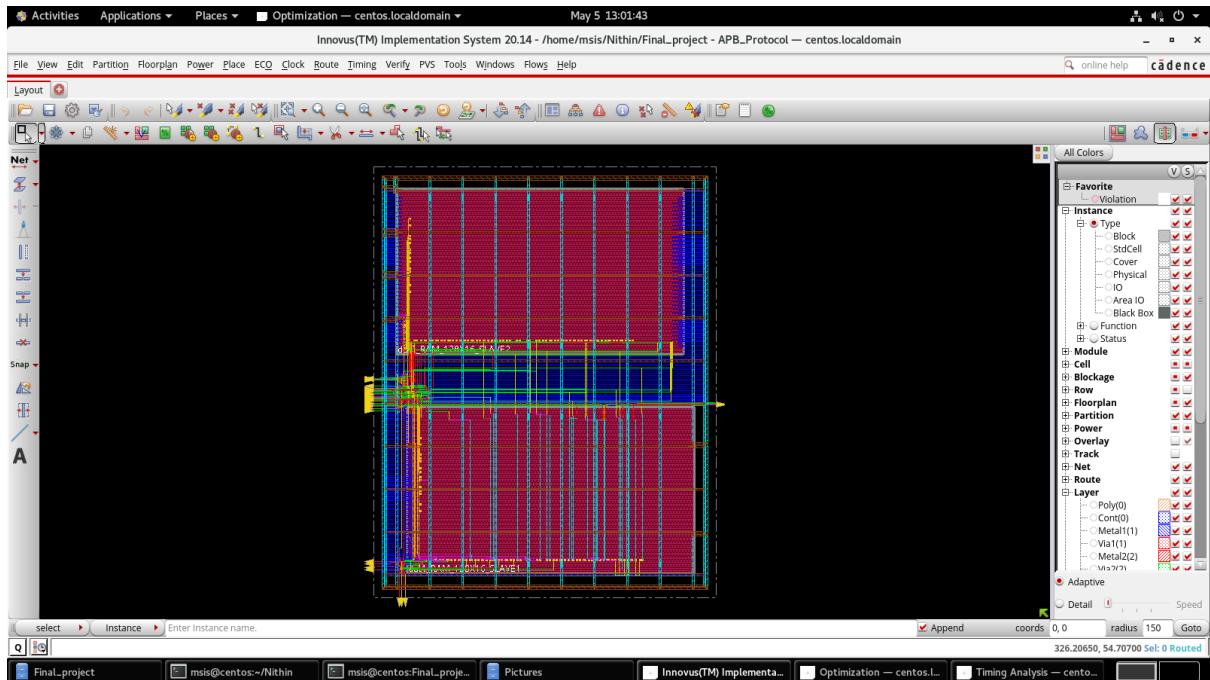


Fig 5.26: Post Routing design

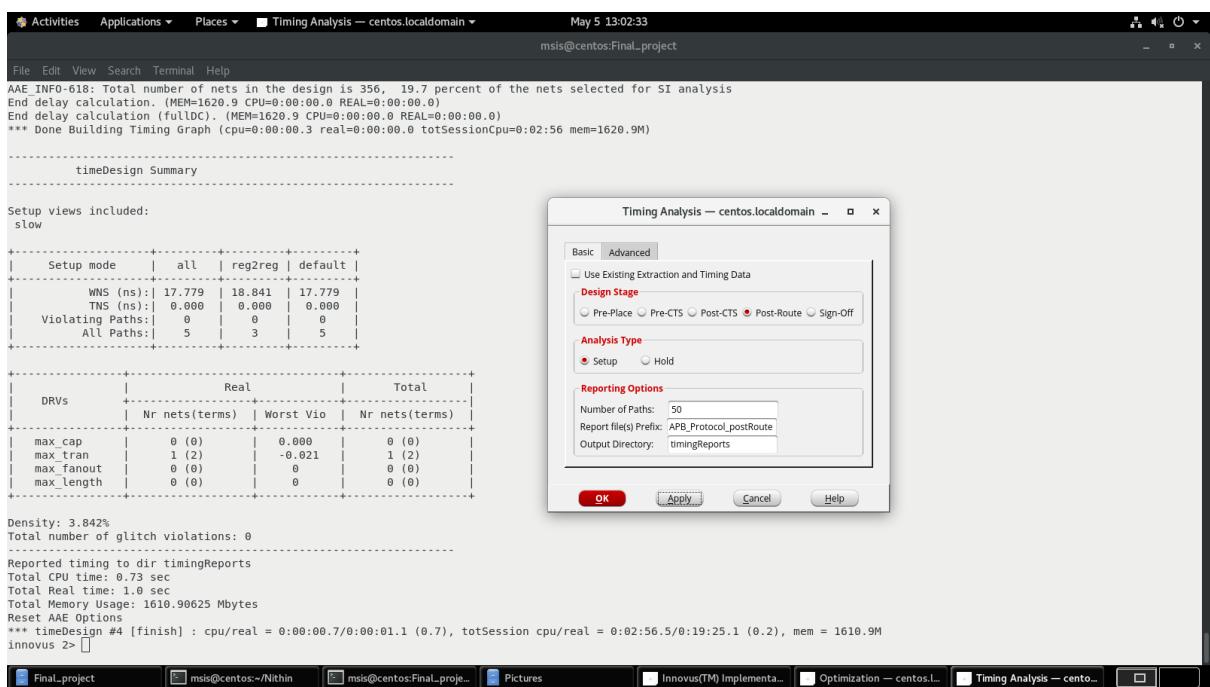


Fig 5.27: Post routing setup timing report

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

