KISTA 1um CMOS SOI Cadence Digital Flow

Version 0.0.2

S. Rodriguez

February 7, 2024

Contents

List of Code Listings

1 Introduction

This manual shows how to use the Cadence Innovus digital flow and the KISTA 1um CMOS SOI PDK . This version has been tested with the following tools:

```
GENUS211 (digital synthesis)
INNOVUS181 (digital place & route)
XCELIUM1803  (digital simulation)
IC618 (Virtuoso Schematic and layout editor)
PVS161 (DRC & LVS physical verification)
```

The manual will show you step by step how to convert Verilog RTL code into a routed layout block that can be used either in an Analog-on-Top flow or Digital-on-Top flow. The example consists of a SPI slave mode 0 block that is implemented in Verilog and a testbench written in SystemVerilog. All files are included in the tar file TEST_LIB2.tgz which is available under the PDK directory KISTA/docs/.

1.1 Directory Structure

Keeping a good file structure and organization is important since there are several tools involved. First create a new directory that will contain the whole project:

```
mkdir WORKSHOP
cd WORKSHOP
```

Now create the following directories:

```
mkdir src
mkdir genus
mkdir innovus
mkdir liberty
mkdir virtuoso
mkdir sim_rtl
mkdir sim_gate
mkdir sim_postlay
```

Enter the virtuoso directory and setup it using the **install_workdir.sh** script located under the directory /afs/ict.kth.se/proj/ektlab/PDK/KISTA/docs/install:

```
cd virtuoso
export PDK_HOME=/afs/ict.kth.se/proj/ektlab/PDK/KISTA
cp $PDK_HOME/docs/install/install_workdir.sh .
source install_workdir.sh
```

At this point we can set all environment variables and paths to the Cadence tools:

```
source init_kth.sh
```

Now we need to unpack the TEST_LIB2.tgz example project since we will reuse some files:

```
cd ..
cp $PDK_HOME/docs/install/TEST_LIB2.tgz .
tar -xzvf TEST_LIB2.tgz
```

Now, we are going to create symbolic links to useful files in the different directories. We start with the liberty timing models for the standard cells:

```
cd ..
cd liberty
ln -s ../virtuoso/models/liberty/*.lib .
```

Now we continue with links to the standard cell verilog model files:

```
cd ..
cd src
ln -s ../virtuoso/models/verilog/*.v .
```

We need to copy the Verilog RTL code and testbench in the same directory:

```
cp -s ../TEST_LIB2/src/SPI*.* .
cd ..
```

2 RTL Simulation

2.1 RTL setup

Now go to the **sim_rtl** directory and create a hdl.var file that contains xcelium definitions:

```
cd sim_rtl
gedit hdl.var
```

Copy the following lines, save and exit:

```
define WORK work
#include $CDSHOME/tools/inca/files/hdl.var
```

Now create the following directories:

```
mkdir work
mkdir work_lib
mkdir work_sub
```

Finally create a **cds.lib** file:

```
gedit cds.lib
```

add the following content, save, and exit:

```
include $CDSHOME/tools/inca/files/cds.lib
define work_lib ./work_lib
define work_sub ./work_sub
define work ./work
```

2.2 RTL scripted simulation

Now you will create a **make.sh** script for loading the libraries, elaborating the design, and simulating it (You can also copy paste these example scripts. They are located under TEST_LIB2/sim_rtl). Create the file:

```
gedit make.sh
```

Copy the following content, save and exit:

```
rm -rf work*/* work*/.* *.log .simvision *.dsn *.trn *.vcd wave* *.X

#load system verilog testbench and include a directory in the path
xmvlog -work work -cdslib ./cds.lib -append_log \
    -logfile xmvlog_presyn.log -errormax 15 -update -linedebug \
    -define presyn -status ../src/SPI_Master_TB.sv -incdir ../src -sv

#elaborate the projet
xmelab -work work -cdslib ./cds.lib -timescale '1ns/1ps' \
    -logfile xmelab_presyn.log -errormax 30 -access +wc \
    -nowarn CUVWSP -nowarn SDFNCAP -status work.SPI_Master_TB


xmsim -cdslib ./cds.lib -logfile xmsim_presyn.log -errormax 15 -status \
    work.SPI_Master_TB:module
```

Then source the script to run the simulation. Check for warnings and errors.

```
source make.sh
```

2.3 View results in simvision

In most of the cases the testbench instructs the simulator to write a vcd file which includes all the saved waveforms that need to be inspected. These waveforms can be viewed with Cadence simvision. Load the vcd file:

```
simvision dump.vcd
```

Click Ok on the pop-up menu that appears during startup. Then go to the Design Browser (Left side) and select SPI_Master_TB. Now all the saved waveforms of this module are visible under Show contents. You can click the waveforms that you want to inspect and they will be visible in the time scope. To zoom out and fit all the waveforms in the scope, click on the = button on the right top.

You can also go inside the hierarchy and inspect waveforms from underlying blocks. For instance, click on SPI_Slave_UUT and all its saved waveforms will be available for selection.

As this is RTL level, there are no delays in the sequential and combinational circuits. You can confirm this by moving red the cursors BaseLine and TimeA.

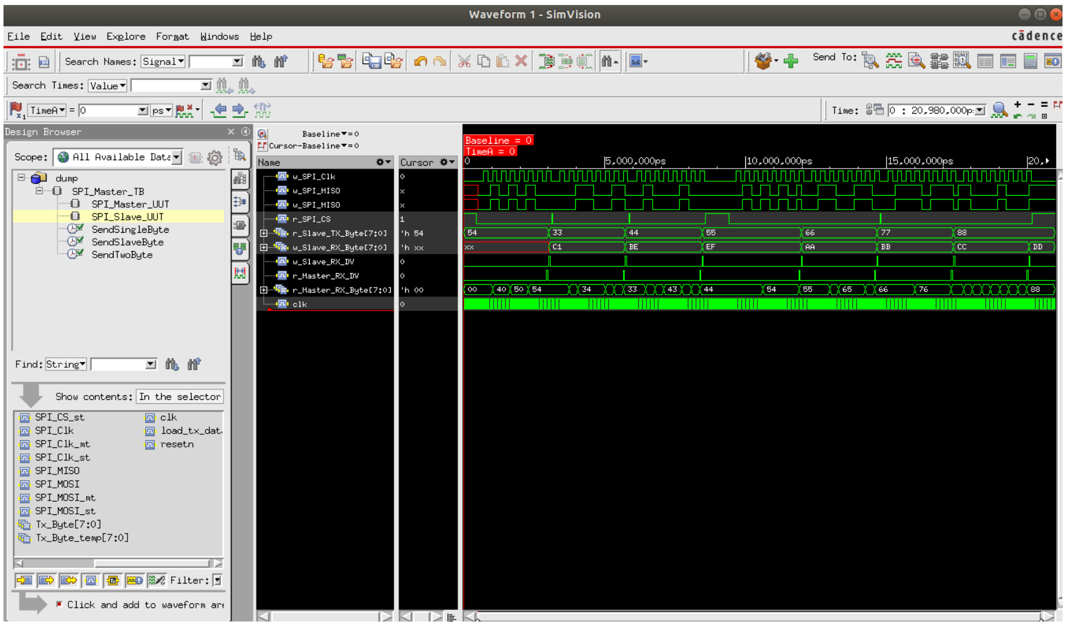
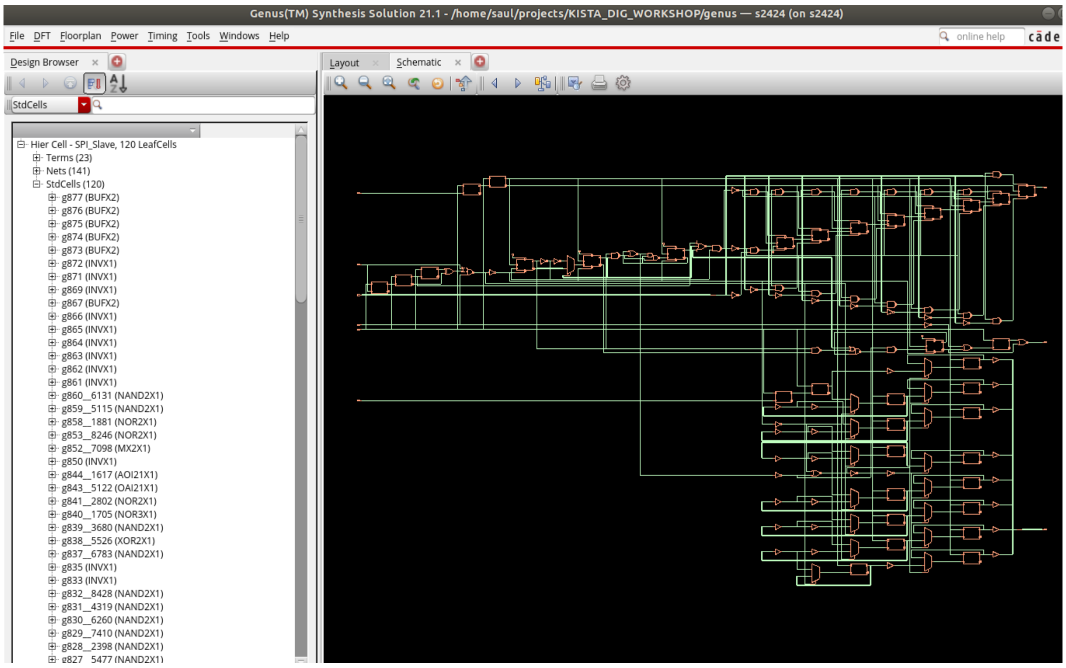


Figure 1: RTL simulation


Genus(TM) Synthesis Solution 21.1 - /home/saul/projects/KISTA_DIG_WORKSHOP/genus — s2424 (on s2424)
File DFT Floorplan Power Timing Tools Windows Help
online help
Design Browser
StdCells
Layout
Schematic

```
Hier Cell - SPI_Slave, 120 LeafCells
  Terms (23)
  Nets (141)
  StdCells (120)
    g877 (BUFX2)
    g876 (BUFX2)
    g875 (BUFX2)
    g874 (BUFX2)
    g873 (BUFX2)
    g872 (INVX1)
    g871 (INVX1)
    g869 (INVX1)
    g867 (BUFX2)
    g866 (INVX1)
    g865 (INVX1)
    g864 (INVX1)
    g863 (INVX1)
    g862 (INVX1)
    g861 (INVX1)
    g860__6131 (NAND2X1)
    g859__5115 (NAND2X1)
    g858__1881 (NOR2X1)
    g853__8246 (NOR2X1)
    g852__7098 (MX2X1)
    g850 (INVX1)
    g844__1617 (AOI21X1)
    g843__5122 (OAI21X1)
    g841__2802 (NOR2X1)
    g840__1705 (NOR3X1)
    g839__3680 (NAND2X1)
    g838__5526 (XOR2X1)
    g837__6783 (NAND2X1)
    g835 (INVX1)
    g833 (INVX1)
    g832__8428 (NAND2X1)
    g831__4319 (NAND2X1)
    g830__6260 (NAND2X1)
    g829__7410 (NAND2X1)
    g828__2398 (NAND2X1)
    g827__5477 (NAND2X1)
```

Figure 2: Synthesized Schematic in Genus

3 Gate Level Synthesis and Simulation

3.1 Synthesis

First, go to the genus directory and copy the example tcl script for synthesis. Inspect its content:

```
cd genus
cp ../TEST_LIB2/genus_notie/synt.tcl .
more synt.tcl
```

The script load the liberty timing files and the verilog code to be synthesized. Then it creates constraints for the clock, inputs, and outputs, sets loads, etc. Finally it perform a generic synthesis, maps it to the available cells in the library, and run optimization. Note that the addition of tiehi and tielo cells are deferred to the place and route step later. The outputs including a synthesized netlist and various reports are saved under the directory genus_output.

To run the genus:

```
genus -gui -files synt.tcl
```

Check for warnings and errors. Once the synthesis finishes, the gui will open. You can click the + button besides the Layout tab and select Schematic. This will show the schematic circuit at gate level. In the Design Browser (Left) you can inspect cells, nets, etc. You can also see timing and power reports by using the Power and Timing menus on the top.

3.2 Gate level simulation

Go to the sim_gate directory and follow the instructions in 2.1 to setup the directory (you can copy and paste these files).

Now, we will need to create a copy of the testbench and modify it so that it does not instantiates the RTL code for the SPI_Slave.

```
cp ../src/SPI_Master_TB.sv ../src/SPI_Master_TB_synt.sv
gedit ../src/SPI_Master_TB_synt.sv
```

Comment the include for the SPI_Slave.v. The synthesized netlist for this module will be imported separately in the make.sh file.

```
`timescale 1ns/1ps
`include "SPI_Master.v"
//`include "SPI_Slave.v"
```

Then you need to create a new make.sh file that loads the Verilog models for the standard cells, the synthesized Verilog netlist, the testbench, and run the simulation. You can modify the make.sh that you wrote previously so that the following content is present:

```
rm -rf work*/* work*/.* *.log .simvision *.dsn *.trn *.vcd wave* *.X

#load verilog standard cell library
xmvlog -work work_lib -cdslib ./cds.lib -logfile xmvlog_postsyn.log \
        -errormax 15 -update -linedebug -status -define DISPLAY_PD_PU_EN \
        ../src/KISTA_SOI_STDLIB2_ECSM_TT.v

#load genus synthesized code for the target block
xmvlog -work work_sub -cdslib ./cds.lib -append_log \
       -logfile xmvlog_postsynt.log -errormax 15 -update -linedebug \
       -status ../genus/genus_output/SPI_Slave_synth.v

#load system verilog testbench and include a directory in the path
xmvlog -work work -cdslib ./cds.lib -append_log \
    -logfile xmvlog_postsynt.log -errormax 15 -update -linedebug \
```

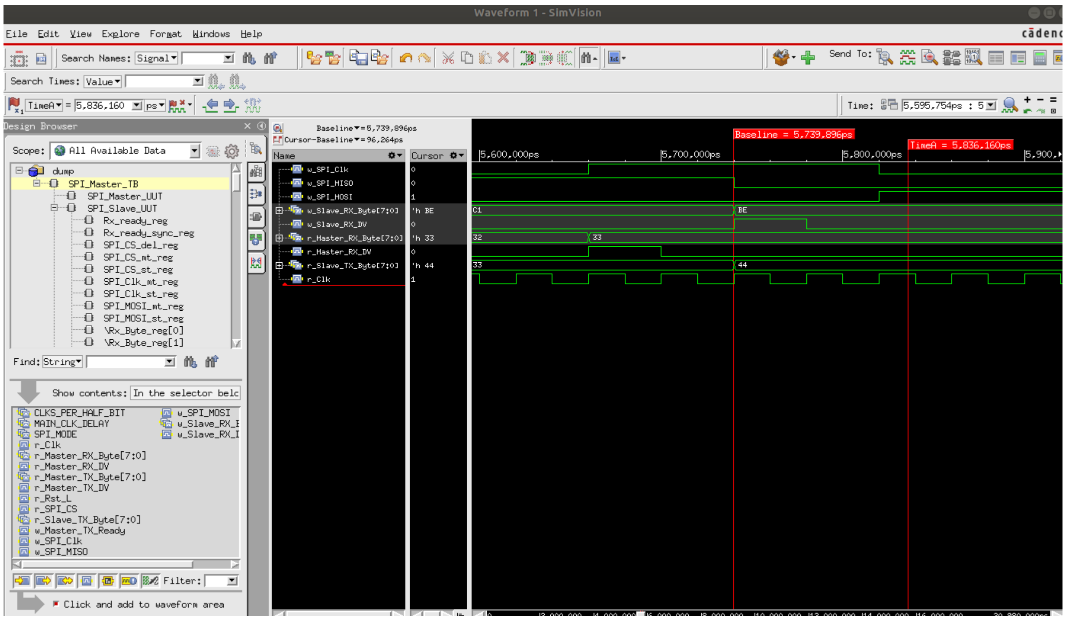


Figure 3: Gate Simulation without delays

```
    -define presyn -status ../src/SPI_Master_TB_synt.sv -incdir ../src \
        -incdir ../genus/genus_output -sv

#elaborate the projet
xmelab -work work -cdslib ./cds.lib -timescale '1ns/1ps' \
    -logfile xmelab_postsyn.log -errormax 30 -access +wc \
    -nowarn CUVWSP -nowarn SDFNCAP -status work.SPI_Master_TB

xmsim -cdslib ./cds.lib -logfile xmsim_postsyn.log \
    -errormax 15 -status work.SPI_Master_TB:module
```

Finally source the script to run the simulation. Check for warnings and errors.

```
source make.sh
```

Open simvision to check the waveforms. Note that KISTA_SOI_STDLIB2_ECSM_TT.v does not contain gate delays, but only placeholders for sdf backannotation. If you want to include the intrinsic gate delays, you need to use instead the verilog model file KISTA_SOI_STDLIB2_ECSM_TT_del.v

```
simvision dump.vcd
```

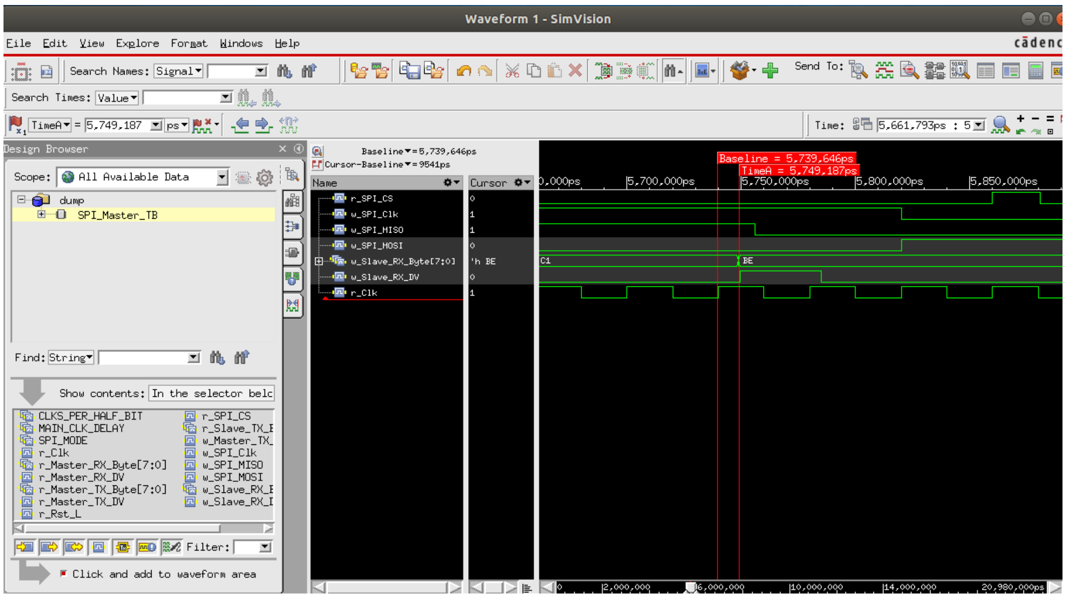


Figure 4: Gate Simulation with delays

4 Place and Route

4.1 Load the design

Go to the innovus directory and start innovus. Note that you should not use innovus & since the terminal needs to be used by innovus for TCL commands.

```
cd innovus
innovus
```

Once innovus starts write the following command to set some global parameters:

```
setDesignMode -process 250
```

Now, go to File -> Design Import and fill the required fields as shown below. The Verilog file is taken from the genus/genus_output directory. We will use the LEF flow, so you have to check LEF Files. The 2 files are located under virtuoso/kista_soi_stdlib2_files/lef. Make sure that the LEF tech file is added before the LEF macro file.

In the “Power Nets” field write VDD! and in “Ground Nets” write VSS!. You need to load a MMMC View Definition File which contains information about corners, captables, constraints, and analysis. You can create a new configuration by clicking on Create Analysis Configuration, or use the example file

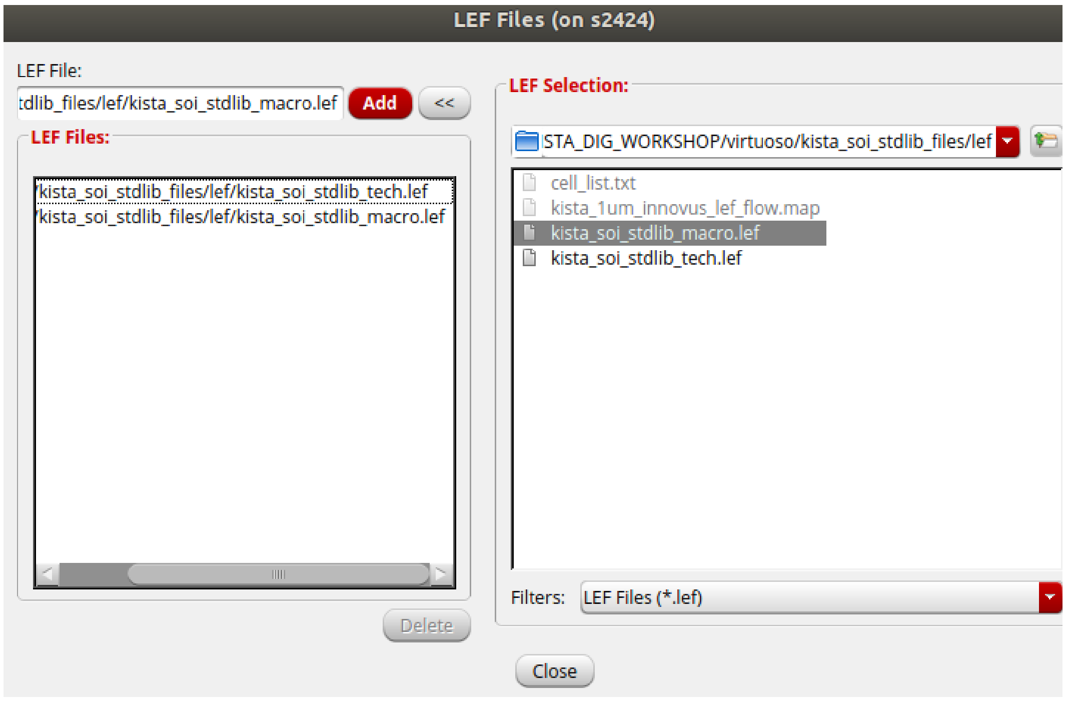

LEF Files (on s2424)
LEF File:
tdlib_files/lef/kista_soi_stdlib_macro.lef
Add
<<
LEF Files:
/kista_soi_stdlib_files/lef/kista_soi_stdlib_tech.lef
/kista_soi_stdlib_files/lef/kista_soi_stdlib_macro.lef
Delete
LEF Selection:
STA_DIG_WORKSHOP/virtuoso/kista_soi_stdlib_files/lef
cell_list.txt
kista_1um_innovus_lef_flow.map
kista_soi_stdlib_macro.lef
kista_soi_stdlib_tech.lef
Filters:
LEF Files (*.lef)
Close

Figure 5: Selection of LEF files

MMMC_KISTA_only_typical.view located under virtuoso/kista_soi_stdlib2_files/innovus_scripts. Note that this example only creates a functional view for typical corners. Fig. 6 shows how the configuration should look. Press OK.

Check for errors and warnings.

Now connect the global nets:

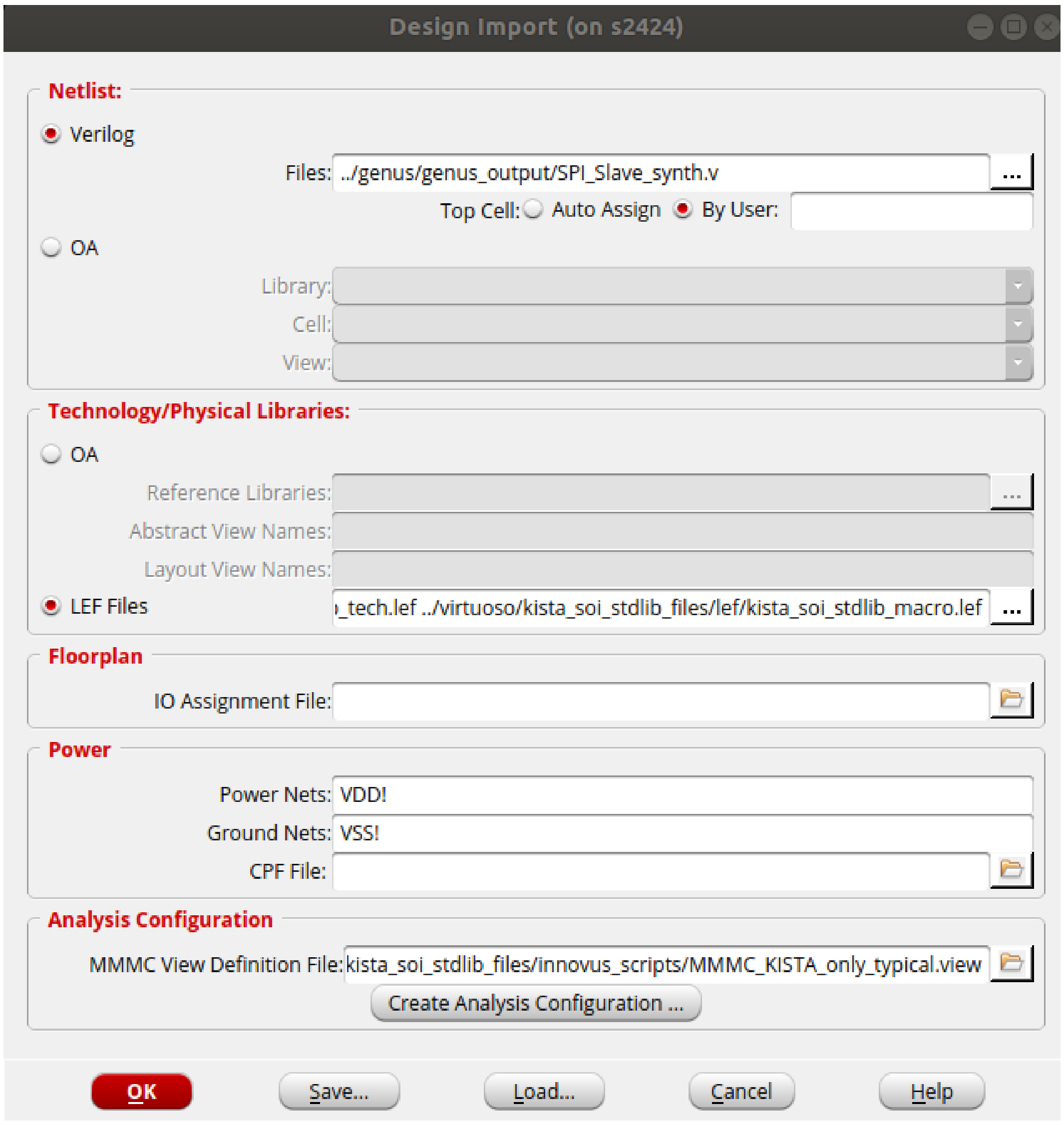
```
clearGlobalNets
globalNetConnect VSS! -type pgpin -pin VSS!
    -instanceBasename * -hierarchicalInstance {}
globalNetConnect VDD! -type pgpin -pin VDD!
    -instanceBasename * -hierarchicalInstance {}
```

Click on "File -> Save...", Data type "Innovus", write "SPI_Slave_loaded" as file name. Then click on "OK".

4.2 Floorplan

Go to Floorplan -> Specify Floorplan

Here there are several parameters that can be changed such as the aspect ratio, core utilization, and margins. A good floorplan is extremely important in


Design Import (on s2424)

Netlist:

[✓ Verilog]

Files: ../genus/genus_output/SPI_Slave_synth.v

Top Cell: Auto Assign [✓ By User:]

OA

Library:

Cell:

View:

Technology/Physical Libraries:

OA

Reference Libraries:

Abstract View Names:

Layout View Names:

[✓ LEF Files] ı_tech.lef ../virtuoso/kista_soi_stdlib_files/lef/kista_soi_stdlib_macro.lef

Floorplan

IO Assignment File:

Power

Power Nets: VDD!

Ground Nets: VSS!

CPF File:

Analysis Configuration

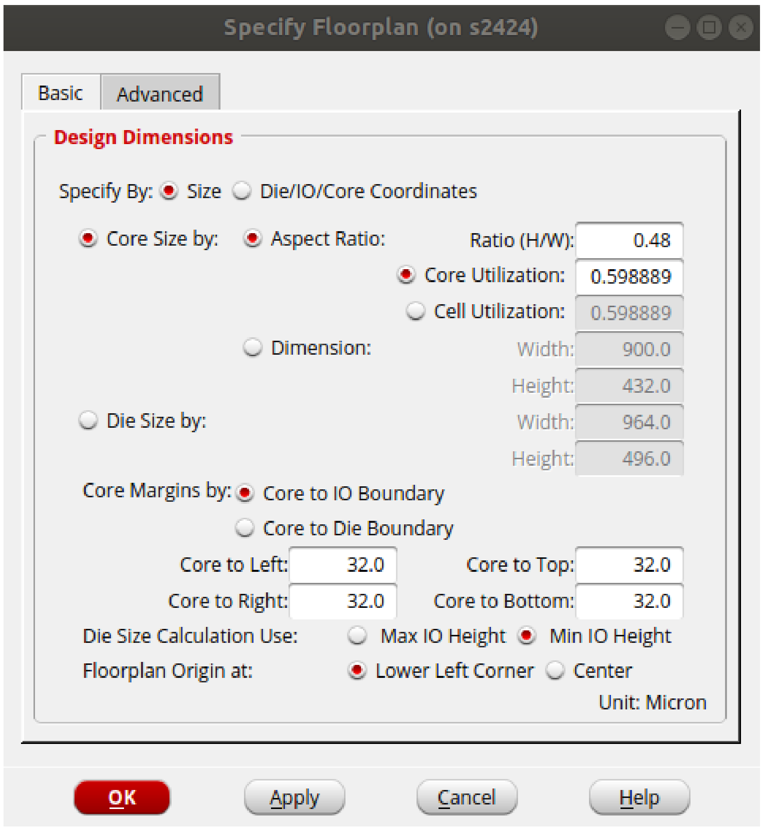
MMMC View Definition File: kista_soi_stdlib_files/innovus_scripts/MMMC_KISTA_only_typical.view

Figure 6: Design Import configuration

order for the routing tools to do a good job. A poor floor plan typically results in DRC violations.

Change the Ratio (H/W) to 0.5 and the Core Utilization to 0.6. Note that small designs have a rather poor core utilization since stripes and routing will take considerable space.

The core margins need to be defined. Here is is necessary to leave space for wide power and ground rings. The margins depend on the width of these tracks which are defined by the maximum power consumption and metal current densities. We will write 32.0 to all sides. The final configuration is



Design Dimensions

Specify By: [✓ Size] Die/IO/Core Coordinates

[✓ Core Size by:] [✓ Aspect Ratio:] Ratio (H/W): 0.48

[✓ Core Utilization:] 0.598889

Cell Utilization: 0.598889

Dimension: Width: 900.0 Height: 432.0

Die Size by: Width: 964.0 Height: 496.0

Core Margins by: [✓ Core to IO Boundary] Core to Die Boundary

Core to Left:	32.0	Core to Top:	32.0
Core to Right:	32.0	Core to Bottom:	32.0

Die Size Calculation Use: Max IO Height [✓ Min IO Height]

Floorplan Origin at: [✓ Lower Left Corner] Center

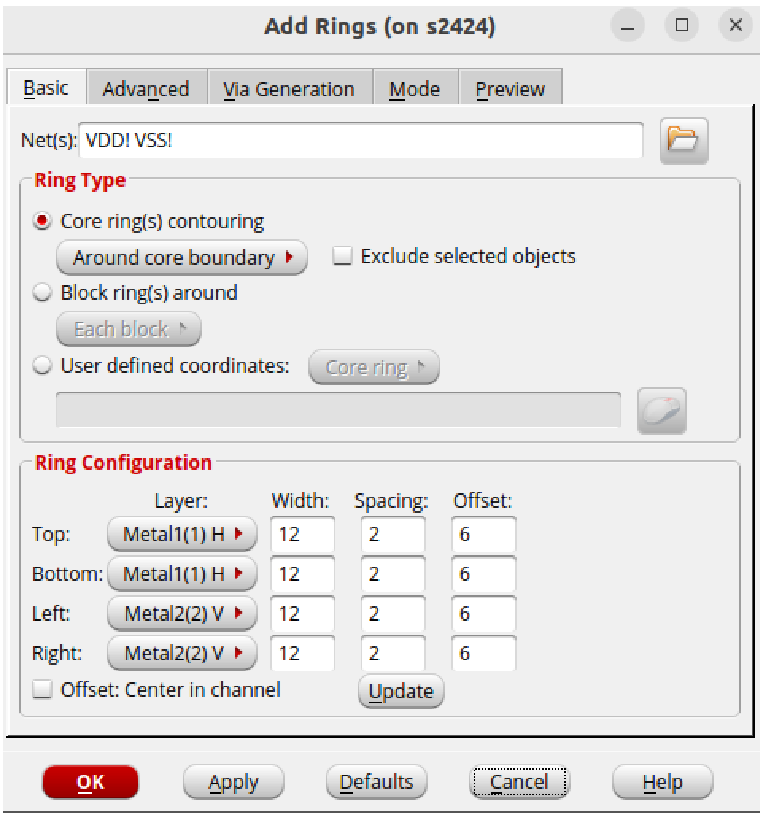
Unit: Micron

Figure 7: Floorplan configuration

shown in fig. 7. Press OK and save the design as SPI_Slave_loaded_floorplan.

4.3 Power Rings

Go to Power->Power Planning->Add Ring. Open the Net(s) folder and select the VSS! and VDD! nets. Then change the ring configuration.Change the widths of the metal tracks depending on the maximum current density that is expected in this design (the output power can be extracted from genus, check max current densities for this process in the PDK manual, and take a generous margins in order to avoid electromigration problems). We will set all Widths to 12, Spacings to 2, and Offsets to at least 4 (recommended 6 for Top and Bottom!). Click on "Update" and spacings will be automatically calculated from the technology files. Press OK and you will see that power rings have been added to the layout (you may need to click on the Floorplan view or Physical view button on the toolbar at the top right corner)



Net(s): VDD! VSS!

Ring Type

- [● Core ring(s) contouring] — Around core boundary; ☐ Exclude selected objects
- ○ Block ring(s) around — Each block
- ○ User defined coordinates: Core ring

Ring Configuration

	Layer:	Width:	Spacing:	Offset:
Top:	Metal1(1) H	12	2	6
Bottom:	Metal1(1) H	12	2	6
Left:	Metal2(2) V	12	2	6
Right:	Metal2(2) V	12	2	6

☐ Offset: Center in channel — Update

Figure 8: Configuration for Adding Rings

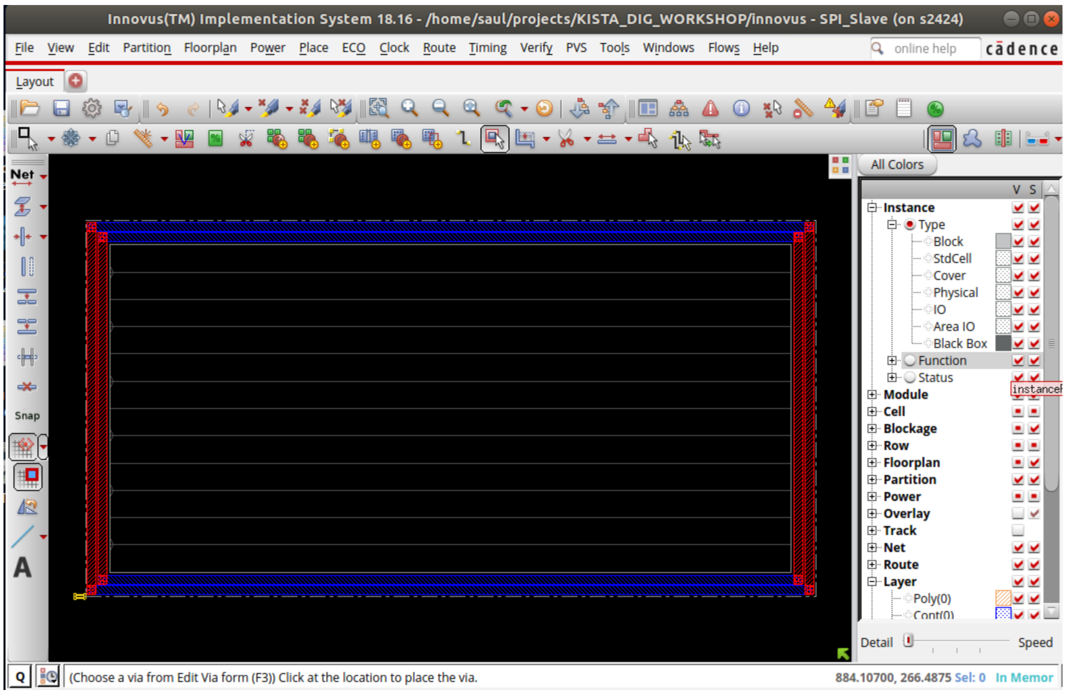
4.4 Optional: Power Stripes

Go to Power->Power Planning -> Add Stripe

Open the folder and select the gnd and vdd nets. Then change Layer to M2 and the direction to Vertical. Change also the width. The "Set Pattern", "Stripe Boundary" and "First/Last Stripe" fields offer different alternatives to place the strips. For instance, use "Number of sets: 2". Then in "First/Last Stripe" select "Start from: left" and the Relative from core or selected area "Start: 400" and "Stop: 400" as shown in Fig. 10.