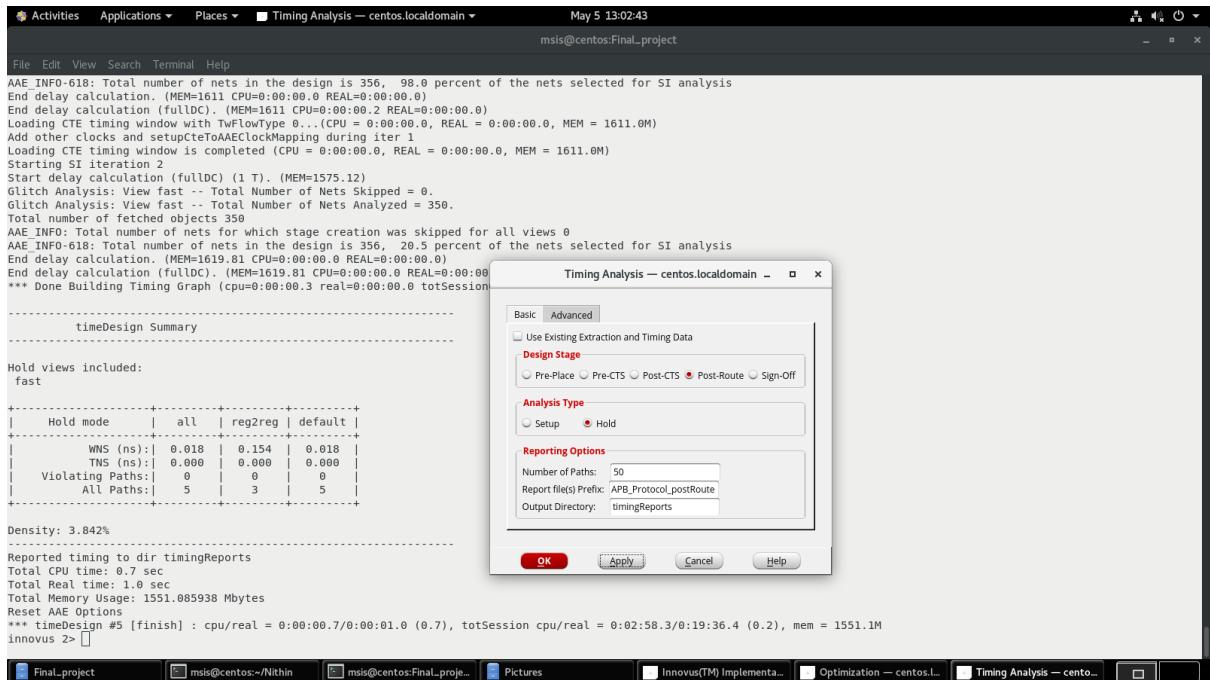


Fig 5.28: Post routing hold timing report

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

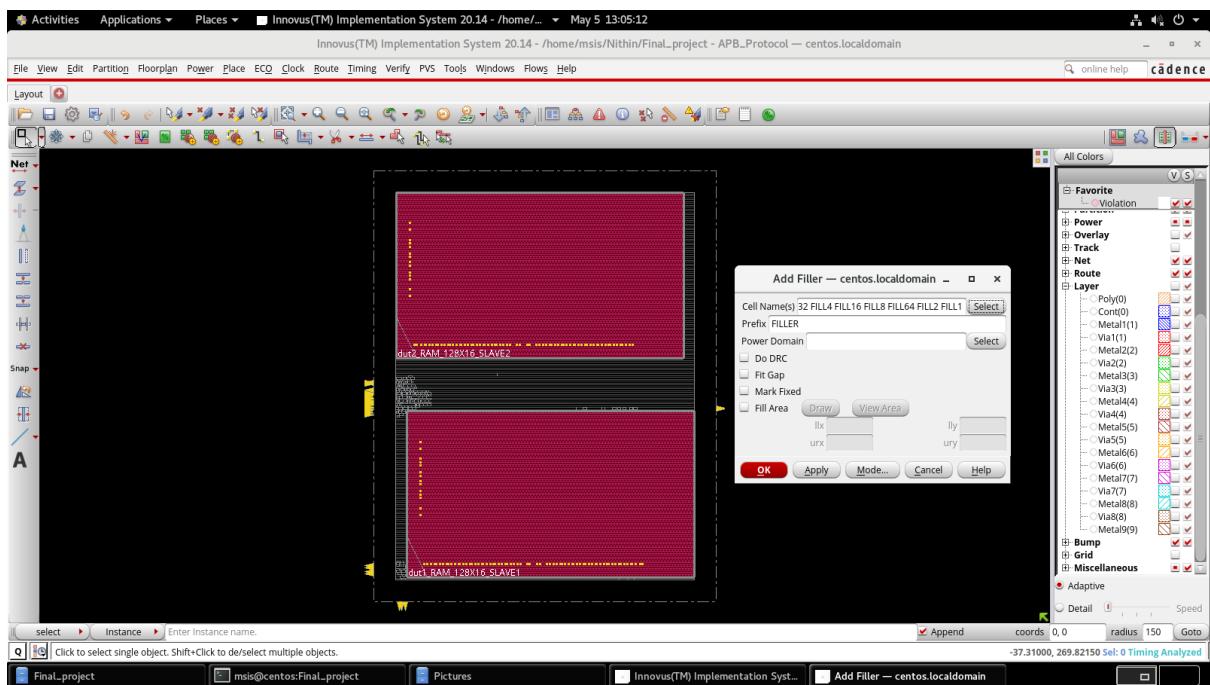


Fig 5.29: Adding fillers

Adding fillers to the design.