4.5 Routing Power and Ground Nets

Go to Route-> Special Route and select the VDD! and VSS! nets. Leave the other options as shown in Fig. 11, and press OK. the VDD! and VSS! nets will be connected to the power ring as shown in Fig. 12. If you use power stripes, then click on the "Via Generation" tab and select "Stripe" in the "Make Via Connections To:" section.


Innovus(TM) Implementation System 18.16 - /home/saul/projects/KISTA_DIG_WORKSHOP/innovus - SPI_Slave (on s2424)

(Choose a via from Edit Via form (F3)) Click at the location to place the via.

884.10700, 266.4875 Sel: 0 In Memor

Figure 9: Power Rings

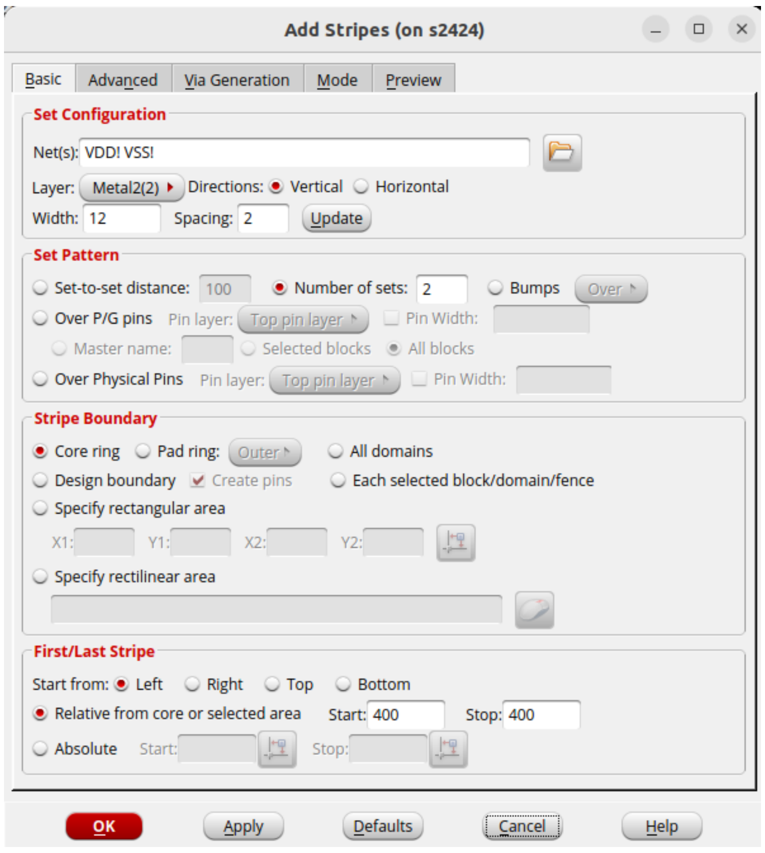
This is a good time to go to Verify->DRC and run a basic DRC check. Save the design as SPI_Slave_loaded_floorplan_power.

4.6 Pin placement

Go to Edit->Pin Editor. Here you can select the pins or arrays individually or in groups (use Ctrl + left click). You can also select any suitable option in the Location group. This will enable the Side/Edge drop-down menu. Here you can select in which side of the block you want to put the pin or pins, etc. You can also select the metal layer for the pin. The options Starting X/Y, spacing, and other parameters should be configured depending on the option that was chosen for Location.

Start by placing all SPI_* pins on the top. First, select the pins. Then Select "Top" in the "Side/Edge" drop-down menu. Select Metal3 in "Layer". In the Location options select "Spread", and in "Spread Type" select "From Center". Change "Spacing" to 12.0. The form should look like Fig. 13. Press "Apply" and you will see in the layout that these pins have been placed on the top. Place the rest of the pins at the bottom.

Save the design as SPI_Slave_loaded_power_pin.


Add Stripes (on s2424)

Basic | Advanced | Via Generation | Mode | Preview

Set Configuration

Net(s): VDD! VSS!

Layer: Metal2(2) Directions: [✓ Vertical] Horizontal

Width: 12 Spacing: 2 Update

Set Pattern

Set-to-set distance: 100 [✓ Number of sets: 2] Bumps Over

Over P/G pins Pin layer: Top pin layer Pin Width:

Master name: Selected blocks [✓ All blocks]

Over Physical Pins Pin layer: Top pin layer Pin Width:

Stripe Boundary

[✓ Core ring] Pad ring: Outer All domains

Design boundary [✓ Create pins] Each selected block/domain/fence

Specify rectangular area

X1: Y1: X2: Y2:

Specify rectilinear area

First/Last Stripe

Start from: [✓ Left] Right Top Bottom

[✓ Relative from core or selected area] Start: 400 Stop: 400

Absolute Start: Stop:

OK Apply Defaults Cancel Help

Figure 10: Configuration for Adding Stripes

4.7 Placement and Pre-CTS Optimization

Go to Tools->Set Mode->Mode Setup and select "Placement" under "List of Modes". Make sure that the following options in Fig. 15 are enabled/disabled. In particular check that the placement is Timing Driven, and that Place IO Pins is disabled. Press Apply.

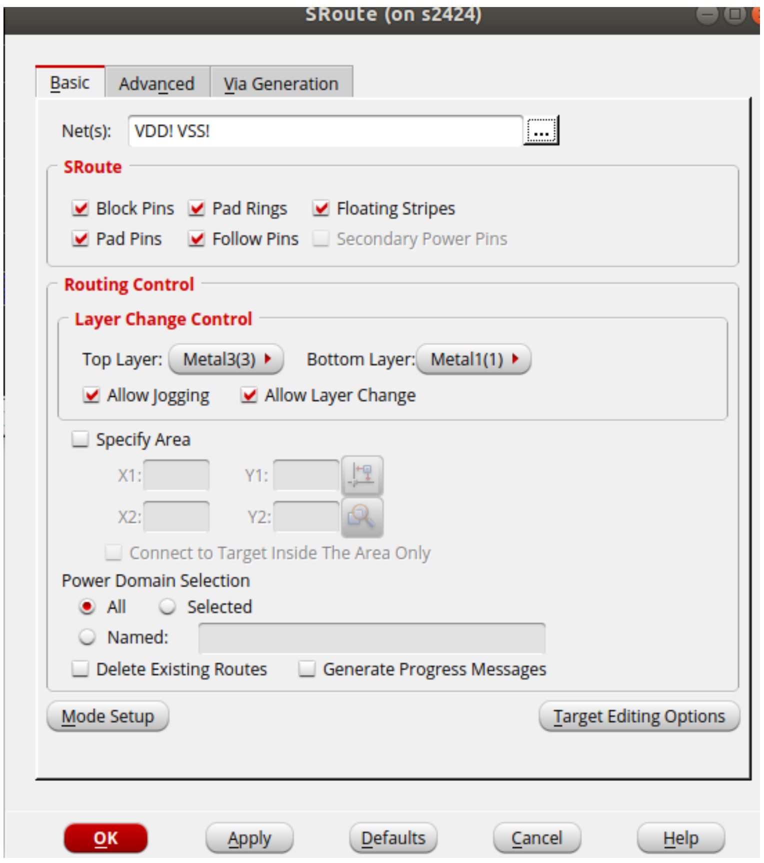
Then select "TieHiLo" and click on "Select" and select the TIEHI and TIELO cells. Enable Maximum Fanout and set it to 10. The configuration should look as in fig. 16. Press Apply, and then OK.

Now you can place the standard cells:

```
place_opt_design
```

Check the log including the timing summary. Then check for placement violations:

```
checkPlace
```



SRoute (on s2424)

Basic | Advanced | Via Generation

Net(s): VDD! VSS!

SRoute

[✓] Block Pins [✓] Pad Rings [✓] Floating Stripes
[✓] Pad Pins [✓] Follow Pins [ ] Secondary Power Pins

Routing Control

Layer Change Control

Top Layer: Metal3(3) Bottom Layer: Metal1(1)
[✓] Allow Jogging [✓] Allow Layer Change

[ ] Specify Area
X1: Y1:
X2: Y2:
[ ] Connect to Target Inside The Area Only

Power Domain Selection
(● All) ( ) Selected
( ) Named:

[ ] Delete Existing Routes [ ] Generate Progress Messages

Figure 11: SRoute configuration

Now you can see the cells placed and pre-routed in the Physical View (Fig. 17).

Note: At this point if you run Verify -> DRC, there will be violations since the early Global autorouter is used during the placement trials.

Now, add the TIEHI and TIELO cells. Go to Place -> Tie HI/LO cell -> Add and make sure that the form looks like Fig. 18. Press OK, and check in the log how many Tie Cells have been added.

Select Timing->Report Timing and check "Pre-CTS". Run the timing checks only for Setup. If there are violations, then it is needed to run Timing Optimization: Eco->Optimize Design... Check timing again after optimization.

At this point if desired it is possible to check a summary for the worst path:

```
report_timing
```

A graphical tool to see the timing of the paths can be invoked by: Timing -> Debug Timing. Save the design as: SPI_Slave_loaded_floorplan_power_pin_placed.

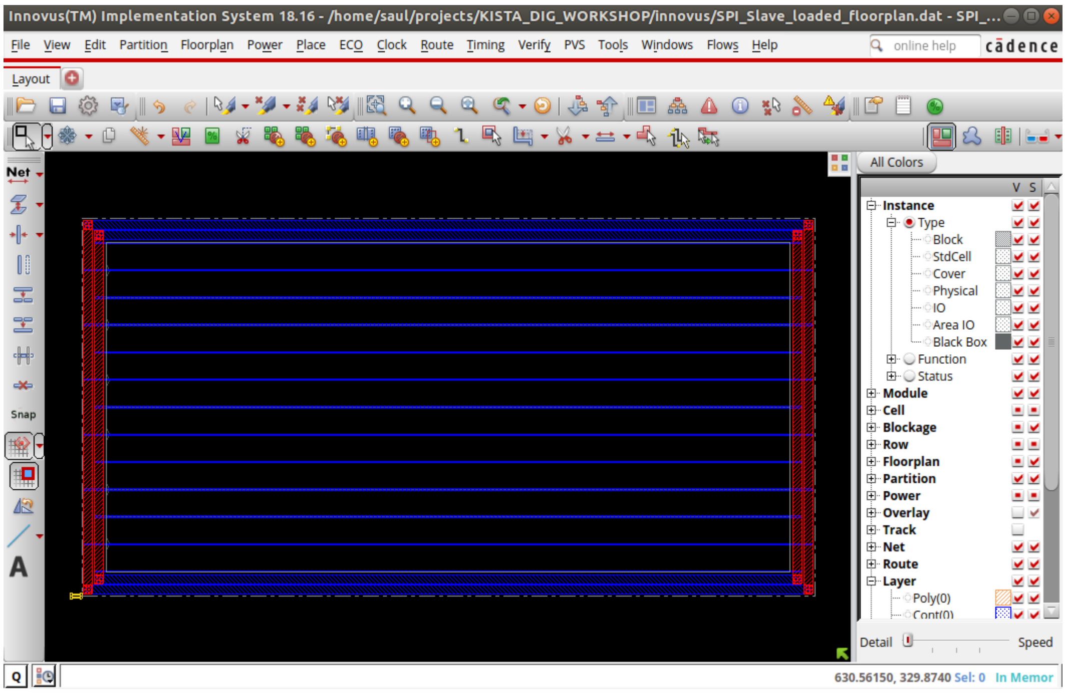

Innovus(TM) Implementation System 18.16 - /home/saul/projects/KISTA_DIG_WORKSHOP/innovus/SPI_Slave_loaded_floorplan.dat - SPI_...
File View Edit Partition Floorplan Power Place ECO Clock Route Timing Verify PVS Tools Windows Flows Help
online help
cadence
Layout
Net
Snap
All Colors
V S
Instance
Type
Block
StdCell
Cover
Physical
IO
Area IO
Black Box
Function
Status
Module
Cell
Blockage
Row
Floorplan
Partition
Power
Overlay
Track
Net
Route
Layer
Poly(0)
Cont(0)
Detail
Speed
630.56150, 329.8740 Sel: 0 In Memor

Figure 12: Routing of VDD! and GND!

4.8 Clock Tree Synthesis CTS

Before creating the clock tree, we need to do a manual setup in innovus. Go to Tools -> Set Mode -> Mode Setup. Select the Route/EM tab and make sure that the "Top Preferred Layer" for Non-Leaf nets and Leaf nets is Metal3 and also that "Bottom Preferred Layer" is Metal1 (Fig. 19). Press Apply and OK.

First create a ccopt spec fila and source it:

```
create_ccopt_clock_tree_spec -file ccopt_native.spec
source ccopt_native.spec
```

Select the inverter and buffer cells that will be used in the clock tree:

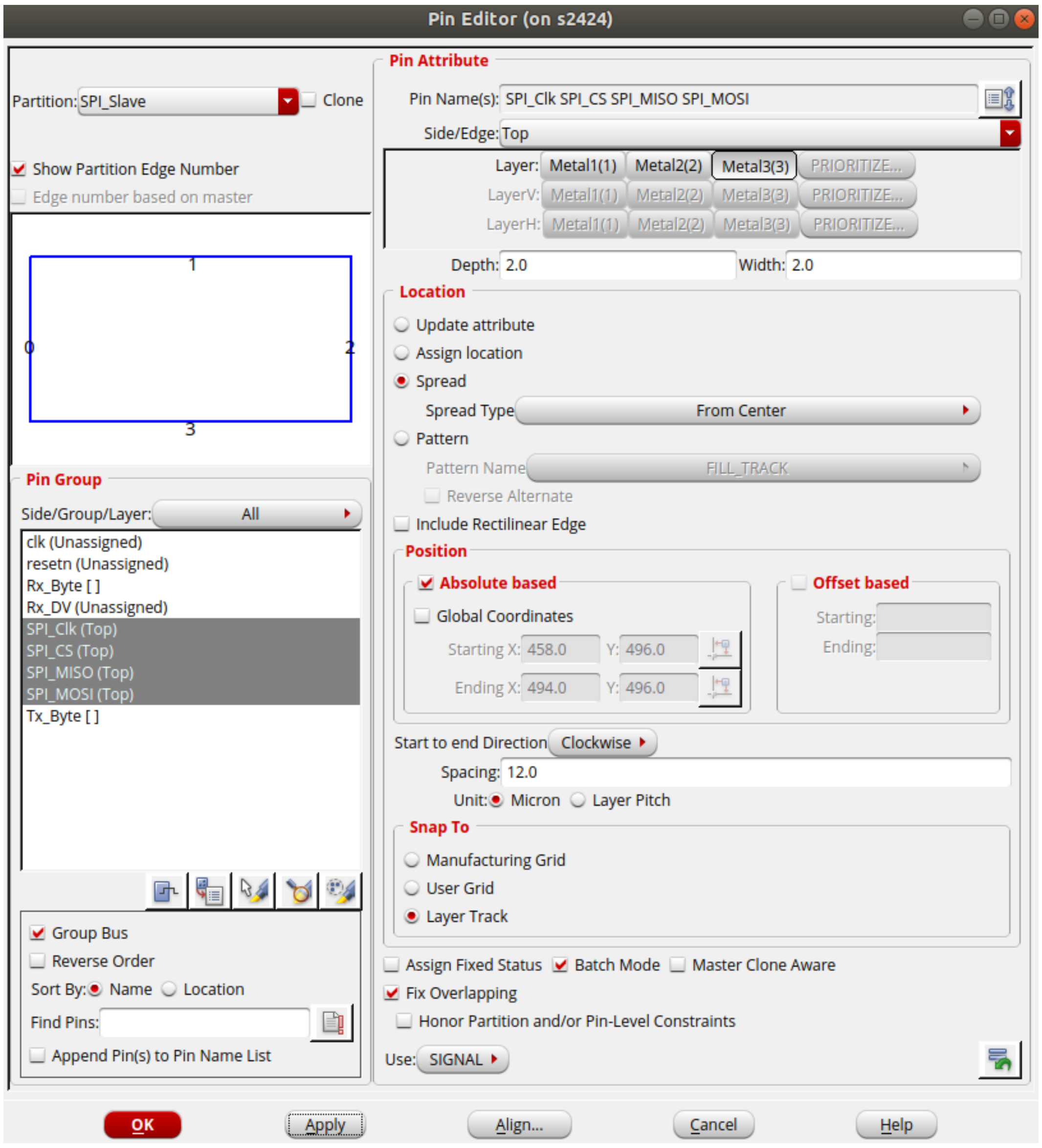
```
set_ccopt_mode -cts_inverter_cells { INVX1 INVX2 INVX3 INVX4 }
set_ccopt_mode -cts_buffer_cells { BUFX2 BUFX3 BUFX4 BUFX8 }
```