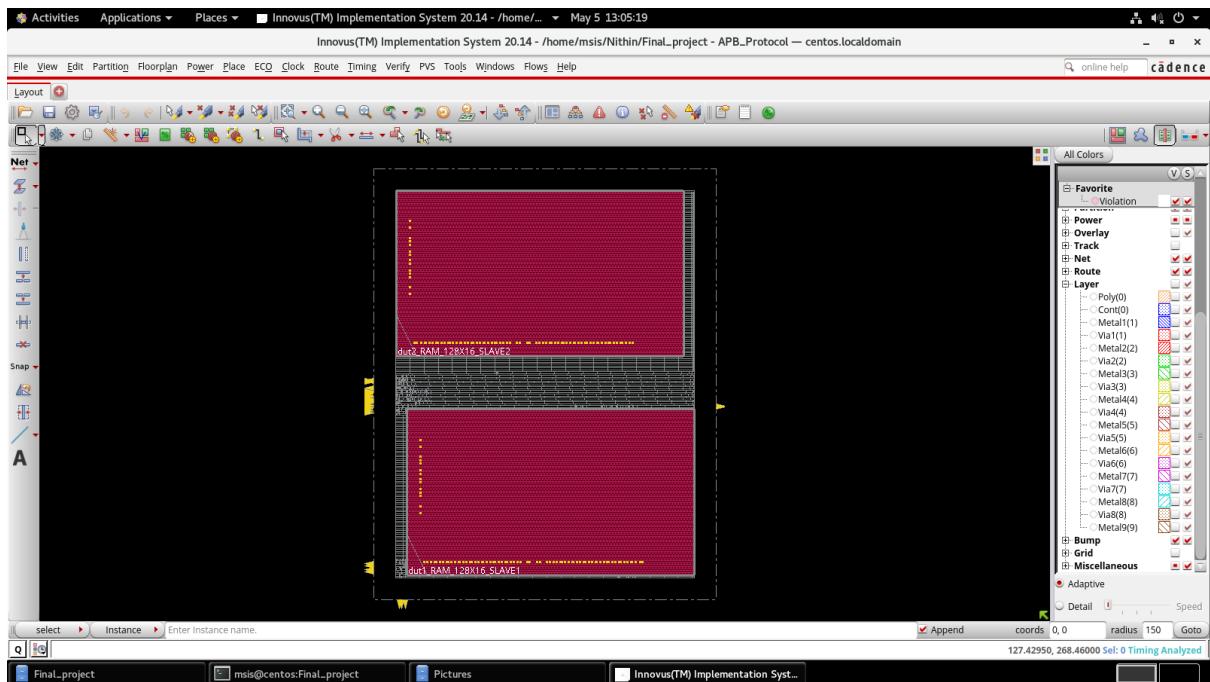


Fig 5.30: After adding fillers

Design has been added with fillers.

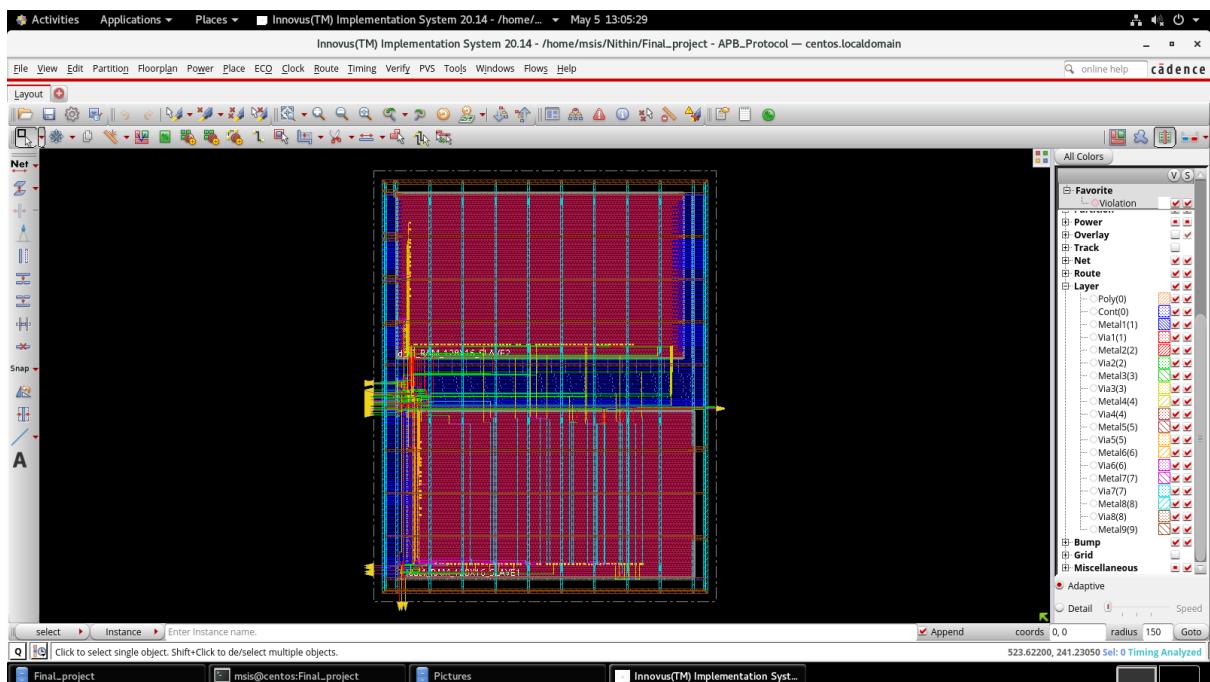


Fig 5.31: Final Physical Design

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

## 5.1.6 DRC check

```
File Edit View Search Terminal Help
VERIFY DRC ..... Sub-Area: {155.520 0.000 233.280 78.720} 3 of 20
VERIFY DRC ..... Sub-Area: 3 complete 0 Viols.
VERIFY DRC ..... Sub-Area: {233.280 0.000 310.000 78.720} 4 of 20
VERIFY DRC ..... Sub-Area: 4 complete 0 Viols.
VERIFY DRC ..... Sub-Area: {0.000 78.720 77.760 157.440} 5 of 20
VERIFY DRC ..... Sub-Area: 5 complete 0 Viols.
VERIFY DRC ..... Sub-Area: {77.760 78.720 155.520 157.440} 6 of 20
VERIFY DRC ..... Sub-Area: 6 complete 0 Viols.
VERIFY DRC ..... Sub-Area: {155.520 78.720 233.280 157.440} 7 of 20
VERIFY DRC ..... Sub-Area: 7 complete 0 Viols.
VERIFY DRC ..... Sub-Area: {233.280 78.720 310.000 157.440} 8 of 20
VERIFY DRC ..... Sub-Area: 8 complete 0 Viols.
VERIFY DRC ..... Sub-Area: {0.000 157.440 77.760 236.160} 9 of 20
VERIFY DRC ..... Sub-Area: 9 complete 0 Viols.
VERIFY DRC ..... Sub-Area: {77.760 157.440 155.520 236.160} 10 of 20
VERIFY DRC ..... Sub-Area: 10 complete 0 Viols.
VERIFY DRC ..... Sub-Area: {155.520 157.440 233.280 236.160} 11 of 20
VERIFY DRC ..... Sub-Area: 11 complete 0 Viols.
VERIFY DRC ..... Sub-Area: {233.280 157.440 310.000 236.160} 12 of 20
VERIFY DRC ..... Sub-Area: 12 complete 0 Viols.
VERIFY DRC ..... Sub-Area: {0.000 236.160 77.760 314.880} 13 of 20
VERIFY DRC ..... Sub-Area: 13 complete 0 Viols.
VERIFY DRC ..... Sub-Area: {77.760 236.160 155.520 314.880} 14 of 20
VERIFY DRC ..... Sub-Area: 14 complete 0 Viols.
VERIFY DRC ..... Sub-Area: {155.520 236.160 233.280 314.880} 15 of 20
VERIFY DRC ..... Sub-Area: 15 complete 0 Viols.
VERIFY DRC ..... Sub-Area: {233.280 236.160 310.000 314.880} 16 of 20
VERIFY DRC ..... Sub-Area: 16 complete 0 Viols.
VERIFY DRC ..... Sub-Area: {0.000 314.880 77.760 389.880} 17 of 20
VERIFY DRC ..... Sub-Area: 17 complete 0 Viols.
VERIFY DRC ..... Sub-Area: {77.760 314.880 155.520 389.880} 18 of 20
VERIFY DRC ..... Sub-Area: 18 complete 0 Viols.
VERIFY DRC ..... Sub-Area: {155.520 314.880 233.280 389.880} 19 of 20
VERIFY DRC ..... Sub-Area: 19 complete 0 Viols.
VERIFY DRC ..... Sub-Area: {233.280 314.880 310.000 389.880} 20 of 20
VERIFY DRC ..... Sub-Area: 20 complete 0 Viols.

Verification Complete : 0 Viols.

*** End Verify DRC (CPU: 0:00:00.4 ELAPSED TIME: 0.00 MEM: 0.0M) ***

innovus 2>
```

Fig 5.32: DRC check result

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

## 5.1.7 Reports generated & constraints used

```
Generated by: Genus(TM) Synthesis Solution 20.11-s111_1
Generated on: May 05 2023 12:34:20 pm
Module: APB_Protocol
Technology libraries: gpdk045wc
Operating conditions: fast (balanced_tree)
Wireload mode: enclosed
Area mode: timing library

Instance Module Cell Count Cell Area Net Area Total Area Wireload
----- APB_Protocol 191 100000.000 0.000 100000.000 <none> (D)
(D) = wireload is default in technology library
```

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.

timing

File Edit View

```

=====
Generated by:          Genus(TM) Synthesis Solution 20.11-s111_1
Generated on:          May 05 2023 12:34:20 pm
Module:                APB_Protocol
Technology libraries: gpdk045wc
                        MEM2_128X16 1.1
Operating conditions: fast (balanced_tree)
Wireload mode:         enclosed
Area mode:             timing library
=====

      Pin           Type     Fanout Load Slew Delay Arrival
                           (fF)   (ps)  (ps)   (ps)
=====
(clock PCLK)           launch
dut_mas_state_reg[1]/CK          0      +0      0 R
dut_mas_state_reg[1]/Q           DFFTRX4    3  11.2   28    +79    79 R
g3717/A                         INVX8      2   8.7   15    +12    91 F
g3717/Y                         INVX8      2   8.7   15    +12    91 F
g3670_8428/B                    NAND3X6     5   8.1   24    +14   105 R
g3670_8428/Y                    NOR2BX4     2  10.6   46    +56   161 R
g2_5122/AN                      NOR2BX4     2  10.6   46    +56   161 R
g2_5122/Y                       NOR2BX4     2  10.6   46    +56   161 R
dut1_RAM_128X16_SLAVE1/CE1 <<< (P) MEM2_128X16          0      +0      0 R
dut1_RAM_128X16_SLAVE1/CK1       setup
                                  capture
                                                0      +503    664 R
                                                20000 R
=====
Cost Group : 'PCLK' (path_group 'PCLK')
Timing slack : 19336ps
Start-point : dut_mas_state_reg[1]/CK
End-point   : dut1_RAM_128X16_SLAVE1/CE1
(P) : Instance is preserved

```

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_uncertainty 1 [get_clocks PCLK]
```

Fig 5.36: Constraints defined

```
qor
File Edit View

=====
Generated by: Genus(TM) Synthesis Solution 20.11-s111_1
Generated on: May 05 2023 12:34:20 pm
Module: APB_Protocol
Technology libraries: gpdk045wc
                      MEM2_128X16 1.1
Operating conditions: fast (balanced_tree)
Wireload mode: enclosed
Area mode: timing library
=====

Timing
-----
Clock Period
-----
PCLK 20000.0

Cost Critical Violating
Group Path Slack TNS Paths
-----
default No paths 0.0
PCLK 19336.2 0.0 0
-----
Total 0.0 0