Now create the clock tree:

```
ccopt_design -cts
```

Check the timings for setup:

```
timeDesign -postCTS
```


Pin Editor (on s2424)

Partition: SPI_Slave [ ] Clone

[✓] Show Partition Edge Number
[ ] Edge number based on master

Pin Group
Side/Group/Layer: All

- clk (Unassigned)
- resetn (Unassigned)
- Rx_Byte [ ]
- Rx_DV (Unassigned)
- SPI_Clk (Top)
- SPI_CS (Top)
- SPI_MISO (Top)
- SPI_MOSI (Top)
- Tx_Byte [ ]

[✓] Group Bus
[ ] Reverse Order
Sort By: [✓ Name] Location
Find Pins:
[ ] Append Pin(s) to Pin Name List

Pin Attribute
Pin Name(s): SPI_Clk SPI_CS SPI_MISO SPI_MOSI
Side/Edge: Top
Layer: Metal1(1) Metal2(2) [✓ Metal3(3)] PRIORITIZE...
LayerV: Metal1(1) Metal2(2) Metal3(3) PRIORITIZE...
LayerH: Metal1(1) Metal2(2) Metal3(3) PRIORITIZE...
Depth: 2.0 Width: 2.0

Location
Update attribute
Assign location
[✓ Spread]
Spread Type: From Center
Pattern
Pattern Name: FILL_TRACK
[ ] Reverse Alternate
[ ] Include Rectilinear Edge

Position
[✓] Absolute based
[ ] Global Coordinates
Starting X: 458.0 Y: 496.0
Ending X: 494.0 Y: 496.0
[ ] Offset based
Starting:
Ending:

Start to end Direction: Clockwise
Spacing: 12.0
Unit: [✓ Micron] Layer Pitch

Snap To
Manufacturing Grid
User Grid
[✓ Layer Track]

[ ] Assign Fixed Status [✓] Batch Mode [ ] Master Clone Aware
[✓] Fix Overlapping
[ ] Honor Partition and/or Pin-Level Constraints
Use: SIGNAL

Figure 13: Pin configuration

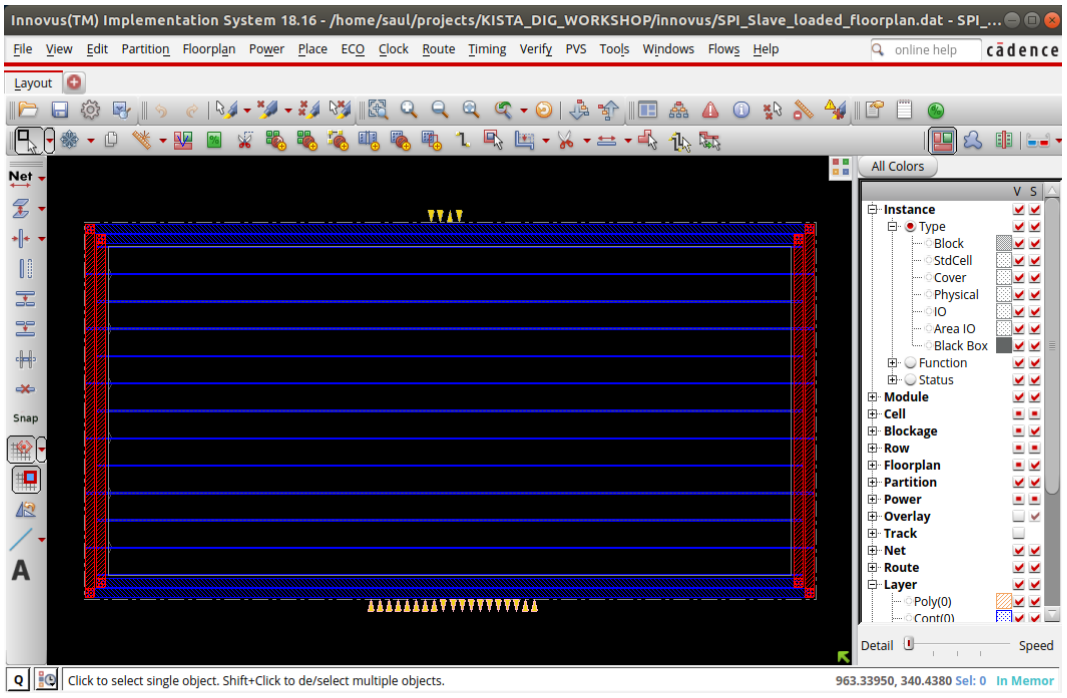
If there are setup violations, you can run:

```
optDesign -postCTS
```

Check the timings for hold:

```
timeDesign -postCTS -hold
```

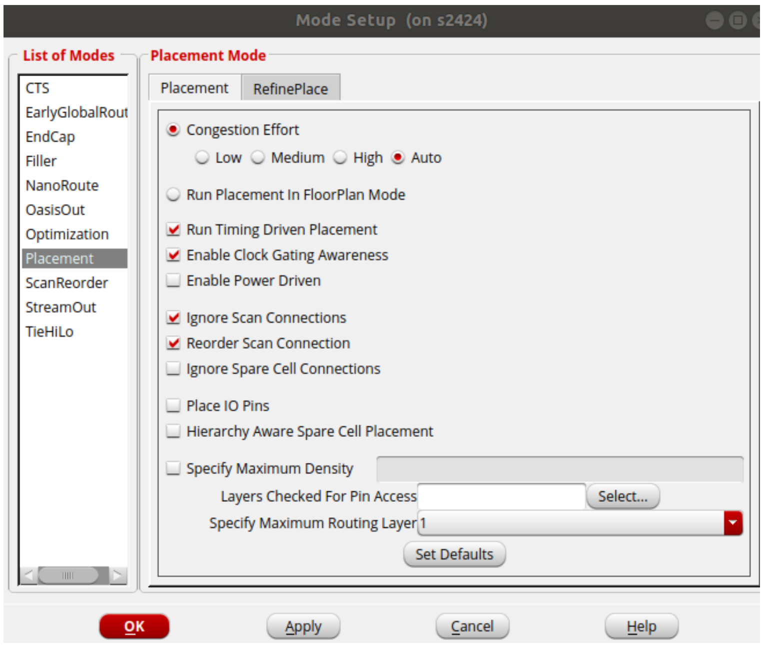
and optimize it if needed. (Maybe not all will be fixed in these steps. That is Ok, as the remaining will be fixed in global routing)


Innovus(TM) Implementation System 18.16 - /home/saul/projects/KISTA_DIG_WORKSHOP/innovus/SPI_Slave_loaded_floorplan.dat - SPI_...

Click to select single object. Shift+Click to de/select multiple objects.

963.33950, 340.4380 Sel: 0 In Memor

Figure 14: Pin placement layout


Mode Setup (on s2424)

List of Modes: CTS, EarlyGlobalRout, EndCap, Filler, NanoRoute, OasisOut, Optimization, Placement, ScanReorder, StreamOut, TieHiLo

Placement Mode

Placement | RefinePlace

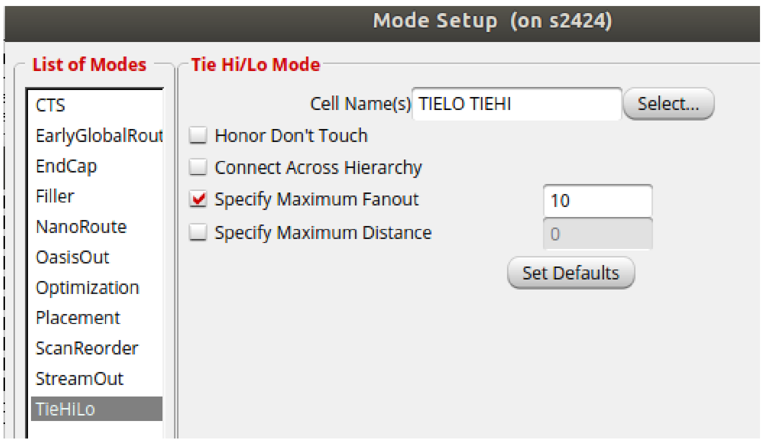
- [✓] Congestion Effort
  - Low Medium High [✓ Auto]
- [ ] Run Placement In FloorPlan Mode
- [✓] Run Timing Driven Placement
- [✓] Enable Clock Gating Awareness
- [ ] Enable Power Driven
- [✓] Ignore Scan Connections
- [✓] Reorder Scan Connection
- [ ] Ignore Spare Cell Connections
- [ ] Place IO Pins
- [ ] Hierarchy Aware Spare Cell Placement
- [ ] Specify Maximum Density

Layers Checked For Pin Access

Specify Maximum Routing Layer 1

Figure 15: Standard Cell Placement Options

```
optDesign -postCTS -hold
```



Mode Setup (on s2424)

List of Modes

- CTS
- EarlyGlobalRout
- EndCap
- Filler
- NanoRoute
- OasisOut
- Optimization
- Placement
- ScanReorder
- StreamOut
- TieHiLo

Tie Hi/Lo Mode

Cell Name(s) TIELO TIEHI Select...

- [ ] Honor Don't Touch
- [ ] Connect Across Hierarchy
- [✓] Specify Maximum Fanout 10
- [ ] Specify Maximum Distance 0

Set Defaults

Figure 16: TieHiLo Options

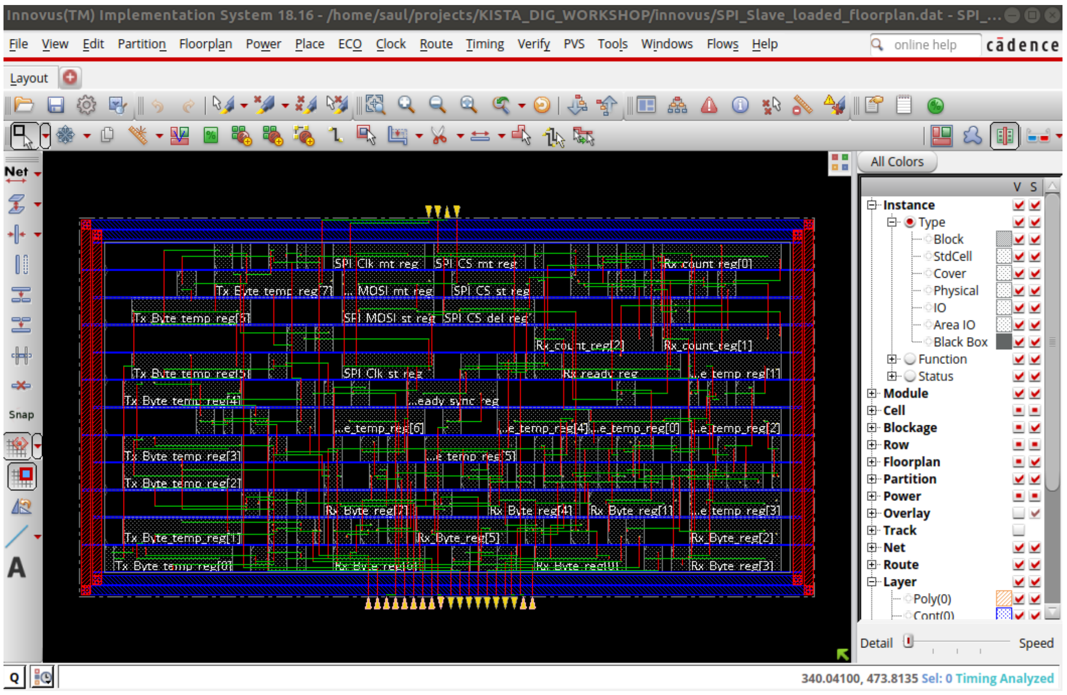
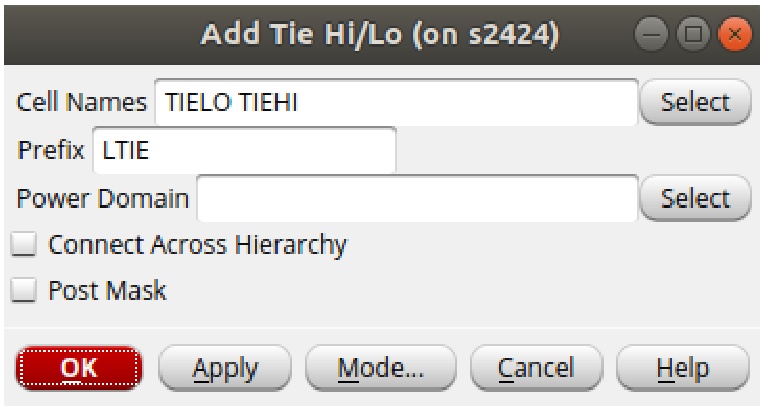

Innovus(TM) Implementation System 18.16 - /home/saul/projects/KISTA_DIG_WORKSHOP/innovus/SPI_Slave_loaded_floorplan.dat - SPI_...
File View Edit Partition Floorplan Power Place ECO Clock Route Timing Verify PVS Tools Windows Flows Help
online help
cadence
Layout
Net
Snap
All Colors
Instance
Type
Block
StdCell
Cover
Physical
IO
Area IO
Black Box
Function
Status
Module
Cell
Blockage
Row
Floorplan
Partition
Power
Overlay
Track
Net
Route
Layer
Poly(0)
Cont(0)
Detail
Speed
340.04100, 473.8135 Sel: 0 Timing Analyzed

Figure 17: Standard cell placement

You can visually inspect the clock by selecting Clock -> CCOpt Clock Tree Debugger. Save the design as: SPI_Slave_loaded_floorplan_power_pin_placed_cts.

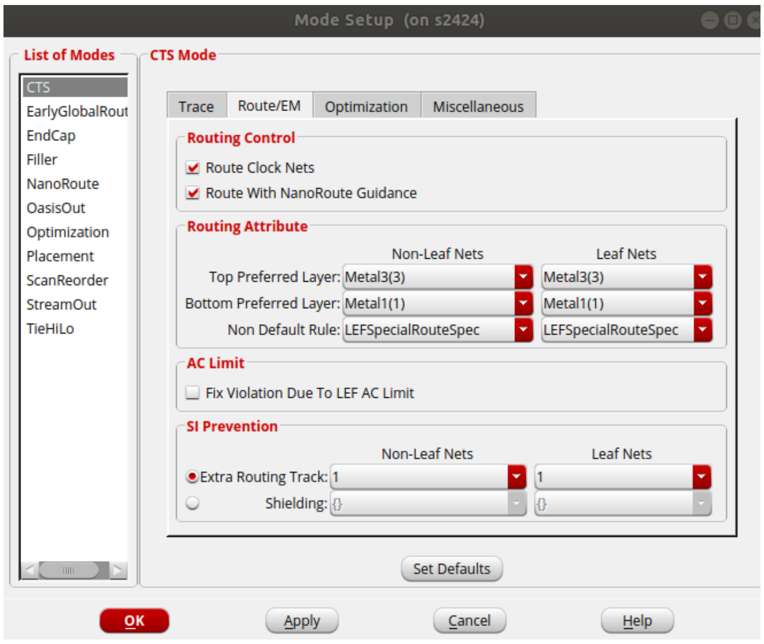
4.9 Route with Nanoroute

We do some settings first:


Add Tie Hi/Lo (on s2424)

Cell Names: TIELO TIEHI
Prefix: LTIE
Power Domain:
Connect Across Hierarchy: ☐
Post Mask: ☐

Figure 18: Add Tie HI/LO


Mode Setup (on s2424)

List of Modes: CTS, EarlyGlobalRout, EndCap, Filler, NanoRoute, OasisOut, Optimization, Placement, ScanReorder, StreamOut, TieHiLo

CTS Mode — Route/EM

Routing Control
- [✓] Route Clock Nets
- [✓] Route With NanoRoute Guidance

Routing Attribute

	Non-Leaf Nets	Leaf Nets
Top Preferred Layer:	Metal3(3)	Metal3(3)
Bottom Preferred Layer:	Metal1(1)	Metal1(1)
Non Default Rule:	LEFSpecialRouteSpec	LEFSpecialRouteSpec

AC Limit
- [ ] Fix Violation Due To LEF AC Limit

SI Prevention

	Non-Leaf Nets	Leaf Nets
● Extra Routing Track:	1	1
○ Shielding:	{}	{}

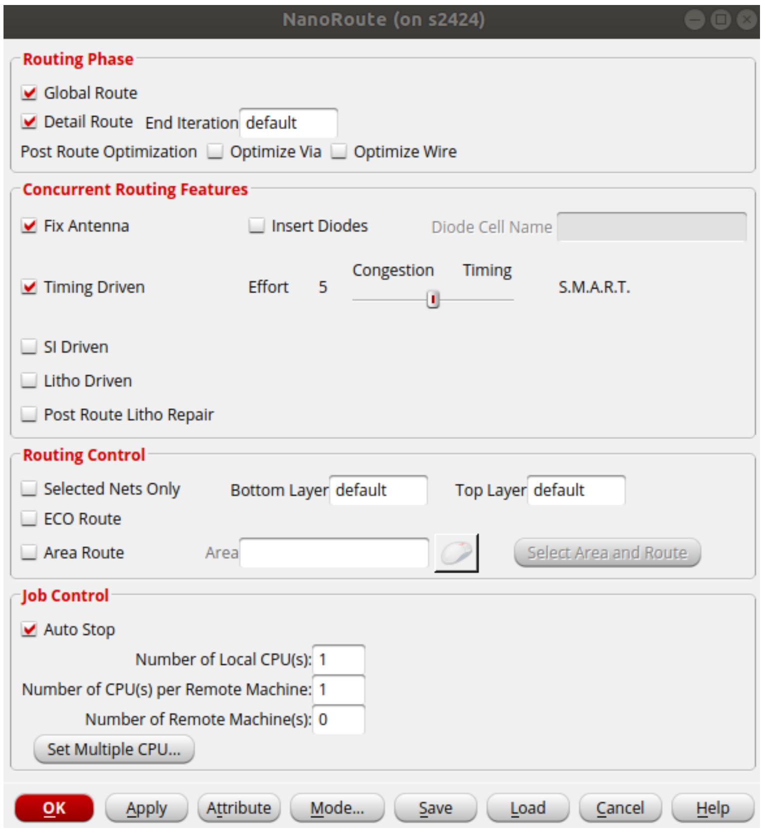
Figure 19: CTS options

```
setAnalysisMode -analysisType onChipVariation -cppr both
setOptMode -fixDrc true
```

Then go to Route-> NanoRoute -> Route and make sure that the form looks like Fig. 20

After the router finishes, go to Verify -> DRC and Verify -> Connectivity. You can run the ECO -> Optimization to try to correct DRCs (Fig. 21). Click on SI Options and make sure that “Fix DRC” is selected.