Instance Count
-----
Leaf Instance Count 191
Physical Instance count 0
Sequential Instance Count 62
Combinational Instance Count 129
Hierarchical Instance Count 0

Area
-----
Cell Area 100000.000
Physical Cell Area 0.000
Total Cell Area (Cell+Physical) 100000.000
Net Area 0.000
Total Area (Cell+Physical+Net) 100000.000

Max Fanout 52 (n_138)
Min Fanout 0 (dut2_PRDATA1)
Average Fanout 2.1
Terms to net ratio 3.2129
Terms to instance ratio 4.4241
Runtime 11.578836 seconds
Elapsed Runtime 12 seconds
Genus peak memory usage 1289.91
Innovus peak memory usage no_value
Hostname centos.localdomain
```

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.

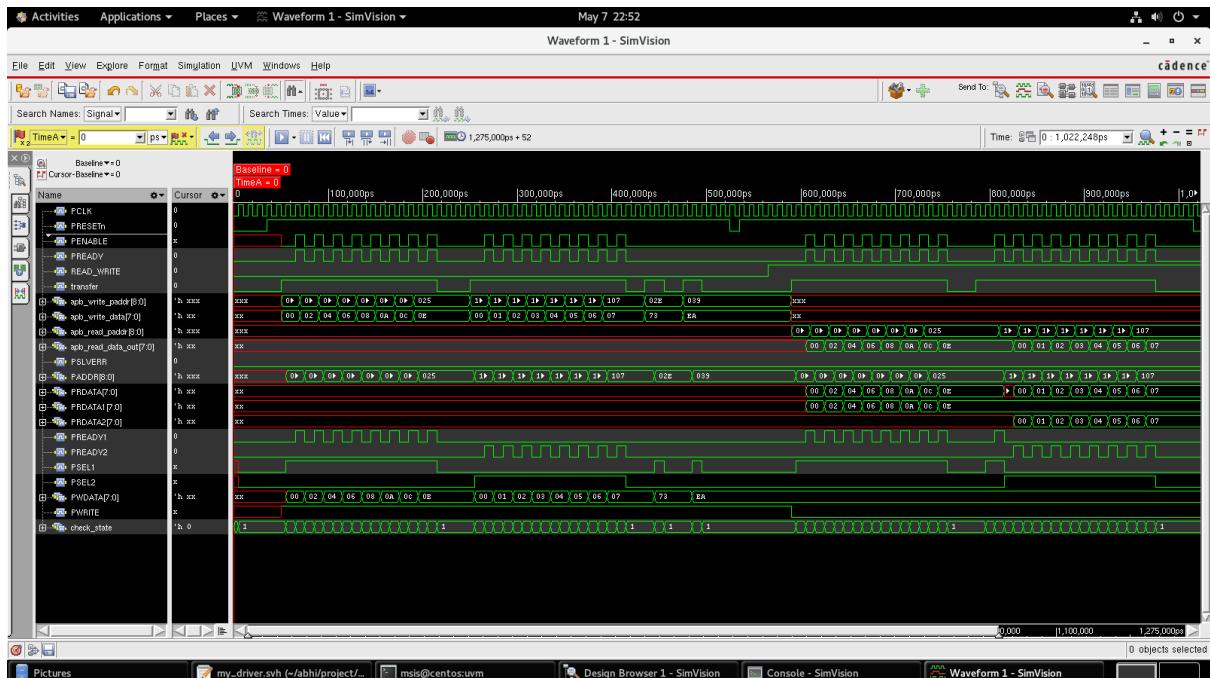


Fig 5.38: Simulation Results

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.

```

Activities Applications Places Terminal May 7 19:42
msis@centos:uvm
File Edit View Search Terminal Help
SLAVE 1 WRITE OPERATION 8X

UVM_INFO my_monitor.svh(38) @ 65000: uvm_test_top.env.agent.monitor [MONITOR] Written Item received
Name Type Size Value
seq seq item - 03755
    apb_write_paddr integral 9 'h1e
    apb_write_data integral 8 'h0
    apb_read_paddr integral 9 'h0xx
    PSLVERR integral 1 'h0
    apb_read_data_out integral 8 'h0x

UVM_INFO my_monitor.svh(38) @ 85000: uvm_test_top.env.agent.monitor [MONITOR] Written Item received
Name Type Size Value
seq seq item - 03755
    apb_write_paddr integral 9 'hf
    apb_write_data integral 8 'h2
    apb_read_paddr integral 9 'h0xx
    PSLVERR integral 1 'h0
    apb_read_data_out integral 8 'h0x

UVM_INFO my_monitor.svh(38) @ 105000: uvm_test_top.env.agent.monitor [MONITOR] Written Item received
Name Type Size Value
seq seq item - 03755
    apb_write_paddr integral 9 'h20
    apb_write_data integral 8 'h1
    apb_read_paddr integral 9 'h0xx
    PSLVERR integral 1 'h0
    apb_read_data_out integral 8 'h0x

UVM_INFO my_monitor.svh(38) @ 125000: uvm_test_top.env.agent.monitor [MONITOR] Written Item received
Name Type Size Value
seq seq item - 03755
    apb_write_paddr integral 9 'h21
    apb_write_data integral 8 'h6
    apb_read_paddr integral 9 'h0xx

UVM_INFO my_monitor.svh(38) @ 145000: uvm_test_top.env.agent.monitor [MONITOR] Written Item received
Name Type Size Value
seq seq item - 03755
    apb_write_paddr integral 9 'h22
    apb_write_data integral 8 'h1
    apb_read_paddr integral 9 'h0xx
    PSLVERR integral 1 'h0
    apb_read_data_out integral 8 'h0x

```

Fig 5.39: Write to Slave 1

```

Activities Applications Places Terminal May 7 19:43
msis@centos:uvm
File Edit View Search Terminal Help
SLAVE 2 WRITE OPERATION 8X

UVM_INFO my_monitor.svh(38) @ 265000: uvm_test_top.env.agent.monitor [MONITOR] Written Item received
Name Type Size Value
seq seq item - 03755
    apb_write_paddr integral 9 'h100
    apb_write_data integral 8 'h0
    apb_read_paddr integral 9 'h0xx
    PSLVERR integral 1 'h0
    apb_read_data_out integral 8 'h0x

UVM_INFO my_monitor.svh(38) @ 285000: uvm_test_top.env.agent.monitor [MONITOR] Written Item received
Name Type Size Value
seq seq item - 03755
    apb_write_paddr integral 9 'h101
    apb_write_data integral 8 'h1
    apb_read_paddr integral 9 'h0xx
    PSLVERR integral 1 'h0
    apb_read_data_out integral 8 'h0x

UVM_INFO my_monitor.svh(38) @ 305000: uvm_test_top.env.agent.monitor [MONITOR] Written Item received
Name Type Size Value
seq seq item - 03755
    apb_write_paddr integral 9 'h102
    apb_write_data integral 8 'h2
    apb_read_paddr integral 9 'h0xx
    PSLVERR integral 1 'h0
    apb_read_data_out integral 8 'h0x

UVM_INFO my_monitor.svh(38) @ 325000: uvm_test_top.env.agent.monitor [MONITOR] Written Item received
Name Type Size Value
seq seq item - 03755
    apb_write_paddr integral 9 'h103
    apb_write_data integral 8 'h3
    apb_read_paddr integral 9 'h0xx

```

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 conditions and displays the sequence packet as shown in Fig 5.39 and 5.40.

```

Activities Applications Places Terminal May 7 19:48
msis@centos:uvm

File Edit View Search Terminal Help
UVM_INFO my_driver.svh(61) @ 435000: uvm_test_top.env.agent.driver [DRIVER] DRIVING RANDOMIZED VALUES
Name Type Size Value
req seq_item @3767
  apb_write_paddr integral 9 'h2e
  apb_write_data integral 8 'h73
  apb_read_paddr integral 9 'h0xx
  PSLVERR integral 1 'h0
  apb_read_data_out integral 8 'h0xx
  begin_time time 64 435000
  depth int 32 'd2
  parent sequence (name) string 3 seq
  parent sequence (full name) string 36 uvm_test_top.env.agent.sequencer.seq
  sequencer string 32 uvm_test_top.env.agent.sequencer
UVM_INFO my_monitor.svh(38) @ 445000: uvm_test_top.env.agent.monitor [MONITOR] Written Item received
Name Type Size Value
seq seq_item - 03755
  apb_write_paddr integral 9 'h2e
  apb_write_data integral 8 'h73
  apb_read_paddr integral 9 'h0xx
  PSLVERR integral 1 'h0
  apb_read_data_out integral 8 'h0xx
UVM_INFO my_driver.svh(61) @ 475000: uvm_test_top.env.agent.driver [DRIVER] DRIVING RANDOMIZED VALUES
Name Type Size Value
req seq_item @3790
  apb_write_paddr integral 9 'h39
  apb_write_data integral 8 'heaa
  apb_read_paddr integral 9 'h0xx
  PSLVERR integral 1 'h0
  apb_read_data_out integral 8 'h0xx
  begin_time time 64 475000
  depth int 32 'd2
  parent sequence (name) string 3 seq
  parent sequence (full name) string 36 uvm_test_top.env.agent.sequencer.seq
  sequencer string 32 uvm_test_top.env.agent.sequencer

```

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.

```

Activities Applications Places Terminal May 7 20:03
msis@centos:uvm

File Edit View Search Terminal Help
SLAVE 1 READ OPERATION 8X

UVM_INFO my_monitor.svh(38) @ 605000: uvm_test_top.env.agent.monitor [MONITOR] Written Item received
Name Type Size Value
seq seq_item - 03755
  apb_write_paddr integral 9 'h0xx
  apb_write_data integral 8 'h0x
  apb_read_paddr integral 9 'h1e
  PSLVERR integral 1 'h0
  apb_read_data_out integral 8 'h0x
UVM_INFO my_monitor.svh(55) @ 625000: uvm_test_top.env.agent.monitor [MONITOR] Read Item received
Name Type Size Value
seq seq_item - 03755
  apb_write_paddr integral 9 'h0xx
  apb_write_data integral 8 'h0x
  apb_read_paddr integral 9 'h1f
  PSLVERR integral 1 'h0
  apb_read_data_out integral 8 'h0
UVM_INFO my_monitor.svh(55) @ 645000: uvm_test_top.env.agent.monitor [MONITOR] Read Item received
Name Type Size Value
seq seq_item - 03755
  apb_write_paddr integral 9 'h0xx
  apb_write_data integral 8 'h0x
  apb_read_paddr integral 9 'h20
  PSLVERR integral 1 'h0
  apb_read_data_out integral 8 'h2
UVM_INFO my_monitor.svh(55) @ 665000: uvm_test_top.env.agent.monitor [MONITOR] Read Item received
Name Type Size Value
seq seq_item - 03755
  apb_write_paddr integral 9 'h0xx
  apb_write_data integral 8 'h0x

```

Fig 5.42: Read from Slave 1