After you have checked that the design does not have DRC/Connectivity



NanoRoute (on s2424)

Routing Phase

[✓ Global Route]

[✓ Detail Route] End Iteration: default

Post Route Optimization: Optimize Via, Optimize Wire

Concurrent Routing Features

[✓ Fix Antenna] Insert Diodes Diode Cell Name

[✓ Timing Driven] Effort 5 Congestion — Timing S.M.A.R.T.

SI Driven

Litho Driven

Post Route Litho Repair

Routing Control

Selected Nets Only Bottom Layer: default Top Layer: default

ECO Route

Area Route Area: Select Area and Route

Job Control

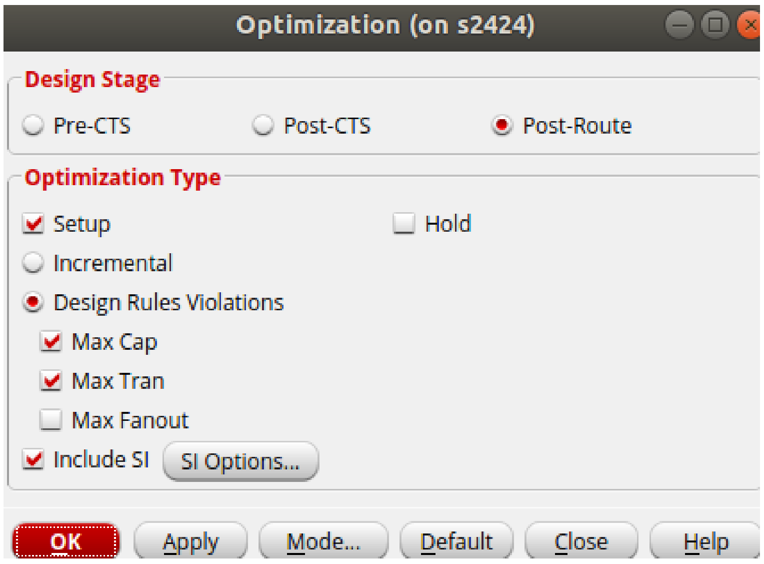
[✓ Auto Stop]

Number of Local CPU(s): 1

Number of CPU(s) per Remote Machine: 1

Number of Remote Machine(s): 0

Figure 20: Nanoroute options



Optimization (on s2424)

Design Stage

Pre-CTS Post-CTS [✓ Post-Route]

Optimization Type

[✓ Setup] Hold

Incremental

[✓ Design Rules Violations]

[✓ Max Cap]

[✓ Max Tran]

Max Fanout

[✓ Include SI]

Figure 21: ECO optimizer options

violations, you can check timings:

```
timeDesign -postRoute
timeDesign -postRoute -hold
```

Run setup and hold optimization simultaneously to fix any remaining timing violations:

```
optDesign -postRoute -setup -hold
```

Check for DRC/Connectivity and save the design as:
SPI_Slave__floorplan_power_pin_placed_cts_routed

4.10 Placement of Filler Cells

Places that are not used by standard cells can be filled with decoupling capacitors and empty fillers:

```
setFillerMode -add_fillers_with_drc false
addFiller -prefix FILLCAP -cell DECAP4 DECAP3 DECAP2
addFiller -prefix FILL -cell FILL4 FILL2 FILL1
```

Check the added cells:

```
pdi report_design
```

Run a last DRC and connectivity check and save the design as:
SPI_Slave__floorplan_power_pin_placed_cts_routed_final

4.11 Export the Design

First you will export the routed netlist. There are at least 2 netlists that are needed. The first netlist does not contain decoupling capacitors and will be used for post-layout simulation. Another list includes the decoupling capacitors and will be used for LVS in virtuoso:

```
saveNetlist output/design_pnr.v
saveNetlist output/design_pnr_cap.v -includePhysicalCell
        { DECAP4 DECAP3 DECAP2}
```

Now the physical implementation will be exported. Go to Save-> GDS/OASIS. Fill the output name of the gds file, and provide a layer mapping file. Note that this mapping file is not exactly the one used in Virtuoso, but one created for innovus. The mapping file is located at:

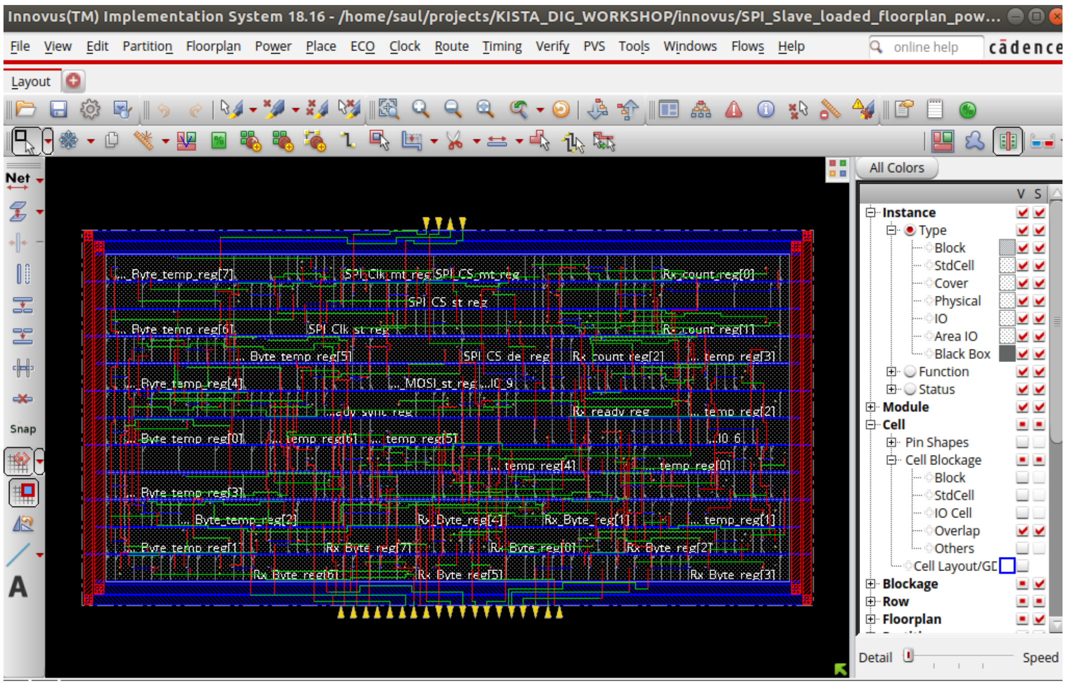
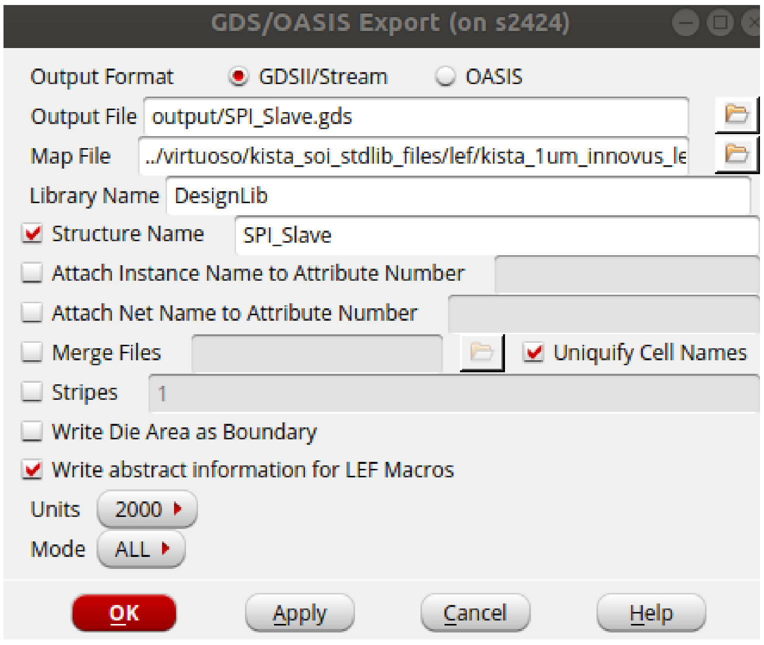

Innovus(TM) Implementation System 18.16 - /home/saul/projects/KISTA_DIG_WORKSHOP/innovus/SPI_Slave_loaded_floorplan_pow...
File View Edit Partition Floorplan Power Place ECO Clock Route Timing Verify PVS Tools Windows Flows Help
online help
cādence
Layout
Net
Snap
All Colors
Instance
Type
Block
StdCell
Cover
Physical
IO
Area IO
Black Box
Function
Status
Module
Cell
Pin Shapes
Cell Blockage
Block
StdCell
IO Cell
Overlap
Others
Cell Layout/GD
Blockage
Row
Floorplan
Detail
Speed

Figure 22: Final routed layout



GDS/OASIS Export (on s2424)

Output Format	[✓ GDSII/Stream] OASIS
Output File	output/SPI_Slave.gds
Map File	../virtuoso/kista_soi_stdlib_files/lef/kista_1um_innovus_le
Library Name	DesignLib
[✓] Structure Name	SPI_Slave
[ ] Attach Instance Name to Attribute Number	
[ ] Attach Net Name to Attribute Number	
[ ] Merge Files	[✓] Uniquify Cell Names
[ ] Stripes	1
[ ] Write Die Area as Boundary	
[✓] Write abstract information for LEF Macros	
Units	2000
Mode	ALL

Figure 23: GDS export form

```
virtuoso/kista_soi_stdlib_files/lef/kista_1um_innovus_lef_flow.map
```

Make sure that "Structure Name", "Uniquify Cell Names", and "Write abstract information for LEF Macros" are checked as shown in Fig. 23.

To export timing information we use the following:

```
extractRC
rcOut -spf output/typ.spf -rc_corner typ_rc
```

To see what analysis views are available:

```
all_hold_analysis_views
all_setup_analysis_views
```

Innovus will answer the available analysis modes, which in this case is only one: typ_functional_mode. We will export the delays for this mode so that they can be annotated during post-layout simulation.

```
write_sdf -view typ_functional_mode "output/typ_functional_1_8V_25C.sdf"
```

5 Post-Layout Simulation Simulation

Go to the sim_postlay directory and follow the instructions in 2.1 to setup the directory (you can copy and paste these files). Now, we will need to create a copy of the testbench and modify it so that it does not instantiates the RTL code for the SPI_Slave.

```
cp ../src/SPI_Master_TB.sv ../src/SPI_Master_TB_pnr.sv
gedit ../src/SPI_Master_TB_pnr.sv
```

Comment the include for the SPI_Slave.v. The placed and routed netlist for this module will be imported separately in the make.sh file.

```
`timescale 1ns/1ps
`include "SPI_Master.v"
//`include "SPI_Slave.v"
```

Before "endmodule" copy the following code:

```
initial begin
    $sdf_annotate ("../innovus/output/typ_functional_1_8V_25C.sdf",SPI_Slave_UUT
end
```

This will annotate the interconnection delays extracted by innovus.

Then you need to create a new make.sh file that loads the Verilog models for the standard cells, the synthesized Verilog netlist, the testbench, and run the simulation. You can reuse most of the make.sh that was used for gate simulation. The following content needs to be included:

```
rm -rf work*/* work*/.* *.log .simvision *.dsn *.trn *.vcd wave* *.X

#load verilog standard cell library
xmvlog -work work_lib -cdslib ./cds.lib -logfile xmvlog_postlay.log \
        -errormax 15 -update -linedebug -status -define DISPLAY_PD_PU_EN \
        ../src/KISTA_SOI_STDLIB2_ECSM_TT_del.v

#load genus synthesized code for the target block
xmvlog -work work_sub -cdslib ./cds.lib -append_log \
       -logfile xmvlog_postlay.log -errormax 15 -update -linedebug \
       -status ../innovus/output/design_pnr.v

#load system verilog testbench and include a directory in the path
xmvlog -work work -cdslib ./cds.lib -append_log \
    -logfile xmvlog_postlay.log -errormax 15 -update -linedebug \
    -define presyn -status ../src/SPI_Master_TB_pnr.sv -incdir ../src -sv


#elaborate the projet
xmelab -work work -cdslib ./cds.lib -timescale '1ns/1ps' \
    -logfile xmelab_postlay.log -errormax 30 -access +wc \
    -nowarn CUVWSP -nowarn SDFNCAP -status work.SPI_Master_TB


xmsim -cdslib ./cds.lib -logfile xmsim_postlay.log \usepackage{} -errormax 15
```

Run the script:

```
source make.sh
```

To check the waveforms:

```
simvision dump.vcd
```

6 Importing the design in Virtuoso

6.1 Importing Gate level Schematic

Go to the virtuoso directory and start virtuoso.

```
cd virtuoso
virtuoso &
```

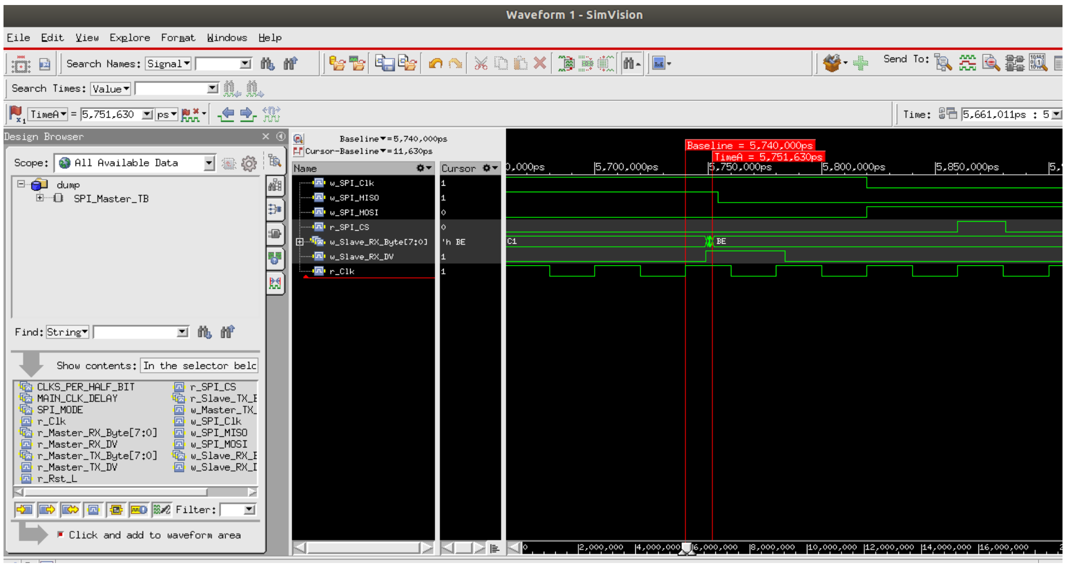


Figure 24: Post-layout simulation

Select Tools -> Library Manager. In the library manager click File -> New -> Library and write SPI_SLAVE in the Name field. Press OK. Then select "Attach to and existing technology library" and select KISTA_SOI_STDLIB_40_4_TECH. Press OK.

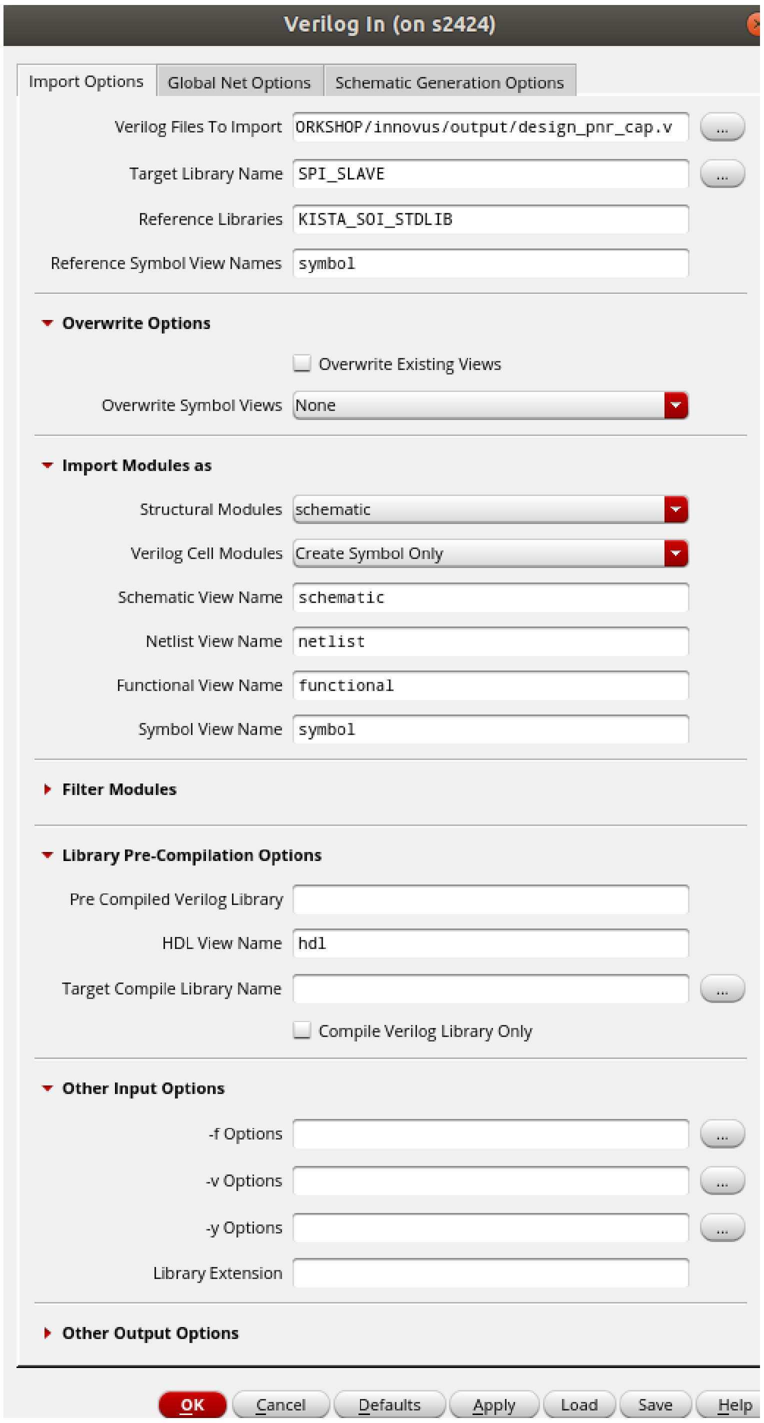
Now go to the CIW window and select File->Import Verilog. In "Verilog Files to Import" select the file place and routed netlist which includes capacitors. It is located under innovus/output/design_pnr_cap.v Change the target library to SPI_SLAVE. Add the reference libraries used in your design separated by a space (in this case it was only KISTA_SOI_STDLIB2). The other options do not need to be changed. The form should look like Fig. 25

Open the schematic, and inspect its content (Fig. 26). It includes gates and decoupling capacitors:

6.2 Importing Layout

To import the layout go to the CIW->Import-> Stream. In "Stream File" select the gds file created in innovus. Select the target Library SPI_SLAVE. Also provide the layer map file for importing the gds. this one is available under pvs/kista_1u.layermap.

Then go to More Options. Here you need to add a Ref Lib File Name. This is just a text file with a list of the names of the reference libraries. To create this file click on the pencil/paper besides "Ref Lib File Name" and select the KISTA_SOI_STDLIB2 and click on "Save and Exit" (Fig. 27). Give it a name myref.reflib.



Verilog In (on s2424)

Import Options | Global Net Options | Schematic Generation Options

Verilog Files To Import	ORKSHOP/innovus/output/design_pnr_cap.v
Target Library Name	SPI_SLAVE
Reference Libraries	KISTA_SOI_STDLIB
Reference Symbol View Names	symbol

Overwrite Options

	☐ Overwrite Existing Views
Overwrite Symbol Views	None

Import Modules as

Structural Modules	schematic
Verilog Cell Modules	Create Symbol Only
Schematic View Name	schematic
Netlist View Name	netlist
Functional View Name	functional
Symbol View Name	symbol

Filter Modules

Library Pre-Compilation Options

Pre Compiled Verilog Library	
HDL View Name	hdl
Target Compile Library Name	
	☐ Compile Verilog Library Only

Other Input Options

-f Options	
-v Options	
-y Options	
Library Extension	

Other Output Options

Figure 25: Import Verilog form

Finally check the "Replace [] with <>" to match bus pins in schematic. The form "More Options" should look like Fig. 28.

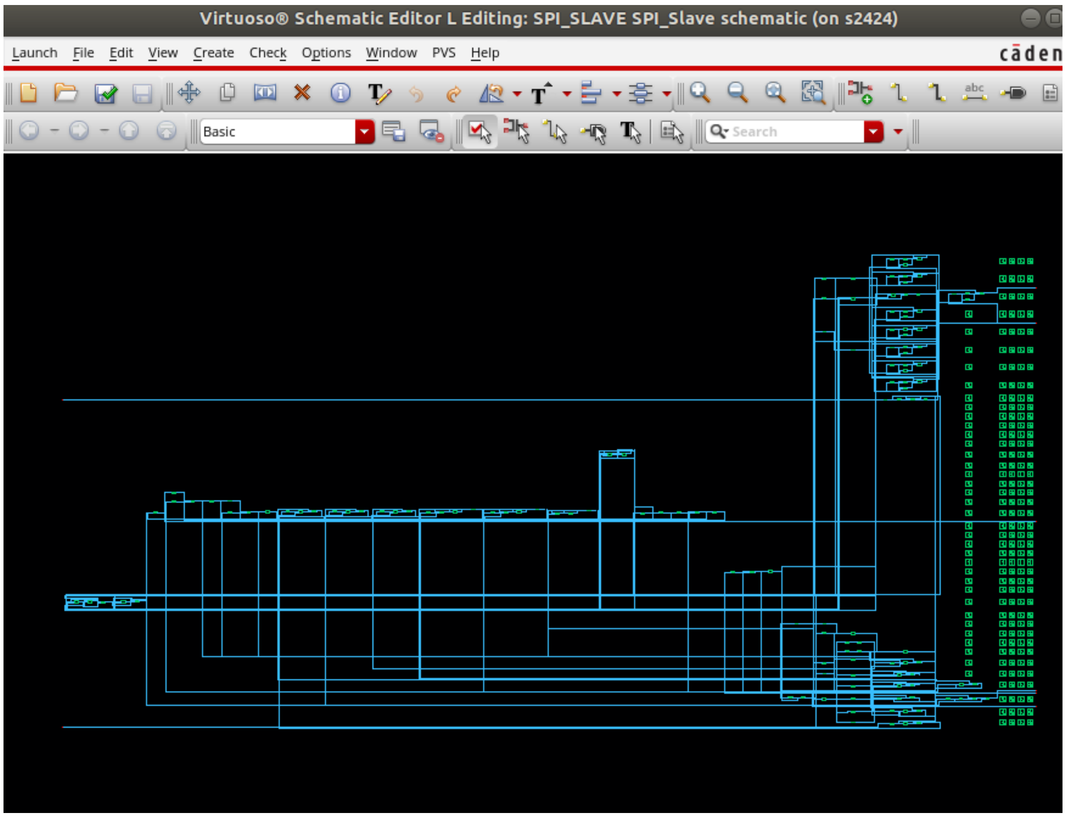

Virtuoso® Schematic Editor L Editing: SPI_SLAVE SPI_Slave schematic (on s2424)
Launch File Edit View Create Check Options Window PVS Help
cāden
Basic
Search

Figure 26: Imported schematic

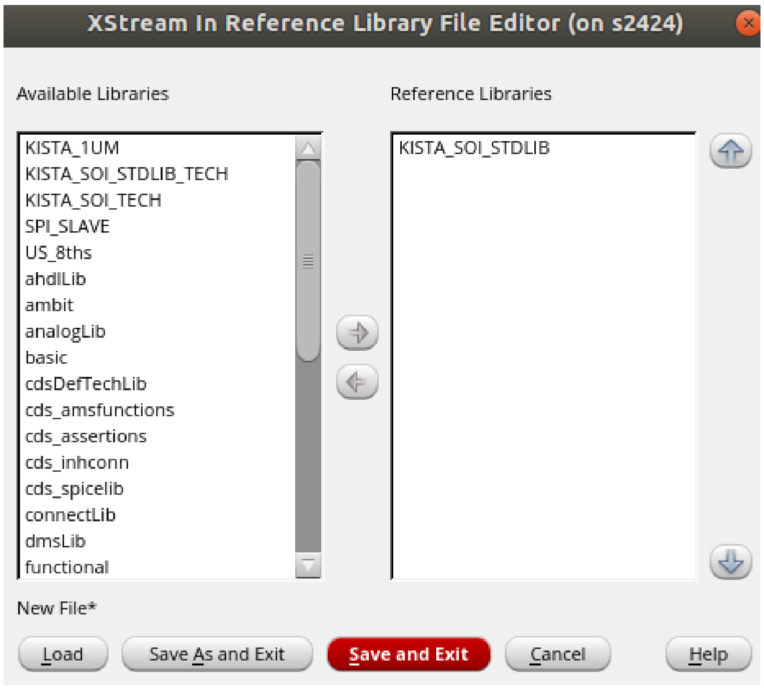
The form for Xtream In should look like: (Fig. 29). Press Translate. A layout view will be created. Open the layout view, and press shift+f to see all the hierarchies (Fig. 40).

Now schematic, layout, and symbol views will be avaliable for the SPI_Slave cell in the SPI_SLAVE library.

Go to PVS -> Run DRC. In "Run Directory" select a directory intended for drc processing. Create a "drc_run" directory if you don't have one. Do not leave this field empty since many intermediate files from the drc will populate the virtuoso directory! The configuration for Rules and Input look like Fig. 31 and Fig. 32. Press submit and check for DRC violations.

Go to PVS -> Run LVS. In "Run Directory" select a directory intended for lvs processing. Create a "lvs_run" directory if you don't have one. Do not leave this field empty since many intermediate files from the lvs will populate the virtuoso directory! The configuration for Rules and Input look like Fig. 33 and Fig. 34. Press submit and check for LVS mismatches.

The design should not have any DRC and LVS since they were checked in innovus; however, it is necessary to cross-check with PVS.


XStream In Reference Library File Editor (on s2424)

Available Libraries

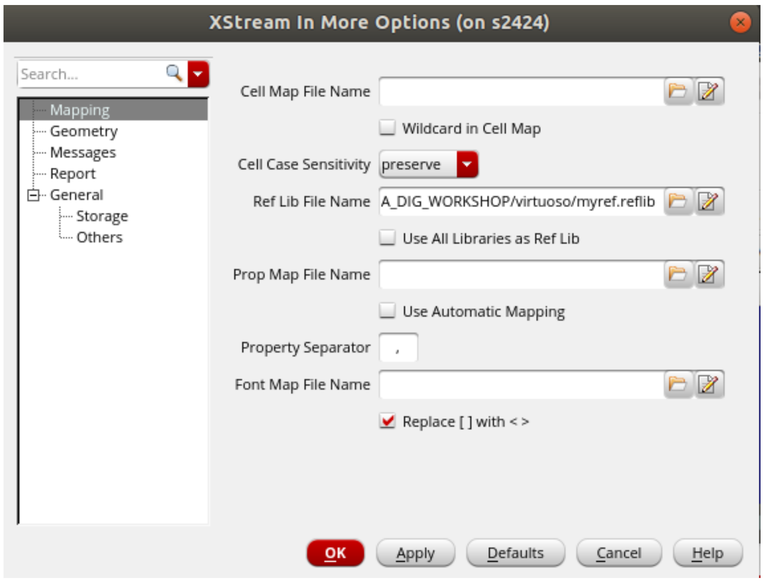
KISTA_1UM
KISTA_SOI_STDLIB_TECH
KISTA_SOI_TECH
SPI_SLAVE
US_8ths
ahdlLib
ambit
analogLib
basic
cdsDefTechLib
cds_amsfunctions
cds_assertions
cds_inhconn
cds_spicelib
connectLib
dmsLib
functional

Reference Libraries

KISTA_SOI_STDLIB

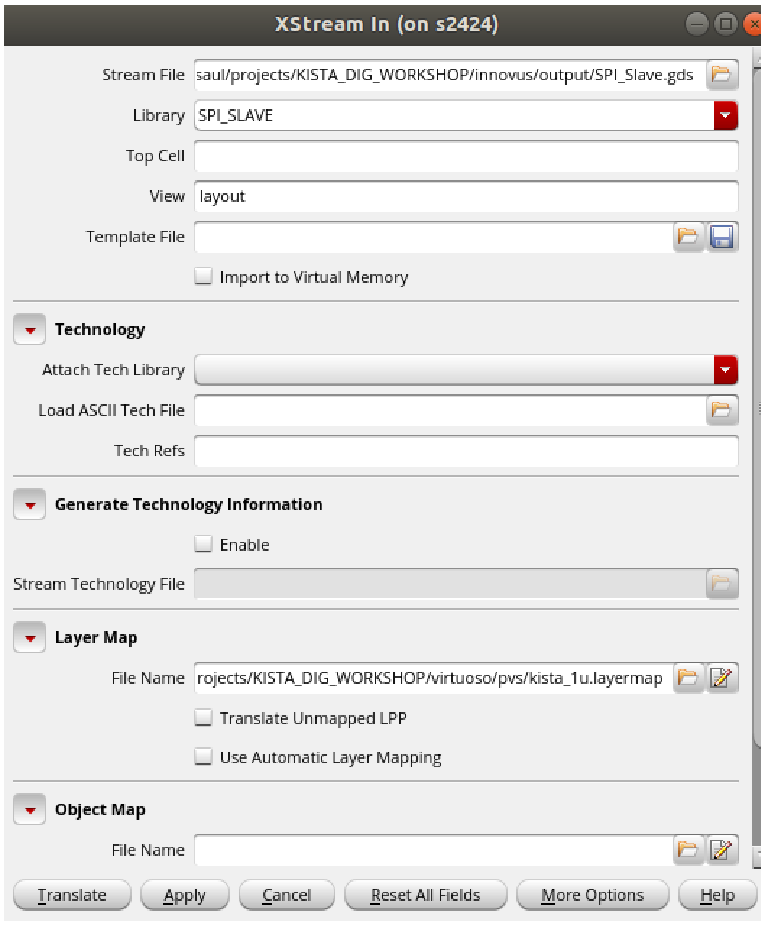
New File*

Figure 27: Creation of reference library file


XStream In More Options (on s2424)

Cell Map File Name	
Wildcard in Cell Map	☐
Cell Case Sensitivity	preserve
Ref Lib File Name	A_DIG_WORKSHOP/virtuoso/myref.reflib
Use All Libraries as Ref Lib	☐
Prop Map File Name	
Use Automatic Mapping	☐
Property Separator	,
Font Map File Name	
Replace [ ] with < >	✓

Figure 28: Form More Options


XStream In (on s2424)

Stream File: saul/projects/KISTA_DIG_WORKSHOP/innovus/output/SPI_Slave.gds

Library: SPI_SLAVE

Top Cell:

View: layout

Template File:

☐ Import to Virtual Memory

Technology

Attach Tech Library:

Load ASCII Tech File:

Tech Refs:

Generate Technology Information

☐ Enable

Stream Technology File:

Layer Map

File Name: rojects/KISTA_DIG_WORKSHOP/virtuoso/pvs/kista_1u.layermap

☐ Translate Unmapped LPP

☐ Use Automatic Layer Mapping

Object Map

File Name:

Figure 29: Creation of reference library file

6.3 Creating Functional View

For completeness, you can create a functional view by importing the RTL Verilog code. This will make possible to simulate the cell in a mixed signal environment. Go to the CIW and select File -> Import -> Verilog and import the original RTL Verilog code for the SPI Slave. Select SPI_SLAVE as Target Library and leave the other options as default (Fig. 36). Press OK, and check that the functional view is available in the Library Manager.

7 AMS Simulation in Virtuoso

Digital blocks imported in virtuoso will almost always be connected to analog and digital blocks. The resulting schematics need to be verified at different levels of abstraction. This section explain how to simulate the imported digital

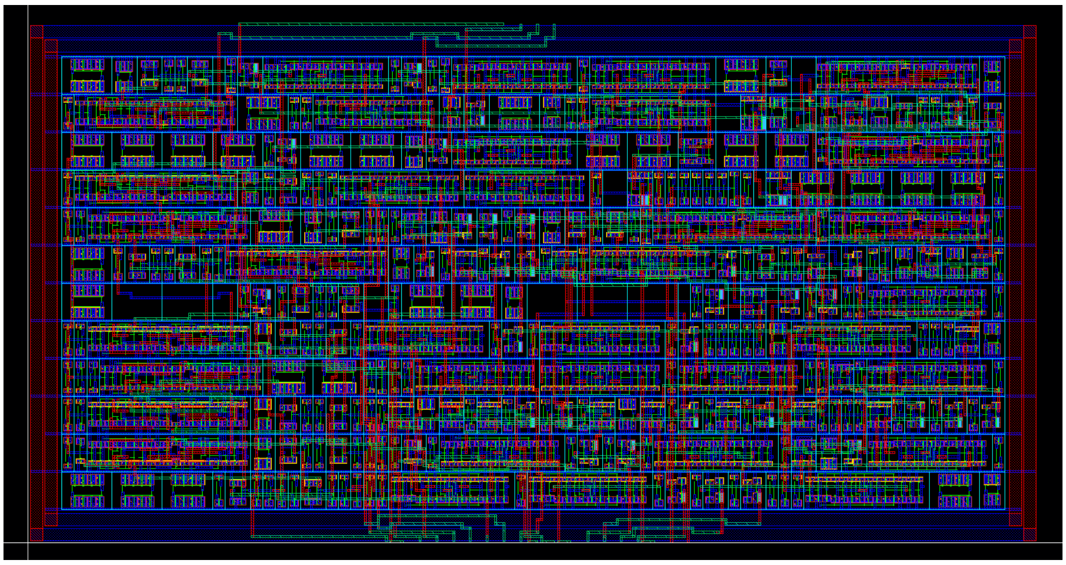
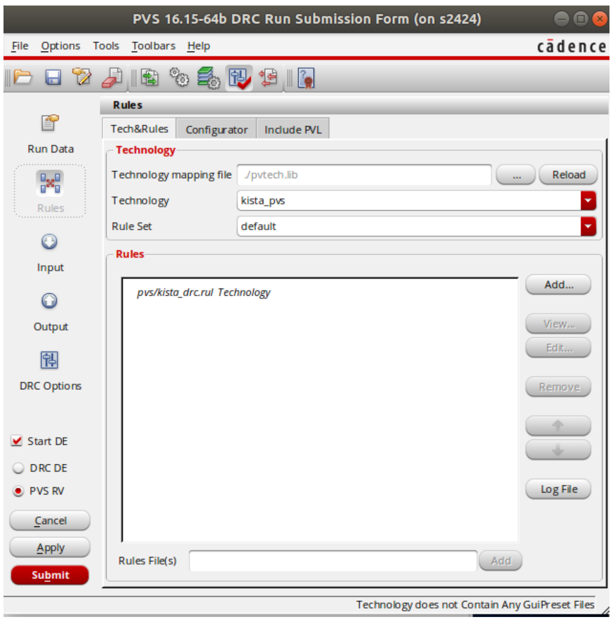


Figure 30: Imported Layout



PVS 16.15-64b DRC Run Submission Form (on s2424)

Rules

Tech&Rules | Configurator | Include PVL

Technology

Technology mapping file	./pvtech.lib
Technology	kista_pvs
Rule Set	default

Rules

pvs/kista_drc.rul Technology

Rules File(s)

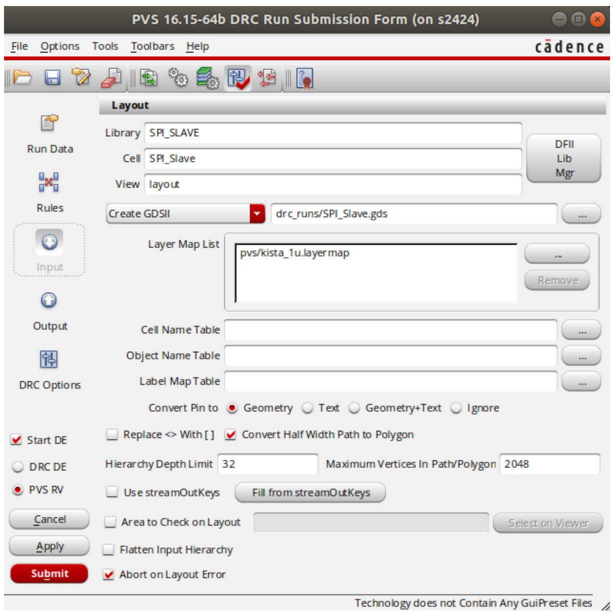
[✓ Start DE]

DRC DE [✓ PVS RV]

Technology does not Contain Any GuiPreset Files

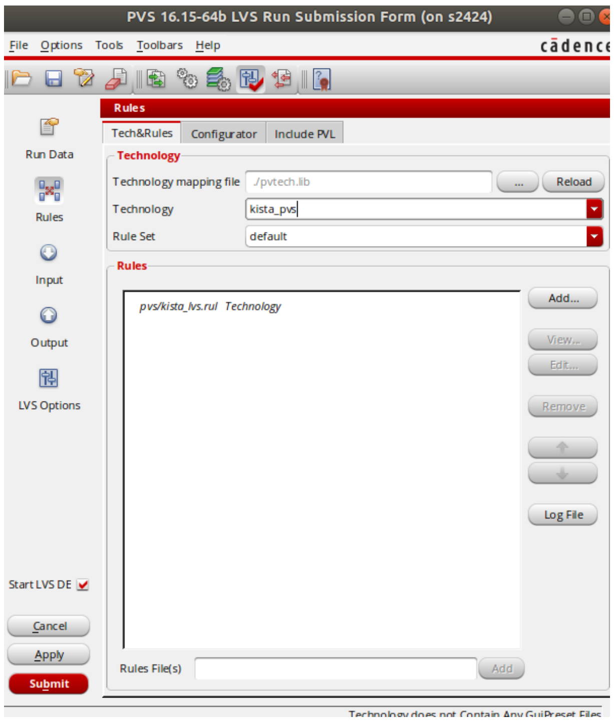
Figure 31: PVS DRC Rules form

block at functional, gate level, parasitic-extracted delay-annotated level.