```

Activities Applications Places Terminal May 7 20:08
msis@centos:uvm
File Edit View Search Terminal Help
-----
SLAVE 2 READ OPERATION BX
UVM_INFO my_monitor.svh(55) @ 815000: uvm_test_top.env.agent.monitor [MONITOR] Read Item received
Name          Type   Size  Value
-----
seq
  apb_write_paddr integral 9    'h0xx
  apb_write_data  integral 8    'h0x
  apb_read_paddr integral 9    'h100
  PSLEVRR        integral 1    'h0
  apb_read_data_out integral 8   'he
-----
UVM_INFO my_monitor.svh(55) @ 835000: uvm_test_top.env.agent.monitor [MONITOR] Read Item received
Name          Type   Size  Value
-----
seq
  apb_write_paddr integral 9    'h0xx
  apb_write_data  integral 8    'h0x
  apb_read_paddr integral 9    'h101
  PSLEVRR        integral 1    'h0
  apb_read_data_out integral 8   'h0
-----
UVM_INFO my_monitor.svh(55) @ 855000: uvm_test_top.env.agent.monitor [MONITOR] Read Item received
Name          Type   Size  Value
-----
seq
  apb_write_paddr integral 9    'h0xx
  apb_write_data  integral 8    'h0x
  apb_read_paddr integral 9    'h102
  PSLEVRR        integral 1    'h0
  apb_read_data_out integral 8   'h1
-----
UVM_INFO my_monitor.svh(55) @ 875000: uvm_test_top.env.agent.monitor [MONITOR] Read Item received
Name          Type   Size  Value
-----
seq
  apb_write_paddr integral 9    'h0xx
  apb_write_data  integral 8    'h0x

```

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.

```

Activities Applications Places Terminal May 7 22:46
msis@centos:uvm
File Edit View Search Terminal Help
OSLERROR: (null)/top/sv/_sv_export.so: cannot open shared object file: No such file or directory or file is not valid ELFCLASS64 library..
ncsim: *W,DSEM2009: This SystemVerilog design is simulated as per IEEE 1800-2009 SystemVerilog simulation semantics. Use -disable_sem2009 option for turning off SV 2009 simulation semantics.
ncsim> source /home/install/INCISIVE152/tools/inca/files/ncsimrc
ncsim> source /home/install/INCISIVE152/tools/methodology/UVM/CDNS-1.1d/additions/sv/files/tcl/uvm_sim.tcl
ncsim> run
-----
CDNS-UVM-1.1d (15.20-s086)
(c) 2007-2013 Mentor Graphics Corporation
(c) 2007-2013 Cadence Design Systems, Inc.
(c) 2006-2013 Synopsys, Inc.
(c) 2011-2013 Cypress Semiconductor Corp.
-----
***** IMPORTANT RELEASE NOTES *****
You are using a version of the UVM library that has been compiled with `UVM_NO_DEPRECATED` undefined.
See http://www.eda.org/svdb/view.php?id=3313 for more details.

You are using a version of the UVM library that has been compiled with `UVM_OBJECT_MUST_HAVE_CONSTRUCTOR` undefined.
See http://www.eda.org/svdb/view.php?id=3770 for more details.

(Specify +UVM_NO_RELNOTES to turn off this notice)

UVM_INFO @ 0: reporter [PNTST] Running test my_test...
UVM_INFO my_monitor.svh(38) @ 85000: uvm_test_top.env.agent.monitor [MONITOR] Written Item received
Name          Type   Size  Value
-----
seq
  apb_write_paddr integral 9    'h22
  apb_write_data  integral 8    'h0x
  apb_read_paddr integral 9    'h0XX
  PSLEVRR        integral 1    'h1
  apb_read_data_out integral 8   'h0x
-----
Simulation complete via $finish(1) at time 105 NS + 0
./my_driver.svh:88           $finish;
ncsim> exit
[msis@centos uvm]$ 

```

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.

```

OSDLError: (null)/top/sv/_sv export.so: cannot open shared object file: No such file or directory or file is not valid ELFCLASS64 library..
ncsim: *W_DSM2009: This SystemVerilog design is simulated as per IEEE 1800-2009 SystemVerilog simulation semantics. Use -disable_sem2009 option for turning off SV 2009 simulation semantics.
ncsim source /home/installs/INCISIVE152/tools/inca/files/ncsimrc
ncsim source /home/installs/INCISIVE152/tools/methodology/UVM/CDNS-1.1d/additions/sv/files/tcl/uvm_sim.tcl
ncsim> run

CDNS-UVM-1.1d (15.20-s086)
(C) 2007-2013 Mentor Graphics Corporation
(C) 2007-2013 Cadence Design Systems, Inc.
(C) 2006-2013 Synopsys, Inc.
(C) 2011-2013 Cypress Semiconductor Corp.

*****
IMPORTANT RELEASE NOTES *****
You are using a version of the UVM library that has been compiled
with `UVM_NO_DEPRECATED` undefined.
See http://www.eda.org/svdb/view.php?id=3313 for more details.

You are using a version of the UVM library that has been compiled
with `UVM_OBJECT_MUST_HAVE_CONSTRUCTOR` undefined.
See http://www.eda.org/svdb/view.php?id=3770 for more details.

(Specify +UVM_NO_RELNOTES to turn off this notice)

UVM_INFO @ 0: reporter [RNTST] Running test my_test...
UVM_INFO my_monitor.svh(38) @ 35000: uvm_test_top.env.agent.monitor [MONITOR] Written Item received
-----
Name      Type   Size Value
-----
seq      seq item    03755
apb_write_paddr integral 9   'h0XXX
apb_write_data  integral 8   'h0XX
apb_read_paddr integral 9   'h0XXX
PSLVERR  integral 1   'h1
apb_read_data_out integral 8   'h0X
-----
Simulation complete via $finish() at time 45 NS + 0
./my_driver.svh:58
$finish;
ncsim> exit
[msis@centos uvm]$

```

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.