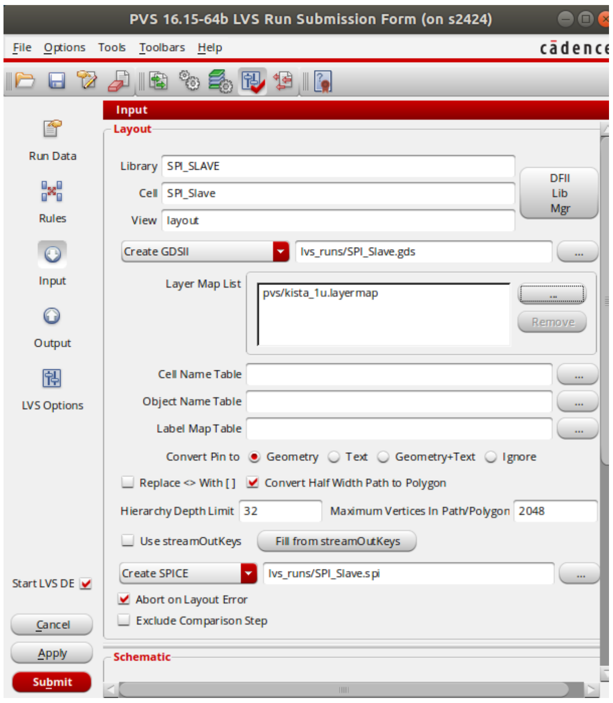
Library SPI_SLAVE
Cell SPI_Slave
View layout
Create GDSII drc_runs/SPI_Slave.gds
Layer Map List pvs/kista_1u.layermap
Cell Name Table
Object Name Table
Label Map Table
Convert Pin to [✓ Geometry] Text Geometry+Text Ignore
[✓ Start DE]
DRC DE [✓ PVS RV]
Replace <> With [] [✓ Convert Half Width Path to Polygon]
Hierarchy Depth Limit 32 Maximum Vertices In Path/Polygon 2048
Use streamOutKeys
Area to Check on Layout
Flatten Input Hierarchy
[✓ Abort on Layout Error]
Technology does not Contain Any GuiPreset Files

Figure 32: PVS DRC Input form



Technology
Technology mapping file ./pvtech.lib
Technology kista_pvs
Rule Set default
Rules
pvs/kista_lvs.rul Technology
Rules File(s)
[✓ Start LVS DE]
Technology does not Contain Any GuiPreset Files

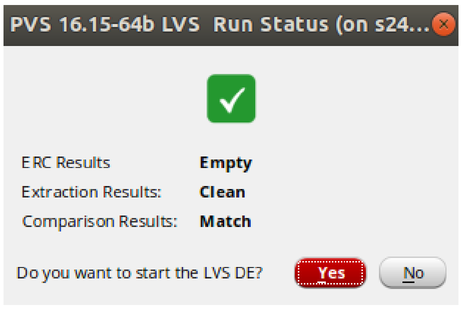
Figure 33: PVS LVS Rules form


PVS 16.15-64b LVS Run Submission Form (on s2424)

Input — Layout

Library	SPI_SLAVE
Cell	SPI_Slave
View	layout
Create GDSII	lvs_runs/SPI_Slave.gds
Layer Map List	pvs/kista_1u.layermap
Cell Name Table	
Object Name Table	
Label Map Table	
Convert Pin to	[✓ Geometry] Text Geometry+Text Ignore
Replace <> With []	☐
Convert Half Width Path to Polygon	✓
Hierarchy Depth Limit	32
Maximum Vertices In Path/Polygon	2048
Use streamOutKeys	☐
Create SPICE	lvs_runs/SPI_Slave.spi
Abort on Layout Error	✓
Exclude Comparison Step	☐
Start LVS DE	✓

Figure 34: PVS LVS Input form


PVS 16.15-64b LVS Run Status (on s24...

ERC Results	**Empty**
Extraction Results:	**Clean**
Comparison Results:	**Match**

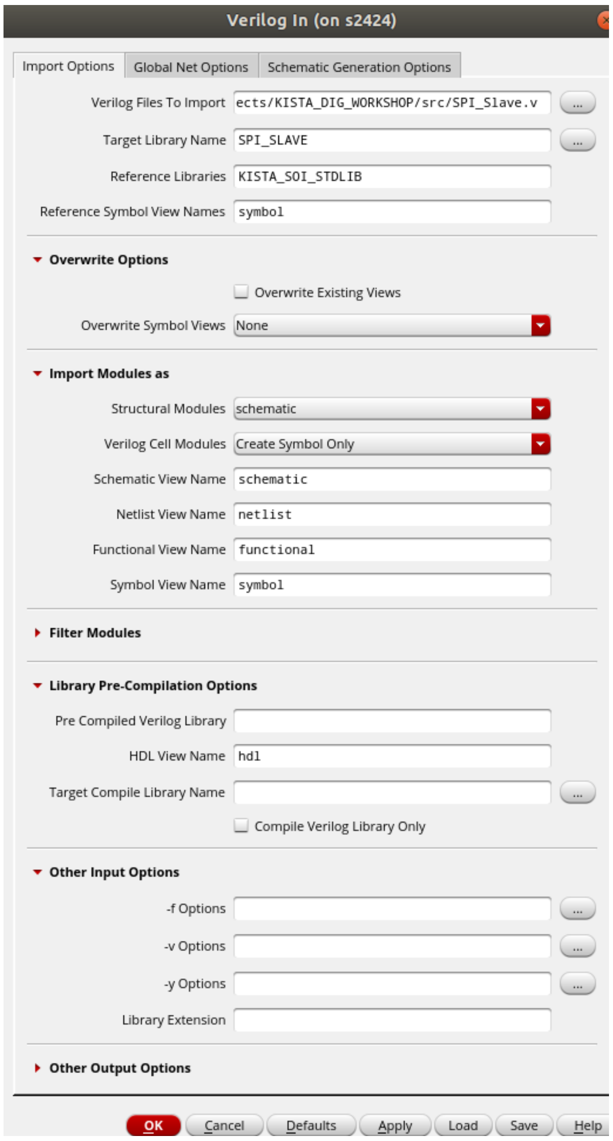
Do you want to start the LVS DE?

Figure 35: LVS results

7.1 Test bench preparation

First, create a library SPI_SLAVE_STIMULUS and import a modified version of the verilog testbench as a new digital block called SPI_stimulus. Like in the RTL, gate level, and post-layout simulation, this block will provide all the stimulus to drive the SPI_Slave and also will read its outputs. Accordingly Modifications from the original testbench should in principle only include the addition of input and output ports to the registers and wires that connect to the instantiated SPI_Slave.

Next, create a test bench schematic SPI_stimulus_TB and instantiate a



Verilog Files To Import: ects/KISTA_DIG_WORKSHOP/src/SPI_Slave.v
Target Library Name: SPI_SLAVE
Reference Libraries: KISTA_SOI_STDLIB
Reference Symbol View Names: symbol

Overwrite Options
☐ Overwrite Existing Views
Overwrite Symbol Views: None

Import Modules as
Structural Modules: schematic
Verilog Cell Modules: Create Symbol Only
Schematic View Name: schematic
Netlist View Name: netlist
Functional View Name: functional
Symbol View Name: symbol

Filter Modules

Library Pre-Compilation Options
Pre Compiled Verilog Library:
HDL View Name: hdl
Target Compile Library Name:
☐ Compile Verilog Library Only

Other Input Options
-f Options:
-v Options:
-y Options:
Library Extension:

Other Output Options

Figure 36: Importing RTL Verilog

SPI_Slave block (Fig. 37).

An important detail for AMS simulation is that there most exist both analog and digital circuits in the netlist. Add at least one analog component to the schematic (a resistor connected to ground, for instance).

In addition, it is needed to add DC power supplies for VDD and VSS in the digital circuit. In typical applications, analog and digital power supplies need to be separated, or different digital blocks need to be powered at different voltages. Therefore, it is very common to require different names for the global nets VDD! and VSS! used internally in each digital block. This is easily

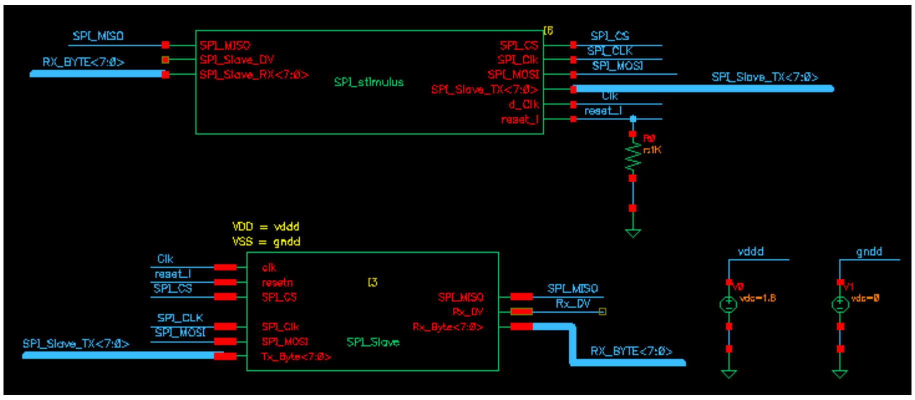
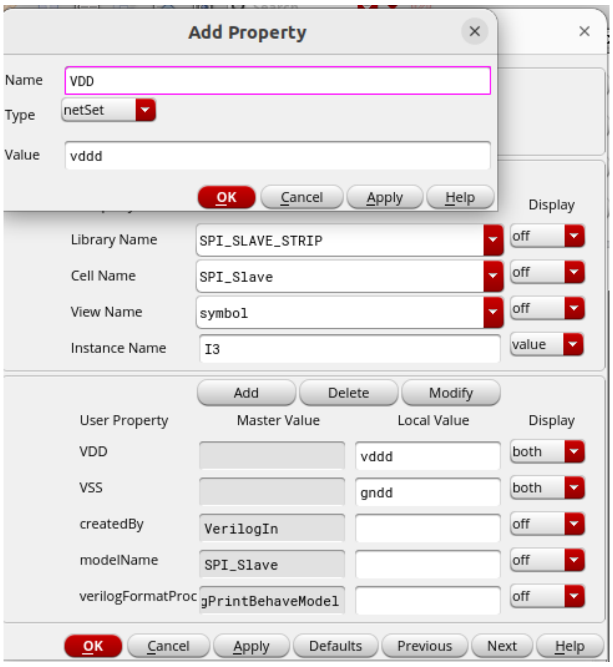


Figure 37: Test bench including stimulus block



Add Property

Name	VDD
Type	netSet
Value	vddd

OK Cancel Apply Help

		Display
Library Name	SPI_SLAVE_STRIP	off
Cell Name	SPI_Slave	off
View Name	symbol	off
Instance Name	I3	value

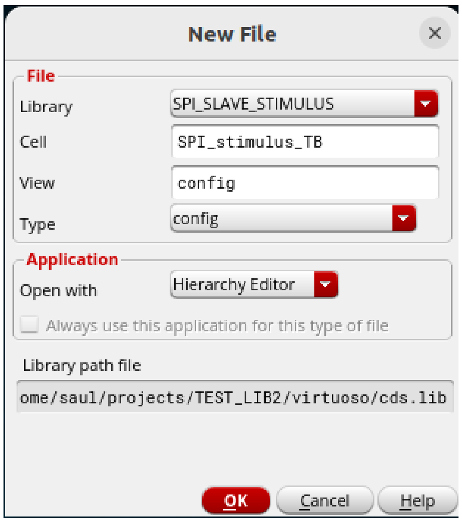
Add Delete Modify

User Property	Master Value	Local Value	Display
VDD		vddd	both
VSS		gndd	both
createdBy	VerilogIn		off
modelName	SPI_Slave		off
verilogFormatProc	gPrintBehaveModel		off

OK Cancel Apply Defaults Previous Next Help

Figure 38: Changing VDD and VSS names using netSet

done by clicking on the digital block and opening its properties (press q). In the pop-up window click on Add and provide the name of the net to change which in this case is VDD or VSS, in the Type combo select netSet, and in Value type the new name of the net in the schematic that is connected to power and ground (Fig. 38). In this example the names of labels are vddd and gndd (the extra d for digital). Note that a DC voltage source of 0V is used for the ground. This is necessary because it is not allowed to label a net connected to the ground symbol.


New File

File

Library: SPI_SLAVE_STIMULUS

Cell: SPI_stimulus_TB

View: config

Type: config

Application

Open with: Hierarchy Editor

Always use this application for this type of file

Library path file: ome/saul/projects/TEST_LIB2/virtuoso/cds.lib

Figure 39: Creation of a config view

7.2 Functional simulation

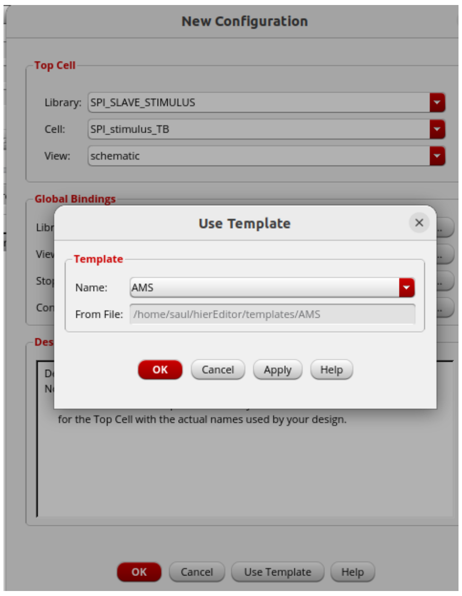
Open the library manager, select the SPI_stimulus_TB and then go to File->New->Cell View. Then in the pop-up dialog select config and click OK (Fig. 39).

A new configuration window will pop-up. In the View field select schematic, and then click on Use Template and select AMS (Fig. 40). Finally a window with the new config view will appear. You need to close this window and open it again by double clicking it in the library manager. Then a new window will appear (Fig. 41). Make sure that both config and schematic options are set to Yes. When you click OK, both config and schematic views will open.

In the config view change the View List field to “functional spectre symbol” and recompile the hierarchy by clicking on the button showing an arrow pointing up in the toolbar

Now in the schematic go to the menu and click on Launch->ADE XL and create a new view. Then in the Data View section double click on Tests and select “Click to Add test”. In the pop-up dialog that appears make sure that the cell name is SPI_stimulus_TB and that the View Name is config Fig. 43.

The ADE XL Test Editor window appears. In this window we will configure the simulation options. First go to Setup -> Simulator and select “ams”. Then go to Setup -> High-performance/Parasitic reduction and select APS. In APS Options check the option Use ++aps. Next, go to Setup -> Model Libraries and add the spectre models which are available at models/spectre/kth/KTH_P20_FET_v1.scs (Fig. 44). Now go to Setup -> Connect Rules/IE Setup and make sure that Connect Rule/Connect Module Based

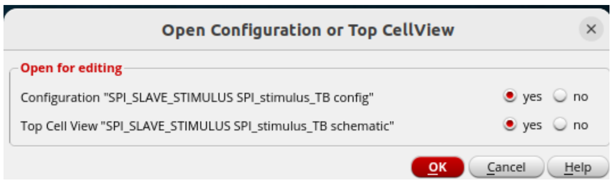


Library: SPI_SLAVE_STIMULUS
Cell: SPI_stimulus_TB
View: schematic

Name: AMS
From File: /home/saul/hierEditor/templates/AMS

for the Top Cell with the actual names used by your design.

Figure 40: Config view - Template

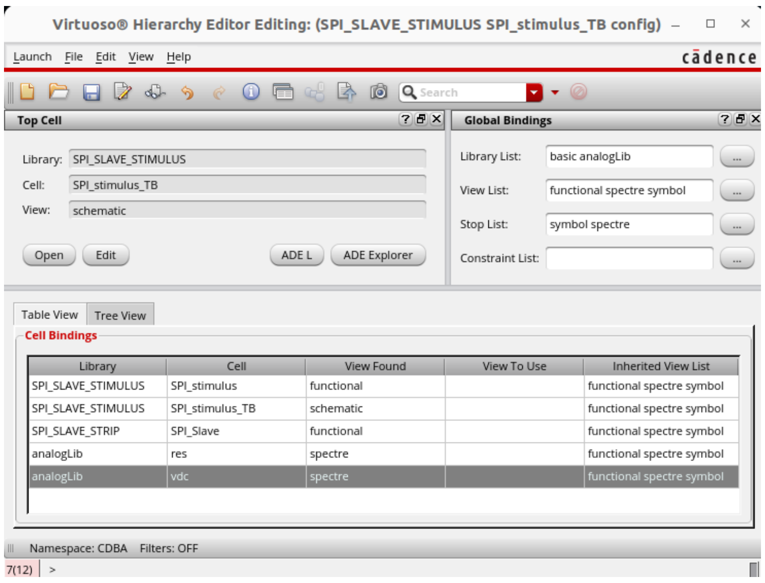


Configuration "SPI_SLAVE_STIMULUS SPI_stimulus_TB config" [✓ yes] no
Top Cell View "SPI_SLAVE_STIMULUS SPI_stimulus_TB schematic" [✓ yes] no

Figure 41: Config view - opening both schematic and config

Setup is selected. In the drop-down menu select connectLib.ConnRules_18V_full_fast (Fig. 45). Finally go to Simulation -> Options -> AMS Simulator. In the Main tab click on "Include Option Settings" and add the library KISTA_SOI_STDLIB2_ECSM_TT_del.v (Fig. 46). Now go to Outputs -> To Be Plotted -> Select on Design and select all the digital wires and buses. Buses should be added as bundles. Finally go to Analyses -> Choose and select tran, write 60u in Stop Time, and select "liberal" in Accuracy Defaults. After clicking OK, the ADE XL Test Editor should look like Fig. 47. Close the window and come back to the ADE XL window which should look like Fig. 48.

Click the run button (Play button, green color). The simulator will start by generating the netlist and then running the transient. As soos as the transient is finished, the results will be available. Check the log as it contains imporant information. In partiular, the stimulus block has some verilog code that checks the input/output signals of the SPI_slave block and prints messages in the log telling if the transmitted/received bytes match (OK) or if there are ERRORs. When doing verification this method is typically more useful than looking into



Virtuoso® Hierarchy Editor Editing: (SPI_SLAVE_STIMULUS SPI_stimulus_TB config)

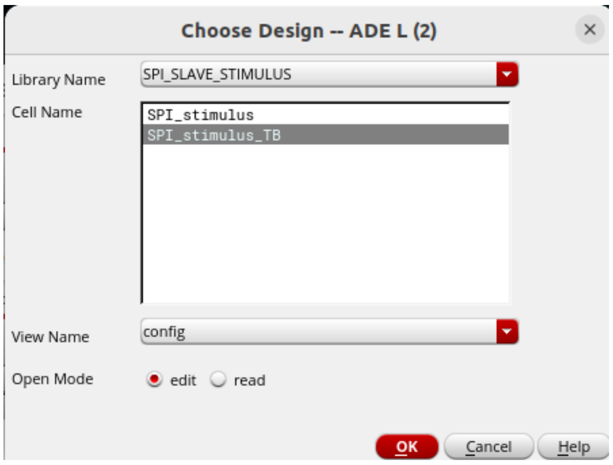
Library: SPI_SLAVE_STIMULUS
Cell: SPI_stimulus_TB
View: schematic

Library List: basic analogLib
View List: functional spectre symbol
Stop List: symbol spectre
Constraint List:

Library	Cell	View Found	View To Use	Inherited View List
SPI_SLAVE_STIMULUS	SPI_stimulus	functional		functional spectre symbol
SPI_SLAVE_STIMULUS	SPI_stimulus_TB	schematic		functional spectre symbol
SPI_SLAVE_STRIP	SPI_Slave	functional		functional spectre symbol
analogLib	res	spectre		functional spectre symbol
analogLib	vdc	spectre		functional spectre symbol

Namespace: CDBA Filters: OFF
7(12) >

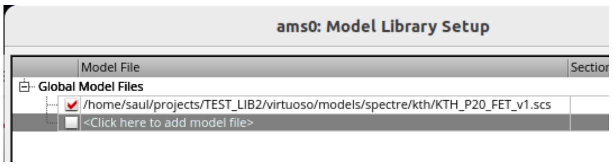
Figure 42: functional view configuration



Choose Design -- ADE L (2)

Library Name SPI_SLAVE_STIMULUS
Cell Name
SPI_stimulus
SPI_stimulus_TB
View Name config
Open Mode [✓ edit] read

Figure 43: Creation of the test in ADE XL

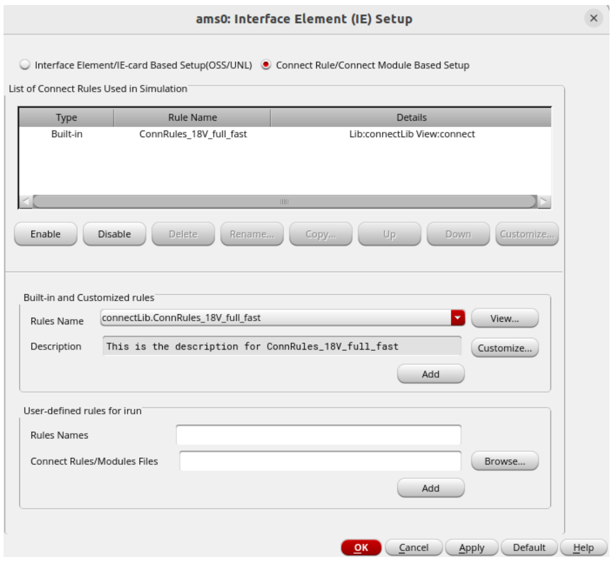


ams0: Model Library Setup

Model File	Section
Global Model Files	
✓ /home/saul/projects/TEST_LIB2/virtuoso/models/spectre/kth/KTH_P20_FET_v1.scs	
<Click here to add model file>	

Figure 44: Addition of spectre models

the time plots. When debugging a digital circuit, on the other hand, it is much more useful to see the signals. Click on the “Plot All” button in the Results


ams0: Interface Element (IE) Setup

Interface Element/IE-card Based Setup(OSS/UNL) [✓ Connect Rule/Connect Module Based Setup]

List of Connect Rules Used in Simulation

Type	Rule Name	Details
Built-in	ConnRules_18V_full_fast	Lib:connectLib View:connect

Built-in and Customized rules

Rules Name: connectLib.ConnRules_18V_full_fast

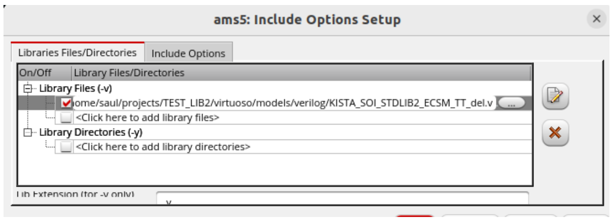
Description: `This is the description for ConnRules_18V_full_fast`

User-defined rules for irun

Rules Names:

Connect Rules/Modules Files:

Figure 45: Configuration of Connect Rules


ams5: Include Options Setup

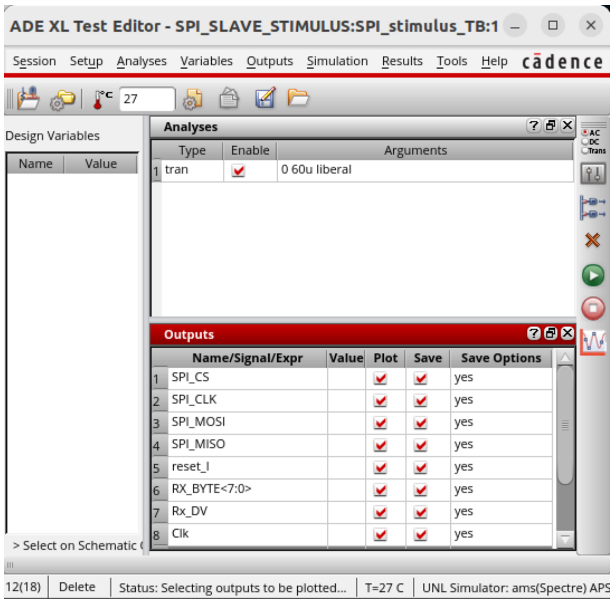
Libraries Files/Directories | Include Options

On/Off	Library Files/Directories
	Library Files (-v)
✓	ome/saul/projects/TEST_LIB2/virtuoso/models/verilog/KISTA_SOI_STDLIB2_ECSM_TT_del.v
	<Click here to add library files>
	Library Directories (-y)
	<Click here to add library directories>

Lib Extension (for -v only): v

Figure 46: Including standard cell verilog library

tab (a small plot icon in red white and blue) and all the selected signals will be plotted.



ADE XL Test Editor - SPI_SLAVE_STIMULUS:SPI_stimulus_TB:1

Design Variables

Name	Value

Analyses

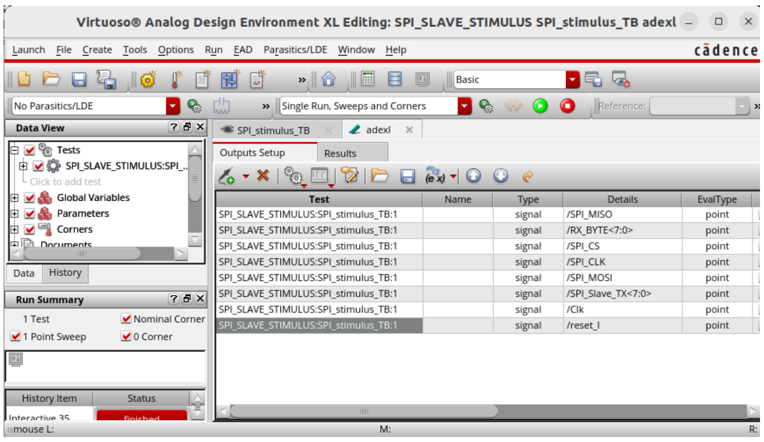
	Type	Enable	Arguments
1	tran	✓	0 60u liberal

Outputs

	Name/Signal/Expr	Value	Plot	Save	Save Options
1	SPI_CS		✓	✓	yes
2	SPI_CLK		✓	✓	yes
3	SPI_MOSI		✓	✓	yes
4	SPI_MISO		✓	✓	yes
5	reset_l		✓	✓	yes
6	RX_BYTE<7:0>		✓	✓	yes
7	Rx_DV		✓	✓	yes
8	Clk		✓	✓	yes

12(18) | Delete | Status: Selecting outputs to be plotted... | T=27 C | UNL Simulator: ams(Spectre) APS

Figure 47: ADE XL Editor configuraton



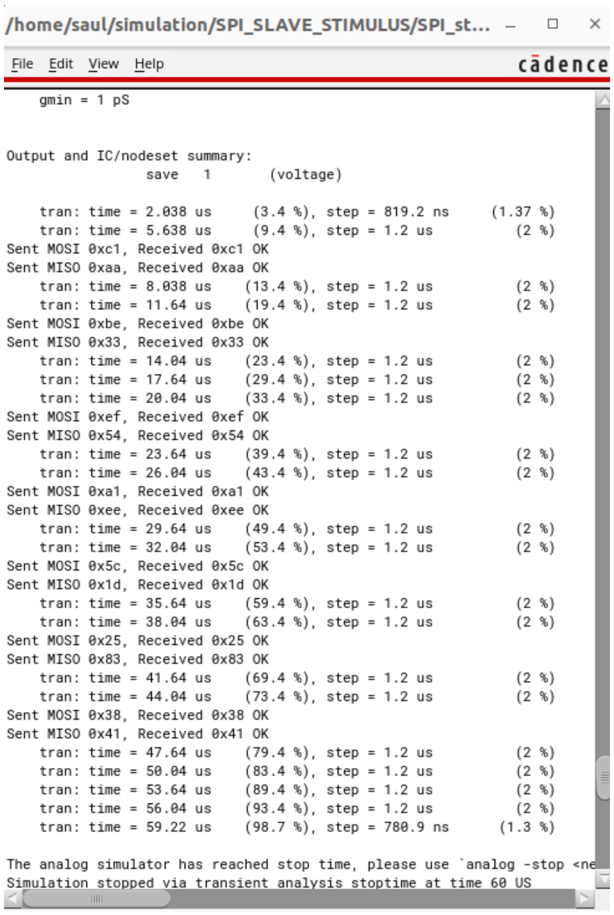
Virtuoso® Analog Design Environment XL Editing: SPI_SLAVE_STIMULUS SPI_stimulus_TB adexl

No Parasitics/LDE | Single Run, Sweeps and Corners | Basic

Run Summary: 1 Test, ✓ Nominal Corner, ✓ 1 Point Sweep, ✓ 0 Corner

Test	Name	Type	Details	EvalType
SPI_SLAVE_STIMULUS:SPI_stimulus_TB:1		signal	/SPI_MISO	point
SPI_SLAVE_STIMULUS:SPI_stimulus_TB:1		signal	/RX_BYTE<7:0>	point
SPI_SLAVE_STIMULUS:SPI_stimulus_TB:1		signal	/SPI_CS	point
SPI_SLAVE_STIMULUS:SPI_stimulus_TB:1		signal	/SPI_CLK	point
SPI_SLAVE_STIMULUS:SPI_stimulus_TB:1		signal	/SPI_MOSI	point
SPI_SLAVE_STIMULUS:SPI_stimulus_TB:1		signal	/SPI_Slave_TX<7:0>	point
SPI_SLAVE_STIMULUS:SPI_stimulus_TB:1		signal	/Clk	point
SPI_SLAVE_STIMULUS:SPI_stimulus_TB:1		signal	/reset_l	point

Figure 48: ADE XL Editor configuraton



```
    gmin = 1 pS


Output and IC/nodeset summary:
                 save   1       (voltage)

    tran: time = 2.038 us     (3.4 %), step = 819.2 ns     (1.37 %)
    tran: time = 5.638 us     (9.4 %), step = 1.2 us          (2 %)
Sent MOSI 0xc1, Received 0xc1 OK
Sent MISO 0xaa, Received 0xaa OK
    tran: time = 8.038 us    (13.4 %), step = 1.2 us          (2 %)
    tran: time = 11.64 us    (19.4 %), step = 1.2 us          (2 %)
Sent MOSI 0xbe, Received 0xbe OK
Sent MISO 0x33, Received 0x33 OK
    tran: time = 14.04 us    (23.4 %), step = 1.2 us          (2 %)
    tran: time = 17.64 us    (29.4 %), step = 1.2 us          (2 %)
    tran: time = 20.04 us    (33.4 %), step = 1.2 us          (2 %)
Sent MOSI 0xef, Received 0xef OK
Sent MISO 0x54, Received 0x54 OK
    tran: time = 23.64 us    (39.4 %), step = 1.2 us          (2 %)
    tran: time = 26.04 us    (43.4 %), step = 1.2 us          (2 %)
Sent MOSI 0xa1, Received 0xa1 OK
Sent MISO 0xee, Received 0xee OK
    tran: time = 29.64 us    (49.4 %), step = 1.2 us          (2 %)
    tran: time = 32.04 us    (53.4 %), step = 1.2 us          (2 %)
Sent MOSI 0x5c, Received 0x5c OK
Sent MISO 0x1d, Received 0x1d OK
    tran: time = 35.64 us    (59.4 %), step = 1.2 us          (2 %)
    tran: time = 38.04 us    (63.4 %), step = 1.2 us          (2 %)
Sent MOSI 0x25, Received 0x25 OK
Sent MISO 0x83, Received 0x83 OK
    tran: time = 41.64 us    (69.4 %), step = 1.2 us          (2 %)
    tran: time = 44.04 us    (73.4 %), step = 1.2 us          (2 %)
Sent MOSI 0x38, Received 0x38 OK
Sent MISO 0x41, Received 0x41 OK
    tran: time = 47.64 us    (79.4 %), step = 1.2 us          (2 %)
    tran: time = 50.04 us    (83.4 %), step = 1.2 us          (2 %)
    tran: time = 53.64 us    (89.4 %), step = 1.2 us          (2 %)
    tran: time = 56.04 us    (93.4 %), step = 1.2 us          (2 %)
    tran: time = 59.22 us    (98.7 %), step = 780.9 ns      (1.3 %)

The analog simulator has reached stop time, please use `analog -stop <ne
Simulation stopped via transient analysis stoptime at time 60 US
```

Figure 49: Simulation log

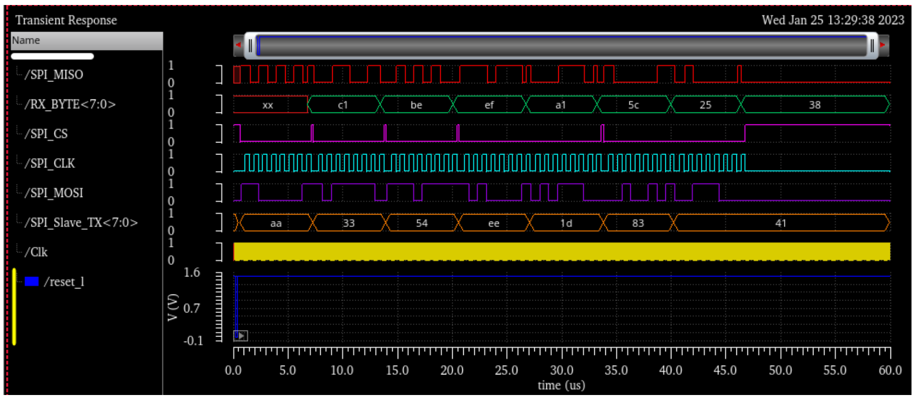


Figure 50: Functional simulation signals