```

UVM_INFO my_monitor.svh(55) @ 95000: uvm_test_top.env.agent.monitor [MONITOR] Read Item received
-----
Name      Type   Size Value
-----
seq      seq item    03755
apb_write_paddr integral 9   'h0XXX
apb_write_data  integral 8   'h0XX
apb_read_paddr integral 9   'h0XXX
PSLVERR  integral 1   'h1
apb_read_data_out integral 8   'h0X
-----
Simulation complete via $finish() at time 105 NS + 0
./my_driver.svh:74
$finish;
ncsim> exit
[msis@centos uvm]$

```

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.

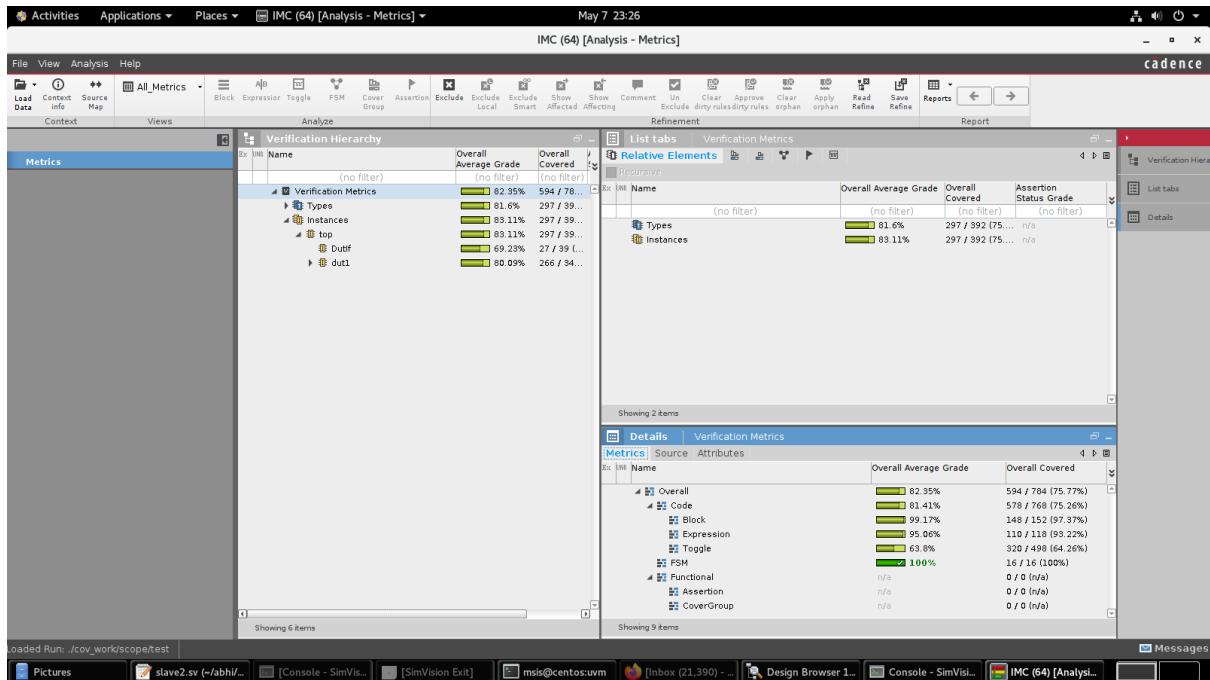


Fig 5.47: Verification Metrics

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.

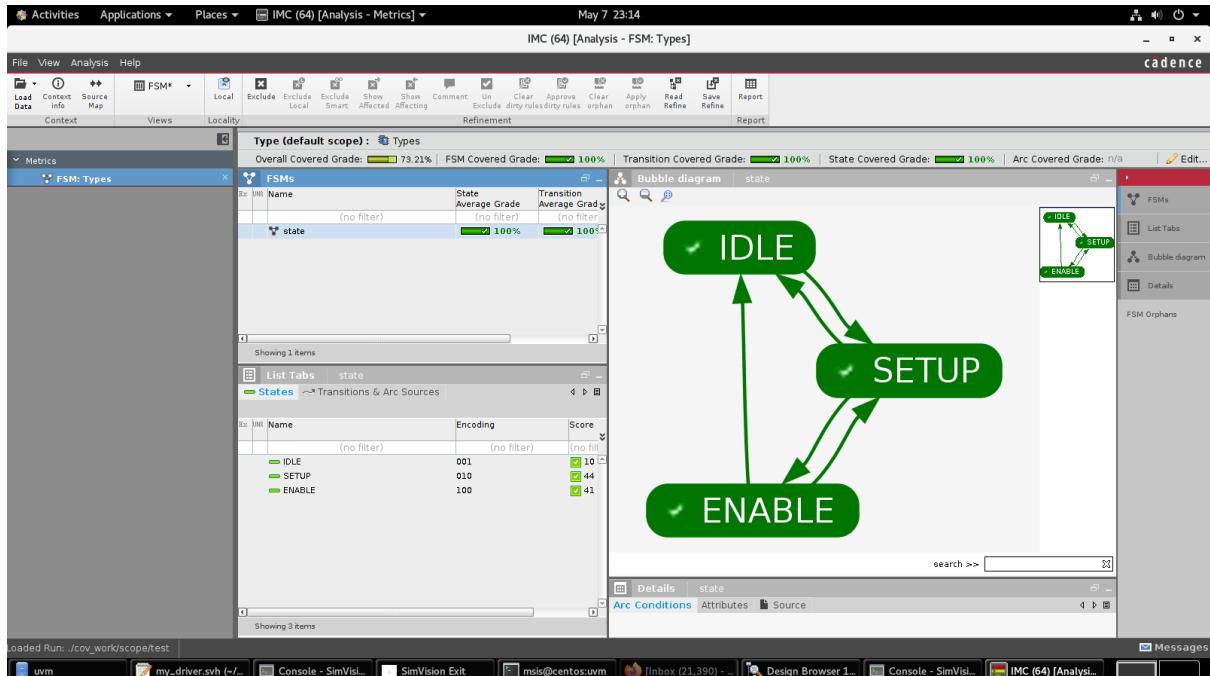


Fig 5.48: FSM Coverage

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